科学计算环境
平台层 API 详细说明

科学计算与系统建模仿真项目组

编制		生效日期	
审核		批准	

文件变更摘要

日期	版本	变更说明	修订	审核	批准

目 录

1. 概述 1

2. 综合案例介绍 2

2.1 Julia 调用平台层 API 2
2.2 C++调用平台层 API 6
2.2.1 开发规范 6
2.2.2 示例代码 6
2.3 Python 调用平台层 API 10
2.3.1 开发规范 10
2.3.2 示例代码 12
2.4 APP 调用平台层 API 14

3. API 详细说明 16

3.1 基础 API 16
3.1.1 输入命令 16
3.1.2 环境和设置 34
3.1.3 数据导入和分析 42
3.2 数学 API 102
3.2.1 初等数学 102
3.2.2 线性代数 428
3.2.3 随机数生成 535
3.2.4 插值 548
3.2.5 数值积分和微分方程 654
3.2.6 傅里叶分析和滤波 729
3.2.7 稀疏数组 776
3.3 图形 API 922
3.3.1 二维和三维图 922
3.3.2 格式和注释 1381
3.3.3 打印和保存 1724
3.3.4 图形对象 1735

3.4 App 构架 API 1753
3.4.1 使用 App 设计工具 1753
3.4.2 以编程方式开发 App 1753
3.4.3 App 管理 1753
3.4.4 App 通信 1759

1. 概述

科学计算环境平台层 API，支持对平台的界面、业务逻辑、数据等不同层次接口调用，支持 App 的扩展开发和集成。按功能划分为基础 API、数学 API、图形 API、App 构建 API。平台层 API 组件视图如下图所示。

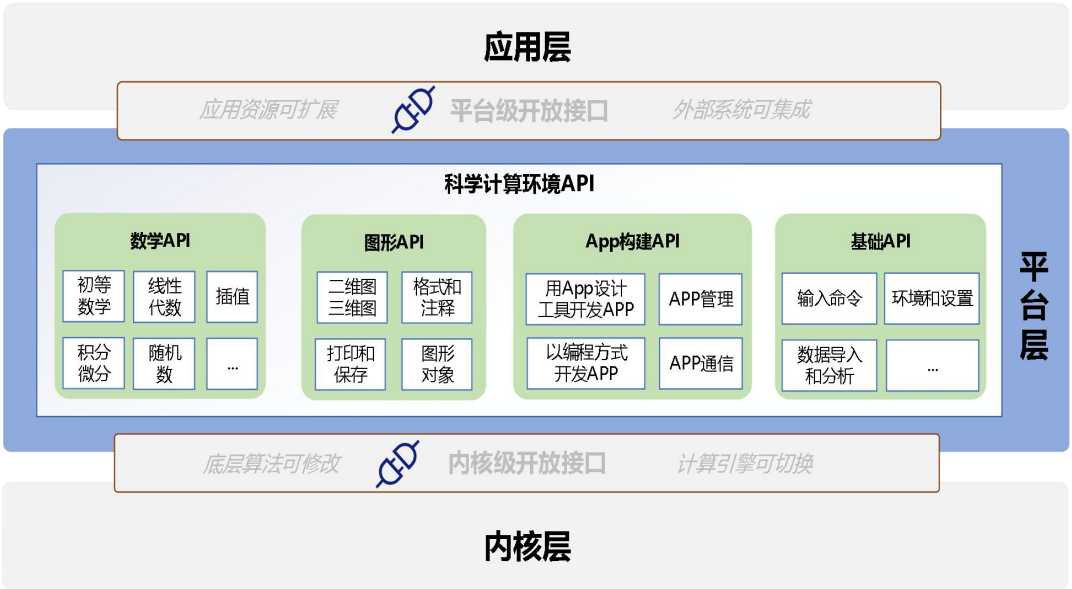


图 1- 1 科学计算环境 API

- 基础 API：提供了科学计算最基础的功能，包括命令行控制，科学计算语言基础、平台环境和设置操作、 数据导入导出和分析的功能、以及外部语言接入和调用的功能。
- 数学 API：提供科学计算核心的专业数学计算函数。
- 图形 API：提供可视化绘图的功能。
- App 构建 API 提供 App 开发、打包、部署、运行相关的功能。

科学计算环境提供的 API，其中基础 API、数学 API 和图形 API 都是 Julia 语言的函数，用户可以在自己的编程环境中，通过调用 Julia 脚本的形式来使用这些 Julia 函数。App 构建 API 中，App 管理 API 是 Syslab 提供的一套 App 管理的函数，可以在 Syslab REPL 环境中实现对 App 的安装、启动、卸载等管理操作；App 通信 API 支持用户在多种图形应用环境中开发的 APP 与 Syslab 平台通信互操作。

2. 综合案例介绍

2.1 Julia 调用平台层 API

本案例使用 FFT 分析周期性数据，通过 Julia 代码调用平台层 API 函数。该案例代码在科学计算环境中可直接运行，案例文件路径为：<Syslab 安装路径>\Examples\04 数学\07 傅里叶分析与滤波\01 使用 FFT 分析周期性数据.jl。

可以使用傅里叶变换来分析数据中的变化，例如一个时间段内的自然事件。下面的代码是天文学家使用苏黎世太阳黑子相对数对大约 300 年的太阳黑子的数量和大小进行统计，定对大约 1700 至 2000 年间的苏黎世数绘图。

```
include("01 sunspot.jl")
year = sunspot[:, 1]
relNums = sunspot[:, 2]
figure(1)
plot(year, relNums)
xlabel("Year")
ylabel("Zurich Number")
title("Sunspot Data")
```

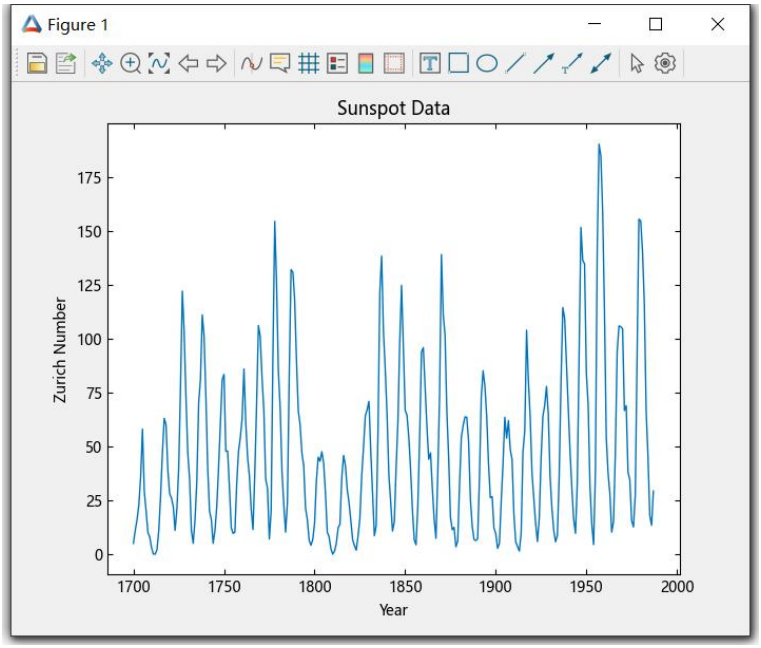


图 2- 1 300 年间苏黎世数绘图

为了更详细地看太阳黑子活动的周期特性，将对前 50 年的数据绘图。

```
figure(2)
plot(year[1:50], relNums[1:50], "b.-", markerfacecolor = "auto",
```

```
markeredgecolor = "none")
xlabel("Year")
ylabel("Zurich Number")
title("Sunspot Data")
```

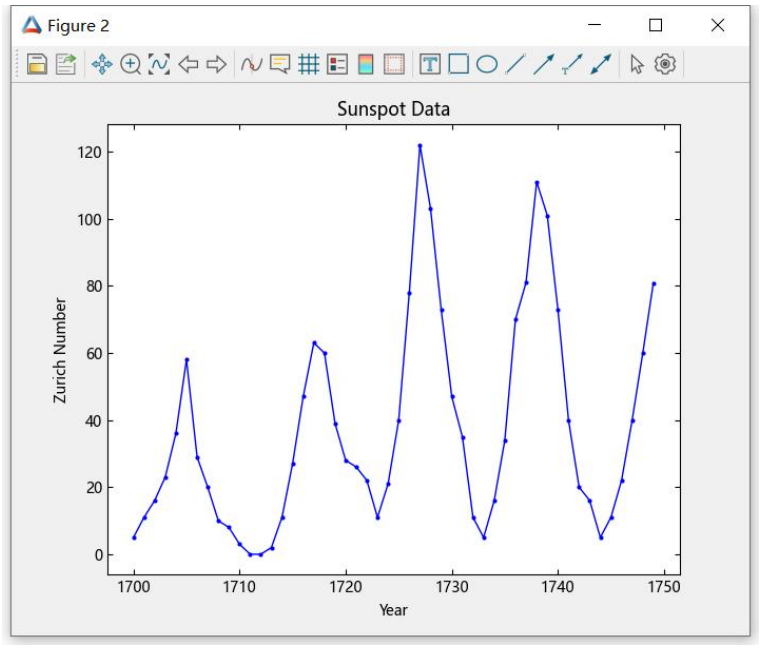


图 2- 2 前 50 年间苏黎世数绘图

傅里叶变换是一种基础的信号处理工具，可确定数据中的频率分量。使用 fft 函数获取苏黎世数据的傅里叶变换。删除存储数据总和的输出的第一个元素。绘制该输出的其余部分，其中包含复傅里叶系数关于实轴的镜像图像。

```
y = fft(relNums)
y = y[2:end]
figure(3)
plot(real(y), imag(y), "ro")
ylabel("imag(y)")
title("Fourier Coefficients")
```

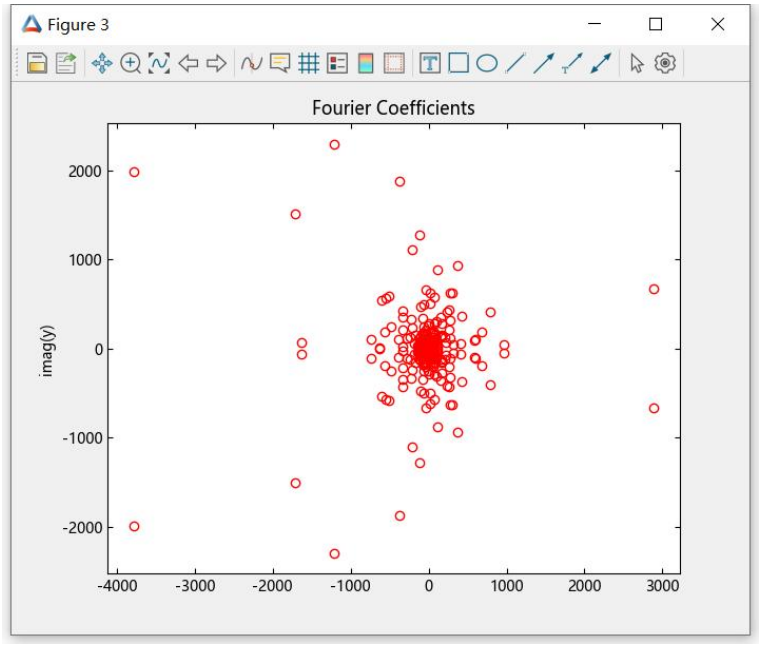


图 2- 3 傅里叶系数关于实轴的镜像图

单独的傅里叶系数难以解释。计算系数更有意义的方法是计算其平方幅值，即计算幂。由于一半的系数在幅值中是重复的，因此您只需要对一半的系数计算幂。以频率函数的形式绘制功率谱图，以每年的周期数为测量单位。

```
n = length(y)
power = abs.(y[1:Int(floor(n / 2))]) .^ 2  # power of first half of
transform data
maxfreq = 1 / 2                  # maximum frequency
freq = [1:n/2...] / (n / 2) * maxfreq     # equally spaced frequency
grid
figure(4)
plot(freq, power)
xlabel("Cycles/Year")
ylabel("Power")
```

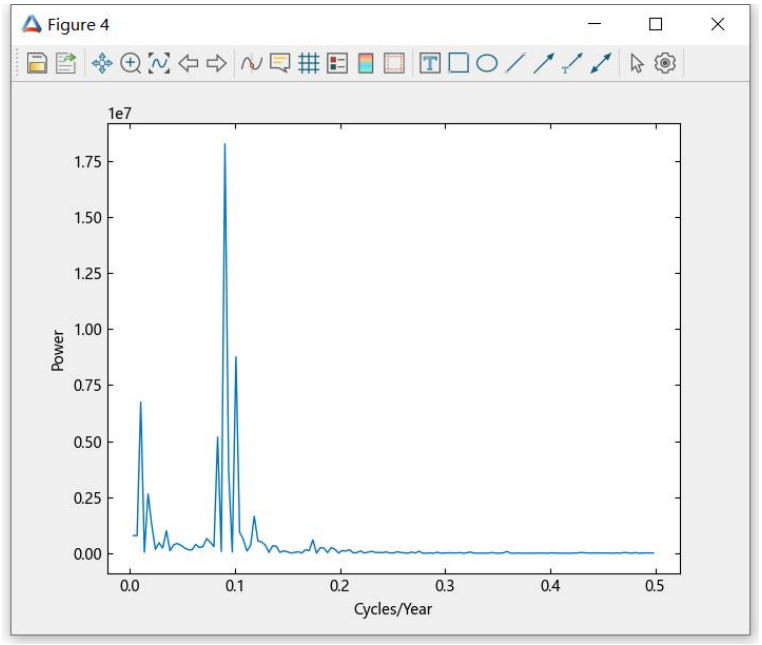


图 2- 4 计算幂图

太阳黑子活动发生的最大频率低于每年一次。为了查看更易解释的周期活动，以周期函数形式绘制幂图，以每周期的年数为测量单位。该绘图揭示了太阳黑子活动约每 11 年出现一次高峰。

```
period = 1 ./ freq
figure(5)
plot(period, power)
xlim([0 50]); #zoom in on max power
xlabel("Years/Cycle")
ylabel("Power")
```

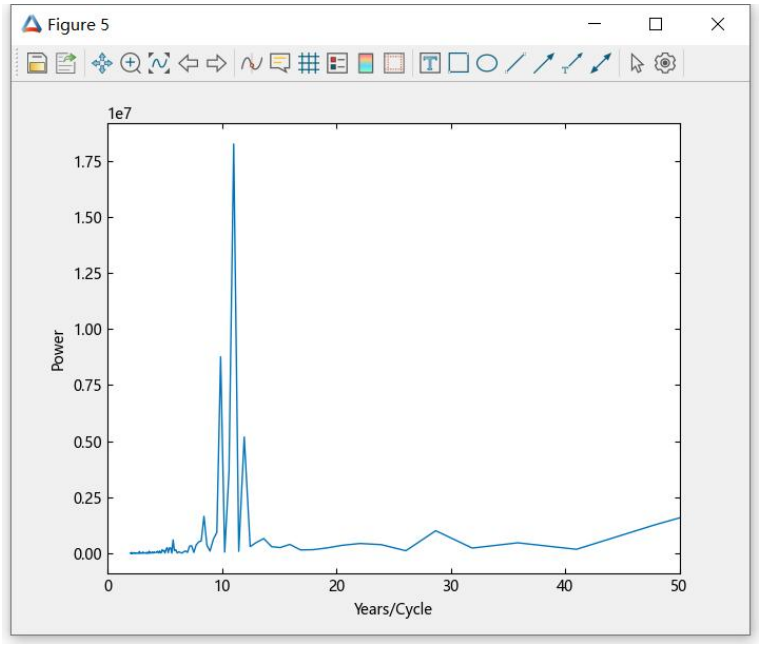


图 2- 5 周期函数形式绘制幂图

上述算法代码中使用了平台层 API 中以下函数。

序号	函数名	简介
1	fft	用快速傅里叶变换 (FFT) 算法计算 X 的离散傅里叶变换 (DFT)
2	figure	创建图窗窗口
3	plot	二维线图
4	xlabel	为 x 轴添加标签
5	ylabel	为 y 轴添加标签
6	title	添加图窗标题
7	length	返回集合元素数量
8	abs	绝对值
9	floor	将 X 的每个元素四舍五入到小于或等于该元素的最接近整数
10	xlim	设置或查询 x 坐标轴范围

2.2 C++调用平台层 API

2.2.1 开发规范

Julia 提供了 C-API 可以用于将 Julia 代码集成到更大的 C/ C++项目中，而不需要用 C/ C++重写所有内容。因为几乎所有的编程语言都有调用 C 函数的方法，Julia C API 也可以用来建立进一步的语言桥梁(例如从 Python 或 C#调用 Julia)。详细参考：https://docs.julialang.org/en/v1/manual/embedding/

2.2.2 示例代码

以下例子演示了如何开发一个简单的 C++工程 ArrayMaker，它调用了 Julia 函数库 TyMath 的 magic 函数。

2.2.2.1 创建 C++工程

使用 Visual Studio 创建 C++工程，其中包括 2 个项目 ：ArrayMaker（算法项目）、ArrayMakerTest（测试项目）、，工程配置如下：

●开发环境：Microsoft Visual Studio

●环境变量：

✓ JULIA_DIR =（本平台 Julia 程序安装路径，如 C:\Program Files\MWorks.Syslab 2022\Tools\julia-1.7.3)

●将 $(JULIA_DIR)\bin 添加到 PATH 路径

●ArrayMarker 项目属性配置：

✓ 常规 - 配置类型：动态链接库（.dll）

✓ C/C++ - 常规 - 附加包含目录：$(JULIA_DIR)\include\julia\

✓ C/C++ - 预处理器 - 预处理器定义：添加宏 ARRAYMAKER_LIB

✓ 链接器 - 常规 - 附加库目录：$(JULIA_DIR)\lib

✓ 链接器 - 输入 - 附加依赖项：libjulia.dll.a;libopenlibm.dll.a

●ArrayMarkerTest 项目属性配置：

✓ 常规 - 配置类型：应用程序（.exe）

✓ 链接器 - 常规 - 附加库目录：$(OutDir)

✓ 链接器 - 输入 - 附加依赖项：ArrayMaker.lib

2.2.2.2 开发代码

开发好的 ArrayMaker 工程代码，其文件结构如下图所示：

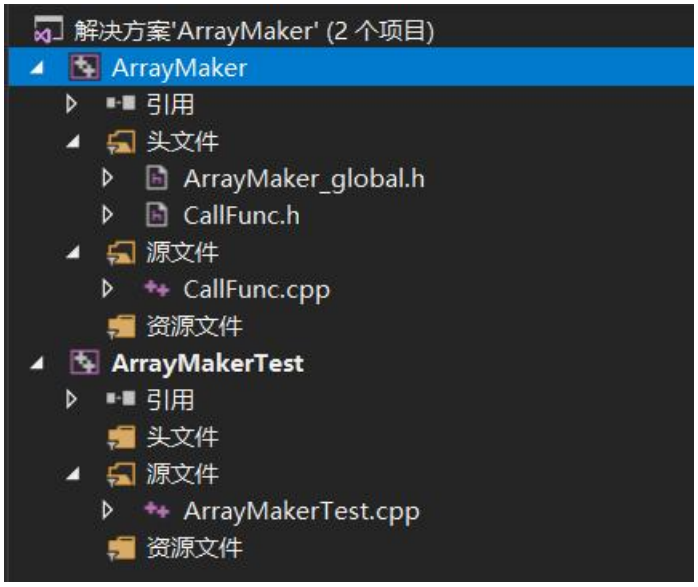


图 2- 6 C++代码文件结构

- ArrayMaker_global.h——动态库导出定义头文件；
- CallFunc.h——CallJulia 函数的头文件；
- CallFunc.cpp——CallJulia 函数的实现文件；
- ArrayMakerTest.cpp——测试代码文件
- start-env.bat——VS 工程启动脚本，因为该 VS 工程依赖同元 Julia 环境变量，所以需要通过 bat 脚本加载变量后启动 VS 工程

A. ArrayMaker_global.h

```
#pragma once
#ifndef BUILD_STATIC
# if defined(ARRAYMAKER_LIB)
#  define ARRAYMAKER_EXPORT __declspec(dllexport)
# else
#  define ARRAYMAKER_EXPORT __declspec(dllimport)
# endif
#else
# define ARRAYMAKER_EXPORT
#endif
```

B. CallFunc.h

```
#ifndef ARRAYMAKER_H
#define ARRAYMAKER_H
#include "ArrayMaker_global.h"
extern "C" ARRAYMAKER_EXPORT void CallJulia();
#endif
```

C. CallFunc.cpp

```
#include "CallFunc.h"
#include <uv.h>
#include <julia.h>
JULIA_DEFINE_FAST_TLS  // only define this once
void CallJulia()
{
          //初始化
          jl_init();
```

```
        //调用 Julia 数学库 TyMath
        jl_eval_string("using TyMath");
        jl_function_t *func = jl_get_function((jl_module_t *)jl_eval_string("TyMath"),
"magic");
        jl_value_t *argument1 = jl_box_int64(3);
        jl_array_t *ret = (jl_array_t*)jl_call1(func, argument1);
        int *yData = (int*)jl_array_data(ret);
        for (size_t i = 0; i < jl_array_len(ret); i++)
        {
                printf("%d ", yData[i]);
        }

        //退出
        jl_atexit_hook(0);
}
```

D. ArrayMakerTest.cpp

```
#include "../ArrayMaker/CallFunc.h"
int main()
{
    //调用同元 Julia 函数库 TyMath 求幻方矩阵 magic
    CallJulia();
}
```

E. start-env.bat

```
set JULIA_DEPOT_PATH=C:/Users/Public/TongYuan/.julia
@echo off
REM julia
set KMP_DUPLICATE_LIB_OK=TRUE
REM PythonCall
set JULIA_CONDAPKG_BACKEND=Null
set PYTHON_JULIAPKG_OFFLINE=yes
set JULIA_PYTHONCALL_EXE=@PyCall
@echo on
start /B "" "./ArrayMaker.sln"
```

上述代码中使用了平台层 API 中以下函数。

序号	函数名	简介
1	TyMath.magic	幻方矩阵

2.3 Python 调用平台层 API

2.3.1 开发规范

2.3.1.1 tyjuliacall 组件

tyjuliacall 组件是同元自研的用于实现 Python 调用 Julia 函数的 SDK。主要包含 2 个特性：

1. 跨平台同时支持静态链接 Python 和动态链接 Python；

2. 支持 Julia 系统镜像。

tyjuliacall 实现了将一些用于操控 Julia 自身的函数封装成 Python 对象，并将其挂载到 Python 可以访问的位置，此后 Python 使用这些对象，即可操控 Julia。其原理图如下所示：

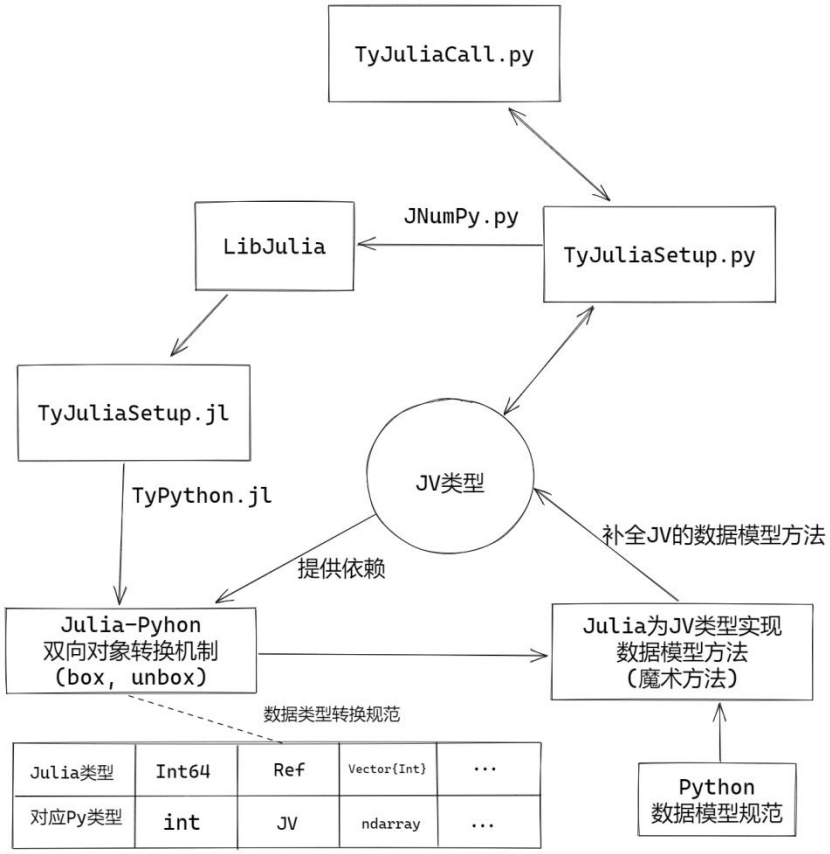


图 2- 7 tyjuliacall 组件原理

虽然 tyjuliacall 允许在 Python 和 Julia 之间传递任意数据，但由于是两门不同的语言，数据转换的类型对应关系是复杂的。为了保证代码的后向兼容性，使得规范的代码在不同版本的 tyjuliacall 上都可以运行，建议只使用如下的数据

类型转换。

2.3.1.2 Python 数据传递到 Julia

Python 向 Julia 函数传参时，推荐只使用下表左边的数据类型，以保证代码的后向兼容。

Python 类型	Julia 类型
基本类型	
int	Int64
float	Float64
bool	Bool
complex	ComplexF64
None	nothing
str	String
组合类型	
numpy.ndarray （dtype 为数字或字符串或 bool）	原生 Array
tuple，且元素均为表中数据类型	Tuple

2.3.1.3 Julia 数据传递到 Python

当获取 Julia 函数返回值，或导入 Julia 模块的非函数对象时，将发生 Julia 到 Python 的数据传递。保证后向兼容的 Julia 到 Python 数据转换关系如下表所示。

Julia 类型	Python 类型
基本类型	
Integer 子类型	int
AbstractFloat 子类型	float
Bool	bool
Complex 子类型	complex
nothing 对象	None
AbstractString 子类型	str
Vector{UInt8}	bytearray
组合类型	
AbstrctArray{T} （T 见下方说明）	numpy.ndarray
Tuple{T1, ..., Tn}，且 Ti 为该表中的类型	tuple

其余 Julia 类型	tyjuliacall.JV

一个 Julia AbstrctArray 能转换为 numpy 数组，当且仅当其元素类型 T 是以下类型之一：

- Int8, Int16, Int32, Int64, UInt8, UInt16, UInt32, UInt64
- Float16, Float32, Float64
- ComplexF16, ComplexF32, ComplexF64
- Bool

注意，当类型为 Vector{String}或者 Array{String, 2}的 Julia 对象被返回给 Python 时，它被封装为一个 tyjuliacall.JV 类型。

2.3.2 示例代码

以下例子演示了开发一个简单的 Python 程序，它调用了 Julia 函数库 TySignalProcessing 和 TyPlot。其文件结构如下图所示：

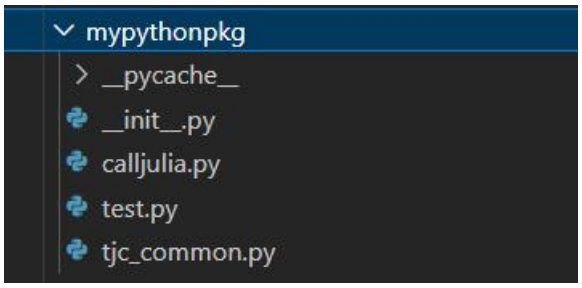


图 2- 8 python 代码文件结构

- __init__.py——Python 函数库入口文件；
- tjc_common.py——使用 Julia 镜像的函数模块；
- calljulia.py——调用了 Julia 函数库 TySignalProcessing 和 TyPlot，绘制中值滤波信号曲线；
- test.py——测试代码文件

A. __init__.py

```
name = "mypythonpkg"
version = __version__ = "0.0.1"
__all__ = ["tjc_common", "calljulia"]
```

B. tjc_common.py

```
import os
```

```
from os import system
from os.path import expanduser
import platform
import time
import sys
def get_timespan():
    global _t0
    t1 = time.time()
    span = t1 - _t0
    _t0 = t1
    return span
_t0 = time.time()
# 使用系统映像文件，提升 julia 包加载速度
from tyjuliasetup import use_sysimage
need_use_sysimage = True
if need_use_sysimage:
    sysimage_path = ""
    if platform.system().lower() == 'windows':
        sysimage_path =
"C:/Users/Public/TongYuan/.julia/environments/v1.7/JuliaSysimage.dll"
    elif platform.system().lower() == 'linux':
        sysimage_path = expanduser("~/TongYuan/.julia/environments/v1.7/JuliaSysimage.so")
    if os.path.exists(sysimage_path):
        use_sysimage(sysimage_path)
    else:
        print("%s 文件不存在！"%sysimage_path)
import tyjuliacall
print(f"导入 tyjuliacall: {get_timespan():.2f} s")
```

C. calljulia.py

```
from tyjuliacall import TySignalProcessing as sp
from tyjuliacall import TyPlot as tp
import numpy as np
def Medfilt1_Plot():
    fs = 100
    t = np.arange(fs + 1) / fs
    print(t)
    x = np.sin(2 * np.pi * t*3) + 0.25*np.sin(2 * np.pi * t*40)
    # 调用信号库函数
    y = sp.medfilt1(x, 9)
    # 调用图形库函数
```

```
    tp.plot(t, x, t, y)
    tp.legend(np.asarray(["Original", "Filtered"]))
    tp.plt.show()
```

D. test.py

```
from tjc_common import *
from calljulia import *
# 调用 Julia 函数库：1.调用信号库进行中值滤波计算；2.调用图形库绘制结果曲线
Medfilt1_Plot()
```

上述代码中使用了平台层 API 中以下函数。

序号	函数名	简介
1	TySignalProcessing.medfilt1	一维中值滤波
2	TyPlot.plot	二维线图
3	TyPlot.legend	在坐标区上添加图例

2.4 APP 调用平台层 API

本案例为使用 C++/Qt 图形应用开发环境开发的曲线拟合 APP。该 APP 分析了从 1790 到 1990 年间人口统计数据，通过多项式拟合算法预估未来的人口增长趋势。该案例文件路径为：<Syslab 安装路径>\Examples\10 AppDemos\cpp\。

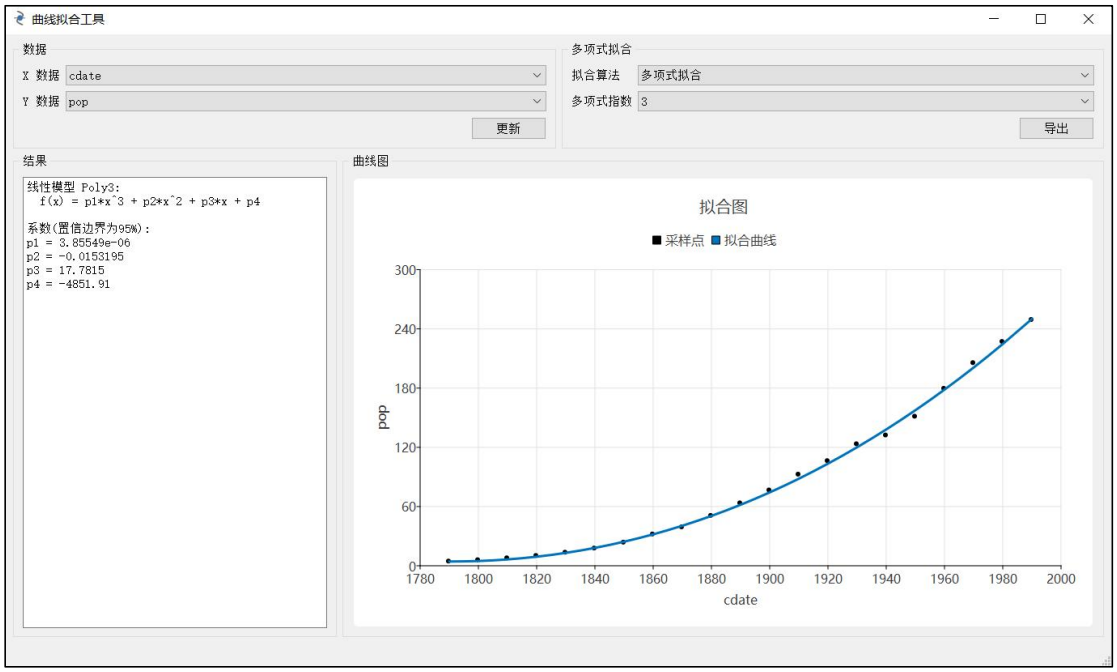


图 2- 9 曲线拟合工具

下文阐述该 APP 的开发过程中对平台层 API 的调用方法。APP 通过“更新”按钮，获取 Syslab 工作空间的变量列表，具体示例代码如下所示。

```
vector<VariableInfo> variables;
if (m_syslabSdk)
{
  m_syslabSdk->MwGetVariables(false, variables);
}
for (VariableInfo var : variables)
{
  QString var_type = QString::fromStdString(var.GetType());
  if (var_type.contains("Vector{Int64}")
    || var_type.contains("Vector{Float64}"))
  {
    ui.comboBox_x->addItem(QString::fromStdString(var.GetName()));
    ui.comboBox_y->addItem(QString::fromStdString(var.GetName()));
  }
}
```

“X 数据”和“Y 数据”下拉框，选中具体的变量名后，实时从 Syslab 工作空间获取变量值，具体示例代码如下所示。

```
  string value;
  m_syslabSdk->MwGetValue(var.toStdString(), value);
  QString qstr_value = QString::fromStdString(value);
```

曲线拟合计算算法过程，通过运行 Julia 脚本的方式，调用 Syslab 平台获取多项式拟合结果。第一步计算曲线拟合参数：

```
  // p = vec(TyMathCore.polyfit(cdate, pop, 4))
  QString str = QString("using TyMathCore;curvefit_coefficient =
vec(TyMathCore.polyfit(%1, %2, %3))").arg(x_datas_str).arg(y_datas_st
r).arg(n);
  m_syslabSdk->MwRunScript(str.toStdString().c_str());
  string res;
  m_syslabSdk->MwGetValue("curvefit_coefficient", res);
```

第二步计算曲线拟合结果：

```
  // res = polyval(p, cdate)
  QString str = QString("using TyMathCore;curvefit_res =
TyMathCore.polyval(curvefit_coefficient, %1)").arg(x_datas_str);
  m_syslabSdk->MwRunScript(str.toStdString().c_str(), true, true);
  String res;
```

```
m_syslabSdk->MwGetValue("curvefit_res", res);
```

上述算法代码中使用了平台层 API 中以下函数。

序号	函数名	简介
1	MwGetVariables	查询工作区变量名列表
2	MwGetValue	查询工作区变量 var 的值
3	MwRunScript	在工作区运行脚本，生成结果
4	TyMathCore.polyfit	返回次数为 n 的多项式 p(x)的系数，该阶数是 y 中数据的最佳拟合
5	TyMathCore.polyval	计算多项式 p 在 x 的每个点处的值

3. API 详细说明

3.1 基础 API

语法、数组索引和操作、数据类型、运算符。

基础知识包括基本运算，例如创建变量、数组索引、算术运算和数据类型。

3.1.1 输入命令

编译和运行 Syslab 语句

在 Syslab 中工作时，您可以发出创建变量和调用函数的命令。

函数名	简介
ans	最近计算的答案
clc	清空命令行窗口
clipboard	在目标与系统剪贴板之间复制和粘贴文本
Ctrl + D	终止 Syslab 程序（与 exit() 等效）
Ctrl + l	清空命令行窗口
Ctrl+R	打开命令历史记录窗口
diary	将命令行窗口文本记录到日志文件中

exit	终止 Syslab 程序
gethostname	获取本地计算机的主机名
import	导入整个库或者库中的模块
iskeyword	确定输入是否为 Syslab 关键字
ty_format	设置命令行窗口输出显示格式
using	加载整个库或者库中的模块，并使其可直接使用
varinfo	展示模块中的变量及大小和类型
VersionNumber	用于将字符串解析为版本号
@show	显示表达式和结果，返回结果
@time	用于执行表达式的宏，打印执行所需的时间、分配数及其字节总数

3.1.1.1 ans

最近计算的答案

A. 语法

```
ans
```

B. 说明

ans 是在未指定输出参数的情况下返回输出时创建的变量。Julia 创建了 ans 变量，并用该变量存储输出。建议不要在脚本或函数中更改或使用 ans 的值，因为该值可能会经常变化。

ans 特定于当前工作区。基础工作区和每个函数工作区可以有自己的 ans 实例。

C. 示例

(1). 简单计算的结果

在命令行窗口中执行简单计算，而不将结果赋给变量。Julia 将结果存储在 ans 变量中。

```
2 + 2
```

ans = 4

在命令行窗口中执行简单计算，并将结果赋给变量 result。

```
result = 4 + 4
```

```
result = 8
```

显示 result 的值，然后显示 ans 的值。Julia 显示 result 的值并返回输出。因此，ans 的值改变。

```
result
```

```
result = 8
```

```
ans
```

```
ans = 8
```

(2). 调用返回输出的函数

假设您有函数 testFunc，它返回输出但不指定输出变量。

```
function testFunc()
  a = 75
end
```

调用 testFunc。Julia 将返回的结果存储在 ans 中。

```
testFunc()
```

```
ans = 75
```

3.1.1.2 clc

清空命令行窗口

A. 语法

```
clc()
```

B. 说明

clc 清除命令行窗口中的所有文本，让屏幕变得干净。运行 clc 后，您可以在命令行窗口中使用向上箭头键 ↑ 从命令历史记录中重新调用语句。

C. 示例

(1). 清除所有输出

在运行命令之前，从命令行窗口清除所有文本。

```
创建一个 50×50 随机数矩阵。
rand(50, 50)
```

清除命令行窗口中的所有文本。

```
clc()
```



3.1.1.3 clipboard

在目标与系统剪贴板之间复制和粘贴文本

A. 语法

```
clipboard()
clipboard(x)
```

B. 说明

clipboard() 返回一个包含操作系统剪贴板内容的字符串

clipboard(x) 将指定的数据复制到剪贴板。

注意

在 CentOS 系统下，若使用 clipboard 函数，系统需安装 xsel 或 xclip 库。

C. 示例

(1). 返回复制文本的字符串

将文字 "这是一个文本示例" 进行复制,在命令行窗口得到内容对应字符串

```
clipboard()
```

```
ans = "这是一个文本示例"
```

(2). 复制和粘贴数据

创建一个向量，并将其复制到剪贴板中

```
a = [1,2,3]
clipboard(a)
```

然后将剪贴板数据粘贴到 Julia 中。

```
julia> [1, 2, 3]
```

3.1.1.4 Ctrl + D

终止 Syslab 程序（与 exit() 等效）

A. 语法

```
Ctrl + D
```

B. 说明

Ctrl + D 终止 Syslab 的当前会话。

3.1.1.5 Ctrl + l

清空命令行窗口

A. 语法

```
Ctrl+l
```

B. 说明

Ctrl + l 清除命令行窗口中的所有文本，让屏幕变得干净。运行 Ctrl + l 后，您可以使用命令行窗口中的滚动条查看以前显示的文本。同时也可以在命令行窗口中使用向上箭头键 ↑ 从命令历史记录中重新调用语句。示例

使用 Julia 代码文件中的 Ctrl + l 始终都能在屏幕的同一起始位置显示输出。

C. 示例

清除所有输出

在运行命令之前，从命令行窗口清除所有文本。

创建一个 50×50 随机数矩阵。

```
rand(50,50)
```

```
ans = 50×50 Matrix{Float64}:
0.475877   0.124215   0.797714   ⋯  0.719639   0.565316   0.785584
0.291716   0.212411   0.708506      0.356619   0.0686172  0.777049
0.0959816  0.333913   0.423491      0.409701   0.843592   0.737802
0.429282   0.12094    0.209566      0.682475   0.98249    0.865777
0.126204   0.745456   0.966458      0.846331   0.88821    0.498557
0.508712   0.0407774  0.360717   ⋯  0.393123   0.98694    0.0758321
0.911458   0.338182   0.280671      0.619247   0.694321   0.781878
0.960495   0.294711   0.872697      0.351884   0.654085   0.0255139
0.668258   0.329244   0.89326       0.502044   0.609722   0.389837
0.220794   0.342415   0.323404      0.41002    0.351955   0.756778
0.319158   0.897222   0.978362   ⋯  0.61991    0.341546   0.123712
0.307578   0.425961   0.996714      0.548505   0.593379   0.597804
0.667548   0.494943   0.0275338     0.0580263  0.54338    0.420912
0.94303    0.325251   0.601861      0.63838    0.827777   0.371517
⋮                                  ⋱
```

清除命令行窗口中的所有文本。

```
Ctrl + l
```



```
rand(10,10)
```

```
ans = 10×10 Matrix{Float64}:
0.31833      0.159274     0.0997747   0.309299     0.748113     0.840115
0.400271   0.808501    0.182633  0.213116
0.645109     0.89052      0.972419     0.226245     0.389063     0.651295
0.418397   0.85585     0.750823  0.823861
0.647417     0.741115     0.316631     0.634422     0.69705      0.27943
0.726211   0.741611    0.54655   0.49543
0.677855     0.141086     0.438171     0.690749     0.222555     0.866474
0.752505   0.518317    0.626641  0.293475
0.727362     0.265377     0.0267834    0.822602     0.0745887    0.0634396
0.640531   0.238078    0.587292  0.0367255
0.0216832    0.162734     0.79191      0.0993459    0.943821     0.996131
0.0911241  0.974532    0.522224  0.673982
0.380204     0.0619821    0.273708     0.42878      0.445843     0.865193
0.0515277  0.435901    0.217719  0.087273
0.0084195    0.991341     0.710098     0.376696     0.818471     0.955646
0.796994   0.00275317  0.854163  0.48134
0.0338127    0.510885     0.795183     0.505696     0.961827     0.773831
0.917231   0.92329     0.720667  0.636724
0.314513     0.823529     0.473819     0.98454      0.543923     0.689055
0.84587    0.0134742   0.510739  0.757475
```

3.1.1.6 Ctrl+R

打开命令历史记录窗口

A. 语法

```
diary(filename)
diary("off")
diary("on")
```

B. 说明

命令历史记录窗口显示您在当前和之前的 Syslab 会话中所运行的语句的记录。

Syslab 会将命令行窗口中运行的语句保存到历史记录文件 repl_history.jl 中。这些语句包括您使用工具（例如编辑器、命令历史记录窗口）中上下文菜单上的执行所选内容选项运行的语句。默认情况下，Syslab 会在每个语句后自动保存命令历史记录文件。历史记录文件并不包括在 Syslab 中执行的每项操作。例

如，在变量编辑器中对值所做的更改并不包括在命令历史记录窗口中。所有条目均会保留，直到将其删除。

您可以在命令历史记录窗口中选择条目，然后对所选条目执行重新运行。

打开 命令历史记录窗口

要打开显示上一条历史记录，请在命令行窗口中按向上箭头键（↑）；要打开显示所有历史记录的命令历史记录窗口，请按快捷键 Ctrl + R。要显示特定语句，请键入语句的任何部分，然后按向上箭头键。

默认情况下，在选择语句或按下两次 Esc 键后，命令历史记录窗口将关闭。

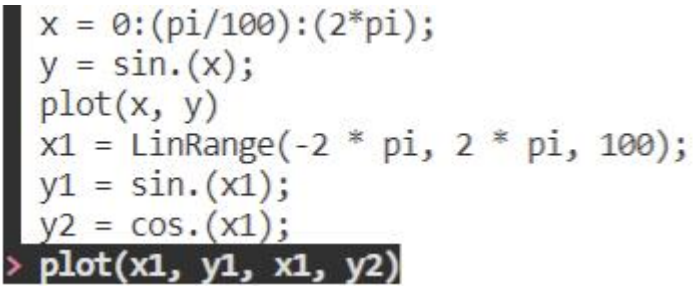
C. 示例

使用命令历史记录搜索功能查找并运行以前的 plot 语句。

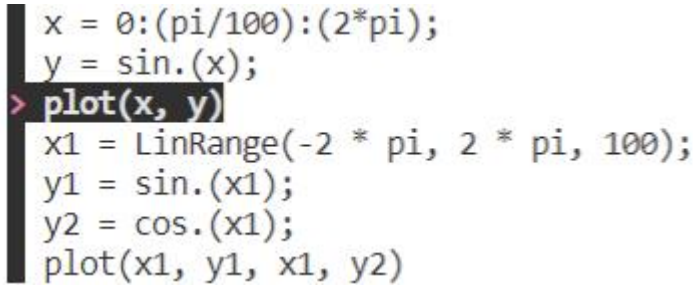
在命令行窗口中，运行以下语句以创建两个线图。

```
x = 0:(pi/100):(2*pi);
y = sin.(x);
plot(x, y)
x1 = LinRange(-2 * pi, 2 * pi, 100);
y1 = sin.(x1);
y2 = cos.(x1);
plot(x1, y1, x1, y2)
```

按 Ctrl+R 键。Syslab 显示命令历史记录。plot 的最后一个运行实例处于选中状态。



按向上箭头键以选择 plot 的上一个运行实例，然后按 Enter。Syslab 会运行语句 plot(x,y)。



3.1.1.7 diary

将命令行窗口文本记录到日志文件中

A. 语法

```
diary(filename)
diary("off")
diary("on")
```

B. 说明

diary filename 将生成的日志保存到 filename。如果该文件已存在，Syslab 会将文本追加到文件末尾。Syslab 在执行其他语句并有日志内容可记录时，才生成日志文件。要查看当前 diary 文件的名称，请键入 get(0,'DiaryFile')。

diary("off") 禁用日志记录。

diary("on") 使用当前 diary 文件名启用日志记录。

C. 示例

(1). 记录语句和输出

创建一个 diary 文件并记录多个语句及其输出。

启用日志记录并将生成的日志保存到 myDiaryFile。

```
using TyVSCodeServerEx
diary("myDiaryFile")
```

执行计算，并在命令行窗口中创建并显示由 1 组成的矩阵。

```
a = 1;
b = sin(a);
x = ones(4, 4)
```

```
x
 1.0  1.0  1.0  1.0
 1.0  1.0  1.0  1.0
 1.0  1.0  1.0  1.0
 1.0  1.0  1.0  1.0
```

禁用日志记录并在命令行窗口中显示日志文件。

```
diary("off")
type("myDiaryFile")
```

```
a = 1;
b = sin(a);
```

```
x = ones(4, 4)
x
 1.0  1.0  1.0  1.0
 1.0  1.0  1.0  1.0
 1.0  1.0  1.0  1.0
 1.0  1.0  1.0  1.0
diary("off")
```

3.1.1.8 exit

终止 Syslab 程序（与 Ctrl+D 等效）

A. 语法

```
exit()
```

B. 说明

exit 终止 Syslab 的当前会话。

3.1.1.9 gethostname

获取本地计算机的主机名

A. 语法

```
gethostname()
```

B. 说明

gethostname() 直接获取本地计算机的主机名。

C. 示例

(1). 获取本地计算机的主机名

```
gethostname()
```

```
"LAPTOP-GMFWN01"
```

3.1.1.10 import

导入整个库或者库中的模块

A. 语法

```
import Pkg
import Pkg.Module
import Pkg.Function
```

B. 说明

import 可导入列表作用域为：库，模块，函数。

import Pkg 导入整个 Pkg 库。

import Pkg.Module 导入 Pkg 库下的模块 Module。

import Pkg.Function 导入 Pkg 库下的函数 Function

C. 示例

(1). 导入整个 Pkg 库

```
import TyMath
```

(2). 导入 Pkg 库下的模块

```
Moduleimport TyMath.LinearAlgebra
```

(3). 导入 Pkg 库下的函数

```
Functionimport LinearAlgebra.ldlt
```

(4). 导入不存在的库

```
import test1
```

返回结果如下：

```
ERROR: ArgumentError: Package test1 not found in current path:\- Run
`import Pkg; Pkg.add("test1")` to install the test1 package.
```

3.1.1.11 iskeyword

确定输入是否为 Syslab 关键字

A. 语法

```
tf = iskeyword(txt)
iskeyword()
```

B. 说明

此 Syslab 函数 将返回逻辑值 1 (true)；否则将返回逻辑值 0 (false)。Syslab 关键字不能用作变量名称。

iskeyword() 返回所有 Syslab 关键字的列表。

C. 示例

(1). 测试单词 while 是否为 Syslab 关键字

```
iskeyword("while")
```

```
ans =
true
```

(2). 获取所有 Syslab 关键字的列表

```
iskeyword()
```

```
ans =
29-element Vector{String}:
 "begin"
 "baremodule"
 "for"
 "break"
 "macro"
 "do"
 ⋮
 "const"
 "function"
 "finally"
 "if"
 "local"
 "using"
```

3.1.1.12 ty_format

设置命令行窗口输出显示格式

A. 语法

```
ty_format("style",x)
ty_format("",x)
```

B. 说明

ty_format("style",x)将命令行窗口中 x 的输出显示格式更改为 style 指定的

格式。

ty_format("",x)自行将输出格式重置为默认值，即 Julia 语言中 x 默认的输出格式和适用于所有输出行的紧密行距。

数值格式仅影响数字显示在命令行窗口输出中的方式，而不是 Syslab 计算或保存它们的方式。

C. 示例

(1). 长格式

将输出格式设置为长固定十进制小数点格式并显示 pi 的值。

```
ty_format("long", pi)
```

```
ans = 3.141592653589793
```

(2). 将格式重置为默认值

以十六进制格式显示整数。

```
x = rand(3, 3);
ty_format("shortEng", x);
ty_format("compact", x)
```

```
594.2141E-3    278.9765E-3    869.2780E-3
400.0349E-3    324.8735E-3    481.5277E-3
470.4801E-3    957.5011E-3    297.4870E-3
```

将显示格式重置为默认值，并再次显示矩阵。

```
ty_format("", x)
```

```
0.5942140781108288     0.27897650637407556     0.869278026112899
0.4000349204388156     0.3248734736903671      0.4815277376512693
0.47048009648557054    0.9575010800104414      0.2974870449436784
```

(3). 十六进制格式

以十六进制格式显示整数的最大值。

```
ty_format("hex", typemax(UInt64))
```

```
ans = ffffffffffffffff
```

(4). 短工程记数法和长工程记数法

显示 shortENG 和 longEng 格式之间的差异。

将输出格式设置为 shortENG

创建变量，然后通过 for 循环每次将该变量的值增加 10 的倍数。

```
A = 5.123456789;
```

```
for k in 1:10
    ty_format("shortENG", A)
    global
A = A * 10
end
```

```
5.1235E0
51.2346E0
512.3457E0
5.1235E3
51.2346E3
512.3457E3
5.1235E6
51.2346E6
512.3457E6
5.1235E9
```

这些值显示为在小数点后包含 4 位数，指数为 3 的倍数。

将输出格式设置为长工程格式并查看相同值。

```
A = 5.123456789;
for k in 1:10
    ty_format("longENG", A)
    global A = A * 10
end
```

```
5.12345678900000E0
51.2345678900000E0
512.345678900000E0
5.12345678900000E3
51.2345678900000E3
512.345678900000E3
5.12345678900000E6
51.2345678900000E6
512.345678900000E6
5.12345678900000E9
```

这些值显示为 15 位数，指数为 3 的倍数。

(5). 大数据范围格式

当数组中的一些值数字少，而指数大时，请使用 shortG 格式。shortG 格式在短固定十进制小数点格式和短科学记数法中选取最紧凑的显示格式。

创建一个变量并以默认的 short 格式显示输出。

```
x = [25 56.31156 255.52675 9876899999];
ty_format("short", x)
```

```
25.0000 56.3116 255.5267       9876899999.0000
```

将格式设置为 shortG 并重新显示这些值。

```
ty_format("shortG", x)
```

```
25      56.31   255.5   9.877e+09
```

(6). 获取当前格式

获取当前显示格式。

```
f = format_get(0, "Format")
S = format_get(0, "FormatSpacing")
```

```
f = "default"
S = "compact"
```

3.1.1.13 using

加载整个库或者库中的模块，并使其可直接使用

A. 语法

```
using Pkg
using Pkg.Module
using Pkg:Function
```

B. 说明

using 可加载作用域为：库，模块，函数。

using Pkg 加载整个 Pkg 库。

using Pkg.Module 加载 Pkg 库下的模块 Module。

using Pkg:Function 加载 Pkg 库下的函数 Function。

C. 示例

(1). 加载整个 Pkg 库

```
using TyMath
```

(2). 加载 Pkg 库下的模块

```
Moduleusing TyMath.LinearAlgebra
```

(3). 加载 Pkg 库下的函数

```
Functionusing LinearAlgebra:ldlt
```

(4). 加载不存在的库

```
using test1
```

返回结果如下：

```
ERROR: ArgumentError: Package test1 not found in current path:\- Run
`import Pkg; Pkg.add("test1")` to install the test1 package.
```

3.1.1.14 varinfo

展示模块中的变量及大小和类型

A. 示例

(1). 显示工作区变量名称

列出当前工作区中以字母 a 开头的变量的名称。

```
a = 1
ab = 2

lion = 3
lions = 4
varinfo(r"^a")
```

```
  name    size summary
  - - - -  - - - - - - - - - - - - - - -
  a    8 bytes Int64
  ab   8 bytes Int64
  ans  8 bytes Int64
```

显示当前工作区中以 ion 结尾的变量的名称。

```
varinfo(r"ion$")
```

```
name    size summary
- - - -  - - - - - - - - - - - - - - -
lion 8 bytes Int64
```

(2). 列出嵌套或匿名函数中的工作区变量

列出当前工作区中在嵌套函数中暂停的所有变量名称。

创建文件 who_demo.m，其中包含以下语句。

```
function who_demo()
   global date_time = Dates.format.(DateTime(now()), "dd-U-yyyy")
```

```
  global date_time_array = split(date_time, "-")


 function get_date(d)
    global day = d[1]
    global mon = d[2]
     global year = d[3]
  end
  get_date(date_time_array)end
```

运行 who_demo。Syslab 将在出现 keyboard 命令的行中暂停。

```
who_demo()
```

调用 getvariables 函数。

```
res = TyBase.getvariables()
```

```
6-element Vector{Any}:
 :ans
 :date_time
 :date_time_array
 :day
 :mon
 :year
```

3.1.1.15 VersionNumber

用于将字符串解析为版本号

A. 语法

```
VersionNumber(str)
```

B. 说明

VersionNumber(str)用于将字符串解析为版本号。示例

C. 示例

将字符串解析为版本号

```
VersionNumber("1.2.3")
```

```
v"1.2.3"
```

```
VersionNumber("2.0.1-rc1")
```

```
v"2.0.1-rc1"
```

3.1.1.16 @show

显示表达式和结果，返回结果

A. 语法

```
@show 表达式
```

B. 说明

@show 表达式 例如 a=1; @show a,则会返回 a 的表达式和 a 的结果。

C. 示例

(1). 显示表达式和结果，返回结果

定义一个表达式

```
a = 1
```

```
1
```

显示 a 的表达式和结果，返回结果

```
@show a
```

```
a = 1
1
```

3.1.1.17 @time

用于执行表达式的宏，打印执行所需的时间、分配数及其字节总数

A. 语法

```
@time 表达式
```

B. 说明

@time 表达式 显示打印执行所需的时间、分配数及其字节总数。

C. 示例

(1). 执行表达式

执行表达式

```
x = rand(10,10);
@time x * x;
```

```
0.508223 seconds (2.41 M allocations: 129.143 MiB, 6.69% gc time,
99.95% compilation time)
```

3.1.2 环境和设置

预设和设置

3.1.2.1 系统命令

函数名	简介
ispc	确定版本是否适用于 Windows (PC) 平台
pause	暂时停止执行 Syslab
perl	使用操作系统可执行文件调用 Perl 脚本
system	执行操作系统命令并返回输出

3.1.2.1.1 ispc

确定版本是否适用于 Windows (PC) 平台。

A. 说明

如果 Syslab 软件的版本适用于 Microsoft® Windows® 平台，tf = Sys.iswindows 将返回 true。否则，将返回 false。

B. 示例

根据平台执行 Syslab 命令

```
if Sys.isapple()
    #Code to run on Apple platform
elseif Sys.isunix()
    #Code to run on Unix platform
elseif Sys.iswindows()
    #Code to run on Windows platform
else
    display("Platform not supported")
```

```
end
```

3.1.2.1.2 pause

暂时停止执行 Syslab

A. 语法

```
pause(n)
pause(state)
oldState = pause(state)
```

B. 说明

pause(n) 暂停执行 n 秒，然后继续执行。必须启用暂停，此调用才能生效。

pause(state) 启用、禁用或显示当前暂停设置。

oldState = pause(state) 返回当前暂停设置并如 state 所示设置暂停状态。例如，如果已启用暂停功能，oldState = pause("off") 会在 oldState 中返回 "on" 并禁用暂停。

C. 示例

(1). 暂停执行

暂停执行 5 秒。Syslab 在暂停执行时会阻止或隐藏命令提示符 (>)。

```
n = 5;
pause(n)
```

(2). 禁用暂停设置

禁用暂停设置并查询当前状态。

```
pause("off")
pause("query")
```

```
ans =
"off"
```

暂停执行 100 秒。由于暂停设置为 off（关闭），Syslab 会忽略暂停执行的请求，并立即返回命令提示符。

```
pause(100)
```

启用暂停设置。

```
pause("on")
```

(3). 保存并还原暂停状态

存储当前暂停设置，然后禁用暂停执行功能。

```
oldState = pause("off")
```

```
oldState =
"on"
```

查询当前暂停设置。

```
pause("query")
```

```
ans =
"off"
```

恢复初始的暂停状态。

```
pause(oldState)
pause("query")
```

```
ans =
"on"
```

您也可以存储暂停状态的查询值，然后禁用暂停执行功能。

```
oldState = pause("query")
pause("off")
```

恢复初始的暂停状态。

```
pause(oldState)
```

3.1.2.1.3 perl

使用操作系统可执行文件调用 Perl 脚本

A. 语法

```
perl(perlfile)
perl(perlfile,args...)
result = perl(___)
```

B. 说明

perl(perlfile) 调用 Perl 脚本 perlfile。

perl(perlfile,args...) 使用参数 arg1,...,argN 调用脚本。

result = perl(___) 返回结果。此选项可与先前语法中的任何输入参数一起使用。

C. 示例

(1). 运行 hello.pl Perl 脚本

创建文件 hello.pl，其中包含以下语句：

$input = $ARGV[0]; print "Hello $input."; 将此文件保存到您的 Syslab 路径中。

在 Syslab 命令行中，键入：

```
filepath                =                pkgdir(TyBase)                *
"/examples/SystemCommands/perl/hello.pl"
perl(filepath, "World")
```

```
ans =
Hello World.Process(`perl 'E:\Syslab\bran
ches\TyBase/examples/SystemCommands/perl/hello.pl' World
`, ProcessExited(0))
```

3.1.2.1.4 system

执行操作系统命令并返回输出

A. 语法

```
cmdout = system(command)
```

B. 说明

cmdout = system(command) 调用操作系统执行指定的命令。操作会等待命令执行完毕，然后再将命令的输出返回到 cmdout。

该函数将启动新的 cmd/shell 进程、执行 command、退出进程，然后返回到 Syslab 进程。command 对系统环境所做的更新对 Syslab 不可见。

C. 示例

(1). Windows:显示操作系统命令状态和输出

使用 cd 命令显示当前文件夹。Syslab 将在 cmdout 中返回一个包含当前文件夹的字符向量。

```
command = "cd c://";
cmdout = system(command)
```

(2). Windows:保存命令退出状态

要创建一个名为 mynew 的文件夹，请调用 mkdir 命令并将退出状态保存到一个变量。status 为零表示 mynew 文件夹已成功创建。

```
command = "mkdir mynew";
status = system(command)
```

(3). Windows:打开并运行 UI 命令

通过将符号 & 附加到 notepad 命令，打开 Microsoft® 记事本并立即将退出状态返回至 Syslab。status 为零表示记事本已成功启动。

```
status = system("notepad &")
```

(4). Windows:保存命令输出

执行 ls 命令并查看退出状态和命令输出。cmdout 包含命令输出。

```
cmdout = system("ls");
```

(5). Windows:保存不成功的命令状态和错误消息

尝试执行名为 badcmd 的命令。然后，查看 results 输出参数。

```
results = system("badcmd")
```

(6). UNIX：保存命令退出状态和输出

列出当前已登录的所有用户，然后保存命令退出状态和输出。然后，查看该状态。Syslab 将在 cmdout 中返回用户列表。

```
command = "who";
cmdout = system(command);
```

3.1.2.2 平台和许可

函数名	简介
ver	Syslab 产品的版本信息

3.1.2.2.1 ver

Syslab 产品的版本信息

A. 语法

```
ver()
ver(name)
```

B. 说明

ver()显示：

标题，包含当前 Syslab 产品系列版本号、操作系统。

Syslab 的版本号和所有其他已安装的 Syslab 的产品。

ver(name)会显示特定版本产品的信息，包括版本号、操作系统。

C. 示例

(1). 所有已安装产品的版本

显示所有已安装产品的版本信息。这里显示的输出内容有代表性。您的结果可能有所不同。

```
ver();
```

```
-----------------------------------------------
Syslab 版本:0.9.8800 (R2023a)
VsCode API 版本:1.69.2
操作系统:Windwos_NT_win32_10.0.19044
-----------------------------------------------
TyStatistics           版本 0.1.8    R2023a
TyBase                 版本 0.7.7    R2023a
TySignalProcessing     版本 0.8.3    R2023a
TyControlSystems       版本 0.2.1    R2023a
TyCommunication        版本 0.3.3    R2023a
TyGeoGraphics          版本 0.6.2    R2023a
TySymbolicMath         版本 0.7.1    R2023a
TyDSPSystem            版本 0.6.2    R2023a
TyCurveFitting         版本 0.6.4    R2023a
TyCEI                  版本 0.1.0    R2023a
TyVSCodeServerEx       版本 0.1.0    R2023a
TyMath                 版本 0.7.1    R2023a
TyGlobalOptimization   版本 0.1.0    R2023a
TyPlot                 版本 0.7.6    R2023a
TyImages               版本 0.7.2    R2023a
TyOptimization         版本 0.1.0    R2023a
```

(2). 特定产品的版本

显示 Syslab 和 TyControlSystems 产品的版本信息。这里显示的输出内容有代表性。您的结果可能有所不同。将 TyControlSystems 产品信息作为参数指定给 ver。

```
ver("TyControlSystems");
```

```
-----------------------------------------------
Syslab 版本:0.9.8800 (R2023a)
VsCode API 版本:1.69.2
操作系统:Windwos_NT_win32_10.0.19044
-----------------------------------------------
TyControlSystems        版本 0.2.1    R2023a
```

(3). 包含 SYSLAB 产品的版本的结构体

创建一个包含 Syslab 版本信息的结构体，然后显示结构体值。这里显示的输出内容有代表性。您的结果可能有所不同。

```
v=ver();
for k = 1:length(v)
  println(v[k].Name)
  println("  Version: ", v[k].Version)
end
```

```
-----------------------------------------------
Syslab 版本:0.9.8800 (R2023a)
VsCode API 版本:1.69.2
操作系统:Windwos_NT_win32_10.0.19044
-----------------------------------------------
TyStatistics          版本 0.1.8    R2023a
TyBase               版本 0.7.7    R2023a
TySignalProcessing     版本 0.8.3    R2023a
TyControlSystems       版本 0.2.1    R2023a
TyCommunication        版本 0.3.3    R2023a
TyGeoGraphics          版本 0.6.2    R2023a
TySymbolicMath         版本 0.7.1    R2023a
TyDSPSystem           版本 0.6.2    R2023a
TyCurveFitting         版本 0.6.4    R2023a
TyCEI                版本 0.1.0    R2023a
TyVSCodeServerEx       版本 0.1.0    R2023a
TyMath               版本 0.7.1    R2023a
TyGlobalOptimization    版本 0.1.0    R2023a
TyPlot               版本 0.7.6    R2023a
TyImages              版本 0.7.2    R2023a
```

```
TyOptimization          版本 0.1.0    R2023a

TyStatistics
  Version: 0.1.8
TyBase
  Version: 0.7.7
TySignalProcessing
  Version: 0.8.3
TyControlSystems
  Version: 0.2.1
TyCommunication
  Version: 0.3.3
TyGeoGraphics
  Version: 0.6.2
TySymbolicMath
  Version: 0.7.1
TyDSPSystem
  Version: 0.6.2
TyCurveFitting
  Version: 0.6.4
TyCEI
  Version: 0.1.0
TyVSCodeServerEx
  Version: 0.1.0
TyMath
  Version: 0.7.1
TyGlobalOptimization
  Version: 0.1.0
TyPlot
  Version: 0.7.6
TyImages
  Version: 0.7.2
TyOptimization
  Version: 0.1.0
```

(4). 包含特定产品的版本的结构体

创建一个只包含 TySymbolicMath 产品的版本信息的结构体。这里显示的输出内容有代表性。您的结果可能有所不同。

```
v=ver("TySymbolicMath")
```

```
-----------------------------------------------
Syslab 版本:0.9.8800 (R2023a)
```

```
VsCode API 版本:1.69.2
操作系统:Windwos_NT_win32_10.0.19044
------------------------------------------------
TySymbolicMath          版本 0.7.1    R2023a

  包含以下字段的结构体:

    名称: TySymbolicMath
    版本: 0.7.1
    发布: R2023a
```

3.1.3 数据导入和分析

导入和导出数据，包括大文件；预处理数据、可视化和浏览

3.1.3.1 数据导入和导出

文本文件和其他文件格式

3.1.3.1.1 低功耗 Bluetooth 通信

函数名	简介
read_serial_port	从串行端口设备读取数据
write_serial_port	将数据写入串行端口设备

read_serial_port

从串行端口设备读取数据

A. 语法

```
data = read_serial_port(device,count,datatype)
```

B. 说明

data = read_serial_port(device,count,datatype)以指定的形式读取值的数量并返回数据。该参数是标准 Syslab 数据类型的字符串或数据类型。

C. 示例

(1). 用串行端口设备读写数据

打开串行端口设备读写数据。在此示例中，COM2 为串行端口设备。

```
using LibSerialPort

device = LibSerialPort.open("COM2", 9600)
```

将值写入设备。

```
write_serial_port(device, 1:5, "UInt8")
```

读取所有数据。

```
read_serial_port(device, 5, "UInt8")
```

write_serial_port

将数据写入串行端口设备

A. 语法

```
write_serial_port(device,data,datatype)
```

B. 说明

write_serial_port(device,data,datatype)以 datatype 指定的形式写入数据，而不考虑 data 中的格式。

C. 示例

(1). 用串行端口设备读写数据

打开串行端口设备读写数据。在此示例中，COM1 为串行端口设备。

```
using LibSerialPort

device = LibSerialPort.open("COM1", 9600)
```

将值写入设备。

```
write_serial_port(device, 1:5, "UInt8")
```

读取所有数据。

```
read_serial_port(device, 5, "UInt8")
```

3.1.3.1.2 低级文件 I/O

函数名	简介
fclose	关闭一个或所有打开的文件
feof	检测文件末尾
fgetl	读取文件中的行，并删除换行符
fgets	读取文件中的行，并保留换行符
fopen	打开文件或获得有关打开文件的信息
fprintf	将数据写入文本文件
fread	读取二进制文件中的数据
fscanf	读取文本文件中的数据
fwrite	将数据写入二进制文件

fclose

关闭一个或所有打开的文件

A. 语法

```
fclose(fileID)
fclose("all")
```

B. 说明

fclose(fileID) 关闭打开的文件。

fclose("all") 关闭所有打开的文件。

C. 示例

(1). 打开和关闭文件

打开 badpoem.txt 文件，读取第一行，然后关闭该文件。

使用 fopen 打开文件。此函数将指定一个唯一的文件 ID，用于读取和写入文件。

```
file_path = pkgdir(TyBase) * "/examples/resources/badpoem.txt"
```

```
fid = fopen(file_path)
```

使用 fgetl 读取文件的第一行。

```
fgetl(fid)
```

```
"Oranges and lemons,"
```

关闭文件。

```
fclose(fid);
```

feof

检测文件末尾

A. 语法

```
status = feof(fileID)
```

B. 说明

status = feof(fileID) 返回文件末尾指示符的状态。如果之前的操作为指定文件设置了文件末尾指示符，feof 将返回 true。否则，feof 将返回 false。

C. 示例

(1). 读取到文件末尾

从这个 badpoem.txt 文件中一次读取一行，一直到文件的末尾。

使用 fopen 打开文件。此函数将指定一个唯一的文件 ID，用于读取和写入文件。

```
file_path = pkgdir(TyBase) * "/examples/resources/badpoem.txt"
fid = fopen(file_path)
```

一次读取并显示一行，一直到文件的末尾。

```
while ~feof(fid)
    tline = fgetl(fid)
    display(tline)
end
```

```
"Oranges and lemons,"
"Pineapples and tea."
"Orangutans and monkeys,"
"Dragonflys or fleas."
```

关闭文件。

```
fclose(fid)
```

fgetl

读取文件中的行，并删除换行符

A. 语法

```
tline = fgetl(fileID)
```

B. 说明

tline = fgetl(fileID) 返回指定文件中的下一行，并删除换行符。

如果文件非空，则 fgetl 以字符向量形式返回 tline。

如果文件为空且仅包含文件末尾标记，则 fgetl 以数值 "" 的形式返回 tline。

C. 示例

(1). 一次读取文件的一行

读取文件的一行内容，第一次读取时先排除换行符，第二次读取时再包含换行符。使用以下文件。

要读取文件 badpoem.txt 的第一行，请使用 fopen 打开该文件。然后使用 fgetl 读取第一行，此函数读取时排除换行符。

```
file_path = pkgdir(TyBase) * "/examples/resources/badpoem.txt"
fid = fopen(file_path)
line_ex = fgetl(fid)
```

```
"Oranges and lemons,"
```

要再次读取文件的第一行，请首先将读取位置指针重置到文件的开头。

```
seekstart(fid)
```

使用 fgets 函数读取文件 badpoem.txt 的第一行，此函数读取时包含换行符。

```
line_in = fgets(fid)
```

```
"Oranges and lemons,\n"
```

通过检查 fgetl 和 fgets 函数返回的行的长度，比较二者的输出。

```
length(line_ex)
```

```
19
```

```
length(line_in)
```

```
20
```

fgetl 返回的输出显示为一行，而 fgets 返回的输出包含换行符，因此显示为两行。

```
line_ex
```

```
"Oranges and lemons,"
```

```
line_in
```

```
"Oranges and lemons,\n"
```

关闭文件。

```
fclose(fid)
```

fgets

读取文件中的行，并保留换行符

A. 语法

```
tline = fgets(fileID)
tline = fgets(fileID,nchar)
```

B. 说明

tline = fgets(fileID) 读取指定文件中的下一行内容，并包含换行符。

tline = fgets(fileID,nchar) 返回下一行中的最多 nchar 个字符。

C. 示例

(1). 一次读取文件的一行

读取文件的一行内容，第一次读取时先排除换行符，第二次读取时再包含换行符。使用以下文件。

要读取文件 badpoem.txt 的第一行，请使用 fopen 打开该文件。然后使用 fgetl 读取第一行，此函数读取时排除换行符。

```
file_path =pkgdir(TyBase) * "/examples/resources/badpoem.txt"
fid = fopen(file_path)
line_ex = fgetl(fid)
```

```
"Oranges and lemons,"
```

要再次读取文件的第一行，请首先将读取位置指针重置到文件的开头。

```
seekstart(fid)
```

使用 fgets 函数读取文件 badpoem.txt 的第一行，此函数读取时包含换行符。

```
line_in = fgets(fid)
```

```
"Oranges and lemons,\n"
```

通过检查 fgetl 和 fgets 函数返回的行的长度，比较二者的输出。

```
length(line_ex)
```

```
19
```

```
length(line_in)
```

```
20
```

```
line_ex
```

```
"Oranges and lemons,"
```

```
line_in
```

```
"Oranges and lemons,\n"
```

关闭文件。

```
fclose(fid)
```

fopen

打开文件或获得有关打开文件的信息

A. 语法

```
fileID = fopen(filename)
fileID = fopen(filename,permission)
fileID = fopen(filename,permission,encodingIn)
```

B. 说明

fileID = fopen(filename)打开文件 filename 以便以二进制读取形式进行访问，并返回一个 IO 类型的对象。

fileID = fopen(filename,permission)将打开由 permission 指定访问类型的文件。

fileID = fopen(filename,permission,encodingIn)可选的 encodingIn 参数指定与文件相关联的字符编码方案。

C. 示例

(1). 打开文件并将标识符传递给文件 I/O 函数

打开文件并将文件标识符传递给 fgetl 函数以读取数据。

打开文件 tsunamis.txt 并获取文件标识符。

```
filename = pkgdir(TyBase) * "/examples/resources/tsunamis.txt"
fileID = fopen(filename);
```

将 fileID 传递给 fgetl 函数以从文件读取一行。

```
tline = fgetl(fileID)
```

```
"A global tsunami data set in xlsx format, comprising the following file:"
```

关闭文件。

```
fclose(fileID)
```

(2). 打开要写入的文件并指定访问类型、写入顺序、字符编码

打开一个文件，使用 Shift-JIS 字符编码写入到文件。

```
using StringEncodings
fileID = fopen("japanese_out.txt", "w", enc"Shift_JIS")
```

```
StringEncoder{UTF-8, Shift_JIS}(IOStream(<filejapanese_out.txt>))
```

"w" 输入指定写入权限，"Shift_JIS" 指定字符编码方案。

fprintf

将数据写入文本文件

A. 语法

```
fprintf(fileID,formatSpec,A1,...,An)
fprintf(formatSpec,A1,...,An)
nbytes = fprintf(___)
```

B. 说明

fprintf(fileID,formatSpec,A1,...,An) 按列顺序将 formatSpec 应用于数组 A1,...An 的所有元素，并将数据写入到一个文本文件。fprintf 使用在对 fopen 的调用中指定的编码方案。

fprintf(formatSpec,A1,...,An)设置数据的格式并在屏幕上显示结果。

nbytes = fprintf(___)使用前述语法中的任意输入参数返回 fprintf 所写入的字节数。

C. 示例

(1). 输出字面文本和数组值

将多个数值和字面文本输出到屏幕。

```
A1 = [9.9, 9900];
A2 = [8.8 7.7
     8800 7700];
formatSpec = "X is %4.2f meters or %8.3f mm\n"
fprintf(formatSpec, A1, A2)
```

```
X is 9.90 meters or 9900.000 mm
X is 8.80 meters or 8800.000 mm
X is 7.70 meters or 7700.000 mm
```

formatSpec 输入中的 %4.2f 指定输出中每行的第一个值为浮点数，字段宽度为四位数，包括小数点后的两位数。formatSpec 输入中的 %8.3f 指定输出中每行的第二个值为浮点数，字段宽度为八位数，包括小数点后的三位数。\n 为新起一行的控制字符。

(2). 将双精度值输出为整数

显式将包含分式的双精度值转换为整数值。

```
a = [1.02 3.04 5.06];
fprintf("%d\n", round.(a));
```

```
1
3
5
```

formatSpec 输入中的 %d 将向量 round(a) 中的每个值作为有符号整数输出。\n 为新起一行的控制字符。

(3). 将表格数据写入文本文件

将指数函数的短表写入到名为 exp.txt 的文本文件。

```
x = 0:0.1:1;
A = hcat(x, exp.(x))';
```

第一个对 fprintf 的调用输出标题文本 x 和 exp(x)，第二个调用输出变量 A 的值。

如果打算使用 Microsoft 记事本读取文件，请使用 '\r\n' 而不是 '\n' 移到新行。例如，将对 fprintf 的调用替换为下列项：

```
fileID = fopen(file_path, "w");
fprintf(fileID, "%6s %12s\n", "x", "exp(x)");
fprintf(fileID, "%6.2f %12.8f\n", A);
fclose(fileID);
```

Syslab 导入函数、所有 UNIX® 应用程序以及 Microsoft Word 和写字板都将 '\n' 识别为换行指示符。

通过 type 命令查看文件的内容。

```
type(file_path)
```

```
x exp(x)
0.00   1.00000000
0.10   1.10517092
0.20   1.22140276
0.30   1.34985881
0.40   1.49182470
0.50   1.64872127
0.60   1.82211880
0.70   2.01375271
0.80   2.22554093
0.90   2.45960311
1.00   2.71828183
```

(4). 获取写入文件的字节数

将数据写入文件并返回所写入的字节数。

将数据数组 A 写入文件并获取 fprintf 所写入的字节数。

```
using TyMath

A = magic(4);

file_path = "myfile.txt"
fileID = fopen(file_path, "w");
nbytes = fprintf(fileID, "%5d %5d %5d %5d\n", A)
```

```
96
```

fprintf 函数向文件中写入了 96 个字节。

关闭文件。

```
fclose(fileID);
```

通过 type 命令查看文件的内容。

```
type(file_path)
```

```
16    5    9    4
 2   11    7   14
 3   10    6   15
13    8   12    1
```

fread

读取二进制文件中的数据

A. 语法

```
A = fread(fileID)
A = fread(fileID,sizeA)
A = fread(fileID,sizeA,precision)
A = fread(fileID,sizeA,precision,skip)
A = fread(fileID,sizeA,precision,skip,machinefmt)
```

B. 说明

A = fread(fileID) 将打开的二进制文件中的数据读取到列向量 A 中，并将文件指针定位在文件结尾标记处。该二进制文件由文件标识符 fileID 指示。使用 fopen 可打开文件并获取 fileID 值。读取文件后，请调用 fclose(fileID) 来关闭文件。

A = fread(fileID,sizeA) 将文件数据读取到维度为 sizeA 的数组 A 中，并将文件指针定位到最后读取的值之后。fread 按列顺序填充 A。

A = fread(fileID,sizeA,precision) 根据 precision 描述的格式和大小解释文件中的值。sizeA 参数为可选参数。

A = fread(fileID,sizeA,precision,skip) 在读取文件中的每个值之后将跳过 skip 指定的字节或位数。sizeA 参数为可选参数。

A = fread(fileID,sizeA,precision,skip,machinefmt) 另外指定在文件中读取字节或位时的顺序。sizeA 和 skip 参数是可选的。

C. 示例

(1). 读取整个文件的 UInt8 数据

将一个九元素向量写入到示例文件 nine.bin 中。

```
file_path = "nine.bin"
fileID = fopen(file_path, "w");
fwrite(fileID, [1:9...]);
fclose(fileID);
```

将文件中的所有数据读取到 Float64 类的向量中。默认情况下，fread 一次读取文件中的 1 个字节，将每个字节解释为一个 8 位无符号整数 (UInt8)，并返回一个 Float64 数组。

```
fileID = fopen(file_path);
A = fread(fileID)
```

```
9-element Vector{Float64}:
 1.0
 2.0
 3.0
 4.0
 5.0
 6.0
 7.0
 8.0
 9.0
```

fread 返回一个列向量，文件中的每个字节对应一个元素。 查看 A 的相关信息。

```
varinfo(r"A")
```

```
name    size      summary
 - - - -  - - - - - - - - -   - - - - - - - - - - - - - - - - - - - - - - - -
A     112 bytes  9-element Vector{Float64}
```

关闭文件。

```
fclose(fileID);
```

(2). 读取包含双精度数据的整个文件

创建一个名为 doubledata.bin 并且包含九个双精度值的文件。

```
using TyMath
file_path = "doubledata.bin"
fileID = fopen(file_path, "w");
fwrite(fileID, magic(3), "Float64");
fclose(fileID);
```

打开文件 doubledata.bin，并将文件中的数据读取到一个 3×3 数组 A。指定源数据为 Float64 类。

```
fileID = fopen(file_path);
A = fread(fileID, [3 3], "Float64")
```

```
3×3 Matrix{Float64}:
8.0  1.0  6.0
3.0  5.0  7.0
4.0  9.0  2.0
```

关闭文件。

```
fclose(fileID);
```

(3). 读取文件中的选定行或列

创建一个名为 nine.bin 并且包含 1 - 9 中的值的文件。以 uint16 值形式写入数据。

```
file_path = "doubledata.bin"
fileID = fopen(file_path, "w");
fwrite(fileID, [1:9...], "UInt16");
fclose(fileID);
```

将前六个值读取到一个 3×2 数组中。指定源数据为 uint16 类。

```
fileID = fopen(file_path);
A = fread(fileID, [3, 2], "UInt16")
```

```
3×2 Matrix{Float64}:
1.0  4.0
2.0  5.0
3.0  6.0
```

fread 返回使用文件 nine.bin 中的前六个值逐列填充的一个数组。 返回到文件的开头。

```
seekstart(fileID)
```

一次读取两个值，并在读取下一个值之前跳过一个值。使用 precision 值 '2*uint16' 指定此格式。因为数据为 uint16 类，所以一个值由 2 个字节表示。因此，将 skip 参数指定为 2。

```
precision = "2*UInt16";
skip = 2;
B = fread(fileID, [2, 3], precision, skip)
```

```
2×3 Matrix{Float64}:
1.0  4.0  7.0
2.0  5.0  8.0
```

fread 返回使用 nine.bin 中的值逐列填充的一个 2×3 数组。 关闭文件。

```
fclose(fileID)
```

fscanf

读取文本文件中的数据

A. 语法

```
A = fscanf(fileID,formatSpec)
A = fscanf(fileID,formatSpec,sizeA)
```

```
A,count = fscanf(___)
```

B. 说明

A = fscanf(fileID,formatSpec) 将打开的文本文件中的数据读取到列向量 A 中，并根据 formatSpec 指定的格式解释文件中的值。fscanf 函数在整个文件中重新应用该格式，并将文件指针定位在文件结尾标记处。如果 fscanf 无法将 formatSpec 与数据相匹配，将只读取匹配的部分并停止处理。

该文本文件由文件标识符 fileID 指示。使用 fopen 打开文件，指定字符编码，以及获取 fileID 值。读取文件后，请调用 fclose(fileID) 来关闭文件。

A = fscanf(fileID,formatSpec,sizeA) 将文件数据读取到维度为 sizeA 的数组 A 中，并将文件指针定位到最后读取的值之后。fscanf 按列顺序填充 A。sizeA 必须为正整数或采用 [m n] 的形式，其中 m 和 n 为正整数。

A,count = fscanf(___) 还将返回 fscanf 读取到 A 中的字段数。对于数值数据，这是已读取的值数。您可以将此语法与前面语法中的任何输入参数结合使用。

C. 示例

(1). 将文件内容读取到列向量中

创建一个包含浮点数的示例文本文件。

```
x = [81.4724
    90.5792
    12.6987
     91.3376
    63.2359
    9.7540
    27.8498
    54.6882]
file_path = "nums1.txt"
fileID = fopen(file_path, "w");
fprintf(fileID, "%4.4f\n", x);
fclose(fileID);
```

查看文件的内容。

```
type(file_path)
```

```
81.4724
90.5792
```

```
12.6987
91.3376
63.2359
9.7540
27.8498
54.6882
```

打开要读取的文件并获取文件标识符 fileID。

```
fileID = fopen(file_path, "r");
```

定义要读取的数据的格式。使用 '%f' 指定浮点数。

```
formatSpec = "%f";
```

读取文件数据并按列顺序填充输出数组 A。fscanf 在整个文件中重新应用格式 formatSpec。

```
A = fscanf(fileID, formatSpec)
```

```
 8-element Vector{Float64}:
 81.4724
 90.5792
 12.6987
 91.3376
 63.2359
  9.754
 27.8498
 54.6882
```

A 是一个包含该文件中的数据的列向量。

关闭文件。

```
fclose(fileID);
```

(2). 将文件内容读取到数组中

创建一个包含整数和浮点数的示例文本文件。

```
x = 1:1:5;
y = [1.0000 2.0000 3.0000 4.0000 5.0000
    0.8147 0.9058 0.1270 0.9134 0.6324]
file_path = "nums2.txt"
fileID = fopen(file_path, "w");
fprintf(fileID, "%d %4.4f\n", y);
fclose(fileID);
```

查看文件的内容。

```
type(file_path)
```

```
1 0.8147
```

```
2 0.9058
3 0.1270
4 0.9134
5 0.6324
```

打开要读取的文件并获取文件标识符 fileID。

```
fileID = fopen(file_path, "r");
```

定义要读取的数据的格式和输出数组的形状。

```
formatSpec = "%d %f";
sizeA = [2 Inf];
```

读取文件数据并按列顺序填充输出数组 A。fscanf 在整个文件中重新使用格式 formatSpec。

```
A = fscanf(fileID, formatSpec, sizeA)
```

```
2×5 Matrix{Float64}:
 1.0    2.0    3.0    4.0    5.0
 0.8147  0.9058  0.127  0.9134  0.6324
```

```
fclose(fileID);
```

转置该数组，以便 A 与文件中数据的方向匹配。

```
A = A'
```

```
5×2 adjoint(::Matrix{Float64}) with eltype Float64:
 1.0  0.8147
 2.0  0.9058
 3.0  0.127
 4.0  0.9134
 5.0  0.6324
```

(3). 跳过文件中的特定字符

跳过示例文件中的特定字符并仅返回数值数据。

创建一个包含温度值的示例文本文件。

```
str = "78° C 72° C 64° C 66° C 49° C";
file_path = "temperature.dat"
fileID = fopen(file_path, "w");
fprintf(fileID, "%s", str);
fclose(fileID);
```

读取文件中的数字并跳过文本 ° C。此外，还返回 fscanf 读取的值数。扩展的 ASCII 代码 176 表示度数符号。

```
fileID = fopen(file_path, "r");
degrees = Char(176);
```

```
A, count = fscanf(fileID, string("%d", degrees, "C"), nargout = 2)
A =
78
72
64
66
49

conut = 5
fclose(fileID);
```

A 是一个包含文件中的数值的向量。count 指示 fscanf 读取五个值。

fwrite

将数据写入二进制文件

A. 语法

```
A = fwrite(fileID,A)
A = fwrite(fileID,A,precision)
A = fwrite(fileID,A,precision,skip)
A = fwrite(fileID,A,precision,skip,machinefmt)
```

B. 说明

A = fwrite(fileID,A) 将数组 A 的元素按列顺序以 8 位无符号整数的形式写入一个二进制文件。该二进制文件由文件标识符 fileID 指示。使用 fopen 可打开文件并获取 fileID 值。完成写入后，请调用 fclose(fileID) 来关闭文件。

A = fwrite(fileID,A,precision) 按照 precision 说明的形式和大小写入 A 中的值。

A = fwrite(fileID,sizeA,precision,skip)在写入每个值之前跳过 skip 指定的字节数。

A = fwrite(fileID,sizeA,precision,skip,machinefmt) 另外还指定将字节写入文件的顺序。skip 参数为可选参数。

C. 示例

(1). 将 uint8 数据写入二进制文件

打开名称为 nine.bin 的文件以便写入。在对 fopen 的调用中使用 "w" 指

定写访问权限。

```
file_path = "nine.bin"
fileID = fopen(file_path, "w");
```

fopen 返回文件标识符 fileID。

将从 1 到 9 的整数以 8 位无符号整数的形式写入。

```
fwrite(fileID, [1:9...]);
```

关闭文件。

```
fclose(fileID);
```

(2). 将 4 字节整数写入二进制文件

打开名称为 magic5.bin 的文件以便写入。

```
using TyMath
file_path = "magic5.bin"
fileID = fopen(file_path, "w");
```

写入 5×5 幻方矩阵的 25 个元素。使用 precision 参数 'Int32' 可以写入 4 字节整数。

```
fwrite(fileID, magic(5), "Int32");
```

关闭文件。

```
fclose(fileID);
```

(3). 将数据附加到二进制文件

写入一个包含 4×4 幻方矩阵、以双精度浮点数形式存储元素的二进制文件。

```
using TyMath

file_path = "magic4.bin"
fileID = fopen(file_path, "w");
fwrite(fileID, magic(4), "Float64");
fclose(fileID);
```

以写访问权限打开文件 magic4.bin，以便向文件追加内容。在 fopen 调用中指定文件访问类型 'a'。

```
fileID = fopen(file_path, "a");
```

将一个由零组成的 4×4 矩阵追加到文件中。然后关闭文件。

```
fwrite(fileID, zeros(4, 4), "Float64");
fclose(fileID);
```

(4). 按 Big-Endian 字节排序写入二进制文件

将随机的双精度数值写入名称为 myfile.bin 的文件，以便在 big-endian 系统上使用。在 fwrite 调用中指定 'ieee-be' 的 machinefmt 值，以便指示 big-endian 字节排序。

```
file_path = "myfile.bin"
fileID = fopen(file_path, "w");
fwrite(fileID, magic(4), "Float64", "ieee-be");
fclose(fileID);
```

3.1.3.1.3 标准文件格式

函数名	简介
h5info	有关 HDF5 文件的信息
h5read	从 HDF5 数据集读取数据
h5readatt	从 HDF5 文件中读取属性
importdata	从文件加载数据
textscan	从文本文件或字符串读取格式化数据

h5info

有关 HDF5 文件的信息

A. 语法

```
info = h5info(filename)
info = h5info(filename,loc)
```

B. 说明

info=h5info(filename)返回有关整个 HDF5 文件的信息，包括有关其中包含的组、数据集和命名数据类型的信息。

info=h5info(filename，loc)返回有关 HDF5 文件中指定位置的信息。

C. 示例

(1). 返回有关整个文件的信息

返回 example.h5 整个文件的有关信息。

```
Import TyBase
dir_path = pkgdir(TyBase)

fn = dir_path * "/examples/Data import and analysis/h5info/test.h5"
file = h5info(fn)
fclose(file)
```

```
HDF5.File: (read-only) C:\Users\Public\TongYuan\.julia\dev\TyBase/test/Data import and analysis/ h5info/test.h5
├─ attr1
├─ attr2
├─ d
├─ dt
├─ g1
│  └─ time
└─ g4
   └─ lat
```

(2). 返回有关组的信息

返回 g4 组的有关信息。

```
import TyBase
dir_path = pkgdir(TyBase)

fn = dir_path * "/examples/Data import and analysis/h5info/test.h5"
file = h5info(fn, "g4")
fclose(file)
```

```
HDF5.Group: /g4 (file: C:\Users\Public\TongYuan\.julia\dev\TyBase/test/Data import and analysis/h5info/test.h5)
└─ lat
```

(3). 返回有关组中数据集的信息。

返回 g4 组中数据集 time 的有关信息。

```
import TyBase
dir_path = pkgdir(TyBase)

fn = dir_path * "/examples/Data import and analysis/h5info/test.h5"
file = h5info(fn, "g4/lat")
fclose(file)
```

```
HDF5.Dataset: /g4/lat (file: C:\Users\Public\TongYuan\.julia\dev\TyBase/test/Data import and analysis/h5info/test.h5 xfer_mode: 0)
```

h5read

从 HDF5 数据集读取数据

A. 语法

```
data = h5read(filename,ds)
data = h5read(filename,ds,start,count)
data = h5read(filename,ds,start,count,stride)
```

B. 说明

data = h5read(filename，ds)读取 HDF5 文件 filename 中包含的数据集 ds 中的所有数据。

data = h5read(filename，ds，start，count)在数据集中从 start 指定的位置开始读取数据子集。count 参数指定沿每个维度读取的元素数。

data = h5read(filename，ds，start，count，stride) 以 stride 所指定的数据集各维度索引间隔返回数据子集。

C. 示例

(1). 读取整个 HDF5 数据集

从 HDF5 文件中获取数据集的元数据，然后读取数据集。

显示来自 HDF5 文件 example.h5 的数据集 /g4/lat 的元数据。

```
import TyBase
dir_path = pkgdir(TyBase) * "/examples/Data import and analysis/h5read"

data = h5read(dir_path * "/example.h5", "/g4/lat")
```

```
19-element Vector{Float64}:
 -90.0
 -80.0
 -70.0
 -60.0
   ⋮
  70.0
  80.0
  90.0
```

(2). 读取 HDF5 数据集的子集

从 HDF5 文件中获取数据集的元数据，然后读取数据集的子集。

显示来自 HDF5 文件 example.h5 的数据集 /g4/world 的元数据。

```
start = 1:5;
count = 1:3;
import TyBase
dir_path = pkgdir(TyBase) * "/examples/Data import and analysis/h5read"
data = h5read(dir_path * "/example.h5", "/g4/world", (start, count))
```

```
5×3 Matrix{Float64}:
 0.0  0.0  0.0
 0.0  0.0  0.0
 0.0  0.0  0.0
 0.0  0.0  0.0
 0.0  0.0  0.0
```

(3). 3-读取具有指定索引间距的 HDF5 数据集

从数据集中读取数据，读取数据时在每个维度上均按指定的数据集索引间距进行采样。

首先，显示来自 HDF5 文件 example.h5 的数据集 /g4/lon 的元数据。数据集中的变量有一个维度包含 36 个元素。

```
start = 1:5;
count = 1:1;
import TyBase
dir_path = pkgdir(TyBase) * "/examples/Data import and analysis/h5read"
data = h5read(dir_path * "/example.h5", "/g4/lon", (1:2:36,))
```

```
18-element Vector{Float64}:
 -180.0
 -160.0
 -140.0
 -120.0
    ⋮
  120.0
  140.0
  160.0
```

h5readatt

从 HDF5 文件中读取属性

A. 语法

```
data = h5readatt(filename,loc,attr)
```

B. 说明

attval = h5readatt（ filename，loc，attr ）从 HDF5 文件中的指定位置检索指定属性的值。

C. 示例

(1). 读取组属性

从 HDF5 文件 example.h5 的根组中读取属性 attr2。

```
import TyBase
dir_path = pkgdir(TyBase) * "/examples/Data import and analysis/h5readatt"
data = h5readatt(dir_path * "/example.h5", "/")data["attr2"]
```

```
2×2 Matrix{Int32}:
 0  2
 1  3
```

(2). 读取数据集属性

从 HDF5 文件 example.h5 中的数据集 lon 中读取属性 units。

```
import TyBase
dir_path = pkgdir(TyBase) * "/examples/Data import and analysis/h5readatt"
data = h5readatt(dir_path * "/example.h5", "/g4/lon")
data["units"]
```

```
"degrees_east"
```

importdata

从文件加载数据

A. 语法

```
A = importdata(filename)
```

```
A = importdata("-pastespecial")
A = importdata(___,delimiterIn)
A = importdata(___,delimiterIn,headerlinesIn)
```

B. 说明

A= importdata(filename) 将数据加载到数组 A 中。

A = importdata("-pastespecial")从系统剪贴板而不是文件加载数据。

A =importdata(___,delimiterOut) 将 delimiterIn 解释为 ASCII 文件 filename 或剪贴板数据中的列分隔符。您可以将 delimiterIn 与以上语法中的任何输入参数结合使用。

A = importdata(___,delimiterOut,headerlinesIn) 从 ASCII 文件 filename 或剪贴板加载数据，并读取从第 headerlinesIn+1 行开始的数值数据。

注意

在 CentOS 系统下，若使用 importdata("-pastespecial") 从系统剪切板加载数据，系统需安装 xsel 或 xclip 库。

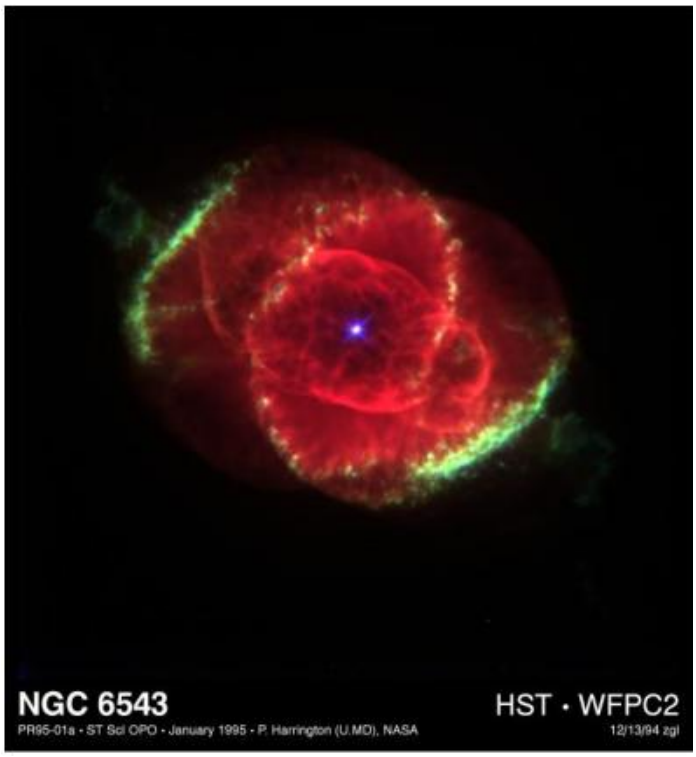
C. 示例

(1). 导入并显示图像

导出并显示示例图像 ngc6543a.jpg。

```
using TyImages

file_path = pkgdir(TyBase) * "/examples/resources/ngc6543a.jpg"
A = imread(file_path);
image(A)
```



使用 imread 导入图像文件。

(2). 导入文本文件并指定分隔符和列标题

使用文本编辑器创建一个带有列标题的称为 myfile01.txt 的空格分隔 ASCII 文件。

```
Day1 Day2 Day3 Day4 Day5 Day6 Day7
95.01 76.21 61.54 40.57 5.79 20.28 1.53
23.11 45.65 79.19 93.55 35.29 19.87 74.68
60.68 1.85 92.18 91.69 81.32 60.38 44.51
48.60 82.14 73.82 41.03 0.99 27.22 93.18
89.13 44.47 17.63 89.36 13.89 19.88 46.60
```

导入该文件并指定空格分隔符和单个列标题。

```
filename = pkgdir(TyBase) * "/examples/resources/myfile01.txt"
delimiterIn = ' ';
headerlinesIn = 1;
A = importdata(filename, delimiterIn, headerlinesIn);
```

查看第 3 列和第 5 列。

```
for k in [3, 5]
   display(A["colheaders"][k])
   display(A["data"][:, k])
   println()
end
```

```
"Day3"
```

```
5-element Vector{Float64}:
 61.54
 79.19
 92.18
 73.82
 17.63

"Day5"

5-element Vector{Float64}:
  5.79
 35.29
 81.32
  0.99
 13.89
```

(3). 导入文本文件并返回检测到的分隔符

使用文本编辑器创建一个称为 myfile02.txt 的逗号分隔 ASCII 文件。

```
1,2,3
4,5,6
7,8,9
```

导入该文件并显示输出数据和检测到的分隔符。

```
filename = pkgdir(TyBase) * "/examples/resources/myfile02.txt"
A, delimiterOut = importdata(filename; nargout = 2)
```

```
A = 3×3 Matrix{Float64}:
 1.0  2.0  3.0
 4.0  5.0  6.0
 7.0  8.0  9.0

delimiterOut = ',': ASCII/Unicode U+002C (category Po: Punctuation,
other)
```

(4). 从剪贴板导入数据

将以下行复制到剪贴板。选择相应文本，右键点击，然后选择复制。

```
1,2,3
4,5,6
7,8,9
```

键入以下内容将剪贴板数据导入到 Syslab 中。

```
A = importdata("-pastespecial")
```

```
3×3 Matrix{Float64}:
```

```
1.0  2.0  3.0
4.0  5.0  6.0
7.0  8.0  9.0
```

textscan

从文本文件或字符串读取格式化数据

A. 语法

```
C = textscan(fileID,formatSpec)
C = textscan(fileID,formatSpec,N)
C = textscan(chr,formatSpec)
C = textscan(chr,formatSpec,N)
C = textscan(___,Key=Value)
C,position = textscan(___)
```

B. 说明

C = textscan(fileID,formatSpec) 将已打开的文本文件中的数据读取到数组 C。该文本文件由文件标识符 fileID 指示。使用 fopen 可打开文件并获取 fileID 值。完成文件读取后，请调用 fclose(fileID) 来关闭文件。

textscan 尝试将文件中的数据与 formatSpec 中的转换设定符匹配。textscan 函数在整个文件中按 formatSpec 重复扫描数据，直至 formatSpec 找不到匹配的数据时才停止。

C = textscan(fileID,formatSpec,N) 按 formatSpec 读取文件数据 N 次，其中 N 是一个正整数。要在 N 个周期后从文件读取其他数据，请使用原 fileID 再次调用 textscan 进行扫描。如果通过调用具有相同文件标识符 (fileID) 的 textscan 恢复文件的文本扫描，则 textscan 将在上次终止读取的点处自动恢复读取。

C = textscan(chr,formatSpec) 将字符 chr 中的文本读取到数组 C 中。从字符向量读取文本时，对 textscan 的每一次重复调用都会从开头位置重新开始扫描。要从上次位置重新开始扫描，需要指定 position 输出参数。

textscan 尝试将字符 chr 中的数据与 formatSpec 中指定的格式匹配。

C = textscan(chr,formatSpec,N) 按 formatSpec N 次，其中 N 是一个正整数。

C = textscan(___,Key=Value) 使用一个或多个 Name,Value 对组参数以及上述语法中的任何输入参数来指定选项。

C,position = textscan(___) 在扫描结束时返回文件或字符向量中的位置作为第二个输出参数。对于文件，该值等同于调用 textscan 后再运行 ftell(fileID) 所返回的值。对于字符向量，position 指示 textscan 读取了多少个字符。

C. 示例

(1). 读取浮点数

读取包含浮点数的字符向量。

```
chr = "0.41 8.24 3.57 6.24 9.27"
C, pos = textscan(chr, "%f")
```

formatSpec 中的设定符 "%f" 指示 textscan 将 chr 中的每个字段与一个双精度浮点数进行匹配。

显示数组 C 的内容。

```
C[1]
```

```
5-element Vector{Float64}:
0.41
8.24
3.57
6.24
9.27
```

读取相同字符向量，将每个值截短至一位小数。

```
C, pos = textscan(chr, "%3.1f %*1d")
```

设定符 %3.1f 指示字段宽度为 3 位数，精度为 1。textscan 函数读取全部 3 位数，包括小数点和小数点后的 1 位数。设定符 %*1d 指示 textscan 跳过其余位数。

显示数组 C 的内容。

```
C[1]
```

```
5-element Vector{Float64}:
0.4
8.2
3.5
6.2
9.2
```

(2). 读取十六进制数

读取一个表示一组十六进制数的字符向量。表示十六进制数的文本包括数字 0-9，字母 a-f 或 A-F，以及可选的前缀 0x 或 0X。

要将 hexnums 中的字段与十六进制数匹配，请使用 "%x" 设定符。textscan 函数将字段转换为无符号 64 位整数。

```
hexnums = "0xFF 0x100 0x3C5E A F 10"
C, pos = textscan(hexnums, "%x")
```

```
(Vector{UInt64}[[0x00000000000000ff, 0x0000000000000100,
0x0000000000003c5e, 0x000000000000000a, 0x000000000000000f,
0x0000000000000010]], 24)
```

将 C 的内容显示为行向量。

```
C[1]
```

```
6-element Vector{UInt64}:
0x00000000000000ff
0x0000000000000100
0x0000000000003c5e
0x000000000000000a
0x000000000000000f
0x0000000000000010
```

您可以将字段转换为 8、16、32 或 64 位的有符号或无符号整数。要将 hexnums 中的字段转换为有符号 32 位整数，请使用 "%xs32" 设定符。

```
C, pos = textscan(hexnums, "%xs32")
C[1]
```

```
6-element Vector{Int32}:
255
256
15454
10
15
16
```

您还可以指定用来解释输入的字段宽度。在这种情况下，前缀将计入字段宽度。例如，如果您将字段宽度设置为三，如在 %3x 中所示，则 textscan 将文本 "0xAF 100" 拆分为三个文本片段，即 "0xA"、"F" 和 "100"。它将这三个文本片段视为不同的十六进制数。

```
C, pos = textscan("0xAF 100", "%3x")
C[1]
```

```
3-element Vector{UInt64}:
0x000000000000000a
0x000000000000000f
0x0000000000000100
```

(3). 读取二进制数

读取一个表示一组二进制数的字符向量。表示二进制数的文本包括数字 0

和 1，以及可选的前缀 0b 或 0B。

要将 binnums 中的字段与二进制数匹配，请使用 "%b" 设定符。textscan 函数将字段转换为无符号 64 位整数。

```
binnums = "0b101010 0b11 0b100 1001 10"
C, pos = textscan(binnums, "%b")
```

```
(Vector{UInt64}[[0x002a, 0x0003, 0x0004, 0x0009, 0x0002]], 27)
```

将 C 的内容显示为行向量。

```
C[1]
```

```
5-element Vector{UInt64}:
0x002a
0x0003
0x0004
0x0009
0x0002
```

您可以将字段转换为 8、16、32 或 64 位的有符号或无符号整数。要将 binnums 中的字段转换为有符号 32 位整数，请使用 "%bs32" 设定符。

```
C, pos = textscan(binnums, "%bs32")
C[1]
```

```
5-element Vector{Int32}:
42
3
4
9
2
```

您还可以指定用来解释输入的字段宽度。在这种情况下，前缀将计入字段宽度。例如，如果您将字段宽度设置为三，如在 %3b 中所示，则 textscan 将文本 "0b1010 100" 拆分为三个文本片段，即 "0b1"、"010" 和 "100"。它将这三个文本片段视为不同的二进制数。

```
C, pos = textscan("0b1010 100", "%3b")
C[1]
```

```
3-element Vector{UInt64}:
0x0001
0x0002
0x0004
```

3.1.3.1.4 工作区变量

函数名	简介

clear	从工作区中删除项目、释放系统内存
load	将文件变量加载到工作区中
save	将工作区变量保存到文件中

clear

从工作区中删除项目、释放系统内存

A. 语法

```
clear()

clear(name1, name2, ..., nameN)
```

B. 说明

clear 从当前工作区中删除所有变量，并将它们从系统内存中释放。

clear(name1, name2, ..., nameN) 删除内存中的变量 name1 ... nameN。

C. 示例

(1). 清除单个变量

定义两个变量 a 和 b，然后清除 a。

```
a = 1;
b = 2;
clear(:a)
```

仅变量 b 保留在工作区中。

```
varinfo()
```

```
name          size         summary
 - - - - - - - - -    - - - - - - - - - - -    - - - - - - - - - - - - - - - - -
Base                       Module
Core                       Module
InteractiveUtils  274.640 KiB     Module
Main                       Module
a              0 bytes     Nothing
ans            0 bytes     Nothing
b              8 bytes     Int64
```

(2). 清除变量集

创建一个包含要清除的变量的名称的数组 vars。然后，清除这些变量。

```
vars = ["v1", "v2", "time"];
clear(:vars)
```

load

将文件变量加载到工作区中

A. 语法

```
load(filename)
load(filename,variables)
load(filename,"-ascii")
load(filename,"-mat")
load(filename,"-mat",variables)
S = load(___)
```

B. 说明

load(filename) 从 filename 加载数据。

如果 filename 是 MAT 文件，load(filename) 会将 MAT 文件中的变量加载到 Syslab 工作区。

如果 filename 是 ASCII 文件，load(filename) 会创建一个包含该文件数据的双精度矩阵。

load(filename,variables) 加载 MAT 文件 filename 中的指定变量。

load(filename,"-ascii") 将 filename 视为 ASCII 文件，而不管文件扩展名如何。

load(filename,"-mat") 将 filename 视为 MAT 文件，而不管文件扩展名如何。

load(filename,"-mat",variables) 加载 filename 中的指定变量。

S = load(___) 使用前面语法组中的任意输入参数将数据加载到 S 中。示例如果 filename 是 MAT 文件，则 S 是字典。

如果 filename 是 ASCII 文件，则 S 是包含该文件数据的双精度矩阵。

C. 示例

(1). 加载 MAT 文件中的所有变量

加载示例 MAT 文件 gong.mat 中的所有变量。请在加载操作前后检查工作区的内容。

```
display("Contents of workspace before loading file:")

display("Contents of gong.mat:")

filename = pkgdir(TyBase) * "/examples/resources/gong.mat"

load(filename)
display("Contents of workspace after loading file:")
varinfo()
```

您也可以使用命令语法加载变量。清除以前加载的变量，并重复 load 操作。

```
clear([:y, :Fs])

load(filename)
```

(2). 加载 MAT 文件中的特定变量

仅加载示例文件 handel.mat 中的变量 y。如果工作区已包含变量 y，load 操作将使用文件中的数据来覆盖它。

```
filename = pkgdir(TyBase) * "/examples/resources/handel.mat"
load(filename, "y")
```

(3). 使用正则表达式加载特定变量

示例文件 accidents.mat 的内容如下。

```
Name   Size   Bytes   Class   Attributes
datasources   3x1   2724   cell
hwycols   1x1   8   double
hwydata   51x17   6936   double
hwyheaders   1x17   2758   cell
hwyidx   51x1   408   double
hwyrows   1x1   8   double
statelabel   51x1   6596   cell
ushwydata   1x17   136   double
uslabel   1x1   138   cell
```

使用函数语法加载文件中所有名称不以 "hwy" 开头的变量。

```
filename = pkgdir(TyBase) * "/examples/resources/accidents.mat"
load(filename, regexp = r"^(?!hwy)...")
```

(4). 将变量列表加载到字典中

文件 durer.mat 包含变量 X、caption 和 map。创建要加载的变量名称数组。

```
filename = pkgdir(TyBase) * "/examples/resources/durer.mat"
myVars = ["X", "caption"];
S = load(filename, myVars...)
```

```
Dict{String, Any} with 2 entries:
  "X"       => [41.0 60.0 … 67.0 39.0; 47.0 82.0 … 106.0 76.0; … ;
66.0 76.0 … 99.0 95.0; 49.0 66.0 … 68.0 69…
  "caption" => ["Albrecht Durer's Melancolia.", "Can you find the
matrix?"]
```

仅将变量 X 和 caption 加载到字典 S 中。

(5). 加载 ASCII 文件

为多个 4 列矩阵创建一个 ASCII 文件，并将数据重新加载回双精度矩阵之中。

```
using TyMath

a = magic(4);
b = ones(2, 4) * -5.7;
c = [8 6 4 2];

filename = "mydata.mat"
save(filename, "a", "b", "c", "-ascii")
clear([:a, :b, :c])

load(filename, "-ascii")
```

load 创建名为 mydata 的 Float64 类型的矩阵。

查看 mydata 的相关信息。

```
varinfo(r"mydata")
```

```
name       size summary
------- ---------- --------------------
mydata 264 bytes 7×4 Matrix{Float64}
```

save

将工作区变量保存到文件中

A. 语法

```
save(filename)
save(filename,variables)
save(filename,variables,fmt)
save(filename,variables,"-append")
save(filename,variables,"-append","-nocompression")
```

B. 说明

save(filename) 将当前工作区中的所有变量保存在 Syslab 格式的二进制文件（jld2 文件）filename 中。如果 filename 已存在，save 会覆盖该文件。示例 save(filename,variables) 仅保存 variables 指定的变量。

save(filename,variables,fmt) 以 fmt 指定的文件格式保存。variables 参数为可选参数。如果您不指定 variables，save 函数将保存工作区中的所有变量。

save(filename,variables,"append") 将新变量添加到一个现有文件中。如果 MAT 文件中已经存在变量，则 save 会使用工作区中的值覆盖它。对于 ASCII 文件，"-append" 会将数据添加到文件末尾。

save(filename,variables,"-append","-nocompression") 将新变量添加到一个现有文件中，而不进行压缩。

C. 示例

(1). 将所有工作区变量保存到 jld2 文件

将工作区中的所有变量保存在二进制 MAT 文件 test.mat 中。如果 filename 为变量，则使用函数语法。

```
filename = "test.jld2"
save(filename)
```

从工作区中删除变量并通过 load 函数获取数据。

```
clear()
load("test.jld2")
```

(2). 将特定变量保存到 jld2 文件

创建两个变量 p 和 q 并将其保存到名为 pqfile.jld2 的文件中。

```
p = 1:10
q = ones(10, 10)
filename = "pqfile.jld2"
save(filename, "p", "q")
```

Syslab 将这些变量保存到当前文件夹中的文件 pqfile.jld

(3). 将数据保存到 ASCII 文件

创建两个变量，将其保存到 ASCII 文件中，然后查看文件内容。

```
p = 1:10
q = ones(10, 10)
filename = "pqfile.txt"
save(filename, "p", "q", "-ascii")
type(filename)
```

type 函数显示文件内容。

(4). 将变量追加到 jld2 文件中

将两个变量保存到 MAT 文件中。然后，将第三个变量追加到同一文件中。

```
p = 1:10
q = ones(10,10)
filename = "test.jld2"
save(filename, "p", "q")
```

创建一个新变量 a，并将其追加到 jld2 文件中。

```
a = 50;
save(filename, "a", "-append")
```

3.1.3.2 描述性统计量

范围、集中趋势、标准差、方差、相关性

函数名	简介
maxk	计算数组的 k 个最大元素
median	数组的中位数值
mink	计算数组的 k 个最小元素
mode	数组中出现次数最多的值
movsum	移动总和

3.1.3.2.1 maxk

计算数组的 k 个最大元素

A. 语法

```
B = maxk(A,k)
B = maxk(A,k,dims)
B = maxk(___;comparisonmethod, nargout)
B,I = maxk(___)
```

B. 说明

B = maxk(A,k) 返回 A 的 k 个最大元素。

如果 A 是向量，则 maxk 返回一个向量，其中包含 A 的 k 个最大元素。

如果 A 是矩阵，则 maxk 返回一个矩阵，此矩阵的列中包含 A 的每一列中的 k 个最大元素。

如果 A 是多维数组，则 maxk 返回大小不等于 1 的第一个维度上的 k 个最大元素。

B = maxk(A,k,dims) 确定 A 在维度 dims 上的 k 个最大元素。

B = maxk(___;comparisonmethod, nargout) comparisonmethod 为上述任何语法指定如何比较 A 的元素，nargout 控制函数返回的输出参数个数，皆为可选项。例如，maxk(A,k,comparisonmethod = "abs") 根据绝对值返回 A 的 k 个最大元素。

B,I = maxk(___) 计算 A 的 k 个最大值的索引，并在 I 中返回这些索引。

C. 示例

(1). 最大向量元素

计算向量的 3 个最大元素。

```
A = 1:10;
B = maxk(A, 3)
```

```
B = 3-element Vector{Int64}:
10
9
8
```

(2). 矩阵行的最大元素

计算矩阵中每一行的 3 个最大元素。

```
A = magic(5)
```

```
A = 5×5 Matrix{Int64}:
17  24   1   8  15
```

```
23   5   7  14  16
4    6  13  20  22
10  12  19  21   3
11  18  25   2   9
```

```
B = maxk(A, 3, 2)
```

```
B = 5×3 Matrix{Int64}:
24  17  15
23  16  14
22  20  13
21  19  12
25  18  11
```

(3). 复数向量

根据模计算复数向量的 2 个最大元素，并返回它们在输入向量中的索引。

```
A = [2 - 2im 5 + im -7 - 3im -1 + im]
```

```
A = 1×4 Matrix{Complex{Int64}}
2-2im  5+1im  -7-3im  -1+1im
```

```
B, I = maxk(A, 2, comparisonmethod = "abs", nargout = 2)
```

```
B = 2-element Vector{Complex{Int64}}
-7 - 3im
 5 + 1im

I = 2-element view(::Vector{Int64}, 1:2) with eltype Int64
3
2
```

3.1.3.2.2 median

数组的中位数值

A. 语法

```
M = median(A)
M = median(A,dims)
```

B. 说明

M = median(A) 返回 A 的中位数值。

如果 A 为向量，则 median(A) 返回 A 的中位数值。

如果 A 为非空矩阵，则 median(A) 将 A 的各列视为向量，并返回中位数

值的行向量。

如果 A 为多维数组，则 median(A) 将沿大小不等于 1 的第一个数组维度的值视为向量。此维度的大小将变为 1，而所有其他维度的大小保持不变。

median 本身在 A 的数值类中计算，例如 typeof(M) = typeof(A)。

M = median(A,dims) 返回维度 dim 上元素的中位数。例如，如果 A 为矩阵，则 median(A,2) 是包含每一行的中位数值的列向量。

C. 示例

(1). 矩阵列的中位数

定义一个 4×3 矩阵。

```
using Statistics
A = [0 1 1; 2 3 2; 1 3 2; 4 2 2]
```

```
A =
0  1  1
2  3  2
1  3  2
4  2  2
```

计算每一列的中位数值。

```
M = median(A, dims = 1)
```

```
M =
1.5  2.5  2.0
```

对于每一列，中位数值在排列顺序上位于中间的两个数值的均值。

(2). 矩阵行的中位数

定义一个 2×3 矩阵。

```
using Statistics
A = [0 1 1; 2 3 2]
```

```
A =
0    1    1
2    3    2
```

计算每一行的中位数值。

```
M = Int64.(median(A, dims = 2))
```

```
M =
1
2
```

(3). 三维数组的中位数

创建一个由 1 和 10 之间的整数组成的 1×3×4 数组。

```
using Statistics
A = zeros(1, 3, 4)
A[:, :, 1] = [10 8 10]
A[:, :, 2] = [6 9 5]
A[:, :, 3] = [9 6 1]
A[:, :, 4] = [4 9 5]
```

```
A[:,:,1] =
10    8   10

A[:,:,2] =
6    9    5

A[:,:,3] =
9    6    1

A[:,:,4] =
4    9    5
```

沿第二个维度计算此三维数组的中位数值。

```
M = Int64.(median(A, dims = 2))
```

```
M[:,:,1] =
10

M[:,:,2] =
6

M[:,:,3] =
6

M[:,:,4] =
5
```

此运算通过沿第二个维度计算三个值的中位数来生成一个 1×1×4 数组。第二个维度的大小降到 1。

沿 A 的第一个维度计算中位数。

```
M = Int64.(median(A, dims = 1));
isequal(A, M)
```

```
ans = true
```

(4). 数组页面的中位数

创建一个三维数组并计算每页数据（行和列）的中位数。

```
using Statistics
A = zeros(2, 2, 3)
A[:, :, 1] = [2 4; -2 1];
A[:, :, 2] = [6 2; -5 3];
A[:, :, 3] = [4 4; 7 -3];
M1 = [median(A[:, :, i]) for i = 1:size(A)[end]]
```

```
M1 =
1.5
2.5
4.0
```

要计算一个数组的所有维度上的中位数，可以在向量维参数中指定每个维度

```
M2 = median(A[:, :, :])
```

```
M2 = 2.5
```

```
Mall = median(A)
```

```
Mall = 2.5
```

(5). 8 位整数数组的中位数息

定义一个 3×4 矩阵。

```
using Statistics
A = Int8[1, 2, 3, 4]
```

```
A =
1
2
3
4
```

计算中位数值。

```
M = median(A)
M = Int8(ceil(M))
```

```
M =
3
```

```
typeof(M)
```

```
typeof(M) =
Int8
```

M 为在排列顺序上位于中间的两个数值的均值（以一个 8 位整数返回）。

(6). 不包括 NaN 的中位数

创建一个向量并计算其中位数，不包括 NaN 值。

```
using Statistics
A = [1.77 -0.005 3.98 -2.95 NaN 0.34 NaN 0.19];
```

```
M = median(A[.!isnan.(A)])
```

```
M =
0.265
```

3.1.3.2.3 mink

计算数组的 k 个最小元素

A. 语法

```
B = mink(A,k)
B = mink(A,k,dims)
B = mink(___;comparisonmethod = "auto", nargout = 1)
B,I = mink(___)
```

B. 说明

B = mink(A,k) 返回 A 的 k 个最小元素。

如果 A 是向量，则 mink 返回一个向量，其中包含 A 的 k 个最小元素。

如果 A 是矩阵，则 mink 返回一个矩阵，该矩阵的列包含 A 的每一列中的 k 个最小元素。

如果 A 是多维数组，则 mink 返回大小不等于 1 的第一个维度上的 k 个最小元素。

B = mink(A,k,dim) 确定 A 在维度 dim 上的 k 个最小元素。

B = mink(___,Method) 为上述任何语法指定如何比较 A 的元素，此项为可选项。例如，mink(A,k,"abs") 根据绝对值返回 A 的 k 个最小元素。

B,I = mink(___;Index="true") 计算 A 的 k 个最小值的索引，并在 I 中返回这些索引。

C. 示例

(1). 最小向量元素

计算向量的 3 个最小元素。A = 1:10;

```
B = TyBase.mink(A,3)
```

```
3-element Vector{Int64}:
  1
  2
  3
```

(2). 矩阵行的最小元素

计算矩阵中每一行的 3 个最小元素。

```
A = magic(5)
```

```
5×5 Matrix{Int64}:
 17  24   1   8  15
 23   5   7  14  16
  4   6  13  20  22
 10  12  19  21   3
 11  18  25   2   9
```

```
B = TyBase.mink(A,3,2)
```

```
5×3 Matrix{Int64}:
 1   8  15
 5   7  14
 4   6  13
 3  10  12
 2   9  11
```

(3). 复数向量

根据模计算复数向量的 2 个最小元素，并返回它们在输入向量中的位置的索引。

```
A = [2-2im 5+im -7-3im -1+im]
```

```
1×4 Matrix{Complex{Int64}}:
 2-2im  5+1im  -7-3im  -1+1im
```

```
B,I = TyBase.mink(A,2,comparisonmethod="abs";nargout=2)
```

```
 B = 2-element Vector{Complex{Int64}}:
   -1 + 1im
    2 - 2im
I = I = 2-element view(::Vector{Int64}, 1:2) with eltype Int64:
   4
   1
```

3.1.3.2.4 mode

数组中出现次数最多的值

A. 语法

```
M = mode(A)
```

```
M = mode(A; all,nargout)
M = mode(A, dims)
M = mode(A, vecdim)
M,F,C = mode(___)
```

B. 说明

M = mode(A) 返回 A 的样本众数，即 A 中出现次数最多的值。如果有多个值以相同的次数出现，mode 将返回其中最小的值。对复杂的输入，最小值是排序列表的第一个值。

如果 A 为向量，则 mode(A) 返回 A 中出现次数最多的值。

如果 A 为非空矩阵，那么 mode(A) 将返回包含 A 每列众数的行向量。

如果 A 为 0×0 空矩阵，mode(A) 返回 NaN。

如果 A 为多维数组，则 mode(A) 将沿大小不等于 1 的第一个数组维度的值视为向量，并返回一个由出现次数最多的值组成的数组。此维度的大小将变为 1，而所有其他维度的大小保持不变。

M = mode(A,all = true) 计算 A 的所有元素的众数。

M = mode(A,dims) 返回维度 dims 上的元素的众数。例如，如果 A 为矩阵，则 mode(A,2) 是包含每一行的出现次数最多值的列向量。

M = mode(A,vecdim) 计算向量 vecdim 所指定的维度上的众数。例如，如果 A 是矩阵，则 mode(A,[1 2]) 是 A 中所有元素的众数，因为矩阵的每个元素都包含在由维度 1 和 2 定义的数组切片中。

M,F,C = mode(___) 返回与 M 和 F 同样大小的数组 C。C 中的每个元素都是与 M 中对应元素出现频率相同的所有值的排序向量。

C. 示例

(1). 矩阵列的众数

定义一个 3×4 矩阵。

```
A = [3 3 1 4; 0 0 1 1; 0 1 2 4]
```

```
3     3     1     4
0     0     1     1
0     1     2     4
```

计算每一列中出现次数最多的值。

```
M = mode(A)
```

```
0     0     1     4
```

(2). 矩阵行的众数

定义一个 3×4 矩阵。

```
A = [3 3 1 4; 0 0 1 1; 0 1 2 4]
```

```
3     3     1     4
0     0     1     1
0     1     2     4
```

计算每一行中出现次数最多的值。

```
M = mode(A, 2)
```

```
3
0
0
```

(3). 三维数组的众数

创建一个由 1 和 10 之间的整数组成的 1×3×4 数组。

```
A = Int32.(zeros(1, 3, 4))
A[:, :, 1] = [10 8 10]
A[:, :, 2] = [6 9 5]
A[:, :, 3] = [9 6 1]
A[:, :, 4] = [4 9 5]
```

```
A[:, :, 1] =
10     8    10

A[:, :, 2] =
6     9     5

A[:, :, 3] =
9     6     1

A[:, :, 4] =
4     9     5
```

沿第二个维度计算此三维数组的出现次数最多的值。

```
M = mode(A)
```

```
M[:, :, 1] =
10

M[:, :, 2] =
5

M[:, :, 3] =
1
```

```
M[:, :, 4] =
4
```

此运算通过沿第二个维度计算出现次数最多的值来生成一个 1×1×4 数组。第二个维度的大小降到 1。

沿 A 的第一个维度计算模式。

```
M = mode(A, 1);
isequal(A, M)
```

```
ans = true
```

(4). 数组页面的众数

创建一个三维数组并计算每页数据（行和列）的众数。

```
A = Int32.(zeros(2, 2, 3))
A[:, :, 1] = [2 4; 2 1];
A[:, :, 2] = [6 2; 3 3];
A[:, :, 3] = [4 4; 7 4];
M1 = mode(A, (1, 2))
```

```
M1[:, :, 1] =
2

M1[:, :, 2] =
3

M1[:, :, 3] =
4
```

要计算一个数组的所有维度上的众数，可以在向量维参数中指定每个维度，或使用 'all' 选项。

```
M2 = mode(A, [1 2 3])
```

```
M2 = 4
```

```
Mall = mode(A; all = true)
```

```
Mall = 4
```

(5). 矩阵列众数及其频次信息

定义一个 3×4 矩阵。

```
A = [3 3 1 4; 0 0 1 1; 0 1 2 4]
```

```
A = 3×4

3     3     1     4
0     0     1     1
0     1     2     4
```

计算每一列出现次数最多的值及其出现次数。

```
M, F = mode(A, nargout = 2)
```

```
M =
0    0    1    4

F =
2    1    2    2
```

F[1] 为 2，因为 M[1] 在第一列中出现两次。

(6). 矩阵行众数及其频次信息和同频次数据信息

定义一个 3×4 矩阵。

```
A = [3 3 1 4; 0 0 1 1; 0 1 2 4]
```

```
A = 3×4
3    3    1    4
0    0    1    1
0    1    2    4
```

计算每行出现次数最多的值、出现次数以及该行中出现次数同样多的值。

```
M, F, C = mode(A, 2, nargout = 3)
```

```
M = 3×1 Matrix{Int64}:
3
0
0

F = 3×1 Matrix{Int64}:
2
2
1

C = 3×1 Matrix{Vector{Int64}}:
[3]
[0, 1]
[0, 1, 2, 4]
```

C[2] 为 2×1 向量 [0;1]，因为第二行中值 0 和 1 出现次数为 F[3]。

C[3] 为 4×1 向量 [0;1;2;4]，因为第三行中所有值出现次数为 F[3]。

(7). 16 位无符号整数数组的众数

定义一个由 16 位无符号整数组成的 1×4 向量。

```
A = UInt16[6 3 2 3]
```

```
A = 1×4 Matrix{UInt16}:
0x0006  0x0003  0x0002  0x0003
```

计算出现次数最多的值及其出现次数。

```
M, F, C = mode(A, 2; nargout = 3)
```

```
1×1 Matrix{UInt16}:
0x0003

1×1 Matrix{Int64}:
2

1×1 Matrix{Vector{UInt16}}:
[0x0003]
```

```
typeof(M[1])
```

UInt16

M 与输入 A 的类相同。

3.1.3.2.5 movsum

移动总和

A. 语法

```
M = movsum(A,k)
M = movsum(A,[kb kf])
M = movsum(___,dim)
M = movsum(___,nanflag)
M = movsum(A,k;key=value)
```

B. 说明

M = movsum(A,k) 返回由局部 k 个数据点总和组成的数组，其中每个总和基于 A 的相邻元素的长度为 k 的移动窗口计算得出。当 k 为奇数时，窗口以当前位置的元素为中心。当 k 为偶数时，窗口以当前元素及其前一个元素为中心。当没有足够的元素填满窗口时，窗口将自动在端点处截断。当窗口被截断时，只根据窗口内的元素计算总和。M 与 A 的大小相同。

如果 A 是向量，movsum 将沿该向量的长度运算。

如果 A 为多维数组，则 movsum 沿大小不等于 1 的第一个数组维度进行运算。

M = movsum(A,[kb kf]) 通过长度为 kb+kf+1 的窗口计算和，其中包括当前位置的元素、后面的 kb 个元素和前面的 kf 个元素。

M = movsum(___,dim) 为上述任意语法返回维度 dim 上的移动总和数组。例如，如果 A 是矩阵，则 movsum(A,k,2) 沿 A 的列运算，计算每行的 k 个元素的移动总和。

M = movsum(___,nanflag) 指定在上述任意语法的计算中包括还是忽略 NaN 值。movsum(A,k,"includenan") 会在计算中包括所有 NaN 值，而 movsum(A,k,"omitnan") 则忽略这些值并基于较少的点计算和。

M = movsum(A,k;key=value) 使用一个或多个名称-值对组参数指定和的其他参数。例如，如果 x 是时间向量，则 movsum(A,k;samplepoints=t) 相对于 x 中的时间计算 A 的移动和。

C. 示例

(1). 向量的中心移动和

计算行向量的三点中心移动和。当端点处的窗口中少于三个元素时，将根据可用元素计算和。

```
using TyBase

A = [4 8 6 -1 -2 -3 -1 3 4 5]
M = movsum(A, 3)
```

```
M = 1×10 Matrix{Int64}:
 12  18  13  3  -6  -6  -1  6  12  9
```

(2). 向量的尾部移动和

计算行向量的三点尾部移动和。当端点处的窗口中少于三个元素时，movsum 将根据可用数量的元素计算和。

```
using TyBase

A = [4 8 6 -1 -2 -3 -1 3 4 5]
M = movsum(A, [2 0])
```

```
M = 1×10 Matrix{Int64}:
 4  12  18  13  3  -6  -6  -1  6  12
```

(3). 矩阵的移动和

计算矩阵中每行的三点中心移动和。窗口从第一行开始，沿水平方向移动到该行的末尾，然后移到第二行，依此类推。维度参数为 2，即跨 A 的列移动窗口。

```
using TyBase

A = [4 8 6; -1 -2 -3; -1 3 4]
```

```
A = 3×3 Matrix{Int64}:
  4   8   6
 -1  -2  -3
 -1   3   4
```

```
M = movsum(A, 3, 2)
```

```
M = 3×3 Matrix{Int64}:
 12  18  14
 -3  -6  -5
  2   6   7
```

(4). 包含 NaN 元素的向量的移动和

计算包含两个 NaN 元素的行向量的三点中心移动和。

```
using TyBase

A = [4 8 NaN -1 -2 -3 NaN 3 4 5];
M = movsum(A, 3)
```

```
M = 1×10 Matrix{Float64}:
 12.0  NaN  NaN  NaN  -6.0  NaN  NaN  NaN  12.0  9.0
```

重新计算和，但忽略 NaN 值。当 movsum 舍弃 NaN 元素时，它将根据窗口中的剩余元素计算和。

```
M = movsum(A, 3, "omitnan")
```

```
M = 1×10 Matrix{Float64}:
 12.0  12.0  7.0  -3.0  -6.0  -5.0  0.0  7.0  12.0  9.0
```

(5). 基于样本点计算移动和

根据时间向量 t，计算 A 中数据的 3 小时中心移动和。

```
using Dates
using TyBase

A = [4 8 6 -1 -2 -3];
k = Hour(3)t = DateTime(0016, 1, 1, 0, 0, 0) + Hour.(0:5)
```

```
t = 6-element Vector{DateTime}:
 0016-01-01T00:00:00
 0016-01-01T01:00:00
 0016-01-01T02:00:00
 0016-01-01T03:00:00
 0016-01-01T04:00:00
 0016-01-01T05:00:00
```

```
M = movsum(A, k; samplepoints=t)
```

```
 M = 1×6 Matrix{Int64}:
 12  18  13  3  -6  -5
```

(6). 仅返回满窗口和

计算行向量的三点中心移动和，但在输出中舍弃使用的点数少于三个的计算。也就是说，只返回从满的三元素窗口计算的和，而舍弃端点计算。

```
A = [4 8 6 -1 -2 -3 -1 3 4 5];
M = movsum(A, 3; endpoints="discard")
```

```
M = 1×8 Matrix{Int64}:
 18  13  3  -6  -6  -1  6  12
```

3.1.3.3 大型文件和大数据

访问和处理文件集合以及大型数据集

函数名	简介
add	向 KeyValue 中添加单个键-值对组

3.1.3.3.1 add

向 kwargs 中添加单个键-值对组

A. 语法

```
add(key,value;kwargs)
```

B. 说明

add(key,value;kwargs)向 kwargs 中添加单个键-值对组，后者是在函数执行期间传入的键值对组。在编写使用 kwargs 的函数中使用 add 存储中间或最终键-值对组信息。

C. 示例

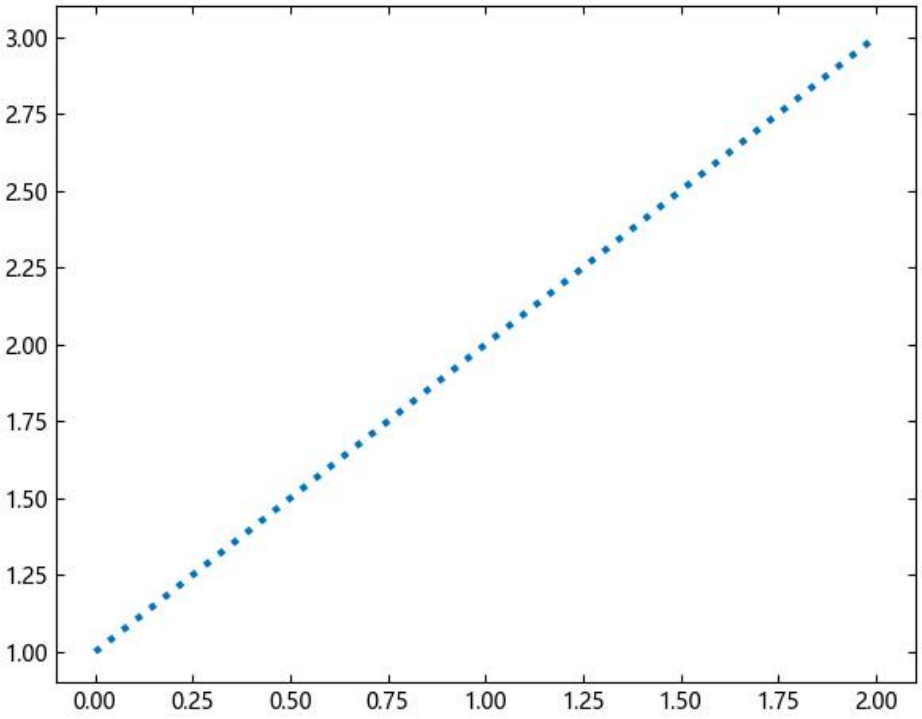
(1). 向 KeyValueStore 中添加键-值对组

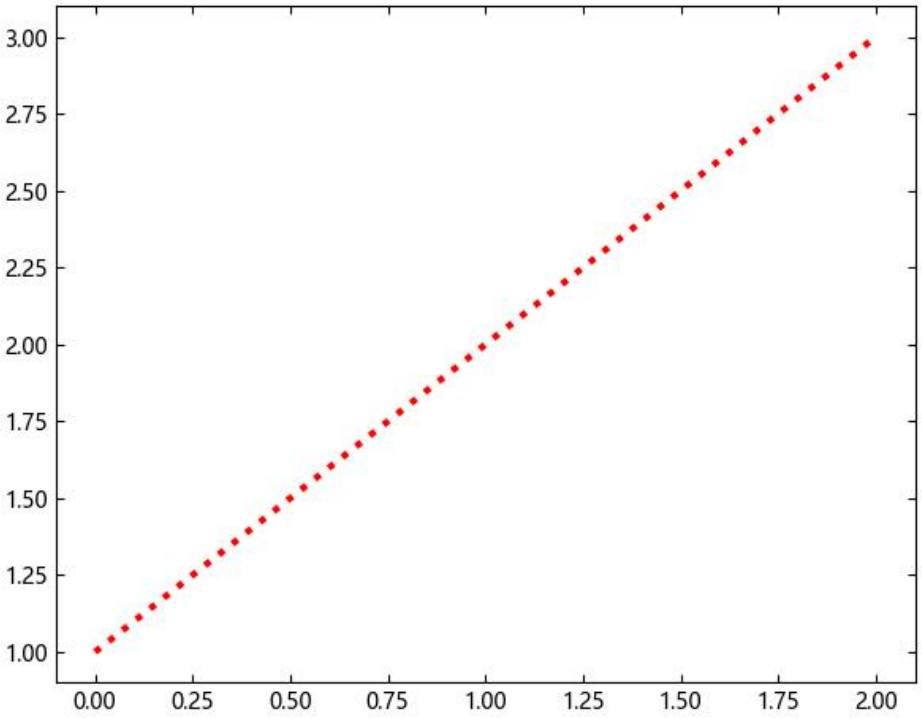
在函数中使用 add 将数据传递给中间和最后的 KeyValue。此示例使用 plot 函数，直接将输入传递给 kwargs。

```
function plots(x; kwargs...)
    plot(x; kwargs...)
```

```
    kwargs = add("color","r";kwargs...)
    figure()
    plot(x; kwargs...)
end

plots([1,2,3],linewidth = 3,linestyle = ":")
```





3.1.3.4 数据的预处理

数据的清理、平滑处理和分组

函数名	简介
fillmissing	填充缺失值
rmmissing	删除缺失的条目
standardizemissing	插入标准缺失值

3.1.3.4.1 fillmissing

填充缺失值

A. 语法

```
F = fillmissing(A,"constant",v)
F = fillmissing(A,method)
F = fillmissing(___,dim)
F = fillmissing(___,Key=Value)
```

```
F,TF = fillmissing(___)
```

B. 说明

F = fillmissing(A,"constant",v) 使用常量值 v 填充缺失的数组或表条目。如果 A 是矩阵或多维数组，则 v 可以是标量或向量。如果 v 是向量，每个元素指定 A 对应列中的填充值。

缺失值的定义取决于 A 的数据类型：

NaN - Float、Complex

"" – String

' ' – Char

F = fillmissing(A,method) 使用 method 指定的方法填充缺失的条目。例如，fillmissing(A,"previous") 对 A 中的缺失条目使用上一个非缺失条目进行填充。

F = fillmissing(___,dim) 指定要沿其进行运算的 A 的维度。默认情况下，fillmissing 沿其大小不为 1 的第一个维度进行运算。例如，如果 A 是矩阵，则 fillmissing(A,2) 跨 A 的各列进行运算，逐行填充缺失的数据。

F = fillmissing(___,Key=Value) 使用一个或多个名称-值对组参数指定用于填充缺失值的其他参数。例如，如果 t 是时间值向量，则 fillmissing(A,"linear",samplepoints=t) 会基于 t 中的时间值对 A 中的数据进行插值。

F,TF = fillmissing(___) 还会返回对应于填充的 A 条目的逻辑数组。

C. 示例

(1). 包含 NaN 值的向量

创建包含 NaN 值的向量，并使用前一个非缺失值替换每个 NaN。

```
A = [1 3 NaN 4 NaN NaN 5];
F = fillmissing(A, "previous")
```

```
1×7 Matrix{Float64}:
1.0  3.0  3.0  4.0  4.0  4.0  5.0
```

(2). 插入缺失数据

使用插值来替换非均匀采样的数据中的 NaN 值。

定义非均匀采样点向量，并计算这些点上的正弦函数。

```
using TyPlot
x = [collect((-4 * pi):0.1:0); collect(0.1:0.2:(4 * pi))];
```

```
A = sin.(x);
```

将 NaN 值插入 A 中。

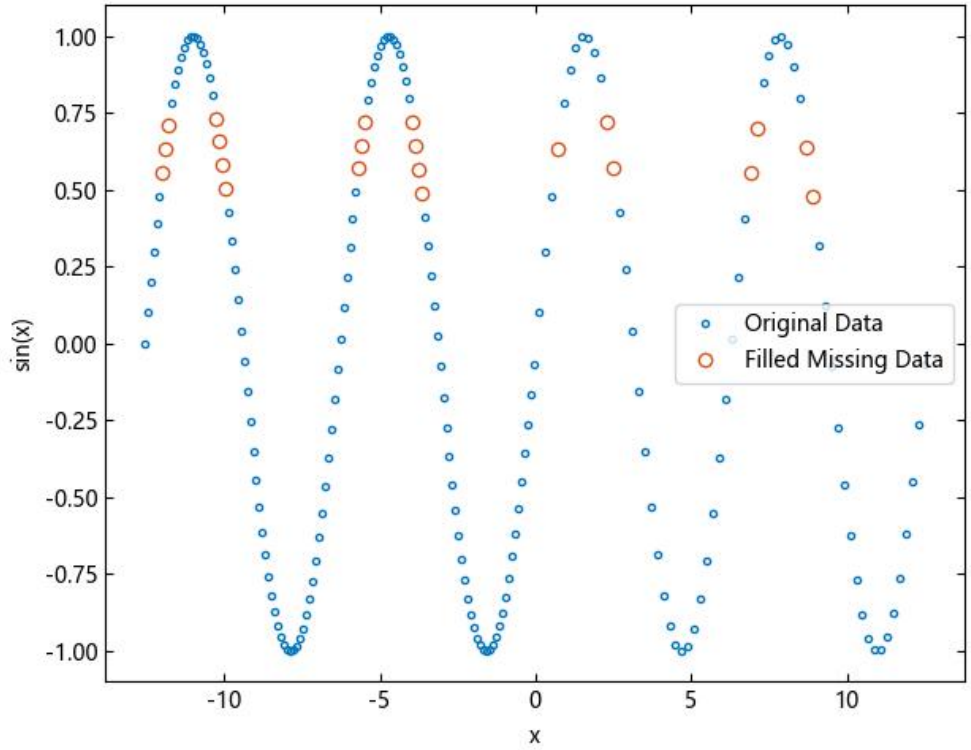
```
@. A[(A < 0.75) & (A > 0.5)] = NaN;
```

使用线性插值填充缺失数据，并返回填充的向量 F 和逻辑向量 TF。TF 项中的值 1 (true) 对应于 F 中的填充值。

```
F, TF = fillmissing(A, "linear"; samplepoints=x, nargout=2);
```

绘制原始数据和填充的数据。

```
plot(x, A, ".", x[TF], F[TF], "o")
xlabel("x");
ylabel("sin(x)")
legend(["Original Data", "Filled Missing Data"])
```



3.1.3.4.2 rmmissing

删除缺失的条目

A. 语法

```
R = rmmissing(A)
R = rmmissing(A,dim)
```

```
R = rmmissing(___,Key=Value)
R,TF = rmmissing(___)
```

B. 说明

R = rmmissing(A) 从数组或表中删除缺失的条目。如果 A 是向量，则 rmmissing 会删除包含缺失数据的所有条目。如果 A 是矩阵，则 rmmissing 会删除包含缺失数据的所有行。缺失值的定义取决于 A 的数据类型：

NaN - Float、Complex

"" – String

' ' – Char

Period(0) - Period (eg. Day(0) - Day)

R = rmmissing(A,dim) 指定要沿其进行运算的 A 的维度。默认情况下，rmmissing 沿其大小不为 1 的第一个维度进行运算。

R = rmmissing(___,Key=Value) 使用一个或多个名称-值对组参数指定用来删除缺失条目的其他参数。例如，您可以使用 rmmissing(A;minnummissing=n) 删除 A 的包含至少 n 个缺失值的行。

R,TF = rmmissing(___) 还返回与 A 中被删除的行或列对应的逻辑向量。

C. 示例

(1). 包含 NaN 值的向量

创建包含 NaN 值的向量，然后删除每个 NaN。

```
A = [1 3 NaN 6 NaN];
R = rmmissing(A)
```

```
R = 1×3 Matrix{Float64}:
 1.0  3.0  6.0
```

(2). 包含多个数据类型的表

从具有多个数据类型的表中删除不完整的行。

首先创建一个表，其变量包括 String、Float64 和 Char 数据类型。

```
A = DataFrame(;
    Gender=[' ', 'F', 'M'], Age=[45; 32; NaN], State=["", "CA",
"MA"], ID=[6051; 7234; NaN]
    )
```

```
3×4 DataFrame
 Row | Gender  Age     State   ID
```

```
     | Char    Float64  String  Float64
─────┼──────────────────────────────────
   1 |           45.0          6051.0
   2 | F         32.0  CA       7234.0
   3 | M         NaN   MA        NaN
```

删除表中包含缺失数据的任何行。

```
R = rmmissing(A)
```

```
1×4 DataFrame
 Row | Gender  Age     State   ID
     | Char    Float64  String  Float64
─────┼──────────────────────────────────
   1 | F         32.0  CA       7234.0
```

只删除在 Age 或 ID 表变量中包含缺失值的行。

```
R = rmmissing(A; datavariables=["Age", "ID"])
```

```
2×4 DataFrame
 Row | Gender  Age     State   ID
     | Char    Float64  String  Float64
─────┼──────────────────────────────────
   1 |           45.0          6051.0
   2 | F         32.0  CA       7234.0
```

或者，使用 ty_isnumeric 函数识别要对其执行运算的数字变量。

```
R = rmmissing(A; datavariables=ty_isnumeric)
```

```
2×4 DataFrame
 Row | Gender  Age     State   ID
     | Char    Float64  String  Float64
─────┼──────────────────────────────────
   1 |           45.0          6051.0
   2 | F         32.0  CA       7234.0
```

(3). 包含缺失数据的矩阵

创建一个包含缺失数据的矩阵，然后删除包含两个或更多缺失值的任何列（第二个维度）。返回新矩阵以及指示 A 的哪些列已被删除的逻辑行向量。

```
A = [
    NaN NaN 5 3 NaN 5 7 NaN 9 2
        8 9 NaN 1 4 5 6 5 NaN 5
        NaN 4 9 8 7 2 4 1 NaN 3
    ]
```

```
A = 3×10 Matrix{Float64}:
NaN   NaN   5.0   3.0   NaN   5.0   7.0   NaN   9.0   2.0
8.0   9.0   NaN   1.0   4.0   5.0   6.0   5.0   NaN   5.0
```

```
NaN    4.0    9.0    8.0    7.0    2.0    4.0    1.0    NaN    3.0
```

```
R, TF = rmmissing(A, 2; minnummissing=2, nargout=2)
```

```
R = 3×8 Matrix{Float64}:
NaN    5.0    3.0    NaN    5.0    7.0    NaN    2.0
9.0    NaN    1.0    4.0    5.0    6.0    5.0    5.0
4.0    9.0    8.0    7.0    2.0    4.0    1.0    3.0

TF = 10-element Vector{Bool}:
 1
 0
 0
 0
 0
 0
 0
 0
 1
 0
```

3.1.3.4.3 standardizemissing

插入标准缺失值

A. 语法

```
B = standardizemissing(A,indicator)
B = standardizemissing(A,indicator,datavariables=vars)
```

B. 说明

B = standardizemissing(A,indicator) 将数组或表中由 indicator 指定的值替换为标准缺失值。

标准缺失值取决于数据类型：

NaN - Float、Complex

"" – String

' ' – Char

Period(0) - Period (eg. Day(0) - Day)

当 A 是表或时间表时，B = standardizemissing(A,indicator,datavariables=vars) 将对 vars 指定的变量中的缺失值进行标准化。

C. 示例

(1). 非标准缺失数值

创建一个行向量并将 -99 的所有实例替换为 double 数据类型的标准缺失值 NaN。

```
A = [0 1 5 -99 8 3 4 -99 16];
B = standardizemissing(A, -99)
```

```
1×9 Matrix{Float64}:
 0.0  1.0  5.0  NaN  8.0  3.0  4.0  NaN  16.0
```

(2). 替换指定值的所有实例

创建一个包含 Inf 和 "N/A" 的表来表示缺少的值。

```
A = DataFrame(;
    dblVar=[NaN; 3; Inf; 7; 9],
    cellstrVar=["one"; "three"; ""; "N/A"; "nine"],
    charVar=['A'; 'C'; 'E'; ' '; 'I'],
    categoryVar=["red"; "yellow"; "blue"; "violet"; undef],
)
```

```
5×4 DataFrame
 Row │ dblVar   cellstrVar  charVar  categoryVar
     │ Float64  String      Char     Any
─────┼──────────────────────────────────────────
   1 │   NaN    one         A        red
   2 │     3.0  three       C        yellow
   3 │   Inf                E        blue
   4 │     7.0  N/A                  violet
   5 │     9.0  nine        I        #undef
```

将 Inf 的所有实例替换为 NaN，将 "N/A" 的所有实例替换为空字符向量 "" 。

```
B = standardizemissing(A, [Inf, "N/A"])
```

```
5×4 DataFrame
 Row │ dblVar   cellstrVar  charVar  categoryVar
     │ Float64  String      Char     Any
─────┼──────────────────────────────────────────
   1 │   NaN    one         A        red
   2 │     3.0  three       C        yellow
   3 │   NaN                E        blue
   4 │     7.0                       violet
   5 │     9.0  nine        I        #undef
```

(3). 仅替换指定变量中的值

将表的指定变量中出现的 Inf 和 "N/A" 的实例替换为标准缺失值指示符。

创建一个包含 Inf 和 "N/A" 的表来表示缺少的值。

```
A = DataFrame(;
    a=["alpha"; "bravo"; "charlie"; ""; "N/A"],
    x=[1; NaN; 3; Inf; 5],
    y=[57; 732; 93; 1398; Inf],
)
```

```
5×3 DataFrame
 Row │ a       x       y
     │ String   Float64  Float64
─────┼──────────────────────────
   1 │ alpha       1.0    57.0
   2 │ bravo       NaN    732.0
   3 │ charlie     3.0    93.0
   4 │             Inf    1398.0
   5 │ N/A         5.0    Inf
```

将 Inf 的所有实例替换为 NaN，将 "N/A" 的所有实例替换为空字符向量 ""。

```
B = standardizemissing(A, [Inf, "N/A"]; datavariables=["a", "x"])
```

```
5×3 DataFrame
 Row │ a       x       y
     │ String   Float64  Float64
─────┼──────────────────────────
   1 │ alpha       1.0    57.0
   2 │ bravo       NaN    732.0
   3 │ charlie     3.0    93.0
   4 │             NaN    1398.0
   5 │             5.0    Inf
```

变量 y 中的 Inf 保持不变，原因是 y 未包含在 datavariables 名称-值对组参数中。

3.2 数学 API

线性代数、微积分、傅里叶变换和其他数学

数学函数为分析数据、开发算法和创建模型提供了一系列数值计算方法。核心函数使用经过处理器优化的库，可以快速进行向量和矩阵计算。

3.2.1 初等数学

三角学、指数和对数、复数值、舍入、余数、离散数学 初等数学函数包括支持算术运算（+、-、*、...）的功能、数学常量函数（Inf、pi、…）、多项式运算函数（poly、roots、…）以及特殊的数学函数（如 gamma 和 beta）。

3.2.1.1 算术运算

加、减、乘、除、幂、四舍五入 算术函数包括用于简单运算（如加法和乘法）的运算符，以及用于常见计算（如求和、移动和、取模运算和舍入）的函数。

3.2.1.1.1 基本算术

函数名	简介
+	添加数字，追加字符串
sum	数组元素总和
cumsum	累积和
-	减法
diff	差分和近似导数
.*	乘法
*	矩阵乘法
cumprod	累积乘积
pagemtimes	按页矩阵乘法
prod	数组元素的乘积

./	数组右除
.\	数组左除
^	矩阵幂
'	复共轭转置
transpose	转置向量或矩阵
pagetranspose	按页转置
pagectranspose	按页复共轭转置

+

一元加法

A. 语法

```
C = +(x,y...)
C = A + B
```

B. 说明

C = +(x,y...)返回数组 A 并将其存储在 C 中 A + B 等效于 +(A,B)。 示例

C. 示例

矩阵的一元加法

创建一个 2×2 矩阵 A。

```
A = [1 -3; -2 4]
```

```
A = 2×2 Matrix{Int64}:
1  -3
-2   4
```

对 A 应用一元加法。

```
C = +A
```

```
C = 2×2 Matrix{Int64}:
1   -3
-2  4
```

C 和 A 相同。

sum

数组求和

A. 语法

```
S = sum(A)
S = sum(A,dims)
S = sum(A,init)
S = sum(f,A)
```

B. 说明

S = sum(A)用于求数组元素之和。

S = sum(A,dims = dim) 可指定对数组的某一维度 dim 进行求和，“dims =”不可省略 ，例如对二维矩阵列进行求和：sum(A,dims = 1)。

S = sum(A,init = a) 指定初始求和元素为 a,"init = "不可省略。

S = sum(f,A) 对于数组 A 的每个元素调用函数 f 的结果进行求和。

C. 示例

(1). 数组元素的总和

创建一个向量并计算各个元素的总和。

```
A = collect(1:10)
S = sum(A)
```

```
S = 55
```

创建一个矩阵并计算元素的总和。

```
A = [1 3 2;4 2 5;6 1 4]
```

```
A = 3×3 Matrix{Int64}:
1     3     2
4     2     5
6     1     4
```

```
S = sum(A)
```

```
S = 28
```

(2). 矩阵列总和

创建一个矩阵并计算每列中元素的总和。

```
A = [1 3 2;4 2 5;6 1 4]
```

```
A = 3×3 Matrix{Int64}:
1     3     2
```

```
4     2     5
6     1     4
```

```
S = sum(A,dims = 1)
```

```
S = 1×3 Matrix{Int64}:
11 6 11
```

(3). 矩阵行总和

创建一个矩阵并计算每行中元素的总和。

```
A = [1 3 2;4 2 5;6 1 4]
```

```
A = 3×3 Matrix{Int64}:
1     3     2
4     2     5
6     1     4
```

```
S = sum(A,dims = 2)
```

```
S = 3×1 Matrix{Int64}:
6
11
11
```

(4). 数组切片总和

使用向量维度参数来操作数组的特定切片。

创建所有元素为 1 的三维数组。

```
A = ones(4,3,2)
```

要对 A 的每个页中的所有元素求和，请使用向量维度参数指定要求和的维度（行和列）。两个页均为 4×3 的全 1 矩阵，因此每页的总和为 12。

```
S1 = sum(A,dims = [1 2])
```

```
S1 = 1×1×2 Array{Float64, 3}:
[:, :, 1] =
12.0
[:, :, 2] =
12.0
```

如果沿第一个维度对 A 进行切片，则可以对得到的 4 个页的元素求和，每个页为 3×2 矩阵。

```
S2 = sum(A,dims = [2 3])
```

```
S2 = 4×1×1 Array{Float64, 3}:
[:, :, 1] =
6.0
6.0
6.0
6.0
```

沿第二个维度切片，每个页的总和是一个 4×2 矩阵的元素之和。

```
S3 = sum(A,dims = [1 3])
```

```
S3 = 1×3×1 Array{Float64, 3}:
[:, :, 1] =
8.0 8.0 8.0
```

(5). 三维数组的总和

创建一个由 1 值组成的 4×2×3 数组，并计算沿第三个维度的总和。

```
A = ones(4,2,3)
S = sum(A,dims = 3)
```

```
S = 4×2×1 Array{Float64, 3}:
[:, :, 1] =
3.0 3.0
3.0 3.0
3.0 3.0
3.0 3.0
```

(6). 设置初始求和元素

```
sum(1:10,init = 0+0im)
```

```
ans = 55 + 0im
```

```
sum(1:10,init = 1+0im)
```

```
ans = 56 + 0im
```

(7). 调用函数并求和

求矩阵元素的平方和

```
sum(abs2,[2 3 4])
```

```
ans = 29
```

cumsum

累积和

A. 语法

```
B = cumsum(A)
B = cumsum(A,dims = dim)
```

B. 说明

B = cumsum(A)返回向量的元素的累积和。

B = cumsum(A,dims=dim)返回沿维度 dim 的元素的累积和，“dims=”不可省略。例如，如果 A 是矩阵，则 cumsum(A,dims=2)返回每行的累积和。

C. 示例

(1). 向量的累积和

计算从 1 到 5 的整数的累积和。元素 B[2] 是 A[1] 和 A[2] 的和，而 B[5] 是元素 A[1] 至 A[5] 的和。

```
A = collect(1:5)
B = cumsum(A)
```

```
B = 5-element Vector{Int64}:
1
3
6
10
15
```

(2). 矩阵中每列的累积和

定义其元素与其线性索引对应的 3×3 矩阵。

```
A = [1 4 7;2 5 8;3 6 9]
```

```
A = 3×3 Matrix{Int64}:
1  4  7
2  5  8
3  6  9
```

```
B = cumsum(A,dims = 1)
```

```
B = 3×3 Matrix{Int64}:
1   4   7
3   9  15
6  15  24
```

(3). 矩阵中每行的累积和

定义其元素与其线性索引对应的 2×3 矩阵。

```
A = [1 3 5;2 4 6]
```

```
A = 2×3 Matrix{Int64}:
1  3  5
2  4  6
```

```
B = cumsum(A,dims = 2)
```

```
B = 2×3 Matrix{Int64}:
1  4   9
```

```
2     6     12
```

(4). 逻辑输入的累积和

创建一个逻辑值数组。

```
A = [true false true; true true false]
```

```
A = 2×3 Matrix{Bool}:
1     0     1
1     1     0
```

计算 A 的行的累积和。

```
B = cumsum(A,dims = 2)
```

```
B = 2×3 Matrix{Int64}:
1     1     2
1     2     2
```

(5). 包含 NaN 值的向量

创建一个包含 NaN 值的向量，并计算累积和。默认情况下，cumsum 包括 NaN 值。如果您在计算中包括 NaN 值，则只要在 A 中遇到第一个 NaN 值，累积和将立即成为 NaN。

```
A = [3,5,NaN,9,0,NaN]
B = cumsum(A)
```

```
B = 6-element Vector{Float64}:
3.0
8.0
NaN
NaN
NaN
NaN
```

-

一元减法

A. 语法

```
C = -(A,B)
C = A - B
```

B. 说明

C = -A 对 A 的各个元素求反，然后将结果存储在 C 中。 A - B 等效于-(A,B)示例。

C. 示例

创建一个 2×2 矩阵 A。

```
A = [1 -3; -2 4]
```

```
A = 2×2 Matrix{Int64}:
1  -3
-2   4
```

对 A 元素求反

```
C = -A
```

```
C = 2×2 Matrix{Int64}:
-1   3
2  -4
```

diff

差分

A. 语法

```
Y = diff(X)
Y = diff(X,dims)
```

B. 说明

Y = diff(X) 计算沿大小不等于 1 的第一个数组维度的 X 相邻元素之间的差分：

如果 X 是长度为 m 的向量，则 Y = diff(X) 返回长度为 m-1 的向量。Y 的元素是 X 相邻元素之间的差分。

Y = [X[2]-X[1] X[3]-X[2] ... X[m]-X[m-1]] 如果 X 是不为空的非向量 p×m 矩阵，则 Y = diff(X) 返回大小为 (p-1)×m 的矩阵，其元素是 X 的行之间的差分。

Y = [X[2,:]-X[1,:]; X[3,:]-X[2,:]; ... X[p,:]-X[p-1,:]] 如果 X 是 0×0 的空矩阵，则 Y = diff(X) 返回 0×0 的空矩阵。

Y = diff(X,dims) 是沿 dims 指定的维计算差分。dims 输入是一个正整数标量。

C. 示例

(1). 向量元素之间的差分

创建一个向量，然后计算元素之间的差分。

```
X = [1 ,1, 2, 3, 5, 8, 13, 21]
Y = diff(X)
```

```
Y = 7-element Vector{Int64}:
0
1
1
2
3
5
8
```

请注意，Y 的元素比 X 少一个。

(2). 矩阵行之间的差分

创建一个 3×3 矩阵，然后计算各行之间的一阶差分。

```
X = [1 1 1; 5 5 5; 25 25 25]
Y = diff(X,dims=1)
```

```
Y = 2×3 Matrix{Int64}:
4   4   4
20 20 20
```

Y 是 2×3 矩阵。

(3). 矩阵列之间的差分

创建一个 3×3 矩阵，然后计算各列之间的一阶差分。

```
X = [1 3 5;7 11 13;17 19 23]
Y = diff(X,dims=2)
```

```
Y = 3×2 Matrix{Int64}:
2 2
4 2
2 4
```

Y 是一个 3×2 矩阵。

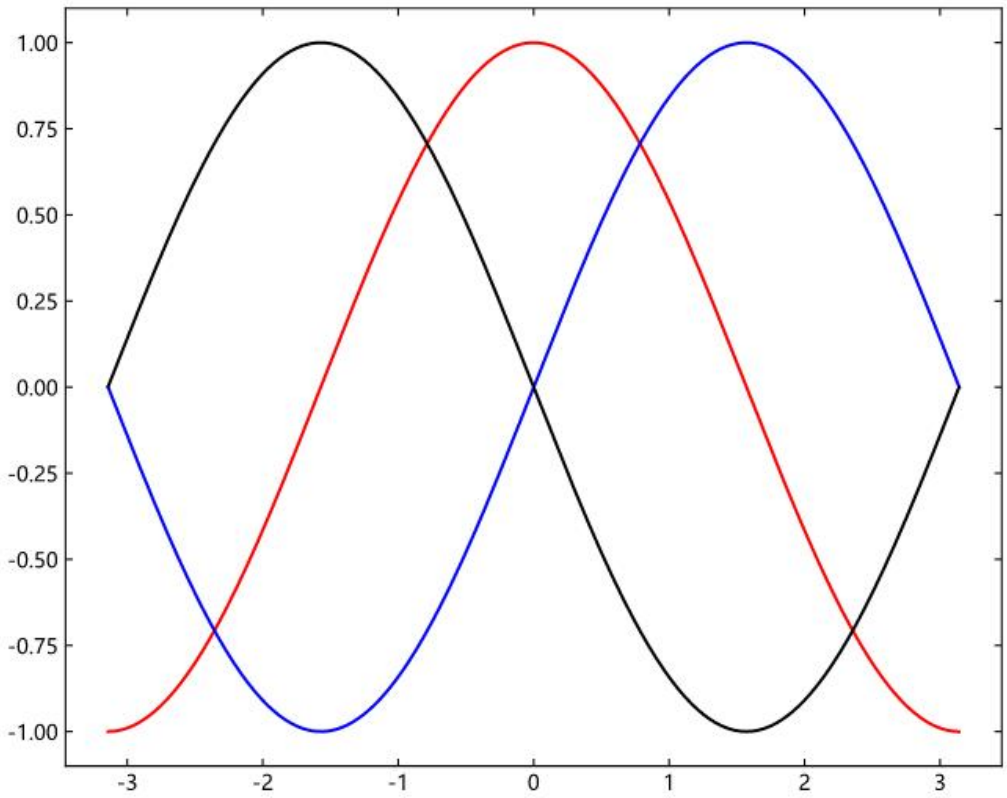
(4). 使用差分求导数近似值

使用 diff 函数和语法 Y = diff(f)/h 求偏导数近似值，其中 f 是函数值在某些域 X 上计算的向量，h 是一个相应的步长。

例如，sin(x) 相对于 x 的第一个导数为 cos(x)，相对于 x 的第二个导数值为 -sin(x)。可以使用 diff 求这些导数的近似值。

```
h = 0.001;       # step size
X = -pi:h:pi;    # domain
f = sin.(X);      # range
```

```
Y = diff(f)/h;   # first derivative
Z = diff(Y)/h;   # second derivative
plot(X[1:length(Y),:],Y,"r")
hold("on")
plot(X,f,"b")
plot(X[1:length(Z),:],Z,"k")
```



在此绘图中，蓝色线条对应原始函数 sin。红色线条对应计算出的第一个导数 cos，黑色线条对应计算出的第二个导数 -sin。

.*

元素乘积

A. 语法

```
C = A.*B
```

B. 说明

A.*B 返回对应元素相乘。 示例

C. 示例

(1). 将两个向量相乘

创建两个向量 A 和 B，并按元素将这两个向量相乘。

```
A = [1 0 3];
B = [2 3 7];
C = A.*B
```

```
1×3 Matrix{Int64}:
2  0  21
```

(2). 将两个数组相乘

创建两个 3×3 数组 A 和 B，并按元素将这两个数组相乘。

```
A = [1 0 3; 5 3 8; 2 4 6];
B = [2 3 7; 9 1 5; 8 8 3];
C = A.*B
```

```
3×3 Matrix{Int64}:
2   0  21
45   3  40
16  32  18
```

(3). 将行和列向量相乘创建

创建行向量 a 和列向量 b，然后将它们相乘。1×3 行向量和 4×1 列向量组合起来产生 4×3 矩阵。

```
a = (1:3)';
b = (1:4);
a.*b
```

```
ans = 4×3 Matrix{Int64}:
1  2   3
2  4   6
3  6   9
4  8  12
```

结果是 4×3 矩阵，矩阵中的每个 [i,j] 元素等于 a[j].*b[i]：

$$a = [a_1 \quad a_2 \quad a_3], \quad b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}, \quad a \cdot * b = \begin{bmatrix} a_1 b_1 & a_2 b_1 & a_3 b_1 \\ a_1 b_2 & a_2 b_2 & a_3 b_2 \\ a_1 b_3 & a_2 b_3 & a_3 b_3 \\ a_1 b_4 & a_2 b_4 & a_3 b_4 \end{bmatrix}.$$

*

矩阵乘法

A. 语法

```
C = A*B
C = *(A,B)
```

B. 说明

C＝A*B 是 A 和 B 的矩阵乘积。如果 A 是 m×p 矩阵，B 是 p×n 矩阵，则 C 是通过以下公式定义的 m×n 矩阵：

$$C(i,j) = \sum_{k=1}^{p} A(i,k)B(k,j)$$

该定义说明 C[i,j] 是 A 第 i 行与 B 第 j 列的内积。

```
C[i,j] = A[i,:]*B[:,j]
```

对于非标量 A 和 B，A 的列数必须等于 B 的行数。对于非标量输入，矩阵乘法不一定能互换位置。也就是说，A * B 不一定等于 B * A。至少一方输入为标量时，A * B 等于 A.*B，这时可以互换位置。示例

C. 示例

(1). 将两个一维数组相乘

创建一个 1×4 矩阵 A 和一个 4×1 向量 B。

```
A = [1 1 0 0]
B = [1; 2; 3; 4];
```

A 乘以 B。

```
C = A*B
```

```
C = 1-element Vector{Int64}:
3
```

结果是 1×1 标量，也称为向量 A 和 B 的点积或内积。或者，可以使用 dot(A,B) 语法来计算 A⋅ B 点积。

B 乘以 A。

```
C = B*A
```

```
C = 4×4 Matrix{Int64}:
1  1  0  0
2  2  0  0
3  3  0  0
```

```
4  4  0  0
```

结果是 4×4 矩阵，也称为向量 A 和 B 的外积。两个向量的外积 A⊗ B 返回一个矩阵。

(2). 将两个数组相乘

创建两个数组 A 和 B。

```
A = [1 3 5; 2 4 7]
B = [-5 8 11; 3 9 21; 4 0 8];
```

计算 A 和 B 的乘积。

```
C = A*B
```

```
C = 2×3 Matrix{Int64}:
24  35  114
30  52  162
```

计算 A 第二行和 B 第三列的内积。

```
A[2,:]'*B[:,3]
```

```
ans = 162
```

cumprod

累计乘积

A. 语法

```
B = cumprod(A)
B = cumprod(A; dims::Integer)
```

A. 说明

B = cumprod(A) 从 A 中的第一个大小不等于 1 的数组维度开始返回 A 的累积乘积。

B = cumprod(A; dims::Integer) 返回沿维度 dims 的累积乘积。

B. 示例

(1). 向量的累计乘积

计算从 1 到 5 的整数的累积乘积。元素 B[2] 是 A[1] 和 A[2] 的乘积，而 B[5] 是元素 A[1] 至 A[5] 的乘积。

```
A = 1 : 5;
B = cumprod(A)
```

```
5-element Vector{Int64}:
```

```
1
2
6
24
120
```

(2). 矩阵中按维度累计乘积

定义其元素与其线性索引对应的 3×3 矩阵。

```
A = [1 4 7; 2 5 8; 3 6 9]
```

```
3×3 Matrix{Int64}:
1     4     7
2     5     8
3     6     9
```

计算 A 的列的累积乘积。元素 B[5] 是 A[4] 和 A[5] 的乘积，而 B[9] 是 A[7]、A[8] 和 A[9] 的乘积。

```
B = cumprod(A , dims = 1)
```

```
3×3 Matrix{Int64}:
1     4     7
2     20     56
6   120   504
```

计算 A 的行的累积乘积。

```
B = cumprod(A , dims = 2)
```

```
B = 3×3 Matrix{Int64}:
1     4     28
2    10     80
3    18    162
```

(3). 矩阵中每行的累计乘积

定义其元素与其线性索引对应的 2×3 矩阵。

```
A = [1 3 5; 2 4 6]
```

```
A = 2×3 Matrix{Int64}:
1  3  5
2  4  6
```

计算 A 的行的累积乘积。元素 B[3] 是 A[1] 和 A[3] 的乘积，而 B[5] 是 A[1]、A[3] 和 A[5] 的乘积。

```
B = cumprod(A,dims = 2)
```

```
B = 2×3 Matrix{Int64}:
1  3  15
2  8  48
```

(4). 乘积包含 NaN

若输入数组中含有 NaN，则从第一个 NaN 出现之后累计乘积计算结果为 NaN。

```
A = [1 3 2 4 NaN 3 NaN 2];
cumprod(A,dims=2)
```

```
1×8 Matrix{Float64}:
1.0  3.0  6.0  24.0  NaN  NaN  NaN  NaN
```

Pagemtimes

按页矩阵乘法

A. 语法

```
Z = pagemtimes(X, Y)
Z = pagemtimes(X, transpX, Y, transpY)
```

B. 说明

Z = pagemtimes(X, Y) 计算 N 维数组 X 和 Y 的对应的页的矩阵乘积。输出数组 Z 的每页都由乘积给出： Z[:,:,i] = X[:,:,i]*Y[:,:,i] 。示例

如果 X 和 Y 之一是矩阵，则 pagemtimes 将其与另一输入的每页相乘。例如，如果 X 是矩阵，则 Z[:,:,i] = X*Y[:,:,i] 。

如果 X 和 Y 有三个以上的维度，则前两个维度以外的所有维度必须具有兼容大小。pagemtimes 隐式扩展额外的维度，以乘以分页矩阵的所有组合：Z[:,:,i,j,k] = Xx[:,:,i,j,k]*Yy[:,:,i,j,k] 。（额外维度已在 Xx 和 Yy 中扩展。）

Z = pagemtimes(X, transpX, Y, transpY) 可以选择使用 X 和 Y 的指定转置来计算矩阵乘积。对于此语法，您必须指定两个转置选项。 transpX 和 transpY 必须分别为: "transpose"、 "ctranspose" 或 []。 示例。

C. 示例

(1). 乘以三维数组

```
A = magic(3)
```

```
A = 3×3 Matrix{Int64}:
 8  1  6
 3  5  7
 4  9  2
```

```
Y=reshape([9,10,2,10,7,1,3,6,10,10,2,10,10,5,9,2,5,10,8,10,7,1,9,10,7
```

```
,8,8],3,3,3)
```

```
Y = 3×3×3 Array{Int64, 3}:
[:, :, 1] =
  9  10   3
 10   7   6
  2   1  10
[:, :, 2] =
 10  10   2
  2   5   5
 10   9  10
[:, :, 3] =
  8   1  7
 10   9  8
  7  10  8
```

```
Z = pagemtimes(A,Y)
```

```
Z = 3×3×3 Array{Int64, 3}:
[:, :, 1] =
  94   93   90
  91   72  109
 130  105   86
[:, :, 2] =
 142  139   81
 110  118  101
  78  103   73
[:, :, 3] =
 116   77  112
 123  118  117
 136  105  116
```

输出 Z[:,:,i] 的第 i 页通过乘以 A*Y[:,:,i] 而形成。

(2). 乘以转置后的数组页

创建两个三维数组 X 和 Y 。使用 pagemtimes 对 X 和 Y 的每页执行运算 X[:,:,i]'*Y[:,:,i] 。

```
X=reshape([0.814723686393179,0.905791937075619,0.126986816293506,0.91
3375856139019,0.63235924622541,0.0975404049994095,0.278498218867048,0
.546881519204984,0.957506835434298,0.964888535199277,0.15761308167754
8,0.970592781760616,0.957166948242946,0.485375648722841,0.80028046888
88,0.141886338627215,0.421761282626275,0.915735525189067,0.7922073295
59554,0.959492426392903,0.655740699156587,0.0357116785741896,0.849129
305868777,0.933993247757551,0.678735154857773,0.757740130578333,0.743
132468124916],3,3,3) .+ 1im
```

```
Y=reshape([0.392227019534168,0.655477890177557,0.171186687811562,0.706046088019609,0.0318328463774207,0.27692298496089,0.0461713906311539,0.0971317812358475,0.823457828327293,0.694828622975817,0.317099480060861,0.950222048838355,0.0344460805029088,0.438744359656398,0.381558457093008,0.765516788149002,0.795199901137063,0.186872604554379,0.489764395788231,0.445586200710899,0.646313010111265,0.709364830858073,0.754686681982361,0.276025076998578,0.679702676853675,0.655098003973841,0.162611735194631],3,3,3)
A = pagemtimes(X,"ctranspose",Y,"none")
```

```
A = 3×3×3 Array{Complex, 3}:
[:, :, 1] =
 0.935022-1.21889im  0.639232-1.0148im  0.230166-0.966761im
 0.789446-1.21889im  0.692026-1.0148im  0.183914-0.966761im
 0.631616-1.21889im  0.479197-1.0148im  0.854445-0.966761im
[:, :, 2] =
 1.64269-1.96215im  0.472726-0.854749im   1.04535-1.74759im
 1.57942-1.96215im   0.55128-0.854749im   1.26825-1.74759im
 1.10248-1.96215im  0.539339-0.854749im  0.615127-1.74759im
[:, :, 3] =
  1.23935-1.58166im   1.46708-1.74008im   1.27366-1.49741im
 0.999503-1.58166im  0.923965-1.74008im  0.732414-1.49741im
  1.15036-1.58166im   1.25845-1.74008im   1.07857-1.49741im
```

现在，对 X 和 Y 的每页执行运算 X[:,:,i]*Y[:,:,j]' 。

```
B = pagemtimes(X,"none",Y,"transpose")
```

```
B = 3×3×3 Array{Complex, 3}:
[:, :, 1] =
 0.977301+1.14444im   0.59016+0.784443im  0.621736+1.27157im
 0.827001+1.14444im  0.666976+0.784443im  0.780508+1.27157im
 0.162885+1.14444im  0.179346+0.784443im  0.837216+1.27157im
[:, :, 2] =
 0.812019+1.49479im  0.838745+1.55104im   1.30859+1.51865im
 0.449099+1.49479im  0.598319+1.55104im  0.413782+1.51865im
  1.40297+1.49479im   1.38709+1.55104im   1.39876+1.51865im
[:, :, 3] =
 0.874666+1.87883im  0.824586+1.85537im  0.632242+1.08495im
  1.58731+1.87883im   1.56476+1.85537im  0.977731+1.08495im
  1.48881+1.87883im   1.48389+1.85537im  0.802461+1.08495im
```

(3). 将 N 维数组相乘

创建一个 3x3x2 数组 X ，并将其与 3x3x1x4 数组 Y 相乘。结果大小为 3x3x2x4 。

```
X = ones(Int,3,3,2)
A = eye(3)
Y = cat(A,A.*2,A.*3,A.*4,dims=4)
Z = pagemtimes(X, Y)
```

```
Z = 3×3×2×4 Array{Int64, 4}:
[:, :, 1, 1] =
 1  1  1
 1  1  1
 1  1  1
[:, :, 2, 1] =
 1  1  1
 1  1  1
 1  1  1
[:, :, 1, 2] =
 2  2  2
 2  2  2
 2  2  2
[:, :, 2, 2] =
 2  2  2
 2  2  2
 2  2  2
[:, :, 1, 3] =
 3  3  3
 3  3  3
 3  3  3
[:, :, 2, 3] =
 3  3  3
 3  3  3
 3  3  3
[:, :, 1, 4] =
 4  4  4
 4  4  4
 4  4  4
[:, :, 2, 4] =
 4  4  4
 4  4  4
 4  4  4
```

大小为 1 的每个维度（在前两个维度后）会隐式扩展以匹配另一个输入的维度大小，然后通过乘以 X[:,:,i,j]*Y[:,:,i,j] 而形成输出 Z[:,:,i,j] 的每页。此运算的一种直观考虑方式是， X 在三维数组中包含两个矩阵作为页， Y 包含四个沿着第四个维度排列的矩阵；因此，将这些矩阵的所有组合相乘得到八个 3x3 矩阵。

```
2  8  48
```

prod

元素乘积

A. 语法

```
prod(f, itr; [init])
```

B. 说明

B = prod(A) 返回 A 的数组元素的乘积。

如果 A 是向量，则 prod(A) 返回元素的乘积。

如果 A 为非空矩阵，则 prod(A) 将 A 的各列视为向量，并返回一个包含每列乘积的行向量。

如果 A 为 0×0 空矩阵，prod(A) 返回 1。

如果 A 为多维数组，则 prod(A) 沿第一个非单一维度运算并返回乘积数组。此维度的大小将减少至 1，而所有其他维度的大小保持不变。

返回应用于 itr 的每个元素的 f 的乘积。

对于小于系统字大小的有符号整数，返回类型为 Int，对于小于系统字大小的无符号整数，返回类型为 UInt。 对于所有其他参数，找到一个公共返回类型，所有参数都被提升到该类型。

可以通过 init 指定为空 itr 返回的值。 它必须是乘法标识（即一），因为未指定 init 是否用于非空集合。

C. 示例

(1). 每列中元素的乘积

创建其元素与其线性索引对应的 3×3 数组。

```
A=[(1:3:7)' ;(2:3:8)' ;(3:3:9)']
```

```
A = 3×3 Matrix{Int64}:
1 4 7
2 5 8
3 6 9
```

计算每列中元素的乘积。

```
B = prod(A,dims = 1)
```

```
B = 1×3 Matrix{Int64}:
6 120 504
```

(2). 逻辑输入的乘积

创建一个逻辑值数组。

```
A = [true false; true true]
```

```
A = 2×2 Matrix{Bool}:
1  0
1  1
```

计算每列中元素的乘积。

```
B = prod(A,dims = 1)
```

```
1×2 Matrix{Bool}:
1  0
```

(3). 每行中元素的乘积

创建其元素与其线性索引对应的 3×3 数组。

```
A=[(1:3:7)' ;(2:3:8)' ;(3:3:9)']
```

```
A = 3×3 Matrix{Int64}:
1  4  7
2  5  8
3  6  9
```

计算每行中元素的乘积，并将第二个维度的长度减少到 1。第一个维度的长度与 size(A,1) 匹配，第二个维度的长度为 1。

```
dim = 2;
B = prod(A,dims = dim)
```

```
B = 3×1 Matrix{Int64}:
28
80
162
```

(4). 数组页面的乘积

创建一个三维数组并计算每页数据（行和列）的乘积。

```
A = zeros(2,2,3)
```

```
A[:,:,1] = [2 4; -2 1];
A[:,:,2] = [1 2; -5 3];
A[:,:,3] = [4 4; 1 -3];
```

```
B1 = prod(A,dims = [1,2])
```

```
B1 = 1×1×3 Array{Float64, 3}:
[:, :, 1] =
 -16.0
[:, :, 2] =
 -30.0
[:, :, 3] =
```

```
 -48.0
```

```
B2 = prod(A,dims = [1 2 3])
```

```
B2 = 1×1×1 Array{Float64, 3}:
[:, :, 1] =
 -23040.0
```

```
Ball = prod(A)
```

```
Ball = -23040.0
```

'

复共轭转置

A. 语法

```
B = transpose(A)B = A'
B = adjoint(A)
```

B. 说明

B = A' 计算 A 的复共轭转置。示例

B = adjoint(A) 是执行 A' 的替代方法，但很少使用。它可以启用类的运算符重载。

C. 示例

(1). 实矩阵的共轭转置

创建一个 4×2 的矩阵。

```
A = [2 1; 9 7; 2 8; 3 5]
```

```
A = 4×2 Matrix{Int64}:
2  1
9  7
2  8
3  5
```

计算 A 的共轭转置。

```
B = A'
```

```
B = 2×4 adjoint(::Matrix{Int64}) with eltype Int64:
2  9  2  3
1  7  8  5
```

结果是一个 2×4 的矩阵。B 的元素与 A 相同，但每个元素的行和列索引将

会交换。

(2). 复矩阵的共轭转置

创建一个包含复数元素的 2×2 的矩阵。

```
A = [0-1im 2+1im;4+2im 0-2im]
```

```
A = 2×2 Matrix{Complex{Int64}}:
0-1im  2+1im
4+2im  0-2im
```

计算 A 的共轭转置。计算 A 的共轭转置。

```
B = adjoint(A)
```

```
B = 2×2 adjoint(::Matrix{Complex{Int64}}) with eltype Complex{Int64}:
0+1im  4-2im
2-1im  0+2im
```

transpose

转置向量或矩阵

A. 语法

```
B = transpose(A)
```

B. 说明

B = transpose(A) 返回 A 的非共轭转置，即每个元素的行和列索引都会互换。

C. 示例

(1). 实矩阵

创建由实数组成的矩阵并计算其转置。B 具有与 A 相同的元素，但是 B 的行是 A 的列，B 的列是 A 的行。

```
A = magic(4)
```

```
 A = 4×4 Matrix{Int64}:
16   2   3  13
 5  11  10   8
 9   7   6  12
 4  14  15   1
```

```
B = transpose(A)
```

```
B = 4×4 transpose(::Matrix{Int64}) with eltype Int64:
```

```
16   5   9   4
2  11   7  14
3  10   6  15
13   8  12   1
```

(2). 复矩阵

创建包含复数元素的矩阵并计算其非共轭转置。B 包含与 A 相同的元素，只是交换了行和列。虚部符号保持不变。

```
A = [1 3 4-1im 2+2im; 0+1im 1-1im 5 6-1im]
```

```
 2×4 Matrix{Complex{Int64}}:
1+0im  3+0im  4-1im  2+2im
0+1im  1-1im  5+0im  6-1im
```

```
B = transpose(A)
```

```
4×2 transpose(::Matrix{Complex{Int64}}) with eltype Complex{Int64}:
1+0im  0+1im
3+0im  1-1im
4-1im  5+0im
2+2im  6-1im
```

pagetranspose

按页转置

A. 语法

```
Y = pagetranspose(X)
```

B. 说明

Y = pagetranspose(X) 将非共轭转置应用于 N 维数组 X 的每页。输出 Y[:,:,i] 的每页都是通过对 X 中的对应页进行转置来找到的，如 X[:,:,i]' 中所示。示例。

C. 示例

转置 N 维数组的页

创建一个三维数组 A ，然后使用 pagetranspose 转置数组的每页。

```
r = repelem((1:3)',3,1)
A = cat(r,2 .*r,3 .*r,dims=3)
```

```
A = 3×3×3 Array{Int64, 3}:
```

```
[:, :, 1] =
 1  2  3
 1  2  3
 1  2  3
[:, :, 2] =
 2  4  6
 2  4  6
 2  4  6
[:, :, 3] =
 3  6  9
 3  6  9
 3  6  9
```

```
B = pagetranspose(A)
```

```
B = 3×3×3 Array{Int64, 3}
[:, :, 1] =
 1  1  1
 2  2  2
 3  3  3
[:, :, 2] =
 2  2  2
 4  4  4
 6  6  6
[:, :, 3] =
 3  3  3
 6  6  6
 9  9  9
```

pagectranspose

按页复共轭转置

A. 语法

```
Y = pagectranspose(X)
```

B. 说明

Y = pagetranspose(X) 将复共轭转置应用于 N 维数组 X 的每页。输出 Y[:,:,i] 的每页均为 X 中对应页的共轭转置，如 X[:,:,i]' 中所示。示例。

C. 示例

恢复复数 N 维数组的页

创建一个复三维数组 A ，然后使用 pagectranspose 得到该数组每页的复共轭转置。

```
r = repelem((1:3)',3,1)
A = cat(r,2 .*r,3 .*r,dims=3)
A = A .+ 1im
```

```
A = 3×3×3 Array{Complex{Int64}, 3}:
[:, :, 1] =
 1+1im  2+1im  3+1im
 1+1im  2+1im  3+1im
 1+1im  2+1im  3+1im
[:, :, 2] =
 2+1im  4+1im  6+1im
 2+1im  4+1im  6+1im
 2+1im  4+1im  6+1im
[:, :, 3] =
 3+1im  6+1im  9+1im
 3+1im  6+1im  9+1im
 3+1im  6+1im  9+1im
```

```
B = pagectranspose(A)
```

```
B = 3×3×3 Array{Complex{Int64}, 3}:
[:, :, 1] =
 1-1im  1-1im  1-1im
 2-1im  2-1im  2-1im
 3-1im  3-1im  3-1im
[:, :, 2] =
 2-1im  2-1im  2-1im
 4-1im  4-1im  4-1im
 6-1im  6-1im  6-1im
[:, :, 3] =
 3-1im  3-1im  3-1im
 6-1im  6-1im  6-1im
 9-1im  9-1im  9-1im
```

3.2.1.1.2 模除法和舍入

函数名	简介
mod	除后的余数（取模运算）
rem	除后的余数

div	带有舍入选项的整除
ceil	向正无穷舍入
trunc	向零舍入
floor	向负无穷舍入
round	四舍五入为最近的小数或整数

mod

除后的余数（取模运算）

A. 语法

```
b = mod(a,m)
```

B. 说明

b = mod(a,m) 返回 a 除以 m 后的余数，其中 a 是被除数，m 是除数。此函数通常称为取模运算，表达式为 b = a - m.*floor(a./m)。mod 函数遵从 mod(a,0) 返回 a 的约定。

C. 示例

(1). 标量被除后的余数

计算 23 对 5 取模。

```
b = mod(23,5)
```

```
b = 3
```

(2). 向量被除后的余数

计算一个整数向量除以除数 3 的余数。

```
a = [1:5...]
m = 3
b = mod.(a,m)
```

```
b = 5-element Vector{Int64}:
1
2
0
1
```

```
2
```

(3). 正值和负值被除后的余数

```
a = [-4 ,-1, 7, 9]
m = 3
b = mod.(a,m)
```

```
b = 4-element Vector{Int64}:
2
2
1
0
```

(4). 用负除数除后的余数

计算一个整数集除以负除数后的余数，该整数集包括正值和负值。请注意，如果除数为负数，则非零结果始终为负数。

```
a = [-4 ,-1 ,7 ,9]
m = -3
b = mod.(a,m)
```

```
b = 4-element Vector{Int64}:
-1
-1
-2
0
```

(5). 浮点值被除后的余数

计算多个角除以模数 2*pi 后的余数。请注意，mod 将尝试补偿浮点四舍五入取整的影响，以尽可能生成精确的整数结果。

```
theta = [0.0, 3.5, 5.9, 6.2, 9.0, 4*pi]
m = 2*pi
b = mod.(theta,m)
```

```
b = 6-element Vector{Float64}:
0.0
3.5
5.9
6.2
2.7168146928204138
0.0
```

rem

除后的余数

A. 语法

```
r = rem(a,b)
```

B. 说明

r = rem(a,b) 返回 a 除以 b 后的余数，其中 a 是被除数，b 是除数。此函数通常称为求余运算，表达式为 r = a - b.*fix(a./b)。rem 函数遵从 rem(a,0) 报错。

C. 示例

(1). 标量被除后的余数

计算 23 对 5 取模。

```
b = rem(23,5)
```

```
b = 3
```

(2). 向量被除后的余数

计算一个整数向量除以除数 3 的余数。

```
a = [1:5...]
m = 3
b = rem.(a,m)
```

```
b = 5-element Vector{Int64}:
1
2
0
1
2
```

(3). 正值和负值被除后的余数

```
a = [-4 ,-1, 7, 9]
m = 3
b = rem.(a,m)
```

```
b = 4-element Vector{Int64}:
-1
-1
```

```
1
0
```

(4). 浮点值被除后的余数

计算多个角除以除数 2*pi 后的余数。rem 将尝试补偿浮点四舍五入取整的影响，以尽可能生成精确的整数结果。

```
theta = [0.0, 3.5, 5.9, 6.2, 9.0, 4*pi]
m = 2*pi
b = rem.(theta,m)
```

```
b = 6-element Vector{Float64}:
0.0
3.5
5.9
6.2
2.7168146928204138
0.0
```

div

带有舍入选项的整除

A. 语法

```
C = idivide(A,B)
C = idivide(A,B,r::RoundingMode=RoundToZero)
```

B. 说明

C = idivide(A,B) 将 A 的每个元素除以 B 的对应元素，朝零方向舍入到最接近的整数。A 和 B 必须包含实数，并且其中有至少一个必须属于整数类。

如果 A 和 B 是数组，则它们必须属于同一整数类，并且大小兼容。

如果 A 或 B 是双精度标量，则另一个输入必须为整数类，但不能是 Int64 或 UInt64。然后，div 函数以相同的整数类形式返回 C。

C = idivide(A,B,r::RoundingMode)可以指定替代选项：

RoundingMode 选项	**对应函数**
RoundToZero	默认

RoundDown	fld
RoundUp	cld
RoundNearest	最近整数舍入
RoundNearestTiesAway	——
RoundNearestTiesUp	——

C. 示例

(1). 整数除法

创建整数数组 A 和整数标量 B。将 A 的每个元素除以 B，朝零方向舍入到最接近的整数。

```
A = Int16.([-7 -4 7 11])
B = Int16(10)
C = div.(A,B)
```

```
C = 1×4 Matrix{Int16}:
0  0  0  1
```

(2). 两个整数数组相除

创建两个整数数组 A 和 B。将 A 的每个元素除以 B 的对应元素，朝零方向舍入到最接近的整数。

```
A = Int64.([-2 3])
B = Int64.([3 5])
C = div.(A,B)
```

```
C = 1×2 Matrix{Int64}:
0  0
```

(3). 整数除法的舍入选项

创建双精度标量 A 和整数数组 B。用默认舍入选项 RoundToZero 将 A 除以 B 的每个元素。

```
A = 2.0
B = Int32.([-3 3 4])
```

```
C = div.(A,B)
```

```
C = 1×3 Matrix{Float64}:
-0.0  0.0  0.0
```

将结果与使用其他舍入选项时的结果进行比较。

```
C = div.(A,B,RoundDown)
```

```
C = 1×3 Matrix{Float64}:
-1.0  0.0  0.0
C = 1×3 Matrix{Float64}:
-0.0  1.0  1.0
```

```
C = div.(Int(A),B,RoundNearestTiesUp)
```

```
1×3 Matrix{Int64}:
-1  1  1
```

(4). 浮点值被除后的余数

计算多个角除以除数 2*pi 后的余数。div 将尝试补偿浮点四舍五入取整的影响，以尽可能生成精确的整数结果。

```
theta = [0.0, 3.5, 5.9, 6.2, 9.0, 4*pi]
m = 2*pi
b = div.(theta,m)
```

```
6-element Vector{Float64}:
0.0
0.0
0.0
0.0
1.0
2.0
```

ceil

向正无穷舍入

A. 语法

```
Y = ceil(X)
```

B. 说明

Y = ceil(X) 将 X 的每个元素四舍五入到大于或等于该元素的最接近整数。

C. 示例

(1). 向正无穷舍入矩阵元素

```
X = [-1.9 -0.2 3.4; 5.6 7 2.4 ]
Y = ceil.(X)
```

```
Y = 2×3 Matrix{Float64}:
-1.0  -0.0  4.0
6.0   7.0  3.0
```

trunc

向零舍入

A. 语法

```
Y = trunc(X)
```

B. 说明

Y = trunc(X) 将 X 的每个元素朝零方向四舍五入为最近的整数。此操作实际上是通过删除 X 中每个数的小数部分，将它们截断为整数：

对于正数，trunc 的行为与 floor 相同。

对于负数，trunc 的行为与 ceil 相同。

C. 示例

(1). 将矩阵元素向零取整

```
X = [-1.9 -3.4; 1.6 2.5; -4.5 4.5]
```

```
X = 3×2 Matrix{Float64}:
-1.9  -3.4
1.6   2.5
-4.5   4.5
```

```
Y = trunc.(X)
```

```
Y = 3×2 Matrix{Float64}:
-1.0  -3.0
1.0   2.0
-4.0   4.0
```

floor

向负无穷舍入

A. 语法

```
Y = floor(X)
```

B. 说明

Y = floor(X) 将 X 的每个元素四舍五入到小于或等于该元素的最接近整数。

C. 示例

(1). 向负无穷舍入矩阵元素

```
X = [-1.9  -0.2  3.4;  5.6  7.0  2.4]
Y = floor.(X)
```

```
Y = 2×3 Matrix{Float64}:
-2.0  -1.0  3.0
5.0   7.0  2.0
```

round

将数字 x 舍入

A. 语法

```
round(z::Complex[, RoundingModeReal, [RoundingModeImaginary]])
round(z::Complex[,    RoundingModeReal,    [RoundingModeImaginary]];
digits=, base=10)
round(z::Complex[,    RoundingModeReal,    [RoundingModeImaginary]];
sigdigits=, base=10)
```

```
round([T,] x, [r::RoundingMode])
round(x, [r::RoundingMode]; digits::Integer=0, base = 10)
round(x, [r::RoundingMode]; sigdigits::Integer, base = 10)
```

B. 说明

```
round(z::Complex[, RoundingModeReal, [RoundingModeImaginary]])

round(z::Complex[,    RoundingModeReal,    [RoundingModeImaginary]];
digits=, base=10)
```

```
round(z::Complex[,    RoundingModeReal,    [RoundingModeImaginary]];
sigdigits=, base=10)
```

返回最接近的整数值输入 z 到 z 的复数值，打破关系使用指定的 RoundingModes。 首先 RoundingMode 用于对实数进行四舍五入组件，而第二个用于舍入虚部。

```
round([T,] x, [r::RoundingMode])

round(x, [r::RoundingMode]; digits::Integer=0, base = 10)

round(x, [r::RoundingMode]; sigdigits::Integer, base = 10)
```

将数字 x 舍入。

如果没有关键字参数，x 将四舍五入为整数值，返回 T 类型的值，或 如果未提供 T，则为相同类型的 x。 如果值不是，将抛出 InexactError 可以用 T 表示，类似于 convert。

如果提供了 digits 关键字参数，则它后四舍五入到指定的位数小数位。

如果提供了 sigdigits 关键字参数，则它舍入到指定数量的重要数字，以基数为基础。

RoundingMode r 控制方向四舍五入; 默认为 RoundNearest，即舍入到最接近的整数，有关系（小数值为 0.5）四舍五入为最接近偶数。 请注意，回合可能会给如果全局舍入模式为改变了。

RoundingMode 关键字	**关键字意义**
digits	舍入位数
sigdigits	舍入到数字位数
base	进制位数

C. 示例

(1). 复数取整

```
round(3.14 + 4.5im)
```

```
ans = 3.0 + 4.0im
```

(2). 调整舍入模式取整

```
round(1.7)
```

```
ans = 2.0
```

设置输出数据类型

```
round(Int, 1.7)
```

```
ans = 2
```

设置舍入位数

```
round(pi; digits=2)
```

```
ans = 3.14
```

舍入到指定数量的重要数字，以基数为基础。

```
round(123.456; sigdigits=2)
```

```
ans = 120.0
```

3.2.1.2 三角学

结果以弧度或度为单位的正弦、余弦和相关函数。

Syslab 中的三角函数计算以弧度或度为单位的标准三角函数值、以弧度为单位的双曲三角函数值以及每个函数的反函数。

函数名	简介
sin	参数的正弦，以弧度为单位
sind	参数的正弦，以度为单位
sinpi	准确地计算 sin(X*pi)
asin	反正弦（以弧度为单位）
asind	反正弦（以度为单位）
sinh	双曲正弦
asinh	反双曲正弦
cos	以弧度为单位的参数的余弦
cosd	以度为单位的参数的余弦
cospi	准确计算 cos(X*pi)
acos	反余弦（以弧度为单位）
acosd	反余弦（以度为单位）
cosh	双曲余弦

acosh	反双曲余弦
tan	以弧度表示的参数的正切
tand	以度表示的参数的正切
atan	反正切（以弧度为单位）
atand	反正切（以度为单位）
tanh	双曲正切
atanh	反双曲正切
csc	输入角的余割（以弧度为单位）
cscd	以度为单位的参数的余割
acsc	反余割（以弧度为单位）
acscd	反余割（以度为单位）
csch	双曲余割
acsch	反双曲余割
sec	角的正割（以弧度为单位）
secd	参数的正割，以度为单位
asec	反正割（以弧度为单位）
sech	双曲正割
asech	反双曲正割
cot	角的余切（以弧度为单位）
cotd	以度为单位的参数的余切
acot	反余切（以弧度为单位）
acotd	反余切（以度为单位）
coth	双曲余切
acoth	反双曲余切
hypot	平方和的平方根（斜边）

deg2rad	将角从以度为单位转换为以弧度为单位
rad2deg	将角的单位从弧度转换为度
cart2pol	将笛卡尔坐标转换为极坐标或柱坐标
cart2sph	将笛卡尔坐标转换为球面坐标
pol2cart	将极坐标或柱坐标转换为笛卡尔坐标
sph2cart	将球面坐标转换为笛卡尔坐标

3.2.1.2.1 sin

参数的正弦，以弧度为单位

A. 语法

```
Y = sin(X)
```

B. 说明

Y = sin(X) 返回 X 的元素的正弦。sin 函数按元素处理数组。该函数同时接受实数和复数输入。

对于 X 的实数值，sin(X) 返回区间 [-1, 1] 内的实数值。

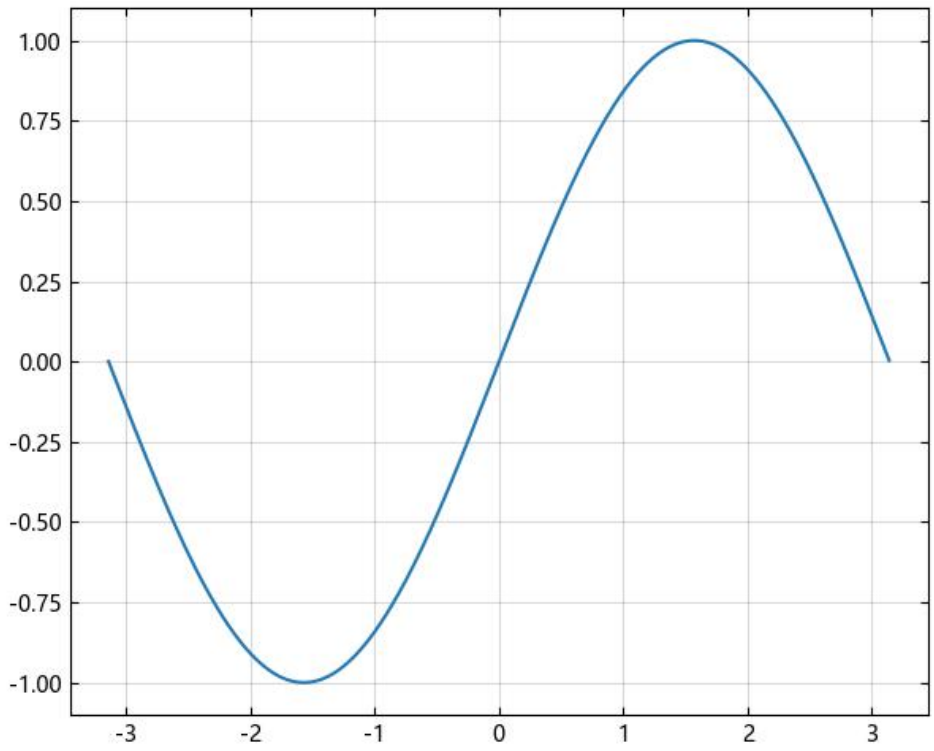
对于 X 的复数值，sin(X) 返回复数值。

C. 示例

(1). 绘制正弦函数

在域 −π≤x≤π 上绘制正弦函数图。

```
x = -pi:0.01:pi
plot(x,sin.(x))
grid("on")
```



(2). 复角向量的正弦

计算复角在向量 x 中的正弦。

```
x = [-im,pi+im*pi/2,-1+im*4]
y = sin.(x)
```

```
y = 3-element Vector{ComplexF64}:
0.0 - 1.1752011936438014im
3.0728573923780145e-16 - 2.3012989023072947im
-22.979085577886128 + 14.744805188558725im
```

3.2.1.2.2 sind

参数的正弦，以度为单位

A. 语法

```
Y = sin(X)
```

B. 说明

Y = sind(X) 返回 X 的元素的正弦（以度为单位）。

C. 示例

(1). 180 度的正弦与弧度的正弦相比较

```
sind(180)
```

```
ans = 0.0
```

```
sin(pi)
```

```
ans = 1.2246467991473532e-16
```

(2). 复数角向量的正弦，以度为单位指定

计算复角在向量 x 中的正弦。

```
z = [90+im,15+2im,10+3im]
y = sind.(z)
```

```
y = 3-element Vector{ComplexF64}:
1.0001523125762564 + 1.0687601995239893e-18im
0.25897674269344456 + 0.03372401964004195im
0.17388626522292966 + 0.05158797772543903 5im
```

3.2.1.2.3 sinpi

准确地计算 sin(X*pi)

A. 语法

```
Y = sinpi(X)
```

B. 说明

Y = sinpi(X) 计算 sin(X*pi) 而不显式计算 X*pi。此计算比 sin(X*pi) 更准确，因为 pi 的浮点值是 π 的近似值。尤其是：

对于整数，sinpi(n) 精确为零。

对于奇整数，sinpi(n/2) 为 +1 或 -1。

C. 示例

(1). 计算 π 的倍数的正弦

比较 sinpi(X) 与 sin(X*pi) 的精确度。

创建一个由值组成的向量。

```
X = [0 1/2 1 3/2 2]
```

使用常规 sin 函数计算 X*pi 的正弦值。

```
Y = sin.(X*pi)
```

```
Y = 1×5 Matrix{Float64}:
0.0  1.0  1.22465e-16  -1.0  -2.44929e-16
```

```
Y[3]
```

```
ans = 1.2246467991473532e-16
```

由于 pi 是 π 的实际值的浮点近似值，因此结果包含小的数值误差。例如，即使 sin(π)=0，Y[3] 也不会正好为零。

```
Z = sinpi.(X)
```

```
Z = 1×5 Matrix{Float64}:
0.0  1.0  0.0  -1.0  0.0
```

```
Z[3]
```

```
ans = 0.0
```

3.2.1.2.4 asin

反正弦（以弧度为单位）

A. 语法

```
Y = asin(X)
```

B. 说明

Y = asin(X) 以弧度为单位返回 X 各元素的 反正弦 (sin)^-1。该函数同时接受实数和复数输入。

对于 X 在区间 [-1, 1] 内的实数值，asin(X) 返回区间 [-π/2, π/2] 内的值。

对于 X 在区间 [-1, 1] 之外的实数值以及 X 的复数值，asin(X) 返回复数值。

C. 示例

(1). 值的反正弦

求值的反正弦。

```
y = asin(1)
```

```
y = 1.5707963267948966
```

(2). 复数值向量的反正弦

计算向量 x 的各元素的反正弦。asin 函数按元素处理 x。

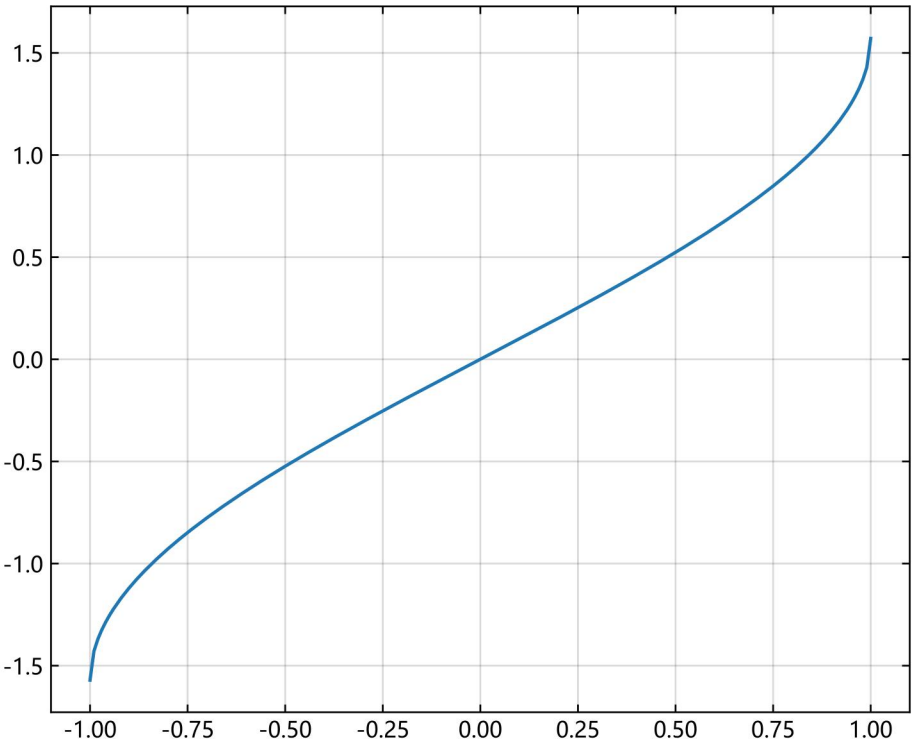
```
x = [0.5im,1+3im,-2.2+im]
y = asin.(x)
```

```
3-element Vector{ComplexF64}:
0.0 + 0.48121182505960347im
0.30760364953071123 + 1.8641615441578825im
-1.1090604166910276 + 1.5479846981255774im
```

(3). 绘制反正弦函数

在区间 −1≤x≤1 上绘制反正弦函数。

```
x = -1:.01:1
plot(x,asin.(x))
grid("on")
```



3.2.1.2.5 asind

反正弦（以度为单位）

A. 语法

```
Y = asin(X)
```

B. 说明

Y = asind(X) 返回 X 的元素的反正弦 (sin-1)（以度为单位）。该函数同时接受实数和复数输入。

C. 示例

(1). 标量的反正弦

说明 1 的反正弦正好是 90° 。

```
asind(1)
```

```
ans = 90.0
```

(2). 复角的来回计算

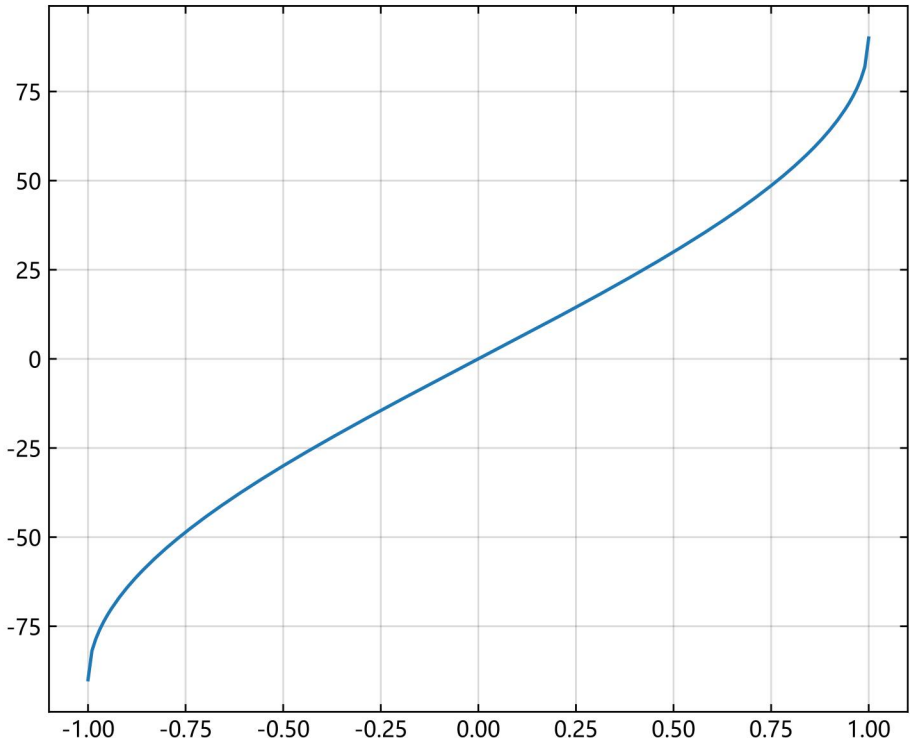
```
sind.(asind.([2+0im 3+0im]))
```

```
ans = 1×2 Matrix{ComplexF64}:
2.0+1.06058e-16im  3.0+1.73191e-16im
```

(3). 反正弦函数图

在域 −1≤x≤1 上绘制反正弦函数。

```
x = -1:.01:1
plot(x,asind.(x))
grid("on")
```



3.2.1.2.6 sinh

双曲正弦

A. 语法

```
Y = sinh(X)
```

B. 说明

Y = sinh(X) 返回 X 的元素的双曲正弦。sinh 函数按元素处理数组。该函数同时接受实数和复数输入。所有的角度都以弧度表示。

对于 X 在区间 [-1, 1] 内的实数值，asin(X) 返回区间 [-π/2, π/2] 内的值。

对于 X 在区间 [-1, 1] 之外的实数值以及 X 的复数值，asin(X) 返回复数值。

C. 示例

(1). 向量的双曲正弦

创建一个向量并计算每个值的双曲正弦。

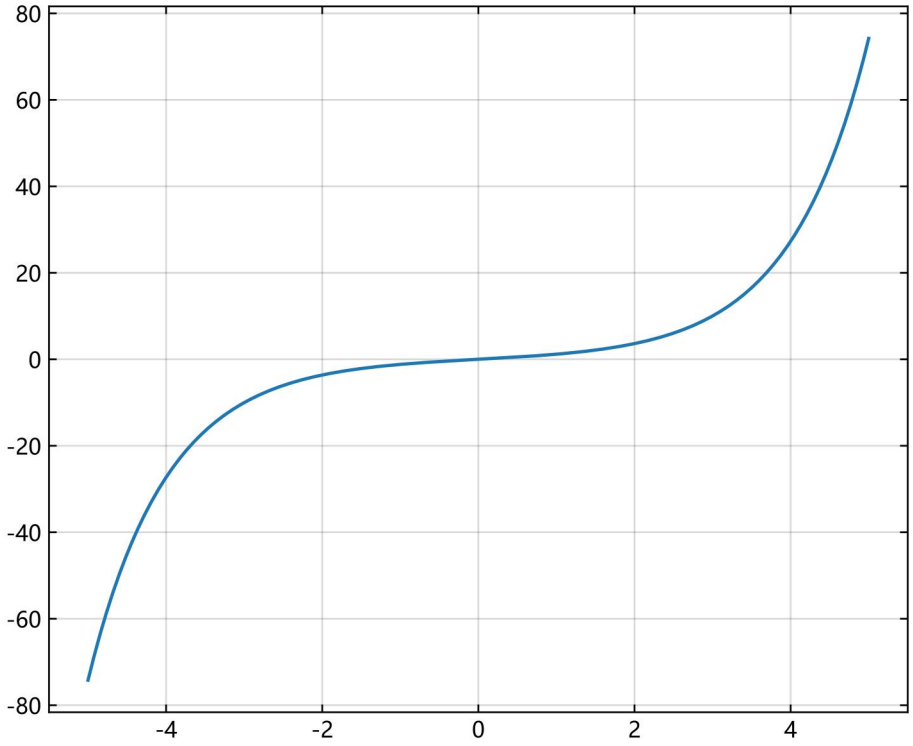
```
X = [0,pi,2*pi,3*pi];
Y = sinh.(X)
```

```
4-element Vector{Float64}:
0.0
11.548739357257748
267.74489404101644
6195.823863608588
```

(2). 双曲正弦函数图

在域 −5≤x≤5 上绘制双曲正弦函数图。

```
x = -5:0.01:5
y = sinh.(x)
plot(x,y)
grid("on")
```



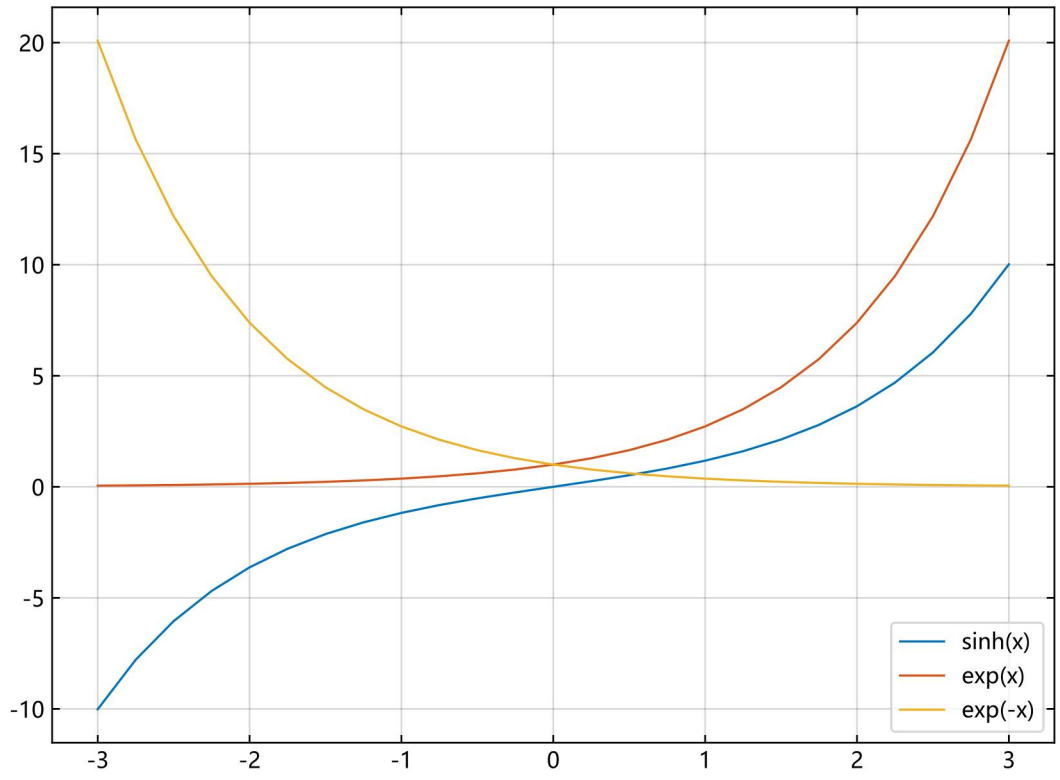
(3). 绘制双曲正弦函数和指数函数

双曲正弦满足恒等式 sinh(x)=(exp(x)−exp(−x))/2。换句话说，sinh(x) 是函数 exp(x) 和 exp(−x) 之差的一半。通过绘制函数可对此进行验证。

创建一个向量，其中包含 -3 和 3 之间的值，步长为 0.25。计算并绘制

sinh(x)、exp(x) 和 exp(-x) 的值。与预期相符，sinh 曲线在 exp(x) 大时为正，而在 exp(-x) 大时为负。

```
x = -3:0.25:3
y1 = sinh.(x)
y2 = exp.(x)
y3 = exp.(-x)
plot(x,y1,x,y2,x,y3)
grid("on")
legend(["sinh(x)","exp(x)","exp(-x)","Location","bestoutside"])
```



3.2.1.2.7 asinh

参数的正弦，以度为单位

A. 语法

```
Y = asinh(X)
```

B. 说明

Y = asinh(X) 返回 X 的元素的反双曲正弦。该函数同时接受实数和复数输入。所有的角度都以弧度表示。

C. 示例

(1). 向量的反双曲正弦

计算向量 X 的各元素的反双曲正弦。asinh 函数按元素处理 X。

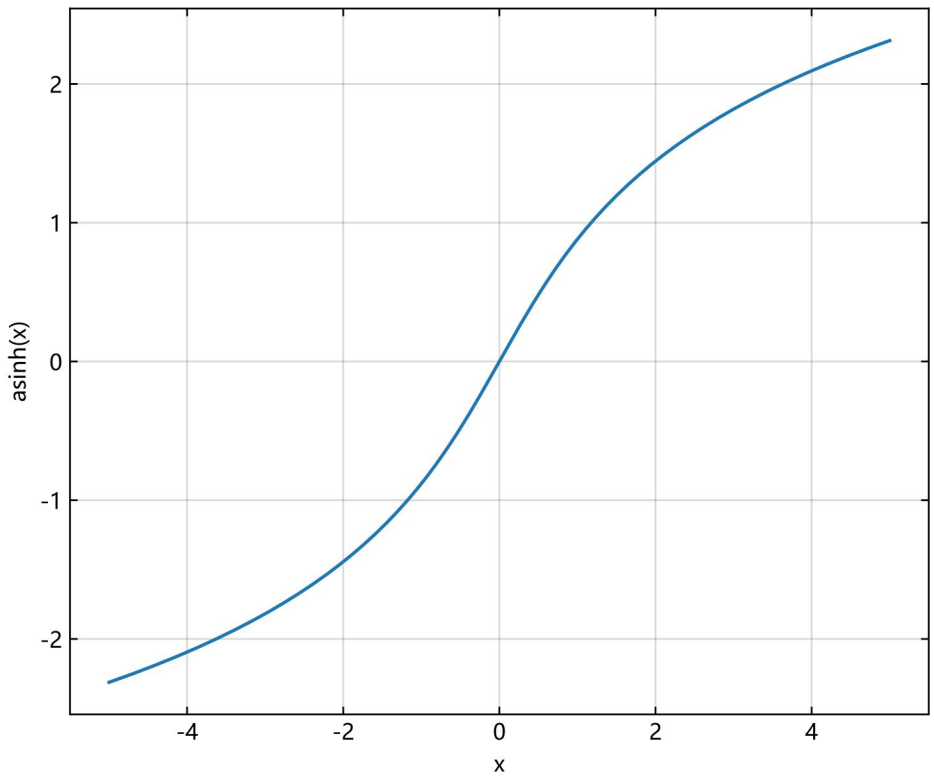
```
X = [2,-3,1+2im]
Y = asinh.(X)
```

```
Y = 3-element Vector{ComplexF64}:
1.4436354751788103 - 0.0im
-1.8184464592320668 - 0.0im
1.4693517443681852 + 1.063440023577752im
```

(2). 绘制反双曲正弦函数

在区间 −5≤x≤5 上绘制反双曲正弦函数。

```
x = -5:.01:5
plot(x,asinh.(x))
grid("on")
xlabel("x")
ylabel("asinh(x)")
```



3.2.1.2.8 cos

以弧度为单位的参数的余弦

A. 语法

```
Y = cos(X)
```

B. 说明

Y = cos(X) 返回 X 语法的每个元素的余弦。cos 函数按元素处理数组。该函数同时接受实数和复数输入。

对于 X 的实数值，cos(X) 返回区间 [-1, 1] 内的实数值。

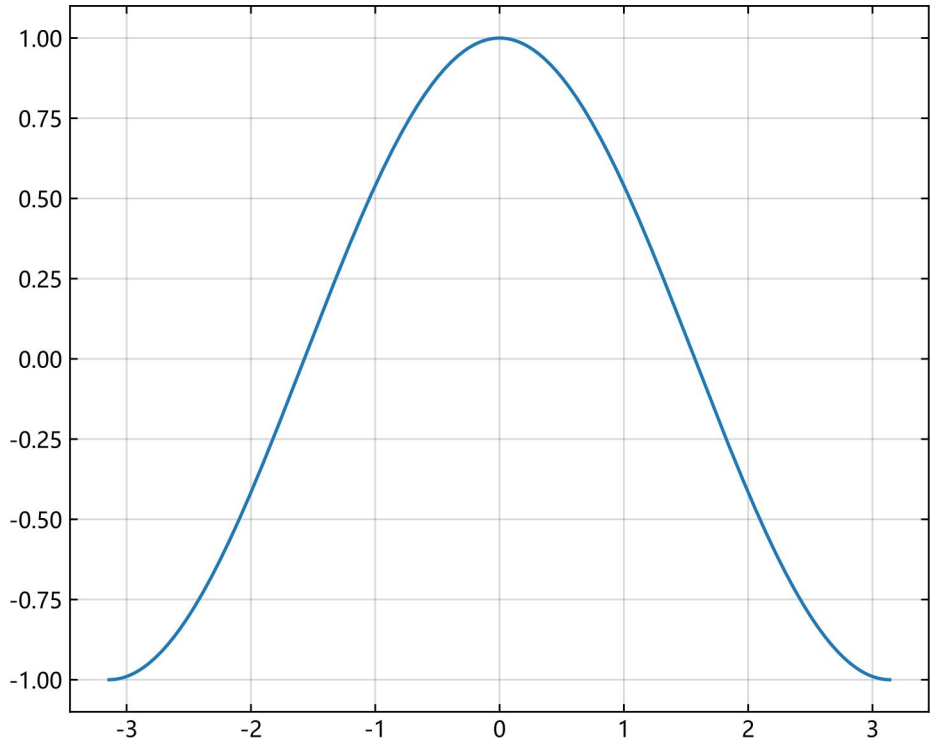
对于 X 的复数值，cos(X) 返回复数值。

C. 示例

(1). 绘制余弦函数图

在域 −π≤x≤π上绘制余弦函数。

```
x=-pi:0.01:pi
plot(x,cos.(x))
grid("on")
```



(2). 复角向量的余弦

计算复角在向量 x 中的余弦。

```
x = [-im,pi+im*pi/2,-1+im*4]
y = cos.(x)
```

```
y = 3-element Vector{ComplexF64}:
1.5430806348152437 + 0.0im
-2.5091784786580567 - 2.818278334591946e-16im
14.754701170483758 + 22.96367349919304im
```

3.2.1.2.9 cosd

以度为单位的参数的余弦

A. 语法

```
Y = cosd(X)
```

B. 说明

Y = cosd(X) 返回 X 的元素的余弦（以度为单位）。

C. 示例

(1). 90 度的余弦与 r/2 弧度的余弦相比较

在域 $-\pi \le x \le \pi$ 上绘制余弦函数。

```
cosd(90)
```

```
ans = 0.0
```

```
cos(pi/2)
```

```
ans = 6.123233995736766e-17
```

(2). 以度为单位指定的复数角的余弦

创建由三个复数角构成的数组并计算余弦。

```
z = [180+im 45+2im 10+3im]
y = cosd.(z)
```

```
y = 1×3 Matrix{ComplexF64}:
-1.00015-2.13752e-18im  0.707538-0.0246877im  0.986158-0.00909635im
```

3.2.1.2.10 cospi

准确计算 cos(X*pi)

A. 语法

```
Y = cospi(X)
```

B. 说明

Y = cospi(X) 计算 cos(X*pi) 而不显式计算 X*pi。此计算比 cos(X*pi) 更准确，因为 pi 的浮点值是 π 的近似值。尤其是：

对于奇整数，cospi(n/2) 正好为零。

对于整数，cospi(n) 为 +1 或 -1。

C. 示例

(1). 计算 π 的倍数的余弦

比较 cospi(X) 与 cos(X*pi) 的精确度。

创建一个由值组成的向量。

```
X = [0 1/2 1 3/2 2]
```

使用常规 cos 函数计算 X*pi 的余弦。

```
Y = cos.(X*pi)
```

```
Y = 1×5 Matrix{Float64}:
1.0  6.12323e-17  -1.0  -1.83697e-16  1.0
```

由于 pi 是 π 的实际值的浮点近似值，因此结果包含小的数值误差。例如，即使 cos(π/2)=0，Y(2) 也不会正好为零。

```
Y[2]
```

```
ans = 6.123233995736766e-17
```

使用 cospi 计算相同的值。在这种情况下，结果真正为零。

```
Z = cospi.(X)
```

```
Z = 1×5 Matrix{Float64}:
1.0  0.0  -1.0  0.0  1.0
```

```
Z[2]
```

```
ans = 0.0
```

3.2.1.2.11 acos

反余弦（以弧度为单位）

A. 语法

```
Y = acos(X)
```

B. 说明

Y = acos(X) 以弧度为单位返回 X 各元素的 反正弦 (sin)^-1。该函数同时接受实数和复数输入。

对于 X 在区间 [-1, 1] 内的实数值，acos(X) 返回区间 [-π/2, π/2] 内的值。

对于 X 在区间 [-1, 1] 之外的实数值以及 X 的复数值，acos(X) 返回复数值。

C. 示例

(1). 值的反余弦

求值的反余弦弦。

```
y = acos(0)
```

```
y = 1.5707963267948966
```

(2). 复数值向量的反余弦

计算向量 x 的各元素的反余弦。acos 函数按元素处理 x。

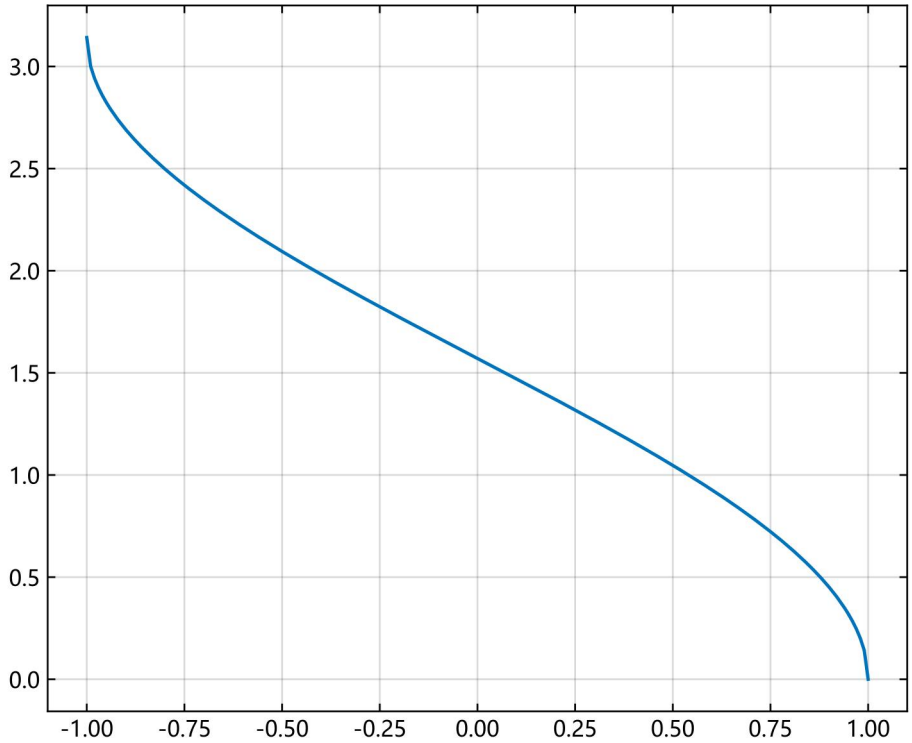
```
x = [0.5im,1+3im,-2.2+im]
y = acos.(x)
```

```
3-element Vector{ComplexF64}:
1.5707963267948966 - 0.48121182505960347im
1.2631926772641853 - 1.8641615441578825im
2.6798567434859244 - 1.5479846981255774im
```

(3). 绘制反余弦函数

在区间 −1≤x≤1 上绘制反余弦函数。

```
x = -1:.01:1
plot(x,acos.(x))
grid("on")
```



3.2.1.2.12 acosd

反余弦（以度为单位）

A. 语法

Y = acosd(X)

B. 说明

Y = acosd(X) 返回 X 的元素的反余弦 (cos)^-1（以度为单位）。该函数同时接受实数和复数输入。

C. 示例

(1). 0 的反余弦

确认零的反余弦正好是 90。

```
acosd(0)
```

```
ans = 90.0
```

(2). 复角的来回计算

说明后跟余弦值的反余弦函数会返回 X 的原始值。

```
cosd.(acosd.([2+0im 3+0im]))
```

```
ans = 1×2 Matrix{ComplexF64}:
2.0+0.0im  3.0+0.0im
```

acosd([2+0im 3+0im]) 返回两个复角，然后将这两个复角传递至 cosd 函数。cosd 返回原始值 2 和 3。

3.2.1.2.13 cosh

双曲余弦

A. 语法

```
Y = cosh(X)
```

B. 说明

Y = cosh(X) 返回 X 的元素的双曲余弦。cosh 函数按元素处理数组。该函数同时接受实数和复数输入。所有的角度都以弧度表示。

C. 示例

(1). 向量的双曲余弦

创建一个向量并计算每个值的双曲余弦。

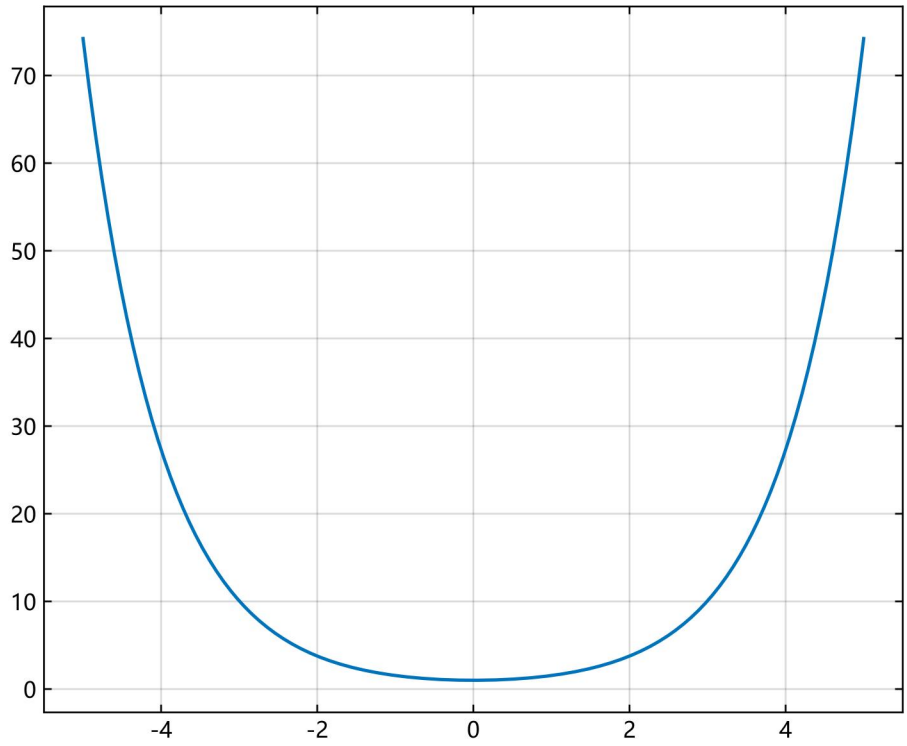
```
X = [0,pi,2*pi,3*pi];
Y = cosh.(X)
```

```
4-element Vector{Float64}:
1.0
11.591953275521519
267.7467614837482
6195.823944308106
```

(2). 双曲余弦图

在域 $-5 \le x \le 5$ 上绘制双曲余弦函数图。

```
x = -5:0.01:5
y = cosh.(x)
plot(x,y)
grid("on")
```



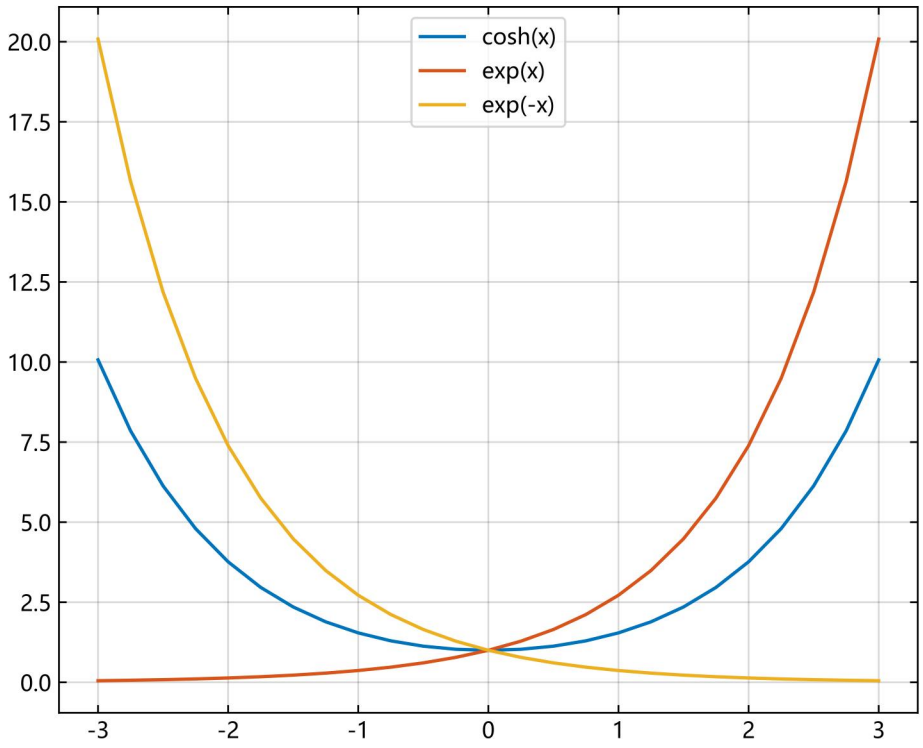
(3). 绘制双曲余弦函数和指数函数

双曲正弦满足恒等式 cosh(x)=(exp(x)−exp(−x))/2。换句话说，sinh(x) 是函数 exp(x) 和 exp(−x) 的平均值。通过绘制函数可对此进行验证。

创建一个向量，其中包含 -3 和 3 之间的值，步长为 0.25。计算并绘制 cosh(x)、exp(x) 和 exp(-x) 的值。与预期相符，cosh 曲线在 exp(x) 大时

为正，而在 exp(-x) 大时为负。

```
x = -3:0.25:3
y1 = cosh.(x)
y2 = exp.(x)
y3 = exp.(-x)
plot(x,y1,x,y2,x,y3)
grid("on")
legend(["cosh(x)","exp(x)","exp(-x)"])
```



3.2.1.2.14 acosh

反双曲余弦

A. 语法

```
Y = acosh(X)
```

B. 说明

Y = acosh(X) 返回 X 的元素的反双曲余弦。该函数同时接受实数和复数输入。所有的角度都以弧度表示。

C. 示例

(1). 向量的反双曲余弦

计算向量 X 的各元素的反双曲余弦。acosh 函数按元素处理 X。

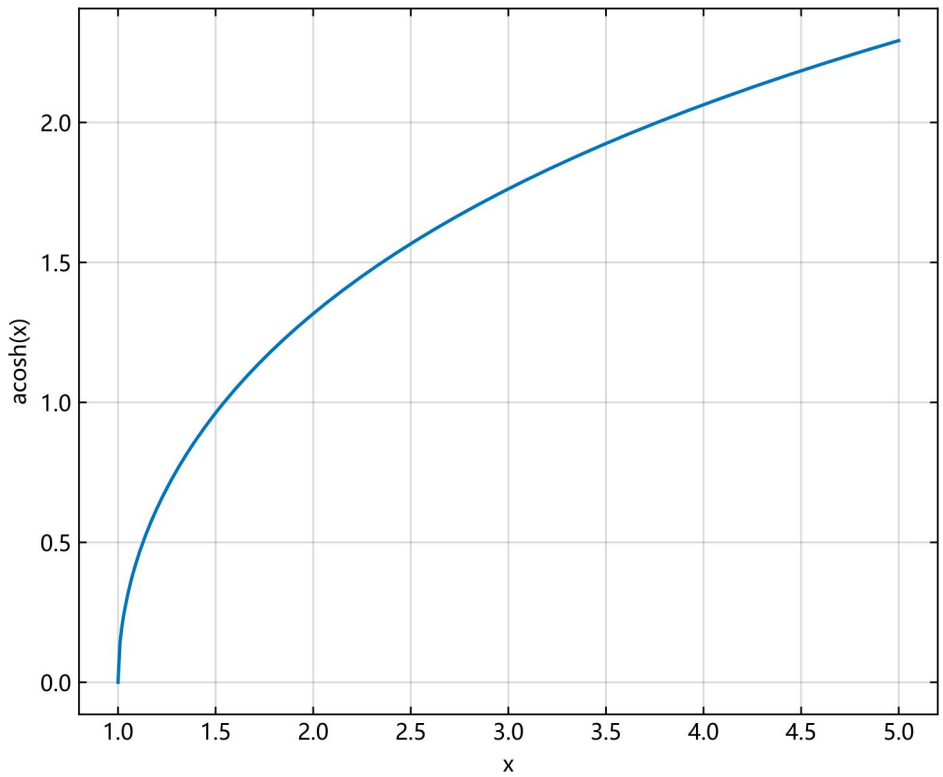
```
X = [2,-3,1+2im]
Y = acosh.(X)
```

```
3-element Vector{ComplexF64}:
1.3169578969248166 + 0.0im
-1.762747174039086 + 3.141592653589793im
1.5285709194809982 + 1.1437177404024204im
```

(2). 绘制反双曲余弦函数

在区间 −5≤x≤5 上绘制反双曲余弦函数。

```
x = 1:0.01:5
plot(x,acosh.(x))
grid("on")
xlabel("x")
ylabel("acosh(x)")
```



3.2.1.2.15 tan

以弧度为单位的参数的正切

A. 语法

```
Y = tan(X)
```

B. 说明

Y = tan(X) 返回 X 的每个元素的正切。tan 函数按元素处理数组。该函数同时接受实数和复数输入。

对于 X 的实数值，tan(X) 返回区间 [-∞，∞] 内的实数值。

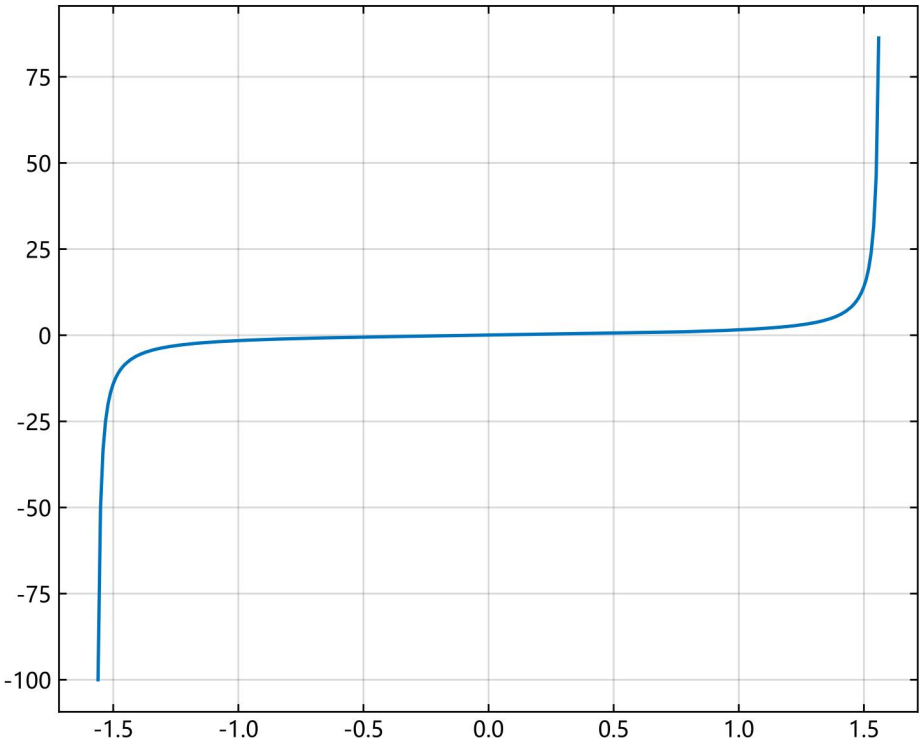
对于 X 的复数值，tan(X) 返回复数值。

C. 示例

(1). 绘制正切函数图

在域 −π≤x≤π 上绘制正切函数。

```
x = (-pi/2+0.01):0.01:(pi/2-0.01)
plot(x,tan.(x))
grid("on")
```



(2). 复角向量的正切

计算复角在向量 x 中的正切。

```
x = [-im,pi+im*pi/2,-1+im*4]
y = tan.(x)
```

```
3-element Vector{ComplexF64}:
0.0 - 0.7615941559557649im
-1.9451260219143916e-17 + 0.9171523356672743im
-0.0006102409213762599 + 1.0002790562344654im
```

3.2.1.2.16 tand

以度为单位的参数的正切

A. 语法

```
Y = tand(X)
```

B. 说明

Y = tand(X) 返回 X 的元素（以度表示）的正切。

C. 示例

(1). 90 度正切与 r/2 弧度正切比较

```
tand(90)
```

```
ans = Inf
```

```
tan(pi/2)
```

```
ans = 1.6331239353195370e16
```

tand(90) 为无穷大，而 tan(pi/2) 为有穷大。

(2). 以度为单位指定的复角向量的正切

创建由三个复数角构成的数组并计算正切。

```
z = [180+im,15+2im, 10+3im];
y = tand.(z)
```

```
3-element Vector{ComplexF64}:
-1.22427382614427e-16 + 0.017451520543541 53im
0.2675995782989953 + 0.03739430836996746im
0.17582949238290824 + 0.05393393751573375 6im
```

3.2.1.2.17 atan

反正切（以弧度为单位）

A. 语法

```
Y = atan(X)
Y = atan(Y,X1)
```

B. 说明

Y = atan(X) 以弧度为单位返回 X 各元素的 反正切 (tan)^-1。该函数同时接受实数和复数输入。

对于 X 的实数值，atan(X) 返回区间 [-π/2, π/2] 内的值。

对于 X 的复数值，atan(X) 返回复数值。

P = atan(Y,X1) 返回 Y 和 X1 的四象限反正切 (tan^{-1}) ，该值必须为实数。

atan 函数遵循当 x 在数学上为零（或者为 0 或 -0）时 atan(x,x) 返回 0 的约定。

C. 示例

(1). 值的反正切

求值的反正切。

```
atan(0.8)
```

```
ans = 0.6747409422235527
```

(2). 复数值向量的反正切

计算向量 x 的各元素的反正切。atan 函数按元素处理 x。

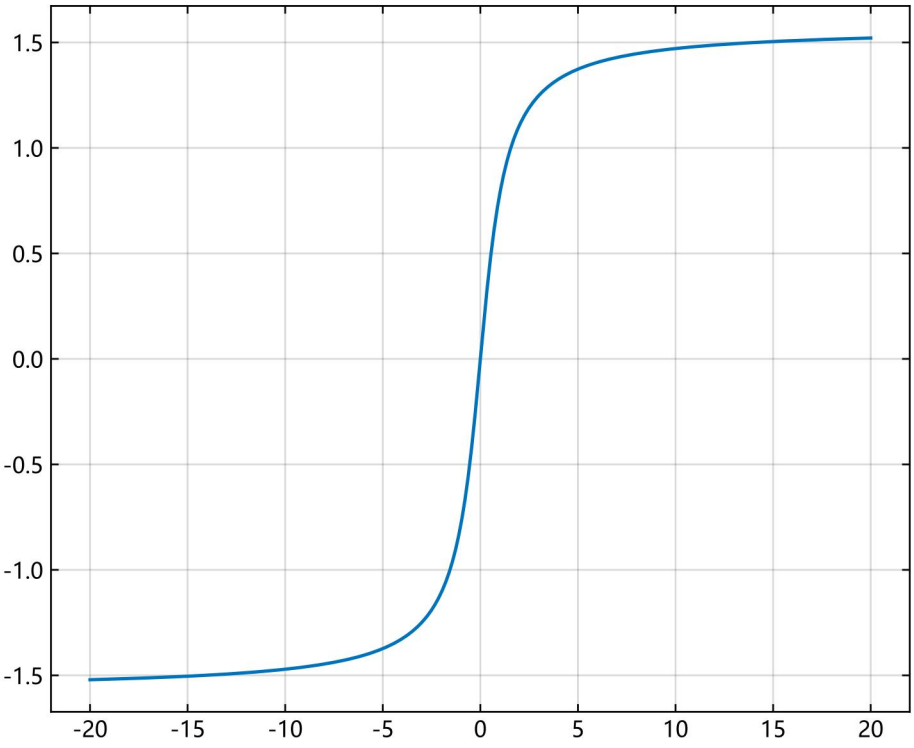
```
x = [0.5im,1+3im,-2.2+im]
y = atan.(x)
```

```
3-element Vector{ComplexF64}:
0.0 + 0.5493061443340549im
1.4614618538579256 + 0.3059438579055289im
-1.201888796734664 + 0.15059303898025284im
```

(3). 绘制反正切函数

在区间 −1≤x≤1 上绘制反正切函数。

```
x = -20:0.01:20
plot(x,atan.(x))
grid("on")
```



(4). 计算点的四象限反正切

计算点 y＝4、x＝-3 的四象限反正切。

```
atan(4,-3)
```

```
ans = 2.214297435588181
```

(5). 将复数转换为极坐标

将 4＋3im 转换为极坐标。

```
z = 4 + 3im;
r = abs(z)
```

```
r = 5.0
```

```
theta = atan(imag(z),real(z))
```

```
theta = 0.6435011087932844
```

半径 r 和角 theta 是 4＋3i 的极坐标表示形式。

或者，使用 angle 计算 theta。

```
theta = angle(z)
```

```
0.6435011087932844
```

将 r 和 theta 转换回原始复数。

```
z = r*exp(im*theta)
```

```
4.0 + 3.0im
```

(6). 绘制四象限反正切图

绘制 atan(Y,X)（对于 $-4 < Y < 4$ 和-4<X<4）。

定义要在其上绘制的区间。

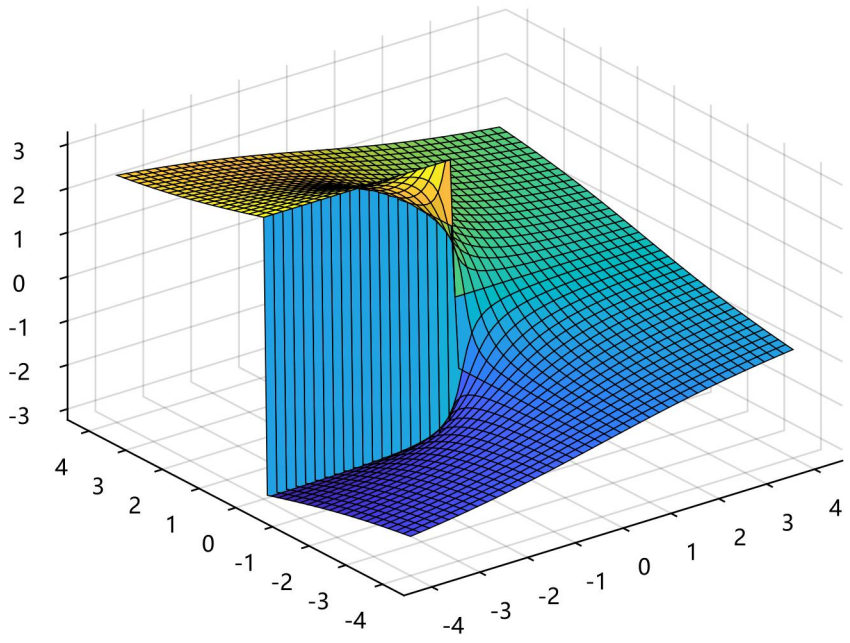
```
using TyBase
X,Y = meshgrid2([-4:0.1:4...],[-4:0.1:4...])
```

计算 atan2(Y,X) 在此区间上的值。

```
P = atan.(Y,X)
```

使用 surf 生成函数的曲面图。请注意，plot 绘制存在于 Y=0 的不连续性（对于所有 X<0）。

```
surf(X,Y,P)
```



3.2.1.2.18 atand

反正切（以度为单位）

A. 语法

```
Y = atand(X)
```

B. 说明

Y = atand(X) 返回 X 的元素的反正切 (tan)^-1（以度为单位）。该函数同时接受实数和复数输入。

对于 X 的实数值，atand(X) 返回区间 [-90, 90] 中的值。

对于 X 的复数值，atand(X) 返回复数值。

C. 示例

(1). 向量的反正切

```
x = [-50 ,-20, 0, 20, 50]
y = atand.(x)
```

```
5-element Vector{Float64}:
-88.8542371618249
-87.13759477388825
0.0
87.13759477388825
88.8542371618249
```

传递向量、矩阵或 N 维数组时，按元素执行 atand 运算。

(2). 复数值的反正切

说明后跟正切值的反正切函数会返回 X 的原始值。

```
atand(10+im)
```

```
ans = 84.34503376298991 + 0.5617953332940616im
```

3.2.1.2.19 tanh

双曲正切

A. 语法

```
Y = tanh(X)
```

B. 说明

Y = tanh(X) 返回 X 的元素的双曲正切。tanh 函数按元素处理数组。该函数同时接受实数和复数输入。所有的角度都以弧度表示。

C. 示例

(1). 向量的双曲正切

创建一个向量并计算每个值的双曲正切。

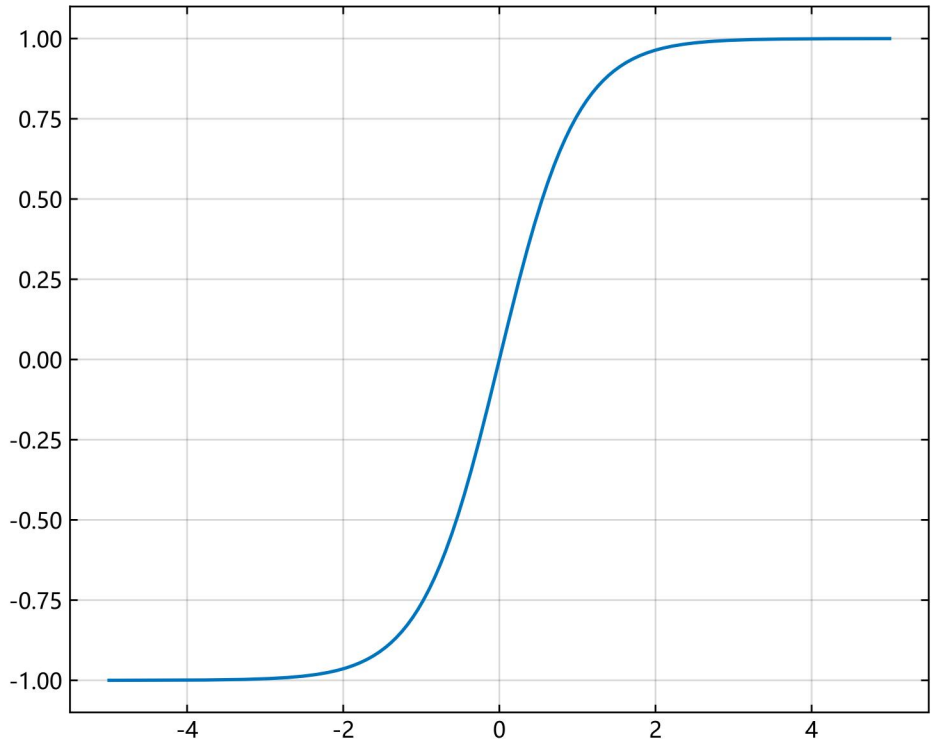
```
X = [0,pi,2*pi,3*pi];
Y = tanh.(X)
```

```
4-element Vector{Float64}:
0.0
0.99627207622075
0.9999930253396107
0.9999999869751758
```

(2). 双曲正切函数图

在域 −5≤x≤5 上绘制双曲正切函数图。

```
x = -5:0.01:5
y = tanh.(x)
plot(x,y)
grid("on")
```



3.2.1.2.20 atanh

反双曲正切

A. 语法

```
Y = atanh(X)
```

B. 说明

Y = atanh(X) 返回 X 的元素的反双曲正切。该函数同时接受实数和复数输入。所有的角度都以弧度表示。

C. 示例

(1). 向量的反双曲正切

计算向量 X 的各元素的反双曲正切。atanh 函数按元素处理 X。

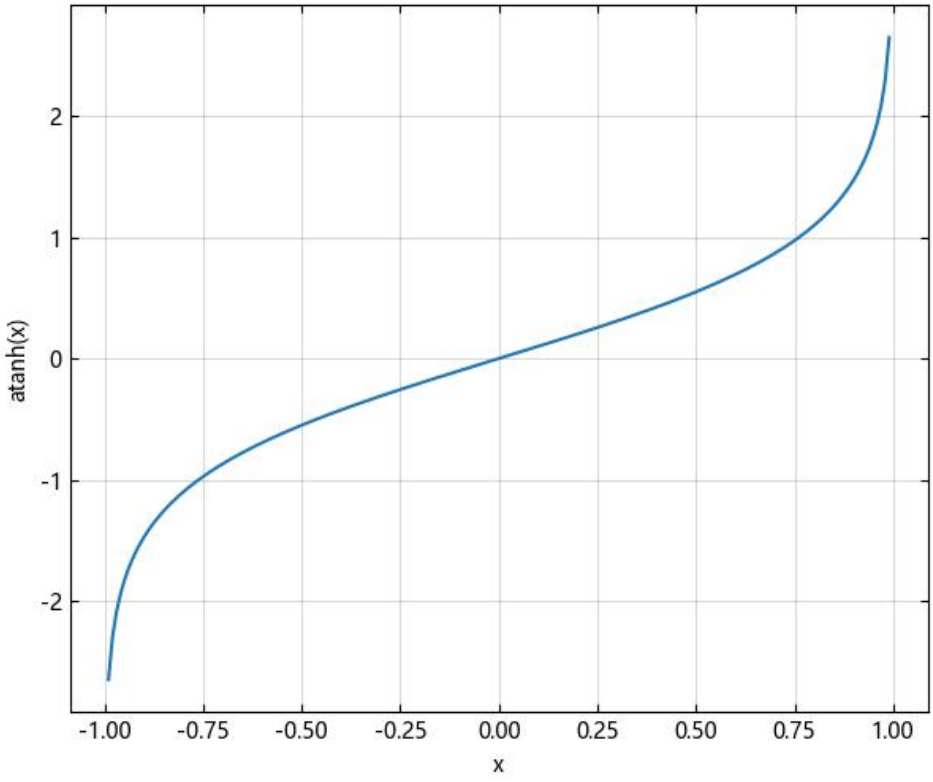
```
X = [2,-3,1+2im]
Y = atanh.(X)
```

```
3-element Vector{ComplexF64}:
0.5493061443340549 + 1.5707963267948966im
-0.34657359027997264 + 1.5707963267948966im
0.1732867951399863 + 1.1780972450961724im
```

(2). 绘制反双曲正切函数

在区间 −5≤x≤5 上绘制反双曲正切函数。

```
x = -0.99:0.01:0.99
plot(x,atanh.(x))
grid("on")
xlabel("x")
ylabel("atanh(x)")
```



3.2.1.2.21 csc

输入角的余割（以弧度为单位）

A. 语法

```
Y = csc(X)
```

B. 说明

Y = csc(X) 返回 X 的元素的余割。csc 函数按元素处理数组。该函数同时接受实数和复数输入。

对于 X 的实数值，csc(X) 返回区间 [-∞, - 1] 和 [1, ∞] 内的实数值。

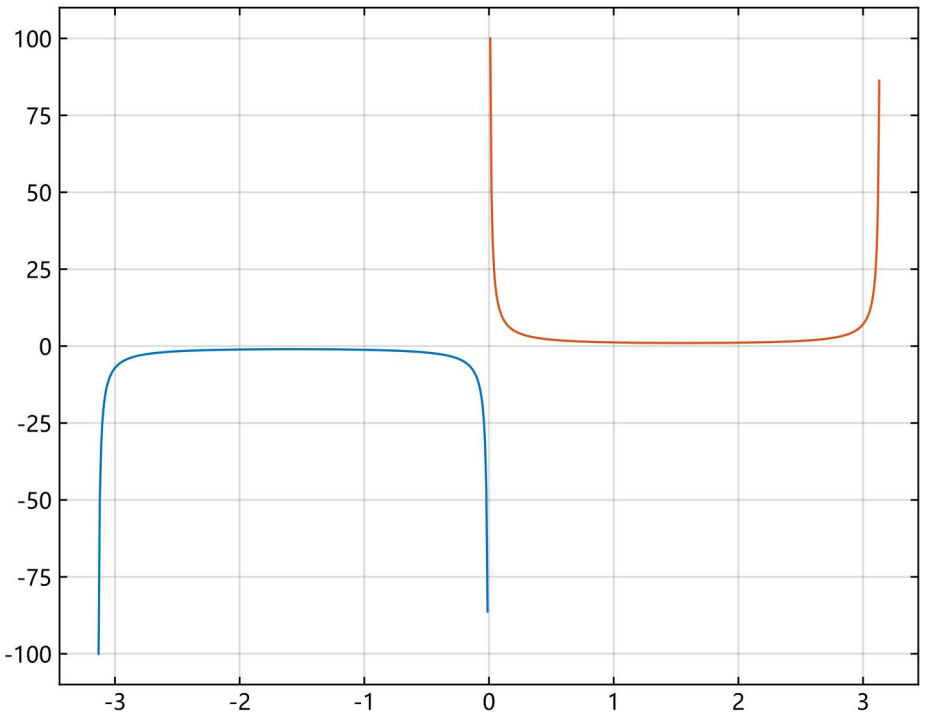
对于 X 的复数值，csc(X) 返回复数值。

C. 示例

(1). 绘制余割函数图

在域 $-\pi < x < 0$ 和 $0 < x < \pi$ 上绘制余割函数，如下所示。

```
x1 = -pi+0.01:0.01:-0.01
x2 = 0.01:0.01:pi-0.01
plot(x1,csc.(x1),x2,csc.(x2))
grid("on")
```



(2). 复角向量的余割

计算复角在向量 x 中的余割。

```
x = [-im , pi+im*pi/2 , -1+im*4]
y = csc.(x)
```

```
y = 3-element Vector{ComplexF64}:
0.0 + 0.8509181282393216im
5.802248768373112e-17 + 0.4345372080946958im
-0.030825887576699802 - 0.019779799572192686im
```

3.2.1.2.22 cscd

以度为单位的参数的余割

A. 语法

```
Y = csc(X)
```

B. 说明

Y=cscd(X)返回 X 的元素的余割。(以度为单位)

对于 X 的实数值，csc(X) 返回区间 [-∞, - 1] 和 [1, ∞] 内的实数值。

对于 X 的复数值，csc(X) 返回复数值。

C. 示例

(1). 180 度的余割与 r 弧度的余割相比较

cscd(180) 为无穷大，而 csc(pi) 为有穷大。

```
cscd(180)
```

```
ans = Inf
```

```
csc(pi)
```

```
ans = 8.165619676597685e15
```

(2). 以度为单位指定的复数角向量的余割

```
z = [35+im , 15+2im , 10+3im]
y = cscd.(z)
```

```
y = 3-element Vector{ComplexF64}:
1.742099141213276 - 0.04341895408538742im
3.7969642702299913 - 0.4944416872728512im
5.285657866047403 - 1.5681307543661496im
```

3.2.1.2.23 acsc

反余割（以弧度为单位）

A. 语法

```
Y = acsc(X)
```

B. 说明

Y = acsc(X) 以弧度为单位返回 X 各元素的 反余割(csc-1)。该函数同时接受实数和复数输入。

对于 X 在区间 [-∞, -1] 和 [1, ∞] 中的实数值，acsc(X) 返回区间 [-π/2, π/2] 中的实数值。

对于 X 在区间 (-1, 1) 内的实数值以及 X 的复数值，acsc(X) 返回复数值。

C. 示例

(1). 值的反余割

求值的反余割。

```
y = acsc(3)
```

```
y = 0.3398369094541219
```

(2). 复角向量的反余割

计算向量 x 的各元素的反余割。acsc 函数按元素处理 x。

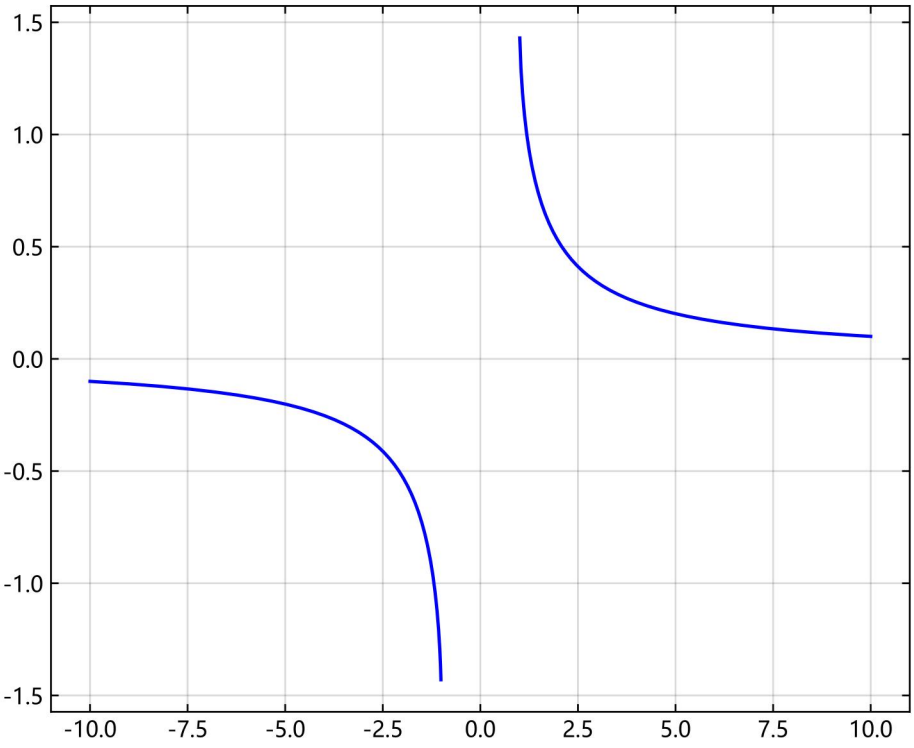
```
x = [0.5im,1+3im,-2.2+im]
y = acsc.(x)
```

```
y = 3-element Vector{ComplexF64}:
0.0 - 1.4436354751788103im
0.0958929592754537 - 0.29699902340838985im
-0.3795146564184349 - 0.18332187294886487im
```

(3). 绘制反余割函数

在区间 −10≤x<−1 和 1<x≤10 上绘制反余割函数。

```
hold("on")
x1 = -10:.01:-1.01
x2 = 1.01:0.01:10
plot(x1,acsc.(x1),"b")
plot(x2,acsc.(x2),"b")
grid("on")
```



3.2.1.2.24 acscd

反余割（以度为单位）

A. 语法

```
Y = acscd(X)
```

B. 说明

Y = acscd(X) 返回 X 的元素的反余割 (cosec-1)（以度为单位）。该函数同时接受实数和复数输入。

对于 X 在区间 [-∞, -1] 和 [1, ∞] 中的实数值，acscd(X) 返回范围 [-90, 90] 中的值。

对于 X 在区间 (-1, 1) 内的实数值以及 X 的复数值，acscd(X) 返回复数值。

C. 示例

(1). 向量的反余割

```
x = [20,10,-Inf]
y = acscd.(x)
```

```
y = 3-element Vector{Float64}:
2.8659839825988622
5.739170477266787

-0.0
```

传递向量、矩阵或 N 维数组时，按元素执行 acscd 运算。

```
cos(pi/2)
```

```
ans = 6.123233995736766e-17
```

(2). 复数值的反余割

```
acscd(1+im)
```

```
ans = 25.913646186493875 - 30.403290974821864im
```

3.2.1.2.25 csch

反余割（以弧度为单位）

A. 语法

```
Y = csch(X)
```

B. 说明

Y = csch(X) 返回 X 的元素的双曲余割。csch 函数按元素处理数组。该函数同时接受实数和复数输入。所有的角度都以弧度表示。

C. 示例

(1). 向量的双曲余割

创建一个向量并计算每个值的双曲余割。

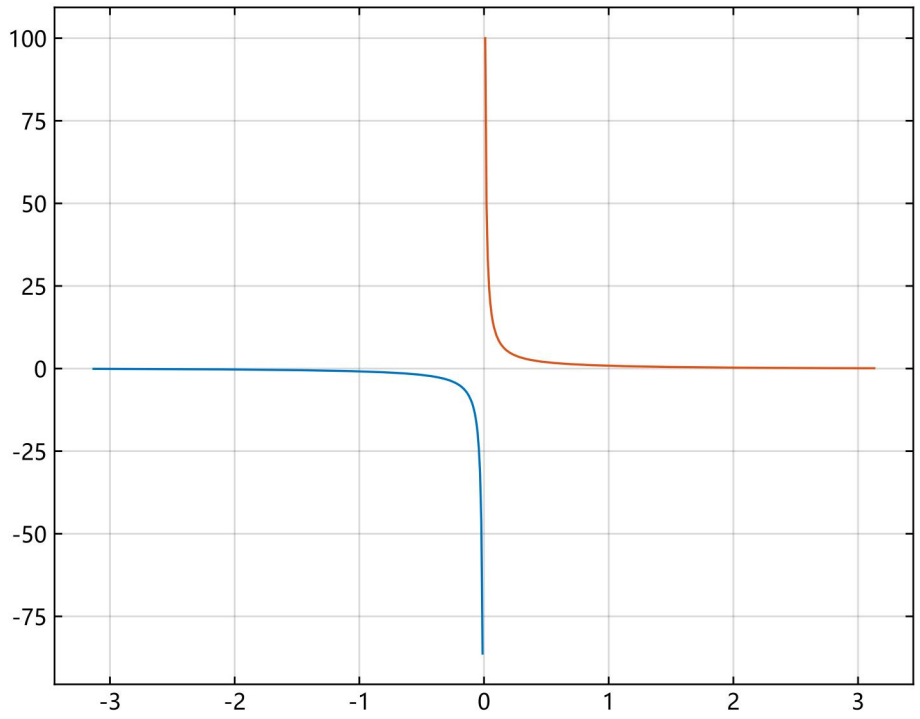
```
X = [0,pi,2*pi,3*pi]
Y = csch.(X)
```

```
Y = 4-element Vector{Float64}:
Inf
0.08658953753004694
0.003734898488285673
0.0001613990361917063
```

(2). 双曲余割图

在域 $-\pi < x < 0$ 和 $0 < x < \pi$ 上绘制双曲余割函数。

```
x1 = -pi+0.01:0.01:-0.01
x2 = 0.01:0.01:pi-0.01
y1 = csch.(x1)
y2 = csch.(x2)
plot(x1,y1,x2,y2)
grid("on")
```



3.2.1.2.26 acsch

反双曲余割

A. 语法

```
Y = acsch(X)
```

B. 说明

Y = acsch(X) 返回 X 各元素的反双曲余割。该函数同时接受实数和复数输入。所有的角度都以弧度表示。

C. 示例

(1). 向量的反双曲余割

计算向量 X 各元素的反双曲余割。acsch 函数按元素处理 X。

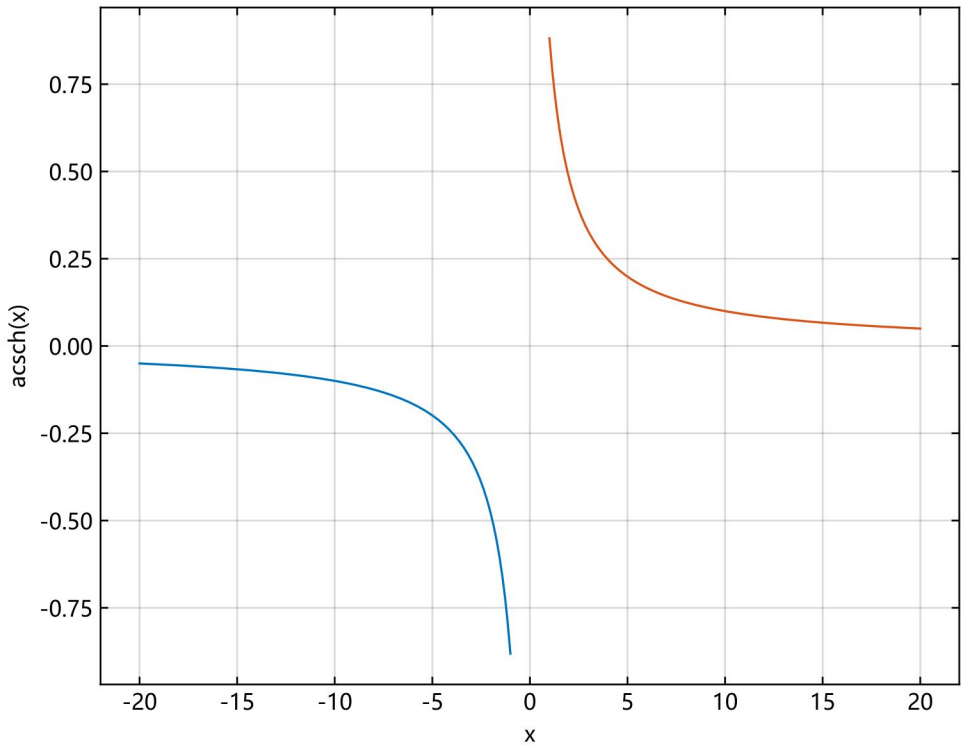
```
x = [2,-3,1+2im]
y = acsch.(x)
```

```
y = 3-element Vector{ComplexF64}:
0.48121182505960347 - 0.0im
-0.32745015023725843 - 0.0im
0.21561241855582966 - 0.40158639166780613im
```

(2). 绘制反双曲余割函数

在 −20≤x≤−1 和 1≤x≤20 区间上绘制反双曲余割函数。

```
x1 = -20:0.01:-1
x2 = 1:0.01:20
plot(x1,acsch.(x1),x2,acsch.(x2))
grid("on")
xlabel("x")
ylabel("acsch(x)")
```



3.2.1.2.27 sec

角的正割（以弧度为单位）

A. 语法

```
Y = sec(X)
```

B. 说明

Y = sec(X) 返回 X 的元素的正割。sec 函数按元素处理数组。该函数同时接受实数和复数输入。

对于 X 的实数值，sec(X) 返回区间 [-∞, - 1] 和 [1, ∞] 内的实数值。

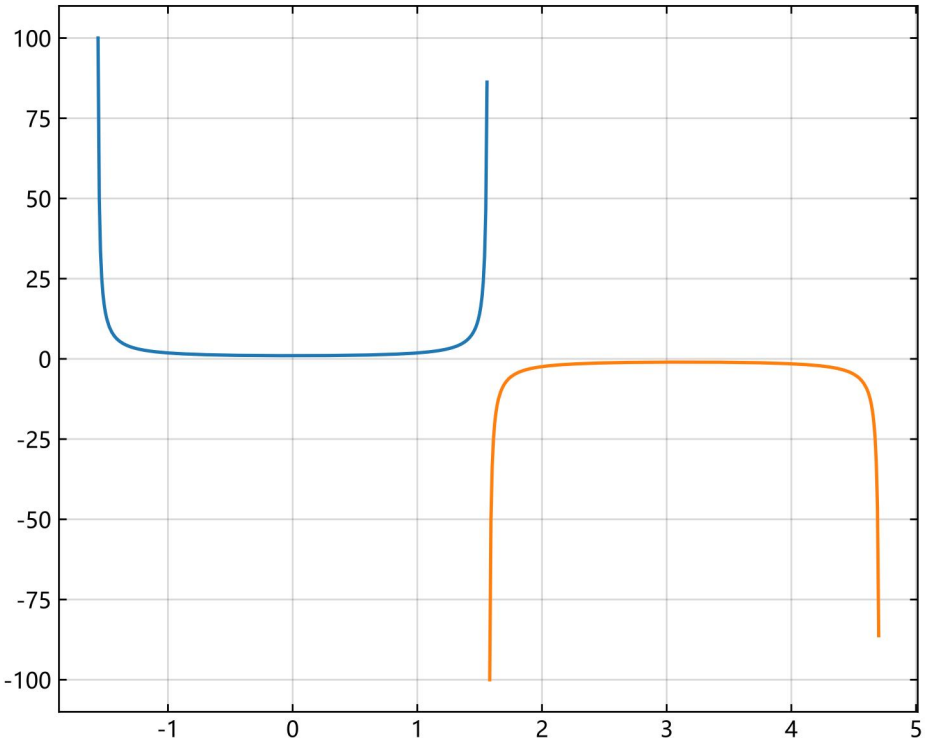
对于 X 的复数值，sec(X) 返回复数值。

C. 示例

(1). 绘制正割函数图

在域 $-\pi/2 < x < \pi/2$ 和 $\pi/2 < x < 3\pi/2$ 上绘制正割函数图。

```
x1 = (-pi/2)+0.01:0.01:(pi/2)-0.01
x2 = pi/2 + 0.01:0.01:(3*pi/2) - 0.01
plot(x1,sec.(x1),x2,sec.(x2))
grid("on")
```



(2). 复角向量的正割

计算复角在向量 x 中的正割。

```
x = [-im,pi+im*pi/2,-1+im*4]
y = sec.(x)
```

```
y = 3-element Vector{ComplexF64}:
0.6480542736638855 - 0.0im
-0.3985368153383867 + 4.476316379080944e-17im
0.019804130466616835 - 0.03082241928969544im
```

3.2.1.2.28 secd

参数的正割，以度为单位

A. 语法

```
Y = secd(X)
```

B. 说明

Y = secd(X) 返回 X 的元素的正割（以度为单位）。

C. 示例

(1). 90 度的正割与 r/2 弧度的正割相比较

```
secd(90)
```

```
ans = Inf
```

```
sec(pi/2)
```

```
ans = 1.6331239353195370e16
```

secd(90) 为无穷大，而 sec(pi/2) 为有穷大。

(2). 复数角向量的正割，以度为单位指定

计算复角在向量 x 中的正割。

```
x = [35+im , 15+2*im , 10+3*im]
y = secd.(x)
```

```
y = 3-element Vector{ComplexF64}:
1.2204064457718822 + 0.014912983849932540im
1.034555342454746 + 0.009672465783070086im
1.0139500056193251 + 0.009352706546720315im
```

3.2.1.2.29 asec

反正割（以弧度为单位）

A. 语法

```
Y = asec(X)
```

B. 说明

Y = asec(X) 以弧度为单位返回 X 各元素的 反正割 (sec-1)。该函数同时接受实数和复数输入。

对于 X 在区间 [-∞, -1] 和 [1, ∞] 中的实数值，asec(X) 返回区间 [0, π] 中的值。

对于 X 在区间 (-1, 1) 内的实数值以及 X 的复数值，asec(X) 返回复数值。

C. 示例

(1). 值的反正割

求值的反正割。

```
asec(-2.8)
```

```
ans = 1.9360035480852633
```

(2). 复数值向量的反正割

计算向量 x 的各元素的反正割。asec 函数按元素处理 x。

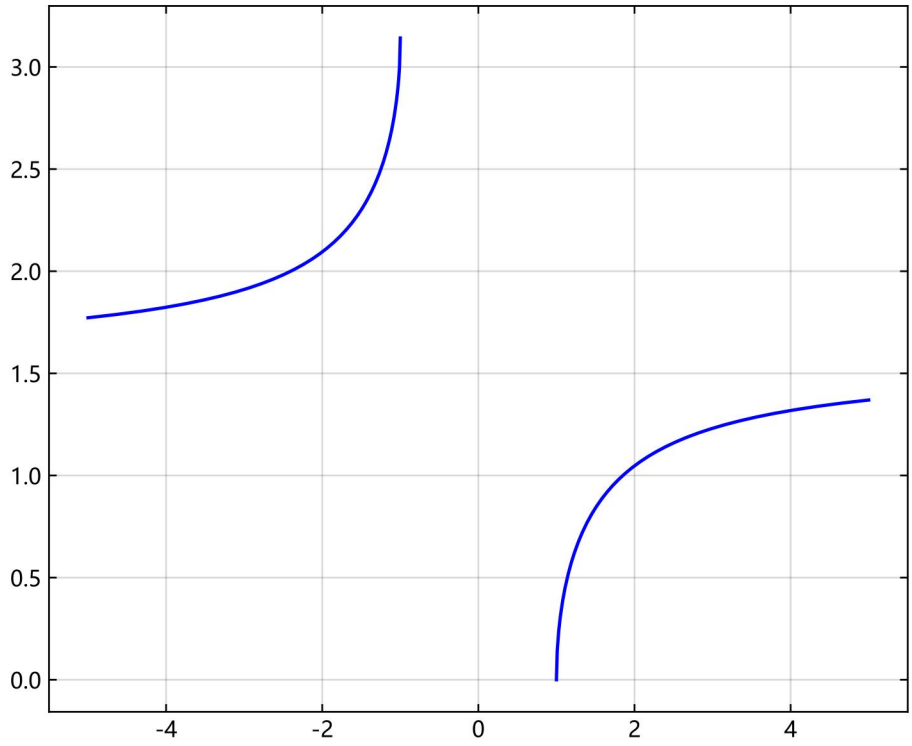
```
x = [0.5im , 1 + 3im , -2.2+im]
y = asec.(x)
```

```
y = 3-element Vector{ComplexF64}:
1.5707963267948966 + 1.4436354751788103im
1.474903367519443 + 0.29699902340838985im
1.9503109832133316 + 0.18332187294886487im
```

(3). 绘制反正割函数

在区间 −5≤x≤−1 和 1≤x≤5 上绘制反正割函数。

```
x1 = -5:0.01:-1
x2 = 1:0.01:5
hold("on")
plot(x1,asec.(x1),"b")
plot(x2,asec.(x2),"b")
grid("on")
```



3.2.1.2.30 sech

双曲正割

A. 语法

```
Y = sech(X)
```

B. 说明

Y = sech(X) 返回 X 的元素的双曲正割。sech 函数按元素处理数组。该函数同时接受实数和复数输入。所有的角度都以弧度表示。

C. 示例

(1). 向量的双曲正割

创建一个向量并计算每个值的双曲正割。

```
X = [0,pi,2*pi,3*pi]
Y = sech.(X)
```

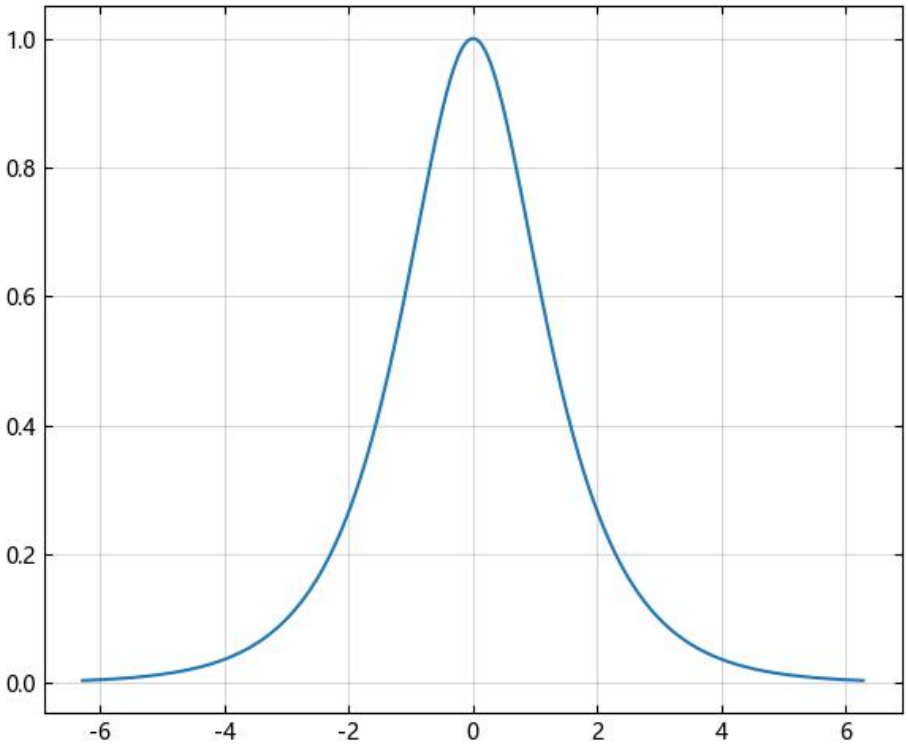
```
4-element Vector{Float64}:
```

```
1.0
0.08626673833405443
0.003734872438637128
0.00016139903408951222
```

(2). 双曲正割函数图

在域 $-2\pi \le x \le 2\pi$ 上绘制双曲正割函数。

```
x = -2*pi:0.01:2*pi
y = sech.(x)
plot(x,y)
grid("on")
```



3.2.1.2.31 asech

反双曲正割

A. 语法

```
Y = asech(X)
```

B. 说明

Y = asech(X) 返回 X 各元素的反双曲正割。该函数同时接受实数和复数输入。所有的角度都以弧度表示。

C. 示例

(1). 向量的反双曲正割

计算向量 X 各元素的反双曲正割。asech 函数按元素处理 X。

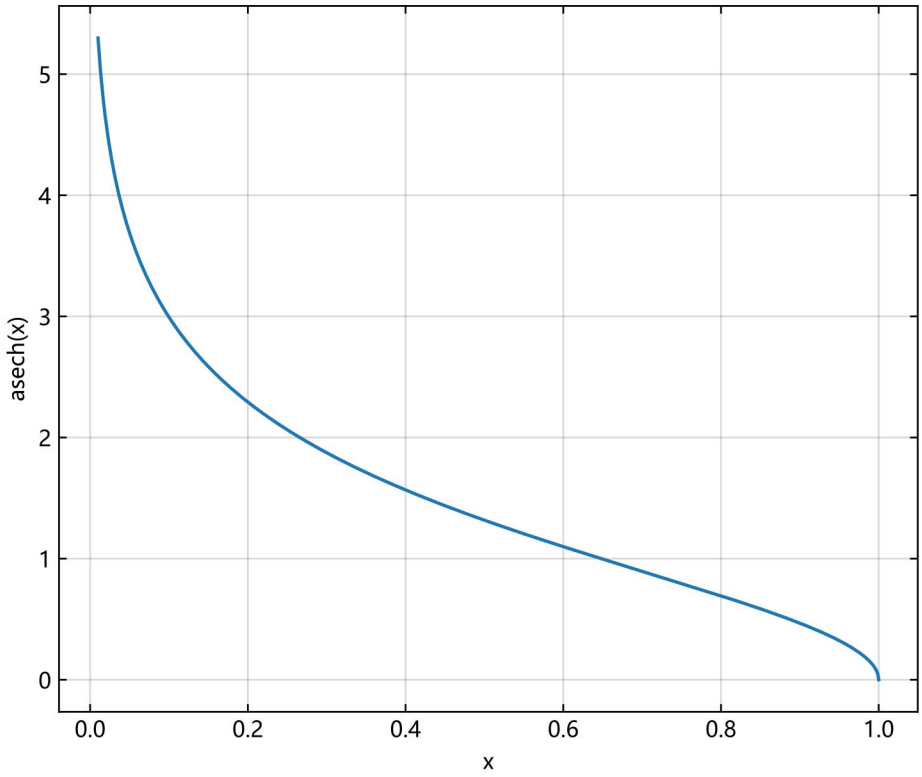
```
x = [2,-3,1+2im]
y = asech.(x)
```

```
3-element Vector{ComplexF64}:
0.0 - 1.0471975511965979im
0.0 - 1.9106332362490184im
0.396568230112329 - 1.3844782726870815im
```

(2). 绘制反双曲正割函数

在 $0.01 \le x \le 1$ 区间上绘制反双曲正割函数。

```
x = 0.01:0.001:1
plot(x,asech.(x))
grid("on")
xlabel("x")
ylabel("asech(x)")
```



3.2.1.2.32 cot

角的余切（以弧度为单位）

A. 语法

```
Y = cot(X)
```

B. 说明

Y = cot(X) 以弧度为单位返回 X 各元素的余切。cot 函数按元素处理数组。该函数同时接受实数和复数输入。

对于 X 的实数值，cot(X) 返回区间 [-Inf, Inf] 内的值。

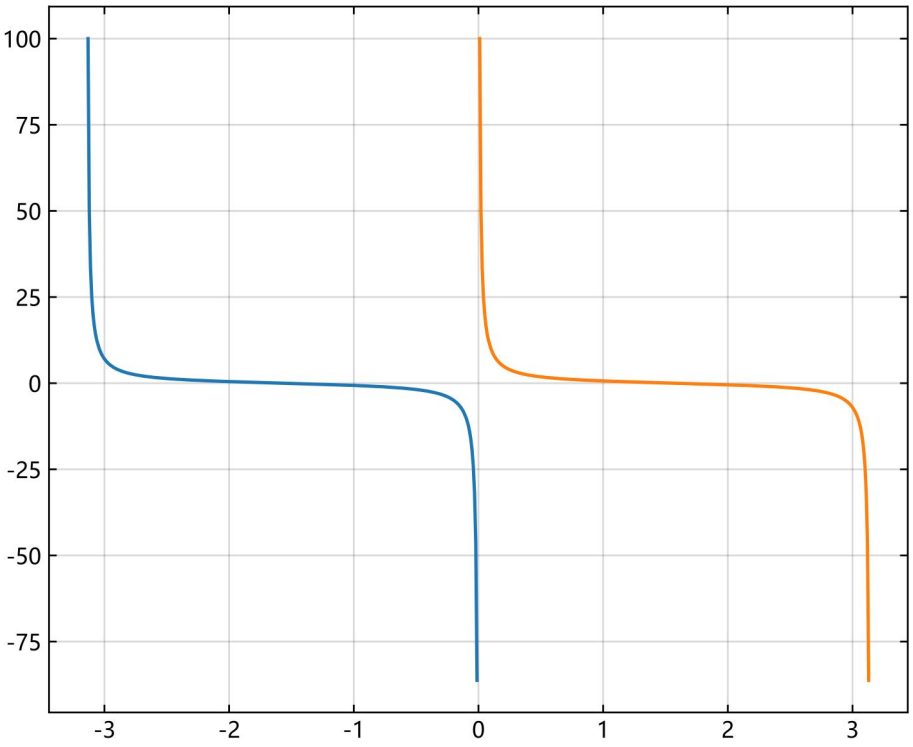
对于 X 的复数值，cot(X) 返回复数值。

C. 示例

(1). 绘制余切函数图

在域 $-\pi < x < 0$ 和 $0 < x < \pi$ 上绘制余切函数。

```
x1 = -pi+0.01:0.01:-0.01;
x2 = 0.01:0.01:pi-0.01;
plot(x1,cot.(x1),x2,cot.(x2))
grid("on")
```



(2). 复角向量的余切

计算复角在向量 x 中的余切。

```
x = [-im,pi+im*pi/2,-1+im*4];
y = cot.(x)
```

```
3-element Vector{ComplexF64}:
0.0 + 1.3130352854993315im
-2.312409745948497e-17 - 1.0903314107273683im
-0.0006099002538222277 - 0.9997206495339308im
```

3.2.1.2.33 cotd

以度为单位的参数的余切

A. 语法

```
Y = cotd(X)
```

B. 说明

Y = cotd(X) 返回 X 的元素（以度表示）的余切。

C. 示例

(1). 90 到 180 度范围的角的余切

创建一个由 90° 及下一个更小或更大的双精度数字组成的输入角向量。然后计算余切。

```
x1 = [90-eps(90.0),90,90+eps(90.0)]
y1 = cotd.(x1)
```

```
3-element Vector{Float64}:
2.4802620430283604e-16
0.0
-2.4802620430283604e-16
```

当输入角恰好为 90° 时，cotd 返回零。下一个更小的双精度角处的计算返回正结果。同样，当输入角是下一个大于 90 的双精度数时，余切值为负。对于接近 180° 的输入角，行为是类似的。

```
x2 = [180-eps(180.0) ,180 ,180+eps(180.0)];
y2 = cotd.(x2)
```

```
 3-element Vector{Float64}:
-2.015916025507966e15
-Inf
2.015916025507966e15
```

(2). 以度为单位指定的复角向量的余切

```
x = 35+5im;
y = cotd(x)
```

```
1.3957570235626786 - 0.26055777260535371m
```

3.2.1.2.34 acot

反余切（以弧度为单位）

A. 语法

```
Y = acot(X)
```

B. 说明

Y = acot(X) 以弧度为单位返回 X 各元素的 反余切 (tan)^-1。该函数同时接受实数和复数输入。

对于 X 的实数值，acot(X) 返回区间 [-π/2, π/2] 内的值。

对于 X 的复数值，acot(X) 返回复数值。

C. 示例

(1). 值的反余切

求值的反余切。

```
acot(2.6)
```

```
ans = 0.36717383381821916
```

(2). 复数值向量的反余切

计算向量 x 的各元素的反余切。acot 函数按元素处理 x。

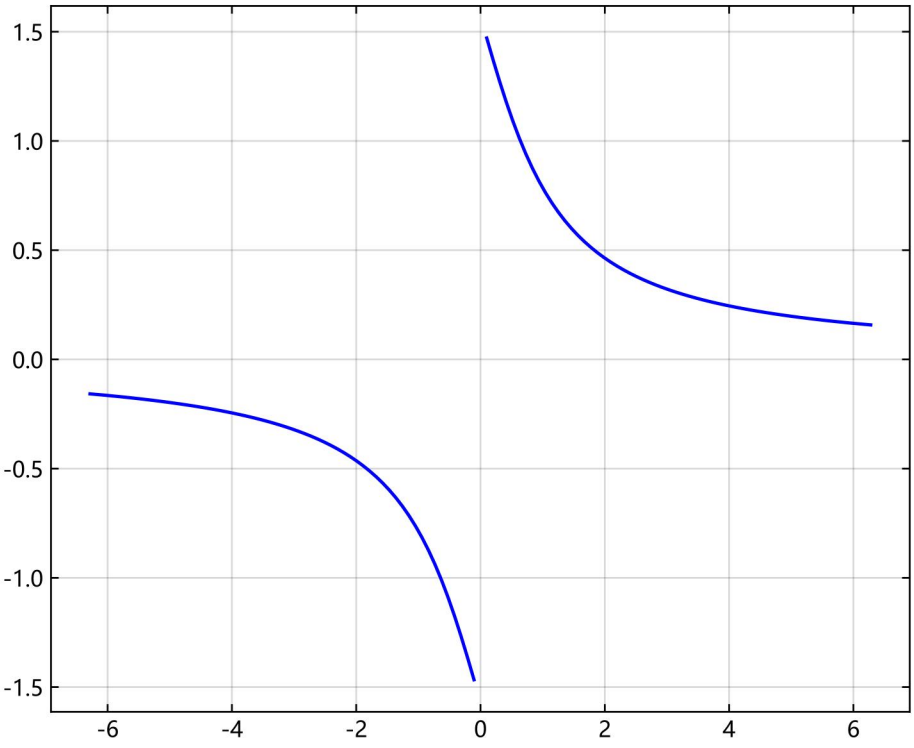
```
x = [0.5im,1+3im,-2.2+im]
y = acot.(x)
```

```
3-element Vector{ComplexF64}:
1.5707963267948966 - 0.5493061443340549im
0.10933447293697099 - 0.30594385790552897im
-0.3689075300602324 - 0.1505930389802529im
```

(3). 绘制反余切函数

在区间 $-2\pi \le x < 0$ 和 $0 < x \le 2\pi$ 上绘制反余切函数。

```
x1 = -2*pi:pi/30:-0.1
x2 = 0.1:pi/30:2*pi
plot(x1,acot.(x1),"b")
hold("on")
plot(x2,acot.(x2),"b")grid("on")
```



3.2.1.2.35 acotd

反余切（以度为单位）

A. 语法

```
Y = acotd(X)
```

B. 说明

Y = acotd(X) 返回 X 的元素的反余切 (cot)^-1（以度为单位）。该函数同时接受实数和复数输入。

对于 X 的实数值，acotd(X) 返回区间 [-90, 90] 中的值。

对于 X 的复数值，acotd(X) 返回复数值。

C. 示例

(1). 向量的反余切

```
x = [0,20,Inf]
y = acotd.(x)
```

```
3-element Vector{Float64}:
90.0
2.862405226111748
0.0
```

传递向量、矩阵或 N 维数组时，按元素执行 acotd 运算。

(2). 复数值的反余切

说明后跟余切值的反余切函数会返回 X 的原始值。

```
acotd(1+im)
```

```
ans = 31.717474411461005 - 23.053499942704924im
```

3.2.1.2.36 coth

双曲余切

A. 语法

```
Y = coth(X)
```

B. 说明

Y = coth(X) 返回 X 的元素的双曲余切。coth 函数按元素处理数组。该函数同时接受实数和复数输入。所有的角度都以弧度表示。

C. 示例

(1). 向量的双曲余切

创建一个向量并计算每个值的双曲余切。

```
X = [0,pi,2*pi,3*pi];
Y = coth.(X)
```

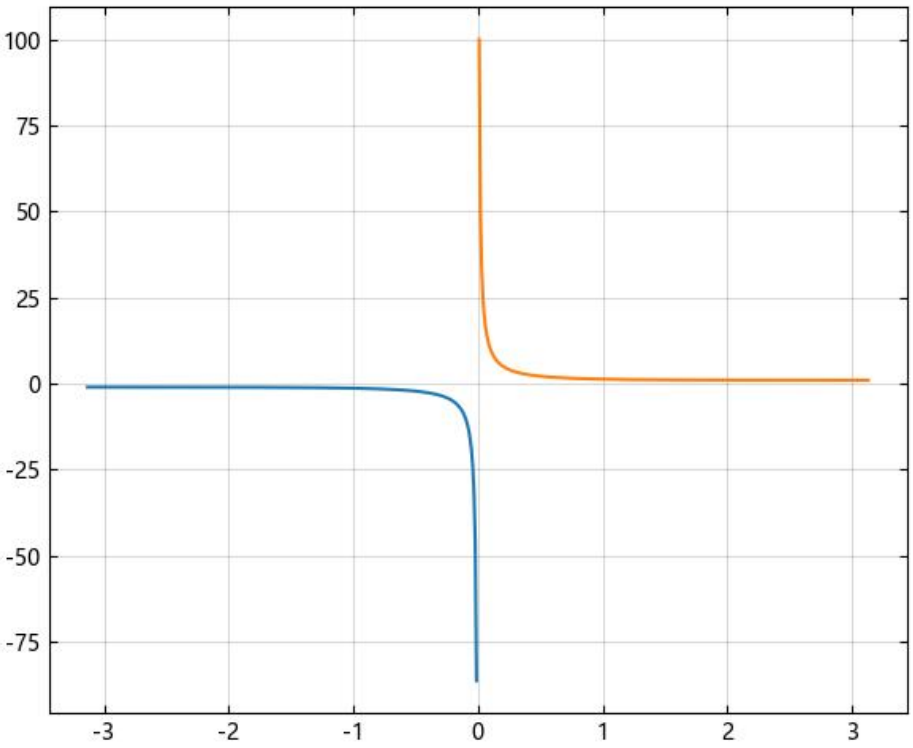
```
4-element Vector{Float64}:
Inf
1.0037418731973213
1.0000069747090357
1.0000000130248243
```

(2). 双曲余切图

在域 $-\pi < x < 0$ 和 $0 < x < \pi$ 上绘制双曲余切函数。

```
x1 = -pi+0.01:0.01:-0.01;
```

```
x2 = 0.01:0.01:pi-0.01;
y1 = coth.(x1);
y2 = coth.(x2);
plot(x1,y1,x2,y2)
grid("on")
```



3.2.1.2.37 acoth

反双曲余切

A. 语法

```
Y = acoth(X)
```

B. 说明

Y = acoth(X) 返回 X 的元素的反双曲余切。该函数同时接受实数和复数输入。所有的角度都以弧度表示。

C. 示例

(1). 向量的反双曲余切

计算向量 X 的各元素的反双曲余切。acoth 函数按元素处理 X。

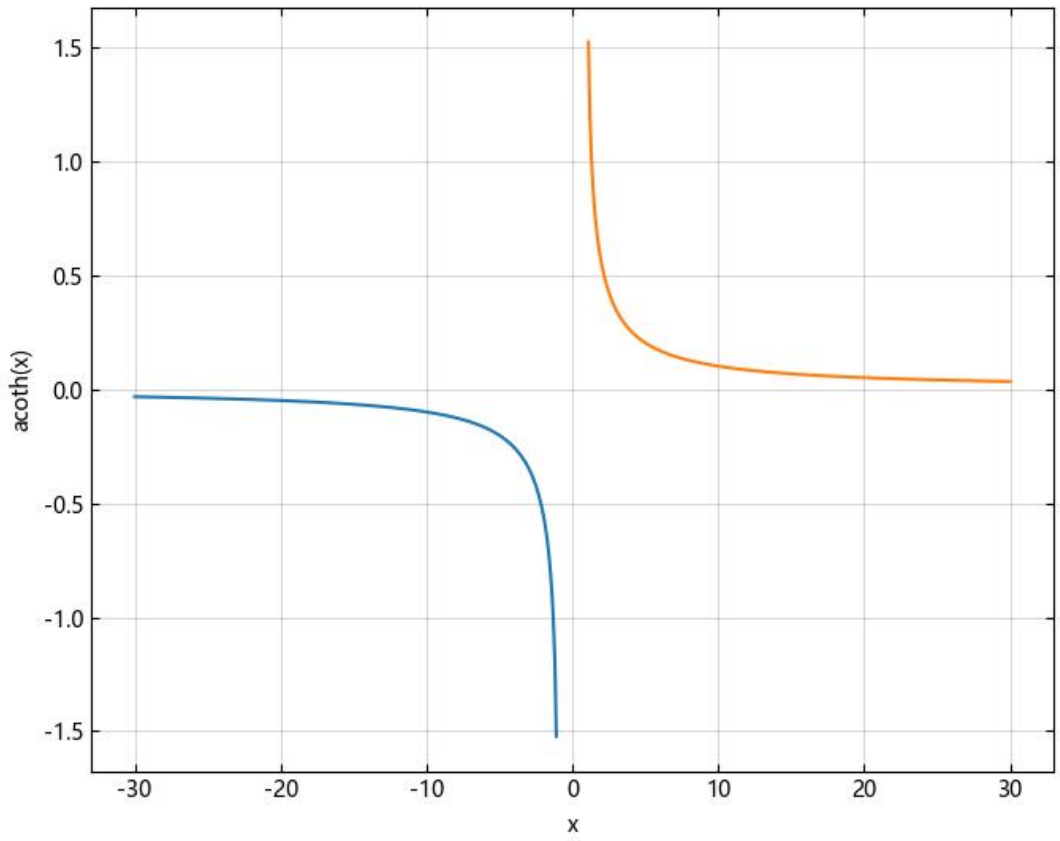
```
X = [2,-3,1+2im]
Y = acoth.(X)
```

```
3-element Vector{ComplexF64}:
0.5493061443340549 - 0.0im
-0.3465735902799726 - 0.0im
0.17328679513998632 - 0.3926990816987242im
```

(2). 绘制反双曲余切函数

在 $-30 \le x < -1$ 和 $1 < x \le 30$ 区间上绘制反双曲余切函数。

```
x1 = -30:0.1:-1.1
x2 = 1.1:0.1:30
plot(x1,acoth.(x1),x2,acoth.(x2))
grid("on")
xlabel("x")
ylabel("acoth(x)")
```



3.2.1.2.38 hypot

平方和的平方根（斜边）

A. 语法

```
C = hypot(A,B)
```

B. 说明

C = hypot(A,B) 返回以下方程的计算结果（计算是为了避免下溢或上溢）：C = sqrt(abs(A).^2 + abs(B).^2)。

C. 示例

(1). 计算斜边

计算边长为 3 和 4 的直角三角形的斜边。

```
C = hypot(3,4)
```

```
C = 5.0
```

(2). 上溢和下溢行为

检查使用 hypot 与使用 Julia 代码编写基本 hypot 方程代码之间的区别。

创建所执行的基本功能与 hypot 基本相同的匿名函数。

```
function myhypot(a,b)
sqrt(abs(a).^2+abs(b).^2)
end
myhypot(1e153,1e153)
```

```
ans = 1.414213562373095e153
```

```
myhypot(1e154,1e154)
```

```
ans = Inf
```

3.2.1.2.39 deg2rad

将角从以度为单位转换为以弧度为单位

A. 语法

```
R = deg2rad(D)
```

B. 说明

对于 D 的每个元素，R = deg2rad(D) 将角的单位从度转换为弧度。

C. 示例

(1). 以弧度为单位的直角

将 90 度角转换为以弧度为单位。

```
R = deg2rad(90)
```

```
ans = 1.5707963267948966
```

(2). 曲面距离

以度为单位指定慕尼黑和班加罗尔之间的球面距离和地球的平均半径（以千米为单位）。计算慕尼黑和班加罗尔之间的距离（沿地球曲面度量，以千米为单位）。

```
D = 64.7;
radEarth = 6371;
R = deg2rad(D);
dist = radEarth*R
```

```
dist = 7194.311753902951
```

3.2.1.2.40 rad2deg

将角的单位从弧度转换为度

A. 语法

```
D = rad2deg(R)
```

B. 说明

D = rad2deg(R) 将 R 中每个元素的角单位从弧度转换为度。

C. 示例

(1). pi 以度为单位

将 pi 转换为以度为单位。

```
D = rad2deg(pi)
```

```
ans = 180.0
```

(2). 球面距离

指定地球的平均半径和沿着地球表面测量的从慕尼黑到班加罗尔的距离（以千米为单位）。以度为单位计算慕尼黑和班加罗尔之间的球面距离。

```
dist = 7194
radEarth = 6371
R = dist/radEarth
D = rad2deg(R)
```

```
dist = 64.69719632979347
```

3.2.1.2.41 cart2pol

将笛卡尔坐标转换为极坐标或柱坐标

A. 语法

```
theta,rho = cart2pol(x,y)

theta,rho,z = cart2pol(x,y,z)
```

B. 说明

theta,rho = cart2pol(x,y) 可将二维笛卡尔坐标数组 x 和 y 的对应元素变换为极坐标 theta 和 rho。

theta,rho,z = cart2pol(x,y,z) 可将三维笛卡尔坐标数组 x、y 和 z 变换为柱坐标 theta、rho 和 z。

C. 示例

(1). 笛卡尔坐标到极坐标

将矩阵 x 和 y 中对应项定义的笛卡尔坐标转换为极坐标 theta 和 rho。

```
x = [5 3.5355 0 -10]
```

```
1×4 Matrix{Float64}:
5.0  3.5355  0.0  -10.0
```

```
y = [0 3.5355 10 0]
```

```
1×4 Matrix{Float64}:
0.0  3.5355  10.0  0.0
```

```
theta,rho = cart2pol(x,y)
```

```
theta =
```

```
1×4 Matrix{Float64}:
0.0  0.785398  1.5708  3.14159
rho =
1×4 Matrix{Float64}:
5.0  4.99995  10.0  10.0
```

(2). 笛卡尔坐标到柱坐标

将矩阵 x、y 和 z 中对应项定义的三维笛卡尔坐标转换为柱坐标 theta、rho 和 z。

```
x = [1 2.1213 0 -5]'
x =
1.0
2.1213
0.0
-5.0
y = [0 2.1213 4 0]'
y =
0.0
2.1213
4.0
0.0
z = [7 8 9 10]'
y =
7
8
9
10
theta,rho,z = cart2pol(x,y,z)
theta =
0.0
0.7853981633974483
1.5707963267948966
3.141592653589793
rho =
1.0
2.9999712298620467
4.0
5.0
z =
7.0
```

```
8.0
9.0
10.0
```

3.2.1.2.42 cart2sph

将笛卡尔坐标转换为球面坐标

A. 语法

```
theta,phi,r = cart2sph(x)

theta,phi,r = cart2sph(x,y,z)
```

B. 说明

theta,phi,r = cart2sph(x,y,z)将笛卡尔坐标数组 x、y 和 z 的对应元素转换为球面坐标 theta、phi 和 r。

C. 示例

(1). 笛卡尔坐标到球面坐标

将矩阵 x、y 和 z 中对应项定义的笛卡尔坐标转换为球面坐标 theta、phi 和 r。这些点对应于立方体的八个顶点。

```
x = [1 1 1 1; -1 -1 -1 -1]
y = [1 1 -1 -1; 1 1 -1 -1]
z = [1 -1 1 -1; 1 -1 1 -1]
theta,phi,r = cart2sph(x,y,z)
```

```
theta = 2×4 Matrix{Float64}:
0.785398  0.785398  -0.785398  -0.785398
2.35619   2.35619   -2.35619   -2.35619
phi = 2×4 Matrix{Float64}:
0.61548  -0.61548  0.61548  -0.61548
0.61548  -0.61548  0.61548  -0.61548
r = 2×4 Matrix{Float64}:
1.73205  1.73205  1.73205  1.73205
1.73205  1.73205  1.73205  1.73205
```

3.2.1.2.43 pol2cart

将极坐标或柱坐标转换为笛卡尔坐标

A. 语法

```
x,y = pol2cart(theta,rho)

 x,y,z = pol2cart(theta,rho,z)
```

B. 说明

x,y = pol2cart(theta,rho) 将极坐标数组 theta 和 rho 的对应元素变换为二维笛卡尔坐标或 xy 坐标。

x,y,z = pol2cart(theta,rho,z) 将柱坐标数组 theta、rho 和 z 的对应元素变换为三维笛卡尔坐标或 xyz 坐标。

C. 示例

(1). 极坐标转换为笛卡尔坐标

将矩阵 theta 和 rho 中对应项定义的极坐标转换为二维笛卡尔坐标 x 和 y。

```
theta = [0 pi/4 pi/2 pi]
```

```
1×4 Matrix{Float64}:
0.0  0.785398  1.5708  3.14159
```

```
rho = [5 5 10 10]
```

```
1×4 Matrix{Int64}:
5  5  10  10
```

```
x,y = pol2cart(theta,rho)
```

```
x =
1×4 Matrix{Float64}:
5.0  3.53553  6.12323e-16  -10.0
y =
1×4 Matrix{Float64}:
0.0  3.53553  10.0  1.22465e-15
```

(2). 柱坐标转换为笛卡尔坐标

将矩阵 theta、rho 和 z 中对应项定义的柱坐标转换为三维笛卡尔坐标 x、y 和 z。

```
theta = [0 pi/4 pi/2 pi]'
```

```
theta =
0.0
0.7853981633974483
1.5707963267948966
3.141592653589793
rho = [1 3 4 5]'
rho =
1
3
4
5
z = [7 8 9 10]'
z =
7
8
9
10
x,y,z = pol2cart(theta,rho,z)
x =
1.0
2.121320343559643
2.4492935982947064e-16
-5.0
y =
0.0
2.1213203435596424
4.0
6.123233995736766e-16
z =
7
8
9
10
```

3.2.1.2.44 sph2cart

将球面坐标转换为笛卡尔坐标

A. 语法

```
x,y,z = sph2cart(theta)

x,y,z = sph2cart(theta, phi, r)
```

B. 说明

x,y,z = sph2cart(theta,phi,r)将球面坐标数组 theta、phi 和 r 的对应元素转换为笛卡尔坐标，即 xyz 坐标。

C. 示例

(1). 球面坐标转换为笛卡尔坐标

将矩阵 az、el 和 r 中对应项定义的球面坐标转换为笛卡尔坐标 x、y 和 z。这些点对应于立方体的八个顶点。

```
theta = [0.7854 0.7854 -0.7854 -0.7854; 2.3562 2.3562 -2.3562 -2.3562]
phi = [0.6155 -0.6155 0.6155 -0.6155; 0.6155 -0.6155 0.6155 -0.6155]
r = 1.7321*ones(2,4)
x,y,z = sph2cart(theta,phi,r)
```

```
x = 2×4 Matrix{Float64}:
1.00001   1.00001   1.00001   1.00001
-1.00002  -1.00002  -1.00002  -1.00002
y = 2×4 Matrix{Float64}:
1.00002  1.00002  -1.00002  -1.00002
1.00001  1.00001  -1.00001  -1.00001
z = 2×4 Matrix{Float64}:
1.00006  -1.00006  1.00006  -1.00006
1.00006  -1.00006  1.00006  -1.00006
```

3.2.1.3 指数和对数

指数、对数、幂和根函数。

除了 exp 和 log 等常用函数，Syslab 还提供了其他几个相关函数，可以进行灵活的数值计算。expm1 和 log1p 函数补偿小参数中的数值舍入误差，而 reallog、realpow 和 realsqrt 函数将这些函数的范围限制为实数。nthroot 计算任意次方根，而专用函数 pow2 和 nextpow2 计算 2 的幂。

函数名	简介
exp	指数
expm1	针对较小的 x 值正确计算 exp(x)-1
exponent	标准化浮点数的指数
log	自然对数
log10	常用对数（以 10 为底）
log1p	针对较小的 x 值正确计算 log(1+x)
log2	以 2 为底的对数和浮点数分解
nextpow2	2 的更高次幂的指数
nthroot	实数的第 n 次实根
pow2	浮点数的以 2 为底的幂运算和缩放
reallog	非负实数数组的自然对数
realpow	仅实数输出的数组幂
realsqrt	非负实数数组的平方根
sqrt	平方根
cbrt	立方根

3.2.1.3.1 exp

指数

A. 语法

```
Y = exp(X)
```

B. 说明

Y = exp(X) 为数组 X 中的每个元素返回指数 e^x。对于复数元素 z = x + iy，它返回以下复指数

e^z=e^x(cos(y)+im*sin(y))

同样可以使用 exp 计算矩阵指数。

C. 示例

(1). e 的数字表示形式

计算 1 的指数，它是欧拉数 e。

```
exp(1)
```

```
ans = 2.718281828459045
```

(2). 欧拉恒等式

欧拉恒等式即等式 e^(i π)+1=0。

计算 e^(i*pi)的值

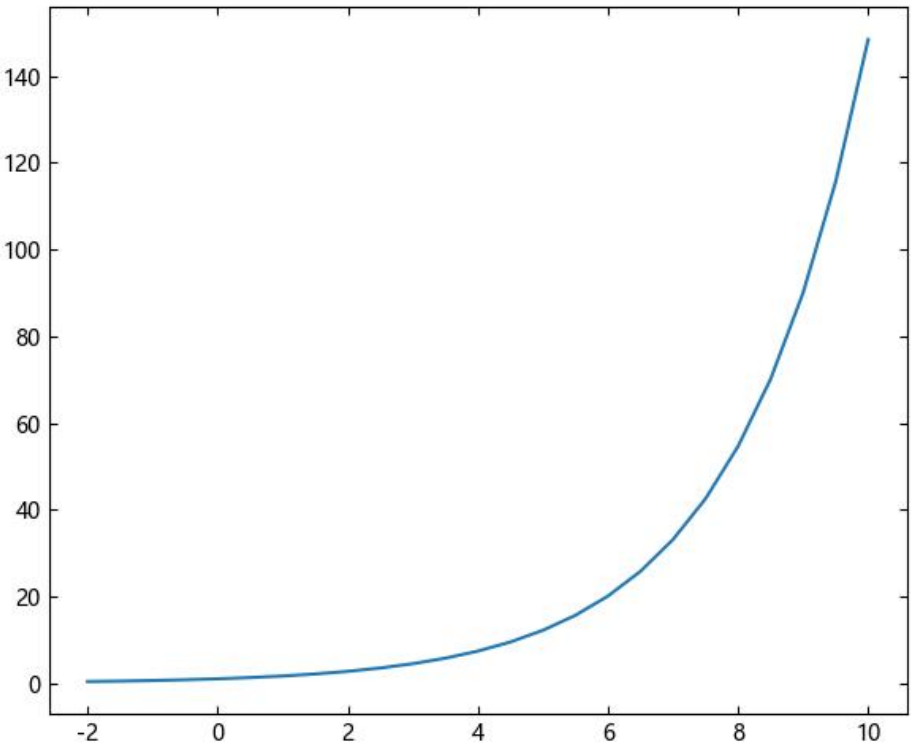
```
Y = exp(1im*pi)
```

```
ans = -1.0 + 1.2246467991473532e-16im
```

(3). 为指数函数绘图

为范围[-2,10]内的 x 值绘制 y=e^(x/2)

```
X = -2:0.5:10
Y = exp.(X./2)
plot(X,Y)
```



3.2.1.3.2 expm1

针对较小的 x 值正确计算 exp(x)-1

A. 语法

```
y = expm1(x)
```

B. 说明

y = expm1(x) 计算 exp(x)-1，并补偿 exp(x) 中的舍入。
对于较小的 x，expm1(x) 约为 x，而 exp(x)-1 可能为零。

3.2.1.3.3 exponent

x 的二进制指数

A. 语法

```
y = exponent(x)
```

B. 说明

获取标准化浮点数的指数。 返回最大的整数 y，使得 2^y ≤ abs(x)。

C. 示例

(1). 标量的指数求 5.6 的标准化浮点数指数

```
exponent(5.6)
```

```
ans = 2
```

3.2.1.3.4 log

自然对数

A. 语法

```
Y = log(X)
```

B. 说明

Y = log(X) 返回数组 X 中每个元素的自然对数 ln(x)。

log 函数的域包含负数和复数，如果使用不当，可能会导致意外结果。对于负数和复数 z = u + i*w，复数对数 log(z) 返回

log(abs(z)) + 1i*angle(z)

如果想要负数和复数返回错误消息而不是返回复数结果，改用 reallog。

同样可以使用 log 计算矩阵指数。

C. 示例

(1). 负数的自然对数

显示 -1 的自然对数是 iπ。

```
log(Complex(-1.0))
```

```
ans = 0.0 + 3.141592653589793im
```

3.2.1.3.5 log10

自然对数

A. 语法

```
Y = log10(X)
```

B. 说明

Y = log10(X) 返回数组 X 中每个元素的常用对数。该函数同时接受实数和复数输入。对于 X 在区间 (0, Inf) 内的实数值，log10 返回区间 (-Inf ,Inf) 内的实数值。对于 X 的复数值和负实数值，log10 函数返回复数值。

同样可以使用 log10 计算矩阵指数。

C. 示例

(1). 计算标量的常用对数值

检查以 10 为底的对数函数的几个值。

计算 1 的常用对数。

```
log10(1)
```

```
ans = 0.0
```

结果是 0，因此这是 log10 函数在 x 轴上的截距。

计算 10 的常用对数。

```
log10(10)
```

```
ans = 1.0
```

结果为 1，因为 10^1=10

计算 100 的常用对数

```
log10(100)
```

```
ans = 2.0
```

结果为 2，因为 10^2=100

计算 0 的常用对数。

```
log10(0)
```

```
-Inf
```

结果为 -Inf，因为 10^(-∞)=0

(2). 实数值的常用对数

创建一个由区间 [0.5 5] 中的数字组成的向量。

```
X = (0.5:0.5:5)
```

计算 X 的常用对数。

```
Y = log10.(X)
```

```
10-element Vector{Float64}:
-0.3010299956639812
0.0
0.17609125905568124
0.3010299956639812
0.3979400086720376
0.47712125471966244
0.5440680443502757
0.6020599913279624
0.6532125137753437
0.6989700043360189
```

(3). 复数值的常用对数

为 X 和 Y 创建两个笛卡尔网格。

```
X,Y = meshgrid2(0:0.5:1.5,-2:0.5:2)
```

基于网格计算以 10 为底的对数复数 log10(X+iY)。利用 1i 来改进复数算术运算的速度和可靠性。

```
Z = log10.(X .+ 1im .* Y)
```

```
9×4 Matrix{ComplexF64}:
0.30103-0.682188im       0.314194-0.575795im       0.349485-0.480829im
```

```
0.39794-0.402719im
0.176091-0.682188im          0.19897-0.542454im     0.255942-0.426822im
0.326606-0.341094im
0.0-0.682188im    0.048455-0.480829im   0.150515-0.341094im   0.255942-
0.255366im
-0.30103-0.682188im       -0.150515-0.341094im       0.048455-0.20136im
0.19897-0.139734im
-Inf+0.0im                    -0.30103+0.0im                  0.0+0.0im
0.176091+0.0im
-0.30103+0.682188im       -0.150515+0.341094im       0.048455+0.20136im
0.19897+0.139734im
0.0+0.682188im           0.048455+0.480829im        0.150515+0.341094im
0.255942+0.255366im
0.176091+0.682188im          0.19897+0.542454im     0.255942+0.426822im
0.326606+0.341094im
0.30103+0.682188im         0.314194+0.575795im      0.349485+0.480829im
0.39794+0.402719im
```

3.2.1.3.6 log1p

针对较小的 x 值正确计算 log(1+x)

A. 语法

```
y = log1p(x)
```

B. 说明

y = log1p(x) 计算 log(1+x)，并补偿 1+x 中的舍入。

对于较小的 x 值，log1p(x) 比 log(1+x) 更准确。对于较小的 x，log1p(x) 约为 x，而 log(1+x) 可能为零。

3.2.1.3.7 log2

以 2 为底的对数和浮点数分解

A. 语法

```
Y = log2(X)
```

```
F,E=log2(X)
```

B. 说明

Y = log2(X) 计算 X 的元素以 2 为底的对数，满足 2^Y=X。

C. 示例

(1). 以 2 为底的对数值

```
X = [0 1 2 10 Inf NaN]
Y = log2.(X)
```

```
1×6 Matrix{Float64}:
-Inf  0.0  1.0  3.32193  Inf  NaN
```

3.2.1.3.8 nextpow2

2 的更高次幂的指数

A. 语法

```
P=nextpow2(A)
```

B. 说明

P = nextpow2(A) 返回对 A 中每个元素满足

2^P>=|A|

的最小的 2 的幂的指数。按照惯例，nextpow2(0) 返回零

您可以使用 nextpow2 填充传递到 fft 的信号。当信号长度并非 2 次幂时，这样做可以加快 FFT 的运算速度。

C. 示例

(1). 下一个双精度整数值的 2 次幂

定义一个 Float64 整数值向量，并计算下一个 2 次幂大于这些值的指数。

```
a = [1, -2, 3 ,-4, 5, 9, 519]
p = nextpow2.(a)
```

```
7-element Vector{Int64}:
0
1
2
```

```
2
3
4
10
```

计算正的 2 的更高次幂。

```
np2 = 2 .^p
```

```
7-element Vector{Int64}:
1
2
4
4
8
16
1024
```

保留原始输入值的符号。

```
np2 .*sign.(a)
```

```
7-element Vector{Int64}:
1
-2
4
-4
8
16
1024
```

(2). 无符号整数值的 2 更高次幂

定义一个无符号整数向量，并计算 2 的更高次幂大于这些值的指数。

```
a = UInt32.([1020,4000 ,32700])
p = nextpow2.(a)
```

```
3-element Vector{Int64}:
10
12
15
```

计算下一个大于 a 中值的 2 次幂。

```
2 .^p
```

```
3-element Vector{Int64}:
1024
4096
32768
```

(3). 带填充的优化 FFT

当信号长度不等于 2 的幂时，使用 nextpow2 函数可提升 fft 的性能。

创建包含 8191 个样本值的一维向量。

```
x = rand(1,8191)
```

计算 2 的更高次幂中大于 8191 的值。

```
p = nextpow2(8191)
n = 2^p
```

```
n = 8192
```

```
x=vcat(vec(x),zeros(n-length(x)))
y = fft(x);
```

3.2.1.3.9 nthroot

实数的第 n 次实根

A. 语法

```
Y=nthroot(X,N)
```

B. 说明

Y = nthroot(X,N) 返回 X 元素的第 n 次实根。X 和 N 必须为大小相同的实数标量或数组。如果 X 中的元素为负数，则 N 必须为奇整数。

C. 示例

(1). 计算负数的实根

求 -27 的实数立方根。

```
nthroot(-27, 3)
```

```
ans = -3.0
```

要进行比较，还计算 (-27)^(1/3)。

```
(Complex(-27.0))^(1/3)
```

```
ans = 1.5 + 2.598076211353316im
```

结果为 -27 的复数立方根。

(2). 计算标量的多个实根

创建一个要计算根的向量 N。

```
N = [5,3,-1]
```

使用 nthroot 计算 -8 的多个实根。

```
Y = nthroot.(-8,N)
```

```
3-element Vector{Float64}:
-1.5157165665103982
-2.0
-0.125
```

结果为一个大小与 N 相同的向量。

(3). 矩阵的按元素运算的根

创建一个由基数组成的矩阵 X 和一个由第 n 个根组成的矩阵 N。

```
X = [-2 -2 -2; 4 -3 -5]
```

```
2×3 Matrix{Int64}:
-2  -2  -2
4  -3  -5
```

```
N = [1 -1 3; 1/2 5 3]
```

```
2×3 Matrix{Float64}:
1.0  -1.0  3.0
0.5   5.0  3.0
```

X 中的每个元素对应于 N 中的一个元素。

计算 X 中元素的第 n 个实根。

```
Y = nthroot.(X,N)
```

```
2×3 Matrix{Float64}:
-2.0  -0.5      -1.25992
16.0  -1.24573  -1.70998
```

除符号之外（单独处理），该结果相当于 abs(X).^(1./N)。相比之下，您可以使用 X.^(1./N) 计算复数根。

3.2.1.3.10 pow2

求以 2 为底的幂值并对浮点数字进行缩放

A. 语法

```
X = pow2(Y)
X = pow2(F,E)
```

B. 说明

X = pow2(Y) 返回数组 X，其元素为 2 的 Y 次幂。

X = pow2(F,E) 针对 F 和 E 中的对应元素计算 x = f * 2^e。只需将 E 添加到 F 的浮点指数中即可快速计算结果。参数 F 和 E 分别为实数和整数数组。

C. 示例

(1). 幂

计算 2 的 E 次方。

```
E = [1 -2 4 -4 3 9];
Y = pow2(E)
```

```
E = 1×6 Matrix{Float64}:
2.0  0.25  16.0  0.0625  8.0  512.0
```

(2). 有效数缩放 2 的指数次方

语句 X = pow2(F,E) 生成以下值：

```
F E X
1/2 1 1
pi/4 2 pi
-3/4 2 -3
1/2 -51 eps()
1-eps()/2 1024 Inf
1/2 -1021 2.2250738585072014e-308
```

3.2.1.3.11 reallog

非负实数数组的自然对数

A. 语法

```
P=reallog(A)
```

B. 说明

Y = reallog(X) 返回数组 X 中每个元素的自然对数。数组 X 只能包含非负实数。Y 的大小与 X 的大小相同。

C. 示例

(1). 矩阵的自然对数

创建一个 4×4 矩阵并计算每个元素的自然对数。

```
X = magic(4)
```

```
4×4 Matrix{Int64}:
16   2   3  13
5  11  10   8
9   7   6  12
4  14  15   1
```

计算正的 2 的更高次幂。

```
Y = reallog.(X)
```

```
4×4 Matrix{Float64}:
2.77259  0.693147  1.09861  2.56495
1.60944  2.3979    2.30259  2.07944
2.19722  1.94591   1.79176  2.48491
1.38629  2.63906   2.70805  0.0
```

3.2.1.3.12 realpow

仅实数输出的数组幂

A. 语法

```
Z=realpow(X,Y)
```

B. 说明

Z = realpow(X,Y) 将数组 X 的每个元素提升到数组 Y 中的对应元素的幂。数组 X 和 Y 的大小必须相同。realpow 的范围是所有实数集合，即输出数组 Z 的所有元素必须为实数。

C. 示例

(1). 矩阵元素的幂

创建一个 3×3 矩阵，其中每个元素的值都相同。

```
X = -2*ones(3,3)
```

```
3×3 Matrix{Float64}:
-2.0  -2.0  -2.0
-2.0  -2.0  -2.0
-2.0  -2.0  -2.0
```

计算由整数值组成的 3×3 矩阵。

```
Y = pascal(3)
```

```
3×3 Matrix{Int64}:
1  1  1
1  2  3
1  3  6
```

使用 Y 中定义的对应指数对 X 中的元素按元素求幂。

```
Z = realpow.(X,Y)
```

```
3×3 Matrix{Float64}:
-2.0  -2.0  -2.0
-2.0   4.0  -8.0
-2.0  -8.0  64.0
```

3.2.1.3.13 realsqrt

非负实数数组的平方根

A. 语法

```
Z=realsqrt(X)
```

B. 说明

Z = realsqrt(X) 返回数组 X 的每个元素的平方根。Y 的大小与 X 的大小相同。

如果您希望负的复数返回复数结果而不是错误消息，请改用 sqrt。

C. 示例

(1). 矩阵元素的平方根

创建一个由正实数值组成的 4×4 矩阵。

```
X = magic(4)
```

```
4×4 Matrix{Int64}:
16   2   3  13
5  11  10   8
9   7   6  12
4  14  15   1
```

计算 X 中每个元素的平方根。

```
Y = realsqrt.(X)
```

```
4×4 Matrix{Float64}:
4.0      1.41421  1.73205  3.60555
```

```
2.23607  3.31662  3.16228  2.82843
3.0      2.64575  2.44949  3.4641
2.0      3.74166  3.87298  1.0
```

3.2.1.3.14 sqrt

平方根

A. 语法

```
Y = sqrt(X)
```

B. 说明

B = sqrt(X) 返回数组 X 的每个元素的平方根。对于 X 的负元素或复数元素，sqrt(X) 生成复数结果。

sqrt 函数的域包含负数和复数，如果使用不当，可能会导致意外结果。对于负的复数 z = u + i*w，复数方根 sqrt(z) 返回

sqrt(r)*(cos(phi/2) + 1i*sin(phi/2))

其中 r = abs(z) 是半径，phi = angle(z) 是在闭区间 -pi <= phi <= pi 内的相位角。

C. 示例

(1). 向量元素的平方根

创建一个既包含负值也包含正值的行向量。

```
X = -2:2
```

```
ans = -2:2
```

计算 X 的每个元素的平方根。

```
Y=sqrt.(Complex.(X))
```

```
5-element Vector{ComplexF64}:
0.0 + 1.4142135623730951im
0.0 + 1.0im
0.0 + 0.0im
1.0 + 0.0im
1.4142135623730951 + 0.0im
```

3.2.1.3.15 cbrt

立方根

A. 语法

```
Y = cbrt(X)
```

B. 说明

B = cbrt(X) 返回数组 X 的每个元素的立方根。对于 X 的负元素或复数元素，cbrt(X) 生成复数结果。

cbrt 函数的域包含负数和复数，如果使用不当，可能会导致意外结果。对于负的复数 z = u + i*w，复数方根 cbrt(z) 返回

cbrt(r)*(cos(phi/2) + 1i*sin(phi/2))

其中 r = abs(z) 是半径，phi = angle(z) 是在闭区间 -pi <= phi <= pi 内的相位角。

如果想要负数和复数返回错误消息而不是返回复数结果，改用 realcbrt。

C. 示例

(1). 向量元素的立方根

创建一个既包含负值也包含正值的行向量。

```
X = -2:2
```

```
ans = -2:2
```

计算 X 的每个元素的立方根。

```
Y=cbrt.(X)
```

```
5-element Vector{Float64}:
-1.2599210498948732
-1.0
0.0
1.0
1.2599210498948732
```

3.2.1.4 复数

实部和虚部、相位角度。

在 Syslab 中，im 表示基本虚数单位。您可以使用它们来创建复数，例如

2im+5。您还可以确定复数的实部和虚部，并计算相位和角度等其他常用值。

函数名	简介
abs	绝对值和复数的模
abs2	绝对值和复数的模的平方
angle	相位角
complex	创建复数数组
conj	复共轭
im	虚数单位
imag	复数的虚部
isreal	确定数组是否使用复数存储
real	复数的实部
sign	Sign 函数（符号函数）
unwrap	平移相位角

3.2.1.4.1 abs

绝对值和复数的模

A. 语法

```
Y = abs(X)
```

B. 说明

Y = abs(X) 返回数组 X 中每个元素的绝对值。

如果 X 是复数，则 abs(X) 返回复数的模数。

C. 示例

(1). 标量的绝对值

```
y = abs(-5)
```

```
ans = 5
```

(2). 向量的绝对值

创建实值的数值向量。

```
x = [1.3 ,-3.56, 8.23, -5 ,-0.01]
```

```
5-element Vector{Float64}:
1.3
-3.56
8.23
-5.0
-0.01
```

计算向量元素的绝对值。

```
y = abs.(x)
```

```
5-element Vector{Float64}:
1.3
3.56
8.23
5.0
0.01
```

(3). 复数的模

```
y=abs(3+4im)
```

```
y = 5.0
```

3.2.1.4.2 abs2

绝对值和复数的模的平方

A. 语法

```
Y = abs2(X)
```

B. 说明

Y = abs2(X) 返回数组 X 中每个元素的绝对值的平方。

如果 X 是复数，则 abs2(X) 返回复数的模数的平方。

C. 示例

(1). 标量的绝对值平方

```
y = abs2(-5)
```

```
ans = 25
```

(2). 向量的绝对值平方

创建实值的数值向量。

```
x = [1.3 ,-3.56, 8.23, -5 ,-0.01]
```

```
5-element Vector{Float64}:
1.3
-3.56
8.23
-5.0
-0.01
```

计算向量元素的绝对值。

```
y = abs2.(x)
```

```
5-element Vector{Float64}:
1.6900000000000002
12.6736
67.7329
25.0
0.0001
```

(3). 复数的模平方

```
y=abs2(3+4im)
```

```
y = 25
```

3.2.1.4.3 angle

相位角

A. 语法

```
theta = angle(z)
```

B. 说明

theta = angle(z) 为复数数组 z 的每个元素返回区间 [-π,π]中的相位角。theta 中的角度表示为 z=abs(z).*exp(i*theta)。

C. 示例

(1). 复数的幅值和相位

创建一个复数，并计算其幅值和相位。

```
z = 2*exp(im*0.5)
```

```
ans = 1.7551651237807455 + 0.958851077208406im
```

```
r = abs(z)
```

```
ans = 2.0
```

```
theta = angle(z)
```

```
theta = 0.5
```

(2). FFT 相位

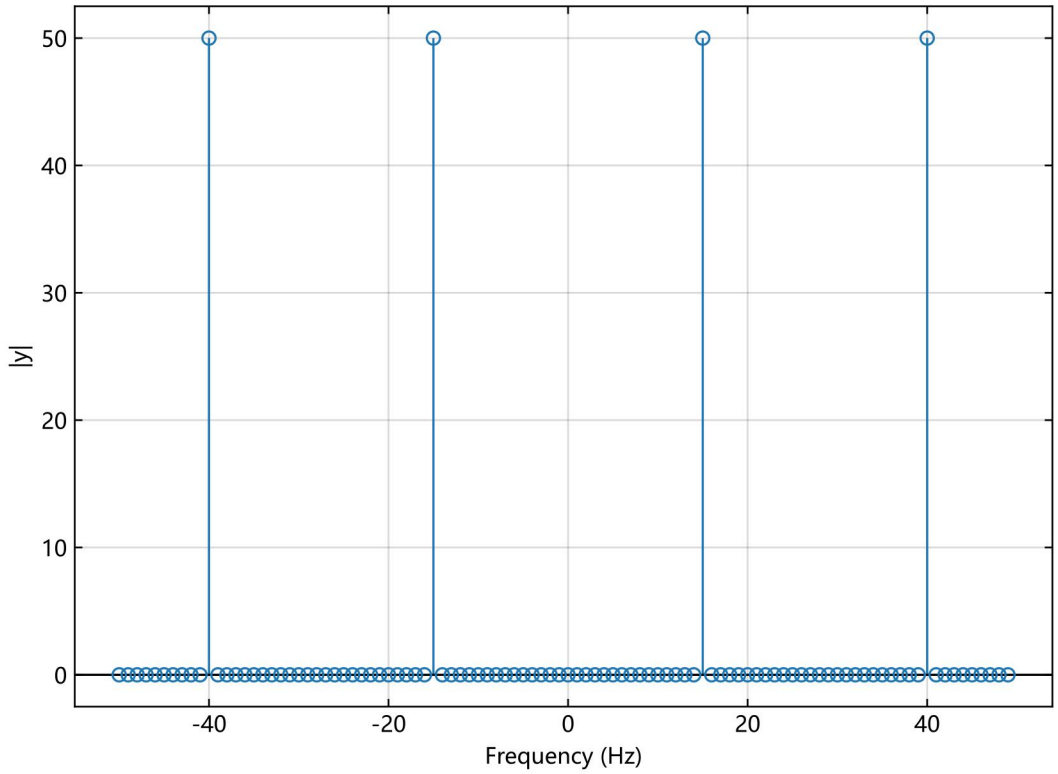
创建一个由频率为 15 Hz 和 40 Hz 的两个正弦波组成的信号。第一个正弦波的相位为 −π/4，第二个正弦波的相位为 π/2。以 100 Hz 的频率对信号进行一秒钟的采样。

```
fs = 100
t = 0:1/fs:1-1/fs
x = cos.(2 .*pi .*15 .*t .- pi/4) .- sin.(2 .*pi .*40 .*t)
```

计算信号的傅里叶变换。将变换幅值绘制为频率函数。

```
y = fft(x)
z = fftshift(y)
ly = length(y)
f = (-ly/2:ly/2-1)/ly*fs

stem([f...],abs.(z))
xlabel("Frequency (Hz)")
ylabel("|y|")
grid("on")
```

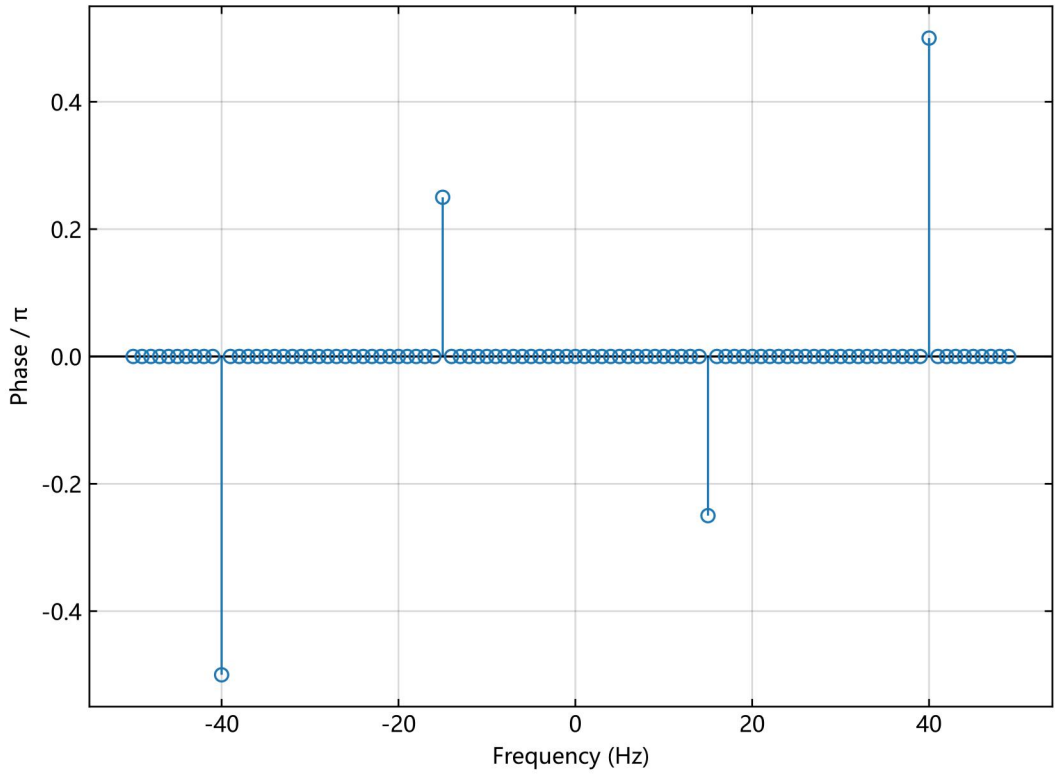


计算变换的相位，删除小幅值变换值。将相位绘制为频率函数。

```
tol = 1e-6
z[abs.(z) .< tol] .= 0

theta = angle.(z)

stem(f,theta/pi)
xlabel("Frequency (Hz)")
ylabel("Phase / π")
grid("on")
```



(3). 复数的模

```
y=angle(3+4im)
```

```
y = 0.9272952180016122
```

3.2.1.4.4 complex

创建复数数组

A. 语法

```
z = complex(a,b)
z = complex(x)
```

B. 说明

z = complex(a,b) 通过两个实数输入创建一个复数输出 z，这样 z = a + bi。

complex 函数非常有用，在一定情况下可替代如下的表达式：a + 1im*b。

a 和 b 不是 Float64 或 Float32。

b 全部为零。

z = complex(x) 返回 x 的等效复数，这样 isreal(z) 返回逻辑值 0 (false)。

如果 x 是实数，则 z 是 x + 0i。

如果 x 是复数，则 z 与 x 相同。

C. 示例

(1). 来自两个实数标量的复数标量

使用 complex 函数创建复数标量 3 + 4i。

```
z = complex(3,4)
```

```
z = 3 + 4im
```

(2). 基于两个实数向量的复数向量

基于两个实数 UInt8 向量创建复数 UInt8 向量。z 的大小（4×1）与输入参数的大小相同。

```
a = UInt8.([1;2;3;4])
b = UInt8.([2;2;7;7])

z = complex.(a,b)
```

```
4-element Vector{Complex{UInt8}}:
0x01 + 0x02im
0x02 + 0x02im
0x03 + 0x07im
0x04 + 0x07im
```

(3). 来自一个实数标量的复数标量

创建一个不包含虚部的复数标量。

```
z = complex(12)
```

```
z = 12 + 0im
```

验证 z 是否为复数。

```
typeof(z)
```

```
ans = Complex{Int64}
```

3.2.1.4.5 conj

复共轭

A. 语法

```
Zc = conj(Z)
```

B. 说明

Zc = conj(Z) 返回 Z 中每个元素的复共轭。

C. 示例

(1). 求复数的复共轭

求复数 Z 的复共轭。

```
Z = 2+3im
```

```
Z = 2 + 3im
```

```
Zc = conj(Z)
```

```
Zc = 2 - 3im
```

(2). 求矩阵中复数值的复共轭

创建一个包含复数元素的 2×2 的矩阵。

```
Z = [0-1im 2+1im; 4+2im 0-2im]
```

```
2×2 Matrix{Complex{Int64}}:
0-1im  2+1im
4+2im  0-2im
```

求矩阵 Z 中每个复数的复共轭。

```
Zc = conj.(Z)
```

```
2×2 Matrix{Complex{Int64}}:
0+1im  2-1im
4-2im  0+2im
```

3.2.1.4.6 im

虚数单位

A. 语法

```
1im
z = a+bim
z= x + 1im*y
```

B. 说明

1i 返回基本虚数单位。i 等效于 sqrt(-1)。

可使用 i 输入复数。还可将 j 字符用作虚数单位。要在不使用 i 和 j 的

情况下创建复数，可使用 complex 函数。

z = a + bi 返回一个复数常量 z。

z = x + 1i*y 返回一个复数数组 z。

C. 示例

(1). 复数标量

创建一个复数标量，在构建复数常量时使用字符 im（不带乘号）作为后缀。

```
z = 1+2im
```

```
z =1 + 2im
```

(2). 复数向量

根据两个 4×1 实数向量创建一个复数向量。z 是一个 4×1 复数向量。

```
x = [1:4]'
y = [8:-2:2]'

z = x+1im .*y
```

```
1×1 Matrix{Adjoint{Complex{Int64}, Vector{Complex{Int64}}}}:
[1 + 8im 2 + 6im 3 + 4im 4 + 2im]
```

(3). 复指数

创建一个复数标量，表示半径为 r、角度为 theta（从原点开始）的复数向量。

```
r = 4
theta = pi/4

z = r*exp(1im*theta)
```

```
z = 2.8284271247461903 + 2.828427124746191im
```

3.2.1.4.7 imag

复数的虚部

A. 语法

```
Y = imag(Z)
```

B. 说明

Y = imag(Z) 返回数组 Z 中每个元素的虚部。

C. 示例

(1). 复数的虚部

求复数 Z 的虚部。

```
Z = 2+3im
Y = imag(Z)
```

```
Y = 3
```

(2). 复数值向量的虚部

求向量 Z 中每个元素的虚部。imag 函数按元素处理 Z。

```
Z = [0.5im ,1+3im ,-2.2]
Y = imag.(Z)
```

```
3-element Vector{Float64}:
0.5
3.0
0.0
```

3.2.1.4.8 isreal

确定数组是否使用复数存储

A. 语法

```
tf = isreal(A)
```

B. 说明

当数值数组 A 没有虚部时，tf = isreal(A) 返回逻辑值 1 (true)，否则返回逻辑值 0 (false)。对于虚部为零的复数值，isreal 返回逻辑值 0 (false)，因为该值仍然存储为复数。

C. 示例

(1). 确定矩阵是否包含所有实数值

定义一个 3×4 矩阵 A。

```
A = [7 3+4im 2 5im;
  2im 1+3im 12 345;
```

```
  52 108 78 3];
```

确定数组是否为实数组。

```
tf = isreal(A)
```

```
tf = false
```

因为 A 包含复数元素，所以 isreal 返回 false。

(2). 定义具有零值虚部的复数

使用 complex 函数创建一个具有零值虚部的标量 A。

```
A = complex(12)
```

```
12 + 0im
```

确定 A 是否为实数。

```
tf = isreal(A)
```

```
tf = true
```

虚部为 0 时，返回 true

(3). 生成零值虚部的计算

定义两个复数标量 x 和 y。

```
x=3+4im
y=5-4im
```

确定两个复数标量 x 和 y 的和是否为实数。

```
A = x+y
```

```
A = 8 + 0im
```

```
isreal(A)
```

```
ans = true
```

3.2.1.4.9 real

复数的实部

A. 语法

```
X = real(Z)
```

B. 说明

X = real(Z) 返回复数 Z 的实部。

C. 示例

(1). 复数的实部求复数 Z 的实部。

```
Z = 2+3im
X = real(Z)
```

```
X = 2
```

(2). 复数值向量的实部

求向量 Z 中每个元素的实部。real 函数按元素处理 Z。

```
Z = [0.5im,1+3im,-2.2]
X = real.(Z)
```

```
3-element Vector{Float64}:
0.0
1.0
-2.2
```

3.2.1.4.10 sign

Sign 函数（符号函数）

A. 语法

```
sign(x)
```

B. 说明

Y = sign(x) 返回与 x 大小相同的数组 Y，其中 Y 的每个元素是：

1，前提是 x 的对应元素大于 0。

0，前提是 x 的对应元素等于 0。

-1，前提是 x 的对应元素小于 0。

x./abs(x)，前提是 x 为复数。

C. 示例

(1). 求符号函数

求数字的符号函数。

```
sign(2)
```

```
ans = 1
```

求向量值的符号函数。

```
V = [-11 ,0 ,1.5, Inf, NaN]
sign.(V)
```

```
5-element Vector{Float64}:
-1.0
0.0
1.0
1.0
NaN
```

求矩阵值的符号函数。

```
M = magic(3) .- 5
sign.(M)
```

```
3×3 Matrix{Int64}:
1  -1   1
-1   0   1
-1   1  -1
```

求复数的符号函数。

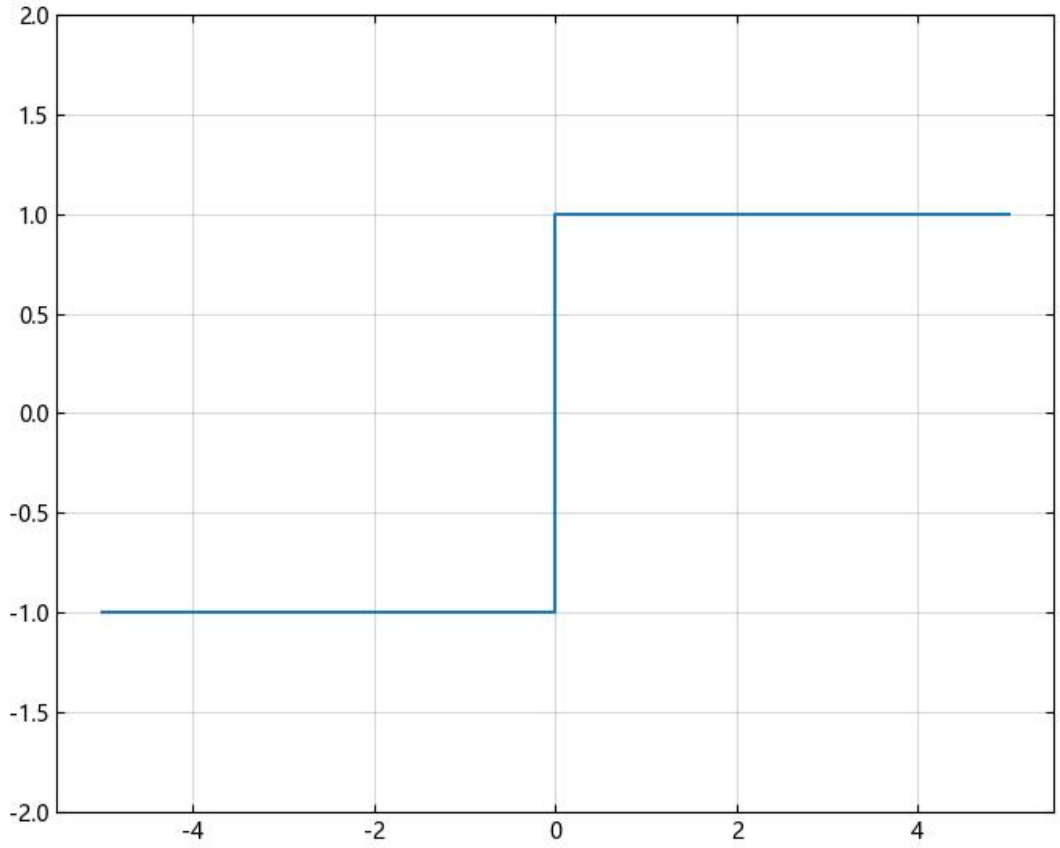
```
z = 4 - 3*im
sign(z)
```

```
ans = 0.8 - 0.6im
```

(2). 绘制符号函数

绘制符号函数，并显示其在过零点处的行为。使用 eps 表示略高于或低于 0 的值。

```
x = [-5,-eps(1.0), 0 ,eps(1.0) ,5]
y = sign.(x)
plot(x,y)
ylim([-2 2])
grid("on")
```



(3). 绘制符号函数的实部和虚部

在 −3<x<−3 和−3<y<−3 范围内绘制符号函数的实部和虚部。

首先，使用 meshgrid 创建 -3 < x < 3 和 -3 < y < 3 范围内的值的网格。然后使用 z = x + 1i*y 根据这些值创建复数。

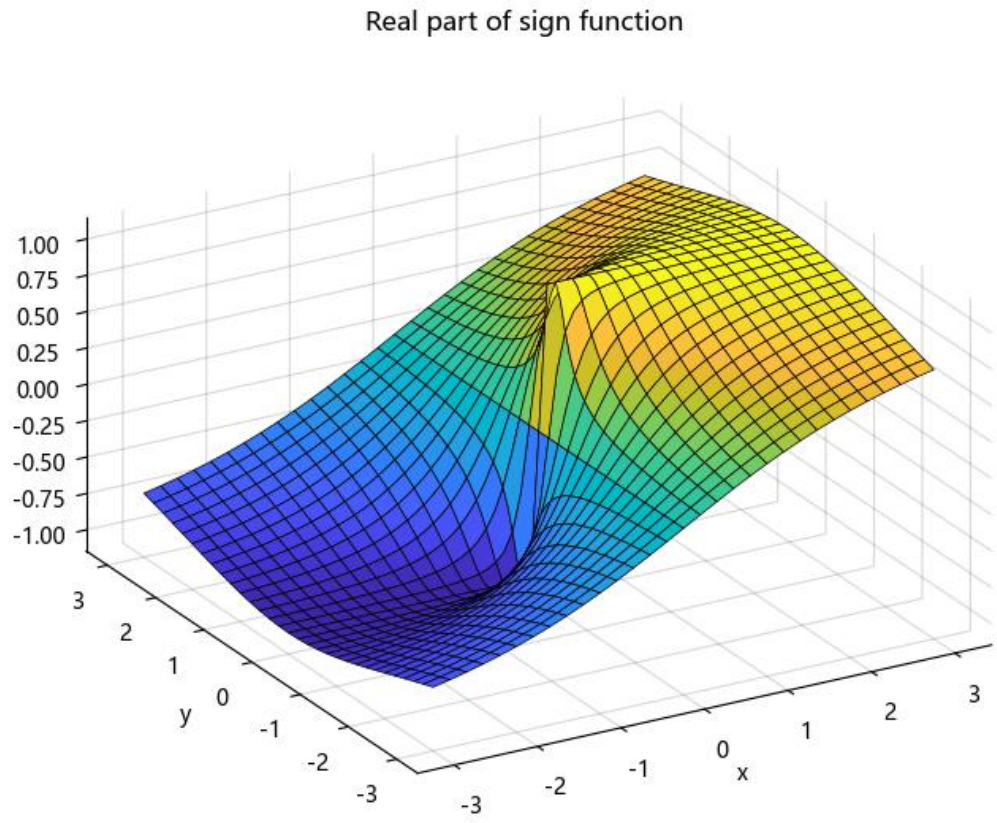
```
v = -3:0.1:3
x, y = meshgrid2(v)
z = x .+ 1im .*y
```

求出 z 的符号函数的实部和虚部。

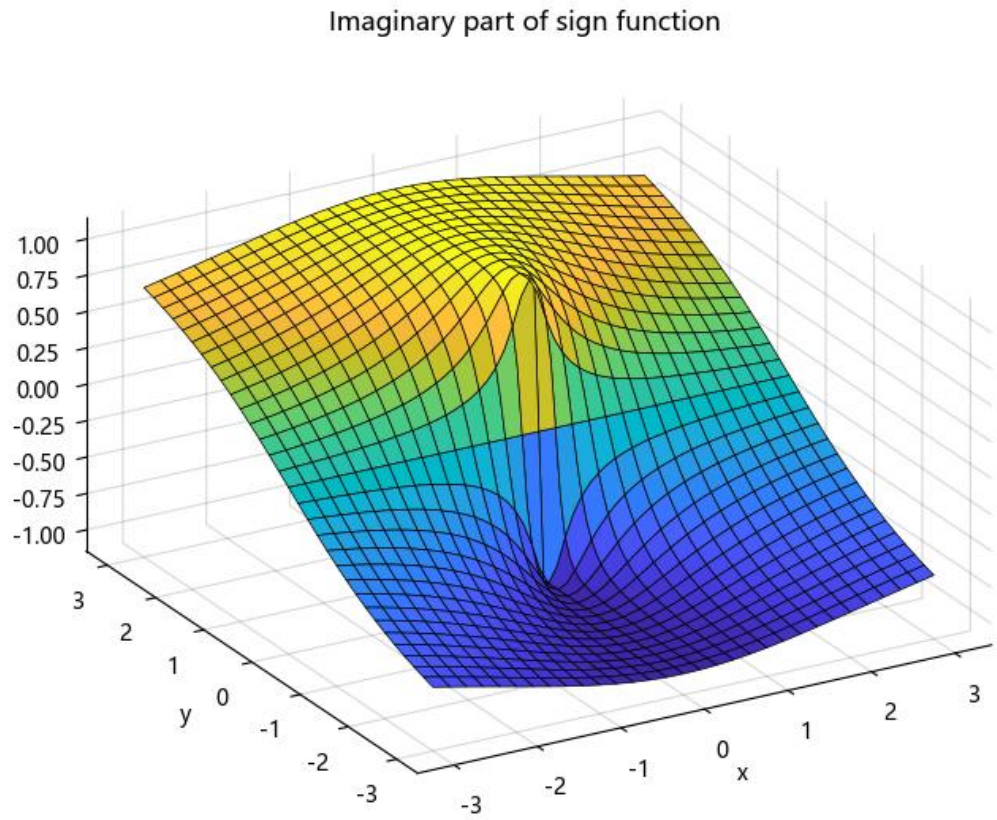
```
s = sign.(z)
re = real.(s)
i = imag.(s)
```

绘制实部和虚部。

```
surf(x,y,re)
title("Real part of sign function")
xlabel("x")
ylabel("y")
```




```
figure(2)
surf(x,y,i)
title("Imaginary part of sign function")
xlabel("x")
ylabel("y")
```



3.2.1.4.11 unwrap

平移相位角

A. 语法

```
Q = unwrap(P)
Q = unwrap(P,tol)
Q = unwrap(P,tol,dim)
```

B. 说明

Q = unwrap(P)展开向量 P 中的弧度相位角。每当连续相位角之间的跳跃大于或等于 π 弧度时，unwrap 就会通过增加 ±2π 的整数倍来平移相位角，直到跳跃小于 π。如果 P 是矩阵，unwrap 将按列运算。如果 P 是多维数组，unwrap 将对大小大于 1 的第一个维度进行运算。

Q = unwrap(P,tol) 将 P 元素之间的跳跃与跳跃阈值 tol 进行比较，而不是与默认值 π 弧度进行比较。如果您指定的跳跃阈值小于 π，unwrap 将使用默认的跳跃阈值 π。

Q = unwrap(P,[],dim)沿维度 dim 展开。

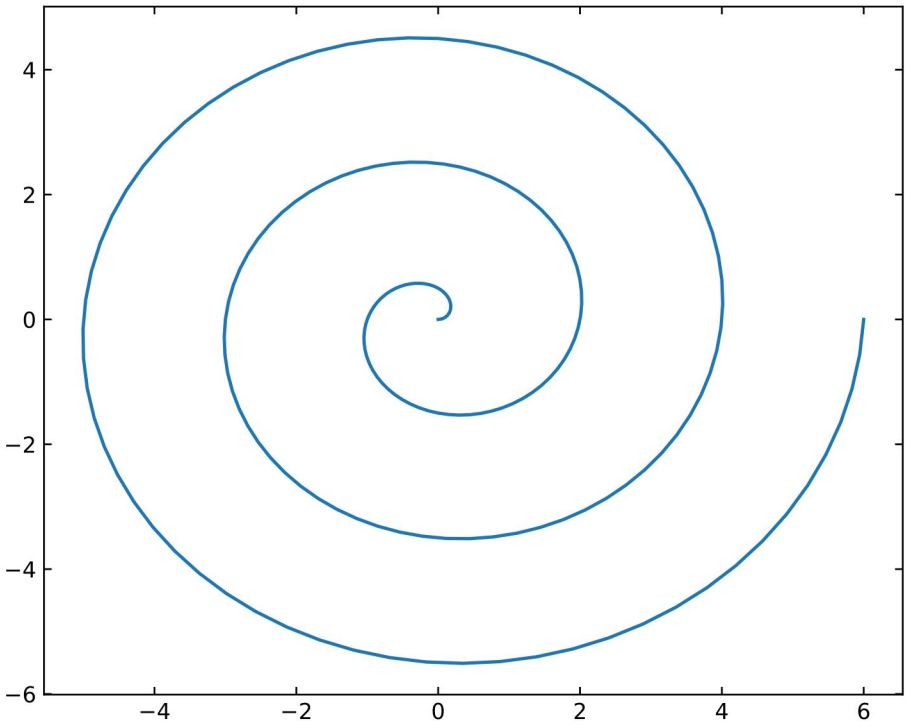
Q = unwrap(P,tol,dim) 使用跳跃阈值 tol 沿维度 dim 展开。

C. 示例

(1). 螺旋线的正确相位角

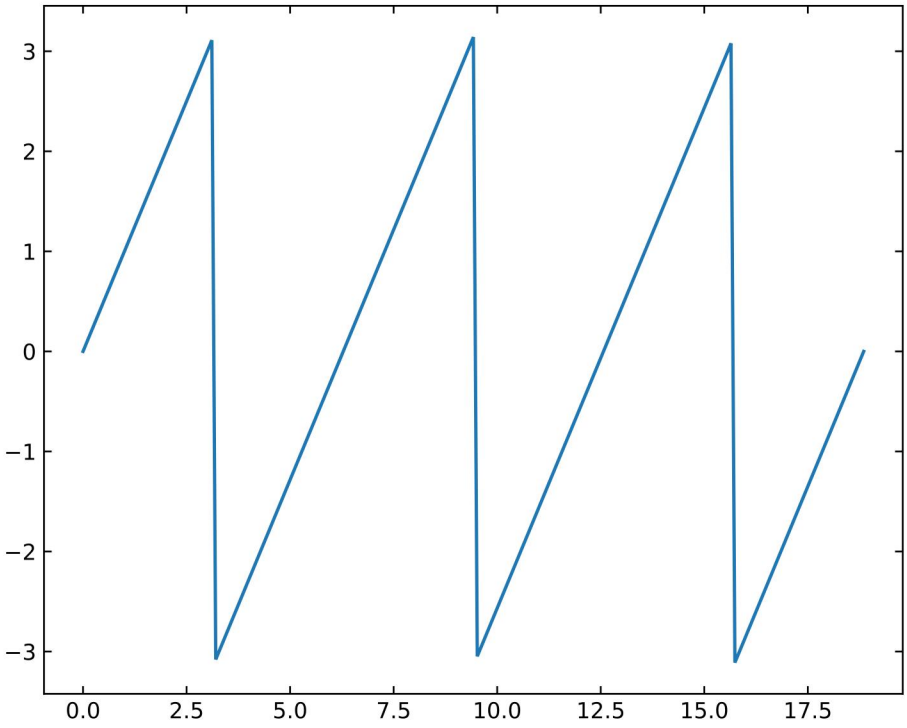
定义相位角为 0 至 6π 的螺旋线的 x 坐标和 y 坐标。绘制螺旋线。

```
t = LinRange(0,6pi,201)
x = t/pi.*cos.(t)
y = t/pi.*sin.(t)
plot(x,y)
```



使用 atan 函数基于螺旋线的 x 和 y 坐标求其相位角。atan 函数返回范围在 −π 到 π 的闭区间内的角度值。

```
P = atan.(y,x)
plot(t,P)
```



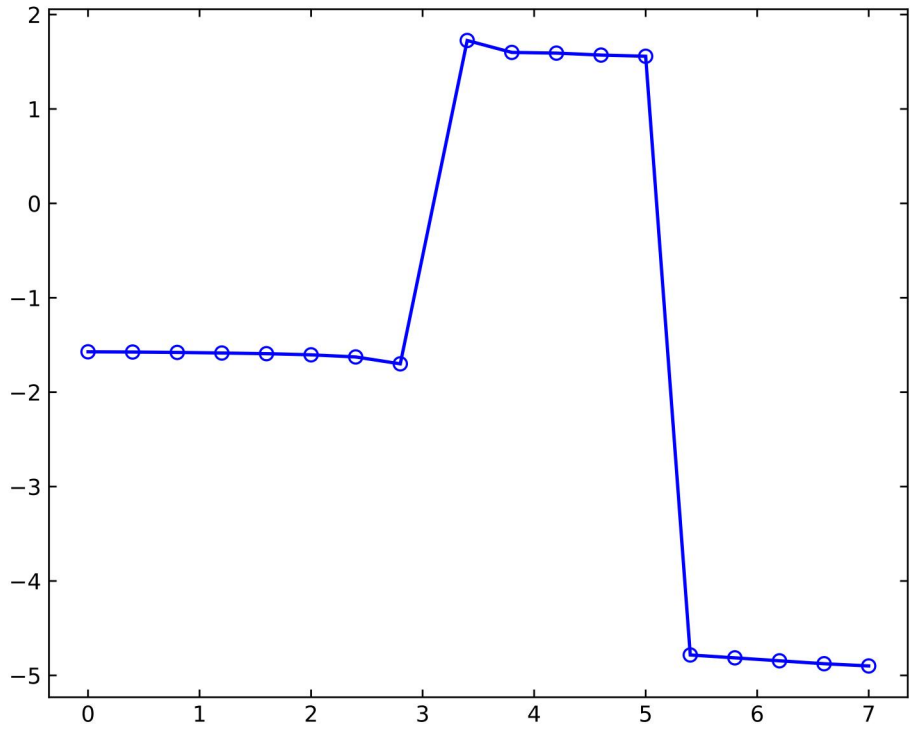
请注意，该图有不连续性。使用 unwrap 消除不连续性。当 P 的连续元素之间的相位差大于或等于跳跃阈值 π 弧度时，unwrap 会将角度增加 ±2π 的倍数。平移后的相位角 Q 位于从 0 到 6π 的区间内。

(2). 使用不同闯值平移相位角

移动频率响应的相位角。相位曲线有两次跳跃。第一次跳跃发生在 W = 3 和 W = 3.4 之间，跳跃幅度为 3.4250 弧度，第二次跳跃发生在 W = 5 和 W = 5.4 之间，跳跃幅度为 6.3420 弧度。绘制相位曲线。

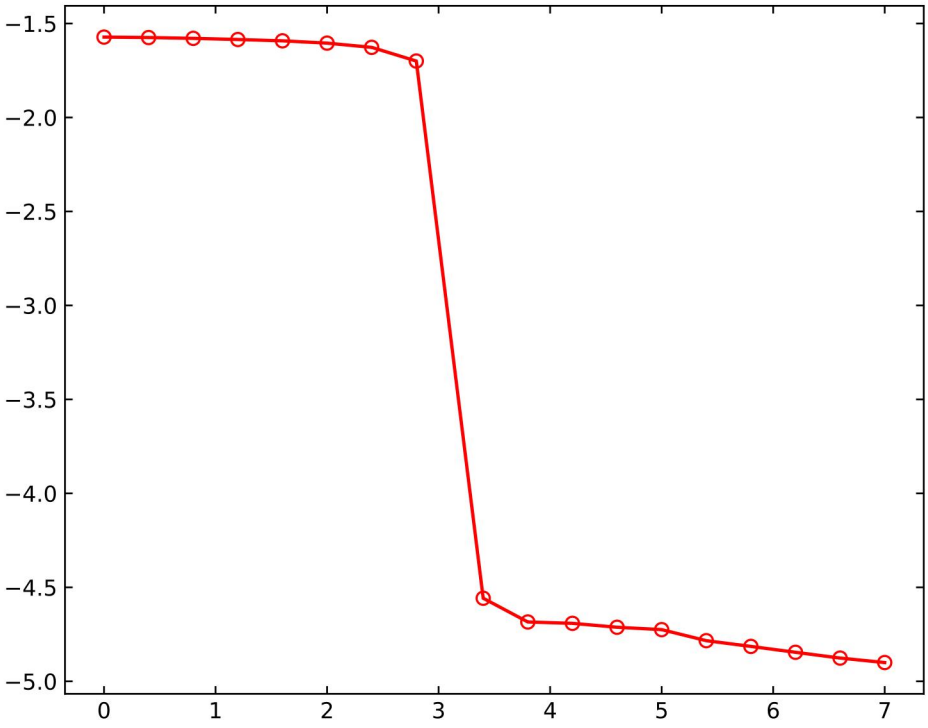
```
W = [0:0.4:3..., 3.4:0.4:5..., 5.4:0.4:7...]
P = [-1.5723
-1.5747
-1.5790
-1.5852
-1.5922
-1.6044
-1.6269
-1.6998
1.7252
1.5989
1.5916
```

```
1.5708
1.5582
-4.7838
-4.8143
-4.8456
-4.8764
-4.9002]
plot(W,P,"bo-")
```



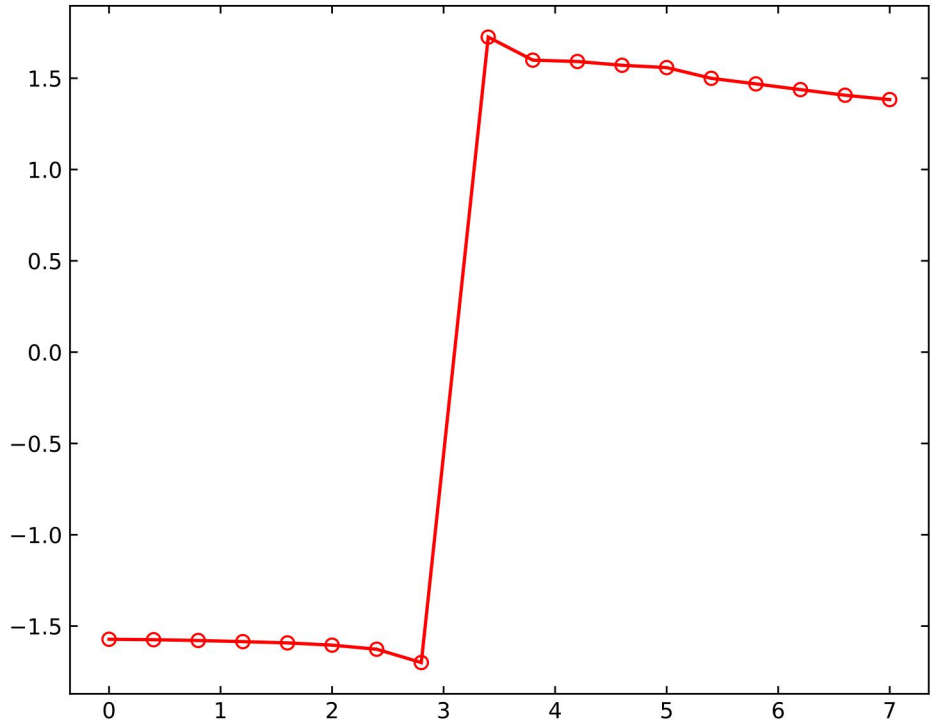
使用 unwrap 按默认跳跃阈值 π 弧度来平移相位角。绘制平移后的相位曲线。两次跳跃都发生了平移，因为它们大于跳跃阈值 π 弧度。

```
plot(W,unwrap(P),"ro-")
```



现在使用 5 弧度的跳跃阈值平移相位角。绘制平移后的相位曲线。第一次跳跃不会平移，因为它小于跳跃阈值 5 弧度。

```
plot(W,unwrap(P,5),"ro-")
```



(3). 将相移应用于矩阵

定义包含相位角的两列矩阵 P。

```
P = [0 7.07; 0.19 0.98; 6.67 1.18; 0.59 1.37; 0.78 1.56]
```

```
P = 5×2 Matrix{Float64}:
0.0   7.07
0.19  0.98
6.67  1.18
0.59  1.37
0.78  1.56
```

与其余数据相比，相位角 `P[1,2] = 7.07` 和 `P[3,1] = 6.67` 的相位差大于 π。

首先按列比较元素以展开相位角。将 `dim` 参数指定为 `1`。通过将第二个参数指定为 `[]`，使用默认跳转阈值 π。

```
dim = 1
P1 = unwrap(P,[],dim)
```

```
P1 = 5×2 Matrix{Float64}:
0.0       7.07
0.19      7.26319
0.386815  7.46319
```

```
0.59      7.65319
0.78      7.84319
```

要按行而不是按列平移相位角，请将 `dim` 指定为 `2` 而不是 `1`。

```
dim = 2
P2 = unwrap(P1,[],dim)
```

```
P2 = 5×2 Matrix{Float64}:
0.0       0.786815
0.19      0.98
0.386815  1.18
0.59      1.37
0.78      1.56
```

3.2.1.5 离散数学

质因数、阶乘、排列、有理分式、最小公倍数、最大公约数。

离散数学函数对整数（...、-2、-1、0、1、2、...）执行运算，或以整数返回离散输出。您可以使用这些函数来分解大数、计算阶乘、计算排列组合或求解最大公分母。

函数名	简介
factor	质因数
factorial	输入的阶乘
gcd	最大公约数
isprime	确定哪些数组元素为质数
lcm	最小公倍数
nchoosek	二项式系数或所有组合
perms	所有可能的排列
primes	小于等于输入值的质数
prevprime	小于等于输入值的质数
nextprime	大于等于输入值的质数
nthprime	返回第 n 个质数
binomial	二项式系数或所有组合
Rational	有理输出

rationalize	将浮点数 x 近似为具有给定整数类型分量的有理数。 结果将与 x 相差不超过 tol
matchpairs	求解线性分配问题
rats	有理输出

3.2.1.5.1 factor

质因数

A. 语法

```
f = factor(n)
```

B. 说明

f = factor(n) 返回包含 n 的质因数的向量。向量 f 与 n 具有相同的数据类型。

C. 示例

(1). 有符号整数值的质因数

```
using TySymbolicMath
f = factor(200)
```

```
f = 5-element Vector{Int64}:
2
2
2
5
5
```

(2). 无符号整数值的质因数

创建实值的数值向量。

```
using TySymbolicMath
n = UInt16(138)
f = factor(n)
```

```
f = 3-element Vector{Int64}:
2
3
23
```

乘以 f 的元素以重新生成 n

```
Int(prod(n))
```

```
ans = 138
```

3.2.1.5.2 factoria

输入的阶乘

A. 语法

```
f = factorial(n)
```

B. 说明

f = factorial(n) 返回所有小于或等于 n 的正整数的乘积，其中 n 为非负整数值。如果 n 为数组，则 f 包含 n 的每个值的阶乘。f 与 n 具有相同的数据类型和大小。

n 的阶乘通常使用感叹号字符以数学表示法写为 n!。

C. 示例

(1). 10!

```
f = factorial(10)
```

```
f = 3628800
```

(2). 22!

```
f = factorial(BigInt(22))
```

```
f = 1124000727777607680000
```

(3). 数组元素的阶乘

```
n = [0 1 2; 3 4 5]
f = factorial.(n)
```

```
2×3 Matrix{Int64}:
1   1    2
6  24  120
```

(4). 无符号整数值的阶乘

```
n = UInt64.([5,10,15,20])
f = factorial.(n)
```

```
4-element Vector{UInt64}:
```

```
0x0000000000000078
0x0000000000375f00
0x0000013077775800
0x21c3677c82b40000
```

3.2.1.5.3 gcd

最大公约数

A. 语法

```
G = gcd(A,B)
```

B. 说明

G = gcd(A,B) 返回 A 和 B 的元素的最大公约数。G 中的元素始终是非负值，gcd(0,0) 返回 0。此语法支持任何数值类型的输入。

C. 示例

(1). 双精度值的最大公约数

```
A = [-5 17; 10 0];
B = [-15 3; 100 0];
G = gcd.(A,B)
```

```
2×2 Matrix{Int64}:
5  1
10  0
```

gcd 返回正值，即使输入为负数也是如此。

(2). 无符号整数的最大公约数

```
A = UInt16.([255 511 15])
B = UInt16.([15 127 1023])
G = gcd.(A,B)
```

```
1×3 Matrix{UInt16}:
0x000f  0x0001  0x0003
```

3.2.1.5.4 isprime

确定哪些数组元素为质数

A. 语法

```
TF = isprime(x)
```

B. 说明

TF = isprime(X) 返回与 X 大小相同的逻辑数组。如果 X(i) 为质数，则 TF(i) 的值为 true。否则，值为 false。

C. 示例

(1). 确定双精度整数值是否为质数

```
tf = isprime.([2,3,0,6,10])
```

```
5-element BitVector:
1
1
0
0
0
```

2 和 3 是质数，但 0、6 和 10 不是。

(2). 确定无符号的整数值是否为质数

```
x = UInt16.([333,71,99])
tf = isprime.(x)
```

```
3-element BitVector:
0
1
0
```

71 是质数，但 333 和 99 不是。

3.2.1.5.5 lcm

最小公倍数

A. 语法

```
L = lcm(A,B)
```

B. 说明

L = lcm(A,B) 返回 A 和 B 元素的最小公倍数。

C. 示例

(1). 双精度数组和标量的最小公倍数

```
A = [5 17; 10 60]
B = 45
L = lcm.(A,B)
```

```
2×2 Matrix{Int64}:
45  765
90  180
```

(2). 无符号整数的最小公倍数

```
A = UInt16.([255 511 15])
B = UInt16.([15 127 1023])
L = lcm.(A,B)
```

```
1×3 Matrix{UInt16}:
0x00ff  0xfd81  0x13fb
```

3.2.1.5.6 nchoosek

二项式系数或所有组合

A. 语法

```
b = nchoosek(n,k)
C = nchoosek(v,k)
```

B. 说明

b = nchoosek(n,k) 返回二项式系数，定义为

$$ {}_nC_k = \binom{n}{k} = \frac{n!}{(n-k)!k!} $$

这是从 n 项中一次取 k 项的组合的数目。n 和 k 必须为非负整数。

C = nchoosek(v,k) 返回一个矩阵，其中包含了从向量 v 的元素中一次取 k 个元素的所有组合。矩阵 C 有 k 列和 m!/((m－k)! k!) 行，其中 m 为 length(v)。

C. 示例

(1). 二项式系数,“5 选 4”

```
b = nchoosek(5,4)
```

```
b = 5
```

(2). 五个数，一次取其中四个的所有组合

```
v = 2:2:10
C = nchoosek(v,4)
```

```
2.0  4.0  6.0   8.0
2.0  4.0  6.0  10.0
2.0  4.0  8.0  10.0
2.0  6.0  8.0  10.0
4.0  6.0  8.0  10.0
```

(3). 三个无符号整数，一次取其中两个的所有组合

```
v = UInt16.([10,20,30])
C = nchoosek(v,UInt16(2))
```

```
3×2 Matrix{UInt16}:
 0x000a  0x0014
 0x000a  0x001e
 0x0014  0x001e
```

3.2.1.5.7 perms

所有可能的排列

A. 语法

```
P = perms(v)
```

B. 说明

P = perms(v) 返回的矩阵包含了向量 v 中元素按字典顺序反序的所有排列。P 的每一行包含 v 中 n 个元素的一个不同排列。矩阵 P 具有与 v 相同的数据类型，包含 n! 行和 n 列。

C. 示例

(1). 双精度整数的所有排列

```
v = [2 4 6]
```

```
P = perms(v)
```

```
6×3 Matrix{Int64}:
6  4  2
6  2  4
4  6  2
4  2  6
2  6  4
2  4  6
```

(2). 无符号整数的所有排列

```
v = UInt16.([1023 4095 65535])
P = perms(v)
```

```
6×3 Matrix{UInt16}:
0xffff  0x0fff  0x03ff
0xffff  0x03ff  0x0fff
0x0fff  0xffff  0x03ff
0x0fff  0x03ff  0xffff
0x03ff  0xffff  0x0fff
0x03ff  0x0fff  0xffff
```

(3). 复数的所有排列

```
v = [1+1im 2+1im 3+1im]
P = perms(v)
```

```
6×3 Matrix{Complex{Int64}}:
3+1im  2+1im  1+1im
3+1im  1+1im  2+1im
2+1im  3+1im  1+1im
2+1im  1+1im  3+1im
1+1im  3+1im  2+1im
1+1im  2+1im  3+1im
```

3.2.1.5.8 primes

小于等于输入值的质数

A. 语法

```
p = primes(n)
```

B. 说明

p = primes(n) 返回包含所有小于或等于 n 的质数的行向量。p 与 n 具有相同的数据类型。

C. 示例

(1). 小于或等于 25 的质数

```
p = primes(25)
```

```
9-element Vector{Int64}:
2
3
5
7
11
13
17
19
23
```

(2). 小于或等于无符号整数的质数

```
n = UInt16(12)
p = primes(n)
```

```
5-element Vector{Int64}:
2
3
5
7
11
```

3.2.1.5.9 prevprime

前一个素数

A. 语法

```
prevprime(n)
```

B. 说明

prevprime(n)返回小于或等于 的最大素数 n。如果 n 是向量或矩阵，则按元素对 进行操作。

C. 示例

(1). 查找上一个质数

找出小于 100 的最大素数。

```
prevprime(100)
```

```
ans = 97
```

通过将输入指定为向量，找到小于 1000、10000、100000 的最大素数。

```
v = [1000,10000,100000]
prevprime.(v)
```

```
3-element Vector{Int64}:
997
9973
99991
```

3.2.1.5.10 nextprime

后一个素数

A. 语法

```
nextprime(n)
```

B. 说明

nextprime(n)返回下一个大于或等于 的素数 n。如果 n 是向量或矩阵，则按元素对 进行操作。

C. 示例

(1). 查找下一个质数

```
nextprime(100)
```

```
ans = 101
```

通过将输入指定为向量 1000、10000、100000 的下一个素数

```
v = [1000,10000,100000]
nextprime.(v)
```

```
3-element Vector{Int64}:
1009
10007
100003
```

3.2.1.5.11 nthprime

第 n 个素数

A. 语法

```
Nthprim
e(n)
```

B. 说明

nthprime(n)返回第 n 个质数。

C. 示例

(1). 找到第 N 个质数

找出第 223 个素数。

```
nthprime(223)
```

```
ans = 1409
```

找出第 10 个、第 100 个和第 1000 个质数。

```
n = [10,100,1000]
nthprime.(n)
```

```
3-element Vector{Int64}:
29
541
7919
```

3.2.1.5.12 binomia

二项式系数或所有组合

A. 语法

```
b = binomial(n,k)
```

B. 说明

b = binomial(n,k)，用于返回二项式系数 n(k),它是多项式(1+X)^n 中第 k 项的系数。它的公式为

$$ {}_nC_k = \binom{n}{k} = \frac{n!}{(n-k)!k!} $$

在这里 n !是的阶乘。且 n 和为整数，如果 n 是负整数，那么其公式为

$$ \binom{n}{k} = (-1)^k \binom{k-n-1}{k} $$

C. 示例

(1). 返回二项式系数

```
b = binomial(6,4)
```

```
b = 15
```

(2). 返回多个二项式系数

```
b1= binomial(6,3)
b2= binomial(-6,3)
b3= binomial(-6,-2)
b4 = binomial(5,2)
return b1,b2,b3,b4
```

```
(20, -56, 0, 10)
```

3.2.1.5.13 Rational

有理输出

A. 语法

```
S = Rational(X)
```

B. 说明

S= Rational(X) 返回一个包含 X 的元素的有理近似值的字符向量。...表示无法在分配的空间中输出，但与 X 中的其他元素相比并非可忽略不计的元素。

C. 示例

(1). 矩阵有理表示形式

```
X=hilb(4)
```

```
4×4 Matrix{Float64}:
```

```
1.0         0.5         0.333333    0.25
0.5         0.333333    0.25        0.2
0.333333    0.25        0.2         0.166667
0.25        0.2         0.166667    0.142857
```

使用 Rational 查看矩阵的有理表示形式。

```
S = Rational.(X)
```

```
4×4 Matrix{Rational{Int64}}:
1//1                                                                    1//2
6004799503160661//18014398509481984                 1//4
1//2                                  6004799503160661//18014398509481984
1//4                     3602879701896397//18014398509481984
6004799503160661//18014398509481984                                    1//4
3602879701896397//18014398509481984
6004799503160661//36028797018963968
1//4                                  3602879701896397//18014398509481984
6004799503160661//36028797018963968
2573485501354569//18014398509481984
```

3.2.1.5.14 rationalize

将浮点数 x 近似为具有给定整数类型分量的有理数。 结果将与 x 相差不超过 tol

A. 语法

```
S= rationalize(X)
```

B. 说明

S= rationalize(X) 返回一个包含 X 的元素的有理近似值的字符向量。

C. 示例

(1). 矩阵有理表示形式

```
X=hilb(4)
```

```
4×4 Matrix{Float64}:
1.0         0.5         0.333333    0.25
0.5         0.333333    0.25        0.2
0.333333    0.25        0.2         0.166667
0.25        0.2         0.166667    0.142857
```

使用 Rational 查看矩阵的有理表示形式。

```
R=rationalize.(X)
```

```
4×4 Matrix{Rational{Int64}}:
1//1    1//2    1//3    1//4
1//2    1//3    1//4    1//5
1//3    1//4    1//5    1//6
1//4    1//5    1//6    1//7
```

3.2.1.5.15 matchpairs

求解线性分配问题

A. 语法

```
M,uR,uC = matchpairs(Cost,costUnmatched)
M,uR,uC = matchpairs(Cost,costUnmatched,goal)
```

B. 说明

M,uR,uC = matchpairs(Cost,costUnmatched) 求解矩阵 Cost 的行和列的线性分配问题。该函数按总成本最小的方法将每行分配给一列。costUnmatched 指定存在未分配行时的每行成本，以及存在未分配行的列时的每列成本。还返回 uR 中未匹配行的索引和 uC 中未匹配列的索引。示例

M,uR,uC = matchpairs(Cost,costUnmatched,goal) 指定优化目标，goal 可以是 "min" 或 "max" 以生成总成本最小化或最大化的匹配。示例。

C. 示例

(1). 以最低成本分配航班

将销售人员分配到各个航班，使总交通成本最小化。

一家公司有四名销售人员，他们需要去全国主要城市出差。该公司必须为他们预订航班，并希望费用支出尽可能少。这些销售人员分布在全国不同地方，因此他们飞往每个城市的费用各不相同。

下表显示了每位销售人员飞往每个主要城市的成本。

	Dallas	Chicago	New York City	St. Louis
Fred	$600	$670	$960	$560
Beth	$900	$280	$970	$540
Sue	$310	$350	$950	$820

Greg	$325	$290	$600	$540

每个城市都代表一个销售机会。如果错过一个城市，则该公司将损失平均 2,000 美元的收益。

创建一个成本矩阵来表示每位销售人员飞往每个城市的成本。

```
C = [600 670 960 560
     900 280 970 540
     310 350 950 820
     325 290 600 540];
```

使用 matchpairs 将销售人员分配到各个城市并使成本最小。将未分配成本指定为 1,000，因为如果有一行和一列未匹配，则计两次未分配成本。

```
M = matchpairs(C,1000)[1]
```

```
M = 4×2 Matrix{Int64}:
 3  1
 2  2
 4  3
 1  4
```

matchpairs 计算为每个城市分配一位销售人员的最经济的方式。

	Dallas	Chicago	New York City	St. Louis
Fred	$600	$670	$960	**$560**
Beth	$900	**$280**	$970	$540
Sue	**$310**	$350	$950	$820
Greg	$325	$290	**$600**	$540

(2). 行数和列数不相等

当成本矩阵中的列数远多于行数时，请将行匹配到列。

创建一个 3×8 成本矩阵。由于只有三行，对于八列，matchpairs 最多只能产生三个匹配。

```
rng = MT19937ar(5489)
C = randi(rng, [10, 100], 3, 8)
```

```
C = 3×8 Matrix{Int64}:
 84  93  35  97  97  22  82  13
 92  67  59  24  54  48  97  87
 21  18  97  98  82  93  69  94
```

使用 matchpairs 匹配成本矩阵的行和列。要获得最大匹配数，请使用较大的未分配成本（相对于成本矩阵中各项的模）。并返回未匹配的行和列的索引。

```
M, uR, uC = matchpairs(C, 1e4)
```

```
M = 3×2 Matrix{Int64}:
 3  2
 2  4
 1  8
 uR = Int64[]
 uC = 5-element Vector{Int64}:
 1
 3
 5
 6
 7
```

C 中有五列不匹配任何行。

(3). 分配出租车以实现利润最大化

为路线分配出租车，使利润最大化。

一家出租车公司有几个来自市内各地的乘车请求。该公司希望以最赚钱的方式派遣数量有限的出租车。

下表显示了五个乘车请求中每个请求的出租车费估计值。五个乘车请求中只有三个可以得到满足。

	Ride 1	Ride 2	Ride 3	Ride 4	Ride 5
Cab A	$5.70	$6.30	$3.10	$4.80	$3.50
Cab B	$5.80	$6.40	$3.30	$4.70	$3.20
Cab C	$5.70	$6.30	$3.20	$4.90	$3.40

创建一个利润矩阵来表示每个行程的利润。

```
P = [5.7 6.3 3.1 4.8 3.5
    5.8 6.4 3.3 4.7 3.2
    5.7 6.3 3.2 4.9 3.4];
```

使用 matchpairs 将出租车匹配到最赚钱的行程。指定三个输出以返回未匹配的行和列，并使用 "max" 选项以最大化利润。将未分配成本指定为零，因为公司不会从未载客的出租车或未满足的乘车请求中赚钱。

```
costUnmatched = 0;
M,uR,uC = matchpairs(P,costUnmatched,"max")
```

```
M = 3×2 Matrix{Int64}:
 1  1
 2  2
 3  4
 uR = Int64[]
 uC = 2-element Vector{Int64}:
 3
```

```
5
```

matchpairs 计算最赚钱的乘车分配方式。在该解中，第 3 和第 5 个乘车请未得到满足。

	Ride 1	Ride 2	Ride 3	Ride 4	Ride 5
Cab A	**$5.70**	$6.30	$3.10	$4.80	$3.50
Cab B	$5.80	**$6.40**	$3.30	$4.70	$3.20
Cab C	$5.70	$6.30	$3.20	**$4.90**	$3.40

根据求出的解计算总利润。由于 costUnmatched 为零，您只需将每个匹配的利润相加即可。

```
TotalProfits = sum(P[sub2ind(size(P), M[:,1], M[:,2])])
```

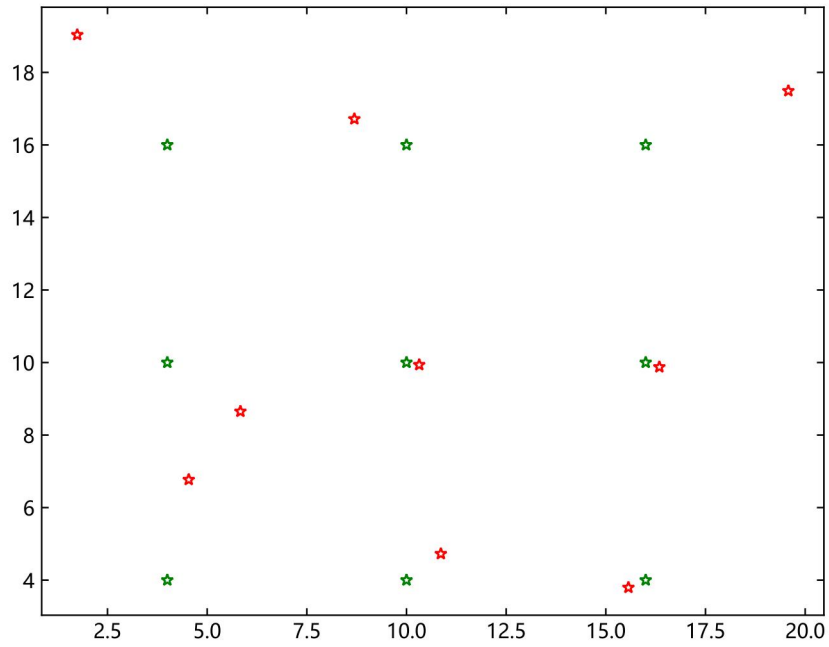
```
TotalProfits = 17.0
```

(4). 跟踪点位置随时间的变化

使用 matchpairs 跟踪几个点的移动，使距离的总变化最小。

用绿色绘制时间 t=0 时的网格点。时间 t=1 时，一些点在随机方向上移动一小段距离。

```
x, y = meshgrid2(4:6:16)
x0 = x[:]
y0 = y[:]
plot(x0, y0, "*g")
hold("on")
rng = MT19937ar(5489)
x1 = x0 + randn(rng, size(x0))
y1 = y0 + randn(rng, size(y0))
plot(x1, y1, "*r")
```



使用 matchpairs 将 t=0 时的点与 t=1 时的点进行匹配。为此，首先计算一个成本矩阵，其中 C[i,j] 是从 i 点到 j 点的欧几里德距离。

```
C = zeros(length(x1), length(x1))
for k in axes(C, 1)
    C[k, :] = vecnorm([-x0 .+ x1[k] -y0 .+ y1[k]], 2, 2)
end
C
```

```
C = 9×9 Matrix{Float64}:
  2.82115    3.275      9.24621    6.12428    6.34615    10.7257    11.7922
11.9089    14.7169
  4.99867    2.27713    7.57522    6.2434     4.37935     8.44851   11.1792
10.2553    12.5447
 15.2037     9.31301    3.78327   17.1539    12.2408      8.79882   20.7211
16.8803    14.5783
  6.90041    8.6551    13.1987     1.12674    5.3446     11.3075     5.18878
7.36333   12.3901
  8.6703     6.31908    8.75714    5.9455     0.324942    6.07143    8.21728
5.68158    8.30885
 13.553      8.1918     4.74644   12.7818     6.84089     1.49027   14.6652
9.92422    7.34256
 11.5682    13.1257    16.815      5.57018    8.33586    13.4144     0.479597
6.2201    12.2127
 13.6699    12.3432    13.7784     8.64607    6.34384     8.81669    5.88584
```

```
0.364422   6.13372
 20.6072    17.2853    15.6495    16.5444    12.159       9.69355   13.9562
8.30063    3.8761
```

接下来，使用 matchpairs 匹配成本矩阵中的行和列。将未分配成本指定为 1。相对于成本矩阵中的项，未分配成本非常低，matchpairs 很可能会留下一些不匹配的点。

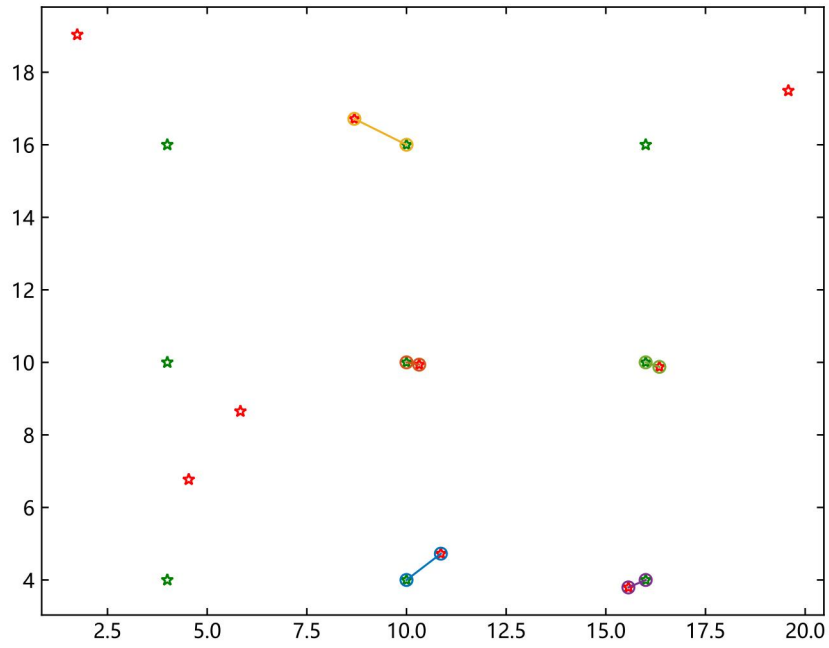
```
M = matchpairs(C,1)[1]
```

```
M = 5×2 Matrix{Int64}:
 4  4
 5  5
 6  6
 7  7
 8  8
```

值 M[:,2] 对应于原始点 (x0,y0)，而值 M[:,1] 对应于移动后的点 (x1,y1)。

绘制匹配的点对组。相对原始点移动的距离超过 2*costUnmatched 的点保持未匹配。

```
xc = [x0[M[:,2]]'; x1[M[:,1]]'];
yc = [y0[M[:,2]]'; y1[M[:,1]]'];
plot(xc,yc,"-o")
```



3.2.1.5.16 rats

有理输出

A. 语法

```
S = rats(X)
S = rats(X,strlen)
```

B. 说明

S = rats(X) 返回一个包含 X 的元素的有理近似值的字符向量。星号指示无法在分配的空间中输出，但与 X 中的其他元素相比并非可忽略不计的元素。示例。

S = rats(X,strlen) 指定用于有理近似值的字符向量的长度。对于实数输入，strlength(S) 等于 strlen+1，而对于复数输入，它等于 2*strlen+3。有理近似值使用与指定的长度成反比的容差，如算法一节中所述。示例。

C. 示例

(1). 矩矩阵有理表示形式

矩阵有理表示形式

```
X=hilb(4)
```

```
4×4 Matrix{Float64}:
1.0        0.5        0.333333   0.25
0.5        0.333333   0.25       0.2
0.333333   0.25       0.2        0.166667
0.25       0.2        0.166667   0.142857
```

使用 rats 查看矩阵的有理表示形式。

```
S = rats(X)
```

```
R = 4-element Vector{String}:
 "     1          1/2         1/3         1/4     "
 "    1/2         1/3         1/4         1/5     "
 "    1/3         1/4         1/5         1/6     "
 "    1/4         1/5         1/6         1/7     "
```

(2). 矩阵有理表示形式

使用默认字符向量长度和逼近容差计算 pi 的有理表示形式

```
rats(pi)
```

```
ans = "   355/113   "
```

调整输出的长度，这也会调整逼近容差。

```
rats(pi,20)
```

```
abs = "   104348/33215    "
```

结果有理近似值具有更高的精度。随着输出长度的增加，容差会减少。

再次调整输出长度以实现更高的精度。

```
rats(pi,25)
```

```
ans = "      1146408/364913      "
```

结果近似值与保留 10 位小数的 pi 一致。

3.2.1.6 多项式

曲线拟合、根、部分分式展开 多项式是包含非负整数指数的单个变量的方程。

函数名	简介
detrend	去除多项式趋势
poly	具有指定根的多项式或特征多项式
polyeig	多项式特征值问题
residue	部分分式展开（部分分式分解）
roots	多项式根
polyval	多项式计算
polyvalm	矩阵多项式计算
polyint	多项式积分
polyder	多项式微分
polyreduce	去除多项式前系数为 0 的无用项
polyfit polyfits2 polyfits3	多项式曲线拟合

3.2.1.6.1 detrend

去除多项式趋势

A. 语法

```
y = detrend(x)
y = detrend(x,n)
y = detrend(x,n,bp)
y = detrend(___,nanflag)
y = detrend(___;Continuous = true,SamplePoints = nothing)
```

B. 说明

y = detrend(x) 从 x 的数据中去除最佳直线拟合线。示例

如果 x 是向量，则 detrend 从 x 的元素中减去趋势。

如果 x 是矩阵，则 detrend 分别对每列进行运算，从对应的列中减去每个趋势。

y = detrend(x,n) 去除 n 次多项式趋势。例如，当 n = 0 时，detrend 从 x 中删除均值。当 n = 1 时，detrend 去除线性趋势，这等效于上述语法。当 n = 2 时，detrend 去除二次趋势。示例

y = detrend(x,n,bp) 去除由断点 bp 定义段的连续分段趋势。示例

y = detrend(___,nanflag) 指定在使用上述任一语法时如何处理 NaN 值。例如，detrend(x,'omitnan') 在计算趋势之前删除 NaN 值，而 detrend(x,'includenan') 包括这些值（默认）。

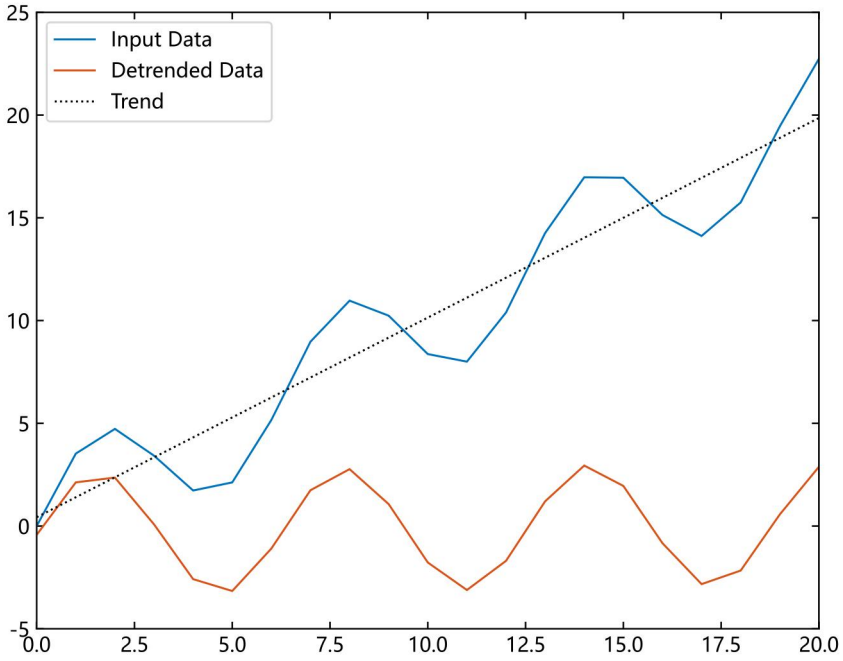
y = detrend(___,Name = Value) 使用一个或多个名称-值对组指定其他参数。例如，detrend(x,1,bp,'Continuous',false) 指定拟合趋势可以有不连续趋势。示例

C. 示例

(1). 连续线性趋势

创建一个数据向量，并去除连续线性趋势。绘制原始数据、去趋势后的数据和线性趋势。

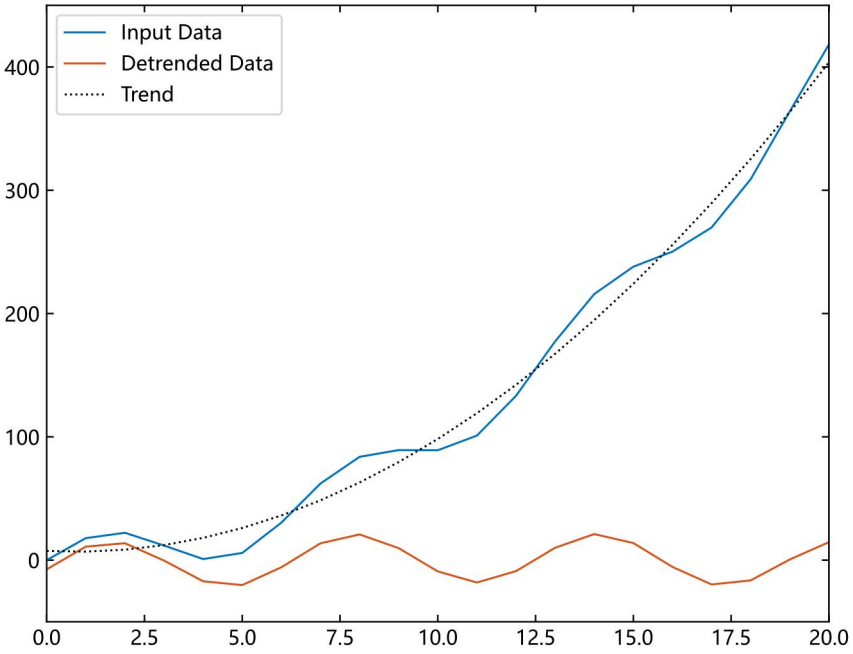
```
t = 0:20;
x = 3*sin.(t) + t;
y = detrend(x)
plot(t,x,t,y,t,x-y,":k")
legend(["Input Data","Detrended Data","Trend"],loc="northwest")
```



(2). 连续二次趋势

创建一个数据向量，并去除连续二次趋势。绘制原始数据、去趋势后的数据和趋势。

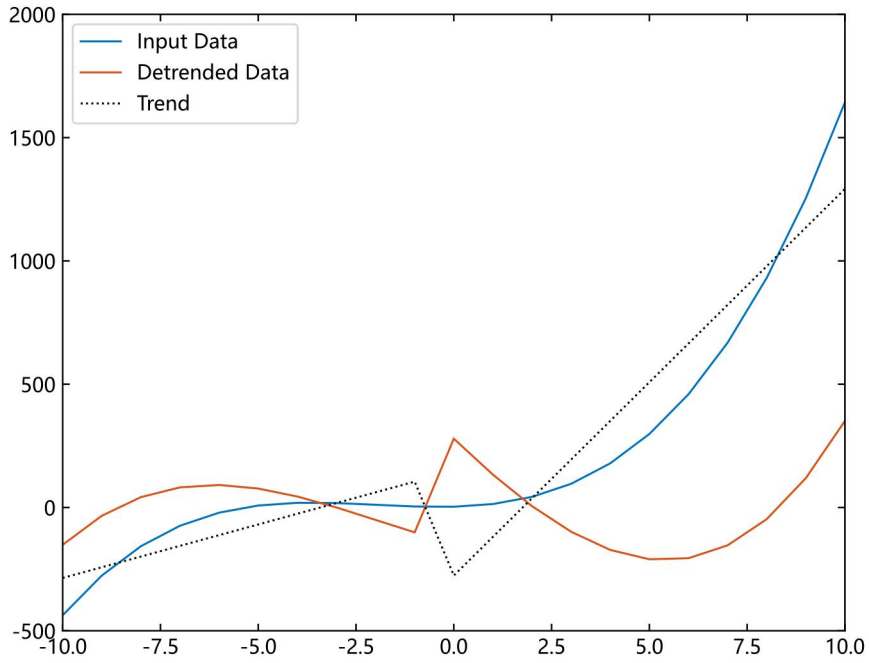
```
t = 0:20;
x = 20*sin.(t) + t.^2;
y = detrend(x,2);
plot(t,x,t,y,t,x-y,":k")
legend(["Input Data","Detrended Data","Trend"],loc="northwest")
```



(3). 不连续线性趋势

创建一个数据向量，并使用 0 处的断点去除分段线性趋势。指定所得到的输出可以是不连续的。绘制原始数据、去趋势后的数据和趋势。

```
t = -10:10;
x = t.^3 .+ 6*t.^2 .+ 4*t .+ 3
bp = 0;
y = detrend(x,1,bp,SamplePoints =t,Continuous = false)
plot(t,x,t,y,t,x-y,":k")
legend(["Input Data","Detrended Data","Trend"],loc="northwest")
```



3.2.1.6.2 poly

具有指定根的多项式或特征多项式

A. 语法

```
p = poly(r)
p = poly(A)
```

B. 说明

p = poly(r)（其中 r 是向量）返回多项式的系数，其中多项式的根是 r 的元素。

p = poly(A)（其中 A 是 n×n 矩阵）返回矩阵 det(λI − A) 的特征多项式的 n+1 个系数。

C. 示例

(1). 来自特征值的特征多项式

计算矩阵 A 的特征值。

```
A = [1 8 -10; -4 2 4; -5 2 8]
```

```
A = 3×3 Matrix{Int64}:
1   8  -10
```

```
-4   2    4
-5   2    8
```

```
e = eigvals(A)
```

```
e = 3-element Vector{ComplexF64}:
-0.3109528850834832 - 2.6704004497568996im
-0.3109528850834832 + 2.6704004497568996im
11.621905770166965 + 0.0im
```

由于 e 中的特征值是 A 的特征多项式的根，使用 poly 可确定来自 e 中的值的特征多项式。

```
p = poly(e)
```

```
p = 4-element Vector{Float64}:
1.0
-10.999999999999998
-3.552713678800501e-15
-83.99999999999984
```

(2). 矩阵的特征多项式

使用 poly 来计算矩阵 A 的特征多项式。

```
A = [1 2 3; 4 5 6; 7 8 0]
```

```
A = 3×3 Matrix{Int64}:
1  2  3
4  5  6
7  8  0
```

```
p = poly(A)
```

```
p = 4-element Vector{Float64}:
1.0
-6.000000000000008
-72.00000000000007
-27.00000000000005
```

```
r = roots(p)
```

```
r = 3-element Vector{Float64}:
12.122893784632407
-0.38838384240732066
-5.734509942225073
```

3.2.1.6.3 polyeig

多项式特征值问题

A. 语法

```
x,e = polyeig(A0,A1,..,AP)
```

B. 说明

X,e = polyeig(A0,A1,...,Ap) 还会返回大小为 n×n*p 的矩阵 X，其列是特征向量。

C. 示例

(1). 质量、阻尼和刚度矩阵的二次特征值问题

解算涉及质量矩阵 M、阻尼矩阵 C 和刚度矩阵 K 的二次特征值问题。运动方程中会发生此二次特征值问题：

$$M\frac{d^2y}{dt^2}+C\frac{dy}{dt}+Ky=f(t)$$

此方程适用于各种振荡系统，包括动态质量弹簧系统或 RLC 电子网络。基本解是 y(t)=xe^λt，因此 λ 和 x 都必须解算二次特征值问题 (QEP)，

$$\left(M\lambda^2+C\lambda+K\right)x=0$$

创建系数矩阵 M、C 和 K 来表示具有四个自由度的质量弹簧系统。系数矩阵全部是对称矩阵和半正定矩阵，M 是对角矩阵。

```
M = diagm([3 ,1 ,3 ,1])
```

```
4×4 Matrix{Int64}:
3  0  0  0
0  1  0  0
0  0  3  0
0  0  0  1
```

```
C = [0.4 0 -0.3 0; 0 0 0 0; -0.3 0 0.5 -0.2; 0 0 -0.2 0.2]
```

```
4×4 Matrix{Float64}:
0.4  0.0  -0.3   0.0
0.0  0.0   0.0   0.0
-0.3  0.0   0.5  -0.2
0.0  0.0  -0.2   0.2
```

```
K = [-7 2 4  0; 2 -4 2 0; 4 2 -9 3; 0 0 3 -3]
```

```
4×4 Matrix{Int64}:
-7   2   4   0
2  -4   2   0
```

```
4   2  -9   3
0   0   3  -3
```

使用 polyeig 针对特征值、特征向量和条件数解算 QEP。

```
X,e=polyeig(K,C,M)
```

```
X = 4×8 Matrix{Float64}:
-0.340934  -0.368074   -0.476676  1.0      0.861761   1.0        0.681127
-0.0724478
-0.658486   1.0        -0.397976  1.09244  0.93808    0.655696  -1.11447
1.0
1.0        0.0490422   0.206033  1.11927  0.961669  -0.402116  -0.599367
-0.409341
-1.38943   -0.139356    1.0       1.16485  1.0       -2.05765    1.0
0.434022
e
8×1 Matrix{Float64}:
2.2279087320479034
2.036350976643702
1.4752411434756634
0.3352944297785459
-0.3465512996736314
-1.6247783405292464
-2.153616198037308
-2.449849443705628
```

3.2.1.6.4 residue

部分分式展开（部分分式分解）

A. 语法

```
r,p,k = residue(b,a)
```

B. 说明

r,p,k = residue(b,a) 计算以如下形式展开的两个多项式之比的 部分分式展开式 的留数、极点和直项.

$$\frac{b(s)}{a(s)}=\frac{b_m s^m+b_{m-1}s^{m-1}+\ldots+b_1 s+b_0}{a_n s^n+a_{n-1}s^{n-1}+\ldots+a_1 s+a_0}=\frac{r_n}{s-p_n}+\ldots+\frac{r_2}{s-p_2}+\frac{r_1}{s-p_1}+k(s)$$

residue 的输入是由多项式 b = [bm ... b1 b0] 和 a = [an ... a1 a0] 的系数组成的向量。输出为留数 r = [rn ... r2 r1]、极点 p = [pn ... p2 p1] 和多项式 k。对

于大多数教科书问题，k 为 0 或常量。

C. 示例

(1). 求解具有实根的部分分式展开式

使用 residue 求以下多项式之比 F(s) 的部分分式展开式

$$F(s)=\frac{b(s)}{a(s)}=\frac{-4s+8}{s^2+6s+8}$$

```
b = [-4 ,8]
a = [1 ,6 ,8]
r,p,k = residue(b,a)
```

```
r = 2×1 Matrix{Float64}:
-12.0
8.0
p = 2-element Vector{Float64}:
-4.0
-2.0
k =
Any[]
```

此结果代表以下部分分式展开式

$$\frac{-4s+8}{s^2+6s+8}=\frac{-12}{s+4}+\frac{8}{s+2}$$

使用 residue 将部分分式展开转换回多项式系数。

```
b,a = residue(r,p,k)
```

```
b =
2-element Vector{Float64}:
-4.0
8.0
a =
3-element Vector{Float64}:
1.0
6.0
8.0
```

此结果表示初始分式 F(s)。

(2). 展开具有复数根和同次分子及分母的分式

如果分子的次数与分母的次数相等，输出 k 可以是非零值。

求解具有复数根和同次分子和分母的两个多项式 F(s) 之比的部分分式展示式，其中 F(s) 为

$$F(s)=\frac{b(s)}{a(s)}=\frac{2s^3+s^2}{s^3+s+1}$$

```
b = [2 ,1, 0, 0]
a = [1 ,0, 1, 1]
r,p,k = residue(b,a)
```

```
r =
3×1 Matrix{ComplexF64}:
-0.07083581187720656 + 2.220446049250313e-16im
0.5354179059386033 - 1.038991708709454im
0.5354179059386033 + 1.0389917087094538im
p =
3-element Vector{ComplexF64}:
-0.6823278038280195 + 0.0im
0.3411639019140098 + 1.1615413999972526im
0.3411639019140098 - 1.1615413999972526im
k =
1-element Vector{Float64}:
2.0
```

residue 返回代表部分分式展开式的复数根和极点，以及常项 k

$$F(s)=\frac{b(s)}{a(s)}=\frac{2s^3+s^2}{s^3+s^2+1}=\frac{0.5354+1.0390i}{s-(0.3412+1.1615i)}+\frac{0.5354-1.0390i}{s-(0.3412-1.1615i)}+\frac{-0.0708}{s+0.6823}+2$$

(3). 展开分子次数高于分母次数的分式

当分子次数大于分母次数时，输出 k 为代表 s 中多项式系数的向量。

使用 residue 执行 F(s) 的以下部分分式展开式。

$$F(s)=\frac{b(s)}{a(s)}=\frac{2s^4+s}{s^2+1}=\frac{0.5-1i}{s-1i}+\frac{0.5+1i}{s+1i}+2s^2-2$$

```
b = [2, 0, 0, 1, 0]
a = [1, 0, 1]
r,p,k = residue(b,a)
```

```
r =
2×1 Matrix{ComplexF64}:
0.5 - 1.0im
0.5 + 1.0im
p =
2-element Vector{ComplexF64}:
-0.0 - 1.0im
0.0 + 1.0im
k =
```

```
3-element Vector{Int64}:
2
0
-2
```

k 代表多项式 $2s^2 - 2$

3.2.1.6.5 roots

多项式根

A. 语法

```
p = roots(p)
```

B. 说明

p = roots(p) 以列向量的形式返回 p 表示的多项式的根。

C. 示例

(1). 二次多项式的根

对方程 3x^2-2x-4=0 求解。

创建一个向量，用它来代表多项式，然后计算多项式的根。

```
p = [3,-2,-4];
r = roots(p)
```

```
2-element Vector{Float64}:
1.5351837584879962
-0.8685170918213299
```

(2). 四次多项式的根对方程

x^4−1=0 求解。

创建一个向量，用它来代表多项式，然后计算多项式的根

```
p = [1,0,0,0,-1];
r = roots(p)
```

```
4-element Vector{ComplexF64}:
-1.0000000000000004 + 0.0im
8.326672684688674e-17 + 0.9999999999999996im
8.326672684688674e-17 - 0.9999999999999996im
0.9999999999999999 + 0.0im
```

3.2.1.6.6 polyval

多项式计算

A. 语法

```
y = polyval(p,x)
y,delta = polyval(p,x,s)
y = polyval(p,x,[],mu)
y,delta = polyval(p,x,s,mu)
```

B. 说明

y = polyval(p,x) 计算多项式 p 在 x 的每个点处的值。参数 p 是长度为 n+1 的向量，其元素是 n 次多项式的系数（降幂排序）：

$$p(x) = p_1 x^n + p_2 x^{n-1} + ... + p_n x + 1$$

虽然可以为不同目的使用 polyint、polyder 和 polyfit 等函数计算 p 中的多项式系数，但您也可以为系数指定任何向量。

要以矩阵方式计算多项式，请改用 polyvalm。

y,delta = polyval(p,x,s) 使用 polyfit 生成的可选输出结构体 S 来生成误差估计值。delta 是使用 p(x) 预测 x 处的未来观测值时的标准误差估计值。

y = polyval(p,x,[],mu) 或 y,delta = polyval(p,x,S,mu) 使用 polyfit 生成的可选输出 mu 来中心化和缩放数据。mu[1] 为 mean(x)，mu[2] 为 std(x)。使用这些值时，polyval 将 x 的中心置于零值处并缩放为具有单位标准差。

$$\hat{x} = \frac{x - \bar{x}}{\sigma_x}$$

这种中心化和缩放变换可改善多项式的数值属性。

C. 示例

(1). 计算几个点处的多项式值

计算多项式 p(x)=3x^2+2x+1 在点 x=5,7,9 处的值。多项式系数可以由向量 [3,2,1] 表示。

```
p = [3,2,1]
x = [5,7,9]
y = polyval(p,x)
```

```
y = 3-element Vector{Int64}:
```

```
86
162
262
```

(2). 对四次多项式求积分

计算定积分 $\int_{-1}^{3}(3x^4-4x^2+10x-25)dx$ 创建一个向量来表示多项式被积函数 3x^4-4x^2+10x-25。x^3 项不存在，因此系数为 0。 项不存在，因此系数为 0。

```
p = [3,0,-4,10,-25]
```

使用 polyint 和等于 0 的积分常量来对多项式求积分。

```
q = polyint(p)
```

```
q = 6-element Vector{Float64}:
0.6
0.0
-1.3333333333333333
5.0
-25.0
0.0
```

通过在积分极限上计算 q，求解积分的值。

```
a = -1
b = 3
I = diff(polyval(q,[a,b]))
```

```
I = 1-element Vector{Float64}:
49.06666666666665
```

3.2.1.6.7 polyvalm

矩阵多项式计算

A. 语法

```
Y = polyvalm(p,X)
```

B. 说明

Y = polyvalm(p,X)以矩阵方式返回多项式 p 的计算值。此计算方法等同于使用多项式 p 替换矩阵 X。

C. 示例

(1). 特征多项式的矩阵计算

求解 4 阶帕斯卡矩阵的特征多项式。

```
X = pascal(4)
```

```
X = 4×4 Matrix{Int64}:
1  1   1   1
1  2   3   4
1  3   6  10
1  4  10  20
```

```
p = poly(X)
```

```
5-element Vector{Float64}:
   1.0
 -29.000000000000007
  72.00000000000016
 -29.00000000000021
   1.000000000000037
```

特征多项式为

$$p(x) = x^4 - 29x^3 + 72x^2 - 29x + 1$$

帕斯卡矩阵拥有一个属性，即它的特征多项式系数的向量向前和向后读的数字相同（回文顺序）。

将矩阵 X 替换到特征方程 p 中。结果非常趋近于零矩阵。这个示例是 Cayley-Hamilton 定律的一种情况，其中矩阵满足其自身的特征方程。

```
Y = polyvalm(p,X)
```

```
Y = 4×4 Matrix{Float64}:
1.55209e-13   4.47642e-13   9.41469e-13   1.62004e-12
3.2685e-13    9.9365e-13    2.07478e-12   3.83693e-12
8.88178e-13   2.84217e-12   6.48614e-12   1.22071e-11
2.03926e-12   6.94556e-12   1.58913e-11   3.00548e-11
```

3.2.1.6.8 polyint

多项式积分

A. 语法

```
q = polyint(p,k)
q = polyint(p)
```

B. 说明

q = polyint(p,k) 使用积分常量 k 返回 p 中系数所表示的多项式积分。

q = polyint(p) 假定积分常量 k = 0。

C. 示例

(1). 对四次多项式求积分

计算定积分

$$I = \int_{-1}^{3} \left(3x^4 - 4x^2 + 10x - 25\right) dx$$

创建一个向量，用它来代表多项式，然后计算多项式的根。

创建一个向量来表示多项式被积函数 $3x^4 - 4x^2 + 10x - 25$。 x^3 项不存在，因此系数为 0。

```
p = [3 ,0 ,-4, 10 ,-25];
```

使用 polyint 和等于 0 的积分常量来对多项式求积分。

```
q = polyint(p)
```

```
6-element Vector{Float64}:
0.6
0.0
-1.3333333333333333
5.0
-25.0
0.0
```

通过在积分极限上计算 q，求解积分的值。

```
a = -1
b = 3
I = diff(polyval(q,[a,b]))
```

```
1-element Vector{Float64}:
49.06666666666665
```

(2). 对两个多项式的乘积求积分

计算

$$I = \int_{0}^{2} \left(x^5 - x^3 + 1\right)\left(x^2 + 1\right) dx$$

创建向量来表示多项式 $p(x) = x^5 - x^3 + 1$ 和 $v(x) = x^2 + 1$

```
p = [1,0 ,-1, 0, 0, 1]
```

```
v = [1 ,0, 1]
```

多项式相乘，并使用积分常量 k = 3 对所生成的表达式求积分。

```
k = 3
q = polyint(conv(p,v),k)
```

```
9-element Vector{Float64}:
0.125
0.0
0.0
0.0
-0.25
0.3333333333333333
0.0
1.0
3.0
```

通过在积分极限上计算 q 来求解 I 的值。

```
a = 0
b = 2
I = diff(polyval(q,[a,b]))
```

```
1-element Vector{Float64}:
32.66666666666667
```

3.2.1.6.9 polyder

多项式微分

A. 语法

```
k = polyder(p)
k = polyder(a,b)[3]
q,d,k = polyder(a,b)
```

B. 说明

k = polyder(p) 返回 p 中的系数表示的多项式的导数，

$$k(x) = \frac{d}{dx}p(x)$$

k = polyder(a,b) 返回多项式 a 和 b 的乘积的导数，

$$k(x) = \frac{d}{dx}[a(x)b(x)]$$

q,d = polyder(a,b)返回多项式 a 和 b 的商的导数，

$$\frac{q(x)}{d(x)} = \frac{d}{dx}\left[\frac{a(x)}{b(x)}\right]$$

C. 示例

(1). 求导多项式

创建一个向量来表示多项式

$p(x) = 3x^5 - 2x^3 + x + 5$

```
p = [3 ,0, -2 ,0, 1, 5]
```

使用 polyder 对多项式求导。结果为

$q(x) = 15x^4 - 6x^2 + 1$

```
q = polyder(p)
```

```
5-element Vector{Int64}:
15
0
-6
0
1
```

(2). 求导多项式乘积

创建两个向量来表示多项式 $a(x) = x^4 - 2x^3 + 11$ 和 $b(x) = x^2 - 10x + 15$

创建向量来表示多项式 $p(x) = x^5 - x^3 + 1$ 和 $v(x) = x^2 + 1$

```
a = [1, -2, 0, 0, 11]
b = [1, -10, 15]
```

使用 polyder 来计算 $q(x) = \frac{d}{dx}[a(x)b(x)]$

```
q = polyder(a,b)[3]
```

```
6-element Vector{Int64}:
6
-60
140
-90
22
-110
```

结果为：

$$q(x)=6x^5-60x^4+140x^3-90x^2+22x-110$$

(3). 求导多项式商

创建两个向量来表示商中的多项式，

$$\frac{x^4-3x^2-1}{x+4}$$

```
p = [1,0, -3, 0, -1]
v = [1 ,4]
```

使用包含两个输出参数的 polyder 来计算 $\frac{q(x)}{d(x)}=\frac{d}{dx}\left[\frac{p(x)}{v(x)}\right]$

```
q,d = polyder(p,v)
```

```
q =
5-element Vector{Float64}:
3.0
16.0
-3.0
-24.0
1.0
d =
3-element Vector{Float64}:
1.0
8.0
16.0
```

结果为：

$$\frac{q(x)}{d(x)}=\frac{3x^4+16x^3-3x^2-24x+1}{x^2+8x+16}$$

3.2.1.6.10 polyreduce

去除多项式前系数为 0 的无用项

A. 语法

```
q = polyreduce(p::Vector)
```

B. 说明

q = polyreduce(p) 去除 p 第一个非 0 项参数前所有系数项，对向量进行化

简，不改变多项式。

C. 示例

(1). 简化多项式

创建一个向量来表示多项式

$$p(x) = 3x^5 - 2x^3 + x + 5$$

```
p = [0,0,0,3 ,0, -2 ,0, 1, 5]
```

```
p = 9-element Vector{Int64}:
0
0
0
3
0
-2
0
1
5
```

使用 polyreduce 对向量进行化简

```
q = polyreduce(p)
```

```
6-element Vector{Int64}:
3
0
-2
0
1
5
```

不改变其原有属性，化简多项式存储方式

3.2.1.6.11 polyfit polyfits2 polyfits3

多项式曲线拟合

A. 语法

```
p=polyfit(x,y,n)
p,S=polyfits2(x,y,n)
p,S,mu=polyfits3(x,y,n)
```

B. 说明

p=polyfit(x,y,n) 返回次数为 n 的多项式 p(x)的系数，该阶数是 y 中数据的最佳拟合（在最小二乘方式中）。p 的系数按降幂排列， p 的长度为 n+1 。

$$p(x) = p_1 x^n + p_2 x^{n-1} + ... + p_n x + p_n + 1$$

p,S=polyfits2(x,y,n)还返回一个结构体 S，后者可用作 polyval 的输入来获取误差估计值。

p,S,mu=polyfits3(x,y,n)还返回 mu,后者是一个二元素向量，包含中心化值和缩放值。mu(1)是 mean(x)，mu(2) 为 std(x)。使用这些值时，polyfit 将 x 的中心置于零值处并缩放为具有单位标准差。

$$\hat{x} = \frac{x - \bar{x}}{\sigma_x}$$

这种中心化和缩放变换可同时改善多项式和拟合算法的数值属性。

C. 示例

(1). 将多项式与三角函数拟合

在区间[0,4*pi]中沿正弦曲线生成 10 个等间距的点。

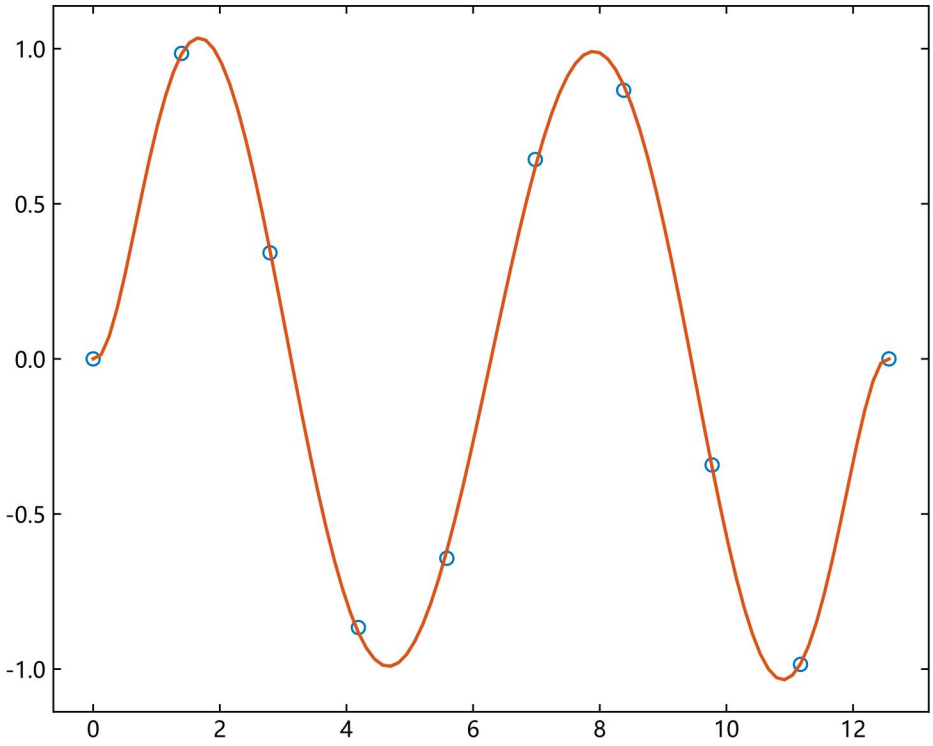
```
x = LinRange(0,4*pi,10)
y = sin.(x)
```

使用 polyfit 将一个 7 次多项式与这些点拟合。

```
p = vec(polyfit(x,y,7))
```

在更精细的网格上计算多项式并绘制结果图。

```
x1 = LinRange(0,4*pi,100)
y1 =polyval(p,x1)
figure(1)
plot(x,y,"o")
hold("on")
plot(x1,y1)
hold("off")
```



(2). 将多项式与点集拟合

创建一个由区间[0,1]中的 5 个等间距点组成的向量，并计算这些点处的

```
y(x)=(1+x)^-1。
x = LinRange(0,1,5)
y = 1 ./(1 .+x)
```

将 4 次多项式与 5 个点拟合。通常，对于 n 个点，可以拟合 n-1 次多项式以便完全通过这些点。

```
p= vec(polyfit(x,y,4))
```

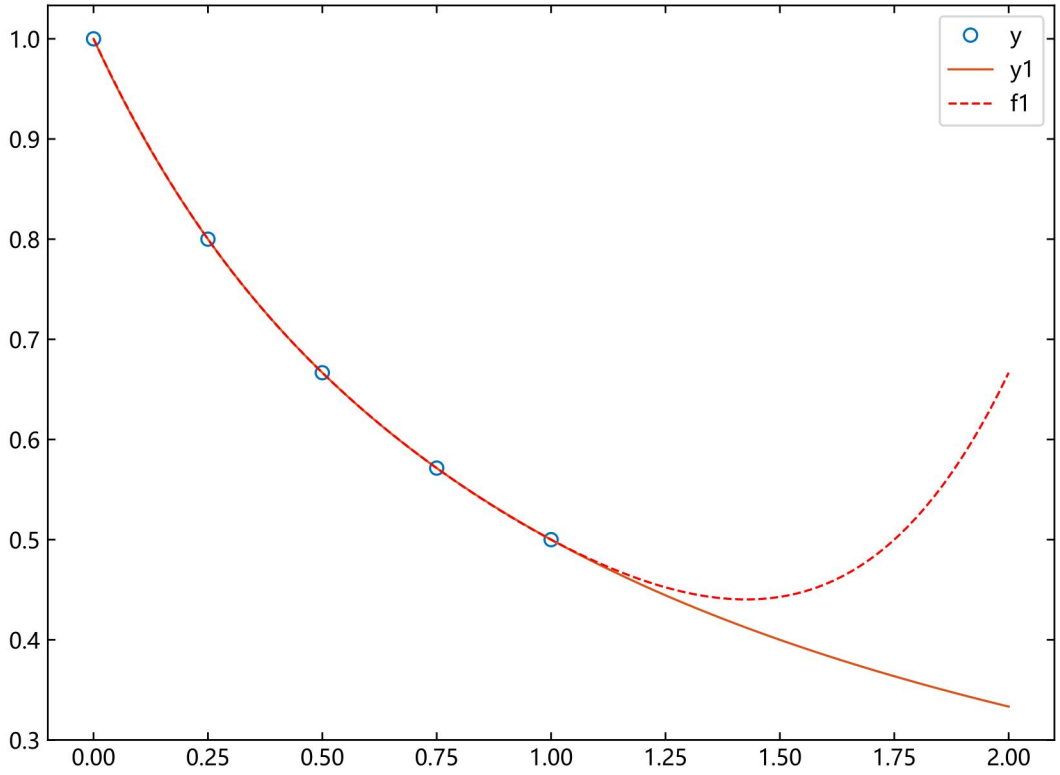
在由 0 和 2 之间的点组成的更精细网格计算原始函数和多项式拟合。

```
x1 = LinRange(0,2,100)
y1=1 ./(1 .+x1)
f1 = polyval(p,x1)
```

在更大区间[0,2]中绘制函数值和多项式拟合，其中包含用于获取以圆形突出显示的多项式拟合的点。多项式拟合在原始[0,1]区间中的效果较好，但在该区间外部很快与拟合函数出现差异。

```
figure(1)
```

```
plot(x,y,"o")
hold("on")
plot(x1,y1)
plot(x1,f1,"r--")
legend(["y","y1","f1"])
hold("off")
```



(3). 对误差函数进行多项式拟合

首先生成 x 点的向量，在区间[0,2.5]内等间距分布，然后计算这些点处的 erf(x)。

```
x = (0:0.1:2.5)
y = erf.(x)
```

确定 6 次逼近多项式的系数。

```
p = vec(polyfit(x,y,6))
```

```
1×7 adjoint(::Vector{Float64}) with eltype Float64:
0.00841937   -0.0982996   0.421736   -0.743463   0.147104   1.10645
0.000441174
```

为了查看拟合情况如何，在各数据点处计算多项式，并生成说明数据、拟合和误差的一个表。

```
f=polyval(p,x)
T=DataFrame(X=x,Y=y,Fit=f,FitError=y-f)
```

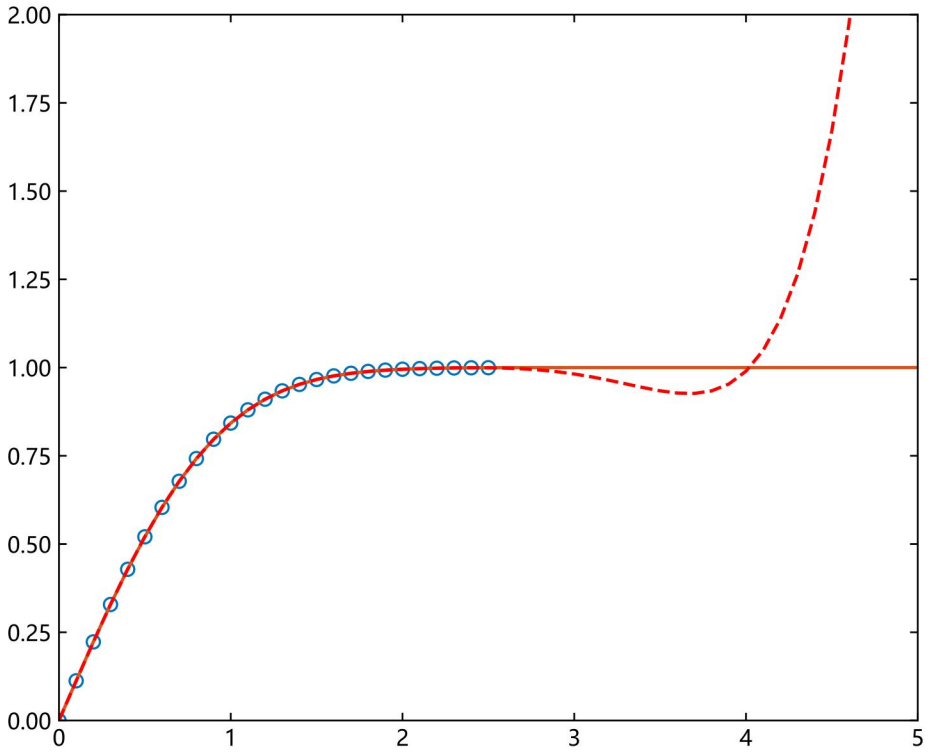
```
26×4 DataFrame

Row │    X       Y           Fit         FitError
    │  Float64   Float64     Float64      Float64
────┼──────────────────────────────────────────────
  1 │   0.0      0.0       0.000441174  -0.000441174
  2 │   0.1      0.112463   0.111855      0.000608361
  3 │   0.2      0.222703   0.222311      0.000391886
  4 │   0.3      0.328627   0.328724     -9.74288e-5
  5 │   0.4      0.428392   0.428799     -0.000406608
  6 │   0.5      0.5205     0.520926     -0.000425678
  7 │   0.6      0.603856   0.604084     -0.000228242
  8 │   0.7      0.677801   0.677755      4.63827e-5
  9 │   0.8      0.742101   0.741831      0.000269919
 10 │   0.9      0.796908   0.796543      0.000365145
 11 │   1.0      0.842701   0.842384      0.000316401
 12 │   1.1      0.880205   0.880046      0.000159482
 13 │   1.2      0.910314   0.910354     -3.99194e-5
 14 │   1.3      0.934008   0.934219     -0.000210995
 15 │   1.4      0.952285   0.952584     -0.000299334
 16 │   1.5      0.966105   0.966386     -0.000280972
 17 │   1.6      0.976348   0.976515     -0.000167043
 18 │   1.7      0.98379    0.98379       8.33065e-7
 19 │   1.8      0.989091   0.988928      0.000162785
 20 │   1.9      0.99279    0.992533      0.000257909
 21 │   2.0      0.995322   0.995079      0.000243468
 22 │   2.1      0.997021   0.996907      0.000113098
 23 │   2.2      0.998137   0.998226     -8.85483e-5
 24 │   2.3      0.998857   0.999114     -0.000256727
 25 │   2.4      0.999311   0.999536     -0.000224508
 26 │   2.5      0.999593   0.999362      0.000231508
```

在该区间中，插值与实际值非常符合。创建一个绘图，以显示在该区间外，外插值与实际数值如何快

```
x1 = (0:0.1:5)
y1 = erf.(x1)
f1=polyval(p,x1)
figure(1)
plot(x,y,"o")
hold("on")
plot(x1,y1,"-")
plot(x1,f1,"r--")
xlim([0, 5])
```

```
ylim([0, 2])
hold("off")
```



(4). 使用中心化和缩放改善数值属性

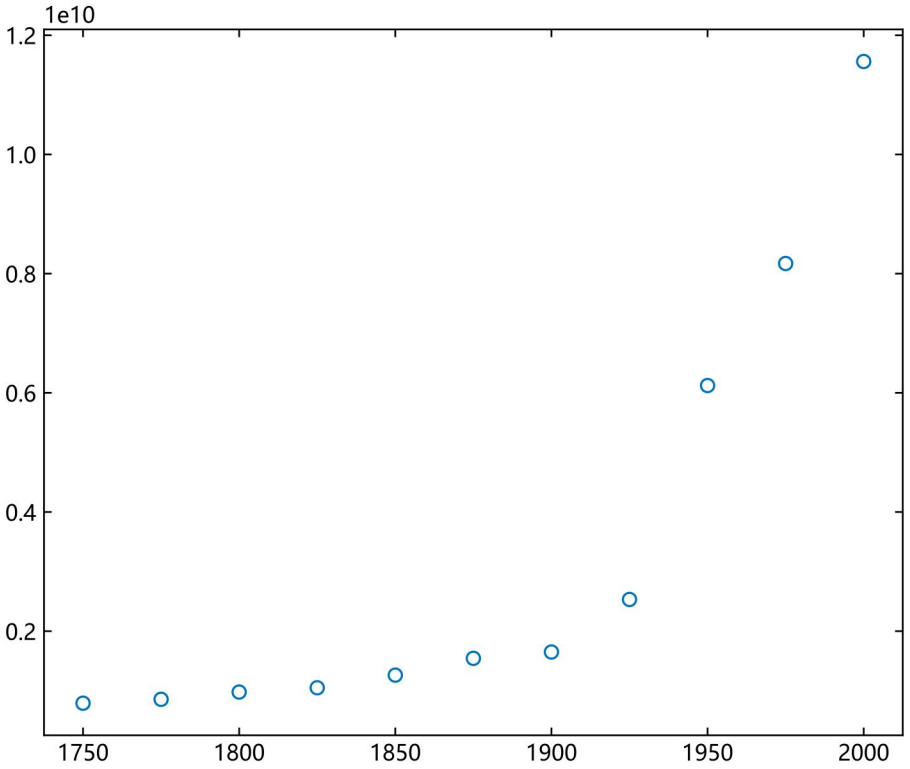
创建一个由 1750 - 2000 年的人口数据组成的表，并绘制数据点。

```
year = 1750:25:2000
pop = 1e6*[791,856,978,1050,1262,1544,1650,2532,6122,8170,11560]
T = DataFrame(year = year,pop = pop)
```

```
T = 11×2 DataFrame
Row | year   pop
    | Int64  Float64
────────────────────
1   |  1750  7.91e8
2   |  1775  8.56e8
3   |  1800  9.78e8
4   |  1825  1.05e9
5   |  1850  1.262e9
6   |  1875  1.544e9
7   |  1900  1.65e9
```

```
8   |  1925  2.532e9
9   |  1950  6.122e9
10  |  1975  8.17e9
11  |  2000  1.156e10
```

```
plot(year,pop,"o")
```

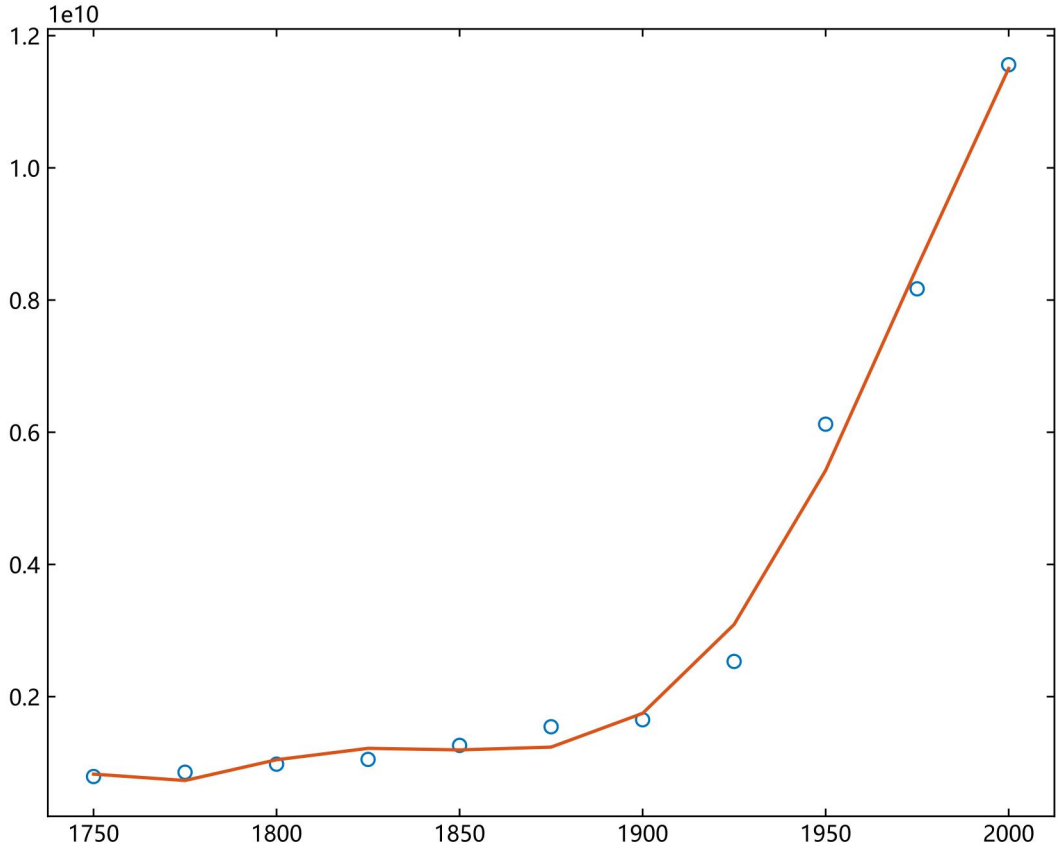


使用带三个输入的 polyfits3 拟合一个使用中心化和缩放的 5 次多项式，这将改善问题的数值属性。polyfits3 将 year 中的数据以 0 为进行中心化，并缩放为具有标准差 1，这可避免在拟合计算中出现病态的 Vandermonde 矩阵。

```
p,sm_1,mu = polyfits3(T.year, T.pop, 5)
```

使用带有四个输入的 polyval 来计算 p 与缩放的年份，(year-mu(1))/mu(2)。 根据原始年份绘制结果。

```
f = polyval(vec(p),year,[],mu)
hold("on")
plot(year,f)
hold("off")
```



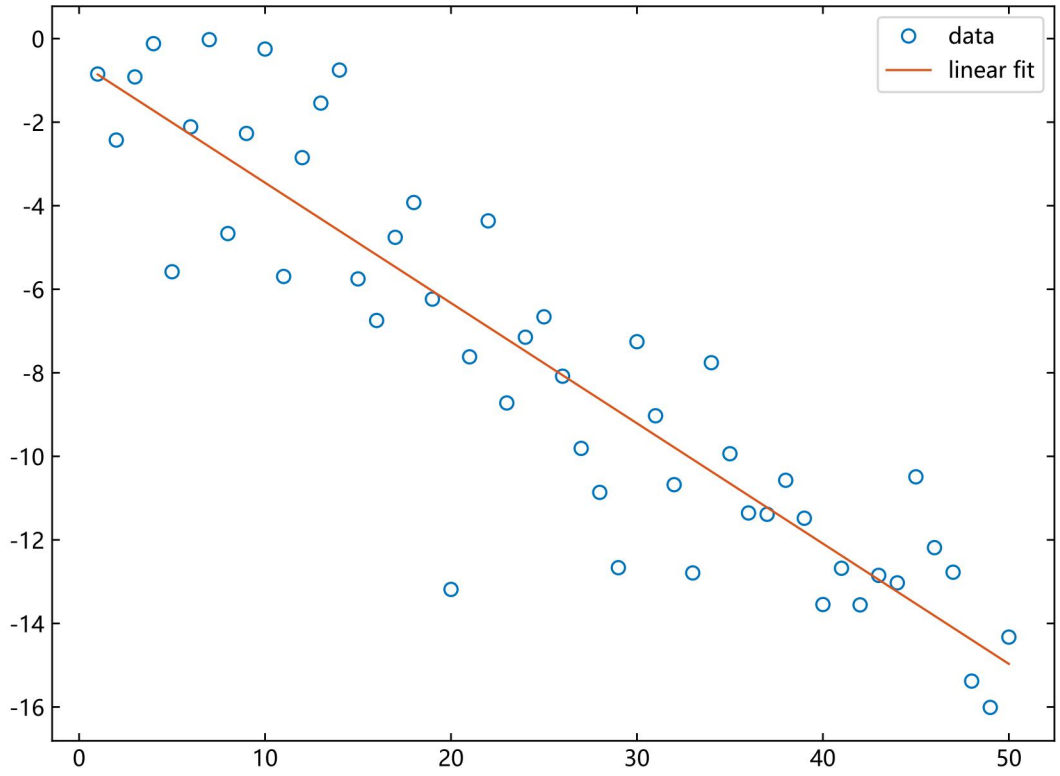
(5). 简单线性回归

将一个简单线性回归模型与一组离散二维数据点拟合。创建几个由样本数据点(x,y)组成的向量，对数据进行一次多项式拟合。

```
x = 1:50
y=vec(-0.3*x+2*randn(50,1))
p=vec(polyfit(x,y,1))
```

计算在 x 中的点处拟合多项式 p,用这些数据绘制得到的线性回归模型。

```
f = polyval(p,x)
plot(x,y,"o",x,f,"-")
legend(["data","linear fit"])
```



(6). 误差估计的线性回归

将线性模型拟合到一组数据点并绘制结果，包括 95% 预测区间的估计值。

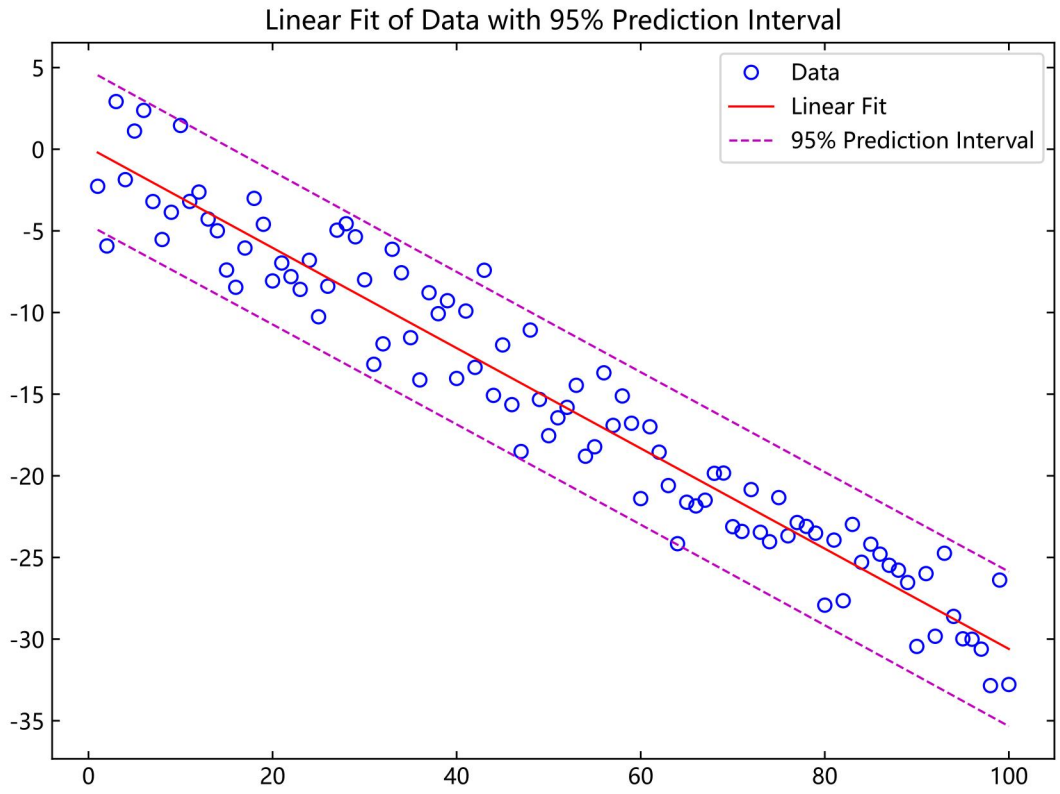
```
x = 1:100
y = vec(-0.3*x+2*randn(1,100)')
p,S = polyfits2(x,y,1)
```

在 x 中的点处评估 p 中的一次多项式拟合。 将误差估计结构指定为第三个输入，以便 polyval 计算标准误差的估计值。 标准误差估计值以 delta 形式返回。

```
y_fit,delta = polyval(vec(p),x,S)
```

绘制图像

```
plot(x,y,"bo")
hold("on")
plot(x,y_fit,"r-")
plot(x,y_fit .+ 2*delta,"m--",x,y_fit .- 2*delta,"m--")
title("Linear Fit of Data with 95% Prediction Interval")
legend(["Data","Linear Fit","95% Prediction Interval"])
```



3.2.1.7 特殊函数

Bessel、Legendre、椭圆、误差、gamma 和其他函数。

特殊函数是一组在实际应用中经常出现的著名数学函数。

3.2.1.7.1 Bessel 函数

函数名	简介
airy	Airy 函数
airyai	第一类 airy 函数
airybi	第二类 airy 函数
besselh	第三类 Bessel 函数（Hankel 函数）
besseli	第一类修正 Bessel 函数
besselj	第一类 Bessel 函数
besselk	第二类修正 Bessel 函数
bessely	第二类 Bessel 函数

airy

Airy 函数

A. 语法

```
W = airy(Z)
W = airy(k,Z)
W = airy(k,Z,scale)
```

B. 说明

W = airy(Z) 为 Z 的每个元素返回 Airy 函数 Ai(Z)。示例

W = airy(k,Z) 返回任何四个不同的 Airy 函数，具体取决于 k 的值，例如第二类 Airy 函数或 Airy 函数的一阶导数。示例

W = airy(k,Z,scale) 缩放生成的 Airy 函数。airy 根据您选择的 k 和 scale 对 W 应用特定的缩放函数。示例。

C. 示例

(1). 实数值 x 的 Airy 函数

定义 x。

```
x = -10:0.01:1
```

计算 Ai(x)

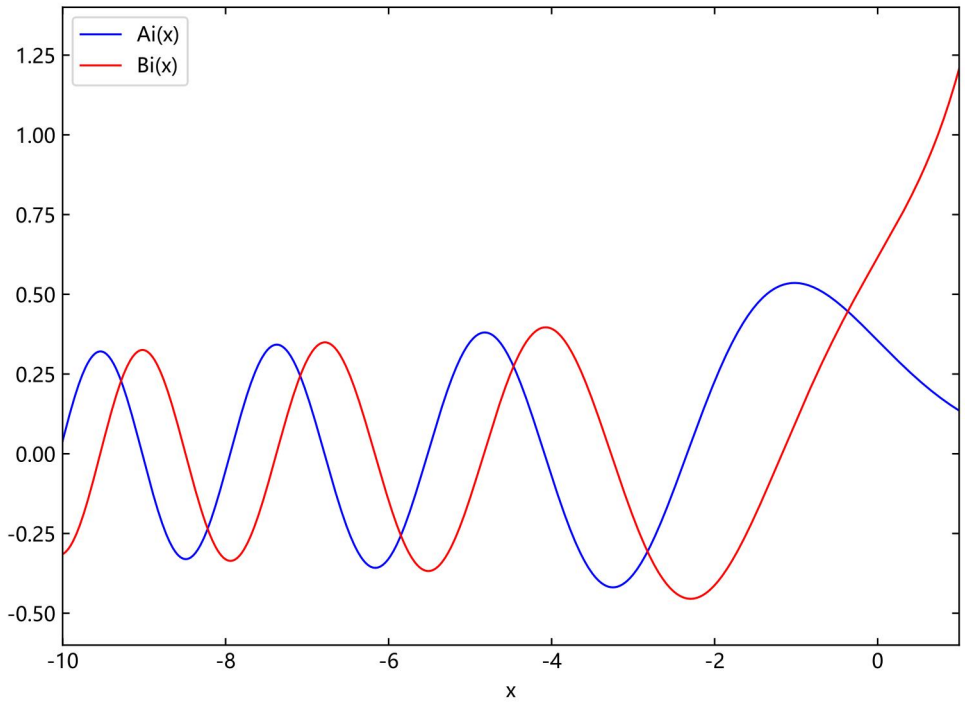
```
x = -10:0.01:1
```

使用 k=2 计算 Bi(x)。

```
bi = airy.(2,x);
```

在同一坐标区上绘制上面两个结果。

```
figure()
plot(x,ai,"-b",x,bi,"-r")
axis([-10 1 -0.6 1.4])
xlabel("x")
legend(["Ai(x)","Bi(x)"],loc = "northwest")
```



(2). 复数值 x 的 Airy 函数

计算切片（穿过位于 x+im 的复平面）的 Airy 函数。

获取穿过复平面的切片。

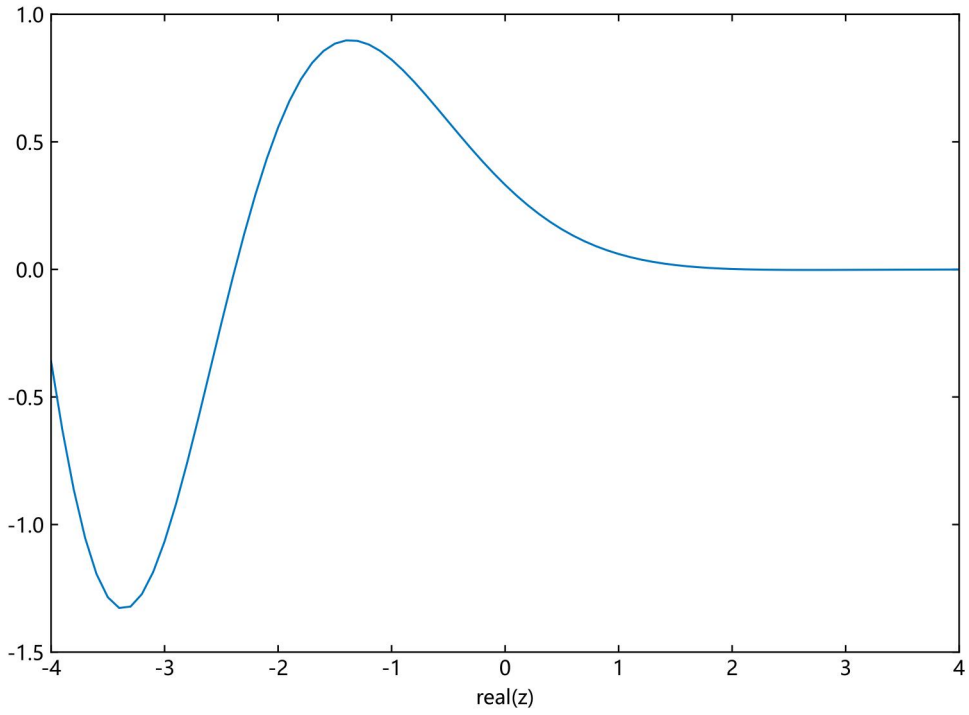
```
x = -4:0.1:4;
z = x .+1im;
```

计算 Ai(x)

```
w = airy.(z);
```

绘制结果的实数部分。

```
figure()
plot(x, real(w))
axis([-4 4 -1.5 1])
xlabel("real(z)")
```



(3). 已缩放的 Airy 函数

定义 x。

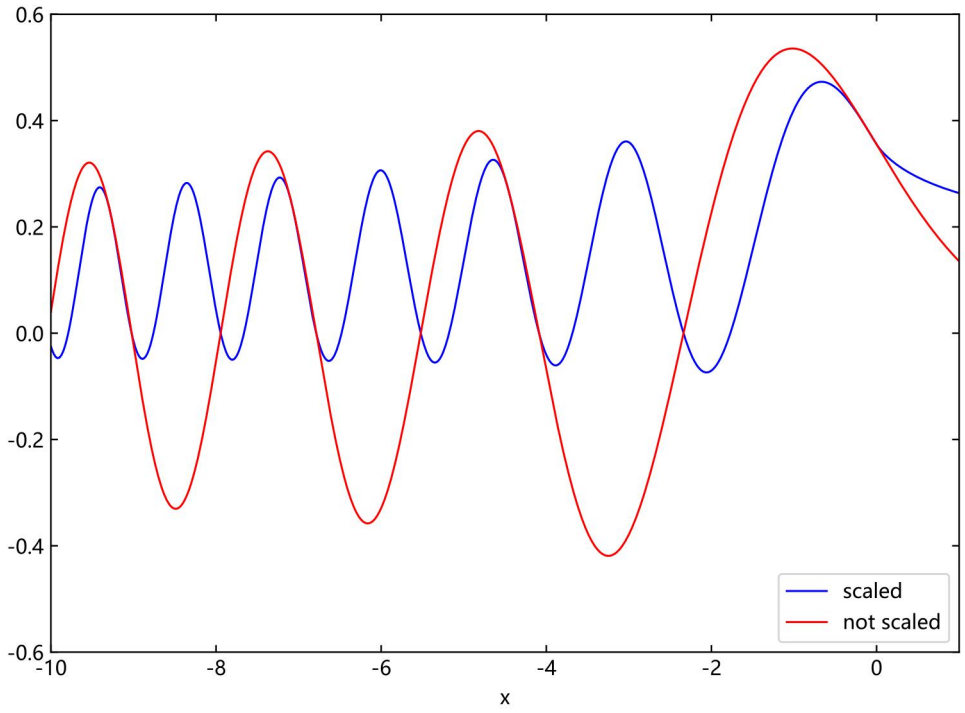
```
x = -10:0.01:1
```

计算已缩放和未缩放的 Airy 函数

```
scaledAi = airy.(0,x,1);
noscaleAi =  airy.(0,x,0);
```

绘制每个结果的实数部分。

```
rscaled = real(scaledAi);
rnoscale = real(noscaleAi);
figure()
plot(x,rscaled,"-b",x,rnoscale,"-r")
axis([-10 1 -0.60 0.60])
xlabel("x"scaled","not scaled"],loc = "southeast")
```



airyai

第一类 airy 函数

D. 语法

```
W = airyai(Z)
```

E. 说明

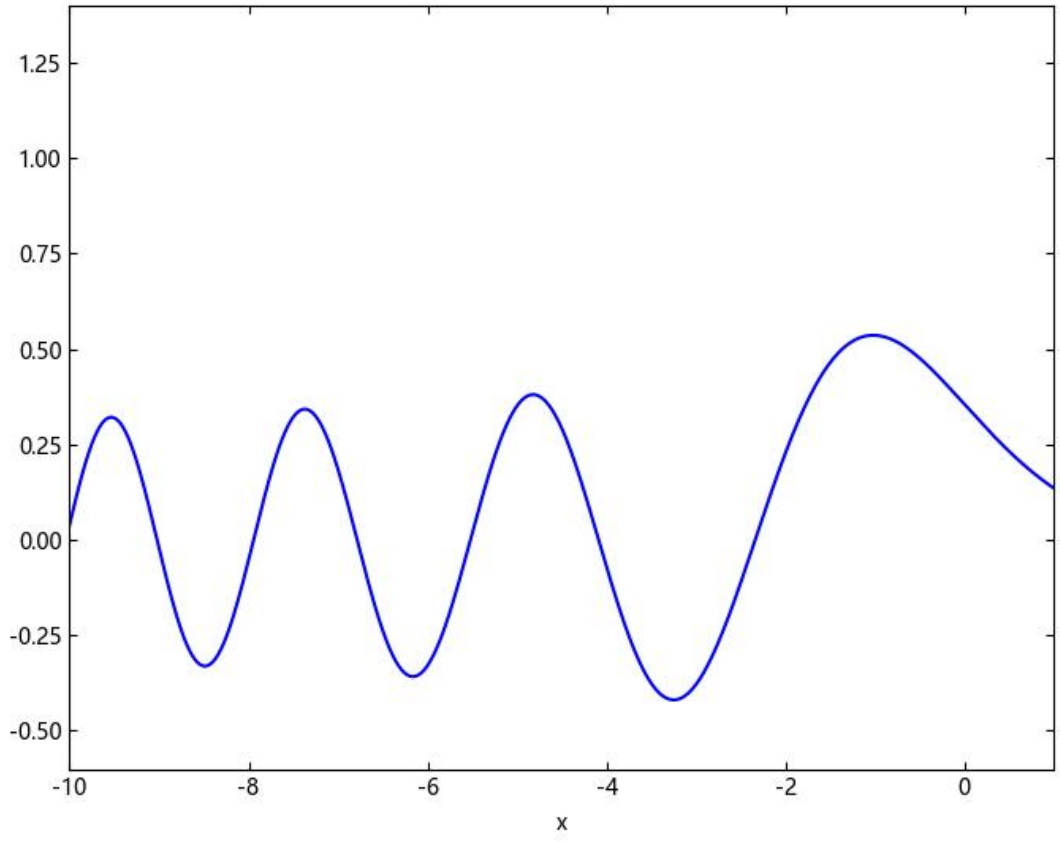
W = airyai(Z)返回第一类 airy 函数。

F. 示例

(1). 实数值 x 的第一类 Airy 函数

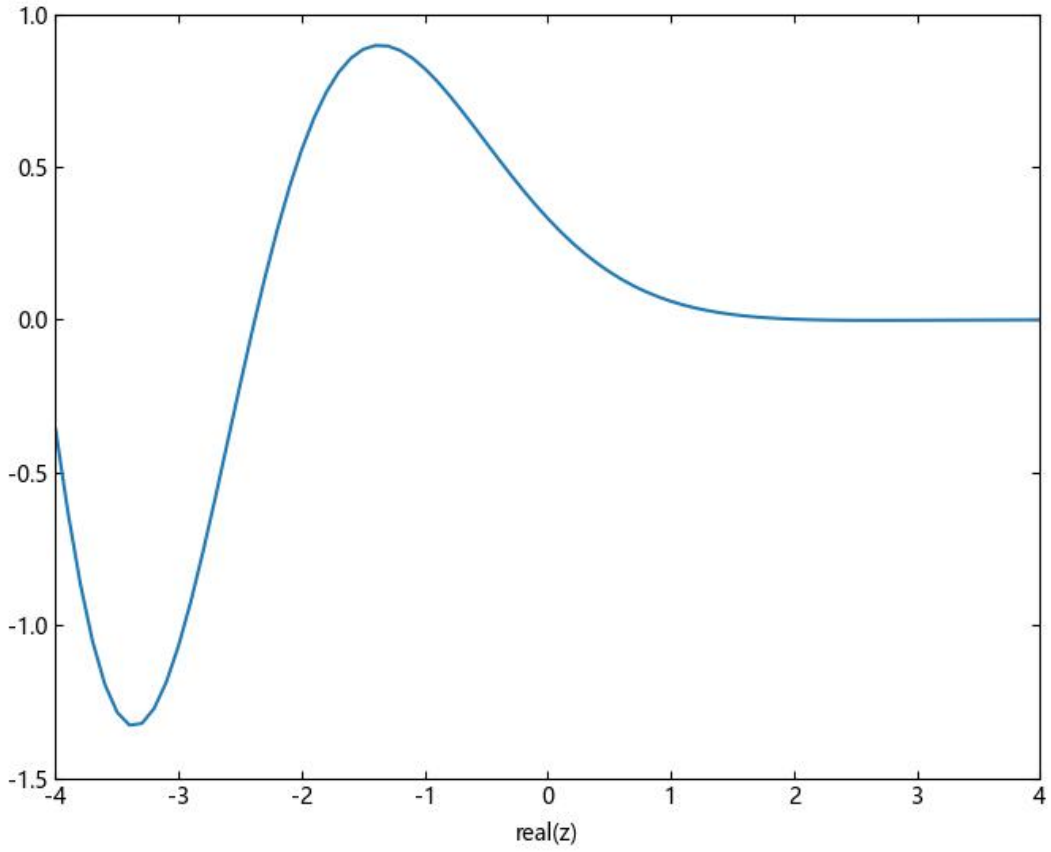
定义 x。

```
x = -10:0.01:1
y = airyai.(x)
plot(x,y,"-b")
 axis([-10 1 -0.6 1.4])
xlabel("x")
```



(2). 复数值 x 的第一类 Airy 函数

```
x = -4:0.1:4
z = x .+ 1im
w = airyai.(z)
plot(x,real.(w))
axis([-4 4 -1.5 1])
xlabel("real(z)")
```



airybi

第二类 airy 函数

A. 语法

```
W = airybi(Z)
```

B. 说明

W = airybi(Z)返回第二类 airy 函数。

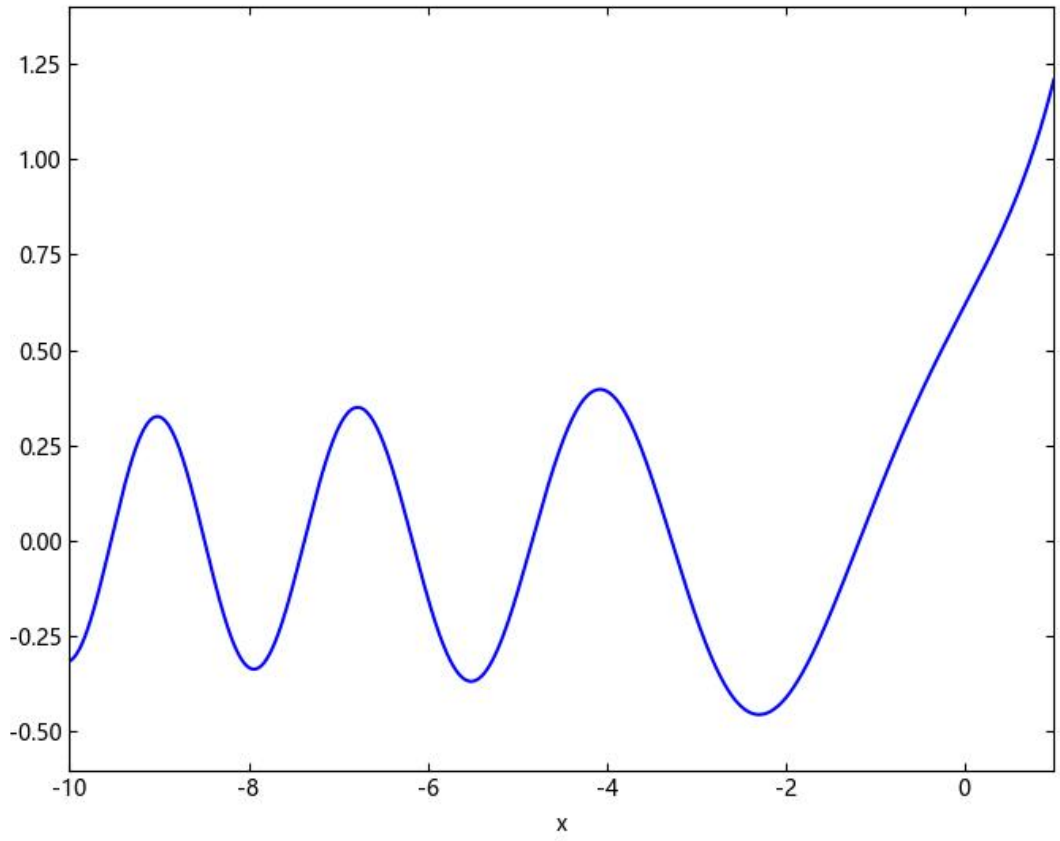
C. 示例

(1). 实数值 x 的第二类 Airy 函数

定义 x。

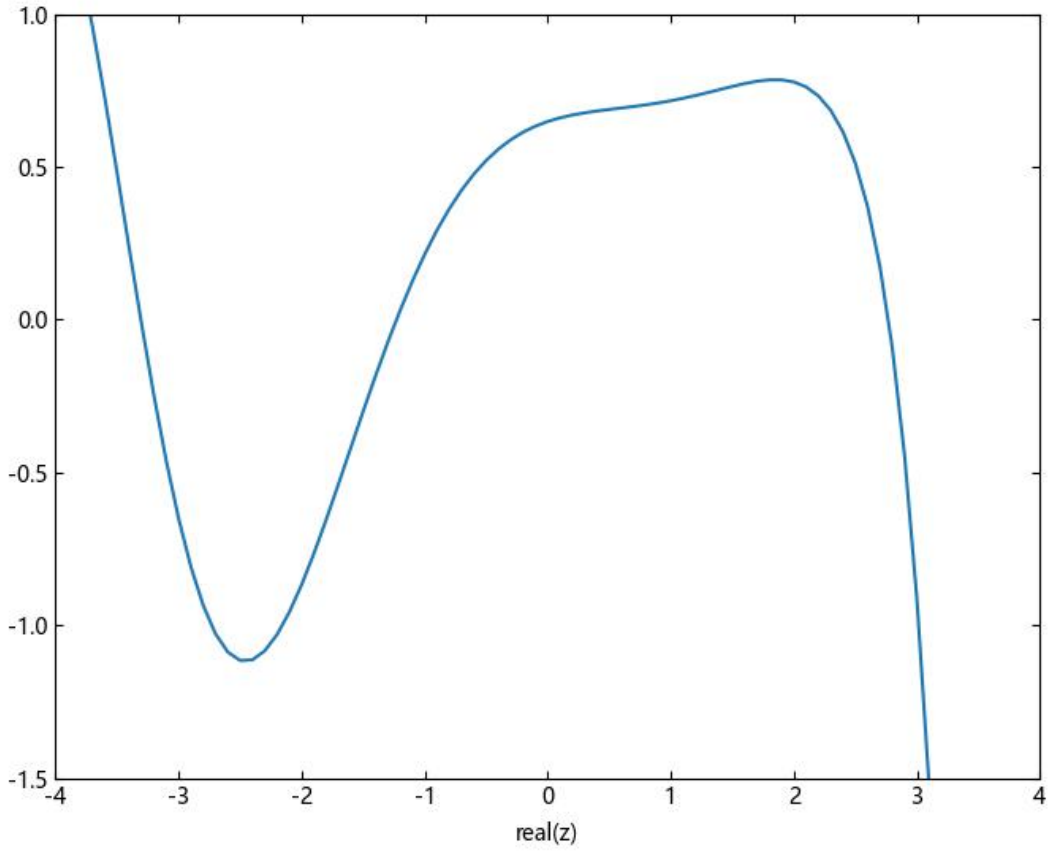
```
x = -10:0.01:1
y = airybi.(x)
plot(x,y,"-b")
axis([-10 1 -0.6 1.4])
```

```
xlabel("x")
```



(2). 复数值 x 的第二类 Airy 函数

```
x = -4:0.1:4
z = x .+ 1im
w = airybi.(z)
plot(x,real.(w))
axis([-4 4 -1.5 1])
xlabel("real(z)")
```



besselh

第三类 Bessel 函数（Hankel 函数）

A. 语法

```
H = besselh(nu,z)
H = besselh(nu,k,z)
```

B. 说明

H = besselh(nu,z)为 z 计算第一类 Hankel 函数 $H_v^{(1)}(z)=J_v(z)+iY_v(z)$。

H = besselh(nu,k,z)为 z 计算第一类或第二类 Hankel 函数 $H_v^K(z)$,其中 k 为 1 或 2。

C. 示例

(1). Hankel 函数的模数和相位

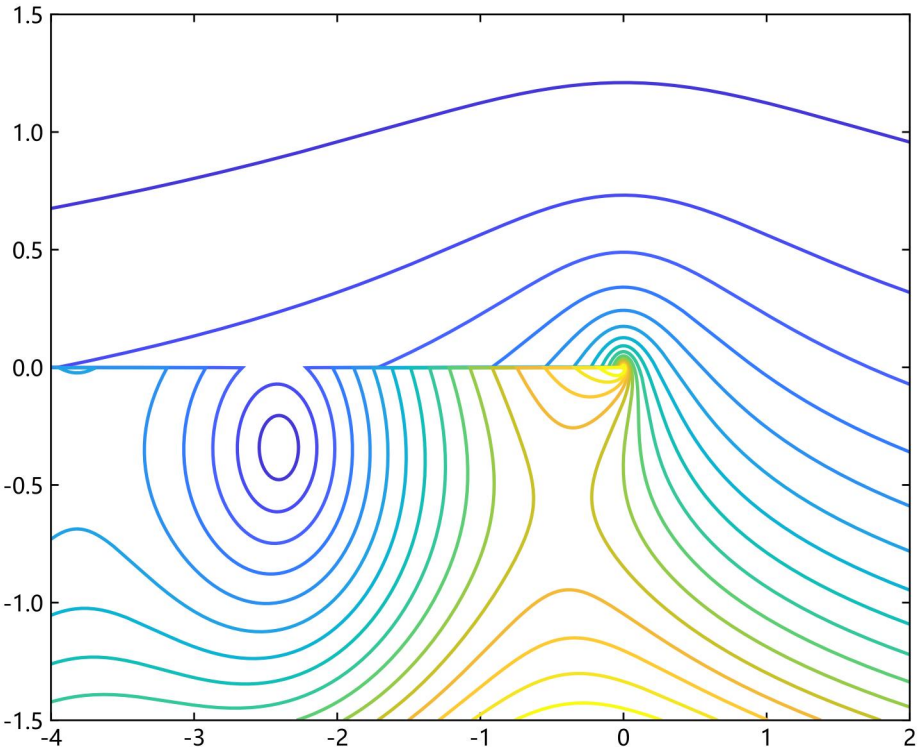
生成 Hankel 函数 $H_0^{(1)}[1]$ 的模和相位的等高线图。

创建一个由域值构成的网格。

```
X,Y = meshgrid2(-4:0.002:2,-1.5:0.002:1.5)
```

在此域中计算 Hankel 函数并生成模量等高线图。

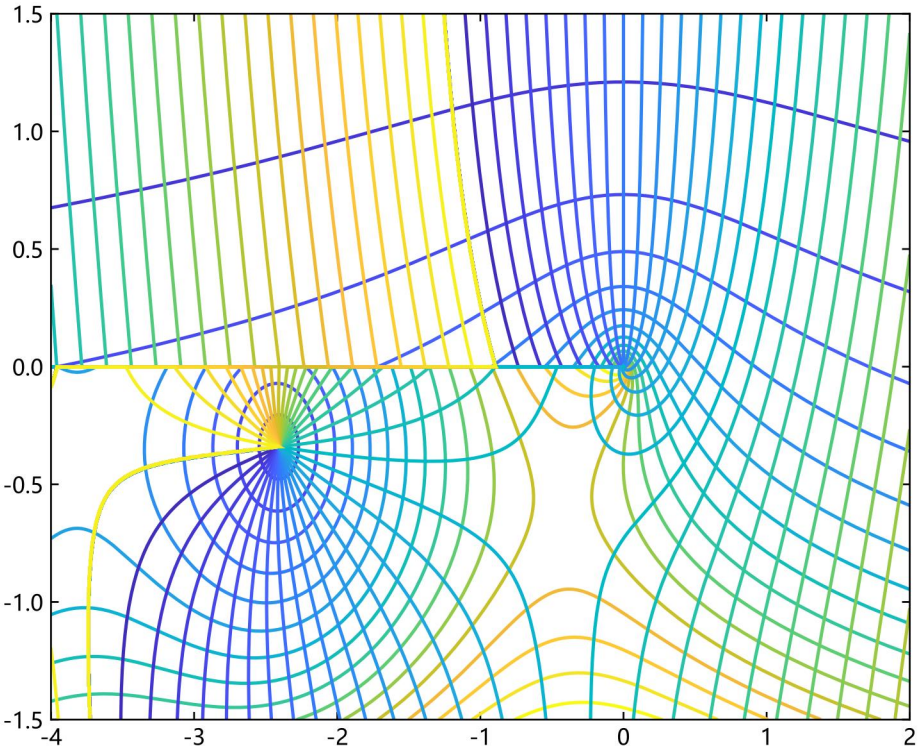
```
Z = X+im*Y
H = zero(Z)
for i in 1:length(Z)
if abs(Z[i]) == 0
H[i] = 0
else
H[i] = besselh(0,Z[i])
end
end
contour(X,Y,abs.(H),levels = 0:0.2:3.2)
hold("on")
```



在同一图窗中，添加该相位的等高线图。

```
contour(X,Y,rad2deg.(angle.(H)),levels = -180:10:180)
```

```
hold("off")
```



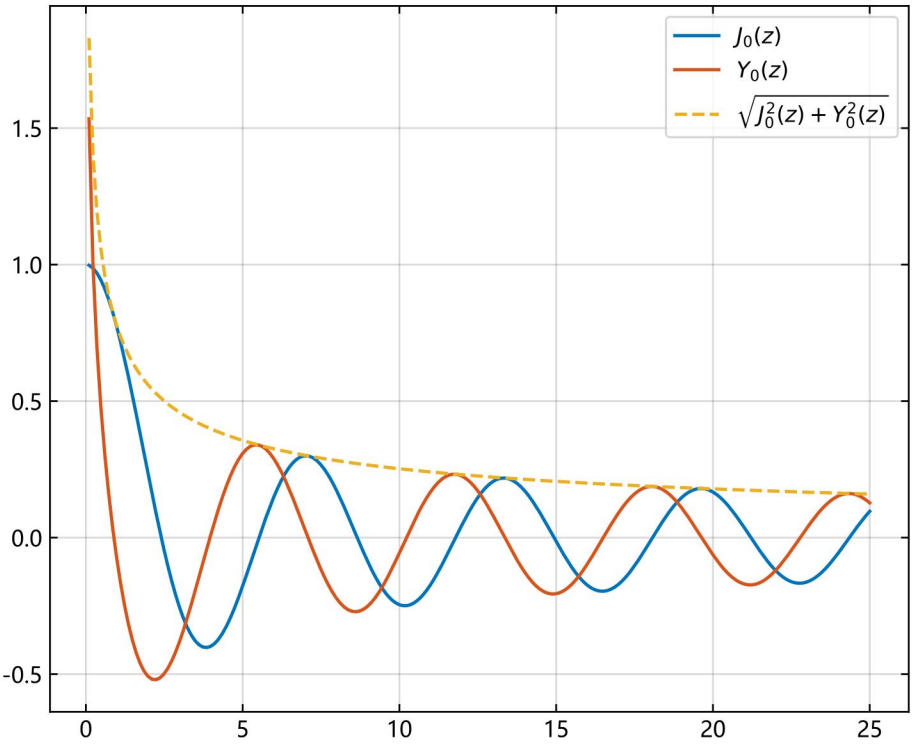
(2). 渐进行为

绘制第二类 Hankel 函数的实部和虚部，并检查其渐近行为。 计算区间 $[0.1, 25]$ 中的第二类 Hankel 函数 $H_0^2(z) = J_0(z) - iY_0(z)$

```
k = 2
nu = 0
z = LinRange(0.1,25,200)
H = besselh.(nu,k,z)
```

绘制函数的实部和虚部。在同一图窗中，绘制线性组合 这将显示实部和虚 部的幅值的渐近行为。

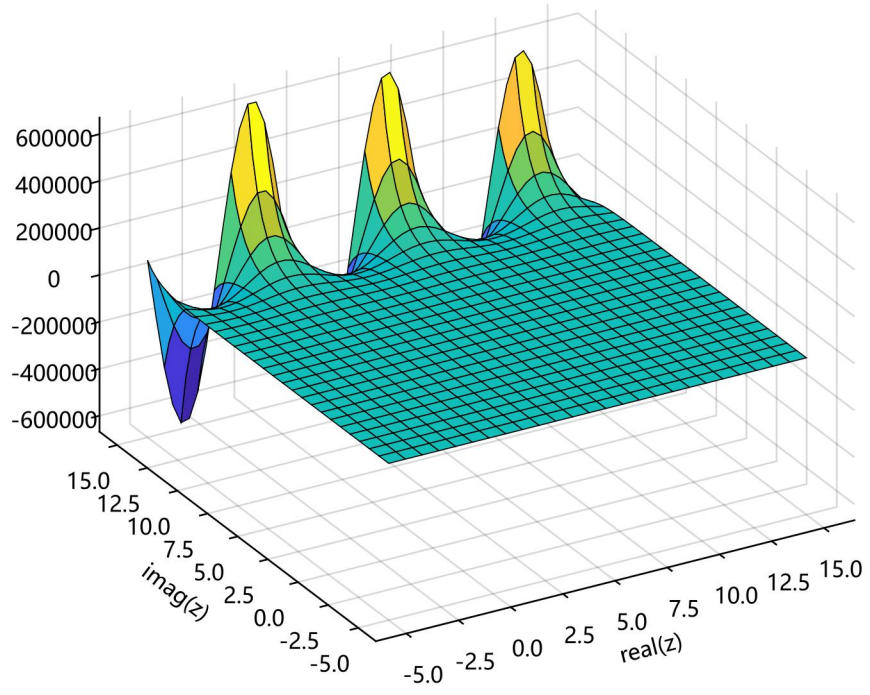
```
plot(z,real.(H),z,imag.(H))
grid("on")
hold("on")
M = sqrt.(real.(H).^2+imag.(H).^2)
plot(z,M,"--")
legend([raw"$J_0(z)$", raw"$Y_0(z)$", raw"$\sqrt{J_0^2 (z) + Y_0^2
(z)}$","interpreter","latex"])
```



(3). 未指数缩放的 Hankel 函数

计算复平面上的二阶未缩放 Hankel 函数。当 z 的正虚部较大时，函数的值很快就会发散。这种现象限制了可计算值的范围。

```
k = 2
nu = 1
x = [-5:0.4:15...]'
y = x'
z = x .+ im*y
scaled = 1
H = besselh.(nu,k,z)
X,Y = meshgrid2(vec(collect(x)),y)
surf(X,Y,imag.(H))
xlabel("real(z)")
ylabel("imag(z)")
```



besseli

第一类修正 Bessel 函数

A. 语法

```
I = besseli(nu,z)
```

B. 说明

I = besseli(nu,z)为 z 计算第一类修正 Hankel 函数 $H_v^{(1)}(z) = J_v(z) + iY_v(z)$

C. 示例

(1). 绘制第一类修正 Bessel 函数定义域

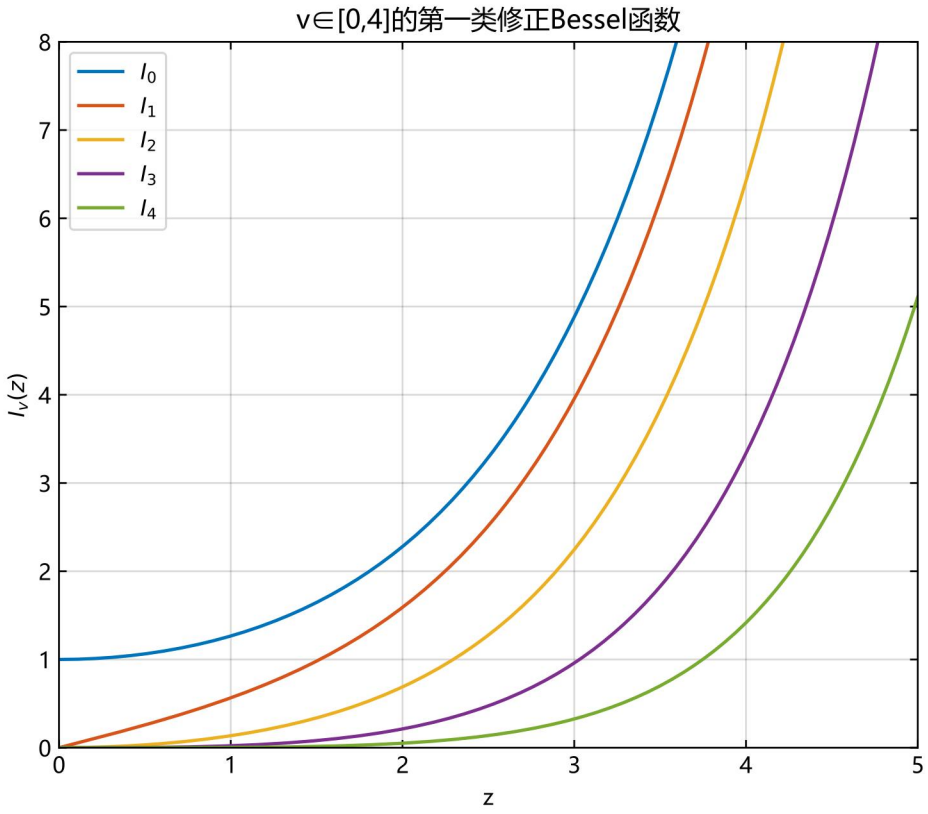
```
z = 0:0.01:5
```

计算前五个第一类修正 Bessel 函数。I 的每一行包含在 z 中的点上计算的某阶函数的值。

```
I = zeros(5,501)
for nu = 0:4
I[nu+1,:] = besseli.(nu,z)
end
```

在同一图窗中绘制所有函数。

```
hold("on")
for i = 1:5
plot(z,I[i,:])
end
axis([0 5 0 8])
grid("on")
legend([raw"$I_0$",raw"$I_1$",raw"$I_2$",raw"$I_3$",raw"$I_4$"])
title("v∈[0,4]的第一类修正 Bessel 函数")
xlabel("z") ylabel(raw"$I_v(z)$")
语法 hold("off")
```



besselj

第一类修正 Bessel 函数

A. 语法

```
I = besselj(nu,z)
```

B. 说明

I = besselj(nu,z)为 z 计算第一类 Hankel 函数 $J_v(z)$。

C. 示例

(1). 绘制第一类 Bessel 函数图定义域

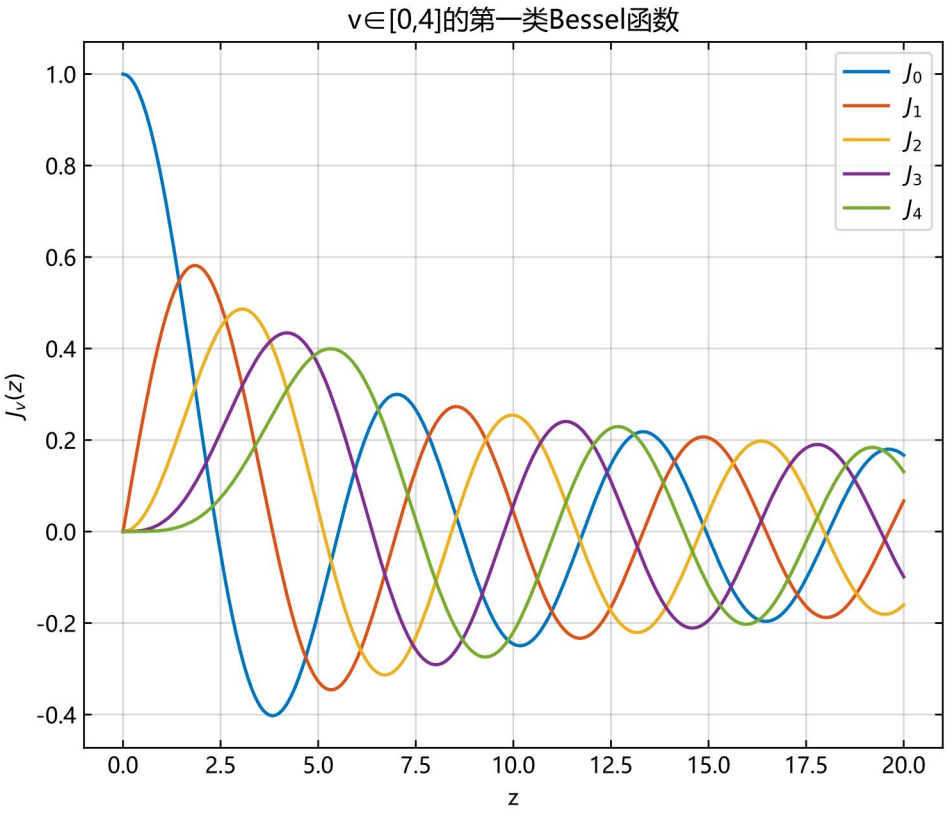
```
z = 0:0.1:20
```

计算前五个第一类 Bessel 函数。I 的每一行包含在 z 中的点上计算的某阶函数的值。

```
J = zeros(5,201)
for i = 0:4
J[i+1,:] = besselj.(i,z)
end
```

在同一图窗中绘制所有函数。

```
hold("on")
for i = 1:5
plot(z,J[i,:])
end
grid("on")
legend([raw"$J_0$",raw"$J_1$",raw"$J_2$",raw"$J_3$",raw"$J_4$"])
title("v∈[0,4]的第一类 Bessel 函数")
xlabel("z")
ylabel(raw"$J_v(z)$")
hold("off")
```



besselk

第二类修正 Bessel 函数

A. 语法

```
K = besselk(nu,z)
```

B. 说明

K = besselk(nu,z)为 z 计算第二类修正 Hankel 函数 $K_v(z)$。

C. 示例

(1). 绘制第二类修正 Bessel 函数

定义域

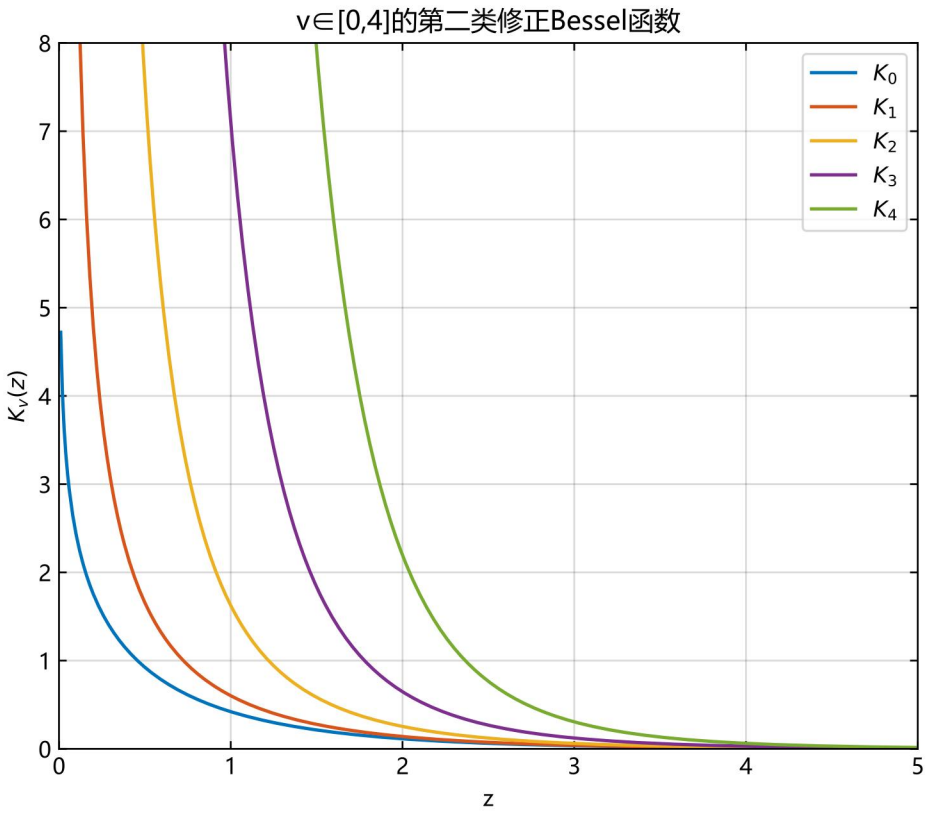
```
z = 0:0.01:5
```

计算前五个第二类修正 Bessel 函数。I 的每一行包含在 z 中的点上计算的某阶函数的值。

```
K = zeros(5,501)
for i = 0:4
K[i+1,:] = besselk.(i,z)
end
```

在同一图窗中绘制所有函数。

```
hold("on")
for i = 1:5
plot(z,K[i,:])
end
axis([0 5 0 8])
grid("on")
legend([raw"$K_0$",raw"$K_1$",raw"$K_2$",raw"$K_3$",raw"$K_4$"])
title("v∈[0,4]的第二类修正 Bessel 函数")
xlabel("z")
ylabel(raw"$K_v(z)$")
hold("off")
```



bessely

第二类 Bessel 函数

A. 语法

```
Y = bessely(nu,z)
```

B. 说明

Y = bessely(nu,z)为 z 计算第二类 Hankel 函数 $Y_v(z)$ 。

C. 示例

(1). 绘制第二类 Bessel 函数图

定义域

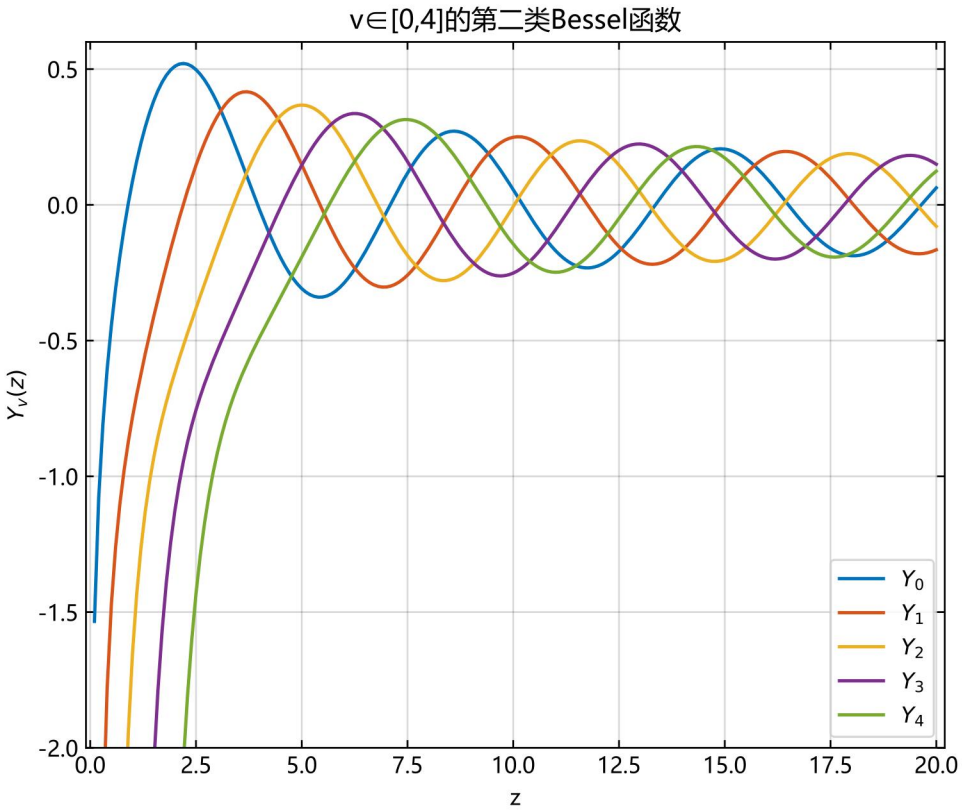
```
z = 0:0.1:20
```

计算前五个第二类 Bessel 函数。I 的每一行包含在 z 中的点上计算的某阶函数的值。

```
Y = zeros(5,201)
for i = 0:4
Y[i+1,:] = bessely.(i,z)
end
```

在同一图窗中绘制所有函数。

```
hold("on")
for i = 1:5
plot(z,Y[i,:])
end
axis([-0.1 20.2 -2 0.6])
grid("on")
legend([raw"$Y_0$",raw"$Y_1$",raw"$Y_2$",raw"$Y_3$",raw"$Y_4$"])
title("v∈[0,4]的第二类 Bessel 函数")
xlabel("z")
ylabel(raw"$Y_v(z)$")
hold("off")
```



3.2.1.7.2 Beta 函数

函数名	简介
beta	Beta 函数
beta_inc	不完全 beta 函数
betainc	不完全 beta 函数
beta_inc_inv	beta 逆累积分布函数
betaincinv	beta 逆累积分布函数
logbeta	beta 函数的对数
betaln	beta 函数的对数

beta

Beta 函数

A. 语法

```
B = beta(Z,W)
```

B. 说明

B = beta(Z,W)返回 Z 和 W 的元素处计算的 beta 函数。Z 和 W 都必须是非负数。

C. 示例

(1). 计算整数参数的 beta 函数

计算整数参数 w=3 和 z=1,...,10 的 beta 函数。根据定义，beta 函数可以计算为

$$B(z,3)=\frac{\Gamma(z)\Gamma(3)}{\Gamma(z+3)}=\frac{(z-1)!2!}{(z+2)!}=\frac{2}{z(z+1)(z+2)}$$

```
B = beta.(1:10,3)
```

```
B = 10-element Vector{Float64}:
0.33333333333333337
0.08333333333333333
0.03333333333333335
0.016666666666666656
0.009523809523809535
0.005952380952380948
0.003968253968253646
0.002777777777777675
0.002020202020202
0.001515151515151514
```

实际计算结果应为

[1//3,1//12,1//30,1//60,1//105,1//168,1//252,1//360,1//495,1//660]

将其化为小数形式与结果进行对比

```
Float64.([1//3,1//12,1//30,1//60,1//105,1//168,1//252,1//360,1//495,1
//660])
```

```
ans = 10-element Vector{Float64}:
0.3333333333333333
0.08333333333333333
0.03333333333333333
0.016666666666666666
0.009523809523809525
0.005952380952380952
0.003968253968253968
```

```
0.002777777777777778
0.00202020202020202
0.0015151515151515152
```

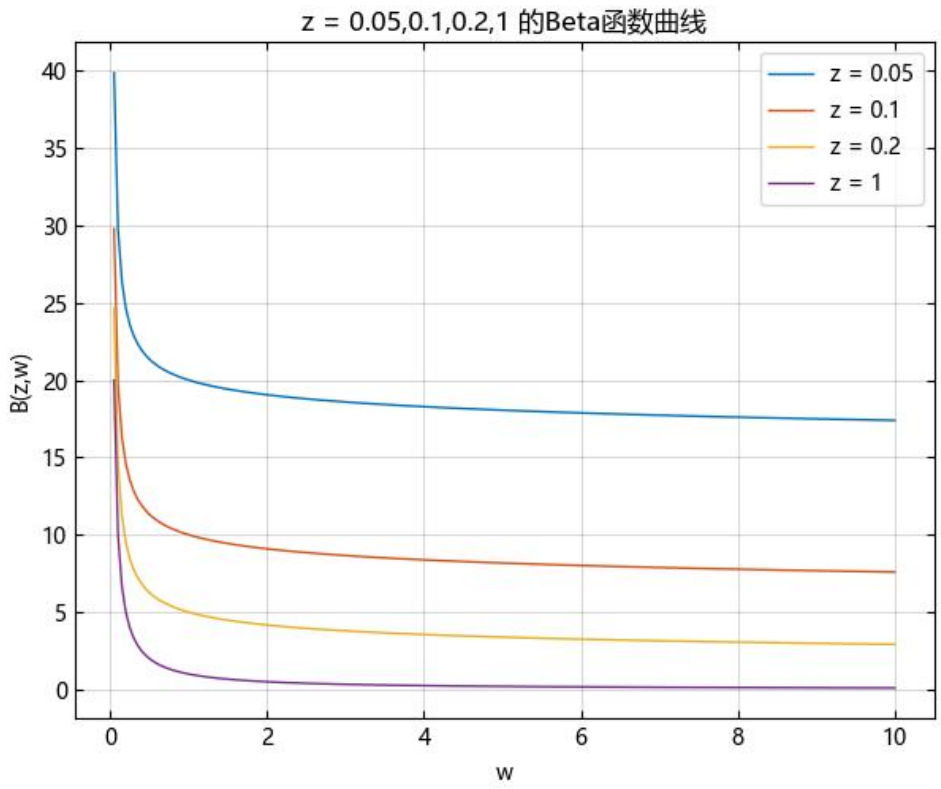
(2). 绘制 beta 函数

在 0≤w≤10 区间内，计算 z = 0.05、0.1、0.2 和 1 时的 beta 函数。逐一在每个 z 值处计算该函数，并将每次计算的结果赋给 B 的一行。

```
Z = [0.05 0.1 0.2 1]
W = 0:0.05:10
B = zeros(4,201)
for i = 1:4
B[i,:] = beta.(Z[i],W)
end
```

在同一图窗中绘制所有 beta 函数。

```
hold("on")
for i in 1:4
plot(W,B[i,:])
end
grid("on")
legend(["z = 0.05","z = 0.1","z = 0.2","z = 1"])
title("z = 0.05,0.1,0.2,1 的Beta函数曲线")
xlabel("w")
ylabel("B(z,w)")
```



beta_inc

不完全 beta 函数

A. 语法

```
I = beta_inc(a,b,x)
```

B. 说明

B = beta_inc(a,b,x)为数组 a,b,和 x 的相应元素计算不完全 beta 函数。x 的元素必须位于闭区间[0,1]中。数组 a 和 b 必须是非负实数。所有数组的大小必须相同，或者任一数组可以为标量。

C. 示例

(1). 计算不完全 Beta 函数

根据参数 b 和 x，计算与 a 的元素对应的不完全 Beta 函数。

```
a = 1:10
b = 3
x = 0.5
```

```
I = beta_inc.(a,b,x)
I =  map(x->x[1],I)
```

```
I = 10-element Vector{Float64}:
0.875
0.6875
0.5
0.3437500000000001
0.22656250000000003
0.1445312500000001
0.08984374999999988
0.05468750000000013
0.03271484374999999
0.019287109375000007
```

betainc

不完全 beta 函数

A. 语法

```
I = betainc(X,Z,W)
I = betainc(X,Z,W,islower)
```

B. 说明

I = betainc(X,Z,W) 为数组 X、Z 和 W 的相应元素计算不完全 beta 函数。X 的元素必须位于闭区间 [0,1] 中。数组 Z 和 W 必须是非负实数。所有数组的大小必须相同，或者任一数组可以为标量。示例

I = betainc(X,Z,W,islower) 指定不完全 beta 函数的尾部。相关选择包括：

true（默认值） - 计算 0 至 x 的积分

false - 计算 x 至 1 的积分

这些函数的关系如下所示：

1 - betainc(X,Z,W) = betainc(X,Z,W,false)

请注意，特别是当尾部上限值接近 0 时，使用 false 选项比用 1 减去 true 值更准确。

C. 示例

计算不完全 Beta 函数

根据参数 X 和 W，计算与 Z 的元素对应的不完全 Beta 函数。

```
X = 0.5
Z = 1:10
W = 3
I = betainc(X,Z,W)
```

```
I = 10-element Vector{Float64}:
 0.875
 0.6875
 0.5
 0.3437500000000001
 0.22656250000000003
 0.1445312500000001
 0.08984374999999988
 0.05468750000000013
 0.03271484374999999
 0.019287109375000007
```

指定不完全 beta 函数尾部为 false 并再次计算。

```
I_new = betainc(X,Z,W,false)
```

```
I_new = 10-element Vector{Float64}:
 0.125
 0.3125
 0.5
 0.6562499999999999
 0.7734375
 0.8554687499999999
 0.9101562500000001
 0.9453124999999999
 0.96728515625
 0.980712890625
```

beta_inc_inv

beta 逆累积分布函数

A. 语法

```
I = beta_inc_inv(a,b,p,q)
```

B. 说明

B = beta_inc_inv(a,b,x,p,q)针对 a、b、p、q 的对应元素计算逆不完全 beta 函数。p,q 的元素必须位于闭区间 [0,1]中，a 和 b 的元素必须为非负值。a,b,p,q

必须均为浮点数，并且其大小必须相同（或者任一元素可以为标量）。

C. 示例

(1). 计算逆不完全 Beta 函数

根据参数 b 和 x，计算与 a 的元素对应的不完全 Beta 函数。

```
a = 1.0
b = 3.0
p = 0:0.1:1
q = 1:-0.1:0
I = beta_inc_inv.(a,b,p,q)
```

```
I = 11-element Vector{Tuple{Float64, Float64}}:
(0.0, 1.0)
(0.034510615394370246, 0.9654893846056297)
(0.07168223327744422, 0.9283177667225557)
(0.11209599825739915, 0.8879040017426009)
(0.15656733469825077, 0.8434326653017492)
(0.2062994740159002, 0.7937005259840998)
(0.26319370027192257, 0.7368062997280774)
(0.3305670499178305, 0.6694329500821695)
(0.41519645235742675, 0.5848035476425733)
(0.5358411166387221, 0.46415888336127786)
(1.0, 0.0)
```

D. 详细信息

(1). 不完全 beta 函数

不完全 beta 函数为

$$I_x(a,b) = \frac{1}{\beta(a,b)} \int_0^x t^{a-1}(1-t)^{b-1} dt$$

beta_inc_inv 使用牛顿方法对积分极限 x 计算不完全 Beta 函数的逆函数。

betaincinv

beta 逆累积分布函数

A. 语法

```
I = betaincinv(y,z,w)
```

```
I = betaincinv(y,z,w,islower)
```

B. 说明

I = betaincinv(y,z,w) 针对 y、z 和 w 的对应元素计算逆不完全 beta 函数，以使 y = betainc(x,z,w)。y 的元素必须位于闭区间 [0,1] 中，z 和 w 的元素必须为非负值。y、z 和 w 必须均为实数，并且其大小必须相同（或者任一元素可以为标量）。示例

I = betaincinv(y,z,w,islower) 指定不完全 beta 函数的尾部。相关选择包括：

true（默认值）- 使用 0 至 x 的积分

false - 使用 x 至 1 的积分。

这两个选项的关系如下所示：betaincinv(y,z,w,true) = betaincinv(1-y,z,w,false)。如果 y 接近 0，同用 1 减去 y 这种方法相比，false 选项提供一种更准确的方法来计算 x。

C. 示例

(1). 计算逆不完全 Beta 函数

根据参数 Y 和 Z 和 W 的元素计算对应的逆不完全 Beta 函数。

```
Y = 0:0.1:1
Z = 3.0
W = 1.0
I = betaincinv(Y,Z,W)
```

```
I = 11-element Vector{Float64}:
 0.0
 0.4641588833612779
 0.5848035476425733
 0.6694329500821695
 0.7368062997280774
 0.7937005259840998
 0.8434326653017493
 0.8879040017426008
 0.9283177667225557
 0.9654893846056297
 1.0
```

指定其尾部为 false 再次计算。。

```
I_new = betaincinv(Y,Z,W,false)
```

```
I_new = 11-element Vector{Float64}:
 1.0
```

```
0.9654893846056297
0.9283177667225557
0.8879040017426008
0.8434326653017493
0.7937005259840998
0.7368062997280774
0.6694329500821696
0.5848035476425731
0.46415888336127786
0.0
```

D. 详细信息

(1). 不完全 beta 函数

不完全 beta 函数为

$$I_x(a,b)=\frac{1}{\beta(a,b)}\int_0^x t^{a-1}(1-t)^{b-1}dt$$

betaincinv 使用牛顿方法对积分极限 x 计算不完全 Beta 函数的逆函数。

logbeta

beta 函数的对数

A. 语法

```
L = logbeta(Z,W)
```

B. 说明

L = logbeta(Z,W) Z 和 W 的对应元素计算 beta 函数的自然对数 log(beta(Z,W))，而不会计算 beta(Z,W)。由于 beta 函数的范围可以涵盖极大或极小的值，因此其对数有时会更有用。

Z 和 W 必须是实数和非负数组。它们的大小必须相同，或者任一数组可以为标量。

C. 示例

(1). 计算 Beta 函数的自然函数

根据 X 的值计算 Beta 函数的自然对数，而不直接计算 Beta 函数。beta(X,X) 将产生浮点算术下溢。

```
X = 510
logbeta(X,X)
```

```
ans = -708.861572312519
```

D. 算法

logbeta(z,w) = loggamma(z) + loggamma(w) - loggamma(z+w)

betaln

beta 函数的对数

A. 语法

```
L = betaln(Z,W)
```

B. 说明

L = betaln(Z,W) Z 和 W 的对应元素计算 beta 函数的自然对数 log(beta(Z,W))，而不会计算 beta(Z,W)。由于 beta 函数的范围可以涵盖极大或极小的值，因此其对数有时会更有用。

Z 和 W 必须是实数和非负数组。它们的大小必须相同，或者任一数组可以为标量。 示例。

C. 示例

(1). 计算 Beta 函数的自然函数

根据 X 的值计算 Beta 函数的自然对数，而不直接计算 Beta 函数。beta(X,X) 将产生浮点算术下溢。

```
X = 510
logbeta(X,X)
```

```
ans = -708.861572312519
```

D. 算法

logbeta(z,w) = loggamma(z) + loggamma(w) - loggamma(z+w)

3.2.1.7.3 误差函数

函数名	简介
erf	误差函数
erfc	补余误差函数
erfcinv	逆补余误差函数
erfcx	换算补余误差函数
erfinv	逆误差函数
fresnelc	菲涅尔余弦积分函数
fresnels	菲涅尔正弦积分函数

ert

误差函数

A. 语法

```
erf(x)
erf(x,y)
```

B. 说明

erf(x)返回为 x 的每个元素计算误差函数。

erf(x,y) 准确计算 erf(y) - erf(x) (仅对实数参数有效)

C. 示例

(1). 求误差函数

求值的误差函数

```
erf(0.76)
```

```
ans = 0.7175367528055908
```

求向量元素的误差函数。

```
V = [-0.5,0,1,0.72]
erf.(V)
```

```
ans = 4-element Vector{Float64}:
-0.5204998778130465
0.0
```

```
0.8427007929497149
0.6914331231387512
```

求矩阵元素的误差函数。

```
M = [0.29 -0.11; 3.1 -2.9]
erf.(M)
```

```
ans = 2×2 Matrix{Float64}:
0.318283  -0.123623
0.999988  -0.999959
```

(2). 求正态分布的累积分布函数

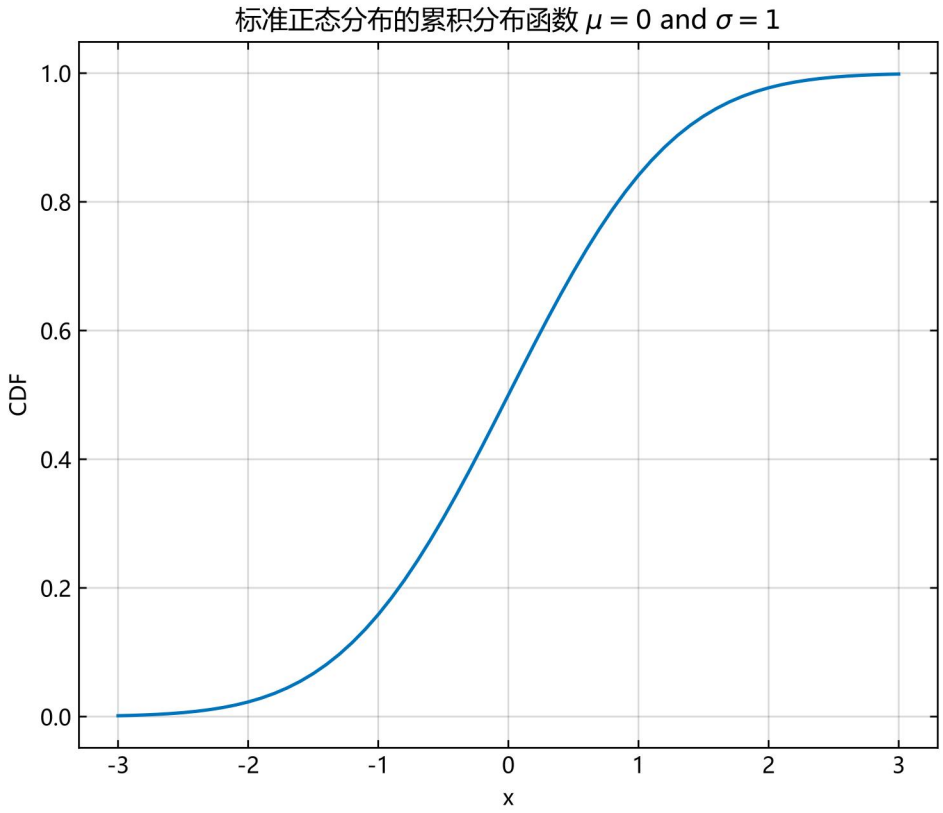
标准差为 σ 且均值为 μ 的正态或高斯分布的累积分布函数 (CDF) 为

$$.\phi(x) = \frac{1}{2}(1 + erf(\frac{x-\mu}{\sigma\sqrt{2}}))$$

请注意，为了提高计算准确性，您可以用 erfc 重写该公式。有关详细信息，请参阅 提示。

通过 μ=0 和 σ=1 绘制正态分布的 CDF。

```
x = -3:0.1:3
y = (1/2)*(1 .+ erf.(x./sqrt(2)))
plot(x,y)
grid("on")
title(raw"标准正态分布的累积分布函数 $\mu = 0$ and $\sigma = 1$")
xlabel("x")
ylabel("CDF")
```



(3). 通过初始条件计算热方程的解

其中， 表示位置为 x、时间为 t 时的温度，热方程为

$$\frac{\partial u}{\partial t} = c\frac{\partial^2 u}{\partial x^2}$$

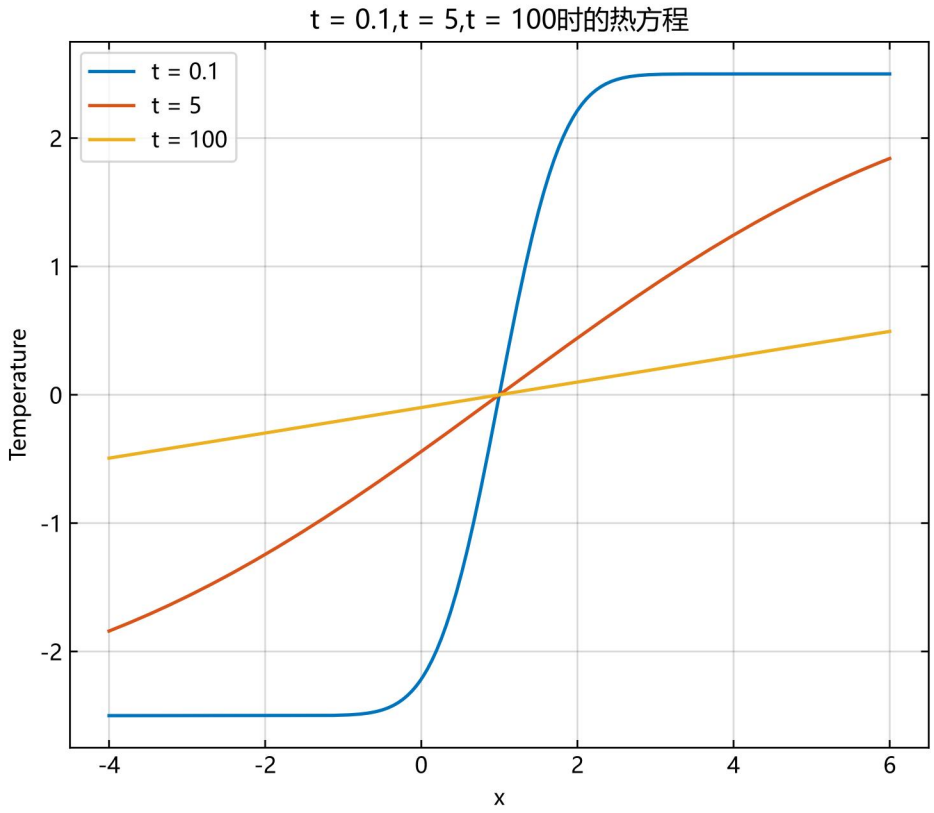
其中，c 是一个常量。

对于热系数为 k 的材料，以及对于其他位置 x>b 和 的初始条件，热方程的解为 $u(x,t) = \frac{a}{2}(erf(\frac{x-b}{\sqrt{4kt}}))$

对于 k = 2、a = 5 和 b = 1，为热方程在 t = 0.1、5 和 100 时的解绘图。

```
x = -4:0.01:6
t = [0.1 5 100]
a = 5
k = 2
b = 1
u = zeros(3,length(x))
```

```
figure(1)
hold("on")
for i = 1:3
u[i,:] = (a/2)*(erf.((x .- b) / sqrt(4*k*t[i])))
plot(x,u[i,:])
end
grid("on")
xlabel("x")
ylabel("Temperature")
legend(["t = 0.1","t = 5","t = 100"])
title("t = 0.1,t = 5,t = 100 时的热方程")
```



erfc

补余误差函数

A. 语法

```
erfc(x)
```

B. 说明

erfc(x)返回为 x 的每个元素计算补余误差函数。当 erf(x)接近 1 时，使用 erfc 函数替换 1-erf(x)以提高准确性。

C. 示例

(1). 求补余误差函数

求值的补余误差函数。

```
erfc(0.35)
```

```
ans = 0.6206179464376897
```

求向量元素的补余误差函数。

```
V = [-0.5,0,1,0.72]
erfc.(V)
```

```
ans = 4-element Vector{Float64}:
1.5204998778130465
1.0
0.15729920705028513
0.3085668768612489
```

求矩阵元素的补余误差函数。

```
M = [0.29 -0.11; 3.1 -2.9]
erfc.(M)
```

```
ans = 2×2 Matrix{Float64}:
0.681717    1.12362
1.16487e-5  1.99996
```

(2). 二相相移键控

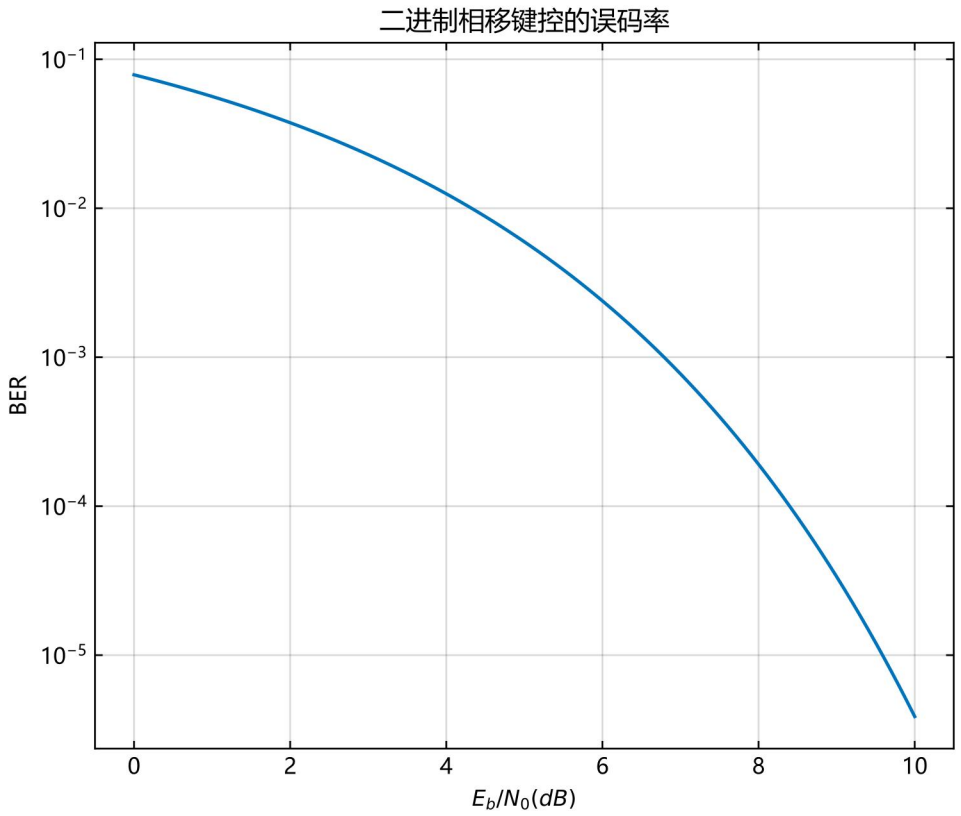
如果为加性高斯白噪声 (AWGN)，二相相移键控 (BPSK) 的误码率 (BER) 为

$$. P_b = \frac{1}{2}(\sqrt{\frac{E_b}{N_0}})$$

对于 E_b / N_0 从 0dB 到 10dB 的值，绘制 BPSK 的 BER。

```
EbN0_dB = 0:0.1:10
EbN0 = 10 .^ (EbN0_dB/10)
BER = 1/2 * erfc.(sqrt.(EbN0))
semilogy(EbN0_dB,BER)
grid("on")
title("二进制相移键控的误码率")
xlabel(raw"$E_b/N_0(dB)$")
```

```
ylabel("BER")
```



(3). 使用补余误差函数避免舍入误差

erf(x) 接近 1 时，您可以使用补余误差函数 erfc 替换 1 - erf(x) 以避免舍入误差。

说明如何通过使用 erfc(10) 计算 1 - erf(10) 来避免舍入误差。原始计算返回 0，而 erfc(10) 返回正确结果。

```
1 - erf(10)
```

```
ans = 0.0
```

```
erfc(10)
```

```
ans = 2.088487583762545e-45
```

erfcinv

逆补余误差函数

A. 语法

```
erfcinv(x)
```

B. 说明

erfcinv(x)返回为 x 的每个元素计算逆补余误差函数。对于[02]区间之外的输入，erfcinv 则会报错。当 x 接近 1 时，使用 erfcinv 函数替换包含 erfinv(1-x) 的表达式以提高准确性。

C. 示例

(1). 求逆补余误差函数

```
erfcinv(0.3)
```

```
ans = 0.7328690779592167
```

求向量元素的逆补余误差函数。

```
V = [0,0.5,1.3,2]
erfcinv.(V)
```

```
ans = 4-element Vector{Float64}:
Inf
0.4769362762044699
-0.2724627147267544
-Inf
```

求矩阵元素的逆补余误差函数。

```
M = [0.1 1.2; 1 0.9]
erfcinv.(M)
```

```
ans = 2×2 Matrix{Float64}:
1.16309  -0.179143
0.0       0.088856
```

(2). 使用逆补余误差函数避免舍入误差

x 接近 0 时，您可以使用逆补余误差函数 erfcinv 替换 erfinv(1-x) 以避免舍入误差。

说明如何通过对 x = 1e-100 使用 erfcinv(x) 计算 erfinv(1-x) 来避免舍入。原始计算返回 Inf，而 erfcinv(x) 返回正确结果。

```
x = 1e-100
erfinv(1-x)
```

```
ans = Inf
```

```
erfcinv(x)
```

```
ans = 15.065574702592645
```

erfcx

换算补余误差函数

A. 语法

```
erfcx(x)
```

B. 说明

erfcx(x)返回为 x 的每个元素计算换算补余误差函数。使用 erfcx 函数替换包含 exp(x^2)*erfc(x)的表达式以避免下溢或溢出错误。

C. 示例

(1). 求换算补余误差函数

```
erfcx(5)
```

```
ans = 0.11070463773306861
```

求向量元素的换算补余误差函数。

```
V = [-Inf,-1,0,1,10,Inf]
erfcx.(V)
```

```
ans = 6-element Vector{Float64}:
Inf
5.008980080762283
1.0
0.427583576155807
0.05614099274382259
0.0
```

求矩阵元素的换算补余误差函数。

```
M = [-0.5 15; 3.2 1]
erfcx.(M)
```

```
ans = 2×2 Matrix{Float64}:
1.95236   0.0375296
0.168728  0.427584
```

(2). 使用换算补余误差函数避免舍入误差

您可以使用换算补余误差函数 erfcx 替换 exp(x^2)*erfc(x) 以避免下溢或溢出错误。

说明如何通过使用 erfcx(35) 计算 exp(35^2)*erfc(35) 来避免舍入误差。原始计算返回 NaN，而 erfcx(35) 返回正确结果。

```
x = 35
exp(x^2)*erfc(x)
```

```
ans = NaN
```

```
erfcx(x)
```

```
ans = 0.01611313095681598
```

erfinv

逆误差函数

A. 语法

```
erfinv(x)
```

B. 说明

erfinv(x)返回为 x 的每个元素计算逆误差函数。对于[-11]区间之外的输入，erfinv 则会报错。

C. 示例

(1). 求值的逆误差函数

```
erfinv(0.25)
```

```
ans = 0.22531205501217808
```

对于 -1 和 1，erfinv 分别返回 -Inf 和 Inf。

```
erfinv.([1 -1])
```

```
ans = 1×2 Matrix{Float64}:
Inf  -Inf
```

求矩阵元素的逆误差函数。

```
M = [0 -0.5; 0.9 -0.2]
erfinv.(M)
```

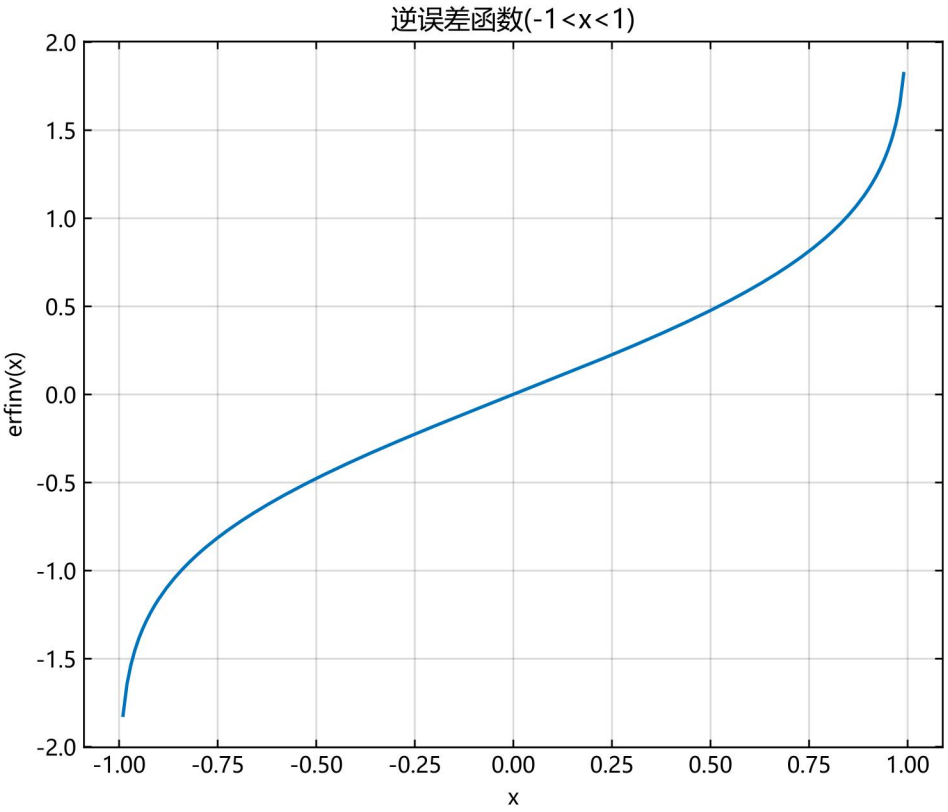
```
ans = 2×2 Matrix{Float64}:
0.0      -0.476936
1.16309  -0.179143
```

(2). 绘制逆误差函数图

绘制 -1 < x < 1 时的逆误差函数图。

```
x = -1:0.01:1
y = erfinv.(x)
```

```
plot(x,y)
grid("on")
title("逆误差函数(-1<x<1)")
xlabel("x")
ylabel("erfinv(x)")
```



(3). 生成高斯分布的随机数

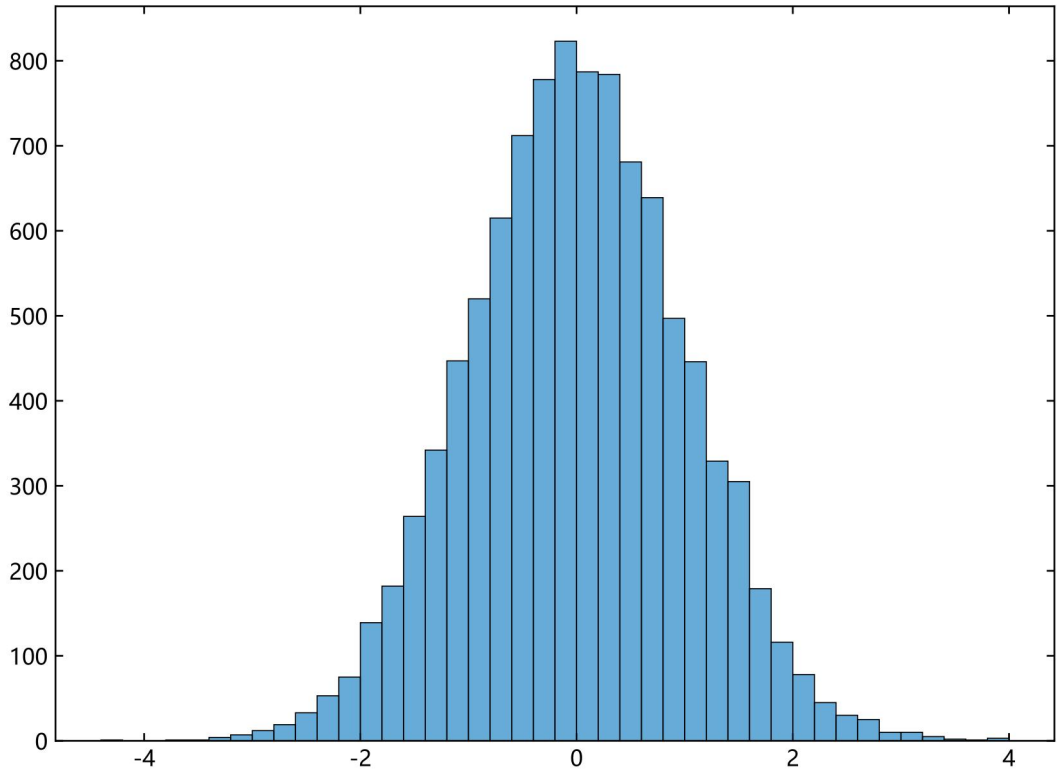
使用均匀分布的随机数生成高斯分布的随机数。要将均匀分布的随机数 x 变换为高斯分布的随机数 y，请使用下面的变换

$$y = \sqrt{2} erf^{-1}(x)$$

请注意，由于 x 采用 -1 .+ 2*rand(1,10000) 的形式，您可以通过使用 erfcinv 代替 erfinv 提高准确性。有关详细信息，请参阅 提示。

在区间 [-1,1] 内生成 10,000 个均匀分布的随机数。将它们转换为高斯分布的随机数。使用直方图说明这些数字遵照高斯分布形式。

```
x = -1 .+ 2*rand(1,10000)
y = sqrt(2)*erfinv.(x)
h = histogram(y)
```



fresnelc

菲涅尔余弦积分函数

A. 语法

```
A = fresnelc(z)
```

B. 说明

A = fresnelc(z) 返回菲涅尔余弦积分的 z。

C. 示例

(1). 数字的菲涅耳余弦积分函数

计算数字的菲涅耳余弦积分函数

```
fresnelc.([-2 0.001 1.22+0.31im])
```

```
ans = 1×3 Matrix{Number}:
-0.488253  0.001  0.861666-0.252365im
```

(2). 特殊值的菲涅耳余弦积分函数

求特殊值的菲涅耳余弦积分函数：

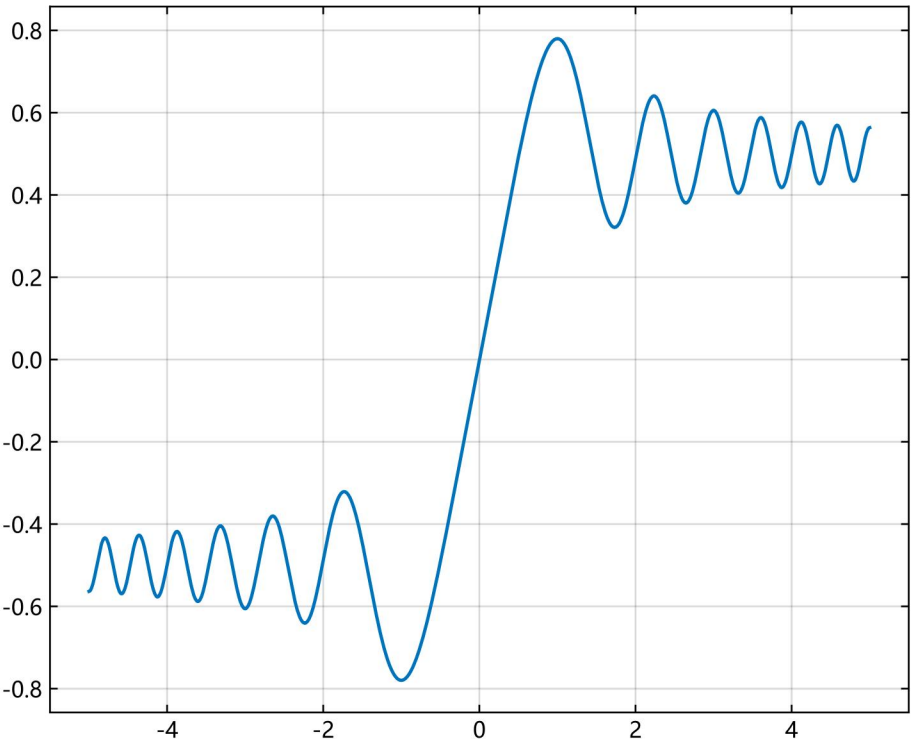
```
fresnelc.([0 Inf -Inf im*Inf -im*Inf])
```

```
ans = 1×5 Matrix{Number}:
0.0+0.0im  0.5  -0.5  0.0-0.5im  NaN+NaN*im
```

(3). 绘制菲涅尔余弦积分函数

从 x = -5 到 5 绘制菲涅耳余弦积分函数

```
x = -5:0.01:5
y = fresnelc.(x)
plot(x,y)
grid("on")
```



fresnels

菲涅尔正弦积分函数

A. 语法

```
A = fresnels(z)
```

B. 说明

A = fresnels(z) 返回菲涅尔正弦积分的 z。

C. 示例

(1). 数字的菲涅耳正弦积分函数

计算数字的菲涅耳正弦积分函数

```
fresnels.([-2 0.001 1.22+0.31im])
```

```
ans = 1×3 Matrix{Number}:
-0.343416  5.23599e-10  0.769692+0.294495im
```

(2). 特殊值的菲涅耳正弦积分函数

求特殊值的菲涅耳正弦积分函数：

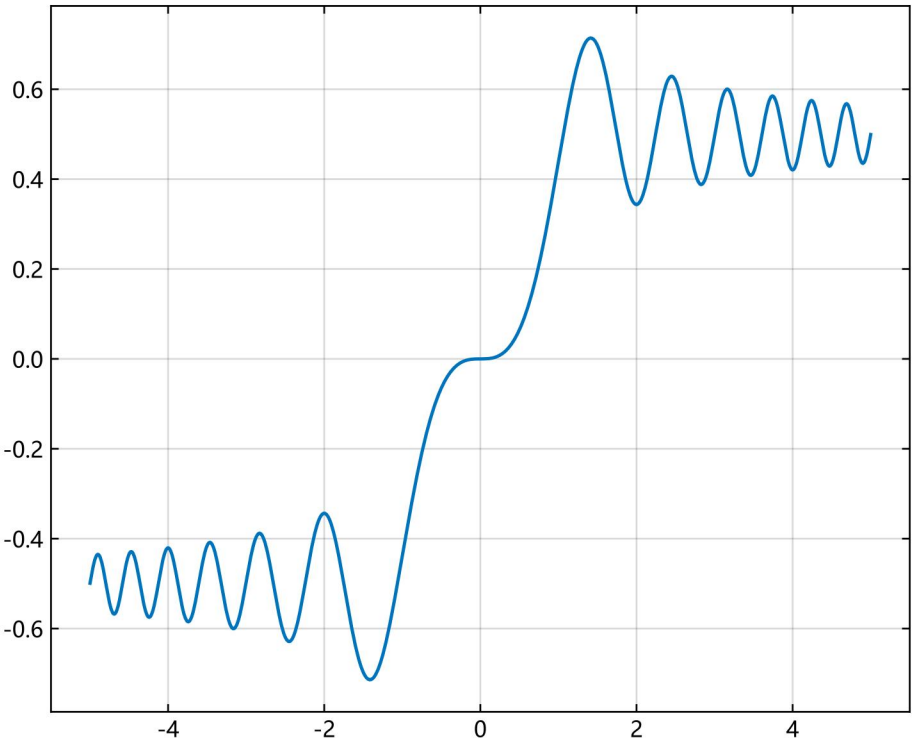
```
fresnels.([0 Inf -Inf im*Inf -im*Inf])
```

```
ans = 1×5 Matrix{Number}:
0.0+0.0im  0.5+0.0im  -0.5+0.0im  0.0+0.5im  NaN+NaN*im
```

(3). 绘制菲涅尔正弦积分函数

从 x = -5 到 5 绘制菲涅耳正弦积分函数

```
x = -5:0.01:5
y = fresnels.(x)
plot(x,y)
grid("on")
```



3.2.1.7.4 gamma 函数

函数名	简介
gamma	Gamma 函数
gamma_inc	不完全 gamma 函数
gammainc	不完全 gamma 函数
gamma_inc_inv	逆不完全 gamma 函数
gammaincinv	逆不完全 gamma 函数
loggamma	gamma 函数的对数
gammaln	gamma 函数的对数
digamma	digamma 函数
trigamma	trigamma 函数
polygamma	polygamma 函数
psi	digamma 和 polygamma 函数

gamma

Gamma 函数

A. 语法

```
gamma(x)
gamma(a,x)
```

B. 说明

gamma(x)返回 x 的元素处计算的 gamma 函数值。

gamma(a,x) 可支持任意实数和复数的计算。

C. 示例

(1). 计算 gamma 函数

使用标量和向量计算 gamma 函数。

计算$\Gamma(0.5)$,它等于$\sqrt{\pi}$

```
gamma(0.5)
```

```
ans = 1.772453850905516
```

计算 gamma 函数在 [-3.5 3.5] 之间的几个值。

```
x = -3.5:3.5
y = gamma.(x)
```

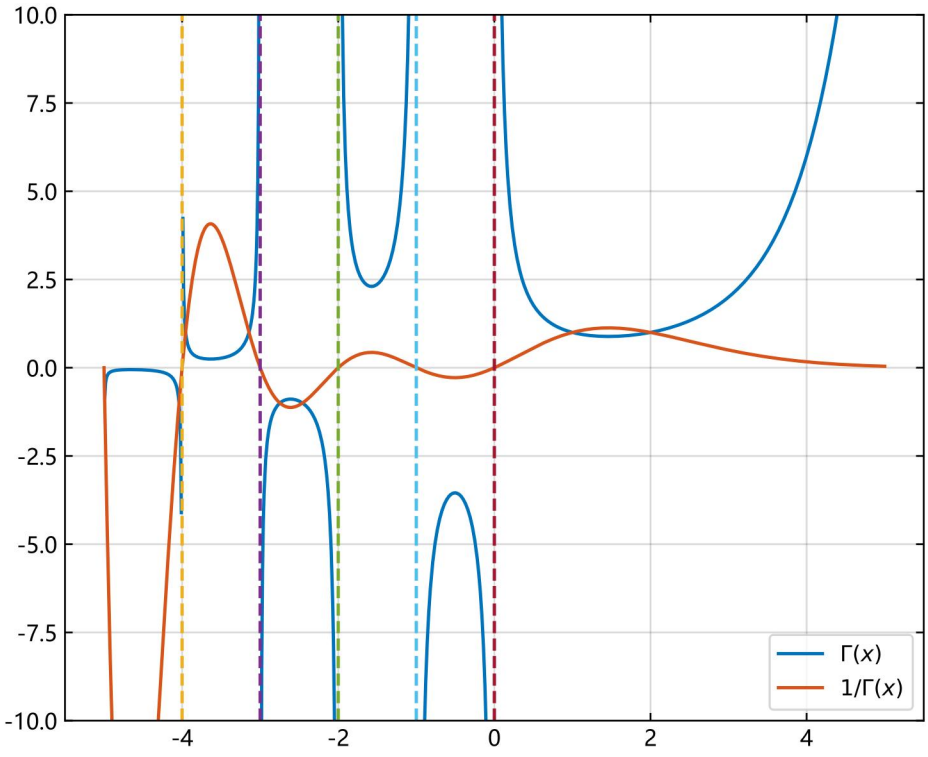
```
y = 8-element Vector{Float64}:
0.2700882058522691
-0.9453087204829418
2.363271801207355
-3.5449077018110318
1.772453850905516
0.886226925452758
1.329340388179137
3.3233509704478426
```

(2). 绘制 gamma 函数

绘制 gamma 函数及其倒数函数。

gamma 函数对于正参数快速增加，并且在所有负整数参数（以及 0）处具有简单的极点。该函数没有任何零点。相反，gamma 倒数函数在所有负整数参数（以及 0）处的值为零。

```
x = Complex.(-5:0.01:5)
y1 = gamma.(x)
y2 = 1 ./ y1
hold("on")
plot(x,y1)
plot(x,y2)
plot([-4,-4],[-10,10],"--",[-3,-3],[-10,10],"--",[-2,-2],[-
10,10],"--",[-1,-1],[-10,10],"--",[0,0],[-10,10],"--")
ylim([-10 10])
grid("on")
legend([raw"$\Gamma(x)$",raw"$1/\Gamma(x)$"])
hold("off")
```



gamma_inc

不完全 gamma 函数

A. 语法

```
I = gamma_inc(a,x,IND = 0)
```

B. 说明

B = gamma_inc(a,x,IND=0)返回在 a 和 x 的元素处计算的不完全 gamma 函数。a 和 x 必须都为实数，a 必须为非负值。

C. 示例

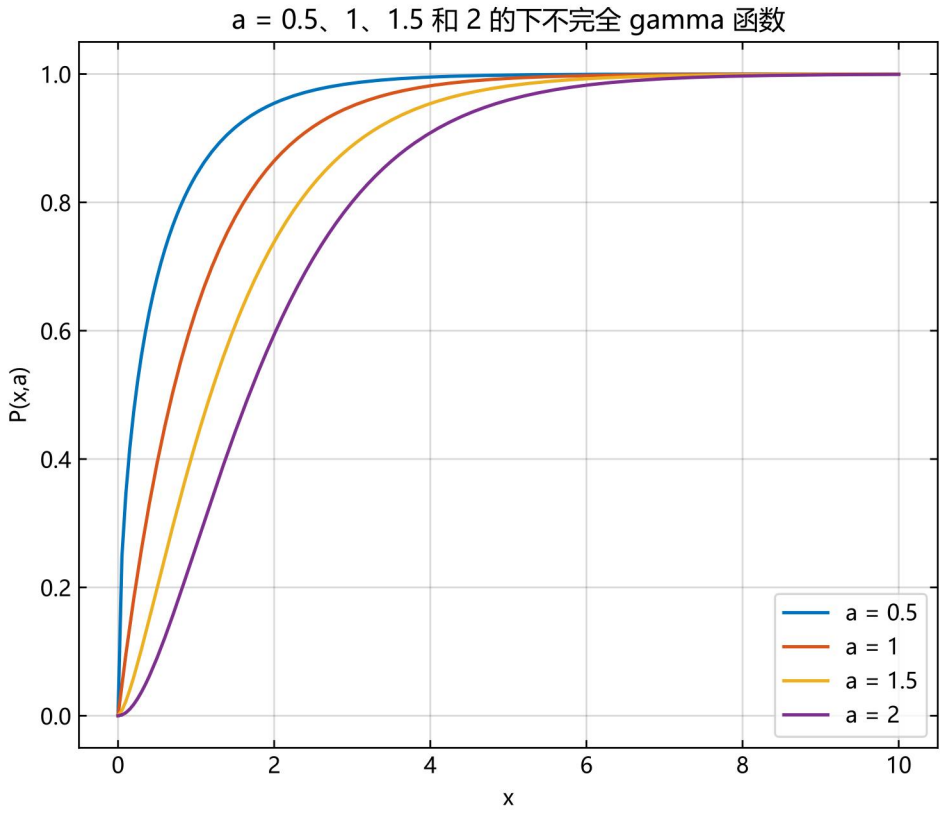
(1). 绘制下不完全 gamma 函数

计算区间 a 内 0≤x≤10= 0.5、1、1.5 和 2 的下不完全 gamma 函数。逐一计算每个 a 值对应的函数，将每个值的计算结果赋给 Y 的一列。

```
A = [0.5 1 1.5 2]
X = 0:0.05:10
Y = zeros(201,4)
for i = 1:4
for j = 1:length(X)
Y[j,i] = gamma_inc.(A[i],X)[j][1]
end
end
```

在同一图窗中绘制所有函数。

```
hold("on")
for i = 1:4
plot(X,Y[:,i])
end
grid("on")
legend(["a = 0.5","a = 1","a = 1.5","a = 2"])
title("a = 0.5、1、1.5 和 2 的下不完全 gamma 函数")
xlabel("x")
ylabel("P(x,a)")
```



(2). 绘制上不完全 gamma 函数

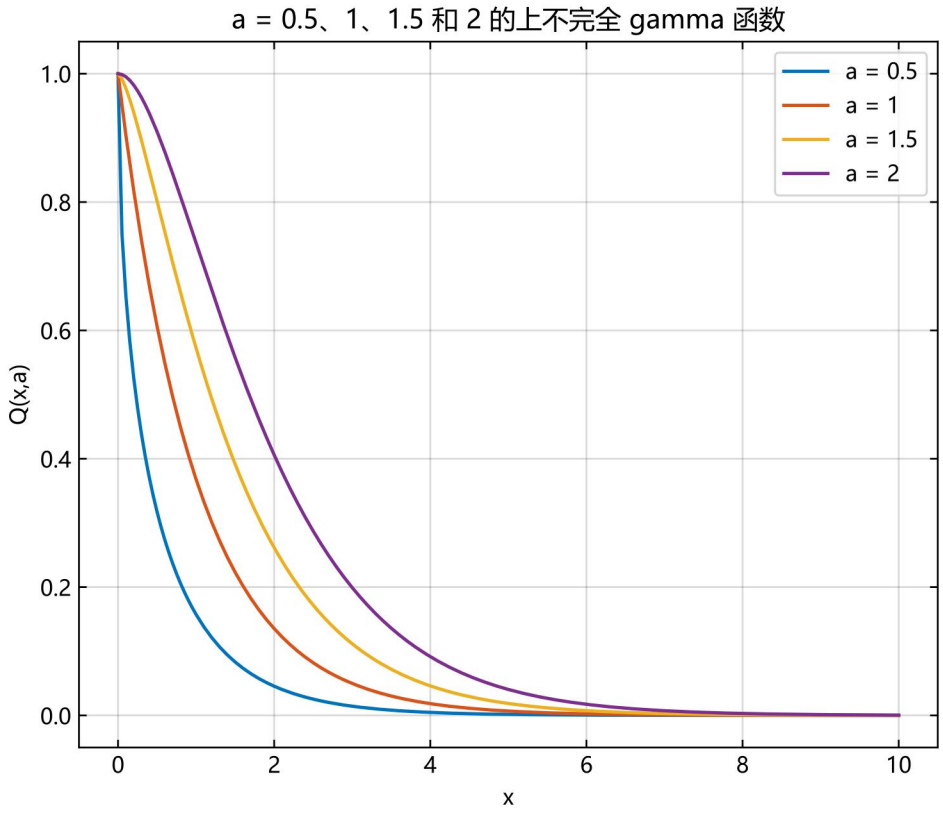
计算区间 a 内 0≤x≤10= 0.5、1、1.5 和 2 的上不完全 gamma 函数。逐一计算每个 a 值对应的函数，将每个值的计算结果赋给 Y 的一列。

```
A = [0.5 1 1.5 2]
X = 0:0.05:10
Y = zeros(201,4)
for i = 1:4
for j = 1:length(X)
Y[j,i] = gamma_inc.(A[i],X)[j][2]
end
end
```

在同一图窗中绘制所有函数。

```
hold("on")
for i = 1:4
plot(X,Y[:,i])
end
grid("on")
legend(["a = 0.5","a = 1","a = 1.5","a = 2"])
```

```
title("a = 0.5、1、1.5 和 2 的上不完全 gamma 函数")
xlabel("x")
ylabel("Q(x,a)")
```



gammainc

不完全 gamma 函数

A. 语法

```
Y = gammainc(X,A)
Y = gammainc(X,A,islower)
Y = gammainc(X,A,islower,scaled)
```

B. 说明

Y = gammainc(X,A) 返回在 X 和 A 的元素处计算的下不完全 gamma 函数 。X 和 A 必须都为实数，A 必须为非负值。示例

Y = gammainc(X,A,islower) 返回下/上不完全 gamma 函数。islower 的选项是 true（默认值）和 false。示例

Y = gammainc(X,A,islower,scaled) 缩放生成的下/上不完全 gamma 函数，以避免下溢为零或损失精度。scaled 的选项是 false（默认值）和 true。示例。

C. 示例

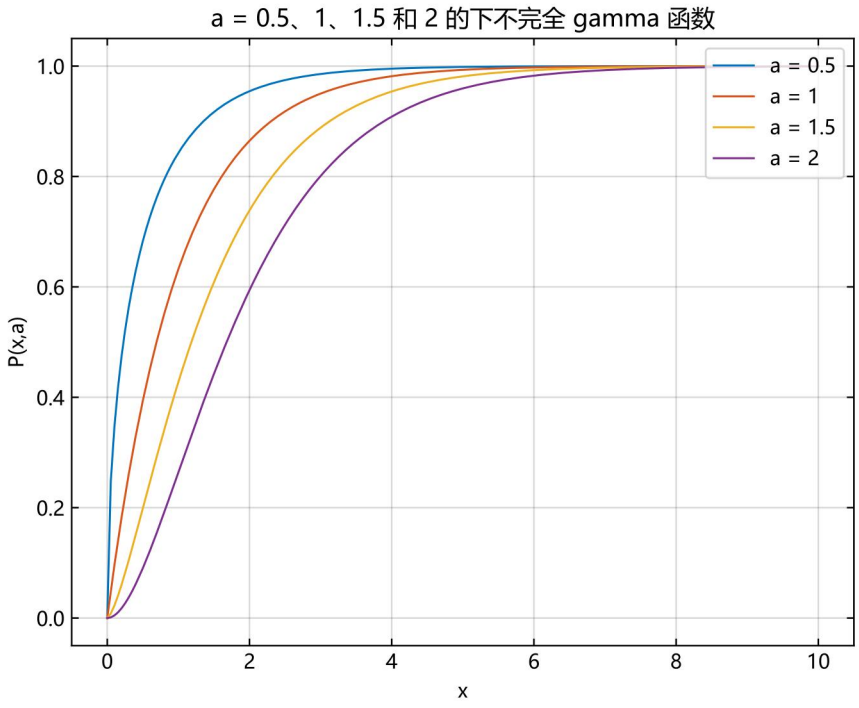
(1). 绘制下不完全 gamma 函数

计算 a = 0.5、1、1.5 和 2 时在区间 0≤x≤10 内的下不完全 gamma 函数。逐一计算每个 a 值对应的函数，将每个值的计算结果赋给 Y 的一列。

```
A = [0.5 1 1.5 2]
X = 0:0.05:10
Y = zeros(201,4)
for i = 1:4
    Y[:,i] = gammainc(X,A[i])
end
```

在同一图窗中绘制所有函数。

```
plot(X,Y)
grid("on")
legend(["a = 0.5","a = 1","a = 1.5","a = 2"],loc="northeast")
title("a = 0.5、1、1.5 和 2 的下不完全 gamma 函数")
xlabel("x")
ylabel("P(x,a)")
```



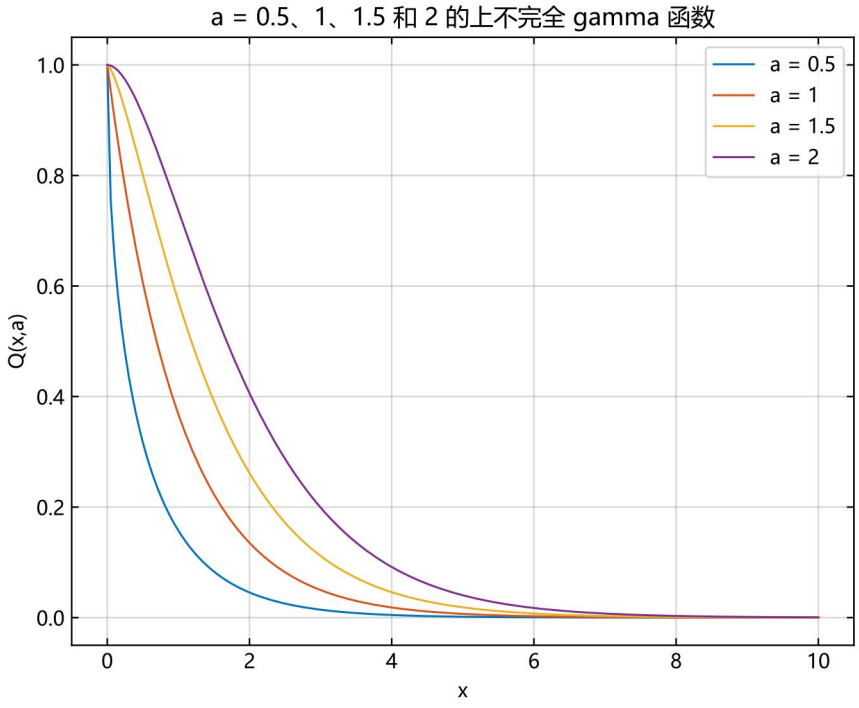
(2). 绘制上不完全 gamma 函数

计算 a = 0.5、1、1.5 和 2 时在区间 0≤x≤10 内的上不完全 gamma 函数。逐一计算每个 a 值对应的函数，将每个值的计算结果赋给 Y 的一列。

```
A = [0.5 1 1.5 2]
X = 0:0.05:10
Y = zeros(201,4)
for i = 1:4
    Y[:,i] = gammainc(X,A[i],false)
end
```

在同一图窗中绘制所有函数。

```
plot(X,Y)
grid("on")
legend(["a = 0.5","a = 1","a = 1.5","a = 2"],loc="northeast")
title("a = 0.5、1、1.5 和 2 的上不完全 gamma 函数")
xlabel("x")
ylabel("Q(x,a)")
```



(3). 制缩放的不完全 gamma 函数

计算未缩放的下不完全 gamma 函数，并将其与经过缩放的函数进行比较。

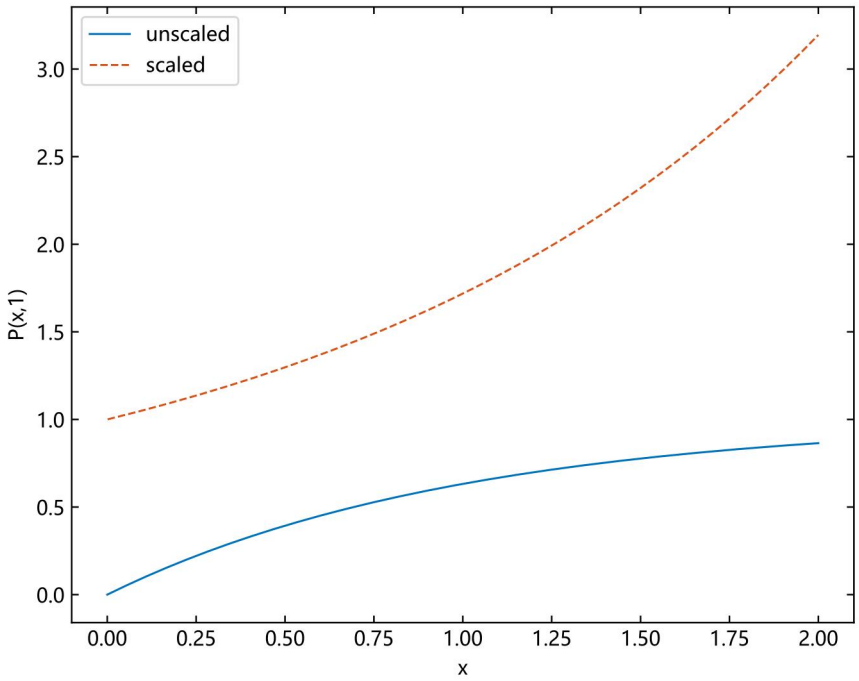
计算区间 0≤x≤2 内 a=1 的未缩放的下不完全 gamma 函数。绘制函数。

```
a = 1;
x = 0:0.001:2;
Y = gammainc(x,a);
plot(x,Y);
```

```
xlabel("x");
ylabel("P(x,1)")
hold("on")
```

接下来，计算缩放的下不完全 gamma 函数。在同一个图上绘制该函数。经过缩放的函数在 0 附近具有不同的渐近行为，可避免当 x 接近 0 时出现下溢。

```
Ys = gammainc(x,a,true,true);
plot(x,Ys,"--");
legend(["unscaled","scaled"])
```



gamma_inc_inv

逆不完全 gamma 函数

A. 语法

```
X = gamma_inc_inv(a,p,q)
```

B. 说明

X = gamma_inc_inv(a,p,q)返回在 p、q 和 a 元素处计算的下不完全 gamma 函数的逆函数，满足 p,q=gamma_inc(a,X)。a、p、q 都必须为实数。p、q 的元素必须在闭区间[0,1]内，a 必须为非负值。

C. 示例

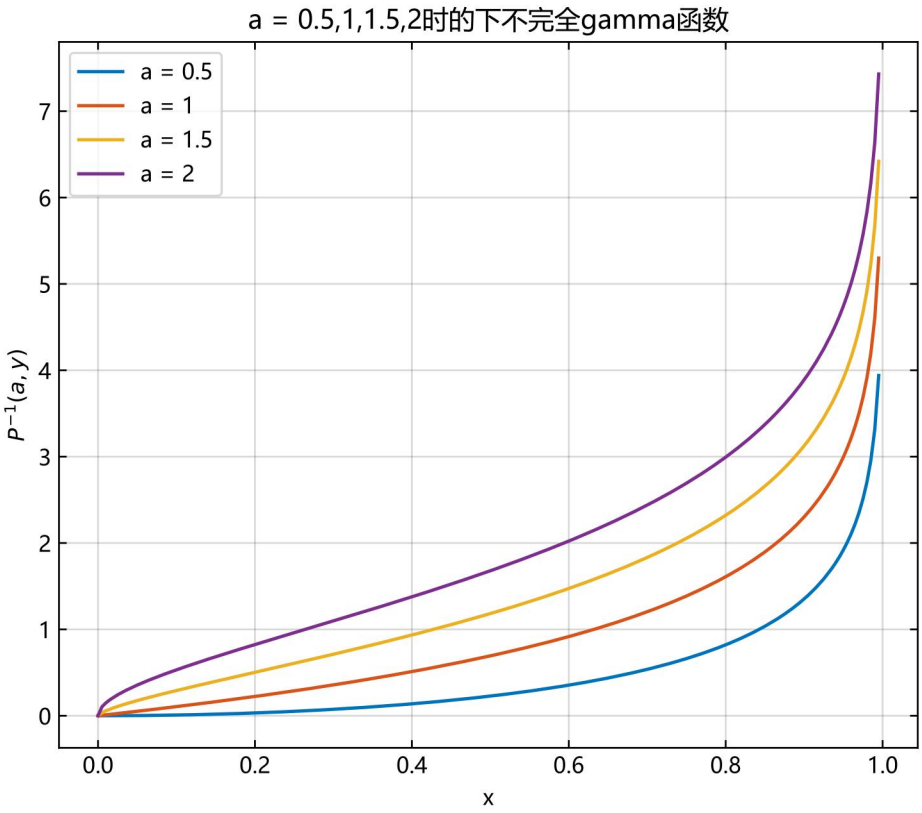
(1). 绘制下不完全 gamma 函数的逆函数

计算区间 0≤y≤1 内 a = 0.5、1、1.5 和 2 的下不完全 gamma 函数的逆函数。逐一计算每个 a 值对应的逆函数，将每个值的结果赋给 X 的一列。

```
A = [0.5 1 1.5 2]
Y = 0:0.005:1
X = zeros(201,4)
for i = 1:4
for j = 1:length(Y)
X[j,i] = gamma_inc_inv(A[i],Y[j],1-Y[j])
end
end
```

在同一图窗中绘制所有逆函数。

```
hold("on")
for i = 1:4
plot(Y,X[:,i])
end
grid("on")
legend(["a = 0.5","a = 1","a = 1.5","a = 2"])
xlabel("x")
ylabel(raw"$P^{-1}(a,y)$")
title("a = 0.5,1,1.5,2 时的下不完全 gamma 函数")
hold("off")
```



(2). 绘制上不完全 gamma 函数的逆函数

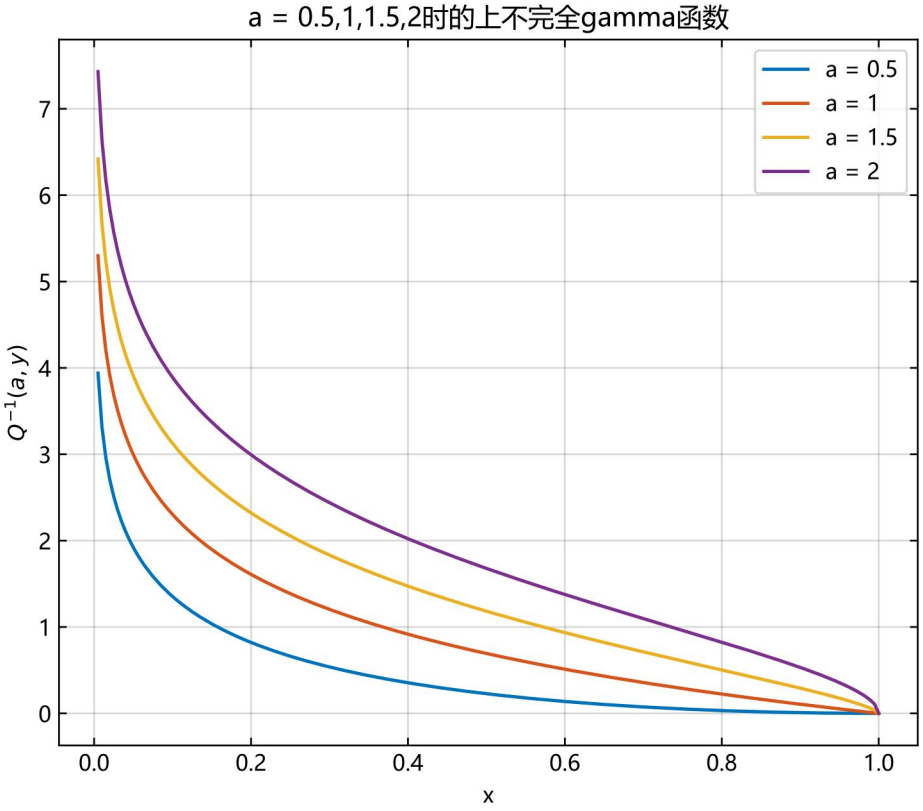
计算区间 0≤y≤1 内 a＝0.5、1、1.5 和 2 的上不完全 gamma 函数的逆函数。逐一计算每个 a 值对应的逆函数，将每个值的结果赋给 X 的一列。

```
A = [0.5 1 1.5 2]
Y = 0:0.005:1
X = zeros(201,4)
for i = 1:4
for j = 1:length(Y)
X[j,i] = gamma_inc_inv(A[i],1-Y[j],Y[j])
end
end
```

在同一图窗中绘制所有逆函数。

```
hold("on")
for i = 1:4
plot(Y,X[:,i])
end
grid("on")
legend(["a = 0.5","a = 1","a = 1.5","a = 2"])
```

```
xlabel("x")
ylabel(raw"$Q^{-1}(a,y)$")
title("a = 0.5,1,1.5,2 时的上不完全 gamma 函数")
hold("off")
```



gammaincinv

逆不完全 gamma 函数

A. 语法

```
X = gammaincinv(Y,A)
X = gammaincinv(Y,A,islower)
```

B. 说明

X = gammaincinv(Y,A) 返回在 Y 和 A 元素处计算的下不完全 gamma 函数的逆函数，满足 Y = gammainc(X,A)。Y 和 A 都必须为实数。Y 的元素必须在闭区间 [0,1] 内，A 必须为非负值。示例

X = gammaincinv(Y,A,islower) 返回下或上不完全 gamma 函数的逆函数。islower 的选项是 true（默认值）和 false。示例。

C. 示例

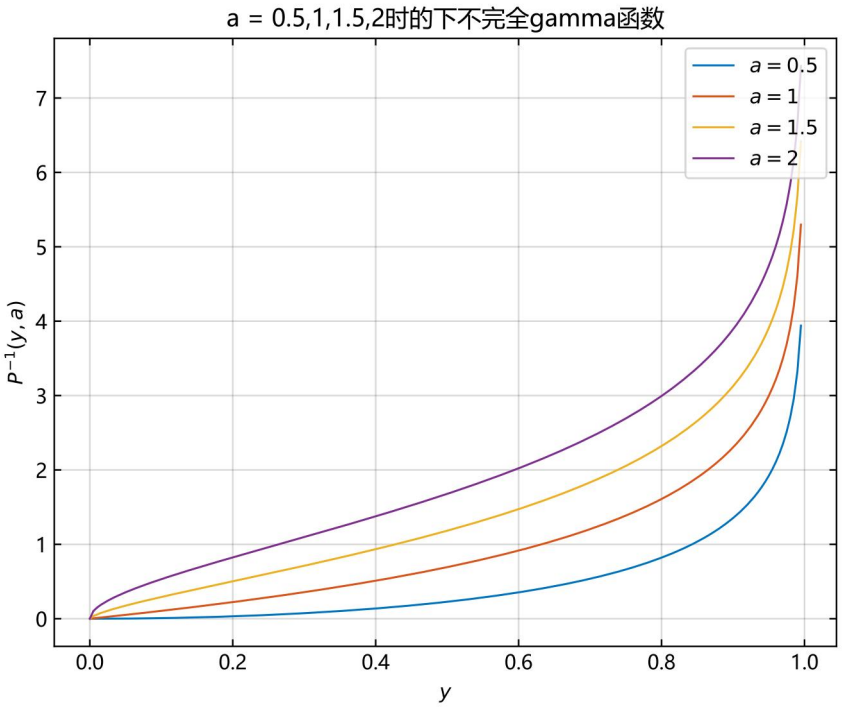
(1). 绘制下不完全 gamma 函数的逆函数

计算区间 0≤y≤1 内 a = 0.5、1、1.5 和 2 的下不完全 gamma 函数的逆函数。逐一计算每个 a 值对应的逆函数，将每个值的结果赋给 X 的一列。

```
A = [0.5 1 1.5 2]
Y = 0:0.005:1
X = zeros(201,4)
for i = 1:4
    X[:,i] = gammaincinv(Y,A[i])
end
```

在同一图窗中绘制所有逆函数。

```
plot(Y,X)
grid("on")
legend([raw"$a = 0.5$",raw"$a = 1$",raw"$a = 1.5$",raw"$a = 2$"],loc="northeast")
xlabel(raw"$y$")
ylabel(raw"$P^{-1}(y,a)$")
title("a = 0.5,1,1.5,2 时的下不完全 gamma 函数")
```



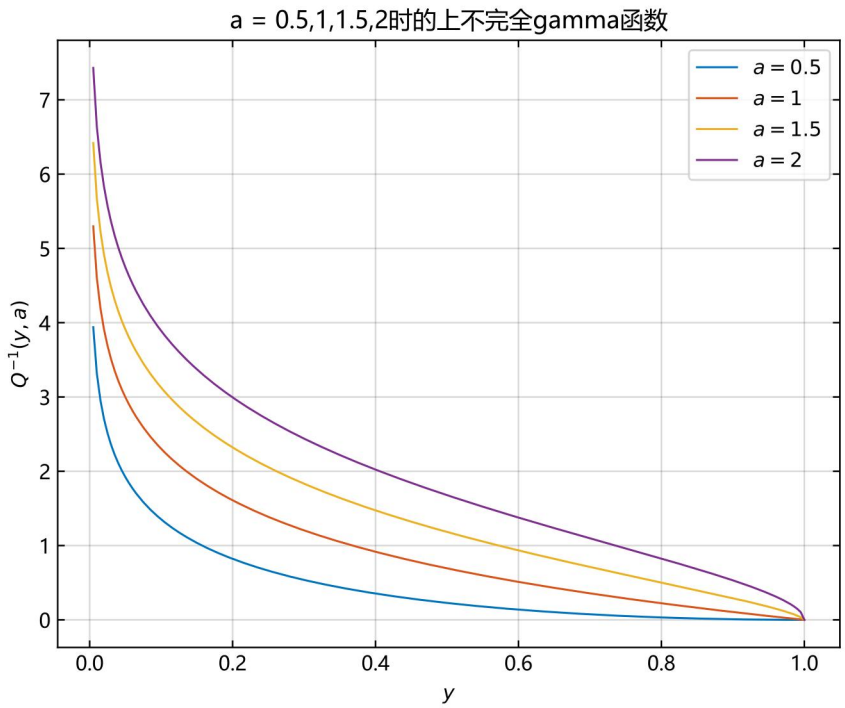
(2). 绘制下不完全 gamma 函数的逆函数

计算区间 0≤y≤1 内 a = 0.5、1、1.5 和 2 的上不完全 gamma 函数的逆函数。逐一计算每个 a 值对应的逆函数，将每个值的结果赋给 X 的一列。

```
A = [0.5 1 1.5 2]
Y = 0:0.005:1
X = zeros(201,4)
for i = 1:4
    X[:,i] = gammaincinv(Y,A[i],false)
end
```

在同一图窗中绘制所有逆函数。

```
plot(Y,X)
grid("on")
legend([raw"$a = 0.5$",raw"$a = 1$",raw"$a = 1.5$",raw"$a = 2$"],loc="northeast")
xlabel(raw"$y$")
ylabel(raw"$Q^{-1}(y,a)$")
title("a = 0.5,1,1.5,2 时的上不完全 gamma 函数")
```



loggamma

gamma 函数的对数

A. 语法

```
Y = loggamma(x)
Y = loggamma(a,x)
```

B. 说明

Y = loggamma(x) 返回 gamma 函数的对数 loggamma(x) = log(gamma(x))。loggamma 命令可避免直接使用 log(gamma(x)) 计算时可能会出现的下溢和上溢。

Y = loggamma(a,x) 返回 gamma 函数的对数 loggamma(a,x) = log(gamma(a,x))。

详细信息请参阅 gamma

C. 示例

(1). 计算 gamma 函数的自然函数

计算 gamma 函数的自然对数

```
x = 3.5
y1 = log(gamma(x))
y2 = loggamma(x)
y1 == y2
```

```
ans = true
```

gammaln

gamma 函数的对数

A. 语法

```
Y = gammaln(A)
```

B. 说明

Y = gammaln(A) 返回 gamma 函数的对数 gammaln(A) = log(gamma(A))。输入 A 必须是非负数和实数。gammaln 命令可避免直接使用 log(gamma(A)) 计算时可能会出现的下溢和上溢。示例

C. 示例

(1). 计算 gamma 函数的自然函数

计算 gamma 函数的自然对数

```
x = 3.5
```

```
y1 = log(gamma(x))
y2 = gammaln(x)
y1 == y2
```

```
ans = true
```

digamma

digamma 函数

A. 语法

```
Y = digamma(x)
```

B. 说明

Y = digamma(x) 计算 x 的 digamma 函数,等效于 polygamma(0,x)。

C. 示例

(1). 计算 Euler-Mascheroni 常量

使用 digamma 函数计算 Euler-Mascheroni 常量 γ，也称为欧拉常量。

```
Y = -digamma(1)
```

```
Y = 0.5772156649015315
```

trigamma

trigamma 函数

A. 语法

```
Y = trigamma(x)
```

B. 说明

Y = trigamma(x) 计算 x 的 trigamma 函数,等效于 polygamma(1,x)。

C. 示例

(1). 计算 trigamma 函数

计算 trigamma 函数

```
Y1 = trigamma(2)
```

```
Y1 = 0.6449340668482261
```

``

检查结果是否等于 $\pi^2/6 - 1$

```julia
Y2 = pi^2/6 - 1
```

```
Y2 = 0.6449340668482264
```

polygamma

polygamma 函数

A. 语法

```
Y = polygamma(k,x)
```

B. 说明

Y = polygamma(k,x) 计算 x 的 polygamma 函数，计算的值是在 x 处的 digamma 函数的 k 阶导数。因此，polygamma(0,x) 是 digamma 函数，polygamma(1,x) 是 trigamma 函数，polygamma(2,x) 是 tetragamma 函数，以此类推。

C. 示例

(1). 计算 trigamma 函数

计算 trigamma 函数

```
Y1 = polygamma(1,2)
```

```
Y1 = 0.6449340668482261
```

检查结果是否等于

```
Y2 = pi^2/6 - 1
```

```
Y2 = 0.6449340668482264
```

(2). 绘制 digamma 和 polygamma 函数定义域

```
X = 0:0.05:5
```

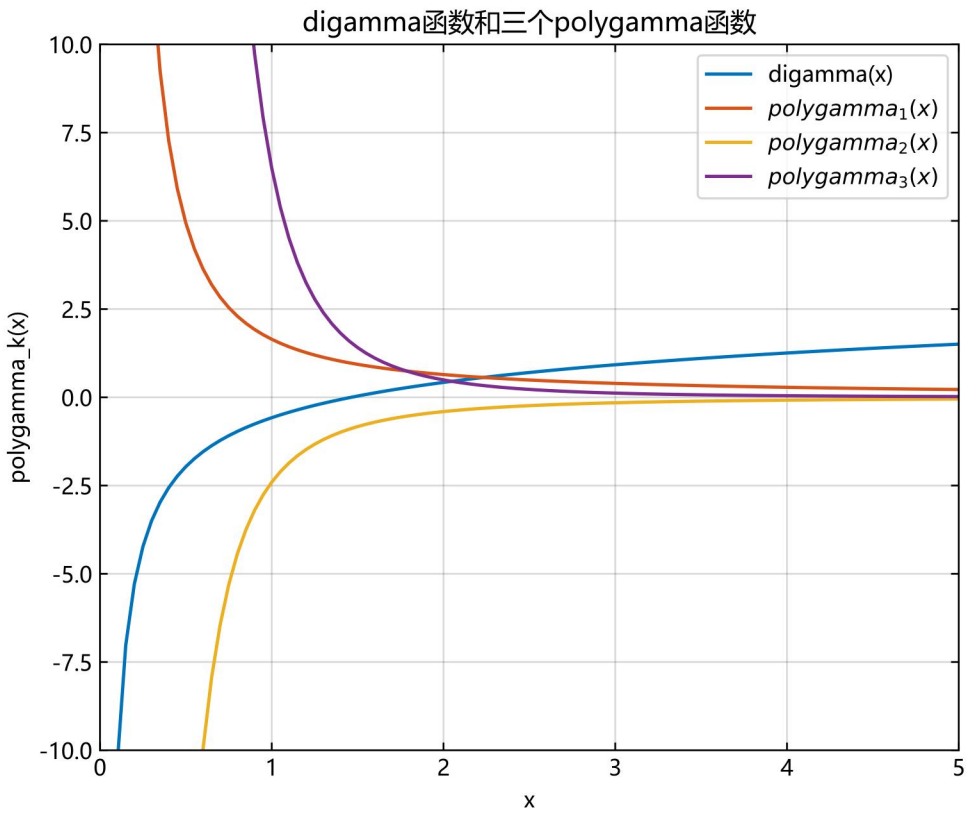
计算 digamma 函数和三个 polygamma 函数

```
Y = zeros(4,101)
for i = 0:3
```

```
Y[i+1,:] = polygamma.(i,X)
end
```

绘制 digamma 函数 和 三个 polygamma 函数

```
hold("on")
for i = 1:4
plot(X,Y[i,:])
end
axis([0 5 -10 10])
grid("on")
legend([raw"digamma(x)",raw"$polygamma_1(x)$",raw"$polygamma_2(x)$",r
aw"$polygamma_3(x)$"])
title("digamma 函数和三个 polygamma 函数")
xlabel("x")
ylabel(raw"polygamma_k(x)")
hold("off")
```



psi

digamma 和 polygamma 函数

A. 语法

```
Y = psi(X)
Y = psi(k, X)
```

B. 说明

Y = psi(X) 为数组 X 的每个元素计算 digamma 函数，各元素必须为非负实数。示例

Y = psi(k,X) 计算 X 的 polygamma 函数，计算的值是在 X 处的 digamma 函数的 k 阶导数。因此，psi(0,X) 是 digamma 函数，psi(1,X) 是 trigamma 函数，psi(2,X) 是 tetragamma 函数，以此类推。示例

C. 示例

(1). 计算 Euler-Mascheroni 常量

使用 psi 函数计算 Euler-Mascheroni 常量 γ，也称为欧拉常量。

```
Y = -psi(1)
```

```
Y = 0.5772156649015315
```

(2). 计算 trigamma 函数

计算 2 的 trigamma 函数。

```
Y1 = psi(1,2)
```

```
Y1 = 0.6449340668482261
```

检查结果是否等于 $\frac{\pi^2}{6}-1$。

```
Y2 = pi^2/6-1
```

```
Y2 = 0.6449340668482264
```

```
isapprox(Y1, Y2)
```

```
ans = true
```

(3). 绘制 digamma 和 polygamma 函数

定义域。

```
X = 0:0.05:5
```

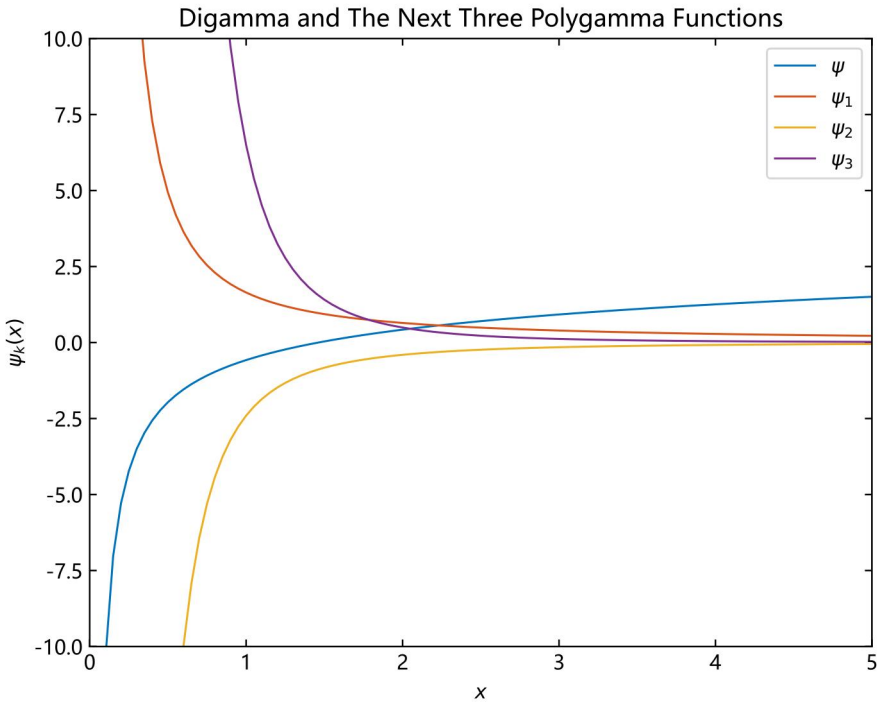
计算 digamma 和接下来的三个 polygamma 函数。

```
Y = zeros(4,101)
```

```
for i = 0:3
    Y[i+1,:] = psi(i,X)
end
```

绘制 digamma 和接下来的三个 polygamma 函数

```
plot(X,Y')
xlim([0,5])
ylim([-10,10])
legend([raw"$\psi$",raw"$\psi_1$",raw"$\psi_2$",raw"$\psi_3$"],loc="n
ortheast")
title("Digamma and The Next Three Polygamma Functions")
xlabel(raw"$x$")
ylabel(raw"$\psi_k(x)$")
```



3.2.1.7.5 椭圆积分

函数名	简介
ellipj	Jacobi 椭圆积分
ellipke	第一类和第二类完全椭圆积分
Elliptic.E	第二类完全和不完全椭圆积分
Elliptic.F	第一类不完全椭圆积分
Elliptic.K	第一类完全椭圆积分

Elliptic.Pi	第三类完全和不完全椭圆积分
ellipticCE	第二类互补完全椭圆积分
ellipticCK	第一类互补完全椭圆积分

ellipj

Jacobi 椭圆积分

A. 语法

```
SN,CN,DN = ellipj(U,M)
SN,CN,DN = ellipj(U,M,tol)
```

B. 说明

SN,CN,DN = ellipj(U,M) 返回 Jacobi 椭圆函数 SN、CN 和 DN，针对实参 U 和形参 M 的相应元素对这些函数求解。输入 U 和 M 的大小必须相同，或者 U 或 M 必须为标量。

SN,CN,DN = ellipj(U,M,tol) 以 tol 精度计算 Jacobi 椭圆函数。tol 的默认值是 eps()。增加 tol 会降低精度，但计算答案的速度会更快。

C. 示例

(1). 求 Jacobi 椭圆函数

求 U = 0.5 和 M = 0.25 的 Jacobi 椭圆函数。

```
s,c,d = ellipj(0.5,0.25)
```

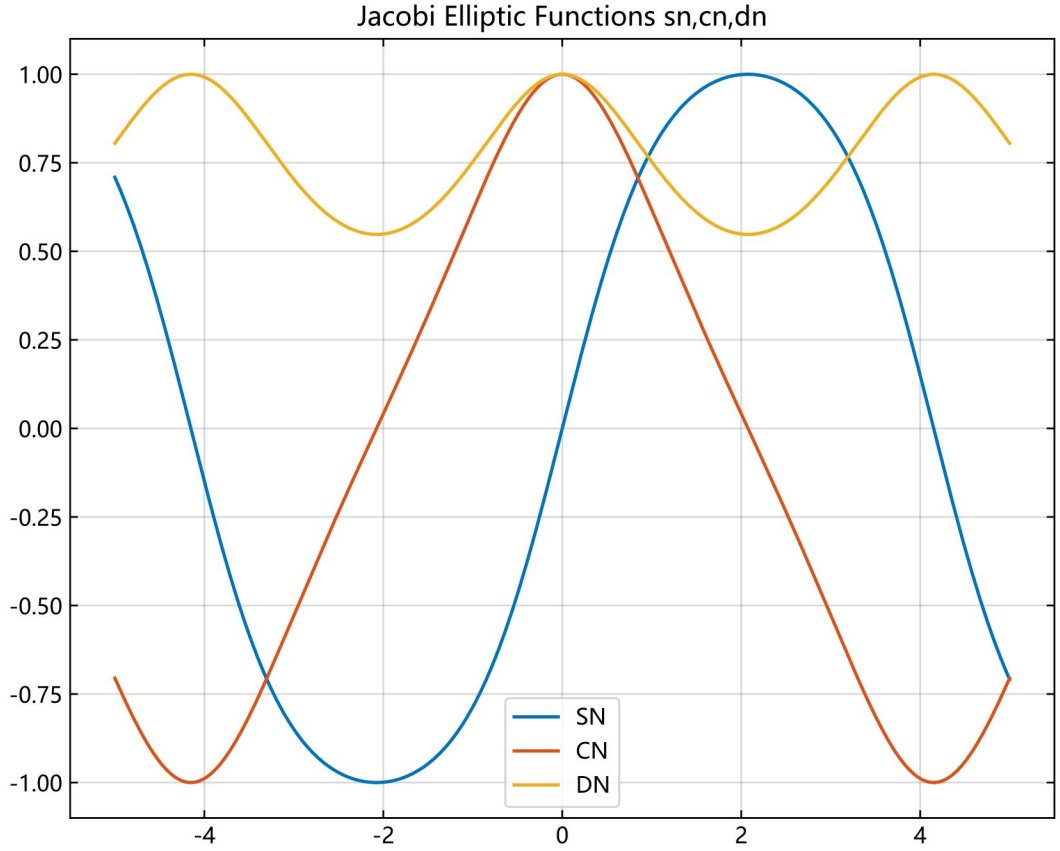
```
s = 0.47508293602853646
c = 0.8799410229637583
d = 0.9713773988381789
```

(2). 绘制 Jacobi 椭圆函数

绘制 -5≤U≤5 和 M = 0.7 的 Jacobi 椭圆函数。

```
M = 0.7
U = -5:0.01:5
SCD = ellipj.(U,M)
S = map(x->x[1],SCD)
C = map(x->x[2],SCD)
D = map(x->x[3],SCD)
plot(U,S,U,C,U,D)
```

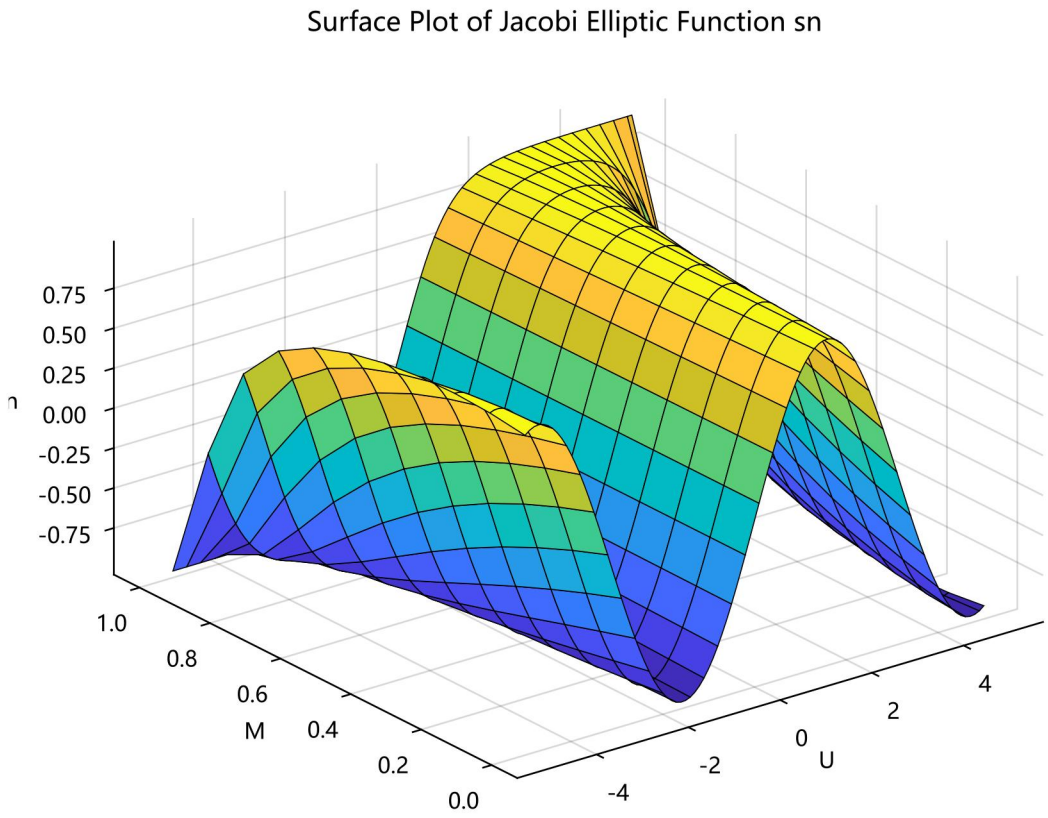
```
legend(["SN","CN","DN"])
grid("on")
title("Jacobi Elliptic Functions sn,cn,dn")
```



(3). 生成 Jacobi 椭圆函数 sn 的曲面图

在 M 的允许范围内和 -5≤U≤5 内生成 Jacobi 椭圆函数 sn 的曲面图。

```
M,U = meshgrid2(0:0.1:1,-5:0.1:5)
S = ellipj.(U,M)
S = map(x->x[1],S)
surf(U,M,S)
xlabel("U")
ylabel("M")
zlabel("sn")
title("Surface plot of Jacobi Elliptic Function sn")
```



(4). 通过更改容差更快地计算 Jacobi 椭圆积分

tol 的默认值是 eps()。使用 tic 和 toc 求任意 M 为默认值时的运行时间。按因子 1000 增加 tol 并计算运行时间。比较运行时间。

```
@time ellipj(0.253,0.937)
```

```
0.000016 seconds
(0.24789831515346017, 0.9687860575710593, 0.9707821591620847)
```

```
@time ellipj(0.253,0.937,eps()*1000)
```

```
0.000002 seconds
(0.24789831515346017, 0.9687860575710593, 0.9707821591620847)
```

容差明显增加时，`ellipj` 的运行速度明显加快。

ellipke

第一类和第二类完全椭圆积分

A. 语法

```
ellipke(M)
```

B. 说明

ellipke(M)返回第一类和第二类完全椭圆积分。

C. 示例

(1). 计算第一类和第二类完全椭圆积分

计算 M = 0.5 的第一类和第二类完全椭圆积分。

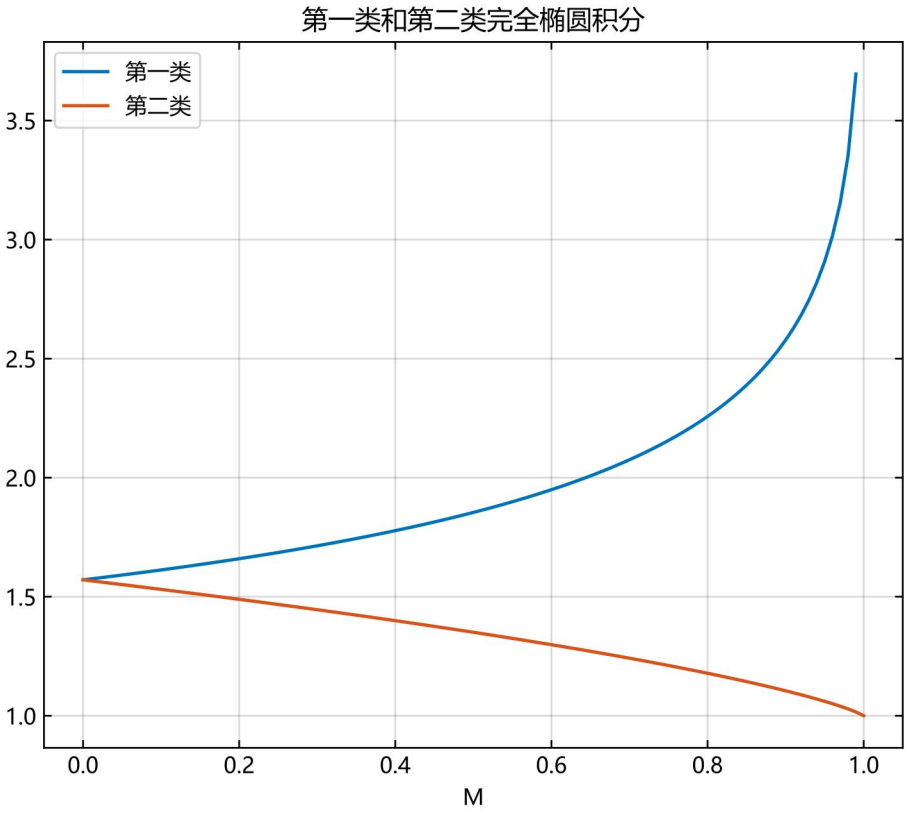
```
M = 0.5
K,E= ellipke(M)
```

```
K = 1.854074677301372
E = 1.3506438810476757
```

(2). 绘制第一类和第二类完全椭圆积分

基于 M 的允许范围绘制第一类和第二类完全椭圆积分。

```
M = 0:0.01:1
K = zero(M)
E = zero(M)
for i = 1:length(M)
K[i],E[i] = ellipke.(M)[i]
end
plot(M,K,M,E)
grid("on")
title("第一类和第二类完全椭圆积分")
xlabel("M")
legend(["第一类","第二类"])
```



Elliptic.E

第二类完全和不完全椭圆积分

A. 语法

```
Elliptic.E(m)
Elliptic.E(phi,m)
```

B. 说明

Elliptic.E(m)返回第二类完全椭圆积分。

Elliptic.E(phi,m)返回第二类不完全椭圆积分。

C. 示例

(1). 求第二类完全椭圆积分

```
s = Elliptic.E.([-10.5 -pi/4 0 1])
```

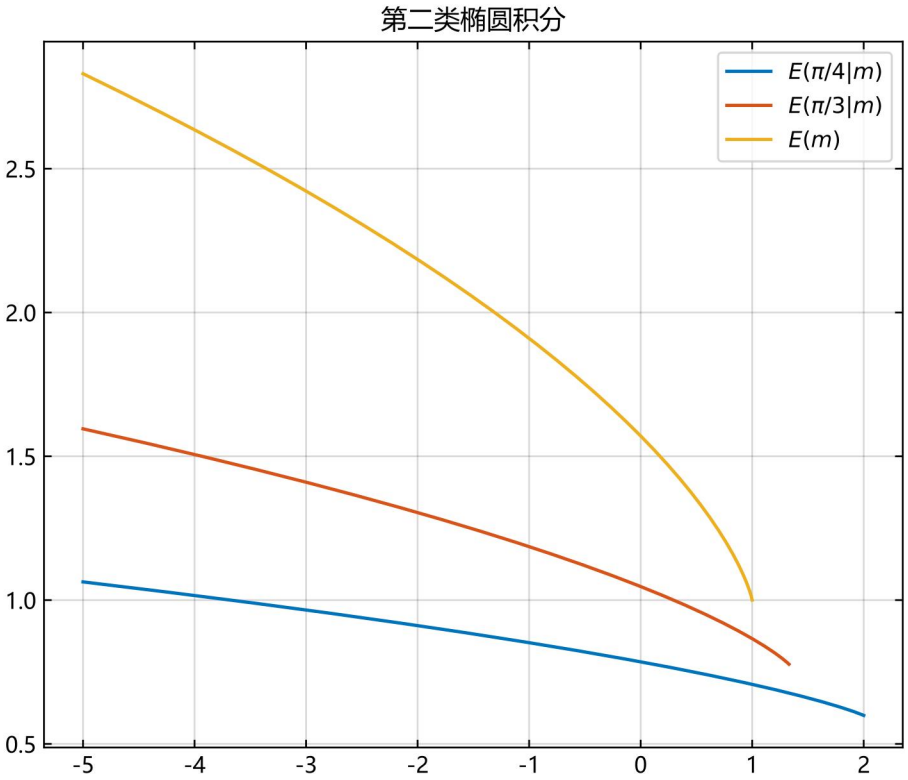
s = 1×4 Matrix{Float64}:

3.70961 1.84435 1.5708 1.0

(2). 绘制第二类完全和不完全椭圆积分

基于 M 的允许范围绘制第一类和第二类完全椭圆积分。

```
x1 = -5:0.01:2
x2 = -5:0.01:1.5
x3 = -5:0.01:1
y1 = Elliptic.E.(pi/4,x1)
y2 = Elliptic.E.(pi/3,x2)
y3 = Elliptic.E.(x3)
plot(x1,y1,x2,y2,x3,y3)
grid("on")
title("第二类椭圆积分")
legend([raw"$E(\pi/4|m)$",raw"$E(\pi/3|m)$",raw"$E(m)$"])
```



Elliptic.F

第一类不完全椭圆积分

A. 语法

```
Elliptic.F(phi,m)
```

B. 说明

Elliptic.F(phi,m)返回第一类不完全椭圆积分。

C. 示例

(1). 求第一类不完全椭圆积分

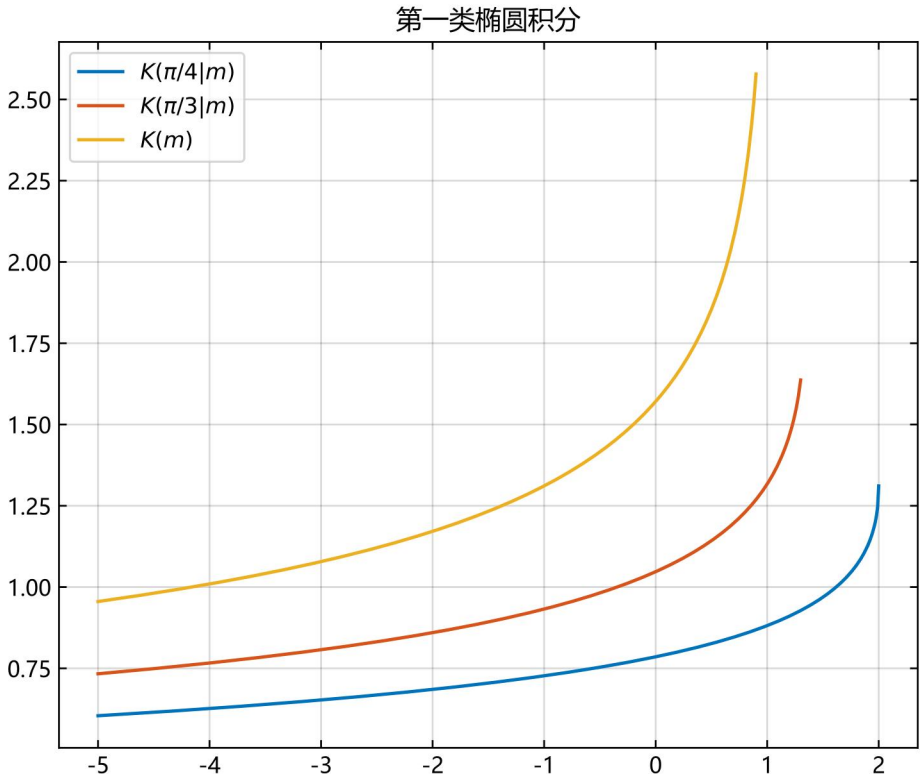
```
s = Elliptic.F.([pi/3 pi/4 1 pi/2],[-10.5 -pi -1 0])
```

```
s = 1×4 Matrix{Float64}:
0.618446   0.648597   0.896394   1.5708
```

(2). 绘制第一类完全和不完全椭圆积分

绘制第一类完全和不完全椭圆积分。

```
X1 = -5:0.01:2
x2 = -5:0.01:1.3
x3 = -5:0.01:0.9
y1 = Elliptic.F.(pi/4,x1)
y2 = Elliptic.F.(pi/3,x2)
y3 = Elliptic.F.(pi/2,x3)
plot(x1,y1,x2,y2,x3,y3)
grid("on")
title("第一类椭圆积分")
legend([raw"$K(\pi/4|m)$",raw"$K(\pi/3|m)$",raw"$K(m)$"])
```



Elliptic.K

第一类完全椭圆积分

A. 语法

```
Elliptic.K(m)
```

B. 说明

Elliptic.K(m)返回第一类完全椭圆积分。

C. 示例

(1). 求第一类完全椭圆积分

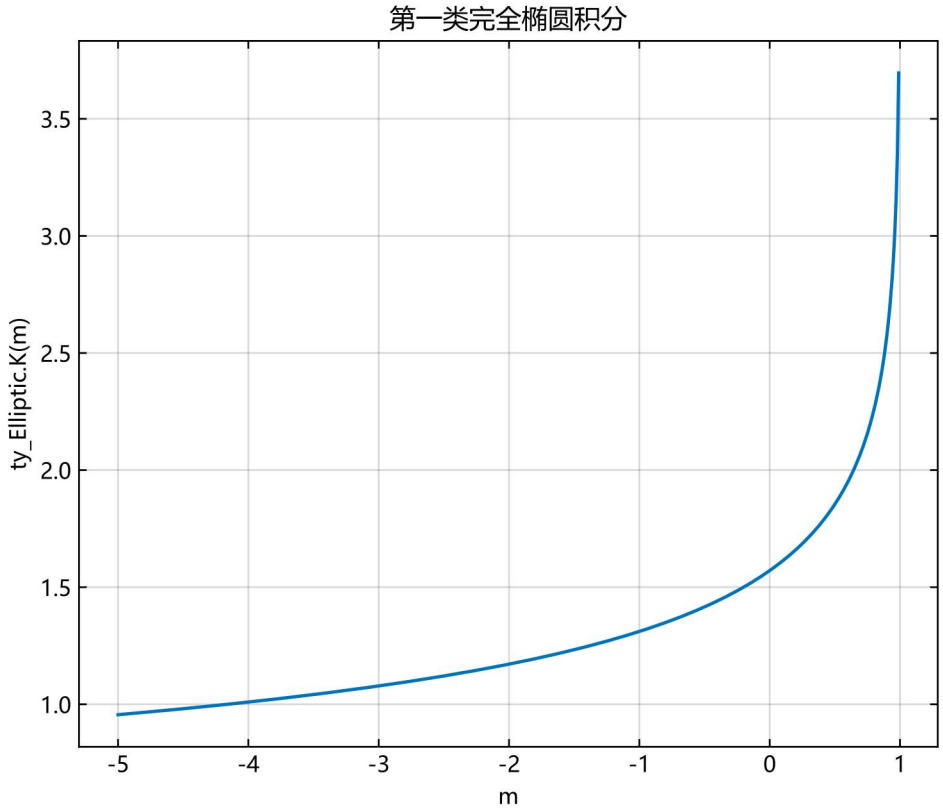
```
s = Elliptic.K.([1/2 pi/4 1 -5.5])
```

```
s = 1×4 Matrix{Float64}:
1.85407   2.22525   Inf   0.932467
```

(2). 绘制第一类完全椭圆积分

绘制第一类完全椭圆积分。

```
x = -5:0.01:1
y = Elliptic.K.(x)
plot(x,y)
grid("on")
title("第一类完全椭圆积分")
xlabel("m")
ylabel("Elliptic.K(m)")
```



Elliptic.Pi

第三类完全和不完全椭圆积分

A. 语法

```
Elliptic.Pi(n,m)
Elliptic.Pi(n,m,phi)
```

B. 说明

Elliptic.Pi(n,m)返回第三类完全椭圆积分。

Elliptic.Pi(n,m,phi) 返回第三类不完全椭圆积分。

C. 示例

(1). 计算第三类不完全椭圆积分

```
s    =    [Elliptic.Pi(-2.3,pi/4,0)        Elliptic.Pi(1/3,pi/3,1/2)
Elliptic.Pi(-1,0,1)  Elliptic.Pi(2,pi/6,2)]
```

```
s = 1×4 Matrix{Float64}:
0.587685  1.28503  0.0  0.750732
```

ellipticCE

第二类互补完全椭圆积分

A. 语法

```
ellipticCE(m)
```

B. 说明

ellipticCE(m)返回第二类互补完全椭圆积分。

C. 示例

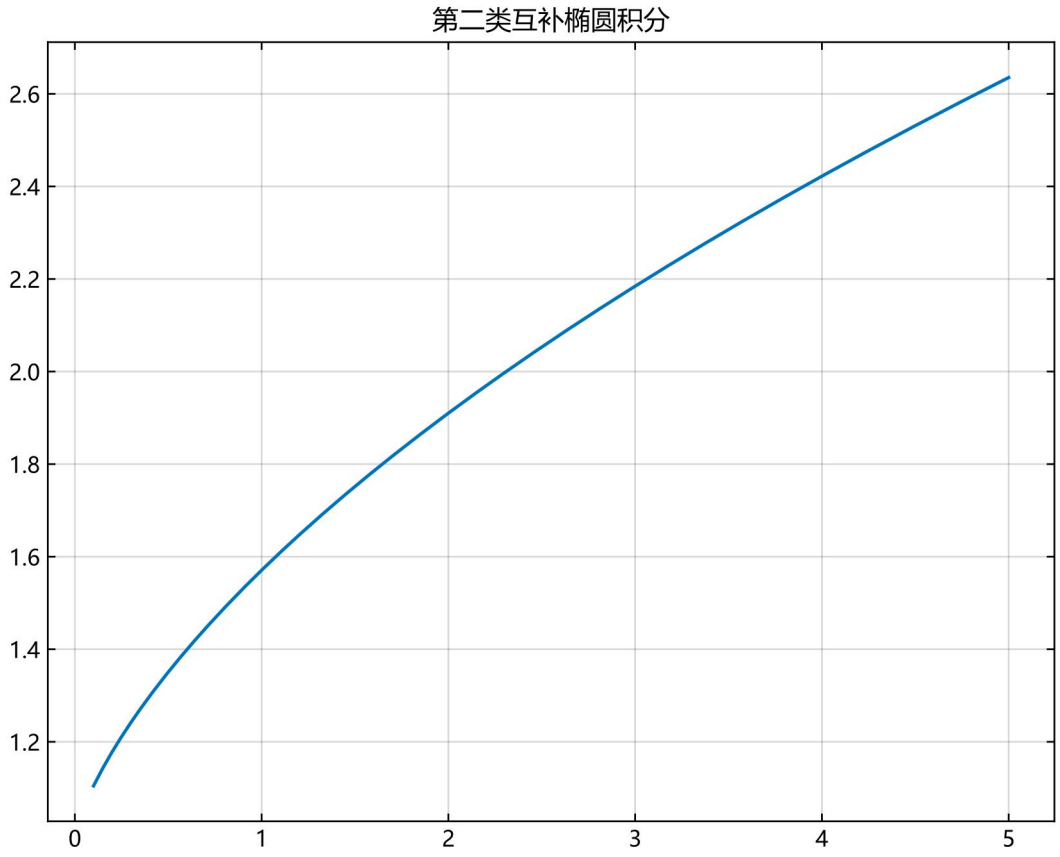
(1). 求第二类互补完全椭圆积分

```
s = ellipticCE.([0 pi/4 1 pi/2])
```

```
1×4 Matrix{Float64}:
1.0  1.48279  1.5708  1.77534
```

(2). 绘制第二类不完全椭圆积分

```
f = x->ellipticCE(x)
fplot(f,[0.1,5])
grid("on")
title("第二类互补椭圆积分")
```



ellipticCK

第一类互补完全椭圆积分

A. 语法

```
ellipticCK(m)
```

B. 说明

ellipticCK(m)返回第一类互补完全椭圆积分。

C. 示例

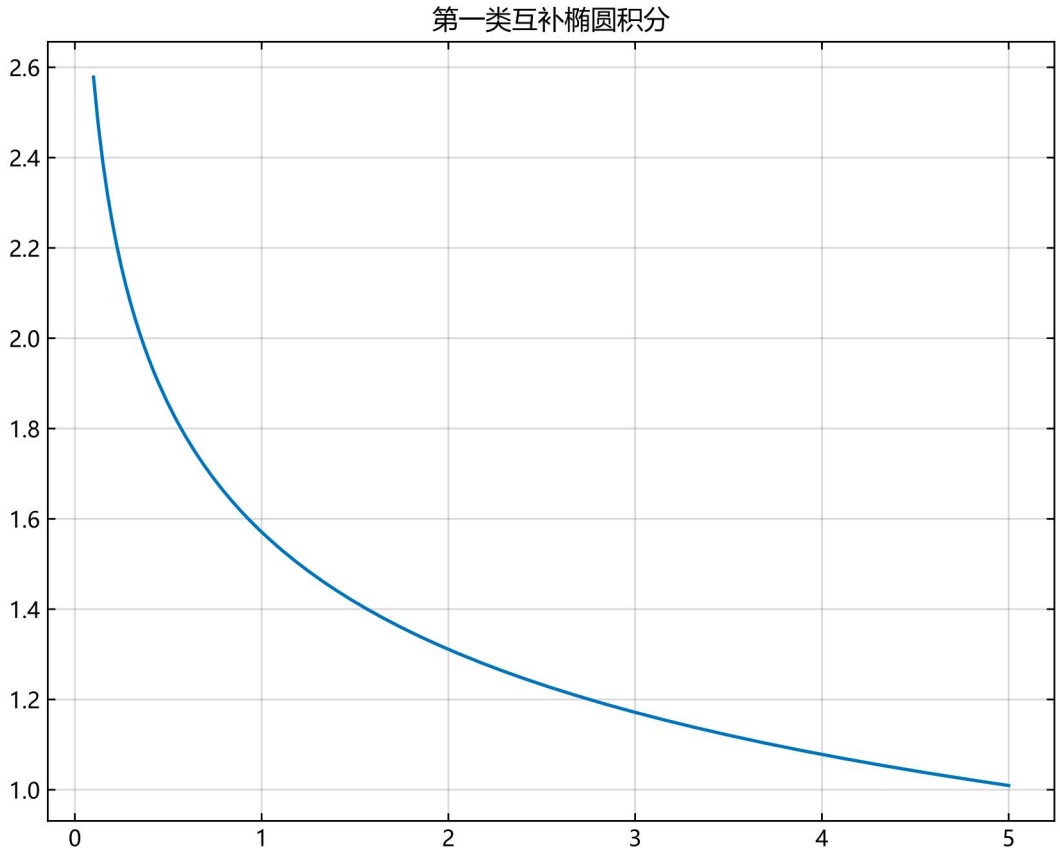
(1). 求第一类互补完全椭圆积分

```
s = ellipticCK.([0 pi/4 1 pi/2])
```

```
1×4 Matrix{Float64}:
Inf  1.66706  1.5708  1.39865
```

(2). 绘制第一类不完全椭圆积分 f

```
= x->ellipticCK(x)
fplot(f,[0.1,5])
grid("on")
title("第一类互补椭圆积分")
```



3.2.1.7.6 Jacobi 椭圆积分

函数名	简介
Jacobi.cd	Jacobi CD 椭圆函数
Jacobi.cn	Jacobi CN 椭圆函数
Jacobi.cs	Jacobi CS 椭圆函数
Jacobi.dc	Jacobi DC 椭圆函数
Jacobi.dn	Jacobi DN 椭圆函数
Jacobi.ds	Jacobi DS 椭圆函数
Jacobi.nc	Jacobi NC 椭圆函数
Jacobi.nd	Jacobi ND 椭圆函数

Jacobi.ns	Jacobi NS 椭圆函数
Jacobi.sc	Jacobi SC 椭圆函数
Jacobi.sd	Jacobi SD 椭圆函数
Jacobi.sn	Jacobi SN 椭圆函数
Jacobi.am	雅克比振幅函数

Jacobi.cd

Jacobi CD 椭圆函数

A. 语法

```
Jacobi.cd(u,m)
```

B. 说明

Jacobi.cd(u,m)返回 u 和 m 的雅可比 CD 椭圆函数。

C. 示例

(1). 计算雅可比 CD 椭圆函数

```
Jacobi.cd(2,1)
```

```
ans = 0.9999999999999998
```

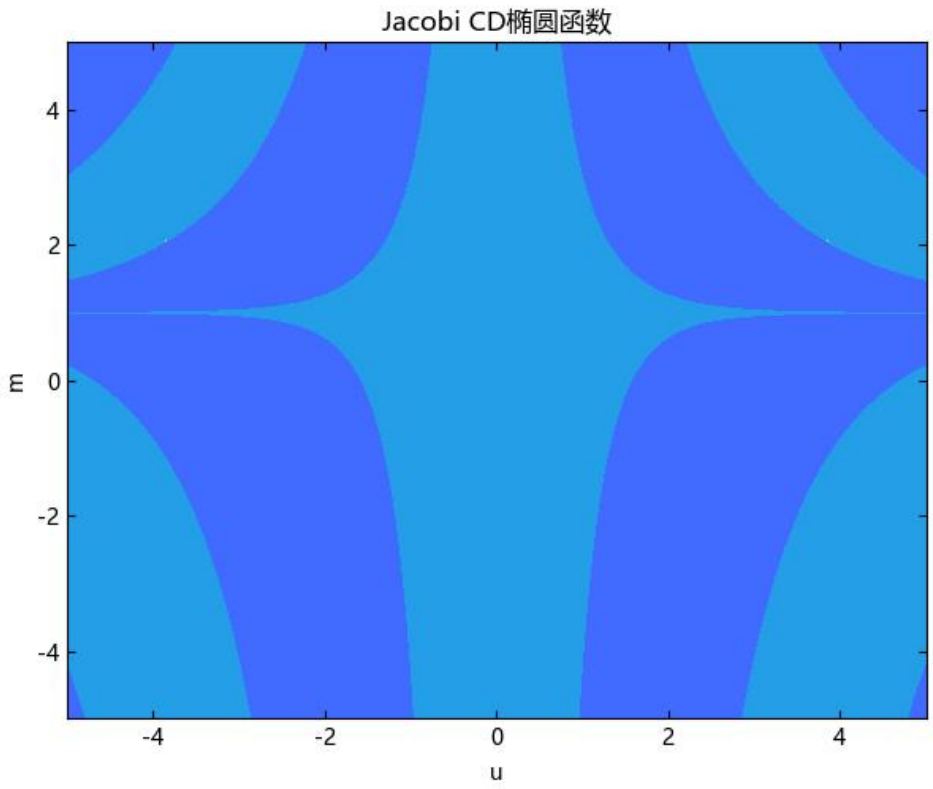
当 u 或 m 是数组时，按元素执行计算 。

```
Jacobi.cd.([2 1 -3],[1 2 3])
```

```
ans = 1×3 Matrix{Float64}:
1.0  2.3829  -178.629
```

(2). 绘制雅可比 CD 椭圆函数图

```
u = -5:0.005:5
m = -5:0.005:5
U,M = meshgrid2(u,m)
Z = Jacobi.cd.(U,M)
contour(U,M,Z,filled = true)
title("Jacobi CD 椭圆函数")
xlabel("u")
ylabel("m")
```



Jacobi.cn

Jacobi CN 椭圆函数

A. 语法

```
Jacobi.cn(u,m)
```

B. 说明

Jacobi.cn(u,m)返回 u 和 m 的雅可比 CN 椭圆函数。

C. 示例

(1). 计算雅可比 CN 椭圆函数

```
Jacobi.cn(2,1)
```

```
ans = 0.26580222883407956
```

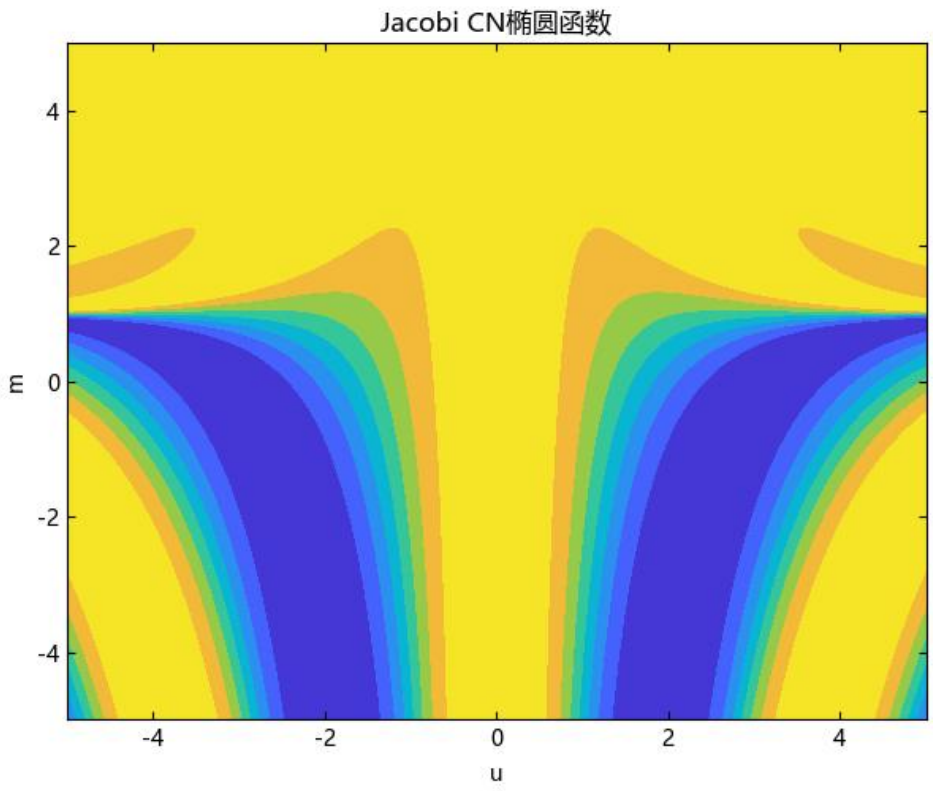
当 u 或 m 是数组时，按元素执行计算 。

```
Jacobi.cn.([2 1 -3],[1 2 3])
```

```
ans = 1×3 Matrix{Float64}:
0.265802  0.740459  0.816501
```

(2). 绘制雅可比 CN 椭圆函数图

```
u = -5:0.005:5
m = -5:0.005:5
U,M = meshgrid2(u,m)
Z = Jacobi.cn.(U,M)
contour(U,M,Z,filled = true)
title("Jacobi CN 椭圆函数")
xlabel("u")
ylabel("m")
```



Jacobi.cs

Jacobi CS 椭圆函数

A. 语法

```
Jacobi.cs(u,m)
```

B. 说明

Jacobi.cs(u,m)返回 u 和 m 的雅可比 CS 椭圆函数。

C. 示例

(1). 计算雅可比 CS 椭圆函数

```
Jacobi.cs(2,1)
```

```
ans = 0.2757205647717831
```

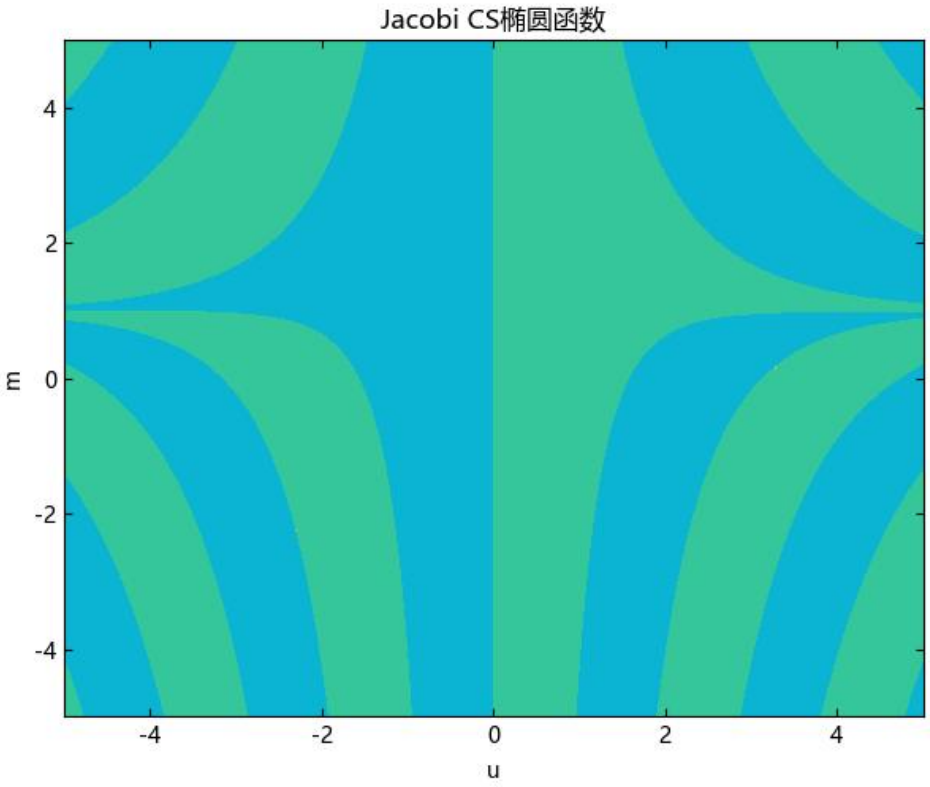
当 u 或 m 是数组时，按元素执行计算 。

```
Jacobi.cs.([2 1 -3],[1 2 3])
```

```
ans = 1×3 Matrix{Float64}:
0.275721  1.10171  1.41424
```

(2). 绘制雅可比 CS 椭圆函数图

```
u = -5:0.005:5
m = -5:0.005:5
U,M = meshgrid2(u,m)
Z = Jacobi.cs.(U,M)
contour(U,M,Z,filled = true)
title("Jacobi CS 椭圆函数")
xlabel("u")
ylabel("m")
```



Jacobi.dc

Jacobi DC 椭圆函数

A. 语法

```
Jacobi.dc(u,m)
```

B. 说明

Jacobi.dc(u,m)返回 u 和 m 的雅可比 DC 椭圆函数。

C. 示例

(1). 计算雅可比 DC 椭圆函数

```
Jacobi.dc(2,1)
```

```
ans = 1.0000000000000002
```

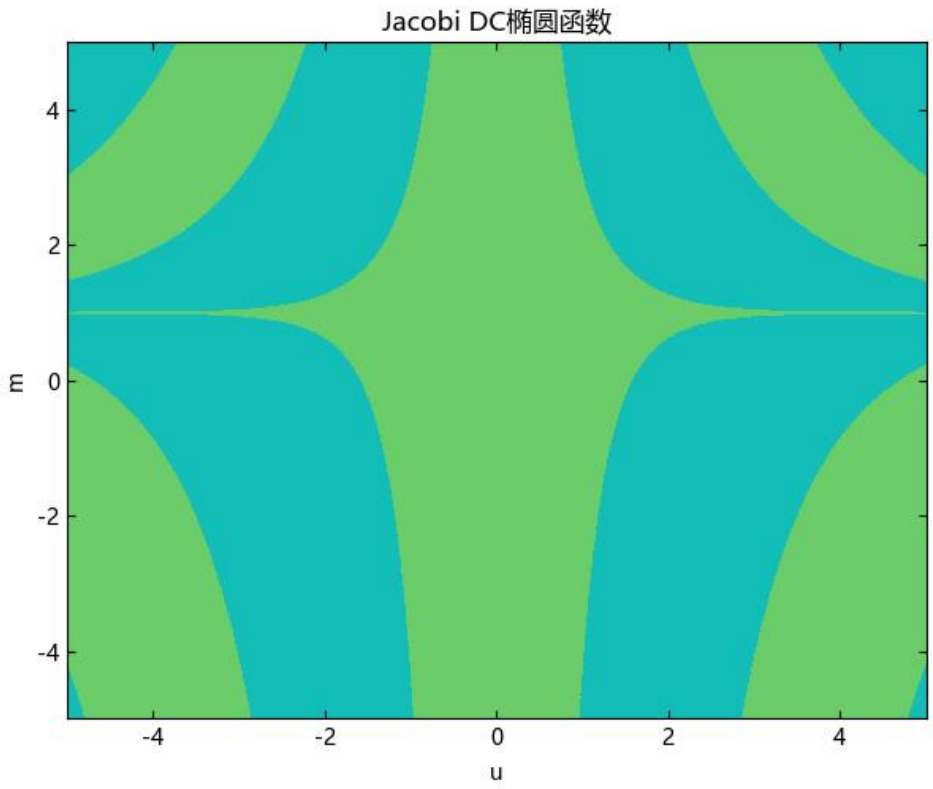
当 u 或 m 是数组时，按元素执行计算 。

```
Jacobi.dc.([2 1 -3],[1 2 3])
```

```
ans = 1×3 Matrix{Float64}:
1.0    0.419656    -0.00559819
```

(2). 绘制雅可比 DC 椭圆函数图

```
u = -5:0.005:5
m = -5:0.005:5
U,M = meshgrid2(u,m)
Z = Jacobi.dc.(U,M)
contour(U,M,Z,filled = true)
title("Jacobi DC 椭圆函数")
xlabel("u")
ylabel("m")
```



Jacobi.dn

Jacobi DN 椭圆函数

A. 语法

```
Jacobi.dn(u,m)
```

B. 说明

Jacobi.dn(u,m)返回 u 和 m 的雅可比 DN 椭圆函数。

C. 示例

(1). 计算雅可比 DN 椭圆函数

```
Jacobi.dn(2,1)
```

```
ans = 0.2658022288340796
```

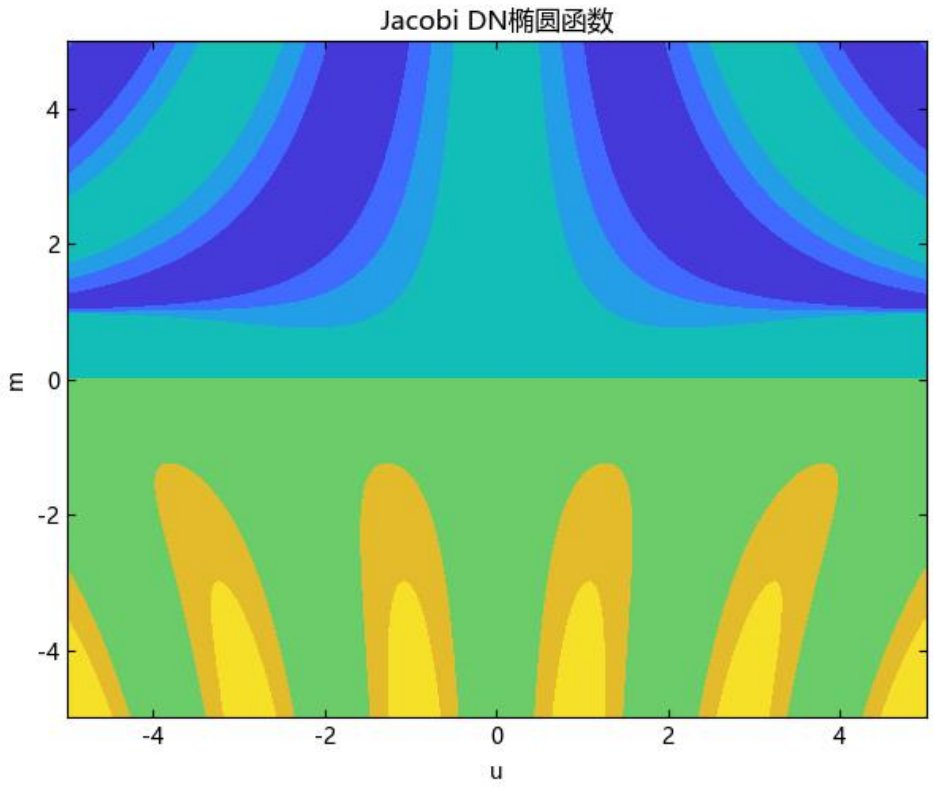
当 u 或 m 是数组时，按元素执行计算 。

```
Jacobi.dn.([2 1 -3],[1 2 3])
```

```
ans = 1×3 Matrix{Float64}:
0.265802    0.310738    -0.00457093
```

(2). 绘制雅可比 DN 椭圆函数图

```
u = -5:0.005:5
m = -5:0.005:5
U,M = meshgrid2(u,m)
Z = Jacobi.dn.(U,M)
contour(U,M,Z,filled = true)
title("Jacobi DC 椭圆函数")
xlabel("u")
ylabel("m")
```



Jacobi.ds

Jacobi DS 椭圆函数

A. 语法

```
Jacobi.ds(u,m)
```

B. 说明

Jacobi.ds(u,m)返回 u 和 m 的雅可比 DS 椭圆函数。

C. 示例

(1). 计算雅可比 DN 椭圆函数

```
Jacobi.ds(2,1)
```

```
ans = 0.27572056477178314
```

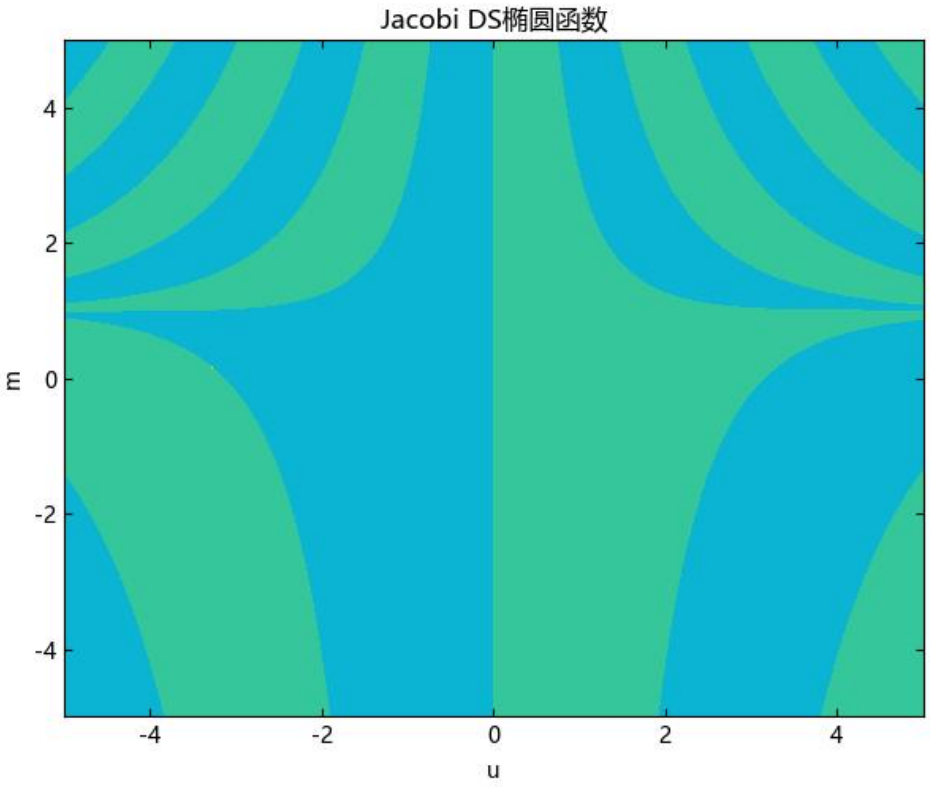
当 u 或 m 是数组时，按元素执行计算 。

```
Jacobi.ds.([2 1 -3],[1 2 3])
```

```
ans = 1×3 Matrix{Float64}:
0.275721    0.462338    -0.00791717
```

(2). 绘制雅可比 DS 椭圆函数图

```
u = -5:0.005:5
m = -5:0.005:5
U,M = meshgrid2(u,m)
Z = Jacobi.ds.(U,M)
contour(U,M,Z,filled = true)
title("Jacobi DS 椭圆函数")
xlabel("u")
ylabel("m")
```



Jacobi.nc

Jacobi NC 椭圆函数

A. 语法

```
Jacobi.nc(u,m)
```

B. 说明

Jacobi.nc(u,m)返回 u 和 m 的雅可比 nc 椭圆函数。

C. 示例

(1). 计算雅可比 nc 椭圆函数

```
Jacobi.nc(2,1)
```

```
ans = 3.762195691083633
```

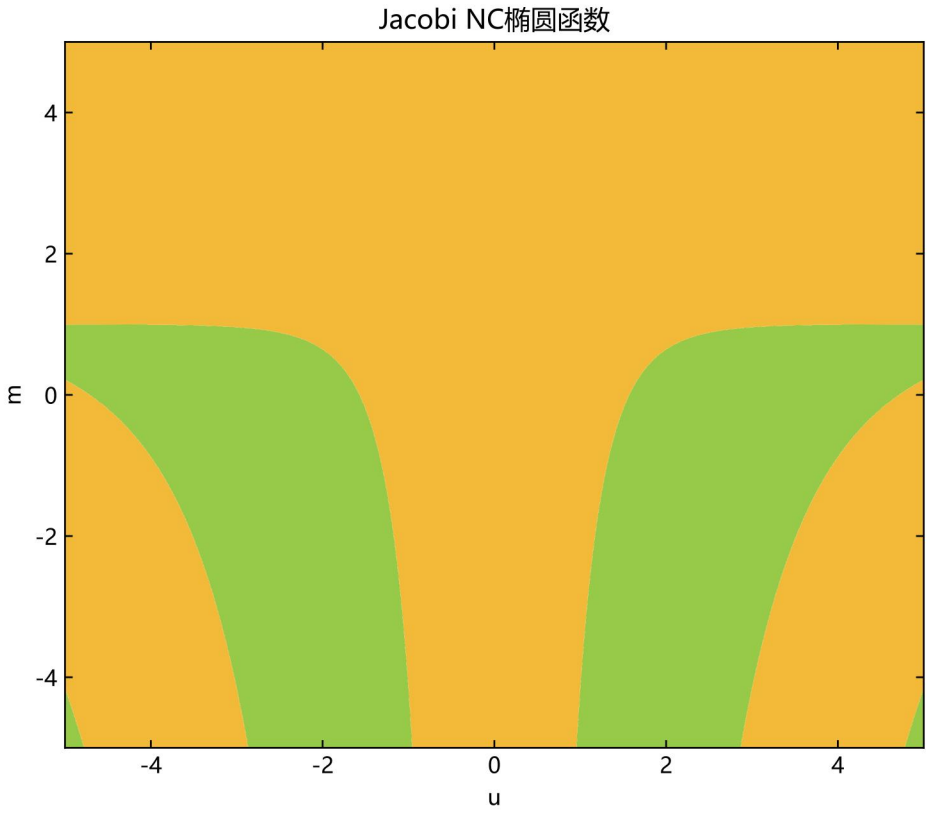
当 u 或 m 是数组时，按元素执行计算 。

```
Jacobi.nc.([2 1 -3],[1 2 3])
```

```
ans = 1×3 Matrix{Float64}:
3.7622    1.35051    1.22474
```

(2). 绘制雅可比 NC 椭圆函数图

```
u = -5:0.005:5
m = -5:0.005:5
U,M = meshgrid2(u,m)
Z = Jacobi.nc.(U,M)
contour(U,M,Z,filled = true)
title("Jacobi NC 椭圆函数")
xlabel("u")
ylabel("m")
```



Jacobi.nd

Jacobi ND 椭圆函数

A. 语法

```
Jacobi.nd(u,m)
```

B. 说明

Jacobi.nd(u,m)返回 u 和 m 的雅可比 ND 椭圆函数。

C. 示例

(1). 计算雅可比 ND 椭圆函数

```
Jacobi.nd(2,1)
```

```
ans = 3.7621956910836327
```

当 u 或 m 是数组时，按元素执行计算 。

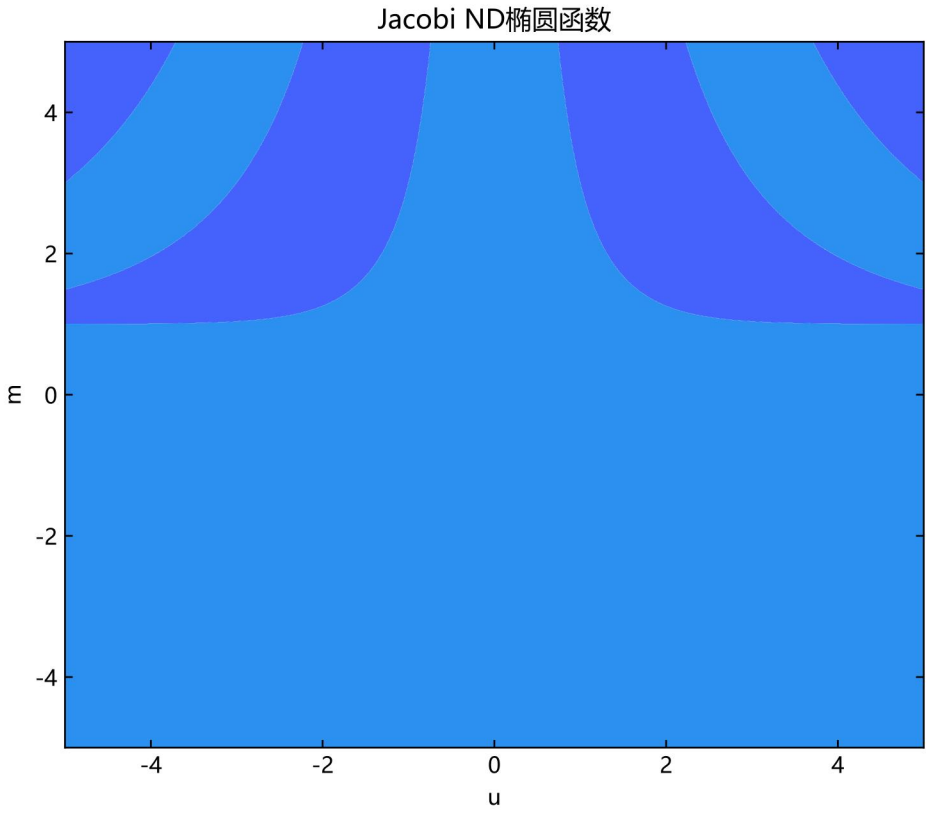
```
Jacobi.nc.([2 1 -3],[1 2 3])
```

```
ans = 1×3 Matrix{Float64}:
```

```
3.7622    3.21815    -218.774
```

(2). 绘制雅可比 ND 椭圆函数图

```
u = -5:0.005:5
m = -5:0.005:5
U,M = meshgrid2(u,m)
Z = Jacobi.nd.(U,M)
contour(U,M,Z,filled = true)
title("Jacobi ND 椭圆函数")
xlabel("u")
ylabel("m")
```



Jacobi.ns

Jacobi NS 椭圆函数

A. 语法

```
Jacobi.ns(u,m)
```

B. 说明

Jacobi.ns(u,m)返回 u 和 m 的雅可比 NS 椭圆函数。

C. 示例

(1). 计算雅可比 NS 椭圆函数

```
Jacobi.nd(2,1)
```

```
ans = 1.0373147207275482
```

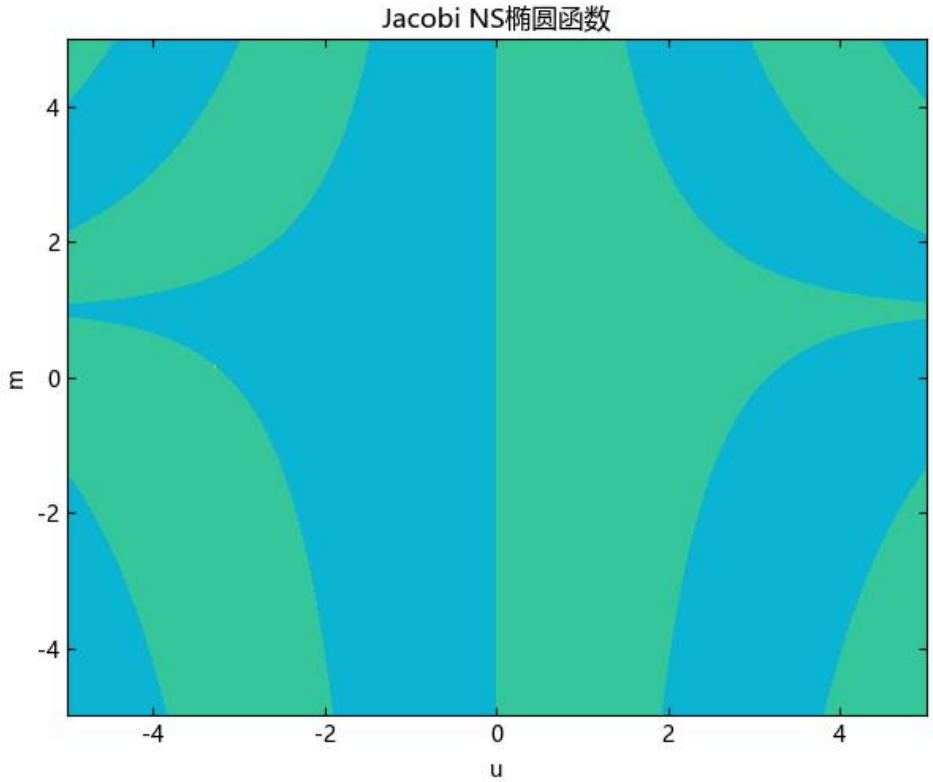
当 u 或 m 是数组时，按元素执行计算 。

```
Jacobi.ns.([2 1 -3],[1 2 3])
```

```
ans = 1×3 Matrix{Float64}:
1.03731   1.48787   1.73207
```

(2). 绘制雅可比 NS 椭圆函数图

```
u = -5:0.005:5
m = -5:0.005:5
U,M = meshgrid2(u,m)
Z = Jacobi.ns.(U,M)
contour(U,M,Z,filled = true)
title("Jacobi NS 椭圆函数")
xlabel("u")
ylabel("m")
```



Jacobi.sc

Jacobi SC 椭圆函数

A. 语法

```
Jacobi.sc(u,m)
```

B. 说明

Jacobi.sc(u,m)返回 u 和 m 的雅可比 SC 椭圆函数。

C. 示例

(1). 计算雅可比 SC 椭圆函数

```
Jacobi.SC(2,1)
```

```
ans = 3.626860407847021
```

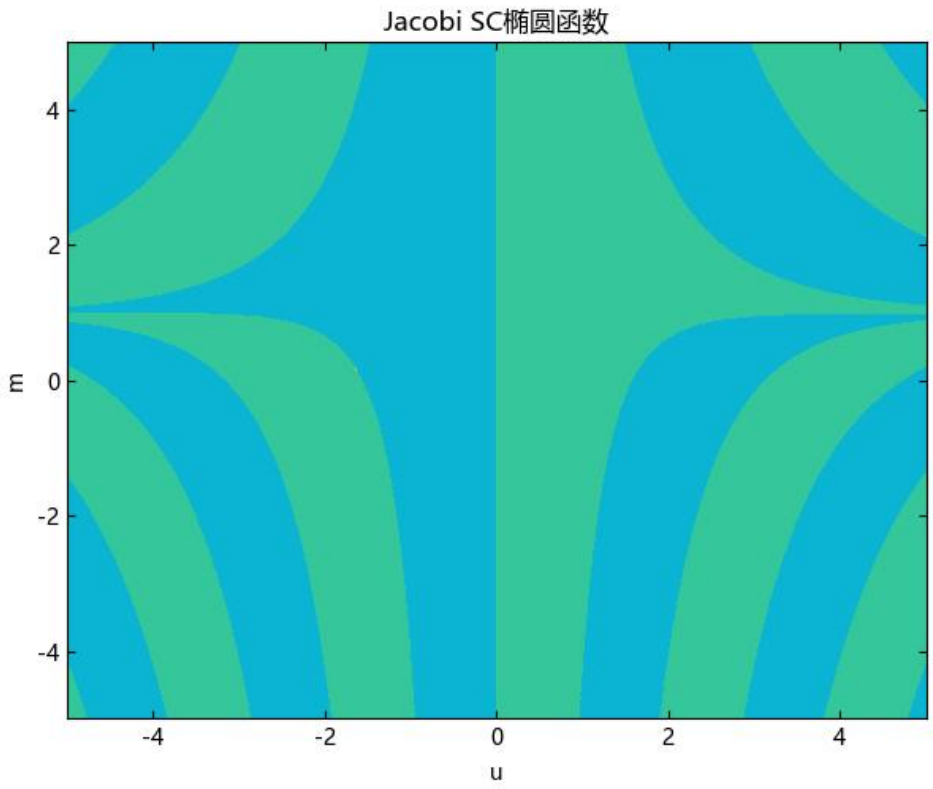
当 u 或 m 是数组时，按元素执行计算 。

```
Jacobi.sc.([2 1 -3],[1 2 3])
```

```
ans = 1×3 Matrix{Float64}:
3.62686    0.907683    0.707096
```

(2). 绘制雅可比 SC 椭圆函数图

```
u = -5:0.005:5
m = -5:0.005:5
U,M = meshgrid2(u,m)
Z = Jacobi.sc.(U,M)
contour(U,M,Z,filled = true)
title("Jacobi SC 椭圆函数")
xlabel("u")
ylabel("m")
```



Jacobi.sd

Jacobi SD 椭圆函数

A. 语法

```
Jacobi.sd(u,m)
```

B. 说明

Jacobi.sd(u,m)返回 u 和 m 的雅可比 SD 椭圆函数。

C. 示例

(1). 计算雅可比 SD 椭圆函数

```
Jacobi.sd(2,1)
```

```
ans = 3.62686040784702
```

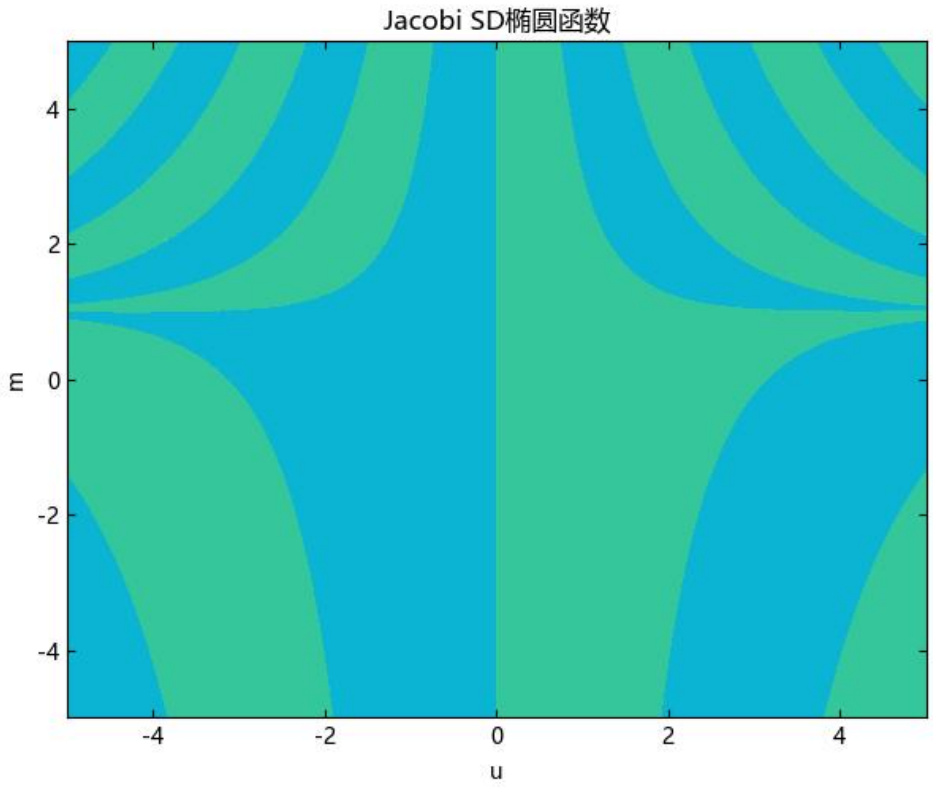
当 u 或 m 是数组时，按元素执行计算 。

```
Jacobi.sd.([2 1 -3],[1 2 3])
```

```
ans = 1×3 Matrix{Float64}:
3.62686   2.16292   -126.308
```

(2). 绘制雅可比 SD 椭圆函数图

```
u = -5:0.005:5
m = -5:0.005:5
U,M = meshgrid2(u,m)
Z = Jacobi.sd.(U,M)
contour(U,M,Z,filled = true)
title("Jacobi SD 椭圆函数")
xlabel("u")
ylabel("m")
```



Jacobi.sn

Jacobi SN 椭圆函数

A. 语法

```
Jacobi.sn(u,m)
```

B. 说明

Jacobi.sn(u,m)返回 u 和 m 的雅可比 SN 椭圆函数。

C. 示例

(1). 计算雅可比 SN 椭圆函数

```
Jacobi.sn(2,1)
```

```
ans = 0.9640275800758169
```

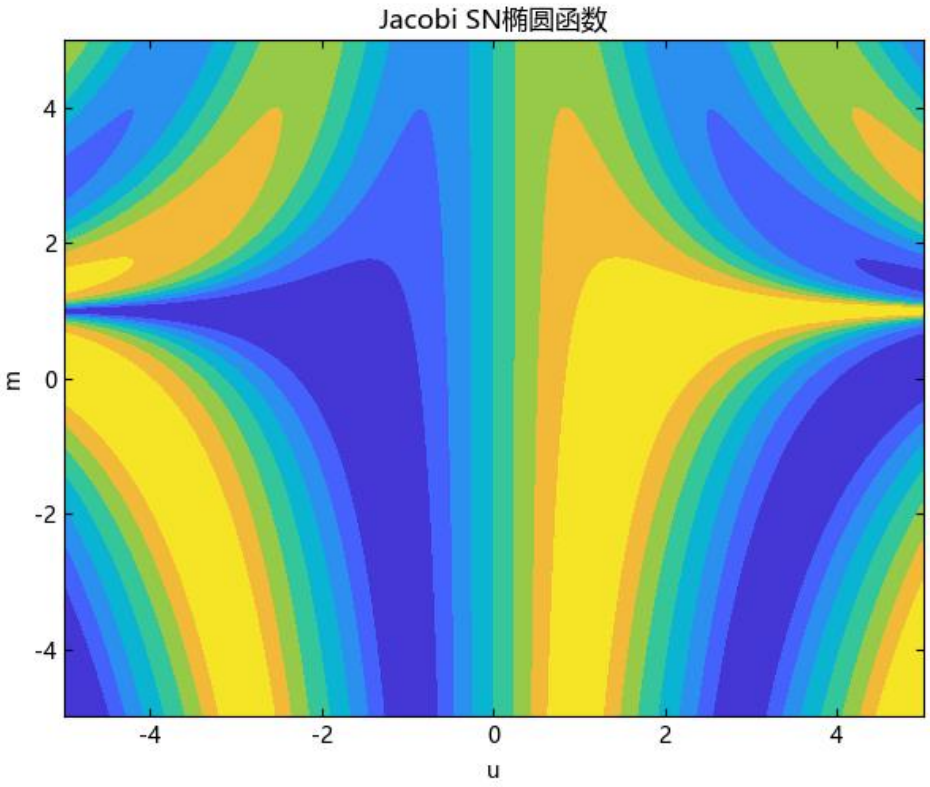
当 u 或 m 是数组时，按元素执行计算 。

```
Jacobi.sn.([2 1 -3],[1 2 3])
```

```
ans = 1×3 Matrix{Float64}:
0.964028    0.672102    0.577344
```

(2). 绘制雅可比 SN 椭圆函数图

```
u = -5:0.005:5
m = -5:0.005:5
U,M = meshgrid2(u,m)
Z = Jacobi.sn.(U,M)
contour(U,M,Z,filled = true)
title("Jacobi SN 椭圆函数")
xlabel("u")
ylabel("m")
```



Jacobi.am

雅克比振幅函数

A. 语法

```
Jacobi.am(u,m)
```

B. 说明

Jacobi.am(u,m)返回 关于 u 和 m 的 Jacobi 振幅函数。如果 u 或 m 是一个数组，则 Jacobi.am 按元素进行操作。

C. 示例

(1). 计算数值输入的雅可比振幅函数

```
Jacobi.am(2,1)
```

```
1.3017603360460153
```

在数组输入上调用 Jacobi.am，当 u 或 m 是数组时，Jacobi.am 按元素进行操作。

```
Jacobi.am.([2 1 -3],[1 2 3])
```

```
ans = 1×3 Matrix{Float64}:
1.30176  0.865769  -3.0
```

3.2.1.7.7 其他特殊函数

函数名	简介
expint	指数积分函数
legendre	连带 Legendre 函数
lambertw	朗博 w 函数 (又称欧米茄函数)
heaviside	单位阶跃函数

expint

指数积分函数

A. 语法

```
Y = expint(z)
Y = expint(ν,z)
```

B. 说明

Y = expint(ν ,z)计算 ν 和 z 的指数积分。

Y = expint(z)默认 ν =1，计算 x 的指数积分。

C. 示例

(1). 计算指数积分

求 X = 1+2i 的指数积分。

```
expint(1+2im)
```

```
ans = -0.12678428559155952 - 0.035081582928186 97im
```

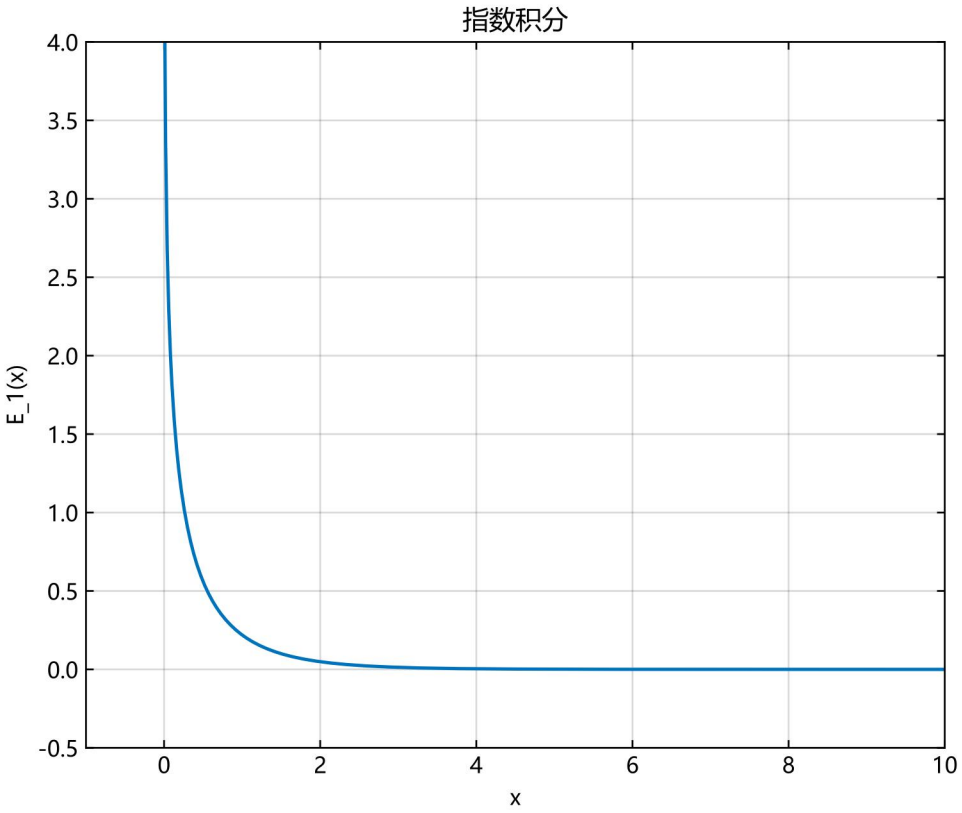
```
expint(2,1+2im)
```

```
-0.09647074593904215 - 0.04586167512795947im
```

(2). 对指数积分绘图

对 [0,10] 区间内 X 的指数积分绘图。

```
X = 0:0.01:10
Y = expint.(X)
plot(X,Y)
axis([-1 10 -0.5 4])
grid("on")
title("指数积分")
xlabel("x")
ylabel(raw"E_1(x)")
```



legendre

连带 Legendre 函数

A. 语法

```
P = legendre(n,X)
P = legendre(n,X,normalization)
```

B. 说明

Y = legendre(n,X)X 中的每个元素计算阶数为 n、级数为 m=0,1,...,n 时的连带 Legendre 函数。

Y = legendre(n,X,normalization) 计算连带 Legendre 函数的归一化版本。normalization 可以是 "unnorm"（默认值）、"sch" 或 "norm"。

C. 示例

(1). 向量的连带 Legendre 函数值

使用 legendre 函数对向量进行运算，然后检查输出的格式。

计算向量的二阶 Legendre 函数值。

```
deg = 2
x = [0,0.1,0.2]
P = legendre(deg,x)
```

```
P = 3×3 Matrix{Float64}:
-0.5      -0.485       -0.44
-0.0    -0.298496   -0.587878
3.0        2.97         2.88
```

输出的格式如下：

每行包含不同 m（连带 Legendre 函数的阶）值的函数值

每列包含不同 x 值的函数值

$$\begin{array}{cccc} & x=0 & x=0.1 & x=0.2 \\ m=0 & P_2^0(0) & P_2^0(0.1) & P_2^0(0.2) \\ m=1 & P_2^1(0) & P_2^1(0.1) & P_2^1(0.2) \\ m=2 & P_2^2(0) & P_2^2(0.1) & P_2^2(0.2) \end{array}$$

二阶连带 Legendre 函数 P_2^m 的方程为

$$P_2^m(x)=(-1)^m(1-x^2)^{m/2}\frac{d^m}{dx^m}[\frac{1}{2}(3x^2-1)].$$

因此， $P_2^0(0)$ 的值为

$$P_2^0(0)=[\frac{1}{2}(3x^2-1)]|_{x=0}=-\frac{1}{2}.$$

该结果与 P(1,1) = -0.5000 一致

(2). 比较 Legendre 归一化

计算几种归一化情况下的连带 Legendre 函数值。

计算非归一化一阶 Legendre 函数值 P_1^m。 值的第一行对应于 m=0，第二行对应于 m=1。

```
x = 0:0.2:1
n = 1
P_unnorm = legendre(n,x)
```

```
 P_unnorm = 2×6 Matrix{Float64}:
0.0        0.2            0.4         0.6     0.8    1.0
-1.0  -0.979796  -0.916515    -0.8   -0.6   -0.0
```

接下来，计算 Schmidt 半归一化函数值。与非归一化值相比，当 m>0 时，

Schmidt 形式的值都乘以缩放因子：

$$(-1)^m \sqrt{\frac{2(n-m)!}{(n+m)!}}.$$

对于第一行，这两个归一化是相同的，因为 m=0。对于第二行，每个值都乘以缩放常量 -1。

```
P_sch = legendre(n,x,"sch")
```

```
P_sch = 2×6 Matrix{Float64}:
0.0       0.2         0.4        0.6   0.8  1.0
1.0  0.979796  0.916515   0.8   0.6  0.0
```

```
C1 = (-1) * sqrt(2 * factorial(0) / factorial(2) )
```

```
C1 = -1.0
```

最后，计算完全归一化的函数值。与非归一化值相比，完全归一化形式的值都乘以缩放因子：

$$(-1)^m \sqrt{\frac{(n+\frac{1}{2})(n-m)!}{(n+m)!}}.$$

此缩放因子适用于 m 的所有值，因此第一行和第二行具有不同缩放因子。

```
P_norm = legendre(n,x,"norm")
```

```
P_norm = 2×6 Matrix{Float64}:
0.0      0.244949  0.489898  0.734847  0.979796  1.22474
0.866025  0.848528  0.793725  0.69282   0.519615  0.0
```

```
Cm0 = sqrt((3/2))
```

```
Cm0 = 1.224744871391589
```

```
Cm1 = (-1) * sqrt((3/2)/2)
```

```
Cm1 = -0.8660254037844386
```

(3). 计算球谐函数

球谐函数出现在拉普拉斯方程的解中，用于表示在球面上定义的函数。使用 legendre 来计算和可视化 Y_3^2 的球谐函数。

球谐函数的方程包括 Legendre 函数的一个项以及一个复指数：

$$Y_l^m(\theta,\phi) = \sqrt{\frac{(2l+1)(l-m)!}{4\pi(l+m)!}} P_l^m(cos\theta) e^{im\phi}$$

首先，创建一个由值组成的网格，以表示 0≤ θ ≤π （余纬度角）和 0≤ ϕ≤2π（方位角）的所有组合。此处，余纬度 θ 的范围从 0 度（位于北极）

到 π/2 度（位于赤道），到 π 度（位于南极）。

```
dx = pi/60;
col = 0:dx:pi;
az = 0:dx:2*pi;
phi,theta = meshgrid2(az,col);
```

计算 l=3 的网格上的 $P_l^m cos(\theta)$

```
l = 3;
Plm = legendre(l,cos.(theta));
```

由于 legendre 为 m 的所有值计算答案，因此 Plm 会包含一些额外的函数值。提取 m=2 的值并丢弃其余值。使用 reshape 函数将结果定向为与 phi 和 theta 具有相同大小的矩阵。

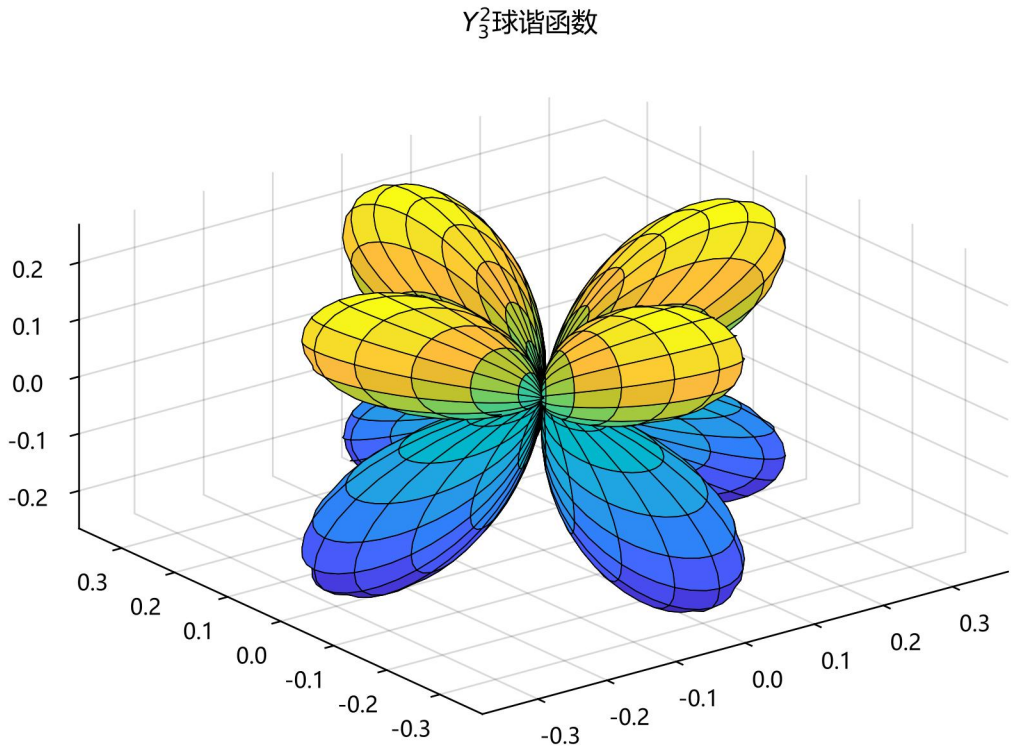
```
m = 2;
if l != 0
Plm = reshape(Plm[m+1,:,:],size(phi))
end
```

计算 的球谐函数值。

```
a = (2*l+1)*factorial(l-m);
b = 4*pi*factorial(l+m);
C = sqrt(a/b);
Ylm = C .* Plm .*exp.(im*m*phi);
```

将球面坐标转换为笛卡尔坐标。此处，π/2−θ 成为纬度角，范围从 π/2（位于北极），到 0 度（位于赤道），再到 −π/2 度（位于南极）。使用正负实数值绘制 Y_3^2 的球谐函数。

```
Xm,Ym,Zm = sph2cart(phi,pi/2 .- theta,abs.(real.(Ylm)))
surf(Xm,Ym,Zm)
title(raw"$Y_3^2$球谐函数")
```



lambertw

朗博 w 函数 (又称欧米茄函数)

A. 语法

```
P = lambertw(x)
P = lambertw(x,k)
```

B. 说明

Y = lambertw(x) 返回 LambertW 函数的主分支。此语法等效于 lambertw(0,x)。

Y = lambertw(x,k)求 Lambert W 函数的 第 k 个分支。仅当 k = 0 或 k = -1 时，此语法才返回实数值。

C. 示例

(1). 计算 Lambertw 函数

```
A = [0 -1/exp(1); pi im]
lambertw.(A)
```

```
ans = 2×2 Matrix{ComplexF64}:
0.565608+0.0im      -1.0+0.0im
3.77497+0.0im    1.18639+1.50459im
```

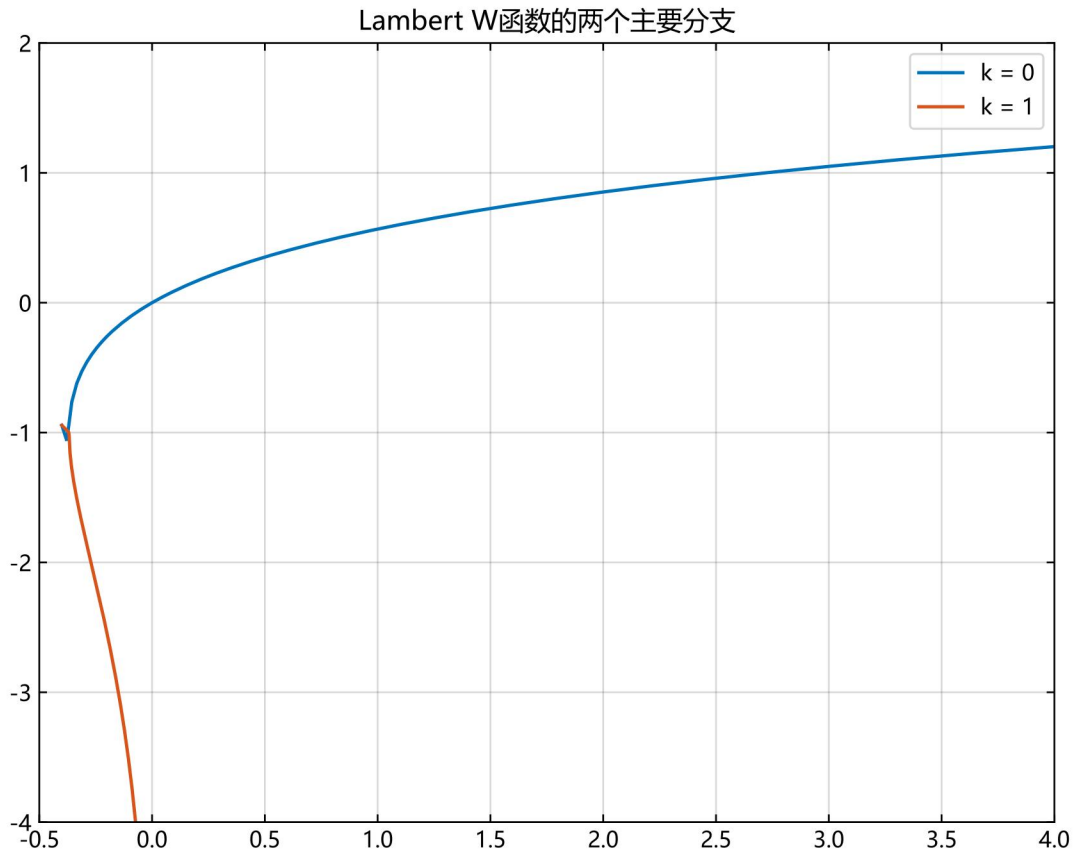
```
lambertw.(A,-1)
```

```
ans = 2×2 Matrix{ComplexF64}:
-Inf+0.0im             -1.0-1.33393e-8im
-0.390974-4.62811im     -1.08965-2.76636im
```

(2). 绘制 Lambert W 函数的两个主要分支

绘制两个主要分支: Lambertw $W_{(0)}$ 和 $W_{-1}(x)$

```
@variables x
fplot(x->lambertw(complex(x)),[-0.4,4])
hold("on")
fplot(x->lambertw(complex(x),-1),[-0.4,0])
hold("off")
axis([-0.5 4 -4 2])
grid("on")
title("Lambert W 函数的两个主要分支")
legend(["k = 0","k = 1"])
```

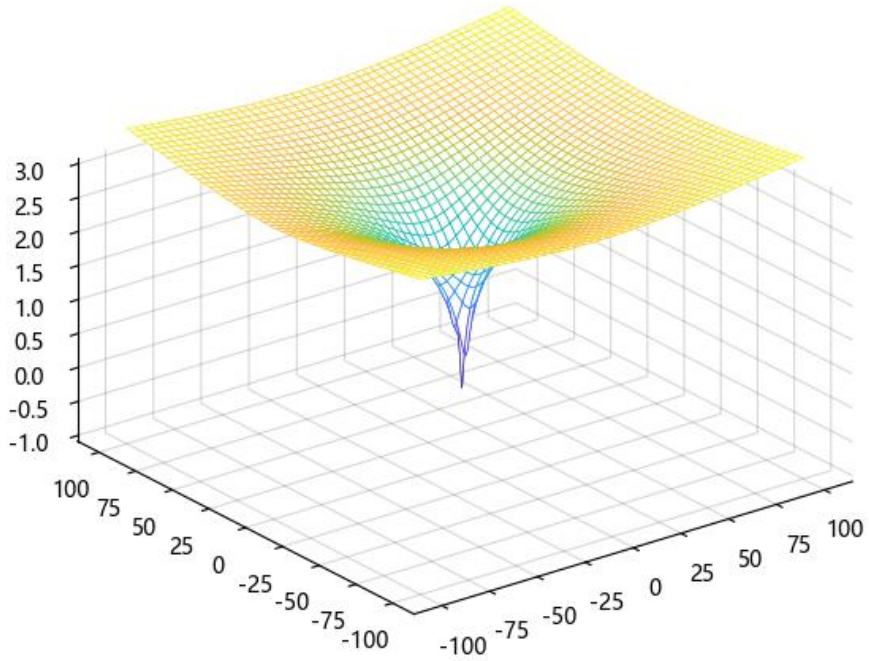


(3). 复平面上的 Lambert W 函数图

在复平面上绘制 Lambert W 函数的主分支。

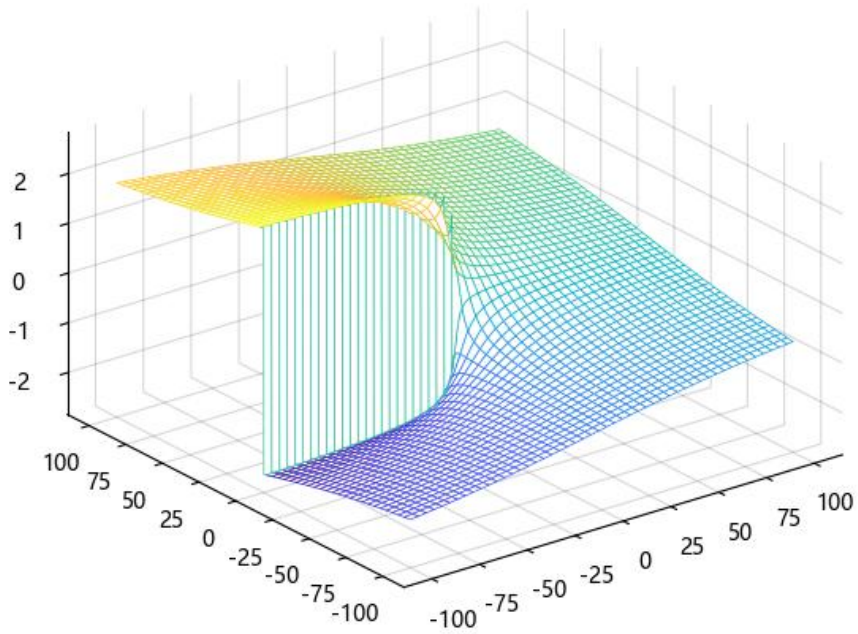
绘制 lambertw 实部值

```
X,Y = meshgrid2(-100:0.1:100)
Z = lambertw.(X+im*Y)
mesh(X,Y,real.(Z))
zlim([-1,3])
```



绘制 lambertw 虚部值

```
mesh(X,Y,imag.(Z))
```



heaviside

单位阶跃函数

A. 语法

```
H = heaviside(x)
```

B. 说明

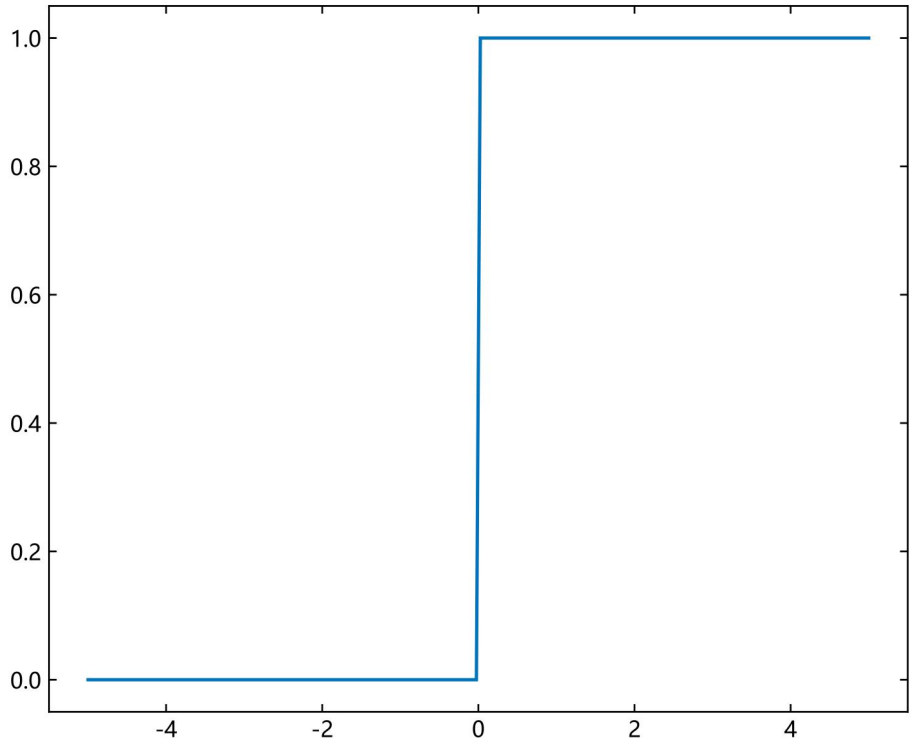
H = heaviside(x) 在 x 处计算 Heaviside 阶跃函数（也称为单位阶跃函数）。该函数是非连续函数，对 x < 0 返回 0，对 x = 0 返回 1/2，对 x > 0 返回 1。

C. 示例

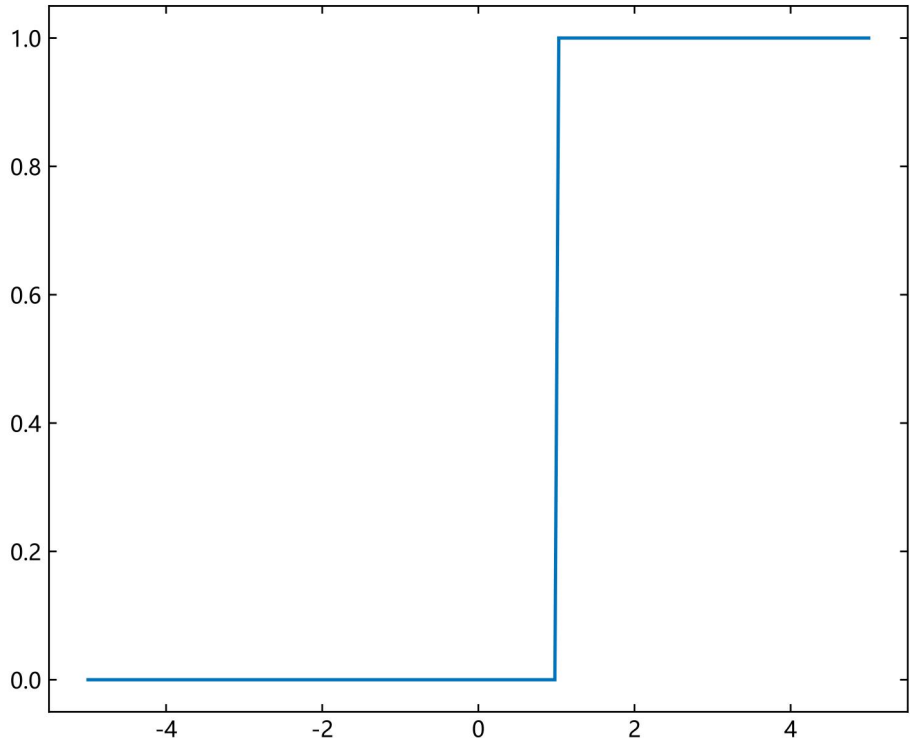
(1). 绘制 Heaviside 函数

对 x 和 x-1 绘制 Heaviside 函数

```
f = x->heaviside(x)
fplot(f)
```




```
fplot(x->heaviside(x-1))
```



3.2.1.8 常量和测试矩阵

Pi、非数字、无穷；Hadamard 矩阵、伴随矩阵、帕斯卡矩阵和其他专用矩阵。

函数名	简介
eps	浮点相对精度
Inf	创建所有值均为 Inf 的数组
pi	圆的周长与其直径的比率
NaN	创建所有值均为 NaN 的数组
isfinite	确定哪些数组元素为有限
isinf	确定哪些数组元素为无限值
isnan	确定哪些数组元素为 NaN
compan	伴随矩阵
TestArrays	测试矩阵生成模块

gallery	测试矩阵
hadamard	Hadamard 矩阵
hankel	Hankel 矩阵
hilb	Hilbert 矩阵
invhilb	Hilbert 矩阵的逆矩阵
magic	幻方矩阵
pascal	帕斯卡矩阵
rosser	典型对称特征值测试问题
toeplitz	托普利茨矩阵
vander	Vandermonde 矩阵
wilkinson	Wilkinson 的特征值测试矩阵

3.2.1.8.1 eps

浮点相对精度

A. 语法

```
d = eps()
d = eps(x)
d = eps(datatype)
```

B. 说明

d = eps() 返回从 1.0 到下一个较大双精度数的距离，即 2^-52.

d = eps(x)（其中，x 的数据类型为 Float）返回从 abs(x) 到与 x 精度相同的下一个较大浮点数的正距离。命令 eps(1.0) 等同于 eps。

d = eps(datatype) 根据 datatype 指定的数据类型返回 eps。

C. 示例

(1). Float64 值的精度

显示从 1.0 到下一个最大双精度数的距离。

```
d = eps()
```

```
d = 2.220446049250313e-16
```

计算 log2(eps)。

```
d = log2(eps())
```

```
d = -52.0
```

底数为 2 时，eps 等于 2^-52。

求从 10.0 到下一个最大双精度数的距离。

```
d = eps(10.0)
```

```
d = 1.7763568394002505e-15
```

(2). Float32 值的精度

显示从 1.0 到下一个最大单精度数的距离。

计算 e^(i*pi)的值

```
d = eps(Float32)
```

```
d = 1.1920929f-7
```

计算 log2(eps(Float32))

```
log2(eps(Float32))
```

```
ans = -23.0f0
```

底数为 2 时，Float16 值 eps 等于 2^-23。

求从 10.0 的单精度表示形式到下一个最大单精度数的距离。

```
d = eps(Float32(10.0))
```

```
d = 9.536743f-7
```

3.2.1.8.2 Inf

创建所有值均为 Inf 的数组

A. 语法

```
X=Inf/0
X=Inf*(A)
```

B. 说明

X = Inf/0 返回正无穷大 Inf，当反斜杠后输入不同的数值值时得到不同的结果。

X = Inf*(A) 若输入的为数组则返回与 A 维度相同的，所有值均为 Inf 的数组。若输入的为正数返回正无穷 Inf，若输入负数则返回负无穷-Inf，输入零时返回 NaN。

C. 示例

(1). 返回不同结果

```
X1=Inf/0
X2=Inf/(-0)
X3=Inf/(0.0)
X4=Inf/(-0.0)
X5=Inf/Inf
return [X1;X2;X3;X4;X5]
```

```
5-element Vector{Float64}:
Inf
Inf
Inf
-Inf
NaN
```

Inf 称为正无穷大，大于所有有限浮点值，-Inf 称为负无穷大，低于所有有限浮点值，NaN 不是数字，不指向任何浮点值（包括自身）。

(2). 返回所有值均为 Inf 的数组

创建一个 3 × 3 的矩阵，利用 Inf 返回所有值均为 Inf 的数组

```
A=ones(3,3)
X=Inf.*(A)
```

```
3×3 Matrix{Float64}:
Inf   Inf   Inf
Inf   Inf   Inf
Inf   Inf   Inf
```

3.2.1.8.3 pi

圆的周长与其直径的比率

A. 语法

```
p = pi
```

B. 说明

p = pi 以 IEEE® 双精度形式返回最接近 π 值的浮点数。示例

C. 示例

(1). Pi 的值。

以双精度形式返回 π 值，小数点后有 15 位。

```
p = pi
```

```
p = 3.141592653589793
```

(2). 圆的面积。

计算半径为 7 的圆的面积。计算半径为 7 的圆的面积。

```
r = 7
A = pi*r^2
```

```
A = 153.93804002589985
```

(3). 球体的表面积和体积。

计算半径为 5 的球体的表面积和体积。

```
r = 5
SA = 4*pi*r^2
```

```
SA = 314.1592653589793
```

```
V = 4/3*pi*r^3
```

```
V = 523.5987755982989
```

3.2.1.8.4 NaN

Float64 类型的非数字值

A. 语法

```
d = NaN
```

B. 说明

Float64 类型的非数字值。

C. 示例

(4). Float64 类型的非数字值创建 Float64 类型的非数字值。

```
d = NaN
```

```
d = NaN
```

计算 log2(NaN)。

```
d = log2(NaN)
```

```
d = NaN
```

3.2.1.8.5 isfinite

确定哪些数组元素为有限

A. 语法

```
TF = isfinite(A)
```

B. 说明

TF = isfinite(A) 返回一个逻辑数组，其中的 1 (true) 对应数组 A 的有限元素，0 (false) 对应无限或 NaN 元素。如果 A 包含复数，则 isfinite(A) 中的 1 对应实部和虚部均为有限值的元素，0 对应实部或虚部中存在无限值或 NaN 值的元素。

C. 示例

(1). 确定有限实数元素

创建一个行向量并确定有限实数元素。

```
A = 1 ./[-2, -1, 0, 1, 2]
```

```
5-element Vector{Float64}:
-0.5
-1.0
Inf
1.0
0.5
```

```
TF = isfinite.(A)
```

```
5-element BitVector:
1
1
0
1
1
```

(2). 确定有限复数元素

创建一个复数数组。确定这些复数中是否包含有限值。

```
A = [2+1im ,3+1im/0 ,0/0-2im]
```

```
3-element Vector{ComplexF64}:
2.0 + 1.0im
NaN + Inf*im
NaN - 2.0im
```

```
TF = isfinite.(A)
```

```
3-element BitVector:
1
0
0
```

A 的第二个元素具有无限虚部。第三个元素具有 NaN 实部。对于这两个元素，isfinite 返回 0。

3.2.1.8.6 isinf

确定哪些数组元素为无限值

A. 语法

```
TF = isinf(A)
```

B. 说明

TF = isinf(A) 返回一个逻辑数组，其中的 1 (true) 对应数组 A 中的 Inf 或 -Inf 元素，0 (false) 对应其他元素。如果 A 包含复数，则 isinf(A) 中的 1 对应实部或虚部为无限值的元素，0 对应实部和虚部均为有限值或 NaN 值的元素。

C. 示例

(1). 确定无限实数元素

创建一个行向量并确定无限元素。

```
A = 1 ./[-2, -1, 0, 1, 2]
```

```
5-element Vector{Float64}:
-0.5
-1.0
Inf
1.0
0.5
```

```
TF = isinf.(A)
```

```
5-element BitVector:
0
0
1
0
0
```

(2). 确定无限复数元素

创建一个复数数组。确定这些复数中是否包含无限值。

```
A = [2+1im ,3+1im/0 ,0/0-2im]
```

```
3-element Vector{ComplexF64}:
2.0 + 1.0im
NaN + Inf*im
NaN - 2.0im
```

```
TF = isinf.(A)
```

```
3-element BitVector:
0
1
0
```

A 的第二个元素具有无限虚部。对于此元素，isinf 返回 1。

3.2.1.8.7 isnan

确定哪些数组元素为 NaN

A. 语法

```
TF = isnan(A)
```

B. 说明

TF = isnan(A) 返回一个逻辑数组，其中的 1 (true) 对应 A 中的 NaN 元素，0 (false) 对应其他元素。如果 A 包含复数，则 isnan(A) 中的 1 对应实部或虚部为 NaN 值的元素，0 对应实部和虚部均非 NaN 值的元素。

C. 示例

(1). 确定实数中的 NaN 元素

创建行向量，并确定哪些元素为 NaN。

```
A = 0 ./[-2 ,-1 ,0 ,1 ,2]
5-element Vector{Float64}:
-0.0
-0.0
NaN
0.0
0.0
TF = isnan.(A)
5-element BitVector:
0
0
1
0
0
```

(2). 确定复数中的 NaN 元素

创建一个复数数组。确定这些复数中是否包含 NaN。

```
A = [2 + 1im, 1/0 + 3im, 1/2 - 1im*NaN]
3-element Vector{ComplexF64}:
2.0 + 1.0im
Inf + 3.0im
NaN + NaN*im
TF = isnan.(A)
3-element BitVector:
0
0
1
```

(3). 替换 NaN 元素

创建数组并找到具有 NaN 值的元素。

```
A = [1,3,5,7,NaN,10,NaN,4,6,8]
10-element Vector{Float64}:
1.0
3.0
5.0
7.0
NaN
10.0
NaN
4.0
```

```
6.0
8.0
```

```
TF = isnan.(A)
```

```
10-element BitVector:
0
0
0
0
1
0
1
0
0
0
```

使用 TF 对 A 进行索引，以访问 A 中的 NaN 元素。将 NaN 值替换为 0。

```
A[TF] .= 0
```

```
A = 10-element Vector{Float64}:
1.0
3.0
5.0
7.0
0.0
10.0
0.0
4.0
6.0
8.0
```

3.2.1.8.8 compan

伴随矩阵

A. 语法

```
A = compan(u)
```

B. 说明

A = compan(u) 返回第一行为 -u(2:n)/u(1) 的对应伴随矩阵，其中 u 是多项式系数向量。compan(u) 的特征值是多项式的根。

C. 示例

(1). 多项式的伴随矩阵

计算与多项式 (x-1)(x-2)(x+3) = x^3-7x+6 对应的伴随矩阵。

```
u = [1,0,-7,6]
A = compan(u)
```

```
3×3 Matrix{Float64}:
0.0  7.0  -6.0
1.0  0.0   0.0
0.0  1.0   0.0
```

A 的特征值是多项式的根

```
eigvals(A)
```

```
3-element Vector{Float64}:
-3.000000000000002
1.0000000000000002
2.0000000000000018
```

3.2.1.8.9 TestArrays

测试矩阵生成模块

测试矩阵是在生产生活以及理论研究中所遇到的特殊矩阵。

A. 语法

```
A = compan(u)
```

B. 说明

TestArrays 模块中的函数以 TestArrays.函数名 形式进行调用，例如 TestArrays.cauchy 调用该模块的 cauchy 函数。

TestArrays 中的函数大多数包含 T（生成数据类型） 作为其第一个可选参数，该参数默认来自于对于用户输入的推断，例如，如果要生成以 Float64 为元素类型的二项分布矩阵，应写为 TestArrays.binomial(Float64,...)。注意，部分函数的生成即使对于全整数输入也会生成浮点数，对于这些函数，指定 T 为 Integer 类型的子类型就是不可行的，这些函数见下表。

函数名称	原因
cauchy,chebspec,cycol,house,kahan, invol,lesp,lotkin,orthog parter,qmult,randcolu,randcorr, randhess,r andjorth,randsvd,ris sampling,smoke,toeppd,wathen, wilk	矩阵生成时产生浮点数
dorr,forsythe,kms,lauchli,prolate	函数默认参数存在浮点数

TestArrays 中的部分矩阵生成过程中会使用随机算法，对于所有牵涉到随机算法的函数都支持随机数种子 rng 作为其第二个可选参数，该参数默认设置随机数种子为全局随机数流。例如，对于 randcolu 函数，就可以指定随机数种子，而对于其他不涉及到随机数算法的函数，指定随机数种子是不可行的。可以指定随机数种子的函数见下表。

函数名称	随机数种子作用
cycol,qmult,randcolu,randcorr,randjorth,r ando,randsvd,wathen	矩阵生成时使用随机数
krylov,randhess,toeppd	使用随机数生成输入参数

在同时指定两个可选参数时，必须保证生成数据类型 T 在随机数种子 rng 之前，例如 TestArrays.rando(Float64,MT19937ar(1)) 为随机整数矩阵指定了数据类型 Float64 与随机数种子 MT19937ar(1)。

C. 内容

该模块包含以下函数作为其模块子函数。

binomial	描述：二项矩阵，它是对合矩阵的倍数 语法： A = TestArrays.binomial(n) 返回一个 n×n 矩阵，其中包含整数项，满足 A^2 = 2^(n-1)*eye(n)。 属性： 矩阵 B = A*2^((1-n)/2) 是对合矩阵（矩阵与其逆矩阵相同）。
cauchy	描述：柯西矩阵 语法： A = TestArrays.cauchy(x,y) 返回一个 n×n 矩阵，其中包含的项满足 A[i,j]

	= 1/(x[i]+y[j])。参数 x 和 y 是长度为 n 的向量。如果您为 x 和 y 传入标量，它们将解释为向量 1:x 和 1:y。 A = TestArrays.cauchy(x) 返回与上述 y = x 的情况相同的结果。换言之，A[i,j] = 1/(x[i]+x[j])。 属性： 柯西矩阵的逆矩阵和行列式的显式公式是已知的。 如果 x 和 y 均具有不同元素，则行列式 det(C) 为非零值。 如果 0 < x[1] < ... < x[n] 且 0 < y[1] < ... < y[n]，则 C 为完全正值。
chebspec	描述：Chebyshev 谱微分矩阵 语法： A = TestArrays.chebspec(n,k) 返回一个大小为 n×n 的 Chebyshev 谱微分矩阵。参数 k 可以取值 0（默认值）或 1，它决定输出矩阵的字符。 对于 k = 0（无边界条件），A 是幂零矩阵，这意味着存在满足 A^c = 0 的正整数 c。矩阵 A 具有空向量 ones(n)。 对于 k = 1，A 是非奇异且条件设置良好的矩阵，其特征值具有负实部。 属性： 对于 k = 0，矩阵 A 与具有特征值零、大小为 n 的 Jordan 块类似。对于两个矩阵 A 和 B，如果存在一个具有相同大小的可逆矩阵 P 满足 B = inv(P)*A*P，则将这两个矩阵称为相似矩阵。 对于这两种 k 值，Chebyshev 谱微分矩阵的特征向量矩阵是病态的。
chebvand	描述：Chebyshev 多项式的 Vandermonde 类矩阵 语法： A = TestArrays.chebvand(x) 基于由点 x 组成的向量生成原始 Chebyshev Vandermonde 矩阵，这些点定义计算 Chebyshev 多项式的位置。参数 x 是长度为 n 的向量，A 的大小为 n×n。A 中的项满足 A[i,j] = Ti – 1(x[j])，其中 Ti – 1 是 i - 1 次第一类 Chebyshev 多项式。如果 x 为标量，则将使用区间 [0,1] 上的 x 个等间距点计算 A。 A = TestArrays.chebvand(m,x) 生成上述矩阵的矩形版本（具有 m 行），其中 m 为标量。
chow	描述：奇异的托普利茨下 Hessenberg 矩阵 语法： A = TestArrays.chow(n,alpha,delta) 返回一个阶数为 n 的 Chow 矩阵，它是 n×n 的下 Hessenberg 矩阵。矩阵 A 定义为

	A=Hα+δI 其中 Hα 的矩阵元素为 Hα[i,j] = α(i – j + 1)（在 (i – j + 1) ≥ 0 时），δ 是系数，I 是 n×n 单位矩阵。 A = TestArrays.chow(n) 分别对 alpha 和 delta 使用默认值 1 和 0。 属性： Hα 具有等于零的 p = floor(n/2) 特征值。其余特征值等于 4*alpha*cos(k*pi/(n+2))^2，其中 k = 1:(n-p)。
circul	描述：循环矩阵 语法： A = TestArrays.circul(v) 返回一个 n×n 循环矩阵，其第一行是长度为 n 的向量 v。循环矩阵是一种特殊的托普利茨矩阵，其中每行都通过周期性地将上一行中的各元向右移一位来获得。如果 v 是标量，则 A = TestArrays.circul(1:v)。 属性： A 的特征系统是显式可知的。如果 t 是 n 次单位根，则 v 和 w = [1,t,t^2,...,t^(n – 1)] 的内积是 A 的特征值，w[n:-1:1] 是特征向量。 另请参阅：toeplitz
clement	描述：具有零值对角线元的 Clement 三对角矩阵 语法： A = TestArrays.clement(n,k) 返回一个主对角线元为 0 的 n×n 三对角矩阵。对于 k = false（默认值），A 是非对称的。对于 k = true，A 是对称的。 属性： A = TestArrays.clement(n,true) 在对角线方面类似于 B = TestArrays.clement(n,0)，其中存在一个具有相同大小的对角线矩阵 D 满足 B = inv(D)*A*D。 如果 n 是奇数，则 A 是奇异矩阵。 A 的特征值是显式可知的，其中包括正负数字 n-1、n-3、n-5、...、1 或 0。 前两个属性也适用于 TestArrays.tridiag(x,y,z)，其中 y = zeros(n)。如果输入向量 y 的长度或 n 是奇数，则 TestArrays.tridiag(x,y,z) 是奇异的。特征值仍然以正负对组形式出现，但它们并不是显式可知的。 对于奇数 n = 2*m+1 和 k = 0，TestArrays.clement(n,false) 有 m+1 个奇异值等于 @. sqrt((2*m+1)^2 - (2*t+1)^2)（对于 t = 0:m）。

<table>
<tr><td>compar</td><td>描述：比较矩阵
语法：
A = TestArrays.compar(B,k) 返回 B 的比较矩阵。 对于 k = false（默认值），如果 i == j，则 A[i,j] = abs(B[i,j])，否则 A[i,j] = -abs(B[i,j])。
对于 k = true，A = TestArrays.compar(B,true) 将 B 的每个对角线元素替换为其绝对值，并将每个非对角线元素替换为同一行中非对角线元素的最大绝对值的负值。
属性：
如果 B 是三角矩阵，则 A = TestArrays.compar(B,true) 也是三角矩阵。</td></tr>
<tr><td>condex</td><td>描述：矩阵条件数估计量的反例
语法：
A = TestArrays.condex(n,k,alpha) 返回条件数估计量的反例矩阵。它采用标量参数 alpha（默认值为 100），并返回大小为 n×n 的矩阵。
该矩阵、其实际大小及其适用的估计量是由 k 指定的：<table><tr><td>k = 1</td><td>4×4</td><td>LINPACK 的 RCOND</td></tr><tr><td>k = 2</td><td>3×3</td><td>LINPACK 的 RCOND</td></tr><tr><td>k = 3</td><td>任意</td><td>LINPACK 的 RCOND</td></tr><tr><td>k = 4（默认值）</td><td>n >== 4</td><td>LAPACK 的 RCOND 的逆矩阵）</td></tr></table>如果 n 不等于该矩阵的实际大小，则为该矩阵补上单位矩阵以构成 n 阶。</td></tr>
<tr><td>cycol</td><td>描述：其列周期性重复的矩阵
语法：
A = TestArrays.cycol(n,k) 返回具有周期性重复列的 n×n 矩阵，其中一个周期由 randn(n,k) 定义的列构成。因此，矩阵 A 的秩不能超过 k，k 必须是标量。参数 k 是标量，默认值为 round(n/4)，它不需要整除 n。
A = TestArrays.cycol((m,n),k) 返回具有周期性重复列的 m×n 矩阵，其中一个周期由 randn(m,k) 定义的列构成。</td></tr>
<tr><td>dorr</td><td>描述：对角占优、病态、三对角矩阵（稀疏矩阵）
语法：
A,v1,v2,v3 = TestArrays.dorr(n,theta) 返回 Dorr 矩阵，即 n×n 的行对角</td></tr>
</table>

<table>
<tr><td></td><td>占优的三对角矩阵 A 以及定义 Dorr 矩阵的向量 v1, v2, v3。如果 theta 为较小的非负值，则该矩阵是病态的。theta 的默认值是 0.01。</td></tr>
<tr><td>dramadah</td><td>描述：由 0 和 1 组成的矩阵
语法：
A = TestArrays.dramadah(n,k) 返回由 0 和 1 组成的 n×n 矩阵。n 和 k 必须均为标量。对于 k = 0 和 1，mu(A) = norm(inv(A),"fro") 相对较大（尽管不一定最大）[2]。参数 k 确定输出矩阵的字符，如下所示。
<table>
<tr><td>k = 1
（默认值）</td><td>A 是满足 abs(det(A)) = 1 和 mu(A) >= c*(1.75)^2n 的托普利茨和幺模矩阵，其中 c 是常量。A 的逆矩阵具有整数项。</td></tr>
<tr><td>k = 2</td><td>A 是上三角托普利茨矩阵。A 的逆矩阵具有整数项。</td></tr>
<tr><td>k = 3</td><td>A 是托普利茨矩阵，它在下 Hessenberg 二值矩阵（所有元素均为 0 或 1）中具有最大行列式。det(A) 等于第 n 个斐波那契数。特征值在复平面中的分布很有意义。</td></tr>
</table></td></tr>
<tr><td>fiedler</td><td>描述：Fiedler 对称矩阵
语法：
A = TestArrays.fiedler(x)，其中 x 是长度为 n 的向量，返回含有元素 A[i,j] = abs(x[i]-x[j]) 的 n×n 对称矩阵。对于标量 x，A = TestArrays.fiedler(1:x)。
属性：
矩阵 A 具有占优正特征值，所有其他特征值是负数。
inv(A) 和 det(A) 的显式公式由 Fiedler 在其著作 [3] 中给出。这些公式说明，除了非零 (1,n) 和 (n,1) 元素外，inv(A) 是三对角矩阵。</td></tr>
<tr><td>forsythe</td><td>描述：Forsythe 矩阵或扰动 Jordan 块
语法：
A = TestArrays.forsythe(n,alpha,lambda) 返回具有特征值 lambda 的等于 Jordan 块的 n×n 矩阵，但 A[n,1] = alpha 除外。标量 alpha 和 lambda 的默认值分别为 sqrt(eps()) 和 0。
属性：
A 的特征多项式由 det(A-t*I) = (lambda-t)^n - alpha*(-1)^n 给出。</td></tr>
<tr><td>frank</td><td>描述：具有病态特征值的 Frank 矩阵
语法：
对于默认值 k = 0，A = TestArrays.frank(n,k) 返回大小为 n×n 的 Frank</td></tr>
</table>

	矩阵。Frank 矩阵是具有行列式 1 的上 Hessenberg 矩阵。如果 k = 1，则围绕反对角线 (1,n)、(2,n-1)、…、(n,1) 反射元素。 属性： 可以按 Hermite 多项式的零值形式获取 A 的特征值。它们是正值并以可逆对组形式出现。 如果 n 是奇数，则 1 是特征值。 最小的一半特征值（即最小的 floor(n/2) 个特征值）是病态的。
galleryint	描述：特殊的测试矩阵 语法： A = TestArrays.galleryint(n) 返回 n×n 矩阵。 对于 n = 3，TestArrays.galleryint(n) 返回一个病态的 3x3 矩阵。 对于 n = 5，TestArrays.galleryint(n) 返回一个具有全零作为特征值的矩阵。
gcdmat	描述：最大公约数矩阵 语法： A = TestArrays.gcdmat(n) 返回 n×n 矩阵，其 A[i,j] 等于 gcd(i,j)。 属性： 对于所有非负 r，矩阵 A 是对称正定矩阵，A.^r 是对称半正定矩阵。
gearmat	描述：Gear 矩阵 语法： A = TestArrays.gearmat(n,i,j) 返回 n×n 矩阵，其中下对角线和上对角线上为 1，(1,abs(i)) 位置为 sign(i)，(n,n+1-abs(j)) 位置为 sign(j)，其余所有位置为 0。参数 i 和 j 的默认值分别为 n 和 -n。它们必须为 -n 到 n 范围内的整数。 属性： 矩阵 A 是奇异矩阵，可具有双重和三重特征值，并且可以是亏损矩阵。 所有特征值的形式均为 2*cos(a)，特征向量的形式为 [sin(w+a), sin(w+2a), …, sin(w+na)]，其中 a 和 w 在 [4] 中给出。
grcar	描述：具有敏感特征值的托普利茨矩阵 语法： A = TestArrays.grcar(n,k) 返回 n×n 托普利茨矩阵，其中下对角线上的元素为 -1，主对角线上的元素为 1，主对角线上方的 k 个对角线上的元素为 1。k 必须为整数，默认值为 k = 3。A 具有对扰动敏感的特征值。
hanowa	描述：其特征值位于复平面中的垂直线上的矩阵

<table>
<tr><td></td><td>语法：
A = TestArrays.hanowa(n,alpha) 返回 2×2 分块矩阵，其中包含四个 n/2×n/2 分块，形式如下：
[alpha*eye(n/2) -diagm(1:n/2)
diagm(1:n/2) alpha*eye(n/2)]
参数 n 必须为偶数整数。alpha 的默认值为 -1。A 具有复数特征值，形式为 alpha ± k*i (1 <= k <= n/2)。</td></tr>
<tr><td>house</td><td>描述：Householder 矩阵
语法：
v,beta,s = TestArrays.house(x) 接受标量或 n 元素列向量形式的 x 并返回 beta 和 v，以使 H*x = s*e1。在此表达式中，e1 是 eye(n) 的第一列，abs(s) = norm(x)，H = eye(n) - beta*v*v 是 Householder 矩阵。Householder 矩阵 H 满足以下属性
H*x = -sign(x[1])*norm(x)*e1
其中，e1 是 eye(n) 的第一列。
请注意，如果 x 是复数，则 sign(x) = exp(i*arg(x))，它在 x 为非零值时等于 x./abs(x)。如果 x 为零，则 v 为零且 beta = 1。
k 确定 s 的符号，如下所示。<table><tr><td>k = 0（默认值）</td><td>sign(s) = -sign(x[1])</td></tr><tr><td>k = 1</td><td>sign(s) = sign(x[1])</td></tr><tr><td>k = 2</td><td>sign(s) = 1（x 必须为实数）</td></tr></table>如果 x 为零，或如果 x = alpha*e1 (alpha >== 0) 且 k = 1 或 k = 2，则 v 为零，beta = 1 且 s = x[1]。在本例中，H = eye(n) 是单位矩阵，该矩阵并不是严格意义上的 Householder 矩阵。</td></tr>
<tr><td>invhess</td><td>描述：上 Hessenberg 矩阵的逆矩阵
语法：
A = TestArrays.invhess(x,y)，其中 x 是长度为 n 的向量，y 是长度为 n-1 的向量。如果 i <= j，它返回包含元素 A[i,j] = x[j] 的矩阵，否则返回包含元素 A[i,j] = y[i] 的矩阵。A 的下三角基于向量 x，而严格上三角基于 y。参数 y 默认为 -x[1:n-1]。
对于标量 n，TestArrays.invhess(n) 与 TestArrays.invhess(1:n) 相同。
属性：
如果对于所有 i，x[1] != 0 且 x[i+1] != y[i]，则矩阵 A 是非奇异矩阵。</td></tr>
</table>

	A 的逆矩阵是上 Hessenberg 矩阵。
invol	描述：对合矩阵（矩阵与其逆矩阵相同） 语法： A = TestArrays.invol(n) 返回 n×n 对合矩阵，其中 A*A = eye(n) 和 A 为病态。它是 hilb(n) 的沿对角线缩放的版本。 属性： 矩阵 B = (eye(n)-A)/2 和 B = (eye(n)+A)/2 是幂等的，即 B*B = B。 另请参阅：hilb
ipjfact	描述：含有阶乘元素的 Hankel 矩阵 语法： A,beta = TestArrays.ipjfact(n,k) 返回 n×n Hankel 矩阵 A 及其行列式 beta，该行列式是显式可知的。A 的逆矩阵也是显式可知的。 如果 k = false（默认值），则 A 的元素是 A[i,j] = factorial(i+j)。如果 k = true，则 A 的元素为 A[i,j] = 1/factorial(i+j)。 另请参阅：hankel | toeplitz
jordbloc	描述：Jordan 分块矩阵 语法： A = TestArrays.jordbloc(n,lambda) 返回特征值为 lambda 的 n×n Jordan 块。lambda 的默认值是 1。
kahan	描述：上梯形 Kahan 矩阵 语法： A = TestArrays.kahan(n,theta,pert) 返回一个上梯形矩阵，该矩阵具有有关条件和秩的估计值的有意义属性。例如，Kahan 矩阵说明使用列置换的 QR 分解无法给出矩阵的良好秩估计。 如果 n 是一个二元素元组，则 A 为 n[1]×n[2]；否则，A 为 n×n。theta 的有用范围是 0 < theta < pi，默认值为 1.2。 为确保使用列主元消去法的 QR 分解不会在存在舍入误差时交换列，将通过 pert*eps()*diagm(n:-1:1) 扰动主对角线。pert 的默认值为 1e3，此值可确保在 IEEE® 算术运算中，TestArrays.kahan(n) 至少在 n = 100 之前不会发生列互换（使用默认值 theta = 1.2 时）。
kms	描述：Kac-Murdock-Szegö 托普利茨矩阵 语法： A = TestArrays.kms(n,rho) 返回 n×n Kac-Murdock-Szegö 托普利茨矩阵以

	使 A[i,j] = rho^(abs(i-j))（对于实数 rho）。对于复数 rho，同一公式也适用，只不过对角线下的元素是共轭的。rho 的默认值为 0.5。 属性： 存在 A 的 LDL 分解，其中 L = inv(TestArrays.triw(n,-rho,1))' 和 D = (1-abs(rho)^2)*eye(n)，但 D 的第一个元素 D[1,1] = 1 除外。 当且仅当 0 < abs(rho) < 1 时 A 为正定矩阵。 inv(A) 的逆矩阵是三对角矩阵。
krylov	描述：Krylov 矩阵 语法： A = TestArrays.krylov(B,x,k) 返回 Krylov 矩阵，其中 B 是 n×n 矩阵，x 是长度为 n 的向量，k 必须为整数。Krylov 矩阵等于 [x B*x B^2*x ... B^(k-1)*x] （对于列向量 x。）如果您没有指定参数 x 和 k，它们将采用默认值 x = ones(n,1) 和 k = n。 对于标量 n，A = TestArrays.krylov(n) 与 B = randn(n) 的 TestArrays.krylov(B) 相同。
lauchli	描述：Lauchli 矩形矩阵 语法： A = TestArrays.lauchli(n,mu) 返回 [ones(1,n); mu*eye(n)] 形式的 (n+1)×n 矩阵。参数 mu 默认为 sqrt(eps)。 属性： Lauchli 矩阵是最小二乘及其他问题的一个众所周知的示例，它说明了求解方程 A'Ax=A'b 时构建 A'*A 的危险。矩阵 A'*A 通常比初始矩阵 A 对舍入误差更敏感。
lehmer	描述：Lehmer 对称正定矩阵 语法： A = TestArrays.lehmer(n) 返回对称正定 n×n 矩阵以使 A[i,j] = i/j（对于 j >== i），否则 A[i,j] = j/i。 属性： A 是完全非负矩阵。 inv(A) 的逆矩阵是三对角和显式已知的矩阵。 阶数 n 满足 n <= cond(A) <= 4*n*n。
leslie	描述：由 Leslie 人口模型的出生率和成活率组成的矩阵

	语法： A = TestArrays.leslie(x,y) 是从 Leslie 人口模型（平均出生率为 x[1:n]，成活率为 y[1:n-1]）得出的 n×n 矩阵。除了包含 x[i] 的第一行和包含 y[i] 的第一个下对角线之外，矩阵元素大多为零。对于有效模型，x 的元素是非负值，y 的元素是正值并且以 1 为界，其中 0 < y[i] <= 1。 A = TestArrays.leslie(n) 生成 Leslie 矩阵并且 x = ones(n) 和 y = ones(n-1)。
lesp	描述：含有实数敏感特征值的三对角矩阵 语法： A = TestArrays.lesp(n) 返回一个 n×n 矩阵，其特征值是实数且均匀分布在区间 [-2*n-3.5,-4.5] 中。 属性： 随着特征值变为越来越小的负值，特征值的敏感度呈指数增加。 该矩阵使用 A = inv(D)*B*D（其中 D = diagm(factorial.(1:n))）进行相似变换，这类似于通过对称三对角矩阵 B，后者具有相同的对角线元和非对角线元 1。
lotkin	描述：Lotkin 矩阵 语法： A = TestArrays.lotkin(n) 返回 Hilbert 矩阵，它的第一行全部更改为 1。 属性： Lotkin 矩阵 A 是非对称且病态的矩阵，具有许多模较小的负特征值。 其逆矩阵具有整数项且是显式已知的 [5]。
minij	描述：对称正定矩阵 语法： A = TestArrays.minij(n) 返回包含项 A[i,j] = min(i,j) 的 n×n 对称正定矩阵。 属性： A 具有特征值 0.25*sec(r*pi/(2*n+1)).^2，其中 r = 1:n。 inv(A) 的逆矩阵是三对角矩阵且等于 -1 乘以第二个差分矩阵，但其 (n,n) 元素为 1。 矩阵 2*A-ones(size(A)) 具有三对角逆矩阵和特征值 @. 0.5*sec((2*r-1)*pi/(4*n))^2，其中 r = 1:n。
moler	描述：Moler 对称正定矩阵

<table>
<tr><td></td><td>语法：
A = TestArrays.moler(n,alpha) 返回对称正定 n×n 矩阵 U'*U（其中 U = TestArrays.triw(n,alpha)）。默认值为 alpha = -1、A[i,j] = min(i,j)-2 和 A[i,i] = i。
属性：
对于 alpha = -1，A 的特征值之一很小，与其余特征值相比有许多数量级的差异。</td></tr>
<tr><td>neumann</td><td>描述：源自离散 Neumann 问题的奇异矩阵（稀疏矩阵）
语法：
A = TestArrays.neumann(n) 返回行对角占优的稀疏 n×n 奇异矩阵，该矩阵由使用规则网格上的普通五点算子离散 Neumann 问题所生成。参数 n 必须为完全平方整数。A 是稀疏矩阵且具有包含零向量 ones(n) 的一维零空间。
A = TestArrays.neumann((m,n)) 返回相同的矩阵，但大小为 m*n×m*n。参数 m 和 n 必须为正整数，其中 m 必须大于 1。</td></tr>
<tr><td>orthog</td><td>描述：正交矩阵和准正交矩阵
语法：
A = TestArrays.orthog(n,k) 返回第 k 个类型的 n 阶矩阵，其中 k >= 0 返回严格正交矩阵，k < 0 返回正交矩阵的不同对角缩放矩阵。
<table>
<tr><td>k = 1
（默认值）</td><td>A[i,j] = sqrt(2/(n+1)) * sin(i*j*pi/(n+1)) 第二个差分矩阵 TestArrays.tridiag(n) 的特征向量矩阵。A 是对称正交矩阵。</td></tr>
<tr><td>k = 2</td><td>A[i,j] = 2/(sqrt(2*n+1)) * sin(2*i*j*pi/(2*n+1))
对称正交矩阵。</td></tr>
<tr><td>k = 3</td><td>A[r,s] = exp(2*pi*i*(r-1)*(s-1)/n) / sqrt(n) 酉复矩阵。A^4 是单位矩阵。这是与 fft(eye(n))/sqrt(n) 完全相同的矩阵。</td></tr>
<tr><td>k = 4</td><td>标准 Helmert 矩阵：下 Hessenberg 矩阵的置换，该矩阵的第一行是 ones(n)/sqrt(n)。</td></tr>
<tr><td>k = 5</td><td>A[i,j] = sin(2*pi*(i-1)*(j-1)/n)/sqrt(n) + cos(2*pi*(i-1)*(j-1)/n)/sqrt(n)
Hartley 变换产生的对称矩阵。</td></tr>
<tr><td>k = 6</td><td>A[i,j] = sqrt(2/n) * cos((i-1/2)*(j-1/2)*pi/n) 以离散余弦变换形式产生的对称矩阵。</td></tr>
<tr><td>k = -1</td><td>A[i,j] = cos((i-1)*(j-1)*pi/(n-1))</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td><table>
<tr><td></td><td>基于第一类 T(n-1) 的 Chebychev 多项式的极值的 Chebyshev Vandermonde 类矩阵。此矩阵是对称矩阵。该矩阵的每个偶数列（行）向量与每个奇数列（行）向量正交，即对于任何偶数 i 和奇数 j，dot(A[i,:],A[j,:]) = 0。</td></tr>
<tr><td>k = -2</td><td>A[i,j] = cos((i-1)*(j-1/2)*pi/n)
基于 T(n) 的零值的 Chebyshev Vandermonde 类矩阵。该矩阵的行向量相互正交，即对于 i 不等于 j 的情况，dot(A[i,:],A[j,:]) = 0。矩阵 A*A' 不是单位矩阵，而是对角矩阵。</td></tr>
</table></td></tr>
<tr><td>parter</td><td>描述：Parter 矩阵
语法：
A = TestArrays.parter(n) 返回矩阵 A 以使 A[i,j] = 1/(i-j+0.5)。
属性：
A 是柯西和托普利茨矩阵。
A 的大多数奇异值非常接近于 pi。</td></tr>
<tr><td>pei</td><td>描述：Pei 矩阵
语法：
A = TestArrays.pei(n,alpha)（其中 alpha 是标量）返回对称矩阵 alpha*eye(n) + ones(n)。alpha 的默认值是 1。对于等于 0 或 -n 的 alpha，该矩阵是奇异矩阵。</td></tr>
<tr><td>poisson</td><td>描述：来自 Poisson 方程的块三对角矩阵（稀疏矩阵）
语法：
A = TestArrays.poisson(n) 返回 n^2 阶稀疏块三对角矩阵，该矩阵使用 n×n 网格上的 5 点算子离散 Poisson 方程所产生。</td></tr>
<tr><td>prolate</td><td>描述：Prolate 矩阵
语法：
A = TestArrays.prolate(n,alpha) 返回具有参数 alpha 的 n×n prolate 矩阵。它是对称病态托普利茨矩阵。如果 0 < alpha < 0.5，则 A 是正定矩阵。
属性：
A 的特征值不同，它们在 (0,1) 区间内且往往聚集在 0 和 1 周围。
w 的默认值为 0.25。</td></tr>
<tr><td>qmult</td><td>描述：随机正交矩阵的左乘矩阵
语法：</td></tr>
</table>

<table>
<tr><td></td><td>A = TestArrays.qmult(S) 返回 Q*S，其中 Q 为来自 S 的行的数量的维度的 Haar 分布的随机实正交矩阵。当 S 为标量，则 qmult(S) 与 qmult(eye(S)) 等价。
A = qmult(S, method) 指定计算方法。method = false 为默认值，采取默认方法，而 method = true 则会使用 qr 分解。</td></tr>
<tr><td>randcolu</td><td>描述：具有归一化列和指定的奇异值的随机矩阵
语法：
A = TestArrays.randcolu(n) 生成一个随机 n×n 矩阵，该矩阵具有单位 2-范数的归一化列和来自均匀分布的随机奇异值。
A'*A 是与由 TestArrays.randcorr(n) 生成的矩阵具有相似形式的相关矩阵。后者的特征值是均匀分布的，而前者的特征值的平方根是均匀分布的。
A = TestArrays.randcolu(x)，其中 x 是长度为 n (n >= 1) 的向量，它生成一个随机 n×n 矩阵，其奇异值由向量 x 给出。向量 x 必须具有其平方和为 n 的非负元素。此矩阵还具有单元 2-范数的归一化列。
A = TestArrays.randcolu(__,m)（其中 m >== n）生成 m×n 矩阵。
A = TestArrays.randcolu(__,m,k) 基于 k 提供更多选项。
<table>
<tr><td>k = false（默认值）</td><td>diag(x) 最初取决于随机的双向正交变换，随后应用 Givens 旋转序列。</td></tr>
<tr><td>k = true</td><td>省略了初始变换。这在生成测试矩阵时更快，但生成的矩阵通常有许多零值元。</td></tr>
</table>
</td></tr>
<tr><td>randcorr</td><td>描述：具有指定特征值的随机相关矩阵
语法：
A = TestArrays.randcorr(n) 是一个随机 n×n 相关矩阵，具有均匀分布的随机特征值。相关矩阵是对角线上为 1 的对称半正定矩阵。
A = TestArrays.randcorr(x) 生成一个随机相关矩阵，其特征值由向量 x 给定，其中 length(x) >= 1。向量 x 必须有其和为 length(x) 的非负元素。
A = TestArrays.randcorr(x,k) 基于 k 提供更多选项。
<table>
<tr><td>k = false（默认值）</td><td>特征值的对角矩阵最初取决于随机正交相似变换，随后应用 Givens 旋转序列 [6]。</td></tr>
<tr><td>k = true</td><td>省略了初始变换。这在生成测试矩阵时更快，但生成的矩阵通常有许多零值元。</td></tr>
</table>
另请参阅：corrcoef</td></tr>
<tr><td>randhess</td><td>描述：随机正交上 Hessenberg 矩阵</td></tr>
</table>

<table>
<tr><td></td><td>语法：
A = TestArrays.randhess(n) 返回 n×n 实数随机正交上 Hessenberg 矩阵。
A = TestArrays.randhess(x) 使用 x 的元素作为参数以非随机方式构造 A。x 必须是长度为 n 的实数向量，其中 n >= 1。 在这两种情况下，矩阵 A 是基于 n-1 Givens 旋转的乘积构造的。</td></tr>
<tr><td>randjorth</td><td>描述：随机 J 正交矩阵
语法：
A = TestArrays.randjorth(n) 生成一个满足关系 A'*J*A = J 的随机 n×n 的 J 正交矩阵 A（此类矩阵也称为双曲矩阵）。此处，J = blkdiag(eye(ceil(n/2))-eye(floor(n/2))) 且 cond(A) = sqrt(1/eps())。
A = TestArrays.randjorth(n,m)（对于正整数 n 和 m）生成随机 (n+m)×(n+m) 正交矩阵 A。此处，J = blkdiag(eye(n),-eye(m)) 且 cond(A) = sqrt(1/eps())。
A = TestArrays.randjorth(n,m,alpha,symm,method) 使用以下可选输入参数
alpha - 指定 cond(A) = alpha，其中 alpha 必须等于或大于 1。
symm - 选择是否强制对称。如果逻辑值 symm 为 false，则 A 是非对称矩阵。如果逻辑值 symm 为 true，则 A 是对称矩阵。
method - 是否选择调用 qr 来执行基础正交变换。如果逻辑值 method 为 false，则不调用 qr。如果逻辑值 method 为 true，则调用 qr。在针对大维度创建 J 正交矩阵时，调用 qr 比默认方法快得多。</td></tr>
<tr><td>rando</td><td>描述：由元素 –1、0 或 1 组成的随机矩阵
语法：
A = TestArrays.rando(n,k) 返回一个随机 n×n 矩阵。输入参数 k 根据以下离散分布之一确定矩阵元素。
<table>
<tr><td>k = 1（默认值）</td><td>A[i,j] = 0 或 1，具有相等概率。</td></tr>
<tr><td>k = 2</td><td>A[i,j] = -1 或 1，具有相等概率。</td></tr>
<tr><td>k = 3</td><td>A[i,j] = -1、0 或 1，具有相等概率。</td></tr>
</table>
A = TestArrays.rando((n,m),k) 返回一个随机 n×m 矩阵。</td></tr>
<tr><td>randsvd</td><td>描述：具有预分配奇异值的随机矩阵
语法：
A = TestArrays.randsvd(n,kappa,mode,kl,ku) 返回 n 阶（多对角）条带随机矩阵，条件数为 cond(A) = abs(kappa) 且必须大于或等于 1。如果 n 是二元素向量，A 的大小为 n(1)×n(2)。</td></tr>
</table>

<table>
<tr><td></td><td>kappa 的默认值是 sqrt(1/eps())。分布众数 mode 决定矩阵的奇异值。
参数 kl 和 ku 分别指定 A 中下非对角线和上非对角线的数目。如果省略它们，将使用 kl = n-1 和 ku = kl 生成一个满矩阵。如果只存在 kl，则 ku 默认为 kl。
mode 的可用值如下所示。
<table>
<tr><td>mode = 1</td><td>一个等于 1 的大奇异值（其余奇异值等于 1/abs(kappa)）。</td></tr>
<tr><td>mode = 2</td><td>一个等于 1/abs(kappa) 的小奇异值（其余奇异值等于 1）。</td></tr>
<tr><td>mode = 3（默认值）</td><td>几何分布的奇异值。</td></tr>
<tr><td>mode = 4</td><td>算术分布的奇异值。</td></tr>
<tr><td>mode = 5</td><td>具有均匀分布对数的随机奇异值。</td></tr>
<tr><td>mode < 0</td><td>如果 mode 是 -1、-2、-3、-4 或 -5，则 randsvd 将 mode 视为 abs(mode)。然而，在原始奇异值矩阵中，对角线元的顺序相反：从小到大，而不是从大到小。</td></tr>
</table>
在 kappa <= 1 的特例中，A 是一个对称正定矩阵，cond(A) = -kappa 且特征值根据 mode 分布。在这种情况下，参数 kl 和 ku 被忽略。
A = TestArrays.randsvd(n,kappa,mode,kl,ku,method) 指定如何执行生成测试矩阵的计算。method = false 是默认值，而 method = true 使用一种替代方法来调用 qr，对于大的维度，该方法快得多，尽管它每秒使用更多浮点运算。</td></tr>
<tr><td>redheff</td><td>描述：由 1 和 0 组成的 Redheffer 矩阵
语法：
A = TestArrays.redheff(n) 返回 A[i,j] = 1 定义的由 0 和 1 组成的 n×n 矩阵（如果 j = 1 或 i 等分 j，否则 A(i,j) = 0）。
属性：
A 具有等于 1 的 n-floor(log2(n))-1 特征值。
A 的一个实数特征值（及其谱半径）近似于 sqrt(n)。
A 的一个负特征值近似于 -sqrt(n)。
其余 floor(log2(n))-1 个特征值具有相对较小的模数，位于圆 log2−εn 内（对于 ε >= 0 和足够大的 n）。
对于每个 ε >= 0 的情况，当且仅当 ｜det(A)｜=O(n1/2+\epsilon) 时，黎曼假设才成立。</td></tr>
</table>

	Barrett 和 Jarvis 推断小特征值的 floor(log2(n)) 均位于单位圆 abs(z) = 1 内。此推断的证明，加上随着 n 趋于无穷某些特征值趋于零这一证明，可得到一个新的质数定理证明 [7]。
riemann	描述：与黎曼假设关联的矩阵 语法： A = TestArrays.riemann(n) 返回一个当且仅当以下条件成立时黎曼假设才为真的 n×n 矩阵： det(A)=O(n!n-1/2+ε) （对于每个 ε >= 0）[8]。 黎曼矩阵由 A = B[2:n+1,2:n+1] 定义，其中，如果 i 等分 j，则 B[i,j] = i-1，否则 B[i,j] = -1。 属性 每个特征值 e[i] 都满足 abs(e[i]) <= m-1/m，其中 m = n+1。 特征值也满足 i <= e[i] < i+1，最多 m-sqrt(m) 个例外。 区间 (m/3,m/2] 中的所有整数都是特征值。
ris	描述：Ris 矩阵 语法： A = TestArrays.ris(n) 返回一个含有元素 A[i,j] = 0.5/(n-i-j+1.5) 的对称 n×n Hankel 矩阵。A 的特征值聚集在 π/2 和 –π/2 周围。
sampling	描述：具有病态整数特征值的非对称矩阵 语法： A = TestArrays.sampling(x)（其中 x 是长度为 n 的向量）返回 n×n 非对称矩阵。矩阵的元素是 A[i,j] = x[i]/(x[i -x[j])（对于 i != j）且 A[j,j] 等于 j 列中的非对角线元素之和。如果 x 是标量，则 A = TestArrays.sampling(1:x)。 属性： A 具有病态整数特征值 0:n-1。 对于特征值 0 和 n-1，对应的特征向量分别是 x 和 ones(n)。 对于 i != j，A 满足 A[i,j] + A[j,i] = 1。 显式公式可用于 A 的左特征向量。 此矩阵的特例出现在抽样理论中，在该理论中，如果其右特征向量经过适当归一化，可给出条件泊松采样设计的包含概率 [9]。
smoke	描述：具有“烟环”伪谱的复矩阵

	语法： A = TestArrays.smoke(n) 返回一个 n×n 矩阵，其上对角线上的元素为 1，(n,1) 位置上的元素为 1。对角线由包含所有 n 次单位根的集合组成。 A = TestArrays.smoke(n,true) 返回与上面相同的矩阵，只不过元素 A[n,1] 为零。 属性： TestArrays.smoke(n,true) 的特征值是 n 次单位根。 TestArrays.smoke(n) 的特征值是 n 次单位根乘以 2^(1/n)。 矩阵 A 的伪谱可以通过在复平面中求矩阵 zI-A 的最小奇异值来计算。此处，z 表示复平面上的点，I 是单位矩阵。TestArrays.smoke(n) 和 TestArrays.smoke(n,true) 的伪谱在其特征值附近具有“烟环”模式。
toeppd	描述：对称正定托普利茨矩阵 语法： A = TestArrays.toeppd(n,m,x,theta) 返回一个 n×n 对称托普利茨矩阵，该矩阵由 m 个托普利茨矩阵（秩为 1 或 2）之和组成。x 和 theta 是长度为 m 的向量。如果 x 的所有元素均为正值，则矩阵 A 是正定矩阵。具体而言，A 通过以下方式生成 A = x[1]*T1 + ... + x[k]*Tk + ... + x[m]*Tm 其中 Tk 是依赖于 theta[k] 的 n×n 矩阵。Tk 的元素是 Tk[i,j] = cos(2*pi*theta[k]*(i-j))。 默认情况下，m = n、x = rand(m) 且 theta = rand(m)。
toeppen	描述：五对角托普利茨矩阵（稀疏矩阵） 语法： A = TestArrays.toeppen(n,a,b,c,d,e) 返回具有以下元素的 n×n 稀疏五对角托普利茨矩阵：主对角线下方的第二个对角线上的元素为 a，下对角线上的元素为 b，主对角线上的元素为 c，上对角线上的元素为 d，主对角线上方的第二个对角线上的元素为 e，其中 a、b、c、d 和 e 为标量。 默认情况下，a,b,c,d,e = 1,-10,0,10,1，从而生成最初由 Rutishauser 提出的矩阵 [10]。此矩阵的特征值大致位于复平面中的曲线 2*cos(2*t) + 20*i*sin(t) 上。
tridiag	描述：三对角矩阵（稀疏矩阵） 语法： A = TestArrays.tridiag(n) 返回大小为 n×n 的稀疏三对角矩阵，其中下对

	角线元素为 -1，对角线元素为 2，上对角线元素为 -1。此矩阵具有特征值 2 + 2*cos(k*pi/(n+1))，其中 k = 1:n。 生成的矩阵是具有实非负特征值的对称正定 M 矩阵。此矩阵也是第二个差分矩阵的负矩阵。 A = TestArrays.tridiag(c,d,e) 返回由向量 c、d 和 e 定义的下对角线为 c、对角线为 d 和上对角线为 e 的三对角矩阵。向量 c 和 e 的长度必须为 length(d)-1。 A = TestArrays.tridiag(n,c,d,e)（其中 c、d 和 e 都是标量）生成大小为 n×n 的托普利茨三对角矩阵，其中下对角线元素为 c，对角线元素为 d，上对角线元素为 e。此矩阵具有特征值 d + 2*sqrt(c*e)*cos(k*pi/(n+1))，其中 k = 1:n。
triw	描述：Wilkinson 和其他人讨论的上三角矩阵 语法： A = TestArrays.triw(n,alpha,k) 返回对角线上为 1，第一个 k >== 0 上对角线上为 alpha 的上三角矩阵。默认情况下，alpha = -1 且 k = n-1。 阶数 n 可以是二元素向量，在这种情况下矩阵为 n[1]×n[2] 上梯形矩阵。 属性： 对于 alpha = 2，矩阵的条件数满足：cond(TestArrays.triw(n,2)) = cot(pi/(4*n))^2。 对于大的 abs(alpha)，cond(TestArrays.triw(n,alpha)) 近似于 abs(alpha)^n*sin(pi/(4*n-2))。 当您将 -2^(2-n) 与其 (n,1) 元素相加时，矩阵 A = TestArrays.triw(n) 会变为奇异矩阵。当您将 -2^(1-n) 与第一列中的元素相加时，该矩阵也会变为奇异矩阵。
wathen	描述：Wathen 矩阵（稀疏矩阵） 语法： A = TestArrays.wathen(nx,ny) 返回稀疏随机 n×n 有限元矩阵，其中 n = 3*nx*ny + 2*nx + 2*ny + 1。 矩阵 A 准确地说是二维 8 节点（巧合）元素的常规 nx×ny 网格的“相容质量矩阵”。对于随机选择的“密度”rho(nx,ny) 的任何（正）值，A 是对称正定矩阵。 B = TestArrays.wathen(nx,ny,true) 基于上述语法返回一个沿对角线缩放的矩阵，其中 B = diagm(diag(A))\A。此矩阵的特征值满足 0.25 <= eigvals(B) <= 4.5（对于任何正整数 nx 和 ny 以及任何密度 rho(nx,ny)。）

wilk	描述：Wilkinson 设计或讨论的各种矩阵 语法： U,b = TestArrays.wilk(3) 返回一个表明不精确解的上三角矩阵方程组 U*x = b。 L,b = TestArrays.wilk(4) 返回一个病态下三角矩阵方程组 L*x = b。 A = TestArrays.wilk(5) 返回一个对称正定矩阵 A = B[1:5,2:6]*1.8144，其中 B = hilb(6)。 A = TestArrays.wilk(21) 返回 W21+，这是一个三对角矩阵，具有几乎相等的特征值对。

3.2.1.8.10 gallery

测试矩阵

A. 语法

```
A1, A2, ..., Am = gallery(matrixname,P1,P2,...,Pn)
_ = gallery(matrixname,typename,P1,P2,...,Pn)
_ = gallery(matrixname,typename,rngstream,P1,P2,...,Pn)
_ = gallery(matrixname,rngstream,P1,P2,...,Pn)
A = gallery(3)
A = gallery(typename,3)
A = gallery(5)
A = gallery(typename,5)
```

B. 说明

A1, A2, ..., Am = gallery(matrixname,P1,P2,...,Pn)生成由 matrixname 指定的一系列测试矩阵。P1,P2,...,Pn 是单个矩阵系列要求的输入参数。调用语法中使用的输入参数 P1,P2,...,Pn 的数目因矩阵而异。matrixname 一节中说明了每个矩阵系列的确切调用语法。示例

A1,A2,..., Am = gallery(matrixname,typename, P1,P2,...,Pn) 还指定生成的测试矩阵的数据类型 typename。

A1,A2,..., Am = gallery(matrixname,typename, rngstream, P1,P2,...,Pn) 指定生成的测试矩阵的数据类型 typename 与随机数使用的随机数流 rngstream。

A1,A2,..., Am = gallery(matrixname, rngstream, P1,P2,...,Pn) 指定生成的测试矩阵中随机数使用的随机数流 rngstream。

A = gallery(3) 生成一个对扰动敏感的病态 3×3 矩阵。示例

A = gallery(typename, 3) 还指定生成的对扰动敏感的病态 3×3 矩阵的数据类型 typename。

A = gallery(5) 生成一个 5×5 矩阵，它具有一个有趣的特征值问题，即对舍入误差很敏感。示例

A = gallery(typename, 5) 还指定 5×5 矩阵的数据类型 typename。。

C. 示例

(1). 以缩放颜色显示矩阵元素

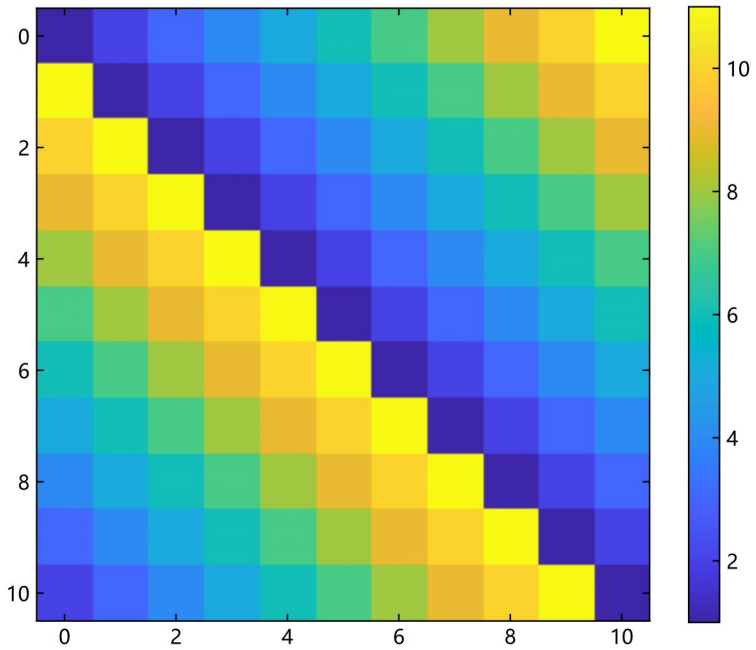
以缩放颜色显示几个测试矩阵的矩阵元素。

创建一个大小为 11×11 的循环矩阵。循环矩阵是一种特殊的托普利茨矩阵，其中每行都通过周期性地将上一行中的各元向右移一位来获得。

```
C = gallery("circul",11)
```

显示 C 中矩阵元素的图像。向图形添加颜色栏以显示当前颜色图。

```
using TyImages
s = imagesc(C)
axis("square")
colorbar(gca(),s)
```

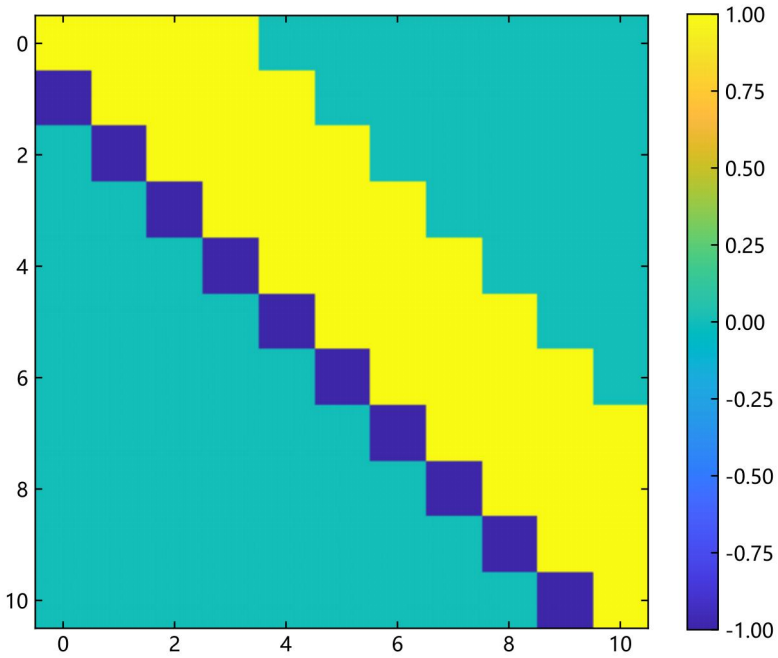


创建一个大小为 11×11 的 grcar 矩阵。grcar 矩阵是一种非对称的托普利茨矩阵，其下对角线上的元素为 -1，主对角线上的元素为 1，主对角线上方的少数几个对角线上的元素为 1。

```
G = gallery("grcar",11)
```

显示 G 中矩阵元素的图像。

```
a = imagesc(G)
axis("square")
colorbar(gca(),a)
```

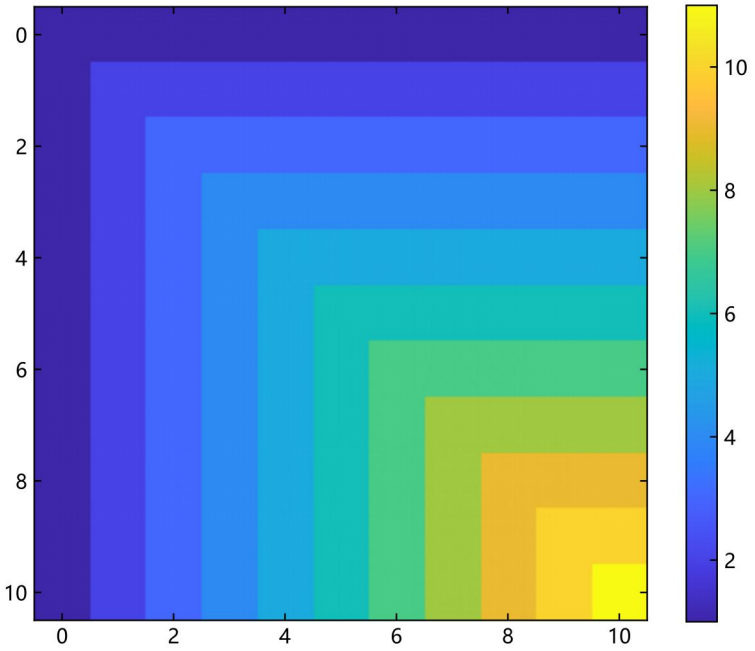


创建一个大小为 11×11 的 minij 矩阵。minij 矩阵 M 是一种其元素满足 M(i,j) = min(i,j) 的对称正定矩阵。

```
M = gallery("minij",11)
```

M = gallery("minij",11)。

```
a = imagesc(M)
axis("square")
colorbar(gca(),a)
```



(2). 具有整数逆矩阵的整数矩阵

当且仅当整数矩阵的行列式正好是 1 或 －1 时，其逆矩阵也是整数矩阵。行列式为 1 或 －1 的整数方阵也称为幺模矩阵。例如，gallery("dramadah",n) 就是这样一种矩阵，它是由 0 和 1 组成的 n×n 矩阵，行列式为 1 或 －1。

创建一个 6×6 的 dramadah 矩阵。计算其行列式和逆矩阵。

```
A = gallery("dramadah",6)
```

```
A = 6×6 Matrix{Int64}:
 1  1  0  1  0  0
 0  1  1  0  1  0
 0  0  1  1  0  1
 1  0  0  1  1  0
 1  1  0  0  1  1
 0  1  1  0  0  1
```

```
detA = det(A)
```

```
A = -1.0
```

```
invA = inv(A)
```

```
invA = 6×6 Matrix{Float64}:
 -1.0  -2.0  -3.0   4.0  -2.0   5.0
  1.0   1.0   1.0  -2.0   1.0  -2.0
 -1.0  -1.0  -2.0   3.0  -2.0   4.0
  1.0   1.0   2.0  -2.0   1.0  -3.0
```

```
 0.0   1.0   1.0  -1.0   1.0  -2.0
 0.0   0.0   1.0  -1.0   1.0  -1.0
```

该矩阵的逆矩阵只包含整数项，因为原始矩阵的行列式是 -1。

(3). 使用 Householder 变换计算 QR 分解

此示例说明如何使用 Householder 变换来计算矩阵 A = QR 的 QR 分解，其中 Q 是正交矩阵，R 是上三角矩阵。

首先，将随机数生成器设置为默认值，并创建由服从标准正态分布的随机数组成的 6×3 矩形矩阵。

```
rng = MT19937ar(5489)
A = randn(rng,6,3)
```

```
A = 6×3 Matrix{Float64}:
  0.537667  -0.433592   0.725404
  1.83389    0.342624  -0.0630549
 -2.25885    3.5784     0.714743
  0.862173   2.76944   -0.204966
  0.318765  -1.34989   -0.124144
 -1.30769    3.03492    1.4897
```

要创建 Householder 矩阵，请使用函数 v,beta = gallery("house",x)。该函数采用列向量 x，并返回 v 和β，满足 $H = I - \beta vv^T$即获得 Householder 矩阵。Householder 变换用于将除了向量 x 的第一个元素之外的所有元素归零。

计算 Householder 矩阵 并执行变换 。矩阵 在第一列的对角线下只包含零。

```
v1, beta1 = gallery("house",A[:,1])
P1 = eye(6) - beta1.*(v1*v1')
A1 = P1*A
```

```
A1 = 6×3 Matrix{Float64}:
 -3.36298      2.88409   1.04207
 -1.94289e-16  1.90243   0.0858248
  2.77556e-16  1.65714   0.531364
 -9.71445e-17  3.50276  -0.134973
 -4.16334e-17 -1.07876  -0.0982661
  2.22045e-16  1.92267   1.38354
```

接下来，计算 Householder 矩阵P_2，使得$A_2 = P_2A_1$在第一列和第二列中的对角线下只包含零。

```
v2, beta2 = gallery("house",A1[2:end,2])
v2 = vcat(0,v2)
P2 = eye(6) - beta2.*(v2*v2')
```

```
A2 = P2*A1
```

```
A2 = 6×3 Matrix{Float64}:
 -3.36298      2.88409      1.04207
 -4.5776e-17   -4.84717     -0.688471
  3.14018e-16   2.22045e-16   0.341261
 -2.00726e-17   2.77556e-16  -0.536799
 -6.53696e-17  -8.32667e-17   0.0254864
  2.6435e-16    0.0          1.16297
```

最后，计算 Householder 矩阵P_3，使得$A_3 = P_3A_2$在下对角线上的元素全部为零。

```
v3, beta3 = gallery("house",A2[3:end,3])
v3 = vcat(0,0,v3)
P3 = eye(6) - beta3.*(v3*v3')
R = P3*A2
```

```
R = 6×3 Matrix{Float64}:
 -3.36298      2.88409      1.04207
 -4.5776e-17   -4.84717     -0.688471
 -3.19581e-16   5.68246e-17  -1.32581
  1.83947e-16   3.30757e-16   5.55112e-17
 -7.50561e-17  -8.57926e-17  -1.73472e-18
 -1.77658e-16  -1.1526e-16   -1.66533e-16
```

矩阵 $\mathrm{Q} = P_3P_2P_1$是上三角矩阵。由于 Householder 矩阵是对合矩阵（矩阵与其逆矩阵相同），A 的 QR 分解成为 $\mathrm{A} = \mathrm{QR}$ 和 $\mathrm{Q} = P_1P_2P_3$ 。

```
Q = P1*P2*P3
```

```
Q = 6×6 Matrix{Float64}:
 -0.159878   -0.00567592  -0.669856    0.498317   -0.20357    -0.485735
 -0.545316   -0.395152    -0.175856   -0.643224    0.134199   -0.289506
  0.671681   -0.338591     0.164658   -0.0991317   0.15511    -0.610903
 -0.256372   -0.723894     0.328998    0.524382    0.0805065   0.143362
 -0.0947866   0.222091    -0.0961927   0.187161    0.946325   -0.0433338
  0.388848   -0.394756    -0.612994   -0.134571    0.120327    0.533493
```

将此结果与使用 qr 函数的计算结果进行比较。

```
Qa, Ra = qr(A)
```

```
Qa = 6×6 LinearAlgebra.QRCompactWYQ{Float64, Matrix{Float64}}:
 -0.159878   -0.00567592  -0.669856    0.498317   -0.20357    -0.485735
 -0.545316   -0.395152    -0.175856   -0.643224    0.134199   -0.289506
  0.671681   -0.338591     0.164658   -0.0991317   0.15511    -0.610903
 -0.256372   -0.723894     0.328998    0.524382    0.0805065   0.143362
 -0.0947866   0.222091    -0.0961927   0.187161    0.946325   -0.0433338
  0.388848   -0.394756    -0.612994   -0.134571    0.120327    0.533493
Ra = 3×3 Matrix{Float64}:
```

```
 -3.36298   2.88409   1.04207
  0.0      -4.84717  -0.688471
  0.0       0.0      -1.32581
```

在计算机精度范围内验证 A = QR，注意虽然这里的 Ra 为 3x3 矩阵，由于我们对 Qa 的特殊类型与矩阵的乘法做了特殊处理，因此这里是可以直接将 Qa 与 Ra 相乘的。

```
norm(A-Q*R)
```

```
ans = 1.472157928491015e-15
```

(4). 复平面上特征值的分布

此示例绘制复平面上 20000 个大小为 18×18 的随机循环矩阵采样的特征值分布。矩阵元素从集合 {﹣0.4,0.4} 中均匀采样。

创建一个大小为 18×20000 的数组 E 来存储特征值。

```
E = Matrix{ComplexF64}(undef, 18,20000)
```

将随机数生成器设置为默认值。在一个 for 循环语句中对以下操作迭代执行 20000 次：

创建一个长度为 18 的向量 x，其中包含的随机元素为 –0.4 或 0.4。

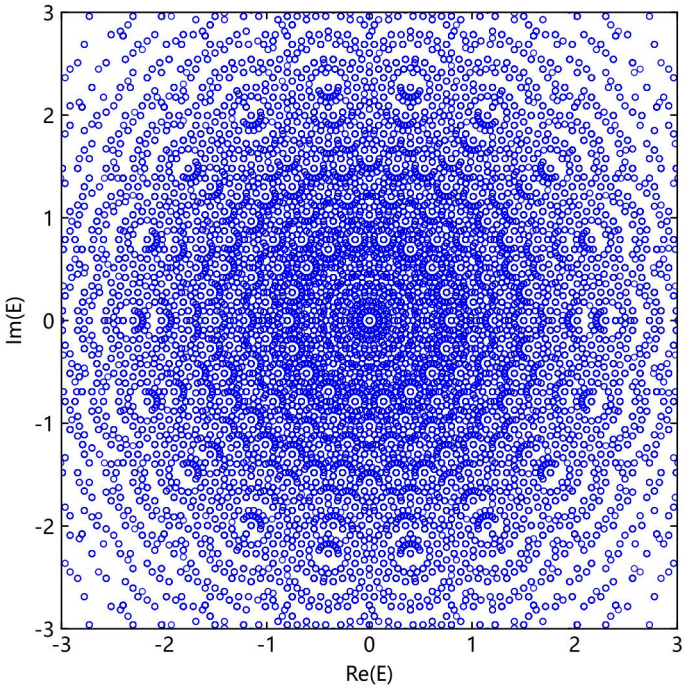
使用向量 x 作为输入，创建随机循环矩阵 A。

找出 A 的特征值，并将其存储在 E 中。

```
rng = MT19937ar(5489)
for i = 1:20000
  x = -0.4 .+ 0.8 .*randi(rng,(0,1),18)
  A = gallery("circul",x)
  E[:,i] = eigen(A).values
end
```

创建一个散点图，以在复平面上显示特征值 E。将 x 和 y 轴范围设置为 –3 至 3。

```
scatter(real(E[:]),imag(E[:]),"b.")
xlabel("Re(E)")
ylabel("Im(E)")
xlim([-3 3])
ylim([-3 3])
axis("square")
```




```
0.0   1.0   1.0  -1.0   1.0  -2.0
0.0   0.0   1.0  -1.0   1.0  -1.0  1。
```

(5). 使用 Householder 变换计算 QR 分解

此示例说明如何使用 Householder 变换来计算矩阵 A = QR 的 QR 分解，其中 Q 是正交矩阵，R 是上三角矩阵。

首先，将随机数生成器设置为默认值，并创建由服从标准正态分布的随机数组成的 6×3 矩形矩阵。

```
rng = MT19937ar(5489)
A = randn(rng,6,3)
```

3.2.1.8.11 hadamard

Hadamard 矩阵

A. 语法

```
P = hadamard(n)
```

B. 说明

P = hadamard(n)返回阶次为 n 的 Hadamard 矩阵。

C. 示例

(1). 计算 Hadamard 矩阵

计算 4×4 Hadamard 矩阵。

```
H = hadamard(4)
```

```
4×4 Matrix{Int64}:
1    1    1    1
1    -1   1    -1
1    1    -1   -1
1    -1   -1   1
```

3.2.1.8.12 hankel

Hankel 矩阵

A. 语法

```
H = hankel(c)
H = hankel(c,r)
```

B. 说明

H = hankel(c)返回正方形 Hankel 矩阵，其中 c 定义矩阵的第一列，主反对角线以下的元素为零。

H = hankel(c,r) 返回 Hankel 矩阵，第一列为 c，最后一行为 r。如果 c 的最后一个元素不同于 r 的第一个元素，则 hankel 会发出警告，并对反对角线使用 c 的最后一个元素。

C. 示例

(1). 创建对称 Hankel 矩阵

创建一个对称 Hankel 矩阵。

```
c = [1,2,3,4]
hankel(c)
```

```
ans = 4×4 Matrix{Float64}:
1.0  2.0  3.0  4.0
2.0  3.0  4.0  0.0
3.0  4.0  0.0  0.0
```

```
4.0  0.0  0.0  0.0
```

(2). 创建非对称 Hankel 矩阵

用指定的列和行向量创建一个非对称 Hankel 矩阵。

```
c = [2,4,6]
r = [6,5,4,3,2,1]
hankel(c,r)
```

```
ans = 3×6 Matrix{Float64}:
2.0  4.0  6.0  5.0  4.0  3.0
4.0  6.0  5.0  4.0  3.0  2.0
6.0  5.0  4.0  3.0  2.0  1.0
```

创建另一个非对称 Hankel 矩阵。如果列向量的最后一个元素与行向量的第一个元素不匹配，则 hankel 会发出警告，并将列的最后一个元素用作反对角线元素。

```
c = [1,2,3]
r = [4,5,7,9]
hankel(c,r)
```

```
ans = 3×4 Matrix{Float64}:
1.0  2.0  3.0  5.0
2.0  3.0  5.0  7.0
3.0  5.0  7.0  9.0
```

(3). 用复数元素创建 Hankel 矩阵

用复数行向量和列向量创建一个 Hankel 矩阵。

```
c = [1+2im,2-4im,-1+3im]
r = [-1+3im,3-im,1-2im]
hankel(c,r)
```

```
ans = 3×3 Matrix{Complex{Int64}}:
1+2im   2-4im  -1+3im
2-4im  -1+3im   3-1im
-1+3im   3-1im   1-2im
```

3.2.1.8.13 hilb

hilbert 矩阵

A. 语法

```
M = hilb(n)
```

B. 说明

M = hilb(n) 返回阶数为 n 的 Hilbert 矩阵。Hilbert 矩阵是病态矩阵的典型示例。Hilbert 矩阵的元素由 H(i,j) = 1/(i + j − 1) 指定。

C. 示例

(1). 四阶 Hilbert 矩阵

计算四阶 Hilbert 矩阵及其条件数，以查看它的病态状况。

```
H = hilb(4)
```

```
4×4 Matrix{Float64}:
1.0          0.5          0.333333     0.25
0.5          0.333333     0.25         0.2
0.333333     0.25         0.2          0.166667
0.25         0.2          0.166667     0.142857
```

```
cond(H)
```

```
15513.73873892924
```

3.2.1.8.14 invhilb

Hilbert 矩阵的逆矩阵

A. 语法

```
M = invhilb(n)
```

B. 说明

对于小于 15 的 n，H = invhilb(n) 生成确切 Hilbert 矩阵的确切逆矩阵。对于较大的 n，invhilb 函数生成逆 Hilbert 矩阵的近似值。

B. 示例

(1). Hilbert 矩阵求逆

计算 Hilbert 矩阵的四阶逆矩阵。

```
H = invhilb(4)
```

```
4×4 Matrix{Float64}:
16.0      -120.0      240.0      -140.0
-120.0    1200.0      -2700.0    1680.0
```

```
240.0     -2700.0      6480.0     -4200.0
-140.0    1680.0       -4200.0    2800.0
```

3.2.1.8.15 magic

幻方矩阵

A. 语法

```
M = magic(n)
```

B. 说明

M = magic(n) 返回由 1 到 n^2 的整数构成并且总行数和总列数相等的 n×n 矩阵。n 的阶数必须是大于或等于 3 的标量才能创建有效的幻方矩阵。

C. 示例

(1). 三阶幻方矩阵

计算三阶幻方矩阵 M。

```
M = magic(3)
```

```
3×3 Matrix{Int64}:
8     1     6
3     5     7
4     9     2
```

每列中元素的总和等于每行中元素的总和。

```
sum(M,dims = 1)
```

```
1×3 Matrix{Float64}:
15  15  15
```

```
sum(M,dims = 2)
```

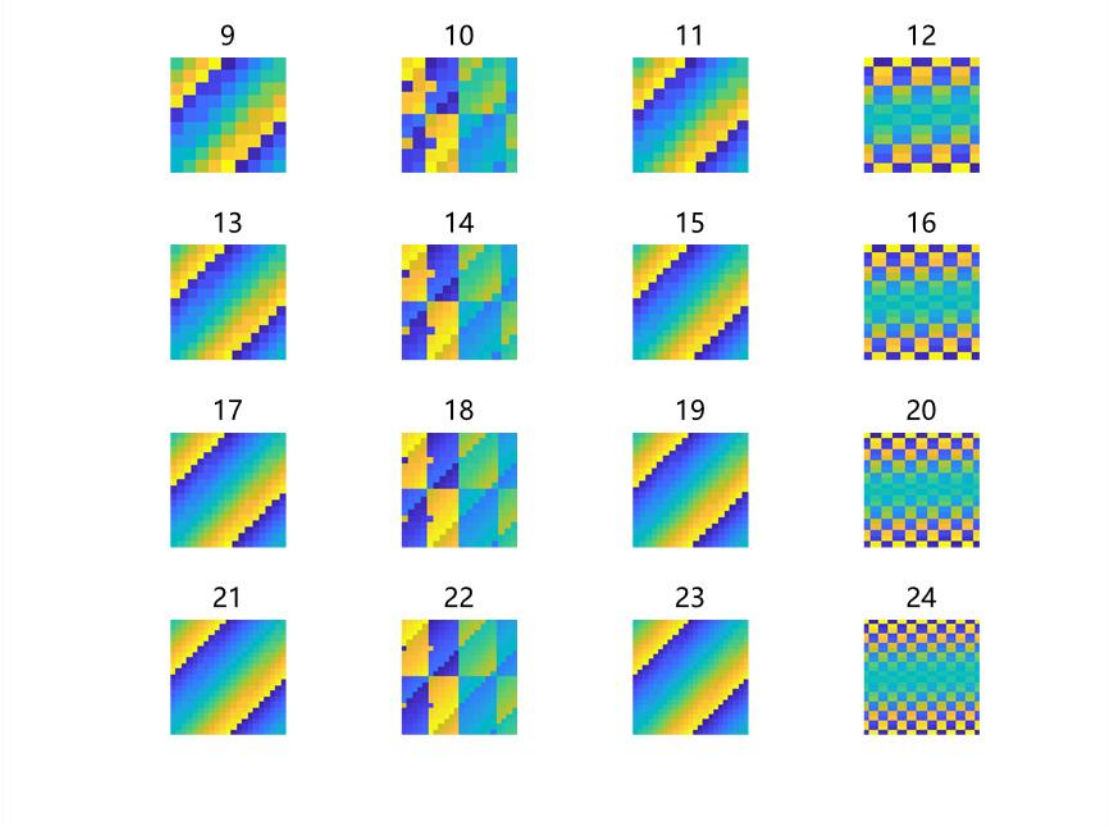
```
3×1 Matrix{Float64}:
15
15
15
```

(2). 幻方矩阵可视化

使用 imagesc 观察 9 阶到 24 阶幻方矩阵的图案。这些图案表明 magic 使用三种不同的算法，取决于 mod(n,4) 的值是 0、2 还是奇数。

```
using TyImages
```

```
for n = 1:16
ax = subplot(4,4,n)
ord = n+8
m = magic(ord)
imagesc(m)
title("$ord")
axis("equal")
axis("off")
end
```



3.2.1.8.16 pascal

帕斯卡矩阵

A. 语法

```
P = pascal(n)
P = pascal(n,1)
P = pascal(n,2)
```

B. 说明

P = pascal(n)返回 n 阶帕斯卡矩阵。P 是一个对称正定矩阵，其整数项来自帕斯卡三角形。P 的逆矩阵具有整数项。

P = pascal(n,1) 返回帕斯卡矩阵的下三角 Cholesky 因子（最高到列符号）。P 是对合矩阵，即，该矩阵是它自身的逆矩阵。

P = pascal(n,2) 返回 pascal(n,1) 的转置和置换版本。在这种情况下，P 是单位矩阵的立方根。

C. 示例

(1). 帕斯卡三角形中的矩阵

计算四阶帕斯卡矩阵。

```
A = pascal(4)
```

```
4×4 Matrix{Int64}:
1  1  1  1
1  2  3  4
1  3  6  10
1  4  10  20
```

计算三阶帕斯卡矩阵的下三角 Cholesky 因子，并验证它是对合矩阵。

```
A = pascal(3,1)
```

```
3×3 Matrix{Int64}:
1   0   0
1  -1   0
1  -2   1
```

```
inv(A)
```

```
3×3 Matrix{Float64}:
1.0   0.0  0.0
1.0  -1.0  0.0
1.0  -2.0  1.0
```

3.2.1.8.17 rosser

典型对称特征值测试问题

A. 语法

```
M = rosser(n)
```

B. 说明

M = rosser(n) 返回双精度类型的 Rosser 矩阵。

C. 示例

(1). 生成 Rosser 矩阵

rosser 返回 Rosser 矩阵。

```
M = rosser()
```

```
8×8 Matrix{Int64}:
611   196   -192   407   -8    -52   -49    29
196   899   113    -192  -71   -43   -8     -44
-192  113   899    196   61    49    8      52
407   -192  196    611   8     44    59     -23
-8    -71   61     8     411   -599  208    208
-52   -43   49     44    -599  411   208    208
-49   -8    8      59    208   208   99     -911
29    -44   52     -23   208   208   -911   99
```

3.2.1.8.18 toeplitz

托普利茨矩阵

A. 语法

```
T = toeplitz(c,r)
T = toeplitz(r)
```

B. 说明

T = toeplitz(c,r) 返回非对称托普利茨矩阵，其中 c 作为第一列，r 作为第一行。如果 c 和 r 的首个元素不同，toeplitz 将发出警告并使用列元素作为对角线。

T = toeplitz(r) 返回对称的托普利茨矩阵，其中：

- 如果 r 是实数向量，则 r 定义矩阵的第一行。
- 如果 r 是第一个元素为实数的复数向量，则 r 定义第一行，r' 定义第一列。
- 如果 r 的第一个元素是复数，则托普利茨矩阵是抽取了主对角线的 Hermitian 矩阵，这意味着对于 i≠j 的情况，T[i,j]=conj(T[j,i])。主对角线的元素会被设置为 r(1)。

C. 示例

(1). 创建对称的托普利茨矩阵

```
r = [1,2,3]
toeplitz(r)
```

```
3×3 Matrix{Int64}:
1     2     3
2     1     2
3     2     1
```

(2). 创建非对称的托普利茨矩阵

创建具有指定的列和行向量的非对称托普利茨矩阵。因为列向量和行向量的首个元素不匹配，toeplitz 发出警告并使用列作为对角线上的元素。

```
c = [1,2,3,4]
r = [4,5,6]
toeplitz(c,r)
```

```
toeplitz: column wins diagonal conflict
4×3 Matrix{Int64}:
1     5     6
2     1     5
3     2     1
4     3     2
```

创建包含复数行和列向量的托普利茨矩阵。

```
c = [1+3im,2-5im,-1+3im]
r = [1+3im,3-1im,-1-2im]
T = toeplitz(c,r)
```

```
3×3 Matrix{Complex{Int64}}:
1+3im  3-1im  -1-2im
2-5im  1+3im   3-1im
-1+3im  2-5im   1+3im
```

(3). 使用 toeplitz 函数创建循环矩阵

您可以使用 toeplitz 创建循环矩阵。循环矩阵在循环卷积等应用中使用。

使用 toeplitz 从向量 v 创建循环矩阵。

```
v = [9,1,3,2]
toeplitz([v[1];reverse(v[2:end])], v)
```

```
4×4 Matrix{Int64}:
9     1     3     2
2     9     1     3
3     2     9     1
```

```
1     3     2     9
```

通过使用 toeplitz 为卷积构建循环矩阵来执行离散时间的循环卷积。定义周期性输入 x 和系统响应 h。

```
x = [1,8,3,2,5]
h = [3,5,2,4,1]
```

构建列向量 c 以创建循环矩阵，其中 length(c) = length(h)。

```
c = [x[1] ;reverse(x[end-length(h)+2:end])]
```

```
5-element Vector{Int64}:
1
5
2
3
8
```

```
r = x
```

使用 toeplitz 构建卷积矩阵 xConv。使用 h*xConv 求卷积。

```
xConv = toeplitz(c,r)
```

```
5×5 Matrix{Int64}:
1  8  3  2  5
5  1  8  3  2
2  5  1  8  3
3  2  5  1  8
8  3  2  5  1
```

```
h'*xConv
```

```
1×5 adjoint(::Vector{Int64}) with eltype Int64:
52  50  73  46  64
```

(4). 使用 Toeplitz 的离散时间卷积

通过使用 toeplitz 为卷积构建数组，执行离散时间卷积。

定义输入 x 和系统响应 h。

```
x = [1,8,3,2,5]
h = [3,5,2]
```

通过用 0 填充 x 构建 r。r 的长度是卷积长度 x + h - 1。

```
r = vec([x;zeros(1,length(h)-1)'])
```

```
7×1 Vector{Float64}:
1.0
8.0
3.0
2.0
```

```
5.0
0.0
0.0
```

构建列向量 c。将第一个元素设置为 x(1)，因为该列确定对角线。填充 c，因为 length(c) 必须等于卷积的 length(h)。

```
c = vec([x[1];zeros(1,length(h)-1)'])
```

使用 toeplitz 构建卷积矩阵 xConv。然后，使用 h*xConv 求卷积

```
xConv = toeplitz(c,r)
```

```
3×7 Matrix{Float64}:
1.0  8.0  3.0  2.0  5.0  0.0  0.0
0.0  1.0  8.0  3.0  2.0  5.0  0.0
0.0  0.0  1.0  8.0  3.0  2.0  5.0
h'*x
```

3.2.1.8.19 vander

Vandermonde 矩阵

A. 语法

```
M = vander(v)
```

B. 说明

M = vander(v)返回 Vandermonde 矩阵 以使其列是向量 v 的幂。

C. 示例

(1). 求出向量输入的 Vandermonde 矩阵

计算 v 的 Vandermonde 矩阵。

```
v=[1:0.5:3...]
```

```
5-element Vector{Float64}:
1.0
1.5
2.0
2.5
3.0
```

```
A = vander(v)
```

```
5×5 Matrix{Float64}:
1.0         1.0        1.0       1.0     1.0
```

```
5.0625    3.375     2.25     1.5     1.0
16.0        8.0        4.0       2.0     1.0
39.0625  15.625   6.25     2.5     1.0
81.0        27.0       9.0      3.0     1.0
```

```
A = reverse(vander(v),dims=2)
```

```
5×5 Matrix{Float64}:
1.0     1.0     1.0    1.0     1.0
1.0     1.5     2.25   3.375   5.0625
1.0     2.0     4.0    8.0     16.0
1.0     2.5     6.25   15.625  39.0625
1.0     3.0     9.0    27.0    81.0
```

3.2.1.8.20 wilkinson

Wilkinson 的特征值测试矩阵

A. 语法

```
W = wilkinson(n)
```

B. 说明

M = wilkinson(n)返回 J.H.Wilkinson 的 n×n 特征值测试矩阵之一。W 是一个对称的三对角矩阵，具有几乎相等的特征值对。

C. 示例

(1). Wilkinson 测试矩阵

计算 7×7 Wilkinson 特征值测试矩阵。最常用的测试矩阵是 wilkinson(21)，它的两个最大的特征值约等于 10.746。这两个特征值的前 14 位小数位均相同，第 15 位不同。

```
W = wilkinson(7)
```

```
7×7 Matrix{Float64}:
3.0   1.0   0.0   0.0   0.0   0.0   0.0
1.0   2.0   1.0   0.0   0.0   0.0   0.0
0.0   1.0   1.0   1.0   0.0   0.0   0.0
0.0   0.0   1.0   0.0   1.0   0.0   0.0
0.0   0.0   0.0   1.0   1.0   1.0   0.0
0.0   0.0   0.0   0.0   1.0   2.0   1.0
```

```
0.0   0.0   0.0   0.0   0.0   1.0   3.0
```

3.2.2 线性代数

线性方程、特征值、奇异值、分解、矩阵运算、矩阵结构。

Syslab 中的线性代数函数提供快速且数值稳健的矩阵计算。功能包括各种矩阵分解、线性方程求解、计算特征值或奇异值等。

3.2.2.1 线性方程

解算多种类型的线性方程。

函数名	简介
inv	矩阵求逆
pinv	Moore-Penrose 伪逆
\	求解关于 x 的线性方程组 Ax = B
/	求解关于 x 的线性方程组 xA = B
linsolve	对线性方程组求解
lscov	存在已知协方差的最小二乘解
lsqnonneg	求解非负线性二乘问题
sylvester	求解关于 X 的 Sylvester 方程 AX + XB = C

3.2.2.1.1 inv

矩阵求逆

A. 语法

```
Y = inv(X)
```

B. 说明

Y = inv(X) 计算方阵 X 的 逆矩阵。

X^(-1) 等效于 inv(X)。

x = A\b 的计算方式与 x = inv(A)*b 不同，建议用于求解线性方程组。

C. 示例

(1). 逆矩阵

计算一个 3×3 矩阵的逆矩阵。

```
X = [1 0 2; -1 5 0; 0 3 -9]
```

```
X = 3×3 Matrix{Int64}:
1  0   2
-1  5   0
0  3  -9
```

```
Y = inv(X)
```

```
Y = 3×3 Matrix{Float64}:
0.882353   -0.117647    0.196078
0.176471    0.176471    0.0392157
0.0588235   0.0588235  -0.0980392
```

检查结果。理想情况下，Y*X 将生成单位矩阵。由于 inv 使用浮点计算执行矩阵求逆，因此，实际上 Y*X 接近但不完全等于单位矩阵 eye(size(X))。

```
Y*X
```

```
ans = 3×3 Matrix{Float64}
1.0  0.0  -2.22045e-16
0.0  1.0  -5.55112e-17
0.0  0.0   1.0
```

(2). 线性方程组求解

了解为何通过使用 inv(A)*b 求逆矩阵对线性方程组求解不如使用反斜杠运算符（即 x = A\b）直接求解。

创建一个 500 阶的随机矩阵 A，其条件数 cond(A) 为 1e10，并且其范数 opnorm(A) 为 1。精确解 x 是一个长度为 500 的随机向量，并且右侧为 b = A*x。因此，线性方程组未正确设置条件但一致。

```
n = 500
Q = orth(randn(n,n))
d = logspace(0,-10,n)
A = Q*diagm(d)*Q'
x = randn(n,1)
b = A*x
```

通过反转系数矩阵 A 对线性方程组 A*x = b 求解。使用@time 获取时间等信息。

```
@time y = inv(A)*b;
```

```
0.151880 seconds (443.98 k allocations: 27.503 MiB, 16.89% gc time,
92.08% compilation time)
```

3.2.2.1.2 pinv

Moore-Penrose 伪逆

A. 语法

```
B = pinv(A)
B = pinv(A,atol)
B = pinv(A,rtol)
B = pinv(A,atol,rtol)
```

B. 说明

B = pinv(A) 返回矩阵 A 的 Moore-Penrose 伪逆。

B = pinv(A,atol) 指定绝对误差的值。pinv 将 A 中小于容差的奇异值视为零。

B = pinv(A,rtol) 指定相对误差的值。

C. 示例

(1). 使用伪逆求解线性方程组

比较通过反斜杠 () 和 pinv 求得的线性方程组的解。

如果矩形系数矩阵 A 低秩，则使 norm(A*x-b) 最小化的最小二乘问题将有无限多个解。x1 = A\b 和 x2 = pinv(A)*b 返回两个解。这两个解具有以下特征：x1 只有 rank(A) 个非零分量，norm(x2) 是小于任何其他解的范数。

创建一个 rank(A) = 3 的 8×6 矩阵。

```
A = magic(8)
A = A[:,1:6]
```

```
8×6 Matrix{Int64}:
64   2   3  61  60   6
9   55  54  12  13  51
17  47  46  20  21  43
40  26  27  37  36  30
32  34  35  29  28  38
```

```
41  23  22  44  45  19
49  15  14  52  53  11
8   58  59   5   4  62
```

为方程组的右侧创建一个向量。

```
b = 260*ones(8,1)
```

```
8×1 Matrix{Float64}:
260.0
260.0
260.0
260.0
260.0
260.0
260.0
260.0
```

为右侧选择的数字 260 是 A 的 8×8 幻数和。如果 A 仍然是 8×8 矩阵，则 x 有一个解是由 1 组成的向量。如果只有六列，方程仍然是一致的，因此仍然存在解，但解并非全部由 1 组成。由于矩阵低秩，因此有无限多个解。

使用反斜杠和 pinv 求两个解。

```
x1 = A\b
```

```
6×1 Matrix{Float64}:
1.153846153846153
1.4615384615384617
1.3846153846153846
1.384615384615385
1.4615384615384626
1.1538461538461524
```

```
x2 = pinv(A)*b
```

```
6×1 Matrix{Float64}:
1.1538461538461524
1.461538461538463
1.384615384615385
1.3846153846153841
1.4615384615384635
1.1538461538461497
```

从 norm(Ax1-b) 和 norm(Ax2-b) 仅存在舍入误差的意义上来说，这两个解都是精确的。解 x1 的特殊之处在于它只有三个非零元素。解 x2 的特殊之处在于 norm(x2) 小于其他任何解的对应值，包括 norm(x1)

```
norm(x1)
```

```
ans = 3.281650616569468
```

```
norm(x2)
```

```
ans = 3.2816506165694674
```

3.2.2.1.3 \

求解关于 x 的线性方程组 Ax = B

A. 语法

```
x = A\B
x = \(A,B)
```

B. 说明

x = A\B 对线性方程组 A*x = B 求解。矩阵 A 和 B 必须具有相同的行数。示例

如果 A 是标量，那么 A\B 等于 A.\B。

如果 A 是 n×n 方阵，B 是 n 行矩阵，那么 x = A\B 是方程 A*x = B 的解（如果存在解的话）。

如果 A 是矩形 m×n 矩阵，且 m ~= n，B 是 m 行矩阵，那么 A\B 返回方程组 A*x= B 的最小二乘解。

C. 示例

(1). 方程组

对简单的线性方程组 A*x = B 求解。

```
A = magic(3)
B = [15; 15; 15]
x = A\B
```

```
3-element Vector{Float64}:
0.9999999999999998
1.0
1.0000000000000002
```

(2). 包含奇异矩阵的线性方程组

对包含奇异矩阵 A 的线性方程组 A*x = b 求解。

```
A = magic(4)
b = [34; 34; 34; 34]
x = A\b
```

```
4-element Vector{Float64}:
```

```
1.5
2.5000000000000004
-0.5000000000000004
0.5
```

```
A = [1 0; 0 0]
b = [1; 1]
x = A\b
```

```
2-element Vector{Float64}:
1.0
Inf
```

(3). 欠定方程组的最小二乘解

对线性方程组 A*x = b 求解。

```
A = [1 2 0; 0 4 3]
b = [8; 18]
x = A\b
```

```
3-element Vector{Float64}:
0.9180327868852435
3.540983606557374
1.278688524590165
```

(4). 包含稀疏矩阵的线性方程组

使用稀疏矩阵对简单的线性方程组求解。

考虑以下矩阵方程 A*x = B。

```
A = sparse([0 2 0 1 0; 4 -1 -1 0 0; 0 0 0 3 -6; -2 0 0 0 2; 0 0 4 2
0])
B = [8; -1; -18; 8; 20]
x = A\B
```

```
5-element Vector{Float64}:
1.0
2.0
3.0
3.9999999999999996
5.0
```

3.2.2.1.4 /

求解关于 x 的线性方程组 xA = B

A. 语法

```
x = B/A
x = /(B,A)
```

B. 说明

x = B/A 对线性方程组 x*A = B 求解 x。矩阵 A 和 B 必须具有相同的列数。示例

如果 A 是标量，那么 B/A 等于 B./A。

如果 A 是 n×n 方阵，B 是 n 列矩阵，那么 x = B/A 是方程 x*A = B 的解（如果存在解的话）。

如果 A 是矩形 m×n 矩阵，且 m ~= n，B 是 n 列矩阵，那么 x=B/A 返回方程组 x*A = B 的最小二乘解。

x = /(B,A) 是执行 x=B/A 这一操作的替代方法。

C. 示例

(1). 方程组

对具有唯一解的方程组 x*A = B 求解。

```
A = [1 1 3; 2 0 4; -1 6 -1]
B = [2 19 8]
x = B/A
```

```
1×3 Matrix{Float64}:
1.0  2.0  3.0
```

(2). 欠定方程组的最小二乘解

对欠定方程组 x*C = D 求解。

```
C = [1 0; 2 0; 1 0]
D = [1 2]
x = D/C
```

```
1×3 Matrix{Float64}:
4-0.166667   0.333333   0.166667 Vector{Float64}:
```

```
x*C-D
```

```
1×2 Matrix{Float64}:
2.22045e-16  -2.0
```

3.2.2.1.5 linsolve

对线性方程组求解

A. 语法

```
X = linsolve(说明A,B)
```

B. 说明

X = linsolve(A,B) 使用以下方法之一求解线性方程组 AX = B：

当 A 是方阵时，linsolve 使用 LU 分解和部分主元消去法。

对于所有其他情况，linsolve 使用 QR 分解和列主元消去法。

C. 示例

(1). 线性方程组求解

对简单的线性方程组 A*x = B 求解。

```
A = magic(3)
B = [15; 15; 15]
x =linsolve(A,B)
```

```
3-element Vector{Int64}:
1
1
1
```

(2). 包含奇异矩阵的线性方程组

对包含奇异矩阵 A 的线性方程组 A*x = b 求解。

```
A = magic(4)
b = [34; 34; 34; 34]
x = linsolve(A,b)
```

```
4-element Vector{Int64}:
2
4
-2
0
```

```
A = [1 0; 0 0]
b = [1; 1]
x = linsolve(A,b)
```

```
The system has no solution
```

(3). 欠定方程组的最小二乘解

对线性方程组 A*x = b 求解。

```
A = [1 2 0; 0 4 3]
b = [8; 18]
```

```
x = linsolve(A,b)
```

```
x = 3-element Vector{Float64}:
-1.0
4.5
-0.0
```

3.2.2.1.6 lscov

存在已知协方差的最小二乘解

A. 语法

```
x = lscov(A,B)
x = lscov(A,B,w)
x = lscov(A,B,V)
x = lscov(A,B,V,alg)
x,stdx = lscov(...)
x,stdx,mse = lscov(...)
x,stdx,mse,S = lscov(...)
```

B. 说明

x = lscov(A,B) 返回线性方程组 A*x = B 的普通最小二乘解，即 x 是最小化平方误差总和的 n×1 向量 (B - A*x)'*(B - A*x)，其中 A 是 m×n，B 是 m×1。 B 也可以是 m×k 矩阵，lscov 为 B 的每一列返回一个解。 当 rank(A) < n 时，lscov 将 x 的最大可能元素数设置为零以获得“基本解” .

x = lscov(A,B,w)，其中 w 是实数正权重的向量长度 m，返回线性系统 A*x = B 的加权最小二乘解，即 x 最小化 (B - A*x)'*diag(w)*(B - A*x)。 w 通常包含计数或逆方差。

x = lscov(A,B,V)，其中 V 是 m×m 实对称正定矩阵，返回线性系统 A*x = B 的广义最小二乘解，协方差矩阵与 V 成正比，即, x 最小化 (B - A*x)'*inv(V)*(B - A*x)。

更一般地说，V 可以是半正定的，并且 lscov 返回最小化 e'*e 的 x，受制于 A*x + T*e = B，其中最小化在 x 和 e 上，并且 T*T' = V。当 V 是半定的，这个问题只有当 B 与 A 和 V 一致时才有解（即 B 在 [AT] 的列空间中），否则 lscov 返回错误。

默认情况下，lscov 计算 V 的 Cholesky 分解，实际上，反转该因子以将问题转换为普通最小二乘法。但是，如果 lscov 确定 V 是半定的，它会使用

正交分解算法来避免反转 V。

x,stdx,mse,S = lscov(...) 返回 x 的估计协方差矩阵。当 A 秩亏时，S 在对应于 x 的必然零元素的行和列中包含零。如果 lscov 用多个右侧调用，即 size(B,2) > 1，则不能返回 S。

当 A 和 V 为满秩时，这些量的标准公式为

x = inv(A'*inv(V)*A)*A'*inv(V)*B

mse = B'*(inv(V) - inv(V)*A*inv(A'*inv(V)*A)*A'*inv(V))*B./(m-n)

S = inv(A'*inv(V)*A)*mse

stdx = sqrt.(diag(S))

但是，lscov 使用更快更稳定的方法，适用于秩亏的情况。

lscov 假设 B 的协方差矩阵仅在一个比例因子内已知。 mse 是对未知比例因子的估计，而 lscov 会适当地缩放输出 S 和 stdx。但是，如果已知 V 恰好是 B 的协方差矩阵，则该缩放是不必要的。在这种情况下，要获得适当的估计值，您应该分别按 1/mse 和 sqrt(1/mse) 重新调整 S 和 stdx。

C. 示例

(1). 计算普通最小二乘法

```
x1 = [.2 .5 .6 .8 1.0 1.1]'
x2 = [.1 .3 .4 .9 1.1 1.4]'
X = [ones(size(x1)) x1 x2]
y = [.17 .26 .28 .23 .27 .34]'
a = X\y
```

```
3×1 Matrix{Float64}:
0.12030264279624887
0.3283887468030694
-0.13120204603580593
```

```
b,se_b,mse = lscov(X,y)
```

```
b = 3×1 Matrix{Float64}:
0.12030264279624961
0.3283887468030693
-0.13120204603580676
se_b = 3×1 Matrix{Float64}:
0.064276511225875
0.2266893917770793
0.14878307645900268
mse = 1×1 Matrix{Float64}:
0.0015455953395851106
```

(2). 计算加权最小二乘法

```
x1 = [.2 .5 .6 .8 1.0 1.1]'
x2 = [.1 .3 .4 .9 1.1 1.4]'
X = [ones(size(x1)) x1 x2]
y = [.17 .26 .28 .23 .27 .34]'
w = [1 1 1 1 1 .1]'
bw,sew_b,msew = lscov(X,y,w)
bw
3×1 Matrix{Float64}:
0.10456357190763761
0.46142569703937786
-0.2620580626616847
sew_b
3×1 Matrix{Float64}:
0.03086425936498791
0.11515846970379383
0.08142933483861849
msew
1×1 Matrix{Float64}:
0.00034741464660981836
```

(3). 计算一般最小二乘法

```
x1 = [.2 .5 .6 .8 1.0 1.1]'
V           =           0.2.*ones(length(x1),length(x1))           +
0.8.*diagm(vec(ones(size(x1))))
x2 = [.1 .3 .4 .9 1.1 1.4]'
X = [ones(size(x1)) x1 x2]
y = [.17 .26 .28 .23 .27 .34]'
bg,sew_b,mseg = lscov(X,y,V)
bg = 3×1 Matrix{Float64}:
0.12030264279624905
0.32838874680306906
-0.13120204603580582
sew_b = 3×1 Matrix{Float64}:
0.06721509302430782
0.2266893917770787
0.14878307645900168
mseg = 1×1 Matrix{Float64}:
0.001931994174481386
```

(4). 估计系数协方差矩阵

```
x1 = [.2 .5 .6 .8 1.0 1.1]'
V           =           0.2.*ones(length(x1),length(x1))           +
0.8.*diagm(vec(ones(size(x1))))
x2 = [.1 .3 .4 .9 1.1 1.4]'
X = [ones(size(x1)) x1 x2]
y = [.17 .26 .28 .23 .27 .34]'
b,se_b,mse,S = lscov(X,y)
[se_b sqrt.(diag(S))]
```

```
3×2 Matrix{Float64}:
0.0642765  0.0642765
0.226689   0.226689
0.148783   0.148783
```

3.2.2.1.7 lsqnonneg

求解非负线性二乘问题

A. 语法

```
x, resnorm, residual, exitflag, output, lambda = lsqnonneg(C, d)
_ = lsqnonneg(problem)
```

B. 说明

求解以下形式的非负最小二乘曲线拟合问题$\min_x \|C \cdot x - d\|_2^2$，
其中 $x \geq 0$

x, resnorm, residual, exitflag, output, lambda = lsqnonneg(C, d) 返回在 $x \geq 0$ 的约束下，使得 norm(C*x-d) 的最小的向量 x 、残差的 2-范数 平方值 resnorm = norm(C*x-d)^2 、残差 residual = d-C*x 、描述 lsqnonneg 的退出条件的值 exitflag 、提供优化过程信息的结构体 output 以及拉格朗日乘量 lambda 。如仅需返回部分值，可以使用 [] 进行指定，例如 x = lsqnonneg(C,d)[1] （下同）。参数 C 和 d 必须为实数。示例

_ = lsqnonneg(problem) 求 problem 的最小值，它是一个结构体。

C. 示例

(1). 非负线性最小二乘

计算线性最小二乘问题的非负解，并将结果与无约束问题的解进行比较。

为问题$\min_x\|Cx-d\|$准备 C 矩阵和 d 向量。

```
C = [0.0372 0.2869
    0.6861 0.7071
    0.6233 0.6245
    0.6344 0.6170]
d = [0.8587
    0.1781
    0.0747
    0.8405]
```

计算有约束和无约束的解。

```
x = lsqnonneg(C, d)[1]
```

```
x = 2-element Vector{Float64}:
 0.0
 0.6929343971302934
```

```
xunc = C\d
```

```
xunc = 2-element Vector{Float64}:
 -2.562746541782972
  3.110776613164541
```

x 中的所有项均为非负数，但 xunc 中某些项为负数。

计算两个解的残差的范数。

```
constrained_norm = norm(C*x-d)
```

```
constrained_norm = 0.9118420648683149
```

```
unconstrained_norm = norm(C*xunc-d)
```

```
unconstrained_norm = 0.667439909369895
```

无约束解的残差范数较小，因为约束只会增加残差范数。

(2). 从非负最小二乘获取残差

以带输出的方式调用 lsqnonneg ，以获取解、残差范数和残差向量。

为问题$\min_x\|Cx-d\|$准备 C 矩阵和 d 向量。

```
C = [0.0372 0.2869
    0.6861 0.7071
    0.6233 0.6245
    0.6344 0.6170]
d = [0.8587
    0.1781
    0.0747
    0.8405]
```

获取解和残差信息。

```
x, resnorm, residual = lsqnonneg(C, d)[1:3]
```

```
x = 2-element Vector{Float64}:
 0.0
 0.6929343971302934
resnorm = 0.8314559512633122
residual = 4-element Vector{Float64}:
  0.6598971214633189
 -0.3118739122108304
 -0.3580375310078683
  0.41295947697060903
```

验证返回的残差范数是返回的残差向量范数的平方。

```
norm(residual)^2
```

```
ans = 0.8314559512633122
```

(3). 检查非负最小二乘的结果

在 lsqnonneg 完成之后，请求所有输出参数，以检查解和求解过程。

为问题$\min_x \|Cx - d\|$准备 C 矩阵和 d 向量。

```
C = [0.0372 0.2869
    0.6861 0.7071
    0.6233 0.6245
    0.6344 0.6170]
d = [0.8587
    0.1781
    0.0747
    0.8405]
```

解算该问题，并请求所有输出参数。

```
x, resnorm, residual, exitflag, output, lambda = lsqnonneg(C, d)
```

```
x = 2-element Vector{Float64}:
 0.0
 0.6929343971302934
resnorm = 0.8314559512633122
residual = 4-element Vector{Float64}:
  0.6598971214633189
 -0.3118739122108304
 -0.3580375310078683
  0.41295947697060903
exitflag = 1
output = NNLSOutput(1, "active-set")
lambda = 2-element Vector{Float64}:
 -0.15061181913646518
  0.0
```

exitflag 为 1 ，表示正确的解。

x[1] = 0 ，对应的 lambda[1] ≠ 0，表示正确的对偶性。同样， x[2] > 0 ，对应的 lambda[2] = 0 。

3.2.2.1.8 sylvester

求解关于 X 的 Sylvester 方程 AX + XB = C

A. 语法

```
X = sylvester(A,B)
```

B. 说明

X = sylvester(A,B,C) 将求解 X 返回到 Sylvester 方程。

输入 A 是 m×m 矩阵，输入 B 是 n×n 矩阵，C 和 X 都是 m×n 矩阵。

C. 示例

(1). 对包含 3×3 输出的 Sylvester 方程求解

创建系数矩阵 A 和 B。

```
A = [1 -1 1; 1 1 -1; 1 1 1]
B = magic(3)
```

将 C 定义为 3×3 单位矩阵。

```
C = Matrix(Int.(eye(3)))
```

使用 sylvester 函数根据以上 A、B 和 C 值求解 Sylvester 方程。

```
X = sylvester(A,B,C)
```

```
3×3 Matrix{Float64}:
-0.122314    0.0724708  -0.013112
0.0805658   0.0161487  -0.158698
0.0164321  -0.178353    0.10723
```

结果为一个 3×3 矩阵。

(2). 对包含 4×2 输出的 Sylvester 方程求解

创建一个 4×4 系数矩阵 A 和一个 2×2 系数矩阵 B。

```
A = [1 0 2 3; 4 1 0 2; 0 5 5 6; 1 7 9 0]
B = [0 -1; 1 0]
```

将 C 定义为与 A 和 B 的相应大小匹配的 4×2 矩阵。

```
C = [1 0; 2 0; 0 3; 1 1]
```

```
4×2 Matrix{Int64}:
1  0
2  0
0  3
1  1
```

使用 sylvester 函数根据以上 A、B 和 C 值求解 Sylvester 方程。

```
X = sylvester(A,B,C)
```

```
4×2 Matrix{Float64}:
-0.473164    0.366425
0.400567    -0.353115
-0.330533    0.11422
-0.0773985   -0.35601
```

结果为一个 4×2 矩阵。

3.2.2.2 特征值和奇异值

特征值、特征向量和奇异值的计算。

函数名	简介
eigvals	返回 A 的特征值
eigvecs	返回 A 的特征向量
eigen	计算 A 的特征值分解
balance	对角线缩放以提高特征值准确性
svd	计算 A 的奇异值分解 (SVD)
svdvals	按降序返回 A 的奇异值
svdsketch	计算低秩矩阵草图的 SVD
ordeig	拟三角矩阵的特征值
ordschur	矩阵 A = ZTZ' 的 Schur 分解 F 重新排
polyeig	多项式特征值问题
hessenberg	矩阵的 Hessenberg 形式

schur	Schur 分解
schur!	Schur 分解
rsf2csf	将实数 Schur 形式变换为复数 Schur 形式
cdf2rdf	拟三角矩阵的特征值

3.2.2.2.1 eigvals

返回 A 的特征值

对于一般的非对称矩阵，可以在特征值计算之前指定矩阵的平衡方式。

permute、scale 和 sortby 关键字与 eigen! 相同。

A. **语法**

```
eigvals(A)
eigvals(A,B)
eigvals(A::Union{SymTridiagonal,       Hermitian,       Symmetric},
irange::UnitRange)
eigvals(A::Union{SymTridiagonal,  Hermitian,  Symmetric},  vl::Real,
vu::Real)
```

B. **说明**

eigvals(A)对于标量输入，eigvals 将返回一个标量.

eigvals(A,B)计算 A 和 B 的广义特征值。

eigvals(A::Union{SymTridiagonal, Hermitian, Symmetric}, irange::UnitRange) 返回 A 的特征值。通过指定一个 UnitRange irange 覆盖排序特征值的索引，可以只计算特征值的一个子集，例如第二到第八个特征值。

eigvals(A::Union{SymTridiagonal, Hermitian, Symmetric}, vl::Real, vu::Real) 返回 A 的特征值。可以通过为特征值的上下边界指定一对 vl 和 vu 来仅计算特征值的子集

C. **示例**

(1). 标量

计算-2 的特征值

```
eigvals(-2)
```

```
ans = -2
```

对于标量输入，eigvals 将返回一个标量。

(2). 广义特征值

```
A = [1 0; 0 -1]
B = [0 1; 1 0]
```

```
A = 2×2 Matrix{Int64}:
1   0
0  -1
B = 2×2 Matrix{Int64}:
0  1
1  0
```

计算 A,B 的广义特征值

```
eigvals(A,B)
```

```
ans = 2-element Vector{ComplexF64}:
0.0 - 1.0im
0.0 + 1.0im
```

(3). 计算特征值的子集

```
A = SymTridiagonal([1.; 2.; 1.], [2.; 3.])
eigvals(A, 2:2)
```

```
ans =
1-element Vector{Float64}:
0.9999999999999996
```

```
eigvals(A)
```

```
ans = 3-element Vector{Float64}:
-2.1400549446402604
1.0000000000000002
5.140054944640259
```

3.2.2.2.2 eigvecs

返回 A 的特征向量

对于一般的非对称矩阵，可以在特征值计算之前指定矩阵的平衡方式。permute、scale 和 sortby 关键字与 eigen! 相同。

A. 语法

```
eigvecs(A)
eigvecs(A,B)
```

B. 说明

eigvecs(A)对于标量输入，eigvecs 将返回 1×1 矩阵。

eigvecs(A)对于矩阵输入，eigvecs 将返回特征向量。

eigvecs(A,B)计算 A 和 B 的广义特征向量。

C. 示例

(1). 标量

计算-2 的特征向量

```
eigvecs(-2)
```

```
ans = 1×1 Matrix{Int64}:
1
```

对于标量输入，eigvecs 将返回一个标量。

(2). A 的特征向量

```
A = SymTridiagonal([1.; 2.; 1.], [2.; 3.])
eigvecs(A)
```

```
ans = 3×3 Matrix{Float64}:
0.418304    -0.83205       0.364299
-0.656749   -7.39009e-16    0.754109
0.627457    0.5547         0.546448
```

(3). 广义特征向量

```
A = [1 0; 0 -1]
B = [0 1; 1 0]
```

```
A = 2×2 Matrix{Int64}:
1   0
0  -1
B = 2×2 Matrix{Int64}:
0  1
1  0
```

计算 A,B 的广义特征向量

```
eigvecs(A,B)
```

```
ans = 2×2 Matrix{ComplexF64}:
0.0+1.0im   0.0-1.0im
-1.0+0.0im  -1.0-0.0im
```

3.2.2.2.3 eigen

计算 A 的特征值分解，返回一个特征分解对象 F，其中包含 F.values 中的特征值和矩阵 F.vectors 列中的特征向量。 (第 k 个特征向量可以从 F.vectors[:, k] 中获得)

对于一般的非对称矩阵，可以在特征向量计算之前指定矩阵的平衡方式。选项 permute=true 使矩阵更接近于上三角矩阵，并且 scale=true 按其对角线元素缩放矩阵，使行和列的范数更相等。这两个选项的默认值都是 true。

默认情况下，特征值和向量按字典顺序按 (real(λ),imag(λ)) 排序。可以将不同的比较函数 by(λ) 传递给 sortby，或者您可以传递 sortby=nothing 以离开任意顺序的特征值。一些特殊的矩阵类型（例如 Diagonal 或 SymTridiagonal）可能会实现自己的排序约定并且不接受 sortby 关键字。

A. 语法

```
eigen(A)
eigen(A,B)
```

B. 说明

eigen(A)计算 A 的特征值分解。

eigen(A,B)计算 A 和 B 的广义特征值分解。

C. 示例

(1). 特征值分解

计算矩阵的特征值分解

```
F = eigen([1.0 0.0 0.0; 0.0 3.0 0.0; 0.0 0.0 18.0])
```

```
F = Eigen{Float64, Float64, Matrix{Float64}, Vector{Float64}}
values:
3-element Vector{Float64}:
1.0
3.0
18.0
vectors:
3×3 Matrix{Float64}:
1.0  0.0  0.0
0.0  1.0  0.0
0.0  0.0  1.0
```

```
values  = F.values
vector = F.vectors
```

```
values =
3-element Vector{Float64}:
1.0
3.0
18.0
vector
3×3 Matrix{Float64}:
1.0  0.0  0.0
0.0  1.0  0.0
0.0  0.0  1.0
```

(2). 广义特征值分解

```
A = [1 0; 0 -1]
B = [0 1; 1 0]
```

```
A = 2×2 Matrix{Int64}:
1   0
0  -1
B = 2×2 Matrix{Int64}:
0  1
1  0
```

计算 A,B 的广义特征值

```
vals,vecs=eigen(A,B)
```

```
vals =
2-element Vector{ComplexF64}:
0.0 - 1.0im
0.0 + 1.0im
vecs =
2×2 Matrix{ComplexF64}:
0.0+1.0im   0.0-1.0im
-1.0+0.0im  -1.0-0.0im
```

3.2.2.2.4 eigen

计算 A 的特征值分解，返回一个特征分解对象 F，其中包含 F.values 中的特征值和矩阵 F.vectors 列中的特征向量。 (第 k 个特征向量可以从 F.vectors[:, k] 中获得)

对于一般的非对称矩阵，可以在特征向量计算之前指定矩阵的平衡方式。

选项 permute=true 使矩阵更接近于上三角矩阵，并且 scale=true 按其对角线元素缩放矩阵，使行和列的范数更相等。这两个选项的默认值都是 true。

默认情况下，特征值和向量按字典顺序按 (real(λ),imag(λ)) 排序。可以将不同的比较函数 by(λ) 传递给 sortby，或者您可以传递 sortby=nothing 以离开任意顺序的特征值。一些特殊的矩阵类型（例如 Diagonal 或 SymTridiagonal）可能会实现自己的排序约定并且不接受 sortby 关键字。

A. 语法

```
eigen(A)
eigen(A,B)
```

B. 说明

eigen(A)计算 A 的特征值分解。

eigen(A,B)计算 A 和 B 的广义特征值分解。

C. 示例

(3). 特征值分解

计算矩阵的特征值分解

```
F = eigen([1.0 0.0 0.0; 0.0 3.0 0.0; 0.0 0.0 18.0])
```

```
F = Eigen{Float64, Float64, Matrix{Float64}, Vector{Float64}}
values:
3-element Vector{Float64}:
1.0
3.0
18.0
vectors:
3×3 Matrix{Float64}:
1.0  0.0  0.0
0.0  1.0  0.0
0.0  0.0  1.0
```

```
values  = F.values
vector = F.vectors
```

```
values =
3-element Vector{Float64}:
1.0
3.0
18.0
```

```
vector
3×3 Matrix{Float64}:
1.0  0.0  0.0
0.0  1.0  0.0
0.0  0.0  1.0
```

(4). 广义特征值分解

```
A = [1 0; 0 -1]
B = [0 1; 1 0]
```

3.2.2.2.5 balance

对角线缩放以提高特征值准确性。

A. 语法

```
B, T, S, P = balance(A)
B, T, S, P = balance(A,noperm)
```

B. 说明

B, T, S, P = balance(A) 返回相似变换矩阵 T 、平衡矩阵 B，以使 B = T\A*T 和 B 具有尽可能接近的、近似相等的行列范数，以及缩放向量 S 和置换向量 P。这四者有关系 T[:,P] = diag(S) 和 B(P,P) = diag(1./S)*A*diag(S) 。T 是对角矩阵的置换矩阵，其元素是 2 的整数次幂，可防止引入舍入误差。如果 A 是对称的，则 B == A 和 T 是单位矩阵。示例

B, T, S, P = balance(A,noperm) 指定非置换指标，若该值为 true 则缩放 A，而不会置换其行和列。

C. 示例

(1). 矩阵对角线缩放以提高特征值准确性

此示例说明基本思路。矩阵 A 右上角的元素较大，左下角的元素较小。该矩阵非常不对称。

```
A = [1 100 10000;.01 1 100;.0001 .01 1]
```

```
A = 3×3 Matrix{Float64}:
 1.0     100.0   10000.0
 0.01      1.0     100.0
 0.0001    0.01      1.03×3 Matrix{Float64}:
-0.490395  -0.548502   -0.677243
```

```
 0.828831  -0.0533533  -0.55695
 0.269355  -0.834445    0.48078
```

平衡处理会生成一个对角矩阵 T（包含是 2 的幂的元素）和一个平衡矩阵 B（比 A 更对称）

```
B,T, = balance(A)
U B = 3×3 Matrix{Float64}:

 1.0     1.5625  1.2207
 0.64    1.0     0.78125
 0.8192  1.28    1.0
T = 3×3 Matrix{Float64}:
 512.0  0.0  0.0
   0.0  8.0  0.0
   0.0  0.0  0.0625
```

要查看特征向量的作用，请先计算 A 的特征向量，如此处的 V 的列所示。

```
V = eigen(A).vectors
A = 4×2 Matrix{Int64}:
 1  2
V = 3×3 Matrix{ComplexF64}:
  -0.999946-0.0im         -0.999946+0.0im          0.99995+0.0im
 0.00585216-0.00854613im  0.00585216+0.00854613im  0.0099995+0.0im
  4.1473e-5+8.54613e-5im   4.1473e-5-8.54613e-5im  9.9995e-5+0.0im
```

请注意，所有三个向量的第一个分量都是最大的。这说明 V 的条件不当；实际上 cond(V) 为 9442.981991064044。然后，查看 B 的特征向量。

```
V = eigen(B).vectors
V = 3×3 Matrix{ComplexF64}:
 -0.699264-0.0im       -0.699264+0.0im       0.69326+0.0im
  0.261915-0.382485im   0.261915+0.382485im  0.443687+0.0im
  0.237586+0.48958im    0.237586-0.48958im   0.567919+0.0im
```

现在，特征向量将正常运行，并且 cond(V) 为 1.483047846695193。条件不当数在缩放矩阵中进行了压缩；cond(T) 为 8192.0。

本示例较小且并未进行大幅缩放，因此，计算的 A 和 B 的特征值在舍入误差界限内一致；平衡几乎不会对计算结果产生任何影响

3.2.2.2.6 svd

计算 A 的奇异值分解 (SVD) 并返回一个 SVD 对象

U、S、V 和 Vt 可以从带有 FU、FS、FV 和 F.Vt 的因式分解 F 中获得，

使得 A = U * Diagonal(S) * Vt。该算法产生 Vt，因此 Vt 的提取效率更高比 V.S 中的奇异值按降序排序。迭代分解产生分量 U、S 和 V。

如果 alg = DivideAndConquer()，则使用分治算法来计算 SVD。另一个（通常较慢但更准确）选项是 alg = QIteration()。

A. 语法

```
svd(A;full = false)
svd(A,B)
```

B. 说明

svd(A,full = true)计算 A 的"满"奇异值分解。

svd(A,B)计算 A 和 B 的广义奇异值分解。

svd(A) 使用上述任一输出参数组合生成 A 的精简分解。如果 A 是 m×n 矩阵，则：

m > n - 只计算 U 的前 n 列，S 是一个 n×n 矩阵。

m = n - svd(A) 等效于 svd(A)。

m < n - 只计算 V 的前 m 列，S 是一个 m×m 矩阵。

C. 示例

(1). 奇异值分解

计算矩阵的奇异值分解

```
rng = MT19937ar(1234)
A = rand(rng,4,3);
F = svd(A,full = true)
```

```
SVD{Float64, Float64, Matrix{Float64}}
U factor:
4×4 Matrix{Float64}:
 -0.571603  -0.755292   -0.309798  -0.0826452
 -0.511633   0.0239883   0.807393   0.29287
 -0.297185   0.269767    0.132409  -0.906297
 -0.568491   0.596812   -0.484366   0.293296
singular values:
3-element Vector{Float64}:
 2.04798165466409
 0.5514593304311445
 0.447409465446397
Vt factor:
```

```
3×3 Matrix{Float64}:
 -0.490395  -0.548502   -0.677243
  0.828831  -0.0533533  -0.55695
  0.269355  -0.834445    0.48078
```

(2). 广义奇异值分解

计算 A 和 B 的广义 SVD，返回一个 GeneralizedSVD 分解对象 F，使得 [A;B] = [F.U * F. D1; F.V * F.D2] * F.R0 * F.Q'

- U 是 M×M 正交矩阵，
- V 是 P×P 正交矩阵，
- Q 是 N×N 正交矩阵，
- D1 是一个 M×(K+L) 对角矩阵，前 K 个条目为 1，
- D2 是一个 P×(K+L) 矩阵，其右上角的 L×L 块是对角线，
- R0 是一个 (K+L)×N 矩阵，其最右边的 (K+L)×(K+L) 块是非奇异上块三角形，

K+L 是矩阵 [A;B]。

迭代分解产生分量 U、V、Q、D1、D2 和 R0。

```
A = randn(3,2);
B = randn(4,2);
F = svd(A, B);
U,V,Q,C,S,R = F
```

3.2.2.2.7 svdvals

按降序返回 A 的奇异值

从 A 和 B 的广义奇异值分解中返回广义奇异值。另见 svd。

A. 语法

```
svdvals(A)
svdvals(A,B)
```

B. 说明

svdvals(A)按降序返回 A 的奇异值。

svdvals(A,B)从 A 和 B 的广义奇异值分解中返回广义奇异值。

C. 示例

(1). 奇异值

计算矩阵奇异值。

```
A = [1. 0. 0. 0. 2.; 0. 0. 3. 0. 0.; 0. 0. 0. 0. 0.; 0. 2. 0. 0. 0.]
svdvals(A)
```

```
ans =  4-element Vector{Float64}:
3.0
2.23606797749979
2.0
0.0
```

(2). 广义奇异值

```
A = [1. 0.; 0. -1.]
```

```
A = 2×2 Matrix{Float64}:
1.0   0.0
0.0  -1.0
```

```
B = [0. 1.; 1. 0.]
```

```
B = 2×2 Matrix{Float64}:
0.0  1.0
1.0  0.0
```

计算 A,B 的广义奇异值

```
svdvals(A, B)
```

```
ans = 2-element Vector{Float64}:
1.0
1.0
```

3.2.2.2.8 svdsketch

计算低秩矩阵草图的 SVD

A. 语法

```
svdsketch(
    A::AbstractArray{<:Number},
    tol::Number=eps(eltype(A))^(1 / 4);
    maxsubspacedimension::Integer=minimum(size(A)),
    blocksize::Integer=min(max(floor(Int, 0.1 * size(A, 1)), 5), maxsubspacedimension),
    maxiterations::Integer=ceil(Int, maxsubspacedimension / blocksize),
    numpoweriterations=1,
```

```
)
```

B. 说明

U,S,V, = svdsketch(A) 返回输入矩阵 A 的低秩矩阵草图的奇异值分解(SVD)。矩阵草图是一种低秩逼近，仅反映 A 的最重要特征（最大容差）。与使用 svd 相比，它能够更快地计算大型矩阵的部分 SVD。示例

U,S,V, = svdsketch(A,tol) 指定矩阵草图的容差。svdsketch 使用 tol 以自适应方式确定矩阵草图逼近的秩。随着容差变大，矩阵草图中使用的 A 的特征越来越少。示例

U,S,V, = svdsketch(A,tol;maxiterations,numpoweriterations,blocksize) 指定具有一个或多个 Name,Value 对组参数的其他选项。例如，您可以指定 MaxIterations 和标量来调整用于形成矩阵草图的迭代次数。示例

U,S,V, apxErr= svdsketch(___) 还返回向量 apxErr，该向量包含每次迭代的相对逼近误差 norm(U*S*V'-A)/norm(A)。向量 apxErr[end] 中的最后一项是 svdsketch 返回的输出的相对逼近误差。示例。

C. 示例

(1). 计算矩阵草图的 SVD

使用 svdsketch 计算低秩矩阵逼近的 SVD 因子。

使用 gallery 创建一个具有几何分布奇异值的 200×200 随机矩阵。

```
A = gallery("randsvd",200);
```

使用 svdsketch 计算 A 的低秩逼近的 SVD。

```
U,S,V, = svdsketch(A);
```

检查输出的大小。

```
size(S)
```

```
(120, 120)
```

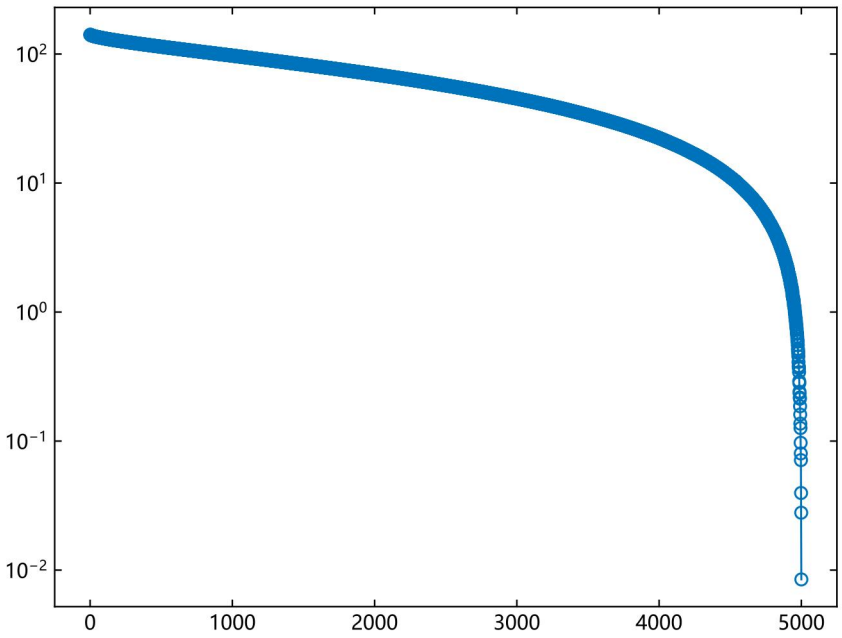
结果表明 A 的低秩矩阵逼近的秩为 120。

(2). 指定矩阵草图容差

对于具有缓慢衰减的奇异值的矩阵，将 svdsketch 与 MaxSubspaceDimension 选项结合使用。您可以使用此选项强制 svdsketch 在矩阵草图中使用 A 的特征子集。

用服从标准正态分布的值创建一个 5000×5000 矩阵。查看矩阵奇异值的分布

```
rng = MT19937ar(5489)
A = randn(rng,5000,5000);
semilogy(svdvals(A),"-o")
```



由于 A 中的奇异值缓慢衰减，因此 A 具有许多重要特征，并且使自身不适用于低秩逼近。要形成合理逼近 A 的矩阵草图，需要保留大部分或几乎所有特征。

对矩阵使用 svdsketch，容差为 1e-5。指定四个输出以返回每个迭代中的 SVD 因子以及相对逼近误差。

```
tol = 1e-5;
U1,S1,V1,apxError1 = svdsketch(A,tol);
size(S1)
```

```
ans = (5000, 5000)
```

S 的大小表明，为了满足容差，svdsketch 需要保留 A 的所有特征。对于大型稀疏输入矩阵，这可能会导致内存问题，因为由 svdsketch 形成的 A 的低秩逼近与 A 的大小大致相同，从而可能无法作为稠密矩阵放入内存中。检查输出的逼近误差。由于 svdsketch 保留 A 中的所有内容，计算出的解是准确的，但计算 svd(X) 的成本非常高。

```
apxError1[end]
```

```
ans = 0.0
```

现在，进行相同的计算但将 MaxSubspaceDimension 指定为 650，以限制用于缩略 A 的子空间的大小。这有助于强制 svdsketch 仅使用 A 中的特

征子集来形成矩阵草图，从而减小输出的大小。

```
U2, S2, V2, apxError2 =svdsketch(A, tol, maxsubspacedimension=650);
size(S2)
```

```
ans = (650, 650)
```

现在，输出比原来小了。

检查新输出的逼近误差。强制输出变小的折衷是，矩阵草图中需要省略 A 的许多重要特征，得到的 A 的秩 650 逼近不满足指定的容差。

```
apxError2[end]
```

```
ans = 0.8214198656827905
```

3.2.2.2.9 ordeig

拟三角矩阵的特征值

A. 语法

```
E = ordeig(T)
E = ordeig(AA,BB)
```

B. 说明

E = ordeig(T) 接受一个拟三角 Schur 矩阵 T（通常由 schur 生成），并按特征值沿 T 的对角线自上而下出现的顺序返回特征值向量 E。

E = ordeig(AA,BB) 接受一个拟三角矩阵对 AA 和 BB（通常由 qz 生成），并按广义特征值沿 AA-λ*BB 的对角线自上而下出现的顺序返回广义特征值。

C. 示例

(1). 示例 1

计算矩阵奇异值

```
T=diagm([1 ,-1 ,3 ,-5 ,2])
```

ordeig(T) 按特征值在对角线上显示的顺序返回 T 的特征值。

```
ordeig(T)
```

```
ans =
5×1 Matrix{Number}:
1.0
-1.0
3.0
-5.0
```

```
2.0
```

而 eig(T) 则按模逐步增加的顺序返回特征值。

```
eigvals(T)
```

```
5-element Vector{Float64}:
-5.0
-1.0
1.0
2.0
3.0
```

(2). 示例 2

```
A = rand(10,10)
U, T = schur(A)
abs.(ordeig(U))
```

```
ans =
10×1 Matrix{Float64}:
4.801984681352989
0.7290946975425873
0.845080184730285
0.8450801847302852
0.6149924365933572
0.6149924365933572
0.42093632898849254
0.4209363289884926
0.347333728415702
0.34733372841570204
```

3.2.2.2.10 ordschur

矩阵 A = Z*T*Z' 的 Schur 分解 F 重新排序

A. 语法

```
F::Schur                    =                    ordschur(F::Schur,
select::Union{Vector{Bool},BitVector})
F::GeneralizedSchur         =         ordschur(F::GeneralizedSchur,
select::Union{Vector{Bool},BitVector})
```

B. 说明

F::Schur = ordschur(F::Schur, select::Union{Vector{Bool},BitVector}) 根据返回重新排序的因式分解 F 对象的逻辑数组选择对矩阵 A = Z*T*Z' 的 Schur 因

式分解 F 重新排序。 所选特征值出现在 F.Schur 的前导对角线上，F.vectors 的相应前导列形成相应右不变子空间的正交/酉基。 在实际情况下，必须通过 select 包含或排除一对复数共轭特征值 示例

F::GeneralizedSchur=ordschur(F::GeneralizedSchur,select::Union{Vector{Bool}, BitVector}) 根据逻辑数组 select 对矩阵对 (A, B) = (Q*S*Z', Q*T*Z') 的广义 Schur 分解 F 重新排序，返回一个 GeneralizedSchur 对象 F。选择的特征值出现在 F.S 和 F.T 的前导对角线，以及左右正交/酉 Schur 向量也被重新排序，使得 (A, B) = F.Q*(F.S, F.T)*F.Z' 仍然成立，并且 A 和 B 的广义特征值可以仍可通过 F.α./F.β 获得。。

C. 示例

(1). 重新排序 Schur 分解

计算一个矩阵的 Schur 分解因子，然后根据指定的特征值顺序对因子进行重新排序。

计算矩阵 X 的 Schur 分解。Schur 分解生成上拟三角矩阵 T 和酉矩阵 U，从而使得 X=UTU'

```
X = magic(6);
F1 = schur(X)
```

```
F1 = Schur{Float64, Matrix{Float64}}
T factor:
6×6 Matrix{Float64}:
 111.0    1.72048e-14    -1.66732e-14     1.66958e-14     8.9994e-15
6.42083e-15
   0.0   27.0            6.21725e-15   -28.3164       -15.5885       -
13.0454
   0.0    0.0          -27.0           18.0           22.0454      -12.7279
   0.0    0.0            0.0            9.79796      -1.54869e-14    6.9282
   0.0    0.0            0.0            0.0           -1.22995e-14
1.24625e-14
   0.0    0.0            0.0            0.0            0.0           -9.79796
Z factor:
6×6 Matrix{Float64}:
 0.408248  -0.288675   0.408248   0.574915   0.5          -0.0529729
 0.408248   0.57735    0.408248  -0.333333   2.05873e-16  -0.471405
 0.408248  -0.288675   0.408248  -0.241582  -0.5           0.524377
 0.408248   0.288675  -0.408248   0.574915  -0.5          -0.0529729
 0.408248  -0.57735   -0.408248  -0.333333  -1.80047e-16  -0.471405
 0.408248   0.288675  -0.408248  -0.241582   0.5           0.524377
```

```
eigenvalues:
6-element Vector{Float64}:
 111.00000000000007
  27.00000000000004
 -27.000000000000004
   9.797958971132674
  -1.2299459029554988e-14
  -9.79795897113271
```

由于 T 是三角矩阵，因此 T 的对角线包含原始矩阵 X 的特征值。

对 Schur 分解进行重新排序，使特征值位于两个组中，负特征值组在 TS 的对角线上先出现。

```
select = F1.values .< 0
F = ordschur(F1,select)
```

```
F = Schur{Float64, Matrix{Float64}}
T factor:
6×6 Matrix{Float64}:
 -27.0  -19.0919      18.6997     1.85708e-14   -9.78876      12.7279
   0.0   -1.22995e-14  -0.379968    1.41001e-14  -15.6493       15.5227
   0.0    0.0         -9.79796     3.3969e-15    -2.4773        8.71848
   0.0    0.0          0.0       111.0           1.35825e-14    1.41279e-
14
   0.0    0.0          0.0         0.0          27.0         -26.36
   0.0    0.0          0.0         0.0           0.0           9.79796
Z factor:
6×6 Matrix{Float64}:
  0.408248  -0.288675   0.274621  0.408248  -0.482644    0.524377
  0.408248  -0.288675   0.298996  0.408248   0.521261   -0.471405
  0.408248   0.57735   -0.573617  0.408248  -0.0386168  -0.0529729
 -0.408248   0.288675   0.207451  0.408248   0.515098    0.524377
 -0.408248   0.288675   0.366165  0.408248  -0.476481   -0.471405
 -0.408248  -0.57735   -0.573617  0.408248  -0.0386168  -0.0529729
eigenvalues:
6-element Vector{Float64}:
 -27.000000000000004
  -1.2299459029554988e-14
  -9.79795897113271
 111.00000000000007
  27.00000000000004
   9.797958971132674
```

3.2.2.2.11 polyeig

多项式特征值问题

A. 语法

```
x,e = polyeig(A0,A1,..,AP)
```

B. 说明

X,e = polyeig(A0,A1,...,Ap) 还会返回大小为 n×n*p 的矩阵 X，其列是特征向量。

C. 示例

(1). 质量、阻尼和刚度矩阵的二次特征值问题

解算涉及质量矩阵 M、阻尼矩阵 C 和刚度矩阵 K 的二次特征值问题。运动方程中会发生此二次特征值问题：

$$M\frac{d^2y}{dt^2}+C\frac{dy}{dt}+Ky=f(t)$$

此方程适用于各种振荡系统，包括动态质量弹簧系统或 RLC 电子网络。基本解是 y(t)=xe^λt，因此 λ 和 x 都必须解算二次特征值问题 (QEP)，

$$\left(M\lambda^2+C\lambda+K\right)x=0$$

创建系数矩阵 M、C 和 K 来表示具有四个自由度的质量弹簧系统。系数矩阵全部是对称矩阵和半正定矩阵，M 是对角矩阵。

```
M = diagm([3 ,1 ,3 ,1])
```

```
M =
4×4 Matrix{Int64}:
 3  0  0  0
 0  1  0  0
 0  0  3  0
 0  0  0  1
```

```
C = [0.4 0 -0.3 0; 0 0 0 0; -0.3 0 0.5 -0.2; 0 0 -0.2 0.2]
```

```
C =
4×4 Matrix{Float64}:
  0.4  0.0  -0.3   0.0
  0.0  0.0   0.0   0.0
```

```
 -0.3  0.0   0.5  -0.2
  0.0  0.0  -0.2   0.2
```

```
K = [-7 2 4  0; 2 -4 2 0; 4 2 -9 3; 0 0 3 -3]
```

```
K =
4×4 Matrix{Int64}:
 -7   2   4   0
  2  -4   2   0
  4   2  -9   3
  0   0   3  -3
```

使用 polyeig 针对特征值、特征向量和条件数解算 QEP。

```
X,e=polyeig(K,C,M)
```

```
X =
4×8 Matrix{Float64}:
 -0.340934   -0.368074     -0.476676   1.0          0.861761     1.0
0.681127  -0.0724478
 -0.658486    1.0         -0.397976   1.09244   0.93808     0.655696   -
1.11447    1.0
  1.0          0.0490422    0.206033   1.11927   0.961669   -0.402116   -
0.599367  -0.409341
 -1.38943   -0.139356     1.0        1.16485   1.0        -2.05765     1.0
0.434022
e =
8×1 Matrix{Float64}:
  2.2279087320479034
  2.036350976643702
  1.4752411434756634
  0.3352944297785459
 -0.3465512996736314
 -1.6247783405292464
 -2.153616198037308
 -2.449849443705628
```

3.2.2.2.12 hessenberg

矩阵的 Hessenberg 形式

A. 语法

```
H = hessenberg(A)
P,H = hessenberg(A)
```

B. 说明

H = hessenberg(A) 查找 H，即矩阵 A 的 Hessenberg 形式。

P,H = hessenberg(A) 生成 Hessenberg 矩阵 H 和酉矩阵 P，这样 A = PHP' 并且 P'*P = eye(size(A))。

C. 示例

H 是 3×3 特征值测试矩阵：

```
H =
   [-149    -50   -154
    537    180    546
    -27     -9    -25]
```

其 Hessenberg 形状在 (3,1) 位置引入一个零：

```
P,H=hessenberg(H)
```

```
P    =3   ×   3    LinearAlgebra.HessenbergQ{Float64,    Matrix{Float64},
Vector{Float64}, false}:
1.0         0.0           0.0
0.0         -0.998738      0.0502159
0.0         0.0502159      0.998738
H =  3×3 UpperHessenberg{Float64, Matrix{Float64}}:
-149.0      42.2037        -156.317
-537.678    152.551        -554.927
·           0.0728473      2.44885
```

D. 详细信息

(2). Hessenberg 矩阵

Hessenberg 矩阵包含第一个下对角线下面的零值。如果矩阵对称或者是 Hermitian 矩阵，则形状为三对角。该矩阵具有和原点相同的特征值，但显示它们所需计算更少。

3.2.2.2.13 schur

Schur 分解

A. 语法

```
T = schur(A)
T,U,value = schur(A,...)
```

B. 说明

schur 函数计算 Schur 形式的矩阵。

E = ordeig(AA,BB) 接受一个拟三角矩阵对 AA 和 BB（通常由 qz 生成），并按广义特征值沿 AA- λ *BB 的对角线自上而下出现的顺序返回广义特征值。

计算矩阵 A 的 Schur 分解。（准）三角 Schur 因子可以从 Schur 对象 F 中获得，使用 F.Schur 或 F.T，正交/酉 Schur 向量可以是 使用 F.vectors 或 F.Z 获得，使得 A = F.vectors * F.Schur * F.vectors'。 A 的特征值可以通过 F.values 获得。

迭代分解产生分量 F.T、F.Z 和 F.values。

C. 示例

H 是 3×3 特征值测试矩阵

```
H = [ -149    -50   -154
       537    180    546
       -27     -9    -25 ]
```

其 Schur 形式为

```
schur(H)
```

```
Schur{Float64, Matrix{Float64}}
T factor:
3×3 Matrix{Float64}:
 1.0    -7.11189     815.871
 0.0    2.0          55.0236
 0.0    0.0          3.0
Z factor:
3×3 Matrix{Float64}:
 -0.316228         0.652929     0.688247
  0.948683         0.217643     0.229416
 -2.67496e-13      -0.725476    0.688247
eigenvalues:
3-element Vector{Float64}:
 0.9999999999918581
 2.0000000000042215
 3.000000000004076
```

3.2.2.2.14 schur!

Schur 分解

A. 语法

```
T = schur!(A)
T,U,value = schur!(A,...)
```

B. 说明

schur! 函数计算 Schur 形式的矩阵。

E = ordeig(AA,BB) 接受一个拟三角矩阵对 AA 和 BB（通常由 qz 生成），并按广义特征值沿 AA-λ*BB 的对角线自上而下出现的顺序返回广义特征值。

计算矩阵 A 的 Schur 分解。（准）三角 Schur 因子可以从 Schur 对象 F 中获得，使用 F.Schur 或 F.T，正交/酉 Schur 向量可以是 使用 F.vectors 或 F.Z 获得，使得 A = F.vectors * F.Schur * F.vectors'。 A 的特征值可以通过 F.values 获得。

迭代分解产生分量 F.T、F.Z 和 F.values。

与 schur! 相同，但使用输入参数 A 作为工作区。

C. 示例

H 是 3×3 特征值测试矩阵

```
H = [5. 7.; -2. -4.]
```

其 Schur 形式为

```
schur!(H)
```

```
Schur{Float64, Matrix{Float64}}
T factor:
2×2 Matrix{Float64}:
3.0   9.0
0.0  -2.0
Z factor:
2×2 Matrix{Float64}:
0.961524  0.274721
-0.274721  0.961524
eigenvalues:
2-element Vector{Float64}:
3.0
```

```
-2.0
```

3.2.2.2.15 rsf2csf

将实数 Schur 形式变换为复数 Schur 形式

A. 语法

```
Tnew,Unew = rsf2csf(T,U)
```

B. 说明

Tnew,Unew = rsf2csf(T,U) 将实矩阵 X 的 T,U =schur(X) 的输出从实数 Schur 形式变换为复数 Schur 形式。此操作变换 X 的特征值在 T 中的表示方式，并变换 U 以使 X = Unew*Tnew*Unew' 且 Unew'*Unew = I(size(X)[1],size(X)[2])。

在实数 Schur 形式中，T 在对角线上有实数特征值，复数特征值表示为沿主对角线的 2×2 实数分块：

$$\begin{bmatrix} \lambda_1 & t_{12} & t_{13} & t_{14} & t_{15} & \cdots \\ & a & e & t_{24} & t_{25} & \cdots \\ & f & a & t_{34} & t_{35} & \cdots \\ & & & c & g & \cdots \\ & & & h & c & \cdots \\ & & & & & \ddots \end{bmatrix}$$

这些分块表示的特征值是 $a \pm i\sqrt{-fe}$ 和 $c \pm i\sqrt{-hg}$.

在复数 Schur 形式中，Tnew 是上三角矩阵且所有特征值（实数或复数）都在主对角线上：

$$\begin{bmatrix} \lambda_1 & \text{tnew}_{12} & \text{tnew}_{13} & \text{tnew}_{14} & \text{tnew}_{15} & \cdots \\ & a+bi & \text{tnew}_{23} & \text{tnew}_{24} & \text{tnew}_{25} & \cdots \\ & & a-bi & \text{tnew}_{34} & \text{tnew}_{35} & \cdots \\ & & c+di & \text{tnew}_{45} & \cdots & \\ & & & c-di & \cdots & \\ & & & & & \ddots \end{bmatrix}$$

C. 示例

(1). 将实数 Schur 形式变换为复数 Schur 形式

对一个实矩阵应用 Schur 分解，然后变换矩阵因子，使特征值恰好在主对角线上。

创建一个实矩阵并计算 Schur 分解。U 因子是酉矩阵，因此，并且 T 因子是实数 Schur 形式，其复共轭特征值对表示为对角线上的 2×2 分块。

```
X = [1    1    1    3
1    2    1    1
1    1    3    1
-2    1    1    4];
T,U = schur(X)
```

```
T =
4×4 Matrix{Float64}:
4.81213  1.19723   -2.22729  -1.00674
0.0      1.92022   -3.04846  -1.83806
0.0      0.712929   1.92022   0.256561
0.0      0.0        0.0       1.34743
U =
4×4 Matrix{Float64}:
-0.491624  -0.489955  -0.633055  -0.342771
-0.497979   0.2403    -0.232494   0.800137
-0.675062   0.428792   0.423036  -0.425992
-0.233698  -0.719956   0.605168   0.246617
```

T 在对角线上有两个实数特征值，还有一个 2×2 分块表示一对复共轭特征值。

变换 U 和 T，使得 Tnew 是上三角矩阵，特征值在对角线上，并且 Unew 满足 X = UnewTnewUnew'。

```
Tnew,Unew = rsf2csf(T,U)
```

```
Tnew =
4×4 Matrix{ComplexF64}:
4.81213+0.0im   -0.969676+1.07781im     -0.521227+2.00513im          -
1.00674+0.0im
0.0+0.0im            1.92022+1.47422im          2.33553-1.11022e-16im
0.111697+1.65472im
0.0+0.0im            0.0+0.0im                       1.92022-1.47422im
0.800217+0.230971im
0.0+0.0im               0.0+0.0im                                0.0+0.0im
1.34743+0.0im
```

```
Unew
4×4 Matrix{ComplexF64}:
-0.491624+0.0im     -0.275607-0.441085im       0.213307+0.569911im     -
0.342771+0.0im
-0.497979+0.0im       -0.101219+0.216332im       -0.104617+0.209304im
0.800137+0.0im
-0.675062+0.0im       0.184173+0.386023im     -0.186679-0.380841im     -
0.425992+0.0im
-0.233698+0.0im         0.263466-0.648145im         0.313441-0.544806im
0.246617+0.0im
```

3.2.2.2.16 cdf2rdf

拟三角矩阵的特征值

A. 语法

```
Vnew,Dnew = cdf2rdf(V,D)
```

B. 说明

Vnew,Dnew = cdf2rdf(V,D) 将实矩阵 X 的 D,V = eigen(X),D = diagm(D)的输出从复数对角形转换为实数对角形。此操作将变换 X 的特征值在 D 中的表示方式，并变换 V 以使 X*Vnew = Vnew*Dnew。

在复数对角形中，D 是主对角线上具有复共轭特征值向量，沿对角线的一些特征值可能是实数，但假定复共轭特征值对是彼此相邻的。

在实数对角形中，Dnew 在对角线上有实数特征值，复数特征值表示为沿主对角线的 2×2 实数块：

$$\begin{bmatrix} \lambda_1 & & & & \\ & a & b & & \\ & -b & a & & \\ & & & c & d \\ & & & -d & c \\ & & & & & \ddots \end{bmatrix}$$

这些分块表示的特征值$a \pm i\sqrt{-fe}$是 和$c \pm i\sqrt{-hg}.$

在复数 Schur 形式中，Tnew 是上三角矩阵且所有特征值（实数或复数）都在主对角线上：

$$\begin{bmatrix} \lambda_1 & \text{tnew}_{12} & \text{tnew}_{13} & \text{tnew}_{14} & \text{tnew}_{15} & \cdots \\ & a+bi & \text{tnew}_{23} & \text{tnew}_{24} & \text{tnew}_{25} & \cdots \\ & & a-bi & \text{tnew}_{34} & \text{tnew}_{35} & \cdots \\ & & c+di & \text{tnew}_{45} & \cdots & \\ & & & c-di & \cdots & \\ & & & & & \ddots \end{bmatrix}$$

C. 示例

(1). 将复数对角矩阵变换为实数块对角矩阵

计算实矩阵的特征值和特征向量，然后将复共轭特征值对变换为实数块。

求实矩阵的特征值和特征向量。

```
X = [1   1   1   1   1
  0   4   5   1   1
  0  -5   4   1   1
  0   0   2   3   1
  0   0  -3  -2   1];
D,V = eigen(X)
D = diagm(D)
```

```
D =
5×5 Matrix{ComplexF64}:
1.0+0.0im       0.0+0.0im             0.0+0.0im             0.0+0.0im
0.0+0.0im
0.0+0.0im   2.11986-0.701806im        0.0+0.0im             0.0+0.0im
0.0+0.0im
0.0+0.0im        0.0+0.0im         2.11986+0.701806im        0.0+0.0im
0.0+0.0im
0.0+0.0im       0.0+0.0im             0.0+0.0im        3.88014-5.10458im
0.0+0.0im
0.0+0.0im        0.0+0.0im             0.0+0.0im             0.0+0.0im
3.88014+5.10458im
V =
5×5 Matrix{ComplexF64}:
1.0+0.0im       0.159282+0.403142im         0.159282-0.403142im       -
0.0178674+0.135128im  -0.0178674-0.135128im
0.0+0.0im          0.0704471+0.0267265im          0.0704471-0.0267265im
0.0130401+0.621446im   0.0130401-0.621446im
0.0+0.0im   -0.126137-0.10316im     -0.126137+0.10316im       0.636346-
0.0im        0.636346+0.0im
0.0+0.0im         -0.227861+0.416112im             -0.227861-0.416112im
```

```
0.10449+0.208668im    0.10449-0.208668im
0.0+0.0im    0.744853-0.0im          0.744853+0.0im          -0.115564-
0.349721im   -0.115564+0.349721im
```

T 在对角线上有两个实数特征值，还有一个 2×2 分块表示一对复共轭特征值。

变换 U 和 T，使得 Tnew 是上三角矩阵，特征值在对角线上，并且 Unew 满足 X = UnewTnewUnew'。

```
Vnew,Dnew = cdf2rdf(V,D)
```

```
Vnew
5×5 Matrix{ComplexF64}:
1.0-0.0im      0.159282-0.0im      0.403142-0.0im    -0.0178674-0.0im
0.135128-0.0im
0.0-0.0im    0.0704471-0.0im    0.0267265-0.0im      0.0130401-0.0im
0.621446-0.0im
0.0-0.0im    -0.126137-0.0im     -0.10316+0.0im       0.636346-0.0im
0.0-0.0im
0.0-0.0im    -0.227861-0.0im     0.416112-0.0im         0.10449-0.0im
0.208668-0.0im
0.0-0.0im    0.744853-0.0im          0.0-0.0im    -0.115564-0.0im   -
0.349721+0.0im
Dnew =
5×5 Matrix{ComplexF64}:
1.0-0.0im      0.0-0.0im       0.0-0.0im     0.0-0.0im      0.0-0.0im
0.0-0.0im    2.11986-0.0im  -0.701806+0.0im      0.0-0.0im       0.0-
0.0im
0.0-0.0im  0.701806-0.0im    2.11986-0.0im      0.0-0.0im       0.0-
0.0im
0.0-0.0im
```

3.2.2.3 矩阵分解

分解矩阵

函数名	简介
lu	LU 矩阵分解
ldlt	实对称三对角矩阵 S 的 LDLt 分解
cholesky	Cholesky 分解
cholupdate	Cholesky 分解的秩 1 更新

qrdelete	从 QR 分解中删除列或行
qrinsert	将列或行插入 QR 分解
qrupdate	QR 分解的秩 1 更新
planerot	Givens 平面旋转

3.2.2.3.1 lu

LU 矩阵分解

A. 语法

```
L,U = lu(A)
L,U,P = lu(A)
L,U,P,Q = lu(S)
L,U,P,Q,D = lu(S)
```

B. 说明

L,U = lu(A) 将满矩阵或稀疏矩阵 A 分解为一个上三角矩阵 U 和一个经过置换的下三角矩阵 L，使得 A = L*U。

L,U,P = lu(A) 还返回一个置换矩阵 P，并满足 A = P'LU。在此语法中，L 是单位下三角矩阵，U 是上三角矩阵。

L,U,P,Q = lu(S) 将稀疏矩阵 S 分解为一个单位下三角矩阵 L、一个上三角矩阵 U、一个行置换矩阵 P 以及一个列置换矩阵 Q，并满足 PSQ = L*U。

L,U,P,Q,D = lu(S) 还返回一个对角缩放矩阵 D，并满足 P*(D\S)Q = LU。行缩放通常会使分解更为稀疏和稳定。

C. 示例

(1). 矩阵的 LU 分解

计算矩阵的 LU 分解并检查生成的因子。通过 LU 分解，可以将一个矩阵 A 分解为一个上三角矩阵 U、一个下三角矩阵 L 和一个置换矩阵 P，并满足 PA=LU。这些矩阵描述了对矩阵执行高斯消去法，直至其化简成为行阶梯形矩阵所需的步骤。L 矩阵包含所有乘数，置换矩阵 P 解释行交换。

创建一个 3×3 矩阵并计算 LU 因子。

```
A = [10 -7 0
    -3  2 6
```

```
     5 -1 5]
```

```
L,U = lu(A)
```

```
L =
3×3 Matrix{Float64}:
1.0   0.0   0.0
0.5   1.0   0.0
-0.3  -0.04  1.0
U =
3×3 Matrix{Float64}:
10.0  -7.0  0.0
0.0   2.5  5.0
0.0   0.0  6.2
```

将因子相乘以重新创建 A。在双输入语法的基础上，lu 将置换矩阵 P 直接合并到 L 因子中，使得返回的 L 实际上等于 P'L，因此 A = LU。

```
L * U
```

```
ans =
3×3 Matrix{Float64}:
10.0  -7.0  0.0
5.0  -1.0  5.0
-3.0   2.0  6.0
```

您可以指定三个输出以将置换矩阵与 L 中的乘数分开。

```
L,U,P = lu(A)
m=diagm([1,1,1])
P=m[P,:]
```

```
L =
3×3 Matrix{Float64}:
1.0   0.0   0.0
0.5   1.0   0.0
-0.3  -0.04  1.0
U =
3×3 Matrix{Float64}:
10.0  -7.0  0.0
0.0   2.5  5.0
0.0   0.0  6.2
P =
3×3 Matrix{Int64}:
1  0  0
0  0  1
0  1  0
```

```
P'*L*U
```

```
ans =
3×3 Matrix{Float64}:
10.0  -7.0  0.0
-3.0   2.0  6.0
5.0  -1.0  5.0
```

(2). 用 LU 分解对线性方程组求解

通过执行 LU 分解并使用因子来简化问题，对线性方程组求解。

创建一个 5×5 幻方矩阵并求解线性方程组 Ax=b，其中 b 的所有元素等于 65，即幻数和。由于 65 是此矩阵的幻数和（各行之和与各列之和均为 65），因此 x 的预期解是由 1 组成的向量。

```
A = magic(5)
b = 65*ones(5,1)
x = A\b
```

```
x = 5×1 Matrix{Float64}:
1.0000000000000002
1.0
0.9999999999999997
1.0000000000000002
1.0
```

对于泛型方阵，反斜杠运算符使用 LU 分解计算线性方程组的解。LU 分解将 A 表示为三角矩阵的乘积，它可以通过代换公式轻松求解涉及三角矩阵的线性方程组。

要重新创建由反斜杠运算符计算的答案，请计算 A 的 LU 分解。然后，使用因子来求解两个三角线性方程组：

```
y = L(P*b);
x = U\y;
```

这种在求解线性方程组之前预先计算矩阵因子的方法可以在求解许多线性方程组时提高性能，因为分解仅发生一次而不需要重复。

```
L,U,P = lu(A)
m=diagm(ones(length(P)))
P=m[P,:]
```

```
L =
5×5 Matrix{Float64}:
1.0       0.0       0.0       0.0       0.0
0.73913   1.0       0.0       0.0       0.0
0.478261  0.768737  1.0       0.0       0.0
```

```
0.173913  0.252677  0.516365  1.0      0.0
0.434783  0.48394   0.723084  0.923077  1.0
U = 5×5 Matrix{Float64}:
23.0   5.0     7.0     14.0      16.0
0.0  20.3043  -4.17391  -2.34783   3.17391
0.0   0.0    24.8608   -2.89079  -1.09208
0.0   0.0     0.0     19.6512    18.9793
0.0   0.0     0.0      0.0     -22.2222
P = 5×5 Matrix{Float64}:
0.0  1.0  0.0  0.0  0.0
1.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  1.0
0.0  0.0  1.0  0.0  0.0
0.0  0.0  0.0  1.0  0.0
```

```
y = L\(P*b)
x = U\y
```

```
x = 5×1 Matrix{Float64}:
1.0000000000000002
1.0
0.9999999999999997
1.0000000000000002
1.0
```

(3). 稀疏矩阵的 LU 分解

计算稀疏矩阵的 LU 分解并验证恒等式 LU = PS*Q。

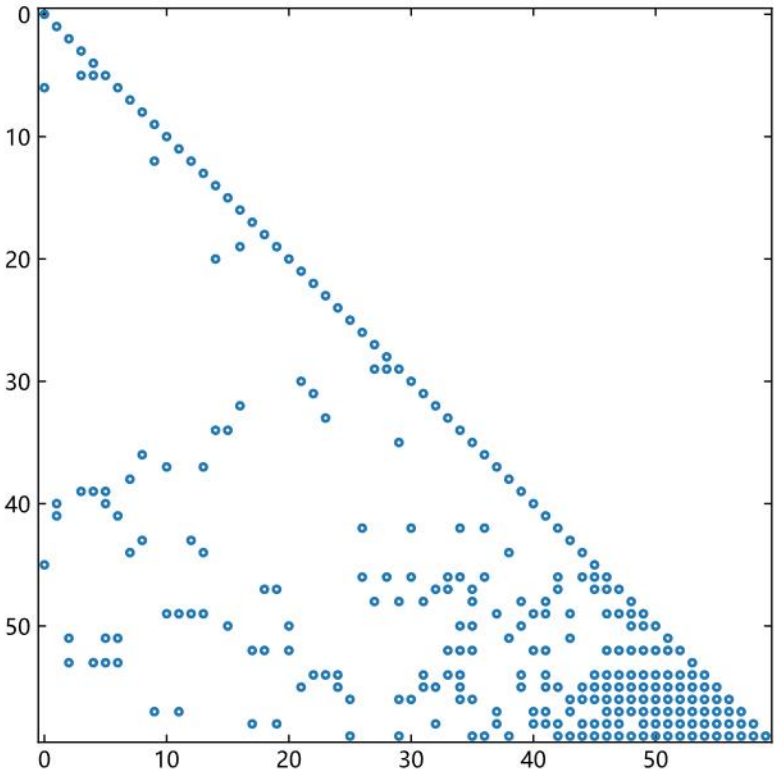
基于 Buckminster-Fuller 多面穹顶连接图创建一个 60×60 稀疏邻接矩阵。

```
S=bucky();
```

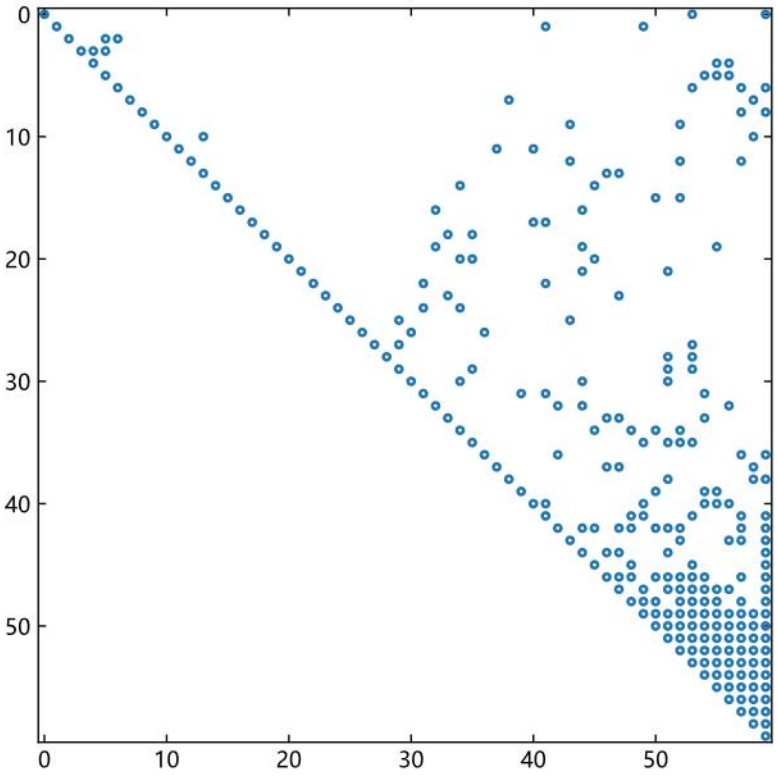
使用带有四个输出的稀疏矩阵语法计算 S 的 LU 分解，以返回行和列置换矩阵。

```
L=lu(S).L
U=lu(S).U
```

```
TyPlot.spy(L)
```




```
TyPlot.spy(U)
```



3.2.2.3.2 ldlt

实对称三对角矩阵 S 的 LDLt 分解

A. 语法

```
L,U = lu(A)
L,U,P = lu(A)
L,U,P,Q = lu(S)
L,U,P,Q,D = lu(S)
```

B. 说明

计算实对称三对角矩阵 S 的 LDLt 分解，使得 S = L* Diagonal(d)*L' 其中 L 是单位下三角矩阵,d 是一个向量。 LDLt 分解 F = ldlt(S) 的主要用途是解决线性方程组 Sx = b 与 F\b。

计算稀疏矩阵 A 的 LDL' 分解。 A 必须是 SparseMatrixCSC 或 Symmetric/Hermitian 视图稀疏矩阵 CSC。请注意，即使 A 没有类型标记，它也必须是对称的或 Hermitian。一种使用填充减少排列。 F = ldlt(A) 最常用于求解方程组 A* x = b，其中 F\b。返回的分解对象 F 还支持 diag、det、logdet 和 inv 方法。你可以提取使用 F.L. 来自 F 的单个因素但是，由于默认情况下启用了透视，因此分解是在内部进行的用置换矩阵 P 表示为 A == P'* L* D * L'* P；只使用 L 而不考虑 P 会给出不正确的答案。为了包括排列的影响，通常最好提取“组合”因子比如 PtL = F.PtL（相当于 P'* L）和 LtP = F.UP（相当于 L'* P）。支持的完整列表因子是:L、:PtL、:D、:UP、:U、:LD、:DU、:PtLD、:DUP。

当 check = true 时，如果分解失败则抛出错误。当 check = false 时，负责检查 分解的有效性（通过 issuccess）取决于用户。

设置可选的 shift 关键字参数会计算 A+shift*I 而不是 A 的因式分解。如果 perm 提供了参数，它应该是 1:size(A,1) 的排列，给出使用的顺序（而不是 CHOLMOD 的 默认 AMD 排序）。

C. 示例 利用 LDLT 分解解线性方程组

```
S = SymTridiagonal([3., 4., 5.], [1., 2.])
```

```
S = 3×3 SymTridiagonal{Float64, Vector{Float64}}:
3.0  1.0   ⋅
1.0  4.0  2.0
```

```
·   2.0  5.0
```

```
ldltS = ldlt(S)
```

```
ldltS = LDLt{Float64, SymTridiagonal{Float64, Vector{Float64}}}
L factor:
3×3        UnitLowerTriangular{Float64,        SymTridiagonal{Float64,
Vector{Float64}}}:
1.0        ·          ·
0.333333  1.0         ·
0.0       0.545455  1.0
D factor:
3×3 Diagonal{Float64, Vector{Float64}}:
3.0   ·        ·
·   3.66667    ·
·     ·       3.90909
```

```
b = [6., 7., 8.]
ldltS \ b
```

```
ans = 3-element Vector{Float64}:
1.7906976744186047
0.627906976744186
1.3488372093023255
```

```
S\b
```

```
ans =
3-element Vector{Float64}:
1.7906976744186047
0.627906976744186
1.3488372093023255
```

3.2.2.3.3 cholesky

Cholesky 分解

A. 语法

```
cholesky(A, Val(false); check = true)
cholesky(A, Val(true); tol = 0.0, check = true)
cholesky(S; shift = 0.0, check = true, perm = nothing)
```

B. 说明

cholesky(A, Val(false); check = true)

计算稠密对称正定矩阵 A 的 Cholesky 分解并返回 Cholesky 分解。 矩阵 A 可以是对称或 Hermitian StridedMatrix 或完全对称或 Hermitian StridedMatrix。

三角 Cholesky 因子可以通过 F.L 和 F.U 从因式分解 F 中获得，其中 A ≈ F.U' * F.U ≈ F.L * F.L'。

以下函数可用于 Cholesky 对象：size、\、inv、det、logdet 和 isposdef。

如果您的矩阵 A 由于其构造中的舍入误差而略微非 Hermitian，请在将其传递给 cholesky 之前将其包装在 Hermitian(A) 中，以便将其视为完美的 Hermitian。

当 check = true 时，如果分解失败则抛出错误。 当 check = false 时，检查分解有效性的责任在于用户。

cholesky(A, Val(true); tol = 0.0, check = true)

计算密集对称半正定矩阵 A 的主元 Cholesky 分解并返回 CholeskyPivoted 分解。 矩阵 A 可以是对称或 Hermitian StridedMatrix 或完全对称或 Hermitian StridedMatrix。

三角 Cholesky 因子可以通过 F.L 和 F.U 从因式分解 F 以及通过 F.p 的排列获得，其中 A[F.p, F.p] ≈ Ur' * Ur ≈ Lr * Lr' 其中 Ur = F.U[1:F.rank, :] 和 Lr = F.L[:, 1:F.rank]，或者 A ≈ Up' * Up ≈ Lp * Lp' with Up = F.U[1:F.rank, invperm(F.p)] and Lp = F.L[invperm（F.p），1：F.rank]。

以下函数可用于 CholeskyPivoted 对象：size、\、inv、det 和 rank。

参数 tol 确定用于确定等级的公差。 对于负值，公差是机器精度。

如果您的矩阵 A 由于其构造中的舍入误差而略微非 Hermitian，请在将其传递给 cholesky 之前将其包装在 Hermitian(A) 中，以便将其视为完美的 Hermitian。

当 check = true 时，如果分解失败则抛出错误。 当 check = false 时，检查分解有效性的责任在于用户。

cholesky(S; shift = 0.0, check = true, perm = nothing)

计算稀疏正定矩阵 A 的 Cholesky 分解。A 必须是 SparseMatrixCSC 或 SparseMatrixCSC 的对称/厄米特视图。 请注意，即使 A 没有类型标签，它仍然必须是对称的或 Hermitian 的。 如果未给出 perm，则使用减少填充的排列。 F = cholesky(A) 最常用于求解包含 F\b 的方程组，但也为 F 定义了方法 diag、det 和 logdet。您还可以使用 F.L 从 F 中提取单个因子。 但是，由于默认情况下启用旋转，因此分解在内部表示为 A == P' * L * L' * P 和置换矩阵 P； 只使用 L 而不考虑 P 会给出错误的答案。 要包括排列的影响，通常最好提取

“组合”因子，例如 PtL = F.PtL（相当于 P'* L）和 LtP = F.UP（相当于 L'*P）。

当 check = true 时，如果分解失败则抛出错误。 当 check = false 时，检查分解有效性的责任在于用户。

设置可选的 shift 关键字参数计算 A+shift*I 而不是 A 的因式分解。如果提供了 perm 参数，它应该是 1:size(A,1) 的排列，给出要使用的顺序。。

C. 示例

(1). 用对称正定矩阵求解线性系统

使用 cholesky 分解对称系数矩阵，然后使用 Cholesky 因子求解线性系统。

创建在对角线上为正值的对称矩阵。

```
A = [1 0 1; 0 2 0; 1 0 3]
```

```
A = 3×3 Matrix{Int64}:
1  0  1
0  2  0
1  0  3
```

计算矩阵的 Cholesky 因子。

```
R = cholesky(A).U
```

```
R = 3×3 UpperTriangular{Float64, Matrix{Float64}}:
1.0  0.0      1.0
 ⋅    1.41421  0.0
 ⋅     ⋅       1.41421
```

为方程 Ax = b 的右侧创建一个向量。

```
b = sum(A,dims=2)
```

由于 $A = RR^T$具有 Cholesky 分解，该线性方程变为 $RR^Tx = b$。使用反斜杠运算符求解 x。

```
x = R\(R'\b)
```

```
x = 3×1 Matrix{Float64}:
1.0
0.9999999999999999
0.9999999999999999U =
```

(2). 矩阵的 Cholesky 分解

计算矩阵的上下 Cholesky 分解，并验证结果。

使用 gallery 函数创建一个 6×6 对称正定测试矩阵。

```
A = gallery("lehmer",6);
```

使用 A 的上三角计算 Cholesky 因子。

```
A = gallery("lehmer",6);
```

使用 A 的上三角计算 Cholesky 因子。

```
R = cholesky(A).U
```

```
R = 6×6 UpperTriangular{Float64, Matrix{Float64}}:
 1.0  0.5       0.333333  0.25      0.2       0.166667
  ·    0.866025  0.57735   0.433013  0.34641   0.288675
  ·     ·        0.745356  0.559017  0.447214  0.372678
  ·     ·         ·        0.661438  0.52915   0.440959
  ·     ·         ·         ·        0.6       0.5
  ·     ·         ·         ·         ·        0.552771
```

验证上三角因子满足$R' * R - A = 0$（在舍入误差内）。

```
norm(R'*R - A)
```

```
ans = 4.1540741810552243e-16
```

使用 A 的下三角形计算 Cholesky 因子。

```
L = cholesky(A).L
```

```
L = 6×6 LowerTriangular{Float64, Matrix{Float64}}:
 1.0       ·         ·         ·         ·     ·
 0.5       0.866025  ·         ·         ·     ·
 0.333333  0.57735   0.745356  ·         ·     ·
 0.25      0.433013  0.559017  0.661438  ·     ·
 0.2       0.34641   0.447214  0.52915   0.6   ·
 0.166667  0.288675  0.372678  0.440959  0.5   0.552771
```

验证下三角因子满足 L * L' − A = 0（在舍入误差内）。

```
norm(L*L' - A)
```

```
ans = 4.1540741810552243e-16
```

3.2.2.3.4 cholupdate

Cholesky 分解的秩 1 更新

A. 语法

```
R1 = cholupdate(R,x)
R1 = cholupdate(R,x,'+')
R1 = cholupdate(R,x,'-')
```

B. 说明

R1 = cholupdate(R,x)（其中 R = chol(A) 是 A 的原始 Cholesky 分解）返回 A + x*x' 的上三角 Cholesky 因子，其中 x 是具有合适长度的一个列向量。cholupdate 仅使用 R 的对角线和上三角。R 的下三角将被忽略。

R1 = cholupdate(R,x,'+') 与 R1 = cholupdate(R,x) 相同。

R1 = cholupdate(R,x,'-') 返回 A - x*x' 的 Cholesky 因子。当 R 不是有效的 Cholesky 因子或旧矩阵不是正定矩阵，将会报告一条错误消息，这样将没有 Cholesky 分解。

C. 示例

```
A = pascal(4)
```

```
A = 4×4 Matrix{Int64}:
1  1   1   1
1  2   3   4
1  3   6  10
1  4  10  20
```

```
R = cholesky(A).U
x = [0,0,0,1]
```

```
R = 4×4 UpperTriangular{Float64, Matrix{Float64}}:
1.0  1.0   1.0  1.0
 ·   1.0   2.0  3.0
 ·    ·    1.0  3.0
 ·    ·     ·   1.0

x = 4-element Vector{Int64}:
0
0
0
1
```

这称为对 A 的秩为一的更新，因为 rank(x*x') 为 1：

```
A + x*x'
```

```
ans = 4×4 Matrix{Int64}:
1  1   1   1
1  2   3   4
1  3   6  10
1  4  10  21
```

不使用 R1 = chol(A + x*x') 计算 Cholesky 因子，可以使用 cholupdate：

```
R1 = cholupdate(R,x)
```

```
R1 = 4×4 UpperTriangular{Float64, Matrix{Float64}}:
1.0   1.0   1.0   1.0
 ·    1.0   2.0   3.0
 ·     ·    1.0   3.0
 ·     ·     ·    1.41421
```

接下来，通过从 A 的最后一个元素中减去 1 来破坏正定性（并且实际上将矩阵设置为奇异矩阵）。旧矩阵为：

```
A - x*x'
```

```
ans = 4×4 Matrix{Int64}:
1  1   1   1
1  2   3   4
1  3   6  10
1  4  10  19
```

将 cholesky 与 cholupdate 进行比较：

```
R1 = cholesky(A-x*x')
```

```
PosDefException: matrix is not positive definite; Cholesky factorization failed.
```

```
R1 = cholupdate(R,x,'-')
```

```
PosDefException: matrix is not positive definite; Cholesky factorization failed.
```

但是，将 0.5 从 A 的最后一个元素中减去将生成一个正定矩阵，可以使用 cholupdate 计算其 Cholesky 因子：

```
x = [0,0,0,1/sqrt(2)]
R1 = cholupdate(R,x,'-')
```

```
R1 = 4×4 UpperTriangular{Float64, Matrix{Float64}}:
1.0    1.0    1.0    1.0
 ·     1.0    2.0    3.0
 ·      ·     1.0    3.0
 ·      ·      ·     0.707107
```

3.2.2.3.5 qrdelete

从 QR 分解中删除列或行

A. 语法

```
Q1,R1 = qrdelete(Q,R,j)
Q1,R1 = qrdelete(Q,R,j,'col')
Q1,R1 = qrdelete(Q,R,j,'row')
```

B. 说明

Q1,R1 = qrdelete(Q,R,j) 返回矩阵 A1 的 QR 分解，其中 A1 是其列 A(:,j) 已删除的 A，Q,R = qr(A) 是 A 的 QR 分解。

Q1,R1 = qrdelete(Q,R,j,'col') 与 qrdelete(Q,R,j) 相同。

Q1,R1 = qrdelete(Q,R,j,'row') 返回矩阵 A1 的 QR 分解，其中 A1 是其行 A[j,:] 已删除的 A，Q,R = qr(A) 是 A 的 QR 分解。

C. 示例

```
A = magic(5)
Q,R = qr(A)
j = 3
Q1,R1 = qrdelete(Q,R,j,"row")
```

```
Q1 =
4×4 Matrix{Float64}:
0.527401  -0.51966   -0.669671   -0.0578013
0.713543   0.691057   0.0158381   0.114218
0.310236  -0.198212   0.467507   -0.803681
0.34126   -0.461633   0.576825    0.581128
R1 =
32.2335   26.0908   19.9482   21.4063    23.3297
0.0     -19.7045  -10.9891    0.431789  -1.48733
0.0       0.0      22.7444    5.83565   -3.19772
0.0       0.0       0.0     -14.5784     3.77958
```

返回有效的 QR 分解，虽然可能与以下项不同：

```
A2 = A
A2 = vcat(A2[1:j-1,:],A2[j+1:end,:])
Q2,R2 = qr(A2)
```

```
Q2 =
4×4 LinearAlgebra.QRCompactWYQ{Float64, Matrix{Float64}}:
 -0.527401   0.51966    0.669671   -0.0578013
 -0.713543  -0.691057  -0.0158381   0.114218
 -0.310236   0.198212  -0.467507   -0.803681
 -0.34126    0.461633  -0.576825    0.581128
R2 =
4×5 Matrix{Float64}:
 -32.2335  -26.0908  -19.9482  -21.4063    -23.3297
   0.0      19.7045   10.9891   -0.431789    1.48733
   0.0       0.0     -22.7444   -5.83565     3.19772
   0.0       0.0       0.0     -14.5784     3.77958
```

D. 算法

qrdelete 函数使用一系列的 Givens 旋转将分解的相应元素置零。

3.2.2.3.6 qrinsert

将列或行插入 QR 分解

A. 语法

```
Q1,R1 = qrinsert(Q,R,j,x)
Q1,R1 = qrinsert(Q,R,j,x,"col")
Q1,R1 = qrinsert(Q,R,j,x,"row")
```

B. 说明

Q1,R1 = qrinsert(Q,R,j,x) 返回矩阵 A1 的 QR 分解，其中 A1 是具有列 x 的 A = Q*R，该列插在 A(:,j) 之前。如果 A 具有 n 列和 j = n+1，则 x 插在 A 的最后一列之后。

Q1,R1 = qrinsert(Q,R,j,x,"col") 与 qrinsert(Q,R,j,x) 相同。

Q1,R1 = qrinsert(Q,R,j,x,"row") 返回矩阵 A1 的 QR 分解，其中 A1 是具有额外行 x 的 A = Q*R，该行插在 A[j,:] 之前。

C. 示例

```
A = magic(5)
Q,R = qr(A)
j = 3
x=[1:5...]'
Q1,R1 = qrinsert(Q,R,j,x,"row")
```

```
Q1 =
6×6 Matrix{Float64}:
0.523139    0.503914   -0.675038    0.120506   0.0411154   0.022543
0.707776   -0.696642    0.0189688  -0.078832   0.0833234  -0.0150287
0.0307729   0.0591583   0.0655614   0.116923   0.152712   -0.976864
0.123091    0.136278    0.354201    0.622243   0.639815    0.210401
0.307729    0.190162    0.409976    0.416119  -0.726417   -0.0150287
0.338502    0.45003     0.496112   -0.636594   0.176072    0.022543
R1 =
6×5 Matrix{Float64}:
32.4962  26.6801  21.4795  23.8182   26.0031
0.0     19.9292  12.4403   2.13397   4.32711
0.0      0.0     24.4514  11.8132    3.99311
0.0      0.0      0.0     20.2382   10.3392
0.0      0.0      0.0      0.0      16.1948
```

```
0.0      0.0      0.0      0.0       4.44089e-16
```

返回有效的 QR 分解，虽然可能与以下项不同：

```
A2 = [A[1:j-1,:]; x; A[j:end,:]]
Q2,R2 = qr(A2)
```

```
Q2 =
6×6 LinearAlgebra.QRCompactWYQ{Float64, Matrix{Float64}}:
-0.523139        0.503914         0.675038     -0.120506        0.0411154
0.022543
-0.707776        -0.696642        -0.0189688    0.078832        0.0833234
-0.0150287
-0.0307729       0.0591583        -0.0655614    -0.116923       0.152712
-0.976864
-0.123091        0.136278         -0.354201    -0.622243        0.639815
0.210401
-0.307729        0.190162         -0.409976    -0.416119        -0.726417
-0.0150287
-0.338502        0.45003          -0.496112     0.636594        0.176072
0.022543
R2 =
5×5 Matrix{Float64}:
-32.4962  -26.6801  -21.4795  -23.8182   -26.0031
0.0      19.9292   12.4403    2.13397    4.32711
0.0       0.0     -24.4514  -11.8132    -3.99311
0.0       0.0       0.0     -20.2382   -10.3392
0.0       0.0       0.0       0.0       16.1948
```

3.2.2.3.7 qrupdate

QR 分解的秩 1 更新

A. 语法

```
Q1,R1 = qrupdate(Q,R,u,v)
```

B. 说明

如果 Q,R = qr(A) 是 A 的原始 QR 分解，Q1,R1 = qrupdate(Q,R,u,v) 返回 A + u*v' 的 QR 分解，其中 u 和 v 是相应长度的列向量。

C. 示例

```
mu = sqrt(eps())
```

```
mu = 1.4901161193847656e-8
```

```
A = Matrix([ones(1,4); mu*I(4)])
```

是公认的最小平方示例，指明组成 A'*A 的危险。但我们使用 QR 分解 – 正交 Q 和上三角 R。

```
Q,R = qr(A)
```

```
R = 4×4 Matrix{Float64}:
-1.0  -1.0        -1.0        -1.0
0.0   2.10734e-8   1.05367e-8   1.05367e-8
0.0   0.0         1.82501e-8   6.08337e-9
0.0   0.0         0.0         1.72064e-8
```

在这种情况下，除了第一行，R 的上三角项都与 sqrt(eps) 类似。请考虑更新向量

```
u = [-1, 0, 0, 0, 0]
v = ones(4,1)
```

通过以下项从头更新到 A，而不是计算该秩亏无价值的 QR 分解：

```
QT,RT = qr(A + u*v')
```

```
QT = 5×5 LinearAlgebra.QRCompactWYQ{Float64, Matrix{Float64}}:
0.0   0.0   0.0   0.0  1.0
-1.0   0.0   0.0   0.0  0.0
0.0  -1.0   0.0   0.0  0.0
0.0   0.0  -1.0   0.0  0.0
0.0   0.0   0.0  -1.0  0.0
RT = 4×4 Matrix{Float64}:
-1.49012e-8   0.0         0.0         0.0
0.0        -1.49012e-8   0.0         0.0
0.0         0.0        -1.49012e-8   0.0
0.0         0.0         0.0        -1.49012e-8
```

我们可以使用 qrupdate。

```
Q1,R1 = qrupdate(Q,R,u,v)
```

```
Q1 = 5×5 LinearAlgebra.QRCompactWYQ{Float64, Matrix{Float64}}:
0.0  2.98023e-8  0.0  0.0  -1.0
-1.0  0.0        0.0  0.0   0.0
0.0  1.0        0.0  0.0   2.98023e-8
0.0  0.0        1.0  0.0   0.0
0.0  0.0        0.0  1.0   0.0
R1 = 4×4 Matrix{Float64}:
-1.49012e-8  -6.61744e-24  -9.92617e-24  -9.92617e-24
0.0         1.49012e-8   -3.30872e-24  -3.30872e-24
```

```
0.0          0.0          1.49012e-8    0.0
0.0          0.0          0.0          1.49012e-8
```

请注意这两个分解都正确，即使它们不同。

D. 提示

qrupdate 仅适用于满矩阵

E. 算法

qrupdate 使用由 Golub 与 van Loan 合著的 Matrix Computations 第三版第 12.5.1 节中的算法。qrupdate 很有用，因为如果我们采用 N = max(m,n)，则从头计算新的 QR 分解大体上属于 O(N3) 算法，而以此方式仅更新现有因子属于 O(N2) 算法。

3.2.2.3.8 planerot

Givens 平面旋转

A. 语法

```
G,y = planerot(x)
```

B. 说明

G,y = planerot(x)，其中 x 是一个包含 2 个分量的列向量，返回一个 2×2 正交矩阵 G，使得 y = G*x 有 y(2) = 0。

C. 示例

```
x = [3,4]
G , y =planerot(x)
```

```
G = 2×2 Matrix{Float64}:
0.6  0.8
-0.8  0.6
y = 2-element Vector{Float64}:
5.0
-4.440892098500626e-16
```

3.2.2.4 矩阵运算

矩阵的计算与转置。

函数名	简介
transpose	转置向量或矩阵
'	复共轭转置
adjoint	复共轭转置
*	矩阵乘法
^	矩阵幂
sqrt	矩阵平方根
exp	矩阵平方根
log	矩阵对数
funm	计算常规矩阵函数
kron	Kronecker 张量积
cross,×	叉积
dot,·	点积

3.2.2.4.1 transpose

转置向量或矩阵

A. 语法

```
B = transpose(A)
```

B. 说明

B = transpose(A) 返回 A 的非共轭转置，即每个元素的行和列索引都会互换。

C. 示例

(1). 实矩阵

创建由实数组成的矩阵并计算其转置。B 具有与 A 相同的元素，但是 B 的行是 A 的列，B 的列是 A 的行。

```
A = magic(4)
```

```
A = 4×4 Matrix{Int64}:
16   2   3  13
5  11  10   8
9   7   6  12
4  14  15   1
```

```
B = transpose(A)
```

```
B = 4×4 transpose(::Matrix{Int64}) with eltype Int64:
16   5   9   4
2  11   7  14
3  10   6  15
13   8  12   1
```

(2). 复矩阵

创建包含复数元素的矩阵并计算其非共轭转置。B 包含与 A 相同的元素，只是交换了行和列。虚部符号保持不变。

```
A = [1 3 4-1im 2+2im; 0+1im 1-1im 5 6-1im]
```

```
2×4 Matrix{Complex{Int64}}:
1+0im  3+0im  4-1im  2+2im
0+1im  1-1im  5+0im  6-1im
```

```
B = transpose(A)
```

```
  4×2 transpose(::Matrix{Complex{Int64}}) with eltype Complex{Int64}:
1+0im  0+1im
3+0im  1-1im
4-1im  5+0im
2+2im  6-1im
```

3.2.2.4.2 '

复共轭转置

A. 语法

```
B = A'
B = adjoint(A)
```

B. 说明

B = A' 计算 A 的复共轭转置。示例

B = adjoint(A) 是执行 A' 的替代方法，但很少使用。它可以启用类的运算符重载。

C. 示例

(1). 实矩阵的共轭转置

创建一个 4×2 的矩阵。

```
A = [2 1; 9 7; 2 8; 3 5]
```

```
A = 4×2 Matrix{Int64}:
2  1
9  7
2  8
3  5
```

计算 A 的共轭转置。

```
B = A'
```

```
B = 2×4 adjoint(::Matrix{Int64}) with eltype Int64:
 2  9  2  3
 1  7  8  5
```

结果是一个 2×4 的矩阵。B 的元素与 A 相同，但每个元素的行和列索引将会交换 。

(2). 复矩阵的共轭转置

复矩阵的共轭转置。

```
A = [0-1im 2+1im;4+2im 0-2im]
```

```
A = 2×2 Matrix{Complex{Int64}}:
0-1im  2+1im
4+2im  0-2im
```

计算 A 的共轭转置。

```
B = adjoint(A)
```

```
B = 2×2 adjoint(::Matrix{Complex{Int64}}) with eltype Complex{Int64}:
0+1im  4-2im
2-1im  0+2im
```

3.2.2.4.3 adjoint

复共轭转置

A. 语法

```
B = A'
B = adjoint(A)
```

B. 说明

B = A' 计算 A 的复共轭转置。

B = adjoint(A) 是执行 A' 的替代方法，但很少使用。它可以启用类的运算符重载。

C. 示例

(1). 实矩阵的共轭转置

创建一个 4×2 的矩阵。

```
A = [2 1; 9 7; 2 8; 3 5]
```

```
A = 4×2 Matrix{Int64}:
2  1
9  7
2  8
3  5
```

计算 A 的共轭转置。

```
B = A'
```

```
B = 4×4 transpose(::Matrix{Int64}) with eltype Int64:
2×4 adjoint(::Matrix{Int64}) with eltype Int64:
2  9  2  3
1  7  8  5
```

结果是一个 2×4 的矩阵。B 的元素与 A 相同，但每个元素的行和列索引将会交换。

(2). 复矩阵的共轭转置

创建一个包含复数元素的 2×2 的矩阵。

```
A = [0-1im 2+1im;4+2im 0-2im]
```

```
A = 2×2 Matrix{Complex{Int64}}:
0-1im  2+1im
4+2im  0-2im
```

计算 A 的共轭转置。

```
B = adjoint(A)
```

```
B = 2×2 adjoint(::Matrix{Complex{Int64}}) with eltype Complex{Int64}:
0+1im  4-2im
2-1im  0+2im
```

3.2.2.4.4 *

矩阵乘法

A. 语法

```
C = A*B
C = *(A,B)
```

B. 说明

C＝A*B 是 A 和 B 的矩阵乘积。如果 A 是 m×p 矩阵，B 是 p×n 矩阵，则 C 是通过以下公式定义的 m×n 矩阵：

$$C(i,j) = \sum_{k=1}^{p} A(i,k)B(k,j)$$

该定义说明 C[i,j] 是 A 第 i 行与 B 第 j 列的内积。

```
C[i,j] = A[i,:]*B[:,j]
```

对于非标量 A 和 B，A 的列数必须等于 B 的行数。对于非标量输入，矩阵乘法不一定能互换位置。也就是说，A * B 不一定等于 B * A。至少一方输入为标量时，A * B 等于 A.*B，这时可以互换位置。示例

C. 示例

(1). 将两个向量相乘

创建一个 1×4 行向量 A 和一个 4×1 列向量 B。

```
A = [1 1 0 0]
B = [1; 2; 3; 4];
```

A 乘以 B。

```
C = A*B
```

```
C = 1-element Vector{Int64}:
3
```

结果是 1×1 标量，也称为向量 A 和 B 的点积或内积。或者，可以使用 dot(A,B) 语法来计算 A⋅ B 点积。

B 乘以 A。

```
C = B*A
```

```
C = 4×4 Matrix{Int64}:
1  1  0  0
2  2  0  0
```

```
3  3  0  0
4  4  0  0
```

结果是 4×4 矩阵，也称为向量 A 和 B 的外积。两个向量的外积 A⊗ B 返回一个矩阵。

(2). 将两个数组相乘

创建两个数组 A 和 B。

```
C = A*B
```

```
C = 2×3 Matrix{Int64}:
24  35  114
30  52  162
```

计算 A 第二行和 B 第三列的内积。

```
A[2,:]'*B[:,3]
```

```
ans = 162
```

3.2.2.4.5 ^

矩阵幂

A. 语法

```
C = A ^ B
C = ^(A,B)
```

B. 说明

C＝A^B 计算 A 的 B 次幂并以 C 形式返回结果。示例。

C. 示例

(1). 矩阵的平方

创建一个 2×2 的矩阵。并对其求平方。

```
A = [1 2; 3 4]
C = A^2
```

```
C = 2×2 Matrix{Int64}:
7  10
15  22
```

(2). 矩阵指数

创建一个 2×2 矩阵并将其用作标量的指数。

```
B = [0 1; 1 0]
C = 2^B
```

```
C = 2×2 Matrix{Float64}:
1.25  0.75
0.75  1.25
```

通过首先找到矩阵 B 的特征值 D 和特征向量 V 来计算 C。

```
D,V=eigen(B)
```

```
D = 2-element Vector{Float64}:
-1.0
1.0
V = 2×2 Matrix{Float64}:
-0.707107  0.707107
0.707107  0.707107
```

接下来，使用公式 2^B = V*2^D/V 来计算幂。

```
D = diagm(D)
c = V*2^D/V
```

```
c = 2×2 Matrix{Float64}:
1.25  0.75
0.75  1.25
```

3.2.2.4.6 sqrt

矩阵平方根

A. 语法

```
X = sqrt(A)
```

B. 说明

X = sqrt(A) 返回矩阵 A 的主平方根，即 X*X = A。

X 是每个特征值都具有非负实部的唯一平方根。 如果 A 具有任何具有负实部的特征值，则会产生复数结果。 如果 A 是奇异的，那么 A 可能没有平方根。 如果检测到精确的奇异点，则会打印警告。

C. 示例

(1). 微分算子的矩阵平方根

创建四阶微分算子的矩阵表示形式 A。此矩阵是对称正定矩阵。

```
A = [5 -4 1 0 0; -4 6 -4 1 0; 1 -4 6 -4 1; 0 1 -4 6 -4; 0 0 1 -4 6]
```

```
A = 5×5 Matrix{Int64}:
5  -4   1   0   0
-4   6  -4   1   0
1  -4   6  -4   1
0   1  -4   6  -4
0   0   1  -4   6
```

使用 sqrt 计算 A 的唯一正定平方根。X 是二阶微分算子的矩阵表示形式。

```
X = round.(sqrt(A))
```

```
X = 5×5 Matrix{Float64}:
2.0  -1.0   0.0   0.0   0.0
-1.0   2.0  -1.0   0.0   0.0
0.0  -1.0   2.0  -1.0   0.0
0.0   0.0  -1.0   2.0  -1.0
0.0   0.0   0.0  -1.0   2.0
```

(2). 具有多个平方根的矩阵

以具有四个平方根的矩阵 A 为例。

$$A = \begin{bmatrix} 7 & 10 \\ 15 & 22 \end{bmatrix}$$

A 的两个平方根由 Y1 和 Y2 给出：

$$Y_1 = \begin{bmatrix} 1.5667 & 1.7408 \\ 2.6112 & 4.1779 \end{bmatrix}$$

$$Y_2 = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

确认 Y1 和 Y2 是矩阵 A 的平方根。

```
A = [7 10; 15 22]
Y1 = [1.5667 1.7408; 2.6112 4.1779]
A - Y1*Y1
```

```
ans = 2×2 Matrix{Float64}:
-0.00012585  -0.00019968
-0.00029952  -0.00042537
```

```
Y2 = [1 2; 3 4]
A - Y2*Y2
```

```
ans = 2×2 Matrix{Int64}:
0  0
0  0
```

A 的另外两个平方根是 -Y1 和 -Y2。可从 A 的特征值和特征向量获得所有这四个根。如果 D,V = eigen(A)，则平方根具有通式 Y = VS/V，其中

D = SS 和 S 具有四个符号选项以生成四个不同的 Y 值：

$$S = \begin{bmatrix} \pm 0.3723 & 0 \\ 0 & \pm 5.3723 \end{bmatrix}$$

使用 sqrt 计算 A 的平方根。sqrt 函数会选择正平方根并生成 Y1，即使 Y2 看似更自然的结果也是如此。

```
Y = sqrt(A)
```

```
2×2 Matrix{Float64}:
1.5667    1.74078
2.61116   4.17786
```

3.2.2.4.7 exp

矩阵指数

A. 语法

```
X = exp(A)
```

B. 说明

Y = exp(X) 计算 X 的矩阵指数。虽然不按此种方式计算，但是如果 X 包含一组完整的特征向量 V 和对应特征值 D，则 D,V = eigen(X) 且

exp(X) = V*diag(exp(diagm(D)))/V

对于逐个元素的指数运算，请使用 exp.

C. 示例

(1). 比较两种指数运算结果

计算和比较 A 的指数和 A 的矩阵指数。

```
A = [1 1 0; 0 0 2; 0 0 -1]
exp.(A)
```

```
ans = 3×3 Matrix{Float64}:
2.71828  2.71828  1.0
1.0      1.0      7.38906
1.0      1.0      0.367879
```

```
exp(A)
```

```
ans = 3×3 Matrix{Float64}:
2.71828  1.71828  1.08616
0.0      1.0      1.26424
```

```
0.0      0.0      0.367879
```

请注意，两种结果的对角线元素相等，这对于任何三角矩阵都是如此。非对角线元素（包括对角线下方的元素）则不相同。

3.2.2.4.8 log

矩阵对数

A. 语法

```
L = log(A)
```

B. 说明

L = log(A) 是 A 的主矩阵对数，即 exp(A) 的倒数。输出 L 是每个特征值都具有严格位于 π 和 π 之间的虚部的唯一对数。如果 A 是奇异矩阵或在负实轴上具有特征值，则未定义主对数。在这种情况下，log 计算非主对数并返回警告消息。

对于逐个元素的对数运算，请使用 log。

C. 示例

(1). 比较矩阵对数

计算矩阵 A 的矩阵指数。

```
A = [1 1 0; 0 0 2; 0 0 -1]
Y = exp(A)
```

```
Y = 3×3 Matrix{Float64}:
2.71828  1.71828  1.08616
0.0      1.0      1.26424
0.0      0.0      0.367879
```

计算 Y 的矩阵对数以重新生成原始矩阵 A。

```
P = log(Y)
```

```
P = 3×3 Matrix{Float64}:
1.0   1.0          -7.45931e-17
0.0  -1.11022e-16   2.0
0.0   0.0          -1.0
```

log(A) 涉及取零的对数，因此生成的结果较差。

```
Q = log.(Complex.(A))
```

```
Q = 3×3 Matrix{ComplexF64}:
0.0+0.0im       0.0+0.0im          -Inf+0.0im
```

```
-Inf+0.0im      -Inf+0.0im       0.693147+0.0im
-Inf+0.0im      -Inf+0.0im       0.0+3.14159im
```

3.2.2.4.9 funm

计算常规矩阵函数

A. 语法

```
F = funm(A, fun)
```

B. 说明

F = funm(A, fun) 计算在方阵参数为 A 时用户定义的函数 fun 。 F = fun(x) 必须接受矩阵 x 。示例

您也可以使用 fnum 计算下表中列出的特殊函数在 A 处的值。

函数	计算矩阵 A 处的函数的语法
exp	funm(A, exp)
log	funm(A, log)
sin	funm(A, sin)
cos	funm(A, cos)
sinh	funm(A, sinh)
cosh	funm(A, cosh)

对于方阵根，请改用 sqrt(A) 。对于矩阵指数， exp(A) 和 fun(A, exp) 哪一个更准确取决于矩阵 A 。

C. 示例

(1). 示例 1

以下命令计算 3x3 幻方矩阵的矩阵正弦值。

```
F = funm(magic(3), sin)
```

```
F = 3×3 Matrix{Float64}:
 -0.384981   1.01909    0.0161816
  0.617925   0.216763  -0.184399
  0.417344  -0.585562   0.818506
```

(2). 示例 2

以下语句

S = funm(X, sin)

C = funm(X, cos)

在舍入误差内生成与下面相同的结果

E = exp(im*X)

C = real(E)

S = imag(E)

在任一情况下，结果在舍入误差内满足 S*S + C*C = I，其中 I = eye(size(X)) 。

(3)．示例 3

要使用一个对 funm 的调用计算函数 exp(x) + cos(x) 在 A 处的值，请使用

F = funm(A, fun_expcos)

其中 fun_expcos 是以下函数。

fun_expcos(A) = exp(A) + cos(A)

3.2.2.4.10 kron

Kronecker 张量积

A. 语法

```
K = kron(A,B)
```

B. 说明

K = kron(A,B) 返回矩阵 A 和 B 的 Kronecker 张量积。如果 A 是 m×n 矩阵，而 B 是 p×q 矩阵，则 kron(A,B)是通过获取 A 元素与矩阵 B 元素之间的所有可能积而形成的一个 mp×nq 矩阵。

C. 示例

(1). 分块对角矩阵

创建分块对角矩阵。

创建一个 4×4 单位矩阵以及一个要沿对角线重复的 2×2 矩阵。

```
A = Int.(I(4))
B = [1 -1;-1 1];
```

使用 kron 计算 Kronecker 张量积。

```
K = kron(A,B)
```

```
K = 8×8 Matrix{Int64}:
1  -1   0   0   0   0   0   0
-1   1   0   0   0   0   0   0
0   0   1  -1   0   0   0   0
0   0  -1   1   0   0   0   0
0   0   0   0   1  -1   0   0
0   0   0   0  -1   1   0   0
0   0   0   0   0   0   1  -1
0   0   0   0   0   0  -1   1
```

结果为一个 8×8 分块对角矩阵。

(2). 重复矩阵元素

通过重复元素来扩展矩阵大小。

创建一个 2×2 的全 1 矩阵，以及一个需要重复其元素的 2×3 矩阵。

```
A = [1 2 3; 4 5 6]
B = ones(2,2)
```

使用 kron 计算 Kronecker 张量积。

```
K = kron(A,B)
```

```
K = 4×6 Matrix{Float64}:
1.0  1.0  2.0  2.0  3.0  3.0
1.0  1.0  2.0  2.0  3.0  3.0
4.0  4.0  5.0  5.0  6.0  6.0
4.0  4.0  5.0  5.0  6.0  6.0
```

结果为一个 4×6 分块矩阵。

(3). 稀疏拉普拉斯算子矩阵

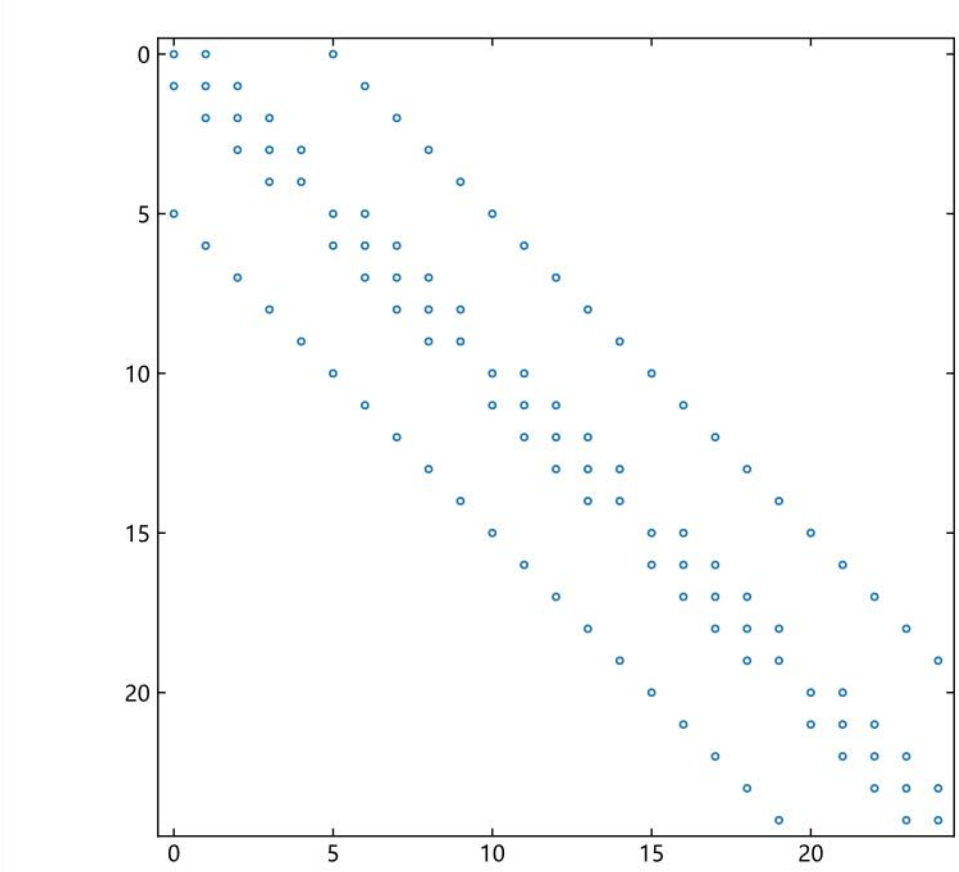
此示例以可视方式呈现稀疏拉普拉斯算子矩阵。

离散拉普拉斯算子在二维 n×n 网格中的矩阵表示形式是 nn×nn 稀疏矩阵。每行或每列中最多有 5 个非零元素。可以采用一维差分算子的 Kronecker 积的形式生成矩阵。在此示例中，n = 5。

```
n = 5
I1 = speye(n,n)
E = sparse(2:n,1:n-1,1,n,n)
D = E+E'-2*I1
A = kron(D,I1)+kron(I1,D)
```

使用 spy 以可视方式呈现稀疏模式。

```
TyPlot.spy(A)
```



3.2.2.4.11 cross,x

叉积

A. 语法

```
C = cross(A,B)
C = ×(A,B)
```

B. 说明

C = cross(A,B) 返回 A 和 B 的叉积。

如果 A 和 B 为向量，则它们的长度必须为 3。

如果 A 和 B 为矩阵或多维数组，则它们必须具有相同大小。在这种情况下，cross 函数将 A 和 B 视为三元素向量集合。该函数计算对应向量沿大小等于 3 的第一个数组维度的叉积。

C. 示例

(1). 向量的叉积

创建两个三维向量。

```
A = [4 ,-2, 1]
B = [1, -1, 3]
```

求出 A 和 B 的叉积。结果 C 是同时与 A 和 B 垂直的一个向量。

```
C = cross(A,B)
```

```
C = 3-element Vector{Int64}:
-5
-11
-2
```

使用点积验证 C 是否与 A 和 B 垂直。

```
dot(C,A)==0 & dot(C,B)==0
```

```
ans = true
```

3.2.2.4.12 dot,•

点积

A. 语法

```
C = dot(A,B)
C = ·(A,B)
```

B. 说明

C = dot(A,B) 返回 A 和 B 的标量点积。

如果 A 和 B 是向量，则它们的长度必须相同。

如果 A 和 B 为矩阵或多维数组，则它们必须具有相同大小。在本例中，dot 函数将 A 和 B 视为向量集合。该函数计算对应向量沿大小不等于 1 的第一个数组维度的叉积。

C. 示例

(1). 实数向量的点积

创建两个简单的三元素向量。

```
A = [4, -1, 2]
B = [2, -2, -1]
```

计算 A 和 B 的点积。

```
C = dot(A,B)
```

```
C = 8
```

结果为 8，因为

```
C = A[1]*B[1] + A[2]*B[2] + A[3]*B[3]
```

(2). 复数向量的点积

创建两个复数向量。

```
A = [1+im ,1-im ,-1+im ,-1-im]
B = [3-4im ,6-2im ,1+2im ,4+3im]
```

计算 A 和 B 的点积。

```
C = dot(A,B)
```

```
C = 1 - 5im
```

结果为一个复数标量，因为 A 和 B 是复数。通常，两个复数向量的点积也是复数。获取一个复数向量与其自身的点积除外。

计算 A 与自身的内积。

```
D = dot(A,A)
```

```
D = 8 + 0im
```

结果为一个实数标量。向量与自身的内积与向量 norm(A) 的欧几里德长度相关。

(3). 矩阵的点积

创建两个矩阵。

```
A = [1 2 3;4 5 6;7 8 9]
B = [9 8 7;6 5 4;3 2 1]
```

求 A 和 B 的点积。

```
C = dot(A,B)
```

```
C = 165
```

结果 C 包含三个不同的点积。dot 将 A 和 B 的各列视为向量，并计算对应列的点积。因此，例如 C(1) = 54 是 A(:,1) 与 B(:,1) 的点积。

(4). 多维数组的点积

创建两个多维数组。

```
A = cat([1 1;1 1],[2 3;4 5],[6 7;8 9],dims=3)
```

```
A = 2×2×3 Array{Int64, 3}:
[:, :, 1] =
1  1
1  1
```

```
[:, :, 2] =
2  3
4  5
[:, :, 3] =
6  7
8  9
```

```
B = cat([2 2;2 2],[10 11;12 13],[14 15; 16 17],dims=3)
```

```
B = 2×2×3 Array{Int64, 3}:
[:, :, 1] =
2  2
2  2
[:, :, 2] =
10  11
12  13
[:, :, 3] =
14  15
16  17
```

```
dot(A,B)
```

```
ans = 644
```

3.2.2.5 矩阵结构

矩阵的带宽与结构。

函数名	简介
bandwidth	矩阵的上下带宽
tril	矩阵的下三角形部分
triu	矩阵的上三角形部分
isbanded	确定矩阵是否在特定带宽范围内
isdiag	确定矩阵是否为对角矩阵
ishermitian	确定矩阵是 Hermitian 矩阵
issymmetric	确定矩阵是对称矩阵
istril	确定矩阵是否为下三角矩阵
istriu	确定矩阵是否为上三角矩阵

3.2.2.5.1 bandwidth

矩阵的上下带宽

A. 语法

```
B = bandwidth(A,type)
lower,upper = bandwidth(A)
```

B. 说明

B = bandwidth(A,type) 返回 type 指定的矩阵 A 的带宽。将 type 指定为下带宽 "lower" 或指定为上带宽 "upper"。

lower,upper = bandwidth(A) 返回矩阵 A 的下带宽 lower 和上带宽 upper。

C. 示例

(1). 计算三角矩阵的带宽

创建一个 6×6 的下三角矩阵。

```
A = tril(magic(6))
```

```
A = 6×6 Matrix{Int64}:
35   0   0   0   0   0
3   32   0   0   0   0
31   9   2   0   0   0
8   28  33  17   0   0
30   5  34  12  14   0
4   36  29  13  18  11
```

通过将 type 指定为 "lower" 来计算 A 的下带宽。结果为 5，原因是主对角线下方的每个对角线都有非零元素。

```
B = bandwidth(A,"lower")
```

```
B = 5
```

通过将 type 指定为 "upper" 来计算 A 的上带宽。结果为 0，原因是主对角线上方没有非零元素。

```
B = bandwidth(A,"upper")
```

```
B = 0
```

(2). 计算稀疏分块矩阵的带宽

创建一个 100×100 的稀疏分块矩阵。

```
B = kron(speye(25),ones(4,4))
```

查看 B 左上角一个 10×10 部分的元素。

```
Matrix(B[1:10,1:10])
```

```
ans = 10×10 Matrix{Float64}:
1.0  1.0  1.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0
1.0  1.0  1.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0
1.0  1.0  1.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0
1.0  1.0  1.0  1.0  0.0  0.0  0.0  0.0  0.0  0.0
0.0  0.0  0.0  0.0  1.0  1.0  1.0  1.0  0.0  0.0
0.0  0.0  0.0  0.0  1.0  1.0  1.0  1.0  0.0  0.0
0.0  0.0  0.0  0.0  1.0  1.0  1.0  1.0  0.0  0.0
0.0  0.0  0.0  0.0  1.0  1.0  1.0  1.0  0.0  0.0
0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  1.0
0.0  0.0  0.0  0.0  0.0  0.0  0.0  0.0  1.0  1.0
```

B 包含 4×4 个由 1 组成的数据块，每个块都以主对角线为中心。

通过指定两个输出参数来计算 B 的下带宽和上带宽。

```
lower,upper = bandwidth(B)
```

```
lower = 3
upper = 3
```

3.2.2.5.2 tril

矩阵的下三角形部分

A. 语法

```
L = tril(A)
L = tril(A,k)
```

B. 说明

L = tril(A) 返回矩阵 A 的下三角部分。

L = tril(A,k) 返回 A 的第 k 个对角线上以及该对角线下方的元素。

C. 示例

(1). 提取矩阵的下三角部分

创建一个 4×4 的全 1 矩阵。提取下三角部分。

```
A = ones(4,4)
```

```
A = 4×4 Matrix{Float64}:
1.0  1.0  1.0  1.0
```

```
1.0  1.0  1.0  1.0
1.0  1.0  1.0  1.0
1.0  1.0  1.0  1.0
```

```
B = tril(A)
```

```
B = 4×4 Matrix{Float64}:
1.0  0.0  0.0  0.0
1.0  1.0  0.0  0.0
1.0  1.0  1.0  0.0
1.0  1.0  1.0  1.0
```

仅提取主对角线下方的元素。

```
C = tril(A,-1)
```

```
C = 4×4 Matrix{Float64}:
0.0  0.0  0.0  0.0
1.0  0.0  0.0  0.0
1.0  1.0  0.0  0.0
1.0  1.0  1.0  0.0
```

3.2.2.5.3 triu

矩阵的上三角形部分

A. 语法

```
L = triu(A)
L = triu(A,k)
```

B. 说明

L = triu(A) 返回矩阵 A 的上三角部分。

L = triu(A,k) 返回 A 的第 k 个对角线上以及该对角线下方的元素。

C. 示例

(1). 提取矩阵的上三角部分

创建一个 4×4 的全 1 矩阵。提取上三角部分。

```
A = ones(4,4)
```

```
A = 4×4 Matrix{Float64}:
1.0  1.0  1.0  1.0
1.0  1.0  1.0  1.0
1.0  1.0  1.0  1.0
```

```
1.0  1.0  1.0  1.0
```

```
B = triu(A)
```

```
B = 4×4 Matrix{Float64}:
1.0  1.0  1.0  1.0
0.0  1.0  1.0  1.0
0.0  0.0  1.0  1.0
0.0  0.0  0.0  1.0
```

仅提取主对角线上方的元素。

```
C = triu(A,1)
```

```
C = 4×4 Matrix{Float64}:
0.0  1.0  1.0  1.0
0.0  0.0  1.0  1.0
0.0  0.0  0.0  1.0
0.0  0.0  0.0  0.0
```

3.2.2.5.4 isbanded

确定矩阵是否在特定带宽范围内

A. 语法

```
tf = isbanded(A,lower,upper)
```

B. 说明示

如果矩阵 A 在指定的下带宽 lower 和上带宽 upper 范围内，则 tf = isbanded(A,lower,upper) 返回逻辑值 true；否则返回逻辑值 false。

C. 示例

(1). 测试方阵

创建一个主对角线上方和下方有非零对角线的 5×5 方阵。

```
A = [2 3 0 0 0 ; 1 -2 -3 0 0; 0 -1 2 3 0 ; 0 0 1 -2 -3; 0 0 0 -1 2]
```

```
A = 5×5 Matrix{Int64}:
2   3   0   0   0
1  -2  -3   0   0
0  -1   2   3   0
0   0   1  -2  -3
0   0   0  -1   2
```

将两个带宽 lower 和 upper 均指定为 1 以测试 A 是否为三对角矩阵。

```
isbanded(A,1,1)
```

```
ans = true
```

通过将 lower 指定为 0 来测试 A 的主对角线下方是否有非零元素。

```
isbanded(A,0,1)
```

```
ans = false
```

结果为逻辑值 0 (false)，这是因为 A 的主对角线下方有非零元素。

(2). 测试非方阵

创建一个 3×5 矩阵。

```
A = [1 0 0 0 0; 2 1 0 0 0; 3 2 1 0 0]
```

```
A = 3×5 Matrix{Int64}:
1  0  0  0  0
2  1  0  0  0
3  2  1  0  0
```

测试 A 的主对角线上方是否有非零元素。

```
isbanded(A,2,0)
```

```
ans = true
```

测试稀疏分块矩阵创建一个 100×100 的稀疏分块矩阵。

```
B = kron(speye(25),ones(4,4))
```

测试 B 的下带宽和上带宽是否为 1。

```
isbanded(B,1,1)
```

```
ans = false
```

测试 B 的下带宽和上带宽是否为 3。

```
isbanded(B,3,3)
```

```
ans = true
```

矩阵 B 的上带宽和下带宽为 3，因为非零对角分块为 4×4。

3.2.2.5.5 isdiag

确定矩阵是否为对角矩阵

A. 语法

```
tf = isdiag(A)
```

B. 说明

如果 A 是一个对角矩阵，则 tf = isdiag(A) 返回逻辑值 1 (true)；否则返回逻辑值 0 (false)。

C. 示例

(1). 测试对角矩阵

创建一个 4×4 单位矩阵。

```
I1 = I(4)
```

```
I1 = 4×4 Diagonal{Bool, Vector{Bool}}:
1   ·   ·   ·
·   1   ·   ·
·   ·   1   ·
·   ·   ·   1
```

通过测试查看该矩阵是否为对角矩阵。

```
isdiag(I1)
```

```
ans = true
```

(2). 测试带状矩阵

创建一个主对角线和第一条对角线上具有非零元素的矩阵。

```
A = 3*I(4) + diagm(1=>[2 ,2 ,2])
```

```
A = 4×4 Matrix{Int64}:
3  2  0  0
0  3  2  0
0  0  3  2
0  0  0  3
```

通过测试查看该矩阵是否为对角矩阵。

```
isdiag(A)
```

```
ans = false
```

该矩阵不是对角矩阵，因为主对角线上方有非零元素。

根据 A 的主对角线元素创建一个新矩阵 B。

```
B = diagm(diag(A))
```

测试以查看 B 是否为对角矩阵。

```
isdiag(B)
```

```
ans = true
```

3.2.2.5.6 ishermitian

确定矩阵是 Hermitian 矩阵

A. 语法

```
tf = ishermitian(A)
```

B. 说明

如果方阵 A 是 Hermitian 矩阵，则 tf = ishermitian(A) 返回 true；否则返回 false。

C. 示例

(1). 测试对称矩阵是否为 Hermitian 矩阵

创建一个 3×3 矩阵。

```
A = [1 0 1im; 0 1 0; 1im 0 1]
```

```
A = 3×3 Matrix{Complex{Int64}}:
1+0im  0+0im  0+1im
0+0im  1+0im  0+0im
0+1im  0+0im  1+0im
```

该矩阵相对于其实数值对角线是对称的。

测试该矩阵是否为 Hermitian 矩阵。

```
tf = ishermitian(A)
```

```
tf = false
```

将 A(3,1) 中的元素更改为 -1im。

```
A[3,1] = -1im
```

确定修改后的矩阵是否为 Hermitian 矩阵。

```
tf = ishermitian(A)
```

```
tf = true
```

3.2.2.5.7 issymmetric

确定矩阵是对称矩阵

A. 语法

```
tf = issymmetric(A)
```

B. 说明

如果方阵 A 是 对称矩阵，则 tf = issymmetric(A) 返回 true；否则返回

false。

C. 示例

(1). 测试 Hermitian 矩阵是否为对称矩阵

创建一个 3×3 矩阵。

```
A = [1 0 1im; 0 1 0;-1im 0 1]
```

```
3×3 Matrix{Complex{Int64}}:
1+0im  0+0im  0+1im
0+0im  1+0im  0+0im
0-1im  0+0im  1+0im
```

该矩阵是 Hermitian 矩阵，具有实数值对角线。

测试该矩阵是否为对称矩阵。

```
tf = issymmetric(A)
```

```
tf = false
```

将 A(3,1) 中的元素更改为 1im。

```
A[3,1] = 1im
```

确定修改后的矩阵是否为对称矩阵。。

```
tf = issymmetric(A)
```

```
tf = true
```

矩阵 A 现在是对称矩阵，这是因为它等于其非共轭转置 A.'。

3.2.2.5.8 istril

确定矩阵是否为下三角矩阵

A. 语法

```
tf = istril(A)
```

B. 说明

如果 A 是一个下三角矩阵，则 tf = istril(A) 返回 true；否则返回 false。

C. 示例

(1). 测试下三角矩阵

创建一个 5×5 矩阵。

```
D = tril(magic(5))
```

```
D = 5×5 Matrix{Int64}:
17   0   0   0  0
23   5   0   0  0
4    6  13   0  0
10  12  19  21  0
11  18  25   2  9
```

测试 D 以查看它是否为下三角矩阵。

```
istril(D)
```

```
ans = true
```

(2). 测试零矩阵

创建一个 5×5 零矩阵。

```
Z = zeros(5,5);
```

测试 Z 以查看它是否为下三角矩阵。

```
istril(Z)
```

```
ans = true
```

3.2.2.5.9 istriu

确定矩阵是否为上三角矩阵

A. 语法

```
tf = istriu(A)
```

B. 说明

如果 A 是一个上三角矩阵，则 tf = istriu(A) 返回 true；否则返回 false。

C. 示例

(1). 测试上三角矩阵

创建一个 5×5 矩阵。

```
D = triu(magic(5))
```

```
D = 5×5 Matrix{Int64}:
17  24   1   8  15
0   5   7  14  16
0   0  13  20  22
0   0   0  21   3
```

```
0   0   0   0   9
```

测试 D 以查看它是否为上三角矩阵。

```
istriu(D)
```

```
ans = true
```

(2). 测试零矩阵

创建一个 5×5 零矩阵。

```
Z = zeros(5,5);
```

测试 Z 以查看它是否为上三角矩阵。

```
istriu(Z)
```

```
ans = true
```

3.2.2.6 矩阵属性

矩阵的属性。

函数名	简介
rref	简化的行阶梯形矩阵（Gauss-Jordan 消元法）
opnorm	矩阵范数
cond	逆运算的条件数
rcond	条件数倒数
condskeel	相对条件数
condeig	逆运算的条件数
det	矩阵行列式
null	矩阵的零空间
orth	适用于矩阵范围的标准正交基
rank	矩阵的秩
tr	对角线元素之和
subspace	两个子空间之间的角度
norm	计算范数

3.2.2.6.1 trref

简化的行阶梯形矩阵（Gauss-Jordan 消元法）

A. 语法

```
R = rref(A,tol)
R,p = rref(A)
```

B. 说明

R = rref(A) 使用 Gauss-Jordan 消元法和部分主元消元法返回简化行阶梯形的 A。

C. 示例

(1). 简化行阶梯形矩阵

创建一个矩阵并计算简化行阶梯形矩阵。在这种形式中，矩阵在每列的主元位置具有前导 1。

```
A = magic(3)
```

```
A = 3×3 Matrix{Int64}:
8    1    6
3    5    7
4    9    2
```

```
RA,p = rref(A)
```

```
 RA = 3×3 Matrix{Float64}:
1.0    0.0    0.0
0.0    1.0    0.0
0.0    0.0    1.0
```

3×3 幻方矩阵为满秩，因此简化行阶梯形矩阵是单位矩阵。

现在，计算 4×4 幻方矩阵的简化行阶梯形矩阵。指定两个输出以返回非零主元列。由于该矩阵秩亏，因此结果不是单位矩阵。

```
B = magic(4)
```

```
B = 4×4 Matrix{Int64}:
16     2     3    13
5    11    10     8
9     7     6    12
4    14    15     1
```

```
RB,p = rref(B)
```

```
RB = 4×4 Matrix{Float64}:
```

```
1.0   0.0   0.0    1.0
0.0   1.0   0.0    3.0
0.0   0.0   1.0   -3.0
0.0   0.0   0.0    0.0
p = 3-element Vector{Any}:
1
2
3
```

(2). 增广矩阵的行缩减

对增广矩阵使用 Gauss-Jordan 消元法来求解线性方程组并计算逆矩阵。这些方法主要用于学术研究，因为有更高效和数值稳定的方法来计算这些值。

创建一个 3×3 幻方矩阵。在矩阵的末尾添加一个额外的列。此增广矩阵表示线性方程组 Ax=b，其中额外的列对应于 b。

```
A = magic(3)
A = hcat(A,[1, 1, 1])
```

```
A = 3×4 Matrix{Int64}:
8  1  6  1
3  5  7  1
4  9  2  1
```

计算 A 的简化行阶梯形矩阵。对 R 进行索引以提取该额外（增广）列中的项，该列包含线性方程组的解。

```
R,p= rref(A)

R = 3×4 Matrix{Float64}:
1.0   0.0   0.0   0.0666667
0.0   1.0   0.0   0.0666667
0.0   0.0   1.0   0.0666667
x = R[:,end]
```

```
x = 3-element Vector{Float64}:
0.06666666666666665
0.06666666666666667
0.06666666666666668
```

求解此线性方程组的更高效方法是使用反斜杠运算符，即 x = A\b。

创建一个类似的幻方矩阵，但这次在末尾列追加一个相同大小的单位矩阵。

```
A = [magic(3) diagm(vec(ones(3,1)))]
```

```
A = 3×6 Matrix{Float64}:
8.0   1.0   6.0   1.0   0.0   0.0
3.0   5.0   7.0   0.0   1.0   0.0
4.0   9.0   2.0   0.0   0.0   1.0
```

计算 A 的简化行阶梯形矩阵。在此形式中，额外的列包含该 3×3 幻方矩阵的逆矩阵。

```
R,p = rref(A)
```

```
R = 3×6 Matrix{Float64}:
1.0   0.0   0.0    0.147222    -0.144444     0.0638889
0.0   1.0   0.0   -0.0611111    0.0222222    0.105556
0.0   0.0   1.0   -0.0194444    0.188889    -0.102778
```

```
inv_A = R[:,4:end]
```

```
inv_A = 3×3 Matrix{Float64}:
0.147222    -0.144444     0.0638889
-0.0611111   0.0222222    0.105556
-0.0194444   0.188889    -0.102778
```

计算逆矩阵的更高效方法是使用 innv(A);

(3). 求解方程组

以一个包含四个方程和三个未知数的线性方程组为例。

$x1 + x2 + 5x3 = 6$

$2x1 + x2 + 8x3 = 8$

$x1 + 2x2 + 7x3 = 10$

$-x1 + x2 - x3 = 2.$

创建一个表示该方程组的增广矩阵。

```
A = [1  1  5;
2  1  8;
1  2  7;
-1  1 -1]
b = [6,8,10,2]
M = [A b]
```

使用 rref 以简化行阶梯形矩阵表示该方程组。

```
R,p = rref(M)
```

```
R = 4×4 Matrix{Float64}:
1.0   0.0   3.0   2.0
0.0   1.0   2.0   4.0
0.0   0.0   0.0   0.0
0.0   0.0   0.0   0.0
```

R 的前两行包含表示 x1 和 x2 关于 x3 的方程。接下来的两行表示存在至少一个适合右侧向量的解（否则其中一个方程将显示为 1=0）。第三列不包含主元，因此 x3 是自变量。因此，x1 和 x2 的解有无多个，可以自由选择 x3。

x1=2-3x3

x2=4-2x3.

例如，如果 x3=1，则 x1=-1 且 x2=2。

从数值的角度来看，求解该方程组的更高效方法是使用 x0 = A\b，此方法（对于矩形矩阵 A）计算最小二乘解。在这种情况下，您可以使用 opnorm(A*x0-b)/opnorm(b) 检查解的精确度，通过检查 rank(A) 是否等于未知数的数目来确定解的唯一性。如果存在多个解，则它们都具有 x=x0+nt 形式，其中 n 是零空间 null(A) 且 t 可以自由选择。

3.2.2.6.2 opnorm

矩阵范数

A. 语法

```
n = opnorm(A::AbstractMatrix, p::Real=2)
```

B. 说明

n = opnorm(A::AbstractMatrix, p::Real=2) 计算由向量 p-范数导出的算子范数（或矩阵范数），其中 p 的有效值为 1、2 或 Inf。示例。

C. 示例

(1). 矩阵的 2-范数

计算矩阵的 2-范数，该范数为最大奇异值。

```
X = [2 0 1;-1 1 0;-3 3 0];
n = opnorm(X)
```

```
n = 4.723421263784789
```

3.2.2.6.3 cond

逆运算的条件数

A. 语法

```
C = cond(A)
C = cond(A,p)
```

B. 说明

C = cond(A) 返回 2-范数逆运算的条件数，等于 A 的最大奇异值与最小奇

异值之比。

C = cond(A,p) 返回 p-范数条件数，其中 p 可以是 1、2、Inf。

C. 示例

(1). 矩阵的条件数

计算矩阵的条件数并检查逆运算的敏感度。

创建一个 2×2 矩阵。

```
A = [4.1 2.8;
9.7 6.6]
```

计算 A 的 2-范数条件数。

```
C = cond(A)
```

```
C = 1622.9993838565106
```

由于 A 的条件数远大于 1，因此矩阵对逆运算非常敏感。计算 A 的逆矩阵，对 A 的第二行进行小小的改动，然后再次进行逆运算。

```
invA = inv(A)
```

```
2×2 Matrix{Float64}:
-66.0   28.0
97.0    -41.0
```

```
A2 = [4.1    2.8;
  9.671  6.608]
```

```
A2 = 2×2 Matrix{Float64}:
4.1    2.8
9.671  6.608
```

```
invA2 = inv(A2)
```

```
invA2 =
2×2 Matrix{Float64}:
472.0       -200.0
-690.786    292.857
```

结果表明，对 A 进行的小小改动可能完全改变逆运算的结果。

(2). 1-范数条件数

计算矩阵的 1-范数条件数。

创建一个 3×3 矩阵

```
A = [1 0 -2;
```

```
3 4  6;
-1 5  7]
```

计算 A 的 1-范数条件数。m×n 矩阵的 1-范数条件数的值为

其中 1-范数是矩阵的最大绝对列之和，由以下公式计算得出：

```
C = cond(A,1)
```

```
C = 18.000000000000004
```

对于此矩阵，条件数不是太大，因此矩阵对逆运算不是特别敏感。

3.2.2.6.4 condskeel

相对条件数

A. 语法

```
C = condskeel(A

C = condskeel(A,p)
```

B. 说明

C = condskeel(A) 返回 Inf 范数逆运算的相对条件数。

$$\kappa_S(M,p) = \left\| |M|\,|M^{-1}| \right\|_p \kappa_S(M,x,p) = \frac{\left\| |M||M^{-1}||x| \right\|_p}{\|x\|_p}$$

C = condskeel(A,p) 返回 p-范数相对条件数，其中 p 可以是 1、2、Inf。

C. 示例

(1). 矩阵的相对条件数

计算矩阵的相对条件数并检查逆运算的敏感度。

创建一个 2×2 矩阵。

```
A = [4.1 2.8;
9.7 6.6]
```

计算 A 的 Inf 范数相对条件数。

```
C = condskeel(A)
```

```
C = 1337.6000000000045
```

```
invA = inv(A)
```

2×2 Matrix{Float64}:

-66.0 28.0

97.0 -41.0

```
A2 = [4.1   2.8;
  9.671  6.608]
```

```
A2 = 2×2 Matrix{Float64}:
4.1    2.8
9.671  6.608
```

```
invA2 = inv(A2)
```

```
invA2 =
2×2 Matrix{Float64}:
472.0      -200.0
-690.786    292.857
```

(2). 1-范数相对条件数

计算矩阵的 1-范数相对条件数。

创建一个 3×3 矩阵

```
A = [1 0 -2;
3 4  6;
-1 5  7]
```

```
C = condskeel(A,1)
```

```
C = 9.600000000000001
```

3.2.2.6.5 condeig

逆运算的条件数

A. 语法

```
V,D,s = condeig(A)
```

B. 说明

V,D,s = condeig(A) 等效于

V,D = eig(A)

s = condeig(A)

较大的条件数表示，A 接近具有多重特征值的矩阵。

3.2.2.6.6 det

矩阵行列式

A. 语法

```
d = det(A)
```

B. 说明

d = det(A) 返回方阵 A 的行列式。

C. 示例

(1). 计算矩阵的行列式

创建一个 3×3 的方阵 A。

```
A = [1 -2 4; -5 2 0; 1 0 3]
```

```
3×3 Matrix{Int64}:
1   -2    4
-5   2    0
1    0    3
```

计算 A 的行列式。

```
d = det(A)
```

```
d = -32.0
```

(2). 确定矩阵是否为奇异矩阵

检查为何该行列式不是精确的奇异性度量。

通过将单位矩阵 I(10) 乘以一个小数字来创建一个 10×10 的矩阵。

```
A = Matrix(I(10)*0.0001);
```

矩阵 A 沿主对角线上的元素数非常少。但是，A 不是奇异的，因为它是单位矩阵的倍数。

计算 A 的行列式。

```
d = det(A)
```

```
d = 1.0000000000000003e-40
```

该行列式非常小。abs(det(A)) < tol 形式的容差检验可能将此矩阵标记为奇异矩阵。虽然矩阵的行列式接近于零，但 A 实际并不是病态的。因此，A 不接近奇异矩阵。矩阵的行列式可以任意接近于零，而不传达有关奇异性的信息。

要研究 A 是否为奇异矩阵，请使用 cond 函数

计算 A 的条件数。

```
c = cond(A)
```

```
 c = 1.0
```

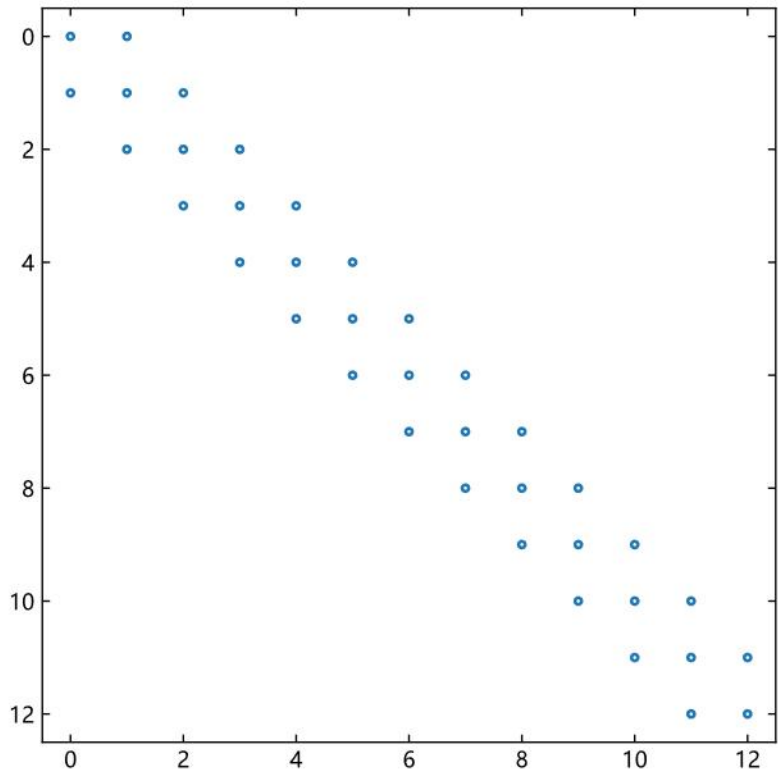
结果确认 A 不是病态的。

(3). 计算奇异矩阵的行列式

检查一个具有较大非零行列式的精确奇异矩阵。从理论上讲，任何奇异矩阵的行列式都为零，但由于浮点计算的性质，这个目标并非总能实现。

创建一个 13×13 的对角占优奇异矩阵 A，并查看非零元素的模式。

```
A = diagm([24, 46 ,64, 78 ,88, 94, 96, 94, 88 ,78 ,64, 46 ,24]);
S = diagm(1=>[-13 ,-24 ,-33, -40, -45, -48, -49, -48, -45, -40, -
33, -24]);
A = A + S + rotl90(S,2);
A=sparse(A)
TyPlot.spy(A)
```



A 是奇异矩阵，因为各行线性相关。例如，sum(A) 生成一个由零值组成的向量。

计算 A 的行列式。

```
d = det(A)
```

```
d = -301705.0341240235
```

A 的行列式相当大，尽管 A 是奇异的。

3.2.2.6.7 null

矩阵的零空间

A. 语法

```
d = det(A)Z = null(A)
Z = null(A,tol)
Z = null(A,"rational")
```

B. 说明

Z = null(A) 返回 A 的零空间的标准正交基。示例

Z = null(A,tol) 也指定容差。小于 tol 的 A 的奇异值被视为零，这会影响 Z 中的列数。示例

Z = null(A,"rational") 返回 A 的零空间的有理基，它通常不是正交基。如果 A 是具有小整数元素的小矩阵，则 Z 的元素是小整数的比率。此方法在数值上不如 null(A) 准确。示例

C. 示例

(1). 矩阵的零空间

使用 null 函数计算矩阵的零空间的标准正交基和有理基向量。矩阵的零空间包含满足 Ax=0 的向量 x。

创建一个由 1 组成的 3×3 矩阵。此矩阵秩亏，其中两个奇异值等于零。

```
A = ones(3,3)
```

```
A = 3×3 Matrix{Float64}:
 1.0  1.0  1.0
 1.0  1.0  1.0
 1.0  1.0  1.0
```

计算 A 的零空间的标准正交基。确认 Ax1=0（在舍入误差内）。

```
x1 = null(A)
```

```
x1 = 3×2 Matrix{Float64}:

  0.816497   0.0
```

```
 -0.408248  -0.707107
 -0.408248   0.707107
```

```
norm(A*x1)
```

```
ans = 8.864419988414313e-16
```

现在计算零空间的有理基。确认 Ax2=0。

```
x2 = null(A,"rational")
```

```
x2 = 3×2 Matrix{Float64}:
 -1.0  -1.0
  1.0   0.0
  0.0   1.0
```

```
norm(A*x2)
```

```
ans = 0.0
```

x1 和 x2 相似，但归一化不同。尽管 x1'*x1 是单位矩阵，但 x2'*x2 不是。

```
x1'*x1
```

```
ans = 2×2 Matrix{Float64}:
1.0         -1.11022e-16
-1.11022e-16   1.0
```

```
x2'*x2
```

```
ans = 2×2 Matrix{Float64}:
2.0  1.0
1.0  2.0
```

正交性通常对数值计算的准确度至关重要。因此，"rational" 选项应仅用于处理全部由整数组成的小型矩阵，此时适合使输出更易于阅读。

(2). 指定零空间的容差

当矩阵具有较小的奇异值时，通过指定容差可更改将哪些奇异值视为零。

创建一个 7×7 Hilbert 矩阵。此矩阵为满秩矩阵，但有一些小的奇异值。

```
H = hilb(7)
```

```
H = 7×7 Matrix{Float64}:
 1.0       0.5       0.333333  0.25      0.2       0.166667   0.142857
 0.5       0.333333  0.25      0.2       0.166667  0.142857   0.125
 0.333333  0.25      0.2       0.166667  0.142857  0.125      0.111111
 0.25      0.2       0.166667  0.142857  0.125     0.111111   0.1
 0.2       0.166667  0.142857  0.125     0.111111  0.1        0.0909091
 0.166667  0.142857  0.125     0.111111  0.1       0.0909091  0.0833333
 0.142857  0.125     0.111111  0.1       0.0909091 0.0833333  0.0769231
```

```
s = svdvals(H)
```

```
s = 7-element Vector{Float64}:
 1.6608853389269314
 0.27192019814934515
 0.021289754908327952
 0.0010085876107701426
 2.9386368145932186e-5
 4.856763361741027e-7
 3.4938985946950782e-9
```

计算 H 的零空间。由于 H 为满秩矩阵，Z 为空。

```
Z = null(H)
```

```
Z = 7×0 Matrix{Float64}
```

现在，再次计算零空间，但指定容差为 1e-4。此容差使 null 将三个奇异值视为零，因此零空间不再为空。

```
Ztol = null(H,1e-4)
```

```
Ztol = 7×3 Matrix{Float64}:
  0.0159939  -0.00246873   0.000245096
 -0.227902    0.0617968   -0.00983864
  0.628753   -0.348675     0.0951956
 -0.200366    0.644687    -0.37132
 -0.497015   -0.174388     0.68253
 -0.184871   -0.543628    -0.591035
  0.480756    0.364742     0.194406
```

验证 H*Ztol 与指定的容差相比具有可忽略元素。

```
norm(H*Ztol)
```

```
ans = 2.9390381530692962e-5
```

(3). 欠定方程组的通解

求某欠定方程组的一个特定解，然后获得所有解的通用形式。

欠定线性方程组 Ax=b 包含的未知数比方程多。欠定方程组可以有无限多个解或没有解。当方程组有无限多个解时，它们都位于一条直线上。该线上的各点均通过零空间向量的线性组合获得。

创建一个 2×4 系数矩阵并使用反斜杠求解方程 Ax0=b，其中 b 是由 1

组成的向量。用反斜杠计算问题的最小二乘解。

```
A = [1 8 15 67; 7 14 16 3]
```

```
A = 2×4 Matrix{Int64}:
 1   8  15  67
 7  14  16   3
```

```
b = ones(2)
x0 = A\b
```

```
x0 = 4-element Vector{Float64}:
0.013882911347701407
0.027624559139987476
0.031433024593985
0.004382466367501843
```

欠定方程组的完全通解具有 x=x0+Ny 形式，其中：N 是 A 的零空间。y 是合适长度的任何向量。x0 是用反斜杠计算的解。

计算 A 的零空间，然后使用结果来构造方程组的另一个解。检查新解是否在舍入误差内满足 Ax=b。

```
N = null(A)
```

```
N = 4×2 Matrix{Float64}:
0.920746    0.213101
-0.378428    0.677809
-0.0809974  -0.699924
0.0495767   0.0725865
```

```
x = x0 + N*[1; -2]
```

```
x = 4-element Vector{Float64}:
0.5084281060949924
-1.7064219415464472
1.3502844762737936
-0.09121372713179647
```

```
norm(A*x-b)
```

```
ans = 4.4782604170571826e-14
```

3.2.2.6.8 orth

适用于矩阵范围的标准正交基

A. 语法

```
Q = orth(A)
Q = orth(A,tol)
```

B. 说明

Q = orth(A) 返回适用于 A 的范围的一个标准正交基。矩阵 Q 的列是涵盖了 A 范围的向量。Q 中列的数量等于 A 的秩。示例

Q = orth(A,tol) 也指定容差。小于 tol 的 A 的奇异值被视为零，这会影响 Q 中的列数。

C. 示例

(1). 适用于满秩矩阵的基向量

计算并验证适用于满秩矩阵范围的标准正交基向量。

定义一个矩阵并计算它的秩。

```
A = [1 0 1;-1 -2 0; 0 1 -1]
r = rank(A)
```

```
r = 3
```

由于 A 是一个满秩方阵，orth(A) 计算出的标准正交基与奇异值分解 U,S = svd(A) 中计算出的矩阵 U 一致。原因是 A 的奇异值均为非零值。

使用 orth 计算适用于 A 的范围的标准正交基)。

```
Q = orth(A)
```

```
Q = 3×3 Matrix{Float64}:
 -0.12      -0.809712  0.574427
  0.901753   0.153123  0.404222
 -0.415261   0.566498  0.711785
```

Q 中的列数等于 rank(A)。由于 A 为满秩矩阵，Q 和 A 具有相同的大小。

在合理误差界限内验证基向量 Q 是正交、归一化向量。

```
E = norm(eye(r)-Q'*Q)
```

```
E = 1.085665291867152e-15
```

误差与 eps() 的量级相当。

(2). 秩亏矩阵的基向量

计算并验证适用于秩亏矩阵范围的标准正交基向量。

定义一个奇异矩阵并计算它的秩。

```
A = [1 0 1; 0 1 0; 1 0 1]
```

```
r = rank(A)
```

```
r = 2
```

由于 A 是一个秩亏矩阵，orth(A) 计算出的标准正交基仅与奇异值分解 U,S = svd(A) 中计算出的矩阵 U 的前 r = 2 列一致。原因是 A 的奇异值并不全是非零值。

使用 orth 计算适用于 A 的范围的标准正交基。

```
Q = orth(A)
```

```
Q = 3×2 Matrix{Float64}:
 -0.707107  0.0
  0.0       1.0
 -0.707107  0.0
```

由于 A 为秩亏矩阵，Q 包含的列比 A 少一个。

(3). 指定基向量的容差

当矩阵具有较小的奇异值时，通过指定容差可更改将哪些奇异值视为零。

创建一个 7×7 Hilbert 矩阵。此矩阵为满秩矩阵，但有一些小的奇异值。

```
H = hilb(7)
```

```
H = 7×7 Matrix{Float64}:
1.0       0.5       0.333333  0.25      0.2       0.166667   0.142857
0.5       0.333333  0.25      0.2       0.166667  0.142857   0.125
0.333333  0.25      0.2       0.166667  0.142857  0.125      0.111111
 0.25      0.2       0.166667  0.142857  0.125     0.111111   0.1
 0.2       0.166667  0.142857  0.125     0.111111  0.1        0.0909091
 0.166667  0.142857  0.125     0.111111  0.1       0.0909091  0.0833333
 0.142857  0.125     0.111111  0.1       0.0909091 0.0833333  0.0769231
```

计算适用于 H 的范围的一个标准正交基。由于 H 为满秩矩阵，Q 和 H 具有相同的大小。

```
Q = orth(H)
```

```
Q = 7×7 Matrix{Float64}:
 -0.733226   0.623239   0.260843   -0.0751873   0.0159939  -0.00246873
0.000245096
 -0.436359  -0.163072  -0.670558    0.526778   -0.227902    0.0617968
-0.00983864
 -0.319779  -0.321514  -0.29533    -0.425656    0.628753   -0.348675
0.0951956
 -0.254886  -0.357367   0.0230466  -0.461676   -0.200366    0.644687
-0.37132
 -0.212844  -0.357068   0.233688   -0.171196   -0.497015   -0.174388
0.68253
```

```
 -0.183143   -0.34457     0.367876     0.182664    -0.184871    -0.543628
-0.591035
 -0.16094    -0.328135    0.452349     0.509755     0.480756     0.364742
0.194406
```

现在，再次计算标准正交基向量，但指定容差为 1e-4。此容差使 orth 将三个奇异值视为零，因此标准正交基只有四列。

```
Qtol = orth(H,1e-4)
```

```
Qtol = 7×4 Matrix{Float64}:
 -0.733226   0.623239   0.260843   -0.0751873
 -0.436359  -0.163072  -0.670558    0.526778
 -0.319779  -0.321514  -0.29533    -0.425656
 -0.254886  -0.357367   0.0230466  -0.461676
 -0.212844  -0.357068   0.233688   -0.171196
 -0.183143  -0.34457    0.367876    0.182664
 -0.16094   -0.328135   0.452349    0.509755
```

3.2.2.6.9 rank

矩阵的秩

A. 语法

```
k = rank(A)
k = rank(A,atol)
k = rank(A,rtol)
```

B. 说明

k = rank(A) 返回矩阵 A 的秩。

k = rank(A,atol) 指定在秩计算中使用绝对容差。秩计算为 A 中大于 atol 的奇异值的个数。

k = rank(A,rtol) 指定在秩计算中使用相对容差。秩计算为 A 中大于 rtol 的奇异值的个数。

C. 示例

(1). 矩阵的秩

确定矩阵是否满秩。

创建一个 3×3 矩阵。第三列中的值是第二列中的值的两倍。

```
A = [3 2 4; -1 1 2; 9 5 10]
```

```
3×3 Matrix{Int64}:
3   2   4
-1  1   2
9   5  10
```

计算矩阵的秩。如果矩阵满秩，则秩等于列数，size(A,2)。

```
rank(A)
```

```
ans = 2
```

```
size(A,2)
```

```
ans = 3
```

由于列是线性相关的，因此该矩阵秩亏。

(2). 指定秩绝对容差

使用相对容差计算矩阵的秩。

创建一个 4×4 对角矩阵。对角线上有一个等于 1e-15 的小值。

```
A = [10 0 0 0; 0 25 0 0; 0 0 34 0; 0 0 0 1e-15]
```

```
4×4 Matrix{Float64}:
10.0    0.0    0.0    0.0
0.0    25.0    0.0    0.0
0.0     0.0   34.0    0.0
0.0     0.0    0.0    1.0e-15
```

计算矩阵的秩。

```
rank(A)
```

```
ans = 3
```

由于默认算法计算大于 max(size(A))*eps(norm(A)) 的奇异值的个数，因此该矩阵不被视为满秩。对于此矩阵，对角线上的小值被排除，因为它小于容差。

再次计算矩阵的秩，但指定容差 1e-16。

```
rank(A,atol=1e-16)
```

```
ans = 4
```

3.2.2.6.10 tr

对角线元素之和

A. 语法

```
b = tr(A)
```

B. 说明

b = tr(A) 计算矩阵 A 的对角线元素之和：示例。

$$tr（A） = \sum_{i=1}^{n} a_{ii} = a_{11} + a_{22} + ... + a_{nn}$$

C. 示例

(1). 矩阵对角线之和

创建一个 3×3 矩阵，并计算对角线元素之和。

```
A = [1 -5 2;-3  7 9;4 -1 6]
b = tr(A)
```

```
b = 14
```

结果 tr(A)=14 与手动计算一致。

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} 1 & -5 & 2 \\ -3 & 7 & 9 \\ 4 & -1 & 6 \end{bmatrix}$$

$$\mathrm{tr}(A) = \sum_{i=1}^{3} a_{ii} = a_{11} + a_{22} + a_{33} = 1 + 7 + 6 = 14$$

(2). 矩阵的迹属性

验证矩阵的迹的几个属性（基于舍入误差）。

创建两个矩阵。验证 tr(A+B)=tr(A)+tr(B)。

```
A = magic(3)
B = rand(3,3)
tr(A+B)
```

```
ans = 16.68561041304877
```

```
tr(A) + tr(B)
```

```
ans = 16.68561041304877
```

验证 $\mathrm{tr}(A) = \mathrm{tr}(A^T)$。

```
tr(A)
```

```
ans = 15
```

```
tr(A')
```

```
ans = 15
```

验证 $\mathrm{tr}(A^T B) = \mathrm{tr}(AB^T)$。

```
tr(A'*B)
```

```
ans = 26.143337768203708
```

验证对于标量 c，满足 tr(cA)=c tr(A)

```
c = 5
tr(c*A)
```

```
ans = 75
```

验证迹等于特征值之和，即 $\mathrm{tr}(A) = \sum_i \lambda_i$。

```
tr(A)
```

```
ans = 15
```

```
sum(eigvals(A))
```

```
ans = 15.000000000000005
```

3.2.2.6.11 subspace

两个子空间之间的角度

A. 语法

```
theta = subspace(A,B)
```

B. 说明

theta = subspace(A,B) 计算 A 和 B 的列指定的两个子空间之间的角度。如果 A 和 B 是单位长度的列向量，则此角度与 acos(abs(A'*B)) 相同。

C. 示例

以列为正交的 Hadamard 矩阵的两个子空间为例。

```
H = hadamard(8)
A = H[:,2:4]
B = H[:,5:8]
```

请注意，矩阵 A 和 B 的大小不同 - A 具有三列，B 具有四列。要计算两个子空间之间的角度，它们的大小无需相同。在几何上，该角度是嵌入更高维度空间中的两个超平面之间的角度。

```
theta = subspace(A,B)
```

```
theta = 1.5707963267948966
```

通过 theta 等于 π/2 这一事实可以得出，A 和 B 是正交的。

```
theta - pi/2
```

```
ans = 0.0
```

3.2.2.6.12 norm

计算范数

A. 语法

```
norm(A,P::Real=2)
```

B. 说明

对于任何可迭代容器 A（包括任何维度）的数字（或任何元素类型范数已定义），计算 p 范数（默认为 p=2）就好像 A 是对应的向量长度。示例

矩阵的 p 范数被定义为

$$\|A\|_p = \left(\sum_{i=1}^{n} |a_i|^p\right)^{1/p}$$

C. 示例

(1). 向量模

创建一个向量并计算模。

```
v = [1 -2 3]
n = norm(v)
```

```
ans = 3.7416573867739413
```

(2). 向量的 1-范数

```
X = [-2 ,3 ,-1]
n = norm(X,1)
```

```
ans = 6.0
```

(3). 两个点之间的欧几里得距离

计算两个点之间的距离作为向量元素之差的范数。

创建两个向量，表示欧几里德平面上两个点的 (x,y) 坐标。

```
a = [0 3]
b = [-2 1];
```

使用 norm 来计算点之间的距离。

```
d = norm(b-a)
```

```
d = 2.8284271247461903
```

在几何学上，点之间的距离等于从一个点延伸到另一个点的向量的模。

$$
\begin{aligned}
a &= 0\hat{i} + 3\hat{j} \\
b &= -2\hat{i} + 1\hat{j} \\
d_{(a,b)} &= \|b - a\| \\
&= \sqrt{(-2-0)^2 + (1-3)^2} \\
&= \sqrt{8}
\end{aligned}
$$

(4). 矩阵的无穷范数

计算矩阵的 1-范数

创建一个 2×2 矩阵。

```
A = [4.1 2.8;
9.7 6.6]
```

计算 A 的 1-范数

```
C = norm(A,Inf)
```

```
C = 9.7
```

3.2.3 随机数生成

种子、分布、算法。

使用 rand 和 randn 函数创建伪随机数序列，使用 randperm 函数创建随机置换整数向量。

3.2.3.1 创建随机值

创建随机数数组。

函数名	简介
mt19937ar	mt19937ar 随机种子算法
rand	均匀分布的随机数
randi	均匀分布的伪随机整数
randn	标准正态分布的随机数
randg	标准高斯分布的随机数
randperm	随机排列

bitrand	生成一个随机布尔值的 BitArray
randpermk	整数的随机排列

3.2.3.1.1 mt19937ar

生成 mt19937ar RNG 对象

A. 语法

```
mt19937ar()
mt19937ar(seed)
```

B. 说明

mt19937ar() 生成随机的 RNG 对象。

mt19937ar(seed) 根据 seed 生成不同的 RNG 对象。

C. 示例

(1). 生成随机数

设置 seed 为 1234 生成 RNG 对象

```
rng = mt19937ar(1234);
```

利用 rng 生成 3×4 矩阵

```
r = rand(rng,3,4)
```

```
3×4 Matrix{Float64}:
 0.191519  0.785359  0.276464  0.875933
 0.622109  0.779976  0.801872  0.357817
 0.437728  0.272593  0.958139  0.500995
```

(2). 控制随机数生成

设置 seed 为 1234 生成 RNG 对象

```
rng = mt19937ar(1234);
```

利用 rng 生成 3×4 矩阵

```
r1 = rand(rng,3,4)
```

```
3×4 Matrix{Float64}:
 0.191519  0.785359  0.276464  0.875933
 0.622109  0.779976  0.801872  0.357817
```

```
 0.437728  0.272593  0.958139  0.500995
```

```
r2 = rand(rng,3,4)
```

```
3×4 Matrix{Float64}:
 0.683463  0.561196   0.772827  0.615396
 0.712702  0.503083   0.882641  0.0753812
 0.370251  0.0137684  0.364886  0.368824
```

```
rng1 = mt19937ar(1234);
r = rand(rng1,3,4)
r == r1
```

```
ans = true
```

3.2.3.1.2 rand

均匀分布的随机数

A. 语法

```
X = rand()
X = rand(n)
X = rand(sz1,...,szN)
X = rand(S)
```

B. 说明

X = rand() 返回一个在区间 (0,1) 内均匀分布的随机数。

X = rand(n) 返回一个长度为 n 的随机数向量。

X = rand(sz1,...,szN) 返回由随机数组成的 sz1 × ... × szN 数组，其中 sz1,...,szN 指示每个维度的大小。例如：rand(3,4) 返回一个 3×4 的矩阵。

X = rand(S)返回数组 S 的元素中的随机数。

C. 示例

(1). 由随机数组成的向量

生成一个由介于 0 和 1 之间的均匀分布的随机数组成的 5 元素向量。

```
r = rand(5)
```

```
r = 5-element Vector{Float64}:
0.1608778003373621
0.7917543589426914
0.04703605439813985
0.8414934941319905
```

```
0.3834737865286717
```

(2). 指定区间内的随机数

生成一个由区间 (-5,5) 内均匀分布的数字组成的 10×1 向量。

```
r = -5 .+ (5+5)*rand(10)
```

```
r = 10-element Vector{Float64}:
-1.0202981638234232
-4.10199231770325
-1.787437578978719
-3.122894375964098
2.598442752032586
-0.748159672688427
3.671559125471786
-0.34762329003954484
-4.908743371337265
-4.881505265978328
```

一般来说，可以使用公式 `r = a + (b-a).*rand(N)` 生成区间 `(a,b)` 内的 `N` 个随机数。

(3). 随机整数

```
r = rand(10:50,5)
```

```
5-element Vector{Int64}:
27
22
39
44
45
```

(4). 随机复数

生成一个实部和虚部位于区间 (0,1) 内的随机复数。

```
a = rand() + 1im*rand()
```

```
a = 0.45296329631735377 + 0.18573297445822812im
```

(5). 由随机数组成的三维数组

创建一个由随机数组成的 3×2×3 数组。

```
X = rand(3,2,3)
```

```
X = 3×2×3 Array{Float64, 3}:
[:, :, 1] =
0.658638  0.748294
```

```
0.237578  0.580883
0.910355  0.774707

[:, :, 2] =
0.0424328  0.822953
0.533143   0.851095
0.408082   0.964115

[:, :, 3] =
0.394665  0.741083
0.41216   0.251097
0.407681  0.807933
```

(6). 指定随机数的数据类型。

创建一个由其元素为 Float16 的随机数组成的向量。

```
r = rand(Float16,4)
```

```
r = 4-element Vector{Float16}:
0.4854
0.668
0.967
0.699
```

```
eltype(r)
```

```
ans = Float16
```

(7). 根据随机数算法生成随机数。

设置随机数种子,生成 5 个元素向量。

```
rng = mt19937ar(1234)
r = rand(rng,5)
```

```
5-element Vector{Float64}:
 0.1915194503788923
 0.6221087710398319
 0.4377277390071145
 0.7853585837137692
 0.7799758081188035
```

3.2.3.1.3 randi

均匀分布的伪随机整数

A. 语法

```
X = randi(imax)
X = randi(imax, n)
X = randi(imax, sz1,...,szN)
X = randi(imax, sz)
X = randi((imin,imax),__)
X = randi(rng, __)
```

B. 说明

X = randi(imax)返回一个介于 1 和 imax 之间的伪随机整数标量。

X = randi(imax , n)返回 n×1 向量，其中包含从区间(1,imax)的均匀离散分布中得到的伪随机整数。

X = randi(imax , sz1, ..., szN)返回 sz1×...×szN 数组，其中 sz1,...,szN 指示每个维度的大小。例如，randi(10, 3, 4)返回一个介于 1 和 10 之间的伪随机整数组成的 3×4 数组。

X = randi(imax , sz)返回一个数组，其中大小元组 sz 定义 size(X)。例如 rand(10, (3, 4))返回一个由介于 1 和 10 之间的伪随机整数组成的 3×4 数组。

X = randi((imin ,imax),___) 使用以上任何语法返回一个数组，其中包含从区间 (imin,imax) 的均匀离散分布中得到的整数。

X = randi(rng , ...)从随机数流 rng 而不是默认全局流生成整数。要创建一个流，请用 mt19937ar。在每次生成随机数前指定 s，后跟上述语法中的任意参数组合。

C. 示例

(1). 由随机整数组成的向量

生成一个由介于 1 和 10 之间的随机整数组成的 5×1 向量。randi 的第一个输入指示采样区间中的最大整数（采样区间中的最小整数为 1）。

```
r = randi(10, 5)
```

```
5-element Vector{Int64}:
 6
 8
 1
 7
 9
```

(2). 指定区间的随机整数

生成一个由样本区间(-5, 5)中均匀分布的随机整数组成的 10×1 向量。

r

```
= randi((-5,5), 10)
```

```
10-element Vector{Int64}:
  1
 -5
 -1
  2
 -1
 -4
  0
  1
  3
  1
```

(3). 控制随机数生成

保存随机数生成器的状态并创建一个由随机整数组成的 1×5 矩阵。

```
rng = mt19937ar(5489);
r = randi(rng, 10, 1, 5)
```

```
1×5 Matrix{Int64}:
 9  10  2  10  7
```

将随机数生成器的状态恢复为 rng，然后创建一个由随机整数组成的新 1×5 矩阵。值与之前相同。

```
rng = mt19937ar(5489);
r = randi(rng, 10, 1, 5)
```

```
1×5 Matrix{Int64}:
 9  10  2  10  7
```

(4). 由随机整数组成的三维数组

创建一个由介于 1 和 500 之间的均匀分布的随机整数组成的 3×2×3 矩阵。

```
X = randi(500, (3, 2, 3))
```

```
3×2×3 Array{Int64, 3}:
[:, :, 1] =
 371  414
  32  101
 432  239

[:, :, 2] =
 236  456
```

```
 236  462
 162  384

[:, :, 3] =
 362   88
 121   71
 294  146
```

(5). 现有数组定义的大小和数值数据类型

创建一个 8 位有符号整数,和一个 2×2 矩阵。

```
imax = Int8(10)
p = [3 2;-2 1]
```

创建一个与 imax 具有相同类型，与 p 具有相同大小的数组。

```
sz = size(p)
X = randi(imax, sz)
```

```
2×2 Matrix{Int8}:
 8   5
 2  10
```

```
eltype(X)
```

```
Int8
```

3.2.3.1.4 randn

标准正态分布的随机数

A. 语法

```
X = randn()
X = randn(n)
X = randn(sz1,...,szN)
X = randn(S)
```

B. 说明

X = randn() 返回一个在标准正态分布的随机数。

X = randn(n) 返回一个的随机数向量。

X = randn(sz1,...,szN) 返回由随机数组成的 sz1 × ... × szN 数组，其中 sz1,...,szN 指示每个维度的大小。例如：randn(3,4) 返回一个 3×4 的矩阵。

X = randn(S)返回数组 S 的元素中的随机数。

C. 示例

(1). 由随机数组成的向量

生成一个由介于 0 和 1 之间的标准正态分布的随机数组成的 5×5 矩阵。

```
r = randn(5)
```

```
r = 5-element Vector{Float64}:
1.0902945078621222
-2.1121288828572378
-0.12611650358916432
-0.6049412869655166
-0.5341253700061359
```

(2). 二元正态随机数

通过指定的平均向量和协方差矩阵基于二元正态分布生成值。

```
mu = [1 2]
sigma = [1 0.5; 0.5 2]
R = cholesky(sigma).U
z = repeat(mu,10,1) + randn(10,2)*R
```

```
z = 10×2 Matrix{Float64}:
0.48824   2.09821
1.36446   0.695447
0.440456  2.16663
1.9197    2.97419
0.549667  1.25502
0.567936  0.392197
0.892299  3.72201
1.8445    0.643431
0.869312  2.77464
1.82582   2.61734
```

(3). 随机复数

生成一个实部和虚部位于区间 (0,1) 内的随机复数。

```
a = randn() + 1im*randn()
```

```
a = 1.0919097585678172 - 1.9378467877791108im
```

(4). 由随机数组成的三维数组

创建一个由随机数组成的 3×2×3 数组。

```
X = randn(3,2,3)
```

```
X = 3×2×3 Array{Float64, 3}:
```

```
[:, :, 1] =
0.117305  -0.462586
0.949292   0.801129
-0.435874   1.37038

[:, :, 2] =
2.04734    0.628816
1.31181   -0.50664
-0.699963  -0.480583

[:, :, 3] =
-0.510132   -0.515821
-0.323743   -0.0147875
0.0381542   0.327881
```

(5). 指定随机数的数据类型。

创建一个由其元素为 Float16 的随机数组成的向量。

```
r = randn(Float16,4)
```

```
r = 4-element Vector{Float16}:
1.379
1.567
1.117
0.11505
```

```
eltype(r)
```

```
ans = Float16
```

(6). 根据随机数算法生成随机数。

设置随机数种子,生成 5 个元素向量。

```
rng = mt19937ar(1234)
r = randn(rng,5)
```

```
5-element Vector{Float64}:
 -0.9472466439573711
  0.5401497470703477
 -0.21660214097627592
  1.1890319749483447
  1.316987831810854
```

3.2.3.1.5 randg

具有单位尺度的 Gamma 随机数

A. 语法

```
Y = randg()
Y = randg(A)
Y = randg(A,m)
Y = randg(A,m,n,p,...)
Y = randg(A,(m,n,p,...))
```

B. 说明

Y = randg() 返回从具有单位尺度和形状的伽马分布中选择的标量随机值。

Y = randg(A) 返回从单位尺度的伽马分布中选择的随机值。

Y = randg(A,m) 返回从具有形状参数 A 的伽马分布中选择的随机值的 m ×1 向量。

Y = randg(A,m,n,p,...) 或 Y = randg(A,(m,n,p,...)) 返回一个 m×n×p×- ... 从具有形状参数 A 的 gamma 分布中选择的随机值数组。

randg 使用函数 rand 和 randn 生成伪随机数。生成的数字序列由位于 rand 和 randn 基础的统一随机数生成器的设置决定

TIP

要生成 gamma 随机数并指定比例和形状参数，您应该调用 gamrnd。

C. 示例

(1). 示例 1

从形状参数为 3 的伽马分布中生成一个 100×1 的值数组。

```
r = randg(3,100,1);
```

(2). 示例 2

生成一个 100×2 数组，该数组从形状参数为 3 和 2 的 gamma 分布中提取。

```
A = [ones(100,1)*3 ones(100,1)*2];
r = randg.(A);
```

(3). 示例 3

要从 randg 创建可重现的输出，请指定 randg 使用的随机数种子。

```
rng = mt19937ar(5489);
randg(rng,3, 1, 5)
```

```
1×5 Matrix{Float64}:
6.92232  4.33691  1.05052  3.26621  11.3269
```

(4). 示例 4

保存 rand 和 randn 使用的随机数生成器的设置，从 randg 生成 5 个值，恢复设置，然后重复这些值。

```
rng1 =  deepcopy(rng);
randg(rng,10, 1, 5)
```

```
1×5 Matrix{Float64}:
9.47194  9.0433  15.0774  14.7763  6.37747
```

```
randg(rng1,10, 1, 5)
```

```
1×5 Matrix{Float64}:
9.47194  9.0433  15.0774  14.7763  6.37747
```

r2 包含与 r1 完全相同的值。

3.2.3.1.6 randperm

随机排列

A. 语法

```
p = randperm(n)
```

B. 说明

p = randperm(n) 返回行向量，其中包含从 1 到 n 没有重复元素的整数随机排列。

C. 示例

整数的随机排列生成一个从 1 到 6 的整数的随机排列。randperm 的输入指示采样间隔中的最大整数（采样区间中的最小整数为 1）。

```
r = randperm(6)
```

```
r = 6-element Vector{Int64}:
2
6
3
5
```

```
4
1
```

3.2.3.1.7 bitrand

生成一个随机布尔值的 BitArray

A. 语法

```
p = bitrand(n...)
```

B. 说明

p = bitrand(n) 生成一个随机 Bool 值的 BitArray。

C. 示例

(1). 生成随机 Bool 值矩阵生成随机 Bool 值 3×4 矩阵。

```
r = bitrand(3,4)
```

```
r =
3×4 BitMatrix:
 0  0  1  1
 0  0  0  0
 0  1  1  1
```

3.2.3.1.8 randpermk

整数的随机排列

A. 语法

```
p = randpermk(n,k)
p = randpermk(rng,__)
```

B. 说明

p = randpermk(n,k)返回行向量，其中包含在 1 到 n 之间随机选择的 k 个唯一整数。示例

p = randpermk(rng,__)从随机数种子 rng 而不是默认全局种子生成整数的随机排列。示例。

C. 示例

(1). 排列所选的整数

生成一个从整数 1 到 8 中随机选择的四个唯一整数（没有重复元素）的随机排列。

```
r1 = randpermk(8,4)
```

```
r1 = 4-element Vector{Int64}:
 6
 4
 7
 3
```

生成四个唯一整数的另一个随机排列。

```
r2 = randpermk(8,4)
```

```
r2 = 4-element Vector{Int64}:
 8
 7
 5
 4
```

(2). 控制随机数生成

使用随机数种子，生成一个从整数 1 到 8 中随机选择的 6 个唯一整数（没有重复元素）的随机排列。

```
rng = MT19937ar(1234)
r = randpermk(rng,8,6)
```

```
r = 6-element Vector{Int64}:
 4
 2
 1
 8
 7
 5
```

3.2.4 插值

网格和散点数据插值、数据网格化、分段多项式

插值是在一组已知数据点的范围内添加新数据点的技术。您可以使用插值来填充缺失的数据、对现有数据进行平滑处理以及进行预测等。

3.2.4.1 一维插值和网格插值

函数名	简介
interp1	一维插值
interp2	二维插值
interp3	三维插值
interpn	n 维插值
griddedInterpolant	网格插值
akima	Akima 分段三次 Hermite 插值
interpft	一维插值（FFT 方法）
makima	修正 Akima 分段三次 Hermite 插值
pchip	分段三次 Hermite 插值多项式 (PCHIP)
interpolate	插值
extrapolate	外插
LagrangeInterp	拉格朗日插值
NewtonInterp	牛顿插值
ConstantInterpolation	最邻近插值
LinearInterpolation	线性插值
CubicSplineInterpolation	三次样条插值
sinc_interpolate	一维插值（FFT 方法）
mkpp	生成分段多项式
unmkpp	提取分段多项式详细信息
ppval	计算分段多项式
padecoef	时滞的 Padé 逼近

3.2.4.1.1 interp1

一维数据插值(表查找)

A. 语法

```
vq = interp1(x,v,xq)
vq = interp1(x,v,xq,method
vq = interp1(x,v,xq,method,extrapolation)
vq = interp1(v,xq)
vq = interp1(v,xq,method)
vq = interp1(v,xq,method,extrapolation)
```

B. 说明

vq=interp1(x,v,xq) 使线性插值返回一维函数在特定查询点的插入值，向量 x 包含样本点，包含对应值 v(x)。向量 xq 包含查询点的坐标。

如果您有多个在同一点坐标采样的数据集，则可以将 v 以数组的形式进行传递，数组 v 的每一列都包含一组不同的一维样本值。

vg =interp1(xv,xg,method) 指定备选插值方法:'inear'、'nearest'、'next'、'previous'、'pchip'、'cubic'、'v5cubic'、'makima'或 'spline'。默认方法为'inear'。

vg = interp1(v,xq)返回插入的值，并假定一个样本点坐标默认集。默认点是从 1 到 n 的数字序列，其中 n 取决于 v 的形状:

当 v 是向量时，默认点是 1:length(v)。

当 v 是数组时，默认点是 1:size(v,1)。

如果您不在意点之间的绝对距离，则可使用此语法。

vq = interp1(v,xq,method) 指定备选插值方法中的任意一种，并使用默认样本点。

vq = interp1(v,xq,method,extrapolation) 指定外插策略，并使用默认样本点。

注意

不建议使用该语法。请改用 griddedInterpolant。

C. 实例

(1). 基于粗略采样的正弦函数进行插值

定义样本点 x 及其对应样本值 v。

```
x = 0:pi/4:2*pi
v = sin.(x)
```

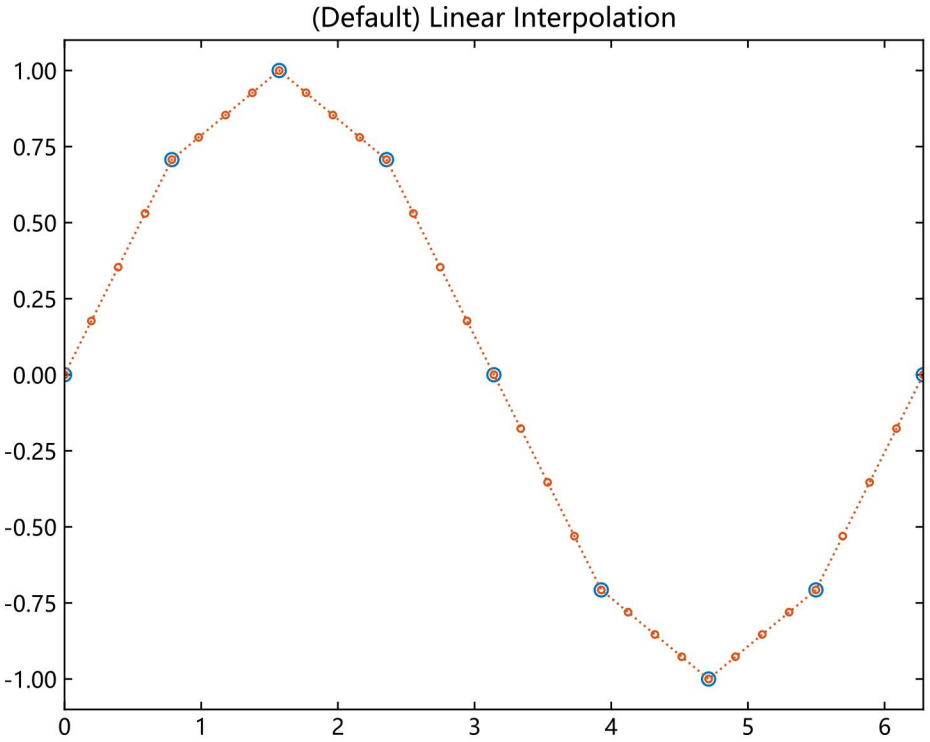
将查询点定义为 x 范围内更精细的采样点。

```
xq = 0:pi/16:2*pi
```

在查询点插入函数并绘制结果。

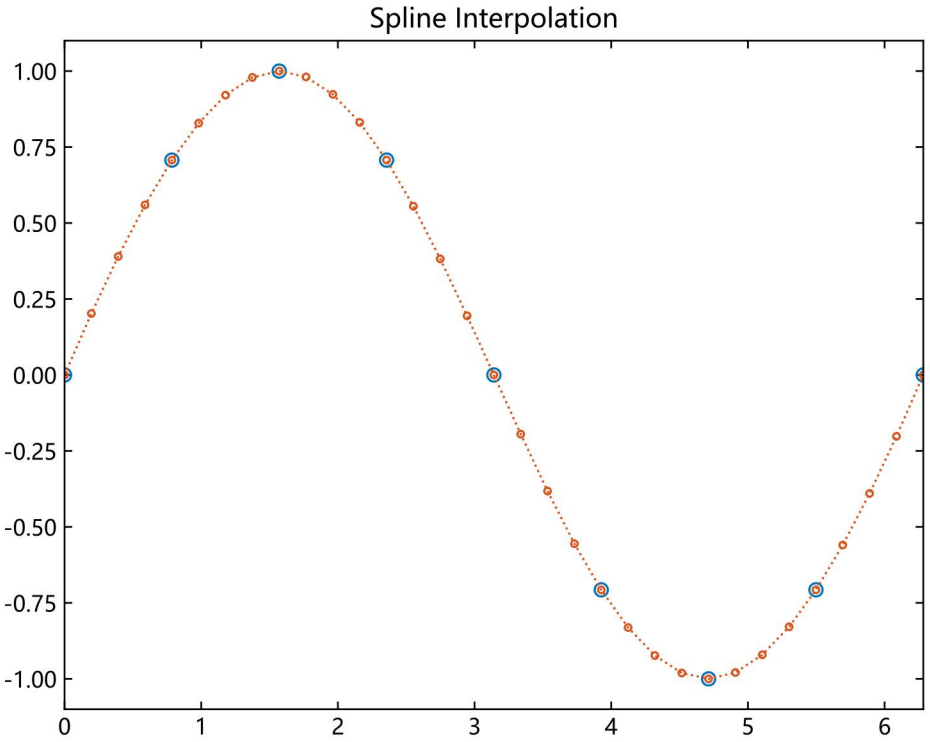
```
figure()
vq1= interp1(x,v,xq)
```

```
plot(x,v,"o",xq,vq1,":.")
xlim([,2*pi])
title("(Default) Linear Interpolation")
```



现在使用“spline”方法计算相同点处的 v。

```
figure()
vq2 =interp1(x,V,xq,"spline")
plot(x,v,"o",xq,vq2,":.")
xlim([0,2*pi])
title("spline Interpolation")
```



(2). 在不指定样本点的情况下进行插值

定义一组函数值。

```
v = [0,1.41,2,1.41,0,-1.41,-2,-1.41,0]
```

定义一组介于默认点 1:9 之间的查询点。在这种情况下，默认点为 1:9，因为 v 包含 9 个值.

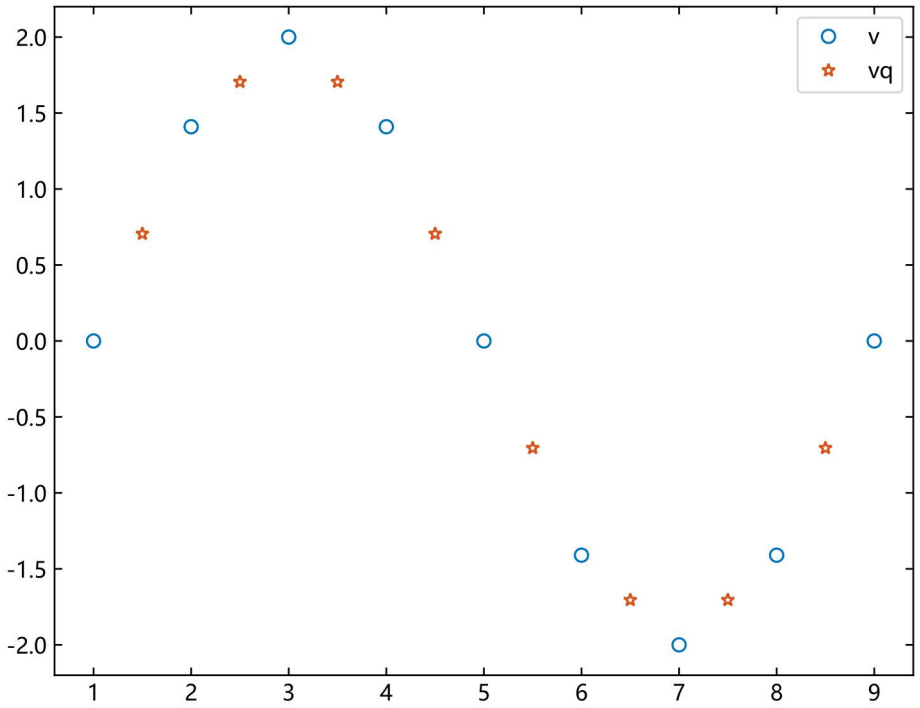
```
xq = 1.5:8.5
```

计算 xq 处的 v。

```
vq = interp1(v,xq)
```

绘制结果。

```
figure()
plot(1:9,v,"o",xq,vq,”*”)
legend(["v","vq"])
```



(3). 复数值插值

定义一组样本点。

```
x= 1:10
```

定义函数 v(z)= 5 x+x^2*1im 在样本点处的值。

```
v=@.(5*x)+(x^2*lim)
```

将查询点定义为 x 范围内更精细的采样点。

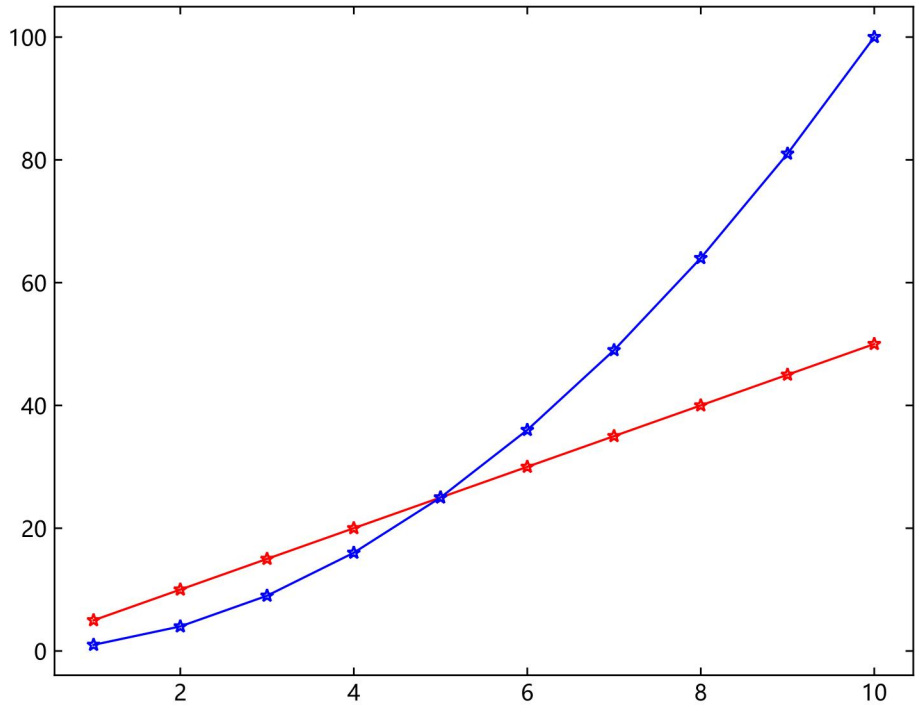
```
xq = 1:0.25:10
```

在查询点处对 v 进行插值。

```
vq = interp1(x,v,xq)
```

用红色绘制结果的实部，用蓝色绘制虚部。

```
figure()
plot(x,real(v),"*r",xq,real(vq),”-r”)
hold("on")
plot(x,imag(v),"*b",xq,imag(vq),"-b")
```

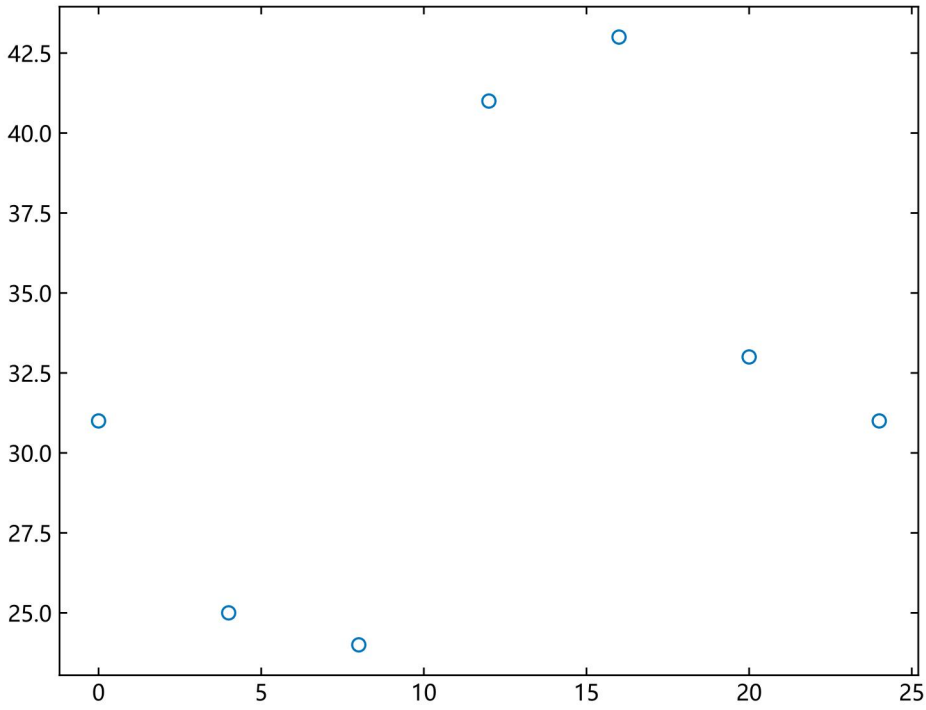


(4). 日期和时间的插值

对时间戳数据点进行插值。

以包含温度读数的数据集为例，这些读数每四个小时测量一次，绘制数据图。

```
x=0:4:24
T =[31,25,24,41,43 ,33,31]
plot(x, T, "o")
```

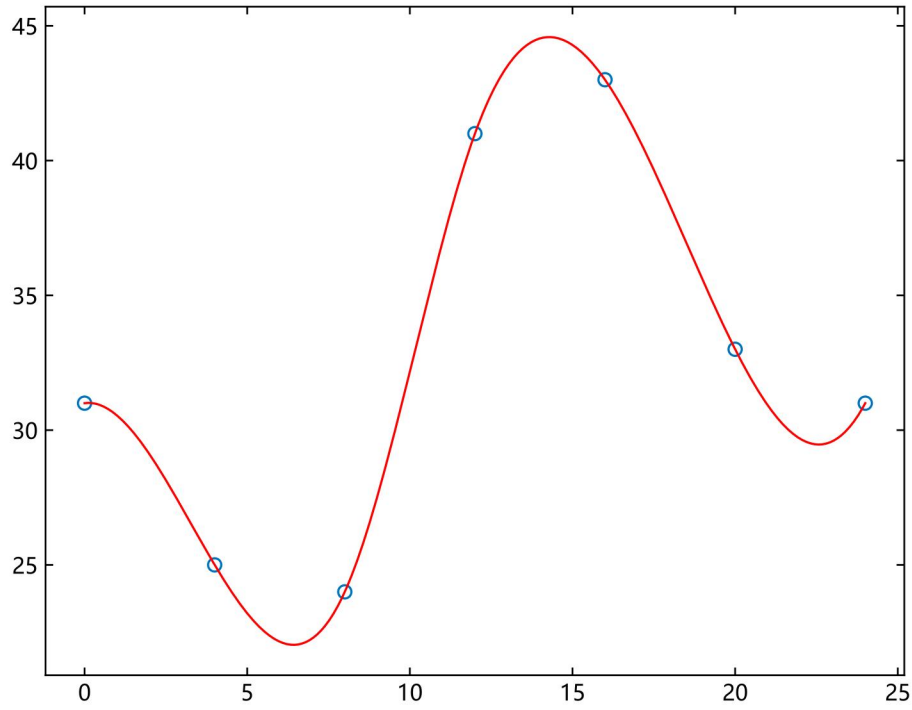


对数据集插值以预测一天中每一分钟内的温度读数。由于数据是周期性的，因此请使用 ″spline″ 插值方法。

```
xq = 0:1/60:24
v = interp1(x, T, xq, ″spline")
```

绘制插入的点。

```
hold(“on”)
plot(xq,V,”r”)
```



(5). 使用两种不同方法进行外插

定义样本点 x 及其对应样本值 v。

```
x =[1,2,3,4,5]
v =[12,16,31,10,6]
```

指定查询点 xq，这些查询点延伸到 x 的定义域以外。

```
xq = [0,0.5,1.5,5.5,6]
```

使用“pchip”方法计算 xq 处的 v。

```
vq1 = interp1(x,v,xq,"pchip")
```

```
vq1 = 5-element Vector[Float64}:
19.36842105263158
13.631578947368421
13.210526315789474
7.48
12.56
```

接着，使用“linear”方法和 NaN 参数计算 xq 处的 v。

```
vq2 = interp1(x,v,xq,"linear",NaN)
```

```
vq2 = 5-element Vector{Float64}:NaN
NaN
14.0
NaN
NaN
```

现在使用“linear”方法。

```
vq3 = interp1(x,v,xq,"linear")
```

```
vq3= 5-element Vector[Float64}:
8.0
10.0
14.0
4.0
2.0
```

此时结果内插采取“linear”方法，外插采取“pchip”方法。

(6). 为 x 域范围外的所有查询点指定常量值

定义样本点 x 及其对应样本值 v。

```
x =[-3,-2,-1,,1,2,3]
v = @.3*x^2
```

指定查询点 xq，这些查询点延伸到 x 的定义域以外。

```
xq =[-4,-2.5,-0.5,.5,2.5,4]
```

现在使用 pchip'方法计算 xg 处的 v，并为 x 域范围外的所有查询点赋予值 27。

```
vq = interp1(x,v,xq,"pchip",27)
```

```
vq = 6-element Vector[Float641:
27.0
18.65625
0.93750.9375
18.65625
27.0
```

(7). 在一个传递点插入多组数据

定义样本点。

```
x=-5:5
```

在 x 所定义的点处对三个不同的抛物线函数采样。

```
v1=@.X^2
v2 = @.2*x^2 + 2
v3 = @.3*x^2 +4
```

创建矩阵 v，其列为向量 v1、v2 和 v3。

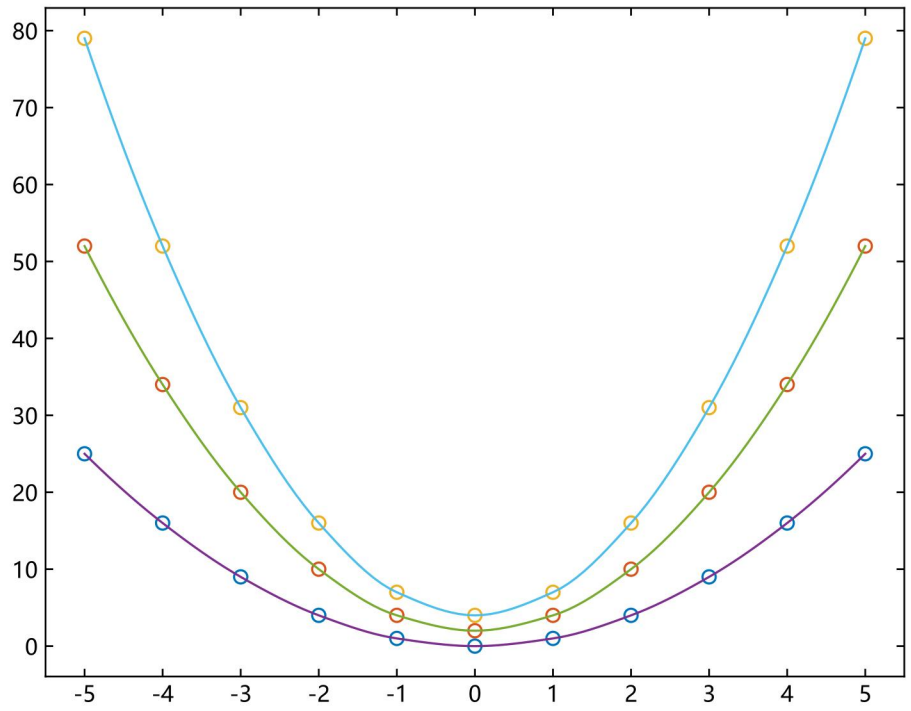
```
v=[v1 v2 v3]
```

将一组查询点 xq 定义为 x 范围内更精细的采样点。

```
xq = -5:0.1:5
```

计算 xq 处的全部三个函数，并绘制结果。

```
vq = interp1(x,v,xq,"pchip")
figure()
plot(x,v,"o",xq,vq)
xticks(-5:5)
```



绘图中的圆圈表示 v，实线表示 vq。

3.2.4.1.2 interp2

meshgrid 格式的二维网格数据插值

A. 语法

```
Vq = interp2(X,Y,V,Xq,Yq)
Vq = interp2(V,Xq,Yq)
Vq = interp2(V)
Vq = interp2(V,k)
Vq = interp2(___,method)
Vq = interp2(___,method,extrapval)
```

B. 说明

Vq = interp2(X,Y,V,Xq,Yq) 使用线性插值返回双变量函数在特定查询点的插入值。结果始终穿过函数的原始采样。X 和 Y 包含样本点的坐标。V 包含各

样本点处的对应函数值。Xq 和 Yq 包含查询点的坐标。

Vq = interp2(V,Xq,Yq) 假定一个默认的样本点网格。默认网格点覆盖矩形区域 X=1:n 和 Y=1:m，其中 m,n = size(V)。如果您希望节省内存且不在意点之间的绝对距离，则可使用此语法。

Vq = interp2(V) 将每个维度上样本值之间的间隔分割一次，形成细化网格，并基于该网格上返回插入值。

Vq = interp2(V,k) 将每个维度上样本值之间的间隔反复分割 k 次，形成细化网格，并基于该网格上返回插入值。这将在样本值之间生成 2^k-1 个插值点

Vq = interp2(___,method) 指定备选插值方法："linear"、"nearest"、"cubic"、"makima" 或 "spline"。默认方法为 "linear"。

Vq = interp2(___,method,extrapval) 还指定标量值或字符串 extrapval，如果该参数为标量值，则其会为处于样本点域范围外的所有查询点赋予该标量值，否则则采用该字符串指定的外插方法。

如果您为样本点域范围外的查询省略 extrapval 参数，则基于 method 参数，interp2 返回下列值之一：

对于 "cubic" 方法，返回 NaN 值。

对于其他内插方法，返回外插值。

C. 示例

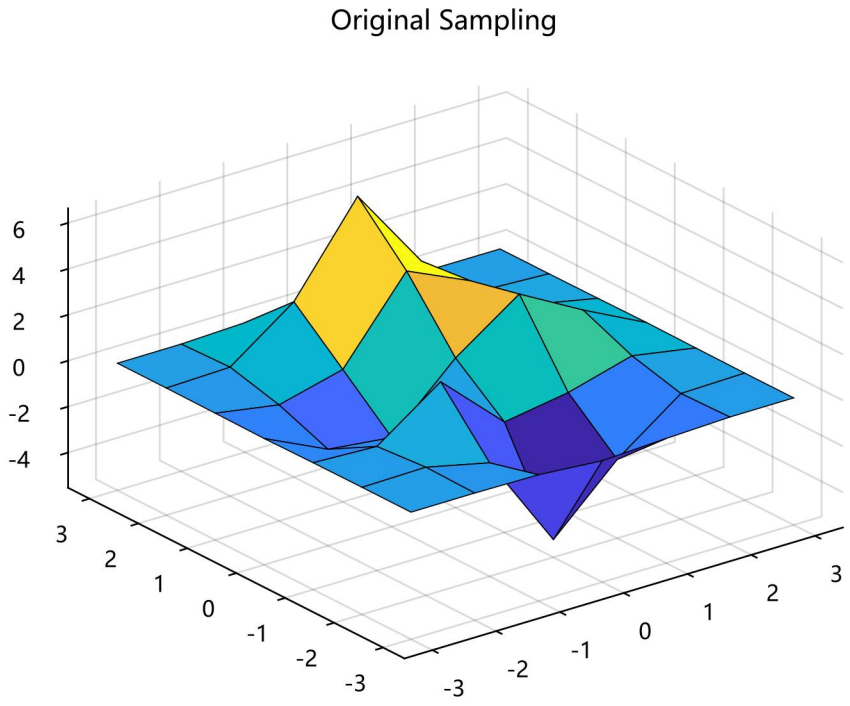
(1). 使用默认方法基于网格进行插值

对 peaks 函数进行粗略采样。

```
X,Y = meshgrid2(-3:3,-3:3)
V = peaks(X,Y)
```

绘制粗略采样。

```
figure()
surf(X,Y,V)
title("Original Sampling")
```



创建间距为 0.25 的查询网格。

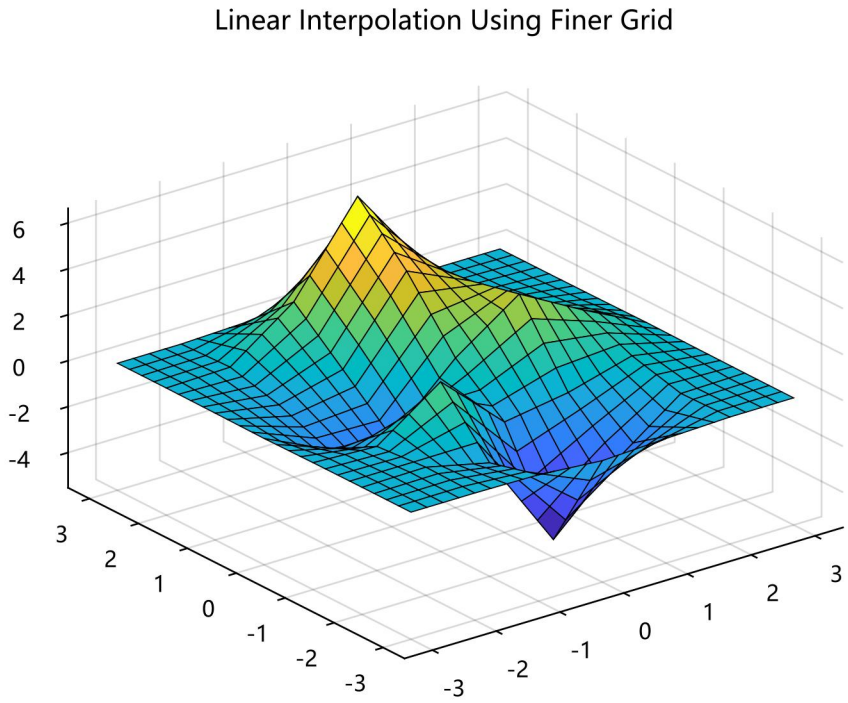
```
Xq,Yq = meshgrid2(-3:0.25:3,-3:0.25:3)
```

对查询点插值。

```
Vq = interp2(X,Y,V,Xq,Yq)
```

绘制结果。

```
figure()
surf(Xq,Yq,Vq)
title("Linear Interpolation Using Finer Grid")
```



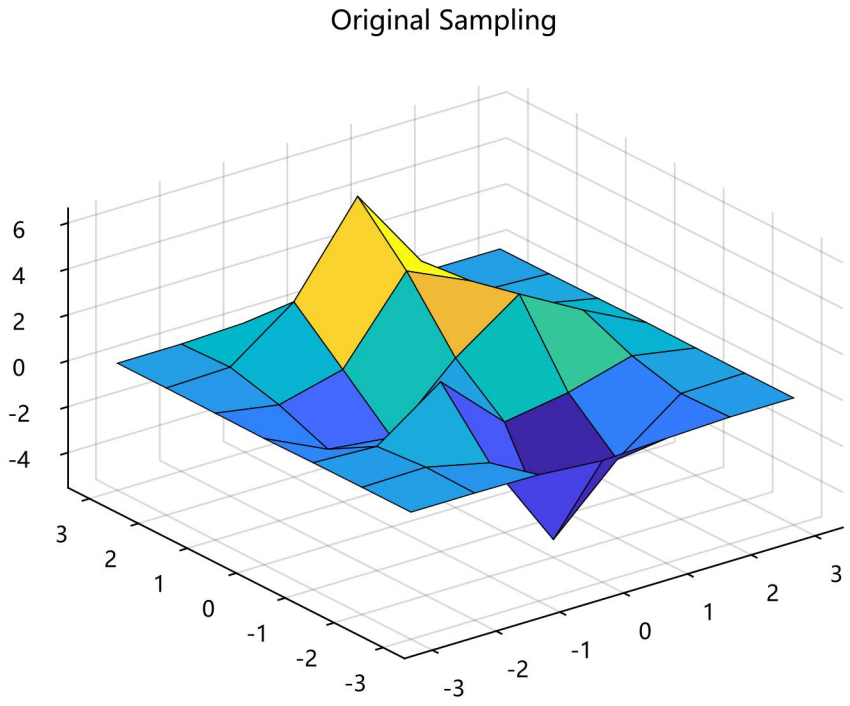
(2). 使用三次插值方法基于网格进行插值

对 peaks 函数进行粗略采样。

```
X,Y = meshgrid2(-3:3,-3:3)
V = peaks(7)[3]
```

绘制粗略采样。

```
figure()
surf(X,Y,V)
title("Original Sampling")
```



创建间距为 0.25 的查询网格。

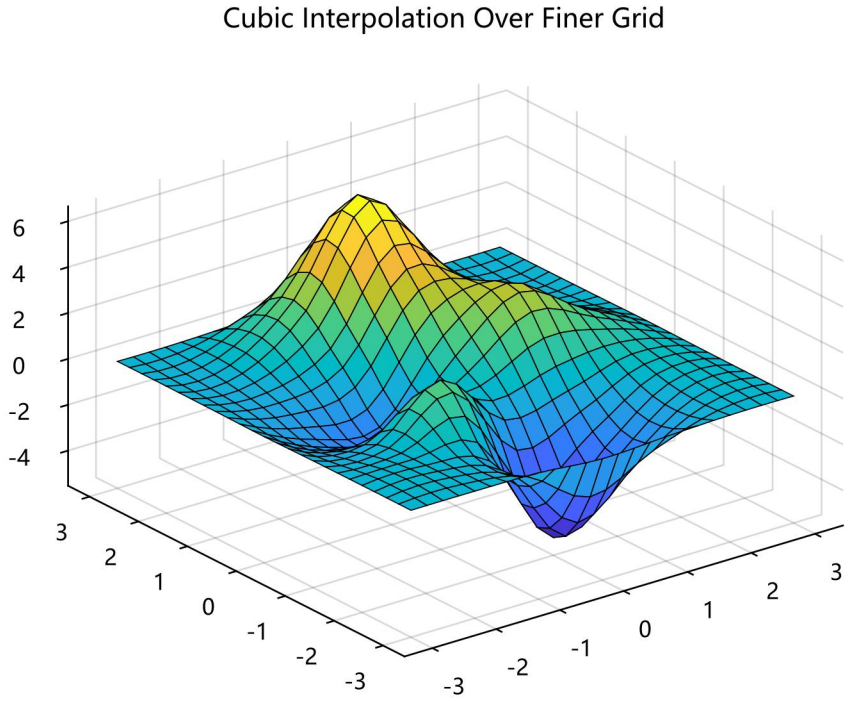
```
Xq,Yq = meshgrid2(-3:0.25:3,-3:0.25:3)
```

对查询点插值，并指定三次插值。

```
Vq = interp2(X,Y,V,Xq,Yq,"cubic")
```

绘制结果。

```
figure()
surf(Xq,Yq,Vq)
title("Cubic Interpolation Over Finer Grid")
```



(3). 优化灰度图像

导入 flujet.jl 文件，在工作区中加载一些图像数据。

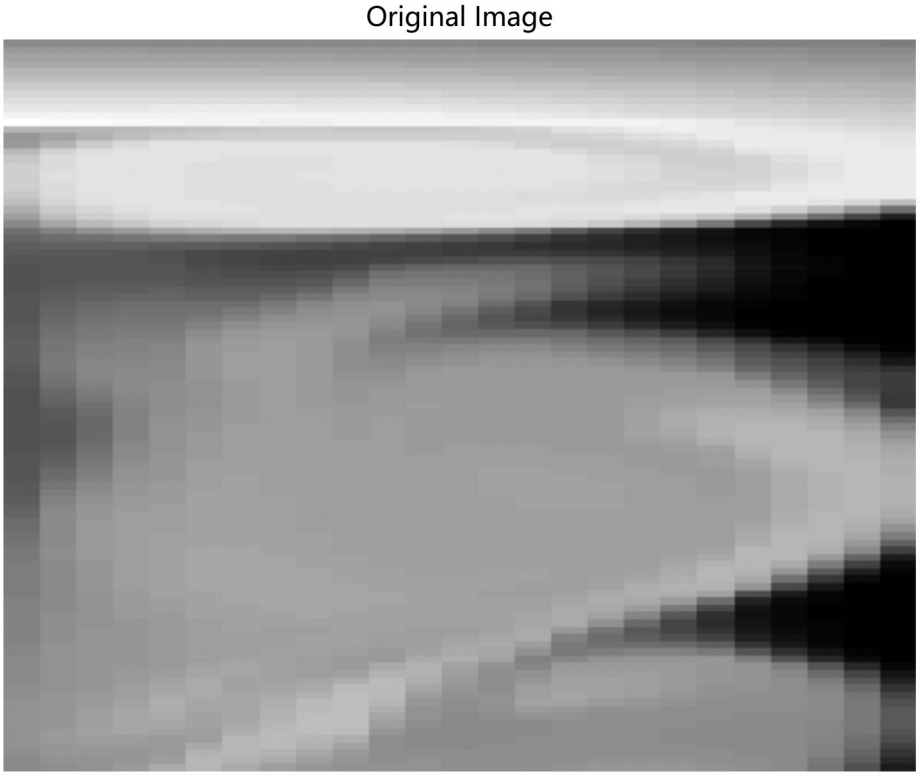
```
include(pkgdir(TyMath)*"/examples/docs/flujet.jl")
```

隔离图像的一个小区域。

```
V = X[200:300, 1:25]
```

显示该图像区域。

```
ims = imagesc(reverse(V, dims = 2), xvalue = [0 125], yvalue = [100
0])
colormap(ims, "gray")
axis("off")
title("Original Image")
```

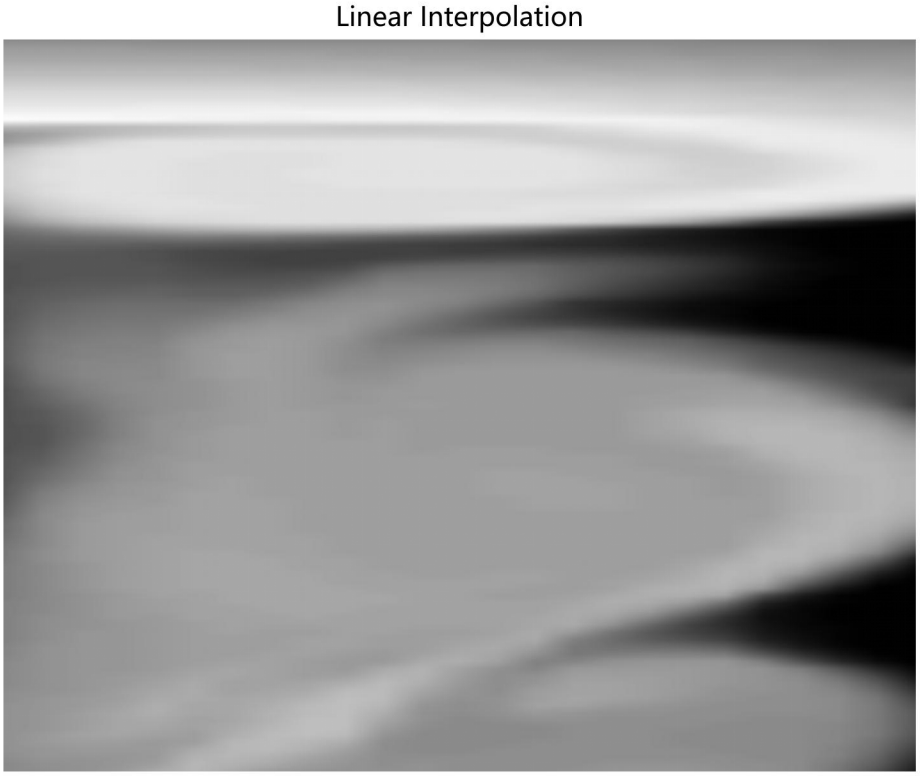


将每个维度上网格点之间的间隔重复分割细化五次来插入插值。

```
Vq = interp2(V,5)
```

显示结果。

```
ims = imagesc(reverse(Vq,dims=2), xvalue = [0 125], yvalue = [100
0])
colormap(ims, "gray")
axis("off")
title("Linear Interpolation")
```



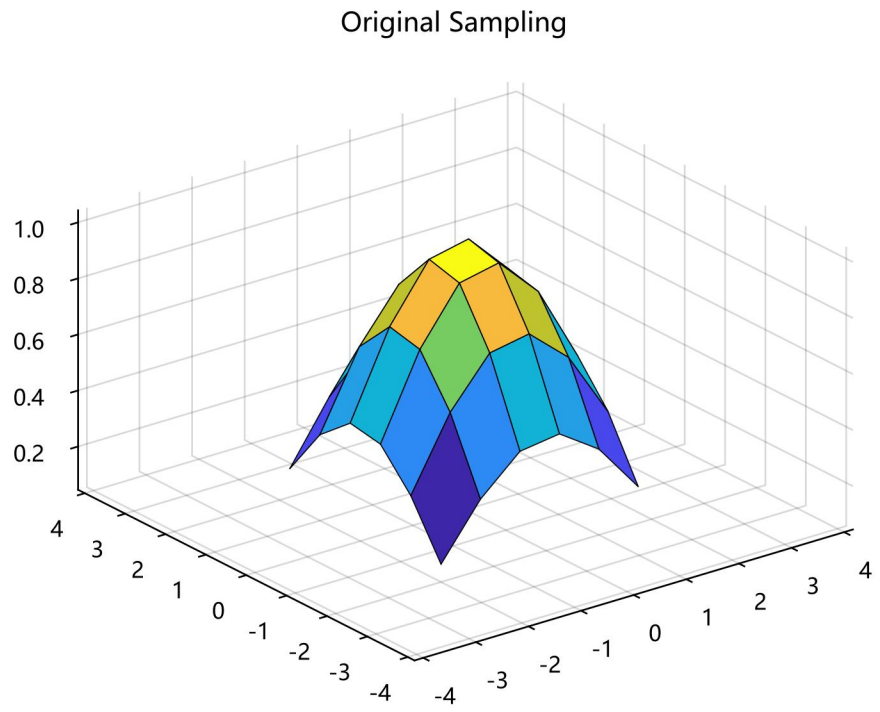
(4). 在 X 和 Y 域范围外计算

在 [-2, 2] 的范围内从两个维度对函数进行粗略采样。

```
X,Y = meshgrid2(-2:0.75:2,-2:0.75:2)
R = @. sqrt(X^2 + Y^2)+ $eps()
V = @. sin(R)/R
```

绘制粗略采样。

```
figure()
surf(X,Y,V)
xlim([-4 4])
ylim([-4 4])
title("Original Sampling")
```



在 X 和 Y 域范围以外延伸，创建查询网格。

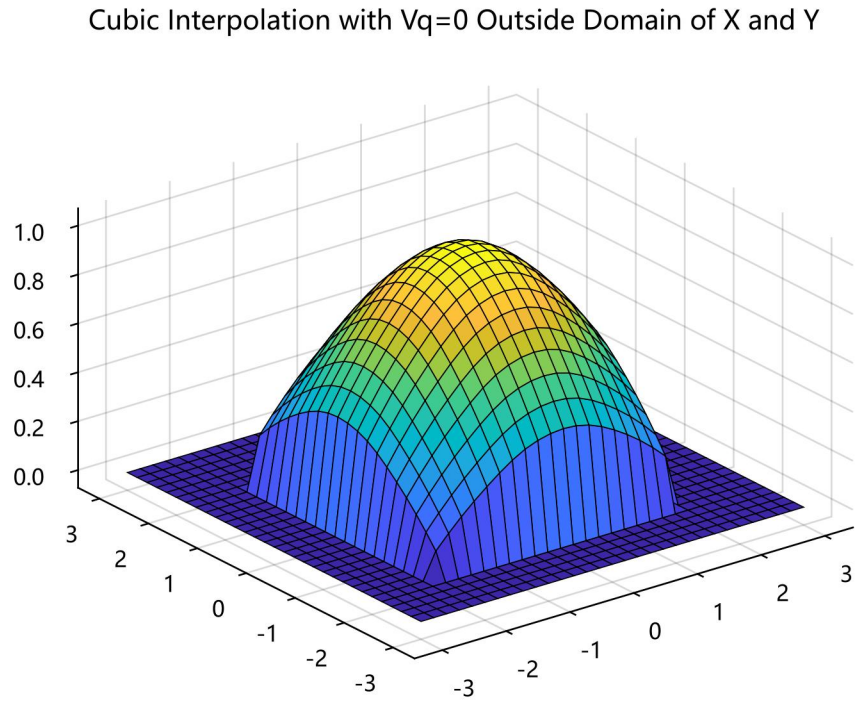
```
Xq,Yq = meshgrid2(-3:0.2:3,-3:0.2:3)
```

在 X 和 Y 域内进行三次插值，并对域外的所有查询赋零值。

```
Vq = interp2(X,Y,V,Xq,Yq,"cubic",0)
```

绘制结果。

```
figure()
surf(Xq,Yq,Vq)
title("Cubic Interpolation with Vq=0 Outside Domain of X and Y")
```



3.2.4.1.3 interp3

meshgrid 格式的三维网格数据的插值

A. 语法

```
Vq = interp3(X,Y,Z,V,Xq,Yq,Zq)
Vq = interp3(V,Xq,Yq,Zq)
Vq = interp3(V)
Vq = interp3(V,k)
Vq = interp3(___,method)
Vq = interp3(___,method,extrapval)
```

B. 说明

Vq = interp3(X,Y,Z,V,Xq,Yq,Zq) 使用线性插值返回三变量函数在特定查询点的插值。结果始终穿过函数的原始采样。X、Y 和 Z 包含样本点的坐标。V 包含各样本点处的对应函数值。Xq、Yq 和 Zq 包含查询点的坐标。

Vq = interp3(V,Xq,Yq,Zq) 假定一个默认的样本点网格。默认网格点覆盖区域 X=1:n、Y=1:m 和 Z=1:p，其中 m,n,p = size(V)。如果您希望节省内存且不在意点之间的绝对距离，则可使用此语法。

Vq = interp3(V) 将每个维度上样本值之间的间隔分割一次，形成细化网格，并基于该网格上返回插入值。

Vq = interp3(V,k) 将每个维度上样本值之间的间隔反复分割 k 次，形成细化网格，并基于该网格上返回插入值。这将在样本值之间生成 2^k-1 个插值点。

Vq = interp3(___,method) 指定备选插值方法：'linear'、'nearest'、'cubic'、'makima' 或 'spline'。默认方法为 'linear'。

Vq = interp3(___,method,extrapval) 还指定标量值或字符串 extrapval，如果此参数为标量值，则会为处于样本点域范围外的所有查询点赋予该标量值，否则则使用字符串指定的外插方法。

如果您为样本点域范围外的查询省略 extrapval 参数，则基于 method 参数，interp3 返回下列值之一：

对于 'spline' 和 'makima' 方法，返回外插值

对于其他内插方法，返回 NaN 值

C. 示例

(1). 使用默认方法进行插值

加载 flow 函数的点和值，每个维度采样 10 个点。

```
X,Y,Z,V = flow(10)
```

flow 函数通过数组 X、Y 和 Z 返回网格。该网格覆盖区域 0.1≤X≤10、−3≤Y≤3、−3≤Z≤3，间距为 ΔX=0.5、ΔY=0.7 和 ΔZ=0.7。

创建间距为 0.25 的查询网格。

```
Xq,Yq,Zq = meshgrid3(.1:.25:10,-3:.25:3,-3:.25:3)
```

对查询网格中的点插值。

```
Vq = interp3(X,Y,Z,V,Xq,Yq,Zq)
```

(2). 使用三次插值方法进行插值

加载 flow 函数的点和值，每个维度采样 10 个点。

```
X,Y,Z,V = flow(10)
```

flow 函数通过数组 X、Y 和 Z 返回网格。该网格覆盖区域 0.1≤X≤10、−3≤Y≤3、−3≤Z≤3，间距为 ΔX=0.5、ΔY=0.7 和 ΔZ=0.7。

创建间距为 0.25 的查询网格。

```
Xq,Yq,Zq = meshgrid3(.1:.25:10,-3:.25:3,-3:.25:3)
```

使用 'cubic' 插值方法在查询网格点处插值。

```
Vq = interp3(X,Y,Z,V,Xq,Yq,Zq,"cubic")
```

(3). 在 X,Y 和 Z 域范围外执行计算

创建网格向量 x、y 和 z。这些向量用于定义与 V 中的值相关联的点。

```
x = 1:100
y = 1:50
z = 1:30
```

将样本值定义为一个 50×100×30 的随机数数组 V。使用 rand 函数创建数组。

```
rng = MersenneTwister(1234)
V = rand(rng, 50,100,30)
```

计算 x、y 和 z 域范围外三个点处的 V。指定 extrapval = -1。

```
xq = [0 0 0]
yq = [0 0 51]
zq = [0 101 102]
vq = interp3(x,y,z,V,xq,yq,zq,"linear",-1)
```

```
vq = 1×3 Matrix{Float64}:
-1.0  -1.0  -1.0
```

三个点计算得到的值均为 -1，因为它们位于 x、y 和 z 域之外。

3.2.4.1.4 interpn

ndgrid 格式的一维、二维、三维和 N 维网格数据的插值

A. 语法

```
Vq = interpn(X1,X2,...,Xn,V,Xq1,Xq2,...,Xqn)
Vq = interpn(V,Xq1,Xq2,...,Xqn)
Vq = interpn(V)
Vq = interpn(V,k)
Vq = interpn(___,method)
Vq = interpn(___,method,extrapval)
```

B. 说明

Vq = interpn(X1,X2,...,Xn,V,Xq1,Xq2,...,Xqn) 使用线性插值返回 n 变量函数在特定查询点的插入值。结果始终穿过函数的原始采样。X1,X2,...,Xn 包含样本点的坐标。V 包含各样本点处的对应函数值。Xq1,Xq2,...,Xqn 包含查询点的坐标。

Vq = interpn(V,Xq1,Xq2,...,Xqn) 假定一个默认的样本点网格。默认网格的每个维度均包含点 1,2,3,...ni。ni 的值为 V 中第 i 个维度的长度。如果您希望

节省内存且不在意点之间的绝对距离，则可使用此语法。

Vq = interpn(V) 将每个维度上样本值之间的间隔分割一次，形成细化网格，并基于该网格上返回插入值。

Vq = interpn(V,k) 将每个维度上样本值之间的间隔反复分割 k 次，形成细化网格，并基于该网格上返回插入值。这将在样本值之间生成 2^k-1 个插值点。

Vq = interpn(___,method) 指定备选插值方法：'linear'、'nearest'、'pchip'、'cubic'、'makima' 或 'spline'。默认方法为 'linear'。示例

Vq = interpn(___,method,extrapval) 还指定标量值或字符串 extrapval，如果该参数为标量值，则其会为处于样本点域范围外的所有查询点赋予该标量值，否则则采用该字符串指定的外插方法。

如果您为样本点域范围外的查询省略 extrapval 参数，则基于 method 参数，interpn 返回下列值之一：

对于 "spline" 和 "makima" 方法，返回外插值

对于其他内插方法，返回 NaN 值

C. 示例

(1). 一维插值

定义样本点和值。

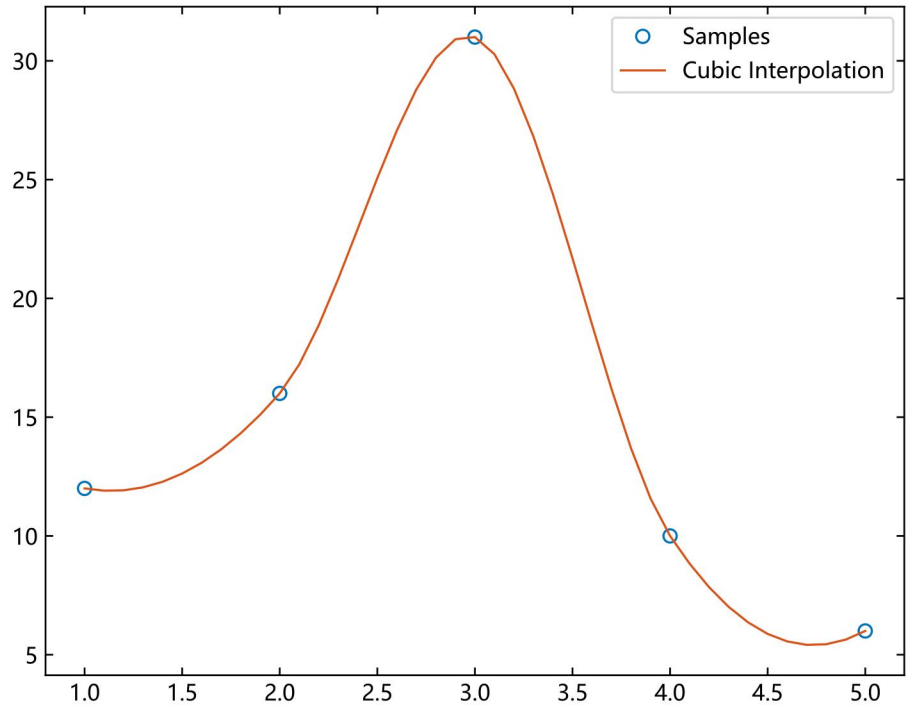
```
x = [1,2,3,4,5]
v = [12,16,31,10,6]
```

定义查询点 xq 并插值。

```
xq = 1:0.1:5
vq = interpn(x,v,xq,"cubic")
```

绘制结果。

```
figure()
plot(x,v,"o",xq,vq,"-")
legend(["Samples","Cubic Interpolation"])
```



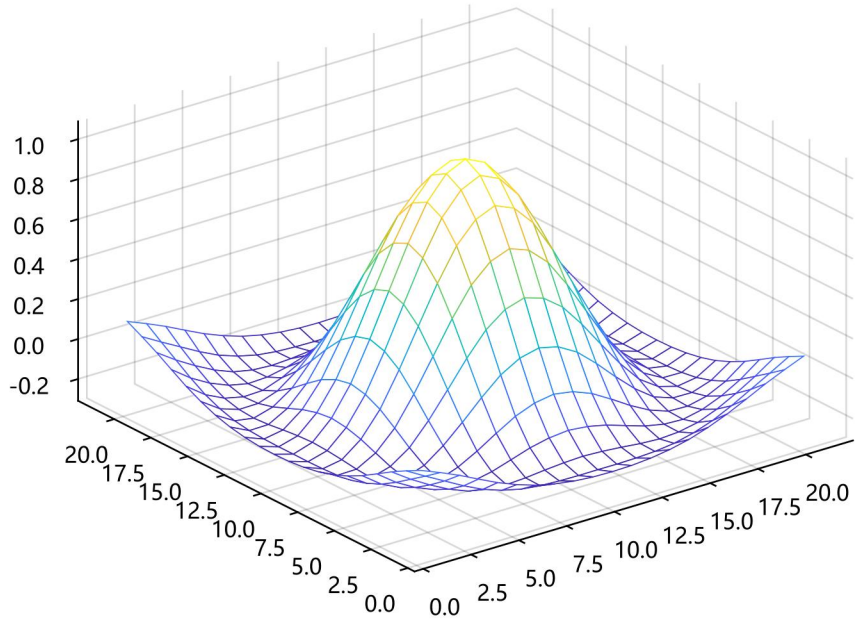
(2). 二维插值

创建一组一维网格点和对应的样本值。

```
X1,X2 = ndgrid(-5:1:5,dims=2)
R = @. sqrt(X1^2 + X2^2)+ eps()
V = @. sin(R)/(R)
```

使用 ntimes=1 在更精细的网格上插值。

```
Vq = interpn(V,"cubic")
mesh(Vq)
```



(3). 在三维函数域范围外执行计算

创建网格向量 x1、x2 和 x3。这些向量用于定义与 V 中的值关联的点。

```
x1 = 1:100
x2 = (1:50)'
x3 = 1:30
```

将样本值定义为一个 100×50×30 的随机数数组 V。

```
rng = MersenneTwister(1234)
V = rand(rng,100,50,30)
```

计算 x1、x2 和 x3 域范围外三个点处的 V。指定 extrapval = -1。

```
xq1 = [0 0 0]
xq2 = [0 0 51]
xq3 = [0 101 102]
vq = interpn(x1,x2,x3,V,xq1,xq2,xq3,"linear",-1)
```

```
vq = 1×3 Matrix{Float64}:
 -1.0  -1.0  -1.0
```

三个点计算得到的值均为 -1，因为它们位于 x1、x2 和 x3 域之外。

(4). 四维插值

定义表示 $f = te^{-x^2-y^2-z^2}$ 的函数句柄。

```
f(x,y,z,t) = t*exp(-x^2-y^2-z^2)
```

在 中创建网格点。然后，将这些点全部传递给函数，以创建样本值 V。

```
x,y,z,t = ndgrid(-1:0.2:1,-1:0.2:1,-1:0.2:1,0:2:10);
V = f.(x,y,z,t)
```

现在创建查询网格。

```
xq,yq,zq,tq = ndgrid(-1:0.05:1,-1:0.08:1,-1:0.05:1,0:0.5:10)
```

在查询点处使用 V 进行插值。

```
Vq = interpn(x,y,z,t,V,xq,yq,zq,tq)
```

3.2.4.1.5 griddedInterpolant

网格数据插值

A. 说明

使用 griddedInterpolant 对一维、二维、三维或 N 维网格数据集进行插值。griddedInterpolant 返回给定数据集的插值 F。您可以计算一组查询点（例如二维 (xq,yq)）处的 F 值，以得出插入的值 vq = F(xq,yq)。

使用 scatteredInterpolant 执行散点数据插值。

B. 创建对象

(1). 语法

```
F = griddedInterpolant()
F = griddedInterpolant(x,v)
F = griddedInterpolant(X1,X2,...,Xn,V)
F = griddedInterpolant(V)
F = griddedInterpolant(gridVecs,V)
F = griddedInterpolant(___,Method)
F = griddedInterpolant(___,Method,ExtrapolationMethod)
```

(2). 说明

F = griddedInterpolant() 创建一个空的网格数据插值对象。

F = griddedInterpolant(x,v) 根据样本点向量 x 和对应的值 v 创建一维插值。

F = griddedInterpolant(X1,X2,...,Xn,V) 使用作为一组 n 维数组 X1,X2,...,Xn 传递的样本点的完整网格创建二维、三维或 N 维插值。V 数组包含与 X1,X2,...,Xn 中的点位置关联的样本值。每个数组 X1,X2,...,Xn 的大小都必须与 V 相同。

F = griddedInterpolant(V) 使用默认网格创建插值。使用此语法时，

griddedInterpolant 将网格定义为第 i 维上间距为 1 且范围为 [1, size(V,i)] 的点集。如果您希望节省内存且不在意点之间的绝对距离，则可使用此语法。

F = griddedInterpolant(gridVecs,V) 指定一个数组 gridVecs，它包含 n 个网格向量，描述一个 n 维样本点网格。在您要使用特定网格而且希望节省内存时可使用此语法。

F = griddedInterpolant(___,Method) 指定插值方法："linear"、"nearest"、"next"、"previous"、"pchip"、"cubic"、"makima" 或 "spline"。您可以在上述任意语法中指定 Method 作为最后一个输入参数。

F = griddedInterpolant(___,Method,ExtrapolationMethod) 指定内插和外插方法。当您的查询点位于样本点域之外时，griddedInterpolant 使用 ExtrapolationMethod 估计值。

C. 示例

(1). 一维插值

使用 griddedInterpolant 进行一维数据集插值。

创建一个由散点样本点组成的向量 v。这些点是在 0 到 20 之间的随机一维位置进行采样的点。

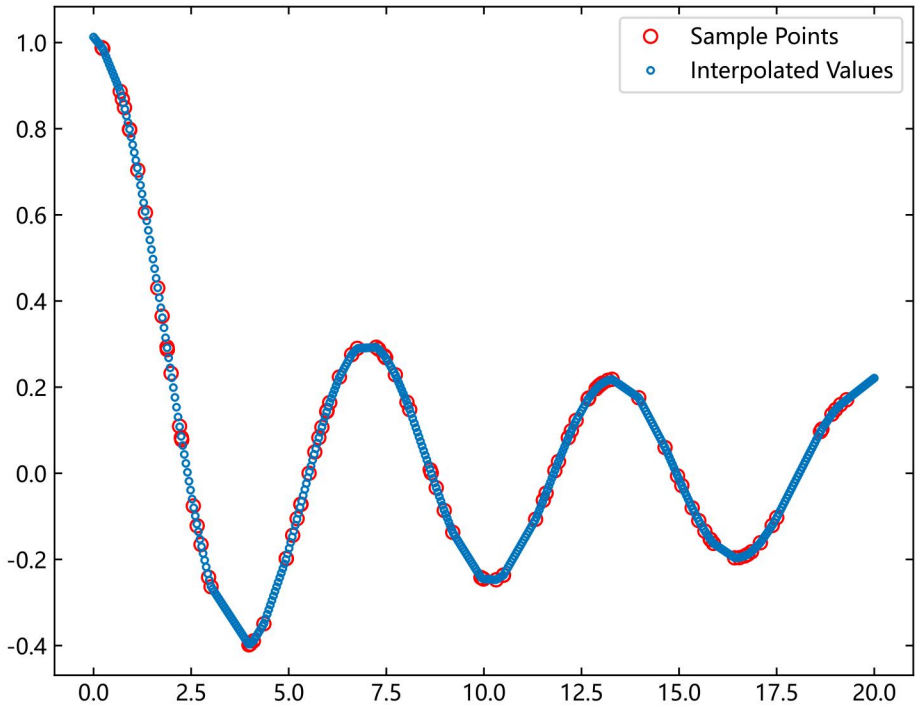
```
rng = MersenneTwister(1234)
x = sort(20 .*rand(rng,100))
v = besselj.(0,x)
```

为数据创建网格插值对象。默认情况下，griddedInterpolant 使用 "linear" 插值方法。

```
F = griddedInterpolant(x,v)
```

查询 0 到 20 之间 500 个均匀间隔点处的插值 F。将插值结果 (xq,vq) 绘制在原始数据 (x,v) 之上。

```
xq = LinRange(0,20,500)
vq = F(xq)
plot(x,v,"ro")
hold("on")plot(xq,vq,".")
legend(["Sample Points","Interpolated Values"])
```

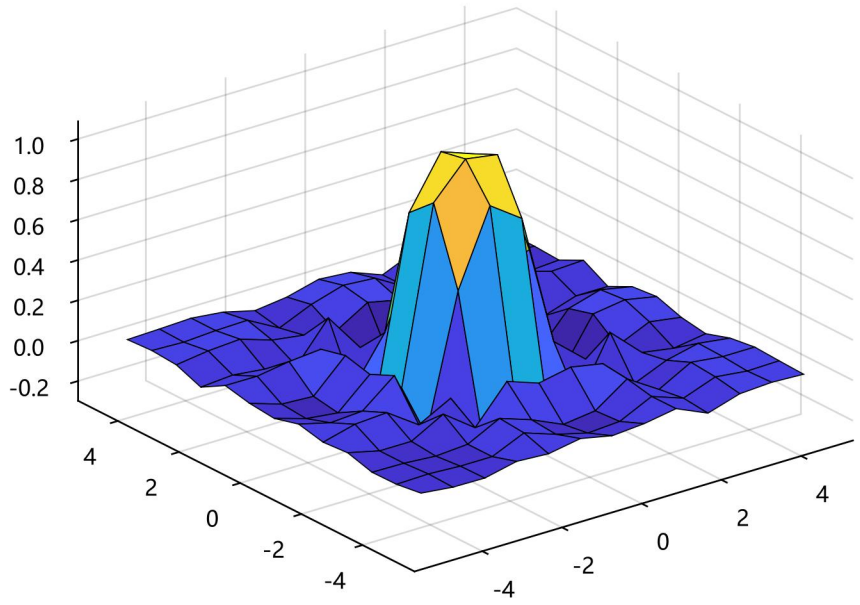


(2). 在不指定样本点的情况下进行插值

使用这两种方法对三维数据插值以指定查询点。

创建并绘制一个三维数据集，表示函数 在 [-5,5] 范围内的一组网格样本点处计算的结果。

```
x,y = ndgrid(-5:0.8:5,dims=2)
z = @. sin(x^2 + y^2) / (x^2 + y^2)
surf(x,y,z)
```

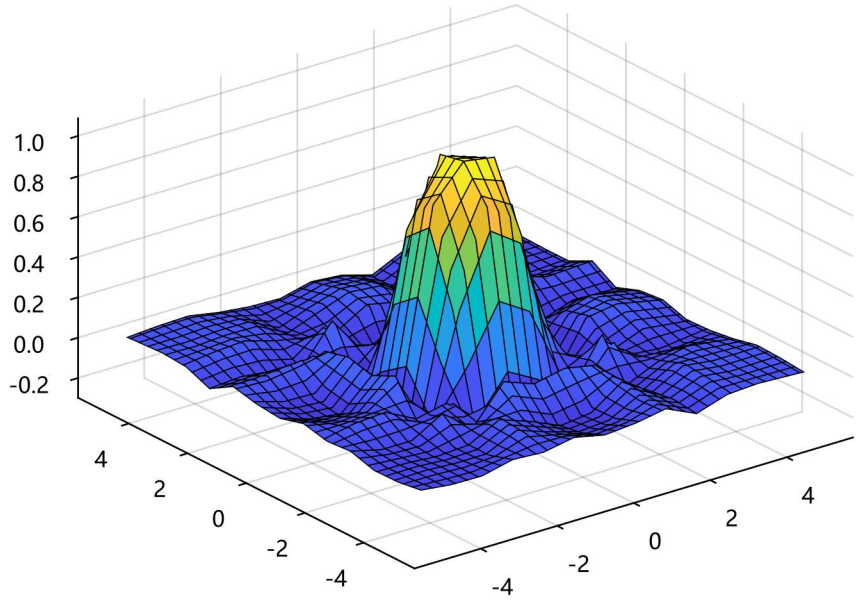


为数据创建网格插值对象。

```
F = griddedInterpolant(x,y,z)
```

使用更精细的网格查询插值并提高分辨率。

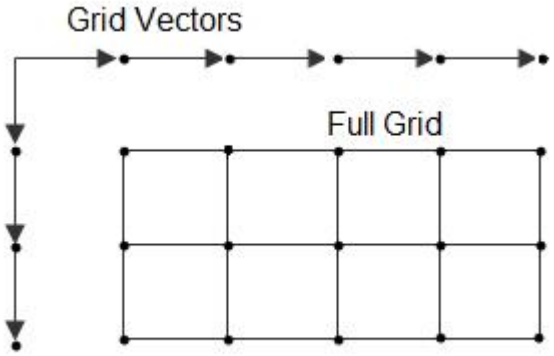
```
xq,yq = ndgrid(-5:0.1:5,dims=2)
vq = F(xq,yq)
surf(xq,yq,vq)
```



如果存在大量样本点或查询点，或者担心内存使用量太大，您可以使用网格向量来提高内存使用率。

如果您指定网格向量而不是使用 ndgrid 来创建完整网格，则 griddedInterpolant 可以避免为执行计算而生成完整的查询网格。

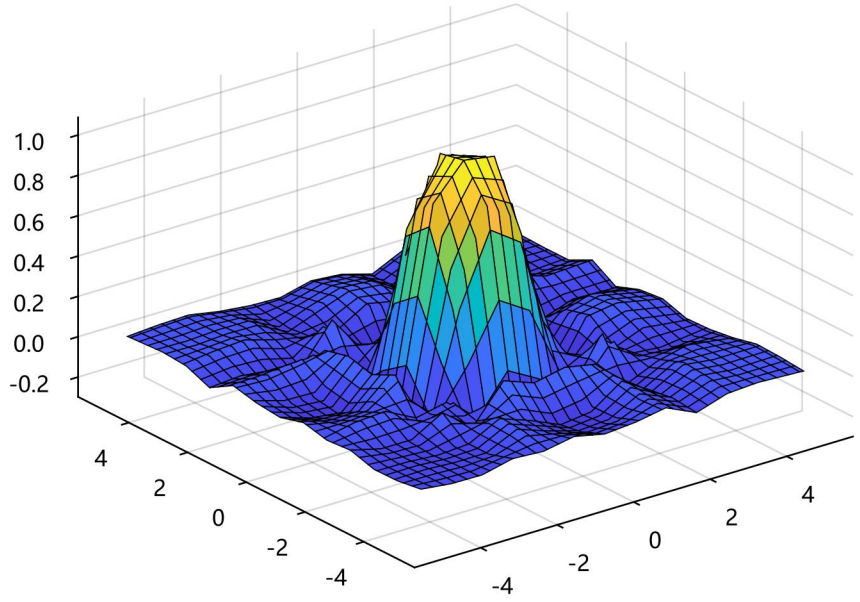
传递网格向量时，它们通常一组为单位，就像元组 (xg1, xg2, ..., xgn) 中的元一样。网格向量是表示完整网格数据点的紧凑方式。



也可以使用网格向量执行上述命令。

```
x = -5:0.8:5
z = zeros(length(x),length(x))
for i in CartesianIndices(z)
```

```
    xp = x[i[1]]
    yp = x[i[2]]
   z[i] = sin(xp^2+yp^2)/(xp^2+yp^2)
end
F = griddedInterpolant((x,x),z)
xq = -5:0.1:5
vq = F((xq,xq))
surf(xq,xq,vq)
```

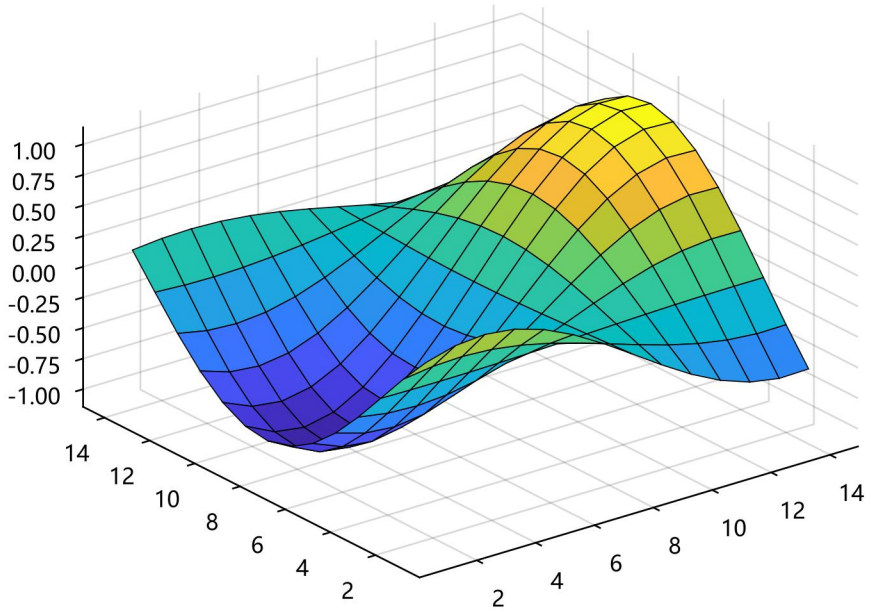


(3). 使用默认网格进行插值

使用默认网格对一组样本点执行快速插值。默认网格使用单一间距点，因此当样本点之间的精确 xy 间距不重要时，可以使用这种插值方法。

创建一个样本函数值矩阵，然后绘制它们对默认网格的图。

```
x = 1:0.3:5
n = length(x)
V = zeros(n,n)
for i in CartesianIndices(V)
   V[i] = cos(x[i[1]])*sin(x[i[2]])
end
surf(1:n,1:n,V)
```

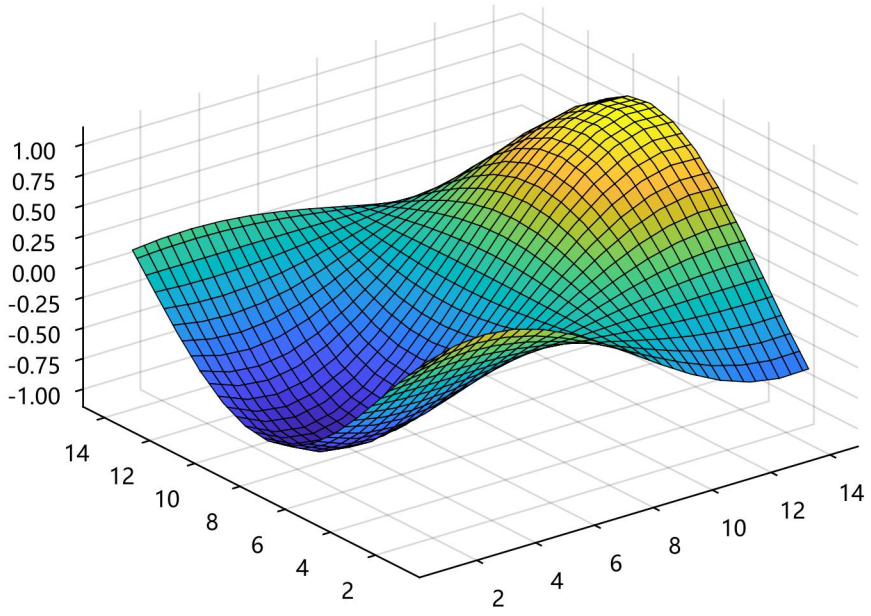


使用默认网格插入数据。

```
F = griddedInterpolant(V)
```

查询插值并绘制结果。

```
xq,yq = ndgrid(1:0.2:n,dims=2)
Vq = F(xq,yq)
surf(xq',yq',Vq)
```



(4). 更精细的网格上的二维插值

使用间距为 0.5 的完整网格进行粗略取样的数据插值。

在两个维度中以完整网格形式定义范围为 [1, 10] 的样本点。

```
X,Y = ndgrid(1:10,1:10)
```

在网格点处对 进行采样。

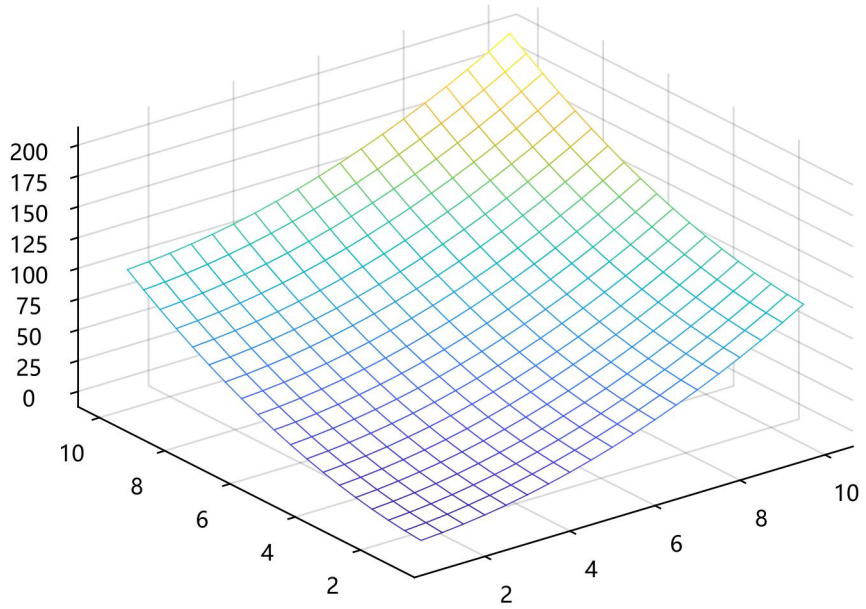
```
V = @. X^2 + Y^2
```

创建指定三次插值的插值。

```
F = griddedInterpolant(X,Y,V,"cubic")
```

使用 0.5 间距定义由查询点构成的完整网格并在这些点计算插值。然后，绘制结果。

```
Xq,Yq = ndgrid(1:0.5:10,1:0.5:10)
Vq = F(Xq,Yq)
mesh(Xq,Yq,Vq)
```



(5). 一维外插

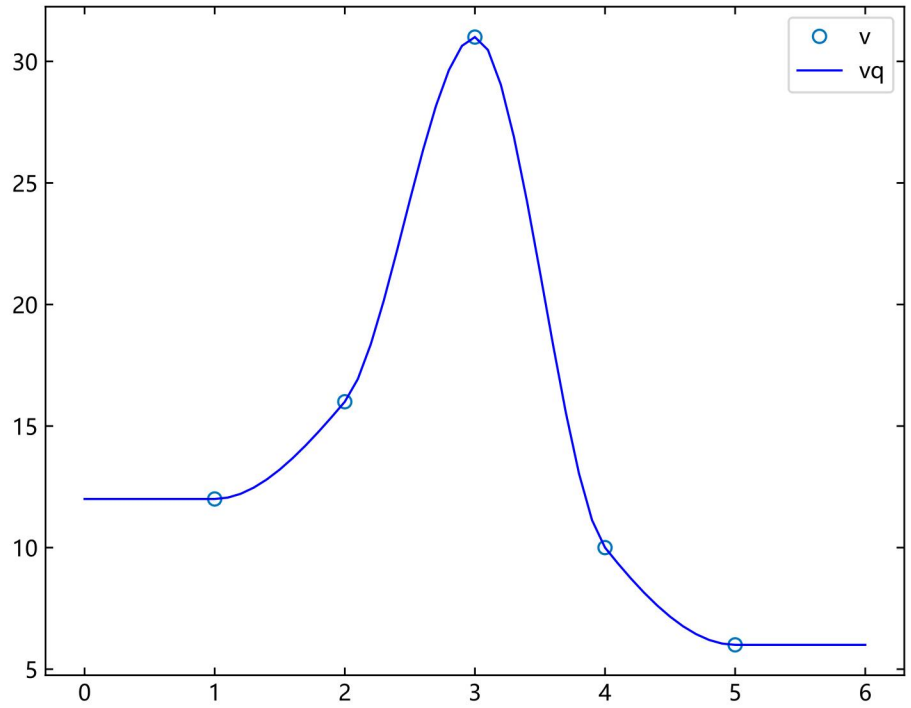
使用 "pchip" 和 "nearest" 外插方法比较查询 F 的域外插值的结果。

创建插值，并指定 "pchip" 作为内插方法，"nearest" 作为外插方法

```
x = [1,2,3,4,5]
v = [12,16,31,10,6]
F = griddedInterpolant(x,v,"pchip","nearest")
```

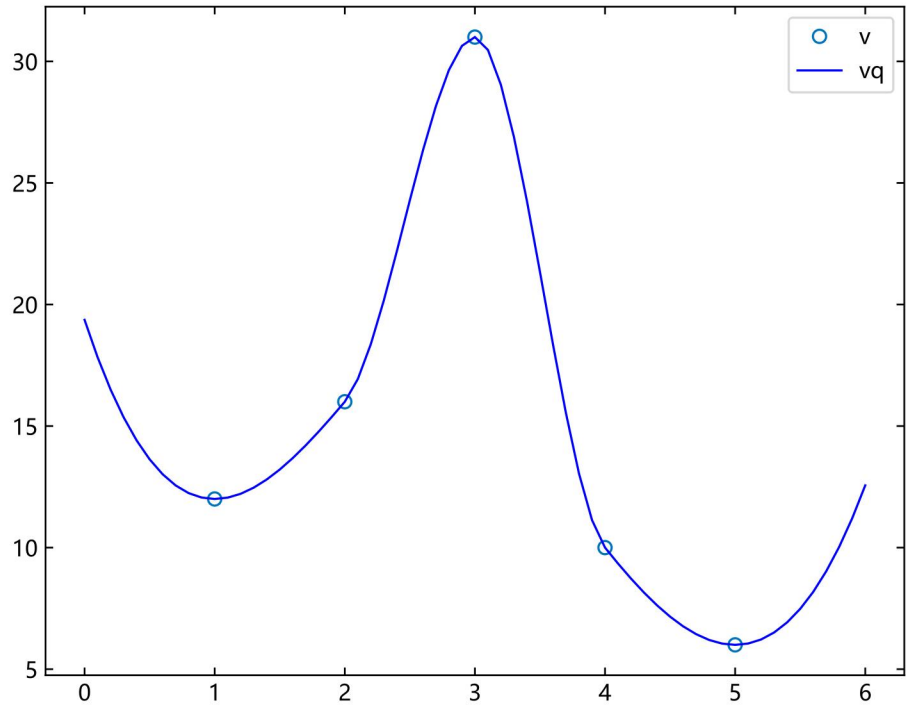
查询插值，并包括 F 的域外部的点。

```
xq = 0:0.1:6
vq = F(xq)
figure()
plot(x,v,"o",xq,vq,"-b")
legend(["v","vq"])
```



再次查询相同点的插值，这次使用 pchip 外插方法。

```
F.extrapolationMethod = "pchip"
figure()
vq = F(xq)
plot(x,v,"o",xq,vq,"-b")
legend(["v","vq"])
```



(6). 为 x 域范围外的所有查询点指定常量值

使用 griddedInterpolant 在相同的查询点上进行三组不同值插值。

使用 −5≤X≤5 和 −3≤Y≤3 创建一个采样点网格。

```
gx = -5:5
gy = -3:3
X,Y = ndgrid(gx,gy)
```

在查询点计算三个不同函数的值，然后将这些值串联成一个三维数组。V 的大小与前两个维度中的 X 和 Y 网格相同，但额外维度的大小反映与每个采样点相关联的值数目（在本例中为三个）。

```
f1 = @. X^2 + Y^2
f2 = @. X^3 + Y^3
f3 = @. X^4 + Y^4
V = cat(f1,f2,f3,dims=3)
```

使用采样点和相关联的值创建插值。

```
F = griddedInterpolant(X,Y,V)
```

创建一个查询点网格，其网格大小比采样点更细。

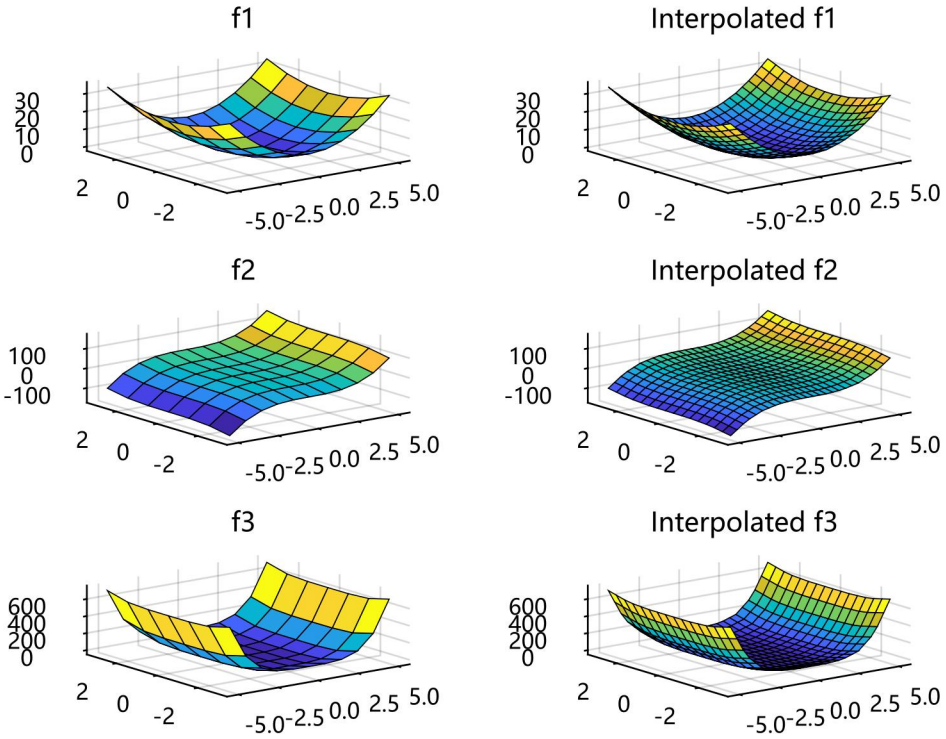
```
qx = -5:0.4:5
qy = -3:0.4:3
XQ,YQ = ndgrid(qx,qy)
```

在查询点对所有三组值进行插值。

```
VQ = F(XQ,YQ)
```

将原始数据与插值后的结果进行比较。

```
subplot(3, 2, 1)
surf(X, Y, f1)
title("f1")
subplot(3, 2, 2)
surf(XQ, YQ, VQ[:,:,1])
title("Interpolated f1")
subplot(3, 2, 3)
surf(X, Y, f2)
title("f2")
subplot(3, 2, 4)
surf(XQ, YQ, VQ[:,:,2])
title("Interpolated f2")
subplot(3, 2, 5)
surf(X, Y, f3)
title("f3")
subplot(3, 2, 6)
surf(XQ, YQ, VQ[:,:,3])
title("Interpolated f3")
tightlayout()
```



3.2.4.1.6 akima

Akima 分段三次 Hermite 插值

A. 语法

```
yq = akima(x,y,xq)
```

B. 说明

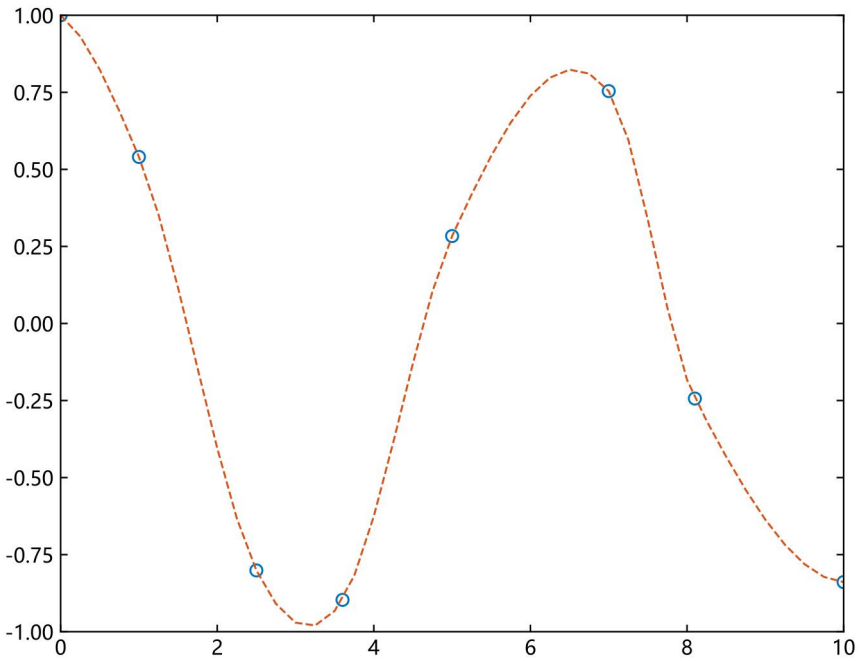
yq = akima(x,y,xq) 使用采样点 x 处的值 y 执行修正 Akima 插值，以求出查询点 xq 处的插值 yq。示例。

C. 示例

(1). 余弦数据的 Akima 插值

使用 akima 对非等距采样点上的余弦曲线进行插值。

```
x = [0 1 2.5 3.6 5 7 8.1 10];
y = cos.(x);
xq = 0:0.25:10;
yq = akima(x, y, xq);
plot(x, y, "o", xq, yq, "--")
```

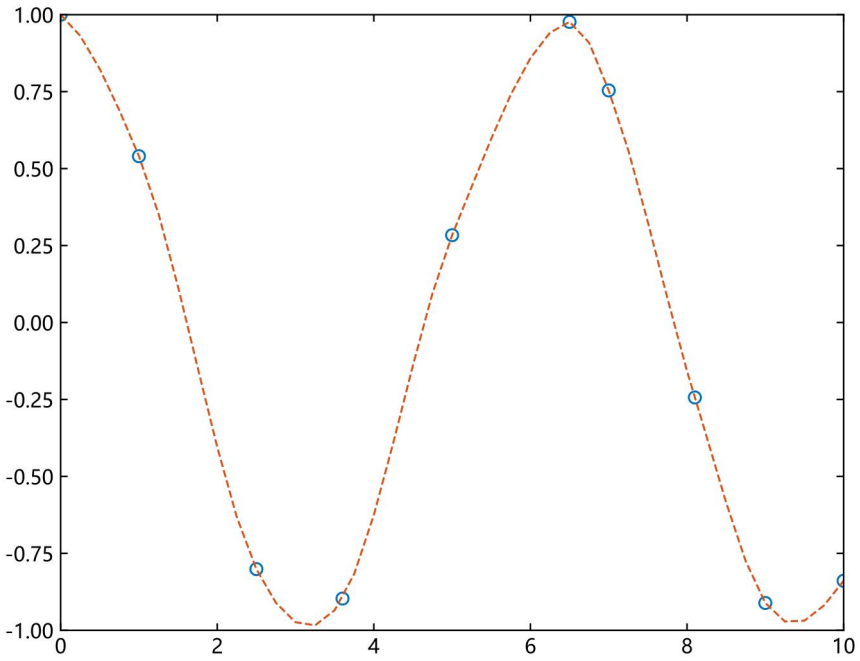


Akima 算法利用振荡函数使局部极值附近的曲线平坦化。为了补偿这种平

坦化，您可以在局部极值附近添加更多采样点。

在 x=6.5 和 x=9 附近添加采样点，然后重新绘制插值。

```
x = [0 1 2.5 3.6 5 6.5 7 8.1 9 10];
y = cos.(x);
xq = 0:0.25:10;
yq = akima(x, y, xq);
plot(x, y, "o", xq, yq, "--")
```



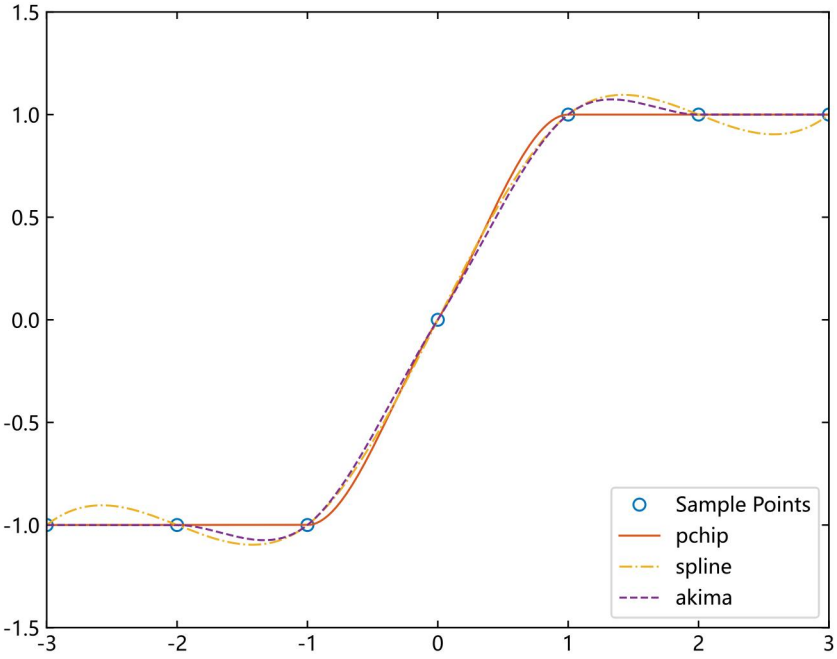
(2). 使用 spline、pchip 和 akima 的数据插值

将 spline、pchip 和 akima 为两个不同数据集生成的插值结果进行比较。这些函数都执行不同形式的分段三次 Hermite 插值。每个函数计算插值斜率的方式不同，因此它们在基础数据的平台区或波动处展现出不同行为。

对连接两个平台区的样本数据进行插值，并比较结果。创建由 x 值、点 y 处的函数值以及查询点 xq 组成的向量。使用 spline、pchip 和 akima 计算查询点处的插值。绘制查询点处的插值函数值以进行比较。

```
x = -3:3;
y = [-1 -1 -1 0 1 1 1];
xq1 = -3:0.01:3;
p = pchip(x, y, xq1);
s = spline(x, y, xq1);
m = akima(x, y, xq1);
plot(x, y, "o", xq1, p, "-", xq1, s, "-.", xq1, m, "--")
```

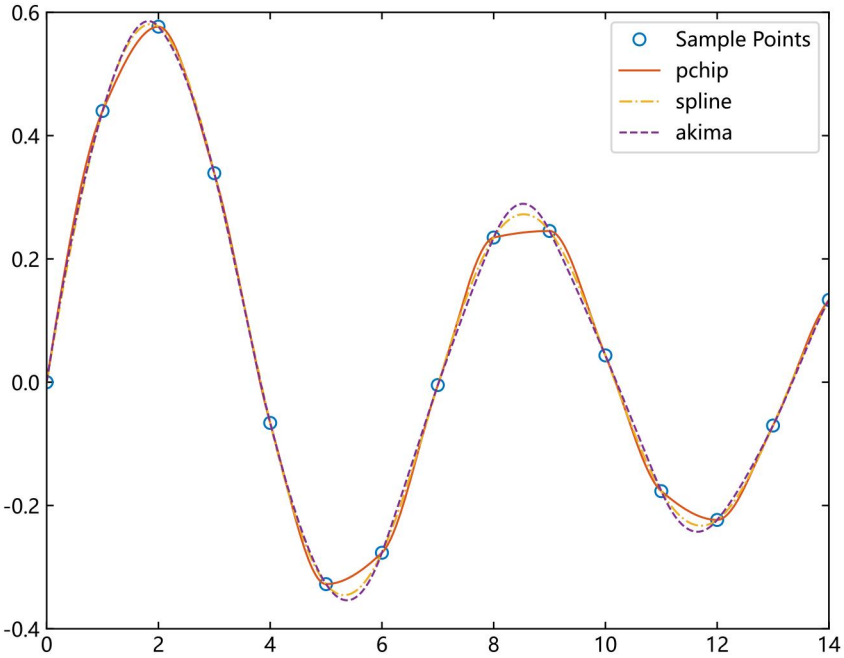
```
legend(["Sample Points", "pchip", "spline", "akima"], loc = "southeast")
```



在本例中，pchip 和 akima 具有相似的行为，因为它们可以避免过冲，并且可以准确地连接平台区。

使用振动采样函数执行第二次比较。

```
x = 0:15;
y = besselj.(1, x);
xq2 = 0:0.01:15;
p = pchip(x, y, xq2);
s = spline(x, y, xq2);
m = akima(x, y, xq2);
plot(x, y, "o", xq2, p, "-", xq2, s, "-.", xq2, m, "--")
legend(["Sample Points", "pchip", "spline", "akima"])
```



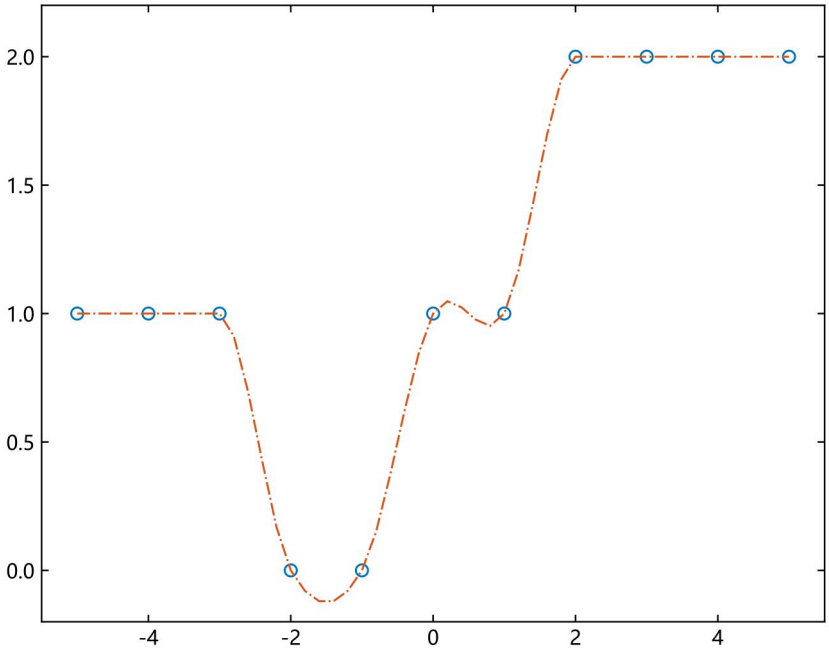
当基础函数振荡时，spline 和 akima 能够更好地捕捉点之间的波动，而 pchip 会在局部极值附近急剧平坦化。

(3). 使用分段多项式结构体进行 Akima 插值

创建采样点 x 及其值 y 的向量。

绘制结果。在有三个或更多连续等值点的区域，Akima 算法用直线将这些点连接起来。

```
x = [-5:5...];
y = vec([1 1 1 0 0 1 1 2 2 2 2])
xq = -5:0.2:5;
m = akima([x...], vec(y), xq)
plot(x, y, "o", xq, m, "-.")
ylim(-0.2,2.2)
```



3.2.4.1.7 interpft

一维插值（FFT 方法）

A. 语法

```
y = interpft(X,n)
```

B. 说明

y = interpft(X,n) 在 X 中内插函数值的傅里叶变换以生成 n 个等间距的点。interpft 对第一个大小不等于 1 的维度进行运算。

y = interpft(X,n,dim) 沿维度 dim 运算。例如，如果 X 是矩阵，interpft(X,n,2) 将在 X 行上进行运算。。

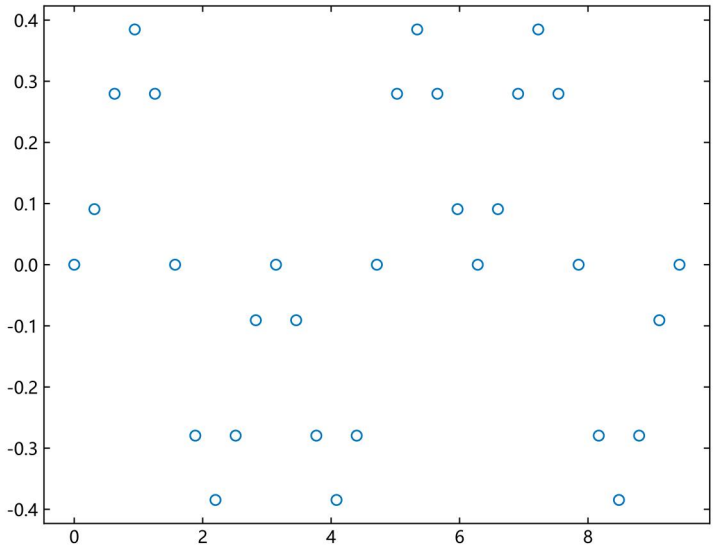
C. 示例

(1). 傅里叶插值

使用 FFT 方法对一维数据插值，并以可视方式呈现结果。

在区间 [0,3 π] 内为函数 $f(x) = \sin^2(x)\cos(x)$ 生成一些样本点。使用间距 dx 以确保数据均匀分布。绘制样本点。

```
dx = 3*pi/30
x = 0:dx:3*pi
f = @. sin(x)^2 * cos(x)
plot(x,f,"o")
```



使用 FFT 插值，以查找 200 个查询点处的函数值。

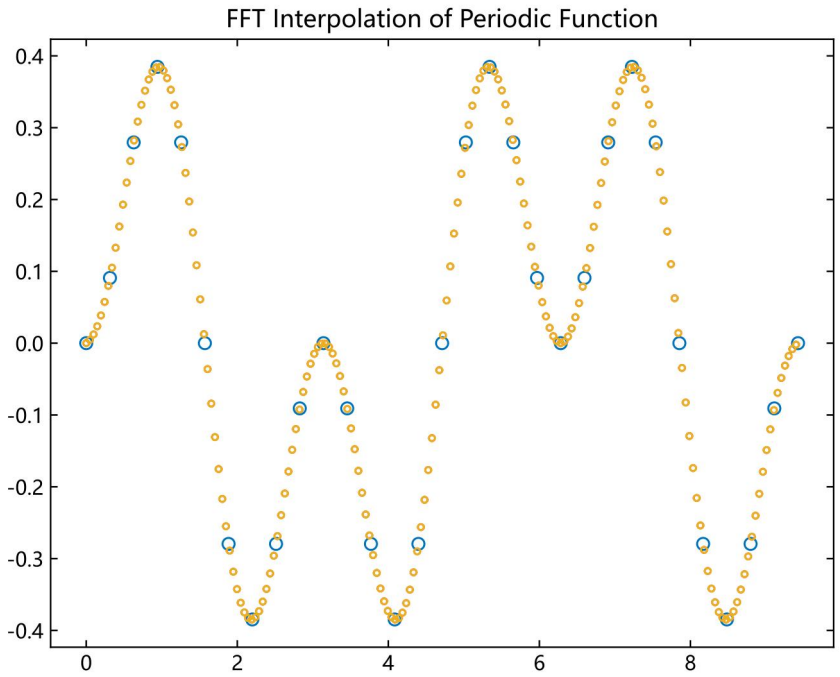
```
N = 200
y = interpft(f,N)
```

使用 dy = dx*length(x)/N 基于样本点的间距计算插值数据的间距，其中 N 是插值点的数目。截断 y 中的数据以匹配 x2 的采样密度。

```
dy = dx*length(x)/N
x2 = 0:dy:3*pi
y = y[1:length(x2)]
```

绘制结果。

```
hold("on")
plot(x2,y,".")
title("FFT Interpolation of Periodic Function")
```



(2). 对数据行进行插值

生成由正态分布的随机数组成的三个单独的数据集。假定在正整数 1:N 这些点处进行数据采样。将数据集存储为矩阵中的行。

```
rng = MersenneTwister(1234)
A = randn(rng,3,20)
x = 1:20
```

对每 500 个查询点的矩阵行插值。指定 dim = 2 使用 interpft 对 A 的行进行插值。

```
N = 500
y = interpft(A,N,2)
```

计算插值数据点之间的间距 dy。截断 y 中的数据以匹配 x2 的采样密度。

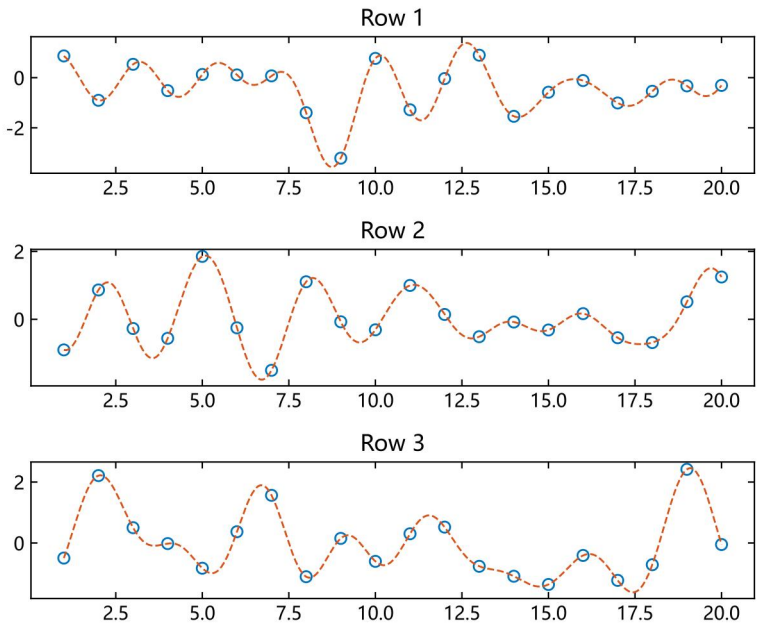
```
dy = length(x)/N
x2 = 1:dy:20
y = y[:,1:length(x2)]
```

绘制结果。

```
subplot(3,1,1)
plot(x,A[1,:],"o")
hold("on")
plot(x2,y[1,:],"--")
title("Row 1")
```

```
subplot(3,1,2)
plot(x,A[2,:],"o")
hold("on")
plot(x2,y[2,:],"--")
title("Row 2")

subplot(3,1,3)
plot(x,A[3,:],"o")
hold("on")
plot(x2,y[3,:],"--")
title("Row 3")
tightlayout()
```



3.2.4.1.8 makima

修正 Akima 分段三次 Hermite 插值

A. 语法

```
yq = makima(x,y,xq)
```

B. 说明

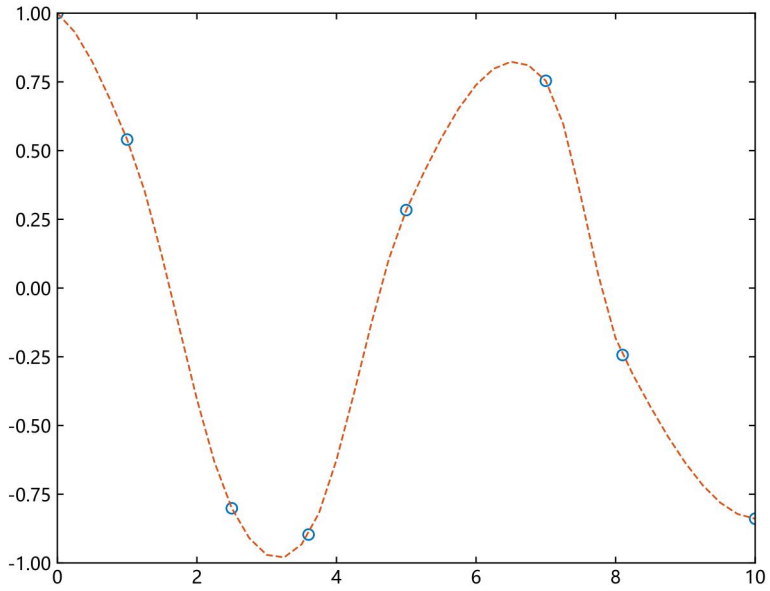
yq = makima(x,y,xq) 使用采样点 x 处的值 y 执行修正 Akima 插值，以求出查询点 xq 处的插值 yq。示例。

C. 示例

(1). 余弦数据的 Akima 插值

使用 makima 对非等距采样点上的余弦曲线进行插值。

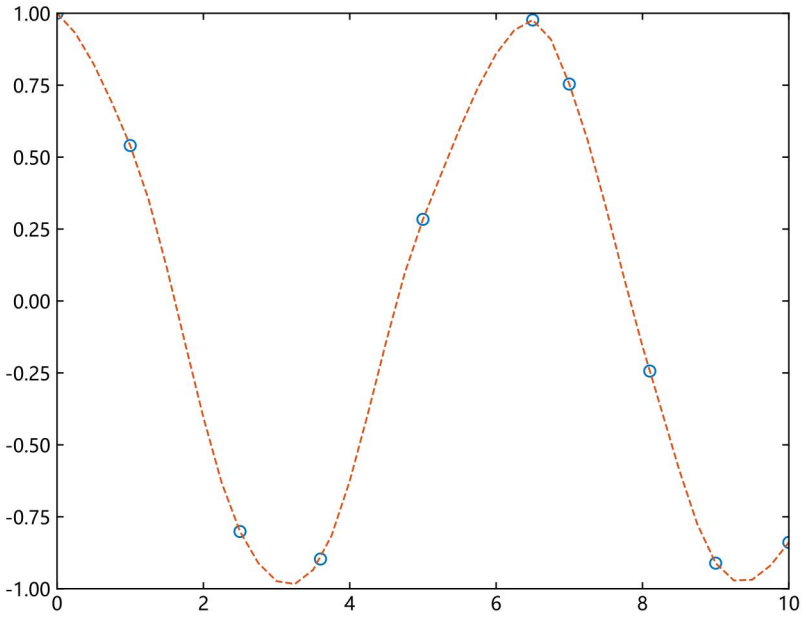
```
x = [0 1 2.5 3.6 5 7 8.1 10];
y = cos.(x);
xq = 0:0.25:10;
yq = makima(x, y, xq);
plot(x, y, "o", xq, yq, "--")
```



Akima 算法利用振荡函数使局部极值附近的曲线平坦化。为了补偿这种平坦化，您可以在局部极值附近添加更多采样点。

在 x=6.5 和 x=9 附近添加采样点，然后重新绘制插值。

```
x = [0 1 2.5 3.6 5 6.5 7 8.1 9 10];
y = cos.(x);
xq = 0:0.25:10;
yq = makima(x, y, xq);
plot(x, y, "o", xq, yq, "--")
```

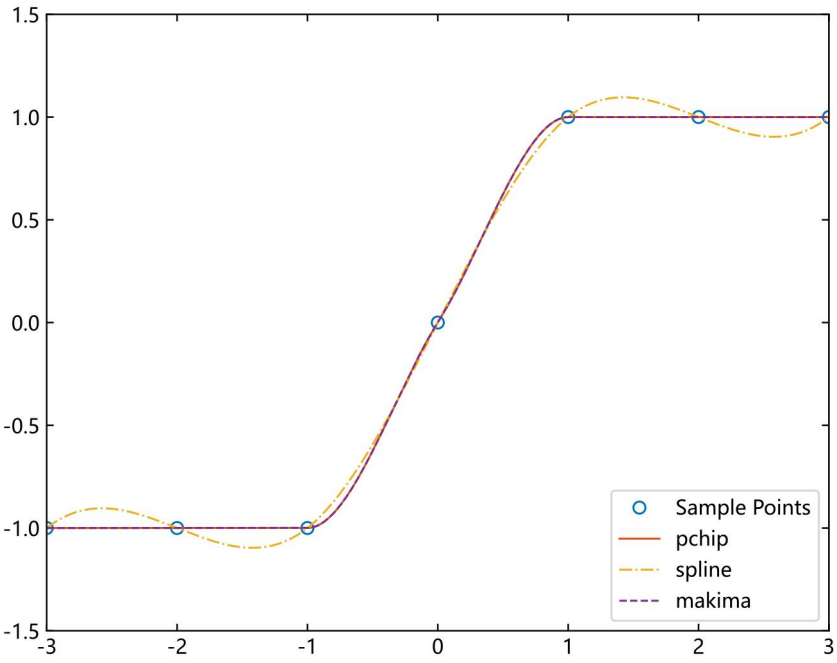


(2). 使用 spline、pchip 和 makima 的数据插值

将 spline、pchip 和 makima 为两个不同数据集生成的插值结果进行比较。这些函数都执行不同形式的分段三次 Hermite 插值。每个函数计算插值斜率的方式不同，因此它们在基础数据的平台区或波动处展现出不同行为。

对连接两个平台区的样本数据进行插值，并比较结果。创建由 x 值、点 y 处的函数值以及查询点 xq 组成的向量。使用 spline、pchip 和 makima 计算查询点处的插值。绘制查询点处的插值函数值以进行比较。

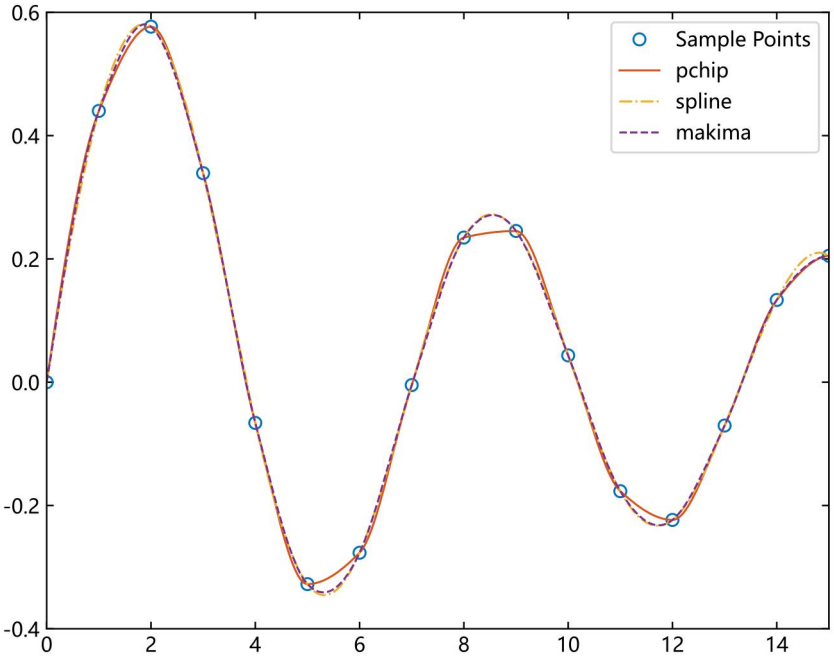
```
x = -3:3;
y = [-1 -1 -1 0 1 1 1];
xq1 = -3:0.01:3;
p = pchip(x, y, xq1);
s = spline(x, y, xq1);
m = makima(x, y, xq1);
plot(x, y, "o", xq1, p, "-", xq1, s, "-.", xq1, m, "--")
legend(["Sample Points", "pchip", "spline", "makima"], loc =
"southeast")
```



在本例中，pchip 和 makima 具有相似的行为，因为它们可以避免过冲，并且可以准确地连接平台区。

使用振动采样函数执行第二次比较。

```
x = 0:15;
y = besselj.(1, x);
xq2 = 0:0.01:15;
p = pchip(x, y, xq2);
s = spline(x, y, xq2);
m = makima(x, y, xq2);
plot(x, y, "o", xq2, p, "-", xq2, s, "-.", xq2, m, "--")
legend(["Sample Points", "pchip", "spline", "makima"])
```



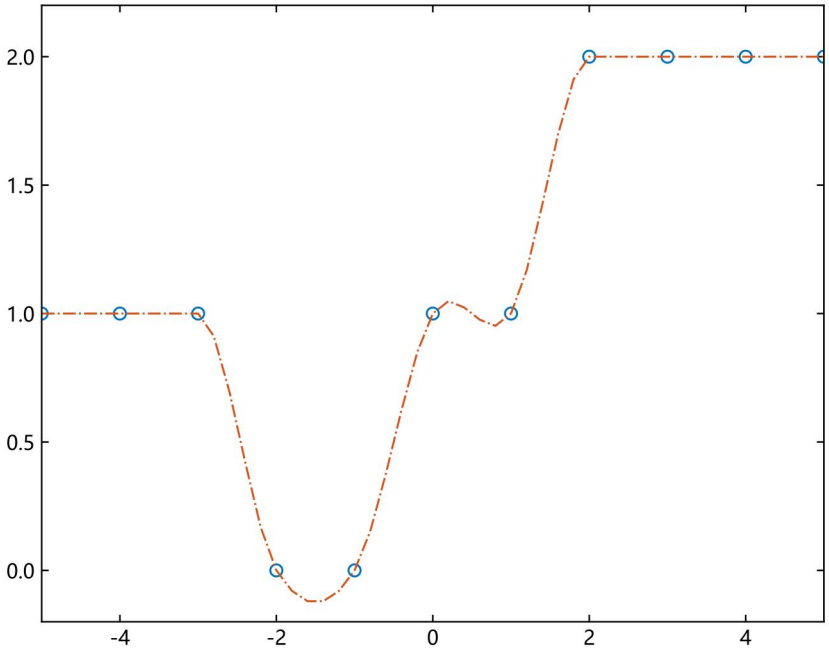
当基础函数振荡时，spline 和 makima 能够更好地捕捉点之间的波动，而 pchip 会在局部极值附近急剧平坦化。

(3). 使用分段多项式结构体进行 Akima 插值

创建采样点 x 及其值 y 的向量。

绘制结果。在有三个或更多连续等值点的区域，Akima 算法用直线将这些点连接起来。

```
x = [-5:5...];
y = vec([1 1 1 0 0 1 1 2 2 2 2])
xq = -5:0.2:5;
m = makima([x...], vec(y), xq)
plot(x, y, "o", xq, m, "-.")
ylim(-0.2,2.2)
```



3.2.4.1.9 chip

分段三次 Hermite 插值多项式 (PCHIP)

A. 语法

```
p = pchip(x,y,xq)
pp = pchip(x,y)
```

B. 说明

p = pchip(x,y,xq) 返回与 xq 中的查询点对应的插值 p 的向量。p 的值由 x 和 y 的保形分段三次插值确定。示例

pp = pchip(x,y) 返回一个分段多项式结构体以用于 ppval 和样条实用工具 unmkpp。

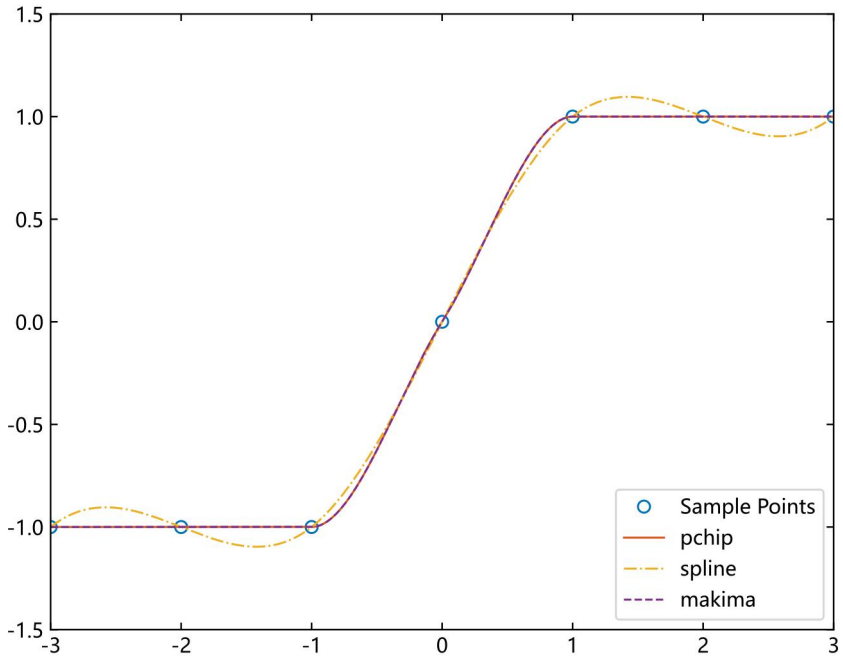
C. 示例

(1). 使用 spline、pchip 和 makima 的数据插值

将 spline、pchip 和 makima 为两个不同数据集生成的插值结果进行比较。这些函数都执行不同形式的分段三次 Hermite 插值。每个函数计算插值斜率的方式不同，因此它们在基础数据的平台区或波动处展现出不同行为。

对连接两个平台区的样本数据进行插值，并比较结果。创建由 x 值、点 y 处的函数值以及查询点 xq 组成的向量。使用 spline、pchip 和 makima 计算查询点处的插值。绘制查询点处的插值函数值以进行比较。

```
x = -3:3;
y = [-1 -1 -1 0 1 1 1];
xq1 = -3:0.01:3;
p = pchip(x, y, xq1);
s = spline(x, y, xq1);
m = makima(x, y, xq1);
plot(x, y, "o", xq1, p, "-", xq1, s, "-.", xq1, m, "--")
legend(["Sample Points", "pchip", "spline", "makima"], loc =
"southeast")
```

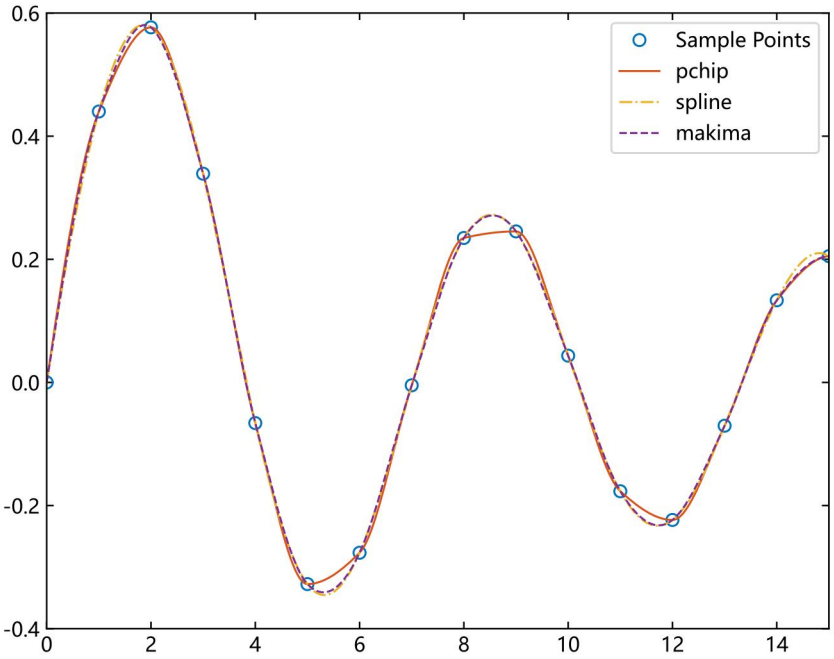


在本例中，pchip 和 makima 具有相似的行为，因为它们可以避免过冲，并且可以准确地连接平台区。

使用振动采样函数执行第二次比较。

```
x = 0:15;
y = besselj.(1, x);
xq2 = 0:0.01:15;
p = pchip(x, y, xq2);
s = spline(x, y, xq2);
m = makima(x, y, xq2);
plot(x, y, "o", xq2, p, "-", xq2, s, "-.", xq2, m, "--")
```

```
legend(["Sample Points", "pchip", "spline", "makima"])
```



当基础函数振荡时，spline 和 makima 能够比 pchip 更好地捕获点之间的移动，后者会在局部极值附近急剧扁平化。

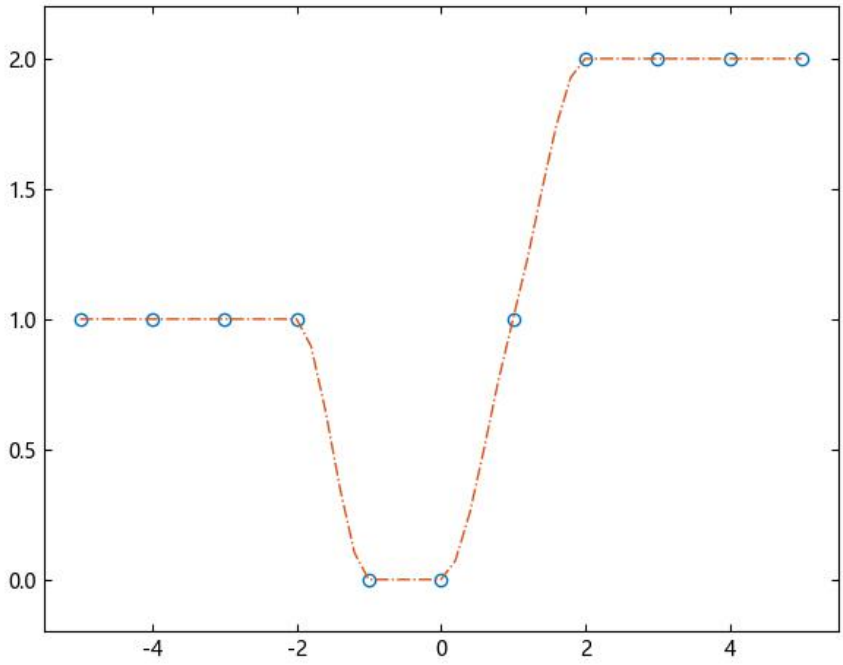
(2). 使用分段多项式结构体进行插值

创建 x 值及其函数值 y 的向量，然后使用 pchip 来构造一个分段多项式结构体。

```
x = -5:5;
y = [1 1 1 1 0 0 1 2 2 2 2];
p = pchip(x, y);
```

结合 ppval 使用该结构体以计算几个查询点处的插值。绘制结果。

```
xq = -5:0.2:5;
pp = ppval(p, xq);
plot(x, y, "o", xq, pp, "-.")
ylim(-0.2, 2.2)
```



3.2.4.1.10 interpolate

插值

A. 语法

```
itp = interpolate(A, interpmode)
itp = interpolate((nodes1, nodes2, ...), A, interpmode)
```

B. 说明

itp = interpolate(A, interpmode) 对数组 A 使用 插值方法 interpmode 进行插值。

itp = interpolate((nodes1, nodes2, ...), A, interpmode)返回对节点 nodes 以及节点对应值 A 采用 插值方法 interpmode 的插值函数

C. 示例

(1). 一维插值

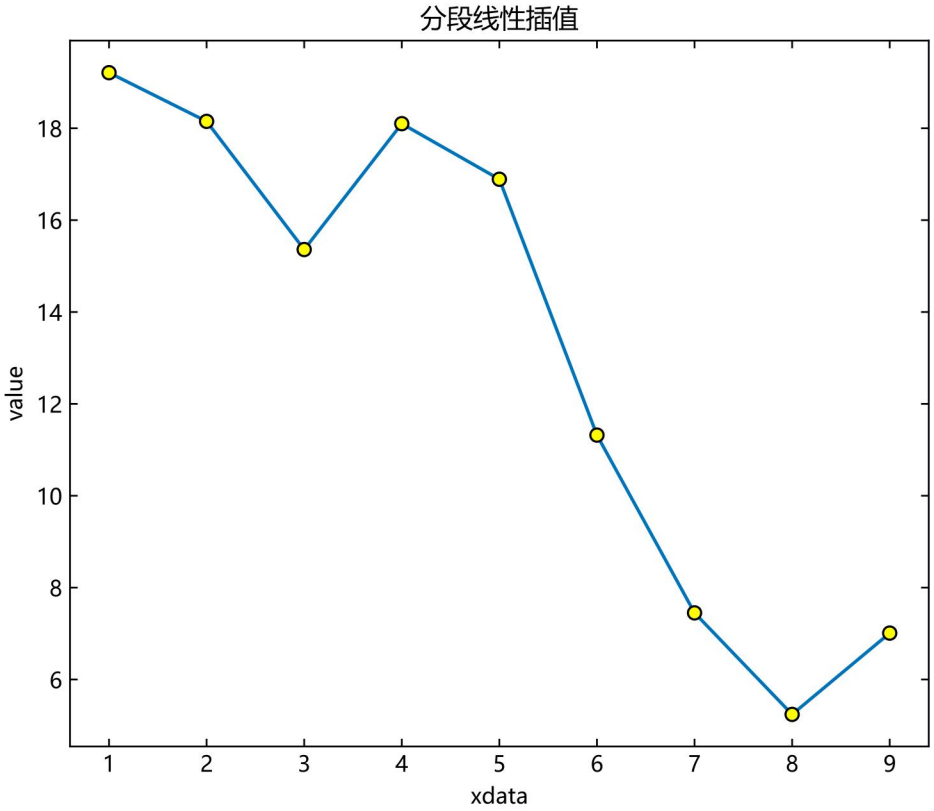
创建平面坐标系内散点。

```
y = [19.21 , 18.15 , 15.36 , 18.10 , 16.89 , 11.32 , 7.45 , 5.24 ,
```

```
7.01]
```

采用分段线性插值并绘图

```
itp = interpolate(y,BSpline(Linear()))
xq = 1:0.1:9 # 插值数据
hold("on")
plot(xq,itp.(xq))
plot(1:9,y,"o",markeredgecolor = "k",markerfacecolor = "#FFFF00")
title("分段线性插值")
xlabel("xdata")
ylabel("value")
hold("off")
```

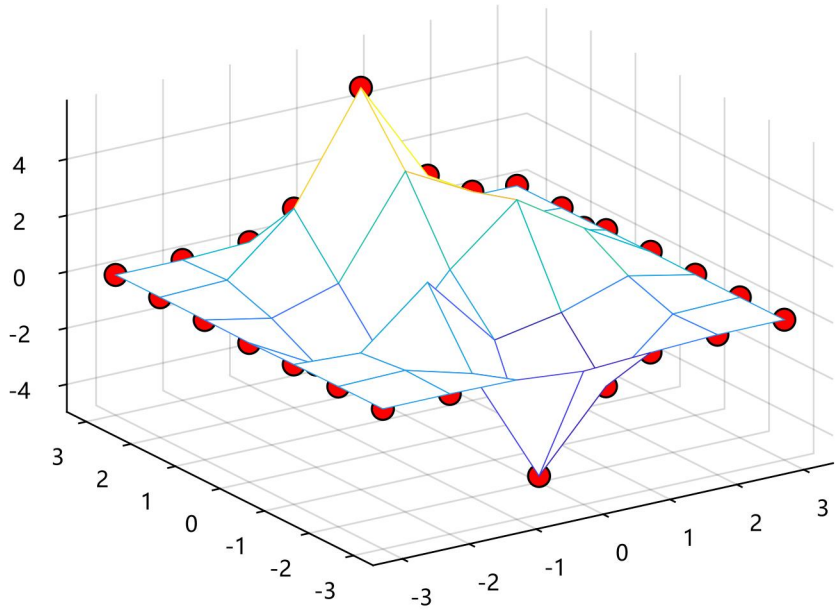


(2). 二维插值

构造初始数据点并绘制原始数据图

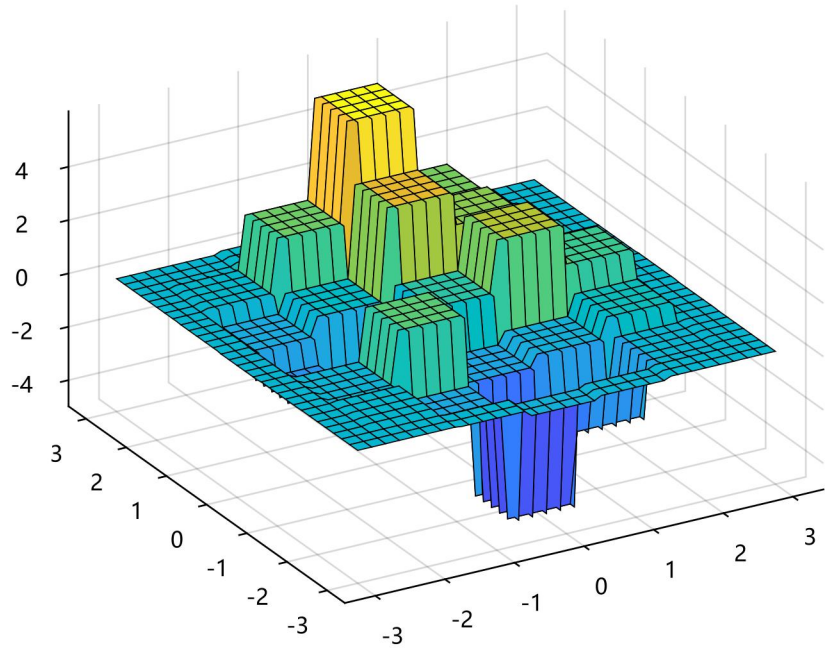
```
f(x,y) = 3*(1-x)^2*exp(-(x^2) - (y+1)^2) - 10*(x/5 - x^3 -
y^5)*exp(-x^2-y^2) - 1/3*exp(-(x+1)^2 - y^2)
x = -3:1:3
y = -3:1:3
X,Y = ndgrid(x,y)
Z = f.(X,Y)
```

```
mesh(X,Y,Z)
hold("on")
plot3(X,Y,Z,"o",markersize = 10,markeredgecolor =
"k",markerfacecolor = "r")
grid("on")
hold("off")
```



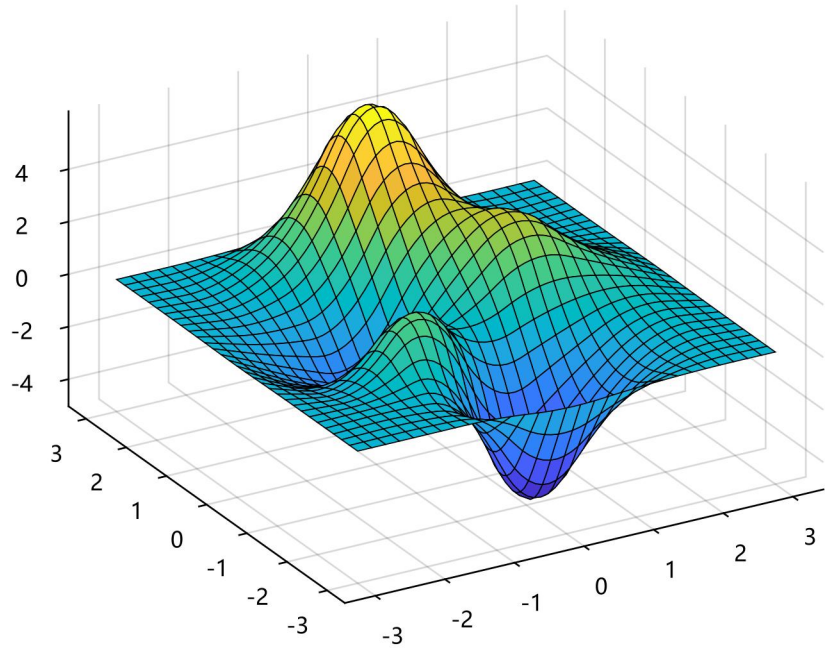
采用最邻近插值并绘制图

```
itp = interpolate((x,y),Z,Gridded(Constant()))
xt = -3:0.1:3
yt = xt
Xt,Yt = meshgrid2(xt,yt)
surf(Xt,Yt,itp.(Xt,Yt))
```



采用三次样条插值并绘图

```
itp = interpolate(Z,BSpline(Quadratic(Line(OnCell()))))
sitp = scale(itp,x,y)
xt = -3:0.1:3
yt = xt
Xt,Yt = meshgrid2(xt,yt)
surf(Xt,Yt,sitp.(Xt,Yt))
```



(3). 四维插值

构造初始数据点

```
f(x,y,z,t) = t * exp( -x^2 - y^2 - z^2)
xx = yy = zz = -1:.2:1
tt = 0:2:10
x,y,z,t = ndgrid(xx,yy,zz,tt)
V = f.(x,y,z,t)
```

对原始数据进行插值,并得到插值函数 sitp

```
itp = interpolate(V,BSpline(Cubic(Line(OnGrid()))))
sitp = scale(itp,xx,yy,zz,tt)
```

对点 (x,y,z,t) = (-1,-0.5,0.2,4) 处进行求值

```
sitp(-1,-0.5,0.2,4)
```

```
ans = 1.1012212045782097
```

3.2.4.1.11 extrapolate

外插

A. 语法

```
sitp =  extrapolate(itp, scheme)
```

B. 说明

exitp = extrapolate(itp, scheme) 对得到的插值方法 itp，使用 scheme 规则转变为外插形式。

C. 示例

(1). 一维插值

创建平面坐标系内散点。

```
y = [19.21 , 18.15 , 15.36 , 18.10 , 16.89 , 11.32 , 7.45 , 5.24 , 7.01]
```

得到线性样条插值，y 长度为 9，因此默认自变量为 1:9 的网格插值

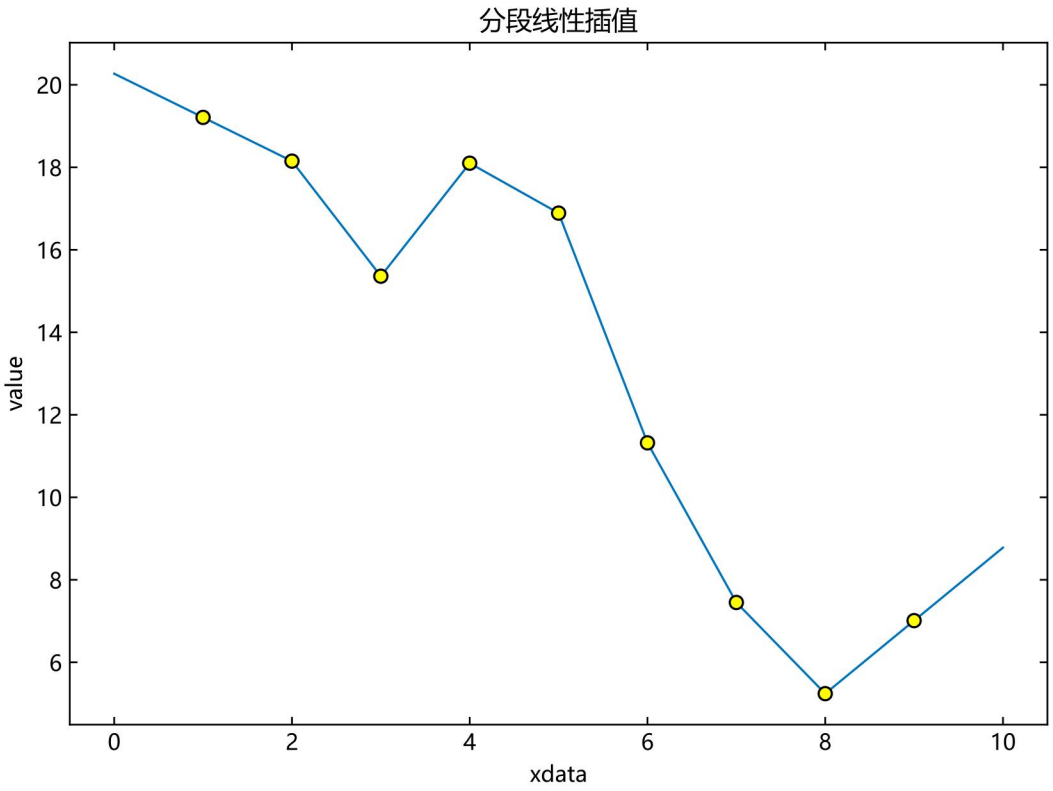
```
itp = interpolate(y,BSpline(Linear()))
```

采用线性边界条件 Line(),将插值方法转变为外插。

```
exitp = extrapolate(itp,Line())
```

对结果进行绘图

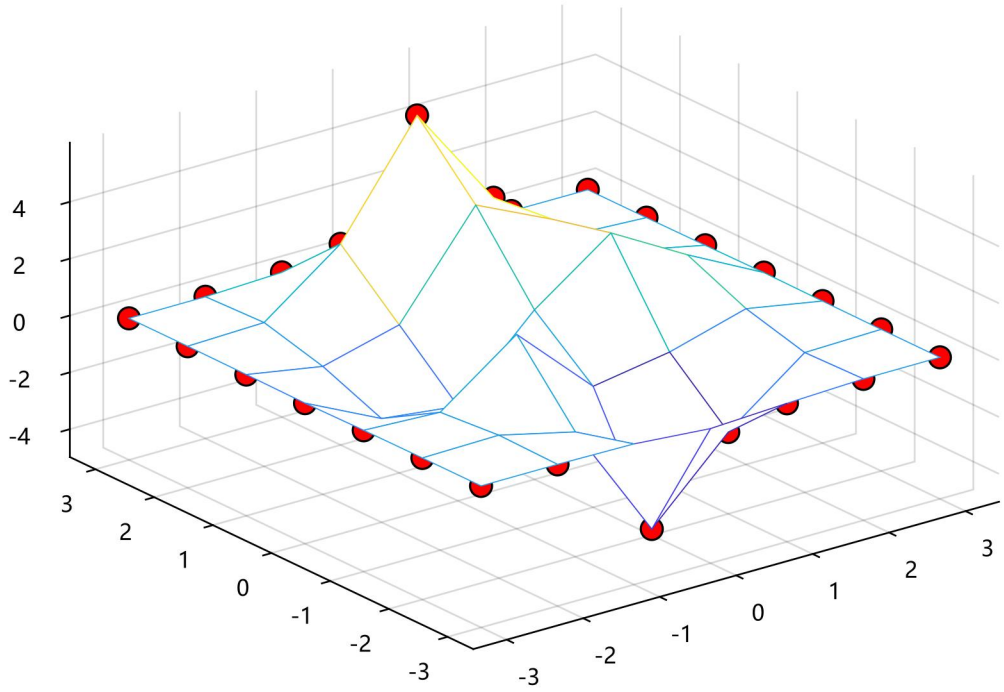
```
xq = 0:0.1:10
hold("on")
plot(xq,exitp.(xq))
plot(1:9,y,"o",markeredgecolor = "k",markerfacecolor = "#FFFF00")
title("分段线性插值")
xlabel("xdata")
ylabel("value")
hold("off")
```



(2). 二维插值

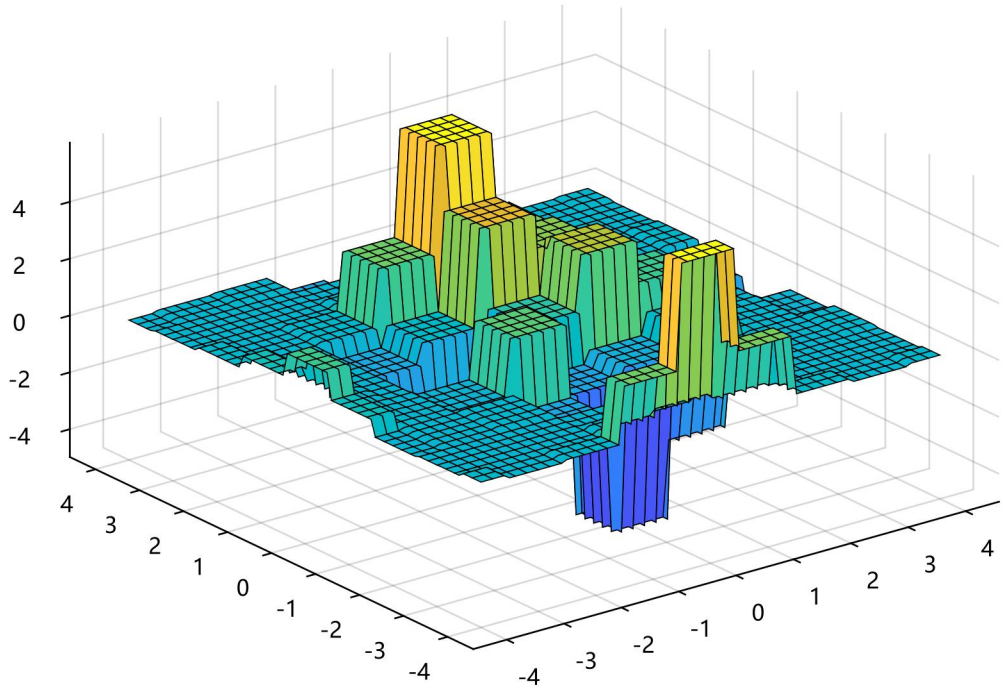
构造初始数据点并绘制原始数据图

```
f(x,y) = 3*(1-x)^2*exp(-(x^2) - (y+1)^2) - 10*(x/5 - x^3 -
y^5)*exp(-x^2-y^2) - 1/3*exp(-(x+1)^2 - y^2)
x = -3:1:3
y = -3:1:3
X,Y = ndgrid(x,y)Z = f.(X,Y)
mesh(X,Y,Z)
hold("on")
plot3(X,Y,Z,"o",markersize = 10,markeredgecolor =
"k",markerfacecolor = "r")
grid("on")
hold("off")
```



采用最邻近插值(外插)并绘制图

```
itp = interpolate((x,y),Z,Gridded(Constant()))
exitp = extrapolate(itp,Periodic())
xt = -4:0.1:4
yt = xt
Xt,Yt = meshgrid2(xt,yt)
surf(Xt,Yt,exitp.(Xt,Yt))
```



3.2.4.1.12 LagrangeInterp

拉格朗日插值

A. 语法

```
Y =  LagrangeInterp(x0,y0,x)
```

B. 说明

Y = LagrangeInterp(x0,y0,x) 返回对节点 x0,y0 做拉格朗日插值，求出插值函数对应 x 的 y 值 。

C. 示例

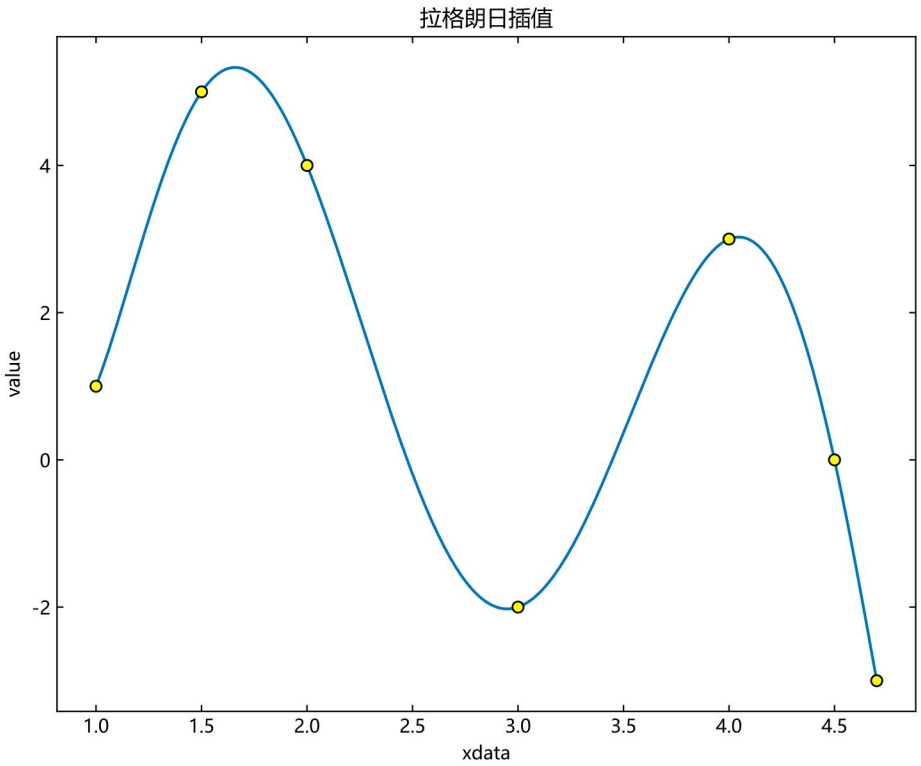
(1). 拉格朗日插值

创建平面坐标系内散点

```
xdata = [1,1.5,2,3,4,4.5,4.7]
ydata = [1,5,4,-2,3,0,-3]
```

进行拉格朗日插值并绘图

```
xq = 1:0.01:4.7
yq = LagrangeInterp(xdata,ydata,xq)
hold("on")
plot(xq,yq)
plot(xdata,ydata,"o",markeredgecolor = "k",markerfacecolor = "#FFFF00")
title("拉格朗日插值")
xlabel("xdata")
ylabel("value")
hold("off")
```



3.2.4.1.13 NewtonInterp

牛顿插值

A. 语法

```
Y = NewtonInterp(x0,y0,x)
```

B. 说明

Y = NewtonInterp(x0,y0,x) 返回对节点 x0,y0 做牛顿插值，求出插值函数

对应 x 的 y 值。

C. 示例

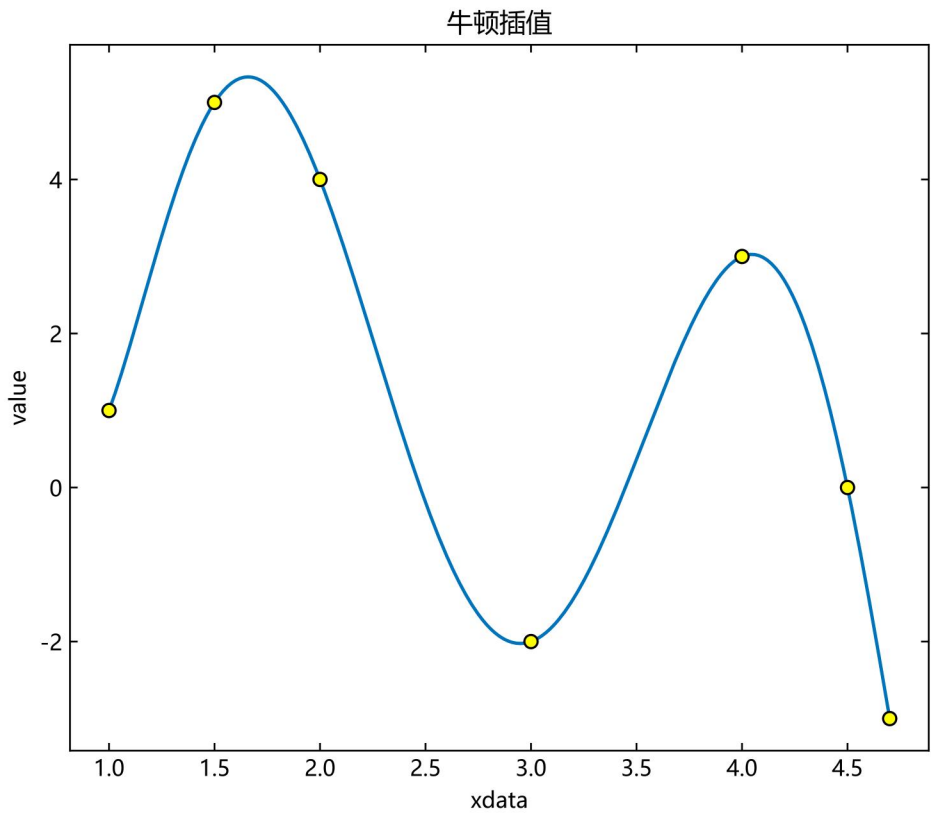
(1). 牛顿插值

创建平面坐标系内散点

```
xdata = [1,1.5,2,3,4,4.5,4.7]
ydata = [1,5,4,-2,3,0,-3]
```

进行牛顿插值并绘图

```
xq = 1:0.01:4.7
yq = NewtonInterp(xdata,ydata,xq)
hold("on")
plot(xq,yq)
plot(xdata,ydata,"o",markeredgecolor = "k",markerfacecolor = "#FFFF00")
title("牛顿插值")
xlabel("xdata")
ylabel("value")
hold("off")
```



3.2.4.1.14 ConstantInterpolation

最邻近插值

A. 语法

```
etp =  ConstantInterpolation(knots,A;extrapolation_bc=Throw())
```

B. 说明

Y = ConstantInterpolation(knots,A;extrapolation_bc=Throw()) 返回对节点 knots 以及节点对应值 A 采用最邻近插值的插值函数，extrapolation_bc 为边界条件，默认为 Throw()。

C. 示例

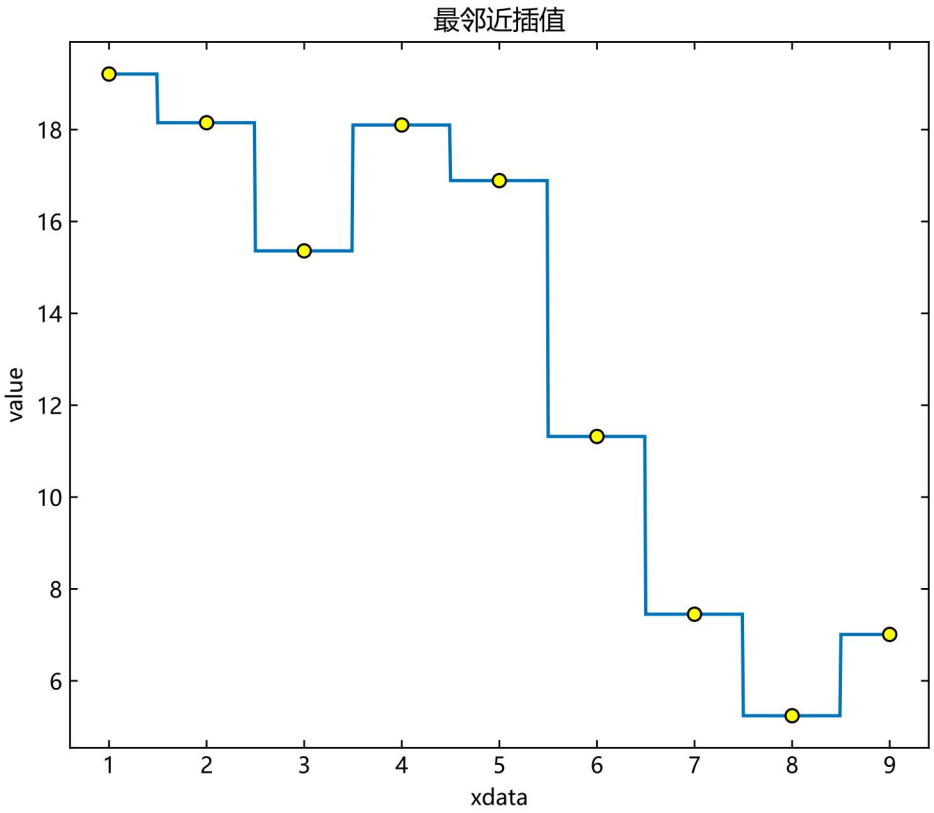
(1). 一维最邻近插值

创建平面坐标系内散点。

```
x = 1:9
y = [19.21 , 18.15 , 15.36 , 18.10 , 16.89 , 11.32 , 7.45 , 5.24 , 7.01]
```

进行最邻近插值并绘图

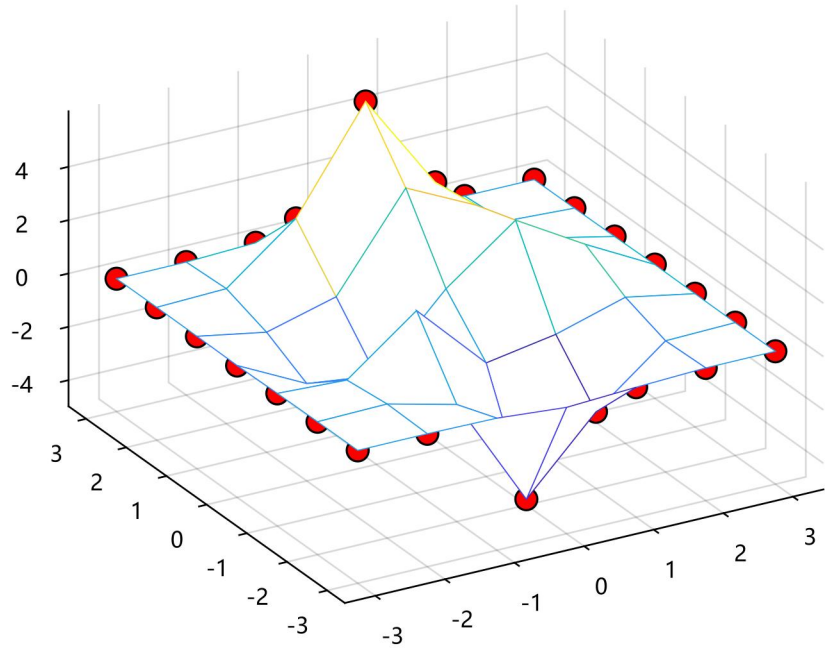
```
xq = 1:0.01:9
etp = ConstantInterpolation(x,y)
hold("on")
plot(xq,etp(xq))
plot(x,y,"o",markeredgecolor = "k",markerfacecolor = "#FFFF00")
title("最邻近插值")
xlabel("xdata")
ylabel("value")
hold("off")
```



(2). 二维最邻近插值

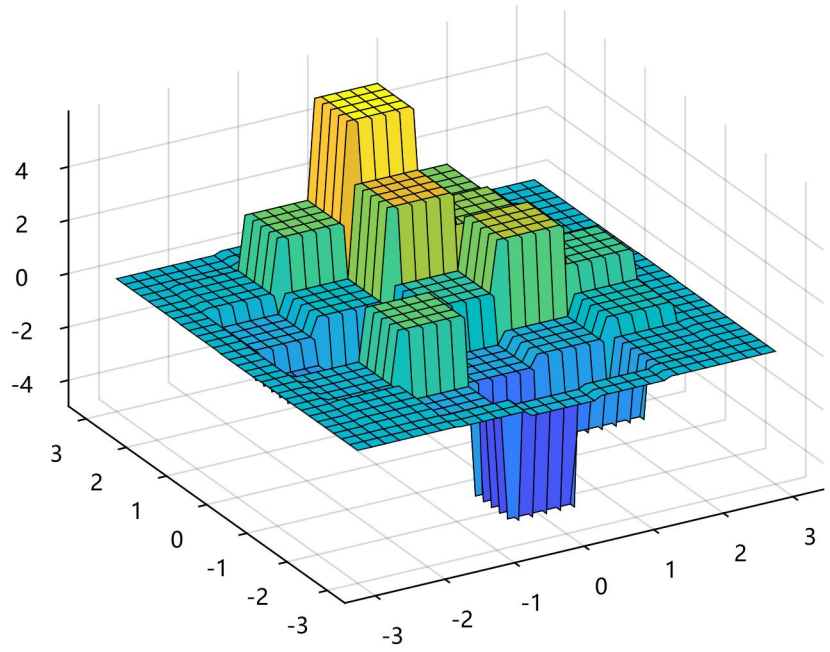
构造初始数据点并绘制原始数据图

```
f(x,y) = 3*(1-x)^2*exp(-(x^2) - (y+1)^2) - 10*(x/5 - x^3 -
y^5)*exp(-x^2-y^2) - 1/3*exp(-(x+1)^2 - y^2)
x = -3:1:3
y = -3:1:3
X,Y = ndgrid(x,y)
Z = f.(X,Y)
mesh(X,Y,Z)
hold("on")
plot3(X,Y,Z,"o",markersize = 10,markeredgecolor =
"k",markerfacecolor = "r")
grid("on")
hold("off")
```



对 x 、 y 都使用最邻近插值并绘图

```
etp = ConstantInterpolation((x,y),Z,extrapolation_bc=Throw())
xt = -3:0.1:3
yt = xt
Xt,Yt = meshgrid2(xt,yt)
surf(Xt,Yt,etp.(Xt,Yt))
```



3.2.4.1.15 LinearInterpolation

线性插值

A. 语法

```
tp =  LinearInterpolation(knots,A;extrapolation_bc=Throw())
```

B. 说明

Y = LinearInterpolation(knots,A;extrapolation_bc=Throw()) 返回对节点 knots 以及节点对应值 A 采用线性插值的插值函数，extrapolation_bc 为边界条件，默认为 Throw()。

C. 示例

(1). 一维线性插值

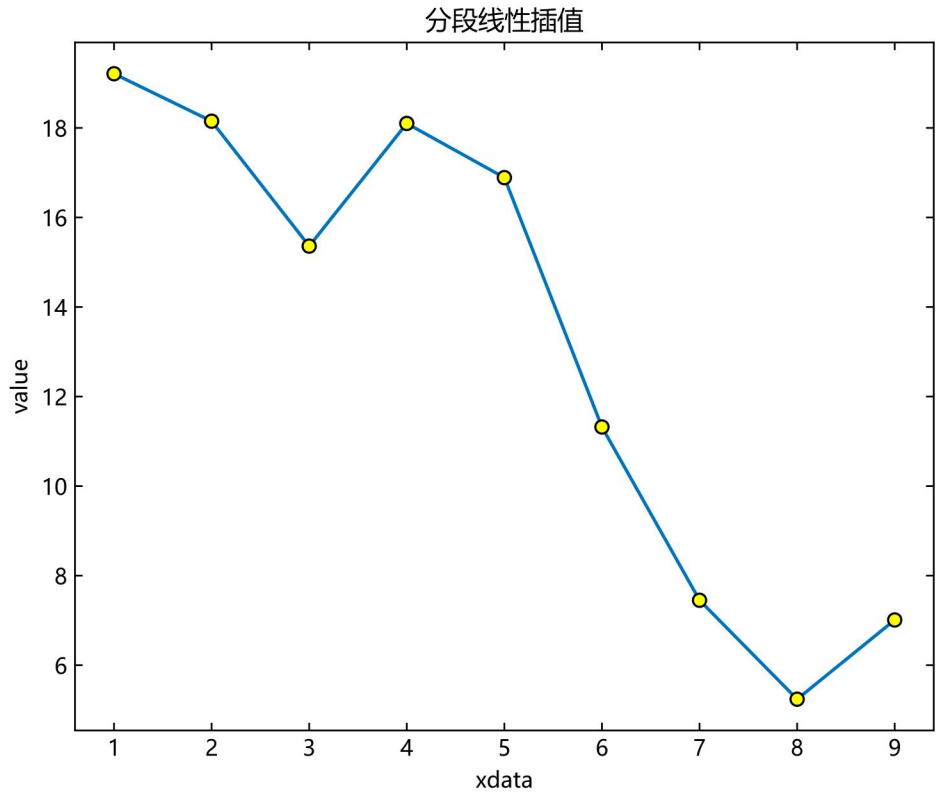
创建平面坐标系内散点。

```
x = 1:9
y = [19.21 , 18.15 , 15.36 , 18.10 , 16.89 , 11.32 , 7.45 , 5.24 ,
```

```
7.01]
```

进行线性插值并绘图

```
xq = 1:0.01:9
etp = LinearInterpolation(x,y)
hold("on")
plot(xq,etp(xq))
plot(x,y,"o",markeredgecolor = "k",markerfacecolor = "#FFFF00")
title("分段线性插值")
xlabel("xdata")
ylabel("value")
hold("off")
```

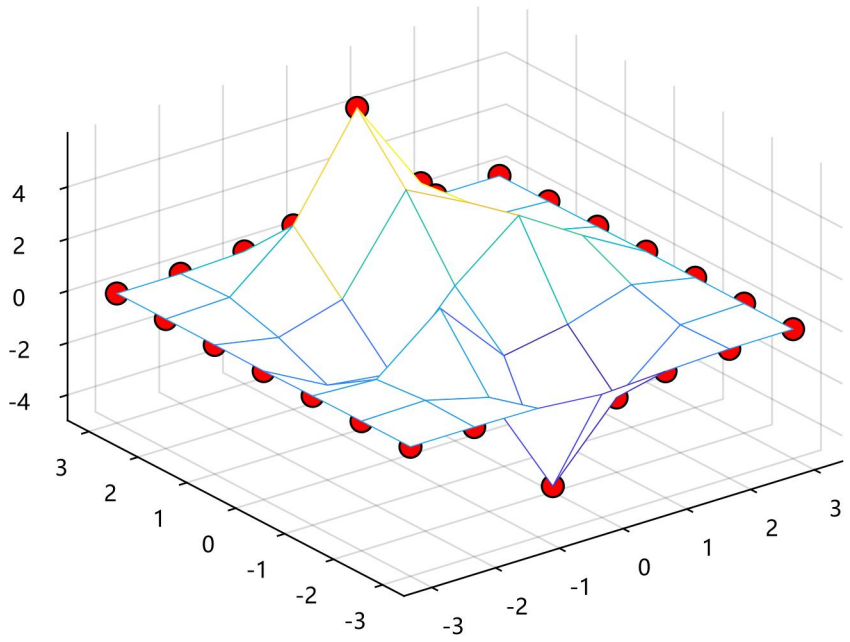


(2). 双线性插值

构造初始数据点并绘制原始数据图

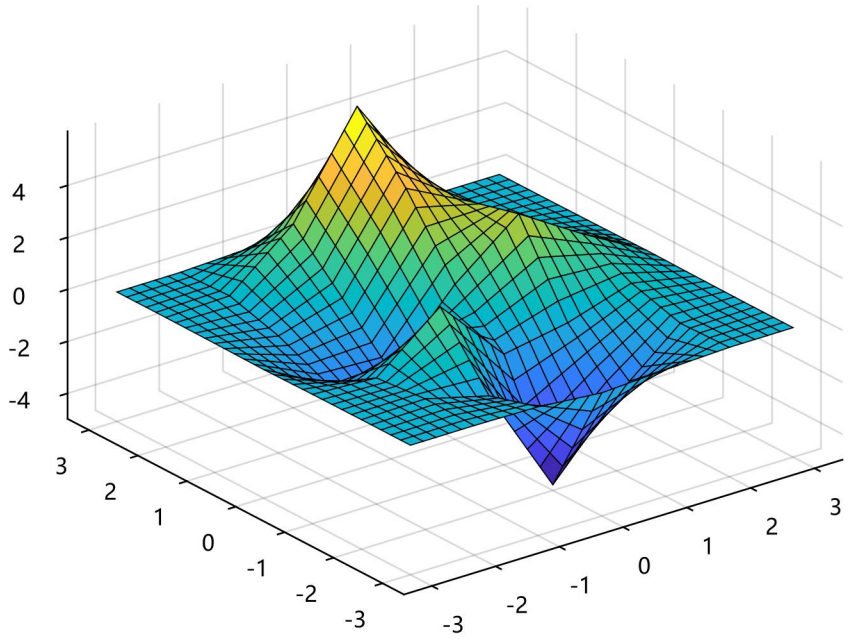
```
f(x,y) = 3*(1-x)^2*exp(-(x^2) - (y+1)^2) - 10*(x/5 - x^3 -
y^5)*exp(-x^2-y^2) - 1/3*exp(-(x+1)^2 - y^2)
x = -3:1:3
y = -3:1:3
X,Y = ndgrid(x,y)
Z = f.(X,Y)
```

```
mesh(X,Y,Z)
hold("on")
plot3(X,Y,Z,"o",markersize = 10,markeredgecolor =
"k",markerfacecolor = "r")
grid("on")
hold("off")
```



对 x 、 y 都使用线性插值并绘图

```
etp = LinearInterpolation((x,y),Z,extrapolation_bc=Throw())
xt = -3:0.1:3
yt = xt
Xt,Yt = meshgrid2(xt,yt)
surf(Xt,Yt,etp.(Xt,Yt))
```



获取 (x,y) = (1.2,2.3) 处的插值结果

```
etp(1.2,2.3)
```

```
ans = 1.2830664663210183
```

3.2.4.1.16 CubicSplineInterpolation

三次样条插值

A. 语法

```
etp =  CubicSplineInterpolation(knots,A;extrapolation_bc=Throw())
```

B. 说明

Y = CubicSplineInterpolation(knots,A;extrapolation_bc=Throw()) 返回对节点 knots 以及节点对应值 A 采用三次样条插值的插值函数，extrapolation_bc 为边界条件，默认为 Throw()。

C. 示例

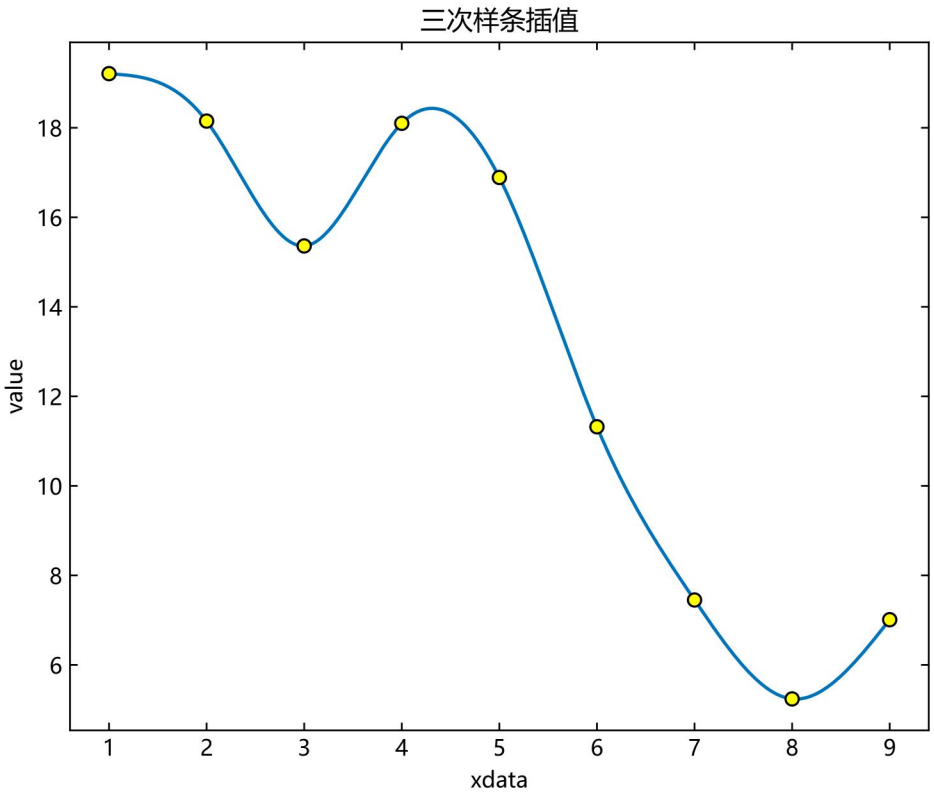
(1). 一维三次样条插值

创建平面坐标系内散点。

```
x = 1:9
y = [19.21 , 18.15 , 15.36 , 18.10 , 16.89 , 11.32 , 7.45 , 5.24 , 7.01]
```

进行三次样条插值并绘图

```
xq = 1:0.01:9
etp = CubicSplineInterpolation(x,y)
hold("on")
plot(xq,etp(xq))
plot(x,y,"o",markeredgecolor = "k",markerfacecolor = "#FFFF00")
title("三次样条插值")
xlabel("xdata")
ylabel("value")
hold("off")
```

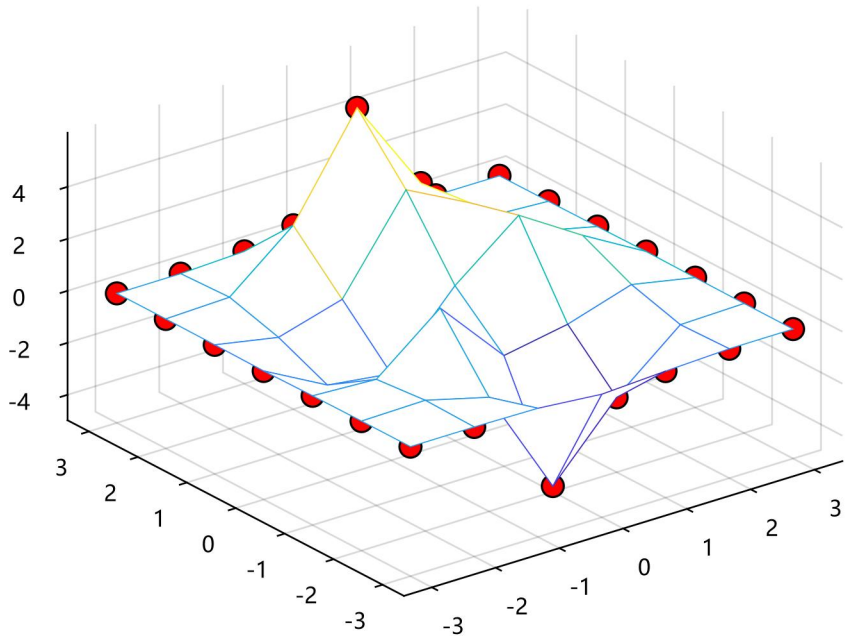


(2). 双三次样条插值

构造初始数据点并绘制原始数据图

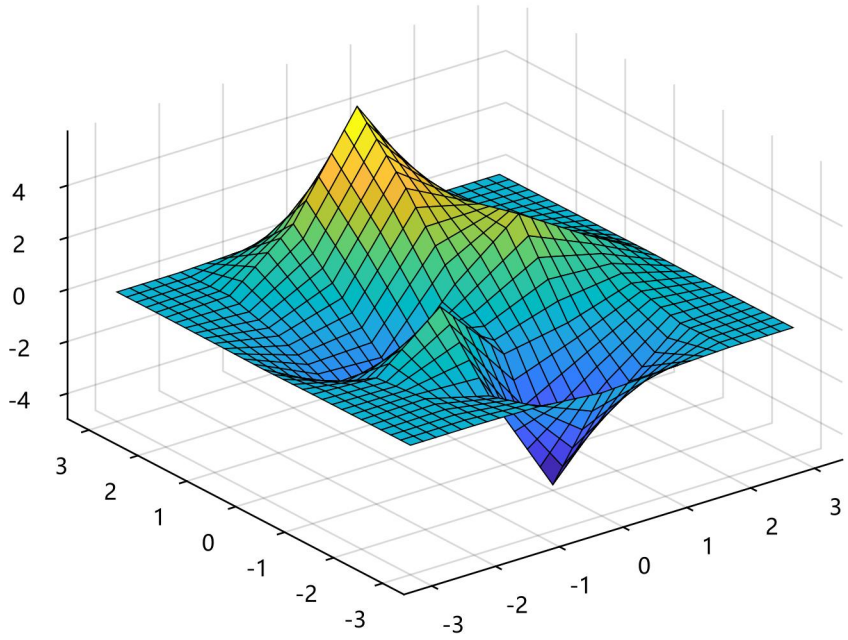
```
f(x,y) = 3*(1-x)^2*exp(-(x^2) - (y+1)^2) - 10*(x/5 - x^3 - y^5)*exp(-x^2-y^2) - 1/3*exp(-(x+1)^2 - y^2)
x = -3:1:3
```

```
y = -3:1:3
X,Y = ndgrid(x,y)Z = f.(X,Y)
mesh(X,Y,Z)
hold("on")
plot3(X,Y,Z,"o",markersize = 10,markeredgecolor =
"k",markerfacecolor = "r")
grid("on")
hold("off")
```



对 x 、 y 都使用三次样条插值并绘图

```
etp = CubicSplineInterpolation((x,y),Z,extrapolation_bc=Throw())
xt = -3:0.1:3
yt = xt
Xt,Yt = meshgrid2(xt,yt)
surf(Xt,Yt,etp.(Xt,Yt))
```



获取 (x,y) = (1.2,2.3) 处的插值结果

```
etp(1.2,2.3)
```

```
ans = 1.1861370547105412
```

3.2.4.1.17 sinc_interpolate

一维插值（FFT 方法）

A. 语法

```
y =  sinc_interpolate(arr,new_size,normalize = true)
```

B. 说明

y = sinc_interpolate(arr,new_size,normalize = true) 计算一个新的数组大小为 new_size 的 arr 的 sinc 插值。这种方法是基于 fft 的，因此隐含地假设周期边界和有限的频率支持。Normalize =true 默认情况下乘以一个适当的因子，使平均强度保持不变。

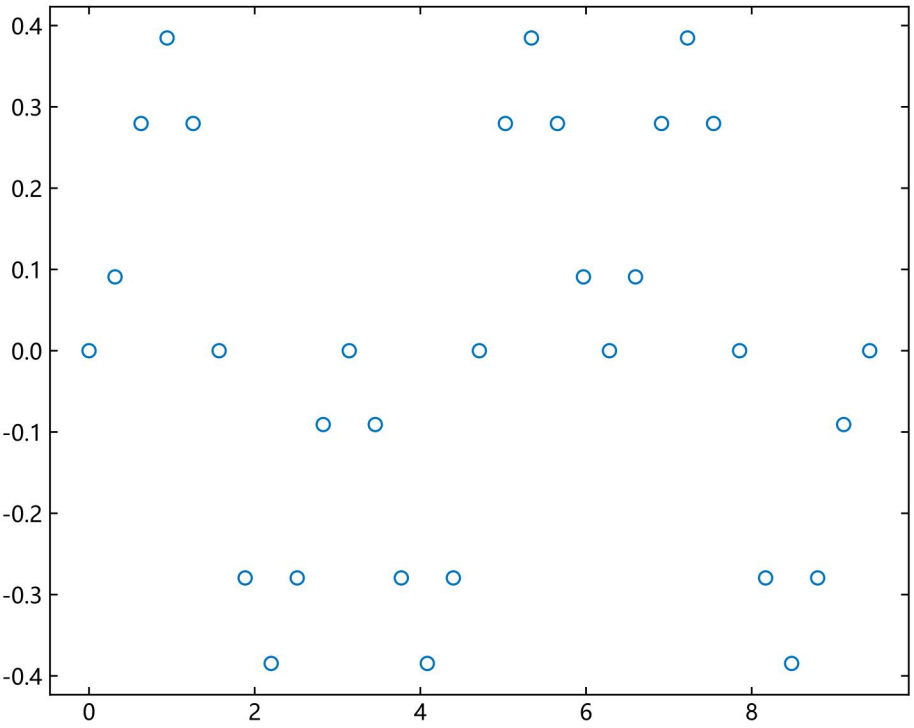
C. 示例

(1). 傅里叶插值

使用 FFT 方法对一维数据插值，并以可视方式呈现结果。

在区间 [0,3 π] 内为函数 $f(x) = sin^2(x)cos(x)$ 生成一些样本点。使用间距 dx 以确保数据均匀分布。绘制样本点。

```
dx = 3*pi/30
x = 0:dx:3*pi
f = sin.(x).^2 .* cos.(x)
plot(x,f,"o")
```



使用 FFT 插值，以查找 200 个查询点处的函数值。

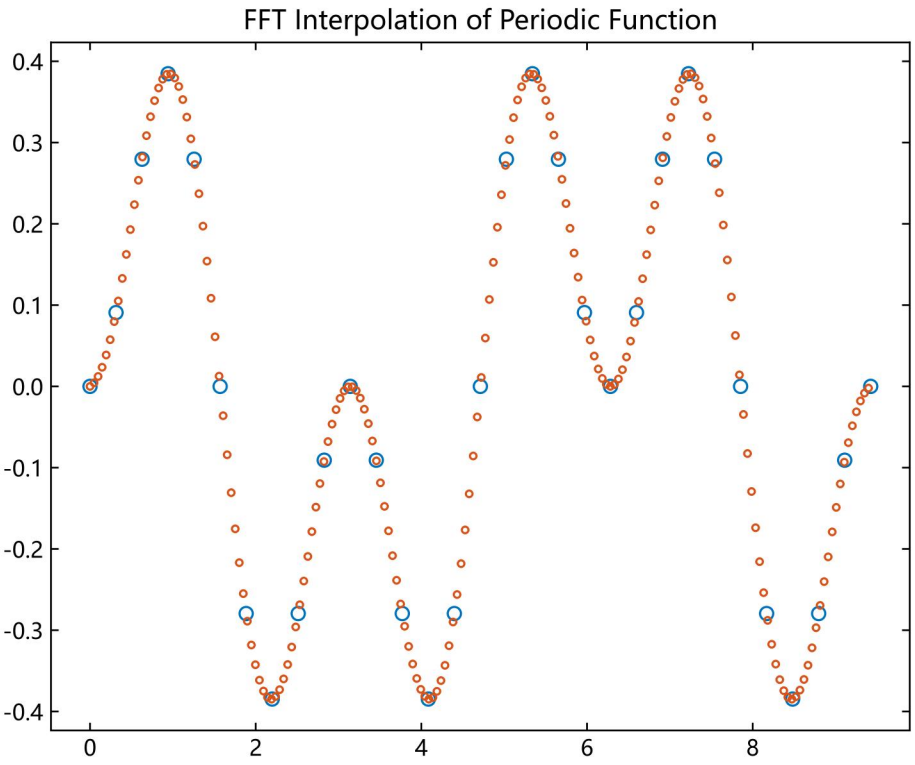
```
N = 200
y = sinc_interpolate(f,N)
```

使用 dy = dx*length(x)/N 基于样本点的间距计算插值数据的间距，其中 N 是插值点的数目。截断 y 中的数据以匹配 x2 的采样密度。

```
dy = dx*length(x)/N
x2 = 0:dy:3*pi
y = y[1:length(x2)]
```

绘制结果。

```
hold("on")
plot(x2,y,".")
title("FFT Interpolation of Periodic Function")
```



(2). 对数据进行插值

生成由正态分布的随机数组成的三个单独的数据集。假定在正整数 1:N 这些点处进行数据采样。将数据集存储为矩阵中的行。

```
A = randn(3,20)
x = 1:20
```

对每 500 个查询点的矩阵行插值。指定 dim = 2 以便 interpft 对 A 的行进行插值。

```
N = 500
y = sinc_interpolate(A,(3,N))
```

计算插值数据点之间的间距 dy。截断 y 中的数据以匹配 x2 的采样密度。

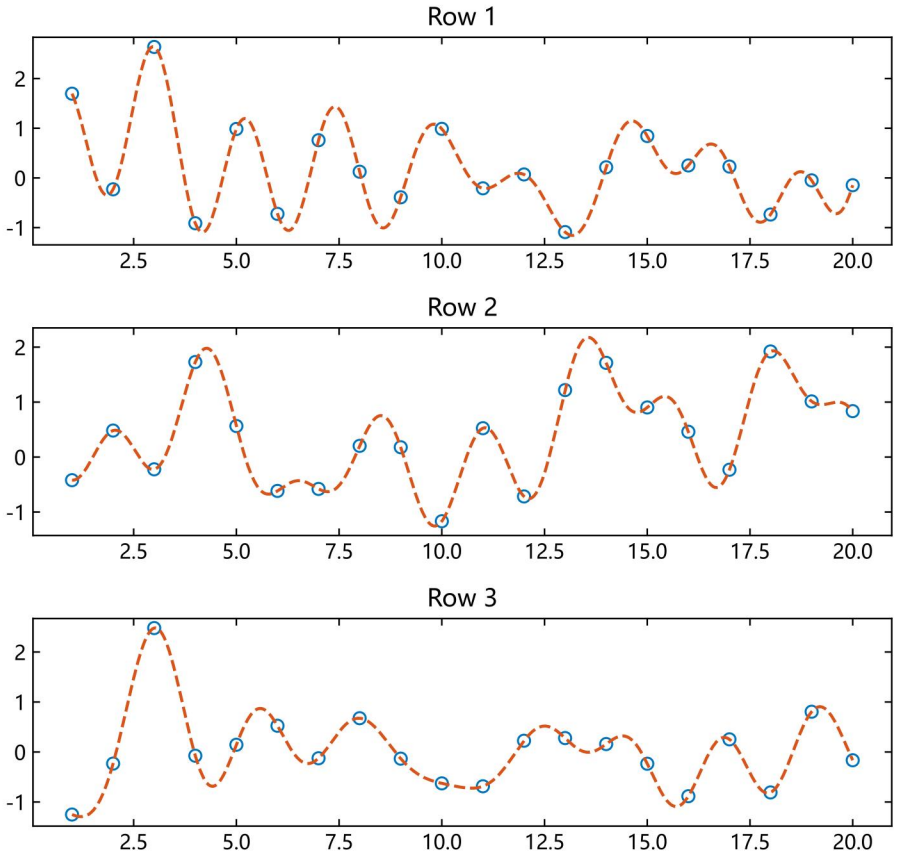
```
dy = length(x)/N
x2 = 1:dy:20
y = y[:,1:length(x2)]
```

绘制结果。

```
subplot(3,1,1)
plot(x,A[1,:],"o")
hold("on")
plot(x2,y[1,:],"--")
title("Row 1")

subplot(3,1,2)
plot(x,A[2,:],"o")
hold("on")
plot(x2,y[2,:],"--")
title("Row 2")

subplot(3,1,3)
plot(x,A[3,:],"o")
hold("on")
plot(x2,y[3,:],"--")
title("Row 3")
```



3.2.4.1.18 mkpp

生成分段多项式

A. 语法

```
pp = mkpp(breaks,coefs)
pp = mkpp(breaks,coefs,d)
```

B. 说明

pp = mkpp(breaks,coefs) 根据其间断数和系数生成分段多项式 pp。使用 ppval 计算特定点处的分段多项式，或使用 unmkpp 提取有关分段多项式的详细信息。

pp = mkpp(breaks,coefs,d) 指定分段多项式为向量值，以使其每个系数的值都是长度为 d 的向量。

C. 示例

(1). 创建包含多阶多项式的分段多项式

创建一个分段多项式，它在区间 [0,4] 内具有三次多项式，在区间 [4,10] 内具有二次多项式，在区间 [10,15] 内具有四次多项式。

```
breaks = [0 4 10 15];
coefs = [0 1 -1 1 1; 0 0 1 -2 53; -1 6 1 4 77];
pp = mkpp(breaks,coefs)
```

```
 pp = MKPP
form: String "pp"
breaks: Array{Int64}((1, 4)) [0 4 10 15]
coefs: Array{Int64}((3, 5)) [0 1 ⋯ 1 1; 0 0 ⋯ -2 53; -1 6 ⋯ 4 77]
pieces: Int64 3
order: Int64 5
dim: Int64 1
```

MKPP 的结构体字段

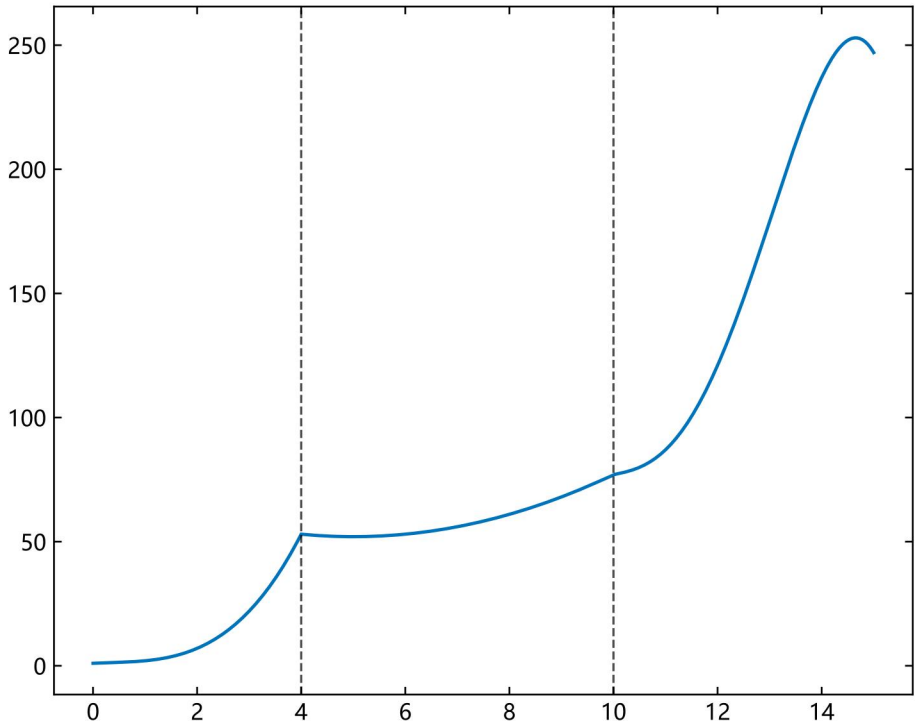
```
fieldnames(TyMath.TyInterpolations.MKPP)
```

```
(:form, :breaks, :coefs, :pieces, :order, :dim)
```

计算区间 [0,15] 内多个点处的分段多项式，并绘制结果图。在多项式汇合的断点处绘制垂直虚线。

```
xq = 0:0.01:15
plot(xq,ppval(pp,xq))
```

```
xline(4,linestyle="--", color="k")
xline(10,linestyle="--", color="k")
```



(2). 创建包含重复段的分段多项式

创建并绘制一个具有四个区间的分段多项式，这四个区间中，两个二次多项式交替出现。

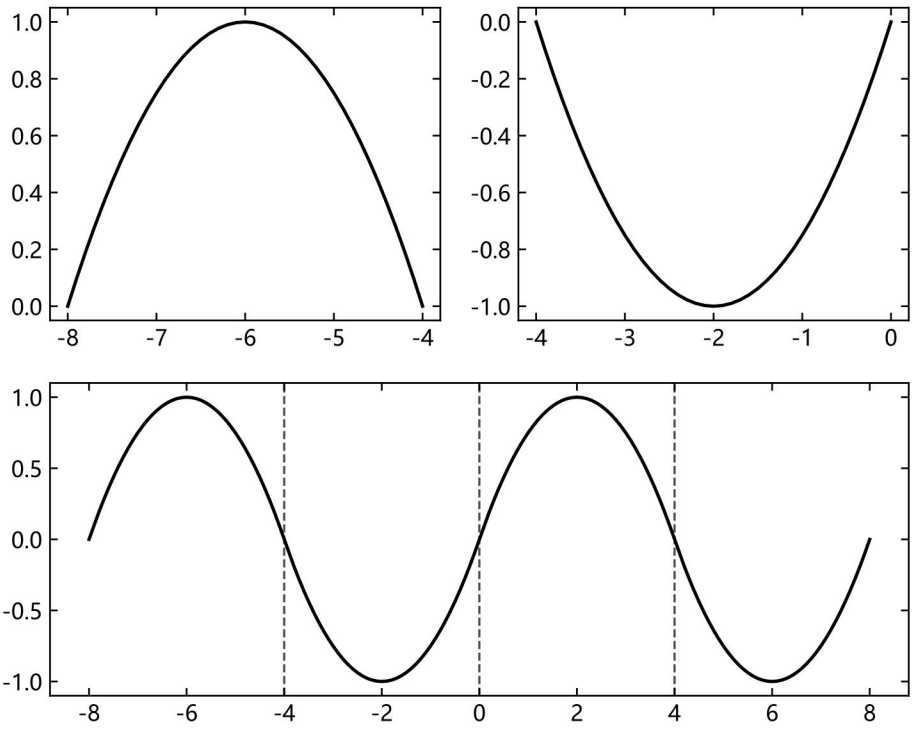
前两个子图显示了一个二次多项式在 [-8,-4] 区间上的结果以及它在 [-4,0] 区间上的求反。多项式为

$$1-(\frac{x}{2}-1)^2=-\frac{x^2}{4}+x$$

第三个子图显示将这两个二次多项式扩展到四个区间形成的分段多项式。添加垂直线是为了显示多项式汇合的点。

```
subplot(2,2,1)
cc = [-1/4 1 0];
pp1 = mkpp([-8 -4],cc);
xx1 = -8:0.1:-4;
plot(xx1,ppval(pp1,xx1),"k-")
subplot(2,2,2)
pp2 = mkpp([-4 0],-cc);
```

```
xx2 = -4:0.1:0;
plot(xx2,ppval(pp2,xx2),"k-")
subplot(2,1,2)
pp = mkpp([-8 -4 0 4 8],[cc;-cc;cc;-cc]);
xx = -8:0.1:8;
plot(xx,ppval(pp,xx),"k-")
hold("on")
xline(-4,color = [0, 0.4470, 0.7410],linestyle = "--")
xline(0,color = [0, 0.4470, 0.7410] , linestyle = "--")
xline(4,color = [0, 0.4470, 0.7410],linestyle = "--")
hold("off")
```



3.2.4.1.19 unmkpp

提取分段多项式详细信息

A. 语法

```
breaks,coefs,L,order,dim = unmkpp(pp)
```

B. 说明

breaks,coefs,L,order,dim = unmkpp(pp)从分段多项式结构体 pp 的字段中提取信息。

C. 示例

(1). 提取分段多项式信息

为区间 [0 3] 内的多项式$f(x) = x^2 + x + 1$创建分段多项式结构体，然后从该结构体的字段中提取信息。

```
pp = mkpp([0 3],[1 1 1])
```

```
pp = MKPP
form: String "pp"
breaks: Array{Int64}((1, 2)) [0 3]
coefs: Array{Int64}((1, 3)) [1 1 1]
pieces: Int64 1
order: Int64 3
dim: Int64 1
```

```
breaks,coefs,L,order,dim = unmkpp(pp)
```

```
breaks = 1×2 Matrix{Int64}:
0  3
coefs = 1×3 Matrix{Int64}:
1  1  1
L = 1
order = 3
dim = 1
```

3.2.4.1.20 ppval

计算分段多项式

A. 语法

```
v = ppval(pp,xq)
```

B. 说明

v = ppval(pp,xq) 在查询点 xq 处计算分段多项式 pp。

C. 示例

(1). 创建包含多阶多项式的分段多项式

创建一个分段多项式，它在区间 [0,4] 内具有三次多项式，在区间 [4,10] 内具有二次多项式，在区间 [10,15] 内具有四次多项式。

```
breaks = [0 4 10 15];
coefs = [0 1 -1 1 1; 0 0 1 -2 53; -1 6 1 4 77];
pp = mkpp(breaks,coefs)
```

```
ans = MKPP("pp", [0 4 10 15], [0 1 ⋯ 1 1; 0 0 ⋯ -2 53; -1 6 ⋯ 4 77],
3, 5, 1)
```

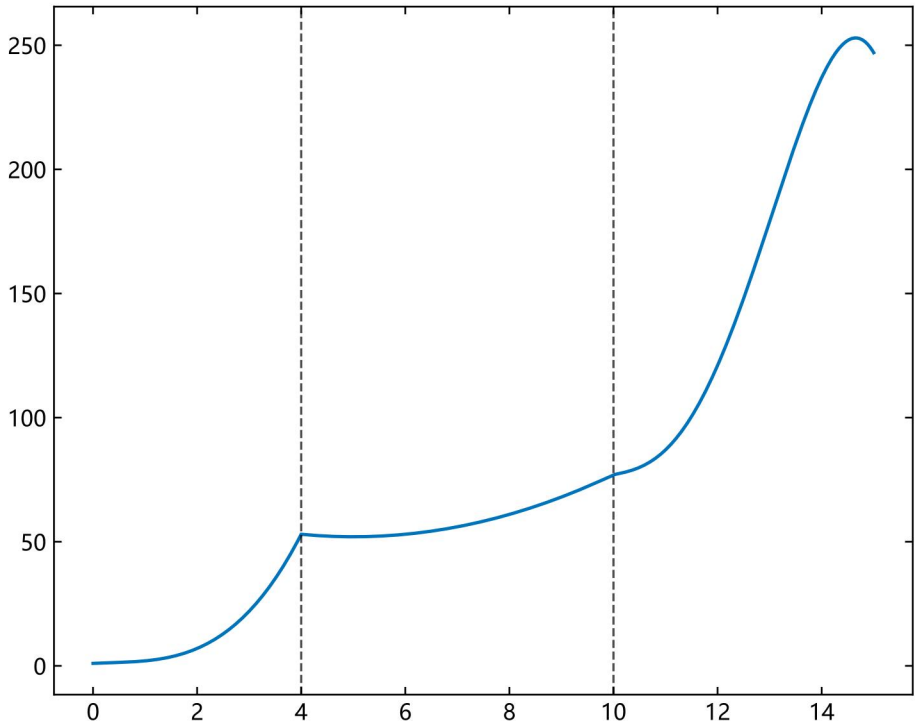
TyMath.MKPP 的结构体字段

```
fieldnames(TyMath.MKPP)
```

```
 (:form, :breaks, :coefs, :pieces, :order, :dim)
```

计算区间 [0,15] 内多个点处的分段多项式，并绘制结果图。在多项式汇合的断点处绘制垂直虚线。

```
xq = 0:0.01:15
plot(xq,ppval(pp,xq))
xline(4,linestyle="--", color="k")
xline(10,linestyle="--", color="k")
```



(2). 创建包含重复段的分段多项式

创建并绘制一个具有四个区间的分段多项式，这四个区间中，两个二次多

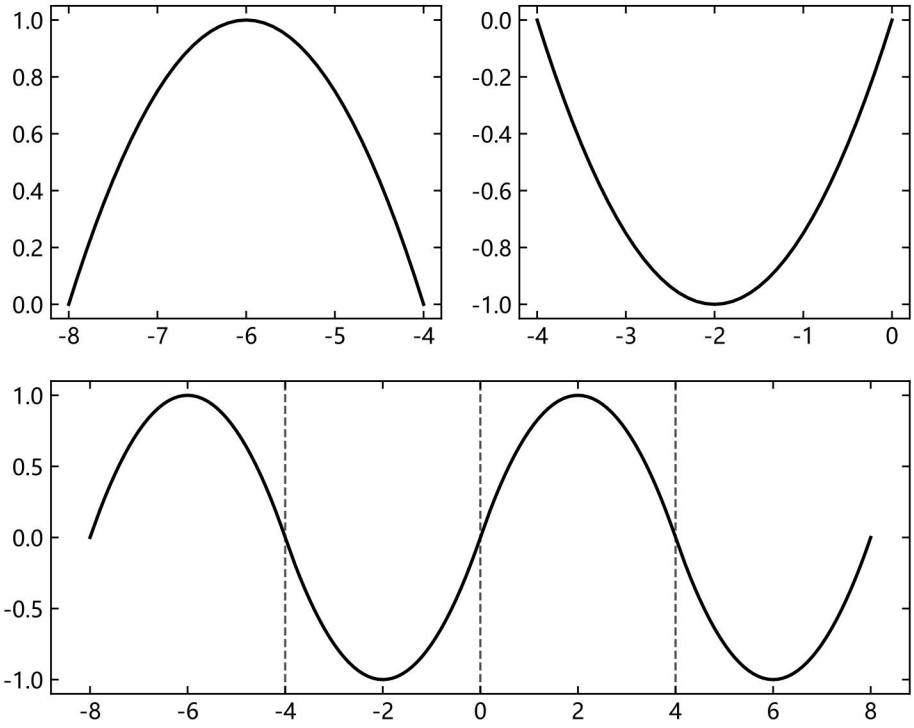
项式交替出现。

前两个子图显示了一个二次多项式在 [-8,-4] 区间上的结果以及它在 [-4,0] 区间上的求反。多项式为 $1-(\frac{x}{2}-1)^2=-\frac{x^2}{4}+x$

第三个子图显示将这两个二次多项式扩展到四个区间形成的分段多项式。添加垂直线是为了显示多项式汇合的点。

```
subplot(2,2,1)
cc = [-1/4 1 0];
pp1 = mkpp([-8 -4],cc);
xx1 = -8:0.1:-4;
plot(xx1,ppval(pp1,xx1),"k-")

subplot(2,2,2)
pp2 = mkpp([-4 0],-cc);
xx2 = -4:0.1:0;
plot(xx2,ppval(pp2,xx2),"k-")
subplot(2,1,2)
pp = mkpp([-8 -4 0 4 8],[cc;-cc;cc;-cc]);
xx = -8:0.1:8;
plot(xx,ppval(pp,xx),"k-")
hold("on")
xline(-4,color = [0, 0.4470, 0.7410],linestyle = "--")
xline(0 ,color = [0, 0.4470, 0.7410], linestyle = "--")
xline(4,color = [0, 0.4470, 0.7410],linestyle = "--")hold("off")
```



3.2.4.1.21 padecoef

时滞的 Padé 逼近

A. 语法

```
num,den = padecoef(T,N)
```

B. 说明

num,den = padecoef(T,N) 以传递函数形式返回持续时滞 exp(-T*s) 的 N 阶 Pade 逼近。行向量 num 和 den 包含了按 s 的降幂排序的分子和分母系数。分子和分母均为 N 次多项式。

C. 示例

使用 padecoef 估计 e^{-2s}的二阶值。

```
a,b = padecoef(2,2)
```

```
a = 1×3 Matrix{Int64};
1 -3  3
```

```
b = 1×3 Matrix{Int64}:
1  3  3
```

结果指示二阶近似值为

$$f(s) \approx \frac{a}{b} = \frac{s^2 - 3s + 3}{s^2 + 3s + 3}$$

将近似值与 s=0.25 处的实际值进行比较。

```
f_approx(s) = (s^2 - 3*s+3)/(s^2 + 3*s + 3)
f_actual(s) = exp(-2*s)
abs(f_approx(0.25) - f_actual(0.25))
```

```
ans = 2.6717336546910886e-5
```

3.2.4.2 网格创建

网格数据集插值简介

函数名	简介
meshgrid2	创建二维网格
meshgrid3	创建三维网格
ndgrid	N 维空间中的矩形网格

3.2.4.2.1 meshgrid2

创建二维网格

A. 语法

```
X,Y = meshgrid2(x)
X,Y = meshgrid2(x,y)
```

B. 说明

X,Y = meshgrid2(x) 基于向量 x 和 y 中包含的坐标返回二维网格坐标。X 是一个矩阵，每一行是 x 的一个副本；Y 也是一个矩阵，每一列是 y 的一个副本。坐标 X 和 Y 表示的网格有 length(y) 个行和 length(x) 个列。

X,Y = meshgrid2(x) 与 X,Y = meshgrid(x,x) 相同，并返回网格大小为 length(x)×length(x) 的方形网格坐标。

C. 示例

(1). 二维网格

使用向量 x 定义的 x 坐标和向量 y 定义的 y 坐标创建二维网格坐标。

```
x = [1:3;]
y = [1:5;]
X,Y = meshgrid2(x,y)
```

```
X = 5×3 Matrix{Int64}:
1     2     3
1     2     3
1     2     3
1     2     3
1     2     3
Y = 5×3 Matrix{Int64}:
1     1     1
2     2     2
3     3     3
4     4     4
5     5     5
```

在二维网格上计算表达式$x^2 + y^2$

```
X.^2 + Y.^2
```

```
ans = 5×3 Matrix{Int64}:
2      5      10
5      8      13
10     13     18
17     20     25
26     29     34
```

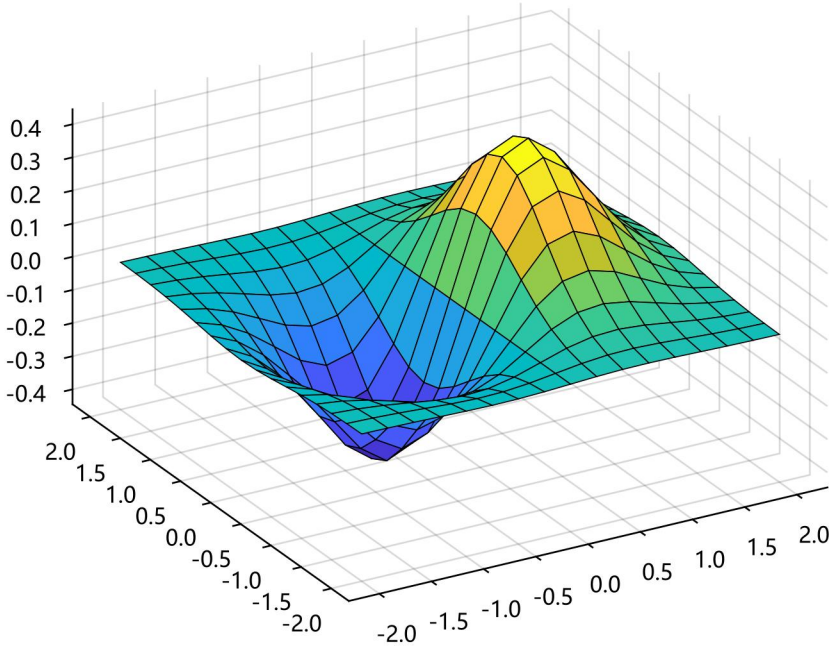
(2). 绘制曲面图

使用均匀分布的 x 坐标和 y 坐标在区间 [-2,2] 内创建二维网格。

```
x = -2:0.25:2
y = x
X,Y = meshgrid2(x)
```

在二维网格上计算并绘制函数 $f(x,y) = xe^{-x^2-y^2}$

```
F = X .* exp.(-X.^2-Y.^2)
surf(X,Y,F)
```



3.2.4.2.2 meshgrid3

创建三维网格

A. 语法

```
X,Y,Z = meshgrid3(x)
X,Y,Z = meshgrid3(x,y)
```

B. 说明

X,Y,Z = meshgrid3(x) 返回由向量 x、y 和 z 定义的三维网格坐标。X、Y 和 Z 表示的网格的大小为 length(y)×length(x)×length(z)。

X,Y,Z = meshgrid3(x) 与 X,Y,Z = meshgrid3(x,x,x) 相同，并返回网格大小为 length(x)×length(x)×length(x) 的三维网格坐标。

C. 示例

(1). 三维网格

在区间 [0,6] 内使用定义的 x、y 和 z 坐标创建三维网格坐标，并计算表达式 $x^2+y^2+z^2$

```
x = [0:2:6;]
y = [0:1:6;]
z = [0:3:6;]
X,Y,Z = meshgrid3(x,y,z)
F = X .^ 2 + Y .^ 2 + Z .^ 2
```

确定网格的大小。三个坐标向量具有不同的长度，构成一个网格点矩形框。

```
gridsize = size(F)
```

```
 girdsize = (7, 4, 3)
```

使用单输入语法，基于 x 中定义的坐标生成均匀分布的三维网格。新网格构成一个网格点正方体。

```
X,Y,Z = meshgrid3(x)
G = X .^ 2 + Y .^ 2 + Z .^ 2
gridsize = size(G)
```

```
 girdsize = (4, 4, 4)
```

3.2.4.2.3 ndgrid

N 维空间中的矩形网格

A. 语法

```
X1,X2,...,Xn = ndgrid(x1,x2,...,xn)
X1,x2,...,Xn = ndgrid(xg,dims = n)
```

B. 说明

X1,X2,...Xn = ndgrid(x1,x2,...,xn)复制网格向量 x1,x2,...,xn 以生成 n 维满网格。

X1,X2,...,Xn = nggrid(xg,dims = n)指定对所有维度使用单一网格向量 xg。dims 决定输出的维度 n 。

C. 示例

(1). 创建二维网格

根据向量 [1:2:19;] 和 [2:2:12;] 创建一个二维网格。

```
X,Y = ndgrid([1:2:19;],[2:2:12;])
```

```
X = 10×6 Matrix{Int64}:
1    1    1    1    1    1
3    3    3    3    3    3
5    5    5    5    5    5
7    7    7    7    7    7
9    9    9    9    9    9
11   11   11   11   11   11
13   13   13   13   13   13
15   15   15   15   15   15
17   17   17   17   17   17
19   19   19   19   19   19
Y = 10×6 Matrix{Int64}:
2    4    6    8    10    12
2    4    6    8    10    12
2    4    6    8    10    12
2    4    6    8    10    12
2    4    6    8    10    12
2    4    6    8    10    12
2    4    6    8    10    12
2    4    6    8    10    12
2    4    6    8    10    12
2    4    6    8    10    12
```

(2). 在网格域上计算函数

计算并绘制函数

$$x_1 e^{-x_1^2 - x_2^2}$$

在网格域 $-2 < x_1 < 2$ 和 $-2 < x_2 < 2$ 。

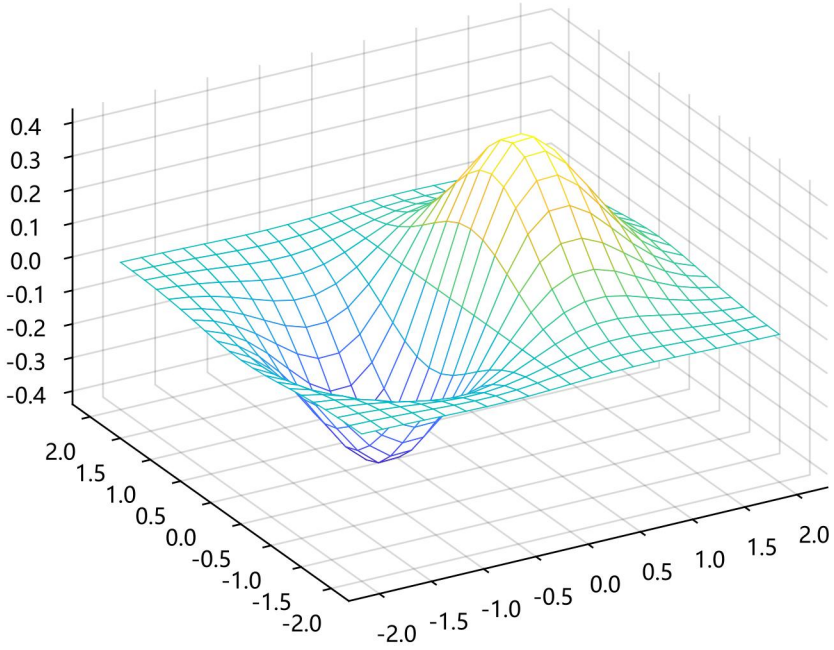
```
X1,X2 = ndgrid(-2:0.2:2,dims = 2)
```

在域上计算函数。

```
Z = X1 .* exp.(-X1.^2 - X2.^2)
```

生成函数的网格图。

```
mesh(X1,X2,Z)
```



(3). 对数据插值

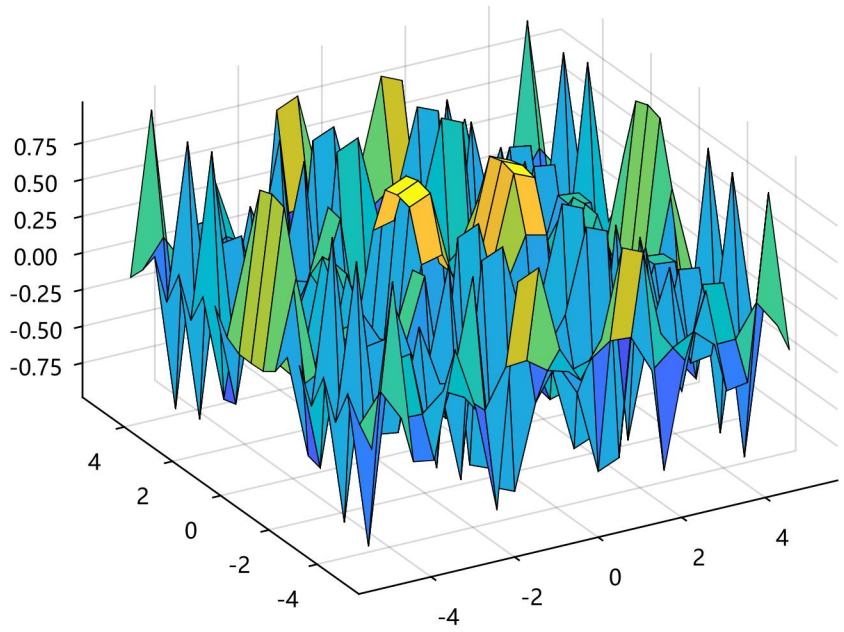
创建一个二维网格并计算网格上的一些函数值。在指定的值之间插值以细化网格。

在范围 [-5,5]内创建(x,y)粗网格

```
X,Y = ndgrid(-5:0.5:5,dims = 2)
```

计算网格上的一些函数值并绘制函数图。

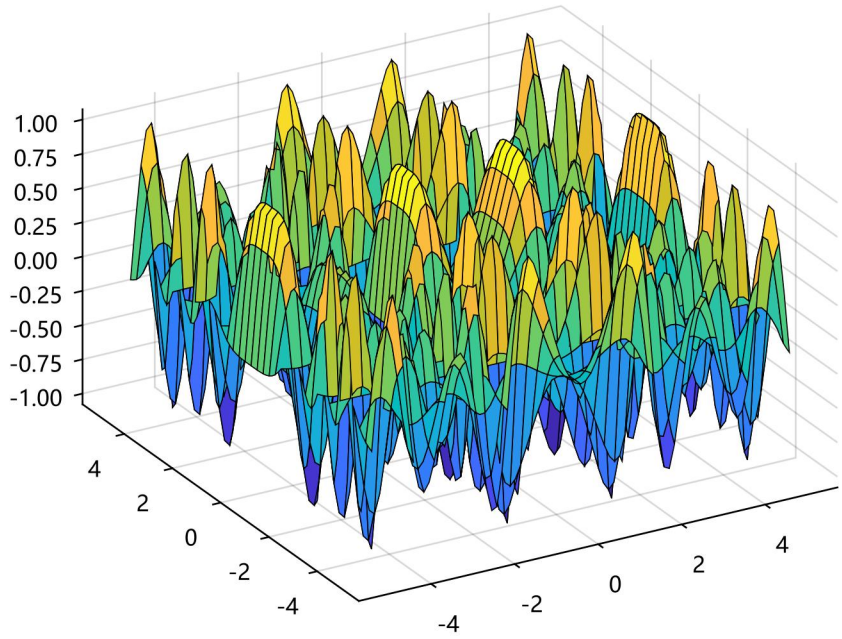
```
f = sin.(X.^2) * cos.(Y.^2)
surf(X,Y,f)
```



使用更精细的网格在点之间插值，并绘制结果图。

```
X1,Y1 = ndgrid(-5:0.125:5,dims = 2)
itp = interpolate(f,BSpline(Cubic(Line(OnGrid()))))
sitp = scale(itp,-5:0.5:5,-5:0.5:5)
F = sitp.(X1,Y1)

surf(X1,Y1,F)
```



3.2.4.3 散点插值

散点数据集插值简介

函数名	简介
scatteredInterpolant	散点数据插值
evaluate	计算插值方法在数据点处的值
griddata	对二维或三维散点数据插值

3.2.4.3.1 scatteredInterpolant

散点数据插值

A. 语法

```
itp = scatteredInterpolant(method, points, samples; metric = Euclidean(), returnRBFmatrix = false; smooth = false)
```

B. 说明

itp = scatteredInterpolant(method, points, samples; metric = Euclidean(), returnRBFmatrix = false; smooth = false) 对自变量 points 与因变量 samples 采用插值方法 method 进行插值。

C. 示例

(1). 二维插值

定义一些样本点，并计算这些位置的三角函数的值。这些点是用于插值的样本值。

```
t = LinRange((3 / 4)pi, 2pi, 50)
x = [3cos.(t); 2cos.(t); 0.7cos.(t)]
y = [3sin.(t); 2sin.(t); 0.7sin.(t)]
v = repeat([-0.5; 1.5; 2], inner = length(t))
```

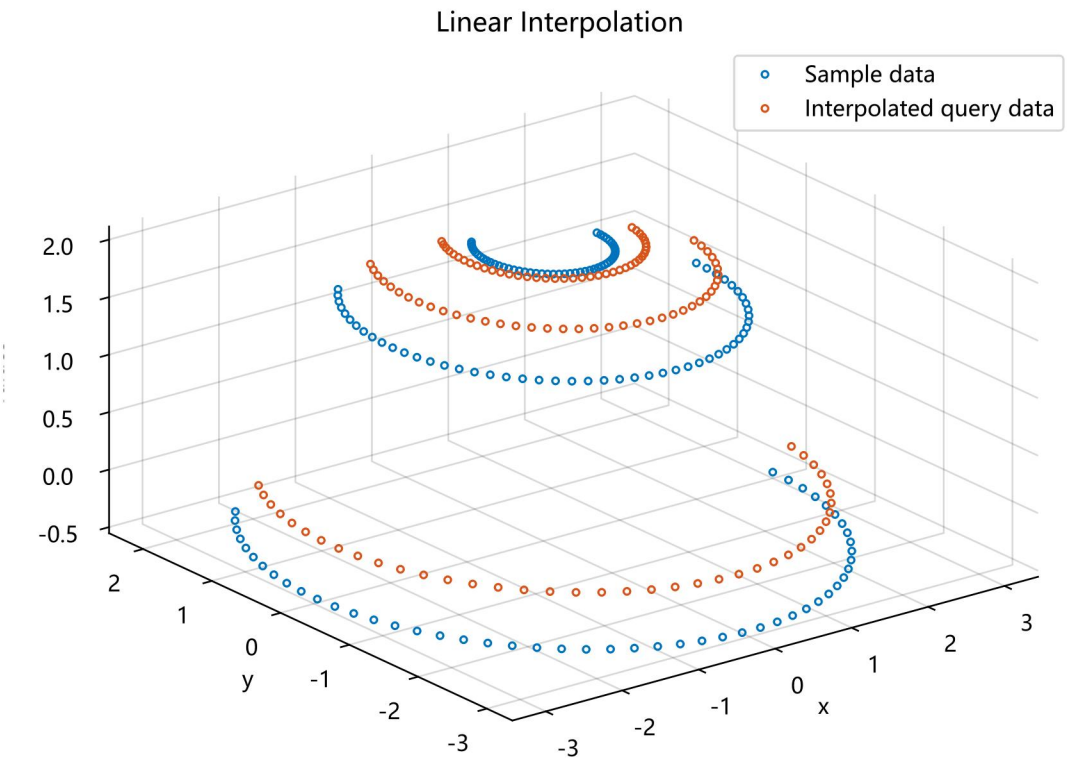
创建插值并计算位于查询位置 (xq, yq) 处的插值。

```
itp = scatteredInterpolant(Multiquadratic(), [x y]', v)

tq = LinRange(3 / 4 * pi + 0.2, 2 * pi - 0.2, 40)
xq = [2.8 * cos.(tq); 1.7 * cos.(tq); cos.(tq)]
yq = [2.8 * sin.(tq); 1.7 * sin.(tq); sin.(tq)]
vq = evaluate(itp,[xq yq]')
```

绘制结果。

```
plot3(x,y,v,".",xq,yq,vq,".")
grid("on")
title("Linear Interpolation")
xlabel("x")
ylabel("y")
zlabel("Values")legend(["Sample data","Interpolated query data"])
```



(2). 基于均匀网格对散点数据插值

基于均匀查询点网格对随机分布的散点数据插值。

对函数介于 -2.5 和 2.5 之间的 200 个随机点采样。

```
xy = -2.5 .+ 5*rand(200,2)
x = xy[:, 1]
y = xy[:, 2]
v = x .* exp.(-x .^ 2 - y .^ 2)
```

x、y 和 v 是包含分散（非均匀）样本点和数据的向量。

定义一个规则网格并基于该网格对散点数据插值。

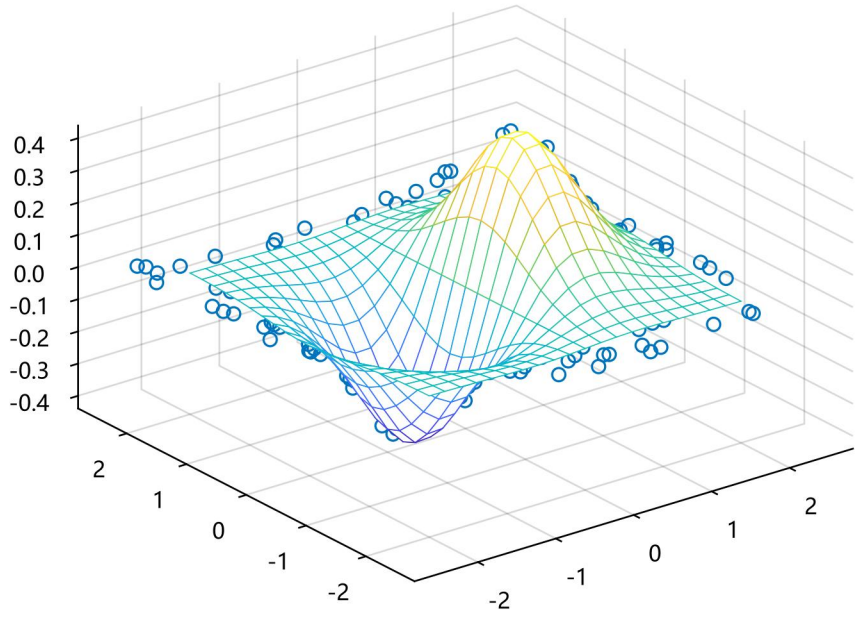
```
xq, yq = meshgrid2(-2:0.2:2)
itp = scatteredInterpolant(Multiquadratic(), [x y]', v)

vq = evaluate(itp, [xq[:] yq[:]]')
vq = reshape(vq, size(xq))
```

将网格数据绘制为网格，将散点数据绘制为点。

```
plot3(x, y, v, "o")
hold("on")
mesh(xq, yq, vq)
xlim([-2.7 2.7])
ylim([-2.7 2.7])
```

```
hold("off")
```



(3). 基于网格对四维数据集插值

对四维函数随机采样散点的三维切片插值。

对四维函数 v(x,y,z) 介于 -1 和 1 之间的 2500 个随机点采样。向量 x、y、z 包含非均匀样本点。

```
x = 2 * rand(2500, 1) .- 1
y = 2 * rand(2500, 1) .- 1
z = 2 * rand(2500, 1) .- 1
v = x .^ 2 + y .^ 3 - z .^ 4
```

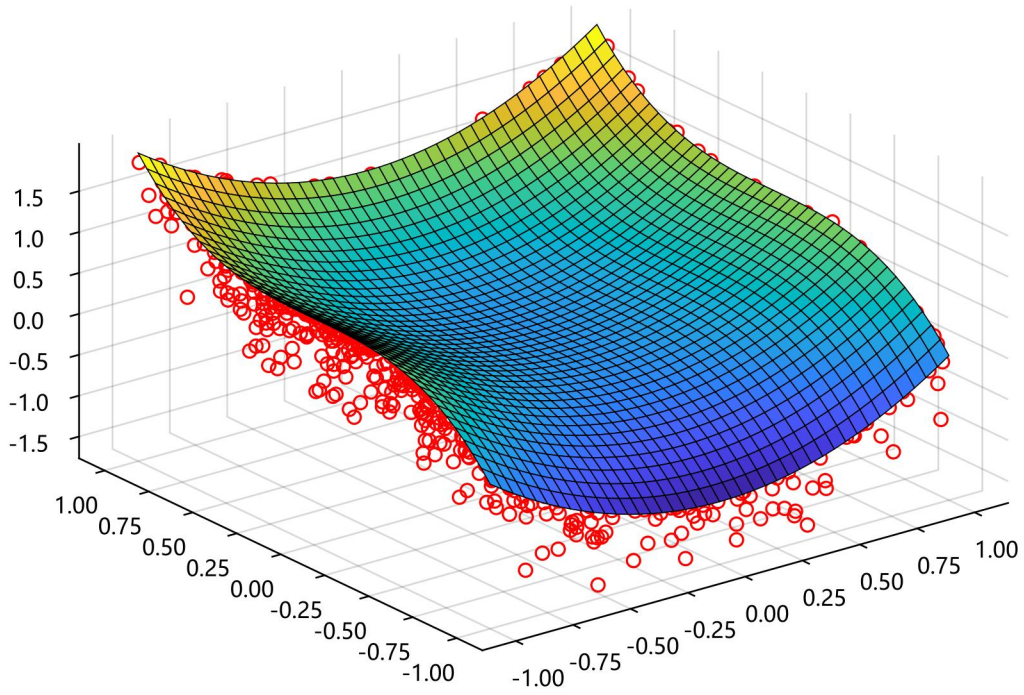
定义一个在 [-1, 1] 范围内具有 xy 个点的规则网格，并设置 z=0。在这个二维查询点网格 (xq,yq,0) 上插值将产生四维数据集 (x,y,z,v) 的三维插值切片 (xq,yq,0,vq)。

```
d = -1:0.05:1
xq, yq = meshgrid2(d, d)
zq = zero(xq)
```

基于网格对散点数据插值。绘制结果。

```
itp = scatteredInterpolant(Multiquadratic(),[x y z]',v)
vq = evaluate(itp,[xq[:] yq[:] zq[:]]')
vq = reshape(vq,size(xq))
```

```
plot3(vec(x),vec(y),vec(v),"ro")
hold("on")
surf(xq,yq,vq)
hold("off")
```



(4). 散点数据插值方法比较

比较 scatteredInterpolant 提供的几种不同插值算法的结果。

创建包含 50 个散点的样本数据集。这里有意使用较少的点数量，目的是为了突出插值方法之间的差异。

```
x = -3 .+ 6 * rand(50, 1)
y = -3 .+ 6 * rand(50, 1)
v = sin.(x) .^ 4 .* cos.(y)
```

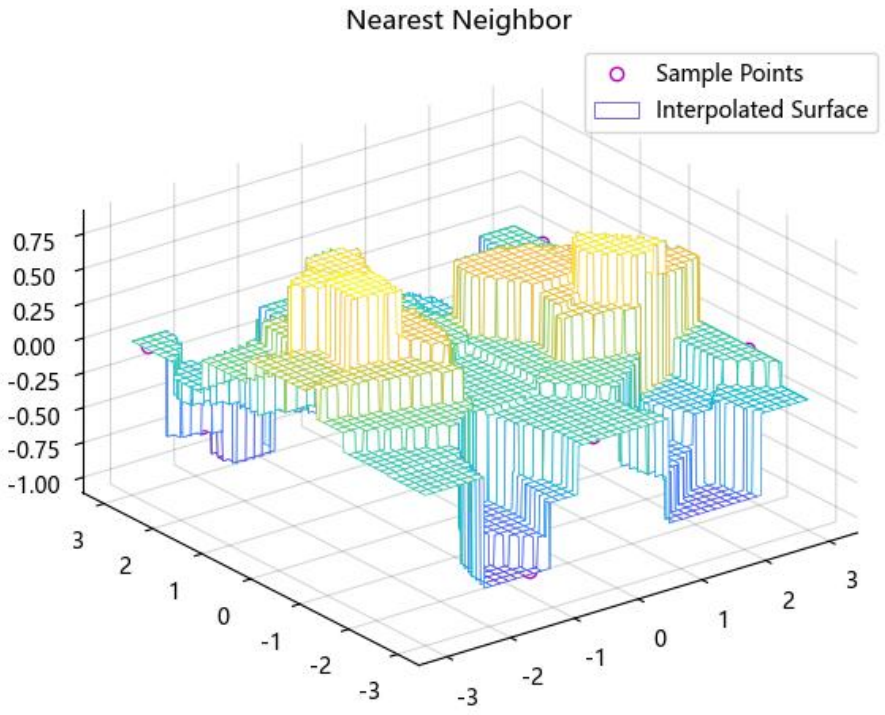
创建一个查询点网格。

```
xq, yq = meshgrid2(-3:0.01:3);
```

使用 NearestNeighbor , Multiquadratic , Shepard , Gaussian 方法进行样本数据插值。绘制结果进行比较。

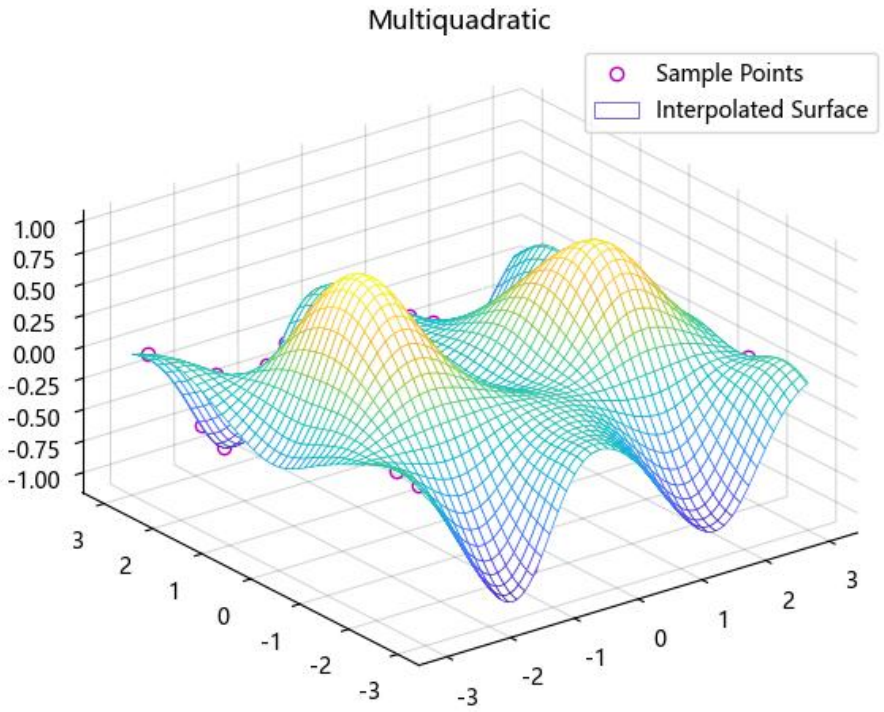
```
itp = scatteredInterpolant(NearestNeighbor(), [x y]', v)
z1 = evaluate(itp, [xq[:] yq[:]]')
z1 = reshape(z1, size(xq))
```

```
figure("NearestNeighbor")
plot3(vec(x), vec(y), vec(v), "mo")hold("on")
mesh(xq, yq, z1)
title("Nearest Neighbor")
legend(["Sample Points", "Interpolated Surface"])
hold("off")
```



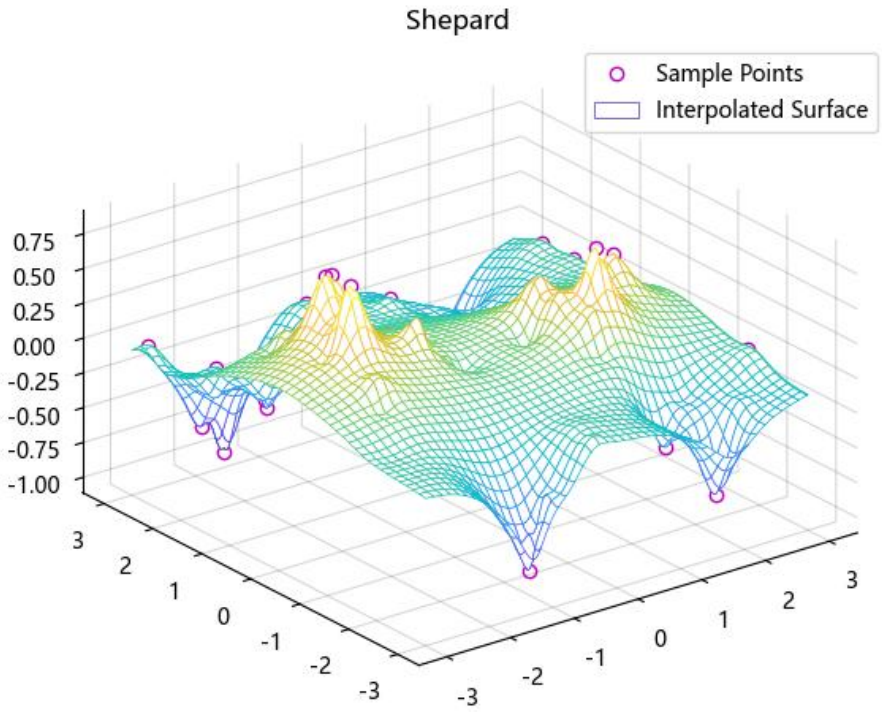

```
itp = scatteredInterpolant(Multiquadratic(), [x y]', v)
z1 = evaluate(itp, [xq[:] yq[:]]')
z1 = reshape(z1, size(xq))

figure("Multiquadratic")
plot3(vec(x), vec(y), vec(v), "mo")
hold("on")
mesh(xq, yq, z1)
title("Multiquadratic")
legend(["Sample Points", "Interpolated Surface"])
hold("off")
```



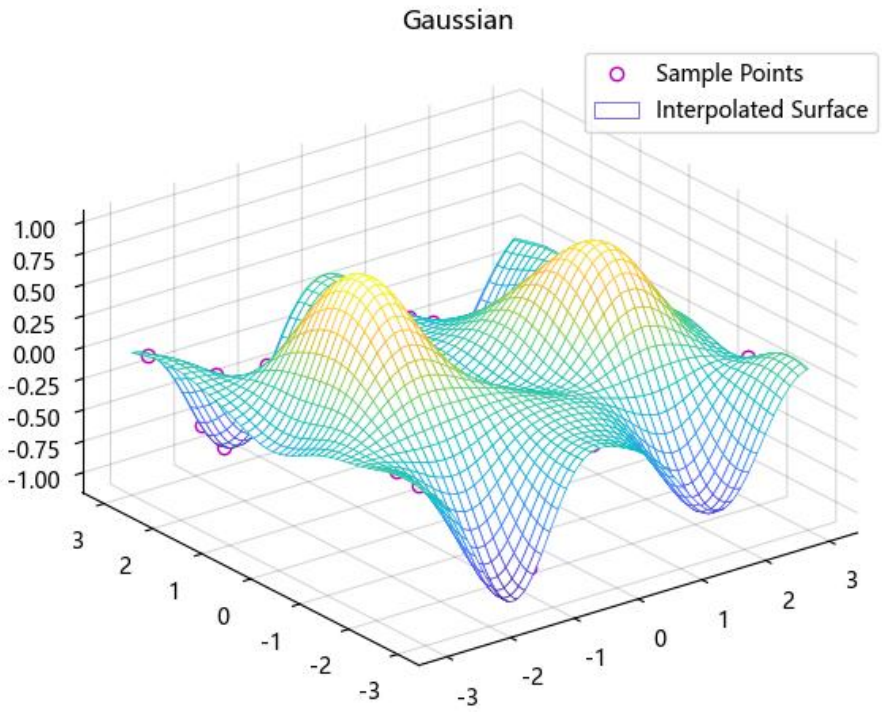

```
itp = scatteredInterpolant(Shepard(), [x y]', v)
z1 = evaluate(itp, [xq[:] yq[:]]')
z1 = reshape(z1, size(xq))

figure("Shepard")
plot3(vec(x), vec(y), vec(v), "mo")
hold("on")
mesh(xq, yq, z1)
title("Shepard")
legend(["Sample Points", "Interpolated Surface"])
hold("off")
```




```
itp = scatteredInterpolant(Gaussian(), [x y]', v)
z1 = evaluate(itp, [xq[:] yq[:]]')
z1 = reshape(z1, size(xq))

figure("Gaussian")
plot3(vec(x), vec(y), vec(v), "mo")
hold("on")
mesh(xq, yq, z1)
title("Gaussian")
legend(["Sample Points", "Interpolated Surface"])、
old("off")
```



(5). 二维外插

使用最近邻点外插查询凸包外部的单点处的插值。

定义包含 200 个随机点的矩阵并对指数函数进行采样。这些点是用于插值的样本值。

```
P = -2.5 .+ 5 * rand(200, 2)
x = P[:, 1]
y = P[:, 2]
v = x .* exp.(-x .^ 2 - y .^ 2)
```

创建插值，使用最近邻点插值。

```
itp = scatteredInterpolant(NearestNeighbor(), [x y]', v)
```

计算凸包外部的插值。

```
evaluate(itp,[3;-1.5])
```

```
1×1 Matrix{Float64}:
0.0014478706282796485
```

3.2.4.3.2 evaluate

计算插值方法在数据点处的值

A. 语法

```
evaluate(itp, points)
```

B. 说明

evaluate(itp, points) 计算插值方法 itp 在给定数据点 points 处的对应值。

C. 示例

(1). 基于均匀网格对散点数据插值

基于均匀查询点网格对随机分布的散点数据插值。

对函数介于 -2.5 和 2.5 之间的 200 个随机点采样。

```
xy = -2.5 .+ 5*rand(200,2)
x = xy[:, 1]
y = xy[:, 2]
v = x .* exp.(-x .^ 2 - y .^ 2)
```

x、y 和 v 是包含分散（非均匀）样本点和数据的向量。

定义一个规则网格并基于该网格对散点数据插值，得到插值方法 itp。

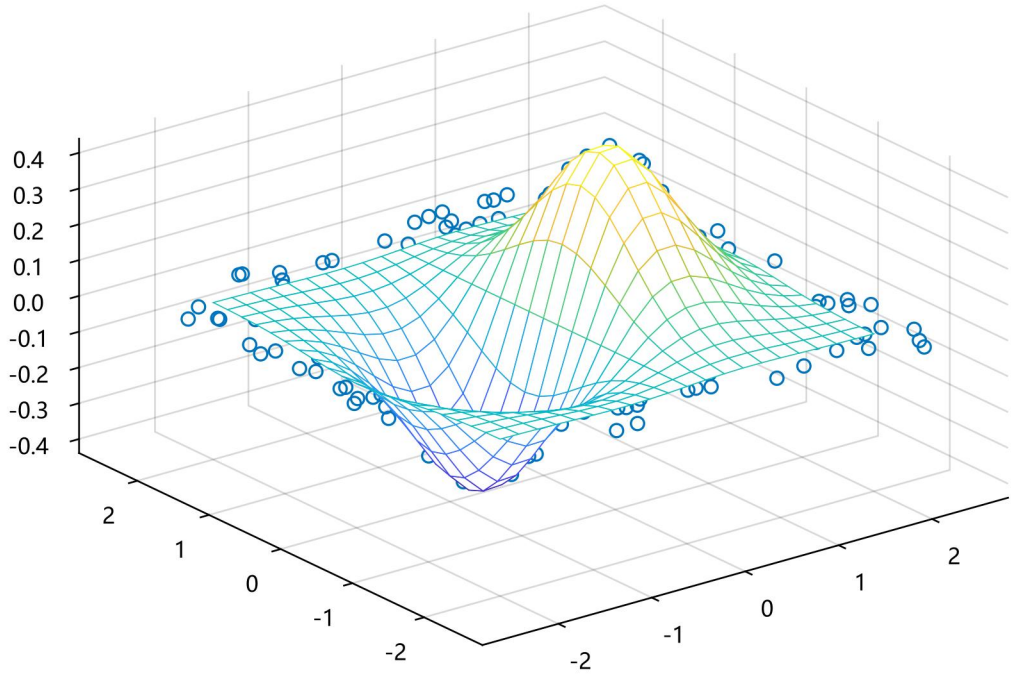
```
xq, yq = meshgrid2(-2:0.2:2)
itp = scatteredInterpolant(Multiquadratic(), [x y]', v)
```

使用 evaluate 函数计算插值方法 itp 在自变量 (xq,yq) 处对应的值。

```
vq = evaluate(itp, [xq[:] yq[:]]') # 观测点
vq = reshape(vq, size(xq))
```

将网格数据绘制为网格，将散点数据绘制为点。

```
plot3(x, y, v, "o")
hold("on")
mesh(xq, yq, vq)
xlim([-2.7 2.7])
ylim([-2.7 2.7])
hold("off")
```



3.2.4.3.3 griddata

对二维或三维散点数据插值

A. 语法

```
vq = griddata(x,y,v,xq,yq)
vq = griddata(x,y,z,v,xq,yq,zq)
vq = griddata(___,method)
```

B. 说明

vq = griddata(x,y,v,xq,yq) 使 v = f(x,y) 形式的曲面与向量 (x,y,v) 中的散点数据拟合。griddata 函数在 (xq,yq) 指定的查询点对曲面进行插值并返回插入的值 vq。曲面始终穿过 x 和 y 定义的数据点。示例

vq = griddata(x,y,z,v,xq,yq,zq) 拟合 v = f(x,y,z) 形式的超曲面。示例

vq = griddata(___,method) 使用上述语法中的任何输入参数指定计算 vq 所用的插值方法。method 可以是 "linear"、"nearest"、"natural" 或 "v4"。默认方法为 "linear"。。

C. 示例

(1). 基于均匀网格对散点数据插值

基于均匀查询点网格对随机分布的散点数据插值。

对函数介于 -2.5 和 2.5 之间的 200 个随机点采样。

```
rng=MT19937ar(5489)
xy = -2.5 .+ 5*rand(rng,200,2);
x = xy[:,1];
y = xy[:,2];
v = x.*exp.(-x.^2-y.^2);
```

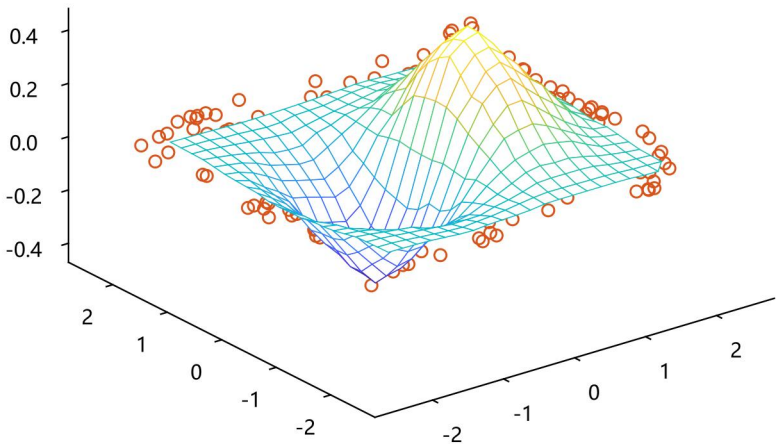
x、y 和 v 是包含分散（非均匀）样本点和数据的向量。

定义一个规则网格并基于该网格对散点数据插值。

```
xq,yq = meshgrid2(-2:0.2:2, -2:0.2:2);
vq = griddata(x,y,v,xq,yq);
```

将网格数据绘制为网格，将散点数据绘制为点。

```
mesh(xq,yq,vq)
hold("on")
plot3(x,y,v,"o")
xlim([-2.7,2.7])
ylim([-2.7,2.7])
```



(2). 基于网格对四维数据集插值

对四维函数随机采样散点的三维切片插值。

对四维函数 v(x,y,z) 介于 -1 和 1 之间的 2500 个随机点采样。向量 x、y、z 包含非均匀样本点。

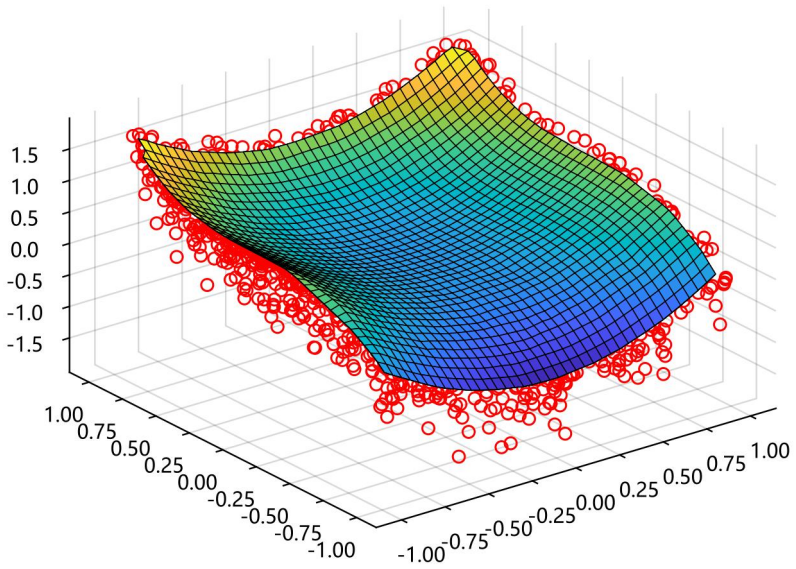
```
rng=MT19937ar(5489)
x = 2*rand(rng,2500) .- 1;
y = 2*rand(rng,2500) .- 1;
z = 2*rand(rng,2500) .- 1;
v = x.^2 .+ y.^3 .- z.^4;
```

定义一个在 [-1, 1] 范围内具有 xy 个点的规则网格，并设置 z=0。在这个二维查询点网格 (xq,yq,0) 上插值将产生四维数据集 (x,y,z,v) 的三维插值切片 (xq,yq,0,vq)。

```
d = -1:0.05:1;
xq,yq = meshgrid2(d,d);
zq=zeros(size(xq));
```

基于网格对散点数据插值。绘制结果。

```
vq = griddata(x,y,z,v,xq,yq,zq);
plot3(x,y,v,"ro")
hold("on")
surf(xq,yq,vq)
```



(3). 散点数据插值方法比较

比较 griddata 提供的几种不同插值算法的结果。

创建包含 50 个散点的样本数据集。这里有意使用较少的点数量，目的是为了突出插值方法之间的差异。

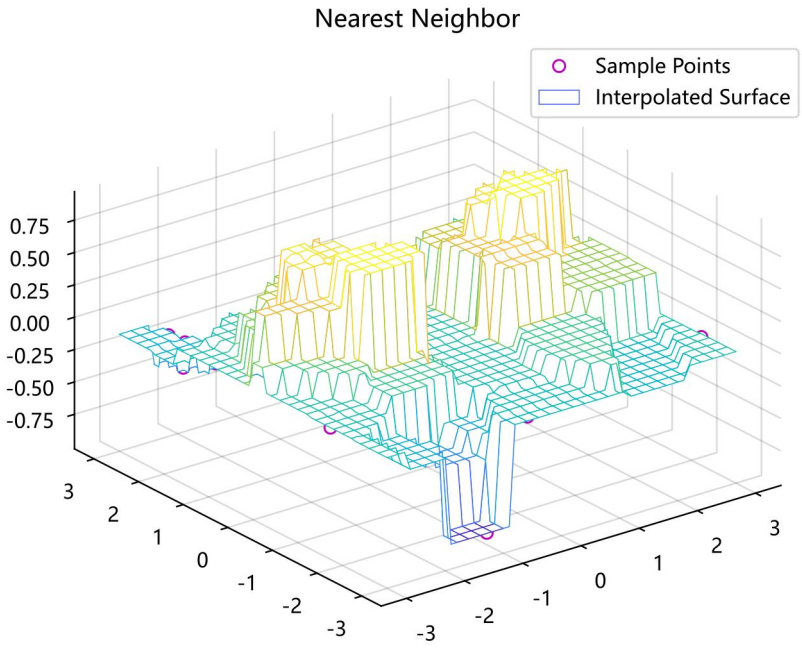
```
rng=MT19937ar(1)
x = -3 .+ 6*rand(rng,50);
y = -3 .+ 6*rand(rng,50);
v = sin.(x).^4 .* cos.(y);
```

创建一个查询点网格。

```
xq,yq = meshgrid2(-3:0.1:3,-3:0.1:3);
```

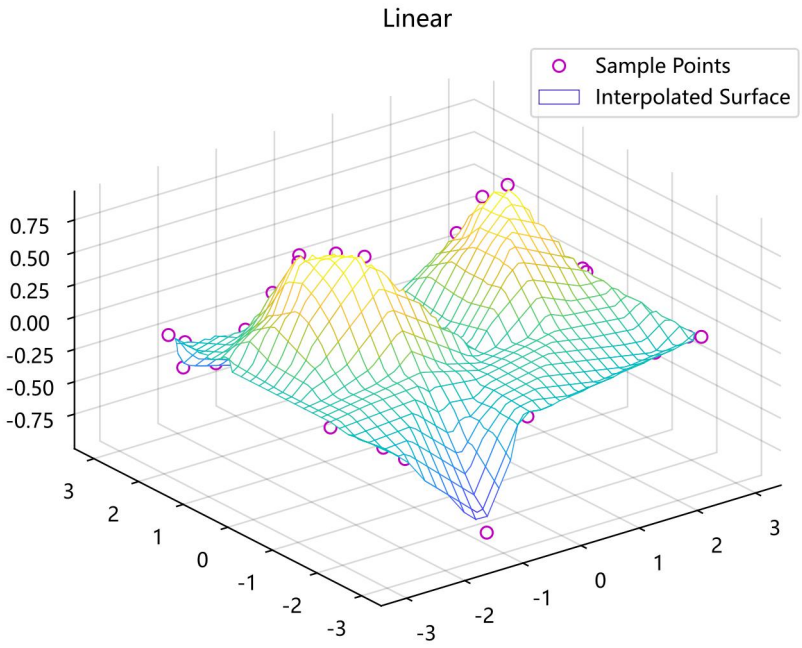
使用 "nearest"、"linear" 和 "natural" 方法进行样本数据插值。绘制结果进行比较。

```
z1 = griddata(x,y,v,xq,yq,"nearest");
plot3(x,y,v,"mo")
hold("on")
mesh(xq,yq,z1)
title("Nearest Neighbor")
legend("Sample                                    Points","Interpolated
Surface","Location","NorthWest")
```

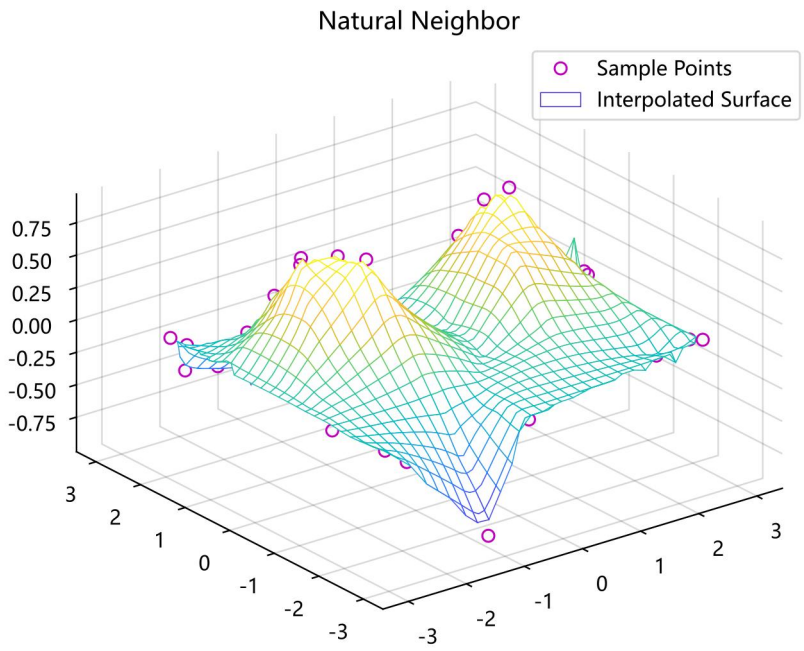



```
z2 = griddata(x,y,v,xq,yq,"linear");
figure()
plot3(x,y,v,"mo")
```

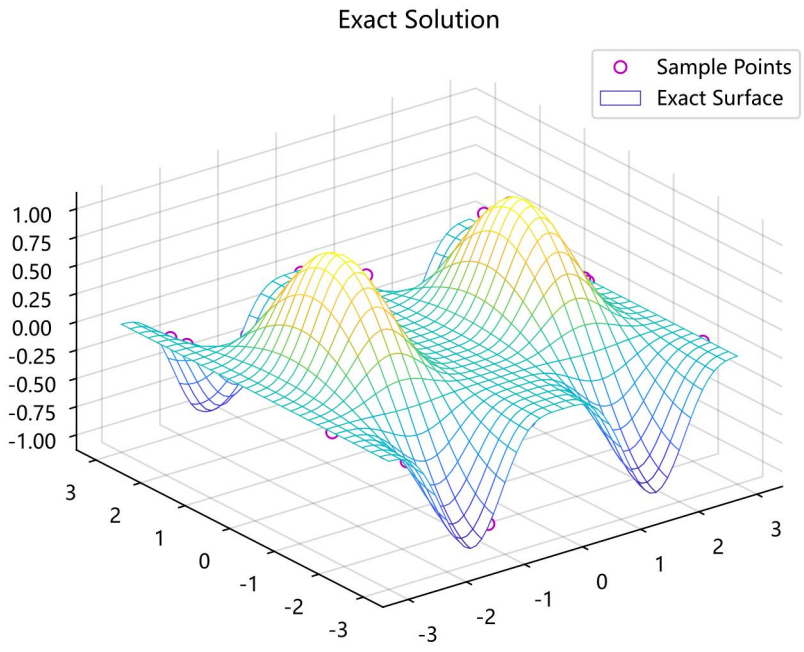
```
hold("on")
mesh(xq,yq,z2)
title("Linear")
legend("Sample                                    Points","Interpolated
Surface","Location","NorthWest")
```




```
z3 = griddata(x,y,v,xq,yq,"natural");
figure()
plot3(x,y,v,"mo")
hold("on")
mesh(xq,yq,z3)
title("Natural Neighbor")
legend("Sample                                    Points","Interpolated
Surface","Location","NorthWest")
```




```
figure()
plot3(x,y,v,"mo")
hold("on")
mesh(xq,yq,sin.(xq).^4 .* cos.(yq))
title("Exact Solution")
legend("Sample Points","Exact Surface","Location","NorthWest")
```



3.2.5 数值积分和微分方程

数值积分、常微分方程、时滞微分方程、边界值问题、偏微分方程

Syslab 中的微分方程解器涵盖了工程和科学上的一系列用途，同时，也有用于解常微分方程初哈值问题或边界值问题的求解器，以及用于时滞做分方程和偏微分方程的求解器。

3.2.5.1 微分方程定义

微分方程简介与用法

函数名	简介
Begin Your First ODE	微分方程图像应用
ODEProblem	定义 ODE 问题
DynamicalODEProblem	定义动力学常微分方程
SecondOrderODEProblem	定义 二阶 ODE 问题
HamiltonianProblem	相应的哈密顿量定义运动方程的简单方法
SplitODEProblem	定义 拆分 ODE 问题
DAEProblem	定义微分代数方程问题

3.2.5.1.1 Begin Your First ODE

接下来的代码您可以直接复制到 Syslab 中运行并得到有趣的微分方程图像

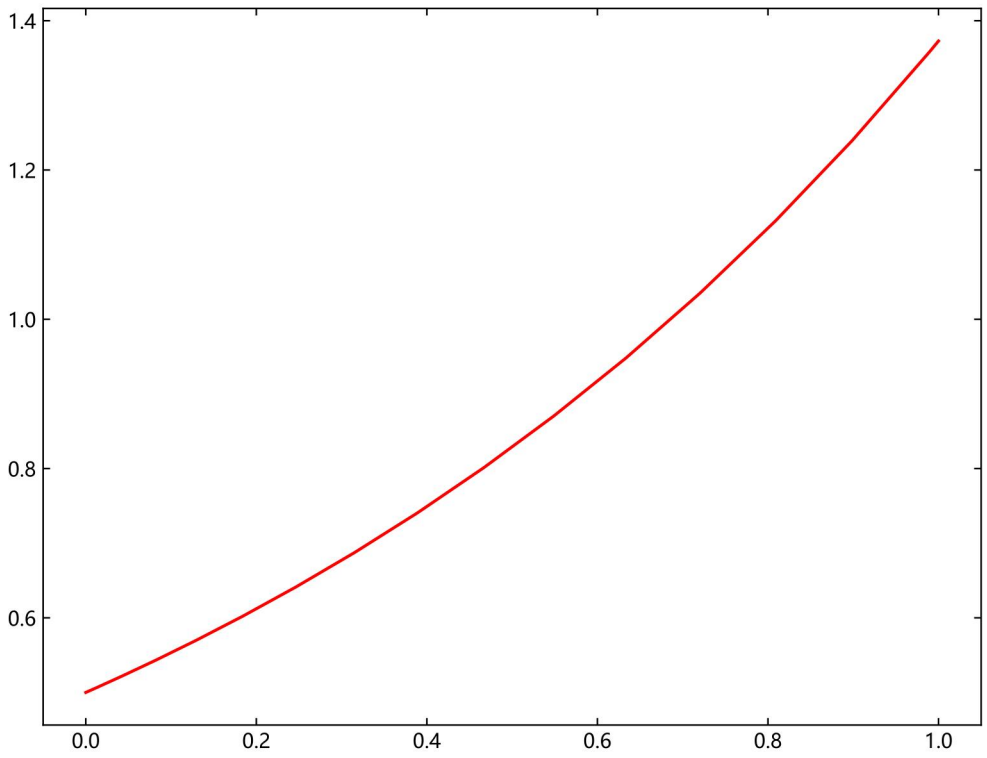
A. 一阶微分方程公式

公式

$$u' = 1.01u$$

代码

```
using DifferentialEquations
f(u,p,t)=1.01*u
u0=1/2
tspan=(0.0,1.0)
prob=ODEProblem(f,u0,tspan)
sol=solve(prob,Tsit5(),reltol=1e-8,abstol=1e-8)
plot(sol.t,sol.u,"r")
```



B. Apollp 卫星运行轨迹数学公式

公式

$$r_1 = \sqrt{(x_1+\lambda)^2 + x_3^2}$$

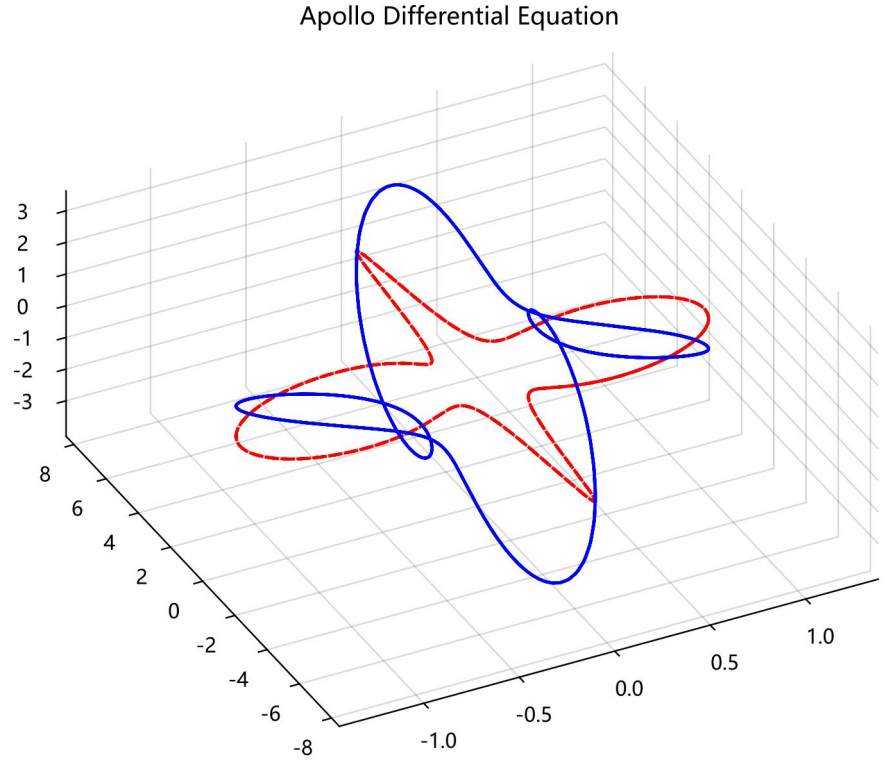
$$r_2 = \sqrt{(x_1+\lambda-1)^2 + x_3^2}$$

$$\begin{cases} \dfrac{dx_1}{dt} = x_2 \\ \dfrac{dx_2}{dt} = 2x_4 + x_1 - \dfrac{(1-\lambda)(x_1+\lambda)}{r_1^3} - \dfrac{\lambda(x_1+\lambda-1)}{r_2^3} \\ \dfrac{dx_3}{dt} = x_4 \\ \dfrac{dx_4}{dt} = -2x_2 + x_3 - (1-\lambda)\dfrac{x_3}{r_1^3} - \dfrac{\lambda x_3}{r_2^3} \end{cases}$$

代码

```
function apolloeq(dx,x,p,t)
    m=1/82.45
```

```
    mm=1-m
    r=[sqrt((x[1]+m)^2+x[3]^2),sqrt((x[1]-mm)^2+x[3]^2)]
    dx[1]=x[2]
    dx[2]=2*x[4]+x[1]-mm*(x[1]+m)/(r[1]^3)-m*(x[1]-mm)/(r[2]^3)
    dx[3]=x[4]
    dx[4]=-2*x[2]+x[3]-mm*x[3]/r[1]^3-m*x[3]/r[2]^3
end
x0=[1.2,0.0,0.0,-1.04935751]
tspan=[0.0,20.0]
prob=ODEProblem(apolloeq,x0,tspan)
solRK4=solve(prob,RK4(),saveat=0.001,reltol=1e-16)
figure("Apollo")
plot3(solRK4[1,:],solRK4[2,:],solRK4[3,:],"--r")
title("Apollo Differential Equation")
hold("on")
plot3(solRK4[1,:],solRK4[2,:],solRK4[4,:],"b")
grid("on")
```



3.2.5.1.2 ODEProblem

定义 ODE 问题

A. 语法

```
ODEProblem(f::ODEFunction,u0,tspan,p=NullParameters();kwargs...)
ODEProblem{isinplace}(f,u0,tspan,p=NullParameters();kwargs...)
```

B. 说明

要定义 ODE 问题，您只需给出函数 f 和定义 ODE 的初始条件 u_0：

$$\frac{du}{dt} = f(u,p,t)$$

f 应指定为 f(u,p,t)或 f(du,u,p,t)，并且 u_0 应是一个 AbstractArray（或数字），其几何形状与 u 的所需几何形状相匹配。

请注意，我们不限于 u_0 的数字或向量； 允许提供 u_0 作为任意矩阵/更高维度的张量。

C. 示例

(1). 创建线性 ODE

创建常量常微分方程问题

$$\frac{du}{dt} = 1.01u$$

首先，定义方程

```
f(u,p,t)=1.01*u;
```

所需参数初值 u0,时间跨度 tspan

```
u0=0.5;
tspan=(0.0,1.0);
```

利用 ODEProblem 定义常微分方程

```
prob = ODEProblem(f,u0,tspan)
```

```
 prob = ODEProblem with uType Float64 and tType Float64. In-place:
false
timespan: (0.0, 1.0)
u0: 0.5
```

(2). 创建 rossler 方程

创建 rossler 方程

首先定义 ODE 函数

```
function rossler(dx,x,p,t)
dx[1]=-x[2]-x[3]
dx[2]=x[1]+p[1]*x[2]
```

```
dx[3]=p[1]+(x[1]-p[2])*x[3]
end
```

给出所需参数初值条件 u0 和时间跨度 tspan 以及常量参数 p

```
x0=[0.0,0.0,1e-9];
tspan=(0.0,100.0);
p=[0.2,5.7];
```

根据所给参数，定义 rossler 方程

```
prob = ODEProblem(rossler,x0,tspan,p)
```

```
 prob = ODEProblem with uType Vector{Float64} and tType Float64. In-
place: true
timespan: (0.0, 100.0)
u0: 3-element Vector{Float64}:
0.0
0.0
1.0e-9
```

查看问题相关信息

查看时间跨度

```
prob.tspan
```

```
ans = (0.0, 100.0)
```

查看初值条件

```
prob.u0
```

```
ans = 3-element Vector{Float64}:
 0.0
 0.0
 1.0e-9
```

查看常量参数

```
prob.p
```

```
p = 2-element Vector{Float64}:
  0.2
  5.7
```

3.2.5.1.3 DynamicalODEProblem

定义动力学常微分方程

A. 语法

```
DynamicalODEProblem(f::DynamicalODEFunction,v0,u0,tspan,p=NullParamet
ers(),callback=CallbackSet())
DynamicalODEProblem{isinplace}(f1,f2,v0,u0,tspan,p=NullParameters(),c
```

```
allback=CallbackSet())
```

B. 说明

所需参数：

- f1 和 f2：ODE 中的函数。
- v0 和 u0：初始条件。
- tspan：问题的时间跨度。
- p：f1 和 f2 的参数值。
- 回调：应用于每个求解器的回调 使用问题。 默认为空。

数学含义：

$$\frac{dv}{dt} = f_1(u,t)$$

$$\frac{du}{dt} = f_2(v)$$

DynamicalODEProblem(f::DynamicalODEFunction,v0,u0,tspan,p=NullParameters(),callback=CallbackSet())从 DynamicalODEFunction 定义动态 ODE 函数。

DynamicalODEProblem{isinplace}(f1,f2,v0,u0,tspan,p=NullParameters(),callback=CallbackSet())从两个函数 f1 和 f2 定义一个动态 ODE 问题。

这是一个分为两组的 Partitioned ODE，因此函数应指定为 f1(dv,v,u,p,t) 和 f2(du,v,u,p,t)（就地形式），其中 f1 与 v 无关（除非求解器指定），f2 与 u 和 t 无关。 这包括由 SecondOrderODEPproblems 引起的离散化，其中加速度函数中不使用速度，以及势能（或可以）与时间相关但动能仅取决于 v 的哈密顿量。

3.2.5.1.4 SecondOrderODEProblem

定义二阶 ODE 问题

A. 语法

```
SecondOrderODEProblem(f,du0,u0,tspan,callback=CallbackSet())
```

B. 说明

数学含义

$$u'' = f\left(u', u, p, t\right)$$

要想定义二阶常微分方程，只需给出方程 f 和初始条件 u0

f 应指定为 f(du,u,p,t)或 f(ddu,du,u,p,t)，并且 u_0 应为几何形状匹配所需的 AbstractArray（或数字）

请注意，不限于 u_0 的数字或向量； 允许提供 u_0 作为任意矩阵/更高维度的张量。

C. 示例

(1). 创建二阶 ODE

创建常量常微分方程问题

$$\frac{d^2u}{dt^2} = 1.01u + 5.0du$$

首先，定义方程

```
f__2(ddu,du,u,p,t)=1.01*u+5.0*du;
```

所需参数初值 du0,u0,时间跨度 tspan

```
u0=[0.5];
du0=[1.0];
tspan=(0.0,1.0);
```

利用 ODEProblem 定义常微分方程

```
prob = SecondOrderODEProblem(f__2,du0,u0,tspan)
```

```
ODEProblem with uType ArrayPartition{Float64, Tuple{Float64, Float64}}
and tType Float64. In-place: true
timespan: (0.0, 1.0)
u0: ([1.0], [0.5])
```

查看初始条件

```
prob.u0
```

```
ans = ([1.0], [0.5])
```

3.2.5.1.5 HamiltonianProblem

相应的哈密顿量定义运动方程的简单方法

A. 语法

```
HamiltonianProblem{T}(H,p0,q0,tspan,param=nothing;kwargs...)
```

B. 说明

HamiltonianProblem{T}(H,p0,q0,tspan,param=nothing;kwargs...)

要定义 HamiltonianProblem，只需指定 Hamiltonian：H（p,q）

C. HamiltonianProblem 结构体

名称	含义
H	返回标量的哈密顿量 H(p,q,params)。
p0	初始动量
q0	初始位置
tspan	将针对 tspan[1]≤t≤ 计算解 u(t)跨度[2]。
kwargs	其他关键字

3.2.5.1.6 SplitODEProblem

定义 拆分 ODE 问题

A. 语法

```
SplitODEProblem(f::SplitFunction,u0,tspan,p=NullParameters();kwargs...)
SplitODEProblem{isinplace}(f1,f2,u0,tspan,p=NullParameters();kwargs...)
```

B. 说明

要定义一个 SplitODEProblem，只需要给出两个方程 f1 和 f2，以及他们的初始条件 u0

所表达的数学含义是：

$$\frac{du}{dt} = f_1(u,p,t) + f_2(u,p,t)$$

f 应该被定义为 f(u,p,t)或者 f(du,u,p,t),初始条件 u0

许多分裂至少是部分线性的。 这就是等式：

$$\frac{du}{dt} = Au + f_2(u,p,t)$$

C. plitODEProblem 结构体

名称	含义
f1、f2	ODE 中的函数
u0	初始条件
tspan	时间跨度
p	问题的参数。 默认为 NullParameters
kwargs	传递给解决方案的关键字参数。

3.2.5.1.7 DAEProblem

定义微分代数方程问题

A. 语法

```
DAEProblem(f::DAEFunction,du0,u0,tspan,p=NullParameters();kwargs...)
DAEProblem{isinplace}(f,du0,u0,tspan,p=NullParameters();kwargs...)
```

B. 说明

要想定义一个微分代数方程，只需要给出定以 ODE 的函数 f 和初始条件 u0

数学含义

$$0 = f(du,u,p,t)$$

f 应该被定义为 f(du,u,p,t)或者 f(resid,du,u,p,t)不限于 u_0 的数字或向量；允许提供 u_0 作为任意矩阵/更高维度的张量。

C. 示例

(1). 创建微分代数方程

创建微分代数方程问题

$$\begin{cases} 0 = -k_1 x_1 + k_3 x_2 x_3 - \dfrac{dx_1}{dt} \\ 0 = k_1 x_1 - k_2 x_2^2 - k_3 x_2 x_3 - \dfrac{dx_2}{dt} \\ 0 = x_1 + x_2 + x_3 - 1 \end{cases}$$

首先，定义方程

```
function daetest(out,du,u,k,t)
out[1]=-k[1]*u[1]+k[3]*u[2]*u[3]-du[1]
out[2]=k[1]*u[1]-k[2]*u[2]^2-k[3]*u[2]*u[3]-du[2]
out[3]=u[1]+u[2]+u[3]-1
end
```

所需参数初值 u0

一阶导数初值 du0

时间跨度 tspan

常量参数 k

```
k=[0.04,3*10^7,10^4]
u0=[1,0,0]
du0=[0.04,0.04,0]
tspan=(0,100.0)differential_vars = [true,true,false]
```

利用 DAEProblem 定义常微分方程

```
prob=DAEProblem(daetest,du0,u0,tspan,k,differential_vars=differential
_vars)
```

```
prob = DAEProblem with uType Vector{Int64} and tType Float64. In-
place: true

timespan: (0.0, 100.0)

u0: 3-element Vector{Int64}:

1

0

0
```

```
du0: 3-element Vector{Float64}:

0.04

0.04

0.0
```

3.2.5.2 积分控制

初始化微分方程问题

函数名	简介
积分器结构体	积分器结构体具体状态
init	初始化微分方程问题
step!	在积分器上执行一个（成功）步骤,并覆盖原有数据
check_error	检查积分器的状态并返回返回代码之一
set_t!	将积分器的当前时间点设置为 t
set_u!	将积分器的当前状态设置为 u
savevalues!	尝试保存当前时间点的状态和时间变量，或者在适当的时候使用插值保存点
get_proposed_dt	获取下一个时间步长的建议 dt
set_proposed_dt!	为下一个时间步设置建议的 dt
add_tstop!	在时间 t 添加一个 tstop
add_saveat!	在 t 添加一个保存时间点
reinit!	以新值重新开始积分
get_du	返回 t 处的导数

3.2.5.2.1 积分器结构体

可以根据以下表格查看积分结构体并了解具体状态

A. 结构体

变量名：integrator

结构体参数	含义	用法
t	积分器当前时间变量	integrator.t
u	积分器当前值	integrator.u
p	用户提供的参数	integrator.p
opts	常见求解器选项	integrator.opts
alg	与解决方案相关的算法	integrator.alg
f	被求解函数	integrator.f
sol	解决方案当前状态	integrator.sol
tprev	前一节点时间	integrator.tprev
uprev	前一节点值	integrator.uprev
tdir	时间方向的标志	integrator.tdir

3.2.5.2.2 init

初始化微分方程问题

A. 语法

```
init(prob::SciMLBase.DEProblem, args...; kwargs...)
```

B. 说明

init(prob::SciMLBase.DEProblem, args...; kwargs...)

prob 为定义的微分方程问题

args 为相应微分方程问题的算法

kwargs 为其他选项。

C. 示例

(1). 初始化常微分方程问题

创建常量常微分方程问题

```
f_init(u,p,t)=1.01*u
u0=0.5
tspan=(0.0,1.0)prob = ODEProblem(f_init,u0,tspan)
```

```
prob = ODEProblem with uType Float64 and tType Float64. In-place:
false
timespan: (0.0, 1.0)
u0: 0.5
```

初始化问题

```
integrator=TyMath.init(prob,Tsit5())
```

```
 integrator =

t: 0.0

u: 0.5
```

```
integratoreuler=TyMath.init(prob,Euler(),dt=0.01)
```

```
integratoreuler =

t: 0.0

u: 0.5
```

由于是定步长算法 所以应给出 dt tstops

3.2.5.2.3 step!

在积分器上执行一个（成功）步骤,并覆盖原有数据

A. 语法

```
step!(integ::DEIntegrator;dt,tstop)
```

B. 说明

step!(integ::DEIntegrator，n=1)根据算法进行步进。可以设置 dt 或 tstop

C. 示例

(1). Tsit5()步进

创建常量常微分方程问题

```
f(u,p,t)=1.01*u
```

```
u0=0.5
tspan=(0.0,1.0)
prob = ODEProblem(f,u0,tspan)
```

```
prob = ODEProblem with uType Float64 and tType Float64. In-place:
false

timespan: (0.0, 1.0)

u0: 0.5
```

初始化问题

```
integrator=init(prob,Tsit5())
```

```
integrator =

t: 0.0

u: 0.5
```

步进一步

```
step!(integrator)
integrator
```

```
 integrator =

t: 0.09964258706516003

u: 0.552938681151017
```

(2). Euler()步进

创建常量常微分方程问题

```
f(u,p,t)=1.01*u
u0=0.5
tspan=(0.0,1.0)
prob = ODEProblem(f,u0,tspan)
```

```
 prob = ODEProblem with uType Float64 and tType Float64. In-place:
false
timespan: (0.0, 1.0)
u0: 0.5
```

利用 Euler 算法初始化问题

```
integrator=init(prob,Euler(),dt=0.01)
```

由于是定步长算法，需要给出 dt 或 tstop 参数

```
integrator =
t: 0.0
u: 0.5
```

步进一步

```
step!(integrator)
integrator
```

```
integrator =
t: 0.01
u: 0.50505
```

3.2.5.2.4 check_error

检查积分器的状态并返回返回代码之一

A. 语法

```
check_error(integrator)
```

B. 说明

check_error(integrator)应返回 init 或步进后所得到的参数。

C. 示例

(1). 检查初始化和步进后结果

创建常量常微分方程问题

```
f(u,p,t)=1.01*u
u0=0.5
tspan=(0.0,1.0)
prob = ODEProblem(f,u0,tspan)
```

```
 prob = ODEProblem with uType Float64 and tType Float64. In-place:
false

timespan: (0.0, 1.0)

u0: 0.5
```

初始化问题

```
integrator=init(prob,Tsit5())
```

```
integrator =

t: 0.0

u: 0.5
```

检查初始化

```
check_error(integrator)
```

```
 ans = :Success
```

步进一步

```
step!(integrator)
integrator
```

```
integrator =

t: 0.09964258706516003

u: 0.552938681151017
```

检查积分器状态

```
check_error(integrator)
```

```
 ans = :Success
```

3.2.5.2.5 set_t!

将积分器的当前时间点设置为 t

A. 语法

```
set_t!(integrator::DEIntegrator, t)
```

B. 说明

set_t!(integrator::DEIntegrator, t)将积分器的当前时间点设置为 t。且使用时需要在初始化过程中添加关键字选项 save_everystep 设置为 false。

C. 示例

(1). 设置积分器当前时间点

创建常量常微分方程问题

```
f(u,p,t)=1.01*u
u0=0.5
tspan=(0.0,1.0)
prob = ODEProblem(f,u0,tspan)
```

```
prob = ODEProblem with uType Float64 and tType Float64. In-place:
false
timespan: (0.0, 1.0)
u0: 0.5
```

初始化问题

```
integrator=init(prob,Tsit5(),save_everystep=false)
```

```
integrator =
t: 0.0
u: 0.5
```

由于需要设置当前时间，所以初始化时应关闭 save_everystep 选项

步进 dt=20

```
step!(integrator,20)
integrator
```

```
integrator =

t: 20.056456464457547

u: 3.1336778448763007e8
```

设置积分器当前时间

```
set_t!(integrator,10)
```

```
 integrator =

t: 10.0

u: 3.1336778448763007e8
```

3.2.5.2.6 set_u!

将积分器的当前状态设置为 u

A. 语法

```
set_u!(integrator::DEIntegrator, u)
```

B. 说明

set_u!(integrator::DEIntegrator, u)将积分器的当前状态设置为 u。且使用时需要在初始化过程中添加关键字选项 save_everystep 设置为 false。

C. 示例

(1). 设置积分器当前值

创建常量常微分方程问题

```
f_setu(u,p,t)=1.01*u
u0=0.5
```

```
tspan=(0.0,1.0)
prob = ODEProblem(f_setu,u0,tspan)
```

```
prob = ODEProblem with uType Float64 and tType Float64. In-place: false

timespan: (0.0, 1.0)

u0: 0.5
```

初始化问题

```
integrator=TyMath.init(prob,Tsit5(),save_everystep=false)
```

```
integrator =

t: 0.0

u: 0.5
```

由于需要设置当前时间，所以初始化时应关闭 save_everystep 选项

步进 dt=20

```
step!(integrator,20)
integrator
```

```
integrator =

t: 20.056456464457547

u: 3.1336778448763007e8
```

设置积分器当前值

```
set_u!(integrator,5.20)
integrator
```

```
integrator =

t: 20.056456464457547

u: 5.2
```

3.2.5.2.7 savevalues!

尝试保存当前时间点的状态和时间变量，或者在适当的时候使用插值保存点

A. 语法

```
savevalues!(integrator::DEIntegrator,force_save=false)
```

B. 说明

savevalues!(integrator::DEIntegrator,force_save=false)

C. 示例

(1). 保存当前时间点

创建常量常微分方程问题

```
f(u,p,t)=1.01*u
u0=0.5
tspan=(0.0,1.0)
prob = ODEProblem(f,u0,tspan)
```

```
prob = ODEProblem with uType Float64 and tType Float64. In-place: false

timespan: (0.0, 1.0)

u0: 0.5
```

初始化问题

```
integrator=init(prob,Tsit5())
```

```
integrator =

t: 0.0

u: 0.5
```

由于需要设置当前时间，所以初始化时应关闭 save_everystep 选项

步进 dt=20,并查看解决方案的当前状态

```
step!(integrator,20)
integrator.sol
```

```
ans = retcode: Default

Interpolation: specialized 4th order "free" interpolation

t: 27-element Vector{Float64}:

0.0

0.09964258706516003
```

```
0.3457024247583422

0.6776921908052249

1.0

1.4438645106861598

1.9387923512389154

2.5179178213047004

⋮

13.056456464457545

14.056456464457545

15.056456464457545

16.056456464457547

17.056456464457547

18.056456464457547

19.056456464457547

20.056456464457547

u: 27-element Vector{Float64}:

0.5

0.552938681151017

0.7089376245893466

0.9913594502399236

1.3728004409033037
```

```
2.1493289195124254

3.54320679423154

6.359483038140653

⋮

266569.48482951004

731840.9491259751

2.0091991217980415e6

5.516063450474026e6

1.5143823058426535e7

4.157591349048431e7



1.1414268219453417e8

3.1336778448763007e8
```

保存当前时间点的状态和时间变量

```
savevalues!(integrator,true)
```

```
ans = (true, true)
```

表示已存储,且存储为当前值

查看解决方案的当前状态

```
integrator.sol
```

```
ans = retcode: Default

Interpolation: specialized 4th order "free" interpolation

t: 28-element Vector{Float64}:

0.0

0.09964258706516003

0.3457024247583422
```

```
0.6776921908052249

1.0

1.4438645106861598

1.9387923512389154

2.5179178213047004

⋮

14.056456464457545

15.056456464457545

16.056456464457547

17.056456464457547

18.056456464457547

19.056456464457547

20.056456464457547

20.056456464457547

u: 28-element Vector{Float64}:

0.5

0.552938681151017

0.7089376245893466

0.9913594502399236

1.3728004409033037

2.1493289195124254
```

```
3.54320679423154

6.359483038140653

⋮

731840.9491259751

2.0091991217980415e6

5.516063450474026e6

1.5143823058426535e7

4.157591349048431e7

1.1414268219453417e8

3.133677844876
```

3.2.5.2.8 get_proposed_dt

获取下一个时间步长的建议 dt

A. 语法

```
get_proposed_dt(integrator::DEIntegrator)
```

B. 说明

get_proposed_dt(integrator::DEIntegrato)用于获取下一个可能或建议的时间步长。

C. 示例

(1). 获得可能得下一个时间步长

创建常量常微分方程问题

```
f1(u,p,t)=1.01*u
u0=0.5
tspan=(0.0,1.0)
prob = ODEProblem(f1,u0,tspan)
```

```
prob = ODEProblem with uType Float64 and tType Float64. In-place:
```

```
false

timespan: (0.0, 1.0)

u0: 0.5
```

初始化问题

```
integrator=TyMath.init(prob,Tsit5())
integrator =

t: 0.0

u: 0.5
```

步进 dt=20

```
step!(integrator,20)
integrator1 = integrator.t
integrator1 =

20.056456464457547
```

获取可能得下一个时间步长

```
get_proposed_dt(integrator)
ans = 1.0
step!(integrator)
integrator.t - integrator1
ans = 1.0
```

表明积分器根据建议步长步进

3.2.5.2.9 set_proposed_dt!

为下一个时间步设置建议的 dt

A. 语法

```
set_proposed_dt!(integrator::DEIntegrator)
set_proposed_dt!(i::DEIntegrator,i2::DEIntegrator)
```

B. 说明

set_proposed_dt!(integrator::DEIntegrator)用于设置下一个可能或建议的时间步长

set_proposed_dt!(i::DEIntegrator,i2::DEIntegrator)设置第一个参数的时间步长

以匹配第二个参数的时间步长。

C. 示例

(1). 设置下一个时间步长

创建常量常微分方程问题

```
f(u,p,t)=1.01*u
u0=0.5
tspan=(0.0,1.0)
prob = ODEProblem(f,u0,tspan)
```

```
prob = ODEProblem with uType Float64 and tType Float64. In-place: false

timespan: (0.0, 1.0)

u0: 0.5
```

初始化问题

```
integrator=init(prob,Tsit5())
```

```
integrator =

t: 0.0

u: 0.5
```

步进 dt=20

```
step!(integrator,20)
integrator1 = integrator.t
```

```
integrator1 =

20.056455263638966
```

设置下一个时间步长

```
set_proposed_dt!(integrator,0.5)
```

```
ans = 0.5
```

```
step!(integrator)
integrator.t - integrator1
```

```
ans = 0.5
```

表明积分器根据建议步长步进

(2). 设置下一个时间步长且大于建议步长

创建常量常微分方程问题

```
f(u,p,t)=1.01*u
```

```
u0=0.5
tspan=(0.0,1.0)
prob = ODEProblem(f,u0,tspan)
```

```
prob = ODEProblem with uType Float64 and tType Float64. In-place:
false

timespan: (0.0, 1.0)

u0: 0.5
```

初始化问题

```
integrator=init(prob,Tsit5())
```

```
integrator =

t: 0.0

u: 0.5
```

步进 dt=20

```
step!(integrator,20)
integrator1 = integrator.t
```

```
integrator1 =

20.056456464457547
```

获取建议下一步步长

```
get_proposed_dt(integrator)
```

```
ans = 1.0
```

设置下一点时间步长且大于 1.0

```
set_proposed_dt!(integrator,2.0)
```

```
step!(integrator)
integrator.t-integrator1
```

```
ans = 1.0
```

结果表明当设置步长大于建议步长时，将采用建议步长

(3). 传入两个积分器

创建常量常微分方程问题

```
f(u,p,t)=1.01*u
u0=0.5
tspan=(0.0,1.0)
prob = ODEProblem(f,u0,tspan)
```

```
prob = ODEProblem with uType Float64 and tType Float64. In-place:
```

```
false

timespan: (0.0, 1.0)

u0: 0.5
```

初始化问题

```
integratorTsit=init(prob,Tsit5())
integratorEuler=init(prob,Euler(),dt=0.01)
```

积分 20 次后，设置下一时间步长,并分别获取下一步期待步长

```
step!(integratorEuler,20)
step!(integratorTsit,20)
E= get_proposed_dt(integratorEuler)
T=get_proposed_dt(integratorTsit)
```

```
E = 0.01

T = 1.0
```

```
set_proposed_dt!(integratorTsit,integratorEuler)
E1= get_proposed_dt(integratorEuler)
T1=get_proposed_dt(integratorTsit)
```

```
E1 = 0.01

T1 = 0.009999999999999343
```

会根据第二个参数算法调整第一个参数的步长

3.2.5.2.10 add_tstop!

在时间 t 添加一个 tstop

A. 语法

```
add_tstop!(i::DEIntegrator,t)
```

B. 说明

add_tstop!用于在时间 t 添加一个 tstop。

C. 示例

(1). 添加 tstop

创建常量常微分方程问题

```
f(u,p,t)=1.01*u
```

```
u0=0.5
tspan=(0.0,1.0)
prob = ODEProblem(f,u0,tspan)
```

```
prob = ODEProblem with uType Float64 and tType Float64. In-place: false

timespan: (0.0, 1.0)

u0: 0.5
```

初始化问题

```
integrator=init(prob,Tsit5())
```

```
integrator =

t: 0.0

u: 0.5
```

步进 dt=20

```
step!(integrator,20)
```

```
integrator =

20.056456464457547
```

添加 tstop

```
add_tstop!(integrator,20.5);
```

```
step!(integrator)integrator
```

```
ans = t: 20.5
u: 4.904661373472997e8
```

表示添加并执行成功

3.2.5.2.11 add_saveat!

在 t 添加一个保存时间点

A. 语法

```
add_saveat!(i::DEIntegrator,t)
```

B. 说明

add_saveat!用于在时间 t 添加一个 saveat。

C. 示例

(1). 添加 saveat

创建常量常微分方程问题

```
f_1(u,p,t)=1.01*u
u0=0.5
tspan=(0.0,1.0)
prob = ODEProblem(f_1,u0,tspan)
```

```
prob = ODEProblem with uType Float64 and tType Float64. In-place:
false

timespan: (0.0, 1.0)

u0: 0.5
```

初始化问题

```
integrator=TyMath.init(prob,Tsit5())
```

```
integrator =

t: 0.0

u: 0.5
```

步进 dt=20

```
step!(integrator,20)
integrator.sol
```

integrator 中存储 27 个值

添加 saveat

```
add_saveat!(integrator,20.5);

step!(integrator)
integrator.sol
```

integrator 中存储 29 个值，添加成功

3.2.5.2.12 reinit!

以新值重新开始积分

A. 语法

```
reinit!(integrator::DEIntegrator,args...; kwargs...)
```

B. 说明

reinit!(integrator::DEIntegrator,args...; kwargs...)

prob 为定义的微分方程问题

args 为相应微分方程问题的算法

kwargs 为其他选项。

C. 示例

(1). 重新初始化常微分方程问题

创建常量常微分方程问题

```
f(u,p,t)=1.01*u
u0=0.5
tspan=(0.0,1.0)
prob = ODEProblem(f,u0,tspan)
```

```
prob = ODEProblem with uType Float64 and tType Float64. In-place: false

timespan: (0.0, 1.0)

u0: 0.5
```

初始化问题

```
integrator=init(prob,Tsit5())
```

```
integrator =

t: 0.0

u: 0.5
```

```
step!(integrator,20)integrator
```

```
ans = t: 20.056456464457547
u: 3.1336778448763007e8
```

```
reinit!(integrator)
```

```
ans =

t: 0.0

u: 0.5
```

(2). 重新初始化且更改 t0

创建常量常微分方程问题

```
f(u,p,t)=1.01*u
u0=0.5
```

```
tspan=(0.0,1.0)
prob = ODEProblem(f,u0,tspan)
```

```
prob = ODEProblem with uType Float64 and tType Float64. In-place:
false
timespan: (0.0, 1.0)
u0: 0.5
```

初始化问题

```
integrator=init(prob,Tsit5())
```

```
integrator =
t: 0.0
u: 0.5
```

```
step!(integrator,20)
integrator
```

```
ans = t: 20.056456464457547
u: 3.1336778448763007e8
```

```
reinit!(integrator,t0=0.6)
integrator
```

```
integrator =
t: 0.6
u: 0.5
```

3.2.5.2.13 get_du

返回 t 处的导数

A. 语法

```
get_du(integrator::DEIntegrator)
```

B. 说明

get_du(integrator::DEIntegrator)返回 t 处的导数。

C. 示例

(1). 获得 t 处的导数

创建常量常微分方程问题

```
f(u,p,t)=1.01*u
u0=0.5
tspan=(0.0,1.0)
prob = ODEProblem(f,u0,tspan)
```

```
prob = ODEProblem with uType Float64 and tType Float64. In-place:
false

timespan: (0.0, 1.0)

u0: 0.5
```

初始化问题

```
integrator=init(prob,Tsit5())
integrator =

t: 0.0

u: 0.5
```

步进 dt=20

```
step!(integrator,20)
integrator =

t: 20.056456464457547

u: 3.1336778448763007e8
```

获取可能得下一个时间步长

```
get_du(integrator)
ans = 3.1650146233250636e8
```

3.2.5.3 重构

微分方程重构

函数名	简介
remake	问题重构

3.2.5.3.1 remake

问题重构

A. 语法

```
remake(f::JumpProblem,prob)
```

B. 说明

许多问题在模拟计算过程中，通常不需要创建和初始化新的问题来进行计算，只需要简单地改变初始条件、时间跨度和参数向量来进行。但是需要注意的是新的跳跃问题可能会改变就问题中的一些内部数据，包括 SSA 聚合器的核心组件。因此只有生成新的问题才能应用于后续的模拟计算。 示例。

C. 示例

(1). SIR 模型的重构

```
using DifferentialEquations
rate1(u,p,t) = (0.1/1000.0)*u[1]*u[2]
function affect1!(integrator)
  integrator.u[1] -= 1
  integrator.u[2] += 1
end
jump = ConstantRateJump(rate1,affect1!)

rate2(u,p,t) = 0.01u[2]
function affect2!(integrator)
  integrator.u[2] -= 1
  integrator.u[3] += 1
end
jump2 = ConstantRateJump(rate2,affect2!)
u0    = [999,1,0]
p     = (0.1/1000,0.01)
tspan = (0.0,250.0)
dprob = DiscreteProblem(u0, tspan, p)
jprob = JumpProblem(dprob, Direct(), jump, jump2)
sol   = solve(jprob, SSAStepper())
```

我们可以改变初始条件、时间跨度和参数向量，或者重构一个 DiscreteProblem。

```
u02    = [10,1,0]
p2     = (0.1/1000,0.0)
tspan2 = (0.0,2500.0)
dprob2 = DiscreteProblem(u02,tspan2,p2)
jprob2 = remake(jprob,prob=dprob2)
sol2   = solve(jprob2,SSAStepper())
```

或直接重新给定参数。

```
jprob2 = remake(jprob,u0=u02,p=p2,tspan=tspan2)
sol2   = solve(jprob2,SSAStepper())
```

而下述的描述则会因为问题不清而出现错误。

```
jprob2 = remake(jprob,prob=dprob2)
```

可以通过更改参数和时间跨度或者通过 DiscreteProblem 使用其他的算法

3.2.5.4 求解

微分方程求解

函数名	简介
solve	微分方程问题求解

3.2.5.4.1 solve

微分方程问题求解

A. 语法

```
sol=solve(prob,alg;kwargs)
```

B. 说明

每种类型的微分方程都有自己的问题类型，从而使解算器调度到正确的方法，常见的调用解算器的用法为 sol = solve(prob,alg;kwargs)，进入命令中，通过定义的微分方程问题 prob，选择一个算法 alg（如果不选择，则使用默认值），使用关键字参数来更改解算器的属性。解算器返回一个保存解决方案 sol 所有的详细信息的解决方案对象。

我们提供了一大套用于函数求解的共同参数。这些参数适用于任何的问题类型，并且仅受特定状态的限制。许多默认值取决于算法或者源自算法的包，但是并非所有的截面都由每个算法提供。

C. 示例

(1). 解 rossler 方程

创建 rossler 方程

首先定义 ODE 函数

```
function rossler(dx,x,p,t)
```

```
dx[1]=-x[2]-x[3]
dx[2]=x[1]+p[1]*x[2]
dx[3]=p[1]+(x[1]-p[2])*x[3]
end
```

给出所需参数初值条件 u0 和时间跨度 tspan 以及常量参数 p

```
x0=[0.0,0.0,1e-9];
tspan=(0.0,100.0);
p=[0.2,5.7];
```

根据所给参数，定义 rossler 方程

```
prob = ODEProblem(rossler,x0,tspan,p)
```

```
prob = ODEProblem with uType Float64 and tType Float64. In-place: true

timespan: (0.0, 100.0)

u0: 3-element Vector{Float64}:

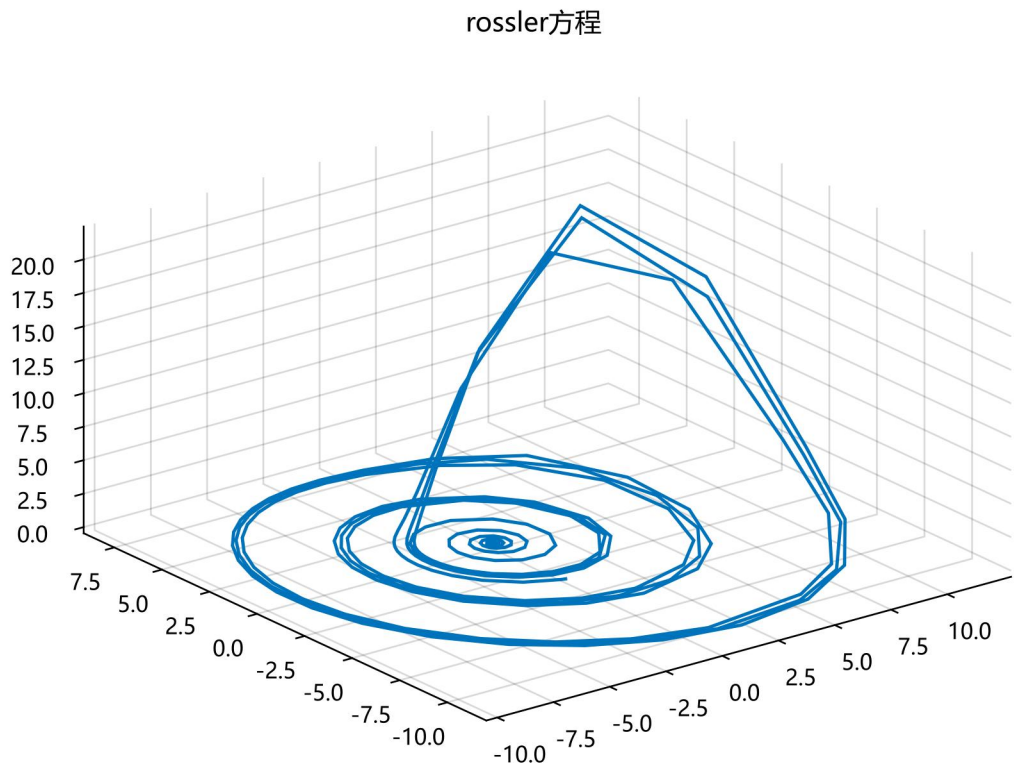
0.0

0.0

1.0e-9
```

解方程并绘图得到图像

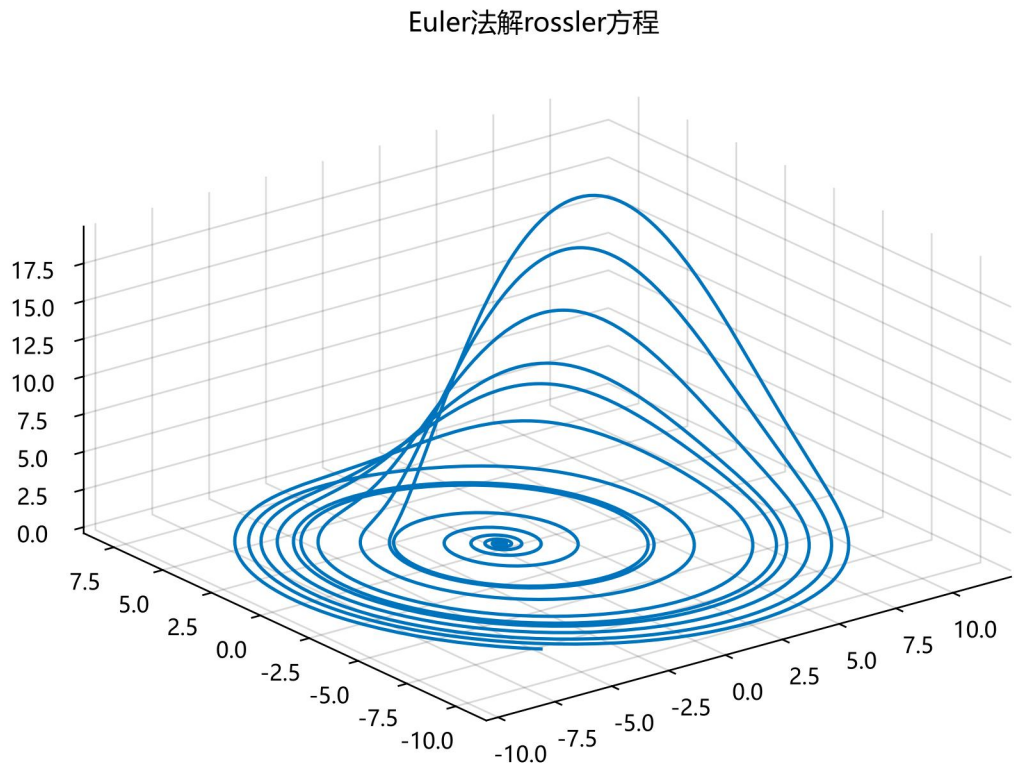
```
sol = solve(prob)
plot3(sol[1,:],sol[2,:],sol[3,:])
title("rossler 方程")
grid("on")
```



(2). 使用定步长算法解 rossler 方程

使用定步长算法时可以给出 dt 或 tstop 关键字分别表示步长或必须到达时间

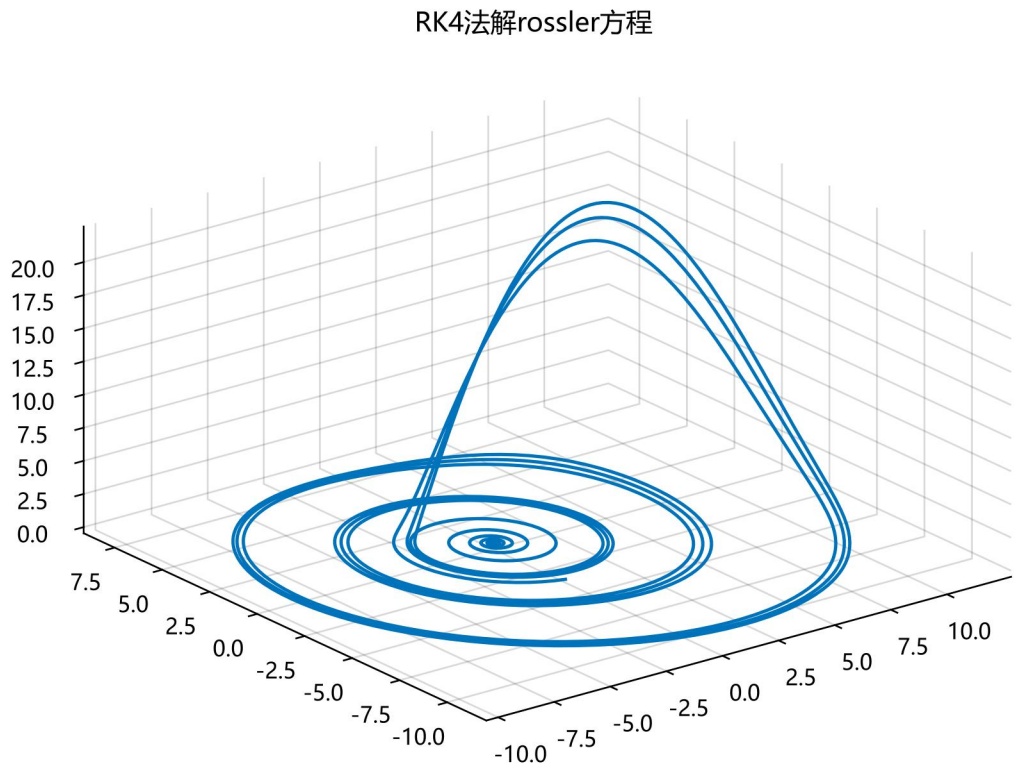
```
sol = solve(prob,Euler(),dt=0.01)
plot3(sol[1,:],sol[2,:],sol[3,:])
title("Euler 法解 rossler 方程")
grid("on")
```



(3). 使用变步长算法解 rossler 方程

使用 RK4 变步长算法时可以给出 dtmax 或 dtmin 关键字分别表示最大步长或最小步长

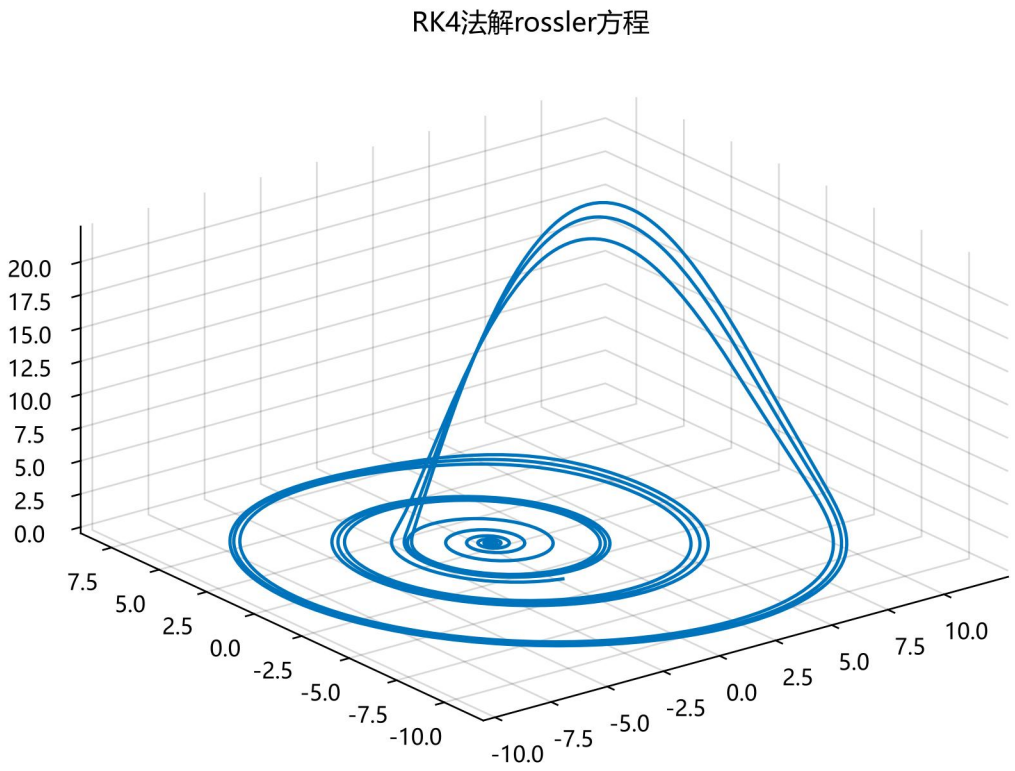
```
sol = solve(prob,RK4(),dtmax=0.01)
plot3(sol[1,:],sol[2,:],sol[3,:])
title("RK4 法解 rossler 方程")
grid("on")
```



(4). 解 rossler 方程设置保存点

使用 saveat 关键字可以使在间隔时间点保存数据

```
sol = solve(prob,saveat=0.01)
plot3(sol[1,:],sol[2,:],sol[3,:])
title("RK4 法解 rossler 方程")
grid("on")
```



(5). 默认算法提示

为了帮助选择默认算法，提供了算法 solve 的关键字 alg_hints。alg_hints 是一个符号向量是在一个高水平上来向求解器描述问题的，其中选项包括：

:auto vs :nonstiff vs :stiff - 表示方程为非刚性/刚性。:auto 允许默认处理算法选择刚度检测算法，使用默认算法使用:auto 来处理。

目前未使用的选项包括：

:interpolant - 表示高精度插值很重要。

:memorybound - 表示解算器将受内存约束。

(6). SDE Specific Alghints

:additive - 表示底层 SDE 添加了干扰。

:stratonovich - 表示解决方案应该遵循 Stratonovich 定理。

(7). 输出控制

以下参数控制解算器的输出行为，它的默认最大输出以提供最佳的交互式用户体验为基准，可以一直减少仅在最终时间点保存解决方案。

以下选项都与输出控制相关，可参考此页末尾的“示例”部分，以示使用。

dense：表示是否保存密集(连续)输出所需的额外件，默认值用于能够产生密

集输出的算法，即默认情况下为 save_everystep&&!isempty(saveat)为真，除非用户已关闭步骤上的保存或已选择 saveat 值。如果 dense=false 该解决方案仍然像一个函数，并且是保存的时间点 sol(t)之间的线性插值。

saveat：表示在解决阶段保存解决方案的具体时间。解算器将以解算器可用的最高效方式在此阵列中的每一个时间点保存。如果只给出 saveat，那么参数和默认情况下 save_everystep 和 dense 均为假。如果 saveat 给一个数字，那么它会自动扩展到 tspan[1]:saveat:tspan[2]。对于无法插值的方法，saveat 可能等同 tstops，默认值是[]。

save_idxs：表示要保存方程组件的指数。默认保存所有指数。例如，如果您正在解决三维 ODE，并且给出 save_idxs =[1, 3]，则只会输出解决方案的第一个和第三个组件。请注意，在这种情况下，输出的解决方案当然是二维的。

tstops：表示定时算法必须达到的“额外”时间。这应该用于帮助解算器处理不连续性和奇点，因为 tstops 在不连续的时候精确的步长将提高精度。如果一种方法不能更改时间步骤（固定时间步方法），则将使用插值，与 saveat 进行匹配。如果方法不能更改时间步骤，tstops 也不能插值，则必须是 dt 的倍数，否则将抛出错误，默认值是[]。

d_discontinuities：表示低阶衍生品中不连续的位置。这将迫使假设衍生连续性的 FSAL 算法在不连续性点重新评估衍生工具。默认值是[]。

save_everystep：每一步结果都保存。默认为真，否则 isempty(saveat)。

save_start：表示初始条件是否应作为第一个时间点包含在解决方案类型中，默认到为真。

save_end：表示是否强制保存最终时间点，而不考虑其他保存设置，默认为真。

initialize_save：表示是否在回调初始阶段（何时）后保存 u_modified=true，默认为真。

(8). 步骤化控制

这些参数用来控制时间步长。

(9). 基本步骤化控制

这些控制时间步长行为的选项的标准是与误差值的比较

$err_scaled = err / (abstol + max(uprev, u) * reltol)$ 对于给定本地容差估计值，并且保证 scaled 的误差值小于 1（注意：容差估计值是本地的，除非特别说

明）。

绝对精度 abstol 控制非缩放错误，因此可以视为零左右的错误，相对误差 reltol 其值受变量影响，因此可以解释 reltol=1e-3 可精确到 3 位，注意公差可以通过 u0 匹配的向量来指定元素。

adaptive：默认为变步长算法，当为定步长时为 false。

abstol：自适应时间步长的绝对容差。这是局部误差估计的容差，不一定是全局误差(尽管这些量是相关的)。对于确定性方程(ODE/DDE/DAE)和 1e-2 随机方程(SDE/RODE)系统默认为 1e-6。

reltol：自适应时间步长的相对容差。这是局部误差估计的容差，不一定是全局误差(尽管这些量是相关的)。对于确定性方程(ODE/DDE/DAE)和 1e-2 随机方程(SDE/RODE)系统默认为 1e-3。

dtmax：自适应时间步长的最大 dt。默认值依赖于包。

dtmin：自适应时间步长的最小 dt。默认值依赖于包。

force_dtmin：声明是否继续，强制使用最小 dt。默认为 false，它让求解器在遇到最小值时发出警告并提前退出 dt。将此设置为 true 允许求解器继续，dt 永远不会低于 dtmin（并在这些情况下忽略容错）。请注意它与大多数互操作包不兼容。

(10). 定步长的使用

如果设置 dt,则算法将按每次迭代 dt 的大小进行计算。

如果 dt 和 tstops 都进行设置，那么算法将步入一个 dt 大小或者使用较小的步长来达到 tstops 点。

如果 tstops 没有设置，那么算法将 dt 直接计算到每个 tstops 值。如果 dt 和 tstops 都没有设置，解算器报错

(11). 自适应步长控制

以下参数控制自适应步长内部更高级的部分，主要用于特定问题上使其更有效率。并且对时间步长算法进行停滞时间的解释。

internalonrm：计算误差估计值的规范函数 internalnorm(u,t),它需要两个量，一是状态变量的 dispatchs 值，另一个是状态变量(scalar 规范)的元素，默认值取决于包。

controller：存在的示例比如控制器，PI 控制器，PID 控制器，预测控制器，默认值取决于包。

gamma：q 方程中的风险因子 γ 使用它的控制器的自适应步长，默认值取决于算法。

beta1：Lund 稳定 α 参数，默认值取决于算法。
beta2：Lund 稳定 β 参数，默认值取决于算法。
qmax：定义自适应 q 可能的最大值，默认值取决于算法。
qmin：定义自适应 q 可能的最小值，默认值取决于算法。
qsteady_min：定义时间步长保持恒定的 1 附近范围的最小值，默认值取决于算法。
qsteady_max：定义时间步长保持恒定的 1 附近范围的最大值，默认值取决于算法。
qoldinit：稳定步进中的初始 qold， 默认值取决于算法。
failfactor：如果隐式方法的牛顿迭代失败，则减少时间步长的量，默认值为 2。

(12). 杂项
maxiters：停止前的最大迭代次数，默认为 1e5。
callback：指定回调， 默认为执行保存例程的回调函数。
isoutofdomain: 指定一个函数 isoutofdomain(u,p,t)，当它返回 true 时，它将拒绝时间步, 默认禁用。
unstable_check：指定一个函数 unstable_check(dt,u,p,t)，当它返回 true 时，它将导致求解器退出并抛出警告。 默认为 any(isnan,u)，即检查是否有任何值是 NaN。
verbose：切换是否在求解器提前退出时抛出警告，默认为真。
merge_callbacks：切换是否将 prob.callback 与 solve 关键字参数回调合并。默认为真。

(13). 进度监测
以下参数控制 Juno IDE 中进度栏的使用。
progress：打开/关闭 Juno 进度条,默认为假。
progress_steps：进度条更新之间的步数,默认值为 1000。
progress_name：控制进度条的名称,默认是问题类型的名称。
progress_message：使用进度条控制消息,默认显示 dt,t,u 的最大值。

(14). 错误计算
如果您正在使用测试问题（例如：ODETestProblem），则以下选项控制计算的错误：

timeseries_errors：在所采取的步骤中打开和关闭错误计算,默认为真。

dense_errors：打开和关闭需要密集输出的步骤的误差计算，并在整个 tspan 的 100 个均匀间隔的点上计算误差，默认为假。

3.2.5.5 数值积分

数值积分问题

函数名	简介
integral	数值积分
integral2	对二重积分进行数值计算
integral3	对三重积分进行数值计算
quad	以自适应 Simpson 积分法计算数值积分
quad2d	计算二重数值积分 - tiled 方法
quadgk	计算数值积分 - 高斯-勒让德积分法
cumtrapz	累积梯形数值积分
trapz	梯形数值积分
del2	离散拉普拉斯算子
diff	差分
gradient	数值梯度
polyder	多项式微分
polyint	多项式积分

3.2.5.5.1 integral

数值积分

A. 语法

```
q,e = integral(fun,xmin,xmax)
q,e = integral(fun, xmin,xmax;Name = Value)
```

B. 说明

q,e = integral(fun,xmin,xmax) 使用全局自适应积分和默认误差容限在 xmin 至 xmax 间以数值形式为函数 fun 求积分。示例

q,e = integral(fun,xmin,xmax;Name = Value) 指定具有一个或多个 Name,Value 对组参数的其他选项。例如，指定 rtol，后跟实数，可调整相对误差。示例。

C. 示例

(1). 广义积分

创建函数

```
fun(x) = exp(-x .^ 2) .* log(x) .^ 2
```

fun(x) = exp(-x .^ 2) .* log(x) .^ 2。

```
q,e = integral(fun, 0, Inf)
```

```
q = 1.9475221777564724
```

(2). 参数化函数

创建带有一个参数 c 的函数 $f(x) = 1/(x^3 - 2x - c)$

```
fun(x, c = 5.0) = 1 ./ (x .^ 3 - 2 * x - c)
```

在 c=5 时，计算从 x=0 至 x=2 的积分。

```
q,e = integral(fun, 0, 2)
```

```
q = -0.4605015338467329
```

(3). 下限的奇异性

创建函数 $f(x) = \ln(x)$

```
fun(x) = log(x)
```

使用默认误差容限计算 x=0 至 x=1 的积分。

```
q1,e1 = integral(fun, 0, 1)
```

```
q1 = -0.9999999983857414 = -0.4605015338467329
```

```
q2,e2 = integral(fun, 0, 1, rtol = 0, atol = 1e-12)
```

```
q2 = -0.9999999999999507
```

(4). 向量值函数

创建向量值函数 $f(x) = [\sin x, \sin 2x, \sin 3x, \sin 4x, \sin 5x]$

并求 x=0 到 x=1 的积分。

```
fun(x) = sin.([1:5...] .* x);
q, = integral.(fun, 0, 1)
q
```

```
q =  5-element Vector{Float64}:
 0.4596976941318603
```

```
0.7080734182735712
0.6633308322001485
0.41341090521590296
0.14326756290735473= -0.4605015338467329
```

(5). Oscillatory 函数的广义积分

创建函数 $f(x)=x^5e^{-x}\sin x$ 。

```
fun(x) = x .^ 5 .* exp(-x) .* sin(x)
```

计算从 x=0 至 x=Inf 的积分，并调整绝对误差和相对误差。

```
q,e = integral(fun, 0, Inf, rtol = 1e-8, atol = 1e-13)
```

```
q =  -15.000000000473056
```

3.2.5.5.2 Integral2

对二重积分进行数值计算

A. 语法

```
q,e = integral2(fun,xmin,xmax,ymin,ymax)
q,e = integral2(fun,xmin,xmax,ymin,ymax;Name=Value)
```

B. 说明

[q,e] = integral2(fun,xmin,xmax,ymin,ymax) 在平面区域 xmin ≤ x ≤ xmax 和 ymin(x) ≤ y ≤ ymax(x) 上逼近函数 z = fun(x,y) 的积分。示例

[q,e] = integral2(fun,xmin,xmax,ymin,ymax,Name=Value) 指定具有一个或多个 Name,Value 对组参数的其他选项。示例。

C. 示例

(1). 将三角形区域与奇异性在边界处集成

以下面的函数为例：

$$f(x,y)=\frac{1}{(\sqrt{x+y})(1+x+y)^2}$$

此函数在 x 和 y 为零时未定义。当奇异性位于积分边界上时，integral2 的性能最佳。

创建匿名函数。

```
fun(y, x) = 1 ./ (sqrt(x + y) .* (1 + x + y) .^ 2)
```

fun (generic function with 1 method)

对$0 \le x \le 1$和$0 \le y \le 1-x$限定的三角形区域计算积分。

```
ymin(x) = 0
ymax(x) = 1 - x
q,e = integral2(fun, 0, 1, ymin, ymax)
```

```
q = 0.2853981634011426
```

(2). 在极坐标中计算二重积分

定义函数 $f(\theta,r)=\dfrac{r}{\sqrt{r\cos\theta+r\sin\theta}(1+r\cos\theta+r\sin\theta)^2}$

```
fun(x,y) = 1 ./ (sqrt(x + y) .* (1 + x + y) .^ 2)
polarfun(r, theta) = fun(r .* cos(theta), r .* sin(theta)) .* r;
```

为 r 的上限定义一个函数。

```
rmax(theta) = 1 ./ (sin(theta) + cos(theta))
rmin(theta) = 0
```

对$0 \le \theta \le \pi/2$和$0 \le r \le r_{max}$限定的区域计算积分。

```
q,e = integral2(polarfun, 0, pi / 2,rmin, rmax)
```

```
q = 0.28539816341070784
```

(3). 使用特定方法和误差容限计算参数化函数的二重积分

使用参数 a=3 和 b=5 创建匿名的参数化函数 $f(x,y)=ax^2+by^2$

```
a = 3;
b = 5;
fun(y, x) = a * x .^ 2 + b * y .^ 2;
```

对区域$0 \le x \le 5$和$-5 \le y \le 0$计算积分。指定约为 10 个有效数字的精度。

```
q,e = integral2(fun, 0, 5, -5, 0, atol = 0, rtol = 1e-10)
```

```
q = 1666.6666666666665
```

3.2.5.5.3 Integral3

对三重积分进行数值计算

A. 语法

```
q,e = integral3(fun,xmin,xmax,ymin,ymax,zmin,zmax)
```

```
q,e = integral3(fun,xmin,xmax,ymin,ymax,zmin,zmax,Name=Value)
```

B. 说明

q,e = integral3(fun,xmin,xmax,ymin,ymax,zmin,zmax) 在区域 xmin ≤ x ≤ xmax、ymin(x) ≤ y ≤ ymax(x) 和 zmin(x,y) ≤ z ≤ zmax(x,y) 逼近函数 z = fun(x,y,z) 的积分。示例

q,e = integral3(fun,xmin,xmax,ymin,ymax,zmin,zmax,Name=Value) 指定具有一个或多个 Name,Value 对组参数的其他选项。示例。

C. 示例

(1). 具有有限范围的三重

定义匿名函数 $f(x,y,z)=y\sin x+z\cos x$

```
fun(z, y, x) = y .* sin(x) + z .* cos(x)
```

对 $0\le x\le\pi$ 、 $0\le y\le 1$ 和 $-1\le z\le 1$ 限定的三角形区域计算积分。

```
q,e = integral3(fun, 0, pi, 0, 1, -1, 1)
```

```
ans = 2.0000000000001297
```

(2). 积分在笛卡尔坐标中对单位球面计算积分

定义匿名函数 ： $f(z,y,x)=y\sin x+z\cos x$

```
fun(z, y, x) = x .* cos(y) + x .^ 2 .* cos(z)
```

定义积分的范围。

```
xmin = -1;
xmax = 1;
ymin(x) = -sqrt(1 - x .^ 2);
ymax(x) = sqrt(1 - x .^ 2);
zmin(x, y) = -sqrt(1 - x .^ 2 - y .^ 2);
zmax(x, y) = sqrt(1 - x .^ 2 - y .^ 2);
q,e = integral3(fun, xmin, xmax, ymin, ymax, zmin, zmax)
```

```
q = 0.7795554546739307
```

(3). 球体的四维积分

使用对 integral3 和 integral 的嵌套调用来计算四维球体的体积。

$$V_4(r)=\int_0^{2\pi}\int_0^{\pi}\int_0^{\pi}\int_0^{r} r^3\sin^2(\theta)\sin(\phi)\,dr\,d\theta\,d\phi\,d\xi$$

Syslab 中的 integral 求积法函数直接支持一维、二维和三维积分。然而，

要求解四维和更高阶积分，需要嵌套对求解器的调用。

使用按元素运算符（.^ 和 .*）为被积函数创建函数句柄 $f(r,\theta,\phi,\xi)$。

```
f(xi, phi, theta) = sin(theta) .^ 2 .* sin(phi)
```

接下来，创建一个函数句柄，它使用 integral3 计算三个积分。

```
Q(r) = r .^ 3 .* integral3(f, 0, pi, 0, pi, 0, 2 * pi)[1]
```

最后，在对 integral 的调用中使用 Q 作为被积函数。求解此积分需要为半径 r 选择一个值，因此请使用 r=2。

```
I,e = integral(Q, 0, 2)
```

```
I = 78.95683520881754
```

确切答案是 $\frac{\pi^2 r^4}{2\Gamma(2)}$

```
I_exact = pi^2*2^4/(2*gamma(2))
```

```
I_exact = 78.95683520871486
```

3.2.5.5.4 quad

（不推荐）以自适应 Simpson 积分法计算数值积分

A. 语法

```
q,e = quad(fun,a,b)
q,e = quad(fun,a,b;Name = Value)
```

B. 说明

求积法是用于计算函数图形下的面积的数值方法，即计算定积分。

$$q = \int_a^b f(x)dx$$

q = quad(fun,a,b) 尝试使用递归自适应 Simpson 积分法求取函数 fun 从 a 到 b 的近似积分，误差小于 1e-6。fun 是函数句柄。范围 a 和 b 必须是有限的。函数 y = fun(x) 应接受向量参数 x 并返回向量结果 y，即在每个 x 元素处计算的被积函数。

参数化函数解释了如何为函数 fun 提供其他参数（如果需要）。

q = quad(fun,a,b,atol) 使用绝对误差容限 atol 代替默认值 1.0e-6。tol 值越大，函数计算量越少并且计算速度加快，但结果不太精确。

q,e = quad(fun,a,b;Name = Value) 返回函数误差。

函数 quadl 可能更有效，其精确度较高并且被积函数均匀。

以下列表包含的信息可以帮助您确定要使用的 Syslab 中的求积法函数：

quad 函数可能最有效，但其精确度较低并且被积函数不均匀。

quadgk 函数的精确度最高，会产生震荡被积函数。它支持无限区间并且可以处理端点处的适度奇异性。它还支持沿分段线性路径的围道积分。

如果区间是无限区间（例如 [a,∞)），则要存在 fun(x) 的积分，fun(x) 必须在 x 接近无限大时衰减，并且 quadgk 要求它快速衰减。特殊方法应该用于无限区间中的振荡函数，但如果 fun(x) 衰变足够快，则可以使用 quadgk。

如果奇异性不太稳固，quadgk 函数将对有限端点处的奇异函数求积分。例如，它将对在端点 c 处的行为与 $p \geq -1/2$ 的 $\log|x-c|$ 或 $|x-c|^p$ 类似的函数求积分。如果函数在 (a,b) 内的各点是奇异函数，则在奇异点为端点的子区间上将积分写入为积分和，通过 quadgk 计算它们，并将结果相加。

C. 示例

要计算积分 $\int_0^2 \frac{1}{x^3-2x-5}dx$

编写一个用于计算被积函数的函数 myfun

myfun(x) = 1/(x.^3-2*x-5);

然后将 myfun 的函数句柄 @myfun 以及 0 至 2 的积分范围一起传递到 quad。

Q,E = quad(myfun,0,2)

Q = -0.46050153384673287

另外，您也可以将被积函数作为匿名函数句柄 F 传递到 quad：

F = x -> 1/(x.^3-2*x-5);

Q,E = quad(F,0,2)

3.2.5.5.5 quad2d

计算二重数值积分 - tiled 方法

A. 语法

```
q,E = integral2(fun,a,b,c,d)
```

```
q,E = integral2(fun,a,b,c,d;Name=Value)
```

B. 说明

求 q,E = integral2(fun,a,b,c,d) 逼近 fun(x,y) 在平面区域 a≤x≤b 和 c(x)≤y≤d(x) 上的积分。边界 c 和 d 均可为标量或函数句柄。示例

q,E = integral2(fun,a,b,c,d,Name=Value) 指定具有一个或多个 Name,Value 对组参数的其他选项。

q,E = quad2d(___) 还返回绝对误差 E = | q - I | 的逼近 上限，其中 I 是积分的确切值。

C. 示例

(1). 计算二重积分

以下方程求积分 $y\sin(x)+x\cos(y)$

区间为 $\pi \le x \le 2\pi$ 和 $0 \le x \le \pi$s

```
fun(y, x) = y .* sin(x) + x .* cos(y)
Q, E = quad2d(fun, pi, 2 * pi, 0, pi)
```

```
Q = -9.86960440109
```

将结果与积分的真值 $-\pi^2$ 进行比较。

```
-pi^2
```

```
ans = -9.869604401089358
```

(2). 位于积分边界具有奇异性的被积函数

求以下函数的积分 $\left[(x+y)^{1/2}(1+x+y)^2\right]^{-1}$

区间为 $0 \le x \le 1$ 和 $0 \le y \le 1-x$。此被积函数在原点 (0,0) 是无限的，它位于积分区域的边界

```
fun(y, x) = 1 ./ (sqrt(x + y) .* (1 + x + y) .^ 2);
ymax(x) = 1 - x
ymin(x) = 0
Q, e = quad2d(fun, 0, 1, ymin, ymax)
```

```
Q = 0.2853981634011426
```

积分的真值为 π/4 - 1/2。

```
pi/4 - 0.5
```

```
ans = 0.2853981633974483
```

3.2.5.5.6 quad3d

对三重积分进行数值计算

A. 语法

```
q,e = quad3d(fun,xmin,xmax,ymin,ymax,zmin,zmax)
q,e = quad3d(fun,xmin,xmax,ymin,ymax,zmin,zmax,Name=Value)
```

B. 说明

q,e = quad3d(fun,xmin,xmax,ymin,ymax,zmin,zmax) 在区域 xmin ≤ x ≤ xmax、ymin(x) ≤ y ≤ ymax(x) 和 zmin(x,y) ≤ z ≤ zmax(x,y) 逼近函数 z = fun(x,y,z) 的积分。示例

q,e = quad3d(fun,xmin,xmax,ymin,ymax,zmin,zmax,Name=Value) 指定具有一个或多个 Name,Value 对组参数的其他选项。示例。

C. 示例

(1). 具有有限范围的三重积分

具有有限范围的三重积分 $f(x,y,z)=y\sin x+z\cos x$

```
fun(z, y, x) = y .* sin(x) + z .* cos(x)
```

对 $0\le x\le\pi$ 、 $0\le y\le 1$ 和 $-1\le z\le 1$ 限定的三角形区域计算积分。

```
q,e = quad3d(fun, 0, pi, 0, 1, -1, 1)
```

```
q = 2.0000000000001297
```

(2). 积分在笛卡尔坐标中对单位球面计算积分

定义匿名函数 ： $f(z,y,x)=x\cos y+x^2\cos z$

```
fun(z, y, x) = x .* cos(y) + x .^ 2 .* cos(z)
```

定义积分的范围。

```
xmin = -1;
xmax = 1;
ymin(x) = -sqrt(1 - x .^ 2);
```

```
ymax(x) = sqrt(1 - x .^ 2);
zmin(x, y) = -sqrt(1 - x .^ 2 - y .^ 2);
zmax(x, y) = sqrt(1 - x .^ 2 - y .^ 2);

q,e = quad3d(fun, xmin, xmax, ymin, ymax, zmin, zmax)
```

```
q = 0.7795554546739307
```

(3). 球体的四维积分

使用对 quad3d 和 integral 的嵌套调用来计算四维球体的体积。

$$V_4(r) = \int_0^{2\pi}\int_0^{\pi}\int_0^{\pi}\int_0^{r} r^3 \sin^2(\theta)\sin(\phi)\,dr\,d\theta\,d\phi\,d\xi$$

Syslab 中的 integral 求积法函数直接支持一维、二维和三维积分。然而，要求解四维和更高阶积分，需要嵌套对求解器的调用。

使用按元素运算符（.^ 和 .*）为被积函数创建函数句柄 $f(r,\theta,\phi,\xi)$。

```
f(xi, phi, theta) = sin(theta) .^ 2 .* sin(phi)
```

接下来，创建一个函数句柄，它使用 integral3 计算三个积分。

```
Q(r) = r .^ 3 .* integral3(f, 0, pi, 0, pi, 0, 2 * pi)[1]
```

最后，在对 integral 的调用中使用 Q 作为被积函数。求解此积分需要为半径 r 选择一个值，因此请使用 r=2。

```
I,e = integral(Q, 0, 2)
```

```
I = 78.95683520881754
```

确切答案是 $\frac{\pi^2 r^4}{2\Gamma(2)}$

```
I_exact = pi^2*2^4/(2*gamma(2))
```

```
I_exact = 78.95683520871486
```

3.2.5.5.7 quadgk

计算数值积分 - 高斯-勒让德积分法

A. 语法

```
quadgk(f, a,b,c...; rtol=sqrt(eps), atol=0, maxevals=10^7, order=7,
norm=norm)
```

B. 说明

对函数 f(x) 进行 a 到 b 的数值积分，以及可选的额外间隔 b 到 c 等等。

关键字选项包括相对误差 rtol（如果 atol==0，默认为 sqrt(eps)端点的精度），绝对值误差 atol（默认为 0），一个函数评估的最大数 maxevals（默认为 10^7），以及顺序积分规则（默认为 7）。

返回估计值的(I,E) 积分 I 和估计的上限 绝对误差 E.

如果 maxevals 不是 超过则 E <= max(atol, rtol*norm(I)) 将举行。 （请注意，这很有用 在以下情况下指定正 atol norm(I) 可能为零。）

端点等也可以是 复数（在这种情况下，积分是 在直线段上执行 复平面）。如果端点是 BigFloat，那么积分将是 也以 BigFloat 精度执行。

C. 示例

(1). 在端点处具有奇异性的积分

计算积分

$$q = \int_0^1 e^x \ln(x)\mathrm{d}x$$

此积分在 x=0 点处具有奇异性，因为 ln(0) 发散到 −∞。

为被积函数创建一个匿名函数。log 函数计算 ln(x)。

```
f = x->exp(x) .* log(x)
```

在从 0 到 1 的区间上对 f 求积分。

```
q = quadgk(f,0,1)
```

```
(-1.3179021482258861, 1.825223310455282e-8)
```

(2). 检查觉得误差

使用 quadgk 计算难以计算的振荡被积函数。

计算积分

$$Q = \int_0^{\pi} \sin(20000\pi x)\mathrm{d}x$$

被积函数振荡非常快，因此很难计算。使用 quadgk 计算积分，并指定两个输出来检查接近误差容限的程度。

```
fun = x -> sin(2e4*pi*x)
Q,errbnd = quadgk(fun,0,pi)
```

```
Q = 1.6656181375389457e-6

errbnd = 9.569988856586691e-13
```

```
Q1,errbnd1 = quadgk(fun,0,pi,maxevals = 10^5)
```

```
Q1 = 0.0004664617225459399
```

```
errbnd1 = 0.11154537033417948
```

3.2.5.5.8 cumtrapz

累积梯形积分

A. 语法

```
Q = cumtrapz(Y)
Q = cumtrapz(X,Y)
Q = cumtrapz(...,dim)
```

B. 说明

Q = cumtrapz(Y)是通过梯形法按单位间距计算 Y 的近似累计积分。Y 的大小确定求积分所沿用的维度：

如果 Y 是向量，则 cumtrapz(Y)是 Y 的累计积分。

如果 Y 是矩阵，则 cumtrapz(Y)是每一列的累计积分。

如果 Y 是多维数组，则 cumtrapz(Y)对大小不等于 1 的第一个维度求积分。

Q = cumtrapz(X, Y)根据 X 指定的坐标或标量间距对 Y 进行积分。

如果 X 是坐标向量，则 length(X)必须等于 Y 的大小不等于 1 的第一个维度的大小。

如果 X 是标量间距，则 cumtrapz(X, Y)等于 X*cumtrapz(Y)。

Q = cumtrapz(...,dim)使用前面的任何语法沿维度 dim 求积分。必须指定 Y，也可以指定 X。如果指定 X，则它可以是长度等于 size(Y, dim)的标量或向量。例如，如果 Y 是矩阵，则 cumtrapz(X,Y,2)对 Y 的每一行进行累计积分

C. 示例

(1). 采用单位间距对数据向量求积分

计算数据点之间间距为 1 的向量的积分。

创建数据的数值向量。

```
Y = [1 4 9 16 25]
```

Y 包含域 [1, 5] 中的 f(x)=x^2 的函数值。

使用 trapz 按单位间距对数据求积分。

```
Q = cumtrapz(Y)
```

```
Q = 1×1 Matrix{Float64}:
```

```
42.0
```

该近似积分生成值 42。在这种情况下，确切答案有些小，41+1//3。trapz 函数高估积分值，因为 f(x) 是向上凹的。

(2). 采用非单位间距对数据向量求积分

计算数据点间距均匀但不等于 1 的向量的积分。

创建域向量。

```
X = 0:pi/5:pi
```

计算 X 的正弦值。

```
Y = sin.(X)
```

使用 cumtrapz 计算 Y 的累计积分。当点之间的间距不变但不等于 1 时，为 x 创建向量的替代方式是指定标量间距值。在这种情况下，cumtrapz(pi/5,Y) 与(pi/5).*cumtrapz(Y)相同。

```
Q = cumtrapz(collect(X), Y)
```

```
Q = 6-element Vector{Float64}:
 0.0
 0.18465818304904566
 0.6680995825722469
 1.2656660155205581
 1.7491074150437596
 1.9337655980928052
```

(3). 采用非均匀间距对矩阵求积分

对具有非均匀数据间距的矩阵的行求积分。

创建一个 x 坐标向量和一个按不规则间隔测得的观测值矩阵。Y 中的行代表在 X 中各时间处测得的速度数据，分别来自三次不同的试验。

```
X = [1 2.5 7 10]
Y = [5.2 7.7 9.6 13.2;4.8 7.0 10.5 14.5;4.9 6.5 10.2 13.8]
```

使用 cumtrapz 分别对每一行进行积分，然后求出每个试验的累积行程距离。

由于数据不是按固定间隔计算的，因此指定 x 来表示数据点之间的间距。

由于数据位于 Y 的行中，因此指定 dim=2。

```
Q1 = cumtrapz(X[:], Y, 2)
```

```
Q1 = 3×4 Matrix{Float64}:
 0.0  9.675  48.6    82.8
 0.0  8.85   48.225  85.725
 0.0  8.55   46.125  82.125
```

(4). 多次累积积分

在 x 和 y 方向执行嵌套积分。绘制结果，以可视方式显示两个方向上的累

积积分值。

创造一个由域值构成的网络。

```
x = -2:0.1:2
y = -2:0.2:2
X, Y = meshgrid2(x,y)
```

计算网格上的函数

$f(x,y)=10x^2+20y^2$

```
F = X.^2 + Y.^2
```

cumtrapz 对数值数据而不是函数表达式求积分，因此要对数据矩阵使用 cumtrapz，通常无需了解基础函数。在已知函数表达式的情况下，可以使用 integral、integral2 或 integral3。

使用 cumtrapz 求二重积分的近似值

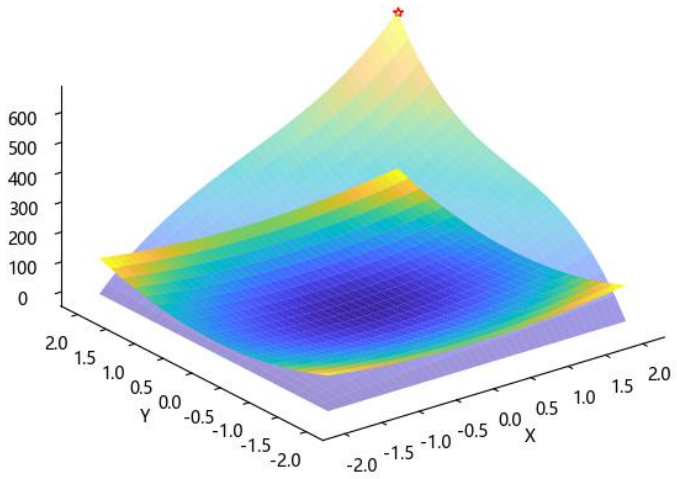
$I(a,b)=\int_{-2}^{b}\int_{-2}^{a}(10x^2+20y^2)dxdy$

要执行此二重积分，请使用 cumtrapz 的嵌套函数调用。内部调用首先对数据的行进行积分，然后外部调用对列进行积分。

```
I = cumtrapz(collect(y), cumtrapz(collect(x), F, 2))
```

绘制表示原始函数的曲面以及表示累积积分的曲面。累计积分曲面上的每个点都给出了二重积分的中间值。I 中的最后一个值给出了二重积分的总体逼近值，I[end] = 642.4。用一个红色的星形在图上标记此点。

```
surf(X, Y, F, edgecolor="none")
xlabel("X")
ylabel("Y")
hold("on")
surf(X, Y, I, facealpha=0.5, edgecolor="none")
plot3(X[end],Y[end],I[end],"r*")
hold("off")
```



3.2.5.5.9 trapz

梯形数值积分

A. 语法

```
Q = trapz(Y)
Q = trapz(X,Y)
Q = trapz(___,dim)
```

B. 说明

Q = trapz(Y) 通过梯形法计算 Y 的近似积分（采用单位间距）。Y 的大小确定求积分所沿用的维度：

如果 Y 为向量，则 trapz(Y) 是 Y 的近似积分。

如果 Y 为矩阵，则 trapz(Y) 对每列求积分并返回积分值的行向量。

如果 Y 为多维数组，则 trapz(Y) 对其大小不等于 1 的第一个维度求积分。该维度的大小变为 1，而其他维度的大小保持不变。

Q = trapz(X,Y) 根据 X 指定的坐标或标量间距对 Y 进行积分。

如果 X 是坐标向量，则 length(X) 必须等于 Y 的大小不等于 1 的第一个维度的大小。

如果 X 是标量间距，则 trapz(X,Y) 等于 X*trapz(Y)。

Q = trapz(___,dim) 使用以前的任何语法沿维度 dim 求积分。必须指定 Y，也可以指定 X。如果指定 X，则它可以是长度等于 size(Y,dim) 的标量或向量。

例如，如果 Y 为矩阵，则 trapz(X,Y,2) 对 Y 的每行求积分

C. 示例

(1). 采用单位间距对数据向量求积分

计算数据点之间间距为 1 的向量的积分。

创建数据的数值向量。

```
Y = [1 4 9 16 25]
```

Y 包含域 [1, 5] 中的 f(x)=x^2 的函数值。

使用 trapz 按单位间距对数据求积分。

```
Q = trapz(Y)
```

```
Q = 1×1 Matrix{Float64}:
42.0
```

该近似积分生成值 42。在这种情况下，确切答案有些小，41+1//3。trapz 函数高估积分值，因为 f(x) 是向上凹的。

(2). 采用非单位间距对数据向量求积分

计算数据点间距均匀但不等于 1 的向量的积分。

创建域向量。

```
X = 0:pi/100:pi
```

计算 X 的正弦值。

```
Y = sin.(X)
```

使用 trapz 对 Y 求积分。

```
Q = trapz(X,Y)
```

```
Q = 1×1 Matrix{Float64}:
1.999342104837625
```

当点之间的间距不变但不等于 1 时，为 X 创建向量的替代方法是指定标量间距值。在这种情况下，trapz(pi/100,Y) 与 pi/100*trapz(Y) 相同

(3). 采用非均匀间距对矩阵求积分

对具有非均匀数据间距的矩阵的行求积分。

创建一个 x 坐标向量和一个按不规则间隔测得的观测值矩阵。Y 中的行代表在 X 中各时间处测得的速度数据，分别来自三次不同的试验。

```
X = [1 2.5 7 10]
Y = [5.2   7.7   9.6   13.2;
```

```
4.8   7.0  10.5   14.5;
4.9   6.5  10.2   13.8]
```

使用 trapz 分别对每一行进行积分，然后求出每次试验中经过的总距离。

由于数据不是按固定间隔计算的，因此指定 X 来表示数据点之间的间距。

由于数据位于 Y 的行中，因此指定 dim = 2。

```
Q1 = trapz(X,Y,2)
```

```
Q1 =

3×1 Matrix{Float64}:

82.8

85.725

82.125
```

(4). 多个数值积分

创建一个由域值构成的网格。

```
x = -3:.1:3
y = -5:.1:5
X,Y = meshgrid2(x,y)
```

计算网格上的函数

$$f(x,y) = x^2 + y^2$$

```
F = X.^2 + Y.^2
```

trapz 对数值数据、而不是函数表达式求积分，因此表达式通常无需已知可对数据矩阵使用 trapz。

使用 trapz 求二重积分的近似值

$$I = \int_{-5}^{5}\int_{-3}^{3}\left(x^2 + y^2\right)dxdy$$

要对数值数据的数组执行二重或三重积分运算，请嵌套对 trapz 的函数调用。

```
I = trapz(y,trapz(x,F,2))
```

```
I = 1×1 Matrix{Float64}:

680.1999999999999
```

trapz 先对 x 求积分以生成列向量。然后，y 上的积分可将列向量减少为

单个标量。trapz 稍微高估计确切答案 680，因为 f(x,y) 是向上凹的。

3.2.5.5.10 del2

离散拉普拉斯算子

A. 语法

```
L = del2(U)
L = del2(U,h)
L = del2(U,hx,hy,...,hN)
```

B. 说明

L = del2(U) 在所有点之间使用默认间距 h = 1，返回应用于 U 的拉普拉斯微分运算子的离散近似值。 示例

L = del2(U,h) 用于在 U 的所有维度中的点之间指定均匀的标量间距 h。示例

L = del2(U,hx,hy,...,hN) 用于在 U 的每个维度中的点之间指定间距 hx,hy,...,hN。将每个间距输入指定为坐标的标量或向量。间距输入数必须等于 U 中的维度数。 示例

第一个间距值 hx 指定点的 x 间距（标量）或 x 坐标（向量）。如果是向量，则其长度必须等于 size(U,2)。

第二个间距值 hy 指定点的 y 间距（标量）或 y 坐标（向量）。如果是向量，则其长度必须等于 size(U,1)。

所有其他间距值指定 U 中对应维度的各点的间距（标量）或坐标（向量）。对于 n > 2 的情况，如果第 n 个间距输入是向量，则其长度必须等于 size(U,n)。

C. 示例

(1). 向量的第二个导数

根据位置数据的向量计算对象的加速度。

创建位置数据的向量。

```
p = [1 3 6 10 16 18 29]
```

要找出对象的加速度，请使用 del2 来计算 p 的第二个数值导数。在数据点之间使用默认间距 h = 1。

```
L = 4*del2(p)
```

```
L = 1×7 Matrix{Float64}:
 1.0  1.0  1.0  2.0  -4.0  9.0  22.0
```

L 的每个值是该点的即时加速度的近似值。

(2). 余弦向量的第二个导数

计算余弦向量的一维离散维拉普拉斯算子。

定义函数的域。

```
x = LinRange(-2*pi,2*pi,100)
```

这会在 $-2\pi \le x \le 2\pi$ 范围内生成 100 个等间距点。

在此域内创建余弦值的向量。

```
U = cos.(x)
```

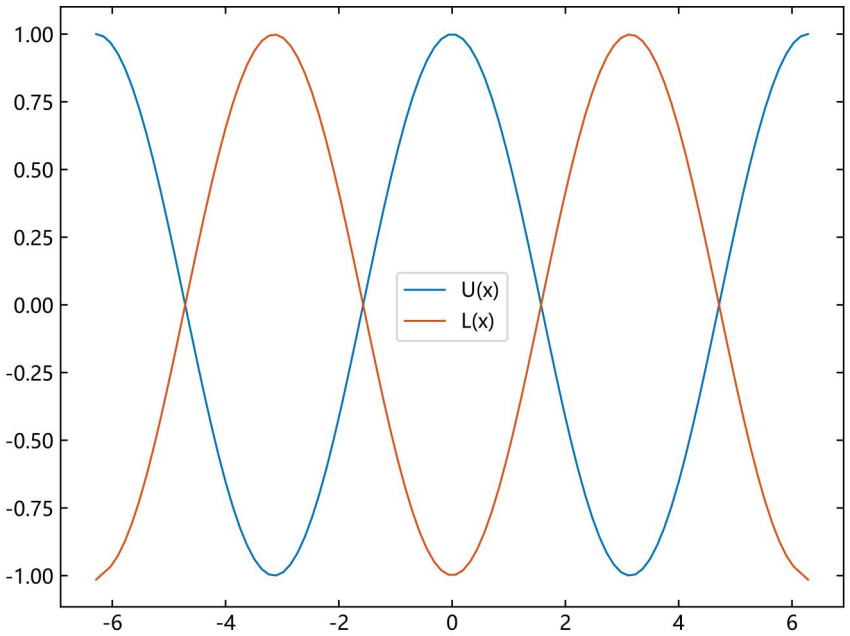
使用 del2 计算 U 的拉普拉斯算子。使用域向量 x 确定 U 中每个点的一维坐标。

```
L = 4*del2(U,x)
```

通过解析，此函数的拉普拉斯算子等于 $\Delta U = -\cos(x)$ 。

绘制结果。

```
plot(x,U,x,L)
legend(["U(x)","L(x)","Location","Best"])
```



U 和 L 的图形与拉普拉斯算子的分析结果一致。

(3). 多元函数的拉普拉斯算子

计算并绘制多元函数的离散拉普拉斯算子。

定义函数的 x 和 y 域。

```
x,y = meshgrid2(-5:0.25:5,-5:0.25:5)
```

在此域中定义$U(x,y)=\frac{1}{3}(x^4+y^4)$函数。

```
U = 1/3 .*(x.^4+y.^4)
```

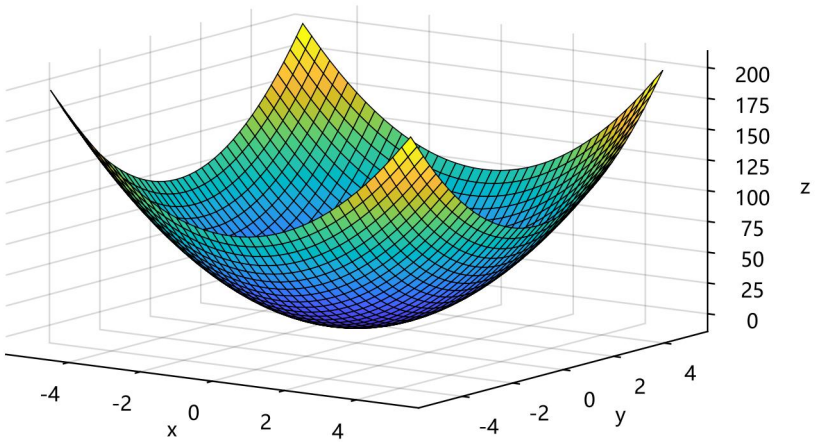
使用 del2 计算此函数的拉普拉斯算子。U 中各点之间的间距在所有方向上都相等，因此只要指定一个间距输入 h 即可。

```
h = 0.25
L = 4*del2(U,h)
```

通过解析，此函数的拉普拉斯算子等于$\Delta U(x,y)=4x^2+4y^2$。

绘制离散拉普拉斯算子 L。

```
figure()
surf(x,y,L)
grid("on")
title(raw"Plot of $ \Delta U(x,y) = 4x^2+4y^2$")
xlabel("x")
ylabel("y")
zlabel("z")
plt_view(35,14)
```



L 的图形与拉普拉斯算子的分析结果一致

(4). 自然对数函数的拉普拉斯算子

计算自然对数函数的离散拉普拉斯算子。

在实数网格上定义函数的 x 和 y 域。

```
x,y = meshgrid2(-5:5,-5:0.5:5)
```

在此域中定义$U(x,y)=\frac{1}{2}\log(x^2y)$函数。

```
U =@. 0.5*log(Complex(x^2*y))
```

当参数 y 为负值时，对数为复数值。

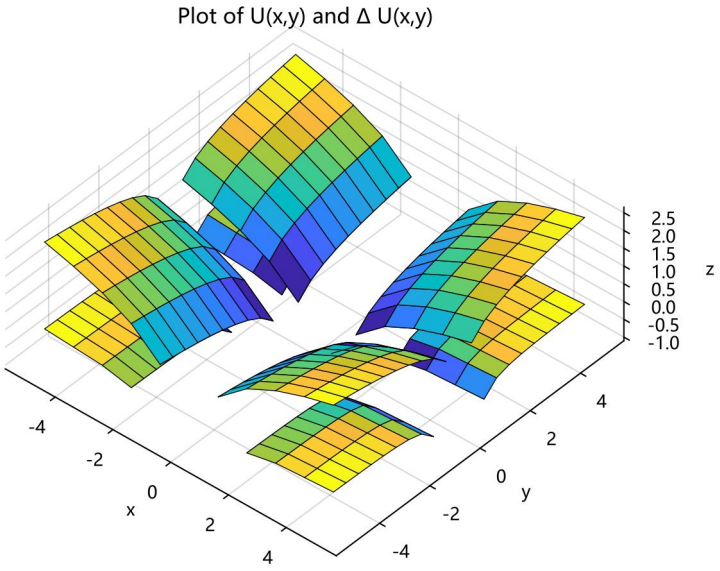
使用 del2 计算此函数的离散拉普拉斯算子。指定每个方向上的网格点之间的间距。

```
hx = 1
hy = 0.5
L = 4*del2(U,hx,hy)
```

通过解析，拉普拉斯算子等于$\Delta U(x,y)=-(1/x^2+1/y^2)$。此函数不在行 或上定义。

将 U 和 L 的实部绘制在同一个图形上。

```
figure()
surf(x,y,real(L))
hold("on")
surf(x,y,real(U))
grid("on")
title(raw"Plot of U(x,y) and $ \Delta $ U(x,y)")
xlabel("x")
ylabel("y")
zlabel("z")
plt_view(41,58)
```



上曲面是 U，下曲面是 L。

3.2.5.5.11 diff

差分

A. 语法

```
D = diff(A::AbstractVector)
D = diff(A::AbstractArray, dims::Integer)
```

B. 说明

D = diff(A::AbstractVector) 计算向量上相邻元素之间的差分。如果 A 是长度为 m 的向量，则 D = diff(A) 返回长度为 m-1 的向量。

D = diff(A::AbstractArray, dims::Integer) 是沿 dims 指定的维度计算相邻元素之间的差分。

C. 示例

(1). 向量元素之间的差分

创建向量 A ，然后计算元素之间的差分。

```
A = [2, 5, 3, 10];
D = diff(A)
```

```
3-element Vector{Int64}:
```

```
3
-2
7
```

请注意，D 的元素比 A 少一个。

(2). 矩阵行之间的差分

创建一个 3×3 矩阵，然后计算各行之间的一阶差分。

```
A = [1 1 1; 5 5 5 ; 25 25 25];
D = diff(A, dims=1)
```

```
2×3 Matrix{Int64}:
4   4   4
20  20  20
```

D 是 2×3 矩阵。

(3). 矩阵列之间的差分

创建一个 3×3 矩阵，然后计算各列之间的一阶差分。

```
A = [1 3 5; 7 11 13; 17 19 23];
D = diff(A, dims=2)
```

```
3×2 Matrix{Int64}:
2  2
4  2
2  4
```

D 是 3×2 矩阵。

(4). 多阶差分

创建向量 X，然后计算元素之间的二阶差分。

```
X = [0,5,15,30,50,75,105];
Y = diff(diff(X))
```

```
5-element Vector{Int64}:
5
5
5
5
5
```

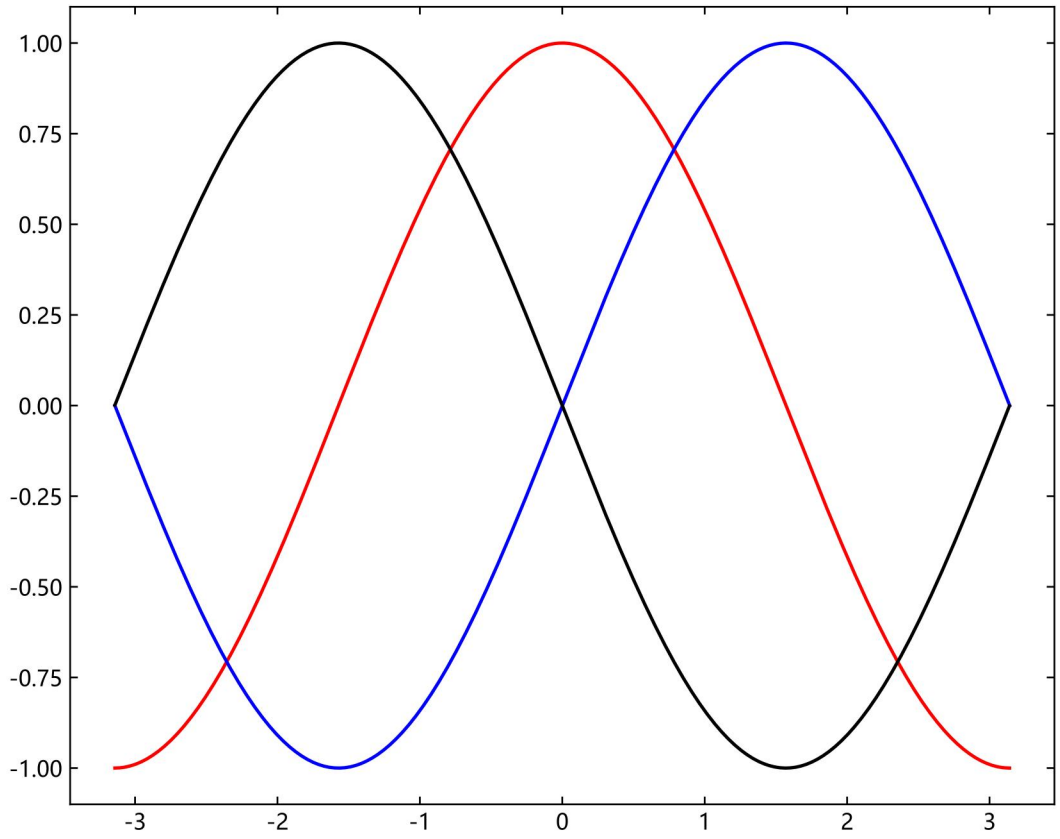
(5). 使用差分求导数近似值

使用 diff 函数和语法 y = diff(f)./h 求偏导数近似值，其中 f 是函数值在某些域 x 上计算的向量，h 是一个相应的步长。

例如，sin.(x) 相对于 x 的第一个导数为 cos.(x)，相对于 x 的第二个导数

值为 -sin.(x)。可以使用 diff 求这些导数的近似值。

```
h = 0.001;              # step size
X = collect(-pi:h:pi);    # domain
f = sin.(X);             # range
Y = diff(f)./h;          # first derivative
Z = diff(Y)./h;          # second derivative
plot(X[1:length(Y)],Y,"r",X,f,"b", X[1:length(Z)],Z,"k")
```



在此绘图中，蓝色线条对应原始函数 sin ，红色线条对应计算出的第一个导数 cos ，黑色线条对应计算出的第二个导数 -sin 。

3.2.5.5.12 gradient

数值梯度

A. 语法

```
FX = gradient(F)
FX, FY = gradient(F)
FX, FY, ..., FN = gradient(F)
```

```
_ = gradient(F, h)
_ = gradient(F, hx, hy,..., hN)
```

B. 说明

FX = gradient(F) 返回向量 F 的一维数值梯度。输出 FX 对应于 ，即 x（水平）方向上的差分。点之间的间距假定为 1。示例

FX, FY = gradient(F) 返回矩阵 F 的二维数值梯度的 x 和 y 分量。附加输出 FY 对应于 ，即 y（垂直）方向上的差分。每个方向上的点之间的间距假定为 1。示例

FX, FY, FZ,...,FN = gradient(F) 返回 F 的数值梯度的 N 个分量，其中 F 是一个 N 维数组。

_ = gradient(F, h) 使用 h 作为每个方向上的点之间的均匀间距。您可以指定上述语法中的任何输出参数。示例

_ = gradient(F, hx, hy,..., hN) 为 F 的每个维度上的间距指定 N 个间距参数。示例。

C. 示例

(1). 向量的梯度

计算单调递增向量的梯度。

```
x = 1:10
```

```
x = 1:10
```

```
fx = TyMath.gradient(x)
```

```
fx = 10-element Vector{Float64}:
  1.0
  1.0
  1.0
  1.0
  1.0
  1.0
  1.0
  1.0
  1.0
  1.0
```

(2). 向量场的等高线图

使用特定点处的梯度线性逼近附近点处的函数值，并将其与实际值进行比较。

函数值的线性逼近方程是 $f(x) \approx f(x_0) + (\nabla f)_{x_0} \cdot (x - x_0)$

也就是说，如果知道函数 $f(x_0)$ 的值和导数 $(\nabla f)_{x_0}$ 在特定点 x_0 的斜率，则可以根据此信息近似求出该函数在附件点 $f(x) = f(x) + e$ 的值。

计算正弦函数在 -1 和 0.5 之间的一些值，然后计算梯度。

```
y = sin.(-1:0.25:0.5)
yp = TyMath.gradient(y, 0.25)
```

根据 $x = 0.5$ 处的函数值和导数来预测 sin(0.5005) 的值。

```
y_guess = y[end] + yp[end]*(0.5005-0.5)
```

```
y_guess = 0.4798895817629023
```

计算实际比较

```
y_actual = sin(0.5005)
```

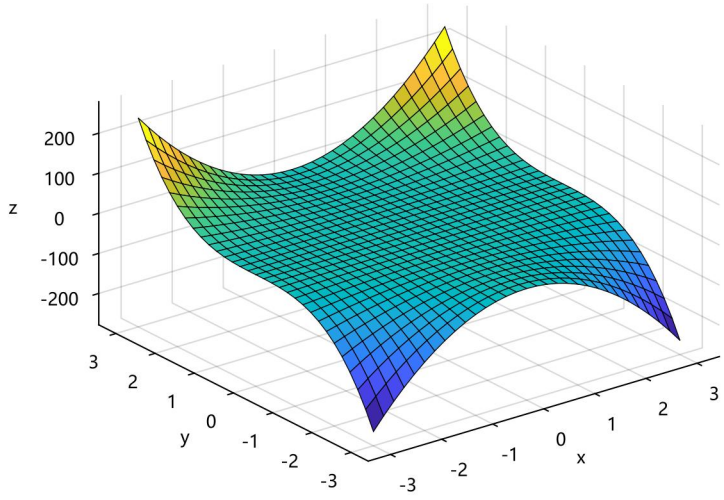
```
y_actual = 0.4798642699386741
```

(3). 计算指定点处的梯度

求多变量函数在指定点处的梯度值。

以多变量函数 $f(x, y) = x^2 y^3$ 为例。

```
x, y = meshgrid2(-3:0.2:3, -3:0.2:3)
f = @. x^2 * y^3
surf(x,y,f)
xlabel("x")
ylabel("y")
zlabel("z")
```



计算网格上的梯度。

```
fx, fy = TyMath.gradient(f,0.2)
```

提取点 (1,-2) 处的梯度值。为此，首先获取要使用的点的索引。然后，使用索引从 fx 和 fy 中提取相应的梯度值

```
x0 = 1
y0 = -2
t = @. (x == x0 && y == y0)
indt = find(t)
f_grad = [fx[indt] fy[indt]]
```

```
f_grad = 1×2 Matrix{Float64}:
 -16.0  12.04
```

$f(x,y)=x^2y^3$ 在点 (1,-2) 处的确切梯度值为 $(\nabla f)_{(1,-2)} = 2xy^3\hat{i} + 3x^3y^2\hat{j}$
$= -16\hat{i} + 12\hat{j}$

3.2.5.5.13 polyder

多项式微分

A. 语法

```
k = polyder(p)
k = polyder(a,b)[3]
q,d,k = polyder(a,b)
```

B. 说明

k = polyder(p) 返回 p 中的系数表示的多项式的导数，

$$k(x)=\frac{d}{dx}p(x)$$

k = polyder(a,b) 返回多项式 a 和 b 的乘积的导数，

$$k(x)=\frac{d}{dx}[a(x)b(x)]$$

q,d = polyder(a,b)返回多项式 a 和 b 的商的导数，

$$\frac{q(x)}{d(x)}=\frac{d}{dx}\left[\frac{a(x)}{b(x)}\right]$$

C. 示例

(1). 求导多项式

创建一个向量来表示多项式 $p(x)=3x^5-2x^3+x+5$

```
p = [3 ,0, -2 ,0, 1, 5]
```

使用 polyder 对多项式求导。结果为 $q(x)=15x^4-6x^2+1$

```
q = polyder(p)
```

```
5-element Vector{Int64}:

15

0

-6

0

1
```

(2). 求导多项式乘积

创建两个向量来表示多项式

$a(x)=x^4-2x^3+11$ 和 $b(x)=x^2-10x+15$

创建向量来表示多项式 $p(x)=x^5-x^3+1$ 和 $v(x)=x^2+1$

```
a = [1, -2, 0, 0, 11]
b = [1, -10, 15]
```

使用 polyder 来计算 $q(x)=\frac{d}{dx}[a(x)b(x)]$

```
q = polyder(a,b)[3]
```

```
6-element Vector{Int64}:

6

-60

140

-90

22

-110
```

结果为： $q(x)=6x^5-60x^4+140x^3-90x^2+22x-110$

(3). 求导多项式商

创建两个向量来表示商中的多项式，

$$\frac{x^4-3x^2-1}{x+4}$$

```
p = [1,0, -3, 0, -1]
v = [1 ,4]
```

使用包含两个输出参数的 polyder 来计算

$$\frac{q(x)}{d(x)}=\frac{d}{dx}\left[\frac{p(x)}{v(x)}\right]$$

```
q,d = polyder(p,v)
```

```
q =

5-element Vector{Float64}:

3.0

16.0
```

```
-3.0

-24.0

1.0

d =

3-element Vector{Float64}:

1.0

8.0

16.0
```

结果为：$\frac{q(x)}{d(x)}=\frac{3x^4+16x^3-3x^2-24x+1}{x^2+8x+16}$

3.2.5.5.14 polyint

多项式积分

A. 语法

```
q = polyint(p,k)
q = polyint(p)
```

B. 说明

q = polyint(p,k) 使用积分常量 k 返回 p 中系数所表示的多项式积分。

q = polyint(p) 假定积分常量 k = 0。

C. 示例

(1). 对四次多项式求积分

计算定积分

$I=\int_{-1}^{3}\left(3x^4-4x^2+10x-25\right)dx$

创建一个向量，用它来代表多项式，然后计算多项式的根。

创建一个向量来表示多项式被积函数 $3x^4-4x^2+10x-25$。x^3 项不存在，因此系数为 0。

```
p = [3 ,0 ,-4, 10 ,-25];
```

使用 polyint 和等于 0 的积分常量来对多项式求积分。

```
q = polyint(p)
```

```
6-element Vector{Float64}:

0.6

0.0

-1.3333333333333333

5.0

-25.0

0.0
```

通过在积分极限上计算 q，求解积分的值。

```
a = -1
b = 3
I = diff(polyval(q,[a,b]))
```

```
1-element Vector{Float64}:

49.06666666666665
```

(2). 对两个多项式的乘积求积分

计算

$$I = \int_0^2 \left(x^5 - x^3 + 1\right)\left(x^2 + 1\right) dx$$

创建向量来表示多项式 $p(x) = x^5 - x^3 + 1$ 和 $v(x) = x^2 + 1$

```
p = [1,0 ,-1, 0, 0, 1]
v = [1 ,0, 1]
```

多项式相乘，并使用积分常量 $k = 3$ 对所生成的表达式求积分。

```
k = 3
q = polyint(conv(p,v),k)
```

```
9-element Vector{Float64}:

0.125

0.0
```

```
0.0

0.0

-0.25

0.3333333333333333

0.0

1.0

3.0
```

通过在积分极限上计算 q 来求解 I 的值。

```
a = 0
b = 2
I = diff(polyval(q,[a,b]))
```

```
1-element Vector{Float64}:

32.66666666666667
```

3.2.5.5.15 polyder

多项式微分

A. 语法

```
k = polyder(p)
k = polyder(a,b)[3]
q,d,k = polyder(a,b)
```

B. 说明

k = polyder(p) 返回 p 中的系数表示的多项式的导数，

$k(x) = \frac{d}{dx} p(x)$ 示例

k = polyder(a,b) 返回多项式 a 和 b 的乘积的导数，

$k(x) = \frac{d}{dx}[a(x)b(x)]$ 示例

q,d = polyder(a,b)返回多项式 a 和 b 的商的导数，

$\frac{q(x)}{d(x)} = \frac{d}{dx}[\frac{a(x)}{b(x)}]$ 示例。

C. 示例

(1). 求导多项式

创建一个向量来表示多项式 $p(x) = 3x^5 - 2x^3 + x + 5$

```
p = [3 ,0, -2 ,0, 1, 5]
```

使用 polyder 对多项式求导。结果为 $q(x) = 15x^4 - 6x^2 + 1$。

```
q = polyder(p)
```

```
5-element Vector{Int64}:
15
0
-6
0
1
```

(2). 求导多项式乘积

创建两个向量来表示多项式 $a(x) = x^4 - 2x^3 + 11$ 和 $b(x) = x^2 - 10x + 15$

创建向量来表示多项式 $p(x) = x^5 - x^3 + 1$ 和 $v(x) = x^2 + 1$

```
a = [1, -2, 0, 0, 11]
b = [1, -10, 15]
```

使用 polyder 来计算 $q(x) = \frac{d}{dx}[a(x)b(x)]$。

```
q = polyder(a,b)[3]
```

```
6-element Vector{Int64}:
6
-60
140

-90
22
-110
```

结果为： $q(x) = 6x^5 - 60x^4 + 140x^3 - 90x^2 + 22x - 110$

(3). 求导多项式商

创建两个向量来表示商中的多项式，$\frac{x^4-3x^2-1}{x+4}$

```
p = [1,0, -3, 0, -1]
v = [1 ,4]
```

使用包含两个输出参数的 polyder 来计算 $\frac{q(x)}{d(x)}=\frac{d}{dx}[\frac{p(x)}{v(x)}]$。

```
q,d = polyder(p,v)
```

```
q =
5-element Vector{Float64}:
3.0
16.0
-3.0
-24.0
1.0
d =
3-element Vector{Float64}:
1.0
8.0
16.0
```

结果为：$\frac{q(x)}{d(x)}=\frac{3x^4+16x^3-3x^2-24x+1}{x^2+8x+16}$

3.2.6 傅里叶分析和滤波

傅里叶变换、卷积、数字滤波

变换和滤波器是用于处理和分析离散数的工具，常用在信号处理应用和计算数学中。当数据表示为时间或空间的函数时，傅里叶变换会将数据分解为频率分量。

3.2.6.1 傅里叶变换

傅里叶变换简介

函数名	简介
fft	傅里叶变换 (一维、二维、多维)

fftshift	将零频分量移到频谱中心
ifft	傅里叶逆变换（一维、二维、多维）
ifftshift	将零频分量移到频谱中心
nextpow2	2 的更高次幂的指数
nufft	非均匀快速傅里叶变换
nufftn	N 维非均匀快速傅里叶变换
ty_fft	傅里叶变换 (一维、二维、多维)
ty_ifft	傅里叶逆变换（一维、二维、多维）

3.2.6.1.1 fft

傅里叶变换 (一维、二维、多维)

A. 语法

```
Y = fft(X)
Y = fft(X,dims)
```

B. 说明

Y = fft(X) 用快速傅里叶变换 (FFT) 算法计算 X 的离散傅里叶变换 (DFT)。

如果 X 是向量，则 fft(X) 返回该向量的傅里叶变换。

如果 X 是矩阵，则 fft(X) 返回该矩阵的二维傅里叶变换。

如果 X 是一个多维数组，返回该多维数组的 N 维傅里叶变换。

Y = fft(X,dim) 返回对数组 X 沿维度 dim 进行计算傅里叶变换。

C. 示例

(1). 含噪信号

使用傅里叶变换求噪声中隐藏的信号的频率分量。

指定信号的参数，采样频率为 1 kHz，信号持续时间为 1.5 秒。

```
Fs = 1000
T = 1/Fs
L = 1500
t = (0:L-1)*T
```

构造一个信号，其中包含幅值为 0.7 的 50 Hz 正弦量和幅值为 1 的 120 Hz 正弦量。

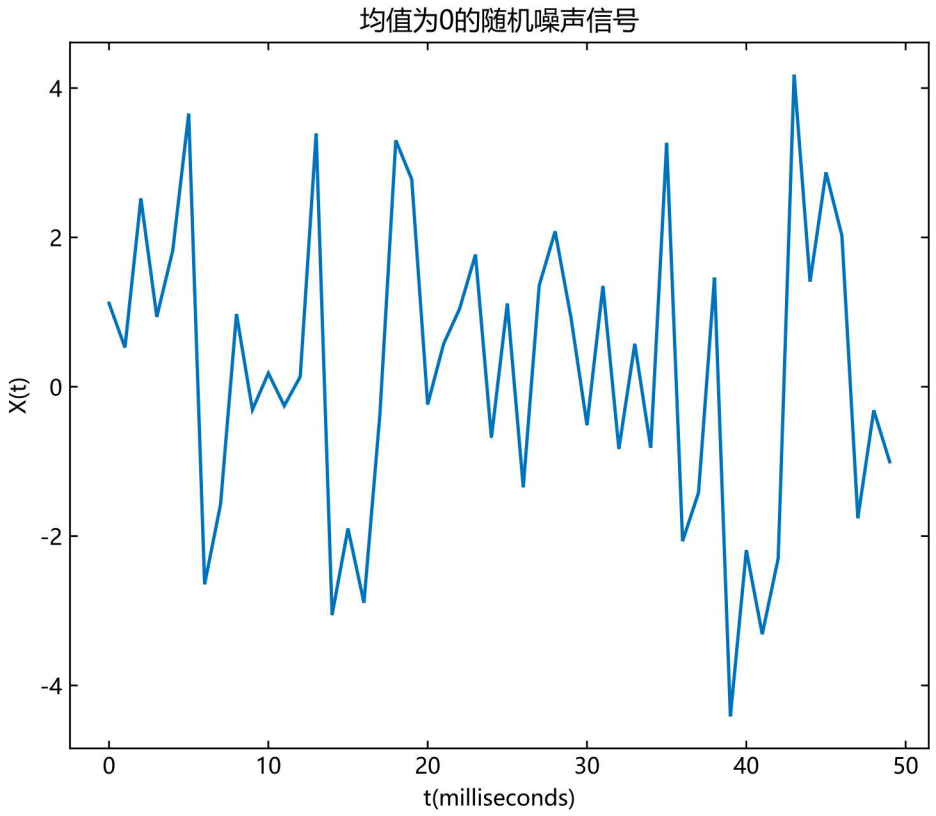
```
S = 0.7*sin.(2*pi*50*t) + sin.(2*pi*120*t)
```

用均值为零、方差为 4 的白噪声扰乱该信号。

```
X = S + 2*randn(size(t))
```

在时域中绘制含噪信号。通过查看信号 X(t) 很难确定频率分量。

```
plot(1000*t[1:50],X[1:50])
title("均值为 0 的随机噪声信号")
xlabel("t(milliseconds)")
ylabel("X(t)")
```



计算信号的傅里叶变换。

```
Y = fft(X)
```

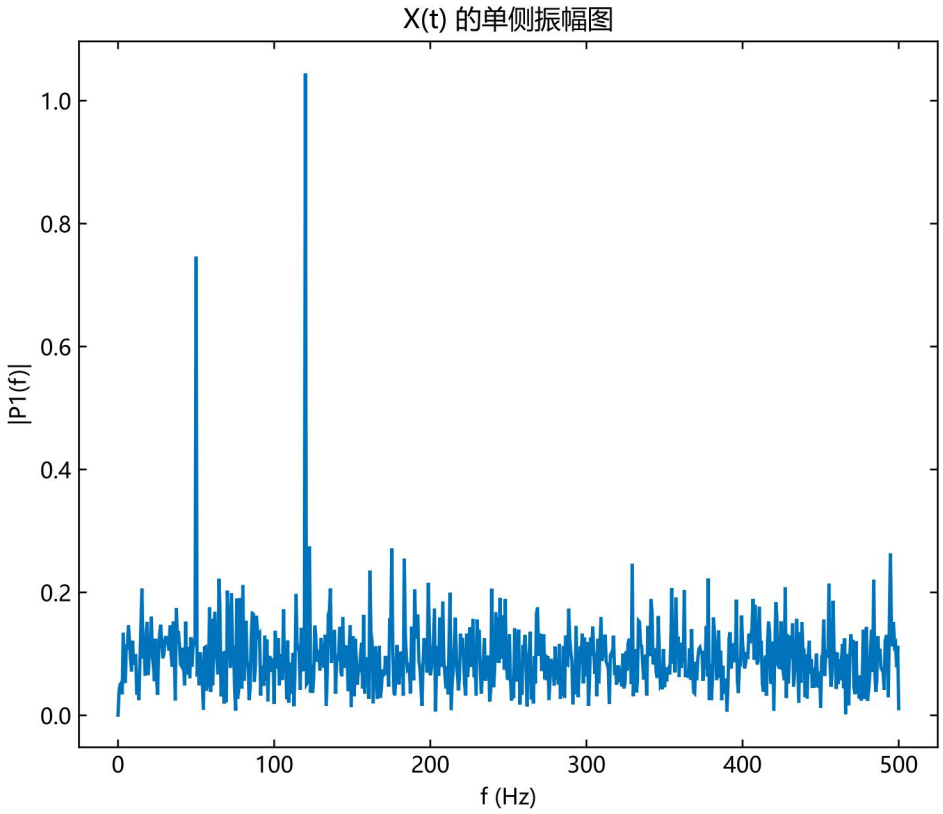
计算双侧频谱 P2。然后基于 P2 和偶数信号长度 L 计算单侧频谱 P1。

```
P2 = abs.(Y/L)
P1 = P2[1:Int(L/2)+1]
P1[2:end-1] = 2*P1[2:end-1]
```

定义频域 f 并绘制单侧幅值频谱 P1。与预期相符，由于增加了噪声，幅值并不精确等于 0.7 和 1。一般情况下，较长的信号会产生更好的频率近似

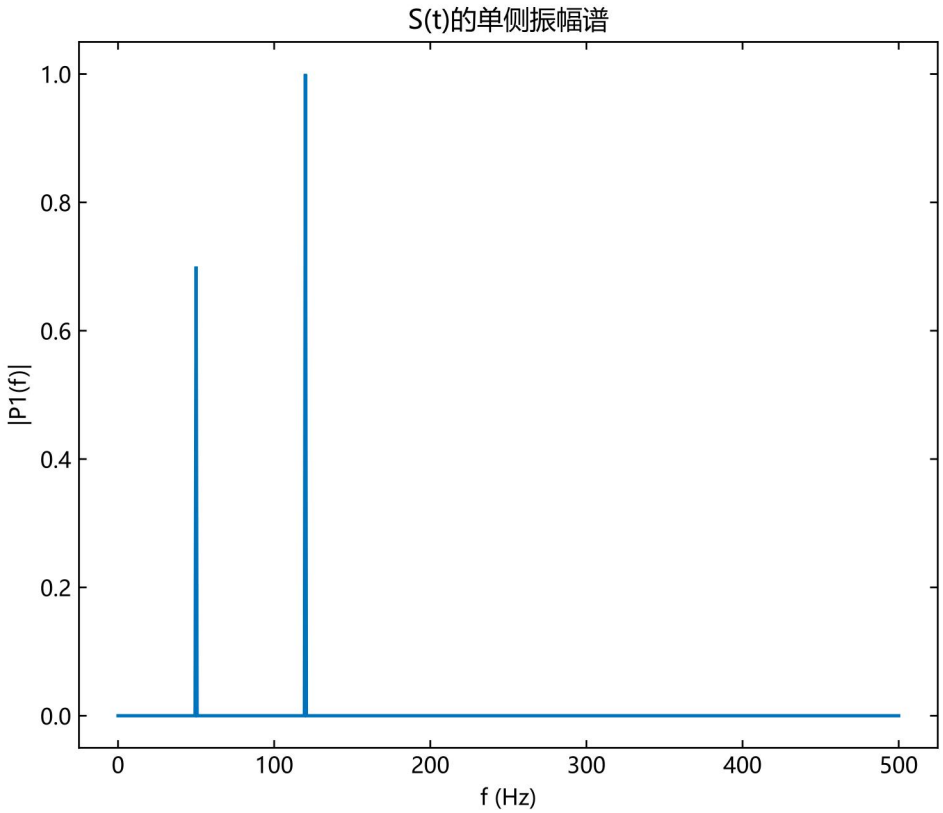
值。

```
f = Fs*(0:(L/2))/L
plot(f,P1)
title("X(t) 的单侧振幅图")
xlabel("f (Hz)")
ylabel("|P1(f)|")
```



现在，采用原始的、未破坏信号的傅里叶变换并检索精确幅值 0.7 和 1.0。

```
Y = fft(S)
P2 = abs.(Y/L)
P1 = P2[1:Int(L/2)+1]
P1[2:end-1] = 2*P1[2:end-1]plot(f,P1)
title("S(t)的单侧振幅谱")
xlabel("f (Hz)")
ylabel("|P1(f)|")
```



(2). 高斯脉冲

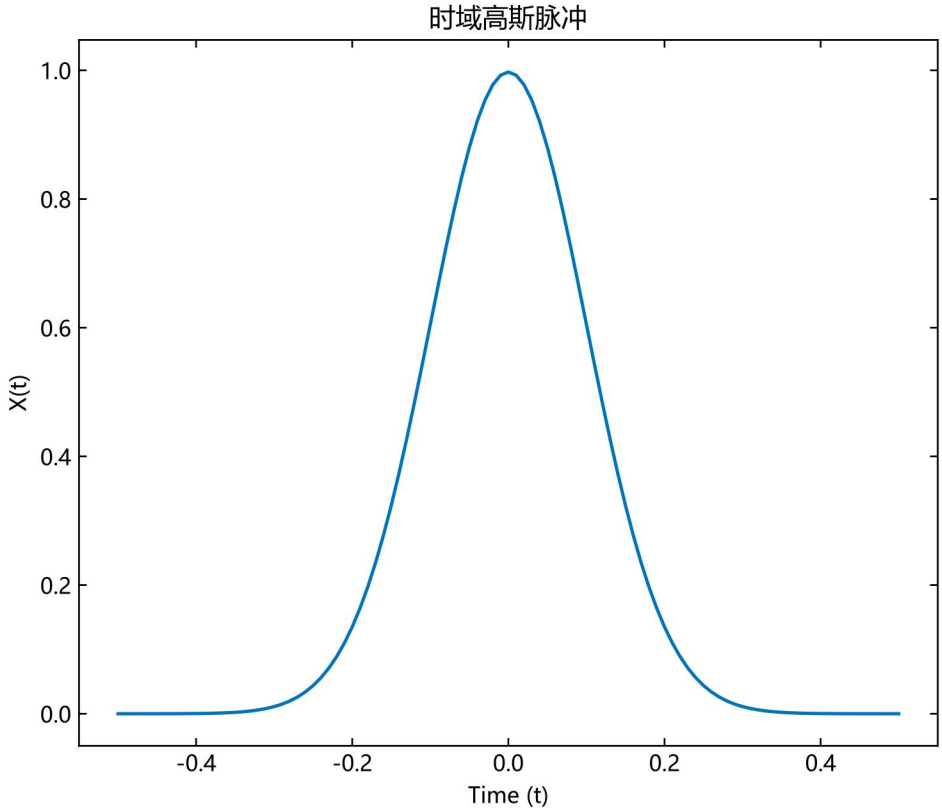
将高斯脉冲从时域转换为频域。

定义信号参数和高斯脉冲 X。

```
Fs = 100
t = -0.5:1/Fs:0.5
L = length(t)
X = 1/(4*sqrt.(2*pi*0.01))*(exp.(-t.^2/(2*0.01)))
```

在时域中绘制脉冲。

```
plot(t,X)
title("时域高斯脉冲")
xlabel("Time (t)")
ylabel("X(t)")
```



要使用 fft 将信号转换为频域，首先从原始信号长度确定是下一个 2 次幂的新输入长度。这将用尾随零填充信号 X 以改善 fft 的性能。

```
n = 2^nextpow2(L)
```

将高斯脉冲转换为频域。

```
X = append!(X,zeros(n-L))
Y = fft(X)
```

定义频域并绘制唯一频率。

```
f = Fs*(0:Int(n/2))/n
P = abs.(Y/n) .^ 2

plot(f,P[1:Int(n/2)+1])
title("频域上的高斯脉冲")
xlabel("Frequency (f)")
ylabel("|P(f)|^2")
```

(3). 余弦波

比较时域和频域中的余弦波。

指定信号的参数，采样频率为 1kHz，信号持续时间为 1 秒。

```
Fs = 1000
T = 1/Fs
L = 1000
t = (0:L-1)*T
```

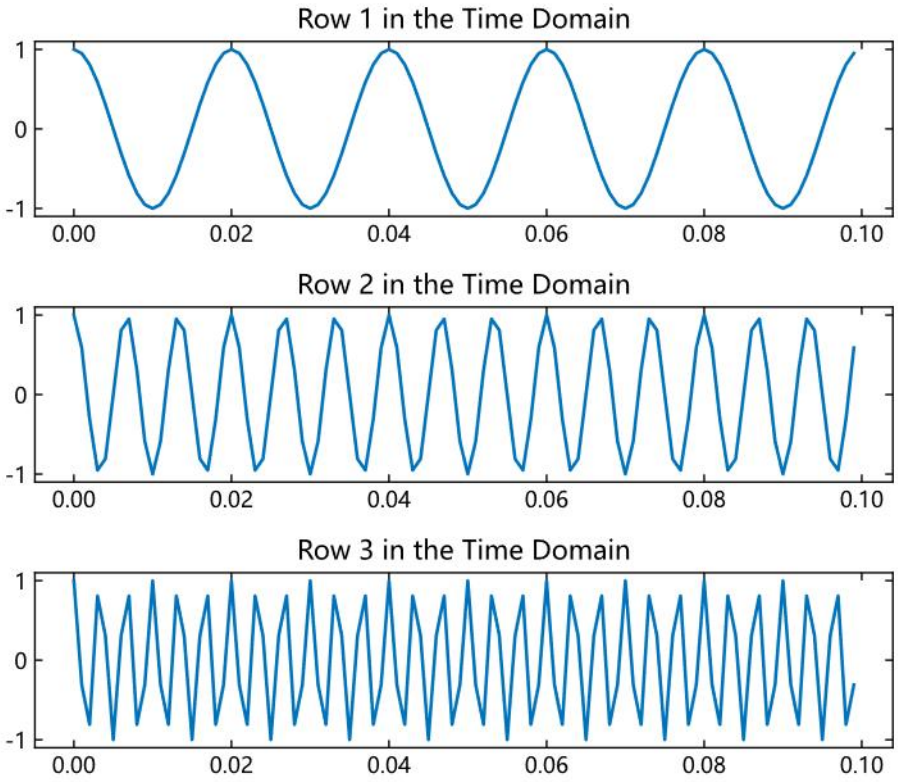
创建一个矩阵，其中每一行代表一个频率经过缩放的余弦波。结果 X 为 3×1000 矩阵。第一行的波频为 50，第二行的波频为 150，第三行的波频为 300。

```
x1 = cos.(2*pi*50*t)
x2 = cos.(2*pi*150*t)
x3 = cos.(2*pi*300*t)

X = hcat(x1,x2,x3)
```

在单个图窗中按顺序绘制 X 的每行的前 100 个项，并比较其频率。

```
for i = 1:3
ax = subplot(3,1,i)
plot(ax,t[1:100],X[1:100,i])
title("Row $i in the Time Domain")
end
```



出于算法性能的考虑，fft 允许您用尾随零填充输入。在这种情况下，用零填充 X 的每一行，以使每行的长度为比当前长度大的下一个最小的 2 的次幂值。使用 nextpow2 函数定义新长度。

```
n = 2^nextpow2(L)
```

指定 dim 参数沿 X 的行（即对每个信号）使用 fft。

```
dim = 1
```

计算信号的傅里叶变换。

```
X = vcat(X,zeros(n-L,3))
Y = fft(X,dim)
```

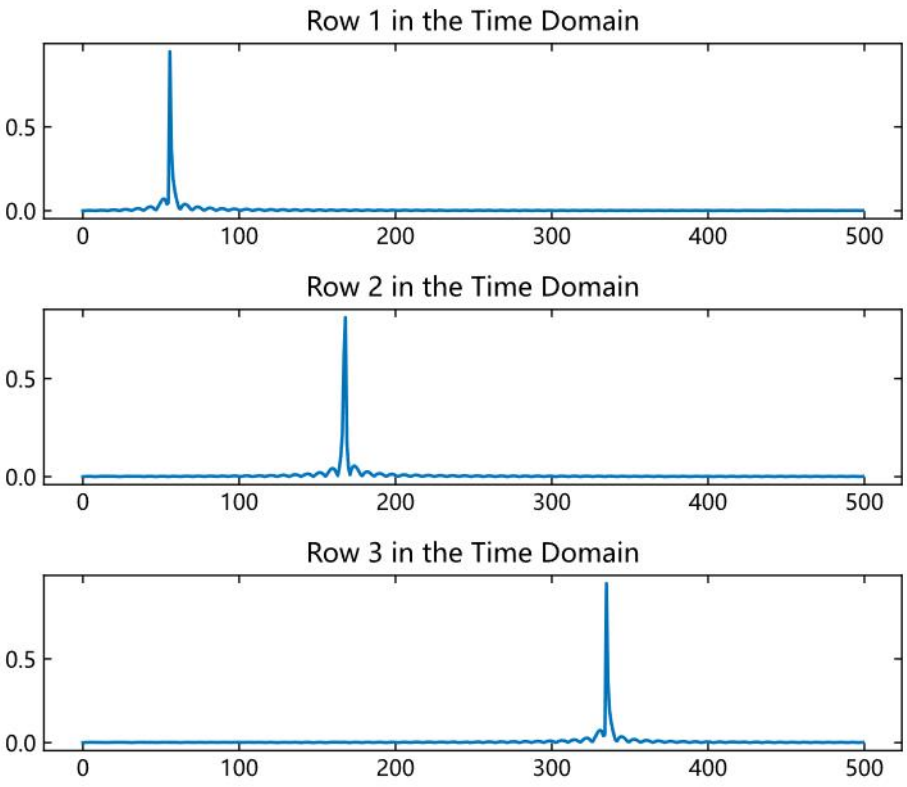
计算每个信号的双侧频谱和单侧频谱。

```
P2 = abs.(Y/L)
P1 = P2[1:Int(n/2)+1,:]
P1[2:end-1,:] = 2*P1[2:end-1,:]
```

在频域内，为单个图窗中的每一行绘制单侧幅值频谱。

```
for i = 1:3
ax = subplot(3,1,i)
plot(ax,0:(Fs/n):(Fs/2-Fs/n),P1[1:Int(n/2),i])
title("Row $i in the Time Domain")
```

```
end
```

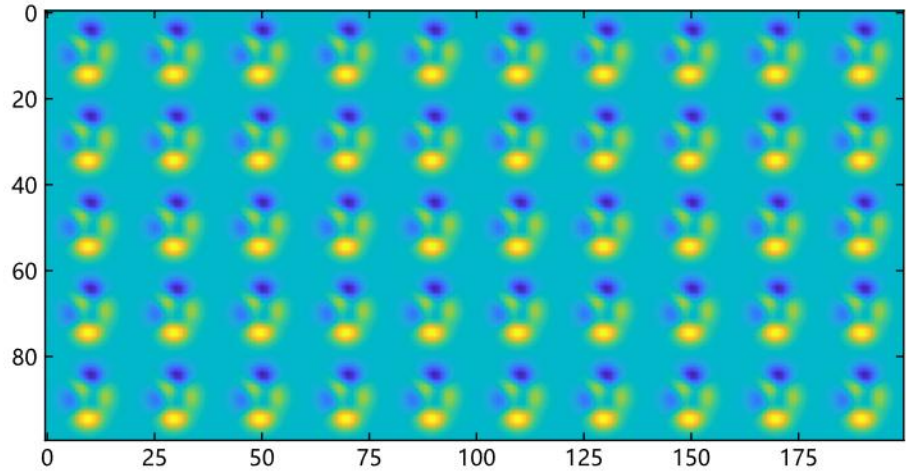


(4). 二维变换

二维傅里叶变换对处理二维信号和其他二维数据（如图像）很有用。

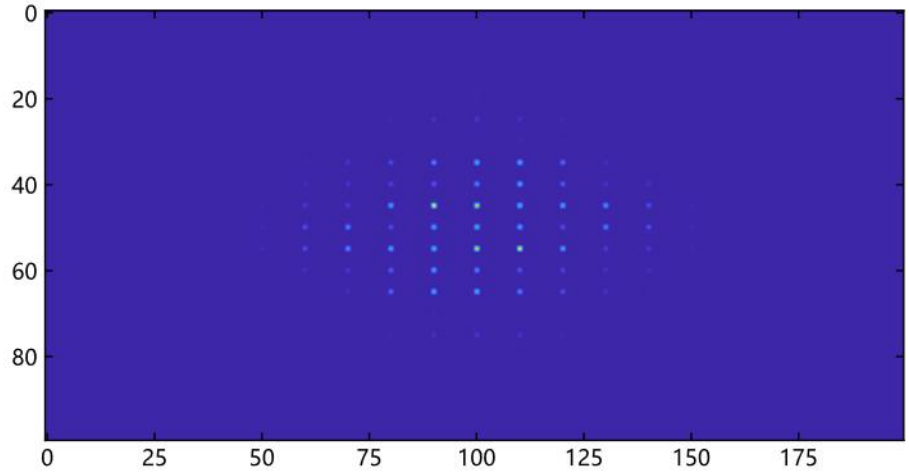
创建并绘制具有重复块的二维数据。

```
P = peaks(20)[3]
X = repeat(P,5,10)
imagesc(X)
```



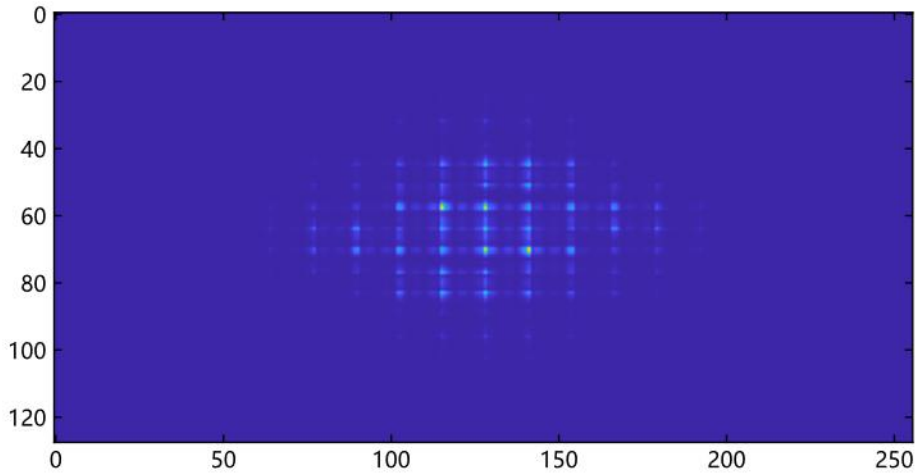
计算数据的二维傅里叶变换。将零频分量移动到输出的中心，并绘制生成的 100×200 矩阵，它与 X 的大小相同。

```
Y = fft(X)
imagesc(abs.(fftshift(Y)))
```



用零填充 X 以计算 128×256 变换。

```
n1 = 2^nextpow2(100)
n2 = 2^nextpow2(200)
X = hcat(X,zeros(100,n2-200))
X = vcat(X,zeros(n1-100,n2))
Y = fft(X)
imagesc(abs.(fftshift(Y)))
```



(5). 三维变换

您可以使用 fftn 函数计算多维数组中每个维度的一维快速傅里叶变换。

创建一个三维信号 X。X 的大小为 20×20×20。

```
x = [1:20;]
y = [1:20;]'
z = reshape(1:20,1,1,20)
X = cos.(2*pi*0.01*x) .+ sin.(2*pi*0.02*y) .+ cos.(2*pi*0.03*z)
```

计算该信号的三维傅里叶变换，这也是一个 20×20×20 数组。

```
Y = fft(X)
```

3.2.6.1.2 fftshift

将零频分量移到频谱中心

A. 语法

```
Y = fftshift(X)
Y = fftshift(X,dim)
```

B. 说明

Y = fftshift(X) 通过将零频分量移动到数组中心，重新排列傅里叶变换 X。

如果 X 是向量，则 fftshift 会将 X 的两半部分进行交换。

如果 X 是矩阵，则 fftshift 会将 X 的第一象限与第三象限交换，将第二象限与第四象限交换。

如果 X 是一个多维数组，则 fftshift 会沿每个维度交换 X 的半空间。

Y = fftshift(X,dim) 沿 X 的维度 dim 执行运算。例如，如果 X 是矩阵，其行表示多个一维变换，则 fftshift(X,2) 会将 X 的每一行的左右两半部分进行交换。

C. 示例

(1). 平移向量元素

交换行向量的两半部分。如果一个向量的元素数为奇数，则中间的元素被视为属于向量的上半部分。

```
Xeven = [1,2,3,4,5,6]
fftshift(Xeven)
```

```
ans = 6-element Vector{Int64}:
4
5
6
1
2
3
```

```
Xodd = [1,2,3,4,5,6,7]
fftshift(Xodd)
```

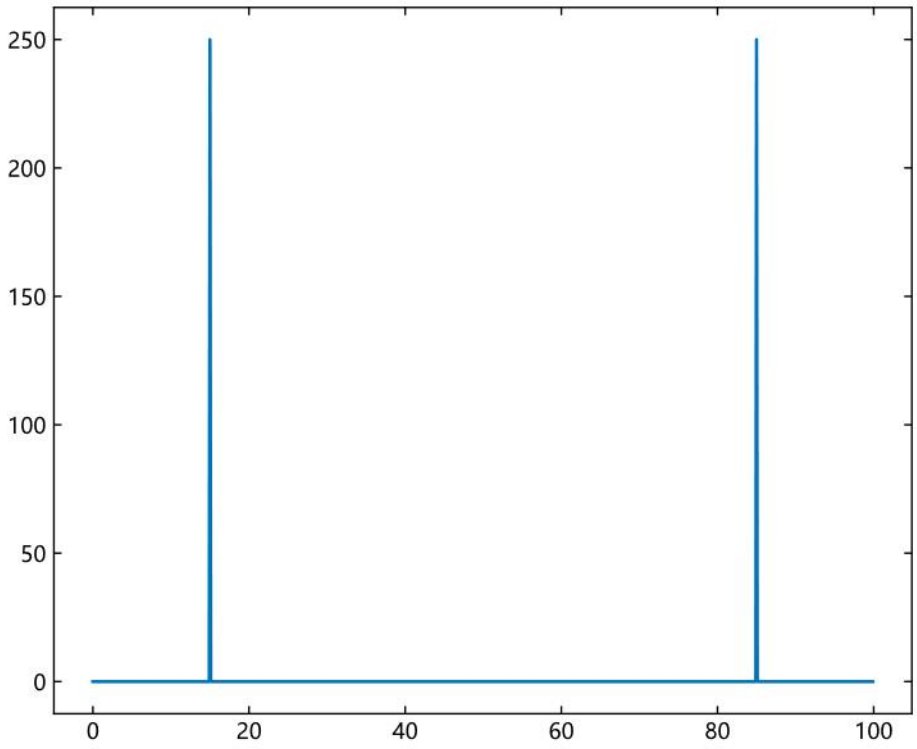
```
ans = 7-element Vector{Int64}:
5
6
7
1
2
3
4
```

(2). 平移一维信号

分析信号的频率分量时，将零频分量平移到中心会很有帮助。

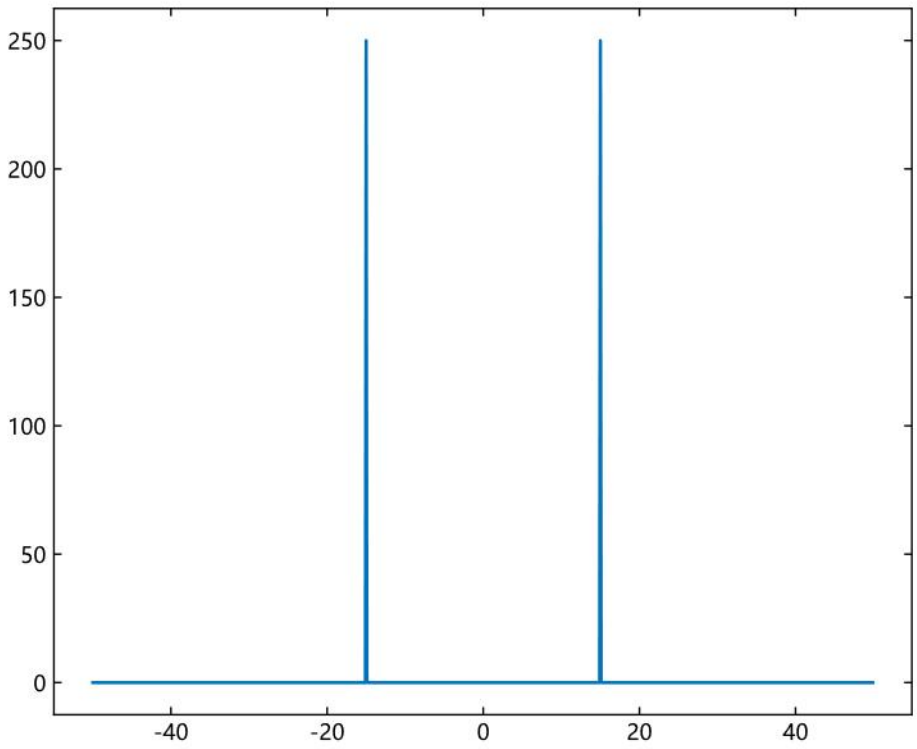
创建信号 S、计算其傅里叶变换，然后绘制功率。

```
fs = 100
t = 0:(1/fs):(10-1/fs)
S = cos.(2*pi*15*t)
n = length(S)
X = fft(S)
f = (0:n-1)*(fs/n)
power = abs.(X) .^2/n
plot(f,power)
```



平移零频分量，然后绘制以零为中心的功率。

```
Y = fftshift(X)
fshift = (-n/2:n/2-1)*(fs/n)
powershift = abs.(Y) .^ 2 /n
plot(fshift,powershift)
```

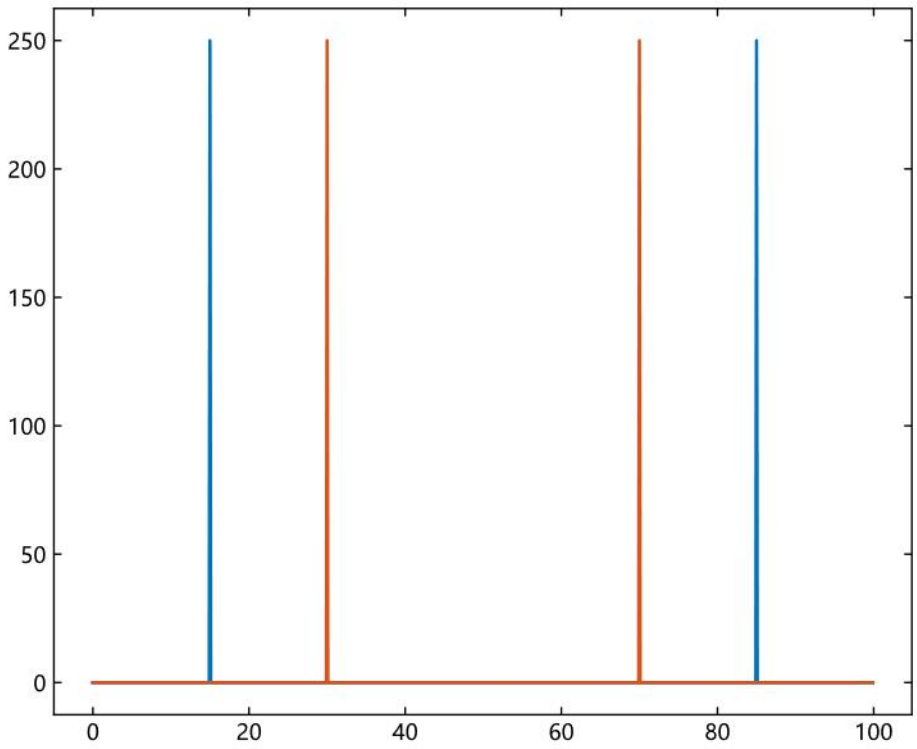


(3). 平移矩阵中的信号

要处理多个一维信号，可将它们表示为一个矩阵的各行。然后使用维度参数计算傅里叶变换，并平移每一行的零频分量。

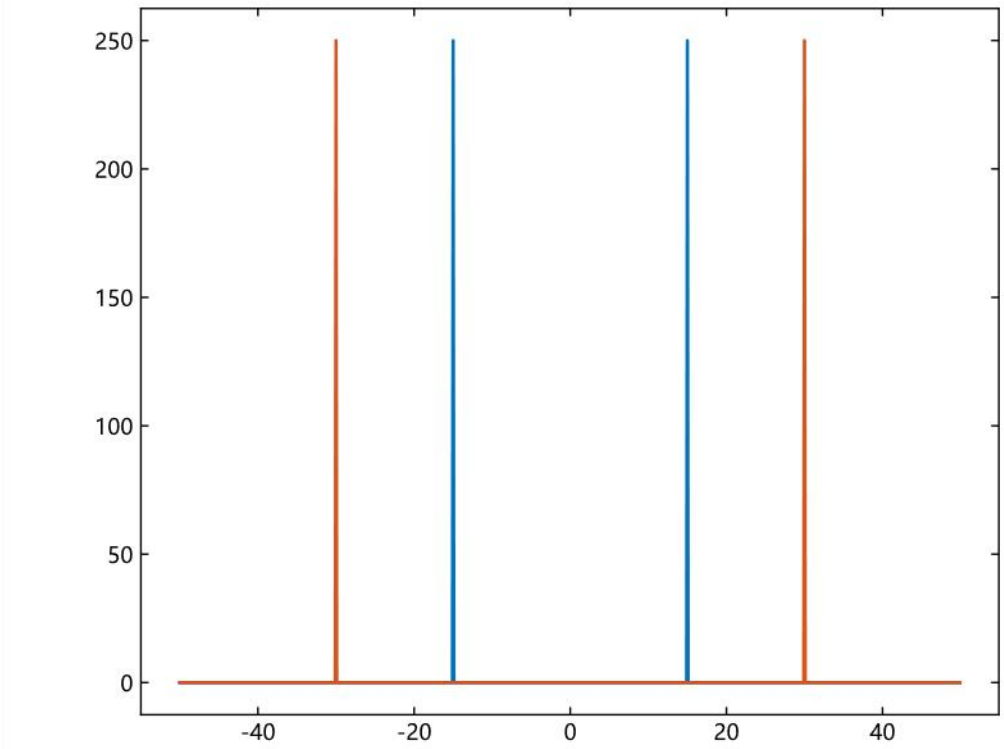
创建矩阵 A，其行表示两个一维信号，然后计算每个信号的傅里叶变换。绘制每个信号的功率。

```
fs = 100
t = 0:(1/fs):(10-1/fs)
S1 = cos.(2*pi*15*t)
S2 = cos.(2*pi*30*t)
n = length(S1)
A = [S1  S2]
X = fft(A,1)
f = (0:n-1)*(fs/n)
power = abs.(X) .^2 /n
plot(f,power[:,1],f,power[:,2])
```



平移零频分量，然后绘制每个信号以零为中心的功率。

```
Y = fftshift(X,1)
fshift = (-n/2:n/2-1)*(fs/n)
powershift = abs.(Y).^2/n
plot(fshift,powershift[:,1],fshift,powershift[:,2])
```



3.2.6.1.3 ifft

傅里叶逆变换（一维、二维、多维）

A. 语法

```
X = ifft(Y)
X = ifft(Y,dim)
```

B. 说明

X = ifft(Y) 使用快速傅里叶变换算法计算 Y 的逆离散傅里叶变换。X 与 Y 的大小相同。

如果 Y 是向量，则 ifft(Y) 返回该向量的逆变换。

如果 Y 是矩阵，则 ifft(Y) 返回对矩阵的二维逆变换。

如果 Y 是多维数组，则 ifft(Y) 返回对多维数组的 N 维逆变换。

X = ifft(Y,dim) 返回沿维度 dim 的傅里叶逆变换。例如，如果 Y 是矩阵，则 ifft(Y,2) 返回每一行的 n 点逆变换。

C. 示例

(1). 向量的逆变换

时空采样数据与频率采样数据间的傅里叶变换及其逆变换。

创建一个向量并计算其傅里叶变换。

```
X = [1,2,3,4,5]
Y = fft(X)
```

```
Y = 5-element Vector{ComplexF64}:
15.0 + 0.0im
-2.5 + 3.440954801177934im
-2.5 + 0.8122992405822659im
-2.5 - 0.8122992405822659im
-2.5 - 3.440954801177934im
```

计算 Y 的逆变换，结果与原始向量 X 相同。

```
ifft(Y)
```

```
ans = 5-element Vector{ComplexF64}:
1.0 + 0.0im
2.0 + 0.0im
3.0 + 0.0im
4.0 + 0.0im
5.0 + 0.0im
```

(2). 矩阵的二维逆变换

您可以使用 ifft 函数将按频率采样的二维信号转换为按时间或空间采样的信号。

创建一个 3×3 矩阵并计算其傅里叶变换。

```
X = magic(3)
```

```
3×3 Matrix{Int64}:
8     1     6
3     5     7
4     9     2
```

```
Y = fft(X)
```

```
Y = 3×3 Matrix{ComplexF64}:
45.0+0.0im                                                    0.0+0.0im
0.0+0.0im
0.0+2.22045e-16im           13.5+7.79423im                8.88178e-16-
5.19615im
0.0-2.22045e-16im        8.88178e-16+5.19615im        13.5-7.79423im
```

计算 Y 的逆变换，结果与原始矩阵 X 相同（基于舍入误差）。

```
ifft(Y)
```

```
ans = 3×3 Matrix{ComplexF64}:
8.0+0.0im     1.0+0.0im     6.0+0.0im
3.0+0.0im     5.0+0.0im     7.0+0.0im
4.0+0.0im     9.0+0.0im     2.0+0.0im
```

(3). 三维逆变换

您可以使用 ifft 函数将按频率采样的三维信号转换为按时间或空间采样的信号。

创建一个 3×3×3 数组并计算其傅里叶变换。

```
X = reshape(1:27,3,3,3)
Y = fft(X)
```

```
3×3×3 Array{ComplexF64, 3}:
[:, :, 1] =
378.0+0.0im     -40.5+23.3827im  -40.5-23.3827im
-13.5+7.79423im    0.0+0.0im       0.0+0.0im
-13.5-7.79423im    0.0+0.0im       0.0+0.0im
[:, :, 2] =
-121.5+70.1481im  0.0+0.0im    0.0+0.0im
0.0+0.0im      0.0+0.0im   0.0+0.0im
0.0+0.0im      0.0+0.0im   0.0+0.0im
[:, :, 3] =
-121.5-70.1481im   0.0+0.0im   0.0+0.0im
0.0+0.0im        0.0+0.0im     0.0+0.0im
0.0+0.0im        0.0+0.0im     0.0+0.0im
```

计算 Y 的逆变换，结果与原始矩阵 X 相同。

```
ifft(Y)
```

```
ans = 3×3×3 Array{ComplexF64, 3}:
[:, :, 1] =
1.0+0.0im    4.0+0.0im    7.0+0.0im
2.0+0.0im    5.0+0.0im    8.0+0.0im
3.0+0.0im    6.0+0.0im    9.0+0.0im
[:, :, 2] =
10.0+0.0im    13.0+0.0im    16.0+0.0im
11.0+0.0im    14.0+0.0im    17.0+0.0im
12.0+0.0im    15.0+0.0im    18.0+0.0im
[:, :, 3] =
19.0+0.0im    22.0+0.0im    25.0+0.0im
20.0+0.0im    23.0+0.0im    26.0+0.0im
21.0+0.0im    24.0+0.0im    27.0+0.0im
```

3.2.6.1.4 ifftshift

将零频分量移到频谱中心

A. 语法

```
X = ifftshift(Y)
X = ifftshift(Y,dim)
```

B. 说明

X = ifftshift(Y) 将进行过零频平移的傅里叶变换 Y 重新排列回原始变换输出的样子。换言之，ifftshift 就是撤消 fftshift 的结果。

如果 Y 是向量，则 ifftshift 会将 Y 的两半部分进行交换。

如果 Y 是矩阵，则 ifftshift 会将 Y 的第一象限与第三象限交换，将第二象限与第四象限交换。

如果 Y 是多维数组，则 ifftshift 会沿每个维度交换 Y 的半空间。

X = ifftshift(Y,dim) 沿 Y 的维度 dim 执行运算。例如，如果 Y 是矩阵，其行表示多个一维变换，则 ifftshift(Y,2) 会将 Y 的每一行的左右两半部分进行交换。

C. 示例

(1). 逆平移向量元素

您可以使用 fftshift 和 ifftshift 函数交换向量的两半部分。

创建包含奇数个元素的向量，然后使用 fftshift 函数交换该向量的两侧。

```
V = [1,2,3,4,5,6,7]
X = fftshift(V)
```

```
X = 7-element Vector{Int64}:
5
6
7
1
2
3
4
```

使用 ifftshift 函数交换 X 的上下两侧。结果与原始向量 V 相同。

```
Y = ifftshift(X)
```

```
Y = 7-element Vector{Int64}:
```

```
1
2
3
4
5
6
7
```

调用 fftshift 函数两次并不一定会重新构造原始输入。

```
fftshift(fftshift(V))
```

```
ans = 7-element Vector{Int64}:
2
3
4
5
6
7
1
```

(2). 逆平移矩阵行

重新排列矩阵行，将非负元素平移到左侧。

```
Y = [-2 -1 0 1 2; -10 -5 0 5 10]
X = ifftshift(Y,2)
```

```
X = 2×5 Matrix{Int64}:
0   1    2   -2   -1
0   5   10  -10   -5
```

3.2.6.1.5 nextpow2

2 的更高次幂的指数

A. 语法

```
P = nextpow2(A)
```

B. 说明

P = nextpow2(A) 返回对 A 中每个元素满足

$$2^p \geq |A|$$

的最小的 2 的幂的指数。按照惯例，nextpow2(0) 返回零。

您可以使用 nextpow2 填充传递到 fft 的信号。当信号长度并非 2 次幂时，这样做可以加快 FFT 的运算速度。

C. 示例

(1). 下一个整数值的 2 次幂

定义一个 Int64 整数值向量，并计算下一个 2 次幂大于这些值的指数。

```
a = [1,-2,3,-4,5,9,519]
p = nextpow2.(a)
```

```
p = 7-element Vector{Int64}:
0
1
2
2
3
4
10
```

计算正的 2 的更高次幂。

```
np2 = 2 .^ p
```

```
np2 = 7-element Vector{Int64}:
1
2
4
4
8
16
1024
```

保留原始输入值的符号。

```
np2.*sign.(a)
```

```
ans = 7-element Vector{Int64}:
1
-2
4
-4
8
16
1024
```

(2). 无符号整数值的 2 更高次幂

定义一个无符号整数向量，并计算 2 的更高次幂大于这些值的指数。

```
a = UInt32.([1020,4000,32700])
p = nextpow2.(a)
```

```
p = 3-element Vector{Int64}:
10
12
15
```

计算下一个大于 a 中值的 2 次幂。

```
2 .^ p
```

```
ans = 3-element Vector{Int64}:
1024
4096
32768
```

(3). 带填充的优化 FFT

当信号长度不等于 2 的幂时，使用 nextpow2 函数可提升 fft 的性能。

创建包含 8191 个样本值的一维向量。

```
x = rand(1,8191)
```

计算 2 的更高次幂中大于 8191 的值。

```
p = nextpow2(8191)
n = 2^p
```

```
n = 8192
```

将信号和 2 的更高次幂传递给 fft 函数。

```
x = append!(vec(x),zeros(n-8191))
y = fft(x)
```

3.2.6.1.6 nufft

非均匀快速傅里叶变换

A. 语法

```
Y = nufft(X, t)
Y = nufft(X, t, f)
Y = nufft(X, t, f, dim)
Y = nufft(X)
```

B. 说明

Y = nufft(X,t) 使用采样点 t 返回 X 的非均匀离散傅里叶变换 (NUDFT)。示例

如果 X 是向量，则 nufft 返回该向量的变换。

如果 X 是矩阵，则 nufft 将 X 的各列视为向量，并返回每列的变换。

如果 X 是一个多维数组，则 nufft 将沿大小不等于 1 的第一个数组维度的值视为向量，并返回每个向量的变换。

Y = nufft(X,t,f) 使用采样点 t 计算查询点 f 的 NUDFT。要指定 f 而不指定采样点，请使用 nufft(X,[],f)。示例

Y = nufft(X,t,f,dim) 返回沿维度 dim 的 NUDFT。例如，nufft(X,t,f,2) 计算矩阵 X 的每行的变换。

Y = nufft(X) 返回 X 的离散傅里叶变换，等效于 fft(X)。

C. 示例

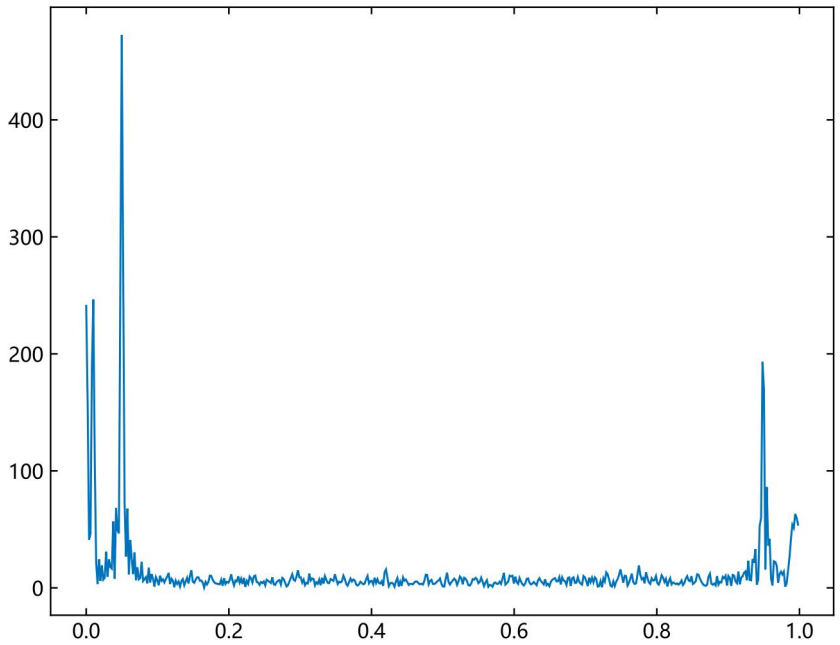
(1). 非均匀采样点

创建在非均匀分布的点 t 上采样的信号 X。计算非均匀快速傅里叶变换 Y。

```
t = [0:300; 500.5:700.5]
S = @. 2*sin(0.1*pi*t) + sin(0.02*pi*t)
rng = MersenneTwister(1234)
X = similar(S)
for i in eachindex(S)
    X[i] = S[i] + rand(rng)
end
Y = nufft(X,t)
```

将变换的绝对值绘制为默认频率的函数。

```
n = length(t);
f = (0:n-1)./n;
plot(f,abs.(Y))
```



(2). 音乐和弦的频率

定义和标注一系列音乐音调的频率。

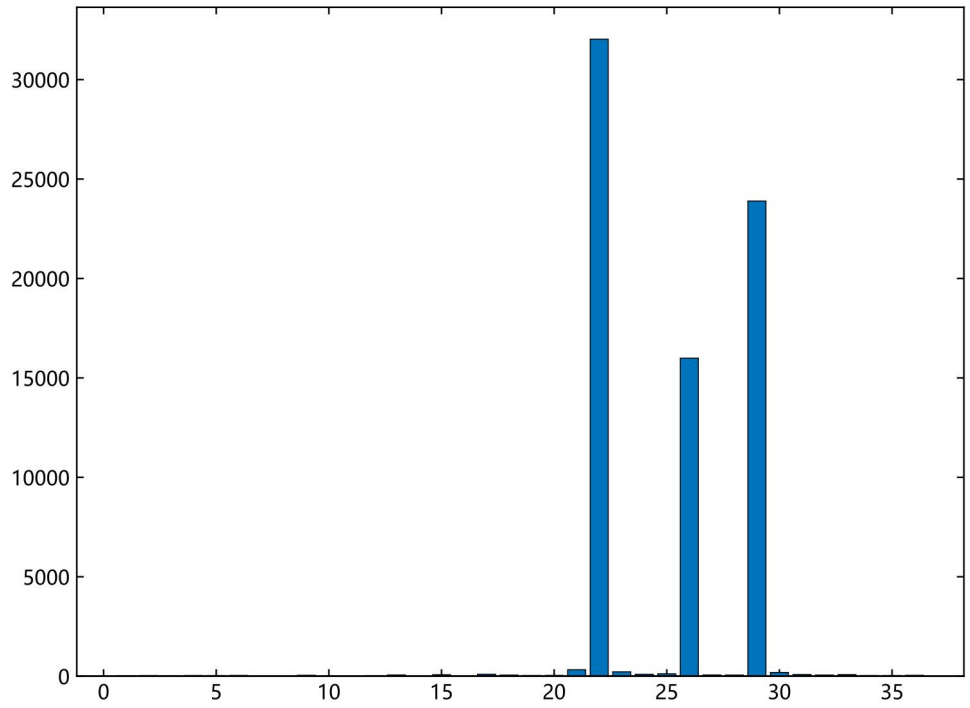
```
C3 = 440 / (2^(21 / 12))
nOctaves = 3
musicalTones = C3 * 2 .^ ((0:(12 * nOctaves - 1)) / 12)
```

定义以 Hz 为单位的音频信号采样频率，采样点为 n 个，以及包含主和弦 X 的信号。

```
fs = 16e3
n = 1:16000
X = @. 4 * cos(2 * pi * (440 / fs) * n) +
    2 * cos(2 * pi * (554.37 / fs) * n) +
    3 * cos(2 * pi * (659.2 / fs) * n)
```

计算并绘制主和弦的频率分量。

```
Y = nufft(X, [], musicalTones / fs)
bar(abs.(Y))
```



3.2.6.1.7 nufftn

n 维非均匀快速傅里叶变换

A. 语法

```
Y = nufftn(X, t)
Y = nufftn(X, t, f)
Y = nufftn(X)
```

B. 说明

Y = nufftn(X,t) 使用采样点 t 返回沿 N 维数组 X 的每个维度的非均匀离散傅里叶变换 (NUDFT)。示例

Y = nufftn(X,t,f) 使用采样点 t 和查询点 f 计算 NUDFT。要指定 f 而不指定采样点，请使用 nufftn(X,[],f)。

Y = nufftn(X) 返回 X 的 N 维离散傅里叶变换。

C. 示例

(1). 非均匀采样点

创建一个在每个维度上的非均匀分布点 t 处采样的三维信号 X。计算非均匀快速傅里叶变换 Y。

```
t = [1:10; 11:2:29]'
X = zeros(20, 20, 20)
for k in axes(X, 3)
    for j in axes(X, 2)
        for i in axes(X, 1)
            X[i, j, k] =
                cos(2 * pi * 0.01 * t[i]) +
                sin(2 * pi * 0.02 * t[j]) +
                cos(2 * pi * 0.03 * t[k])
        end
    end
end
Y = nufftn(X, (t, t, t))
```

3.2.6.1.8 ty_fft

快速傅里叶变换

A. 语法

```
Y = ty_fft(X)
Y = ty_fft(X,n)
Y = ty_fft(X,dims)
```

B. 说明

Y = ty_fft(X) 用快速傅里叶变换 (FFT) 算法计算 X 的离散傅里叶变换 (DFT)。示例

如果 X 是向量，则 ty_fft(X) 返回该向量的傅里叶变换。

如果 X 是矩阵，则 ty_fft(X) 将 X 的各列视为向量，并返回每列的傅里叶变换。

如果 X 是一个多维数组，则 ty_fft(X) 将沿大小不等于 1 的第一个数组维度的值视为向量，并返回每个向量的傅里叶变换。

Y = ty_fft(X,n) 返回 n 点 DFT。如果未指定任何值，则 Y 的大小与 X 相同。示例

如果 X 是向量且 X 的长度小于 n，则为 X 补上尾零以达到长度 n。

如果 X 是向量且 X 的长度大于 n，则对 X 进行截断以达到长度 n。

如果 X 是矩阵，则每列的处理与在向量情况下相同。

如果 X 为多维数组，则大小不等于 1 的第一个数组维度的处理与在向量情况下相同。

Y = ty_fft(X,n,dim) 返回沿维度 dim 的傅里叶变换。例如，如果 X 是矩阵，则 ty_fft(X,n,2) 返回每行的 n 点傅里叶变换。

C. 示例

(1). FFT 的插值

通过填充零来对信号的傅里叶变换进行插值。

指定信号的参数，采样频率为 80 Hz，信号持续时间为 0.8 秒。

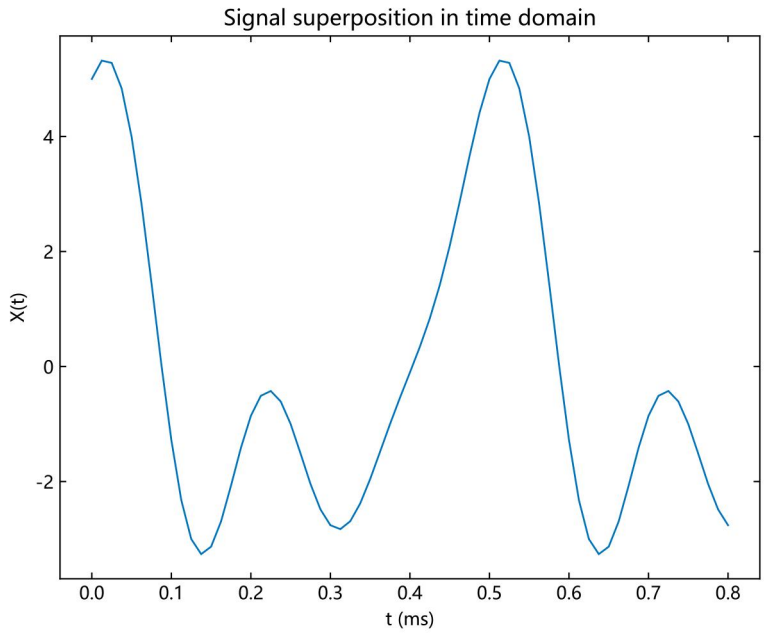
```
Fs = 80
T = 1/Fs
L = 65
t = (0:L-1)*T
```

创建一个 2 Hz 正弦信号及其高次谐波的叠加。该信号包含一个 2 Hz 余弦波、一个 4 Hz 余弦波和一个 6 Hz 正弦波。

```
X = @. 3*cos(2*π*2*t) + 2*cos(2*π*4*t) + sin(2*π*6*t);
```

在时域中绘制该信号。

```
plot(t,X)
title("Signal superposition in time domain")
xlabel("t (ms)")
ylabel("X(t)")
```

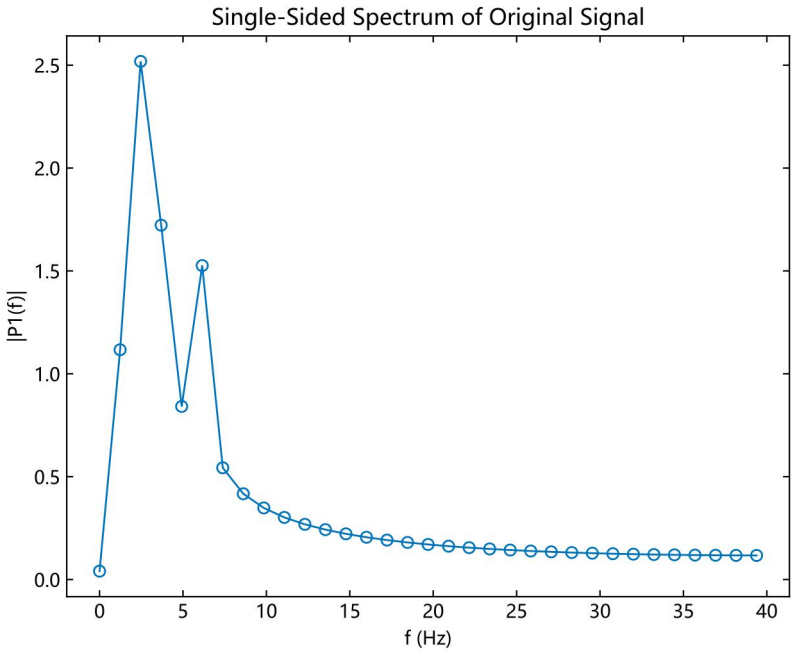


计算信号的傅里叶变换。

```
f = Fs*(0:(L-1)/2)/L;
P2 = abs.(Y/L);
P1 = P2[1:(L+1)÷2];
P1[2:end] = 2*P1[2:end];
```

在频域中绘制单侧频谱。由于信号的时间采样相当短，傅里叶变换的频率分辨率不够精确，不足以显示 4 Hz 附近的峰值频率。

```
plot(f,P1,"-o")
title("Single-Sided Spectrum of Original Signal")
xlabel("f (Hz)")
ylabel("|P1(f)|")
```



为了更好地评估峰值频率，您可以通过用零填充原始信号来增加分析窗的长度。这种方法以更精确的频率分辨率自动对信号的傅里叶变换进行插值。从原始信号长度确定是下一个 2 次幂的新输入长度。用尾随零填充信号 X 以扩展其长度。计算填零后的信号的傅里叶变换。

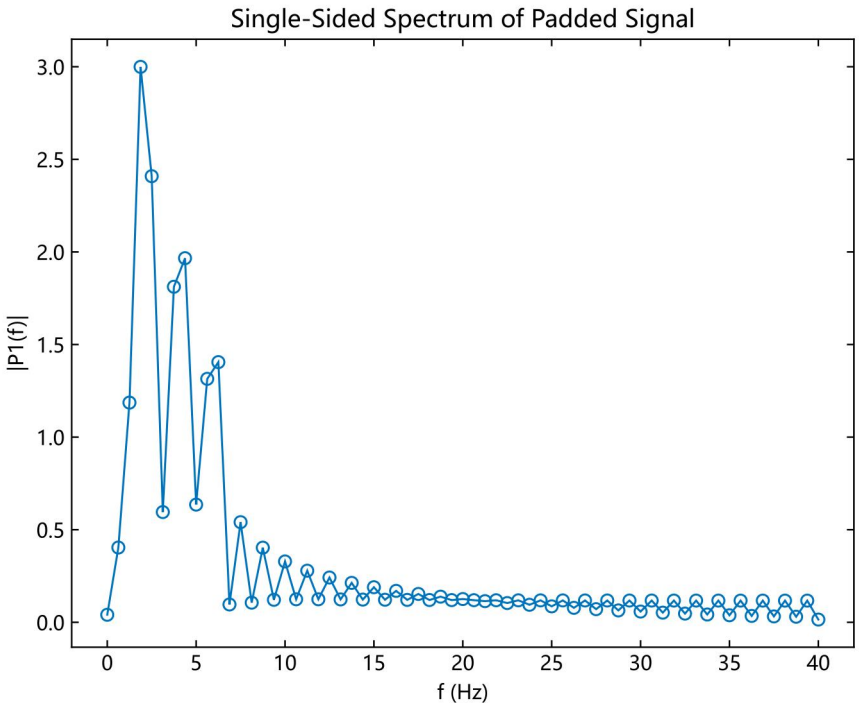
```
n = nextpow(2,L);
Y = ty_fft(X,n);
```

计算填零后的信号的单侧幅值频谱。由于信号长度 n 从 65 增加到 128，频率分辨率变为 Fs/n，即 0.625 Hz。

```
f = Fs*(0:(n/2))/n;
P2 = abs.(Y/L);
P1 = P2[1:n÷2+1];
P1[2:end-1] = 2*P1[2:end-1];
```

绘制填零后的信号的单侧频谱。此新频谱在 0.625 Hz 的频率分辨率内显示 2 Hz、4 Hz 和 6 Hz 附近的峰值频率。

```
plot(f,P1,"-o")
title("Single-Sided Spectrum of Padded Signal")
xlabel("f (Hz)")
ylabel("|P1(f)|")
```



3.2.6.1.9 ty_ifft

快速傅里叶逆变换

A. 语法

```
X = ty_ifft(Y)
X = ty_ifft(Y,n)
X = ty_ifft(Y,dims)
```

B. 说明

X = ty_ifft(Y) 使用快速傅里叶变换算法计算 Y 的逆离散傅里叶变换。X 与 Y 的大小相同。示例

如果 X 是向量，则 ty_ifft(X) 返回该向量的逆变换。

如果 X 是矩阵，则 ty_ifft(X) 返回该矩阵每一列的逆变换。

如果 X 是一个多维数组，则 ty_ifft(X) 将大小不等于 1 的第一个维度上的值视为向量，并返回每个向量的逆变换。

X = ty_ifft(Y,n) 通过用尾随零填充 Y 以达到长度 n，返回 Y 的 n 点傅里叶逆变换示例

X = ty_ifft(Y,n,dim) 回沿维度 dim 的傅里叶逆变换。例如，如果 Y 是矩阵，则 ty_ifft(Y,n,2) 返回每一行的 n 点逆变换。示例。

C. 示例

(1). 向量的逆变换

时空采样数据与频率采样数据间的傅里叶变换及其逆变换。

创建一个向量并计算其傅里叶变换。

```
X = [1 2 3 4 5];
Y = ty_fft(X)
```

```
Y = 1×5 Matrix{ComplexF64}:
15.0+0.0im   -2.5+3.44095im   -2.5+0.812299im   -2.5-0.812299im   -2.5-3.44095im
```

计算 Y 的逆变换，结果与原始向量 X 相同。

```
ty_ifft(X)
```

```
1×5 Matrix{ComplexF64}:
1.0+0.0im  2.0+0.0im  3.0+0.0im  4.0+0.0im  5.0+0.0im
```

(2). 填充的矩阵逆变换

ty_ifft 函数允许您控制变换的大小。

创建一个随机的 3×5 矩阵，并计算每一行的 8 点傅里叶逆变换。每一行结果的长度均为 8。

```
Y = rand(3,5);
n = 8;
X = ty_ifft(Y,n,2);
size(X)
```

```
(3,8)
```

3.2.6.2 滤波

数字滤波器

函数名	简介
filter1	一 维数字滤波器
filter2	二 维数字滤波器

3.2.6.2.1 filter1

1 维数字滤波器

A. 语法

```
y = filter(b,a,x)
y = filter(b,a,x,zi)
y = filter(b,a,x,zi,dim)
[y,zf] = filter(___)
```

B. 说明

y = filter(b,a,x) 使用由分子和分母系数 b 和 a 定义的有理传递函数对输入数据 x 进行滤波。

如果 a[1] 不等于 1，则 filter 按 a[1] 对滤波器系数进行归一化。因此，a[1] 必须是非零值。

如果 x 为向量，则 filter 将滤波后数据以大小与 x 相同的向量形式返回。

如果 x 为矩阵，则 filter 沿着第一维度操作并返回每列的滤波后的数据。

如果 x 为多维数组，则 filter 沿大小不等于 1 的第一个数组维度进行计算。

y = filter(b,a,x,zi) 将初始条件 zi 用于滤波器延迟。zi 的长度必须等于 max(length(a),length(b))-1。

y = filter(b,a,x,zi,dim) 沿维度 dim 进行计算。例如，如果 x 为矩阵，则 filter(b,a,x,zi,2) 返回每行滤波后的数据。

[y,zf] = filter(___) 还使用任一上述语法返回滤波器延迟的最终条件 zf。

C. 示例

(1). 移动平均滤波器

移动平均滤波器是用于对含噪数据进行平滑处理的常用方法。此示例使用 filter 函数计算沿数据向量的平均值。

创建一个由正弦曲线数据组成的 1×100 行向量，其中的正弦曲线被随机干扰所损坏。

导入 filter1_test1_data.jl 文件。

```
pkg_dir = pkgdir(TyMath)
source_path = pkg_dir * "/examples/Fourier Analysis And Filtering/Filter/filter1_tests/filter1_test1_data.jl"
include(source_path)
t = LinRange(-pi, pi, 100);
```

移动平均值滤波器沿数据移动长度为 的窗口，并计算每个窗口中包含的数据的平均值。以下差分方程定义向量 的移动平均值滤波器：

$$y(n) = \frac{1}{windowSize}(x[n] + x[n-1] + \ldots + x[n-(windowSize-1)])$$

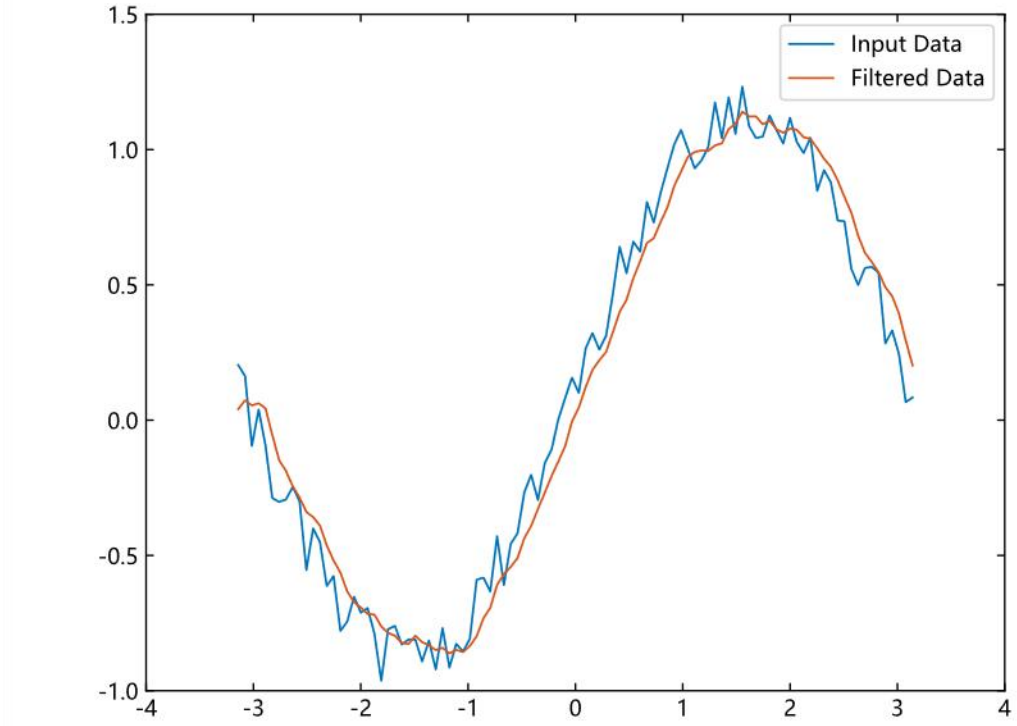
窗口大小为 5 时，计算有理传递函数的分子和分母系数。

```
windowSize = 5;
b = (1 / windowSize) * ones(1, windowSize);
a = 1;
```

求数据的移动平均值，并绘制其对原始数据的图。

```
y1, y2 = filter1(b', a, x)

plot(t, x)
hold("on")
plot(t, y1)
legend(["Input Data", "Filtered Data"])
```



(2). 对矩阵行进行滤波

此示例使用以下有理传递函数对数据的矩阵进行滤波。

导入 filter1_test2_data.jl 文件。

$$H(z) = \frac{b[1]}{a[1]+a[2]z^{-1}} = \frac{1}{1-0.2z^{-1}}$$

创建一个由随机输入数据组成的 2 × 15 矩阵，此处数据由 filter1_test2_data.jl 导入。

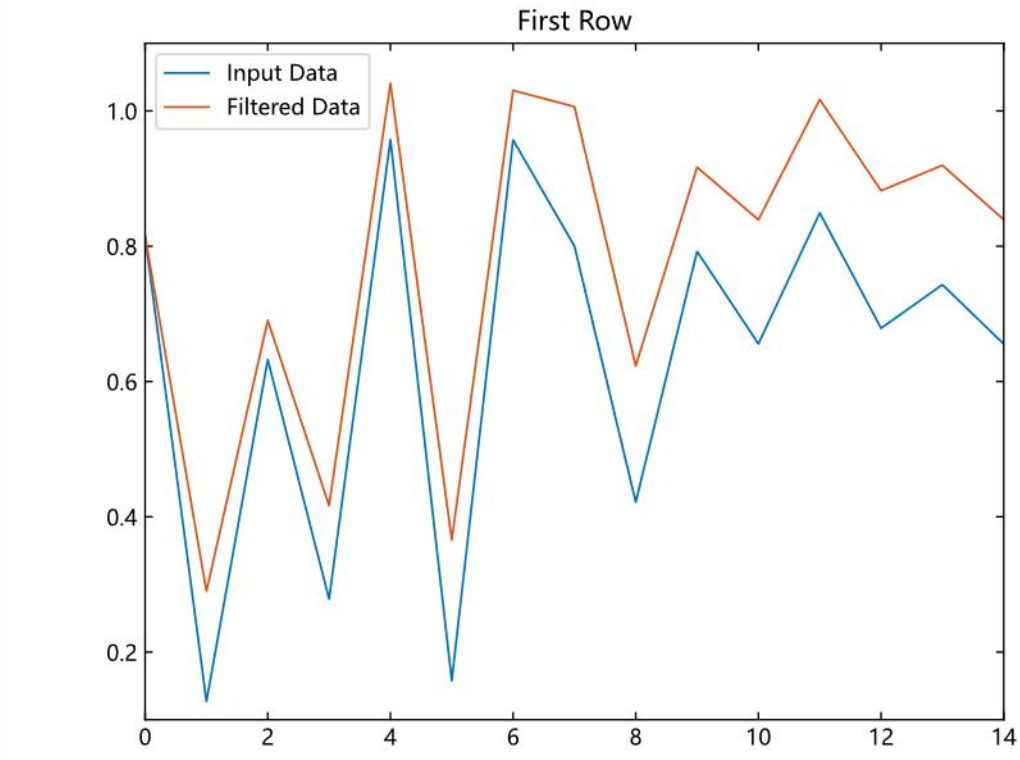
```
pkg_dir = pkgdir(TyMath)
source_path = pkg_dir * "/examples/Fourier Analysis And Filtering/Filter/filter1_tests/filter1_test2_data.jl"
include(source_path)
```

定义有理传递函数的分子和分母系数。

```
b = 1
a = [1 -0.2]
```

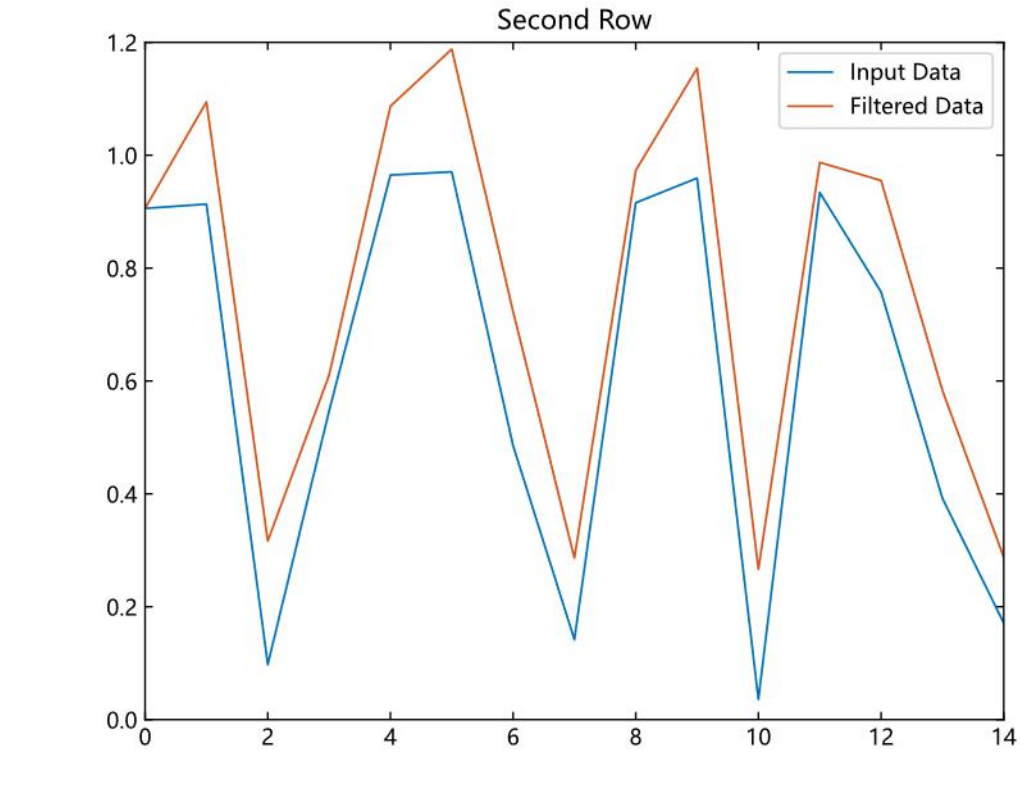
沿着 x 的第二维度应用传递函数并返回每行的一维数字滤波结果。绘制原始数据的第一行对已滤波数据的图。

```
y, ~ = filter1([b], vec(a), x, [], 2)
t = 0:length(x)/2-1
plot(t, x[1, :])
hold("on")
plot(t, y[1, :])
legend(["Input Data", "Filtered Data"])
title("First Row")
```



绘制输入数据的第二行对已滤波数据的图。

```
figure()
plot(t, x[2, :])
hold("on")
plot(t, y[2, :])
legend(["Input Data", "Filtered Data"])
title("Second Row")
```



(3). 对各部分中的数据进行滤波

使用滤波器延迟的初始条件和最终条件对各部分中的数据进行滤波，尤其是需要考虑内存限制时请执行此操作。

生成一个大型的随机数据序列并将其拆分为两段：x1 和 x2。

```
x = randn(10000, 1)

x1 = x[1:5000]
x2 = x[5001:end]
```

整个序列 x 是 x1 和 x2 的垂直串联。

定义有理传递函数的分子和分母系数，

$$H(z) = \frac{b[1]+b[2]z^{-1}}{a[1]+a[2]z^{-1}} = \frac{2+3z^{-1}}{1+0.2z^{-1}}$$

```
b = [2, 3]
a = [1, 0.2]
```

对子序列 x1 和 x2 进行滤波，一次一个。输出对 x1 进行滤波的最终条件，以便在第一段末尾存储滤波器的内部状态。

```
y1, zf = filter1(b, a, x1)
```

将对 x1 进行滤波的最终条件用作对第二段也就是 x2 进行滤波的初始条件。

```
y2, ~ = filter1(b, a, x2, zf)
```

y1 是来自 x1 的滤波后的数据，而 y2 是来自 x2 的滤波后的数据。整个滤波后的序列是 y1 和 y2 的垂直串联。

同时对整个序列进行滤波以供比较。

```
y, ~ = filter1(b, a, x)

isequal(vec(y), [y1; y2])
```

```
ans = true
```

3.2.6.2.2 filter2

二维数字滤波器

A. 语法

```
Y = filter2(H, X)
Y = filter2(H, X, shape)
```

B. 说明

Y = filter2(H,X) 根据矩阵 H 中的系数，对数据矩阵 X 应用有限脉冲响应滤波器。示例

Y = filter2(H,X,shape) 根据 shape 返回滤波数据的子区。例如，Y = filter2(H,X,"valid") 仅返回计算的没有补零边缘的滤波数据。示例。

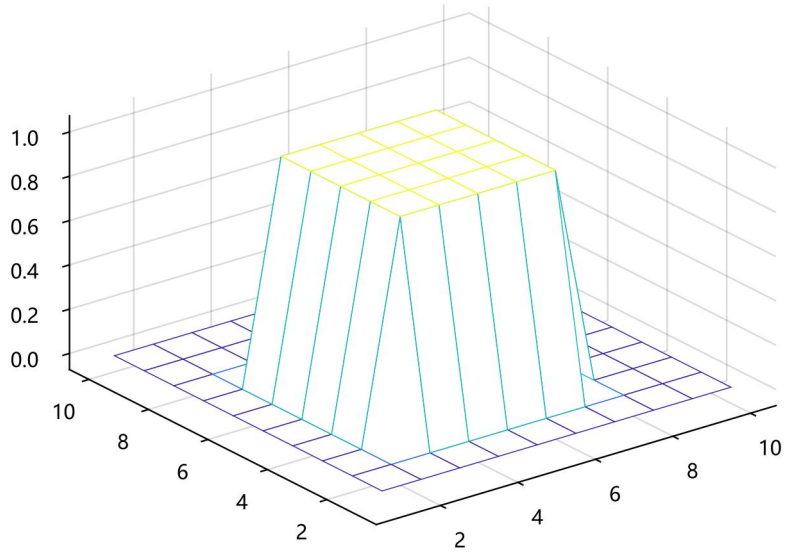
C. 示例

(1). 二维台座

您可以使用与 conv2 函数紧密相关的 filter2 函数对图像和其他二维数据进行数字滤波。

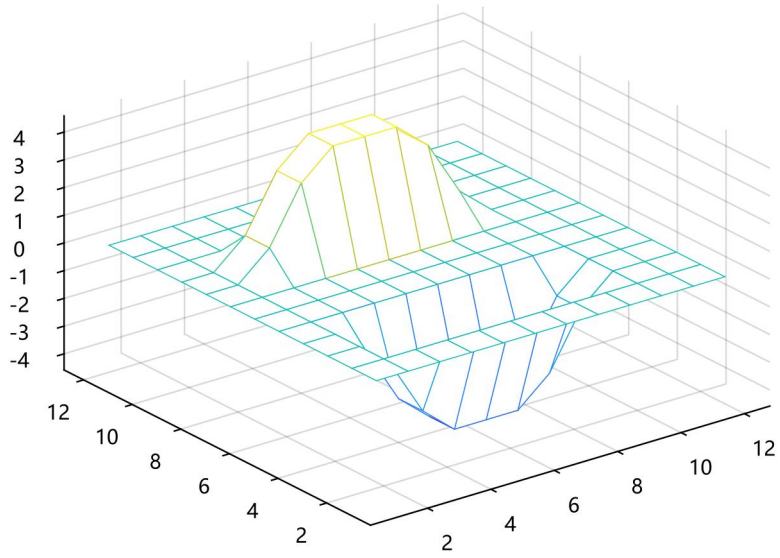
创建并绘制一个内部高度等于 1 的二维台座。

```
A = zeros(10, 10)
A[3:7,3:7] = ones(5,5)
mesh(A)
```



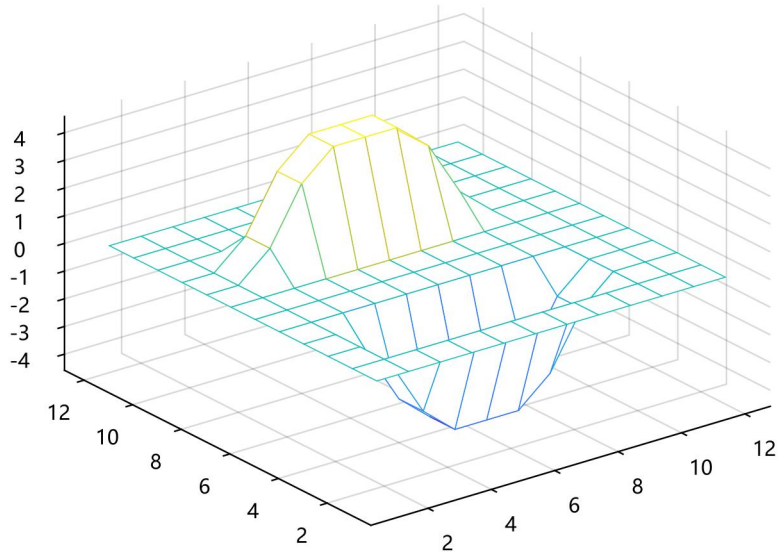
根据滤波器系数矩阵 H 对 A 中的数据进行滤波，并返回已滤波数据的满矩阵。

```
H = [1 2 1;0 0 0;-1 -2 -1]
Y = filter2(H,A,"full")
mesh(Y)
```



将 H 旋转 180 度，并将结果与 A 进行卷积。该输出等同于按照 H 中的系数对 A 中的数据进行滤波。

```
C = conv2(A, rot190(H, 2))
mesh(C)
```



3.2.6.3 卷积

卷积和多项式

函数名	简介
conv	卷积和多项式
conv2	二维卷积
convn	N 维卷积
deconv	去卷积和多项式除法

3.2.6.3.1 conv

卷积和多项式

A. 语法

```
w = conv(u,v)
w = conv(u,v,shape)
```

B. 说明

w = conv(u, v) 返回向量 u 和 v 的卷积。如果 u 和 v 是多项式系数的向量，对其卷积与将这两个多项式相乘等效。

w = conv(u,v,shape) 返回 shape 指定的卷积的分段。例如，conv(u, v,"same") 仅返回与 u 等大小的卷积的中心部分，而 conv(u,v,"valid")仅返回计算的没有补零边缘的卷积部分。

C. 示例

(1). 通过卷积计算多项式乘法

创建包含多项式 x^2+1 和 2x+7 的系数的向量 u 和 v。

```
u = [1,0,1]
v = [2,7]
```

使用卷积将多个多项式相乘。

```
w = conv(u,v)
```

```
w = 4-element Vector{Int64}:
2
7
```

```
2
7
```

w 包含$2x^3 + 7x^2 + 2x + 7$的多项式系数。

(2). 向量卷积

创建两个向量并求其卷积。

```
u = [1,1,1]
v = [1,1,0,0,0,1,1]
w = conv(u,v)
```

```
w = 9-element Vector{Int64}:
1
2
2
1
0
1
2
2
1
```

w 的长度为 length(u)+length(v)-1,在本例中为 9。

(3). 卷积的中心部分

创建两个向量。计算与 u 大小相同的 u 和 v 的卷积的中心部分。

```
u = [-1,2,3,-2,0,1,2]
v = [2,4,-1,1]
w = conv(u,v,"same")
```

```
w = 7-element Vector{Int64}:
15
5
-9
7
6
7
-1
```

w 的长度为 7。全卷积的长度将为 length(u)+length(v)-1，在本例中为 10。

3.2.6.3.2 conv2

二维卷积

A. 语法

```
C = conv2(A,B)
C = conv2(u,v,A)
C = conv2(_,shape)
```

B. 说明

C = conv2(A,B) 返回矩阵 A 和 B 的二维卷积。

C = conv2(u,v,A) 首先求 A 的各列与向量 u 的卷积，然后求每行结果与向量 v 的卷积。

C = conv2(___,shape) 根据 shape 返回卷积的子区。例如，C = conv2(A,B,"same") 返回卷积中大小与 A 相同的中心部分。

C. 示例

(1). 二维卷积

在图像处理等应用程序中，它可用于将卷积的输入直接与输出进行比较。

conv2 函数允许您控制输出的大小。

创建一个 3×3 随机矩阵 A 和一个 4×4 随机矩阵 B。计算 A 和 B 的完整卷积，结果是一个 6×6 矩阵。

```
A = rand(3,3)
B = rand(4,4)
Cfull = conv2(A,B)
```

```
Cfull = 6×6 Matrix{Float64}:
0.0854619   0.0842572   0.342003   0.414123   0.166764   0.0954828
0.169111    0.556699    1.05077    1.2924     1.02881    0.441641
0.510104    1.22835     1.96437    2.56831    1.73169    0.640071
0.607265    1.61147     2.41983    2.29293    1.26286    0.477385
0.521804    1.30936     1.6086     1.23783    0.781328   0.352568
0.100576    0.20879     0.262689   0.178891   0.0873845  0.000685633
```

计算卷积 Csame 的中心部分，它是 Cfull 的子矩阵，大小与 A 相同。Csame 等于 Cfull[3:5,3:5]。

```
Csame = conv2(A,B,"same")
```

```
Csame = 3×3 Matrix{Float64}:
```

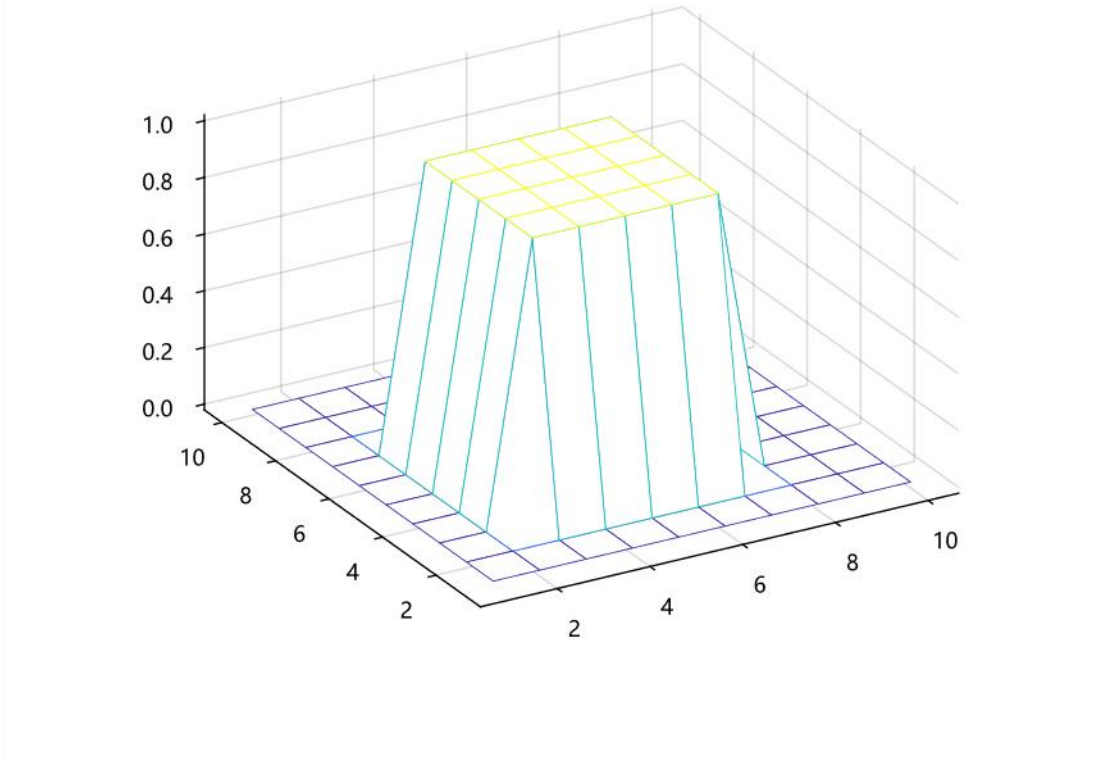
```
1.96437    2.56831    1.73169
2.41983    2.29293    1.26286
1.6086     1.23783    0.781328
```

(2). 提取二维台座边

Sobel 求边运算利用二维卷积来检测图像的边和其他二维数据。

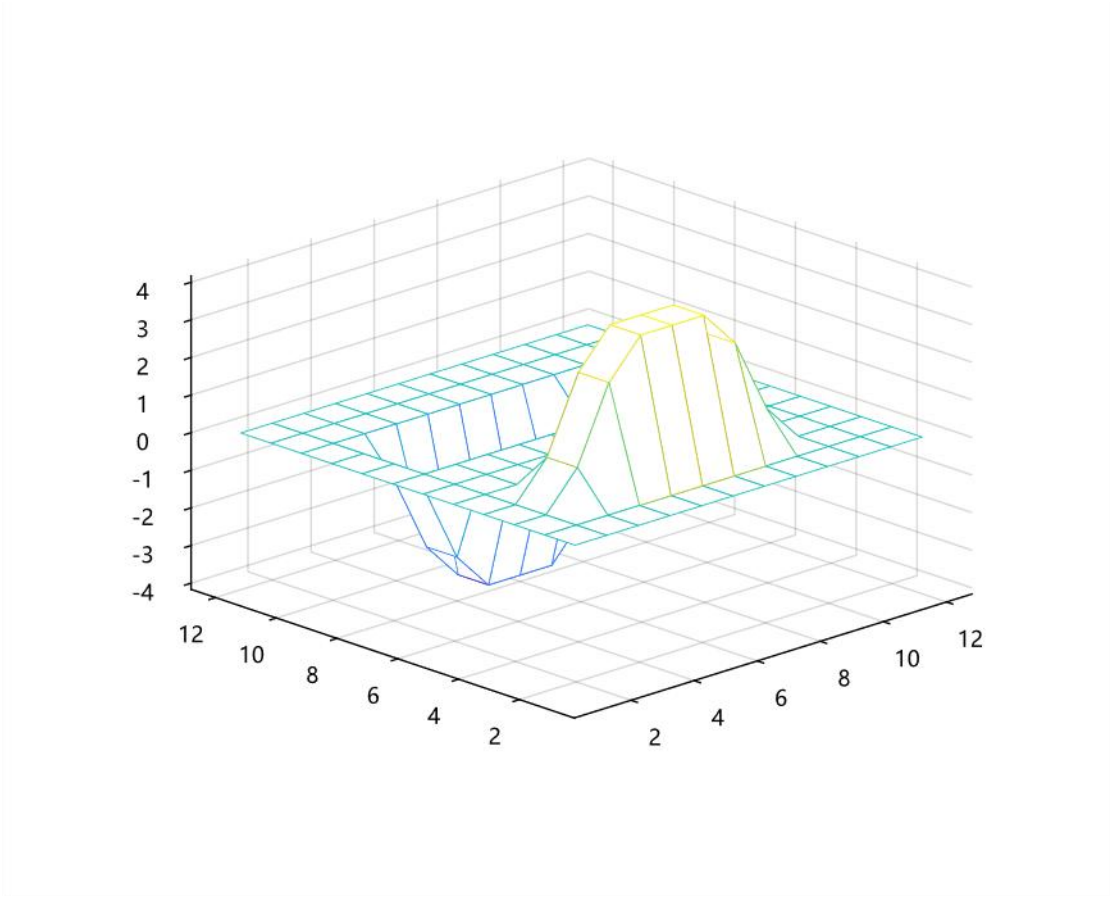
创建并绘制一个内部高度等于 1 的二维台座。

```
A = zeros(10,10)
A[3:7,3:7] .= ones(5,5)
mesh(A)
```



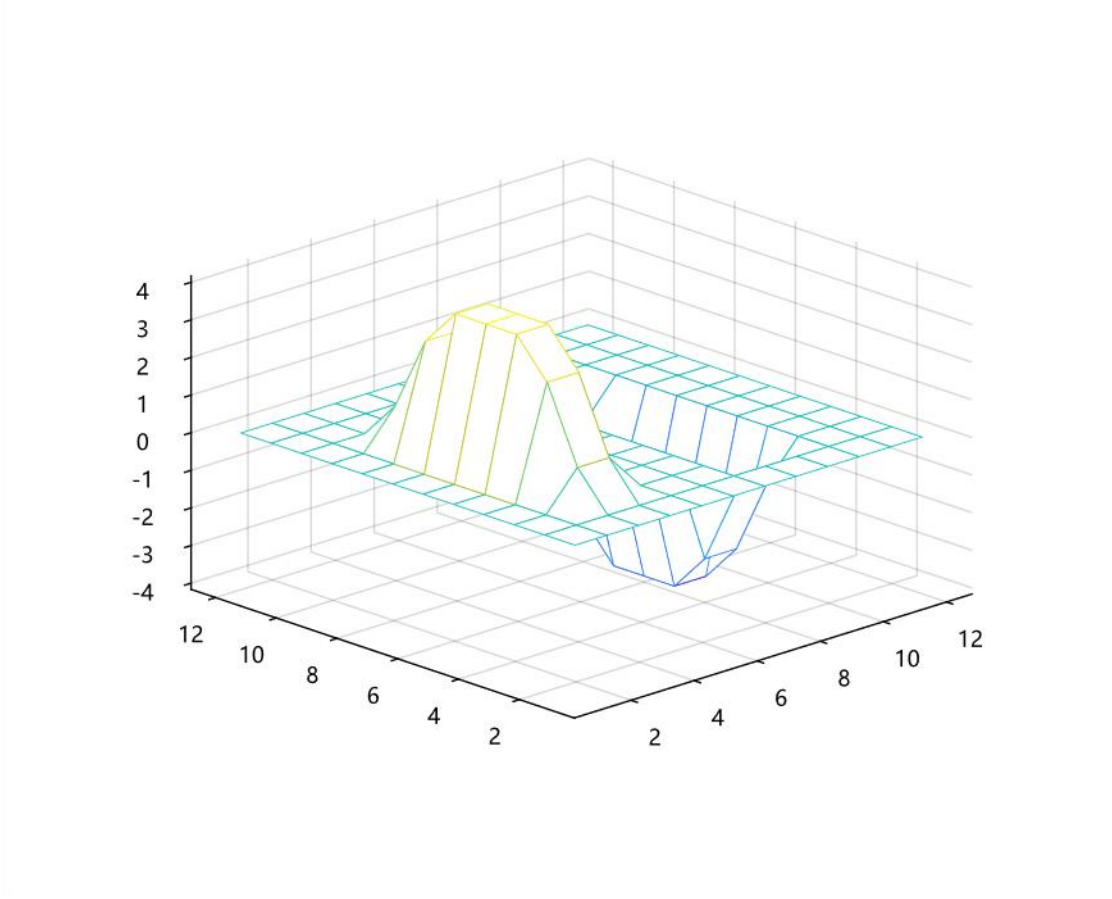
首先求 A 的各行与向量 u 的卷积，然后求卷积结果的各行与向量 v 的卷积。卷积提取台座的水平边。

```
u = [1,0,-1]
v = [1,2,1]
Ch = conv2(u,v,A)
mesh(Ch)
```



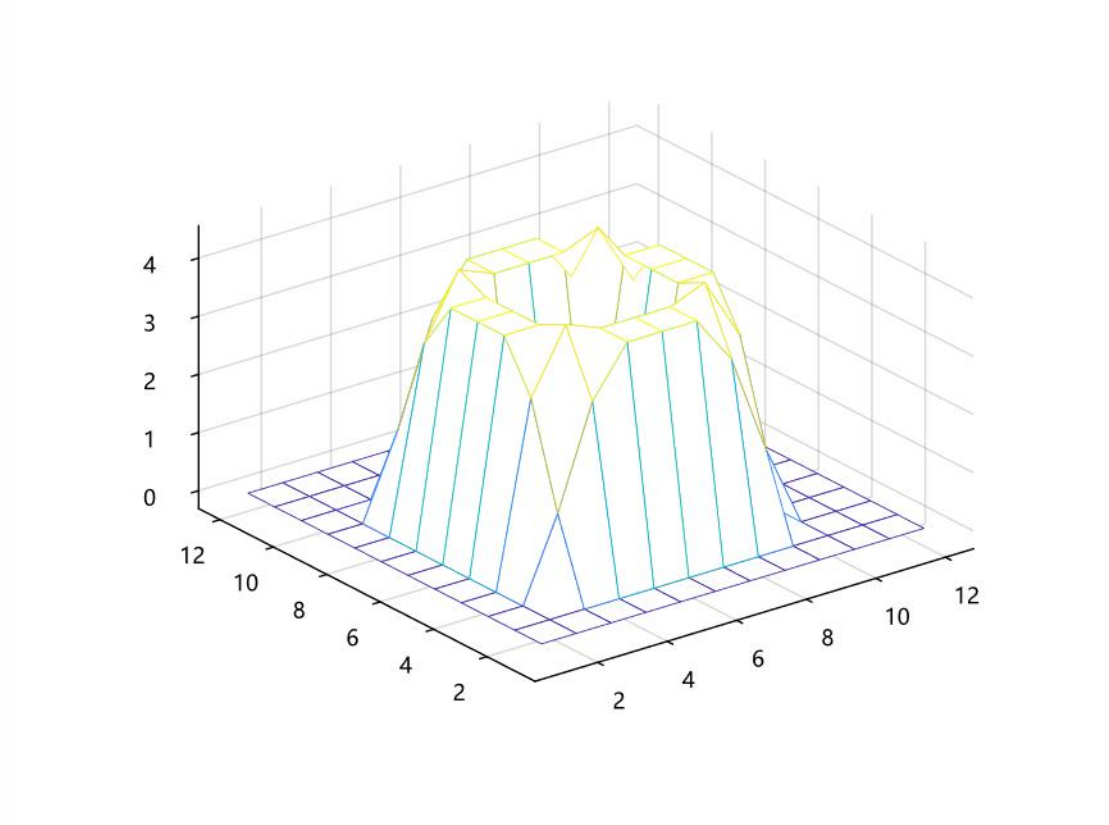
要提取台座的垂直边，请反转与 u 和 v 的卷积顺序。

```
Cv  = conv2(v,u,A)
mesh(Cv)
```



计算并绘制台座的组合边。

```
Figure
mesh(sqrt.(Ch.^2 + Cv.^2))
```



3.2.6.3.3 convn

N 维卷积

A. 语法

```
C = convn(A,B)
```

B. 说明

C = convn(A,B) 返回数组 A 和 B 的 N 维卷积。

C. 示例

(1). 三维卷积

求随机 2×3×2 数组 A 与 2×2×2 核 B 的卷积。结果是一个 3×4×3 数组，大小等于 size(A) .+ size(B) .- 1。

```
A = rand(2,3,2)
B = 0.25*ones(2,2,2)
C = convn(A,B)
```

```
C = 3×4×3 Array{Float64, 3}:
```

```
[:, :, 1] =
 0.112639   0.163295  0.277345  0.226689
 0.16276    0.402652  0.579311  0.339419
 0.0501206  0.239358  0.301966  0.112729
[:, :, 2] =
 0.19428   0.318495  0.454886  0.330671
 0.295464  0.704197  0.949237  0.540505
 0.101184  0.385702  0.494351  0.209834
[:, :, 3] =
 0.0816409  0.1552    0.177541  0.103982
 0.132704   0.301545  0.369926  0.201086
 0.0510635  0.146344  0.192385  0.0971043
```

```
size(A) .+ size(B) .- 1
```

```
ans = (3, 4, 3)
```

3.2.6.3.4 deconv

去卷积和多项式除法

A. 语法

```
q,r = deconv(u,v)
```

B. 说明

q,r = deconv(u,v)使用长除法将向量 v 从向量 u 中去卷积，并返回商 q 和余数 r，以使 u = conv(v,q) + r。如果 u 和 v 是由多项式系数组成的向量，则对它们去卷积相当于将 u 表示的多项式除以 v 表示的多项式。

C. 示例

(1). 多项式除法

创建两个向量 u 和 v，分别包含多项式 2x^3+7x2+4x+9 和 x^2+1 的系数。通过将 v 从 u 中去卷积，将第一个多项式除以第二个多项式，得出与多项式 2x+7 对应的商系数以及与 2x+2 对应的余数系数。

```
u = [2,7,4,9]
v = [1,0,1]
q,r = deconv(u,v)
```

```
q = 2-element Vector{Float64}:
 2.0
```

```
 7.0
r = 4-element Vector{Float64}:
 0.0
 0.0
 2.0
 2.0
```

3.2.7 稀疏数组

初等稀疏矩阵、重新排序算法、迭代法、稀疏线性代数使用稀疏矩阵可极大地减少存储数据所需的内存量。

3.2.7.1 创建

创建稀疏数组

函数名	简介
spalloc	为稀疏矩阵分配空间
spzeros	创建长度为 m 的稀疏向量或大小为 m x n 的稀疏矩阵
speye	稀疏单位矩阵
sprandsym	稀疏对称随机矩阵
spdiagm	创建稀疏带状对角矩阵
sprand	创建一个随机长度 m 稀疏向量或 m × n 稀疏矩阵
sparse	创建稀疏矩阵
sprandn	创建一个长度为 m 的稀疏正态分布随机向量或大小为 m × n 的稀疏正态分布的矩阵
spconvert	从稀疏矩阵外部格式导入

3.2.7.1.1 spalloc

为稀疏矩阵分配空间。

A. 语法

```
S = spalloc(m,n)
```

B. 说明

S = spalloc(m,n) 创建一个大小为 m×n 的全零稀疏矩阵 S,其中 nz >= 1。示例。

C. 示例

(1). 创建具有指定大小和预分配的稀疏矩阵

使用 spalloc 初始化一个 10×10 全零稀疏矩阵

```
S = spalloc(10,10);
```

定义矩阵中的几个元素。

```
S[1:3,1:3] = magic(3)
S
```

```
S = 10×10 SparseMatrixCSC{Float64, Int64} with 9 stored entries:
 8.0  1.0  6.0   ·    ·    ·    ·    ·    ·    ·
 3.0  5.0  7.0   ·    ·    ·    ·    ·    ·    ·
 4.0  9.0  2.0   ·    ·    ·    ·    ·    ·    ·
  ·    ·    ·    ·    ·    ·    ·    ·    ·    ·
  ·    ·    ·    ·    ·    ·    ·    ·    ·    ·
  ·    ·    ·    ·    ·    ·    ·    ·    ·    ·
  ·    ·    ·    ·    ·    ·    ·    ·    ·    ·
  ·    ·    ·    ·    ·    ·    ·    ·    ·    ·
  ·    ·    ·    ·    ·    ·    ·    ·    ·    ·
  ·    ·    ·    ·    ·    ·    ·    ·    ·    ·
```

显示矩阵中非零元素的数量

```
n1 = nnz(S)
```

```
n1 = 9
```

(2). 创建具有非零列的稀疏矩阵

使用 spalloc 初始化一个 20×20 全零稀疏矩阵

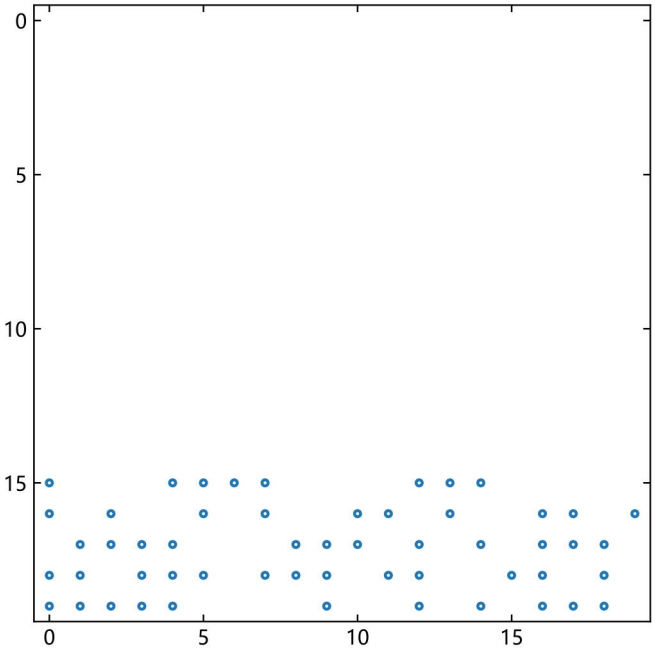
```
n = 20;
S = spalloc(n,n);
```

然后使用 for 循环填充 S 的列，一次填充一列，平均每列最多五个非零元素

```
rng = MT19937ar(5489)
for j = 1:n
    S[:,j] = [zeros(n-5,1); nearest.(rand(rng,5,1))];
end
```

绘制矩阵 S 的稀疏模式。圆点表示非零元素

```
spy(S)
```



显示矩阵中非零元素的数量

```
n1 = nnz(S)
```

```
n1 = 54
```

3.2.7.1.2 spzeros

创建长度为 m 的稀疏向量或大小为 m x n 的稀疏矩阵

这个稀疏数组将不包含任何非零值。 期间不会为非零值分配存储空间建造。如果未指定，则类型默认为 Float64。

A. 语法

```
p = spzeros(a,b=1)
```

```
p = spzeros(eltype,a,b=1)
```

B. 说明

p = spzeros(a,b=1)创建全 0 稀疏数组.

p = spzeros(eltype,a,b=1)创建指定数据类型的稀疏数组

C. 示例

(3). 创建稀疏数组

```
v = spzeros(5)
```

```
v = 5-element SparseVector{Float64, Int64} with 0 stored entries
```

```
length(v)
```

```
ans = 5
```

```
r = spzeros(4,4)
```

```
r = 4×4 SparseMatrixCSC{Float64, Int64} with 0 stored entries:
 .    .    .    .
 .    .    .    .
 .    .    .    .
 .    .    .    .
```

(4). 创建指定元素类型的稀疏数组

```
z = spzeros(Float64,5,5)
```

```
5×5 SparseMatrixCSC{Float64, Int64} with 0 stored entries:
 .    .    .    .    .
 .    .    .    .    .
 .    .    .    .    .
 .    .    .    .    .
 .    .    .    .    .
```

```
eltype(z)
```

```
ans = Float64
```

3.2.7.1.3 speye

稀疏单位矩阵

A. 语法

```
S = speye()
```

```
S = speye(n)
S = speye(n,m)
```

B. 说明

S = speye()返回稀疏标量 1

S = speye(n)返回一个主对角线元素为 1 且其他位置元素为 0 的 n×n 稀疏单位矩阵。

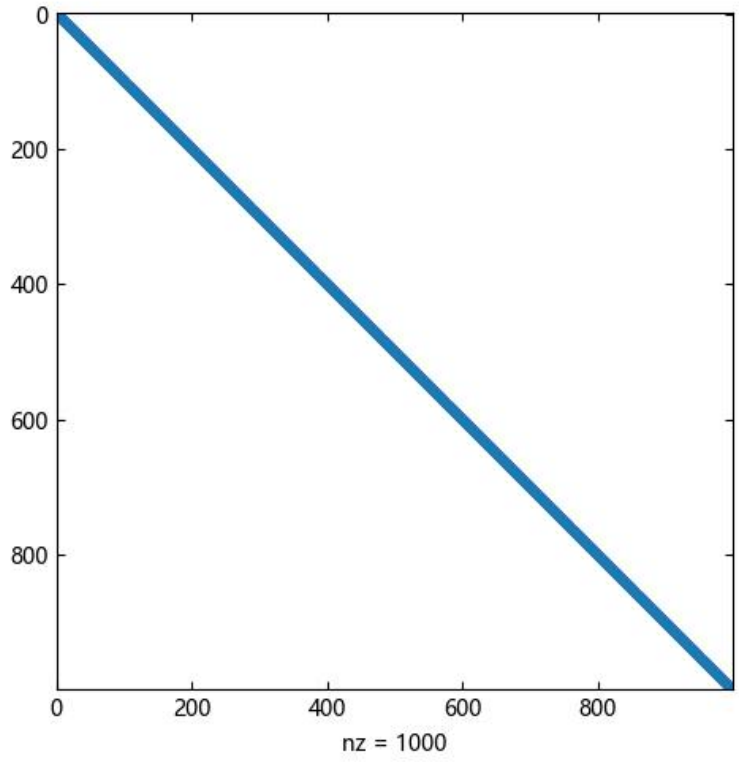
S = speye(n,m) 返回一个主对角线元素为 1 且其他位置元素为 0 的 n×m 稀疏矩阵。

C. 示例示

(1). 稀疏单位方阵

创建 1000×1000 稀疏单位方阵，并查看稀疏模式。

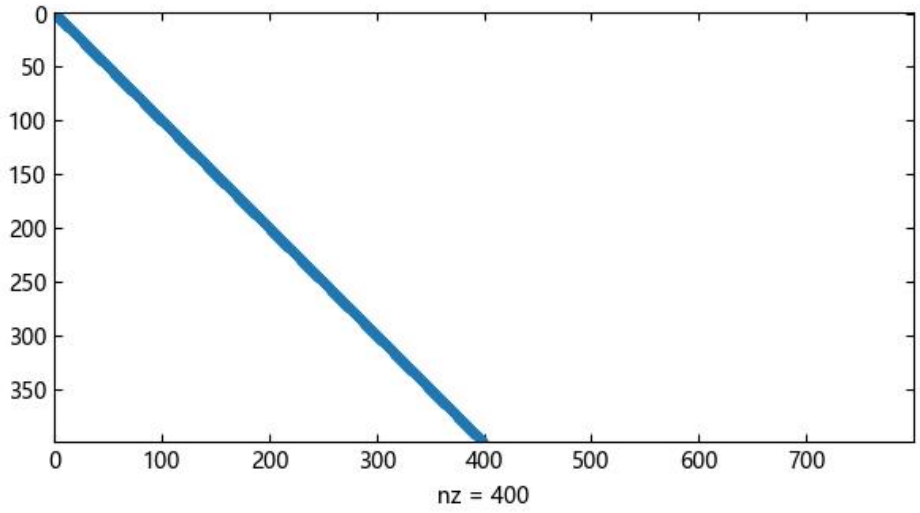
```
I =speye(1000)
TyPlot.spy(I)
nz = nnz(I)
xlabel("nz = $nz")
```



(2). 矩形稀疏单位矩阵

创建 400×800 稀疏单位矩阵，并查看非零模式。

```
S = speye(400,800)
TyPlot.spy(S)
nz = nnz(S)
xlabel("nz = $nz")
```



3.2.7.1.4 sprandsym

阵稀疏对称随机矩阵

A. 语法

```
R = sprandsym(S)
R = sprandsym(n,density)
R = sprandsym(n,density,rc)
R = sprandsym(n,density,rc,kind)
R = sprandsym(S,rc)
R = sprandsym(rng, _)
```

B. 说明

R = sprandsym(S) 返回一个下三角和对角线的结构与 S 相同的对称随机矩阵。其元素是正态分布的，均值为 0，方差为 1。

R = sprandsym(n,density) 返回一个对称随机的、n×n 的稀疏矩阵（具有大约 density*n*n 个非零项）；每一项均是一个或多个正态分布的随机样本之和，并且 (0 <= density <= 1)。

R = sprandsym(n,density,rc) 返回一个条件数倒数等于 rc 的矩阵。各项的分布是不均匀的；它大致围绕 0 对称；所有项均在 [−1,1] 区间内。

如果 rc 是一个长度为 n 的向量，则 R 的特征值为 rc。因而，如果 rc 是一个正（非特征值）向量，则 R 是一个正定（非负）矩阵。在任一情况下，R 是由应用于包含给定特征值或条件数的对角矩阵的随机 Jacobi 旋转生成的。它具有大量拓扑和代数结构。

R = sprandsym(n,density,rc,kind) 是正定矩阵。

如果 kind = true，则 R 由正定对角矩阵的随机 Jacobi 旋转生成。R 恰好具有所需的条件数。

如果 kind = false，则 R 是外积的移位和。R 只具有所需的近似条件数，但具有较少结构体。

R = sprandsym(S,rc) 与矩阵 S 具有相同的结构体并且具有近似条件数 1/rc。

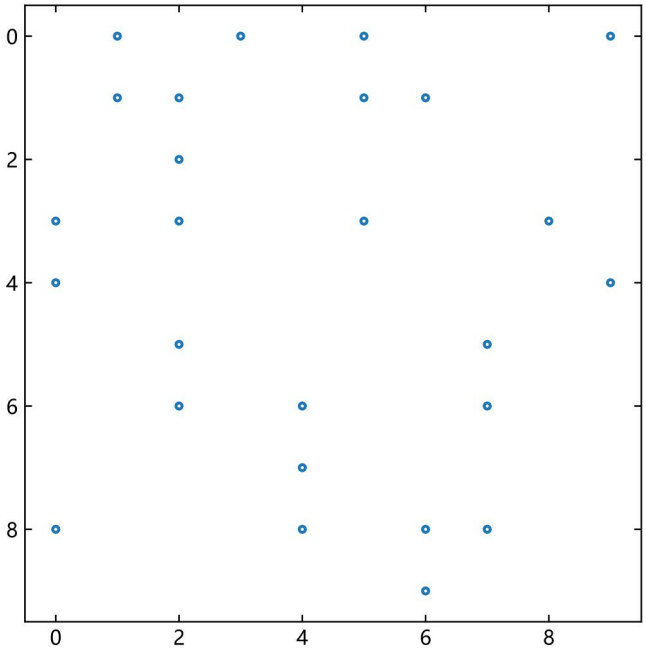
R = sprandsym(rng, _) 使用 rng 控制随机数生成器。示例。

C. 示例

(1). 生成与给定矩阵结构相同的对称随机矩阵

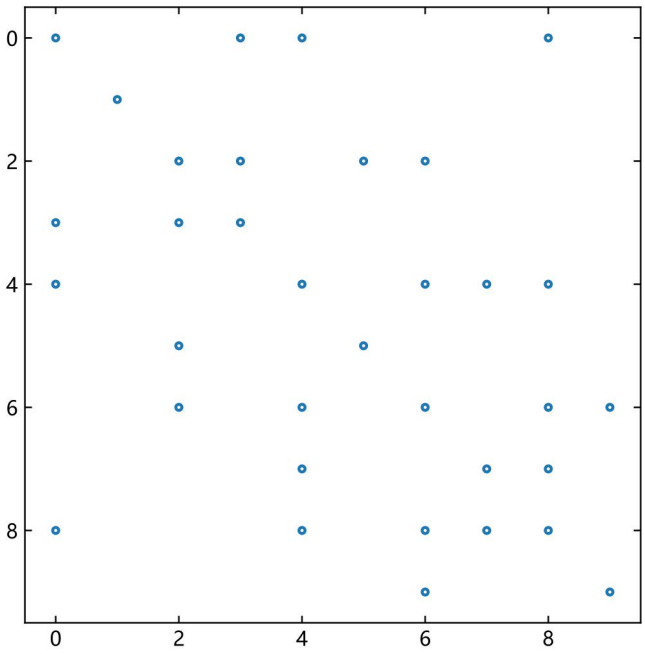
构造一个 10x10 稀疏矩阵 R ，绘制其稀疏模式。

```
rng = MT19937ar(1)
R = sprand(rng, 10, 10, 0.3)
spy(R)
```



构造一个与其结构相同、具近似条件数 3 的对称随机矩阵，并绘制其稀疏模式。

```
S = sprandsym(rng, R, 1/3)
spy(S)
```



查看 S 的逆条件数。

```
rcond(full(S))
```

```
ans = 0.21503576062729038
```

3.2.7.1.5 spdiagm

创建稀疏带状对角矩阵

A. 语法

```
S = spdiagm(A)
S = spdiagm(m,n,A)
```

B. 说明

S = spdiagm(A)构造一个稀疏矩阵 S，其中向量 A 的元素作为对角线元素。默认情况下(没有给定 m 和 n)，矩阵是方形的，它的大小由 length(A)给出，可以通过将 m 和 n 作为第一个参数来指定非正方形大小 m×n。在向量 A 前可以采用(p =>A)的形式设置向量 A 在矩阵对角线的位置。

S = spdiagm(m,n,A) 创建一个 m×n 的稀疏矩阵 S，在向量 A 前可以采用(n=>A)的形式设置向量 A 在矩阵对角线的位置。

C. 示例

(2). 创建稀疏矩阵

默认情况下输入向量 A 元素为矩阵主对角线元素。

```
S = spdiagm([1,2,3])
```

```
3×3 SparseMatrixCSC{Int64, Int64} with 3 stored entries:
1   ·   ·
·   2   ·
·   ·   3
```

通过(p=>A)设置向量元素在矩阵对角线中位置。

```
S = spdiagm(-1=>[1,2,3,4],1=>[4,3,2,1])
```

```
5×5 SparseMatrixCSC{Int64, Int64} with 8 stored entries:
·   4   ·   ·   ·
1   ·   3   ·   ·
·   2   ·   2   ·
·   ·   3   ·   1
·   ·   ·   4   ·
```

(3). 创建指定维度的稀疏矩阵

```
S = spdiagm(4,5,[1,2,3,4])
```

```
4×5 SparseMatrixCSC{Int64, Int64} with 4 stored entries:
1   ·   ·   ·   ·
·   2   ·   ·   ·
·   ·   3   ·   ·
·   ·   ·   4   ·
```

```
S = spdiagm(5,4,-1=>[1,2,3,4])
```

```
5×4 SparseMatrixCSC{Int64, Int64} with 4 stored entries:
·   ·   ·   ·
1   ·   ·   ·
·   2   ·   ·
·   ·   3   ·
·   ·   ·   4
```

3.2.7.1.6 sprand

创建一个随机长度 m 稀疏向量或 m × n 稀疏矩阵

A. 语法

```
S = sprand(m,f)
S = sprand(m,n,f)
S = sprand(type,m,n,f)
```

B. 说明

S = sprand(m,f)创建长度为 m 的稀疏向量

S = sprand(m,n,f)创建维度大小为 m × n 的稀疏矩阵

S = sprand(type,m,n,f)创建指定数据类型的 m × n 的稀疏矩阵

C. 示例

(1). 创建长度为 m 的稀疏向量

```
S = sprand(5,0.3)
```

```
5-element SparseVector{Float64, Int64} with 1 stored entry:

[1]  =  0.49563
```

(2). 创建维度大小为 m × n 的稀疏矩阵

```
S = sprand(5,4,0.5)
```

```
5×4 SparseMatrixCSC{Float64, Int64} with 8 stored entries:
⋅          ⋅         ⋅          ⋅
0.420714   0.366227  0.0708164   ⋅
⋅         0.538278  0.262181     ⋅
⋅          ⋅         ⋅        0.00736784
0.49543    ⋅        0.387745     ⋅
```

(3). 创建指定数据类型的 m × n 的稀疏矩阵

```
z = sprand(Float64,4,3,0.75)
```

```
4×3 SparseMatrixCSC{Float64, Int64} with 6 stored entries:
0.0400319    ⋅        0.730627
⋅        ⋅        0.722801
⋅    0.699256    0.275217
⋅        ⋅        0.64531
```

3.2.7.1.7 sparse

创建稀疏矩阵

A. 语法

```
S = sparse(A)
S = sparse(i,j,v,)
S = sparse(i,j,v,m,n)
S = sparse(i,j,v,m,n,combine)
```

B. 说明

S = sparse(A) 通过挤出任何零元素将满矩阵转换为稀疏格式。

S = sparse(i,j,v) 根据 i、j 和 v 三元组生成稀疏矩阵 S,以便 S(i(k),j(k)) = v(k)。max(i) × max(j) 输出矩阵为 length(v) 个非零值元素分配了空间，其中 length(I) == length(J) == length(V), 参数 I 中所有元素必须满足 1 <= I[k]<=m,J 的所有元素必须满足 1<=J[k]<=n。

S = sparse(i,j,v,m,n) 将 S 的大小指定为 m × n, 当 m, n 未设置时默认取值 m = max(I), n = max(j)。

S = sparse(i,j,v,m,n,combine) 其中 combine 函数用于合并重复项,可以选择基本的运算符号(+、-、*、^)。combine 默认为 +，除非 V 的元素是布尔值，在

这种情况下，默认为 |。

C. 示例

(1). 将满矩阵转换为稀疏格式

```
A = Matrix(1.0I, 5, 5);
S = sparse(A)
```

```
5×5 SparseMatrixCSC{Float64, Int64} with 5 stored entries:
1.0   ·    ·    ·    ·
 ·   1.0   ·    ·    ·
 ·    ·   1.0   ·    ·
 ·    ·    ·   1.0   ·
 ·    ·    ·    ·   1.0
```

(2). 根据 i、j 和 v 三元组生成稀疏矩阵

```
Si = [5, 10];
j = [5, 10];
v = [10, 10];
S = sparse(i,j,v)
```

```
10×10 SparseMatrixCSC{Int64, Int64} with 2 stored entries:
 ·   ·   ·   ·   ·   ·   ·   ·   ·   ·
 ·   ·   ·   ·   ·   ·   ·   ·   ·   ·
 ·   ·   ·   ·   ·   ·   ·   ·   ·   ·
 ·   ·   ·   ·   ·   ·   ·   ·   ·   ·
 ·   ·   ·   ·  10   ·   ·   ·   ·   ·
 ·   ·   ·   ·   ·   ·   ·   ·   ·   ·
 ·   ·   ·   ·   ·   ·   ·   ·   ·   ·
 ·   ·   ·   ·   ·   ·   ·   ·   ·   ·
 ·   ·   ·   ·   ·   ·   ·   ·   ·   ·
 ·   ·   ·   ·   ·   ·   ·   ·   ·  10
```

```
size(S)
```

```
 (10, 10)
```

```
i = [1,3,5];
j = [2,4,6];
v = [1,2,3];
S = sparse(i,j,v)
```

```
5×6 SparseMatrixCSC{Int64, Int64} with 3 stored entries:
 ·   1   ·   ·   ·   ·
 ·   ·   ·   ·   ·   ·
```

```
·   ·   ·   2   ·   ·
·   ·   ·   ·   ·   ·
·   ·   ·   ·   ·   3
```

(3). 创建指定大小的稀疏矩阵

```
i = [5,7,9,10];
j = [4,6,8,10];
v = [10,10,10,10];
S = sparse(i,j,v,10,10)
```

```
10×10 SparseMatrixCSC{Int64, Int64} with 4 stored entries:
·   ·   ·   ·   ·   ·   ·   ·   ·   ·
·   ·   ·   ·   ·   ·   ·   ·   ·   ·
·   ·   ·   ·   ·   ·   ·   ·   ·   ·
·   ·   ·   ·   ·   ·   ·   ·   ·   ·
·   ·   ·  10   ·   ·   ·   ·   ·   ·
·   ·   ·   ·   ·   ·   ·   ·   ·   ·
·   ·   ·   ·   · 10   ·   ·   ·   ·
·   ·   ·   ·   ·   ·   ·   ·   ·   ·
·   ·   ·   ·   ·   ·   · 10   ·   ·
·   ·   ·   ·   ·   ·   ·   ·   · 10
```

```
i = [5,7,9,10];
j = [4,6,8,10];
v = [10,10,10,10];
S = sparse(i,j,v,20,20);
```

(4). 创建指定大小的稀疏矩阵，并进行重复项合并操作

```
sparse([2,2,3],[2,2,3],[2,3,4],3,3,-)
```

```
3×3 SparseMatrixCSC{Int64, Int64} with 2 stored entries:
·   ·   ·
·  -1   ·
·   ·   4
```

```
sparse([2,2,3],[2,2,3],[2,3,4],3,3,^)
```

```
3×3 SparseMatrixCSC{Int64, Int64} with 2 stored entries:
·   ·   ·
·   8   ·
·   ·   4
```

```
sparse([2,2,3],[2,2,3],[true,false,true],3,3,|)
```

```
3×3 SparseMatrixCSC{Bool, Int64} with 2 stored entries:
·   ·   ·
·   1   ·
```

```
 ⋅    ⋅   1
```

3.2.7.1.8 sprandn

创建一个长度为 m 的稀疏正态分布随机向量或大小为 m × n 的稀疏正态分布的矩阵

A. 语法

```
S = sprandn(m,f)
S = sprandn(m,n,f)
S = sprandn(type,m,n,f)
```

B. 说明

S = sprandn(m,f)创建长度为 m 的稀疏向量,矩阵中元素非零的概率由 f 独立给出,满足正态分布。

S = sprandn(m,n,f)创建维度大小为 m × n 的稀疏矩阵,矩阵中元素非零的概率由 f 独立给出,满足正态分布。

S = sprandn(type,m,n,f)创建指定数据类型的 m×n 的稀疏矩阵,矩阵中元素非零的概率由 f 独立给出,满足正态分布。

C. 示例

(1). 创建长度为 m 的稀疏向量

```
S = sprandn(5,0.3)
```

```
5-element SparseVector{Float64, Int64} with 3 stored entries:
[1]  =  0.0299735
[4]  =  1.28501
[5]  =  1.11874
```

(2). 创建维度大小为 m×n 的稀疏矩阵

```
S = sprandn(5,4,0.5)
```

```
0.862833    0.604216      ⋅           ⋅
 ⋅          0.22834   -0.145427       ⋅
-0.759753     ⋅          ⋅           ⋅
 0.29852      ⋅          ⋅           ⋅
-0.0483773  -1.58875      ⋅       -0.5832
```

(3). 创建指定数据类型的 m×n 的稀疏矩阵

```
S = sprandn(Float64,4,3,0.75)
```

```
4×3 SparseMatrixCSC{Float64, Int64} with 10 stored entries:
-1.18009    1.36672    0.513201
-1.12415   -0.612964  -0.333651
-0.843464  -1.49557       ·
·          -0.627063   0.144007
```

```
S = sprandn(Float16,2,3,0.5)
```

```
2×3 SparseMatrixCSC{Float16, Int64} with 4 stored entries:
0.0912  -1.862   0.5728
·       0.6562    ·
```

3.2.7.1.9 spconvert

从稀疏矩阵外部格式导入

A. 语法

```
S = spconvert(D)
```

B. 说明

S = spconvert(D) 根据 D 的列，按与 sparse 函数类似的方式构造稀疏矩阵 S。示例

如果 D 的大小为 N×3，则 spconvert 将使用 D 的列 [i,j,re] 构造 S，以使 S[i[k], j[k]] = re[k]。

如果 D 的大小为 N×4，则 spconvert 将使用 D 的列 [i,j,re,im] 构造 S，以使 S[i[k], j[k]] = re[k] + 1im*im[k]。

C. 示例

(1). 将数据转换为稀疏矩阵

创建一个包含下列值的矩阵 uphill

```
uphill = [ 1.0  1.0  1.0      0.142857
 1.0  2.0  0.5      0.166667
 2.0  2.0  0.333333  0.2
 1.0  3.0  0.333333  0.25
 2.0  3.0  0.25      0.2
```

```
 3.0  3.0  0.2       0.25
 1.0  4.0  0.25      0.333333
 2.0  4.0  0.2       0.333333
 3.0  4.0  0.166667  0.5
 4.0  4.0  0.142857  1.0
 4.0  4.0  0.0       0.0]
```

有时，会特意在文件的最后一行包含预期大小的零值矩阵，这种做法很常见。它可以确保转换后的稀疏矩阵具有该预期大小。

将其转换为稀疏矩阵

```
H = spconvert(uphill)
```

```
H = 4×4 SparseMatrixCSC{ComplexF64, Int64} with 10 stored entries:
 1.0+0.142857im                    0.5+0.166667im      0.333333+0.25im
0.25+0.333333im
     ·           0.333333+0.2im          0.25+0.2im       0.2+0.333333im
     ·                   ·              0.2+0.25im  0.166667+0.5im
     ·                   ·                   ·       0.142857+1.0im
```

本例中，不需要矩阵的最后一行，这是因为前面的行已指定该矩阵至少为4×4

3.2.7.2 操作

操作稀疏数组

函数名	简介
issparse	确定输入是否为稀疏矩阵
nnz	非零矩阵元素的数目
nonzeros	非零矩阵元素
spones	将非零稀疏矩阵元素替换为一
find	查找非零元素的索引和值
full	将稀疏矩阵转换为满存储
permute	将矩阵元素进行置换
dropzeros	生成矩阵或向量的副本，并从该副本中删除存储的数字零
dropzeros!	删除矩阵或向量中存储的数字零

droptol!	删除 A 中绝对值小于或等于 tol 的存储值
nzrange	将索引范围返回到稀疏矩阵列的结构非零值
rowvals	返回 A 的行索引的向量
blockdiag	以块对角方式连接矩阵
findnz	返回一个元组包含 A 中存储的（“结构上非零”）值的索引和对应值的向量
map	将函数应用于非零稀疏矩阵元素
nzmax	为非零矩阵元素分配的存储量
spfun	将函数应用于非零稀疏矩阵元素

3.2.7.2.1 issparse

确定输入是否为稀疏矩阵

A. 语法

```
issparse(S)
```

B. 说明

如果 S 是稀疏的，则 issparse(S) 返回 true，否则返回 false。

C. 示例

(1). 判断 sv 是否稀疏

创建一个稀疏向量 sv 。

```
sv = sparsevec([1, 4], [2.3, 2.2], 10)
```

```
10-element SparseVector{Float64, Int64} with 2 stored entries:
[1 ]  =  2.3
[4 ]  =  2.2
```

判断 sv 是否稀疏，结果为真。

```
issparse(sv)
```

```
true
```

若将 sv 转换为由 0.0 填充空值的数组，再判断是否稀疏，结果为假。

```
issparse(Array(sv))
```

```
false
```

3.2.7.2.2 nnz

非零矩阵元素的数目

A. 语法

```
N = nnz(X)
```

B. 说明

N = nnz(X) 返回矩阵 X 中的非零元素数。

C. 示例

(1). 非零元素数

创建一个单位矩阵并确定它包含的非零元素数。

```
X = eye(4)
```

```
X = 4×4 Matrix{Int64}:
1    0    0    0
0    1    0    0
0    0    1    0
0    0    0    1
```

```
nnz(sparse(X))
```

```
ans = 4
```

(2). 满足条件的元素数

将 nnz 与关系运算符结合使用可确定有多少矩阵元素满足条件。由于关系运算符产生由 1 和 0 组成的逻辑矩阵，因此 nnz 函数会对值为 1（即条件为 true）的元素进行计数。

创建一个矩阵并确定有多少元素大于 10。

```
X = sparse(magic(5))
```

```
X = 5×5 SparseMatrixCSC{Int64, Int64} with 25
stored entries:
17  24   1   8  15
23   5   7  14  16
4   6  13  20  22
10  12  19  21   3
11  18  25   2   9
```

```
nnz(X.>10)
```

```
ans = 15
```

3.2.7.2.3 nonzeros

非零矩阵元素

A. 语法

```
v = nonzeros(A)
```

B. 说明

v = nonzeros(A) 返回 A 中非零元素的满列向量。v 中的元素按列排序。

C. 示例

(1). 非零矩阵元素

使用 nonzeros 返回稀疏矩阵中的非零元素。

创建一个包含几个非零元素的 10×10 稀疏矩阵。稀疏矩阵的典型显示是一个包含非零值及其位置的列表。

```
A = sparse([1 ,3 ,2 ,1],[1 ,1 ,2 ,3],[1:4...],10,10)
```

```
A = 10×10 SparseMatrixCSC{Int64, Int64} with 4 stored entries:
1  ·  4  ·  ·  ·  ·  ·  ·  ·
·  3  ·  ·  ·  ·  ·  ·  ·  ·
2  ·  ·  ·  ·  ·  ·  ·  ·  ·
·  ·  ·  ·  ·  ·  ·  ·  ·  ·
·  ·  ·  ·  ·  ·  ·  ·  ·  ·
·  ·  ·  ·  ·  ·  ·  ·  ·  ·
·  ·  ·  ·  ·  ·  ·  ·  ·  ·
·  ·  ·  ·  ·  ·  ·  ·  ·  ·
·  ·  ·  ·  ·  ·  ·  ·  ·  ·
·  ·  ·  ·  ·  ·  ·  ·  ·  ·
```

找到非零元素的值。

```
v = nonzeros(A)
```

```
v = 4-element Vector{Int64}:
1
2
3
```

4

3.2.7.2.4 spones

将非零稀疏矩阵元素替换为一

A. 语法

```
R = spones(S)
```

B. 说明

R = spones(S) 生成与 S 具有相同稀疏结构的矩阵 R，但 1 位于非零位置。

C. 示例

c = sum(spones(S)) 是每列中的非零数。
r = sum(spones(S'))' 是每行中的非零数。
sum(c) 和 sum(r) 相等，并且都等于 nnz(S)。

3.2.7.2.5 find

3.2.7.2.6 full

将稀疏矩阵转换为满存储

A. 语法

```
A = full(S)
```

B. 说明

A = full(S) 将稀疏矩阵 S 转换为满存储结构，这样 issparse(A)返回 false。

C. 示例

(1). 将稀疏矩阵转换为满存储
更改矩阵的存储格式，并比较存储要求。

创建一个随机稀疏矩阵。在 Syslab 中，稀疏矩阵的显示会忽略所有零，只显示非零元素的位置和值。

```
S = sprand(8,8,0.3)
```

```
S = 8×8 SparseMatrixCSC{Float64, Int64} with 18 stored entries:
·          ·          0.327591       ·          ·          ·          ·
·
0.955838     ·          ·          ·          ·          ·          ·
·
0.902954    0.224593     ·          ·          ·          ·          ·
·
·          ·          ·          .887038     ·          .317868     ·
0.00790958
·          ·          0.768008        .432194     .308172     .960727
·          ·
0.325054     ·          0.979851        0.95914      ·          ·
·          ·
·          ·          ·          ·          ·          .0681023     ·
·
0.0116099    ·          ·          ·          0.866826    0.396224
·          ·
```

将矩阵转换为满存储。

```
A = full(S)
```

```
8×8 Matrix{Float64}:
0.0             0.0             0.327591        0.0             0.0
0.0          0.0         0.0
0.955838        0.0             0.0             0.0             0.0
0.0          0.0         0.0
0.902954        0.224593        0.0             0.0             0.0
0.0          0.0         0.0
0.0             0.0             0.0             0.887038        0.0
0.317868      0.0         0.00790958
0.0           0.0           0.768008        0.432194        0.308172
0.960727      0.0         0.0
0.325054        0.0             0.979851        0.95914         0.0
0.0          0.0         0.0
0.0             0.0             0.0             0.0             0.0
0.0681023     0.0         0.0
0.0116099      0.0           0.0             0.0             0.866826
0.396224      0.0         0.0
```

比较两种格式的存储要求：

```
varinfo()
```

```
* A 552 bytes 8×8 Matrix{Float64}
* S 520 bytes 8×8 SparseMatrixCSC{Float64, Int64}
```

3.2.7.2.7 permute

双边置换稀疏矩阵

A. 语法

```
S = permute(A,c1,c2)
```

B. 说明

S= permute(A,c1,c2)双边置换 A,返回 A[p,q]。列置换 q 的长度必须与 A 的列数匹配（length(q)==size(A,2)）。行排列 p 的长度必须与 A 的行数匹配（length(p)==size(A,1)）。

C. 示例

(1). 将矩阵元素进行置换

```
A = spdiagm(0 => [1, 2, 3, 4], 1 => [5, 6, 7])
4×4 SparseMatrixCSC{Int64, Int64} with 7 stored entries:
1  5  ·  ·
·  2  6  ·
·  ·  3  7
·  ·  ·  4
permute(A, [4, 3, 2, 1], [1, 2, 3, 4])
4×4 SparseMatrixCSC{Int64, Int64} with 7 stored entries:
·  ·  ·  4
·  ·  3  7
·  2  6  ·
1  5  ·  ·
permute(A, [1, 2, 3, 4], [4, 3, 2, 1])
4×4 SparseMatrixCSC{Int64, Int64} with 7 stored entries:
·  ·  5  1
·  6  2  ·
7  3  ·  ·
4  ·  ·  ·
```

3.2.7.2.8 dropzeros

生成矩阵或向量的副本，并从该副本中删除存储的数字零

A. 语法

```
S = dropzeros(A)
```

B. 说明

S = dropzeros(A) 生成 A 的副本，并从该副本中删除存储的数字零。

C. 示例

(1). 生成 A 的副本，并从该副本中删除存储的数字零

创建一个单位矩阵并确定它包含的非零元素数。

```
A = sparse([1, 2, 3], [1, 2, 3], [1.0, 0.0, 1.0])
```

```
3×3 SparseMatrixCSC{Float64, Int64} with 3 stored entries:
1.0     ·       ·
·      0.0      ·
·       ·      1.0
```

```
S = dropzeros(A)
```

```
3×3 SparseMatrixCSC{Float64, Int64} with 2 stored entries:
1.0     ·      ·
·       ·      ·
·       ·     1.0
```

```
A = sparsevec([1, 2, 3], [1.0, 0.0, 1.0])
```

```
3-element SparseVector{Float64, Int64} with 3 stored entries:
[1]  =  1.0
[2]  =  0.0
[3]  =  1.0
```

3.2.7.2.9 dropzeros!

删除矩阵或向量中存储的数字零

A. 语法

```
S = dropzeros!(A)
```

B. 说明

S = dropzeros!(A) 从 A 中删除存储的数字零。

C. 示例

(1). 从 A 中删除存储的数字零

创建一个单位矩阵并确定它包含的非零元素数。

```
A = sparsevec([1, 2, 3], [1.0, 0.0, 1.0])
```

```
3-element SparseVector{Float64, Int64} with 3 stored entries:
[1]  =  1.0
[2]  =  0.0
[3]  =  1.0
```

```
dropzeros!(A)
```

```
A = 3-element SparseVector{Float64, Int64} with 2 stored entries:
[1]  =  1.0
[3]  =  1.0
```

3.2.7.2.10 droptol!

删除 A 中绝对值小于或等于 tol 的存储值

A. 语法

```
S = droptol!(A,tol)
```

B. 说明

S = droptol!(A,tol) 删除 A 中绝对值小于或等于 tol 的存储值。

C. 示例

(1). 删除 A 中绝对值小于或等于 tol 的存储值

```
A = sparsevec([1, 2, 3], [1.0, 0.0, 1.0])
```

```
3-element SparseVector{Float64, Int64} with 3 stored entries:
[1]  =  1.0
[2]  =  0.0
[3]  =  1.0
```

```
S= droptol!(A,0.5)
```

```
3-element SparseVector{Float64, Int64} with 2 stored entries:
[1]  =  1.0
[3]  =  1.0
```

3.2.7.2.11 nzrange

将索引范围返回到稀疏矩阵列的结构非零值

A. 语法

```
S = nzrange(A,col)
```

B. 说明

S = nzrange(A,col) 将索引范围返回到稀疏矩阵列的结构非零值,与 nonzero 结合使用和 rowvals，这允许方便地迭代稀疏矩阵。

```
A = sparse(I,J,V)
rows = rowvals(A)
vals = nonzeros(A)
m, n = size(A)
for j = 1:n    for i in nzrange(A, j)
    row = rows[i]
    val = vals[i]
    end
end
```

C. 示例

(1). 将索引范围返回到稀疏矩阵列的结构非零值

```
A = sparse([1, 2, 3], [1, 2, 3], [1.0, 0.0, 1.0])
```

```
3×3 SparseMatrixCSC{Float64, Int64} with 3 stored entries:
1.0    ·      ·
 ·    0.0     ·
 ·     ·     1.0
```

```
S =  nzrange(A,1)
```

```
1:1
```

3.2.7.2.12 rowvals

返回 A 的行索引的向量

A. 语法

```
S = rowvals(A)
```

B. 说明

S = rowvals(A) 返回 A 中非零元素的行索引的向量。对返回向量的任何修改也会使 A 发生变异。 提供对如何在内部存储行索引的访问与迭代结构性非零值结合使用非常有用。

C. 示例

(1). 返回 A 的行索引的向量

创建一个稀疏矩阵

```
A = sparse(2I, 3, 3)
```

```
3×3 SparseMatrixCSC{Int64, Int64} with 3 stored entries:
2    ·    ·
·    2    ·
·    ·    2
```

返回得到 A 的行索引

```
S = rowvals(A)
```

```
3-element Vector{Int64}:
1
2
3
```

3.2.7.2.13 blockdiag

以块对角方式连接矩阵

A. 语法

```
S = blockdiag(A...)
```

B. 说明

S = blockdiag(A...) 以块对角方式连接矩阵。目前仅针对稀疏矩阵实现。

C. 示例

(2). 以块对角方式连接矩阵

```
S = blockdiag(sparse(2I, 3, 3), sparse(4I, 2, 2))
```

```
=5×5 SparseMatrixCSC{Int64, Int64} with 5 stored entries:
2  ·  ·  ·  ·
·  2  ·  ·  ·
·  ·  2  ·  ·
·  ·  ·  4  ·
·  ·  ·  ·  4
```

3.2.7.2.14 findnz

返回一个元组包含 A 中存储的（“结构上非零”）值的索引和对应值的向量

A. 语法

```
(I, J, V)= findnz(A)
(I, V) = findz(A)
```

B. 说明

(I, J, V)= findnz(A) 返回一个元组 (I, J, V) ，其中 I 和 J 是稀疏矩阵 A 中存储的（“结构上非零”）值的行索引和列索引，而 V 是值的向量。

(I, V) = findz(A)返回一个元组 (I, V) ，其中 I 是稀疏向量 A 中存储的（“结构上非零”）值的索引，而 V 是值的向量。

C. 示例

(1). 返回一个元组(I,J,V)

```
A = sparse([1 2 0; 0 0 3; 0 4 0])
```

```
3×3 SparseMatrixCSC{Int64, Int64} with 4 stored entries:
1  2  ·
·  ·  3
·  4  ·
```

```
S = findnz(A)
```

```
([1, 1, 3, 2], [1, 2, 2, 3], [1, 2, 4, 3])
```

(2). 返回一个元组(I,V)

```
A = sparsevec([1 2 0; 0 0 3; 0 4 0])
```

```
9-element SparseVector{Int64, Int64} with 4 stored entries:
[1]  =  1
[4]  =  2
[6]  =  4
[8]  =  3
```

```
S = findnz(A)
```

```
([1, 4, 6, 8], [1, 2, 4, 3])
```

3.2.7.2.15 map

将函数应用于非零稀疏矩阵元素

A. 语法

```
S = map(f, c...)
```

B. 说明

S = map(f,c...) 将函数有选择地将函数仅应用于稀疏矩阵 S 的非零元素。通过将 f 应用于每个元素来转换参数 c。 对于多个参数,按元素应用 f。

C. 示例

(1). 将函数应用于非零矩阵元素

创建一个单位矩阵并确定它包含的非零元素数。

```
S = map(x -> x * 2, [1, 2, 3])
```

```
3-element Vector{Int64}:
2
4
6
```

```
magic(4)
S = map(x ->x*2,ans)
```

```
4×4 Matrix{Int64}:
32   4   6   26
10  22  20   16
18  14  12   24
8   28  30   2
```

```
map(+, [1, 2, 3], [10, 20, 30])
```

```
3-element Vector{Int64}:
```

```
11
22
33
```

3.2.7.2.16 nzmax

为非零矩阵元素分配的存储量

A. 语法

```
n = nzmax(S)
```

B. 说明

n = nzmax(S) 返回为稀疏矩阵 S 中的非零元素分配的存储空间量。对于稀疏矩阵，nzmax(S) >= 1。

如果 S 为满矩阵，则 nzmax(S) 等于 prod(size(S))。

通常，nnz(S) 和 nzmax(S) 相同。但如果 S 由可生成填充矩阵元素的运算所创建，例如稀疏矩阵乘法或稀疏 LU 分解，则分配的存储空间可能大于实际需要的存储空间，并且 nzmax(S) 会反映这一情况。示例。

C. 示例

(1). 为矩阵元素分配存储量

创建一个 10x10 的随机矩阵，且将绝对值小于 0.5 的元素设为 0 ，赋值给 A ，然后计算其存储量。

```
rng = MersenneTwister(1234)
A = map(x->abs(x)<0.5 ? 0 : x, rand(rng,10,10))
nzmax_full = nzmax(A)
nnz_full = length(find(A.!=0))
```

```
nzmax_full = 100
nnz_full = 43
```

因为矩阵 A 为满矩阵， nzmax(A) 返回 prod(size(A)) 。

将 A 转为稀疏矩阵，然后计算其存储量

```
A = sparse(A)
nzmax(A)
```

```
ans = 43
```

可以看到，此时的 nzmax 与 A 的非零元数目相同。发现 A 的第一个元素

非零，将其设为零然后计算其存储量

```
A[1,1] = 0
nzmax_full = nzmax(A)
nnz_ful = 100 - length(find(A.==0))
```

```
nzmax_full = 43
nnz_full = 42
```

可以看到，此时的 nzmax 不等于矩阵的非零元数量。这是因为 Syslab 并不动态地为稀疏矩阵析出零元，因此一些情况下即使将稀疏矩阵中数据置零，为其分配的存储量也没有改变。

3.2.7.2.17 spfun

将函数应用于非零稀疏矩阵元素

A. 语法

```
F = spfun(func, S)
```

B. 说明

F = spfun(func,S) 将函数 func 应用于稀疏矩阵 S 的非零元素。输入参数 func 是一个函数的函数句柄，此函数接受一个输入参数。示例

此操作保留原始矩阵 S 的稀疏性，除非函数 func 对 S 的某些非零元素返回零。

C. 示例

(1). 将指数函数应用于稀疏矩阵

创建一个 4x4 稀疏对角矩阵。

```
S = sparse(diagm(1:4))
```

```
S = 4×4 SparseMatrixCSC{Int64, Int64} with 4 stored entries:

 1  ·  ·  ·
 ·  2  ·  ·
 ·  ·  3  ·
 ·  ·  ·  4
```

将指数函数应用于 S 的非零元素。生成的矩阵具有与 S 相同的稀疏模式。

```
F = spfun(exp, S)
```

```
F = 4×4 SparseMatrixCSC{Float64, Int64} with 4 stored entries:

 2.71828   ·         ·         ·
  ·       7.38906    ·         ·
  ·        ·       20.0855    ·
  ·        ·         ·       54.5982
```

由于 spfun 仅应用于 S 的非零元素，因此只要 S[i] 为零，F[i] 的值就为零。这不同于将函数应用于 S 的所有元素。例如，将结果与将指数函数应用于 S 的所有元素进行比较。exp(S) 函数对 S 的 0 元素返回 1。

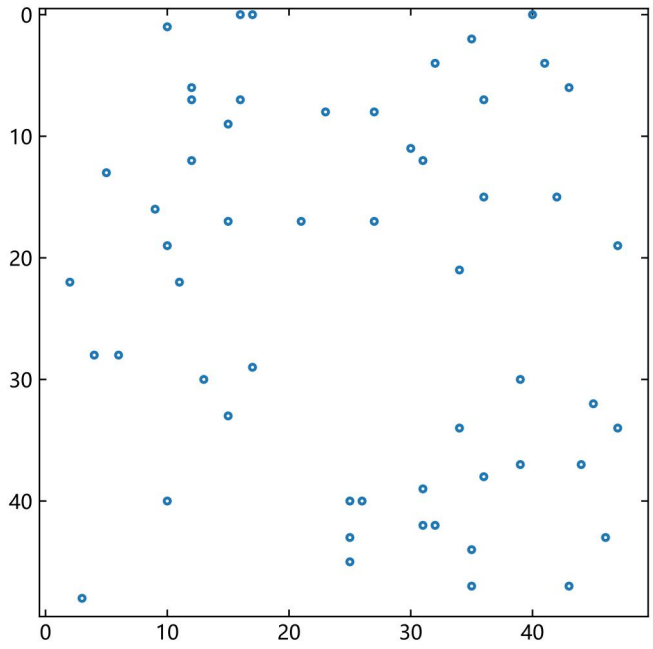
```
full(exp.(S))
```

```
ans = 4×4 Matrix{Float64}:
 2.71828  1.0      1.0      1.0
 1.0      7.38906  1.0      1.0
 1.0      1.0     20.0855   1.0
 1.0      1.0      1.0     54.5982
```

(2). 应用二次函数并绘制稀疏模式

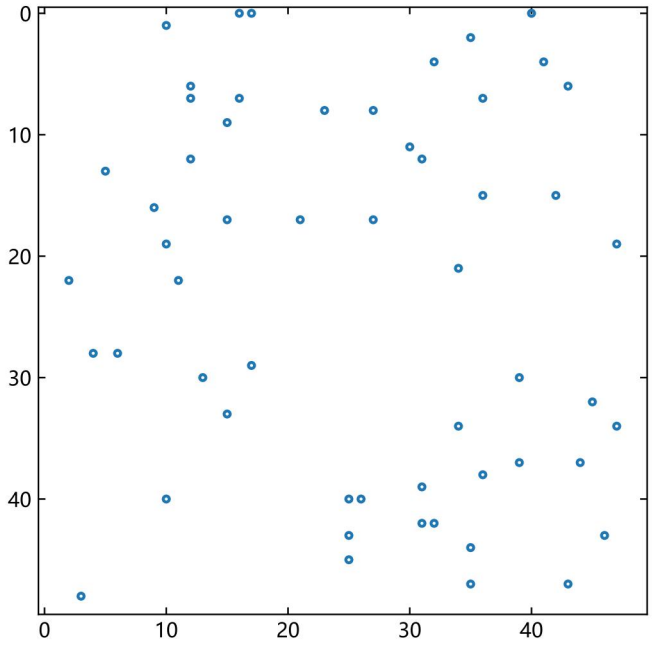
创建一个密度为 0.02 的随机 50×50 稀疏矩阵，其中矩阵具有 50 个非零元素。绘制矩阵 S 的稀疏模式。

```
rng = MersenneTwister(1234)
S = sprand(rng, 50, 50, 0.02)
spy(S)
```



基于 S 的非零元素计算二次函数 x^2+x+1。使用 spfun 计算的函数与矩阵 S 具有相同的稀疏模式。

```
fun(x) = @. x^2 + x + 1
F = spfun(fun, S)
spy(F)
```



3.2.7.3 重排序算法

稀疏数组重排序算法

函数名	简介
amd	近似最小度置换
colamd	列近似最小度排列
colperm	基于非零项计数的稀疏列置换
symamd	对称近似最小度置换
symrcm	稀疏反向 Cuthill-McKee 排序

3.2.7.3.1 amd

近似最小度置换

A. 语法

```
P = amd(A)
P = amd(A,opts)
```

B. 说明

P = amd(A) 为稀疏矩阵 C = A + A' 返回近似最小度置换向量。C[P,P] 或 A[P,P] 的 Cholesky 分解可能比 C 或 A 的 Cholesky 分解稀疏。amd 函数可能比 symamd 函数快，还可能比 symamd 返回更好的排序。矩阵 A 必须是方阵。如果 A 为满矩阵，则 amd(A) 等效于 amd(sparse(A))。

P = amd(A,opts) 允许使用更多的重新排序选项。opts 输入是具有如下所示两个字段的结构体。只需设置相关字段：

AMD_DENSE - 指示视为稠密项的非负标量值。如果 A 为 n×n，则 A + A' 中具有多于 max(16,(dense*sqrt(n))) 个条目的行和列被视为“稠密”项并在排序期间忽略。** Syslab 软件将这些行和列放在输出的置换形式的最后。如果未显示此选项，则此字段的默认值为 10.0。

AMD_AGGRESSIVE - 控制主动吸收的标量值。如果此字段设置为非零值，则执行主动吸收。如果未显示此选项，这就是默认值。

Syslab 软件执行装配树后排序，此运算通常与消去树后排序相同。由于使用了近似度更新，且“稠密”行和列不参与后排序，因此二者有时存在差异。此方法非常适合于后续的 chol 运算。

C. 示例

(1). 预先排序对 Cholesky 因子稀疏性的影响

在使用 amd 对矩阵进行排序之前和之后计算矩阵的 Cholesky 因子，以检查排序操作对稀疏性的影响。

加载杠铃图稀疏矩阵并添加稀疏单位矩阵，以确保它是正定矩阵。计算两个 Cholesky 因子：一个属于原始矩阵，另一个属于使用 amd 预先排序后的原始矩阵。

导入 data.jl 文件。

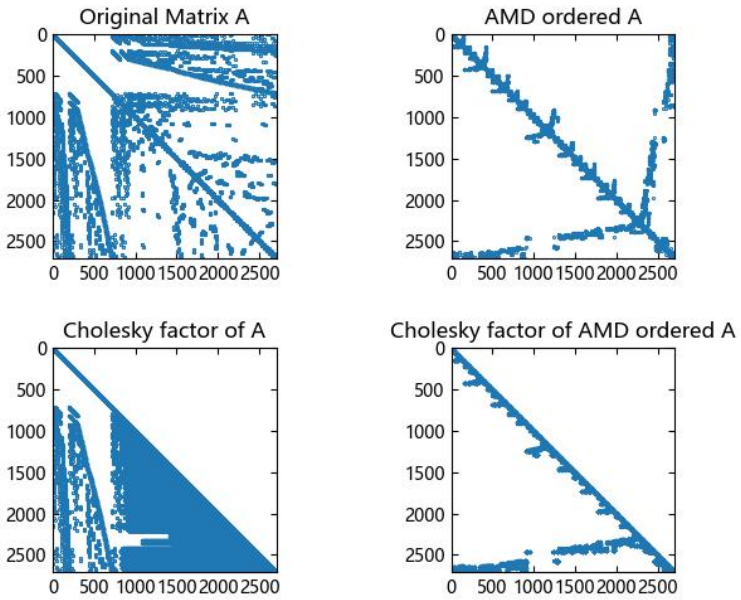
```
pkg_dir = pkgdir(TyMath)
```

```
source_path = pkg_dir * "/examples/Sparse Matrices/Manipulation/amd_tests/data.jl"
include(source_path)

A = A + speye(size(A))
p = amd(A)
L = cholesky(full(A)).L
Lp = cholesky(full(A[p, p])).L
```

绘制所有四个矩阵的稀疏模式。与自然排序的矩阵相比，从经过预先排序的矩阵获取的 Cholesky 因子要稀疏得多。

```
figure
subplot(2,2,1)
spy(A,markersize = 1.5)
title("Original Matrix A")
subplot(2,2,2)
spy(A[p,p],markersize = 1.5)
title("AMD ordered A")
subplot(2,2,3)
spy(L,markersize = 1.5)
title("Cholesky factor of A")
subplot(2,2,4)
spy(Lp,markersize = 1.5)
title("Cholesky factor of AMD ordered A")
```



3.2.7.3.2 colamd

列近似最小排列

A. 语法

```
p = colamd(S)
```

B. 说明

p = colamd(S) 为稀疏矩阵 S 返回列近似最小度置换向量。对于非对称矩阵 S，S[:,p] 可能比 S 具有更稀疏的 LU 因子。S[:,p]' * S[:,p] 的 Cholesky 分解也可能比 S'*S 的 Cholesky 分解稀疏。

knobs 是一个二元素向量。如果 S 是 m×n，则具有超过 (knobs[1])*n 项的行将被忽略。具有超过 (knobs[2])*m 项的列在排序前将被删除，并最后在输出置换 p 中排序。如果 knobs 参数不存在，则 knobs[1] = knobs[2] = spparms('wh_frac')

C. 示例

(1). 比较稀疏矩阵和 LU 分解

由稀疏矩阵组成的 Harwell-Boeing 集合和 demos 目录包含测试矩阵 west0479。它是一个 479 阶的矩阵，产生于八个阶段的化工蒸馏列的 Westerberg 模型。spy 图显示了八个阶段的证据。colamd 排序使此结构体变得混乱。

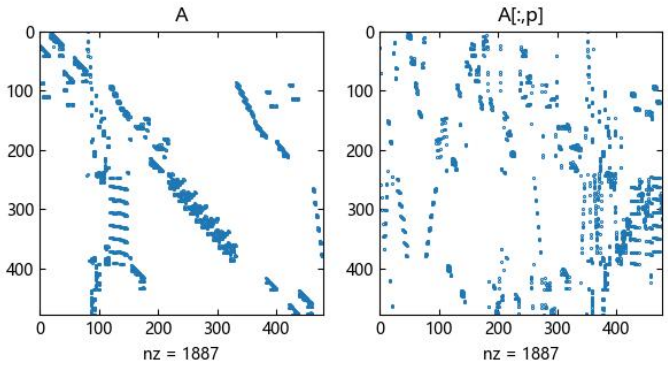
导入 west0479.jl 文件。

```
pkg_dir = pkgdir(TyMath)
source_path        =        pkg_dir        *        "/examples/Sparse
Matrices/Manipulation/colamd_tests/west0479.jl"
include(source_path)

A = west0479;
a = Colamd{Clong}()
p = colamd(A)
A[:, p]

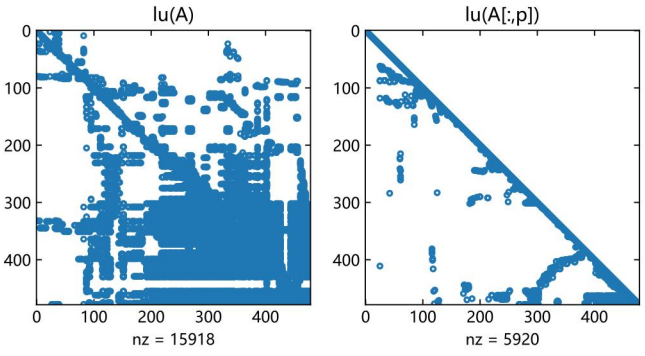
figure()
subplot(1, 2, 1), spy(A,markersize = 1.0), title("A"), xlabel("nz =
1887")
```

```
subplot(1, 2, 2), spy(A[:, p],markersize = 1.0), title("A[:,p]"),
xlabel("nz = 1887")
```



将原始矩阵的 LU 分解的 spy 图与重新排序后的矩阵的 spy 图进行比较，结果表明最小度比因子为 2.8 时降低了时间和存储要求。非零计数分别为 15918 和 5920

```
figure()
subplot(1, 2, 1), spy(lu(Matrix(A)).L + lu(Matrix(A)).U),
title("lu(A)"), xlabel("nz = 15918")
subplot(1, 2, 2), spy(lu(A[:, p]).L), title("lu(A(:,p))"),
xlabel("nz = 5920")
```



3.2.7.3.3 colperm

基于非零项计数的稀疏列置换

A. 语法

```
j = colperm(S)
```

B. 说明

j = colperm(S) 生成置换向量 j，根据非零项数升序对 S[:,j] 的列进行排序。这有时可用于对 LU 分解进行预先排序；在这种情况下，请使用 lu(S[:,j])。

如果 S 是对称矩阵，则 j = colperm(S) 生成置换矩阵 j，根据非零项数升序对 S[j,j] 的行和列两者都进行排序。如果 S 是正定矩阵，则这有时可用于对 Cholesky 分解进行预先排序；在这种情况下，请使用 chol(S[j,j])。

C. 示例

n×n 箭头矩阵

A = [ones(1, n); ones(n - 1, 1) speye(n - 1, n - 1)] 包含全满的第一行和第一列。其 LU 分解 lu(A) 也几乎是全满的。语句

j = colperm(A)

返回 j = [2:n 1]。因此 A[j,j] 将该全满的行和列移到底部和后部，且 lu(A[j,j]) 具有与 A 自身相同的非零结构体。

另一方面，布基球示例

B = bucky()

在每行和每列中恰好有三个非零元素，这样 j = colperm(B) 是单位置换，对于降低后续分解的填充元没有任何作用。

3.2.7.3.4 symamd

对称近似最小度置换

A. 语法

```
p = symamd(S)
```

```
p = symamd(S,knobs)
```

B. 说明

p = symamd(S)（对于对称正定矩阵 S）返回置换向量 p，这样 S[p,p] 倾向于比 S 具有更稀疏的 Cholesky 因子。要计算 S 的排序，symamd 构造一个矩阵 M，这样 spones(M'*M) = spones (S)，然后计算 p = colamd(M)。symamd 函数也可能适用于对称非正定矩阵。

S 必须是方阵；仅引用严格下三角部分。

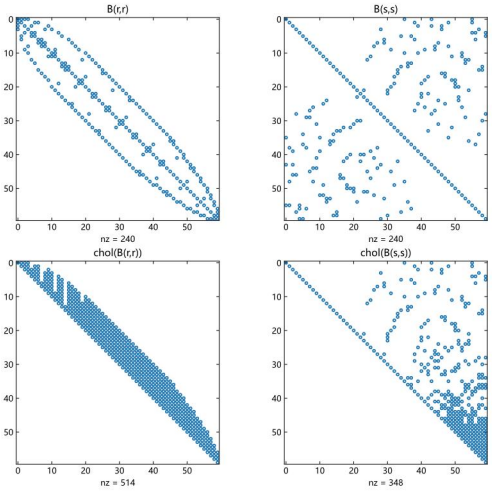
p = symamd(S,knobs)，其中 knobs 为标量。如果 S 为 n×n，则排序前，将会删除包含多个 knobs*n 项的行和列，并最后在输出置换 p 中排序。如果 knobs 参数不存在，则 knobs = spparms('wh_frac')。

C. 示例

(1). 比较反向 Cuthill-McKee 和最小度

此示例是关于反向 Cuthill-McKee 和最小度与 symrcm 参考页中提及的布基球示例的比较。

```
B = bucky() .+ 4 * speye(60);
r = symrcm(B)
p = symamd(B)
R = B[r, r]
S = B[p, p]
subplot(2, 2, 1), spy(R), title("B(r,r)"), xlabel("nz = 240")
subplot(2, 2, 2), spy(S), title("B(s,s)"), xlabel("nz = 240")
subplot(2, 2, 3), spy(cholesky(full(R)).U), title("chol(B(r,r))"),
xlabel("nz = 514")
subplot(2, 2, 4), spy(cholesky(full(S)).U), title("chol(B(s,s))"),
xlabel("nz = 348")
```



尽管这是一个很小的问题，但两种排序的行为都很典型。RCM 生成一个具有窄带宽的矩阵，它在 Cholesky 分解期间几乎完全填满。最小度算法生成一个包含大块连续零的结构体，这些块在分解期间不进行填充。因此，最小度排序需要的分解时间和存储空间都较少

3.2.7.3.5 symrcm

稀疏反向 Cuthill-McKee 排序

A. 语法

```
r = symrcm(S)
```

B. 说明

r = symrcm(S) 返回 S 的对称反向 Cuthill-McKee 排序。此置换向量 r 使得 S(r,r) 可能会使其非零元素更接近于对角线。对于源自长、瘦问题的矩阵的 LU 或 Cholesky 分解来说，这是一种很好的预排序方法。该排序方法对于对称和不对称 S 都有效。

对于实对称稀疏矩阵 S，S(r,r) 的特征值与 S 的特征值相同，但 eig(S(r,r)) 可能比 eig(S) 花费的计算时间更短。

C. 示例

(1). 反向 Cuthill-McKee 排序

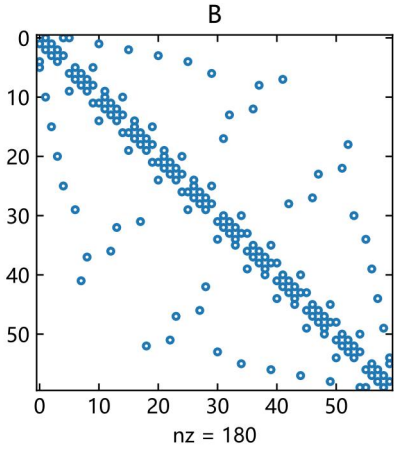
语句。

```
B = bucky()
```

使用 demos 工具箱中的函数生成截顶二十面体的邻接图。这更适合称为足球、Buckminster Fuller 多面穹顶（由此命名为 bucky），或者近来称为由 60 个原子组成的碳分子。顶点有 60 个。这些顶点已通过以下方式进行排序：逐五角形对一个半球中的半数顶点进行编号，然后映射到另一个半球并将两个半球粘合在一起。

通过这种编号，矩阵的带宽就不会特别窄，如第一个 spy 图所示：

```
figure()
subplot(1, 2, 1)
spy(B)
title("B")
xlabel("nz = 180")
```

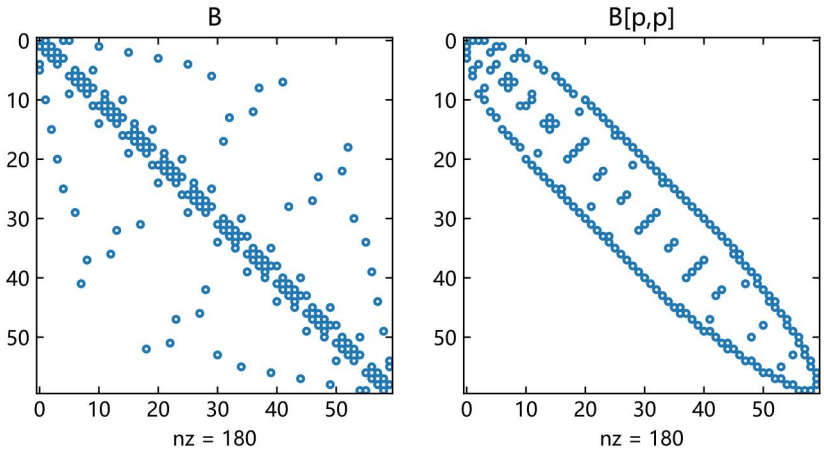


反向 Cuthill-McKee 排序可通过以下命令获取：

```
p = symrcm(B)
R = B[p, p]
```

spy 图显示了更窄的带宽。

```
subplot(1, 2, 2)
spy(R)
title("B[p,p]")
xlabel("nz = 180")
```



symamd 的参考页中会继续对此示例进行说明。

带宽也可以通过以下命令进行计算：

```
i, j = find(B,nargout = 2)
bw = maximum(i - j) + 1
```

B 和 R 的带宽分别为 35 和 12。

3.2.7.4 结构分析

稀疏数组结构分析

函数名	简介
sprank	结构秩
etree	消去树
spaugment	构造最小二乘增广方程组
DMDecomposition	Dulmage-Mendelsohn 分解模块
treelayout	设置树或森林的布局
treeplot	绘制树形图
etreeplot	绘制消去树

gplot	绘制邻接矩阵中的节点和边
unmesh	unmesh

3.2.7.4.1 sprank

结构秩

A. 语法

```
r = sprank(A)
```

B. 说明

r = sprank(A) 计算稀疏矩阵 A 的结构秩。

C. 示例

(1). 计算矩阵的结构秩

计算 2×4 矩阵的结构秩。

```
A = [1 0 2 0;2 0 4 0]
A = sparse(A)
rs = sprank(A)
```

```
rs = 2
```

将结构秩与常规秩计算进行比较。

```
rf = rank(full(A))
```

```
rf = 1
```

对于此矩阵，结构秩是 2，因为两列非零。但是，矩阵的实际秩是 1，因为列互为倍数。

3.2.7.4.2 etree

消去树

A. 语法

```
p,q = etree(A)
p,q = etree(A,method)
```

B. 说明

p,q = etree(A) 为其上三角是 A 的上三角的对称方阵返回消去树 p 以及树的后序置换 q。p[j] 是列 j 在树中的父级，如果 j 是根，则父级为 0。示例

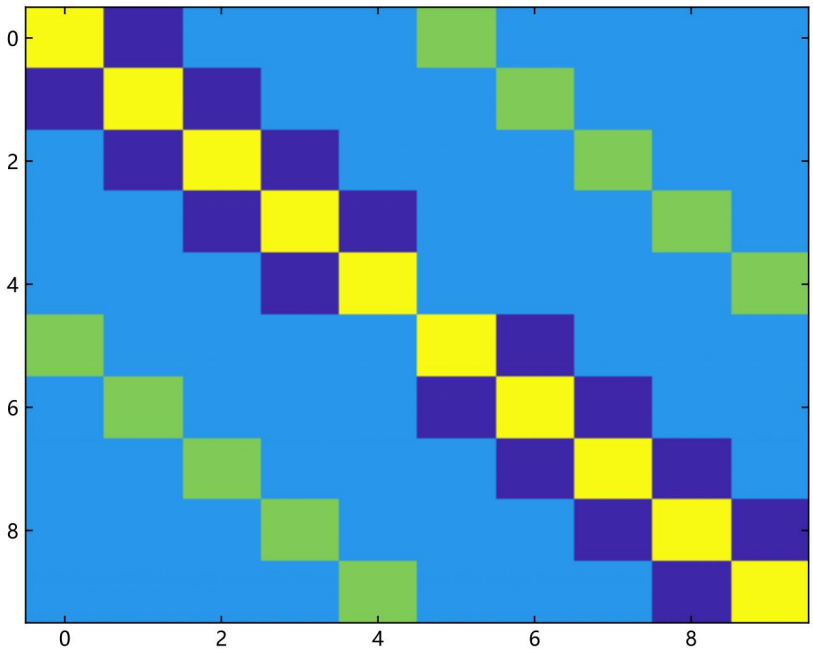
p,q = etree(A,method) 还指定生成生成消去树矩阵的方法，具体选择见下表。

method	选用矩阵
"sym"（默认）	triu(A)
"lo"	tril(A)
"col"	A'*A
"row"	A*A'

C. 示例

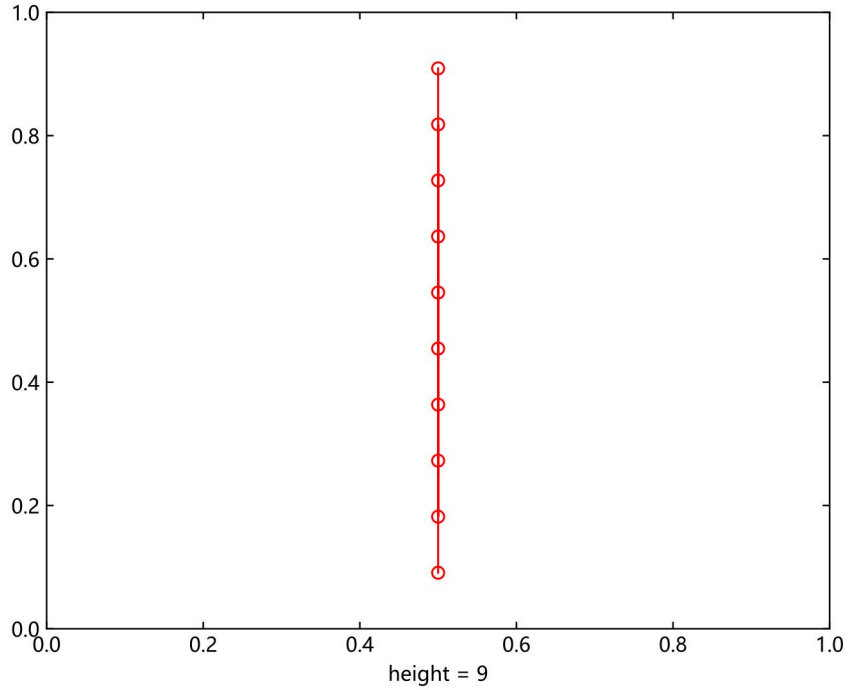
(1). 稀疏矩阵的消去树

构造一个稀疏矩阵，该矩阵对角为两个三对角阵，而复对角为单位阵，绘制其稀疏模式。



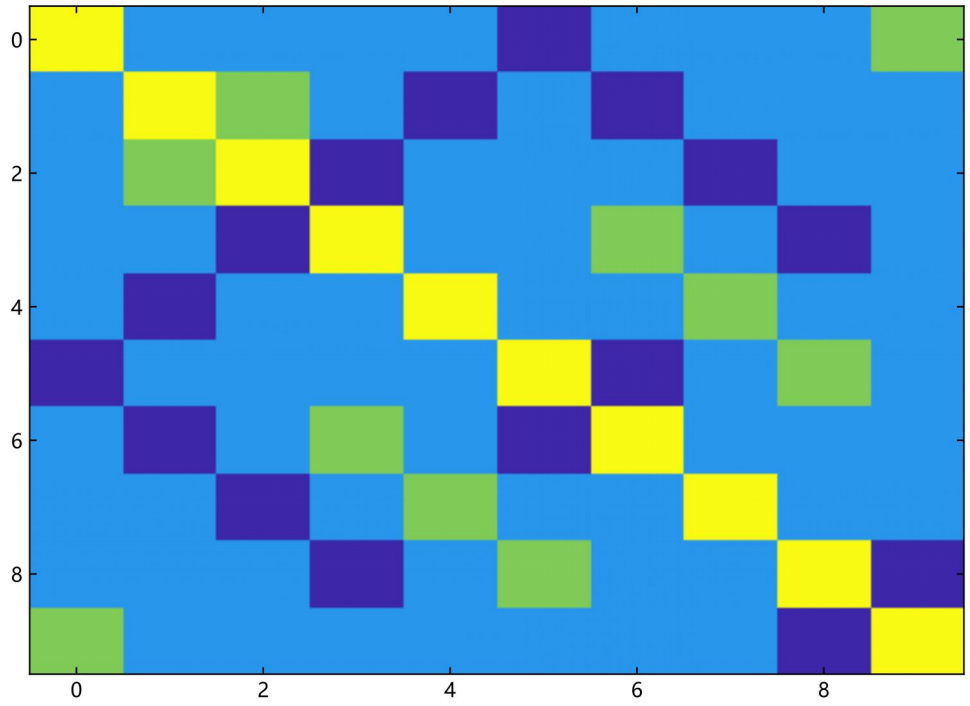
矩阵 A 的消去树没有分叉

```
S1, = etree(A)
treeplot(S1)
```



使用一个向量对矩阵 A 进行重排序，绘制排序后矩阵的稀疏模式。

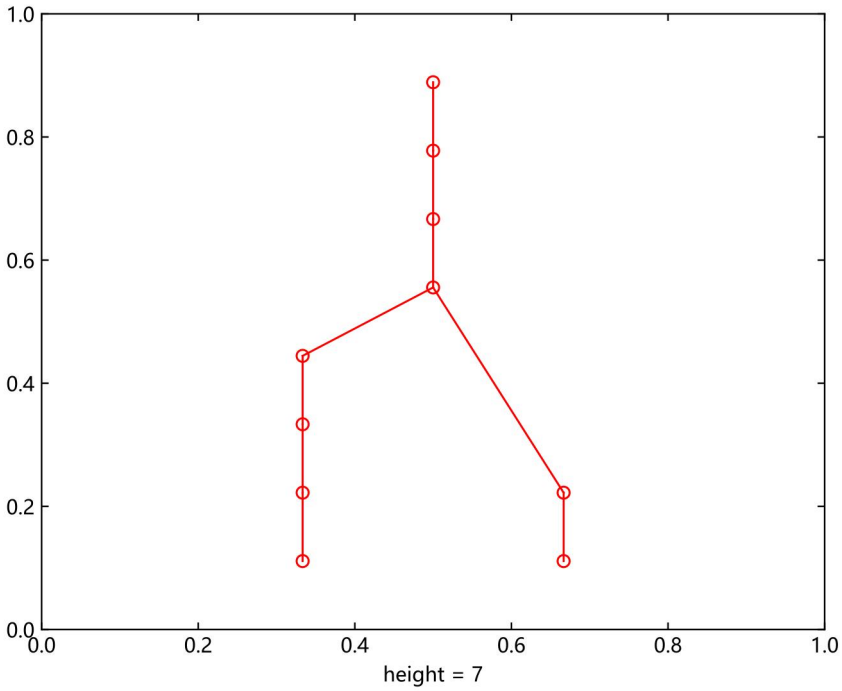
```
v = [1, 4, 9, 8, 5, 2, 3, 10, 7, 6]
B = A[v,v]
imagesc(B)
```



而矩阵 B 的消去树存在分叉，这表明 B 的 cholesky 分解会比 A 更加稀

疏。

```
S2, = etree(B)
treeplot(S2)
```



3.2.7.4.3 spaugment

构造最小二乘增广方程组

A. 语法

```
S = spaugment(A,c)
S = spaugment(A)
```

B. 说明

S = spaugment(A,c) 创建对称不定稀疏方阵 S = [c*I A; A' 0]。矩阵 S 与最小二乘问题有关

```
min norm(b - A*x)
```

by

```
r = b - A*x
S * [r/c; x] = [b; 0]
```

残差缩放因子 c 的最优值，涉及计算成本往往非常昂贵的 min(svd(A)) 和 norm(r)。

S = spaugment(A)（不带 c 的指定值）使用 max(max(abs(A)))/1000。

3.2.7.4.4 DMDecomposition

Dulmage-Mendelsohn 分解模块

A. 语法

DMDecomposition 模块中的函数以 DMDecomposition.函数名 形式进行调用，例如 DMDecomposition.dmperm 调用该模块的 dmperm 函数。

B. 内容

该模块包含以下函数作为其模块子函数。

函数名	简介
dmperm	Dulmage-Mendelsohn 分解
maxtrans	最大匹配

3.2.7.4.5 treelayout

设置树或森林的布局

A. 语法

```
x,y,h,s=treelayout(tree)
x,y,h,s=treelayout(tree,permutation)
```

B. 说明

参数 TREE 是父节点的向量。

参数 PERMUTATION 是一个可选的后序排列。

算法的复杂度在时间和内存方面都是 O(n)要求。

输出结果分别为 x 轴坐标、y 轴坐标、树高度、顶点数

3.2.7.4.6 treeplot

绘制树形图

A. 语法

```
treeplot(p,node_style = "ko", edge_style = "r")
```

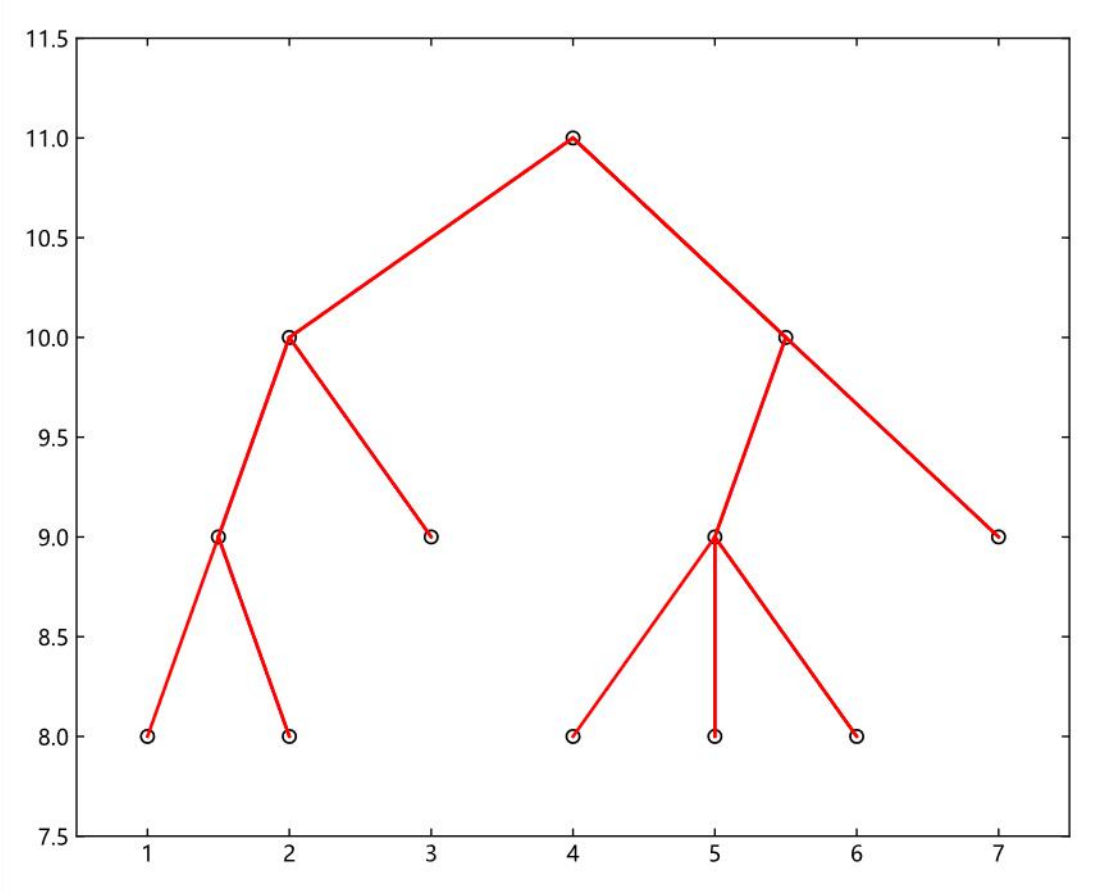
B. 说明

treeplot(p,nodeSpec,edgeSpec) 可使用可选参数 nodeSpec 和 edgeSpec 设置节点或边颜色、标记和线型。

C. 示例

要绘制 12 个节点的树形图，请通过一个 12 元素输入向量来调用 treeplot。该向量中每个元素的索引显示在下图中每个节点的旁边。（显示这些索引仅用于演示示例；它们并不是 treeplot 输出的一部分。）

```
nodes = [0, 1, 2, 2, 4, 4, 4, 1, 8, 8, 10, 10]
treeplot(nodes)
```



3.2.7.4.7 etreeplot

绘制消去树

A. 语法

```
etreeplot(A)
etreeplot(A,nodeSpec,edgeSpec)
```

B. 说明

etreeplot(A) 绘制 A 的消去树（如果不是对称的，则为 A+A'）。

etreeplot(A,nodeSpec,edgeSpec) 可使用可选参数 nodeSpec 和 edgeSpec 设置节点或边颜色、标记和线型。使用 "" 可省略一项或者都省略。

3.2.7.4.8 gplot

绘制邻接矩阵中的节点和边

A. 语法

```
gplot(A,XYCoords)
gplot(A,XYCoords,LineSpec)
x,y = gplot(A,XYCoords)
```

B. 说明

gplot(A,XYCoords) 在 XYCoords 中指定的坐标处绘制邻接矩阵 A 中定义的节点和边的图。邻接矩阵 A 是 n×n 矩阵，其中 n 是节点数。XYCoords 是 n x 2 矩阵，指定每个节点的 xy 坐标。

gplot(A,XYCoords,LineSpec) 进一步使用 LineSpec 指定绘图中要使用的线型、标记符号和颜色。例如，gplot(A,XY,'r-*') 使用红线表示边，红色星号表示节点。

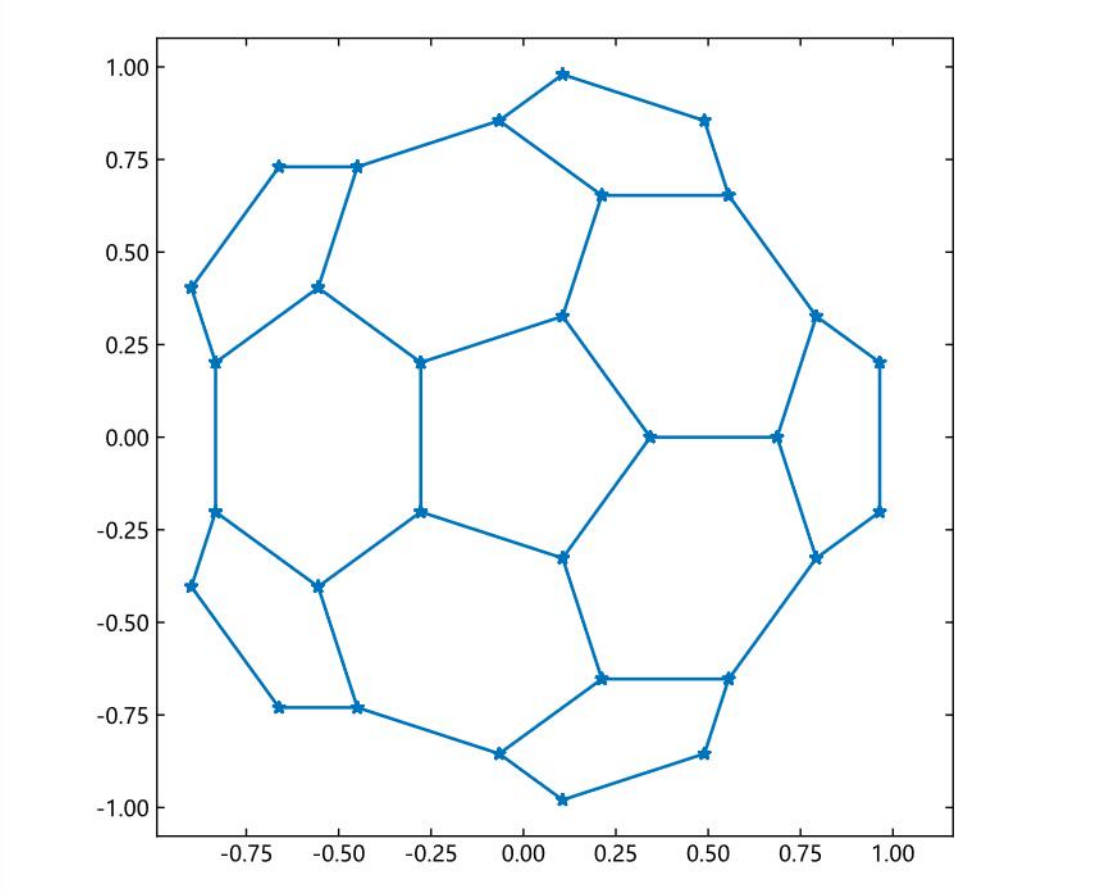
x,y = gplot(A,XYCoords) 返回以 NaN 分隔的向量 x 和 y，而不生成绘图。使用 x 和 y，可在稍后使用 plot(x,y) 生成绘图。

C. 示例

(1). 使用星号绘制节点图。

绘制半个碳 60 分子，在每个节点上放置星号。

```
import TyMath
pkg_dir = pkgdir(TyMath)
source_path = pkg_dir * "/examples/Sparse Matrices/Structural
Analysis/gplot_tests/data.jl"
include(source_path)
gplot(B[k,k],XY[k,1:2],"-*")
axis("square")
```



3.2.7.4.9 unmesh

将边矩阵转换为坐标和拉普拉斯矩阵

A. 语法

```
L,XY= unmesh(E)
```

B. 说明

L,XY= unmesh(E)返回 M×4 边矩阵 E 的拉普拉斯矩阵 L 和网格顶点坐标矩阵 XY。边矩阵的每一行必须包含边端点 [x1 y1 x2 y2] 的坐标。

3.2.7.5 迭代算法

稀疏矩阵迭代算法

函数名	简介
bicg	求解线性方程组 - 双共轭梯度法

bicgstab	求解线性方程组 - 稳定双共轭梯度法
bicgstabl	求解线性方程组 - 稳定双共轭梯度（I）法
cg	求解线性方程-预条件共轭梯度法
cgs	求解线性方程组-共轭梯度二乘法
qmr	求解线性方程组-拟最小残差法
lsqr	求解线性方程组-最小二乘法
jacobi	求解线性方程-Jacobid 迭代算法（基础算法）
gauss_seidel	求解线性方程-gauss_seidel 迭代法（基础算法）
sor	求解线性方程-连续过度松弛迭代法（基础算法）
minres	求解线性方程-最小残差法
symmlq	求解线性方程组 - 对称的 LQ 方法
gmres	求解线性方程-广义最小残差法
tfqmr	求解线性方程组 - 无转置拟最小残差法

3.2.7.5.1 bicg

求解线性方程- 双共轭梯度法

A. 语法

```
x, output = bicg(A, b)
_ = bicg(A, b, atol=sqrt(eps()))
_ = bicg(A, b, rtol=sqrt(eps()))
_ = bicg(A, b, itmax=800)
_ = bicg(A, b, verbose=0)
_ = bicg(A, b, history=false)
```

B. 说明

x, output = bicg(A, b) 尝试使用双共轭梯度法求解关于 x 的线性方程组 A*x = b。 x 为迭代的最终解， output 为迭代信息结构体。示例

_ = bicg(A, b,atol=sqrt(eps())) 指定该方法的绝对容差。

_ = bicg(A,b,rtol=sqrt(eps())) 指定该方法的相对容差。

_ = bicg(A,b,itmax=0) 指定要使用的最大迭代次数。如果该项为 0，则最大

迭代次数为 2^size(A,1)。

_ = bicg(A,b,verbose=0) 指定打印迭代过程信息的间隔，如果该项为 0，则不会打印迭代过程信息。

_ = bicg(A,b,history=false) 指定残差范数向量返回指示，如果该项为 true ，还会在 output 中返回在每次迭代中的残差范数向量（包括第一个残差 norm(b-A*x0)）。

C. 示例

(1). 线性方程组的迭代解

使用采用默认设置的 bicg 求解系数矩阵为方阵的线性方程组，然后在求解过程中调整使用的容差和迭代次数。

创建密度为 50% 的随机稀疏矩阵 A。另为 $Ax = b$ 的右侧创建随机向量 b

```
rng = MersenneTwister(1234)
A = sprand(rng,400,400,0.5)
A = A'*A
b = rand(rng, 400)
```

使用 bicg 求解 $Ax = b$ 。

```
println(bicg(A, b)[2].status)
```

```
maximum number of iterations exceeded
```

我们为迭代赋予更大的容差以及更大的迭代最大次数。

```
x, output = bicg(A,b,atol=1e-4,rtol=1e-4,itmax=1000, history=true)
isapprox(A*x-b,zeros(400),atol=1e-3,rtol=1e-3)
```

```
ans = true
```

3.2.7.5.2 bicgstab

求解线性方程-稳定双共轭梯度法

A. 语法

```
x, stats = bicgstab(A, b; kwargs...)
x, stats = bicgstab(A, b, x0; kwargs...)
```

B. 说明

x = bicgstab(A, b; kwargs...) 尝试使用双共轭梯度稳定法求解关于 x 的线性系统 A*x = b。

x = bicgstab(A, b, x0; kwargs...)。

C. 示例

(1). 线性系统的迭代解

使用采用默认设置的 bicgstab 求解系数矩阵为方阵的线性系统，然后在求解过程中调整使用的容差和迭代次数。

创建密度为 50% 的随机稀疏矩阵 A。另为 Ax=b 的右侧创建随机向量 b。

```
rng = MT19937ar(5489);
A = sprand(rng,400,400,0.5);
A = A'*A;
b = rand(rng,400);
```

使用 bicgstab 求解 $Ax = b$。输出显示包括相对残差 $\frac{\left | b-Ax \right | }{\left | b \right | } $ 的值

```
x, stats = bicgstab(A, b; atol=1e-6, rtol=1e-6, itmax=10000);
```

```
SimpleStats
 niter: 1044
 solved: true
 inconsistent: false
 residuals: []
 Aresiduals: []
 κ₂(A): []
 timer: 168.11ms
 status: solution good enough given atol and rtol
```

设置容差为 1e-6 ，再经过 1044 次迭代之后，得到结果。

(2). 使用指定了预条件子的 bicgstab

使用指定了预条件子的 bicgstab

检查使用指定了预条件子矩阵的 bicgstab 来求解线性系统的效果。

加载矩阵 A，它是一个非对称的 479×479 实稀疏矩阵。

```
filename = pkgdir(TyMath) * "/examples/Sparse Matrices/Iterative
Methods and Preconditioners/bicgstab_tests/bicgstab_data1.jl";
include(filename);
```

定义 b 以使 $Ax = b$ 的实际解是全为 1 的向量。

```
b = vec(sum(A,dims=2));
```

设置容差和最大迭代次数。

```
atol = 1e-12;
rtol = 1e-12;
maxit = 20;
```

使用 bicgstab 根据请求的容差和迭代次数求解。

```
x, stats = bicgstab(A, b; atol=atol, rtol=rtol, itmax=maxit);
```

```
stats =
SimpleStats
 niter: 20
 solved: false
 inconsistent: false
 residuals: []
 Aresiduals: []
 κ₂(A): []
 timer: 96.40μs
 status: maximum number of iterations exceeded
```

bicgstab 未在请求的 20 次迭代内收敛至请求的容差 1e-12 。

(3). 提供初始估计值

检查向 bicgstab 提供解的初始估计值的效果。

创建一个三对角稀疏矩阵。使用每行的总和作为 $Ax = b$ 右侧的向量，使 x 的预期解是由 1 组成的向量。

```
n = 900;
e = ones(n);
A = spdiagm(-1=>ones(n-1),0=>2*e,1=>ones(n-1));
b = vec(sum(A,dims=2));
```

使用 bicgstab 求解 $Ax = b$ 两次：一次是使用默认的初始估计值，一次是使用解的良好初始估计值。对两次求解均使用 50 次迭代。将第二种求解中的初始估计值指定为所有元素都等于 0.99 的向量。

```
maxit = 50;
x1,stats1 = bicgstab(A, b; atol=1e-6, rtol=1e-6, itmax=maxit);
```

```
stats1 =
SimpleStats
 niter: 21
 solved: true
 inconsistent: false
 residuals: []
 Aresiduals: []
```

```
 κ₂(A): []
 timer: 177.10μs
 status: solution good enough given atol and rtol
```

```
x0 = 0.99 .* e;
x2, stats2 = bicgstab(A, b, x0; atol=1e-6, rtol=1e-6, itmax=maxit);
```

```
stats2 =
SimpleStats
 niter: 17
 solved: true
 inconsistent: false
 residuals: []
 Aresiduals: []
 κ₂(A): []
 timer: 124.60μs
 status: solution good enough given atol and rtol
```

在这种情况下，提供初始估计值可以使 bicgstab 更快地收敛。

(4). 返回中间结果

可以通过设置关键字参数 verbose 或者 history 的值来返回中间结果。例如：

使用 verbose 参数查看每次迭代的详细信息

```
x, stats = bicgstab(A, b, x0; atol=1e-6, rtol=1e-6, itmax=maxit,
verbose=1);
```

```
BICGSTAB: system of size 900
    k     ‖ rₖ ‖      |αₖ |      |ωₖ |  timer
    0  1.2e+00   1.0e+00   1.0e+00  0.14s
    1  2.0e-03   2.5e-01   3.3e-01  0.23s
    2  5.0e-04   3.6e-01   5.0e-01  0.23s
    3  2.1e-04   4.7e-01   4.2e-01  0.23s
    4  1.0e-04   5.5e-01   5.3e-01  0.24s
    5  6.0e-05   6.0e-01   4.3e-01  0.24s
    6  3.8e-05   6.5e-01   5.4e-01  0.24s
    7  2.5e-05   6.9e-01   4.3e-01  0.24s
    8  1.8e-05   7.2e-01   5.4e-01  0.24s
    9  1.3e-05   7.4e-01   4.4e-01  0.24s
   10  9.5e-06   7.6e-01   5.5e-01  0.24s
   11  7.3e-06   7.8e-01   4.4e-01  0.24s
   12  5.7e-06   7.9e-01   5.5e-01  0.24s
   13  4.5e-06   8.1e-01   4.4e-01  0.24s
   14  3.6e-06   8.2e-01   5.5e-01  0.24s
   15  3.0e-06   8.3e-01   4.4e-01  0.24s
```

```
  16  2.5e-06   8.4e-01   5.5e-01  0.24s
  17  2.1e-06   8.5e-01   4.4e-01  0.24s
```

使用 history 参数查看残差。

```
x, stats = bicgstab(A, b, x0; atol=1e-6, rtol=1e-6, itmax=200,
history=true);
stats.residuals
```

```
stats.residuals =
18-element Vector{Float64}:
 1.1994165248152935
 0.0020409947280188084
 0.000503154163801355
 0.00020690975995630938
 0.00010472368707288741
 6.0281095783356894e-5
 3.7771533360471885e-5
 2.5171935973188128e-5
 1.75733884082934e-5
 1.2728231515782644e-5
 9.496611123495845e-6
 7.262267478675408e-6
 5.669786203530408e-6
 4.505704409311849e-6
 3.6358451636050975e-6
 2.9734915463588405e-6
 2.4606425170589943e-6
 2.0577190722919773e-6
```

3.2.7.5.3 bicgstabl

求解线性方程-稳定双共轭梯度（I）法

A. 语法

```
x = bicgstabl(A, b)
x = bicgstabl(A, b, max_mv_products=size(A,2))
x = bicgstabl(A, b, abstol = zero(real(eltype(b))))
x = bicgstabl(A, b, reltol = sqrt(eps(real(eltype(b)))))
x, msg = bicgstabl(A, b, log = false)
x = bicgstabl(A, b, initial_zero = true)
x = bicgstabl(A, b, verbose = false)
```

B. 说明

x = bicgstabl(A, b) 尝试使用稳定双共轭梯度（I）法求解关于 x 的线性方程组 A*x = b。示例

x = bicgstabl(A,b,max_mv_products=size(A,2)) 指定该方法的最大矩阵-向量积的次数，表征最大迭代次数。默认容差是为 size(A,2) 。

x = bicgstabl(A,b,abstol=zero(real(eltype(b)))) 指定停止条件中的绝对容差，默认为 0 。

x = bicgstabl(A,b,reltol=sqrt(eps(real(eltype(b))))) 指定停止条件中的相对容差，默认为 sqrt(eps()) 。

x, msg = bicgstabl(A,b,log = false) 指定是否同时返回迭代法的收敛信息，默认值为 false 。当该项为 true，msg 中将包含迭代是否收敛及其算法停止时的迭代次数。

x = bicgstabl(A,b,initial_zero = true) 指定是否使用全零向量作为迭代法的初始值，默认为 true。当指定该值为 false 时，算法将使用随机数作为初始值。

x = bicgstabl(A,b,verbose = false) 指定算法是否打印迭代的过程信息，默认值为 false。如果该项为 true，则会打印迭代过程信息，包括迭代次数及其对应的容差。

C. 示例

(1). 提供绝对容差与相对容差

创建一个三对角稀疏矩阵。使用每行的总和作为 $Ax = b$ 右侧的向量，使 x 的预期解是由 1 组成的向量

```
n = 900
e = vec(ones(n,1))
A = spdiagm(n,n,-1=>e[1:end-1],0=>2 .*e,1=>e[1:end-1])
b = sum(A,dims=2)
```

使用 bicgstabl 求解 $Ax = b$，绝对容差选为 1e-6，相对容差选为 1e-6。

```
x = bicgstabl(A,b, abstol=1e-6, reltol=1e-6)
```

检验 x 与 1 之间的误差。

```
isapprox(x, ones(900), atol=1e-3, rtol=1e-3)
```

```
ans = true
```

3.2.7.5.4 cg

求解线性方程-预条件共轭梯度法

A. 语法

```
(x,stats)=cg(A,b::AbstrctVector{T};atol::T= √ eps(T), rtol::T= √ eps(T))
```

B. 说明

(x,stats)=cg(A,b::AbstrctVector{T};atol::T= √ eps(T), rtol::T= √ eps(T))，用于求解对称线性系统 Ax=b 的共轭梯度法，

其中如果 A 不确定，则该方法不会终止，可以以线性算子的形式提供预处理器 M，并假设它是对称且正定的。

M 也表示测量残差的加权范数，如果 itmax = 0,则默认迭代次数设置为 2*n,其中 n=length(b)。

C. 示例

(1). 正定矩阵-预条件共轭梯度法求解

创建线性对称系统，利用预条件共轭梯度法求解

```
cg_tol=1e-6
A,b=[[1 2 3;2 8 10;3.0 10 5],[1e-9,5.5 ,9.998]]##确定系数矩阵和 b
(x,stats)=cg(A,b)
```

```
([-2.1877499981249975,    1.9372499993750085,    -0.5622499998749952],
Simple stats

niter: 3

solved: true

inconsistent: false

residuals: []

Aresiduals: []

κ₂(A): []
```

```
status: solution good enough given atol and rtol
```

(2). 对角矩阵-预条件共轭梯度法求解

创建线性对称系统，利用预条件共轭梯度法求解

```
cg_tol=1e-6
A, b = [1.0 0 0;0 5 0;0 0 8],[5.0,5.0,5.0]
(x,stats)=cg(A,b)
```

```
([5.0, 1.0, 0.625], Simple stats

niter: 3
solved: true
inconsistent: false
residuals: []
Aresiduals: []
κ₂(A): []
status: solution good enough given atol and rtol
)
```

3.2.7.5.5 cgs

求解线性方程-共轭梯度二乘法

A. 语法

```
x, stats = cgs(A, b; kwargs...)
x, stats = cgs(A, b, x0; kwargs...)
```

B. 说明

x = cgs(A, b; kwargs...) 尝试使用共轭梯度二乘法求解关于 x 的线性系统 。

x = cgs(A, b, x0; kwargs...)。

C. 示例

(1). 线性系统的迭代解

使用采用默认设置的 cgs 求解系数矩阵为方阵的线性系统，然后在求解过程中调整使用的容差和迭代次数。

创建密度为 50% 的随机稀疏矩阵 A。另为 $Ax = b$ 的右侧创建随机向量

```
rng = MT19937ar(5489);
A = sprand(rng,600,600,0.5);
```

```
A = A'*A + speye(size(A));
b = rand(rng,600);
```

使用 cgs 求解 。输出显示包括相对残差 $\frac{\left | b-Ax \right | }{\left | b \right | } $ 的值

```
x, stats = cgs(A,b;atol=1e-6,rtol=1e-6)
```

```
stats =
SimpleStats
 niter: 74
 solved: true
 inconsistent: false
 residuals: []
 Aresiduals: []
 κ₂(A): []
 timer: 38.76ms
 status: solution good enough given atol and rtol
```

设置容差为 1e-6 ，再经过 74 次迭代之后得到结果

(2). 使用指定了预条件子的 cgs

检查使用指定了预条件子矩阵的 cgs 来求解线性系统的效果。

加载矩阵 A，它是一个非对称的 479×479 实稀疏矩阵

```
filename = pkgdir(TyMath) * "/examples/Sparse Matrices/Iterative
Methods and Preconditioners/cgs_tests/cgs_data1.jl";
include(filename);
```

定义 b 以使 $Ax = b$ 的实际解是全为 1 的向量。

```
b = vec(sum(A,dims=2));
```

设置容差和最大迭代次数

```
atol = 1e-12;
rtol = 1e-12;
maxit = 20;
```

使用 cgs 根据请求的容差和迭代次数求解。

```
x, stats = cgs(A, b; atol=atol, rtol=rtol, itmax=maxit)
```

```
stats =
SimpleStats
 niter: 20
 solved: false
 inconsistent: false
 residuals: []
 Aresiduals: []
 κ₂(A): []
```

```
 timer: 94.50μs
 status: maximum number of iterations exceeded
```

(3). 提供初始估计值

检查向 cgs 提供解的初始估计值的效果。

创建一个三对角稀疏矩阵。使用每行的总和作为 $Ax = b$ 右侧的向量，使 x 的预期解是由 1 组成的向量

```
n = 900;
e = ones(n);
A=spdiagm(-1=>ones(n-1),0=>2*e,1=>ones(n-1));
b = vec(sum(A,dims=2));
```

使用 cgs 求解 $Ax = b$ 两次：一次是使用默认的初始估计值，一次是使用解的良好初始估计值。对两次求解均使用 200 次迭代和默认容差。将第二种求解中的初始估计值指定为所有元素都等于 0.99 的向量。。

```
maxit = 200;
x1,stats1 = cgs(A, b; atol=1e-6, rtol=1e-6, itmax=maxit);
```

```
stats1 =
SimpleStats
 niter: 17
 solved: true
 inconsistent: false
 residuals: []
 Aresiduals: []
 κ₂(A): []
 timer: 136.80μs
 status: solution good enough given atol and rtol
```

```
x0 = 0.99 .* e;
x2, stats2 = cgs(A, b, x0; atol=1e-6, rtol=1e-6, itmax=200);
```

```
stats2 =
SimpleStats
 niter: 14
 solved: true
 inconsistent: false
 residuals: []
 Aresiduals: []
 κ₂(A): []
 timer: 119.60μs
 status: solution good enough given atol and rtol
```

在这种情况下，提供初始估计值可以使 cgs 更快地收敛。

3.2.7.5.6 qmr

求解线性方程-拟最小残差法

A. 语法

```
x, stats = qmr(A, b; kwargs...)
x, stats = qmr(A, b, x0; kwargs...)
```

B. 说明

x = qmr(A, b; kwargs...) 尝试使用拟最小残差法求解关于 x 的线性系统 $Ax = b$。

x = qmr(A, b, x0; kwargs...)。

C. 示例

(1). 线性系统的迭代解

使用采用默认设置的 qmr 求解系数矩阵为方阵的线性系统，然后在求解过程中调整使用的容差和迭代次数。

创建密度为 50% 的随机稀疏矩阵 A。另为 $Ax = b$ 的右侧创建随机向量.

```
rng = MT19937ar(5489);
A = sprand(rng,400,400,0.5);
A = A'*A;
b = rand(rng,400);
```

使用 qmr 求解 。输出显示包括相对残差 $\frac{\left | b-Ax \right | }{\left | b \right | } $ 的值

```
x, stats = qmr(A,b;atol=1e-6,rtol=1e-6,itmax=1000)
```

```
SimpleStats
 niter: 904
 solved: true
 inconsistent: false
 residuals: []
 Aresiduals: []
 κ₂(A): []
 timer: 201.74ms
 status: solution good enough given atol and rtol
```

设置容差为 1e-6 ，再经过 904 次迭代之后得到结果。

(2). 使用指定了预条件子的 qmr

检查使用指定了预条件子矩阵的 qmr 来求解线性系统的效果。

加载矩阵 A，它是一个非对称的 479×479 实稀疏矩阵

```
filename = pkgdir(TyMath) * "/examples/Sparse Matrices/Iterative Methods and Preconditioners/qmr_tests/qmr_data1.jl";
include(filename);
```

定义 b 以使 $Ax = b$ 的实际解是全为 1 的向量。

```
b = vec(sum(A,dims=2));
```

设置容差和最大迭代次数

```
atol = 1e-12;
rtol = 1e-12;
maxit = 20;
```

使用 qmr 根据请求的容差和迭代次数求解。

```
x, stats = qmr(A, b; atol=atol, rtol=rtol, itmax=maxit)
```

```
SimpleStats
 niter: 20
 solved: false
 inconsistent: false
 residuals: []
 Aresiduals: []
 κ₂(A): []
 timer: 104.70μs
 status: maximum number of iterations exceeded
```

qmr 未在请求的 20 次迭代内收敛至请求的容差 1e-12

```
x, stats = qmr(A, b; atol=atol, rtol=rtol)
```

```
stats =
SimpleStats
 niter: 958
 solved: false
 inconsistent: false
 residuals: []
 Aresiduals: []
 κ₂(A): []
 timer: 4.44ms
 status: maximum number of iterations exceeded
```

(3). 提供初始估计值

检查向 qmr 提供解的初始估计值的效果。

创建一个三对角稀疏矩阵。使用每行的总和作为 $Ax = b$ 右侧的向量，使 x 的预期解是由 1 组成的向量

```
n = 900;
e = ones(n);
A=spdiagm(-1=>ones(n-1),0=>2*e,1=>ones(n-1));
b = vec(sum(A,dims=2));
```

使用 qmr 求解 $Ax = b$ 两次：一次是使用默认的初始估计值，一次是使用解的良好初始估计值。对两次求解均使用 200 次迭代和默认容差。将第二种求解中的初始估计值指定为所有元素都等于 0.99 的向量。。

```
maxit = 200;
x1,stats1 = qmr(A, b; atol=1e-6, rtol=1e-6, itmax=maxit);
```

```
stats1 =
SimpleStats
 niter: 48
 solved: true
 inconsistent: false
 residuals: []
 Aresiduals: []
 κ₂(A): []
 timer: 357.90μs
 status: solution good enough given atol and rtol
```

```
x0 = 0.99 .* e;
x2, stats2 = qmr(A, b, x0; atol=1e-6, rtol=1e-6, itmax=maxit);
```

```
stats =
SimpleStats
 niter: 39
 solved: true
 inconsistent: false
 residuals: []
 Aresiduals: []
 κ₂(A): []
 timer: 856.90μs
 status: solution good enough given atol and rtol
```

在这种情况下，提供初始估计值可以使 cgs 更快地收敛。

(4). 返回中间结果

可以通过设置关键字参数 verbose 或者 history 的值来返回中间结果。例如：使用 verbose 参数查看每次迭代的详细信息

```
x2, stats2 = qmr(A, b, x0; atol=1e-6, rtol=1e-6, itmax=200,
```

```
verbose=1);
```

```
QMR: system of size 900
    k     ‖ r_k ‖  timer
    0  1.2e+00  0.10s
    1  1.1e-02  0.22s
    2  3.5e-03  0.22s
    3  1.6e-03  0.22s
    4  8.5e-04  0.22s
    5  5.1e-04  0.22s
    6  3.3e-04  0.22s
    7  2.3e-04  0.22s
    8  1.6e-04  0.22s
    9  1.2e-04  0.22s
   10  9.1e-05  0.22s
   11  7.1e-05  0.22s
   12  5.6e-05  0.22s
   13  4.6e-05  0.22s
   14  3.7e-05  0.22s
   15  3.1e-05  0.22s
   16  2.6e-05  0.22s
   17  2.2e-05  0.22s
   18  1.9e-05  0.22s
   19  1.6e-05  0.22s
   20  1.4e-05  0.22s
   21  1.2e-05  0.22s
   22  1.1e-05  0.22s
   23  9.5e-06  0.22s
   24  8.4e-06  0.22s
   25  7.5e-06  0.22s
   26  6.7e-06  0.22s
   27  6.0e-06  0.22s
   28  5.4e-06  0.22s
   29  4.9e-06  0.22s
   30  4.5e-06  0.22s
   31  4.1e-06  0.22s
   32  3.7e-06  0.22s
   33  3.4e-06  0.22s
   34  3.1e-06  0.22s
   35  2.9e-06  0.22s
   36  2.6e-06  0.22s
   37  2.4e-06  0.22s
   38  2.3e-06  0.22s
```

```
  39  2.1e-06  0.22s
```

使用 history 参数查看残差。

```
x2, stats2 = qmr(A, b, x0; atol=1e-6, rtol=1e-6, itmax=200, history=true);
stats2.residuals
```

```
stats2.residuals =
40-element Vector{Float64}:
 1.1994165248152935
 0.011168425392719557
 0.0035332663166038583
 0.0015883549480437361
 0.0008531648023268813
 0.0005119306096699625
 0.00033154245861271274
 0.00022710141942937893
 0.0001624036635123474
 0.00012017062740675049
 9.142359519732712e-5
 7.117416416156292e-5
 5.649684441500204e-5
 4.559746805442629e-5
 3.733346434903592e-5
 3.0953427686894546e-5
 2.594926101991688e-5
 2.196893904549386e-5
 1.876328615922416e-5
 1.6152538143363236e-5
 1.4004779738815684e-5
 1.2221713748285086e-5
 1.072907102756515e-5
 9.470025107196667e-6
 8.400592132597955e-6
 7.486367134522183e-6
 6.700175136563179e-6
 6.020358340042901e-6
 5.4295119264609195e-6
 4.913540463814536e-6
 4.460946247957008e-6
 4.062287348407142e-6
 3.7097611451221146e-6
 3.396881582787852e-6
 3.118227064064024e-6
```

```
2.8692420502865257e-6
2.646079830755324e-6
2.445477092404471e-6
2.2646532320743694e-6
2.101229052286789e-6
```

3.2.7.5.7 lsqr

求解线性方程组-最小二乘法

A. 语法

```
(x,stats)=lsqr(A,b::AbstrctVector{T};atol::T= √ eps(T),  rtol::T= √ eps(T))
```

B. 说明

(x,stats)=lsqr(A,b::AbstrctVector{T};atol::T= √ eps(T), rtol::T= √ eps(T)) 解决正则化线性最小二乘问题

minimize ‖ b - Ax ‖ $_2^2$ + λ² ‖ x ‖ $_2^2$,使用 LSQR 方法，其中 λ ≥ 0 是正则化参数。

LSQR 在形式上等同于将 CG 应用于 正规方程($A^TA + λ^2I$) x = A^Tb,因此相对 CGLS 但更稳定。LSQR 产生单调残差 ‖ r ‖ $_2$ 但不产生最优残差 ‖ A^Tr ‖ $_2$。

它在形式上等同于 CGLS，但准确性更高一点。预处理器 M 和 N 可以以线性算子的形式提供，并假定为对称和正定的。

如果设置了 sqd 为真，我们求解对称和拟定系统

[E A] [r] [b]

[A^T -F] [x] = [0],

上面的系统代表了最优条件，最小化 ‖ b - Ax ‖ ² _E^{-1}。在这种情况下，N 仍然可以指定，并表示应该测量 x 和 A^Tr 的加权范数。

r 可以通过计算恢复 E^{-1}(b - Ax)。

C. 示例示

(1). 求解正则化线性最小二乘问题

创建线性对称系统，利用预条件共轭梯度法求解

```
A,b=[[1 2 3;2 8 10;3.0 10 5],[1e-9,5.5 ,9.998]]##确定系数矩阵和 b
```

```
(x,stats)=lsqr(A,b)
```

```
([-2.1877499981255406,    1.9372499993730874,    -0.5622499998767085],
Simple stats

niter: 3

solved: true

inconsistent: false

residuals: []

Aresiduals: []

κ₂(A): []

status: found approximate zero-residual solution

)
```

(2). 对角矩阵-求解正则化线性最小二乘问题

创建线性对称系统，利用预条件共轭梯度法求解

```
lsqr_tol=1e-6
A, b = [1.0 0 0;0 5 0;0 0 8],[5.0,5.0,5.0]
(x,stats)=lsqr(A,b,atol=lsqr_tol)
```

```
([5.000000000000001, 1.0000000000000004, 0.6250000000000043], Simple
stats

niter: 3

solved: true

inconsistent: false

residuals: []

Aresiduals: []

κ₂(A): []

status: found approximate zero-residual solution
```

)

3.2.7.5.8 jacobi

求解线性方程-Jacobid 迭代算法（基础算法）

A. 语法

```
x=jacobi(A,b)
```

B. 说明

x=jacobi(A,b) 返回零初始化的解向量 x。

C. 示例

(1). 利用雅克比算法求解方程组

求解方程组 $\begin{cases} 8x_1 - 3x_2 + 2x_3 = 20 \\ 4x_1 + 11x_2 - x_3 = 33 \\ 6x_1 + 3x_2 + 12x_3 = 36 \end{cases}$

```
A=[8 -3 2;4 11 -1;6 3 12]
b=[20,33,36]
x=jacobi(A,b)
```

```
3-element Vector{Float64}:
3.0000318140697306
1.9998740186108073
0.9998812605453541
```

3.2.7.5.9 gauss_seidel

求解线性方程-gauss_seidel 迭代法（基础算法）

A. 语法

```
x=gauss_seidel(A,b)
```

B. 说明

x = gauss_seidel(A,b) 返回零初始化的解向量 x。

C. 示例

(1). 利用 gauss_seidel 算法求解方程组

求解方程组 $\begin{cases} 8x_1 - 3x_2 + 2x_3 = 20 \\ 4x_1 + 11x_2 - x_3 = 33 \\ 6x_1 + 3x_2 + 12x_3 = 36 \end{cases}$

```
A=[8 -3 2;4 11 -1;6 3 12]
b=[20,33,36]
x=gauss_seidel(A,b)
```

```
3-element Vector{Float64}:
3.0000000063222574
1.9999999980087824
0.9999999973366757
```

3.2.7.5.10 sor

求解线性方程-连续过度松弛迭代法（基础算法）

A. 语法

```
x=sor(A,b,w::Real)
```

B. 说明

x=sor(A,b,w::Real) 返回零初始化的解向量 x。

C. 示例

(1). 连续过度松弛迭代法求解

求解方程组 $\begin{cases} 8x_1 - 3x_2 + 2x_3 = 20 \\ 4x_1 + 11x_2 - x_3 = 33 \\ 6x_1 + 3x_2 + 12x_3 = 36 \end{cases}$

```
A=[8 -3 2;4 11 -1;6 3 12]
b=[20,33,36]
x=sor(A,b,1)
```

```
3-element Vector{Float64}:
3.0000016187261678
1.9999933501140443
```

```
0.9999966157508744
```

3.2.7.5.11 minres

求解线性方程-最小残差法

A. 语法

```
(x,stats)=minres(A,b::AbstrctVector{T};atol::T=  √  eps(T)/100,
rtol::T=√eps(T)/100)
```

B. 说明

(x,stats)=minres(A,b::AbstrctVector{T};atol::T= √ eps(T)/100, rtol::T= √ eps(T)/100)，解决移位线性最小二乘问题, minimize ‖ b - (A + λI)x ‖ ₂²或平移线性系统 (A + λI) x = b,使用 MINRES 方法，其中 λ ≥ 0 是移位参数， 其中 A 是正方形且对称的。

当 A 为正定时，MINRES 在形式上等同于将 CR 应用于 Ax=b，但通常更稳定，

也适用于这种情况,其中 A 是不定的,MINRES 产生单调残差 ‖ r ‖ ₂ 和最优残差 ‖ A^Tr ‖ ₂。可以以线性

算子的形式提供预处理器 M 并假定它是对称的和正定的。

C. 示例

(1). 正定矩阵-预条件共轭梯度法求解

创建线性对称系统，利用预条件共轭梯度法求解

```
cg_tol=1e-6
A,b=[[1 2 3;2 8 10;3.0 10 5],[1e-9,5.5 ,9.998]]##确定系数矩阵和 b
(x,stats)=minres(A,b)
```

```
x = 3-element Vector{Float64}:
 -2.1877499981250006
  1.9372499993750012
 -0.562249999874999

stats = Simple stats

niter: 3
```

```
solved: true

inconsistent: false

residuals: []

Aresiduals: []

κ₂(A): []

status: found approximate zero-residual solution
```

(2). 对角矩阵-预条件共轭梯度法求解

创建线性对称系统，利用预条件共轭梯度法求解

```
cg_tol=1e-6
A, b = [1.0 0 0;0 5 0;0 0 8],[5.0,5.0,5.0]
(x,stats)=minres(A,b)
```

```
x = 3-element Vector{Float64}:
4.999999999999998
1.0000000000000009
0.6250000000000004

 stats = Simple stats

niter: 3

solved: true

inconsistent: false

residuals: []

Aresiduals: []

κ₂(A): []

status: found approximate zero-residual solution
```

3.2.7.5.12 symmlq

求解线性方程-对称的 LQ 方法

A. 语法

```
x, stats = symmlq(A, b; kwargs...)
x, stats = symmlq(A, b, x0; kwargs...)
```

B. 说明

x = symmlq(A, b; kwargs...) 尝试使用对称的 LQ 方法求解关于 x 的线性系统 A*x = b。

x = symmlq(A, b, x0; kwargs...)。

C. 示例

(1). 线性系统的迭代解

使用采用默认设置的 symmlq 求解系数矩阵为方阵的线性系统，然后在求解过程中调整使用的容差和迭代次数。

创建密度为 50% 的随机稀疏矩阵 A。另为 $Ax = b$ 的右侧创建随机向量.

```
n = 400;
e = ones(n);
A=spdiagm(-1=>-2*ones(n-1),0=>4*e,1=>-2*ones(n-1));
b = vec(sum(A,dims=2));
```

使用 symmlq 求解 。输出显示包括相对残差 $\frac{\left | b-Ax \right |}{\left | b \right |} $ 的值

```
x, stats = symmlq(A, b; atol=1e-6, rtol=1e-6, itmax=20, history=true);
```

```
stats =
SymmlqStats
 niter: 20
 solved: false
 residuals: [ 2.8e+00  2.8e+00  2.3e+00 ...  2.0e+00  2.0e+00  2.0e+00 ]
 residuals (cg): [ 2.0e+00  9.4e-01  7.1e-01 ...  1.4e-01  1.3e-01
1.3e-01 ]
 errors: []
 errors (cg): []
 ‖A‖F: 22.360679774997898
 κ₂(A): 2.0818352612462596
```

```
 timer: 48.10μs
 status: maximum number of iterations exceeded
```

symmlq 使用 20 次迭代和容差 1e-6，对于此矩阵，算法无法在 20 次迭代后收敛。由于残差的数量级较大，因此需要更多迭代。也可以使用更大的容差，使算法更容易收敛。

使用容差 1e-4 和 250 次迭代再次求解方程组。

```
x, stats = symmlq(A, b; atol=1e-4, rtol=1e-4, itmax=250, history=true);
```

```
stats =
SymmlqStats
 niter: 166
 solved: true
 residuals: [ 2.8e+00  2.8e+00  2.3e+00 ...  2.0e+00  2.0e+00  2.0e+00 ]
 residuals (cg): [ 2.0e+00  9.4e-01  7.1e-01 ...  1.7e-02  1.7e-02  1.7e-02 ]
 errors: []
 errors (cg): []
 ‖A‖F: 63.2771680782255
 κ₂(A): 2.2160732862959436
 timer: 290.10μs
 status: found approximate solution
```

在经过了 166 次迭代之后得到结果。

(2). 使用指定了预条件子的 symmlq

检查使用指定了预条件子矩阵的 symmlq 来求解线性系统的效果。

创建一个对称正定带状系数矩阵。

```
A = delsq(numgrid("S",102));
```

定义 b 以使 $Ax = b$ 的实际解是全为 1 的向量。

```
b = vec(sum(A,dims=2));
```

设置容差和最大迭代次数

```
atol = 1e-12;
rtol = 1e-12;
maxit = 100;
```

使用 symmlq 根据请求的容差和迭代次数求解。

```
A = float.(A);
b = float.(b);
x, stats = symmlq(A, b; atol=atol, rtol=rtol, itmax=maxit);
```

```
stats =
SymmlqStats
 niter: 100
 solved: false
 residuals: []
 residuals (cg): []
 errors: []
 errors (cg): []
 ‖A‖F: 48.66326619769666
 κ2(A): 1.4415861938635752
 timer: 5.95ms
 status: maximum number of iterations exceeded
```

symmlq 未在请求的 100 次迭代内收敛至请求的容差 1e-12。

不限制迭代次数，再次进行求解：

```
x, stats = symmlq(A, b; atol=atol, rtol=rtol);
stats =
SymmlqStats
 niter: 201
 solved: true
 residuals: []
 residuals (cg): []
 errors: []
 errors (cg): []
 ‖A‖F: 68.97571761738863
 κ2(A): 1.4415861938635752
 timer: 11.65ms
 status: found approximate solution
```

结果显示，再经过 201 次迭代之后，收敛至 1e-12

(3). 提供初始估计值

检查向 symmlq 提供解的初始估计值的效果。

创建一个三对角稀疏矩阵。使用每行的总和作为 $Ax = b$ 右侧的向量，使 x 的预期解是由 1 组成的向量

```
n = 900;
e = ones(n);
A = spdiagm(-1=>ones(n-1),0=>2*e,1=>ones(n-1));
b = vec(sum(A,dims=2));
```

使用 symmlq 求解 $Ax = b$ 两次：一次是使用默认的初始估计值，一次是使用解的良好初始估计值。对两次求解均使用 200 次迭代和默认容差。将第二种求解中的初始估计值指定为所有元素都等于 0.99 的向量。。

```
maxit = 200;
x1,stats1 = symmlq(A, b; atol=1e-6, rtol=1e-6, itmax=maxit);
```

```
stats1 =
SymmlqStats
 niter: 3
 solved: true
 residuals: []
 residuals (cg): []
 errors: []
 errors (cg): []
 ‖A‖F: 6.133519887623151
 κ₂(A): 1.779437200555305
 timer: 27.60μs
 status: found approximate solution
```

```
x0 = 0.99 .* e;
x2, stats2 = symmlq(A, b, x0; atol=1e-6, rtol=1e-6, itmax=maxit);
```

```
stats2 =
SymmlqStats
 niter: 27
 solved: true
 residuals: []
 residuals (cg): []
 errors: []
 errors (cg): []
 ‖A‖F: 13.519435317809174
 κ₂(A): 3.5352588605005755
 timer: 118.30μs
 status: solution xᶜ good enough given atol and rtol
```

(4). 返回中间结果

可以通过设置关键字参数 verbose 或者 history 的值来返回中间结果。例如：使用 verbose 参数查看每次迭代的详细信息

```
x, stats = symmlq(A, b, x0; atol=1e-6, rtol=1e-6, itmax=maxit,
verbose=1);
```

```
SYMMLQ: system of size 900
   k      ‖r‖        β       cos       sin       ‖A‖      κ(A)    test1
timer
   0  1.2e+00  2.6e-02   1.0e+00   0.0e+00  0.0e+00  0.0e+00  ✗ ✗ ✗
✗  0.00s
   1  1.2e+00  7.5e-01   1.0e+00   6.6e-03  4.9e+00  1.0e+00  2.7e+00
0.00s
   2  1.4e-02  8.7e-01  -9.7e-01   2.6e-01  5.6e+00  1.4e+00  2.7e-02
0.00s
   3  4.8e-03  9.2e-01   9.2e-01   3.9e-01  6.1e+00  1.8e+00  8.7e-03
0.00s
   4  2.5e-03  9.5e-01  -8.8e-01   4.8e-01  6.6e+00  2.1e+00  4.1e-03
0.00s
   5  1.5e-03  9.6e-01   8.4e-01   5.5e-01  7.1e+00  2.3e+00  2.4e-03
0.00s
   6  1.0e-03  9.7e-01  -8.0e-01   6.0e-01  7.5e+00  2.5e+00  1.5e-03
0.00s
   7  7.4e-04  9.8e-01   7.7e-01   6.4e-01  7.9e+00  2.6e+00  1.0e-03
0.00s
   8  5.7e-04  9.8e-01  -7.4e-01   6.7e-01  8.3e+00  2.8e+00  7.6e-04
0.00s
   9  4.5e-04  9.9e-01   7.1e-01   7.0e-01  8.6e+00  2.9e+00  5.7e-04
0.00s
  10  3.6e-04  9.9e-01  -6.9e-01   7.3e-01  9.0e+00  2.9e+00  4.5e-04
0.00s
  11  3.0e-04  9.9e-01   6.7e-01   7.5e-01  9.3e+00  3.0e+00  3.6e-04
0.00s
  12  2.5e-04  9.9e-01  -6.5e-01   7.6e-01  9.6e+00  3.1e+00  2.9e-04
0.00s
  13  2.1e-04  9.9e-01   6.3e-01   7.8e-01  9.9e+00  3.1e+00  2.4e-04
0.00s
  14  1.8e-04  9.9e-01  -6.1e-01   7.9e-01  1.0e+01  3.2e+00  2.0e-04
0.00s
  15  1.6e-04  9.9e-01   6.0e-01   8.0e-01  1.1e+01  3.2e+00  1.7e-04
0.00s
  16  1.4e-04  1.0e+00  -5.8e-01   8.1e-01  1.1e+01  3.3e+00  1.5e-04
0.00s
  17  1.3e-04  1.0e+00   5.7e-01   8.2e-01  1.1e+01  3.3e+00  1.3e-04
0.00s
  18  1.1e-04  1.0e+00  -5.6e-01   8.3e-01  1.1e+01  3.3e+00  1.1e-04
```

```
0.00s
   19  1.0e-04  1.0e+00   5.4e-01   8.4e-01  1.2e+01  3.4e+00  9.6e-05
0.00s
   20  9.1e-05  1.0e+00  -5.3e-01   8.5e-01  1.2e+01  3.4e+00  8.5e-05
0.00s
   21  8.2e-05  1.0e+00   5.2e-01   8.5e-01  1.2e+01  3.4e+00  7.5e-05
0.00s
   22  7.5e-05  1.0e+00  -5.1e-01   8.6e-01  1.2e+01  3.4e+00  6.7e-05
0.00s
   23  6.9e-05  1.0e+00   5.0e-01   8.6e-01  1.3e+01  3.5e+00  6.1e-05
0.00s
   24  6.3e-05  1.0e+00  -4.9e-01   8.7e-01  1.3e+01  3.5e+00  5.5e-05
0.00s
   25  5.8e-05  1.0e+00   4.9e-01   8.7e-01  1.3e+01  3.5e+00  5.0e-05
0.00s
   26  5.4e-05  1.0e+00  -4.8e-01   8.8e-01  1.3e+01  3.5e+00  4.5e-05
0.00s
   27  5.0e-05  1.0e+00   4.7e-01   8.8e-01  1.4e+01  3.5e+00  4.1e-05
0.00s
```

使用 history 参数查看残差。

```
x, stats = symmlq(A, b, x0; atol=1e-6, rtol=1e-6, itmax=200,
history=true);
stats.residuals
```

```
stats.residuals =
28-element Vector{Float64}:
 1.1994165248152955
 1.1994165248152955
 0.013507007784122404
 0.004791833885385078
 0.0024639936033118023
 0.0015103423231520414
 0.0010254365899847006
 0.0007441671294114127
 0.0005659035098004135
 0.00044550797098223574
 0.00036021460053363094
 0.0002974965993141578
 0.00024998204998004256
 0.00021309451327023173
 0.00018386664236290747
 0.00016030368223383677
 0.00014102322699645356
```

```
0.0001250419733313508
0.0001116445722573539 7
0.0001003002720724299 3
9.060839598284451e-5
8.226176998360386e-5
7.502163754867763e-5
6.870010719968239e-5
6.314764934650617e-5
5.824404494446906e-5
5.389173630440551e-5
5.001087686319369e-5
```

3.2.7.5.13 gmres

求解线性方程组-广义最小残差法

A. 语法

```
x, stats = gmres(A, b; kwargs...)
x, stats = gmres(A, b, x0; kwargs...)
```

B. 说明

x = gmres(A, b; kwargs...) 尝试使用广义最小残差法求解关于 x 的线性系统 A*x = b。

x = gmres(A, b, x0; kwargs...)。

r 可以通过计算恢复 $E^{-1}(b - Ax)$。

C. 示例

(1). 线性系统的迭代解

使用采用默认设置的 gmres 求解系数矩阵为方阵的线性系统，然后在求解过程中调整使用的容差和迭代次数。

创建密度为 50% 的随机稀疏矩阵 A。另为 $Ax = b$ 的右侧创建随机向量.

```
rng = MT19937ar(5489);
A = sprand(rng,400,400,0.5) + 12*speye(400);
b = rand(rng,400);
```

使用 gmres 求解 。输出显示包括相对残差 $\frac{\left | b-Ax \right | }{\left | b \right | } $ 的值

```
x, stats = gmres(A, b; atol=1e-6, rtol=1e-6);
```

```
stats =
SimpleStats
 niter: 20
 solved: true
 inconsistent: false
 residuals: []
 Aresiduals: []
 κ₂(A): []
 timer: 771.80μs
 status: solution good enough given atol and rtol
```

(2). 使用 gmres 预条件子而不重启

检查使用预条件子矩阵和非重启 gmres 求解线性系统的效果。

加载矩阵 A，它是一个非对称的 479×479 实稀疏矩阵。

```
filename = pkgdir(TyMath) * "/examples/Sparse Matrices/Iterative Methods and Preconditioners/bicgstab_tests/bicgstab_data1.jl";
include(filename);
```

定义 b 以使 $Ax = b$ 的实际解是全为 1 的向量。

```
b = vec(sum(A,dims=2));
```

设置容差和最大迭代次数

```
atol = 1e-12;
rtol = 1e-12;
maxit = 20;
```

使用 gmres 根据请求的容差和迭代次数求解。

```
x, stats = gmres(A, b; atol=atol, rtol=rtol, itmax=maxit);
```

```
stats = SimpleStats
 niter: 20
 solved: false
 inconsistent: false
 residuals: []
 Aresiduals: []
 κ₂(A): []
 timer: 196.70μs
 status: maximum number of iterations exceeded
```

gmres 未在请求的 20 次迭代内收敛至请求的容差 1e-12。

3.2.7.5.14 tfqmr

求解线性方程组-无转置拟最小残差法

A. 语法

```
x, flag, relres, iter, resvec = tfqmr(A, b)
_ = tfqmr(A, b, tol)
_ = tfqmr(A, b, tol, maxit)
_ = tfqmr(A, b, tol, maxit, M)
_ = tfqmr(A, b, tol, maxit, M1, M2)
_ = tfqmr(A, b, tol, maxit, M1, M2, x0)
_ = tfqmr(_, verbose=true)
```

B. 说明

x, flag, relres, iter, resvec = tfqmr(A, b) 尝试使用无转置拟最小残差法求解关于 x 的线性方程组 A*x = b。其中 flag 为一个指示算法是否成功收敛的标志。当 flag = 0 时，收敛成功。relres 为相对残差 norm(b-A*x)/norm(b)。如果 flag 为 0，则 relres <= tol。iter 为计算出 x 时的迭代次数。resvec 为每次迭代中的残差范数向量包括第一个残差 norm(b-A*x0)）。您总可以使用 [] 返回部分输出，例如 x = tfqmr(A,b)[1] 。示例

_ = tfqmr(A,b,tol) 指定该方法的容差。默认容差是 1e-6。示例

_ = tfqmr(A,b,tol,maxit) 指定要使用的最大迭代次数。示例

_ = tfqmr(A,b,tol,maxit,M) 指定预条件子矩阵 M 并通过有效求解关于 y 的方程组 来计算 x，其中 。使用预条件子矩阵可以改善问题的数值属性和计算的效率。

_ = tfqmr(A,b,tol,maxit,M1,M2) 指定预条件子矩阵 M 的因子，使得 M = M1*M2。

_ = tfqmr(A,b,tol,maxit,M1,M2,x0) 指定解向量 x 的初始估计值。默认值为由零组成的向量。示例

_ = tfqmr(_, verbose = true) 在上述用法的基础上增加了 verbose 变量。如果该变量值为 true（默认值），则 tfqmr 会显示收敛信息：如果尝试成功，tfqmr 会显示一条消息来确认收敛。如果 tfqmr 无法在达到最大迭代次数后收敛或出于任何原因暂停，则会显示一条包含相对残差 norm(b-A*x)/norm(b) 以及该方法停止时的迭代次数的诊断消息。否则不会显示。示例。

C. 示例

(1). 线性系统的迭代解

使用采用默认设置的 tfqmr 求解系数矩阵为方阵的线性系统，然后在求解过程中调整使用的容差和迭代次数。

创建密度为 50% 的随机稀疏矩阵 A。另为 $Ax = b$ 的右侧创建随机向量.

```
rng = MersenneTwister(1234)
A = sprand(rng, 400, 400, 0.5)
A = A'*A
b = rand(rng, 400)
```

使用 tfqmr 求解 $Ax = b$。输出显示包括相对残差 $\frac{\|b - Ax\|}{\|b\|}$ 的值。

```
x = tfqmr(A, b)[1];
```

```
tfqmr stopped at iteration 40 without converging to the desired tolerance 1.0e-6 because the maximum number of iterations was reached.The iterate returned (number 12) has relative residual 0.25994835328907334.
```

默认情况下，tfqmr 使用 40 次迭代和容差 1e-6，对于此矩阵，算法无法在 40 次迭代后收敛。由于残差仍然很大，这说明需要更多的迭代（或预条件子矩阵）。您也可以使用更大的容差，使算法更容易收敛。

使用容差 1e-4 和 100 次迭代再次求解方程组。

```
x = tfqmr(A,b,1e-4,100)[1];
```

```
tfqmr stopped at iteration 200 without converging to the desired tolerance 0.0001 because the maximum number of iterations was reached.The iterate returned (number 12) has relative residual 0.25994835328907334.
```

即使使用更宽松的容差和更多迭代，残差也并未改进。当迭代算法以这种方式停滞时，显然需要预条件子矩阵。

(2). 提供初始估计值

检查向 tfqmr 提供解的初始估计值的效果。

创建一个三对角稀疏矩阵。使用每行的总和作为 $Ax = b$ 右侧的向量，使 x 的预期解是由 1 组成的向量

```
n = 900
e = vec(ones(n,1))
A = spdiagm(n,n,-1=>e[1:end-1],0=>2 .*e,1=>e[1:end-1])
b = sum(A,dims=2)
```

使用 tfqmr 求解 $Ax = b$ 两次：一次是使用默认的初始估计值，一次是使用解的良好初始估计值。对两次求解均使用 200 次迭代和默认容差。将第二种求解中的初始估计值指定为所有元素都等于 0.99 的向量。

```
maxit = 200
x1 = tfqmr(A,b,1e-6,maxit)[1];
```

```
tfqmr converged at iteration 19 to a solution with relative residual 9.565651220398758e-7.
```

```
x0 = 0.99 .* e
x2 = tfqmr(A,b,1e-6,maxit,[],[],x0)[1];
```

```
tfqmr converged at iteration 4 to a solution with relative residual 7.926006665268268e-7.
```

在这种情况下，提供初始估计值可以使 tfqmr 更快地收敛。

3.2.8 图和网络算法

有向图和无向图、网络分析

表示网络连接的图形，该类图形广泛应用于各种物理、生物和信息系统。您可以使用图形表示大脑中的神经元、航空公司的飞行模式及更多领域的相关内容。图形的结构由“节点”和“边”组成。每个节点表示一个实体，每个边表示两个节点之间的连接。有关详细信息，请参阅有向图和无向图。

3.2.8.1 构造

Graph	具有无向边的图
DiGraph	具备有向边的图
Graph	具有无向边的图

3.2.8.1.1 Graph

具备无向边的图

A. 语法

```
G = Graph()
G = Graph(A)
```

```
G = Graph(A,nodenames)
G = Graph(A,nodenames,nodeprops)
G = Graph(A, ___; type="")
G = Graph(A, ___; omitselfloops=false)
G = Graph(s,t)
G = Graph(s,t,weights)
G = Graph(s,t,weights,nodenames)
G = Graph(s,t,weights,nodenames,nodeprops)
G = Graph(s,t,weights,nodenames,nodeprops,edgeprops)
G = Graph(s,t,weights,num)
G = Graph(s,t,___;omitselfloops=false)
```

B. 说明

Graph 对象表示无向图，无向图具有连接相应节点的无向边。创建 Graph 对象后，您可以通过使用对象函数针对对象执行查询，了解有关图的详细信息。例如，您可以添加或删除节点或边、确定两个节点之间的最短路径，或定位特定的节点或边。

```
G = Graph([1,1], [2,3])
e = G.Edges
addedge!(G,2,3)
addnode!(G,4)
```

G = Graph 创建空无向图对象 G，该对象没有节点或边。示例

G = Graph(A) 使用对称邻接方阵 A 创建一个无向图。示例

对于逻辑邻接矩阵，图没有边权重。

对于非逻辑邻接矩阵，图有边权重。A 中的每个非零项的位置指定图的一条边，边的权重等于该项的值。例如，如果 A[2,1] = 10，则 G 包含一条从节点 2 到节点 1 的边，该边的权重为 10。

G = Graph(A,nodenames) 还指定节点名称。nodenames 中的元素数量必须等于 size(A,1)。 示例

G = Graph(A,nodenames,nodeprops) 使用 nodenames 指定节点名称并使用 nodeprops 指定节点属性。

G = Graph(A,___;type = "") 指定在构造图时要使用的邻接矩阵的一个三角矩阵。必须指定 A，而 nodenames 和 nodeProps 是可选的。要仅使用 A 的上或下三角矩阵构造图，type 可以是 "upper" 或 "lower"。示例

G = Graph(A,___;omitselfloops = true) 忽略 A 的对角线元素，并返回没有任何自环的图。您可以使用上述语法中的任何输入参数组合。示例

G = Graph(s,t) 以节点对组形式指定图边 (s,t)。s 和 t 可以指定节点索引或节点名称。Graph 首先按源节点、然后按目标节点对 G 中的边进行排序。示例

G = Graph(s,t,weights) 还使用数组 weights 指定边的权重。示例

G = Graph(s,t,weights,nodenames) 还使用字符串数组 nodenames 指定节点名称。s 和 t 不能包含 nodenames 中没有的节点名称。示例

G = Graph(s,t,weights,nodenames,nodeprops) 使用 nodenames 指定节点名称并使用 nodeprops 指定节点属性。

G = Graph(s,t,weights,nodenames,nodeprops,edgeprops) 使用 edgeprops 指定边属性。edgeprops 长度必须与边数相同，每个元素对应 s 和 t 中的每一对元素。若不想输入前几项参数可输入指定类型的空数组。

G = Graph(s,t,weights,num) 使用数值标量 num 指定图中的节点数。示例

G = Graph(s,t,___;omitselfloops = true) 不会将任何自环添加到图。即，将忽略满足 s(k) == t(k) 的任何 k。您可以使用上述语法中的任何输入参数组合。

C. 示例

(1). 创建和修改图对象

创建具有三个节点和两条边的 Graph 对象。一条边位于节点 1 和节点 2 之间，另一条位于节点 1 和节点 3 之间。

```
G = Graph([1,1],[2,3])
```

```
G =
Graph with properties:
Edges:2×1
Nodes:3×0
```

查看图的边表。

```
G.Edges
```

```
ans =
2×1
EndNodes
1      2
1      3
```

向图中添加节点名称，然后查看新节点和边。

```
G.Nodes.nodeNames = ["A","B","C"];
G.Nodes
```

```
ans =
3×1
Name
```

```
A
B
C
```

```
G.Edges
```

```
ans =
3×1
EndNodes
1      2
1      3
```

您可以在 Nodes 和 Edges 中修改其他变量，以描述图节点或边的属性。但是，您无法通过修改这些结构体直接更改图中节点或边的数量。此时，应使用 addedge、rmedge、addnode 或 rmnode 函数修改图中节点或边的数量。

```
G = addedge(G,2,3)
```

```
G =
Graph with properties:
Edges:3×1
Nodes:3×1
```

```
G.Edges
```

```
ans =
4×1
EndNodes
1      2
1      3
2      3
```

(2). 构造邻接矩阵图

创建一个对称邻接矩阵 A，该矩阵用于创建 4 阶完整图。使用逻辑邻接矩阵创建不带权重的图。

```
A = ones(Int, 4, 4) - diagm([1, 1, 1, 1])
```

```
A = 4×4 Matrix{Int64}:
 0  1  1  1
 1  0  1  1
 1  1  0  1
 1  1  1  0
```

```
G = Graph(A .!= 0)
```

```
G =
Graph with properties:
Edges:6×1
Nodes:4×0
```

查看图的边列表。

```
G.Edges
```

```
ans =
6×1
EndNodes
1      2
1      3
1      4
2      3
2      4
3      4
```

(3). 使用节点名称构造邻接矩阵图

创建一个上三角邻接矩阵。

```
A = triu(magic(4))
```

```
A = 4×4 Matrix{Int64}:
 16   2   3  13
  0  11  10   8
  0   0   6  12
  0   0   0  10
```

使用邻接矩阵创建具有命名节点的图。指定 omitselfloops=true 以忽略 A 对角线上的元素，并将 type 指定为 "upper" 以指示 A 为上三角矩阵。

```
names = ["alpha", "beta", "gamma", "delta"];
G = Graph(A, names; type="upper",omitselfloops=true)
```

```
G =
Graph with properties:
Edges:6×2
Nodes:4×1
```

查看边和节点信息。

```
G.Edges
```

```
ans =
6×2
EndNodes       Weights
1      2      2
1      3      3
1      4      13
2      3      10
2      4      8
3      4      12
```

```
G.Nodes
```

```
ans =
4×1
Name
alpha
beta
gamma
Delta
```

(4). 构造边列表图

使用每条边的端节点列表创建一个立方体图。

```
s = [1, 1, 1, 2, 2, 3, 3, 4, 5, 5, 6, 7];
t = [2, 4, 8, 3, 7, 4, 6, 5, 6, 8, 7, 8];
G = Graph(s, t)
```

```
G =
Graph with properties:
Edges:12×1
Nodes:8×0
```

(5). 使用节点名称和边权重构造边列表图

使用每条边的端节点列表创建并绘制一个立方体图。将节点名称和边权重指定为单独的输入。

```
s = [1, 1, 1, 2, 2, 3, 3, 4, 5, 5, 6, 7];
t = [2, 4, 8, 3, 7, 4, 6, 5, 6, 8, 7, 8];
weights = [10, 10, 1, 10, 1, 10, 1, 1, 12, 12, 12, 12];
names = ["A", "B", "C", "D", "E", "F", "G", "H"];
G = Graph(s, t, weights, names)
```

```
G =
Graph with properties:
Edges:12×2
Nodes:8×1
```

(6). 具有额外节点的边列表构造

使用每条边的端节点的列表创建加权图。指定图应包含共 10 个节点。

```
s = [1, 1, 1, 1, 1];
t = [2, 3, 4, 5, 6];
weights = [5, 5, 5, 6, 9];
G = Graph(s, t, weights, 10)
```

```
G =
Graph with properties:
Edges:5×2
Nodes:10×0
```

(7). 将节点和边添加到空图

创建一个空图对象 G。

```
G = Graph();
```

将三个节点和三条边添加到该图。s 和 t 中的相应项定义各条边的源节点和目标节点。addedge 自动将适当的节点添加到图（如果这些节点尚不存在）

```
s = [1, 2, 1];
t = [2, 3, 3];
G = addedge(G, s, t)
```

```
G =
Graph with properties:
Edges:3×1
Nodes:3×0
```

查看边列表。每行描述图中的一条边。

```
G.Edges
```

```
ans =
3×1
EndNodes
1      2
1      3
2      3
```

要获得最佳性能，请通过调用一次 Graph 来一次性构造多个图。对于大图来说，以循环方式添加节点或边会比较慢

3.2.8.1.2 DiGraph

具备有向边的图

A. 语法

```
G = DiGraph()
G = DiGraph(A)
G = DiGraph(A,nodenames)
G = DiGraph(A,nodenames,nodeprops)
G = DiGraph(A,___;omitselfloops=false)
G = DiGraph(s,t)
G = DiGraph(s,t,weights)
G = DiGraph(s,t,weights,nodenames)
G = DiGraph(s,t,weights,nodenames,nodeprops)
G = DiGraph(s,t,weights,nodenames,nodeprops,edgeprops)
G = DiGraph(s,t,weights,num)
```

```
G = DiGraph(s,t,___;omitselfloops=false)
```

B. 说明

DiGraph 对象表示有向图，有向图具有连接相应节点的有向边。创建 DiGraph 对象后，您可以通过使用对象函数针对对象执行查询，了解有关图的详细信息。例如，您可以添加或删除节点或边、确定两个节点之间的最短路径，或定位特定的节点或边。

```
G = DiGraph([1,1], [2,3])
e = G.Edges
```

G = DiGraph 创建空有向图对象 G，该对象没有节点或边。示例

G = DiGraph(A) 使用邻接方阵 A 创建一个有向图。示例

对于逻辑邻接矩阵，图没有边权重。

对于非逻辑邻接矩阵，图有边权重。A 中的每个非零项的位置指定图的一条边，边的权重等于该项的值。例如，如果 A[2,1] = 10，则 G 包含一条从节点 2 到节点 1 的边，该边的权重为 10。

G = DiGraph(A,nodenames) 还指定节点名称。nodenames 中的元素数量必须等于 size(A,1)。 示例

G = DiGraph(A,nodenames,nodeprops) 使用 nodenames 指定节点名称并使用 nodeprops 指定节点属性。

G = DiGraph(A,___;omitselfloops = true) 忽略 A 的对角线元素，并返回没有任何自环的图。您可以使用上述语法中的任何输入参数组合。示例

G = DiGraph(s,t) 以对组形式指定有向图边 (s,t) 来表示源节点和目标节点。s 和 t 可以指定节点索引或节点名称。DiGraph 首先按源节点、然后按目标节点对 G 中的边进行排序。示例

G = DiGraph(s,t,weights) 还使用数组 weights 指定边的权重。示例

G = DiGraph(s,t,weights,nodenames) 还使用字符串数组 nodenames 指定节点名称。s 和 t 不能包含 nodenames 中没有的节点名称。示例

G = DiGraph(s,t,weights,nodenames,nodeprops) 使用 nodenames 指定节点名称并使用 nodeprops 指定节点属性。

G = DiGraph(s,t,weights,nodenames,nodeprops,edgeprops) 使用 edgeprops 指定边属性。edgeprops 长度必须与边数相同，每个元素对应 s 和 t 中的每一对元素。若不想输入前几项参数可输入指定类型的空数组。

G = DiGraph(s,t,weights,num) 使用数值标量 num 指定图中的节点数。示例

G = DiGraph(s,t,___;omitselfloops = true) 不会将任何自环添加到图。即，将忽略满足 s(k) == t(k) 的任何 k。您可以使用上述语法中的任何输入参数组合。

C. 示例

(1). 创建和修改有向图对象

创建具有三个节点和三条边的 DiGraph 对象。一条边是从节点 1 到节点 2，另一条是从节点 1 到节点 3，第三条是从节点 2 到节点 1。

```
G = DiGraph([1,1,2],[2,3,1])
G =
DiGraph with properties:
Edges:3×1
Nodes:3×0
```

查看图的边。对于有向图，第一列指示每条边是源节点，第二列指示目标节点。

另一条是从节点 1 到节点 3，第三条是从节点 2 到节点 1。

```
G = DiGraph([1,1,2],[2,3,1])
G =
DiGraph with properties:
Edges:3×1
Nodes:3×0
```

查看图的边。对于有向图，第一列指示每条边是源节点，第二列指示目标节点

```
G.Edges
ans =
3×1
Name
A
B
C
G.Edges
ans =
3×1
EndNodes
1      2
1      3
2      1
```

您可以在 Nodes 和 Edges 中修改其他变量，以描述图节点或边的属性。

但是，您无法通过修改这些结构体直接更改图中节点或边的数量。此时，应使用 addedge、rmedge、addnode 或 rmnode 函数修改图中节点或边的数量。

```
G = addedge(G,2,3)
```

```
G =
DiGraph with properties:
Edges:4×1
Nodes:3×1
```

```
G.Edges
```

```
ans =
4×1
EndNodes
1      2
1      3
2      1
2      3
```

(2). 构造邻接矩阵图

创建一个对称邻接矩阵 A，该矩阵用于创建 4 阶完整图。使用逻辑邻接矩阵创建不带权重的图。

```
A = ones(Int, 4, 4) - diagm([1, 1, 1, 1])
```

```
A = 4×4 Matrix{Int64}:
 0  1  1  1
 1  0  1  1
 1  1  0  1
 1  1  1  0
```

```
G = DiGraph(A .!= 0)
```

```
G =
Graph with properties:
Edges:12×1
Nodes:4×0
```

查看图的边列表。

```
G.Edges
```

```
ans =
12×1
EndNodes
1      2
1      3
```

```
1       4
2       1
2       3
2       4
3       1
3       2
3       4
4       1
4       2
4       3
```

(3). 使用节点名称构造邻接矩阵图

创建对称邻接矩阵。

```
A = magic(4);
A[A .> 10] .= 0;
A
```

```
ans = 4×4 Matrix{Int64}:
 0  2   3  0
 5  0  10  8
 9  7   6  0
 4  0   0  1
```

使用邻接矩阵创建具有命名节点的图。指定 omitselfloops=true 以忽略 A 对角线上的元素。

```
names = ["alpha", "beta", "gamma", "delta"];
G = DiGraph(A, names; omitselfloops=true)
```

```
G =
DiGraph with properties:
Edges:8×2
Nodes:4×1
```

查看边和节点信息。

```
G.Edges
```

```
ans =
8×2
EndNodes       Weights
1       2       2
1       3       3
2       1       5
2       3       10
2       4       8
3       1       9
3       2       7
```

```
4      1      4
```

```
G.Nodes
```

```
ans =
4×1
Name
alpha
beta
gamma
Delta
```

(4). 构造边列表图

使用每条边的端节点列表创建一个立方体图。

```
s = [1, 1, 1, 2, 2, 3, 3, 4, 5, 5, 6, 7];
t = [2, 4, 8, 3, 7, 4, 6, 5, 6, 8, 7, 8];
G = DiGraph(s, t)
```

```
G =
Graph with properties:
Edges:12×1
Nodes:8×0
```

(5). 使用节点名称和边权重构造边列表图

使用每条边的端节点列表创建并绘制一个立方体图。将节点名称和边权重指定为单独的输入。

```
s = [1, 1, 1, 2, 2, 3, 3, 4, 5, 5, 6, 7];
t = [2, 4, 8, 3, 7, 4, 6, 5, 6, 8, 7, 8];
weights = [10, 10, 1, 10, 1, 10, 1, 1, 12, 12, 12, 12];
names = ["A", "B", "C", "D", "E", "F", "G", "H"];
G = DiGraph(s, t, weights, names)
```

```
G =
Graph with properties:
Edges:12×2
Nodes:8×1
```

(6). 具有额外节点的边列表构造

使用每条边的端节点的列表创建加权图。指定图应包含共 10 个节点。

```
s = [1, 1, 1, 1, 1];
t = [2, 3, 4, 5, 6];
weights = [5, 5, 5, 6, 9];
G = DiGraph(s, t, weights, 10)
```

```
G =
Graph with properties:
Edges:5×2
```

```
Nodes:10×0
```

(7). 将节点和边添加到空图

创建一个空图对象 G。

```
G = DiGraph();
```

将三个节点和三条边添加到该图。s 和 t 中的相应项定义各条边的源节点和目标节点。addedge 自动将适当的节点添加到图（如果这些节点尚不存在）

```
s = [1, 2, 1];
t = [2, 3, 3];
G = addedge(G, s, t)
```

```
G =
Graph with properties:
Edges:3×1
Nodes:3×0
```

查看边列表。每行描述图中的一条边。

```
G.Edges
```

```
ans =
3×1
EndNodes
1       2
1       3
2       3
```

要获得最佳性能，请通过调用一次 DiGraph 来一次性构造多个图。对于大图来说，以循环方式添加节点或边会比较慢。

3.2.8.2 修改节点和边

函数名	简介
addnode	将新节点添加到图
rmnode	从图中删除节点
addedge	向图添加新边
rmedge	从图中删除边

flipedge	反转边的方向
numnodes	图中节点的数量
numedges	图中边的数量
findnode	定位图中的节点
findedge	定位图中的边
edgecount	两个节点之间的边数
subgraph	提取子图

3.2.8.2.1 addnode

将新节点添加到图

A. 语法

```
H = addnode(G, numNodes)
H = addnode(G, nodeIDs)
H = addnode(G, nodeIDs, nodeProps)
addnode!(G, _)
```

B. 说明

H = addnode(G, numNodes) 将一定数目的新节点添加到 G，节点数量等于 numNodes。如果 G 包含具有名称的节点，则会为新节点分配序列名，用于指示它们在 G.Nodes.Name 中的行位置。例如，'Node5' 所在的位置是 G.Nodes.nodeNames[5]。示例

H = addnode(G,nodeIDs) 将 nodeIDs 指定的节点添加到图 G。nodeIDs 中的节点名称不能是 G 中已存在的节点。示例

H = addnode(G,nodeIDs, nodeProps) 使用 nodeIDs 指定的节点以及 nodeProps 中指定的节点属性将新节点添加到 G。nodeProps 的长度必须与 nodeIDs 的长度相同。示例

addnode!(G, _) 不会返回新的图，而是会在原有图 G 的基础上插入节点，并返回 G，您可以使用上述语法的任意输入。

C. 示例

(1). 将节点添加到图

将两个节点添加到不包含节点名称的图中。

```
G = Graph([1,2,3],[2,3,4])
```

```
 G =
Graph with properties:
Edges:3×1
Nodes:4×0
```

```
addnode!(G,2)
```

```
ans =
Graph with properties:
Edges:3×1
Nodes:6×0
```

```
G.Nodes.nodeNames=["A","B","C","D","E","F"]
```

```
 6-element Vector{String}:
 "A"
 "B"
 "C"
 "D"
 "E"
 "F"
```

```
addnode!(G,5);
G.Nodes
```

```
ans =
11 × 1
Name
A
B
C
D
E
F
Node7
Node8
Node9
Node10
```

```
Node11
```

(2). 将命名节点添加到图

创建一个具有命名节点的有向图，然后将两个命名节点添加到图。

```
G = DiGraph(["A","B","C"],["D","C","D"])
```

```
G =
DiGraph with properties:
Edges:3×1
Nodes:4×1
```

```
addnode!(G,["E","F"])
```

```
ans =
DiGraph with properties:
Edges:3×1
Nodes:6×1
```

如果图还没有节点名称，则将命名节点添加到图会自动为其他节点生成名称。

创建不具有节点名称的有向图，然后将两个命名节点添加到图。

```
H = DiGraph([1,2,3],[4,3,4])
```

```
H =
DiGraph with properties:
Edges:3×1
Nodes:4×0
```

```
addnode!(H,["E","F"]);
H.Nodes
```

```
ans =
6×1
Name
Node1
Node2
Node3
Node4
E
F
```

(3). 将具有属性的节点添加到图

创建一个图，其节点名称表示机场。

```
G = Graph(["JFK","LAX"], ["LAX","DEN"])
```

```
Graph with properties:
Edges:2×1
Nodes:3×1
```

设定各节点的属性，规定属性表示该机场有无 WIFI。

```
G.Nodes.props = [false,true,true];
G.Nodes
```

```
ans =
3×2
Name    NodesProp
JFK     false
LAX     true
DEN     true
```

使用 addnode 将两个带有属性的节点串联到 G.Nodes。

```
addnode!(G,["ATL","ANC"],[false,true])
```

```
 ans =
Graph with properties:
Edges:2×1
Nodes:5×2
```

3.2.8.2.2 rmnode

从图中删除节点

A. 语法

```
H = rmnode(G,nodeIDs)
rmnode!(G, nodeIDs)
```

B. 说明

H = rmnode(G, nodeID) 从图 G 中删除 nodeIDs 指定的节点。与 nodeIDs 中的节点相连的任何边也将删除。rmnode 会更新 H 中节点的编号，这样，如果您删除了节点 k，则节点 1:k-1 在 H 中具有相同的节点编号，并且 G 中的节点 k+1:numnodes(G) 成为 H 中的 k:numnodes(H)。示例

rmnode!(G, nodeID) 直接在图 G 中原地删除 nodeIDs 指定的节点。示例。

C. 示例

(1). 确定命名节点的索引

创建一个图。

```
s = [1, 1, 1, 2, 2, 3];
t = [2, 3, 4, 3, 4, 4];
```

```
G = Graph(s, t);
```

从图中删除节点 1。新图中的节点会自动重新编号。

```
H = rmnode(G,1);
```

(2). 从图中删除若干命名节点

创建一个具有命名节点的图。

```
s = [1, 1, 1, 1, 2, 2, 3, 3, 3, 5, 5];
t = [2, 3, 4, 6, 1, 5, 4, 5, 6, 4, 6];
names = ["New York", "Los Angeles", "Washington D.C.",
"Pittsburgh", "Denver", "Austin"];
G = DiGraph(s, t, Float64[], names);
```

从该图中删除节点 "New York" 和 "Pittsburgh"。

```
rmnode!(G,["New York","Pittsburgh"]);
```

3.2.8.2.3 addedge

向图添加新边

A. 语法

```
H = addedge(G,s,t)
H = addedge(G,s,t,w)
H = addedge(G,s,t,w,edgeProps)
addedge!(G, _)
```

B. 说明

H = addedge(G,s,t) 在图 G 中的节点 s 和 t 之间添加一条边。如果 s 或 t 指定的节点在 G 中不存在，则还会添加该节点。新图 H 等价于 G，但包括新边和任何必需的新节点。示例

H = addedge(G,s,t,w) 还对 s 和 t 之间的边指定权重 w。示例

H = addedge(G,s,t,w,edgeProps) 还对 s 和 t 之间的边指定属性 edgeProps。示例

addedge!(G, _) 不会返回新的图，而是会在原有图 G 的基础上插入边，并返回 G，您可以使用上述语法的任意输入。

C. 示例

(1). 将边添加到图

将两条新边添加到现有图。由于其中一条新边引用不存在的节点，addedge 会自动将所需的第四个节点添加到图。

```
G = Graph([1,2],[2,3])
```

```
G =
Graph with properties:
Edges:2×1
Nodes:3×0
```

```
addedge!(G, [2, 1], [4, 3])
```

```
G =
Graph with properties:
Edges:4×1
Nodes:4×0
```

(2). 将加权边添加到图

创建具有加权边的有向图。

```
G = DiGraph(["A","B","C"], ["D","C","D"], [10, 20, 45])
```

```
G =
DiGraph with properties:
Edges:3×2
Nodes:4×1
```

将三条新加权边添加到图。addedge 还会自动将所需的任何新节点添加到图

```
G = addedge(G, ["A", "D", "E"], ["E", "B", "D"], [5, 30, 5])
```

```
G =
DiGraph with properties:
Edges:6×2
Nodes:5×1
```

(3). 将具有属性的边添加到图

创建一个图，该图的边具有权重和边属性。

```
s = [1, 2, 2, 2, 5, 5, 5];
t = [2, 3, 4, 5, 6, 7, 8];
edgeprop = ["on", "off", "off", "on", "on", "on", "off"];
weight = [10, 20, 20, 10, 10, 10, 20];
G = Graph(s, t, weight, String[], Float64[], edgeprop)
```

```
G =
Graph with properties:
Edges:7×3
Nodes:8×0
```

将两条同样具有权重和属性的新边添加到图 I。

```
addedge!(G, [5, 3], [9, 6], [10, 20], ["on", "off"])
```

```
G.Edges
```

```
G =
Graph with properties:
Edges:9×3
Nodes:9×0
```

3.2.8.2.4 rmedge

从图中删除边

A. 语法

```
H = rmedge(G,s,t)
H = rmedge(G,idx)
rmedge!(G, __)
```

B. 说明

H = rmedge(G, s,t) 从图 G 删除节点对组 s 和 t 指定的边。如果 s 和 t 指定了多条边，则将它们全部删除。示例

H = rmedge(G, idx) 使用边索引 idx 指定要删除的边。边索引是 G.Edges.endnodes 中的行号。示例

rmedge!(G,__) 直接在图 G 中原地删除对应边，您可以使用上述语法的任意输入。

C. 示例

(1). 删除具有指定端节点的边

创建一个图。

```
s = [1, 1, 1, 2, 2, 3, 3, 4, 5, 5, 6, 7];
t = [2, 4, 5, 3, 6, 4, 7, 8, 6, 8, 7, 8];
G = Graph(s, t);
```

从图中删除若干条边 。

```
rmedge!(G, [1,2,3,4], [5,6,7,8]);
```

(2). 删除具有指定索引的边

创建一个图并查看边列表。

```
s = ["BOS", "NYC", "NYC", "NYC", "LAX"];
```

```
t = ["NYC", "LAX", "DEN", "LAS", "DCA"];
G = DiGraph(s, t);
G.Edges
```

```
5×1
EndNodes
1      2
2      3
2      4
2      5
3      6
```

使用边索引删除节点 "NYC" 和 "DEN" 之间的边。

```
rmedge!(G,3);
G.Edges
```

```
ans =
4×1
EndNodes
1      2
2      3
2      5
3      6
```

(3). 删除自环

此示例说明如何删除图中的所有自环。自环是指使节点自身相连的边。

创建包含两个自环的图。

```
G = Graph([1,1,1,2],[1,2,3,2]);
```

使用 rmedge 删除图中的所有自环。尽管 G 只有两个自环，但此方法可以删除任何有向图或无向图中的所有自环。

```
rmedge!(G, 1:numnodes(G), 1:numnodes(G));
```

3.2.8.2.5 flipedge

反转边的方向

A. 语法

```
H = flipedge(G)
H = flipedge(G,s,t)
H = flipedge(G,idx)
flipedge!(G, __)
```

B. 说明

H = flipedge(G) 返回一个与 G 具有相同边但方向相反的有向图。H 包含与 G 相同的节点和边属性。示例

H = flipedge(G, s,t) 使用节点对组 s 和 t 反转部分边。示例

H = flipedge(G, idx) 使用边索引 idx 反转部分边。

flipedge!(G,__) 直接在图 G 中原地反转对应边，您可以使用上述语法的任意输入。

C. 示例

(1). 反转图中的所有边

创建一个有向图。然后反转图中所有边的方向。

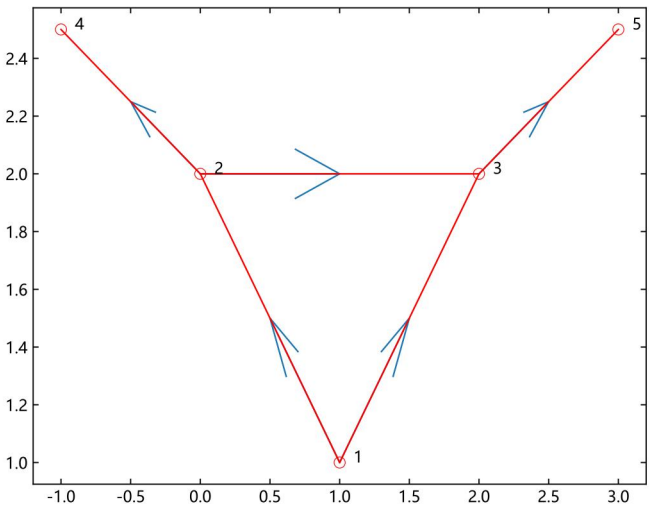
```
G = DiGraph([1, 1, 1, 1], [2, 3, 4, 5]);
```

```
flipedge!(G);
```

(2). 反转图的部分边

创建并绘制一个有向图。为绘图指定自定义 xy 节点坐标。

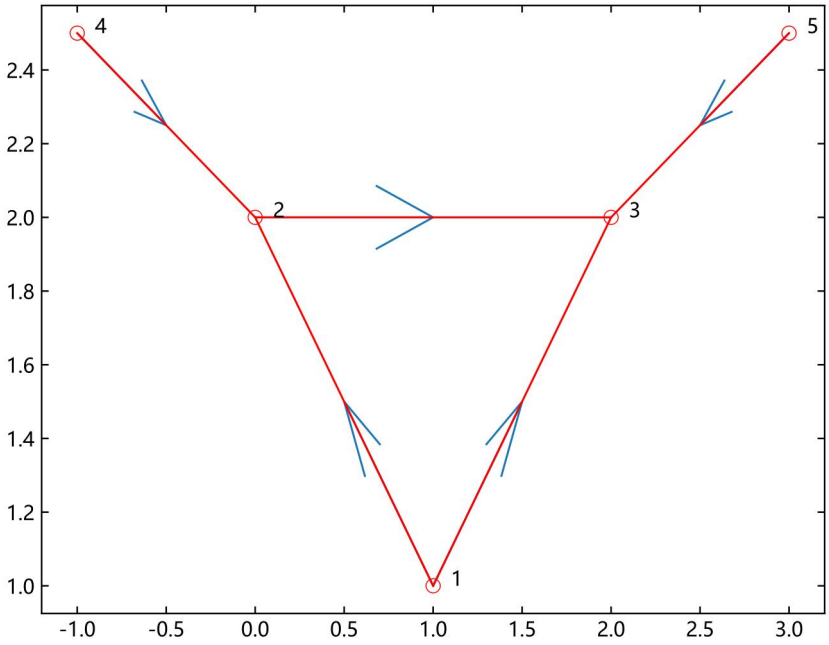
```
G = DiGraph([1, 1, 2, 2, 3], [2, 3, 3, 4, 5]);
x = [1, 0, 2, -1, 3];
y = [1, 2, 2, 2.5, 2.5];
plotG(G, x, y)
```



反转边 (2,4) 和 (3,5) 的方向，然后重新绘制图形。

```
G = DiGraph([1, 1, 2, 2, 3], [2, 3, 3, 4, 5]);
```

```
x = [1, 0, 2, -1, 3];
y = [1, 2, 2, 2.5, 2.5];
flipedge!(G,[2, 3],[4, 5])
plotG(G, x, y)
```



3.2.8.2.6 numnodes

定位图中的节点

A. 语法

```
N = numnodes(G)
```

B. 说明

N = numnodes(G) 返回图 G 中节点的数量。示例。

C. 示例

(1). 图节点数

使用 bucky 邻接矩阵创建一个有向图，然后确定该图包含多少个节点。

```
G = DiGraph(bucky())
```

```
G =
DiGraph with properties:
```

```
Edges:180×2
Nodes:60×0
```

```
N = numnodes(G)
```

```
N = 60
```

3.2.8.2.7 numedges

图中边的数量

A. 语法

```
N = numedges(G)
```

B. 说明

N = numedges(G) 返回图 G 中边的数量。示例。

C. 示例

(1). 图边的数量

使用 bucky 邻接矩阵创建一个有向图，然后确定该图包含多少条边。

```
G = DiGraph(bucky())
```

```
G =
DiGraph with properties:
Edges:180×2
Nodes:60×0
```

```
N = numedges(G)
```

```
N = 180
```

3.2.8.2.8 findnode

定位图中的节点

A. 语法

```
k = findnode(G,nodeID)
```

B. 说明

k = findnode(G, nodeID) 返回图 G 中名称或索引为 nodeID 的节点的数值

节点 ID k。如果节点不在图中，则数值节点 ID 为零。示例。

C. 示例

(1). 确定命名节点的索引

创建一个图，然后确定名为 "AB" 和 "BC" 的节点的数值节点索引。

```
s = ["AA", "AA", "AA", "AB", "AC", "BB"];
t = ["BA", "BB", "BC", "BA", "AB", "BC"];
G = Graph(s,t)
```

```
G =
Graph with properties:
Edges:6×1
Nodes:6×1
```

```
k = findnode(G, ["AB", "BC"])
```

```
k =2-element Vector{Int64}:
 5
 4
```

k 包含每个命名节点的 G.Nodes.nodeNames 的行索引。

3.2.8.2.9 findedge

定位图中的边

A. 语法

```
sOut,tOut = findedge(G)
sOut,tOut = findedge(G,s,t)
idxOut,m = findedge(G,idx)
```

B. 说明

sOut,tOut = findedge(G) 对于图 G 中的所有边，返回的源和目标节点 ID sOut 和 tOut。示例

sOut,tOut = findedge(G, idx) 查找 idx 指定的边的源和目标节点。示例

idxOut,m = findedge(G, s,t) 对于源和目标节点对组 s 和 t 指定的边，返回数值边索引 idxOut 。边索引对应于图的 G.Edges 表中的行 G.Edges.endnodes[idxOut,:]。如果 s 和 t 之间存在多条边，则返回它们的所有索引。边索引 0 表示边不在图中。向量 m 指示哪个节点对 (s,t) 与 idxOut 中的每个边索引关联。这在相同两个节点之间存在多条边时很有用。示例。

C. 示例

(1). 定位具有指定端节点的边

定位具有指定端节点的边。

```
s = [1, 1, 2, 2, 2, 3, 3, 3];
t = [2, 3, 3, 4, 5, 6, 7, 5];
G = Graph(s, t)
```

```
G =
Graph with properties:
Edges:8×1
Nodes:7×0
```

```
idxOut, m = findedge(G, [1, 3], [2, 5])
```

```
idxOut = 2-element Vector{Int64}:
 1
 6
m = 2-element Vector{Int64}:
 1
 2
```

idxOut 包含每条指定边的 G.Edges.EndNodes 的行索引。

m 指示哪个节点对 (s,t) 与 idxOut 中的每个边索引关联。

(2). 所有图边的端节点

创建一个图，然后确定图中所有边的端节点。

```
s = ["a", "a", "b", "b", "c", "c"];
t = ["b", "c", "d", "e", "f", "g"];
G = Graph(s, t);
G.Edges
```

```
ans =
6×1
EndNodes
1      2
1      3
2      4
2      5
3      6
3      7
```

```
sOut, tOut = findedge(G)
```

```
sOut = 6-element Vector{Int64}:
 1
 1
```

```
 2
 2
 3
 3

 tOut = 6-element Vector{Int64}:
 2
 3
 4
 5
 6
 7
```

(3). 定位具有指定边索引的边

创建一个图，然后确定其索引是 3 和 7 的边的端节点。

```
s = [1, 1, 1, 1, 2, 2, 3, 3, 4, 4];
t = [2, 3, 4, 5, 6, 7, 8, 9, 10, 11];
G = DiGraph(s, t)
```

```
G =
DiGraph with properties:
Edges:10×1
Nodes:11×0
```

```
sOut,tOut = findedge(G,[3,7])
```

```
sOut = 2-element Vector{Int64}:
1
3

 tOut = 2-element Vector{Int64}:
4
8
```

(4). 确定指定边的权重

创建一个图。

```
s = [1, 1, 2, 3];
t = [2, 3, 3, 4];
weights = [10, 20, 30, 40];
G = Graph(s, t, weights)
```

```
G =
Graph with properties:
Edges:4×2
Nodes:4×0
```

```
G.Edges.weights[findedge(G, 1, 3)[1]]
```

```
ans = 1-element Vector{Float64}:
 20.0
```

(5). 更改多重图边的权重

使用 findedge 更改多重图中数条边的权重。

创建一个多重图。此图在节点 2 和节点 4 之间有两条边。

```
s = [1,1,2,3,2,2];
t = [2,3,3,4,4,4];
weights = [10,20,30,40,10,10];
G = Graph(s, t, weights);
```

更改节点 (3,2) 和 (2,4) 之间的边的权重。为 findedge 指定两个输出，以获取端节点索引 m。当两个节点之间存在多条边时，此输出非常有用，因为 idxOut 中的元素数量可能多于 s 和 t 中的节点对数量。边 idxOut[1] = 3 连接节点对 (s[1],t[1]) = (3,2)，边 idxOut[2] = 4 和 idxOut[3] = 5 连接节点对 (s[2],t[2]) = (2,4)。

```
s = [3, 2];
t = [2, 4];
w = [1, 4];
idxOut, m = findedge(G, s, t)
```

```
idxOut = 3-element Vector{Int64}:
 3
 4
 5

 m = 3-element Vector{Int64}:
 1
 2
 2
```

```
G.Edges.weights[idxOut] = w[m];
```

3.2.8.2.10 edgecount

两个节点之间的边数

A. 语法

```
C = edgecount(G,s,t)
```

B. 说明

C = edgecount(G,s,t) 返回图 G 中源节点 s 与目标节点 t 之间的边数。示例。

C. 示例

(1). 节点之间的边数

创建一个图并显示边的表作为参考。计算节点 1 和节点 2 之间的边数。

```
G = Graph([1, 1, 1, 1, 2, 2], [2, 2, 2, 3, 4, 5]);
G.Edges
```

```
ans =
6×1
EndNodes
1       2
1       2
1       2
1       3
2       4
2       5
```

```
N = edgecount(G, 1, 2)
```

```
N = 1-element Vector{Int64}:
 3
```

您还可以指定单个源节点和多个目标节点。计算节点 1 和所有其他节点之间的边数（包括自环）。

```
N = edgecount(G, 1, 1:numnodes(G))
```

```
N = 5-element Vector{Int64}:
 0
 3
 1
 0
 0
```

3.2.8.2.11 subgraph

提取子图

A. 语法

```
H = subgraph(G,nodeIDs)
H = subgraph(G,idx)
```

B. 说明

H = subgraph(G,nodeIDs) 返回 G 的一个子图，其中仅包含 nodeIDs 指定的节点。示例

H = subgraph(G,idx) 使用逻辑向量指定子图节点。

C. 示例

(1). 节点之间的边数提取子图

创建一个图。

```
s = [1, 1, 1, 2, 5, 3, 6, 4, 7, 8, 8, 8];
t = [2, 3, 4, 5, 3, 6, 4, 7, 2, 6, 7, 5];
G = Graph(s, t);
```

使用置换向量对图节点重新排列

```
s = [1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 15, 15, 15, 15, 15];
t = [3, 5, 4, 2, 14, 6, 11, 12, 13, 10, 7, 9, 8, 15, 16, 17, 19,
18, 20];
G = Graph(s, t);
```

通过指定要包含的节点，从 G 提取子图。子图中的节点编号会重置。

```
idx = [2,15,16,17,18,19,20,1,3,4,5];
H = subgraph(G, idx);
```

(2). 具有节点和边属性的子图

创建一个具有命名节点的加权图。

```
s = [1, 1, 1, 2, 2, 2, 8, 8, 8, 8];
t = [2, 3, 4, 5, 6, 7, 9, 10, 11, 12];
weights = [10, 30, 40, 80, 60, 60, 20, 30, 90, 80];
names = ["A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K",
"L"];
G = Graph(s, t, weights, names);
```

提取一个子图，其中包含节点 "B" 及其所有相邻节点。subgraph 保留节点名称和边权重。但是，与 G 相比较，H 中的数值节点 ID 经过重新编号。

```
N = TyGraphTheory.neighbors(G, "B");
H = TyGraphTheory.subgraph(G, ["B"; N]);
```

3.2.8.3 分析结构

函数名	简介
conncomp	图的连通分量
biconncomp	双连通图分量
condensation	图凝聚
bctree	块割点树图
toposort	有向无环图的拓扑顺序
isdag	确定图是否为无环
transreduction	传递归约
transclosure	传递闭包
ismultigraph	确定图是否具有多条边
simplify	将多重图简化为简单图

3.2.8.3.1 conncomp

图的连通分量

A. 语法

```
bins,binsizes = conncomp(G)
bins,binsizes = conncomp(G,outputform)
bins,binsizes = conncomp(G,type)
bins,binsizes = conncomp(G,outputform,type)
```

B. 说明

bins,binsizes = conncomp(G) 以 bin 形式返回图 G 的连通分量。bin 编号

指示图中的每个节点所属的分量。以 binsize 返回连通分量的大小。binsize[i] 会给出分量 i 中的节点数量。该方法默认返回 bins 的向量类型，且对于有向图找的是强连通分量。示例

如果 G 是无向图，则有路径相连的两个节点属于同一分量。

如果 G 是有向图，则仅当两个节点之间存在一条双向连接路径时，这两个节点才属于同一强分量。

bins,binsizes = conncomp(G， outputform) 使用 outputform 指定返回 bins 的格式，默认为 "vector" 。

bins,binsizes = conncomp(G，type) 使用 type 指定有向图时寻找的是强连通分量还是弱连通分量，默认为 "strong" 。示例。

C. 示例

(1). 查找图分量

创建一个具有三个连通分量的无向图。使用 conncomp 确定每个节点所属的分量。

```
G = Graph([1,1,4],[2,3,5],[1,1,1],6);
```

```
bins,binsizes = conncomp(G)
```

```
bins = 6-element Vector{Int64}:
 1
 1
 1
 2
 2
 3

 binsizes = 3-element Vector{Int64}:
 3
 2
 1
```

(2). 强和弱图分量

创建一个有向图，然后计算强连通分量和弱连通分量。弱连通分量会忽略连接边的方向。

```
s = [1,2,2,3,3,3,4,5,5,5,8,8];
t = [2,3,4,1,4,5,5,3,6,7,9,10];
G = DiGraph(s,t);
```

```
str_bins,str_binsizes = conncomp(G)
```

```
str_bins = 10-element Vector{Int64}:
 4
 4
 4
 4
 4
 6
 5
 1
 3
 2
 str_binsizes = 6-element Vector{Int64}:
 1
 1
 1
 5
 1
 1
```

```
weak_bins,weak_binsizes = conncomp(G,"weak")
```

```
weak_bins = 10-element Vector{Int64}:
 1
 1
 1
 1
 1
 1
 1
 2
 2
 2
weak_binsizes = 2-element Vector{Int64}:
 7
 3
```

(3). 基于大小放弃图分量

使用 conncomp 的第二个输出提取图的最大分量或删除低于某个大小的分量。

创建一个有向图。该图有一个大分量、一个小分量和几个只包含一个节点的分量。

```
s = [1,2,2,3,3,3,4,5,5,5,8,8,9];
t = [2,3,4,1,4,5,5,3,6,7,9,10,10];
G = DiGraph(s,t,Float64[],20);
```

```
bin,binsize = conncomp(G,"weak")
```

```
bin = 20-element Vector{Int64}:
 1
 1
 1
 1
 1
 1
 1
 2
 2
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12

binsize = 12-element Vector{Int64}:
7
3
1
1
1
1
1
1
1
1
1
1
```

使用 binsize 提取图的最大分量。idx 是一个逻辑索引，指示每个节点是否

属于最大分量。subgraph 函数从 G 中提取由 idx 选择的节点

```
idx = binsize[bin] .== maximum(binsize);
SG = subgraph(G, idx);
```

binsizes 的一个类似用途是根据大小滤出分量。此过程与提取最大分量类似，不过在这一示例中，每个节点可以属于满足大小要求的任何分量。

滤出 G 中少于 3 个节点的任何分量。idx 是逻辑索引，指示每个节点是否属于具有 3 个或更多节点的分量。

```
idx = binsize[bin] .>= 3;
SG = subgraph(G, idx);
```

3.2.8.3.2 biconncomp

双连通图分量

A. 语法

```
bins,iC = biconncomp(G)
bins,iC = biconncomp(G,outputform)
```

B. 说明

bins,iC = biconncomp(G) 以 bin 形式返回图 G 的双连通分量。bin 编号指示图中的每个边所属的双连通分量。G 中的每条边属于一个双连通分量，而 G 中的节点可以属于多个双连通分量。如果两个节点无论从图中删除哪一个都不会使它们断开连接，则说明它们属于同一个双连通分量。节点索引 iC，指示哪些节点是割点（也称为关节点）。示例

bins,iC = biconncomp(G,outputform) outputform 的默认值为 "vector"，以向量形式返回输出。指定为 "tuple" 时以元组形式返回输出，因此 bins[j] 包含分量 j 中所有节点的节点 ID。示例。

C. 示例

(1). 标识割点

标识图中的割点。

创建一个图。计算每条图边所属的双连通分量，其中第二个输出返回标识割点的向量。

```
G = Graph([1,1,4],[2,3,5],[1,1,1],6);
```

```
edgebins,iC = biconncomp(G)
```

```
edgebins = 3-element Vector{Int64}:
 2
 1
 3

 iC = 1-element Vector{Int64}:
 1
```

(2). 以子图的形式提取双连通分量

此示例说明如何以子图的形式从图中提取双连通分量，然后使用原图中的节点索引标记每个子图中的节点。

创建一个图。

```
s = [1,1,2,2,3,4,4,5,6,6,7,7,8];
t = [2,3,3,4,4,5,7,6,7,10,8,9,9];
G = Graph(s,t);
```

```
bintuple = biconncomp(G, "tuple")[1];
n = length(bintuple);
for ii = 1:n
    subgraph(G, bintuple[ii]);
end
```

3.2.8.3.3 condensation

图凝聚

A. 语法

```
C = condensation(G)
```

B. 说明

C = condensation(G) 返回一个有向图 C，其节点表示 G 中的强连通分量。示例。

C. 示例

(1). 将强连通分量压缩为单个节点

创建包含多个强连通分量的图。

```
s = [1 1 2 3 3 4 4 4 4 5 5 6 6 6 7 8 8 9 9 10 10 10 11 11 12 13 13
14 15];
t = [1 3 1 2 5 1 2 12 13 6 8 7 8 10 10 9 10 5 11 9 11 14 12 14 13
11 15 13 14];
G = DiGraph(s[:],t[:]);
```

使用 condensation 将每个分量表示为单个节点。

```
C = condensation(G);
```

3.2.8.3.4 bctree

块割点树图

A. 语法

```
tree,ind = bctree(G)
```

B. 说明

tree,ind = bctree(G) 返回图 G 的块割点树，使 tree 中的每个节点代表 G 的一个双连通分量或割点。代表割点的节点与代表包含该割点的双连通分量的所有节点相连。示例

C. 示例

(1). 计算图的块割点树及块割点树的节点索引

计算一个图的块割点树、查看生成的节点属性。

创建一个图。

```
s = [1, 1, 2, 2, 3, 4, 4, 5, 6, 6, 7, 7, 8];
t = [2, 3, 3, 4, 4, 5, 7, 6, 7, 10, 8, 9, 9];
G = Graph(s, t);
```

计算图的块割点树并查看节点属性。

```
tree,ix = bctree(G);
tree.Nodes.props
```

```
ans = 7×3 Matrix{Int64}:
 1  1  0
 1  2  0
 1  3  0
 1  4  0
 0  0  4
```

```
 0  0  6
 0  0  7
```

计算图的块割点树并查看节点属性。

```
tree,ix = bctree(G);
tree.Nodes.props
```

```
ans = 7×3 Matrix{Int64}:
 1  1  0
 1  2  0
 1  3  0
 1  4  0
 0  0  4
 0  0  6
 0  0  7
```

```
ix
```

```
ix = 10-element Vector{Int64}:
 4
 4
 4
 5
 3
 6
 7
 1
 1
 2
```

每个索引 ix[j] 指示块割点树中的一个节点，它代表输入图中的节点 j。例如，tr 中的节点 4 代表 G 中的一个分量，该分量包含节点 1、2 和 3，所以 ix 中的前三个项都是 4。

3.2.8.3.5 toposort

有向无环图的拓扑顺序

A. 语法

```
n, H = toposort(G)
n, H = toposort(G, algorithm)
```

B. 说明

n,H = toposort(G) 返回 G 中符合以下条件的节点拓扑顺序：G 中每条边 (n[i],n[j]) 的 i < j。有向图 G 不能包含任何循环。示例

n,H = toposort(G,algorithm) 指定排序算法。例如，toposort(G,"stable") 对节点使用基于字典编纂顺序的稳定排序算法。示例。

C. 示例

(1). 节点的拓扑排序

创建一个图，它表示大学数学课程的进阶顺序。两个课程之间的边表示课程要求建包含多个强连通分量的图。

```
A = [0 1 1 0 0 0 0
     0 0 0 1 0 0 0
     0 1 0 1 0 0 1
     0 0 0 0 1 1 0
     0 0 0 0 0 0 0
     0 0 0 0 1 0 0
     0 0 0 0 1 0 0];
names = ["Calculus I", "Linear Algebra", "Calculus II",
"Multivariate Calculus", "Topology", "Differential Equations",
"Real Analysis"];
G = DiGraph(A,names);
```

找出课程的拓扑排序来确定正确的课程完成顺序。

```
N = toposort(G)[1]
```

```
N = 7-element Vector{Int64}:
 1
 3
 7
 2
 4
 6
 5
```

```
G.Nodes.nodeNames[N,:]
```

```
ans = 7×1 Matrix{String}:
 "Calculus I"
 "Calculus II"
 "Real Analysis"
 "Linear Algebra"
 "Multivariate Calculus"
 "Differential Equations"
 "Topology"
```

(2). 稳定拓扑排序

使用逻辑邻接矩阵创建有向图。

```
G=DiGraph([6,7,7,7,7,8,9,9,9,10,10,10,10,10],[4,1,2,3,6,5,2,7,8,1,2,4
,7,8]);
G = toposort(G)[2];
```

找出图节点的拓扑排序。即使 G 已为拓扑顺序 (1 2 3 4 5 6 7 8 9 10)，toposort 仍会对节点重新排序。

```
toposort(G)[1]
```

```
ans = 10-element Vector{Int64}:
  2
  1
  4
 10
  9
  8
  5
  7
  3
  6
```

指定 algorithm 为 "stable" 以使用稳定排序算法，这样会按照较小索引优先的顺序对节点进行排序。稳定排序不对已形成拓扑顺序的 G 进行重新排列。

```
toposort(G,"stable")
```

```
ans = 10-element Vector{Int64}:
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
```

3.2.8.3.6 isdag

确定图是否为无环

A. 语法

```
tf = isdag(G)
```

B. 说明

如果 G 是有向无环图，则 tf = isdag(G) 返回逻辑值 1 (true)；否则返回逻辑值 0 (false)。示例。

C. 示例

(1). 确定有向图是否无环

创建有向图，然后检验该图以确定它是否无环。

```
s = [1,1,2,2,3,3,4,4,4,5];
t = [2,3,4,5,6,7,8,9,10,4];
G = DiGraph(s,t)
```

```
G =
DiGraph with properties:
Edges:10×1
Nodes:10×0
```

```
tf = isdag(G)
```

```
tf = true
```

3.2.8.3.7 transreduction

传递规约

A. 语法

```
H = transreduction(G)
```

B. 说明

H = transreduction(G) 将图 G 的传递规约以新图 H 形式返回。H 中的节点与 G 中的节点相同，但 H 具有不同的边。H 包含最少的边数，因此，如果 G 中存在一条从节点 i 到节点 j 的路径，则 H 中也会有一条从节点 i 到节点 j 的路径。示例。

C. 示例

(1). 完整图的传递规约

创建一个完整的四阶图。

```
G = DiGraph([1,1,1,2,2,2,3,3,3,4,4,4],[2,3,4,1,3,4,1,2,4,1,2,3]);
```

查找传递归约所得的图。由于整个图中的可达性相当广泛，理论上有几个可能的传递归约，因为通过四个节点的任何循环都是一个候选传递归约。

```
H = transreduction(G);
```

两个具有相同可达性的图也具有相同的传递归约。因此，四个节点的任何循环会生成与 H 相同的传递归约。

创建一个包含不同的四节点循环的有向图：(1,3,4,2,1)。

```
G1 = DiGraph([1,3,4,2],[3,4,2,1]);
```

查找 G1 的传递归约。将 G1 中的循环重新排序，以使传递归约 H 和 H1 具有相同的循环 (1,2,3,4,1)。

```
H1 = transreduction(G1);
```

3.2.8.3.8 transclosure

传递闭包

A. 语法

```
H = transclosure(G)
```

B. 说明

H = transclosure(G) 将图 G 的传递闭包以新图 H 形式返回。H 中的节点与 G 中的节点相同，但 H 有其他边。如果 G 中有从节点 i 到节点 j 的路径，则 H 中的节点 i 和节点 j 之间有一条边。对于在同样两个节点之间具有多条边的多重图来说，输出图将用单条边代替它们。示例。

C. 示例

(1). 图的传递闭包

创建一个有向图。

```
G = DiGraph([1, 2, 3, 4, 4, 4, 5, 5, 5, 6, 7, 8], [2, 3, 5, 1, 3,
6, 6, 7, 8, 9, 9, 9]);
```

查找图 G 的传递闭包。H 包含与 G 相同的节点，但具有其他边。

```
H = transclosure(G);
```

H 中的传递闭包信息可用于回答有关原始图形 G 的可达性问题。

确定 G 中可从节点 1 到达的节点。这些节点是传递闭包图 H 中节点 1 的后继节点。

```
N = successors(H,1)
```

```
N = 7-element Vector{Int64}:
 2
 3
 5
 6
 7
 8
 9
```

(2). 计算可达性矩阵

创建一个有向图。

```
s = [1,1,2,2,3,4,4,5];
t = [2,4,3,4,5,5,6,6];
G = DiGraph(s,t);
```

计算 G 的传递闭包的邻接矩阵。结果将获得一个可达性矩阵，该矩阵具有非零值以指示从每个节点可到达哪些节点。

```
D = transclosure(G);
R = full(adjacency(D))
```

```
R = 6×6 Matrix{Bool}:
 0  1  1  1  1  1
 0  0  1  1  1  1
 0  0  0  0  1  1
 0  0  0  0  1  1
 0  0  0  0  0  1
 0  0  0  0  0  0
```

例如，要回答问题“哪些节点可从节点 3 到达？”，您可以查看矩阵中的第三行。该行指示只有节点 5 和 6 可从节点 3 到达：

```
findall(==(true),R[3,:])
```

3.2.8.3.9 ismultigraph

确定图是否具有多条边

A. 语法

```
tf = ismultigraph(G)
```

B. 说明

如果 G 的任意两个节点之间存在多条边，则 tf = ismultigraph(G) 返回逻辑值 1 (true)。否则，将返回逻辑值 0 (false)。

按照惯例，对于包含自环但没有重复边的图，ismultigraph 返回逻辑值 0 (false)。但是，包含多个自环的图被视为多重图。示例。

C. 示例

(1). 测试是否有多条边

使用 ismultigraph 确定输入图的两个节点之间是否有多条边。

创建一个图。

```
G = Graph([1,1,1,1,1,2,2,2], [2,2,3,4,5,6,7,8]);
```

检查 G 是否为多重图。结果是逻辑值 1 (true)，因为节点 1 和 2 之间有两条边 。

```
tf = ismultigraph(G)
```

```
tf = true
```

3.2.8.3.10 simplify

将多重图简化为简单图

A. 语法

```
H,eind,ecount = simplify(G)
H, eind, ecount = simplify(G, pickmethod="first")
H = simplify(G, aggregatemethod)
H,eind,ecount = simplify(___; propflag=false)
H,eind,ecount = simplify(___; selfloopflag=false)
H = simplify(___)
```

B. 说明

H,eind,ecount = simplify(G) 返回没有多条边或自环的图。当相同的两个节

点之间出现多条边时，只保留第一条边（如 G.Edges 中所定义）。边属性会保留。示例

还返回边索引 eind 和边计数 ecount：

H.Edges.endnodes[eind[i],:] 是 H 中表示 G 中的边 i 的边。

ecount[j] 是 G 中与 H 中的边 j 对应的边的条数。

H,eind,ecount = simplify(G,pickmethod="first") 指定在多条边之间进行选择的方法。边属性会保留。pickmethod 可以是 "first"（默认值）、"last"、"min" 或 "max"。示例

H = simplify(G,aggregatemethod) 指定的方法是将多条边的权重合并为一条新边的权重。G 中所有其他边属性均舍弃。aggregatemethod 可以是 "sum" 或 "mean"。无法返回 eind 和 ecount。示例

___ = simplify(___;selfloopflag=false) 使用上述语法中的任何输入参数组合指定是保留还是删除图中的自环。selfloopflag=true 指定具有一个或多个自环的节点在简化图中将只有一个自环。示例

___ = simplify(___;propflag=false) propflag 为 false（默认值） 时，根据边权重使用指定的方法简化图，propflag 为 true 时使用边属性。您可以使用上述语法中的任何输入参数组合。

C. 示例

(1). 将多重图简化为简单图

创建一个加权的无向多重图，在节点 1 和节点 2 之间有多条边。

```
G = Graph([1,1,1,1,2,3],[2,2,2,3,3,4], 1:6);
G.Edges
```

```
ans =
6×2
EndNodes       Weights
1      2       1
1      2       2
1      2       3
1      3       4
2      3       5
3      4       6
```

```
G = TyGraphTheory.simplify(G)[1];
G.Edges
```

```
ans =
4×2
```

```
EndNodes       Weights
1      2      1
1      3      4
2      3      5
3      4      6
```

(2). 选择或合并图中的多条边

使用 simplify 的第二个输入选择一种方法：是从多条边之中进行选择，还是将多条边合并成一条边。

创建一个加权的多重图。在此图中，节点 3 和节点 4 之间出现五条边，但边的权重是随机的。查看边。

```
G = Graph([1, 2, 3, 3, 3, 3, 3, 3], [2, 3, 1, 4, 4, 4, 4, 4], randi(10, 8));
G.Edges
```

```
ans =
8×2
EndNodes       Weights
1      2      10
1      3      7
2      3      3
3      4      10
3      4      7
3      4      3
3      4      10
3      4      10
```

命令 simplify(G) 保留重复边中的第一条边。但是，您可以使用第二个输入指定不同的选择/合并方法。

从多条边之中进行选择的选项包括："first"（默认值）、"last"、"min" 和 "max"。保留权重最大的重复边。

```
H_pick = TyGraphTheory.simplify(G,"max");
```

将多条边合并成一条边的选项包括："sum" 和 "mean"。将重复边相加生成一条权重较大的边

```
H_comb = TyGraphTheory.simplify(G,"sum");
```

(3). 保留图中的自环

使用 selfloopflag=true 在简化图的同时保留自环。

创建一个具有两个节点和几个自环的多重图。简化图并保留自环。

```
G = Graph([1,1,1,1,1,1,1,2,2,2,2],[1,1,1,1,2,2,2,2,2,2,2]);
```

```
G = TyGraphTheory.simplify(G; selfloopflag=true)[1];
```

(4). 边索引和重复边数

simplify 的第二个和第三个输出，获取有关合并了多少条边（以及哪些边）的信息。

创建一个具有三个节点和四条边的无向多重图。

```
G = Graph([1,1,1,2],[2,2,3,3]);
G.Edges
```

```
ans =
4×1
EndNodes
1       2
1       2
1       3
2       3
```

简化图并指定三个输出，以获取有关合并边的更多信息。

```
G, ei, ec = TyGraphTheory.simplify(G)
```

```
G = Graph with properties:
Edges:3×1
Nodes:3×0

ei = 4-element Vector{Int64}:
 1
 1
 2
 3

 ec = 3-element Vector{Int64}:
 2
 1
 1
```

ei[i] 是简化图中表示旧图中的边 i 的边。由于前两条边重复，因此 ei[1] = ei[2] = 1。而且 ec[1] = 2，因为新图中有两条边对应于旧图中的边 1。

3.2.8.4 遍历、最短路径和循环

函数名	简介

dfsearch	深度优先图搜索

3.2.8.4.1 dfsearch

深度优先图搜索

A. 语法

```
v = dfsearch(G,s)
T = dfsearch(G,s,events)
T,E = dfsearch(G,s,events)
___ = dfsearch(___; restart=false)
```

B. 说明

v = dfsearch(G,s) 从节点 s 开始对图 G 应用深度优先搜索。结果是按发现顺序排序的节点 ID 向量。示例

T = dfsearch(G,s,events) 通过标记一个或多个搜索事件，自定义深度优先搜索的输出。例如，T = dfsearch(G,s,"allevents") 返回包含所有已标记的事件，X = dfsearch(G,s,"edgetonew") 返回边的矩阵。示例

当 events 设置为 "edgetonew"、"edgetodiscovered" 或 "edgetofinished" 时，T,E = dfsearch(G,s,events) 还返回边索引 E 的向量。在多重图中，边索引是边的唯一性标识。

如果没有可从已发现的节点到达的新节点，___ = dfsearch(___;restart=false)（其中 restart=true）会重新启动搜索。您可以使用上述语法中的任何输入或输出参数组合。此选项可确保深度优先搜索到达图中的所有节点和边，即使这些节点和边无法从起始节点 s 到达。示例。

C. 示例

(1). 执行深度优先图搜索

创建一个图。

```
s = [1,1,1,1,2,2,2,2,2];
t = [3,5,4,2,6,10,7,9,8];
G = Graph(s,t);
```

从节点 7 开始对图执行深度优先搜索。结果指示节点发现顺序。

```
v = dfsearch(G,7)
```

```
v = 10-element Vector{Int64}:
```

```
 7
 2
 1
 3
 4
 5
 6
 8
 9
10
```

(2). 使用所有事件的深度优先图搜索

创建一个有向图。

```
A = [
       0 1 0 1 1 0 0
       0 0 0 0 0 0 0
       0 0 0 1 0 1 1
       0 0 0 0 0 1 0
       0 0 0 0 0 0 0
       0 0 0 0 0 0 0
       0 0 0 0 0 0 0
   ];
   G = DiGraph(A);
```

在节点 3 处开始对图执行深度优先搜索。指定 "allevents" 返回一个矩阵，其中包含算法中的所有事件。

```
T = dfsearch(G, 3, "allevents")
```

```
T = 13×5 Matrix{Any}:
 "startnode"        3  NaN  NaN  NaN
 "discovernode"     3  NaN  NaN  NaN
 "edgetonew"      NaN    3    4    4
 "discovernode"     4  NaN  NaN  NaN
 "edgetonew"      NaN    4    6    7
 "discovernode"     6  NaN  NaN  NaN
 "finishnode"       6  NaN  NaN  NaN
 "finishnode"       4  NaN  NaN  NaN
 "edgetofinished"  NaN    3    6    5
 "edgetonew"      NaN    3    7    6
 "discovernode"     7  NaN  NaN  NaN
 "finishnode"       7  NaN  NaN  NaN
 "finishnode"       3  NaN  NaN  NaN
```

要按照算法中的步骤执行，需要按从上到下的顺序读取表中的事件。例如：

1.算法在节点 3 处开始

2.在节点 3 和节点 4 之间发现一条边

3.发现节点 4

4.以此类推.

(3). 使用多个分量的深度优先图搜索

使用多个分量对图执行深度优先搜索。

创建一个有向图。此图有两个弱连通分量。

```
s = [1,1,2,2,2,3,4,7,8,8,8,8];
t = [3,4,7,5,6,2,6,2,9,10,11,12];
G = DiGraph(s,t);
```

```
c = conncomp(G,"weak")[1]
```

```
c = 12-element Vector{Int64}:
 1
 1
 1
 1
 1
 1
 1
 2
 2
 2
 2
 2
```

在节点 4 处开始执行图的深度优先搜索，并标记 "edgetonew"、"edgetodiscovered"、"edgetofinished" 和 "startnode" 事件。指定 restart = true，以使只要存在无法到达的剩余节点即重新启动搜索

```
events                                                              =
["edgetonew","edgetodiscovered","edgetofinished","startnode"];
T = dfsearch(G,4,events,restart=true)
```

```
T = 15×5 Matrix{Any}:
 "startnode"          4  NaN  NaN  NaN
 "edgetonew"        NaN    4    6    7
 "startnode"          1  NaN  NaN  NaN
 "edgetonew"        NaN    1    3    1
 "edgetonew"        NaN    3    2    6
```

```
 "edgetonew"        NaN    2    5    3
 "edgetofinished"    NaN    2    6    4
 "edgetonew"        NaN    2    7    5
 "edgetodiscovered"  NaN    7    2    8
 "edgetofinished"    NaN    1    4    2
 "startnode"          8  NaN  NaN  NaN
 "edgetonew"        NaN    8    9    9
 "edgetonew"        NaN    8   10   10
 "edgetonew"        NaN    8   11   11
 "edgetonew"        NaN    8   12   12
```

当 restart = true 时，"startnode" 事件会返回算法重新启动搜索时的位置和时间信息。

(4). 从图中删除循环

通过逆转有向图某些边的方向来生成无环图。

创建并绘制一个有向图

```
s = [1,2,3,3,3,3,4,5,6,7,8,9,9,9,10];
t = [7,6,1,5,6,8,2,4,4,3,7,1,6,8,2];
g = DiGraph(s,t);
```

对图执行深度优先搜索，从而标记 "edgetodiscovered" 事件。此事件对应于构成循环的各边。

```
e,edge_indices = dfsearch(g, 1, "edgetodiscovered", restart=true)
```

```
e = 3×2 Matrix{Int64}:
 3  1
 6  4
 8  7
edge_indices = 3-element Vector{Int64}:
  3
  9
 11
```

使用 flipedge 将标记的边反向，使其不再构成循环。这会删除图中的所有循环。使用 isdag 确认图为无环图

```
gnew = flipedge(g, edge_indices);
isdag(gnew)
```

```
ans = true
```

3.2.8.5 矩阵表示

函数名	简介
adjacency	图邻接矩阵
incidence	图关联矩阵
laplacian	图拉普拉斯矩阵

3.2.8.5.1 adjacency

图邻接矩阵

A. 语法

```
A = adjacency(G)
A = adjacency(G,"weighted")
A = adjacency(G,weights)
```

B. 说明

A = adjacency(G) 返回图 G 的稀疏邻接矩阵。如果 (i,j) 是 G 中的一条边，则 A(i,j) = 1。否则，A(i,j) = 0。示例

A = adjacency(G,"weighted") 返回加权邻接矩阵，其中对于每条边 (i,j)，值 A[i,j] 包含边的权重。如果图没有边权重，则 A[i,j] 设置为 1。对于此语法，G 必须是简单图，从而使 ismultigraph(G) 返回 false。

A = adjacency(G,weights) 返回加权邻接矩阵，其边权重由向量 weights 指定。对于 G 中的每条边 (i,j)，邻接矩阵的值 A[i,j] = weights(findedge(G,i,j))。对于此语法，G 必须是简单图，从而使 ismultigraph(G) 返回 false。

C. 示例

(1). 图的邻接矩阵

使用边列表创建一个有向图，然后求该图的等价邻接矩阵表示形式。邻接

矩阵以稀疏矩阵形式返回。

```
s = [1, 1, 1, 2, 2, 3];
t = [2, 3, 4, 5, 6, 7];
G = DiGraph(s, t)
```

```
G =
DiGraph with properties:
Edges:6×1
Nodes:7×0
```

```
A = adjacency(G)
```

```
A = 7×7 SparseArrays.SparseMatrixCSC{Bool, Int64} with 6 stored entries:
 ·  1  1  1  ·  ·  ·
 ·  ·  ·  ·  1  1  ·
 ·  ·  ·  ·  ·  ·  1
 ·  ·  ·  ·  ·  ·  ·
 ·  ·  ·  ·  ·  ·  ·
 ·  ·  ·  ·  ·  ·  ·
 ·  ·  ·  ·  ·  ·  ·
```

(2). 加权图的邻接矩阵

使用上三角邻接矩阵创建一个无向图。使用邻接矩阵构造一个图时，矩阵中的非零值对应于边权重。

```
A = [0 5 3 0;0 0 1 2; 0 0 0 11; 0 0 0 0]
```

```
A = 4×4 Matrix{Int64}:
 0  5  3   0
 0  0  1   2
 0  0  0  11
 0  0  0   0
```

```
G = Graph(A; type="upper")
```

```
G =
Graph with properties:
Edges:5×2
Nodes:4×0
```

```
G.Edges
```

```
ans =
5×2
EndNodes       Weights
1      2      5
1      3      3
2      3      1
2      4      2
```

```
3      4      11
```

使用 adjacency 返回图的邻接矩阵。无论用于构造图的邻接矩阵形式如何，adjacency 函数始终返回一个对称的稀疏邻接矩阵，其中只包含 1 和 0。

```
B = adjacency(G)
```

```
B = 4×4 SparseArrays.SparseMatrixCSC{Bool, Int64} with 10 stored entries:
 ·  1  1  ·
 1  ·  1  1
 1  1  ·  1
 ·  1  1  ·
```

(3). 重新构造加权邻接矩阵

创建一个加权图。

```
G = DiGraph([1,1,1,2,3,4],[2,3,4,4,2,3],[5,6,7,8,9,10]);
G.Edges
```

```
ans =
6×2
EndNodes       Weights
1      2      5
1      3      6
1      4      7
2      4      8
3      2      9
4      3      10
```

求图的邻接矩阵。

```
A = adjacency(G)
```

```
A = 4×4 SparseArrays.SparseMatrixCSC{Bool, Int64} with 6 stored entries:
 ·  1  1  1
 ·  ·  ·  1
 ·  1  ·  ·
 ·  ·  1  ·
```

这种形式的邻接矩阵不包含边权重。请使用 "weighted" 选项在邻接矩阵中包含边权重。

```
A = adjacency(G,"weighted")
```

```
A = 4×4 SparseMatrixCSC{Int64, Int64} with 6 stored entries:
 ·  5  6  7
 ·  ·  ·  8
 ·  9  ·  ·
 ·  ·  10 ·
```

预览矩阵的满存储版本。由于 G 是有向图，因此邻接矩阵不对称。但对无向图来说，邻接矩阵是对称的。

```
B = full(A)
```

```
B = 4×4 Matrix{Int64}:
 0  5   6  7
 0  0   0  8
 0  9   0  0
 0  0  10  0
```

3.2.8.5.2 incidence

图关联矩阵

A. 语法

```
I = incidence(G)
```

B. 说明

I = incidence(G) 返回图 G 的稀疏关联矩阵。如果 s 和 t 是 G 中第 j 条边的源和目标节点的节点 ID，则 I[s,j] = -1 且 I[t,j] = 1。即 I 的每一列指示 G 中单条边的源和目标节点。示例。

C. 示例

(1). 图关联矩阵

使用边列表创建一个图，然后计算图关联矩阵。

```
s = [1, 1, 1, 1, 1];
t = [2, 3, 4, 5, 6];
G = Graph(s, t);
I = incidence(G)
```

```
I = 6×5 SparseMatrixCSC{Float64, Int64} with 10 stored entries:
-1.0  -1.0  -1.0  -1.0  -1.0
 1.0    ·     ·     ·     ·
  ·    1.0    ·     ·     ·
  ·     ·    1.0    ·     ·
  ·     ·     ·    1.0    ·
  ·     ·     ·     ·    1.0
```

I 中的每一列包含两个非零项，指示 G 中单条边的端节点。

计算图拉普拉斯矩阵 L，并确认无向图的关系 L = I*I'。

```
L = TyMath.TyGraphTheory.laplacian(G);
L - I*I'
```

```
ans = 6×6 SparseMatrixCSC{Float64, Int64} with 0 stored entries:
  ⋅    ⋅    ⋅    ⋅    ⋅    ⋅
  ⋅    ⋅    ⋅    ⋅    ⋅    ⋅
  ⋅    ⋅    ⋅    ⋅    ⋅    ⋅
  ⋅    ⋅    ⋅    ⋅    ⋅    ⋅
  ⋅    ⋅    ⋅    ⋅    ⋅    ⋅
  ⋅    ⋅    ⋅    ⋅    ⋅    ⋅
```

3.2.8.5.3 laplacian

拉普拉斯矩阵

A. 语法

```
L = laplacian(G)
```

B. 说明

L = laplacian(G) 返回图拉普拉斯矩阵 L。每个对角条目 L[j,j] 由节点 j 的度 degree(G,j) 给出。L 的非对角条目表示 G 中的边，这样，如果节点 i 和 j 之间存在一条边，则 L[i,j] = L[j,i] = -1；否则 L[i,j] = L[j,i] = 0。输入图 G 不能是多重图，也不能包含自环，且边权重将被忽略。示例。

C. 示例

(1). 图关联矩阵

使用边列表创建一个图，然后计算图拉普拉斯矩阵。

```
s = [1, 1, 1, 1, 1];
t = [2, 3, 4, 5, 6];
G = Graph(s, t);
L = TyMath.TyGraphTheory.laplacian(G)
```

```
L = 6×6 SparseMatrixCSC{Int64, Int64} with 16 stored entries:
  5  -1  -1  -1  -1  -1
 -1   1   ⋅   ⋅   ⋅   ⋅
 -1   ⋅   1   ⋅   ⋅   ⋅
 -1   ⋅   ⋅   1   ⋅   ⋅
 -1   ⋅   ⋅   ⋅   1   ⋅
 -1   ⋅   ⋅   ⋅   ⋅   1
```

L 的对角线元素指示节点的度，即 L[j,j] 是节点 j 的度。

计算图关联矩阵 I，并确认关系 L = I*I'。

```
I = incidence(G);
L - I*I'
```

```
ans = 6×6 SparseMatrixCSC{Float64, Int64} with 0 stored entries:
  .    .    .    .    .    .
  .    .    .    .    .    .
  .    .    .    .    .    .
  .    .    .    .    .    .
  .    .    .    .    .    .
  .    .    .    .    .    .
```

3.2.8.6 节点信息

函数名	简介	测例路径	测试方法	测试结论
degree	图节点的度	Doc/TyMath/TyGraphTheory/NodeInformation/degree.html	手动测试	通过
neighbors	图节点的相邻节点	Doc/TyMath/TyGraphTheory/NodeInformation/neighbors.html	手动测试	通过
indegree	节点的入度	Doc/TyMath/TyGraphTheory/NodeInformation/indegree.html	手动测试	通过
outdegree	节点的出度	Doc/TyMath/TyGraphTheory/NodeInformation/outdegree.html	手动测试	通过
predecessors	前趋节点	Doc/TyMath/TyGraphTheory/NodeInformation/predecessors.html	手动测试	通过
successors	后继节点	TyMath/TyGraphTheory/NodeInformation/successors.html	手动测试	通过
inedges	进入节点的入向边	Doc/TyMath/TyGraphTheory/NodeInformation/inedges.html	手动测试	通过
outedges	节点的出向边	TyMath/TyGraphTheory/NodeInformation/outedges.html	手动测试	通过

3.2.8.6.1 degree

图节点的度

A. 语法

```
D = degree(G)
D = degree(G,nodeIDs)
```

B. 说明

D = degree(G) 返回图 G 中每个节点的度。度是连接到每个节点的边的数量。示例

D = degree(G,nodeIDs) 返回由 nodeIDs 指定的节点的度。示例。

C. 示例

(1). 所有图节点的度

创建一个图，然后使用 degree 求出图中每个节点的度。

```
s = [1,1,1,4,4,6,6,6];
t = [2,3,4,5,6,7,8,9];
G = Graph(s,t);
```

```
deg = TyGraphTheory.degree(G)
```

```
Adeg = 9-element Vector{Int64}:
 3
 1
 1
 3
 1
 4
 1
 1
 1
```

deg[j] 指示节点 j 的度。

(2). 图节点子集的度

创建一个图，然后求出图中第一个、第三个和第五个节点的度。

```
s = ["a","a","a","d","d","f","f","f"];
t = ["b","c","d","e","f","g","h","i"];
```

```
G = Graph(s,t);
```

```
nodeIDs = ["a","c","e"];
deg = TyGraphTheory.degree(G,nodeIDs)
```

```
deg = 3-element Vector{Int64}:
 3
 1
 1
```

deg[j] 指示节点 nodeIDs[j] 的度。

3.2.8.6.2 indegree

节点的入度

A. 语法

```
D = indegree(G)
D = indegree(G,nodeIDs)
```

B. 说明

D = indegree(G) 返回一个列向量，其中包含 G 中每个节点的入度。示例

D = indegree(G,nodeIDs) 返回 nodeIDs 指定的节点的入度。示例。

C. 示例

(1). 所有图节点的入度

创建有向图，然后计算图中每个节点的入度。节点的入度等于将以该节点为目标节点的边的数量。

```
s = [1, 3, 2, 2, 4, 5, 1, 2];
t = [2, 2, 4, 5, 6, 6, 6, 6];
G = DiGraph(s, t);
```

```
indeg = indegree(G)
```

```
indeg = 6-element Vector{Int64}:
 0
 2
 0
```

```
1
1
4
```

indeg[j] 指示节点 j 的入度。

(2). 图节点的子集的入度

创建一个具有指定节点的有向图。然后计算将 "a"、"b" 和 "f" 节点作为其目标的边的数量。

```
s = ["a", "c", "b", "b", "d", "e", "a", "b"];
t = ["b", "b", "d", "e", "f", "f", "f", "f"];
G = DiGraph(s, t);
```

```
nodeID = ["a","b","f"];
indeg = indegree(G,nodeID)
```

```
iindeg = 3-element Vector{Int64}:
 0
 2
 4
```

indeg[j] 指示节点 nodeID[j] 的入度。

3.2.8.6.3 inedges

进入节点的入向边

A. 语法

```
eid, nid = inedges(G,nodeID)
```

B. 说明

eid,nid = inedges(G,nodeID) 返回有向图 G 中进入节点 nodeID 的所有入向边的索引和通过 eid 中的边连接到 nodeID 的前趋节点的节点 nid。示例。

C. 示例

(1). 找到入向边和前趋节点

使用 bucky 邻接矩阵创建有向图。

```
G = DiGraph(bucky());
G = DiGraph(s, t)
```

确定节点 1 的入向边和前趋节点 1

```
eid,nid = inedges(G,1)
```

```
eid = 3-element Vector{Int64}:
  4
 13
 16
nid = 3-element Vector{Int64}:
 2
 5
 6
```

```
X = G.Edges.endnodes[eid,:]
```

```
X = 3×2 Matrix{Int64}:
 2  1
 5  1
 6  1
```

3.2.8.6.4 neighbors

图节点的相邻节点

A. 语法

```
N = neighbors(G,nodeID)
```

B. 说明

N = neighbors(G,nodeID) 返回通过边连接到由 nodeID 指定的节点的所有节点的节点 ID。示例。

C. 示例

(1). 相邻图节点

创建一个图，然后确定节点 10 的相邻节点。

```
G = Graph(bucky());
```

```
N = TyGraphTheory.neighbors(G, 10)
```

```
N = 3-element Vector{Int64}:
  6
  9
 12
```

3.2.8.6.5 outdegree

节点的出度

A. 语法

```
D = outdegree(G)
D = outdegree(G,nodeIDs)
```

B. 说明

D = outdegree(G) 返回一个列向量，其中包含 G 中每个节点的出度。示例

D = outdegree(G,nodeIDs) 返回 nodeIDs 指定的节点的出度。示例。

C. 示例

(1). 所有图节点的出度

创建有向图，然后计算图中每个节点的出度。节点的出度等于将以该节点为目标节点的边的数量。

```
s = [1, 3, 2, 2, 4, 5, 1, 2];
t = [2, 2, 4, 5, 6, 6, 6, 6];
G = DiGraph(s, t);
```

```
outdeg = TyGraphTheory.outdegree(G)
```

```
outdeg = 6-element Vector{Int64}:
 2
 3
 1
 1
 1
 0
```

outdeg[j] 指示节点 j 的出度。

(2). 节点的子集的出度

创建一个具有指定节点的有向图。然后计算将 "a"、"b" 和 "f" 节点作为其源的边的数量。

```
s = ["a", "c", "b", "b", "d", "e", "a", "b"];
t = ["b", "b", "d", "e", "f", "f", "f", "f"];
G = DiGraph(s, t);
```

```
nodeID = ["a","b","f"];
outdeg = TyGraphTheory.outdegree(G, nodeID)
```

```
outdeg = 3-element Vector{Int64}:
2
3
0
```

outdeg[j] 指示节点 nodeID[j] 的出度

3.2.8.6.6 outedges

进入节点的出向边

A. 语法

```
eid, nid = outedges(G,nodeID)
```

B. 说明

eid,nid = outedges(G,nodeID) 返回有向图 G 中进入节点 nodeID 的所有出向边的索引和通过 eid 中的边连接到 nodeID 的后继节点的节点 nid。示例。

C. 示例

(1). 找到出向边和后继节点

使用 bucky 邻接矩阵创建有向图。

```
G = DiGraph(bucky());
```

确定节点 10 的出向边和后继节点。

```
eid,nid = outedges(G,10)
```

```
eid = 3-element Vector{Int64}:
 28
 29
 30

 nid = 3-element Vector{Int64}:
  6
  9
 12
```

```
X = G.Edges.endnodes[eid,:]
```

```
X = 3×2 Matrix{Int64}:
 10   6
 10   9
 10  12
```

3.2.8.6.7 predecessors

前趋节点

A. 语法

```
preIDs = predecessors(G,nodeID)
```

B. 说明

preIDs = predecessors(G,nodeID) 返回 nodeID 指定的有向图 G 中节点的前趋节点。前趋节点构成有向边，其中 preIDs 为源节点，nodeID 为目标节点。示例。

C. 示例

(1). 前趋节点

创建有向图，并确定节点 "e" 的前趋节点。

```
s = [1,1,1,2,2,3,3,7,8];
t = [2,3,4,5,6,7,8,5,5];
names = ["a","b","c","d","e","f","g","h"];
G = DiGraph(s, t, Float64[], names);
```

```
preIDs = predecessors(G,"e")
```

```
preIDs = 3-element Vector{String}:
 "b"
 "g"
 "h"
```

3.2.8.6.8 successors

后继节点

A. 语法

```
sucIDs = successors(G,nodeID)
```

B. 说明

sucIDs = successors(G,nodeID) 返回由 nodeID 指定的有向图 G 中节点的后继节点。后继节点构成有向边，其中 nodeID 为源节点，sucIDs 为目标节点。示例。

C. 示例

(1). 后继节点

创建有向图，并确定节点 "a" 的后继节点。

```
s = [1,1,1,2,2,3,3,7,8];
t = [2,3,4,5,6,7,8,5,5];
names = ["a","b","c","d","e","f","g","h"];
G = DiGraph(s, t, Float64[], names);
```

```
sucIDs = successors(G,"a")
```

```
sucIDs = 3-element Vector{String}:
 "b"
 "c"
 "d"
```

3.3 图形 API

二维和三维绘图、图像、动画

图形函数包括二维和三维绘图函数，用于以可视化形式呈现数据和通信的结果。以交互方式或编程方式自定义绘图。

3.3.1 二维和三维图

绘制连续、离散、曲面以及三维体数据图

使用绘图以可视化形式呈现数据。例如，您可以比较多组数据、跟踪数据随时间所发生的更改或显示数据分布。

3.3.1.1 线图

线图、对数图和函数图

在比较数据集或跟踪数据随时间变化方面，线图是一个非常有用的方法。您可以使用线性刻度或对数刻度在二维或三维视图中绘制数据。此外，还可以

在特定区间上绘制表达式或函数。

3.3.1.1.1 线图

函数名	简介
plot	二维线图
plot3	三维点或线图
stairs	阶梯图
errorbar	含误差条的线图
ezplot	易用的函数绘图函数
area	填充区二维绘图

plot

二维线图

A. 语法

```
plot(X,Y)
plot(X,Y,fmt)
plot(X1,Y1,...,Xn,Yn)
plot(X1,Y1,fmt1,...,Xn,Yn,fmtn)

plot(Y)
plot(Y,fmt)

plot(___,Key=Value)
plot(ax,___)


h = plot(___)
```

B. 说明

plot(X,Y) 创建 Y 中数据对 X 中对应值的二维线图。

如果 X 和 Y 都是向量，则它们的长度必须相同。plot 函数绘制 Y 对 X 的图。

如果 X 和 Y 均为矩阵，则它们的大小必须相同。plot 函数绘制 Y 的列对 X 的列的图。

如果 X 或 Y 中的一个是向量而另一个是矩阵，则矩阵的各维中必须有一维与向量的长度相等。如果矩阵的行数等于向量长度，则 plot 函数绘制矩阵中的每一列对向量的图。 如果矩阵的列数等于向量长度，则该函数绘制矩阵中的每一行对向量的图。如果矩阵为方阵，则该函数绘制每一列对向量的图。

如果 X 或 Y 之一为标量，而另一个为标量或向量，则 plot 函数会绘制离散点。但是，要查看这些点，您必须指定标记符号，例如 plot(X,Y,"o")。

plot(X,Y,fmt) 设置线型、标记符号和颜色。

plot(X1,Y1,...,Xn,Yn)绘制多个 X、Y 对组的图，所有线条都使用相同的坐标区。

plot(X1,Y1,fmt1,...,Xn,Yn,fmtn)设置每个线条的线型、标记符号和颜色。您可以混用 X、Y、fmt 三元组和 X、Y 对组：例如，plot(X1,Y1,X2,Y2,fmt2,X3,Y3)。

plot(Y)创建 Y 中数据对每个值索引的二维线图。

如果 Y 是向量，x 轴的刻度范围是从 1 至 length(Y)。

如果 Y 是矩阵，则 plot 函数绘制 Y 中各列对其行号的图。x 轴的刻度范围是从 1 到 Y 的行数。

如果 Y 是复数，则 plot 函数绘制 Y 的虚部对 Y 的实部的图，使得 plot(Y) 等效于 plot(real(Y),imag(Y))。

plot(Y,fmt) 设置线型、标记符号和颜色。

plot(___,Key=Value)使用一个或多个 Key=Value 对组参数指定线条属性。可以将此选项与前面语法中的任何输入参数组合一起使用。名称-值对组设置将应用于绘制的所有线条。

plot(ax,___)将在由 ax 指定的坐标区中，而不是在当前坐标区(gca)中创建线条。选项 ax 可以位于前面的语法中的任何输入参数组合之前。

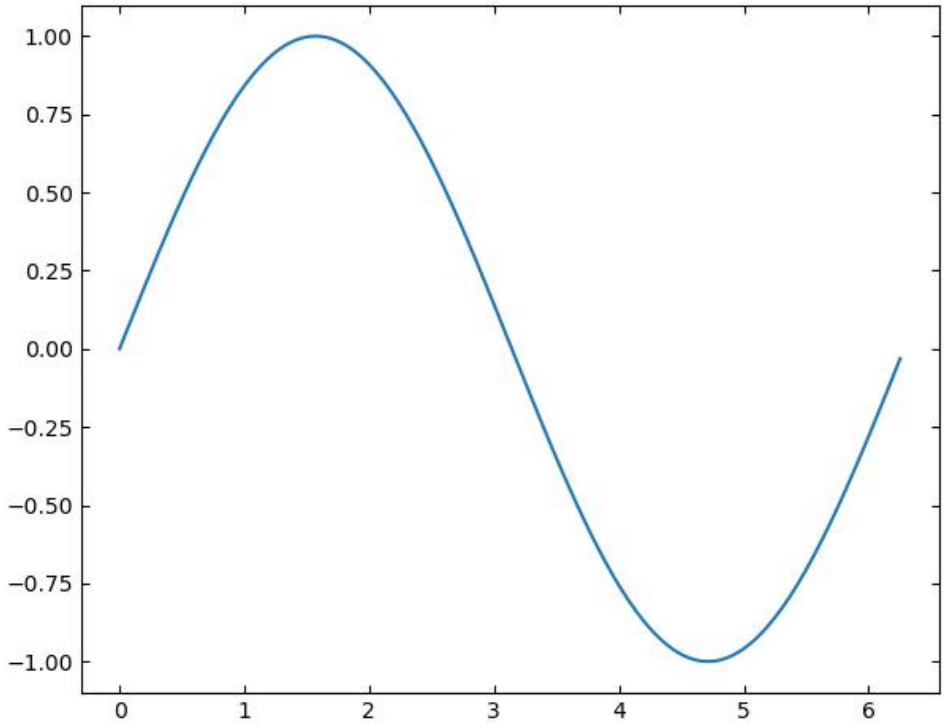
h=plot(___)返回由图形线条对象组成的列向量。在创建特定的图形线条后，可以使用 h 修改其属性。

C. 示例

(1). 创建线图

将 x 创建为由 0 和 2π 之间的线性间隔值组成的向量。在各值之间使用递增量 π/100。将 y 创建为 x 的正弦值。创建数据的线图。

```
x = 0:(pi / 100):(2 * pi);
y = sin.(x);
# x, y = get_xy()
plot(x, y)
```



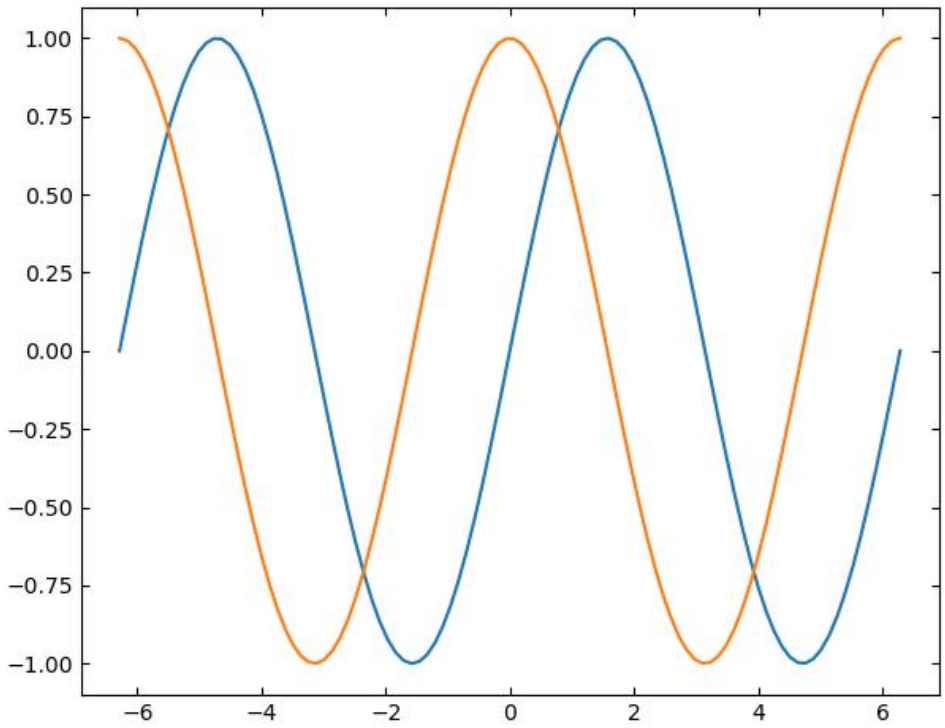
(2). 绘制多个线条

将 x 定义为 100 个介于 −2π 和 2π 之间的线性间隔值。将 y1 和 y2 定义为 x 的正弦和余弦值。创建上述两个数据集的线图。

```
using TyPlot

x = LinRange(-2 * pi, 2 * pi, 100);
y1 = sin.(x);
y2 = cos.(x);

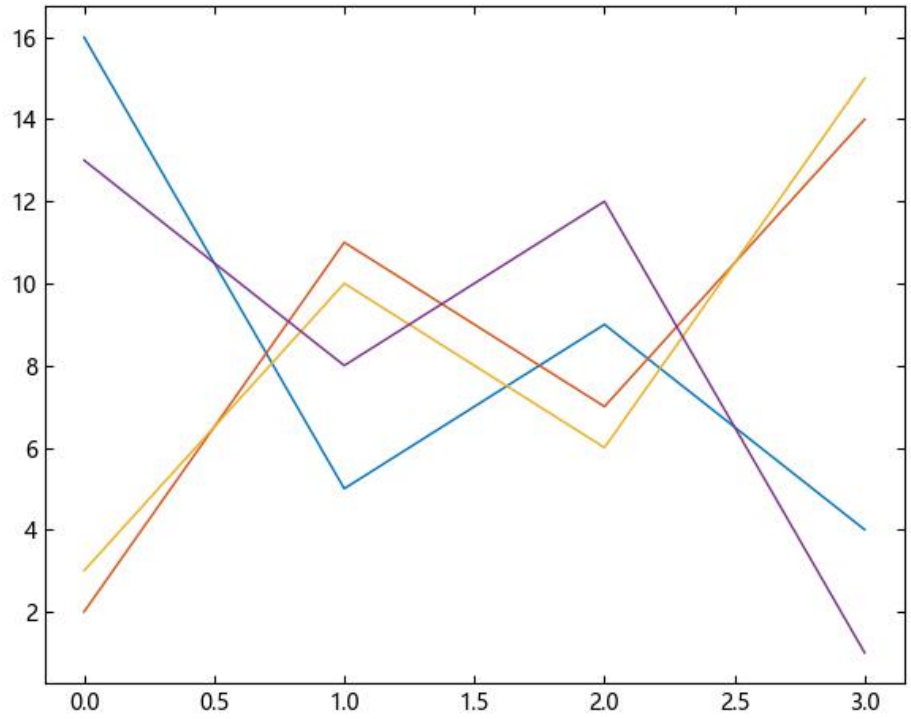
figure();
plot(x, y1, x, y2)
```



(3). 根据矩阵创建线图

将 Y 定义为 4×4 矩阵。创建 Y 的二维线图。将矩阵的每一列绘制为单独的线条。

```
using TyPlot
Y = [
    16 2 3 13
    5 11 10 8
    9 7 6 12
    4 14 15 1
]
figure();
plot(Y)
```

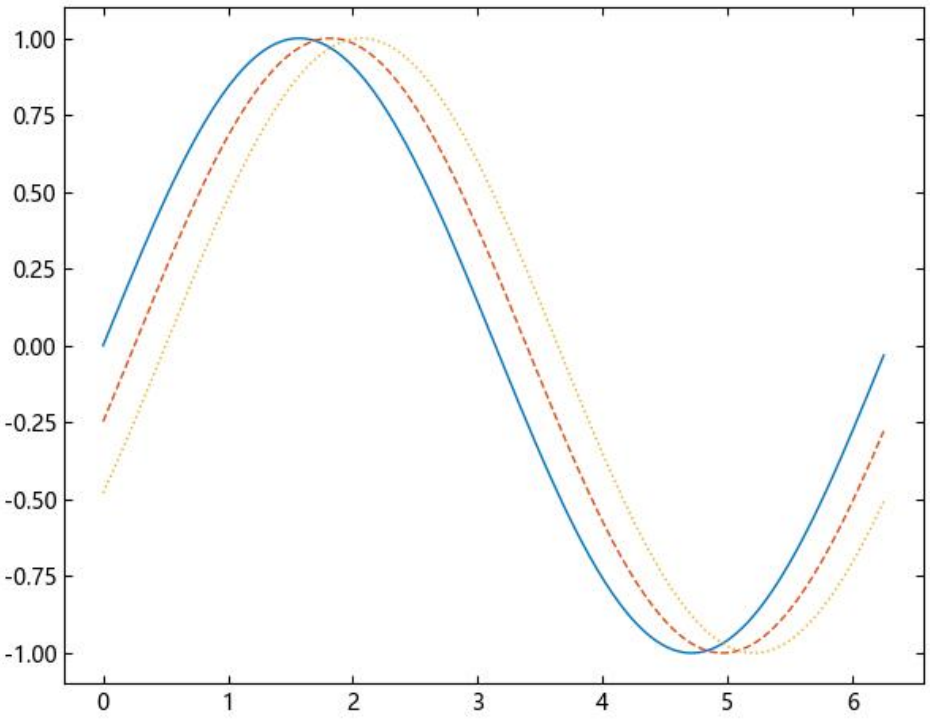


(4). 指定线型

绘制三条正弦曲线，每条曲线之间存在较小的相移。第一条曲线使用默认的线型。为第二条曲线指定虚线样式，第三条曲线指定点式线样式。

```
using TyPlot
x = 0:(pi / 100):(2 * pi);
y1 = sin.(x);
y2 = sin.(x .- 0.25);
y3 = sin.(x .- 0.5);

figure();
plot(x, y1, x, y2, "--", x, y3, ":")
```

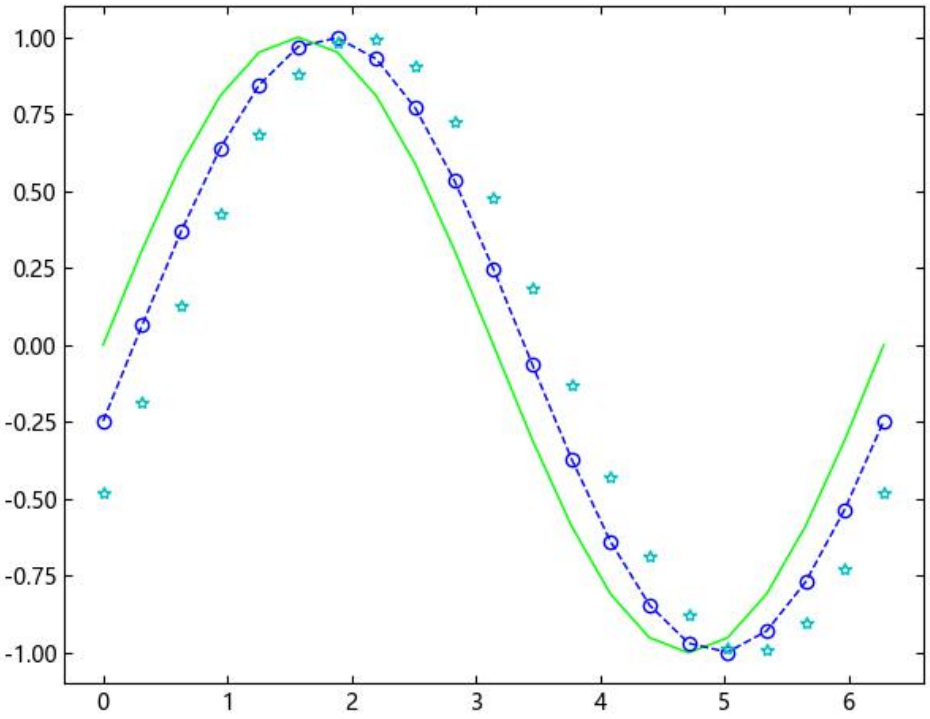


(5). 指定线型、颜色和标记

绘制三条正弦曲线，每条曲线之间存在较小的相移。第一条正弦曲线使用绿色线条，不带标记。第二条正弦曲线使用蓝色虚线，带圆形标记。第三条正弦曲线只使用青蓝色星号标记。

```
using TyPlot
x = 0:(pi / 10):(2 * pi);
y1 = sin.(x);
y2 = sin.(x .- 0.25);
y3 = sin.(x .- 0.5);

figure();
plot(x, y1, "#00FF00", x, y2, "b--o", x, y3, "c*")
```

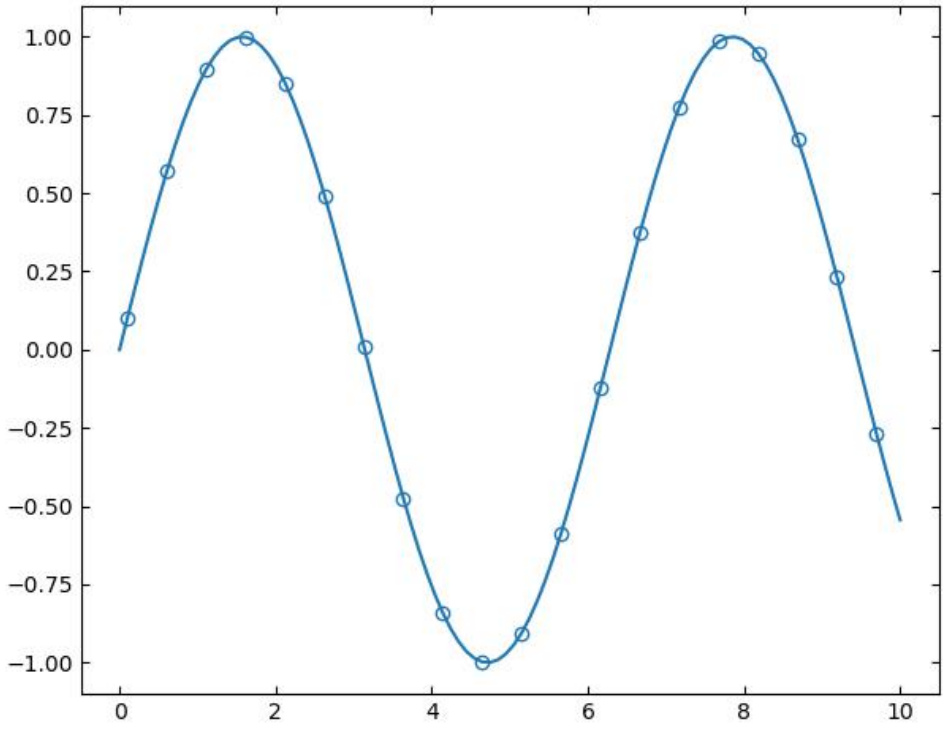


(6). 在特定的数据点显示标记

通过指定标记符号并将 markevery 属性设置为名称-值对组，创建一个线图并每隔四个数据点显示一个标记。

```
using TyPlot

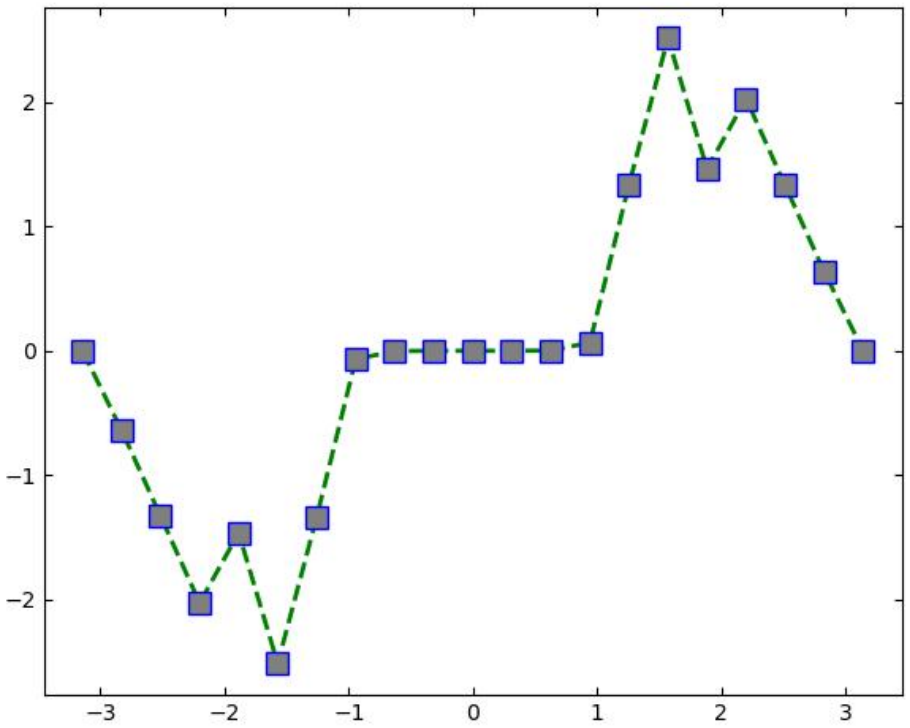
x = LinRange(0, 10, 100);
y = sin.(x);
plot(x, y, "-o"; markevery=1:5:length(y))
```



(7). 指定线宽、标记大小和标记颜色

创建线图并使用 fmt 选项指定带正方形标记的绿色虚线。使用 Key=Value 对组来指定线宽、标记大小和标记颜色。将标记边颜色设置为蓝色，并使用 RGB 颜色值设置标记面颜色。

```
using TyPlot
x = (-pi):(pi / 10):pi;
y = tan.(sin.(x)) - sin.(tan.(x));
figure()
plot(
    x,
    y,
    "--gs";
    linewidth=2,
    markersize=10,
    markeredgecolor="b",
    markerfacecolor=[0.5, 0.5, 0.5],
)
```



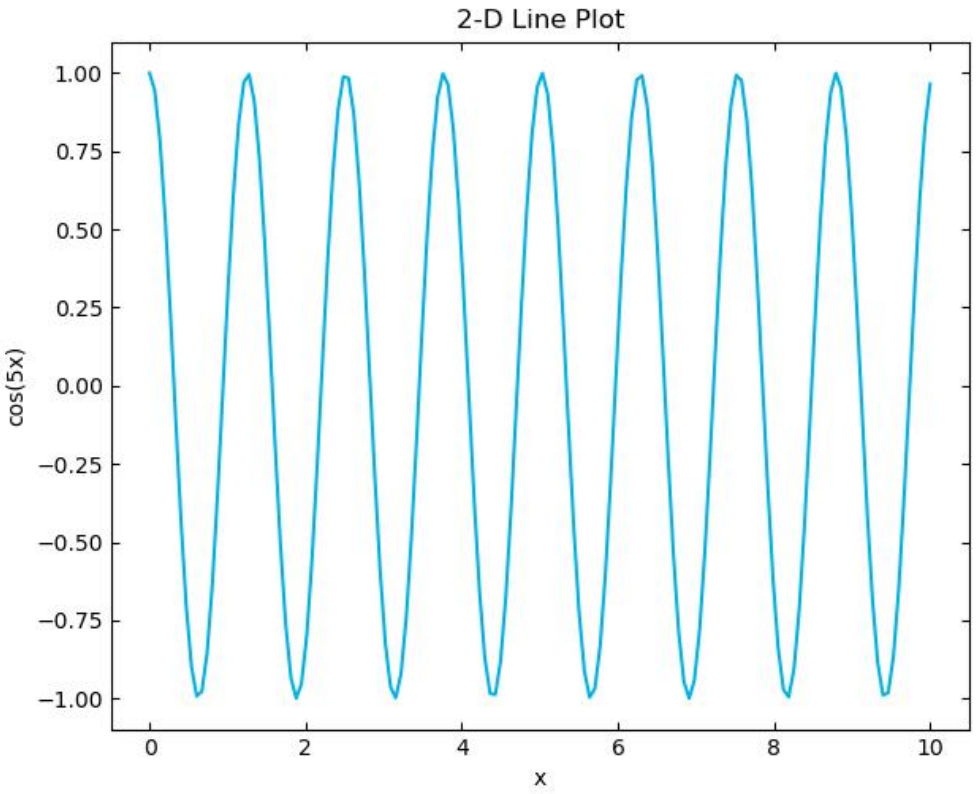
(8). 添加标题和轴标签

使用 LinRange 函数将 x 定义为 0 到 10 之间 150 个值组成的向量。将 y 定义为 x 的余弦值。

```
using TyPlot

x = LinRange(0, 10, 150);
y = cos.(5 * x);

figure()
plot(x, y; color=[0, 0.7, 0.9])
title("2-D Line Plot")
xlabel("x")
ylabel("cos(5x)")
```

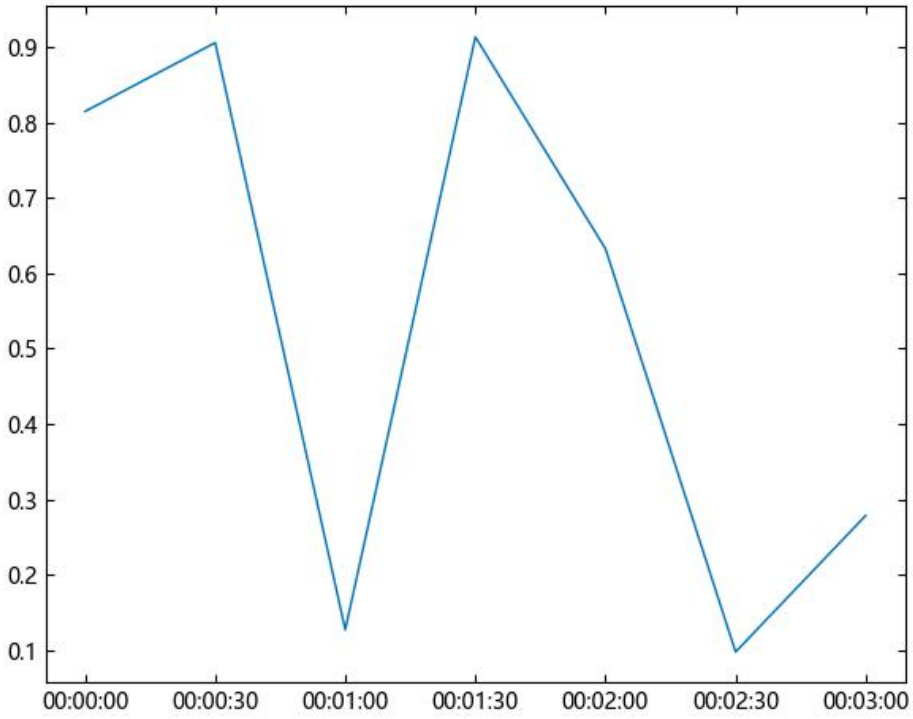


(9). 绘制持续时间并指定刻度格式

绘制持续时间并指定刻度格式。

```
using TyPlot
import Dates: Second, unix2datetime
t = range(Second(0), Second(180); step=Second(30))
t = [t...]
unix_t = unix2datetime(0)
dt = unix_t .+ t
y = [0.8147 0.9058 0.1270 0.9 134 0.6324 0.0975 0.2785]
y = [y...]

plot(dt, y);
```



(10). 指定线图的坐标区

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。通过将 ax1 传递给 plot 函数来创建顶部绘图。通过将坐标区传递给 title 和 ylabel 函数，为图添加标题和 y 轴标签。重复该过程以创建底部绘图。

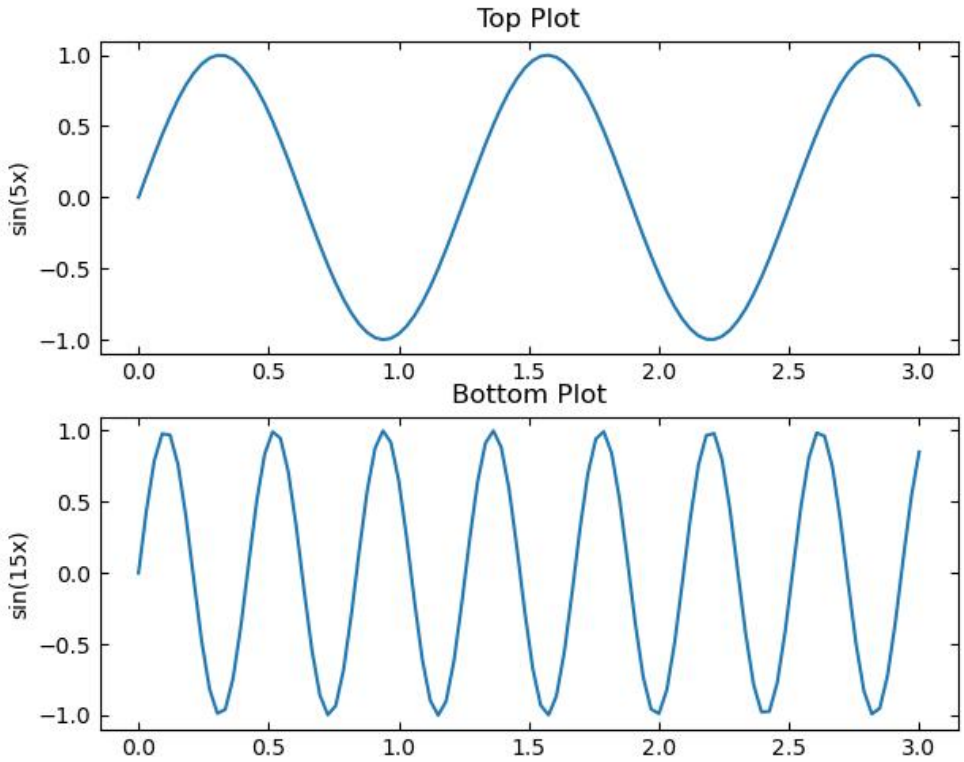
```
using TyPlot
x = LinRange(0, 3, 100);
y1 = sin.(5 * x);
y2 = sin.(15 * x);

ax1 = subplot(2, 1, 1)
ax2 = subplot(2, 1, 2)

# Top plot
plot(ax1, x, y1)
title(ax1, "Top Plot")
ylabel(ax1, "sin(5x)")

# Bottom plot
```

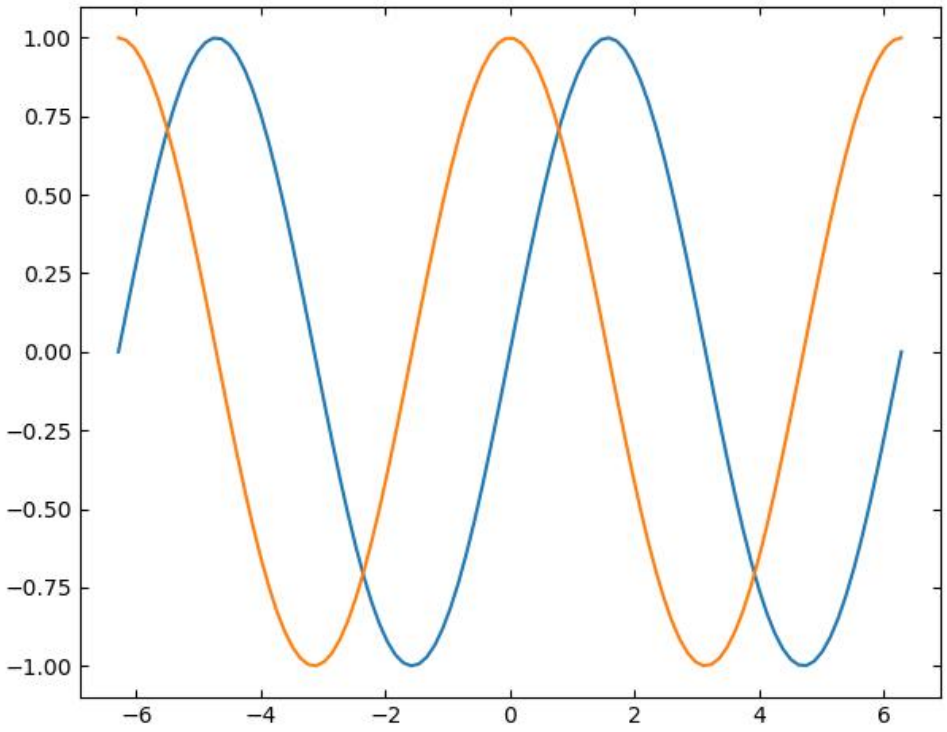
```
plot(ax2, x, y2)
title(ax2, "Bottom Plot")
ylabel(ax2, "sin(15x)")
```



(11). 创建并修改线条

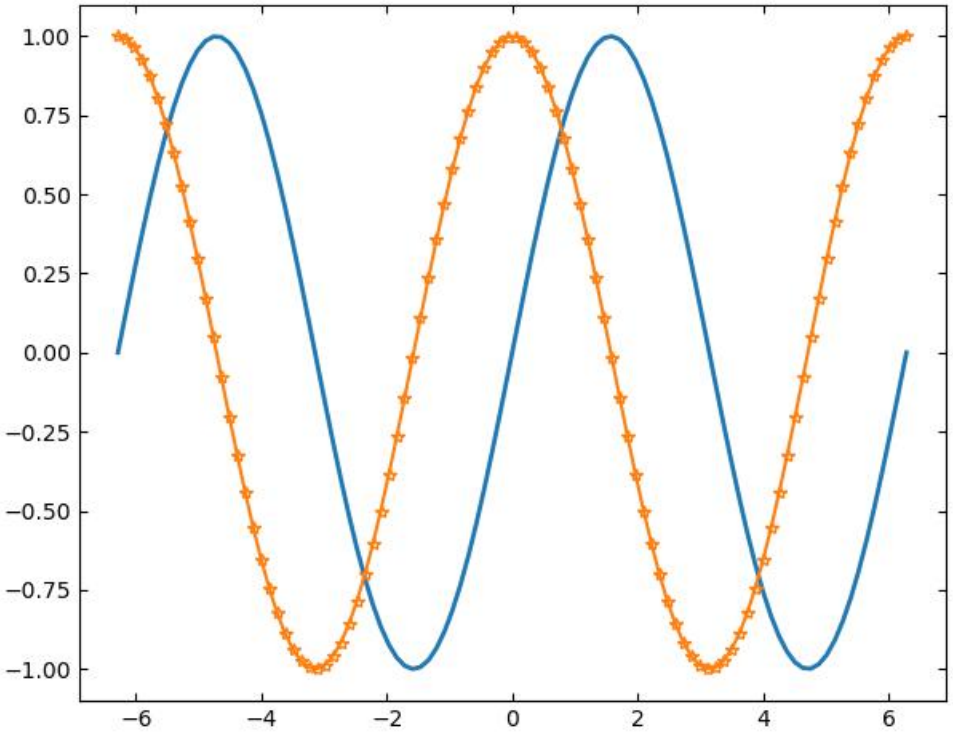
将 x 定义为 100 个介于 −2π 和 2π 之间的线性间隔值。将 y1 和 y2 定义为 x 的正弦和余弦值。为上述两个数据集分别创建线图，并在 p 中返回两个图形线条。

```
using TyPlot
x = LinRange(-2 * pi, 2 * pi, 100);
y1 = sin.(x);
y2 = cos.(x);
p = plot(x, y1, x, y2);
```



将第一个线条的线宽更改为 2。向第二行添加星形标记。使用圆点表示法设置属性。

```
p[1].set_linewidth(2)
p[2].set_marker("*")
```

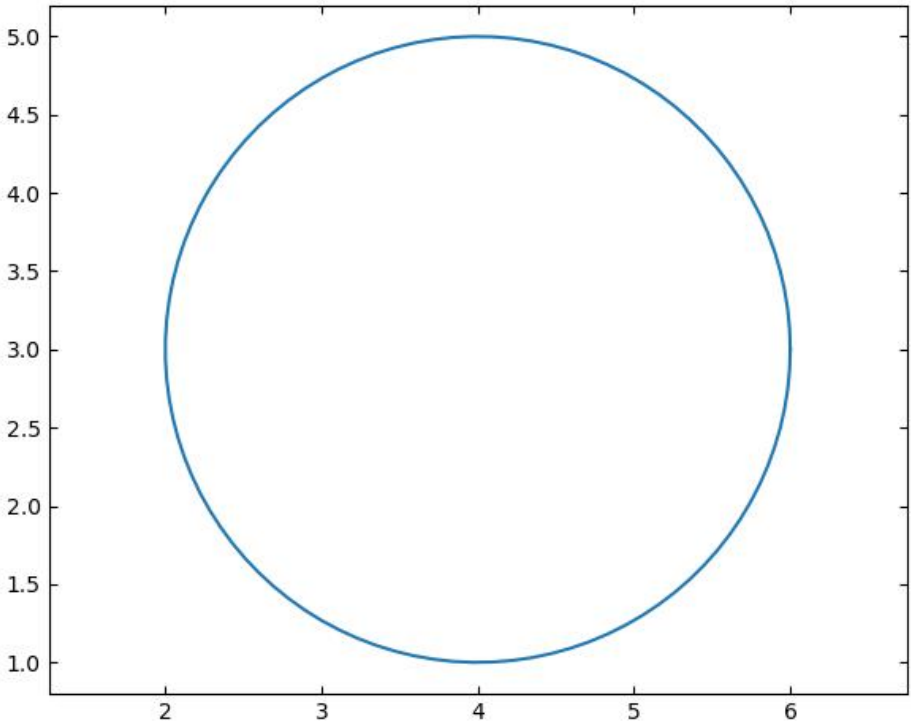


(12). 绘制圆形

绘制以点 (4,3) 为中心以 2 为半径的圆。使用 axis("equal") 可沿每个坐标方向使用相等的数据单位。

```
using TyPlot

r = 2;
xc = 4;
yc = 3;
theta = LinRange(0, 2 * pi, 100);
x = r * cos.(theta) .+ xc;
y = r * sin.(theta) .+ yc;
plot(x, y)
axis("equal")
```



plot3

三维点或线图

A. 语法

```
plot3(X,Y,Z)
plot3(X,Y,Z,fmt)
plot3(X1,Y1,Z1,...,Xn,Yn,Zn)
plot3(X1,Y1,Z1,fmt1,...,Xn,Yn,Zn,fmtn)
plot3(___,Key=Value)
plot3(ax,___)
p = plot3(___)
```

B. 说明

plot3(X,Y,Z) 绘制三维空间中的坐标。

要绘制由线段连接的一组坐标，请将 X、Y、Z 指定为相同长度的向量。

要在同一组坐标轴上绘制多组坐标，请将 X、Y 或 Z 中的至少一个指定为矩阵，其他指定为向量.

plot3(X,Y,Z,fmt)使用指定的线型、标记和颜色创建绘图。

plot3(X1,Y1,Z1,...,Xn,Yn,Zn)在同一组坐标轴上绘制多组坐标。使用此语法作为将多组坐标指定为矩阵的替代方法。

plot3(X1,Y1,Z1,fmt1,...,Xn,Yn,Zn,fmtn)可为每个 XYZ 三元组指定特定的线型、标记和颜色。您可以对某些三元组指定 fmt，而对其他三元组省略它。例如，plot3(X1,Y1,Z1,"o",X2,Y2,Z2)对第一个三元组指定标记，但没有对第二个三元组指定标记。

plot3(___,Key=Value)使用一个或多个名称-值对组参数指定 Line3D 属性。在所有其他输入参数后指定属性。

plot3(ax,___)在目标坐标区上显示绘图。将坐标区指定为上述任一语法中的第一个参数。

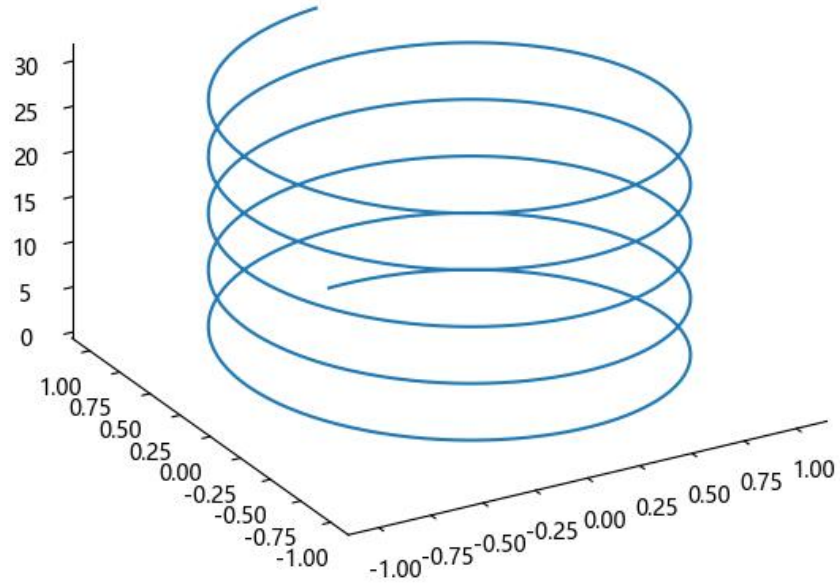
p = plot3(___) 返回一个 Line3D 对象或 Line3D 对象数组。创建绘图后，使用 p 修改该绘图的属性。

C. 示例

(1). 绘制三维螺旋图

将 t 定义为由介于 0 和 10π 之间的值组成的向量。将 st 和 ct 定义为正弦和余弦值向量。然后绘制 st、ct 和 t。

```
t = 0:pi/50:10*pi;
st = sin.(t);
ct = cos.(t);
l=plot3(st,ct,t);
```



(2). 绘制多个线条

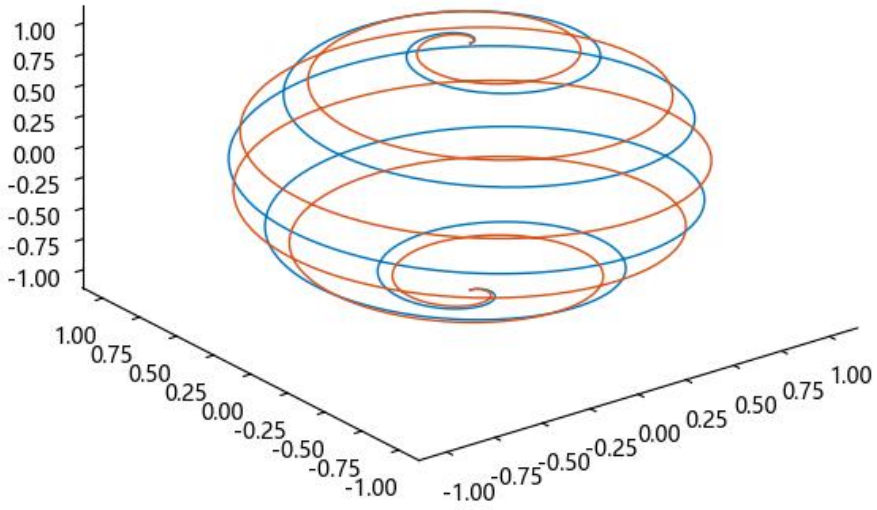
创建两组 x、y 和 z 坐标。

```
t = 0:pi/500:pi;
xt1 = sin.(t).*cos.(10*t);
yt1 = sin.(t).*sin.(10*t);
zt1 = cos.(t);

xt2 = sin.(t).*cos.(12*t);
yt2 = sin.(t).*sin.(12*t);
zt2 = cos.(t);
```

调用 plot3 函数，并指定连续的 XYZ 三元组。

```
plot3(xt1,yt1,zt1,xt2,yt2,zt2)
```



(3). 根据矩阵创建线图

创建包含三行 x 坐标的矩阵 X。创建包含三行 y 坐标的矩阵 Y。

```
t = 0:pi/500:pi;
X1 = sin.(t) .* cos.(10*t);
X2 = sin.(t) .* cos.(12*t);
X3 = sin.(t) .* cos.(20*t);
X = [X1  X2  X3]

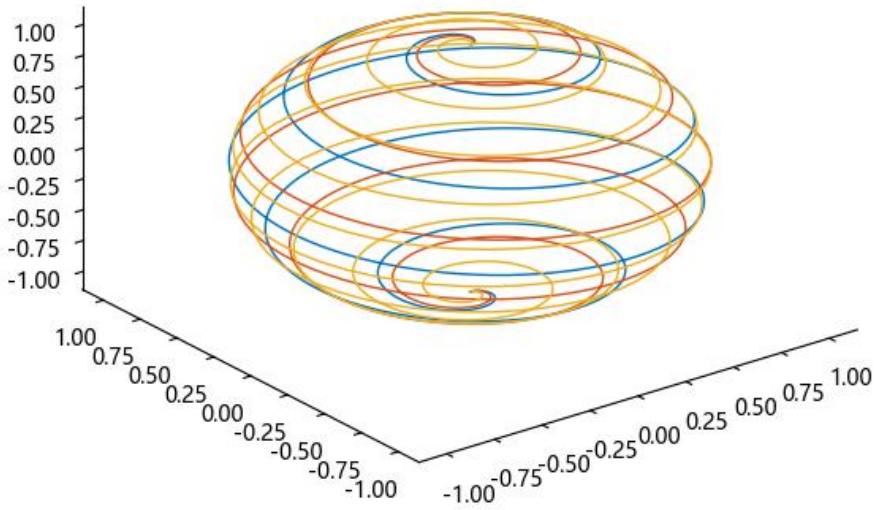
Y1 = sin.(t) .* sin.(10*t);
Y2 = sin.(t) .* sin.(12*t);
Y3 = sin.(t) .* sin.(20*t);
Y = [Y1  Y2  Y3]
```

创建矩阵 Z，其中包含所有三组坐标的 z 坐标。

```
Z = cos.(t);
```

在同一组坐标轴上绘制所有三组坐标。

```
lines = plot3(X,Y,Z)
```



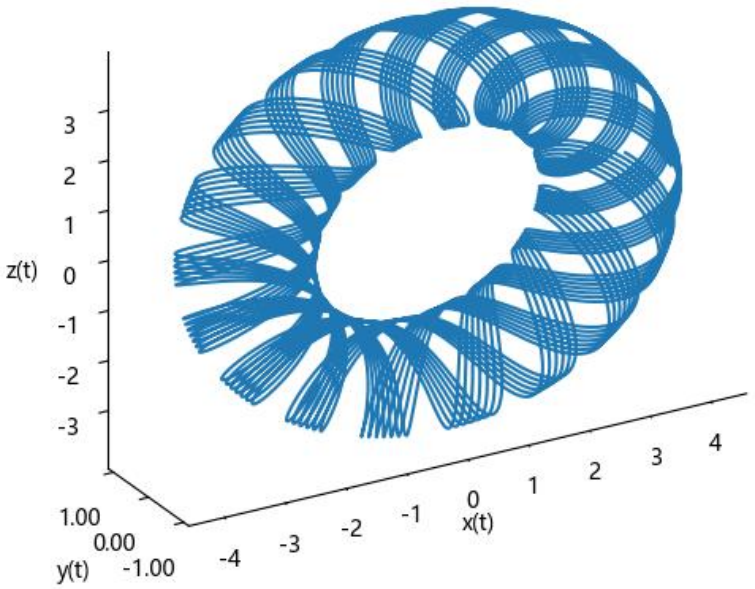
(4). 指定等间距刻度单位和轴标签

创建向量 xt、yt 和 zt。

```
t = 0:pi/500:40*pi;
xt = (3 .+ cos.(sqrt(32) .* t)) .* cos.(t);
yt = sin.(sqrt(32) .* t);
zt = (3 .+ cos.(sqrt(32) .* t)) .* sin.(t);
```

绘制数据，并使用 axis("equal") 命令沿每个轴等间距隔开刻度单位。然后为每个轴指定标签。

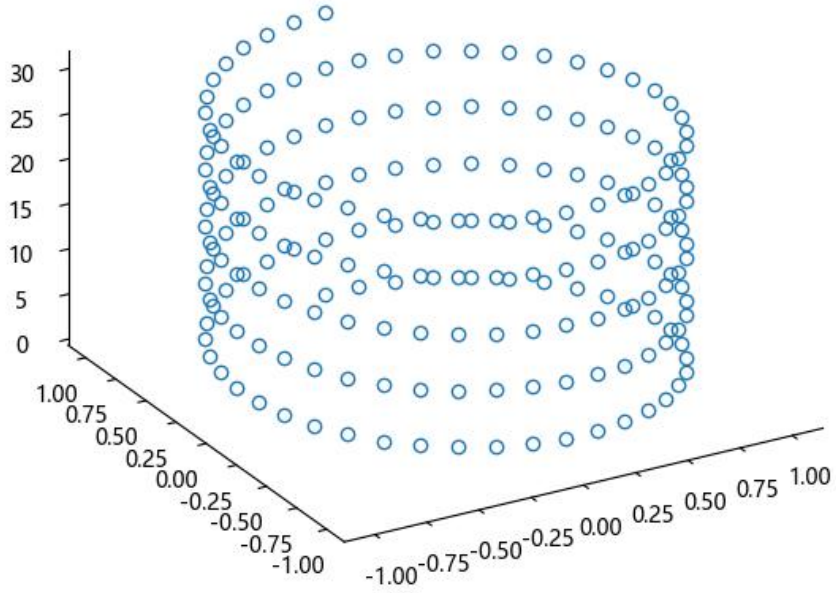
```
plot3(xt,yt,zt)
axis("equal")
xlabel("x(t)")
ylabel("y(t)")
zlabel("z(t)")
```



(5). 将点绘制为不带线的标记

创建向量 t、xt 和 yt，并使用圆形标记绘制这些向量中的点。

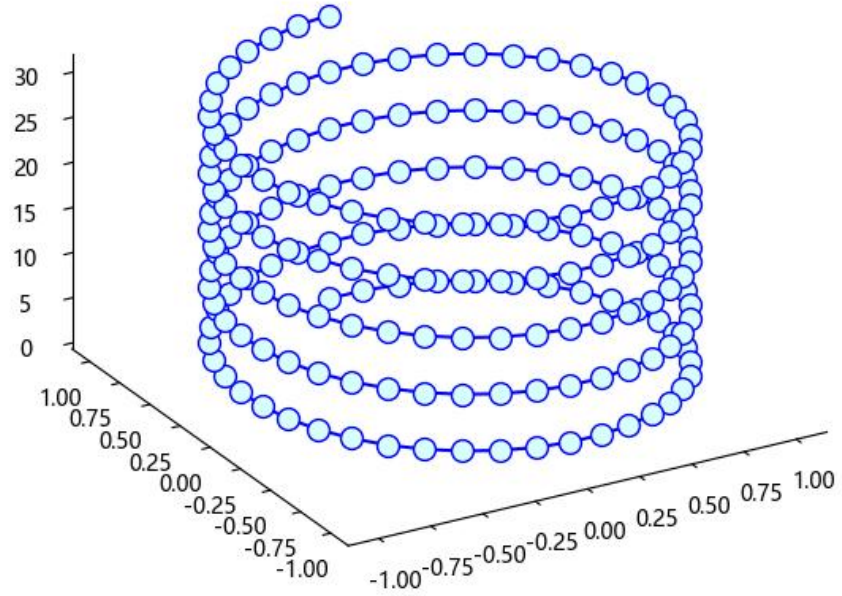
```
t = 0:pi/20:10*pi;
xt = sin.(t);
yt = cos.(t);
plot3(xt,yt,t,"o")
```



(6). 自定义颜色和标记

创建向量 t、xt 和 yt，并将这些向量中的点绘制为带 10 磅圆形标记的蓝线。使用十六进制颜色代码指定标记的填充颜色为浅蓝色。

```
t = 0:pi/20:10*pi;
xt = sin.(t);
yt = cos.(t);
plot3(xt,yt,t,"-o",color = "b", markersize=10, markerfacecolor=
"#D9FFFF")
```



(7). 指定线型

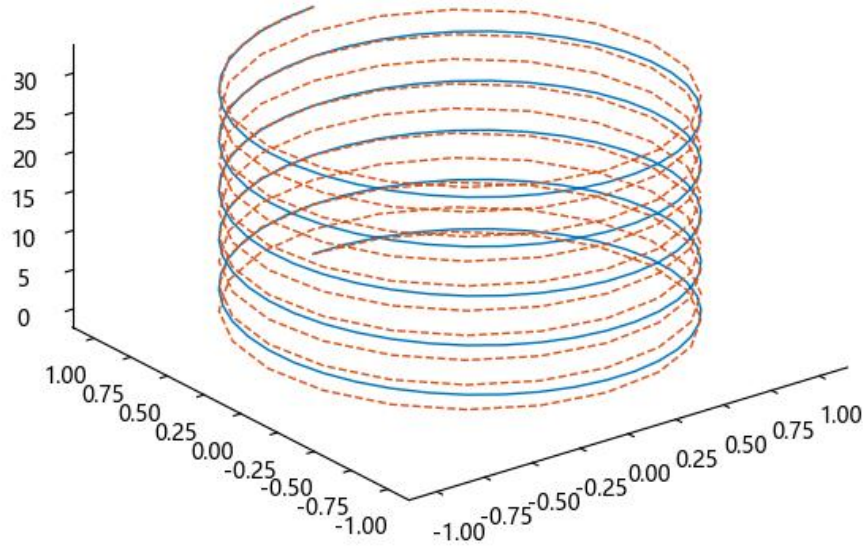
创建向量 t。然后使用 t 计算两组 x 和 y 值。

```
t = 0:pi/20:10*pi;
xt1 = sin.(t);
yt1 = cos.(t);

xt2 = sin.(2*t);
yt2 = cos.(2*t);
```

绘制这两组值。第一组使用默认线条，第二组使用虚线。

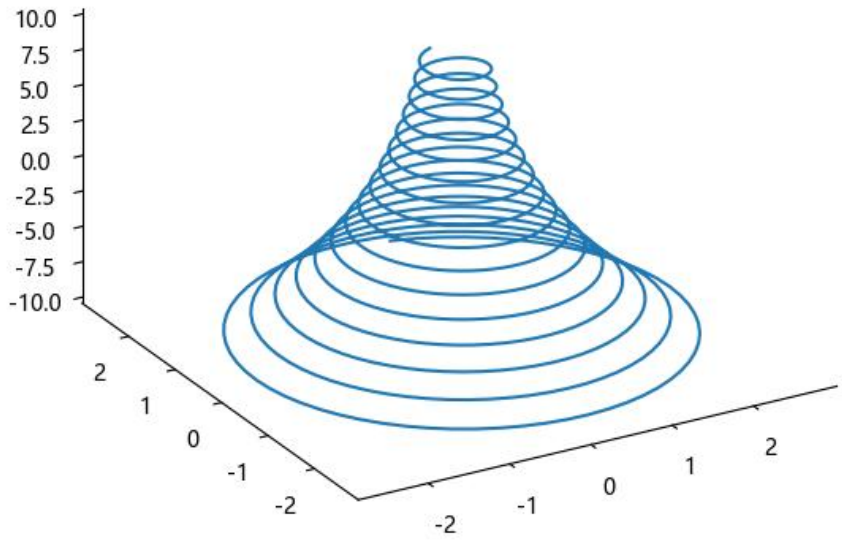
```
plot3(xt1,yt1,t,xt2,yt2,t,"--")
```



(8). 在绘图后修改线条

创建向量 t、xt 和 yt，并绘制这些向量中的数据。返回输出变量 p 中的图线条。

```
using TyPlot
t = LinRange(-10, 10, 1000);
xt = exp.(-t ./ 10) .* sin.(5 * t);
yt = exp.(-t ./ 10) .* cos.(5 * t);
p = plot3(xt, yt, t);
```



将线宽更改为 3。

```
p[1].set_linewidth(3)
```

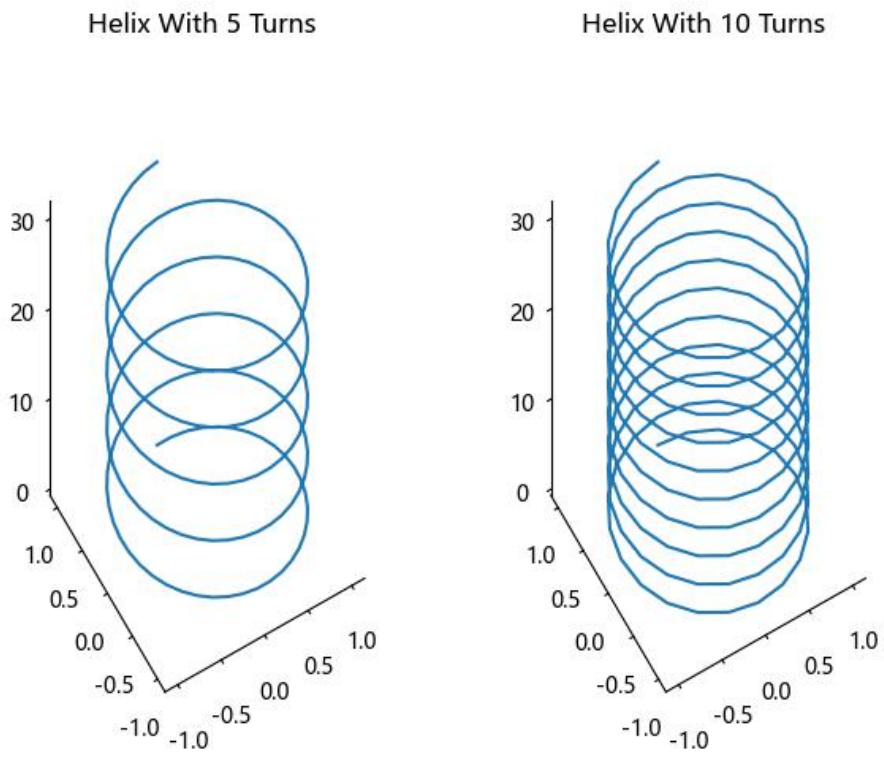
(9). 指定目标坐标区

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。通过将坐标区对象指定为 plot3 的第一个参数，在坐标区中创建单独的线图。

```
ax1 = subplot(1,2,1, projection="3d");
t = 0:pi/20:10*pi;
xt1 = sin.(t);
yt1 = cos.(t);
plot3(ax1,xt1,yt1,t)
title(ax1,"Helix With 5 Turns")

ax2 = subplot(1,2,2, projection="3d");
t = 0:pi/20:10*pi;
xt2 = sin.(2*t);
yt2 = cos.(2*t);
plot3(ax2,xt2,yt2,t)
```

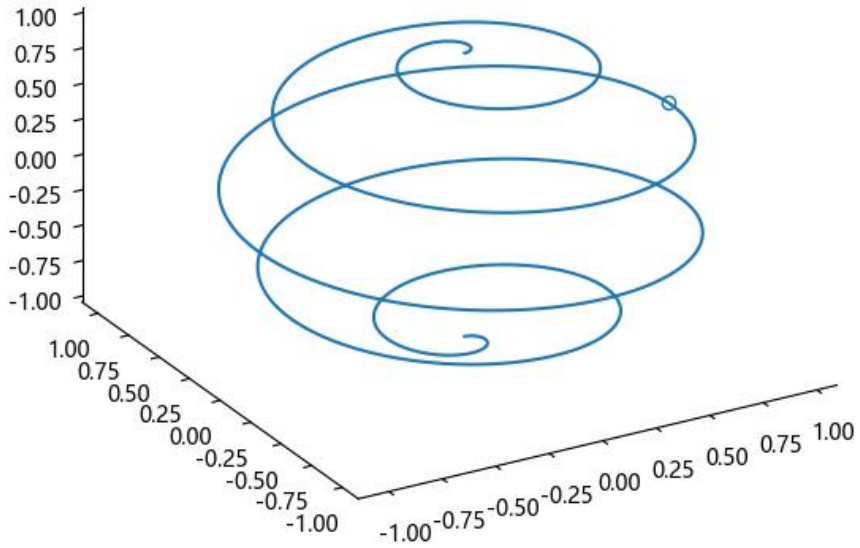
```
title(ax2,"Helix With 10 Turns")
```



(10). 绘制线条并在一个数据点上放置标记

创建向量 xt、yt 和 zt。绘制这些值，使用 fmt 参数指定具有圆形标记的实线。指定 markevery 属性以在第 200 个数据点上放置一个标记。

```
t = 0:pi/500:pi;
xt = sin.(t) .* cos.(10*t);
yt = sin.(t) .* sin.(10*t);
zt = cos.(t);
plot3(xt,yt,zt,"-o", markevery = [200])
```



(11). 洛伦兹函数

创建洛伦兹函数。

```
p=10;
r=28;
b=8/3;

x=[]
y=[]
z=[]
push!(x, 10)
push!(y, 1)
push!(z, 0)

h=0.01;
for i=1:1:10000
    K1=p*(y[i]-x[i]);
    L1=r*x[i]-y[i]-x[i]*z[i];
    M1=-b*z[i]+x[i]*y[i];
```

```
    K2=p*((y[i]+h/2*L1)-(x[i]+h/2*K1));
    L2=r*(x[i]+h/2*K1)-(y[i]+h/2*L1)-(x[i]+h/2*K1)*(z[i]+h/2*M1);
    M2=-b*(z[i]+h/2*M1)+(x[i]+h/2*K1)*(y[i]+h/2*L1);

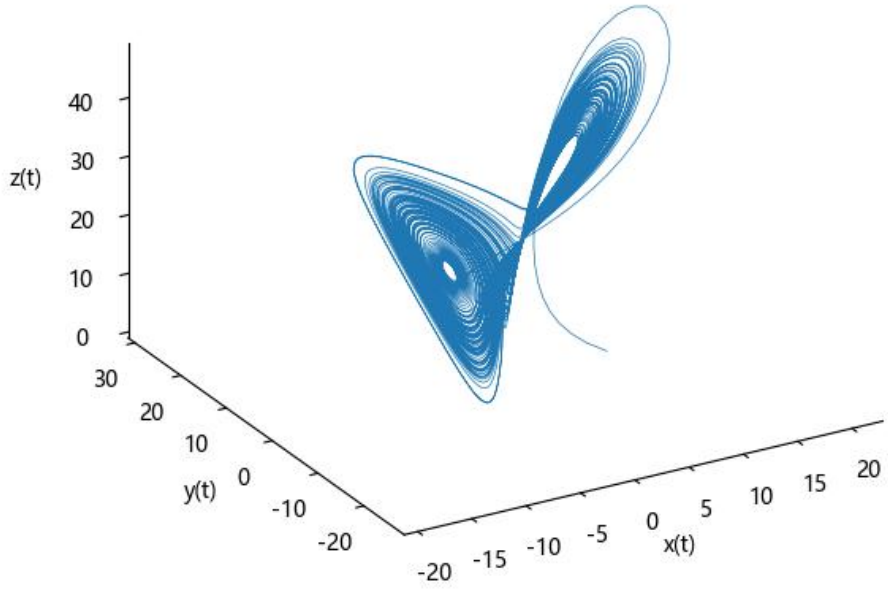
    K3=p*((y[i]+h/2*L2)-(x[i]+h/2*K1));
    L3=r*(x[i]+h/2*K1)-(y[i]+h/2*L2)-(x[i]+h/2*K1)*(z[i]+h/2*M2);
    M3=-b*(z[i]+h/2*M1)+(x[i]+h/2*K1)*(y[i]+h/2*L2);

    K4=p*((y[i]+h*L3)-(x[i]+h*K3));
    L4=r*(x[i]+h*K3)-(y[i]+h*L3)-(x[i]+h*K3)*(z[i]+h*M3);
    M4=-b*(z[i]+h*M3)+(x[i]+h*K3)*(y[i]+h*L3);

    push!(x, x[i]+h/6*(K1+2*K2+2*K3+K4))
    push!(y, y[i]+h/6*(L1+2*L2+2*L3+L4))
    push!(z, z[i]+h/6*(M1+2*M2+2*M3+M4))
end

figure(1);
plot3(x,y,z, linewidth=0.5);
xlabel("x(t)"),ylabel("y(t)"),zlabel("z(t)");
title("Lorenz 吸引子图");
```

Lorenz吸引子图



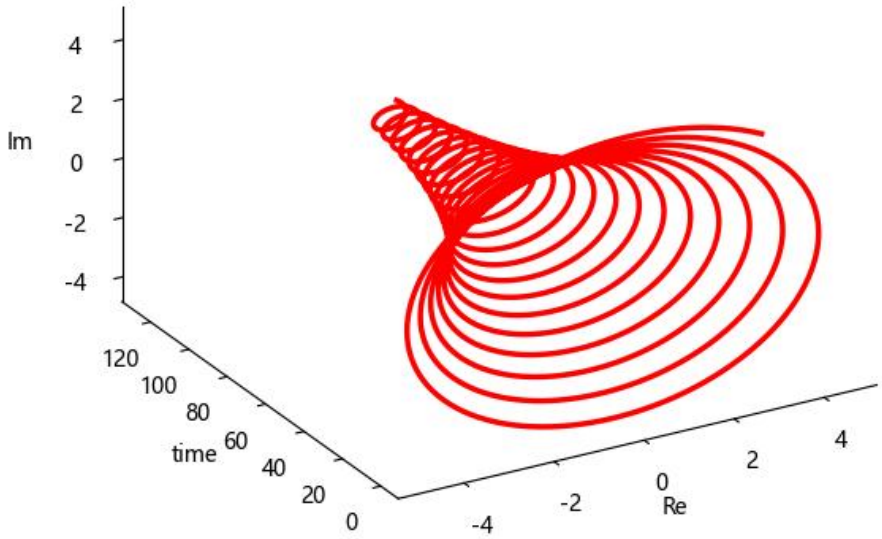
(12). 复指数函数

创建复指数函数。

```
using TyPlot
C = 3 + 4im;
r = -0.02;
w = 1;
a = r + im * w
t = 0:0.05:(40 * pi);
y = C .* exp.(a .* t);
Re = real(y);
Im = imag(y);
figure();
plot3(Re, t, Im, "-r"; linewidth=2.5);

xlabel("Re")
ylabel("time")
zlabel("Im")
title("(a+bj)" * raw"${e^{({r+j{\omega}})t}}$")
```

$$(a+bj)e^{(r+j\omega)t}$$



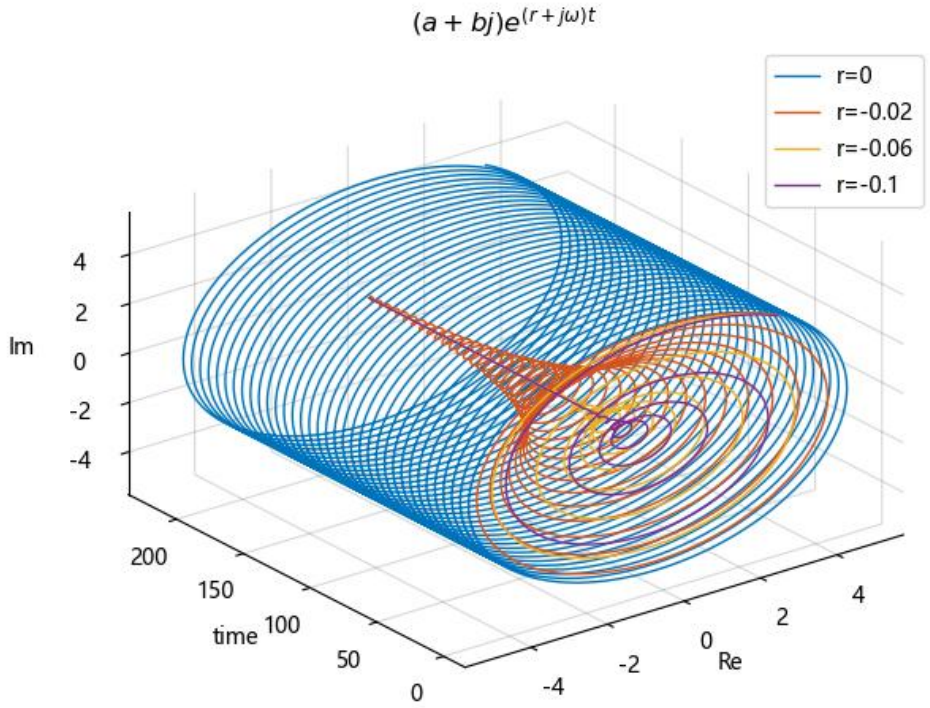
(13). 多个复指数函数创建

包含多个复指数函数的绘图。

```
C=3+4im;
r=[0 -0.02 -0.06 -0.1];
w=1;
t=0:0.05:70*pi;
a=r .+ im*w

for m=1:4
    y = C .* exp.(a[m].*t);
    Re = real(y);
    Im = ima g(y);
    plot3(Re,[t...],Im,linewidth=1);
    hold("on")
end

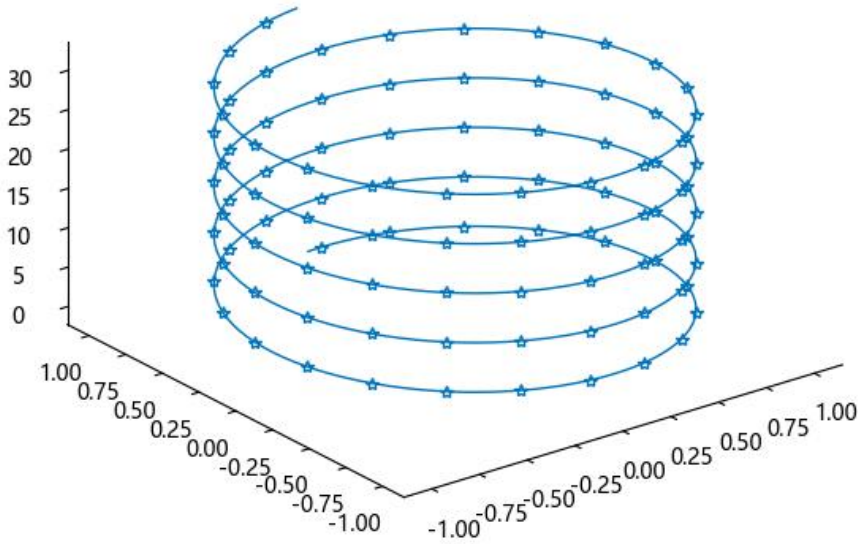
grid("on")
xlabel("Re")
ylabel("time")
zlabel("Im")
title(raw"${(a+bj)e^{({r+j{\omega}})t}}$")
legend(["r=0","r=-0.02","r=-0.06","r=-0.1"])
hold("off")
```



(14). 在特定的数据点显示标记

将 t 定义为由介于 0 和 10π 之间的值组成的向量。将 st 和 ct 定义为正弦和余弦值向量。然后绘制 st、ct 和 t，并将 markevery 属性设置为名称-值对组的形式，创建一个三维线图并每隔四个数据点显示一个标记。

```
t = 0:(pi / 50):(10 * pi);
st = sin.(t);
ct = cos.(t);
l = plot3(st, ct, t; marker="*", markevery=1:5:length(st));
```



stairs

阶梯图

A. 语法

```
stairs(Y)
stairs(X,Y)
stairs(___,Fmt)
stairs(___,Key=Value)
```

```
stairs(ax,___)

h = stairs(___)
```

B. 说明

stairs(Y) 绘制 Y 中元素的阶梯图。

如果 Y 为向量，则 stairs 绘制一个线条。

如果 Y 为矩阵，则 stairs 为每个矩阵列绘制一个线条。

stairs(X,Y)在 Y 中由 X 指定的位置绘制元素。输入 X 和 Y 必须是相同大小的向量或矩阵。另外，X 可以是行或列向量，Y 必须是包含 length(X)行的矩阵。

stairs(___,Fmt)指定线型、标记符号和颜色。例如，":*r"指定带星号标记的红色点线。可以将此选项与前面语法中的任何输入参数组合一起使用。

stairs(___,Key=Value)使用一个或多个名称-值对组参数修改阶梯图。例如，marker="o",markersize=8 指定大小为 8 磅的圆形标记。

stairs(ax,___)将图形绘制到 ax 指定的坐标区中，而不是当前坐标区(gca)中。选项 ax 可以位于前面的语法中的任何输入参数组合之前。

h=stairs(___)返回一个或多个 Line 对象。在创建特定 Line 对象后，使用 h 更改该对象的属性。

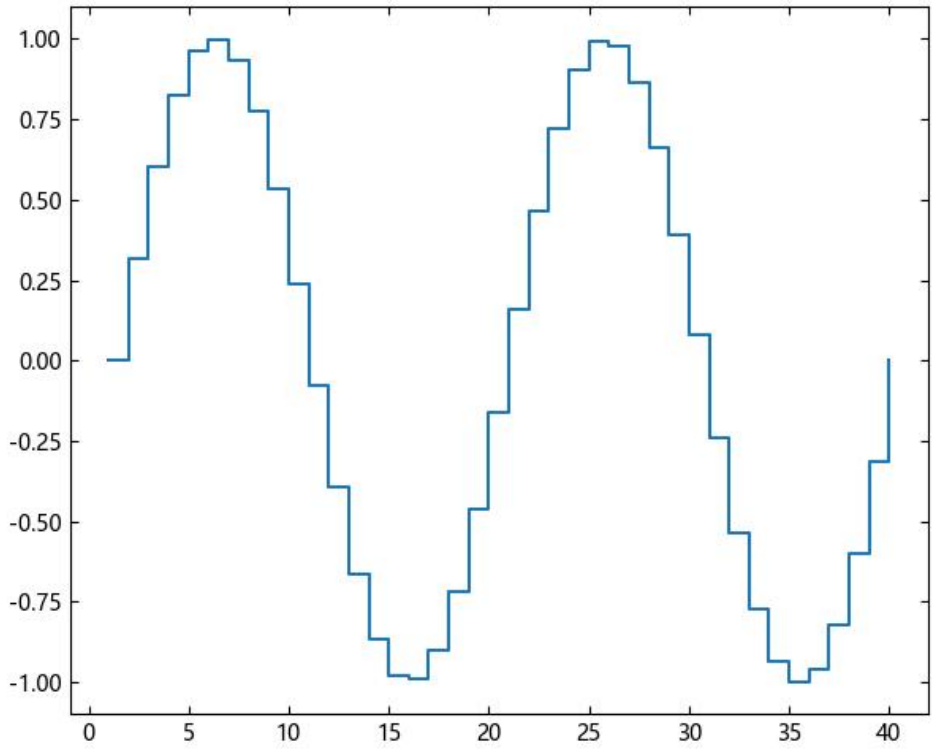
C. 示例

(1). 绘制单数据序列

创建在 0 到 4π 区间内的 40 个均匀分布的值处计算的正弦阶梯图。

```
X = LinRange(0, 4 * pi, 40);
Y = sin.(X);

figure()
stairs(Y)
```

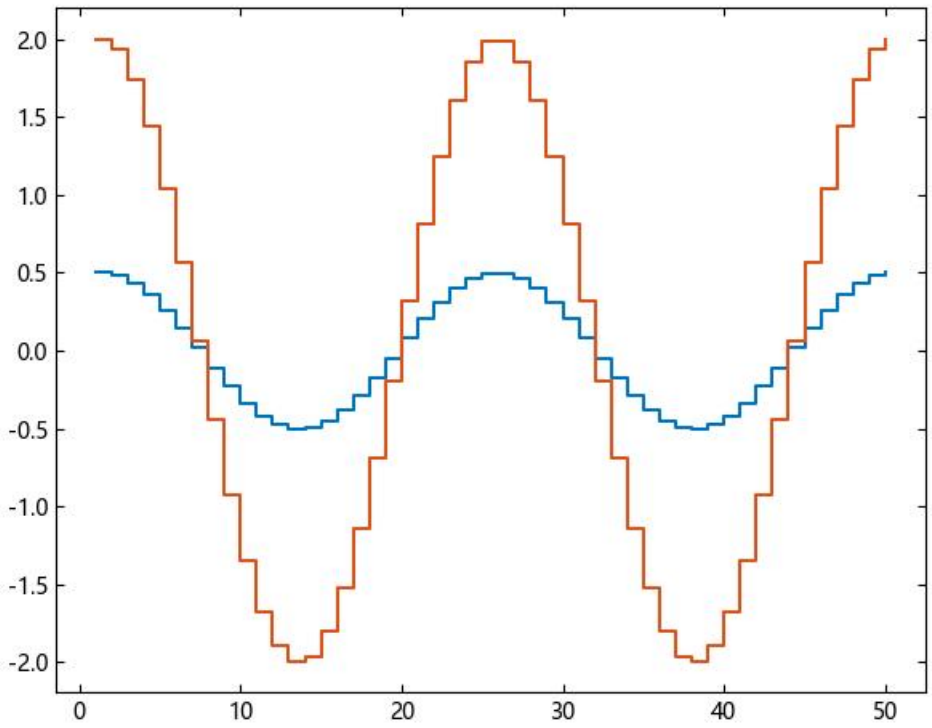


Y 的长度自动确定并生成 x 轴刻度。

(2). 绘制多个数据序列

创建在 0 到 4π 区间内的 50 个均匀分布的值处计算的两个余弦函数阶梯图。

```
X = LinRange(0, 4 * pi, 50)';
Y = [0.5 * cos.(X), 2 * cos.(X)];
figure()
stairs(Y)
```

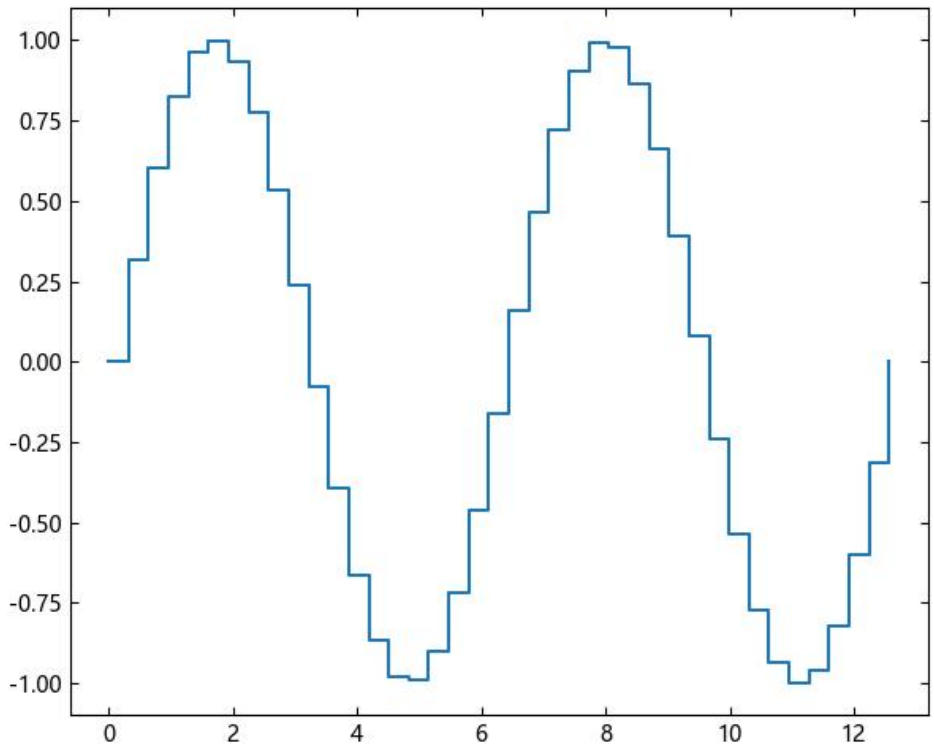


Y 中的行数自动确定并生成 x 轴刻度。

(3). 在指定的 x 值位置绘制单个数据序列

创建在 0 到 4π 区间内均匀分布的值处计算的正弦波阶梯图。指定该图的 x 值集。

```
X = LinRange(0, 4 * pi, 40);
Y = sin.(X);
figure()
stairs(X, Y)
```

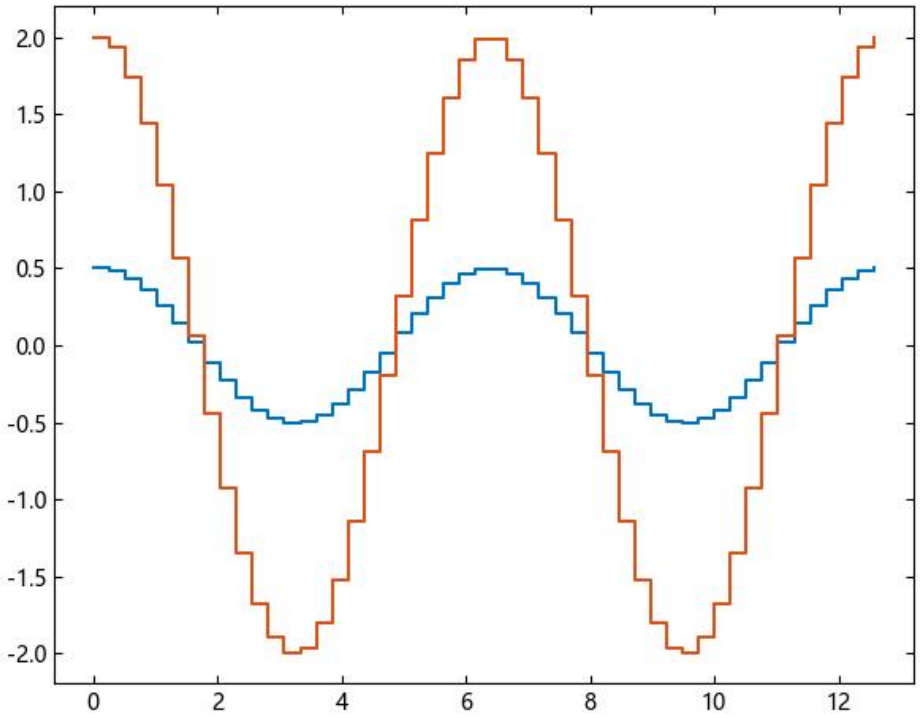


绘制 Y 中的项对 X 中的对应项的图。

(4). 在指定的 x 值处绘制多个数据序列

创建在 0 到 4π 区间内的均匀分布的值处计算的两个余弦波阶梯图。指定该图的 x 值集。

```
X = LinRange(0, 4 * pi, 50)';
Y = [0.5 * cos.(X), 2 * cos.(X)];
figure()
stairs(X, Y)
```

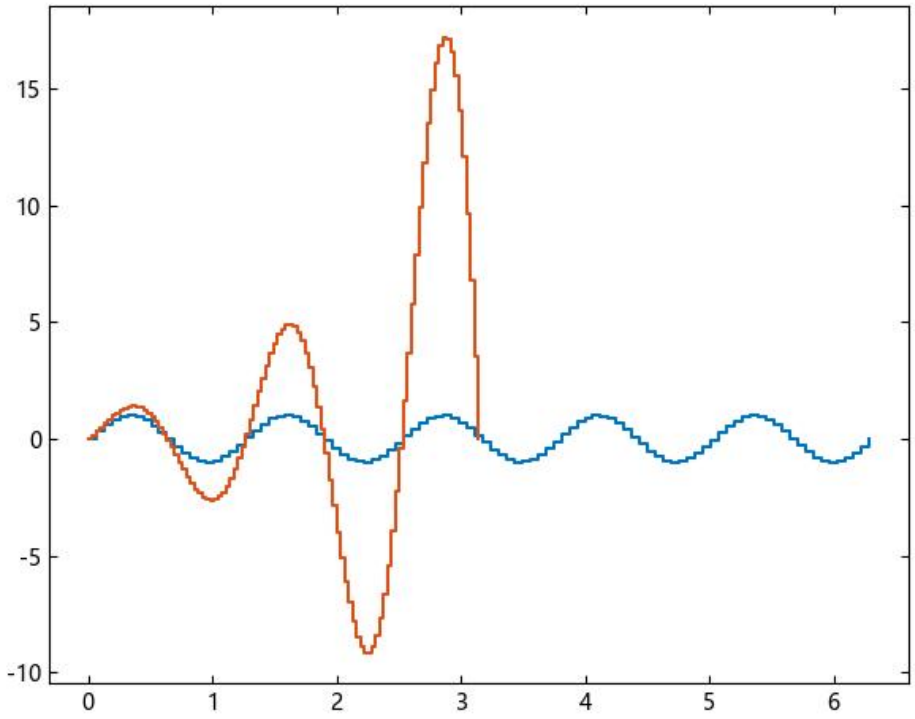


第一个向量输入 X 确定两个数据序列的 x 轴位置。

(5). 在唯一的 x 值集合处绘制多个数据

序列创建在不同的值处计算的两个余弦波的阶梯图。指定用于绘制每个数据序列的唯一 x 值集。

```
x1 = LinRange(0, 2 * pi, 100)';
x2 = LinRange(0, pi, 100)';
X = [x1, x2];
Y = [sin.(5 .* x1), exp.(x2') .* sin.(5 .* x2)'];
figure()
stairs(X, Y)
```

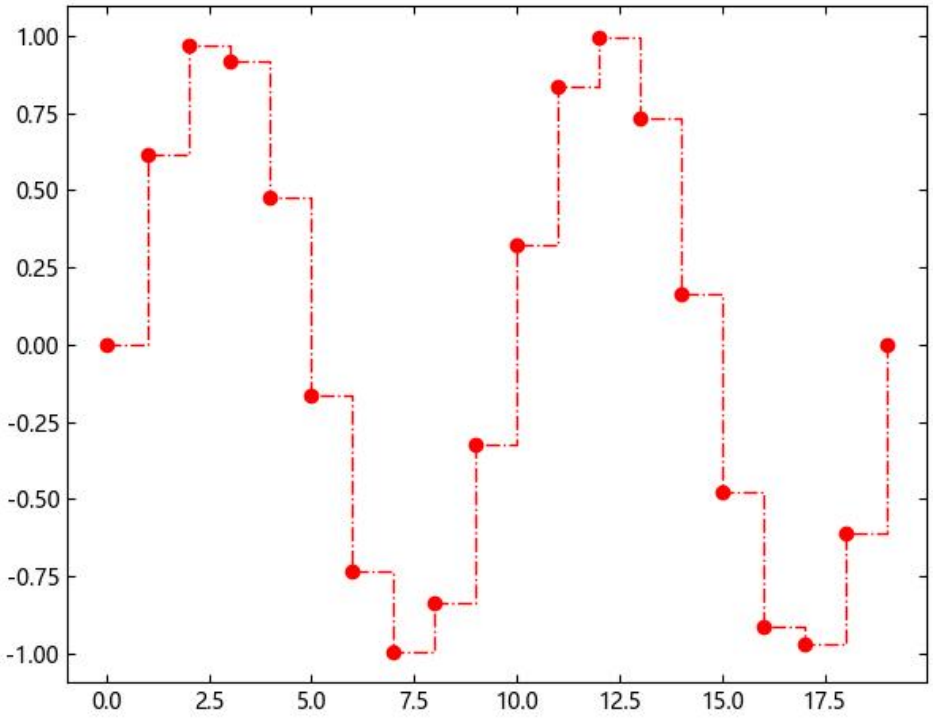


绘制 X 的每一列对 Y 的对应列的图。

(6). 指定线型、标记符号和颜色

创建一个阶梯图，将线型设置为点划线，将标记符号设置为圆，将颜色设置为红色。

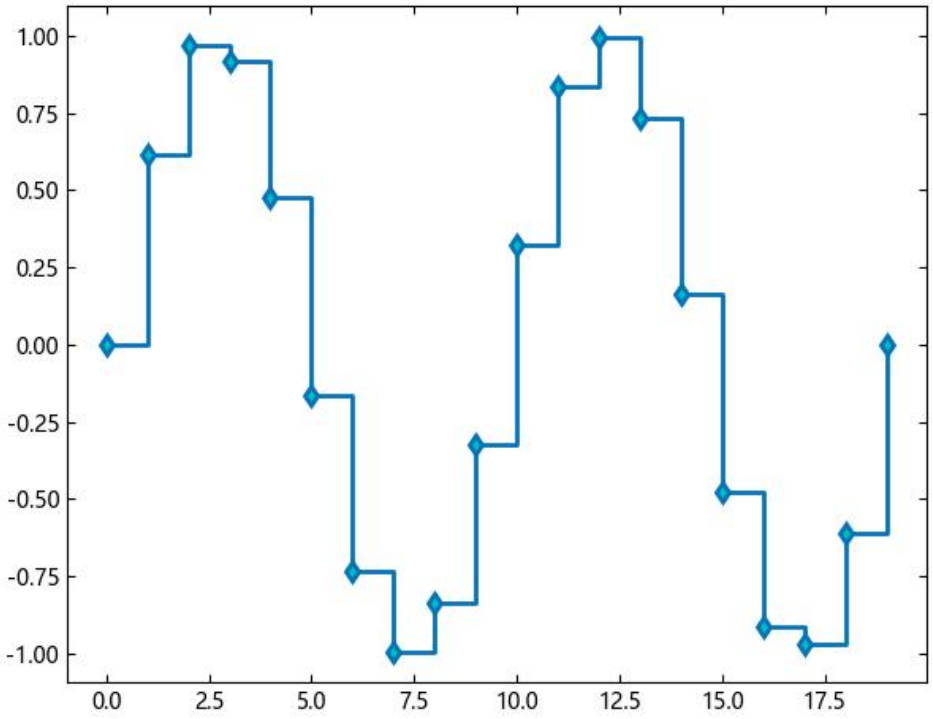
```
X = LinRange(0, 4 * pi, 20);
Y = sin.(X);
figure()
stairs(Y, "-.or")
```



(7). 指定其他样式选项

使用 Key=Value 对组参数创建一个阶梯图并将线宽设置为 2，将标记符号设置为菱形，并将标记面颜色设置为青色。

```
X = LinRange(0, 4 * pi, 20);
Y = sin.(X);
figure()
stairs(Y; linewidth=2, marker="d", markerfacecolor="c")
```



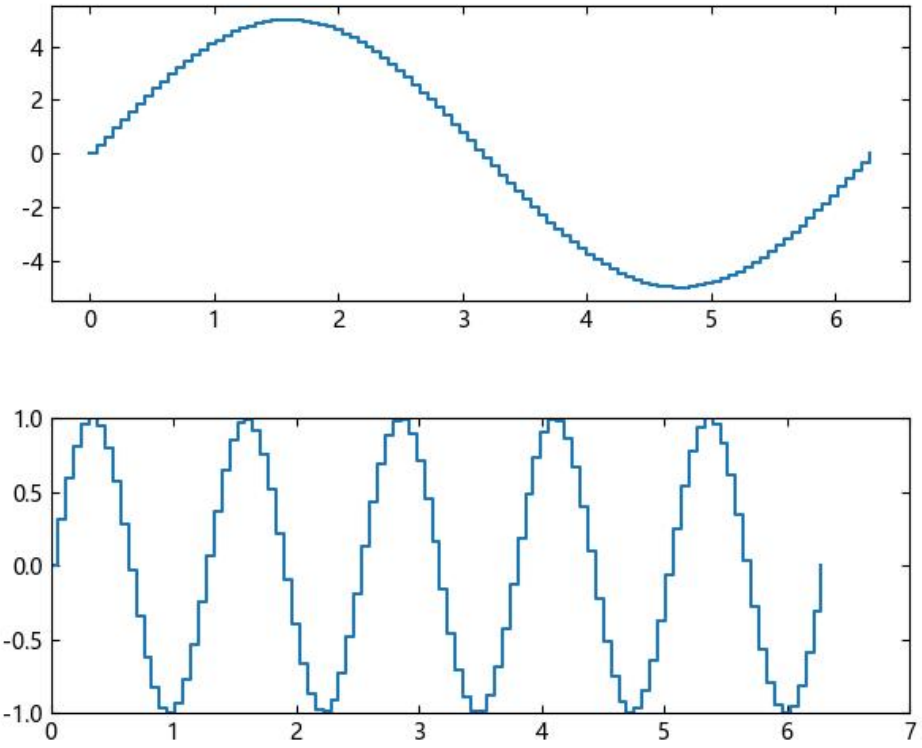
(8). 为阶梯图指定坐标区

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。通过将坐标区对象指定为 stairs 的第一个参数，在坐标区中创建单独的阶梯图。

```
x = LinRange(0, 2 * pi, 100);
y1 = 5 .* sin.(x);
y2 = sin.(5 * x);

ax1 = subplot(2, 1, 1)
stairs(ax1, x, y1)
xlim(0, 7)
ylim(-5, 5)

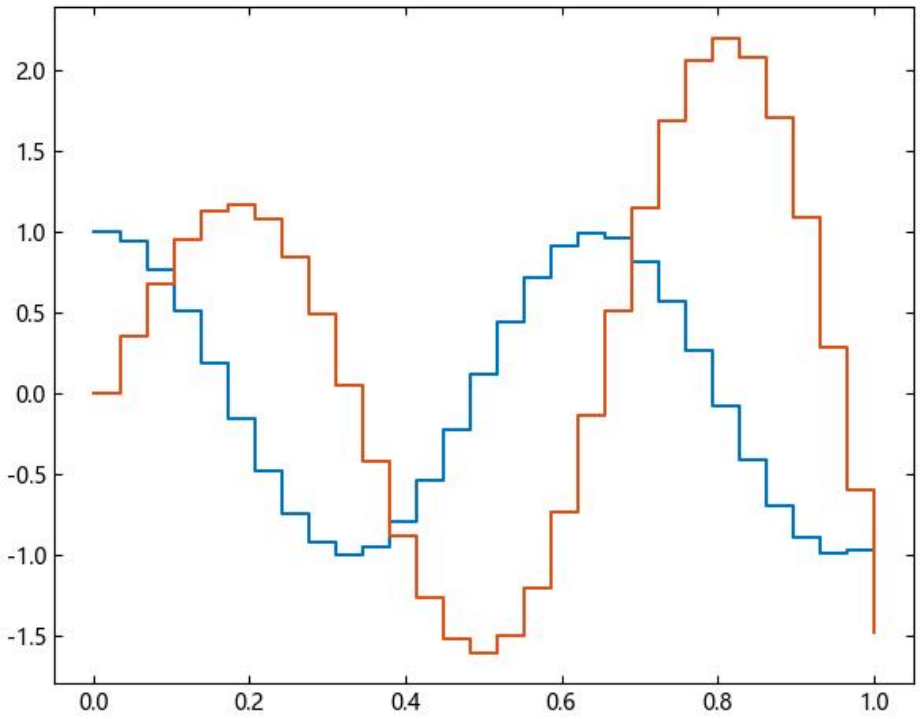
ax2 = subplot(2, 1, 2)
stairs(ax2, x, y2)
xlim(0, 7)
ylim(-1, 1)
```



(9). 创建并修改阶梯图

创建两个数据序列的阶梯图并返回两个阶梯图对象。

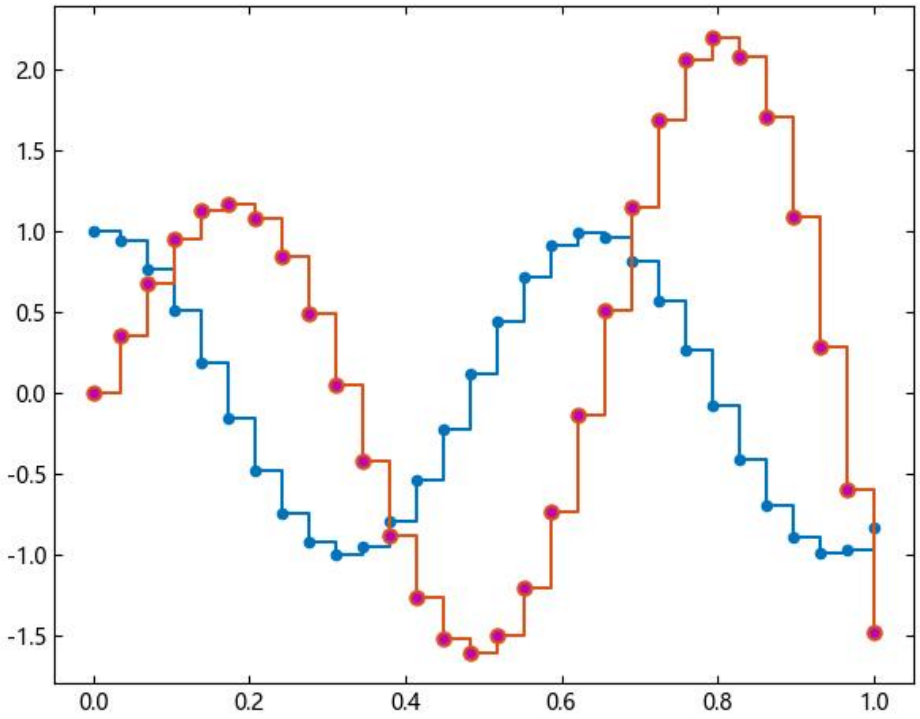
```
X = LinRange(0, 1, 30)';
Y = [cos.(10 * X), exp.(X') .* sin.(10 * X)'];
h = stairs(X, Y);
```



对第一个数据序列使用小圆形标记。对第二个序列使用填充品红色的圆形。使用圆点表示法设置属性。

```
h[1].set_marker("o")
h[1].set_markersize(4)

h[2].set_marker("o")
h[2].set_markerfacecolor("m")
```



errorbar

含误差条的线图

A. 语法

```
errorbar(y,err)
errorbar(x,y,err)

errorbar(___,Key=Value)
errorbar(ax,___)
e = errorbar(___)
```

B. 说明

errorbar(y,err) 创建 y 中数据的线图，并在每个数据点处绘制一个垂直误差条。err 中的值确定数据点上方和下方的每个误差条的长度，因此，总误差条长度是 err 值的两倍。

errorbar(x,y,err)绘制 y 对 x 的图，并在每个数据点处绘制一个垂直误差条。

errorbar(___,Key=Value)使用一个或多个名称-值对组参数修改线和误差条的

外观。例如，capsize=10 将每个误差条末端的端盖长度设置为 10 磅。

errorbar(ax,___) 在由 ax 指定的坐标区（而不是当前坐标区）中创建绘图。指定坐标区作为第一个输入参数。

当 y 为向量时，e = errorbar(___) 返回一个 ErrorBar 对象。如果 y 是矩阵，它将为 y 中的每一列返回一个 ErrorBar 对象。可在创建特定的 ErrorBar 对象后使用 e 修改其属性。

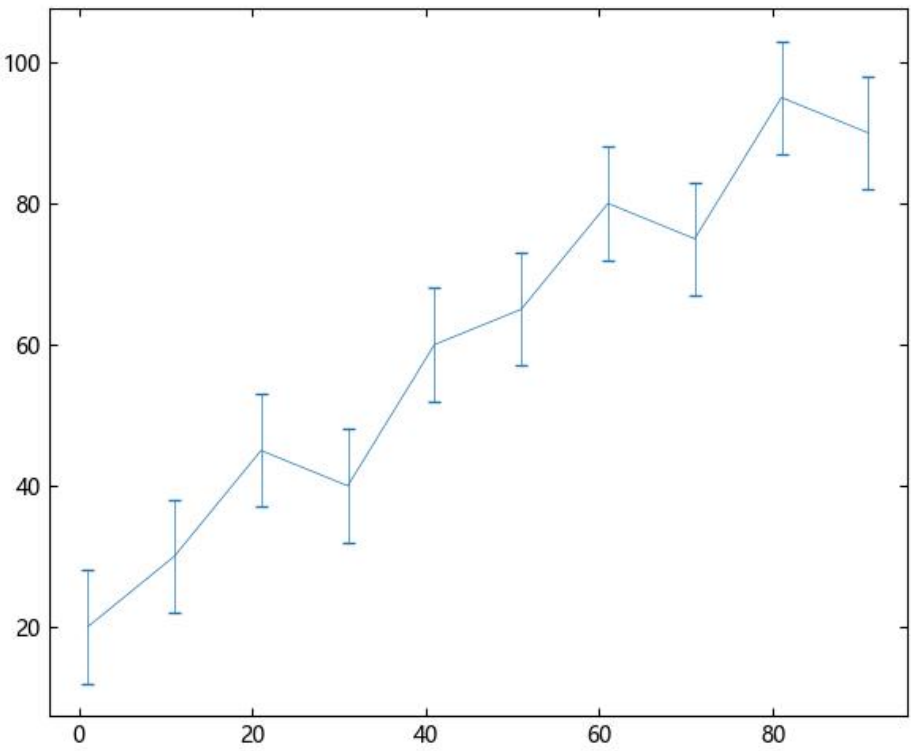
C. 示例

(1). 绘制长度相等的垂直误差条

创建向量 x 和 y。绘制 y 对 x 的图。在每个数据点处，显示长度相等的垂直误差条。

```
x = 1:10:100
y = [20, 30, 45, 40, 60, 65, 80, 75, 95, 90]
err = 8*ones(size(y))

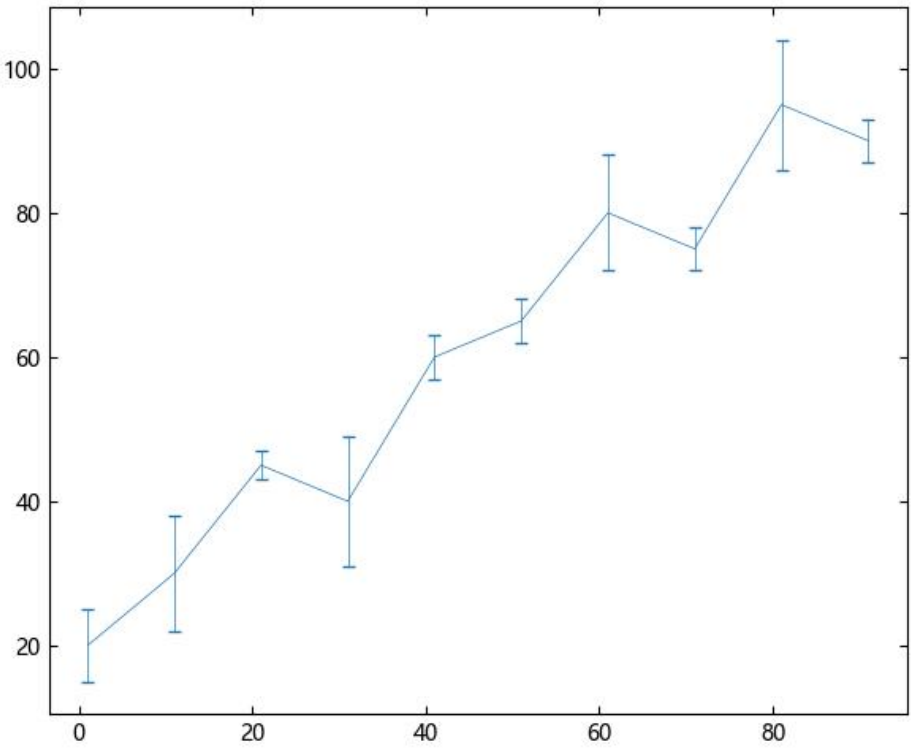
errorbar(x, y, err)
```



(2). 绘制不同长度的垂直误差条

在每个数据点处创建带误差条的线图。改变误差条的长度。

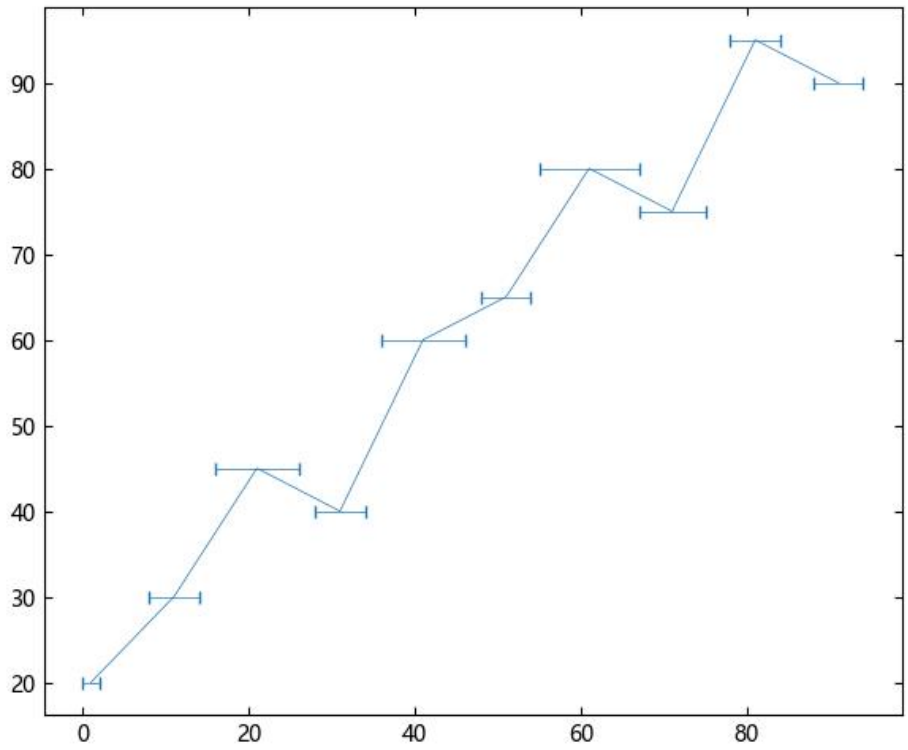
```
x = 1:10:100
y = [20, 30, 45, 40, 60, 65, 80, 75, 95, 90]
err = [5, 8, 2, 9, 3, 3, 8, 3, 9, 3]
errorbar(x, y, err)
```



(3). 绘制水平误差条

在每个数据点处创建带水平误差条的线图。

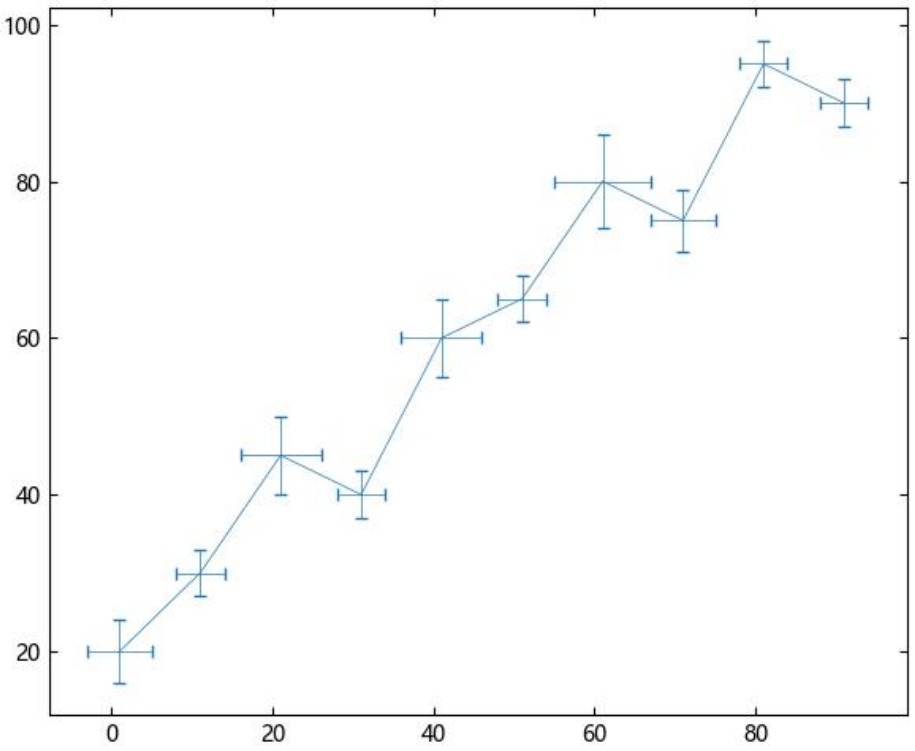
```
x = 1:10:100
y = [20, 30, 45, 40, 60, 65, 80, 75, 95, 90]
err = [1, 3, 5, 3, 5, 3, 6, 4, 3, 3]
errorbar(x, y, err; ornt="horizontal")
```



(4). 绘制垂直和水平误差条

在每个数据点处创建带垂直和水平误差条的线图。

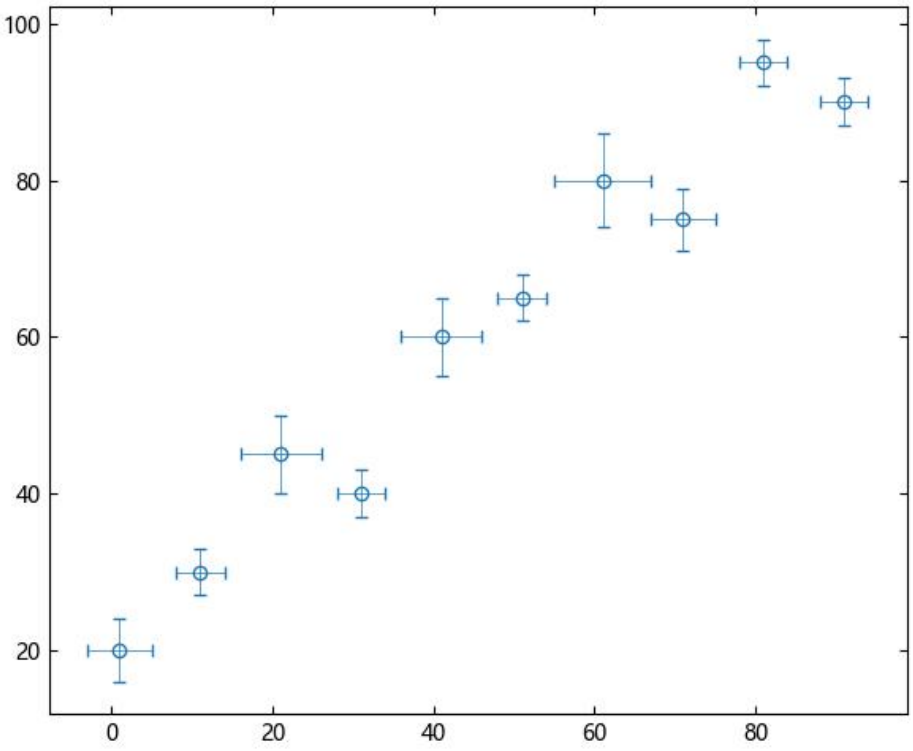
```
x = 1:10:100
y = [20, 30, 45, 40, 60, 65, 80, 75, 95, 90]
err = [4, 3, 5, 3, 5, 3, 6, 4, 3, 3]
errorbar(x, y, err; ornt="both")
```



(5). 绘制不带线条的误差条

绘制向量 y 对 x 的图。在每个数据点处显示一个圆形标记以及垂直和水平误差条。通过省略 fmt 输入参数的线型选项，不显示连接数据点的线。

```
x = 1:10:100
y = [20, 30, 45, 40, 60, 65, 80, 75, 95, 90]
err = [4, 3, 5, 3, 5, 3, 6, 4, 3, 3]
errorbar(x, y, err; ornt="both", fmt="o")
```

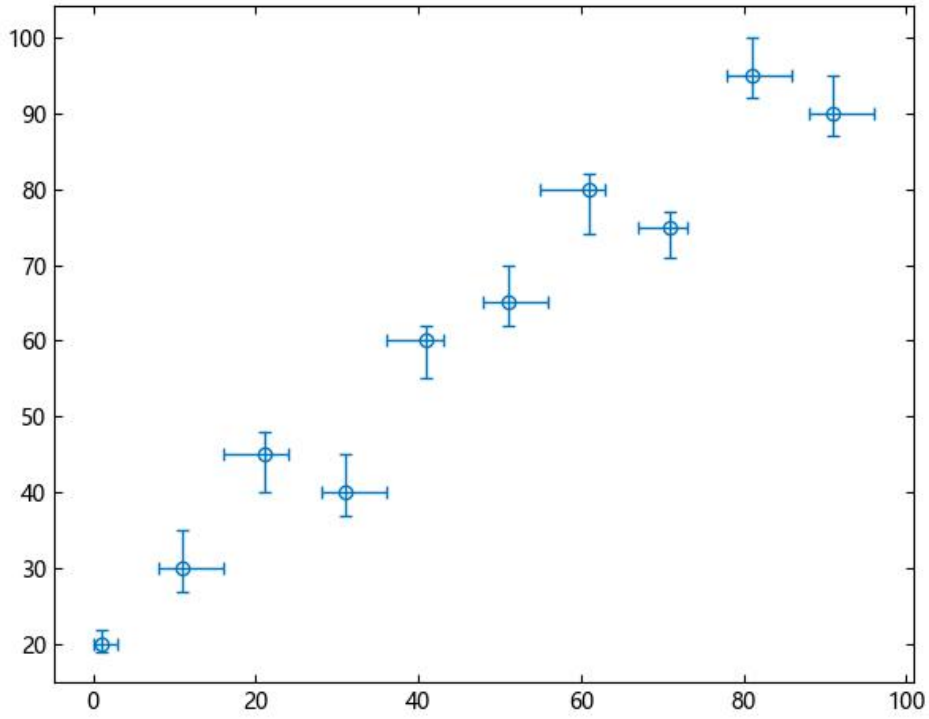


(6). 控制所有方向上的误差条长度

在每个数据点处显示垂直和水平误差条。使用 yneg 和 ypos 输入参数选项分别控制垂直误差条下部和上部的长度。使用 xneg 和 xpos 输入参数选项分别控制水平误差条左侧和右侧的长度。

```
x = 1:10:100
y = [20 30 45 40 60 65 80 75 95 90]
yneg = [1 3 5 3 5 3 6 4 3 3]
ypos = [2 5 3 5 2 5 2 2 5 5]
xneg = [1 3 5 3 5 3 6 4 3 3]
xpos = [2 5 3 5 2 5 2 2 5 5]

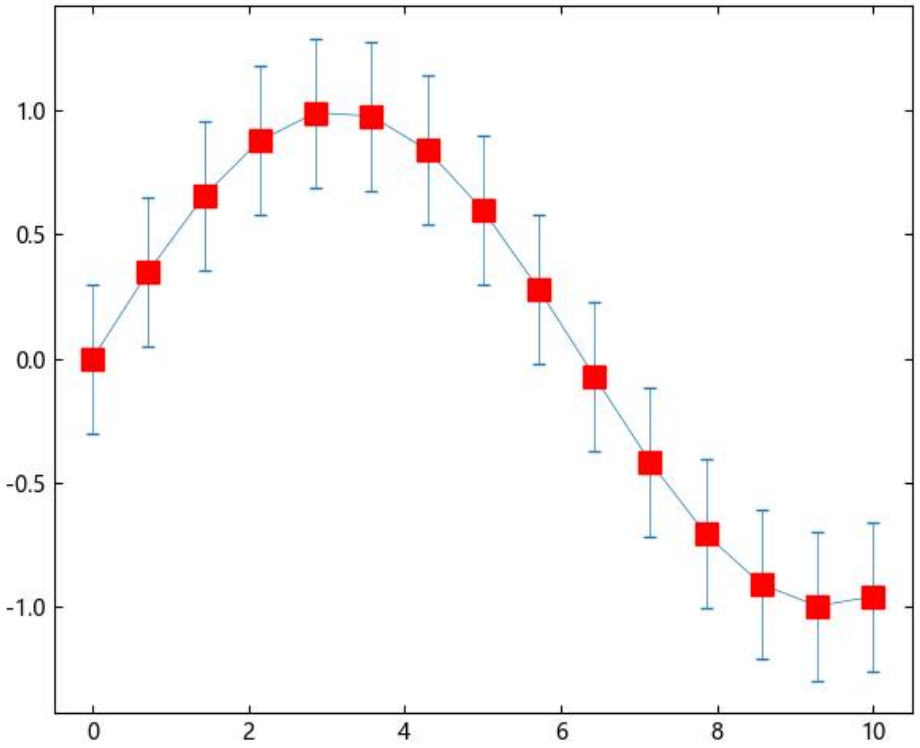
errorbar(x, y, yneg, ypos, xneg, xpos; fmt="o")
```



(7). 向每个数据点添加彩色标记

创建带误差条的线图。在每个数据点处显示标记。使用名称-值对组参数控制标记的外观。使用 markersize 以磅为单位指定标记大小。使用 markeredgecolor 和 markerfacecolor 分别指定标记轮廓和内部颜色。将颜色设置为颜色名称字符向量（例如 "red"）或 RGB 三元组。

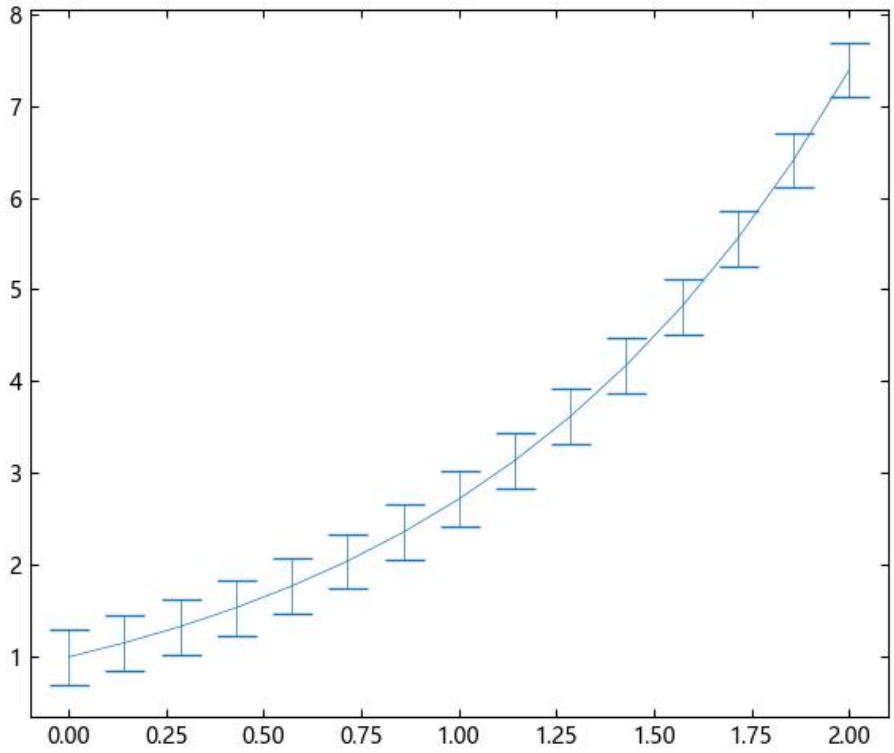
```
x = LinRange(0, 10, 15)
y = sin.(x / 2)
err = 0.3 * ones(size(y));
errorbar(x, y, err; fmt="-s", markersize=10, markeredgecolor="r",
markerfacecolor="r")
```



(8). 控制误差条端盖大小

通过将 capsize 属性设置为以磅为单位的正值，控制每个误差条末端的端盖大小。

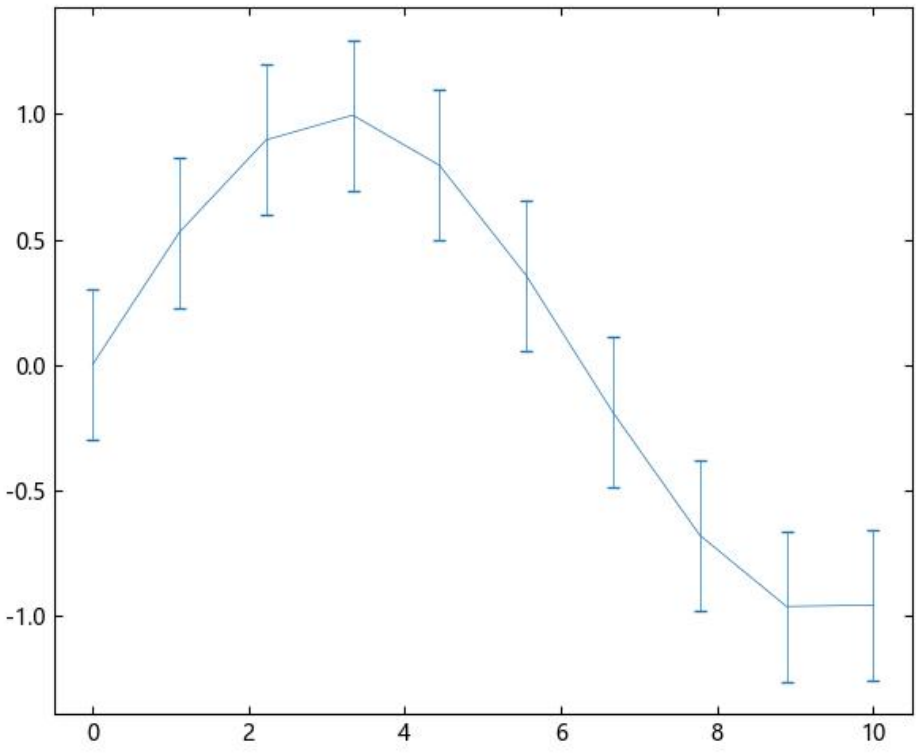
```
x = LinRange(0, 2, 15);
y = exp.(x);
err = 0.3 * ones(size(y));
errorbar(x, y, err; capsize=18)
```



(9). 创建后修改误差条

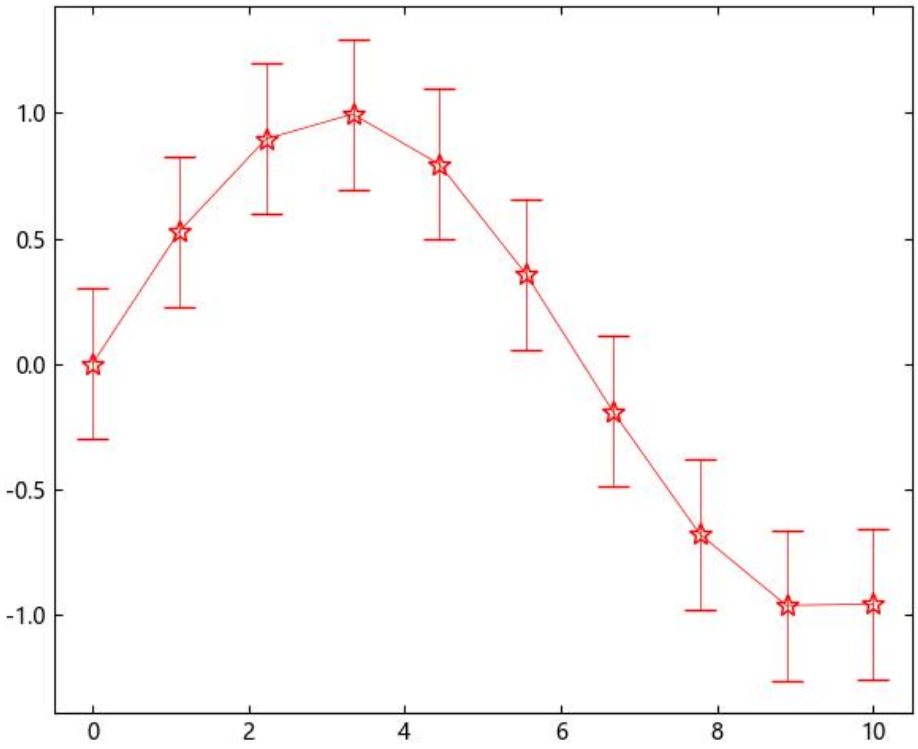
创建带误差条的线图。将误差条对象赋给变量 e。

```
x = LinRange(0, 10, 10);
y = sin.(x / 2);
err = 0.3 * ones(size(y));
```



在创建误差条对象后，使用 e 访问其属性。

```
e = errorbar(x, y, err; capsize=15, ecolor="r")
e[1].set_marker("*")
e[1].set_markersize(10)
e[1].set_color("r")
```



ezplot

易用的函数绘图函数

A. 语法

```
ezplot(fun)
ezplot(fun,[xmin,xmax])
ezplot(fun2,[xmin,xmax,ymin,ymax])
ezplot(funx,funy)
ezplot(funx,funy,[tmin,tmax])
ezplot(ax,...)
h = ezplot(...)
```

B. 说明

ezplot(fun) 绘制表达式 fun(x) 在默认定义域 -2π < x < 2π 上的图形，其中 fun(x) 仅是 x 的显函数。

fun 可以是函数句柄或字符串。

ezplot(fun,[xmin,xmax]) 绘制 fun(x) 在以下域上的图形：xmin < x < xmax。

ezplot(fun2,[xmin,xmax,ymin,ymax]) 在 xmin < x < xmax 和 ymin < y < ymax 域中绘制 fun2(x,y) = 0。

ezplot(funx,funy) 绘制以参数定义的平面曲线 funx(t) 和 funy(t) 在默认域 0 < t < 2π 上的图形。

ezplot(funx,funy,[tmin,tmax]) 绘制 funx(t) 和 funy(t) 在 tmin < t < tmax 上的图形。

ezplot(ax,...) 将图形绘制到坐标区 ax 中，而不是当前坐标区 (gca) 中。

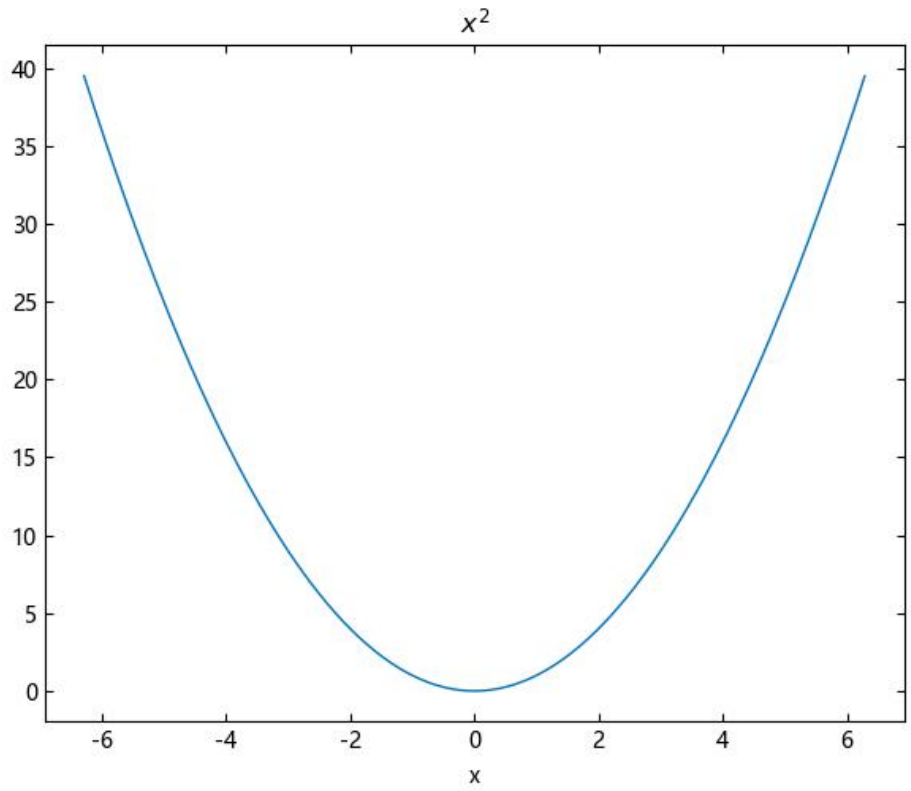
h = ezplot(...) 返回图形线条或等高线对象。

C. 示例

(1). 绘制显函数

在域 [−2π,2π] 中绘制显函数 。

```
ezplot(x -> x^2)
title(raw"$x^2$")
xlabel("x")
```

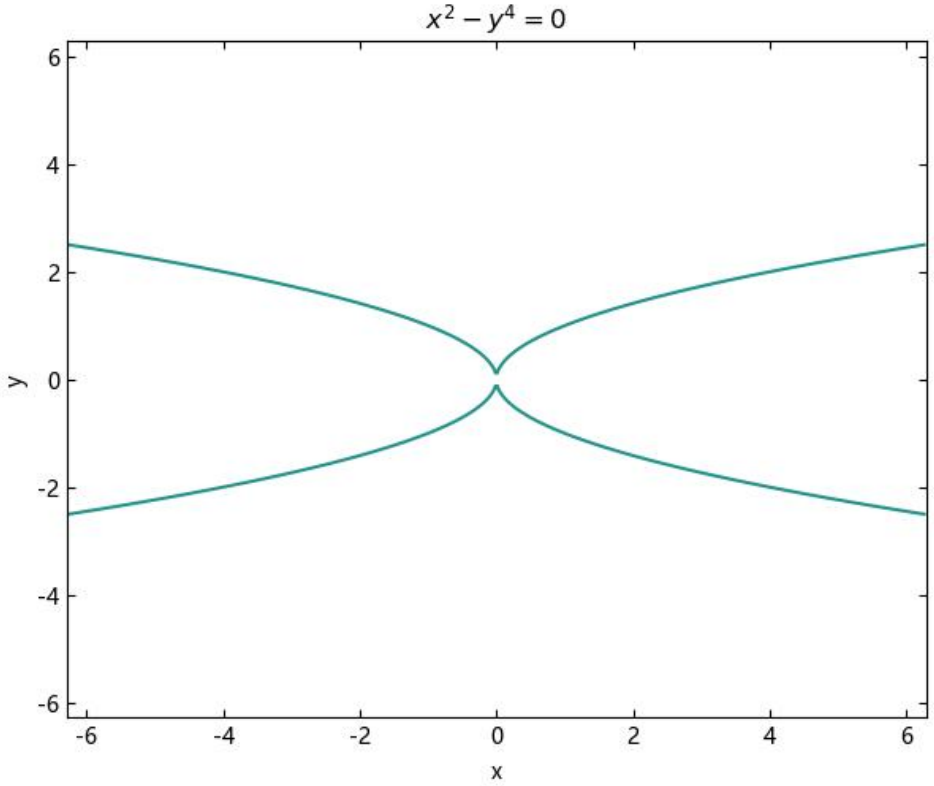


默认域为 [−2π,2π]。

(2). 绘制隐函数

在域 [−2π,2π] 中绘制隐式定义的函数 。

```
ezplot((x, y) -> x^2 - y^4)
title(raw"$x^2-y^4 = 0$")
xlabel("x")
ylabel("y")
```



默认域为 [−2π,2π]。

area

填充区二维绘图

A. 语法

```
area(X,Y)
area(Y)
area(___,Key=Value)
area(ax,___)
a = area(___)
```

B. 说明

area(X,Y) 绘制 Y 中的值对 x 坐标 X 的图。然后，该函数根据 Y 的形状填充曲线之间的区域：

如果 Y 是向量，则该图包含一条曲线。area 填充该曲线和水平轴之间的区域。

如果 Y 是矩阵，则该图对 Y 中的每列都包含一条曲线。area 填充这些曲线之间的区域并堆叠它们，从而显示在每个 x 坐标处每个行元素在总高度中的相对量。

area(Y) 绘制 Y 对一组隐式 x 坐标的图，并填充曲线之间的区域。

如果 Y 是向量，则 x 坐标范围从 1 到 length(Y)。

如果 Y 是矩阵，则 x 坐标的范围是从 1 到 Y 中的行数。

area(___,Key=Value) 使用一个或多个名称-值对组参数修改区域图的属性。这些属性适用于所有显示区域。例如，linestyle="-" 为绘图指定实线线型。需要在上述任一语法中的所有参数之后指定名称-值对组。

area(ax,___) 在目标坐标区上显示区域图。将坐标区指定为上述任一语法中的第一个参数。

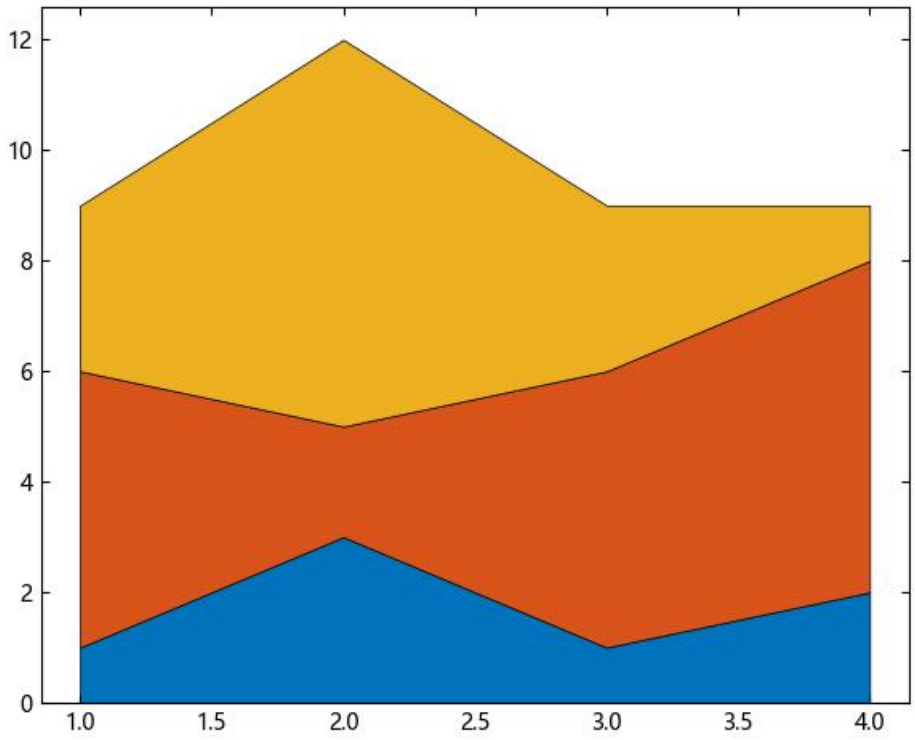
a = area(___) 返回一个或多个 Area 对象。对象的数量等于绘制区域的数量。创建区域后，使用 a 修改区域的属性。

C. 示例

(1). 创建区域图

将矩阵 Y 中的数据绘制成区域图。

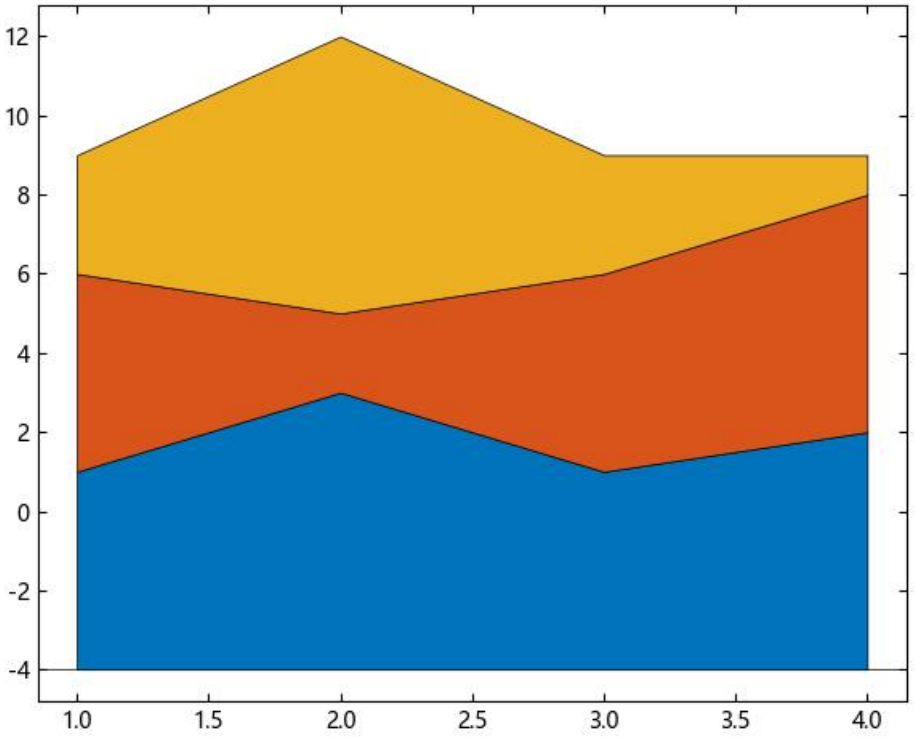
```
Y = [[1, 5, 3],
    [3, 2, 7],
    [1, 5, 3],
    [2, 6, 1]]
figure()
area(Y)
```



(2). 调整区域图的基准值

默认情况下，area 使用 y=0 作为基值。通过将 basevalue 输入参数设置为 -4 改变基值。

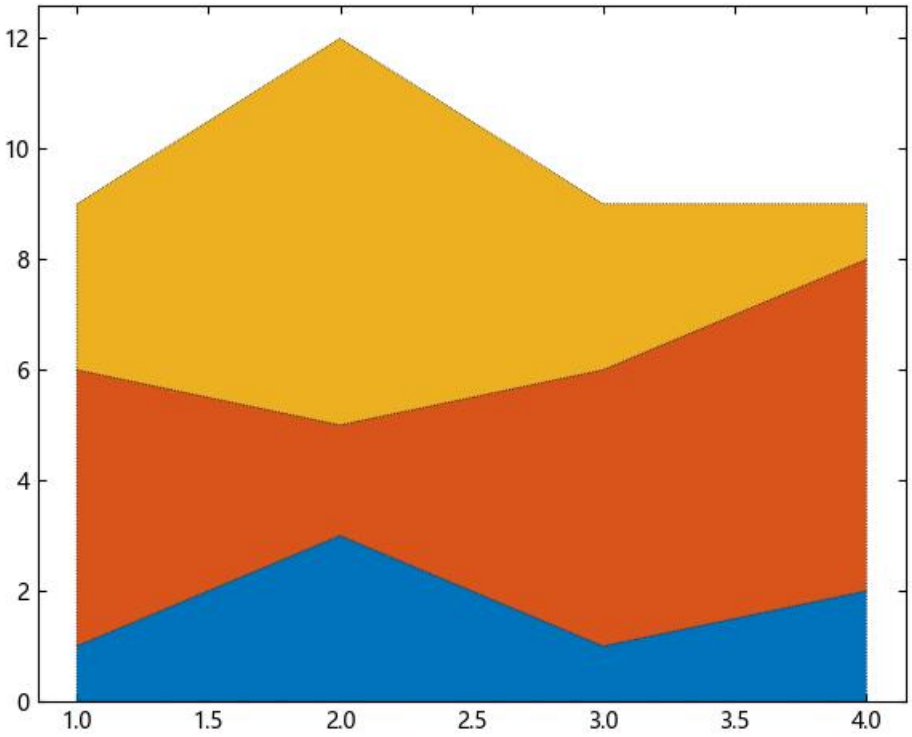
```
Y = [[1, 5, 3],[3, 2, 7],[1, 5, 3],[2, 6, 1]]
figure()
area(Y, basevalue = -4)
```



(3). 指定区域图的颜色和线型

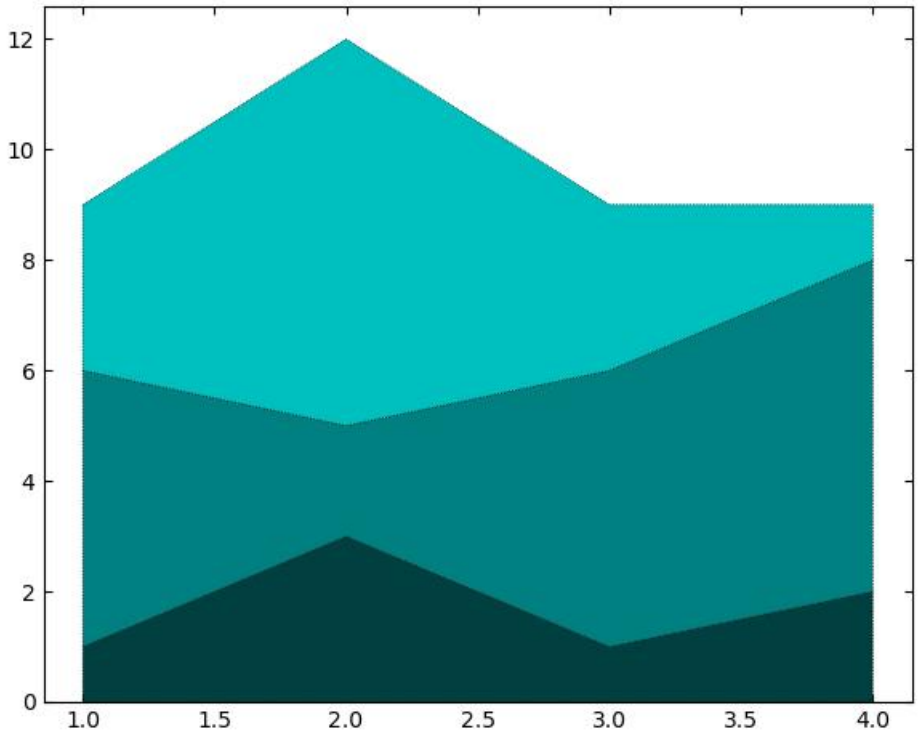
创建 Y 区域图，并使用点线样式。返回数组 h 中的三个区域对象。area 函数为 Y 中的每列创建一个区域对象。

```
Y = [[1, 5, 3],[3, 2, 7],[1, 5, 3],[2, 6, 1]]
figure()
h = area(Y, linestyle = ":")
```



使用 RGB 三元组颜色值更改区域颜色。使用圆点表示法设置属性。

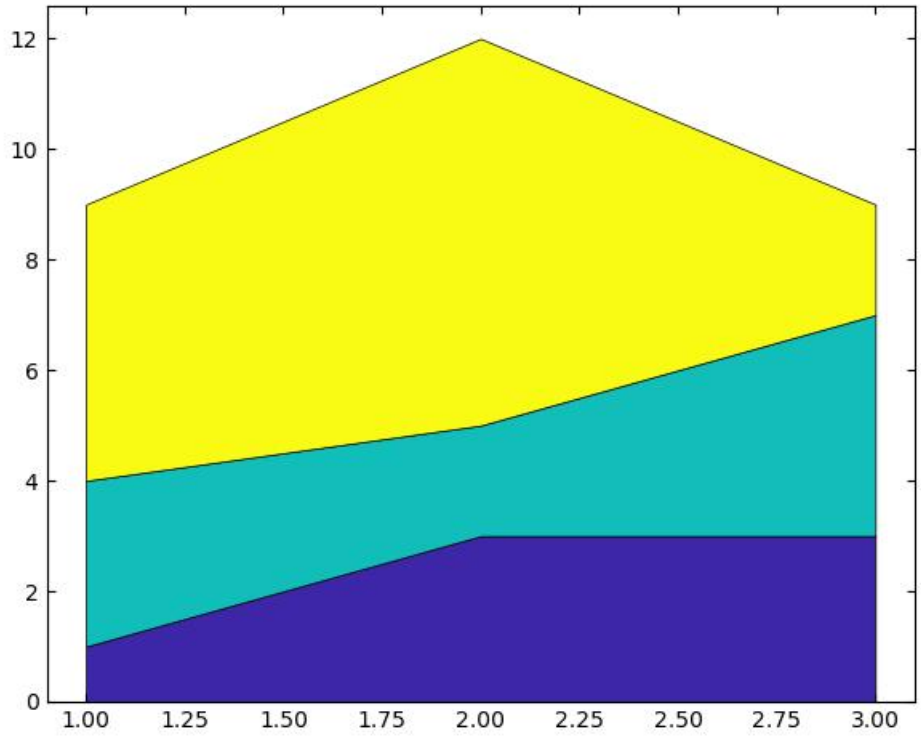
```
h[1].set_facecolor([0 0.25 0.25])
h[2].set_facecolor([0 0.5 0.5])
h[3].set_facecolor([0 0.75 0.75])
```



(4). 使用颜色图颜色的区域图

通过设置属性 facecolor，创建一个使用颜色图颜色的区域图。

```
y = [[1 3 5],[3 2 7], [3 4 2]]
figure()
h = area(y, facecolor = [[0.2422 0.1504 0.6603],
    [0.0704 0.7457 0.7258],
    [0.9769 0.9839 0.0805]])
```



3.3.1.1.2 对数图

函数名	简介
loglog	双对数刻度图
semilogx	半对数图(x 轴有对数刻度)
semilogy	半对数图(y 轴有对数刻度)

loglog

双对数刻度图

A. 语法

```
loglog(X,Y)
loglog(X,Y,fmt)
loglog(X1,Y1,...,Xn,Yn)
loglog(X1,Y1,fmt1,...,Xn,Yn,fmtn)
```

```
loglog(Y)
loglog(Y,fmt)

loglog(___,Key=Value)
loglog(ax,___)
lineobj = loglog(___)
```

B. 说明

loglog(X,Y) 在 x 轴和 y 轴上应用对数刻度来绘制 x 和 y 坐标。

要绘制由线段连接的一组坐标，请将 X 和 Y 指定为相同长度的向量。

要在同一组坐标轴上绘制多组坐标，请将 X 或 Y 中的至少一个指定为矩阵。

loglog(X,Y,fmt)使用指定的线型、标记和颜色创建绘图。

loglog(X1,Y1,...,Xn,Yn)在同一组坐标轴上绘制多对 x 和 y 坐标。此语法可替代将坐标指定为矩阵的形式。

loglog(X1,Y1,fmt1,...,Xn,Yn,fmtn)可为每个 x-y 对组指定特定的线型、标记和颜色。您可以对某些 x-y 对组指定 fmt，而对其他对组省略它。例如，loglog(X1,Y1,"o",X2,Y2)对第一个 x-y 对组指定标记，但没有对第二个对组指定标记。

loglog(Y) 绘制 Y 对一组隐式 x 坐标的图。

如果 Y 是向量，则 x 坐标范围从 1 到 length(Y)。

如果 Y 是矩阵，则对于 Y 中的每个列，图中包含一个对应的行。x 坐标的范围是从 1 到 Y 的行数。

loglog(Y,fmt) 指定线型、标记和颜色。

loglog(___,Key=Value)使用一个或多个 Key=Value 对组参数指定 Line 属性。这些属性应用于绘制的所有线条。需要在上述任一语法中的所有参数之后指定 Key=Value 对组。

loglog(ax,___)在目标坐标区上显示绘图。将坐标区指定为上述任一语法中的第一个参数。

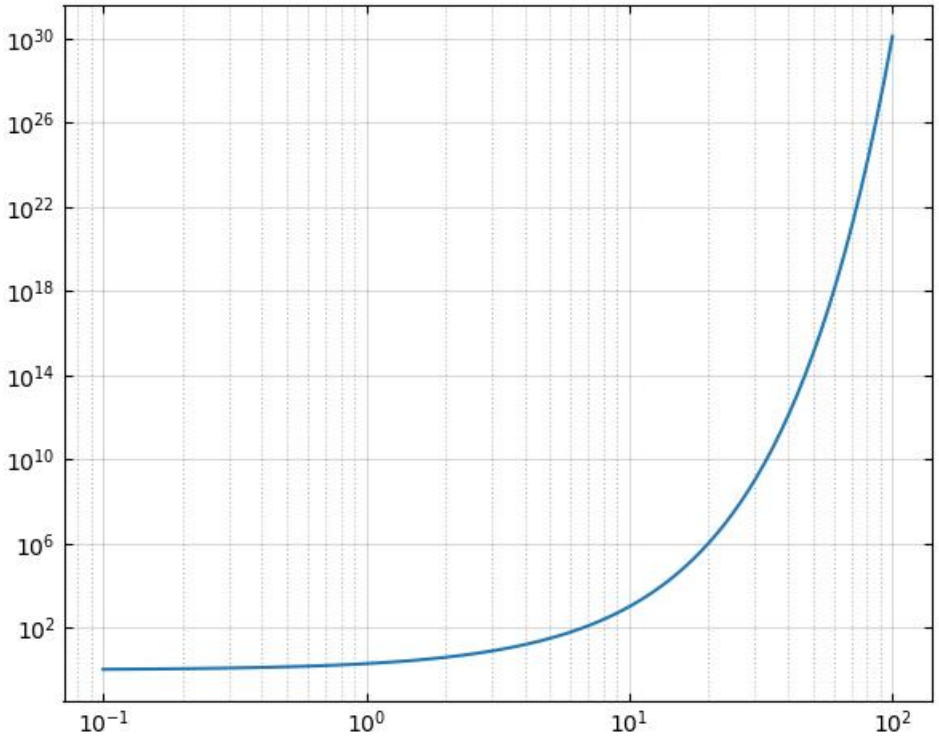
lineobj = loglog(___) 返回一个 Line 对象或 Line 对象数组。创建绘图后，使用 lineobj 修改该绘图的属性。

C. 示例

(1). 绘制一个线条

将 x 定义为一个由区间 内的 50 个对数间距数组成的向量。将 y 定义为 。然后绘制 x 和 y，并调用 grid 函数显示网格线。

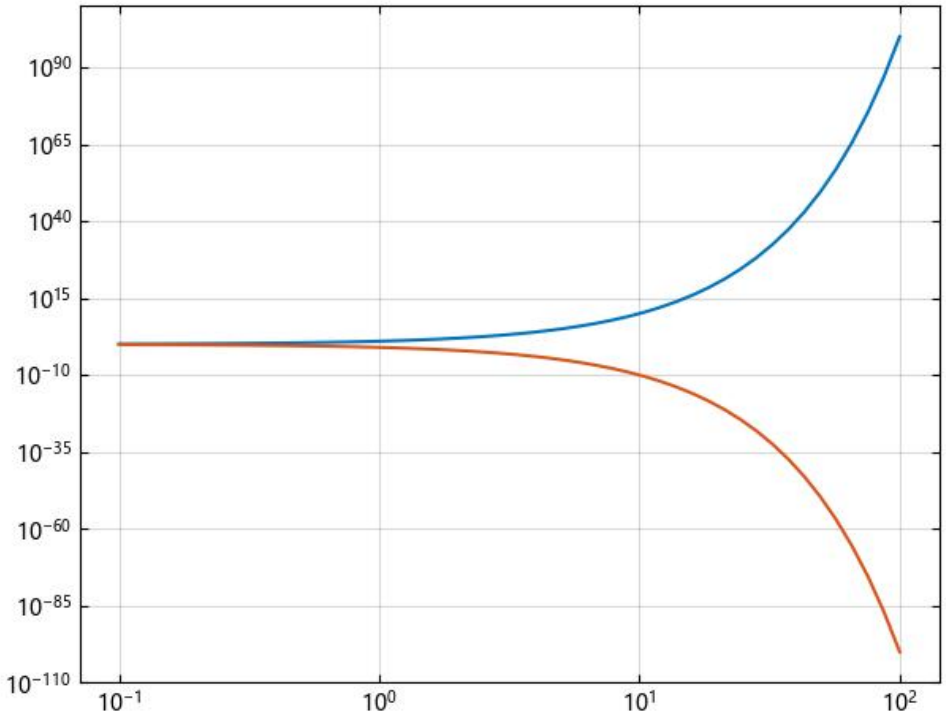
```
x = logspace(-1, 2, 100);
y = 2 .^ x;
loglog(x, y)
grid("on", "both")
```



(2). 绘制多个线条

创建一个由 x 坐标组成的向量和两个由 y 坐标组成的向量。通过将以逗号分隔的 x-y 对组传递给 loglog 绘制两个线条。

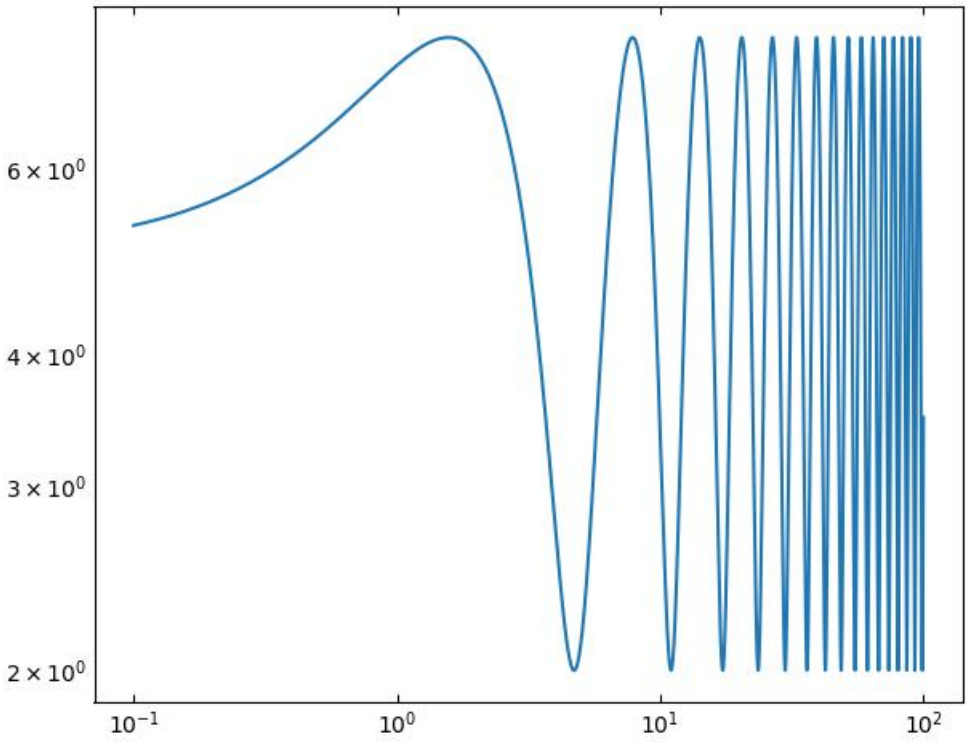
```
x = logspace(-1, 2, 50);
y1 = 10 .^ x;
y2 = 1 ./ 10 .^ x;
loglog(x, y1, x, y2)
grid("on")
```



(3). 指定轴标签和刻度值

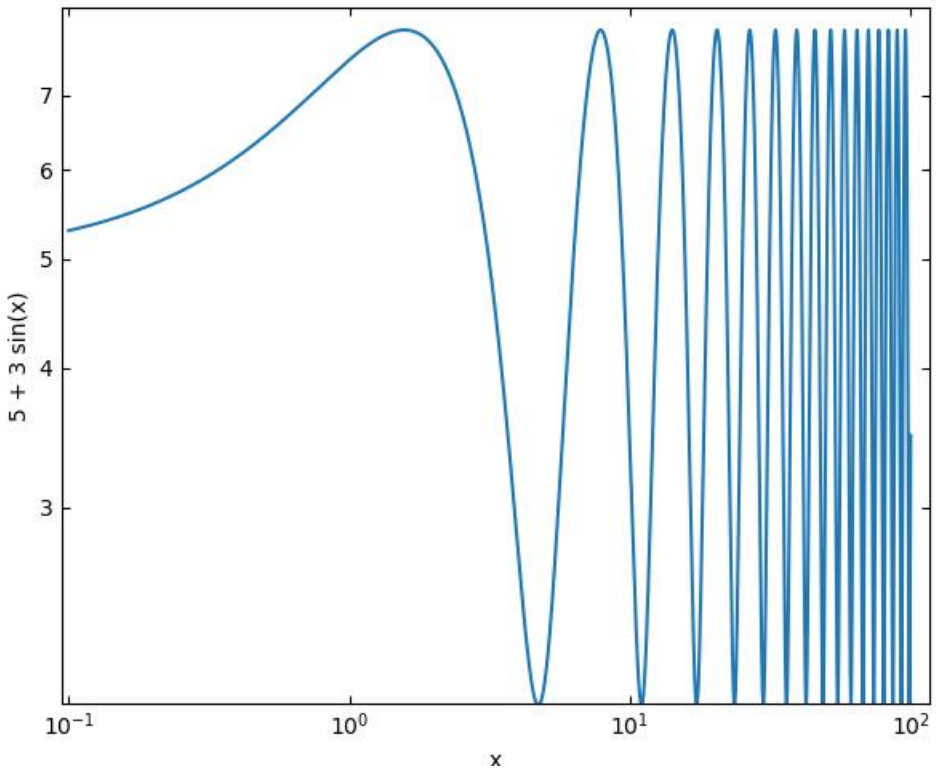
创建一组 x 坐标和 y 坐标，并将其显示在双对数图中。

```
x = logspace(-1, 2, 10000);
y = 5 .+ 3 * sin.(x);
loglog(x, y)
```



调用 yticks 函数，沿 y 轴以将 y 轴刻度值置于整数增量值位置。然后通过调用 xlabel 和 ylabel 函数创建 x 和 y 轴标签。

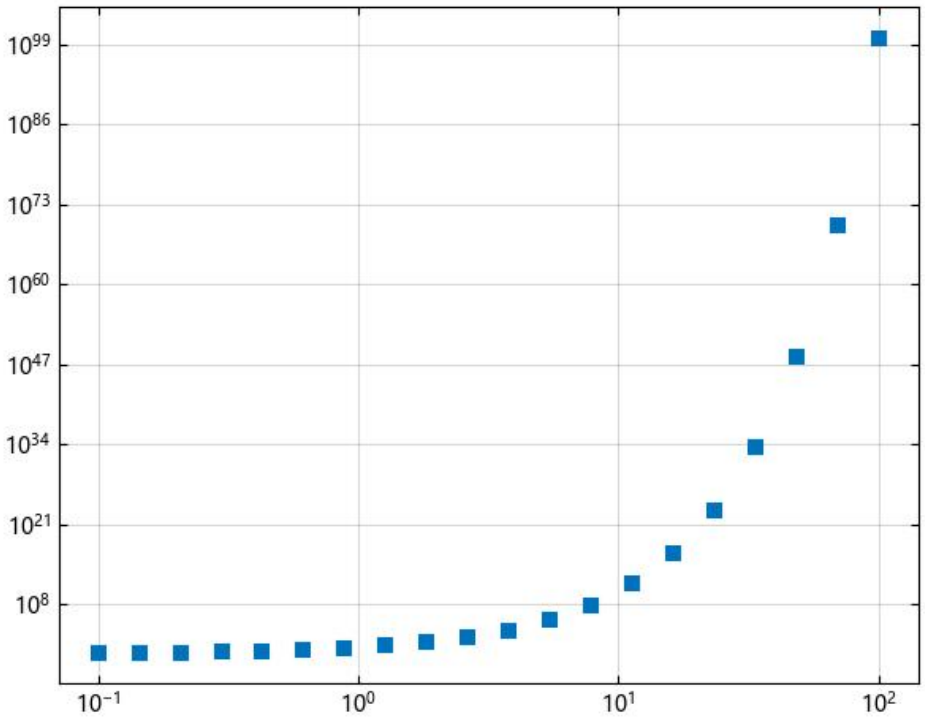
```
yticks([3, 4, 5, 6, 7])
xlabel("x")
ylabel("5 + 3 sin(x)")
```



(4). 将点绘制为不带线的标记

创建一组 x 坐标和 y 坐标，并将其显示在双对数图中。将线型指定为 "s" 以显示不带连接线的方形标记。将标记填充颜色指定为 RGB 三元组 [0 0.447 0.741]，它对应于深蓝色。

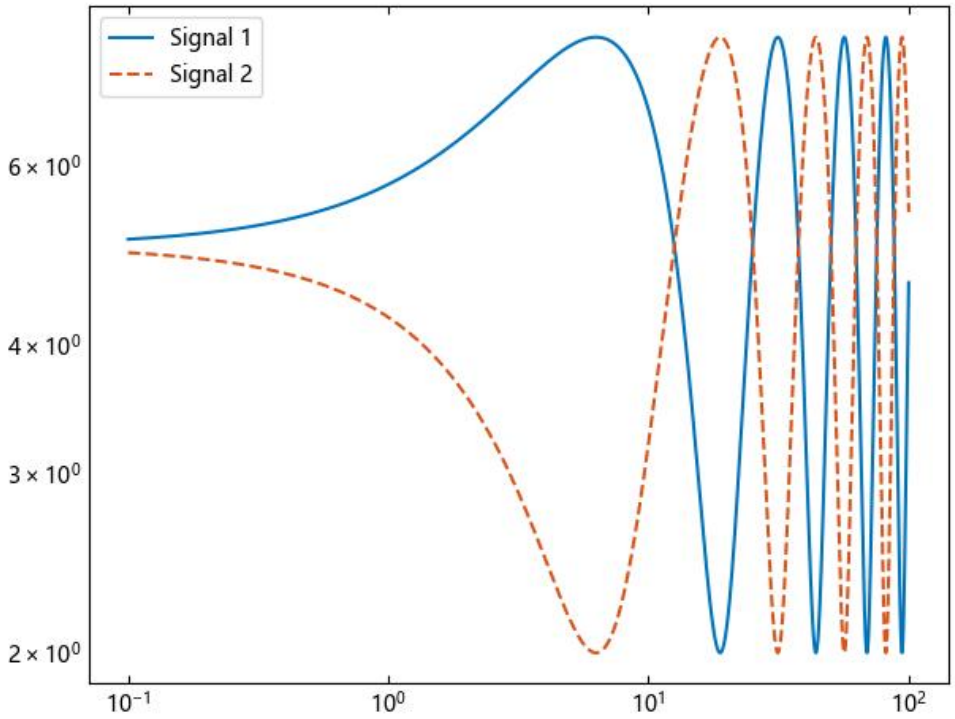
```
x = logspace(-1, 2, 20);
y = 10 .^ x;
loglog(x, y, "s"; markerfacecolor=[0, 0.447, 0.741])
grid("on")
```



(5). 添加图例

创建两组 x 坐标和 y 坐标，并将其显示在一个双对数图中。通过调用 legend 函数并将位置指定为 "northwest"，在绘图的左上角显示图例。

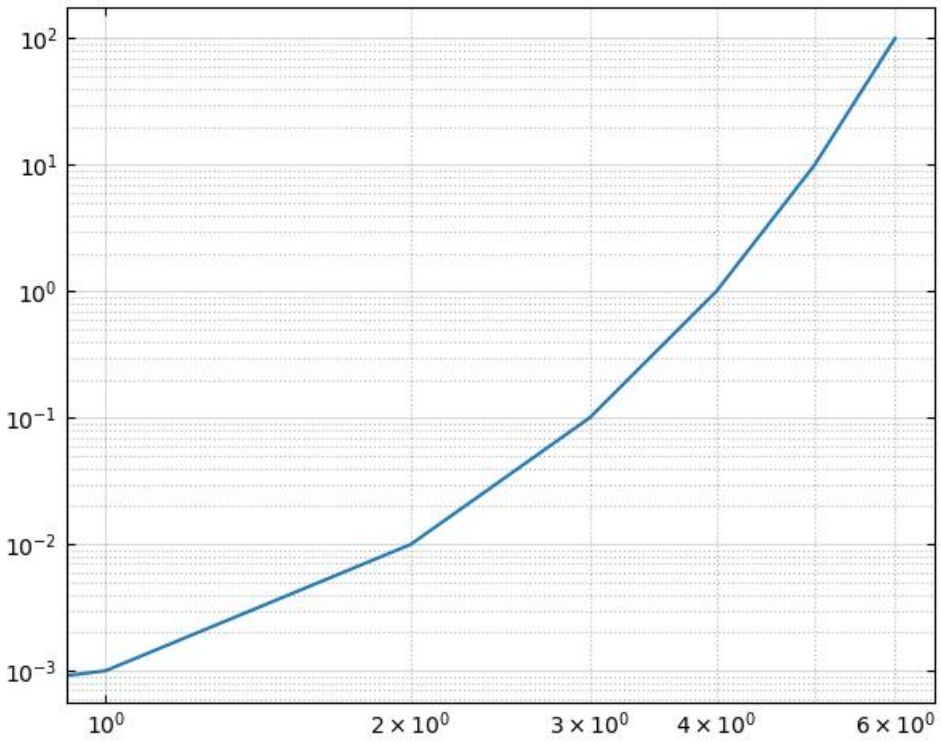
```
x = logspace(-1, 2, 10000)
y1 = 5 .+ 3 * sin.(x / 4)
y2 = 5 .- 3 * sin.(x / 4)
loglog(x, y1, x, y2, "--")
legend(["Signal 1","Signal 2"], loc = "northwest")
```



(6). 仅指定 y 坐标

当您仅指定一个坐标向量时，loglog 会绘制这些坐标对值 1:length(y) 的图。例如，将 y 定义为一个由 0.001 和 100 之间的 6 个值组成的向量。创建 y 的双对数图。

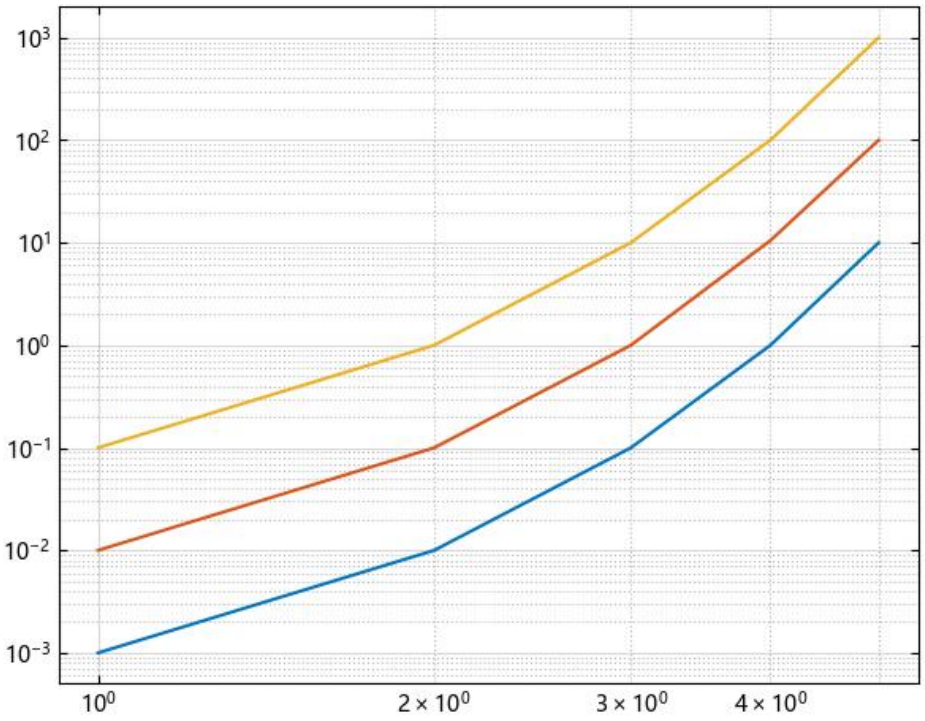
```
y = [0, 0.001, 0.01, 0.1, 1, 10, 100]
loglog(y)
grid("on","both")
```



如果您将 y 指定为矩阵，则绘制 y 的列对值 1:size(y,1) 的图。例如，将 y 定义为 5×3 矩阵，并将其传递给 loglog 函数。生成的图包含 3 个线条，其中每个线条的 x 坐标都是从 0 到 4。

```
y
 = [0.0010    0.0100    0.1000
    0.0100    0.1000    1.0000
    0.1000    1.0000   10.0000
    1.0000   10.5000  100.0000
    10.0000  100.0000 1000.0000];

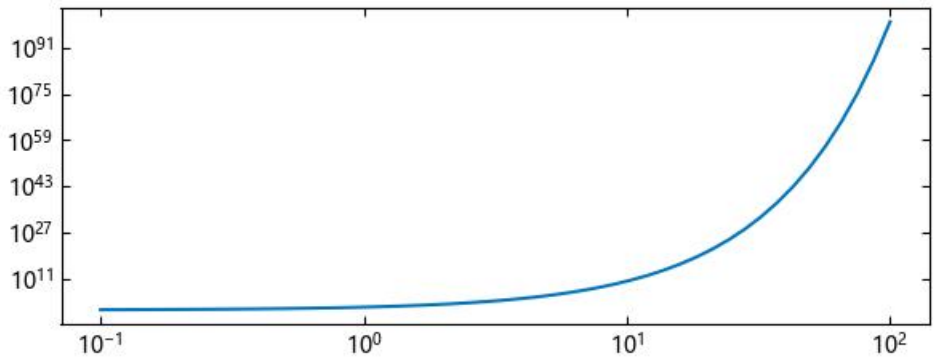
loglog(y)
grid("on", "both")
```



(7). 指定目标坐标区

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，以便坐标区填充布局中的可用空间。接下来，将该对象返回为 ax1。通过将 ax1 传递给 loglog 函数来显示一个双对数图。

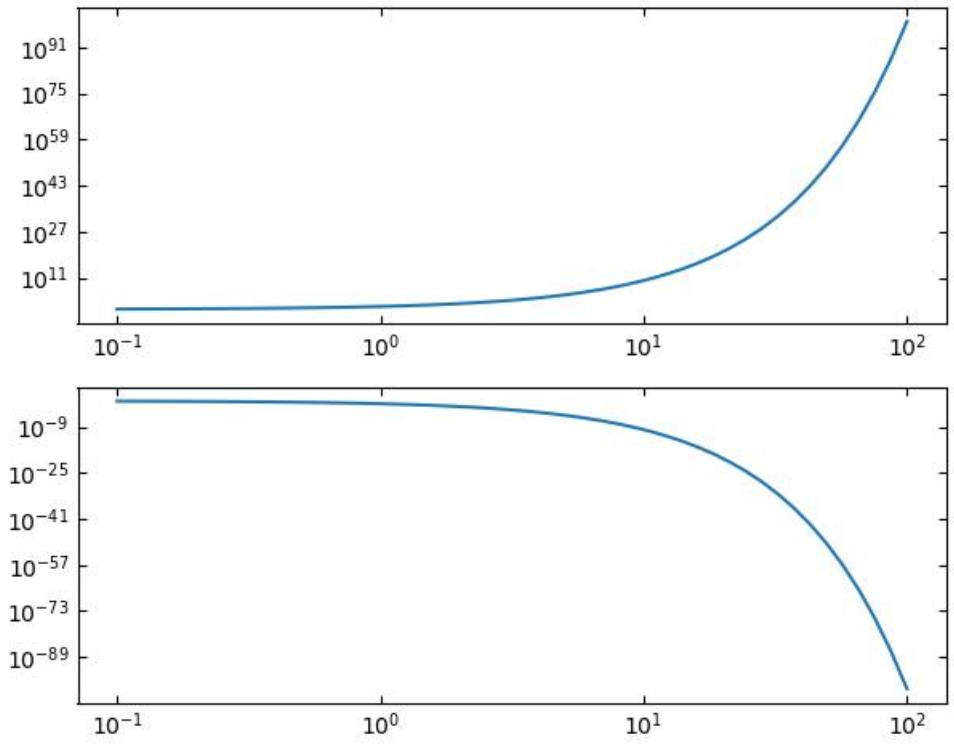
```
ax1 = subplot(2, 1, 1)
x = logspace(-1, 2)
y1 = 10 .^ x;
loglog(ax1, x, y1)
```



重复该过程以创建第二个坐标区对象和第二个双对数图。

```
ax2 = subplot(2, 1, 2)
y2 = 1 ./ 10 .^ x;
```

```
loglog(ax2, x, y2)
```



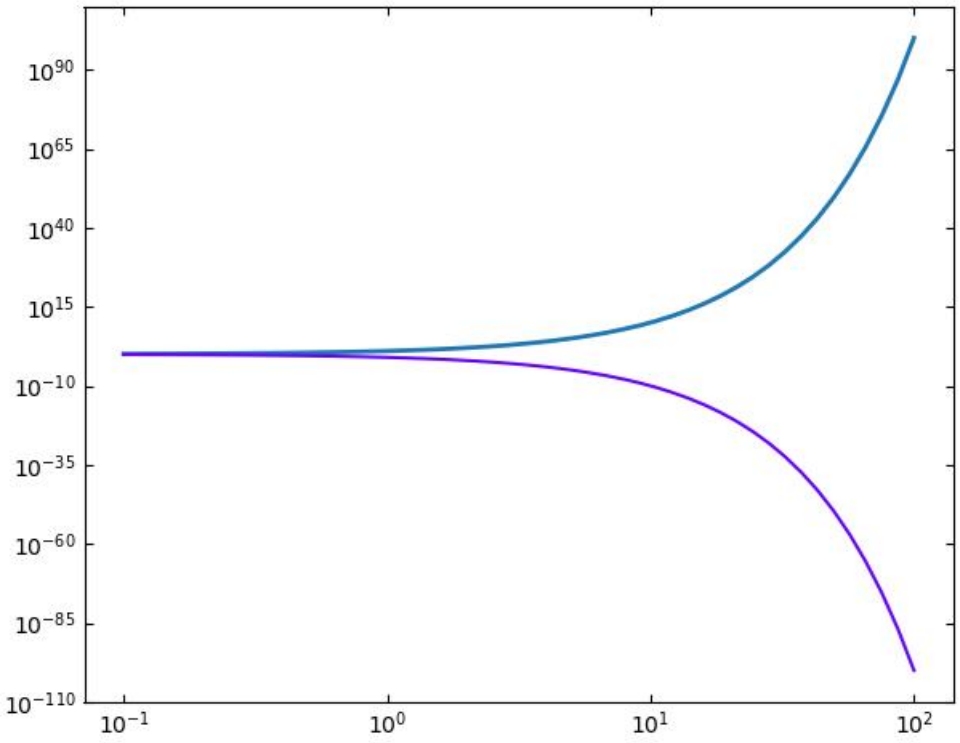
(8). 绘图后更改线特性

创建一个包含两个线条的双对数图，并以变量 lg 形式返回行对象。

```
x = logspace(-1, 2);
y1 = 10 .^ x;
y2 = 1 ./ 10 .^ x;
lg = loglog(x, y1, x, y2);
```

将第一个线条的宽度更改为 2，并将第二个线条的颜色更改为紫色。

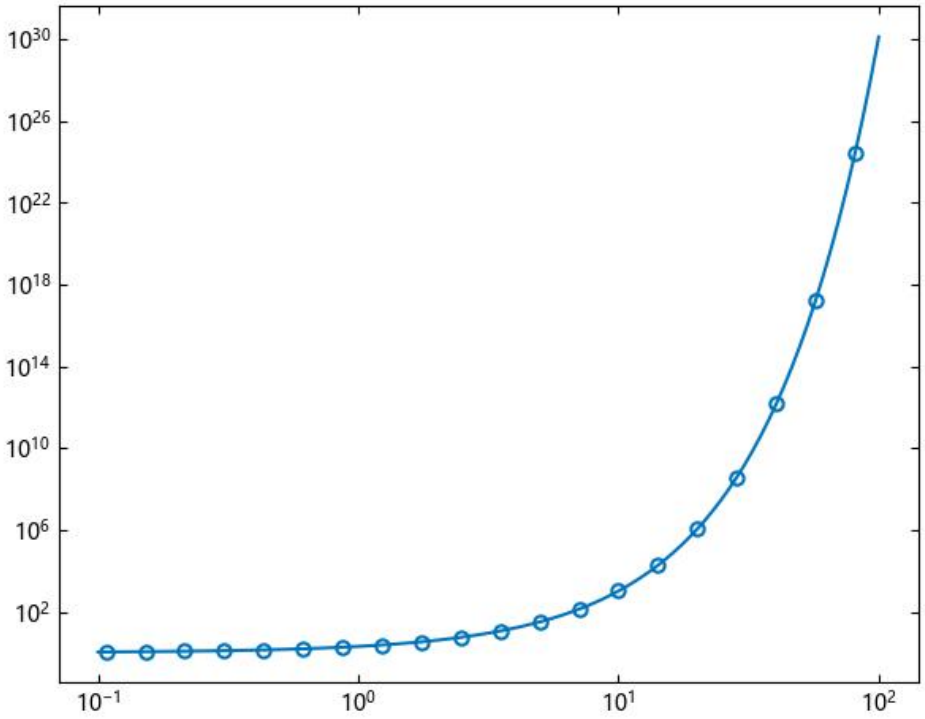
```
lg[1].set_linewidth(2);
lg[2].set_color([0.4,0,1]);
```



(9). 在特定的数据点显示标记

将 x 定义为一个由区间 内的 50 个对数间距数组成的向量。将 y 定义为 。然后绘制 x 和 y，并将 markevery 属性设置为名称-值对组的形式，创建一个对数图并每隔四个数据点显示一个标记。

```
x = logspace(-1, 2, 100);
y = 2 .^ x;
loglog(x, y, "-o"; markevery=1:5:length(y))
```



semilogx

半对数图（x 轴有对数刻度）

A. 语法

```
semilogx(X,Y)
semilogx(X,Y,Fmt)
semilogx(X1,Y1,...,Xn,Yn)
semilogx(X1,Y1,Fmt1,...,Xn,Yn,Fmtn)

semilogx(Y)
semilogx(Y,Fmt)

semilogx(___,Key=Value)
semilogx(ax,___)
lineobj = semilogx(___)
```

B. 说明

semilogx(X,Y) 在 x 轴上使用以 10 为底的对数刻度、在 y 轴上使用线性刻度来绘制 x 和 y 坐标。

要绘制由线段连接的一组坐标，请将 X 和 Y 指定为相同长度的向量。

要在同一组坐标轴上绘制多组坐标，请将 X 或 Y 中的至少一个指定为矩阵。

semilogx(X,Y,Fmt)使用指定的线型、标记和颜色创建绘图。

semilogx(X1,Y1,...,Xn,Yn)在同一组坐标轴上绘制多对 x 和 y 坐标。此语法可替代将坐标指定为矩阵的形式。

semilogx(X1,Y1,Fmt1,...,Xn,Yn,Fmtn)可为每个 x-y 对组指定特定的线型、标记和颜色。您可以对某些 x-y 对组指定 Fmt，而对其他对组省略它。例如，semilogx(X1,Y1,"o",X2,Y2)对第一个 x-y 对组指定标记，但没有对第二个对组指定标记。

semilogx(Y) 绘制 Y 对一组隐式 x 坐标的图。

如果 Y 是向量，则 x 坐标范围从 1 到 length(Y)。

如果 Y 是矩阵，则对于 Y 中的每个列，图中包含一个对应的行。x 坐标的范围是从 1 到 Y 的行数。

semilogx(Y,Fmt) 指定线型、标记和颜色。

semilogx(___,Key=Value)使用一个或多个 Key=Value 对组参数指定 Line 属性。这些属性应用于绘制的所有线条。需要在上述任一语法中的所有参数之后指定 Key=Value 对组。有关属性列表，请参阅 Line 属性 (opens new window)。

semilogx(ax,___)在目标坐标区上显示绘图。将坐标区指定为上述任一语法中的第一个参数。

lineobj = semilogx(___) 返回一个 Line 对象或 Line 对象数组。创建绘图后，使用 lineobj 修改该绘图的属性。有关属性列表，请参阅 Line 属性 (opens new window)。

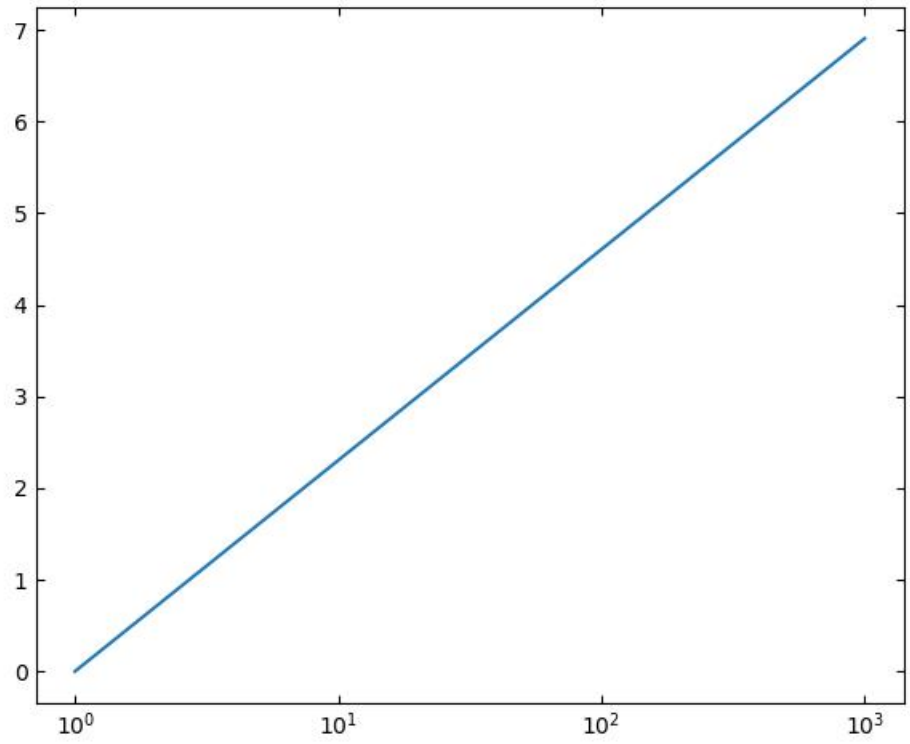
C. 示例

(1). x 轴的对数刻度

使用 x 轴的对数刻度和 y 轴的线性刻度创建一个绘图。

```
using TyPlot

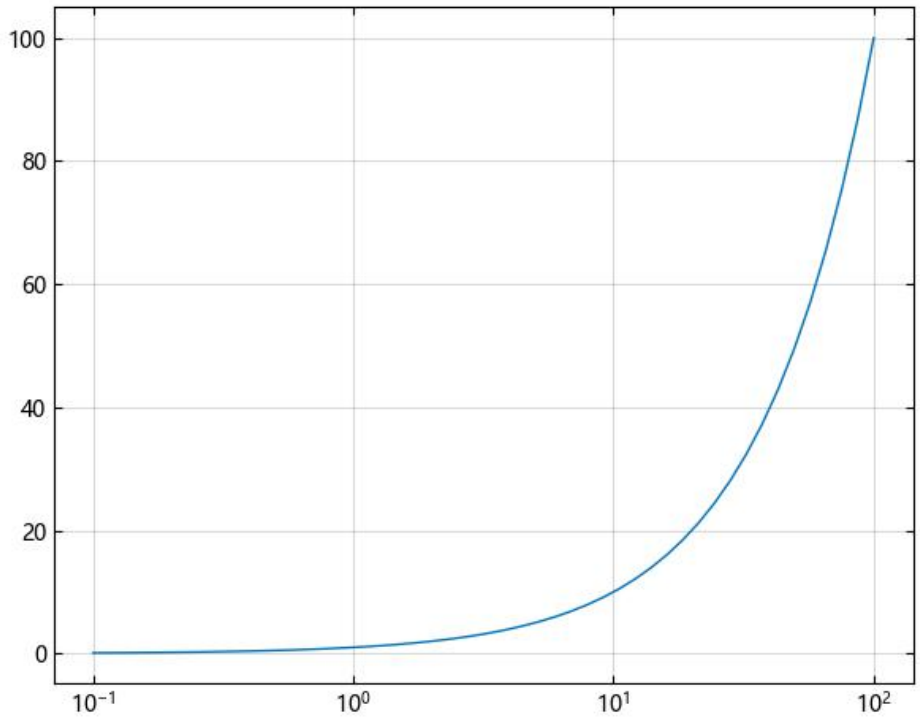
x = 0:1000
y = log.(x)
semilogx(x, y)
```



(2). 绘制一个线条

将 x 定义为一个由从 0.1 到 100 的对数间距值组成的向量，将 y 定义为 x 的副本。创建 x 和 y 的线性对数图，并调用 grid 函数显示网格线。

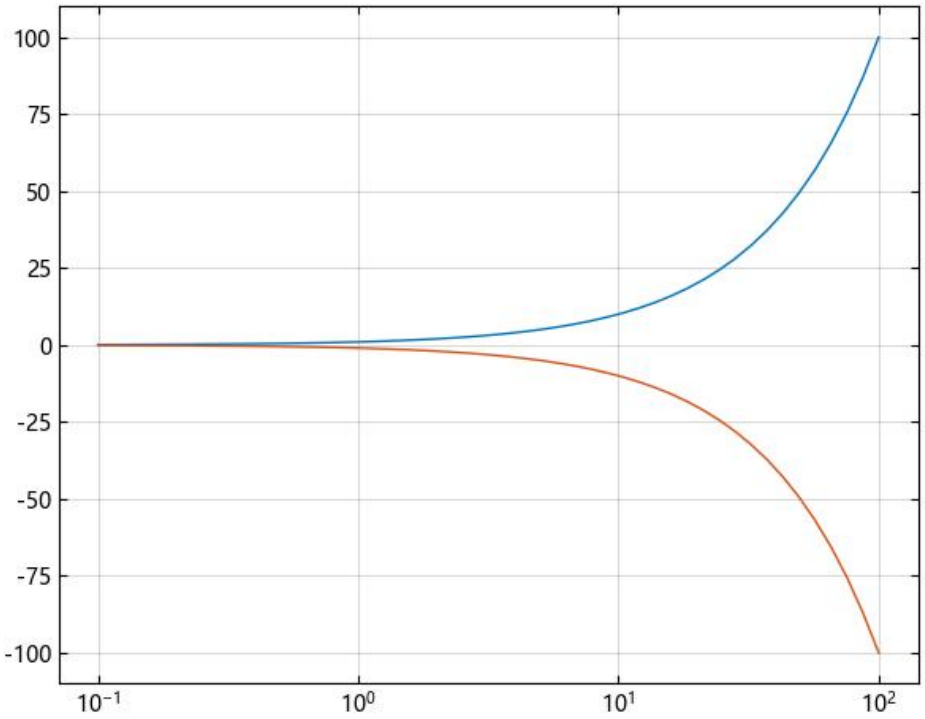
```
x = logspace(-1, 2)
y = x
semilogx(x, y)
grid("on")
```



(3). 绘制多个线条

创建一个由对数间距 x 坐标组成的向量和两个由 y 坐标组成的向量。通过将以逗号分隔的 x-y 对组传递给 semilogx 绘制两个线条。

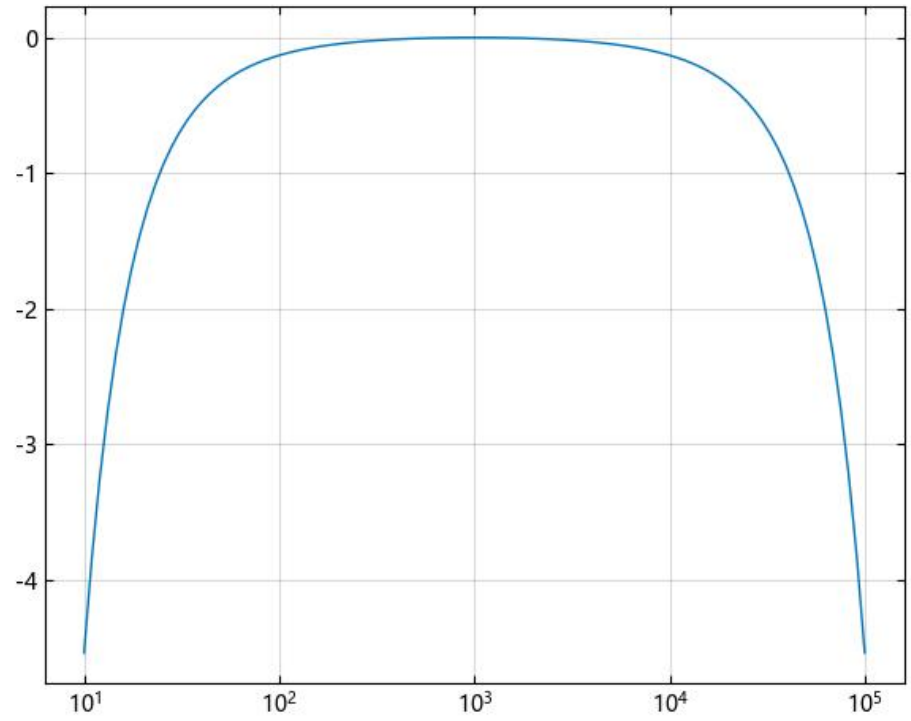
```
x = logspace(-1, 2)
y1 = x
y2 = -x
semilogx(x, y1, x, y2)
grid("on")
```



(4). 指定轴标签和刻度值

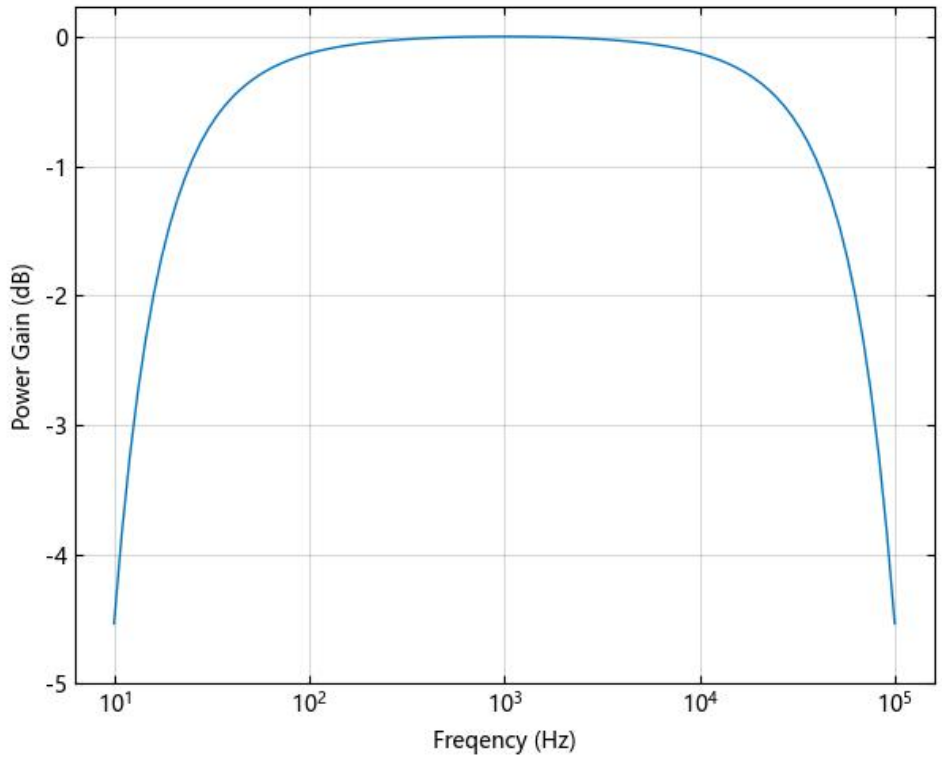
将 f 定义为一个包含从 10 Hz 到 100000 Hz 的频率的向量。将 gain 定义为以分贝为单位的功率增益值的向量。然后绘制增益值对频率的图。

```
f = logspace(1, 5, 100);
v = LinRange(-50, 50, 100);
gain = (1 .- exp.(5 * (2.5 * v .^ 2) ./ 7500)) / 14;
semilogx(f, gain)
grid("on")
```



调用 yticks 函数，以将 y 轴刻度值重新定位于沿 y 轴的整数增量值位置。然后通过调用 xlabel 和 ylabel 函数创建 x 和 y 轴标签。

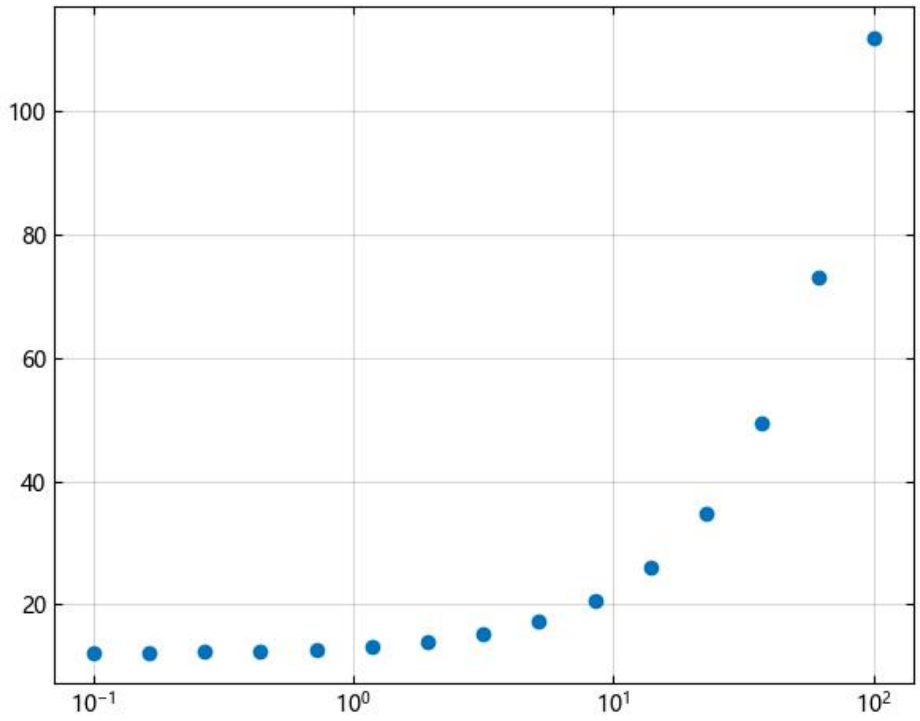
```
gca().set_yticks([-5, -4, -3, -2, -1, 0])
xlabel("Freqency (Hz)")
ylabel("Power Gain (dB)")
```



(5). 将点绘制为不带线的标记

创建一组 x 坐标和 y 坐标，并将其显示在线性对数图中。将线型指定为 "o" 以显示不带连接线的圆形标记。将标记填充颜色指定为 RGB 三元组 [0 0.447 0.741]，它对应于深蓝色。

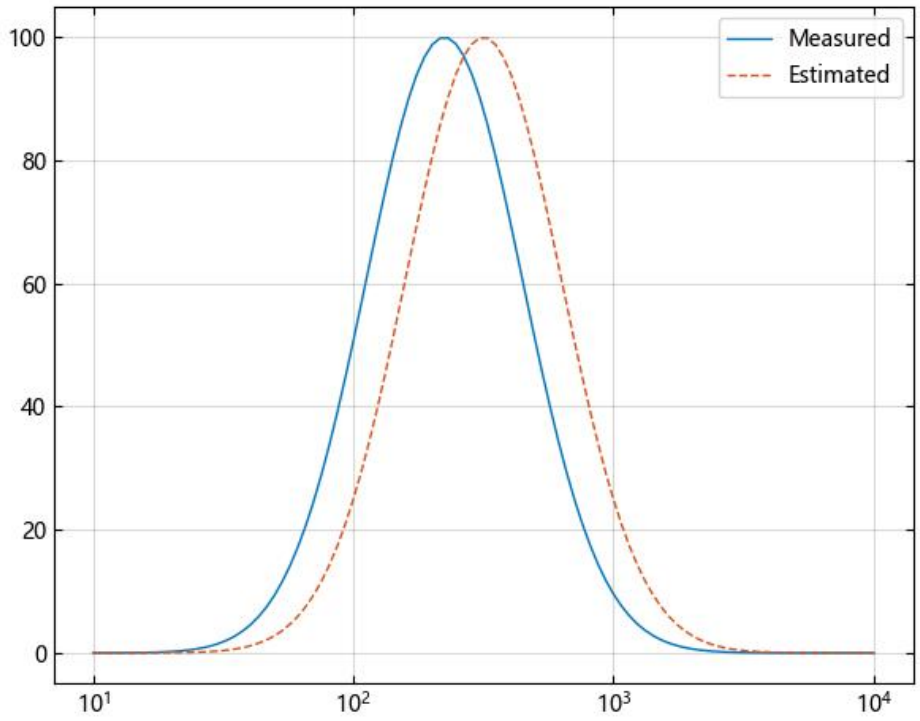
```
x = logspace(-1, 2, 15);
y = 12 .+ x;
semilogx(x, y, "o"; markerfacecolor=[0, 0.447, 0.741])
grid("on")
```



(6). 添加图例

创建一个由对数间距 x 坐标组成的向量和两个由 y 坐标组成的向量。然后通过将以逗号分隔的 x-y 对组传递给 semilogx 绘制两个线条。通过调用 legend 函数显示图例。

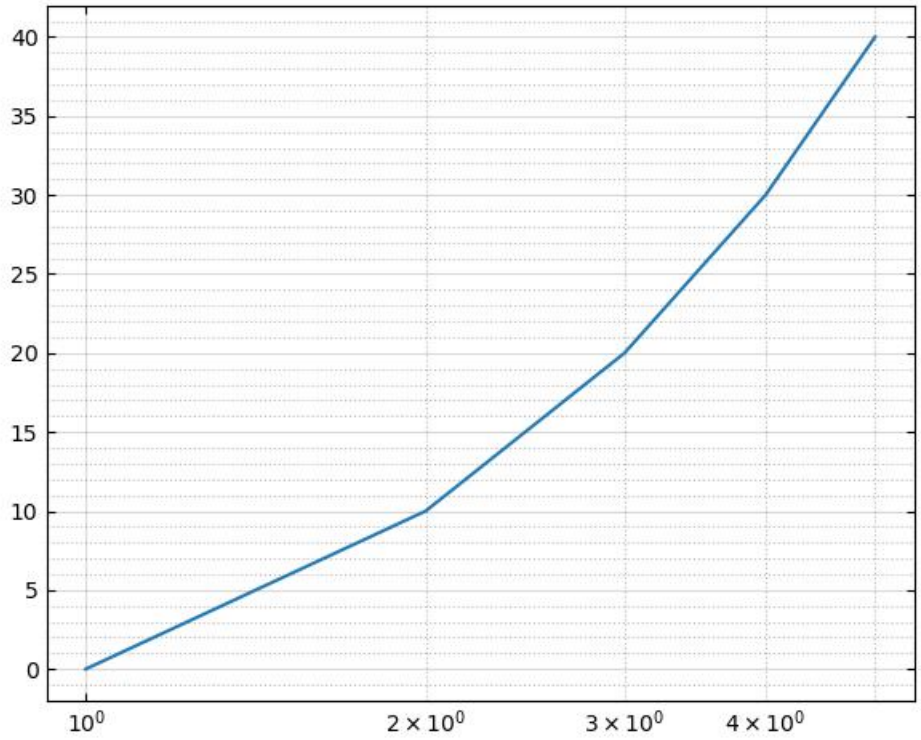
```
x = logspace(1, 4, 100)
v = LinRange(-50, 50, 100)
y1 = 100 .* exp.(-1 * ((v .+ 5) .^ 2) ./ 200)
y2 = 100 .* exp.(-1 * (v .^ 2) ./ 200)
semilogx(x, y1, x, y2, "--")
legend(["Measured", "Estimated"])
grid("on")
```



(7). 仅指定 y 坐标

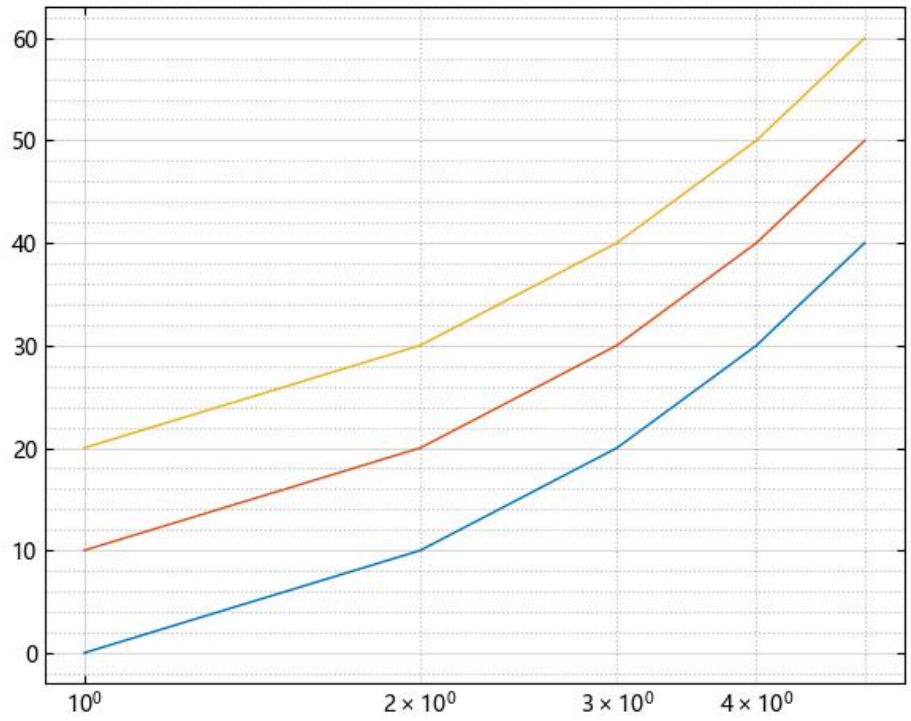
当您仅指定一个坐标向量时，semilogx 会绘制这些坐标对值 1:length(y) 的图。例如，将 y 定义为一个由 0 和 40 之间的 5 个值组成的向量。创建 y 的线性对数图。

```
y = [0, 10, 20, 30, 40];
semilogx(y)
grid("on","both")
```



如果您将 y 指定为矩阵，则绘制 y 的列对值 1:size(y,1) 的图。例如，将 y 定义为 5×3 矩阵，并将其传递给 semilogx 函数。生成的图包含 3 个线条，其中每个线条的 x 坐标都是从 0 到 4。

```
y = [ 0   10   20
   10   20   30
   20   30   40
   30   40   50
   40   50   60];
semilogx([1,2,3,4,5], y)
grid("on","both")
```



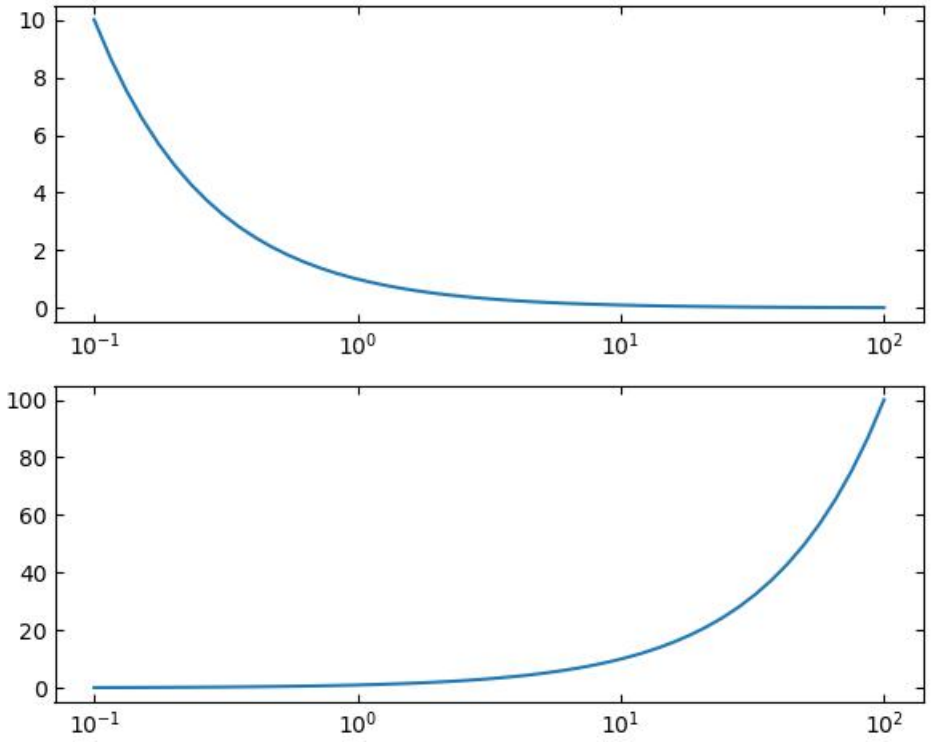
(8). 指定目标坐标区

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，以便坐标区填充布局中的可用空间。接下来，将该对象返回为 ax1。通过将 ax1 传递给 semilogx 函数来显示线性对数图。

```
ax1 = subplot(2, 1, 1)
x = logspace(-1, 2)
y1 = 1 ./ x
semilogx(ax1, x, y1)
```

重复该过程以创建第二个线性对数图。

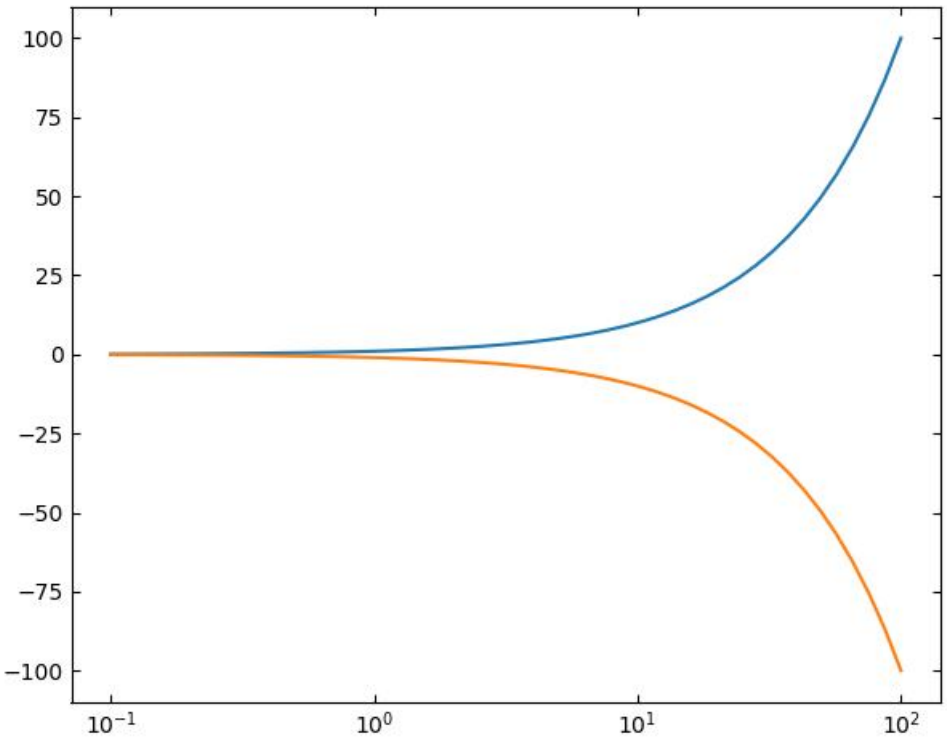
```
ax2 = subplot(2, 1, 2)
y2 = x
semilogx(ax2, x, y2)
```



(9). 绘制后更改线条外观

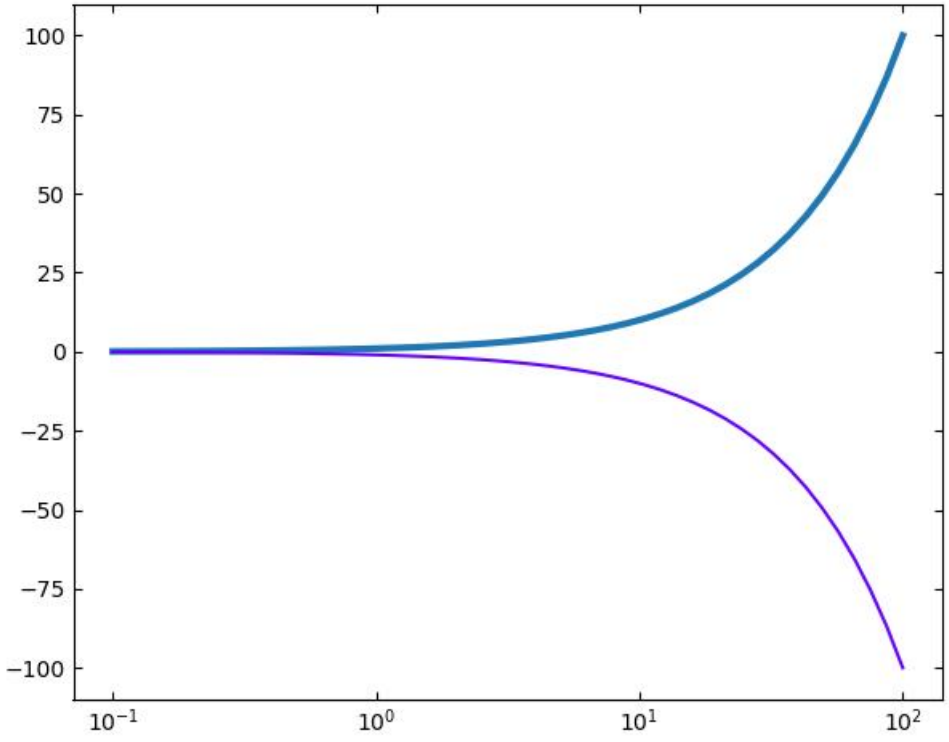
创建一个包含两个线条的线性对数图，并以变量 slg 形式返回线条对象。

```
x = logspace(-1, 2)
y1 = x
y2 = -x
slg = semilogx(x, y1, x, y2)
```



将第一个线条的宽度更改为 3，并将第二个线条的颜色更改为紫色。

```
slg[1].set_linewidth(3)
slg[2].set_color([0.4,0,1])
```

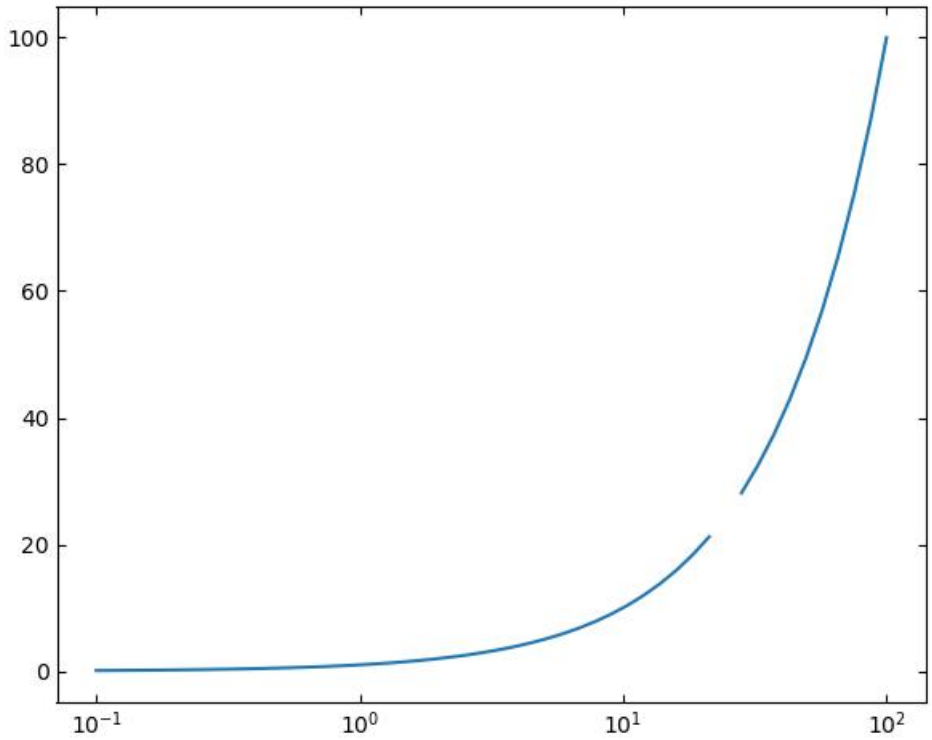


(10). 绘制不连续函数

在数据中任何不连续的位置插入 NaN 值。semilogx 函数显示这些位置的空缺。

创建一对 x 坐标向量和 y 坐标向量。用 NaN 值替换第四十个 y 坐标。然后创建一个 x 和 y 的线性对数图。

```
x = logspace(-1, 2);
y = x;
y[40] = NaN;
semilogx(x, y)
```



semilogy

半对数图（y 轴有对数刻度）

A. 语法

```
semilogy(X,Y)
semilogy(X,Y,Fmt)
semilogy(X1,Y1,...,Xn,Yn)
semilogy(X1,Y1,Fmt1,...,Xn,Yn,Fmtn)
```

```
semilogy(Y)
semilogy(Y,Fmt)

semilogy(___,Key=Value)
semilogy(ax,___)
lineobj = semilogy(___)
```

B. 说明

semilogy(X,Y) 在 x 轴上使用线性刻度、在 y 轴上使用以 10 为底的对数刻度来绘制 x 和 y 坐标。

要绘制由线段连接的一组坐标，请将 X 和 Y 指定为相同长度的向量。

要在同一组坐标轴上绘制多组坐标，请将 X 或 Y 中的至少一个指定为矩阵。

semilogy(X,Y,Fmt)使用指定的线型、标记和颜色创建绘图。

semilogy(X1,Y1,...,Xn,Yn)在同一组坐标轴上绘制多对 x 和 y 坐标。此语法可替代将坐标指定为矩阵的形式。

semilogy(X1,Y1,Fmt1,...,Xn,Yn,Fmtn)可为每个 x-y 对组指定特定的线型、标记和颜色。您可以对某些 x-y 对组指定 Fmt，而对其他对组省略它。例如，semilogy(X1,Y1,"o",X2,Y2)对第一个 x-y 对组指定标记，但没有对第二个对组指定标记。

semilogy(Y) 绘制 Y 对一组隐式 x 坐标的图。

如果 Y 是向量，则 x 坐标范围从 1 到 length(Y)。

如果 Y 是矩阵，则对于 Y 中的每个列，图中包含一个对应的行。x 坐标的范围是从 1 到 Y 的行数。

semilogy(Y,Fmt) 指定线型、标记和颜色。

semilogy(___,Key=Value)使用一个或多个 Key=Value 对组参数指定 Line 属性。这些属性应用于绘制的所有线条。需要在上述任一语法中的所有参数之后指定 Key=Value 对组。有关属性列表，请参阅 Line 属性 (opens new window)。

semilogy(ax,___)在目标坐标区上显示绘图。将坐标区指定为上述任一语法中的第一个参数。

lineobj = semilogy(___) 返回一个 Line 对象或 Line 对象数组。创建绘图后，使用 lineobj 修改该绘图的属性。有关属性列表，请参阅 Line 属性 (opens new window)。

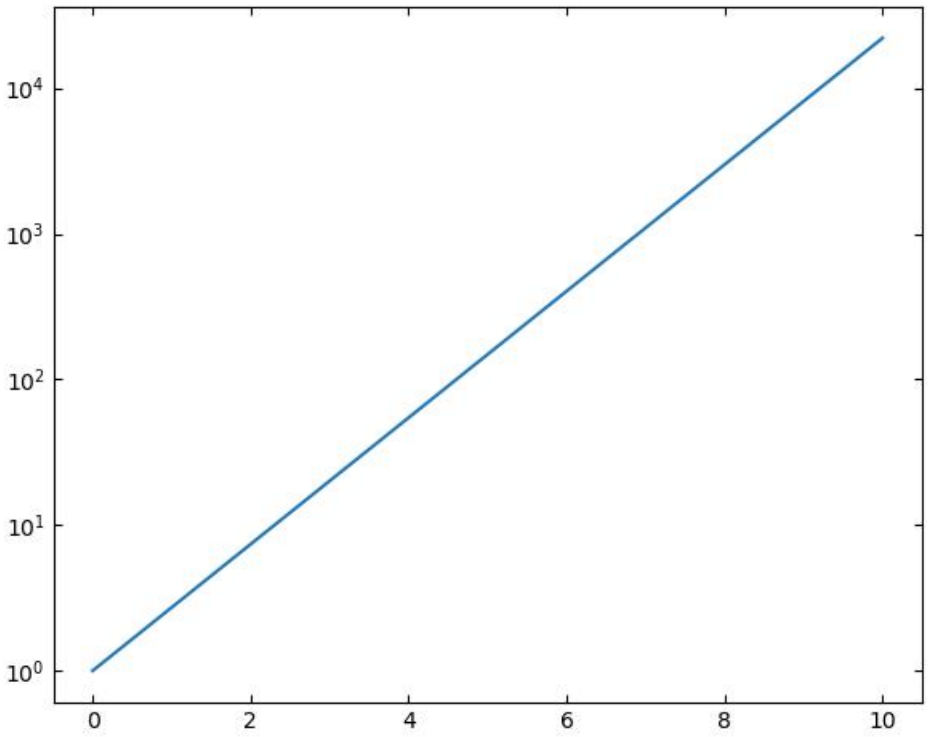
C. 示例

(1). y 轴的对数刻度

使用 y 轴的对数刻度和 x 轴的线性刻度创建一个绘图。

```
using TyPlot

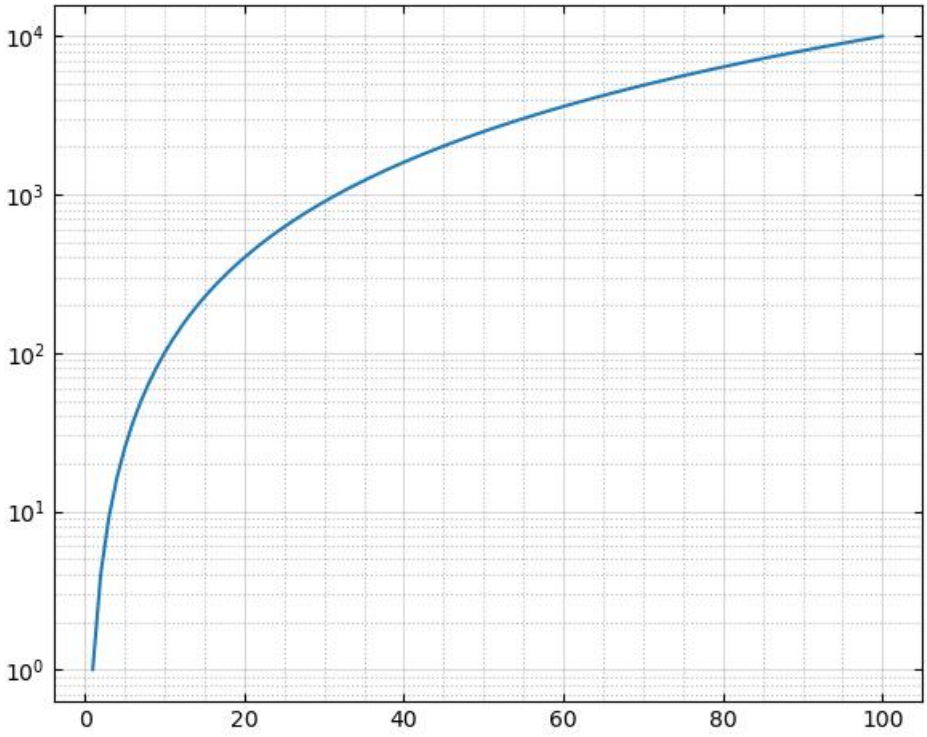
x = 1:100;
y = x .^ 2;
semilogy(x, y)
```



(2). 绘制一个线条

创建一个由 x 坐标组成的向量和一个由 y 坐标组成的向量。创建 x 和 y 的对数线性图，并调用 grid 函数显示网格线。

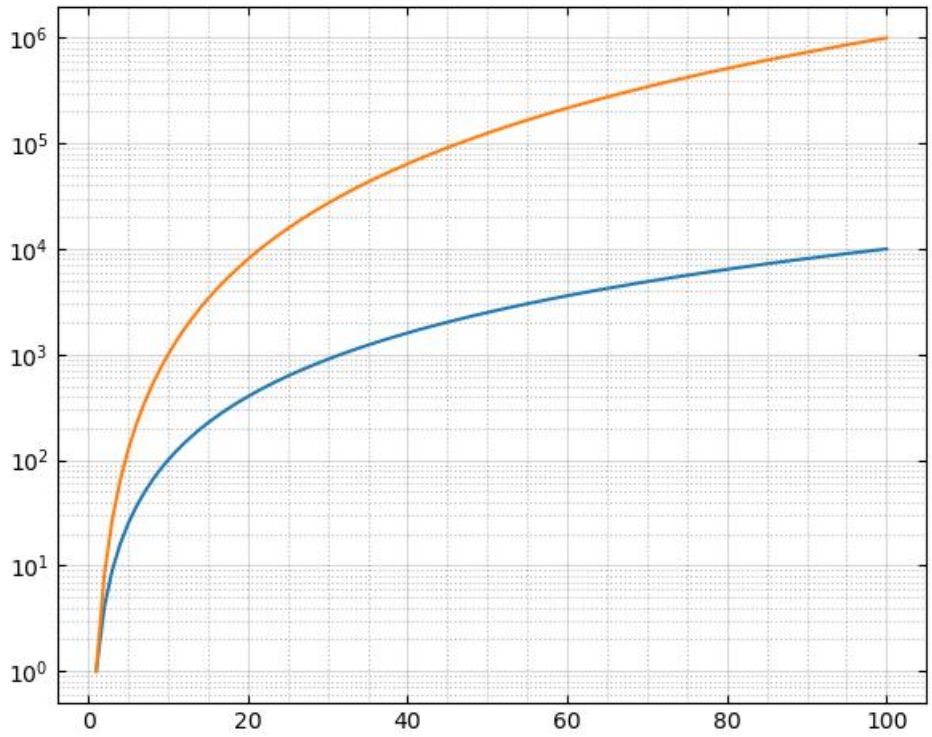
```
x = 1:100;
y = x.^2;
semilogy(x,y)
grid("on","both")
```



(3). 绘制多个线条

创建一个由 x 坐标组成的向量和两个由 y 坐标组成的向量。通过将以逗号分隔的 x-y 对组传递给 semilogy 绘制两个线条。

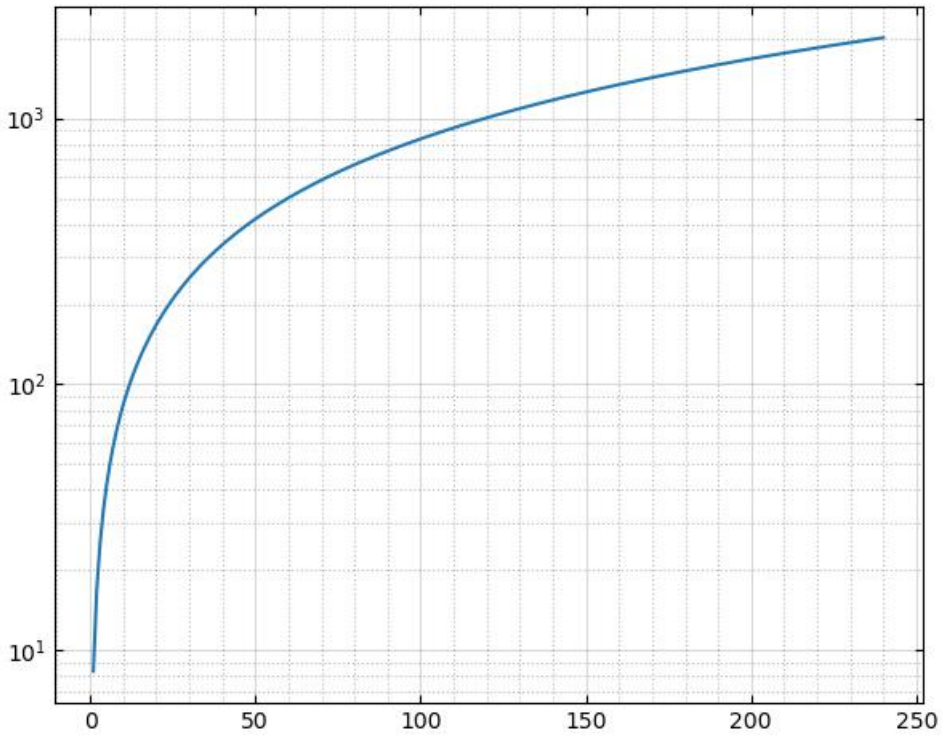
```
x = 1:100;
y1 = x.^2;
y2 = x.^3;
semilogy(x,y1,x,y2)
grid("on","both")
```



(4). 指定刻度位置、刻度标签和轴标签

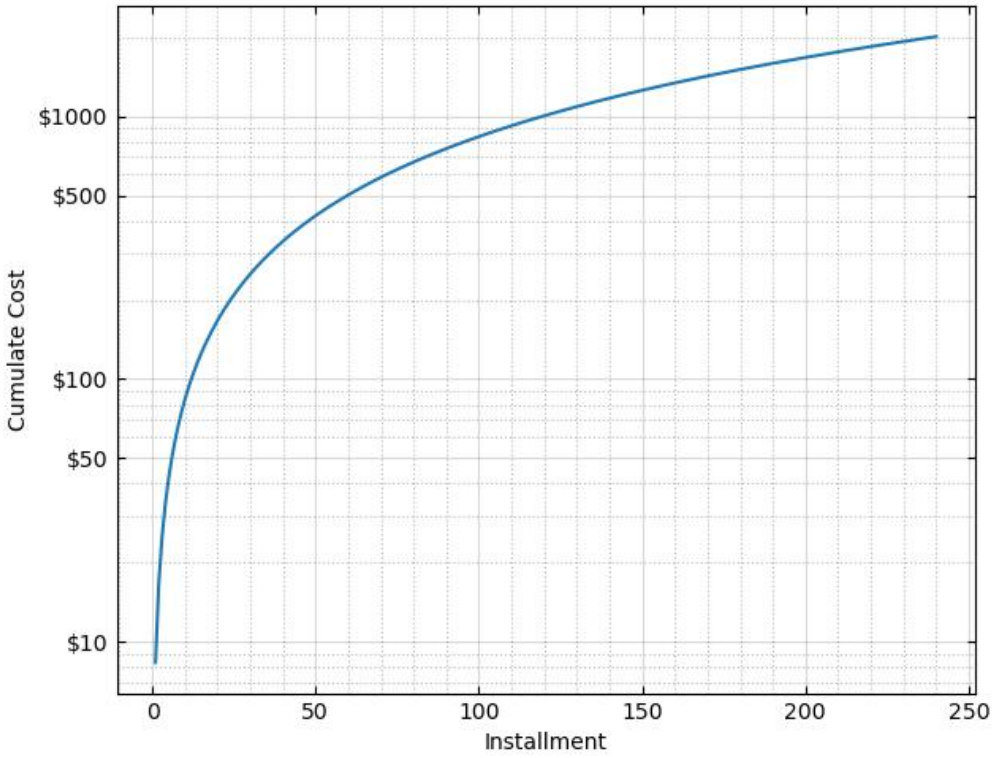
将向量 x 定义为 20 年期贷款的分期付款。将向量 y 定义为利率为 8% 的 1000 美元贷款的累积成本。绘制每期付款的累积成本。

```
P = 1000;
npayments = 240;
rate = 0.08/12;
mpayment = P*(rate*(1+rate)^npayments)/(((1+rate)^npayments) - 1);
x = 1:240;
y = x * mpayment;
semilogy(x,y);
grid("on","both")
```



通过调用 yticks 和 yticklabels 函数，更改 y 轴刻度值和刻度标签。然后通过调用 xlabel 和 ylabel 函数创建 x 和 y 轴标签。

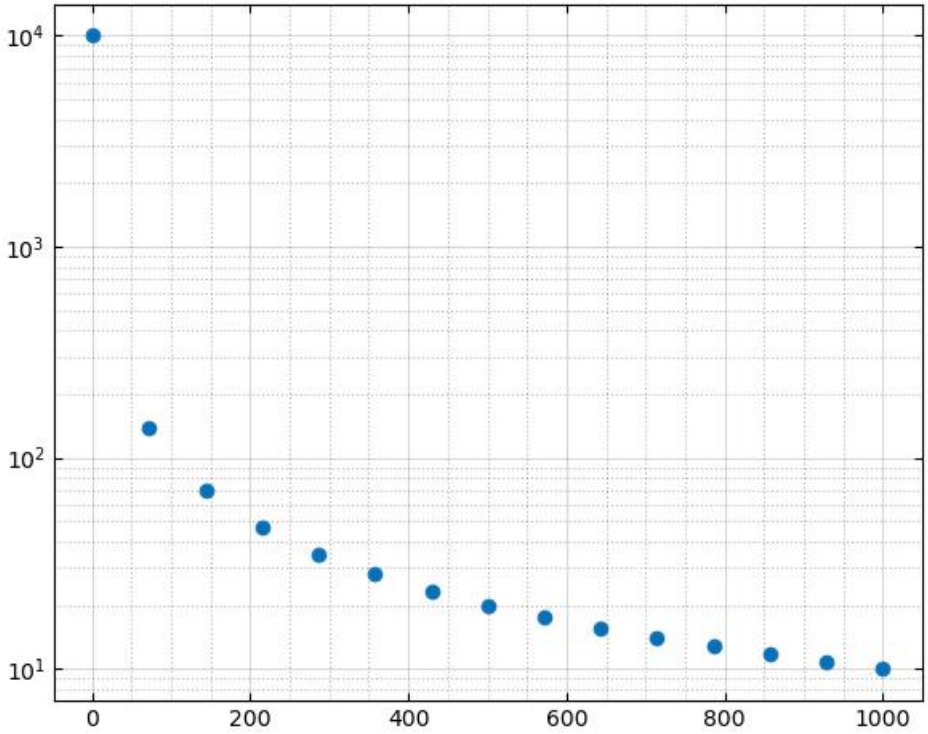
```
gca().set_yticks([10,50,100,500,1000])
yticklabels([raw"$10",raw"$50",raw"$100",raw"$500",raw"$1000"])
xlabel("Installment")
ylabel("Cumulate Cost")
```



(5). 将点绘制为不带线的标记

创建一组 x 和 y 坐标，并将它们绘制在一个对数线性图中。将线型指定为 "o" 以显示不带连接线的圆形标记。将标记填充颜色指定为 RGB 三元组 [0 0.447 0.741]，它对应于深蓝色。

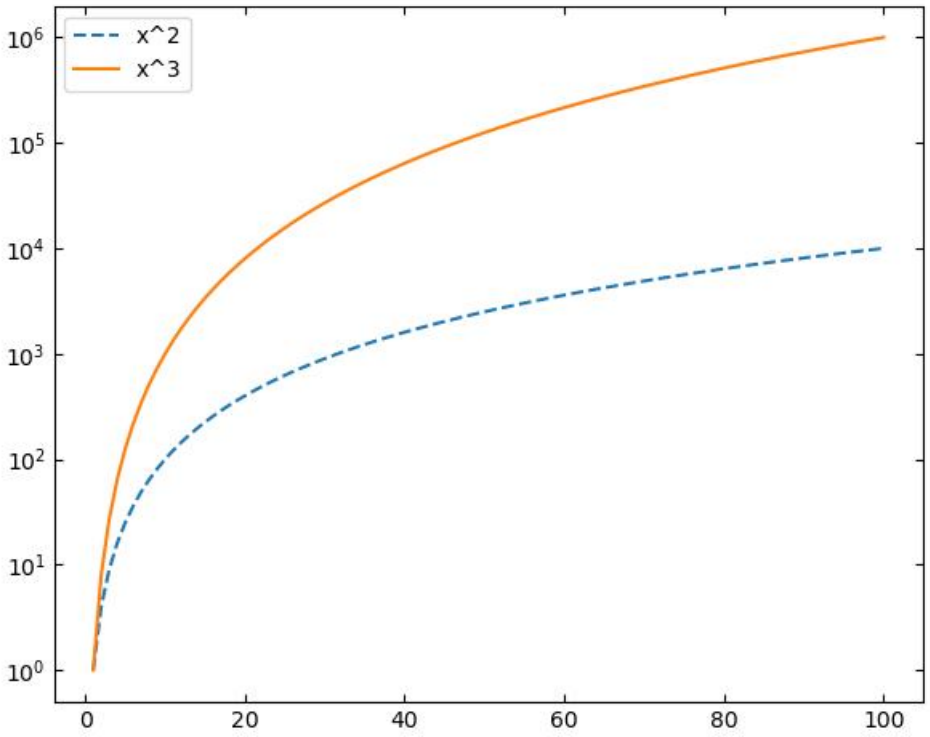
```
x = LinRange(1,1000,15);
y = (1 ./x) * 10000;
semilogy(x,y,"o", markerfacecolor = [0,0.447,0.741])
grid("on","both")
```



(6). 添加图例

创建两组 x 坐标和 y 坐标，并将其显示在一个对数线性图中。为第一组坐标指定一条虚线。然后通过调用 legend 函数并将位置指定为 "northwest"，在绘图的左上角显示图例。

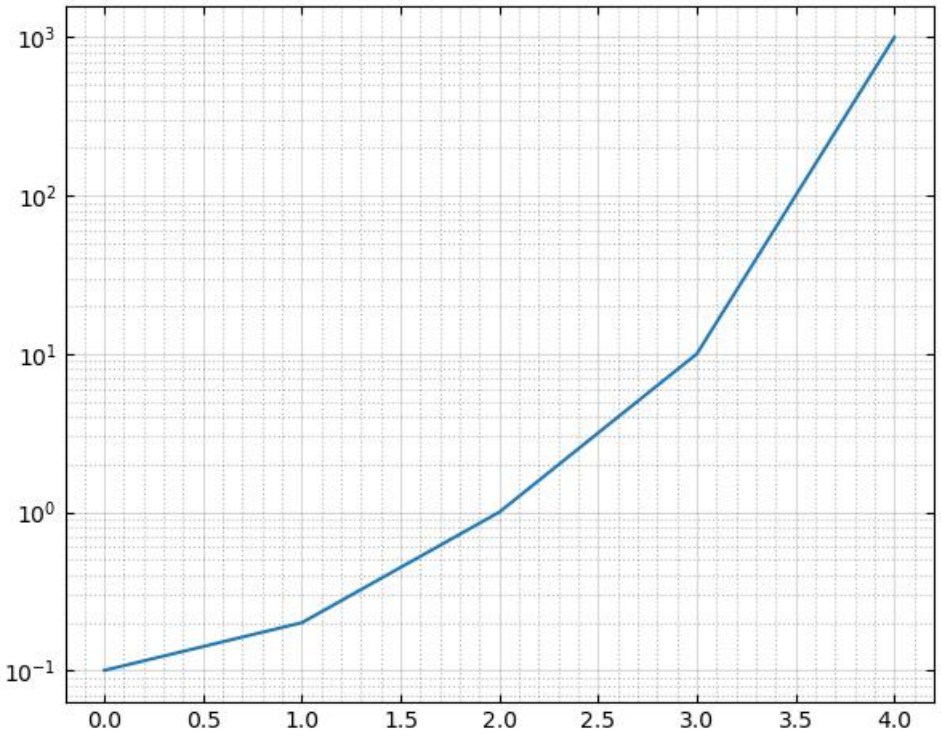
```
x = 1:100;
y1 = x.^2;
y2 = x.^3;
semilogy(x,y1,"--",x,y2)
legend(["x^2","x^3"],loc="northwest")
```



(7). 仅指定 y 坐标

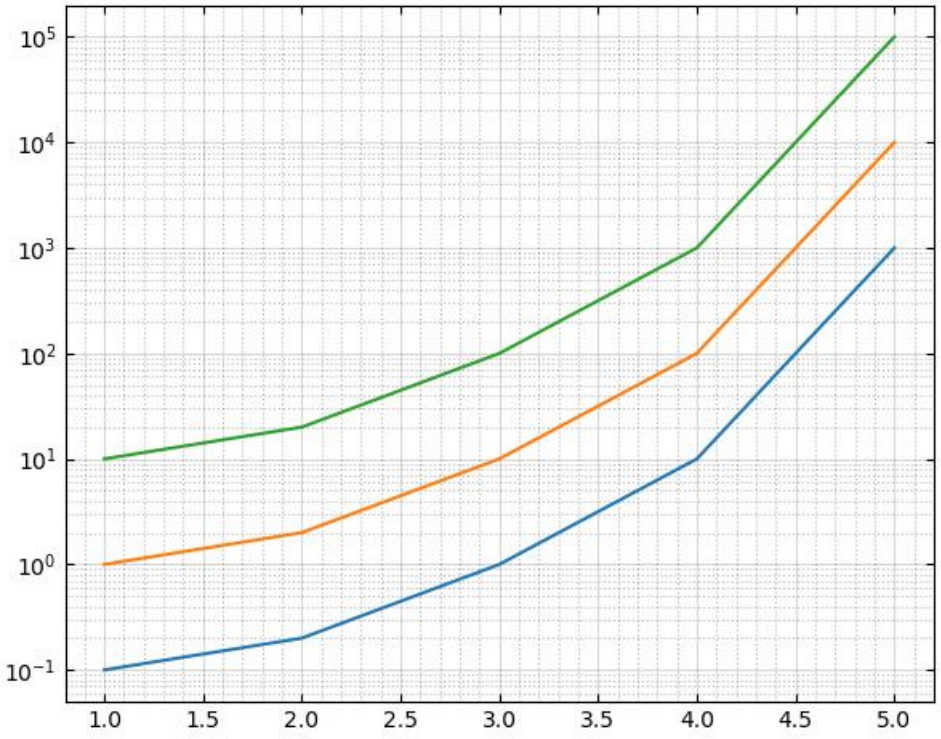
当您仅指定一个坐标向量时，semilogy 会绘制这些坐标对值 1:length(y) 的图。例如，将 y 定义为一个由 5 个值组成的向量。创建 y 的对数线性图。

```
y = [0.1, 0.2, 1, 10, 1000];
semilogy(y)
grid("on","both")
```



如果您将 y 指定为矩阵，则绘制 y 的列对值 1:size(y,1) 的图。例如，将 y 定义为 5×3 矩阵，并将其传递给 semilogy 函数。生成的图包含 3 个线条，其中每个线条的 x 坐标都是从 0 到 4。

```
y = [ 0.1   1   10
0.2   2   20
1.0   10   100
10    100   1000
1000   10000 100000];
semilogy([1,2,3,4,5], y)
grid("on","both")
```



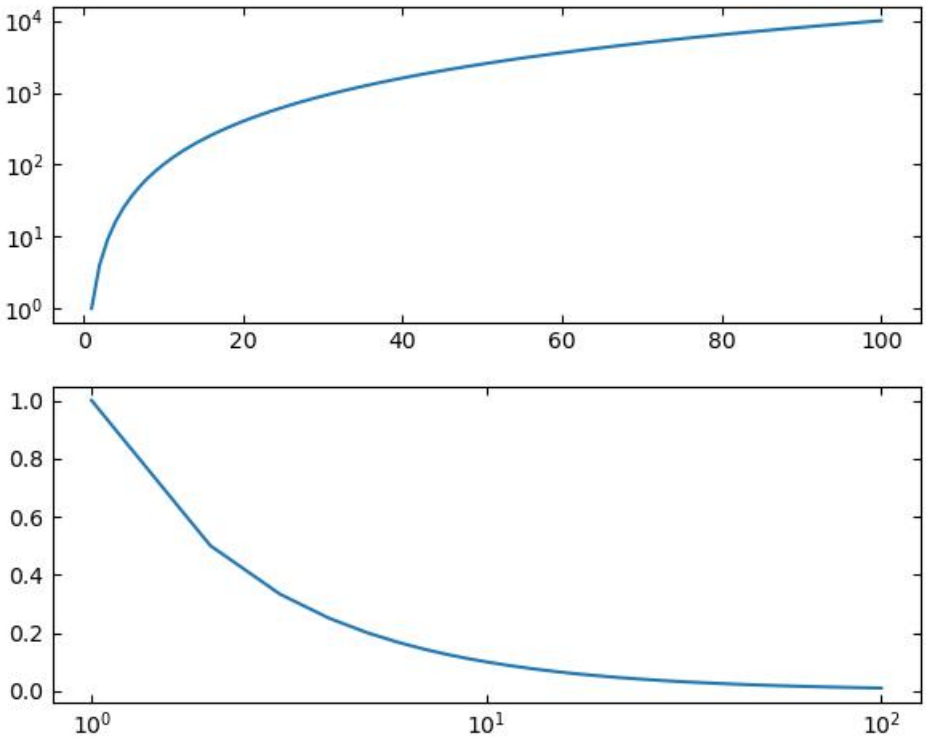
(8). 指定目标坐标区

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，以便坐标区填充布局中的可用空间。接下来，将该对象返回为 ax1。然后通过将 ax1 传递给 semilogy 函数来显示对数线性图。

```
ax1 = subplot(2,1,1);
x = 1:100;
y1 = x.^2;
semilogy(ax1,x,y1)
```

重复该过程以创建第二个线性对数图。

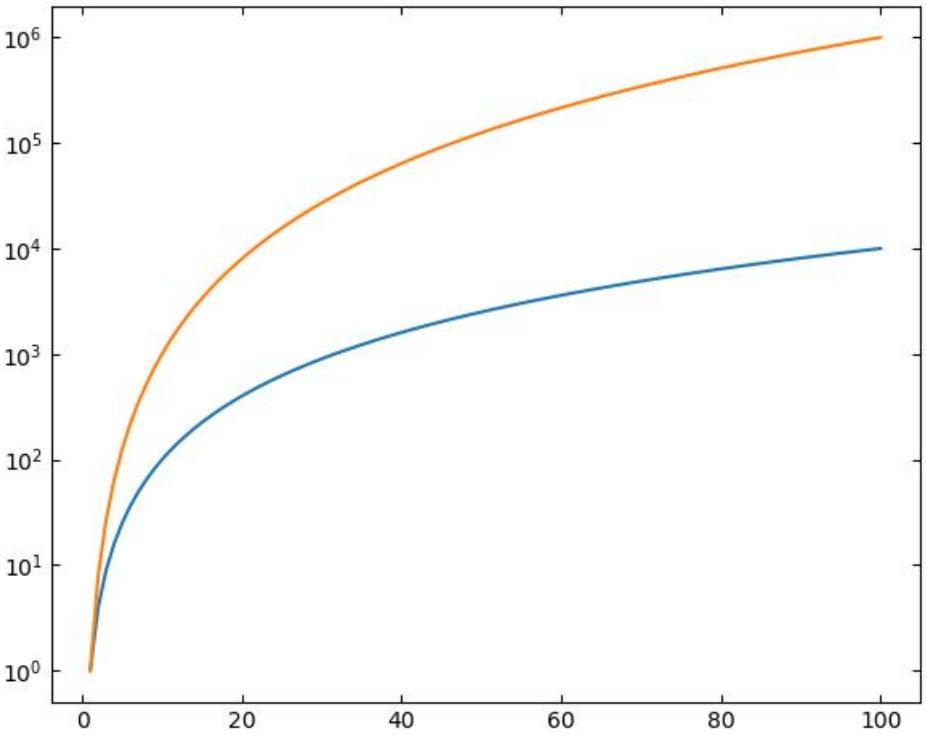
```
ax2 = subplot(2,1,2)
y2 = 1 ./ x
semilogx(ax2, x, y2)
```



(9). 绘制后更改线条外观

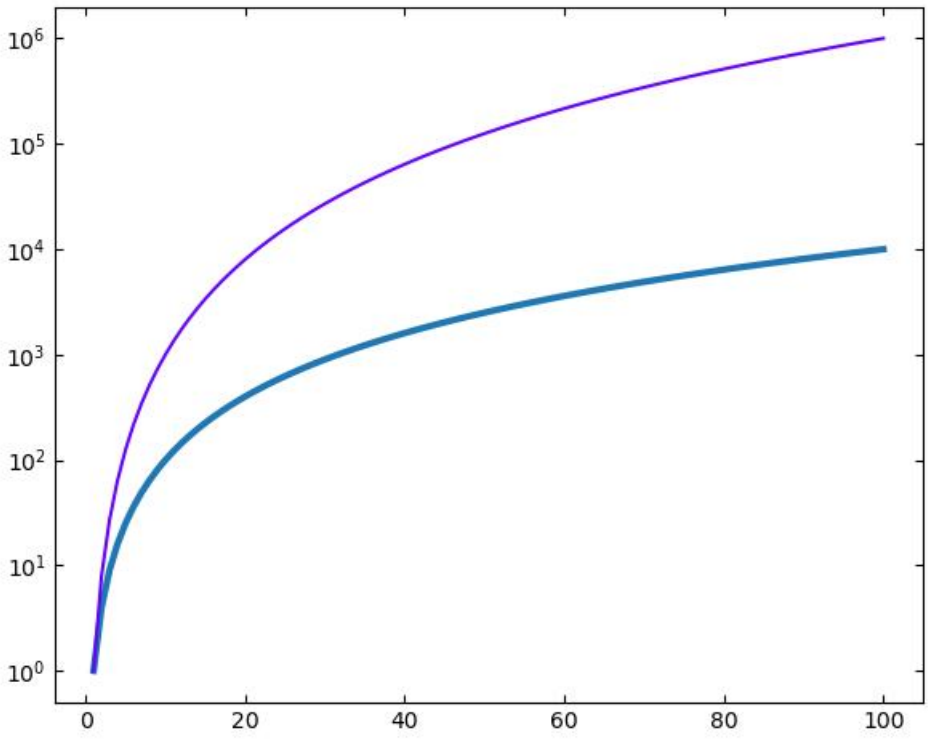
创建一个包含两个线条的对数线性图，并以变量 slg 形式返回线条对象。

```
x = 1:100;
y1 = x.^2;
y2 = x.^3;
slg = semilogy(x,y1,x,y2);
```



将第一个线条的宽度更改为 3，并将第二个线条的颜色更改为紫色。

```
slg[1].set_linewidth(3)
slg[2].set_color([0.4,0,1])
```

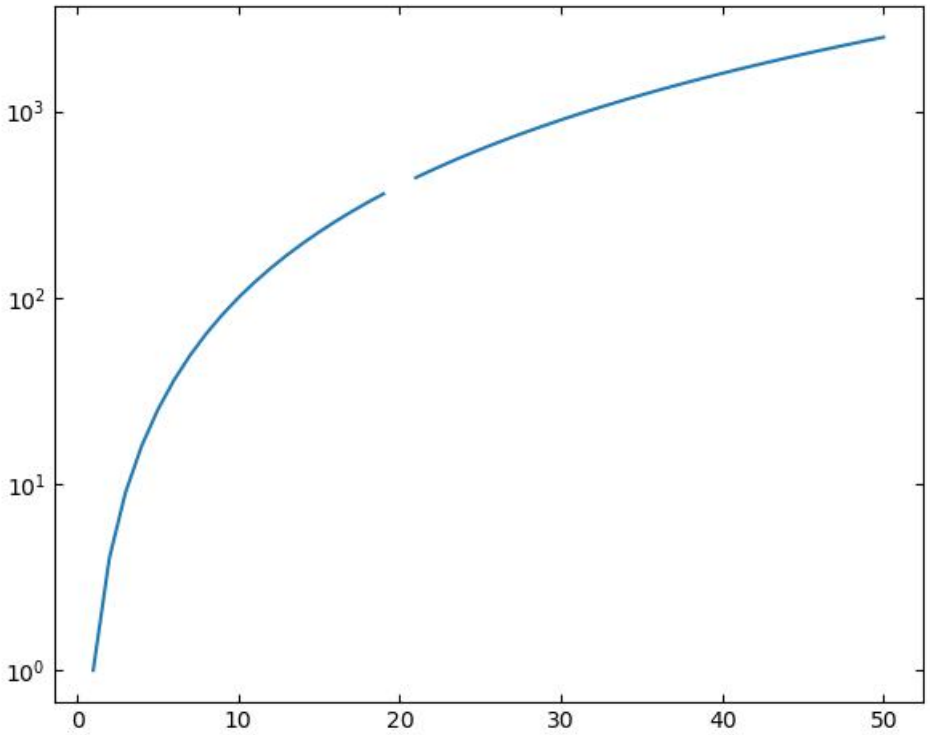


(10). 绘制不连续函数

在数据中任何不连续的位置插入 NaN 值。semilogy 函数显示这些位置的空缺。

创建一对 x 坐标向量和 y 坐标向量。用 NaN 值替换第二十个 y 坐标。然后创建 x 和 y 的对数线性图。

```
x = 1:50;
y = x.^2.0;
y[20] = NaN;
semilogy(x,y)
```



3.3.1.1.3 函数图

函数名	简介
fplot	绘制表达式或函数
fimplicit	绘制隐函数
fplot3	三维参数化曲线绘图函数

fplot

绘制表达式或函数

A. 语法

```
fplot(f)
fplot(f,xinterval)

fplot(funx,funy)
fplot(funx,funy,tinterval)



fplot(___,Fmt)
fplot(___,Key=Value)
fplot(ax,___)
fp = fplot(___)
```

B. 说明

fplot(f)在默认区间[-5 5]（对于 x）绘制由函数 y=f(x)定义的曲线。

fplot(f,xinterval)将在指定区间绘图。将区间指定为[xminxmax]形式的二元素向量。

fplot(funx,funy)在默认区间[-5 5]（对于 t）绘制由 x=funx(t)和 y=funy(t)定义的曲线。

fplot(funx,funy,tinterval) 将在指定区间绘图。将区间指定为 [tmin tmax] 形式的二元素向量。

fplot(___,Fmt) 指定线型、标记符号和线条颜色。例如，"-r" 绘制一根红色线条。在前面语法中的任何输入参数组合后使用此选项。

fplot(___,Key=Value)使用一个或多个名称-值对组参数指定线条属性。例如，linewidth=2 指定 2 磅的线宽。

fplot(ax,___) 将图形绘制到 ax 指定的坐标区中，而不是当前坐标区 (gca) 中。指定坐标区作为第一个输入参数。

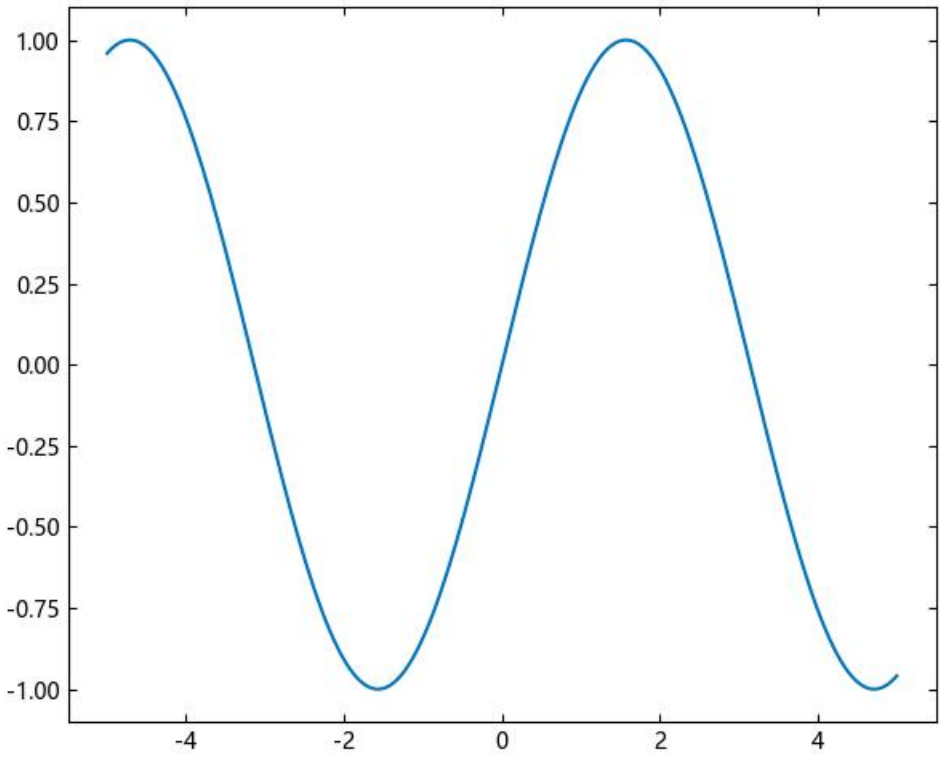
fp=fplot(___)返回 Line2D 对象。使用 fp 查询和修改特定线条的属性。有关属性列表，请参阅 Line2D 属性。

C. 示例

(1). 绘制表达式

在 x 的默认区间 [-5 5] 绘制 sin(x)。

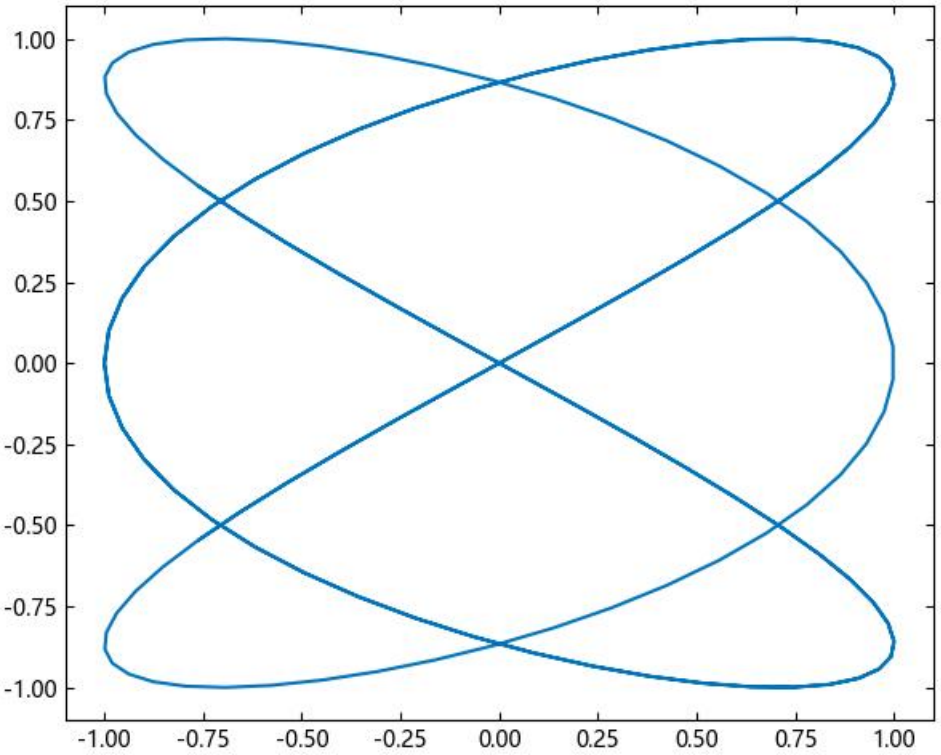
```
f = (x)->sin(x)
fplot(f)
```



(2). 绘制参数曲线

绘制参数化曲线 $x = \cos(3t)$和 $y = \sin(2t)$。

```
xt = t->cos(3*t)
yt = t->sin(2*t)
fplot(xt, yt)
```



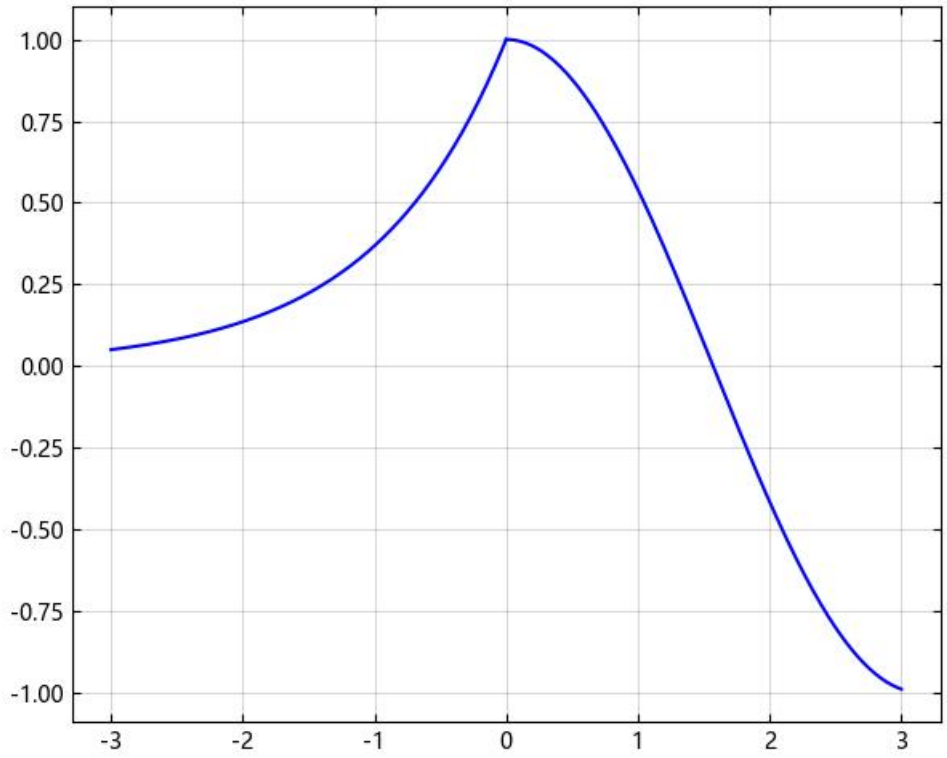
(3). 指定绘图区间并绘制分段函数

绘制分段函数

$$\begin{cases} e^x, -3 < x < 0 \\ \cos(x), 0 < x < 3 \end{cases}$$

使用 hold("on") 绘制多个线条。使用 fplot 的第二个输入参数指定绘图区间。使用 "b" 将绘制的线条颜色指定为蓝色。在相同坐标区中绘制多个线条时，坐标轴范围会调整以容纳所有数据。

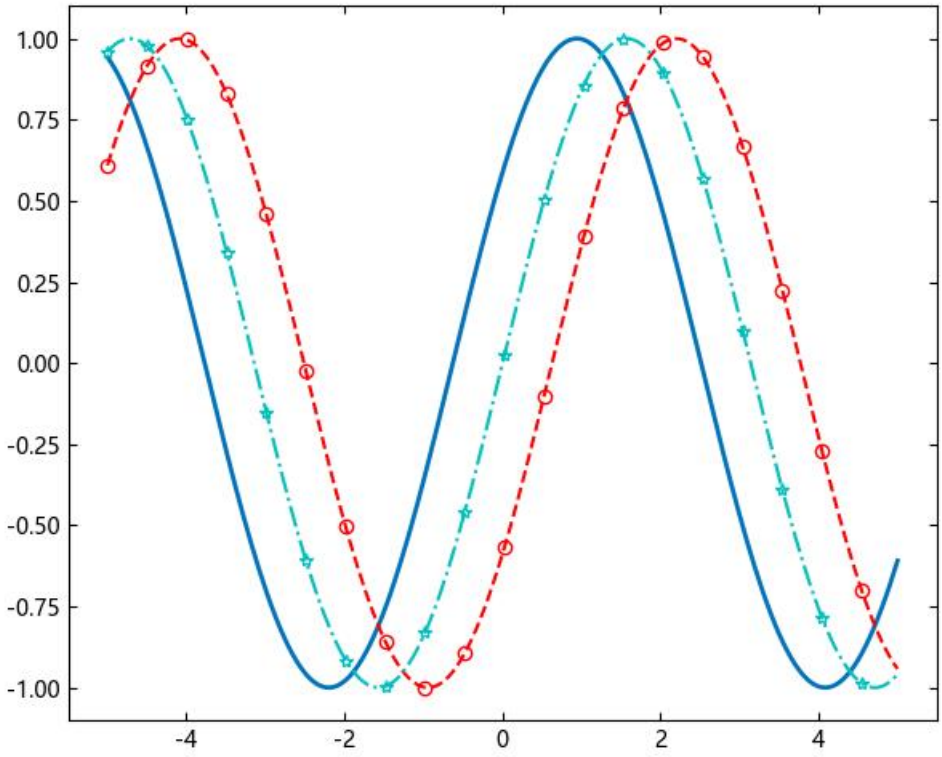
```
fplot(x->exp(x),[-3 0], "b")
hold("on")
fplot(x->cos(x),[0 3], "b")
hold("off")
grid("on")
```



(4). 指定线条属性并显示标记

绘制具有不同相位的三个正弦波。对于第一个，使用 2 磅的线宽。对于第二个，指定带有圆形标记的红色虚线线型。对于第三个，指定带有星号标记的青蓝色点划线线型。

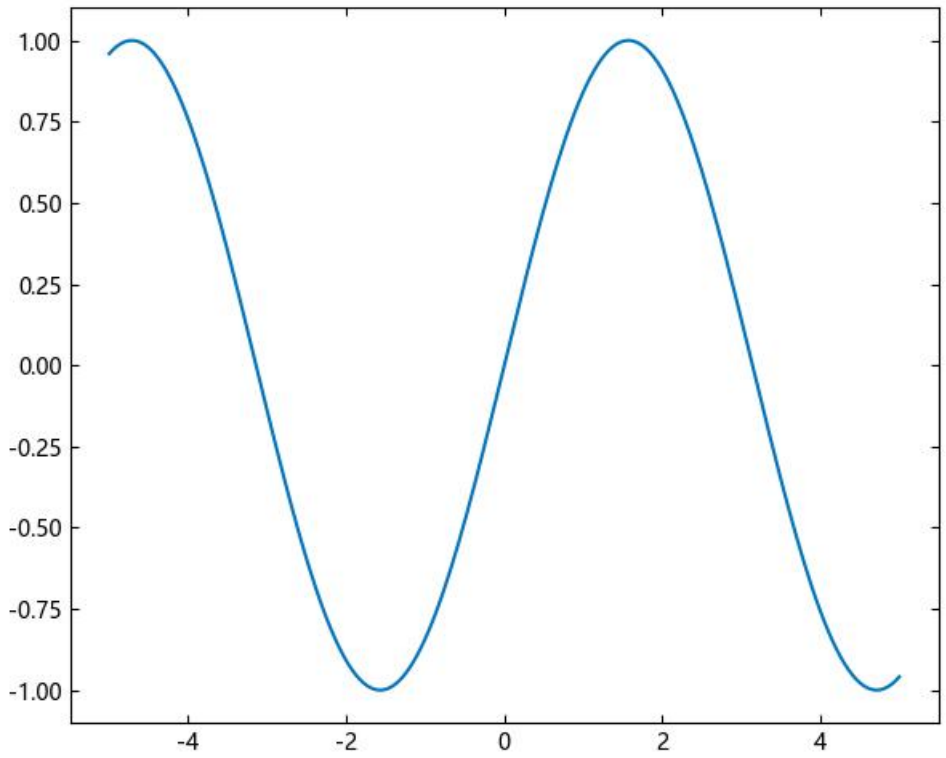
```
fplot(x->sin(x + pi/5), linewidth = 2)
hold("on")
fplot(x->sin(x - pi/5), [-5, 5], "--or", markevery=10)
fplot(x->sin(x), [-5, 5], "-.*c", markevery=10)
hold("off")
```



(5). 创建后修改线条属性

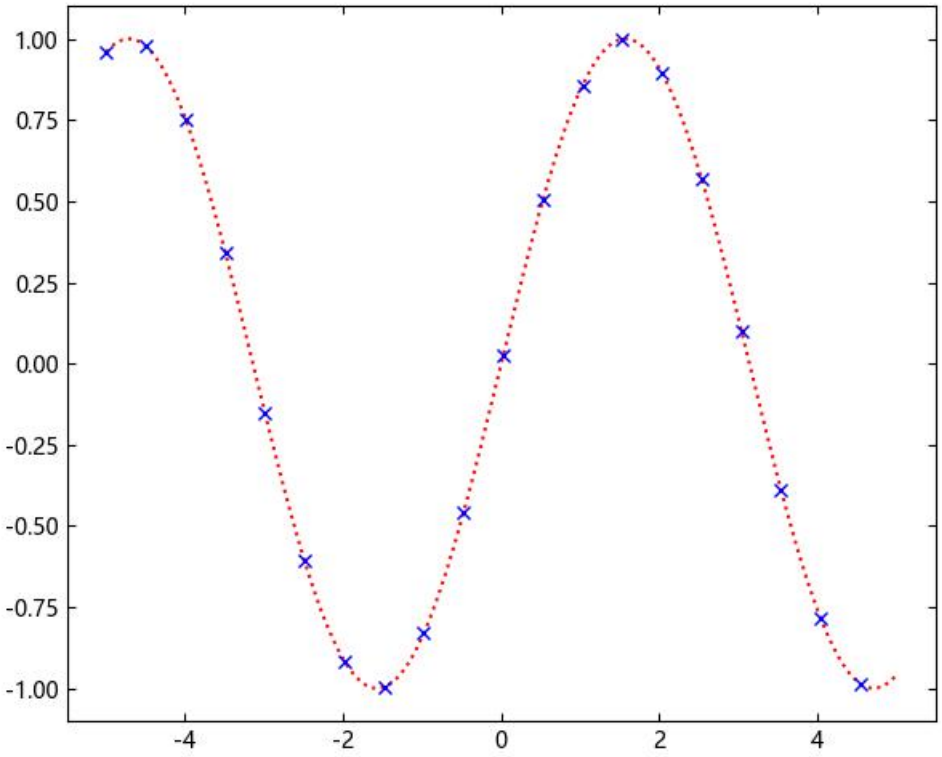
绘制 sin(x) 并将函数行对象指定给变量。

```
fp, = fplot(x->sin(x))
```



通过使用圆点表示法设置属性，将线条更改为红色点线。添加交叉标记，并将标记颜色设置为蓝色。

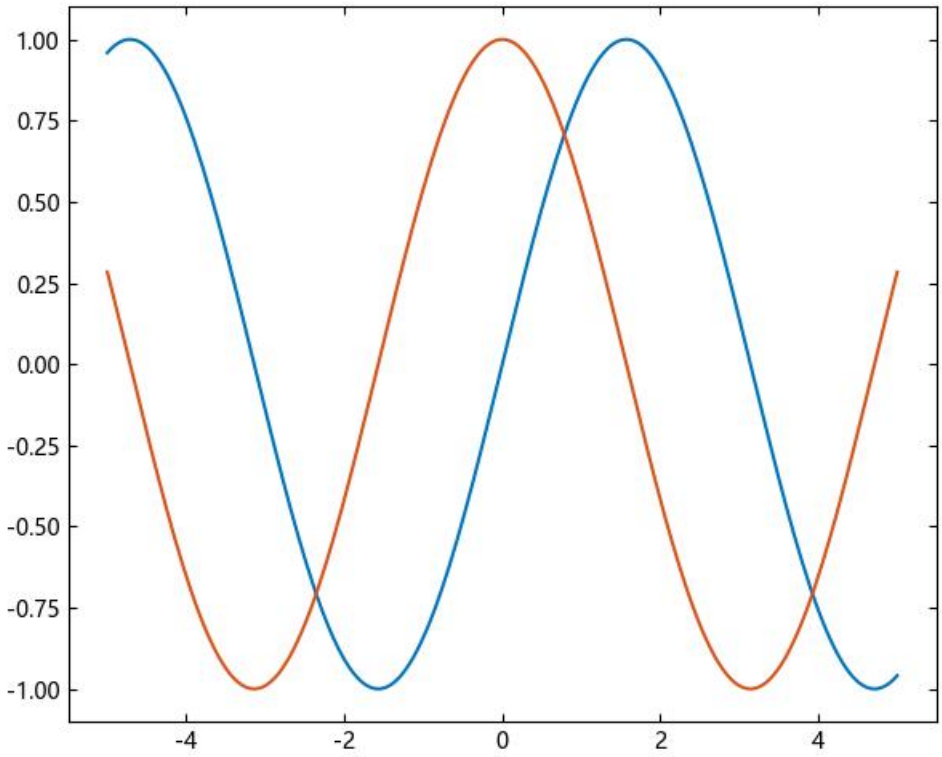
```
fp.set_linestyle(":")
fp.set_color("r")
fp.set_marker("x")
fp.set_markeredgecolor("b")
fp.set_markevery(10)
```



(6). 在相同坐标区中绘制多个线条

使用 hold("on") 绘制两个线条。

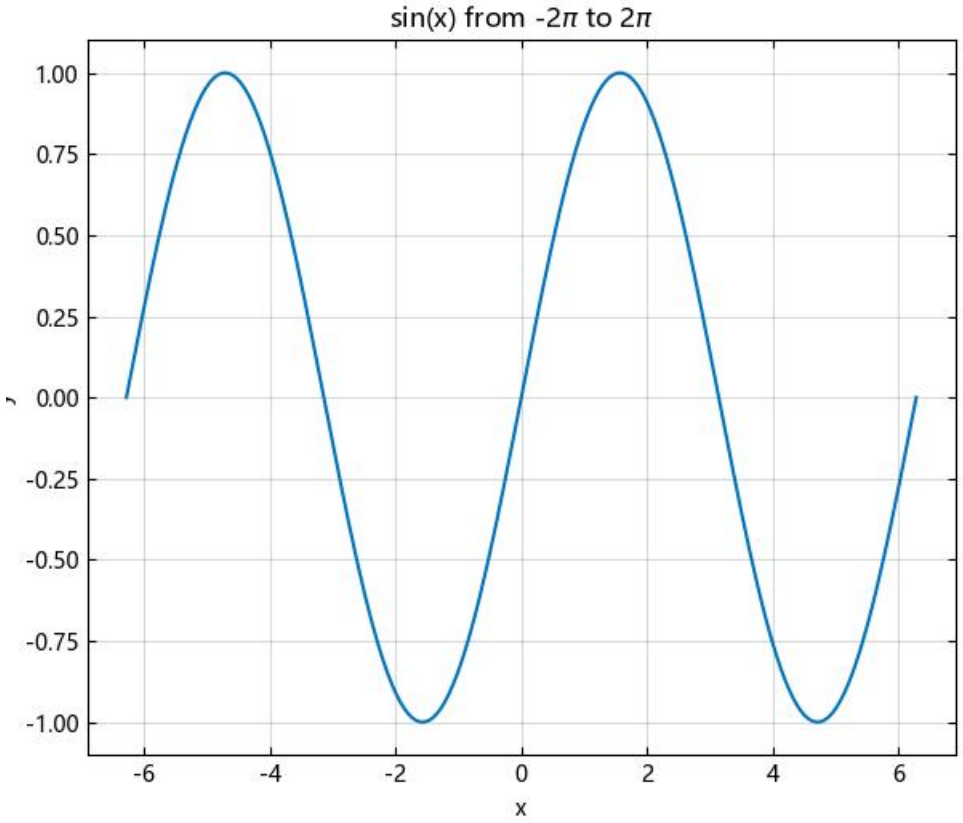
```
fplot(x->sin(x))
hold("on")
fplot(x->cos(x))
hold("off")
```



(7). 添加标题和轴标签以及格式化刻度

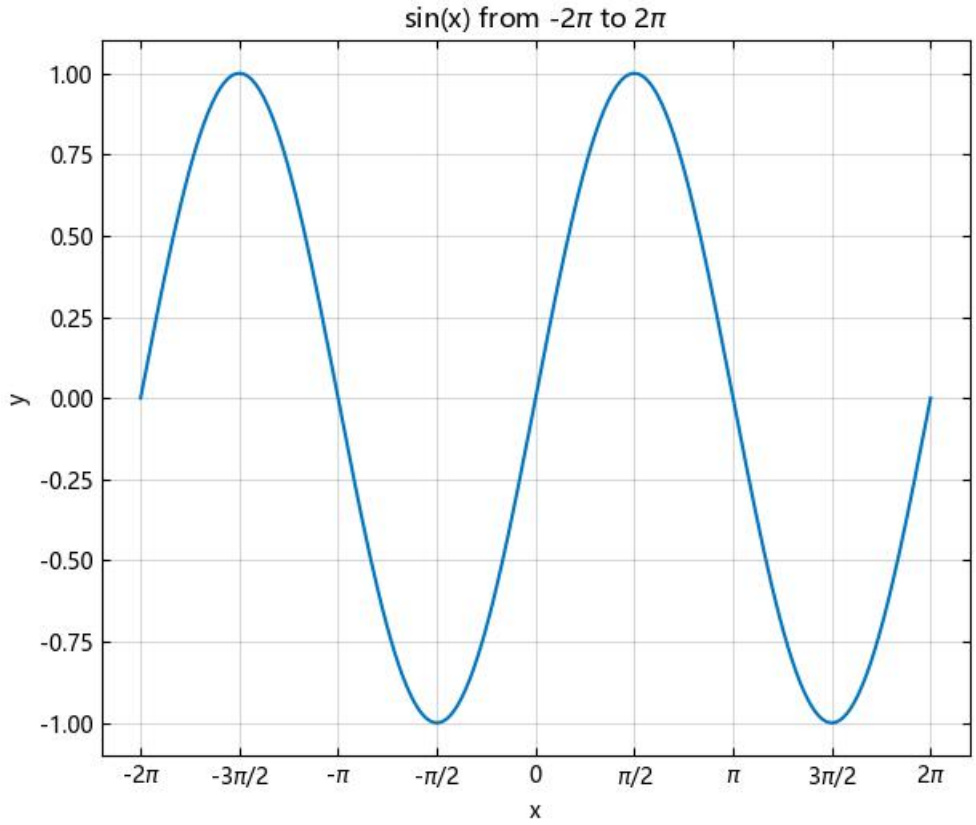
使用函数句柄从 −2π 到 2π 绘制 sin(x)。网格线的显示方式。然后添加一个标题，并为 x 轴和 y 轴添加标签。

```
fplot(sin, [-2*pi 2*pi])
grid("on")
title(raw"sin(x) from -2$\pi$ to 2$\pi$")
xlabel("x");
ylabel("y");
```



使用 gca 访问当前坐标区对象。沿 x 轴以 π/2 为间隔显示刻度线。通过设置坐标区对象的 XTick 和 XTickLabel 属性，格式化 x 轴刻度值。y 轴存在类似属性。

```
ax = gca();
xticks(ax, -2*pi:pi/2:2*pi);
xticklabels(ax,       [raw"-2$\pi$",raw"-3$\pi$/2",raw"-$\pi$",raw"-
$\pi$/2","0",raw"$\pi$/2",raw"$\pi$",raw"3$\pi$/2",raw"2$\pi$"]);
```



fimplicit

绘制隐函数

A. 语法

```
fimplicit(f)
fimplicit(f,xinterval)
fimplicit(ax,___)

fimplicit(___,Key=Value)
fp = fimplicit(___)
```

B. 说明

fimplicit(f)在默认区间[-55]（对于 x 和 y）上绘制 f(x,y)=0 定义的隐函数。

fimplicit(f,interval)为 x 和 y 指定绘图区间。

fimplicit(ax,___) 将在 ax 指定的坐标区中，而不是在当前坐标区中绘制图形。将坐标区指定为第一个输入参数，先于上述任何输入参数。

fimplicit(___,Key=Value)使用一个或多个名称-值对组参数指定线条属性。例如，linewidth=2 指定 2 磅的线宽。

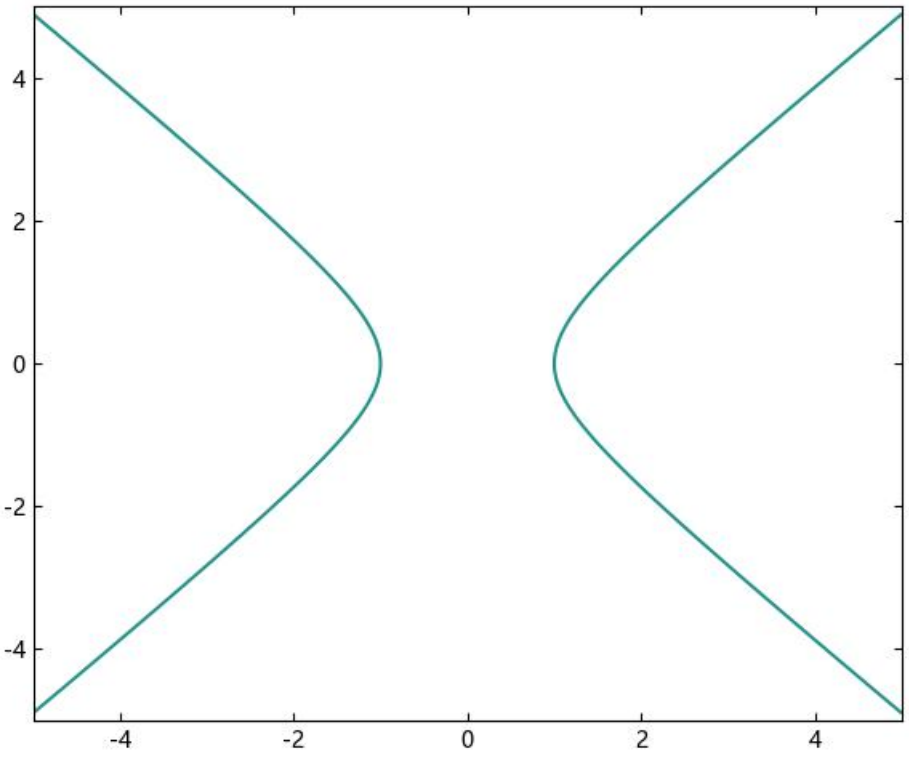
fp = fimplicit(___) 返回等高线图对象。使用 fp 查询和修改特定线条的属性。有关属性列表，请参阅等高线图属性。

C. 示例

(1). 绘制隐函数

在 x 和 y 的默认区间 [-5 5] 绘制由函数 $x^2 - y^2 - 1 = 0$ 描述的双曲线。

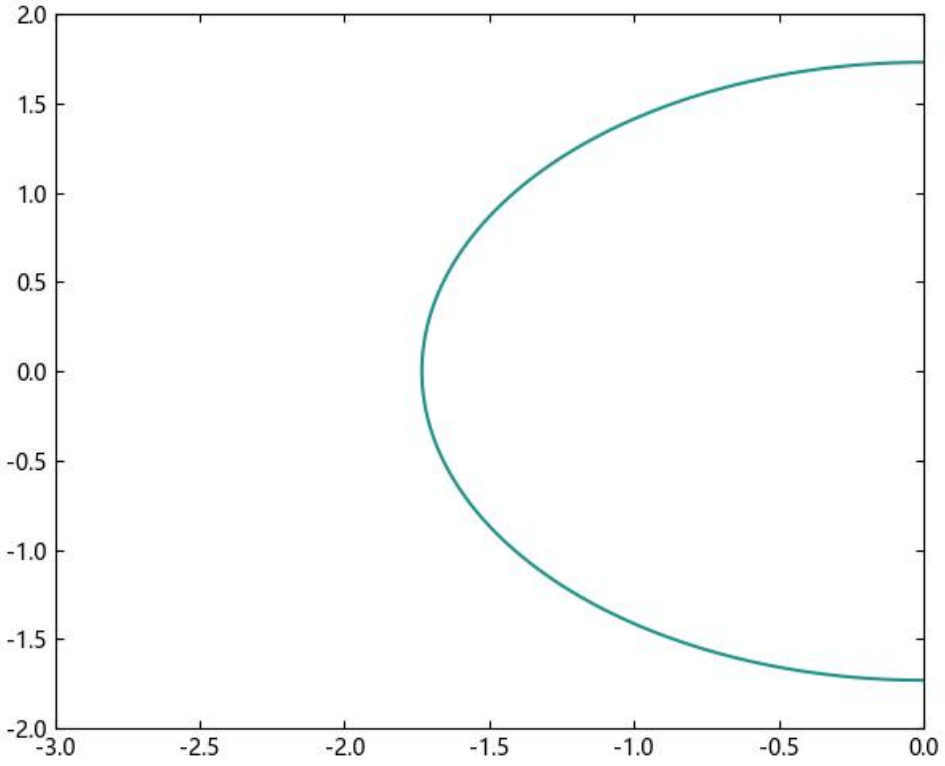
```
f = (x, y) -> x .^ 2 - y .^ 2 - 1
fimplicit(f)
```



(2). 指定绘图区间

在 x 的 [-3 0] 区间和 y 的 [-2 2] 区间绘制函数 $x^2 + y^2 - 3 = 0$ 。

```
f = (x, y) -> x .^ 2 + y .^ 2 - 3;
fimplicit(f, [-3 0 -2 2])
```



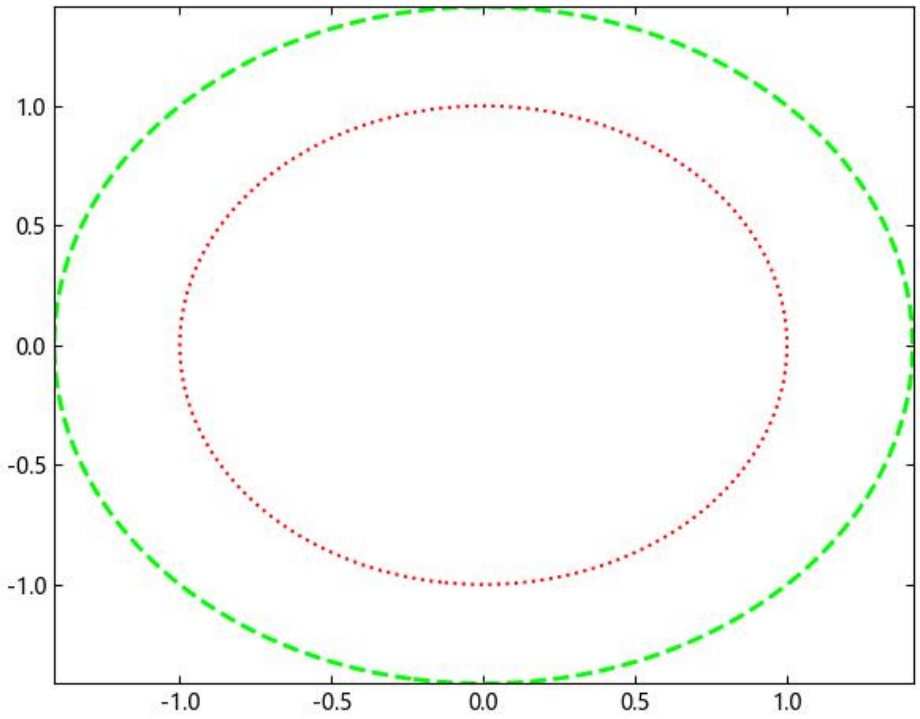
(3). 修改隐式绘图的外观

使用不同的半径值绘制两个以 (0,0) 为中心的圆。第一个圆使用红色点线。第二个圆使用线宽为 2 磅的绿色虚线。

```
f1 = (x, y) -> x .^ 2 + y .^ 2 - 1;
fimplicit(f1; linestyle=":", color="r")


hold("on")
f2 = (x, y) -> x .^ 2 + y .^ 2 - 2;
fimplicit(f2; linestyle="--", color="#00FF00", linewidth=2)
hold("off")

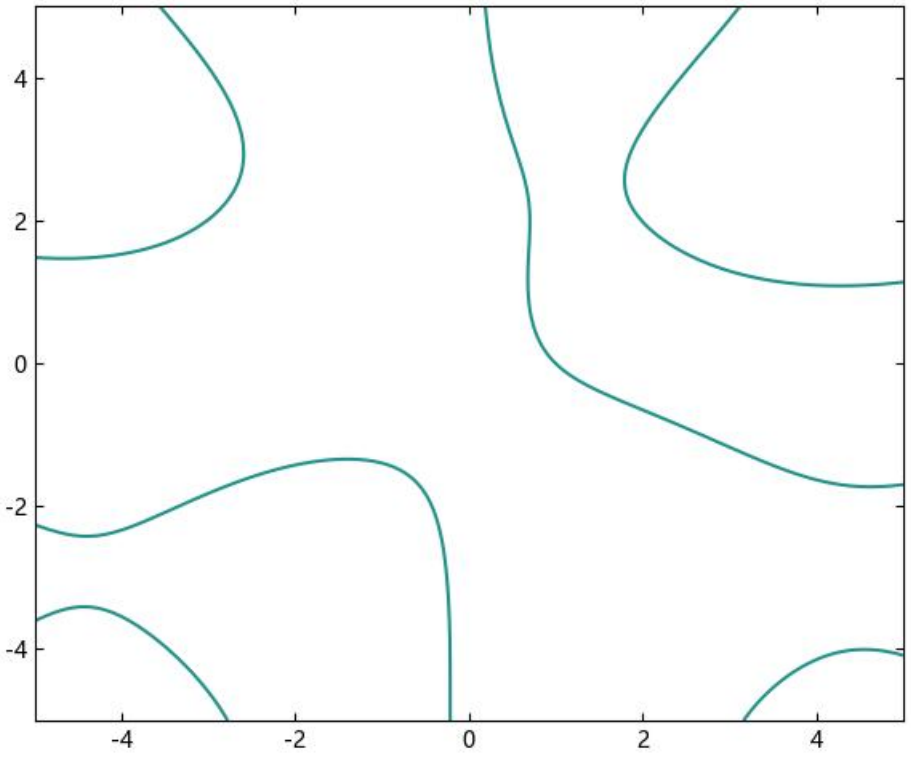
ax = gca()
ax.set_xlim(-sqrt(2), sqrt(2));
ax.set_ylim(-sqrt(2), sqrt(2));
```



(4). 创建后修改隐式绘图

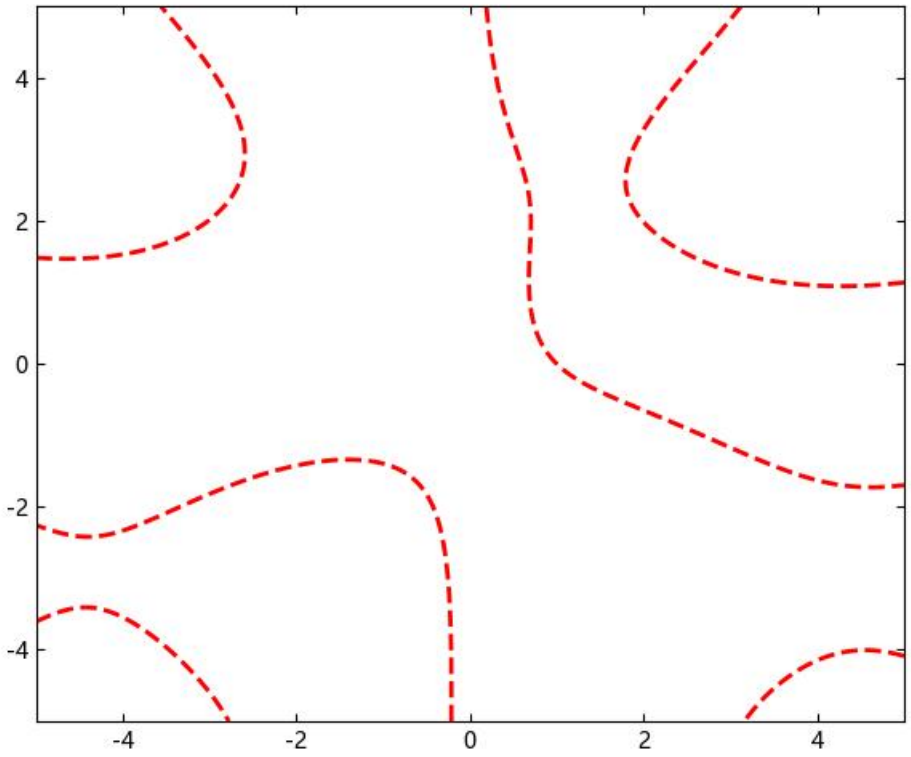
绘制隐函数 并将隐函数行对象赋给变量 fp。

```
fp = fimplicit((x, y) -> y .* sin(x) + x .* cos(y) - 1)
```



使用 fp 可在创建隐函数行对象后访问并修改其属性。例如，更改颜色、线型和线宽。

```
for c in fp.collections
    c.set_edgecolor("r")
    c.set_linestyle("--")
    c.set_linewidth(2)
end
```



fplot3

三维参数化曲线绘图函数

A. 语法

```
fplot3(funx,funy,funz)
fplot3(funx,funy,funz,tinterval)

fplot3(___,Fmt)
fplot3(___,Key=Value)
fplot3(ax,___)
fp = fplot3(___)
```

B. 说明

fplot3(funx,funy,funz)在默认区间[-5,5]（对于 t）绘制由 x=funx(t)、y=funy(t)和 z=funz(t)定义的参数化曲线。

fplot3(funx,funy,funz,tinterval)将在指定区间绘图。将区间指定为[tmintmax]形式的二元素向量。

fplot3(___,Fmt) 指定线型、标记符号和线条颜色。例如，"-r" 绘制一根红色线条。在前面语法中的任何输入参数组合后使用此选项。

fplot3(___,Key=Value) 使用一个或多个名称-值对组参数指定线条属性。例如，linewidth = 2 指定 2 磅的线宽。

fplot3(ax,___) 将图形绘制到 ax 指定的坐标区中，而不是当前坐标区 (gca)中。指定坐标区作为第一个输入参数。

fp=fplot3(___)返回 Line3D 对象。使用 fp 查询和修改特定线条的属性。有关属性列表，请参阅 Line3D 属性。

C. 示例

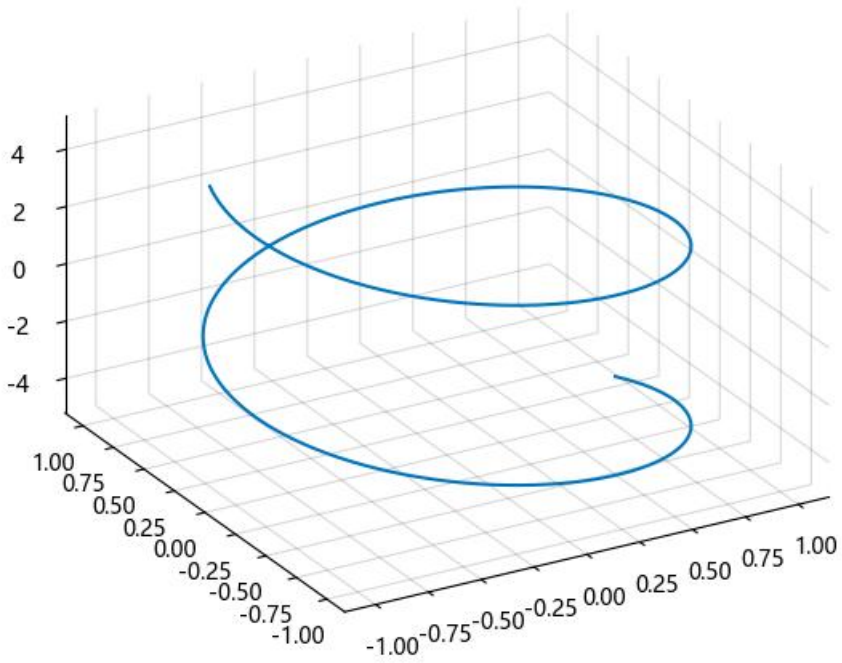
(1). 绘制三维参数化线条

绘制三维参数化线条。$\begin{cases} x = \sin(t) \\ y = \cos(t) \\ z = t \end{cases}$

（在默认参数范围 [-5 5] 内。）

```
xt = t->sin(t)
yt = t->cos(t)
zt = t->t

fplot3(xt, yt, zt)
```



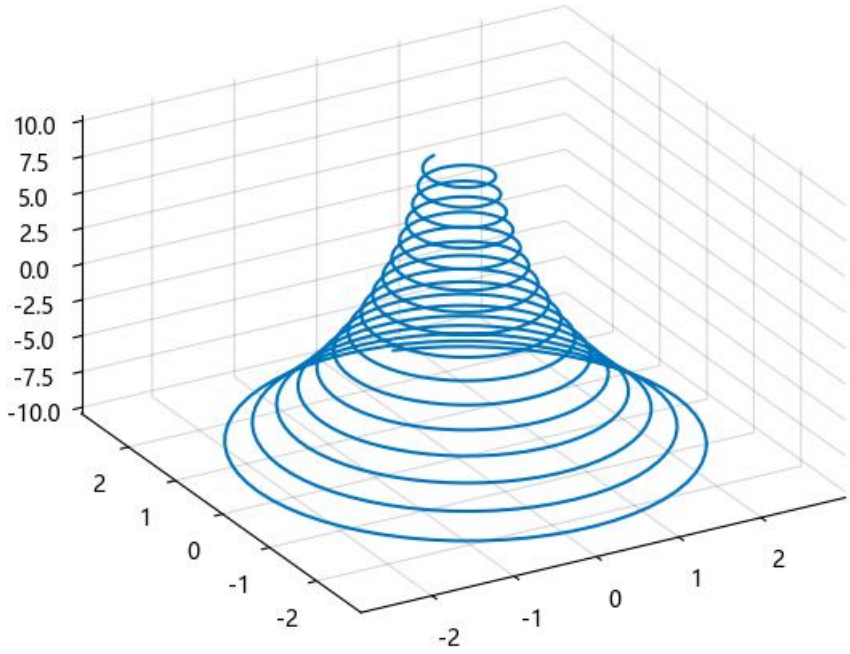
(2). 指定参数范围

绘制参数化线条。$\begin{cases} e^{-\frac{t}{10}} sin(5t) \\ e^{-\frac{t}{10}} cos(5t) \\ z = t \end{cases}$

（通过指定 fplot3 的第四个输入实参，在形参范围 [-10 10] 内绘制）。

```
xt = t->exp(-t/10).*sin(5*t);
yt = t->exp(-t/10).*cos(5*t);
zt = t->t

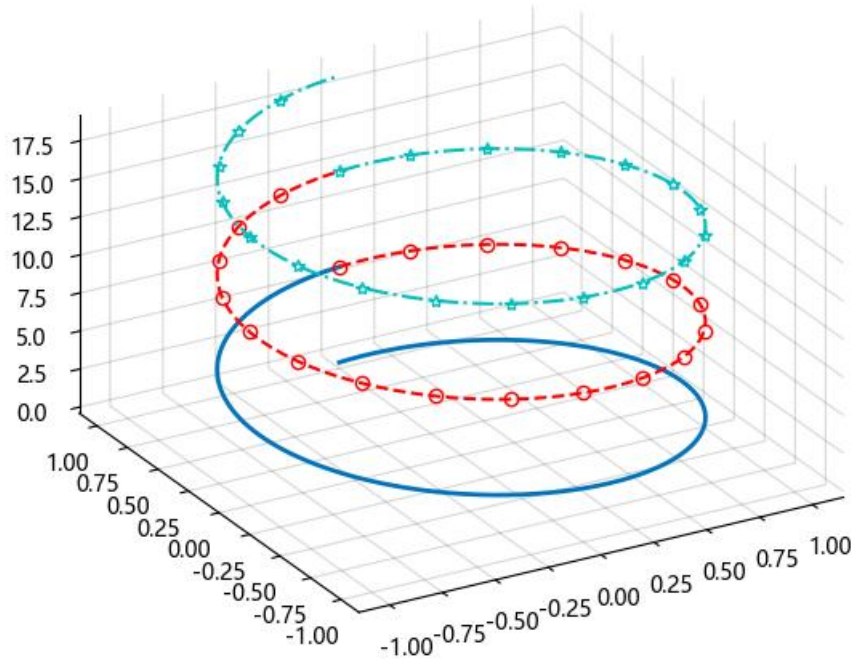
fplot3(xt, yt, zt, [-10 10])
```



(3). 指定线条属性并显示标记

在参数的不同区间，将同一条三维参数化曲线绘制三次。对于第一个区间，使用 2 磅的线宽。对于第二个，指定带有圆形标记的红色虚线线型。对于第三个，指定带有星号标记的青蓝色点划线线型。

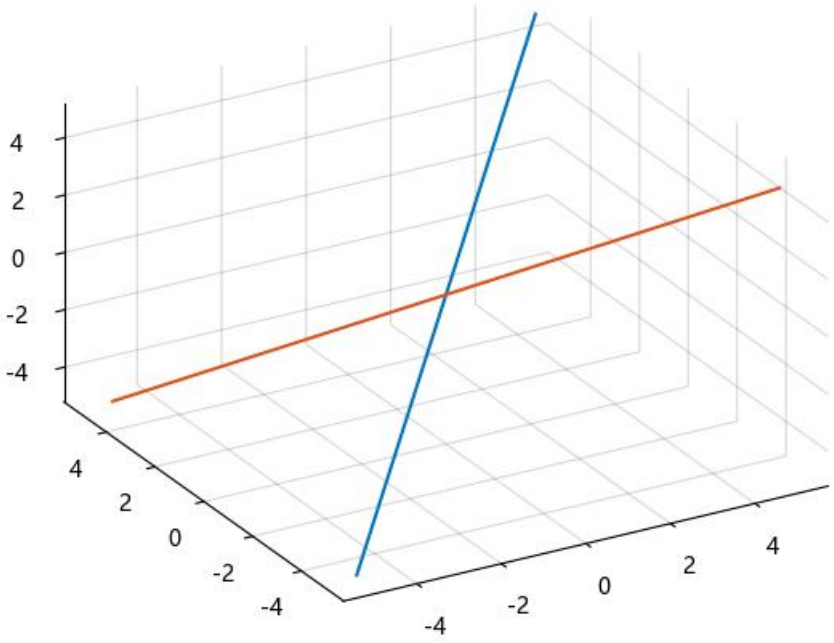
```
fplot3(t->sin(t), t->cos(t), t->t, [0 2*pi], linewidth = 2)
hold("on")
fplot3(t->sin(t),    t->cos(t),    t->t,    [2*pi    4*pi],    "--or",
markevery=50)
fplot3(t->sin(t),    t->cos(t),    t->t,    [4*pi    6*pi],    "-.*c",
markevery=50)
hold("off")
```



(4). 在相同坐标区中绘制多个线条

使用 hold("on") 在相同的坐标区中绘制多个线条。

```
fplot3(t->t, t->t, t->t)
hold("on")
fplot3(t->-t, t->t, t->-t)
hold("off")
```



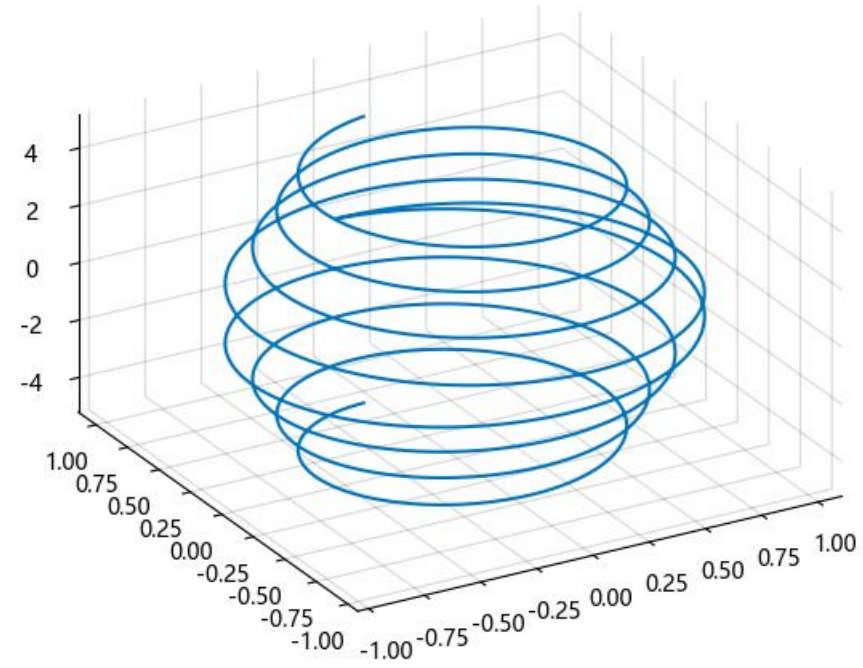
(5). 创建后修改三维参数化线条

绘制参数化线条。

将参数化函数行对象指定给变量。

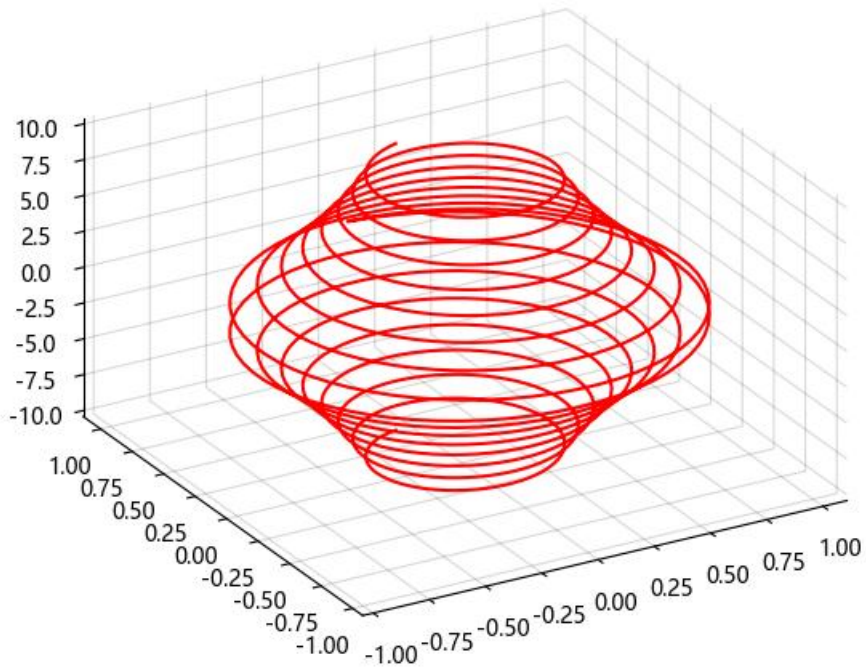
```
xt = t->exp(-abs(t)/10).*sin(5*abs(t));
yt = t->exp(-abs(t)/10).*cos(5*abs(t));
zt = t->t;

fp, = fplot3(xt,yt,zt)
```



将参数值范围更改为 [-10 10]，并将线条颜色更改为红色。

```
fp, = fplot3(xt,yt,zt,[-10,10])
fp.set_color("r")
```



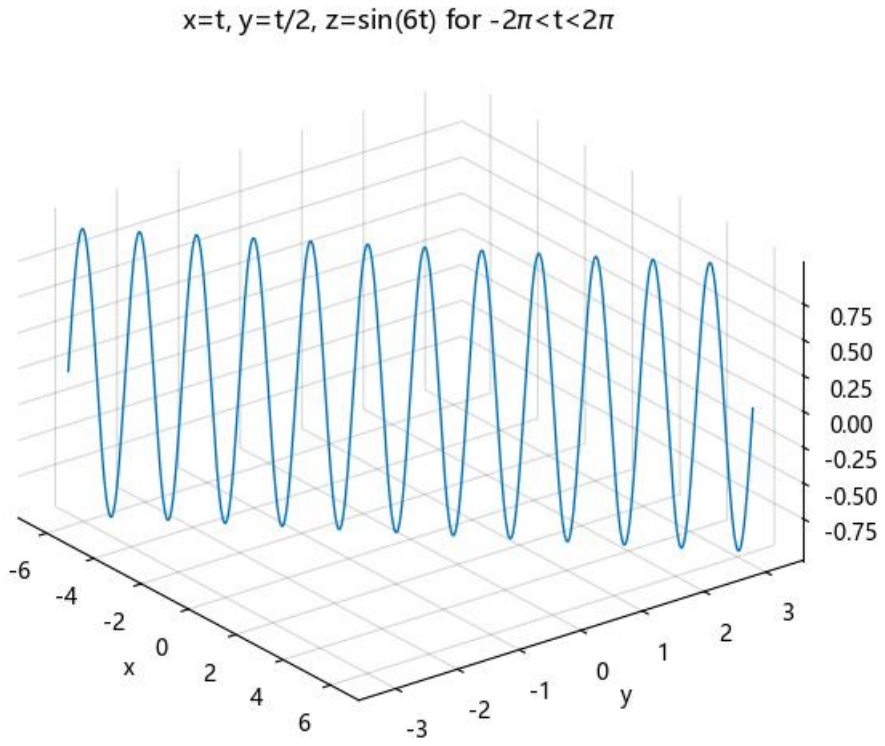
(6). 添加标题和轴标签以及格式化刻度

为从 −2π 到 2π 范围内的 t 值绘制参数化线条。$\begin{cases} x = t \\ y = t/2 \\ z = sin(6t) \end{cases}$

添加标题、x 轴标签和 y 轴标签。此外，还可以更改坐标区视图并显示坐标区框轮廓。

```
xt = t->t;
yt = t->t/2;
zt = t->sin(6*t);
fplot3(xt,yt,zt,[-2*pi 2*pi], linewidth = 1);

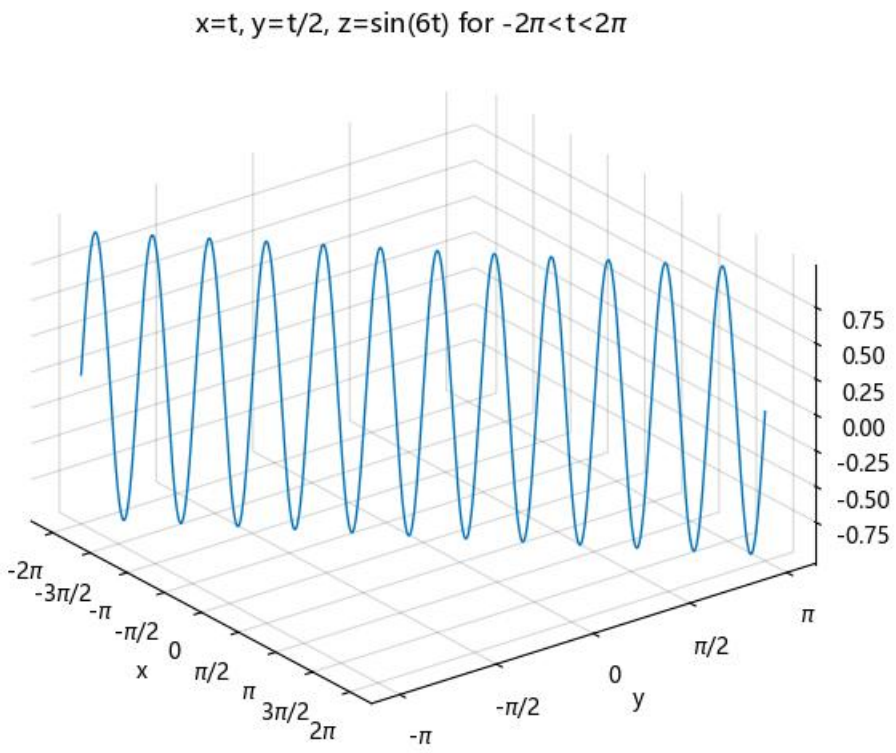
title(raw"x=t, y=t/2, z=sin(6t) for -2$\pi$<t<2$\pi$")
xlabel("x");
ylabel("y");
ax = gca()
plt_view(52.5, 30)
```



使用 gca 访问坐标区对象。使用坐标区对象的 XTick 和 XTickLabel 属性指定 x 轴刻度值和关联的标签。按照同样的方式指定 y 轴刻度值和关联的标签。

```
ax = gca();
xticks(ax, -2*pi:pi/2:2*pi);
```

```
xticklabels(ax,       [raw"-2$\pi$",raw"-3$\pi$/2",raw"-$\pi$",raw"-
$\pi$/2","0",raw"$\pi$/2",raw"$\pi$",raw"3$\pi$/2",raw"2$\pi$"]);
yticks(ax, -pi:pi/2:pi);
yticklabels(ax,                                    [raw"-$\pi$",raw"-
$\pi$/2","0",raw"$\pi$/2",raw"$\pi$"]);
```



3.3.1.2 数据分布图

直方图、饼图、文字云等

使用直方图、饼图等将数据分布可视化。例如，使用直方图将数据分组到各个 bin 中并显示每个 bin 中的元素数量。

3.3.1.2.1 直方图

函数名	简介
hist	直方图
histogram	直方图
histogram2	二元直方图

morebins	增加直方图的 bin 数量
fewerbins	减少直方图 bin 数量
histcounts	直方图 bin 计数
histcounts2	二元直方图 bin 计数
boxchart	创建箱线图

hist

直方图(不推荐;请改用 histogram)

A. 语法

```
hist(x)
hist(x,nbins)
hist(x,xbins)
hist(ax,___)
res = hist(___)
```

B. 说明

hist(x) 基于向量 x 中的元素创建直方图条形图。x 中的元素有序划分入 x 轴上介于 x 的最小值和最大值间的 10 个等间距 bin 中。hist 将 bin 显示为矩形，这样每个矩形的高度就表示 bin 中的元素数量。

如果输入是多列数组，则 hist 为每列 x 创建直方图并将它们叠加到一个绘图上。

hist(x,nbins) 将 x 有序划分入标量 nbins 所指定数量的 bin 中。

hist(x,xbins) 使用由向量 xbins 确定的间隔或类别将 x 有序划分入 bin 中。

如果 xbins 是一个包含等间距值的向量，则 hist 将使用这些值作为直方图中心。

如果 xbins 是一个包含非等间距值的向量，则 hist 将使用连续值之间的中点作为 bin 边界。

向量 xbins 的长度等于 bin 的数量。

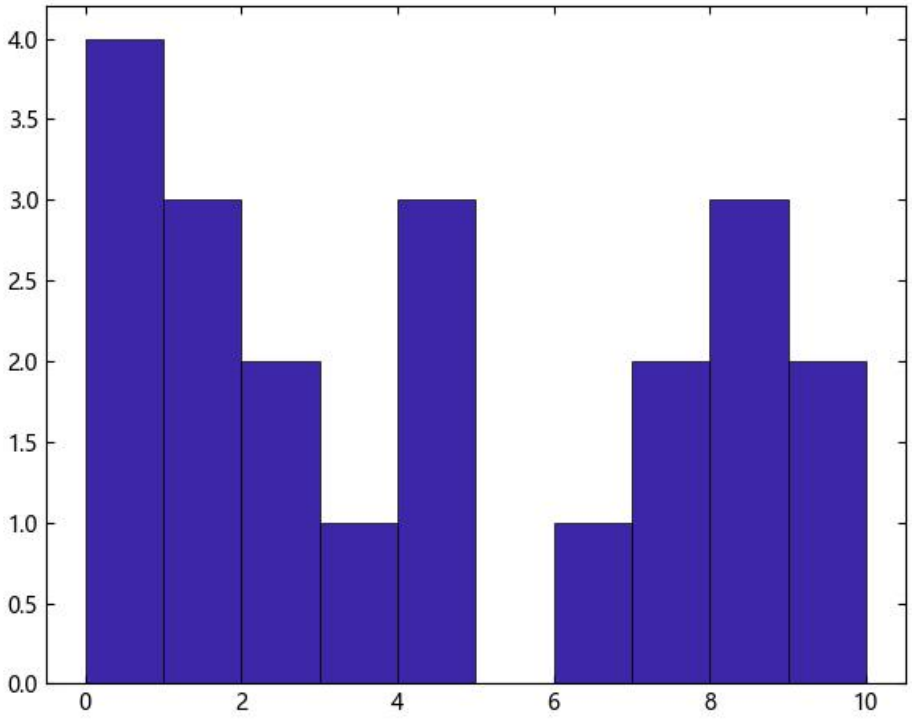
hist(ax,___) 将图形绘制到 ax 指定的坐标区中，而不是当前坐标区 (gca) 中。选项 ax 可以位于前面的语法中的任何输入参数组合之前。

res = hist(___) 返回一个行向量 counts，指示每个 bin 中的元素数目。

C. 示例

(1). 向量直方图

```
x = [0 2 9 2 5 8 7 3 1 9 4 3 5 8 10 0 1 2 9 5 10];
hist(x)
```

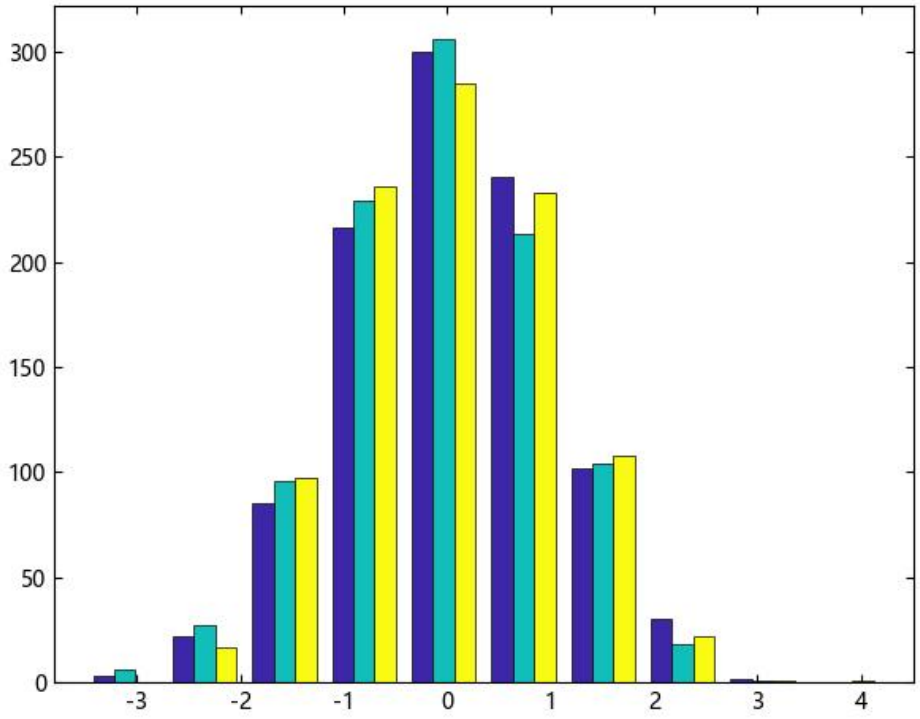


hist 将 x 中的值有序划分入介于向量中最小值和最大值间的 10 个等间距 bin 中，在本例中，最小值是 0，最大值是 10。

(2). 多列组成的直方图

生成由 1,000 个随机数组成的三列并绘制三列叠加的直方图。

```
x = randn(1000, 3);
hist(x)
```

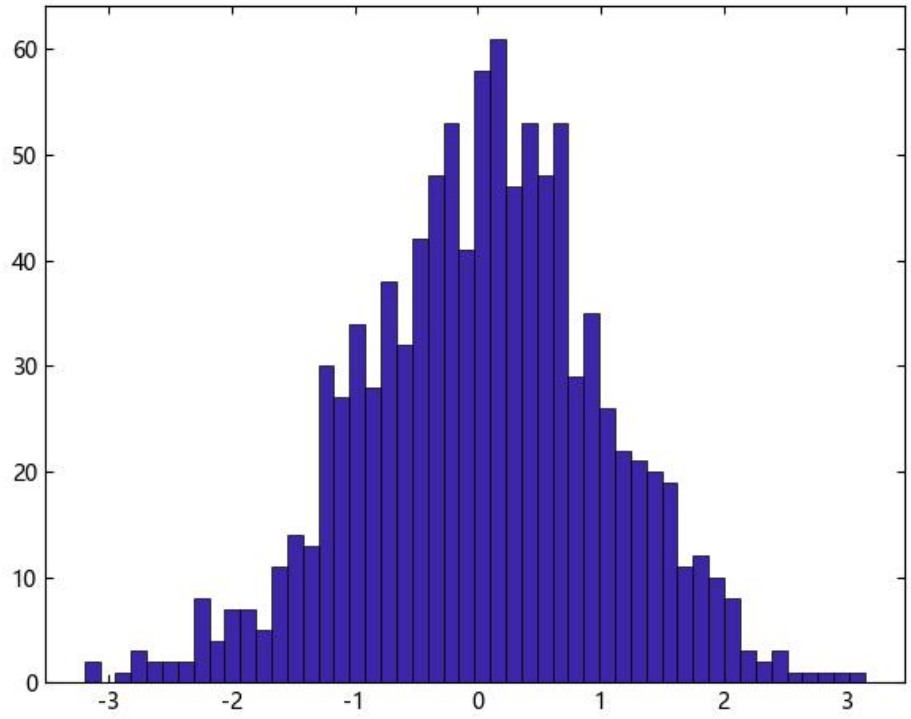


x 中的值将有序划分入介于最小值和最大值间的 10 个等间距 bin 中。hist 分别对 x 的各列排序、存储，并使用不同的颜色绘制每列。

(3). 指定直方图的 bin 数量

对分类为 50 个等距 bin 的 1,000 个随机数绘制直方图。

```
x = randn(1000, 1);
nbins = 50;
hist(x, nbins);
```



x 中的值将有序划分入介于最小值和最大值间的 10 个等间距 bin 中。hist 分别对 x 的各列排序、存储，并使用不同的颜色绘制每列。

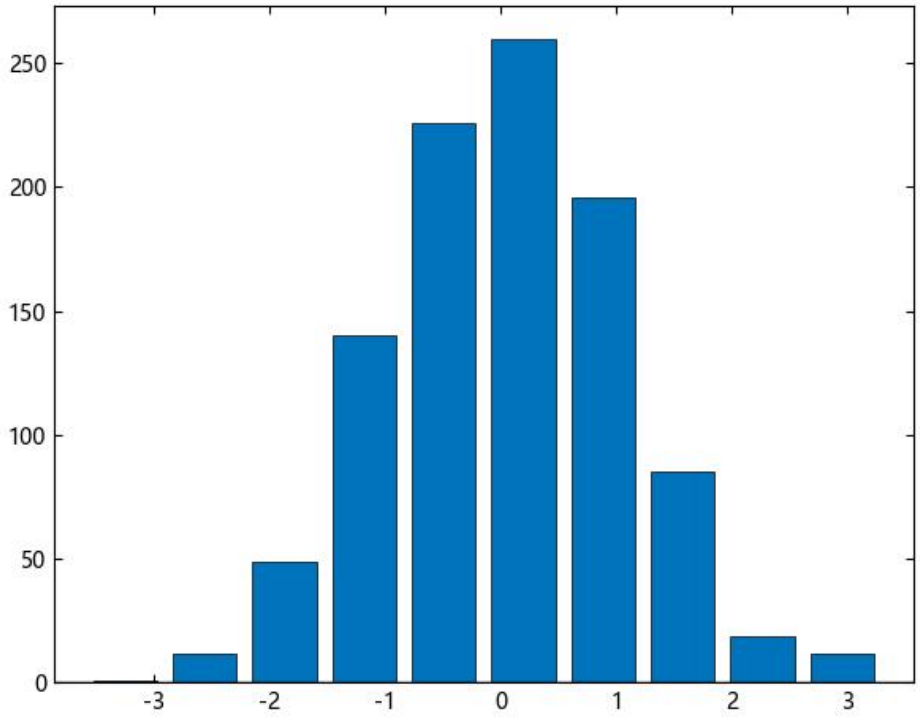
(4). 仅使用 hist 计算

生成 1,000 个随机数。计算 10 个等间距 bin 的每一个中有多少个数字。返回 bin 计数和 bin 中心。

```
x = randn(1000, 1);
res = hist(x);
```

使用 bar 绘制直方图。

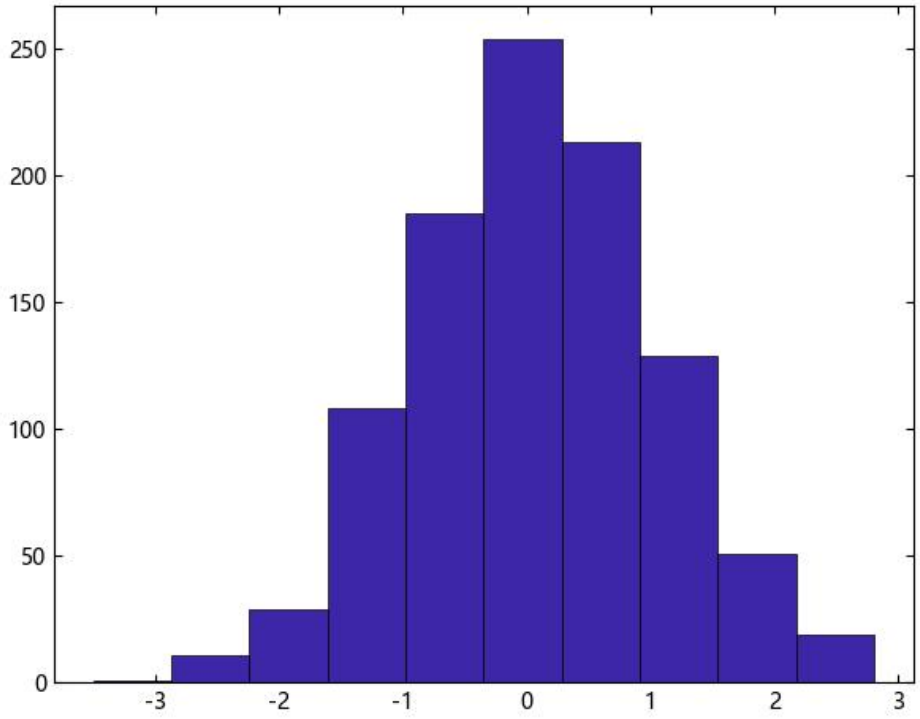
```
bar(res[2], res[1])
```



(5). 指定直方图的颜色

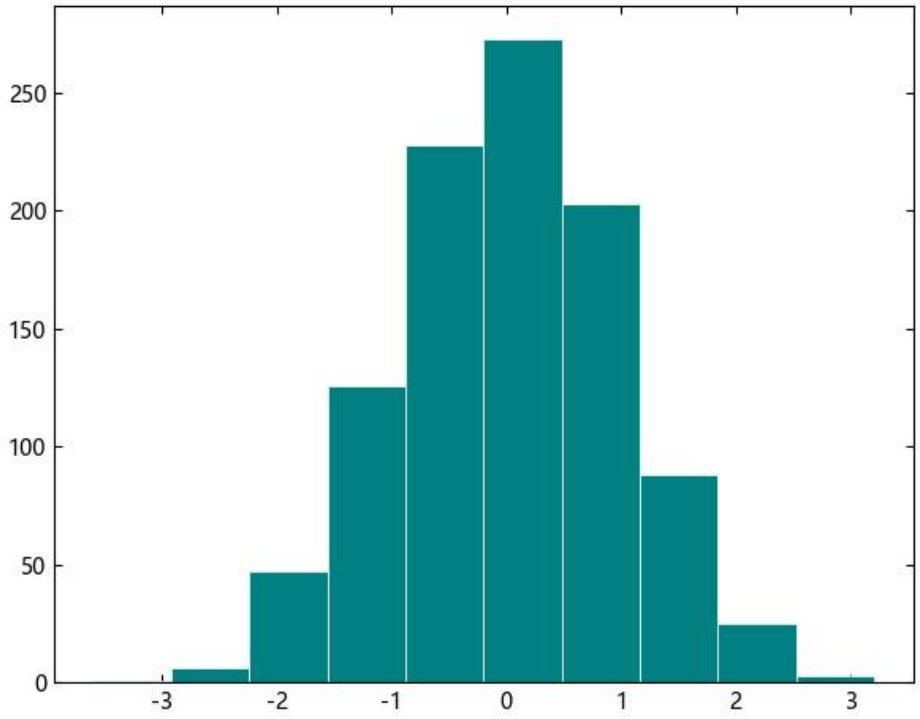
生成 1,000 个随机数并创建直方图。

```
data = randn(1000,1);
h = hist(data)
```



将已绘制条形的面颜色设置为 RGB 三元值 [0 0.5 0.5]。将边颜色设置为白色。

```
for rect in h[3]
    rect.set_facecolor([0 0.5 0.5])
    rect.set_edgecolor("w")
end
```



histogram

直方图

直方图属于数值数据的条形图类型，将数据分组为 bin。创建 Histogram 对象后，可以通过更改直方图的属性值修改它的各个方面。

A. 语法

```
histogram(X)
histogram(X,nbins)
histogram(X,edges)

histogram(___,Key=Value)
histogram(ax,___)
h = histogram(___)
```

B. 说明

histogram(X)基于 X 创建直方图。histogram 函数使用自动 bin 划分算法，然后返回均匀宽度的 bin，这些 bin 可涵盖 X 中的元素范围并显示分布的基本形状。histogram 将 bin 显示为矩形，这样每个矩形的高度就表示 bin 中的元素数

量。

histogram(X,nbins)使用标量 nbins 指定的 bin 数量。

histogram(X,edges)将 X 划分到由向量 edges 来指定 bin 边界的 bin 内。每个 bin 都包含左边界，但不包含右边界，除了同时包含两个边界的最后一个 bin 外。

histogram(___,Key=Value)使用前面的任何语法指定具有一个或多个 Key=Value 对组参数的其他选项。例如，可以指定"binwidth"和一个标量以调整 bin 的宽度，或指定"normalization"和一个有效选项（"count"、"pdf"、"cumcount"或"cdf"）以使用不同类型的归一化。

histogram(ax,___) 将图形绘制到 ax 指定的坐标区中，而不是当前坐标区 (gca) 中。选项 ax 可以位于前面的语法中的任何输入参数组合之前。

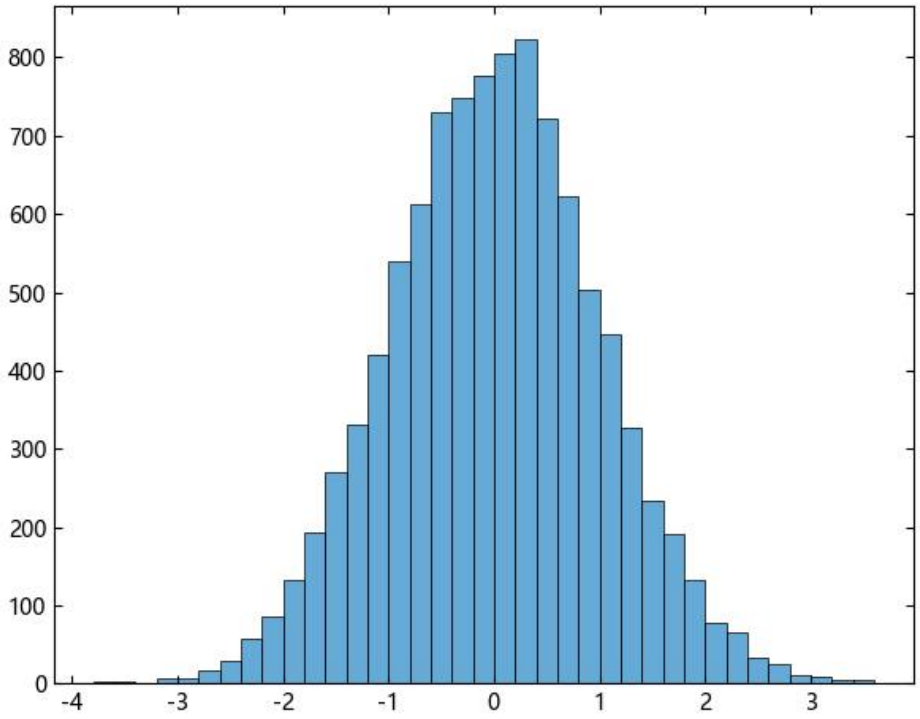
h=histogram(___)返回 Histogram 对象。使用此语法可检查并调整直方图的属性。

C. 示例

(1). 向量直方图

生成 10,000 个随机数并创建直方图。histogram 函数自动选择合适的 bin 数量，以便涵盖 x 中的值范围并显示基本分布的形状。

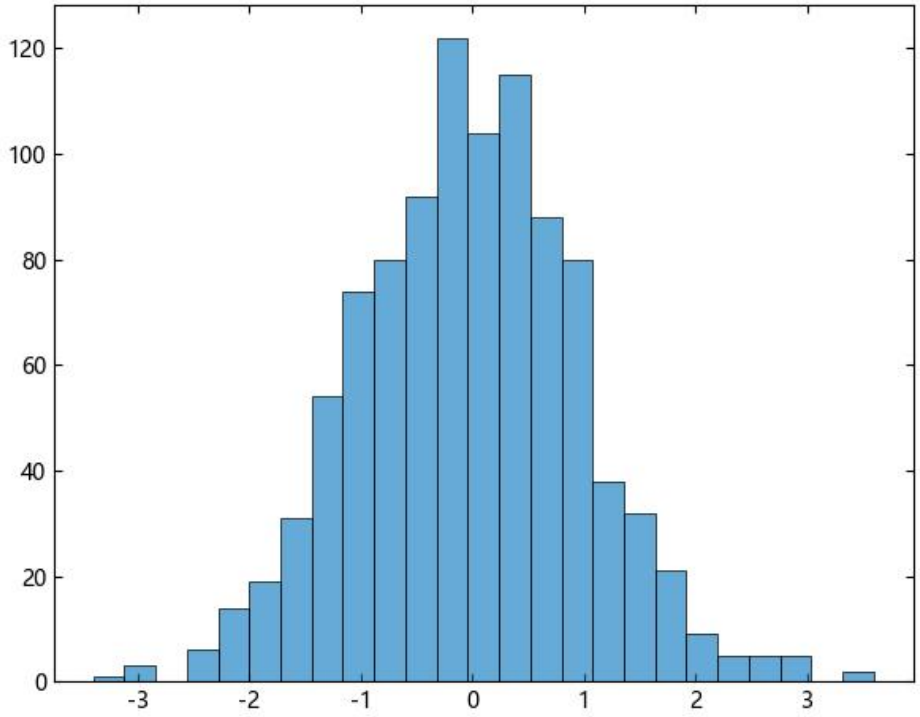
```
using TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/histogram/0-rand_data.jl"
include(source_path)
h = histogram(x_10000)
```



(2). 指定直方图的 bin 数量

对分类为 25 个等距 bin 的 1,000 个随机数绘制直方图。

```
using TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/histogram/0-rand_data.jl"
include(source_path)
nbins = 25
h = histogram(x_1000, nbins)
```

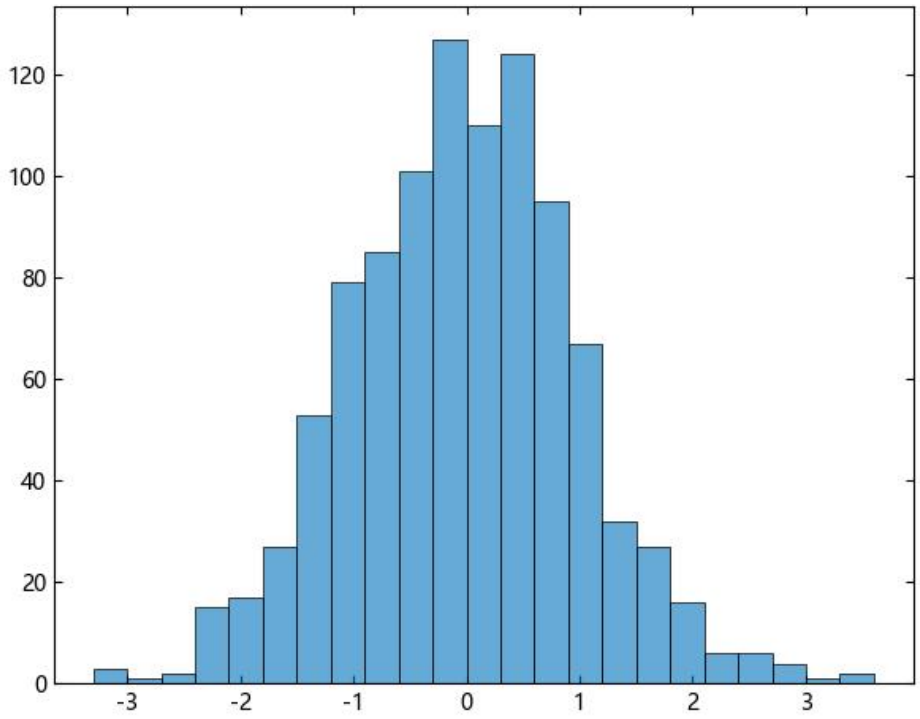


更改直方图的 bin 数量生成 1,000 个随机数并创建直方图。

```
function morebins(nbins)
    return Int(ceil(nbins * 1.1))
end

function fewerbins(nbins)
    return Int(floor(nbins * 0.9))
end

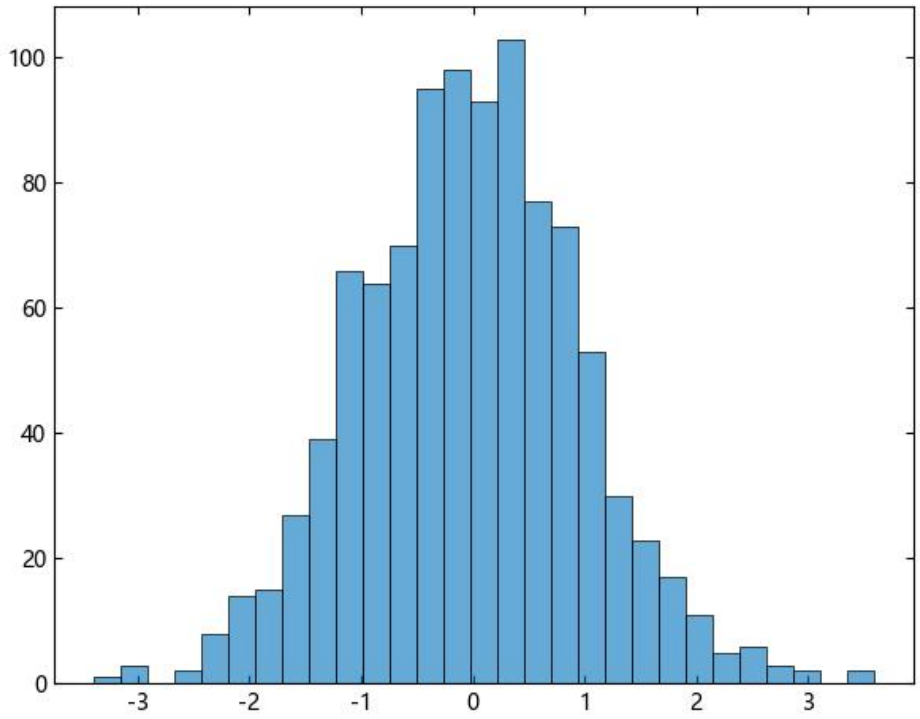
using TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/histogram/0-rand_data.jl"
include(source_path)
h = histogram(x_1000)
```



使用 morebins 函数粗略调整 bin 数量。

```
numbins = length(h[1])
Nbins = morebins(numbins)
Nbins = morebins(Nbins)

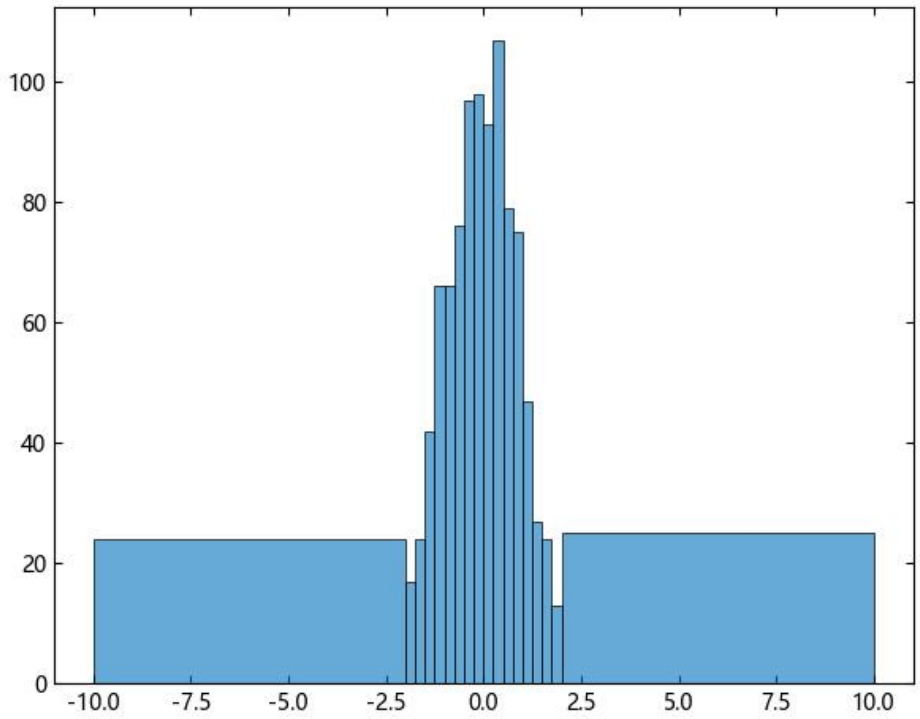
cla()
h = histogram(x_1000, Nbins)
println(Nbins)
```



(3). 指定直方图的 bin 边界

生成 1,000 个随机数并创建直方图。将 bin 边界指定为向量，使宽 bin 在直方图的两边，以捕获不满足 |x|<2 的离群值。第一个向量元素是第一个 bin 的左边界，而最后一个向量元素是最后一个 bin 的右边界。

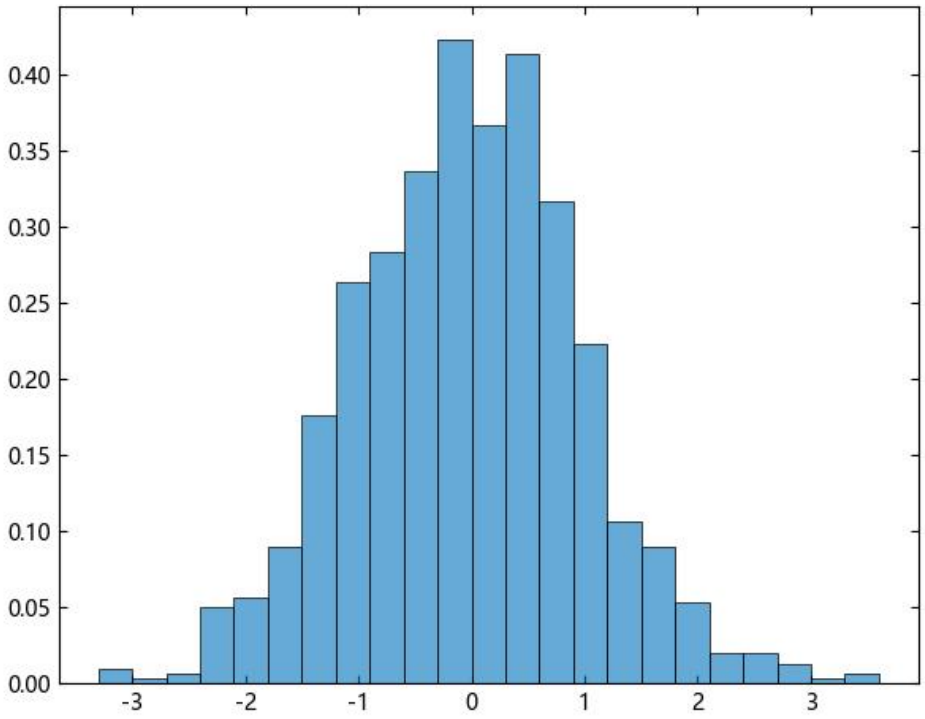
```
using TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/histogram/0-rand_data.jl"
include(source_path)
edges = [-10 -2:0.25:2... 10]
h = histogram(x_1000, edges)
```



(4). 具有指定归一化的直方图

生成 1,000 个随机数并使用 "pdf" 归一化创建直方图。

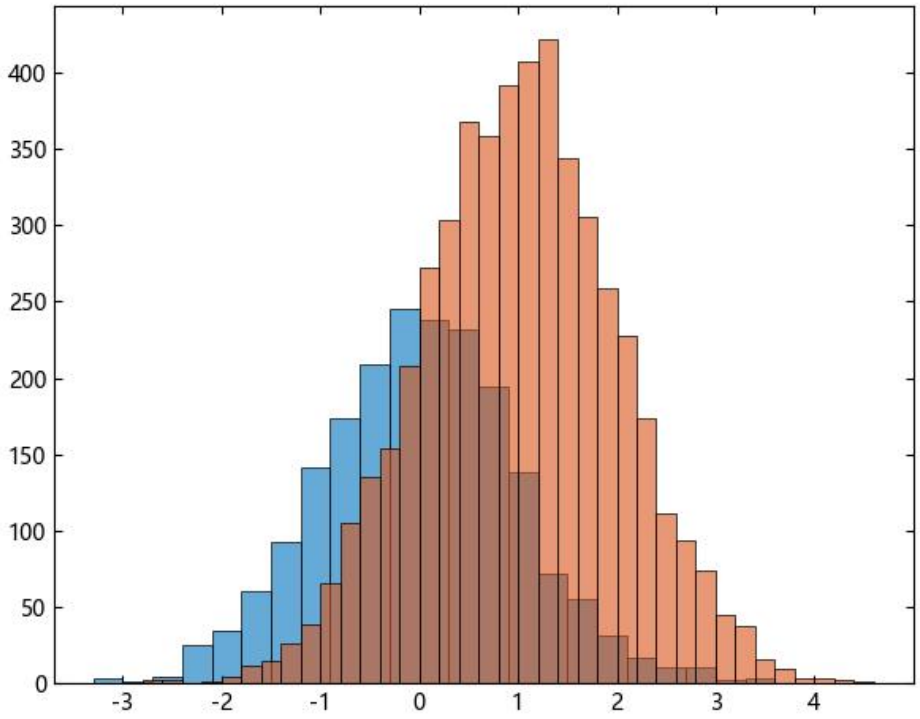
```
using TyPlot: TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/histogram/0-rand_data.jl"
include(source_path)
h = histogram(x_1000; normalization="pdf")
```



(5). 绘制多个直方图

生成两个随机数向量并在同一图窗中针对每个向量绘制对应的一个直方图。

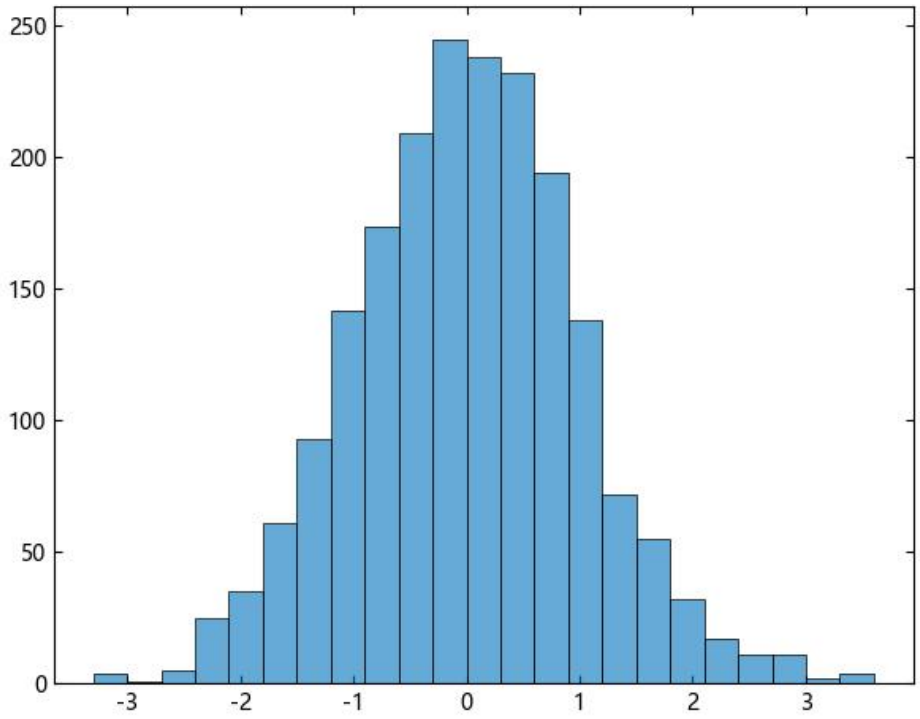
```
using TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/histogram/0-rand_data.jl"
include(source_path)
x = x_2000
y = 1 .+ x_5000_2
h1 = histogram(x; facealpha=0.6)
hold("on")
h2 = histogram(y; facealpha=0.6)
hold("off")
```



(6). 调整直方图属性

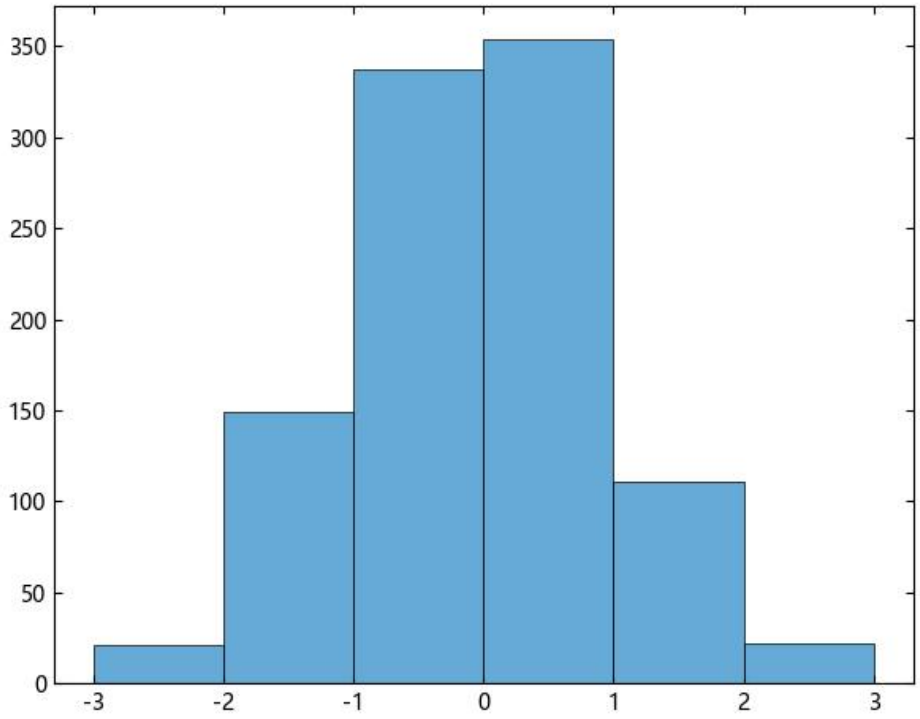
生成 1,000 个随机数并创建直方图。返回直方图对象以调整该直方图的属性，无需重新创建整个绘图。

```
using TyPlot: TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/histogram/0-rand_data.jl"
include(source_path)
```



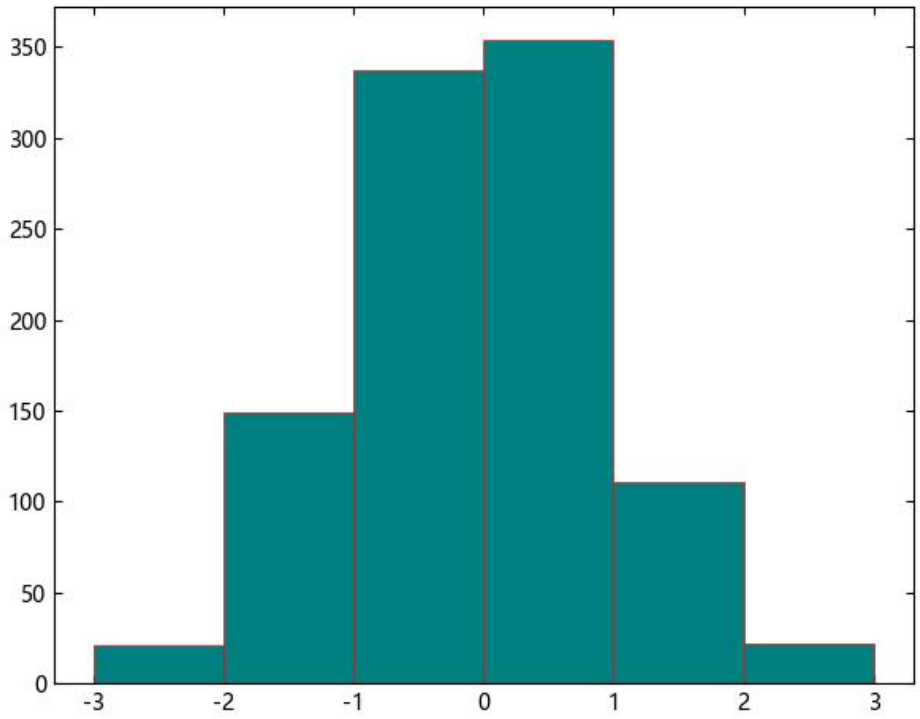
通过向量指定 bin 边界。向量中的第一个值是第一个 bin 的左边界。最后一个值是最后一个 bin 的右边界。

```
h = histogram(x_1000, [-3:3...])
```



更改直方图条形的颜色。

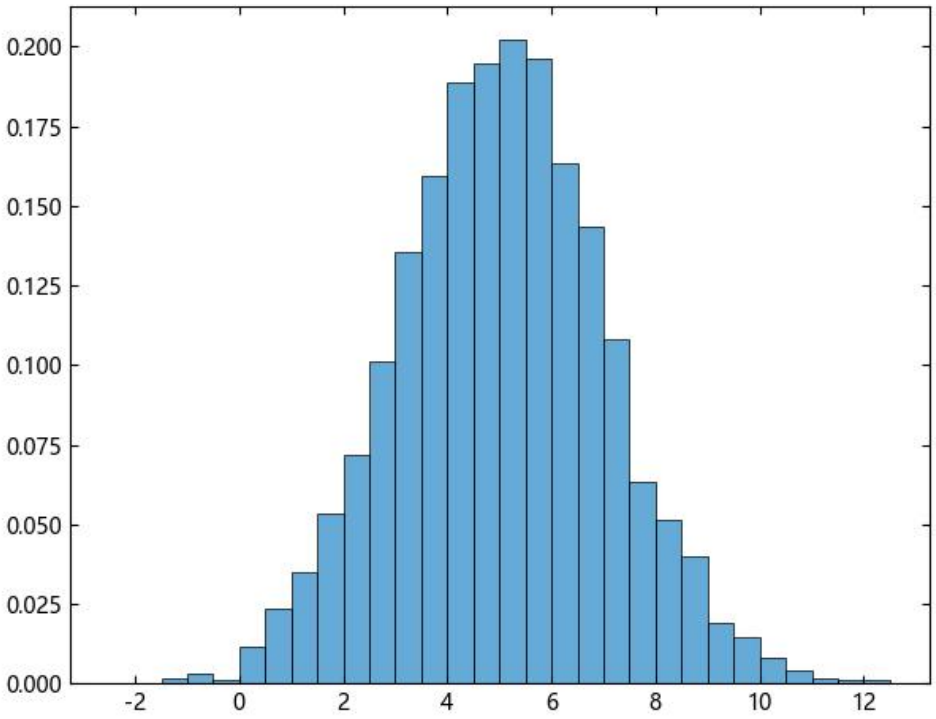
```
for rect in h[3]
    rect.set_facecolor((0, 0.5, 0.5))
    rect.set_edgecolor("r")
end
```



(7). 确定基本概率分布

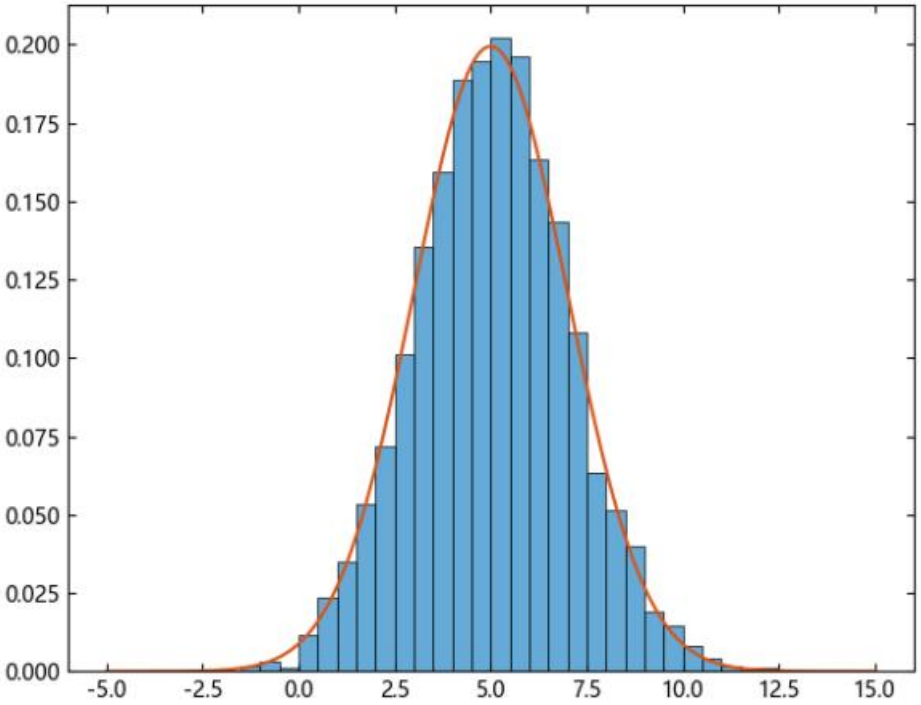
生成 5,000 个均值为 5、标准差为 2 的正态分布随机数。在 normalization 设为 "pdf" 的情况下绘制直方图可生成概率密度函数的估计值。

```
using TyPlot: TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/histogram/0-rand_data.jl"
include(source_path)
x = 2 .* x_5000 .+ 5
h = histogram(x; normalization="pdf")
```



对于均值为 5、标准差为 2 的正态分布，叠加一个概率密度函数图。

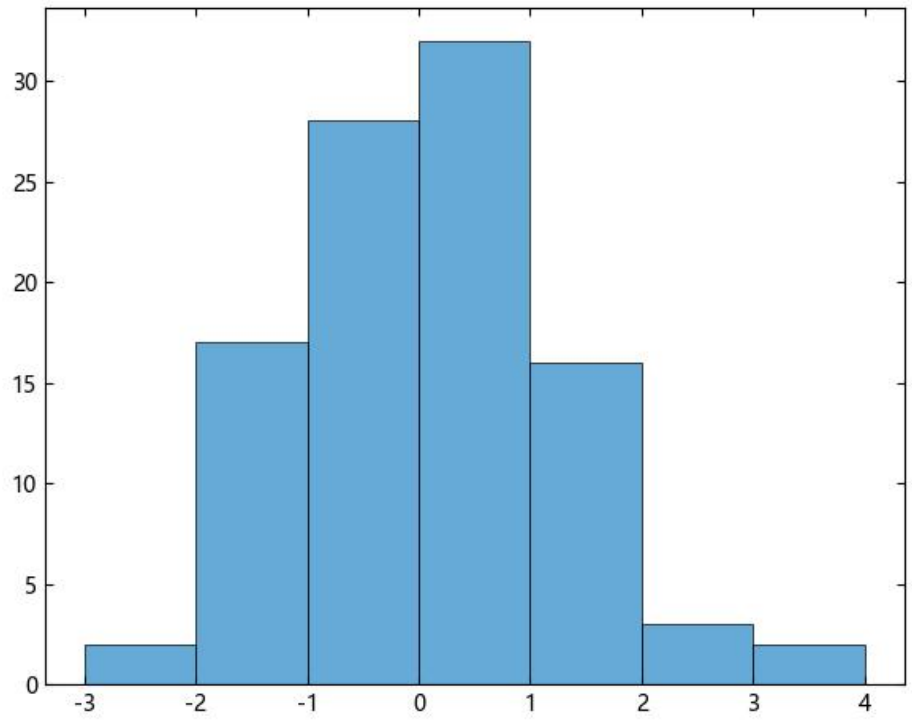
```
hold("on")
y = -5:0.1:15;
mu = 5;
sigma = 2;
f = exp.(-(y .- mu) .^ 2 ./ (2 * sigma^2)) ./ (sigma * sqrt(2 *
pi))
plot(y, f; linewidth=1.5)
hold("off")
```



(8). 保存并加载直方图对象

使用 saveas 函数保存直方图。

```
y = histogram(randn(10));
saveas(gcf(), "histogram.png")
```



histogram2

二元直方图

二元直方图是一种数值数据条形图，它将数据分组到二维 bin 中。创建 Histogram2 对象后，可以通过更改直方图的属性值修改它的各个方面。这对快速修改 bin 属性或更改显示特别有用。

A. 语法

```
histogram2(X,Y)
histogram2(X,Y,nbins)
histogram2(X,Y,Xedges,Yedges)

histogram2(___,Key=Value)
histogram2(ax,___)
h = histogram2(___)
```

B. 说明

histogram2(X,Y) 创建 X 和 Y 的二元直方图。histogram2 函数使用自动 bin 划分算法，然后返回均匀面积的 bin，这些 bin 可涵盖 X 和 Y 中的元素范围并显示分布的基本形状。histogram2 将 bin 显示为三维矩形条形，这样每个条形的高度就表示 bin 中的元素数量。

histogram2(X,Y,nbins)指定要在直方图的每个维度中使用的 bin 数量。示例

histogram2(X,Y,Xedges,Yedges)使用向量 Xedges 和 Yedges 指定每个维度中 bin 的边界。

histogram2(___,Key=Value)使用前面的任何语法指定具有一个或多个 Name,Value 对组参数的其他选项。例如，可以指定 "binwidth" 和一个二元素向量以调整每个维度中 bin 的宽度，或指定 "normalization" 和一个有效选项（"count"、"probability" 或 "countdensity"）以使用不同类型的归一化。

histogram2(ax,___) 将图形绘制到 ax 指定的坐标区中，而不是当前坐标区 (gca) 中。选项 ax 可以位于前面的语法中的任何输入参数组合之前。

h=histogram2(___)返回 Histogram 对象。使用此语法可检查并调整二元直方图的属性。

C. 示例

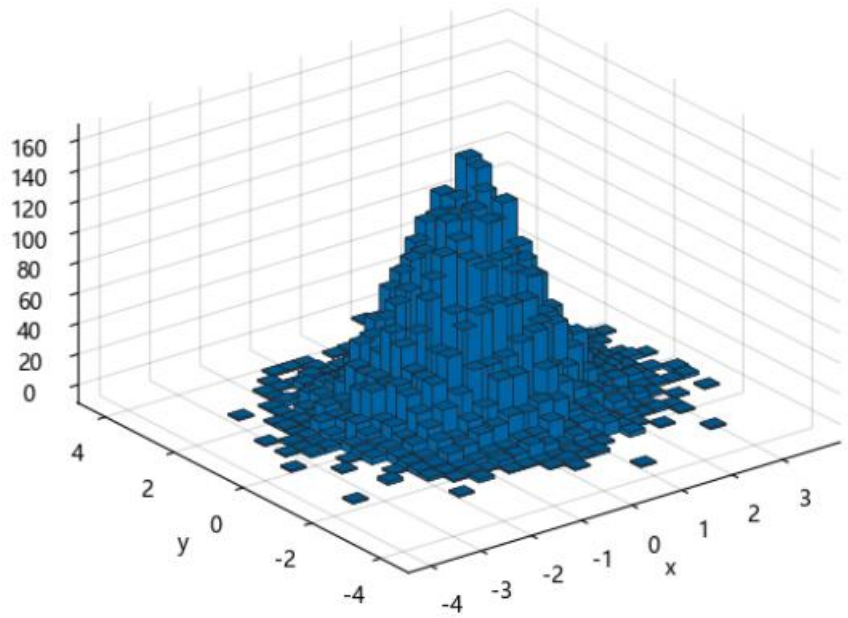
(1). 向量的直方图

生成 10000 个随机数对组并创建一个二元直方图。histogram2 函数自动选择合适的 bin 数量，以便涵盖 x 和 y 中的值范围并显示基本分布的形状。

```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/histogram2/data_10000.jl"
include(source_path)

h = histogram2(x, y)

xlabel("x")
ylabel("y")
```



指定 histogram2 函数的输出参数时，它返回一个二元直方图对象。可以使用该对象检查直方图的属性，例如 bin 数量或宽度。

计算每个维度的直方图 bin 数量。

```
nXnY = h.NumBins
```

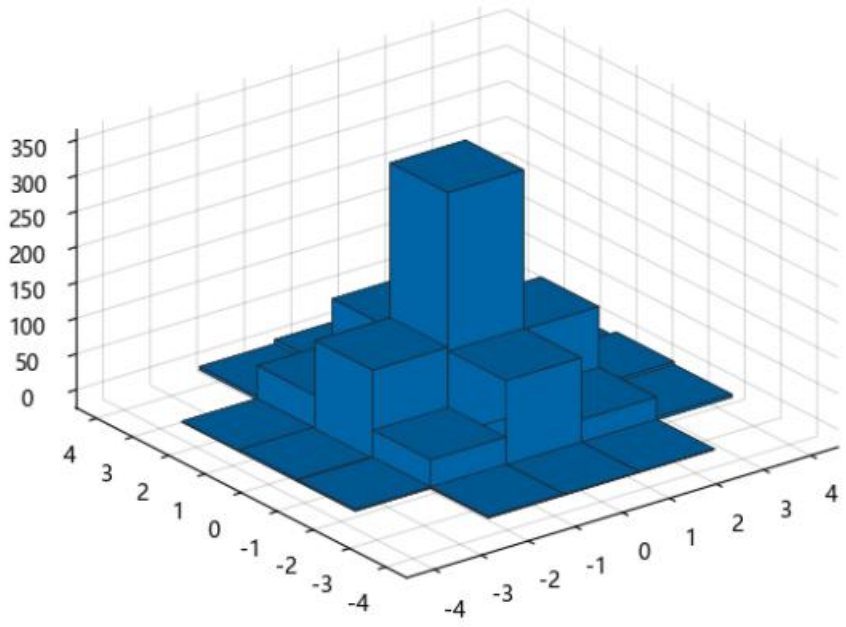
```
nXnY =
2-element Vector{Int64}:
 25
 28
```

(2). 指定直方图的 bin 数量

为划分入 25 个等距 bin 的 1,000 个随机数对组绘制一个二元直方图，其中每个维度使用 5 个 bin。

```
file_path = (pkgdir(TyPlot)) * "/examples/histogram2/data_1000.jl"
include(file_path)

nbins = 5;
h = histogram2(x, y, nbins)
```



求生成的 bin 计数。

```
counts = h.Values
```

```
counts =
5×5 Matrix{Float64}:
 0.0    2.0    3.0    1.0   0.0
 2.0   40.0  124.0   47.0   4.0
 1.0  119.0  341.0  109.0  10.0
 1.0   32.0  117.0   33.0   1.0
 0.0    4.0    8.0    1.0   0.0
```

(3). 调整直方图 bin 的数量

生成 1,000 个随机数对组并创建一个二元直方图。

```
function morebins(nbins)
    return Int(ceil(nbins * 1.1))
end

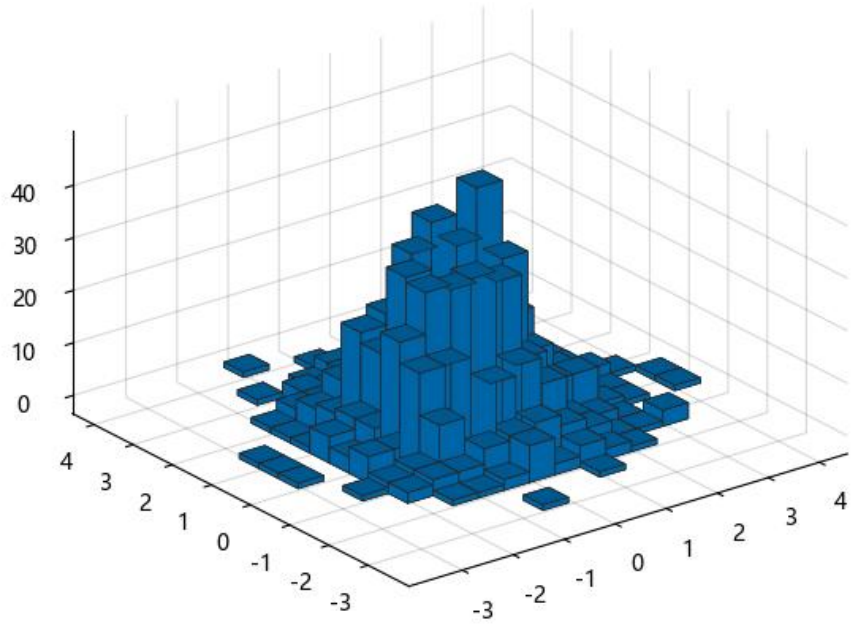
function fewerbins(nbins)
    return Int(floor(nbins * 0.9))
```

```
end

using TyPlot

file_path = (pkgdir(TyPlot)) * "/examples/histogram2/data_1000.jl"
include(file_path)

h = histogram2(x, y)
```



使用 morebins 函数精略调整 x 维度中的 bin 数量。

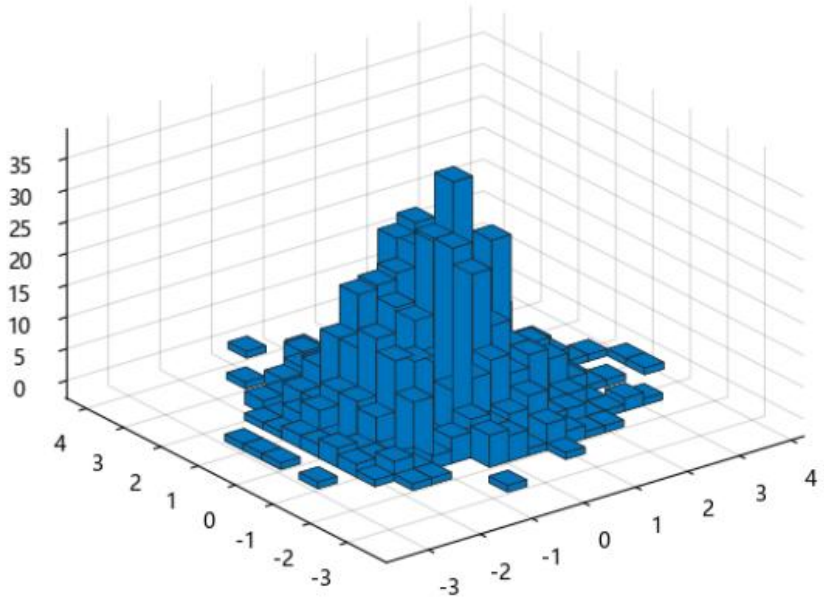
```
nbins_x = morebins(h.NumBins[1]);
nbins_x = morebins(nbins_x)
nbins = [nbins_x, h.NumBins[2]]
println(nbins)

yedges = h.YBinEdges

cla()
h = histogram2(x, y, nbins, ybinedges = yedges)
```

```
nbins =
2-element Vector{Int64}:
 19
```

```
15
```

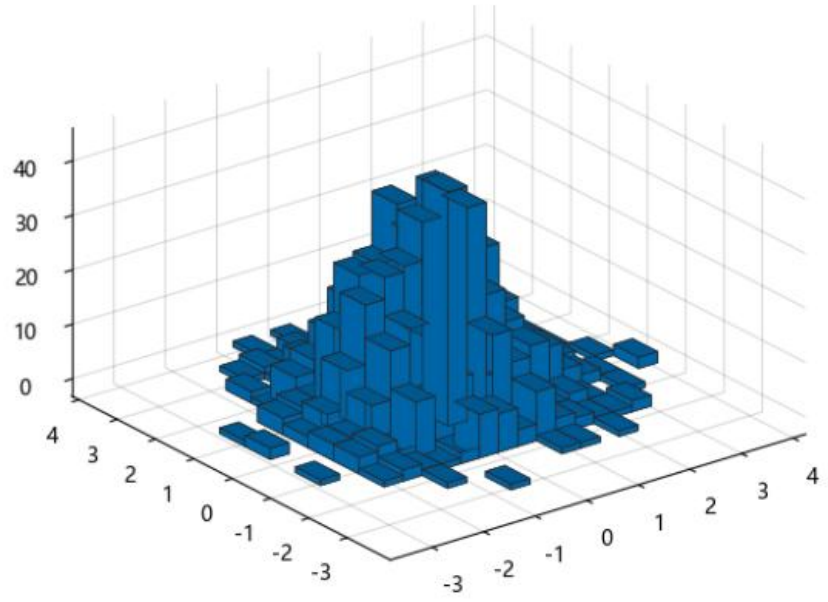


使用 fewerbins 函数调整 y 维度中的 bin 数量。

```
nbins_y = fewerbins(h.NumBins[2]);
nbins_y = fewerbins(nbins_y)
nbins = [h.NumBins[1], nbins_y]
println(nbins) #[19 15]

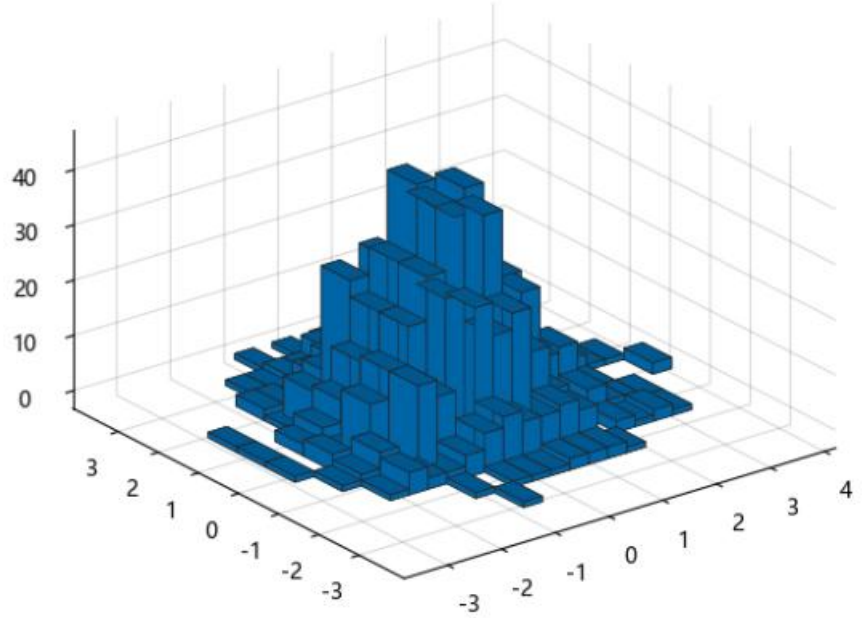
cla()
h = histogram2(x, y, nbins)
```

```
nbins =
2-element Vector{Int64}:
 19
 11
```



通过显式设置 bin 数按精细粒度级别调整 bin 数量。

```
cla()
h = histogram2(x, y, [20, 10])
```



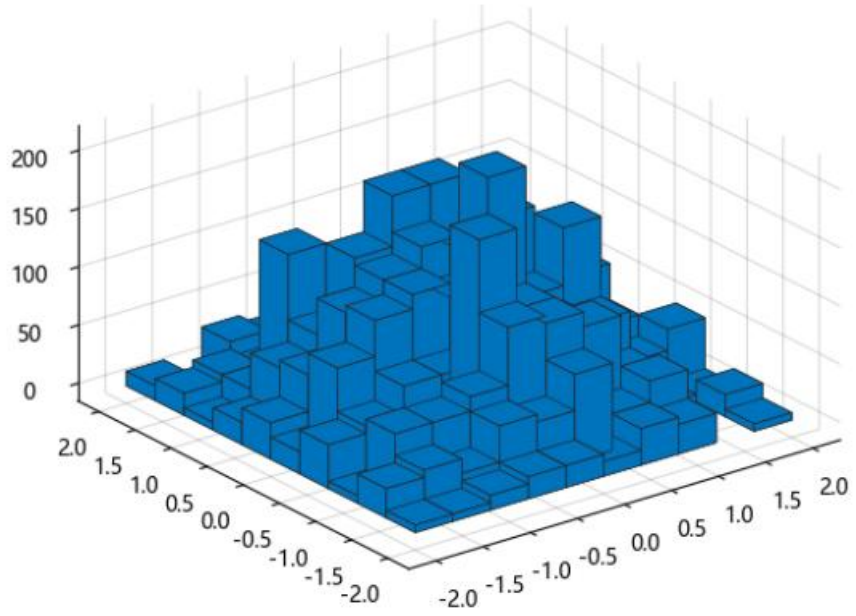
(4). 指定直方图的 bin 边界

生成 1,000 个随机数对组并创建一个二元直方图。使用两个向量指定 bin 边界，使具有无限宽的 bin 在直方图的边缘，以捕获不满足 x <2 的所

有离群值。

```
file_path = (pkgdir(TyPlot)) * "/examples/histogram2/data_1000.jl"
include(file_path)

Xedges = [-2:0.4:2...];
Yedges = [-2:0.4:2...];
h = histogram2(x, y, Xedges, Yedges, normalization = "countdensity")
```

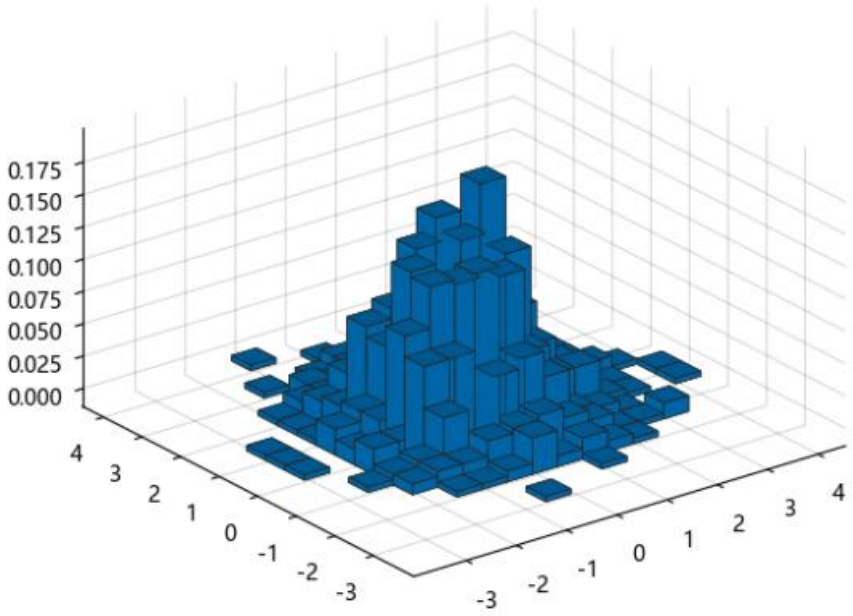


(5). 归一化的直方图

生成 1000 个随机数对组并使用 "probability" 归一化创建一个二元直方图。

```
file_path = (pkgdir(TyPlot)) * "/examples/histogram2/data_1000.jl"
include(file_path)

h = histogram2(x, y, normalization = "probability")
```



计算条形高度的总和。通过此归一化，每个条形的高度等于选取位于该 bin 间隔内的观测值的概率，并且所有条形的高度总和为 1。

```
S = sum(h.Values)
```

```
S = 1.0
```

morebins

增加直方图的 bin 数量

A. 语法

```
N = morebins(h)
```

B. 说明

N = morebins(h) 将直方图 h 中 bin 的数量增加 10%（向上四舍五入为最近的整数），并返回新的 bin 数量。示例

对于二元直方图，这会增加 x 和 y 方向的 bin 数量。

C. 示例

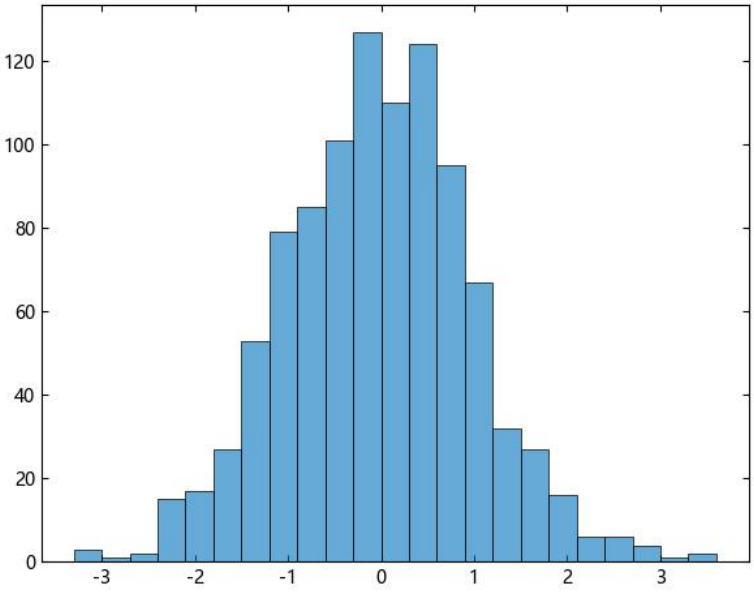
(1). 增加直方图的 bin 数量

绘制由 1,000 个随机数构成的直方图并返回直方图对象的句柄。

```
using TyPlot: TyPlot
```

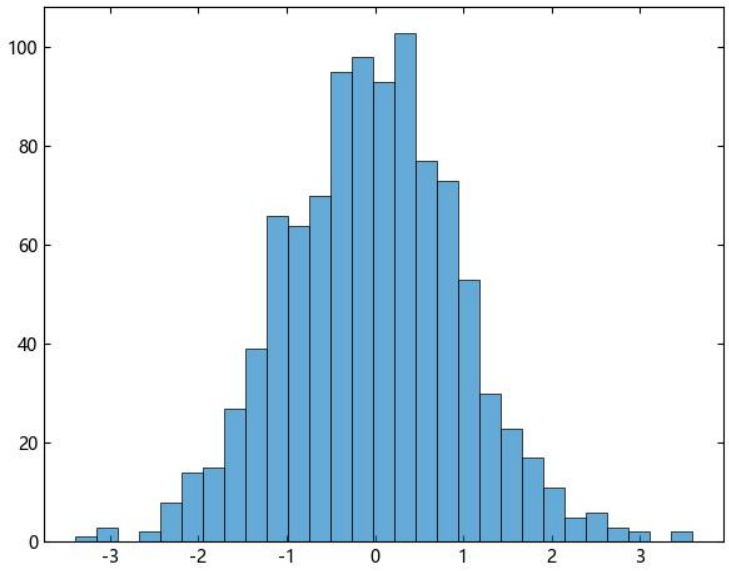
```
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/histogram/0-rand_data.jl"

include(source_path)
h = histogram(x_1000)
```



使用 morebins 增加直方图中的 bin 数量。

```
function morebins(nbins)
    return Int(ceil(nbins * 1.1))
end
numbins = length(h[1])
Nbins = morebins(numbins)
Nbins = morebins(Nbins)
cla()
h = histogram(x_1000, Nbins)
println(Nbins)
```



29

fewerbins

减少直方图 bin 数量

A. 语法

```
N = fewerbins(h)
```

B. 说明

N = fewerbins(h) 将直方图 h 中的 bin 数量减少 10%（向下四舍五入为最近的整数），并返回新的 bin 数量。示例

对于二元直方图，这会减少 x 和 y 方向的 bin 数量。

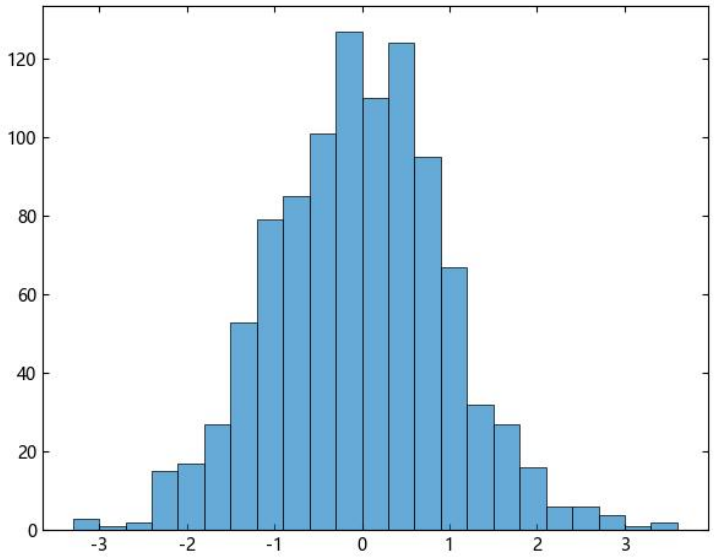
C. 示例

(1). 减少直方图的 bin 数量

绘制由 1,000 个随机数构成的直方图并返回直方图对象的句柄。

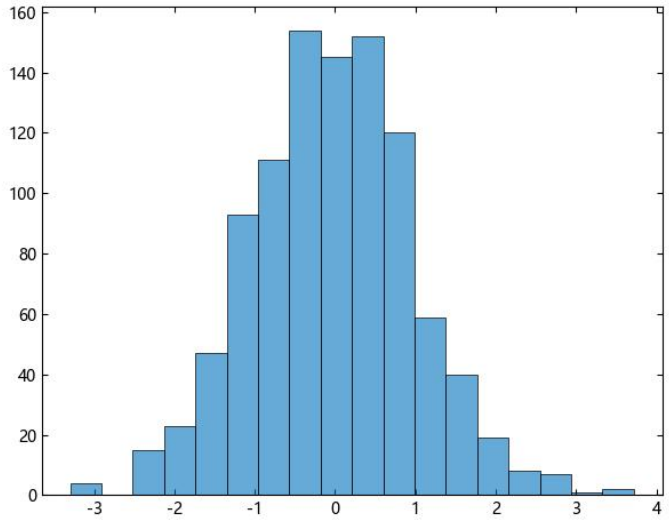
```
using TyPlot: TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/histogram/0-rand_data.jl"

include(source_path)
h = histogram(x_1000)
```



使用 fewerbins 减少直方图中的 bin 数量。

```
function fewerbins(nbins)
    return Int(floor(nbins * 0.9))
end
numbins = length(h[1])
Nbins = fewerbins(numbins)
Nbins = fewerbins(Nbins)
cla()
h = histogram(x_1000, Nbins)
println(Nbins)
```




```
18
```

histcounts

直方图 bin 计数

A. 语法

```
edges = histcounts(X)
edges = histcounts(X,nbins)
edges = histcounts(X,edges)

edges = histcounts(___)

edges = histcounts(___,Key=Value)
```

B. 说明

edges=histcounts(X)将 X 的值划分为多个 bin，并返回每个 bin 中的计数以及 bin 边界。histcounts 函数使用自动 bin 划分算法，返回均匀宽度的 bin，这些 bin 可涵盖 X 中的元素范围并显示基本分布的形状。

edges=histcounts(X,nbins)使用标量 nbins 指定的 bin 数量。

edges=histcounts(X,edges) 将 X 划分到由向量 edges 来指定 bin 边界的 bin 内。如果 edges[k] ≤ X[i] < edges[k+1]，值 X[i] 位于第 k 个 bin 中。最后一个 bin 也包含 bin 的右边界，这样如果 edges[end-1] ≤ X[i] ≤ edges[end]，它包含 X[i]。

edges = histcounts(___) ，返回对应于 N 中每个计数的类别。

edges=histcounts(___,Key=Value)使用一个或多个 Key=Value 对组指定的其他选项，这些选项使用上述语法中的任意输入或输出参数组合。例如，您可以指定"binwidth"和一个标量来调整数值数据的 bin 的宽度。

C. 示例

(2). bin 计数和 bin 边界

将 100 个随机值分布到多个 bin 内。histcounts 自动选择合适的 bin 宽度以显示数据的基本分布。

```
X = randn(100,1)  # 返回由正态分布的随机数组成的 m×n 矩阵
edges = histcounts(X)
```

```
8-element Vector{Float64}:
 -3.0
```

```
 -2.0
 -1.0
  0.0
  1.0
  2.0
  3.0
  4.0
```

(3). 指定 bin 数

将 10 个随机数分布到 6 个等间距 bin 内。

```
X = [2 3 5 7 11 13 17 19 23 29];
edges = histcounts(X,6)

using Formatting
for n in edges
println(fmt(".1f", n))
end
```

```
0.0
4.9
9.8
14.7
19.6
24.5
29.4
```

(4). 归一化的 bin 计数

将小于 100 的所有质数分布到多个 bin 内。将 "normalization" 指定为 "probability" 以对 bin 计数进行归一化，从而 sum(N) 为 1。即，每个 bin 计数代表观测值属于该 bin 的可能性。

```
X=[2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83
89 97]
edges = histcounts(X, normalization="probability")
```

```
5-element Vector{Float64}:
   0.0
  30.0
  60.0
  90.0
 120.0
```

(5). 确定 bin 放置

将介于 -5 和 5 之间的 100 个随机整数分布到多个 bin 内，并将 "binmethod" 指定为 "integers" 以使用以整数为中心的单位宽度 bin。指定 histcounts 的第三个输出以返回代表数据 bin 索引的向量。

```
using TyPlot
import TyPythonCall: @pyimport
@pyimport numpy as np
X = []
for i in 1:100
push!(X, np.random.randint(-5,6))
end
edges = histcounts(X,binmethod="integers")
```

```
12-element Vector{Float64}:
 -5.5
 -4.5
 -3.5
 -2.5
 -1.5
 -0.5
  0.5
  1.5
  2.5
  3.5
  4.5
  5.5
```

Histcounts2

二元直方图 bin 计数

A. 语法

```
n,Xedges,Yedges, = histcounts2(X,Y)
n,Xedges,Yedges, = histcounts2(X,Y,nbins)
n,Xedges,Yedges, = histcounts2(X,Y,Xedges,Yedges)
n,Xedges,Yedges, = histcounts2(___,Key=Value)
n,Xedges,Yedges,binx,biny = histcounts2(___)
```

B. 说明

n,Xedges,Yedges, = histcounts2(X,Y) 将 X 和 Y 中的值划分为多个二维 bin，并返回 bin 计数以及每个维度的 bin 边界。histcounts2 函数使用自动分

bin 算法，然后返回均匀 bin，这些 bin 可涵盖 X 和 Y 中值的范围并显示分布的基本形状。示例

n,Xedges,Yedges, = histcounts2(X,Y,nbins) 指定要在每个维度中使用的 bin 数量。示例

n,Xedges,Yedges, = histcounts2(X,Y,Xedges,Yedges) 按照 Xedges 和 Yedges 指定的 bin 边界，将 X 和 Y 划分为多个 bin。示例

n(i,j) 对符合 Xedges(i) ≤ X(k) < Xedges(i+1) 且 Yedges(j) ≤ Y(k) < Yedges(j+1) 条件的值 [X(k),Y(k)] 进行计数。每个维度中的最后 bin 还包含最后一个（外部）边界。例如，如果 Xedges(end-1) ≤ X(k) ≤ Xedges(end) 且 Yedges(i) ≤ Y(k) < Yedges(i+1)，则 [X(k),Y(k)] 属于最后一行中的第 i 个 bin。

n,Xedges,Yedges, = histcounts2(___,Key=Value) 还可使用一个或多个 Key=Value 对组参数（采用上述语法中有任意输入参数）指定的其他选项。例如，可以指定 "binwidth" 和一个正整数来调整每个维度中 bin 的宽度。示例

n,Xedges,Yedges,binx,biny= histcounts2(___) 还使用任何上述语法返回索引数组 binx 和 biny。binx 和 biny 数组的大小与 X 和 Y 相同，其元素是 X 和 Y 中对应元素的 bin 索引。第 (i,j) 个 bin 中的元素数量等于 nnz(binx==i & biny==j)；若 normalization 为 "count"，则它与 n(i,j) 相同。示例。

C. 示例

(1). bin 计数和 bin 边界

将 100 个随机数对组分布到多个 bin 内。histcounts2 自动选择合适的 bin 宽度以显示数据的基本分布。

```
using TyPlot
x = randn(100, 1);
y = randn(100, 1);
n, Xedges, Yedges, = histcounts2(x, y)
```

```
n =
6×6 Matrix{Float64}:
 0.0  0.0   2.0   2.0  0.0  0.0
 0.0  1.0   5.0   8.0  4.0  0.0
 1.0  4.0   9.0  10.0  3.0  0.0
 0.0  3.0  14.0  10.0  2.0  0.0
 1.0  2.0   3.0   8.0  5.0  1.0
 0.0  0.0   0.0   2.0  0.0  0.0

Xedges =
```

```
7-element Vector{Float64}:
 -3.0
 -2.0
 -1.0
  0.0
  1.0
  2.0
  3.0

Yedges =
7-element Vector{Float64}:
 -3.0
 -2.0
 -1.0
  0.0
  1.0
  2.0
  3.0
```

(2). 指定每个维度中的 bin 数量

将 10 个数字对组分布到 12 个 bin 中。在 x 维度指定 3 个 bin，在 y 维度指定 4 个 bin。

```
x = [1 1 2 3 2 2 1 1 2 3];
y = [5 6 3 8 9 1 2 7 5 1];
nbins = [3 4];
n, Xedges, Yedges, = histcounts2(x, y, nbins)
```

```
n =
3×4 Matrix{Float64}:
 1.0  0.0  2.0  1.0
 1.0  1.0  1.0  1.0
 1.0  0.0  0.0  1.0

Xedges =
4-element Vector{Float64}:
 0.6000000000000001
 1.4000000000000001
 2.2
 3.0000000000000004

Yedges =
5-element Vector{Float64}:
 0.0
```

```
 2.3000000000000003
 4.6000000000000005
 6.9
 9.200000000000001.0
4.9
9.8
14.7
19.6
24.5
29.4
```

(3). 指定 bin 边界

将 1,000 个随机数对组分布到多个 bin 内。使用两个向量（分别针对 x 和 y 维度）定义 bin 边界。每个向量中的第一个元素指定第一个 bin 的第一个边界，最后一个元素指定最后一个 bin 的最后一个边界。

```
x = randn(1000, 1);
y = randn(1000, 1);
Xedges = -5:5;
Yedges = [-5 -4 -2 -1 -0.5 0 0.5 1 2 4 5];
n, = histcounts2(x, y, Xedges, Yedges)
```

```
n =
10×10 Matrix{Float64}:
 0.0  0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0  0.0
 0.0  0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0  0.0
 0.0  0.0   5.0   3.0   6.0   4.0   3.0   1.0   0.0  0.0
 0.0  2.0  13.0  20.0  24.0  27.0  28.0  13.0   1.0  0.0
 0.0  8.0  47.0  53.0  51.0  64.0  60.0  47.0  11.0  0.0
 0.0  4.0  43.0  54.0  61.0  70.0  57.0  45.0   8.0  0.0
 0.0  3.0  17.0  27.0  24.0  27.0  22.0  17.0   1.0  0.0
 0.0  0.0   1.0   3.0   8.0   5.0   6.0   3.0   2.0  0.0
 0.0  0.0   0.0   0.0   0.0   0.0   0.0   1.0   0.0  0.0
 0.0  0.0   0.0   0.0   0.0   0.0   0.0   0.0   0.0  0.0
```

(4). 归一化的 bin 计数

将 1,000 个随机数对组分布到多个 bin 内。将 normalization 指定为 "probability" 以对 bin 计数进行归一化处理，从而 sum(n[:]) 为 1。即，每个 bin 计数代表观测值属于该 bin 的可能性。

```
x = randn(1000, 1);
y = randn(1000, 1);
n,     Xedges,     Yedges,     =     histcounts2(x,     y,     6;
normalization="probability")
```

```
n =
```

```
6×6 Matrix{Float64}:
 0.0    0.0    0.003  0.005  0.002  0.001
 0.0    0.002  0.015  0.036  0.018  0.002
 0.0    0.003  0.068  0.157  0.078  0.009
 0.001  0.003  0.078  0.215  0.102  0.005
 0.0    0.001  0.032  0.089  0.047  0.004
 0.0    0.0    0.004  0.011  0.009  0.0

Xedges =
7-element Vector{Float64}:
 -3.6
 -2.5
 -1.4
 -0.2999999999999998
  0.8000000000000003
  1.9000000000000004
  3.0000000000000004

Yedges =
7-element Vector{Float64}:
 -5.0
 -3.5999999999999996
 -2.1999999999999997
 -0.7999999999999996
  0.6000000000000003
  2.000000000000001
  3.4000000000000004
```

(5). 确定 bin 放置

将介于 -10 和 10 之间的 1000 个随机整数对组分布到多个 bin 内，并将 binmethod 指定为 "integers" 以使用以整数为中心的单位宽度 bin。指定 histcounts2 的五个输出以返回表示数据 bin 位置的向量。

```
x = rand(-10:10, 1000, 1);
y = rand(-10:10, 1000, 1);
n, Xedges, Yedges, binx, biny = histcounts2(x, y;
binmethod="integers");
```

确定值 (x[3], y[3]) 属于哪个 bin。

```
[x[3], y[3]]
```

```
2-element Vector{Int64}:
 8
 3
```

```
bin = [binx[3] biny[3]]
```

```
bin =
1×2 Matrix{Float64}:
 19.0  14.0
```

boxchart

创建箱线图

A. 语法

```
boxchart(ydata)

boxchart(xgroupdata,ydata)

boxchart(___,Key=Value)
boxchart(ax,___)
b = boxchart(___)
```

B. 说明

boxchart(ydata)为矩阵 ydata 的每列创建一个箱线图。如果 ydata 是向量，则 boxchart 只创建一个箱线图。

每个箱线图显示以下信息：中位数、下四分位数和上四分位数、任何离群值（使用四分位差计算得出）以及不是离群值的最小值和最大值。有关详细信息，请参阅箱线图。

boxchart(xgroupdata,ydata)根据 xgroupdata 中的唯一值对向量 ydata 中的数据进行分组，并将每组数据绘制为一个单独的箱线图。xgroupdata 确定每个箱线图在 x 轴上的位置。ydata 必须为向量，xgroupdata 必须与 ydata 长度相同。

boxchart(___,Key=Value)使用一个或多个名称-值对组参数指定其他图选项。例如，您可以通过指定 notch=true 使用缺口来比较样本中位数。在所有其他输入参数之后指定名称-值对组参数。有关属性列表，请参阅 BoxChart 属性。示例

boxchart(ax,___)将图形绘制到 ax 指定的坐标区中，而不是当前坐标区(gca)中。参数 ax 可以置于前面的语法中的任何输入参数组合之前。

b=boxchart(___)返回 BoxChart 对象。创建箱线图后，使用 b 设置箱线图的属性。有关属性列表，请参阅 BoxChart 属性。

C. 示例

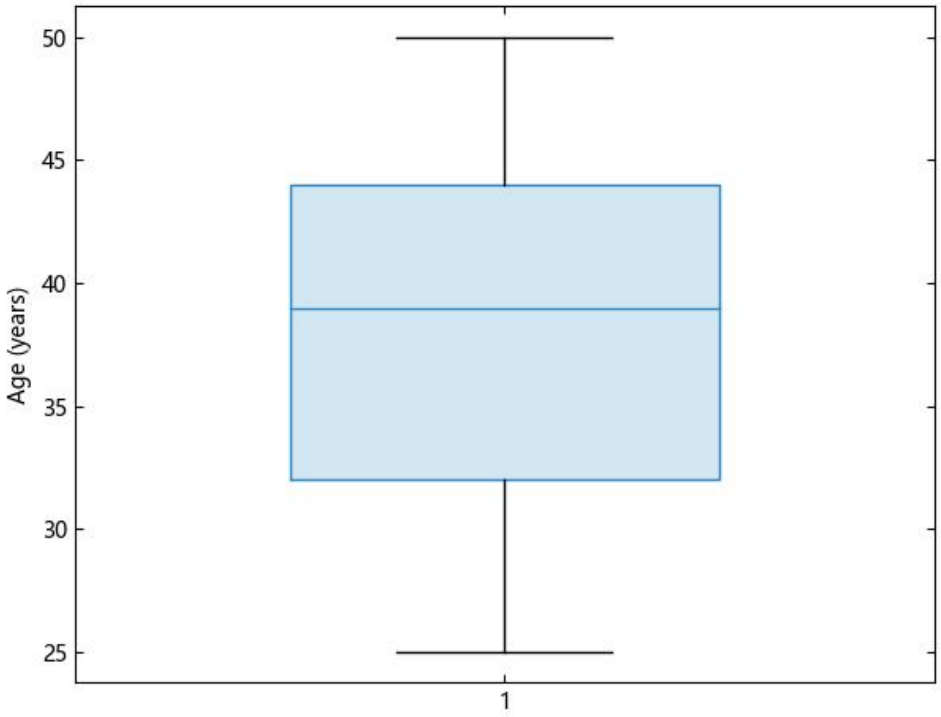
(1). 基于向量数据创建箱线图

导入 0-data.jl 文件。

基于年龄向量创建一个箱线图。使用箱线图了解年龄分布。

加载 patients 数据集。Age 变量包含 100 个患者的年龄信息。创建一个箱线图来可视化年龄分布。

```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/boxchart/0-data.jl"
include(source_path)
res = boxchart(Age)
ylabel("Age (years)")
```



患者的中位数年龄为 39 岁，如方框内的线条所示。32 岁和 44 岁的下四分位数和上四分位数分别显示为框的底部和顶部边缘。须线端点对应于最年轻和最年长的患者。年龄最小的患者是 25 岁，最大的是 50 岁。数据集不包含由小圆圈表示的离群值。

(2). 基于矩阵数据创建箱线图

使用箱线图来比较沿幻方的列和行的值的分布。

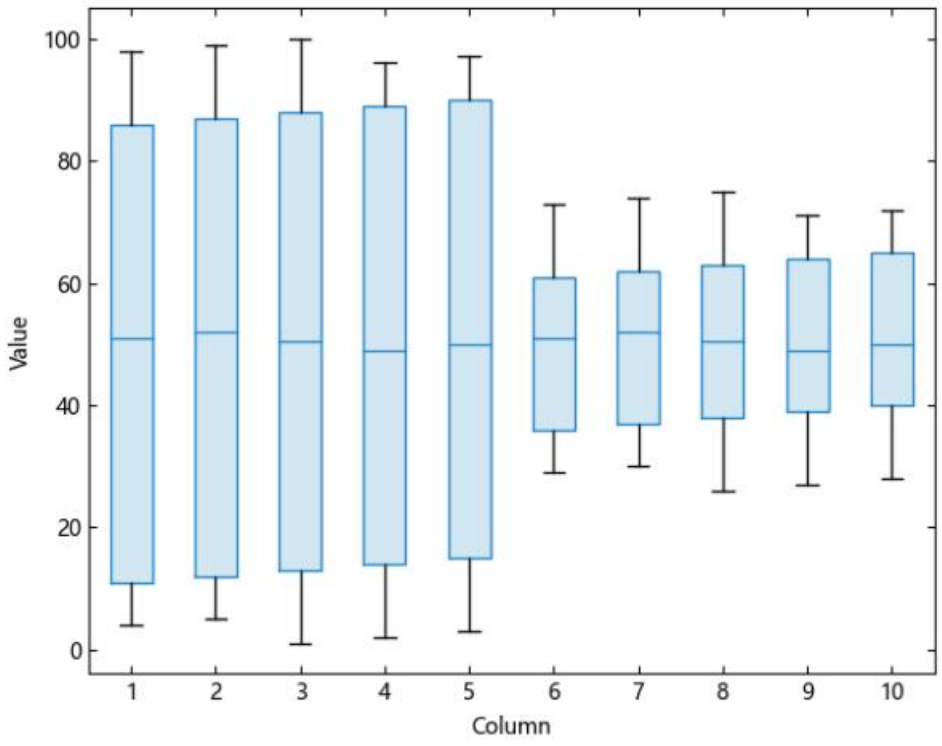
创建一个包含 10 行和 10 列的幻方。

```
Y = magic(10)
```

```
Y =
10×10 Matrix{Int64}:
 92  99    1   8  15  67  74  51  58  40
 98  80    7  14  16  73  55  57  64  41
  4  81   88  20  22  54  56  63  70  47
 85  87   19  21   3  60  62  69  71  28
 86  93   25   2   9  61  68  75  52  34
 17  24   76  83  90  42  49  26  33  65
 23   5   82  89  91  48  30  32  39  66
 79   6   13  95  97  29  31  38  45  72
 10  12   94  96  78  35  37  44  46  53
 11  18  100  77  84  36  43  50  27  59
```

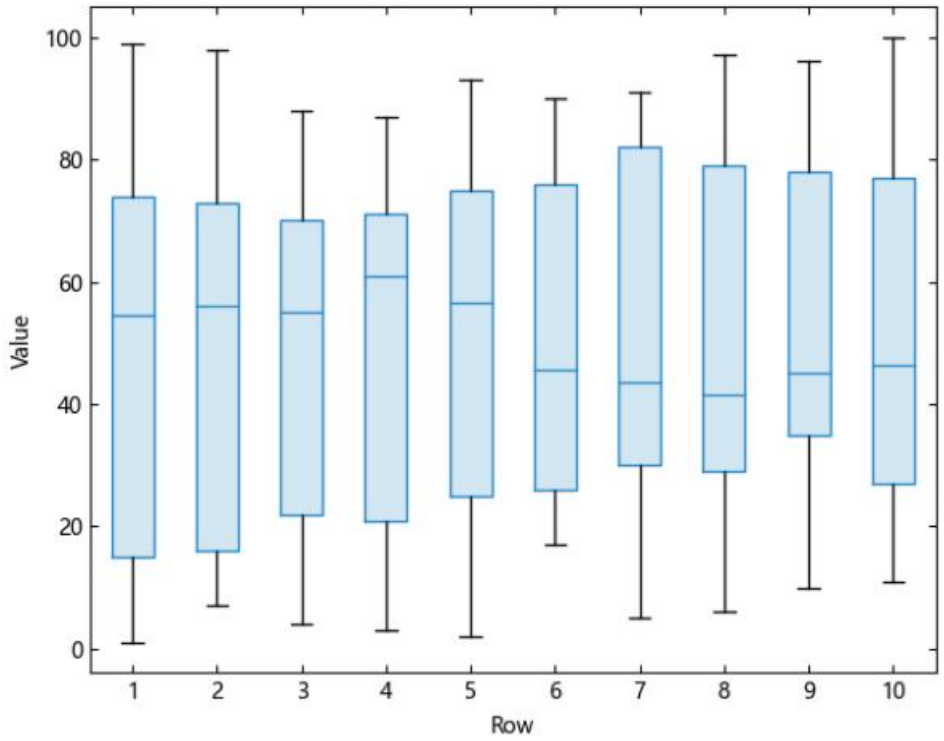
为幻方的每列创建一个箱线图。每列都有一个相似的中位数值（在 50 附近）。但是，Y 的前五列的四分位差大于 Y 的后五列。四分位差是上四分位数（框的上边缘）和下四分位数（框的下边缘）之间的距离。

```
boxchart(Y)
xlabel("Column")
ylabel("Value")
```



为幻方的每行创建一个箱线图。每行都有相似的四分位差，但各行的中位数值不同。

```
figure()
boxchart(Y')
xlabel("Row")
ylabel("Value")
```



(3). 使用分组变量创建多个箱线图

导入 0-data.jl 文件。

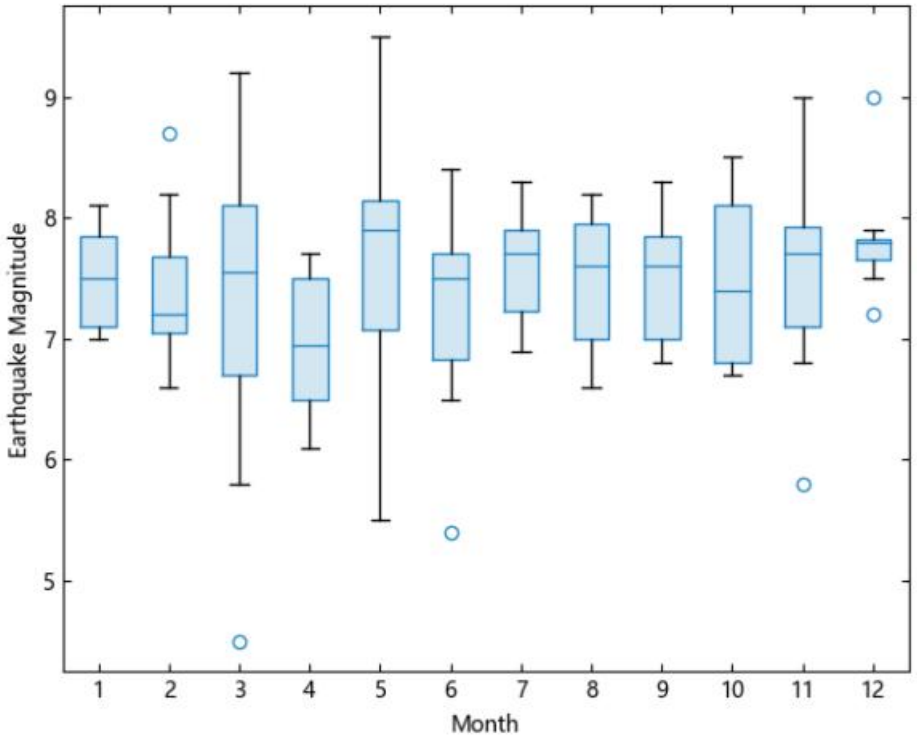
根据地震发生的月份绘制地震的幅值。使用地震幅值向量和分组变量来表示每次地震的月份。对于每组数据，创建一个箱线图，并将其放置在 x 轴上的指定位置。

将一组海啸数据以表形式读入工作区中。该数据集包括关于地震以及海啸的其他原因的信息。显示表的前八行，内容包括月份列、原因列和地震幅值列。

根据对应海啸发生的月份对地震幅值进行分组。对于每个月份，显示一个单独的箱线图。例如，boxchart 使用第四个、第五个和第八个地震幅值及其他幅值来创建第三个箱线图，该箱线图对应于第三个月。

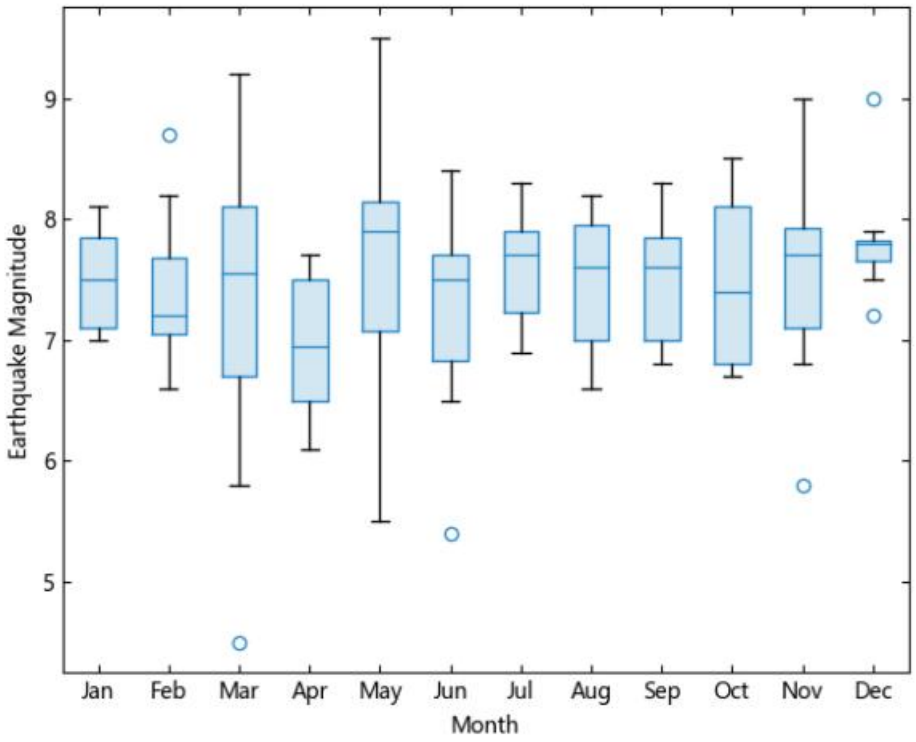
```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/boxchart/0-data.jl"
include(source_path)
res = boxchart(Month, EarthquakeMagnitude)
xlabel("Month")
```

```
ylabel("Earthquake Magnitude")
```



创建与上面相同的箱线图，但使用 monthOrder 代替数值月份值。

```
monthOrder = ["Jan","Feb","Mar","Apr","May","Jun","Jul",
"Aug","Sep","Oct","Nov","Dec"];
res = boxchart(named_Months, EarthquakeMagnitude, order=monthOrder)
xlabel("Month")
ylabel("Earthquake Magnitude")
```



(4). 对数据分 bin 以创建分组变量

导入 0-data.jl 文件。

根据年龄对医疗患者进行分组，并为每个年龄组创建一个关于舒张压值的箱线图。

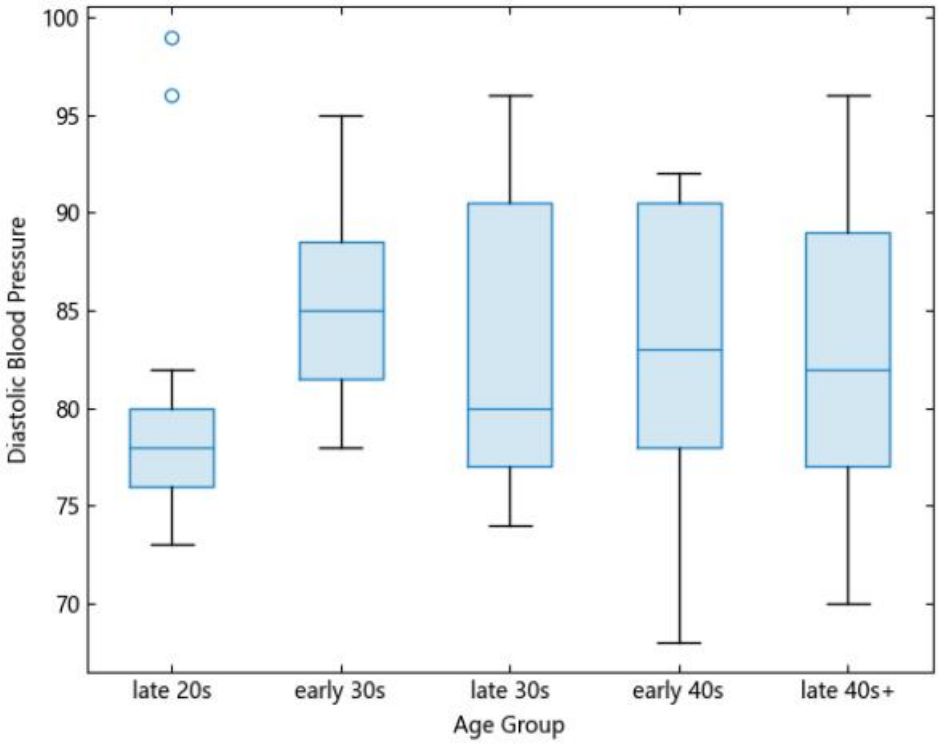
加载 patients 数据集。Age 和 Diastolic 变量包含 100 个患者的年龄和舒张压水平值。

将患者数据分为五个年龄 bin。找出最小和最大年龄，然后将它们之间的范围划分为以五年为一档的 bin。通过使用 discretize 函数对 Age 变量中的值分 bin。使用 bins 中的 bin 名称。

```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/boxchart/0-data.jl"
include(source_path)
bins = ["late 20s","early 30s","late 30s","early 40s","late 40s+"];
boxchart(groupAge,Diastolic,order=bins)
```

为每个年龄组创建一个箱线图。每个箱线图显示该组内患者的舒张压值。

```
xlabel("Age Group")
ylabel("Diastolic Blood Pressure")
```



(5). 合并分组变量

将两个分组变量合并为一个，并使用该变量对生成的箱线图进行分组和定位。

导入 0-data.jl 文件。此文件包含 2015 年 1 月至 2016 年 7 月间的日均温度。将该文件读入一个表中。

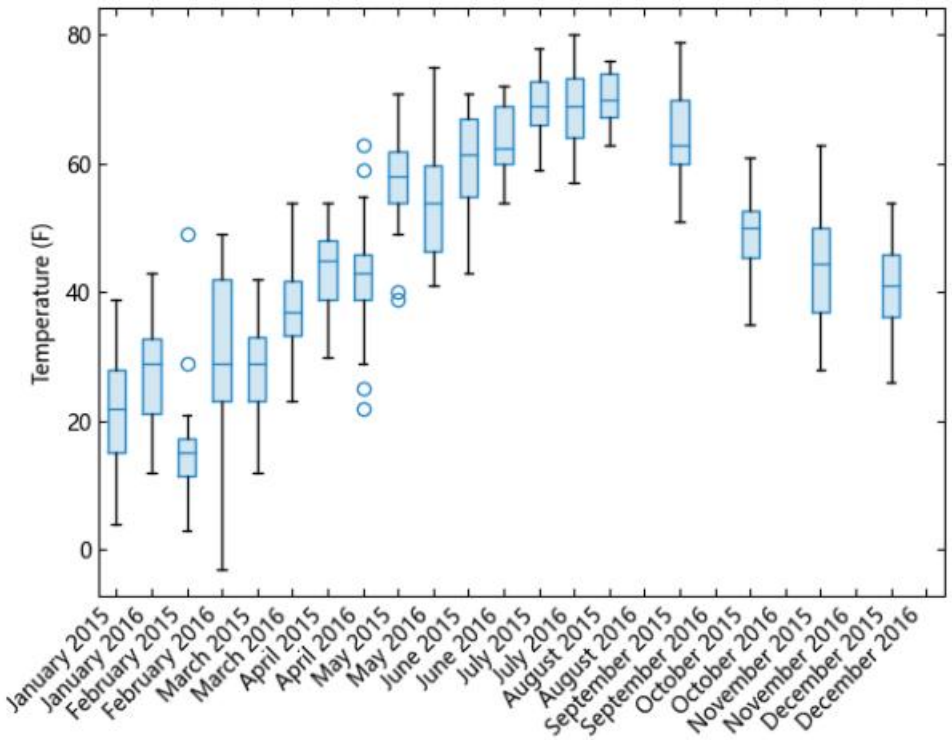
```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/boxchart/0-data.jl"
include(source_path)

order = ["January 2015", "January 2016", "February 2015", "February
2016", "March 2015", "March 2016", "April 2015", "April 2016",
   "May 2015", "May 2016", "June 2015", "June 2016", "July 2015",
"July 2016", "August 2015", "August 2016", "September 2015",
"September 2016",
   "October 2015", "October 2016", "November 2015", "November
2016", "December 2015", "December 2016"];
```

将 Month 和 Year 变量合并成一个分组变量 xgroupdata。创建箱线图，显示每个月-年对组期间的温度分布。

```
boxchart(xgroupdata, TemperatureF, order = order)
```

```
ylabel("Temperature (F)")
```



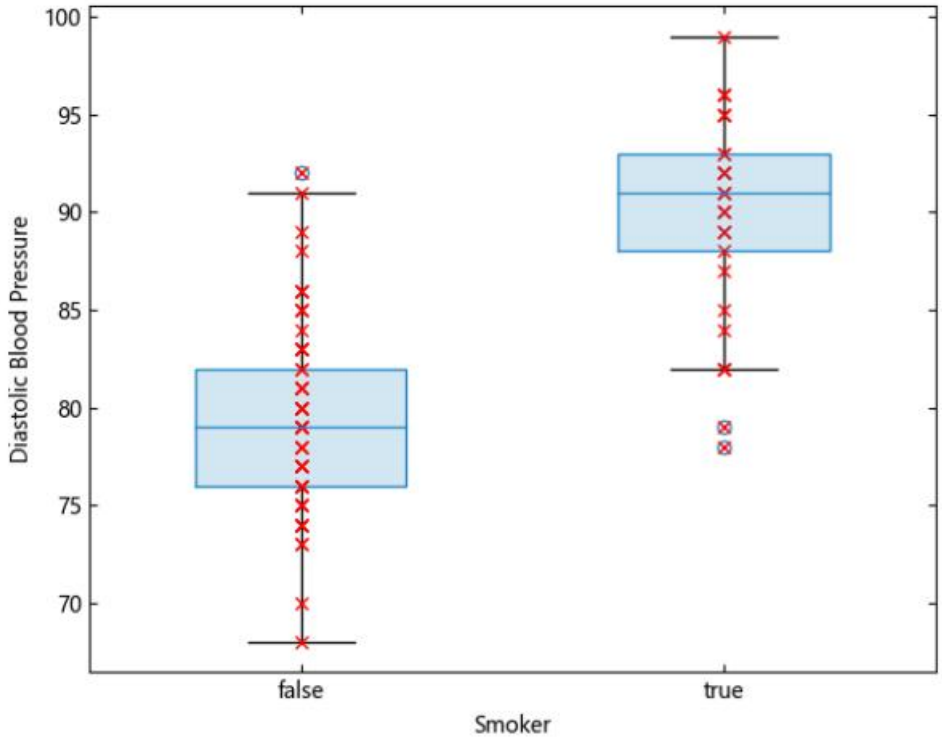
在此图中，您可以轻松比较某个特定月份在多个年份中的的温度分布情况。例如，您可以看到 2016 年 2 月的气温变化比 2015 年大得多。如果您喜欢按时间顺序排列温度，可以将 xgroupdata 设置为 Year.*Month。

(6). 在箱线图上绘制数据

创建箱线图，并使用 hold("on") 在箱线图上绘制数据值。

导入 0-data.jl 文件。将患者分为吸烟者和非吸烟者。使用箱线图比较各组患者的舒张压值。在箱线图上绘制个人的舒张压值。

```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/boxchart/0-data.jl"
include(source_path)
boxchart(Smoker, Diastolic)
hold("on")
Smoker = map(x->(if(x=="true") 2 else 1 end), Smoker)
plot(Smoker,Diastolic,"rx")
xlabel("Smoker")
ylabel("Diastolic Blood Pressure")
hold("off")
```



(7). 使用缺口比较中位数

使用缺口来确定中位数值是否彼此存在显著差异。

导入 0-data.jl 文件。根据患者的位置拆分患者数据。为每组患者创建关于体重的箱线图。指定 notch=true，这样每个框都包括一个称为缺口的锥形着色区域。具有重叠缺口的箱线图没有明显不同的中位数。

```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/boxchart/0-data.jl"
include(source_path)
res = boxchart(Location, Weight, notch=true)
legend([res["boxes"][1]],["box"])
```

在此示例中，三个缺口重叠，表明三个体重中位数没有显著差异。

(8). 指定箱线图的坐标区

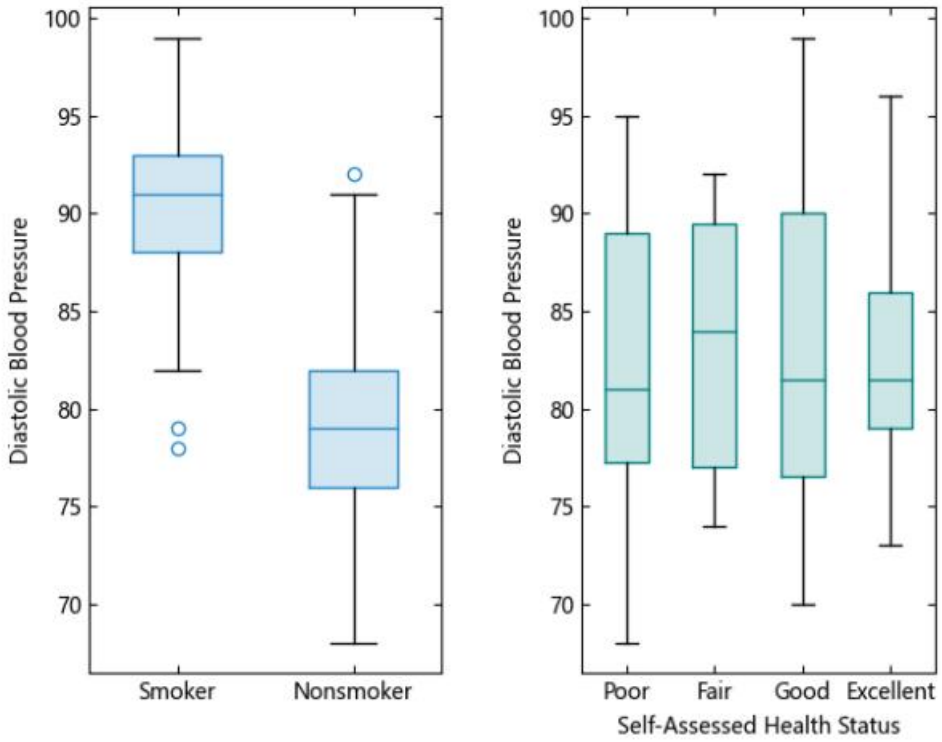
使用缺口来确定中位数值是否彼此存在显著差异。

导入 0-data.jl 文件。调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。通过将 ax1 传递给 plot 函数来创建顶部绘图。通过将坐标区传递给 ylabel 函数，为图添加标题和 y 轴标签。重复该过程以创建底部绘图。在该坐标区中，显示两个关于舒张压值的箱线图，一个用于吸烟者，另一个用于非吸烟者。在该坐标区中，显示四个

关于舒张压值的箱线图，根据自我评估的健康状况对患者进行分组。使用 boxfacecolor 名称-值对组参数指定框的颜色。

```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/boxchart/0-data.jl"
include(source_path)
ax1 = subplot(1,2,1)
Smoker = map(x->(if(x=="true") "Smoker" else "Nonsmoker" end), Smoker)
boxchart(ax1, Smoker, Diastolic, order=["Smoker", "Nonsmoker"])
ylabel("Diastolic Blood Pressure")

ax2 = subplot(1,2,2)
boxchart(ax2, SelfAssessedHealthStatus, Diastolic, order=["Poor","Fair","Good","Excellent"], boxfacecolor=[0, 0.5, 0.5])
xlabel(ax2,"Self-Assessed Health Status")
ylabel(ax2,"Diastolic Blood Pressure")
```



(9). 可视化和查找离群值

使用缺口来确定中位数值是否彼此存在显著差异。

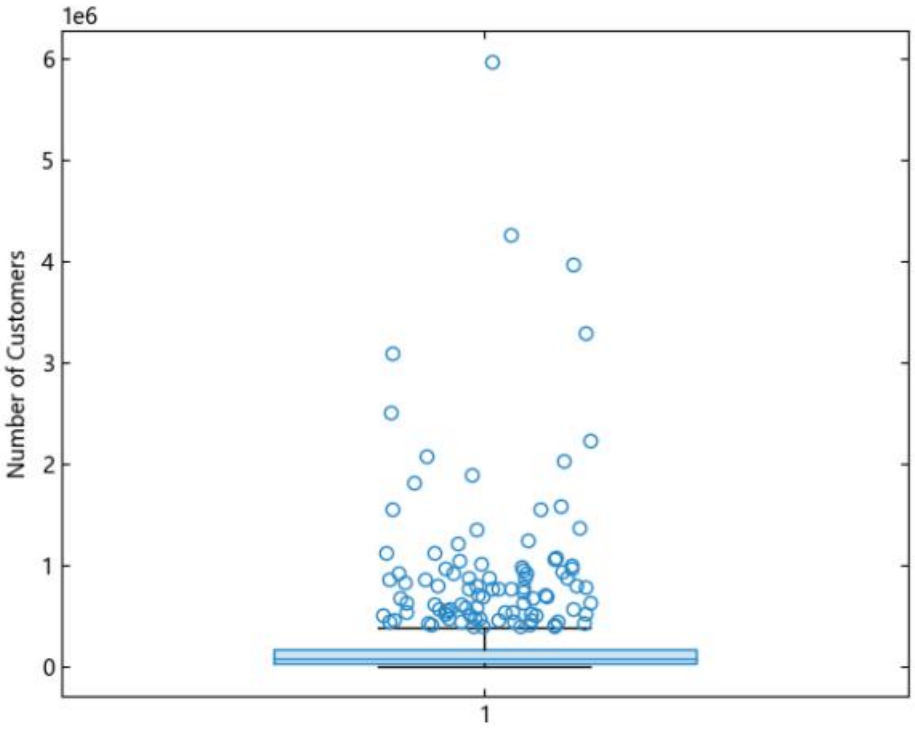
导入 0-data.jl 文件。基于 outages.Customers 值创建一个 BoxChart 对象 b，

这些值指示每次停电所影响的客户数量。boxchart 丢弃值为 NaN 的条目。

```
using TyPlot

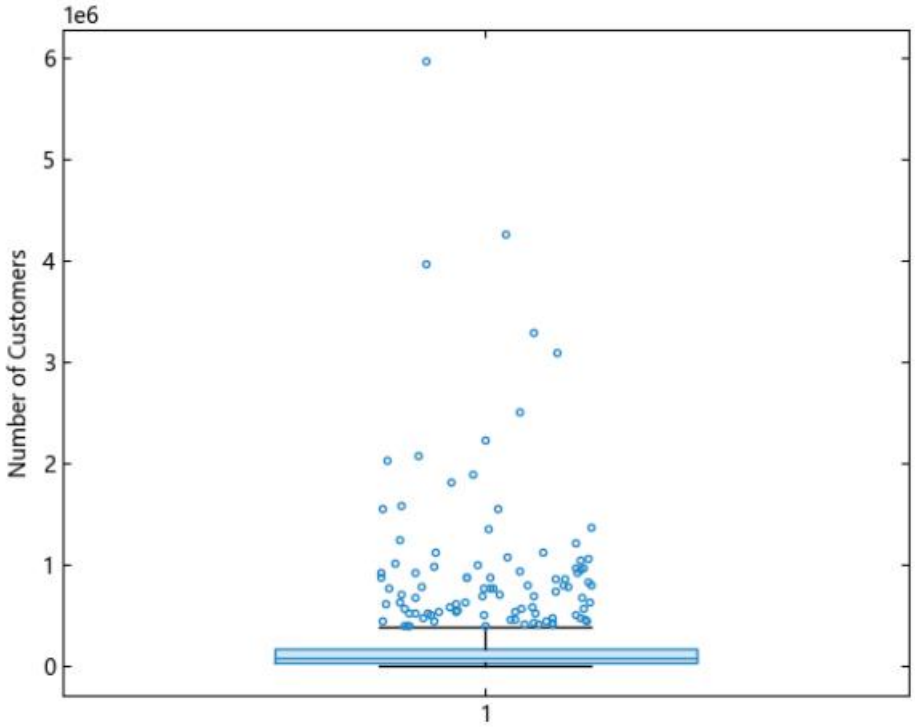
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/boxchart/0-data.jl"
include(source_path)

b = boxchart(Customers; jitteroutliers="on");
ylabel("Number of Customers")
```



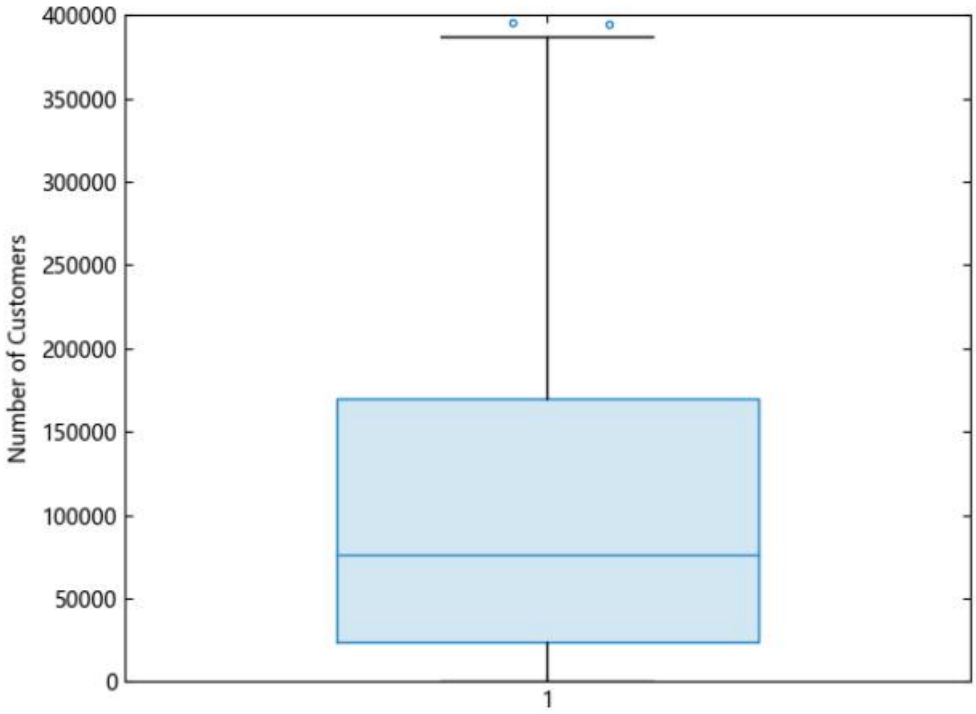
该图包含许多离群值。为了更好地查看它们，抖动离群值并更改离群值标记样式。当您将 BoxChart 对象的 jitteroutliers 属性设置为 "on" 时，软件会随机对离群值标记进行水平位移，使它们不会完全重叠。离群值的值和垂直位置不变。

```
b["fliers"][1].set_marker(".")
```



由于显示了所有离群值，很难查看箱线图的四分位数。要检查它们，请更改 y 轴范围。

```
ylim([0 4e5])
```



3.3.1.2.2 散点图

函数名	简介
scatter	散点图
scatter3	三维散点图
spy	可视化矩阵的稀疏模式
plotmatrix	散点图矩阵

scatter

散点图

A. 语法

```
scatter(x,y)
scatter(x,y,sz)
scatter(x,y,sz,c)
scatter(___,Key=Value)

scatter(ax,___)

s = scatter(___)
```

B. 说明

scatter(X,Y)在向量 x 和 y 指定的位置创建一个包含圆形的散点图。该类型的图形也称为气泡图。

scatter(X,Y,sz)指定圆大小。要绘制大小相等的圆圈，请将 sz 指定为标量。要绘制大小不等的圆，请将 sz 指定为长度等于 x 和 y 的长度的向量。

scatter(X,Y,sz,c)指定圆颜色。要以相同的颜色绘制所有圆圈，请将 c 指定为颜色名称或 RGB 三元组。要使用不同的颜色，请将 c 指定为向量或由 RGB 三元组组成的三列矩阵。

scatter(___,Key=Value)使用一个或多个名称-值对组参数修改散点图。例如，"linewidth=2"将标记轮廓宽度设置为 2 磅。

scatter(ax,___)将在 ax 指定的坐标区中，而不是在当前坐标区中绘制图形。选项 ax 可以位于前面的语法中的任何输入参数组合之前。

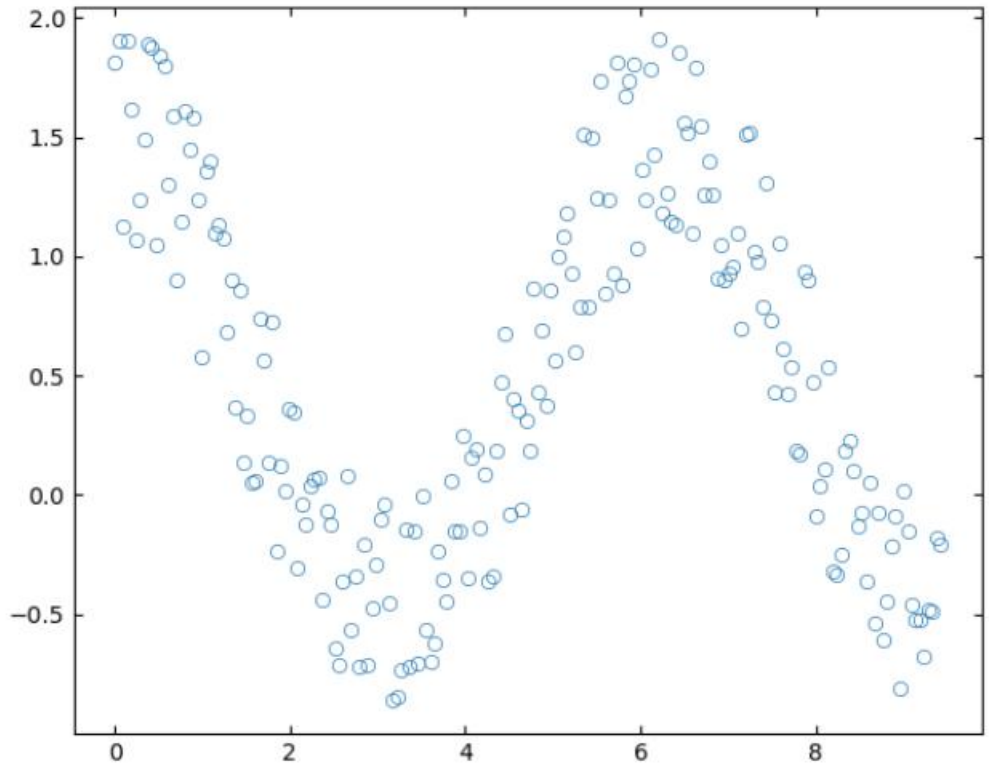
s=scatter(___)返回 Scatter 对象。在创建散点图后，以后可使用 s 对其进行修改。

C. 示例

(1). 创建散点图

创建 x 为 0 和 3π 之间的 200 个等间距值。创建 y 为带随机干扰的余弦值。然后，创建一个散点图。

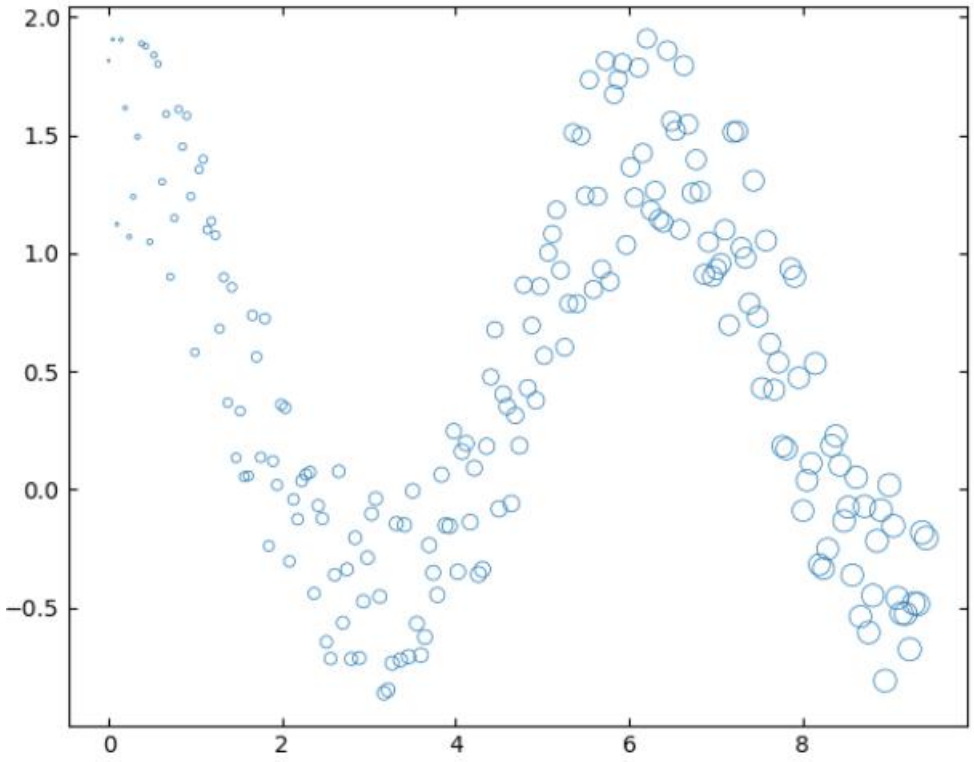
```
x = LinRange(0, 3 * pi, 200)
y = cos.(x) + rand(200)
scatter(x, y)
```



(2). 改变圆圈大小

使用大小不同的圆圈创建一个散点图。以平方磅为单位指定大小。

```
x = LinRange(0, 3 * pi, 200)
y = cos.(x) + rand(200)
s = LinRange(1, 100, 200)
scatter(x, y; s=s)
```



x、y 和 sz 中的相应元素确定每个圆圈的位置和大小。要按照相同的面积绘制所有圆圈，请将 sz 指定为数值标量。

(3). 改变圆圈颜色

创建一个散点图并改变圆圈的颜色。

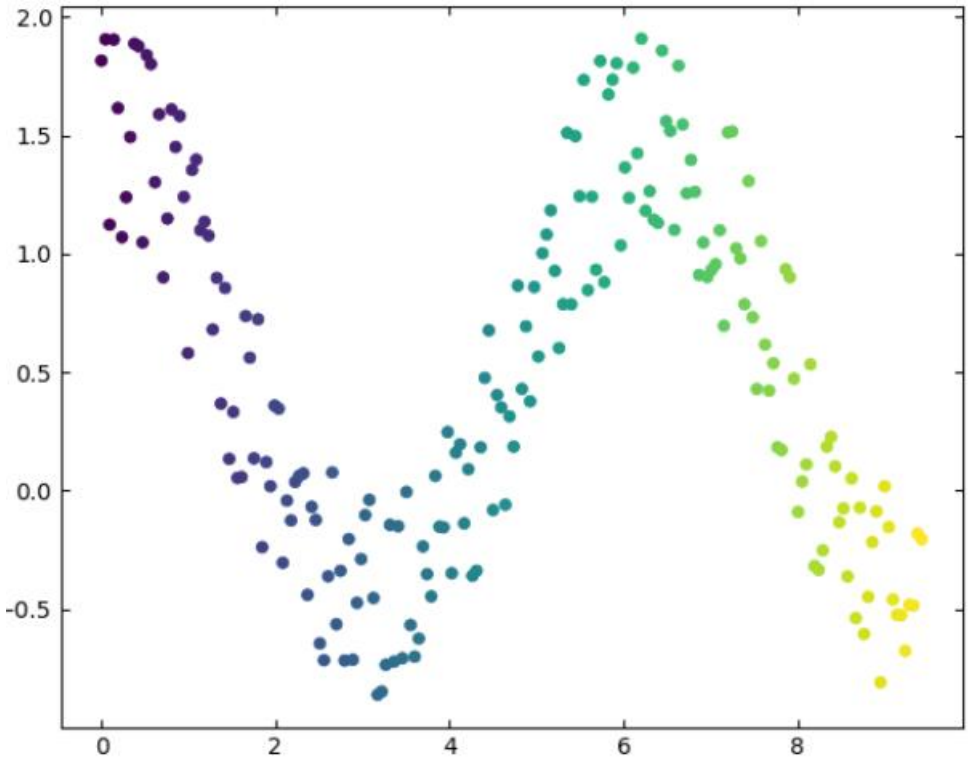
```
x = LinRange(0, 3 * pi, 200)
y = cos.(x) + rand(200)
c = LinRange(1, 10, length(x))
s = scatter(x, y; c=c)
```

x、y 和 c 中的相应元素确定每个圆形的位置和颜色。scatter 函数将 c 中的元素映射到当前颜色图中的元素。

(4). 填充标记

创建一个散点图并填充标记。scatter 使用标记边的颜色填充每个标记。

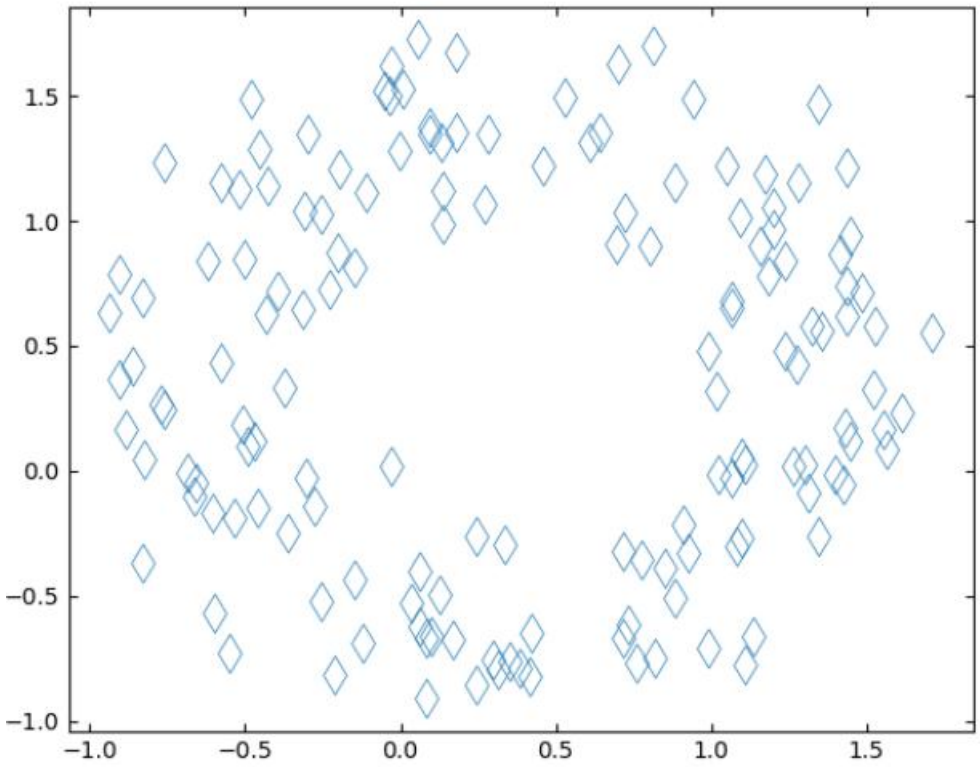
```
x = LinRange(0, 3 * pi, 200)
y = cos.(x) + rand(200)
sz = 25
c = LinRange(1, 10, length(x))
scatter(x, y, sz, c; filled=true)
```



(5). 指定标记符号

创建向量 x 和 y，作为带随机干扰的正弦和余弦值。然后，创建一个散点图，并使用面积为 140 平方磅的菱形标记。

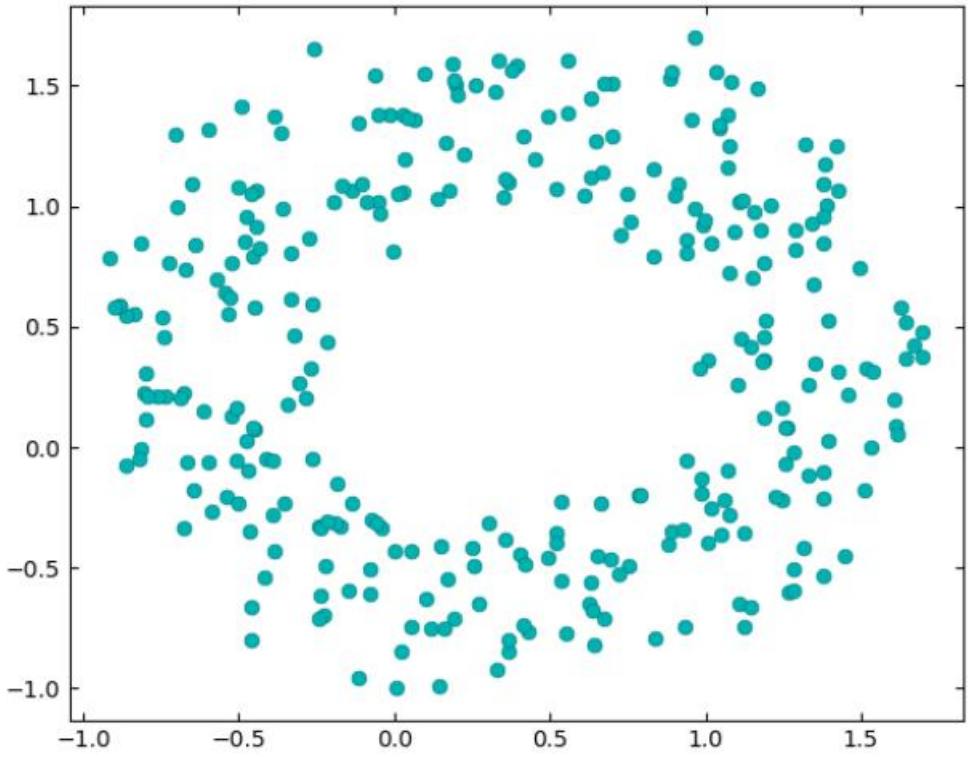
```
theta = LinRange(0, 2 * pi, 150)
x = sin.(theta) + 0.75 .* rand(150)
y = cos.(theta) + 0.75 .* rand(150)
sz = 140
scatter(x, y, sz; marker="d")
```



(6). 更改标记颜色和线条宽度

创建向量 x 和 y，作为带随机干扰的正弦和余弦值。创建一个散点图并设置标记边颜色、标记面颜色和线条宽度。

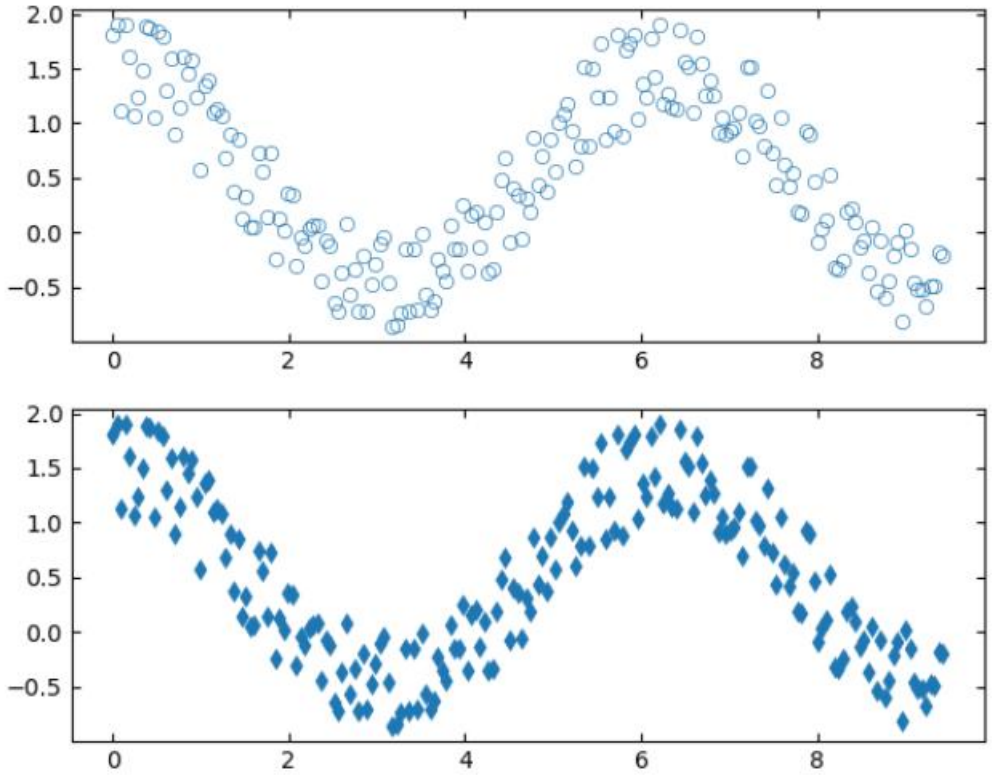
```
theta = LinRange(0, 2 * pi, 300)
x = sin.(theta) + 0.75 .* rand(300)
y = cos.(theta) + 0.75 .* rand(300)
sz = 40
scatter(x, y, sz; linewidths=1.5, markeredgecolor=[0 0.5 0.5],
markerfacecolor=[0 0.7 0.7])
```



(7). 指定目标坐标区和标记类型

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象 ax1 和 ax2 。将在每个坐标区中绘制散点数据。在底部散点图中，指定使用实心菱形标记。

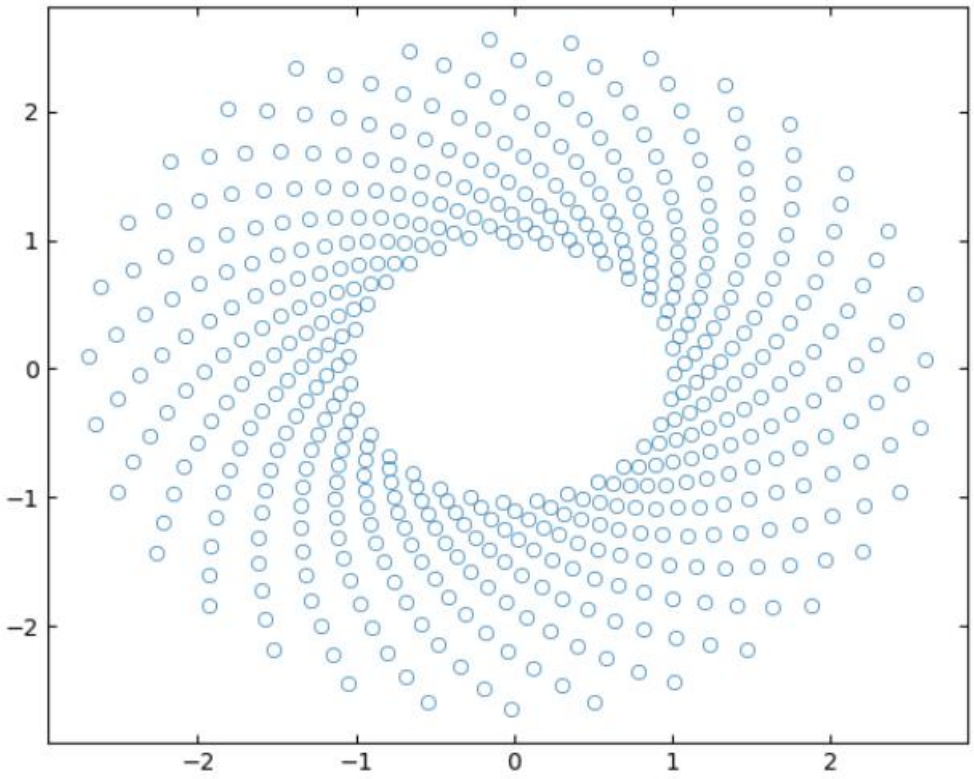
```
x = LinRange(0, 3 * pi, 200)
y = cos.(x) + rand(200)
ax1 = subplot(2, 1, 1)
scatter(ax1, x, y)
ax2 = subplot(2, 1, 2)
scatter(ax2, x, y; filled=true, marker="d")
```



(8). 创建并修改散点序列

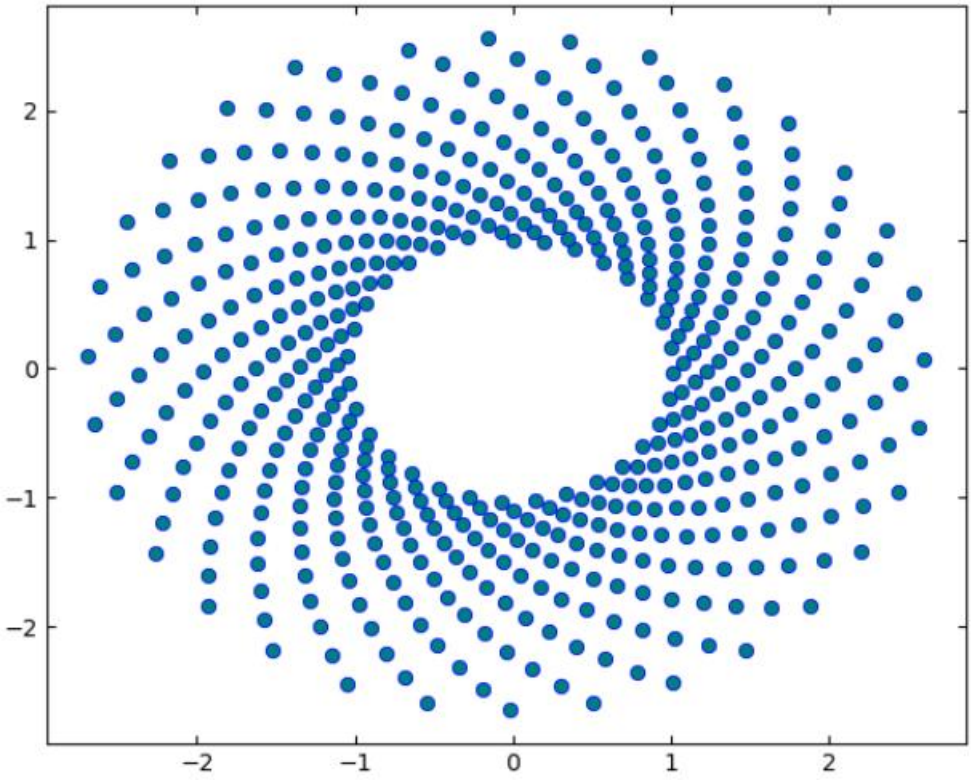
创建一个散点图并返回散点序列对象 s。

```
theta = LinRange(0, 1, 500);
x = exp.(theta) .* sin.(100 * theta);
y = exp.(theta) .* cos.(100 * theta);
s = scatter(x, y)
```



在创建散点序列后，使用 s 查询并设置其属性。将线宽设置为 0.6 磅。将标记边颜色设置为蓝色。使用 RGB 三元组颜色设置标记面。

```
s.set_linewidth(0.6)
s.set_edgecolor("b");
s.set_facecolor([0 0.5 0.5])
```



scatter3

三维散点图

A. 语法

```
scatter3(X,Y,Z)
scatter3(___,Key=Value)

scatter3(ax,___)

h = scatter3(___)
```

B. 说明

scatter3(X,Y,Z)在向量 X、Y 和 Z 指定的位置显示圆圈。

scatter3(___,Key=Value)使用一个或多个名称-值对组参数修改散点图。

scatter3(ax,___)将图形绘制到 ax 指定的坐标区中，而不是当前坐标区(gca)中。选项 ax 可以位于前面的语法中的任何输入参数组合之前。

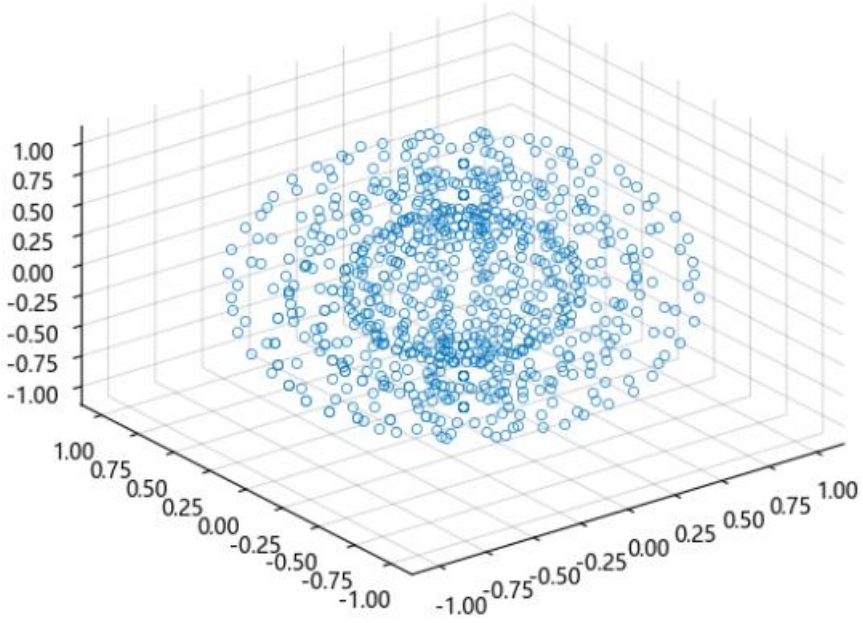
h=scatter3(___)返回 Scatter 对象。在创建散点图后，可使用 h 修改其属性。

C. 示例

(1). 创建三维散点图

导入 0-sphere(16).jl 文件,创建三维散点图。使用 sphere 定义向量 x、y 和 z。

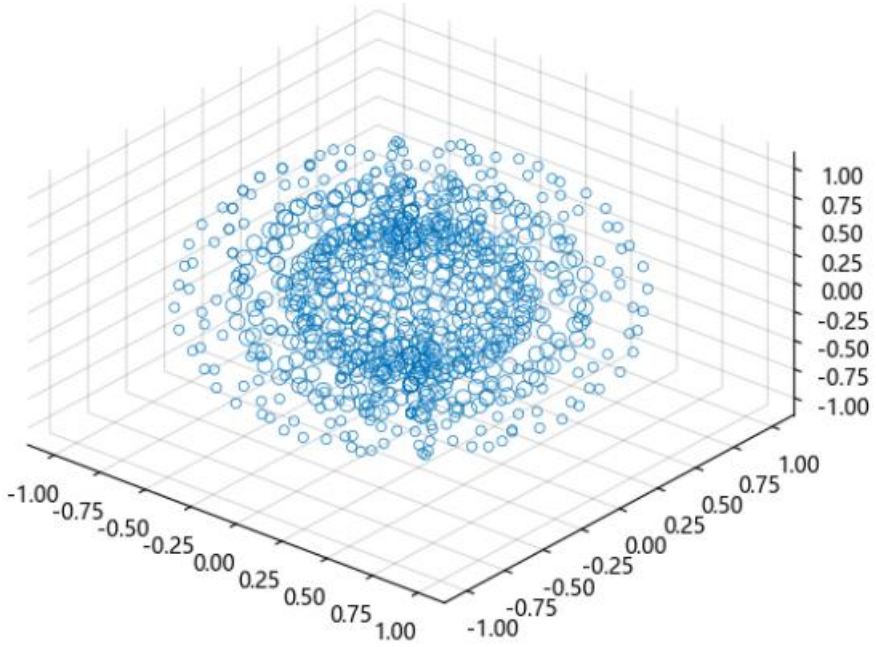
```
using TyPlot: TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/scatter3/0-sphere(16).jl"
include(source_path)
figure()
s = scatter3(x, y, z)
```



(2). 改变标记大小

导入 0-sphere(16).jl 文件,使用 sphere 定义向量 x、y 和 z。定义向量 s 可指定标记大小。创建一个三维散点图。

```
using TyPlot: TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/scatter3/0-sphere(16).jl"
include(source_path)
figure()
scatter3(x,y,z,s=s)
plt_view(40, 35)
```

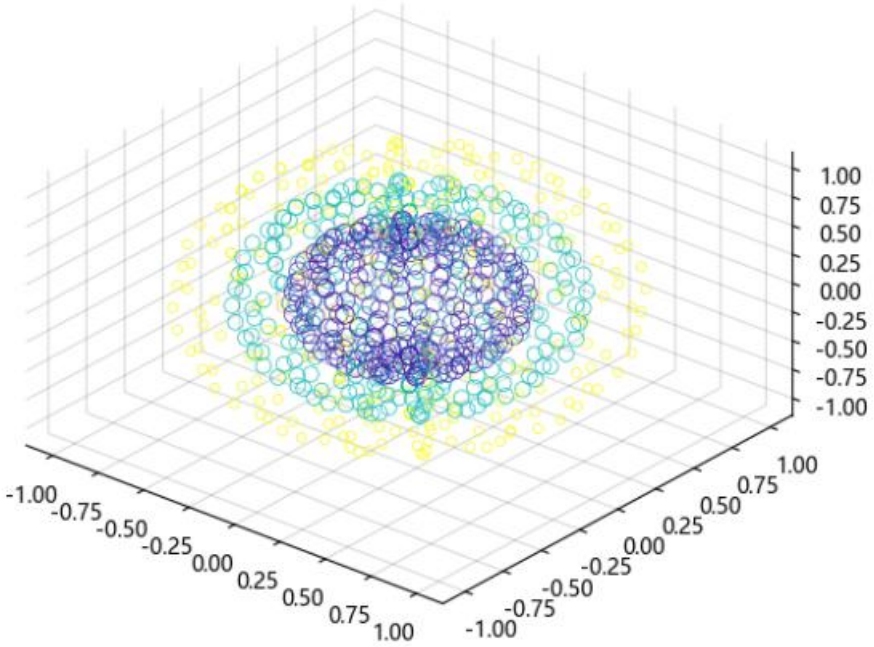


x、y、z 和 s 中的相应项确定每个标记的位置和大小。

(3). 改变标记颜色

导入 0-sphere(16).jl 文件,使用 sphere 定义向量 x、y 和 z。定义向量 s 和 c 以指定每个标记的大小和颜色。创建一个三维散点图。

```
using TyPlot: TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/scatter3/0-sphere(16).jl"
include(source_path)
figure()
s = scatter3(x, y, z; s=s, c=c)
plt_view(40, 35)
```



x、y、z 和 c 中的相应项确定每个标记的位置和颜色。

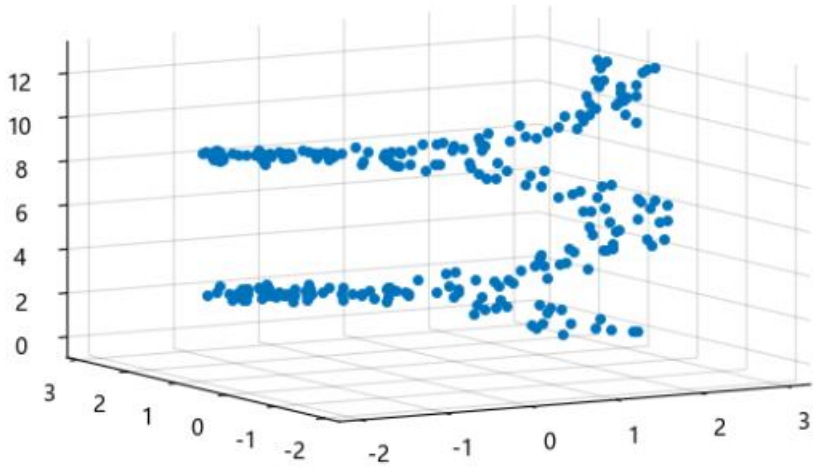
(4). 填充标记

将向量 x 和 y 创建为带随机干扰的余弦和正弦值。

```
z = LinRange(0, 4 * pi, 250);
x = 2 .* cos.(z) .+ rand(250);
y = 2 .* sin.(z) .+ rand(250);
```

创建一个三维散点图并填充标记。

```
scatter3(x, y, z, filled = true)
plt_view(-30, 10)
```



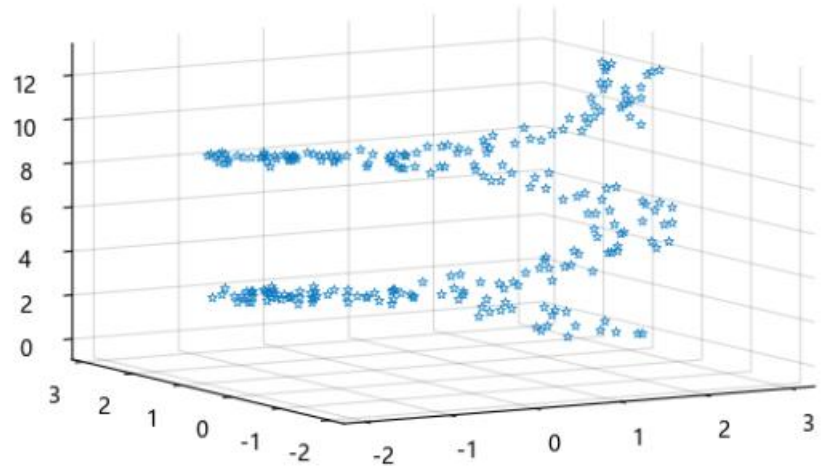
(5). 设置标记类型

初始化随机数生成器以使 rand 的输出可重复。将向量 x 和 y 定义为带随机干扰的余弦和正弦值。

```
z = LinRange(0, 4 * pi, 250);
x = 2 .* cos.(z) .+ rand(250);
y = 2 .* sin.(z) .+ rand(250);
```

创建一个三维散点图并设置标记类型。

```
scatter3(x, y, z, marker = "*")
plt_view(-30, 10)
```



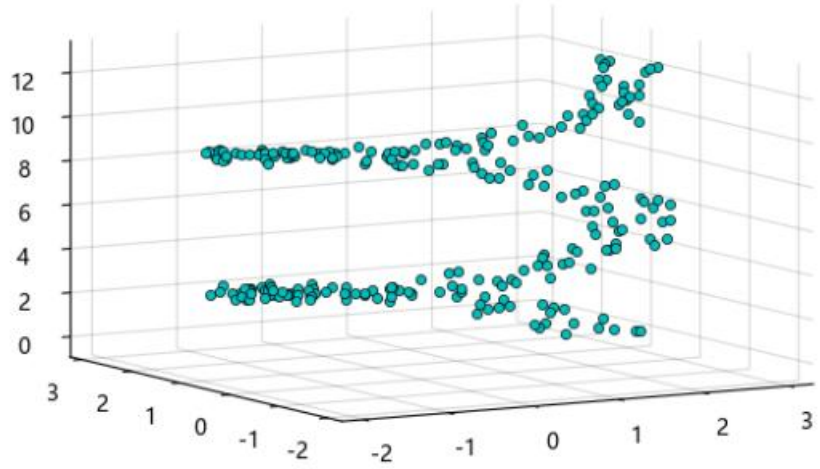
(6). 设置标记属性

初始化随机数生成器以使 rand 的输出可重复。将向量 x 和 y 定义为带随机干扰的余弦和正弦值。

```
z = LinRange(0, 4 * pi, 250)
x = 2 .* cos.(z) .+ rand(250);
y = 2 .* sin.(z) .+ rand(250);
```

创建一个三维散点图并设置标记边颜色和标记面颜色。

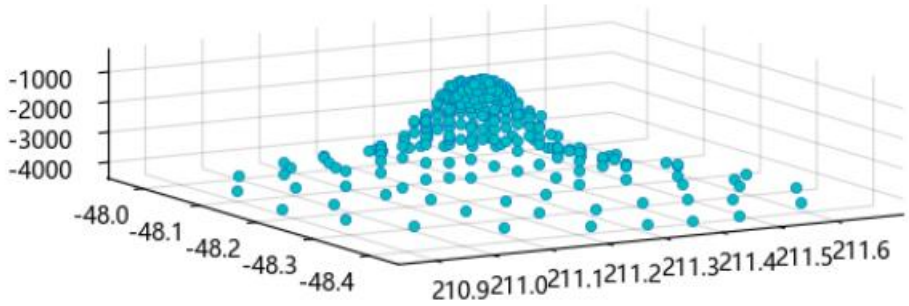
```
scatter3(x, y, z, markeredgecolor = "k", markerfacecolor = [0 0.75
0.75])
plt_view(-30, 10)
```

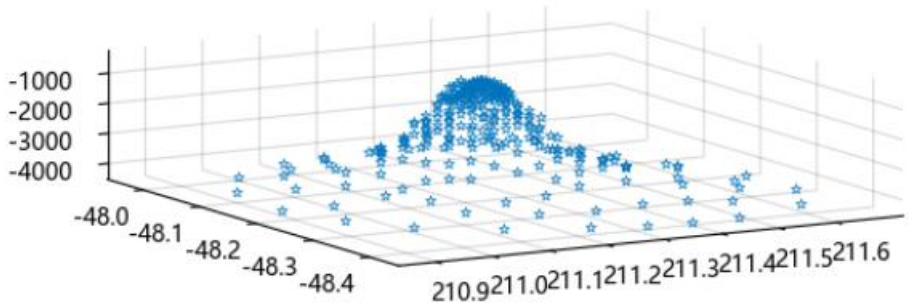


(7). 指定三维散点图的坐标区

导入 0-seamount.jl 文件,调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象 ax1 和 ax2。加载 seamount 数据集以获取向量 x、y 和 z。然后通过将坐标区对象指定为 scatter3 的第一个参数，在坐标区中创建单独的散点图。

```
using TyPlot: TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/scatter3/0-seamount.jl"
include(source_path)
ax1 = subplot(2, 1, 1; projection="3d")
scatter3(ax1, x, y, z; markerfacecolor=[0 0.75 0.75])
ax2 = subplot(2, 1, 2; projection="3d")
scatter3(ax2, x, y, z; marker="*")
```

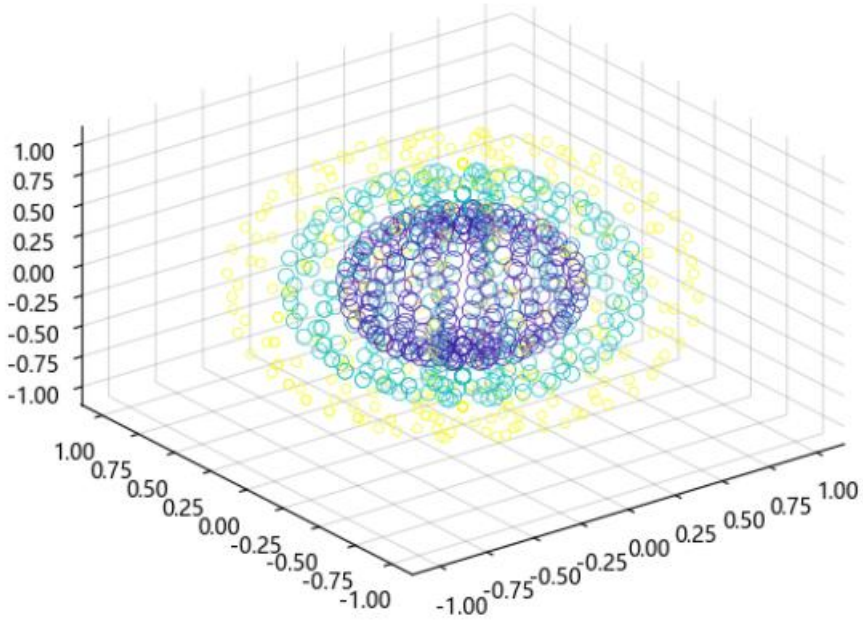




(8). 使用句柄设置散点序列属性

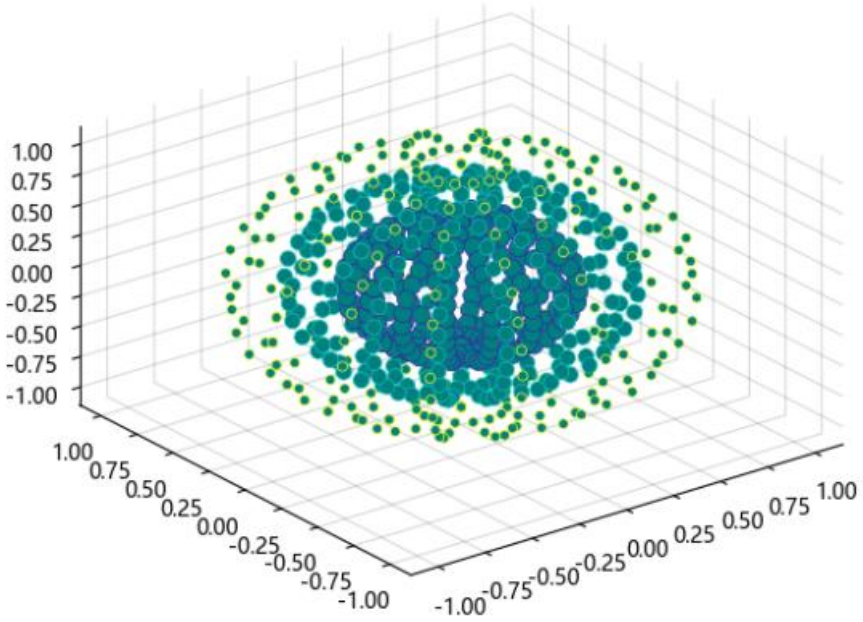
导入 0-sphere(16).jl 文件,创建向量 x、y 和 z。创建向量 s 和 c 以指定每个标记的大小和颜色。创建一个三维散点图并返回散点序列对象。

```
using TyPlot: TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/scatter3/0-sphere(16).jl"
include(source_path)
s = [s...]
figure()
s = scatter3(x, y, z; s=s, c=c)
```



使用 RGB 三元组颜色值设置标记面颜色。使用圆点表示法设置属性。

```
s.set_facecolor([0 0.5 0.5])
```



spy

可视化矩阵的稀疏模式

A. 语法

```
spy(S)
```

```
spy(___,Key=Value)
```

B. 说明

spy(S)绘制矩阵 S 的稀疏模式。非零值是彩色，而零值是白色。该图显示矩阵中的非零元素数。

spy(___,Key=Value)使用一个或多个名称-值对组参数修改稀疏矩阵。例如，marker="o",markersize=8 指定大小为 8 磅的圆形标记。

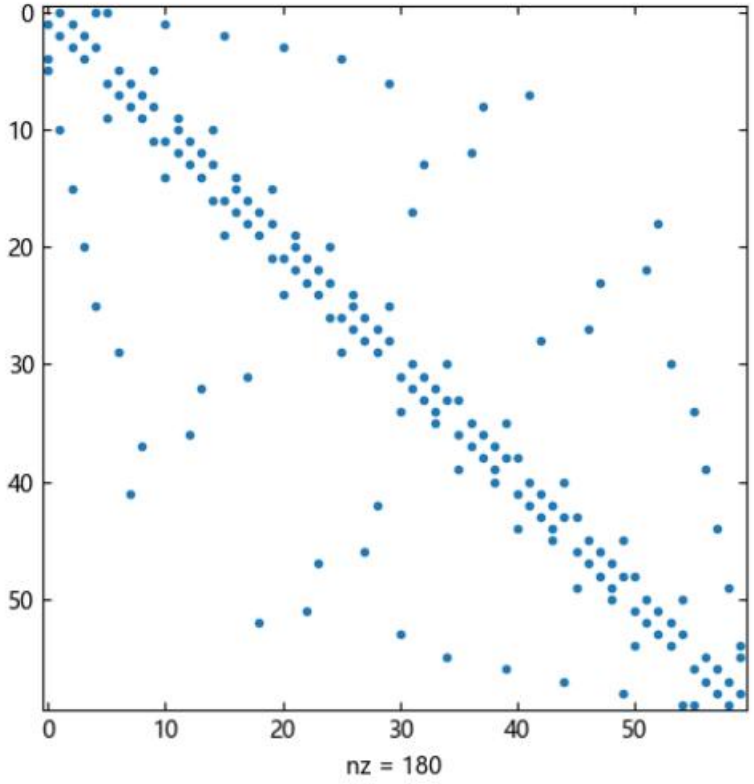
C. 示例

(1). 绘制稀疏模式

导入 0-data.jl 文件,绘制 Buckminster Fuller 多面穹顶的连接图形的 60×60 稀疏邻接矩阵。此矩阵也可表示碳 60 分子和足球。

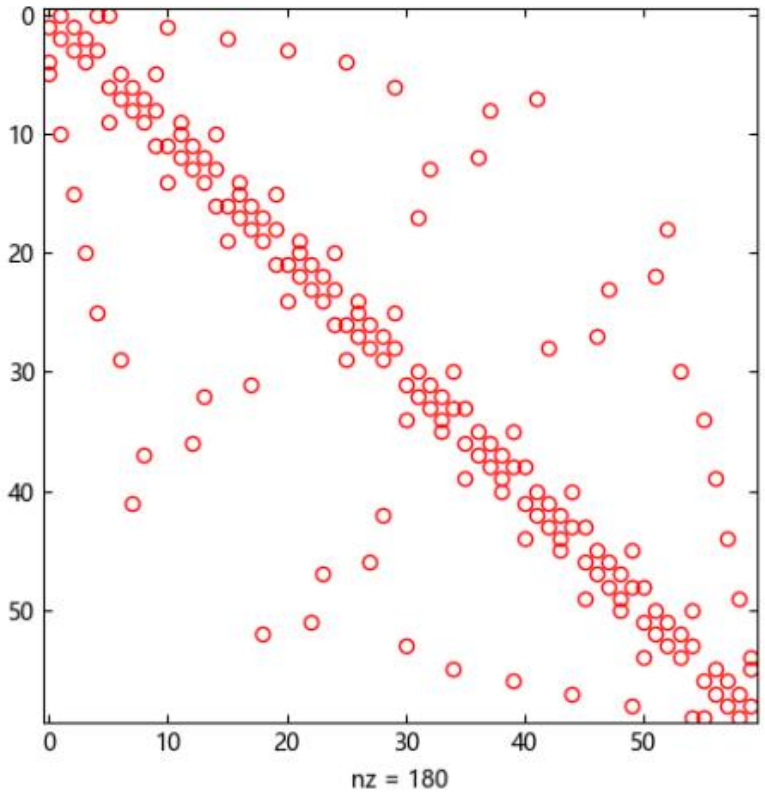
```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/spy/0-data.jl"
include(source_path)
spy(s)

using Printf
n = sum(map(x->(if(x!=0) 1 else 0 end), s))
str_c = @sprintf("%g", n)  # 返回值为字符串
xlabel("nz = " *str_c)
```



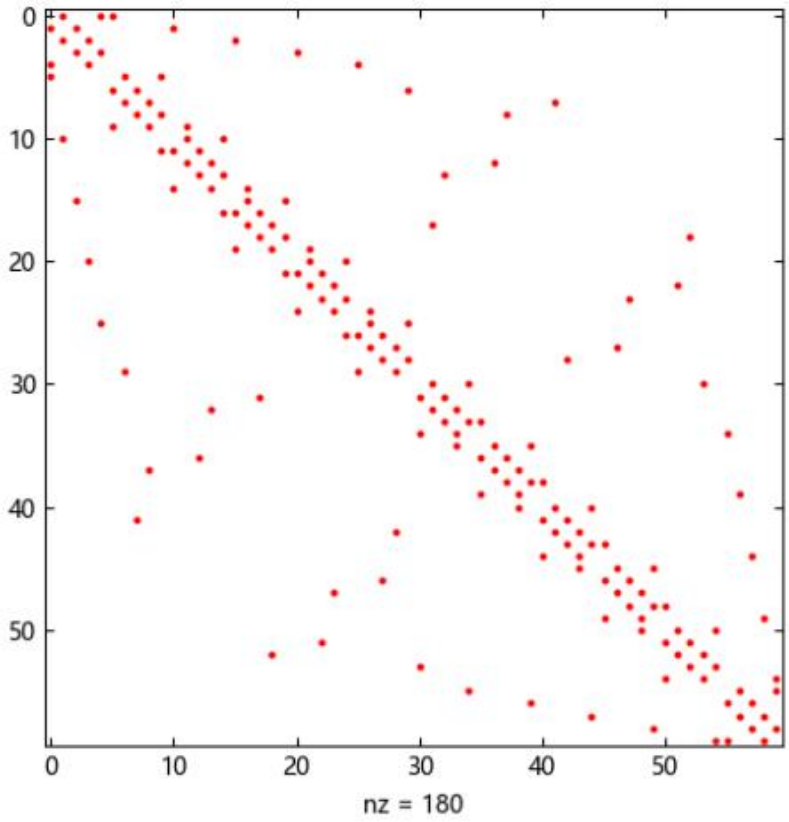
指定颜色和标记。

```
spy(s, marker="o", color="r")
xlabel("nz = " *str_c)
```



还要指定标记大小。

```
spy(s, marker="o", color="r", markersize=2)
xlabel("nz = " *str_c)
```



plotmatrix

散点图矩阵

A. 语法

```
plotmatrix(X,Y)
plotmatrix(X)
plotmatrix(___,Fmt)

[S,AX,BigAx,H,HAx] = plotmatrix(___)
```

B. 说明

plotmatrix(X,Y)创建一个子坐标区矩阵，包含了由 X 的各列相对 Y 的各列数据组成的散点图。如果 X 是 p×n 且 Y 是 p×m，则 plotmatrix 生成一个 n×m 子坐标区矩阵。

除了用 X 对应列中数据的直方图替换对角线上的子坐标区外，

plotmatrix(X)与 plotmatrix(X,X)相同。例如，用 histogram(X[:,i])替换了第 i 列中对角线上的子坐标区。

plotmatrix(___,Fmt)指定散点图的线型、标记符号和颜色。选项 Fmt 可以位于前述语法中的任何输入参数组合之后。

[S,AX,BigAx,H,HAx]=plotmatrix(___)按以下方式返回创建的图形对象：

S - 散点图的图形线条对象

AX - 每个子坐标区的坐标区对象

BigAx - 容纳子坐标区的主坐标区的坐标区对象

H - 直方图的直方图对象

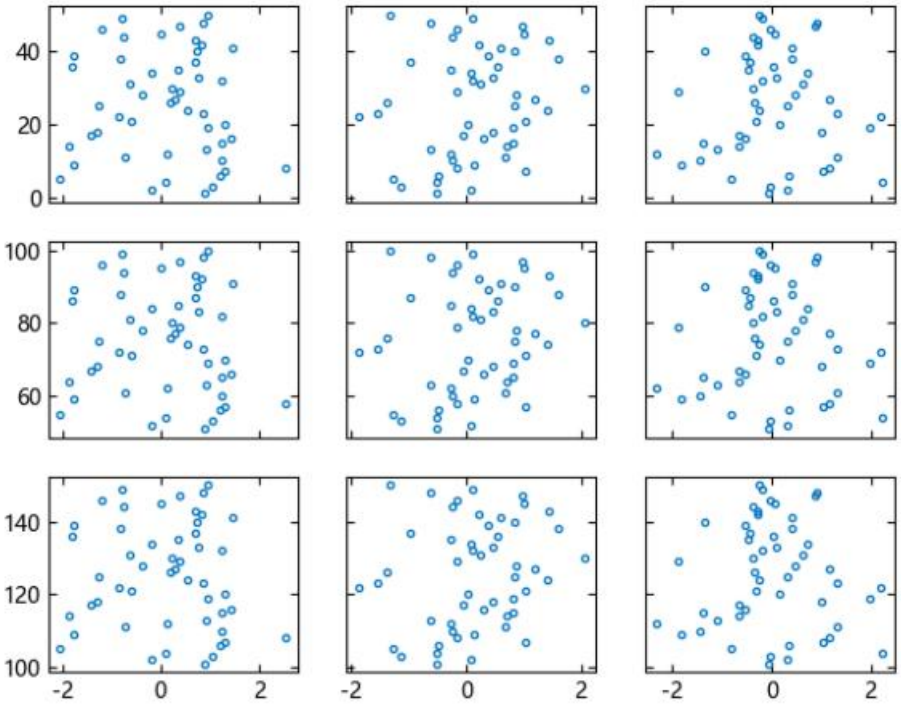
HAx - 不可见的直方图坐标区的坐标区对象 BigAx 被当作当前坐标区(gca)，因此后续的 title、xlabel 或 ylabel 命令都将依此主坐标区使文本居中。

C. 示例

(1). 使用两个矩阵输入创建散点图矩阵

创建一个由随机数据组成的矩阵 X，以及一个由整数值组成的矩阵 Y。然后，创建 X 的各列对 Y 的各列的散点图矩阵。

```
X = randn(50,3);
Y = reshape(1:150,50,3);
plotmatrix(X,Y)
```



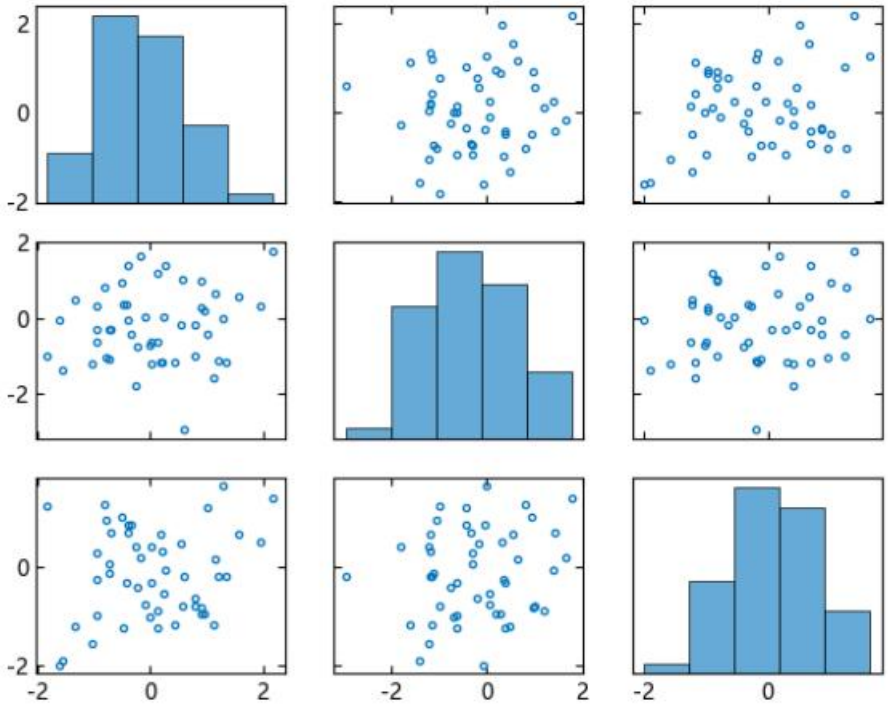
图窗的第 i 行、第 j 列中的子图是 Y 的第 i 列相对于 X 的第 j 列的散

点图。

(2). 使用一个矩阵输入创建散点图矩阵

创建包含随机数据的散点图矩阵。矩阵的第 i 行、第 j 列中的子图是 X 的第 i 列相对于 X 的第 j 列的散点图。沿对角线方向是 X 的每一列的直方图。

```
X = randn(50,3);
plotmatrix(X)
```



(3). 指定标记类型和颜色

创建包含随机数据的散点图矩阵。指定散点图的标记类型和颜色。

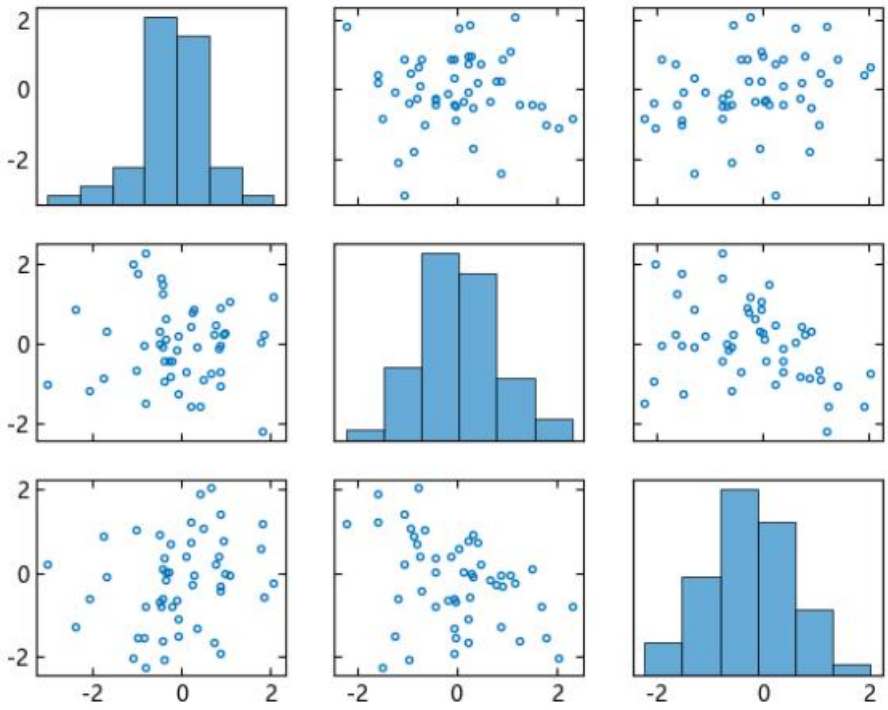
```
X = randn(50,3);
plotmatrix(X,"*r")
```

Fmt 选项设置散点图的属性。要设置直方图的属性，需返回直方图对象。

(4). 创建并修改散点图矩阵

创建包含随机数据的散点图矩阵。

```
X = randn(50,3);
S,AX,BigAx,H,HAx = plotmatrix(X);
```

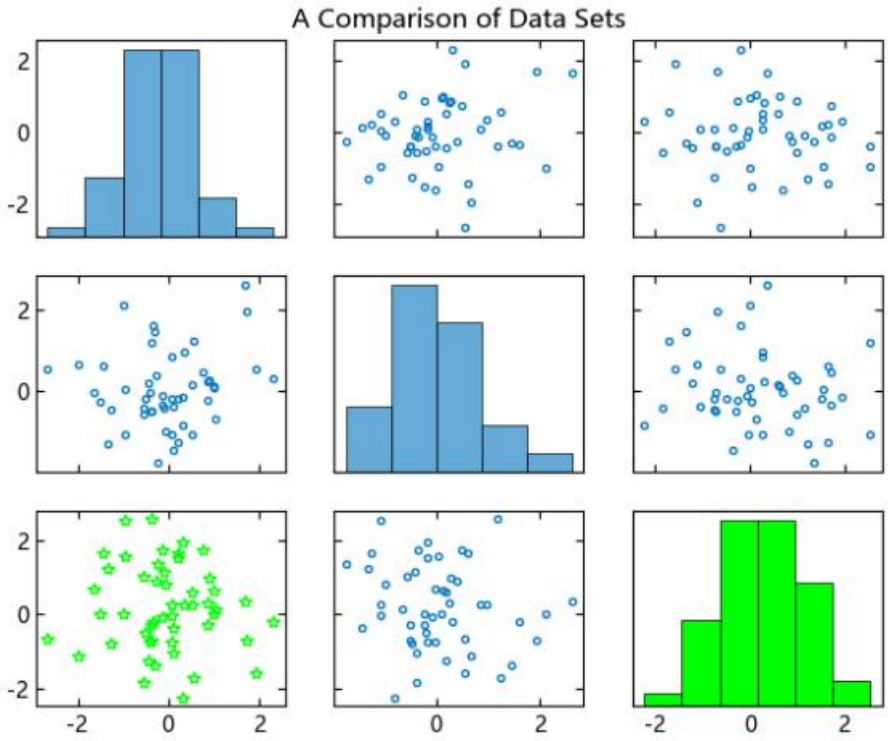


要设置散点图的属性，请使用 S。要设置直方图的属性，请使用 H。要设置坐标区属性，请使用 AX、BigAx 和 HAx。使用圆点表示法设置属性。在图窗的左下角设置散点图的颜色和标记类型。设置位于右下角的直方图的颜色。使用 title 命令为图窗添加标题。

```
S[3][1].set_color([0, 1, 0]);
S[3][1].set_marker("*");

for h in H[3][3]
    h.set_edgecolor("k");
    h.set_facecolor([0,1,0]);
end

title(BigAx, "A Comparison of Data Sets")
```



3.3.1.2.3 饼图和热图

函数名	简介
pie	饼图
heatmap	创建热图
sortx	对热图行中的元素进行排序
sorty	对热图列中的元素进行排序
wordcloud	使用文本数据创建文字云图

pie

饼图

A. 语法

```
pie(X)
pie(X, Key=Value)
```

```
pie(ax,___)

p = pie(___)
```

B. 说明

pie(X) 使用 X 中的数据绘制饼图。饼图的每个扇区代表 X 中的一个元素。

如果 sum(X) ≤ 1，X 中的值直接指定饼图扇区的面积。如果 sum(X) < 1，pie 仅绘制部分饼图。

如果 sum(X) > 1，则 pie 通过 X/sum(X) 对值进行归一化，以确定饼图的每个扇区的面积。

pie(X,Key=Value) 使用一个或多个 Key=Value 对组参数指定饼图属性。有关属性列表，请参阅 Pie 属性 (opens new window)。可以将此选项与前面语法中的任何输入参数组合一起使用。名称-值对组设置将应用于绘制的所有饼图。

pie(ax,___)将图形绘制到 ax 指定的坐标区中，而不是当前坐标区(gca)中。选项 ax 可以位于前面的语法中的任何输入参数组合之前。

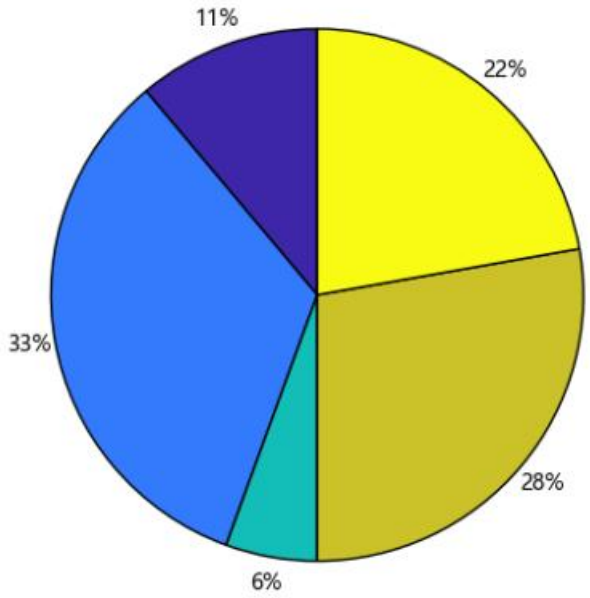
p=pie(___)返回一个由补片和文本图形对象组成的向量。该输入可以是先前语法中的任意输入参数组合。

C. 示例

(1). 创建带偏移扇区的饼图

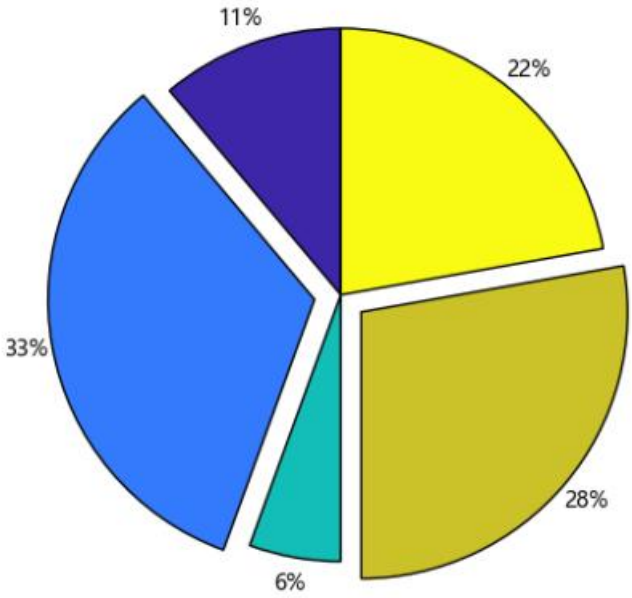
创建向量 X 的饼图。

```
X = [1, 3, 0.5, 2.5, 2]
pie(X)
```



通过将对应的 explode 元素设置为 0.1 来偏移第二和第四块饼图扇区。

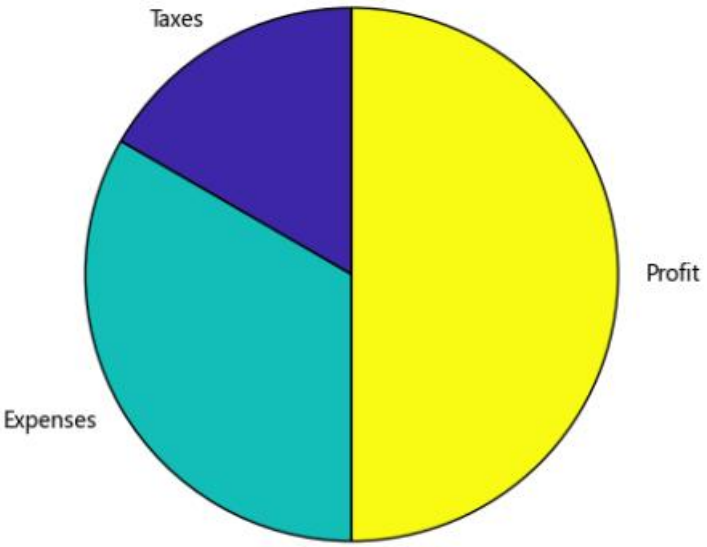
```
pie(X, explode = [0, 0.1, 0, 0.1, 0], autopct = "%.0f%%")
```



(2). 指定饼图的文本标签

创建向量 X 的饼图并标记扇区。

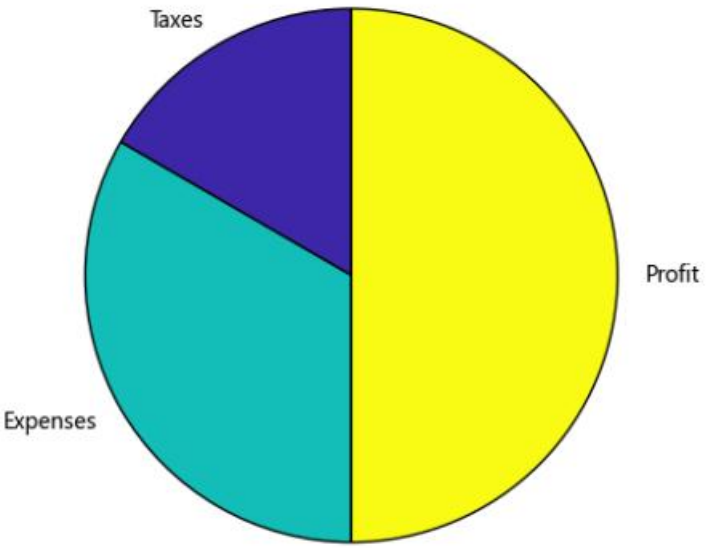
```
X = 1:3
labels = ["Taxes","Expenses","Profit"]
pie(X, labels = labels)
```



(3). 修改饼图的文本标签

创建一个标签饼图，然后修改文件标签的颜色和字体大小。

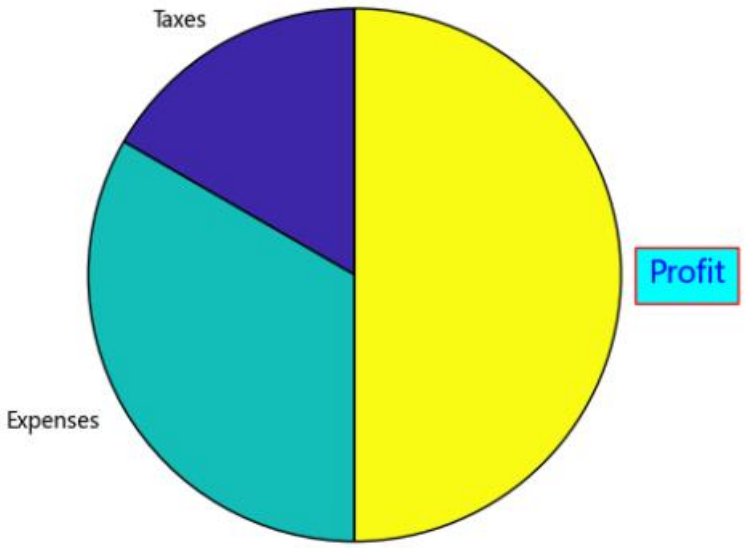
```
X = 1:3
labels = ["Taxes","Expenses","Profit"]
t = pie(X, labels = labels)
```



获取标签“Profit”的文本对象。更改其颜色和字体大小。使用圆点表示法设置属性。

```
t[2][3].set_backgroundcolor("cyan")
if t[2][3]._bbox_patch === nothing
```

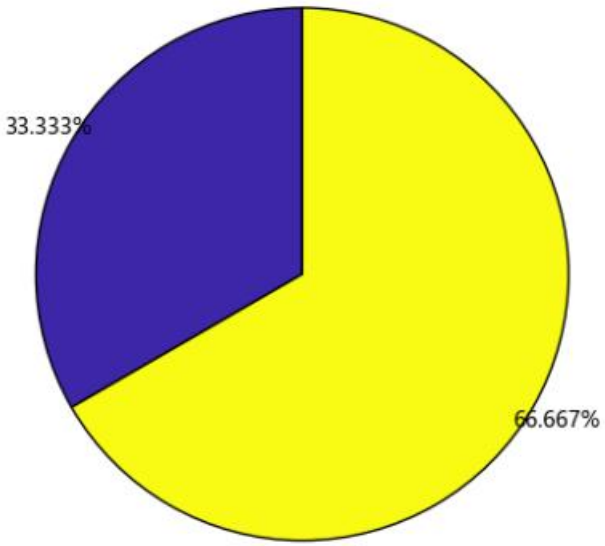
```
    t[2][3].set_bbox(Dict())
end
t[2][3]._bbox_patch.set_edgecolor("red")
t[2][3].set_color("b")
t[2][3].set_fontsize(14)
```



(4). 指定百分比标签的格式

创建一个饼图，并指定格式表达式以使每个标签显示小数点后三位数。要在标签中包含百分号，请在表达式末尾指定 "%%"。

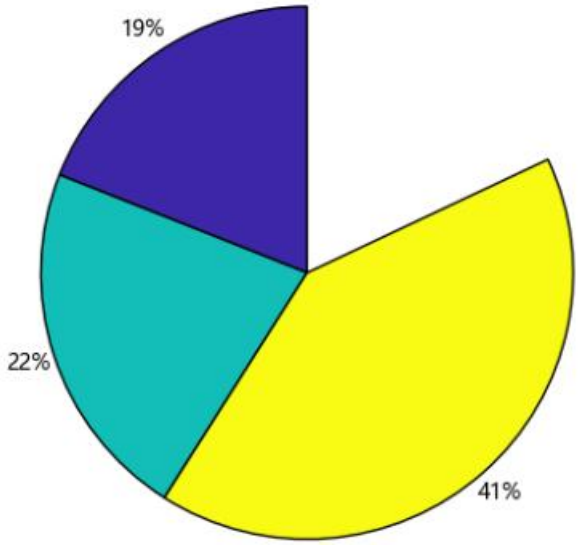
```
X = [1/3, 2/3]
pie(X, autopct = "%.3f%%")
```



(5). 绘制部分饼图

创建一个由向量 X 组成的饼图，其中各个元素之和小于 1。

```
X = [0.19, 0.22, 0.41]
pie(X, autopct = "%.0f%%")
```



由于元素的总和小于 1，因此 pie 绘制部分饼图。

(6). 比较两个饼图

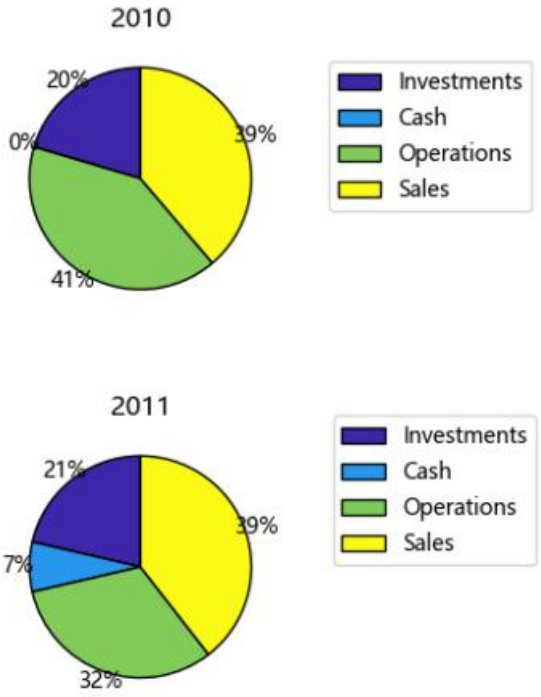
创建包含两年财务数据的向量 y2010 和 y2011。然后创建一个包含值标签的数组。

```
y2010 = [50, 0, 100, 95]
y2011 = [65, 22, 97, 120];
labels = ["Investments","Cash","Operations","Sales"]
```

创建一个 2×1 分块图布局，并在第一个图块中显示 y2010 数据的饼图和图例。然后在第二个图块中显示 y2011 数据的饼图和图例。

```
ax1 = subplot(2,1,1)
pie(ax1, y2010)
title("2010")
legend(ax1, labels)

ax2 = subplot(2,1,2)
pie(ax2, y2011)
legend(ax2, labels)
title("2011")
```



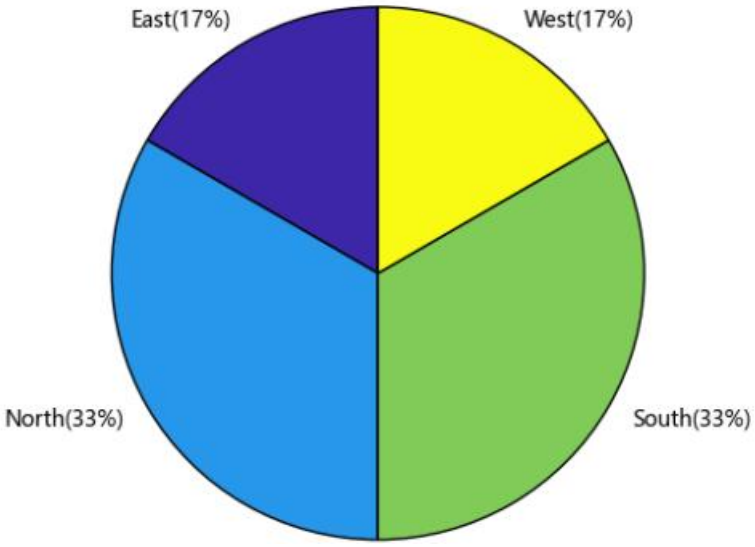
(7). 绘制具有偏移的分类饼图

绘制一个分类饼图，其中的偏移扇区与类别相对应。

```
X = ["North","South","North","East","South","West"]
x_names = ["East", "North", "South", "West"]
x_values =[1, 2, 2, 1]

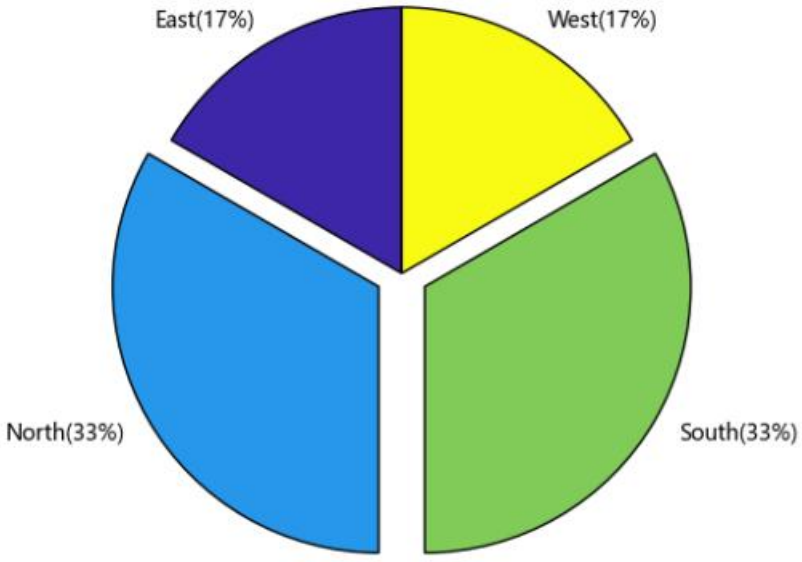
x_labels = []
amount = sum(x_values)
for i in 1:length(x_names)
    value::Int = round(x_values[i]/amount*100)
    name = x_names[i] * "(" * string(value) * "%)"
    push!(x_labels, name)
end

pie(x_values, labels = x_labels, autopct = "%.0f%%")
```



现在，使用逻辑向量将相同扇区偏移一定位置。

```
pie(x_values, labels = x_labels, explode = [0, 0.1, 0.1, 0],
autopct = "%.0f%%")
```



heatmap

创建热图

D. 语法

```
heatmap(xvar,yvar)
```

```
heatmap(cdata)
heatmap(___,Key=Value)
heatmap(ax,___)
h = heatmap(___)
```

E. 说明

heatmap(xvar,yvar)基于数据创建一个热图。xvar 输入参数指示沿 x 轴显示的表变量。yvar 输入参数指示沿 y 轴显示的表变量。默认颜色基于计数聚合，这种方法计算每对 x 和 y 值一起出现在表中的总次数。

heatmap(cdata)基于矩阵 cdata 创建一个热图。热图上的每个单元格对应 cdata 中的一个值。

heatmap(___,Key=Value)使用一个或多个名称-值对组参数指定热图的其他选项。请在所有其他输入参数之后指定这些选项。

heatmap(ax,___) 在 ax 指定的坐标区中创建热图。

h = heatmap(___) 返回 HeatmapChart 对象。创建图后，使用 h 修改图属性。

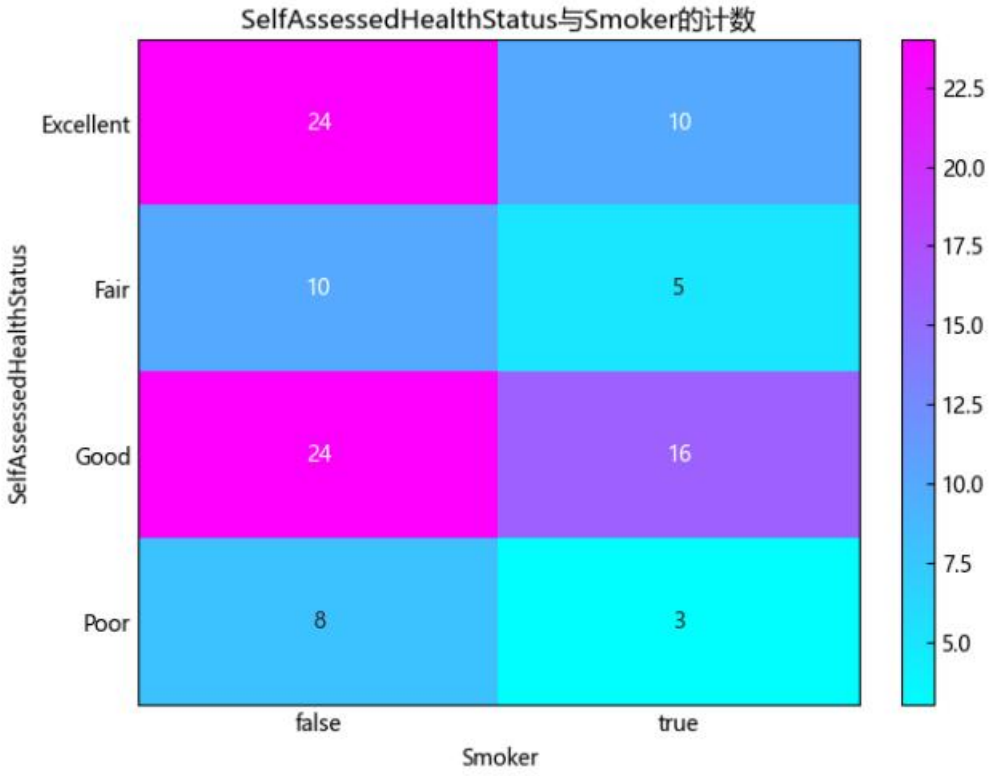
F. 示例

(1). 基于表格数据创建热图

导入 patients_data.jl 文件,基于内科病人数据表创建一个热图。

加载 patients 数据集。然后创建一个热图，计算具有一组相同 Smoker 和 SelfAssessedHealthStatus 值的患者总数。

```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/数据文件/patients_data.jl"
include(source_path)
h = heatmap(Smoker, SelfAssessedHealthStatus, xlabel = "Smoker",
ylabel = "SelfAssessedHealthStatus")
title("SelfAssessedHealthStatus 与 Smoker 的计数")
```

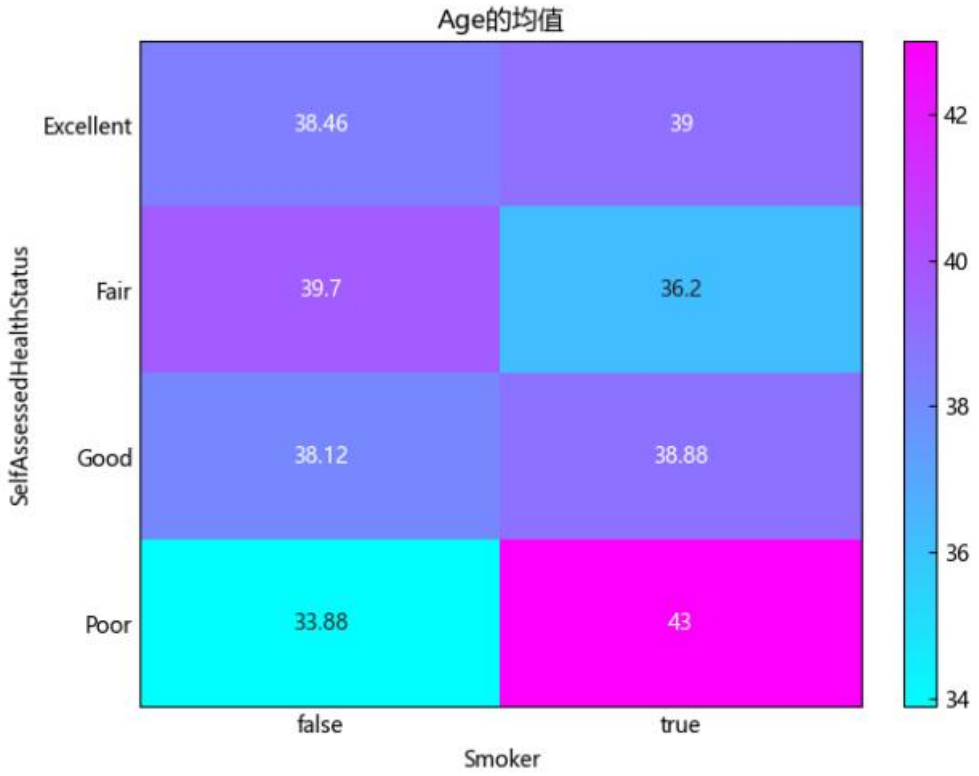


(2). 为热图颜色指定表变量

导入 patients_data.jl 文件,创建一个热图，并指定用来确定热图单元格颜色的表变量。

加载 patients 数据集，并基于这些数据创建一个热图。通过将 colorvariable 选项设置为 "Age"，使用具有一组特定 Smoker 和 SelfAssessedHealthStatus 值的患者平均年龄为每个单元格着色。

```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/数据文件/patients_data.jl"
include(source_path)
h = heatmap(Smoker, SelfAssessedHealthStatus, colorvariable = Age,
xlabel = "Smoker", ylabel = "SelfAssessedHealthStatus", fmt =
".2f")
title("Age 的均值")
```

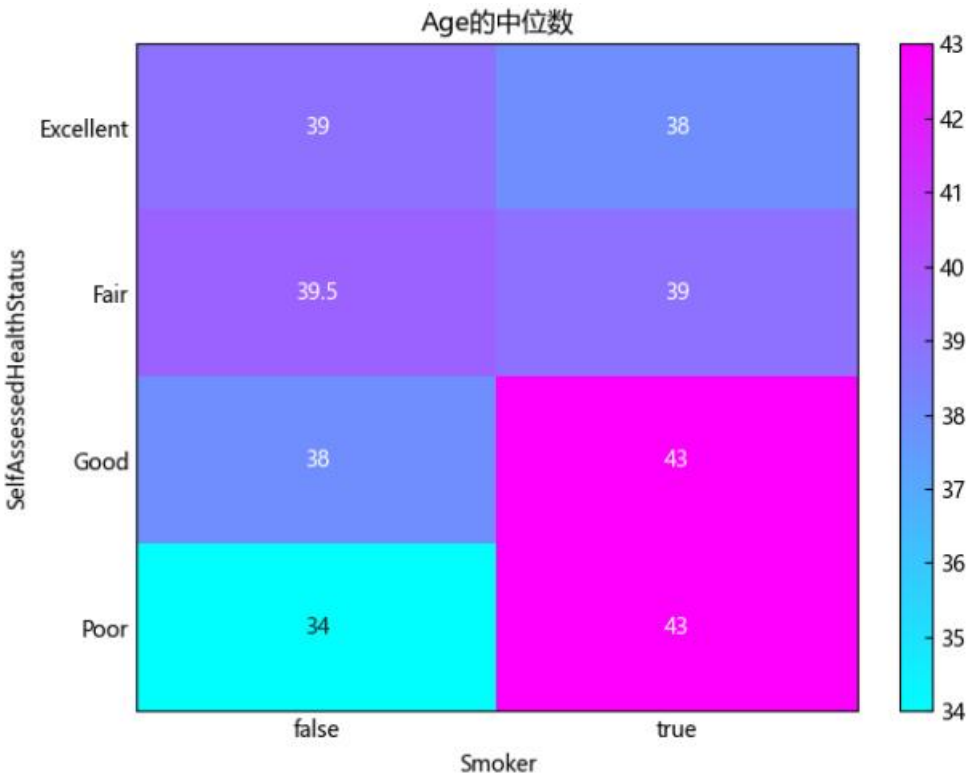


(3). 指定颜色数据的计算方法

导入 patients_data.jl 文件,创建一个热图，并指定用来确定热图单元格颜色的表变量和计算方法。

加载 patients 数据集，并基于这些数据创建一个热图。使用具有一组特定 Smoker 和 SelfAssessedHealthStatus 值的患者中位数年龄为每个单元格着色。将 colorvariable 选项指定为 "Age"，并将 colormethod 选项指定为 "median"。

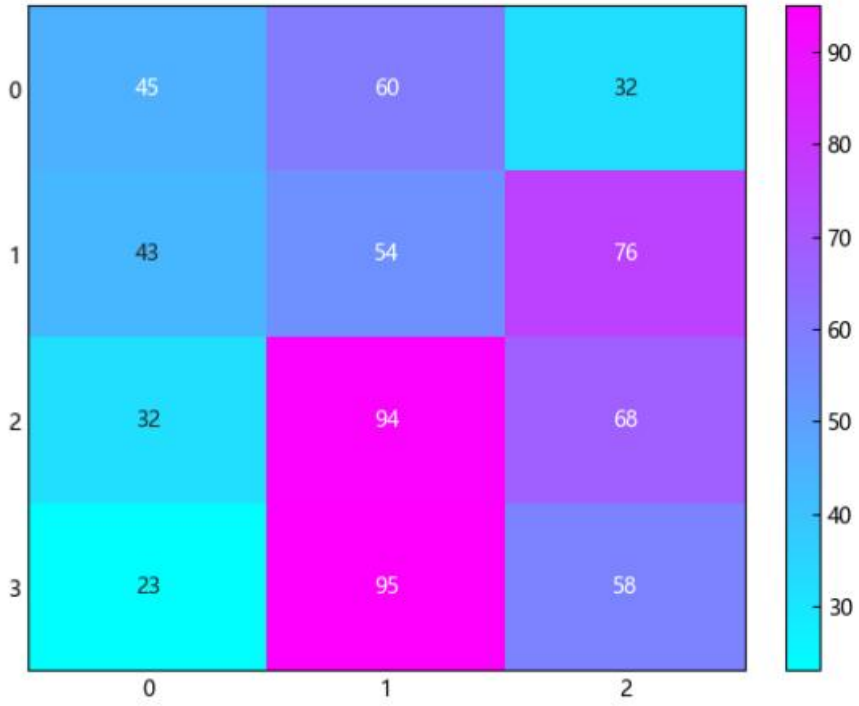
```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/数据文件/patients_data.jl"
include(source_path)
h = heatmap(Smoker, SelfAssessedHealthStatus, colorvariable = Age,
colormethod = "median", xlabel = "Smoker", ylabel =
"SelfAssessedHealthStatus", fmt = ".1f")
title("Age 的中位数")
```



(4). 基于矩阵数据创建热图

创建一个数据矩阵。然后创建矩阵值的热图。x 轴和 y 轴上的默认标签显示为 1、2、3 等。

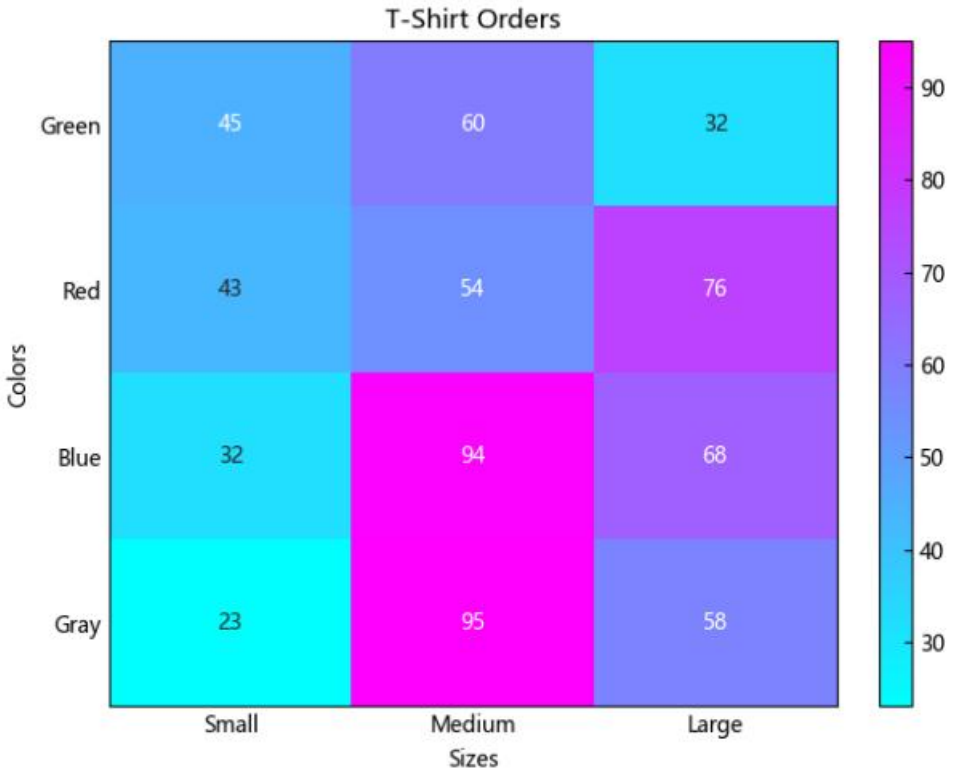
```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/数据文件/patients_data.jl"
include(source_path)
cdata = [45 60 32; 43 54 76; 32 94 68; 23 95 58];
h = heatmap(cdata)
```



(5). 使用自定义轴标签基于矩阵数据创建热图

创建一个数据矩阵。然后创建矩阵值的热图。通过将前两个输入参数指定为所需的标签，在 x 轴和 y 轴上使用自定义标签。通过设置 HeatmapChart 对象的属性，指定标题和轴标签。

```
cdata = [45 60 32; 43 54 76; 32 94 68; 23 95 58];
xvalues = ["Small", "Medium", "Large"]
yvalues = ["Green", "Red", "Blue", "Gray"]
h = heatmap(cdata, xvalues = xvalues, yvalues = yvalues)
xlabel("Sizes")
ylabel("Colors")
title("T-Shirt Orders”)
```



(6). 使用自定义聚合方法计算颜色数据

创建一个热图，并使用通过自定义聚合方法计算的数据为单元格着色。使用 accumarray 函数计算颜色数据。

根据 Month 和 Year 数据创建分类数组。然后确定用作 x 轴和 y 轴标签的唯一月份和年份。

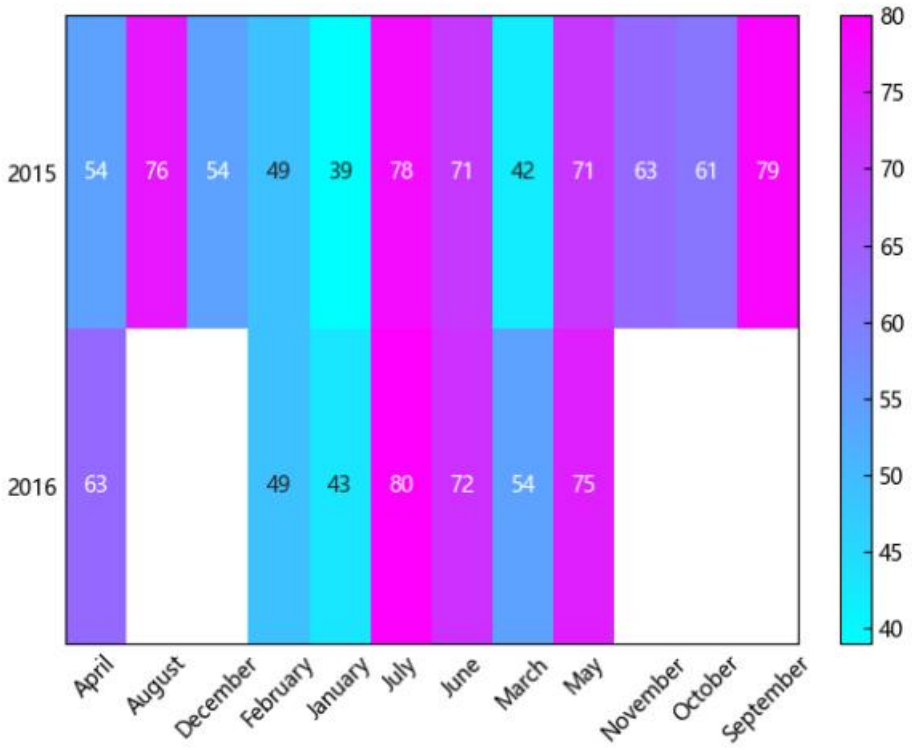
基于唯一月份和年份的数量确定生成的颜色数据的最终大小。

将分类数组 months 和 years 转换为数字索引，计算颜色数据，作为每个月份和年份组合的最高温度。使用 NaN 表示缺失的月份和年份组合。

```
xlabels = ["April", "August", "December", "February", "January",
"July", "June", "March", "May", "November", "October", "September"]
ylabels = ["2015", "2016"]
cdata = [54 76 54 49 39 78 71 42 71 63 61 79; 63 NaN NaN 49 43 80
72 54 75 NaN NaN NaN]
```

创建热图。将 x 轴和 y 轴分别标记为月份和年份。使用计算的矩阵数据为热图单元格着色。

```
h = heatmap(cdata, xvalues = xlabels, yvalues = ylabels, rotation =
45)
```



sortx

对热图列中的元素进行排序

G. 语法

```
sortx(h,row)
sortx(h,row,direction)
sortx(___,missingplacement=lcn)
sortx(h)

C = sortx(___)
```

H. 说明

sortx(h,row)按升序（从左到右）显示 row 中的元素。此函数通过重新排列各个列，对行元素进行排序。可将 row 指定为 YData 中的一个元素或由元素组成的数组。如果您指定的第一行包含重复的值，则 sortx 根据第二行重新排列其余的列，依此类推。

sortx(h,row,direction)（其中 direction 为"descend"）按降序而不是升序（默认值）来排序值。要对 row 中的每一列按不同的方向排序，请将 direction 指定

为数组，数组的元素为"ascend"或"descend"。

sortx(___,missingplacement=lcn) 指定将 NaN 元素放在排序顺序的开头还是末尾。将 lcn 指定为 "first"、"last" 或 "auto"。默认设置为 "auto"，即升序时将 NaN 元素放在最后，降序时放在开头。在前面的任何输入参数组合之后使用此选项

sortx(h) 按升序显示第一列中的元素。如果第一列包含重复的值，则 sortx 根据第二列重新排列其余的行，依此类推。

C=sortx(___)以值矩阵的形式返回排序的颜色数据，值的显示方式与在热图中相同。

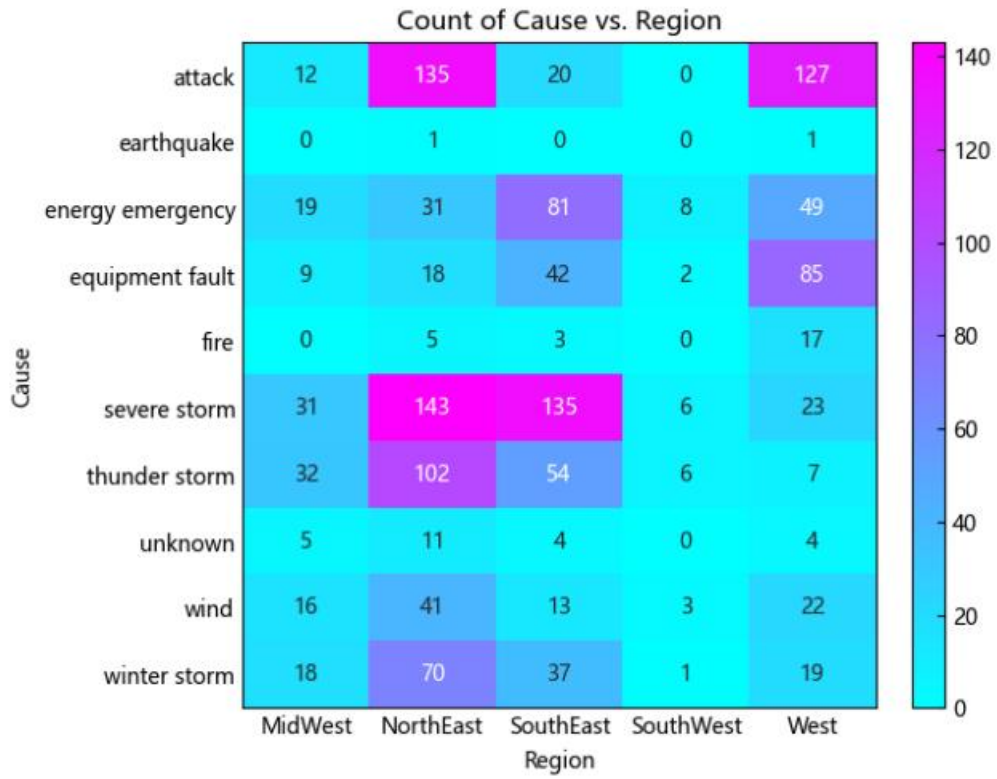
I. 示例

(1). 按升序显示热图行

创建一个电力中断热图，并对特定行中的值进行排序，使它们按升序从左到右显示。

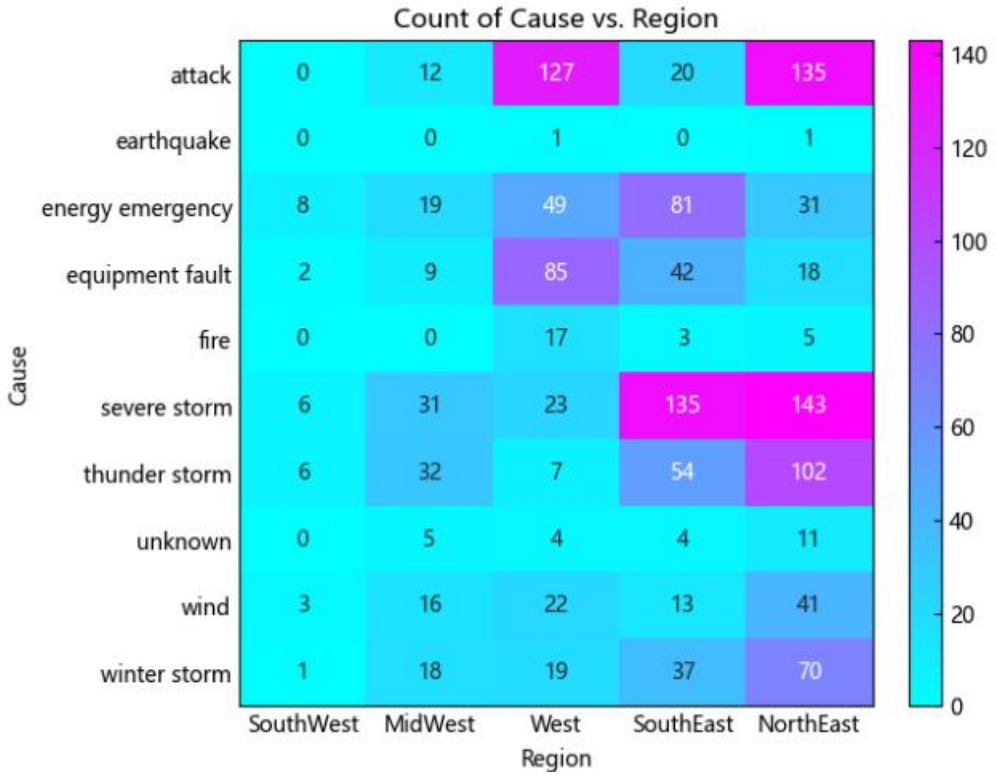
首先创建一个热图。导入 0-outages_data.jl 文件。示例文件包含表示美国电力中断情况的数据。该表包含六列：Region、OutageTime、Loss、Customers、RestorationTime 和 Cause。创建一个热图，x 轴显示不同的区域，y 轴显示不同的停电原因。在每个单元格中，显示每个区域由于特定原因经历停电的次数。

```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/格式和注释/sortx/0-outages_data.jl"
include(source_path)
h = heatmap(Region, Cause, xlabel = "Region", ylabel = "Cause", fmt = ".2f")
title("Count of Cause vs. Region")
```



对 "winter storm" 行中的值进行排序，使它们按升序从左到右显示。

```
sortx(h[1],"winter storm")
```



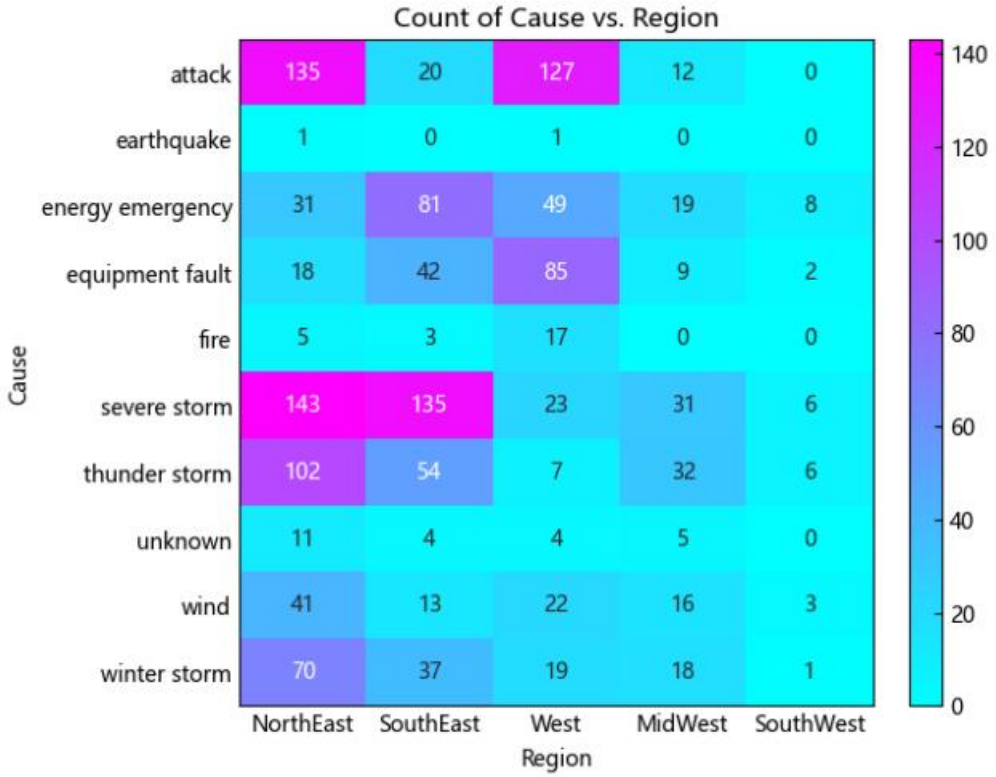
(2). 按降序显示热图行

创建一个电力中断热图，并通过重新排列各列，按降序显示 "winter storm"

行中的值。

```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/格式和注释/sortx/0-outages_data.jl"
include(source_path)
h = heatmap(Region, Cause, xlabel = "Region", ylabel = "Cause", fmt = ".2f")
title("Count of Cause vs. Region")

sortx(h[1],"winter storm", "descend")
```



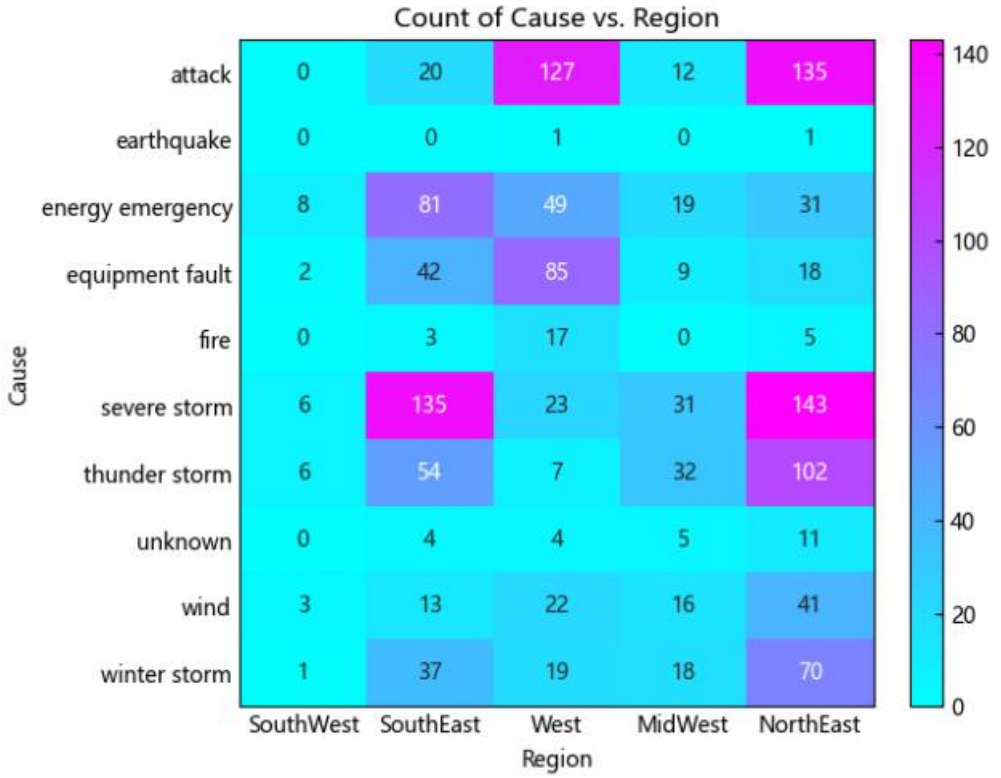
(3). 基于多行重新排列热图的列

创建一个电力中断热图，并使用多个行中的值重新排列各列。对列进行排序，使 "unknown" 行按升序显示。由于该行包含重复的值，因此使用"earthquake"行对其余的列进行排序。

```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/格式和注释/sortx/0-outages_data.jl"
include(source_path)
h = heatmap(Region, Cause, xlabel = "Region", ylabel = "Cause", fmt
```

```
= ".2f")
title("Count of Cause vs. Region")

sortx(h[1],["unknown" "earthquake"])
```

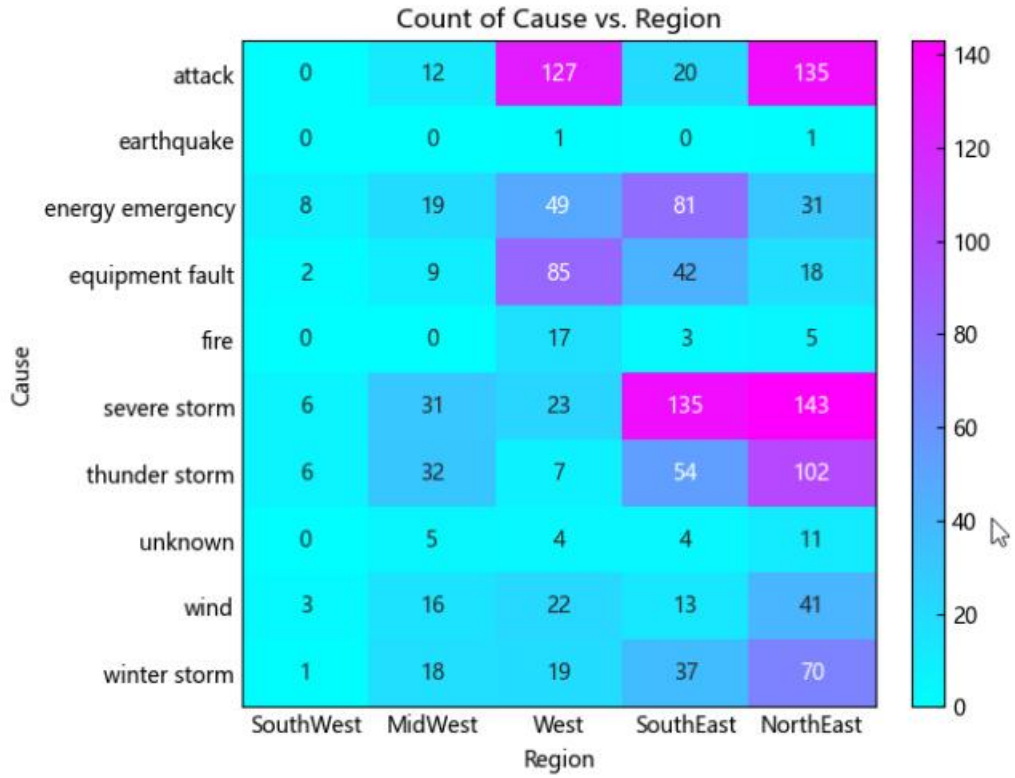


(4). 返回重新排列后的热图列值

创建一个电力中断热图，并对 "winter storm" 行中的值进行排序，使它们按升序显示。按照 x 轴上的显示顺序返回排序的颜色数据和列值（x 值）。

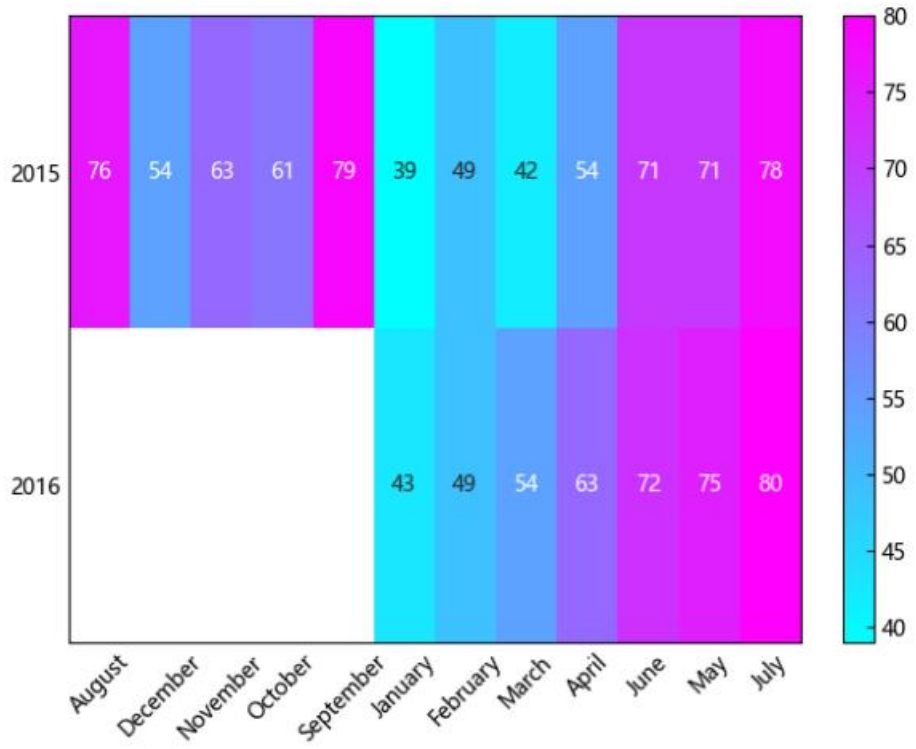
```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/格式和注释/sortx/0-outages_data.jl"
include(source_path)
h = heatmap(Region, Cause, xlabel = "Region", ylabel = "Cause", fmt = ".2f")
title("Count of Cause vs. Region")

C,x = sortx(h[1], "winter storm")
```



(5). 带空值的热图排序

```
xlabels = ["April", "August", "December", "February", "January",
"July", "June", "March", "May", "November", "October", "September"]
ylabels = ["2015", "2016"]
cdata = [54 76 54 49 39 78 71 42 71 63 61 79; 63 NaN NaN 49 43 80
72 54 75 NaN NaN NaN]
h = heatmap(cdata, xvalues = xlabels, yvalues = ylabels, rotation =
45)
sortx(h[1], "2016", missingplacement = "first")
```



sorty

对热图列中的元素进行排序

J. 语法

```
sorty(h,column)
sorty(h,column,direction)
sorty(___,missingplacement=lcn)
sorty(h)

C = sorty(___)
```

K. 说明

sorty(h,column)按升序（从上到下）显示 column 中的元素。此函数通过重新排列各个行，对列元素进行排序。可将 column 指定为 XData 中的一个元素或由元素组成的数组。如果您指定的第一列包含重复的值，则 sorty 根据第二列重新排列其余的行，依此类推。

sorty(h,column,direction)（其中 direction 为"descend"）按降序而不是升序（默认值）来排序值。要对 column 中的每一列按不同的方向排序，请将

direction 指定为数组，数组的元素为"ascend"或"descend"。

sorty(___,missingplacement=lcn) 指定将 NaN 元素放在排序顺序的开头还是末尾。将 lcn 指定为 "first"、"last" 或 "auto"。默认设置为 "auto"，即升序时将 NaN 元素放在最后，降序时放在开头。在前面的任何输入参数组合之后使用此选项。

sorty(h) 按升序显示第一列中的元素。如果第一列包含重复的值，则 sorty 根据第二列重新排列其余的行，依此类推。

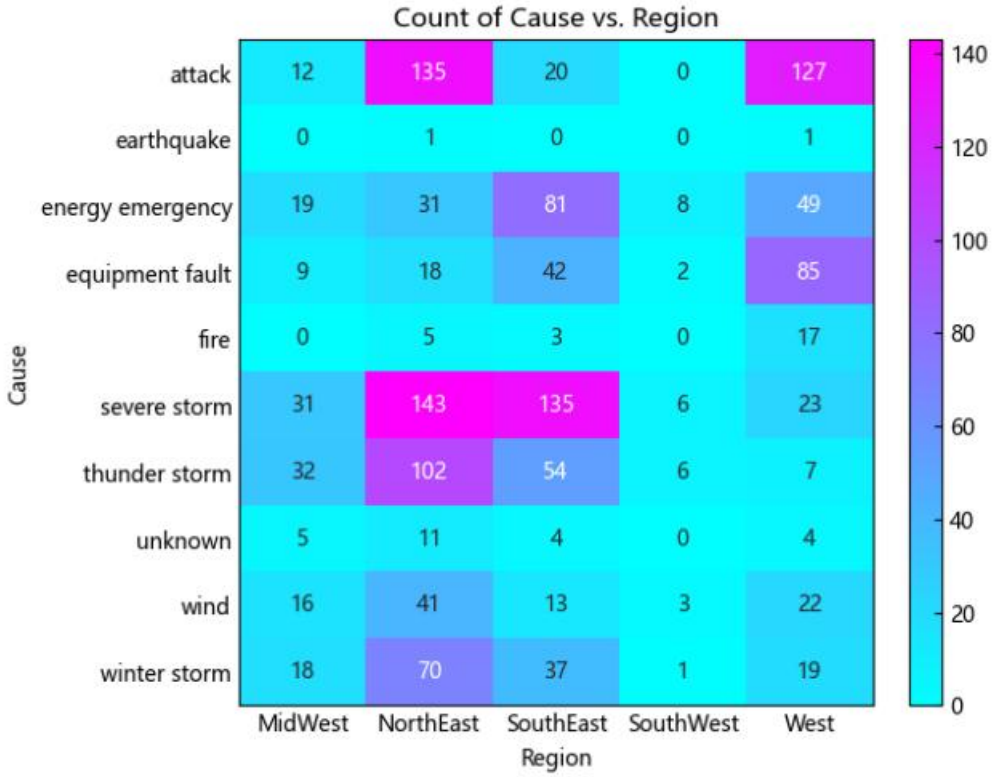
C = sorty(___) 以值矩阵的形式返回排序的颜色数据，值的显示方式与在热图中相同。

L. 示例

(1). 按升序显示热图列

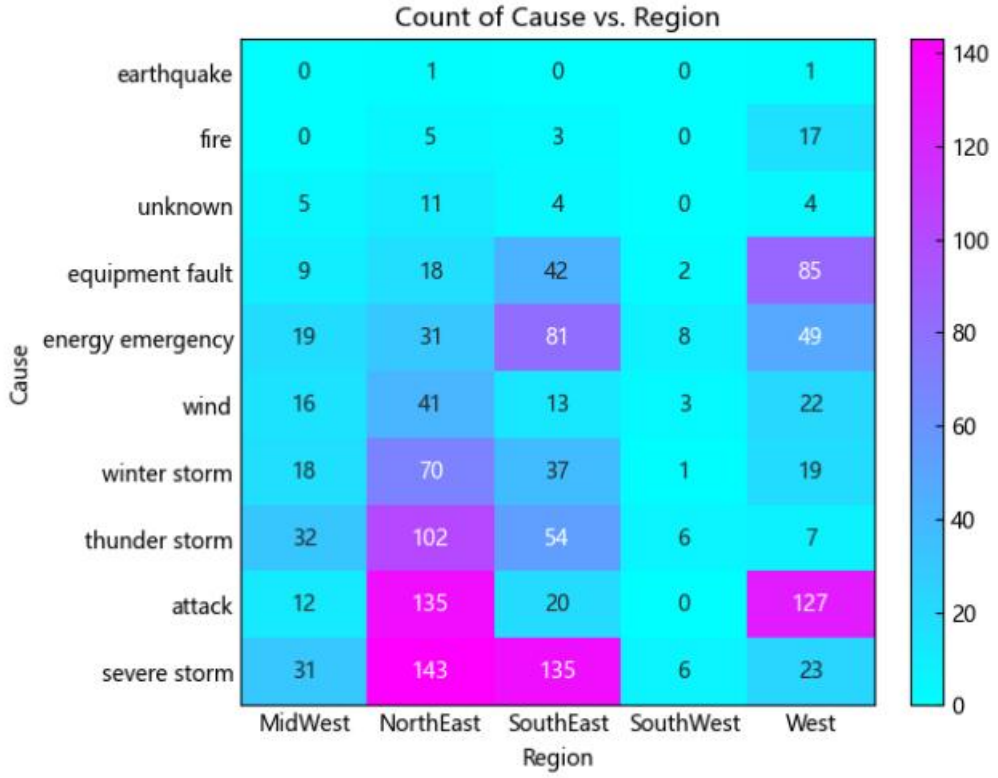
导入 0-outages_data.jl 文件。

```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/格式和注释/sorty/0-outages_data.jl"
include(source_path)
h = heatmap(Region, Cause, xlabel = "Region", ylabel = "Cause", fmt = ".2f")
title("Count of Cause vs. Region")
```



对 "NorthEast" 列中的值进行排序，使它们从上到下按升序显示。

```
sorty(h[1],"NorthEast")
```

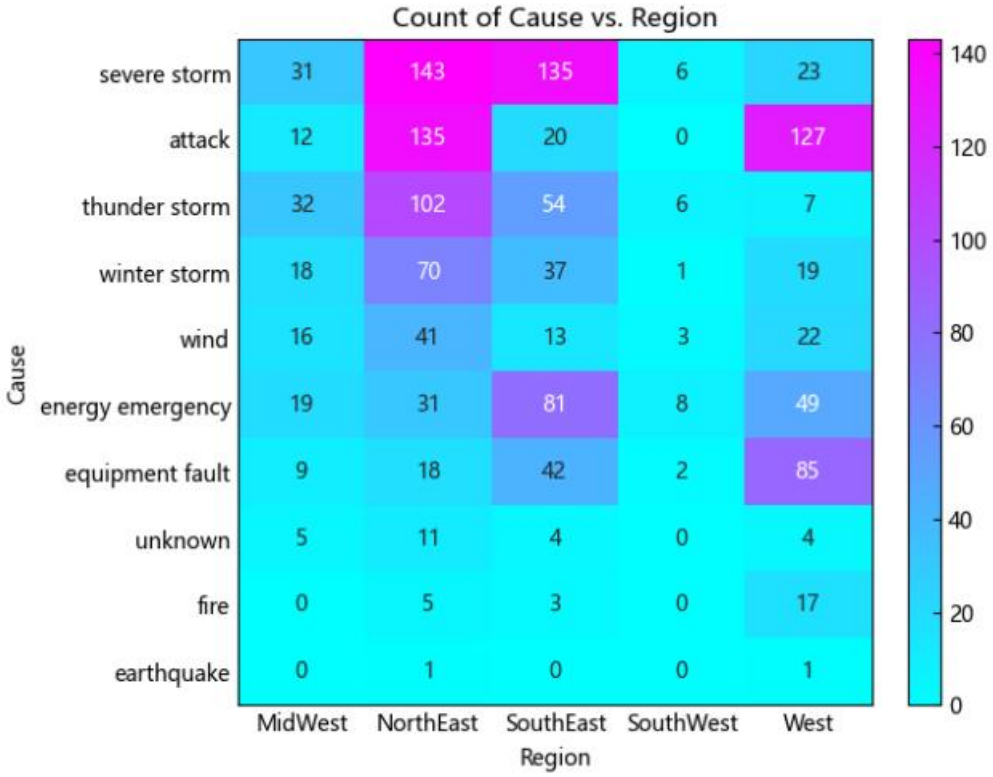


(2). 按降序显示热图列

创建一个电力中断热图，并通过重新排列各行，按降序显示 "NorthEast" 列中的值。

```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/格式和注释/sorty/0-outages_data.jl"
include(source_path)
h = heatmap(Region, Cause, xlabel = "Region", ylabel = "Cause", fmt = ".2f")
title("Count of Cause vs. Region")

sorty(h[1],"NorthEast", "descend")
```



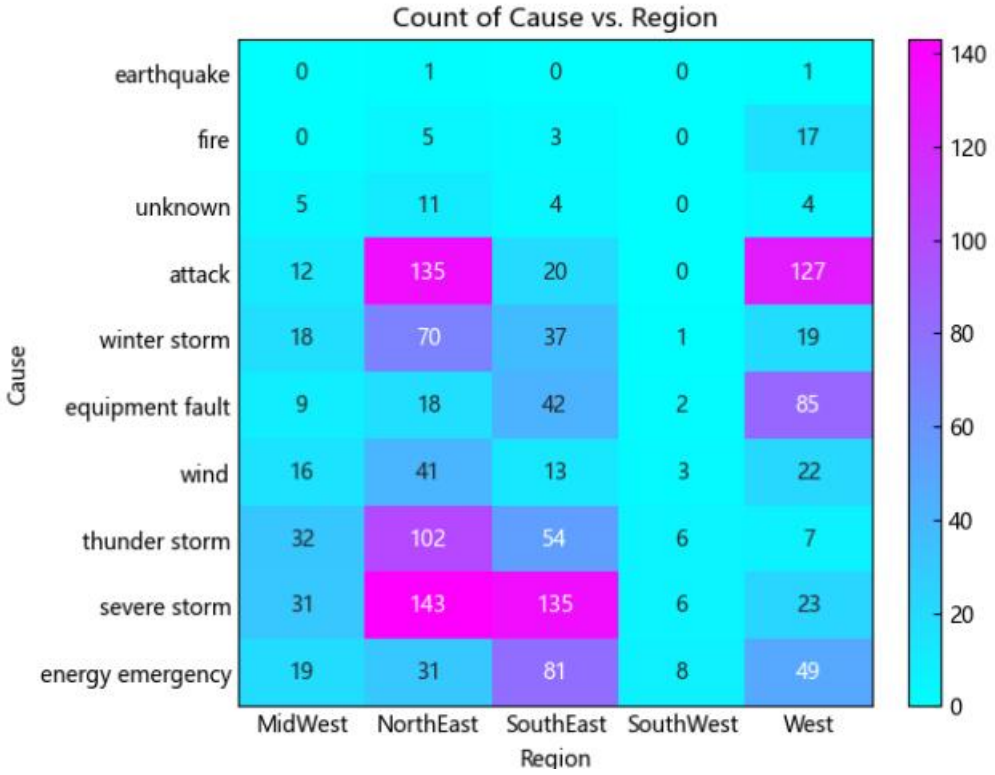
(3). 基于多列重新排列热图的行

创建一个电力中断热图，并使用多个列中的值重新排列各行。对行进行排序，使 "SouthWest" 列按升序显示。由于该列包含重复的值，因此使用 "NorthEast" 列对其余的行进行排序。

```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/格式和注释/sorty/0-outages_data.jl"
include(source_path)
```

```
h = heatmap(Region, Cause, xlabel = "Region", ylabel = "Cause", fmt
= ".2f")
title("Count of Cause vs. Region")

sorty(h[1],["SouthWest" "NorthEast"])
```

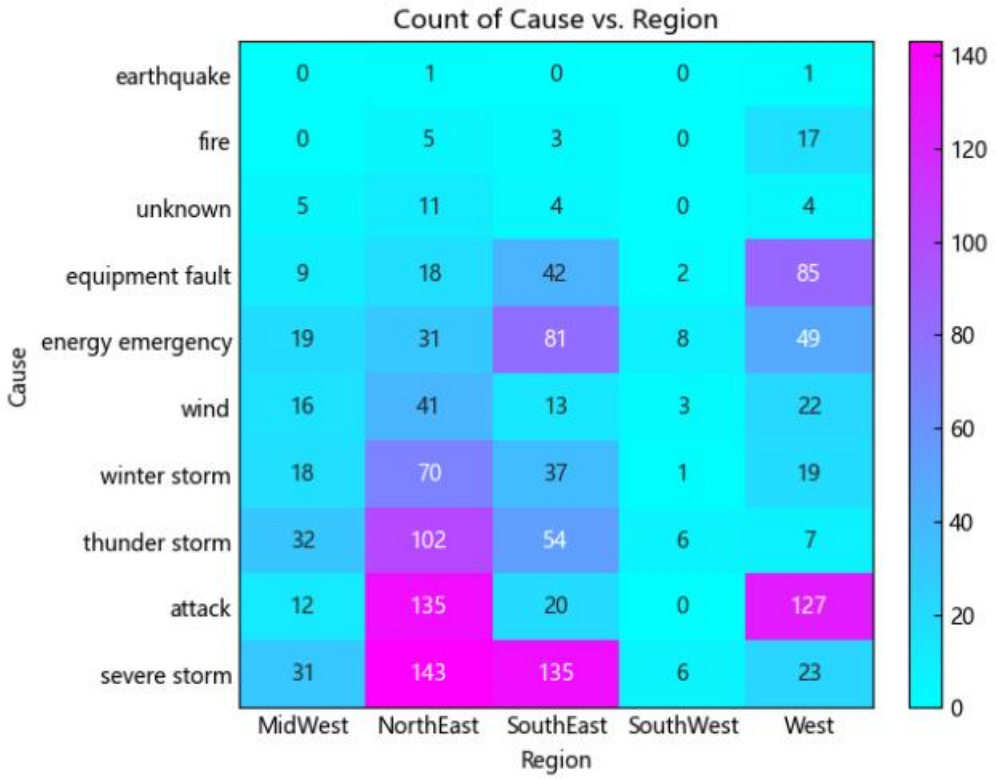


(4). 返回重新排列后的热图行值

创建一个电力中断热图，并对 "NorthEast" 列中的值进行排序，使它们按升序从上到下显示。按照 y 轴上的显示顺序返回排序的颜色数据和行值（y 值）。

```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/格式和注释/sorty/0-
outages_data.jl"
include(source_path)
h = heatmap(Region, Cause, xlabel = "Region", ylabel = "Cause", fmt
= ".2f")
title("Count of Cause vs. Region")

C,y = sorty(h[1], "NorthEast")
```



wordcloud

使用文本数据创建文字云图

M. 语法

```
wordcloud(words,sizeData)
wordcloud(C)

wordcloud(___,Key=Value)

wordcloud(ax,___)

wc = wordcloud(___)
```

N. 说明

wordcloud(words,sizeData)使用 words 的元素（单词大小由 SizeData 指定）创建文字云图。

wordcloud(C)根据字符串数组 C 的唯一元素创建文字云图，大小与这些元素的频率计数对应。C 可以是字符串数组、字符向量。

wordcloud(___,Key=Value)使用一个或多个名称-值对组参数指定其他

WordCloudChart 属性。

wordcloud(ax,___) 在由 ax 指定的坐标区上创建文字云。

wc = wordcloud(___) 返回 WordCloudChart 对象。创建文字云后，使用 wc 修改其属性。有关属性列表，请参阅 WordCloudChart 属性。

0. 示例

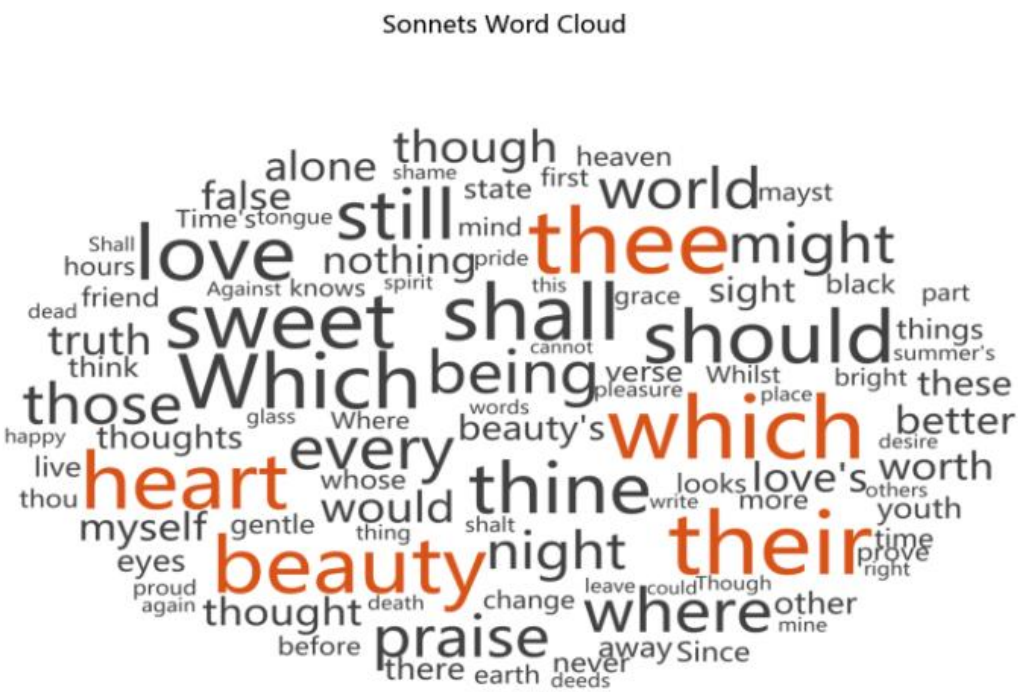
(1). 使用表创建文字云

导入 0-data.jl 文件。

```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/wordcloud/0-data.jl"
include(source_path)
```

使用 wordcloud 绘制表数据。将单词和相应的单词大小分别指定为 Word 和 Count 变量。

```
wordcloud(Word, Count)
title("Sonnets Word Cloud")
```



(2). 准备文本数据以创建文字云

此示例说明如何通过将纯文本读入字符串数组、进行预处理并传递给 wordcloud 函数，使用纯文本创建文字云。

使用 read 函数从莎士比亚的十四行诗中读取文本并将其转换为字符串。

```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/wordcloud/sonnets.txt"
sonnets = read(source_path, String);
index = findfirst("II", sonnets);
SubString(sonnets, 1, index[1]-1);
```

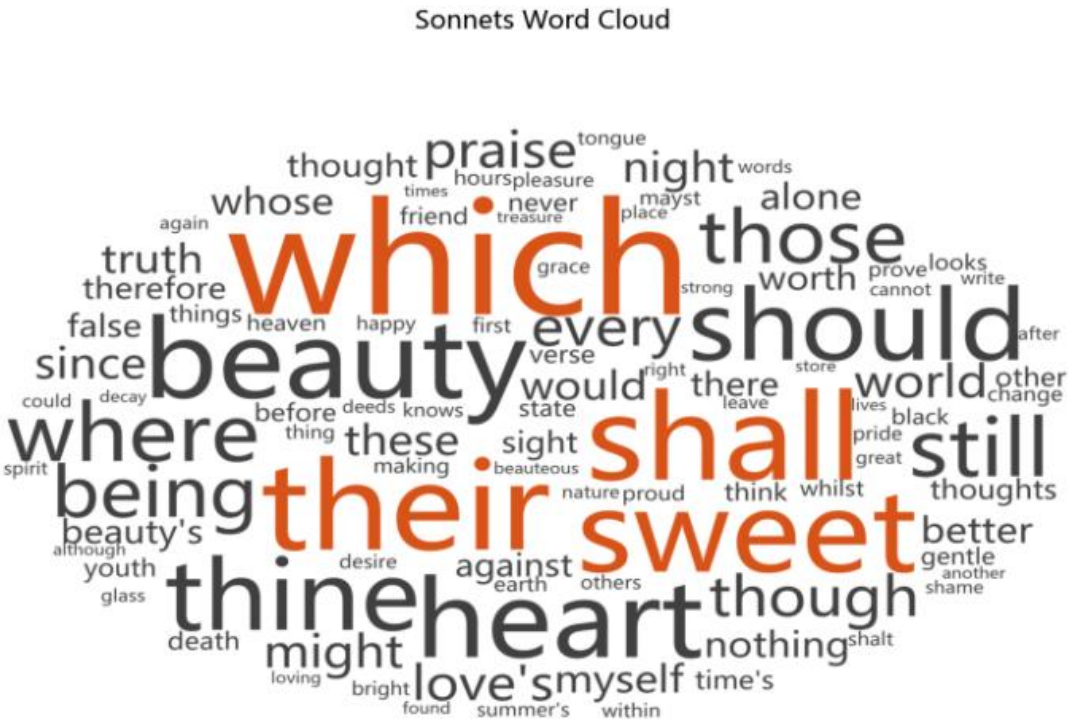
将 sonnets 拆分为其元素包含单个单词的字符串数组。要完成此操作，需要删除所有标点字符，将所有字符串元素合并成一个 1×1 字符串，然后在空白字符处进行拆分。然后，删除少于五个字符的单词并将单词转换为小写。

```
punctuationCharacters = ['.', '?', '!', ',', ';', ':'];
sonnets = map(x->(if (x in punctuationCharacters) ' ' else x
end),sonnets);
words = split(join(sonnets));

new_words = Base.filter(x->(length(x)>=5), words)
new_words = lowercase.(new_words);
new_words[1:10]
```

将 sonnets 转换为分类数组，然后使用 wordcloud 进行绘图。此函数绘制 C 的唯一元素，大小与这些元素的频率计数对应。

```
wordcloud(new_words)
title("Sonnets Word Cloud")
```



(3). 指定单词大小

通过将纯文本读入一个字符串数组，对其进行预处理并传递给 wordcloud 函数，即可从纯文本创文字云。

使用 read 函数从莎士比亚的十四行诗中读取文本并将其转换为字符串。

```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/wordcloud/sonnets.txt"
sonnets = read(source_path, String);
index = findfirst("II", sonnets);
SubString(sonnets, 1, index[1]-1);
```

将 sonnets 拆分为其元素包含单个单词的字符串数组。要完成此操作，需要删除所有标点字符，将所有字符串元素合并成一个 1×1 字符串，然后在空白字符处进行拆分。然后，删除少于五个字符的单词并将单词转换为小写。

```
punctuationCharacters = ['.', '?', '!', ',', ';', ':'];
sonnets = map(x->(if (x in punctuationCharacters) ' ' else x
end),sonnets);
words = split(join(sonnets));

new_words = Base.filter(x->(length(x)>=5), words)
new_words = lowercase.(new_words);
new_words[1:10]
```

查找 sonnets 中的唯一单词并计算它们出现的频率。使用频率计数作为大小数据创建文字云。

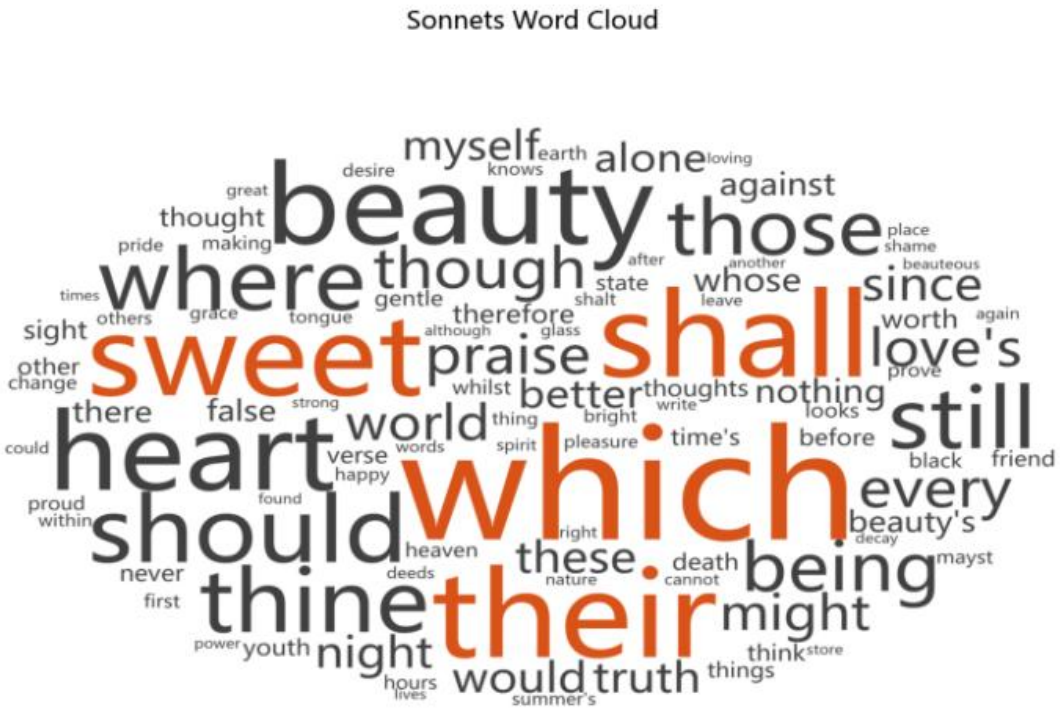
```
function histcounts1(words)
    dc = Dict()
    for w in words
        if !haskey(dc, w)
            dc[w] = 1
        else
            dc[w] += 1
        end
    end

    unique_names = collect(keys(dc))
    sort!(unique_names)
    num_occ = map(x -> dc[x], unique_names)

    return num_occ, unique_names
```

```
end


numOccurrences,uniqueWords = histcounts1(new_words);
figure()
wordcloud(uniqueWords,numOccurrences);
title("Sonnets Word Cloud")
```



(4). 指定单词颜色

导入 0-data.jl 文件。

```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/wordcloud/0-data.jl"
include(source_path)
```

使用 wordcloud 绘制表数据。将单词和相应的单词大小分别指定为 Word 和 Count 变量。要将单词颜色设置为随机值，请将 color 设置为随机矩阵或 RGB 三元组，每一行对应一个单词。

```
wordcloud(Word, Count, color = colors)
title("Sonnets Word Cloud")
```



(5). 使用 Text Analytics Toolbox 创建文字云

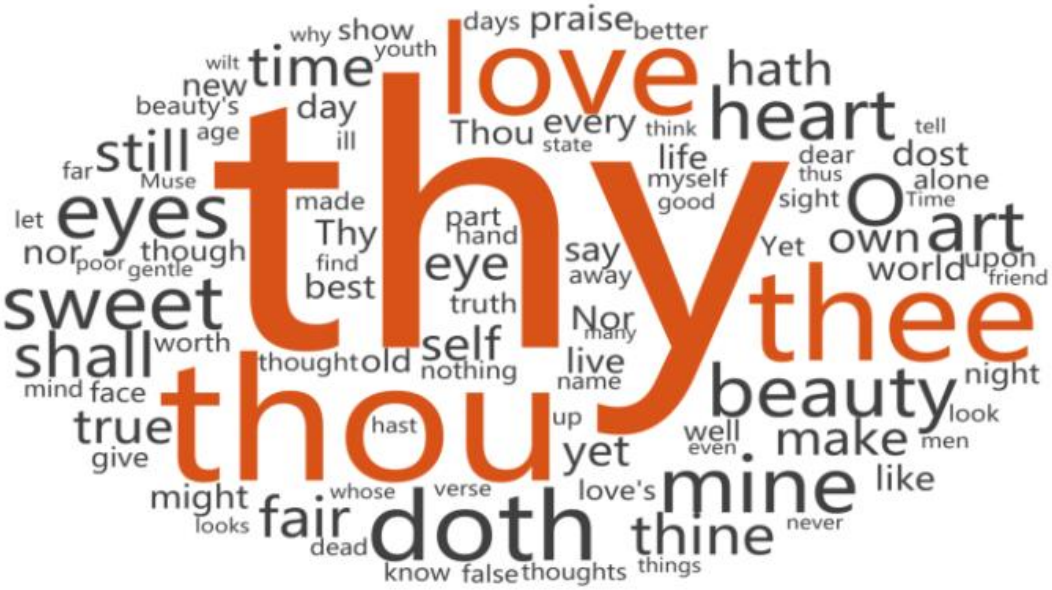
通过将纯文本读入一个字符串数组，对其进行预处理并传递给 wordcloud 函数，即可从纯文本创文字云。

使用 read 函数从莎士比亚的十四行诗中读取文本并将其转换为字符串。

```
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/wordcloud/sonnets.txt"
str = read(source_path, String);
index = findfirst("II", str);
```

在文字云中显示十四行诗中的单词。

```
figure();
wordcloud(str)
```



3.3.1.3 离散数据图

条形图、散点图等

使用条形图或针状图等将离散数据可视化。例如，可以创建垂直或水平条形图，其中条形长度与它们所代表的值成比例。

3.3.1.3.1 条形图

函数名	简介
bar	条形图
barh	水平条形图
bar3	绘制三维条形图
bar3h	绘制水平三维条形图
pareto	帕累托图

bar

条形图

A. 语法

```
bar(y)
bar(x,y)

bar(___,Key=Value)

bar(ax,___)

b = bar(___)
```

B. 说明

bar(y)创建一个条形图，y 中的每个元素对应一个条形。如果 y 是 m×n 矩阵，则 bar 创建每组包含 n 个条形的 m 个组。

bar(x,y)在 x 指定的位置绘制条形。

bar(___,Key=Value)使用一个或多个名称-值对组参数指定条形图的属性。仅使用默认"grouped"或"stacked"样式的条形图支持设置条形属性。在所有其他输入参数之后指定名称-值对组参数。

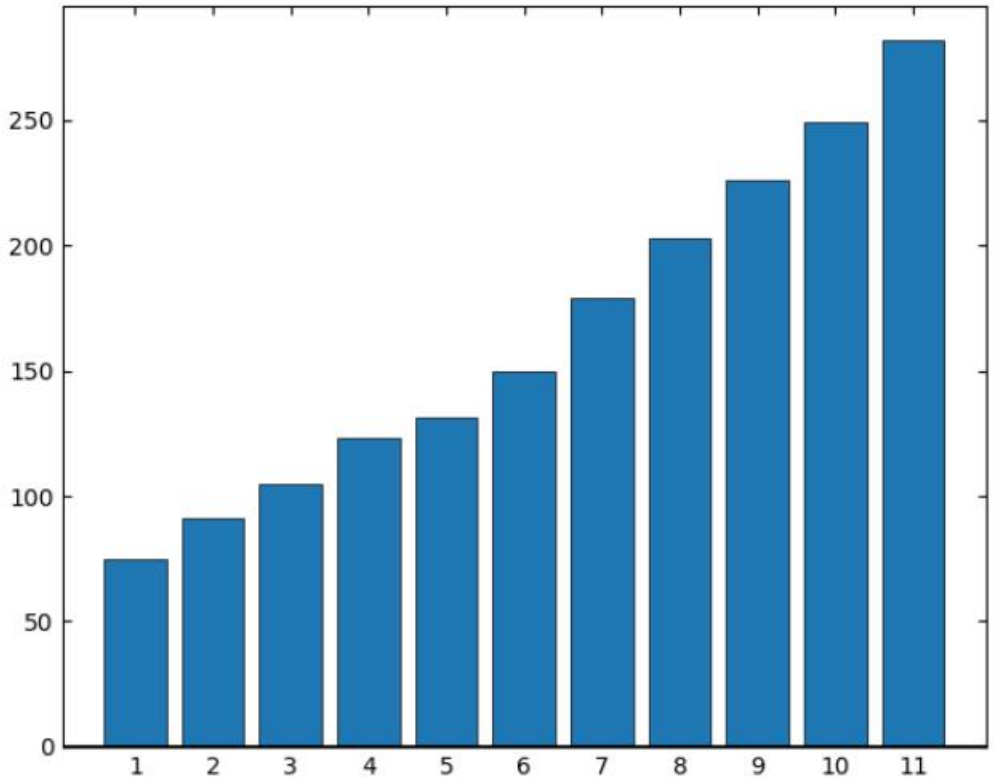
bar(ax,___)将图形绘制到 ax 指定的坐标区中，而不是当前坐标区(gca)中。选项 ax 可以位于前面的语法中的任何输入参数组合之前。

b=bar(___)返回一个或多个 Bar 对象。如果 y 是向量，则 bar 将创建一个 Bar 对象。如果 y 是矩阵，则 bar 为每个序列返回一个 Bar 对象。显示条形图后，使用 b 设置条形的属性。

C. 示例

(1). 创建条形图

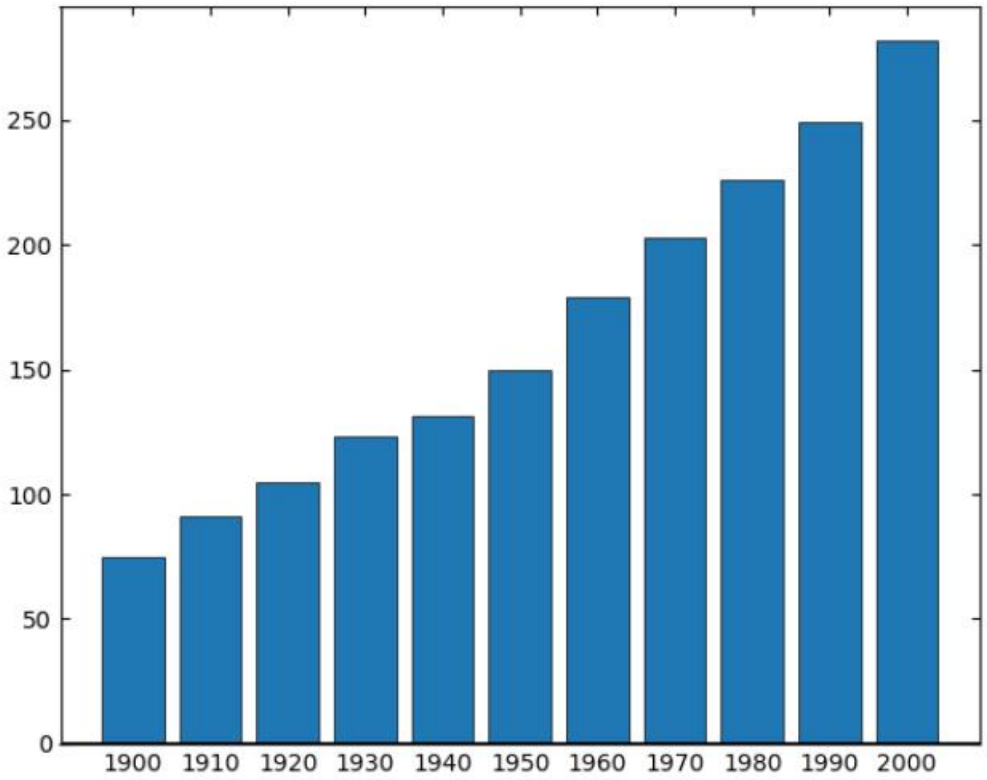
```
y = [75 91 105 123.5 131 150 179 203 226 249 281.5]
bar(y)
```



(2). 指定条形位置

指定沿 x 轴的条形位置。

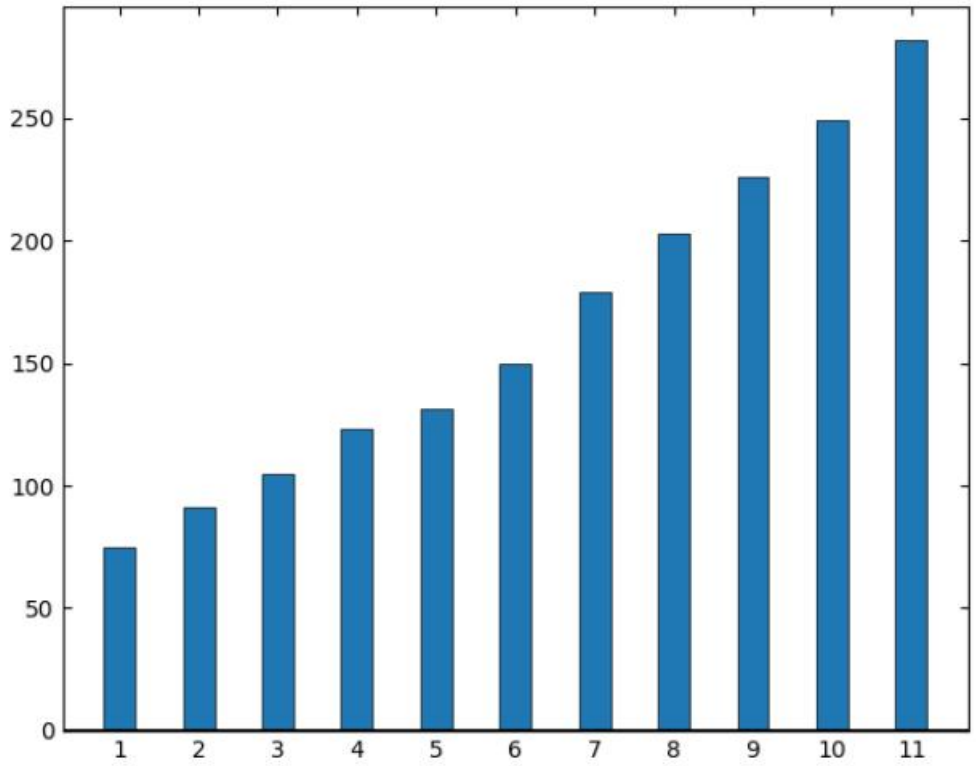
```
x = 1900:10:2000
y = [75, 91, 105, 123.5, 131, 150, 179, 203, 226, 249, 281.5]
bar(x,y)
```



(3). 指定条形宽度

将各条形的宽度设置为各条形可用总空间的 40%。

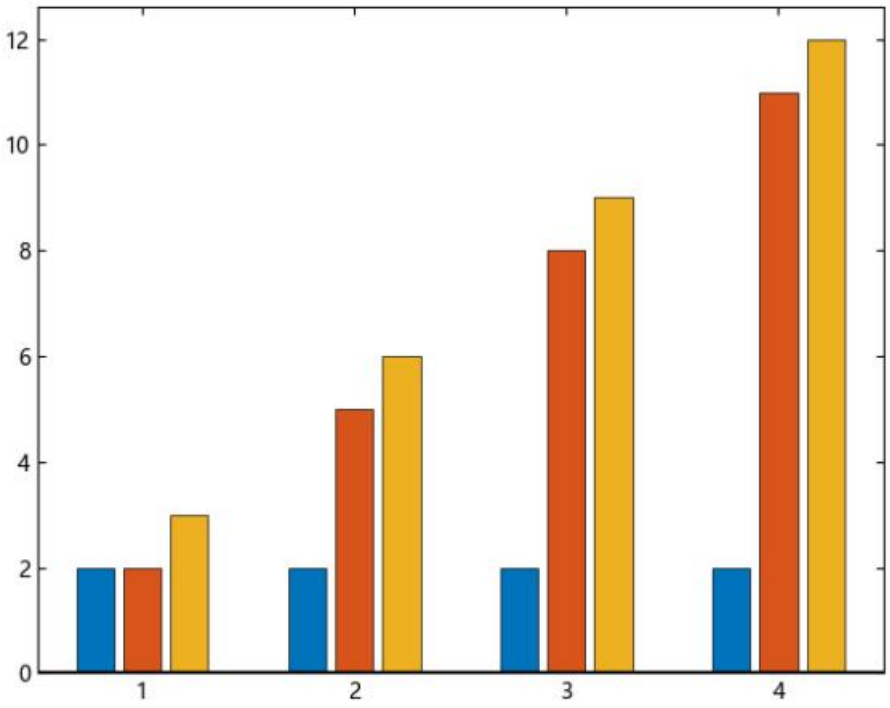
```
y = [75, 91, 105, 123.5, 131, 150, 179, 203, 226, 249, 281.5]
bar(y, width = 0.4)
```



(4). 显示条形组

显示四个条形组，每一组包含三个条形。

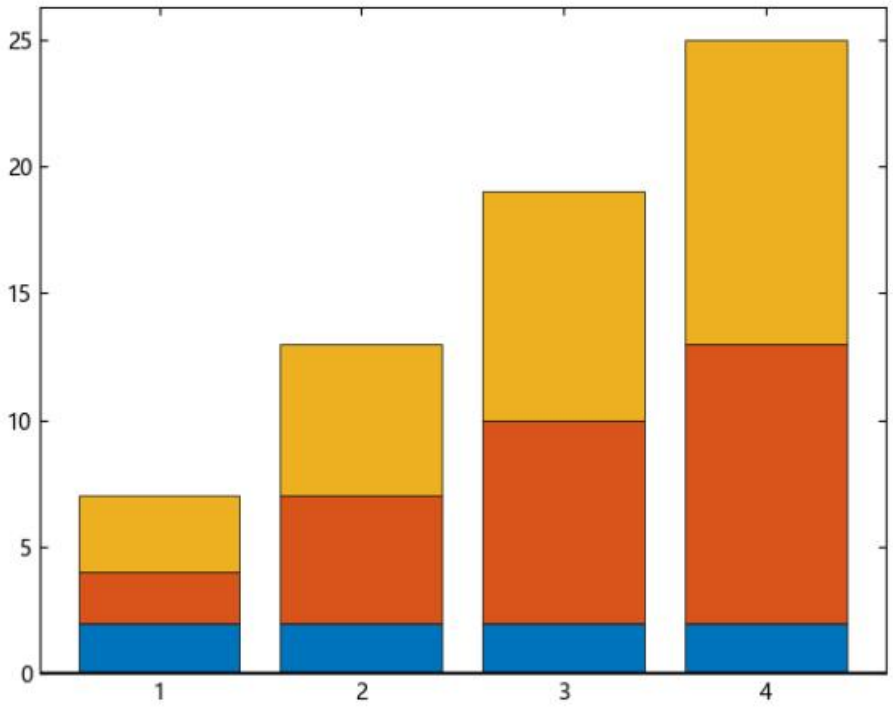
```
y = [2 2 3; 2 5 6; 2 8 9; 2 11 12]
bar(y)
```



(5). 显示堆叠条形图

为矩阵中的每一行显示一个条形。每个条形的高度是行中各元素之和。

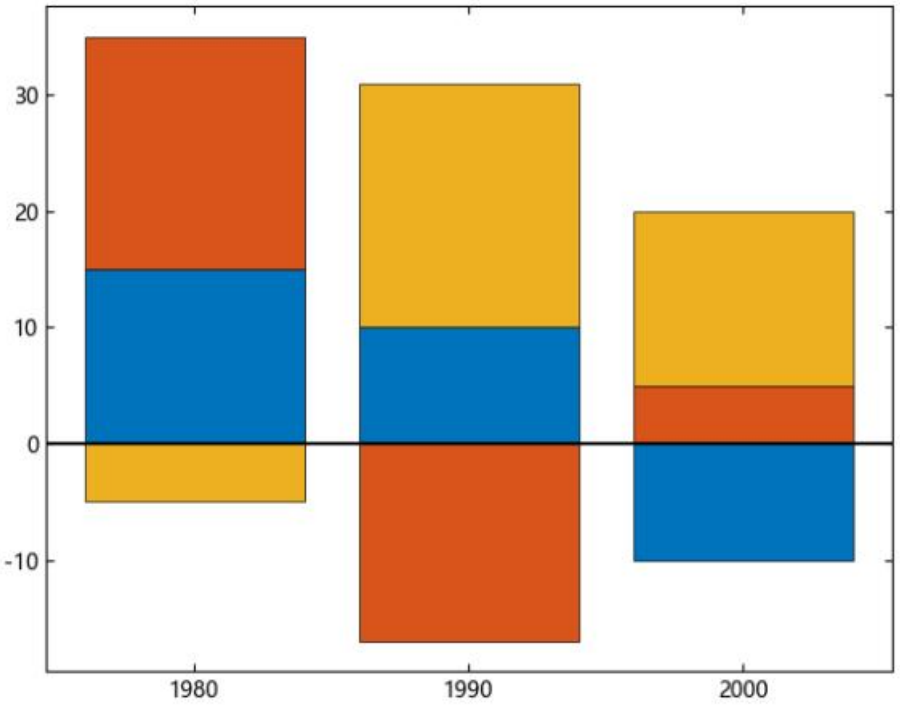
```
y = [2 2 3; 2 5 6; 2 8 9; 2 11 12]
bar(y, style = "stacked")
```



(6). 显示具有负数据的堆叠条形

将 x 定义为一个包含三个年份值的向量。将 y 定义为包含负值和正值组合的矩阵。在条形图中显示这些值。

```
x = [1980, 1990, 2000]
y = [15 20 -5; 10 -17 21; -10 5 15]
bar(x, y, style = "stacked")
```

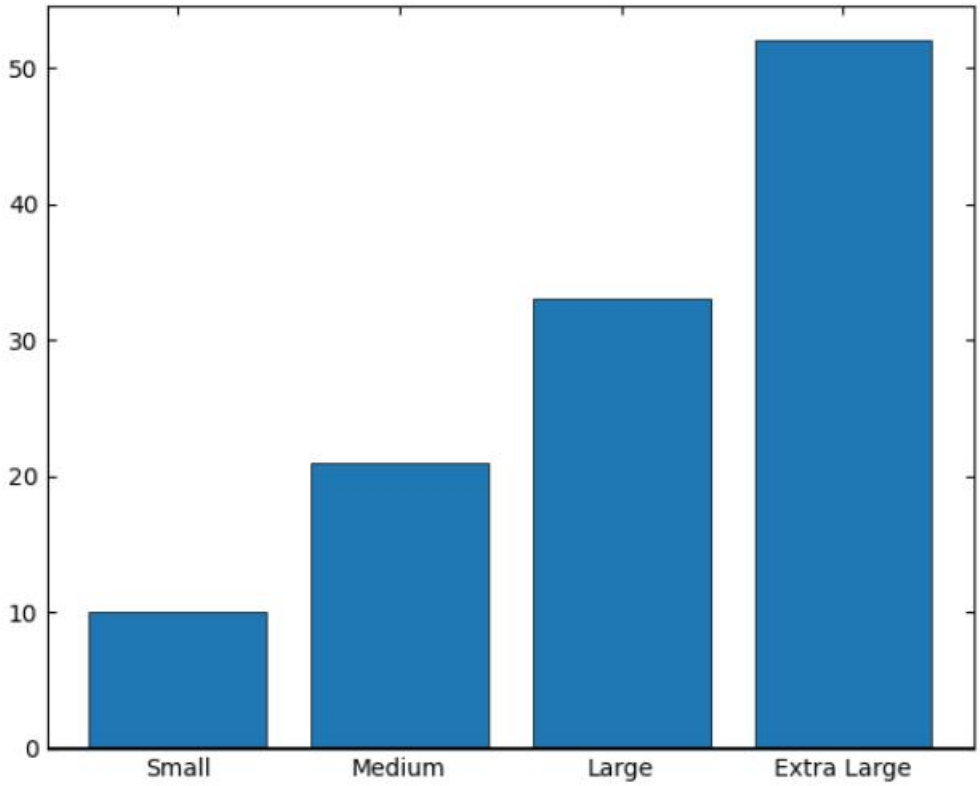


(7). 指定分类数据

指示条形图类别的一种方法是将 X 指定为分类数组。bar 函数使用经过排序的类别列表，因此条形的显示顺序可能与您预期的有所不同。

将 X 定义为分类数组，然后将 Y 定义为条形高度向量，并显示条形图。

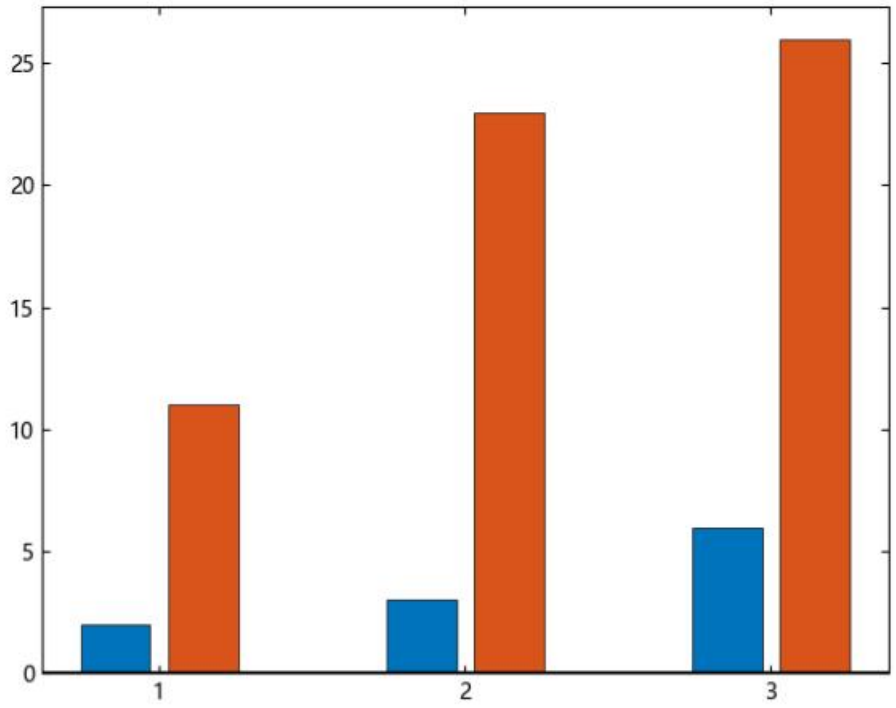
```
x = ["Small","Medium","Large","Extra Large"]
y = [10, 21, 33, 52]
bar(x, y)
```



(8). 在条形末端指定标签

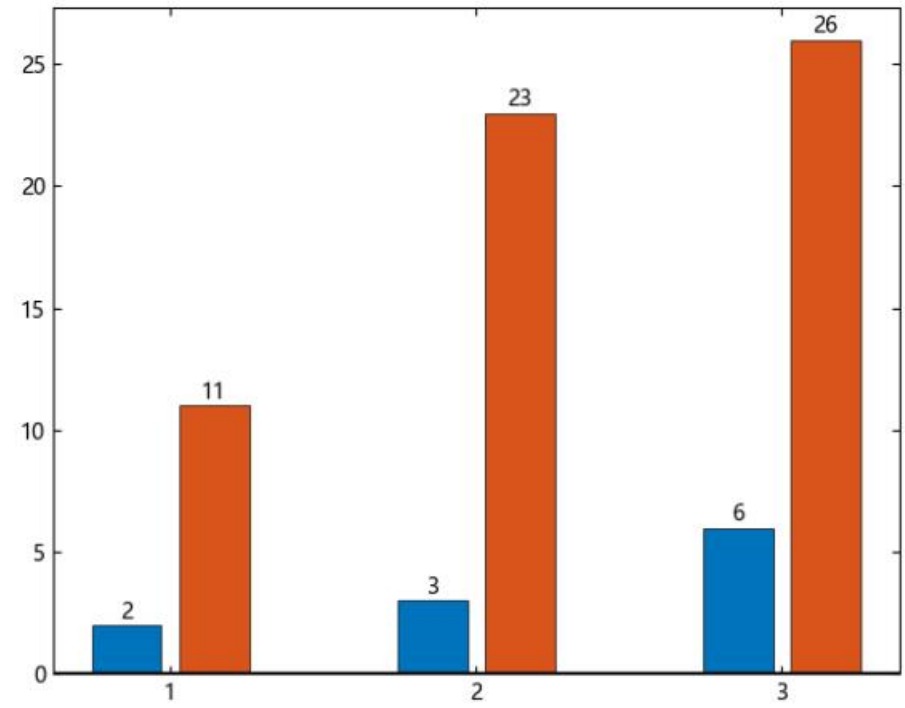
将 vals 定义为一个包含两个数据集的值的矩阵。在条形图中显示值，并指定输出参数。由于有两个数据集，bar 返回包含两个 Bar 对象的向量。

```
x = [1, 2, 3]
vals = [2 11; 3 23; 6 26]
b = bar(x,vals)
```



接下来，在条形序列的末端上方显示值。

```
x = [1, 2, 3]
vals = [2 11; 3 23; 6 26]
b = bar(x,vals, text = vals)
```



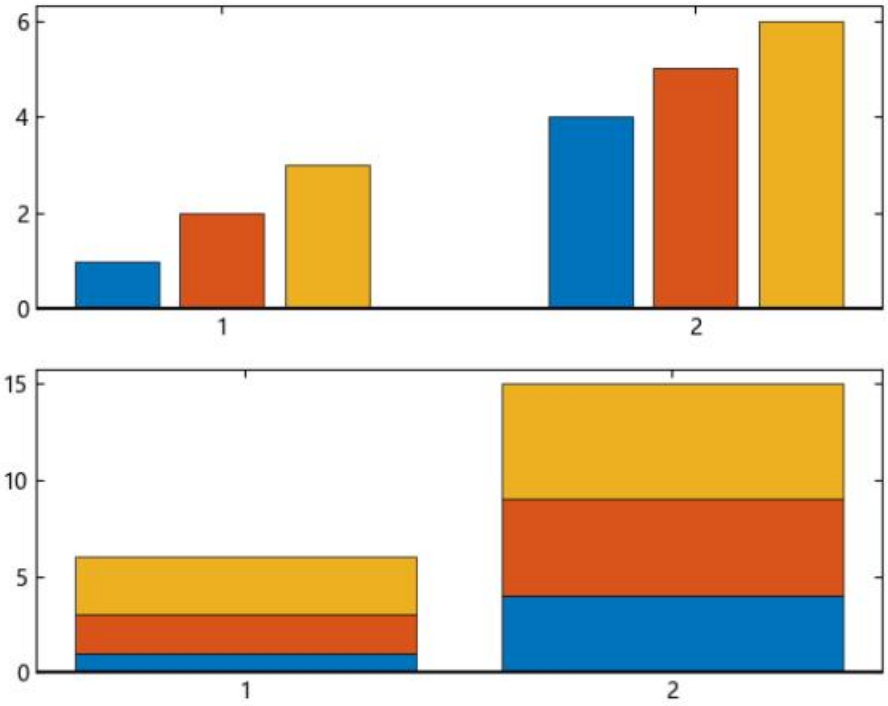
(9). 为条形图指定坐标区

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1 和 ax2。在顶部坐标区中显示条形图。在底部坐标区中，显示相同数据的堆叠条形图。

```
y = [1 2 3; 4 5 6]

ax1 = subplot(2,1,1)
ax2 = subplot(2,1,2)

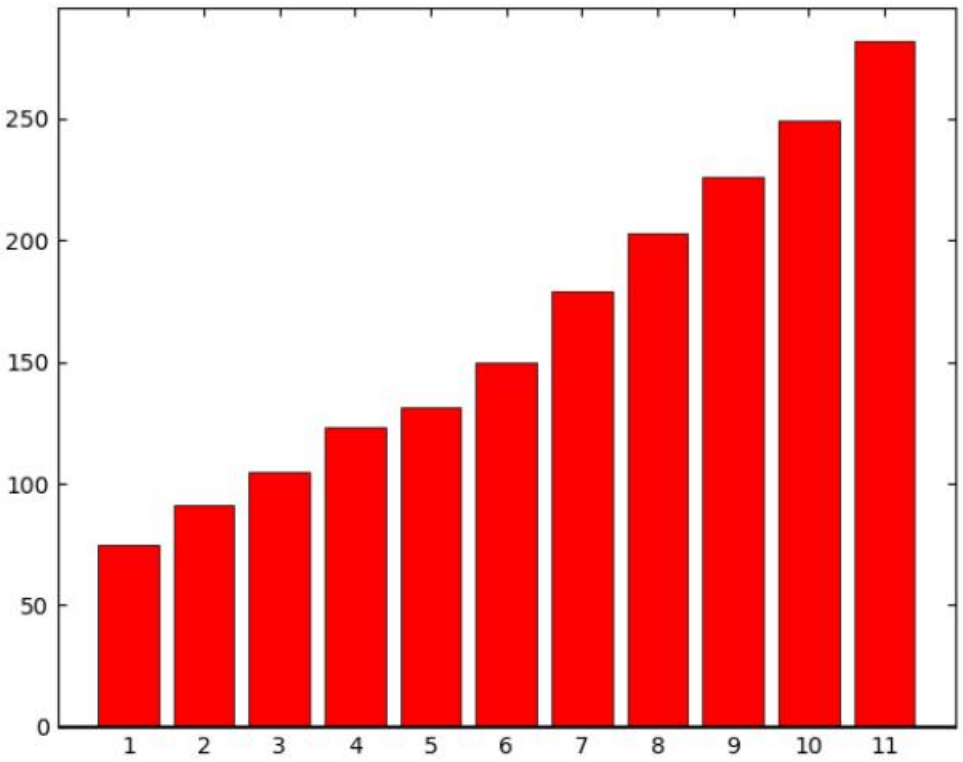
b1 = bar(ax1, y)
b2 = bar(ax2, y, style = "stacked")
```



(10). 指定条形颜色

使用红色条形创建一个条形图。

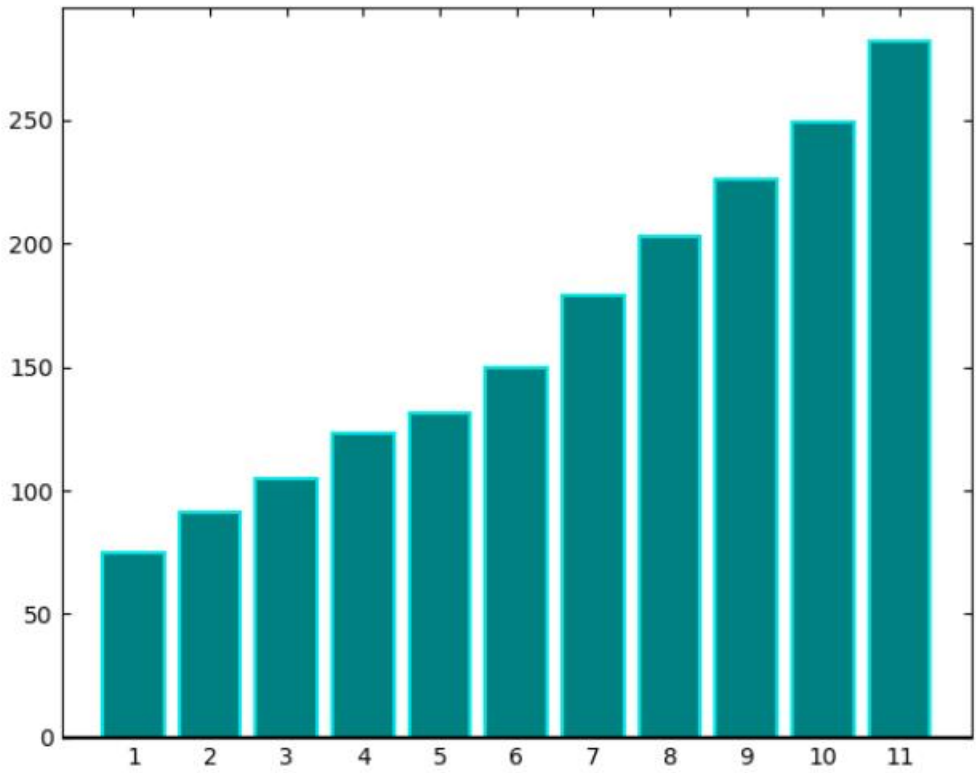
```
y = [75, 91, 105, 123.5, 131, 150, 179, 203, 226, 249, 281.5]
bar(y, color = "r")
```



(11). 指定条形和轮廓颜色

使用 RGB 三元组设置条形内部颜色和轮廓颜色。设置条形轮廓的宽度。

```
y = [75, 91, 105, 123.5, 131, 150, 179, 203, 226, 249, 281.5]
bar(y, facecolor = (0,0.5,0.5), edgecolor = (0,0.9,0.9), linewidth
= 1.5)
```

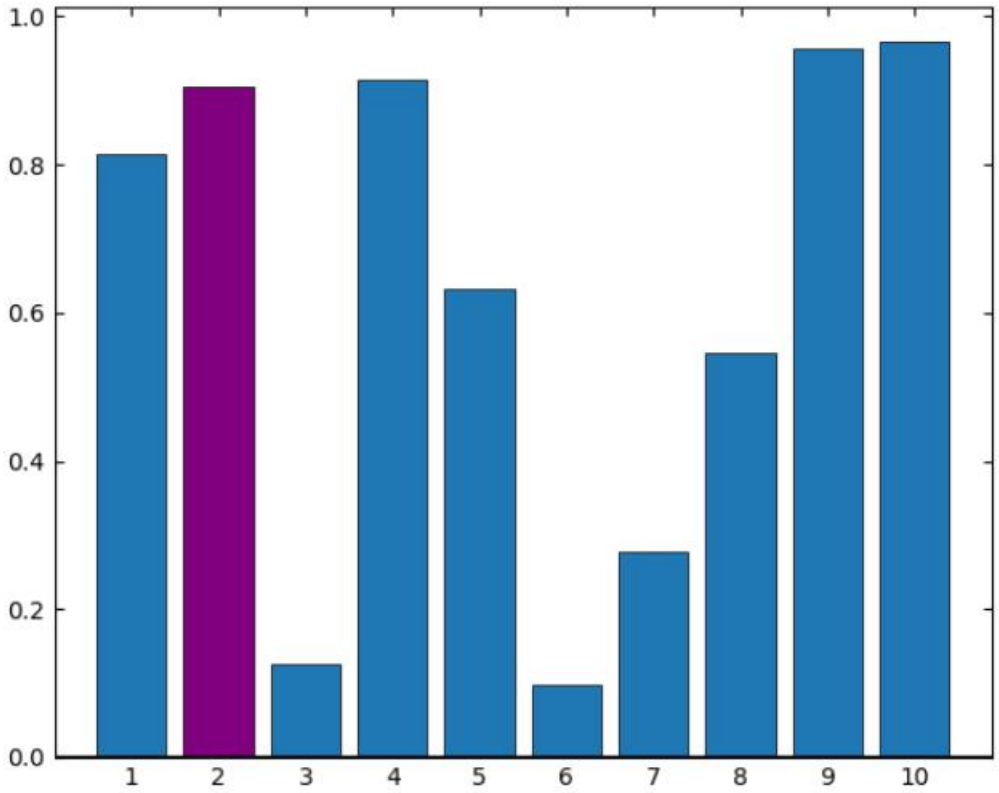


(12). 控制单个条形的颜色

可以使用 Bar 对象的 CData 属性控制单个条形的颜色。

创建一个条形图并将 Bar 对象赋给一个变量。默认情况下，CData 属性预先填充由默认 RGB 颜色值组成的矩阵。要更改特定的颜色，请更改矩阵中的对应行。例如，更改第二个条形的颜色。

```
y=[0.8147
0.9058
0.1270
0.9134
0.6324
0.0975
0.2785
0.5469
0.9575
0.9649]
b=bar(y)
b[1][2].set_facecolor((0.5, 0, 0.5))
```



(13). 使用颜色图颜色的条形图

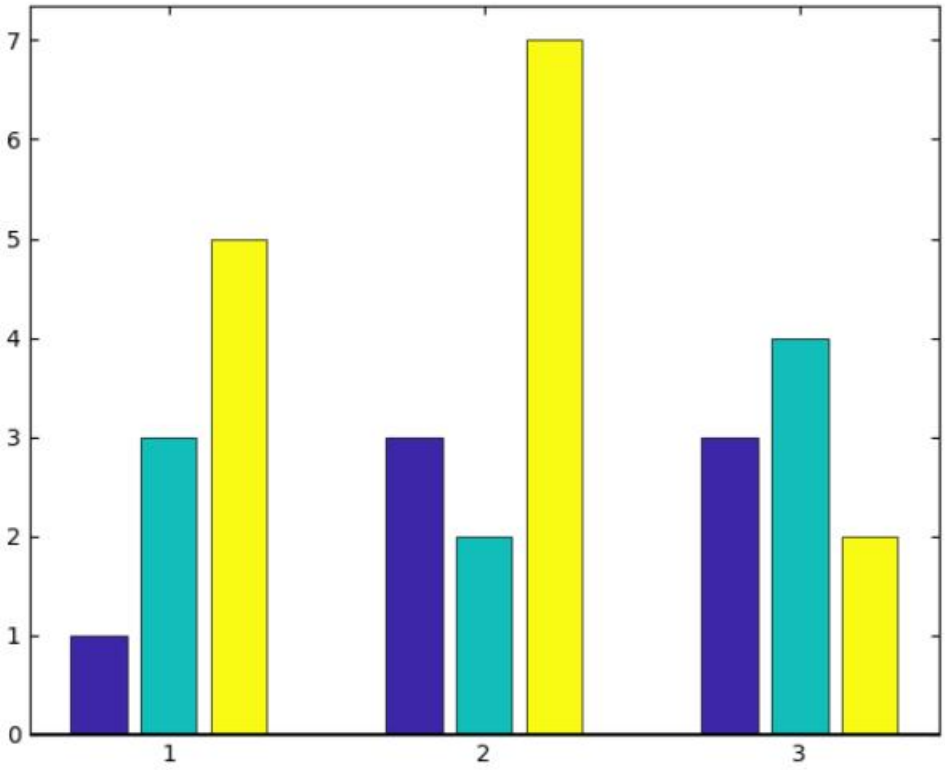
```
c = [0.2422 0.1504 0.6603
    0.0704 0.7457 0.7258
    0.9769 0.9839 0.0805]

y = [1 3 5; 3 2 7; 3 4 2];
bars = bar(y);

for b in bars[1]
    b.set_facecolor([0.2422, 0.1504, 0.6603])
end

for b in bars[2]
    b.set_facecolor([0.0704, 0.7457, 0.7258])
end

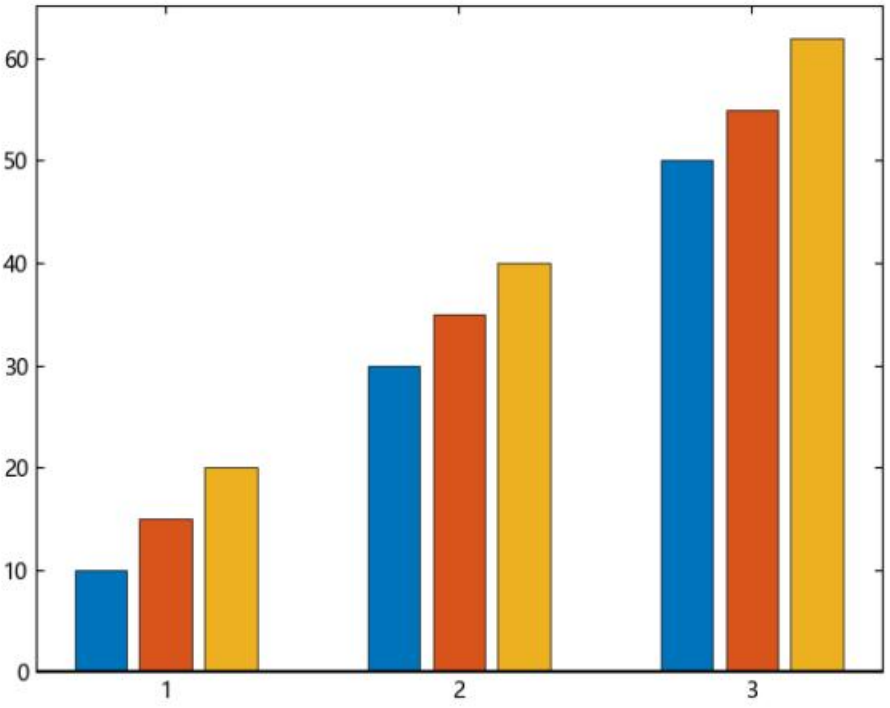
for b in bars[3]
    b.set_facecolor([0.9769, 0.9839, 0.0805])
end
```



(14). 自定义分组或堆叠条形中的一个序列

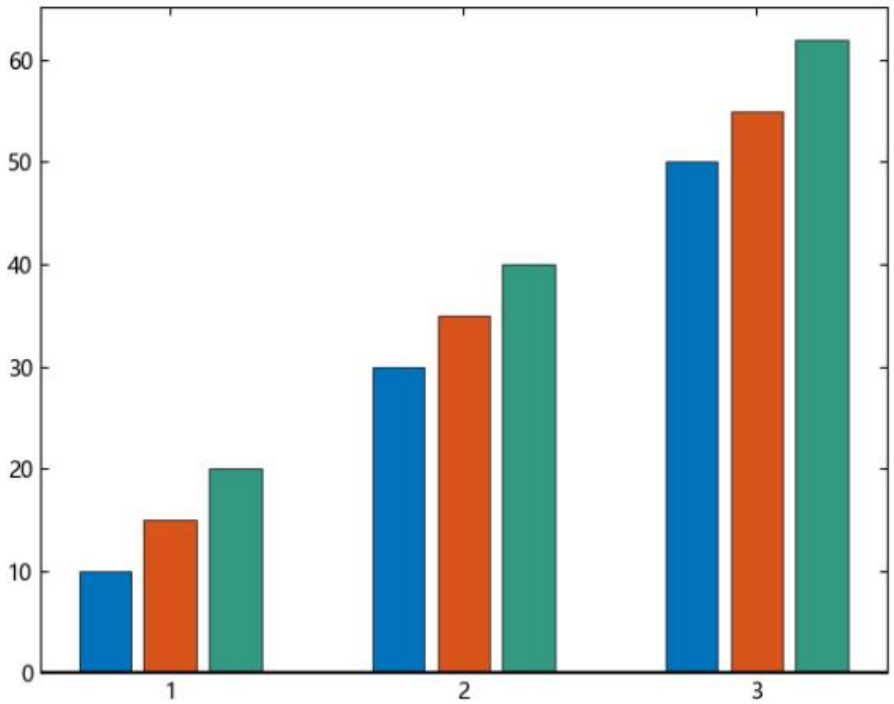
创建矩阵 y，其中每列为一个数据序列。调用 bar 函数以在条形图中显示数据，并指定输出参数。输出是包含三个 Bar 对象的向量，其中每个对象对应一个不同序列。无论条形是分组还是堆叠，均为如此。

```
y = [10 15 20; 30 35 40; 50 55 62];
bars = bar(y);
```



将第三个条形序列设置为绿色。

```
for b in bars[3]
    b.set_facecolor([0.2, 0.6, 0.5])
end
```



barh

水平条形图

A. 语法

```
barh(y)
barh(x,y)

barh(___,Key=Value)

barh(ax,___)

b = barh(___)
```

B. 说明

barh(y)创建一个水平条形图，每个条形对应 y 中一个元素。如果 y 是 m×n 矩阵，则 barh 创建每组包含 n 个条形的 m 个组。

barh(x,y)沿垂直轴在 x 指定的位置绘制条形。

barh(___,Key=Value)使用一个或多个名称-值对组参数指定条形图的属性。仅使用默认"grouped"或"stacked"样式的条形图支持设置条形属性。在所有其他输入参数之后指定名称-值对组参数。

barh(ax,___)在目标坐标区中显示条形图。将坐标区指定为上述任一语法中的第一个参数。

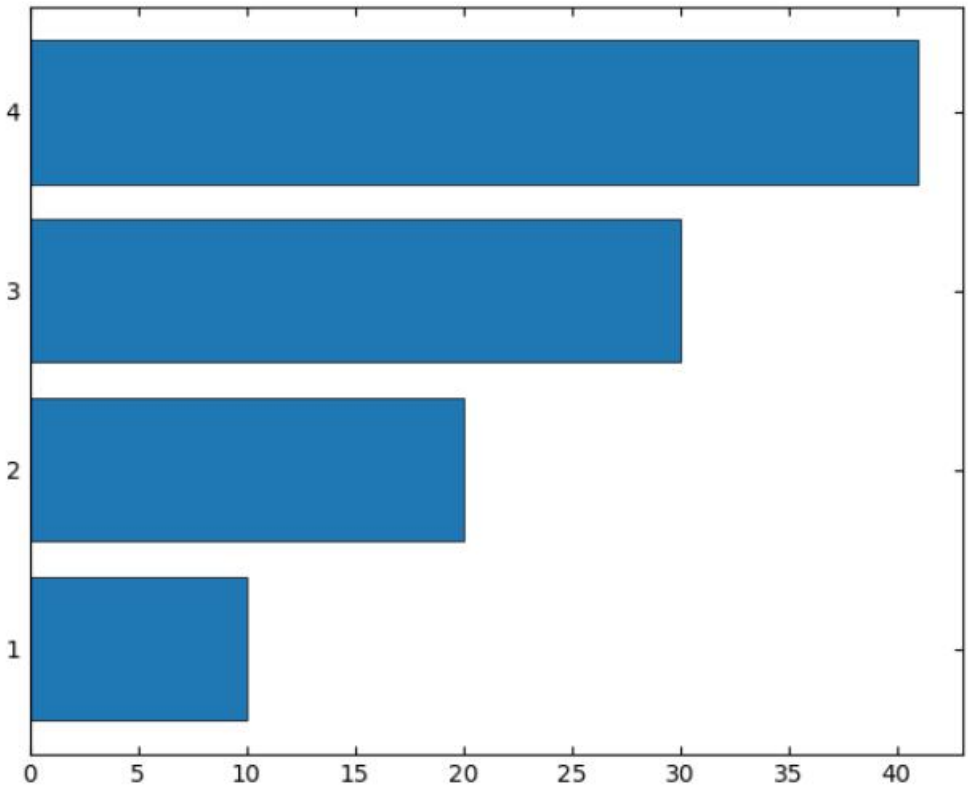
b=barh(___)返回一个或多个 Bar 对象。如果 y 是向量，则 barh 返回一个 Bar 对象。如果 y 是矩阵，则 barh 为每个序列返回一个 Bar 对象。显示条形图后，使用 b 设置条形的属性。

C. 示例

(1). 显示一个条形序列

创建一个包含四个值的向量。在一个条形图中显示这些值，每个水平条形对应一个值。

```
y = [10 20 30 41];
barh(y)
```

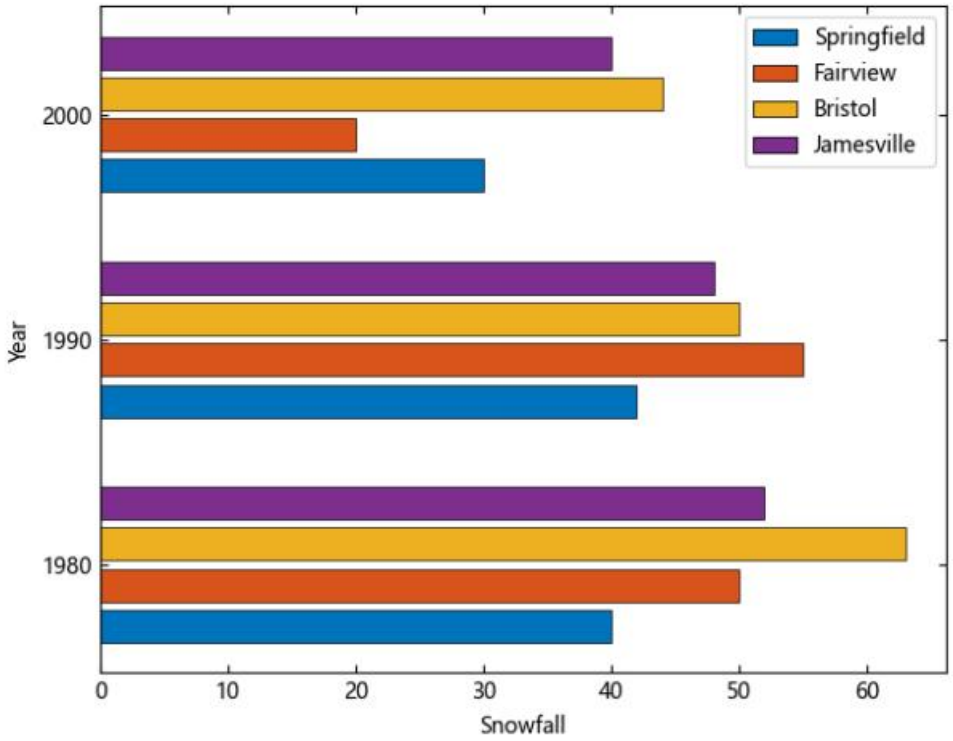


(2). 显示具有轴标签和图例的四个条形序列

将 x 定义为一个包含三个年份值的矩阵。将 y 定义为包含四个城市的降雪数据的矩阵。在每年的组中显示四个条形序列。然后添加轴标签和图例。

```
x = [1980, 1990, 2000]
y = [40 50 63 52; 42 55 50 48; 30 20 44 40]
labels = ["Springfield", "Fairview", "Bristol", "Jamesville"]

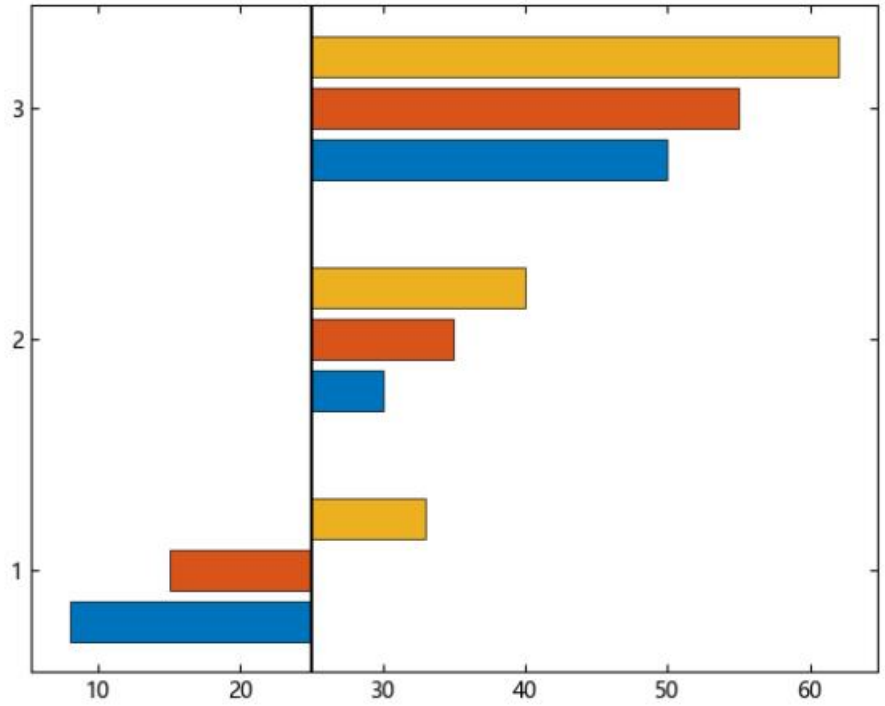
bars = barh(x, y)
xlabel("Snowfall")
ylabel("Year")
legend(bars, labels)
```



(3). 更改基准值

创建矩阵 y。然后在条形图中以 25 为基准值显示 y 值。小于 25 的值显示在基线的左侧。

```
y = [8 15 33; 30 35 40; 50 55 62]
barh(y, basevalue = 25)
```

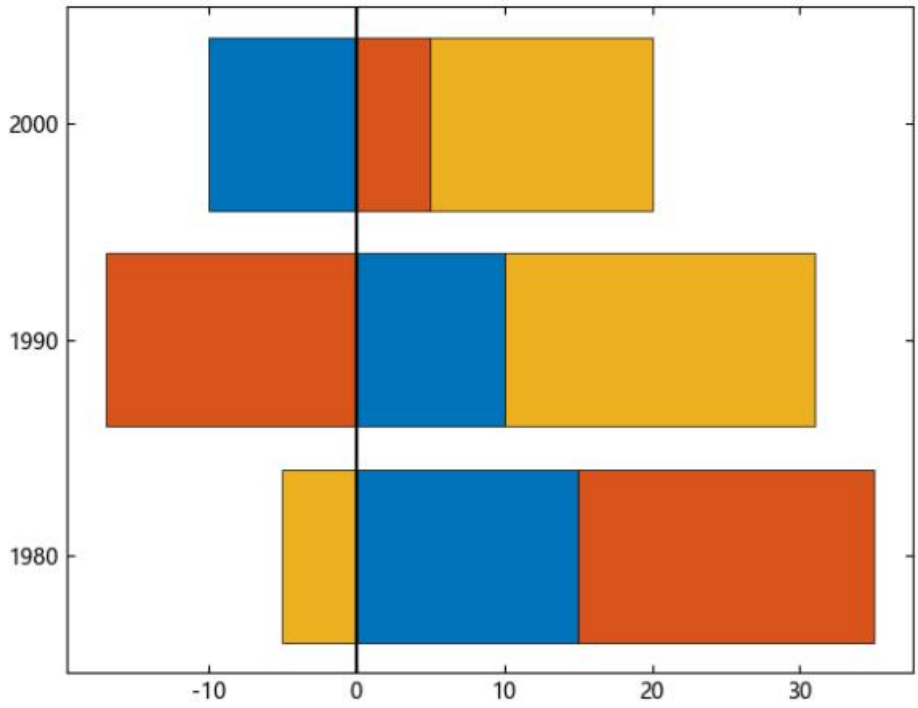


(4). 显示具有负数据的堆叠条形

将 x 定义为一个包含三个年份值的向量。将 y 定义为包含负值和正值组合的矩阵。在堆叠水平条形图中显示值。

```
x = [1980, 1990, 2000]
y = [15 20 -5; 10 -17 21; -10 5 15]

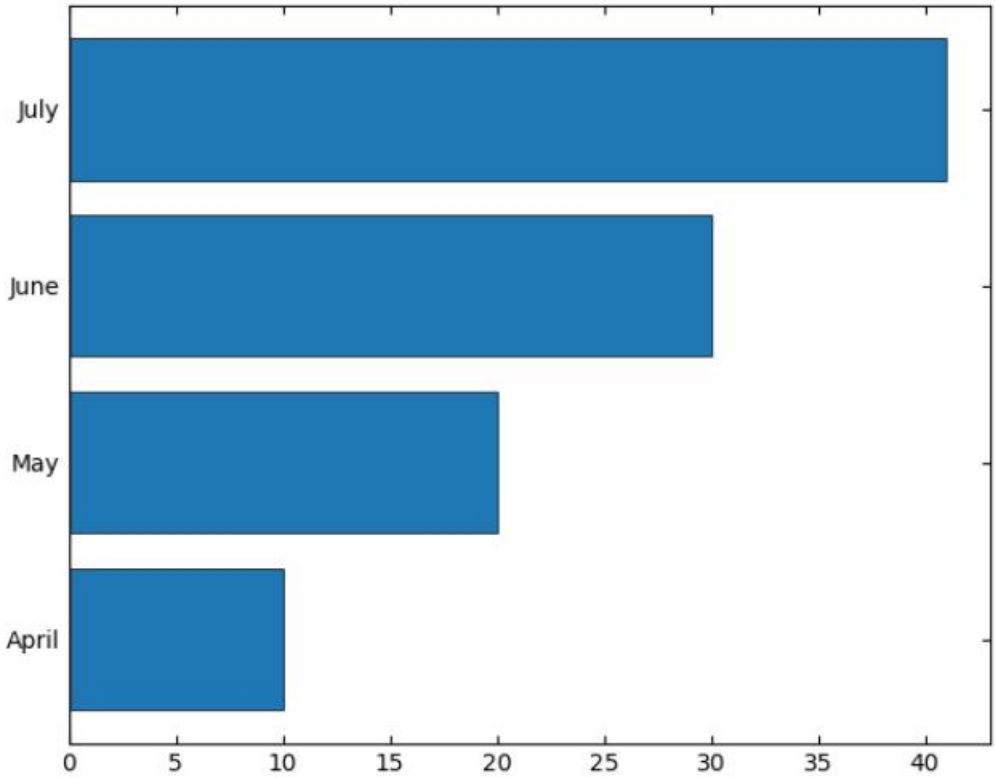
barh(x, y, style = "stacked")
```



(5). 自定义垂直轴刻度标签

将 y 定义为包含四个值的向量，并在水平条形图中显示值。然后调用 yticklabels 函数来更改垂直轴上的刻度标签。

```
x = ["April","May","June","July"]
y = [10, 20, 30, 41]
barh(y)
yticklabels(x)
```

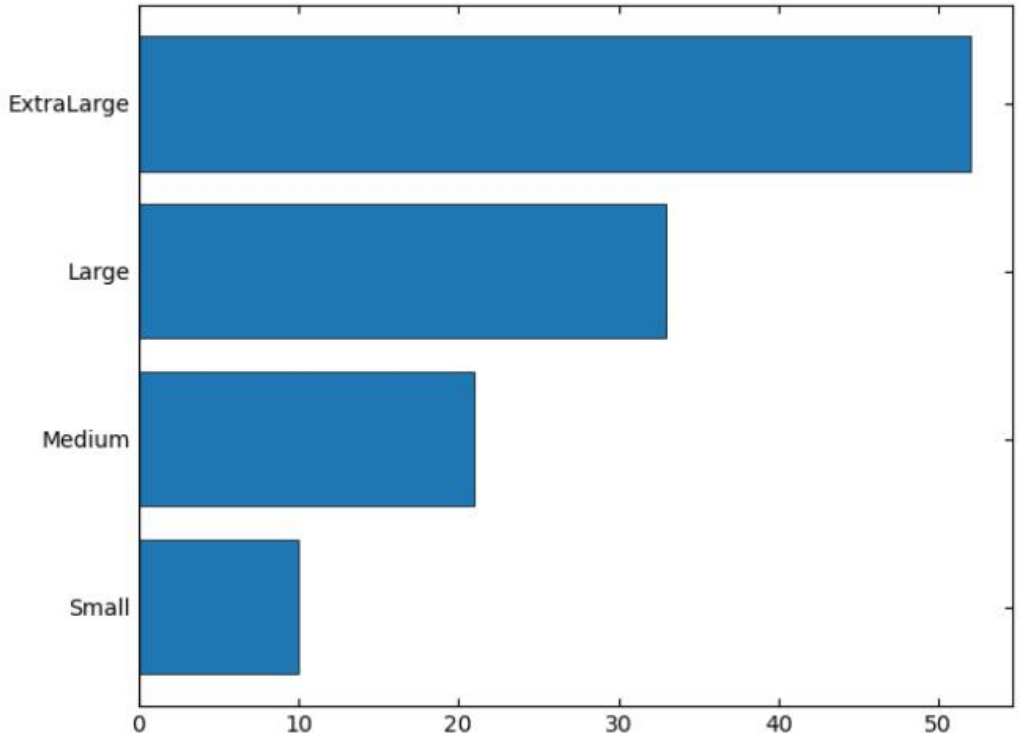


(6). 指定分类数据

指示条形图类别的一种方法是将 X 指定为分类数组。barh 函数使用经过排序的类别列表，因此条形的显示顺序可能与您预期的有所不同。

将 X 定义为分类数组，然后将 Y 定义为条形长度的向量，并显示条形图。

```
x = ["Small","Medium","Large","ExtraLarge"]
y = [10,21,33,52]
barh(x, y)
```

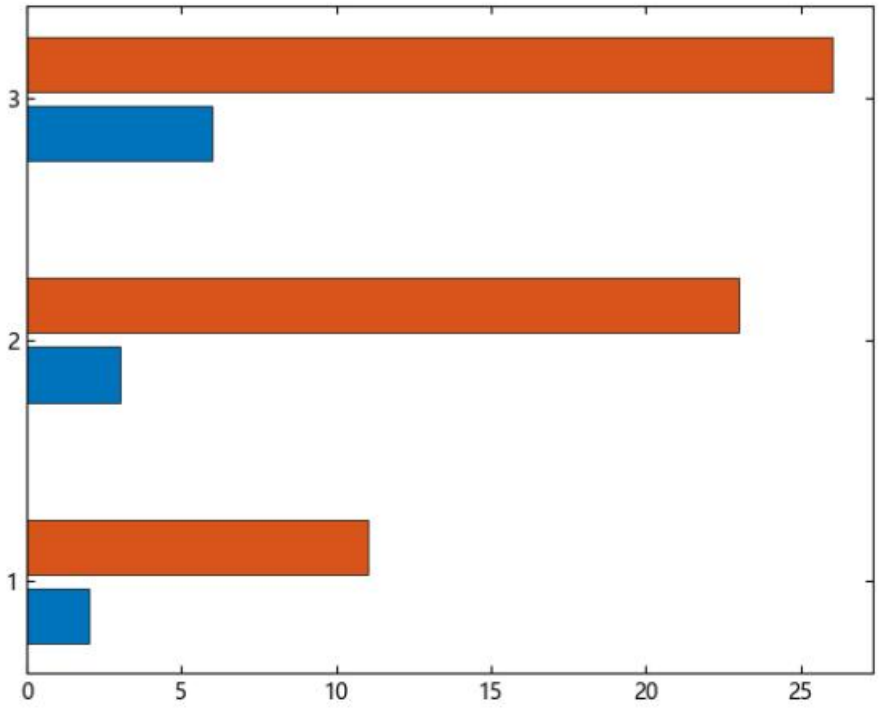


(7). 在条形末端添加标签

将 vals 定义为包含两个数据序列的矩阵。在水平条形图中显示数据，并指定输出参数。由于有两个序列，barh 返回包含两个 Bar 对象的向量。

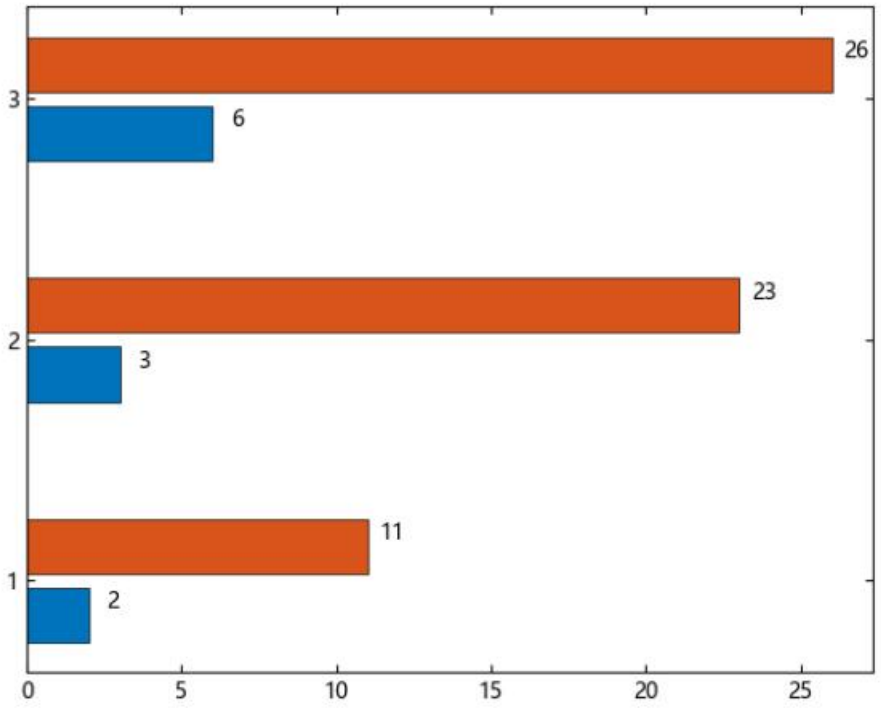
```
x = [1,2,3]
y = [2 3 6; 11 23 26]'

barh(x,y)
```



接下来，在条形序列的末端显示标签。

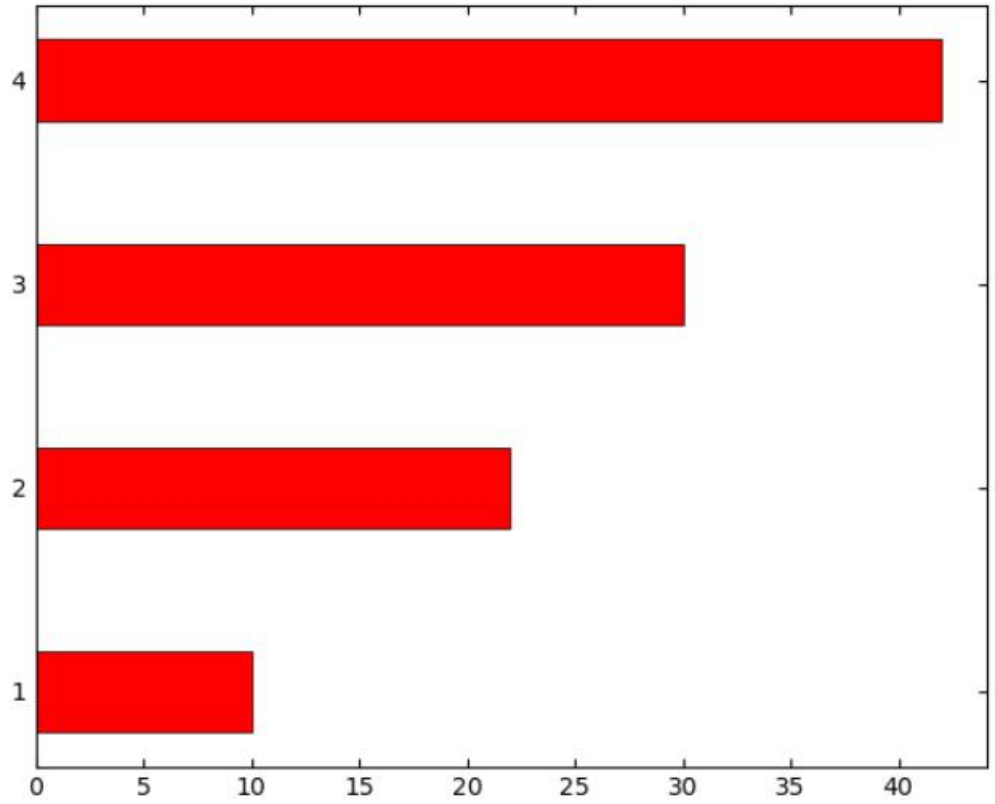
```
barh(x,y, text = y)
```



(8). 指定条形宽度和颜色

创建数据并在水平条形图中显示数据，该条形图中条形的宽度为 0.4，颜色为红色。

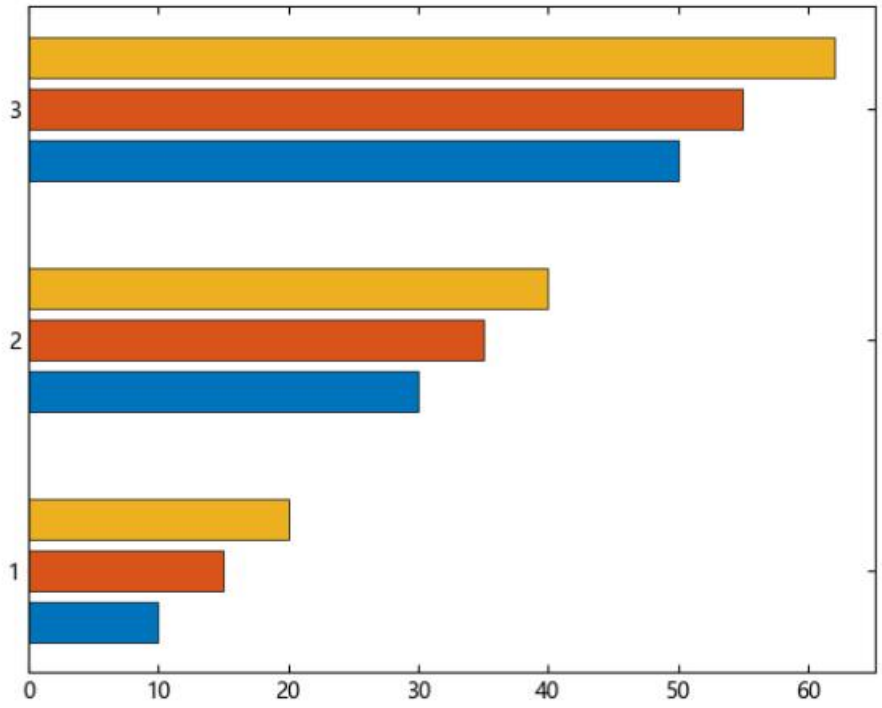
```
y = [10,22,30,42]
barh(y, height = 0.4, color = "r")
```



(9). 自定义一个条形序列

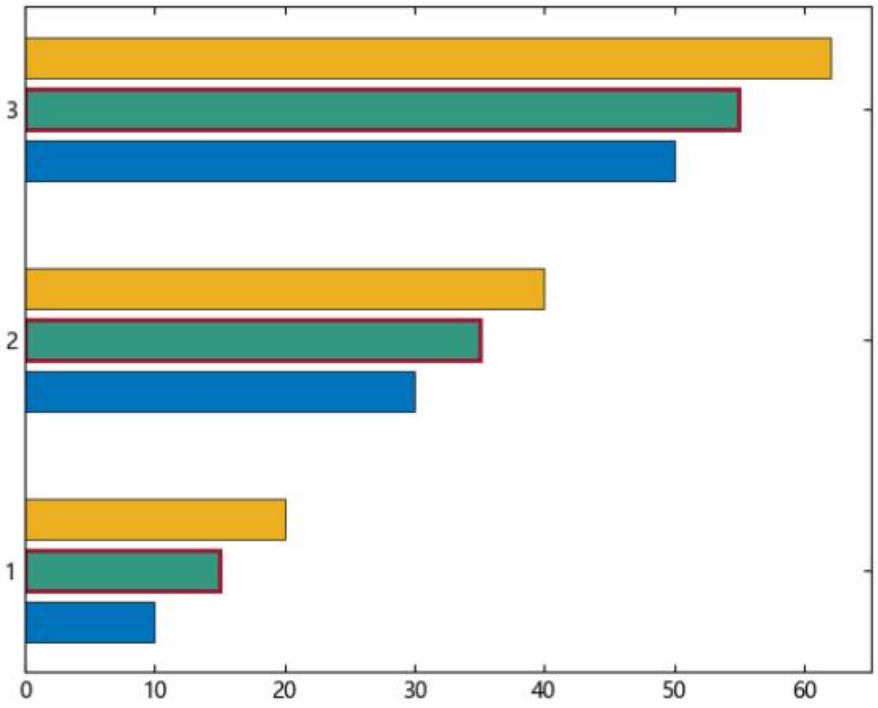
创建矩阵 y，其中每列为一个数据序列。然后在条形图中显示数据，指定在调用 barh 函数时的输出参数。在本例中，barh 返回包含三个 Bar 对象的向量。每个对象对应于一个不同序列。

```
y = [10 15 20; 30 35 40; 50 55 62]
bars = barh(y)
```



将第二个条形序列改为绿色，并使用加粗的红边。

```
for b in bars[2]
    b.set_facecolor([.2, .6, .5])
    b.set_edgecolor([.63, .08, .18])
    b.set_linewidth(2)
end
```



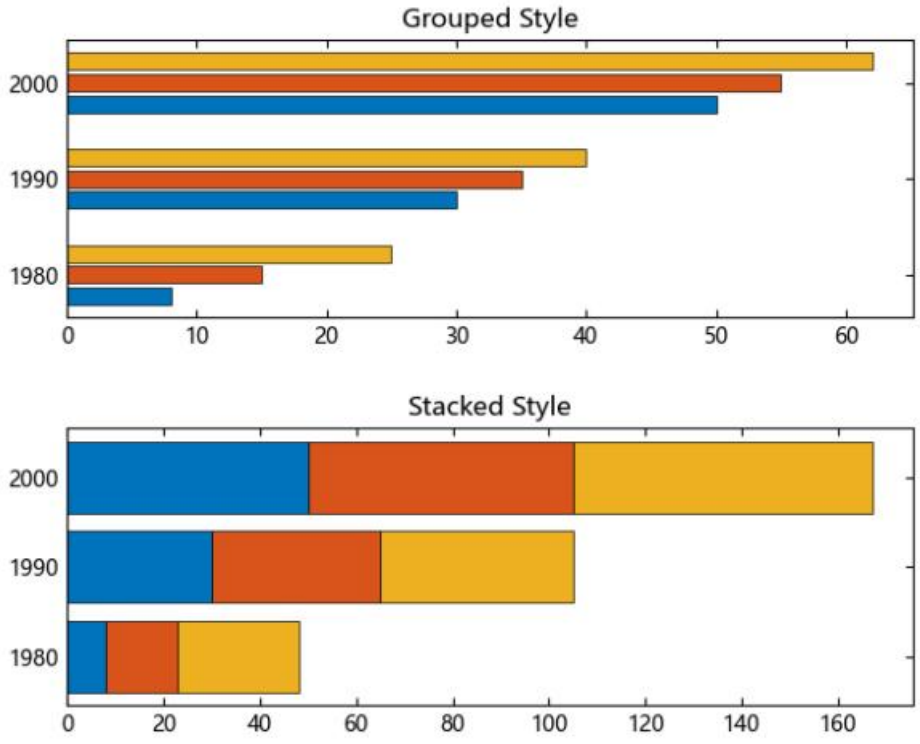
(10). 比较不同的条形样式

创建数据，并使用使用默认的 "grouped" 样式和 "stacked" 样式在两个不同的条形图中显示数据。

```
x = [1980, 1990, 2000]
y = [8 15 25; 30 35 40; 50 55 62]

ax1 = subplot(2,1,1)
barh(ax1, x, y)
title("Grouped Style")

ax2 = subplot(2,1,2)
barh(ax2, x, y, style="stacked")
title("Stacked Style”)
```



bar3

绘制三维条形图

A. 语法

```
bar3(Z)
bar3(Y,Z)
bar3(___,Key=Value)
bar3(ax,___)
```

```
h = bar3(___)
```

B. 说明

bar3() 绘制三维条形图。

bar3(Z) 绘制三维条形图，Z 中的每个元素对应一个条形图。如果 Z 是向量，y 轴的刻度范围是从 1 至 length(Z)。如果 Z 是矩阵，则 y 轴的刻度范围是从 1 到 Z 的行数。

bar3(Y,Z) 在 Y 指定的位置绘制 Z 中各元素的条形图，其中 Y 是为垂直条形定义 y 值的向量。y 值可以是非单调的，但不能包含重复值。如果 Z 是矩阵，则 Z 中位于同一行内的元素将出现在 y 轴上的相同位置。

bar3(___, Key=Value) 使用一个或多个名称-值对组参数修改三维条形图图。比如设置条形宽度并控制组中各个条形的间隔，width = 1。

bar3(ax,___) 将图形绘制到 ax 坐标区中，而不是当前坐标区 (gca) 中。

h = bar3(___) 返回由 Poly3DCollection 对象组成的向量。如果 Z 是矩阵，则 bar3 将为 Z 中的每一列创建一个 Poly3DCollection 对象。

C. 示例

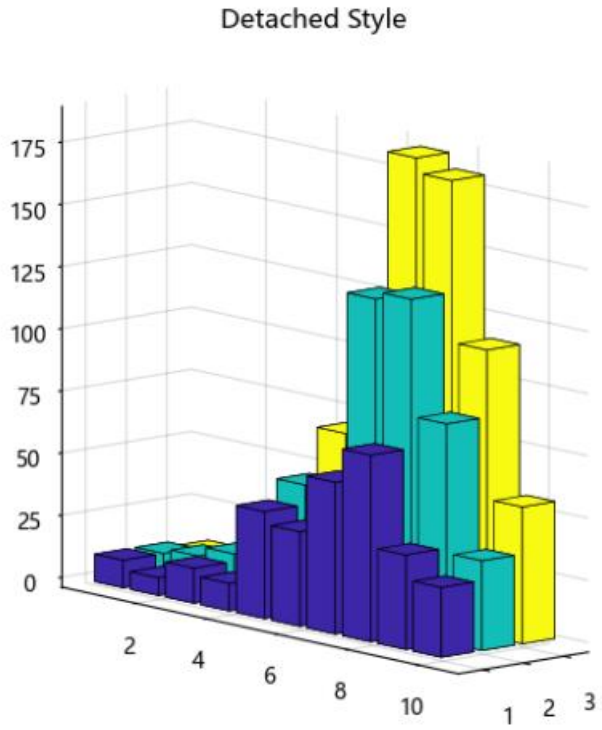
(1). 创建三维条形图

加载数据 z 矩阵，它会返回一个三列矩阵 count。将 Z 保存为 count 的前 10 行。

```
z = [11 11 9
7 13 11
14 17 20
11 13 9
43 51 69
38 46 76
61 132 186
75 135 180
38 88 115
28 36 55]
```

创建 Z 的三维条形图。默认情况下，样式为 detached。

```
figure()
b3 = bar3(z)
title("Detached Style")
```



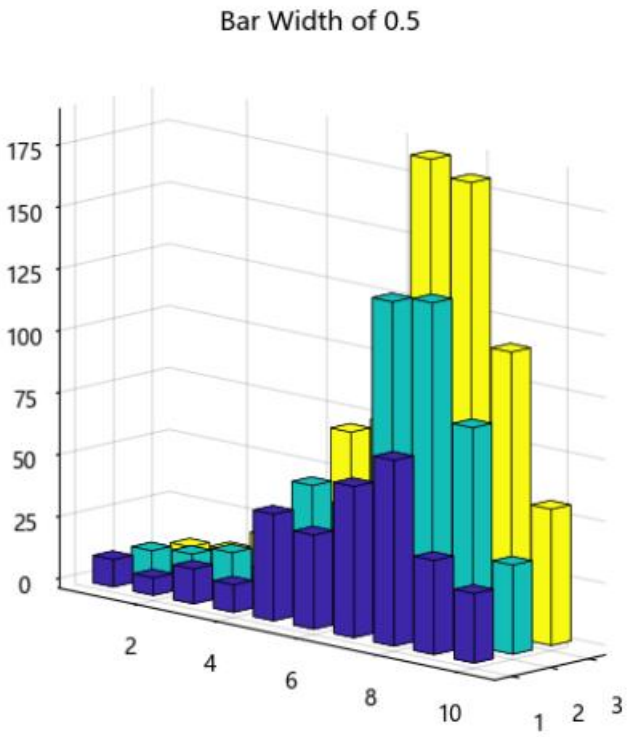
(2). 为三维条形图指定条形宽度

加载数据 z 矩阵，它会返回一个三列矩阵 count。将 Z 保存为 count 的前 10 行。

```
z = [11 11 9
7 13 11
14 17 20
11 13 9
43 51 69
38 46 76
61 132 186
75 135 180
38 88 115
28 36 55]
```

创建 Z 的三维条形图，并将条形宽度设置为 0.5。

```
figure()
b3 = bar3(z, width = 0.5)
title("Bar Width of 0.5")
```



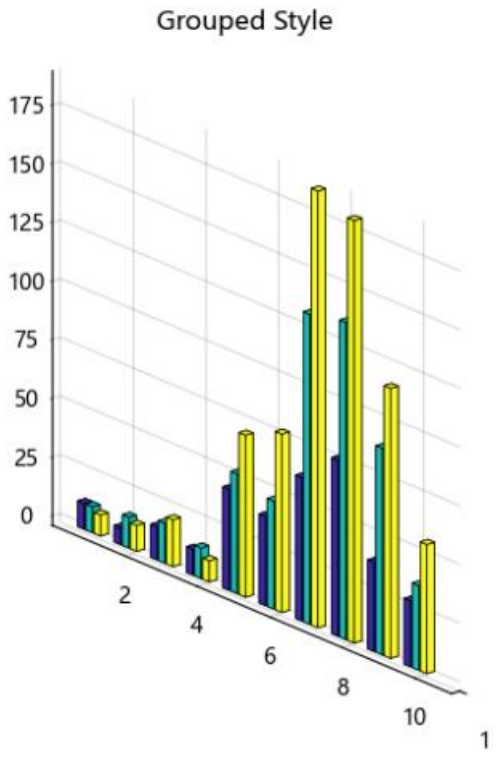
(3). 分组样式的三维条形图

加载数据 z 矩阵，它会返回一个三列矩阵 count。将 Z 保存为 count 的前 10 行。

```
z = [11 11 9
7 13 11
14 17 20
11 13 9
43 51 69
38 46 76
61 132 186
75 135 180
38 88 115
28 36 55]
```

创建 Z 的三维条形图。通过指定样式选项为 grouped 对 Z 每行元素进行分组。

```
figure()
b3 = bar3(z, style="grouped")
title("Grouped Style")
```



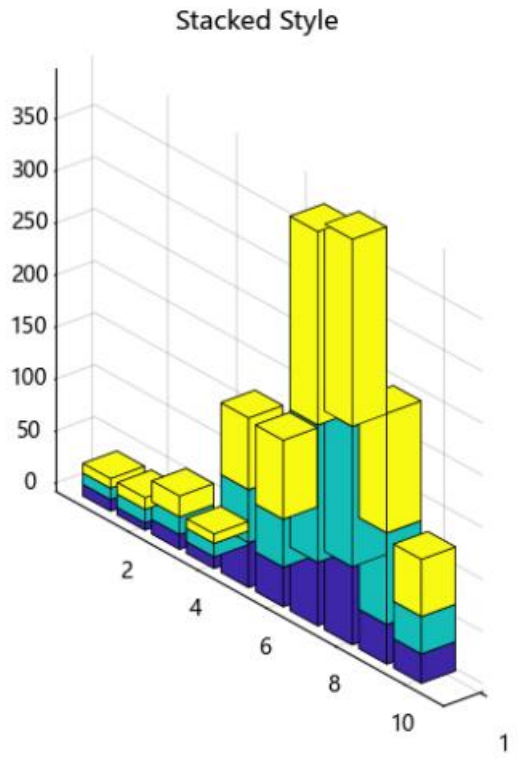
(4). 堆叠样式的三维条形图

加载数据 z 矩阵，它会返回一个三列矩阵 count。将 Z 保存为 count 的前 10 行。

```
z = [11 11 9
7 13 11
14 17 20
11 13 9
43 51 69
38 46 76
61 132 186
75 135 180
38 88 115
28 36 55]
```

创建 Z 的三维条形图。通过指定样式选项为 stacked 对 Z 每行元素进行堆叠。

```
figure()
b3 = bar3(z, style="stacked")
title("Stacked Style")
```



bar3h

绘制水平三维条形图

A. 语法

```
bar3h(Y)
bar3h(Z,Y)
bar3h(___,Key=Value)
bar3h(ax,___)
h = bar3h(___)
```

B. 说明

bar3h() 绘制三维水平条形图。

bar3h(Y) 绘制三维条形图，Y 中的每个元素对应一个条形图。如果 Y 是向量，则 z 轴的刻度范围是从 1 到 length(Y)。如果 Y 是矩阵，则 z 轴的刻度范围是从 1 到 Y 的行数。

bar3h(Z,Y) 在 Z 指定的位置绘制 Y 中各元素的条形图，其中 Z 是为水平条形定义 z 值的向量。z 值可以是非单调的，但不能包含重复值。如果 Y

是矩阵，则 Y 中位于同一行内的元素将出现在 z 轴上的相同位置。

bar3h(___, Key=Value) 使用一个或多个名称-值对组参数修改三维条形图图。比如设置条形宽度并控制组中各个条形的间隔，height = 1。

bar3h(ax,___) 将图形绘制到 ax 坐标区中，而不是当前坐标区 (gca) 中。

h = bar3h(___) 返回由 Poly3DCollection 对象组成的向量。如果 Y 是矩阵，则 bar3h 将为 Y 中的每一列创建一个 Poly3DCollection 对象。

C. 示例

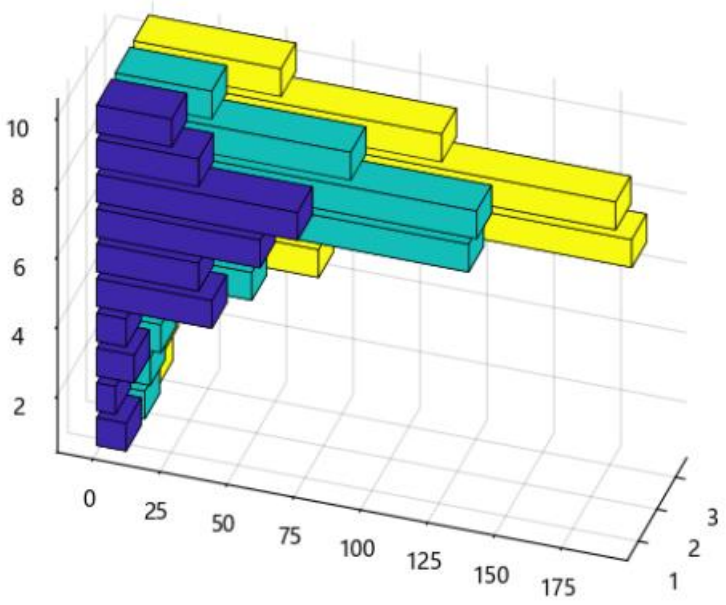
(1). 创建三维水平条形图

加载数据 y 矩阵，它会返回一个三列矩阵 count。将 Y 保存为 count 的前 10 行。

```
y = [11 11 9
7 13 11
14 17 20
11 13 9
43 51 69
38 46 76
61 132 186
75 135 180
38 88 115
28 36 55]
```

创建 Y 的三维水平条形图。默认情况下，样式为 detached。

```
figure()
b3 = bar3h(y)
```



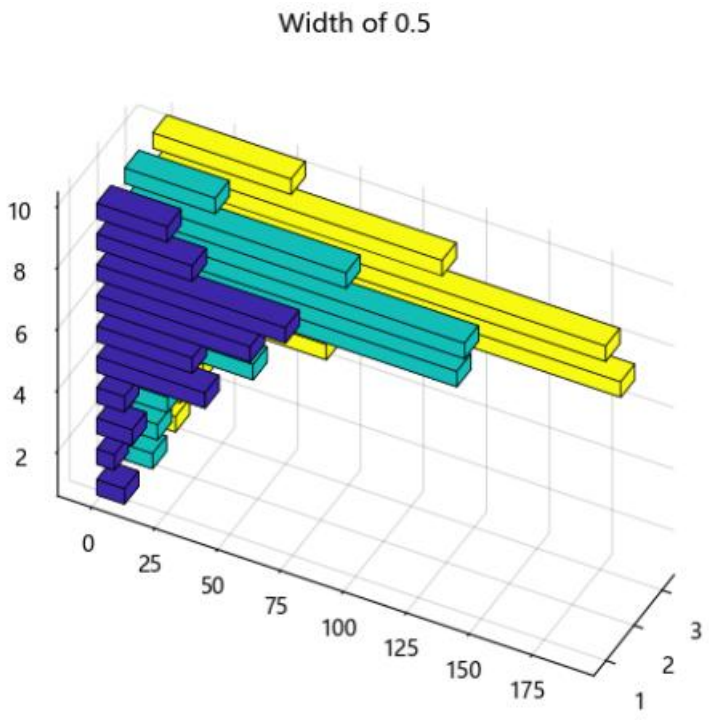
(2). 为三维水平条形图指定条形宽度

加载数据 y 矩阵，它会返回一个三列矩阵 count。将 Y 保存为 count 的前 10 行。

```
y = [11 11 9
7 13 11
14 17 20
11 13 9
43 51 69
38 46 76
61 132 186
75 135 180
38 88 115
28 36 55]
```

创建 Y 的三维条形图，并将条形宽度设置为 0.5。

```
figure()
b3 = bar3h(y, width = 0.5)
title("Width of 0.5")
```



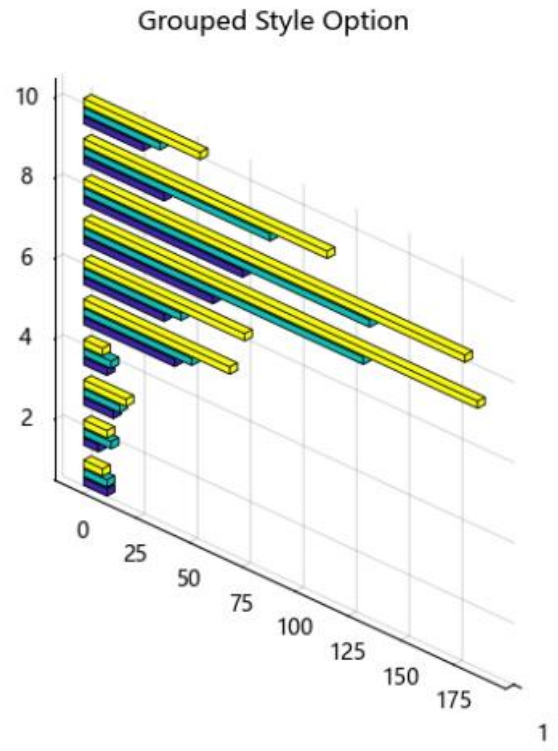
(3). 分组样式的三维水平条形图

加载数据 y 矩阵，它会返回一个三列矩阵 count。将 Y 保存为 count 的前 10 行。

```
y = [11 11 9
7 13 11
14 17 20
11 13 9
43 51 69
38 46 76
61 132 186
75 135 180
38 88 115
28 36 55]
```

创建 Y 的三维水平条形图，并将样式选项指定为 grouped。

```
figure()
b3 = bar3h(y, style="grouped")
title("Grouped Style Option")
```



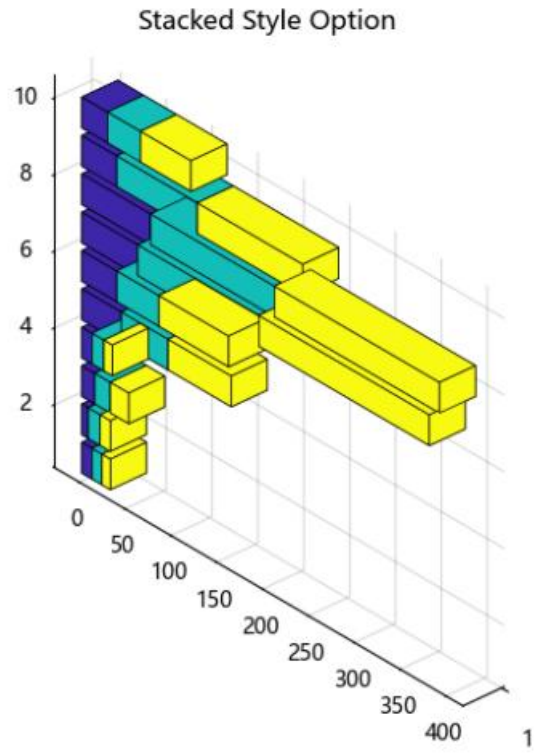
(4). 堆叠样式的三维水平条形图

加载数据 y 矩阵，它会返回一个三列矩阵 count。将 Y 保存为 count 的前 10 行。

```
y = [11 11 9
7 13 11
14 17 20
11 13 9
43 51 69
38 46 76
61 132 186
75 135 180
38 88 115
28 36 55]
```

创建 Y 的三维水平条形图，并将样式选项指定为 stacked。

```
figure()
b3 = bar3h(y, style="stacked")
title("Stacked Style Option")
```



pareto

帕累托图

A. 语法

```
pareto(Y)
pareto(Y,X)
H = pareto(__)
H,ax = pareto(__)
```

B. 说明

帕累托图将向量 Y 中的值显示为降序排列的条形图。Y 中的值必须是非负的且不能包含 NaN。默认情况下，显示最高的 10 个条形或累积分布的前 95% 的值，以两者中较小者为准。

pareto(Y) 用 Y 中的元素索引标识每个条形，并绘制线条以显示 Y 的累积和。

pareto(Y,X) 用 X 中的关联值标识每个条形。

pareto(ax,__) 在坐标区 ax 而不是当前坐标区 gca 中绘图。

pareto(__,threshold) 指定 0 和 1 之间的一个阈值。阈值是要包含在图中的累积直方图的比例。

H = pareto(__) 返回创建的原始 Line 和 Bar 对象以及坐标轴集合。

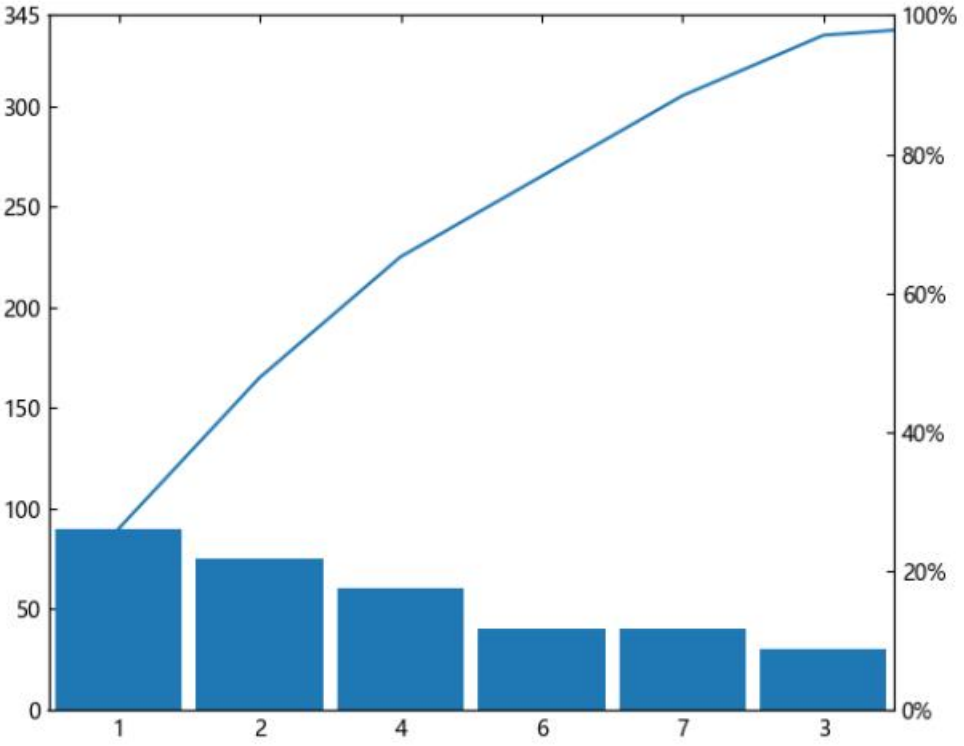
此外，H,ax = pareto(__) 还返回创建的两个坐标区对象。

C. 示例

(1). 创建帕累托图

创建向量 y 的帕累托图。

```
y = [90,75,30,60,5,40,40,5];
figure()
pareto(y)
```



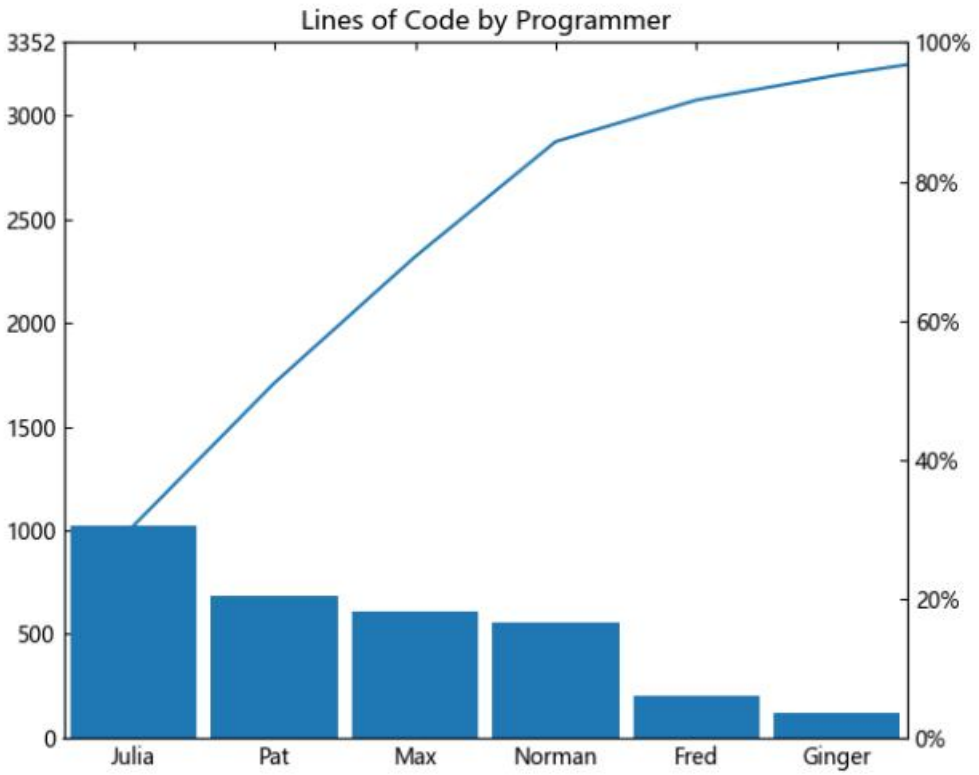
pareto 将 y 中的元素绘制为降序排列的条形图并用其在 y 中的索引标记每个条形。由于 pareto 仅绘制累积分布的前 95%，因此 y 中的部分元素并未显示。

(2). 标记帕累托图中的条形

查看一组编程人员的累积生产率以了解其是否为正态分布。用编程人员的姓名标记每个条形。

```
codelines = [200 120 555 608 1024 101 57 687];
coders                                                              =
["Fred","Ginger","Norman","Max","Julia","Wally","Heidi","Pat"];

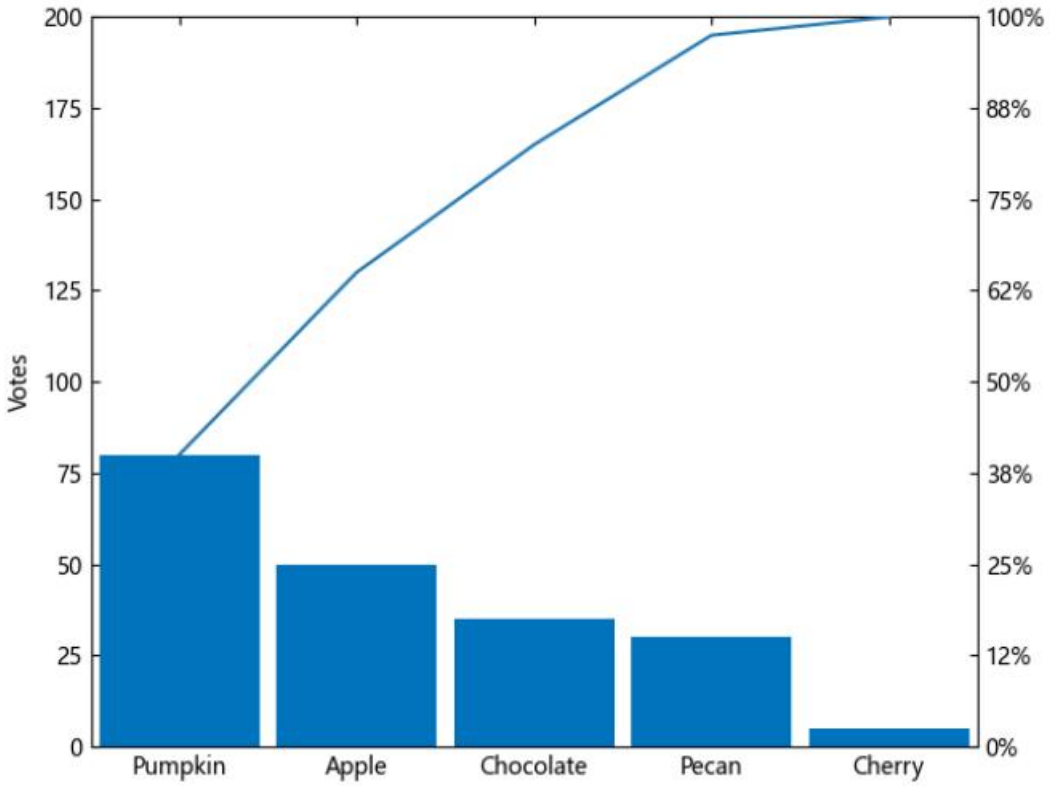
figure()
pareto(codelines, coders)
title("Lines of Code by Programmer")
```



(3). 显示累积分布中的所有值

创建一个帕累托图，在包含 200 名参与者的一项调查中显示所青睐的馅饼类型情况。通过将 threshold 参数设置为 1，包括累积分布中的所有值。

```
pies = ["Chocolate","Apple","Pecan","Cherry","Pumpkin"];
votes = [35 50 30 5 80];
pareto(votes,pies,1)
ylabel("Votes")
```



3.3.1.3.2 针状图

函数名	简介
stem	针状图
stem3	三维针状图

stem

绘制离散序列数据

A. 语法

```
stem(Y)
stem(X,Y)
stem(___,Key=Value)

stem(ax,___)
```

```
h = stem(___)
```

B. 说明

stem(Y) 将数据序列 Y 绘制为从沿 x 轴的基线延伸的针状图。各个数据值由终止每个针状图的圆指示。

如果 Y 是向量，x 轴的刻度范围是从 1 至 length(Y)。

如果 Y 是矩阵，则 stem 将根据相同的 x 值绘制行中的所有元素，并且 x 轴的刻度范围是从 1 至 Y 中的行数。

stem(X,Y) 在 X 指定的值的位置绘制数据序列 Y。X 和 Y 输入必须是大小相同的向量或矩阵。另外，X 可以是行或列向量，Y 必须是包含 length(X) 行的矩阵。

如果 X 和 Y 都是向量，则 stem 将根据 X 中的对应项绘制 Y 中的各项。

如果 X 是向量，Y 是矩阵，则 stem 将根据 X 指定的值集绘制 Y 的每列，这样 Y 的一行中的所有元素都是根据相同的值而绘制。

如果 X 和 Y 都是矩阵，则 stem 将根据 X 的对应列绘制 Y 的列。

stem(___,Key=Value)使用一个或多个 Key=Value 对组参数修改针状图。示例

stem(ax,___)将图形绘制到 ax 指定的坐标区中，而不是当前坐标区(gca)中。选项 ax 可以位于前面的语法中的任何输入参数组合之前。

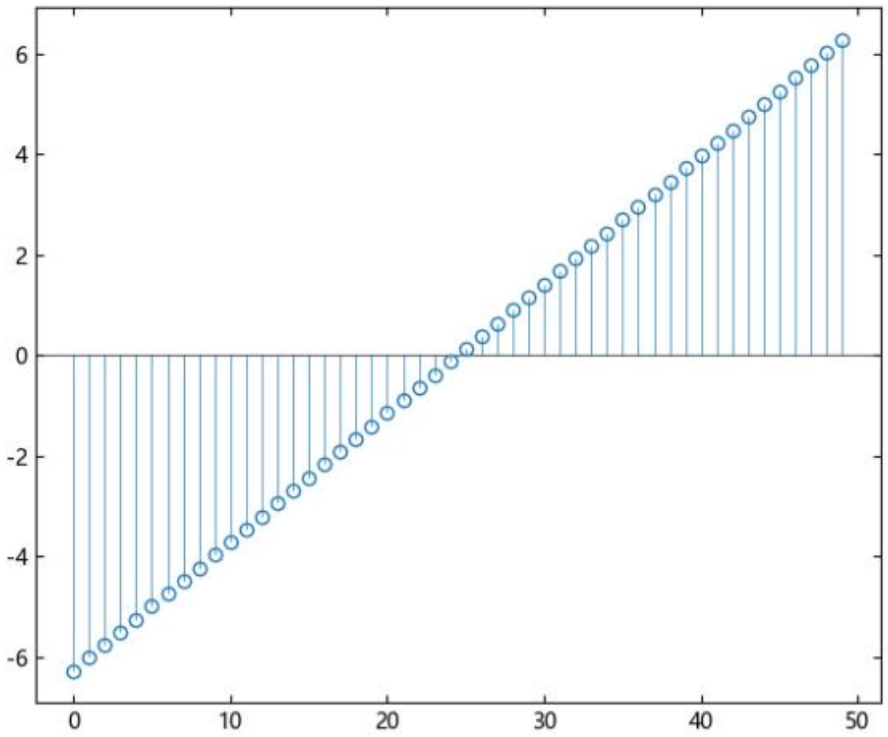
h = stem(___) 在 h 中返回由 Stem 对象构成的向量。使用 h 可在创建针状图后对其进行修改。

C. 示例

(1). 绘制单数据序列

创建一个包含 −2π 和 2π 之间的 50 个数据值的针状图。

```
using TyPlot
Y = LinRange(-2 * pi, 2 * pi, 50);
stem(Y)
```

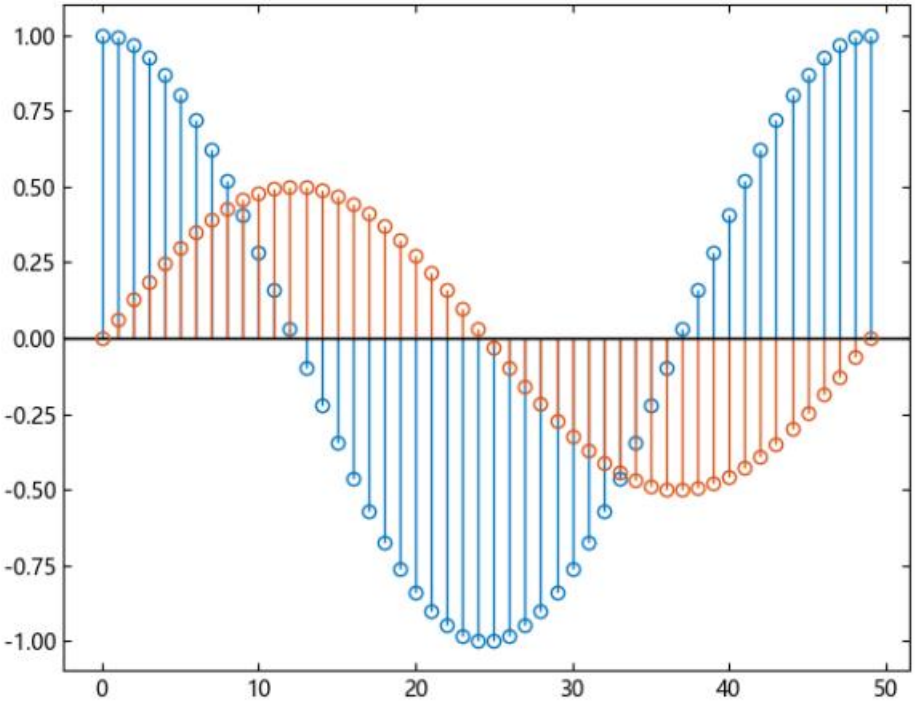


数据值绘制为从基线开始延伸并在数据值位置终止的针状线条。Y 的长度自动确定每个针状图在 x 轴上的位置。

(2). 绘制多个数据序列

使用两列矩阵绘制两个数据序列。

```
using TyPlot
x = LinRange(0, 2 * pi, 50)'
y = cos.(x)
y1 = 0.5 .* sin.(x)
stem(y)
hold("on")
h = stem(y1)
hold("off")
```

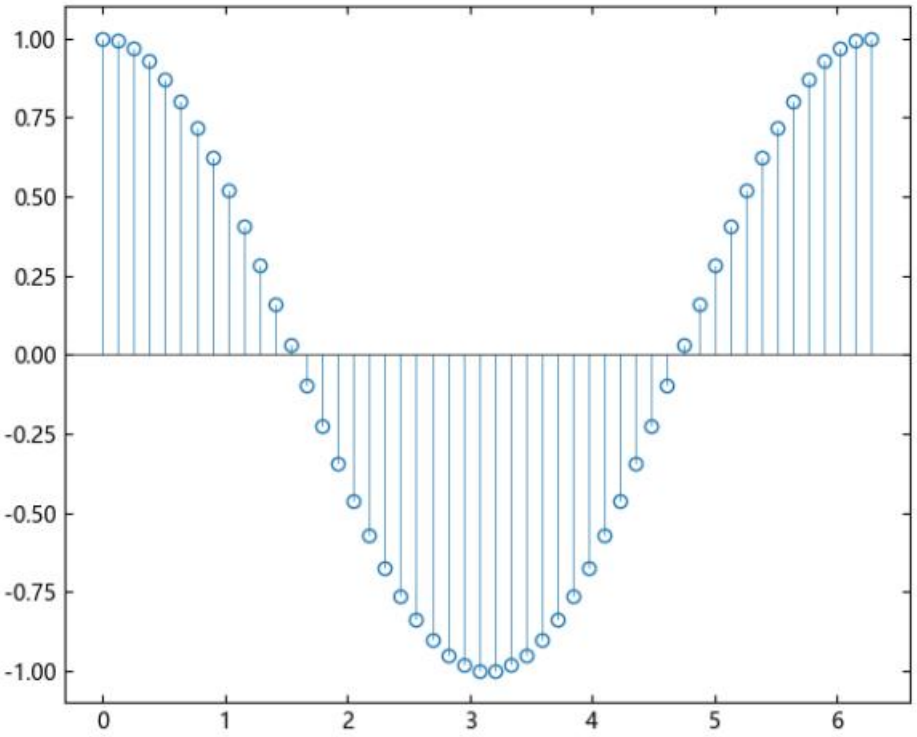


Y 的每列绘制为一个单独的序列，且 Y 的同一行的各项会针对同一 x 值进行绘制。Y 中的行数自动生成每个针状线条在 x 轴上的位置。

(3). 在指定的 x 值位置绘制单个数据序列

绘制在 0 和 2π 之间计算的 50 个余弦数据值并为针状图指定 x 值集。

```
using TyPlot
X = LinRange(0, 2 * pi, 50)';
Y = cos.(X);
stem(X, Y)
```

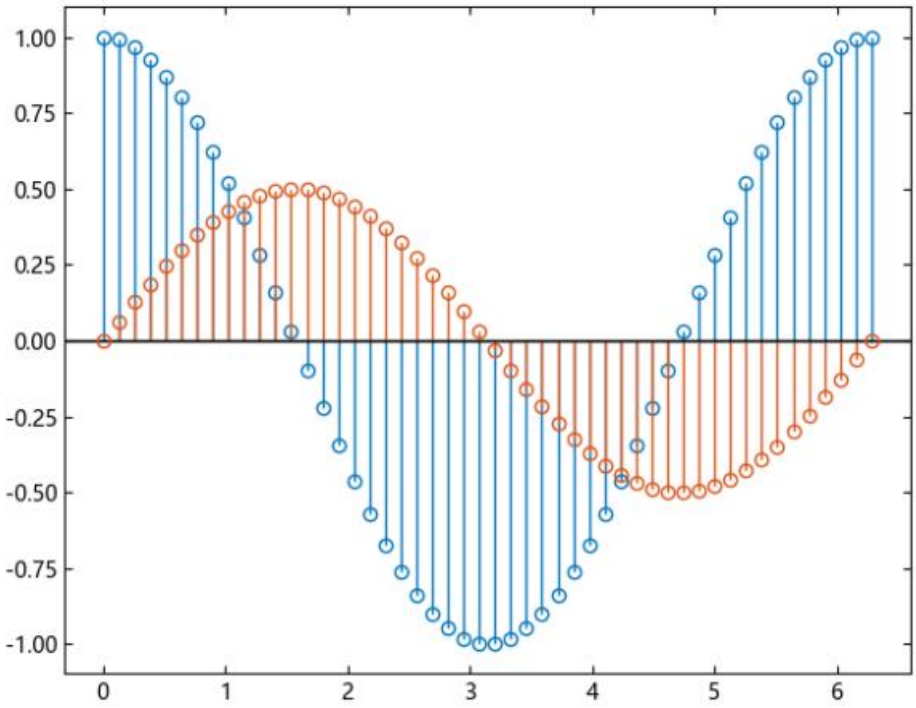


第一个向量输入确定每个针状线条在 x 轴上的位置。

(4). 在指定的 x 值处绘制多个数据序列

绘制在 0 和 2π 之间计算的 50 个正弦和余弦数据值并为针状图指定 x 值集。

```
using TyPlot
x = LinRange(0, 2 * pi, 50)'
y = cos.(x)
y1 = 0.5 .* sin.(x)
stem(x, y)
hold("on")
stem(x, y1)
hold("off")
```



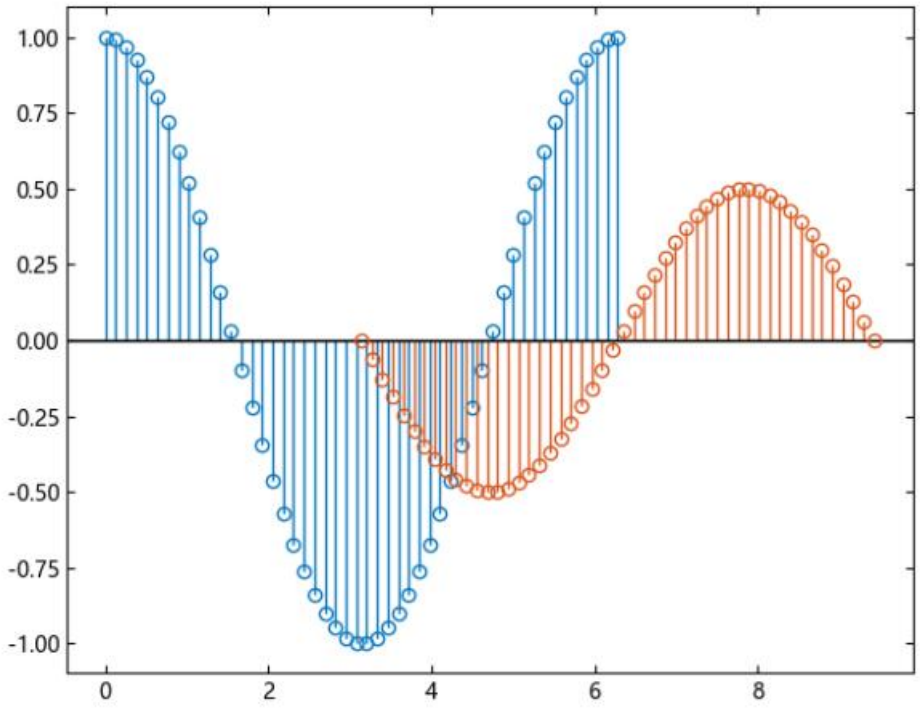
向量输入确定这两个数据序列的 x 轴位置。

(5). 在唯一的 x 值集合处绘制多个数据序列

绘制根据不同的 x 值集计算的 50 个正弦和余弦数据值。为每个序列指定对应的 x 值集。

```
using TyPlot

x1 = LinRange(0, 2 * pi, 50)'
x2 = LinRange(pi, 3 * pi, 50)
y1 = cos.(x1)
y2 = 0.5 .* sin.(x2)
stem(x1, y1)
hold("on")
stem(x2, y2)
hold("off")
```

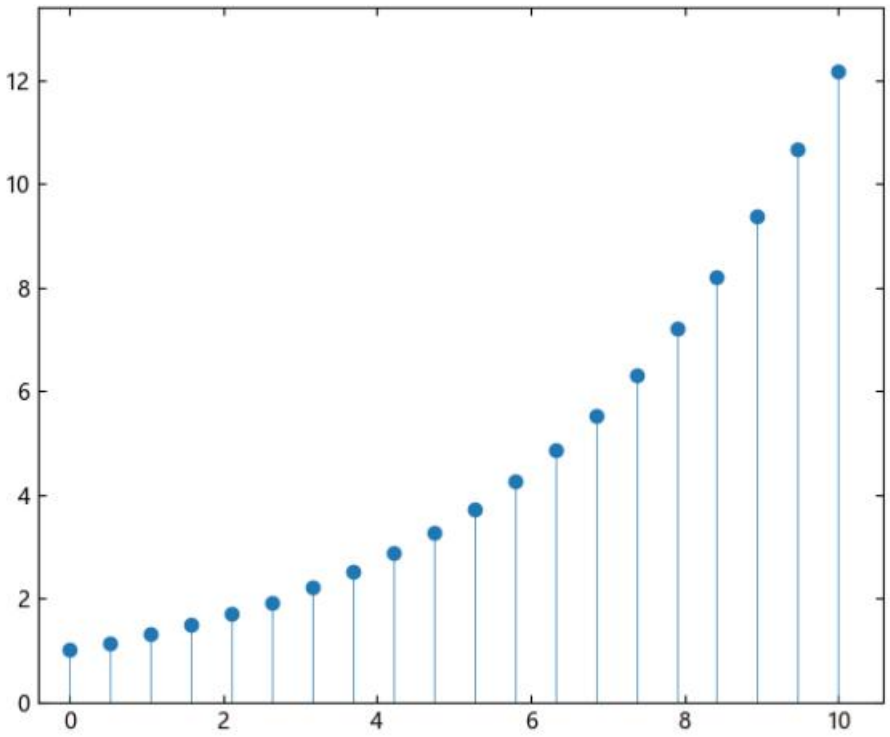


绘制 X 的每一列对 Y 的对应列的图。

(6). 填充图标记

创建一个针状图并填充终止每个针状图的圆。

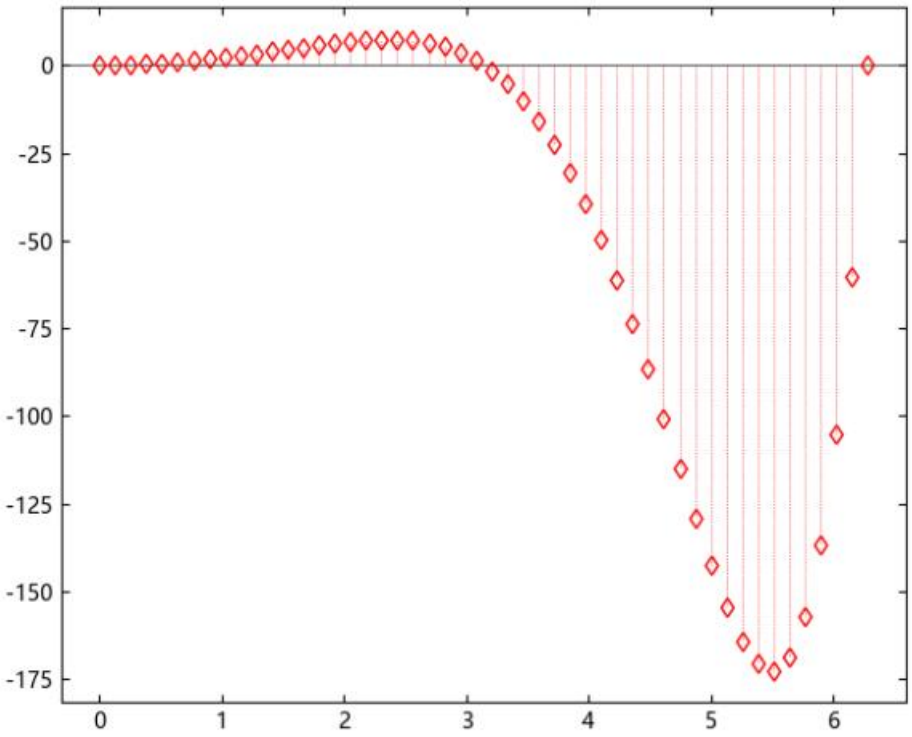
```
using TyPlot
X = LinRange(0, 10, 20)'
Y = (exp.(0.25 * X))
stem(X, Y; filled=true)
```



(7). 指定针状线条和标记选项

创建一个针状图并使用 fmt 选项将线型设置为点线，将标记符号设置为菱形，将颜色设置为红色。

```
using TyPlot
X = LinRange(0, 2 * pi, 50)';
Y = (exp.(X) .* sin.(X));
stem(X, Y; markerfmt="dr", linefmt=":r")
```

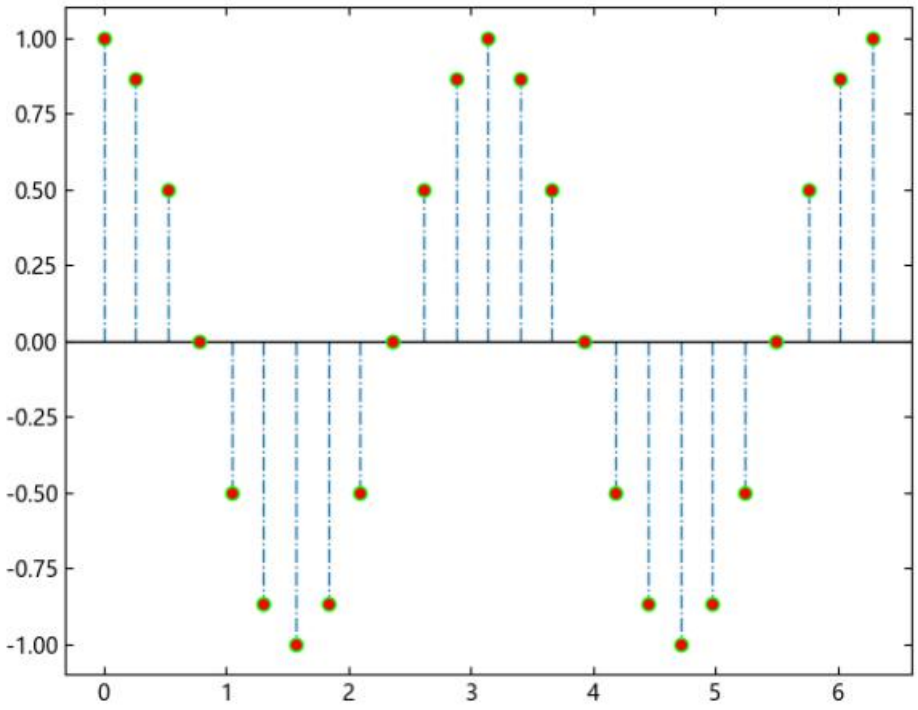


要对菱形内部着色，请使用 "fill" 选项。

(8). 指定其他针状线条和标记选项

创建一个针状图，并使用 Key=Value 对组参数将线型设置为点划线，将标记面颜色设置为红色，以及将标记边颜色设置为绿色。

```
using TyPlot
X = LinRange(0, 2 * pi, 25)'
Y = (cos.(2 * X))
lines = stem(X, Y; linefmt="-.", markerfacecolor="r",
markeredgecolor="#00FF00")
```



针状图仍保持默认颜色。

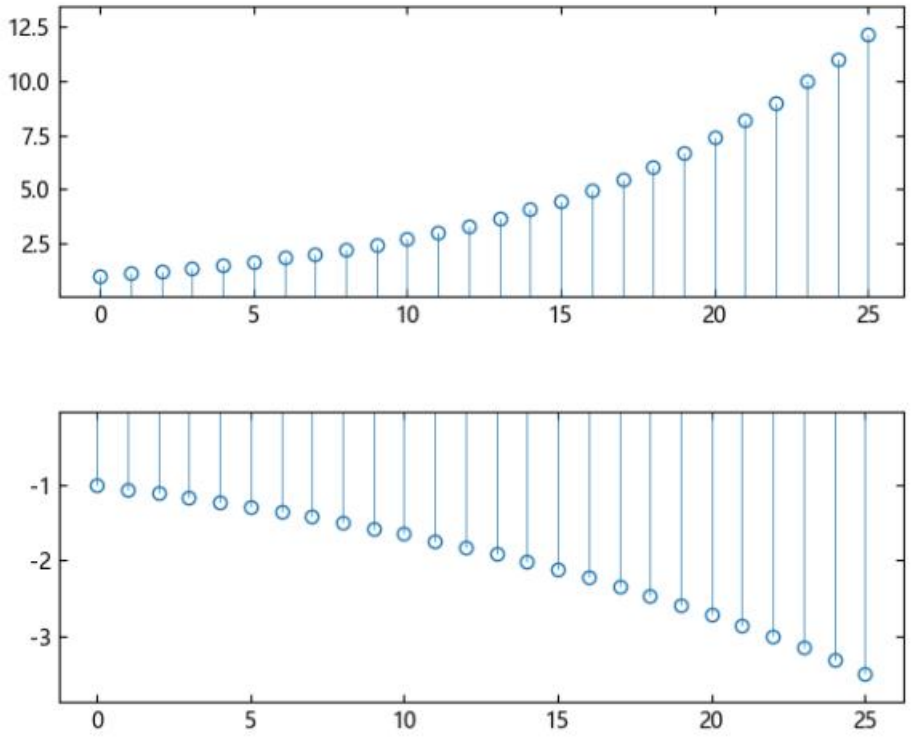
(9). 指定针状图的坐标区

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象 ax1 和 ax2。通过将坐标区对象指定为 stem 的第一个参数，在坐标区中创建单独的针状图。

```
using TyPlot
x = 0:25;
y1 = exp.(0.1 * x);
y2 = -exp.(0.05 * x);

ax1 = subplot(2, 1, 1)
ax2 = subplot(2, 1, 2)

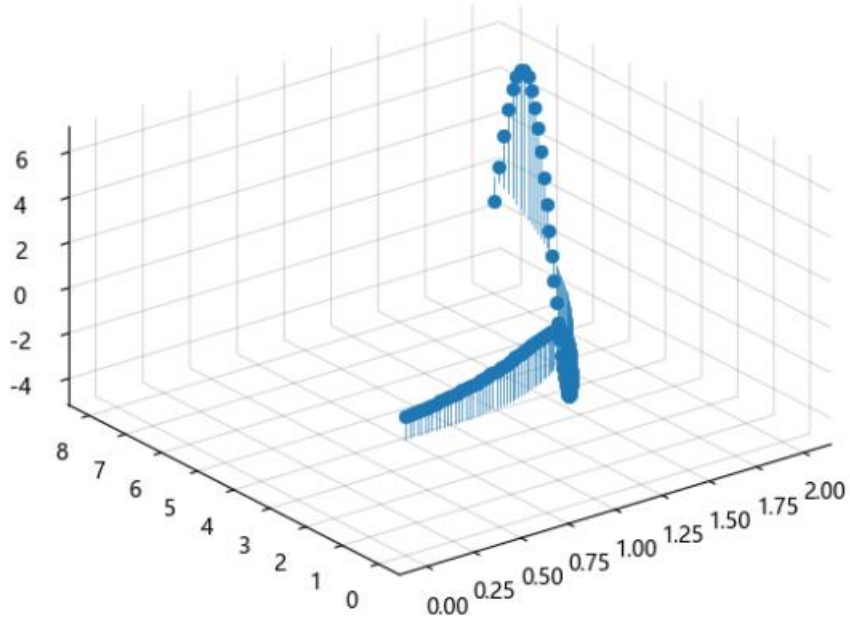
stem(ax1, x, y1)
stem(ax2, x, y2)
```



(10). 创建后修改针状序列

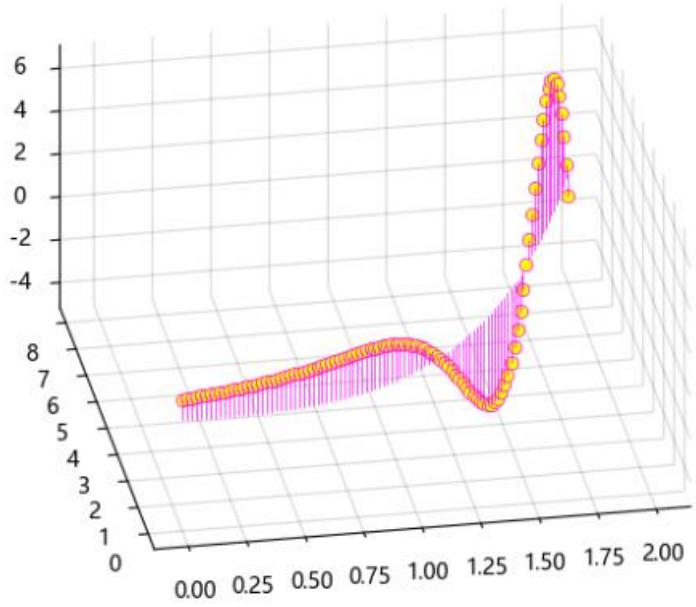
创建一个三维针状图并返回针状序列对象。

```
X = LinRange(0, 2, 100);
Y = X .^ 3;
Z = exp.(X) .* cos.(Y);
h = stem3(X, Y, Z; filled=true)
```



将颜色更改为品红色，将标记面颜色设置为黄色。使用 view 可调整图窗中坐标区的角度。使用圆点表示法设置属性。

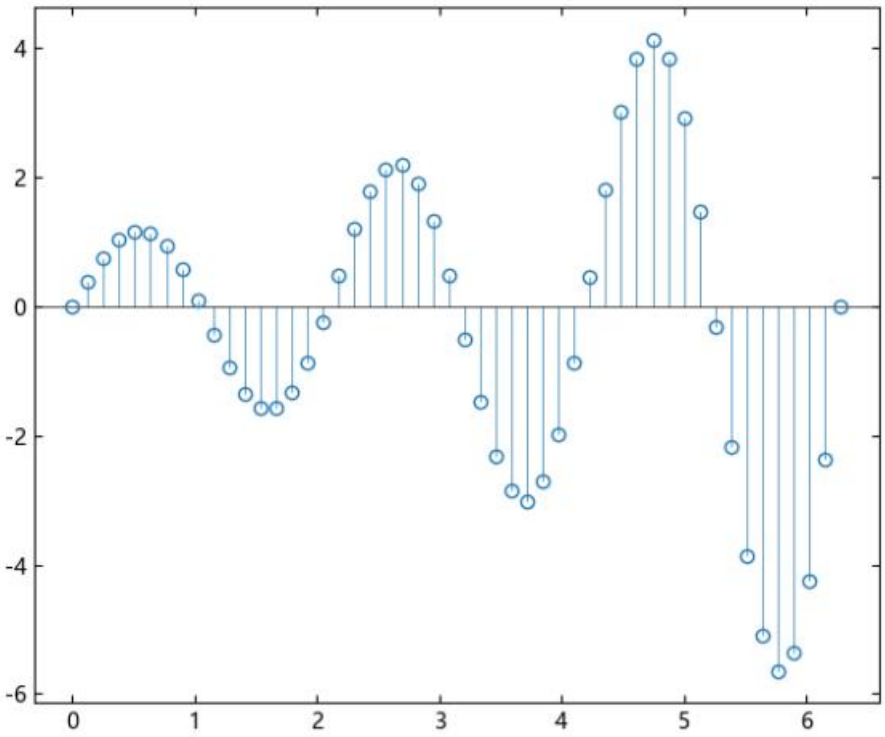
```
h[1].set_color([1, 0, 1])
h[2].set_color([1, 0, 1])
h[1].set_markerfacecolor([1, 1, 0])
plt_view(-10, 35)
```



(11). 调整基线属性

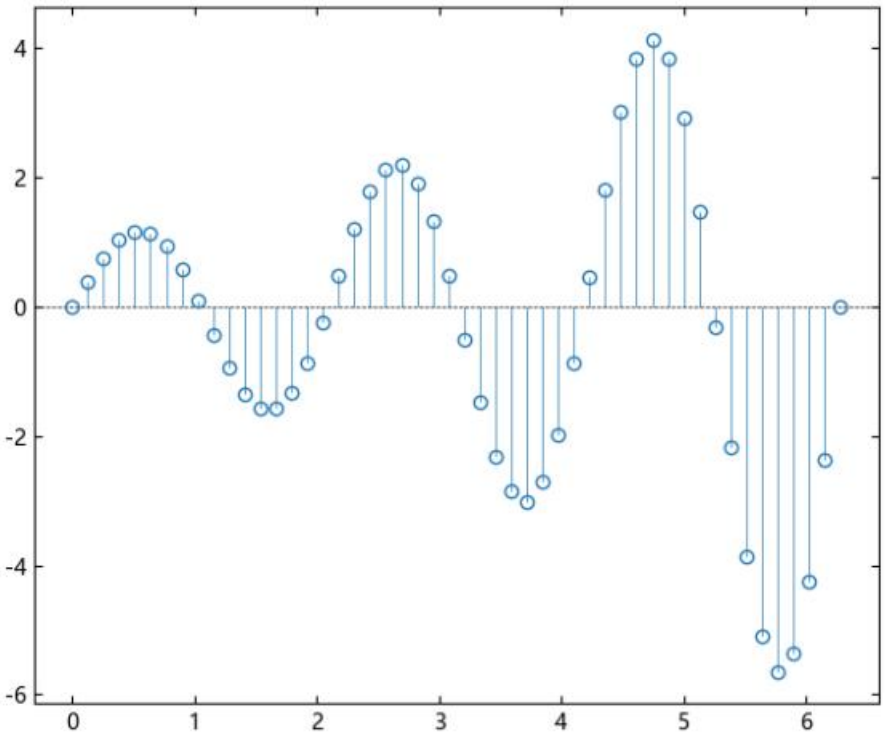
创建一个针状图并更改基线的属性。

```
using TyPlot
X = LinRange(0, 2 * pi, 50);
Y = exp.(0.3 * X) .* sin.(3 * X);
h = stem(X, Y);
```



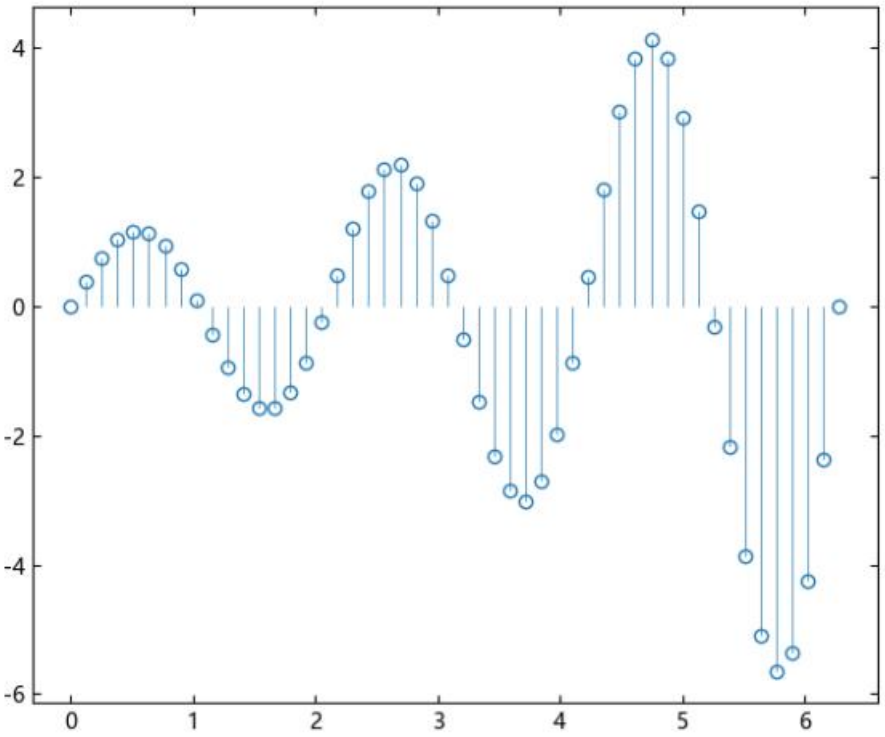
更改基线的线型。使用圆点表示法设置属性。

```
h[1][3].set_linestyle("--")
```



将基线的 Visible 属性设置为 false 以隐藏基线。

```
h[1][3].set_visible(false)
```

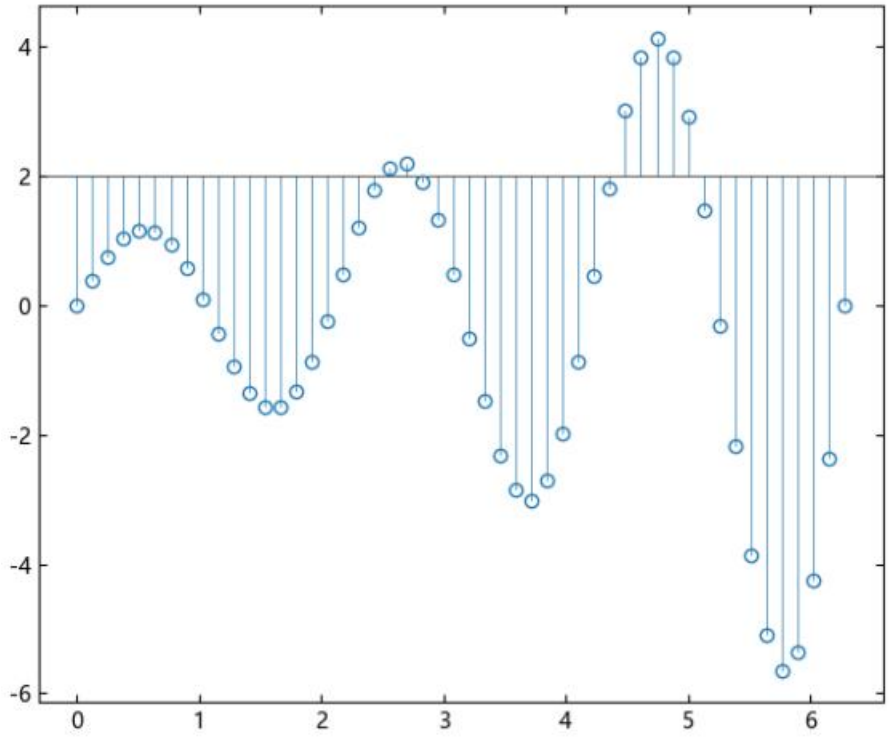


(12). 更改基线级别

创建一个针状图并将基线级别设置为 2。

```
using TyPlot
```

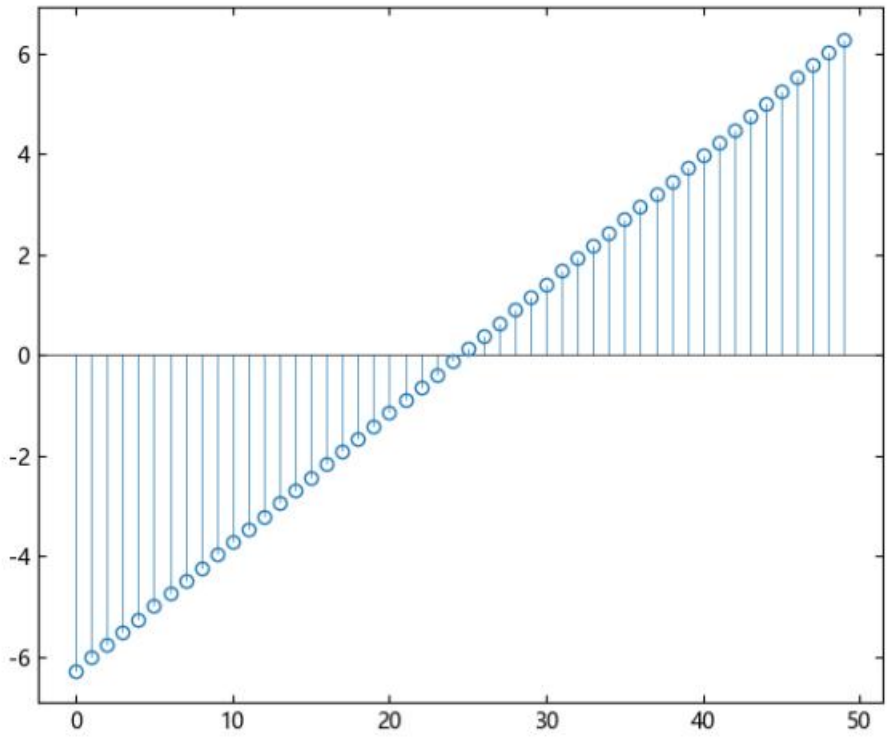
```
X = LinRange(0, 2 * pi, 50)';
Y = (exp.(0.3 * X) .* sin.(3 * X));
stem(X, Y; bottom=2);
```



绘制水平方向的针状图

创建一个包含 −2π 和 2π 之间的 50 个数据值，且在水平方向的针状图。

```
using TyPlot
Y = LinRange(-2 * pi, 2 * pi, 50);
stem(Y, orientation="horizontal")
```



数据值绘制为从基线开始延伸并在数据值位置终止的针状线条。Y 的长度自动确定每个针状图在 x 轴上的位置。

stem3

绘制三维离散序列数据

A. 语法

```
stem3(Z)
stem3(X,Y,Z)
stem3(___,Key=Value)

stem3(ax,___)

h = stem3(___)
```

B. 说明

stem3(Z)将 Z 中的各项绘制为针状图，这些针状图从 xy 平面开始延伸并在各项值处以圆圈终止。xy 平面中的针状线条位置是自动生成的。

stem3(X,Y,Z)将 Z 中的各项绘制为针状图，这些针状图从 xy 平面开始延伸，其中 X 和 Y 指定 xy 平面中的针状图位置。X、Y 和 Z 输入必须是大小相同的

向量或矩阵。

stem3(___,Key=Value)使用一个或多个名称-值对组参数修改针状图。

stem3(ax,___)将图形绘制到 ax 指定的坐标区中，而不是当前坐标区(gca)中。选项 ax 可以位于前面的语法中的任何输入参数组合之前。

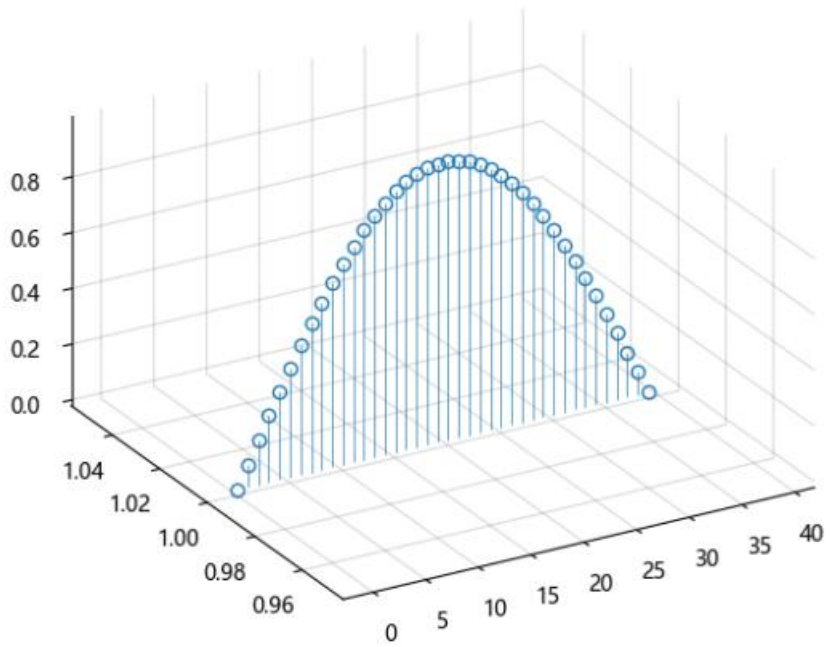
h=stem3(___)返回 Stem3 对象 h。

C. 示例

(1). 行向量输入

使用一个行向量输入创建在 −π/2 和 π/2 之间的余弦值的三维针状图。

```
figure()
X = LinRange(-pi / 2, pi / 2, 40);
Z = cos.(X);
stem3(Z)
```



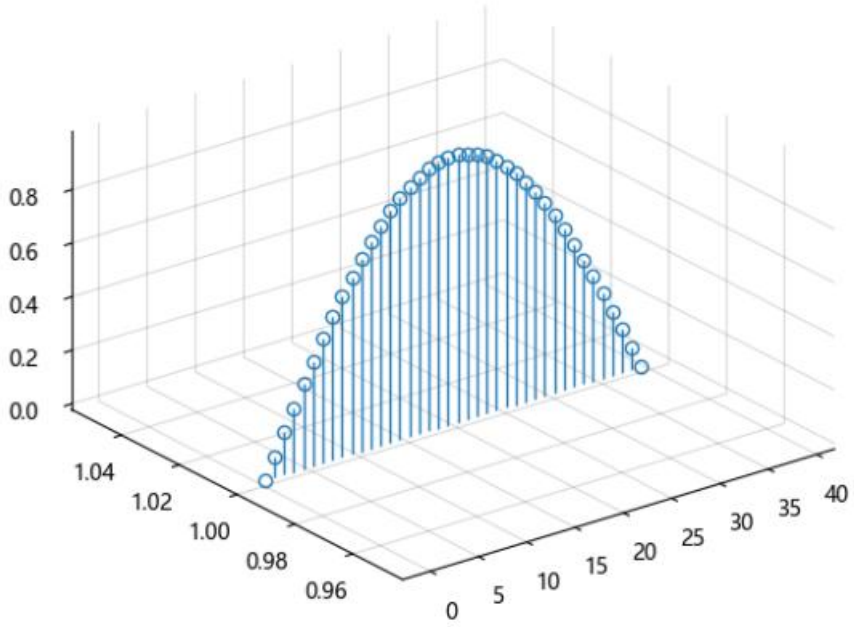
stem3 在等间距的 x 值位置绘制 Z 的元素对相同的 y 值的图。

(2). 列向量输入

使用一个列向量输入创建在 −π/2 和 π/2 之间的余弦值的三维针状图。

```
figure()
X = LinRange(-pi / 2, pi / 2, 40)';
Z = cos.(X);
```

```
stem3(Z)
```

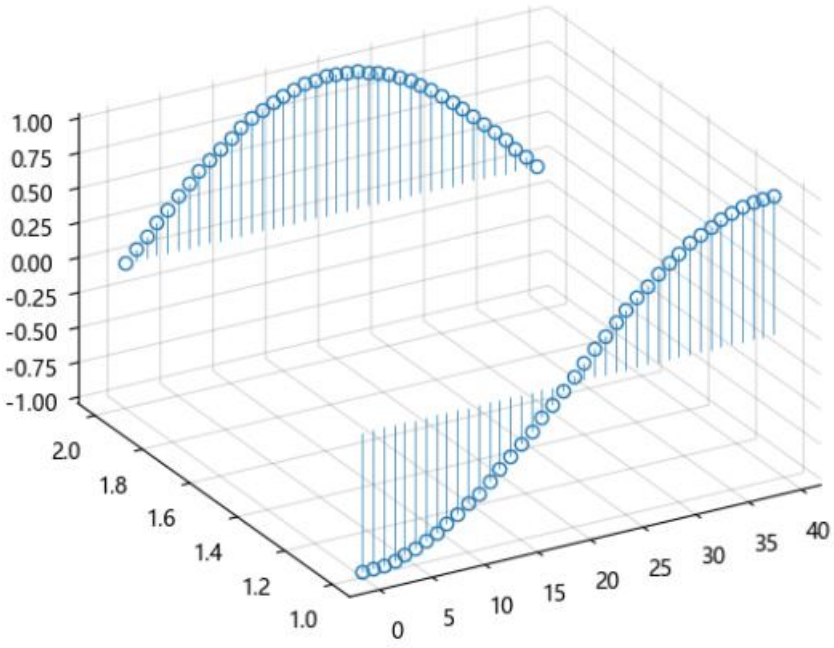


stem3 在等间距的 y 值位置绘制 Z 的元素对相同的 x 值的图。

(3). 矩阵输入

使用一个矩阵输入创建在 $-\pi/2$ 和 $\pi/2$ 之间的正弦和余弦值的三维针状图。

```
figure()
X = LinRange(-pi / 2, pi / 2, 40)';
Z = [sin.(X); cos.(X)];
stem3(Z)
```

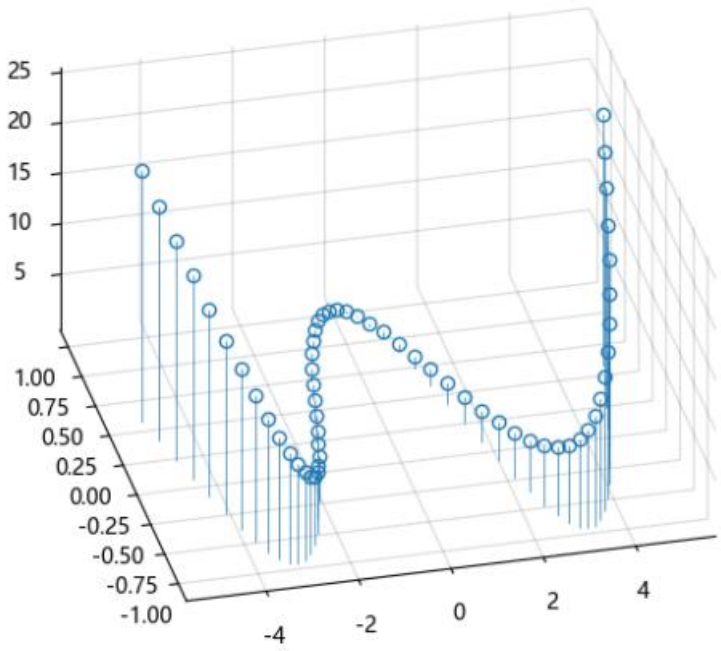


stem3 在等间距的 x 值位置绘制 Z 的每一行对相同的 y 值的图。

(4). 使用向量输入指定针状线条位置

创建一个三维针状图并指定针状线条沿曲线的位置。

```
figure()
X = LinRange(-5, 5, 60);
Y = cos.(X);
Z = X .^ 2;
stem3(X, Y, Z)
plt_view(-8, 30)
```



X 和 Y 确定针状线条位置。Z 确定标记高度。

(5). 填充标记

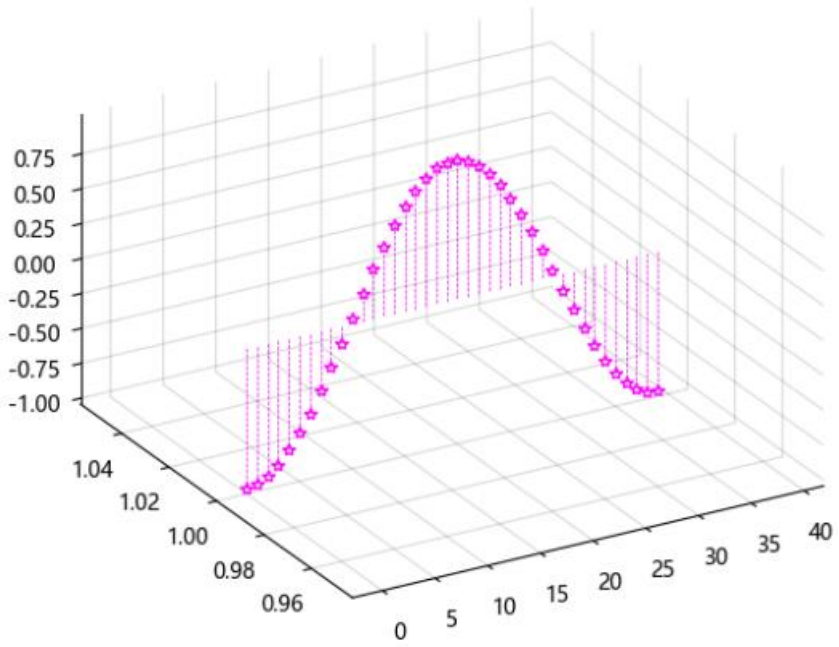
创建在 −π 和 π 之间的余弦值的三维针状图并填充标记。

```
X = LinRange(-pi, pi, 40);
Z = cos.(X);
stem3(Z; filled=true)
```

(6). 线型、标记符号和颜色选项

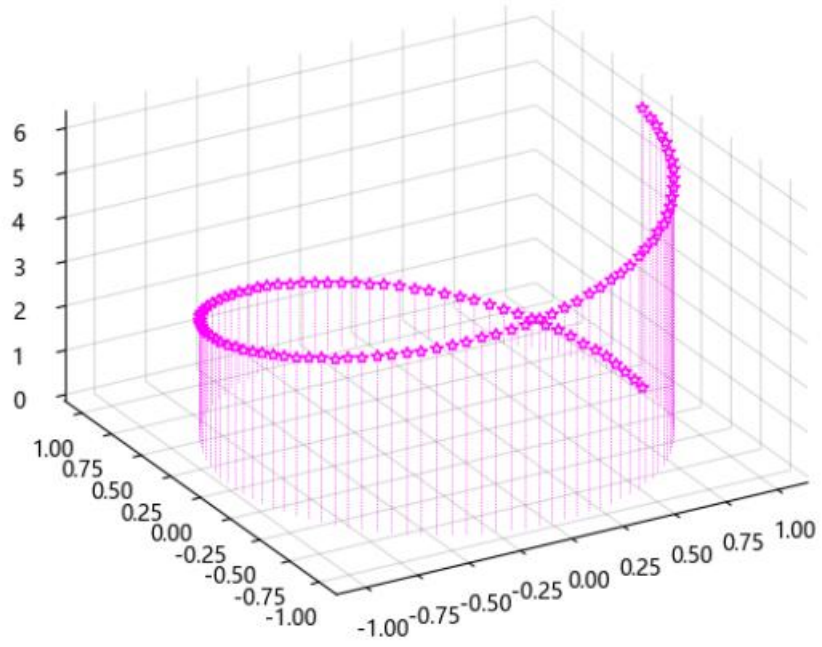
创建在 −π 和 π 之间的余弦值的三维针状图。对针状图使用虚线线型，将标记符号设置为星号，以及将颜色设置为品红色。

```
figure()
X = LinRange(-pi, pi, 40);
Z = cos.(X);
h = stem3(Z; linefmt="--m", markerfmt="*m")
```



创建一个三维针状图并指定针状线条沿圆的位置。将针状图设置为点线样式，将标记符号设置为星号，以及将颜色设置为品红色。

```
figure()
theta = LinRange(0, 2 * pi, 100);
X = cos.(theta);
Y = sin.(theta);
Z = theta;
h = stem3(X, Y, Z; linefmt=":m", markerfmt="*m")
```

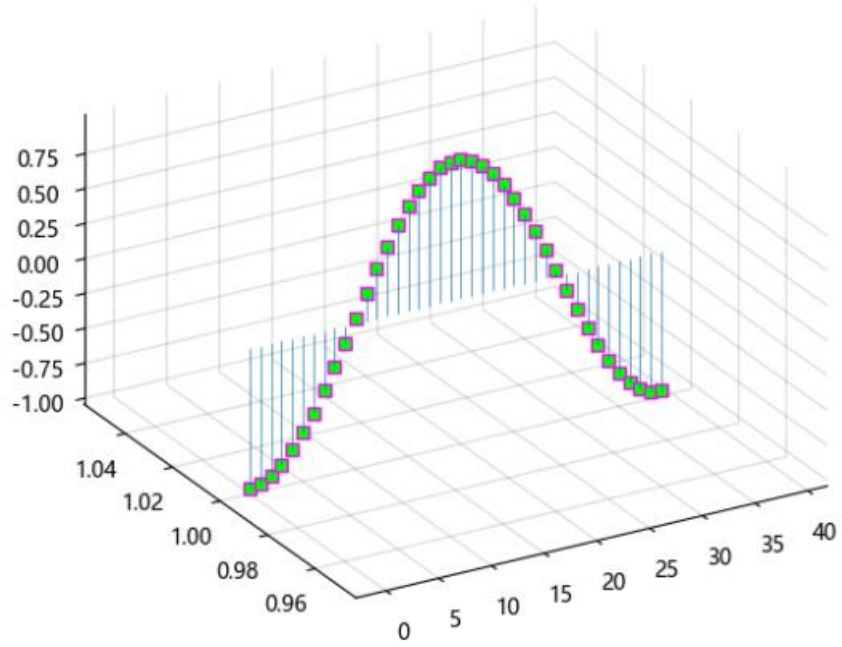


X 和 Y 确定针状线条位置。Z 确定标记高度。

(7). 其他样式选项

创建在 −π 和 π 之间的余弦值的三维针状图。将标记符号设置为具有绿色面和品红色边的方形。

```
figure()
X = LinRange(-pi, pi, 40);
Z = cos.(X);
stem3(Z;          markerfmt="s",          markeredgecolor="#FF00FF",
markerfacecolor=[0, 1, 0])
```



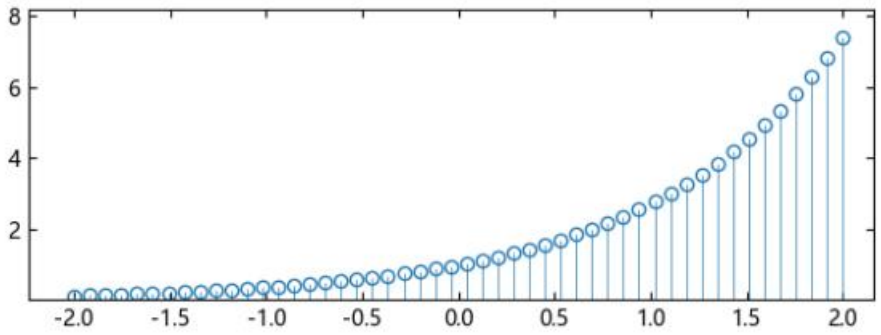
(8). 特定坐标区上的针状图

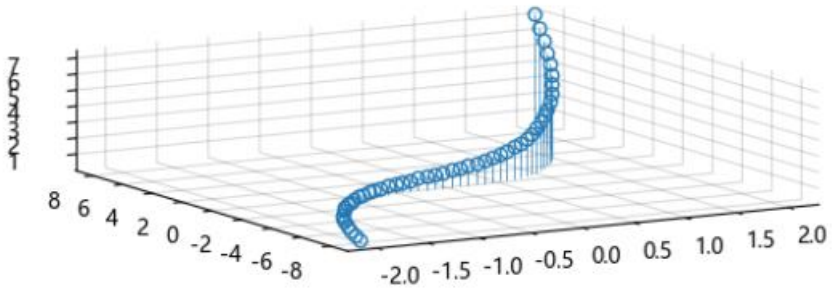
调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象 ax1 和 ax2。通过将坐标区对象指定为 stem3 的第一个参数，在坐标区中创建单独的针状图。

```
X = LinRange(-2, 2, 50);
Y = X .^ 3;
Z = exp.(X);

ax1 = subplot(2, 1, 1);
stem(ax1, X, Z)
```

```
ax2 = subplot(2, 1, 2; projection="3d");
stem3(ax2, X, Y, Z)
```

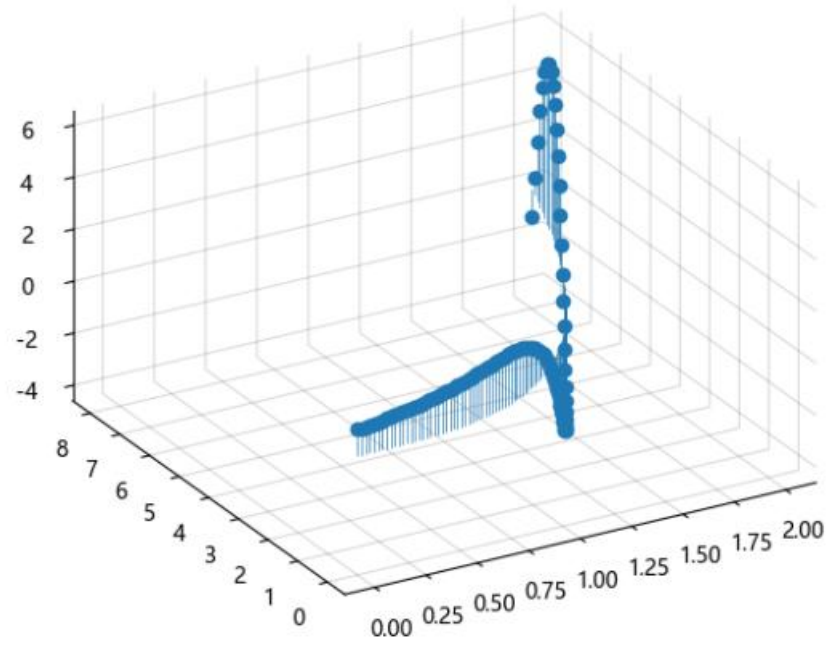




(9). 创建后修改针状序列

创建一个三维针状图并返回针状序列对象。

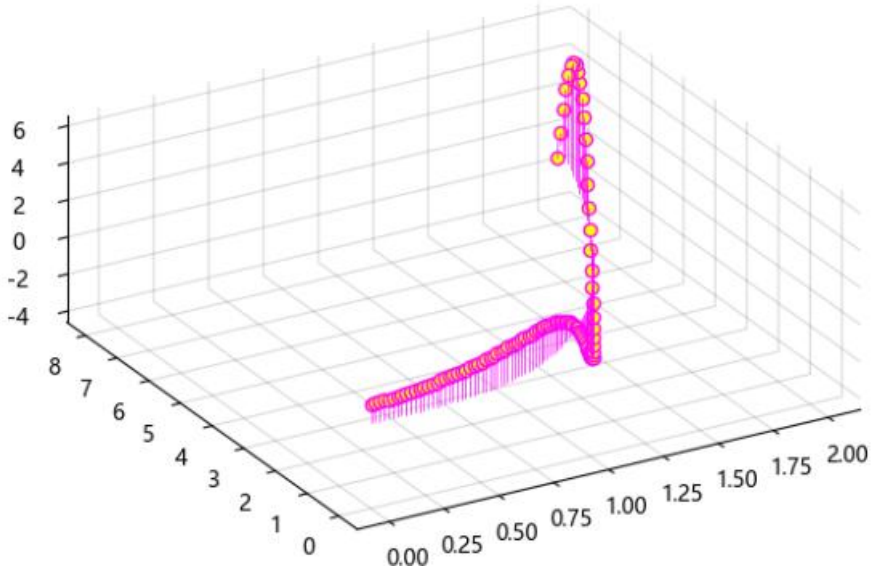
```
X = LinRange(0, 2, 100);
Y = X .^ 3;
Z = exp.(X) .* cos.(Y);
h = stem3(X, Y, Z; filled=true);
```



将颜色更改为品红色，将标记面颜色设置为黄色。使用圆点表示法设置属性。

```
h[2].set_color("#FF00FF")
h[1].set_markeredgecolor("#FF00FF")
h[1].set_markerfacecolor("#FFFF00")

gca().set_box_aspect((2, 2, 1))
```



3.3.1.3.3 散点图

函数名	简介
scatter	散点图
scatter3	三维散点图

scatter

详情见图形 API->二维和三维图->数据分布图->散点图->scatter

scatter3

详情见图形 API->二维和三维图->数据分布图->散点图->scatter3

3.3.1.3.4 阶梯图

数名	简介
stairs	阶梯图

stairs

详情见图形 API->二维和三维图->线图->线图->stairs

3.3.1.4 极坐标图

在极坐标中绘图

3.3.1.4.1 对数据绘图

函数名	简介
polar	极坐标图
polarplot	在极坐标中绘制线条
polarscatter	极坐标中的散点图
polarhistogram	极坐标中的直方图
ezpolar	易用的极坐标绘图函数

polar

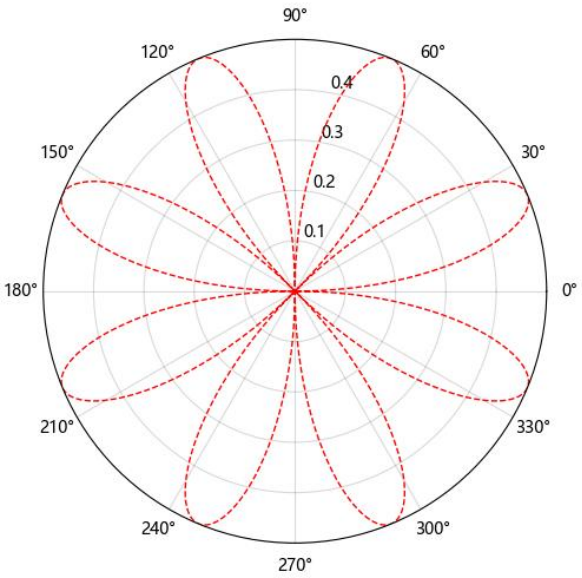
极坐标图

A. 示例

简单极坐标图

使用红色虚线创建简单的极坐标图。

```
using TyPlot
theta = 0:0.01:(2 * pi);
rho = sin.(2 * theta) .* cos.(2 * theta);
polarplot(theta, rho, "--r")
```



polarplot

在极坐标中绘制线条

B. 语法

```
Polarplot(theta,rho)
polarplot(theta,rho,fmt)
polarplot(theta1,rho1,...,thetaN,rhoN)
polarplot(theta1,rho1,fmt1,...,thetaN,rhoN,fmtN)

polarplot(rho)
polarplot(rho,fmt)
```

```
polarplot(Z)
polarplot(Z,fmt)

polarplot(___,Key=Value)
polarplot(pax,___)
p = ppolarplot(___)
```

C. 说明

polarplot(theta,rho)在极坐标中绘制线条，由 theta 表示弧度角，rho 表示每个点的半径值。输入必须是长度相等的向量或大小相等的矩阵。如果输入为矩阵，polarplot 将绘制 rho 的列对 theta 的列的图。也可以一个输入为向量，另一个为矩阵，但向量的长度必须与矩阵的一个维度相等。

polarplot(theta,rho,fmt)设置线条的线型、标记符号和颜色。

polarplot(theta1,rho1,...,thetaN,rhoN) 绘制多个 rho,theta 对组。

polarplot(theta1,rho1,fmt1,...,thetaN,rhoN,fmtN) 指定每个线条的线型、标记符号和颜色。

polarplot(rho)按等间隔角度（介于 0 和 2 π 之间）绘制 rho 中的半径值。示例

polarplot(rho,fmt) 设置线条的线型、标记符号和颜色。

polarplot(Z)绘制 Z 中的复数值。

polarplot(Z,fmt) 设置线条的线型、标记符号和颜色。

polarplot(___,Key=Value) 使用一个或多个 Key=Value 对组参数指定图形线条的属性。属性设置适用于所有线条。无法使用 Key=Value 对组为不同的线条指定不同的属性值。

polarplot(pax,___) 使用 pax 指定的 PolarAxes 对象，而不是使用当前坐标区。

p = polarplot(___) 返回一个或多个图形线条对象。在创建图形线条对象之后使用 p 为其设置属性。

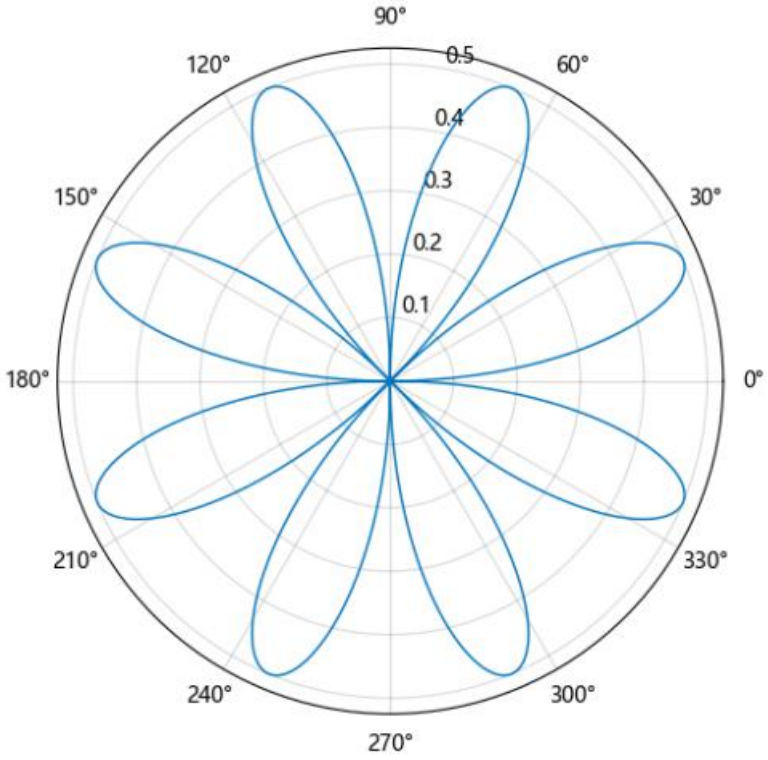
D. 示例

(1). 创建极坐标图

在极坐标中绘制线条。

```
using TyPlot
theta = 0:0.01:(2 * pi);
```

```
rho = sin.(2 * theta) .* cos.(2 * theta);
p = polarplot(theta, rho)
```



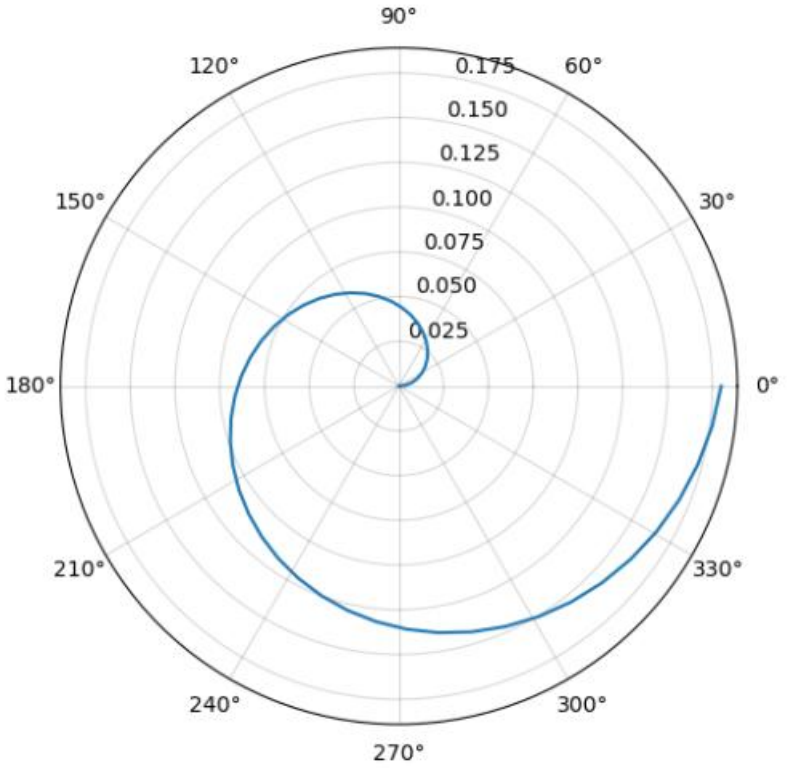
(2). 绘图之前从度转换为弧度

创建要绘图的数据。

```
using TyPlot
theta = LinRange(0, 360, 50);
rho = 0.005 .* theta ./ 10;
```

将 theta 中的值从度转换为弧度。然后在极坐标中绘制数据。

```
theta_radians = deg2rad.(theta)
polarplot(theta_radians, rho)
```

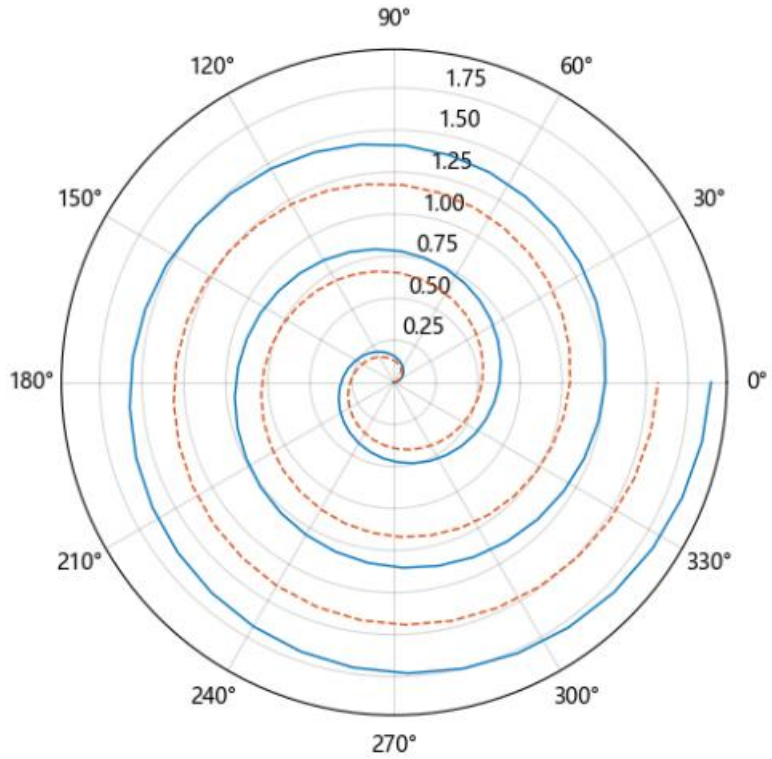


(3). 在极坐标中绘制多个线条

在极坐标中绘制两个线条。第二个线条使用虚线。

```
using TyPlot
theta = LinRange(0, 6 * pi, 100);
rho1 = theta ./ 10;
rho2 = theta ./ 12;

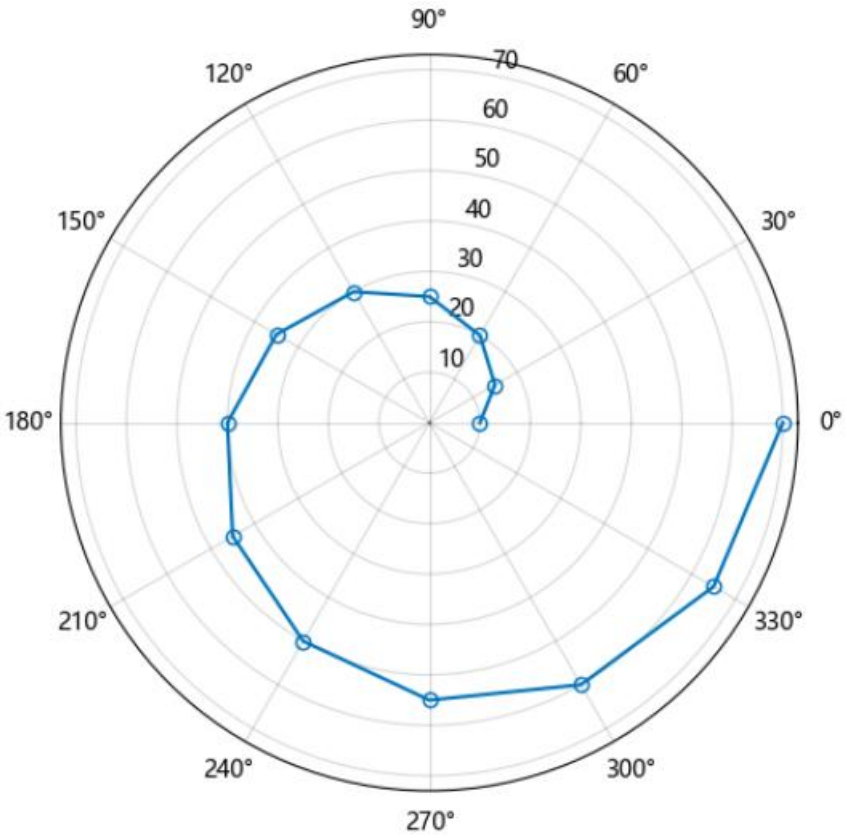
polarplot(theta, rho1)
hold("on")
polarplot(theta, rho2, "--")
hold("off")
```



(4). 按等间隔角度绘制半径值

只指定半径值，不指定角度值。polarplot 将按照等间隔角度（从 0 到 2π）绘制半径值。在每个数据点处显示一个圆形标记。

```
using TyPlot
rho = 10:5:70;
polarplot(rho, "-o")
```

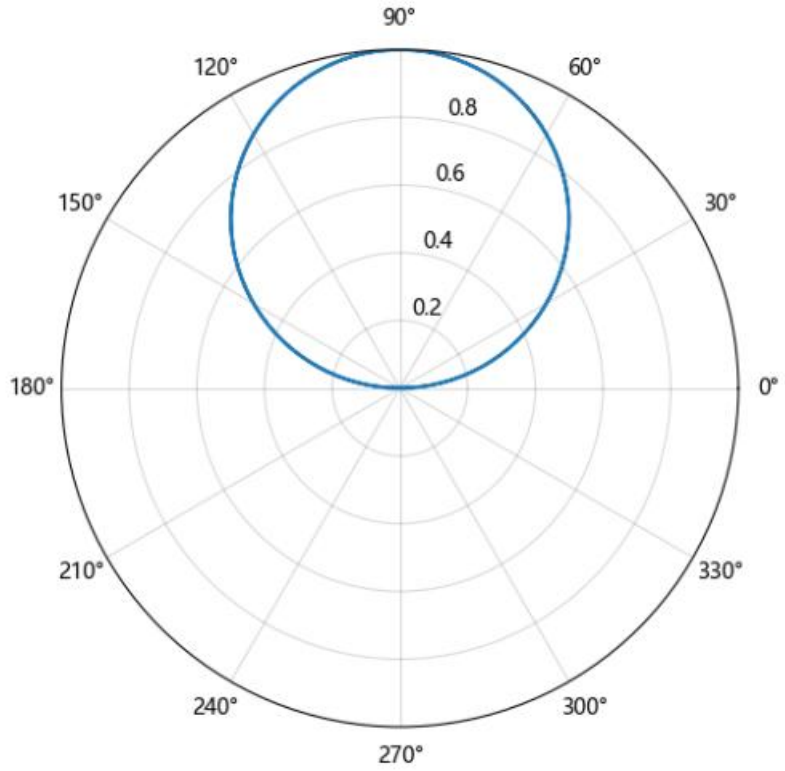


(5). 绘制负半径值

使用负半径值创建极坐标图。默认情况下，polarplot 会将负值反射穿过原点。

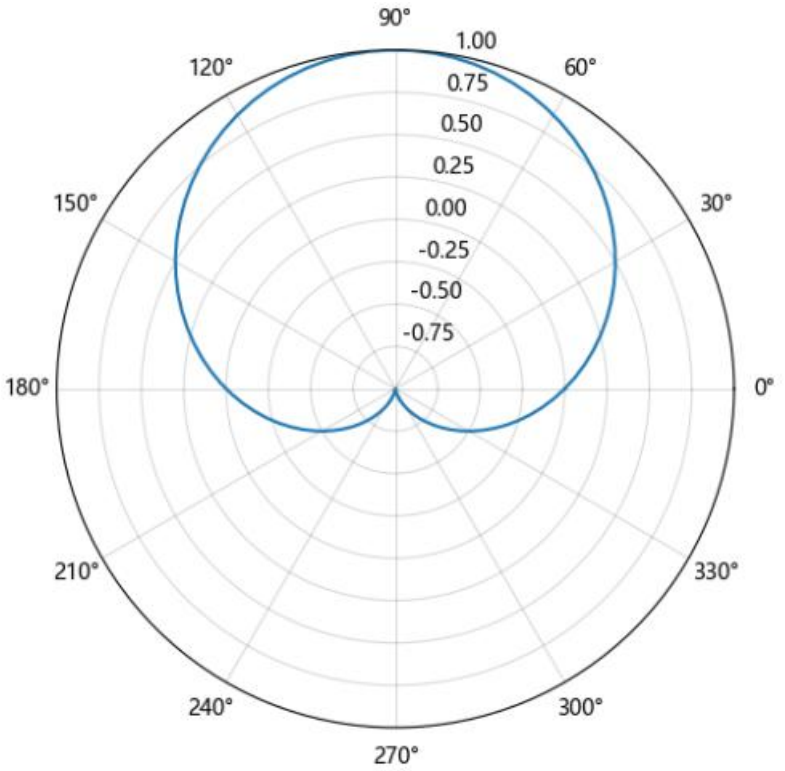
```
using TyPlot

theta = LinRange(0, 2 * pi, 100);
rho = sin.(theta);
polarplot(theta, rho)
```



更改 r 轴的范围，使其从 -1 到 1。

```
polarplot(theta, rho; rlim=(-1, 1))
```

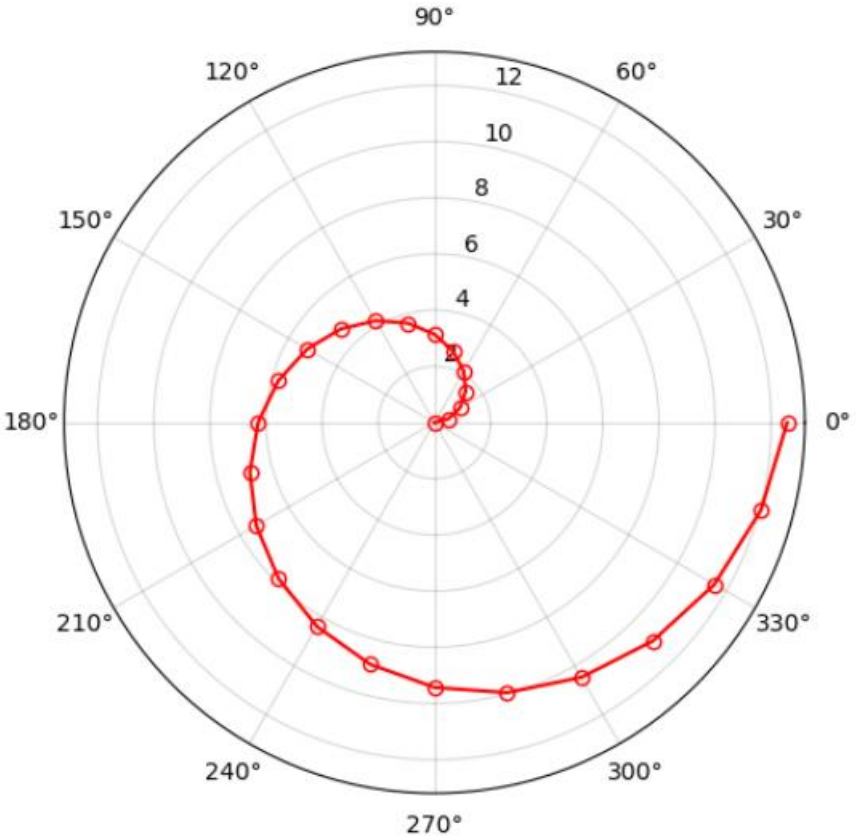


(6). 指定极坐标图的线条颜色

使用带有圆形标记的红色线条创建一个极坐标图。

```
using TyPlot

theta = LinRange(0, 2 * pi, 25);
rho = 2 .* theta;
polarplot(theta, rho, "r-o")
```

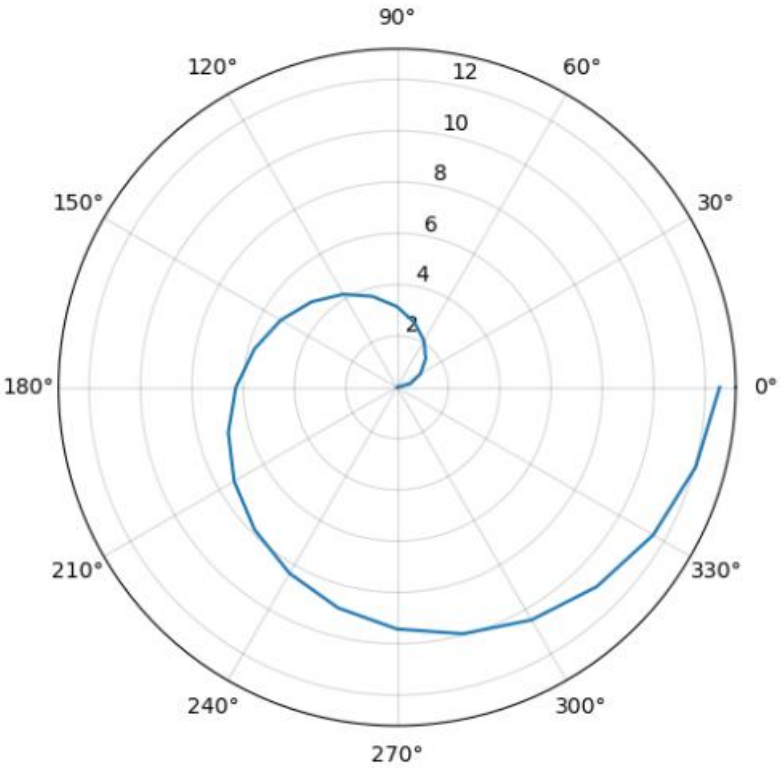


(7). 创建后指定线条颜色

创建一个极坐标图并返回图形线条对象。

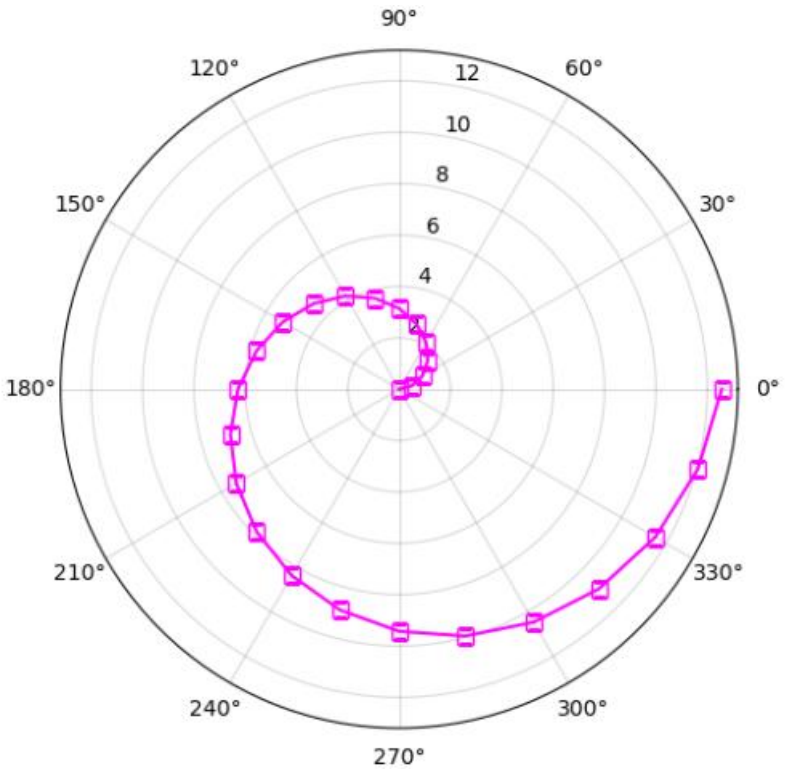
```
using TyPlot

theta = LinRange(0, 2 * pi, 25);
rho = 2 .* theta;
p = polarplot(theta, rho)
```



更改线条颜色和宽度并添加标记。

```
p[1].set_color("magenta");
p[1].set_marker("s");
p[1].set_markersize(7);
p[1].set_markerfacecolor("none")
```

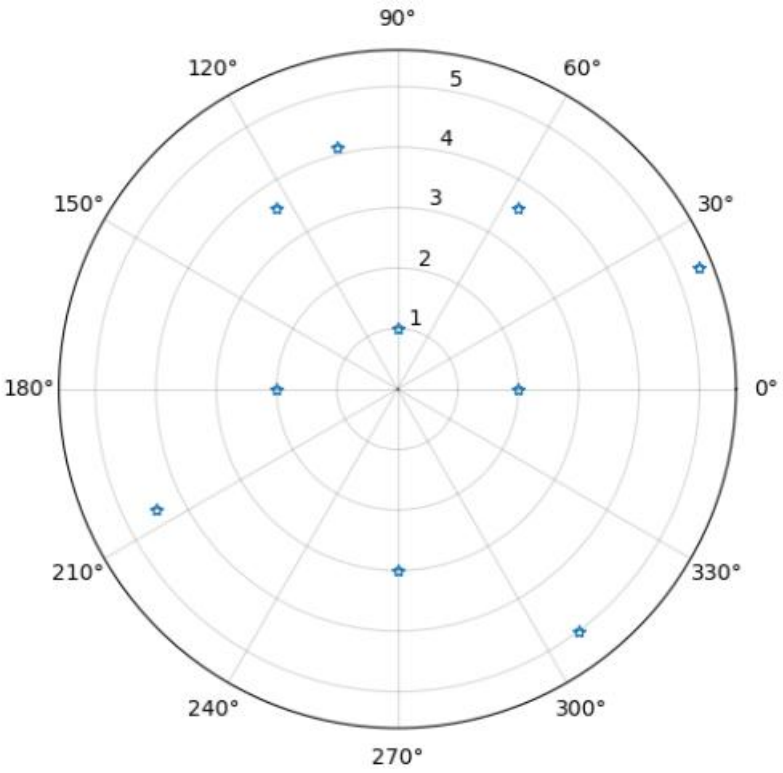


(8). 使用复数值创建极坐标图

在极坐标中绘制复数值。在每个点处显示标记，标记间无连接线。

```
using TyPlot

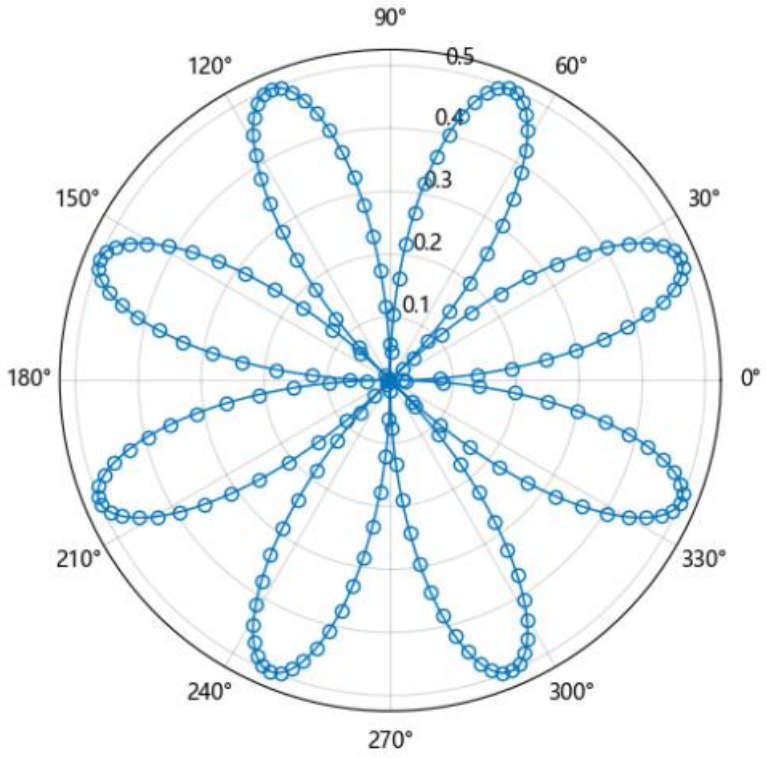
Z = [2 + 3im 2 -1 + 4im 3 - 4im 5 + 2im -4 - 2im -2 + 3im -2 -3im
3im - 2im];
theta = angle.(Z)
rho = abs.(Z)
p = polarplot(Z, "*")
```



(9). 在特定的数据点显示标记

在极坐标中绘制线条，并将 markevery 属性设置为名称-值对组的形式，创建一个三维线图并每隔两个数据点显示一个标记。

```
using TyPlot
theta = 0:0.01:(2 * pi);
rho = sin.(2 * theta) .* cos.(2 * theta);
p = polarplot(theta, rho)
polarplot(theta, rho; marker="o", markevery=1:3:length(theta))
```



polarscatter

极坐标中的散点图

A. 语法

```
polarscatter(th,r)
polarscatter(th,r,sz)
polarscatter(th,r,sz,c)

polarscatter(___,Key=Value)
polarscatter(pax,___)
ps = polarscatter(___)
```

B. 说明

polarscatter(th,r)绘制 th 对 r 的图，并在每个数据点显示一个圆圈。th 和 r 必须是具有相同长度的向量。必须以弧度为单位指定 th。

polarscatter(th,r,sz) 设置标记大小，其中 sz 以平方磅为单位指定每个标记的面积。要以相同的大小绘制所有标记，请将 sz 指定为标量。要以不同的大小绘制标记，请将 sz 指定为长度与 th 相同的向量。

polarscatter(th,r,sz,c)设置标记颜色，其中 c 是向量、三列矩阵、RGB 三元

组或颜色名称，例如"red"。

polarscatter(___,Key=Value) 使用一个或多个名称-值对组参数修改散点图的外观。例如，您可以指定 alpha 是一个介于 0 和 1 之间的标量值，从而使用半透明标记。

polarscatter(pax,___) 将在 pax 指定的极坐标区（而不是当前坐标区）中绘制图形。

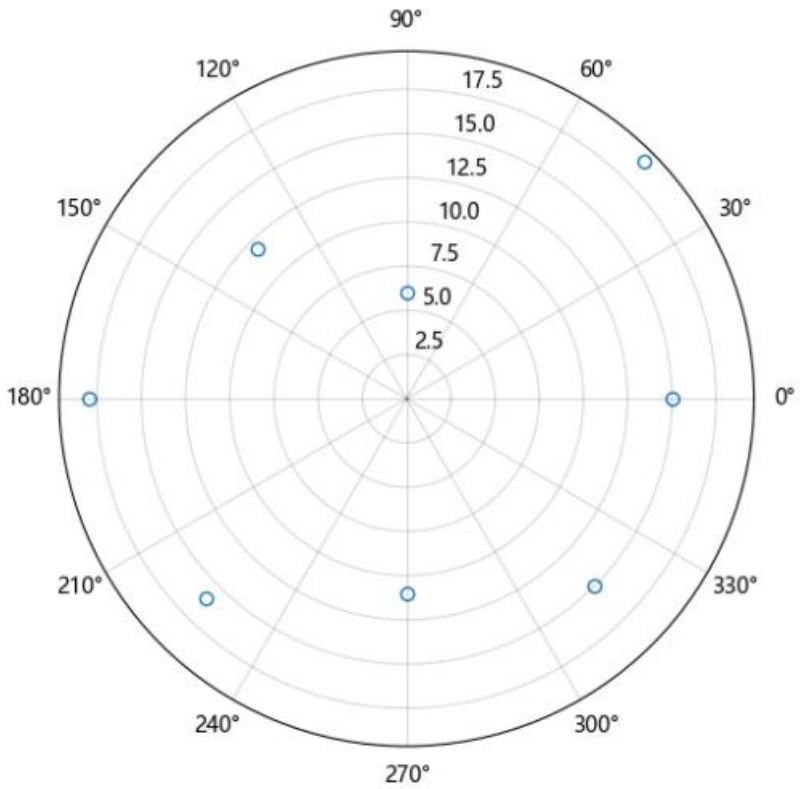
ps = polarscatter(___) 返回 Scatter 对象。在创建 Scatter 对象之后可使用 ps 修改其外观。有关属性列表，请参阅 Scatter 属性 (opens new window)。

C. 示例

(1). 创建散点图

在极坐标中创建一个散点图。

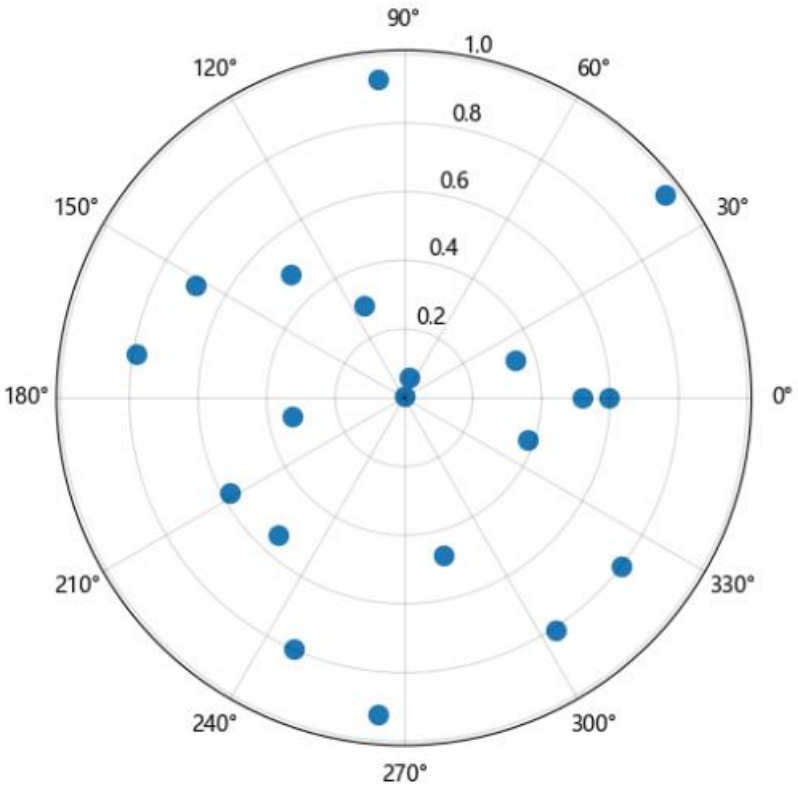
```
th = pi/4:pi/4:2*pi;
r = [19 6 12 18 16 11 15 15];
s = polarscatter(th,r)
```



(2). 使用已填充标记并设置标记大小

通过指定可选的输入参数 filled = true，创建一个使用已填充标记的散点图。将标记大小设置为 75 平方磅。

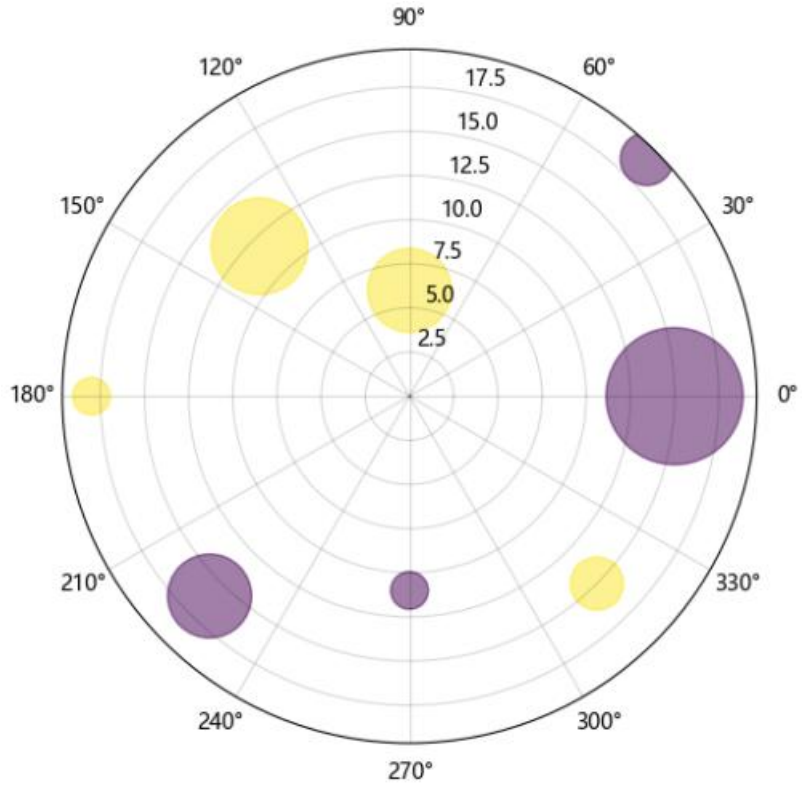
```
using TyPlot
th = LinRange(0, 2 * pi, 20);
r = rand(1,20);
sz = 75;
polarscatter(th,r,sz,filled=true)
```



(3). 使用具有不同大小和颜色的标记

使用具有不同大小和颜色的标记创建散点图。将可选的大小和颜色输入参数指定为向量。使用颜色向量中的唯一性值指定所需的不同颜色。这些值映射到颜色图中的不同颜色。

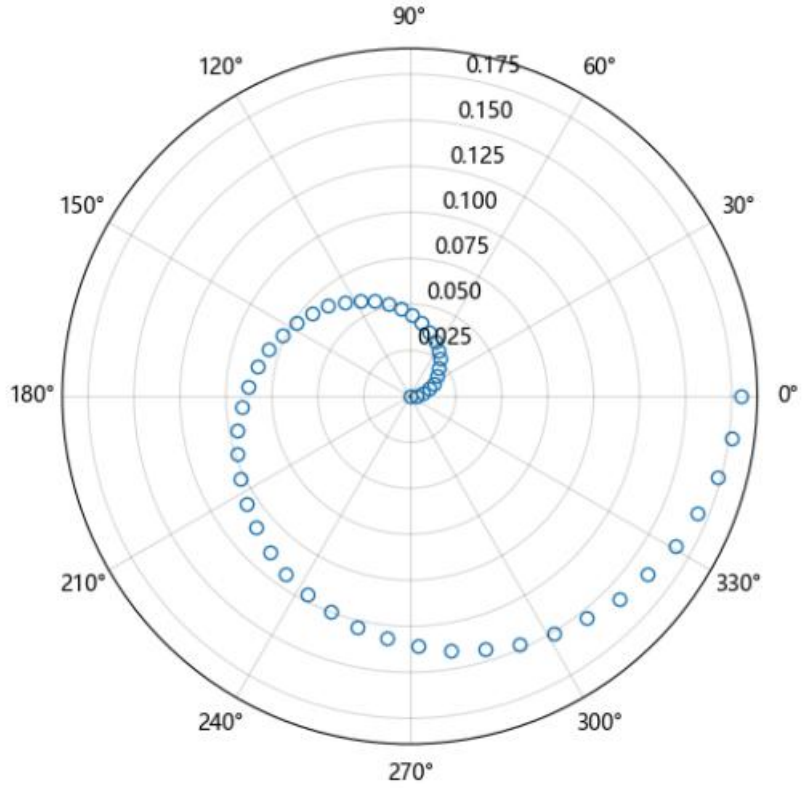
```
th = pi/4:pi/4:2*pi;
r = [19 6 12 18 16 11 15 15];
sz = 100*[6 15 20 3 15 3 6 40];
c = [1 2 2 2 1 1 2 1];
s = polarscatter(th, r, sz, c, filled=true, alpha=.5)
```



(4). 绘图之前从度转换为弧度

创建角度值以度为单位的数据。由于 polarscatter 要求角度值以弧度为单位，因此要在绘制之前使用 deg2rad 将值转换为弧度。

```
using TyPlot
th = LinRange(0, 360, 50);
r = 0.005 * th / 10;
th_radians = deg2rad.(th);
polarscatter(th_radians, r)
```



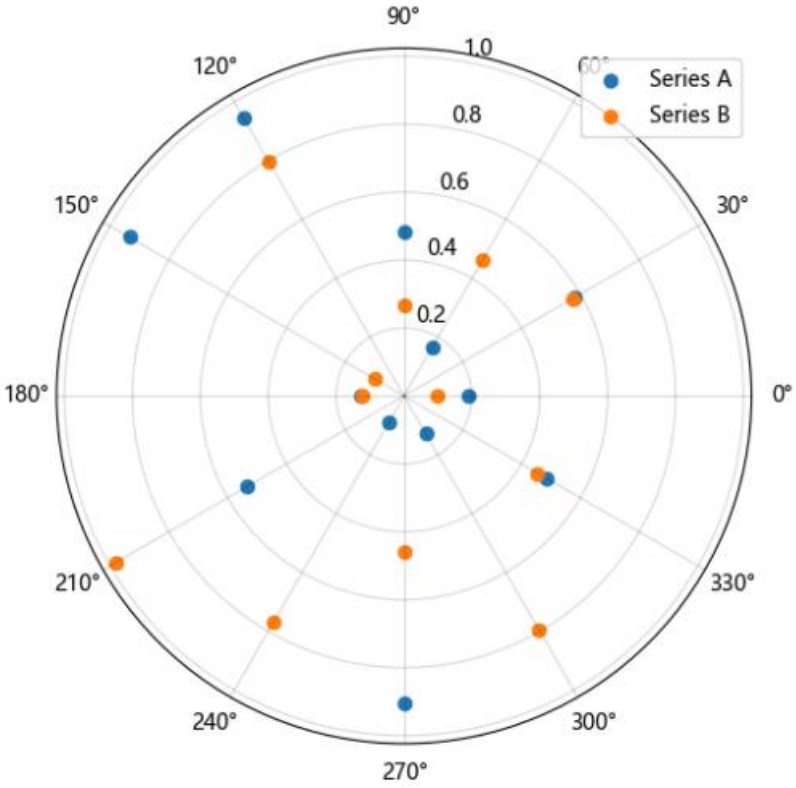
(5). 合并两个散点图

使用 hold 命令将位于相同极坐标区中的两个散点图合并起来。添加包含每个图的说明信息的图例。

```
th = pi/6:pi/6:2*pi;
r1 = rand(12,1);  # 返回由正态分布的随机数组成的 12×1 矩阵
polarscatter(th,r1,filled=true)

hold("on")
r2 = rand(12,1);  # 返回由正态分布的随机数组成的 12×1 矩阵
polarscatter(th,r2,filled=true)
hold("off")

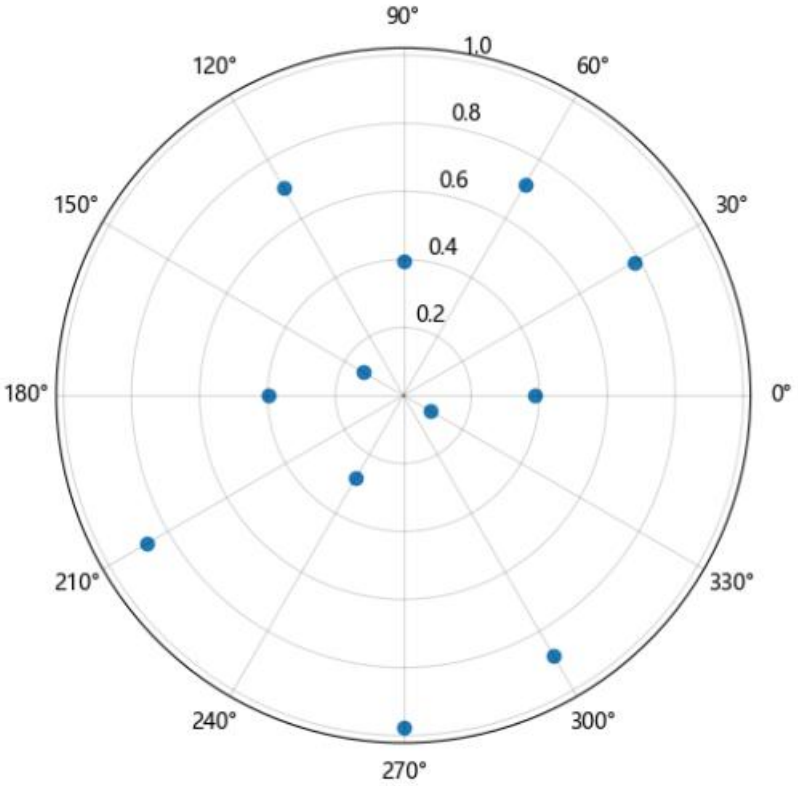
legend(["Series A","Series B"])
```



(6). 创建散点图之后进行修改

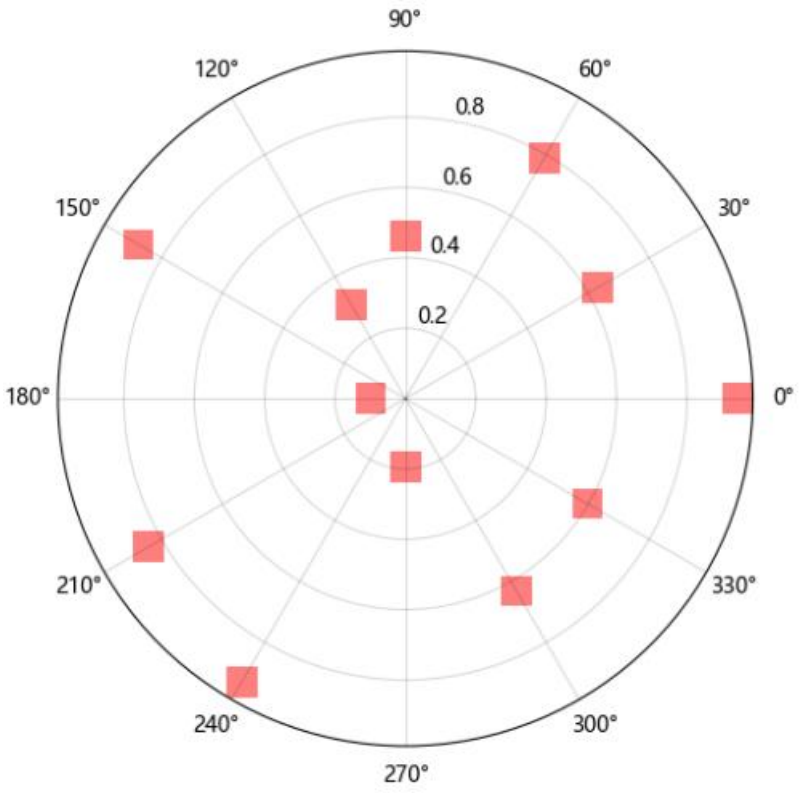
创建一个散点图，并将散点图对象赋给变量 ps。

```
th = pi/6:pi/6:2*pi;
r = rand(12,1); # 返回由正态分布的随机数组成的 12×1 矩阵
ps = polarscatter(th,r,filled=true)
```



在创建散点图对象之后可使用 ps 修改其属性。

```
ps = polarscatter(th,r,200,filled=true, marker="s")
ps.set_facecolor("red")
ps.set_edgecolor(nothing)
ps.set_alpha(.5)
```



tpolarhistogram

极坐标中的直方图

A. 语法

```
polarhistogram(theta)
polarhistogram(theta,nbins)
polarhistogram(theta,edges)

polarhistogram(___,Key=Value)
polarhistogram(pax,___)
h = polarhistogram(___)
```

B. 说明

polarhistogram(theta)通过将 theta 中的值分为等间距的 bin，在极坐标中创建一个直方图。指定弧度值。

polarhistogram(theta,nbins)使用正整数 nbins 指定的 bin 数目。

polarhistogram(theta,edges) 将 theta 划分为由向量 edges 指定 bin 边界的 bin。所有 bin 都有左边界，但只有最后一个 bin 有右边界。换言之，最后一

个 bin 有两个边界。

polarhistogram(___,Key=Value)使用一个或多个名称-值对组参数指定其他选项。例如，您可以指定"facealpha"和一个介于 0 和 1 之间的标量值，从而使用半透明条形。

polarhistogram(pax,___) 将在 pax 指定的极坐标区（而不是当前坐标区）中绘制图形。

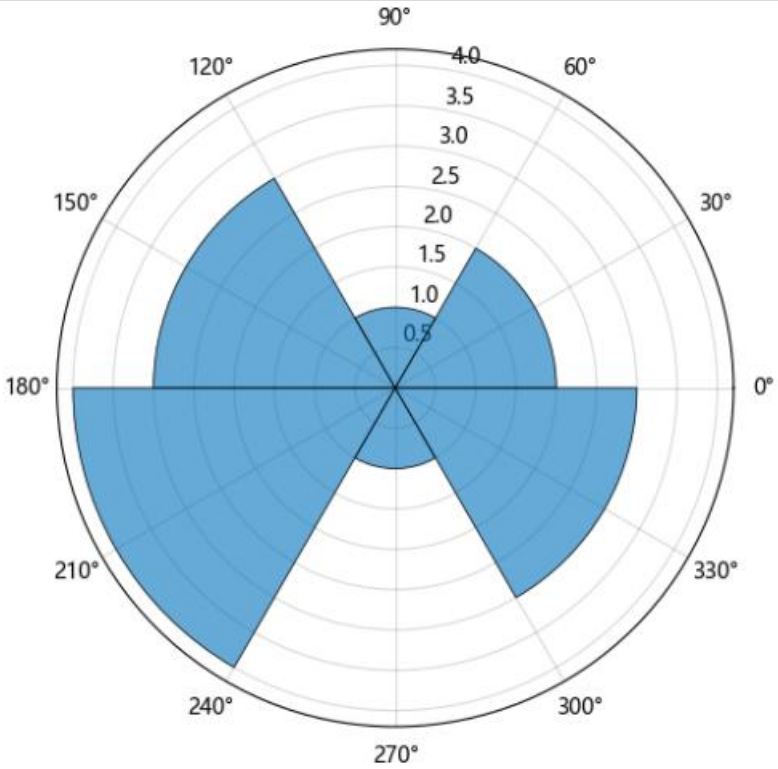
h = polarhistogram(___) 返回 Histogram 对象。在创建直方图之后可使用 h 对其进行修改。有关属性列表，请参阅 Histogram 属性 (opens new window)。

C. 示例

(1). 在极坐标中创建直方图

创建由介于 0 和 2π 之间的值组成的向量。创建一个直方图，该直方图显示划分为六个 bin 的数据。

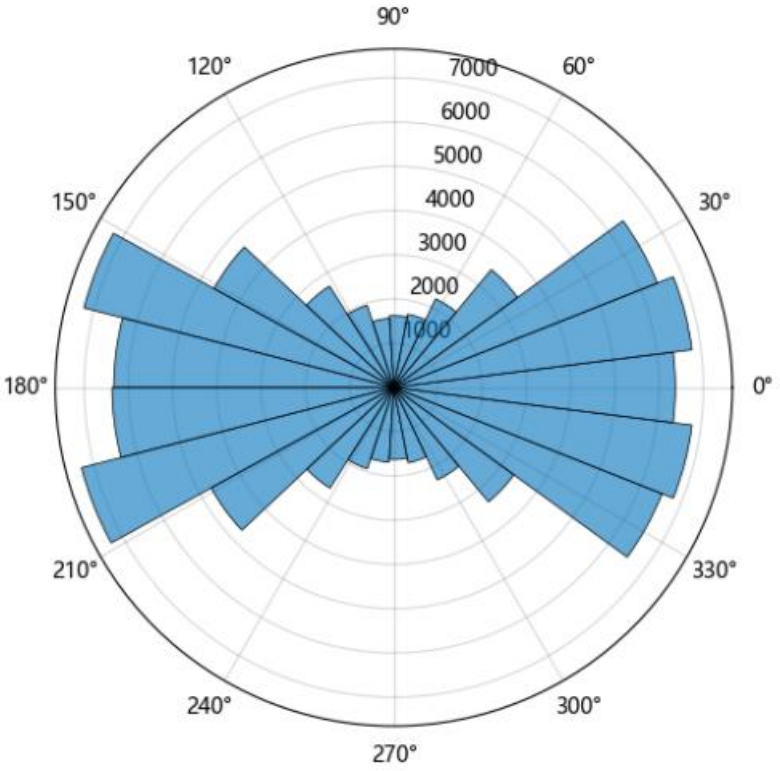
```
theta = [0.1, 1.1, 5.4, 3.4, 2.3, 4.5, 3.2, 3.4, 5.6, 2.3, 2.1, 3.5, 0.6, 6.1]
polarhistogram(theta,6)
```



(2). 为极坐标直方图指定 bin 数

使用介于 −π 和 π 之间的 100,000 个值创建一个直方图，然后将数据划分到 25 个 bin 中。

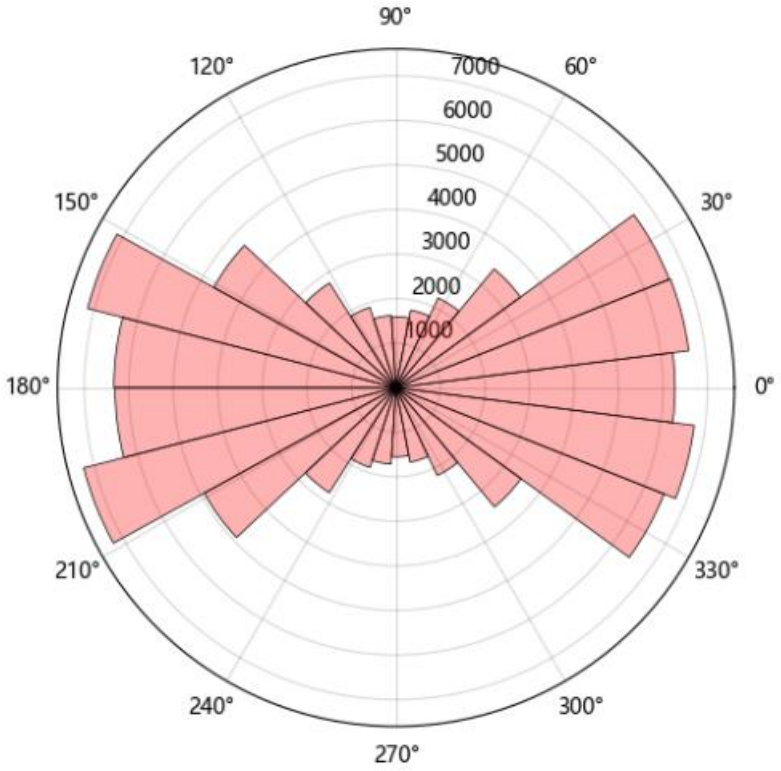
```
theta = (atan.(rand(100000,1).-0.5, 2 .*(rand(100000,1).-0.5)));
a = polarhistogram(theta,25)
```



(3). 修改直方图外观

在极坐标中创建一个直方图，然后更改其外观。通过将 facecolor 属性设置为颜色名称字符向量（例如 "red"）或 RGB 三元组，指定条形的颜色。通过将 facealpha 属性设置为介于 0 和 1 之间的值，指定透明度。

```
theta = atan.(rand(100000,1).-0.5, 2 .*(rand(100000,1).-0.5));
polarhistogram(theta,25, facecolor = "r", facealpha = 0.3)
```



ezpolar

易用的极坐标绘图函数

A. 语法

```
ezpolar(fun)
ezpolar(fun,[a,b])
ezpolar(ax,...)
h = ezpolar(...)
```

B. 说明

ezpolar(fun) 在默认域 0 < theta < 2π 中绘制极坐标曲线 rho = fun(theta)。fun 是函数句柄。

ezpolar(fun,[a,b]) 绘制 a < theta < b 的 fun。

ezpolar(ax,...) 将图形绘制到带有句柄 ax 的坐标区中，而不是当前坐标区(gca) 中。

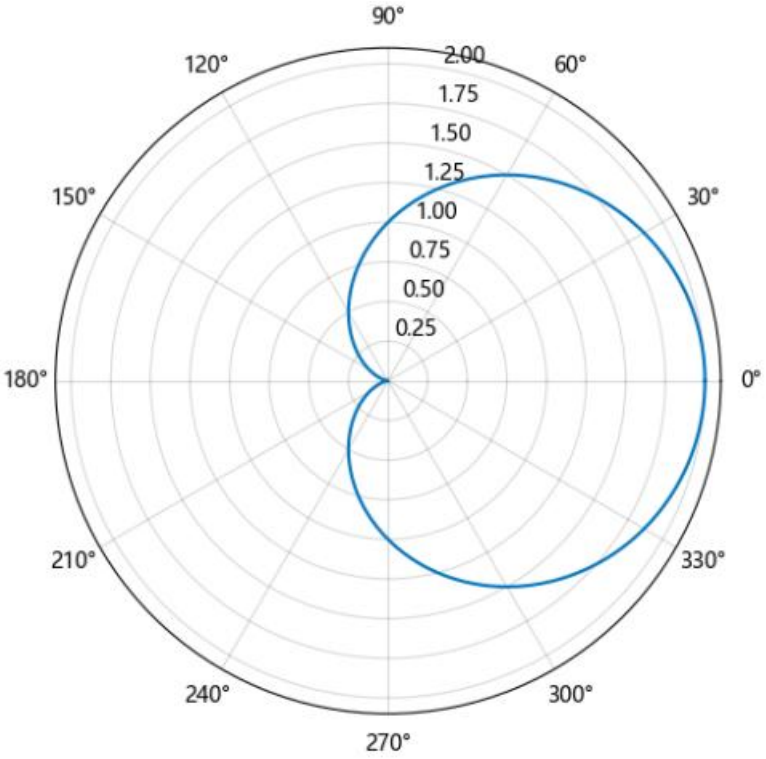
h = ezpolar(...) 将句柄返回给 h 中的线对象。

C. 示例

(1). 数学函数的极坐标图

在域 [0,2π] 上绘制函数 $1+\cos(t)$ 。

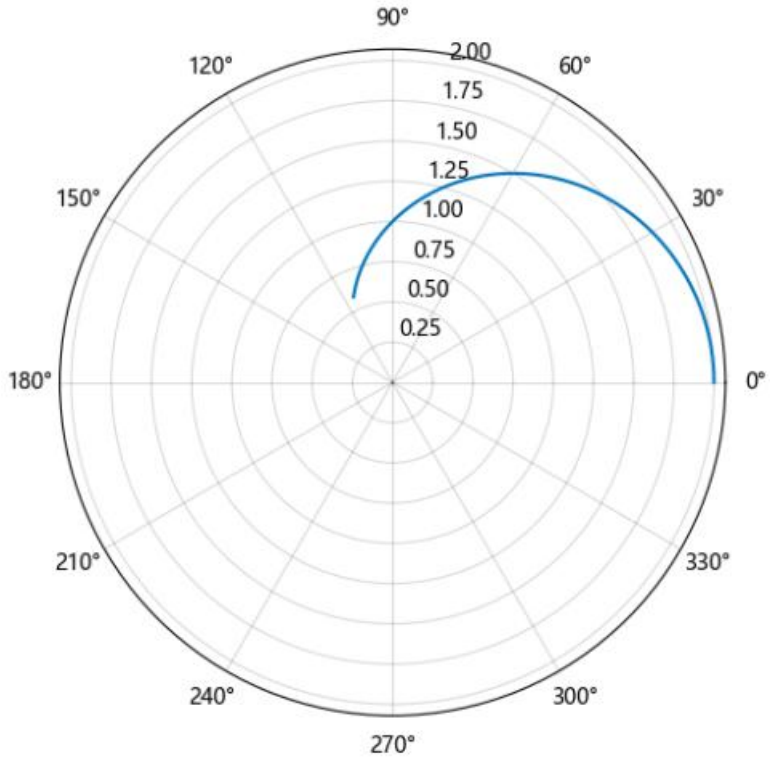
```
figure()
ezpolar((t)->1+cos(t))
```



(2). 数学函数设置范围

在域 [0,2] 上绘制函数 $1+\cos(t)$ 。

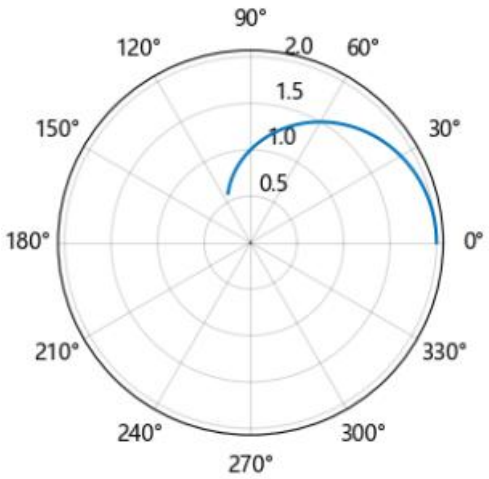
```
figure()
ezpolar((t)->1+cos(t),[0,2])
```



(3). 在指定的极坐标轴绘制数学函数

在指定坐标轴上绘制函数 。

```
figure()
ax = subplot(1,2,1,projection="polar")
ezpolar(ax, (t)->1+cos(t), [0,2])
```



3.3.1.4.2 自定义极坐标区

函数名	简介

rlim	在极坐标中绘制线条
thetalim	极坐标中的散点图
rticks	极坐标中的直方图
thetaticks	易用的极坐标绘图函数
rticklabels	在极坐标中绘制线条
thetaticklabels	极坐标中的散点图
rtickformat	极坐标中的直方图
thetatickformat	易用的极坐标绘图函数
rtickangle	在极坐标中绘制线条
polaraxes	极坐标中的散点图

rlim

设置或查询极坐标区的 r 坐标轴范围

A. 语法

```
rlim(limits)

rlim("auto")
rlim("manual")

rl = rlim()

___ = rlim(pax,___)
```

B. 说明

rlim(limits)指定当前极坐标区的 r 坐标轴范围。将 limits 指定为[rminrmax]形式的二元素向量，其中 rmax 是大于 rmin 的数值。

rlim("auto") 允许 Syslab 选择 r 坐标轴范围。此命令将极坐标区对象的 RLimMode 属性设置为 "auto"。

rlim("manual") 防止范围自动更改。当使用 hold("on") 命令向极坐标区添加新数据时，如果要保留当前范围，请使用此选项。此命令将极坐标区对象的 RLimMode 属性设置为 "manual"。

rl=rlim()返回包含当前极坐标区范围的二元素向量。

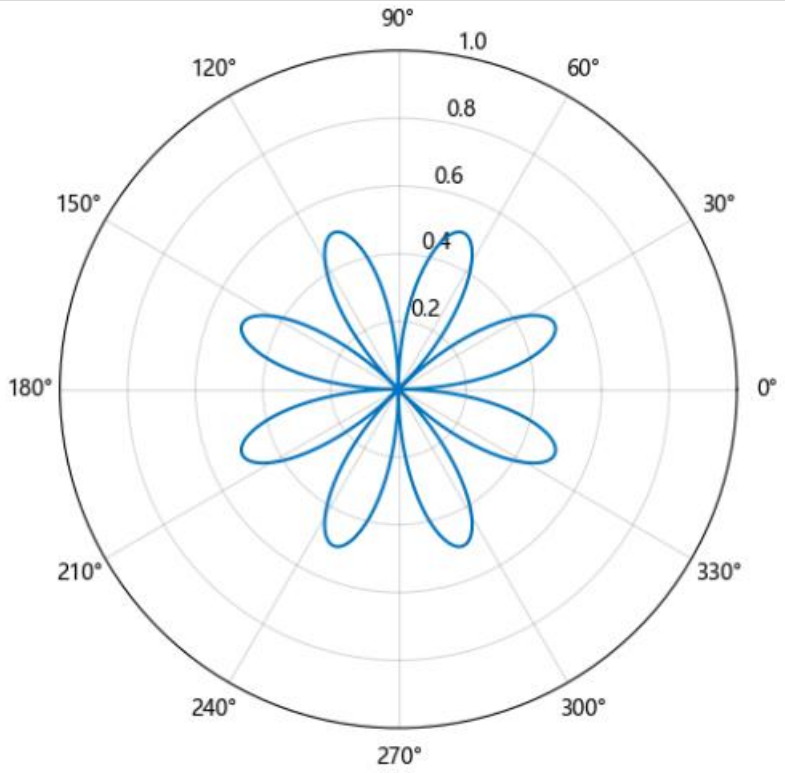
___ = rlim(pax ,___) 使用 pax 指定的极坐标区，而不是使用当前极坐标区。指定 pax 作为第一个输入参数。仅当原始语法支持时才能包括更多输入或输出参数。

C. 示例

(1). 设置 r 坐标轴范围

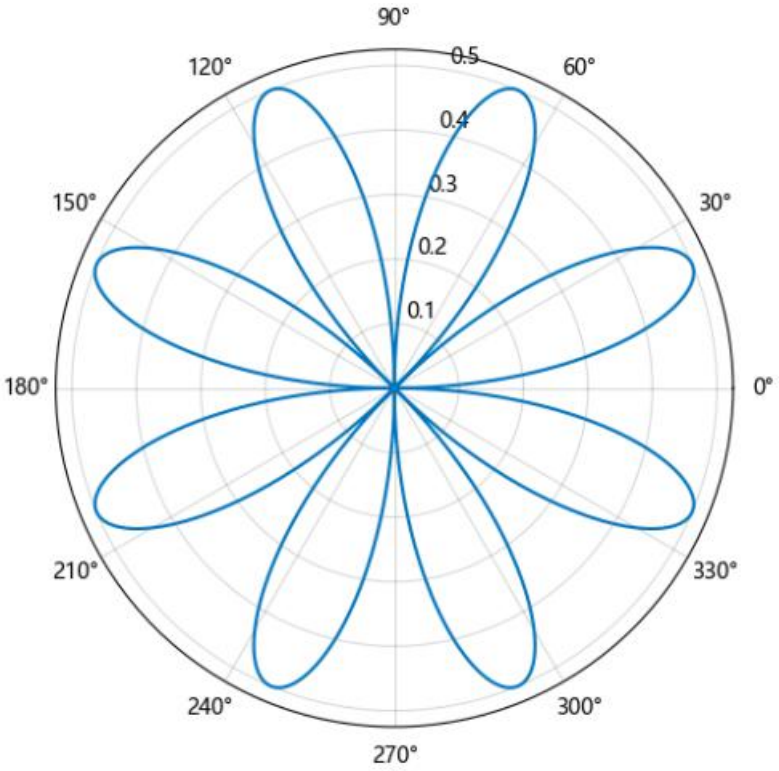
创建一个极坐标图并更改 r 坐标轴范围。

```
theta = 0:0.01:2*pi;
rho = sin.(2*theta) .* cos.(2*theta);
polarplot(theta, rho)
rlim([0 1])
```



将范围恢复为原来的值。

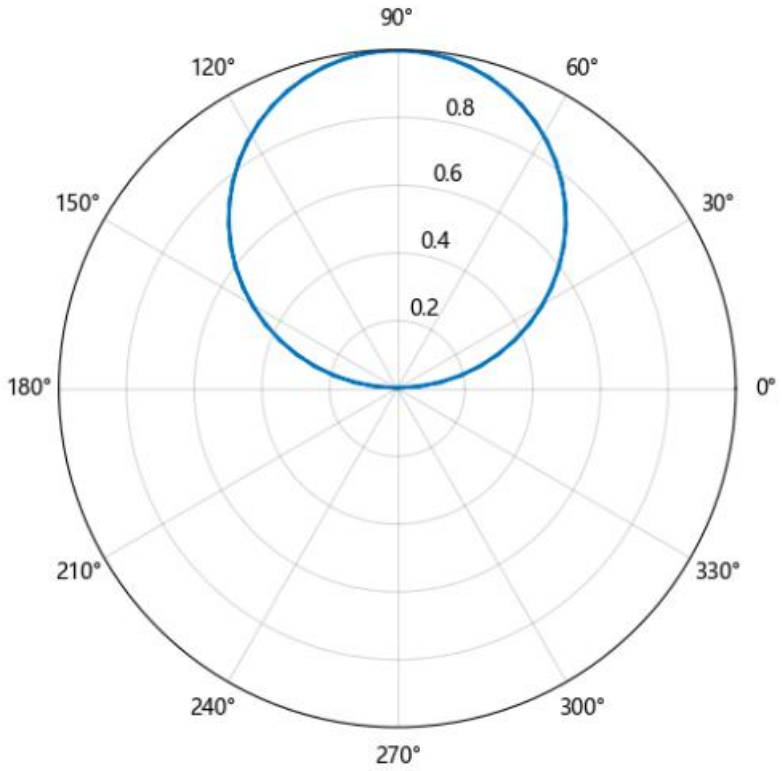
```
rlim("auto")
```



(2). 绘制负半径值

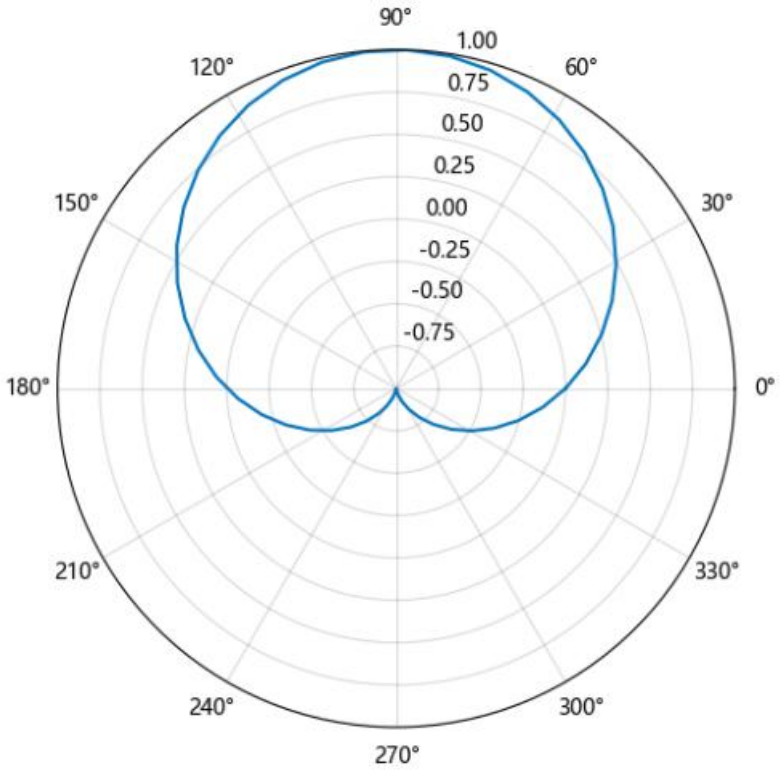
使用负半径值创建极坐标图。默认情况下，polarplot 会将负值反射穿过原点。

```
theta = LinRange(0, 2 * pi, 50);
rho = sin.(theta);
polarplot(theta, rho)
```



更改 r 轴的范围，使其从 -1 到 1。

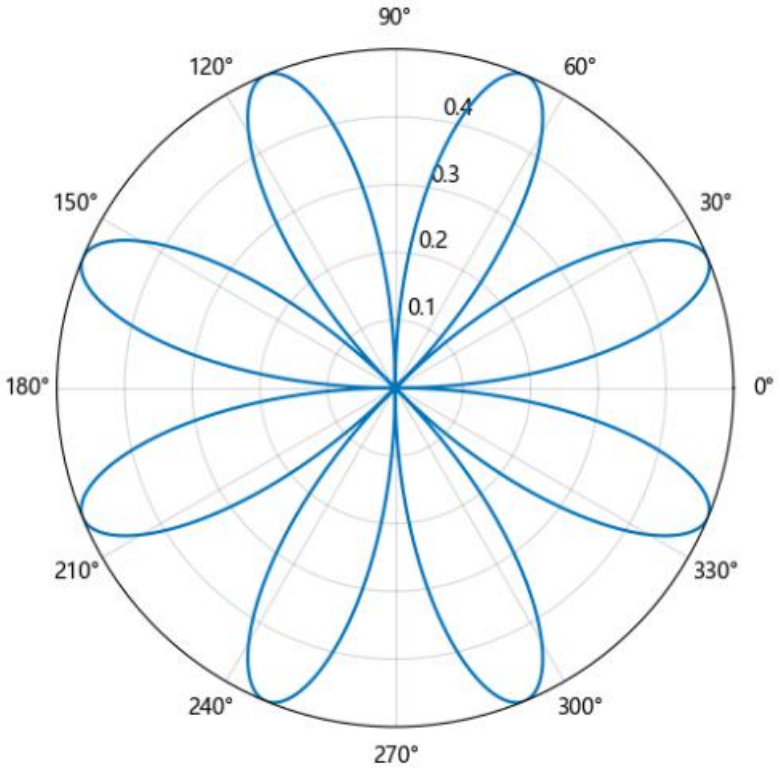
```
polarplot(theta, rho, rlim=(-1,1))
```



(3). 返回 r 坐标轴范围

创建一个极坐标图并返回 r 坐标轴范围。

```
theta = 0:0.01:2*pi;
rho = sin.(2*theta) .* cos.(2*theta);
p = polarplot(theta, rho)
```



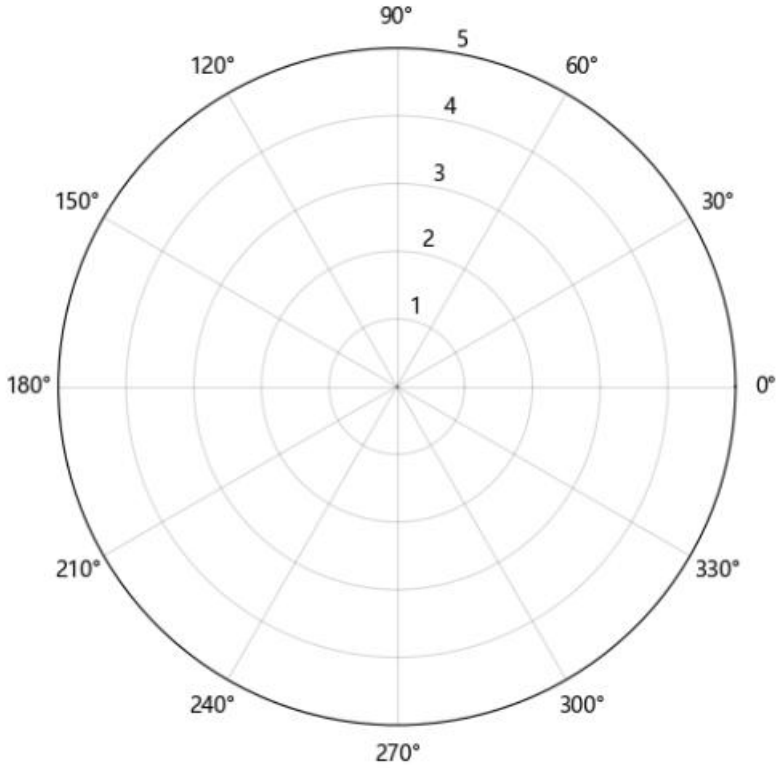

```
rl = rlim()
```

```
rl = 1×2
0 0.5000
```

(4). 指定特定极坐标区的 r 坐标轴范围

通过将极坐标区对象指定为 rlim 的第一个输入，为特定的极坐标区设置范围。否则，rlim 将为当前坐标区设置范围。

```
pax = subplot(polar = true)
rlim([0 5])
```



thetalim

设置或查询极坐标区的 theta 坐标轴范围

A. 语法

```
thetalim(limits)
   thetalim("auto")
thetalim("manual")

tl = thetalim()

___ = thetalim(pax,___)
```

B. 说明

thetalim(limits)指定当前极坐标区的 theta 坐标轴范围。将 limits 指定为[thetaminthetamax]形式的二元素向量，其中 thetamax 是大于 thetamin 的数值。

thetalim("auto") 允许 Syslab 选择 theta 坐标轴范围。此命令将极坐标区对象的 ThetaLimMode 属性设置为 "auto"。

thetalim("manual") 防止范围自动更改。此命令将极坐标区对象的 ThetaLimMode 属性设置为 "manual"。

tl = thetalim() 返回包含当前极坐标区范围的二元素向量。

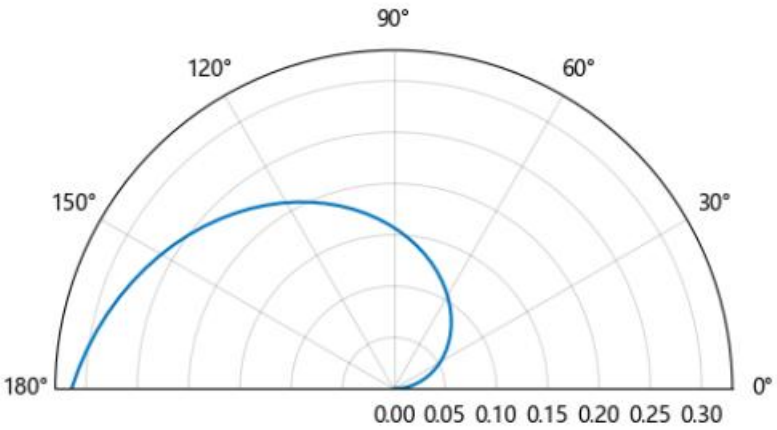
___ = thetalim(pax,___)使用 pax 指定的极坐标区对象，而不是使用当前极坐标区。指定 pax 作为第一个输入参数。仅当原始语法支持时才能包括更多输入或输出参数。

C. 示例

(1). 指定 theta 坐标轴范围

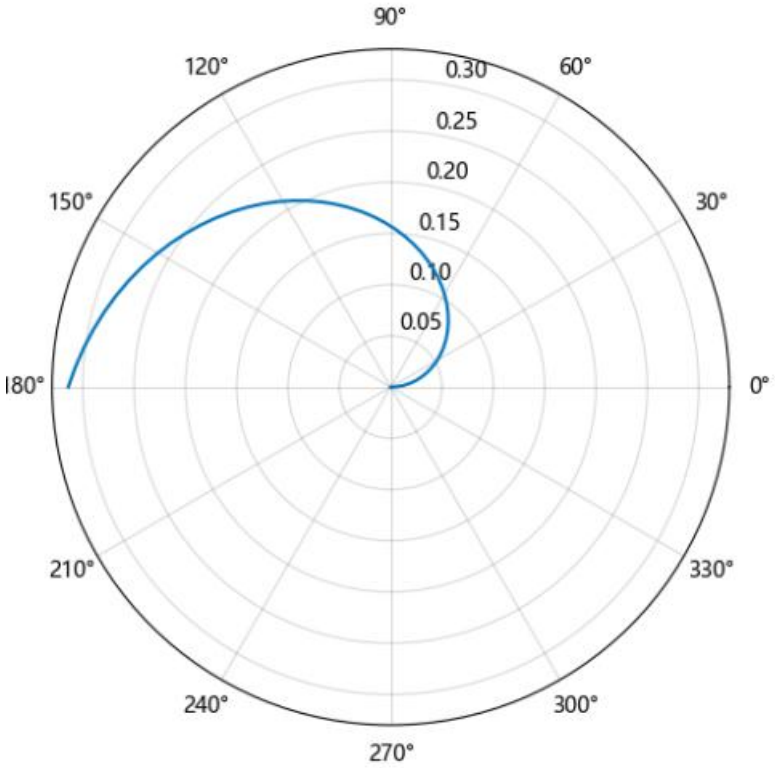
创建一个极坐标图并更改 theta 坐标轴范围。

```
theta = LinRange(0, pi, 50);
rho = theta ./ 10;
polarplot(theta, rho)
thetalim([0 180])
```



将范围恢复为原来的值。

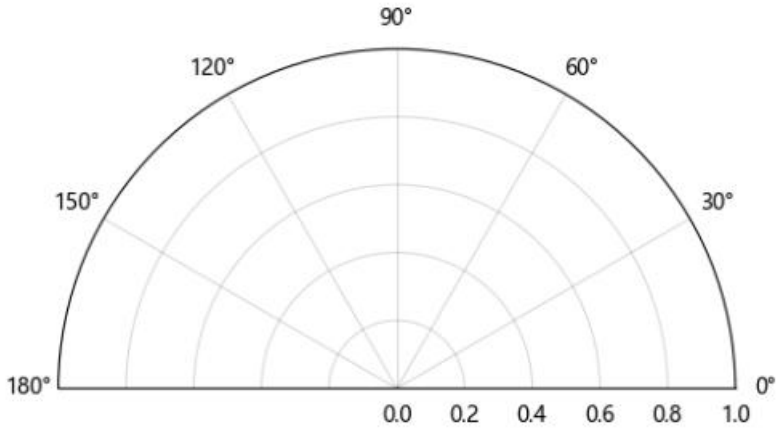
```
thetalim("auto")
```



(2). 指定特定极坐标区的 theta 坐标轴范围

通过将极坐标区对象指定为 thetalim 的第一个输入，为特定的极坐标区设置范围。否则，thetalim 将为当前坐标区设置范围。

```
pax = subplot(polar = true)
thetalim(pax, [0 180])
```



rticks

设置或查询 r 轴刻度值

A. 语法

```
rticks(ticks)
rt = rticks()

rticks("auto")
rticks("manual")

___ = rticks(pax,___)
```

B. 说明

rticks(ticks)设置 r 轴刻度值，即 r 轴上显示刻度线和网格线的位置。指定 ticks 为递增值向量；例如[0 2 4 6]。此命令作用于当前坐标区。

rt = rticks() 以向量的形式返回当前 r 轴刻度值。

rticks("auto") 设置自动模式，从而允许坐标区确定 r 轴刻度值。如果您更改了刻度值，然后又想将它们设置回默认值，则可以使用此选项。

rticks("manual") 设置手动模式，从而将 r 轴刻度值冻结在当前值。如果您希望在调整坐标区大小或者向坐标区添加新数据时保留当前刻度值，则可以使用此选项。

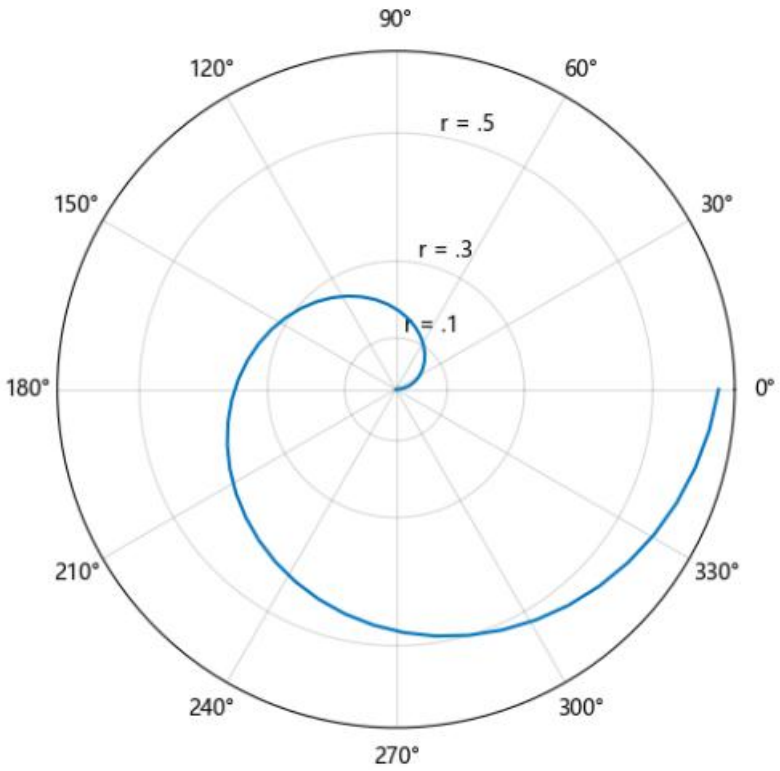
___ = rticks(pax,___)使用坐标区对象 pax，而不是当前坐标区。指定 pax 作为上述任何语法的第一个输入参数。

C. 示例

(1). 指定 r 轴刻度值和标签

创建一个极坐标图。在 r 轴上的值 0.1、0.3 和 0.5 处显示刻度线和网格线。然后为每个刻度线指定一个标签。

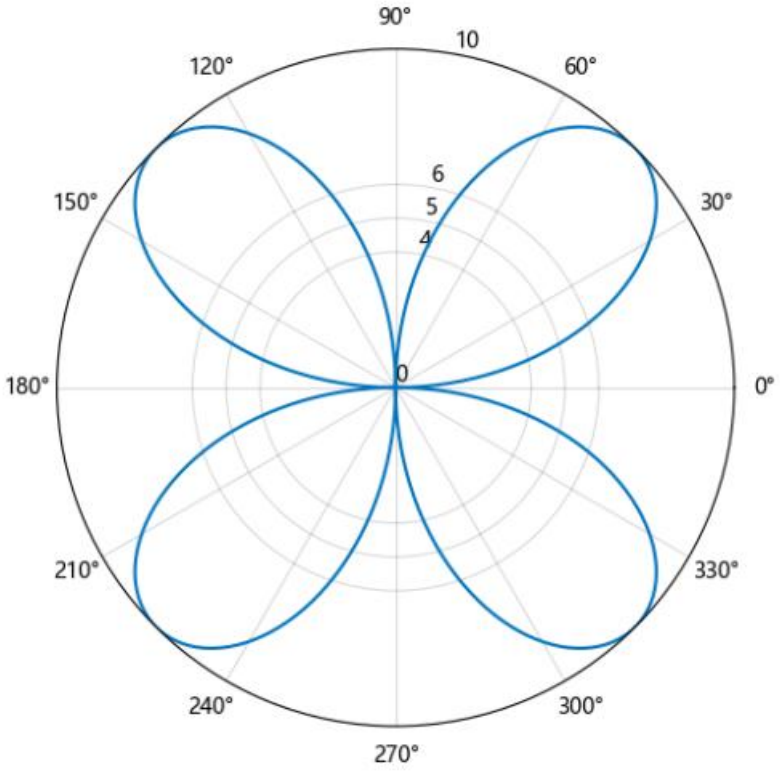
```
theta = LinRange(0, 2 * pi, 50);
rho = theta ./ 10;
polarplot(theta, rho)
rticks([0.1 0.25 0.5])
rticklabels(["r = .1", "r = .3", "r = .5"])
```



(2). 指定非均匀分布的 r 轴刻度值

在 r 轴上 0 和 10 之间的非均匀分布的值处显示刻度线和网格线。Syslab 用数值标注刻度线。

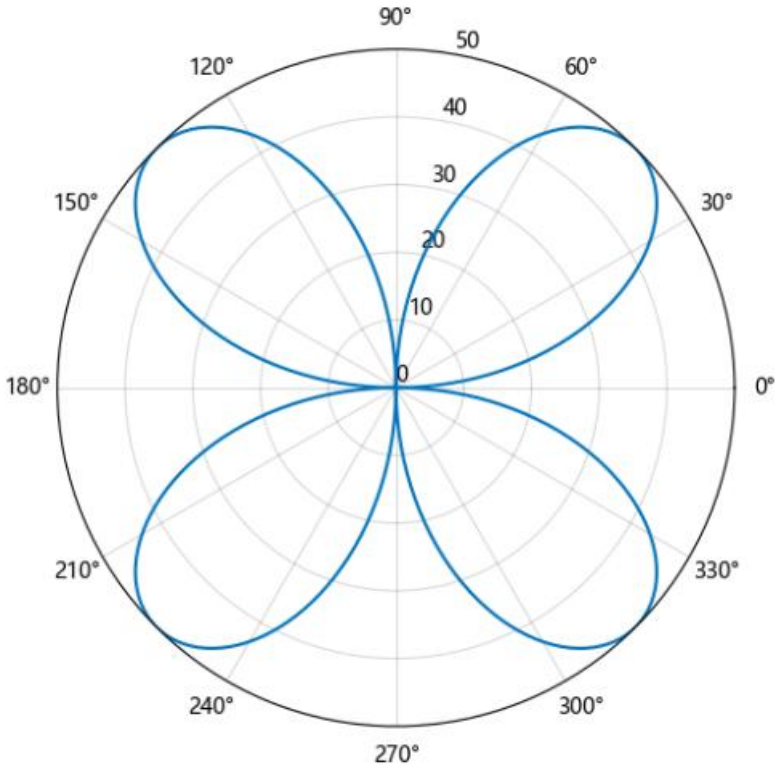
```
theta = 0:0.01:2*pi;
rho = 10 .*sin.(2 .*theta);
polarplot(theta,rho)
rticks([0 4 5 6 10])
```



(3). 按 10 递增 r 轴刻度值

从 0 开始，到 50 结束，以 10 为增量在 r 轴上显示刻度线和网格线。

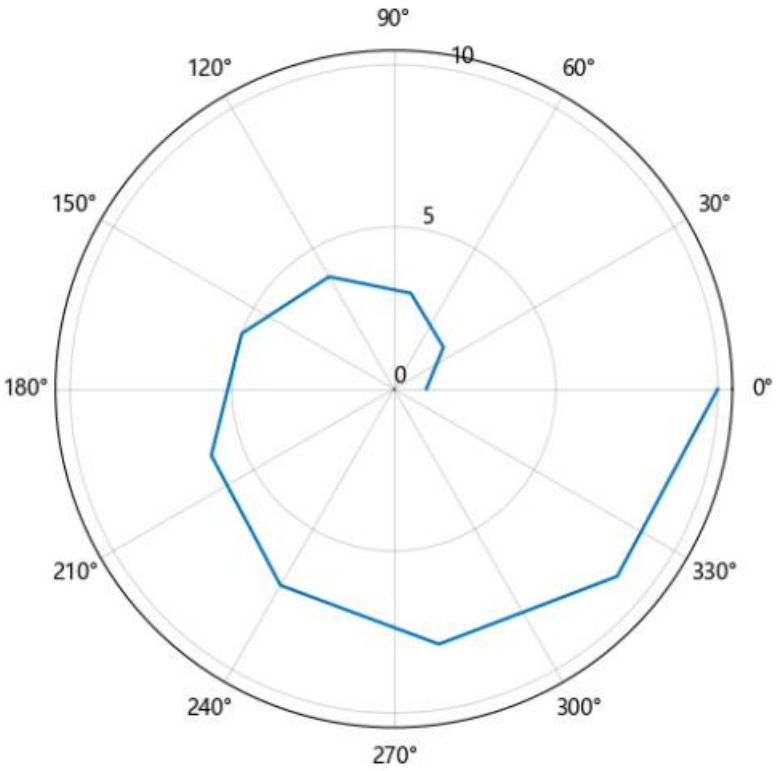
```
theta = 0:0.01:2*pi;
rho = 50 .*sin.(2 .*theta);
polarplot(theta,rho)
rticks(0:10:50)
```



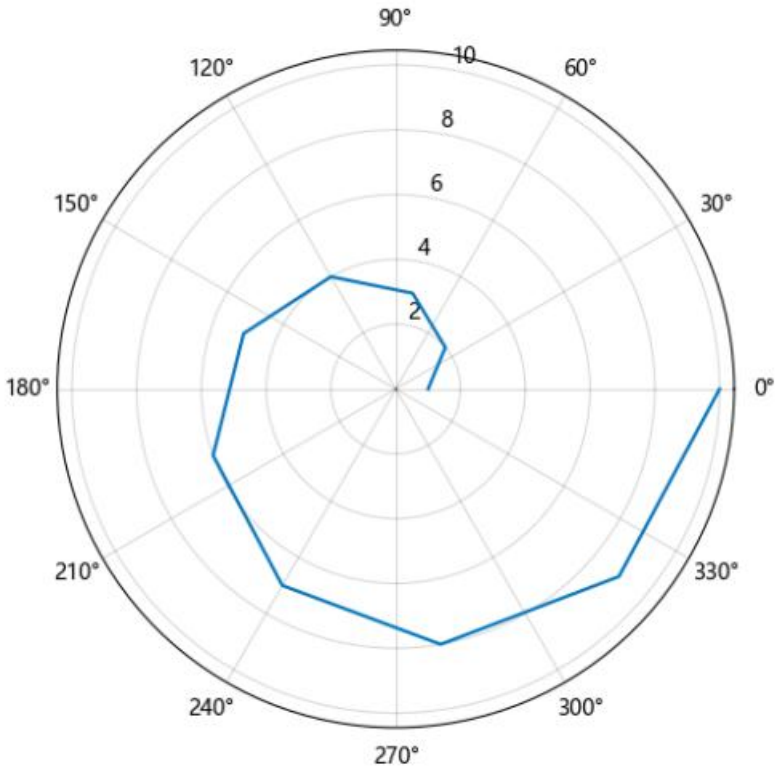
(4). 将 r 轴刻度值设置回默认值

创建一个极坐标图，并指定 r 轴刻度值。然后，将 r 轴刻度值设置回默认值。

```
polarplot(1:1:10)
rticks([0 5 10])
```



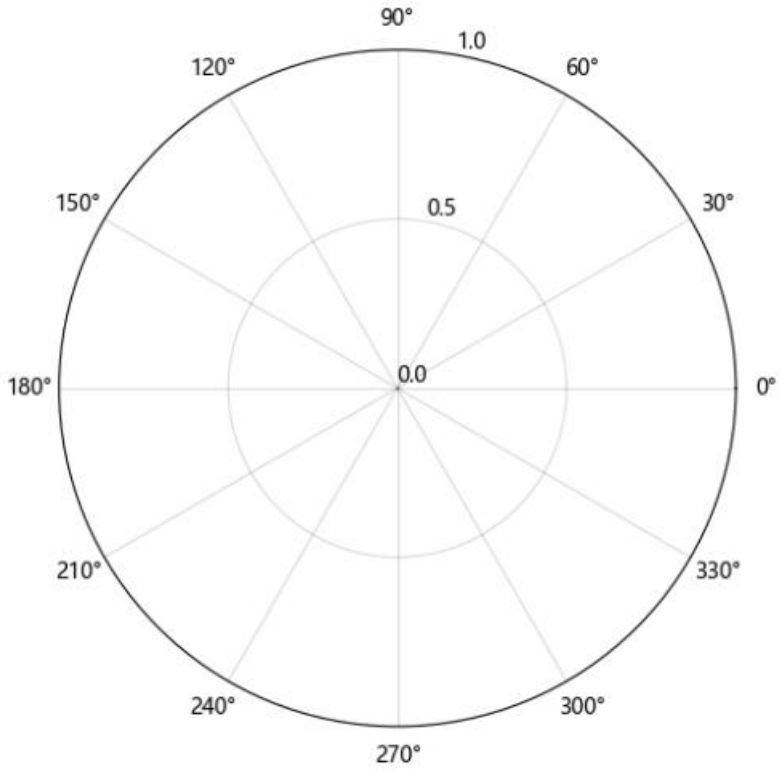

```
rticks("auto")
```



(5). 指定特定极坐标区的 r 轴刻度值

创建极坐标区，并返回极坐标区对象 pax。通过将 pax 作为 rticks 函数的第一个输入参数进行传递，确保该函数作用于您刚创建的极坐标区。

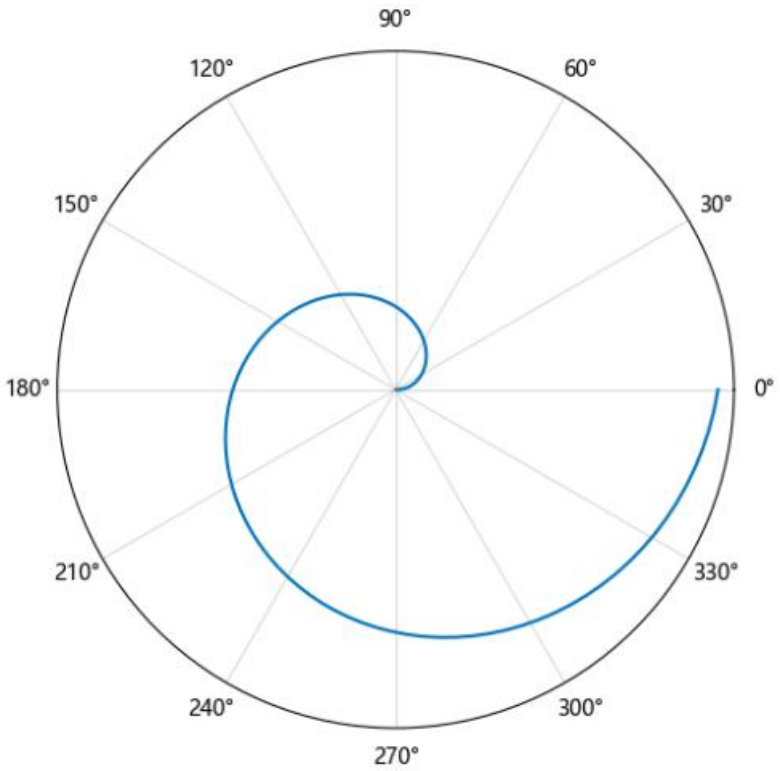
```
pax = subplot(polar = true)
rticks(pax, [0 .5 1])
```



(6). 删除 r 轴刻度线和网格线

通过指定刻度值为空数组，删除 r 轴上的刻度线和网格线。

```
polarplot(1:100)
rticks([])
```



thetaticks

设置或查询极坐标区的 r 坐标轴范围

A. 语法

```
thetaticks(ticks)
tv = thetaticks()

thetaticks("auto")
thetaticks("manual")

___ = thetaticks(pax,___)
```

B. 说明

thetaticks(ticks)设置 theta 轴刻度值，这些值是沿 theta 轴出现刻度线和网格线的位置。指定 ticks 为递增值向量；例如[0 90 180 270]。此命令作用于当前坐标区。

tv = thetaticks() 以向量形式返回当前 theta 轴刻度值。

thetaticks("auto") 设置自动模式，使坐标区确定 theta 轴刻度值。如果您更改了刻度值，然后又想将它们设置回默认值，则可以使用此选项。

hetaticks("manual") 设置手动模式，将 theta 轴刻度值冻结在当前值。如果您希望在调整坐标区大小或者向坐标区添加新数据时保留当前刻度值，则可以使用此选项。

___ = thetaticks(pax,___)使用 pax 指定的坐标区，而不是使用当前坐标区。指定 pax 作为上述任何语法的第一个输入参数。

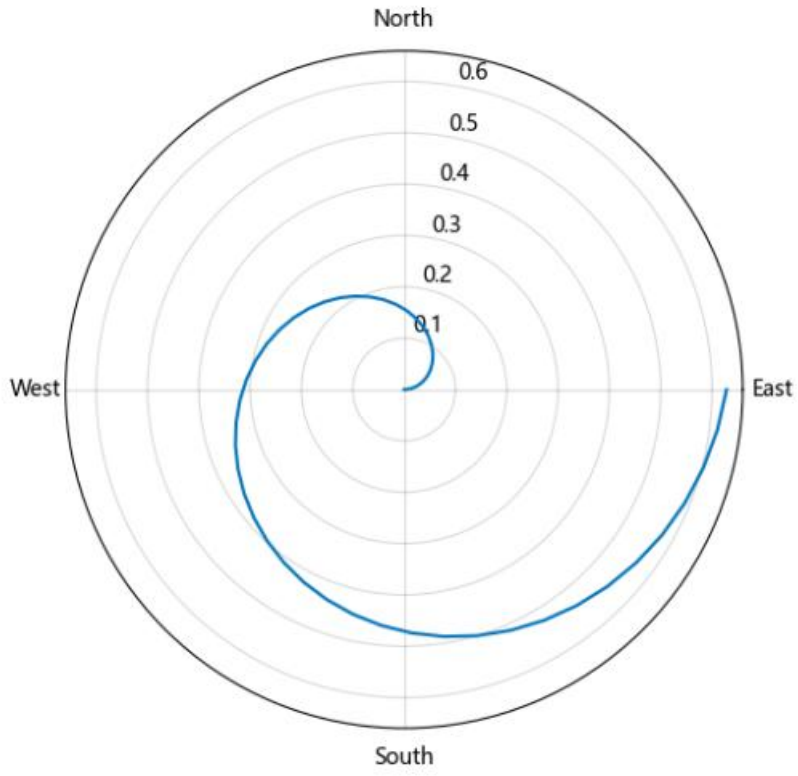
C. 示例

(1). 指定 theta 轴刻度值和标签

创建一个极坐标图。在 0 度、90 度、180 度和 270 度处显示沿 theta 轴的刻度线和网格线。然后为每个刻度线指定一个标签。

```
theta = LinRange(0, 2 * pi, 50);
rho = theta ./ 10;
polarplot(theta, rho)

thetaticks([0 90 180 270])
thetaticklabels(["East","North","West","South"])
```

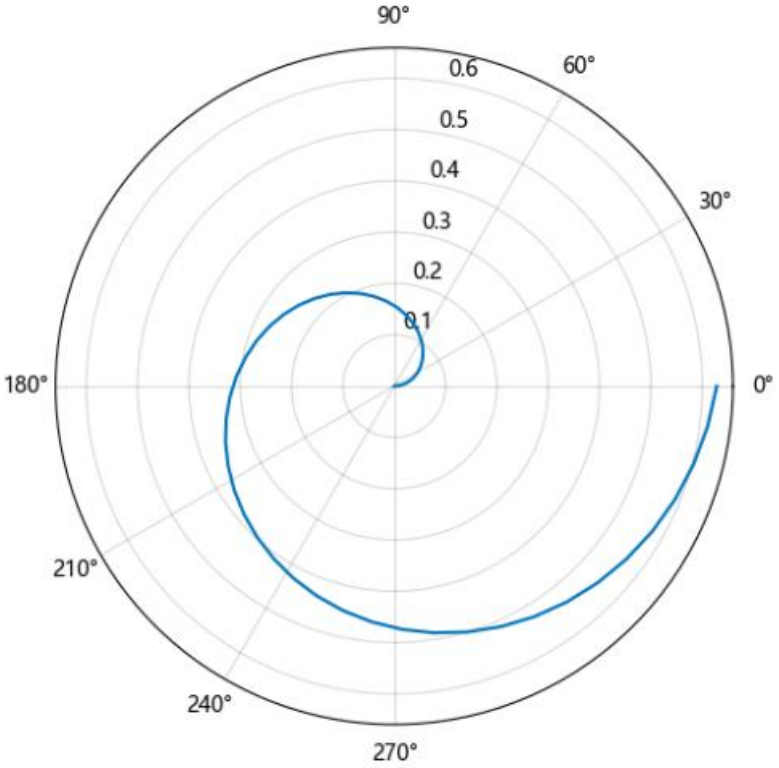


(2). 指定非均匀 theta 轴刻度值

沿 theta 轴的 0 到 360 度之间的非均匀值处显示刻度线。

```
theta = LinRange(0, 2 * pi, 50);
```

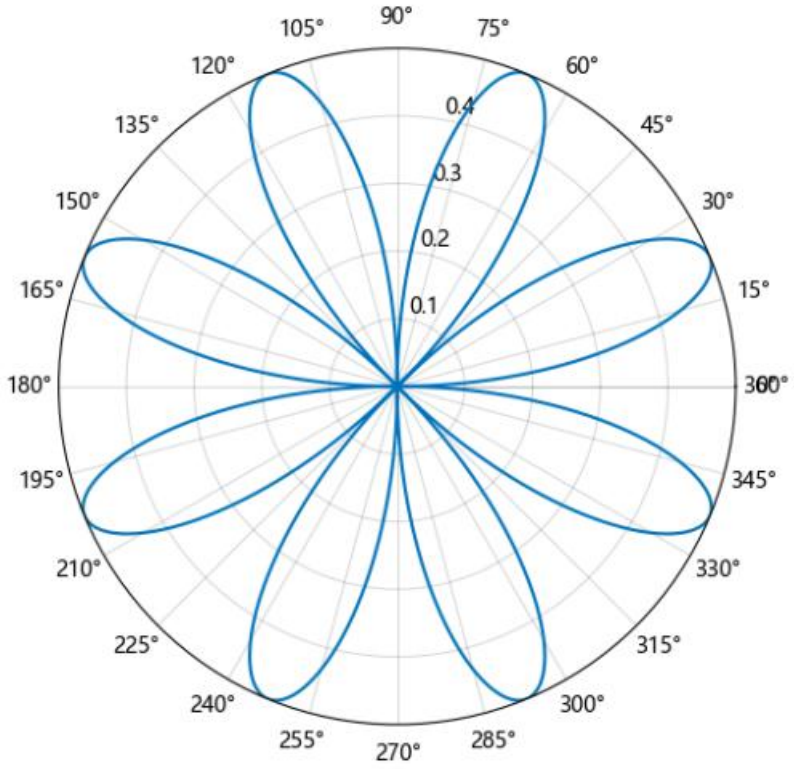
```
rho = theta ./ 10;
polarplot(theta, rho)
thetaticks([0 30 60 90 180 210 240 270])
```



(3). 以 15 度为增量递增 theta 轴的值

沿 theta 轴每 15 度显示一个刻度线。

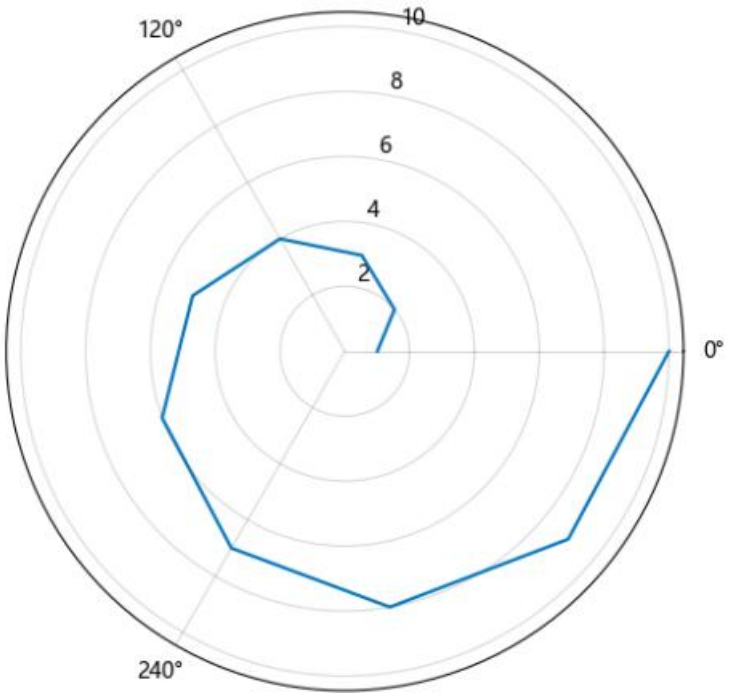
```
theta = 0:0.01:2*pi;
rho = sin.(2 .*theta).*cos.(2 .*theta);
polarplot(theta,rho)
thetaticks(0:15:450)
```



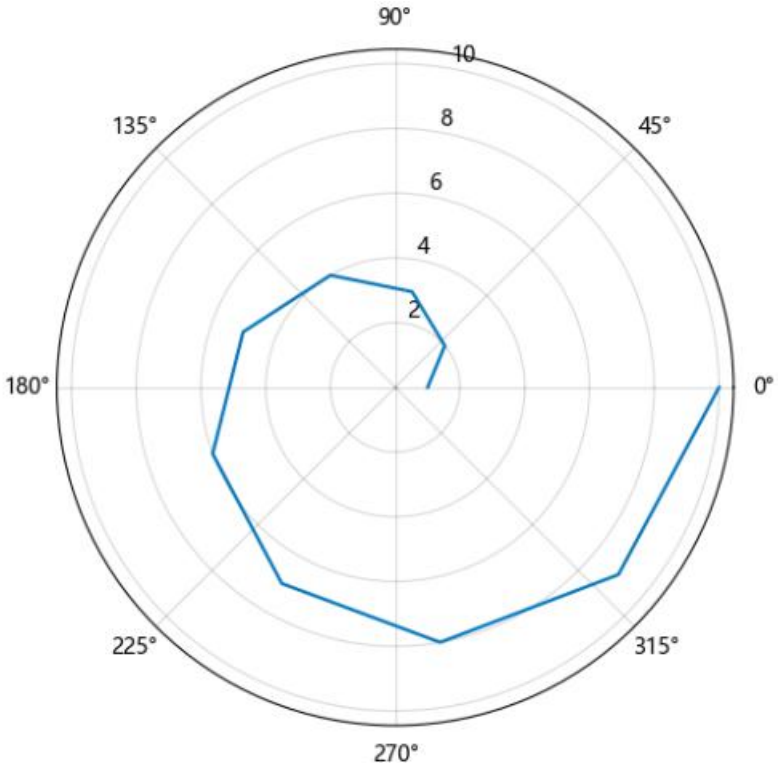
(4). 将 theta 轴刻度值设置回默认值

创建一个极坐标图并指定 theta 轴刻度值。然后，将 theta 轴刻度值设置回默认值。

```
polarplot(1:1:10)
thetaticks([0 120 240])
```



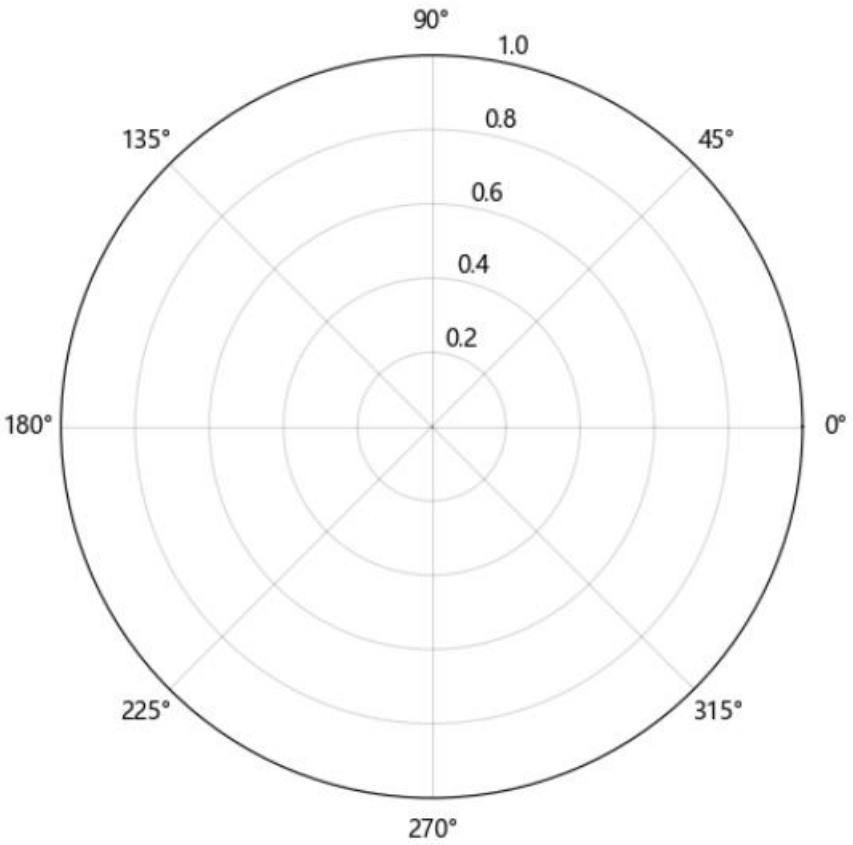
```
thetaticks("auto")
```



(5). 设置特定极坐标区的 theta 轴刻度值

创建极坐标区，并返回极坐标区对象 pax。通过将 pax 作为 thetaticks 函数的第一个输入参数进行传递，确保该函数作用于您刚创建的极坐标区。

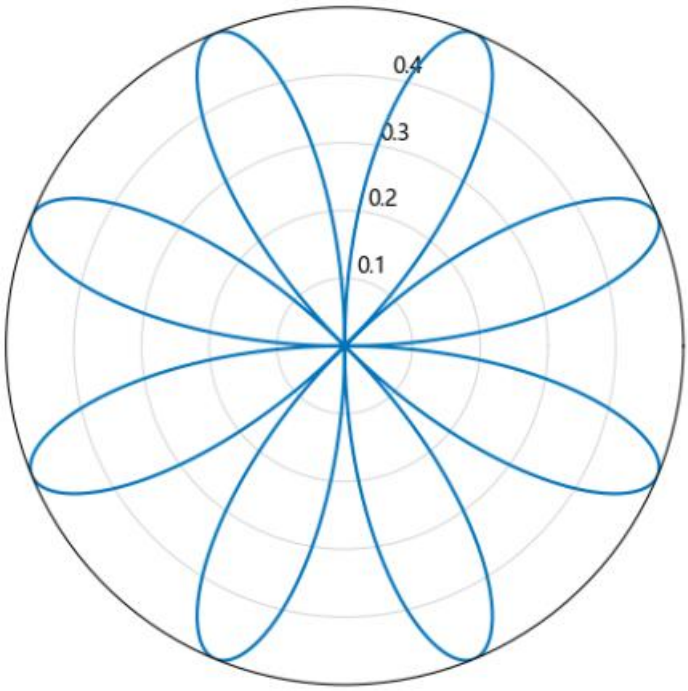
```
pax = subplot(polar = true)
thetaticks(pax,0:45:360)
```



(6). 删除 theta 轴刻度线和网格线

通过将刻度值指定为空数组来删除沿 theta 轴的刻度线和网格线。

```
theta = 0:0.01:2*pi;
rho = sin.(2 .*theta).*cos.(2 .*theta);
polarplot(theta,rho)
thetaticks([])
```



rticklabels

设置或查询 r 轴刻度标签

A. 语法

```
rticklabels(labels)
rl = rticklabels()

rticklabels("auto")
rticklabels("manual")

___ = rticklabels(pax,___)
```

B. 说明

rticklabels(labels)设置当前坐标区的 r 轴刻度标签。可将 labels 指定为字符串数组，例如["January","February","March"]。如果您指定标签，则 r 轴刻度值和刻度标签不会再根据对坐标区所做的更改而自动更新。

rl = rticklabels() 返回当前坐标区的 r 轴刻度标签。

rticklabels("auto") 设置自动模式，从而允许坐标区确定 r 轴刻度标签。如果您设置了标签，然后又想将它们设置回默认值，则可以使用此选项。

rticklabels("manual") 设置手动模式，从而将 r 轴刻度标签冻结在当前值。

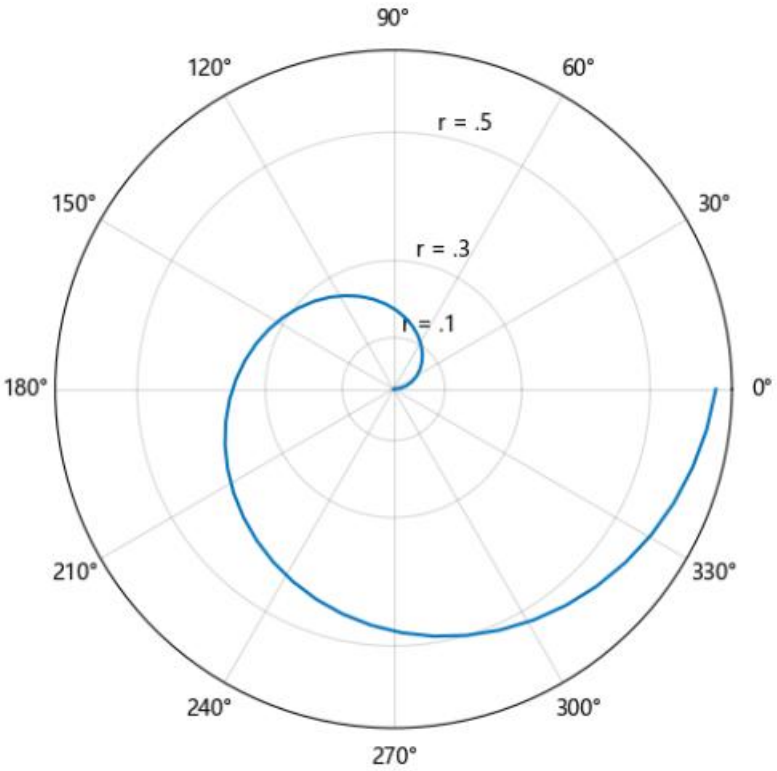
___ = rticklabels(pax,___)使用 pax 指定的坐标区，而不是使用当前坐标区。指定 ax 作为上述任何语法的第一个输入参数。

C. 示例

(1). 指定 r 轴刻度值和标签

创建一个极坐标图。在 r 轴上的值 0.1、0.3 和 0.5 处显示刻度线和网格线。然后为每个刻度线指定一个标签。

```
theta = LinRange(0, 2 * pi, 50);
rho = theta ./ 10;
polarplot(theta, rho)
rticks([0.1 0.25 0.5])
rticklabels(["r = .1", "r = .3", "r = .5"])
```



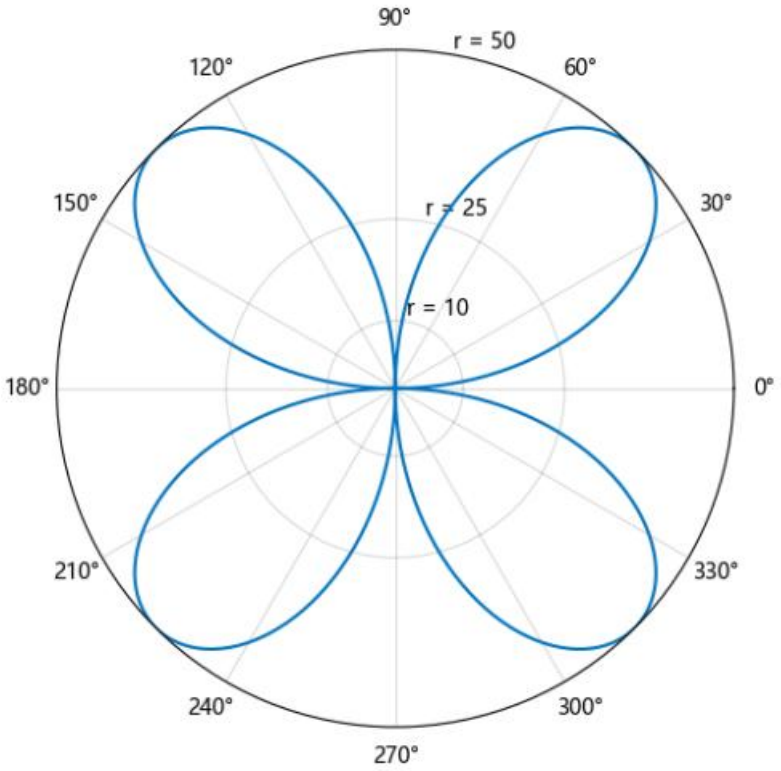
(2). 指定特定极坐标区的 r 轴刻度标签

创建一个极坐标图，并将极坐标区对象赋给变量 pax。通过指定 pax 作为 rticks 和 rticklabels 函数的第一个输入参数，确保这两个函数作用于您刚创建的极坐标区。

```
theta = 0:0.01:2*pi;
rho = 50 * sin.(2 *theta);
polarplot(theta,rho)
```

```
pax = gca()

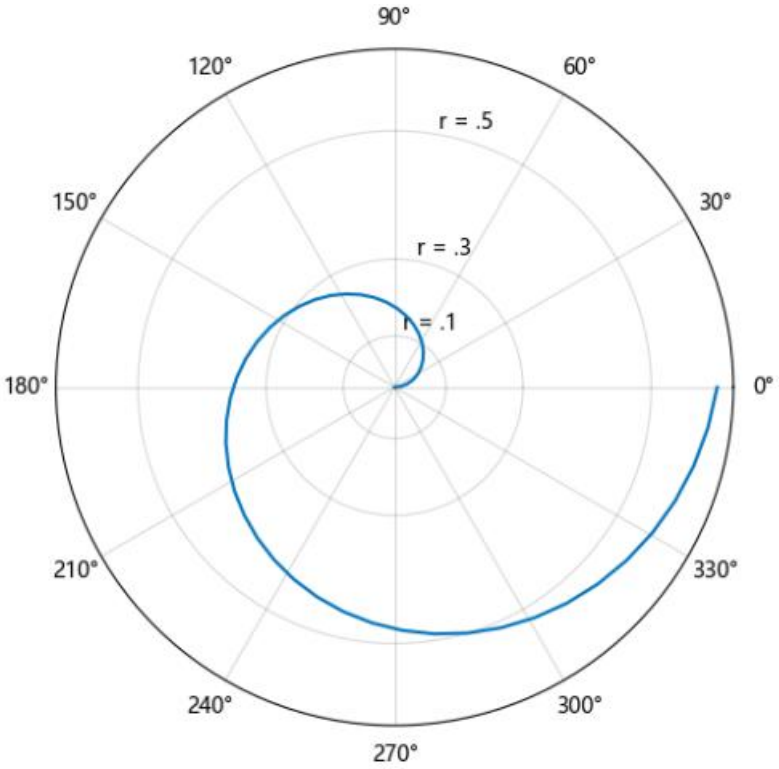
rticks(pax,[10 25 50])
rticklabels(pax,["r = 10","r = 25","r = 50"])
```



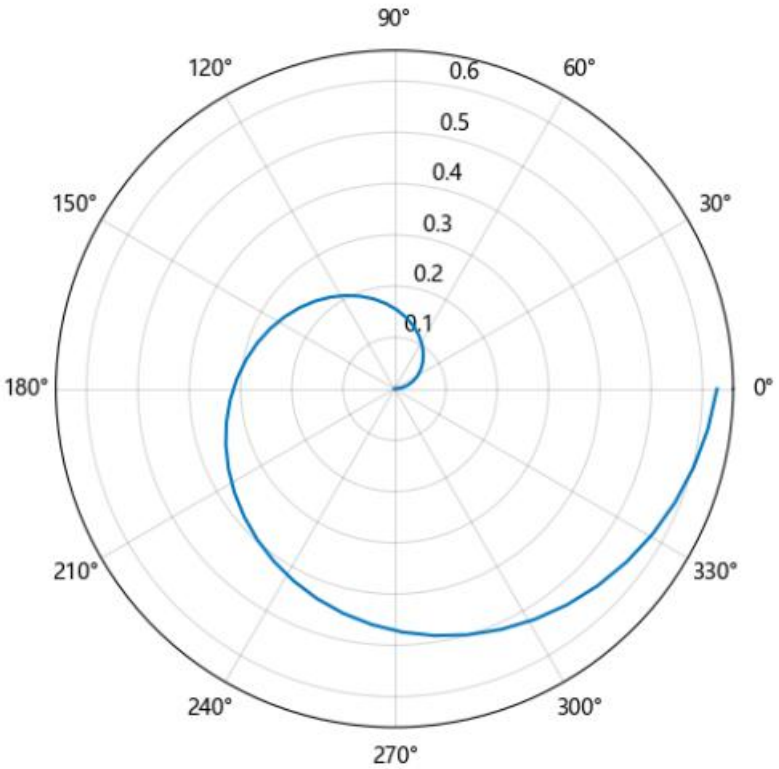
(3). 将 r 轴刻度标签设置回默认标签

创建一个极坐标图并指定 r 轴刻度值和对应的标签。然后，将 r 轴刻度值和标签设置回默认值。

```
theta = LinRange(0, 2 * pi, 50);
rho = theta ./ 10;
polarplot(theta, rho)
rticks([0.1 0.25 0.5])
rticklabels(["r = .1", "r = .3", "r = .5"])
```



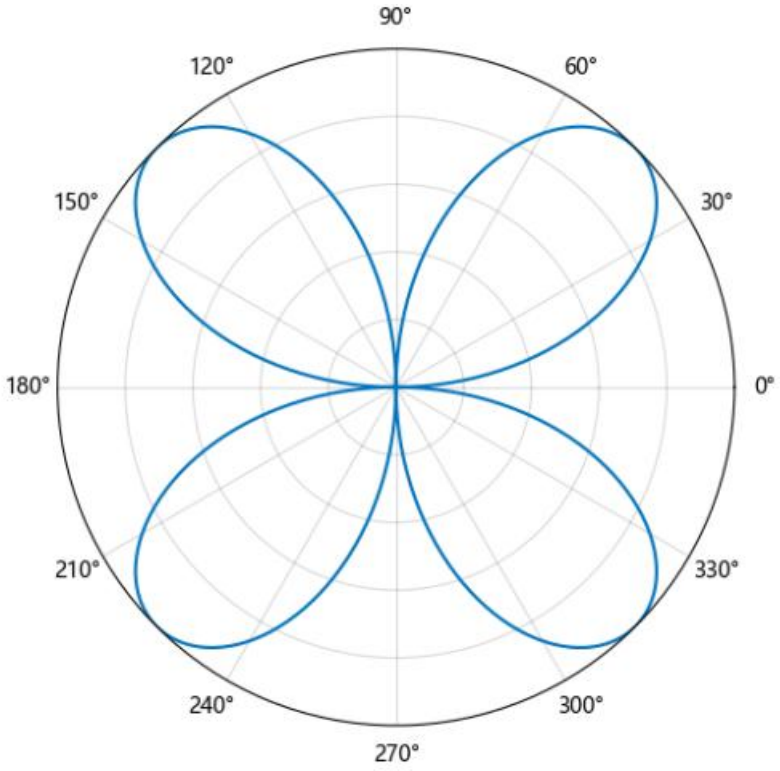

```
rticks("auto")
rticklabels("auto")
```



(4). 删除 r 轴刻度标签

通过指定刻度标签为空数组，删除 r 轴上的刻度标签。

```
theta = 0:0.01:2*pi;
rho = 50 .*sin.(2 .*theta);
polarplot(theta,rho)
rticklabels([])
```



thetaticklabels

设置或查询 theta 轴刻度标签

A. 语法

```
thetaticklabels(labels)
tl = thetaticklabels()

thetaticklabels("auto")
thetaticklabels("manual")

___ = thetaticklabels(pax,___)
```

B. 说明

thetaticklabels(labels)设置当前坐标区的 theta 轴刻度标签。可将 labels 指定

为字符串数组或字符向量数组，例如["E","N","W","S"]。如果指定标签，则 theta 轴刻度值和刻度标签不再基于坐标区的更改而自动更新。

tl = thetaticklabels() 返回当前坐标区的 theta 轴刻度标签。

thetaticklabels("auto") 设置自动模式，使坐标区确定 theta 轴刻度标签。如果您设置了标签，然后又想将它们设置回默认值，则可以使用此选项。

thetaticklabels("manual") 设置手动模式，将 theta 轴刻度标签冻结在当前值。

___ = thetaticklabels(pax,___)使用 pax 指定的坐标区，而不是使用当前坐标区。指定 ax 作为上述任何语法的第一个输入参数。

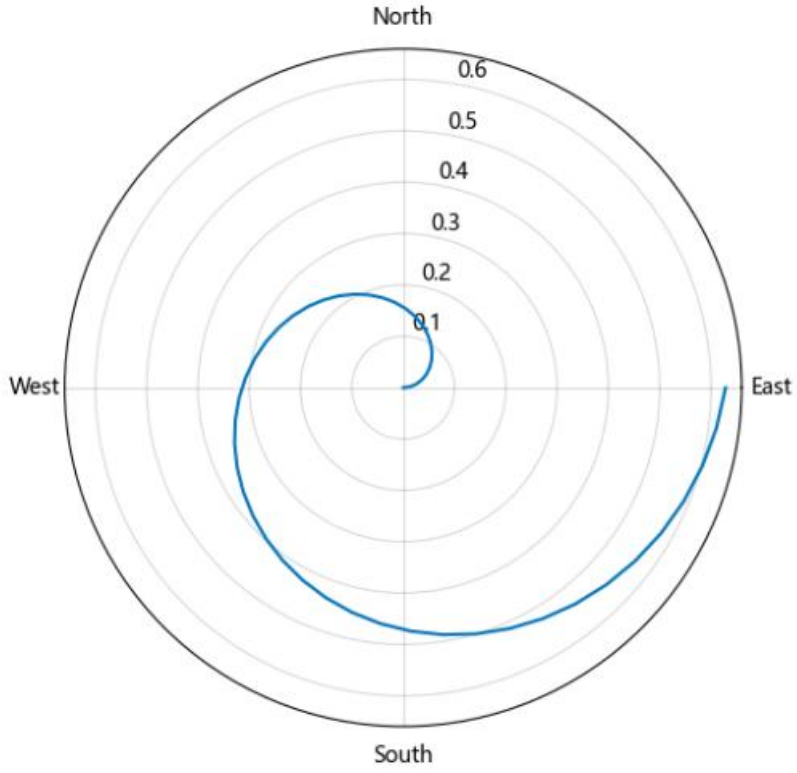
C. 示例

(1). 指定 theta 轴刻度值和标签

创建一个极坐标图。在 0 度、90 度、180 度和 270 度处显示沿 theta 轴的刻度线和网格线。然后为每个刻度线指定一个标签。

```
theta = LinRange(0, 2 * pi, 50);
rho = theta ./ 10;
polarplot(theta, rho)

thetaticks([0 90 180 270])
thetaticklabels(["East","North","West","South"])
```

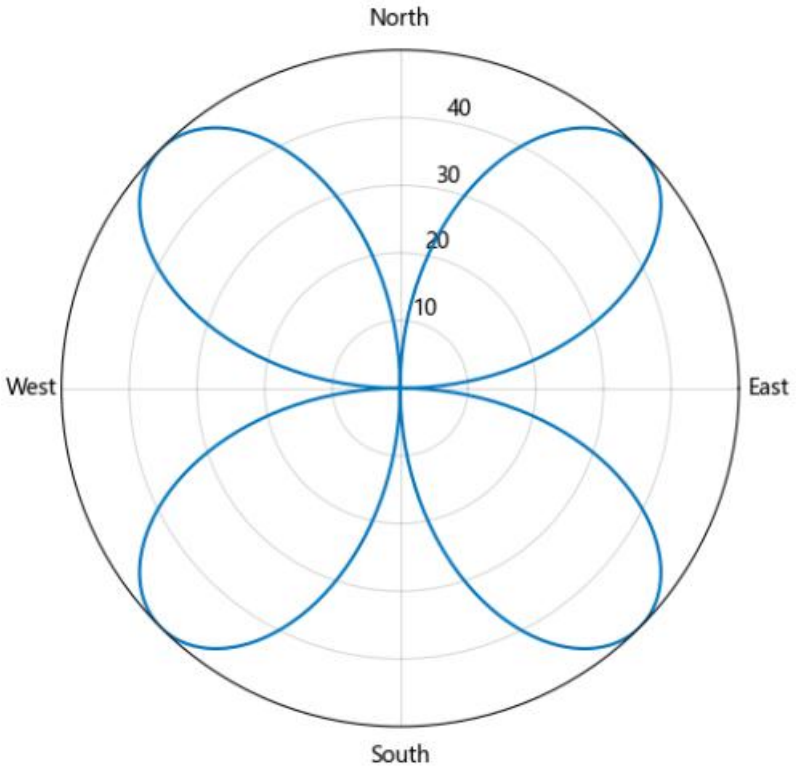


(2). 设置特定极坐标区的 theta 轴刻度标签

创建一个极坐标图，并将极坐标区对象赋给变量 pax。然后，指定极坐标区的 theta 轴刻度值和标签。通过将极坐标区对象作为 thetaticks 和 thetaticklabels 函数的第一个输入参数进行传递，确保这两个函数作用于 pax。

```
theta = 0:0.01:2*pi;
rho = 50 .*sin.(2 .*theta);
polarplot(theta,rho)
pax = gca();

thetaticks(pax,[0 90 180 270])
thetaticklabels(pax, ["East","North","West","South"])
```

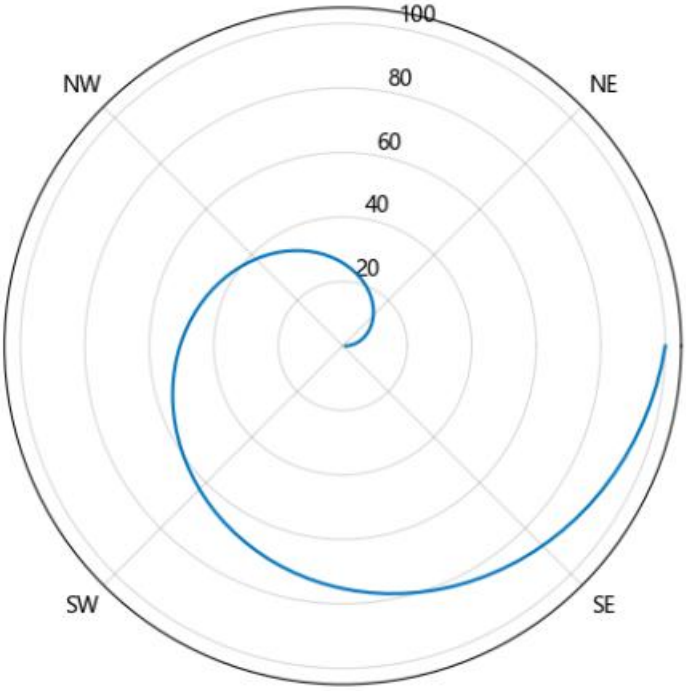


(3). 将 theta 轴刻度标签设置回默认标签

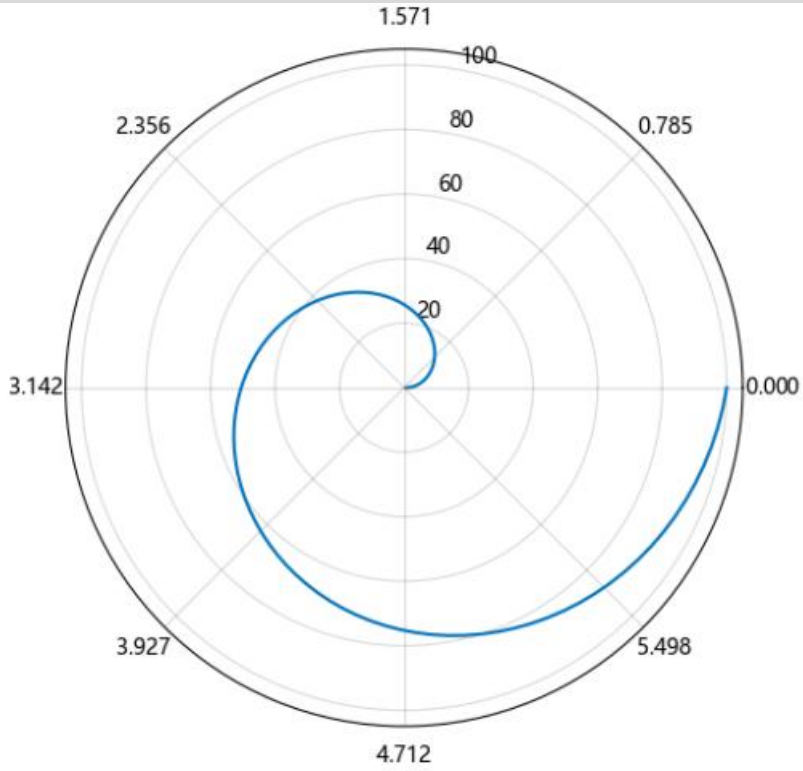
创建一个极坐标图并指定 theta 轴刻度值和对应的标签。然后，将 theta 轴刻度值和标签设置回默认值。

```
polarplot(1:100)
pax = gca();

thetaticks([45 135 225 315])
thetaticklabels(pax, ["NE","NW","SW","SE"])
```




```
thetaticks("auto")
thetaticklabels("auto")
```

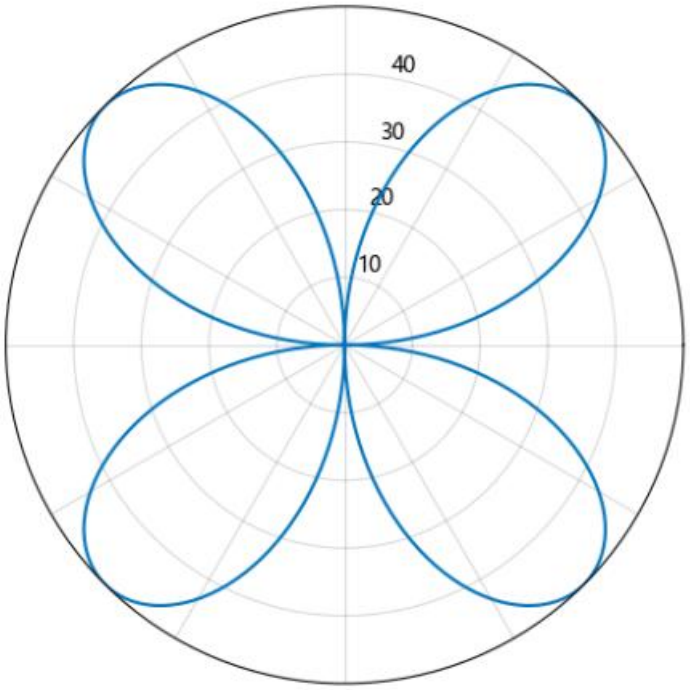


(4). 删除 theta 轴刻度标签

通过将刻度标签指定为空数组来删除沿 theta 轴的刻度标签。

```
theta = 0:0.01:2*pi;
rho = 50 * sin.(2*theta);
```

```
polarplot(theta,rho)
thetaticklabels([])
```



rtickformat

指定 r 轴刻度标签格式

A. 语法

```
rtickformat(fmt)
rtickformat(pax,___)

rfmt = rtickformat()
rfmt = rtickformat(pax)
```

B. 说明

rtickformat(fmt)设置 r 轴刻度标签的格式。例如，将 fmt 指定为"usd"将以美元符号显示标签。

rtickformat(pax,___)使用 pax 指定的坐标区，而不是使用当前坐标区。指定 ax 作为第一个输入参数。

rfmt = rtickformat() 返回当前坐标区的 r 轴刻度标签使用的格式。根据 r 轴上的标签类型，rfmt 是包含一个数值格式的字符向量或字符串。

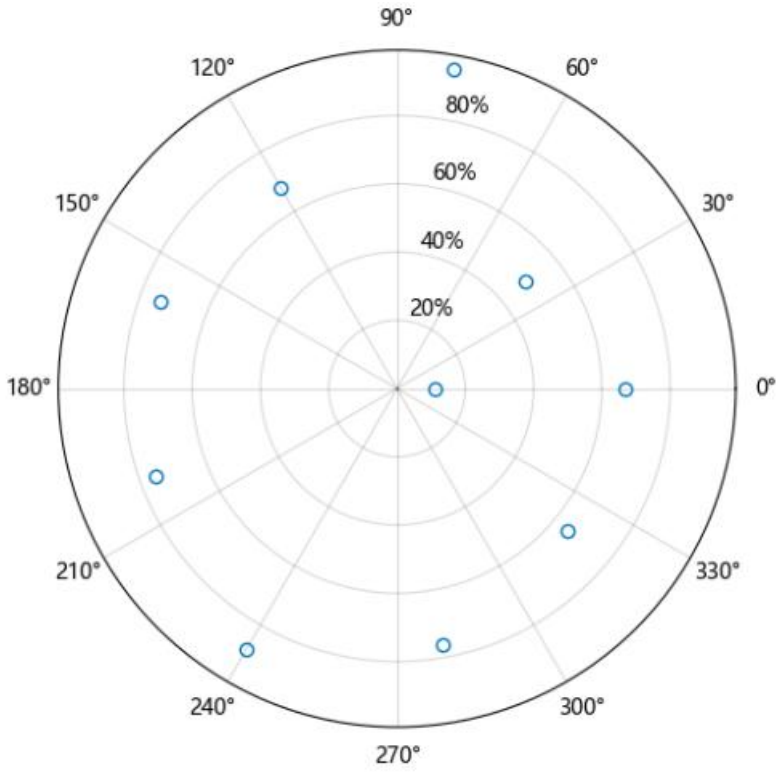
rfmt = rtickformat(pax) 返回 pax 指定的坐标区所使用的格式样式，而不是返回当前坐标区使用的格式样式。

C. 示例

(1). 将 r 轴刻度标签显示为百分比值

创建一个极坐标图。将 r 轴上的刻度标签显示为百分比值。

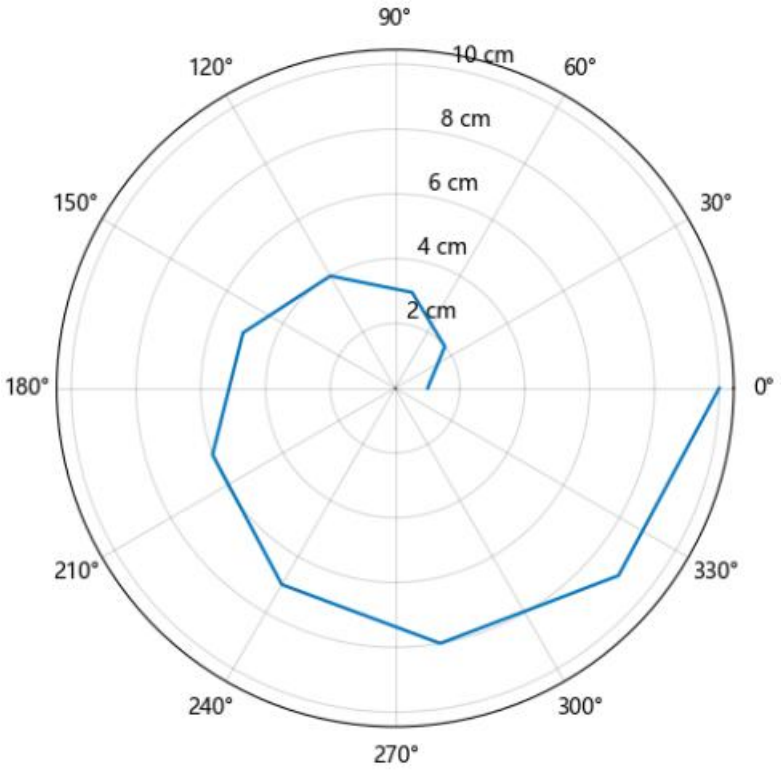
```
th = LinRange(0, 2 * pi, 10);
r = [11, 49, 95, 68, 74, 75, 88, 76, 65, 67];
polarplot(th, r, "o")
rtickformat("percentage")
```



(2). 在每个 r 轴刻度标签之后显示文本

显示 r 轴上的刻度标签，并在每个值之后显示文本“cm”。

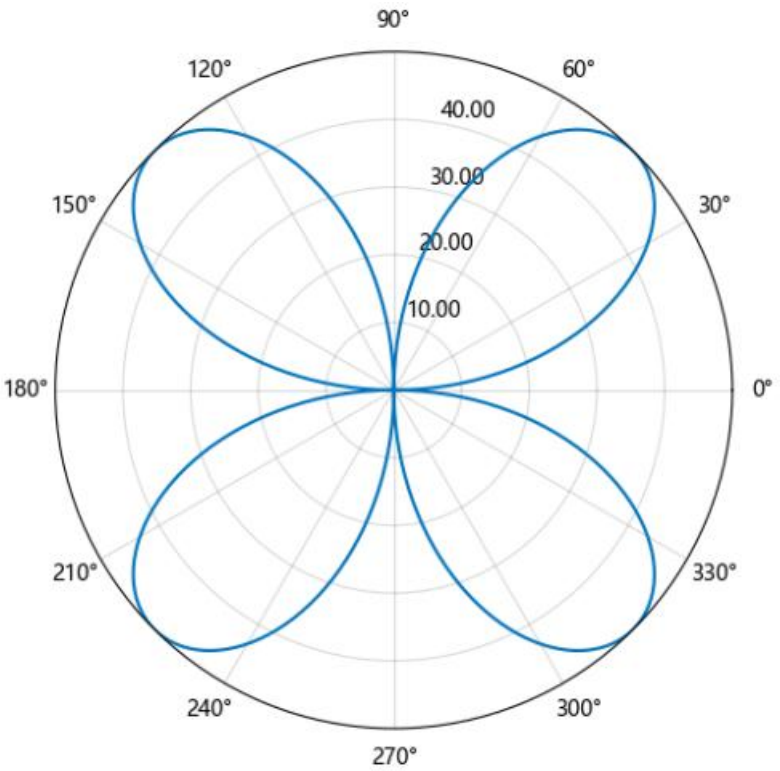
```
polarplot(1:10)
rtickformat("%g cm")
```



(3). 控制 r 轴刻度标签的小数位数

显示带有两位小数的 r 轴刻度标签。通过向 rtickformat 传递一个数值格式的字符向量来控制小数位数，该数值格式对转换字符使用定点记数法且精度值为 2。

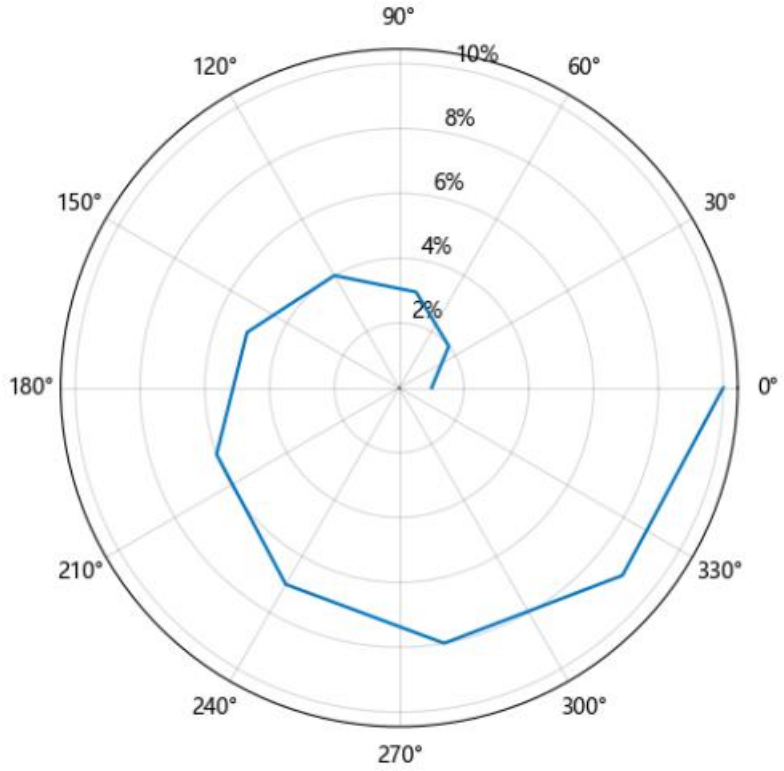
```
theta = 0:0.01:2*pi;
rho = 50 .*sin.(2 .*theta);
polarplot(theta,rho)
rtickformat("%.2f")
```



(4). 指定特定极坐标区的 r 轴刻度标签格式

创建一个极坐标图，并将极坐标区对象赋给变量 pax。通过将 pax 作为 rtickformat 函数的第一个输入参数进行传递，确保该函数作用于您刚创建的极坐标区。

```
polarplot(1:1:10)
pax = gca()
rtickformat(pax, "percentage")
```



thetatickformat

指定 theta 轴刻度标签格式

A. 语法

```
thetatickformat(fmt)
thetatickformat(pax,___)

tfmt = thetatickformat()
tfmt = thetatickformat(pax)
```

B. 说明

thetatickformat(fmt)设置 theta 轴刻度标签的格式。例如，将 fmt 指定为"usd"将以美元符号显示标签。

thetatickformat(pax,___)使用 ax 指定的坐标区，而不是使用当前坐标区。指定 pax 作为第一个输入参数。

tfmt = thetatickformat() 返回当前坐标区的 theta 轴刻度标签所使用的格式样式。根据 theta 轴的标签类型，tfmt 为数值格式的字符向量。

tfmt = thetatickformat(pax) 返回 pax 指定的坐标区所使用的格式样式，而

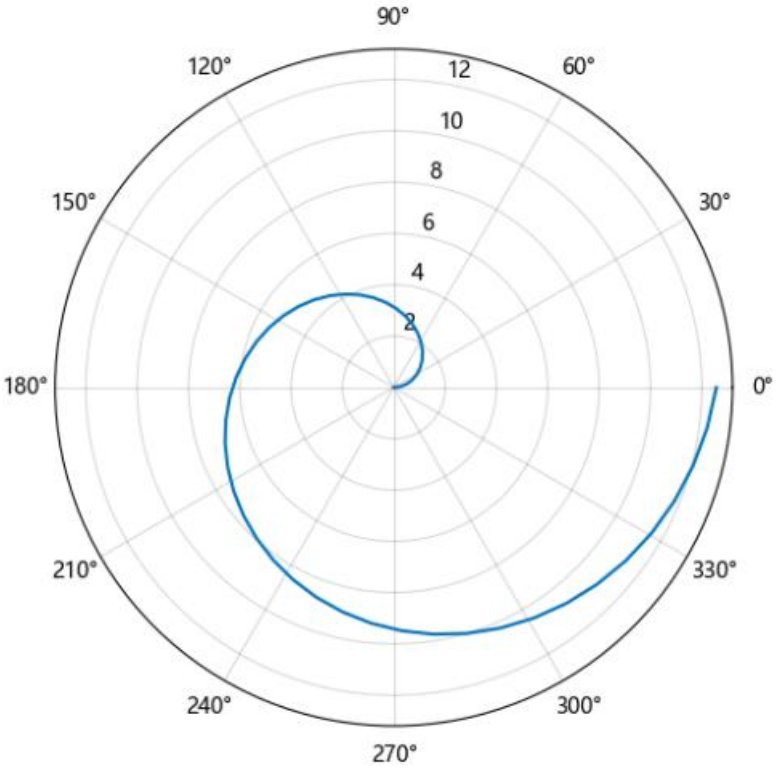
不是返回当前坐标区使用的格式样式。

C. 示例

(1). 以度的形式显示 theta 轴刻度标签

创建一个极坐标图。以度值的形式显示沿 theta 轴的刻度标签。

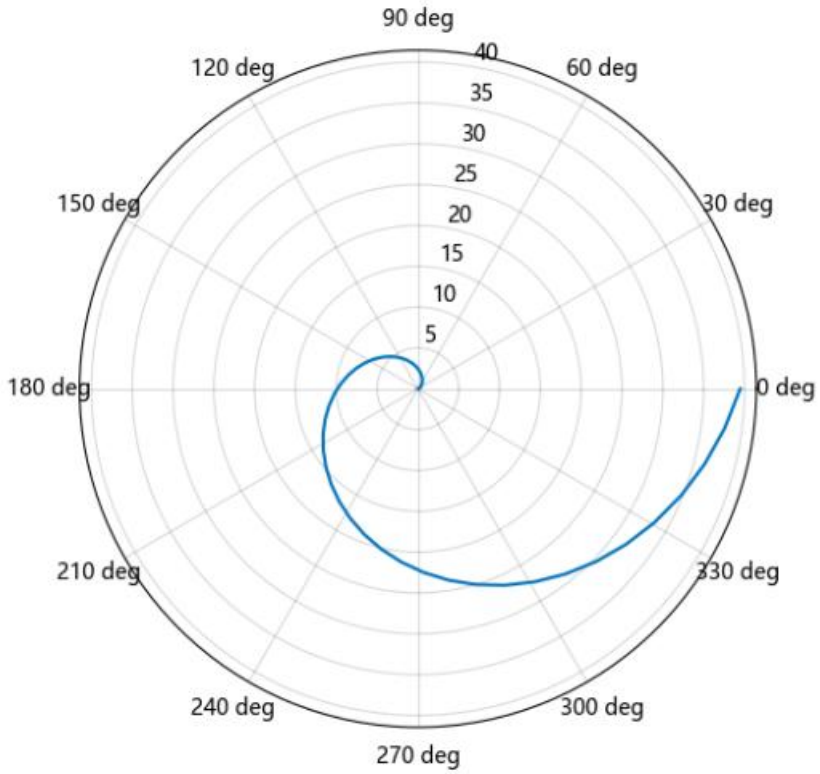
```
th = LinRange(0, 2 * pi, 50);
r = 2 .* th;
polarplot(th, r)
thetatickformat("degrees")
```



(2). 在每个 theta 轴刻度标签后显示文本

创建一个极坐标图。显示沿 theta 轴的刻度标签，并在每个值后附加文本“deg”。

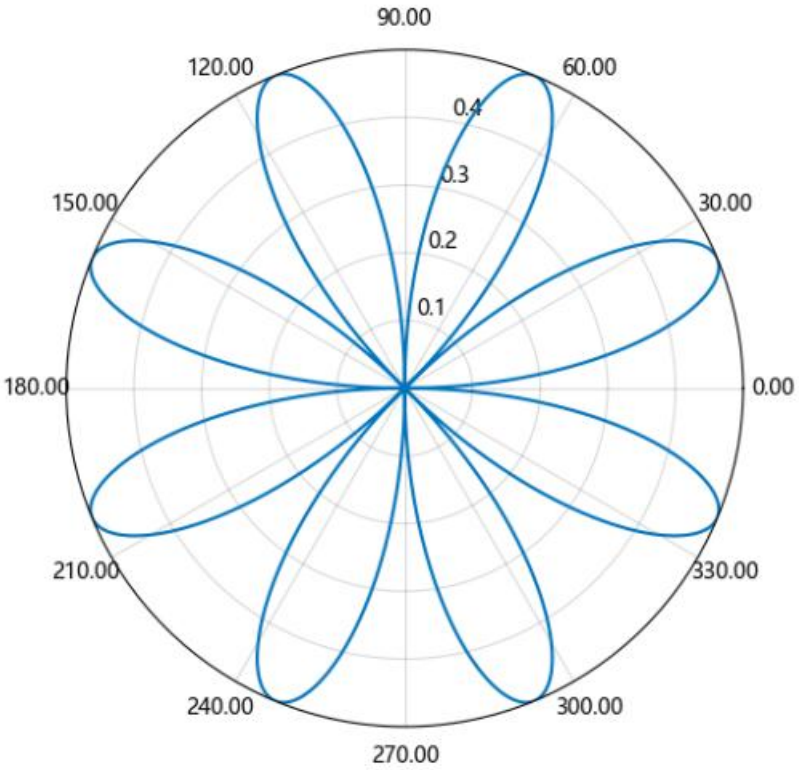
```
th = LinRange(0, 2 * pi, 50);
r = th .^ 2;
polarplot(th, r)
thetatickformat("%g deg")
```



(3). 控制 theta 轴刻度标签的小数位数

显示带两位小数的 theta 轴刻度标签。通过向 thetatickformat 传递一个数值格式的字符向量来控制小数位数，该数值格式对转换字符使用定点记数法且精度值为 2。

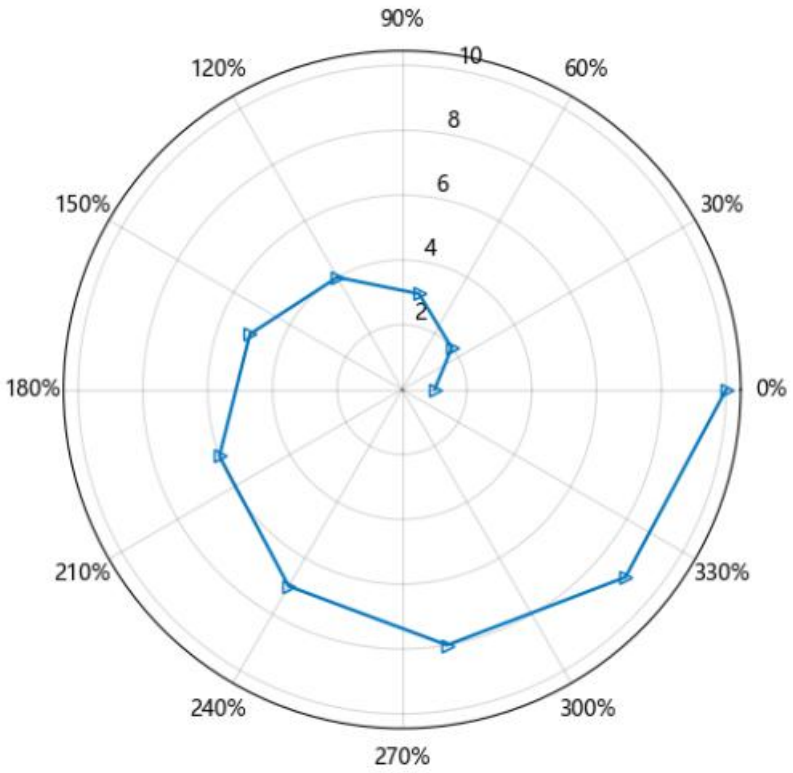
```
theta = 0:0.01:2*pi;
rho = sin.(2 .*theta).*cos.(2 .*theta);
polarplot(theta,rho)
thetatickformat("%.2f")
```



(4). 指定特定极坐标区的 theta 轴刻度标签格式

创建一个极坐标图，并将极坐标区对象赋给变量 pax。通过将 pax 作为 thetatickformat 函数的第一个输入参数进行传递，确保该函数作用于您刚创建的极坐标区。

```
polarplot(1:1:10, "->")
pax = gca()
thetatickformat(pax, "percentage")
```



rtickangle

旋转 r 轴刻度标签

A. 语法

```
rtickangle(angle)
rtickangle(pax,angle)

ang = rtickangle()
ang = rtickangle(pax)
```

B. 说明

rtickangle(angle)将当前坐标区的 r 轴刻度标签旋转到指定的角度，其中 0 表示水平。指定正值表示逆时针旋转，负值表示顺时针旋转。

rtickangle(pax,angle)旋转 pax 指定的坐标区的刻度标签，而不是旋转当前坐标区的刻度标签。

ang = rtickangle() 以度数标量值的形式返回当前坐标区的 r 轴刻度标签的旋转角度。正值表示逆时针旋转。负值表示顺时针旋转。

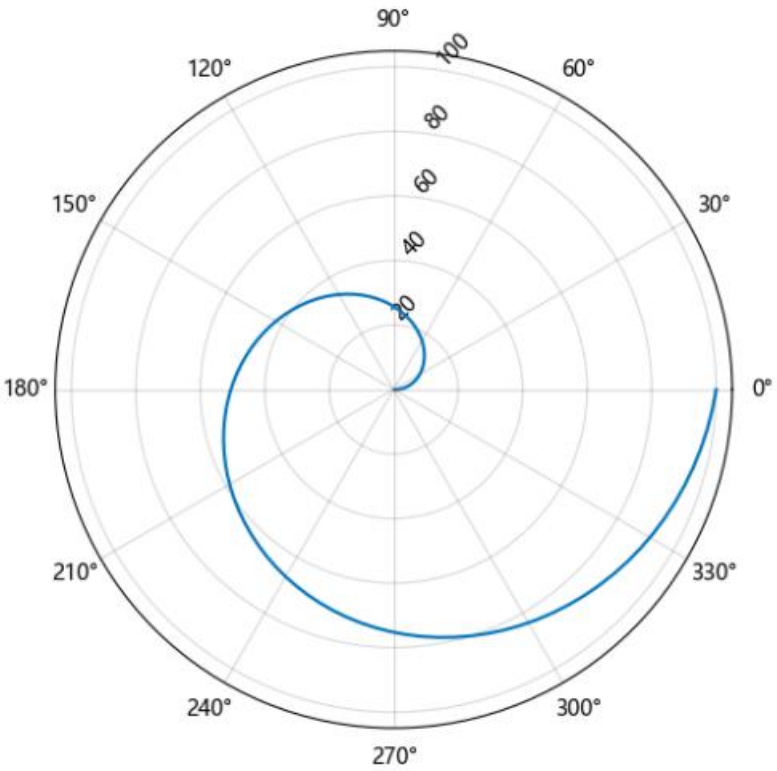
ang = rtickangle(pax)使用 pax 指定的极坐标区，而不是使用当前坐标区。

C. 示例

(1). 旋转 r 轴刻度标签

创建一个极坐标图，然后将 r 轴刻度标签逆时针旋转 45 度。

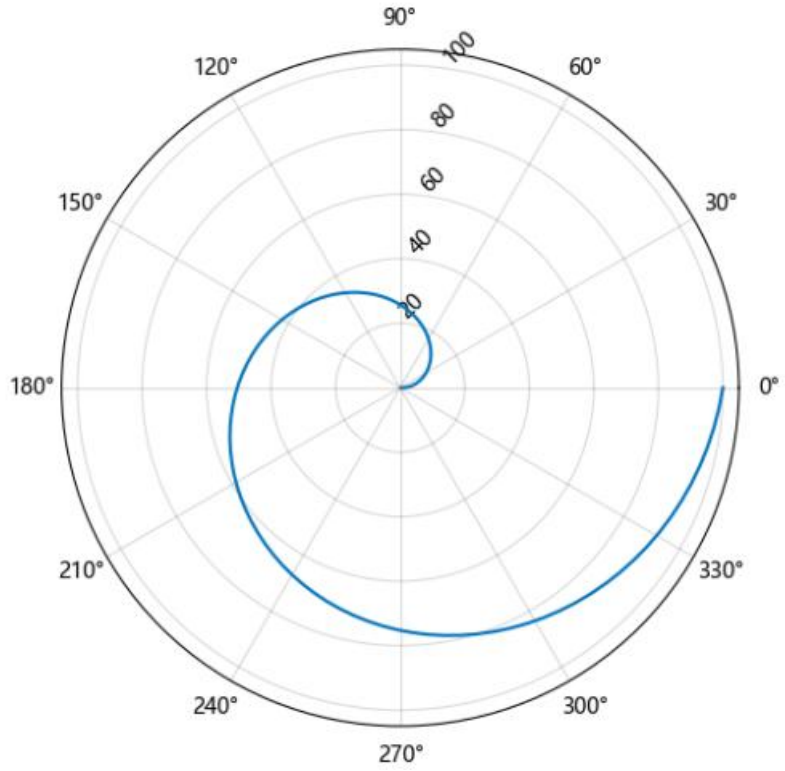
```
polarplot(1:100)
rtickangle(45)
```



(2). 旋转特定极坐标区的 r 轴刻度标签

创建一个极坐标图，并将极坐标区对象赋给变量 pax。然后旋转极坐标区的 r 轴刻度标签。通过指定极坐标区对象作为第一个输入参数，确保 rtickangle 影响 pax。

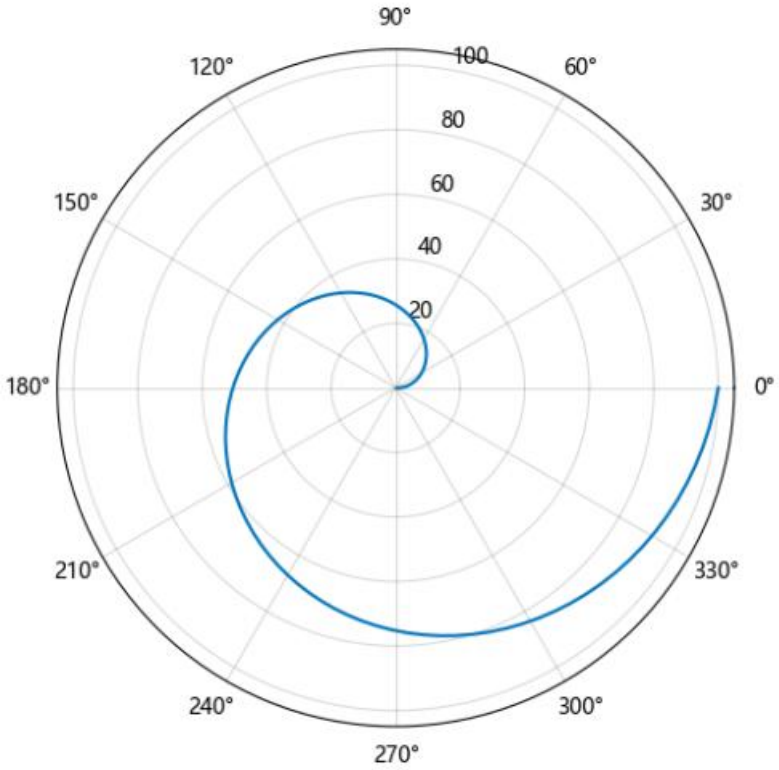
```
polarplot(1:100)
pax = gca();
rtickangle(pax,45)
```



(3). 查询 r 轴刻度标签的当前角度

创建一个极坐标图。然后，查询 r 轴刻度标签的旋转角度。默认情况下，不会旋转标签。

```
polarplot(1:100)
```




```
ang = rtickangle()
```

```
ang = 0
```

polaraxes

创建极坐标区

A. 语法

```
polaraxes()
polaraxes(pos)
polaraxes(Key=Value)
pax = polaraxes(___)

polaraxes(pax_in)
```

B. 说明

polaraxes() 在当前图窗中创建默认的极坐标区。

polaraxes(pos) 根据输入的位置来创建极坐标区。

polaraxes(Key=Value) 使用一个或多个名称-值对组参数指定 PolarAxes 对象的属性。有关属性列表，请参阅 PolarAxes 属性 (opens new window)。

pax = polaraxes(___) 返回创建的 PolarAxes 对象。可使用 pax 在创建 PolarAxes 对象后查询和设置其属性。有关属性列表，请参阅 PolarAxes 属性 (opens new window)。

polaraxes(pax_in)使 PolarAxes 对象 pax_in 成为当前坐标区。

C. 示例

(1). 修改极坐标区属性

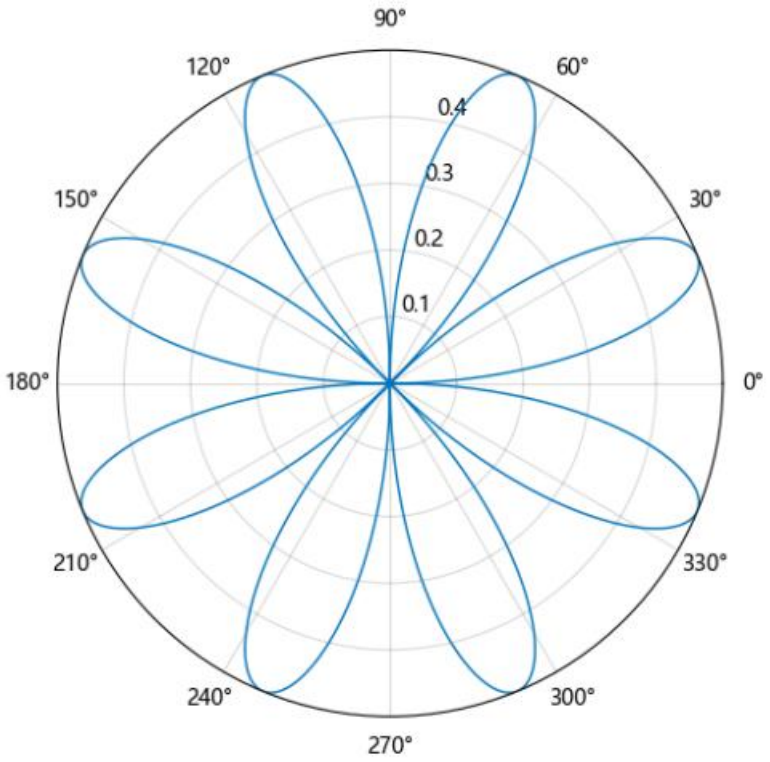
创建带有极坐标区的新图窗，并将极坐标区对象赋予 pax。向坐标区添加绘图。然后使用 pax 修改坐标区属性。

指定第一个 Axes 对象的位置，使其左下角位于点 (0.1 0.1) 处，宽度和高度均为 0.7。指定第二个 Axes 对象的位置，使其左下角位于点 (0.65 0.65) 处，宽度和高度均为 0.28。默认情况下，所有值为基于图窗的归一化值。

将这两个 Axes 对象返回为 ax1 和 ax2。

```
figure()
pax = polaraxes()
theta = 0:0.01:2*pi;
```

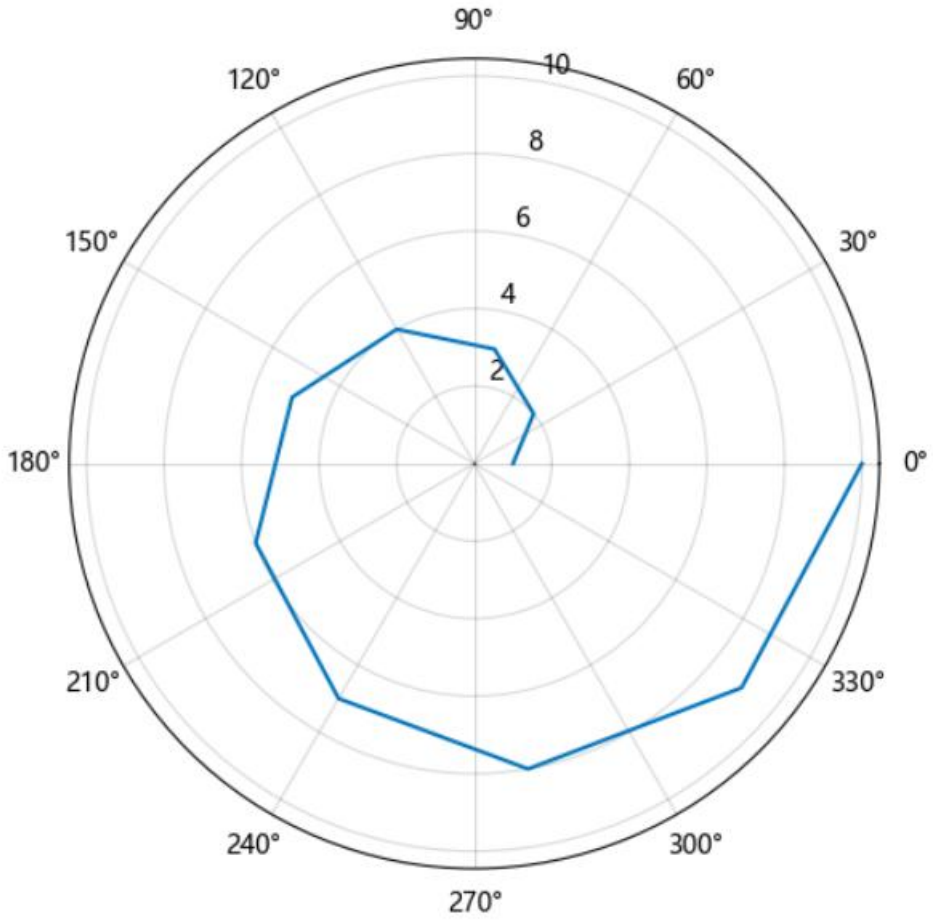
```
rho = sin.(2 * theta) .* cos.(2 * theta);
polarplot(theta, rho)
```



(2). 使极坐标区成为当前坐标区

创建带有极坐标区的图窗，并将极坐标区对象赋给 pax。然后，在调用 polarplot 函数之前，确保 pax 是当前坐标区。

```
figure()
pax = polaraxes()
polaraxes(pax)
polarplot(1:10)
```



3.3.1.5 等高线图

二维和三维等值线图

函数名	简介
contour	矩阵的等高线图
contourf	填充的二维等高线图
contour3	三维等高线图
clabel	为等高线图添加高程标签
fcontour	绘制等高线

3.3.1.5.1 contour

矩阵的等高线图

A. 语法

```
contour(Z)
contour(X,Y,Z)
contour(___,levels)
contour(___,Key=Value)
contour(ax,___)
c= contour(___)
```

B. 说明

contour(Z)创建一个包含矩阵 Z 的等值线的等高线图，其中 Z 包含 x-y 平面上的高度值。Syslab 会自动选择要显示的等高线。Z 的列和行索引分别是平面中的 x 和 y 坐标。

contour(X,Y,Z)指定 Z 中各值的 x 和 y 坐标。

contour(___,levels)将要显示的等高线指定为上述任一语法中的最后一个参数。将 levels 指定为标量值 n，以在 n 个自动选择的层级（高度）上显示等高线。要在某些特定高度绘制等高线，请将 levels 指定为单调递增值的向量。要在一个高度(k)绘制等高线，请将 levels 指定为二元素行向量[k]。

contour(___,Key=Value)使用一个或多个名称-值对组参数指定等高线图的其他选项。请在所有其他输入参数之后指定这些选项。

contour(ax,___) 在目标坐标区中显示等高线图。将坐标区指定为上述任一语法中的第一个参数。

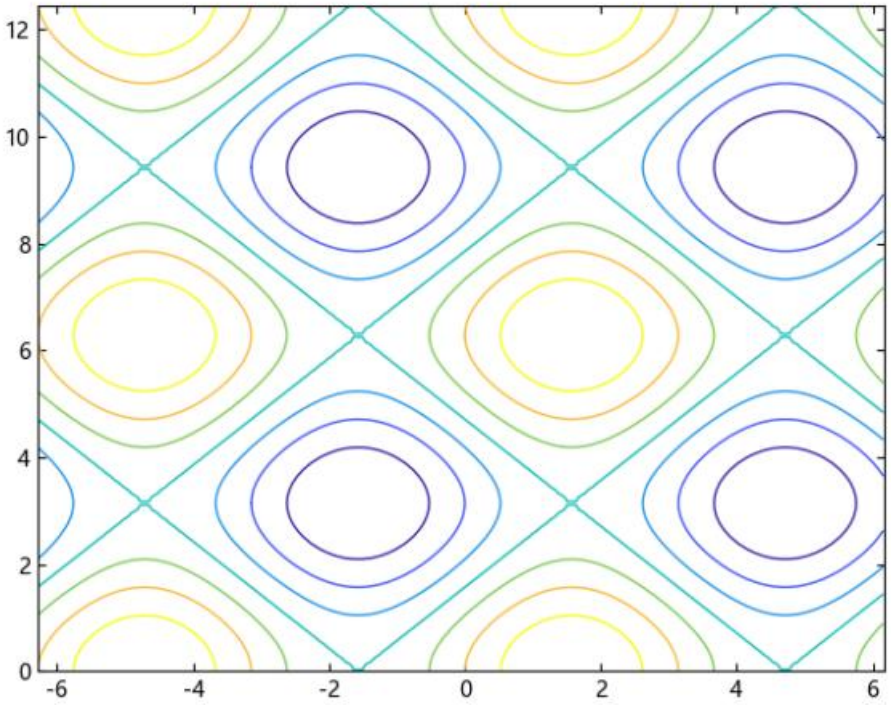
c=contour(___)返回等高线对象 c。显示等高线图后，使用 c 设置属性。

C. 示例

(1). 函数的等高线

创建矩阵 X 和 Y，用于在 x-y 平面中定义一个网格。将矩阵 Z 定义为该网格上方的高度。然后绘制 Z 的等高线。

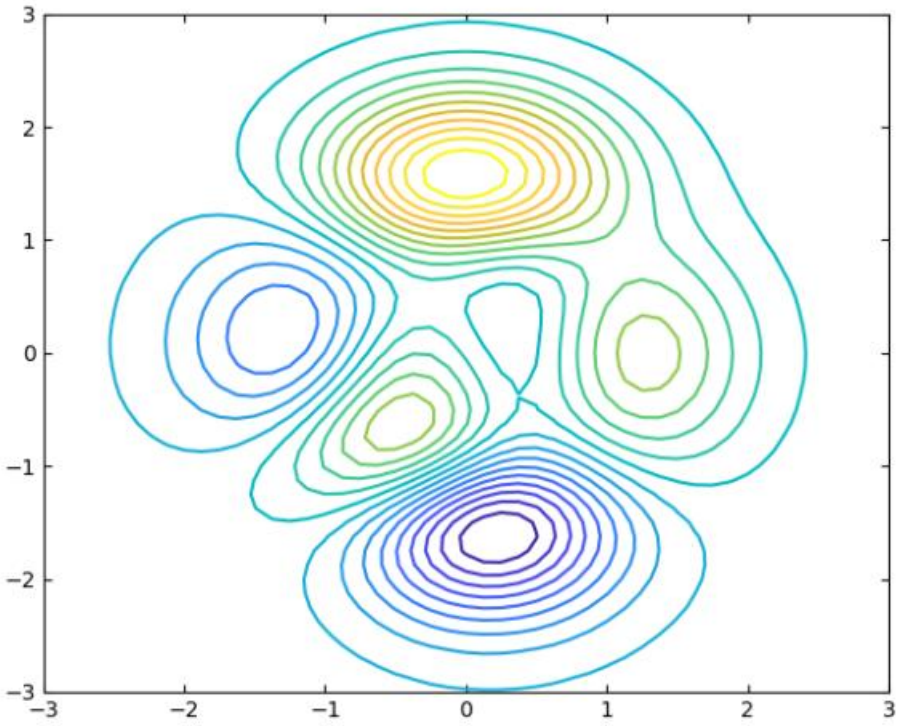
```
using TyPlot
using TyMath
xs = collect((-2 * pi):((4 * pi) / 100):(2 * pi))
ys = collect(0:((4 * pi) / 100):(4 * pi))
X, Y = meshgrid2(xs, ys)
Z = sin.(X) + cos.(Y)
contour(X, Y, Z)
```



(2). 二十个层级的等高线

将 Z 定义为 X 和 Y 的函数。在本例中，调用 peaks 函数以创建 X、Y 和 Z。然后绘制 Z 的 20 个等高线。

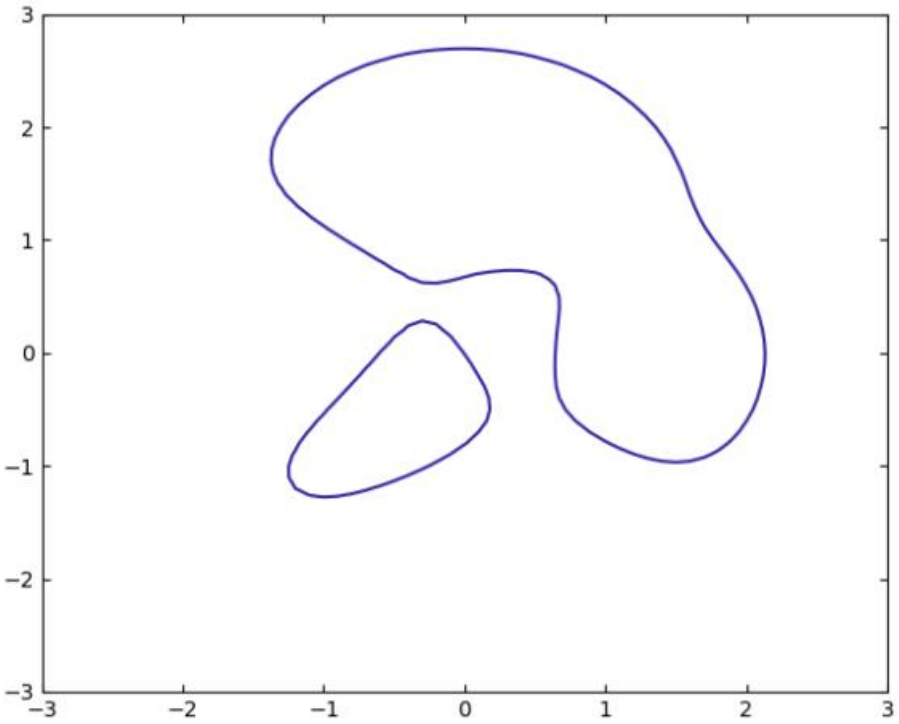
```
using TyPlot
X, Y, Z = peaks()
con = contour(X, Y, Z, 20)
```



(3). 一个层级的等高线

显示 peaks 函数在 $Z = 1$ 处的等高线。

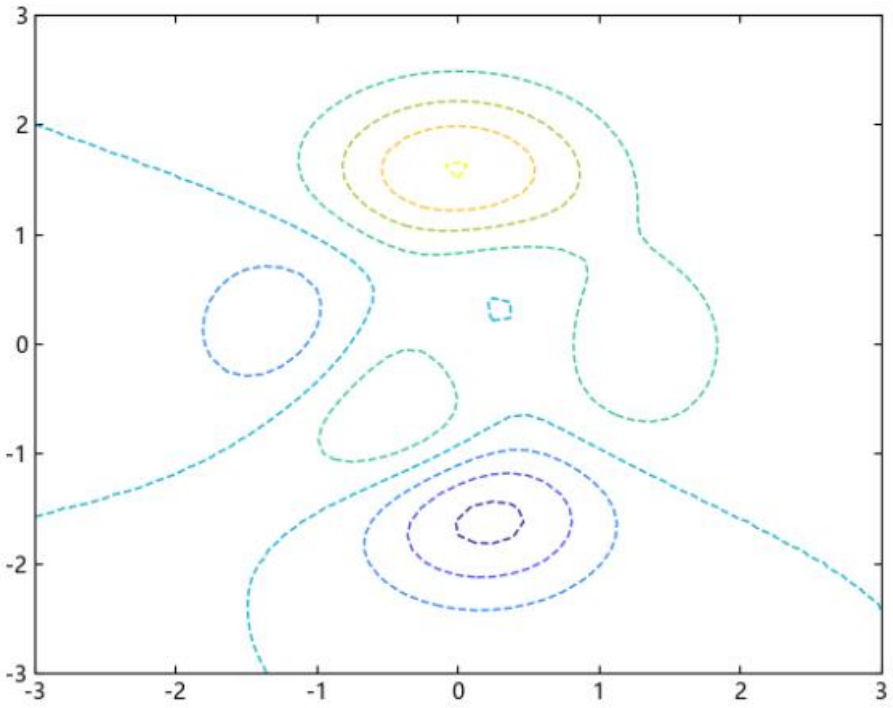
```
using TyPlot
X, Y, Z = peaks()
con = contour(X, Y, Z, [1])
```



(4). 虚线等高线

创建 peaks 函数的等高线图，并指定虚线线型。

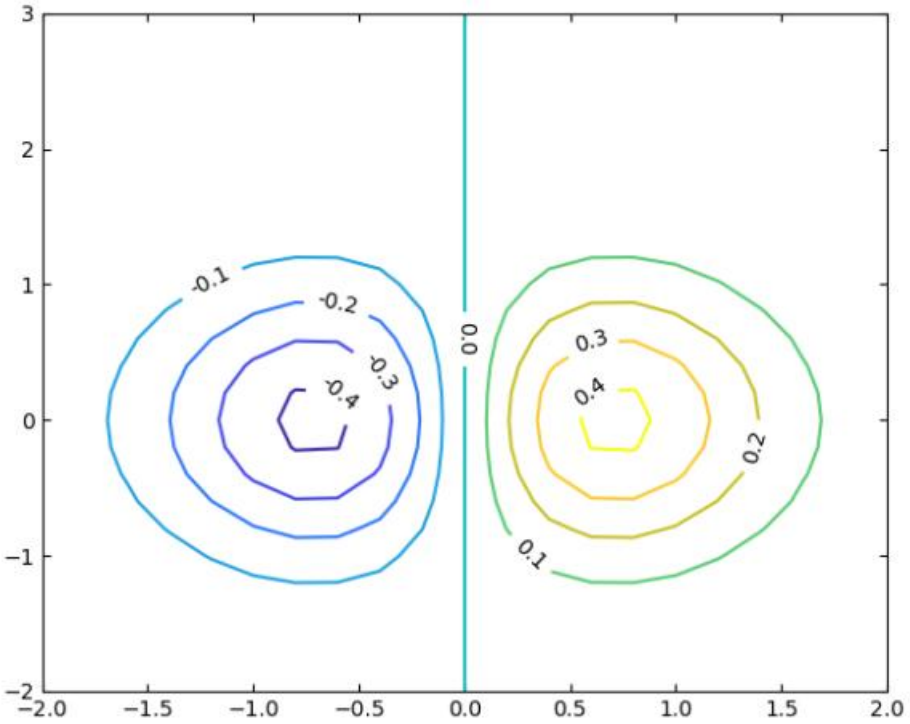
```
using TyPlot
X, Y, Z = peaks()
contour(X, Y, Z; linestyle="--")
```



(5). 带标签的等高线

将 Z 定义为两个变量 X 和 Y 的函数。然后创建该函数的等高线图，并显示标签。

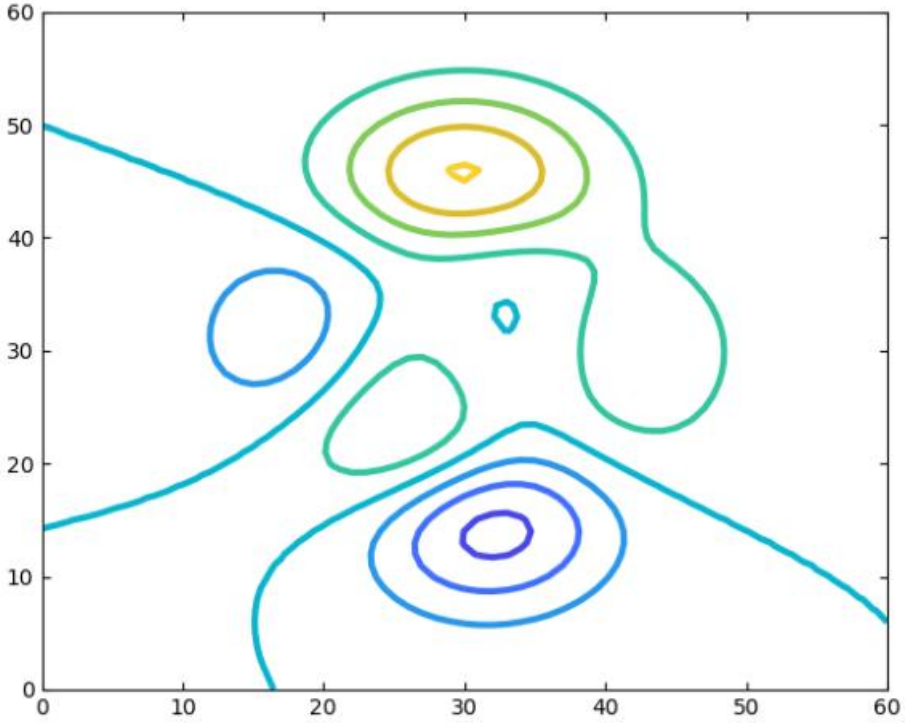
```
using TyPlot
x = -2:0.2:2;
y = -2:0.2:3;
X = [x for x in x for y in y]
Y = [y for x in x for y in y]
Z = X .* exp.(-X .^ 2 .- Y .^ 2);
XM = reshape(X, (length(y), length(x)))
YM = reshape(Y, (length(y), length(x)))
ZM = reshape(Z, (length(y), length(x)))
con = contour(
    XM, YM, ZM; levels=[-0.4, -0.3, -0.2, -0.1, 0, 0.1, 0.2, 0.3,
0.4], showtext="on"
)
```



(6). 自定义线宽

创建 peaks 函数的等高线图。通过将 linewidth 属性设置为 3，使等高线更粗。

```
using TyPlot
X, Y, Z = peaks()
con = contour(Z);
for collection in con.collections
    collection.set_linewidth(3)
    collection.set_label("label")
end
```

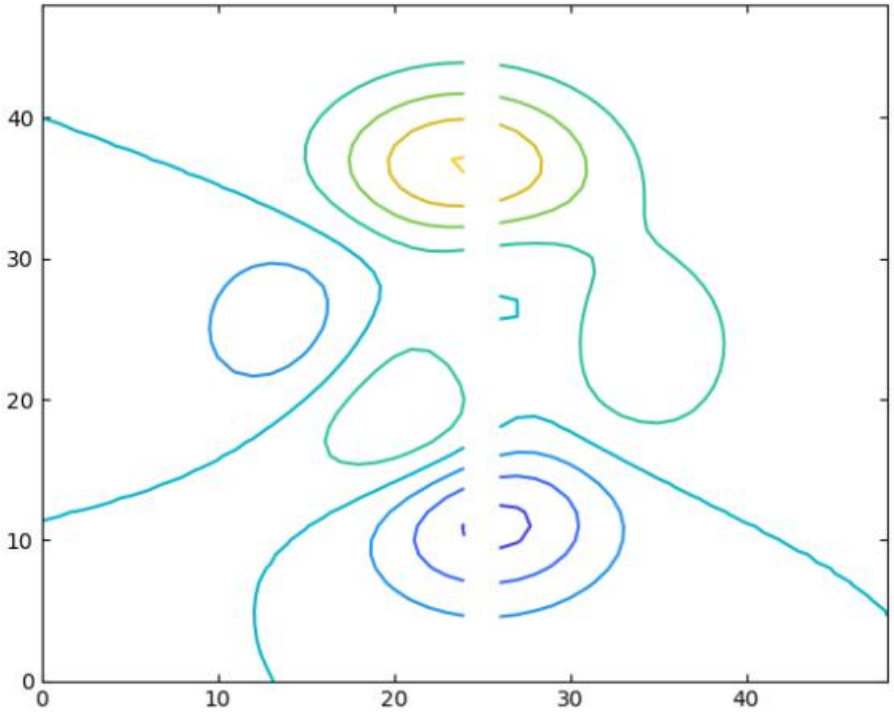


(7). 不连续曲面上的等高线

在曲面上任何不连续的位置插入 NaN 值。contour 函数不会在这些区域中绘制等高线。

将矩阵 Z 定义为 peaks 函数的采样。将列 26 中的所有值替换为 NaN 值。然后绘制修改后的 Z 矩阵的等高线。

```
using TyPlot
X, Y, Z = peaks()
Z[:, 26] .= NaN;
con = contour(Z);
```



3.3.1.5.2 contourf

填充的二维等高线图

A. 语法

```
contourf(Z)
contourf(X,Y,Z)
contourf(___,levels)
contourf(___,Key=Value)
contourf(ax,___)
C = contourf(___)
```

B. 说明

contourf(Z)创建一个包含矩阵 Z 的等值线的填充等高线图，其中 Z 包含 x-y 平面上的高度值。Syslab 会自动选择要显示的等高线。Z 的列和行索引分别是平面中的 x 和 y 坐标。

contourf(X,Y,Z) 指定 Z 中各值的 x 和 y 坐标。

contourf(___,levels)将要显示的等高线指定为上述任一语法中的最后一个参数。将 levels 指定为标量值 n，以在 n 个自动选择的层级（高度）上显示等高线。要在某些特定高度绘制等高线，请将 levels 指定为单调递增值的向量。要在一个高度(k)绘制等高线，请将 levels 指定为行向量[k]。

contourf(___,Key=Value)使用一个或多个名称-值对组参数指定等高线图的其他选项。请在所有其他输入参数之后指定这些选项。

contourf(ax,___) 在目标坐标区中显示等高线图。将坐标区指定为上述任一语法中的第一个参数。

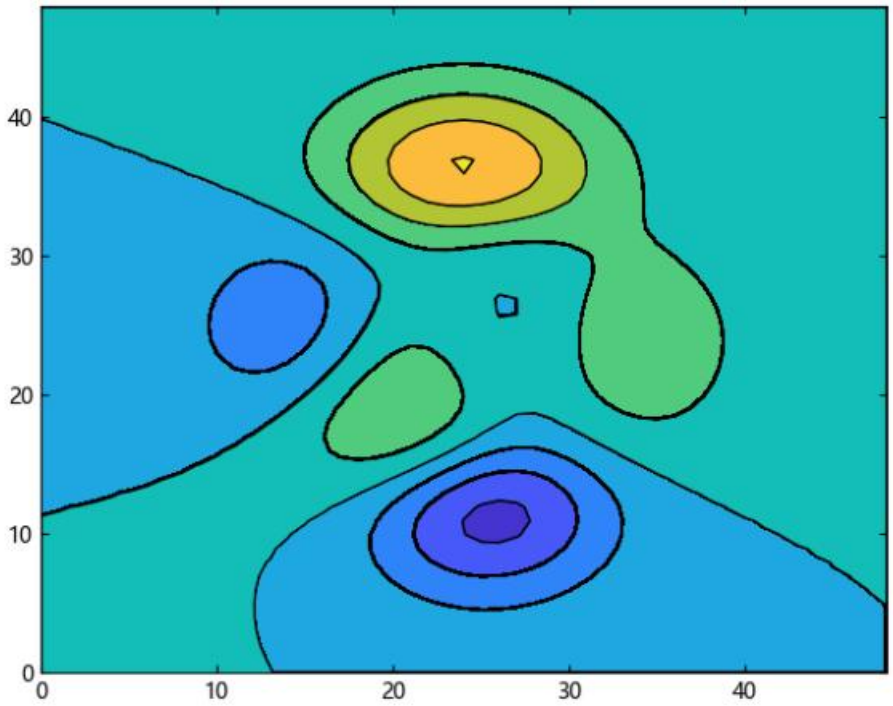
C=contourf(___)返回等高线对象 C。显示等高线图后，使用 C 设置属性。

C. 示例

(1). peaks 函数的等高线

将 Z 定义为两个变量的函数。在本例中，调用 peaks 函数以创建 Z。然后显示 Z 的填充等高线图，让 Syslab 选择等高线层级。

```
X,Y,Z = peaks();
contourf(Z)
```

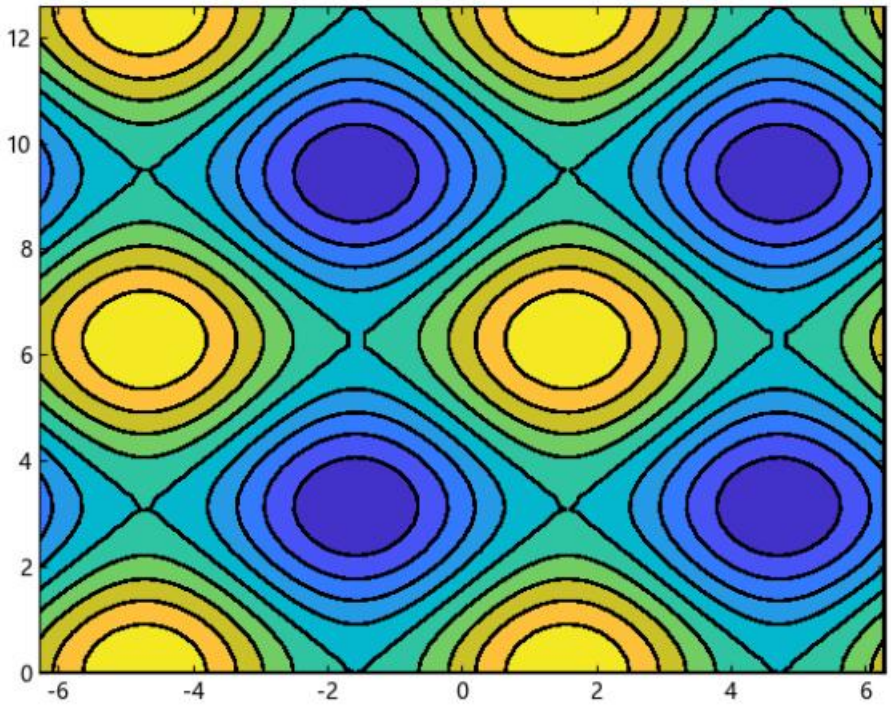


(2). 十个层级的等高线

将 Z 定义为两个变量 X 和 Y 的函数。然后在 Z 的 10 个层级上显示等高线。

```
using TyPlot
using TyMath
x = LinRange(-2 * pi, 2 * pi, 100);
y = LinRange(0, 4 * pi, 50);
X, Y = meshgrid2(x, y);
```

```
Z = sin.(X) .+ cos.(Y);
contourf(X, Y, Z, 10)
```

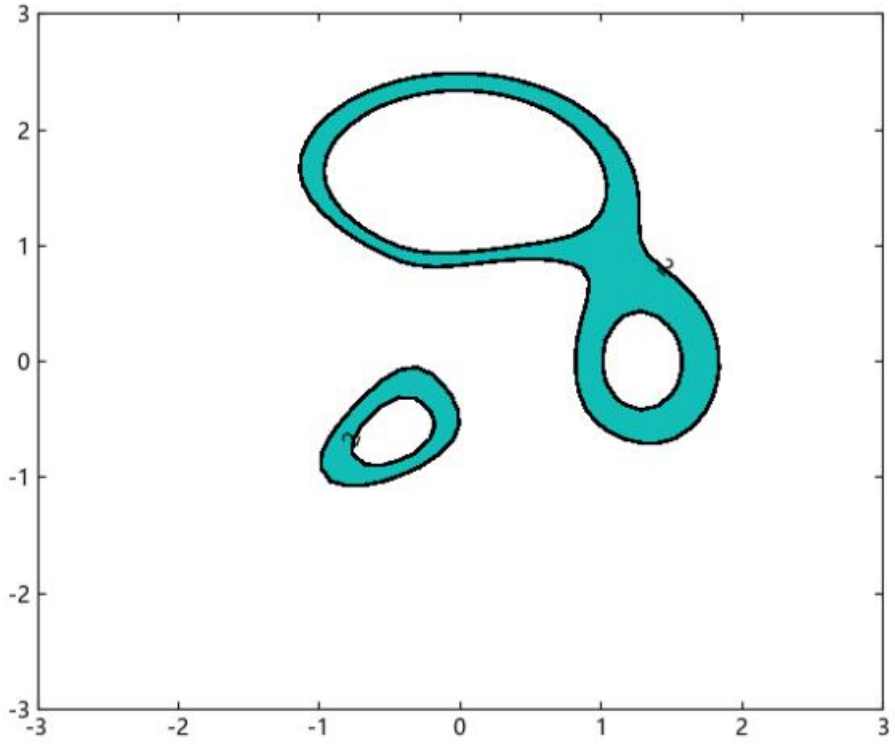


(3). 具有标签的特定层级的等高线

将 Z 定义为 X 和 Y 的函数。在本例中，调用 peaks 函数以创建 X、Y 和 Z。然后显示层级 2 和 3 上的等高线。

绿色区域对应于 2 和 3 的高度。

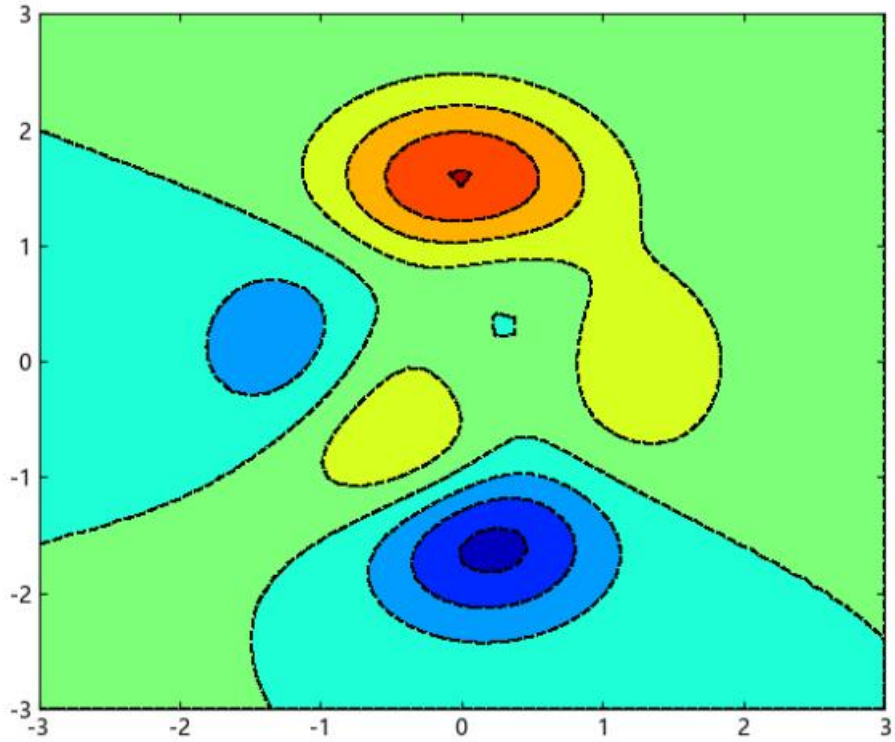
```
X,Y,Z = peaks(50);
con = contourf(X,Y,Z,[2,3])
clabel(con, inline=false, fontsize=10, fmt="%.0f", colors="k")
```



(4). 虚线等高线

创建等高线图，并指定虚线线型。

```
X,Y,Z = peaks();
contourf(X,Y,Z,linestyle = "--", cmap="jet")
```

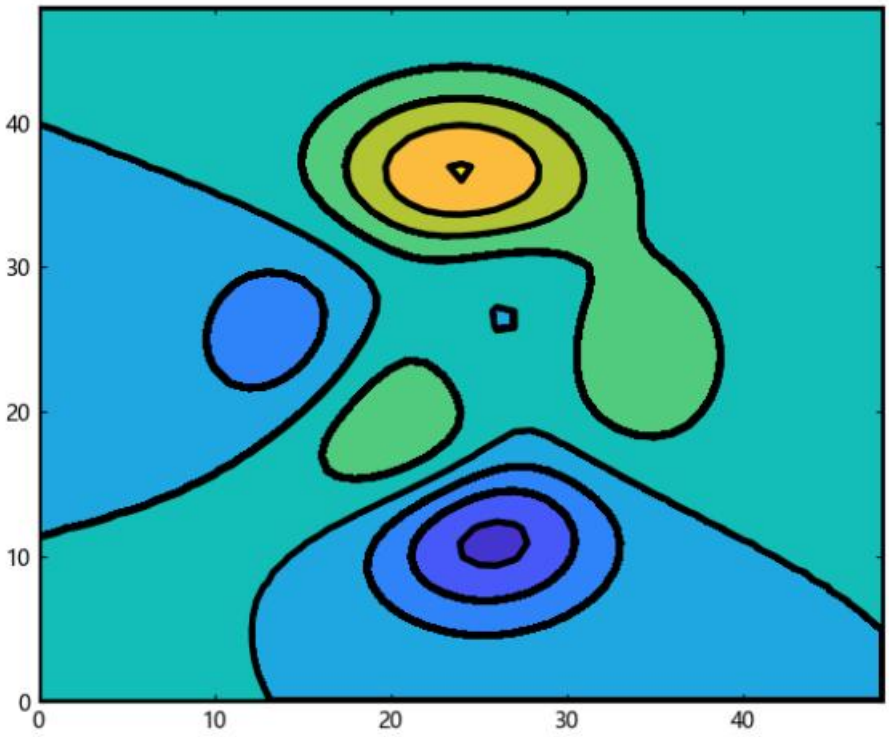


(5). 自定义线宽

创建填充等高线图。通过将 linewidth 属性设置为 3，使等高线更粗。

```
X,Y,Z = peaks()
con = contourf(Z);

for collection in con.collections
   collection.set_linewidth(3)
end
```

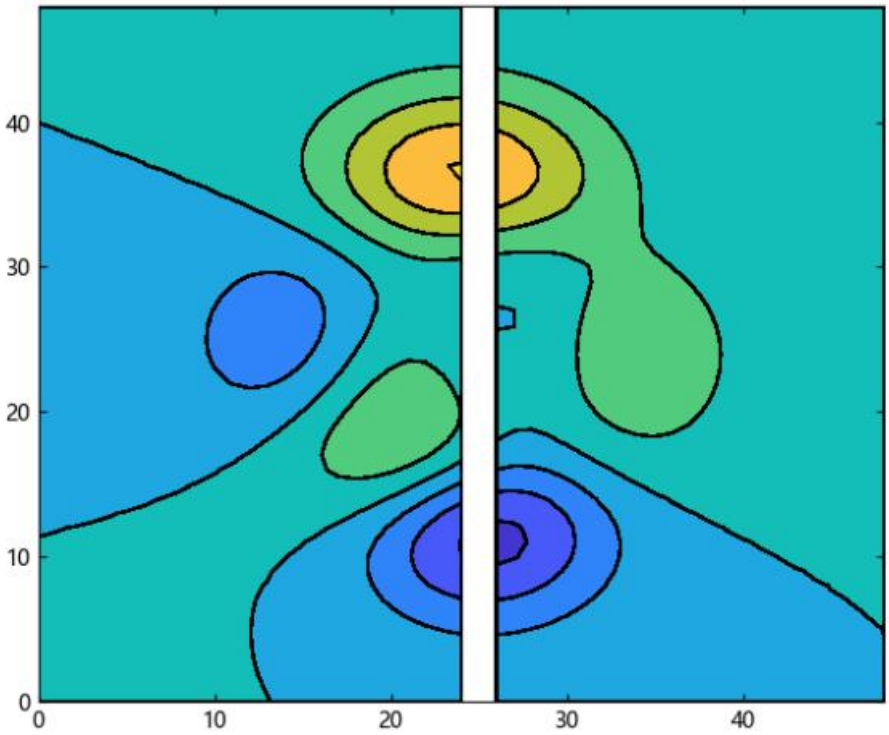


(6). 不连续曲面上的等高线

在曲面上任何不连续的位置插入 NaN 值。contourf 函数不会在这些区域中绘制等高线。

将矩阵 Z 定义为 peaks 函数的采样。将列 26 中的所有值替换为 NaN 值。然后绘制修改后的 Z 矩阵的等高线。

```
X,Y,Z = peaks();
Z[:,26] .= NaN;
con = contourf(Z);
```



3.3.1.5.3 contour3

三维等高线图

A. 语法

```
contour3(Z)
contour3(X,Y,Z)
contour3(___,levels)
contour3(___,Key=Value)
contour3(ax,___)
C= contour3(___)
```

B. 说明

contour3(Z)创建一个包含矩阵 Z 的等值线的三维等高线图，其中 Z 包含 x-y 平面上的高度值。Syslab 会自动选择要显示的等高线。Z 的列和行索引分别是平面中的 x 和 y 坐标。

contour3(X,Y,Z)指定 Z 中各值的 x 和 y 坐标。

contour3(___,levels) 将要显示的等高线指定为上述任一语法中的最后一个参数。将 levels 指定为标量值 n，以在 n 个自动选择的层级（高度）上显示等高线。要在某些特定高度绘制等高线，请将 levels 指定为单调递增值的向量。

要在一个高度 (k) 绘制等高线，请将 levels 指定为行向量 [k]。

contour3(___,Key=Value) 使用一个或多个名称-值对组参数指定等高线图的其他选项。请在所有其他输入参数之后指定这些选项。

contour3(ax,___) 在目标坐标区中显示等高线图。将坐标区指定为上述任一语法中的第一个参数。

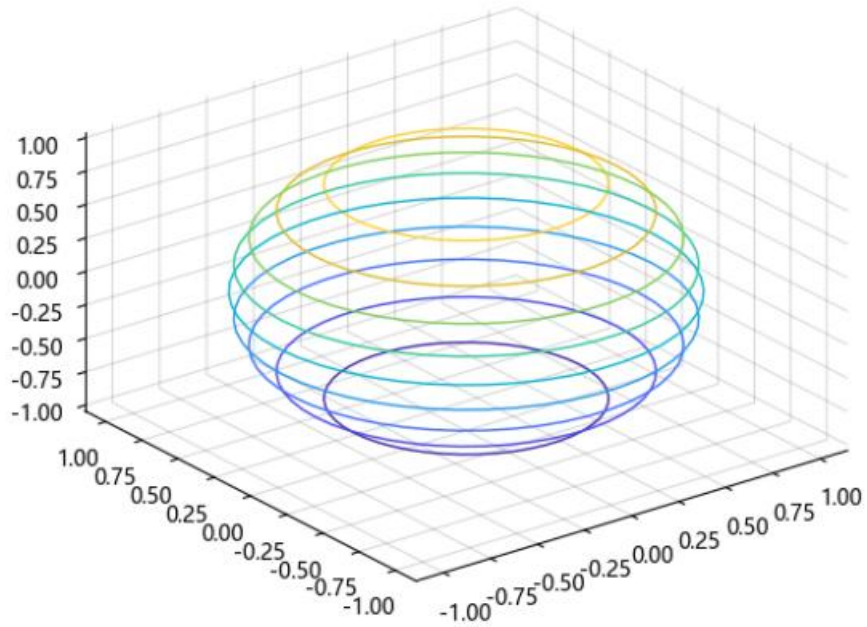
C = contour3(___) 返回等高线对象 c，其中包含每个层级的顶点的 (x, y) 坐标。

C. 示例

(1). 球体的等高线

将 Z 定义为 X 和 Y 的函数。绘制 Z 的等高线。

```
using TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/contour3/0-data.jl"
include(source_path)
contour3(X,Y,Z)
```

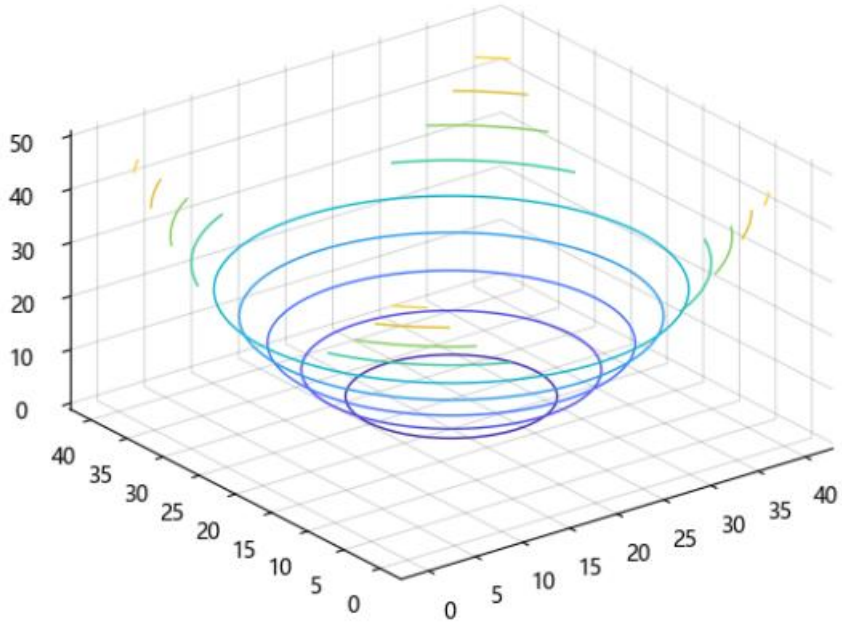


(2). 五十个层级的等高线

将 Z 定义为两个变量 X 和 Y 的函数。然后绘制 Z 的等高线。在本例中，让 Syslab 选择 x 和 y 轴的等高线和范围。

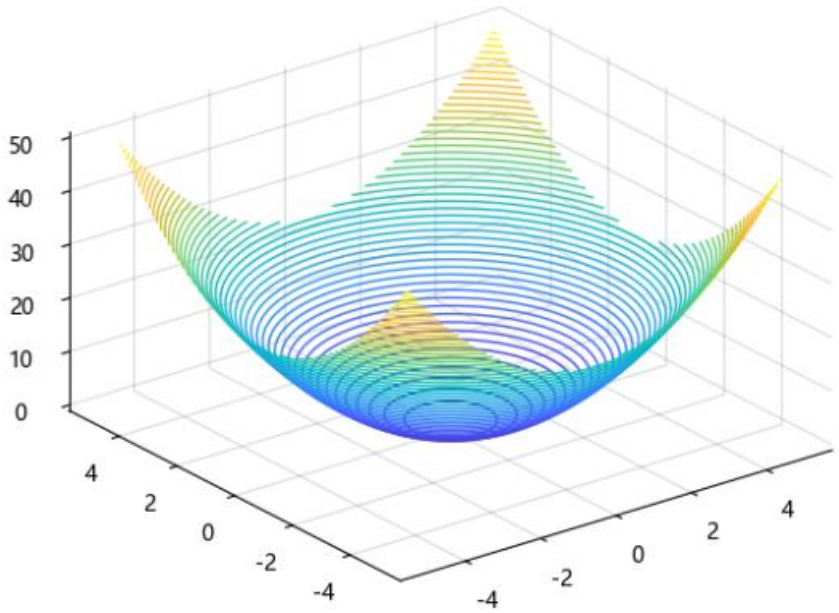
```
using TyPlot
using TyMath
X, Y = meshgrid2(-5:0.25:5, -5:0.25:5);
```

```
Z = X .^ 2 + Y .^ 2;
contour3(Z)
```



现在指定 50 等高线层级，并在用于计算 Z 的 x 和 y 范围内显示结果。

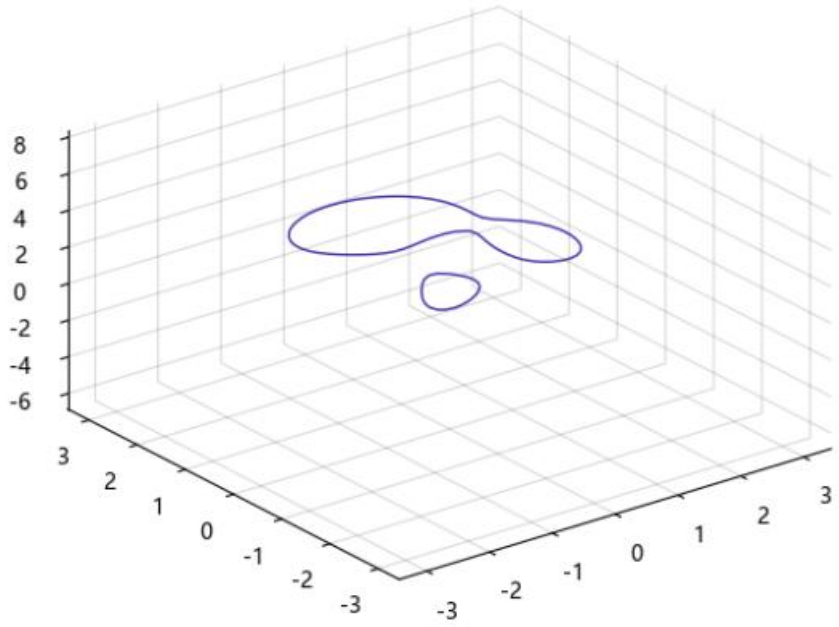
```
contour3(X,Y,Z,50)
```



(3). 一个层级的等高线

将 Z 定义为 X 和 Y 的函数。在本例中，调用 peaks 函数以创建 X、Y 和 Z。然后在 Z = 2 处显示等高线。

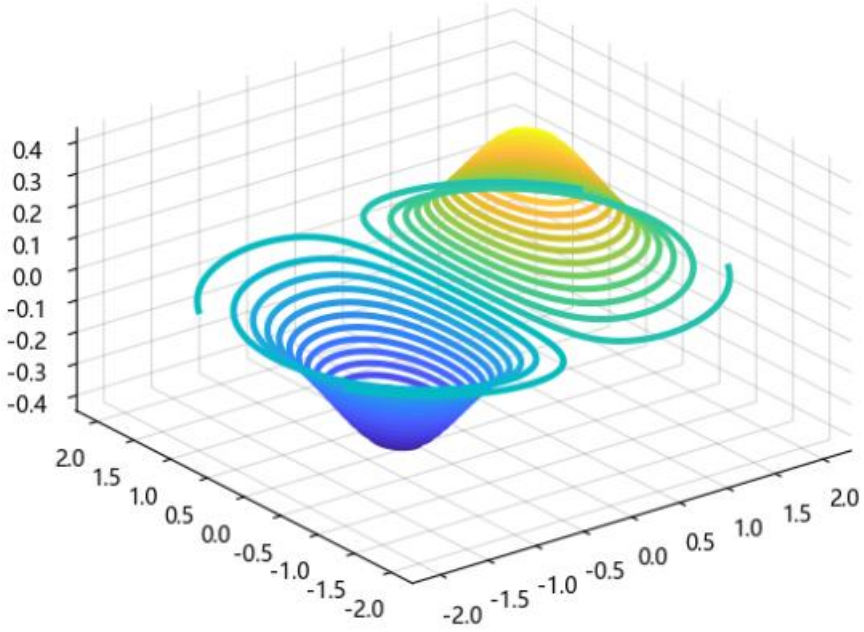
```
X,Y,Z = peaks()
contour3(X,Y,Z,[2])
```



(4). 自定义线宽

将 Z 定义为两个变量 X 和 Y 的函数。绘制 Z 的 30 个等高线，然后将线宽设置为 3。

```
using TyPlot
using TyMath
X, Y = meshgrid2(-2:0.0125:2, -2:0.0125:2);
Z = X .* exp.(-X .^ 2 - Y .^ 2);
c = contour3(X, Y, Z, 30);
for collection in c.collections
   collection.set_linewidth(3)
end
```



3.3.1.5.4 clabel

为等高线图添加高程标签

A. 语法

```
clabel(C)

clabel(___,Key=Value)
```

B. 说明

clabel(C)为当前等高线图添加标签，将旋转文本插入每条等高线。等高线必须足够长以容纳标签，否则 clabel 无法插入标签。

clabel(___,Key=Value)使用一个或多个 Key=Value 对组参数修改标签外观。使用先前语法中的任意输入参数组合。例如，fontsize=14 将字体大小设置为 14 磅。

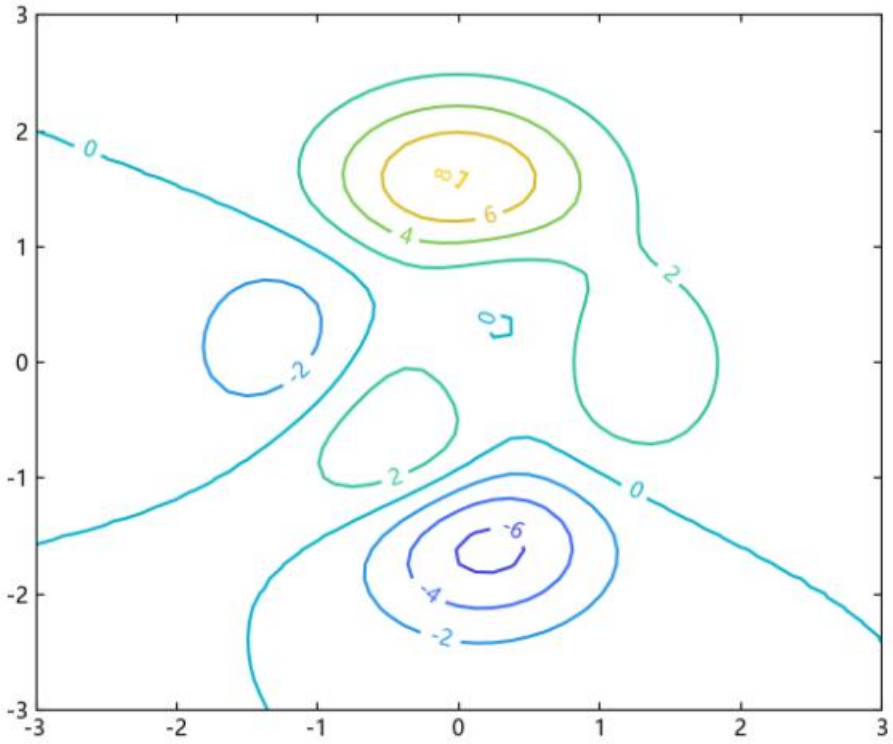
C. 示例

(1). 为等高线图添加层级标签

创建一个等高线图并获取等高线对象 c 。然后，为等高线图添加标签。

```
x,y,z = peaks();
```

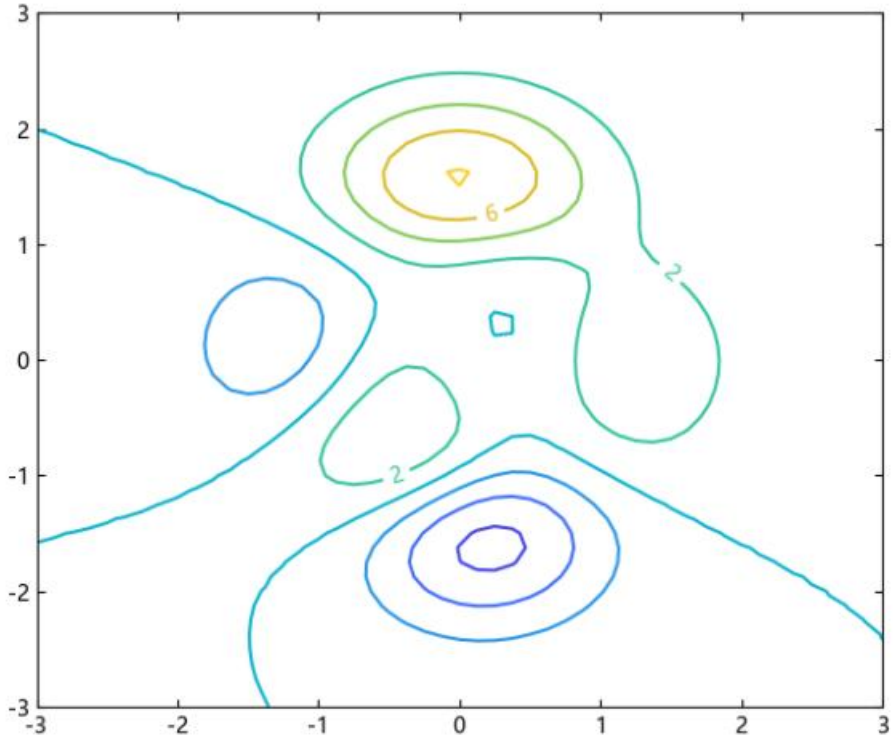
```
c = contour(x,y,z, cmap="parula");
clabel(c)
```



(2). 为特定的等高线层级添加标签

仅为等高线层级为 2 或 6 的等高线添加标签。

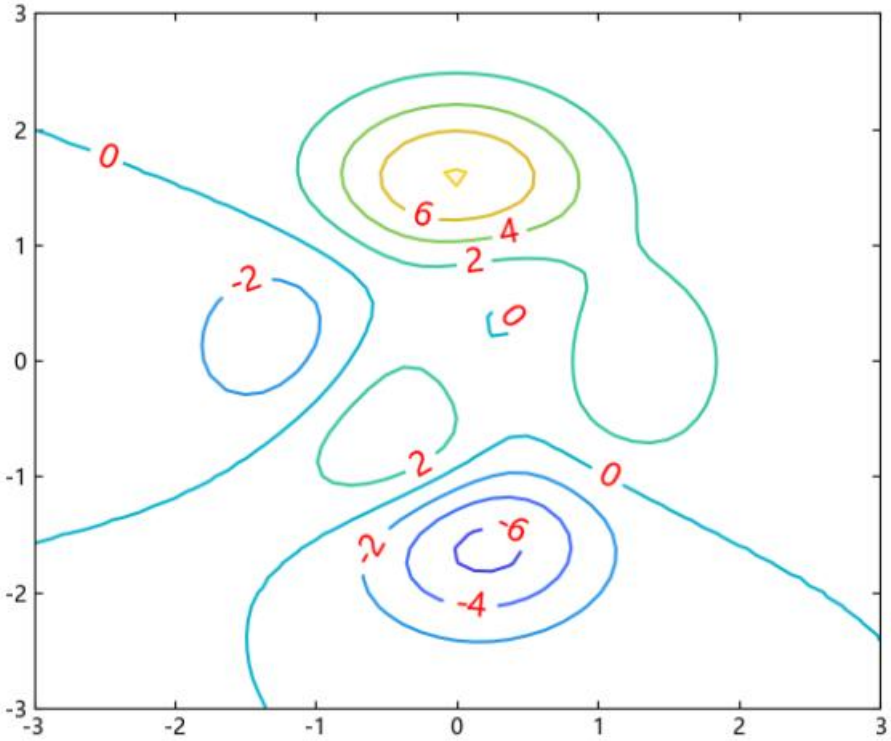
```
x,y,z = peaks();
c = contour(x,y,z);
v = [2,6];
clabel(c,v)
```



(3). 设置等高线标签属性

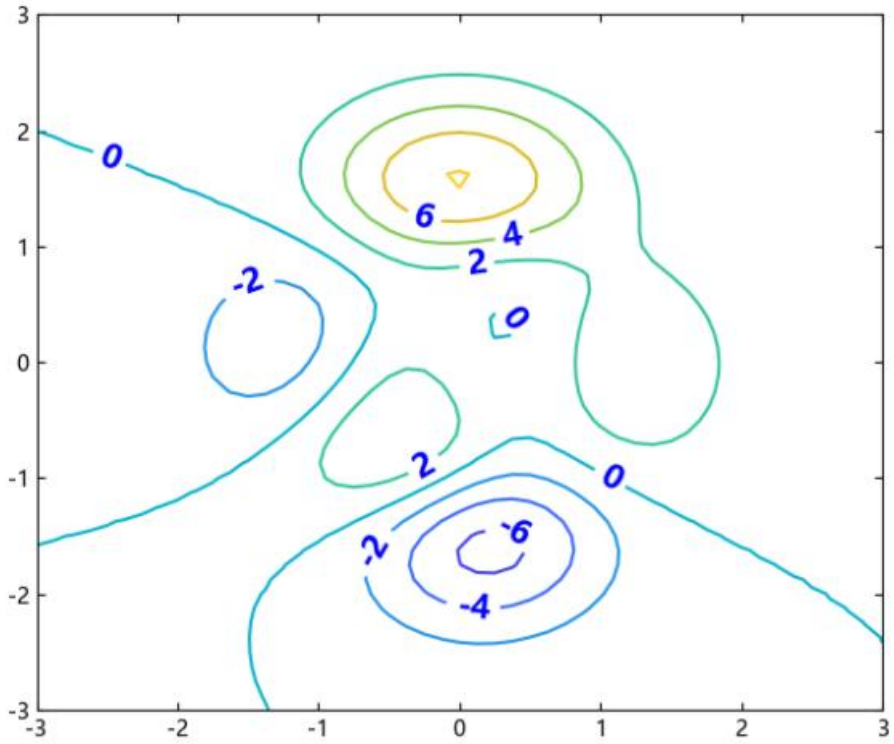
使用 Key=Value 对组参数，将标签的字体大小设置为 15 磅并将颜色设置为红色。

```
x,y,z = peaks();
c = contour(x,y,z);
texts = clabel(c, fontsize=15, colors="red")
```



通过重新发出 clabel 命令设置其他属性。例如，将字体粗细设置为粗体并将颜色更改为蓝色。

```
for i in 1:length(texts)
    texts[i].set_color("blue")
    texts[i].set_fontweight("bold")
end
```



3.3.1.5.5 fcontour

绘制等高线

A. 语法

```
fcontour(f)
fcontour(f,xyinterval)
fcontour(___,Key=Value)
fcontour(ax,___)
fc = fcontour(___)
```

B. 说明

fcontour(f) 根据 x 和 y 的默认区间 [-5 5] 和 z 的固定级别值绘制 z = f(x,y)函数的等高线。

fcontour(f,xyinterval)将在指定区间绘图。要对 x 和 y 使用相同的区间，请将 xyinterval 指定为 [min max] 形式的二元素向量。要使用不同的区间，请指定 [xmin xmax ymin ymax] 形式的四元素向量。

fcontour(___,Key=Value)使用一个或多个名称-值对组参数指定等高线图的其他选项。请在所有其他输入参数之后指定这些选项。

fcontour(ax,___) 将图形绘制到 ax 指定的坐标区中，而不是当前坐标区中。

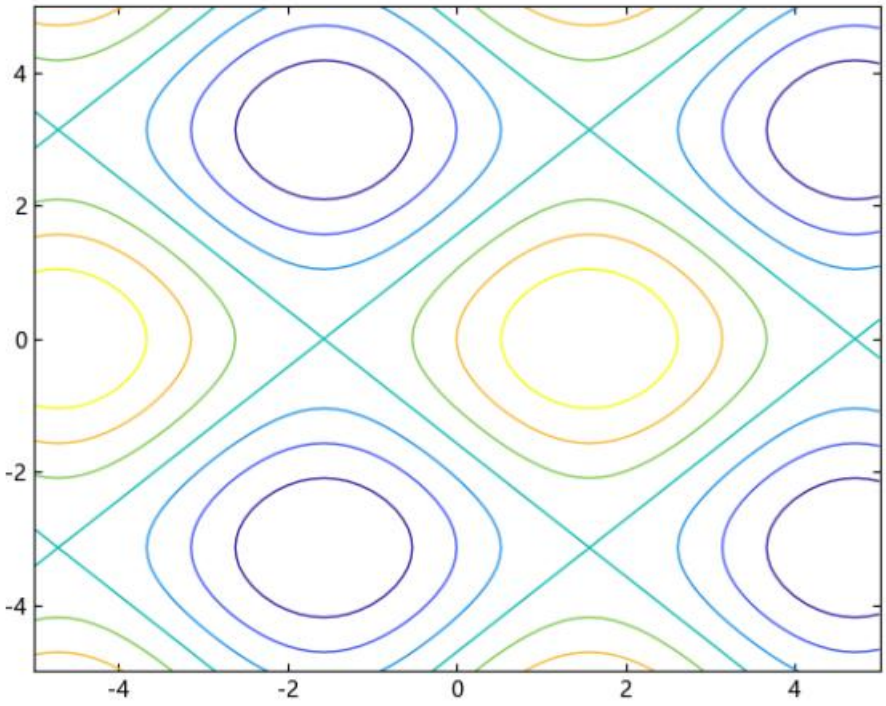
fc=fcontour(___)返回 fcontour 对象。可使用 fc 查询和修改特定 fcontour 对象的属性。

C. 示例

(1). 绘制函数的等高线

在 $-5 < x < 5$ 和 $-5 < y < 5$ 的默认区间绘制 的等高线。

```
f = (x,y)->sin(x)+cos(y)
fcontour(f)
```



(2). 指定绘图区间并绘制分段等高线图

指定绘图区间作为 fcontour 的第二个参数。在相同坐标区的不同区间绘制多个输入时，坐标轴范围会调整以显示所有数据。这种行为可以让您绘制分段输入。

$$erf(x) + \cos(y) \quad -5 < x < 0$$

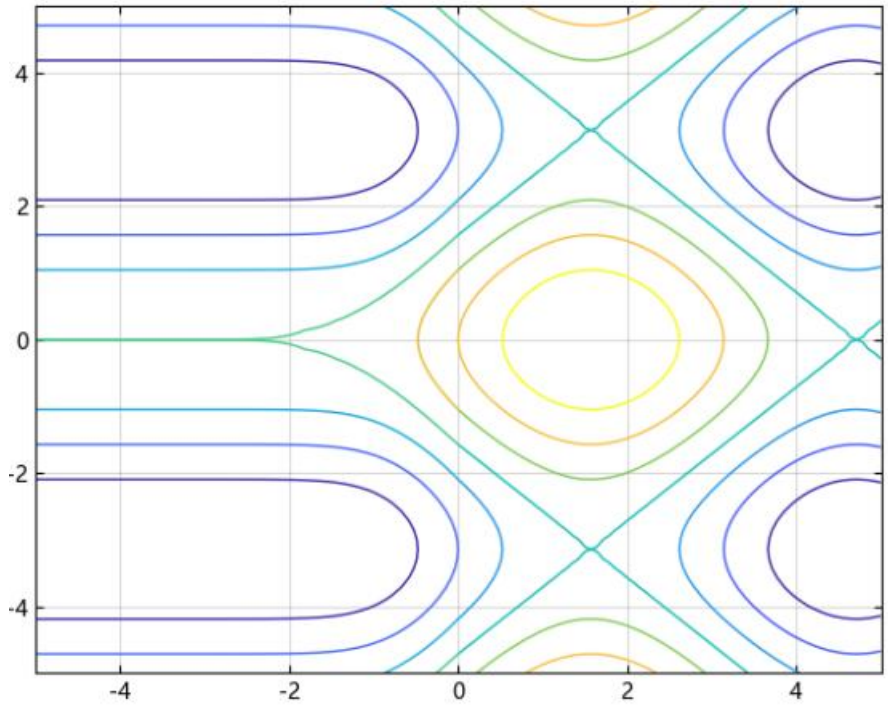
绘制分段输入 $\sin(x) + \cos(y) \quad 0 < x < 5$

区间为 $-5 < y < 5$。

```
using SpecialFunctions

fcontour((x,y)->erf(x) + cos(y),[-5 0 -5 5])
hold("on")
fcontour((x,y)->sin(x) + cos(y),[0 5 -5 5])
hold("off")
```

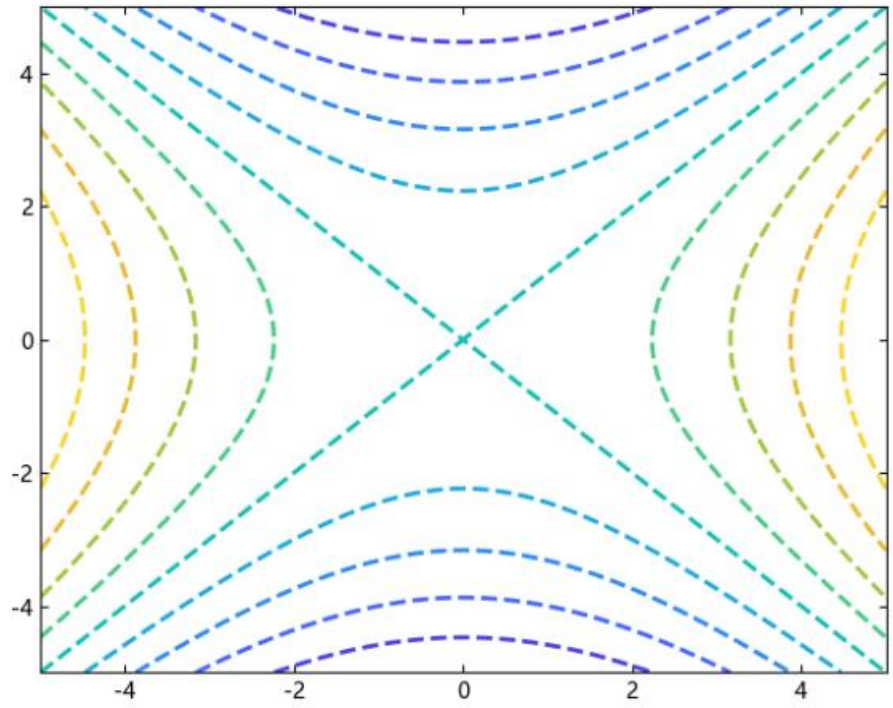
```
grid("on")
```



(3). 更改线型和线宽

以线宽为 2 的虚线绘制 $x^2 - y^2$ 的等高线。

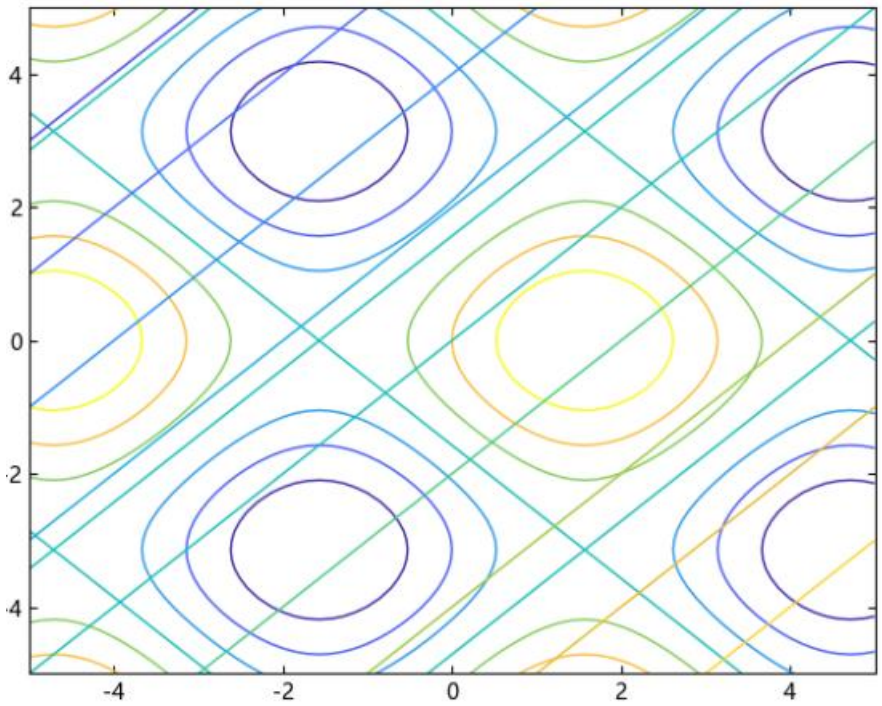
```
f = (x,y)-> x.^2 - y.^2;
fcontour(f, linestyle="--", linewidth=2)
```



(4). 绘制多个等高线图

使用 hold("on") 在相同的坐标区上绘制 $\sin(x)+\cos(y)$ 和 $x-y$。

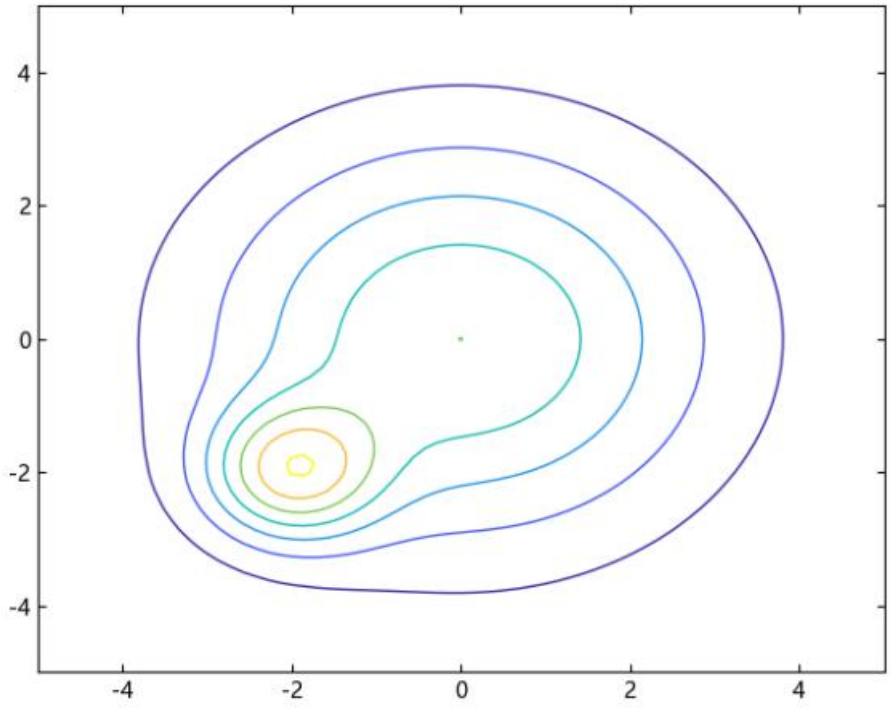
```
fcontour((x,y)->sin(x)+cos(y))
hold("on")
fcontour((x,y)->x-y)
hold("off")
```



(5). 创建后修改等高线图

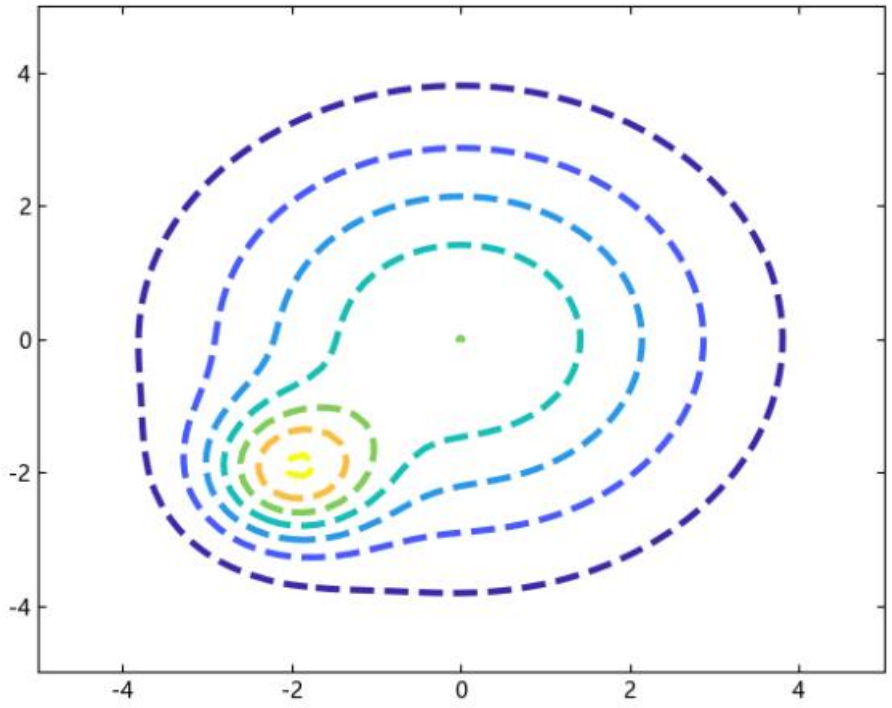
绘制的等高线$e^{-(x/3)^2-(y/3)^2}+e^{-(x+2)^2-(y+2)^2}$。将函数等高线对象指定给变量。

```
f = (x,y)-> exp(-(x/3).^2-(y/3).^2) + exp(-(x+2).^2-(y+2).^2);
fc = fcontour(f)
```



通过使用圆点表示法设置函数等高线对象的属性，可以将线宽更改为 1，将线型更改为虚线。通过设置 levellist 属性，可以显示靠近 0 和 1 的等高线。添加颜色栏。

```
for collection in fc.collections
    collection.set_linewidth(3)
    collection.set_linestyle("--")
end
```

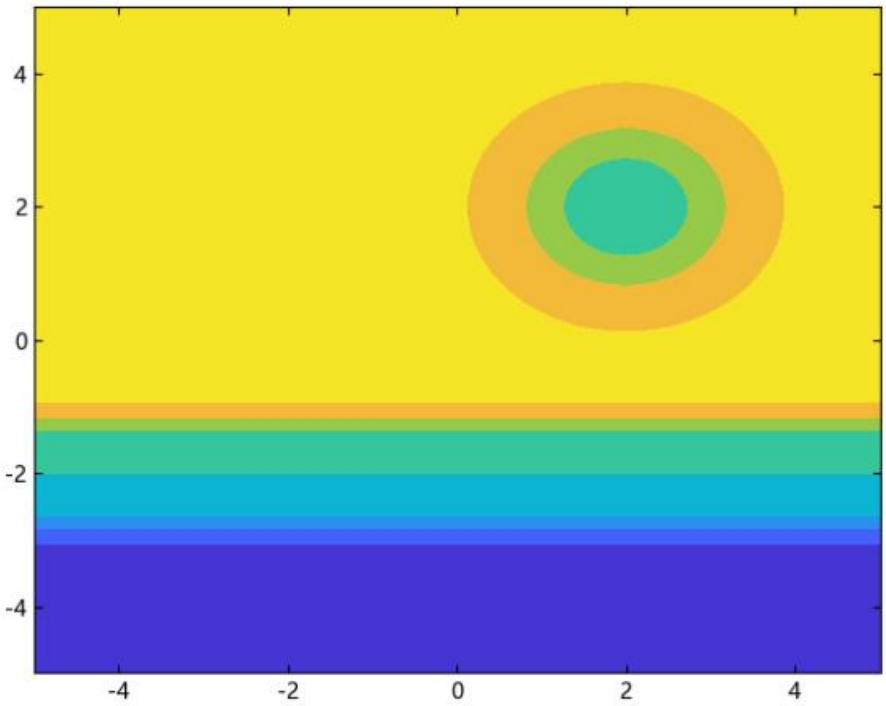


(6). 填充等高线之间的区域

通过填充下列方程的等高线之间的区域，创建看起来像日落的绘图：

$$erf\left((y+2)^3\right) - e^{\left(-0.65((x-2)^2+(y-2)^2)\right)}$$

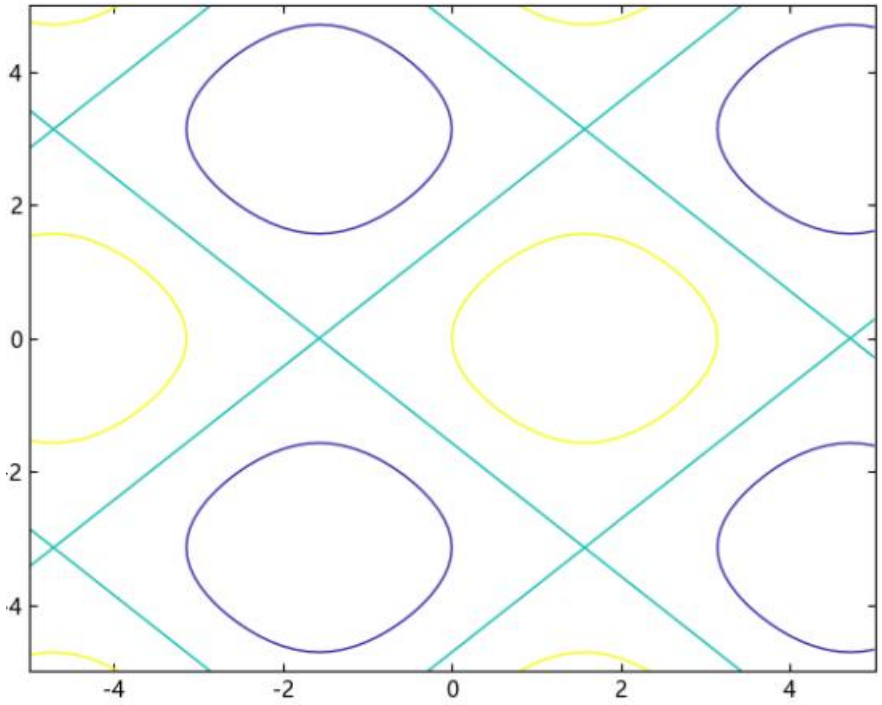
```
f = (x,y)-> erf((y+2).^3) - exp(-0.65*((x-2).^2+(y-2).^2));
fcontour(f,filled=true);
```



(7). 指定等高线的级别

使用 levellist 选项设置 fcontour 绘制等高线时所用的级别。

```
f = (x,y)-> sin(x) + cos(y);
fcontour(f,levellist=[-1, 0, 1])
```



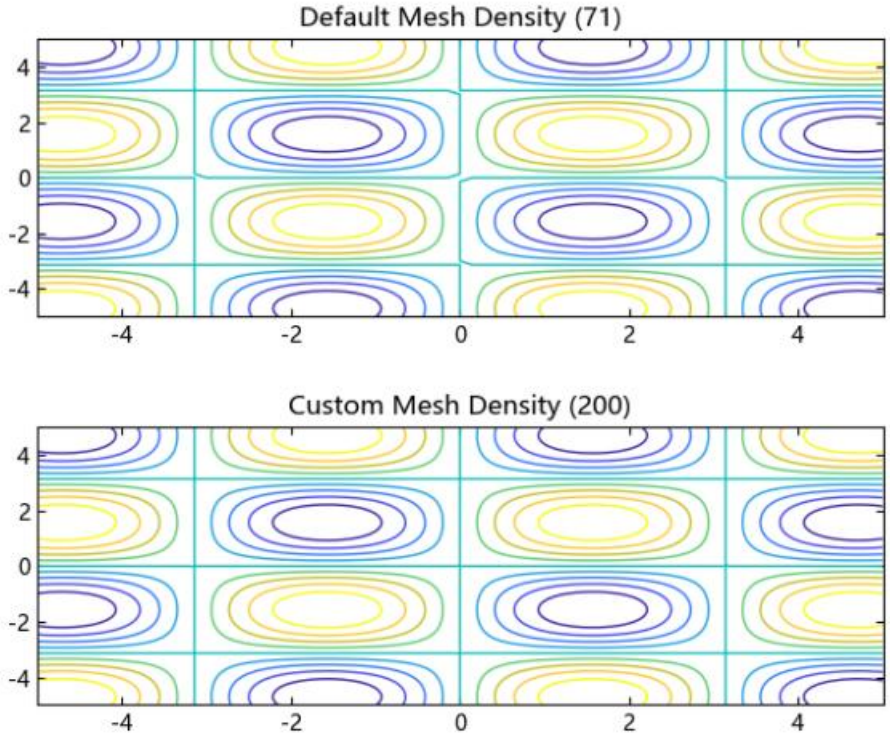
(8). 控制等高线的分辨率

使用 meshdensity 选项控制等高线的分辨率。增大 meshdensity 可以使绘图更圆滑、更准确，减小该值可以提高绘图速度。

在 2×1 分块图布局中创建两个绘图。在第一个绘图中，显示 的等高线。方形的角不重合。要解决此问题，请将第二个绘图中的 meshdensity 增大到 200。现在的角彼此重合，这说明增大 meshdensity 可以提高分辨率。

```
f = (x,y)-> sin(x).*sin(y);
ax1 = subplot(2,1,1)
fcontour(f)
title("Default Mesh Density (71)")

ax2 = subplot(2,1,2)
fcontour(ax2,f, meshdensity=200)
title("Custom Mesh Density (200)")
```

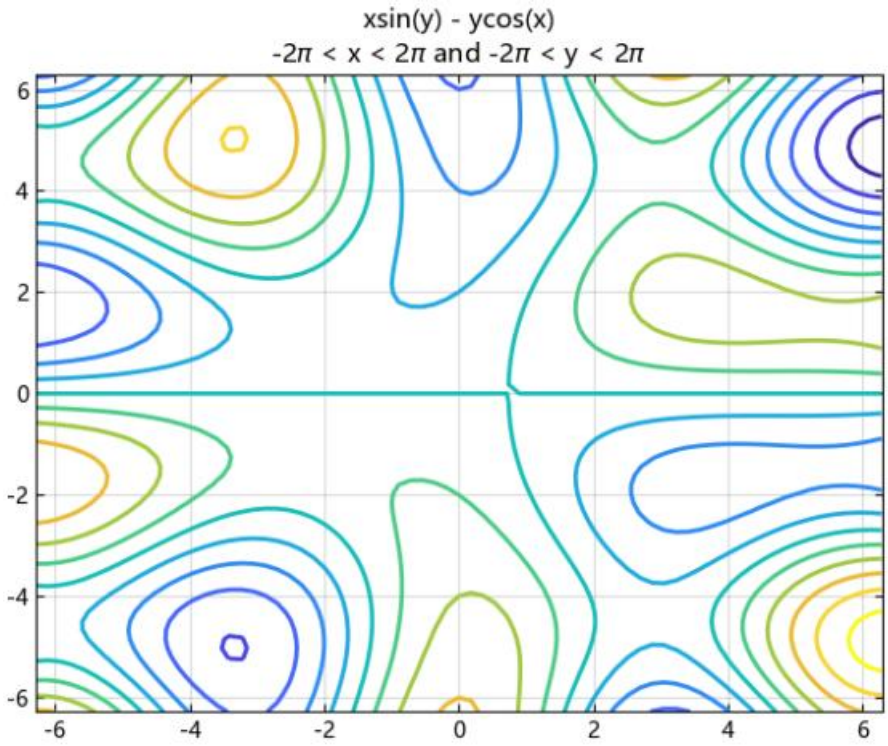


(9). 添加标题和轴标签以及格式化刻度

绘制$x\sin(y) - y\cos(x)$ 。显示网格线、添加标题、添加轴标签。

```
fcontour((x,y)->x.*sin(y) - y.*cos(x), [-2*pi 2*pi], linewidth=2)
grid("on")
title("xsin(y) - ycos(x)\n"*raw"-2$\pi$ < x < 2$\pi$ and -2$\pi$ <
y < 2$\pi$")

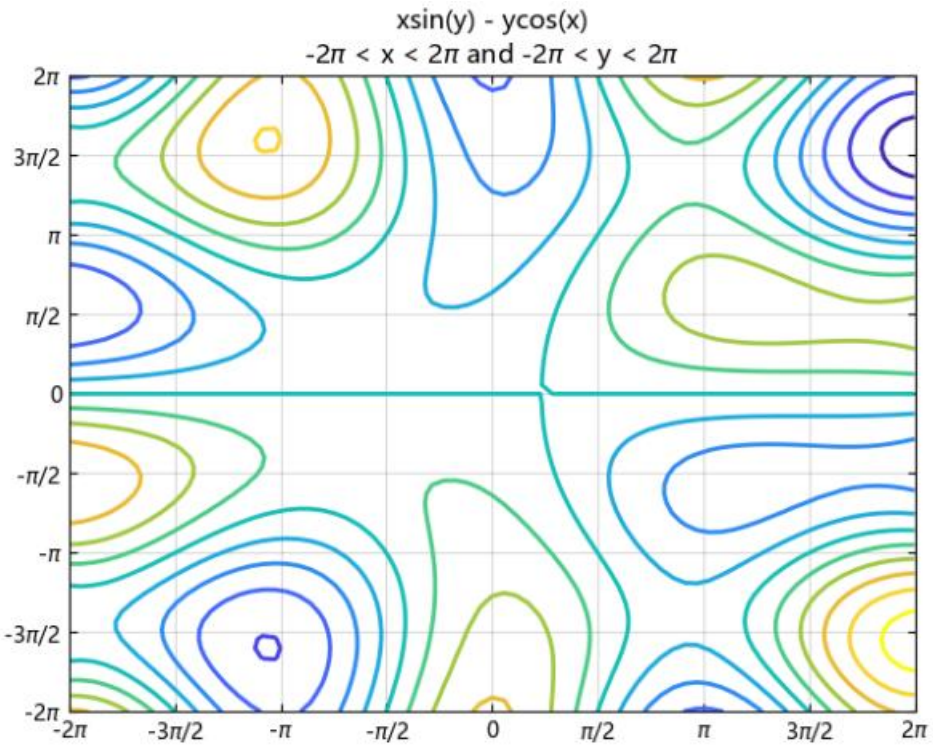
xlabel("x")
ylabel("y")
```



通过设置坐标区对象的 xticklabels 和 xticks 属性设置 x 轴刻度值和关联的标签。使用 gca 访问坐标区对象。按照同样的方式设置 y 轴刻度值和关联的标签。

```
ax = gca();
xticks(xlim()[1]:pi/2:xlim()[2])
xticklabels(ax,  [raw"-2$\pi$",  raw"-3$\pi$/2",  raw"-$\pi$",raw"-
$\pi$/2","0",raw"$\pi$/2",raw"$\pi$",raw"3$\pi$/2",raw"2$\pi$"]);

yticks(ylim()[1]:pi/2:ylim()[2])
yticklabels(ax,   [raw"-2$\pi$",   raw"-3$\pi$/2",raw"-$\pi$",raw"-
$\pi$/2","0",raw"$\pi$/2",raw"$\pi$",raw"3$\pi$/2",raw"2$\pi$"]);
```



3.3.1.6 曲面、体积和多边形

网格曲面和三维体数据、非网格多边形数据

函数名	简介
surf	曲面图
surfc	曲面图下的等高线图
meshc	网格曲面图下的等高线图
surface	基本曲面图
mesh	网格曲面图
fsurf	绘制三维曲面
fmesh	绘制三维网格图
peaks	包含两个变量的示例函数
cylinder	创建圆柱

ellipsoid	创建椭圆体
sphere	创建球面
pcolor	伪彩图
ribbon	条带图
flow	包含三个变量的简单函数
plt_fill	填充的二维多边形
patch	绘制一个或多个填充多边形区域

3.3.1.6.1 surf

曲面图

A. 语法

```
surf(X,Y,Z)

surf(Z)

surf(ax,___)
surf(___,Key=Value)
s = surf(___)
```

B. 说明

surf(X,Y,Z)创建一个三维曲面图，它是一个具有实色边和实色面的三维曲面。该函数将矩阵 Z 中的值绘制为由 X 和 Y 定义的 x-y 平面中的网格上方的高度。曲面的颜色根据 Z 指定的高度而变化。

surf(Z) 创建一个曲面图，并将 Z 中元素的列索引和行索引用作 x 坐标和 y 坐标。

surf(ax,___) 将图形绘制到 ax 指定的坐标区中，而不是当前坐标区中。指定坐标区作为第一个输入参数。

surf(___,Key=Value)使用一个或多个名称-值对组参数指定曲面属性。例如，alpha=0.5 创建半透明曲面。

s = surf(___) 将返回一个图曲面对象。在创建曲面之后可使用 s 对其进行

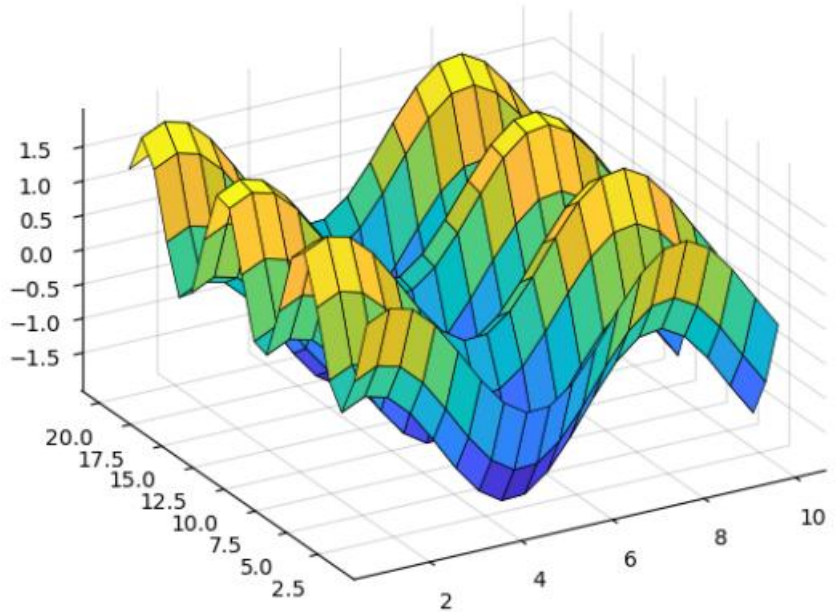
修改。有关属性列表，请参阅 Surface 属性。

C. 示例

(1). 创建曲面图

创建三个相同大小的矩阵。然后将它们绘制为一个曲面。曲面图对高度和颜色均使用 Z。

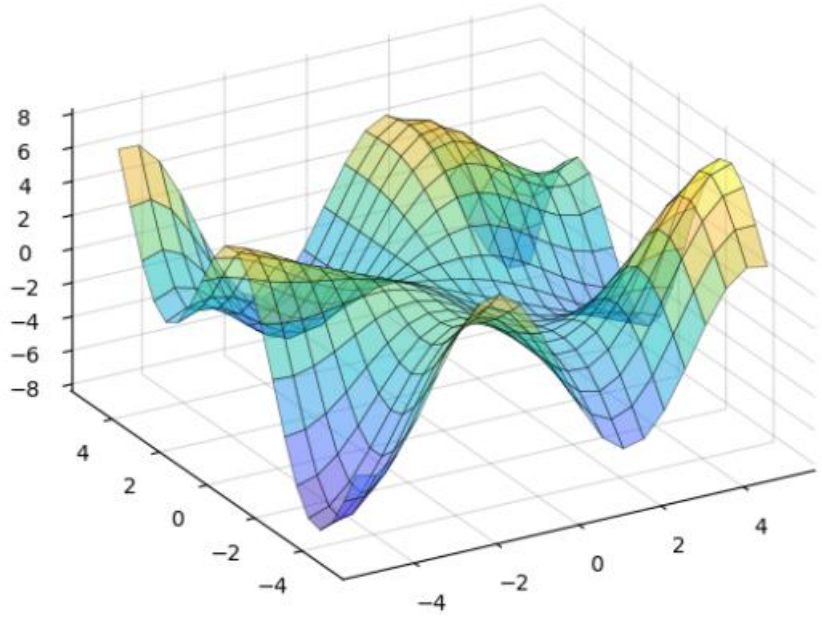
```
X, Y = meshgrid2(1:0.5:10, 1:20)
Z = sin.(X) + cos.(Y)
fig = figure()
ax = subplot(; projection="3d")
surf(ax, X, Y, Z)
```



(2). 修改曲面图的外观

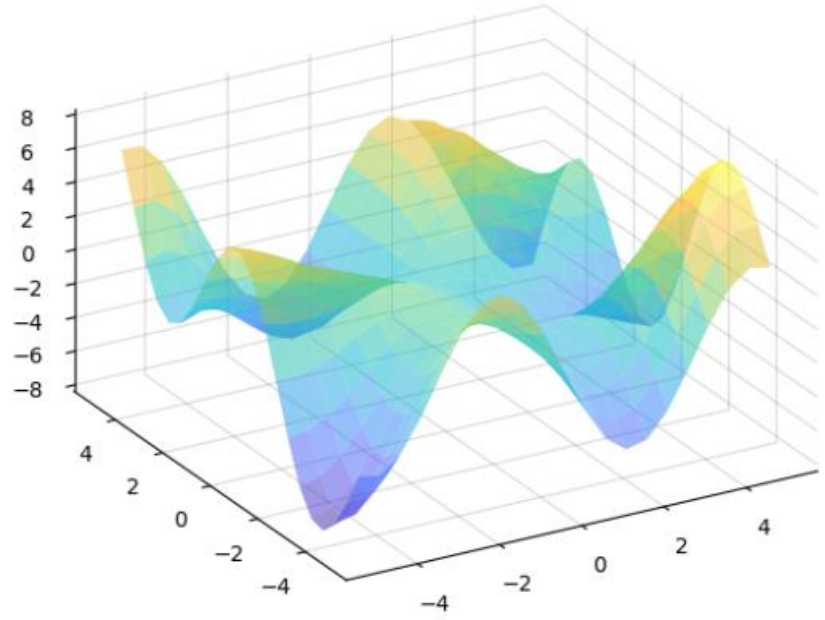
通过指定以 0.5 为值的 alpha 名称-值对组，来创建半透明曲面。要允许进一步修改，请将曲面对象赋给变量 s。

```
X, Y = meshgrid2(-5:0.5:5, -5:0.5:5)
Z = Y .* sin.(X) - X .* cos.(Y)
s = surf(X, Y, Z; alpha=0.5)
```



在创建曲面对象之后可使用 s 访问并修改其属性。例如，通过设置 edgecolor 属性来隐藏边。

```
s.set_edgecolor("none")
```



3.3.1.6.2 surfc

曲面图下的等高线图

A. 语法

```
surfc(X,Y,Z)
surfc(Z)
surfc(ax,___)
surfc(___,Key=Value)
sc = surfc(___)
```

B. 说明

surfc(X,Y,Z) 创建一个三维曲面图，其下方有等高线图。曲面图是一个具有实色边和实色面的三维曲面。该函数将矩阵 Z 中的值绘制为由 X 和 Y 定义的 x-y 平面中的网格上方的高度。曲面的颜色根据 Z 指定的高度而变化。示例

surfc(Z) 创建一个曲面和等高线图，并将 Z 中元素的列索引和行索引分别用作 x 坐标和 y 坐标。

surfc(ax,___) 将图形绘制到 ax 指定的坐标区中，而不是当前坐标区中。指定坐标区作为第一个输入参数。

surfc(___,Key=Value) 使用一个或多个名称-值对组参数指定曲面属性。例如，"facealpha"=0.5 创建半透明曲面。示例

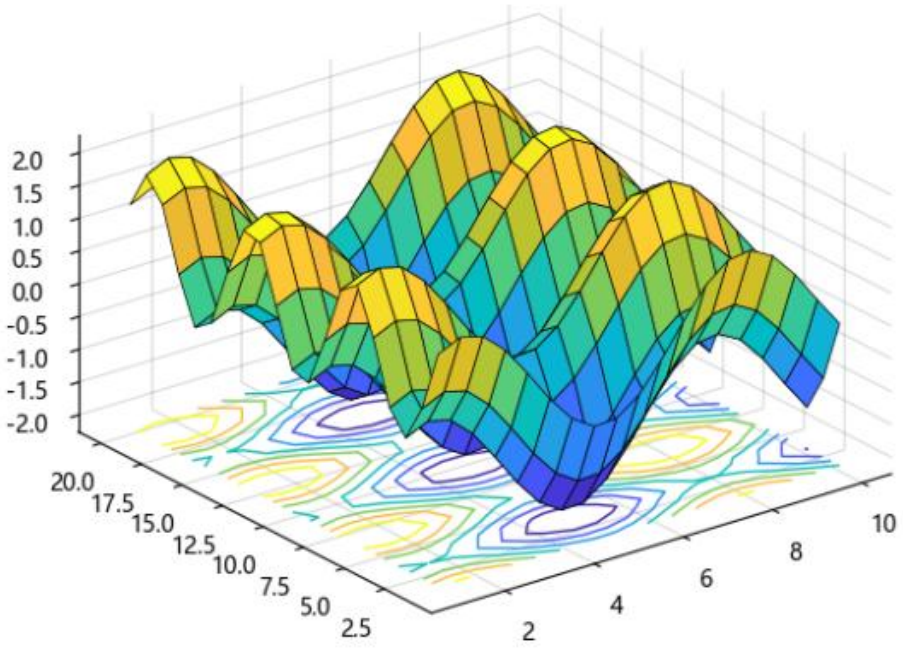
sc = surfc(___) 返回包含图曲面对象和等高线对象的图形数组。使用 sc 修改所创建的曲面图和等高线图。示例。

C. 示例

(1). 显示曲面图下的等高线图

创建三个相同大小的矩阵。然后将它们绘制为曲面，并在曲面图下显示等高线图。曲面的高度和颜色均使用 Z 确定。

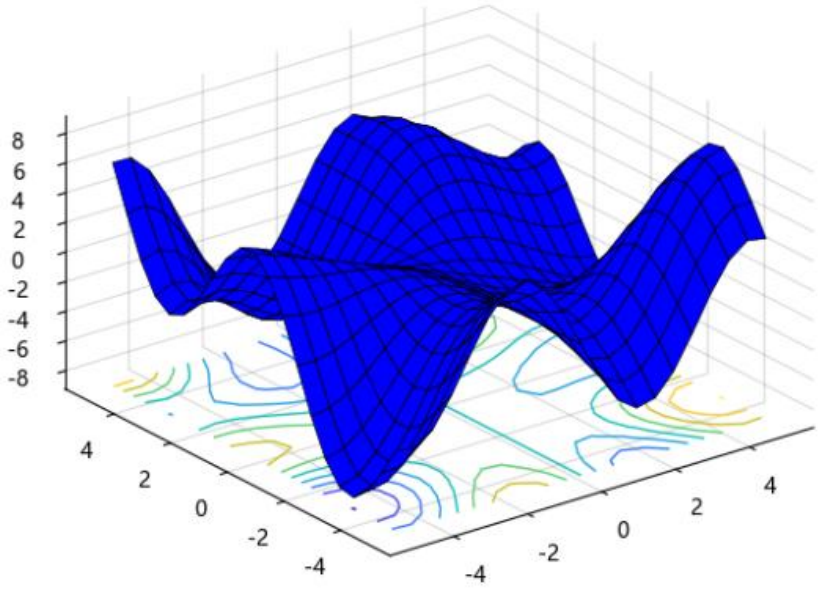
```
X, Y = meshgrid2(1:0.5:10, 1:20)
Z = sin.(X) + cos.(Y)
s = surfc(X, Y, Z)
```



(2). 修改曲面图和等高线图的外观

通过将 facecolor 名称-值对组的值指定为 "b"，创建一个蓝色曲面图，其下方有等高线图。要允许进一步修改，请将包含曲面和等高线对象的图形数组赋给变量 s。

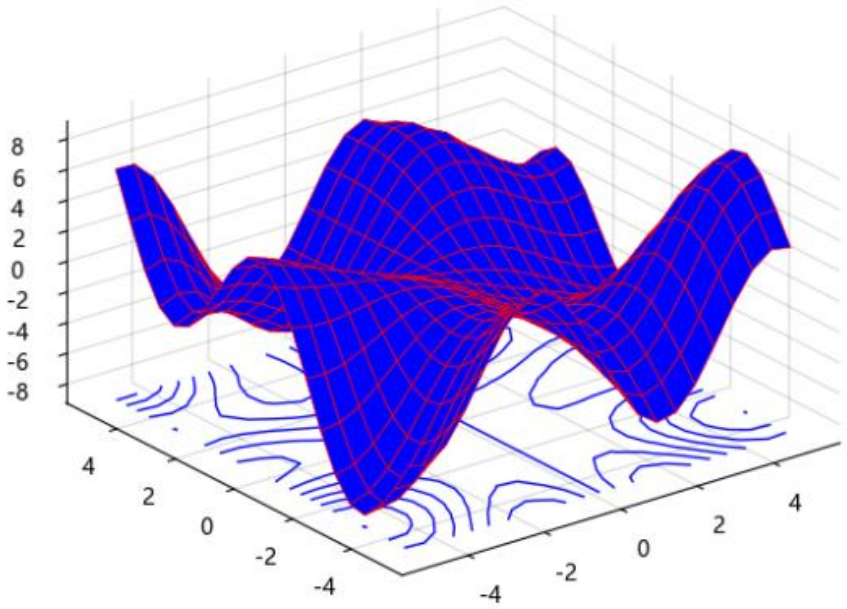
```
X, Y = meshgrid2(-5:0.5:5)
Z = Y .* sin.(X) - X .* cos.(Y)
s = surfc(X, Y, Z; facecolor="b")
```



对 s 进行索引，以访问和修改所创建的曲面和等高线图的属性。可通过

s[1] 访问曲面图，通过 s[2] 访问等高线图。例如，通过设置 edgecolor 属性来更改两个绘图的边颜色。

```
s[1].set_edgecolor("r")
for collection in s[2].collections
    collection.set_edgecolor("b")
end
plt_update()
```

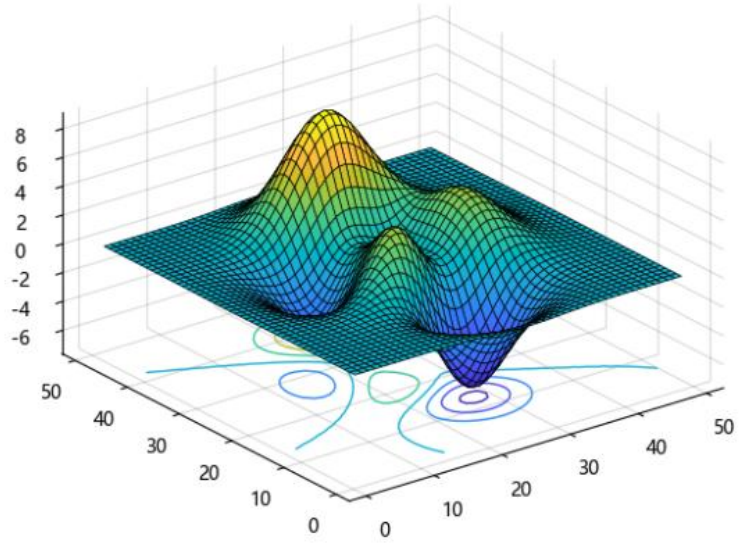


(3). 更改等高线的位置

默认情况下，等高线显示在最小 z 层级，但您可以通过设置 zlim 属性来更改其位置。

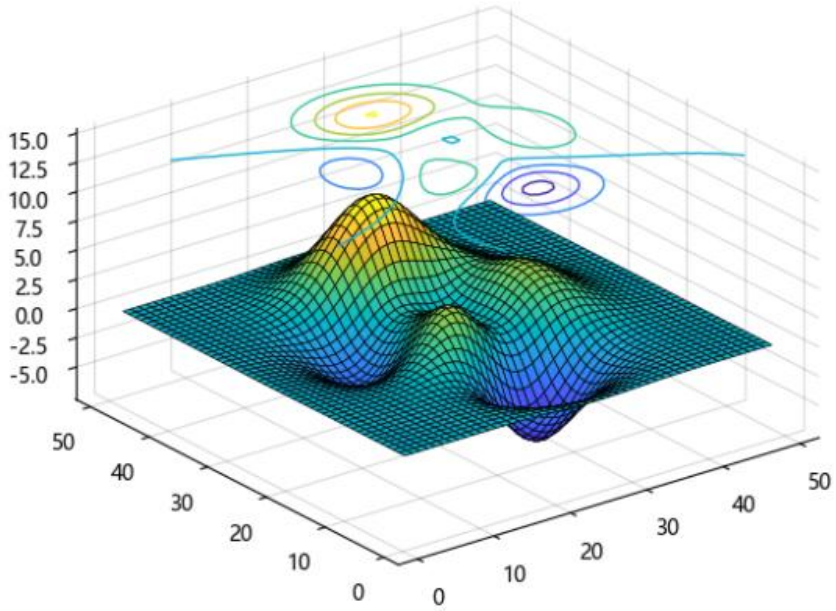
将 peaks 数据集显示为曲面图，等高线位于最小 z 层级。当您调用 surfc 函数时，请指定返回参数，以便您 contour 对象。

```
x, y, z = peaks()
sc = surfc(z);
```



获取当前坐标区，并将 z 轴的上限扩展到 15。然后将等高线移至最大 z 层级。

```
s = surfc(z; offset=15)
ax = gca()
ax.set_zlim(; zmax=15)
```



3.3.1.6.3 meshc

网格曲面图下的等高线图

A. 语法

```
meshc(X,Y,Z)
meshc(Z)
meshc(___,Key=Value)
meshc(ax,___)
sc = meshc(___)
```

B. 说明

meshc(X,Y,Z) 创建一个网格图，其下方有等高线图。网格图是一个三维曲面，有实色边颜色，无面颜色。该函数将矩阵 Z 中的值绘制为由 X 和 Y 定义的 x-y 平面中的网格上方的高度。边颜色因 Z 指定的高度而异。示例

meshc(Z) 创建一个网格和等高线图，并将 Z 中元素的列索引和行索引用作 x 坐标和 y 坐标。

meshc(___,Key=Value) 使用一个或多个名称-值对组参数指定 meshc 图的其他选项。请在所有其他输入参数之后指定这些选项。

meshc(ax,___) 将图形绘制到 ax 指定的坐标区中，而不是当前坐标区中。指定坐标区作为第一个输入参数。

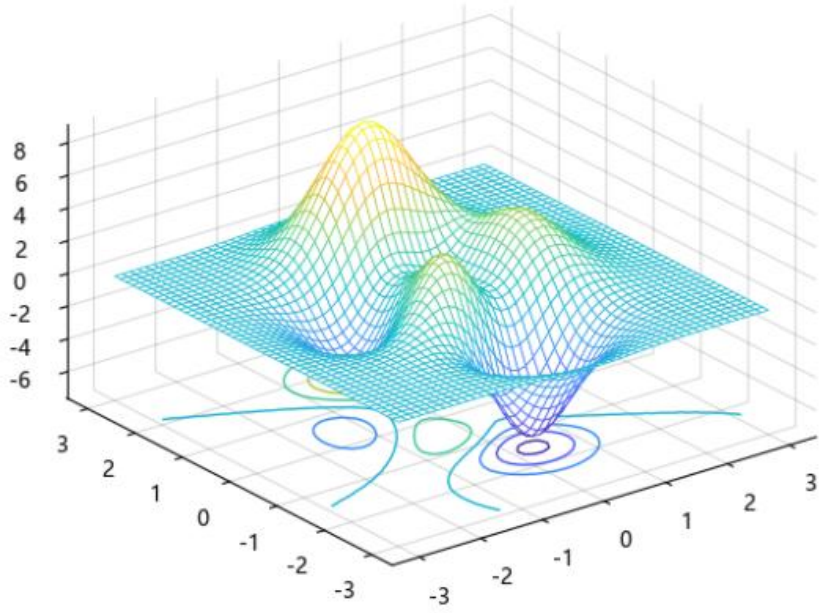
sc = meshc(___) 返回包含图曲面对象和等高线对象的图形数组。创建网格图和等高线图后，使用 sc 对其进行修改。示例

C. 示例

(1). 在网格图下显示等高线图

创建三个相同大小的矩阵。然后将它们绘制为一个网格图，其下方有一个等高线图。网格图使用 Z 确定高度和颜色。

```
X, Y = meshgrid2(-3:0.125:3)
Z = peaks(X, Y)
meshc(X, Y, Z)
```

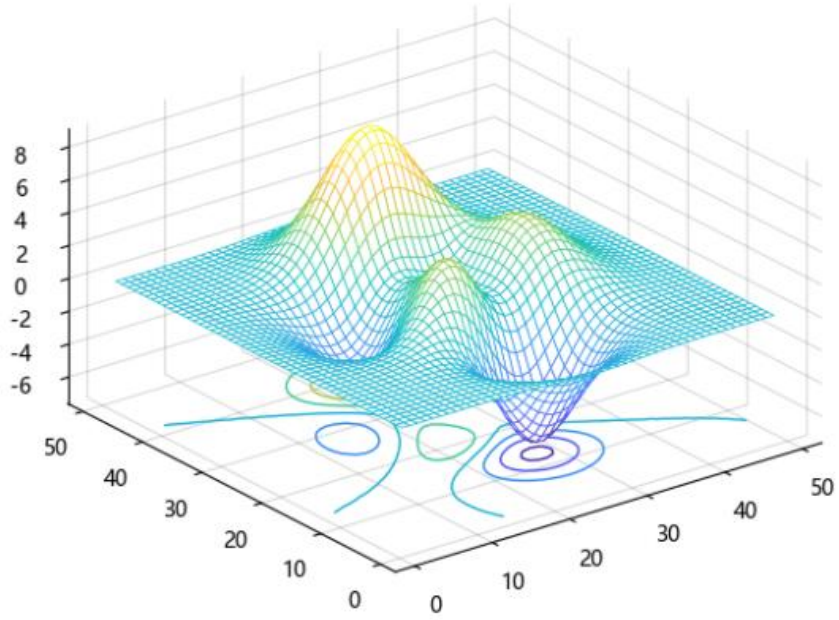


(2). 修改网格和等高线图的外观

默认情况下，等高线显示在最小 z 层级，但您可以通过设置 zlim 属性来更改其位置。

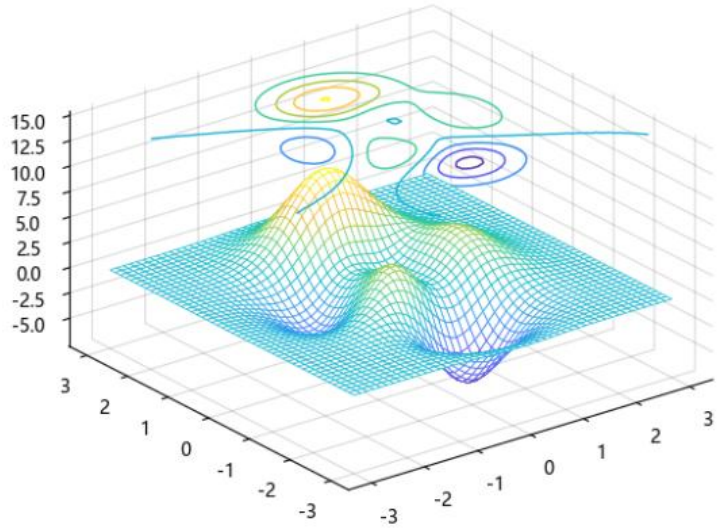
将 peaks 数据集显示为网格图，等高线位于最小 z 层级。当您调用 meshc 函数时，请指定返回参数，以便您 contour 对象。

```
x, y, z = peaks()
sc = meshc(z)
```



获取当前坐标区，并将 z 轴的上限扩展到 15。然后将等高线移至最大 z 层级。

```
sc = meshc(x, y, z; offset=15)
gca().set_zlim(; zmax=15)
```



3.3.1.6.4 surface

基本曲面图

A. 语法

```
surface(X,Y,Z)
surface(Z)
surface(ax,___)
surface(___,Key=Value)
s = surface(___)
```

B. 说明

surface(X,Y,Z) 创建一个基本三维曲面图。该函数将矩阵 Z 中的值绘制为由 X 和 Y 定义的 x-y 平面中的网格上方的高度。曲面的颜色根据 Z 指定的高度而变化。

surface(Z) 创建一个基本曲面图，并将 Z 中元素的列索引和行索引用作 x 坐标和 y 坐标。

surface(ax,___) 将图形绘制到 ax 指定的坐标区中，而不是当前坐标区中。指定坐标区作为第一个输入参数。

surface(___,Key=Value) 使用一个或多个名称-值对组参数指定曲面属性。例如，'FaceAlpha',0.5 创建半透明曲面。

s = surface(___) 返回图基本曲面对象。在创建曲面之后可使用 s 对其进行修改。

mesh(___,Key=Value)使用一个或多个名称-值对组参数指定曲面属性。例如，alpha=0.5 创建半透明网格图。

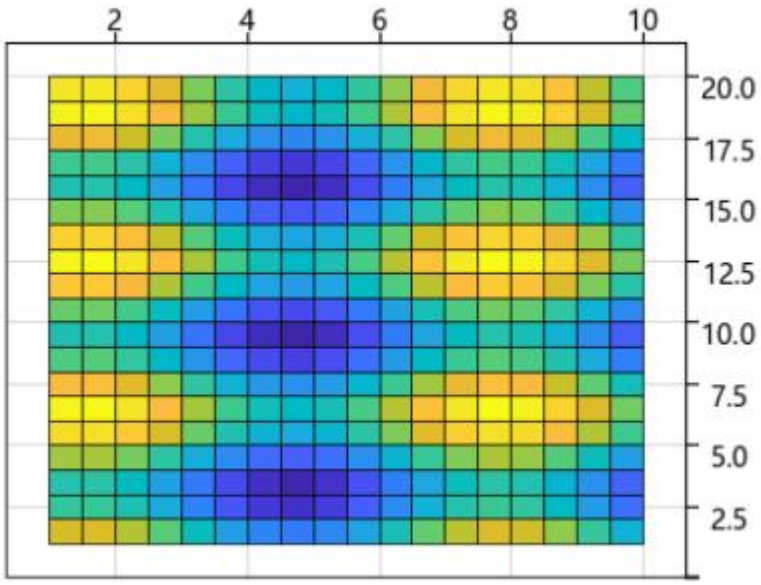
s = mesh(___) 将返回一个图曲面对象。在创建网格图后，使用 s 修改网格图。有关属性列表，请参阅 Surface 属性。

C. 示例

(1). 创建曲面图

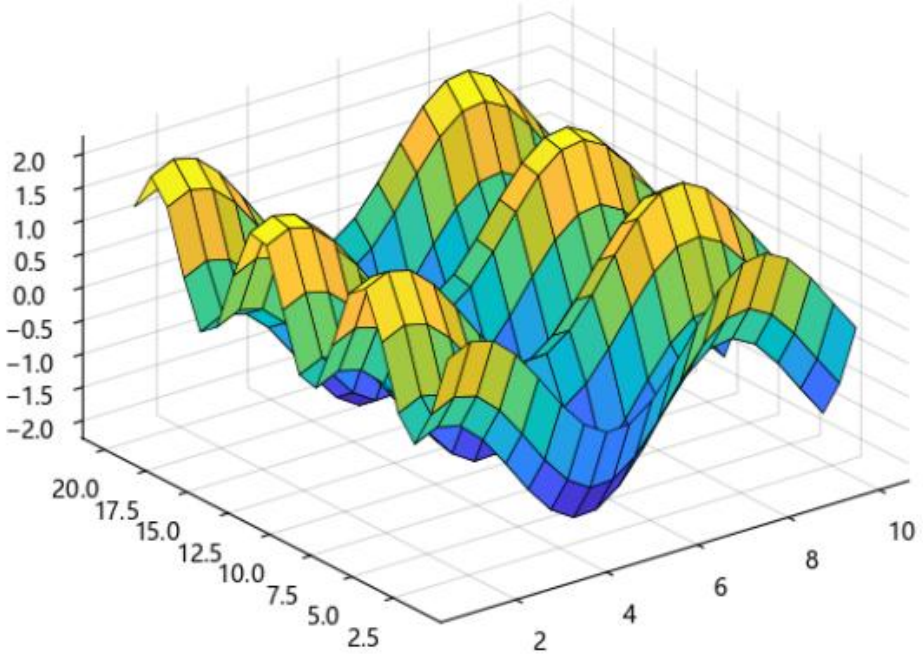
创建三个相同大小的矩阵。然后将它们绘制为一个曲面。曲面对高度和颜色均使用 Z。

```
x = 1:0.5:10
y = 1:20
X = x' .* ones(length(y), 1)
Y = y .* ones(1, length(x))
Z = sin.(X) + cos.(Y)
s = surface(X, Y, Z)
```



默认情况下，在二维视图坐标区中显示曲面。将坐标区更改为三维视图。

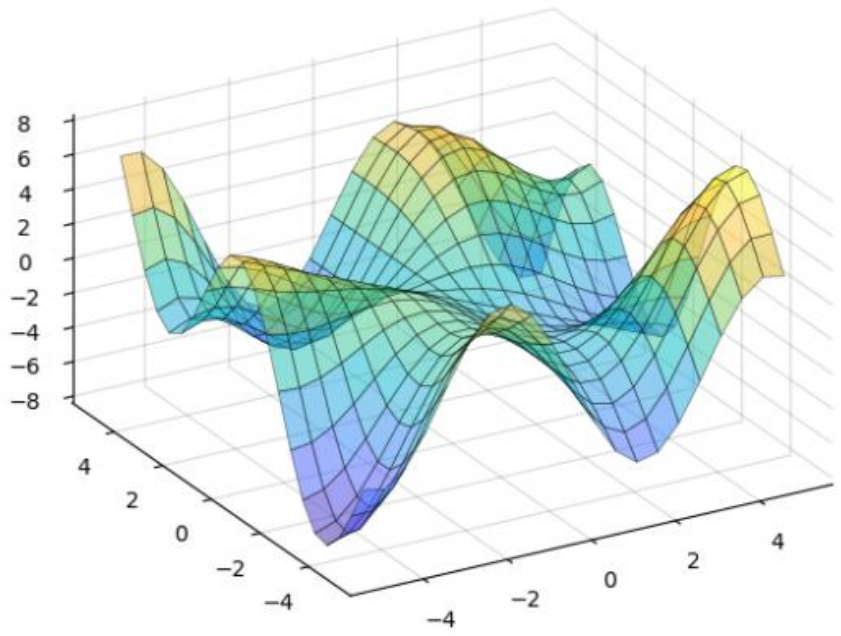
```
plt_view(3)
```



(2). 修改曲面图的外观

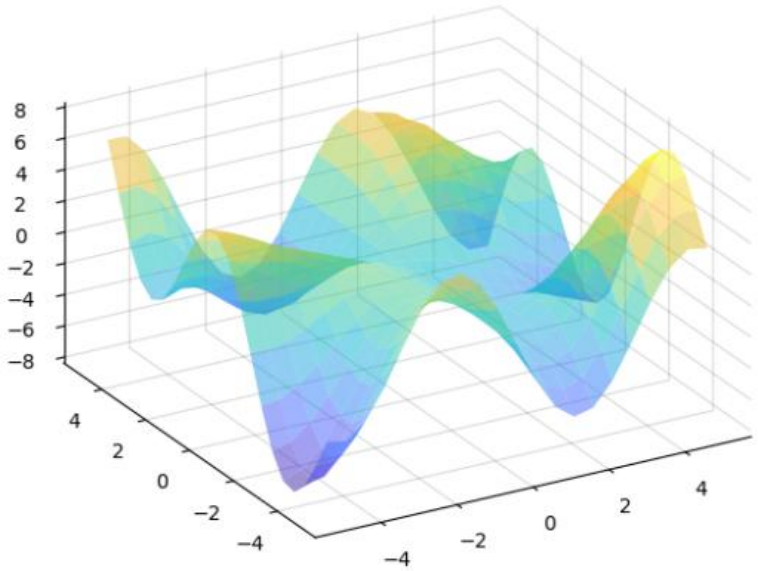
通过指定以 0.5 为值的 alpha 键-值对组，来创建半透明曲面。要允许进一步修改，请将曲面对象赋给变量 s。

```
x = -5:0.5:5
y = -5:0.5:5
X = x' .* ones(length(y), 1)
Y = y .* ones(1, length(x))
Z = Y .* (sin.(X)) - X .* (cos.(Y))
s = surface(X, Y, Z, ; alpha=0.5)
plt_view(3)
```



在创建曲面对象之后可使用 s 访问并修改其属性。例如，通过设置 edgecolor 属性来隐藏边。

```
s.set_edgecolor("none")
plt_update()
```



3.3.1.6.5 mesh

网格曲面图

A. 语法

```
mesh(X,Y,Z)

mesh(Z)

mesh(ax,___)
mesh(___,Key=Value)
s = mesh(___)
```

B. 说明

mesh(X,Y,Z)创建一个网格图，该网格图为三维曲面，有实色边颜色，无面颜色。该函数将矩阵 Z 中的值绘制为由 X 和 Y 定义的 x-y 平面中的网格上方的高度。边颜色因 Z 指定的高度而异。

mesh(Z) 创建一个网格图，并将 Z 中元素的列索引和行索引用作 x 坐标和 y 坐标。

mesh(ax,___) 将图形绘制到 ax 指定的坐标区中，而不是当前坐标区中。指定坐标区作为第一个输入参数。

mesh(___,Key=Value)使用一个或多个名称-值对组参数指定曲面属性。例如，alpha=0.5 创建半透明网格图。

s = mesh(___) 将返回一个图曲面对象。在创建网格图后，使用 s 修改网格图。有关属性列表，请参阅 Surface 属性。

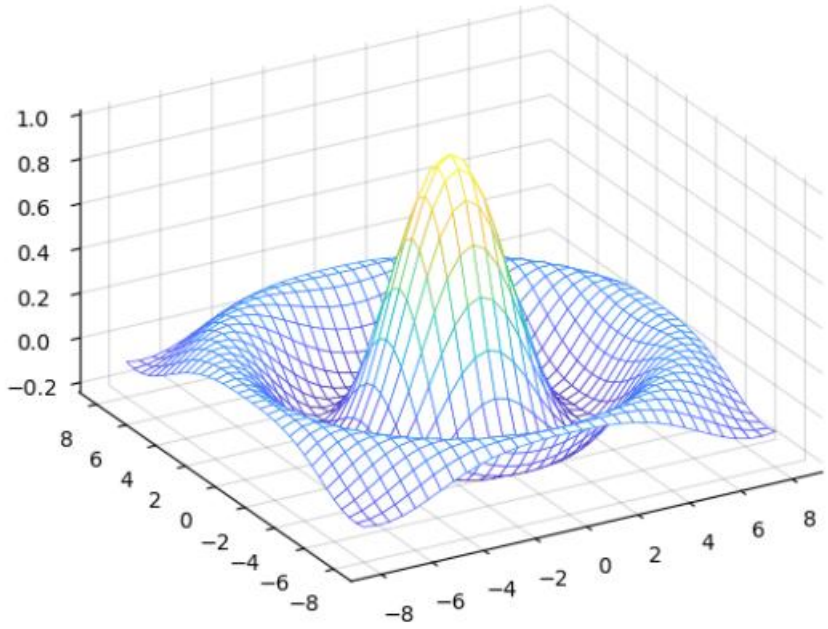
C. 示例

(3). 创建网格图

创建三个相同大小的矩阵。然后将它们绘制为一个网格图。该绘图使用 Z 确定高度和颜色。

```
X, Y = meshgrid2(-8:0.5:8, -8:0.5:8)

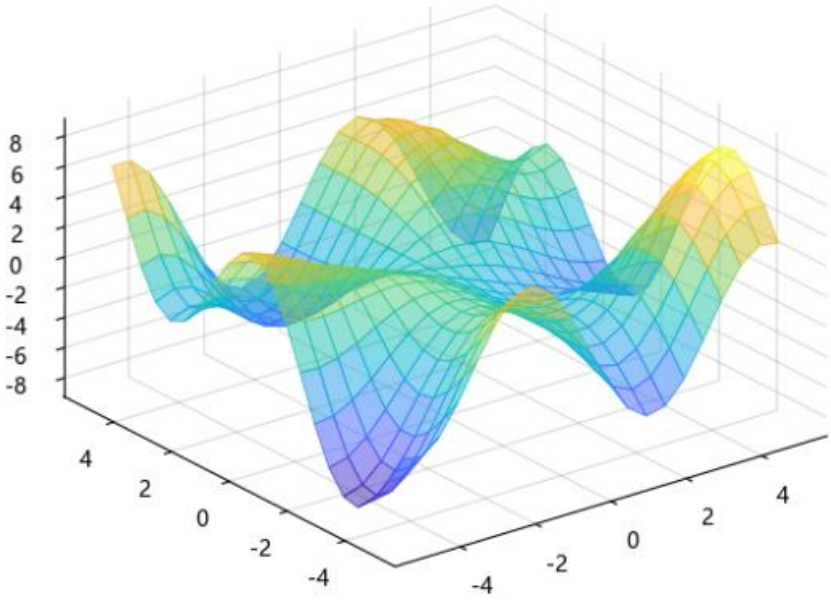
R = sqrt.(X .^ 2 .+ Y .^ 2) .+ eps()
Z = sin.(R) ./ R
m = mesh(X, Y, Z)
```



(4). 修改网格图外观

指定值为 0.5 的 alpha 名称-值对组，以创建半透明网格曲面。要允许进一步修改，请将曲面对象赋给变量 s。

```
X, Y = meshgrid2(-5:0.5:5, -5:0.5:5)
Z = Y .* sin.(X) - X .* cos.(Y)
s = mesh(X, Y, Z; facealpha=0.5)
s.set_facecolor("flat")
```



3.3.1.6.6 fsurf

绘制三维曲面

A. 语法

```
fsurf(f)
fsurf(f,xyinterval)

fsurf(funx,funy,funz)
fsurf(funx,funy,funz,uvinterval)

fsurf(___,Key=Value)
fsurf(ax,___)
fs = fsurf(___)
```

B. 说明

fsurf(f)在默认区间[-5 5]（对于 x 和 y）为函数 z=f(x,y)创建曲面图。

fsurf(f,xyinterval) 将在指定区间绘图。要对 x 和 y 使用相同的区间，请将 xyinterval 指定为 [min max] 形式的二元素向量。要使用不同的区间，请指定 [xmin xmax ymin ymax] 形式的四元素向量。

fsurf(funx,funy,funz)在默认区间[-55]（对于 u 和 v）绘制由 x=funx(u,v)、y=funy(u,v)、z=funz(u,v)定义的参数化曲面。

fsurf(funx,funy,funz,uvinterval) 将在指定区间绘图。要对 u 和 v 使用相同的区间，请将 uvinterval 指定为 [min max] 形式的二元素向量。要使用不同的区间，请指定 [umin umax vmin vmax] 形式的四元素向量。

fsurf(___,Key=Value)使用一个或多个名称-值对组参数指定曲面属性。在前面语法中的任何输入参数组合后使用此选项。

fsurf(ax,___) 将图形绘制到 ax 指定的坐标区中，而不是当前坐标区 (gca) 中。

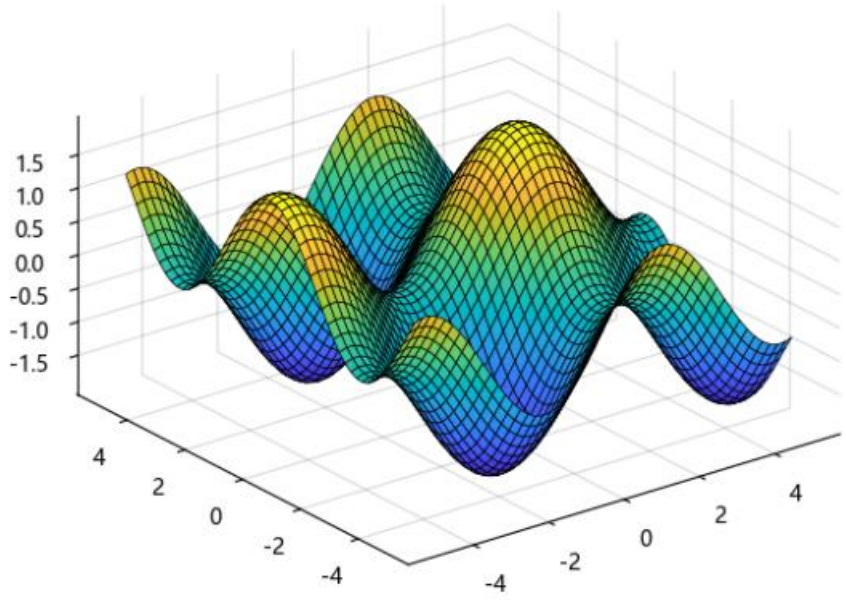
fs=fsurf(___)返回 surf 对象。在创建曲面之后可使用 s 对其进行修改。有关属性列表，请参阅 Surface 属性。

C. 示例

(1). 表达式的三维曲面图

在默认区间$-5 < x < 5$ 和 $-5 < y < 5$ 绘制表达式$\sin(x) + \cos(y)$ 。

```
funz = (x,y)->sin(x)+cos(y)
fsurf(funz)
```



(2). 指定曲面图区间并绘制分段表达式

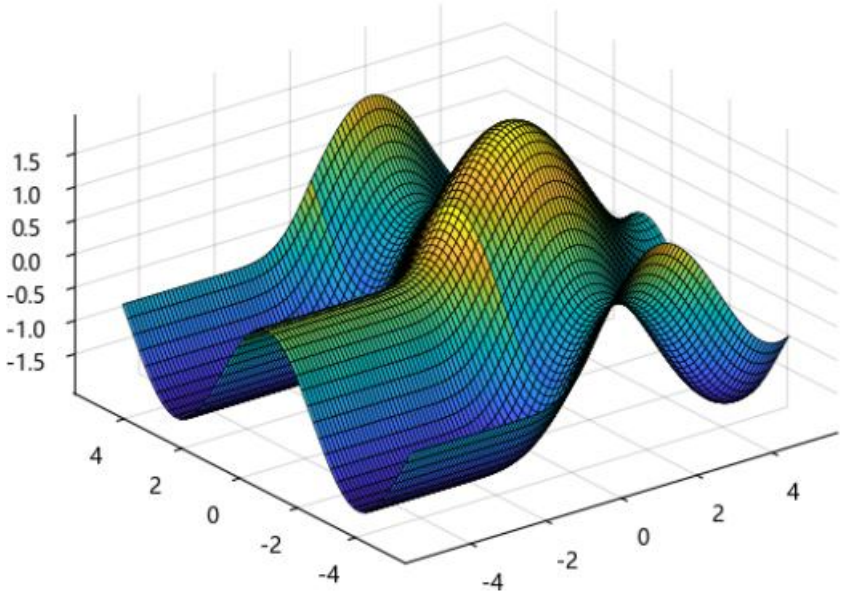
$$\begin{matrix} erf(x)+\cos(y) & -5<x<0 \\ \sin(x)+\cos(y) & 0<x<5 \end{matrix}$$

绘制分段表达式。区间为 $-5<y<5$。

指定绘图区间为 fsurf 的第二个输入参数。在相同坐标区的不同区间绘制多个曲面时，坐标轴范围会调整以包括所有数据。

```
f1 = (x,y)->erf(x)+cos(y)
fsurf(f1,[-5 0 -5 5])
hold("on")

f2 = (x,y)->sin(x)+cos(y)
fsurf(f2,[0 5 -5 5])
hold("off")
```



(3). 参数化曲面图

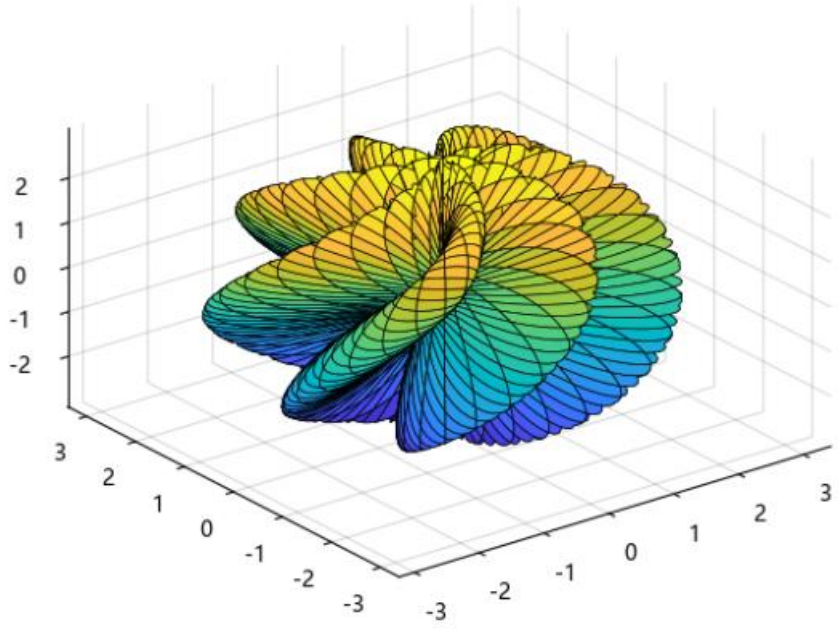
$$x = r\cos(u)\sin(v)$$
$$y = r\sin(u)\sin(v)$$
$$z = r\cos(v)$$

绘制参数化曲面。 where $r = 2 + \sin(7u + 5v)$

对于 0<u<2π，0<v<π。使用 meshdensity。

```
funr = (u,v)->2+sin(7 .*u + 5 .*v)
funx = (u,v)->funr(u,v) .* cos(u) .* sin(v)
funy = (u,v)->funr(u,v) .* sin(u) .* sin(v)
funz = (u,v)->funr(u,v) .* cos(v)

fsurf(funx,funy,funz,[0 2*pi 0 pi], meshdensity=186)
```

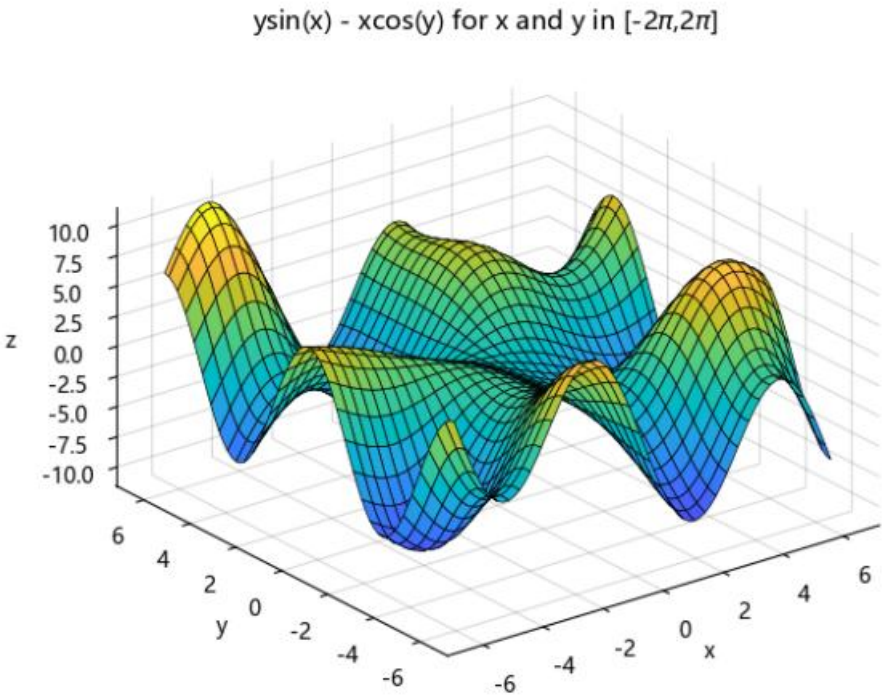


(4). 添加标题和轴标签以及格式化刻度

在从 -2π 到 2π 的区间为 x 和 y 绘制三维曲面 。添加标题和轴标签，并显示轴轮廓。

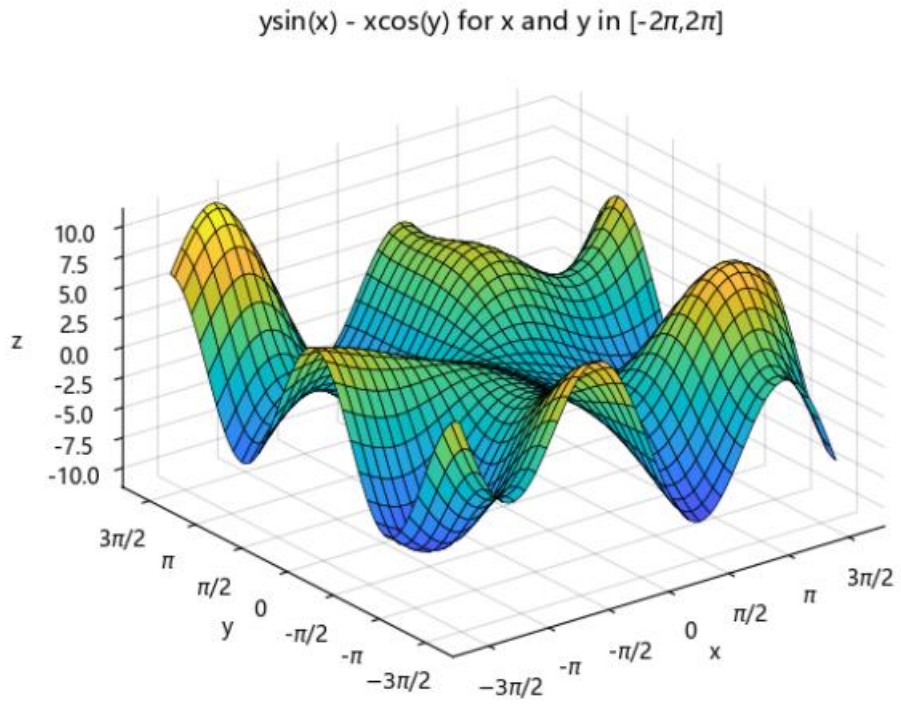
```
funz = (x,y)->y.*sin(x)-x.*cos(y)
fsurf(funz,[-2*pi 2*pi], meshdensity=70)

title(raw"ysin(x) - xcos(y) for x and y in [-2$\pi$,2$\pi$]")
xlabel("x");
ylabel("y");
zlabel("z");
```



使用坐标区对象的 XTickLabel 和 XTick 属性设置 x 轴刻度值和关联的标签。使用 gca 访问坐标区对象。按照同样的方式设置 y 轴刻度值和关联的标签。

```
ax = gca();
xticklabels(ax,       [raw"$-2\pi$",raw"$-3\pi$/2",raw"-$\pi$",raw"-
$\pi$/2","0",raw"$\pi$/2",raw"$\pi$",raw"3$\pi$/2",raw"2$\pi$"])
yticklabels(ax,       [raw"$-2\pi$",raw"$-3\pi$/2",raw"-$\pi$",raw"-
$\pi$/2","0",raw"$\pi$/2",raw"$\pi$",raw"3$\pi$/2",raw"2$\pi$"])
```

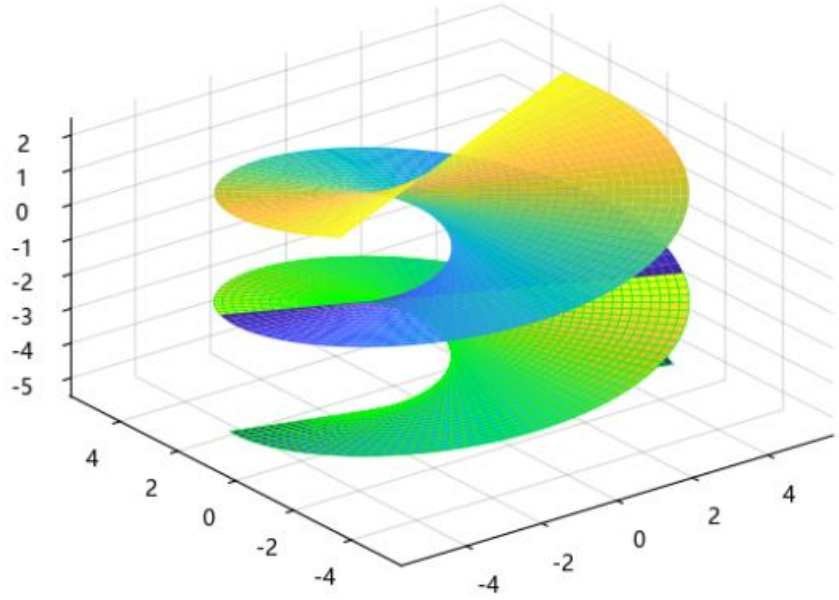


(5). 指定曲面属性

使用不同的线型为不同的 v 值绘制参数化曲面 、 和 z=v。对于 −5<v<−2，使用绿色虚线绘制曲面边。对于 −2<v<2，通过将 edgecolor 属性设置为 "none" 来关闭边。

```
funx = (u,v)->u.*sin(v)
funy = (u,v)->-u.*cos(v)
funz = (u,v)->v

fsurf(funx,funy,funz,[-5 5 -5 -2], edgecolor="g")
hold("on")
s = fsurf(funx,funy,funz,[-5 5 -2 2])
s.set_edgecolor("none")
hold("off")
```



(6). 创建后修改曲面

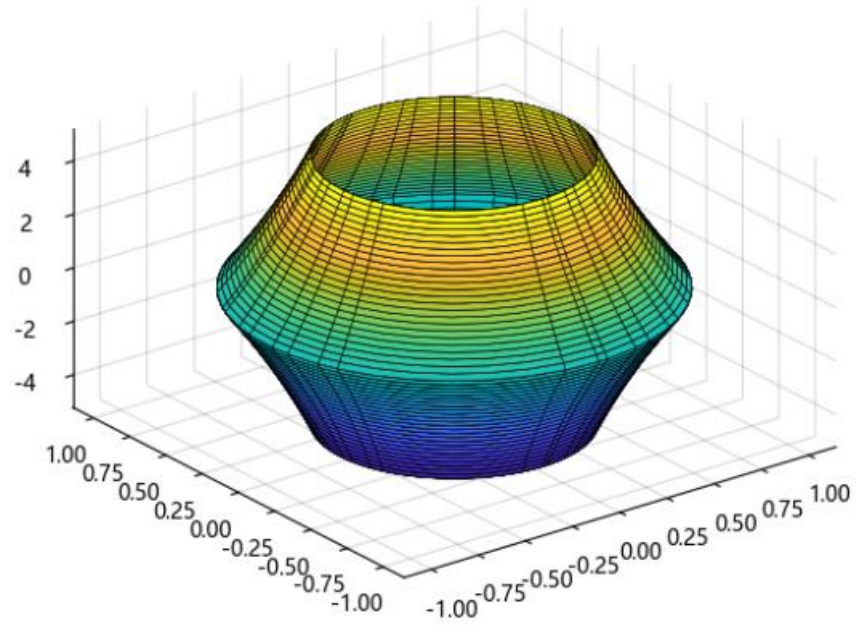
$$x = e^{-|u|/10}\sin(5|v|)$$
$$y = e^{-|u|/10}\cos(5|v|)$$
$$z = u$$

绘制参数化曲面。

将参数化函数曲面对象指定给变量。

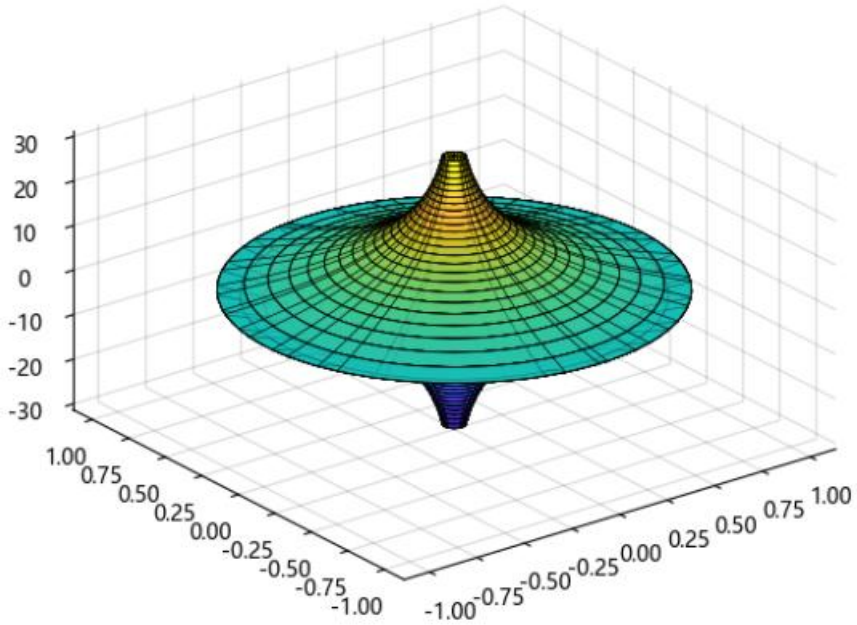
```
funx = (u,v)->exp(-abs(u)/10).*sin(5*abs(v))
funy = (u,v)->exp(-abs(u)/10).*cos(5*abs(v))
funz = (u,v)->u

figure()
s = fsurf(funx,funy,funz, meshdensity=300)
```



将 u 的绘图区间更改为 [-30 30]。通过将 facealpha 属性设置为 0（透明）到 1（不透明）之间的值，为曲面添加透明度。

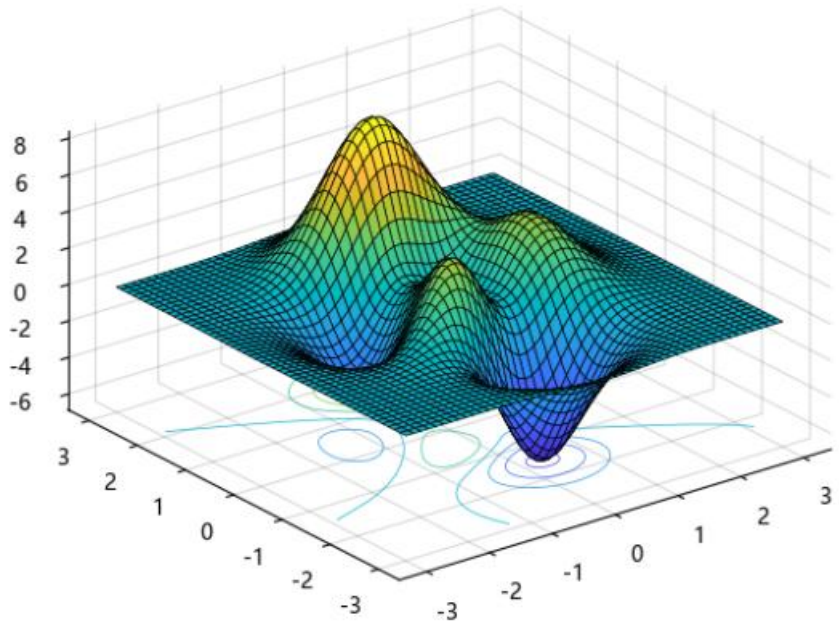
```
s = fsurf(funx,funy,funz,[-30 30 -5 5], meshdensity=300)
s.set_alpha(0.5)
```



(7). 显示曲面图下的等高线

通过将 showcontours 选项设置为 "on"，显示曲面图下的等高线。

```
funz = (x,y)->3*(1-x).^2 .*exp(-(x.^2)-(y+1).^2) - 10*(x/5 - x.^3 -
y.^5).*exp(-x.^2-y.^2) - 1/3*exp(-(x+1).^2 - y.^2);
fsurf(funz,[-3 3], showcontours="on")
```

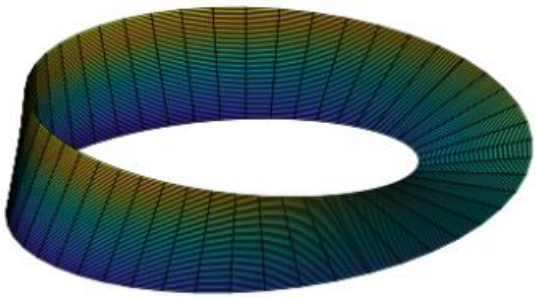


(8). 莫比乌斯环

绘制莫比乌斯环。

```
u = LinRange(-0.5, 0.5, 50);
v = LinRange(0, 2 * pi, 50);
U, V = meshgrid2(u, v);
r = 2 .+ U .* cos.(V / 2);
x = r .* cos.(V);
y = r .* sin.(V);
z = U .* sin.(V / 2);

surf(x, y, z)
plt_view(20, 30)
daspect([1, 1, 1])
axis("off")
```



3.3.1.6.7 fmesh

绘制三维网格图

A. 语法

```
fmesh(f)
fmesh(f,xyinterval)

fmesh(funx,funy,funz)
fmesh(funx,funy,funz,uvinterval)



fmesh(___,Key=Value)
fmesh(ax,___)
fs = fmesh(___)
```

B. 说明

fmesh(f)在 x 和 y 的默认区间[-5 5]为表达式 z=f(x,y)创建网格图。

fmesh(f,xyinterval)将在指定区间绘图。要对 x 和 y 使用相同的区间，请将 xyinterval 指定为[minmax]形式的二元素向量。要使用不同的区间，请指定[xminxmaxyminymax]形式的四元素向量。

fmesh(funx,funy,funz)在默认区间[-55]（对于 u 和 v）绘制由 x=funx(u,v)、y=funy(u,v)、z=funz(u,v)定义的参数化网格。

fmesh(funx,funy,funz,uvinterval) 在指定区间绘制参数化网格。要对 u 和 v 使用相同的区间，请将 uvinterval 指定为 [min max] 形式的二元素向量。要使用不同的区间，请指定 [umin umax vmin vmax] 形式的四元素向量。

mesh(___,Key=Value)使用一个或多个名称-值对组参数指定网格的属性。可以将此选项与前面语法中的任何输入参数组合一起使用。

fmesh(ax,___) 将图形绘制到 ax 指定的坐标区中，而不是当前坐标区 gca 中。

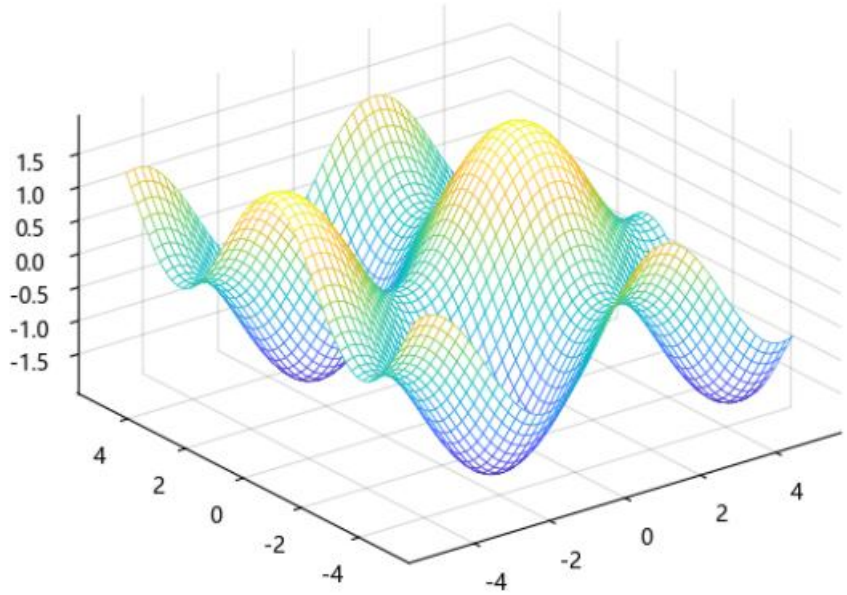
fs = fmesh(___) 将返回一个图曲面对象。在创建网格图后，使用 s 修改网格图。有关属性列表，请参阅 Surface 属性。

C. 示例

(1). 表达式的三维网格图

在默认区间 $-5 < x < 5$ 和 $-5 < y < 5$ 绘制 $\sin(x) + \cos(y)$ 输入的网格。

```
funz = (x,y)->sin(x)+cos(y)
fmesh(funz)
```



(2). 参数化网格图

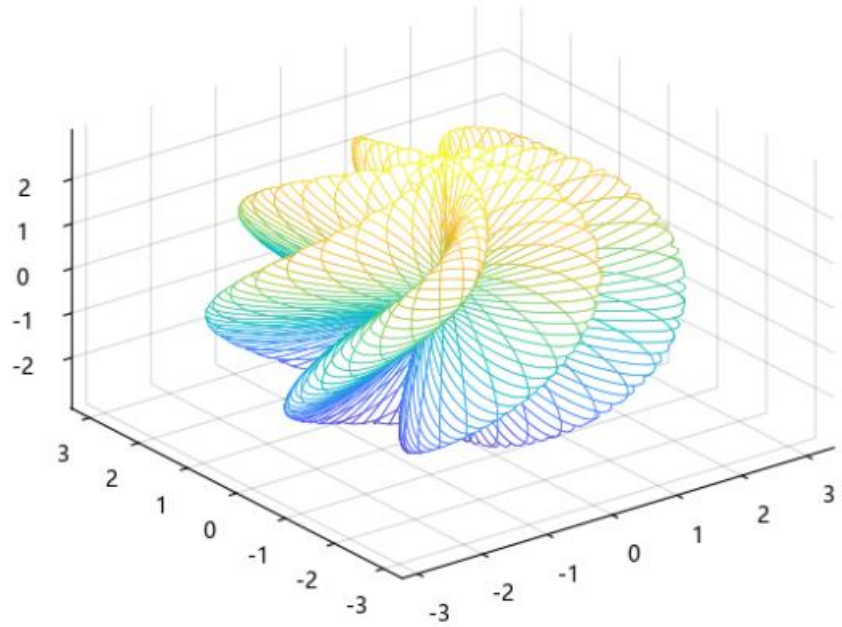
$$x = r\cos(s)\sin(t)$$
$$y = r\sin(s)\sin(t)$$
$$z = r\cos(t)$$
$$\text{where} \quad r = 2 + \sin(7s + 5t)$$

绘制参数化网格图。

对于 $0 < s < 2\pi$ ， $0 < t < \pi$ 。使用 alpha 使网格部分透明。

```
funr = (u,v)->2+sin(7 .*u + 5 .*v)
funx = (u,v)->funr(u,v) .* cos(u) .* sin(v)
funy = (u,v)->funr(u,v) .* sin(u) .* sin(v)
funz = (u,v)->funr(u,v) .* cos(v)
```

```
fmesh(funx,funy,funz,[0 2*pi 0 pi])
```



(3). 指定网格图区间并绘制分段输入

$$\begin{matrix} erf(x)+\cos(y) & -5<x<0 \\ \sin(x)+\cos(y) & 0<x<5 \end{matrix}$$

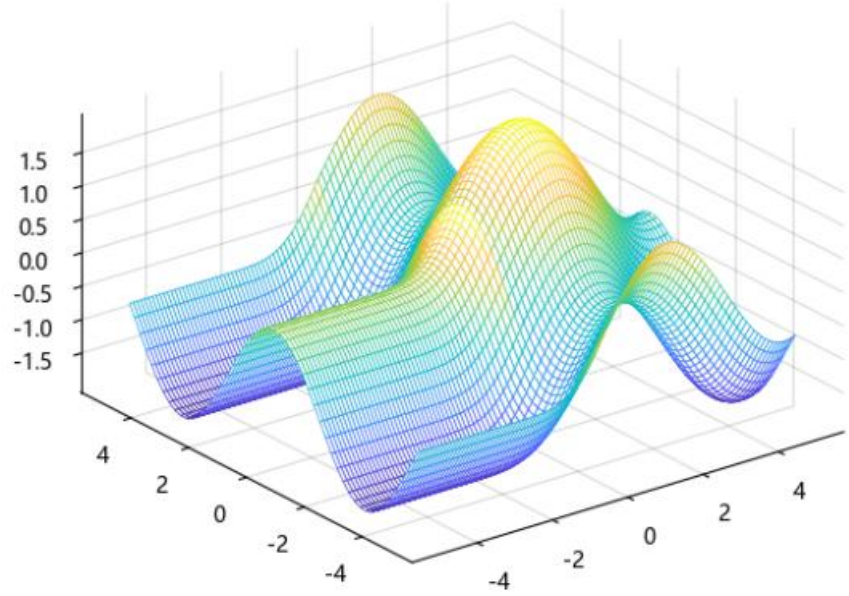
绘制分段输入。

区间为 $-5<y<5$。

指定绘图区间作为 fmesh 的第二个参数。在相同坐标区的不同区间绘制多个输入时，坐标轴范围会调整以包括所有数据。

```
using SpecialFunctions
f1 = (x,y)->erf(x)+cos(y)
fmesh(f1,[-5 0 -5 5])
hold("on")

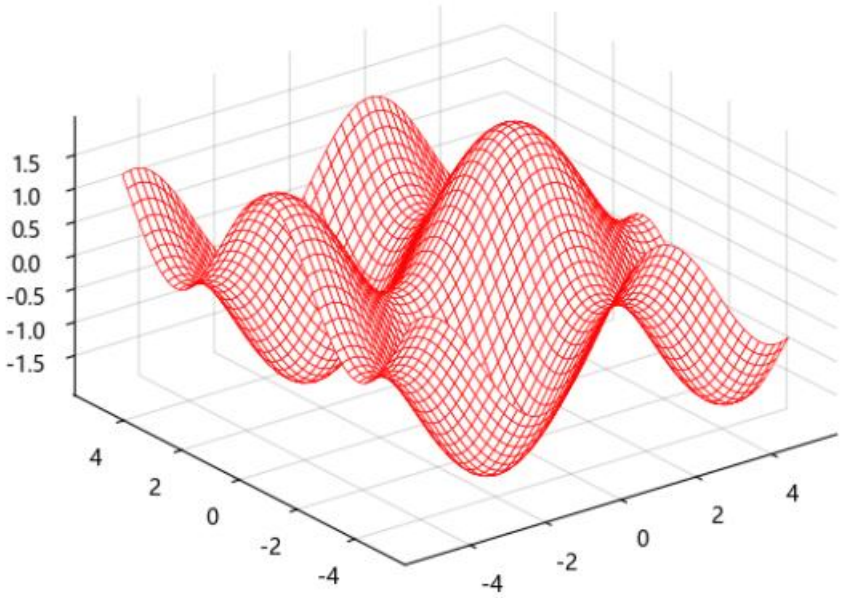
f2 = (x,y)->sin(x)+cos(y)
fmesh(f2,[0 5 -5 5])
hold("off")
```



(4). 指定网格图属性

使用红线创建网格图。

```
funz = (x,y)->sin(x)+cos(y)
fmesh(funz,edgecolor="red")
```



3.3.1.6.8 peaks

包含两个变量的示例函数

A. 语法

```
Z = peaks();
Z = peaks(n);
```

```
Z = peaks(X,Y);
X,Y,Z = peaks(...);
```

B. 说明

peaks 是从高斯分布转换和缩放得来的包含两个变量的函数，在演示 mesh、surf、contour 等函数中很有用。

Z = peaks(); 返回一个 49×49 矩阵。

Z = peaks(n); 返回一个 n×n 矩阵。

Z = peaks(X,Y); 在给定的 X 和 Y（必须大小相同）处计算 peaks 并返回大小相同的矩阵。

X,Y,Z = peaks(...); 返回另外两个矩阵 X 和 Y 用于参数绘图，例如 surf(X,Y,Z)。默认的 n 是 49。

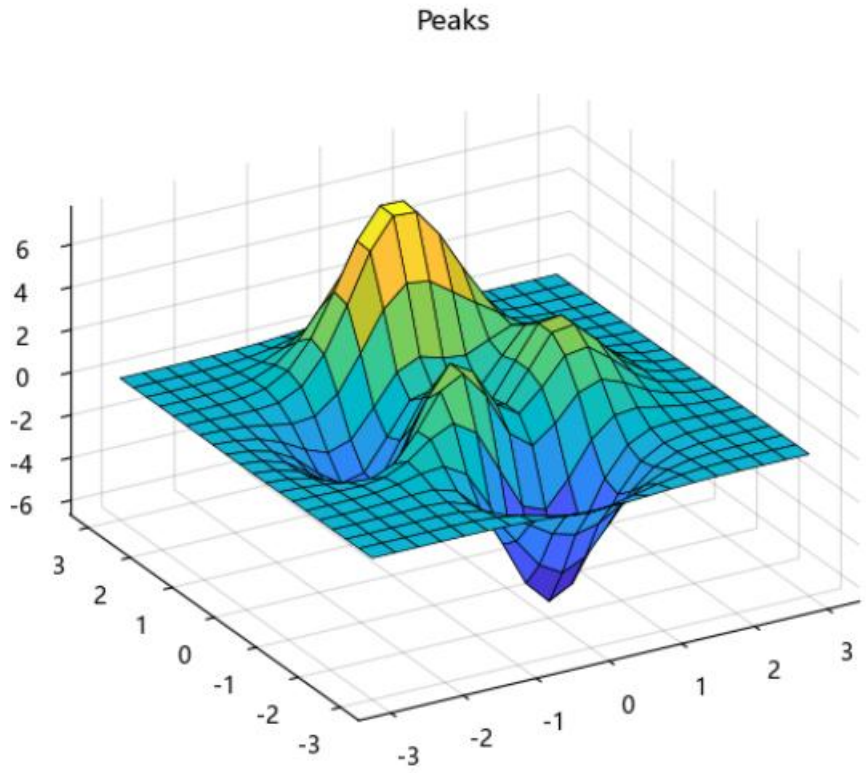
C. 示例

(1). 峰值曲面图

创建一个由峰值组成的 20×20 矩阵并显示该曲面。

```
figure()
X,Y,Z = peaks(20)

surf(X,Y,Z)
title("Peaks")
```



3.3.1.6.9 cylinder

创建圆柱

A. 语法

```
c = cylinder()
c = cylinder(r)
c = cylinder(r, n)
X, Y, Z = cylinder(___; fig=false)
```

B. 说明

c = cylinder() 返回圆柱对象，圆柱的 x、y 和 z 坐标为计算得到的三个 2×21 矩阵。圆柱的半径为 1，圆周上有 20 个等间距点。底面平行于 xy 平面。示例

c = cylinder(r) 返回圆柱对象，圆柱具有指定剖面曲线 r 和圆周上 20 个等距点的 x、y 和 z 坐标。该函数将 r 中的每个元素视为沿圆柱单位高度的等距高度的半径。每个坐标矩阵的大小为 m×21，其中 m=numel(r)。但是，如果 r 是标量，则 m=2。示例

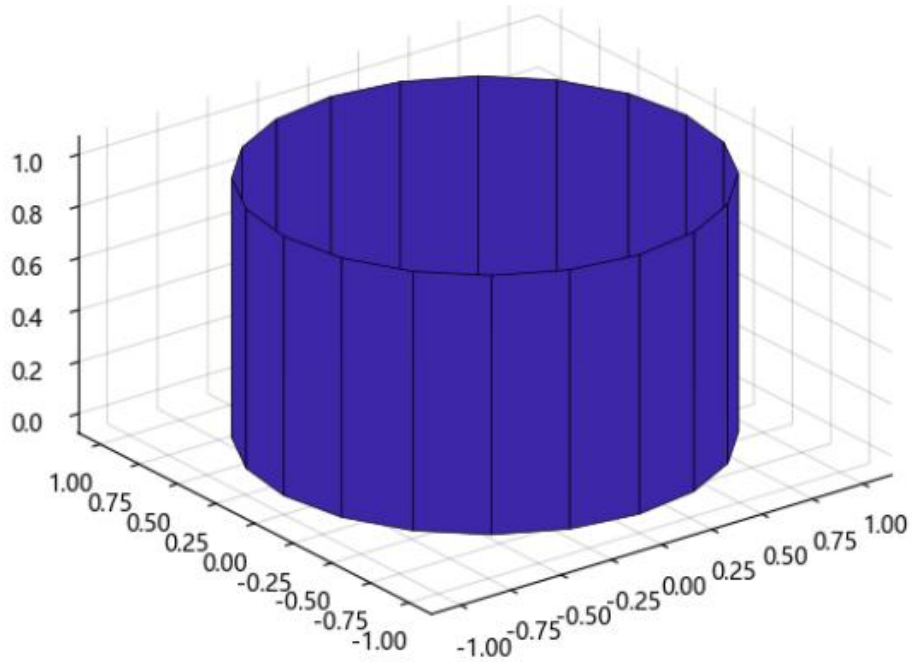
X, Y, Z = cylinder(r,n; fig=false) 返回具有指定剖面曲线 r 和圆周上 n 个等距点的圆柱的 x、y 和 z 坐标。每个坐标矩阵的大小为 m×(n+1)，其中 m=numel(r)。但是，如果 r 是标量，则 m=2。

C. 示例

(1). 显示单位圆柱

创建并绘制半径等于 1 的圆柱。

```
using TyPlot
cylinder()
```



(2). 指定圆柱半径和高度

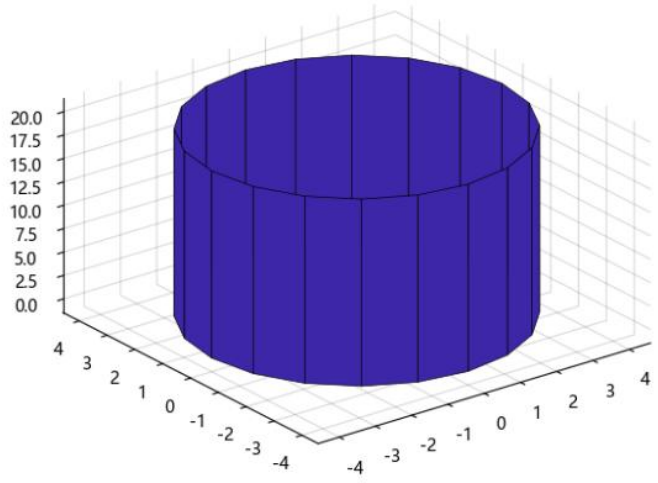
通过包括输入 r 来指定圆柱的半径。然后，通过修改返回的 Z 坐标来指定圆柱的高度。

将 X、Y 和 Z 定义为半径为 4 的圆柱的坐标。

```
using TyPlot
X, Y, Z = cylinder(4; fig=false)
```

通过修改 Z 坐标将高度指定为 20。绘制该圆柱。

```
h = 20
Z = Z .* h
surf(X, Y, Z)
```



(3). 在不同位置显示多个圆柱

创建一个圆柱，并使用返回的坐标在不同位置绘制多个圆柱。

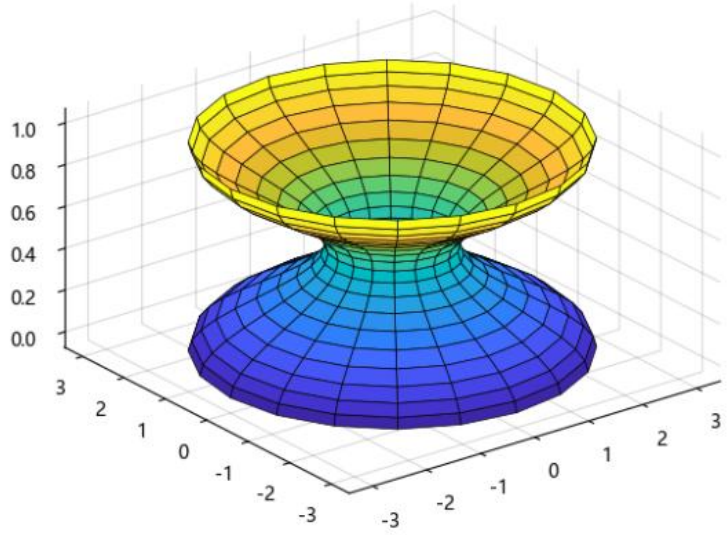
创建一个由剖面函数 $2 + \cos(t)$ 定义的圆柱。

```
using TyPlot

t = 0:(pi / 10):(2 * pi)
r = 2 .+ cos.(t')
X, Y, Z = cylinder(r; fig=false)
```

以原点为底面中心绘制圆柱。

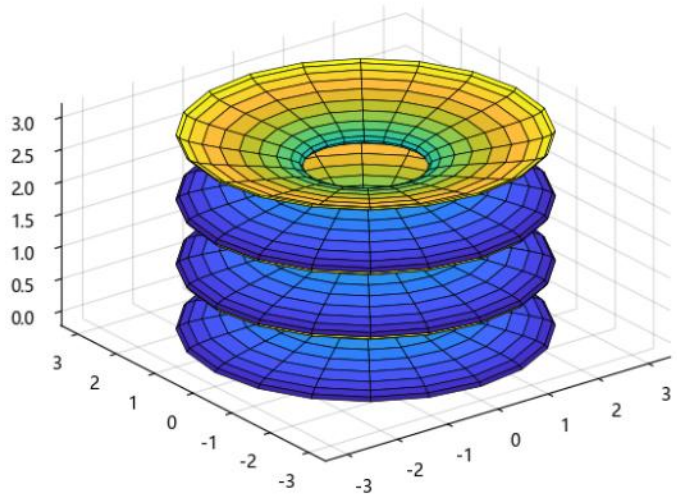
```
surf(X, Y, Z)
```



在第一个圆柱的顶部再绘制两个圆柱。

```
hold("on")
```

```
surf(X, Y, Z .+ 1)
surf(X, Y, Z .+ 2)
```



3.3.1.6.10 ellipsoid

创建椭圆体

A. 语法

```
X,Y,Z = ellipsoid(xc,yc,zc,xr,yr,zr)
X,Y,Z = ellipsoid(xc,yc,zc,xr,yr,zr,n)

ellipsoid(___)
ellipsoid(ax,___)
```

B. 说明

X,Y,Z = ellipsoid(xc,yc,zc,xr,yr,zr) 返回椭圆体的 x、y 和 z 坐标，但不进行绘制。返回的椭圆体的中心坐标为 (xc,yc,zc)，半轴长度为 (xr,yr,zr)，由 20×20 个面组成。示例

该函数以三个 21×21 矩阵形式返回 x、y 和 z 坐标。

要使用返回的坐标绘制椭圆体，请使用 surf 或 mesh 函数。

X,Y,Z = ellipsoid(xc,yc,zc,xr,yr,zr,n) 返回具有 n×n 个面的椭圆体的 x、y 和 z 坐标。该函数以三个 (n+1)×(n+1) 矩阵形式返回 x、y 和 z 坐标。示例

ellipsoid(___) 绘制椭圆体，但不返回坐标。此语法与上述语法中的任一前

面的输入参数结合使用。示例

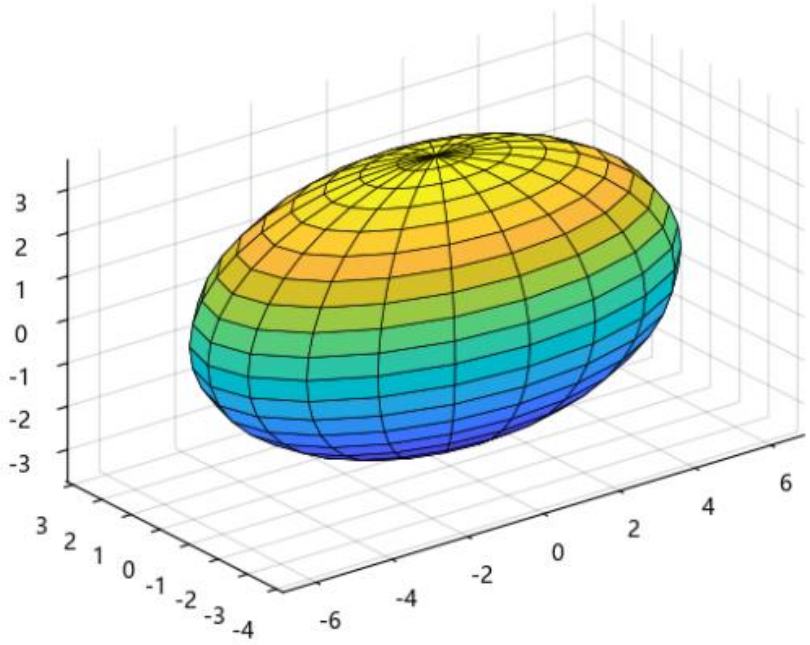
ellipsoid(ax,___) 将图形绘制到 ax 指定的坐标区中，而不是当前坐标区中。指定坐标区作为第一个输入参数。示例。

C. 示例

(1). 椭球创建和显示椭圆体

创建并绘制一个以 (0, －0.5, 0) 为中心、半轴长度为 (6, 3.25, 3.25) 的椭圆体。使用 axis("equal") 可沿每个坐标方向使用相等的数据单位。

```
ellipsoid(0, -0.5, 0, 6, 3.25, 3.25)
axis("equal");
```



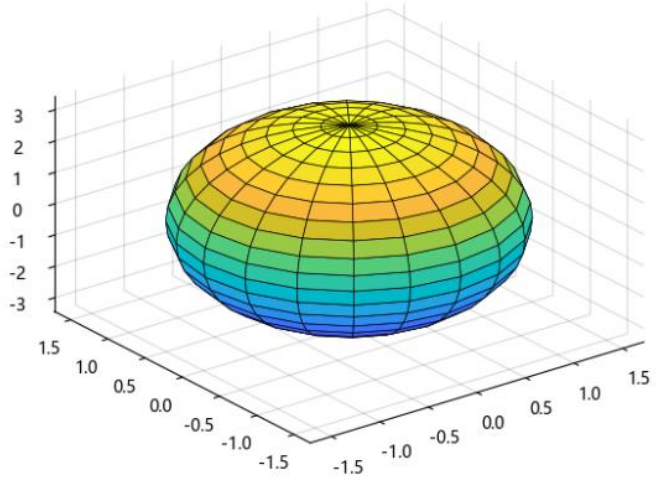
(2). 对椭圆体应用平移和旋转

生成以 (0, 0, 0) 为中心、半轴长度为 (1.5, 1.5, 3) 的椭圆体的坐标。

```
x, y, z = ellipsoid(0, 0, 0, 1.5, 1.5, 3; fig=false)
```

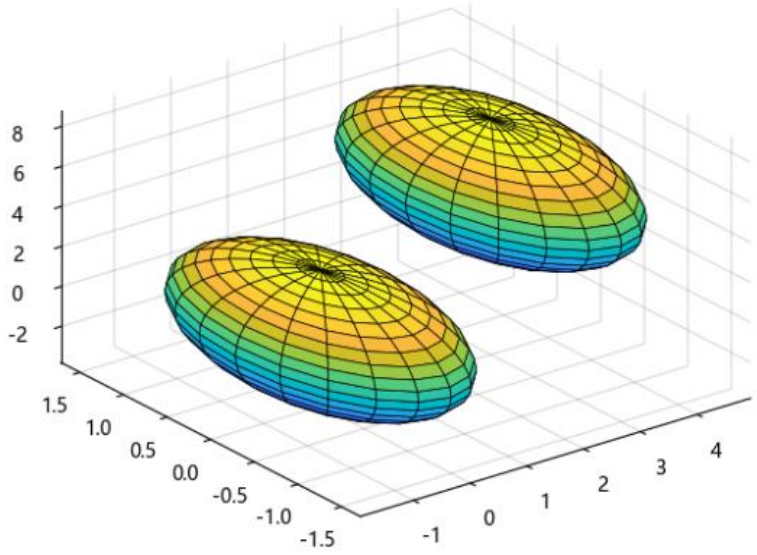
创建椭圆体的曲面图。

```
surf(x, y, z)
```



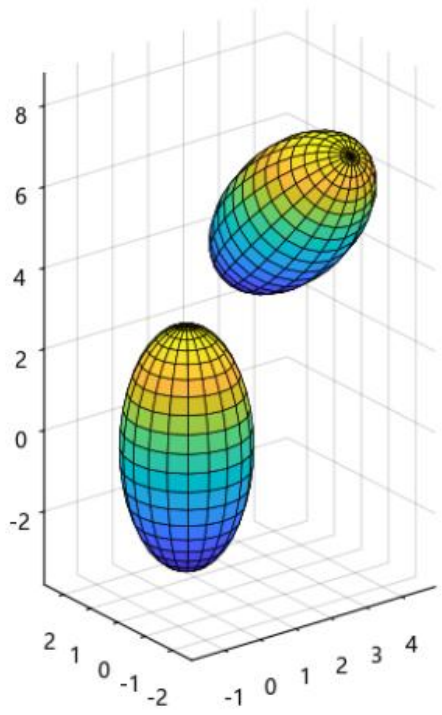
绘制第二个椭圆体，其中心从第一个椭圆体平移 (3, 0, 5)。为了能够在下一步骤中旋转第二个椭圆体，将曲面对象返回为 s。

```
hold("on")
s = surf(x .+ 3, y, z .+ 5)
```



将第二个椭圆体围绕其 x 轴旋转 45 度。平移和旋转后的椭圆体的新坐标存储在 s.Xdata、s.Ydata 和 s.Zdata 中。

```
s.remove()
direction = [1, 0, 0];
rotate(x .+ 3, y, z .+ 5, direction, 45)
axis("equal");
```



(3). 显示具有不同面数的椭圆体

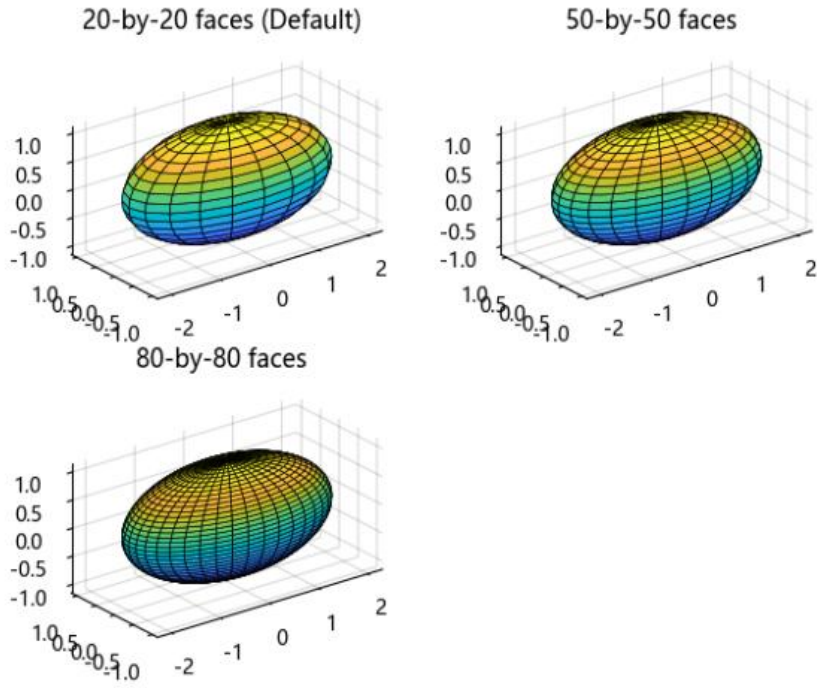
显示均以坐标 (0, 0, 0) 为中心、半轴长度为 (2, 1, 1) 但具有不同面数的多个椭圆体。

调用 subplot 函数以创建 2×2 分块图布局，调用 projection 参数来设置坐标区的投影方式，创建坐标区。然后，使用 ellipsoid 函数绘制三个具有不同面数的椭圆体。通过指定坐标区，在图的不同图块中绘制椭圆体。

```
ax1 = subplot(2, 2, 1; projection="3d")
ax2 = subplot(2, 2, 2; projection="3d")
ax3 = subplot(2, 2, 3; projection="3d")
ellipsoid(ax1, 0, 0, 0, 2, 1, 1)
axis("equal")

title("20-by-20 faces (Default)")
ellipsoid(ax2, 0, 0, 0, 2, 1, 1, 50)
title("50-by-50 faces")
axis("equal")

ellipsoid(ax3, 0, 0, 0, 2, 1, 1, 80)
title("80-by-80 faces")
axis("equal");
```



3.3.1.6.11 sphere

创建球面

A. 语法

```
X,Y,Z = sphere()
X,Y,Z = sphere(n)

sphere(___)
sphere(ax,___)
```

B. 说明

X,Y,Z = sphere() 返回球面的 x、y 和 z 坐标而不对其绘图。返回的球面的半径等于 1，由 20×20 个面组成。

该函数以三个 21×21 矩阵形式返回 x、y 和 z 坐标。示例

要使用返回的坐标绘制球面，请使用 surf 或 mesh 函数。

X,Y,Z = sphere(n) 返回半径等于 1 且包含 n×n 个面的球面的 x、y 和 z 坐标。该函数以三个 (n+1)×(n+1) 矩阵形式返回 x、y 和 z 坐标。示例

sphere(___) 绘制球面而不返回坐标。请将此语法与上述语法中的任何输入参数结合使用。示例

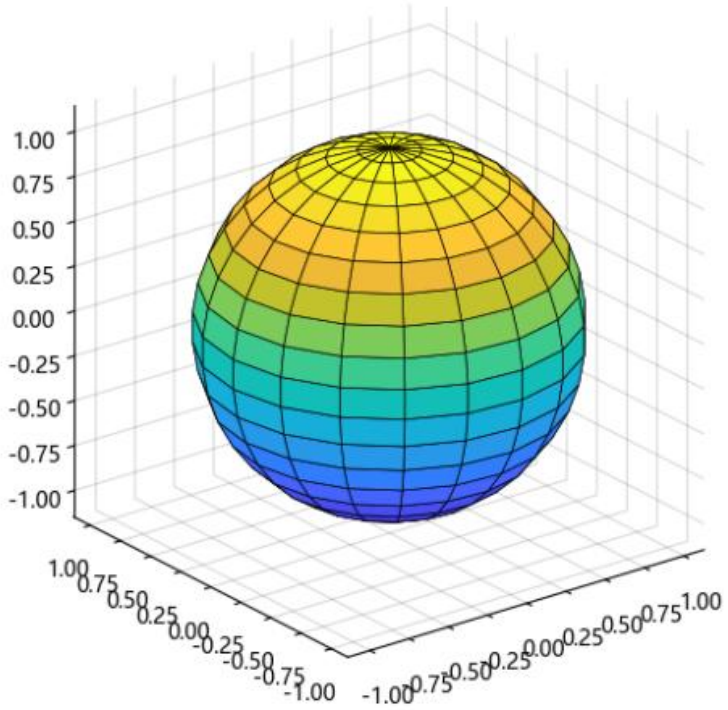
sphere(ax,___) 将图形绘制到 ax 指定的坐标区中，而不是当前坐标区中。指定坐标区作为第一个输入参数。示例。

C. 示例

(1). 显示单位球面

创建并绘制一个半径等于 1 的球面。使用 axis("equal") 可沿每个坐标方向使用相等的数据单位。

```
sphere()
axis("equal");
```



(2). 指定球面半径和位置

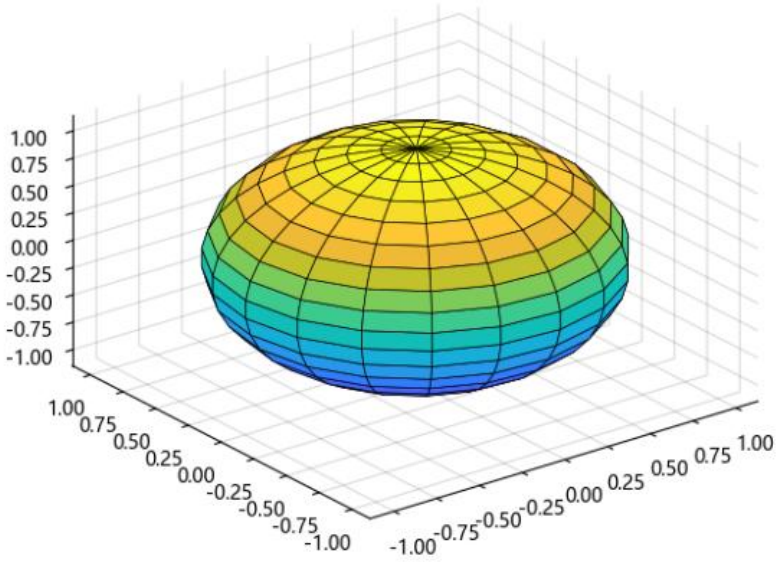
通过修改返回的 X、Y、Z 坐标来指定球面的半径和位置。

将 X、Y 和 Z 定义为单位球面的坐标。

```
X, Y, Z = sphere(; fig=false)
```

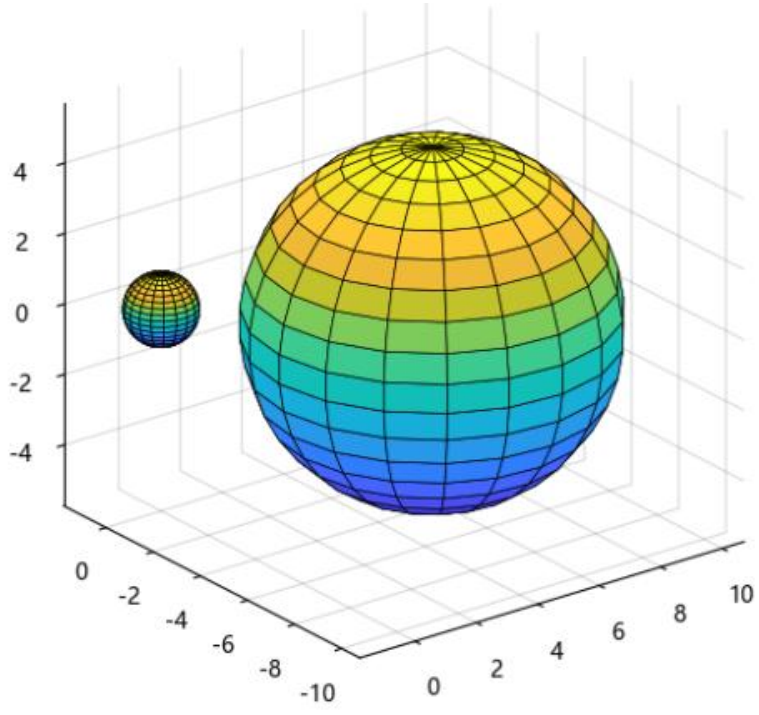
以原点为中心绘制单位球面。

```
surf(X, Y, Z)
```



通过乘以单位球面的坐标，将 X2、Y2 和 Z2 定义为半径为 5 的球面的坐标。以 (5,-5,0) 为中心绘制第二个球面。

```
hold("on")
r = 5
X2 = X .* r
Y2 = Y .* r
Z2 = Z .* r
s = surf(X2 .+ 5, Y2 .- 5, Z2)
axis("equal");
```



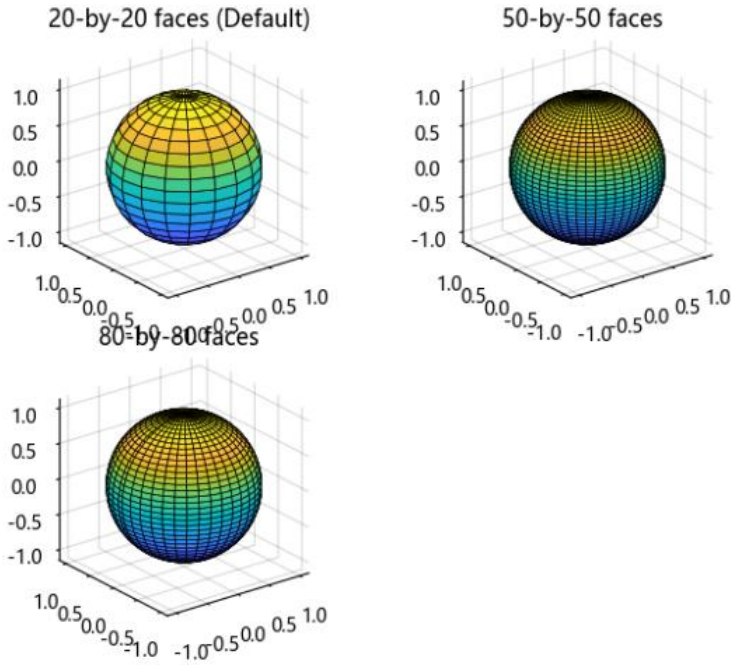
(3). 显示包含不同面数的球面

调用 subplot 函数以创建 2×2 分块图布局。调用 projection 参数来设置坐标区的投影方式，创建坐标区。然后，使用 sphere 函数，通过指定坐标轴，将三个具有不同面数的球面绘制到图的不同图块中。

```
ax1 = subplot(2, 2, 1; projection="3d")
ax2 = subplot(2, 2, 2; projection="3d")
ax3 = subplot(2, 2, 3; projection="3d")
sphere(ax1)
axis("equal")

title("20-by-20 faces (Default)")
sphere(ax2, 50)
title("50-by-50 faces")
axis("equal")

sphere(ax3, 80)
title("80-by-80 faces")
axis("equal");
```



3.3.1.6.12 pcolor

伪彩图

A. 语法

```
pcolor(C)
pcolor(X,Y,C)
pcolor(ax,___)
s = pcolor(___)
```

B. 说明

pcolor(C) 使用矩阵 C 中的值创建一个伪彩图。伪彩图以彩色单元（称为面）阵列形式显示矩阵数据。Syslab 在 x-y 平面上创建该图的单一着色平面图。该平面由对应于各面的角（即顶点）的 x 坐标和 y 坐标的网格定义。网格覆盖区域 X=1:n 和 Y=1:m，其中 [m,n] = size(C)。矩阵 C 指定顶点处的颜色。示例

pcolor(X,Y,C) 指定顶点的 x 坐标和 y 坐标。C 的大小必须与 x-y 坐标网格的大小匹配。例如，如果 X 和 Y 定义一个 m×n 网格，则 C 必须为 m×n 矩阵。示例

pcolor(ax,___) 指定绘图的目标坐标区。指定 ax 作为上述任何语法中的第一个参数。示例

s = pcolor(___) 返回 Surface 对象。创建绘图后，使用 s 设置绘图属性。示例。

C. 示例

(1). 用四种颜色绘制四个面

创建坐标向量 X 和 Y 以及名为 mymap 的颜色图，其中包含五种颜色：红色、绿色、蓝色、黄色和黑色。

```
using TyPlot

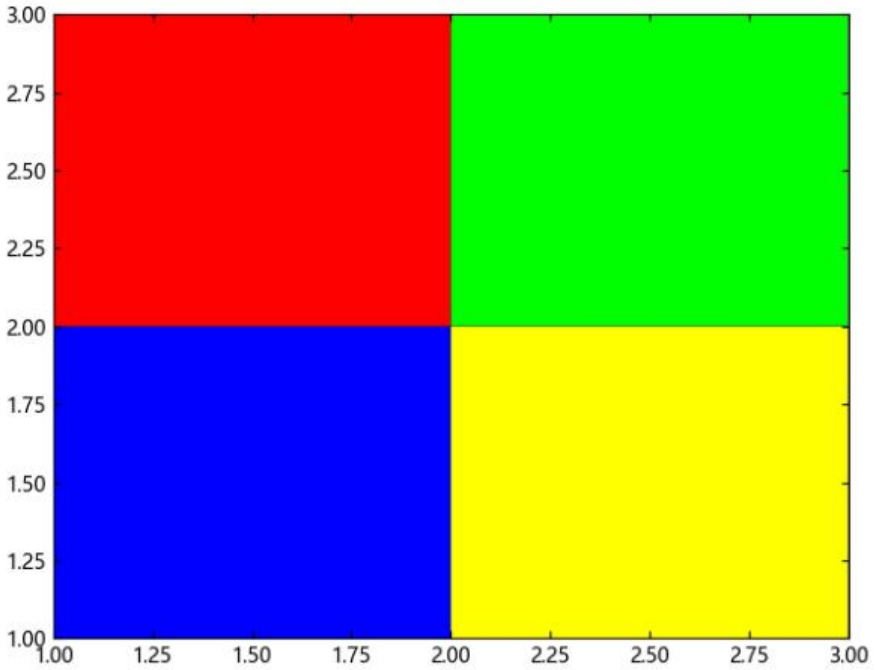
X = [1 2 3; 1 2 3; 1 2 3];
Y = X';
mymap = [1 0 0; 0 1 0; 0 0 1; 1 1 0; 0 0 0];
```

创建矩阵 C，它将颜色图映射到九个顶点。九个顶点中的四个顶点决定面的颜色。指定这些顶点的颜色，使各面分别为 红色 (1)、绿色 (2)、蓝色 (3) 和黄色 (4)。将其他顶点上的颜色设置为黑色 (5)。

```
C = [3 4 5; 1 2 5; 5 5 5];
```

对这些面绘图，并调用 colormap 函数以用 mymap 替换默认颜色图。

```
p = pcolor(X, Y, C)
colormap(p, mymap)
```



(2). 对 Hadamard 矩阵绘图

Hadamard 矩阵的元素是 1 或 -1。我们可以使用双色颜色图来可视化此矩阵。

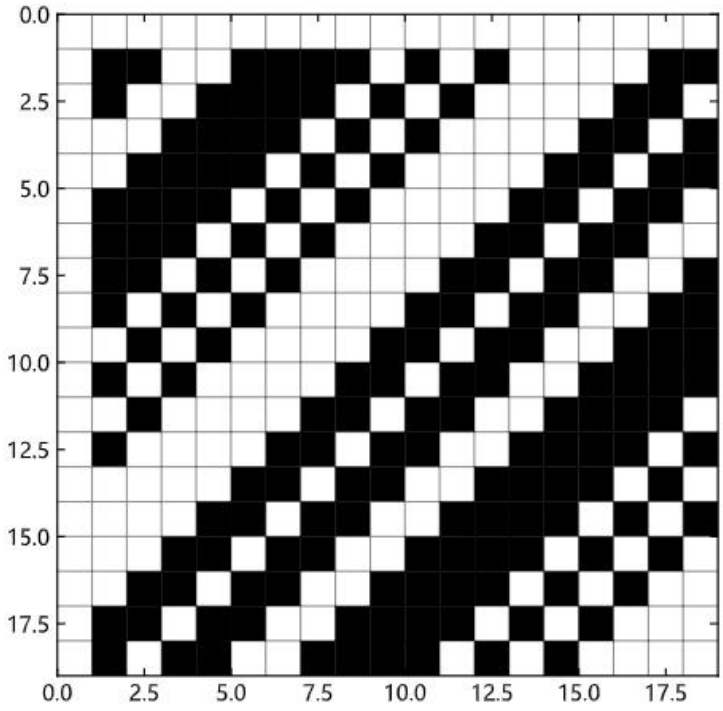
创建一个 20×20 Hadamard 矩阵。然后用黑白颜色图绘制该矩阵。将轴线设置为相等的长度。

```
using TyPlot

C = [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
    1 -1 -1 1 1 -1 -1 -1 -1 1 -1 1 -1 1 1 1 1 -1 -1 1
    1 -1 1 1 -1 -1 -1 -1 1 -1 1 -1 1 1 1 1 -1 -1 1 -1
    1 1 1 -1 -1 -1 -1 1 -1 1 -1 1 1 1 1 -1 -1 1 -1 -1
    1 1 -1 -1 -1 -1 1 -1 1 -1 1 1 1 1 -1 -1 1 -1 -1 1
    1 -1 -1 -1 -1 1 -1 1 -1 1 1 1 1 -1 -1 1 -1 -1 1 1
    1 -1 -1 -1 1 -1 1 -1 1 1 1 1 -1 -1 1 -1 -1 1 1 -1
    1 -1 -1 1 -1 1 -1 1 1 1 1 -1 -1 1 -1 -1 1 1 -1 -1
    1 -1 1 -1 1 -1 1 1 1 1 -1 -1 1 -1 -1 1 1 -1 -1 -1
    1 1 -1 1 -1 1 1 1 1 -1 -1 1 -1 -1 1 1 -1 -1 -1 -1
    1 -1 1 -1 1 1 1 1 -1 -1 1 -1 -1 1 1 -1 -1 -1 -1 1
    1 1 -1 1 1 1 1 -1 -1 1 -1 -1 1 1 -1 -1 -1 -1 1 -1
    1 -1 1 1 1 1 -1 -1 1 -1 -1 1 1 -1 -1 -1 -1 1 -1 1
    1 1 1 1 1 -1 -1 1 -1 -1 1 1 -1 -1 -1 -1 1 -1 1 -1
    1 1 1 1 -1 -1 1 -1 -1 1 1 -1 -1 -1 -1 1 -1 1 -1 1
```

```
    1  1  1 -1 -1  1 -1 -1  1  1 -1 -1 -1 -1  1 -1  1 -1  1  1
    1  1 -1 -1  1 -1 -1  1  1 -1 -1 -1 -1  1 -1  1 -1  1  1  1
    1 -1 -1  1 -1 -1  1  1 -1 -1 -1 -1  1 -1  1 -1  1  1  1  1
    1 -1  1 -1 -1  1  1 -1 -1 -1 -1  1 -1  1 -1  1  1  1  1 -1
    1  1 -1 -1  1  1 -1 -1 -1 -1  1 -1  1 -1  1  1  1  1 -1 -1];

img = pcolor(C)
colormap(img, gray(2))
axis("ij")
axis("square")
```

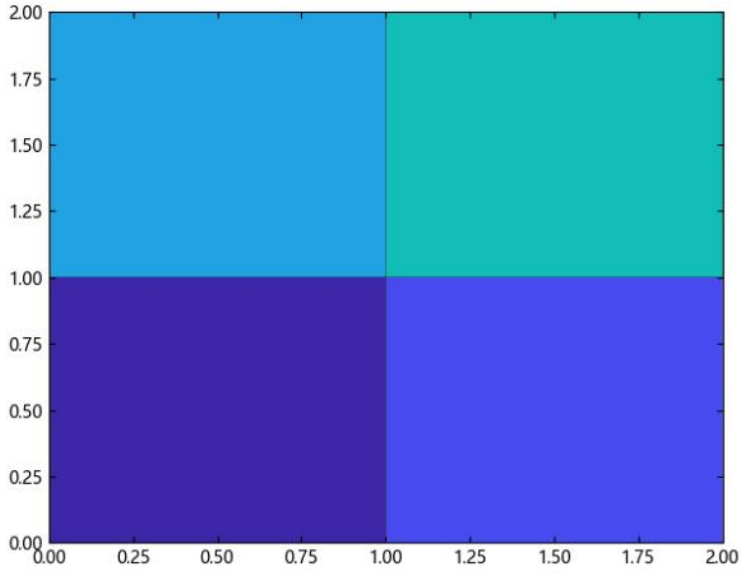


(3). 修改边框

创建颜色矩阵 C。然后创建 C 的伪彩图，并将 Surface 对象存储在返回参数 s 中。

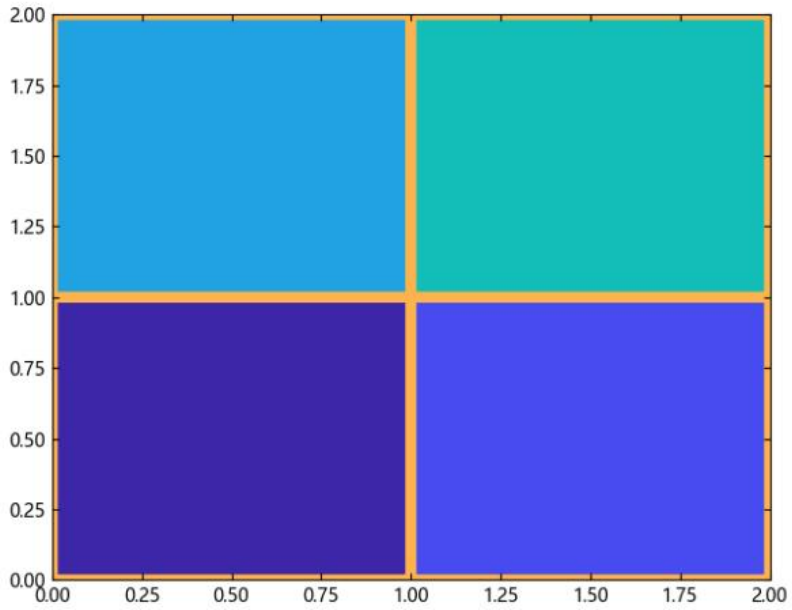
```
using TyPlot

C = [1 2 3; 4 5 6; 7 8 9];
s = pcolor(C);
```



通过设置 s 的 EdgeColor 属性来更改边框颜色。通过设置 LineWidth 属性使边框更粗。

```
s.set_edgecolor([1, 0.7 ,0.3])
s.set_linewidth(6);
plt_update()
```

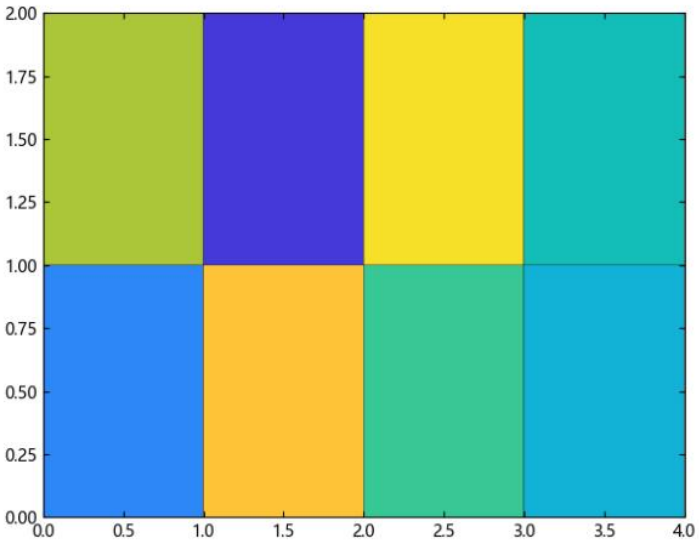


(4). 跨面进行颜色插值

创建颜色矩阵 C。然后创建 C 的伪彩图，并将 Surface 对象存储在返回参数 s 中。

```
using TyPlot
C = [5 13 9 7 12; 11 2 14 8 10; 6 1 3 4 15];
```

```
s = pcolor(C);
```



(5). 指定半对数网格

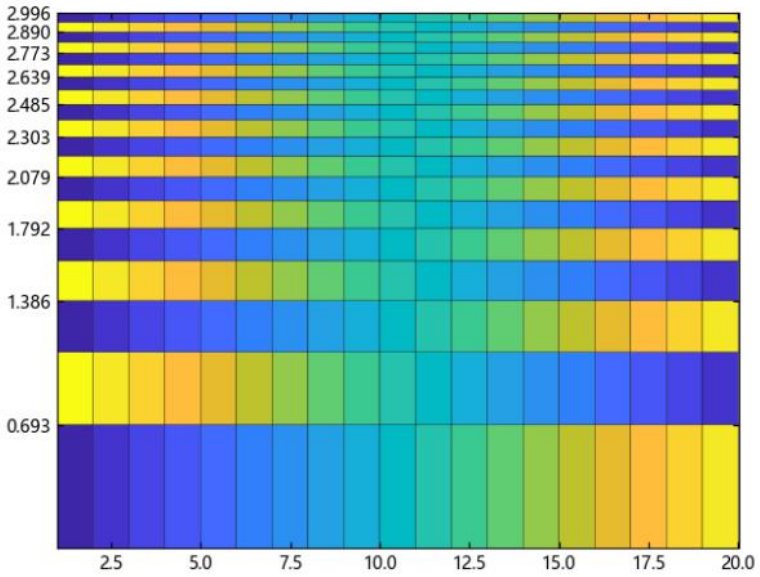
创建矩阵 X 和 Y，它们定义顶点的固定间隔网格。通过计算 Y 的对数得到矩阵 LY 。然后创建矩阵 C，其中包含交替的颜色索引行组。

```
using TyMath

X, Y = meshgrid2(1:20)
LY = log.(Y)
colorscale = [1:20 20:-1:1]';
C = repeat(colorscale, 10, 1);
```

使用 C 中指定的颜色绘制 X 和 LY。然后调整 y 轴上的刻度标签。

```
s = pcolor(X, LY, C);
tickvals = LY[2:2:20,1];
gca().set_yticks(tickvals);
plt_update()
```



(6). 指定参数化网格

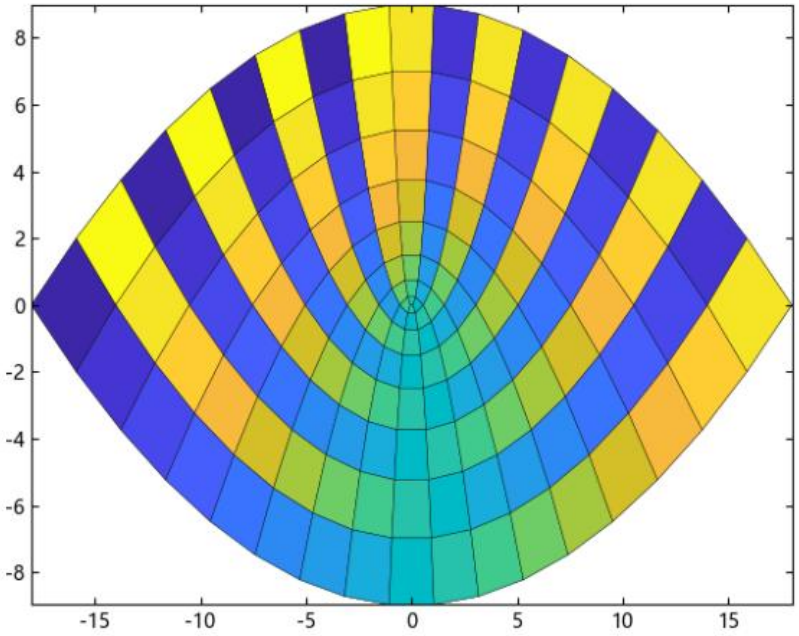
创建矩阵 X 和 Y，它们定义顶点的固定间隔网格。通过计算 X 和 Y 的函数得到矩阵 XX 和 YY 。然后创建矩阵 C，其中包含交替的颜色索引行组。

```
using TyMath

X, Y = meshgrid2(-3:6/17:3);
XX = 2 * X .* Y;
YY = X .^ 2 - Y .^ 2;
colorscale = [1:18 18:-1:1]';
C = repeat(colorscale, 9, 1);
```

使用 C 中的颜色绘制 XX 和 YY。

```
s = pcolor(XX, YY, C);
```



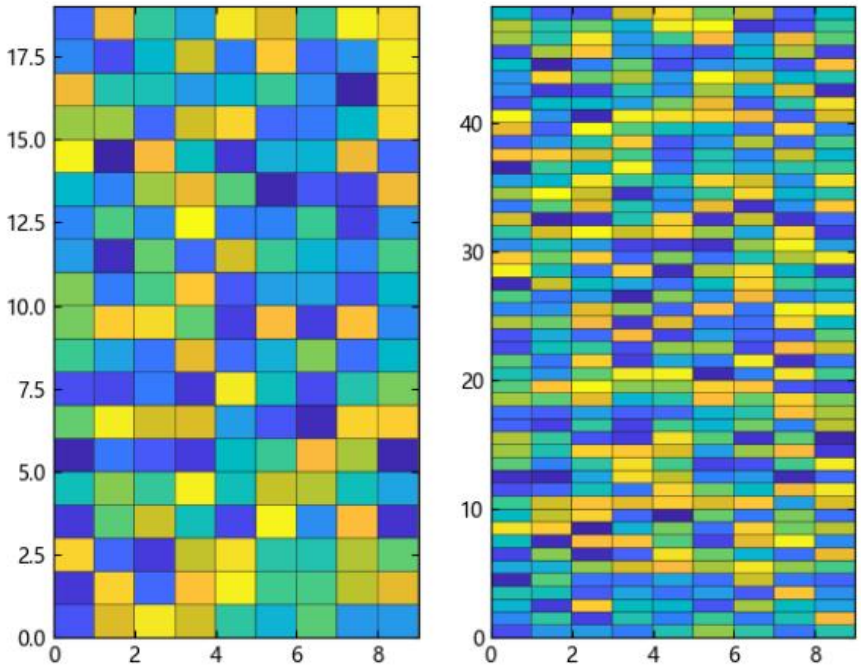
(7). 指定目标坐标区

调用 subplot 函数以创建坐标区对象 ax1 和 ax2。通过将这些坐标区指定为 pcolor 的第一个参数，创建两个伪彩图。

```
using TyPlot

ax1 = subplot(1, 2, 1)
C1 = rand(20, 10);
s1 = pcolor(ax1, C1);

ax2 = subplot(1, 2, 2)
C2 = rand(50, 10);
s2 = pcolor(ax2, C2);
```



3.3.1.6.13 ribbon

条带图

A. 语法

```
ribbon(Y)
ribbon(X,Y)
ribbon(X,Y,width)
ribbon(axes_handle,...)
h = ribbon(...)
```

B. 说明

ribbon(Y) 使用 X = 1:size(Y, 1) 将 Y 列绘制为宽度均匀的三维条带。条带以单位间隔沿 x 轴前进，在刻度线上居中显示，宽度为四分之三单位。条带以线性方式将 X 中的值映射到颜色图中的颜色。要更改图形中的条带颜色，请更改 colormap。

ribbon(X, Y) 为 Y 中的数据绘制三维条带，在 X 中指定的位置居中显示。X 和 Y 是大小相同的向量或矩阵。另外，X 可以是行或列向量，Y 是包含 length(X) 行的矩阵。当 Y 为矩阵时，ribbon 将 Y 中的每列绘制为位于对应 X 位置的条带。

ribbon(X, Y, width) 指定条带的宽度。默认值为 0.75。如果 width = 1，则各条带相互接触，沿 z 轴向下查看时它们紧挨在一起。如果 width > 1，则条带相互重叠并可能相交。

ribbon(axes_handle, ...) 将图形绘制到带有句柄 axes_handle 的坐标区中，而不是当前坐标区 (gca) 中。

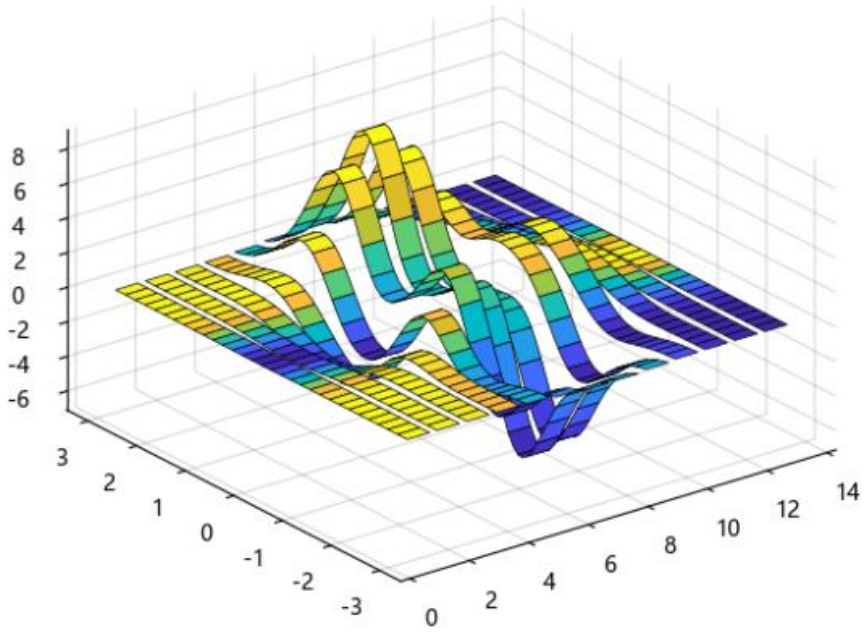
h = ribbon(...) 将句柄向量返回给曲面图形对象。ribbon 为每个条带返回一个句柄。

C. 示例

(1). 条带图

创建条带图。

```
x, y = meshgrid2(-3:0.5:3, -3:0.1:3)
source_path = pkgdir(TyPlot) * "/examples/ribbon/0-zdata.jl"
include(source_path)
ribbon(y, z)
```



3.3.1.6.14 flow

包含三个变量的简单函数

A. 语法

```
v = flow()
```

```
v = flow(n)
v = flow(x,y,z)
x,y,z,v = flow(...)
```

B. 说明

flow 是一个包含三个变量的函数，可生成用于演示 slice、interp3 和其他可视化标量体数据的函数的流体流动数据。

v = flow( ) 生成一个 25×50×25 数组。

v = flow(n) 生成 n×2n×n 数组。

v = flow(x, y ,z) 计算 x、y 和 z 位置的速度剖面。

x, y, z, v = flow(...) 返回坐标以及体数据。。

3.3.1.6.15 plt_fill

填充的二维多边形

A. 语法

```
plt_fill(X,Y,C)
plt_fill(X,Y,ColorSpec)
plt_fill(X1,Y1,C1,X2,Y2,C2,...)
plt_fill(...,Key=Value)
plt_fill(ax,...)
h = plt_fill(...)
```

B. 说明

plt_fill() 函数创建彩色多边形。

plt_fill(X,Y,C) 根据 X 和 Y 中的数据创建填充的多边形（顶点颜色由 C 指定）。C 是一个用作颜色图索引的向量或矩阵。如果 C 为行向量，length(C) 必须等于 size(X,2) 和 size(Y,2)；如果 C 为列向量，length(C) 必须等于 size(X,1) 和 size(Y,1)。必要时，plt_fill 可将最后一个顶点与第一个顶点相连以

闭合多边形。

plt_fill(X,Y,ColorSpec) 用 ColorSpec 指定的颜色填充多边形，指定的颜色可以是下列值之一：

颜色名称或短名称："red"（或 "r"）、"green"（或 "g"）、"blue"（或 "b"）、"cyan"（或 "c"）、"magenta"（或 "m"）、"yellow"（或 "y"）、"black"（或 "k"）、"white"（或 "w"）。

指定为 1×3 向量的 RGB 三元组，其元素是颜色的红色、绿色和蓝色分量的强度。强度必须在 [0, 1] 范围内。例如，[0.7,0.8,1] 指定浅蓝色。

plt_fill(X1,Y1,C1,X2,Y2,C2,...) 指定多个二维填充区。

plt_fill(...,Key=Value) 允许您为补片图形对象指定属性名称和值。

plt_fill(ax,...) 将在由 ax 指定的坐标区而不是当前坐标区 (gca()) 中创建多边形。选项 ax 可以位于前面的语法中的任何输入参数组合之前。

h = plt_fill(...) 返回由补片对象构成的向量。

C. 示例

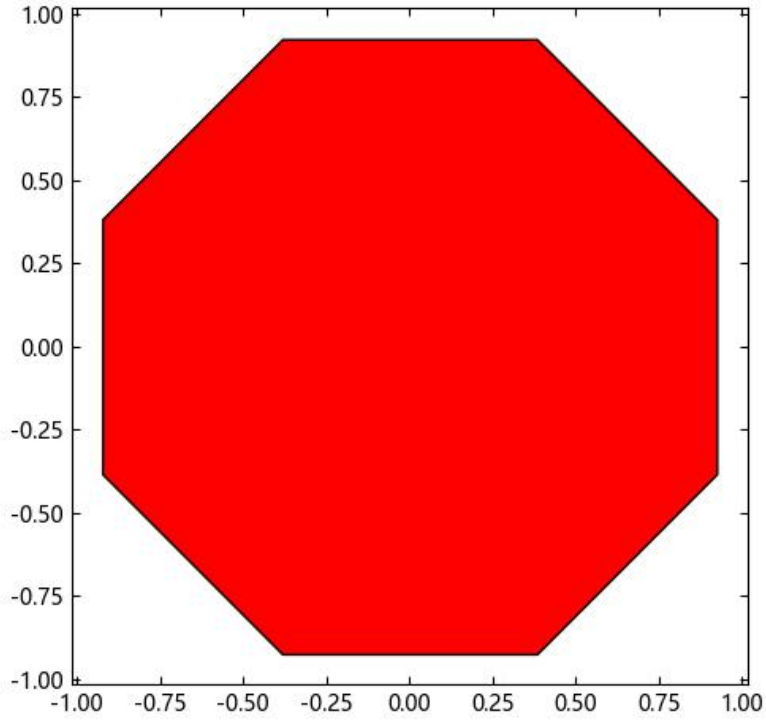
(1). 创建红色八边形

定义该数据。

```
t = (1/16:1/8:1)'*2*pi;
x = cos.(t);
y = sin.(t);
```

使用 plt_fill 函数创建一个红色八边形。

```
plt_fill(x,y,"r")
axis("square")
```



3.3.1.6.16 patch

绘制一个或多个填充多边形区域

A. 语法

```
patch(X,Y,C)
patch(X,Y,Z,C)

patch(S)

patch(___,Key=Value)
patch(ax,___)

p = patch(___)
```

B. 说明

patch(X,Y,C)使用 X 和 Y 的元素作为每个顶点的坐标，以绘制一个或多个填充多边形区域。patch 以您指定顶点的顺序连接这些顶点。要创建一个多边形，请将 X 和 Y 指定为向量。要创建多个多边形，请将 X 和 Y 指定为矩阵，其中

每一列对应于一个多边形。C 决定多边形的颜色。

patch(X,Y,Z,C) 使用 X、Y 和 Z 在三维坐标中创建多边形。要在三维视图中查看这些多边形，请使用 view(3) 命令。C 确定多边形的颜色。

patch(S)使用结构体 S 创建一个或多个多边形。结构体字段对应于补片属性名称，字段值对应于属性值。例如，S 可以包含字段 Faces 和 Vertices。

patch(___,Key=Value)创建多边形，并使用名称-值对组参数指定一个或多个补片属性。补片是包含所有已创建多边形的数据的对象。您可以使用上述语法中的任意输入参数组合指定补片属性。例如，linewidth=2 将所有多边形的轮廓宽度设置为 2 磅。

patch(ax,___) 将在由 ax 指定的坐标区中，而不是当前坐标区 (gca) 中创建补片。选项 ax 可以位于前面的语法中的任何输入参数组合之前。

p = patch(___) 返回包含所有多边形的数据的补片对象。使用 p 可在创建补片对象后查询并修改其属性。

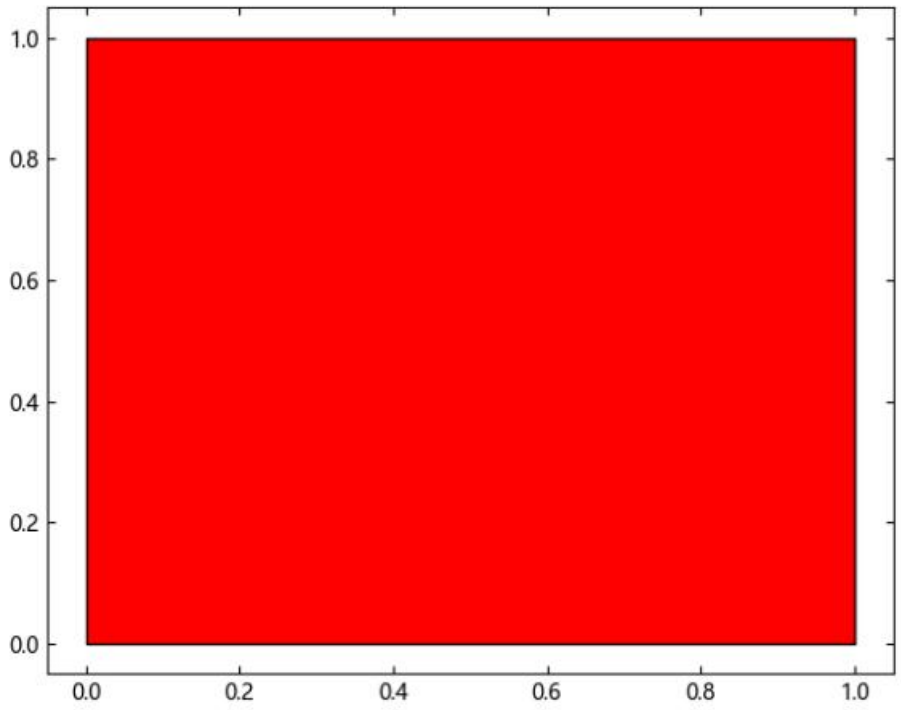
C. 示例

(1). 指定坐标

通过指定每个顶点的 (x,y) 坐标创建一个多边形。然后，继续将两个多边形添加到图窗中。

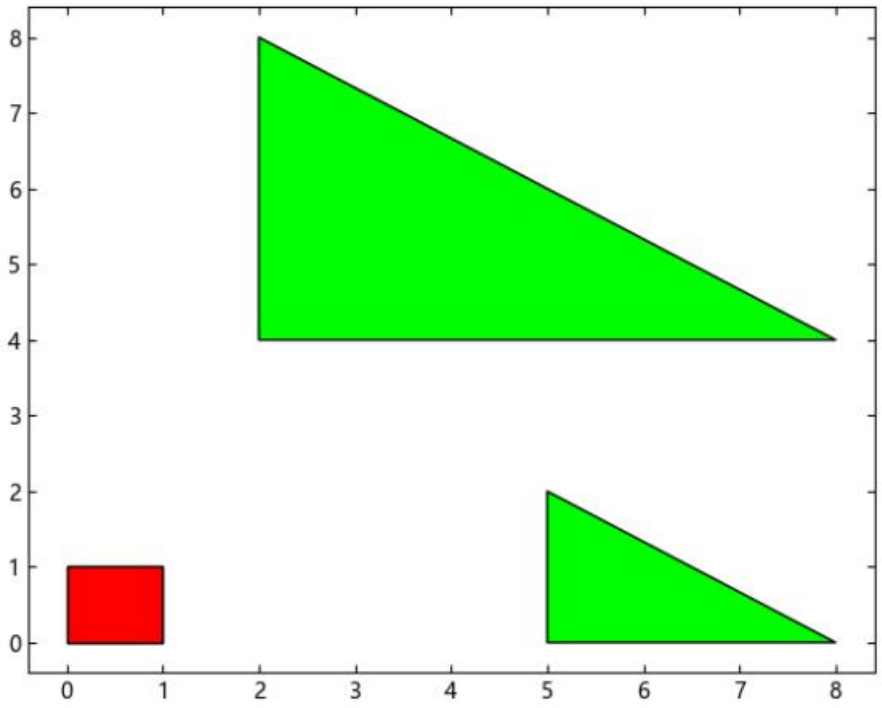
创建一个红色正方形，顶点位于 (0,0)、(1,0)、(1,1) 和 (0,1)。将 x 指定为顶点的 x 坐标，并将 y 指定为 y 坐标。patch 会自动将最后一个 (x,y) 坐标与第一个 (x,y) 坐标连接。

```
x = [0 1 1 0];
y = [0 0 1 1];
patch(x,y,"red")
```



通过将 x 和 y 指定为两列矩阵，创建两个多边形。每一列定义其中一个多边形的坐标。patch 将这些多边形添加到当前坐标区中，而不清除坐标区。

```
x2 = [2 5; 2 5; 8 8];
y2 = [4 0; 8 2; 4 0];
patch(x2,y2,[0 1 0])
```

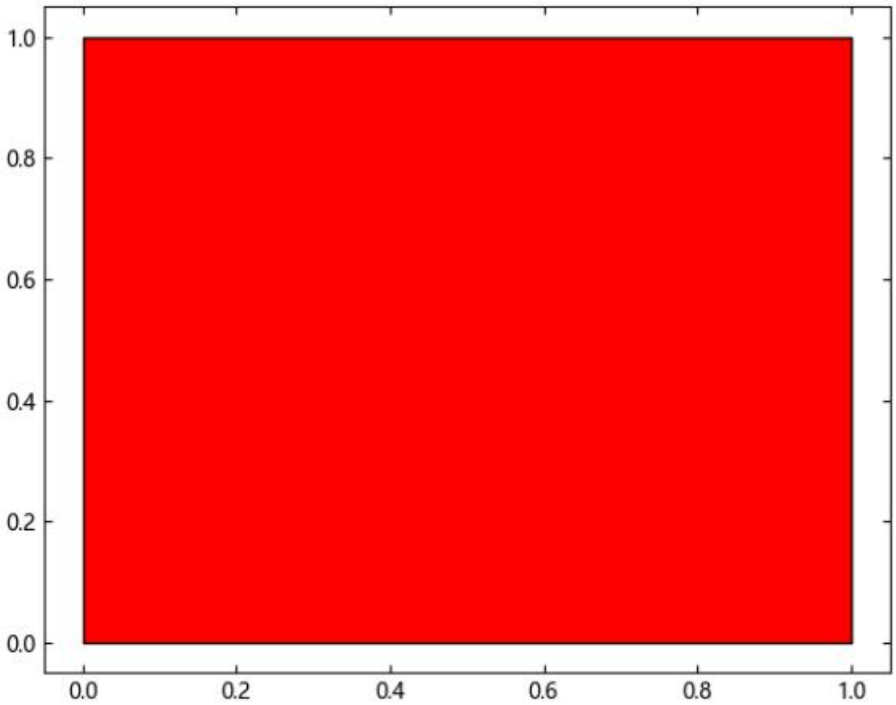


(2). 指定面和顶点

通过指定每个唯一顶点的坐标和定义如何连接这些顶点的矩阵，创建一个多边形。然后，继续将两个多边形添加到图窗中。

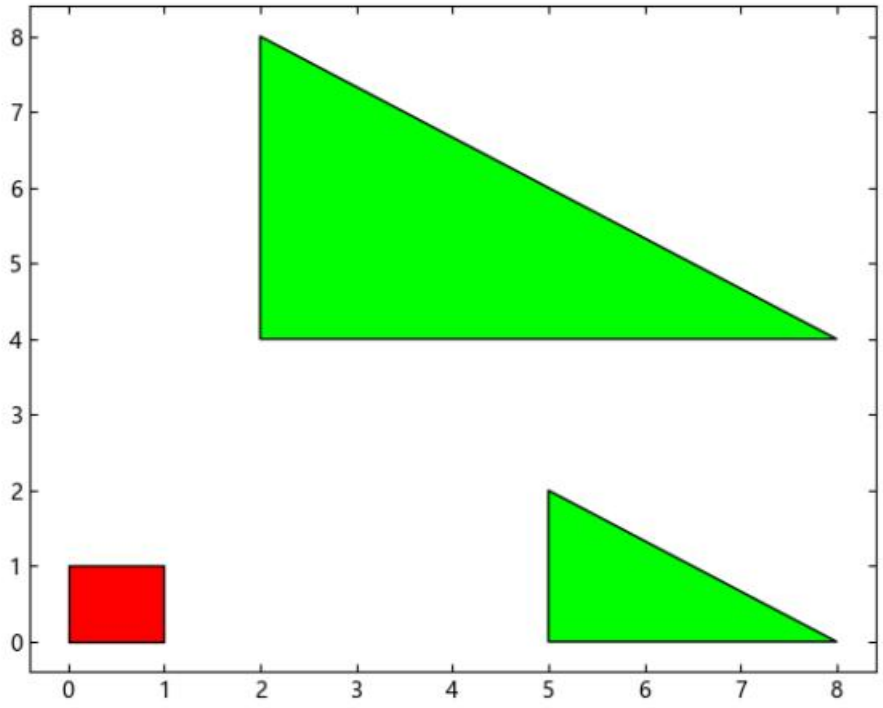
创建一个红色正方形，四个角分别位于 (0,0)、(1,0)、(1,1) 和 (0,1)。指定 v 以使每行定义一个顶点的 (x,y) 坐标。然后，将 f 指定为要连接的顶点。通过指定 facecolor 属性来设置颜色。

```
v = [0 0; 1 0; 1 1; 0 1];
f = [1 2 3 4];
patch(faces = f, vertices = v, facecolor = "red")
```



通过将 f 指定为两行矩阵来创建两个多边形。每一行定义一个补片的面。

```
v2 = [2 4; 2 8; 8 4; 5 0; 5 2; 8 0];
f2 = [1 2 3;
    4 5 6];
patch(faces = f2, vertices = v2, facecolor = "#00FF00")
```

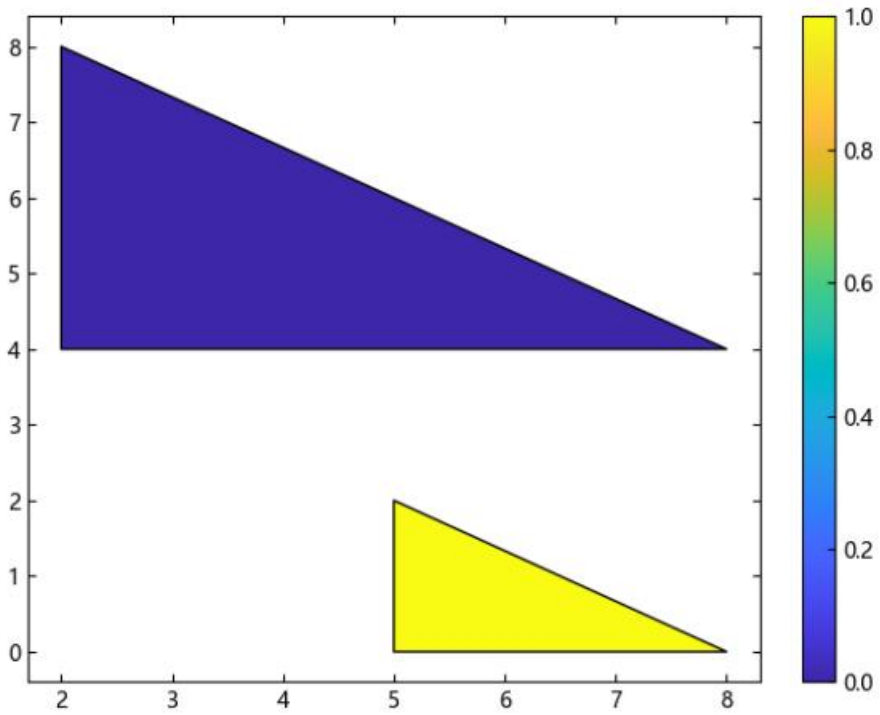


(3). 不同多边形面颜色

创建两个多边形，并为每个多边形面使用不同颜色。使用颜色栏以显示颜色映入颜色图的方式。

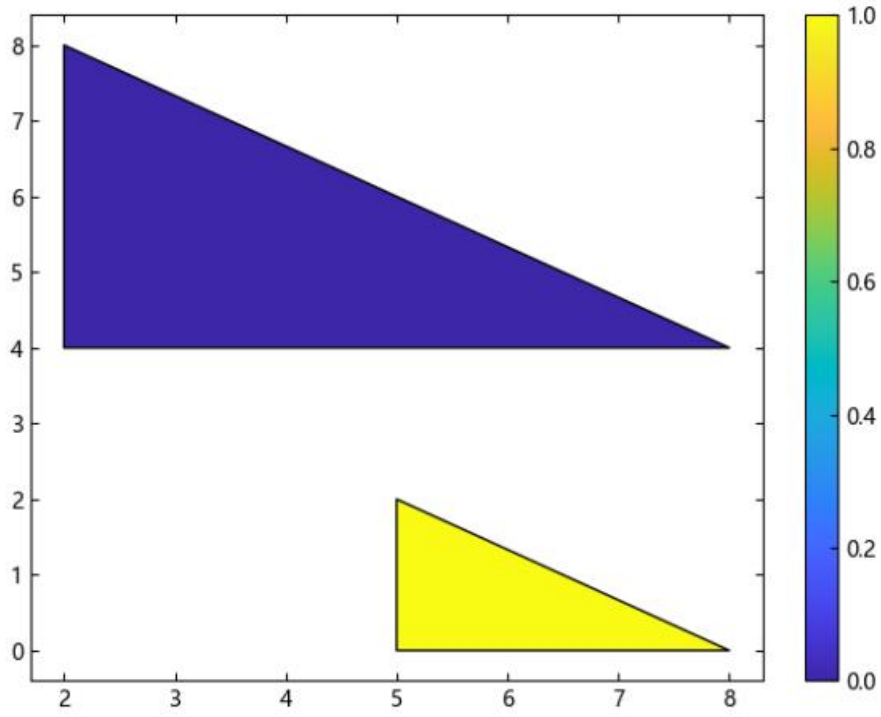
使用矩阵 x 和 y 创建多边形。将 c 指定为一个具有两个元素的列向量（因为有两个多边形面），并添加颜色栏。

```
x = [2 5; 2 5; 8 8];
y = [4 0; 8 2; 4 0];
c = [0; 1];
figure()
p = patch(x,y,c)
colorbar(gca(), p)
```



您也可以使用 f 和 v 得到相同的结果。当您创建多边形时，将 facevertexcdata 设置为一个具有两个元素的列向量（因为有两个多边形面）。将 facecolor 设置为 "flat"。

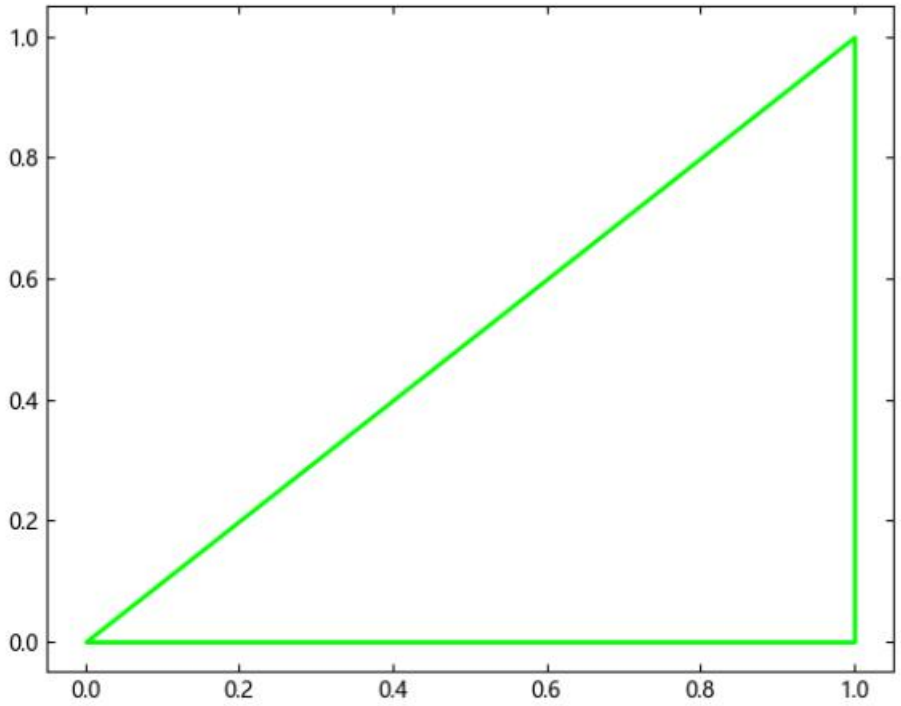
```
v = [2 4; 2 8; 8 4; 5 0; 5 2; 8 0];
f = [1 2 3; 4 5 6];
col = [0; 1];
figure()
p = patch(faces = f, vertices = v,facevertexcdata = col, facecolor
= "flat");
colorbar(gca(), p)
```



(4). 多边形边，不含面

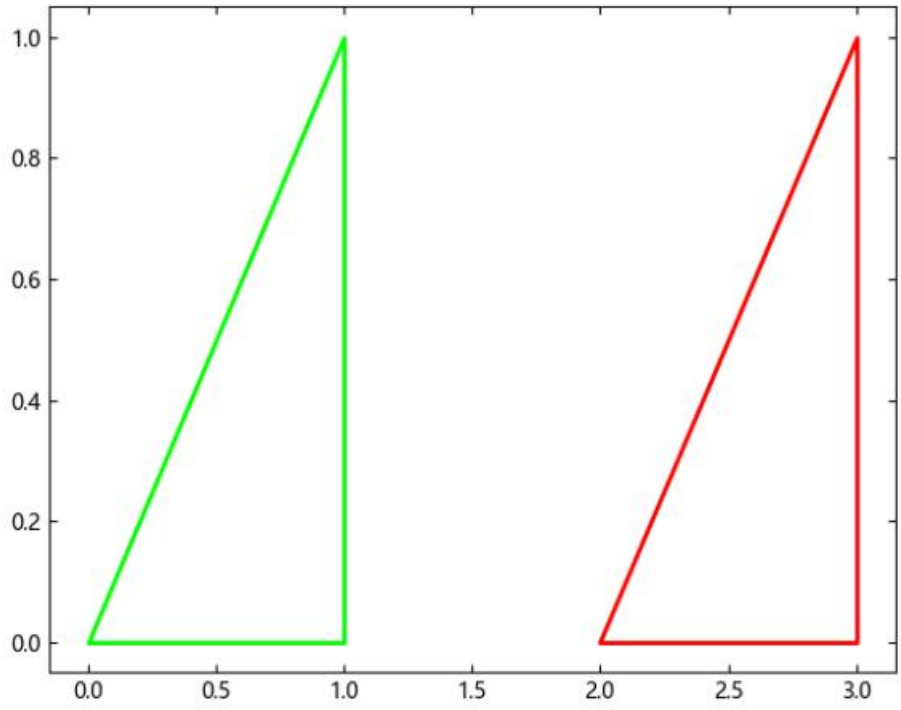
创建一个具有绿色边的多边形，且不显示面。然后，创建第二个多边形，其中每个边具有不同颜色。

```
v = [0 0; 1 0; 1 1];
f = [1 2 3];
figure()
patch(faces = f, vertices = v, edgecolor = "#00FF00", facecolor =
"none", linewidth = 2);
```



通过为每个顶点指定一种颜色并将 EdgeColor 设置为 "flat"。

```
v = [2 0; 3 0; 3 1];
f = [1 2 3];
c = [1 0 0;
    0 1 0;
    0 0 1];
    patch(faces = f, vertices = v, facevertexcdata = c,
    edgecolor = "flat", facecolor = "none", linewidth = 2);
```



(5). 使用结构体的多边形

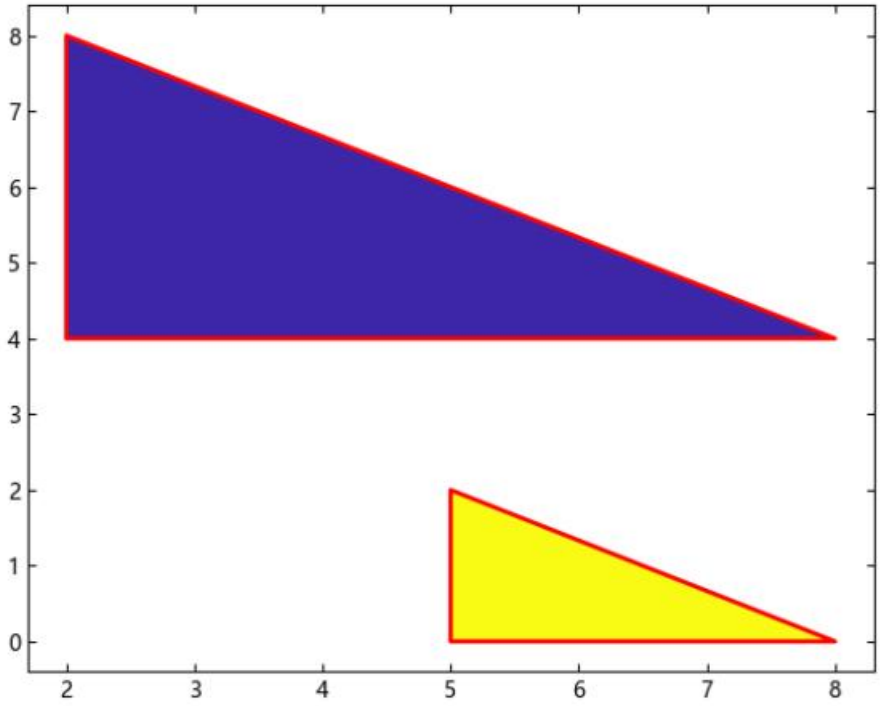
使用一个结构体创建两个多边形。首先，创建一个结构体，其中的字段名称与补片属性名称相匹配。然后，使用该结构体创建多边形。

```
mutable struct ST
    Vertices
    Faces
    FaceVertexCData
    FaceColor
    EdgeColor
    LineWidth
end

S = ST(nothing,nothing,nothing,nothing,nothing,nothing)

S.Vertices = [2 4; 2 8; 8 4; 5 0; 5 2; 8 0];
S.Faces = [1 2 3; 4 5 6];
S.FaceVertexCData = [0; 1];
S.FaceColor = "flat";
S.EdgeColor = "red";
S.LineWidth = 2;

figure
patch(S)
```

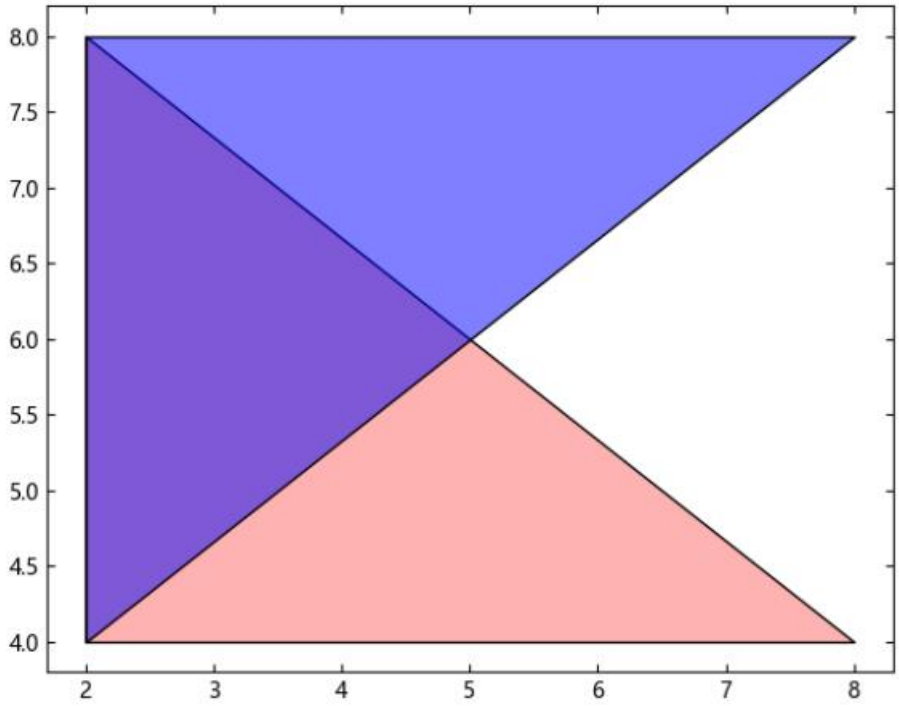


(6). 半透明的多边形

通过将 facecolor 属性设置为 0 和 1 之间的值，创建两个半透明的多边形。

```
v1 = [2 4; 2 8; 8 4];
f1 = [1 2 3];
figure
patch(faces = f1, vertices = v1, facecolor = "red", facealpha
= .3);

v2 = [2 4; 2 8; 8 8];
f2 = [1 2 3];
patch(faces = f2, vertices = v2, facecolor = "blue", facealpha
= .5);
```



(7). 创建多种颜色的线条

创建一个多种颜色的线条，其中每个顶点上都有标记。插入颜色，并使用颜色栏显示值映射到颜色图的方式。

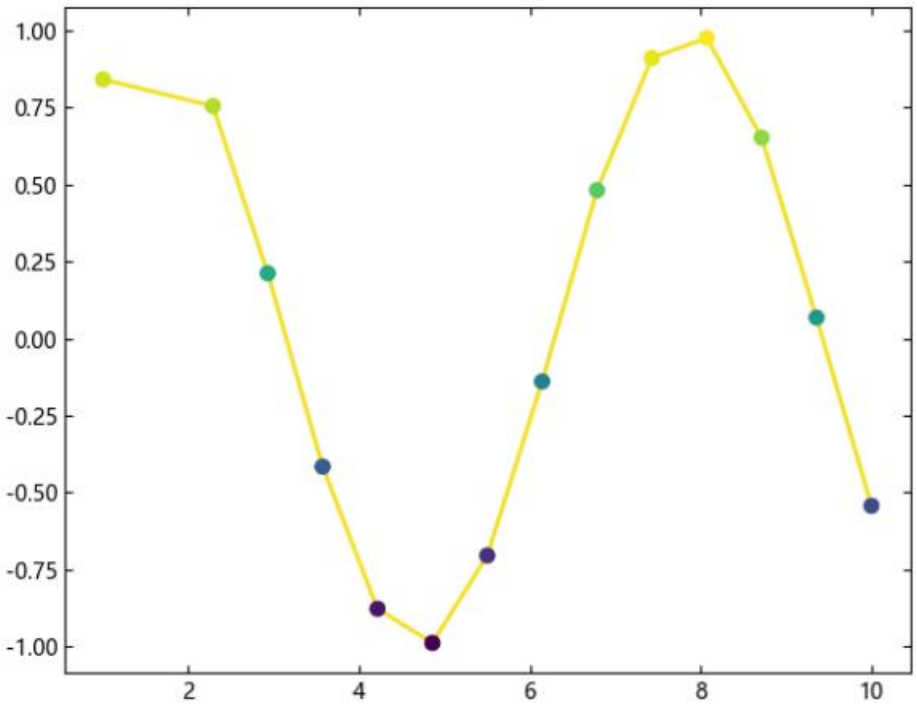
创建数据。将 y 的最后一项设置为 NaN，使 patch 创建一个线条，而非一个闭合多边形。使用 y 值为每个顶点定义一种颜色。c 中的值映射到颜色图中的颜色。

```
x = LinRange(1, 10, 15);
y = sin.(x);
y[2] = NaN;

f = (!isnan).(y[:])
x = x[f]
y = y[f]
c = y;
```

创建线条。在每个顶点上显示标记，并将 edgecolor 设置为 "flat" 以在顶点之间进行颜色插值。添加颜色栏。

```
figure()
p = patch(x, y, c, edgecolor="flat", linewidth=2, closed=false);
hold("on")
scatter(x,y,50,c,filled=true);
hold("off")
```



3.3.1.7 向量场

罗盘状图、羽状图、箭状图和流线图

函数名	简介
feather	绘制速度向量
quiver	箭头图或速度图
compass	绘制从原点发射出的箭头
quiver3	三维箭头图或速度图
streamline	根据二维或三维向量数据绘制流线图

3.3.1.7.1 feather

创建以 x 轴为起点的箭头

A. 语法

```
feather(U,V)
feather(Z)
```

```
feather(___,Fmt)
feather(ax,___)
f = feather(___)
```

B. 说明

feather(U,V)绘制以 x 轴为起点的箭头。使用笛卡尔分量 U 和 V 指定箭头方向，其中 U 表示 x 分量，V 表示 y 分量。第 n 个箭头的起始点位于 x 轴上的 n。箭头的数量与 U 和 V 中的元素数相匹配。

feather(Z)使用 Z 指定的复数值绘制箭头，实部表示 x 分量，虚部表示 y 分量。此语法等效于 feather(real(Z),imag(Z))。

feather(___,Fmt)设置箭头的线型、标记符号和颜色。

feather(ax,___)在指定坐标区（而不是当前坐标区）中绘制箭头。

f = feather(___) 返回由包含 length(U)+1 个元素的 Line 对象组成的向量。前 length(U) 个元素表示各个箭头，最后一个元素表示沿 x 轴的水平线。创建绘图后，使用这些 Line 对象来控制绘图的外观。

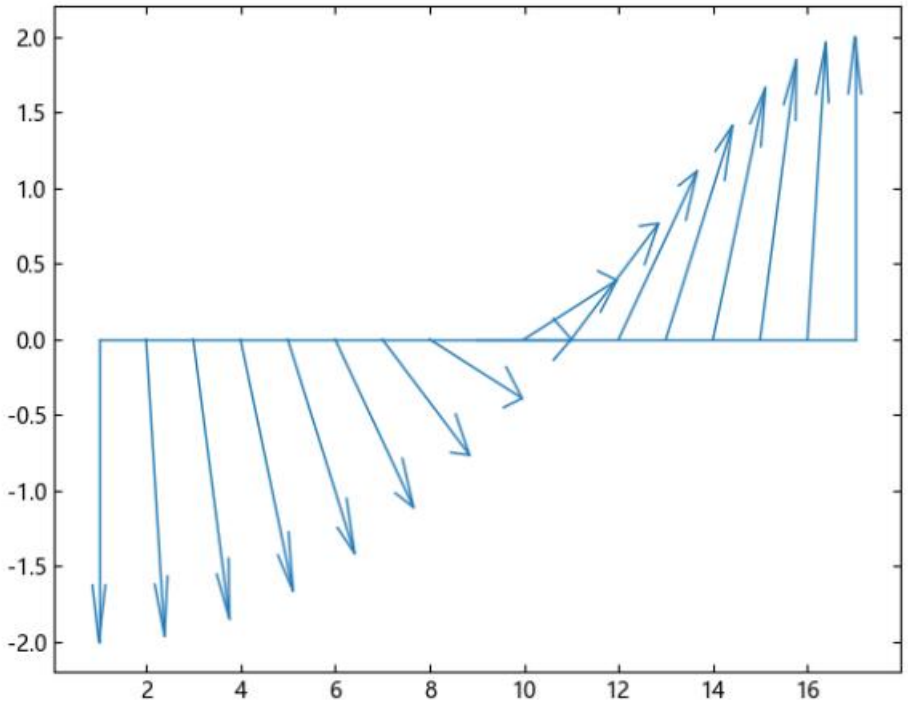
C. 示例

(1). 绘制羽状图

将 u, v 中的数据绘制成羽状图。

```
theta = -pi/2:pi/16:pi/2
r = 2 * ones(size(theta))
u, v = pol2cart(theta, r)

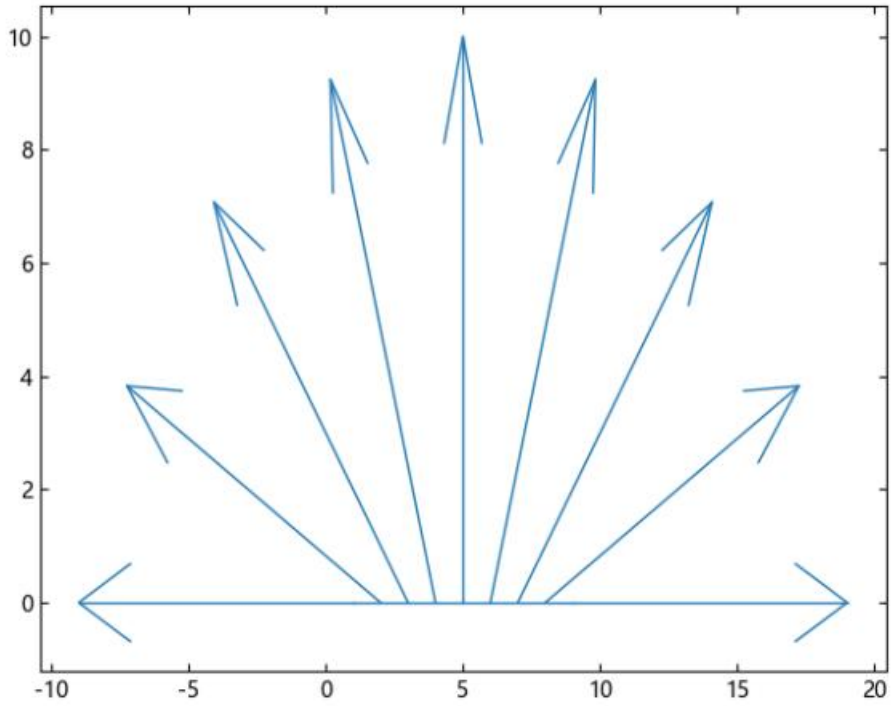
feather(u, v)
```



(2). 使用笛卡尔值创建羽状图

通过将每个箭头的分量指定为笛卡尔值来创建羽状图。第 n 个箭头以 x 轴上的 n 为起点。

```
t = (-pi / 2):(pi / 8):(pi / 2)
u = 10 * sin.(t);
v = 10 * cos.(t);
feather(u, v)
```



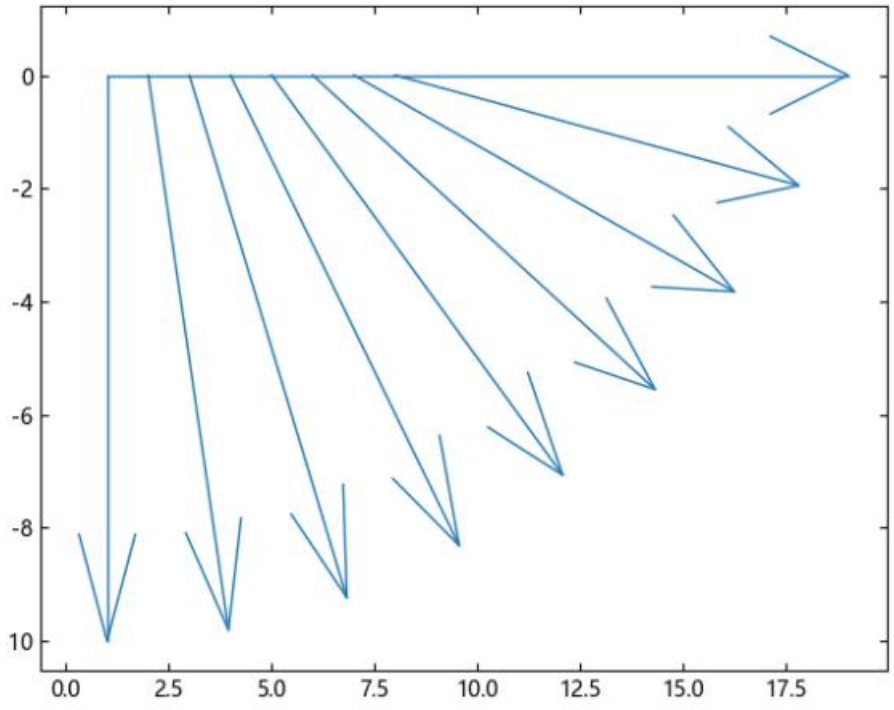
(3). 使用极坐标值创建羽状图

通过首先将极坐标转换为笛卡尔坐标，使用极坐标创建羽状图。

要完成此操作，请先用极坐标创建向量。使用 pol2cart 函数将它们转换为笛卡尔坐标。然后，创建绘图。

```
th = -pi/2:pi/16:0
r = 10 * ones(size(th))
u, v = pol2cart(th, r)

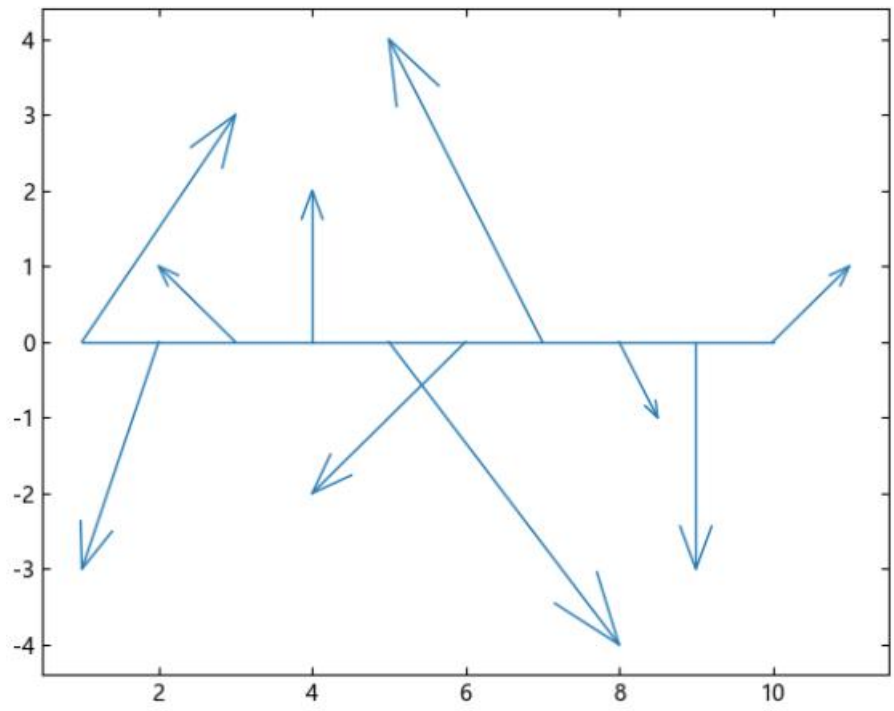
feather(u, v)
```



(4). 使用复数值创建羽状图

创建一个复数值向量。然后，使用羽状图显示这些复数值。实部确定每个箭头的 x 分量，虚部确定 y 分量。

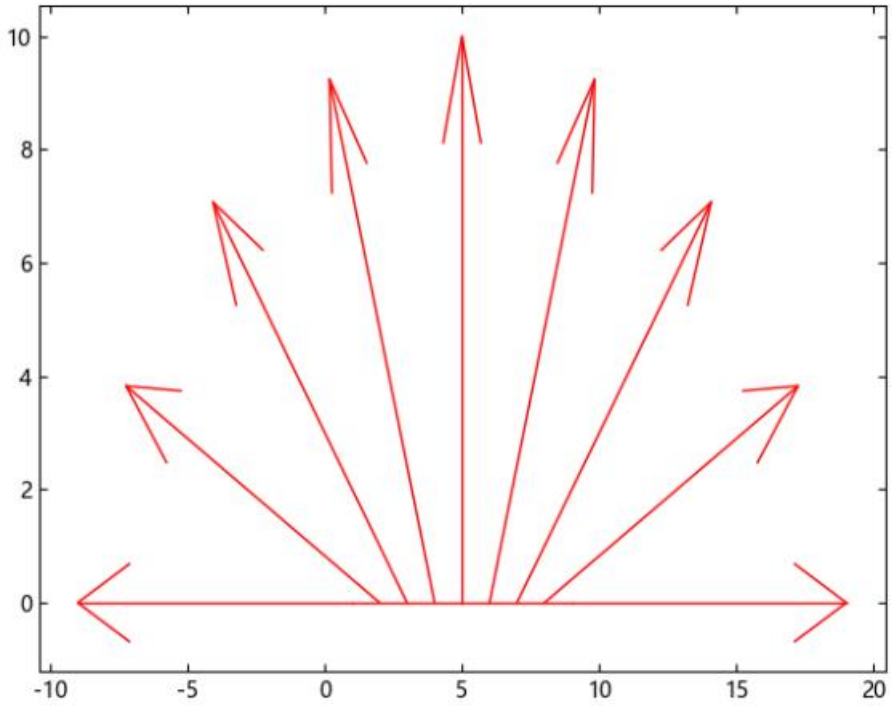
```
Z = [2 + 3im -1 - 3im -1 + im 2im 3 - 4im -2 - 2im -2 + 4im 0.5 -
im -3im 1 + im];
feather(Z)
```



(5). 指定羽状图的颜色

创建一个具有红色箭头的羽状图。

```
t = (-pi / 2):(pi / 8):(pi / 2)
u = 10 * sin.(t);
v = 10 * cos.(t);
f = feather(u, v, "r");
```



(6). 指定一个箭头的线宽和颜色

通过为单个箭头赋予变量，然后设置其属性来指定该箭头的线宽和颜色。

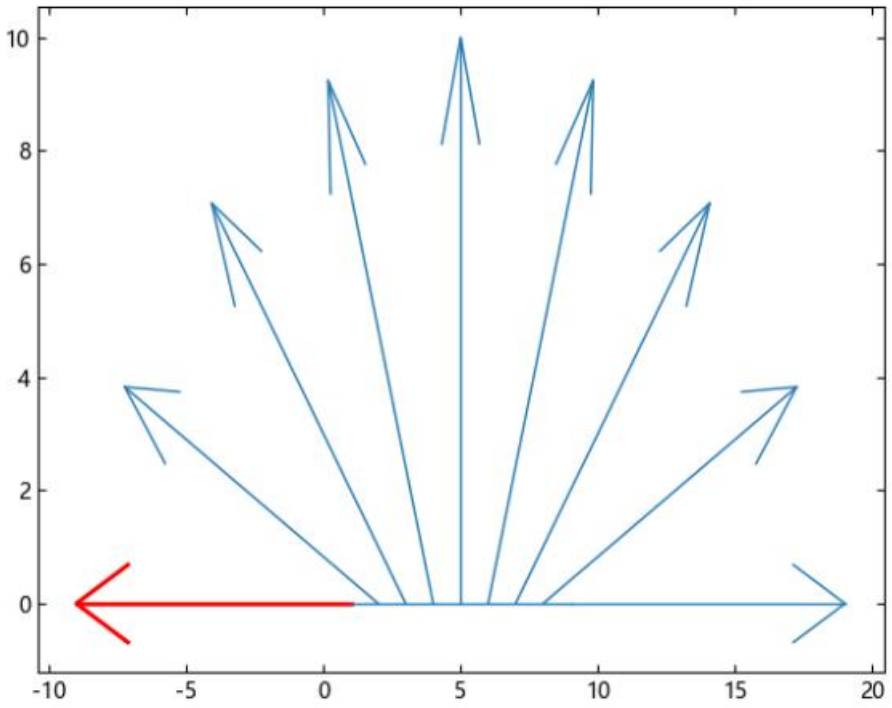
要完成此操作，请先创建一个羽状图并返回一个 Line 对象数组。

```
t = (-pi / 2):(pi / 8):(pi / 2)
u = 10 * sin.(t);
v = 10 * cos.(t);
f = feather(u, v)
```

为第一个箭头赋予一个变量。第一个箭头对应于 u 和 v 的第一组元素。

然后，更改线宽和颜色。

```
f1 = f[1];
f1.set_color("r")
f1.set_linewidth(2)
```



(7). 指定羽状图的坐标区

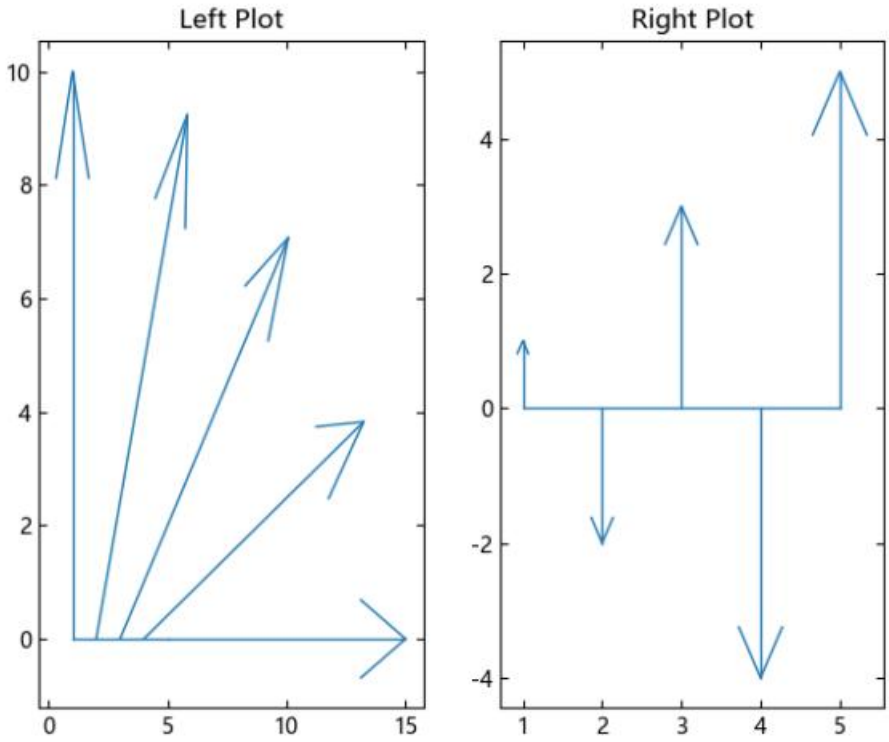
调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，以便坐标区填充布局中的可用空间。接下来，将该对象返回为 ax1。通过将 ax1 传递给 feather 函数来显示线性对数图。

```
ax1 = subplot(1, 2, 1)
ax2 = subplot(1, 2, 2)

t = 0:pi/8:pi/2;
u1 = 10 * sin.(t);
v1 = 10 * cos.(t);
feather(ax1, u1, v1)
title(ax1, "Left Plot")

u2 = zeros(5, 1);
v2 = [1, -2, 3, -4, 5];

feather(ax2, u2, v2)
title(ax2, "Right Plot")
```



3.3.1.7.2 quiver

箭头图或速度图

A. 语法

```
quiver(x,y,u,v)
quiver(u,v)
quiver(___,Fmt)
quiver(ax,___)
h = quiver(___)
```

B. 说明

箭头图将速度向量显示为箭头，其中分量 (u, v) 位于点 (x, y) 处。

例如，第一个向量由分量 u(1), v(1) 定义并显示在点 x(1), y(1) 处。

quiver(x, y, u, v) 在 x 和 y 中每个对应元素对组所指定的坐标处将向量绘制为箭头。矩阵 x、y、u 和 v 必须大小相同并包含对应的位置和速度分量。但是，如下节所述，x 和 y 还可以是向量。默认情况下，箭头缩放到刚好不重叠，但您可以根据需要将箭头缩放的长一些或短一些。

quiver(u, v) 在 x-y 平面的等距点处绘制 u 和 v 指定的向量。

quiver(___, Fmt) 使用任何有效的 Fmt 指定线型、标记符号和颜色。quiver 在向量原点处绘制标记。

quiver(ax, ___) 将图形绘制到 ax 坐标区中，而不是当前坐标区 (gca) 中。

h = quiver(___) 返回 Quiver 对象。

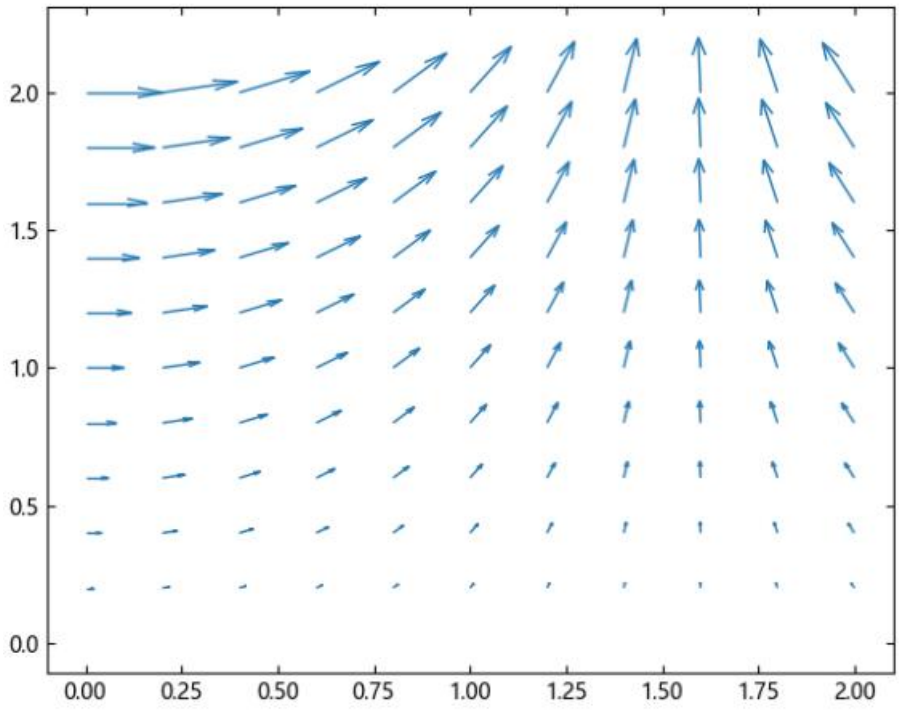
C. 示例

(1). 绘制速度向量

使用 quiver 在 x 和 y 的每个数据点处显示箭头，这样箭头方向和长度分别由 u 和 v 中的相应值表示。

```
x, y = meshgrid2(0:0.2:2, 0:0.2:2);
u = cos.(x) .* y;
v = sin.(x) .* y;

figure()
q = quiver(x, y, u, v)
```



(2). 显示箭头图的梯度

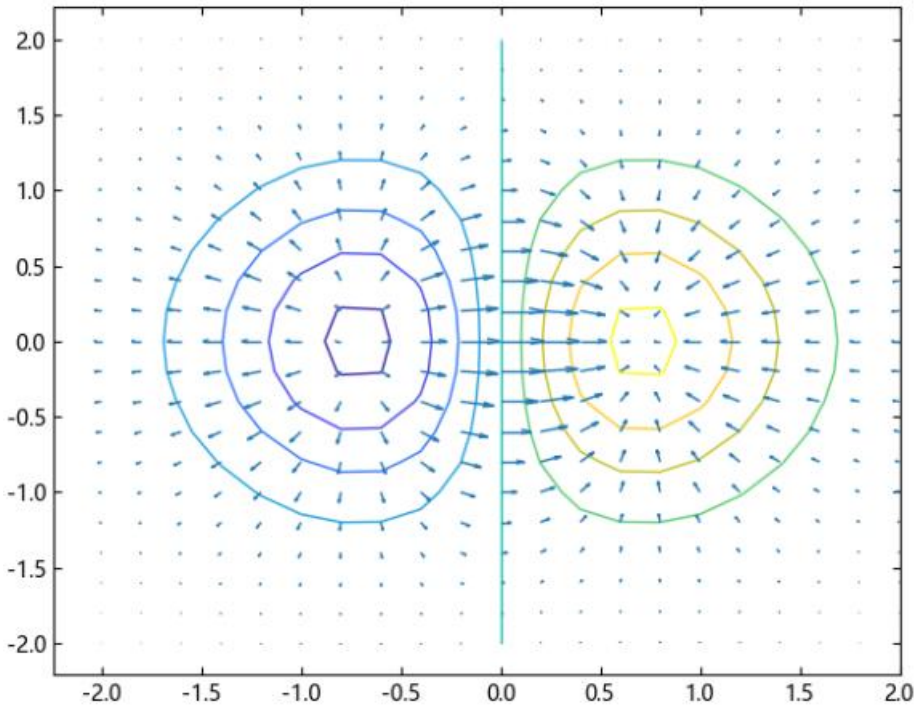
绘制函数 $z = xe^{-x^2-y^2}$ 的梯度。

```
import TyPythonCall: @pyimport
@pyimport numpy as np
X, Y = meshgrid2(-2:0.2:2, -2:0.2:2);
```

```
Z = X .* exp.(-X .^ 2 - Y .^ 2);
DY, DX = np.gradient(Z, 0.2, 0.2);

contour(X, Y, Z)

hold("on")
quiver(X, Y, DX, DY; length=0.23)
hold("off")
```



3.3.1.7.3 compass

从原点发射出的箭头

A. 语法

```
compass(U,V)
compass(Z)
compass(___,Fmt)
compass(ax,___)
c = compass(___)
```

B. 说明

罗盘图显示包含分量 (U,V) 的向量，就像箭头从原点射出一样。U、V 和 Z 位于笛卡尔坐标中，绘制于一个环状网格上。

compass(U,V) 显示具有 n 个箭头的罗盘图，其中 n 是 U 或 V 中的元素数目。每个箭头的基点的位置为原点。每个箭头的尖端的位置是相对于基点的一个点，并由 [U(i),V(i)] 确定。

compass(Z) 显示具有 n 个箭头的罗盘图，其中 n 是 Z 中的元素数目。每个箭头的基点的位置为原点。每个箭头的尖端的位置相对于基点，由 Z 的实部和虚部确定。此语法等效于 compass(real(Z),imag(Z))。

compass(___,Fmt) 使用 Fmt 指定的线型、标记符号和颜色来绘制罗盘图。

compass(ax,___) 将图形绘制到带有句柄 ax 的坐标区中，而不是当前坐标区 (gca) 中。

h = compass(___) 返回线条对象的句柄。

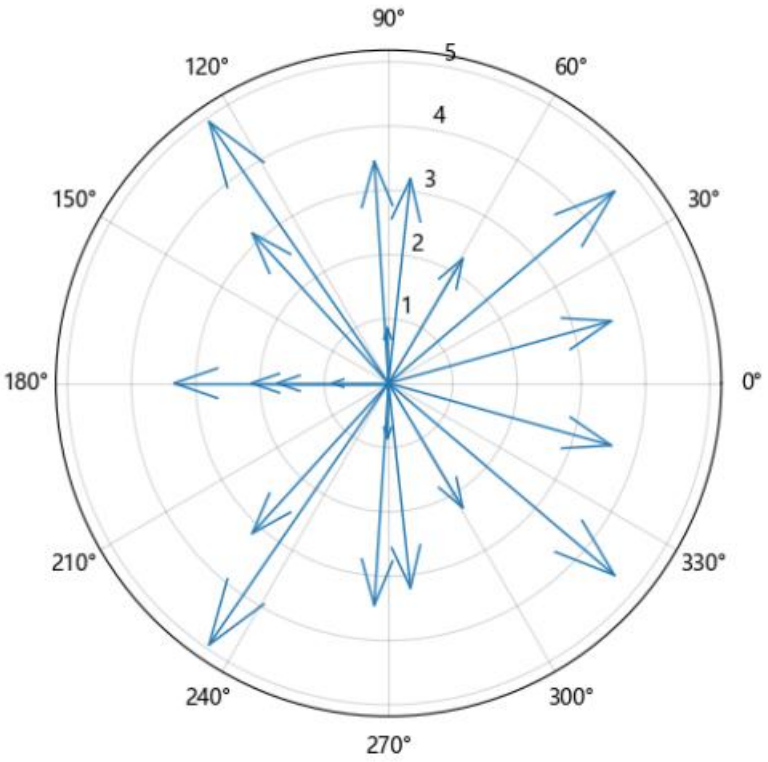
C. 示例

(1). 创建罗盘图

创建一个由随机矩阵的特征值构成的罗盘图。

```
M = randn(20, 20);
Z = eigvals(M);

figure()
c = compass(Z)
```



3.3.1.7.4 quiver3

三维箭头图或速度图

A. 语法

```
quiver3(x,y,z,u,v,w)
quiver3(z,u,v,w)
quiver3(___,Fmt)
quiver3(ax,___)
h = quiver3(___)
```

B. 说明

三维箭头图显示分量 (u, v, w) 位于点 (x, y, z) 处的向量，其中，u、v、w、x、y 和 z 都有实数（非复数）值。

quiver3(x, y, z, u, v, w) 在 (x, y, z) 确定的点处绘制向量，其方向由分量 (u, v, w) 确定。矩阵 x、y、z、u、v 和 w 必须具有相同大小并包含对应的位置和向量分量。

quiver3(z, u, v, w) 在沿曲面 z 的等间距点处绘制向量，其方向由分量 (u, v, w) 确定。对于每个向量 (u(i, j),v(i, j),w(i, j))，列索引 j 确定曲面上点的 x 值，i 确定 y 值，z(i, j) 确定 z 值。即，quiver3 将向量定位在曲面上的点 (j, i, z(i,

j)) 处。quiver3 函数基于向量之间的距离自动缩放向量以避免它们重叠在一起。

quiver3(___,Fmt) 使用任何有效的 Fmt 指定线型、标记符号和颜色。quiver3 在向量原点处绘制标记。

quiver3(ax, ___) 将图形绘制到 ax 坐标区中，而不是当前坐标区 (gca) 中。

h = quiver3(___) 返回 Quiver 对象。

C. 示例

(1). 创建三维箭头图

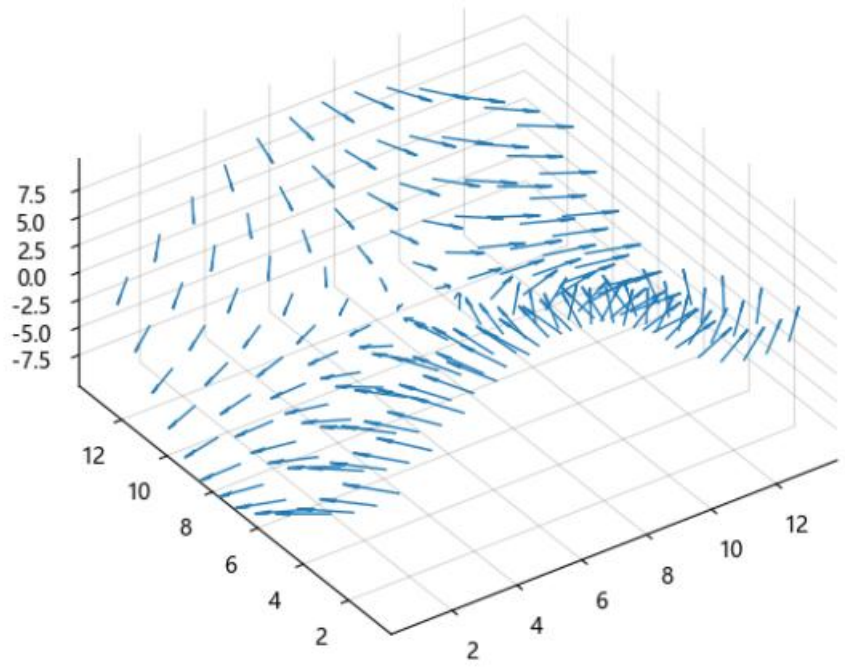
定义该数据。

```
using TyPlot
using TyMath
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/quiver3/0-data.jl"
include(source_path)

x = -3:0.5:3;
y = -3:0.5:3;
X, Y = meshgrid2(x, y);
Z = Y .^ 2 - X .^ 2;
```

绘制向量，其中包含在 x 方向和 y 方向上等间距的点处的分量 (U,V,W)，并且具有由 Z 确定的高度。

```
figure()
quiver3(Z, U, V, W, length = 1.5)
plt_view(-35, 45)
```



(2). 绘制曲面图法线

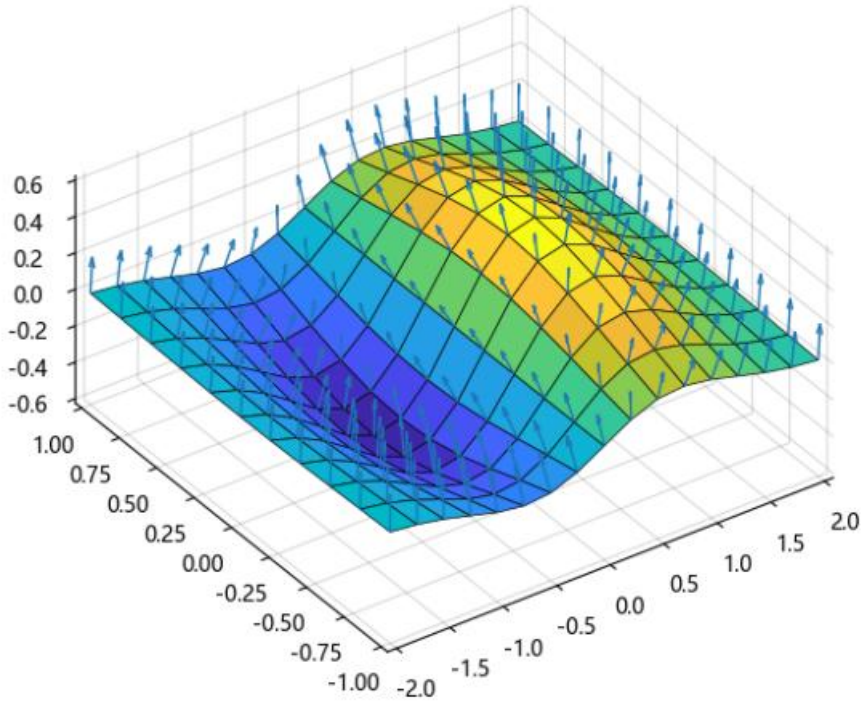
绘制函数$z = xe^{-x^2-y^2}$的曲面图法线。

```
using TyPlot
using TyMath
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/quiver3/1-data.jl"

include(source_path)

X, Y = meshgrid2(-2:0.25:2, -1:0.2:1);
Z = X .* exp.(-X .^ 2 - Y .^ 2);
figure()
quiver3(X, Y, Z, U, V, W, length = 0.2)

hold("on")
surf(X, Y, Z)
plt_view(-35, 45)
axis([-2 2 -1 1 -.6 0.6])
hold("off")
```



3.3.1.7.5 streamline

根据二维或三维向量数据绘制流线图

A. 语法

```
streamline(X,Y,U,V,startx,starty)
streamline(U,V,startx,starty)
streamline(ax,___)
h = streamline(___)
```

B. 说明

streamline(X,Y,U,V,startx,starty) 根据二维向量数据 U、V 绘制流线图。

数组 X 和 Y 用于定义 U 和 V 的坐标，它们必须是单调的，无需间距均匀。X 和 Y 必须具有相同数量的元素，就像由 meshgrid 生成一样。startx 和 starty 定义流线图的起始位置。输出参数 h 包含一个线条句柄向量，每个流线图一个句柄。

streamline(U,V,startx,starty) 假定数组 X 和 Y 定义为 [X,Y] = meshgrid(1:N,1:M)，其中 [M,N] = size(U)。

streamline(ax,___) 将图形绘制到句柄为 axes_handle 的坐标区对象中，而不是当前坐标区对象 (gca) 中。

h = streamline(___) 返回一个线条句柄向量，每个流线图一个句柄。

C. 示例

(1). 绘制流线图

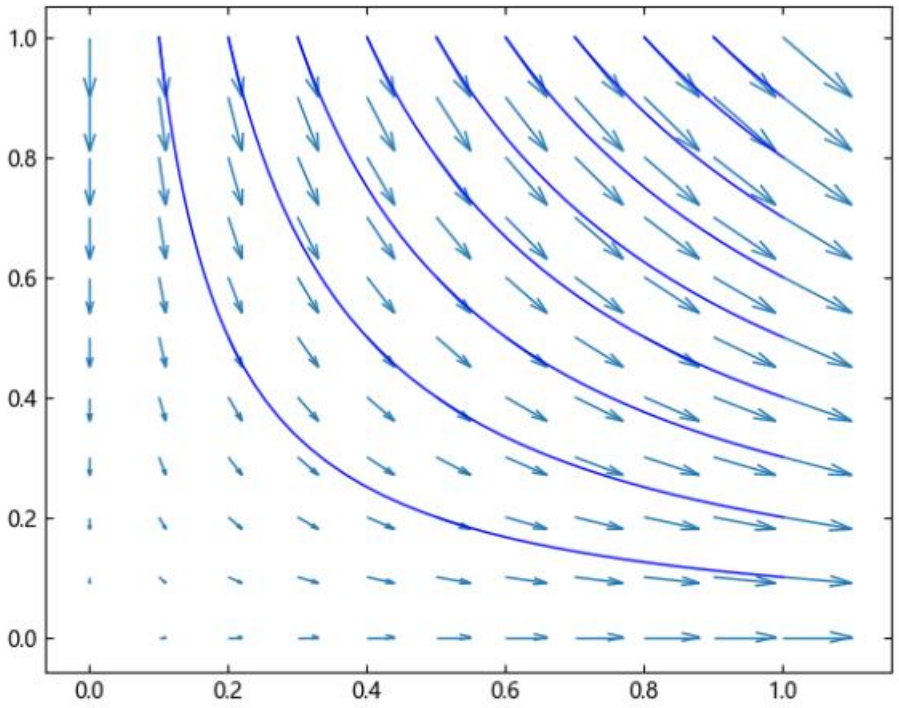
定义该数据。

```
x, y = meshgrid2(0:0.1:1, 0:0.1:1);
u = x;
v = -y;
```

创建数据的箭头图。绘制沿线条 y=1 上的不同点开始的流线图。

```
figure()
quiver(x, y, u, v)

startx = 0.1:0.1:1;
starty = ones(size(startx));
hold("on");
streamline(x, y, u, v, startx, starty);
hold("off");
```



3.3.2 格式和注释

添加标签、调整颜色、定义坐标轴范围、应用光照或透明度、设置照相机视图

自定义绘图的外观，以传递其他信息或增强数据的显示。例如，您可以添加标题和标签、更改坐标轴范围或添加网格线。此外，也可以通过在一个坐标区中显示所有数据或在单幅图窗中使用多个坐标区，将多组数据合并在一幅图窗中。

3.3.2.1 标签和注释

添加标题、轴标签、信息性文本以及其他图表注释

添加标题、坐标轴标签或向图添加注释以帮助传达重要信息。可以创建图例来标记绘制的数据系列，或在数据点旁边添加描述性文本。此外，还可以创建注释，例如突出显示特定区域数据的矩形、椭圆形、箭头、垂直线或水平线。

3.3.2.1.1 标签

函数名	简介
title	添加标题
sgtitle	在子图网格上添加标题
xlabel	为 x 轴添加标签
ylabel	为 y 轴添加标签
zlabel	为 z 轴添加标签
legend	在坐标区上添加图例

title

添加标题

A. 语法

```
title(txt)
title(target,txt)
```

```
title(___,Key=Value)
t = title(___)
```

B. 说明

title(txt)将指定的标题添加到 gca 命令返回的坐标区或图中。重新发出 title 命令可使新标题替换旧标题。

title(target，txt)将标题添加到指定的目标对象。

title(___,Key=Value)使用一个或多个名称-值对组参数修改标题外观。例如，fontsize=12 将字体大小设置为 12 磅。在所有其他输入参数之后指定名称-值对组参数。并非所有图类型均支持修改标题外观。

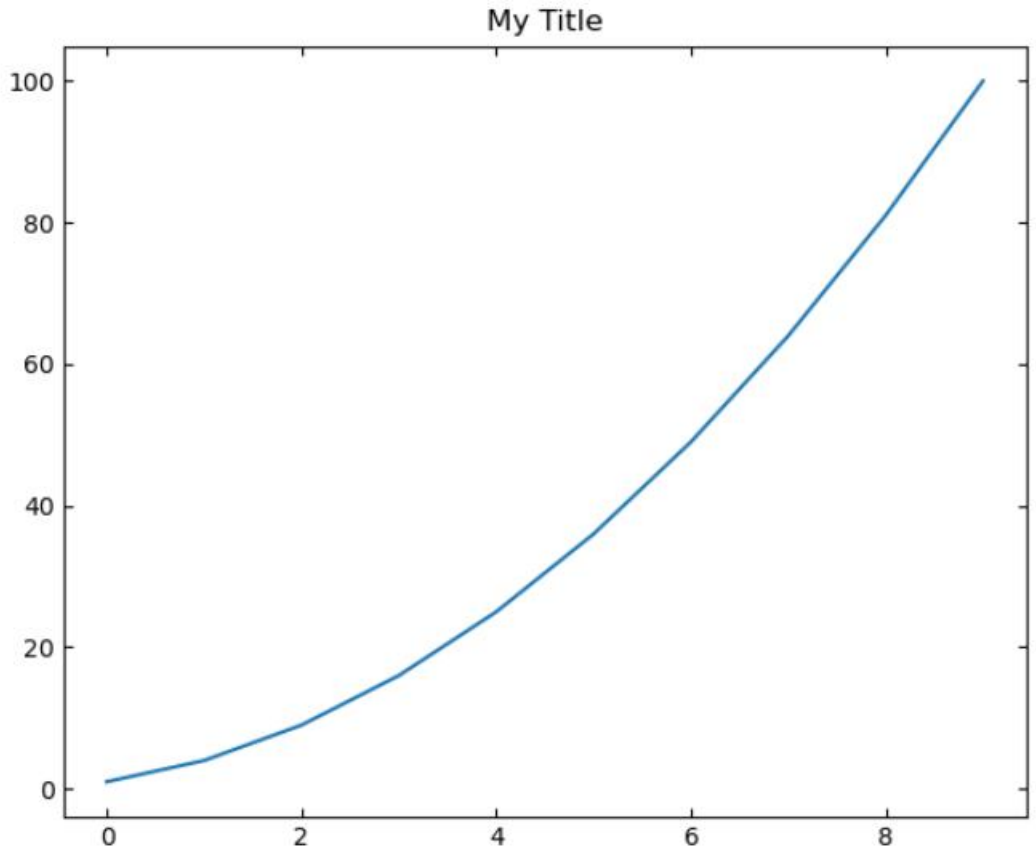
t = title(___) 返回用于标题的对象。以后可以使用 t 修改标题。

C. 示例

(1). 在当前坐标区上添加标题

在当前坐标区中创建一个图窗并显示标题。

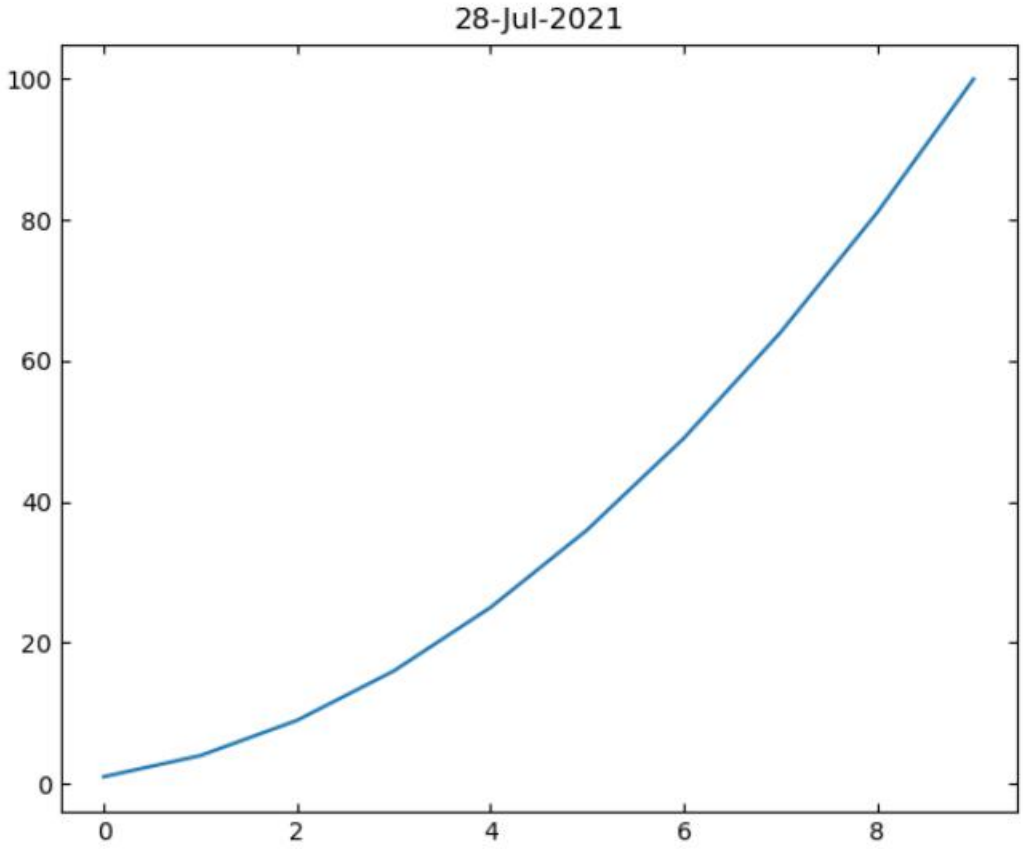
```
plot((1:10).^2)
title("My Title")
```



您也可以通过可返回文本的函数来调用 title。例如，Dates 函数返回包含今

日日期的文本。

```
using Dates
dt = Dates.format(today(),"dd-uuu-yyyy")
title(dt)
```

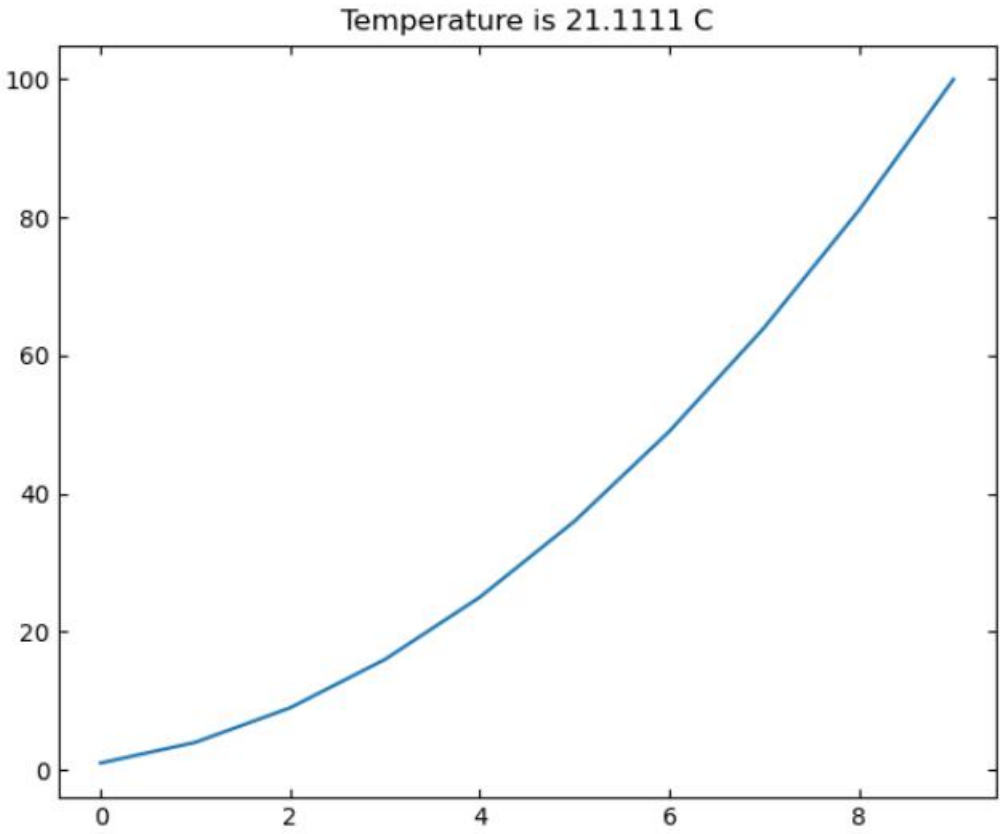


Syslab 将 Dates 的输出设置为坐标区的标题。

(2). 在标题中包含变量的值

在标题中包含变量 c 的值。

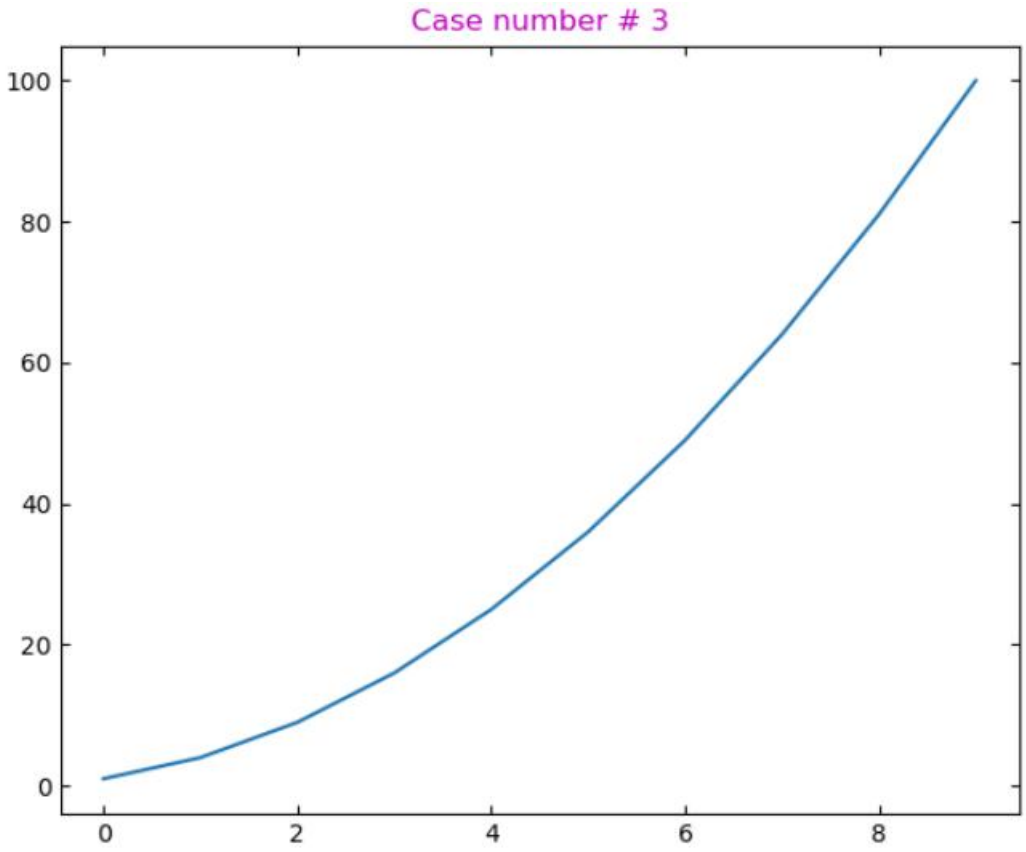
```
plot((1:10).^2)
f = 70;
c = (f-32)/1.8;
using Printf
str_c = @sprintf("%0.4f", c)
title("Temperature is "*str_c*" C")
```



(3). 使用 Key=Value 对组参数创建彩色标题

使用 Key=Value 对组 color="m" 将标题的颜色设置为品红色。

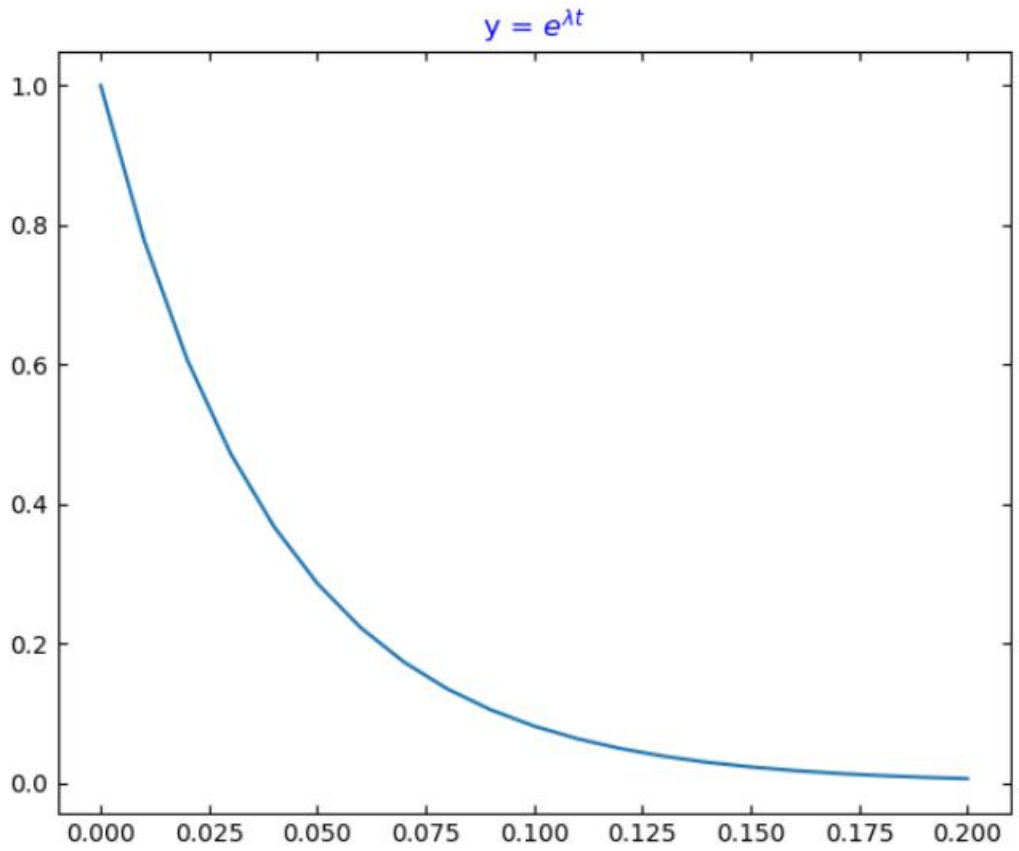
```
plot((1:10).^2)
title("Case number # 3",color="m")
```



(4). 在标题中包含希腊符号

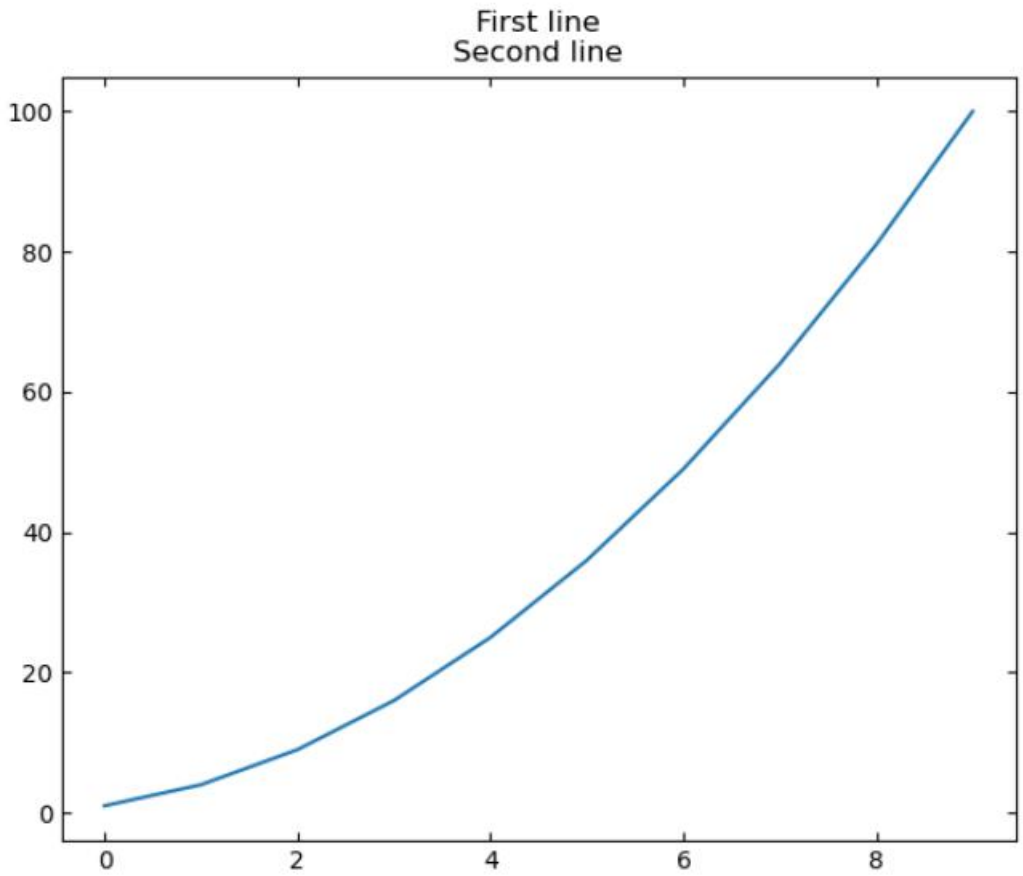
使用 TeX 标记在标题中包含希腊符号。

```
t = (0:0.01:0.2);
y = exp.(-25*t);
plot(t,y)
title(raw"y = $\ite^{\lambda t}$",color="b")
```



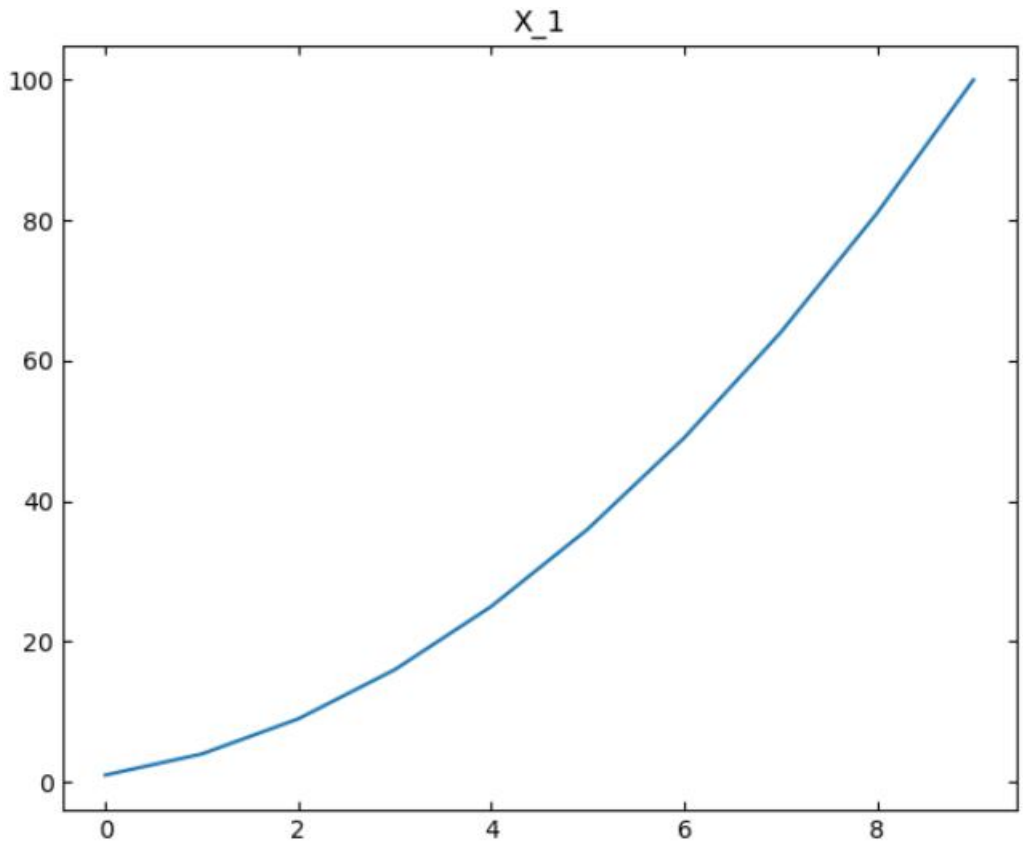
(5). 创建多行标题

```
plot((1:10).^2)
title("First line\nSecond line")
```



(6). 按键入原样显示文本

```
plot((1:10).^2)
title("X_1")
```

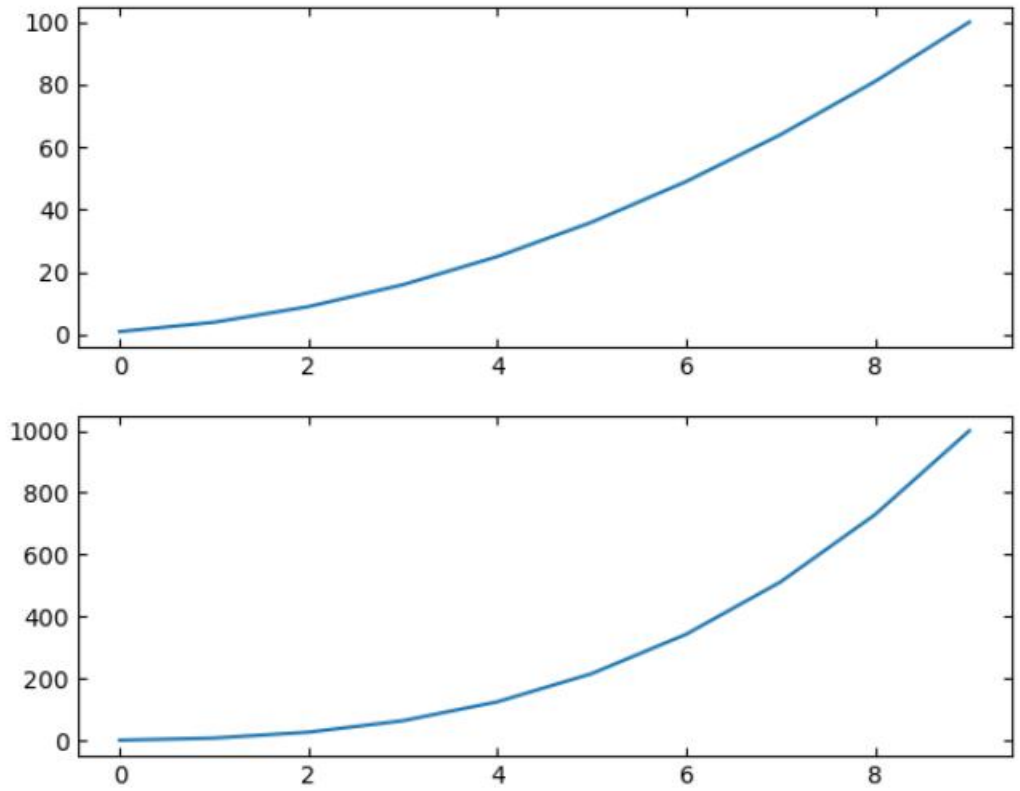


(7). 在特定的坐标区上添加标题

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。通过将 ax1 传递给 plot 函数来创建顶部绘图。然后将数据绘制到每个坐标区中。

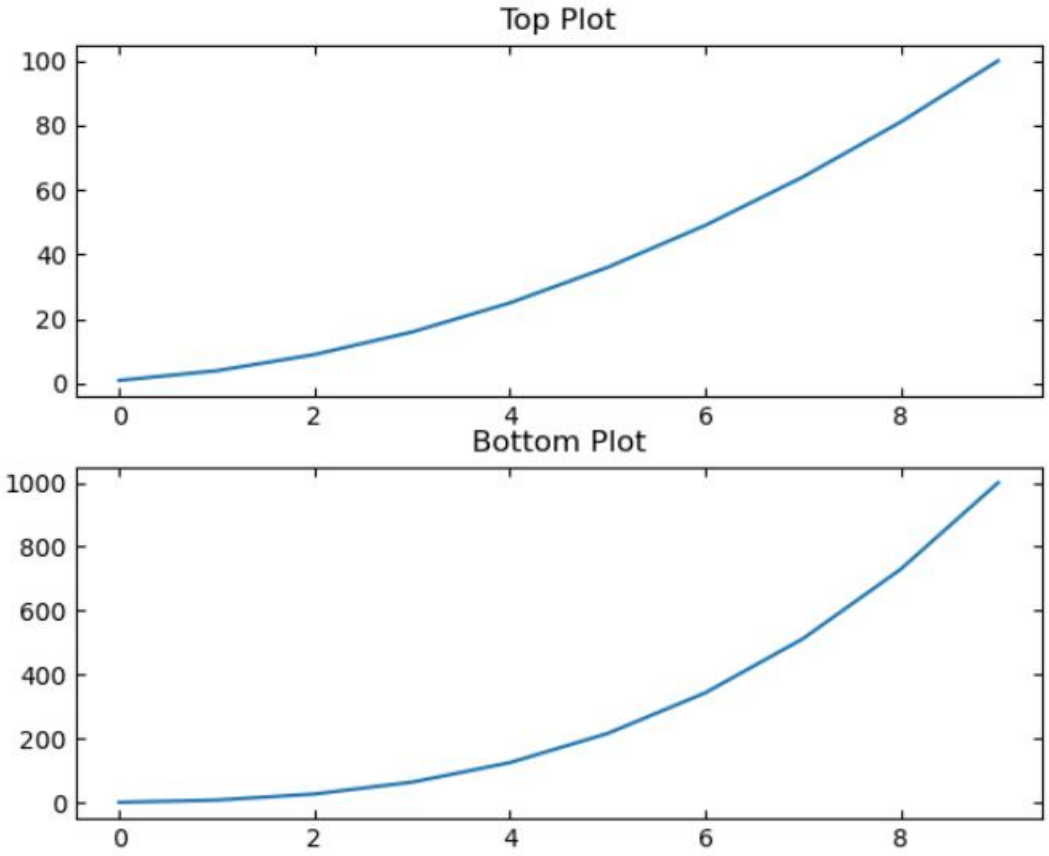
```
subplot(2,1,1)
ax1 = subplot(2,1,1)
ax2 = subplot(2,1,2)

plot(ax1,(1:10).^2)
plot(ax2,(1:10).^3)
```



通过将 ax1 和 ax2 传递给 title 函数，为每个坐标区添加标题。

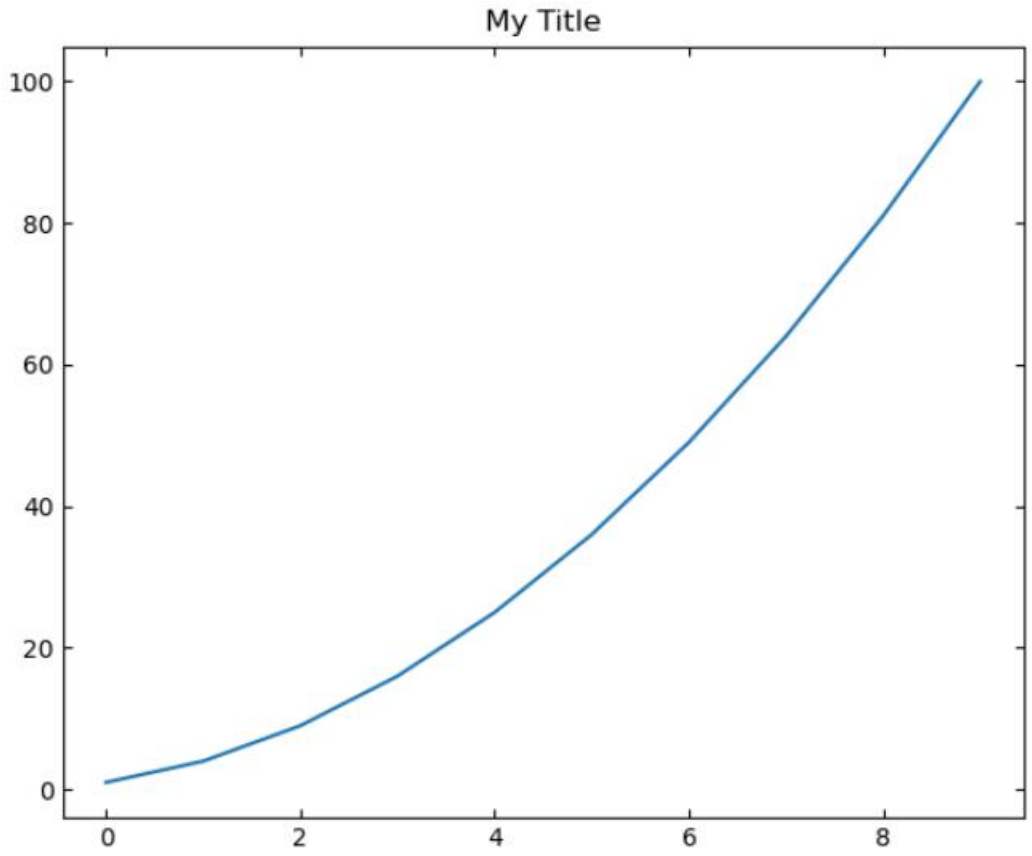
```
title(ax1,"Top Plot")
title(ax2,"Bottom Plot")
```



(8). 添加标题并返回文本句柄

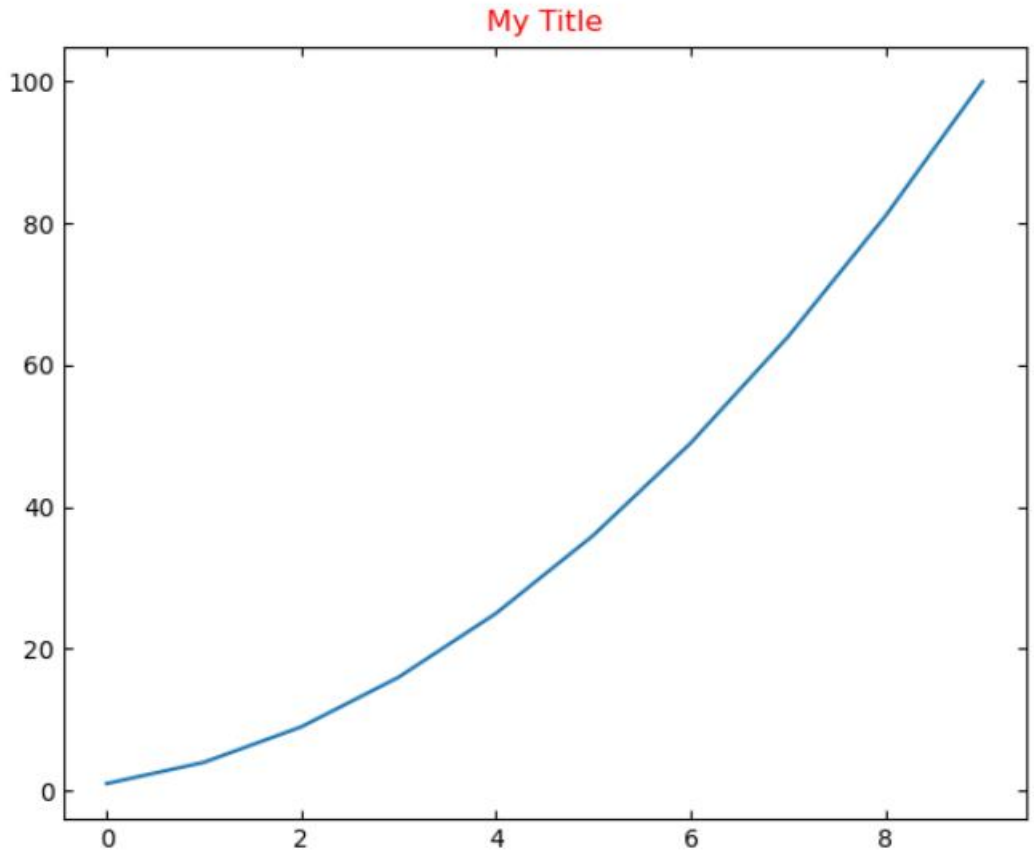
在绘图上添加标题，并返回文本对象。

```
plot((1:10).^2)
t = title("My Title");
```



将标题的颜色设置为红色。

```
t.set_color("red");
```



sgtitle

在子图网格上添加标题

A. 语法

```
sgtitle(txt)
sgtitle(target,txt)
sgtitle(___,Key=Value)
sgt = sgtitle(___)
```

B. 说明

sgtitle(txt)在当前图窗中子图的网格上方添加标题。如果图窗不存在，则此命令会创建一个图窗。

sgtitle(target,txt) 将标题添加到指定的图窗、面板或选项卡（而不是当前图窗）中的子图网格。

sgtitle(___,Key=Value) 使用一个或多个名称-值对组参数修改文本属性。例如，fontsize=12 指定 12 磅字体。有关属性列表，请参阅 Text 属性 (opens new

window)。在所有其他输入参数之后指定名称-值对组参数。

sgt = sgtitle(___) 返回用于创建标题的子图 Text 对象。

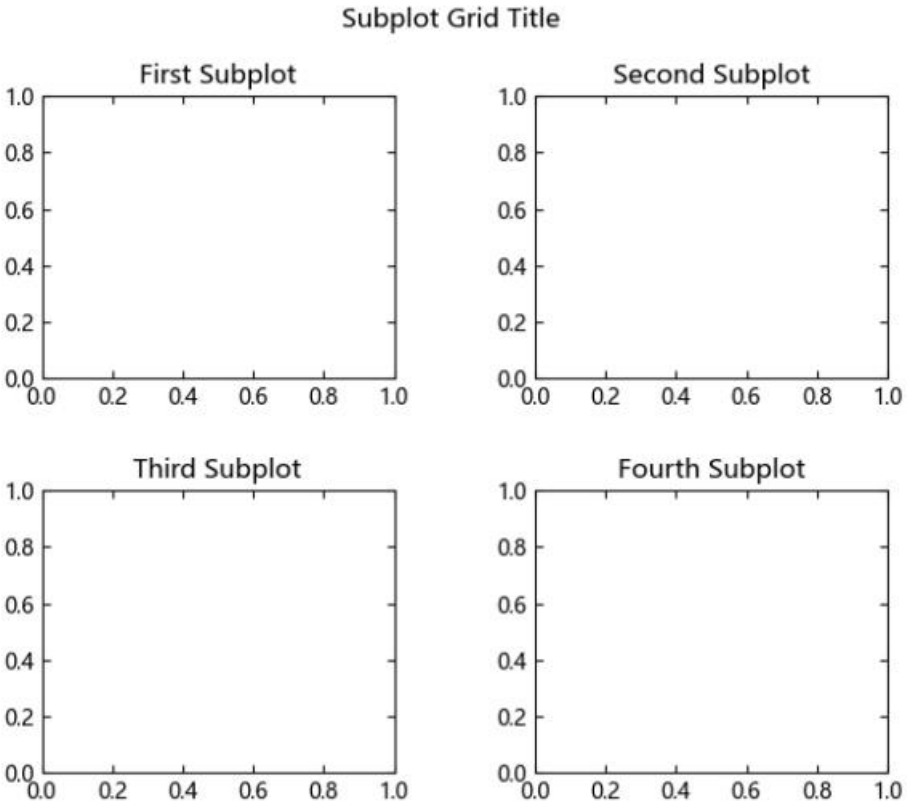
C. 示例

(1). 在子图网格上添加标题

创建带有四个子图的图窗。为每个子图添加标题，然后将总标题添加到子图网格中。

```
subplot(2,2,1)
title("First Subplot")
subplot(2,2,2)
title("Second Subplot")
subplot(2,2,3)
title("Third Subplot")
subplot(2,2,4)
title("Fourth Subplot")

sgtitle("Subplot Grid Title")
```



(2). 修改标题外观

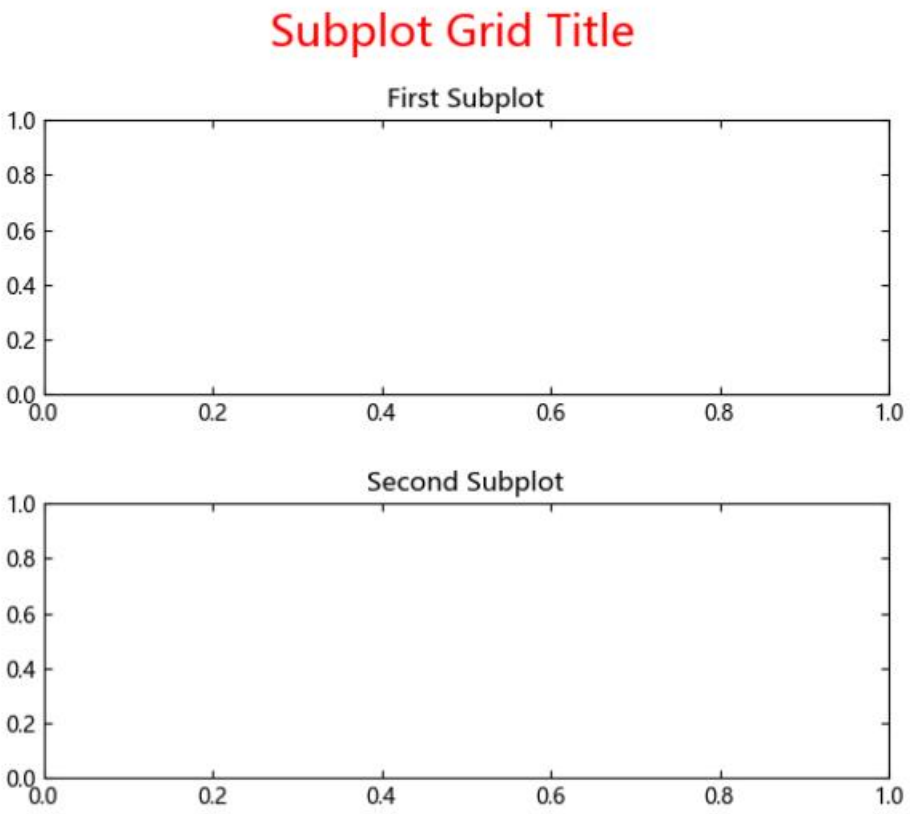
通过设置属性来修改标题外观，首先使用名称-值对组参数，然后返回创建

的 Text 对象并使用圆点表示法。

使用名称-值对组参数更改 color 属性。接下来，使用圆点表示法更改 fontsize 属性。

```
subplot(2,1,1)
title("First Subplot")
subplot(2,1,2)
title("Second Subplot")

sgt = sgtitle("Subplot Grid Title",color="red");
sgt.set_fontsize(20)
tightlayout()
```



xlabel

为 x 轴添加标签

A. 语法

```
xlabel(txt)
xlabel(target,txt)
```

```
xlabel(___,Key=Value)
t = xlabel(___)
```

B. 说明

xlabel(txt)对当前坐标区或独立可视化的 x 轴加标签。重新发出 xlabel 命令会将旧标签替换为新标签。

xlabel(target，txt)为指定的目标对象添加标签。

xlabel(___,Key=Value)使用一个或多个名称-值对组参数修改标签外观。例如，fontsize=12 将字体大小设置为 12 磅。在所有其他输入参数之后指定名称-值对组参数。并非所有图形类型均支持修改标签外观。

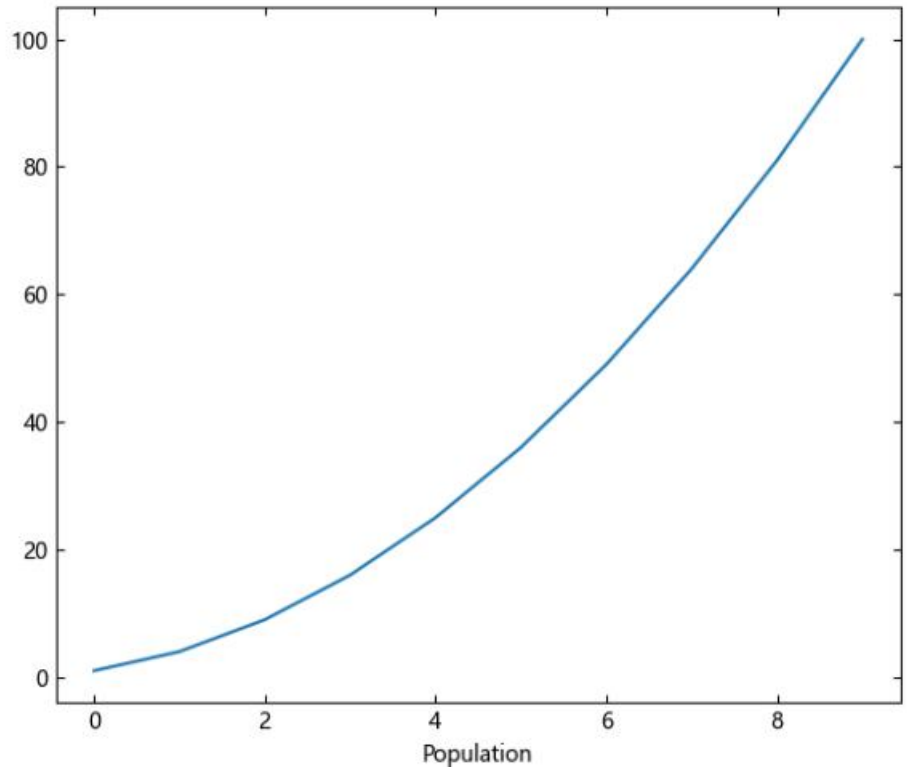
t=xlabel(___)返回用作 x 轴标签的文本对象。使用 t 可在创建标签后对其进行修改。

C. 示例

(1). 为 x 轴添加标签

将 Population 显示在 x 轴下

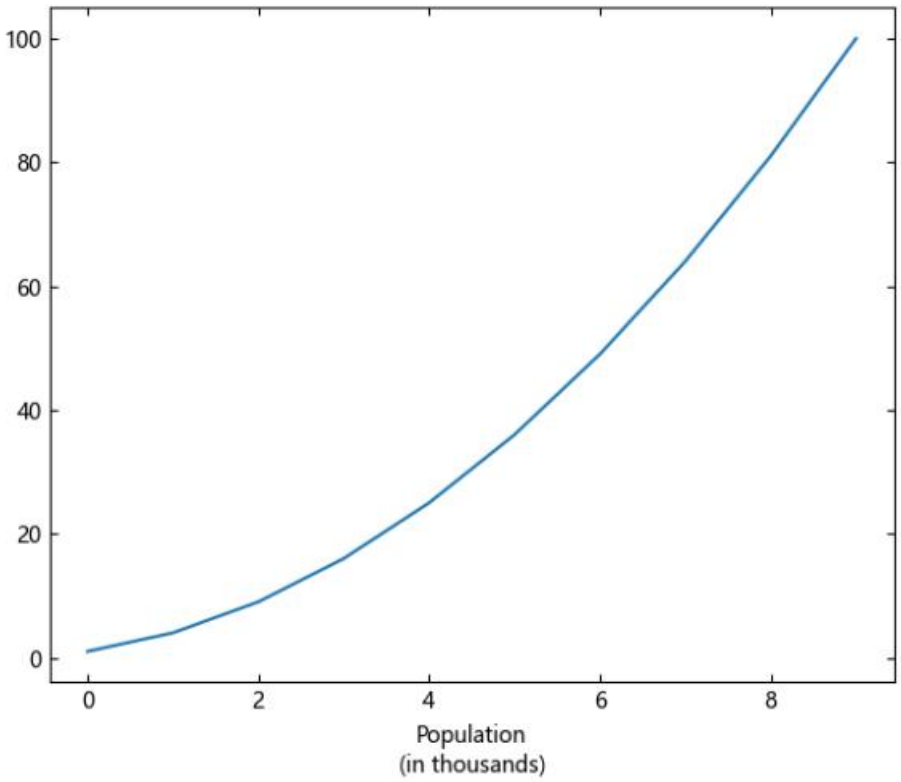
```
plot((1:10).^2)
xlabel("Population")
```



(2). 创建多行 x 轴标签

使用"\n"创建多行标签。

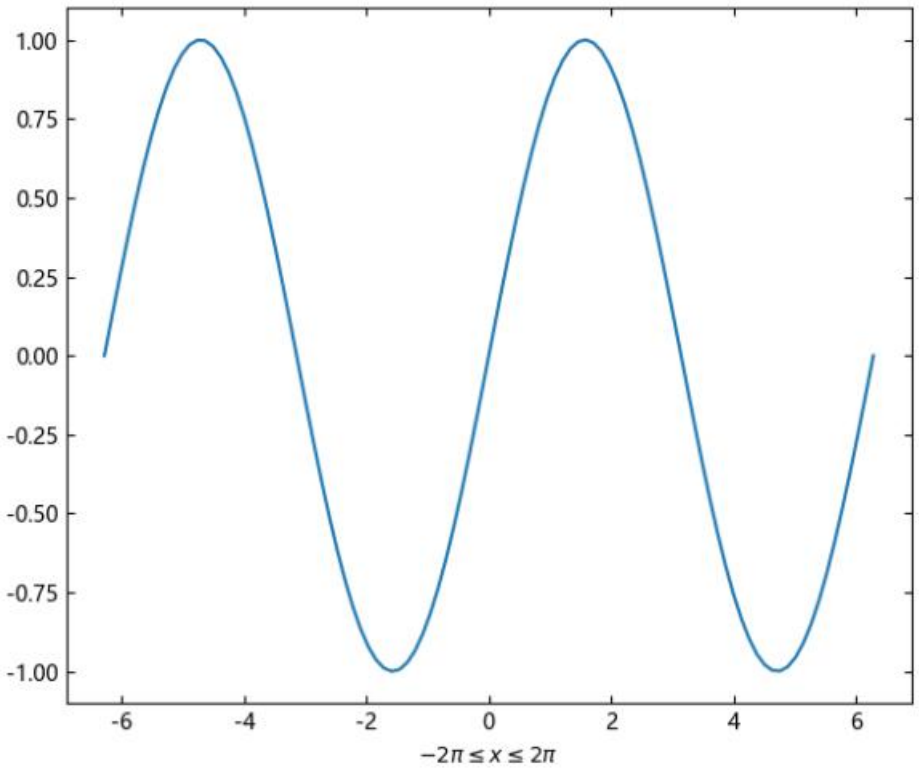
```
plot((1:10).^2)
xlabel("Population\n(in thousands)")
```



(3). 将希腊字母包括在 x 轴标签中

使用 TeX 标记将希腊字母和其他特殊字符包括在标签中。

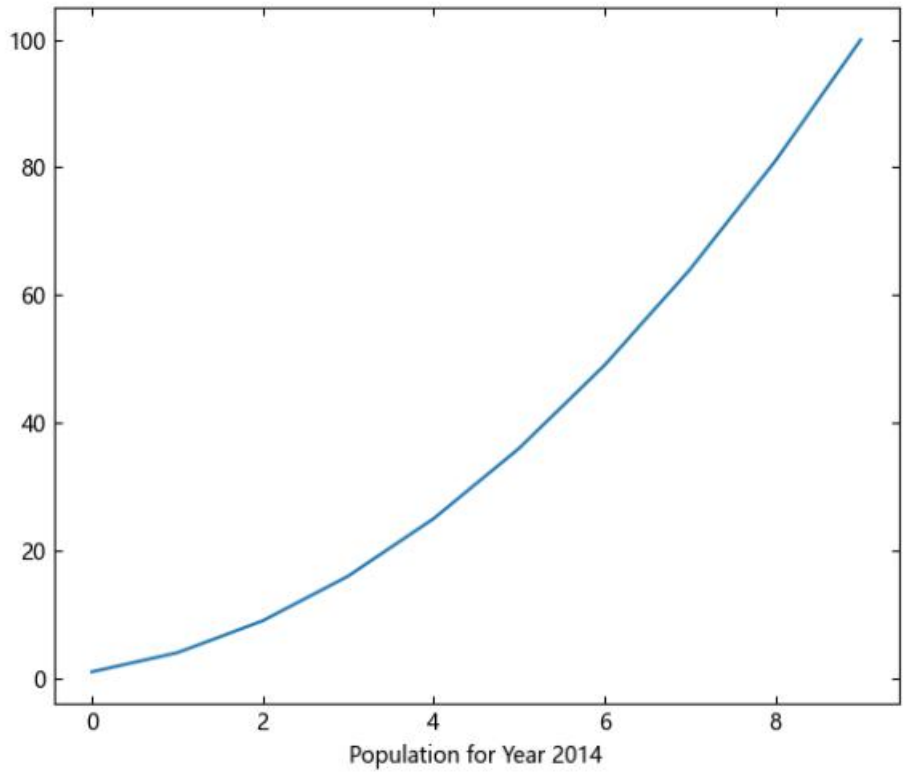
```
x = LinRange(-2 * pi, 2 * pi, 100);
y = sin.(x);
plot(x, y)
xlabel(raw"$-2\pi \leq x \leq 2\pi$")
```



(4). 在 x 轴标签中包括变量值

添加带有文本和变量值的标签。使用 string 函数在标签中包含变量值。

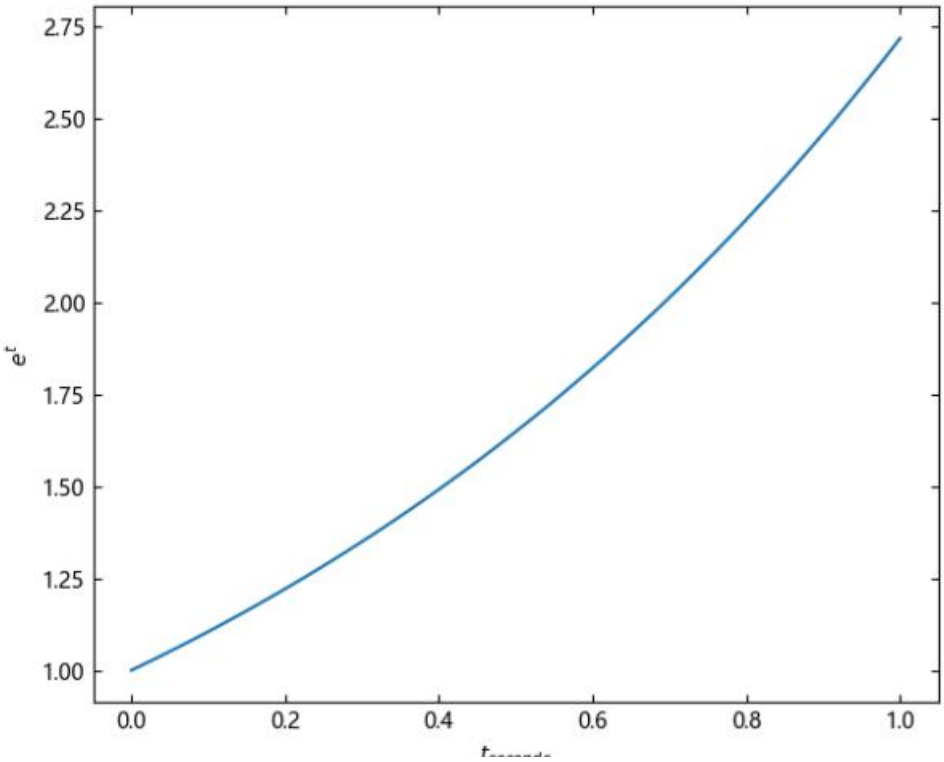
```
plot((1:10).^2)
year = 2014;
xlabel("Population for Year "*string(year))
```



(5). 在轴标签中包括上标和下标

使用 "^" 和 "_" 字符在轴标签中包含上标和下标。使用花括号 {} 修改多个字符。

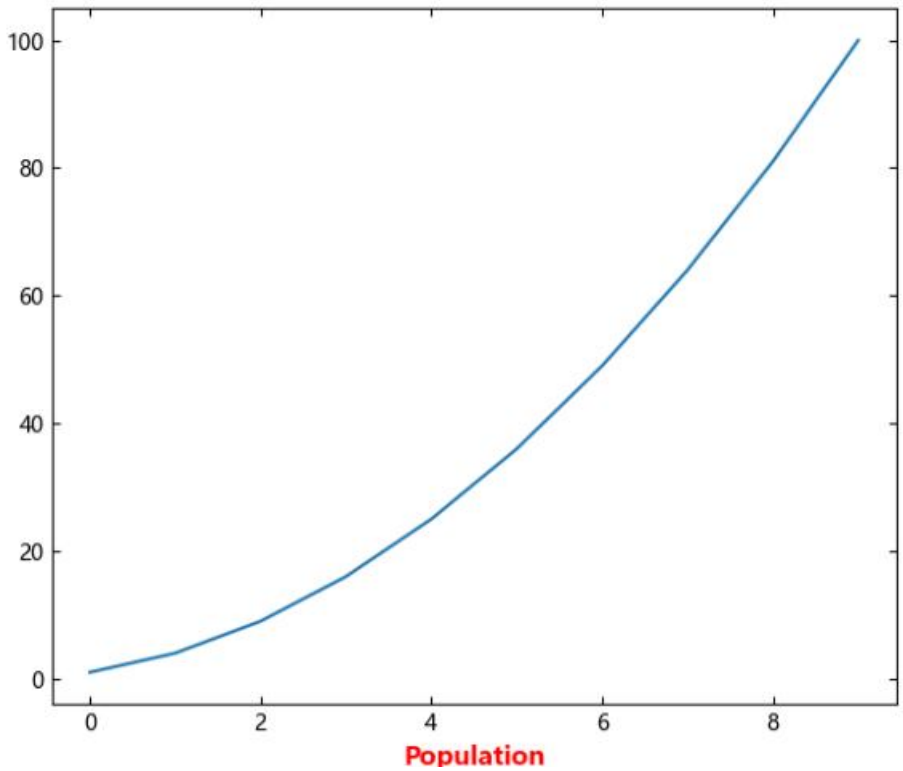
```
t = LinRange(0, 1, 100);
y = exp.(t);
plot(t, y)
xlabel(raw"$t_{seconds}$")
ylabel(raw"$e^t$")
```



(6). 更改 x 轴标签字体大小和颜色

使用 Key=Value 对组参数设置 x 轴标签的字体大小、字体粗细和文本颜色属性。

```
plot((1:10).^2)
xlabel("Population",fontsize=12,fontweight="bold",color="r")
```

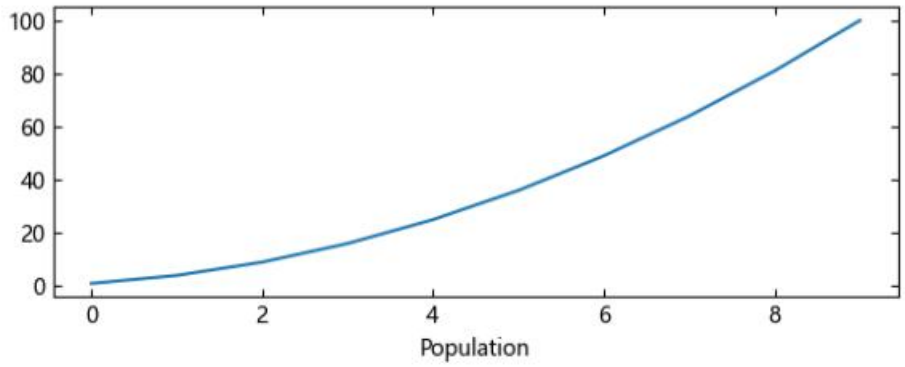


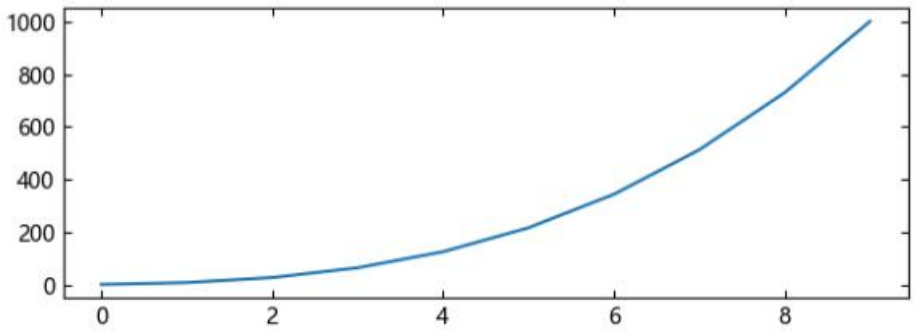
(7). 为特定绘图的 x 轴添加标签

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，创建坐标区对象 ax1 和 ax2。将数据绘制到每个坐标区中，并为顶部图创建 x 轴标签。

```
ax1 = subplot(2,1,1)
plot(ax1, (1:10).^2)
xlabel(ax1, "Population")

ax2 = subplot(2,1,2)
plot(ax2, (1:10).^3)
```





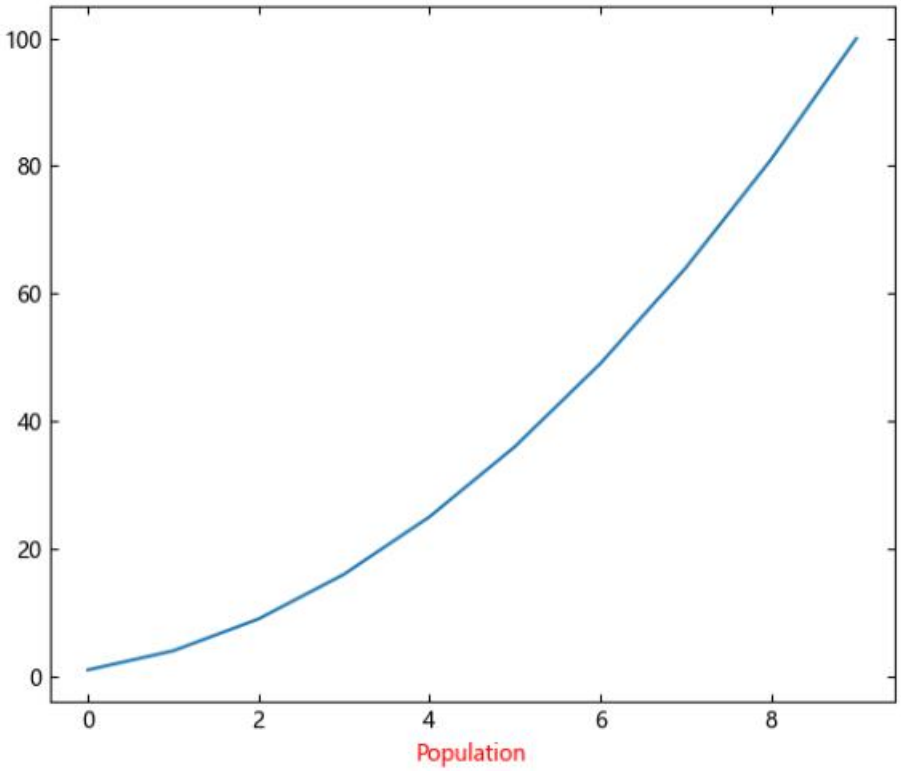
(8). 在创建后修改 x 轴标签

为 x 轴添加标签并返回用作标签的文本对象。

```
plot((1:10).^2)
t = xlabel("Population");
```

使用 t 可在创建标签后设置标签的文本属性。例如，将标签的颜色设置为红色。使用圆点表示法设置属性。

```
t.set_color("red")
gca().stale=true
```



ylabel

为 y 轴添加标签

A. 语法

```
ylabel(txt)
ylabel(target,txt)

ylabel(___,Key=Value)
t = ylabel(___)
```

B. 说明

ylabel(txt)对当前坐标区或独立可视化的 x 轴加标签。重新发出 ylabel 命令会将旧标签替换为新标签。

ylabel(target，txt) 为指定的目标对象添加标签。

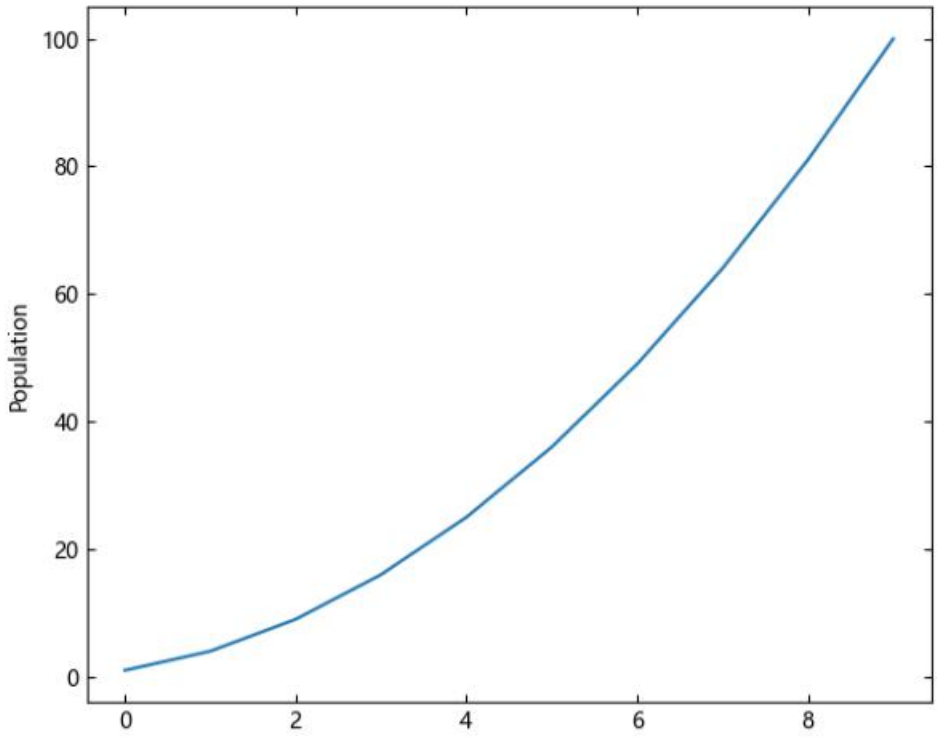
ylabel(___,Key=Value)使用一个或多个名称-值对组参数修改标签外观。例如，fontsize=12 将字体大小设置为 12 磅。在所有其他输入参数之后指定名称-值对组参数。并非所有图形类型均支持修改标签外观。

t = ylabel(___) 返回用作 x 轴标签的文本对象。使用 t 可在创建标签后对其进行修改。

C. 示例

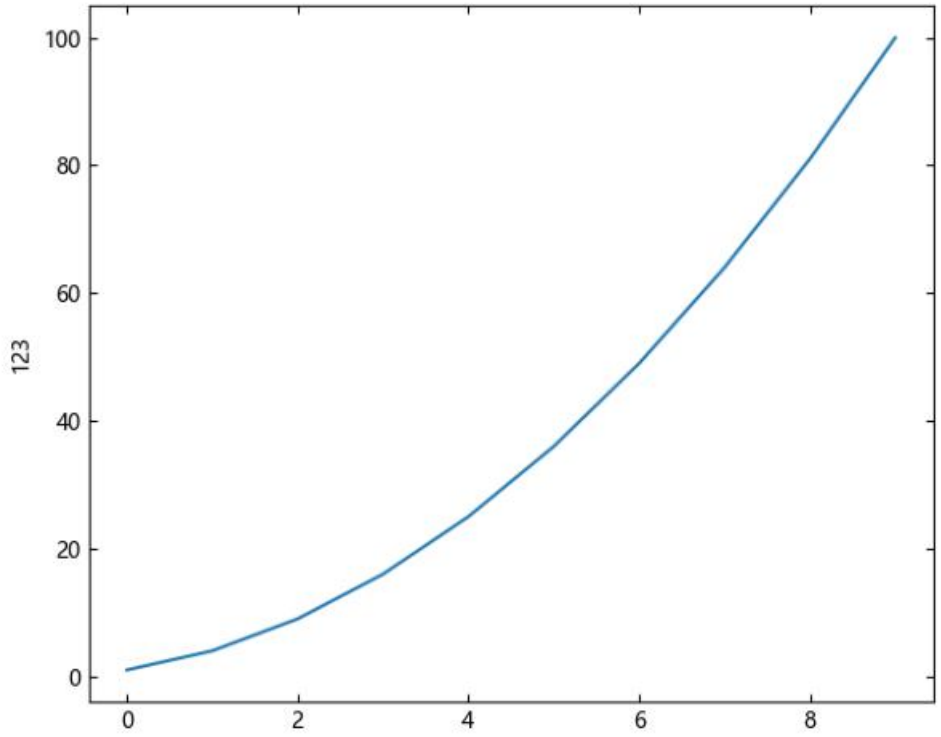
(1). 为 y 轴添加标签

```
plot((1:10).^2)
ylabel("Population")
```



(2). 使用数值输入为 y 轴添加标签

```
plot((1:10).^2)
ylabel(123)
```

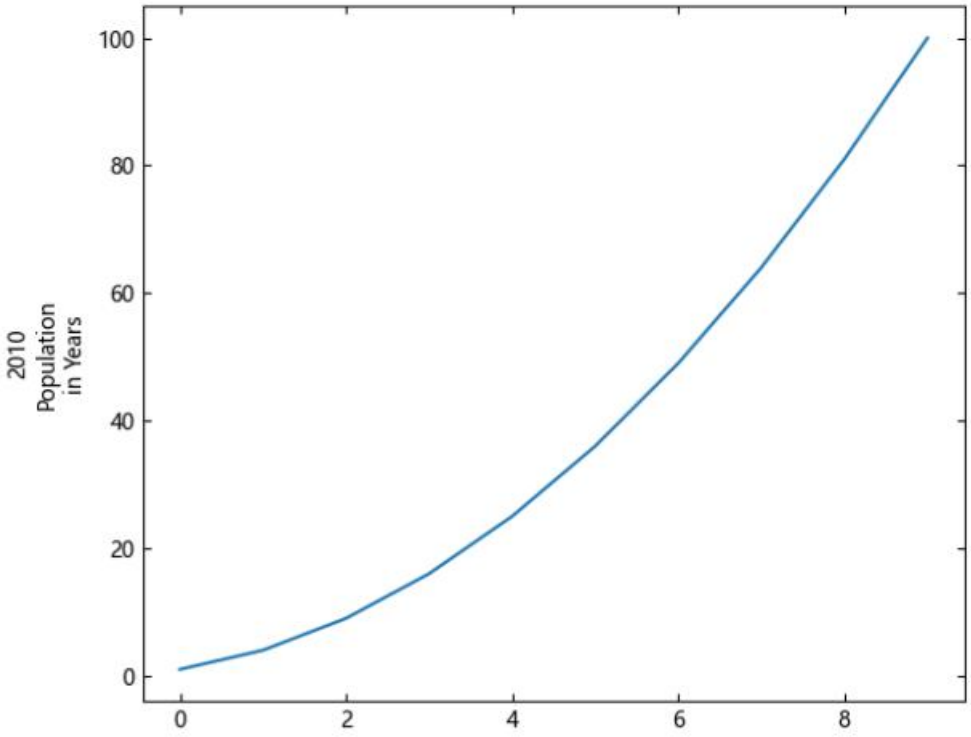


Syslab 将 123 显示在 y 轴旁边。

(3). 创建多行标签

使用"\n"创建多行标签。

```
plot((1:10).^2)
ylabel(string(2010)*"\nPopulation\nin Years")
```



(4). 创建 y 轴标签并设置字体属性

使用 Key=Value 对组设置 x 轴标签的字体大小、字体粗细和文本颜色属性。

```
plot((1:10).^2)
ylabel("Population", fontsize=12,
fontweight="bold",color="r")
```

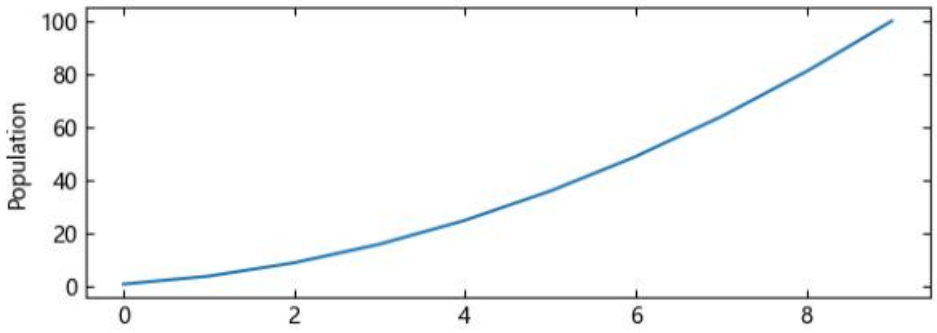
fontsize=12 采用 12 磅字体显示标签文本。fontweight="bold" 将文本设置为粗体。color="r" 将文本颜色设置为红色。

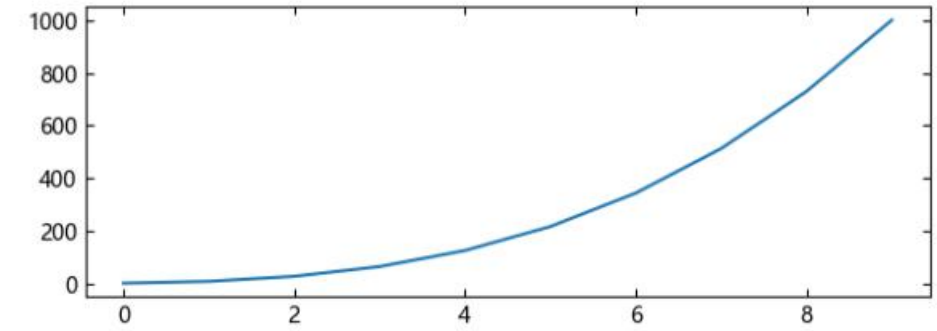
(5). 为特定坐标区的 y 轴添加标签

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，创建坐标区对象 ax1 和 ax2。将数据绘制到每个坐标区中，并为顶部图创建 y 轴标签。

```
ax1 = subplot(2,1,1)
plot(ax1, (1:10).^2)
ylabel(ax1, "Population")

ax2 = subplot(2,1,2)
plot(ax2, (1:10).^3)
```

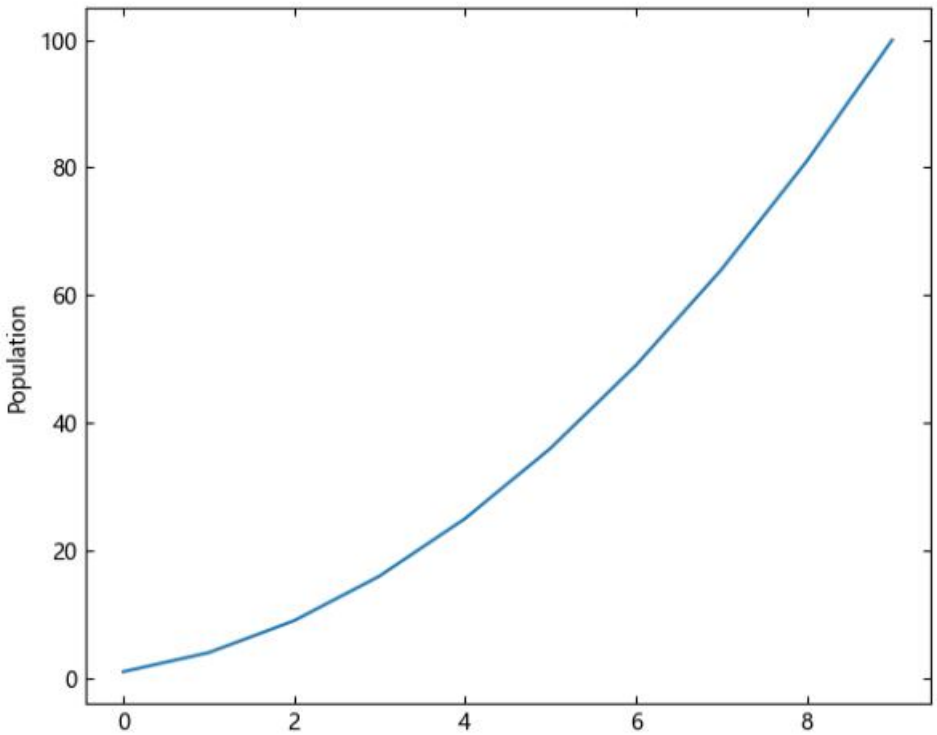




(6). 为 y 轴添加标签并返回对象句柄

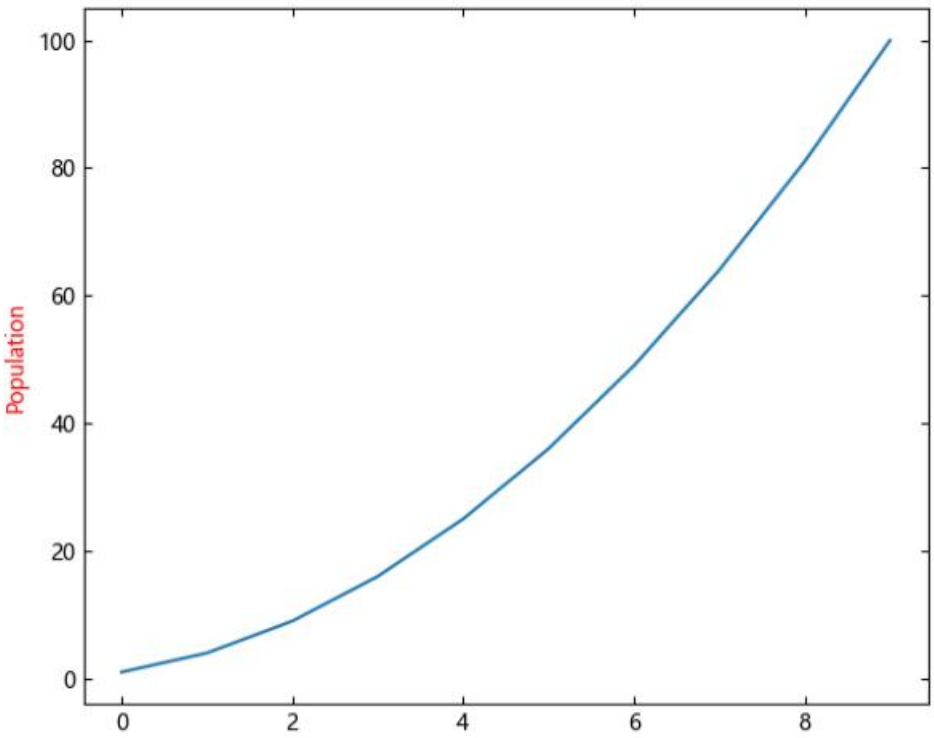
为 y 轴添加标签并返回用作标签的文本对象的句柄。

```
plot((1:10).^2)
t = ylabel("Population");
```



将标签的颜色设置为红色。使用圆点表示法设置属性。

```
t.set_color("red")
gca().stale=true
```



zlabel

为 z 轴添加标签

A. 语法

```
zlabel(txt)
zlabel(target,txt)

zlabel(___,Key=Value)
t = zlabel(___)
```

B. 说明

zlabel(txt)对当前坐标区或独立可视化的 z 轴加标签。重新发出 zlabel 命令会将旧标签替换为新标签。

zlabel(target，txt)为指定的目标对象添加标签。

zlabel(___,Key=Value)使用一个或多个名称-值对组参数修改标签外观。例如，fontsize=12 将字体大小设置为 12 磅。在所有其他输入参数之后指定名称-

值对组参数。并非所有图形类型均支持修改标签外观。

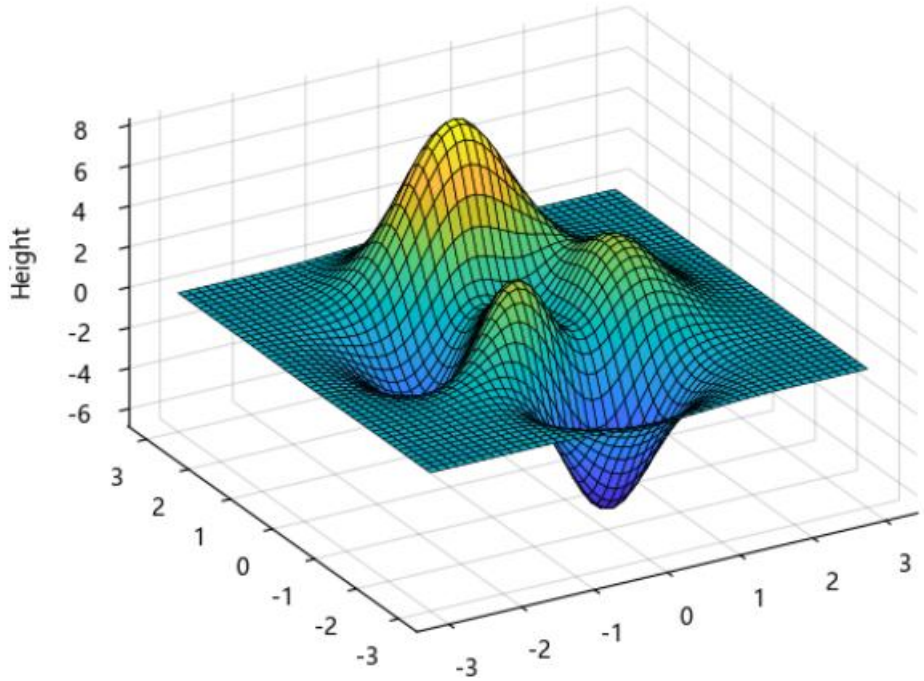
t=zlabel(___)返回用作 z 轴标签的文本对象。使用 t 可在创建标签后对其进行修改。

C. 示例

(1). 为 z 轴添加标签

将 Height 显示在 z 轴下。

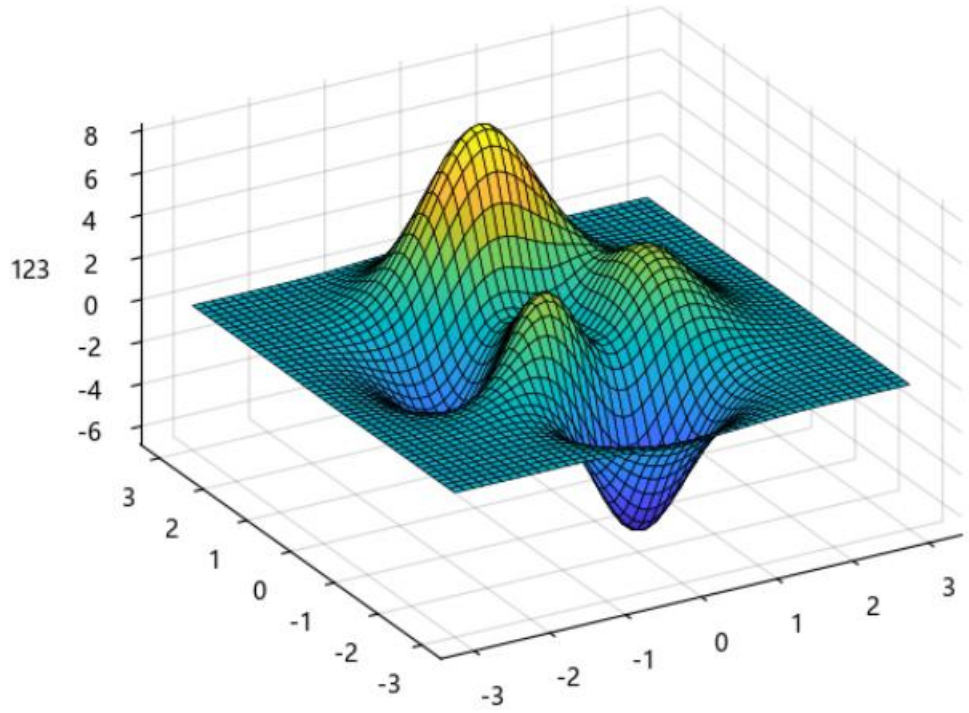
```
X,Y,Z = peaks()
surf(X,Y,Z)
zlabel("Height")
```



(2). 使用数值输入为 z 轴添加标签

使用数值输入为 z 轴添加标签。

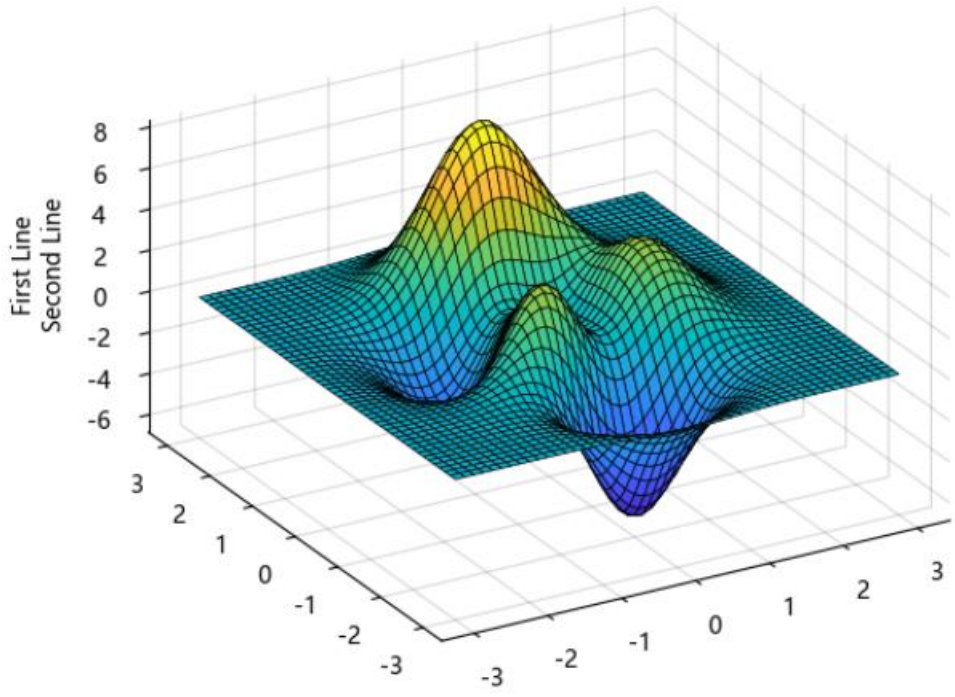
```
figure();
X,Y,Z = peaks()
surf(X,Y,Z)
zlabel("123")
```



(3). 创建多行 z 轴标签

使用"\n"创建多行标签。

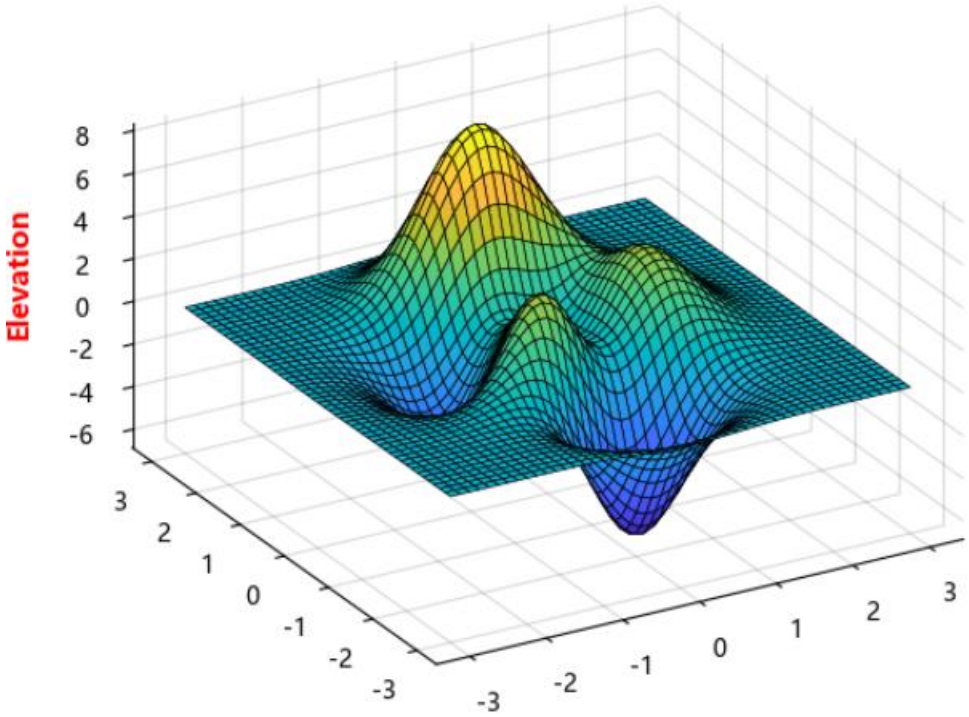
```
figure();
X,Y,Z = peaks()
surf(X,Y,Z)
zlabel("First Line\nSecond Line")
```



(4). 为 z 轴添加标签并设置字体属性

为 z 轴添加标签并设置字体属性。

```
X,Y,Z = peaks()
surf(X,Y,Z)
zlabel("Elevation", fontsize=12,
      fontweight="bold",color="r")
```

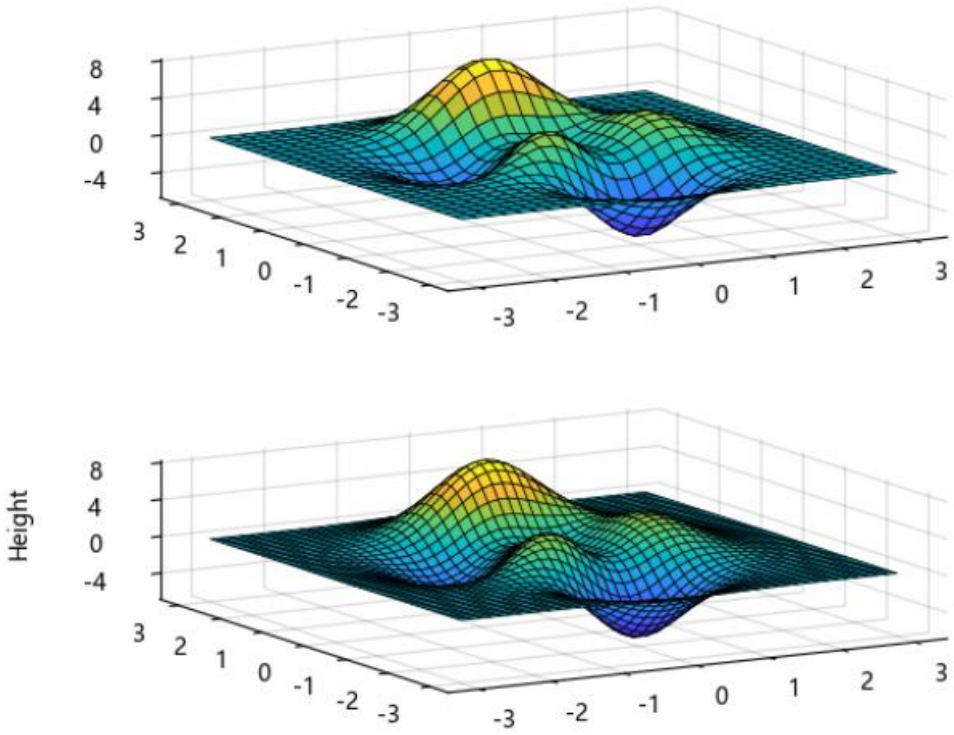


(5). 为特定坐标区的 z 轴添加标签

为特定坐标区的 z 轴添加标签。

```
ax1 = subplot(2,1,1,projection="3d")
X,Y,Z = peaks(30)
surf(ax1, X,Y,Z)

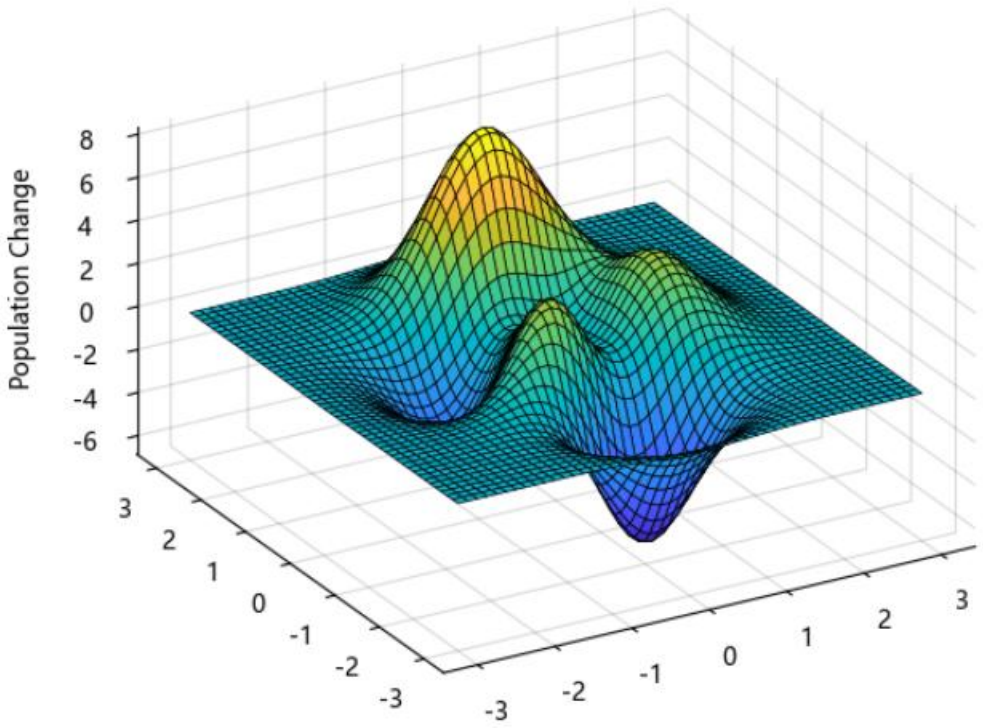
ax2 = subplot(2,1,2,projection="3d")
X2,Y2,Z2 = peaks(45)
surf(ax2, X2, Y2, Z2)
zlabel(ax2, "Height")
```



(6). 为 z 轴添加标签并返回对象句柄

为 z 轴添加标签并返回对象句柄。

```
X,Y,Z = peaks()
surf(X,Y,Z)
t = zlabel("Population Change");
```



legend

在坐标区上添加图例

A. 语法

```
legend()
legend(labels)
legend(subset,___)
legend(target,___)



legend(___,Key=Value)
lgd = legend(___)

legend("off")
```

B. 说明

legend 为每个绘制的数据序列创建一个带有描述性标签的图例。对于标签，图例默认使用 "dataN" 形式的标签。当您在坐标区上添加或删除数据序列时，图例会自动更新。此命令为 gca 返回的当前坐标区或图形创建图例。如果当前坐标区为空，则图例为空。如果坐标区不存在，此命令将创建坐标区。

legend(labels) 使用字符向量数组、字符串数组设置标签，例如 legend(["Jan","Feb","Mar"])。

legend(subset,___)仅在图例中包括 subset 中列出的数据序列的项。subset 以图形对象向量的形式指定。您可以在指定标签之前或不指定其他输入参数的情况下指定 subset。

legend(target,___)使用由 target 指定的坐标区或图，而不是当前坐标区或图。指定 target 作为第一个输入参数。

legend(___,Key=Value)使用一个或多个名称-值对组参数来设置图例属性。设置属性时，必须使用数组指定标签，例如 legend(["A","B"],fontsize=12)。

lgd = legend(___) 返回 legend 对象。可使用 lgd 在创建图例后查询和设置图例属性。

legend("off") 删除图例。

C. 示例

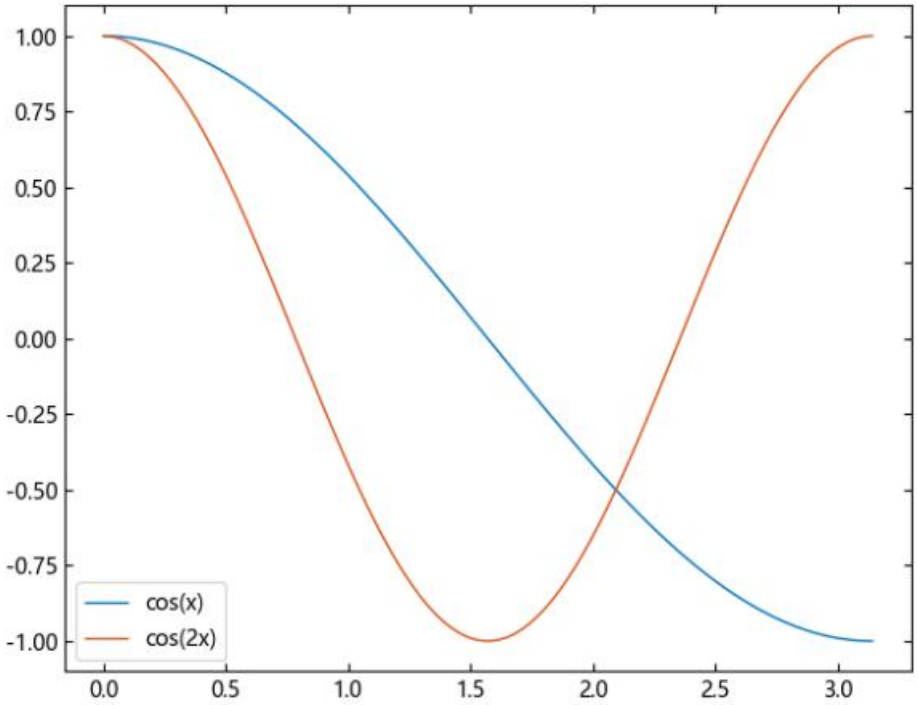
(1). 在当前坐标区上添加图例

绘制两个线条并在当前坐标区上添加一个图例。将图例标签指定为 legend 函数的输入参数。

```
x = LinRange(0, pi, 100);
y1 = cos.(x);
plot(x, y1)

hold("on")
y2 = cos.(2 * x);
plot(x, y2)

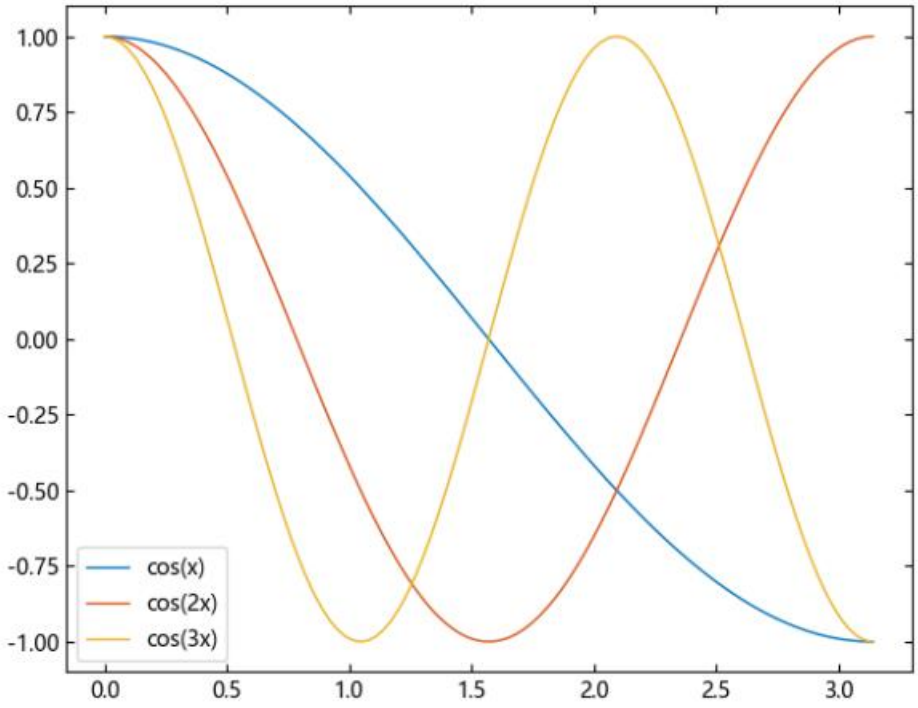
legend(["cos(x)", "cos(2x)"])
```



如果您在坐标区添加或删除数据序列，图例会相应地更新。创建数据序列时，可通过将 label 属性设置为名称-值对组来控制新数据序列的标签。如果您不指定标签，则图例使用 "dataN" 形式的标签。

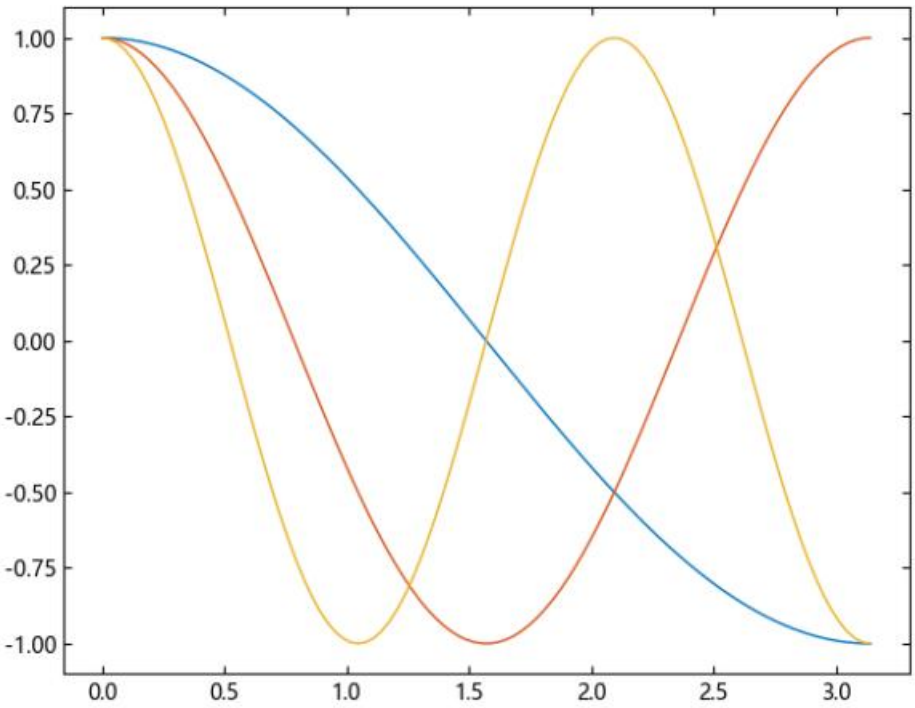
```
y3 = cos.(3 * x);
plot(x, y3; label="cos(3x)")

hold("off")
```



删除图例。

```
legend("off")
```



(2). 在特定坐标区上添加图例

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。通过将 ax1 传递给 plot 函数来创建顶部绘图。重复该过程以创建底部绘图。在每个坐标区中对矩阵数据绘图。通过指定 ax1 作为

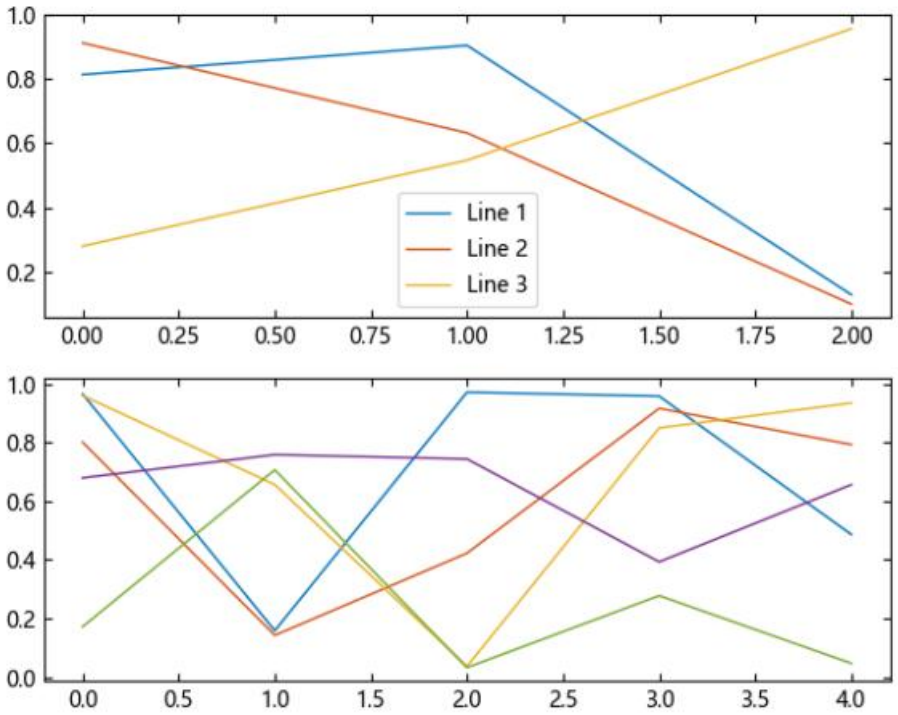
legend 的第一个输入参数，在上部绘图中添加一个图例。

```
y1 = [
    0.8147 0.9134 0.2785
    0.9058 0.6324 0.5469
    0.1270 0.0975 0.9575
]

y2 = [
    0.9649 0.8003 0.9595 0.6787 0.1712
    0.1576 0.1419 0.6557 0.7577 0.7060
    0.9706 0.4218 0.0357 0.7431 0.0318
    0.9572 0.9157 0.8491 0.3922 0.2769
    0.4854 0.7922 0.9340 0.6555 0.0462
]

ax1 = subplot(2, 1, 1)
ax2 = subplot(2, 1, 2)

plot(ax1, y1)
plot(ax2, y2)
legend(ax1, ["Line 1", "Line 2", "Line 3"])
```



(3). 在执行绘图命令的过程中指定图例标签

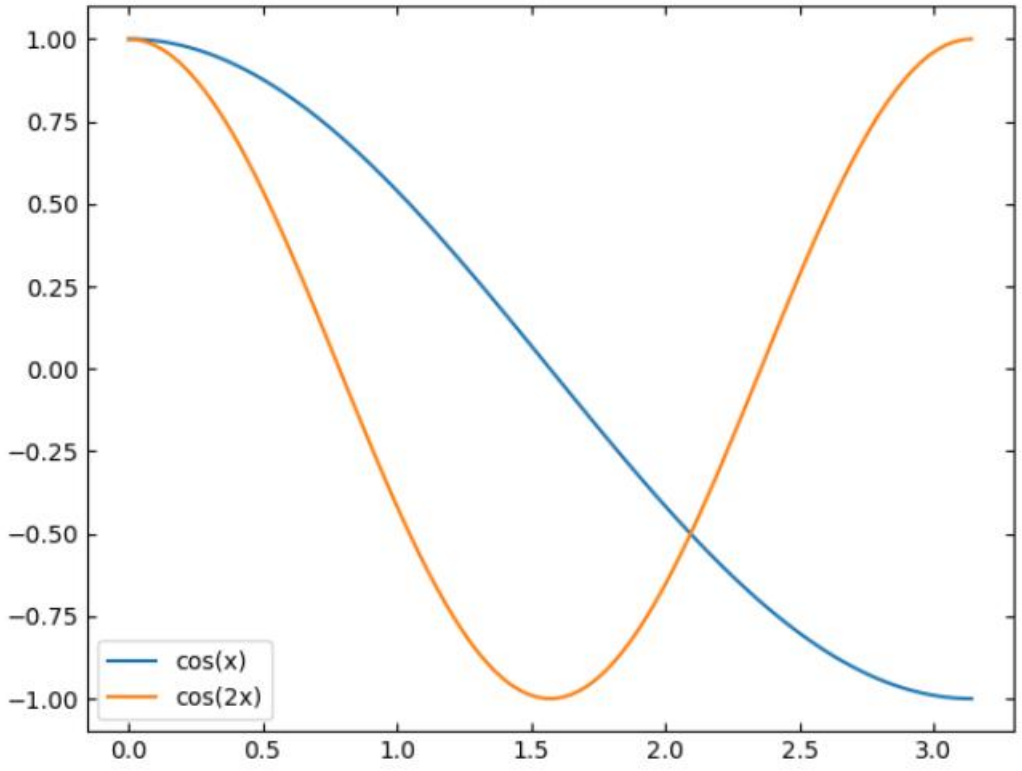
绘制两个线条。通过将 label 标签设置为所需的文本，在执行绘图命令的过

程中指定图例标签。然后，添加一个图例。

```
x = LinRange(0, pi, 100);
y1 = cos.(x);
plot(x, y1; label="cos(x)")
hold("on")

y2 = cos.(2 * x);
plot(x, y2; label="cos(2x)")

hold("off")
legend()
```



(4). 图例位置和列数

绘制四个线条。在坐标区的西北角创建一个图例。使用 ncol 属性指定图例的列数。

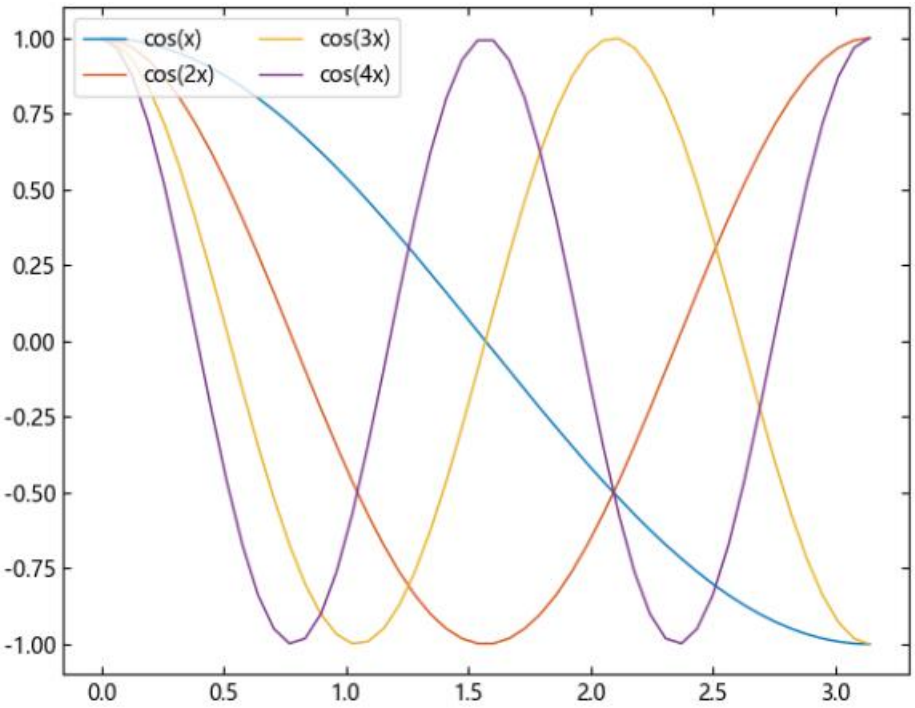
```
x = LinRange(0, pi, 50);
y1 = cos.(x);
plot(x, y1)

hold("on")
y2 = cos.(2 * x);
plot(x, y2)
```

```
y3 = cos.(3 * x);
plot(x, y3)

y4 = cos.(4 * x);
plot(x, y4)
hold("off")

legend(["cos(x)", "cos(2x)", "cos(3x)", "cos(4x)"]; loc="upper
left", ncol=2)
```



(5). 在图例中包含部分图形对象

如果您不想将绘制的所有图形对象都包含在图例中，可以指定要包含的图形对象。

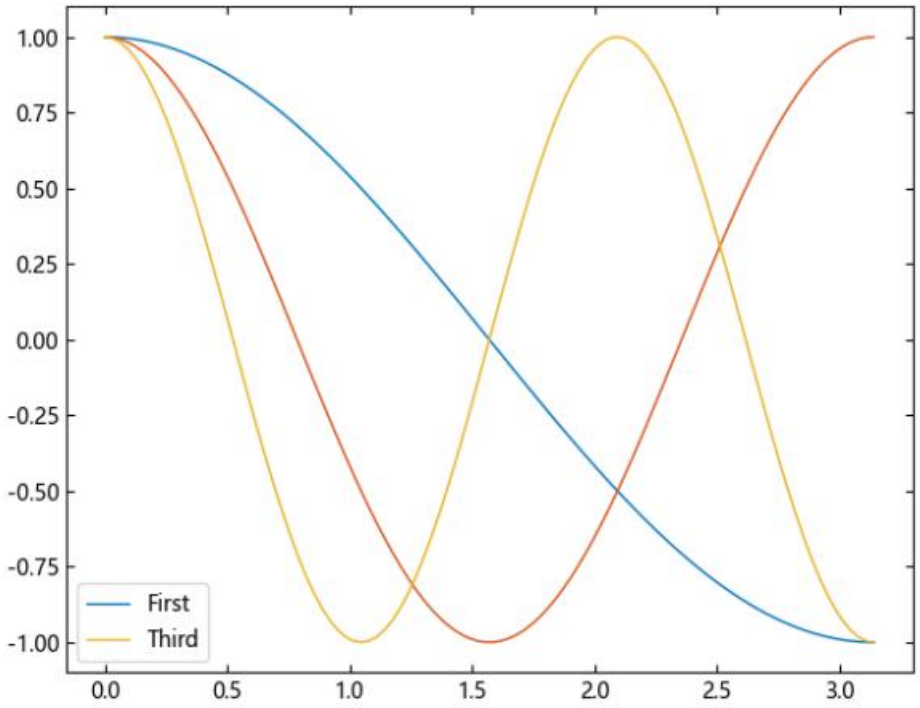
绘制三个线条并返回创建的 Line 对象。创建只包含其中两条线的图例。将第一个输入参数指定为要包含的 Line 对象的向量。

```
x = LinRange(0, pi, 100);
y1 = cos.(x);
p1 = plot(x, y1);

hold("on")
y2 = cos.(2 * x);
p2 = plot(x, y2);
```

```
y3 = cos.(3 * x);
p3 = plot(x, y3);
hold("off")

legend([p1[1], p3[1]], ["First", "Third"])
```



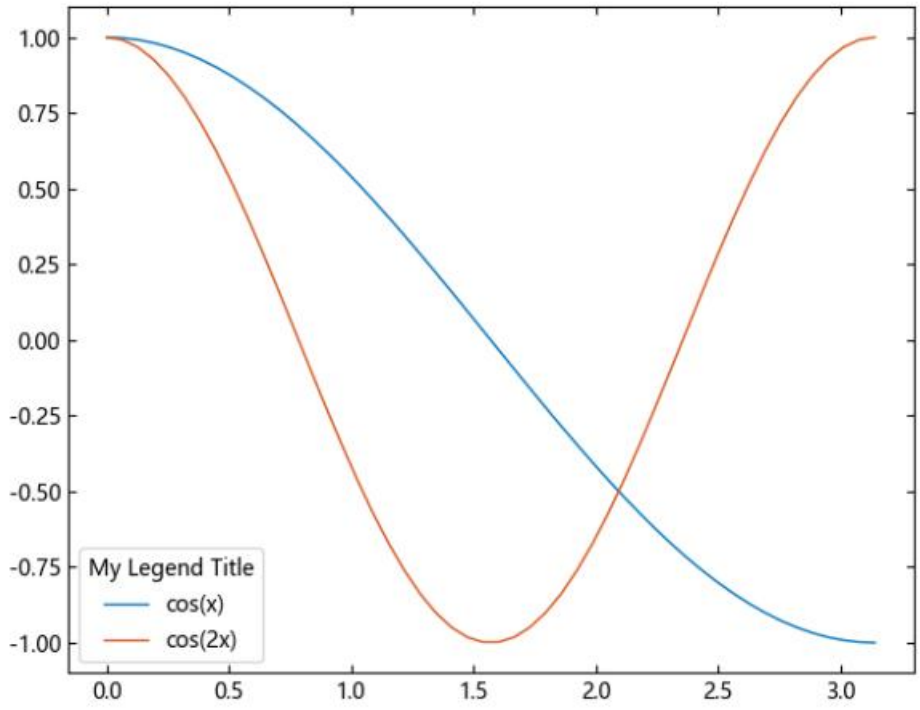
(6). 为图例添加标题

绘制两个线条并创建一个图例。然后为图例添加标题。

```
x = LinRange(0, pi, 50);
y1 = cos.(x);
plot(x, y1)

hold("on")
y2 = cos.(2 * x);
plot(x, y2)
hold("off")

lgd = legend(["cos(x)", "cos(2x)"]);
title(lgd, "My Legend Title")
```



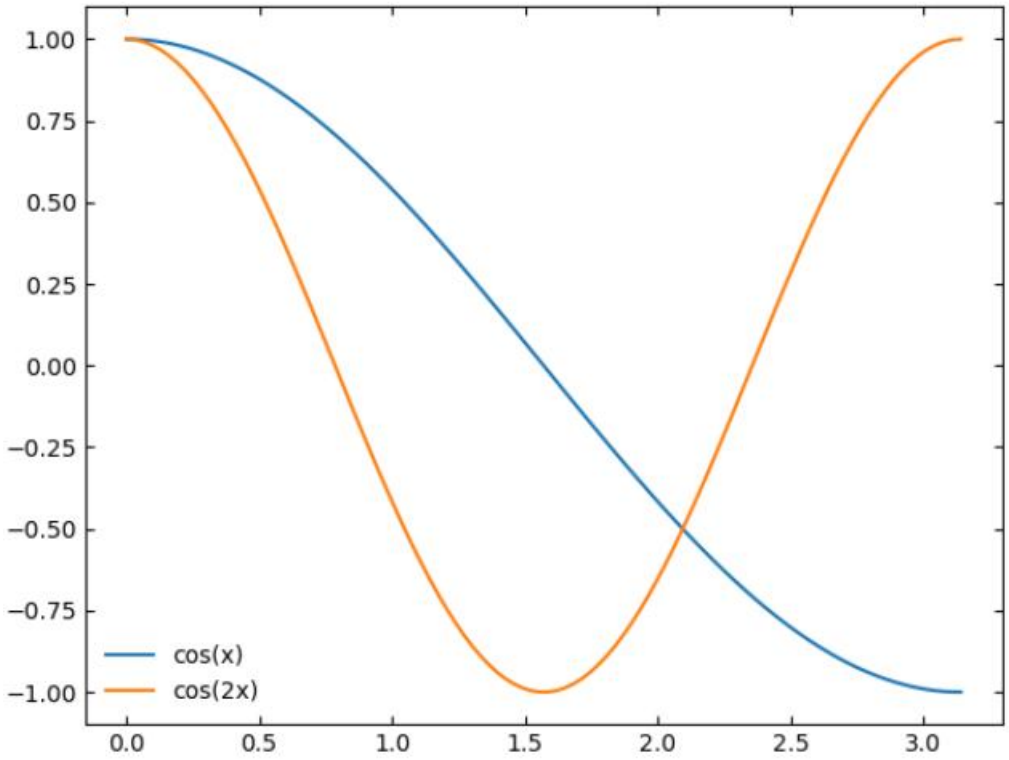
(7). 删除图例背景

绘制两个线条并在坐标区左下角创建一个图例。然后，删除图例的背景和轮廓。

```
x = LinRange(0, pi, 100);
y1 = cos.(x);
plot(x, y1)

hold("on")
y2 = cos.(2 * x);
plot(x, y2)
hold("off")

lgd = legend(["cos(x)", "cos(2x)"])
lgd.set_frame_on(false)
```



(8). 修改图例外观

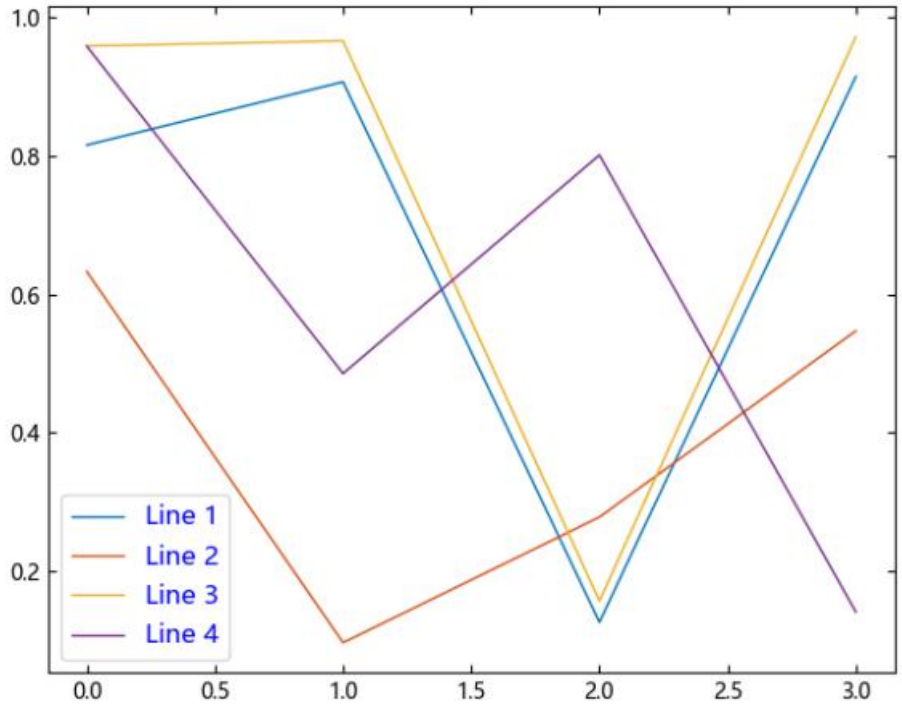
通过设置 legend 属性来修改图例外观。创建图例时，可以在 legend 命令中使用名称-值对组来设置属性。还可以在创建图例后使用 legend 对象来设置属性。

绘制四行随机数据。创建图例并将 legend 对象赋给变量 lgd。使用名称-值对组设置 fontsize 和 labelcolor 属性。当您指定名称-值对组参数时，必须使用数组来指定图例标签。

```
rdm = [
    0.8147 0.6324 0.9575 0.9572
    0.9058 0.0975 0.9649 0.4854
    0.1270 0.2785 0.1576 0.8003
    0.9134 0.5469 0.9706 0.1419
]

plot(rdm)

lgd = legend(["Line 1", "Line 2", "Line 3", "Line 4"]; fontsize=12,
labelcolor="blue")
```



3.3.2.1.2 注释

函数名	简介
text	向数据点添加文本说明
gtext	使用鼠标将文本添加到图窗
xline	具有常量 x 值的垂直线
yline	具有常量 y 值的水平线
annotation	创建注释
datacursormode	开启数据提示
datatip	创建数据提示
line	创建基本线条
rectangle	创建带有尖角或圆角的矩形
ginput	标识坐标区坐标

text

向数据点添加文本说明

A. 语法

```
text(x,y,txt)
text(x,y,z,txt)
text(___,Key=Value)
text(ax,___)

t = text(___)
```

B. 说明

text(x,y,txt)使用由 txt 指定的文本，向当前坐标区中的一个或多个数据点添加文本说明。若要将文本添加到一个点，请将 x 和 y 指定为标量。若要将文本添加到多个点，请将 x 和 y 指定为长度相同的向量。

text(x,y,z,txt) 在三维坐标中定位文本。

text(___,Key=Value)使用一个或多个名称-值对组指定 Text 对象的属性。例如，fontzize=14 将字体大小设置为 14 磅。您可以使用上述语法中的任意输入参数组合指定文本属性。

text(ax,___) 将在由 ax 指定的笛卡尔坐标区、极坐标区中创建文本，而不是在当前坐标区 (gca) 中创建文本。选项 ax 可以位于前面的语法中的任何输入参数组合之前。

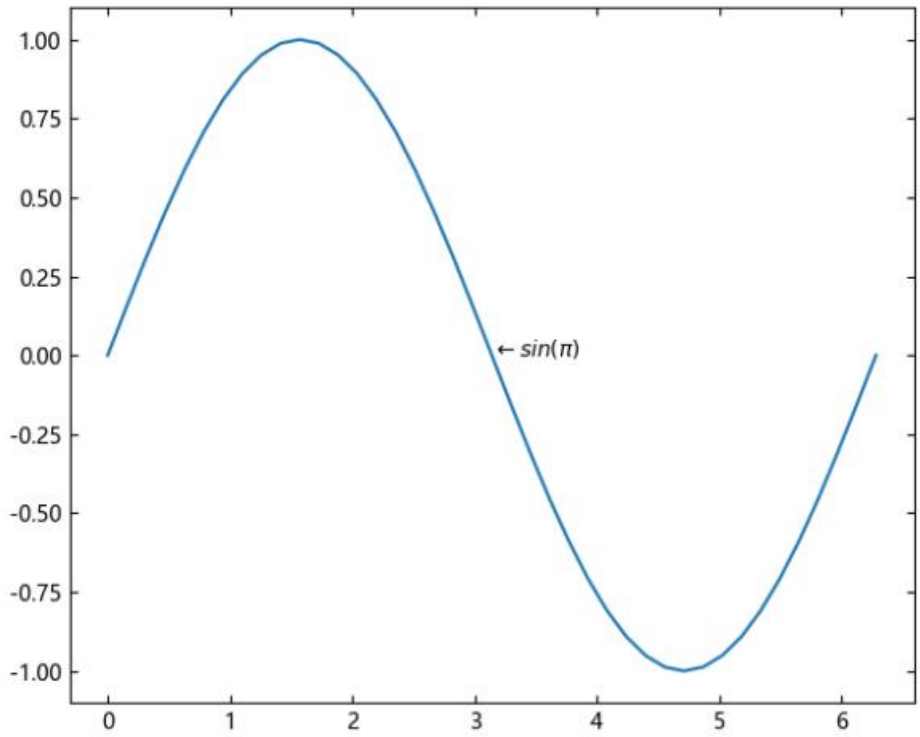
t=text(___)返回一个或多个文本对象。使用 t 修改所创建的文本对象的属性。

C. 示例

(1). 向数据点添加文本说明

绘制一条正弦曲线。在点 (π,0) 处，添加文本说明 sin(π)。使用 TeX 标记 \pi 表示希腊字母 π。使用 \leftarrow 显示一个向左箭头。

```
x = 0:pi/20:2*pi;
y = sin.(x);
plot(x,y)
text(pi,0, raw"$\leftarrow sin(\pi)$")
```

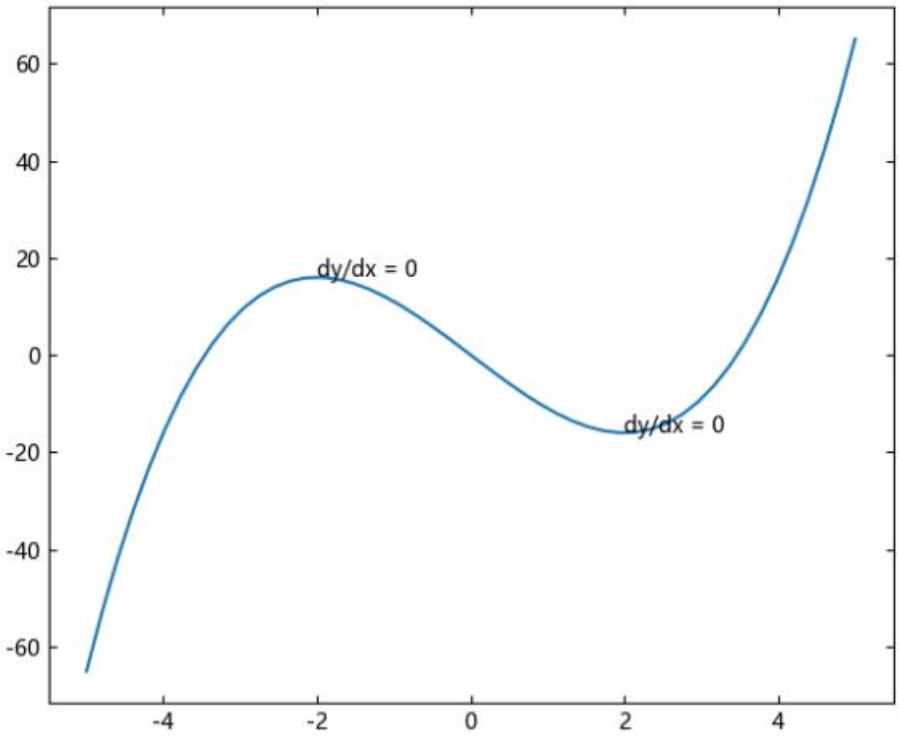


(2). 向多个数据点添加文本

绘制一个线条。将相同的文本添加到沿该线条的两个点。

```
x = LinRange(-5, 5, 50);
y = x .^ 3 - 12 .* x;
plot(x, y)

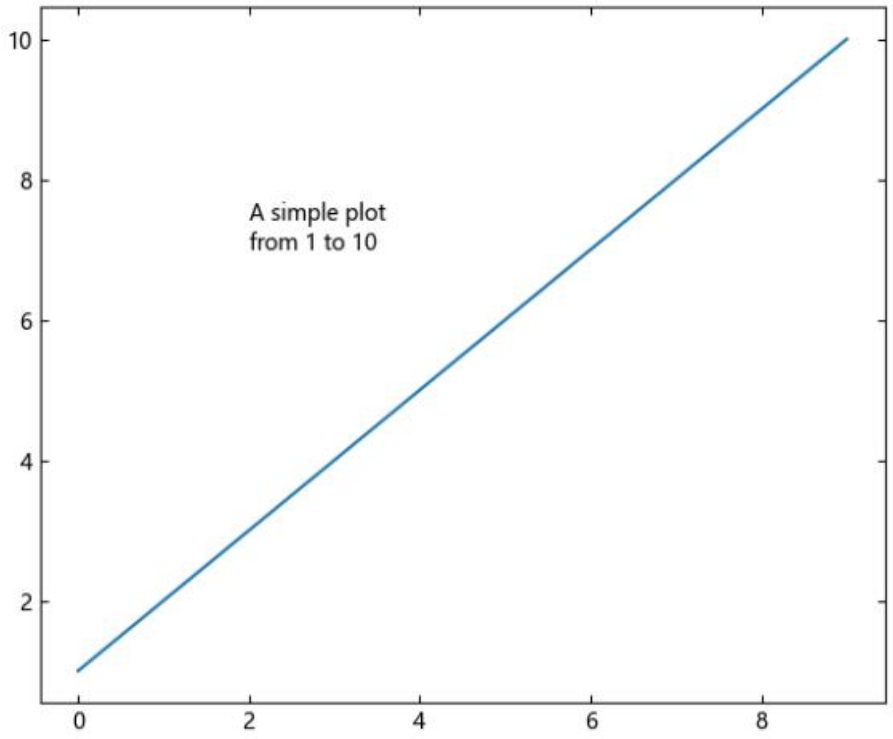
xt = [-2 2];
yt = [16 -16];
str = "dy/dx = 0";
text(xt, yt, str)
```



(3). 显示多行文本

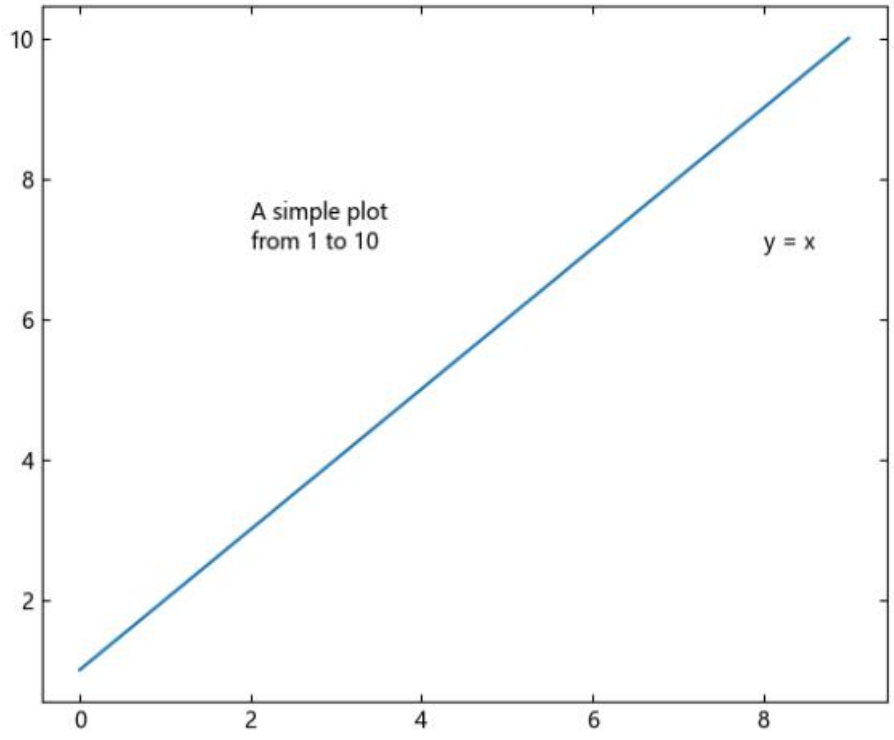
创建一个线图，并向坐标区添加一个文本说明。通过将 str 指定为数组显示多行文本。

```
plot(1:10)
str = ["A simple plot\nfrom 1 to 10"];
text([2],[7],str)
```



创建一个线图，并向坐标区添加两个文本说明。在向坐标区添加多个文本说明时，通过指定嵌套数组显示多行文本。

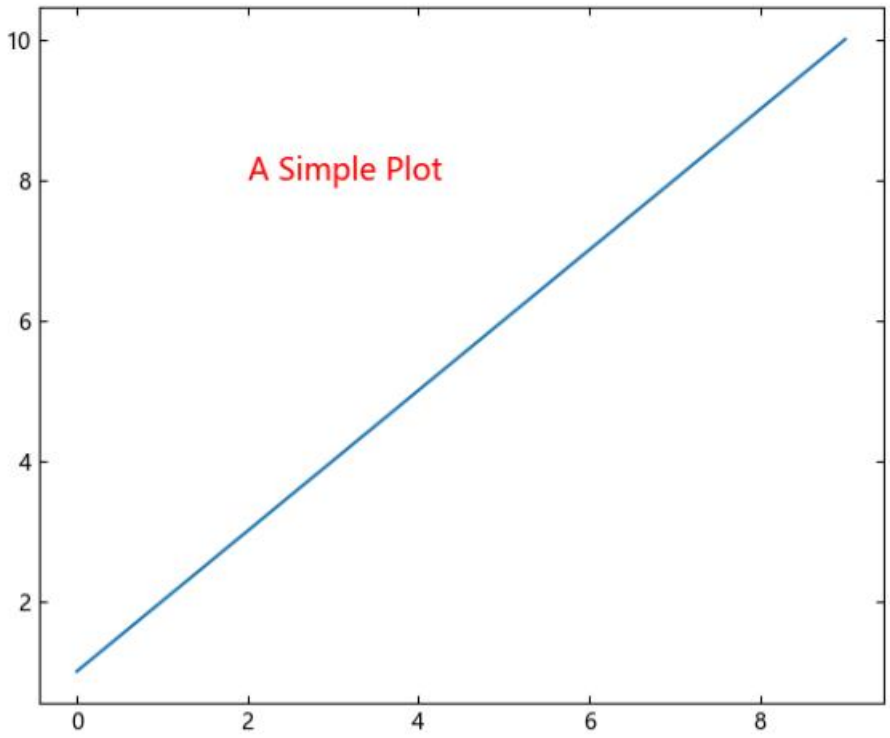
```
str = ["y = x"];
text([8],[7],str)
```



(4). 指定文本大小和颜色

创建一个线图，并向坐标区添加文本说明。使用红色 14 号字体。

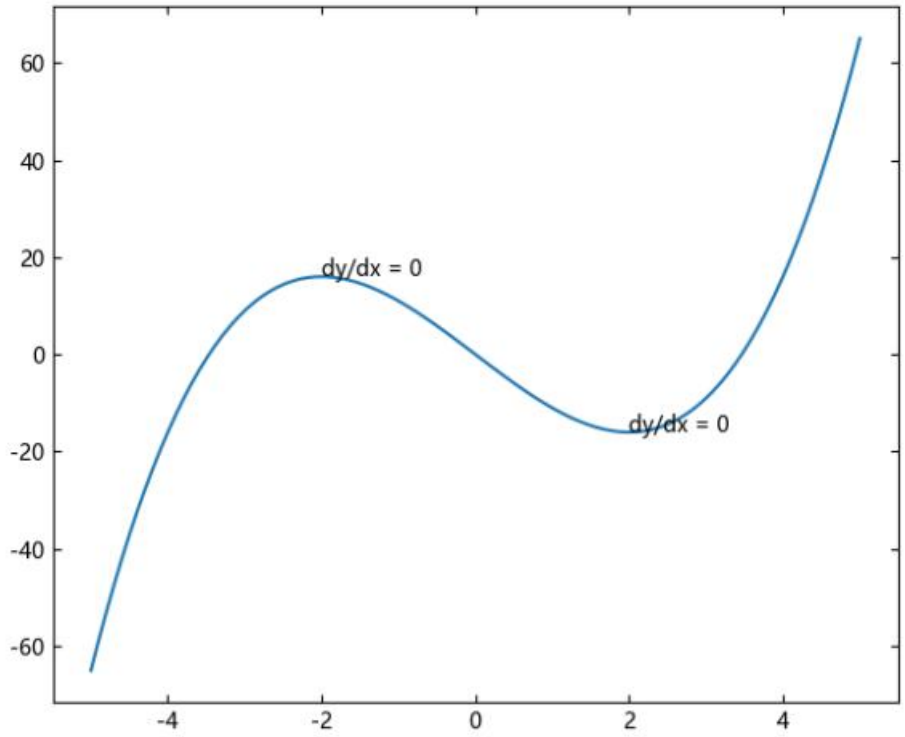
```
plot(1:10)
text(2, 8, "A Simple Plot", color="red", fontsize=14)
```



(5). 修改现有文本

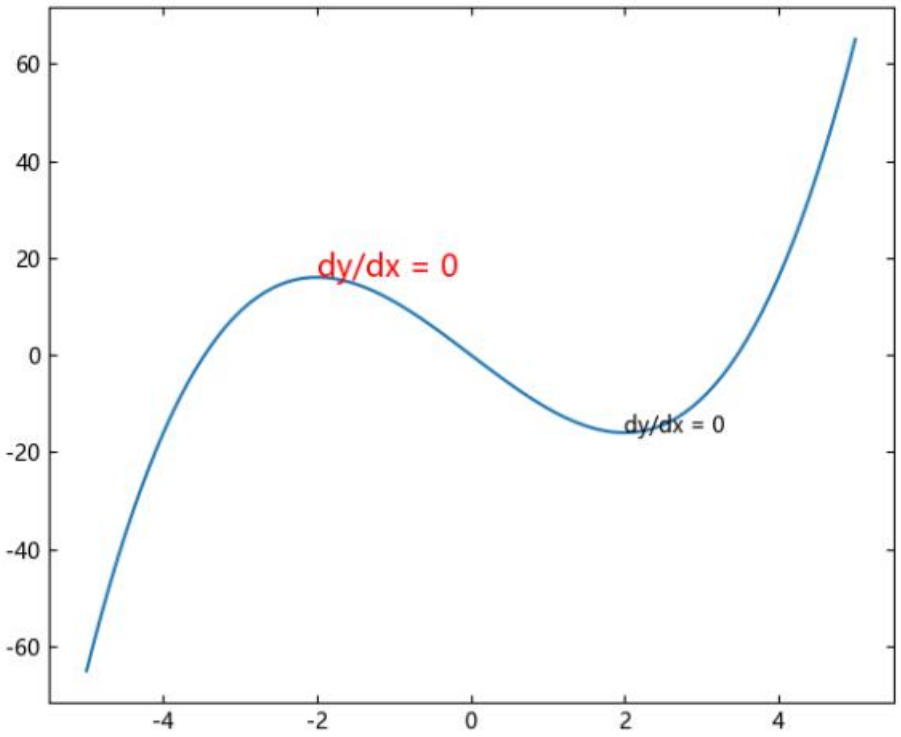
创建一个线图，并沿线条添加两个文本说明。返回文本对象 t。

```
x = LinRange(-5, 5, 100);
y = x .^ 3 - 12 .* x;
plot(x, y)
t = text([-2 2], [16 -16], "dy/dx = 0")
```



text 函数为每个文本说明创建一个文本对象。因此，t 包含两个文本对象。使用 t(1) 更改第一个文本对象的颜色和字体大小。使用圆点表示法设置属性。

```
t[1].set_color("red")
t[1].set_fontsize(14)
```



gtext

使用鼠标将文本添加到图窗

A. 语法

```
gtext(str)
gtext(str,Key=Value)

t = gtext(___)
```

B. 说明

gtext(str)在您使用鼠标选择的位置插入文本 str。gtext 将等待您选择位置。将鼠标指针移至所需位置并点击图窗或按任意键（Enter 键除外）。

gtext(str,Key=Value) 使用一个或多个名称-值对组参数指定文本属性。例如，fontsize=14 指定 14 磅字体。

t = gtext(___) 返回由 gtext 创建的文本对象的数组。使用 t 修改所创建的文本对象的属性。

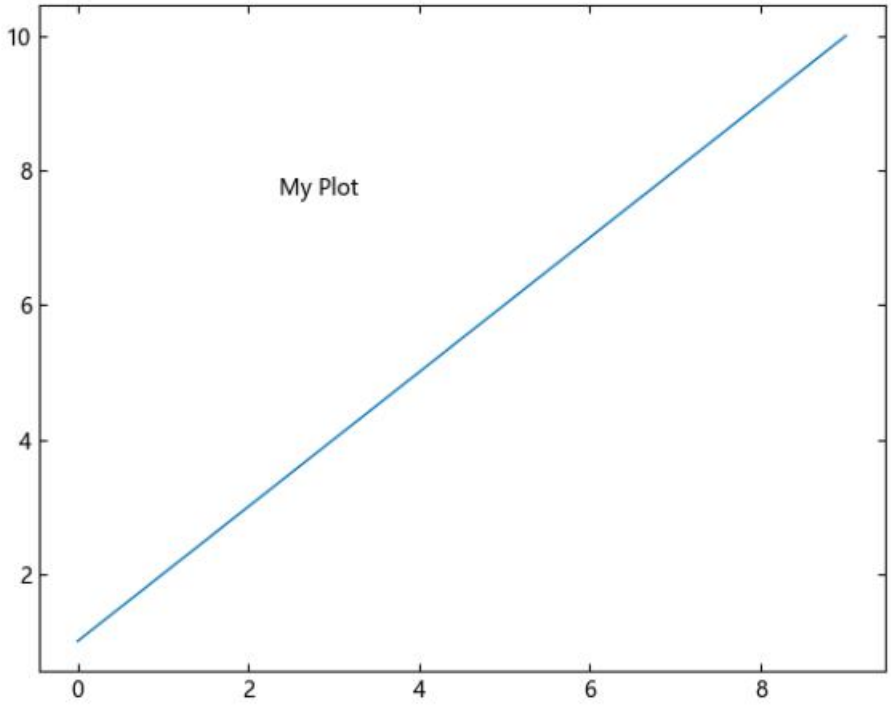
C. 示例

(1). 使用鼠标将文本添加到图窗

创建简单的线图并通过 gtext 使用鼠标将文本添加到图窗。

点击图窗以将文本放置在所选位置。

```
plot(1:10)
gtext("My Plot")
```

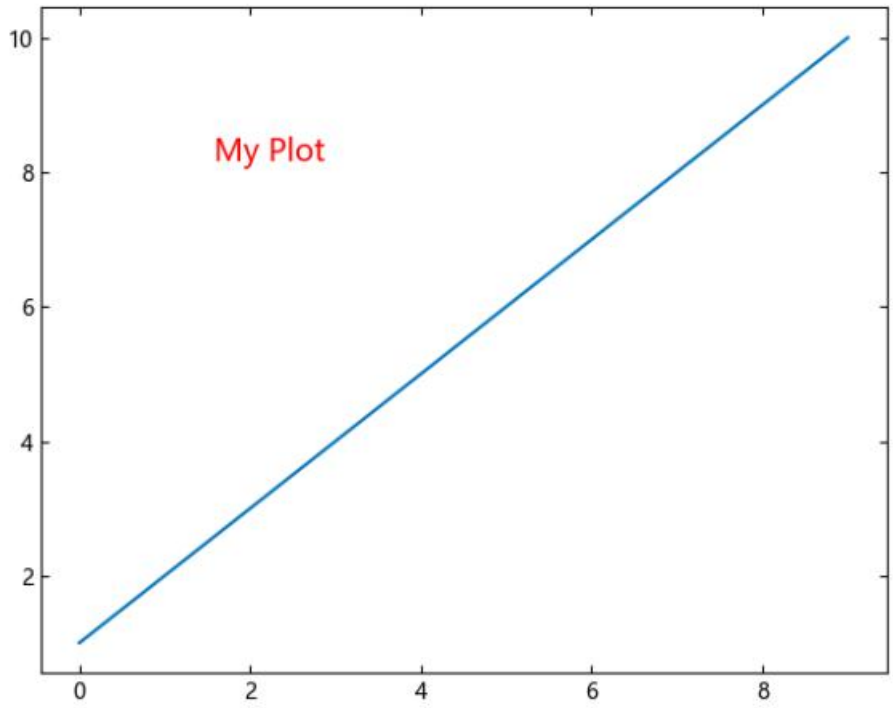


(2). 指定字体大小和颜色

创建简单的线图并使用鼠标将文本添加到图窗。使用红色、14 磅字体。

点击图窗中要显示文本的位置。

```
plot(1:10)
gtext("My Plot",color="red",fontsize=14)
```

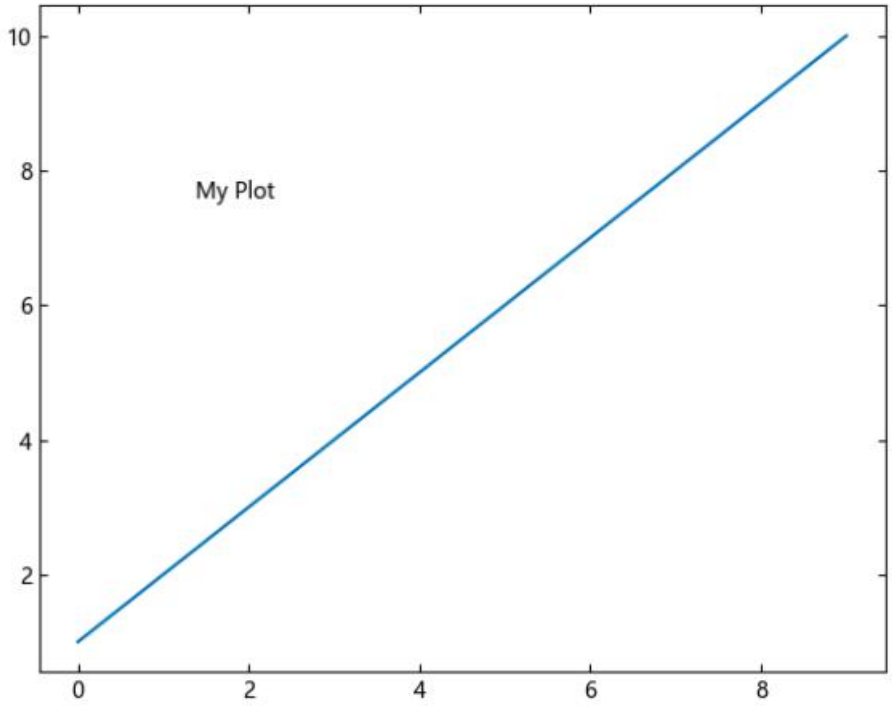


(3). 在创建后修改文本

创建简单的线图并使用鼠标将文本添加到图窗。返回创建的文本对象 t。

```
plot(1:10)
t = gtext("My Plot")
```

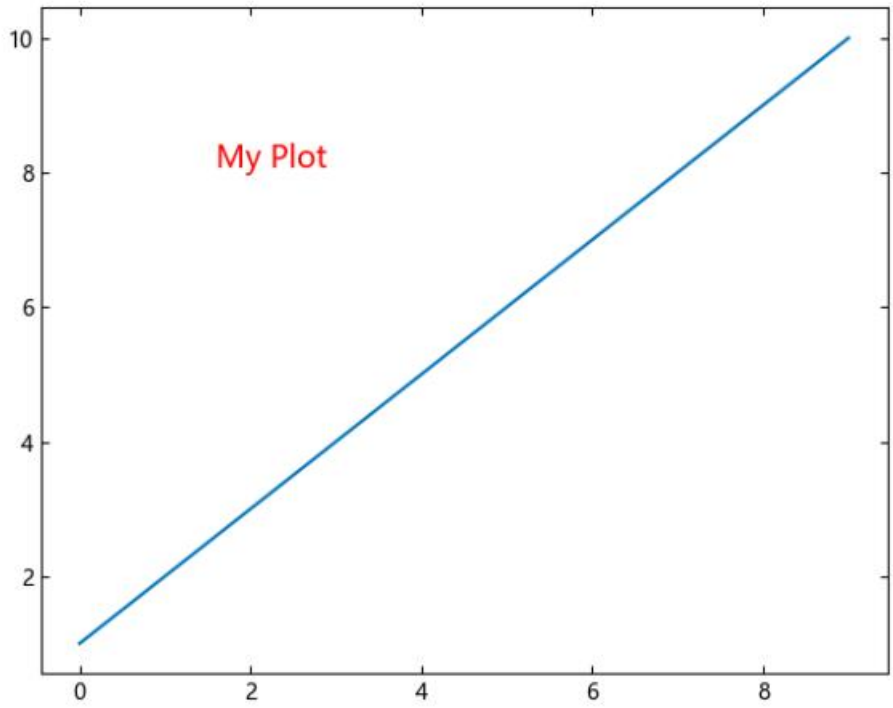
点击图窗以放置文本并创建文本对象。



使用 t 通过设置文本属性更改字体大小和文本颜色。使用圆点表示法设置

属性。

```
t.set_color("red")
t.set_fontsize(14)
plt_update()
```



xline

具有常量 x 值的垂直线

A. 语法

```
xline(xvalue)
xline(___,Key=Value)
xline(ax,___)
xl = xline(___)
```

B. 说明

xline(xvalue)在当前坐标区中的指定 x 值处创建一条常量垂直线。例如，xline(2)在 x=2 处创建一条线。

xline(___,Key=Value)使用一个或多个名称-值对组参数指定常量线属性。

xline(ax,___)在指定的坐标区（而不是当前坐标区）中创建线条。要确定当前坐标区，请使用 gca。

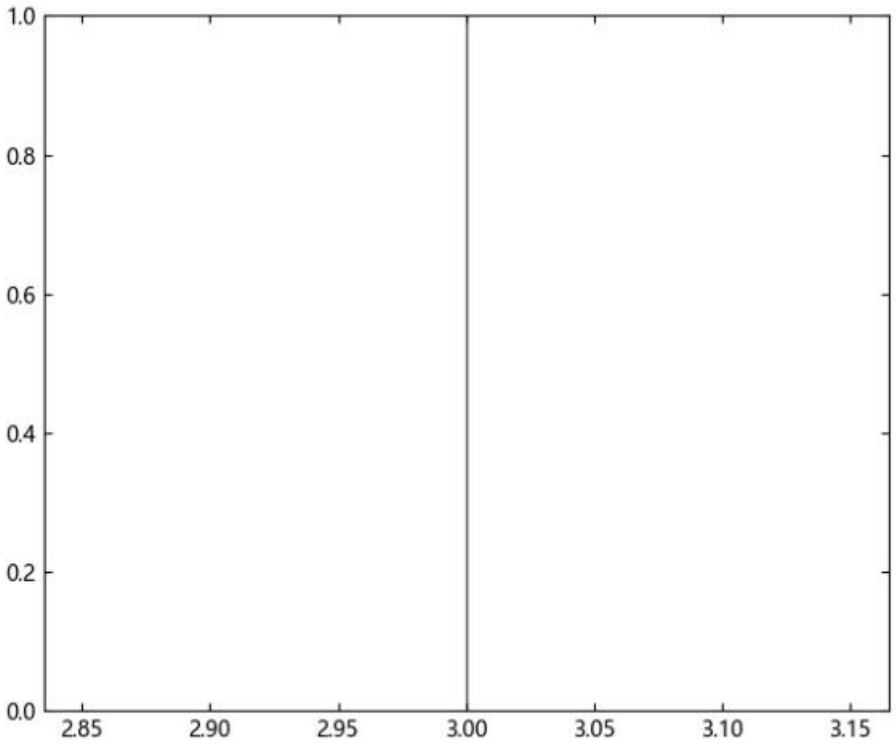
xl = xline(___) 返回创建的 ConstantLine 对象。使用 xl 修改或查询所创建的线条的属性。

C. 示例

(1). 创建垂直线

在 x = 3 处创建一条垂直线。

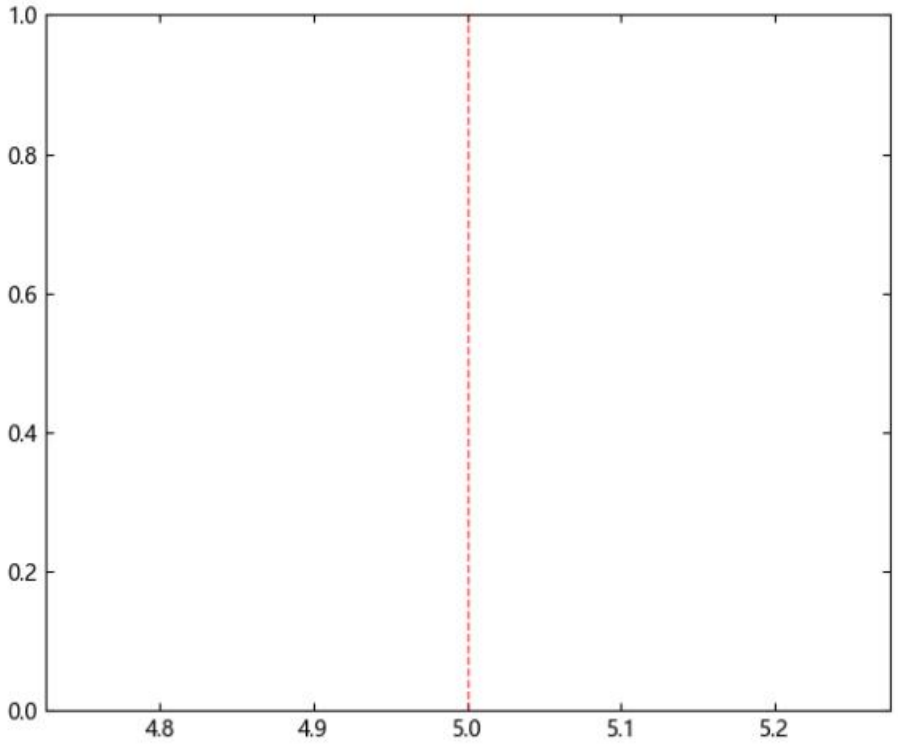
```
xline(3);
```



(2). 指定垂直线的线型和颜色

在 x = 5 处创建一条红色虚线。

```
xline(5, linestyle="--", color="r");
```

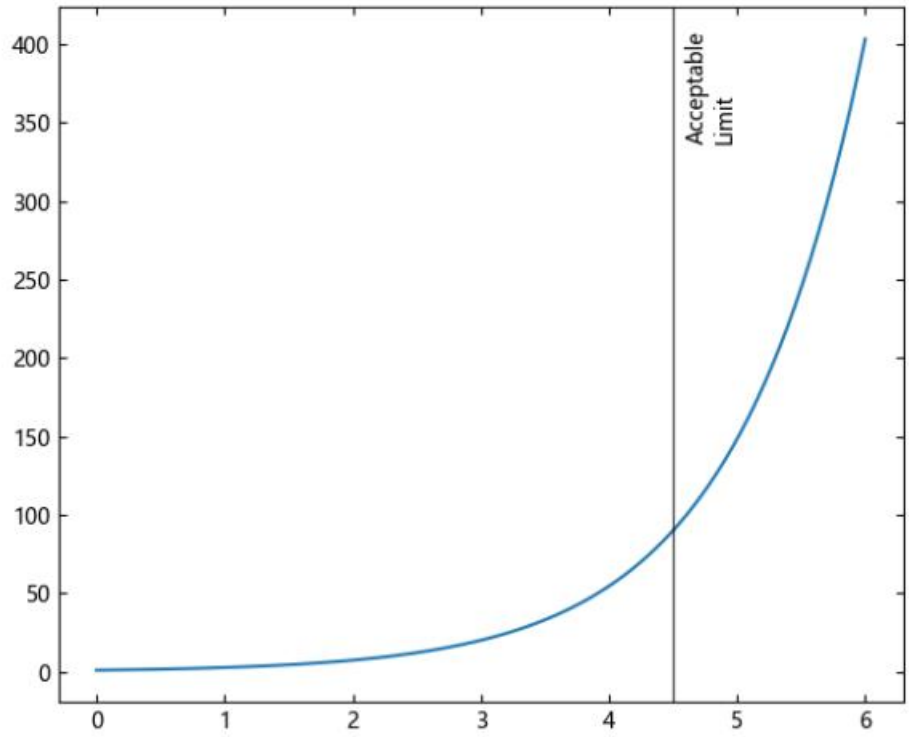


(3). 创建具有标签的垂直线

要创建一条具有标签的线条，还必须指定线型。默认线型是实线 "-"。

```
x = LinRange(0, 6, 100);
y = exp.(x);
plot(x, y)
xline(4.5; linestyle="-");

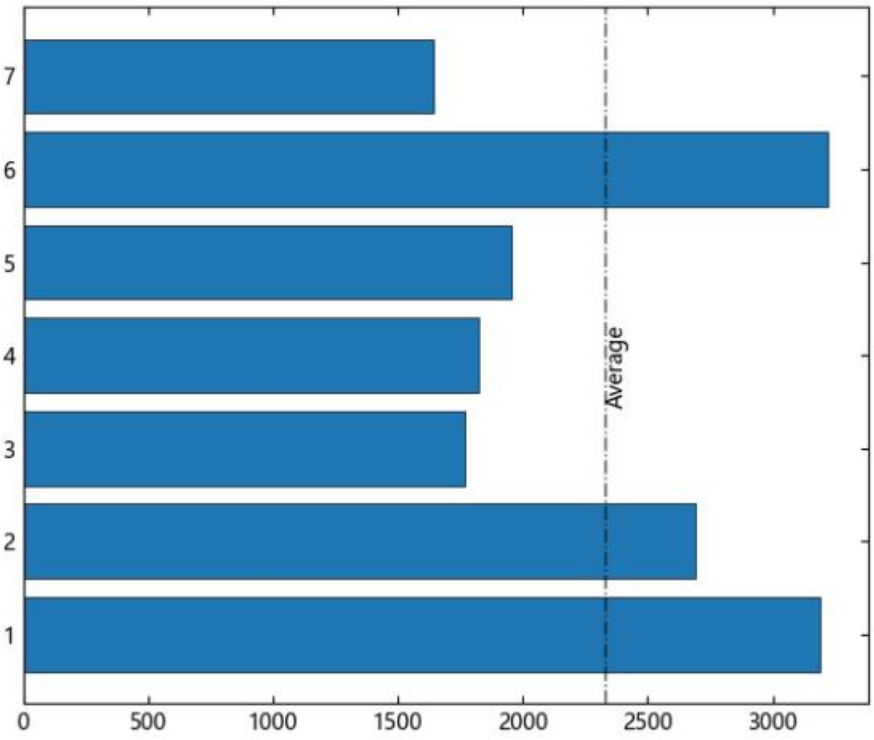
text(4.5 * 1.02, 340, "Acceptable\nLimit"; rotation=90)
```



(4). 指定垂直线属性

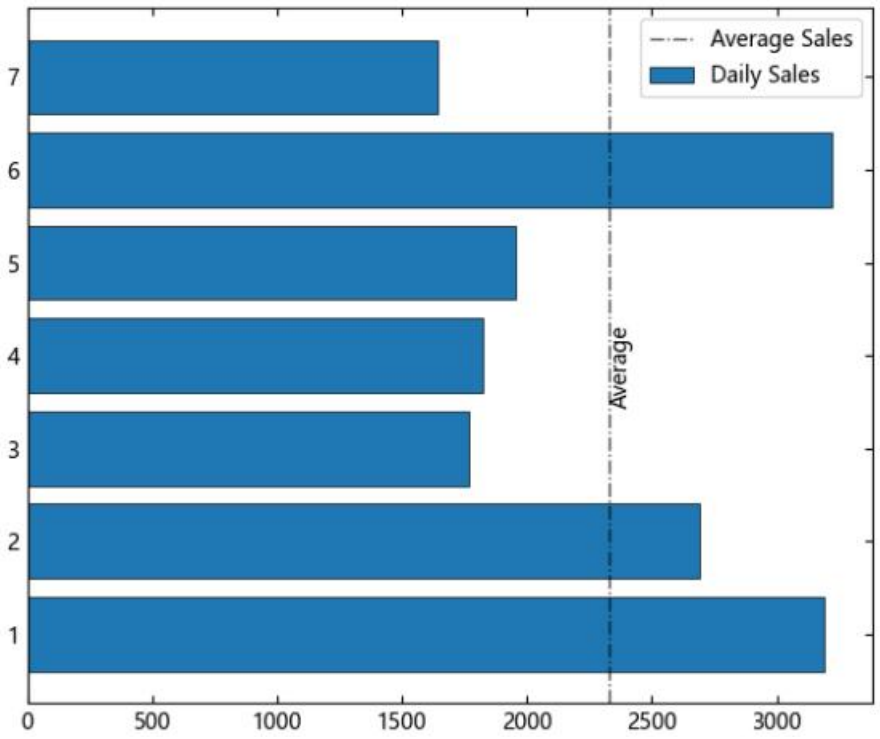
为图例创建一条既具有标签又具有显示名称的垂直点划线。

```
y = [3187 2693 1771 1826 1958 3222 1645];
barh(y)
xl = xline(2328, linestyle="-.");
text(2328, 3.5, "Average", rotation=90)
```



调整该线条的垂直和水平对齐，然后显示图例。

```
barh(y, labels=["Daily Sales"])
xl = xline(2328, linestyle="-.", label="Average Sales");
legend()
```



(5). 在特定坐标区中绘制垂直线

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，创建坐标区对象 ax1 和 ax2。然后在每个坐标区中绘图。通过将坐标区传递给 xline 函数，为每个图添加一条垂直虚线和标签。

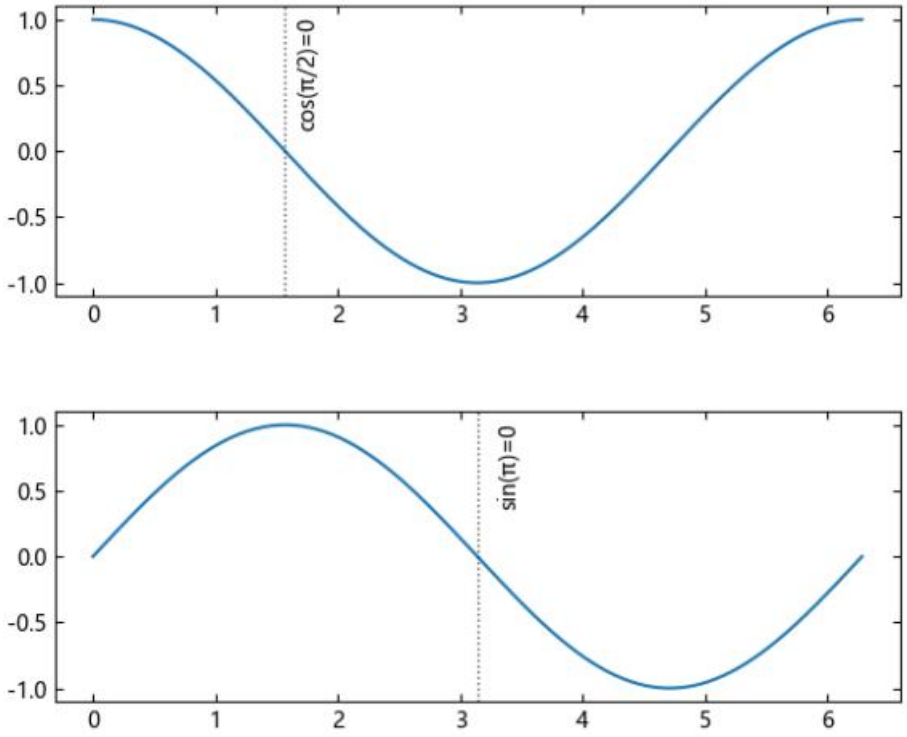
```
ax1 = subplot(2, 1, 1);
x = LinRange(0, 2 * pi, 200);
y1 = cos.(x);
plot(ax1, x, y1)

ax2 = subplot(2, 1, 2);
x = LinRange(0, 2 * pi, 200);
y2 = sin.(x);
plot(ax2, x, y2)

xlcosine = xline(ax1, pi / 2; linestyle=":");
xlsine = xline(ax2, pi; linestyle=":");

ax1.text(pi / 2 * 1.05, 0.2, "cos(π/2)=0"; rotation=90)
```

```
ax2.text(pi * 1.05, 0.4, "sin(π)=0"; rotation=90)
```



yline

具有常量 y 值的水平线

A. 语法

```
yline(yvalue)
yline(___,Key=Value)
yline(ax,___)
yl = yline(___)
```

B. 说明

yline(yvalue)在当前坐标区中的指定 y 值处创建一条常量水平线。例如，yline(6)在 y=6 处创建一条线。

yline(___,Key=Value)使用一个或多个名称-值对组参数指定常量线属性。

yline(ax,___)在指定的坐标区（而不是当前坐标区）中创建线条。要确定当前坐标区，请使用 gca。

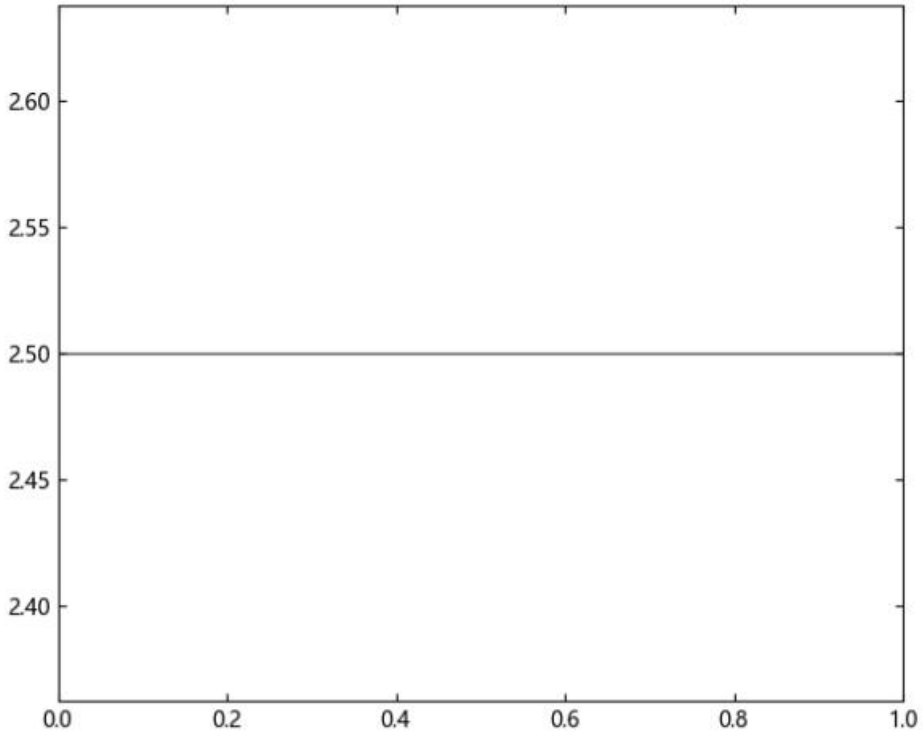
yl = yline(___) 返回创建的 ConstantLine 对象。使用 yl 修改或查询所创建的线条的属性。

C. 示例

(1). 创建水平线

在 y = 2.5 处创建一条水平线。

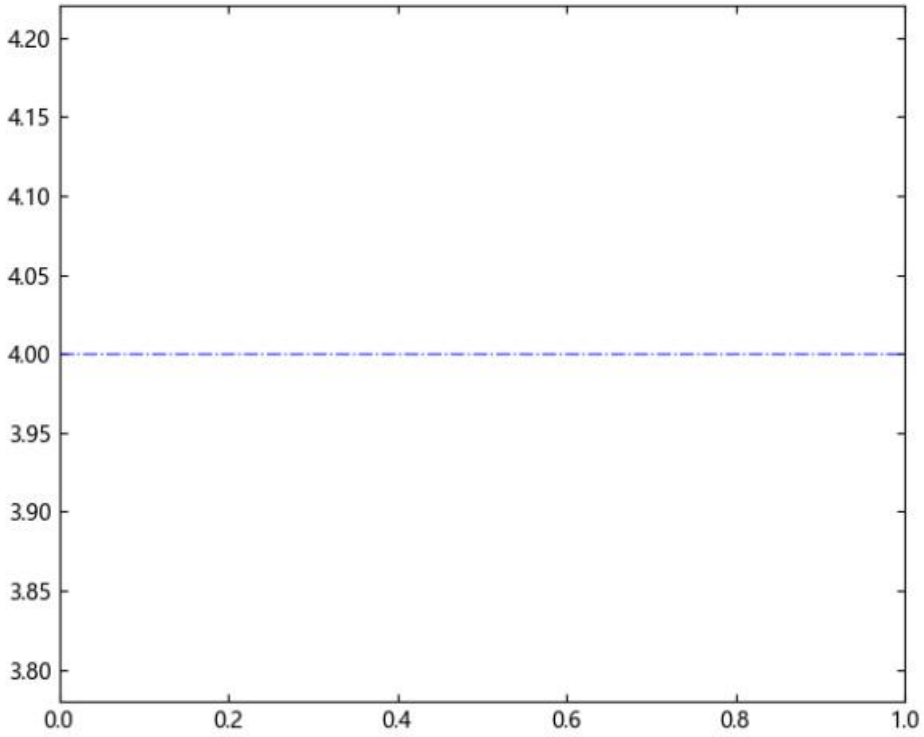
```
yline(2.5);
```



(2). 指定水平线的线型和颜色

在 y = 4 处创建一条蓝色点划线。

```
yline(4, linestyle="-.", color="b");
```

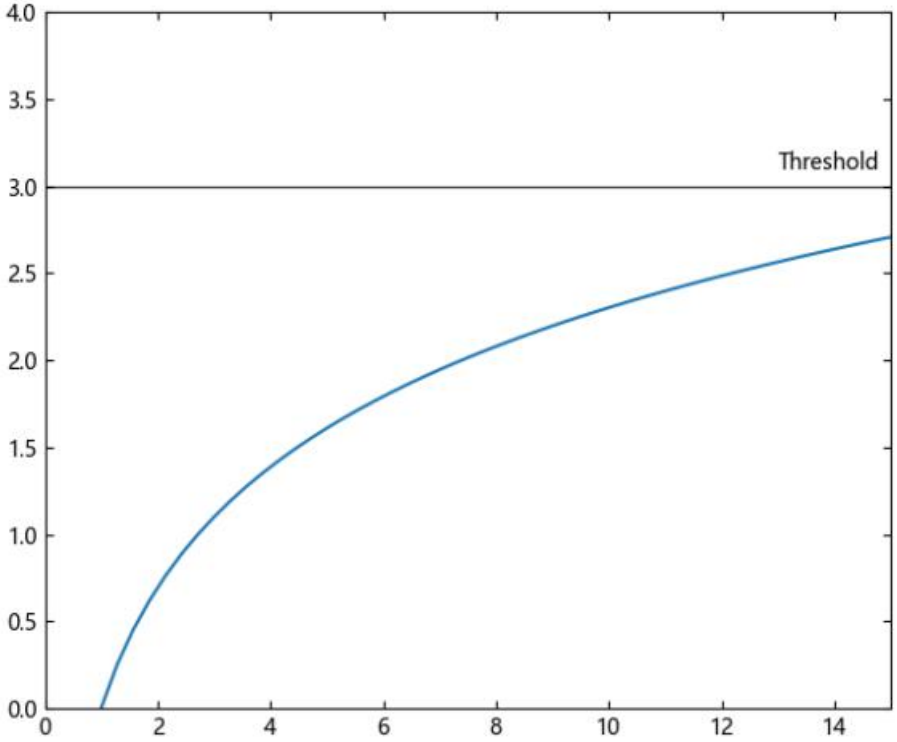


(3). 创建具有标签的水平线

要创建一条具有标签的线条，还必须指定线型。默认线型是实线 "-"。

```
x = LinRange(1, 15, 50);
y = log.(x);
plot(x, y)
xlim([0 15]);
ylim([0 4]);
yline(3; linestyle="-");

text(13, 3 + 0.1, "Threshold")
```

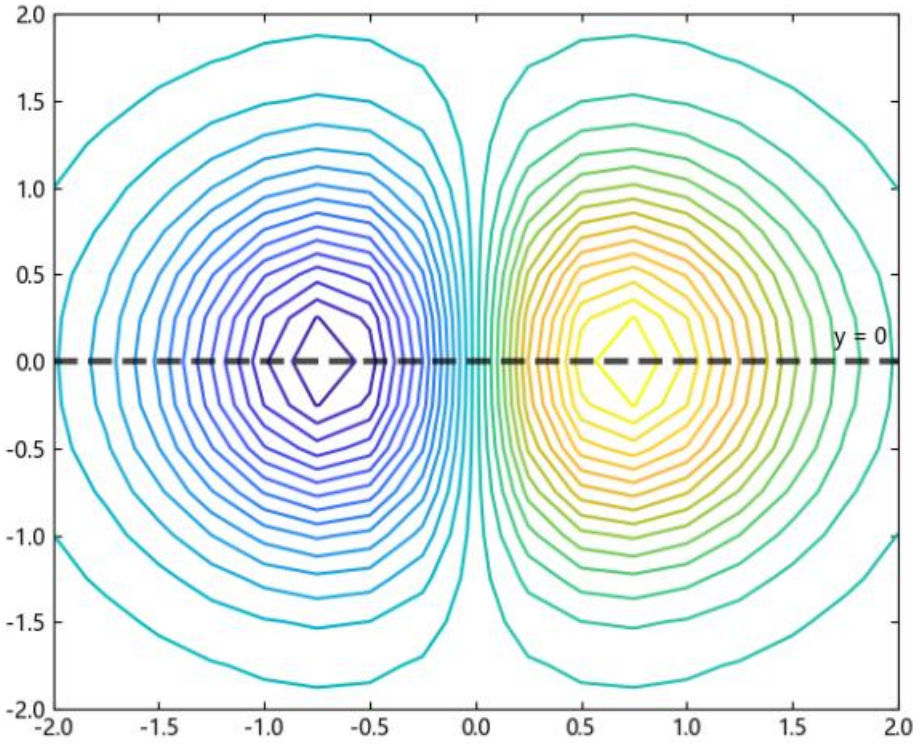


(4). 指定水平线属性

创建一条具有标签和指定线宽的水平虚线。

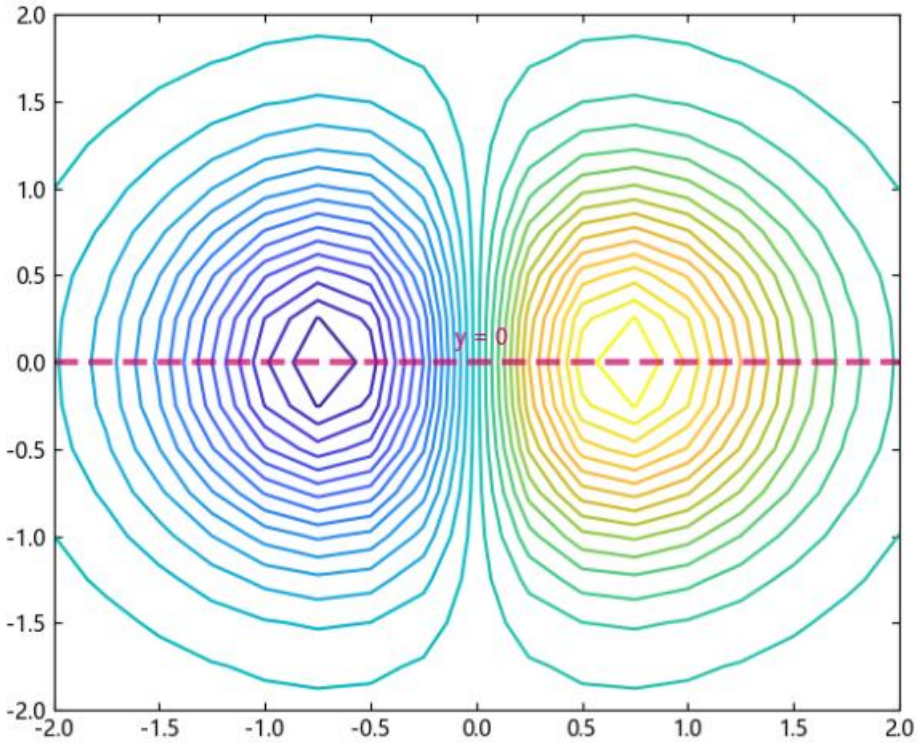
```
x = -2:0.25:2;
X, Y = meshgrid2(x, x)
Z = X .* exp.(-X .^ 2 - Y .^ 2);
contour(X, Y, Z, 30)

y1 = yline(0; linestyle="--", linewidth=3)
txt = text(1.7, 0.1, "y = 0")
```



调整线条的水平对齐和颜色。

```
y1.set_color([.80, 0, .40])
txt.set_position((-0.1,0.1))
txt.set_color([.80, 0, .40])
```



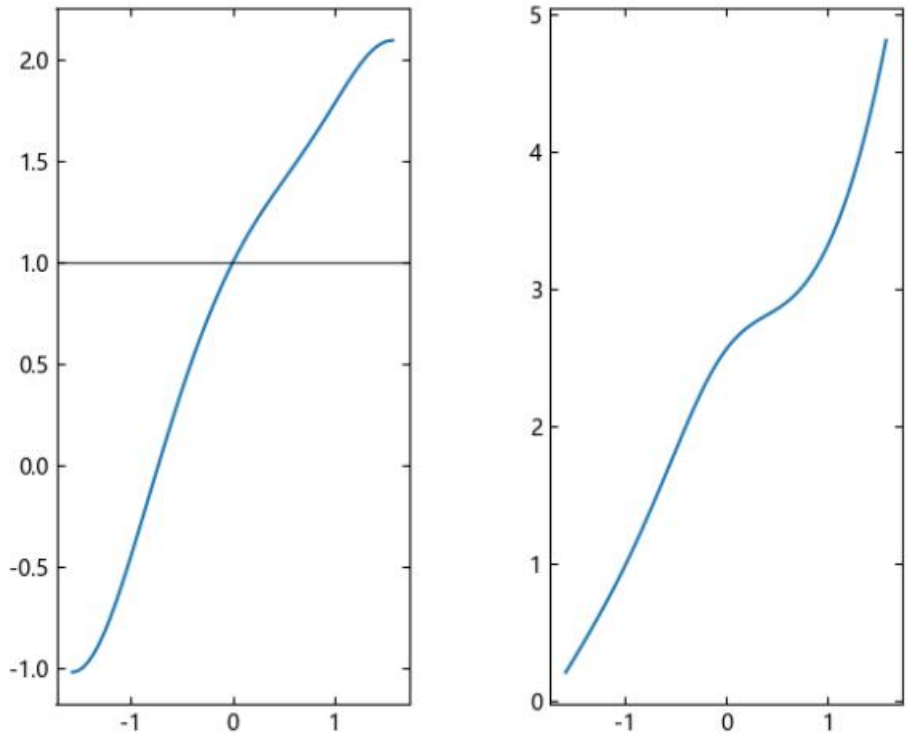
(5). 在特定坐标区中绘制水平线

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，创建坐标区对象 ax1 和 ax2。然后在每个坐标区中绘图。通过将 ax1 传递给 yline 函数，在第一个图中添加一条水平线。

```
ax1 = subplot(1,2,1);
x = -pi/2:pi/60:pi/2;
y1 = tan.(sin.(x)) + cos.(sin.(x));
plot(ax1,x,y1)

ax2 = subplot(1,2,2);
x = -pi/2:pi/60:pi/2;
y2 = tan.(cos.(x)) + exp.(x);
plot(ax2,x,y2)

yline(ax1, 1);
```



annotation

创建注释

A. 语法

```
annotation(lineType,x,y)
annotation(lineType)

annotation(shapeType,dim)
annotation(shapeType)

annotation(___,Key=Value)

an = annotation(___)
```

B. 说明

annotation(lineType,x,y)创建一个在当前图窗中的两个点之间延伸的线条或箭头注释。将 lineType 指定为"line"、"arrow"、"doublearrow"或"textarrow"。将 x 和 y 分别指定为[x_beginx_end]和[y_beginy_end]形式的二元素向量。

annotation(lineType) 在点 (0.3,0.3) 和 (0.4,0.4) 之间的默认位置创建注释。

annotation(shapeType,dim)在当前图窗中创建具有特定大小和位置的矩形、椭圆或文本框注释。将 shapeType 指定为"rectangle"、"ellipse"或"textbox"。指定 dim 作为[xywh]形式的四元素向量。x 和 y 元素确定位置，w 和 h 元素确定大小。

annotation(shapeType) 会在默认位置创建注释，以使左下角位于 (0.3,0.3) 且宽度和高度均为 0.1。

annotation(___,Key=Value)用于创建注释并将属性指定为名称-值对组参数。不同类型的注释支持不同属性。您可以使用上述语法中的任意输入参数组合指定属性。

an=annotation(___)返回注释对象。返回的对象的类型取决于第一个输入参数。创建对象之后，可使用 an 修改其属性。可以使用上述任意语法指定输出参数。

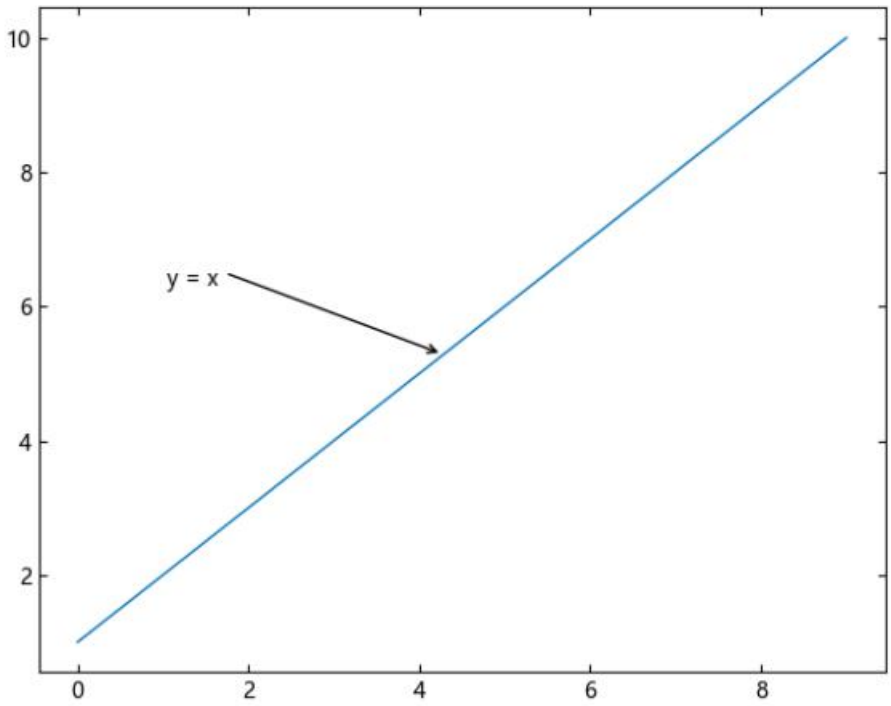
C. 示例

(1). 创建文本箭头注释

创建一个简单线图并向图窗添加文本箭头。用归一化的图窗坐标指定文本箭头位置，起点为 (0.3,0.6)，终点为 (0.5,0.5)。通过设置 string 属性指定文本说明。

```
plot(1:10)
x = [0.3 0.5];
y = [0.6 0.5];
```

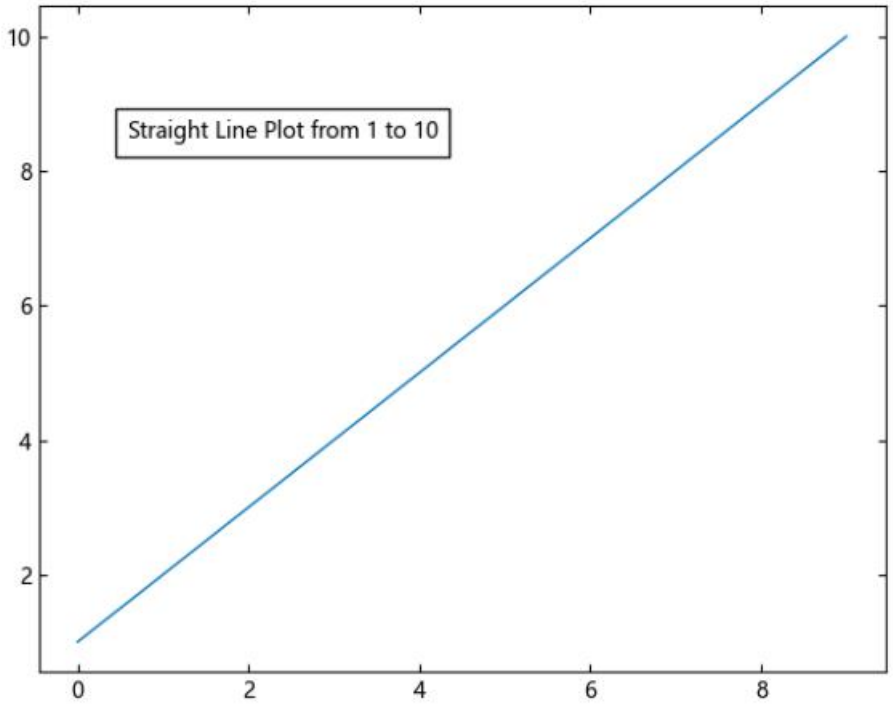
```
annotation("textarrow", x, y, string ="y = x ")
```



(2). 创建文本框注释

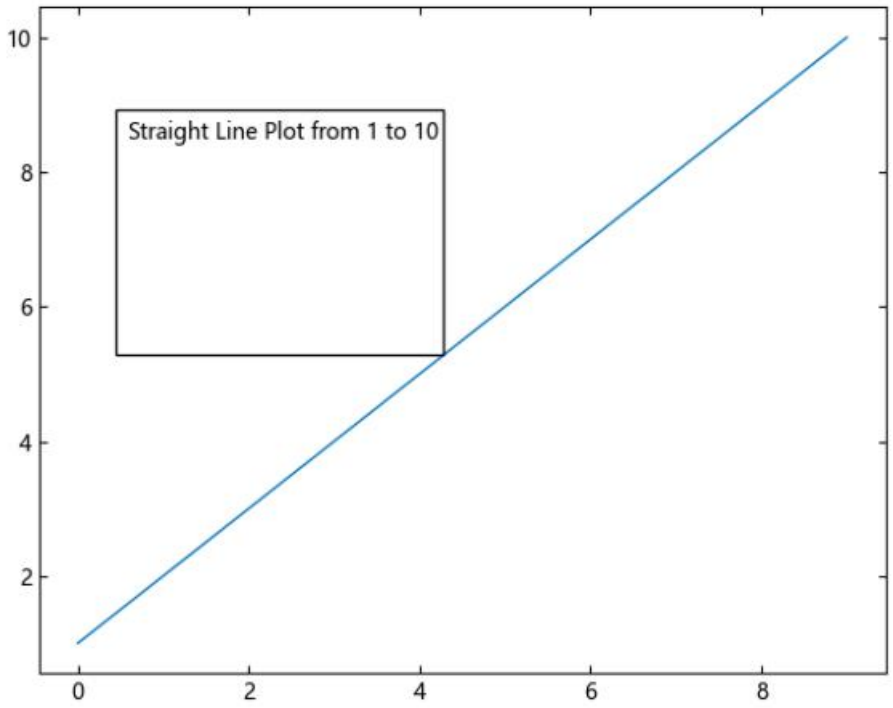
创建一个简单线图并向图窗添加文本框注释。通过设置 string 属性指定文本说明。通过将 fitboxtotext 属性设置为 "on"，强制使文本框紧贴文本。

```
figure()
plot(1:10)
dim = [.2 .5 .3 .3];
str = "Straight Line Plot from 1 to 10";
annotation("textbox", dim, string=str, fitboxtotext="on");
```



创建文本框注释而不设置 fitboxtotext 属性。文本框使用指定的宽度和高度，并根据需要进行文本换行。

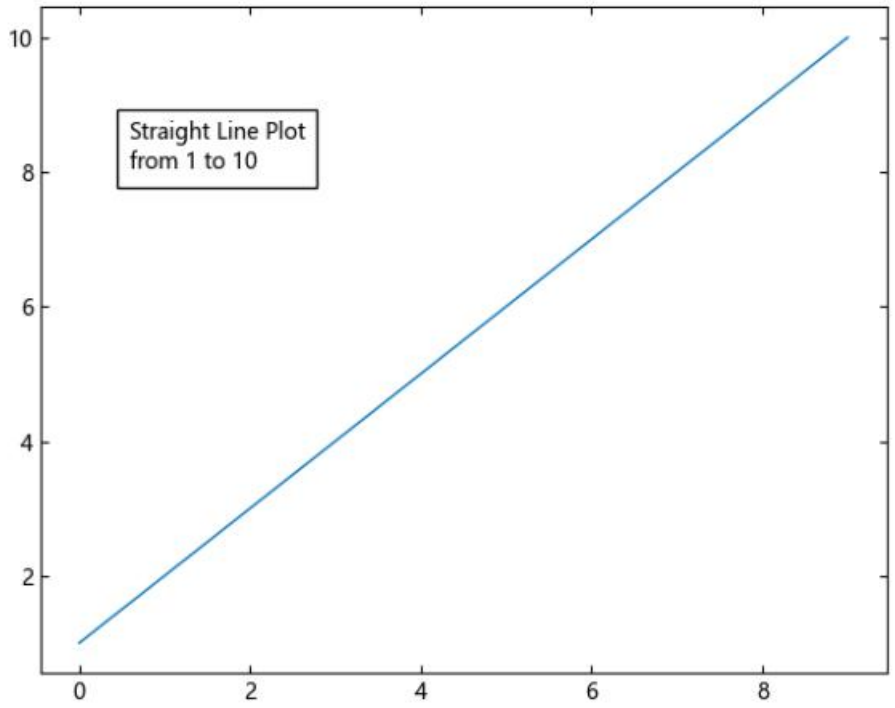
```
figure()
plot(1:10)
dim = [.2 .5 .3 .3];
str = "Straight Line Plot from 1 to 10";
annotation("textbox", dim, string=str);
```



(3). 创建包含多行文本的文本框注释

通过将 string 属性设置为字符串矩阵，创建包含多行文本的文本框注释。字符串矩阵的每个元素显示在单独的行上。通过将 fitboxtotext 属性设置为 "on"，强制使文本框紧贴文本。

```
figure()
plot(1:10)
dim = [.2 .5 .3 .3];
str = ["Straight Line Plot" "from 1 to 10"];
annotation("textbox", dim, string=str, fitboxtotext="on");
```

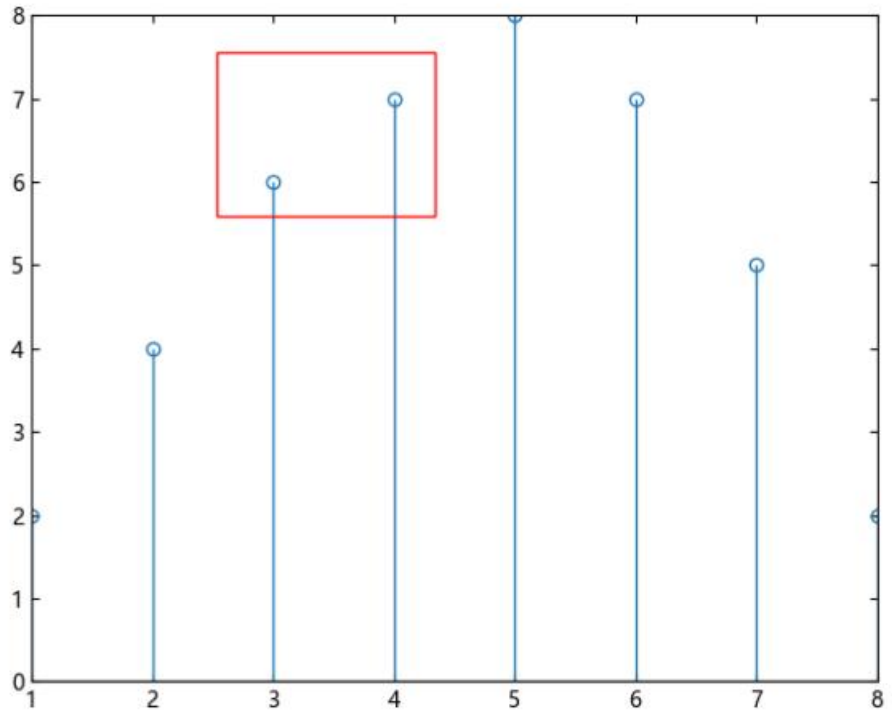


(4). 创建矩形注释

创建一个针状图并向图窗添加矩形注释。通过指定 Color 属性更改矩形轮廓的颜色。

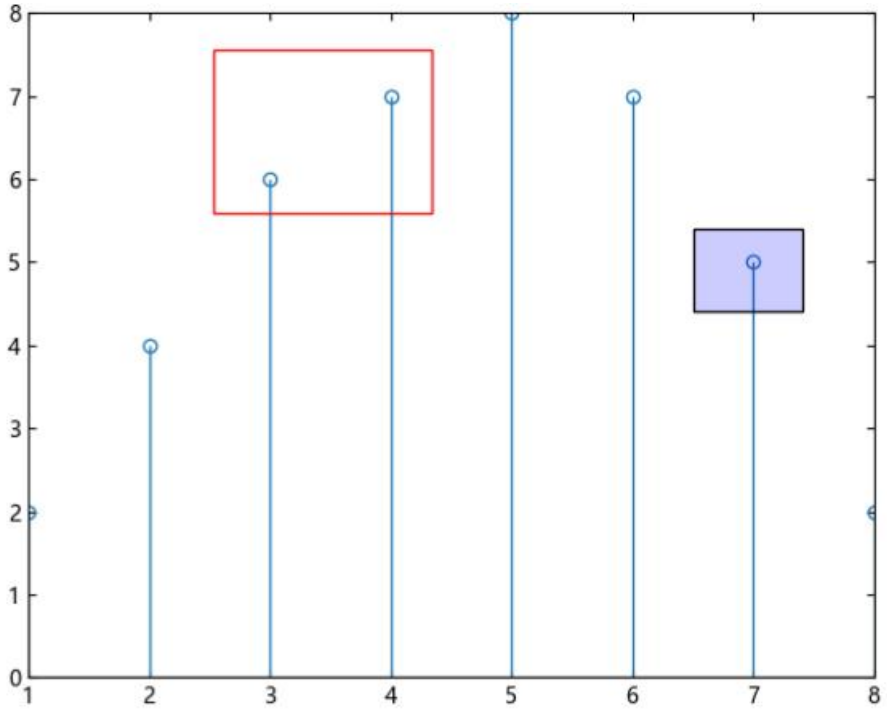
```
figure()
data = [2 4 6 7 8 7 5 2];
stem(1:length(data), data)
xlim([1,8])
ylim([0,8])

dim = [.3 .68 .2 .2];
annotation("rectangle", dim, edgecolor="red");
```



向图窗添加另一个矩形注释。通过设置 facecolor 属性指定填充颜色。通过将 facealpha 属性设置为 0（完全透明）到 1（完全不透明）之间的值添加透明度。

```
dim2 = [.74 .56 .1 .1];
annotation("rectangle", dim2, facecolor="blue", facealpha=0.2);
```



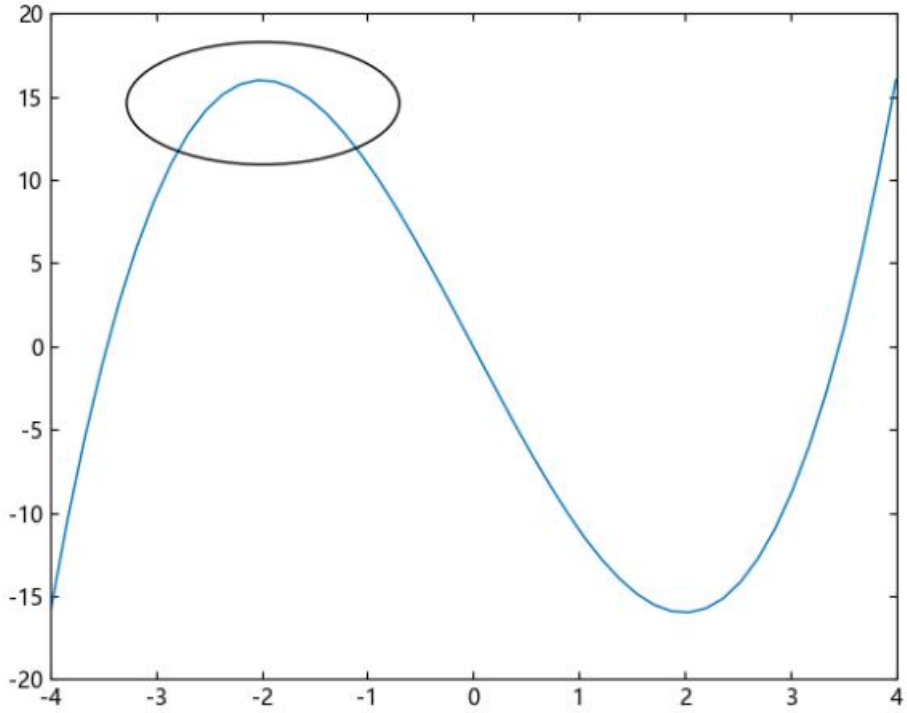
(5). 创建椭圆注释

创建一个简单线图并向图窗添加椭圆注释。将 dim 指定为包围椭圆的最小矩形的尺寸和位置。

```
figure()
x = LinRange(-4, 4, 50);
y = x .^ 3 - 12 * x;
plot(x, y)

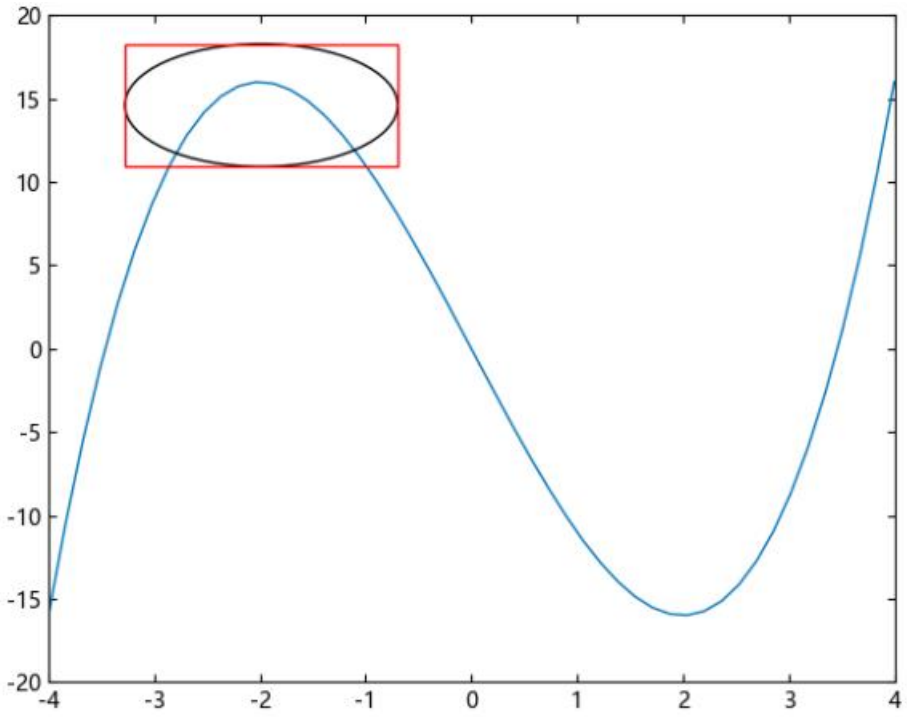
dim = [0.2 0.74 0.25 0.15];
annotation("ellipse", dim);

xlim([-4, 4])
ylim([-20, 20])
```



使用相同的尺寸绘制一个红色矩形，以显示椭圆如何填充矩形区域。

```
annotation("rectangle", dim; edgecolor="red")
```



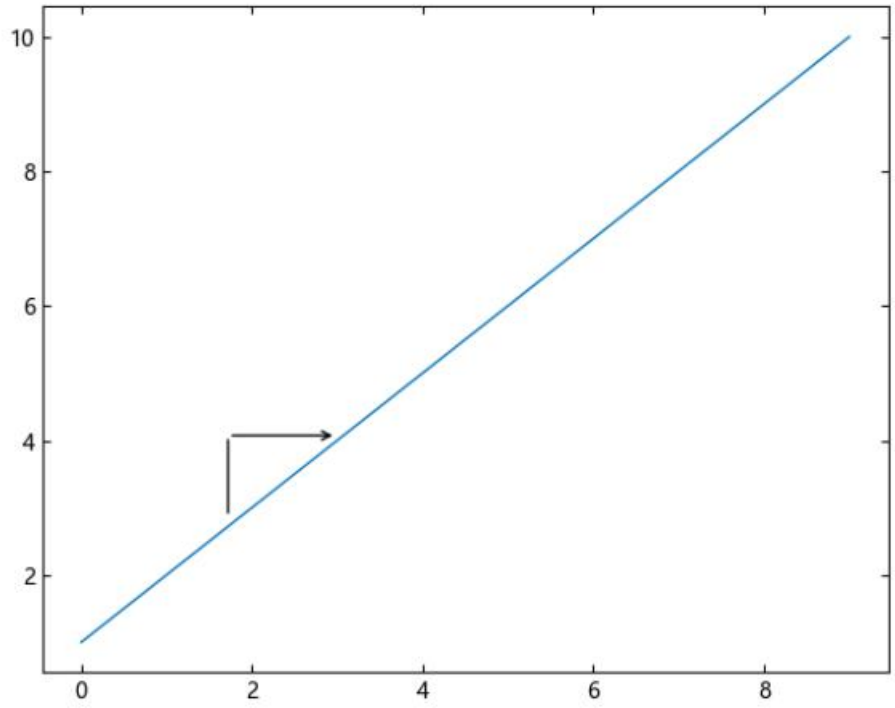
(6). 组合使用两种类型的注释

创建一个简单的线图。然后，通过组合使用线条和箭头注释向图形添加一个弯曲箭头。

```
figure()
plot(1:10)

xl = [.3 .3];
yl = [.3 .4];
annotation("line", xl, yl);

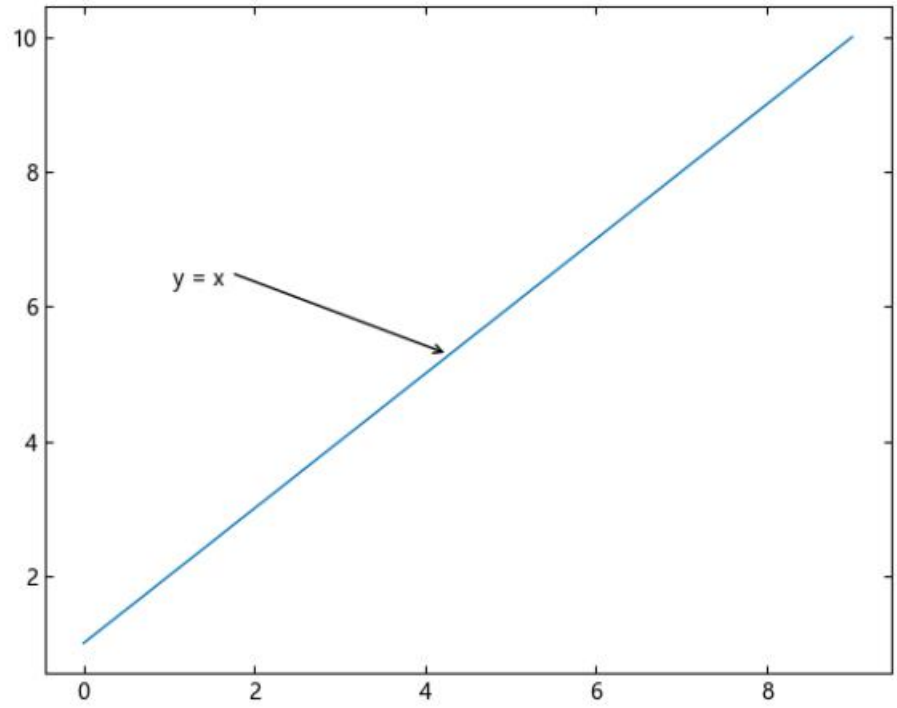
xa = [.3 .4];
ya = [.4 .4];
annotation("arrow", xa, ya);
```



(7). 创建后修改注释

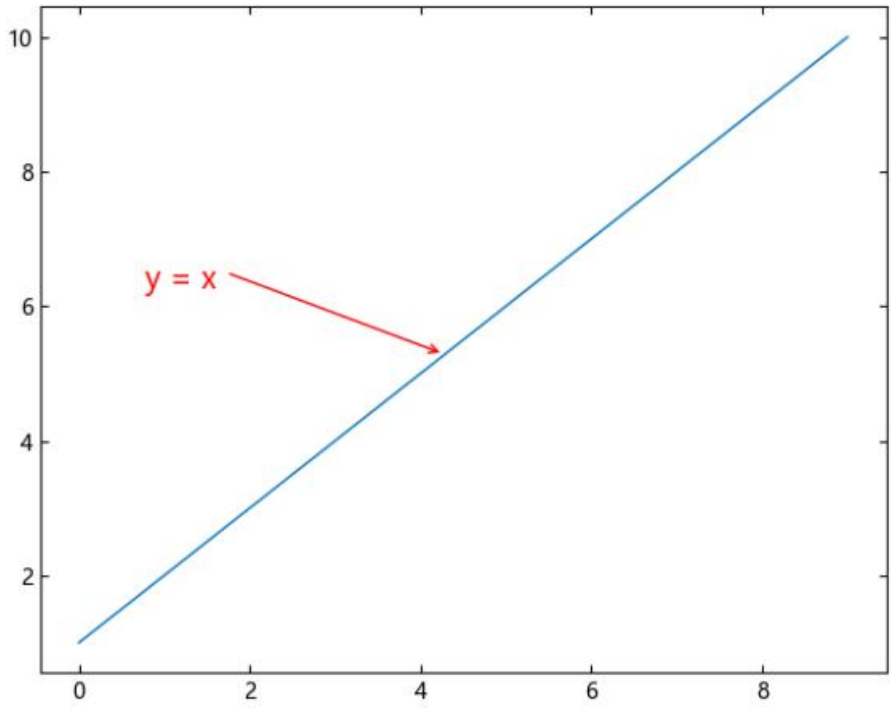
向图窗添加一个文本箭头，并返回注释文本箭头对象 a。

```
figure()
plot(1:10)
x = [0.3,0.5];
y = [0.6,0.5];
a = annotation("textarrow",x,y,string="y = x ");
```



使用 a 修改注释文本箭头的属性。例如，将颜色更改为红色，将字体大小更改为 14 磅。

```
a.set_color("red");
a.set_fontsize(14);
```



datacursormode

开启数据提示

显示数据点的坐标。

A. 语法

```
datacursormode(target)
datacursormode(___,Key=Value)
dt = datacursormode(___)
```

B. 说明

datacursormode(target)为当前图窗中的图启用数据提示。

datacursormode(___,Key=Value)为指定图窗中的图设置数据提示。

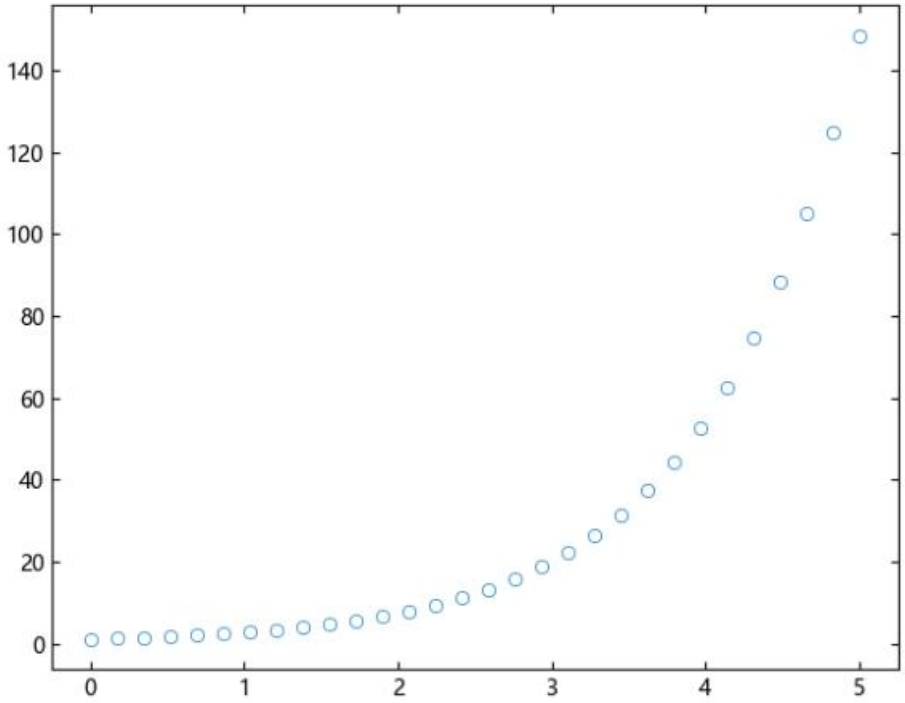
dt = datacursormode(___)为指定图窗返回一个数据提示对象。

C. 示例

(1). 在可移动窗口中显示数据提示

绘制一些数据，并启用数据游标模式。

```
x = LinRange(0, 5, 30);
y = exp.(x);
scatter(x, y)
dcm = datacursormode();
```



datatip

创建数据提示

A. 语法

```
datatip(target)
datatip(target,x,y)
datatip(target,x,y,z)
datatip(___,Key=Value)
dt = datatip(___)
```

B. 说明

datatip(target) 在指定图（如线图或散点图）的第一个绘图数据点上创建数据提示。

datatip(target,x,y)在 x 和 y 指定的二维绘图数据点上创建数据提示。如果指定近似坐标，则 datatip 会在最近的数据点上创建数据提示。

datatip(target,x,y,z) 在 x、y 和 z 指定的三维绘图数据点上创建数据提示。如果指定近似坐标，则 datatip 会在最近的数据点上创建数据提示。

datatip(___,Key=Value) 使用一个或多个名称-值对组参数指定属性。请在所

有其他参数后指定名称-值对组。

dt = datatip(___) 返回 Annotation 对象。您可以在上述语法中指定任何输入参数组合。此语法可用于控制数据提示的属性。

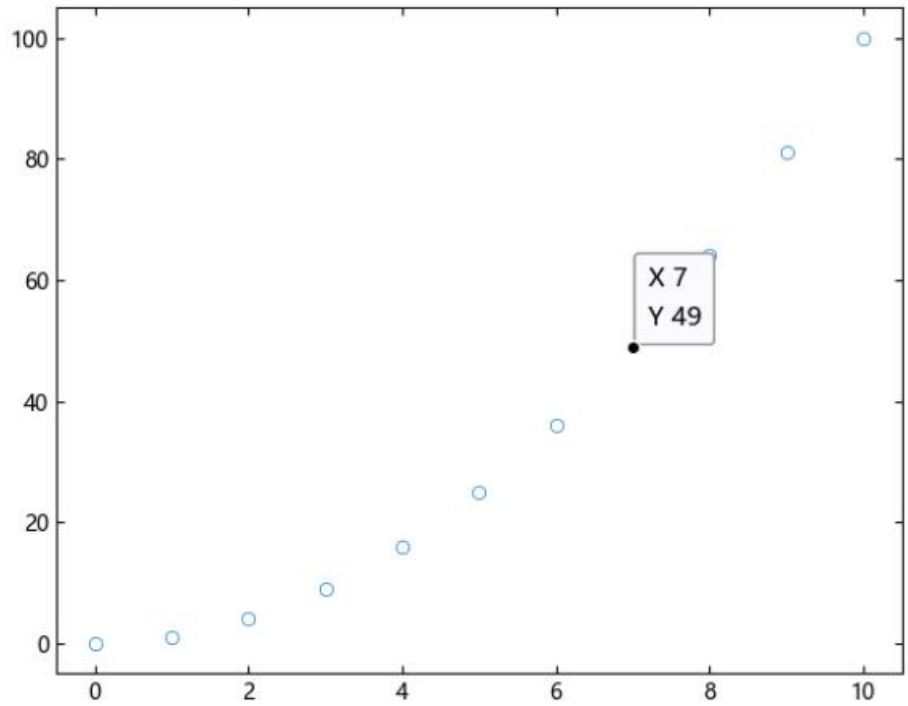
C. 示例

(1). 使用坐标创建数据提示

在散点图上绘制数据，并返回 Scatter 对象。然后，通过指定 x 和 y 坐标，在特定数据点上创建数据提示。

```
x = LinRange(0, 10, 11);
y = x .^ 2;
sc = scatter(x, y);

dt = datatip(sc, 7, 49);
```

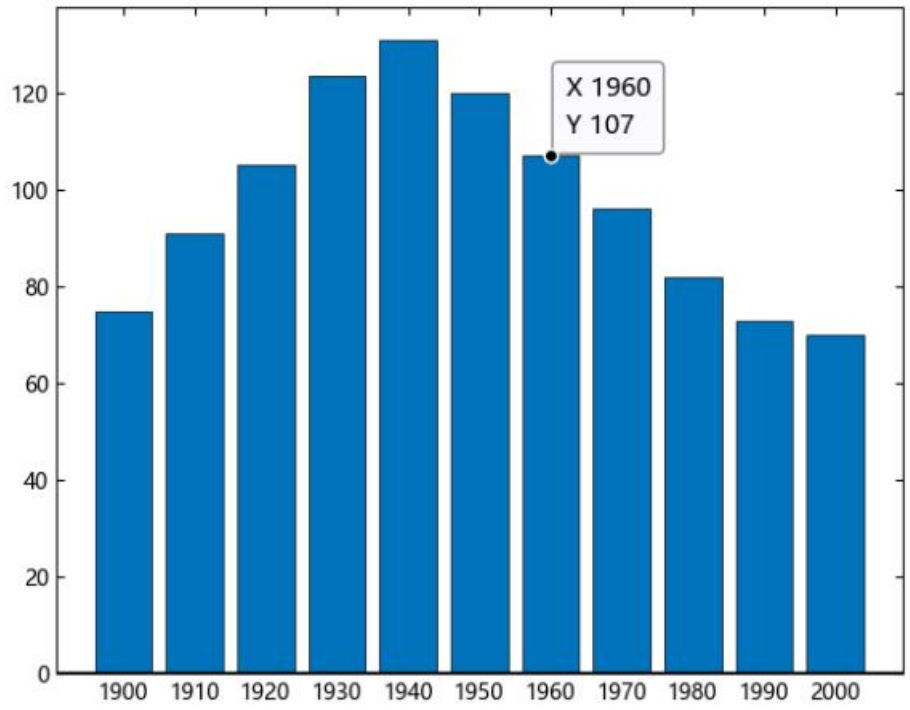


(2). 使用索引创建数据提示

在条形图上绘制数据，并返回 Bar 对象。然后，通过指定 dataindex 名称-值对组参数，在第七个条形上创建数据提示。

```
x = 1900:10:2000;
y = [75 91 105 123.5 131 120 107 96 82 73 70];
b = bar(x, y);

dt = datatip(b; dataindex=7);
```

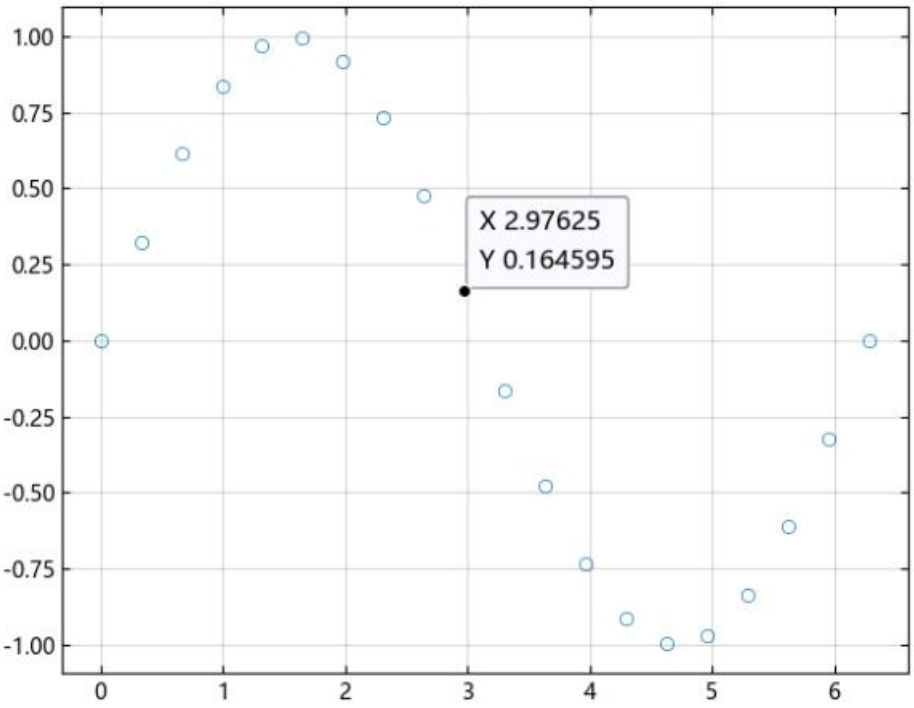


(3). 清除坐标区并重置所有坐标区属性

在散点图上绘制数据，并返回 Scatter 对象。通过将 x = 3 和 y = 0 指定为近似坐标，在 [3,0] 附近的数据点上创建数据提示。使用近似坐标创建数据提示时，数据提示会出现在最近的数据点上。

```
x = LinRange(0, 2 * pi, 20);
y = sin.(x);
sc = scatter(x, y);
grid("on");

dt = datatip(sc, 3, 0);
```

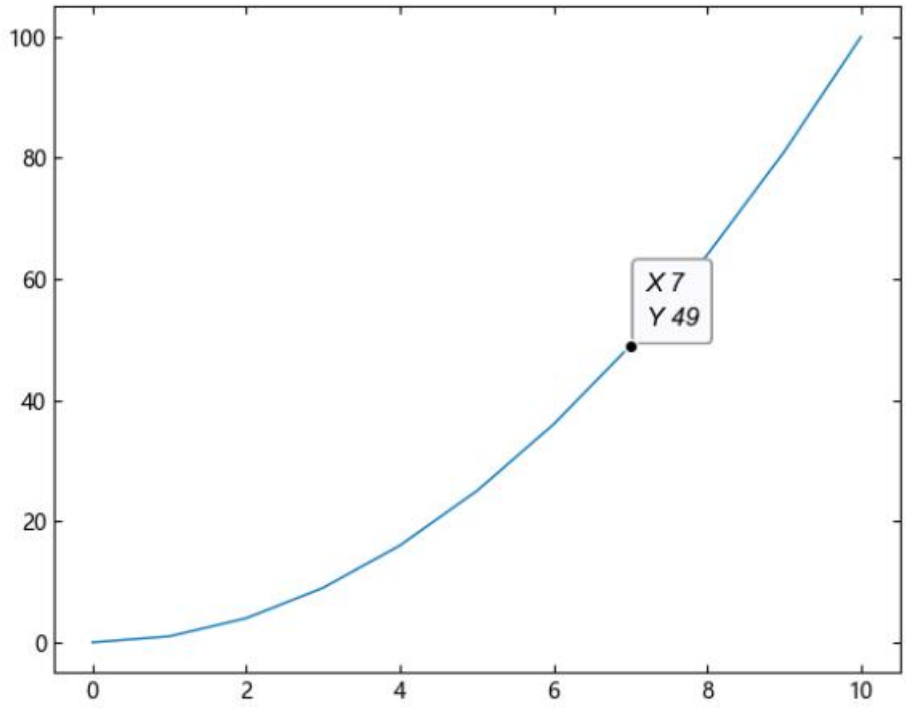


(4). 使用坐标创建数据提示-线图

在线图上绘制数据。然后，通过指定 x 和 y 坐标，在特定数据点上创建数据提示。

```
x = LinRange(0, 10, 11);
y = x .^ 2;
p = plot(x, y);

dt = datatip(p, 7, 49; fontfamily="Arial", fontsize=12,
fontstyle="italic");
```

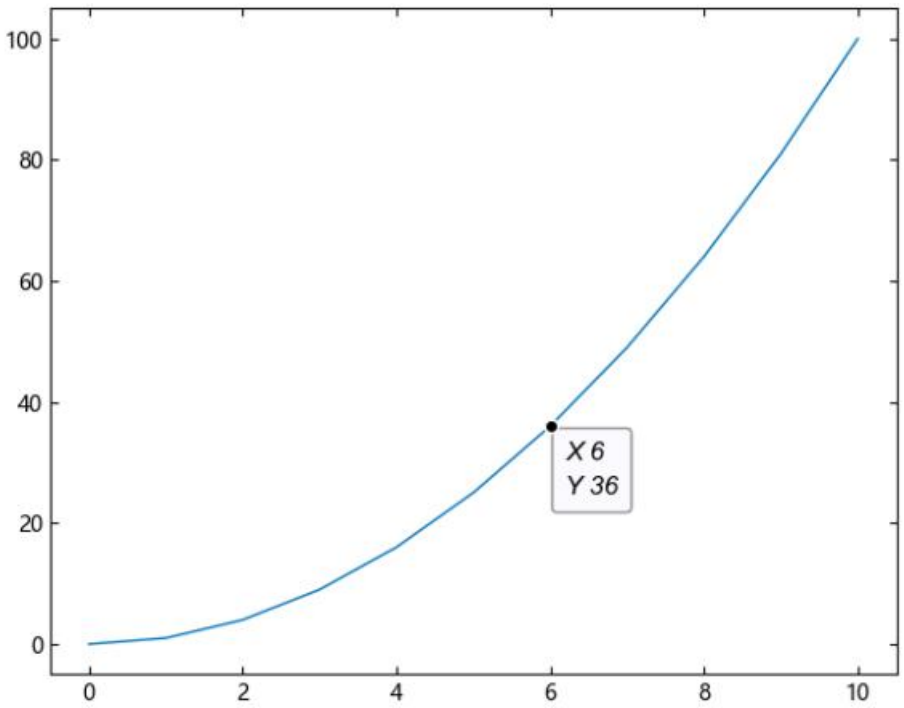


(5). 使用索引创建数据提示-线图

在线图上绘制数据。然后，通过指定 dataindex 名称-值对组参数，在第七个数据点创建数据提示。

```
x = LinRange(0, 10, 11);
y = x .^ 2;
p = plot(x, y);

dt = datatip(
    p;
    dataindex=7,
    fontfamily="Arial",
    fontsize=12,
    fontstyle="italic",
    location="southeast",
);
```

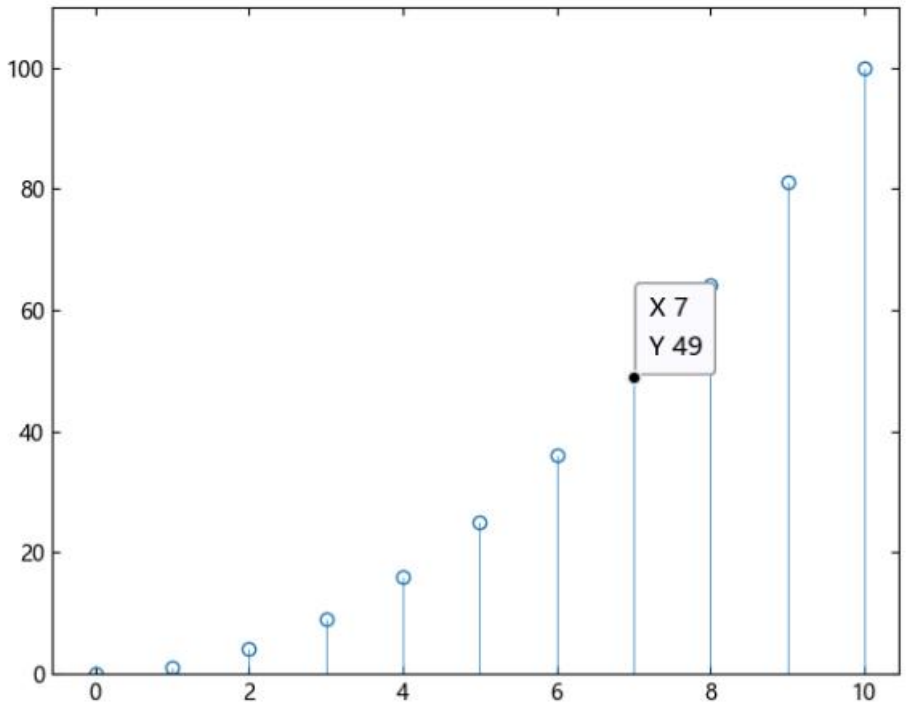


(6). 使用坐标创建数据提示-针状图

在针状图上绘制数据。然后，通过指定 x 和 y 坐标，在特定数据点上创建数据提示。

```
x = LinRange(0, 10, 11);
y = x .^ 2;
st = stem(x, y);

dt = datatip(st, 7, 49);
```

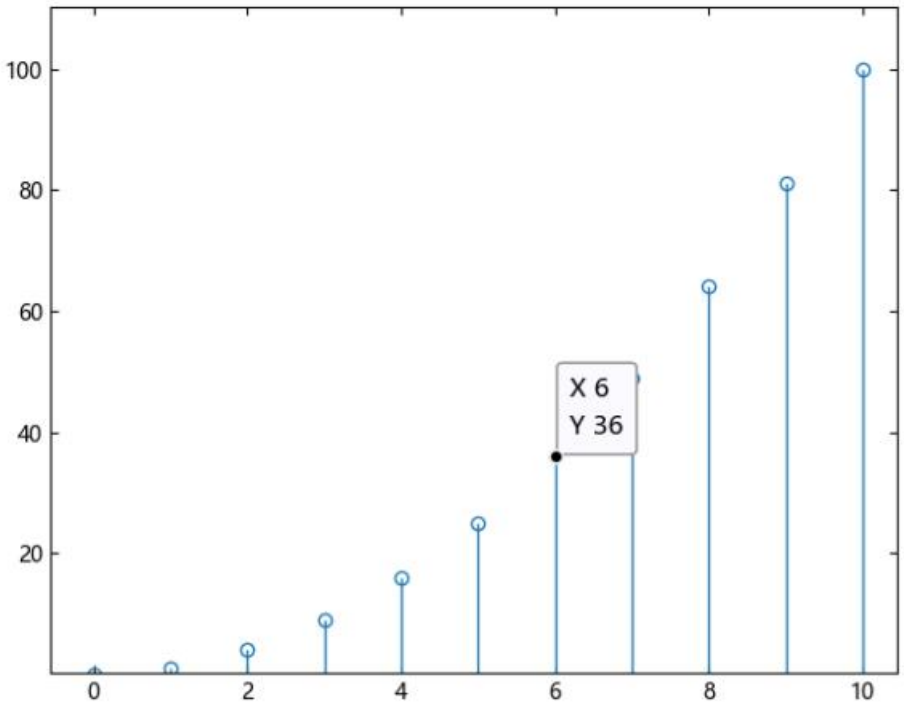


(7). 使用索引创建数据提示-针状图

在针状图上绘制数据。然后，通过指定 dataindex 名称-值对组参数，在第七个数据点创建数据提示。

```
x = LinRange(0, 10, 11);
y = x .^ 2;
st = stem(x, y);

dt = datatip(st; dataindex=7);
```

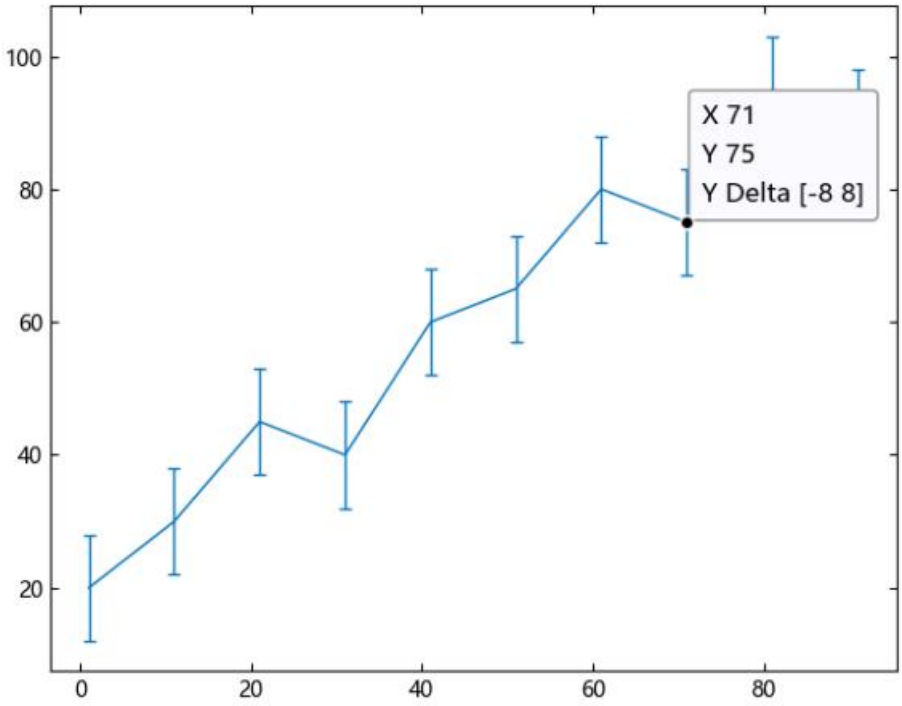


(8). 使用坐标创建数据提示-误差图

在误差图上绘制数据。然后，通过指定 x 和 y 坐标，在特定数据点上创建数据提示。

```
x = 1:10:100
y = [20, 30, 45, 40, 60, 65, 80, 75, 95, 90]
err = 8 * ones(size(y))
er = errorbar(x, y, err)

dt = datatip(er, 71, 75);
```

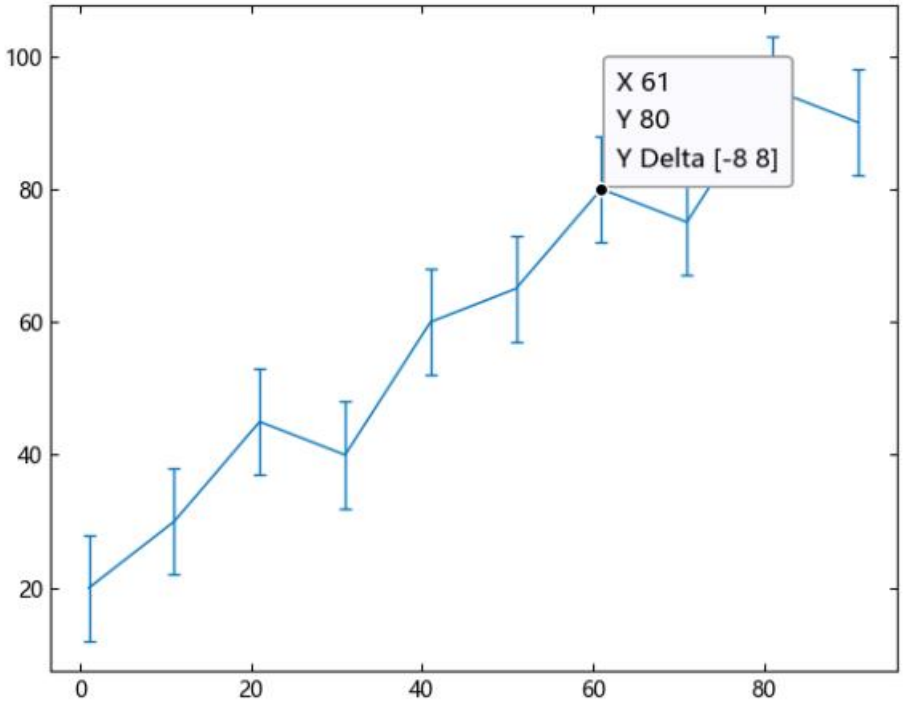


(9). 使用索引创建数据提示-误差图

在误差图上绘制数据。然后，通过指定 dataindex 名称-值对组参数，在第七个数据点创建数据提示。

```
x = 1:10:100
y = [20, 30, 45, 40, 60, 65, 80, 75, 95, 90]
err = 8 * ones(size(y))
er = errorbar(x, y, err)

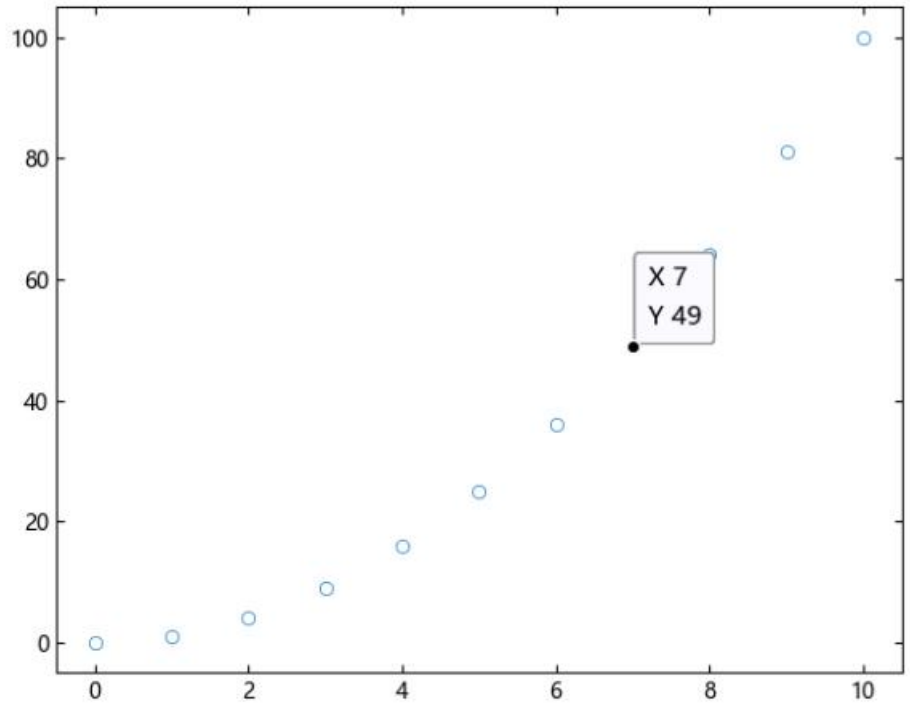
dt = datatip(er; dataindex=7);
```



(10). 使用坐标创建数据提示-散点图在散点图上绘制数据。然后，通过指定 x 和 y 坐标，在特定数据点上创建数据提示。

```
x = LinRange(0, 10, 11);
y = x .^ 2;
sc = scatter(x, y);

dt = datatip(sc, 7, 49);
```

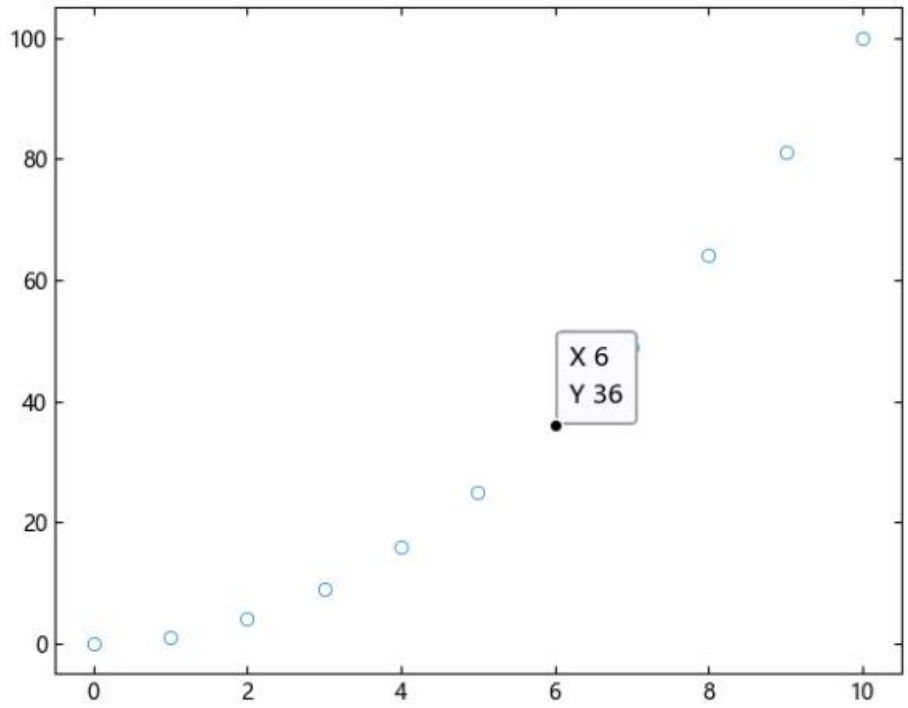


(11). 使用索引创建数据提示-散点图

在散点图上绘制数据。然后，通过指定 dataindex 名称-值对组参数，在第七个数据点创建数据提示。

```
x = LinRange(0, 10, 11);
y = x .^ 2;
sc = scatter(x, y);

dt = datatip(sc; dataindex=7);
```

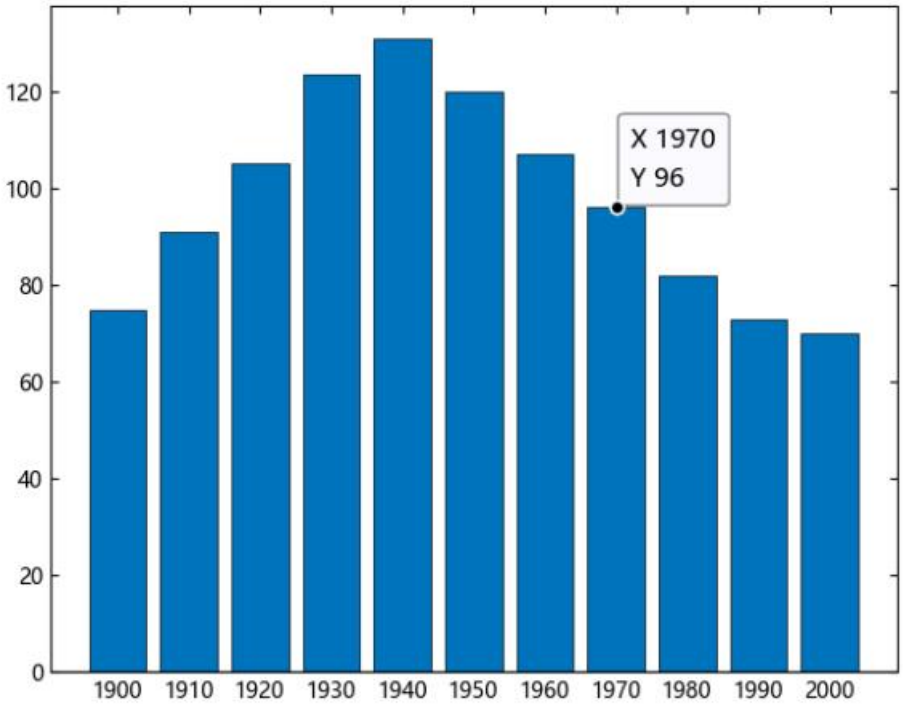


(12). 使用坐标创建数据提示-条形图

在条形图上绘制数据。然后，通过指定 x 和 y 坐标，在特定数据点上创建数据提示。

```
x = 1900:10:2000;
y = [75 91 105 123.5 131 120 107 96 82 73 70];
b = bar(x, y);

dt = datatip(b, 1970, 96);
```

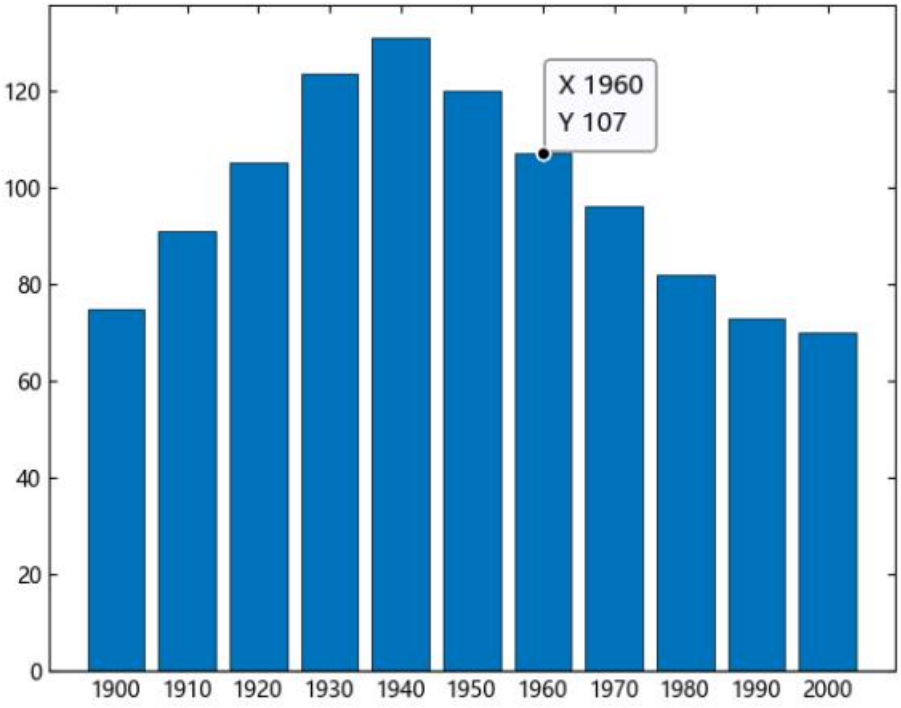


(13). 使用索引创建数据提示-条形图

在条形图上绘制数据。然后，通过指定 dataindex 名称-值对组参数，在第七个数据点创建数据提示。

```
x = 1900:10:2000;
y = [75 91 105 123.5 131 120 107 96 82 73 70];
b = bar(x, y);

dt = datatip(b; dataindex=7);
```



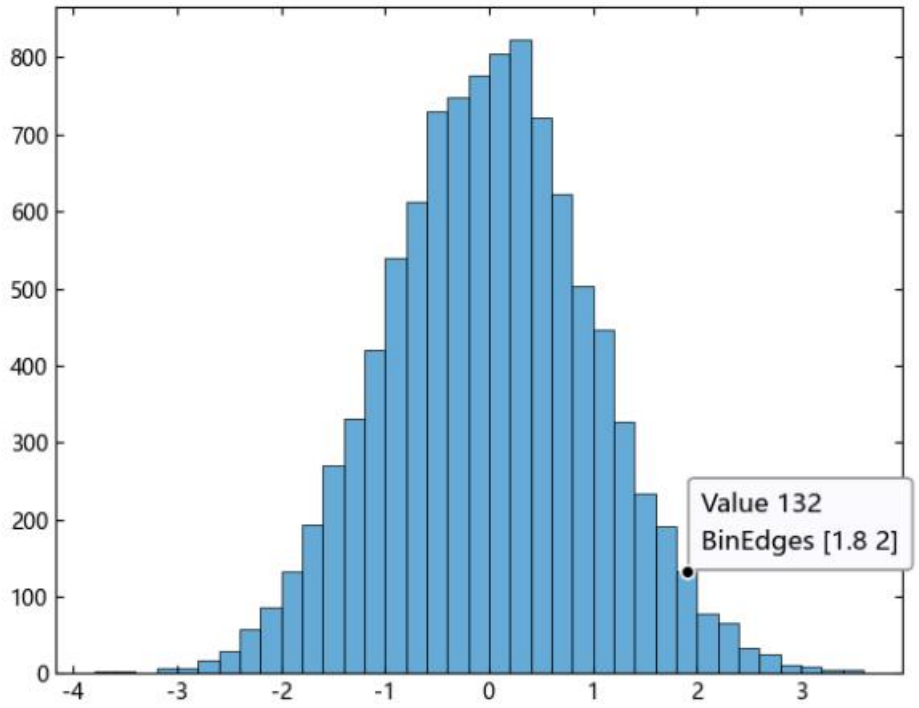
(14). 使用坐标创建数据提示-直方图

在直方图上绘制数据。然后，通过指定 x 和 y 坐标，在特定数据点上创建数据提示。

```
using TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/格式和注释/datatip/0-rand_data.jl"

x = rand(10000, 1);
include(source_path)
h = histogram(x_10000)

dt = datatip(h, 2, 132);
```



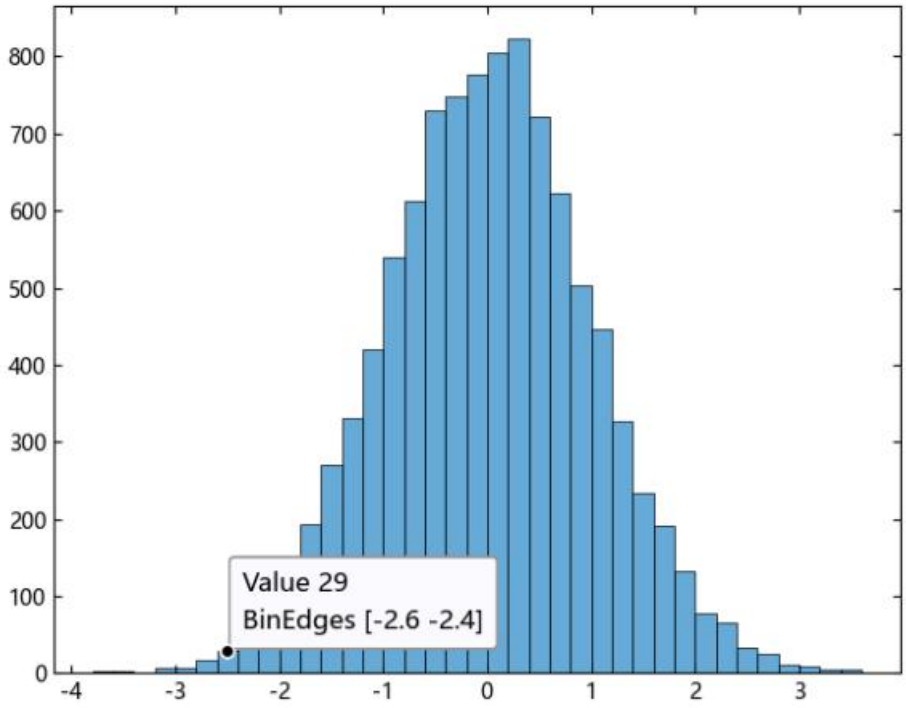
(15). 使用索引创建数据提示-直方图

在直方图上绘制数据。然后，通过指定 dataindex 名称-值对组参数，在第七个数据点创建数据提示。

```
using TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/格式和注释/datatip/0-rand_data.jl"

x = rand(10000, 1)
h = histogram(x_10000)

dt = datatip(h; dataindex=7);
```

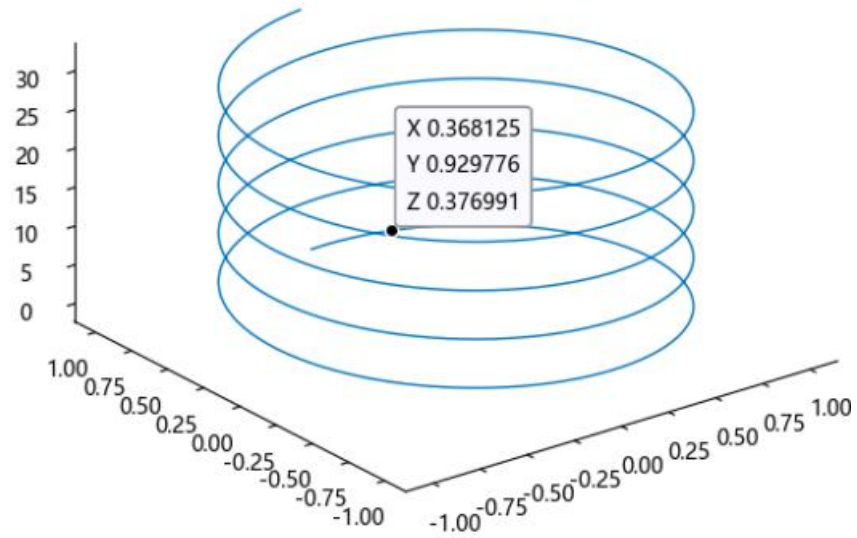


(16). 使用坐标创建数据提示-三维线图

在三维线图上绘制数据。然后，通过指定 x 和 y 坐标，在特定数据点上创建数据提示。

```
t = 0:(pi / 50):(10 * pi);
st = sin.(t);
ct = cos.(t);
p3 = plot3(st, ct, t);

dt = datatip(p3, 0.36, 0.92, 0.37);
```

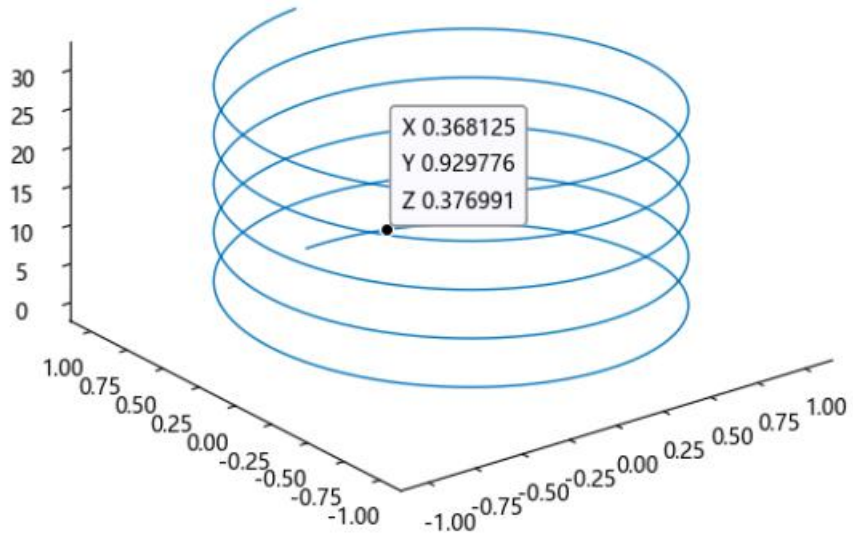


(17). 使用索引创建数据提示-三维线图

在三维线图上绘制数据。然后，通过指定 dataindex 名称-值对组参数，在第七个数据点创建数据提示。

```
t = 0:(pi / 50):(10 * pi);
st = sin.(t);
ct = cos.(t);
p3 = plot3(st, ct, t);

dt = datatip(p3; dataindex=7);
```

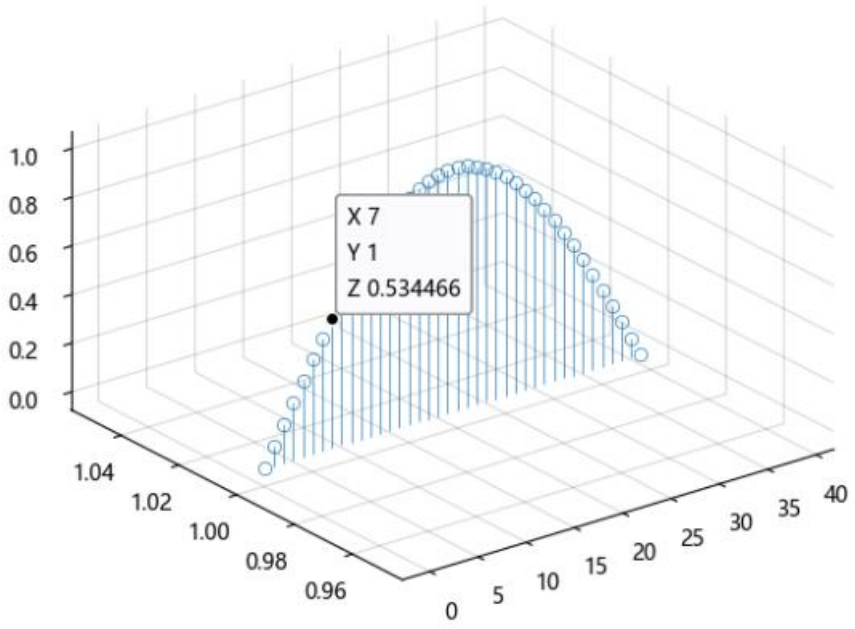


(18). 使用坐标创建数据提示-三维针状图

在三维线图上绘制数据。然后，通过指定 x 和 y 坐标，在特定数据点上创建数据提示。

```
figure()
X = LinRange(-pi / 2, pi / 2, 40);
Z = cos.(X);
st3 = stem3(Z)

dt = datatip(st3, 7, 1, 0.53);
```

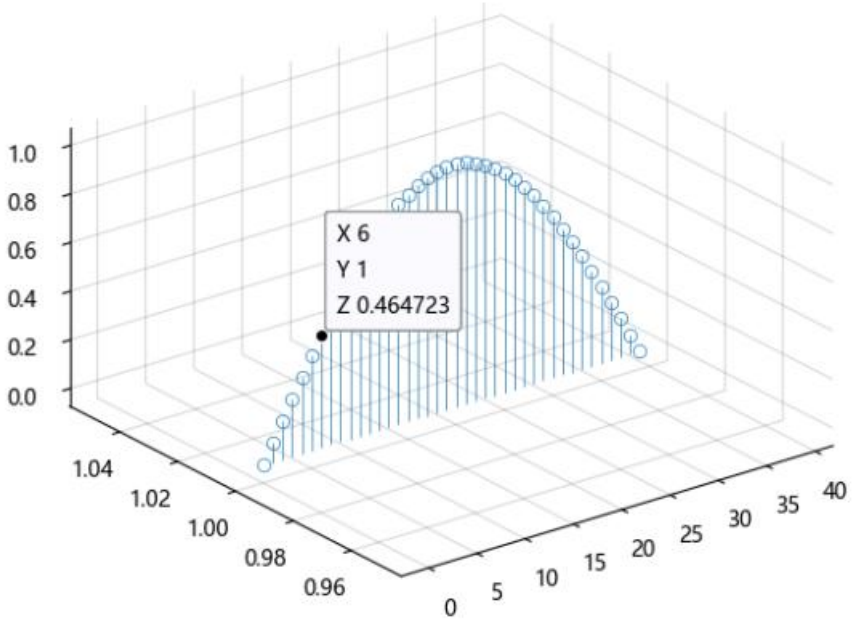


(19). 使用索引创建数据提示-三维针状图

在三维线图上绘制数据。然后，通过指定 dataindex 名称-值对组参数，在第七个数据点创建数据提示。

```
figure()
X = LinRange(-pi / 2, pi / 2, 40);
Z = cos.(X);
st3 = stem3(Z; linewidth=0.5)

dt = datatip(st3; dataindex=7);
```



line

创建基本线条

A. 语法

```
line(x,y)
line(x,y,z)
line()

line(___,Key=Value)
line(ax,___)

pl = line(___)
```

B. 说明

line(x,y)使用向量 x 和 y 中的数据在当前坐标区中绘制线条。如果 x 和 y 中有一个是矩阵或两者都是矩阵，则 line 将绘制多个线条。与 plot 函数不同，line 会向当前坐标区添加线条，而不删除其他图形对象或重置坐标区属性。

line(x,y,z)在三维坐标中绘制线条。

line() 使用默认属性设置绘制一条从点 (0,0) 到 (1,1) 的线条。

line(___,Key=Value)使用一个或多个名称-值对组参数修改线条的外观。例如，linewidth=3 将线宽设置为 3 磅。请在所有其他输入参数之后指定名称-值对组。如果使用名称-值对组指定数据，例如 line(xdata=x,ydata=y)，则必须指定向量数据。

line(ax,___) 将在由 ax 指定的笛卡尔坐标区、极坐标区或地理坐标区中创建线条，而不是在当前坐标区 (gca) 中创建线条。指定 ax 作为第一个输入参数。

pl=line(___)返回创建的所有基本 Line 对象。可在创建特定的 Line 对象后使用 pl 修改其属性。

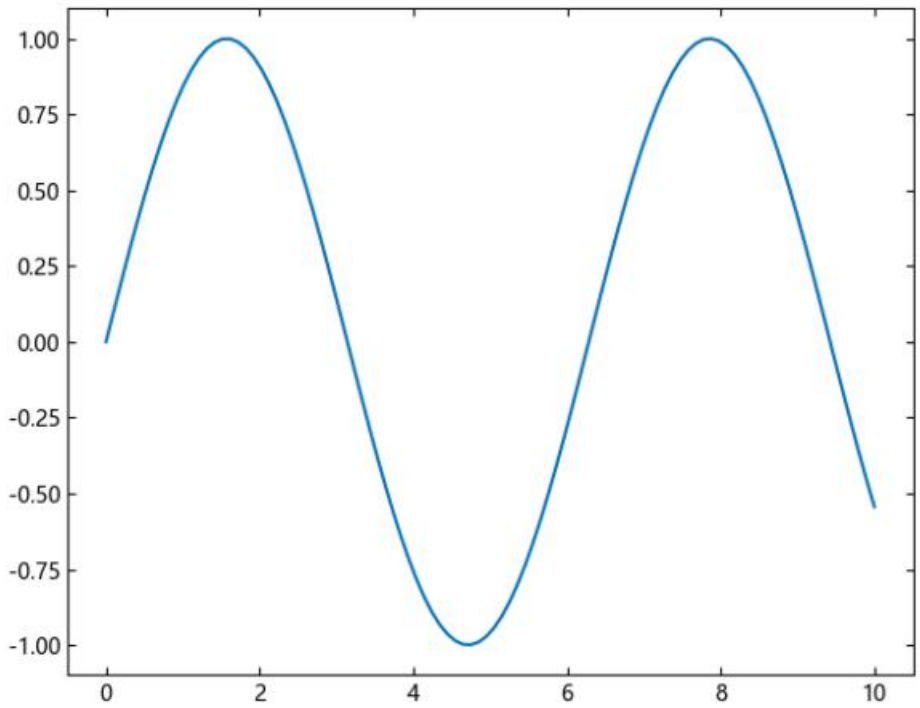
C. 示例

(1). 使用向量数据绘制线条

以向量形式创建 x 和 y。然后，绘制 y 对 x 的图。

```
x = LinRange(0,10,100);
y = sin.(x);
```

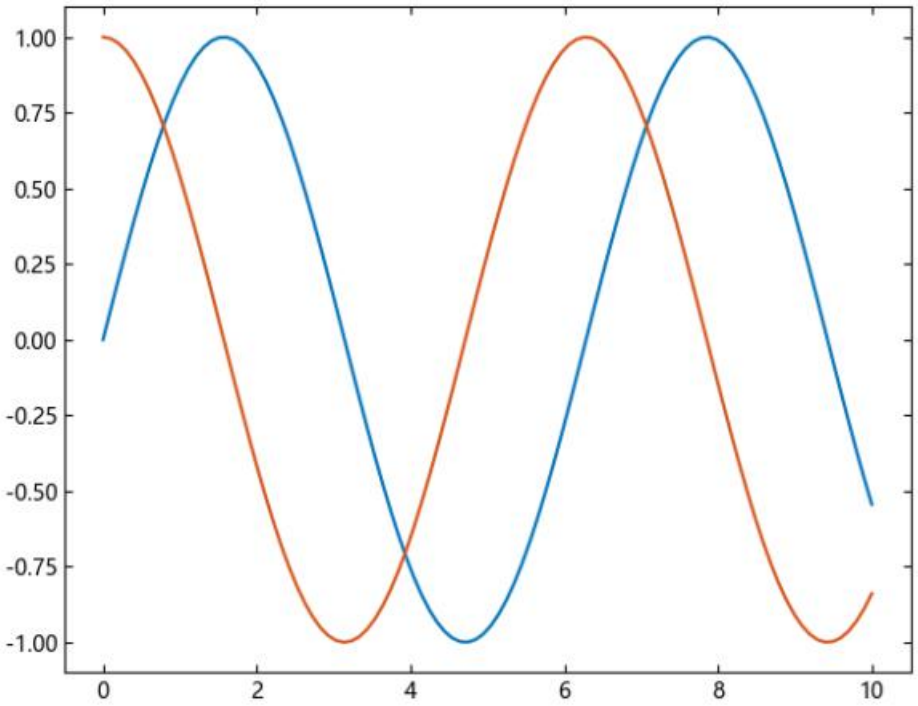
```
line(x,y)
```



(2). 使用矩阵数据绘制多个线条

通过将 x 和 y 指定为矩阵来绘制两个线条。使用 line 绘制 y 的列对 x 的列的单独线条。

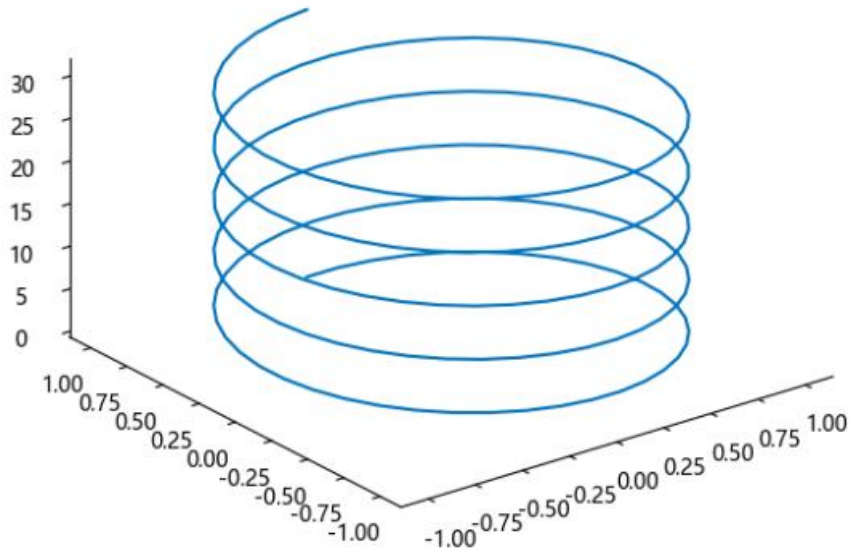
```
x = LinRange(0,10,100)';
y = [sin.(x), cos.(x)];
line(x,y)
```



(3). 使用三维坐标绘制线条

通过指定 x、y 和 z 的值，在三维坐标中绘制线条。使用 plt_view(3) 将坐标区更改为三维视图。

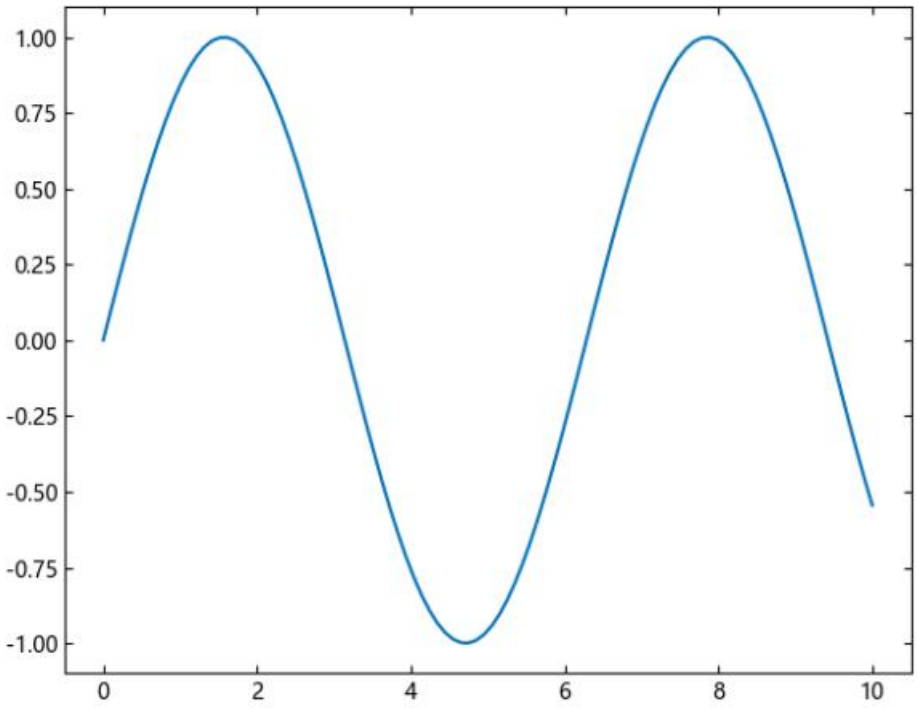
```
t = LinRange(0,10*pi,200);
x = sin.(t);
y = cos.(t);
z = t;
line(x,y,z)
plt_view(3)
```



(4). 使用低级别语法绘制线条

以向量形式创建 x 和 y。然后，使用 line 函数的低级版本绘制 y 对 x 的图。

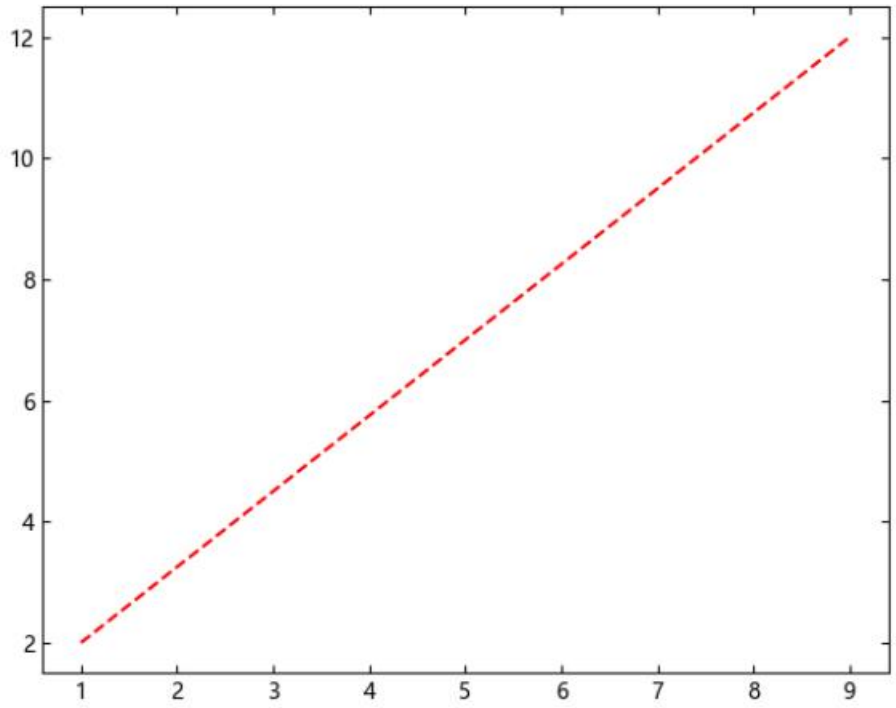
```
x = LinRange(0,10,100);
y = sin.(x);
line(xdata=x, ydata=y)
```



(5). 指定线条属性

在点 [1,2] 和 [9,12] 之间绘制一条红色虚线。将 color 和 linestyle 属性设置为名称-值对组。

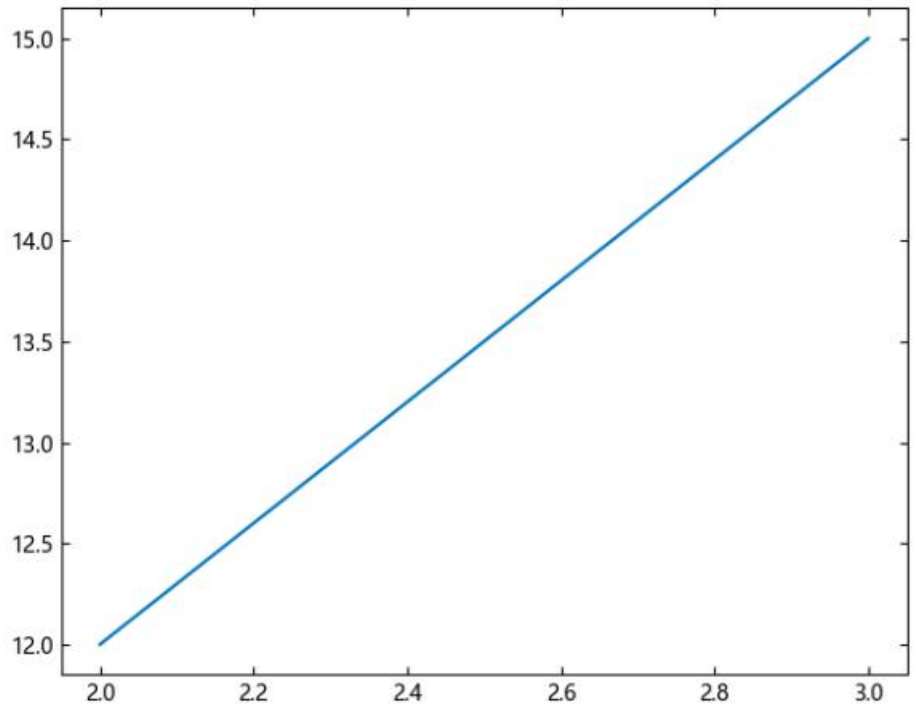
```
x = [1 9];
y = [2 12];
line(x,y,color="red",linestyle="--")
```



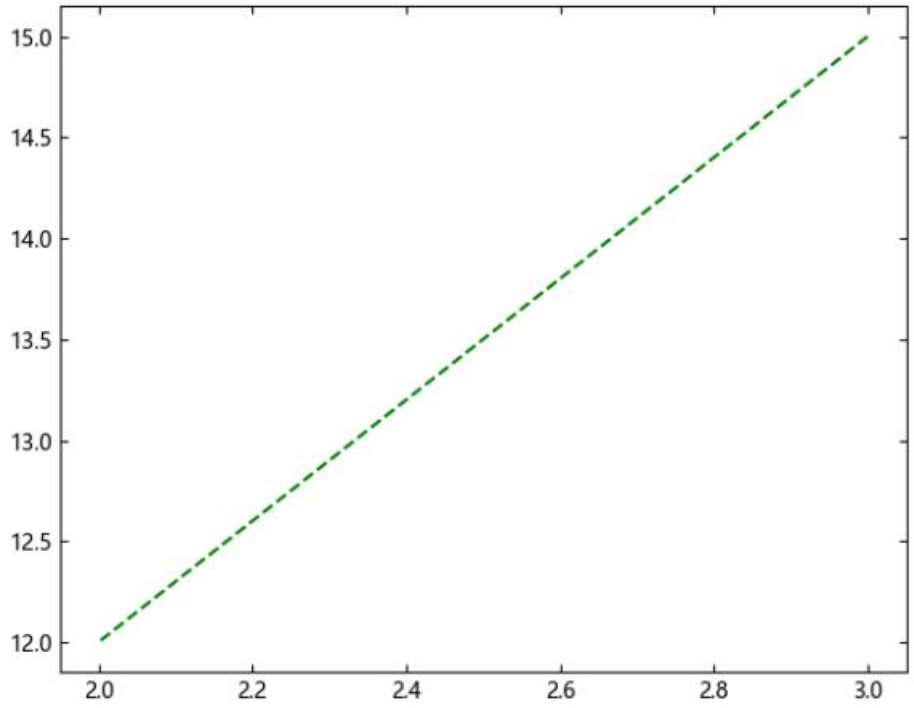
(6). 创建后更改线条属性

首先，绘制一条从点 [3,15] 到 [2,12] 的线条并返回 Line 对象。然后将线条更改为绿色虚线。使用圆点表示法设置属性。

```
x = [3 2];
y = [15 12];
pl = line(x,y)
```



```
pl[1].set_color("green")
pl[1].set_linestyle("--")
gca().stale=true
```



rectangle

创建带有尖角或圆角的矩形

A. 语法

```
rectangle(position=pos)
rectangle(position=pos,curvature=cur)
rectangle(___,Key=Value)
rectangle(ax,___)
r = rectangle(___)
```

B. 说明

rectangle(position=pos)在二维坐标中创建一个矩形。将 pos 指定为[xywh]形式的四元素向量（以数据单位表示）。x 和 y 元素确定位置，w 和 h 元素确定大小。函数在当前坐标区中绘制图形，而不清除坐标区中的现有内容。

rectangle(position=pos,curvature=cur)为矩形的边添加曲率。要沿水平边和垂直边使用不同的曲率，请将 cur 指定为[horizontalvertical]形式的二元素向量。要

沿所有边使用相同的曲率，请将 cur 指定为一个标量值。请将值指定为 0（无曲率）和 1（最大曲率）之间的值。使用[11]创建一个椭圆形或圆形。

rectangle(___,Key=Value)使用一个或多个名称-值对组参数指定矩形的属性。例如，facecolor="red"指定红色填充颜色。您可以使用上述语法中的任意输入参数组合指定矩形的属性。

rectangle(ax,___) 将在由 ax 指定的坐标区中，而不是在当前坐标区 (gca) 中创建矩形。选项 ax 可以位于前面的语法中的任何输入参数组合之前。

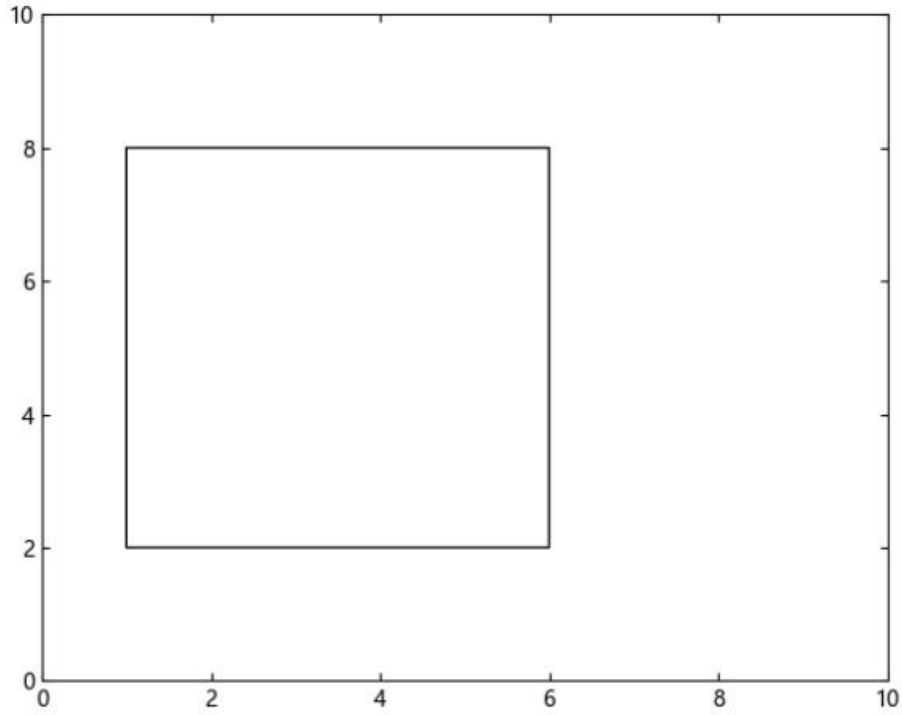
r = rectangle(___) 返回矩形对象。在创建矩形后，可以使用 r 修改其属性。

C. 示例

(1). 绘制矩形

绘制一个左下角位于点 (1,2) 位置处的矩形。将矩形的宽度设置为 5 个单位，将高度设置为 6 个单位。然后，更改坐标轴范围。

```
rectangle(position=[1 2 5 6])
axis([0 10 0 10])
```

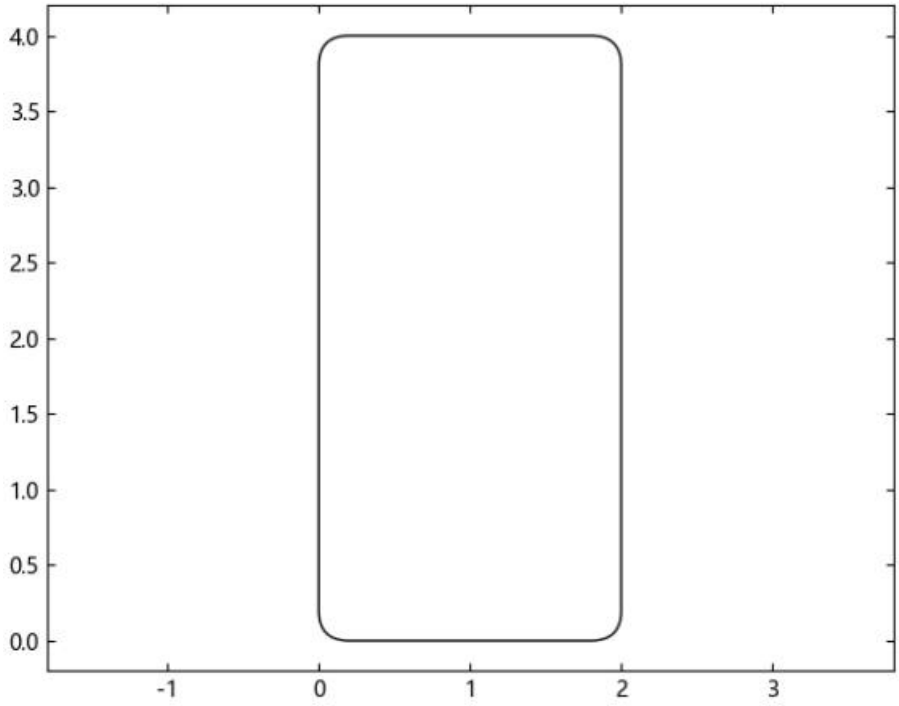


(2). 用曲线边绘制矩形

绘制一个矩形，其左下角位于点 (0,0) 位置处，右上角位于点 (2,4) 位置处。通过将曲率指定为标量值 0.2 创建具有圆角的矩形。要沿 x 轴和 y 轴使用长度相等的数据单位，请使用 axis("equal")。

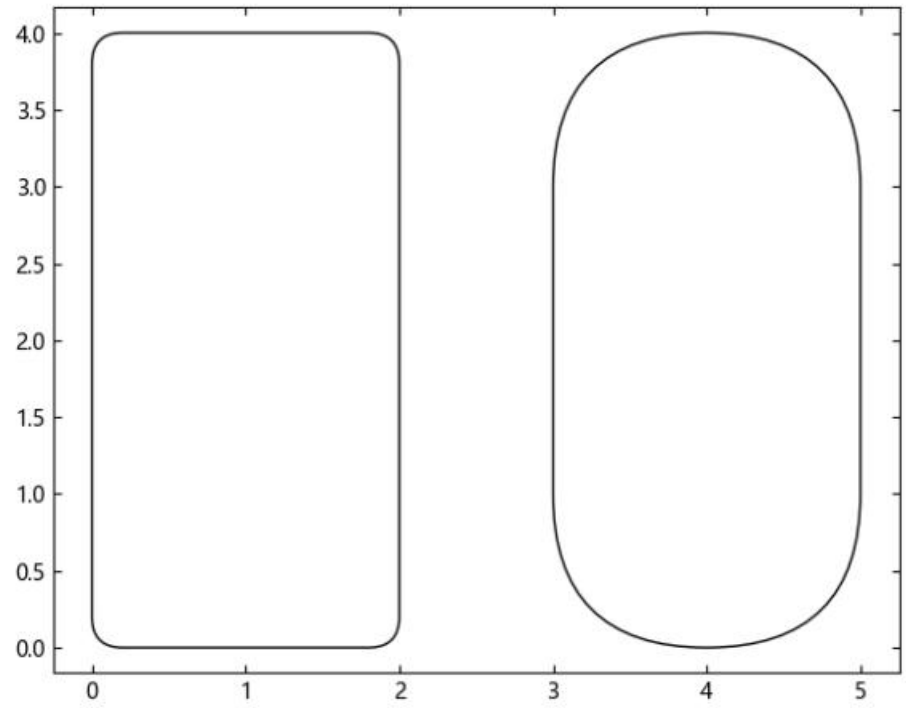
```
figure()
```

```
rectangle(position=[0 0 2 4],curvature=0.2)
axis("equal")
```



通过将曲率指定为标量值 1，添加最短边完全弯曲的另一个矩形。

```
rectangle(position=[3 0 2 4],curvature=1)
```

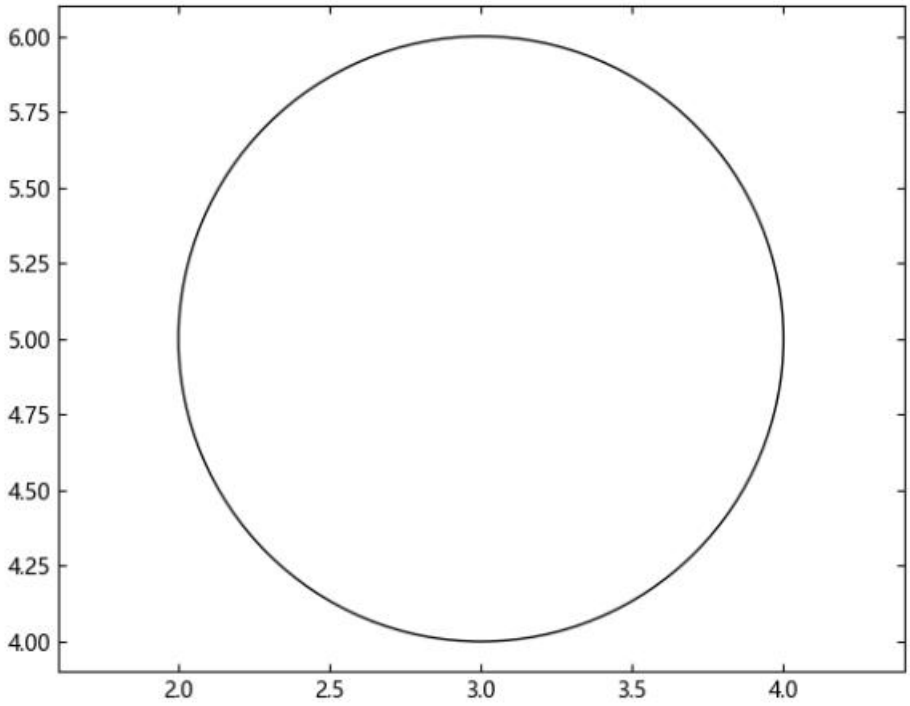


(3). 绘制圆形

通过将 Curvature 属性设置为 [1 1] 绘制一个圆。绘制一个圆形，用它来填充点 (2,4) 与 (4,6) 之间的圆形区域。Position 属性定义包含该圆形的最小

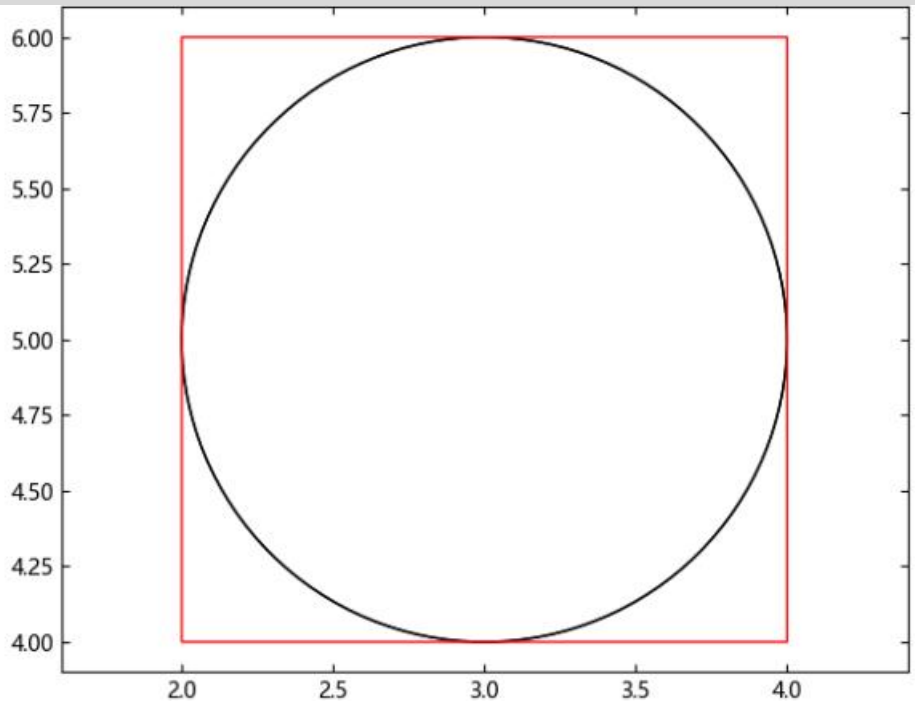
矩形。

```
pos = [2 4 2 2];
rectangle(position=pos,curvature=[1 1])
axis("equal")
```



使用相同的位置值绘制一个红色的矩形，以显示圆形如何填充矩形区域。

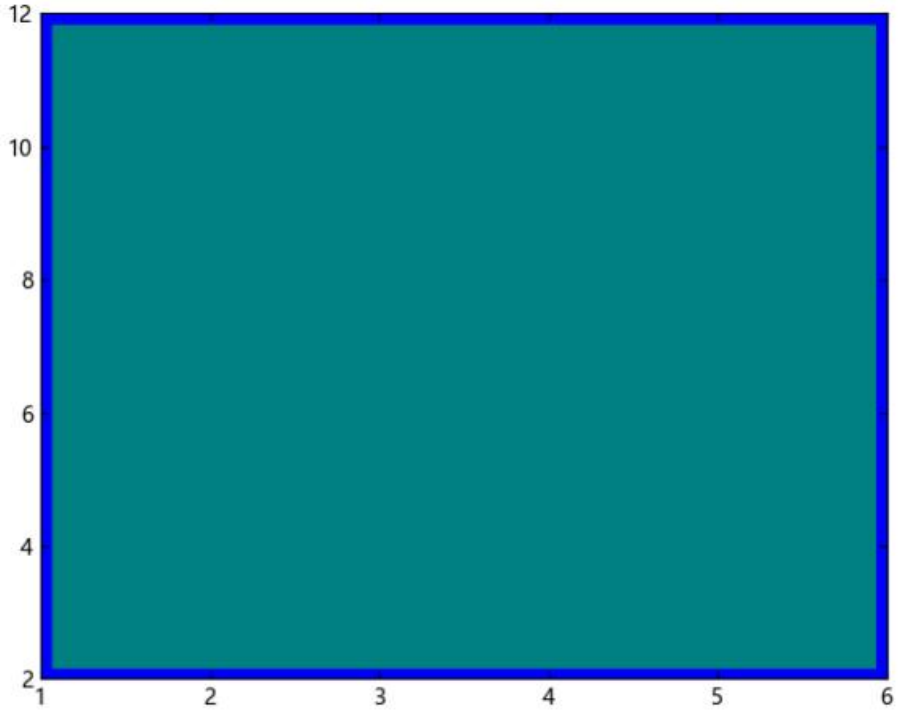
```
rectangle(position=pos,edgecolor="r")
```



(4). 指定矩形轮廓和填充颜色

绘制一个矩形并指定其填充颜色、轮廓颜色和轮廓宽度。

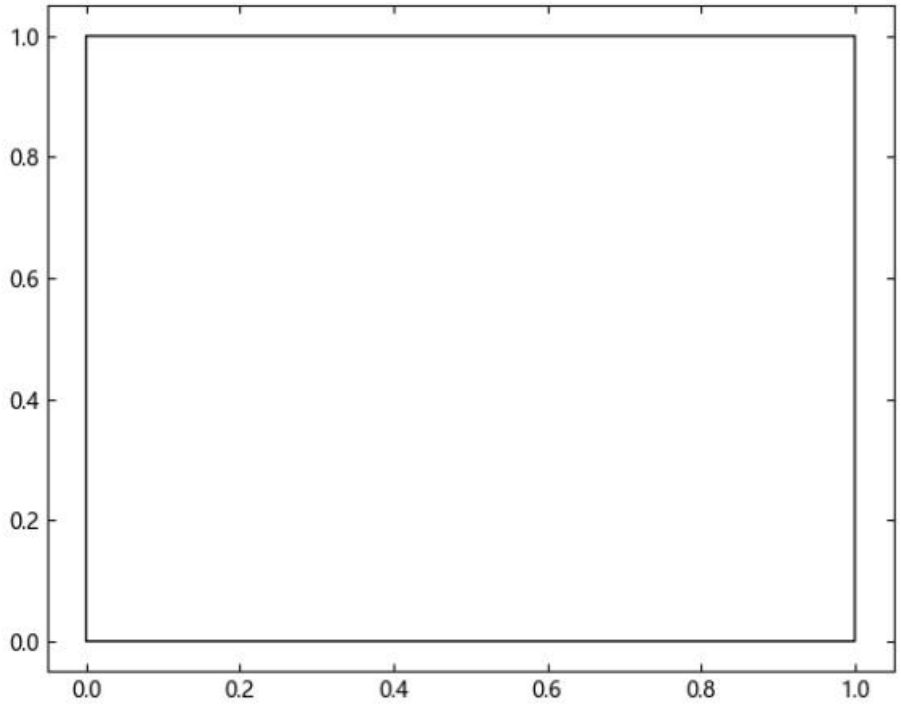
```
rectangle(position=[1,2,5,10],facecolor=[0   .5   .5],edgecolor="b",
linewidth=15)
xlim([1,6])
ylim([2,12])
```



(5). 创建并修改矩形

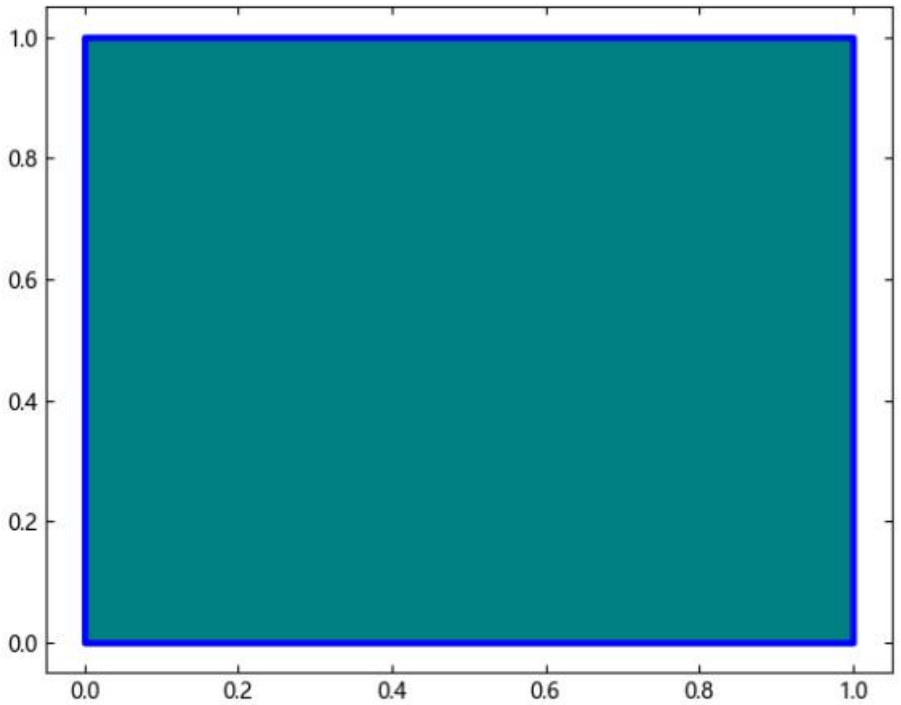
将矩形添加到当前坐标区，并返回矩形对象 r。

```
r = rectangle(position=[0 0 1 1]')
```



在创建矩形后使用 r 对其进行修改。更改填充颜色、轮廓颜色和轮廓宽度。

```
r.set_facecolor([0 .5 .5]);
r.set_edgecolor("b");
r.set_linewidth(3);
```



ginput

标识坐标区坐标

A. 语法

```
x,y = ginput(n)
x,y,button = ginput(___)
```

B. 说明

x,y=ginput(n)可用于标识 n 个点的坐标。要选择一个点，请将光标移至所需位置，然后按下鼠标按键或键盘上的键。在选中全部 n 个点之前，按 Enter 键可停止选择。Syslab 返回所选点的坐标。如果没有当前坐标区，调用 ginput 会创建一个笛卡尔坐标区。

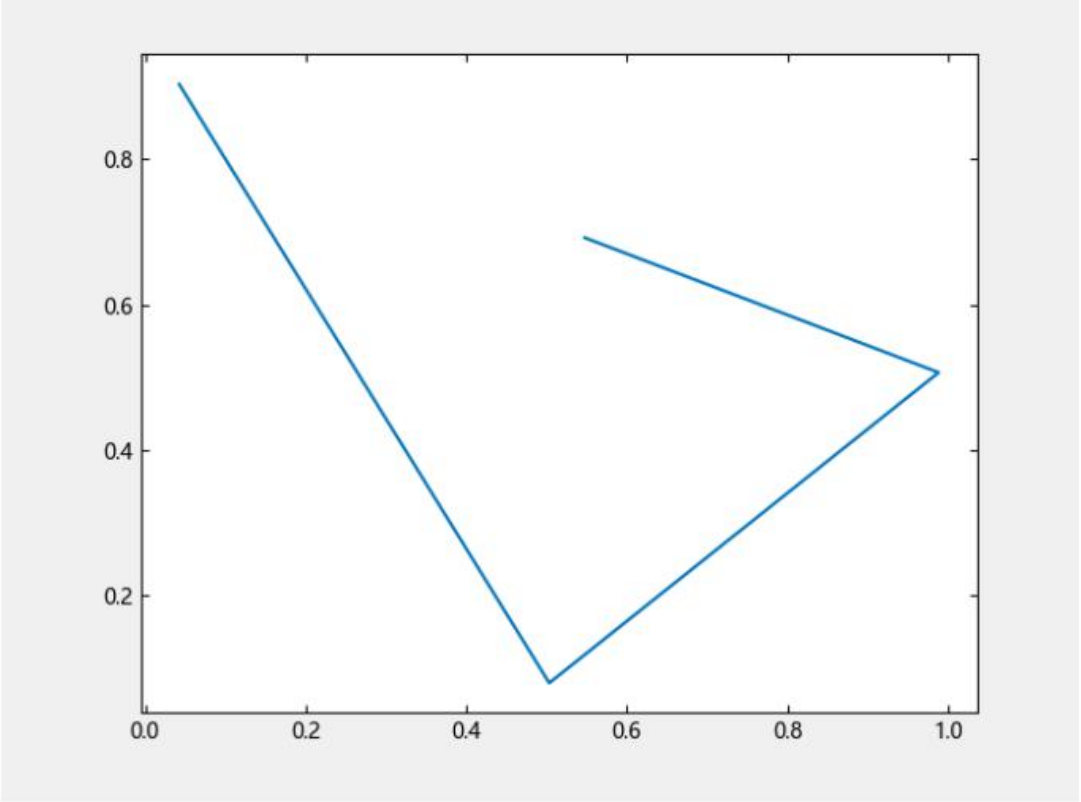
x,y,button=ginput(___)还返回用于选择每个点的鼠标按键或键盘上的键。

C. 示例

(1). 标识点和绘制坐标

使用 ginput 标识一个坐标区中的四个点。要选择各点，请将光标移至所需位置，然后按鼠标按键或键。绘制这些点。

```
x,y = ginput(4)
plot(x,y);
```



(2). 返回用于选择坐标的按键

使用 ginput 标识一个坐标区中的五个坐标。要选择各点，请将光标移至所需位置，然后按鼠标按键、小写字母键、大写字母键、数字键或空格键。返回用于选择每个点的鼠标按键或键的 ASCII 数字。

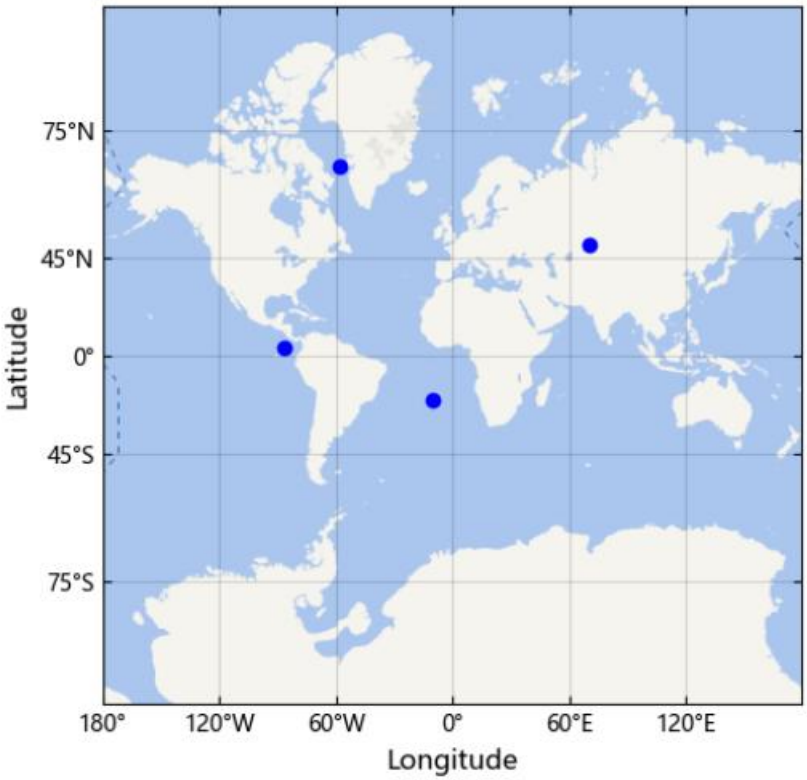
```
x,y,button = ginput(5, showbutton=true);
println(button)
```

(3). 标识地理坐标区上的点

创建地理坐标区并标识四个点的纬度和经度坐标。然后，绘制您标识的点。使用 hold("on") 命令和 geolimits() 函数来保持地图范围不变。绘制标识的点。

```
using TyPlot, TyGeoGraphics
geoaxes();
lat,lon = ginput(4);

hold("on")
geolimits("manual")
geoscatter(lat, lon, filled=true, markerfacecolor="b")
hold("off")
```



3.3.2.2 坐标区外观

修改坐标轴范围和刻度值、添加网格线、合并多个绘图

您可以通过更改范围、控制刻度线的位置、格式化刻度标签或添加网格线来自定义坐标区。您也可以通过在同一图窗中使用独立的坐标区，或通过在相同的坐标区中合并绘图（可选择添加第二个 y 轴），来合并多个绘图。

3.3.2.2.1 坐标区范围和纵横比

函数名	简介
xlim	设置或查询 x 坐标轴范围
ylim	设置或查询 y 坐标轴范围
zlim	设置或查询 z 坐标轴范围
axis	设置坐标轴范围和纵横比
daspect	控制沿每个轴的数据单位长度
pbaspect	控制每个轴的相对长度

xlim

设置或查询 x 坐标轴范围

A. 语法

```
xlim(limits)
xl = xlim()

xlim("auto")
xlim("manual")

___ = xlim(target,___)
```

B. 说明

xlim(limits)设置当前坐标区或图的 x 坐标轴范围。将 limits 指定为[xmin,xmax]形式的二元素向量，其中 xmax 大于 xmin。

xl=xlim()以二元素向量形式返回当前范围。

xlim("auto") 允许 Syslab 选择 x 坐标轴范围。此命令将极坐标区对象的 RLimMode 属性设置为 "auto"。

xlim("manual") 防止范围自动更改。当使用 hold("on") 命令向极坐标区添加新数据时，如果要保留当前范围，请使用此选项。此命令将极坐标区对象的 RLimMode 属性设置为 "manual"。

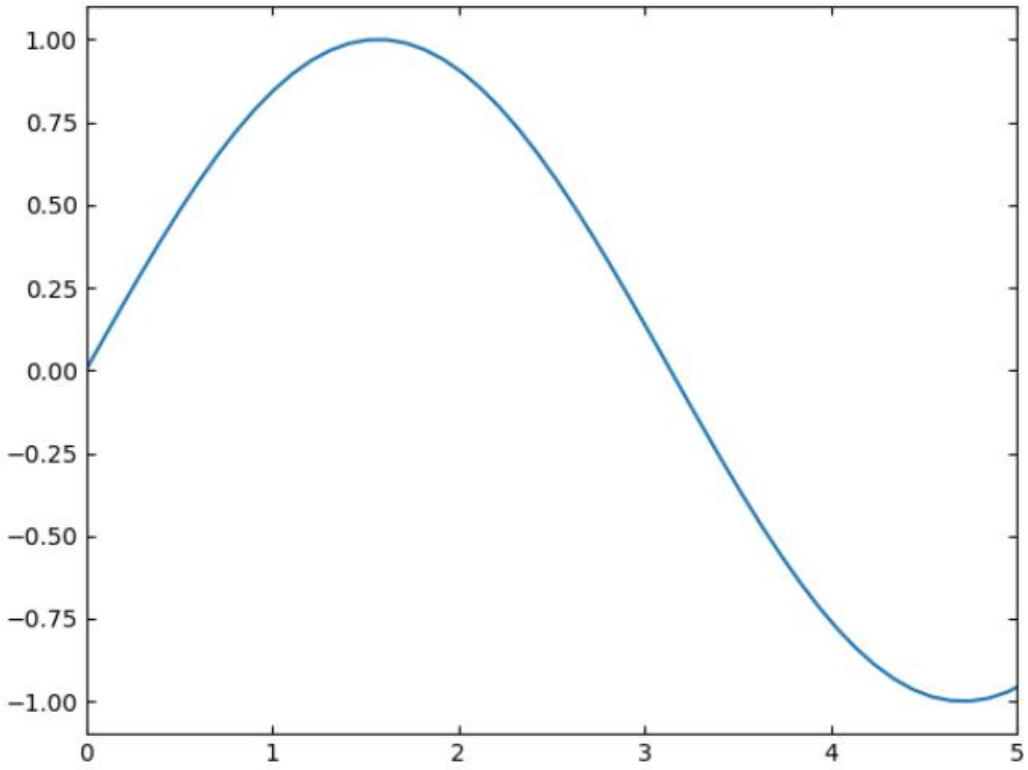
___ = xlim(target,___)使用由 target 指定的坐标区或图，而不是当前坐标区。指定 target 作为上述任何语法的第一个输入参数。如果原始语法支持输出参数，则可以包含输出参数。xlim(target,auto=true)。

C. 示例

(1). 设置 x 坐标轴范围

绘制一个线条，并将 x 坐标轴范围设置为从 0 至 5。

```
x = LinRange(0, 10, 100);
y = sin.(x);
plot(x, y)
xlim([0, 5])
```



(2). 设置具有日期值的 x 轴的范围

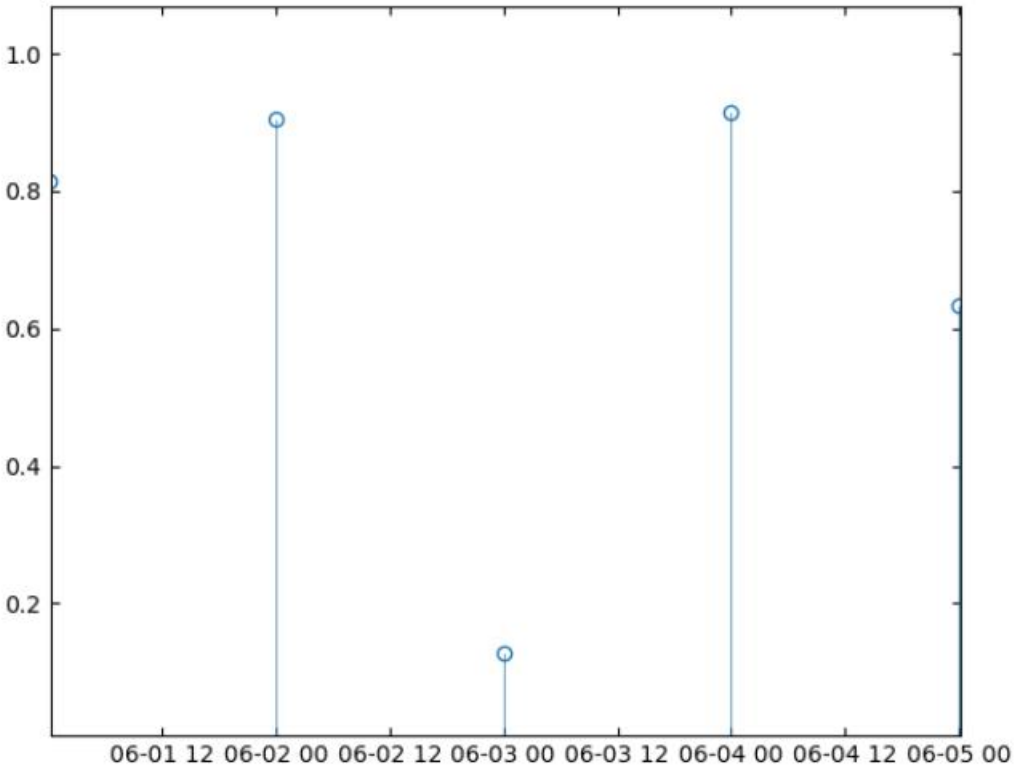
以 x 轴为日期值创建一个针状图。将 x 坐标轴范围设置为从 2014 年 6 月 1 日到 2014 年 6 月 5 日。

```
using Dates
days = []
for i in 0:10
   push!(days, Dates.Day(i))
end
t = Date(2014,06,1) .+ days

y = rand(11,1);   #返回一个 11×1 的随机数矩阵

stem(t,y)

tstart = Date(2014,06,1)
tend = Date(2014,06,5)
xlim([tstart,tend])
```

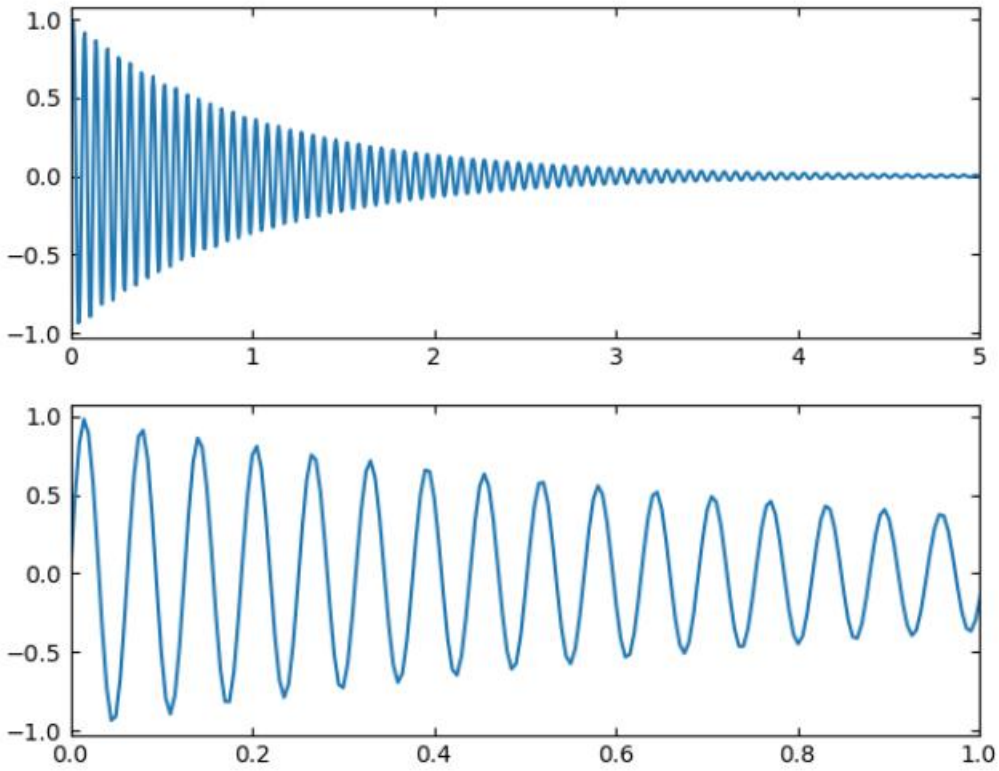


(3). 设置特定坐标区的 x 坐标轴范围

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。通过将 ax1 传递给 plot 函数来创建顶部绘图。重复该过程以创建底部绘图。在每个坐标区中绘制数据。通过指定 ax2 作为 xlim 的第一个输入参数，设置底部图的 x 坐标轴范围。

```
ax1 = subplot(2, 1, 1)
x = LinRange(0, 5, 1000);
y = sin.(100 * x) ./ exp.(x);
plot(ax1, x, y)
xlim(ax1, [0, 5])

ax2 = subplot(2, 1, 2)
plot(ax2, x, y)
xlim(ax2, [0, 1])
```

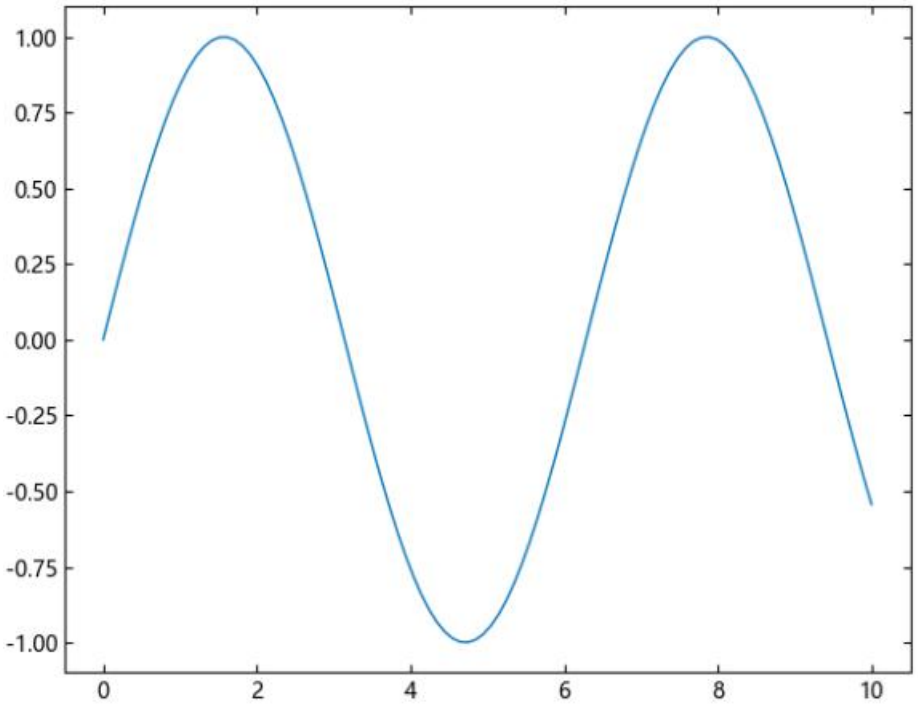


(4). 保持当前 x 坐标轴范围

当将更多绘图添加到坐标区时，使用手动模式可保持当前 x 坐标轴范围。

首先，绘制一个线条。

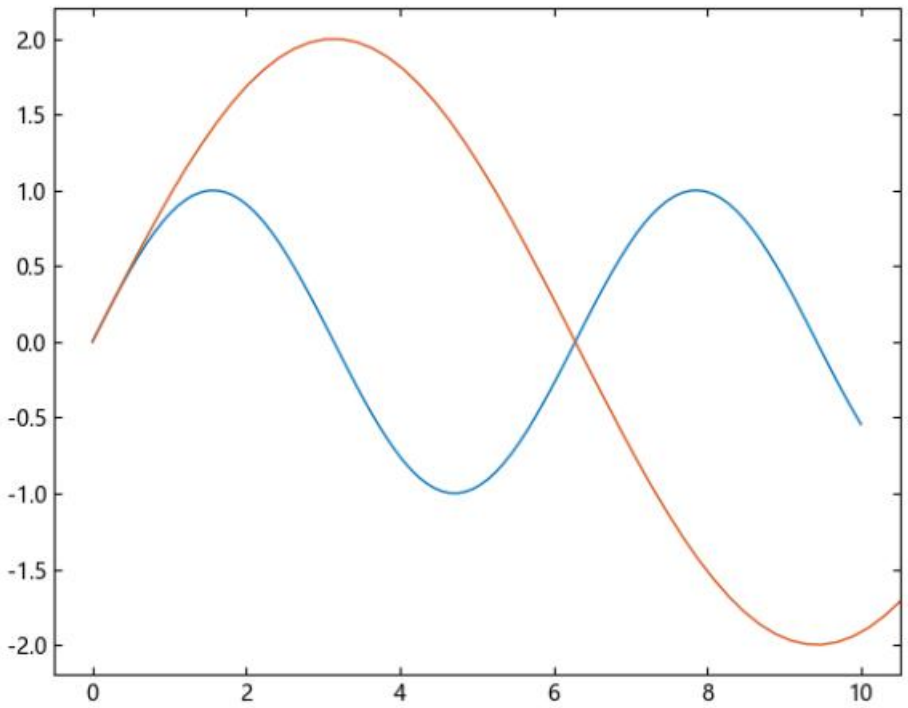
```
x = LinRange(0, 10, 100);
y = sin.(x);
plot(x, y);
```



将 x 坐标轴范围模式设置为手动，以便范围不发生变化。使用 hold("on") 将第二个绘图添加到坐标区。

```
xlim("manual")

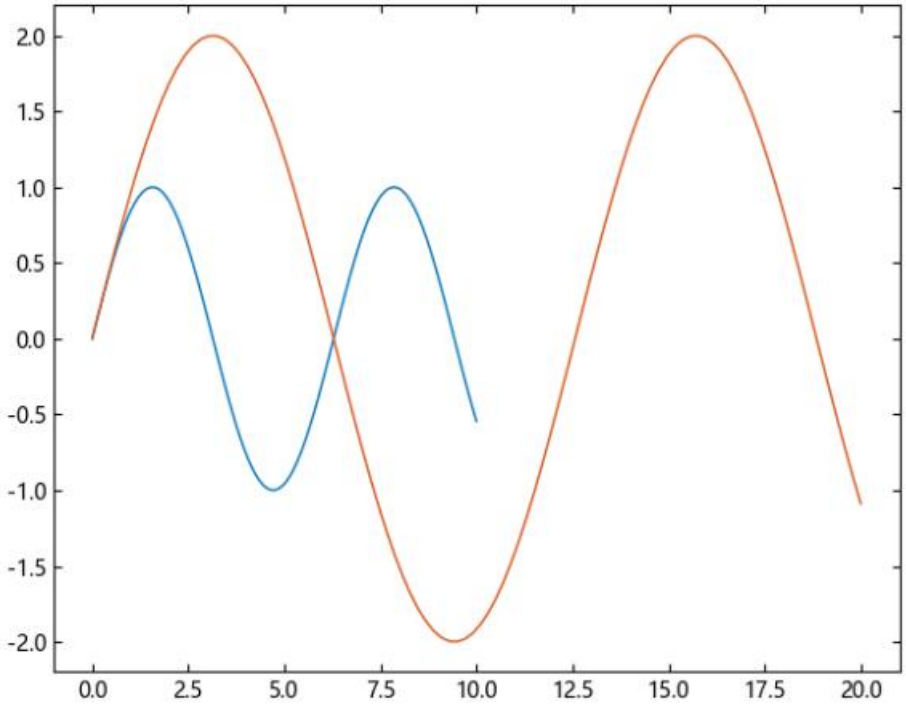
hold("on")
plot(2*x, 2*y)
hold("off")
```



x 坐标轴范围不会更新以包含该新绘图。

通过将模式重置为自动来重新切换回自动更新的范围。

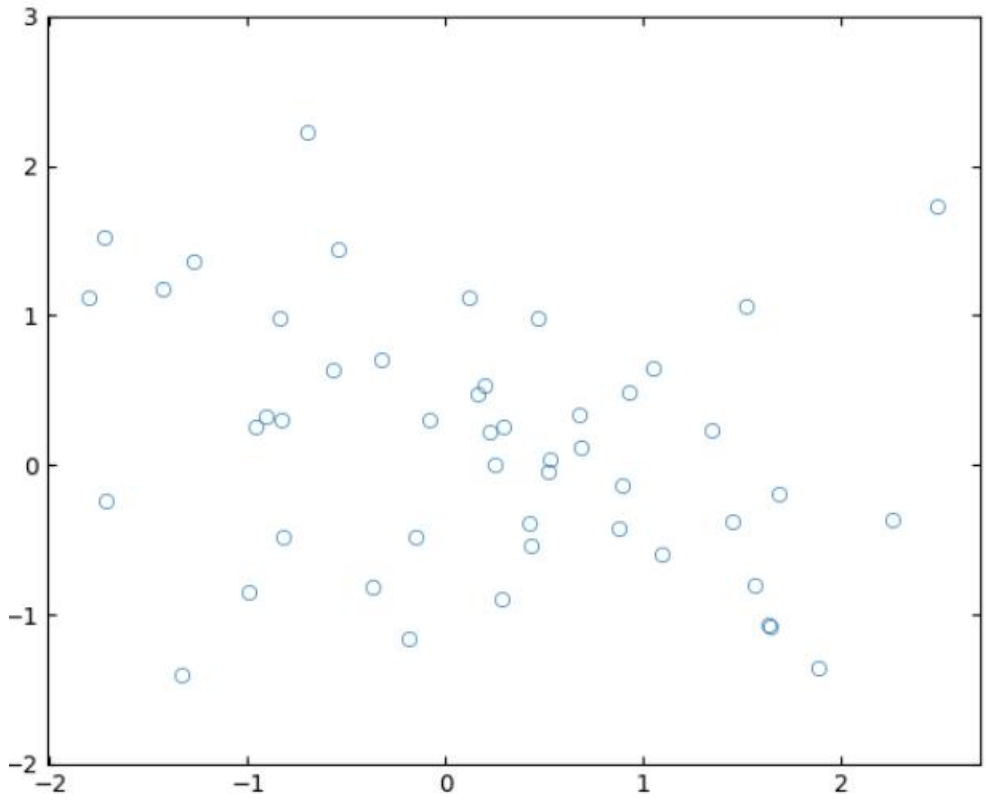
```
xlim("auto")
```



(5). 返回 x 坐标轴范围

创建一个由随机数据组成的散点图。返回 x 坐标轴范围的值。

```
x = randn(50,1); # 返回由正态分布的随机数组成的 50×1 矩阵
y = randn(50,1); # 返回由正态分布的随机数组成的 50×1 矩阵
scatter(x,y)
xlim([-3, 4])
ylim([-2,3])
```




```
xl = xlim()
println(xl)
```

```
xl = (-3.270435, 3.904535)
```

ylim

设置或查询 y 坐标轴范围

A. 语法

```
ylim(limits)
yl = ylim()

ylim("auto")
ylim("manual")

___ = ylim(target,___)
```

B. 说明

ylim(limits) 设置当前坐标区或图的 y 坐标轴范围。将 limits 指定为

[ymin,ymax]形式的二元素向量，其中 ymax 大于 ymin。

yl=ylim()以二元素向量形式返回当前范围。

ylim("auto") 允许 Syslab 选择 y 坐标轴范围。此命令将极坐标区对象的 RLimMode 属性设置为 "auto"。

ylim("manual") 防止范围自动更改。当使用 hold("on") 命令向极坐标区添加新数据时，如果要保留当前范围，请使用此选项。此命令将极坐标区对象的 RLimMode 属性设置为 "manual"。

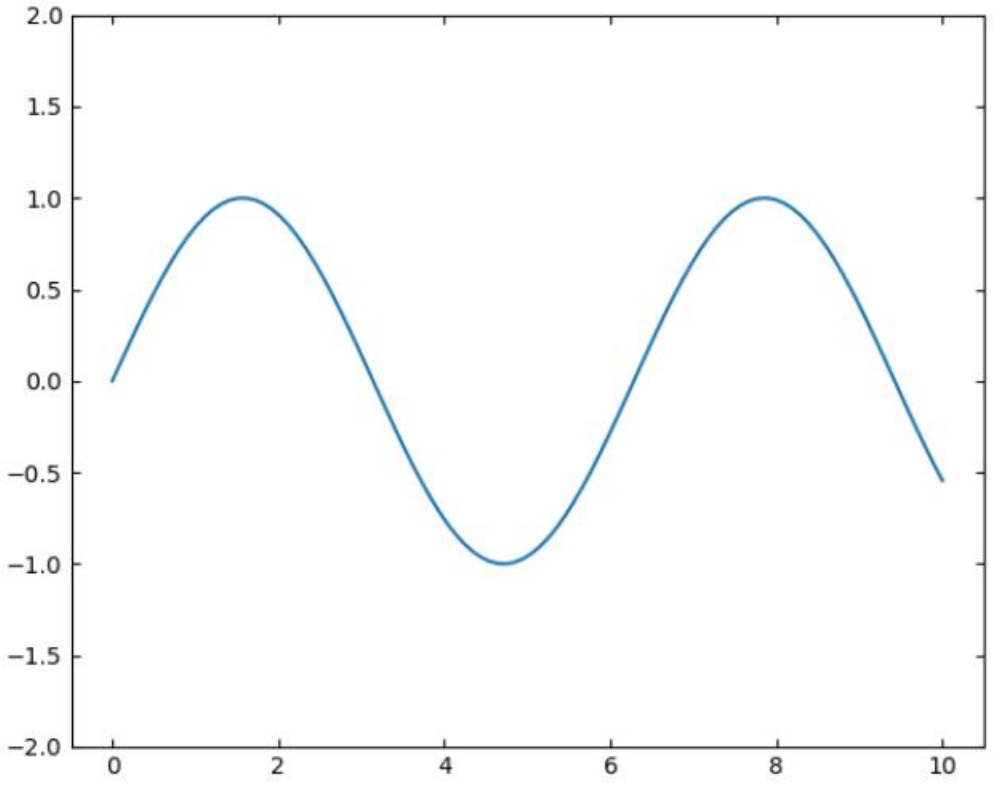
___ = ylim(target,___)使用由 target 指定的坐标区或图，而不是当前坐标区。指定 target 作为上述任何语法的第一个输入参数。如果原始语法支持输出参数，则可以包含输出参数。ylim(target,auto=true)。

C. 示例

(1). 设置 y 坐标轴范围

绘制一个线条，并将 y 坐标轴范围设置为从 -2 至 2。

```
x = LinRange(0, 10, 100);
y = sin.(x);
plot(x, y)
ylim([-2, 2])
```

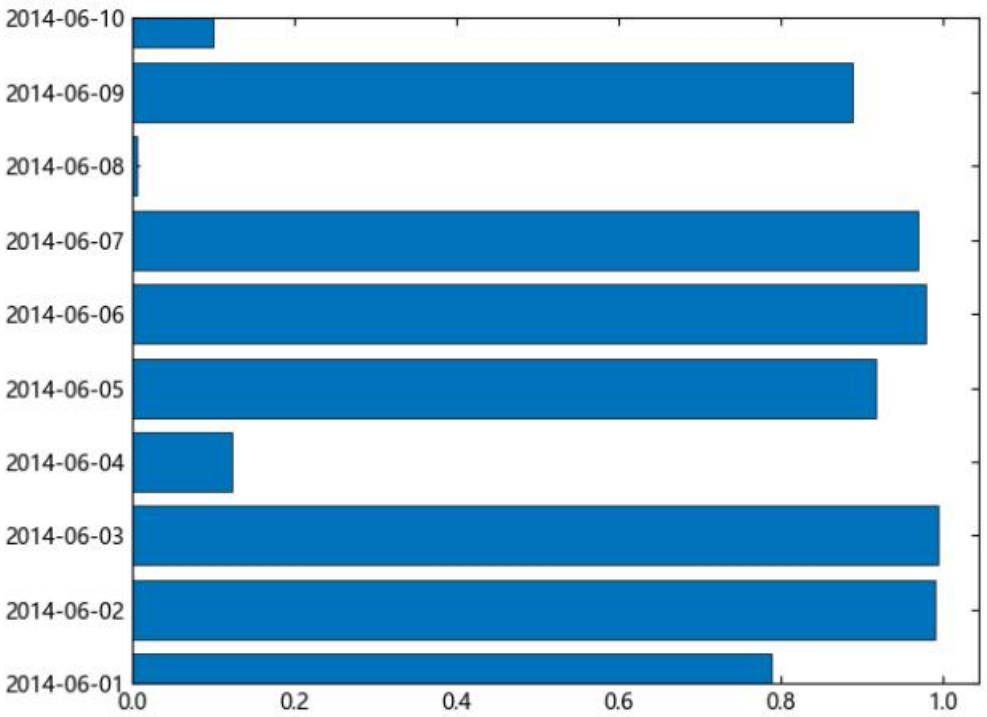


(2). 设置具有日期值的 y 轴的范围

以 y 轴为日期值创建一个水平条形图。将 y 坐标轴范围设置为从 2014 年 6 月 1 日到 2014 年 6 月 10 日。

```
using Dates
days = []
for i in 0:20
   push!(days, Dates.Day(i))
end
t = Date(2014,06,1) .+ days
y = rand(21);
barh(t,y)

tstart = Date(2014,06,1)
tend = Date(2014,06,10)
ylim([tstart,tend])
```



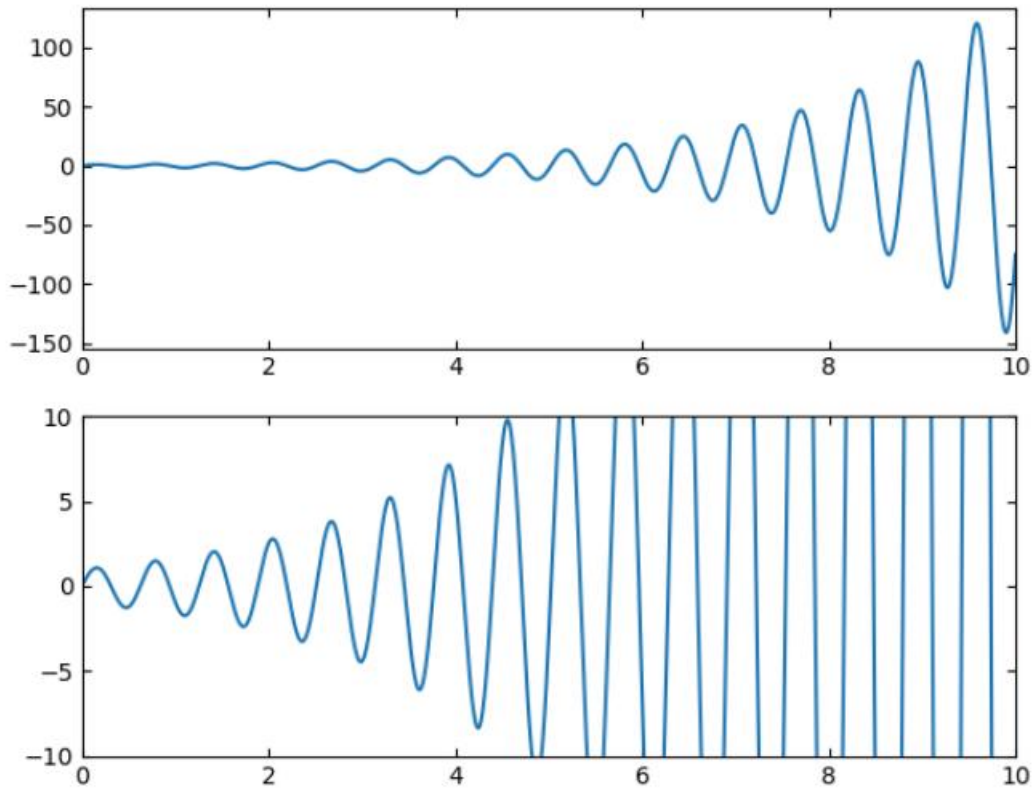
(3). 设置特定坐标区的 y 坐标轴范围

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。通过将 ax1 传递给 plot 函数来创建顶部绘图。重复该过程以创建底部绘图。在每个坐标区中绘制数据。通过指定 ax2 作为 ylim 的第一个输入参数，设置底部图的 y 坐标轴范围。

```
ax1 = subplot(2, 1, 1)
x = LinRange(0, 10, 1000);
y = sin.(10 * x) .* exp.(0.5 * x);
```

```
plot(ax1, x, y)
xlim(ax1, [0, 10])

ax2 = subplot(2, 1, 2)
plot(ax2, x, y)
xlim(ax2, [0, 10])
ylim(ax2, [-10, 10])
```

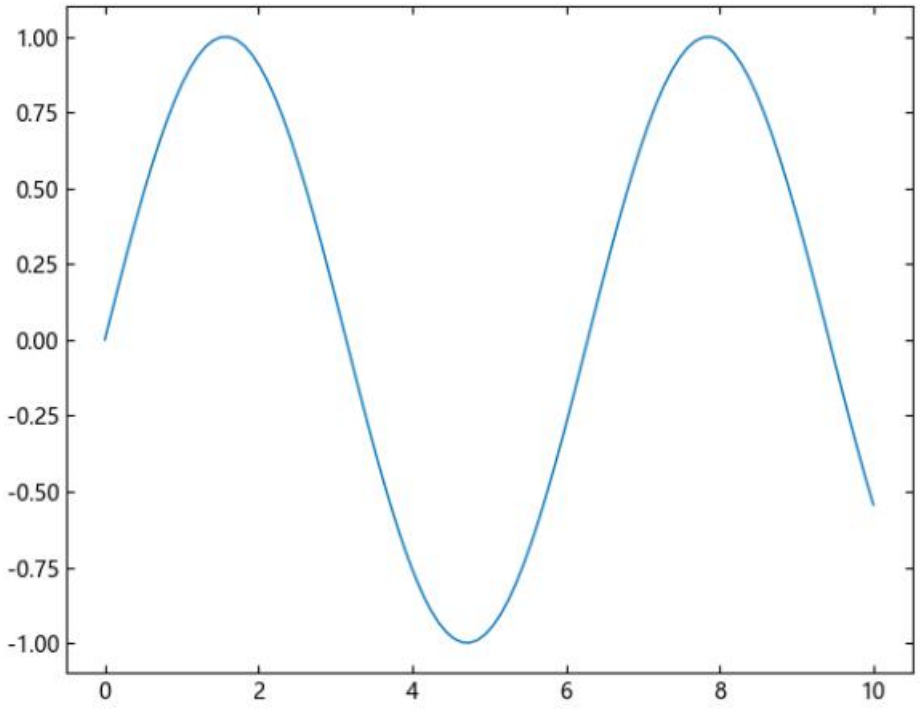


(4). 保持当前 y 坐标轴范围

当将更多绘图添加到坐标区时，使用手动模式可保持当前 y 坐标轴范围。

首先，绘制一个线条。

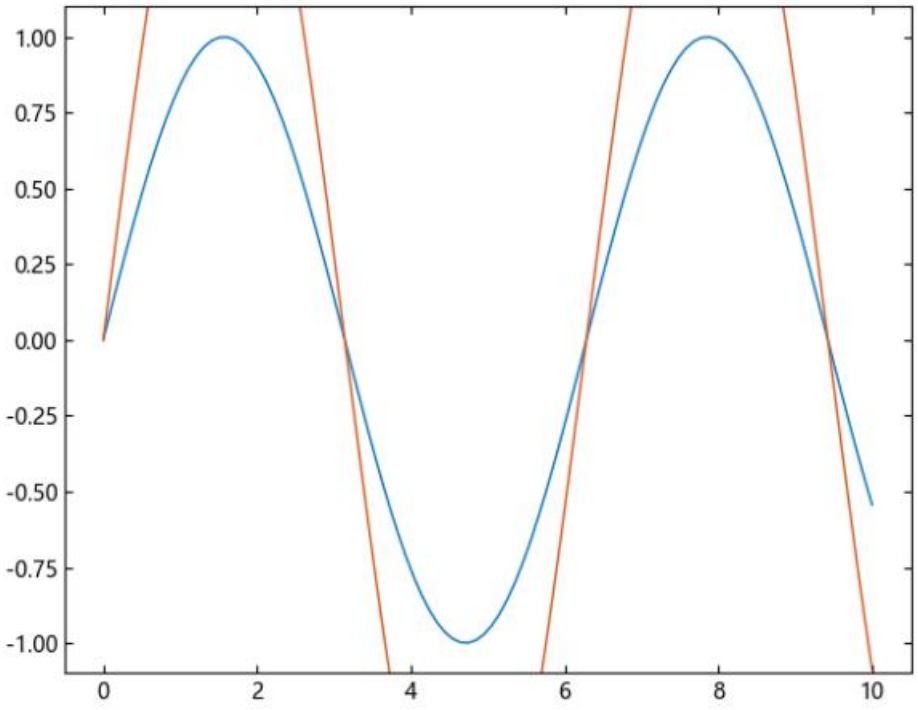
```
x = LinRange(0, 10, 100);
y = sin.(x);
plot(x, y);
```



将 y 坐标轴范围模式设置为手动，以便范围不发生变化。使用 hold("on") 将第二个绘图添加到坐标区。

```
ylim("manual")

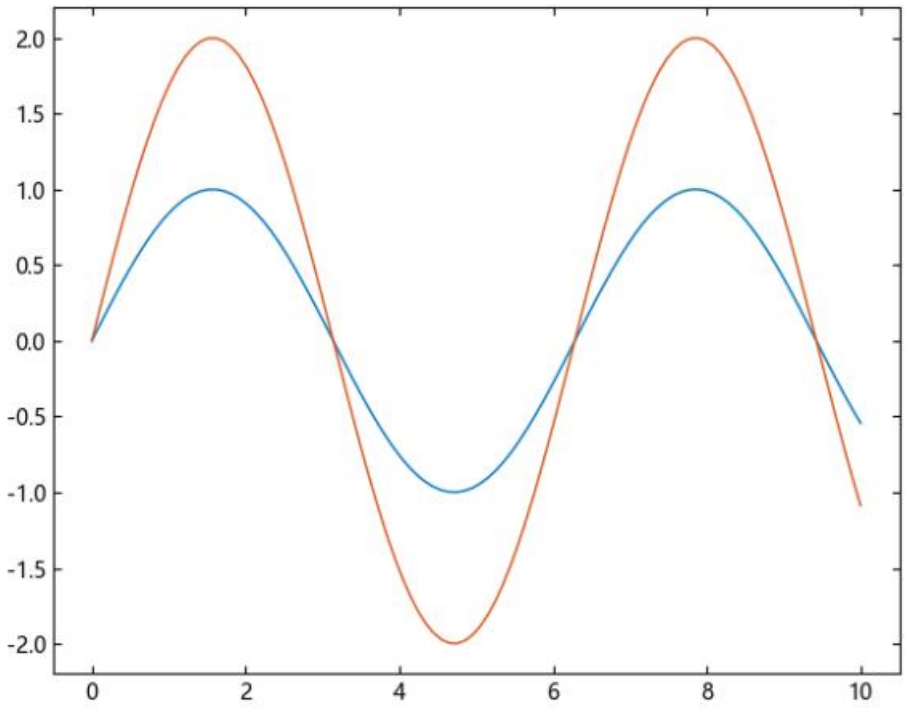
hold("on")
y2 = 2 * sin.(x);
plot(x, y2)
hold("off")
```



y 坐标轴范围不会更新以包含该新绘图。

通过将模式重置为自动来重新切换回自动更新的范围。

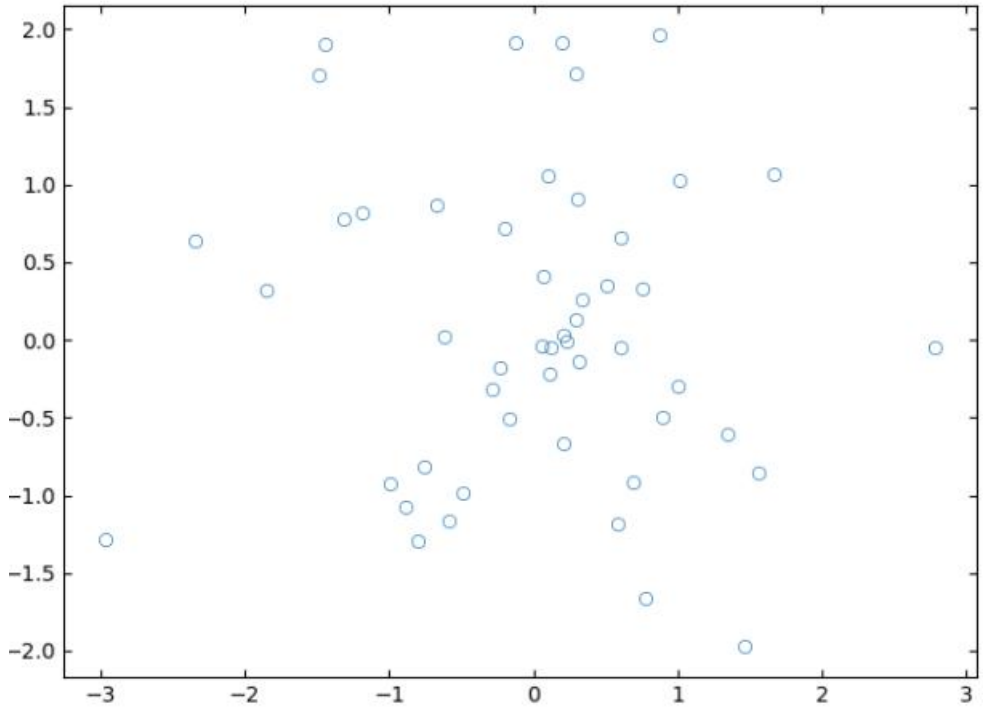
```
ylim("auto")
```



(5). 返回 y 坐标轴范围

创建一个由随机数据组成的散点图。返回 y 坐标轴范围的值。

```
x = randn(50,1); #返回由正态分布的随机数组成的 m×n 矩阵
y = randn(50,1); #返回由正态分布的随机数组成的 m×n 矩阵
scatter(x,y)
xlim([-3, 4])
ylim([-2,3])
```




```
yl = ylim()
println(yl)
```

```
yl = (-2.0, 3.0)
```

zlim

设置或查询 z 坐标轴范围

A. 语法

```
zlim(limits)
zl = zlim()

zlim("auto")
zlim("manual")

___ = zlim(target,___)
```

B. 说明

zlim(limits)设置当前坐标区或图的 z 坐标轴范围。将 limits 指定为[zmin,zmax]形式的二元素向量，其中 zmax 大于 zmin。

zl=zlim()以二元素向量形式返回当前范围。

zlim("auto") 允许 Syslab 选择 z 坐标轴范围。

zlim("manual") 防止范围自动更改。当使用 hold("on") 命令向极坐标区添加新数据时，如果要保留当前范围，请使用此选项。此命令将极坐标区对象的 RLimMode 属性设置为 "manual"。

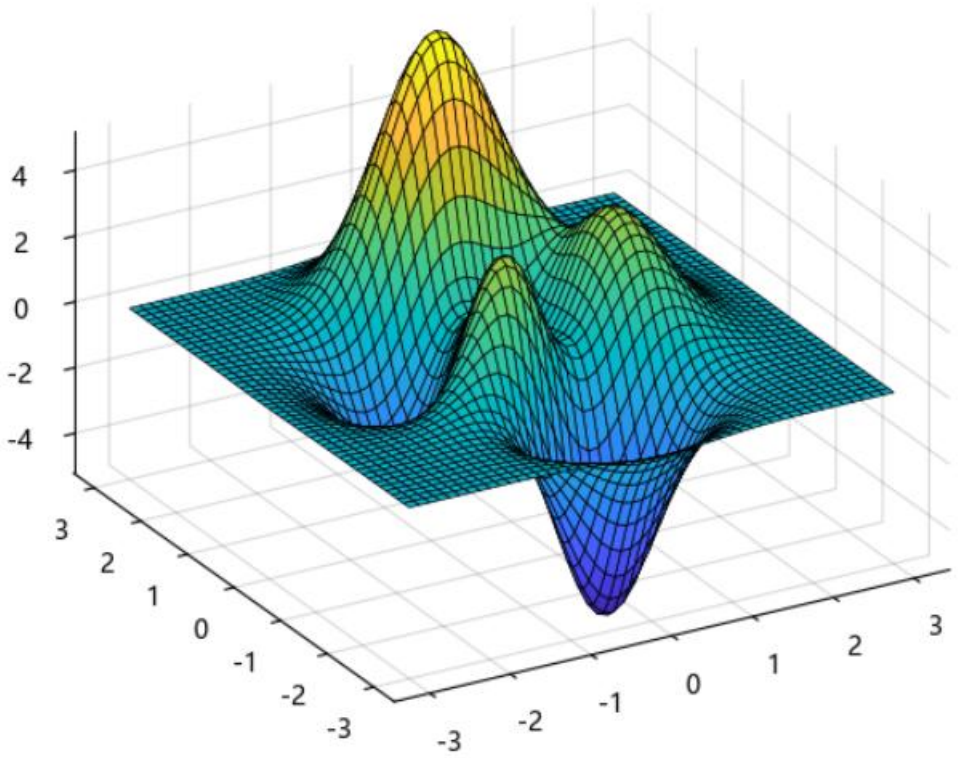
___ = zlim(target,___)使用由 target 指定的坐标区或图，而不是当前坐标区。指定 target 作为上述任何语法的第一个输入参数。如果原始语法支持输出参数，则可以包含输出参数。zlim(target,auto=true)。

C. 示例

(1). 设置 z 坐标轴范围

绘制一个 peaks 函数图，并将 z 坐标轴范围设置为从 -5 至 5。

```
X,Y,Z = peaks();
surf(X,Y,Z);
zlim([-5 5])
```



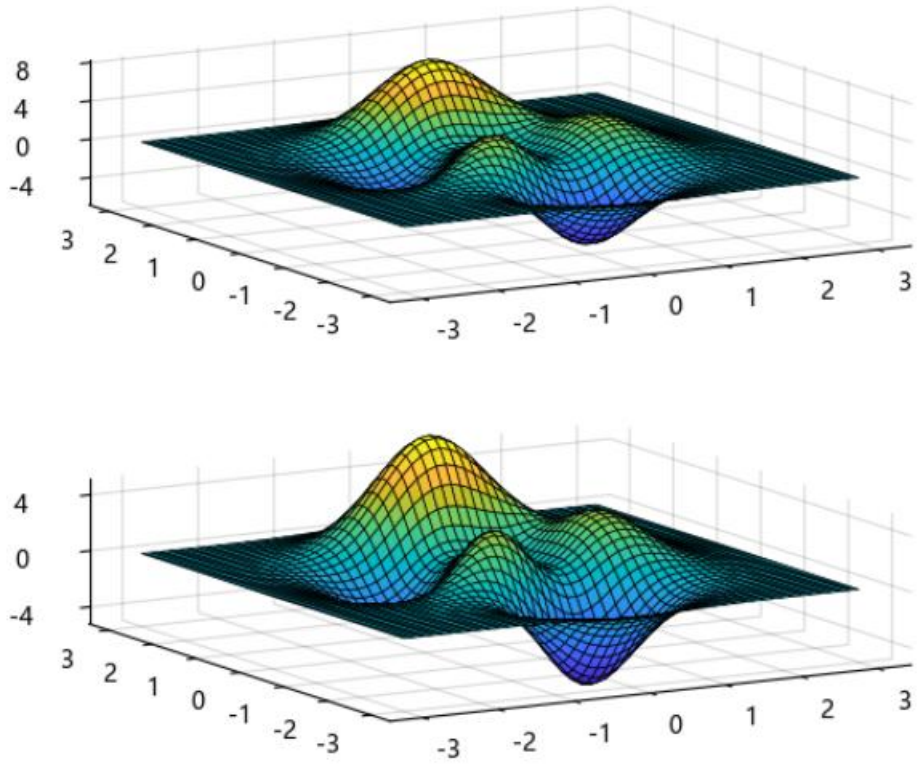
(2). 设置特定坐标区的 z 坐标轴范围

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该

对象返回为 ax1。通过将 ax1 传递给 surf 函数来创建顶部绘图。重复该过程以创建底部绘图。在每个坐标区中绘制数据。通过指定 ax2 作为 zlim 的第一个输入参数，设置底部图的 z 坐标轴范围。

```
ax1 = subplot(2,1,1,projection="3d")
X,Y,Z = peaks();
surf(ax1, X,Y,Z);

ax2 = subplot(2,1,2,projection="3d")
X,Y,Z = peaks();
surf(ax2, X,Y,Z);
zlim(ax2, [-5 5])
```

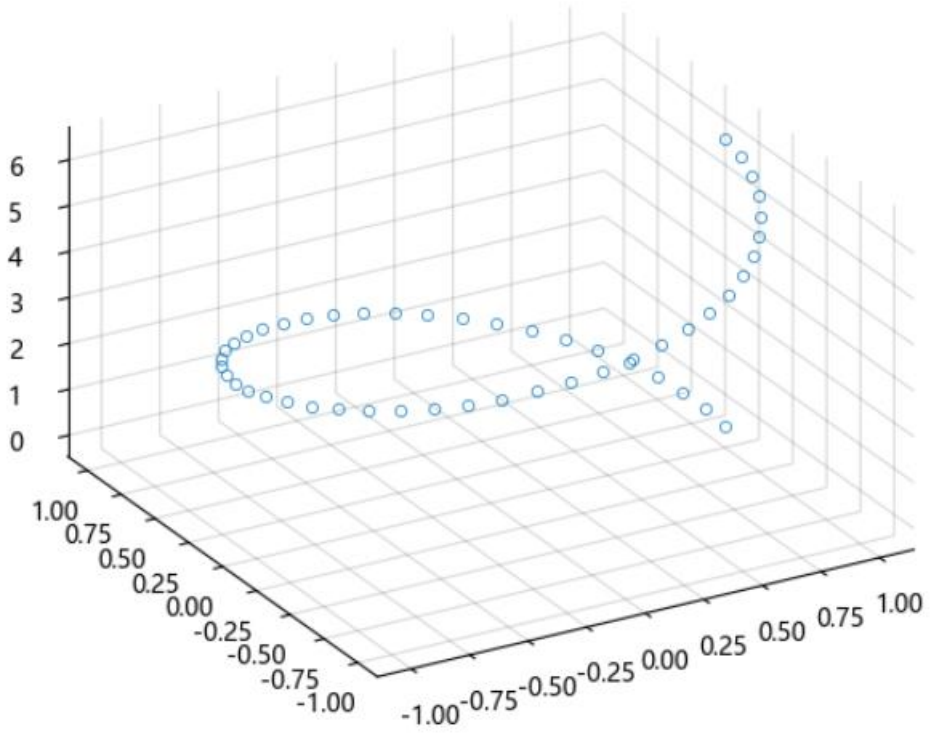


(3). 保持当前 z 坐标轴范围

当将更多绘图添加到坐标区时，使用手动模式可保持当前 z 坐标轴范围。

首先，绘制一个三维散点图。

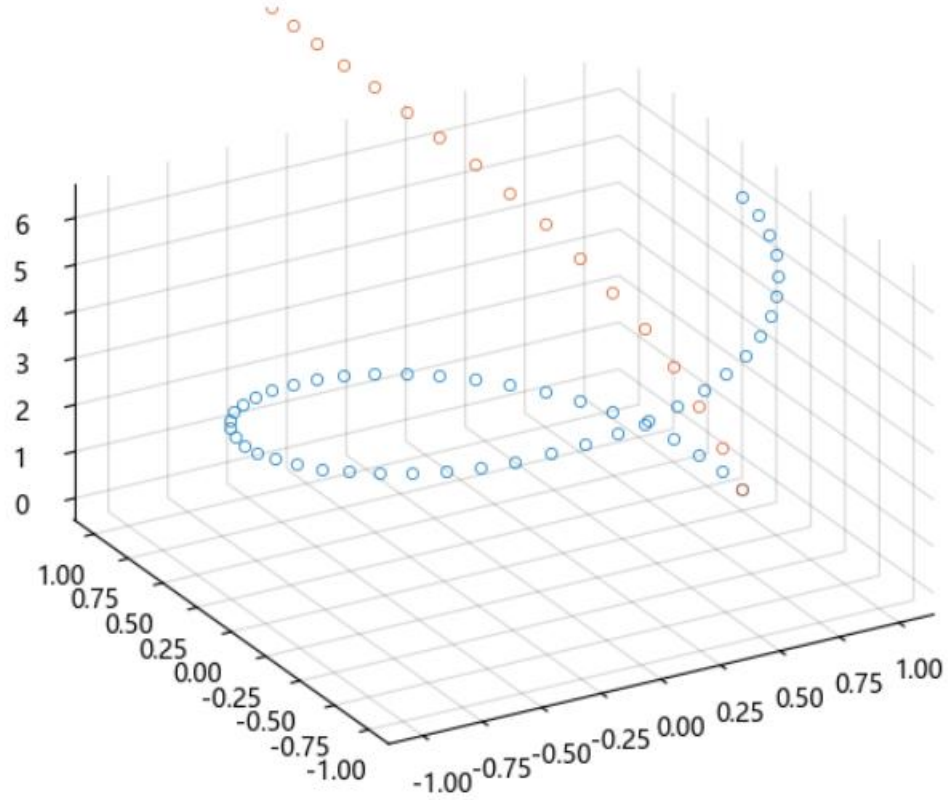
```
theta = LinRange(0, 2 * pi, 50);
X = cos.(theta);
Y = sin.(theta);
Z = theta;
scatter3(X, Y, Z)
```



将 z 坐标轴范围模式设置为手动，以便范围不发生变化。使用 hold("on") 将第二个绘图添加到坐标区。

```
zlim("manual")

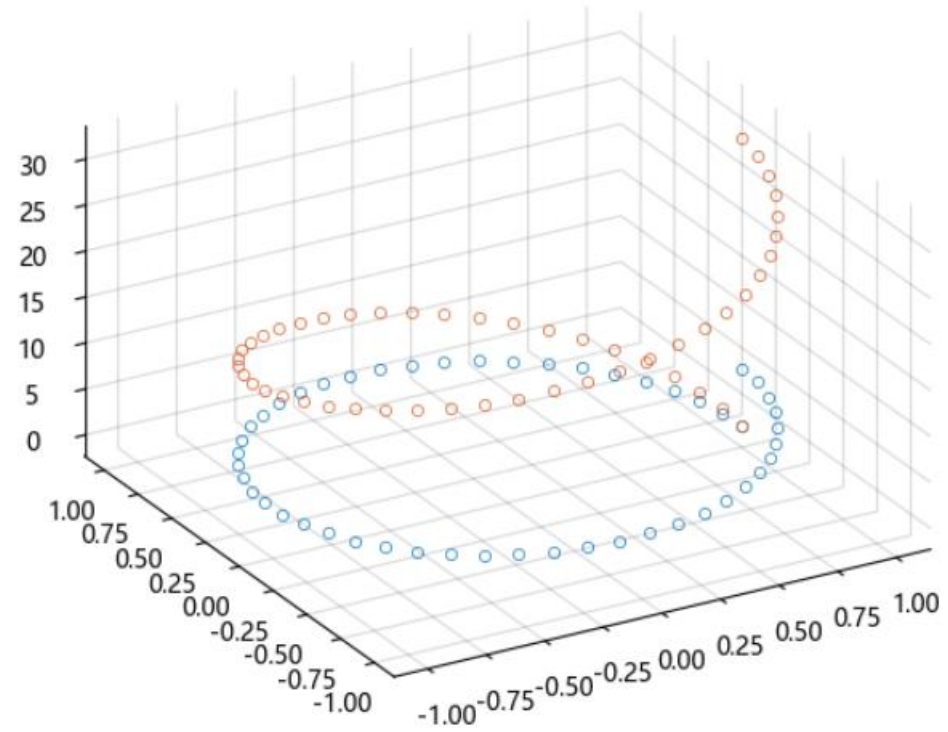
hold("on")
Znew = 5 .* theta;
scatter3(X, Y, Znew)
hold("off")
```



z 坐标轴范围不会更新以包含该新绘图。

通过将模式重置为自动来重新切换回自动更新的范围。

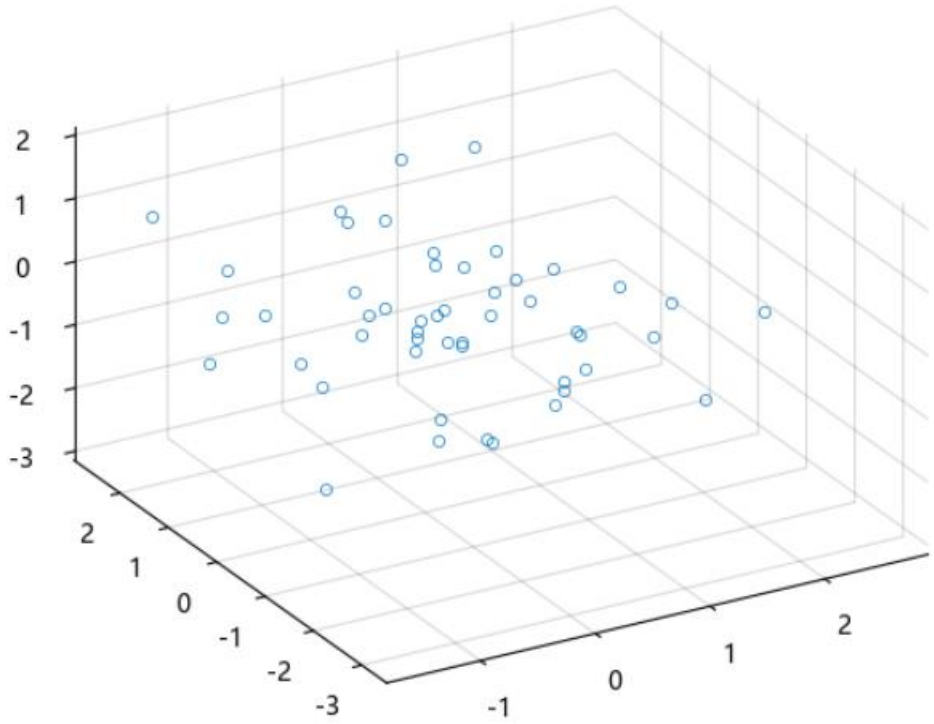
```
zlim("auto")
```



(4). 返回 z 坐标轴范围

创建一个由随机数据组成的散点图。返回 z 坐标轴范围的值。

```
x = randn(50,1);
y = randn(50,1);
z = randn(50,1);
scatter3(x,y,z)
```




```
zl = zlim()
println(zl)
```

```
zl = (-3.0368535713681877, 2.0344936272532332)
```

axis

设置坐标轴范围和纵横比

A. 语法

```
axis(limits)

axis(style)

lim = axis()

___ = axis(ax,___)
```

B. 说明

axis(limits)指定当前坐标区的范围。以包含 4 个、6 个元素的向量形式指定范围。

axis(style)使用预定义样式设置轴范围和尺度。例如，将样式指定为 equal 以便沿着每个坐标轴使用相等的数据单位长度。

lim=axis()返回当前坐标区的 x 轴和 y 坐标轴范围。对于三维坐标区，还会返回 z 坐标轴范围。对于极坐标区，它返回 theta 轴和 r 坐标轴范围。

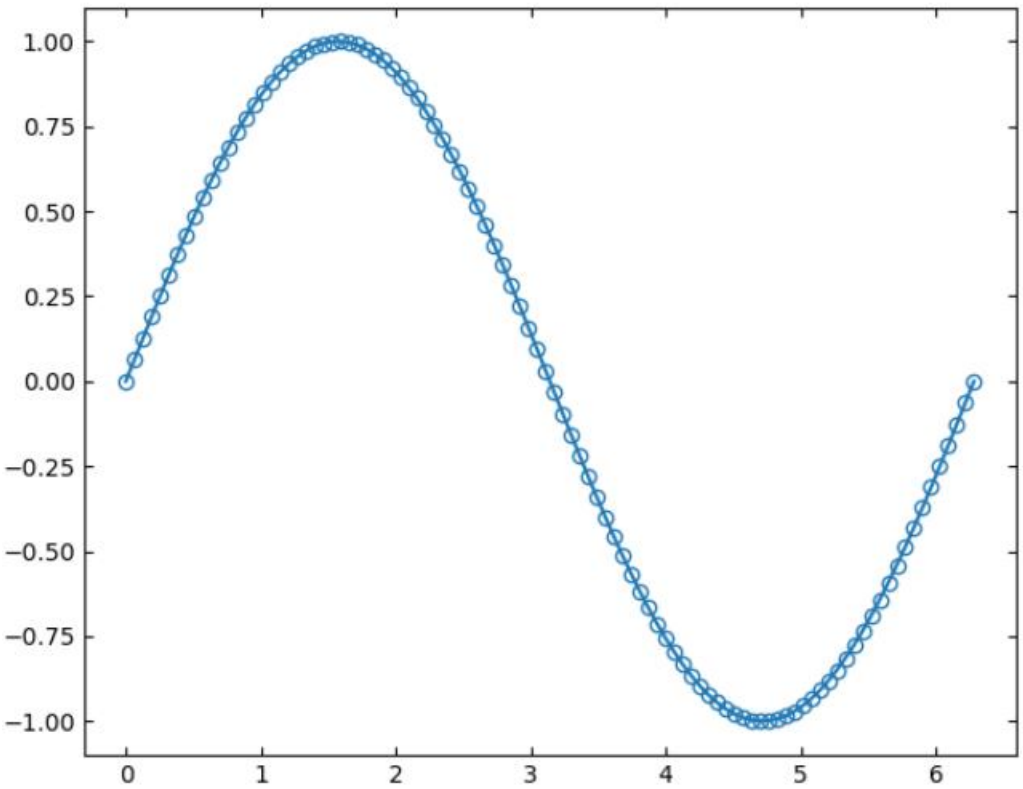
___ = axis(ax,___)使用 ax 指定的坐标区或极坐标区，而不是使用当前坐标区。指定 ax 作为上述任何语法的第一个输入参数。将字符向量类型的输入参数用双引号引起来，例如 axis(ax,"equal")。

C. 示例

(1). 设置坐标轴范围

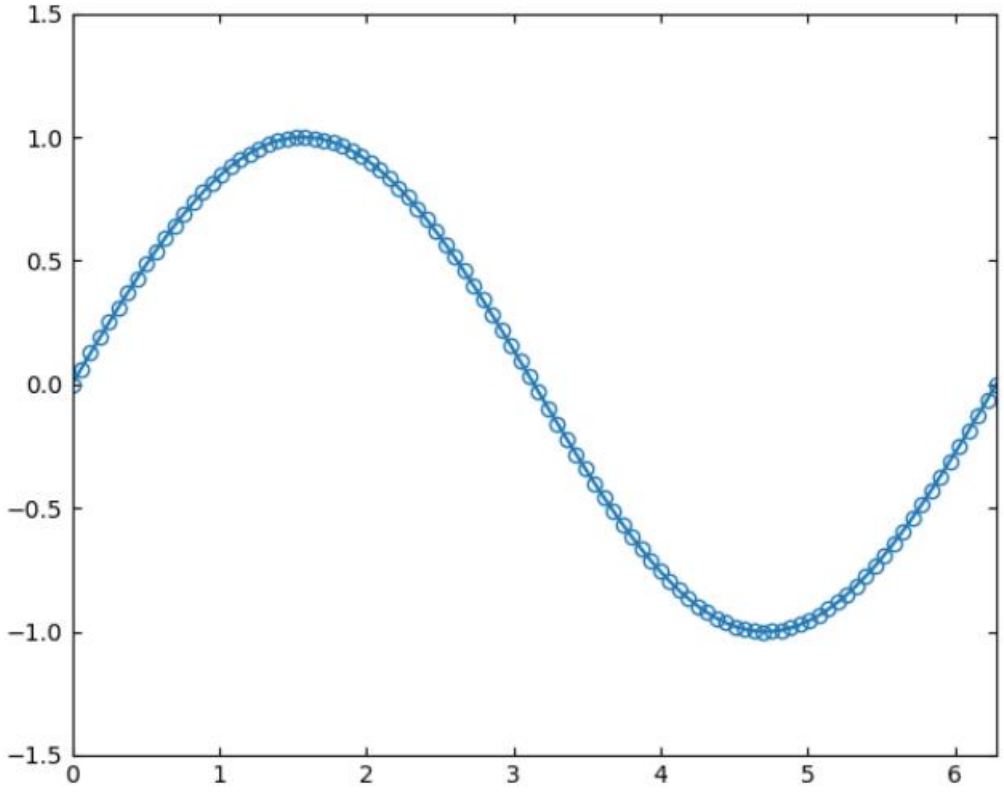
绘制正弦函数。

```
x = LinRange(0, 2 * pi, 100);
y = sin.(x);
plot(x, y, "-o")
```



更改坐标轴范围，使 x 轴的范围从 0 到 2π，y 轴的范围从 -1.5 到 1.5。

```
axis([0 2*pi -1.5 1.5])
```

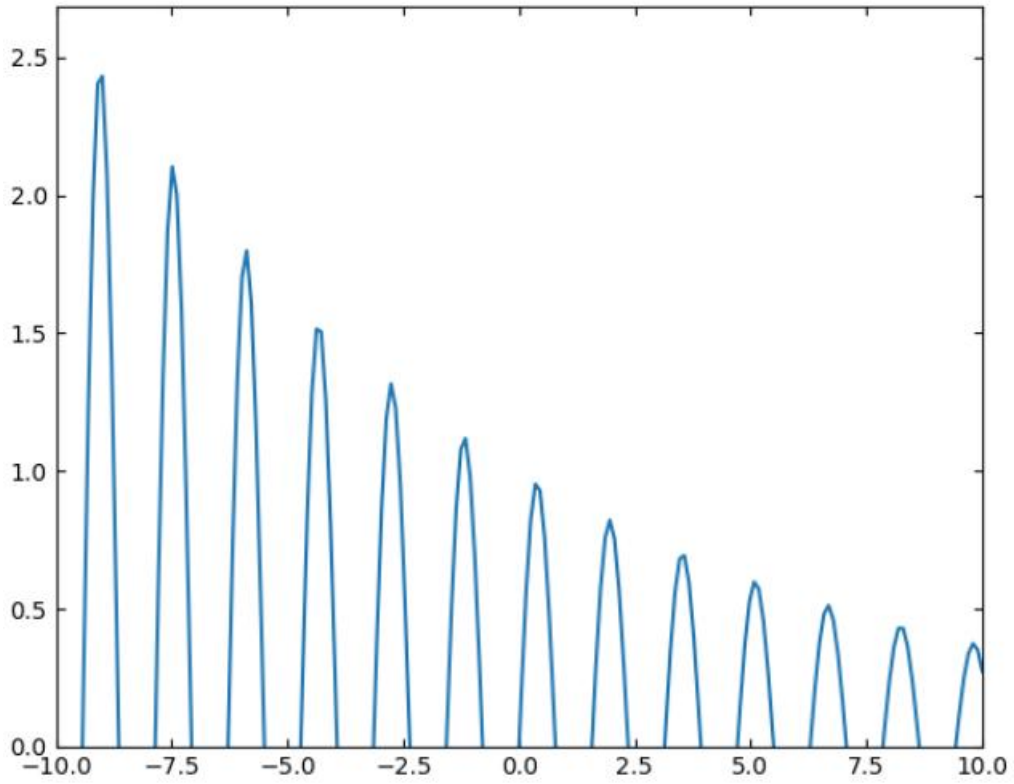


(2). 使用半自动坐标轴范围

创建一个绘图。设置 x 轴的范围和 y 轴的最小范围。对最大的 y 轴范围使用自动计算的值。

```
x = LinRange(-10, 10, 200);
y = sin.(4 * x) ./ exp.(0.1 * x);
plot(x, y)

yl = ylim()
axis([-10 10 0 yl[2]])
println(yl)
```



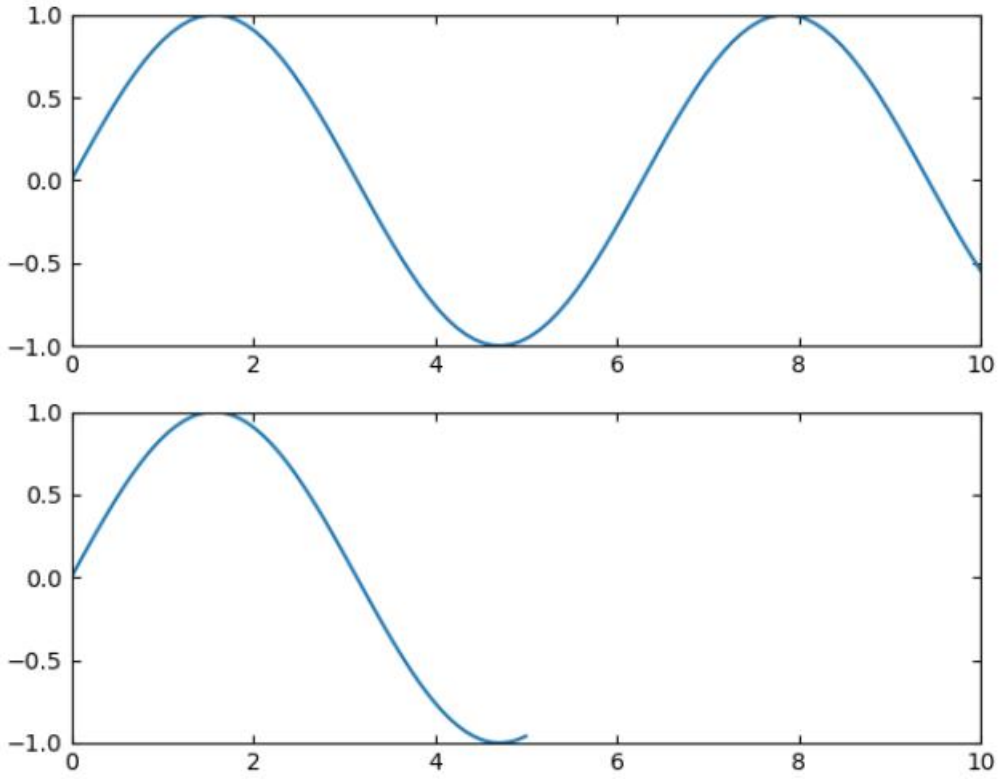
(3). 设置多个坐标轴的坐标轴范围

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。通过将 ax1 传递给 plot 函数来创建顶部绘图。重复该过程以创建底部绘图。在每个坐标区中绘制数据。然后将两个坐标区的轴范围设置为相同的值。

```
ax1 = subplot(2, 1, 1)
x1 = LinRange(0, 10, 100);
y1 = sin.(x1);
plot(ax1, x1, y1)

x2 = LinRange(0, 5, 100);
y2 = sin.(x2);
ax2 = subplot(2, 1, 2)
plot(ax2, x2, y2)

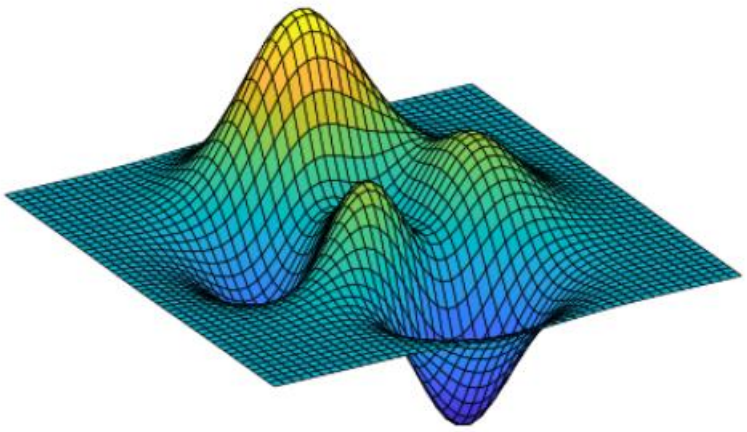
axis([ax1 ax2],[0 10 -1 1])
```



(4). 显示绘图而不显示坐标区背景

绘制曲面而不显示坐标区的线条和背景。

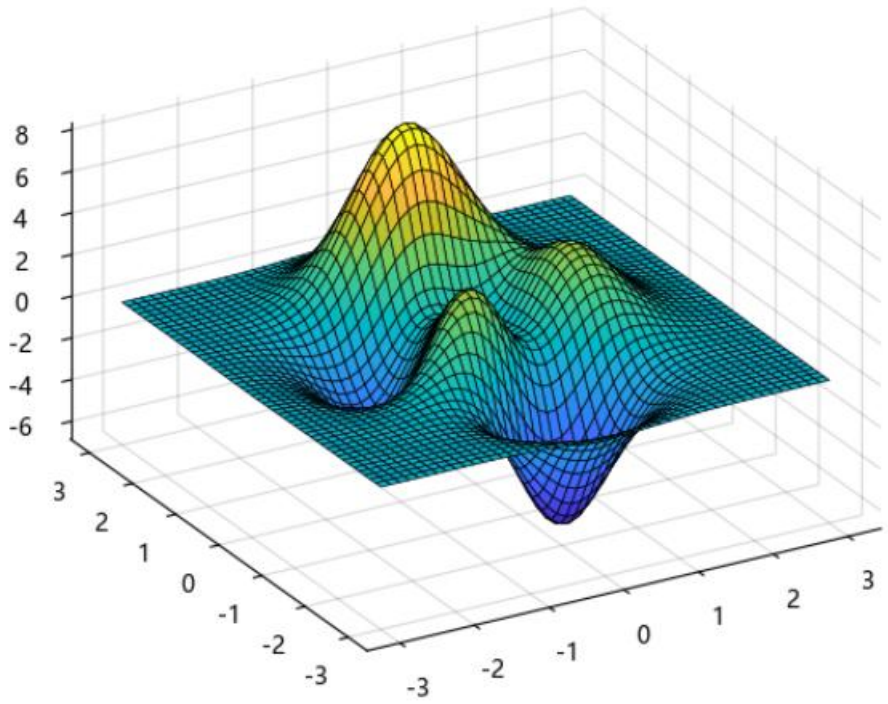
```
X,Y,Z = peaks()
surf(X,Y,Z)
axis("off")
```



(5). 使用紧凑的坐标轴范围并返回值

绘制一个曲面。将坐标轴范围设置为等于数据范围，这样绘图可以扩展到坐标区边缘。

```
X,Y,Z = peaks()
surf(X,Y,Z)
axis("tight")
```



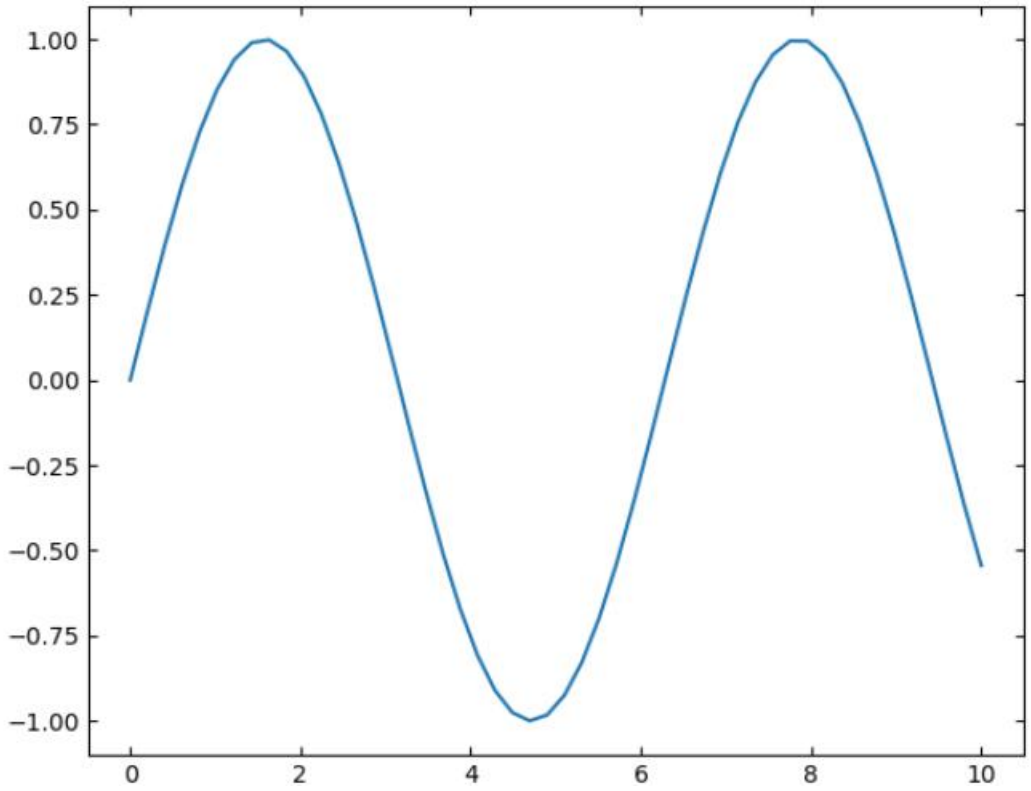

```
l = axis()
```

```
(-3.3, 3.3, -3.3, 3.3, -6.546644524320429, 8.075173545159231)
```

(6). 添加新绘图时保留当前的坐标轴范围

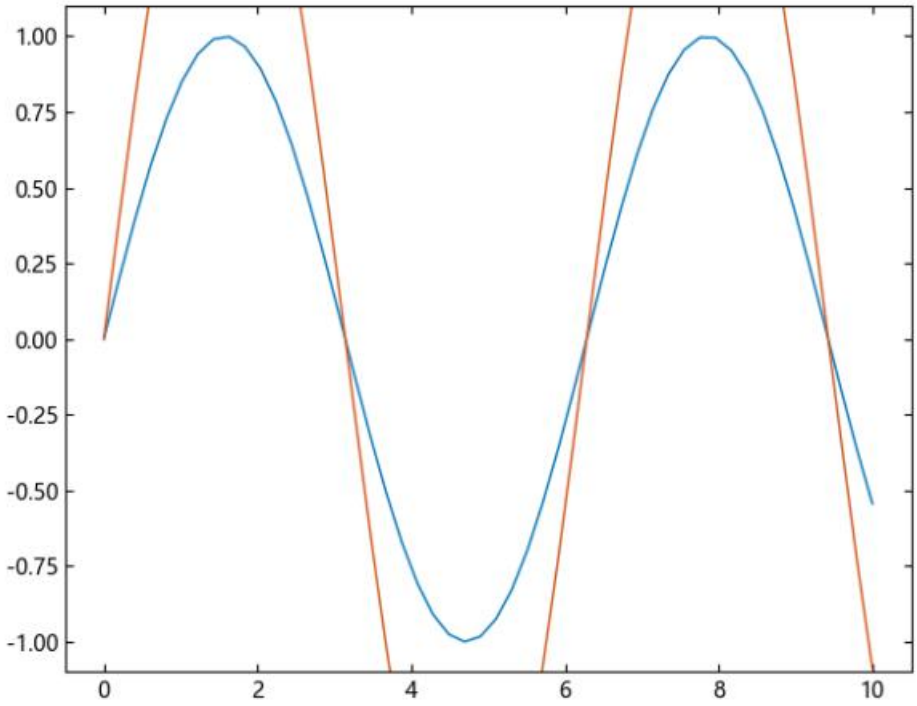
绘制一个正弦波。

```
x = LinRange(0, 10, 50);
y = sin.(x);
plot(x, y)
```



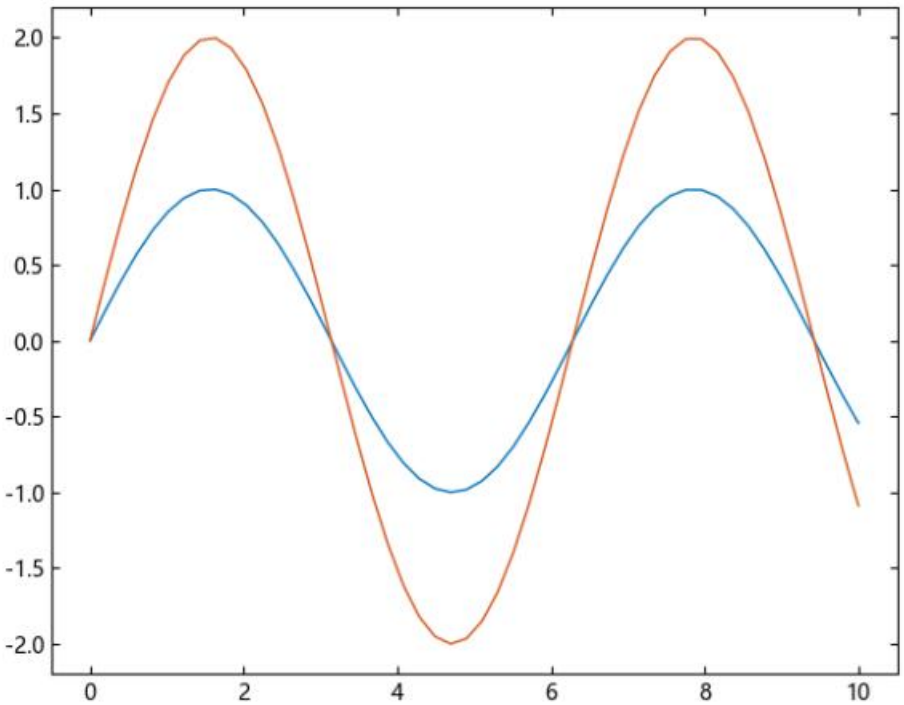
使用 hold("on") 将另一正弦波添加到坐标区中。通过将范围模式设置为手动来保留当前的坐标轴范围。

```
y2 = 2 * sin.(x);
hold("on")
axis("manual")
plot(x, y2)
hold("off")
```



如果您希望由这些坐标区自动选择合适的范围，请将范围模式设置回自动。

```
axis("auto")
```



daspect

控制沿每个轴的数据单位长度

A. 语法

```
daspect(ratio)
d = daspect()

daspect("auto")

___ = daspect(ax,___)
```

B. 说明

daspect(ratio)设置当前坐标区的数据纵横比。数据纵横比是沿 x 轴、y 轴和 z 轴的数据单位的相对长度。指定 ratio 为一个由正值组成的三元素向量，这些正值表示沿每个轴的数据单位的相对长度。例如，[1,2,3]表示沿 x 轴从 0 到 1 的长度等于沿 y 轴从 0 到 2 的长度和沿 z 轴 0 到 3 的长度。若要在所有方向上采用相同的数据单位长度，请使用[1,1,1]。

d=daspect()返回当前坐标区的数据纵横比。

daspect("auto") 设置自动模式，允许坐标区选择数据纵横比。模式必须为自动，才能启用坐标区的“伸展填充”功能。

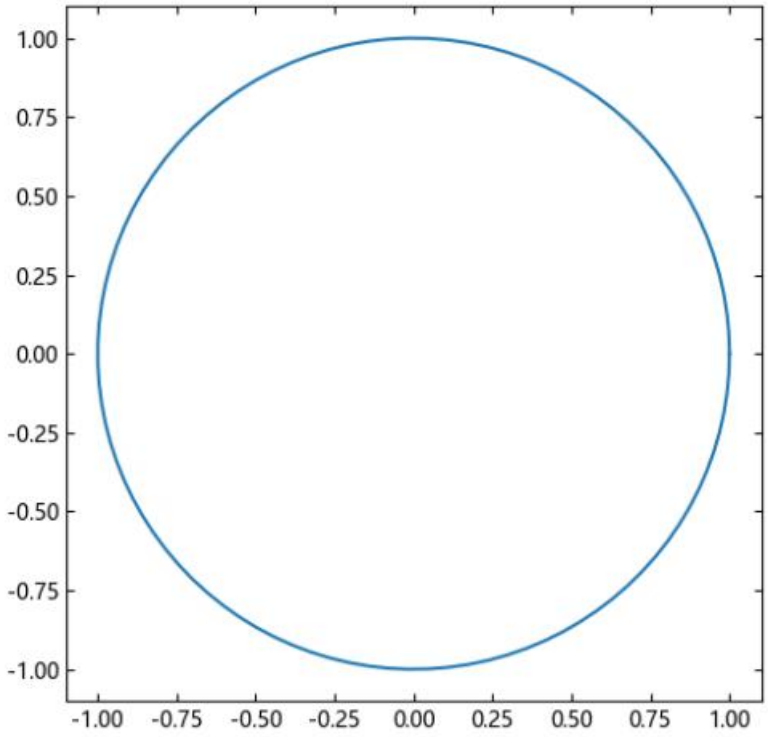
___ = daspect(ax,___)使用 ax 指定的坐标区，而不是使用当前坐标区。指定坐标区作为以上任何语法的第一个输入参数。

C. 示例

(1). 沿每个轴使用相等的数据单位长度

绘制一组数据。通过将数据纵横比设置为 [1,1,1]，在所有方向上使用相等的数据单位长度。

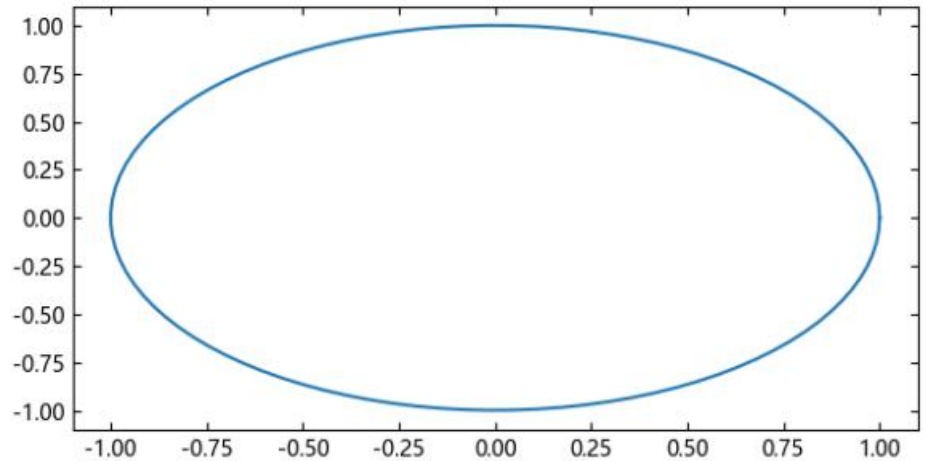
```
t = LinRange(0, 2 * pi, 100);
x = cos.(t);
y = sin.(t);
plot(x, y)
daspect([1 1 1])
```



(2). 沿每个轴使用不同的数据单位长度

绘制一组数据。设置数据纵横比，使 x 轴方向的一个数据单位的长度等于 y 轴方向和 z 轴方向（不显示）的两个数据单位的长度。

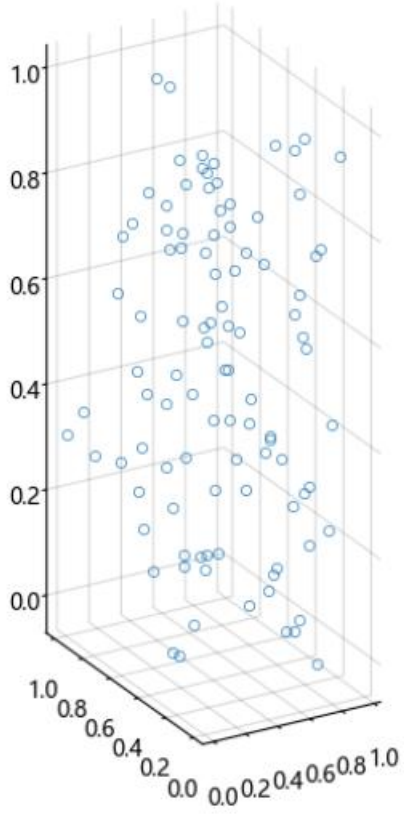
```
t = LinRange(0, 2 * pi, 100);
x = cos.(t);
y = sin.(t);
plot(x, y)
daspect([1 2 2])
```



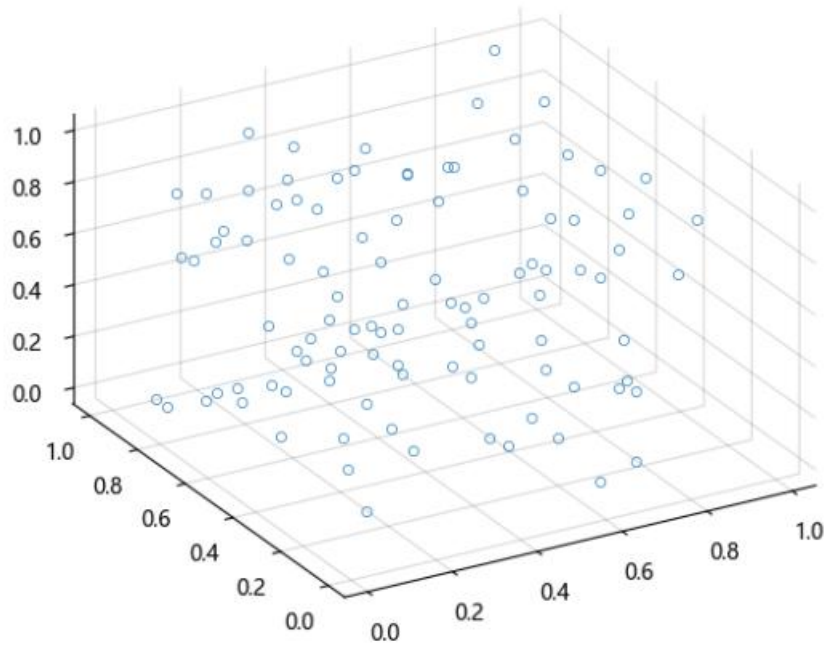
(3). 还原为默认的数据纵横比

创建一个由随机数据构成的三维散点图，并设置数据纵横比。然后还原为默认的数据纵横比。

```
X = rand(100,1);
Y = rand(100,1);
Z = rand(100,1);
s = scatter3(X,Y,Z)
daspect([3 2 1])
```




```
daspect("auto")
```

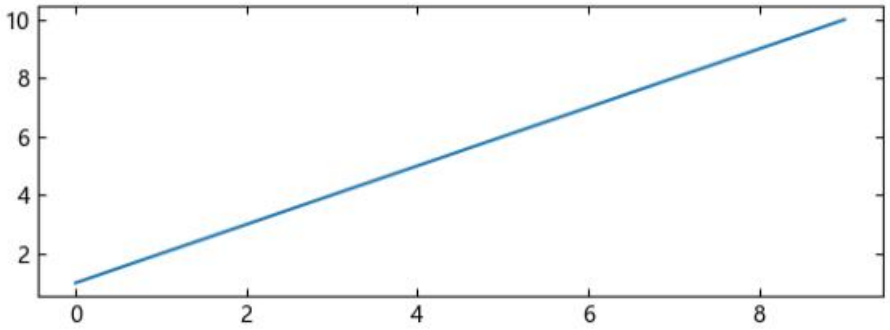


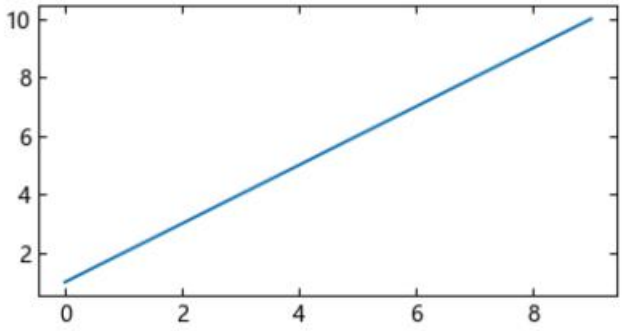
(4). 设置特定坐标区对象的数据纵横比

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，创建坐标区对象 ax1 和 ax2。在坐标区中绘图，并通过指定 ax2 为 daspect 函数的第一个输入参数，设置下部图的数据纵横比。

```
ax1 = subplot(2,1,1)
plot(ax1,1:10)

ax2 = subplot(2,1,2)
plot(ax2,1:10)
daspect(ax2,[1 2 1])
```





pbaspect

控制每个轴的相对长度

A. 语法

```
pbaspect(ratio)
pb = pbaspect()

pbaspect("auto")

___ = pbaspect(ax,___)
```

B. 说明

pbaspect(ratio)设置当前坐标区的图框纵横比。图框纵横比是 x 轴、y 轴和 z 轴的相对长度。将 ratio 指定为由正值组成的三元素向量，以表示 x 轴、y 轴和 z 轴长度的比率。例如，[3 1 1]指定 x 轴的长度等于 y 轴和 z 轴长度的三倍。如需轴长度在所有方向上都相等，请使用[1 1 1]。

pb=pbaspect()返回当前坐标区的图框纵横比。

pbaspect("auto") 设置自动模式，从而允许坐标区选择适当的图框纵横比。

模式必须为自动，才能启用坐标区的“伸展填充”功能。

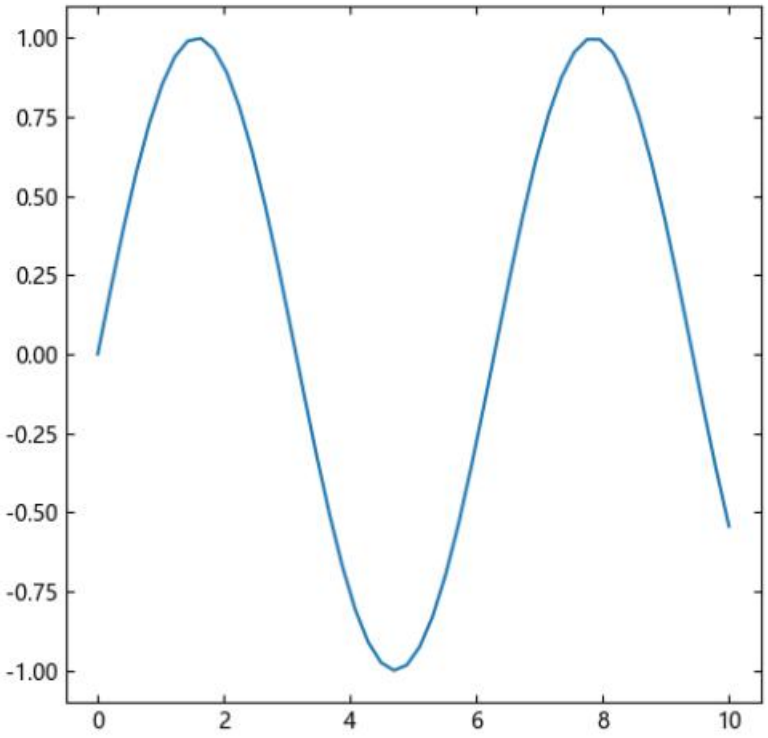
___ = pbaspect(ax,___)使用 ax 指定的坐标区，而不是使用当前坐标区。指定坐标区作为以上任何语法的第一个输入参数。

C. 示例

(1). 使用相等的轴长度

创建一个简单的线图。使 x 轴、y 轴和 z 轴（不显示）长度相等。

```
x = LinRange(0, 10, 50);
y = sin.(x);
plot(x, y)
```

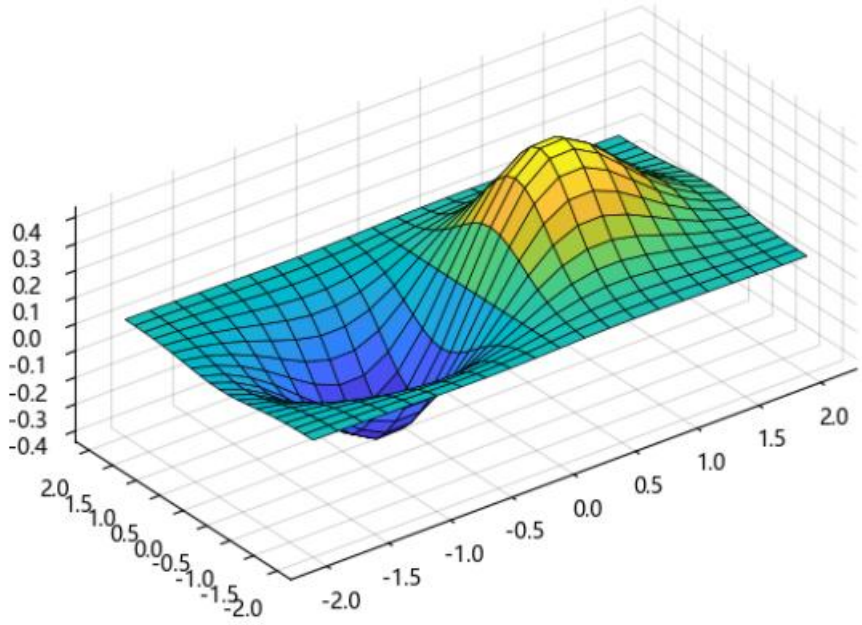


(2). 使用不同的轴长度

绘制一个曲面，并使 x 轴的长度是 y 轴和 z 轴的两倍。

```
xs = collect(-2:0.2:2)
ys = collect(-2:0.2:2)

x, y = meshgrid2(xs, ys);
z = x .* exp.(-x .^ 2 - y .^ 2);
surf(x, y, z)
pbaspect([2 1 1])
```

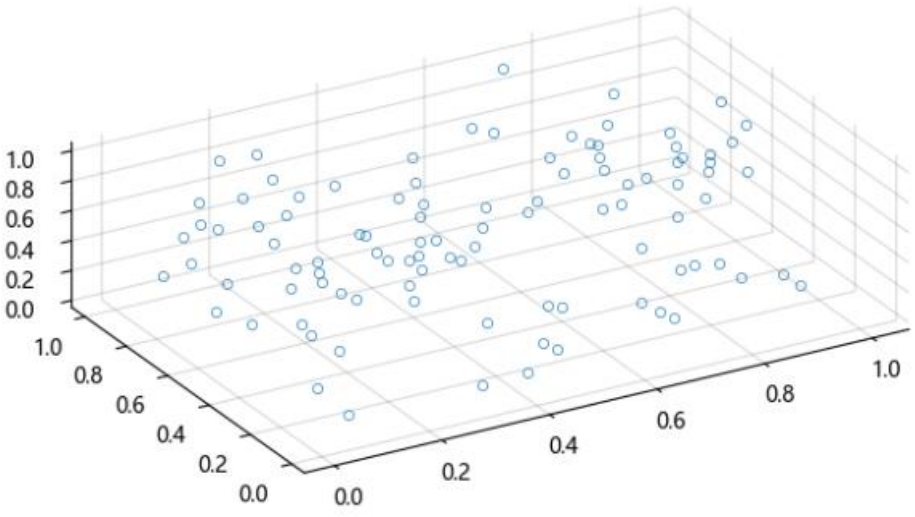


(3). 还原为默认图框纵横比

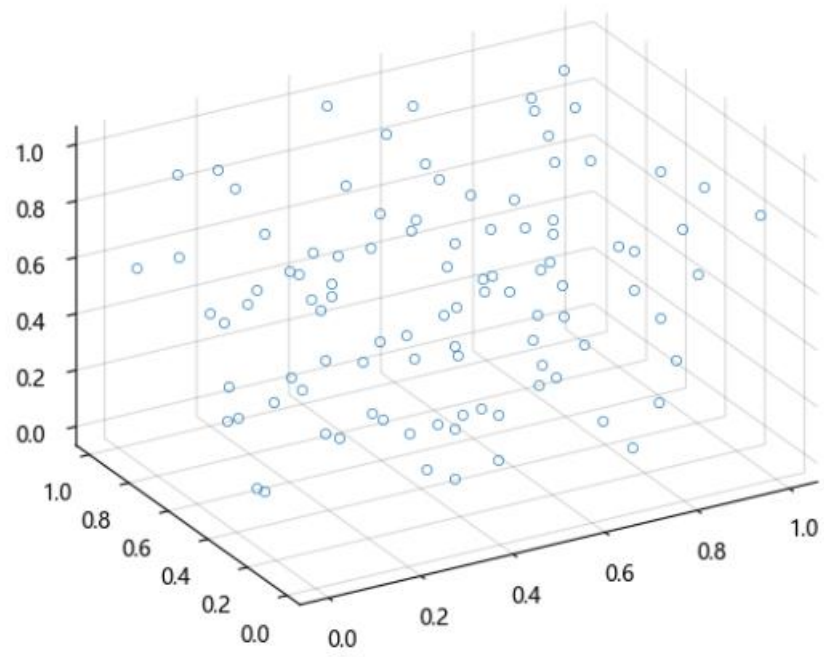
创建随机数据的三维散点图并设置图框纵横比。然后还原为默认图框纵横比。

```
X = rand(100,1);
Y = rand(100,1);
Z = rand(100,1);

scatter3(X,Y,Z)
pbaspect([3 2 1])
```



```
pbaspect("auto")
```

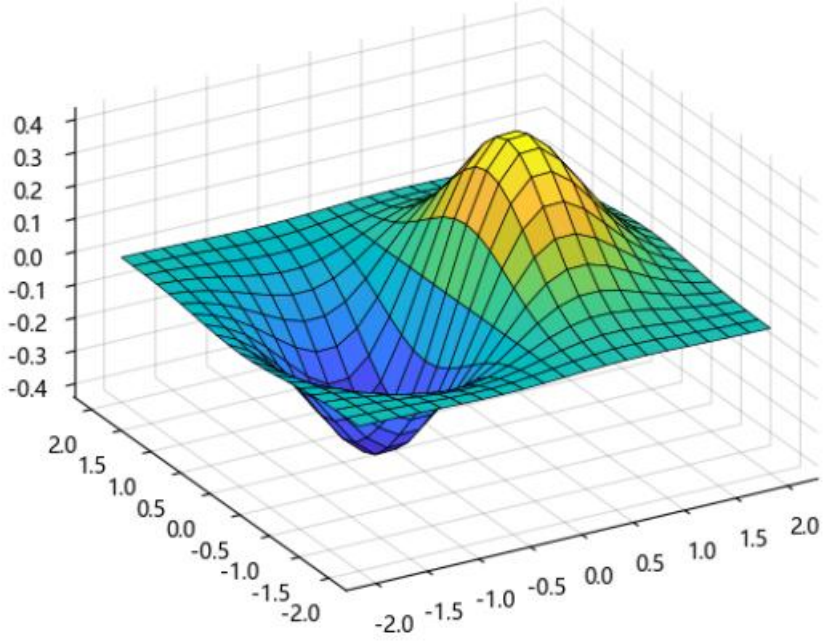


(4). 查询图框纵横比

创建一个曲面图。

```
xs = collect(-2:0.2:2)
ys = collect(-2:0.2:2)

x, y = meshgrid2(xs, ys);
z = x .* exp.(-x .^ 2 - y .^ 2);
surf(x, y, z)
```



查询图框纵横比。这些值指示 x 轴长度、y 轴长度与 z 轴长度的比率。

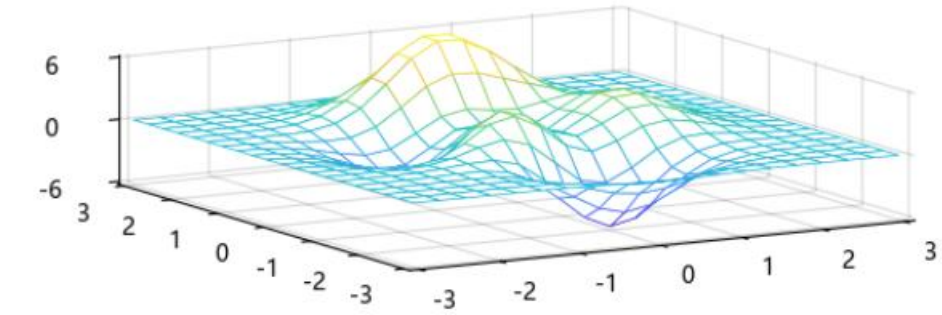
```
d = pbaspect()
```

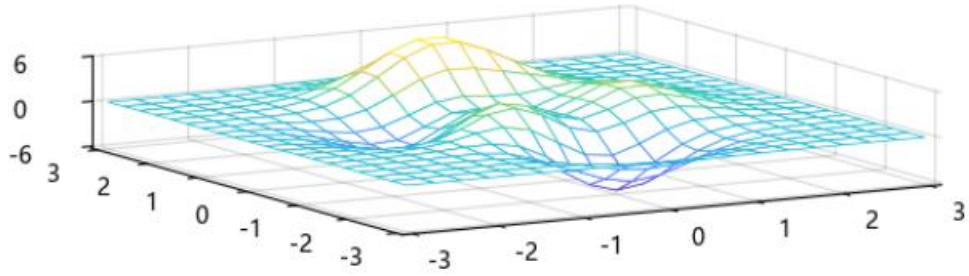
(5). 为特定坐标区设置图框纵横比

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，创建坐标区对象 ax1 和 ax2。在每个坐标区中绘制。然后通过将 ax2 指定为 pbaspect 函数的第一个输入参数，设置下部图的图框纵横比。

```
ax1 = subplot(2,1,1, projection="3d")
X,Y,Z = peaks(20)
mesh(X,Y,Z)

ax2 = subplot(2,1,2, projection="3d")
mesh(X,Y,Z)
```





3.3.2.2.2 网格线、刻度值和标签

函数名	简介
grid	显示或隐藏坐标区网格线
xticks	设置或查询 x 轴刻度值
yticks	设置或查询 y 轴刻度值
zticks	设置或查询 z 轴刻度值
xticklabels	设置或查询 x 轴刻度标签
yticklabels	设置或查询 y 轴刻度标签
zticklabels	设置或查询 z 轴刻度标签
xtickformat	指定 x 轴刻度标签格式
ytickformat	指定 y 轴刻度标签格式
ztickformat	指定 z 轴刻度标签格式
xtickangle	旋转 x 轴刻度标签
ytickangle	旋转 y 轴刻度标签

ztickangle	旋转 z 轴刻度标签

grid

显示或隐藏坐标区网格线

A. 语法

```
grid("on")
grid("off")
grid()
grid("minor")
grid(target,___)
```

B. 说明

grid("on") 显示 gca 命令返回的当前坐标区或图的主网格线。主网格线从每个刻度线延伸。

grid("off") 删除当前坐标区或图上的所有网格线。

grid() 切换改变主网格线的可见性。

grid("minor") 切换改变次网格线的可见性。次网格线出现在刻度线之间。并非所有类型的图都支持次网格线。

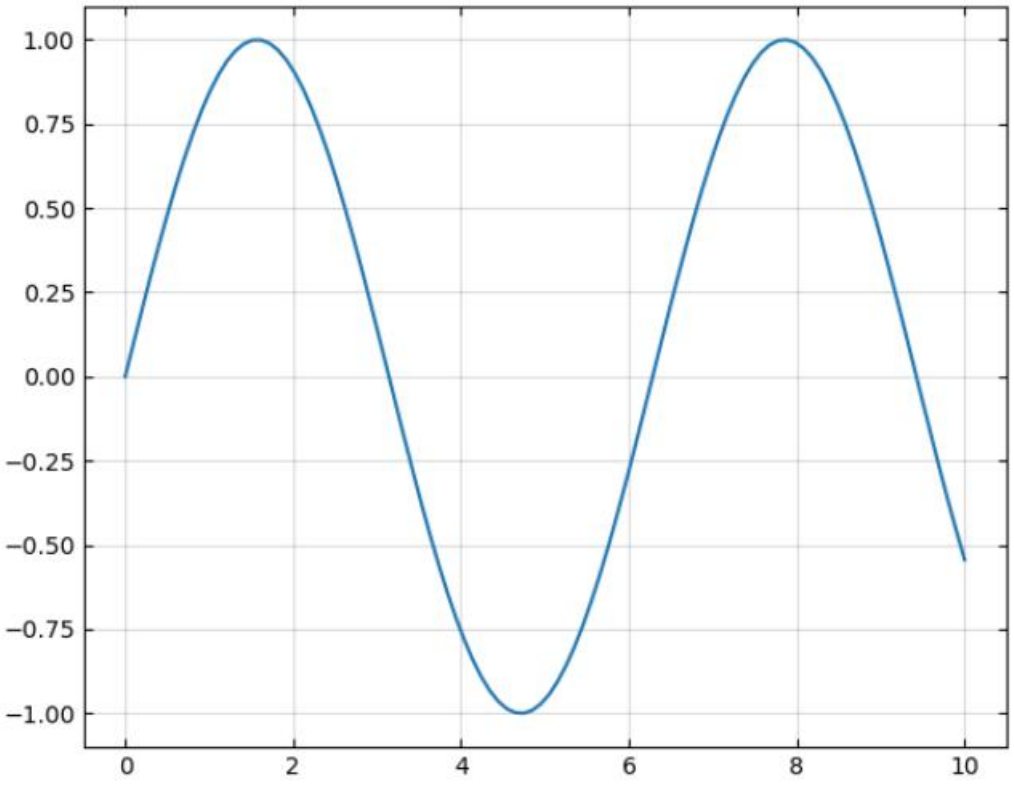
grid(target,___)使用由 target 指定的坐标区或图，而不是当前坐标区或图。指定 target 作为第一个输入参数。使用双引号将其他输入参数引起来，例如，grid(target,"on")。

C. 示例

(1). 显示网格线

显示正弦图的网格线。

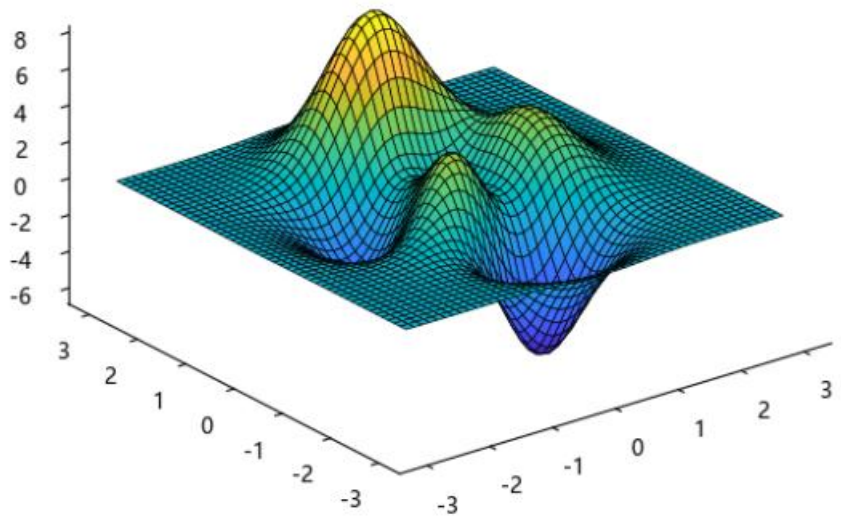
```
x = LinRange(0, 10, 100);
y = sin.(x);
plot(x, y)
grid("on")
```



(2). 删除网格线

创建一个曲面图并删除网格线。

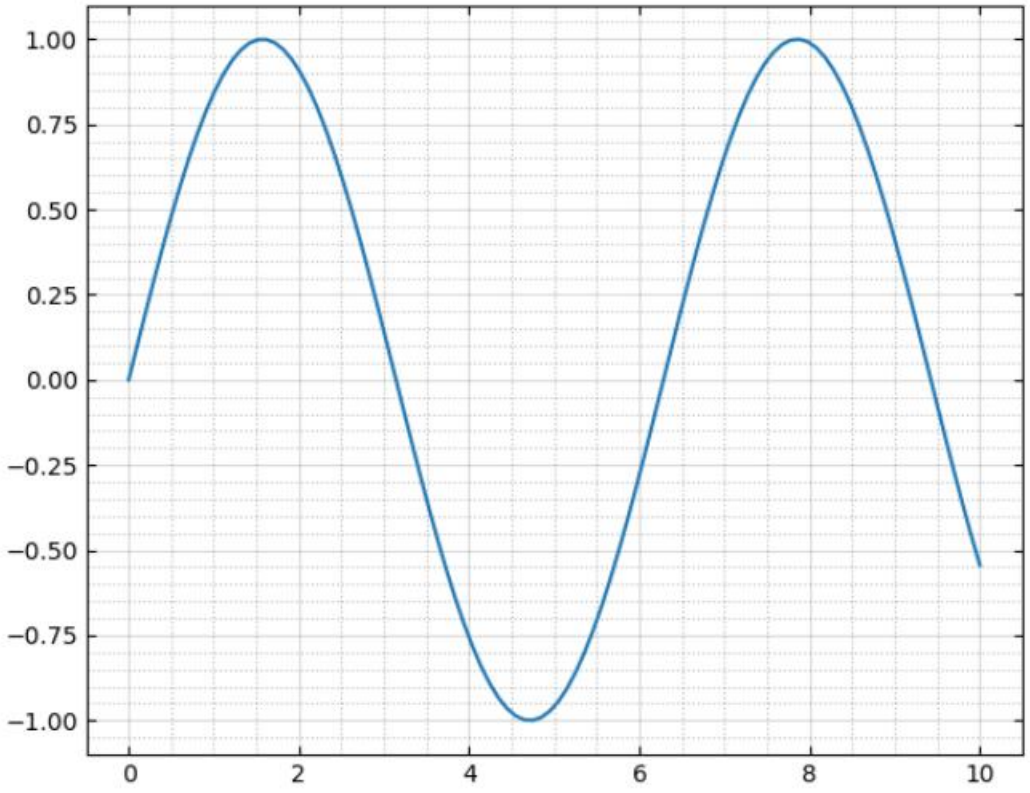
```
X, Y, Z = peaks()
surf(X, Y, Z)
grid("off")
```



(3). 显示主要和次网格线

显示正弦图的主网格线和次网格线。

```
x = LinRange(0, 10, 100);
y = sin.(x);
plot(x, y)
grid("on")
grid("on", "minor")
```



(4). 在特定坐标区上显示网格线

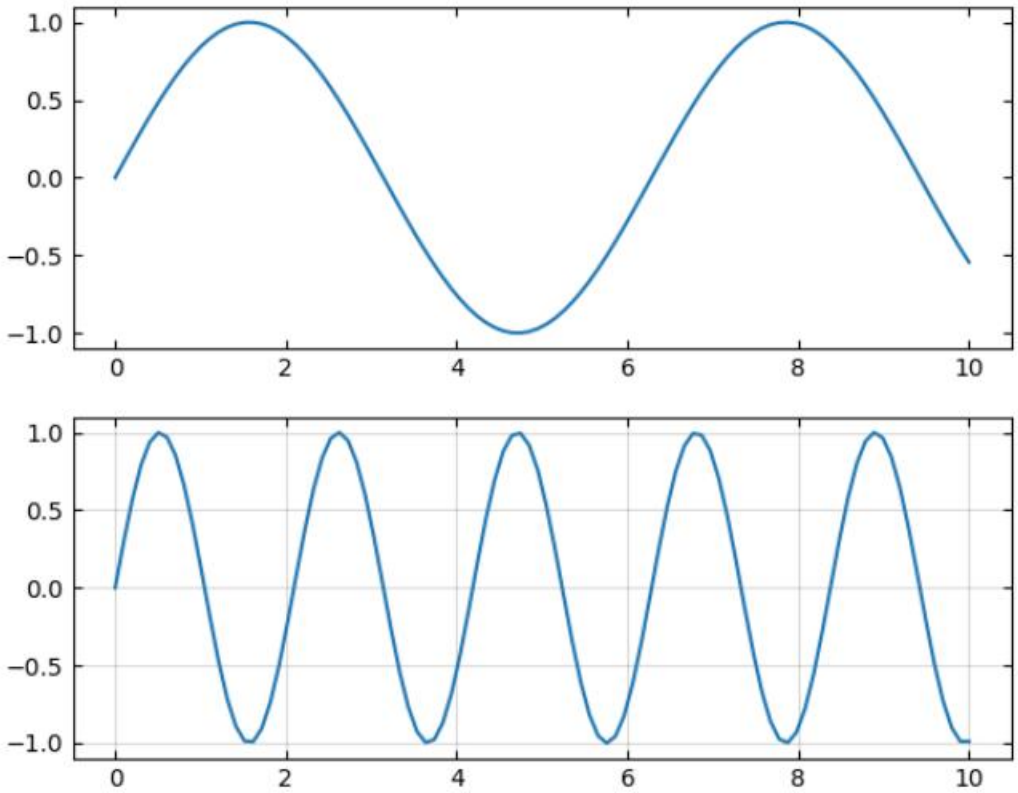
调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。通过将 ax1 传递给 plot 函数来创建顶部绘图。重复该过程以创建底部绘图。在每个坐标区中绘制数据。然后将 ax2 传递给 grid 函数，以在底部图中显示网格线。

```
x = LinRange(0, 10, 100);
y1 = sin.(x);
y2 = sin.(3 * x);

ax1 = subplot(2, 1, 1)
plot(ax1, x, y1)

ax2 = subplot(2, 1, 2)
```

```
plot(ax2, x, y2)
grid(ax2, "on")
```



xticks

设置或查询 x 轴刻度值

A. 语法

```
xticks(ticks)
xt = xticks()

xticks("auto")
xticks("manual")

___ = xticks(ax,___)
```

B. 说明

xticks(ticks)设置 x 轴刻度值，这些值是 x 轴上显示刻度线的位置。指定 ticks 为递增值向量；例如[0，2，4，6]。此命令作用于当前坐标区。

xt = xticks() 以向量形式返回当前 x 轴刻度值。

xticks("auto") 设置自动模式，使坐标区确定 x 轴刻度值。如果您更改了刻度值，然后又想将它们设置回默认值，则可以使用此选项。

xticks("manual") 设置手动模式，将 x 轴刻度值冻结在当前值。如果您希望在调整坐标区大小或者向坐标区添加新数据时保留当前刻度值，则可以使用此选项。

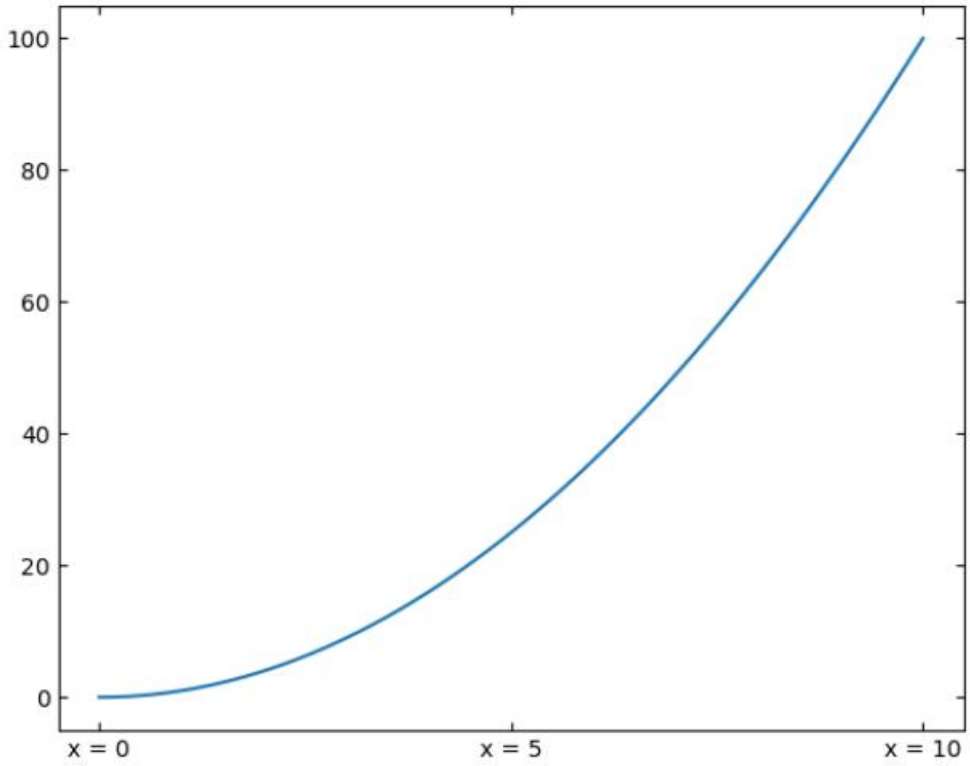
___ = xticks(ax,___)使用 ax 指定的坐标区，而不是使用当前坐标区。指定 ax 作为上述任何语法的第一个输入参数。

C. 示例

(1). 指定 x 轴刻度值和标签

创建一个线图。在值 0、5 和 10 处显示 x 轴的刻度线。然后为每个刻度线指定一个标签。

```
x = LinRange(0, 10, 100);
y = x .^ 2;
plot(x, y)
xticks([0, 5, 10])
xticklabels(["x = 0", "x = 5", "x = 10"])
```

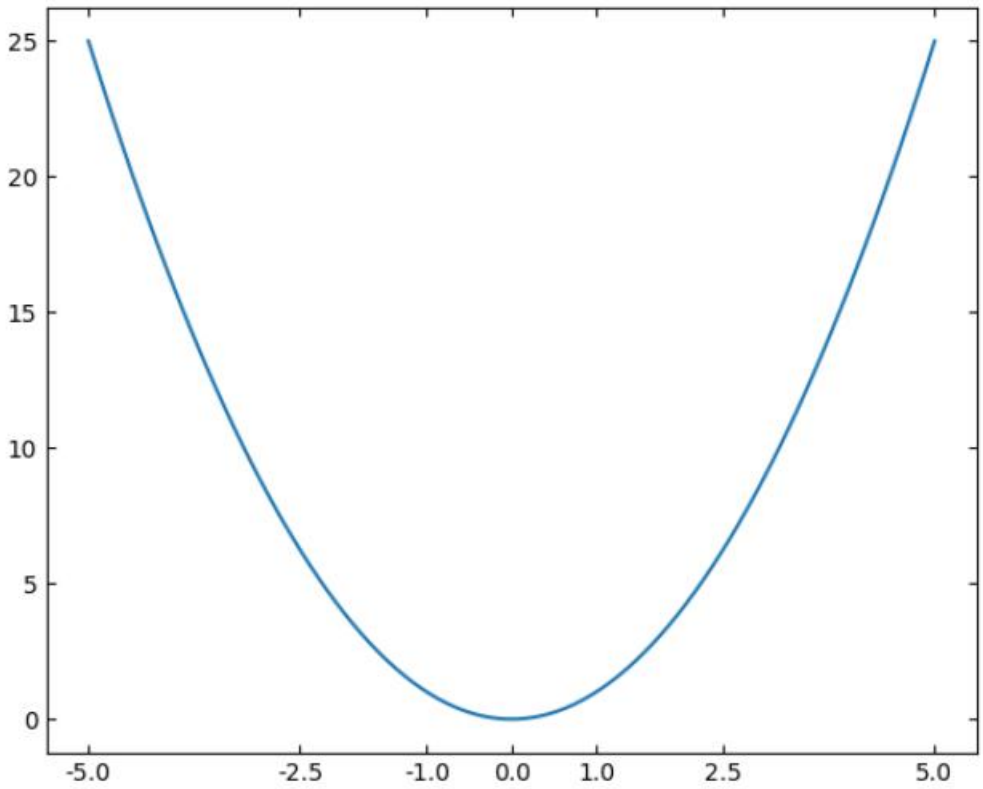


(2). 指定非均匀 x 轴刻度值

沿 x 轴以介于 -5 和 5 之间的非均匀值显示刻度线。Syslab 用数值标注

刻度线。

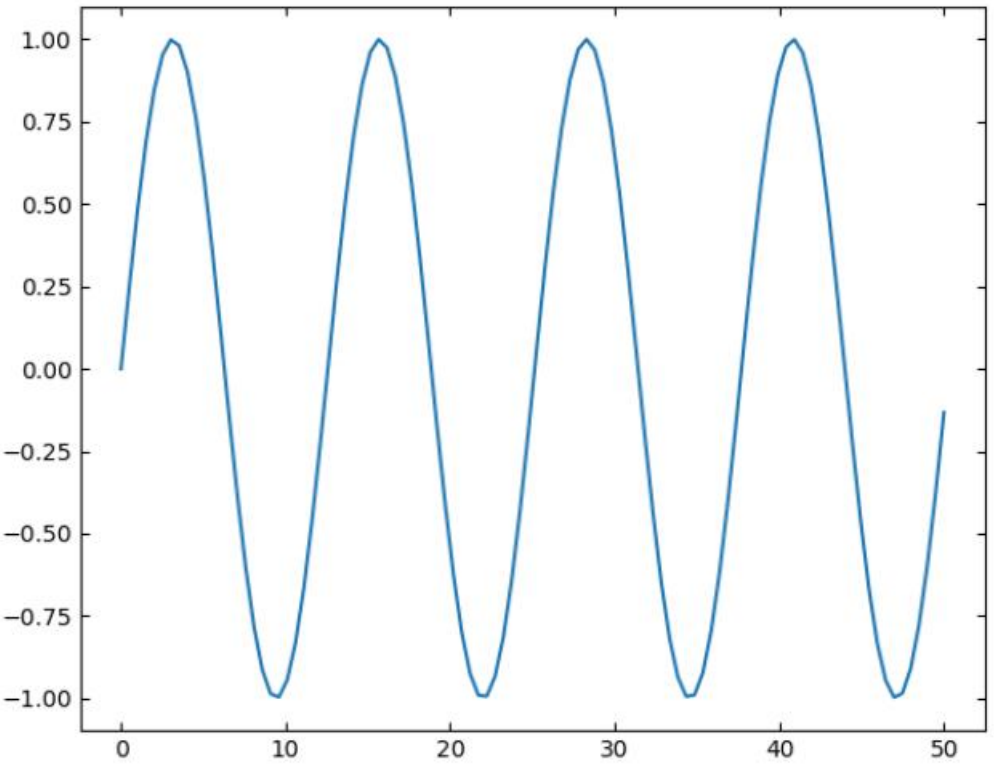
```
x = LinRange(-5, 5, 100);
y = x .^ 2;
plot(x, y)
xticks([-5, -2.5, -1, 0, 1, 2.5, 5])
```



(3). 以 10 为增量递增 x 轴刻度值

沿 x 轴以 10 为增量显示刻度线，从 0 开始，在 50 结束。

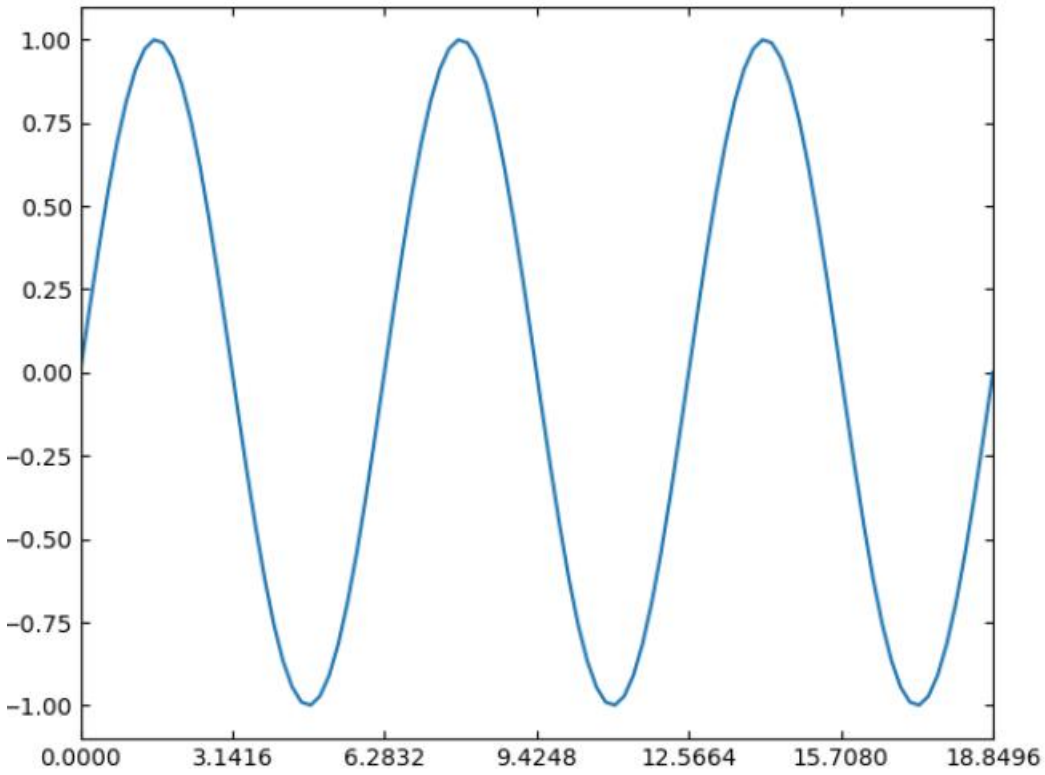
```
x = LinRange(0, 50, 100);
y = sin.(x / 2);
plot(x, y)
xticks(0:10:50)
```



(4). 以 Pi 为增量递增 x 轴刻度值

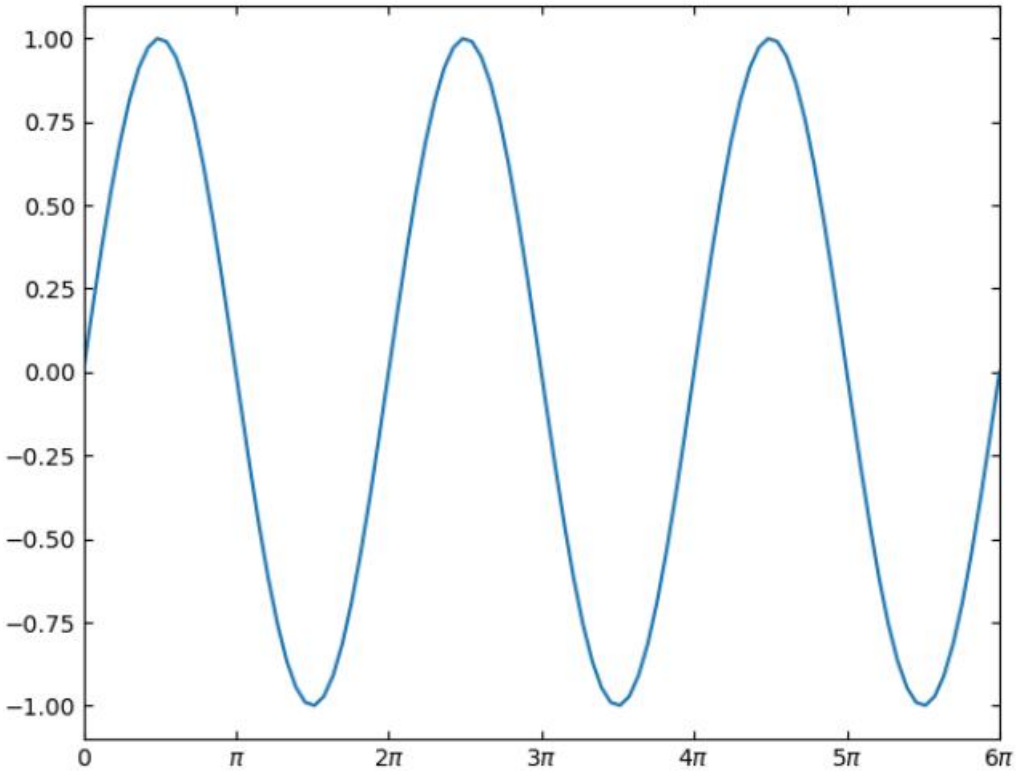
创建一个线图。将 x 坐标轴范围指定为 0 到 6π。然后，以 π 为增量显示 x 轴的刻度线。

```
x = LinRange(0, 6 * pi, 100);
y = sin.(x);
plot(x,y)
xlim([0,6*pi])
xticks(0:pi:6*pi)
xtickformat("%.4f")
```



Syslab 用数值标注刻度线。通过为每个标签指定文本来更改标签以显示 π 符号。

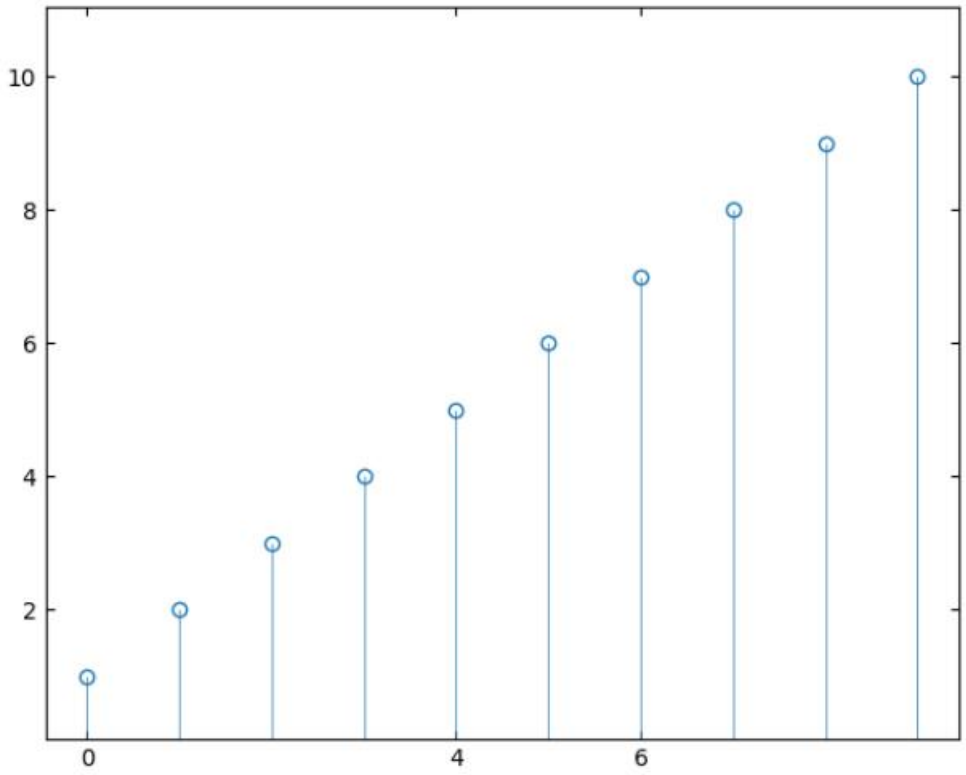
```
xticklabels(["0",raw"$\pi$",raw"2$\pi$",raw"3$\pi$",raw"4$\pi$",raw"5$\pi$",raw"6$\pi$"])
```



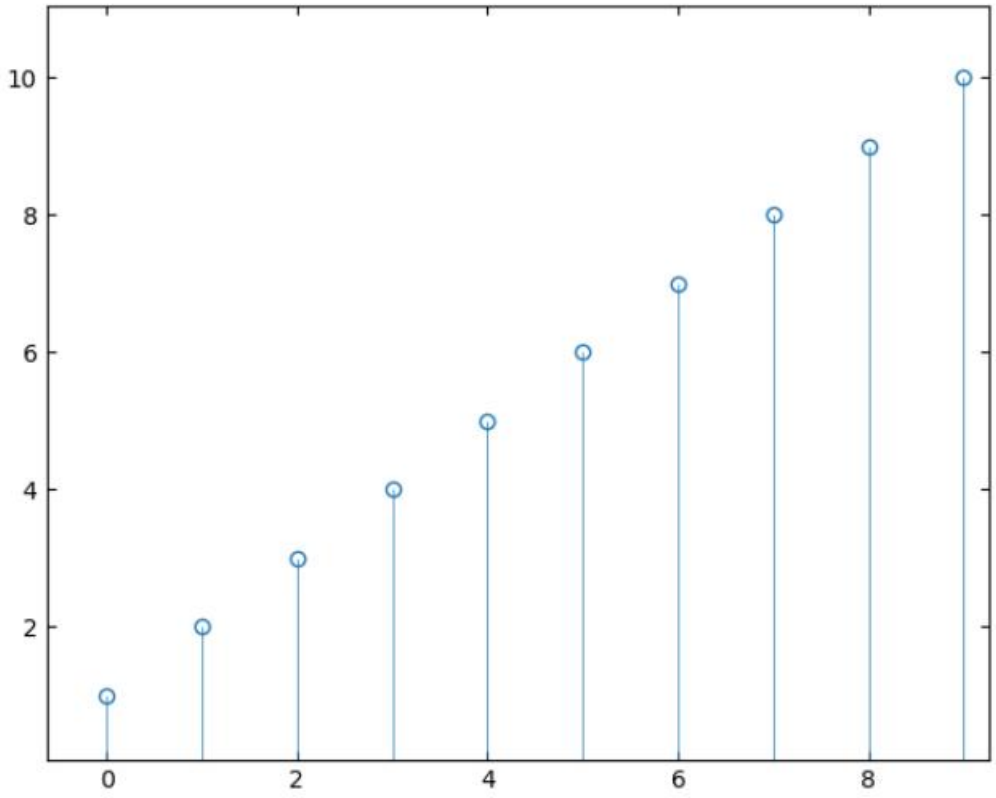
(5). 将 x 轴刻度值设置回默认值

创建一个针状图并指定 x 轴刻度值。然后，将 x 轴刻度值设置回默认值。

```
stem(1:10)
t=xticks([0,4,6,10])
```




```
xticks("auto")
```



(6). 指定特定坐标区的 x 轴刻度值

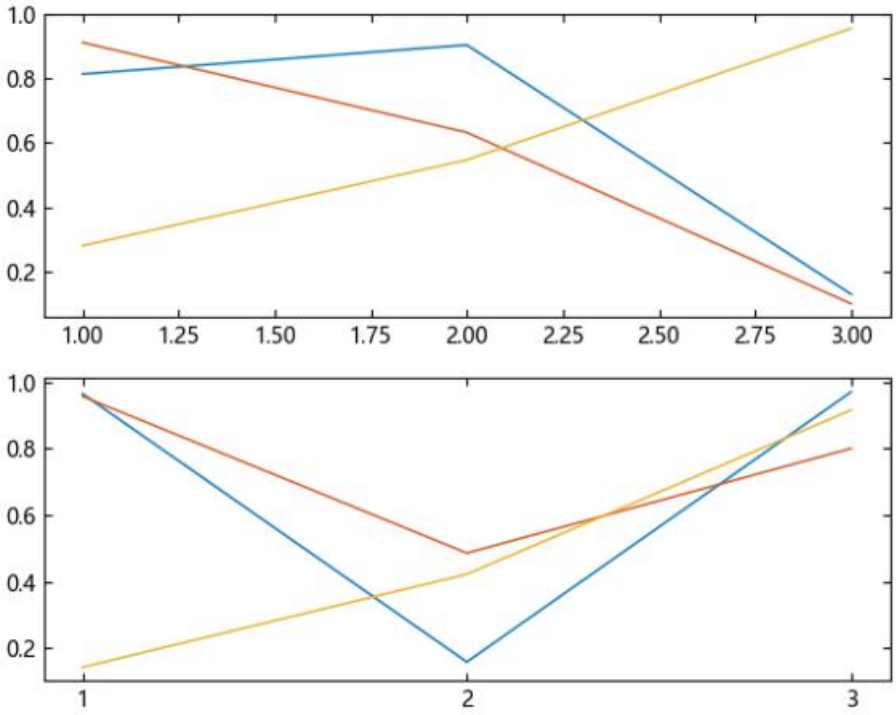
调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。通过将 ax1 传递给 plot 函数来创建顶部绘图。重复该过程以创建底部绘图。在每个坐标区中绘制数据。将随机数据绘制到每个坐标区中。然后通过将 ax2 作为 xticks 函数的第一个输入参数进行传递，设置下部图的 x 轴刻度值。

```
y = [
    0.8147    0.9134    0.2785
    0.9058    0.6324    0.5469
    0.1270    0.0975    0.9575
]

y2=[
    0.9649    0.9572    0.1419
    0.1576    0.4854    0.4218
    0.9706    0.8003    0.9157
]

x = 1:3
subplot(2,1,1)
plot(x,y)
```

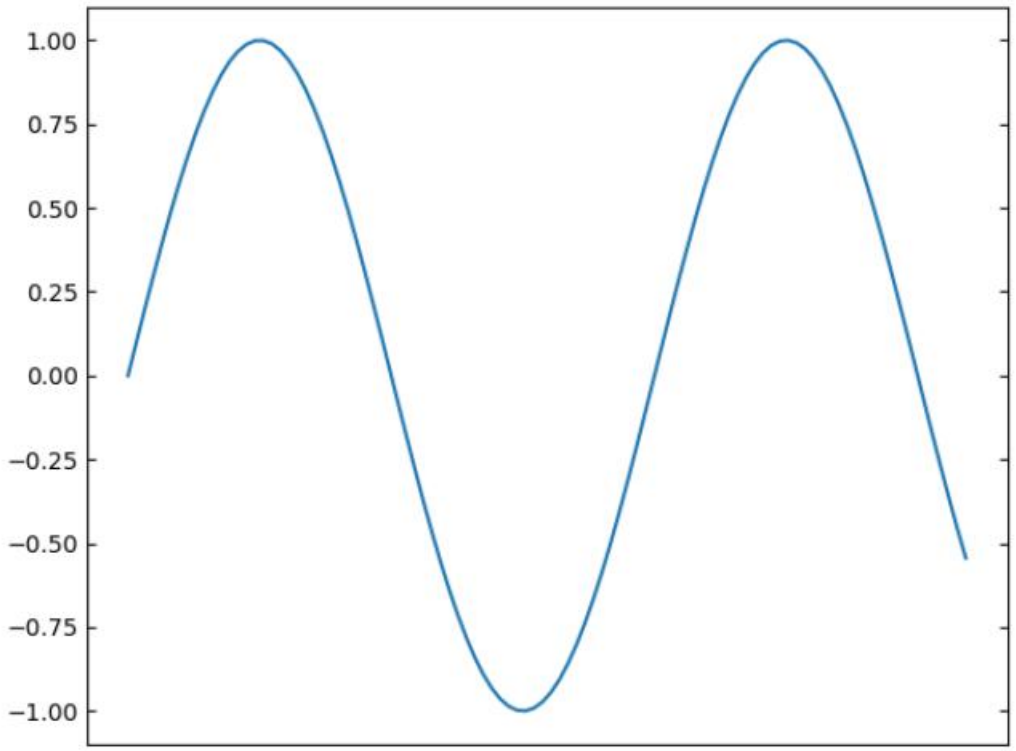
```
ax2 = subplot(2,1,2)
plot(x, y2)
xticks(ax2, [1,2,3])
```



(7). 删除 x 轴刻度线

通过将刻度值指定为空数组来删除沿 x 轴的刻度线。

```
x = LinRange(0, 10, 100);
y = sin.(x);
plot(x, y)
xticks([])
```



yticks

设置或查询 y 轴刻度值

A. 语法

```
yticks(ticks)
yt = yticks()

yticks("auto")
yticks("manual")

___ = yticks(ax,___)
```

B. 说明

yticks(ticks)设置 y 轴刻度值，这些值是 y 轴上显示刻度线的位置。指定 ticks 为递增值向量；例如[0，2，4，6]。此命令作用于当前坐标区。

yt = yticks() 以向量形式返回当前 y 轴刻度值。

yticks("auto") 设置自动模式，使坐标区确定 y 轴刻度值。如果您更改了刻度值，然后又想将它们设置回默认值，则可以使用此选项。

yticks("manual") 设置手动模式，将 y 轴刻度值冻结在当前值。如果您希

望在调整坐标区大小或者向坐标区添加新数据时保留当前刻度值，则可以使用此选项。

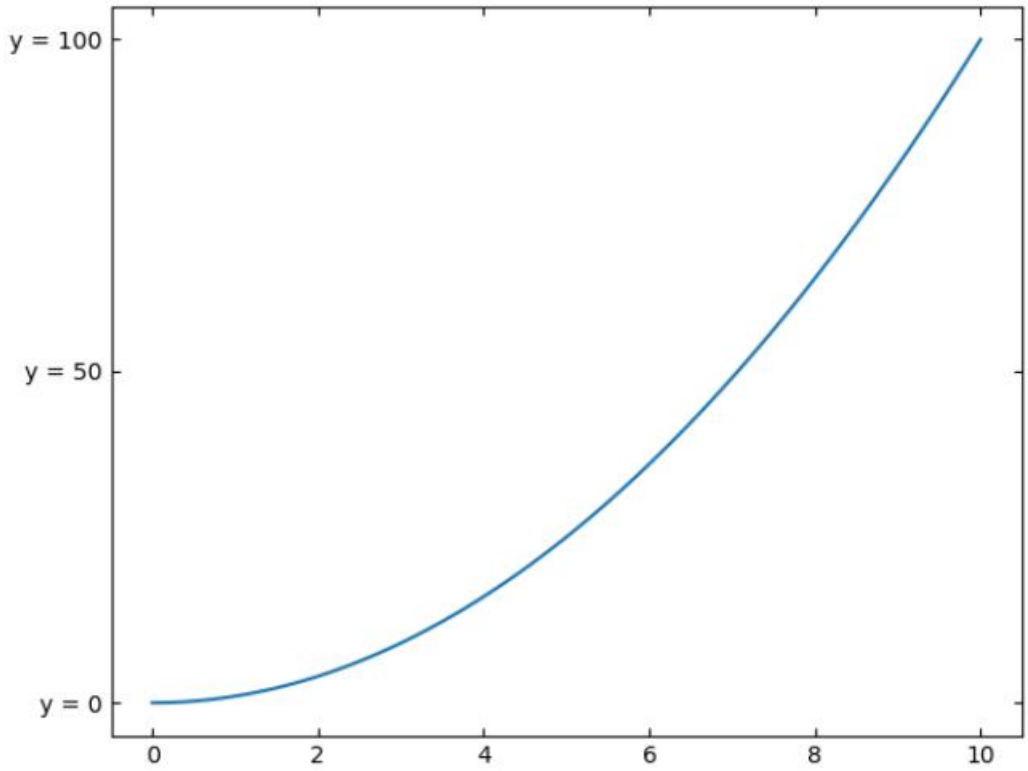
___ = yticks(ax,___)使用 ax 指定的坐标区，而不是使用当前坐标区。指定 ax 作为上述任何语法的第一个输入参数。

C. 示例

(1). 指定 y 轴刻度值和标签

创建一个线图。在 y 轴的值 0、50 和 100 处显示刻度线。然后为每个刻度线指定一个标签。

```
x = LinRange(0, 10, 100);
y = x .^ 2;
plot(x, y)
yticks([0, 50, 100])
yticklabels(["y = 0", "y = 50", "y = 100"])
```

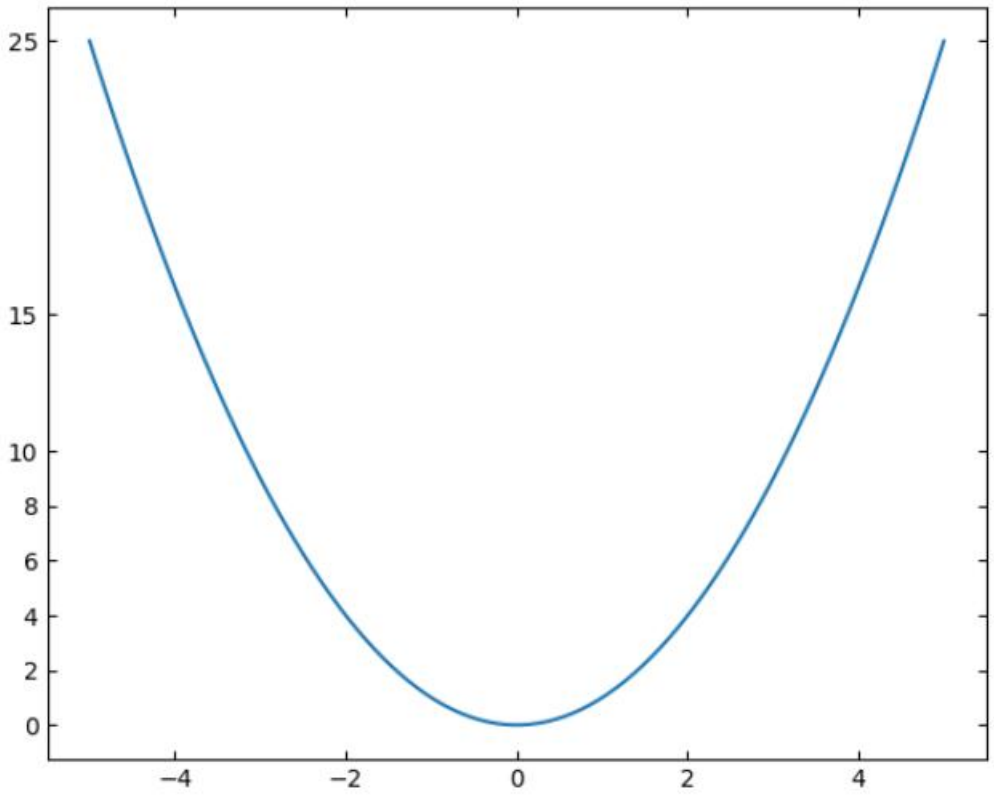


(2). 指定非均匀 y 轴刻度值

沿 y 轴以介于 0 和 25 之间的非均匀值显示刻度线。Syslab 用数值标注刻度线。

```
x = LinRange(-5, 5, 100);
y = x .^ 2;
```

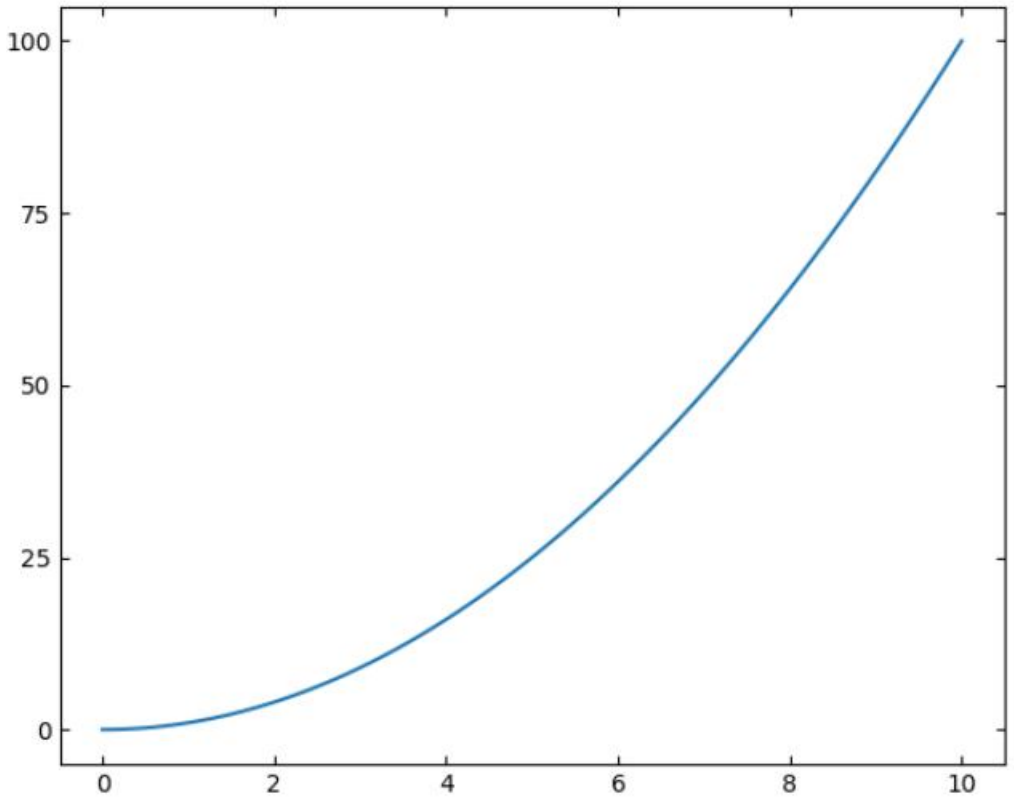
```
plot(x, y)
yticks([0, 2, 4, 6, 8, 10, 15, 25])
```



(3). 以 25 为增量递增 y 轴刻度值

沿 y 轴以 25 为增量显示刻度线，从 0 开始，在 100 结束。

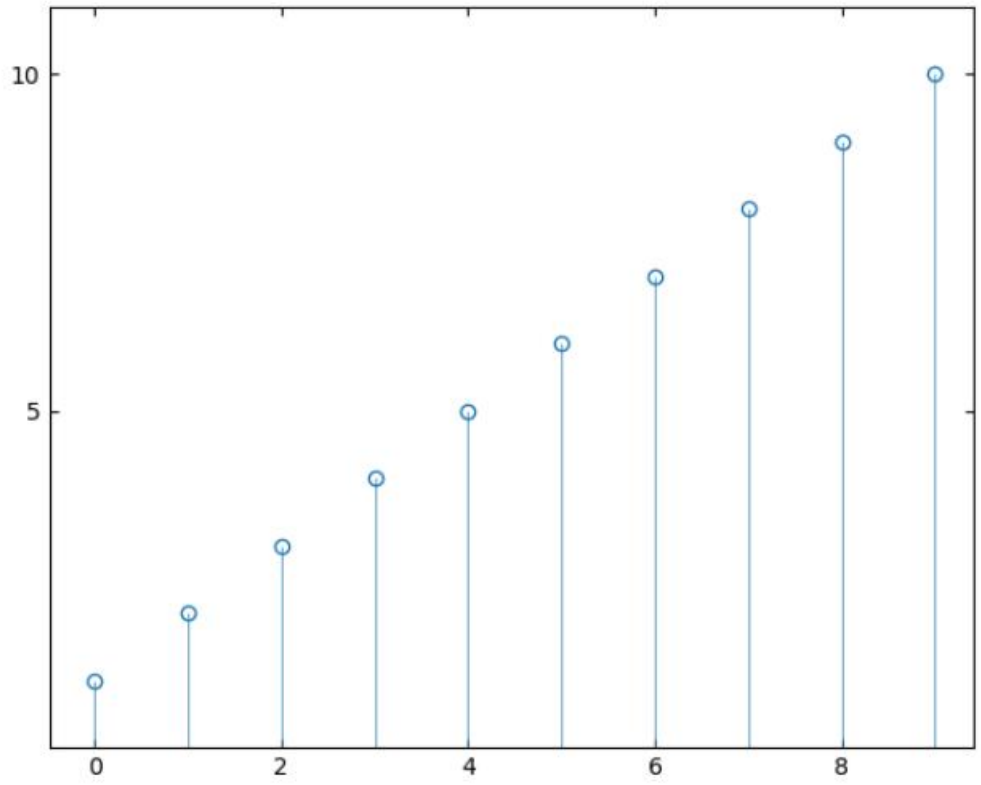
```
x = LinRange(0, 10, 100);
y = x .^ 2;
plot(x, y)
yticks(0:25:100)
```



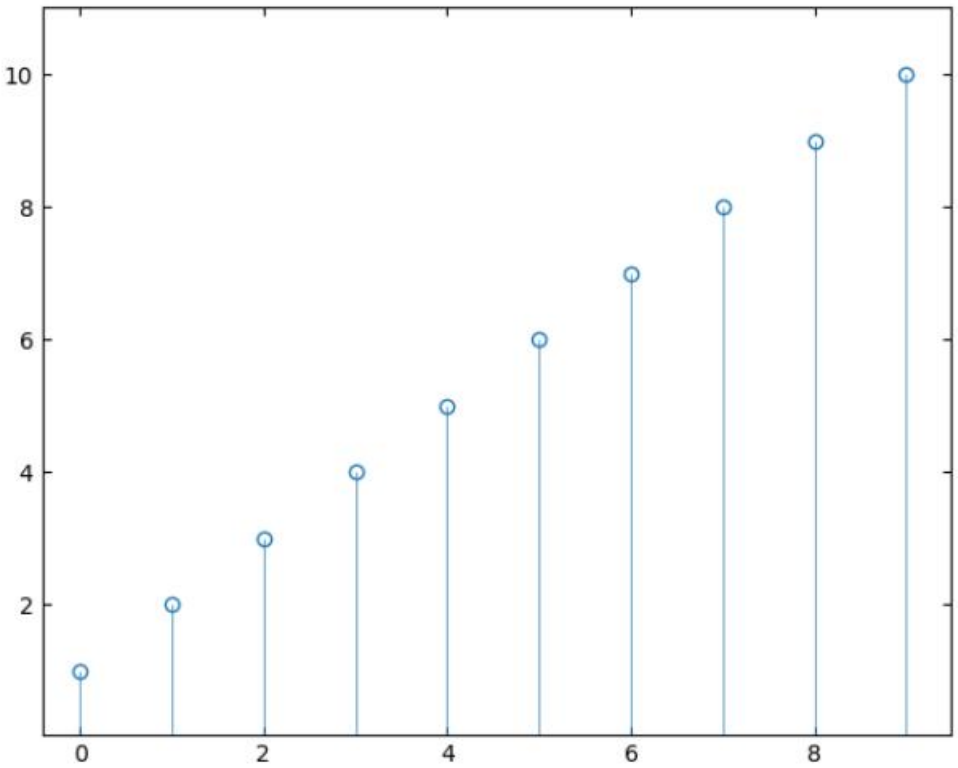
(4). 将 y 轴刻度值设置回默认值

创建一个针状图并指定 y 轴刻度值。然后，将 y 轴刻度值设置回默认值。

```
stem(1:10)
yticks([0,5,10])
```




```
yticks("auto")
```



(5). 指定特定坐标区的 y 轴刻度值

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该

对象返回为 ax1。通过将 ax1 传递给 plot 函数来创建顶部绘图。重复该过程以创建底部绘图。在每个坐标区中绘制数据。将随机数据绘制到每个坐标区中。然后通过将 ax2 作为 yticks 函数的第一个输入参数进行传递，设置下部图的 y 轴刻度值。

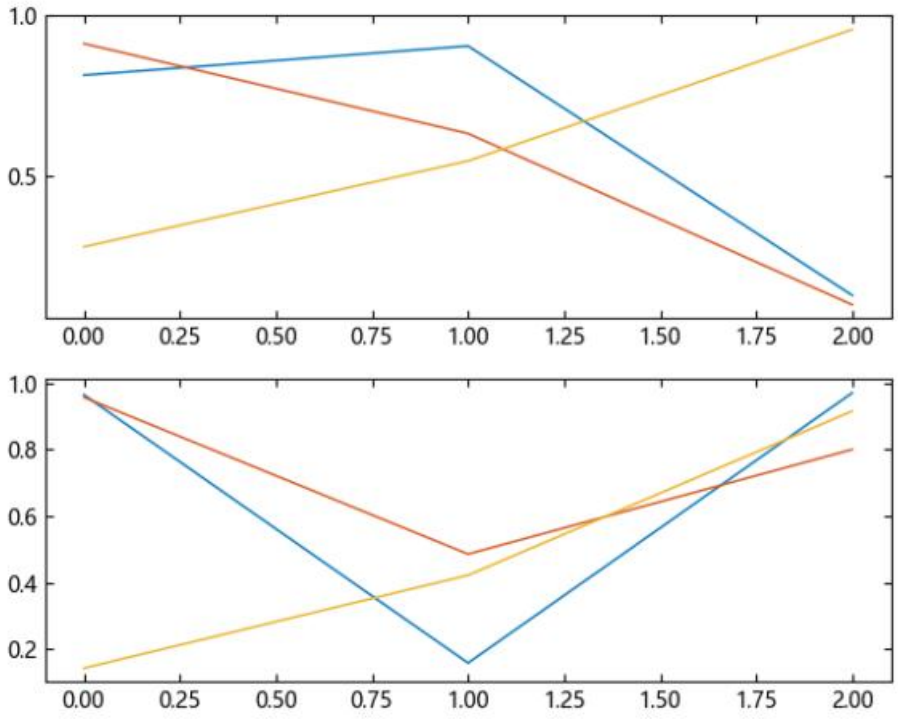
```
y = [
    0.8147    0.9134    0.2785
    0.9058    0.6324    0.5469
    0.1270    0.0975    0.9575
]

y2=[
    0.9649    0.9572    0.1419
    0.1576    0.4854    0.4218
    0.9706    0.8003    0.9157
]

ax1 = subplot(2,1,1)
plot(y)

ax2 = subplot(2,1,2)
plot(y2)

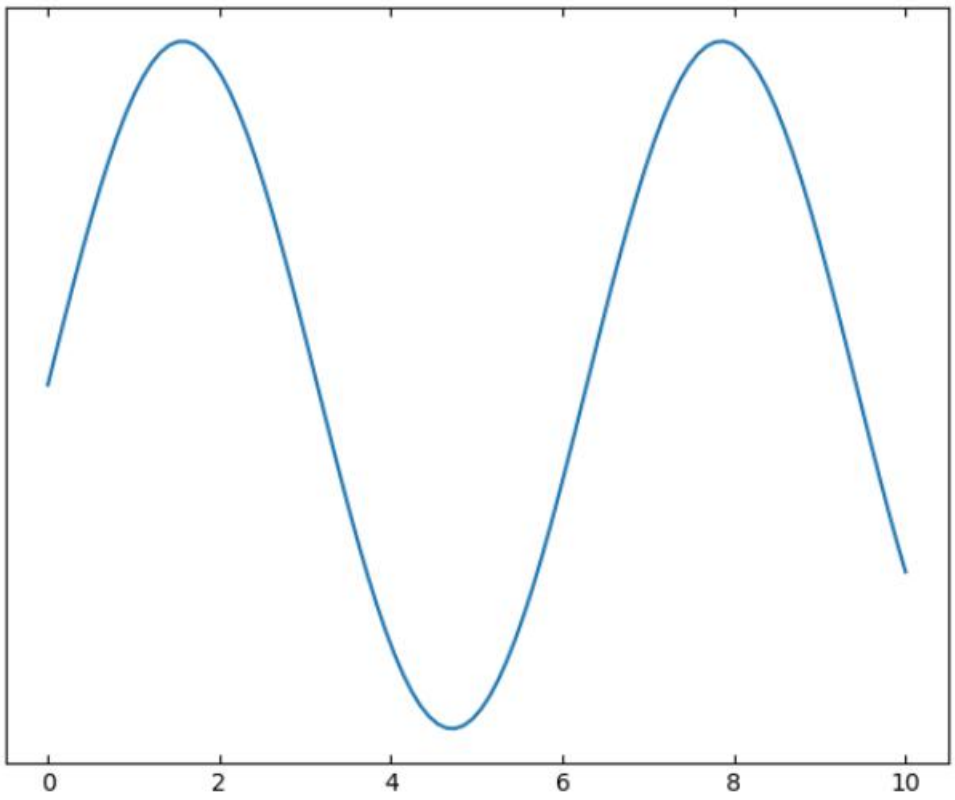
yticks(ax1, [0,0.5,1])
yticks(ax2, 0:0.2:1)
```



(6). 删除 y 轴刻度线

通过将刻度值指定为空数组来删除沿 y 轴的刻度线。

```
x = LinRange(0, 10, 100);
y = sin.(x);
plot(x, y)
yticks([])
```



zticks

设置或查询 z 轴刻度值

A. 语法

```
zticks(ticks)
zt = zticks()

zticks("auto")
zticks("manual")

___ = zticks(ax,___)
```

B. 说明

zticks(ticks)设置 z 轴刻度值，这些值是 z 轴上显示刻度线的位置。指定 ticks 为递增值向量；例如[0，2，4，6]。此命令作用于当前坐标区。

zt = zticks() 以向量形式返回当前 z 轴刻度值。

zticks("auto") 设置自动模式，使坐标区确定 z 轴刻度值。如果您更改了刻度值，然后又想将它们设置回默认值，则可以使用此选项。

zticks("manual") 设置手动模式，将 z 轴刻度值冻结在当前值。如果您希望在调整坐标区大小或者向坐标区添加新数据时保留当前刻度值，则可以使用此选项。

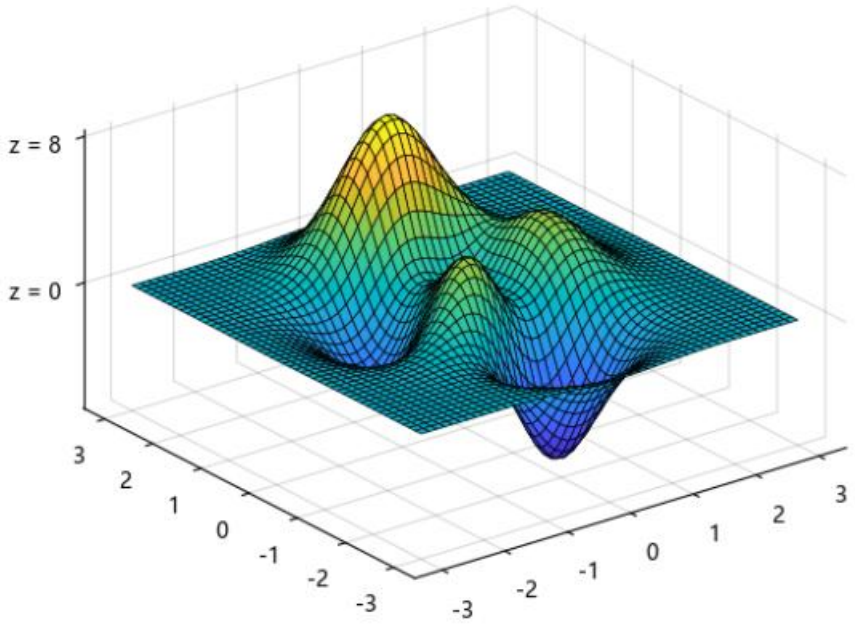
___ = zticks(ax,___)使用 ax 指定的坐标区，而不是使用当前坐标区。指定 ax 作为上述任何语法的第一个输入参数。

C. 示例

(1). 指定 z 轴刻度值和标签

创建一个 surf 图。在值 z = -8、z = 0 和 z = 8 处显示 z 轴的刻度线。然后为每个刻度线指定一个标签。

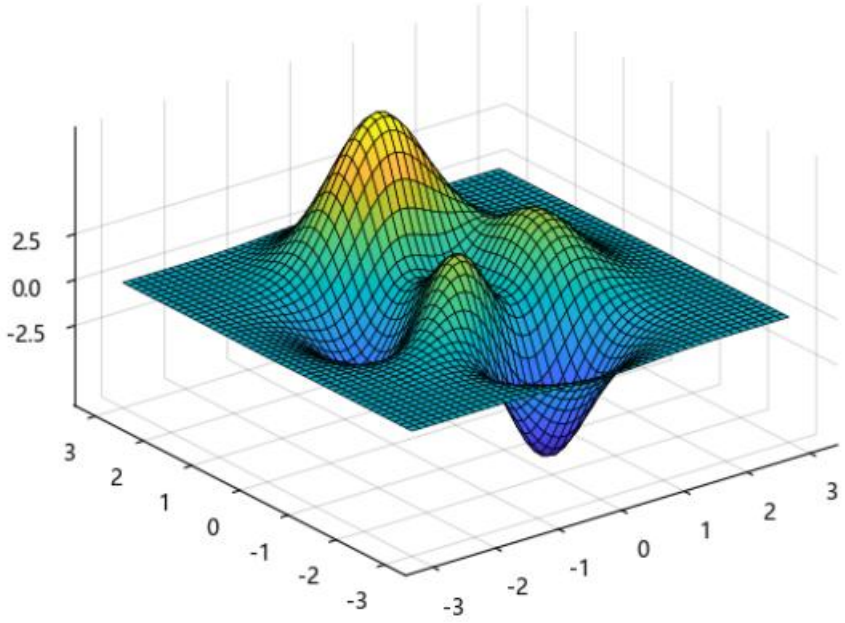
```
x,y,z = peaks();
surf(x,y,z)
zticks([-8 0 8])
zticklabels(["z = -8","z = 0","z = 8"])
```



(2). 指定非均匀 z 轴刻度值

沿 z 轴以介于 -10 和 10 之间的非均匀值显示刻度线。Syslab 用数值标注刻度线。

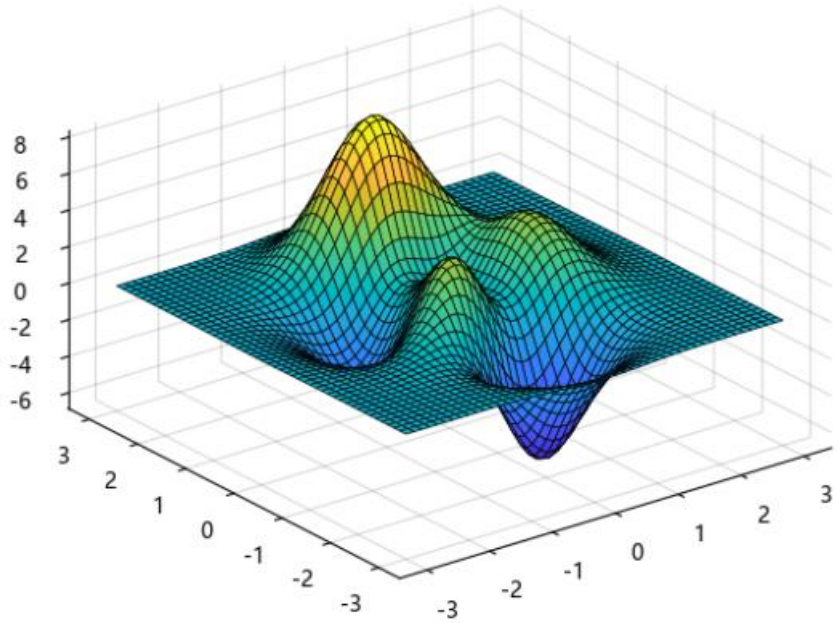
```
x,y,z = peaks();
surf(x,y,z)
zticks([-10 -2.5 0 2.5 10])
```



(3). 以 2 为增量递增 z 轴刻度值

沿 z 轴以 2 为增量显示刻度线，从 -10 开始，在 10 结束。

```
x,y,z = peaks();
surf(x,y,z)
zticks(-10:2:10)
```

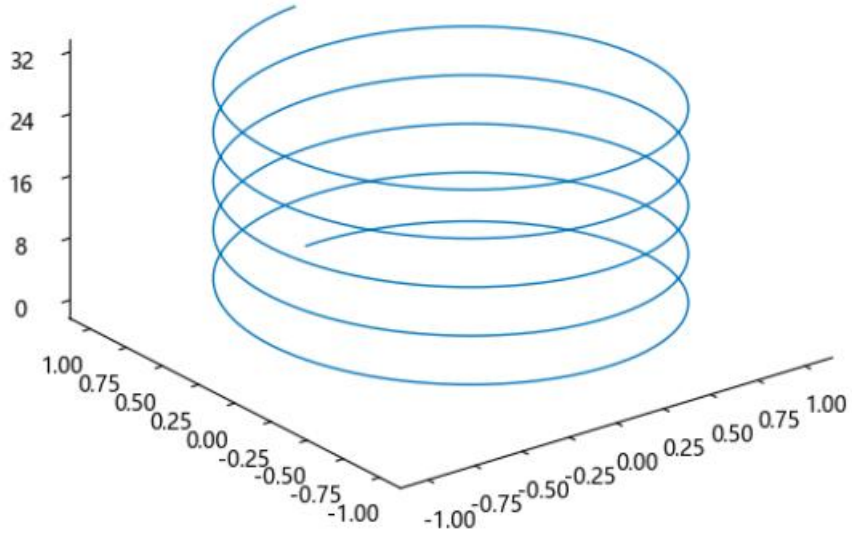


(4). 将 z 轴刻度值设置回默认值

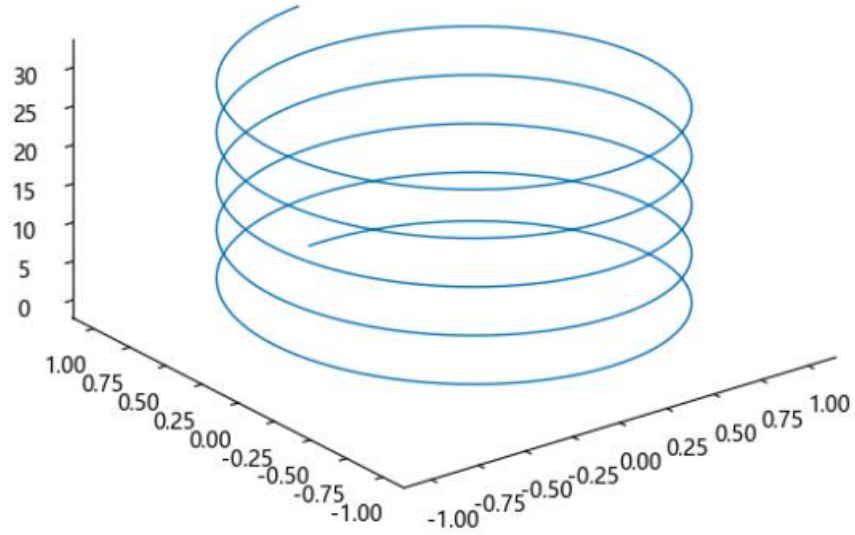
创建一个三维线图并指定 z 轴刻度值。然后，将 z 轴刻度值设置回默认值。

```
t = 0:pi/50:10*pi;
st = sin.(t);
```

```
ct = cos.(t);
plot3(st,ct,t)
zticks(0:8:40)
```




```
zticks("auto")
```



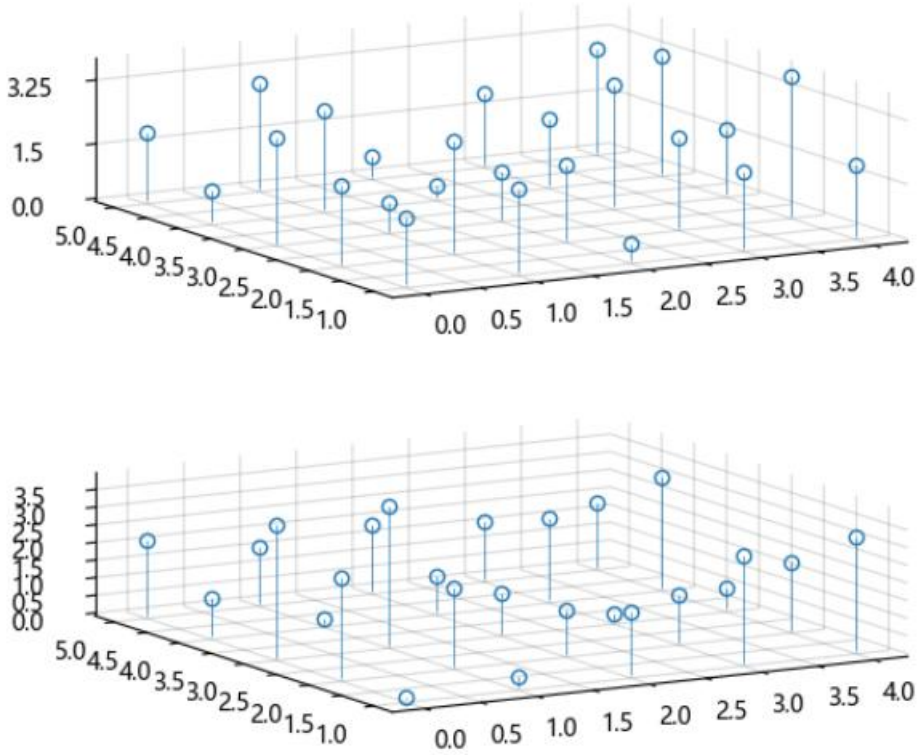
(5). 指定特定坐标区的 z 轴刻度值

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。通过将 ax1 传递给 stem3 函数来创建顶部绘图。重复该过程以创建底部绘图。在每个坐标区中绘制数据。将随机数据绘制到每个坐标区中。然后通过将 ax1 作为 zticks 函数的第一个输入参数进行传递，设置下部图的 z 轴刻度值。

```
ax1 = subplot(2,1,1,projection="3d")
```

```
stem3(ax1, 4*rand(5,5))
zticks(ax1,[0 1.5 3.25])

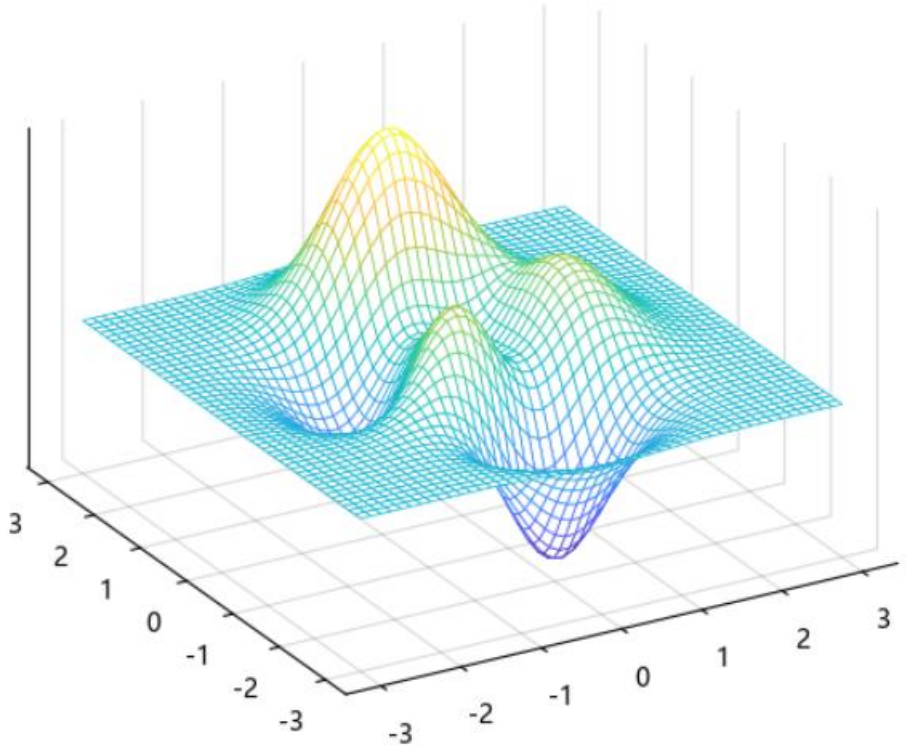
ax2 = subplot(2,1,2,projection="3d")
stem3(ax2, 4*rand(5,5))
```



(6). 删除 z 轴刻度线

通过将刻度值指定为空数组来删除沿 z 轴的刻度线。

```
x,y,z = peaks();
mesh(x,y,z)
zticks([])
```



xticklabels

设置或查询 x 轴刻度标签

A. 语法

```
xticklabels(labels)
xl = xticklabels()

xticklabels("auto")
xticklabels("manual")

___ = xticklabels(ax,___)
```

B. 说明

xticklabels(labels)设置当前坐标区的 x 轴刻度标签。可将 labels 指定为字符串数组，例如["January","February","March"]。如果指定标签，则 x 轴刻度值和刻度标签不会再基于坐标区的更改而自动更新。

xl = xticklabels() 返回当前坐标区的 x 轴刻度标签。

xticklabels("auto") 设置自动模式，使坐标区确定 x 轴刻度标签。如果您设

置了标签，然后又想将它们设置回默认值，则可以使用此选项。

xticklabels("manual") 设置手动模式，将 x 轴刻度标签冻结在当前值。

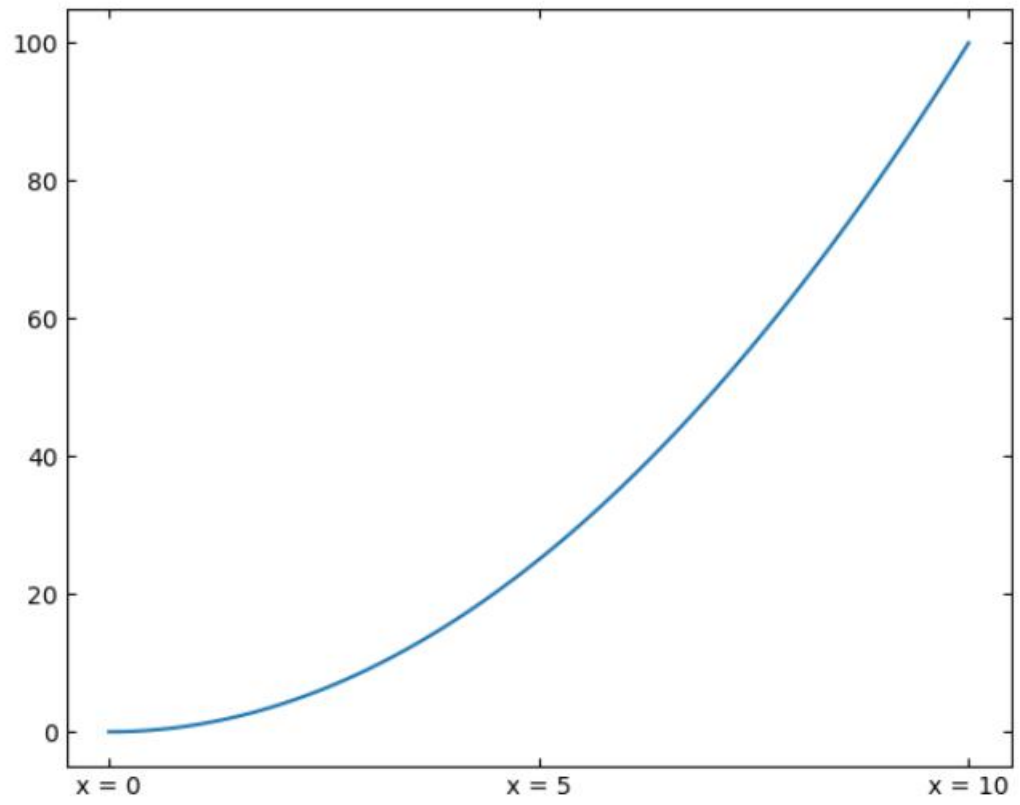
___ = xticklabels(ax,___)使用 ax 指定的坐标区，而不是使用当前坐标区。指定 ax 作为上述任何语法的第一个输入参数。

C. 示例

(1). 指定 x 轴刻度值和标签

创建一个线图。在值 0、5 和 10 处显示 x 轴的刻度线。然后为每个刻度线指定一个标签。

```
x = LinRange(0, 10, 100);
y = x .^ 2;
plot(x, y)
xticks([0 5 10])
xticklabels(["x = 0", "x = 5", "x = 10"])
```

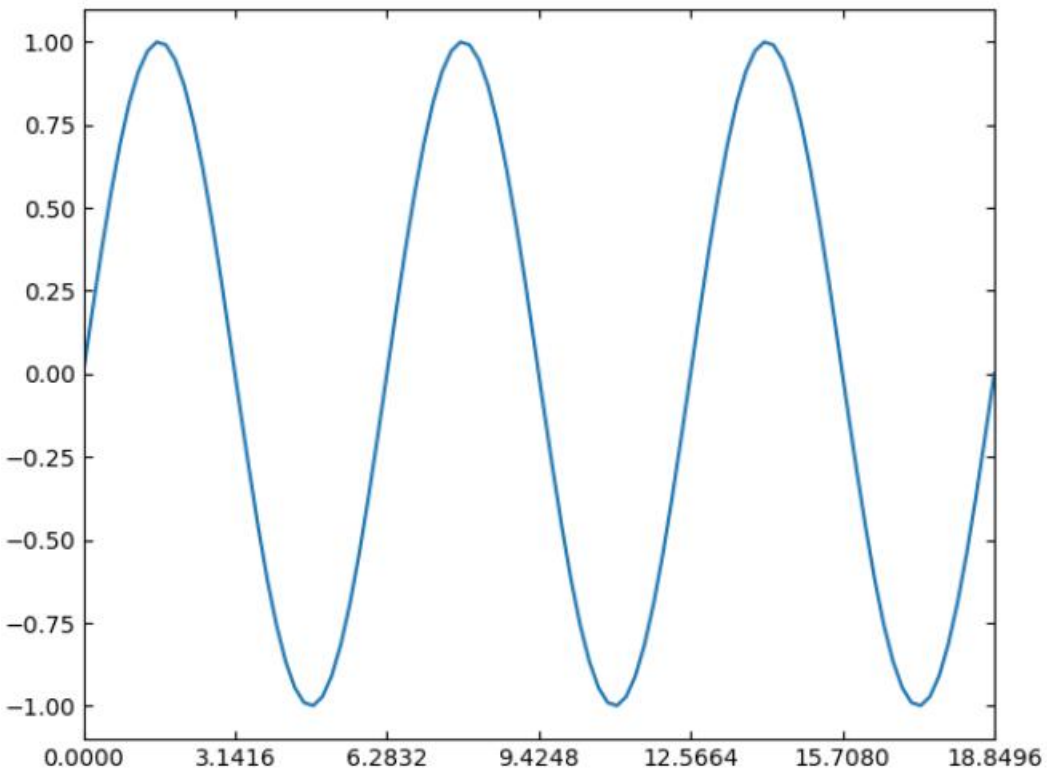


(2). 以 Pi 为单位显示 x 轴刻度标签

创建一个线图。指定 x 坐标轴范围并沿 x 轴显示以 π 为增量的刻度线。

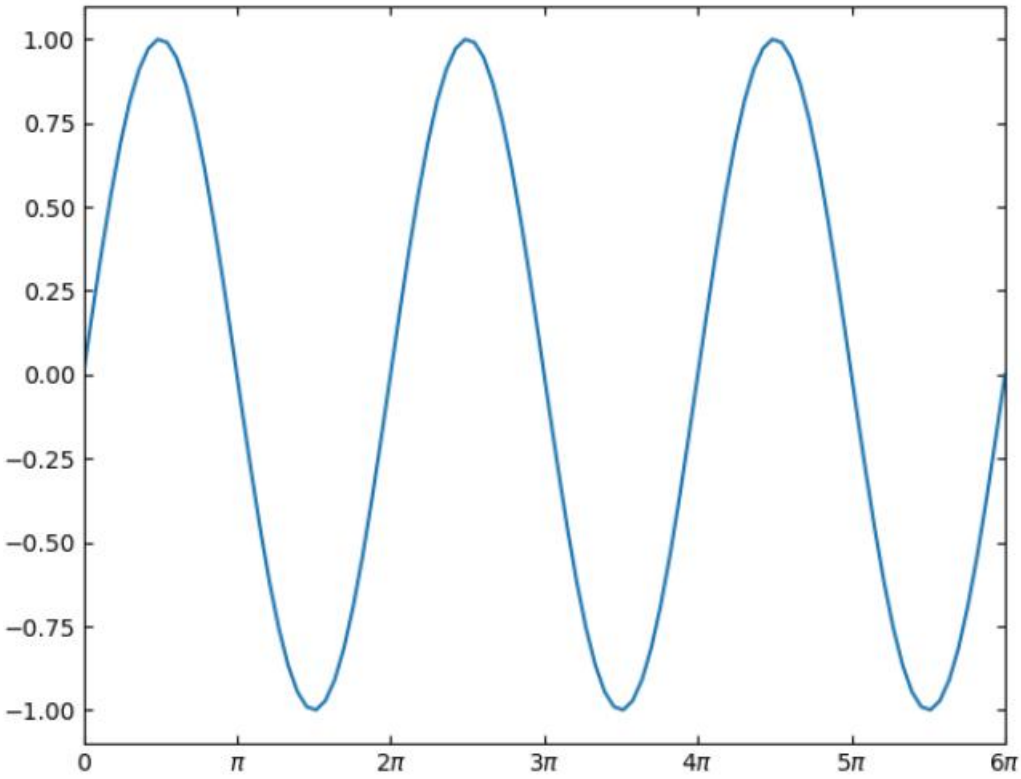
```
x = LinRange(0, 6 * pi, 100);
```

```
y = sin.(x);
plot(x, y)
xlim([0, 6 * pi])
xticks(0:pi:(6 * pi))
xtickformat("%.4f")
```



Syslab 用数值标注刻度线。通过为每个标签指定文本来更改标签以显示 π 符号。

```
xticklabels(["0",raw"$\pi$",raw"2$\pi$",raw"3$\pi$",raw"4$\pi$",raw"5
$\pi$",raw"6$\pi$"])
```



(3). 指定特定坐标区的 x 轴刻度标签

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。通过将 ax1 传递给 plot 函数来创建顶部绘图。重复该过程以创建底部绘图。在每个坐标区中绘制数据。将随机数据绘制到每个坐标区中。然后通过将 ax2 指定为第一个输入参数，设置第二个图的 x 轴刻度值和标签。

```
y = [
    0.8147    0.9134    0.2785
    0.9058    0.6324    0.5469
    0.1270    0.0975    0.9575
]

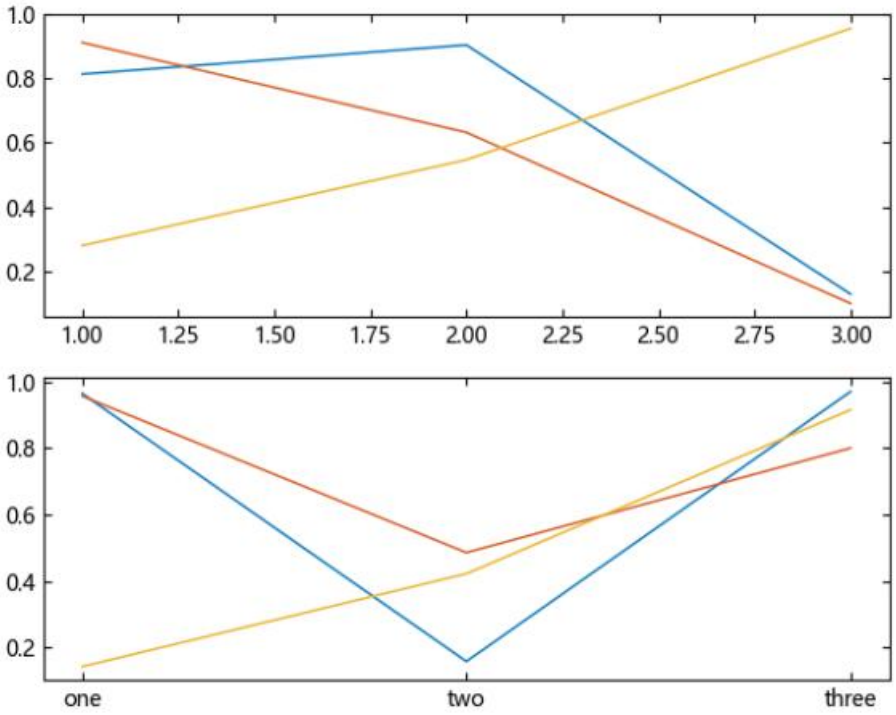
y2=[
    0.9649    0.9572    0.1419
    0.1576    0.4854    0.4218
    0.9706    0.8003    0.9157
]

x=1:3

ax1=subplot(2,1,1)
```

```
plot(ax1,x,y)

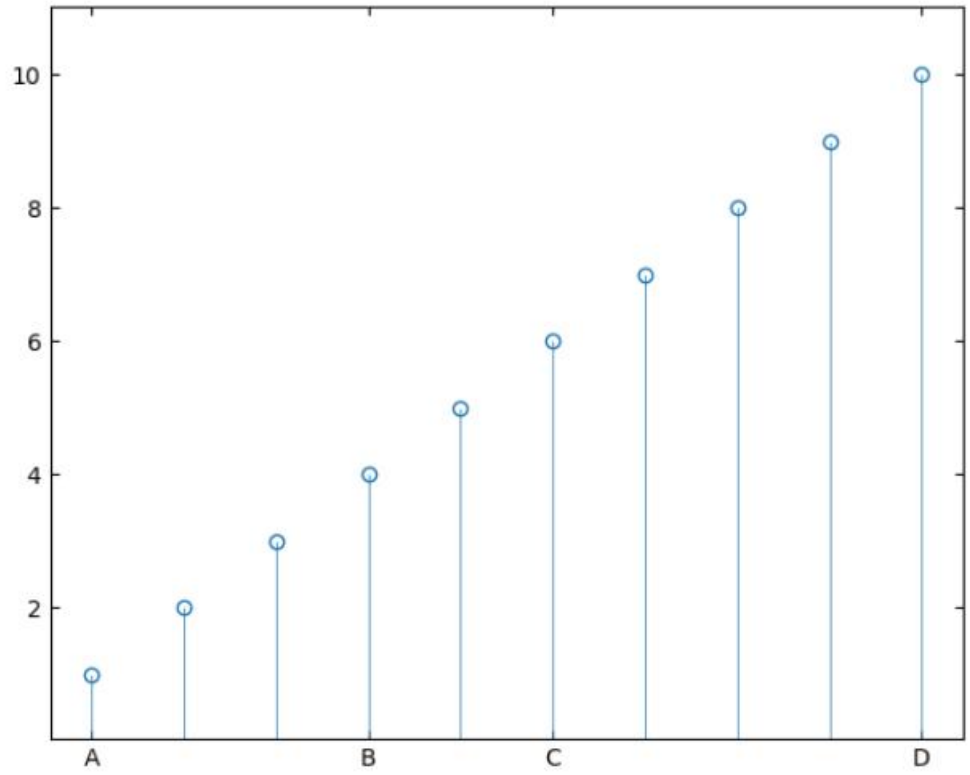
ax2 = subplot(2,1,2)
plot(ax2,x,y2)
xticks(ax2,[1,2,3])
xticklabels(ax2,["one","two","three"])
```



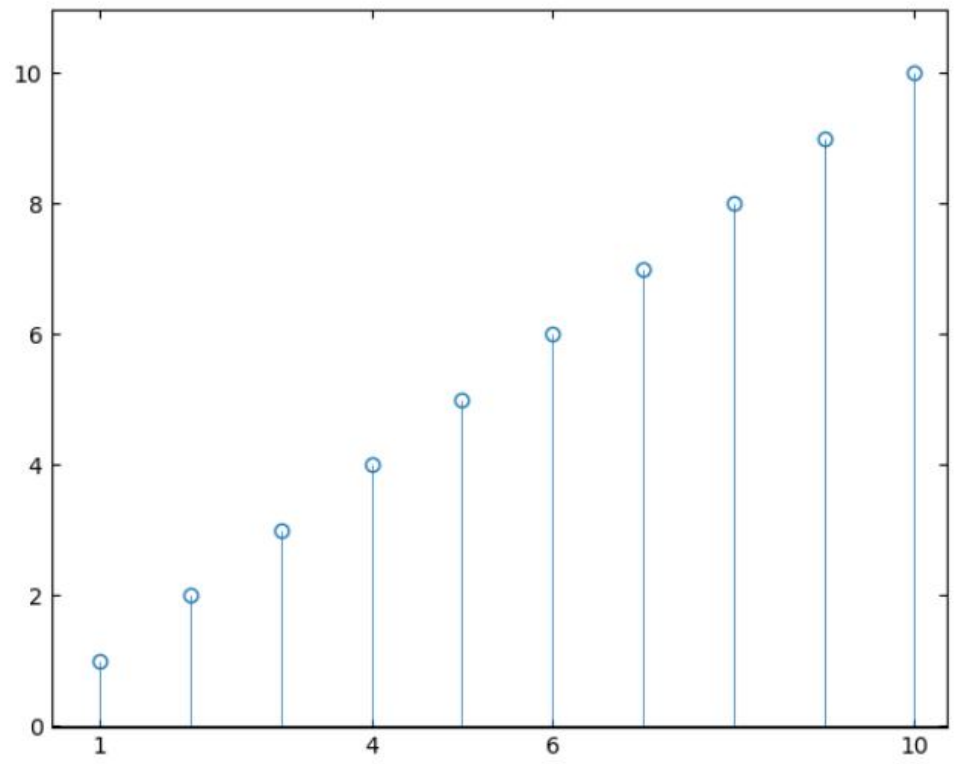
(4). 将 x 轴刻度标签设置回默认标签

创建一个针状图并指定 x 轴刻度值和对应的标签。然后将 x 轴刻度值和标签设置回默认值。

```
x=1:10
stem(x,1:10)
xticks([1,4,6,10])
xticklabels(["A","B","C","D"])
```




```
xticklabels("auto")
```

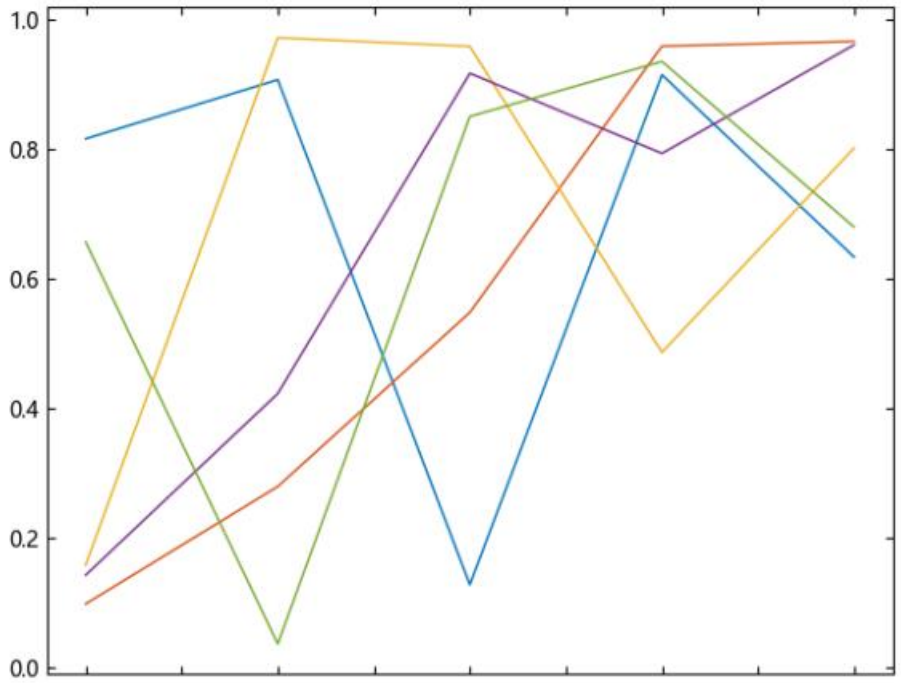


(5). 删除 x 轴刻度标签

通过将刻度值指定为空数组来删除沿 x 轴的刻度线。

```
y=[
    0.8147    0.0975    0.1576    0.1419    0.6557
    0.9058    0.2785    0.9706    0.4218    0.0357
    0.1270    0.5469    0.9572    0.9157    0.8491
    0.9134    0.9575    0.4854    0.7922    0.9340
    0.6324    0.9649    0.8003    0.9595    0.6787
]

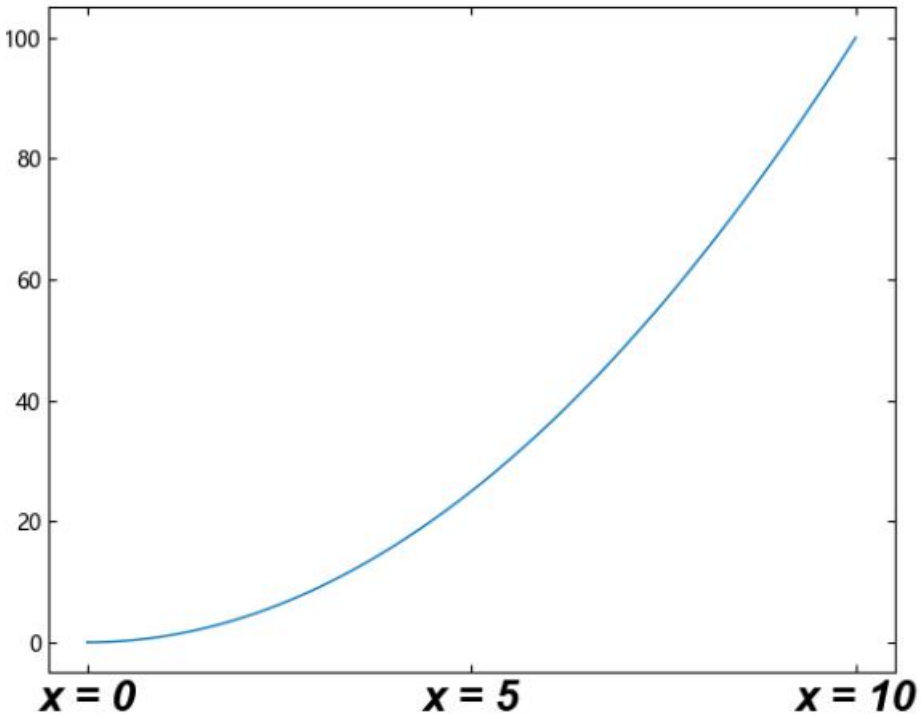
plot(y)
xticklabels([])
```



(6). 自定义 X 坐标轴的标签

创建一个线图。在值 0、5 和 10 处显示 x 轴的刻度线。然后为每个刻度线指定一个标签，设置标签字体的名称、大小、样式及粗细。

```
x = LinRange(0, 10, 100);
y = x .^ 2;
plot(x, y)
xticks([0 5 10])
xticklabels(
    ["x = 0", "x = 5", "x = 10"];
    fontfamily="Arial",
    fontsize=20,
    fontstyle="italic",
    fontweight="bold",
)
```



yticklabels

设置或查询 y 轴刻度标签

A. 语法

```
yticklabels(labels)
yl = yticklabels()

yticklabels("auto")
yticklabels("manual")

___ = yticklabels(ax,___)
```

B. 说明

yticklabels(labels)设置当前坐标区的 y 轴刻度标签。可将 labels 指定为字符串数组，例如["January","February","March"]。如果指定标签，则 y 轴刻度值和刻度标签不会再基于坐标区的更改而自动更新。

yl = yticklabels() 返回当前坐标区的 y 轴刻度标签。

yticklabels("auto") 设置自动模式，使坐标区确定 y 轴刻度标签。如果您设

置了标签，然后又想将它们设置回默认值，则可以使用此选项。

yticklabels("manual") 设置手动模式，将 y 轴刻度标签冻结在当前值。

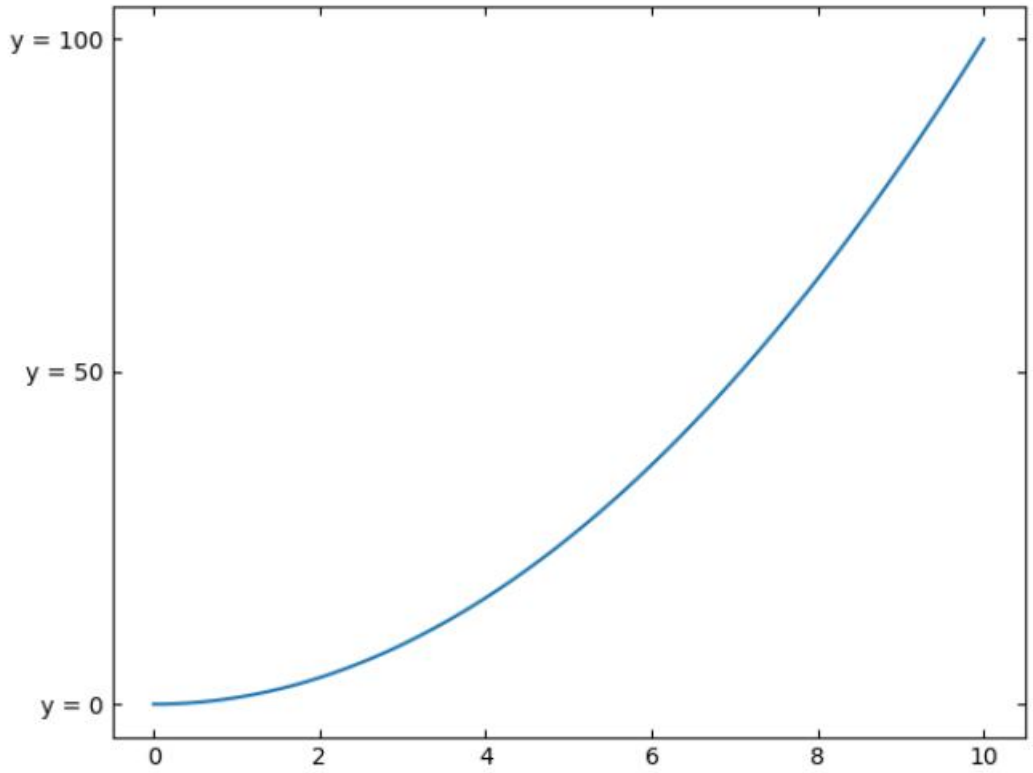
___ = yticklabels(ax,___)使用 ax 指定的坐标区，而不是使用当前坐标区。指定 ax 作为上述任何语法的第一个输入参数。

C. 示例

(1). 指定 y 轴刻度值和标签

创建一个线图。在 y 轴的值 0、50 和 100 处显示刻度线。然后为每个刻度线指定一个标签。

```
x = LinRange(0, 10, 100);
y = x .^ 2;
plot(x, y)
yticks([0, 50, 100])
yticklabels(["y = 0", "y = 50", "y = 100"])
```



(2). 指定特定坐标区的 y 轴刻度标签

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。通过将 ax1 传递给 plot 函数来创建顶部绘图。重复该过程以创建底部绘图。创建两个随机数据的绘图。通过将 ax2 作为 yticks 和 yticklabels 函数的第一个输入参数进行传递，设置第二个图的 y 轴刻度值

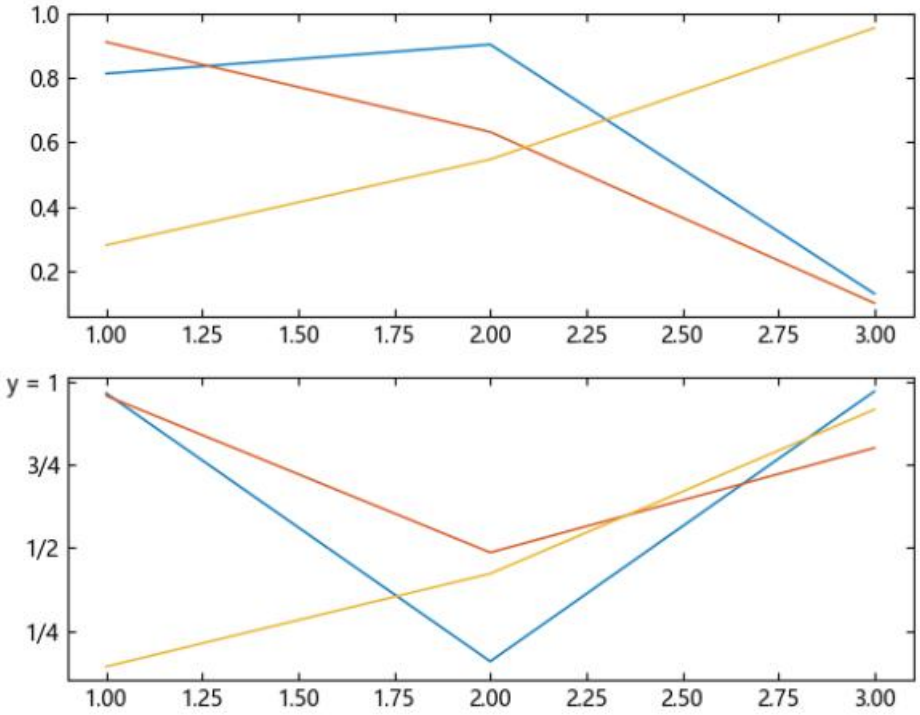
和标签。

```
y = [
    0.8147    0.9134    0.2785
    0.9058    0.6324    0.5469
    0.1270    0.0975    0.9575
]

y2=[
    0.9649    0.9572    0.1419
    0.1576    0.4854    0.4218
    0.9706    0.8003    0.9157
]

x = 1:3
ax1 = subplot(2,1,1)
plot(x,y)

ax2 = subplot(2,1,2)
plot(x,y2)
yticks(ax2,[0, .25, .5, .75, 1])
yticklabels(ax2, ["y = 0","1/4","1/2","3/4","y = 1"])
```

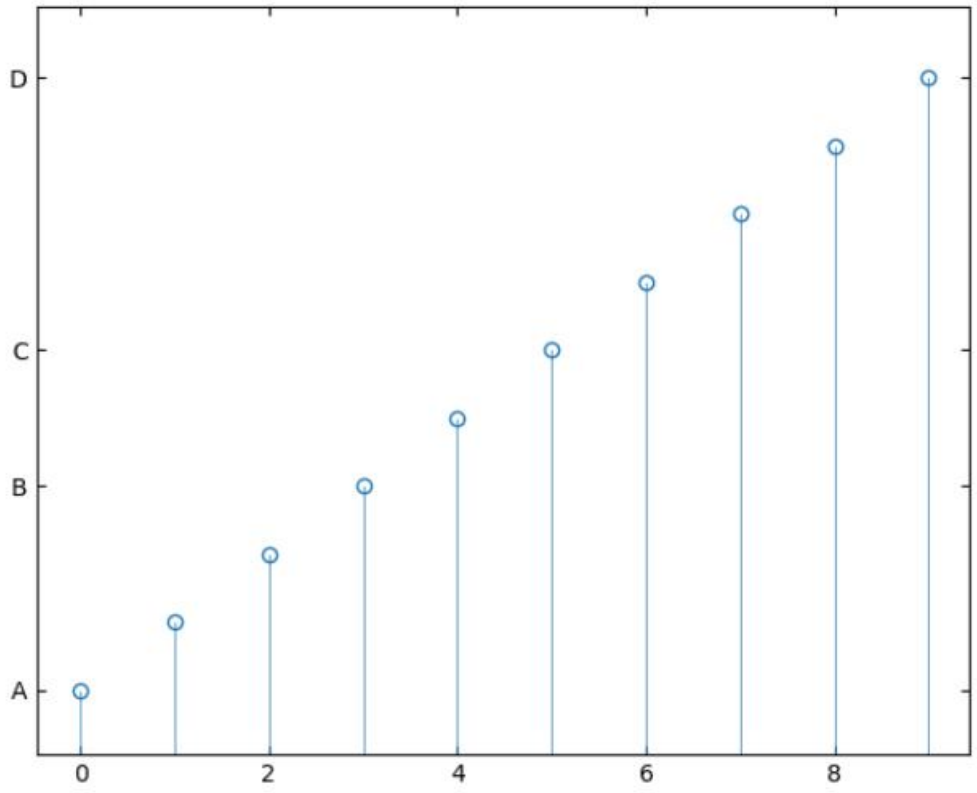


(3). 将 y 轴刻度标签设置回默认标签

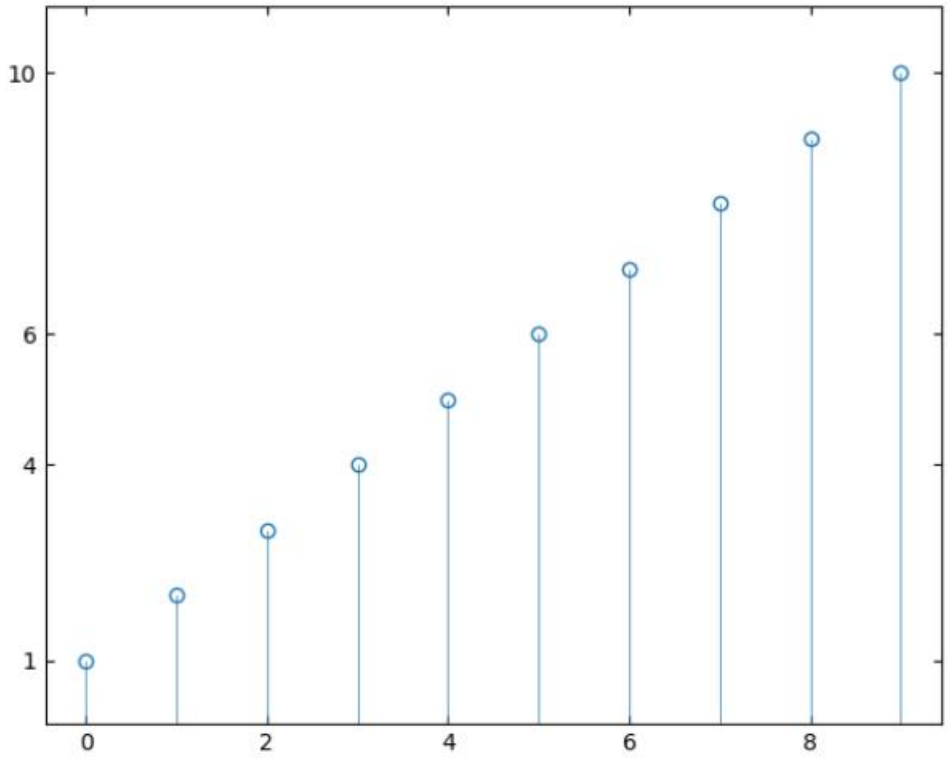
创建一个线图。在 y 轴的值 1、4、6、10 处显示刻度线。然后为每个刻

度线指定一个标签。

```
stem(1:10)
yticks([1 4 6 10])
yticklabels(["A","B","C","D"])
```




```
yticklabels("auto")
```

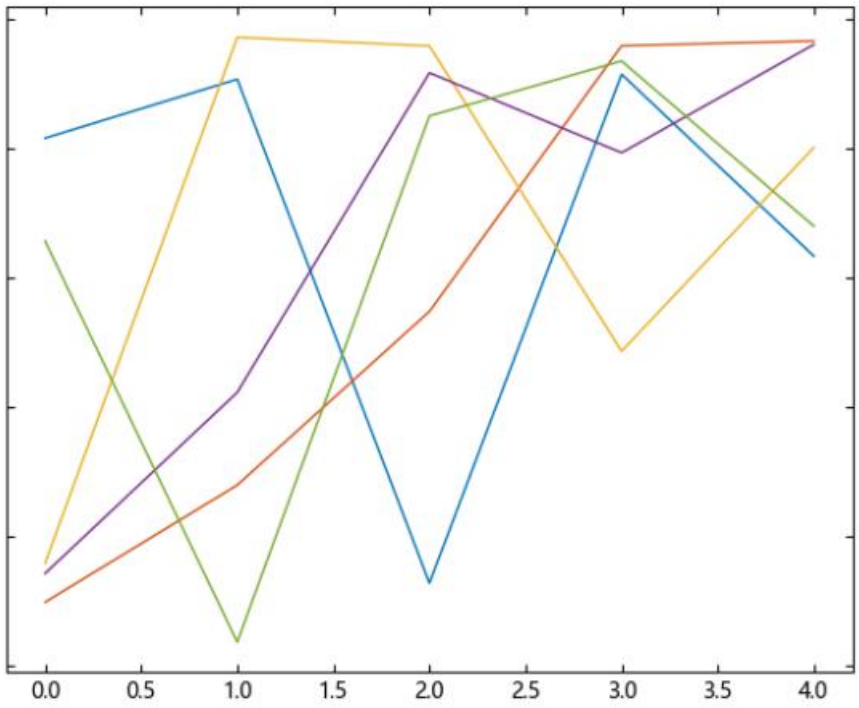


(4). 删除 y 轴刻度标签

通过将刻度标签指定为空数组来删除 y 轴的刻度标签。

```
y=[
  0.8147    0.0975    0.1576    0.1419    0.6557
  0.9058    0.2785    0.9706    0.4218    0.0357
  0.1270    0.5469    0.9572    0.9157    0.8491
  0.9134    0.9575    0.4854    0.7922    0.9340
  0.6324    0.9649    0.8003    0.9595    0.6787
]

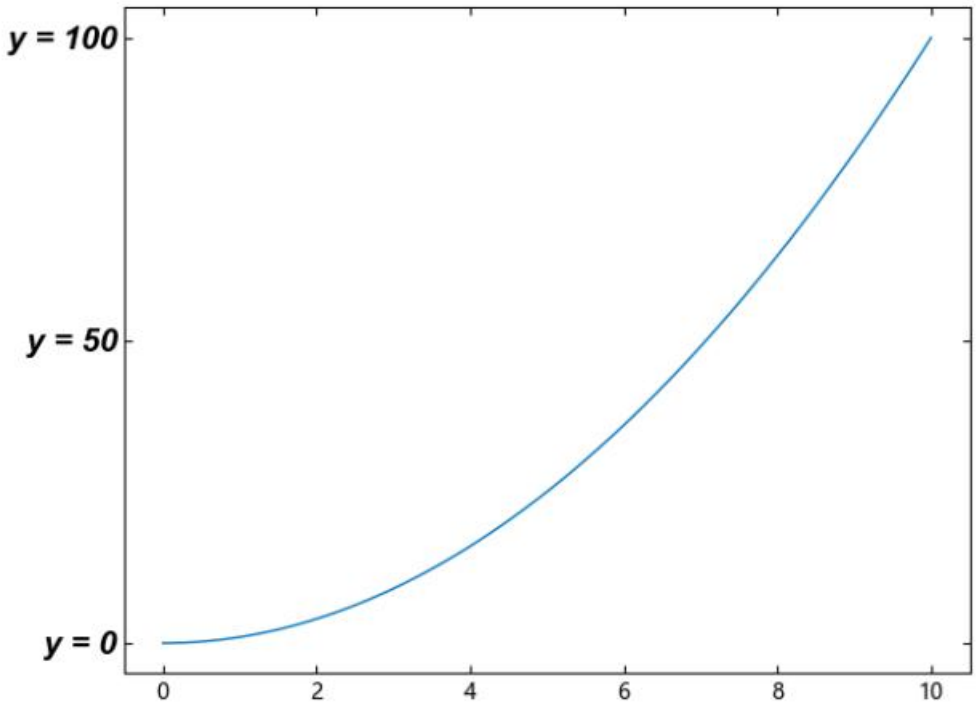
plot(y)
yticklabels([])
```



(5). 自定义 Y 坐标轴的标签

创建一个线图。在 y 轴的值 0、50 和 100 处显示刻度线。然后为每个刻度线指定一个标签，，设置标签字体的名称、大小、样式及粗细。

```
x = LinRange(0, 10, 100);
y = x .^ 2;
plot(x, y)
yticks([0, 50, 100])
yticklabels(
    ["y = 0", "y = 50", "y = 100"];
    fontfamily="Arial",
    fontsize=15,
    fontstyle="italic",
    fontweight="bold",
)
```



zticklabels

设置或查询 z 轴刻度标签

A. 语法

```
zticklabels(labels)
zl = zticklabels()

zticklabels("auto")
zticklabels("manual")

___ = zticklabels(ax,___)
```

B. 说明

zticklabels(labels)设置当前坐标区的 z 轴刻度标签。可将 labels 指定为字符串数组，例如["January","February","March"]。如果指定标签，则 z 轴刻度值和刻度标签不会再基于坐标区的更改而自动更新。

zl = zticklabels() 返回当前坐标区的 z 轴刻度标签。

zticklabels("auto") 设置自动模式，使坐标区确定 z 轴刻度标签。如果您设置了标签，然后又想将它们设置回默认值，则可以使用此选项。

zticklabels("manual") 设置手动模式，将 z 轴刻度标签冻结在当前值。

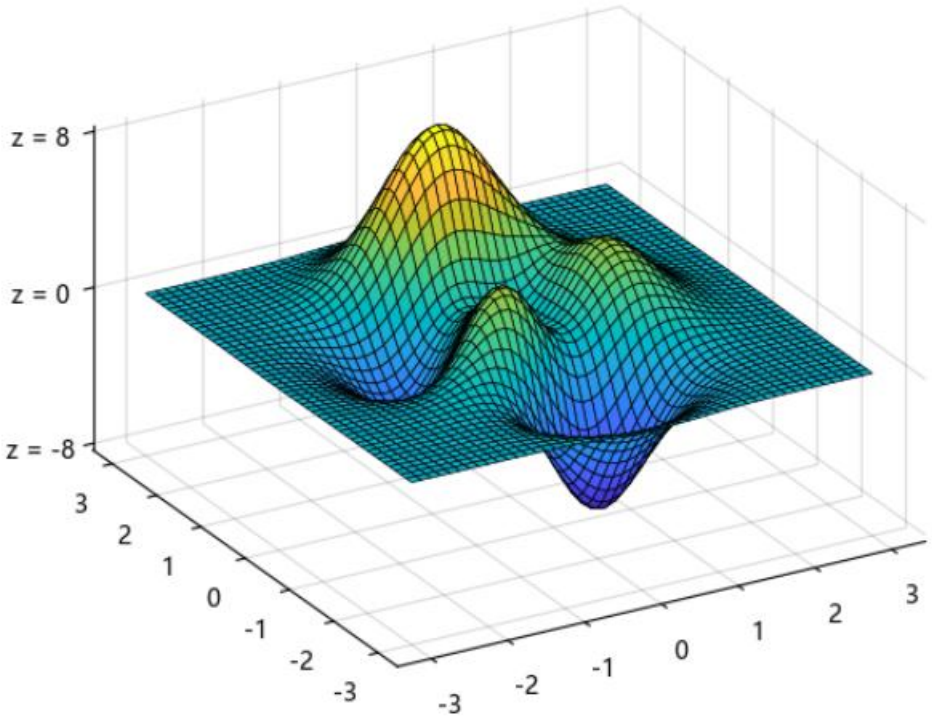
___ = zticklabels(ax,___)使用 ax 指定的坐标区，而不是使用当前坐标区。指定 ax 作为上述任何语法的第一个输入参数。

C. 示例

(1). 指定 z 轴刻度值和标签

创建一个 surf 图。在值 -8、0 和 8 处显示 z 轴的刻度线。然后为每个刻度线指定一个标签。

```
x,y,z = peaks();
surf(x,y,z)
zticks([-8 0 8])
zticklabels(["z = -8","z = 0","z = 8"])
```



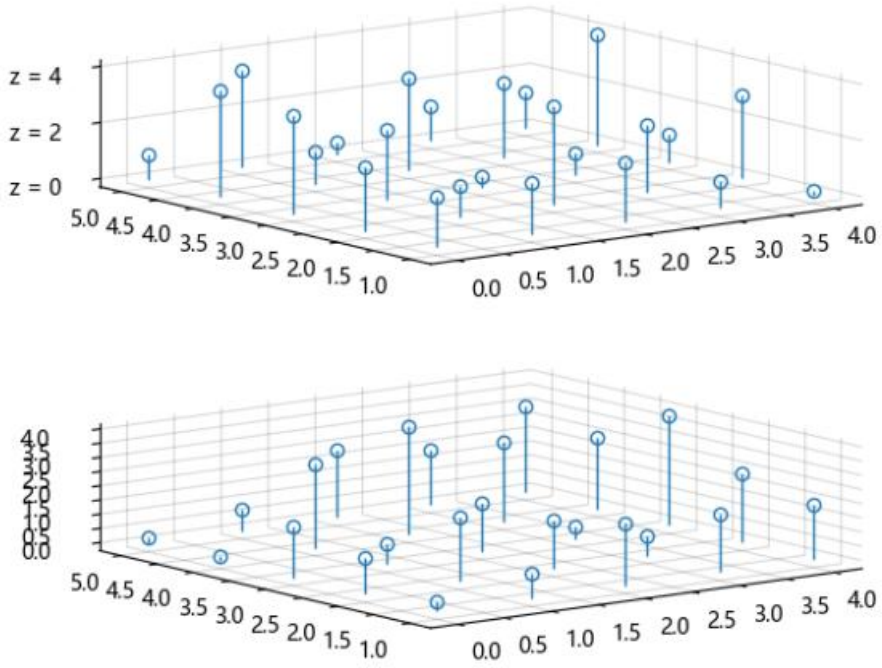
(2). 指定特定坐标区的 z 轴刻度标签

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。通过将 ax1 传递给 stem3 函数来创建顶部绘图。重复该过程以创建底部绘图。在每个坐标区中绘制数据。将随机数据绘制到每个坐标区中。然后通过将 ax2 指定为第一个输入参数，设置第二个图的 z 轴刻度值和标签。

```
ax1 = subplot(2,1,1,projection="3d")
stem3(ax1, 4*rand(5,5))
```

```
zticks(ax1,[0 2 4])
zticklabels(ax1,["z = 0","z = 2","z = 4"])

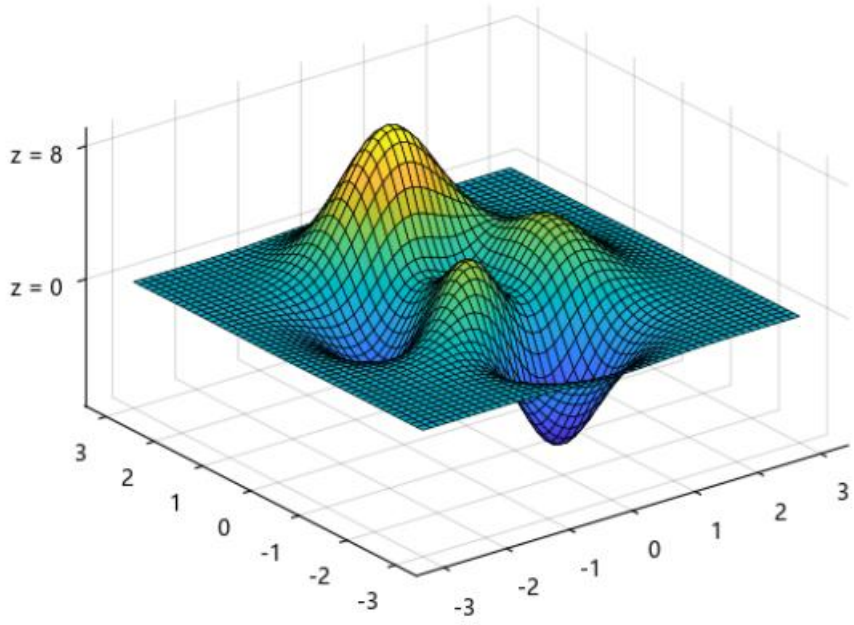
ax2 = subplot(2,1,2,projection="3d")
stem3(ax2, 4*rand(5,5))
```



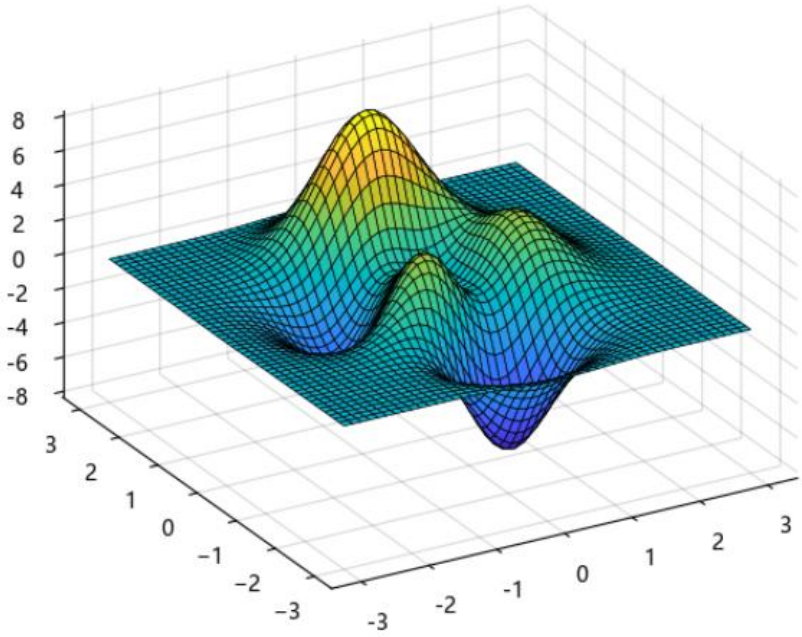
(3). 将 z 轴刻度标签设置回默认标签

创建一个 surf 图并指定 z 轴刻度值和对应的标签。然后将 z 轴刻度值和标签设置回默认值。

```
x,y,z = peaks();
surf(x,y,z)
zticks([-8 0 8])
zticklabels(["z = -8","z = 0","z = 8"])
```



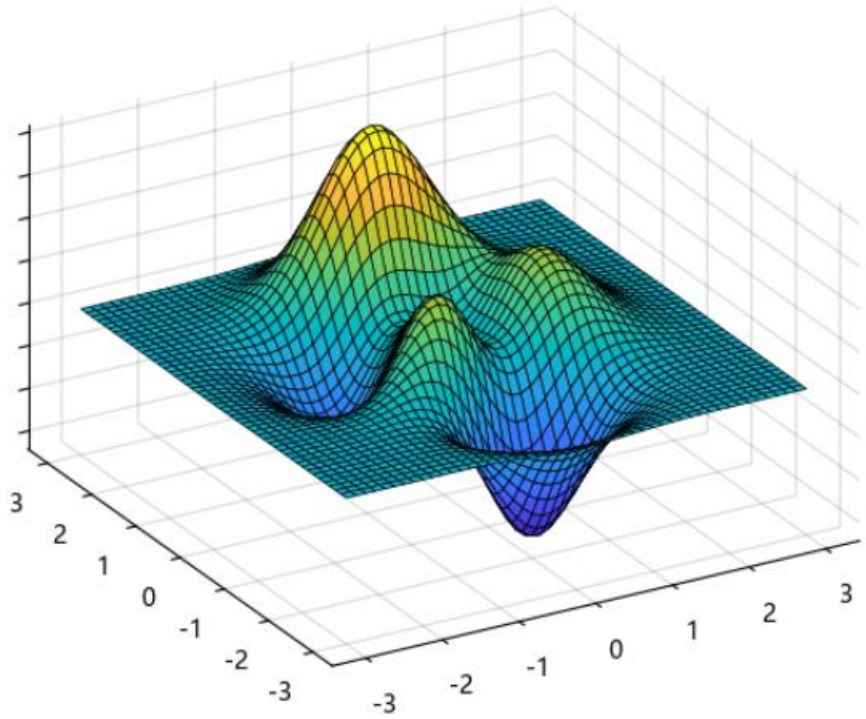

```
zticks("auto")
zticklabels("auto")
```



(4). 删除 z 轴刻度标签

通过将刻度值指定为空数组来删除沿 z 轴的刻度线。

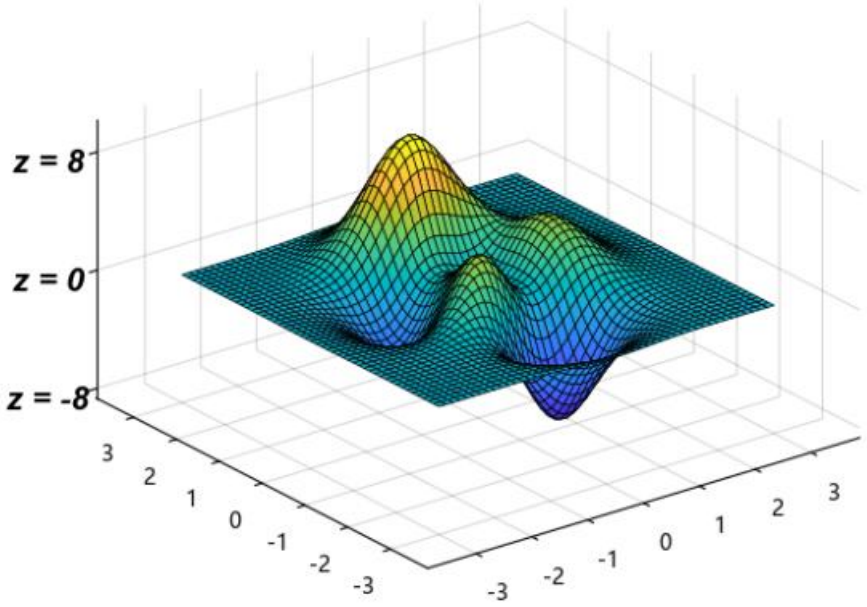
```
x,y,z = peaks();
surf(x,y,z)
zticklabels([])
```



(5). 自定义 Z 坐标轴的标签

创建一个 surf 图。在值 -8、0 和 8 处显示 z 轴的刻度线。然后为每个刻度线指定一个标签，设置标签字体的名称、大小、样式及粗细。

```
x,y,z = peaks();
surf(x,y,z)
zticks([-8 0 8])
zticklabels(["z = -8","z = 0","z = 8"];
    fontfamily="Arial",
    fontsize=20,
    fontstyle="italic",
    fontweight="bold",
)
```



xtickformat

指定 x 轴刻度标签格式

A. 语法

```
xtickformat(fmt)
xtickformat(datefmt)
xtickformat(durationfmt)
xtickformat(ax,___)

xfmt = xtickformat()
xfmt = xtickformat(ax)
```

B. 说明

xtickformat(fmt)设置数值 x 轴刻度标签的格式。例如，将 fmt 指定为"usd"将以美元符号显示标签。

xtickformat(datefmt)设置显示日期或时间的标签的格式。例如，将 datefmt 指定为"%m-%d-%y"可显示 04-19-16 之类的日期。此选项仅适用于具有日期时间值的 x 轴。

xtickformat(durationfmt)设置显示持续时间的标签的格式。例如，将 durationfmt 指定为"%M"可显示以分钟为单位的持续时间。此选项仅适用于具有持续时间值的 x 轴。

xtickformat(ax,___)使用 ax 指定的坐标区，而不是使用当前坐标区。指定 ax 作为第一个输入参数。

xfmt = xtickformat() 返回当前坐标区的 x 轴刻度标签所使用的格式样式。xfmt 可以为数值格式、日期格式或持续时间格式的字符向量，具体取决于 x 轴的标签的类型。

xfmt = xtickformat(ax) 返回 ax 指定的坐标区所使用的格式样式，而不是返回当前坐标区使用的格式样式。

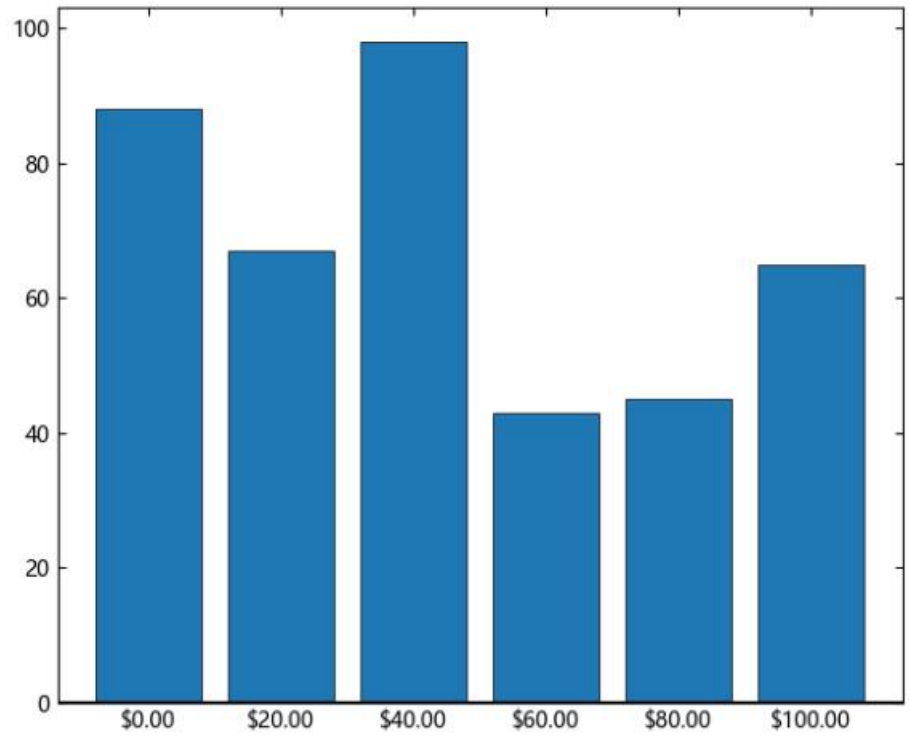
C. 示例

(1). 显示以美元为单位的刻度标签

显示以美元为单位的带和不带小数值的 x 轴刻度标签。

创建一个条形图。显示以美元为单位的沿 x 轴的刻度标签。

```
x = 0:20:100;
y = [88 67 98 43 45 65];
bar(x,y)
xtickformat("usd")
```

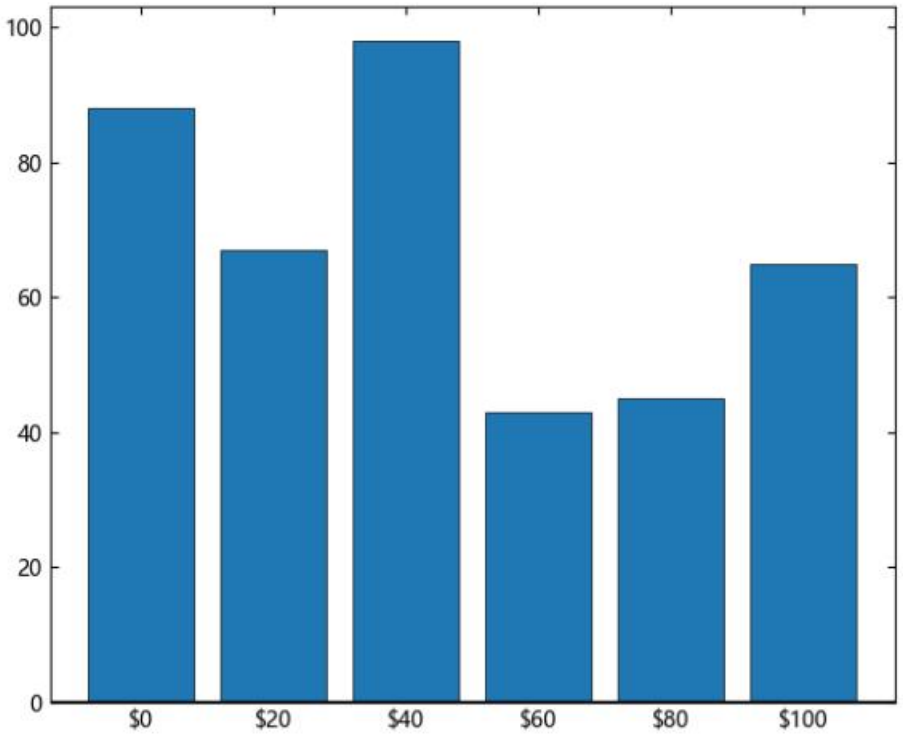


查询刻度标签格式。Syslab 返回的格式为包含格式化操作符的字符向量。

```
fmt = xtickformat()
```

fmt = "$%.0f"通过将数值格式调整为使用精度值 0 而不是 2，可显示不带小数值的刻度标签。

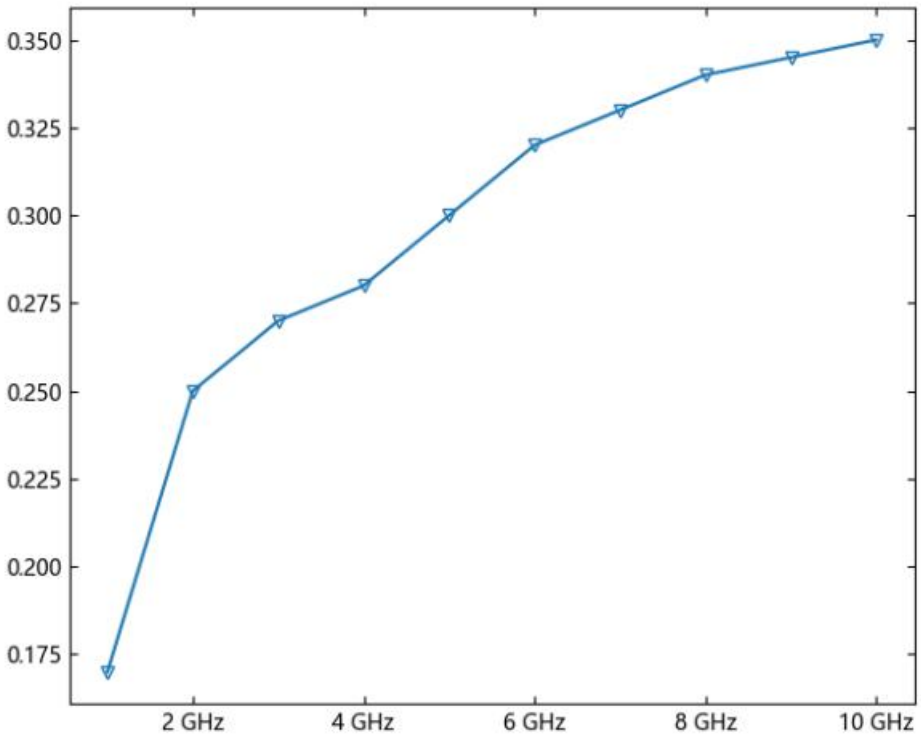
```
xtickformat("\$%.0f")
```



(2). 在每个刻度标签后显示文本

创建线图后，显示沿 x 轴的刻度标签，并在每个值后附加文本“GHz”。

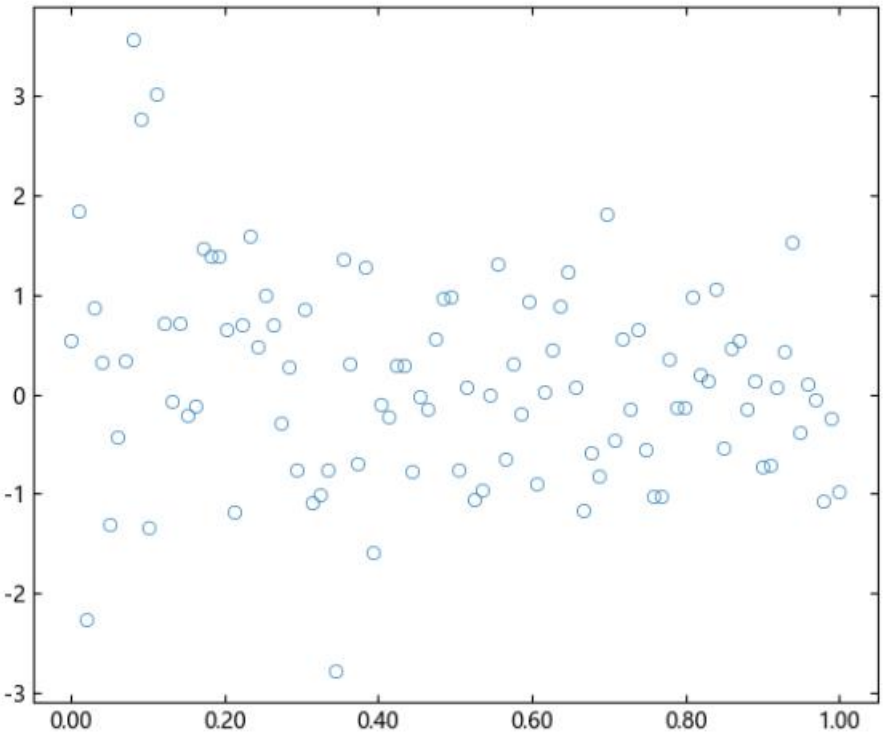
```
x = 1:10;
y = [.17, .25, .27, .28, .3, .32, .33, .34, .345, .35];
plot(x,y,"-v")
xtickformat("%g GHz")
```



(3). 控制显示的小数位数

创建散点图后，显示带两位小数的 x 轴刻度标签。通过向 xtickformat 传递一个数值格式的字符向量来控制小数位数，该数值格式对转换字符使用定点记数法且精度值为 2。

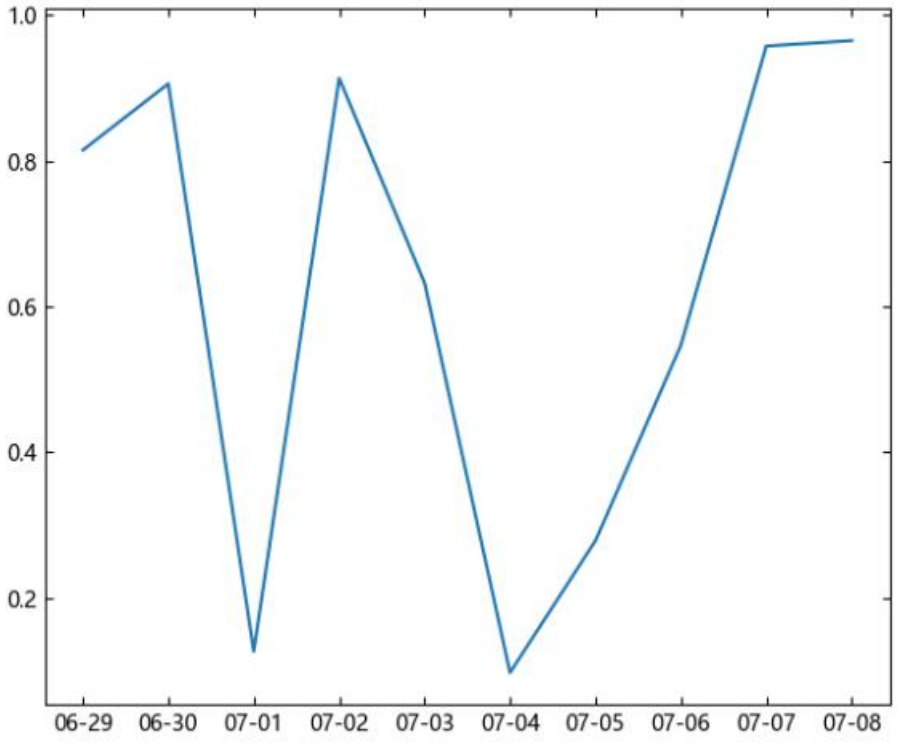
```
x = LinRange(0, 1, 100);
y = randn(100) .* cos.(x);
scatter(x, y)
xtickformat("%.2f")
```



(4). 日期刻度标签的格式

创建 x 轴为日期时间值的线图。然后，更改日期的格式

```
using Dates
days = []
for i in 1:10
   push!(days, Dates.Day(i))
end
t = Date(2014,06,28) .+ days
y=[0.8147  0.9058  0.1270  0.9134  0.6324  0.0975  0.2785  0.5469
0.9575  0.9649]
y=[y...]
plot(t,y);
xtickformat("%m-%d", true)
```



(5). 持续时间刻度标签的格式

创建 x 轴为持续时间值的线图。然后，更改刻度标签的格式。

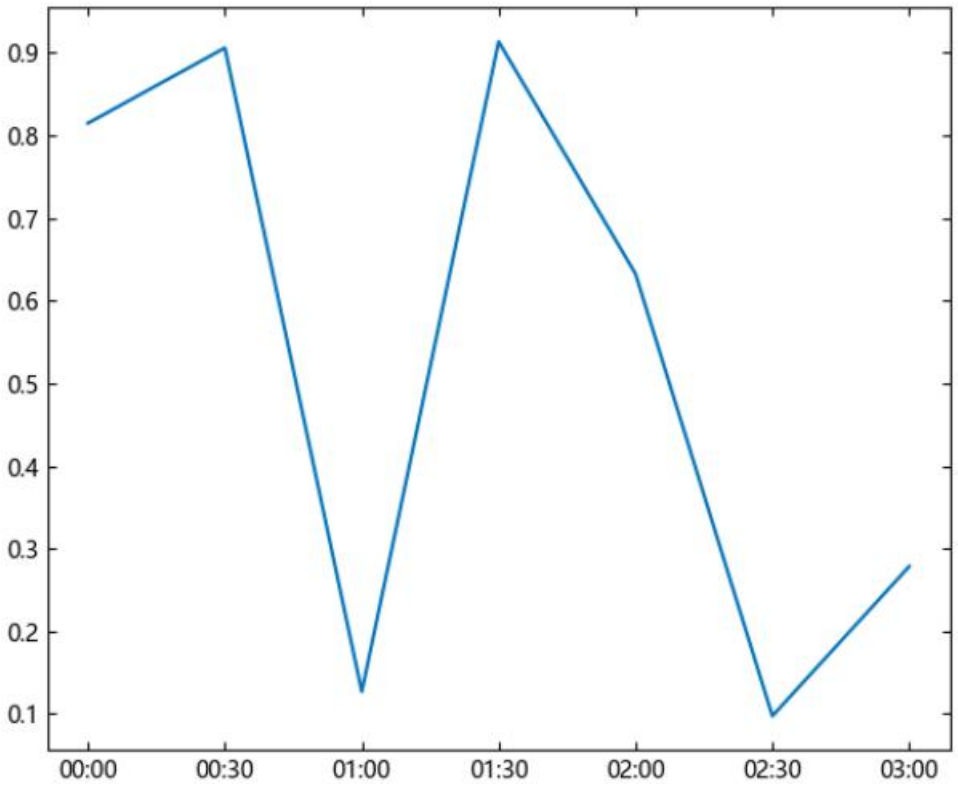
```
using Dates

t = range(Second(0), Second(180), step=Second(30))
t = [t...]

unix_t = unix2datetime(0)
dt = unix_t .+ t

y=[0.8147 0.9058 0.1270 0.9134 0.6324 0.0975 0.2785]
y=[y...]

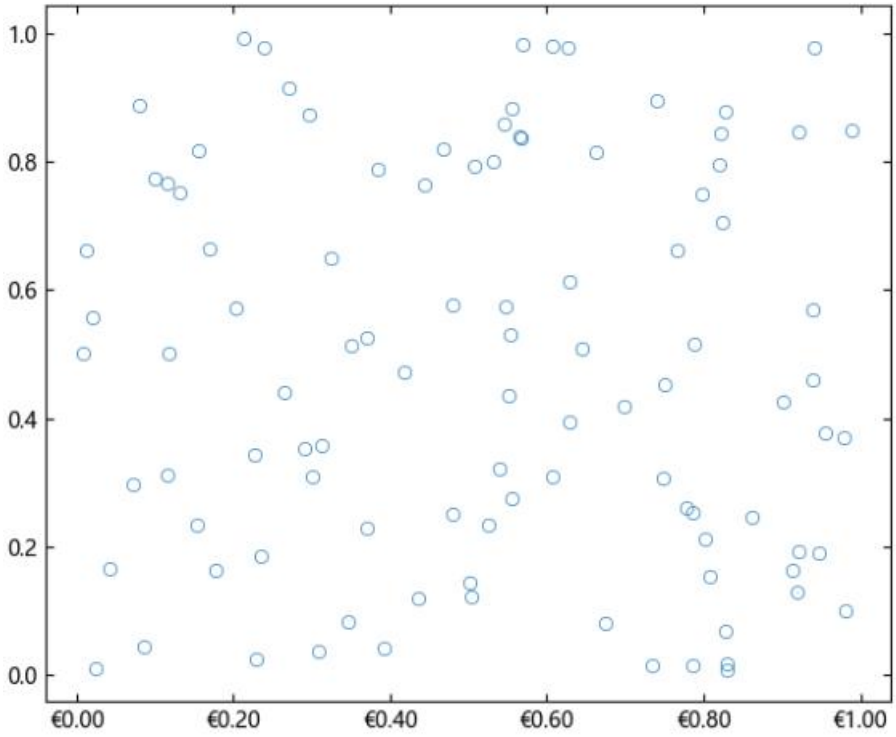
plot(dt,y);
xtickformat("%M:%S", true)
```



(6). 查询当前刻度标签格式

创建散点图并显示以欧元为单位的 x 轴刻度标签。然后，查询刻度标签格式。Syslab 返回的格式为包含格式化操作符的字符向量。

```
x = rand(100,1);  # 返回一个 100×1 的随机数矩阵
y = rand(100,1);  # 返回一个 100×1 的随机数矩阵
scatter(x,y)
xtickformat("eur")
```




```
xfmt = xtickformat()
```

```
xfmt = "\x20AC%,.2f"
```

(7). 特定坐标区的刻度标签格式

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。通过将 ax2 指定为 xtickformat 的第一个输入参数，指定下部图的 x 轴刻度标签格式。显示以美元为单位的刻度标签。

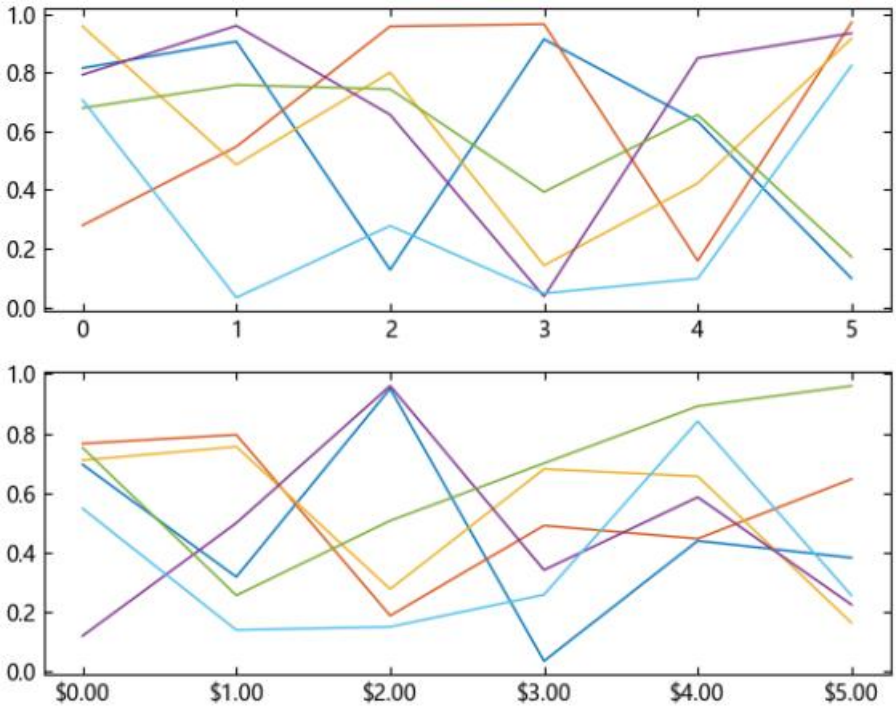
```
rand6=[
    0.8147    0.2785    0.9572    0.7922    0.6787    0.7060
    0.9058    0.5469    0.4854    0.9595    0.7577    0.0318
    0.1270    0.9575    0.8003    0.6557    0.7431    0.2769
    0.9134    0.9649    0.1419    0.0357    0.3922    0.0462
    0.6324    0.1576    0.4218    0.8491    0.6555    0.0971
    0.0975    0.9706    0.9157    0.9340    0.1712    0.8235
]
rand6_2=[
    0.6948    0.7655    0.7094    0.1190    0.7513    0.5472
    0.3171    0.7952    0.7547    0.4984    0.2551    0.1386
    0.9502    0.1869    0.2760    0.9597    0.5060    0.1493
    0.0344    0.4898    0.6797    0.3404    0.6991    0.2575
    0.4387    0.4456    0.6551    0.5853    0.8909    0.8407
    0.3816    0.6463    0.1626    0.2238    0.9593    0.2543
```

```
]

ax1 = subplot(2,1,1)
plot(ax1, rand6)

ax2 = subplot(2,1,2)
plot(ax2, rand6_2)

xtickformat(ax2, "usd")
```



ytickformat

指定 y 轴刻度标签格式

A. 语法

```
ytickformat(fmt)
ytickformat(datefmt)
ytickformat(durationfmt)
ytickformat(ax,___)

yfmt = ytickformat()
yfmt = ytickformat(ax)
```

B. 说明

ytickformat(fmt)设置数值 y 轴刻度标签的格式。例如，将 fmt 指定为"usd"将以美元符号显示标签。示例

ytickformat(datefmt) 设置显示日期或时间的标签的格式。例如，将 datefmt 指定为 "%M-%d-%y" 可显示 04-19-16 之类的日期。此选项仅适用于具有日期时间值的 y 轴。

ytickformat(durationfmt) 设置显示持续时间的标签的格式。例如，将 durationfmt 指定为 "%M" 可显示以分钟为单位的持续时间。此选项仅适用于具有持续时间值的 y 轴。

ytickformat(ax,___)使用 ax 指定的坐标区，而不是使用当前坐标区。指定 ax 作为第一个输入参数。

yfmt = ytickformat() 返回当前坐标区的 y 轴刻度标签所使用的格式样式。yfmt 可以为数值格式、日期格式或持续时间格式的字符向量，具体取决于 y 轴的标签类型。

yfmt = ytickformat(ax) 返回 ax 指定的坐标区所使用的格式样式，而不是返回当前坐标区使用的格式样式。

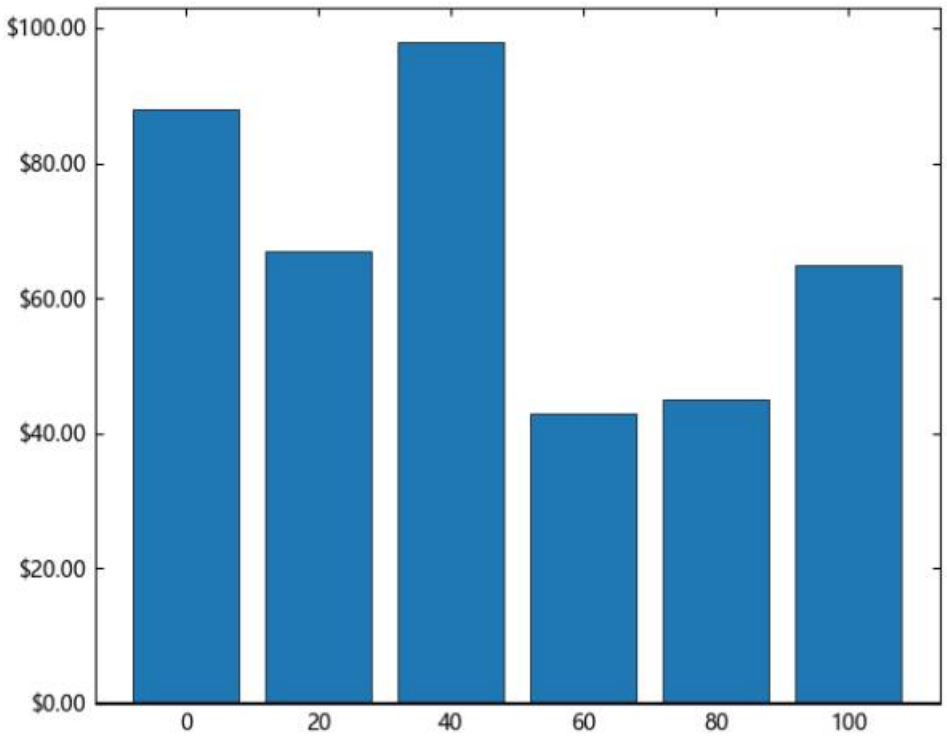
C. 示例

(1). 显示以美元为单位的刻度标签

显示以美元为单位的带和不带小数值的 y 轴刻度标签。

创建一个条形图。显示以美元为单位的沿 y 轴的刻度标签。

```
x = 0:20:100;
y = [88 67 98 43 45 65];
bar(x,y)
ytickformat("usd")
```

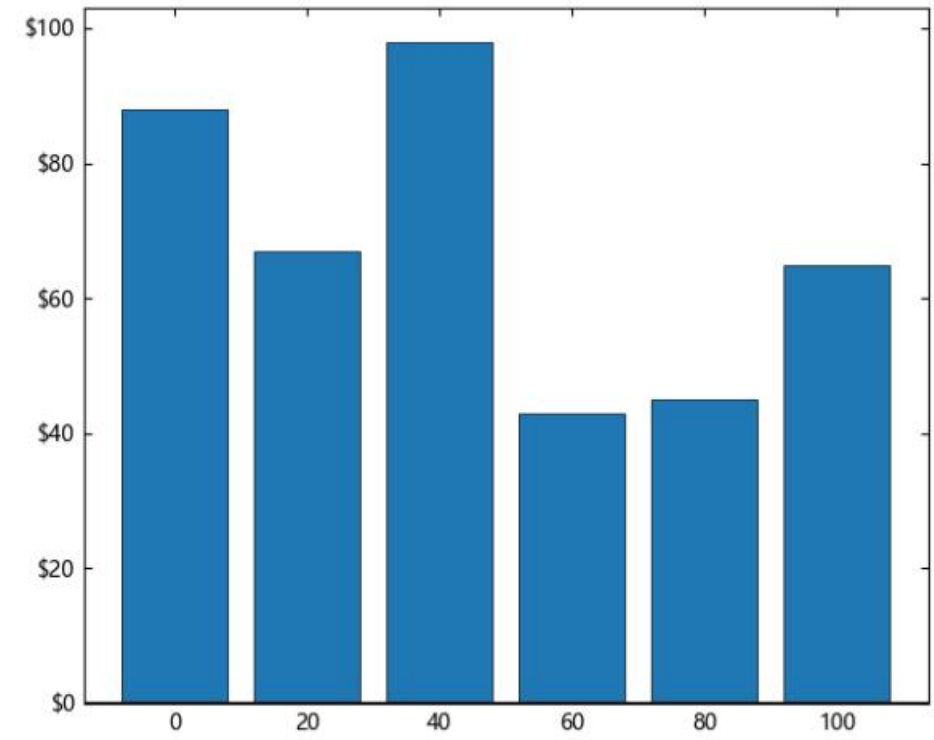


查询刻度标签格式。Syslab 返回的格式为包含格式化操作符的字符向量。

```
fmt = ytickformat()
```

fmt = "$%.0f"通过将数值格式调整为使用精度值 0 而不是 2，可显示不带小数值的刻度标签。

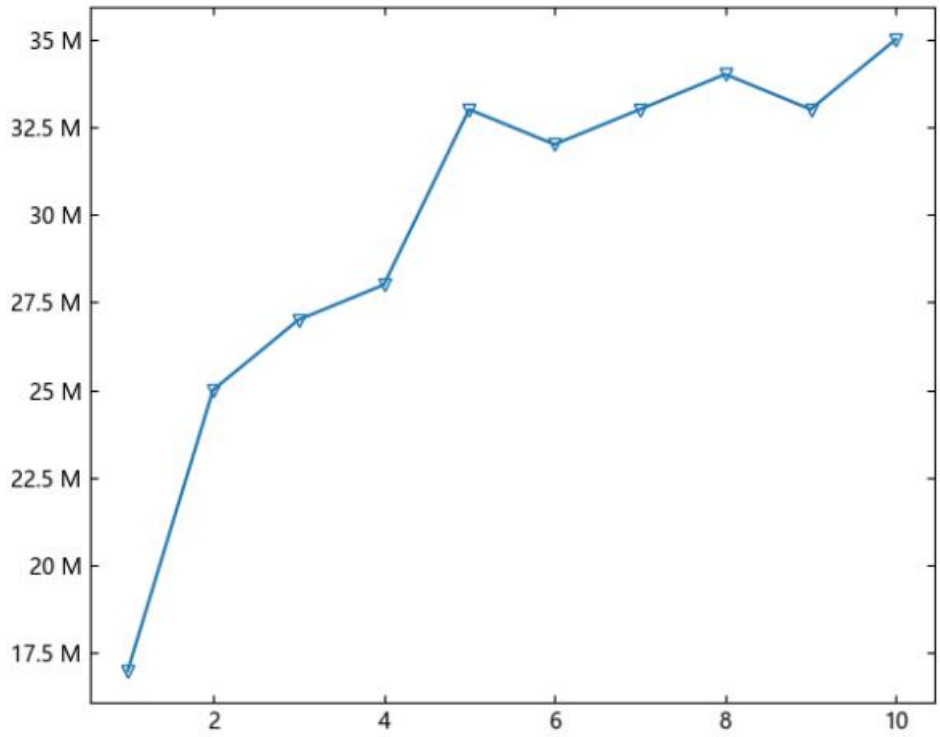
```
ytickformat("\$%.0f")
```



(2). 在每个刻度标签后显示文本

创建一个线图。显示沿 y 轴的刻度标签，并在每个值后附加文本“M”。

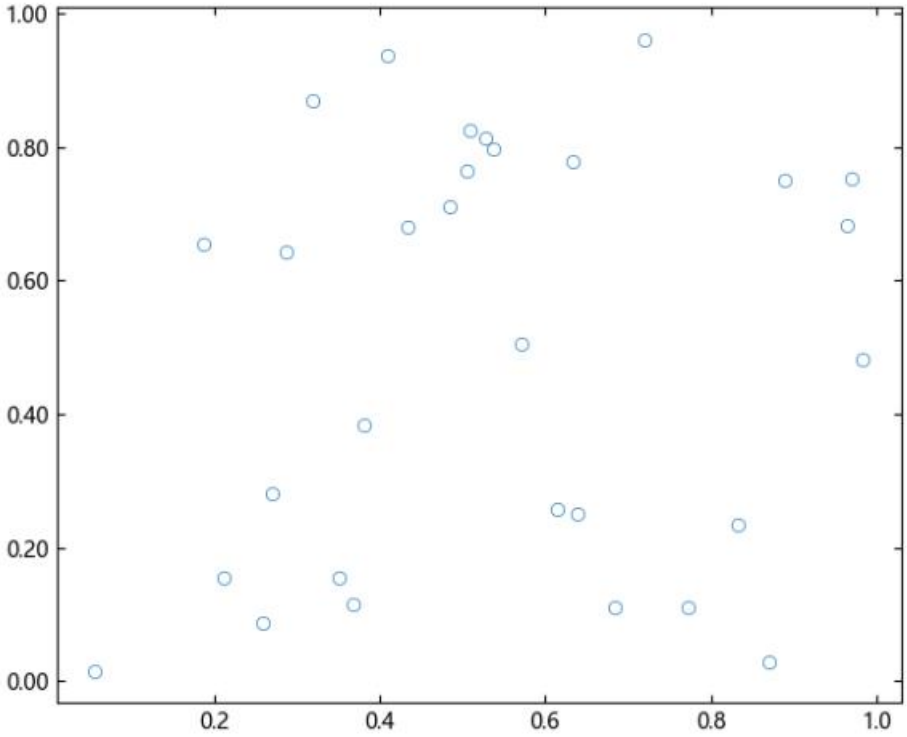
```
x = 1:10;
y = [17 25 27 28 33 32 33 34 33 35];
y=[y...]
plot(x,y,"-v")
ytickformat("%g M")
```



(3). 控制显示的小数位数

创建一个由随机数据组成的散点图。显示带两位小数的 y 轴刻度标签。通过向 ytickformat 传递一个数值格式的字符向量来控制小数位数，该数值格式对转换字符使用定点记数法且精度值为 2。

```
x = rand(30,1);  # 返回一个 30×1 的随机数矩阵
y = rand(30,1);  # 返回一个 30×1 的随机数矩阵
scatter(x,y)
ytickformat("%.2f")
```



(4). 特定坐标区的刻度标签格式

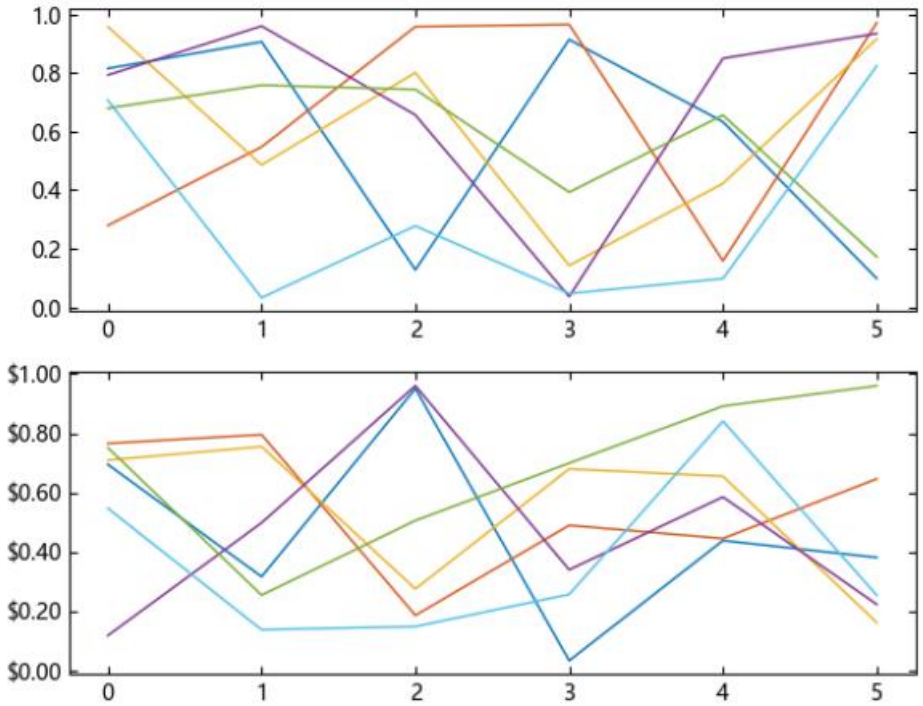
调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。通过将 ax2 指定为 ytickformat 的第一个输入参数，指定下部图的 y 轴刻度标签格式。显示以美元为单位的刻度标签。

```
rand6=[
    0.8147    0.2785    0.9572    0.7922    0.6787    0.7060
    0.9058    0.5469    0.4854    0.9595    0.7577    0.0318
    0.1270    0.9575    0.8003    0.6557    0.7431    0.2769
    0.9134    0.9649    0.1419    0.0357    0.3922    0.0462
    0.6324    0.1576    0.4218    0.8491    0.6555    0.0971
    0.0975    0.9706    0.9157    0.9340    0.1712    0.8235
]
rand6_2=[
    0.6948    0.7655    0.7094    0.1190    0.7513    0.5472
    0.3171    0.7952    0.7547    0.4984    0.2551    0.1386
    0.9502    0.1869    0.2760    0.9597    0.5060    0.1493
    0.0344    0.4898    0.6797    0.3404    0.6991    0.2575
    0.4387    0.4456    0.6551    0.5853    0.8909    0.8407
    0.3816    0.6463    0.1626    0.2238    0.9593    0.2543
]

ax1 = subplot(2,1,1)
plot(ax1, rand6)
```

```
ax2 = subplot(2,1,2)
plot(ax2, rand6_2)

ytickformat(ax2, "usd")
```



ztickformat

指定 z 轴刻度标签格式

A. 语法

```
ztickformat(fmt)
ztickformat(datefmt)
ztickformat(durationfmt)
ztickformat(ax,___)

zfmt = ztickformat()
zfmt = ztickformat(ax)
```

B. 说明

ztickformat(fmt) 设置数值 z 轴刻度标签的格式。例如，将 fmt 指定为

"usd" 将以美元符号显示标签。

ztickformat(datefmt) 设置显示日期或时间的标签的格式。例如，将 datefmt 指定为 "%m-%d-%y" 可显示 04-19-16 之类的日期。此选项仅适用于具有日期时间值的 z 轴。

ztickformat(durationfmt) 设置显示持续时间的标签的格式。例如，将 durationfmt 指定为 "%M" 可显示以分钟为单位的持续时间。此选项仅适用于具有持续时间值的 z 轴。

ztickformat(ax,___) 使用 ax 指定的坐标区，而不是使用当前坐标区。指定 ax 作为第一个输入参数。

zfmt = ztickformat 返回当前坐标区的 z 轴刻度标签所使用的格式样式。zfmt 可以为数值格式、日期格式或持续时间格式的字符向量，具体取决于 z 轴的标签类型。

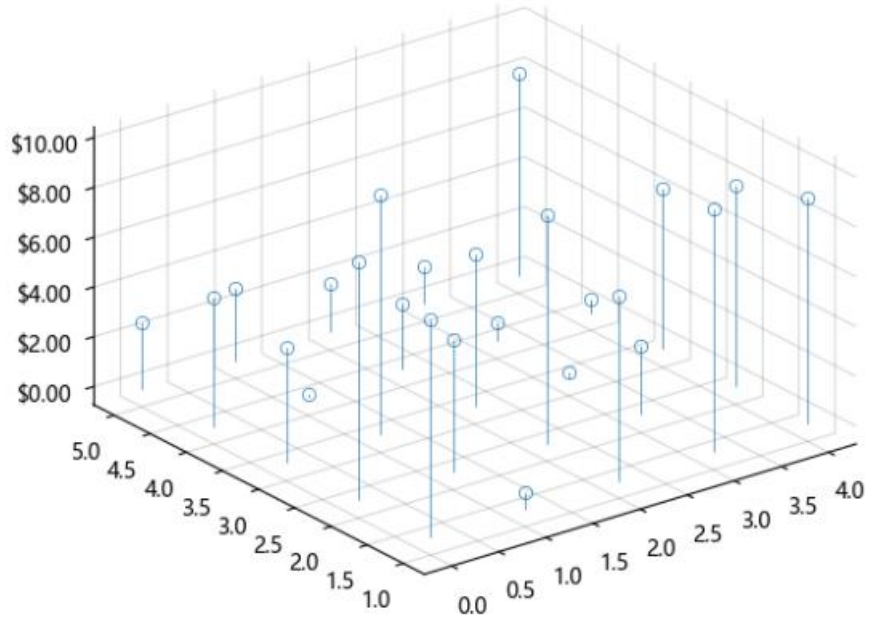
zfmt = ztickformat(ax) 返回 ax 指定的坐标区所使用的格式样式，而不是返回当前坐标区使用的格式样式。

C. 示例

(1). 显示以美元为单位的刻度标签

创建一个由随机数据组成的三维针状图。显示以美元为单位的沿 z 轴的刻度标签。

```
z = 10 * rand(5, 5);
stem3(z)
ztickformat("usd")
```



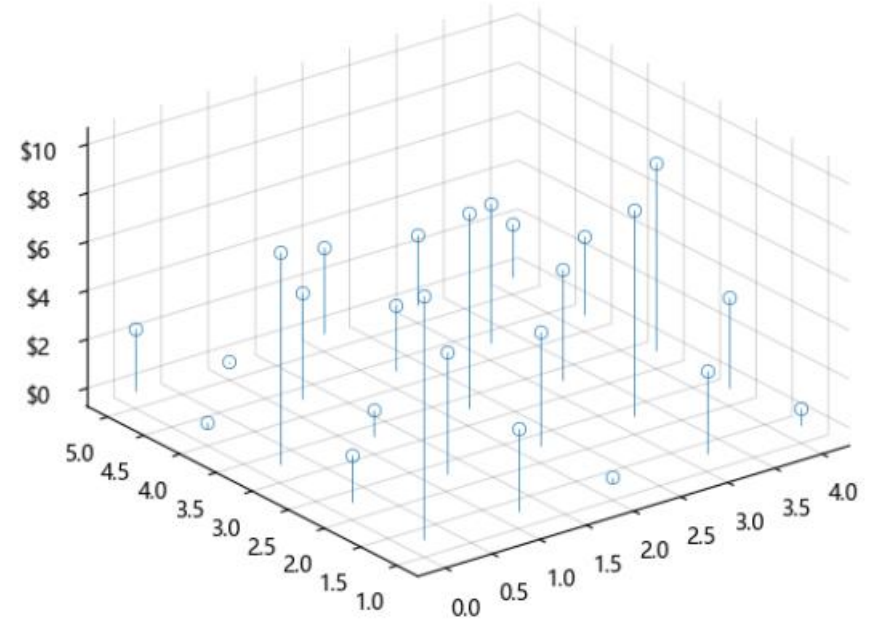
查询刻度标签格式。Syslab 返回的格式为包含格式化操作符的字符向量。

```
fmt = ztickformat()
```

```
fmt =
"\$%.2f"
```

通过将数值格式调整为使用精度值 0 而不是 2，可显示不带小数值的刻度标签。

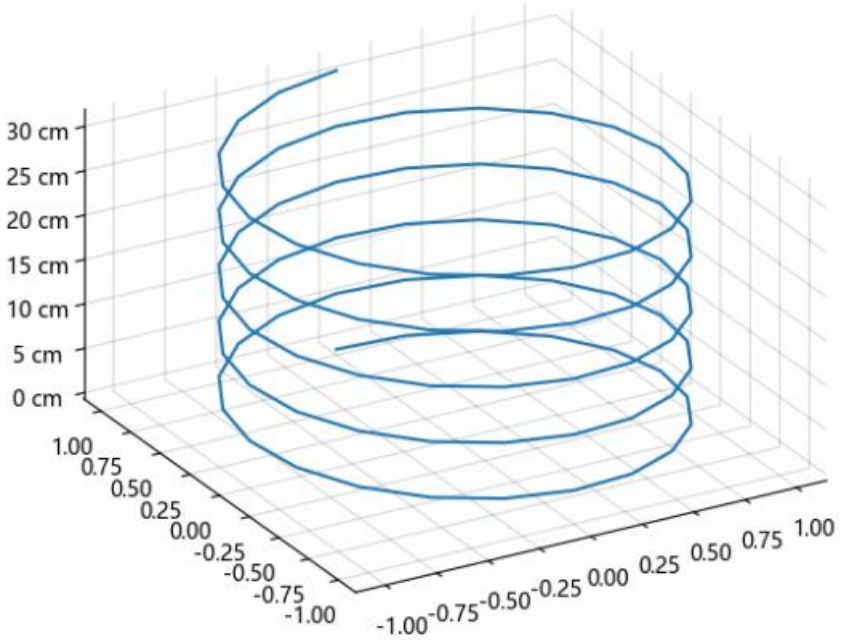
```
ztickformat("\$%.0f")
```



(2). 在每个刻度标签后显示文本

创建一个三维线图。显示沿 z 轴的刻度标签，并在每个值后附加文本“cm”。

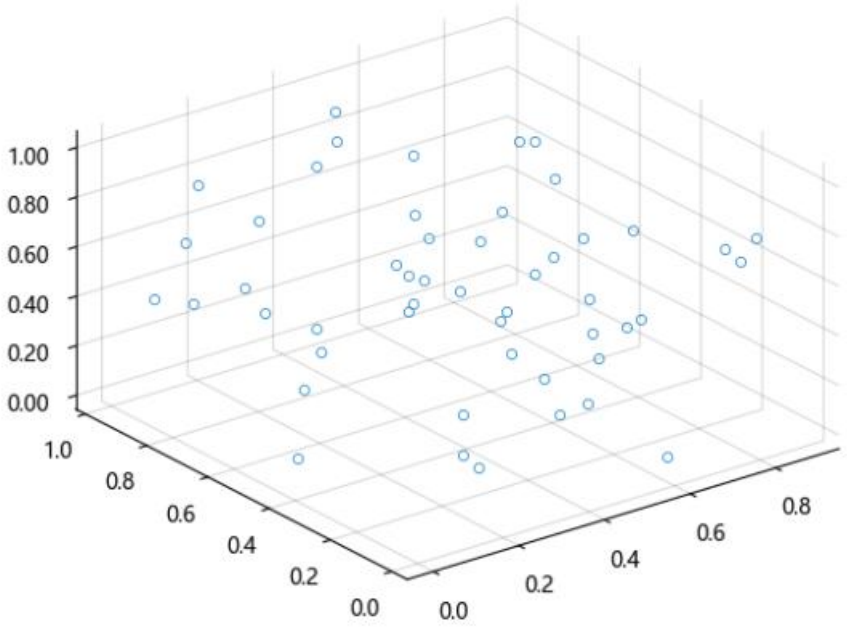
```
t = 0:pi/10:10*pi;
st = sin.(t);
ct = cos.(t);
plot3(st,ct,t)
grid("on")
ztickformat("%g cm")
```



(3). 控制显示的小数位数

创建一个由随机数据组成的三维散点图。显示带两位小数的 z 轴刻度标签。通过向 ztickformat 传递一个数值格式的字符向量来控制小数位数，该数值格式对转换字符使用定点记数法且精度值为 2。

```
x = rand(50, 1);
y = rand(50, 1);
z = rand(50, 1);
scatter3(x, y, z)
ztickformat("%.2f")
```

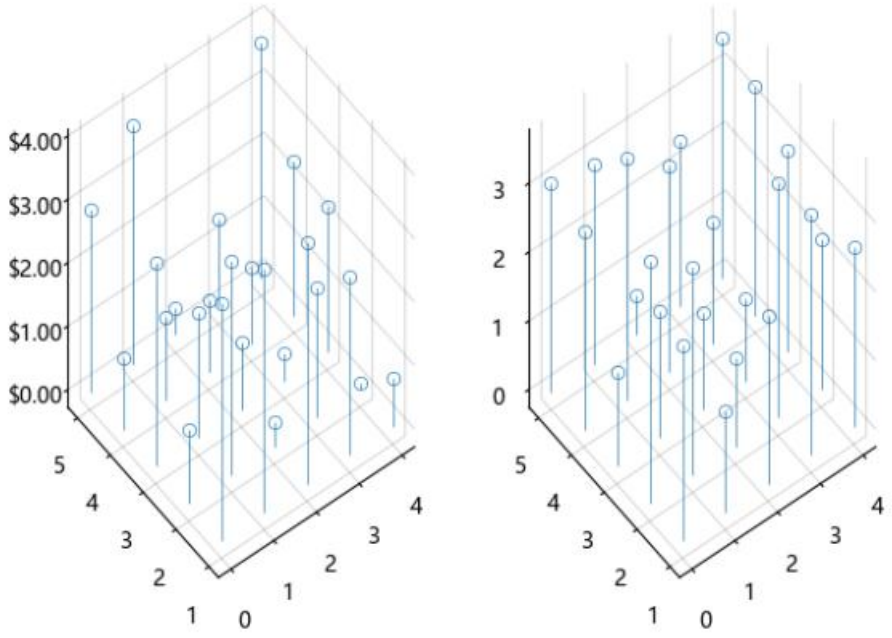


(4). 特定坐标区的刻度标签格式

您可以使用 subplot 和 stem3 函数显示分块图。调用 subplot 函数以创建 1×2 分块图布局。调用 stem3 函数以创建坐标区对象 ax1 和 ax2。创建两个三维针状图，通过将 ax1 指定为 ztickformat 的第一个输入参数来指定左侧图的 z 轴的刻度标签格式。显示以美元为单位的刻度标签。

```
ax1 = subplot(1, 2, 1; projection="3d");
stem3(ax1, 4 * rand(5, 5))
ztickformat(ax1, "usd")

ax2 = subplot(1, 2, 2; projection="3d");
stem3(ax2, 4 * rand(5, 5))
```



xtickangle

旋转 x 轴刻度标签

A. 语法

```
xtickangle(angle)
xtickangle(ax,angle)

ang = xtickangle()
ang = xtickangle(ax)
```

B. 说明

xtickangle(angle)将当前坐标区的 x 轴刻度标签旋转到指定角度（以度为单位），其中 0 表示水平。指定正值表示逆时针旋转，负值表示顺时针旋转。示例

xtickangle(ax,angle)旋转 ax 指定的坐标区的刻度标签，而不是旋转当前坐标区的刻度标签。

ang = xtickangle() 以标量值形式返回当前坐标区的 x 轴刻度标签的旋转角度（以度为单位）。正值表示逆时针旋转。负值表示顺时针旋转。

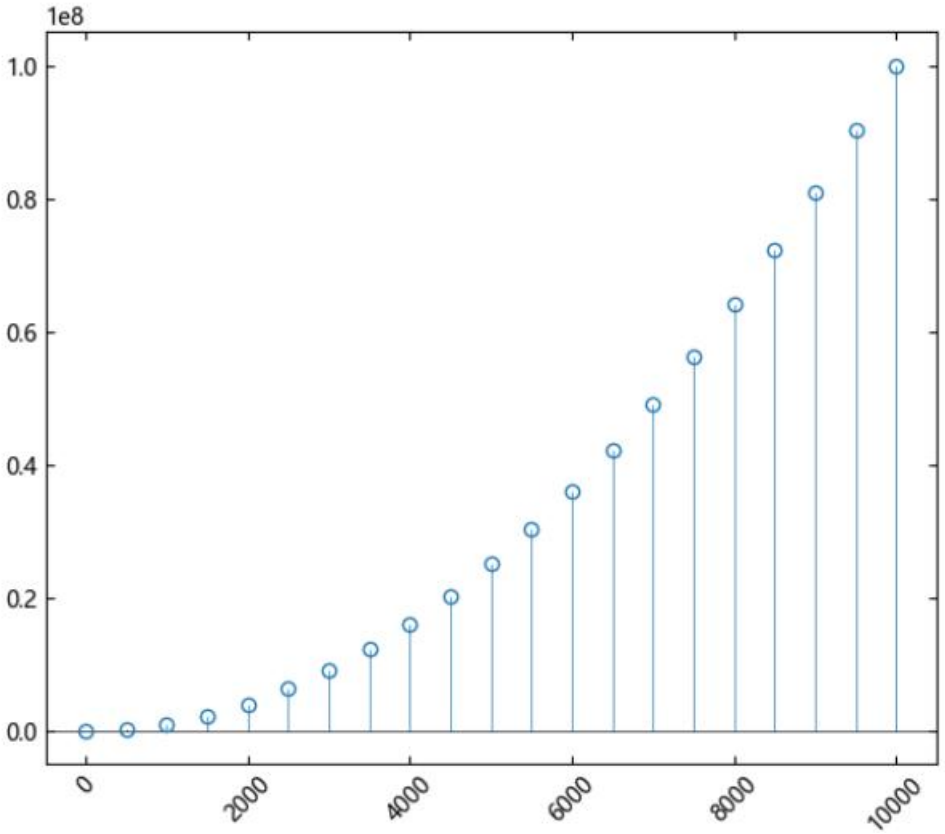
ang = xtickangle(ax) 使用 ax 指定的坐标区，而不是使用当前坐标区。

C. 示例

(1). 旋转 x 轴刻度标签

创建一个针状图并旋转 x 轴刻度标签，以使它们与水平平面呈 45 度角显示。

```
x = LinRange(0, 10000, 21);
y = x .^ 2;
stem(x, y)
ang = xtickangle(45)
```



(2). 旋转特定坐标区的 x 轴刻度标签

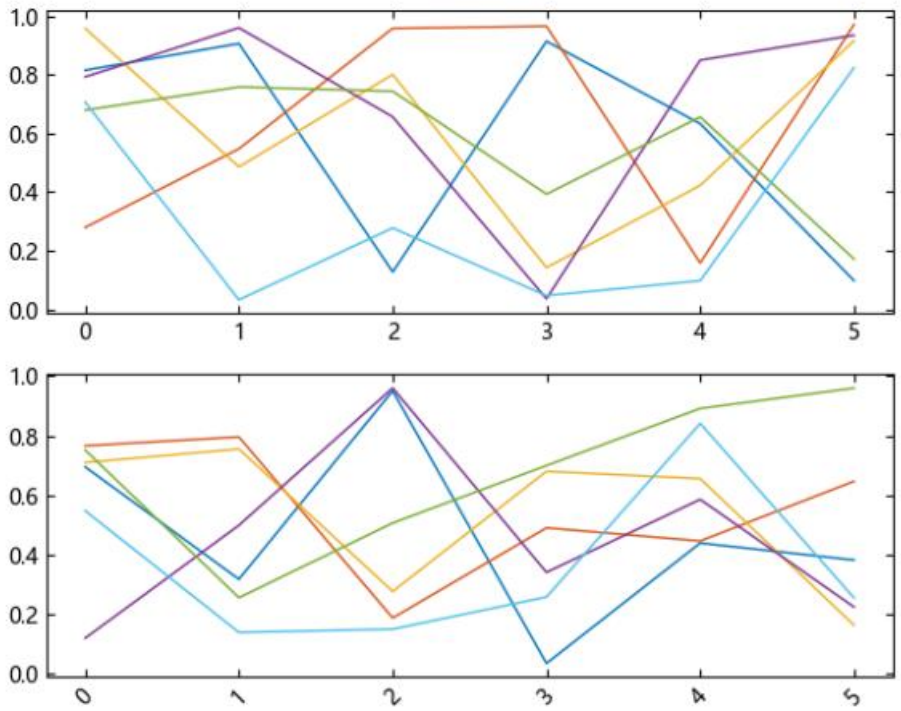
调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。在每个坐标区中绘制。然后通过将 ax2 指定为 xtickangle 的第一个输入参数，旋转下部图的 x 轴刻度标签。

```
rand6=[
    0.8147    0.2785    0.9572    0.7922    0.6787    0.7060
    0.9058    0.5469    0.4854    0.9595    0.7577    0.0318
    0.1270    0.9575    0.8003    0.6557    0.7431    0.2769
    0.9134    0.9649    0.1419    0.0357    0.3922    0.0462
    0.6324    0.1576    0.4218    0.8491    0.6555    0.0971
```

```
    0.0975    0.9706    0.9157    0.9340    0.1712    0.8235
]
rand6_2=[
    0.6948    0.7655    0.7094    0.1190    0.7513    0.5472
    0.3171    0.7952    0.7547    0.4984    0.2551    0.1386
    0.9502    0.1869    0.2760    0.9597    0.5060    0.1493
    0.0344    0.4898    0.6797    0.3404    0.6991    0.2575
    0.4387    0.4456    0.6551    0.5853    0.8909    0.8407
    0.3816    0.6463    0.1626    0.2238    0.9593    0.2543
]

ax1=subplot(2,1,1)
plot(ax1,rand6)

ax2=subplot(2,1,2)
plot(ax2,rand6_2)
xtickangle(ax2,45)
```



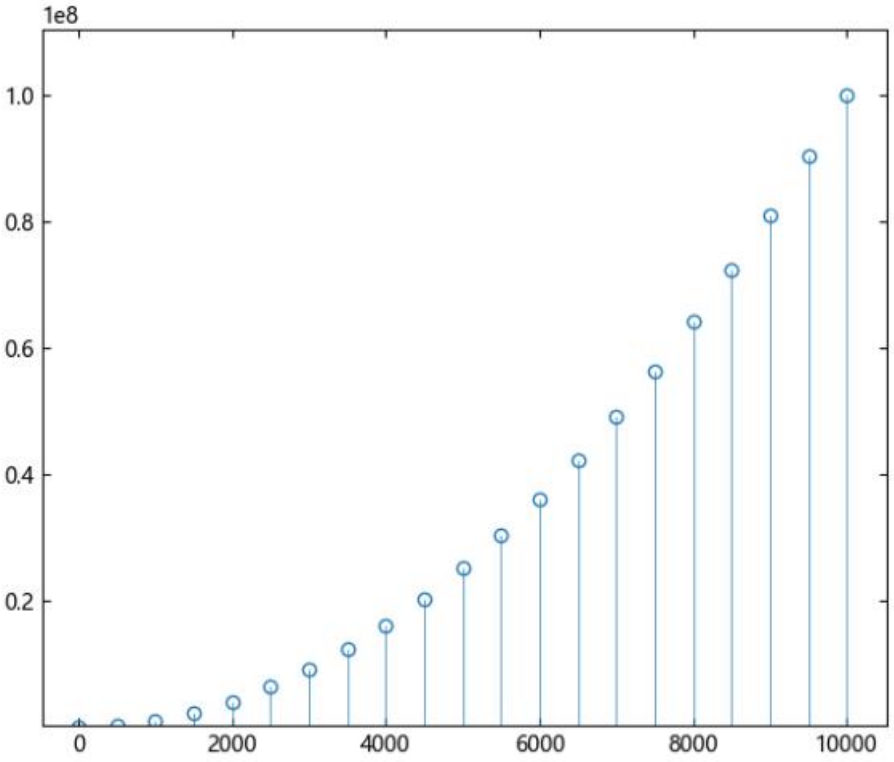
(3). 查询 x 轴刻度标签的当前角度

创建一个针状图。然后，查询 x 轴刻度标签的旋转角度。默认情况下，不会旋转标签。

```
x = LinRange(0, 10000, 21);

y = x .^ 2;
```

```
stem(x, y)
```




```
ang = xtickangle()
```

```
ang = 0
```

ytickangle

旋转 y 轴刻度标签

A. 语法

```
ytickangle(angle)
ytickangle(ax,angle)

ang = ytickangle()
ang = ytickangle(ax)
```

B. 说明

ytickangle(angle)将当前坐标区的 y 轴刻度标签旋转到指定角度（以度为单位），其中 0 表示水平。指定正值表示逆时针旋转，负值表示顺时针旋转。

ytickangle(ax,angle)旋转 ax 指定的坐标区的刻度标签，而不是旋转当前坐标区的刻度标签。

ang = ytickangle() 以标量值形式返回当前坐标区的 y 轴刻度标签的旋转角度（以度为单位）。正值表示逆时针旋转。负值表示顺时针旋转。

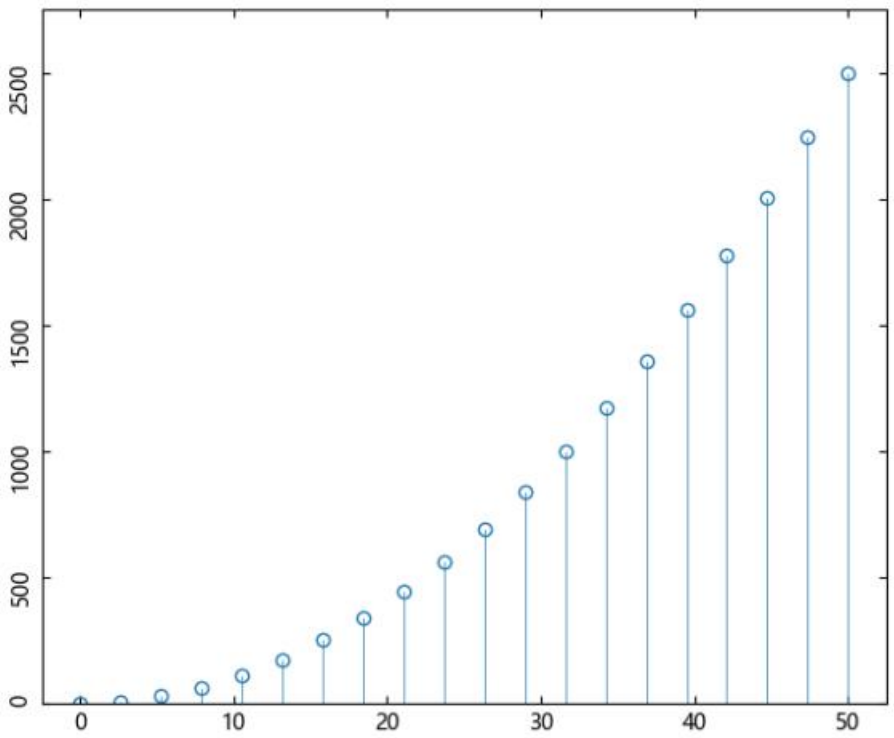
ang = ytickangle(ax) 使用 ax 指定的坐标区，而不是使用当前坐标区。

C. 示例

(1). 旋转 y 轴刻度标签

创建一个针状图并将 y 轴刻度标签旋转 90 度，以使它们垂直显示。

```
x = LinRange(0, 50, 20);
y = x .^ 2;
stem(x, y)
ytickangle(90)
```



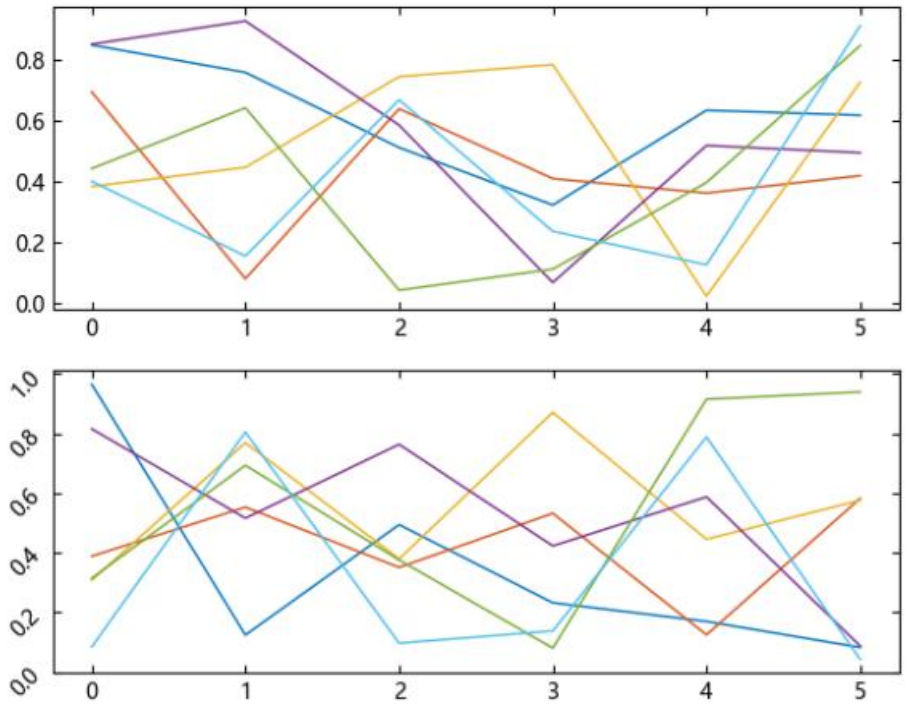
(2). 旋转特定坐标区的 y 轴刻度标签

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。在每个坐标区中绘制。然后通过将 ax2 指定为 ytickangle 的第一个输入参数，旋转下部图的 y 轴刻度标签。

```
ax1=subplot(2,1,1)
plot(ax1,rand(6,6))

ax2=subplot(2,1,2)
```

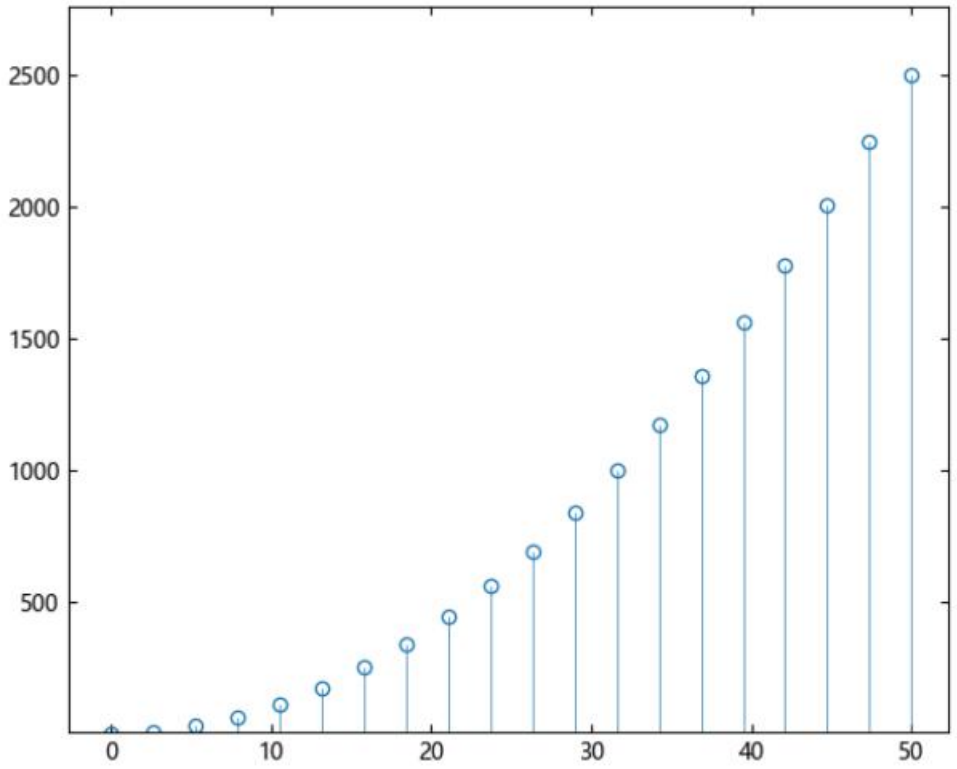
```
plot(ax2,rand(6,6))
ytickangle(ax2,45)
```



(3). 查询 y 轴刻度标签的当前角度

创建一个针状图。然后，查询 y 轴刻度标签的旋转角度。默认情况下，不会旋转标签。

```
x = LinRange(0, 50, 20);
y = x .^ 2;
stem(x, y)
```




```
ang = ytickangle()
```

```
ang = 0
```

ztickangle

旋转 z 轴刻度标签

A. 语法

```
ztickangle(angle)
ztickangle(ax,angle)

ang = ztickangle()
ang = ztickangle(ax)
```

B. 说明

ztickangle(angle)将当前坐标区的 z 轴刻度标签旋转到指定角度（以度为单位），其中 0 表示水平。指定正值表示逆时针旋转，负值表示顺时针旋转。示例

ztickangle(ax,angle) 旋转 ax 指定的坐标区的刻度标签，而不是旋转当前坐标区的刻度标签。

ang = ztickangle() 以标量值形式返回当前坐标区的 z 轴刻度标签的旋转角度（以度为单位）。正值表示逆时针旋转。负值表示顺时针旋转。

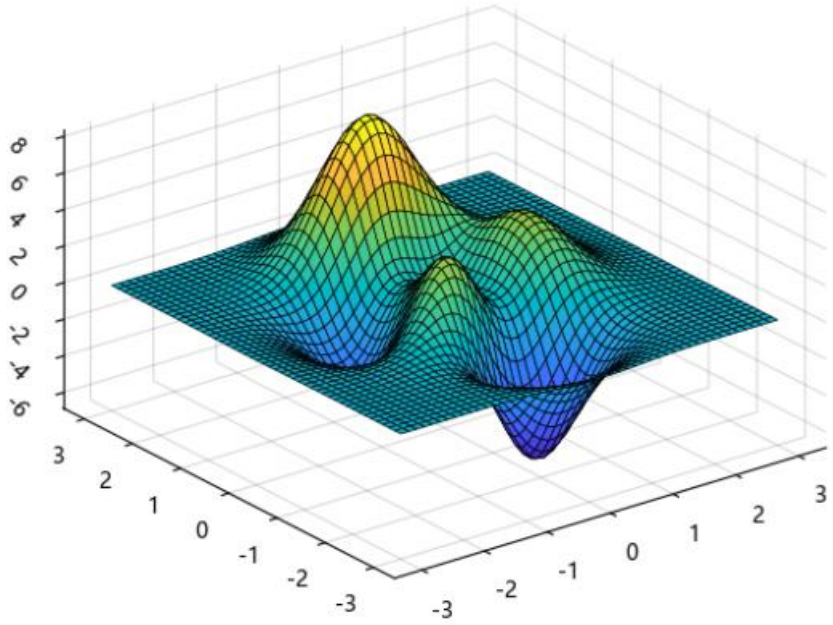
ang = ztickangle(ax) 使用 ax 指定的坐标区，而不是使用当前坐标区。

C. 示例

(1). 旋转 z 轴刻度标签

创建一个曲面图并将 z 轴刻度标签顺时针旋转 45 度。

```
using TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/数据文件/peaks49.jl"
include(source_path)
surf(X, Y, Z)
ztickangle(-45)
```



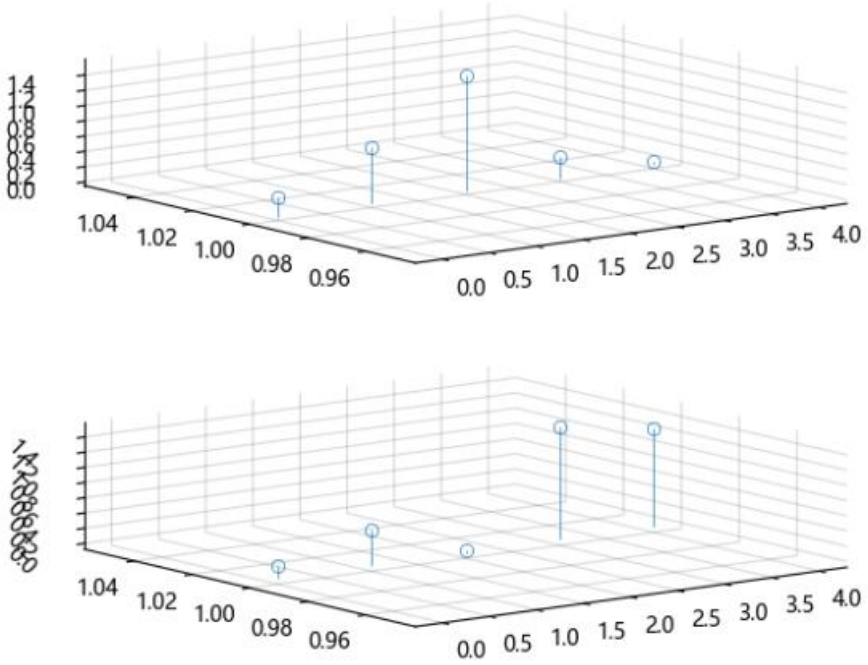
(2). 旋转特定坐标区的 z 轴刻度标签

您可以使用 subplot 和 stem3 函数显示分块图。调用 subplot 函数以创建一个 2×1 分块图布局。调用 stem3 函数以创建坐标区对象 ax1 和 ax2。在每个坐标区中绘制。然后通过将 ax2 指定为 ztickangle 的第一个输入参数，旋转下部图的 z 轴刻度标签。

```
ax1 = subplot(2, 1, 1; projection="3d")
stem3(ax1, 2 * rand(5))

ax2 = subplot(2, 1, 2; projection="3d")
```

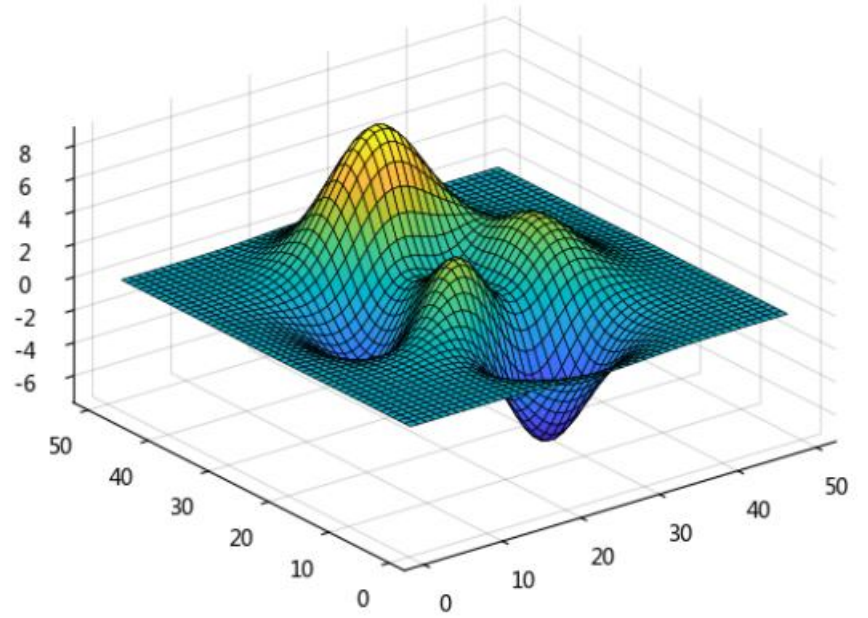
```
stem3(ax2, 2 * rand(5))
ztickangle(ax2, -45)
```



(3). 查询 z 轴刻度标签的当前角度

创建一个曲面图。然后，查询 z 轴刻度标签的旋转角度。默认情况下，不会旋转标签。

```
using TyPlot
x, y, z = peaks()
surf(z)
```



```
ang = ztickangle()
```

```
ang = 0
```

3.3.2.2.3 多个绘图

组合绘图

函数名	简介
hold	添加新绘图时保留当前绘图
yyaxis	创建具有两个 y 轴的图
plotyy	创建具有两个 y 轴的图形
legend	在坐标区上添加图例
colororder	为可视化多个数据序列设置色序

图块化绘图

函数名	简介
subplot	在各个分块位置创建坐标区

hold

添加新绘图时保留当前绘图

A. 语法

```
hold("on")
hold("off")

hold(ax,___)
```

B. 说明

hold("on")保留当前坐标区中的绘图，从而使新添加到坐标区中的绘图不会删除现有绘图。新绘图基于坐标区的 ColorOrder 和 LineStyleOrder 属性使用后续的颜色和线型。Syslab 将调整坐标区的范围、刻度线和刻度标签以显示完整范围的数据。如果不存在坐标区，hold 命令会创建坐标区。

hold("off")将保留状态设置为 "off"，从而使新添加到坐标区中的绘图清除现有绘图并重置所有的坐标区属性。添加到坐标区的下一个绘图基于坐标区的 ColorOrder 和 LineStyleOrder 属性使用第一个颜色和线型。此选项为默认行为。

hold(ax,___)为 ax 指定的坐标区而非当前坐标区设置 hold 状态。请在前面任何语法中的所有其他参数之前指定 ax。使用双引号将"on"和"off"输入引起来，例如 hold(ax,"on")。

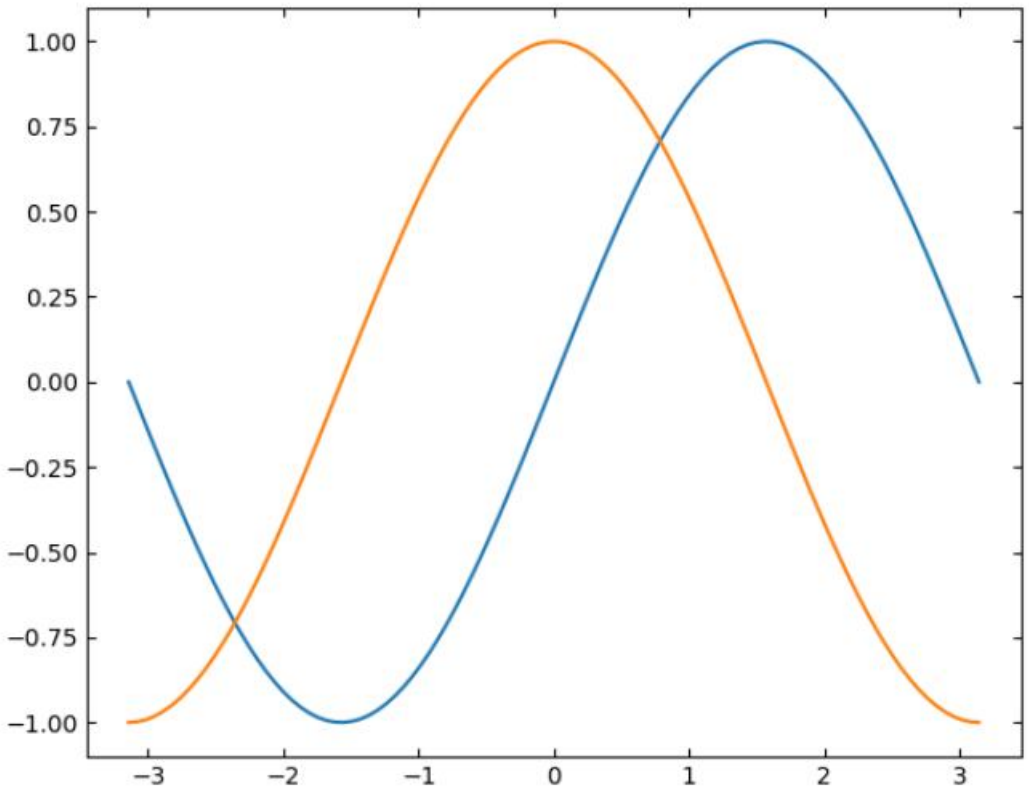
C. 示例

(1). 在现有坐标区上添加线图

创建一个线图。使用 hold("on") 添加第二个线图，而不删除已有的线图。新绘图基于坐标区的 ColorOrder 和 LineStyleOrder 属性使用下一种颜色和线型。然后将 hold 状态重置为 "off"。

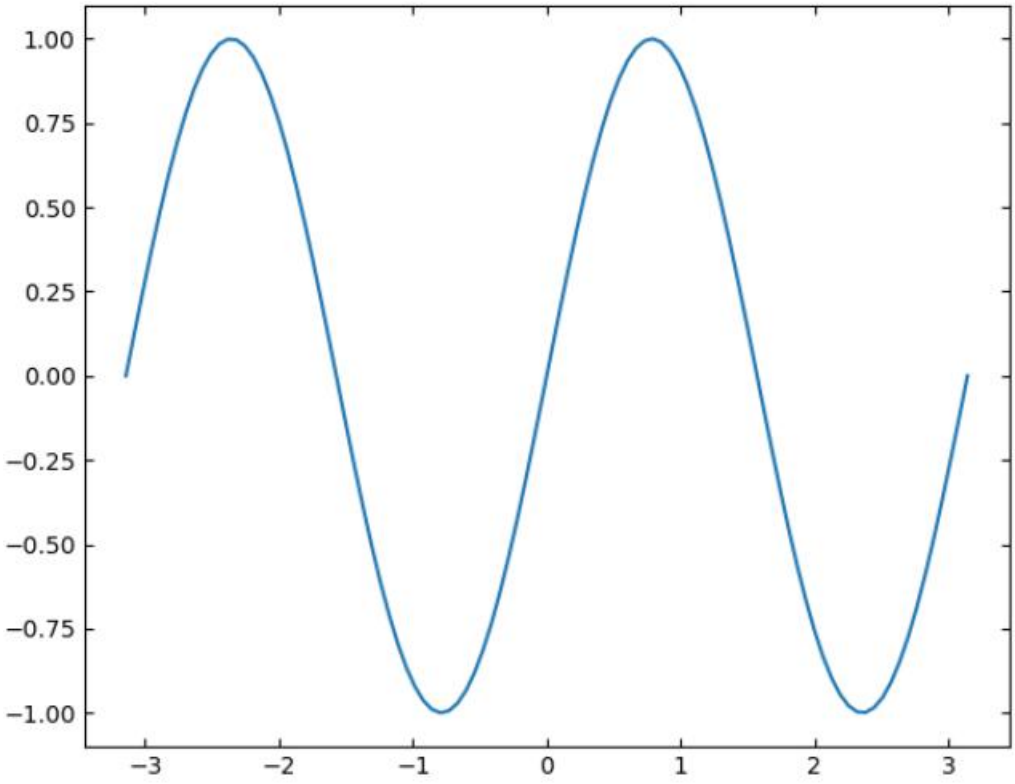
```
x = LinRange(-pi, pi, 100);
y1 = sin.(x);
plot(x, y1)

hold("on")
y2 = cos.(x);
plot(x, y2)
hold("off")
```



当 hold 状态为 "off" 时，新绘图将删除现有绘图。新绘图从色序和线型序列的开头开始。

```
y3 = sin.(2 * x);
plot(x, y3)
```



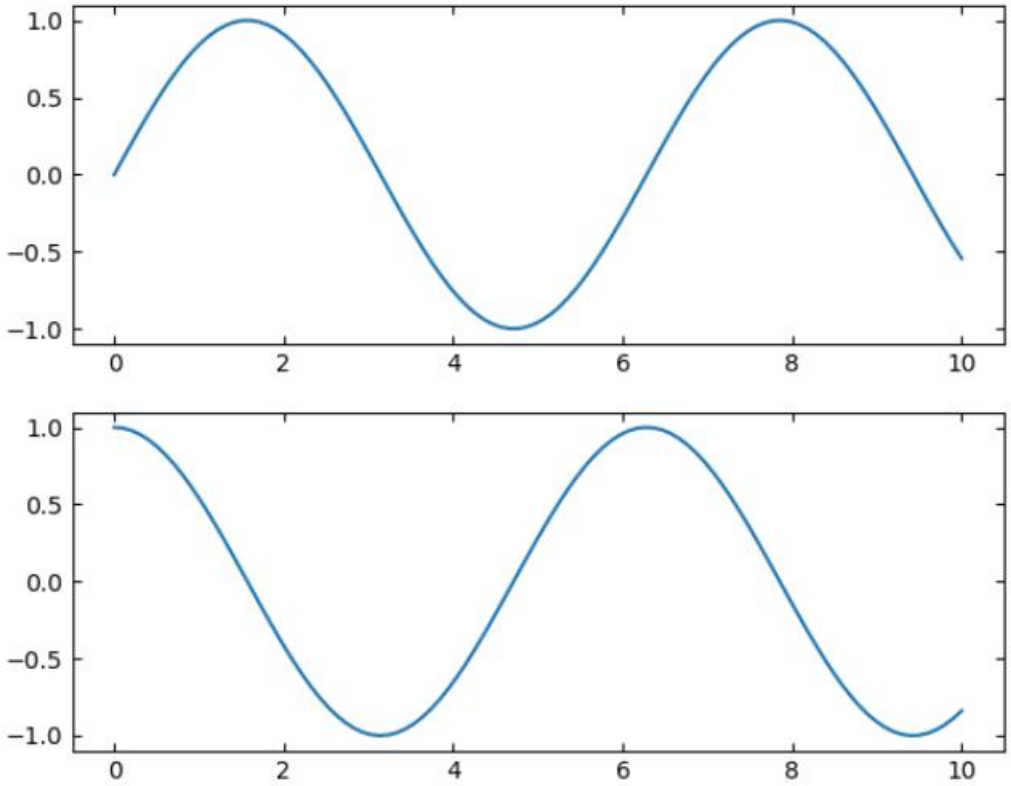
(2). 指定特定坐标区的保留状态

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。通过将 ax1 传递给 plot 函数来创建顶部绘图。重复该过程以创建底部绘图。在每个坐标区中绘制一个正弦波图。

```
x = LinRange(0, 10, 100);
y1 = sin.(x);
y2 = cos.(x);

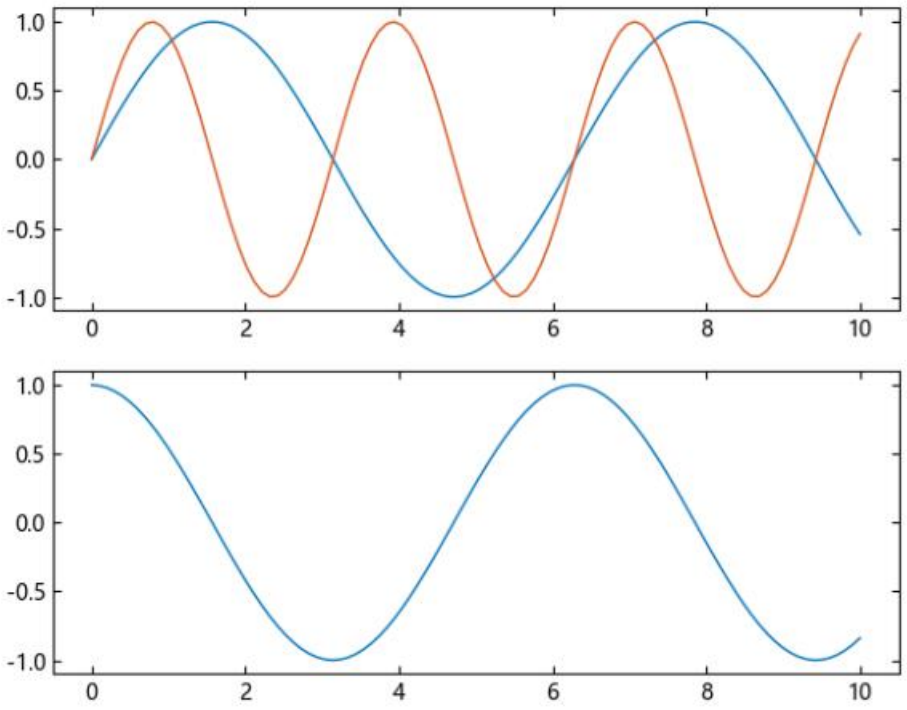
ax1 = subplot(2, 1, 1)
plot(ax1, x, y1)

ax2 = subplot(2, 1, 2)
plot(ax2, x, y2)
```



向上方坐标区添加第二个正弦波。

```
hold(ax1, "on")
y3 = sin.(2 * x);
plot(ax1, x, y3)
hold(ax1, "off")
```

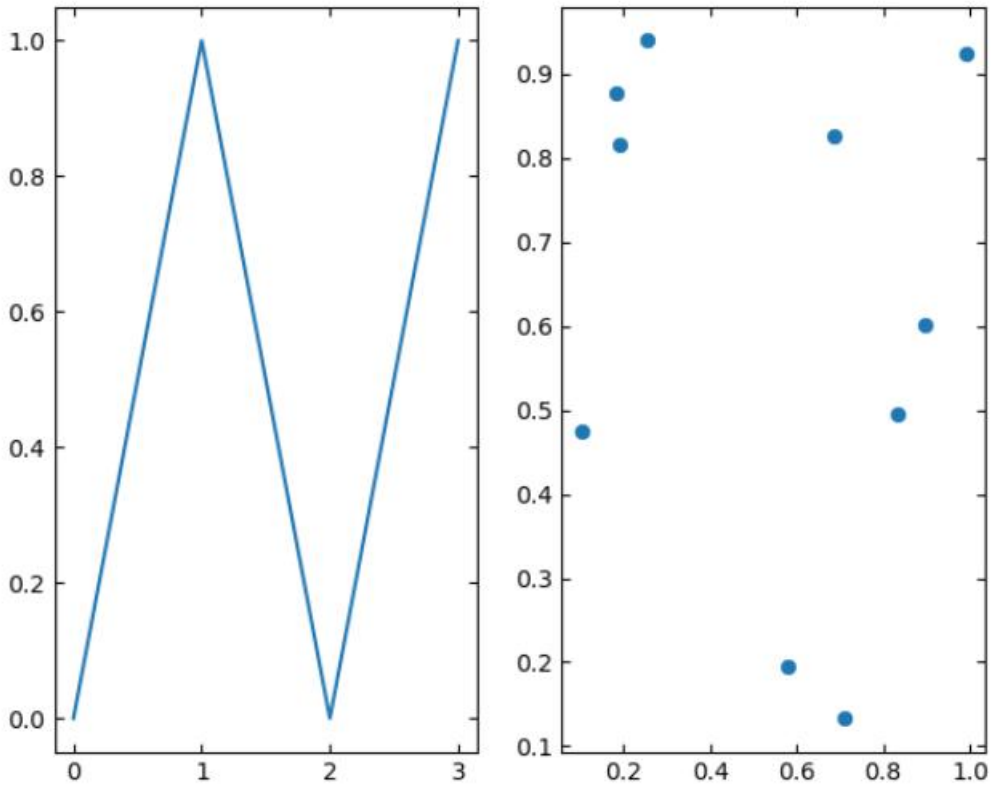


(3). 设置多个坐标区的保留状态

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。通过将 ax1 传递给 plot 函数来创建左部绘图。重复该过程以创建右部绘图。在每个坐标区中绘制图形。

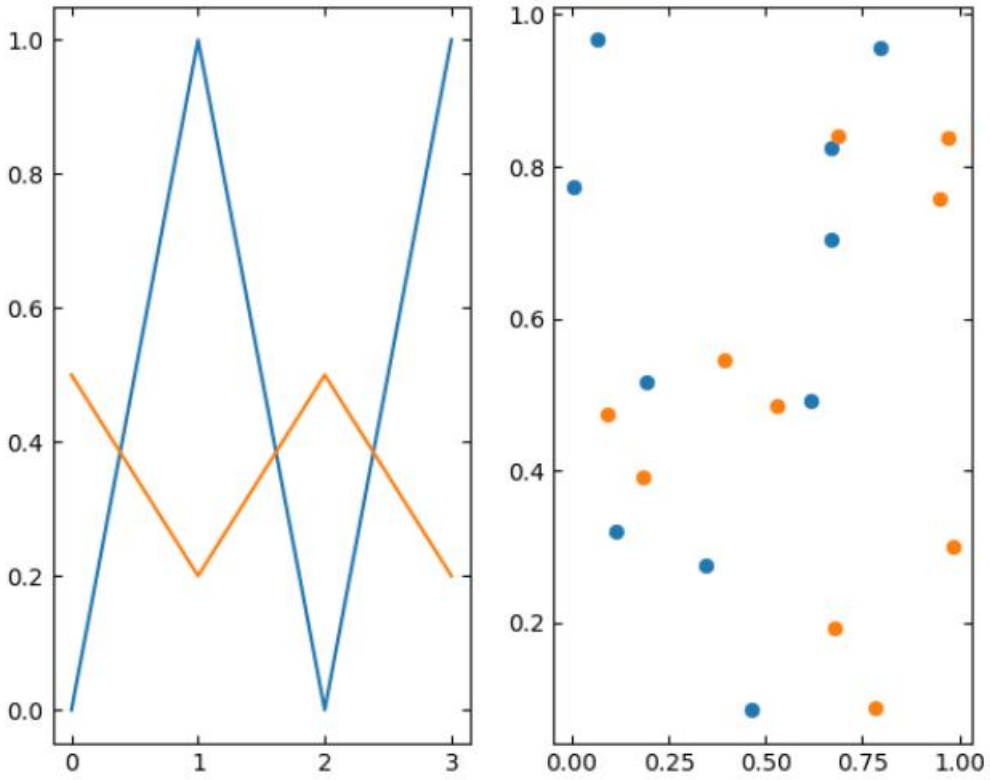
```
ax1 = subplot(1,2,1)
ax2 = subplot(1,2,2)

plot(ax1,[0,1,0,1])
scatter(ax2, rand(1,10),rand(1,10),filled=true)
```



将两个坐标区的 hold 状态都设置为 "on"。然后在每个图中显示附加数据。

```
hold([ax1, ax2], "on")
plot(ax1,[.5, .2, .5, .2])
scatter(ax2,rand(1,10),rand(1,10),filled=true)
```



yyaxis

创建具有两个 y 轴的图

A. 语法

```
yyaxis("left")
yyaxis("right")

___ = yyaxis(ax,___)
```

B. 说明

yyaxis("left") 激活当前坐标区中与左侧 y 轴关联的一侧。后续图形命令的目标为左侧。如果当前坐标区中没有两个 y 轴，此命令将添加第二个 y 轴。如果没有坐标区，此命令将首先创建坐标区。

yyaxis("right") 激活当前坐标区中与右侧 y 轴关联的一侧。后续图形命令的目标为右侧。

yyaxis(ax,___) 指定 ax 坐标区（而不是当前坐标区）的活动侧。如果坐标区中没有两个 y 轴，此命令将添加第二个 y 轴。指定坐标区作为第一个输入参数。使用双引号将 "left" 和 "right" 引起来。

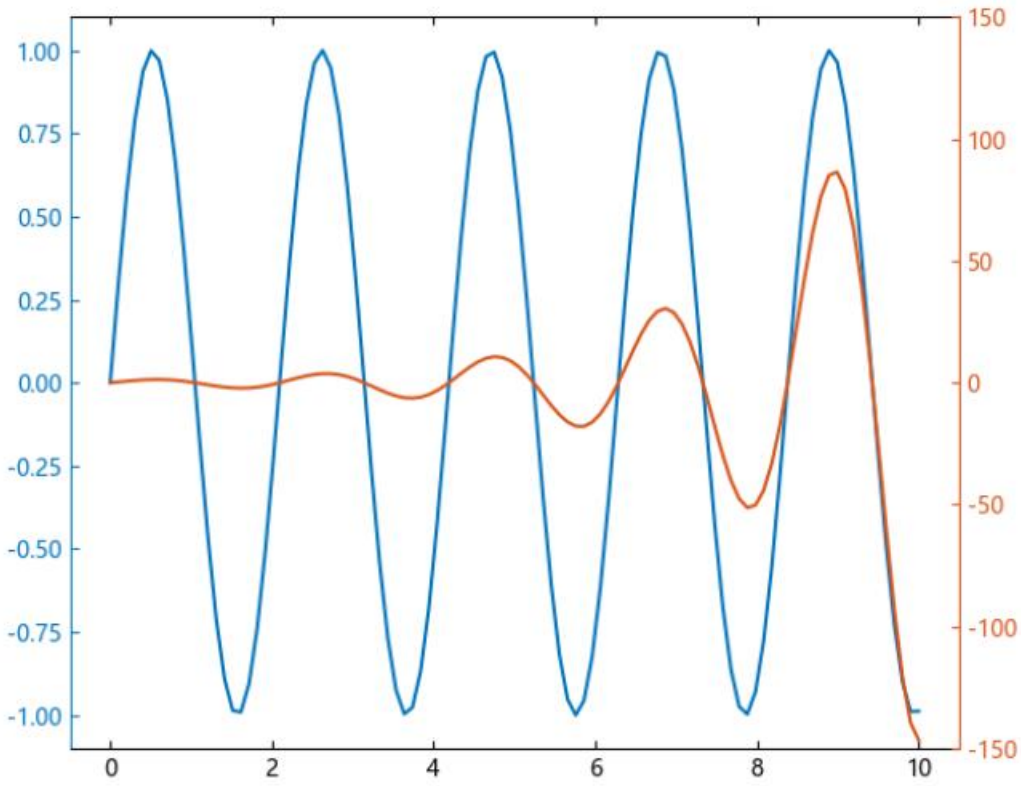
C. 示例

(1). 绘制带有两个 y 轴的数据图

创建左右两侧都有 y 轴的坐标区。基于左侧 y 轴绘制一组数据的图。然后使用 yyaxis("right") 激活右侧，使后续图形函数作用于该侧。基于右侧 y 轴绘制第二组数据的图，并为右侧 y 轴设置范围。

```
x = LinRange(0, 10, 100);
y = sin.(3 .* x);
yyaxis("left");
plot(x, y)

z = sin.(3 * x) .* exp.(0.5 * x);
yyaxis("right");
plot(x, z)
ylim([-150 150])
```

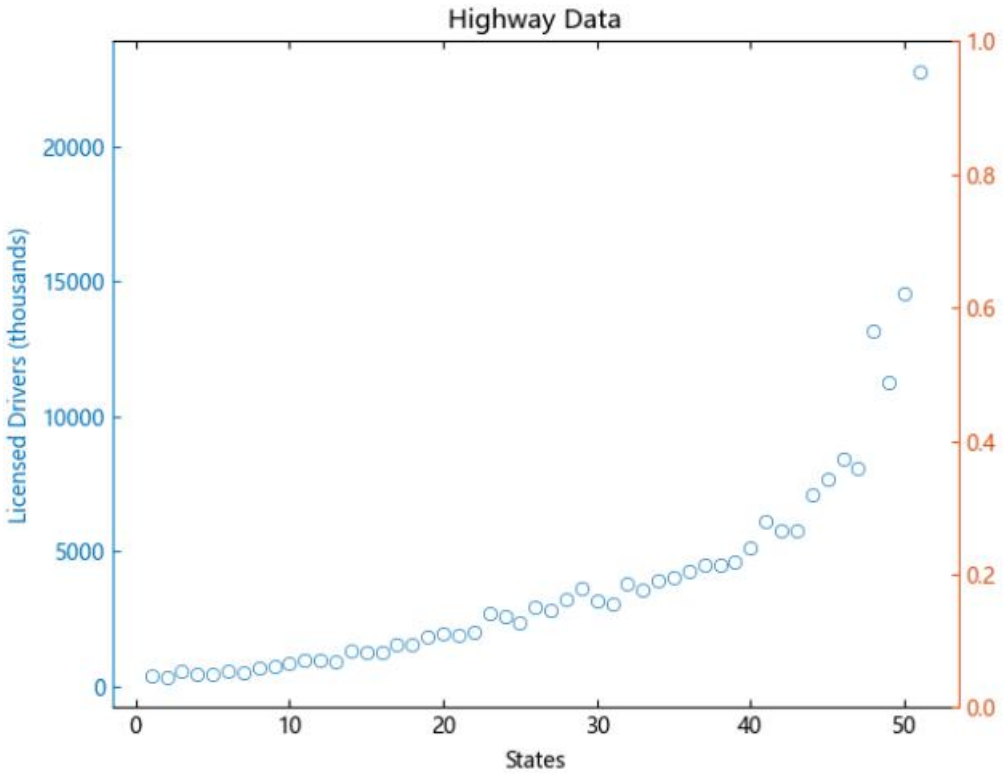


(2). 为每一侧添加标题和轴标签

导入 "0-data.jl" 文件,创建具有两个 y 轴的图，并为每一侧添加标题和轴标签。

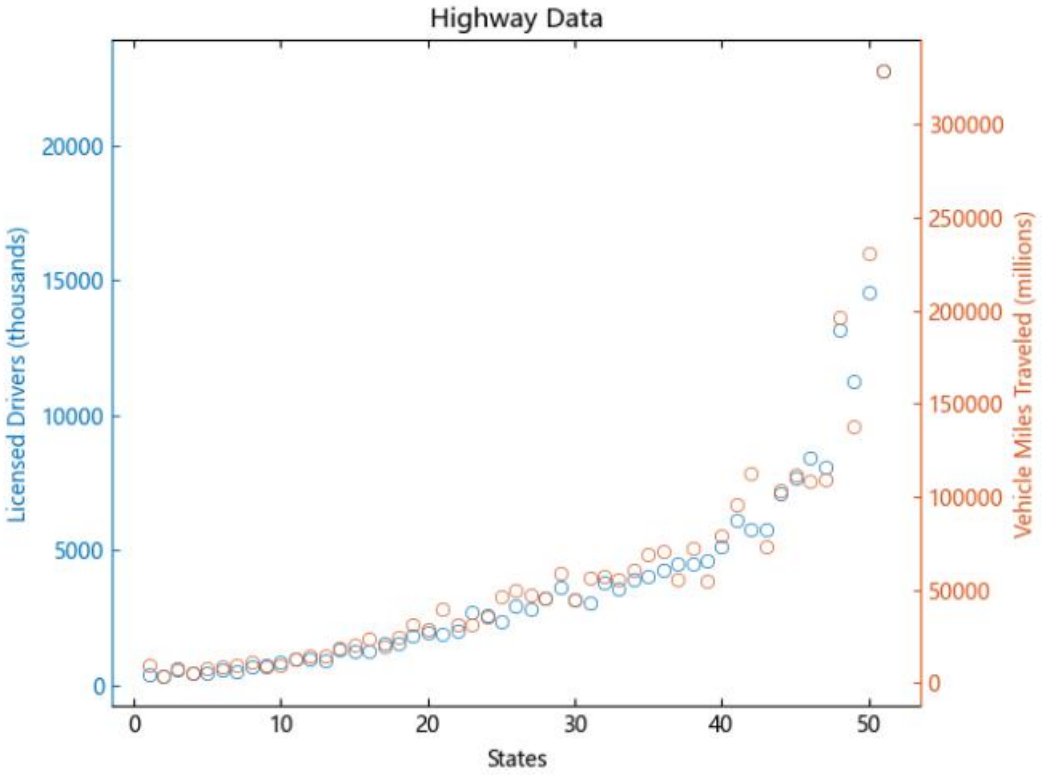
从示例文件 "0-data.jl" 加载矩阵 hwydata。基于左侧 y 轴和 hwydata 中的第五列数据创建一个散点图。添加标题和轴标签。

```
using TyPlot: TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/格式和注释/yyaxis/0-data.jl"
include(source_path)
ind = 1:51;
drivers = hwydata[:, 5];
yyaxis("left")
scatter(ind, drivers)
title("Highway Data")
xlabel("States")
ylabel("Licensed Drivers (thousands)")
```



基于右侧 y 轴和 hwydata 中的第七列数据创建第二个散点图。然后为右侧 y 轴添加标签。

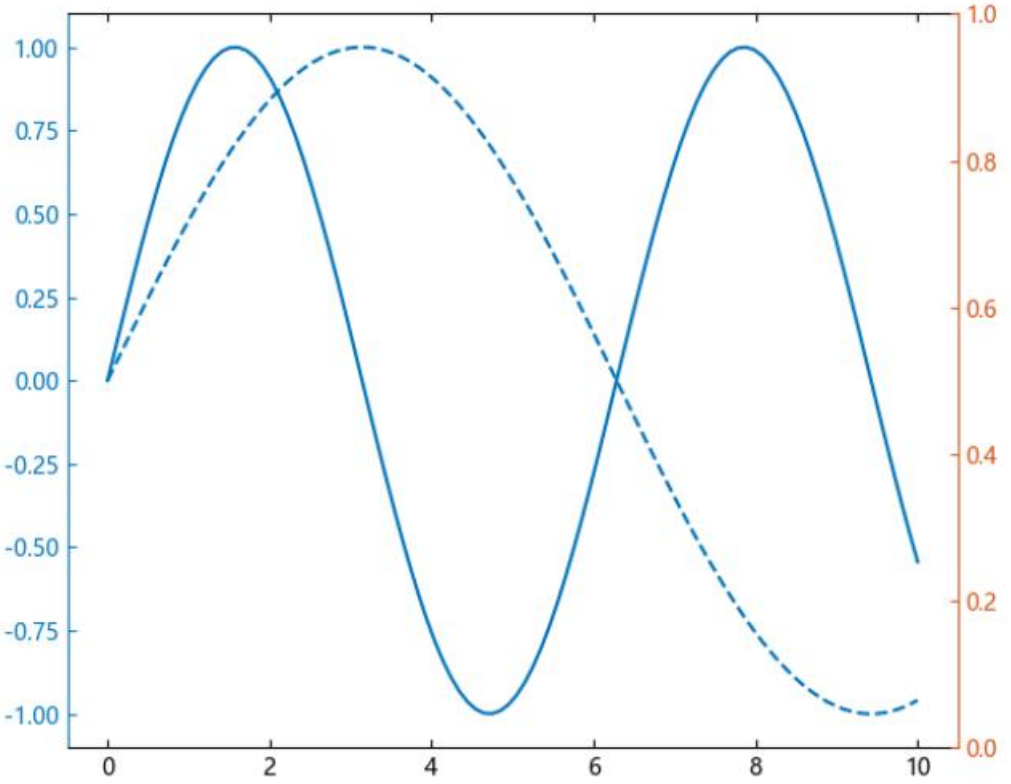
```
pop = hwydata[:, 7];
yyaxis("right")
scatter(ind, pop)
ylabel("Vehicle Miles Traveled (millions)")
tightlayout()
```



(3). 在每一侧绘制多组数据

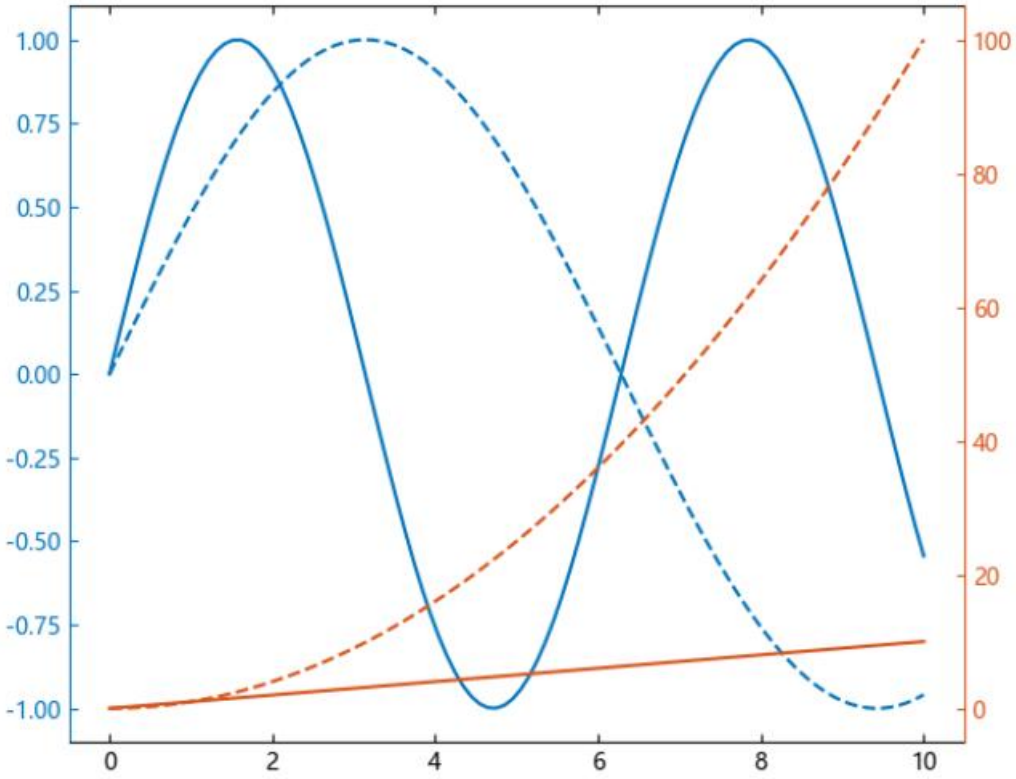
使用 hold("on") 命令基于左侧 y 轴绘制两个线条。

```
x = LinRange(0, 10, 100);
yl1 = sin.(x);
yl2 = sin.(x ./ 2);
yyaxis("left")
plot(x, yl1)
hold("on")
plot(x, yl2)
hold("off")
```



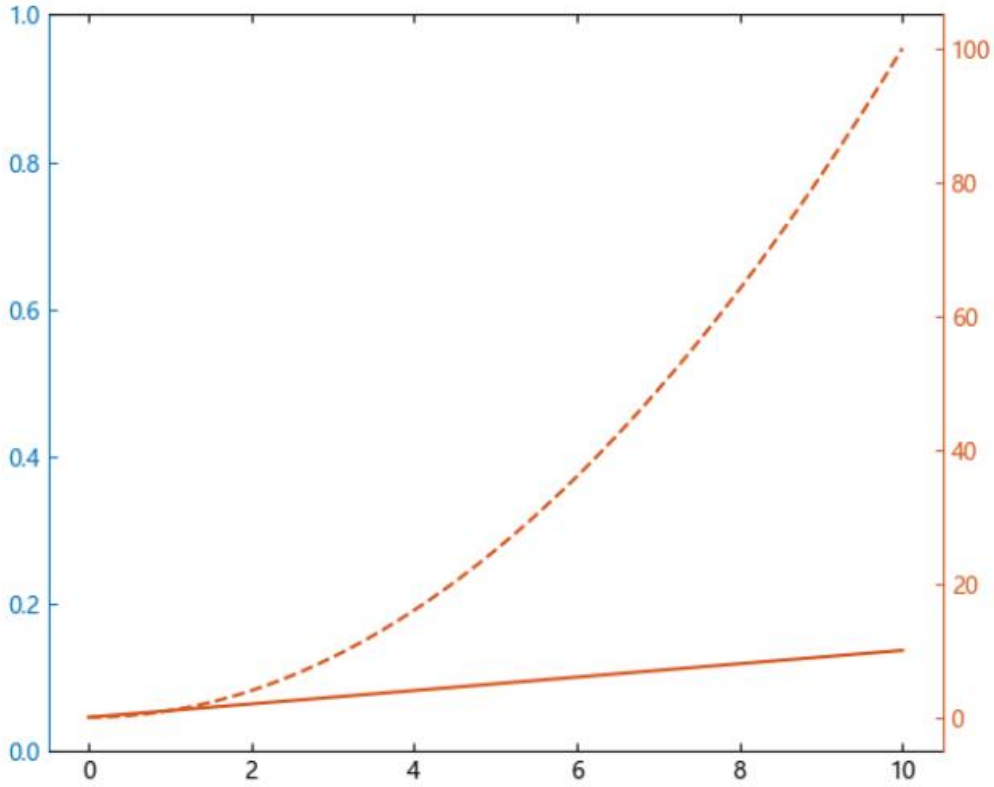
基于右侧 y 轴绘制两个线条。hold 命令同时影响左侧和右侧 y 轴，因此您不需要再次发出此命令。完成绘图后，关闭 hold。

```
yr1 = x;
yr2 = x .^ 2;
yyaxis("right")
hold("on")
plot(x, yr1)
plot(x, yr2)
hold("off")
```



通过激活左侧并使用 cla 命令来清除左侧。

```
yyaxis("left");
cla()
```



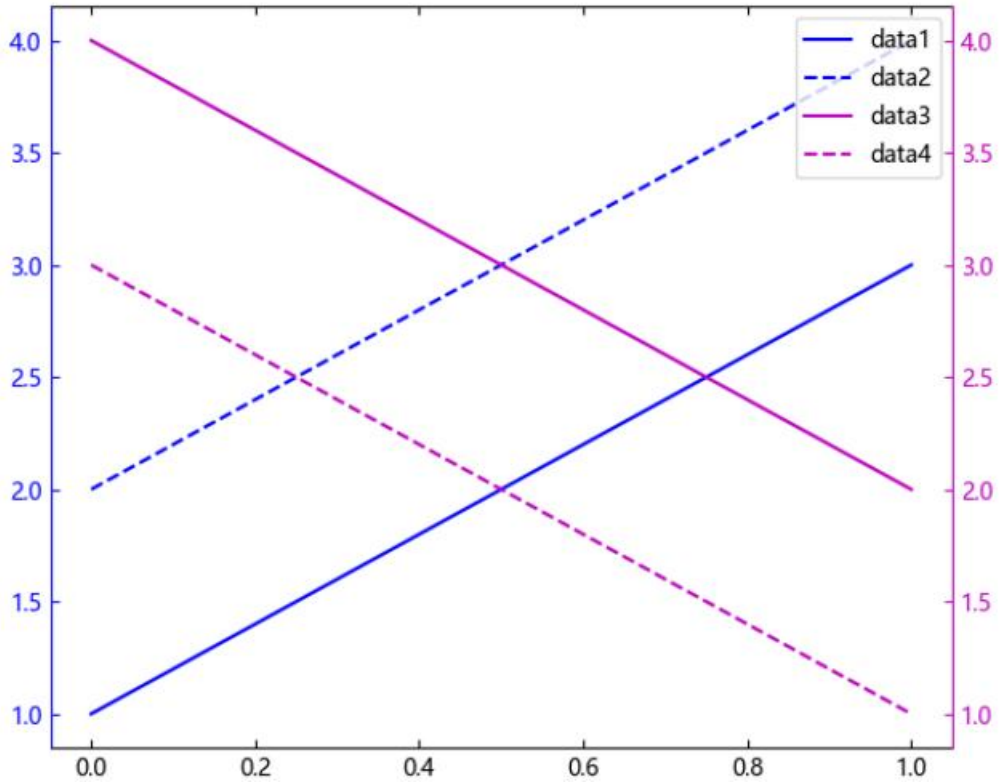
(4). 控制每侧的颜色

通过将色序设置为您想使用的两种颜色，为坐标区的每侧指定颜色方案。使用 colororder 函数设置色序。然后，基于左侧 y 轴和右侧 y 轴分别绘制两个线条。添加一个图例。

```
colororder(["b","m"])

yyaxis("left");
y = [1 2; 3 4];
plot(y)

yyaxis("right");
z = [4 3; 2 1];
plot(z)
legend()
```



(5). 控制单个绘图的颜色

通过设置坐标区每侧的色序来控制各个绘图颜色。

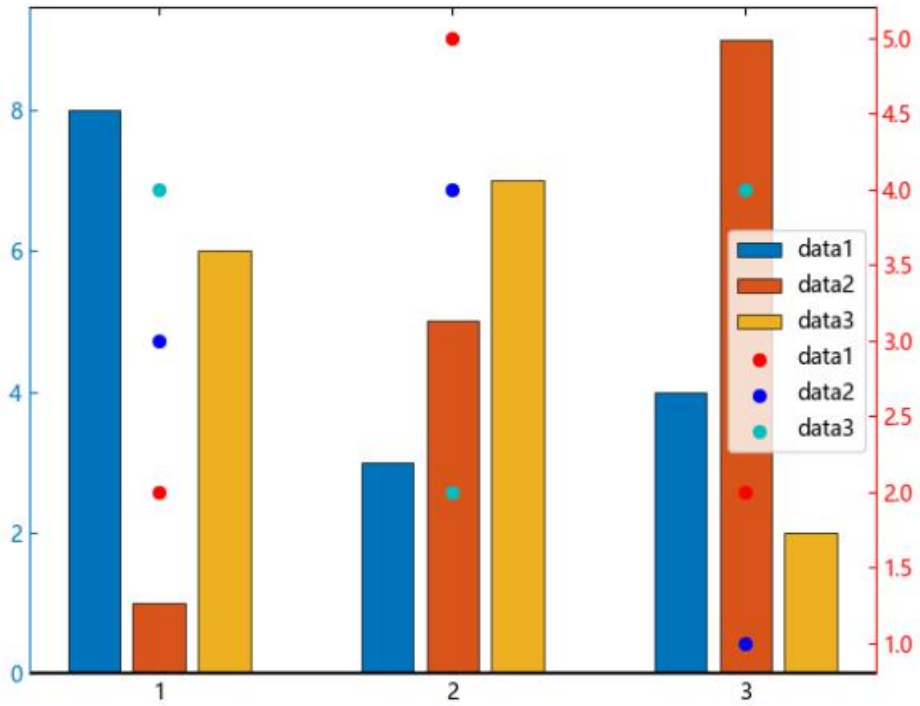
基于左侧坐标轴绘制三组条形图。通过将左侧的色序设置为默认色序，为每个条形序列设置不同的颜色。

```
yyaxis("left")
bar([8 1 6; 3 5 7; 4 9 2]);
```

```
colororder("default")
```

基于右侧坐标轴绘制三个散点图。通过将色序设置为一个包含颜色名称的数组，为每个散点图使用不同颜色。您也可以使用 RGB 三元组矩阵指定颜色。然后，添加一个图例。

```
yyaxis("right")
scatter([1 2 3],[2 5 2], filled = true)
hold("on")
scatter([1 2 3],[3 4 1], filled = true)
scatter([1 2 3],[4 2 4], filled = true)
hold("off")
colororder(["r","b","c"])
legend()
```



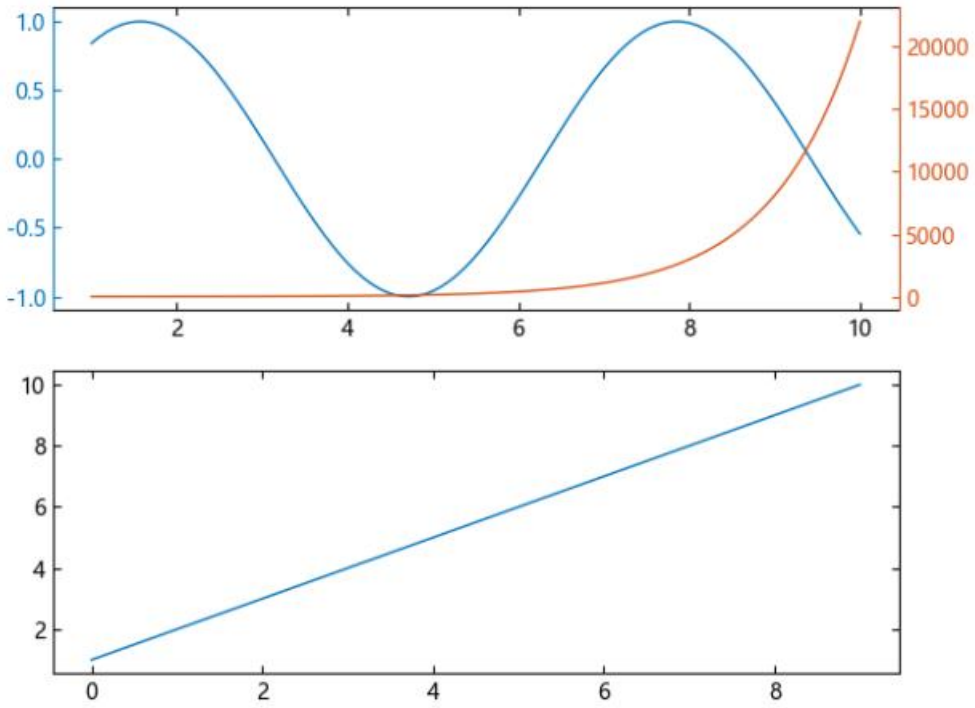
(6). 为特定坐标区添加第二个 y 轴

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。通过指定 ax1 作为 yyaxis 的第一个输入，在顶部坐标区中添加第二个 y 轴。如果不指定坐标区，yyaxis 将向当前坐标区添加第二个 y 轴。

```
x = LinRange(1, 10, 100);
ax1 = subplot(2, 1, 1)
ax2 = subplot(2, 1, 2)
```

```
yyaxis(ax1, "left")
plot(ax1, x, sin.(x))
yyaxis(ax1, "right")
plot(ax1, x, exp.(x))

plot(ax2, 1:10)
```



plotyy

创建具有两个 y 轴的图形

A. 注意

此函数未提供。该函数可以采用 `yyaxis` 实现同等功能

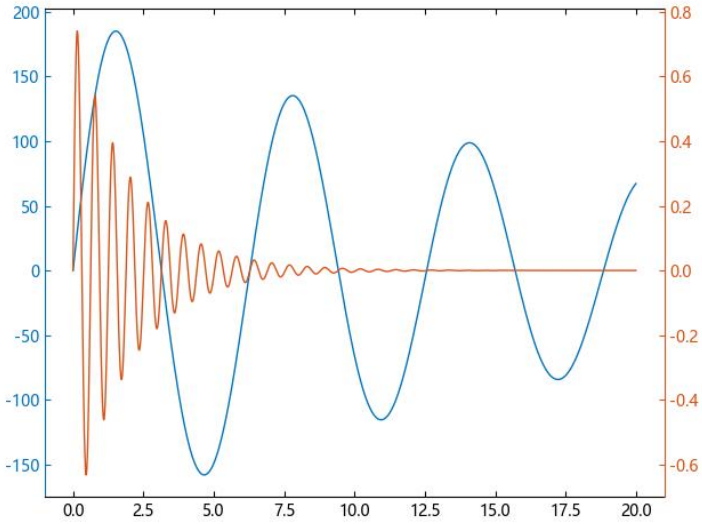
B. 示例

(1). 使用不同 y 轴绘制两个数据集

使用两个 y 轴在一个图上绘制两个数据集。

```
using TyPlot
x = 0:0.01:20
y1 = @. 200exp(-0.05x) * sin(x)
y2 = @. 0.8exp(-0.5x) * sin(10x)
figure()

yyaxis("left")
plot(x,y1)
yyaxis("right")
plot(x,y2)
```



(2). 添加标题和轴标签

使用两个 y 轴在一个图上绘制两个数据集。添加标题和轴标签。

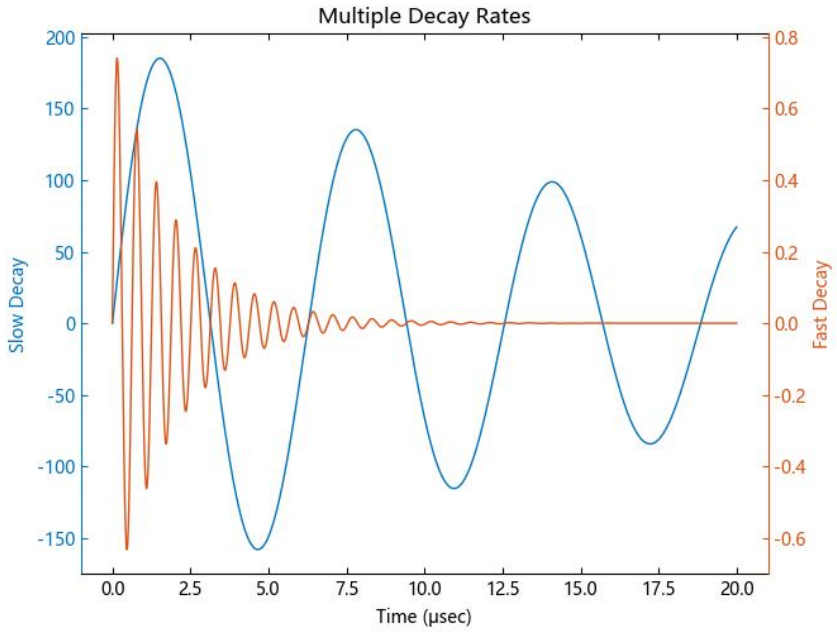
```
using TyPlot
x = 0:0.01:20
y1 = @. 200exp(-0.05x) * sin(x)
y2 = @. 0.8exp(-0.5x) * sin(10x)
figure()

yyaxis("left")
plot(x, y1)
ylabel("Slow Decay")

yyaxis("right")
plot(x, y2)
ylabel("Fast Decay")

title("Multiple Decay Rates")
```

```
xlabel("Time (μsec)")
```



(3). 更改线型

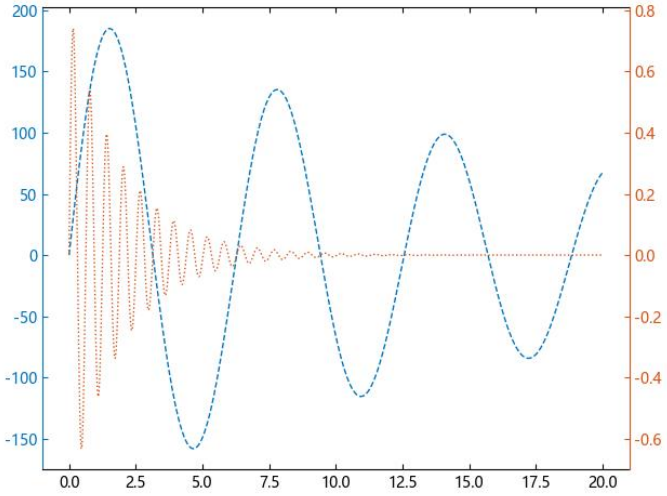
使用两个 y 轴在一个图上绘制两个数据集。更改线型。

```
using TyPlot
x = 0:0.01:20
y1 = @. 200exp(-0.05x) * sin(x)
y2 = @. 0.8exp(-0.5x) * sin(10x)
figure()

yyaxis("left")
hLine1 = plot(x, y1)
yyaxis("right")
hLine2 = plot(x, y2)

hLine1[1].set_linestyle("--")
hLine2[1].set_linestyle(":")

plt_update()
```



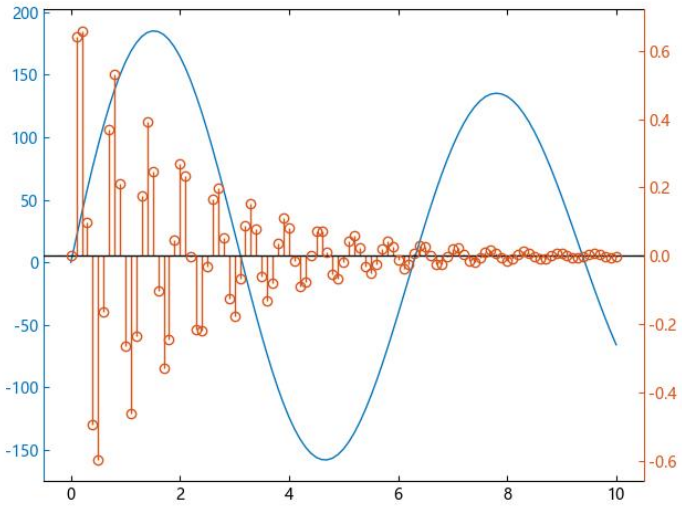
(4). 组合不同类型的绘图

使用两个 y 轴在一个图上绘制两个数据集。对与左 y 轴关联的数据使用线图。对与右 y 轴关联的数据使用针状图。

```
using TyPlot
x = 0:0.1:10
y1 = @. 200exp(-0.05x) * sin(x)
y2 = @. 0.8exp(-0.5x) * sin(10x)
figure()

yyaxis("left")
plot(x, y1)

yyaxis("right")
stem(x, y2)
```



(5). 对两个数据集使用右 y 轴

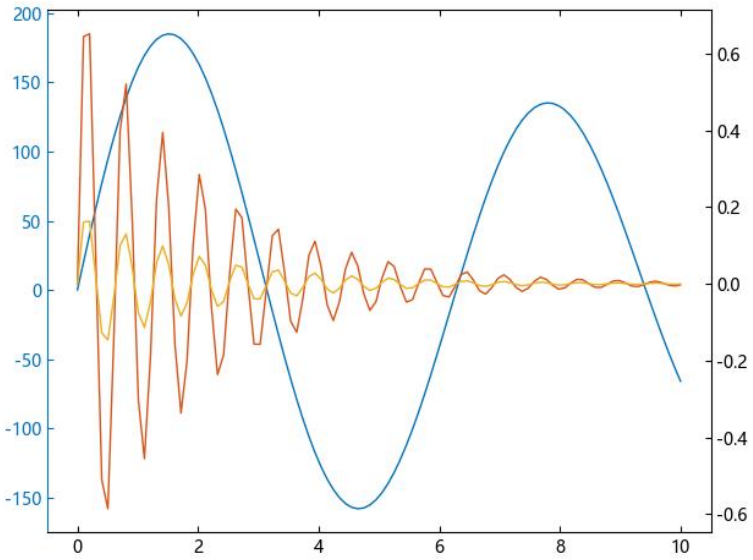
使用两个 y 轴在一个图表上绘制三个数据集。绘制一个与左 y 轴关联的数据集。使用包含两列的矩阵绘制与右 y 轴关联的两个数据集。

```
using TyPlot
x = LinRange(0, 10, 100)

y1 = @. 200exp(-0.05x) * sin(x)
y2 = @. 0.8exp(-0.5x) * sin(10x)
y3 = @. 0.2exp(-0.5x) * sin(10x)

figure()
colororder(["#0072BD", "black"])
yyaxis("left")
plot(x, y1)

yyaxis("right")
hold("on")
h = plot([x x], [y2 y3])
newcolors = [(0.85, 0.33, 0.10), (0.93, 0.69, 0.13)]
colororder(newcolors)
hold("off")
```



legend

详情见图形 API->格式和注释->标签和注释->标签->legend

colororder

为可视化多个数据序列设置色序

A. 语法

```
colororder(newcolors)
colororder(target,newcolors)

C = colororder()
C = colororder(target)
```

B. 说明

colororder(newcolors)为当前图窗设置色序。如果图窗不存在，Syslab 会创建一个图窗并为该图窗设置色序。在您设置图窗的色序时，会为该图窗中的所有坐标区设置色序。

colororder(target,newcolors)设置目标坐标区、图窗或图的色序，而不是当前图窗的色序。

C = colororder() 返回当前图窗的色序矩阵。

C = colororder(target)返回目标图窗、坐标区的色序矩阵。

C. 示例

(1). 在绘图前设置颜色

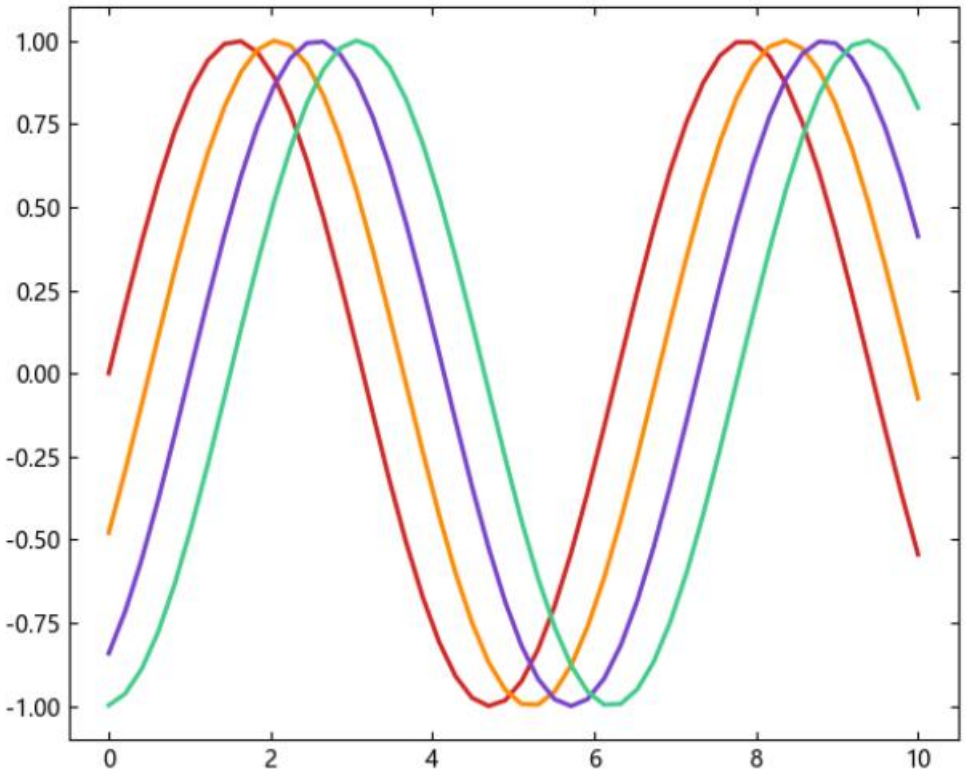
将图窗的色序设置为四种颜色。定义一个 x 坐标向量和四个 y 坐标向量。然后绘制每组坐标。

```
newcolors = [0.83 0.14 0.14
             1.00 0.54 0.00
             0.47 0.25 0.80
             0.25 0.80 0.54];

colororder(newcolors)

x = LinRange(0, 10, 50);
y1 = sin.(x);
y2 = sin.(x .- 0.5);
y3 = sin.(x .- 1);
y4 = sin.(x .- 1.5);

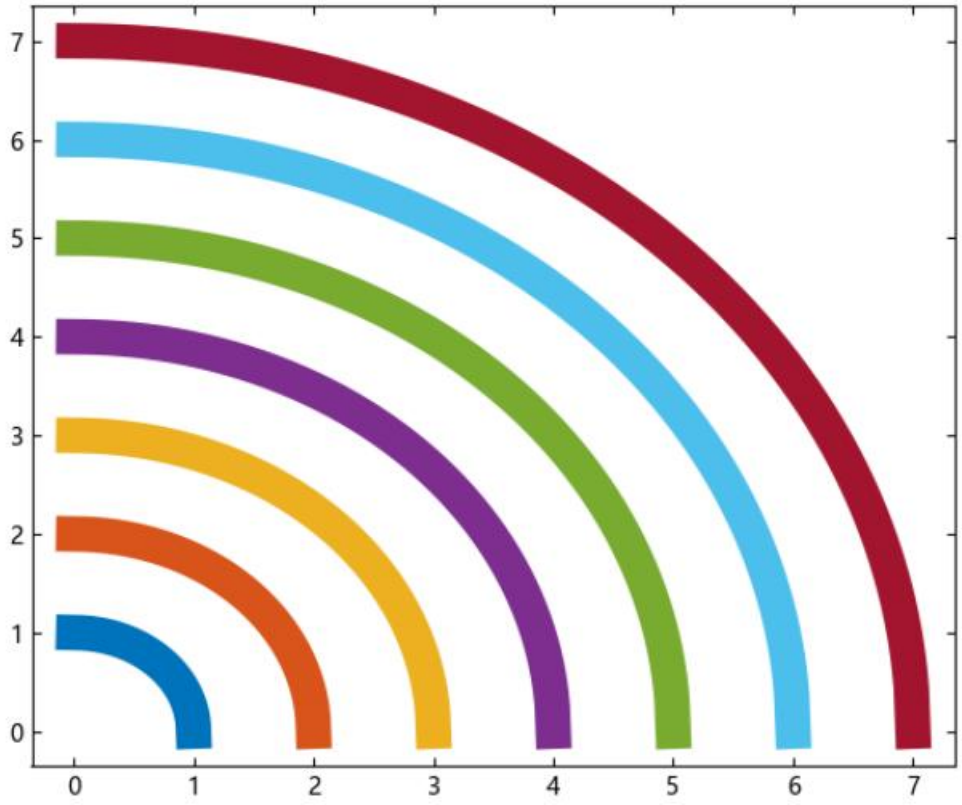
plot(x, y1; linewidth=2)
hold("on")
plot(x, y2; linewidth=2)
plot(x, y3; linewidth=2)
plot(x, y4; linewidth=2)
hold("off")
```



(2). 在绘图后设置颜色

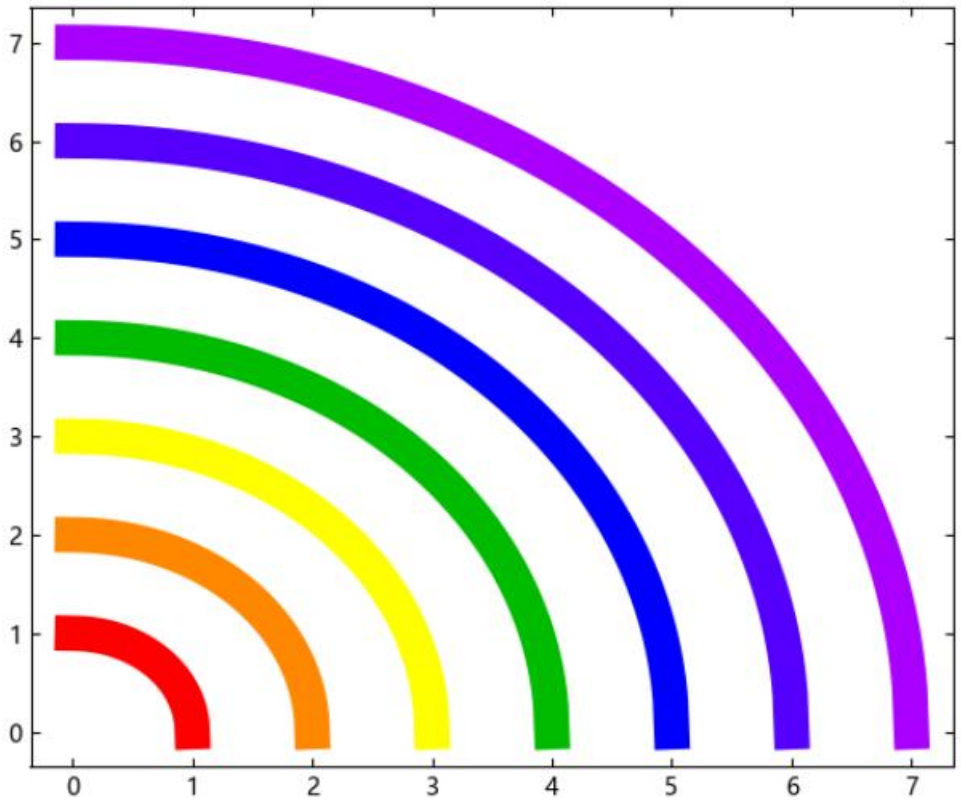
绘制七个同心四分之一圆。

```
hold("on")
for r in 1:7
    global x = LinRange(0, r, 500)
    global y = sqrt.(r .^ 2 .- x .^ 2)
    plot(x, y; linewidth=15)
end
hold("off")
```



将色序更改为七个十六进制颜色代码。

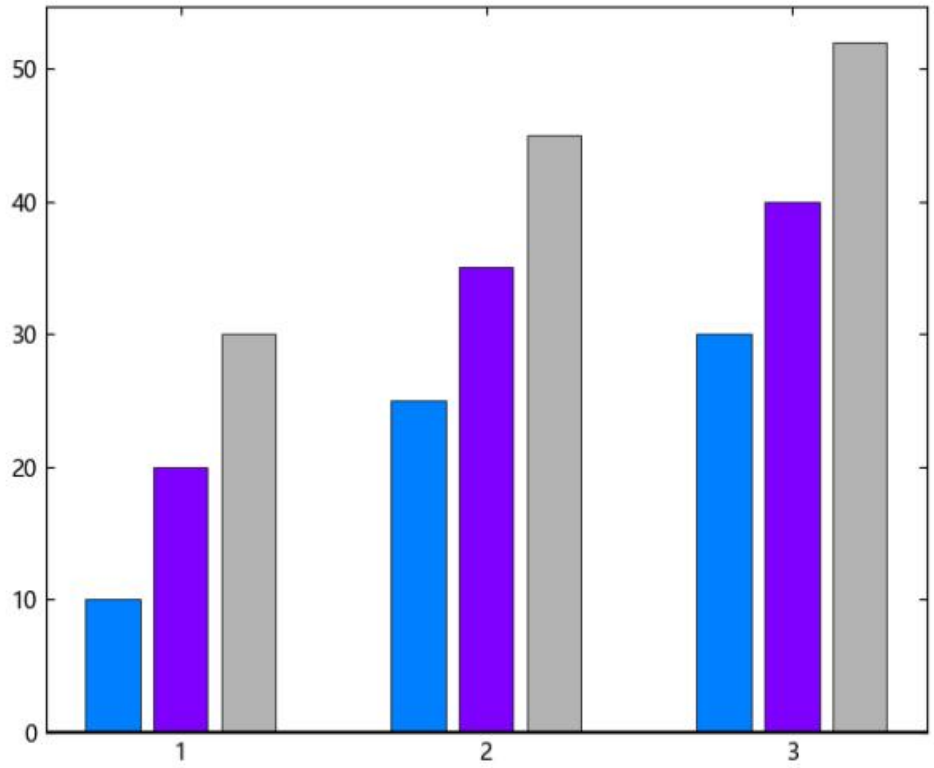
```
newcolors = ["#F00","#F80","#FF0","#0B0","#00F","#50F","#A0F"];
colororder(newcolors)
```



(3). 在条形图中设置颜色

显示三个条形序列。然后将色序设置为蓝色、紫色和灰色。

```
bar([10 20 30; 25 35 45; 30 40 52])
newcolors = [0 0.5 1; 0.5 0 1; 0.7 0.7 0.7];
colororder(newcolors)
```



(4). 为双 y 轴图的两侧设置颜色

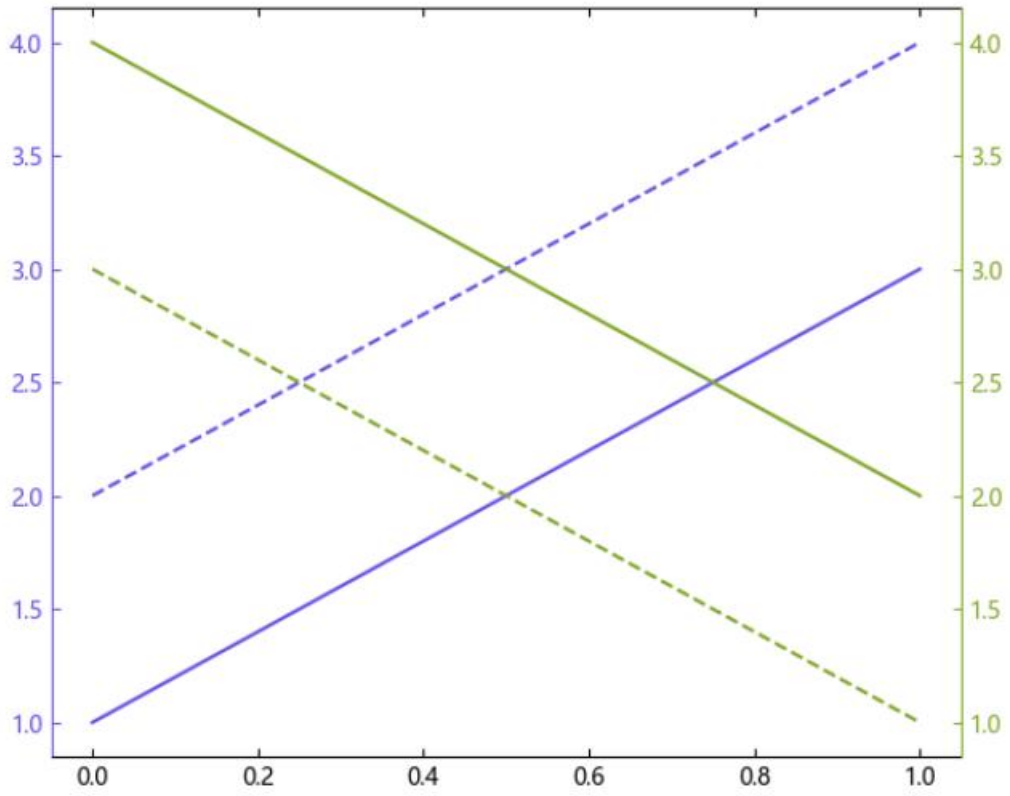
在调用 yyaxis 之前设置图窗的色序会为每个 y 轴设置颜色。左侧使用第一种颜色，右侧使用第二种颜色。如果指定两种以上的颜色，则任一侧都不会使用多余的颜色。

将 newcolors 定义为包含两个 RGB 三元组的矩阵。设置图窗的色序，并基于左侧绘制两条线。然后基于右侧绘制两条线。

```
newcolors = [0.40 0.30 0.90; 0.50 0.65 0.15];
colororder(newcolors)

# Left side
yyaxis("left");
plot([1 2; 3 4])

# Right side
yyaxis("right");
plot([4 3; 2 1])
```

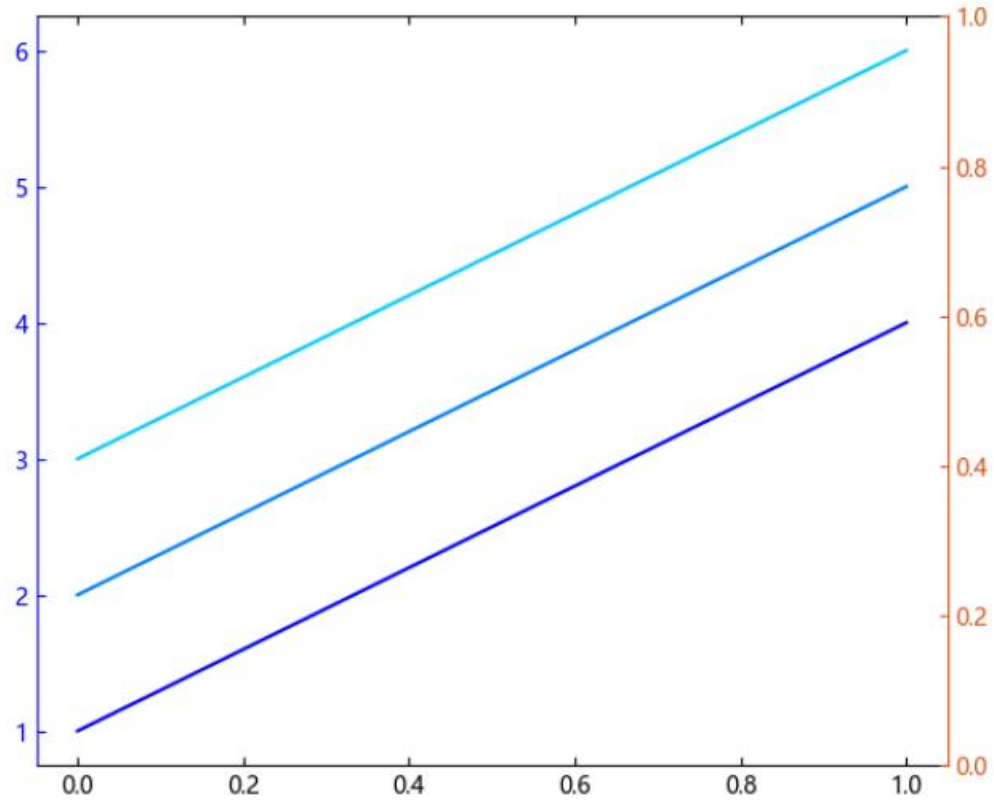


(5). 为双 y 轴图的活动侧设置颜色

在调用 yyaxis 之后设置图窗的色序会为活动侧设置颜色。

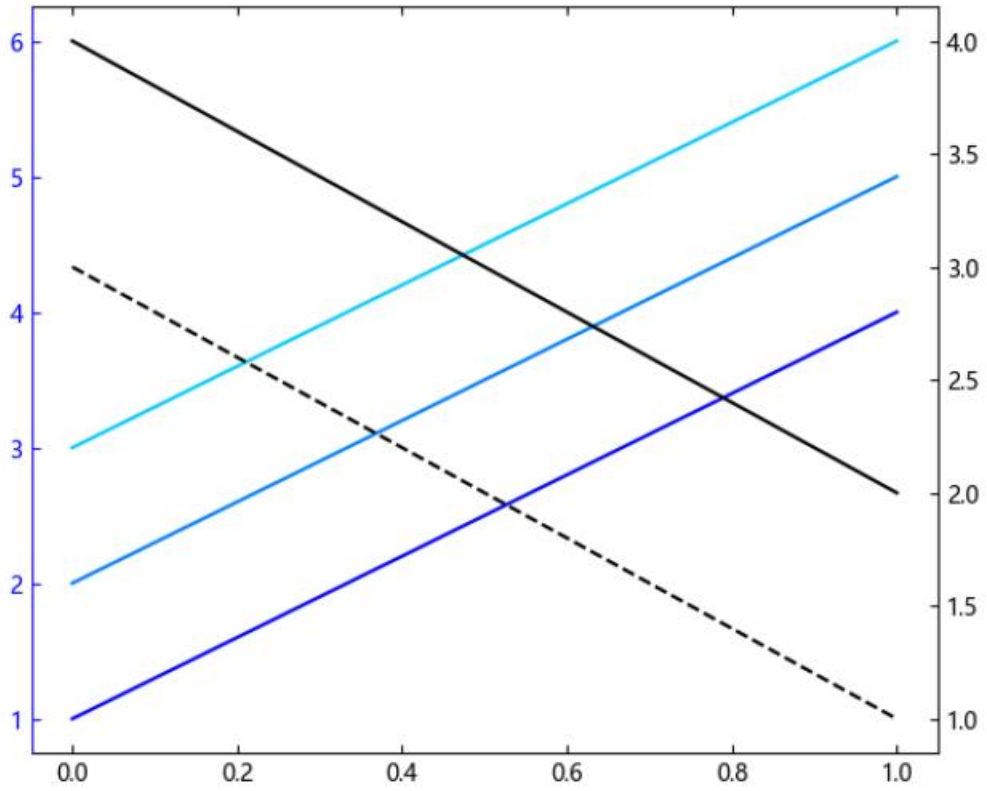
激活左侧 y 轴并绘制三条线。将线型序列设置为实线，并将左侧 y 轴颜色更改为蓝色。然后将色序设置为三个蓝色梯度。

```
yyaxis("left");
plot([1 2 3; 4 5 6])
ax = gca();
ax.spines["left"].set_color("blue");
ax.tick_params(which = "both", axis = "y", colors = "blue");
leftcolors = [0 0 1; 0 0.50 1; 0 0.80 1];
colororder(leftcolors)
```



激活右侧 y 轴并绘制两条线。将右侧 y 轴颜色更改为黑色。然后将色序设置为黑色。

```
yyaxis("right");
plot([4 3; 2 1])
ax = gca();
ax.spines["right"].set_color("black");
ax.tick_params(which = "both", axis = "y", colors = "black");
colororder([0 0 0])
```

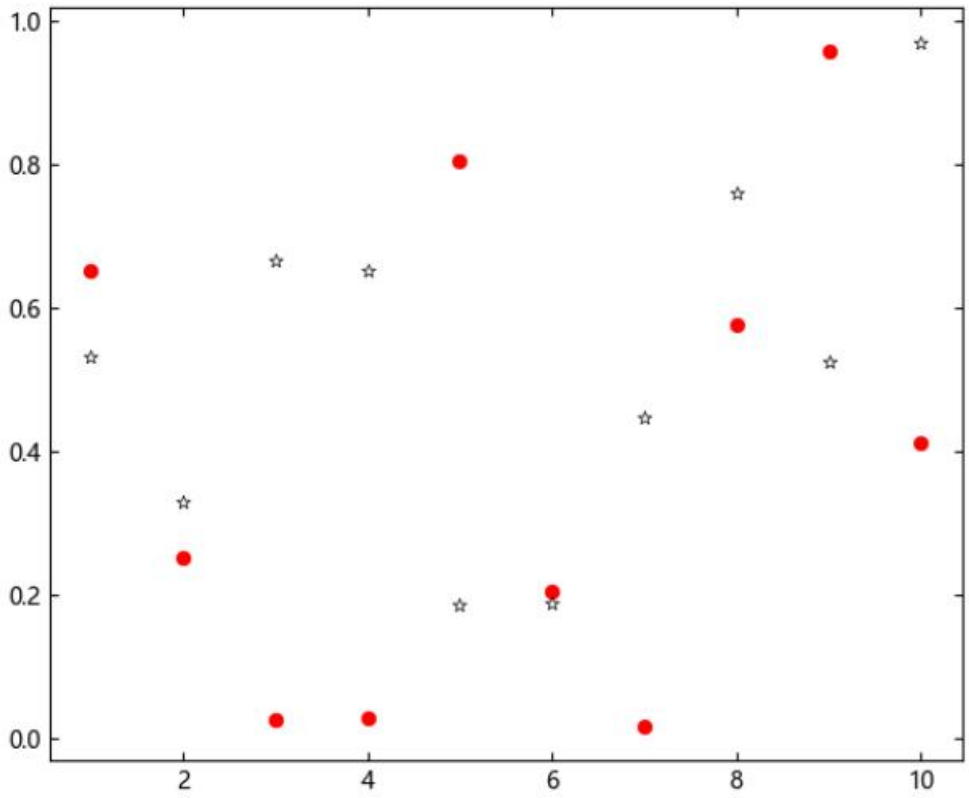


(6). 从色序中排除数据序列

当使用颜色参数调用绘图函数时，绘图函数会使用该颜色，而不是使用色序中的下一种颜色。

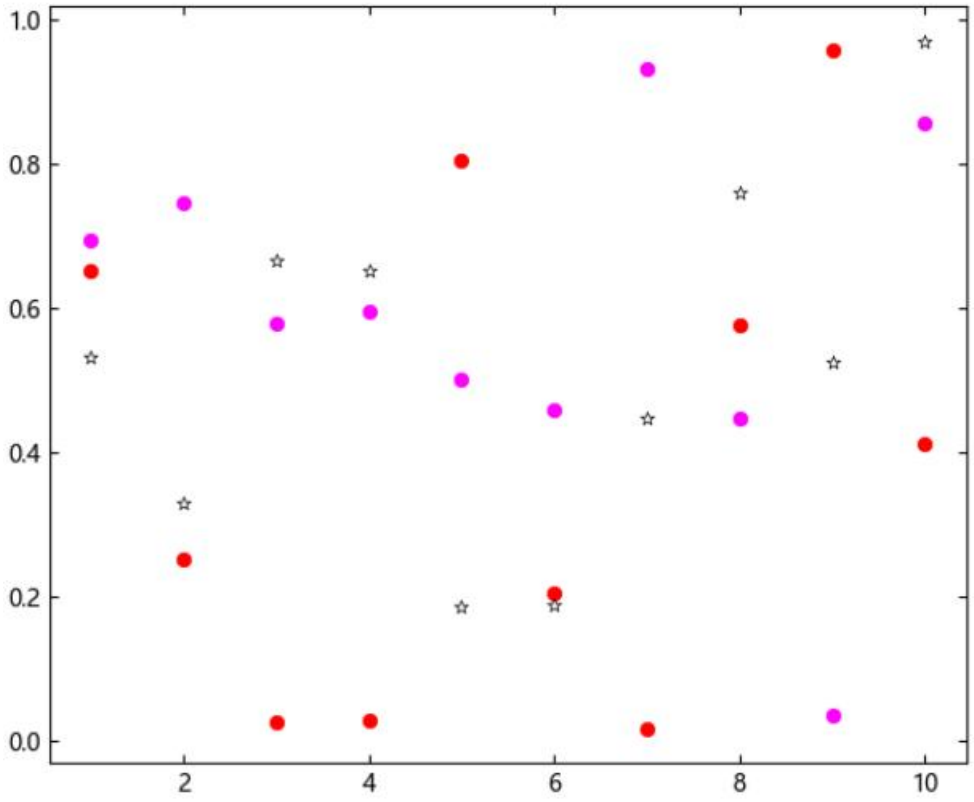
将图窗的色序设置为红色、品红色和蓝色。调用 scatter 函数以绘制一组散点。然后绘制第二组散点，并将标记指定为黑色星号。

```
newcolors = ["red","magenta","blue"];
colororder(newcolors)
scatter(1:10,rand(1,10),filled = true )
hold("on")
scatter(1:10,rand(1,10),marker = "*",color = "k")
```



绘制第三组散点，不指定标记颜色。请注意，这组散点使用色序中的第二个颜色，即红色。

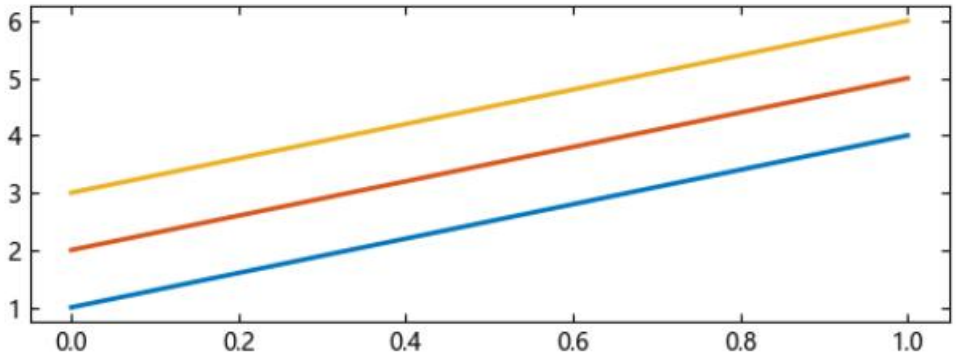
```
scatter(1:10,rand(1,10),filled = true)
hold("off")
```



(7). 修改指定坐标区的色序

创建分块图布局，并在第一个图块中绘制三条线。

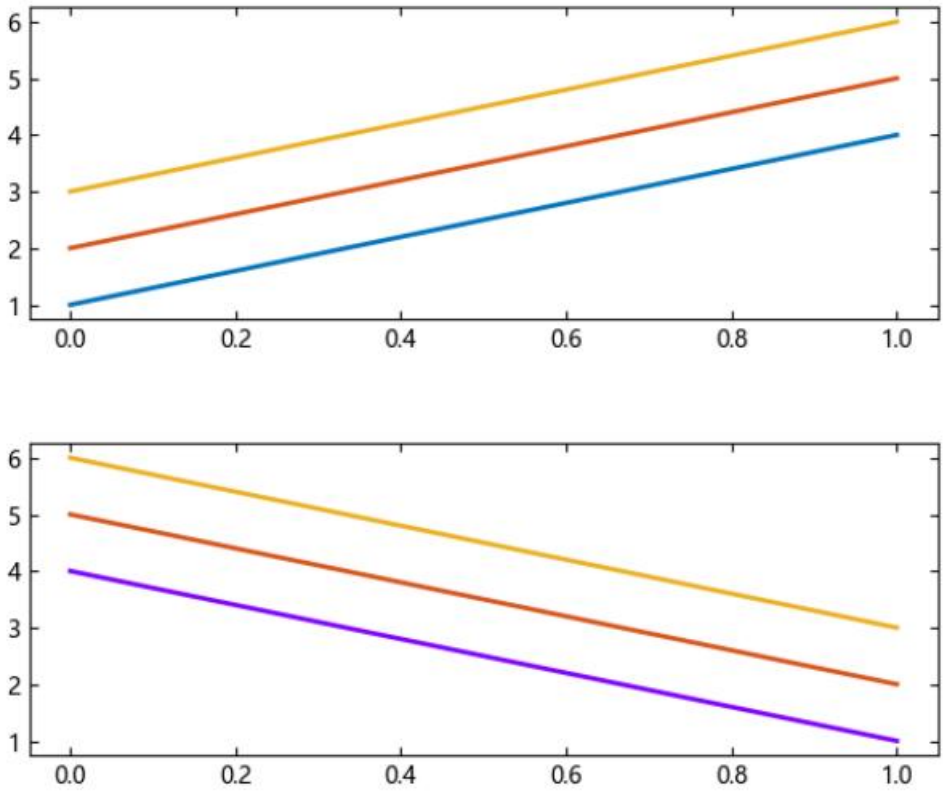
```
ax1 = subplot(2,1,1)
plot([1 2 3; 4 5 6],linewidth = 2)
```



带返回参数调用 subplot 函数，以获取第二个图块的坐标区对象。在第二个图块中绘制三条线。然后获取坐标区的色序矩阵，并在 C 中返回输出。将 C 中的第一种颜色更改为紫色，并将坐标区色序设置为修改后的 C 矩阵。

```
ax = subplot(2,1,2)
plot(ax,[4 5 6; 1 2 3],linewidth = 2)
C = colororder(ax);
C[1] = (0.5, 0, 1);
```

```
colororder(ax,C)
```



subplot

在平铺位置创建坐标区

A. 语法

```
subplot(m,n,p)

subplot(___,Key=Value)
ax = subplot()

subplot(ax)
```

B. 说明

subplot(m,n,p)将当前图窗划分为 m×n 网格，并在 p 指定的位置创建坐标区。Syslab 按行号对子图位置进行编号。第一个子图是第一行的第一列，第二个子图是第一行的第二列，依此类推。如果指定的位置已存在坐标区，则此命令会将该坐标区设为当前坐标区。

subplot(___,Key=Value) 使用一个或多个名称-值对组参数修改坐标区属性。在所有其他输入参数之后设置坐标区属性。

ax=subplot()创建一个 Axes 对象、PolarAxes 对象。以后可以使用 ax 修改坐标区。

subplot(ax)将 ax 指定的坐标区设为父图窗的当前坐标区。如果父图窗尚不是当前图窗，此选项不会使父图窗成为当前图窗。

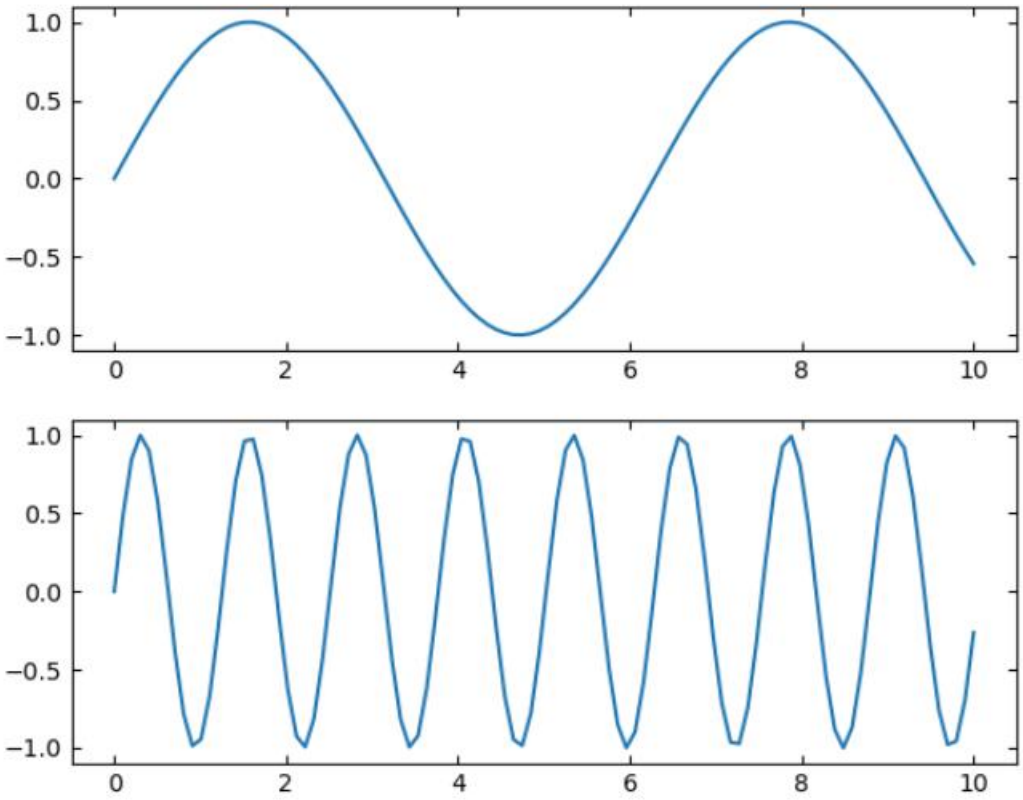
C. 示例

(8). 上部和下部子图

创建带有两个堆叠子图的图窗。在每个子图上绘制一条正弦波。

```
subplot(2, 1, 1);
x = LinRange(0, 10, 100);
y1 = sin.(x);
plot(x, y1)

subplot(2, 1, 2);
y2 = sin.(5 * x);
plot(x, y2)
```



(9). 子图的象限

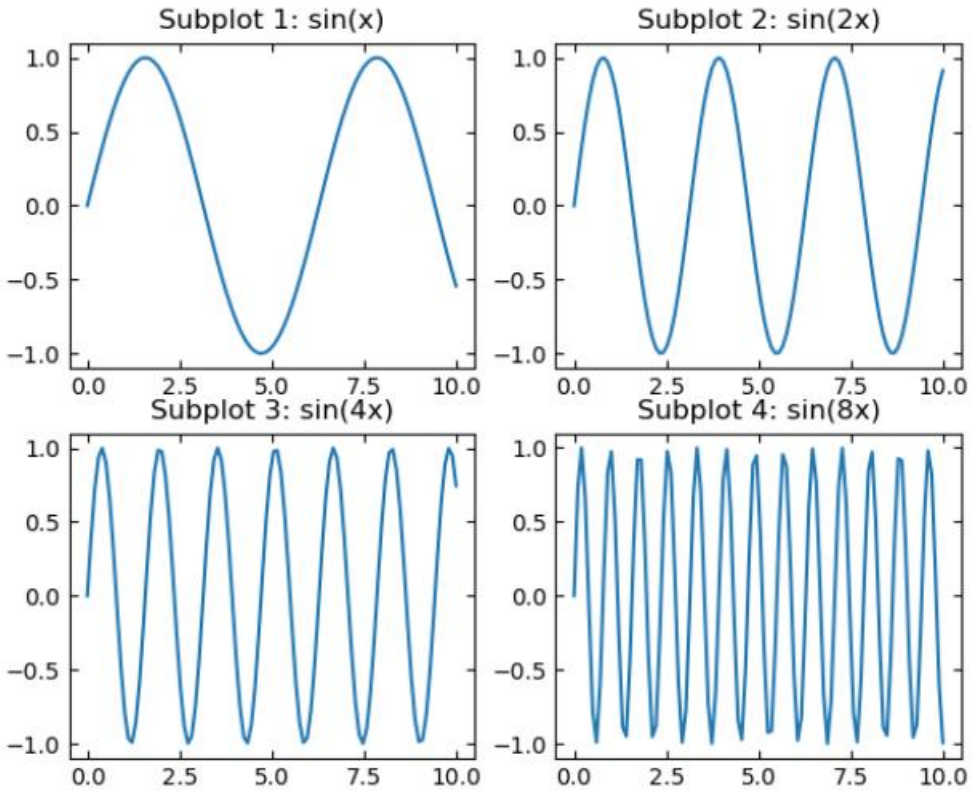
创建一个分为四个子图的图窗。在每个子图上绘制一条正弦波并为每个子图指定标题。

```
subplot(2, 2, 1)
x = LinRange(0, 10, 100);
y1 = sin.(x);
plot(x, y1)
title("Subplot 1: sin(x)")

subplot(2, 2, 2)
y2 = sin.(2 * x);
plot(x, y2)
title("Subplot 2: sin(2x)")

subplot(2, 2, 3)
y3 = sin.(4 * x);
plot(x, y3)
title("Subplot 3: sin(4x)")

subplot(2, 2, 4)
y4 = sin.(8 * x);
plot(x, y4)
title("Subplot 4: sin(8x)")
```



(10). 大小不同的子图

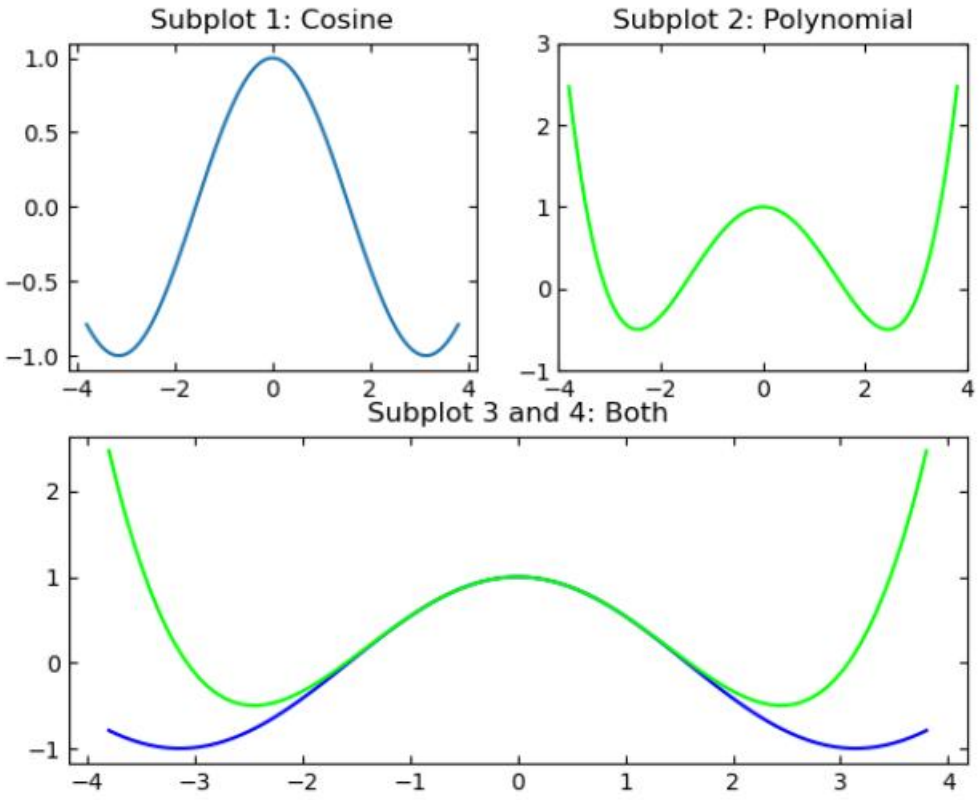
创建一个包含三个子图的图窗。在图窗的上半部分创建两个子图，并在图窗的下半部分创建第三个子图。在每个子图上添加标题。

```
x = LinRange(-3.8, 3.8, 100);
y_cos = cos.(x);
y_poly = 1 .- x .^ 2 ./ 2 .+ x .^ 4 ./ 24;

subplot(2,2,1);
plot(x,y_cos);
title("Subplot 1: Cosine")

subplot(2,2,2);
plot(x,y_poly,"#00FF00");
title("Subplot 2: Polynomial")
xlim([-4,4])
ylim([-1,3])

subplot(2,1,2);
plot(x,y_cos,"b",x,y_poly,"#00FF00");
title("Subplot 3 and 4: Both")
```



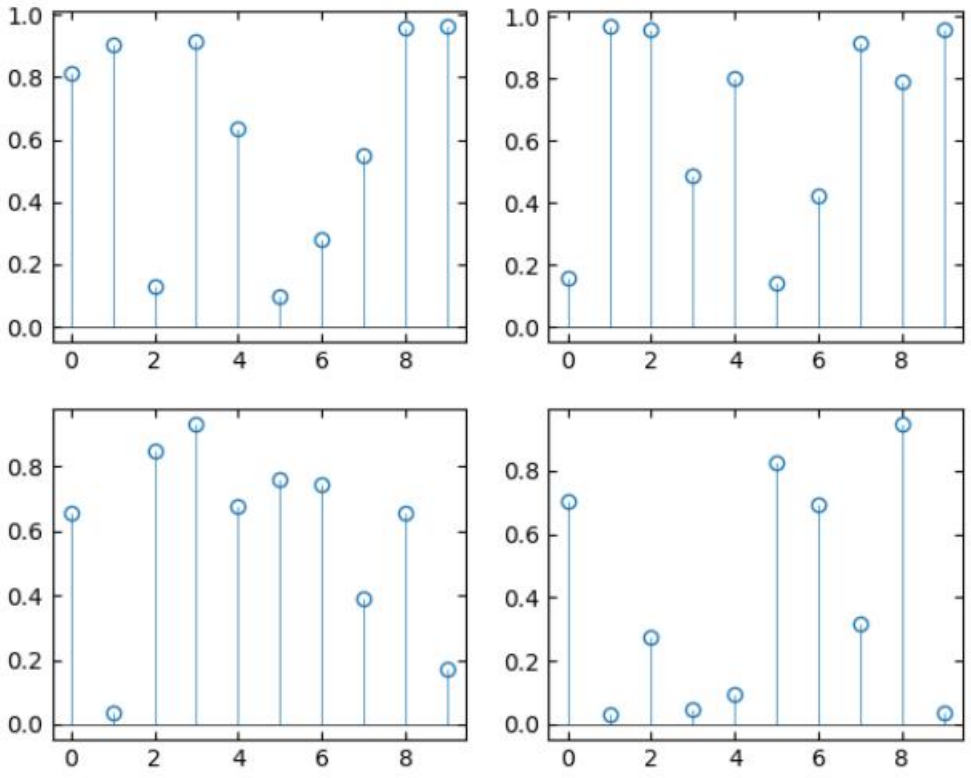
(11). 将子图替换为空坐标区

创建一个包含四个随机数据针状图的图窗。然后将第二个子图替换为空坐

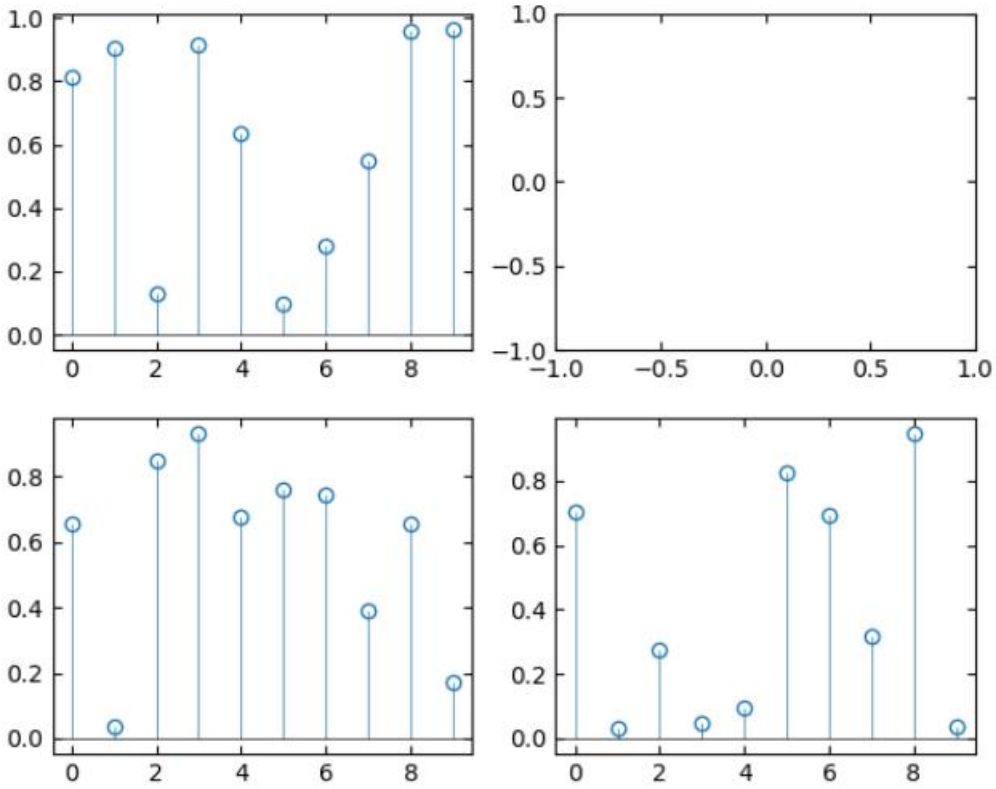
标区。

```
data_array = [
    0.8147  0.9058  0.1270  0.9134  0.6324  0.0975  0.2785  0.5469  0.9575
0.9649
    0.1576  0.9706  0.9572  0.4854  0.8003  0.1419  0.4218  0.9157  0.7922
0.9595
    0.6557  0.0357  0.8491  0.9340  0.6787  0.7577  0.7431  0.3922  0.6555
0.1712
    0.7060  0.0318  0.2769  0.0462  0.0971  0.8235  0.6948  0.3171  0.9502
0.0344
]

for k in 1:4
    data = data_array[k,:]
    subplot(2,2,k)
    stem(data)
end
```




```
ax2=subplot(2,2,2)
cla(ax2)
```



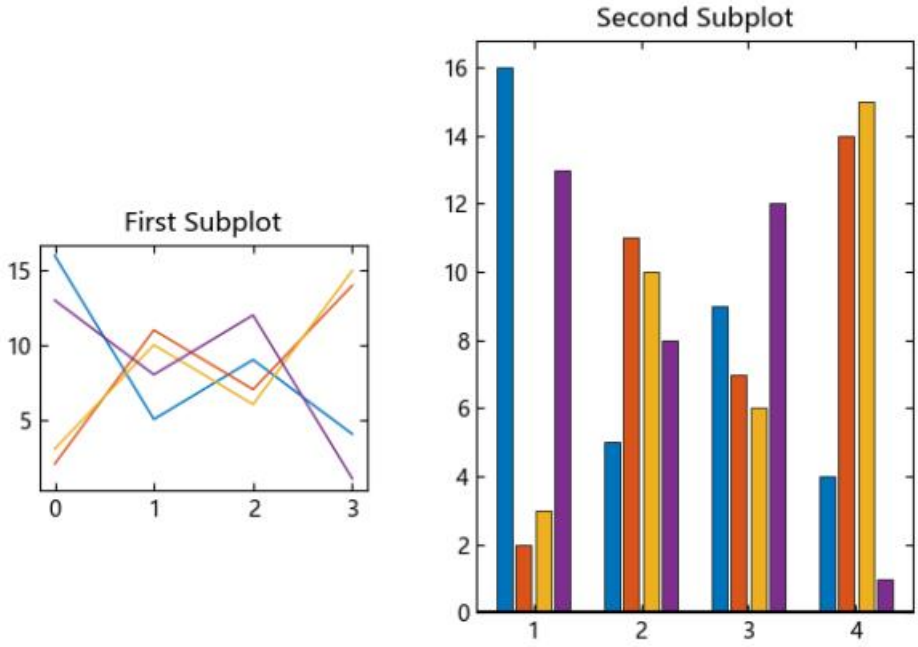
(12). 自定义位置上的子图

创建包含两个未与网格位置对齐的子图的图窗。为每个子图指定一个自定义位置。

```
pos1 = [0.1,0.3,0.3,0.3];
subplot(position=pos1)
y = [
    16     2     3    13
    5    11    10     8
    9     7     6    12
    4    14    15     1
]

plot(y)
title("First Subplot")

pos2 = [0.5,0.15,0.4,0.7];
subplot(position=pos2)
bar(y)
title("Second Subplot")
```

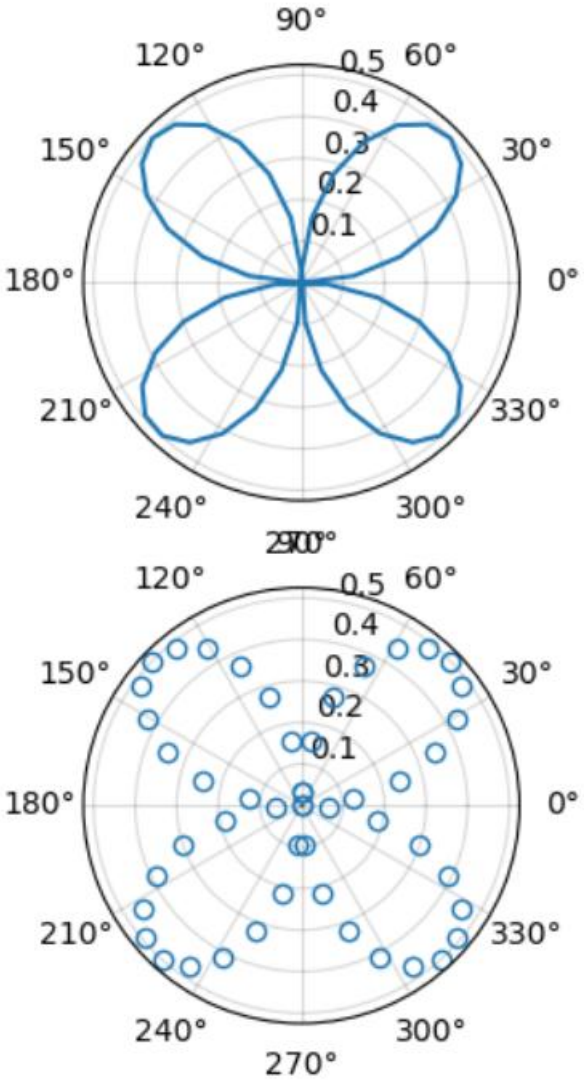


(13). 创建包含极坐标区的子图

创建包含两个极坐标区的图窗。在上部子图中创建极坐标线图，在下部子图中创建极坐标散点图。

```
figure()
ax1 = subplot(2, 1, 1; projection="polar");
theta = LinRange(0, 2 * pi, 50);
rho = sin.(theta) .* cos.(theta);
polarplot(ax1, theta, rho; rlim=(0, 0.5))

ax2 = subplot(2, 1, 2; projection="polar");
polarplot(ax2, theta, rho, "o"; rlim=(0, 0.5))
```



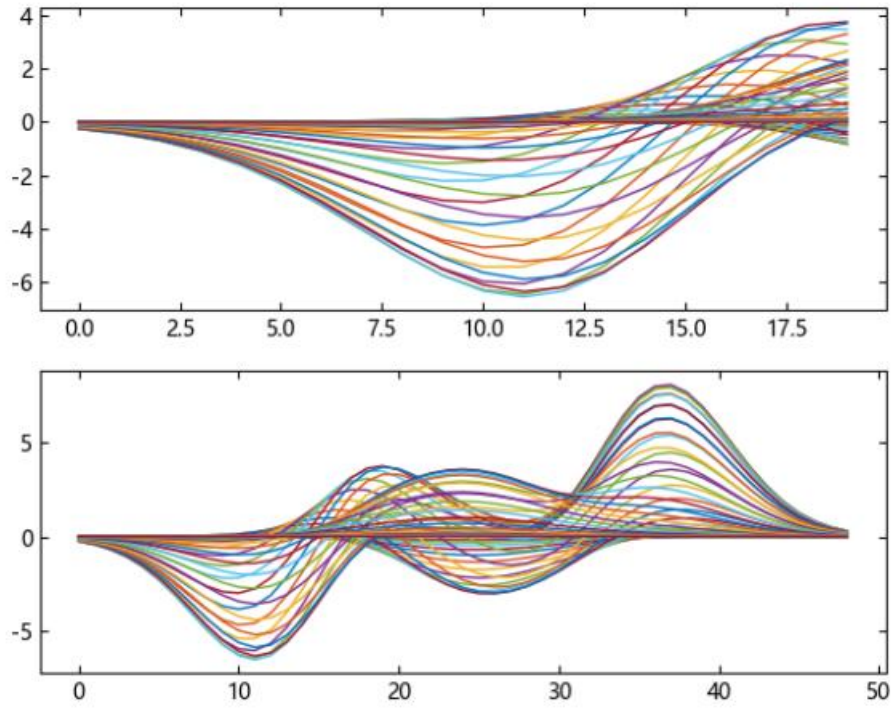
(14). 创建后修改坐标区属性

创建带有两个子图的图窗。将 Axes 对象赋给变量 ax1 和 ax2。将 Axes 对象指定为绘图函数的输入，以确保绘图函数在特定的子图中进行绘制。

```
Z = peaks();
Z = Z[3]

ax1 = subplot(2,1,1);
plot(ax1,Z[1:20,:])

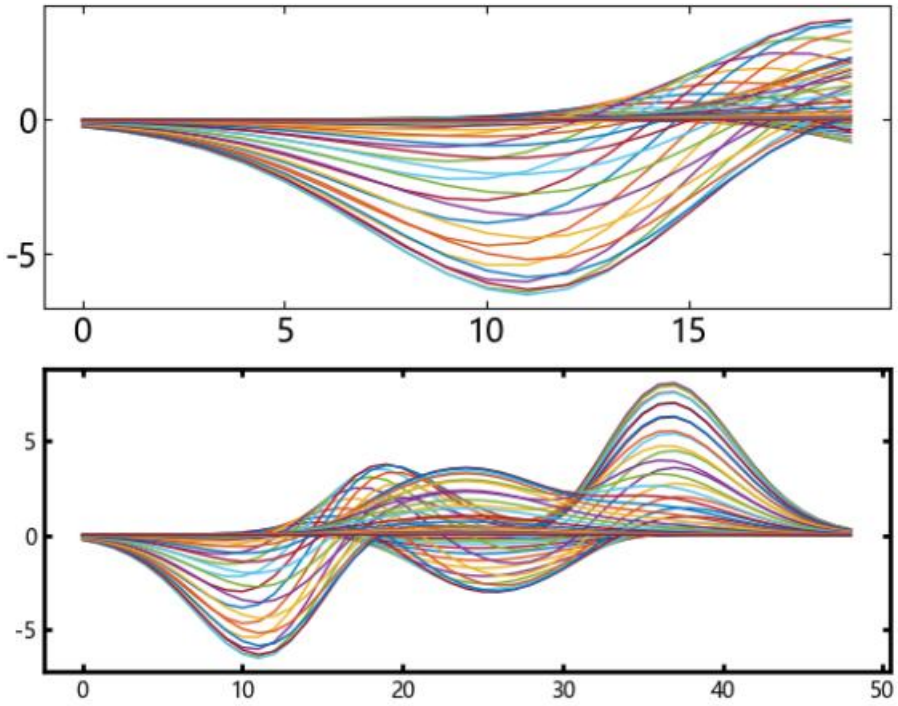
ax2 = subplot(2,1,2);
plot(ax2,Z)
```



通过设置 Axes 对象的属性来修改坐标区。更改上部子图的字体大小和下部子图的线宽。某些绘图函数可设置坐标区属性。先执行绘图函数，然后指定坐标区属性以免覆盖现有坐标区属性的设置。使用圆点表示法设置属性。

```
ax1.tick_params(labelsize=15)

ax2.tick_params(which = "major", width = 2)
ax2.spines["top"].set_linewidth(2)
ax2.spines["bottom"].set_linewidth(2)
ax2.spines["left"].set_linewidth(2)
ax2.spines["right"].set_linewidth(2)
```

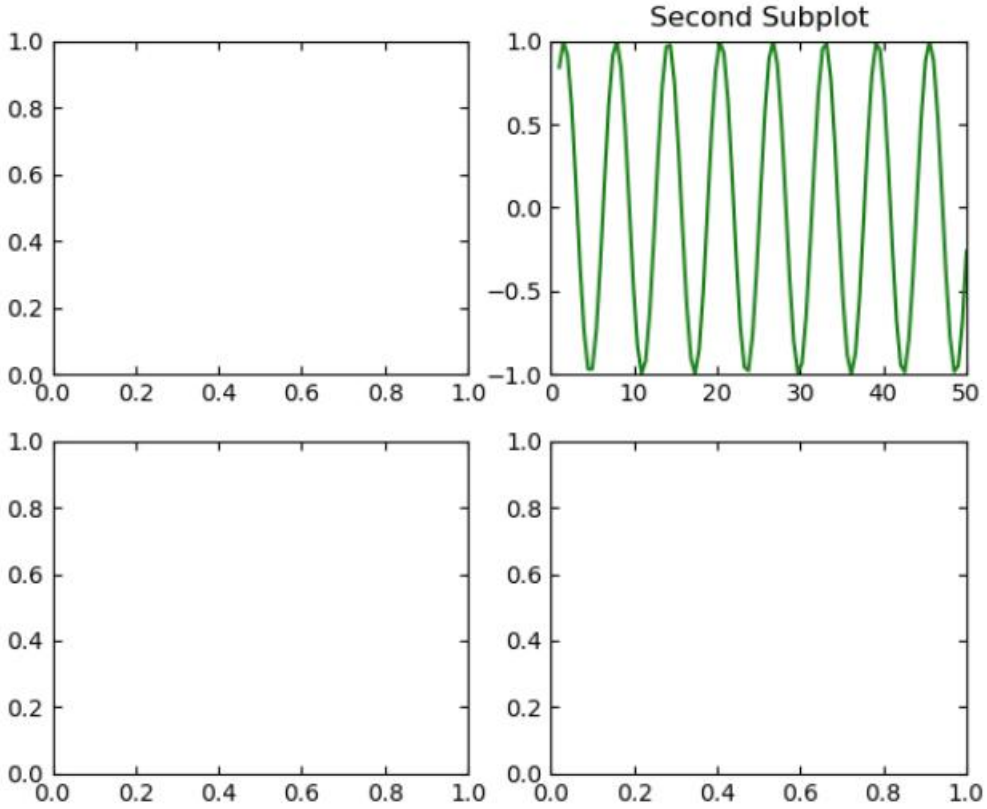


(15). 使子图成为当前坐标区

创建一个包含多个子图的图窗。将 Axes 对象存储在向量 ax 中。然后使第二个子图成为当前坐标区。创建一个线图并更改第二个子图的坐标轴范围。默认情况下，图形函数作用于当前坐标区。

```
ax = []
for k in 1:4
    push!(ax, subplot(2, 2, k))
end

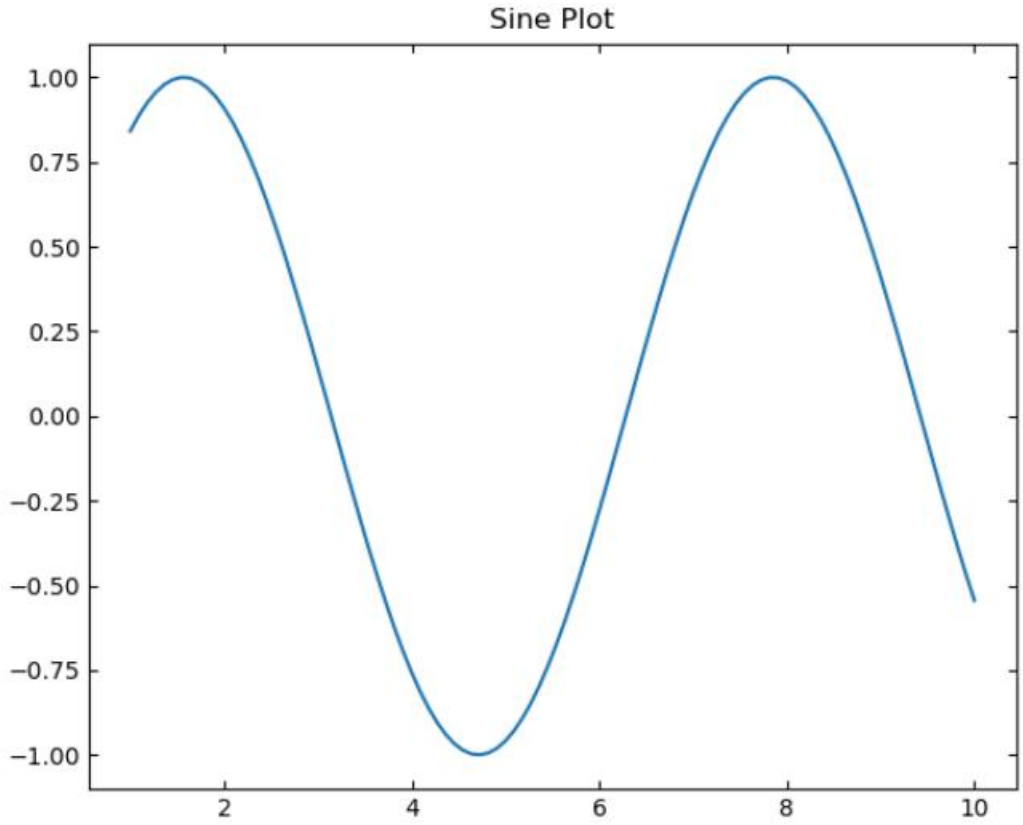
subplot(ax[2])
x = LinRange(1, 50, 100);
y = sin.(x);
plot(x, y; color=[0.1, 0.5, 0.1])
title("Second Subplot")
axis([0, 50, -1, 1])
```



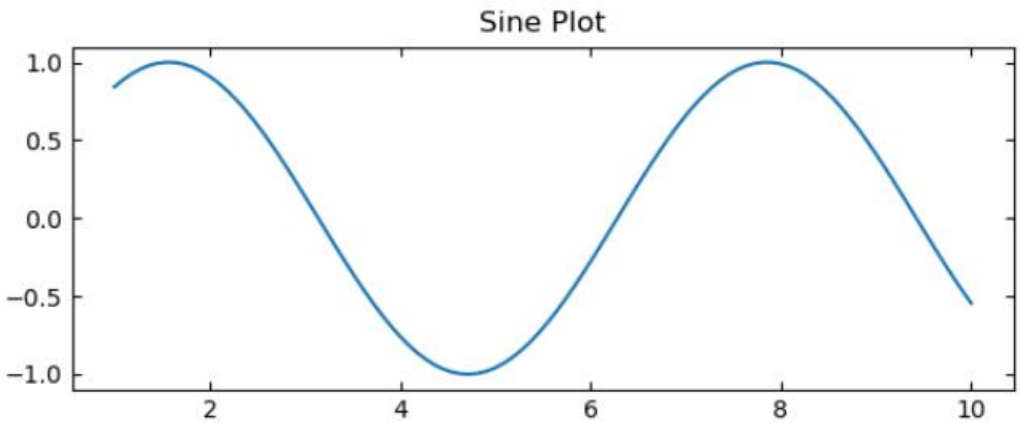
(16). 将现有坐标区转换为子图

创建一个线图。然后转换坐标区，使其成为图窗的下部子图。subplot 函数使用原始坐标区所在的图窗。

```
x = LinRange(1, 10, 100);
y = sin.(x);
plot(x, y)
title("Sine Plot")
```




```
ax = gca()
subplot(2,1,2,ax)
```



3.3.2.2.4 清除或创建坐标区

函数名	简介
cla	清除坐标区
plt_axes	创建笛卡尔坐标区
figure	创建图窗窗口

cla

清除坐标区

A. 语法

```
cla()
cla(ax)
```

```
cla("reset")
cla(ax,"reset")
```

B. 说明

cla() 从当前坐标区删除包含可见句柄的所有图形对象。如果坐标区不存在，此命令会创建一个坐标区。

cla(ax)删除 ax 指定的坐标区、极坐标区或地理坐标区中的图形对象，而不是当前坐标区中的图形对象。

cla("reset") 从当前坐标区删除图形对象，而不管其句柄可见性如何。它还将坐标区属性重置为默认值，但 Position 属性除外。

cla(ax,"reset")重置指定坐标区的属性。

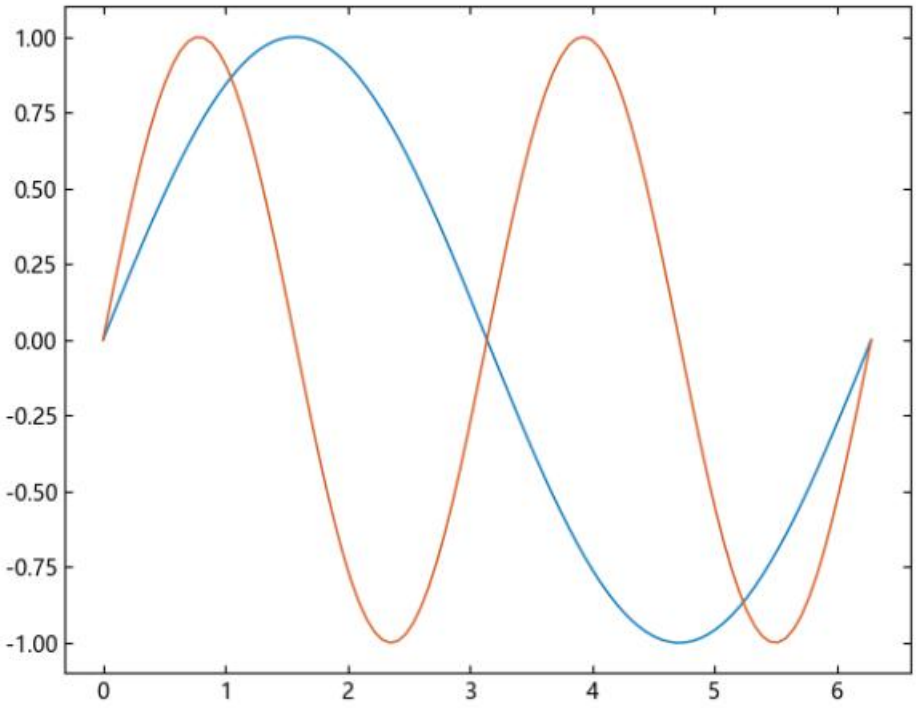
C. 示例

(1). 清除当前坐标区

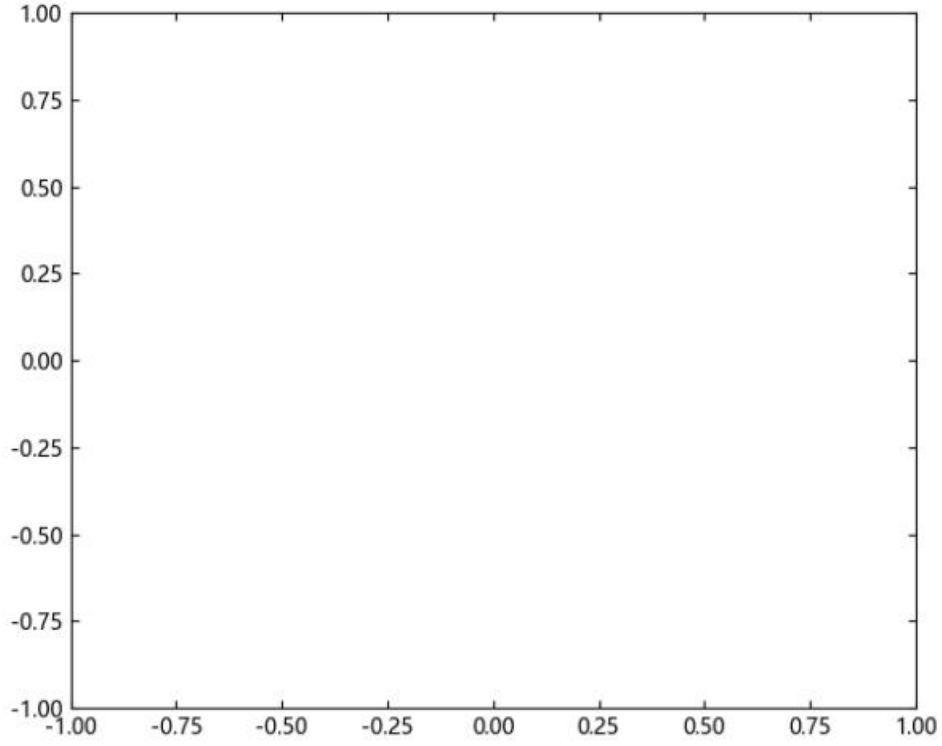
绘制两个正弦波。然后，从轴中清除线图。

```
x = LinRange(0, 2 * pi, 100);
y1 = sin.(x);
plot(x, y1)

hold("on")
y2 = sin.(2 * x);
plot(x, y2)
```

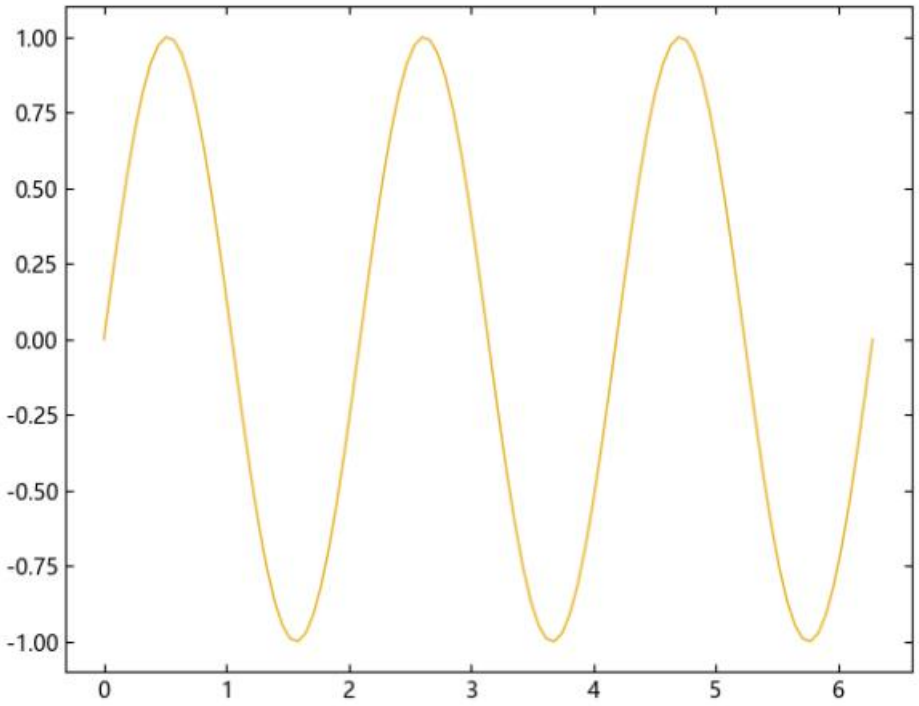



```
cla()
```



绘制另一个正弦波。

```
y3 = sin.(3 * x);
plot(x, y3)
hold("off")
```

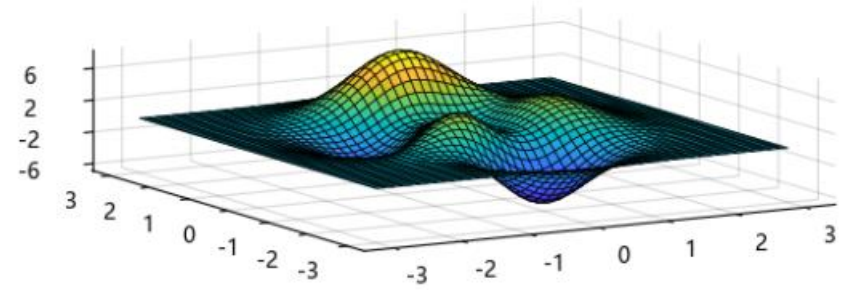


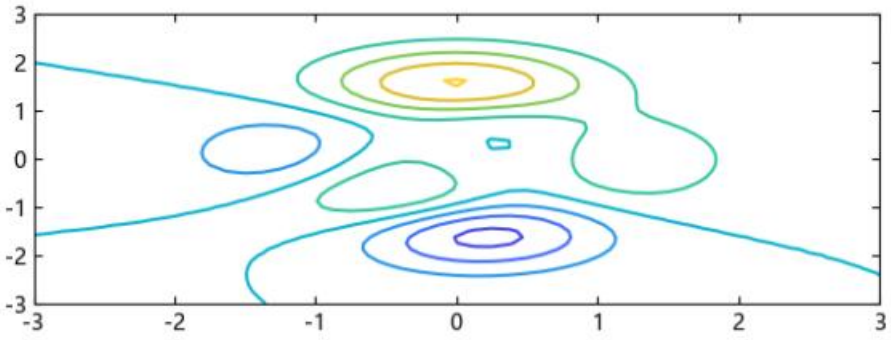
(2). 清除特定坐标区

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1。将绘图添加到两个坐标区中。

```
ax1 = subplot(2,1,1, projection = "3d")
X,Y,Z = peaks()
surf(ax1,X,Y,Z)

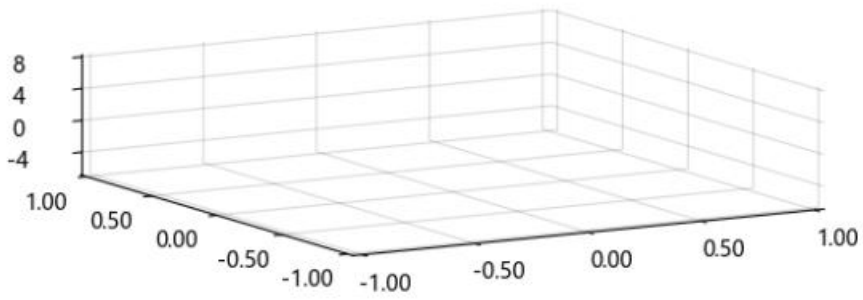
ax2 = subplot(2,1,2)
contour(ax2, X,Y,Z)
```

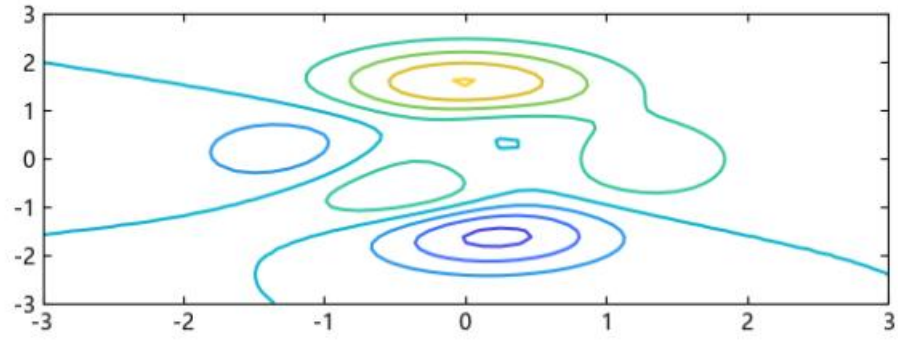




通过指定 ax1 作为 cla 的输入参数，清除上方坐标区中的曲面图。

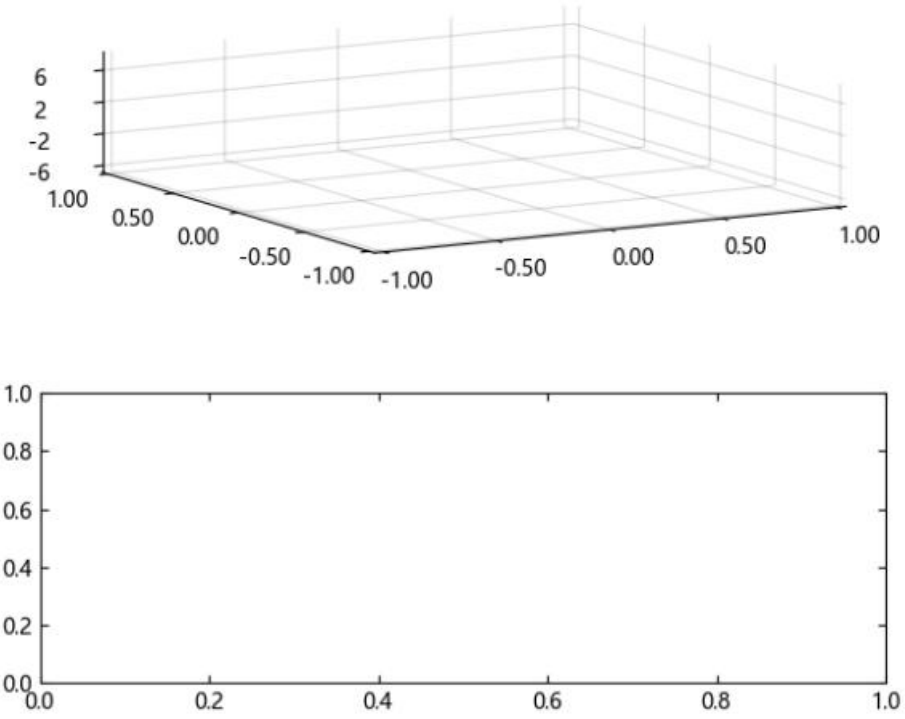
```
cla(ax1)
```





现在，使用可选输入参数 "reset" 重置坐标区的所有属性，包括用于控制视图的相机属性。

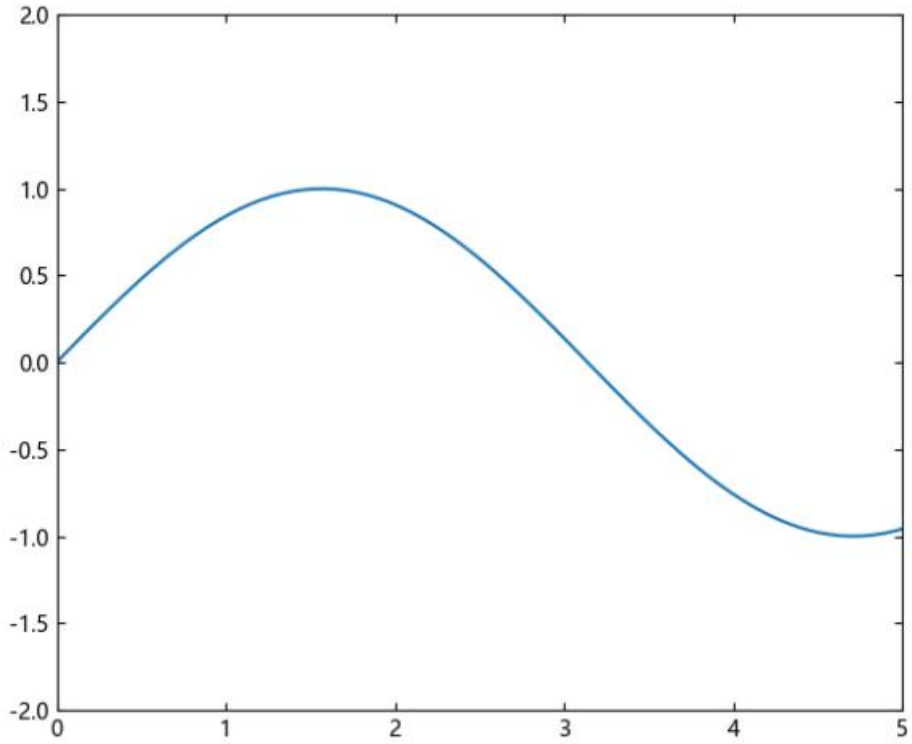
```
cla(ax2,"reset")
```



(3). 清除坐标区并重置所有坐标区属性

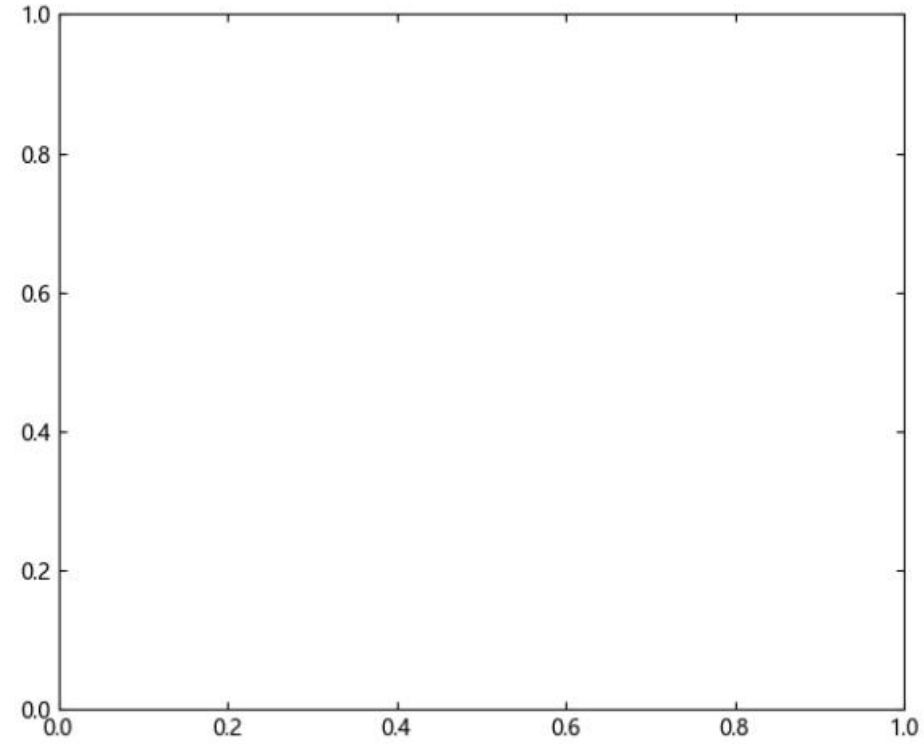
创建一个线图并设置坐标轴范围。

```
x = LinRange(0, 2 * pi, 100);
y = sin.(x);
plot(x, y)
axis([0, 5, -2, 2])
```



清除轴中的线图并将所有轴属性重置为其默认值。cla("reset") 重置当前轴的所有属性，但 position 属性除外。

```
cla("reset")
```



plt_axes

创建笛卡尔坐标区

A. 语法

```
plt_axes()
plt_axes(pos)
plt_axes(Key=Value)
ax = plt_axes(___)

plt_axes(cax)
```

B. 说明

plt_axes() 在当前图窗中创建默认的笛卡尔坐标区，并将其设置为当前坐标区。通常情况下，您不需要在绘图之前创建坐标区，因为如果不存在坐标区，图形函数会在绘图时自动创建坐标区。

plt_axes(pos)根据输入的位置来创建笛卡尔坐标区。

plt_axes(Key=Value) 使用一个或多个名称-值对组参数修改坐标区的外观，或控制数据的显示方式。例如，fontsize=14 可设置坐标区文本的字体大小为 14。

ax = plt_axes(___) 返回创建的 Axes 对象。可在创建 Axes 对象后使用 ax 查询和修改对象属性。

plt_axes(cax) 将 cax 指定的坐标区或图设置为当前坐标区，并使父图窗成为焦点。此命令还将 cax 设置为图窗的 Children 属性中列出的第一个对象，并将图窗的 CurrentAxes 属性设置为 cax。

C. 示例

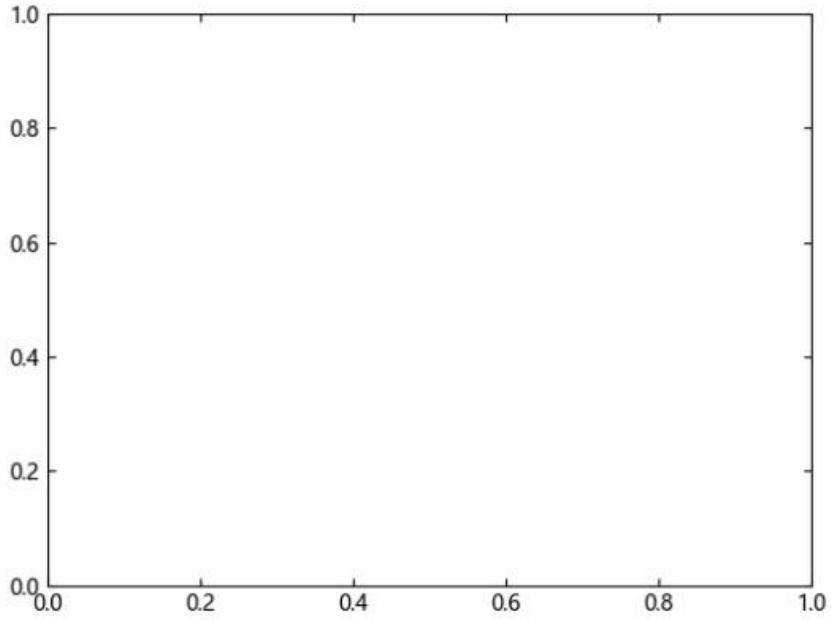
(1). 在图窗中定位多个坐标区

在图窗中放置两个 Axes 对象，并为每个对象添加一个绘图。

指定第一个 Axes 对象的位置，使其左下角位于点 (0.1 0.1) 处，宽度和高度均为 0.7。指定第二个 Axes 对象的位置，使其左下角位于点 (0.65 0.65) 处，宽度和高度均为 0.28。默认情况下，所有值为基于图窗的归一化值。

将这两个 Axes 对象返回为 ax1 和 ax2。

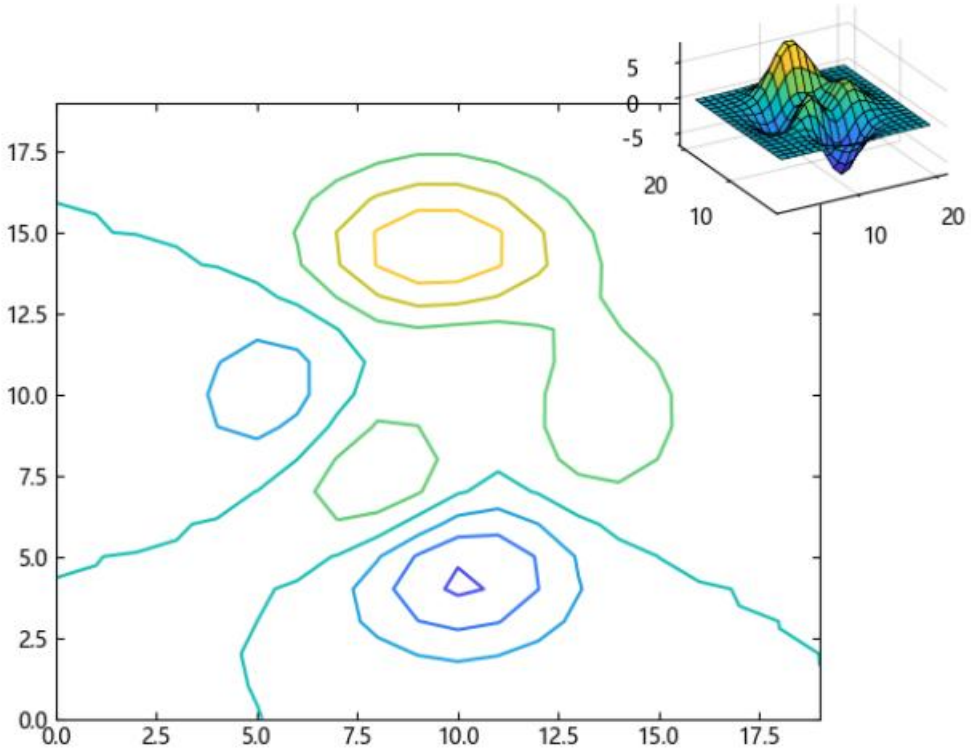
```
figure()
ax1 = plt_axes([0.1 0.1 0.7 0.7]);
```



在每个 Axes 对象上添加一个绘图。通过将坐标区作为第一个输入参数传递给图形函数来指定坐标区。大多数图形函数会重置某些坐标区属性，如刻度值和标签。但是，它们不会重置坐标区的位置。

```
ax2 = plt_axes([0.65 0.65 0.28 0.28], projection="3d");

X,Y,Z = peaks(20)
contour(ax1,Z)
surf(ax2,Z)
```

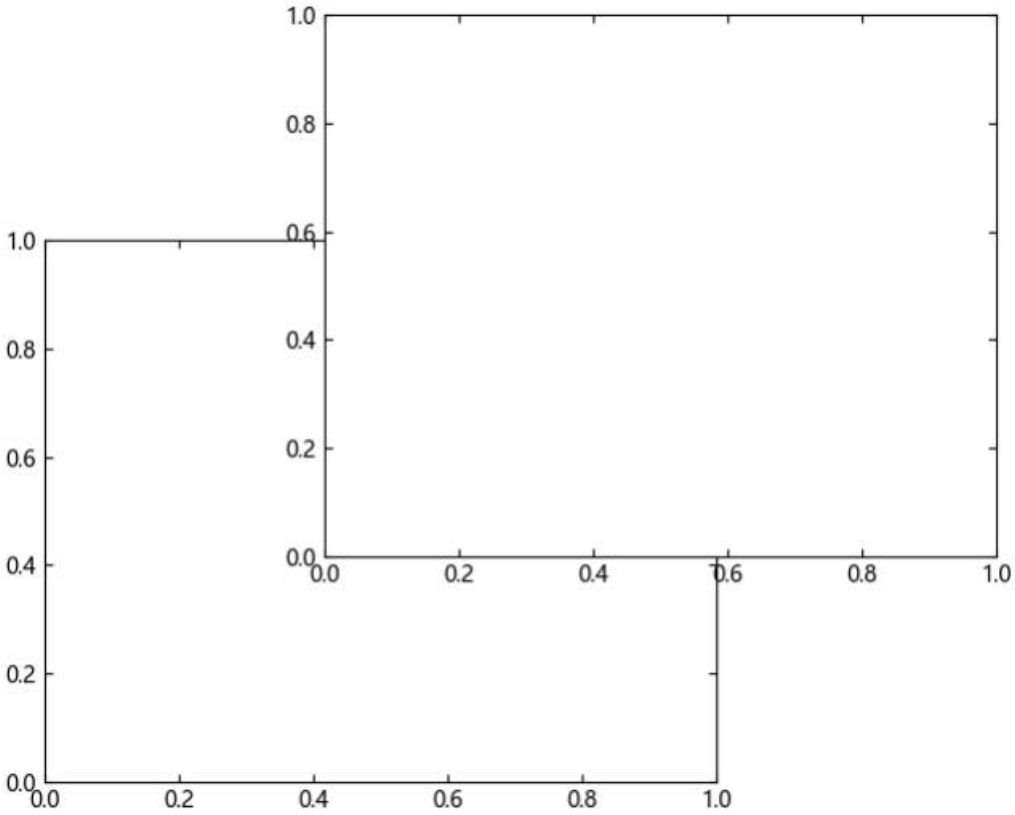


(2). 将坐标区设置为当前坐标区

创建两个重叠的 Axes 对象。然后指定当前坐标区并添加一个绘图。

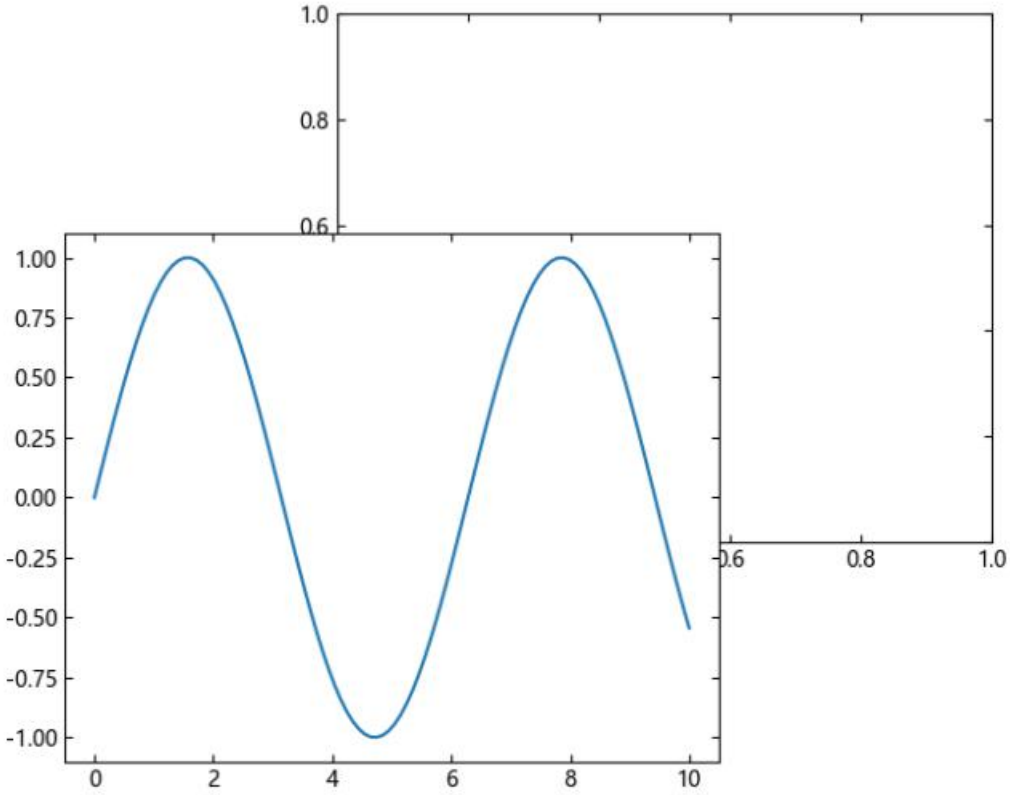
首先创建两个 Axes 对象并指定它们的位置。在每个坐标区周围显示框轮廓。将这两个 Axes 对象返回为 ax1 和 ax2。

```
figure()
ax1 = plt_axes([0.1 0.1 .6 .6]);
ax2 = plt_axes([.35 .35 .6 .6]);
```



将 ax1 设置为当前坐标区。此操作将使该坐标区显示在最前面，并使其成为后续图形函数的目标。在坐标区上添加一个线图。

```
plt_axes(ax1)
x = LinRange(0, 10, 100);
y = sin.(x);
plot(x, y)
```



figure

创建图窗窗口

A. 语法

```
figure()
figure(Key=Value)
f = figure(___)

figure(f)
figure(n)
```

B. 说明

figure() 使用默认属性值创建一个新的图窗窗口。生成的图窗为当前图窗。

figure(Key=Value)使用一个或多个名称-值对组参数修改图窗的属性。例如，figure(color="white")将背景色设置为白色。

f=figure(___)返回 Figure 对象。可使用 f 在创建图窗后查询或修改其属性。

figure(f)将 f 指定的图窗作为当前图窗，并将其显示在其他所有图窗的上面。

figure(n) 查找 Number 属性等于 n 的图窗，并将其作为当前图窗。如果

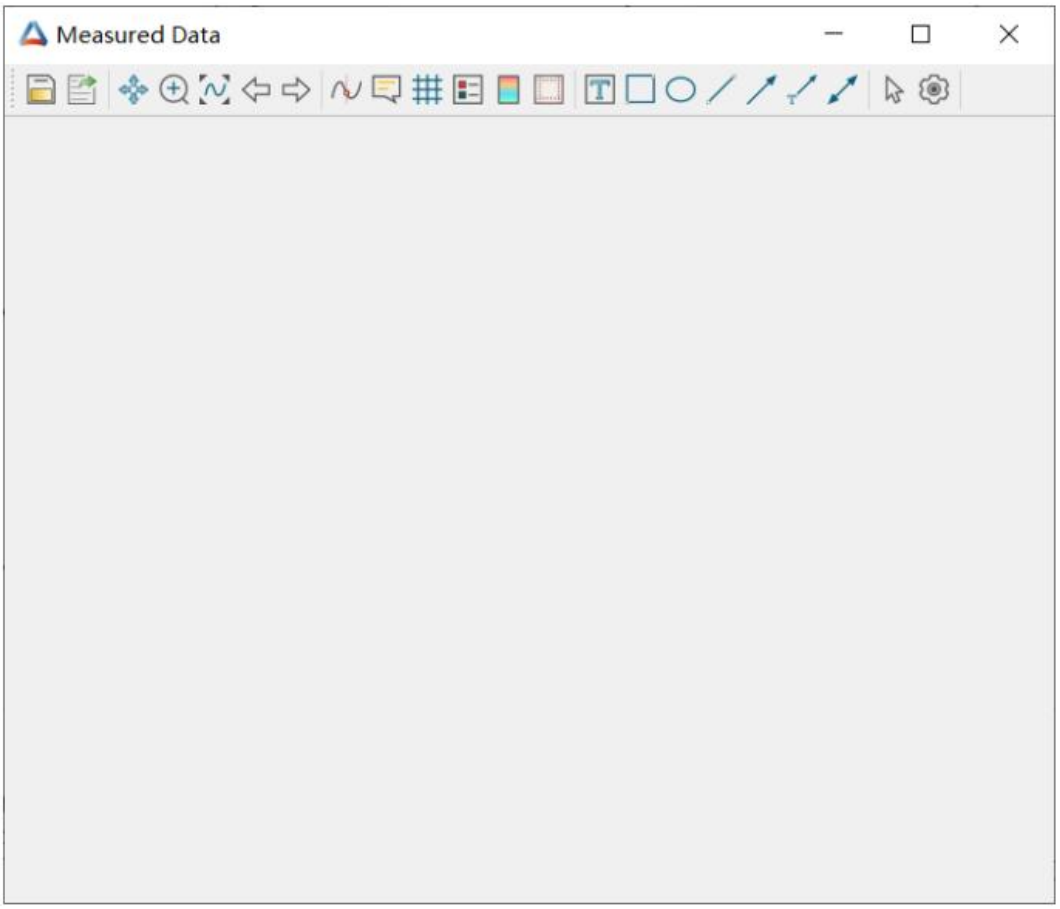
不存在具有该属性值的图窗，Syslab 将创建一个新图窗并将其 Number 属性设置为 n。

C. 示例

(1). 指定图窗标题

创建一个图窗并指定 Name 属性。

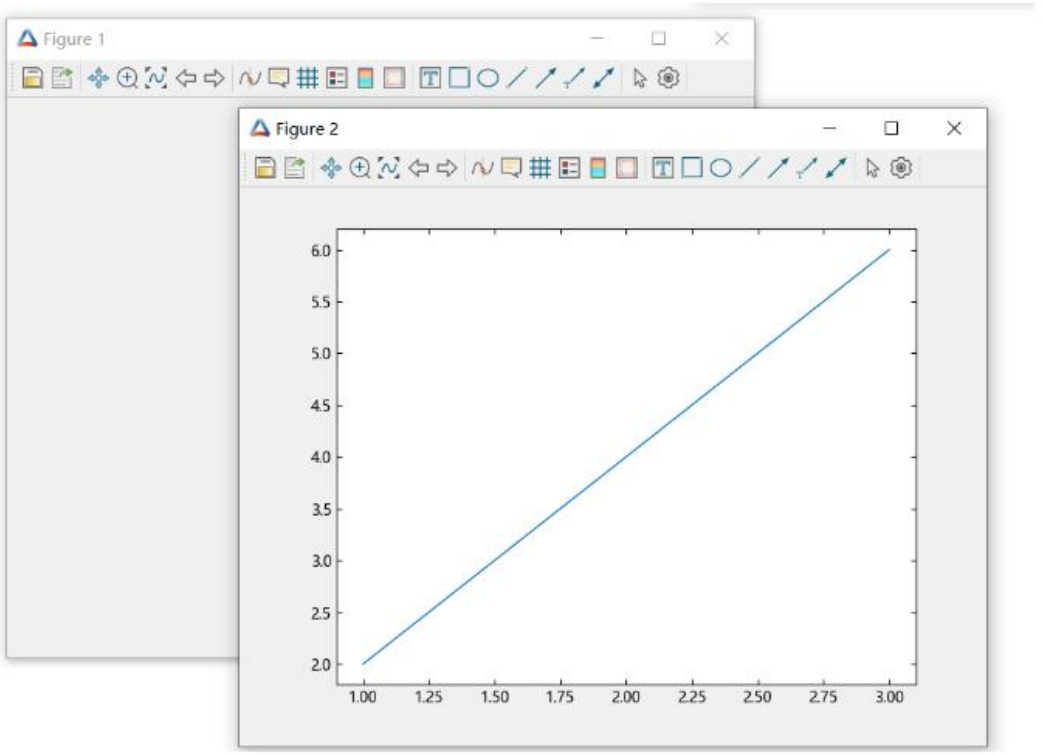
```
f1=figure("Measured Data")
```



(2). 同时使用多个图窗

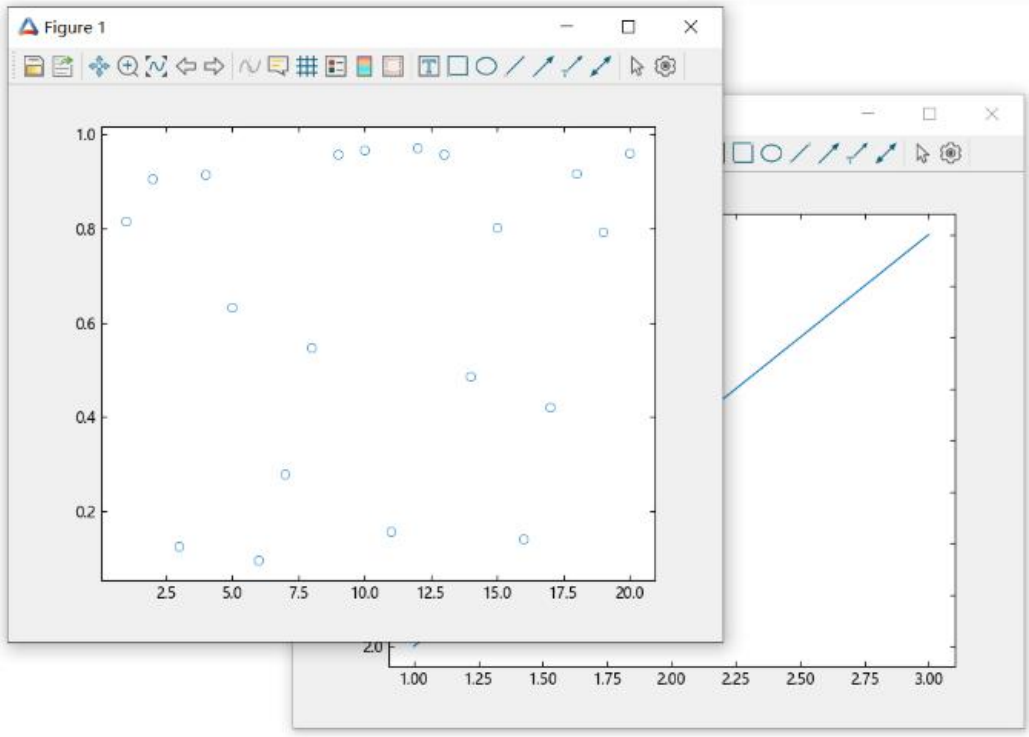
创建两个图窗，然后创建一个线图。默认情况下，plot 命令的目标为当前图窗。

```
f1 = figure()
f2 = figure()
plot([1,2,3],[2,4,6])
```



将当前图窗设置为 f1，使其成为下一个绘图的目标。然后创建一个散点图。

```
figure(f1)
scatter(1:20, rand(1,20)) #返回一个 1×20 的随机数矩阵
```



3.3.2.3 颜色图

条形图、散点图等

使用条形图或针状图等将离散数据可视化。例如，可以创建垂直或水平条形图，其中条形长度与它们所代表的值成比例。

3.3.2.3.1 颜色图和颜色栏

函数名	简介
colormap	查看并设置当前颜色图
colorbar	显示色阶的颜色栏
rgbplot	绘制颜色图
caxis	设置颜色图范围

colormap

查看并设置当前颜色图

A. 语法

```
colormap(target,map)

cmap = colormap(target)
```

B. 说明

colormap(target,map)将当前对象的颜色图设置为 map 指定的颜色图。

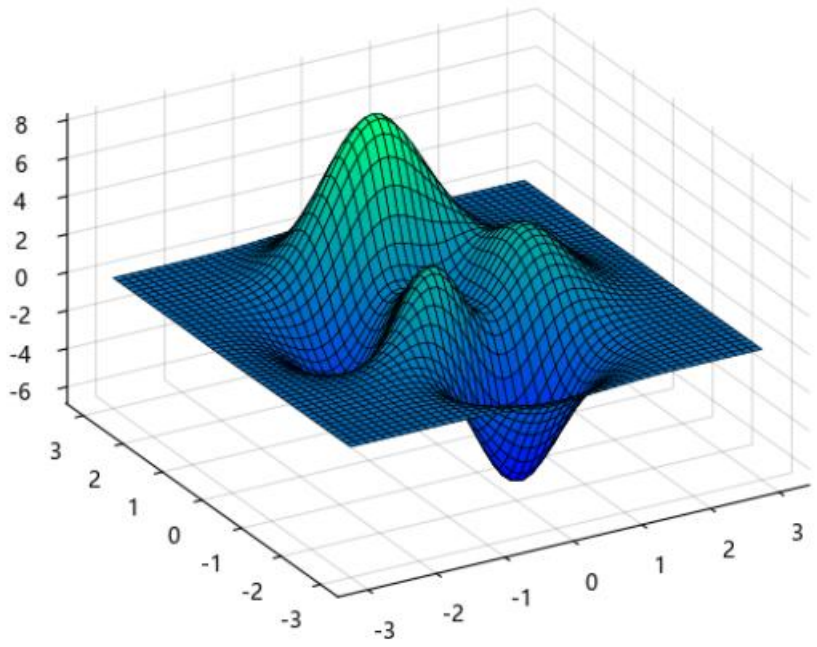
cmap=colormap(target)返回 target 指定对象的颜色图。

C. 示例

(1). 更改图窗的颜色图

创建一个曲面图并将颜色图设置为 winter。

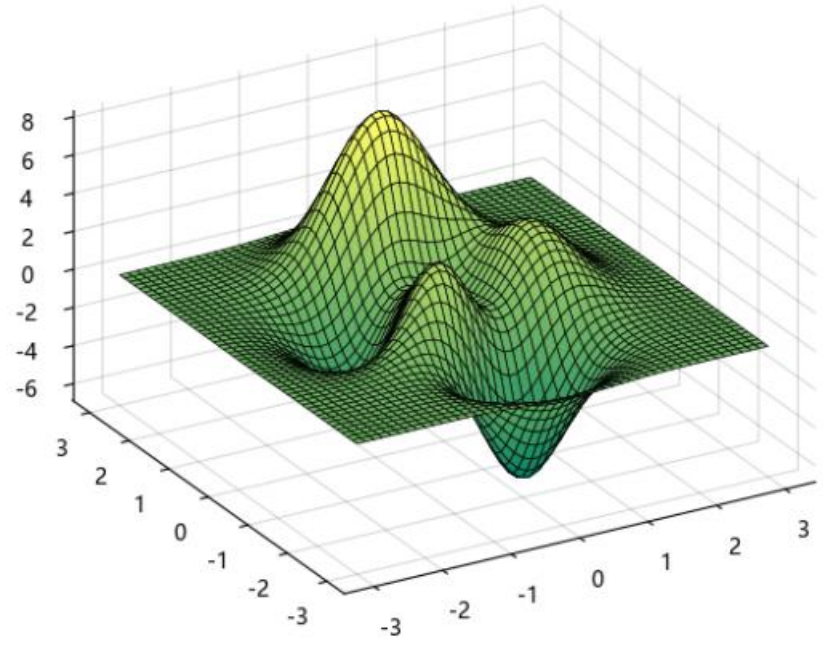
```
x,y,z = peaks();
c = surf(x,y,z);
colormap(c, "winter")
```



(2). 将颜色图设置回默认值

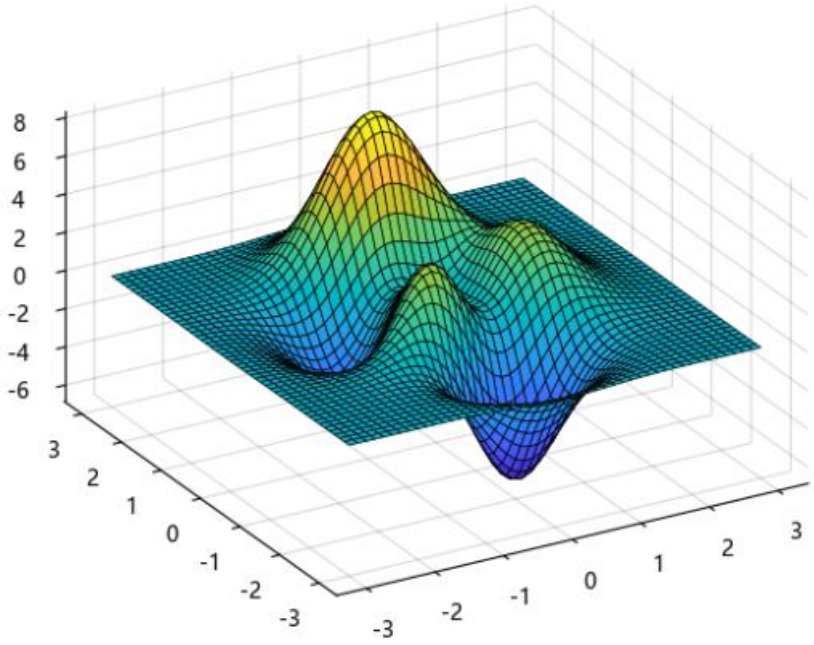
首先，将当前图窗的颜色图更改为 summer。

```
x,y,z = peaks();
c = surf(x,y,z);
colormap(c, "summer")
```



现在将颜色图设置回您系统的默认值。如果您尚未指定不同默认值，则默认颜色图是 parula。

```
colormap(c, "default")
```

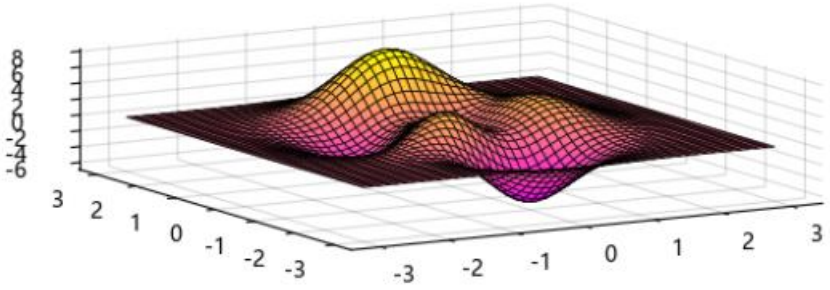


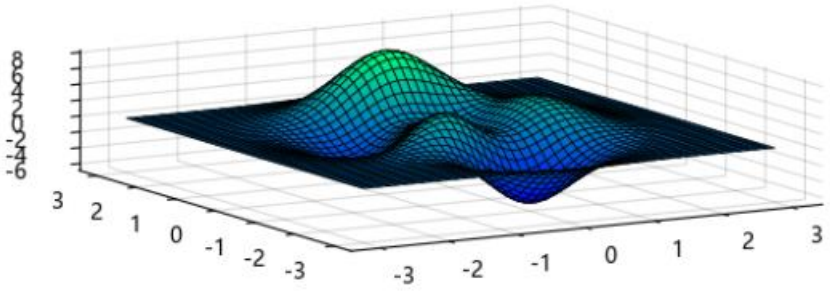
(3). 对图窗中的每个坐标区使用不同的颜色图

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，通过将坐标区对象传递给 colormap 函数，为每个坐标区指定不同的颜色图。在上坐标区中，使用 spring 颜色图创建一个曲面图。在下坐标区中，使用 winter 颜色图创建一个曲面图。

```
ax1 = subplot(2,1,1,projection="3d")
x,y,z = peaks();
c1 = surf(ax1, x, y, z);
colormap(c1, "spring")

ax2 = subplot(2,1,2,projection="3d")
x,y,z = peaks();
c2 = surf(ax2, x, y, z);
colormap(c2, "winter")
```

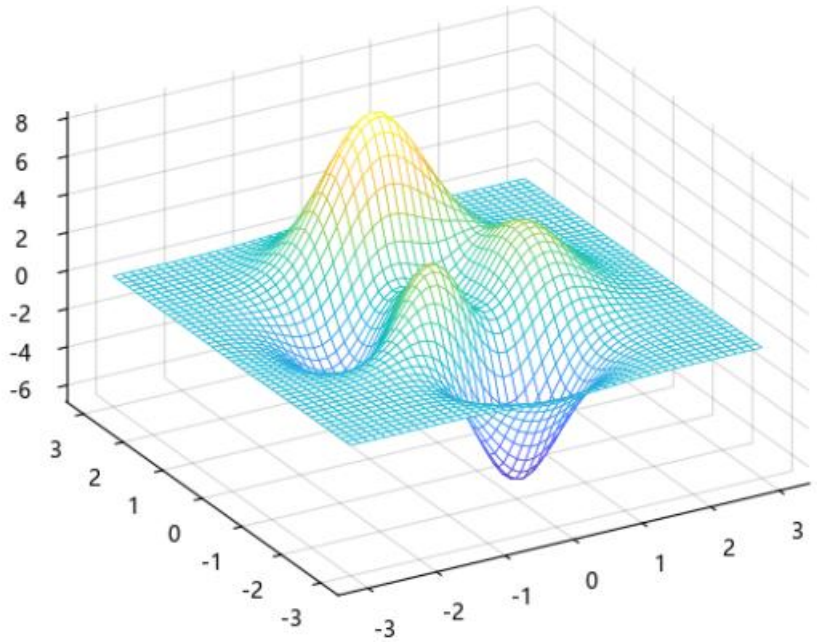




(4). 指定颜色图的颜色数

通过将整数作为内置颜色图的输入参数传递来指定用于颜色图的颜色数。使用 parula 颜色图中的五种颜色。

```
x,y,z = peaks();
c = mesh(x,y,z);
colormap(c, "parula");
```



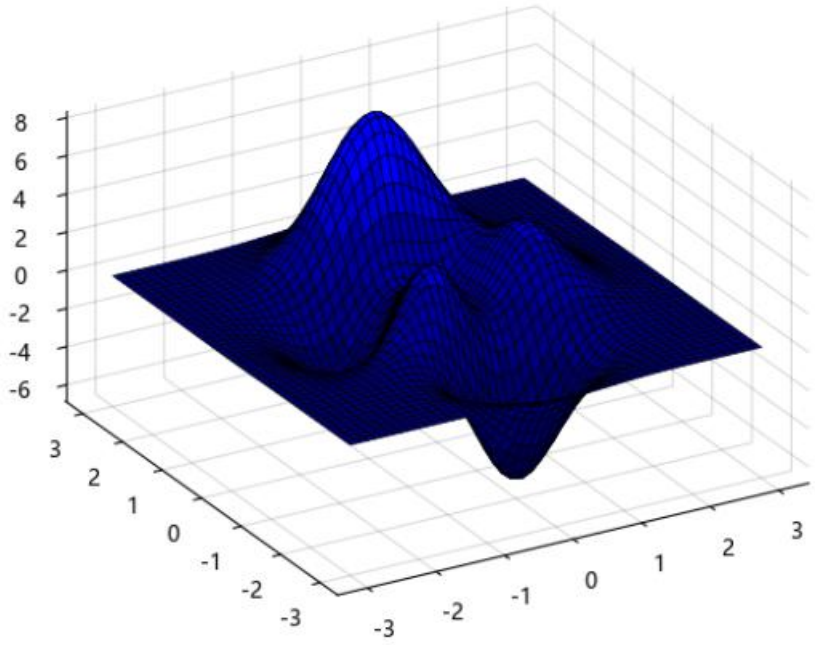
(5). 创建自定义颜色图

通过定义一个由介于 0.0 和 1.0 之间的值组成的三列矩阵来创建一个自定义颜色图。每行定义一个三元素 RGB 三元组。第一列指定红色强度。第二列指定绿色强度。第三列指定蓝色强度。

通过将前两个列设置为零来使用蓝色值的颜色图。

```
cmap = [0 0 0.3
    0 0 0.4
    0 0 0.5
    0 0 0.6
    0 0 0.8
    0 0 1.0];

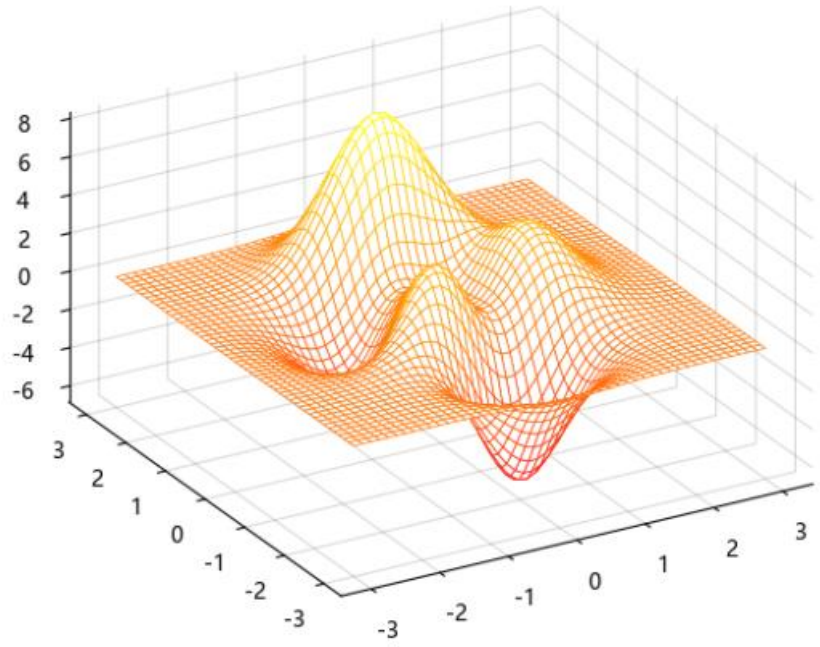
x,y,z = peaks();
c = surf(x,y,z);
colormap(c, cmap);
```



(6). 返回用在绘图中的颜色图

值创建 peaks 函数的曲面图并指定颜色图。

```
x,y,z = peaks();
c = mesh(x,y,z);
colormap(c, "autumn");
```



(7). 返回特定对象的颜色图

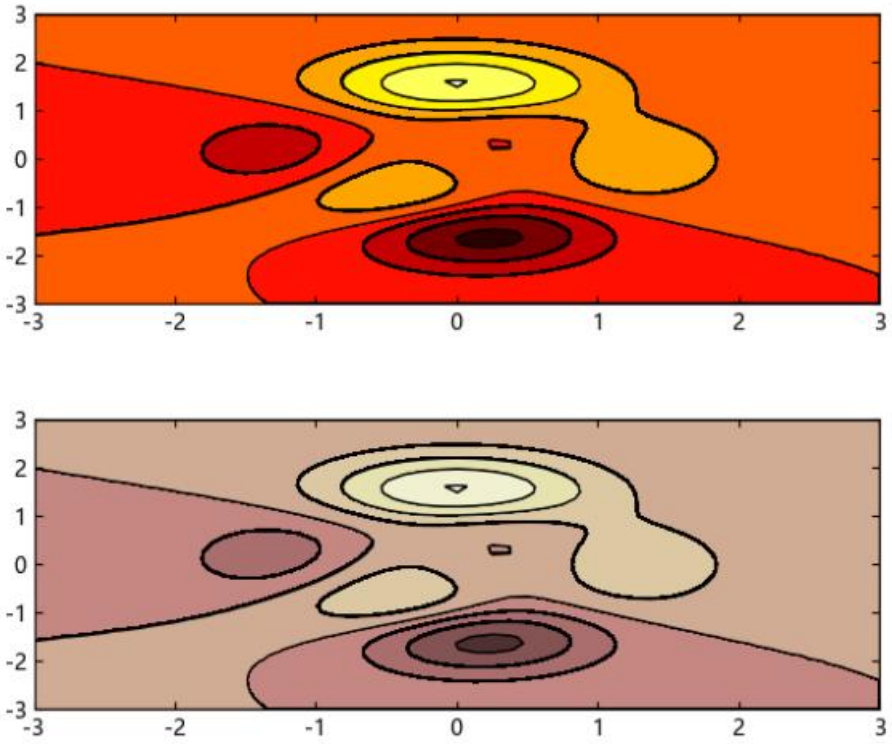
值通过将坐标区对象传递给 colormap 函数，返回特定对象的颜色图值。

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，然后显示两个以不同颜色图填充的等高线图。

```
ax1 = subplot(2,1,1)
x,y,z = peaks();
c1 = contourf(ax1, x, y, z);
colormap(c1, "hot")



ax2 = subplot(2,1,2)
x,y,z = peaks();
c2 = contourf(ax2, x, y, z);
colormap(c2, "pink")
```

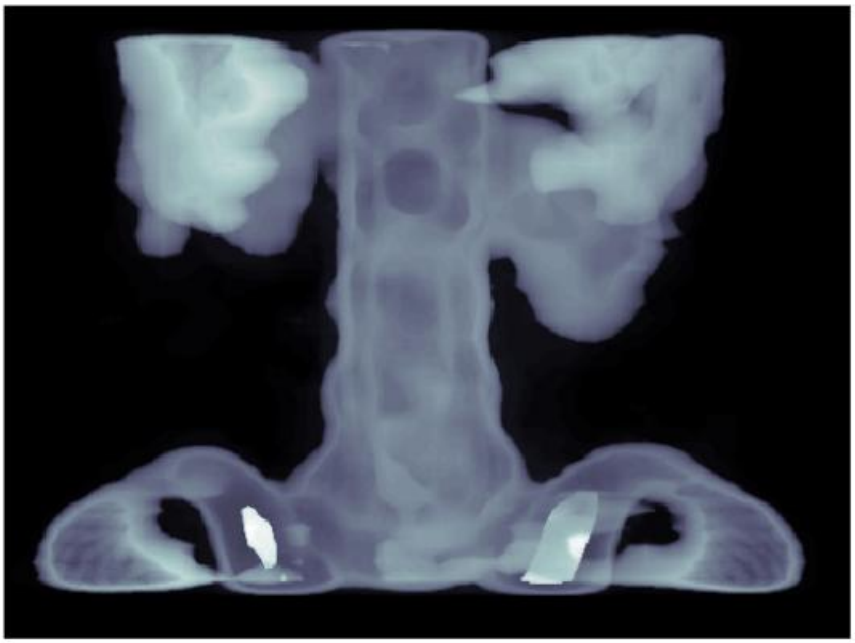


(8). 将图窗的颜色图更改为图像

导入 "0-data.jl" 文件,加载 spine 数据集以返回 X 及其关联的颜色图 cmap。

使用 imshow 函数显示 X 并将颜色图设置为 cmap。

```
using TyImages
import TyPlot
pkg_dir = pkgdir(TyPlot)
source_path = pkg_dir * "/examples/格式和注释/colormap/0-data.jl"
include(source_path)
gcf();
m = imshow(X,cmap);
```



colorbar

显示色阶的颜色栏

A. 语法

```
colorbar(target)
colorbar(target, Key=Value)
c = colorbar(___)
colorbar(ax,___)

colorbar("off")
colorbar(ax, "off")
```

B. 说明

colorbar(target)在给定图的右侧显示一个垂直颜色栏。颜色栏显示当前颜色图并指示数据值到颜色图的映射。

colorbar(target,Key=Value)使用一个或多个名称-值对组参数修改颜色栏外观。例如，direction="reverse"将反转色阶。指定 Key=Value 作为上述任一语法中的最后一个参数对组。并非所有类型的图都支持修改颜色栏外观。

colorbar(ax, ___) 根据给定坐标区创建图形对象。在给定坐标区的右侧显示一个垂直颜色栏。

c = colorbar(___) 返回 ColorBar 对象。您可以在创建颜色栏后使用此对象设置属性。可将返回参数 c 指定到上述任一语法中。

colorbar("off") 删除与当前坐标区或图关联的所有颜色栏。

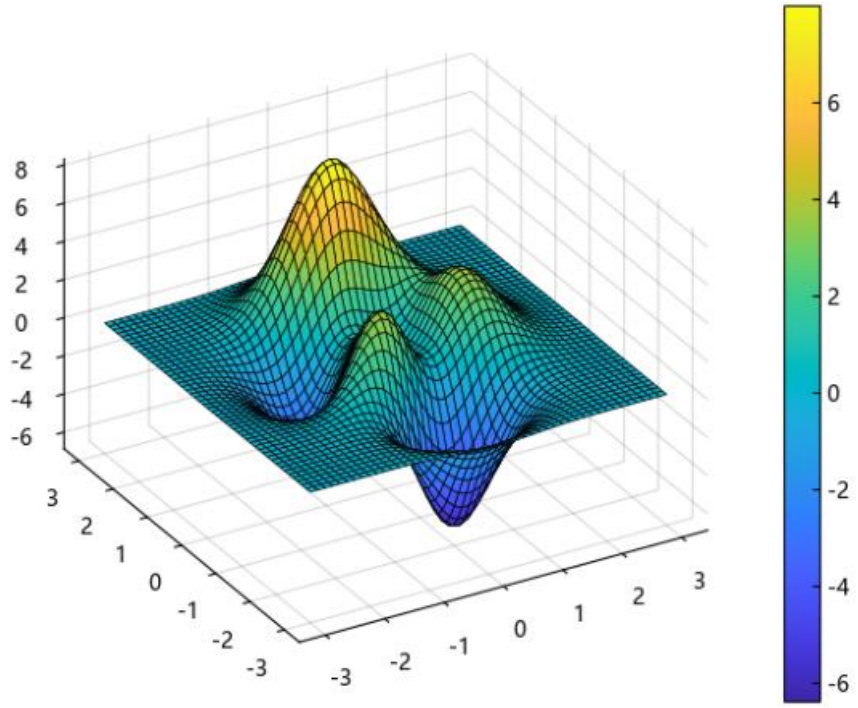
colorbar(ax, "off") 删除与目标坐标区关联的所有颜色栏。您也可以将 ColorBar 对象指定为目标。

C. 示例

(1). 向图表添加颜色栏

在曲面图中添加指示色阶的颜色栏。

```
x,y,z = peaks();
s = surf(x,y,z)
colorbar(gca(), s)
```



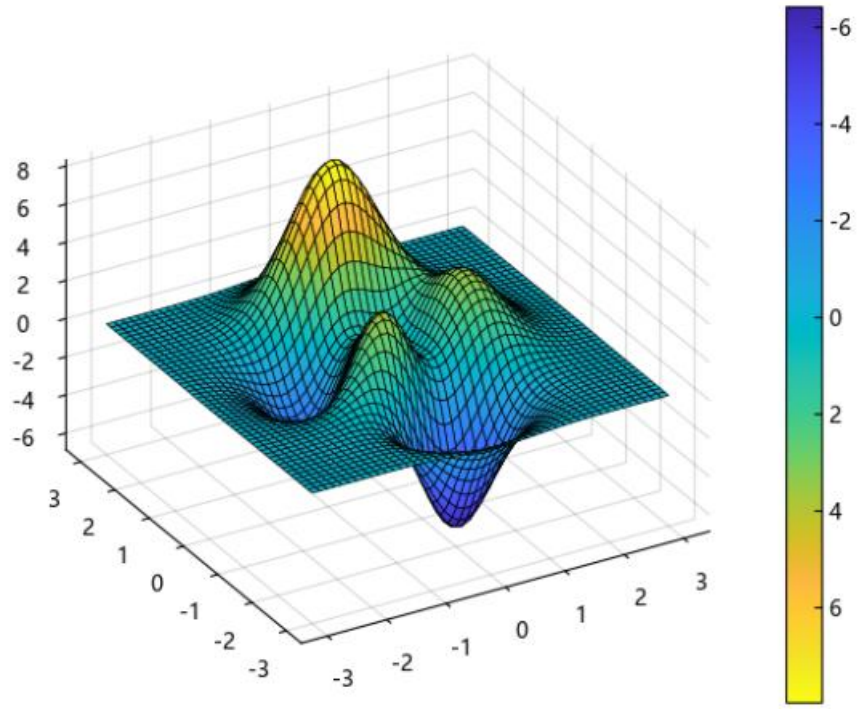
默认情况下，colorbar 函数向图右侧添加一个垂直颜色栏。

(2). 反转颜色栏方向

通过将颜色栏的 "direction" 属性设置为 "reverse" 反转图上颜色栏中的值方向。

```
x,y,z = peaks();
s = surf(x,y,z)
```

```
colorbar(gca(), s, direction="reverse")
```



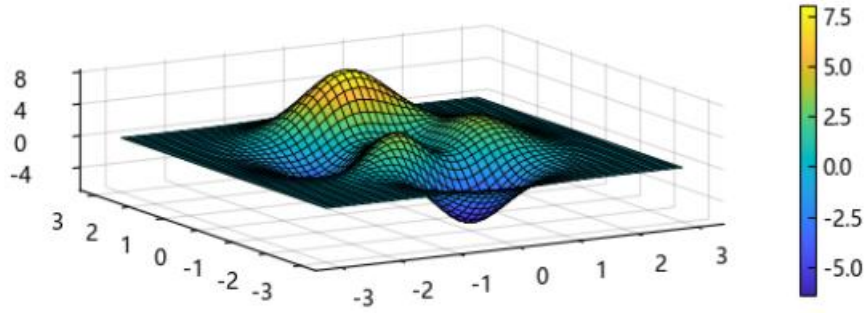
颜色栏值自上而下上升，而不是自下而上上升。

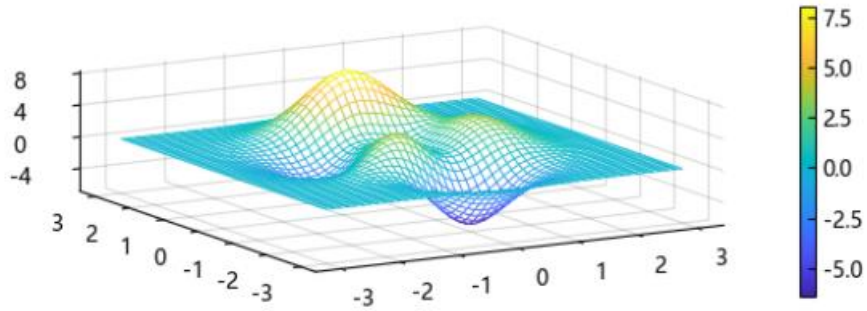
(3). 向分块图布局添加颜色栏

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，然后在每个坐标区显示一个曲面图，且每个坐标区都显示一个颜色栏。

```
ax1 = subplot(2,1,1, projection="3d")
x,y,z = peaks();
s = surf(ax1, x,y,z)
colorbar(s)

ax2 = subplot(2,1,2, projection="3d")
m = mesh(ax2, x,y,z)
colorbar(m)
```

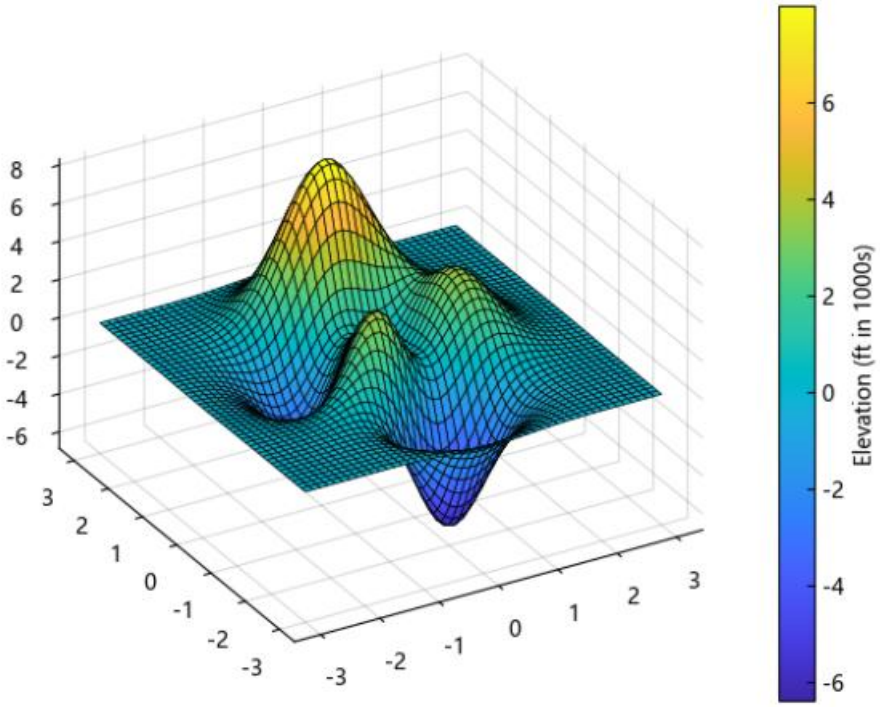




(4). 标注颜色栏

沿颜色栏添加一个文本标签。

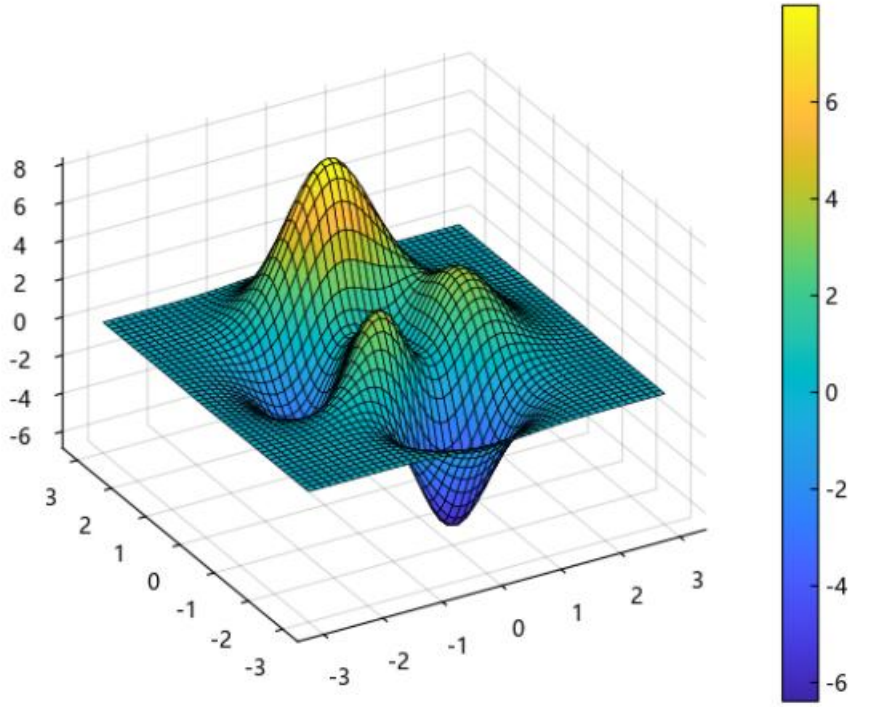
```
x,y,z = peaks();
s = surf(x,y,z)
c = colorbar(s)
c.set_label("Elevation (ft in 1000s)")
```



(5). 删除颜色栏

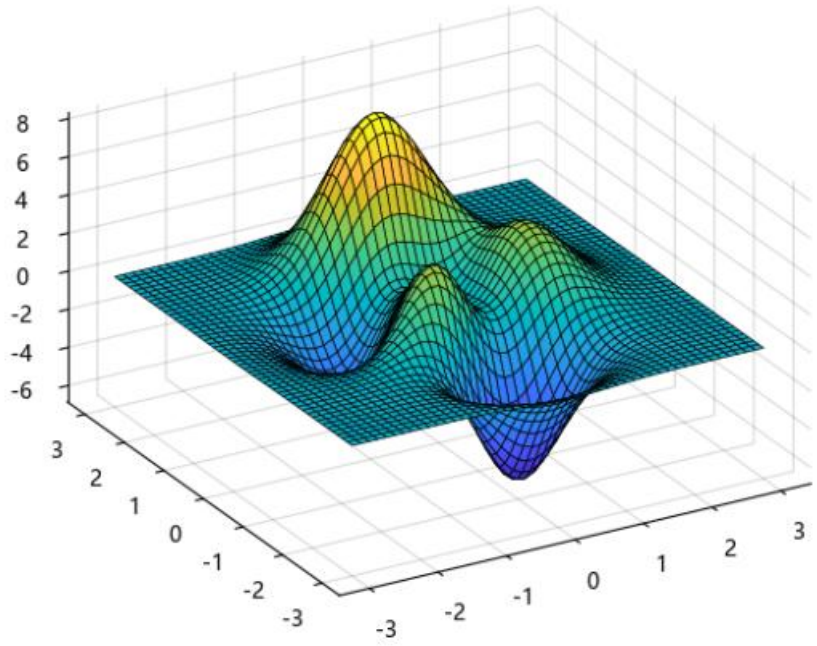
将颜色栏添加到曲面图。

```
x,y,z = peaks();
s = surf(x,y,z)
c = colorbar(s)
```



从曲面图中删除颜色栏。

```
colorbar("off")
```



rgbplot

绘制颜色图

A. 语法

```
rgbplot(map)
```

B. 说明

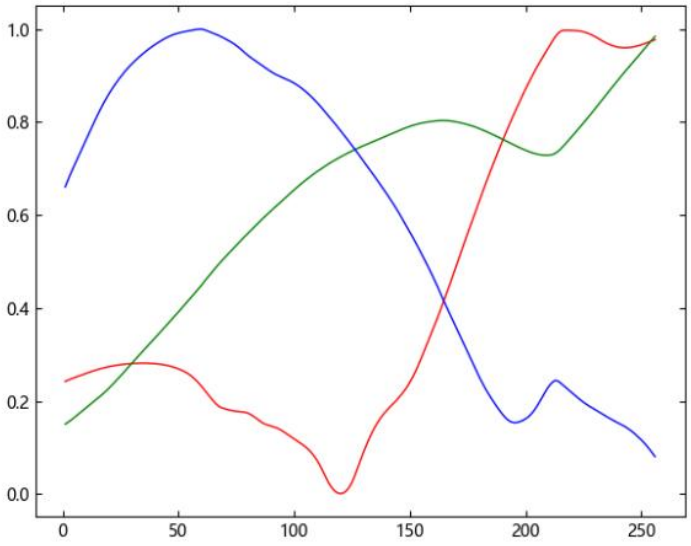
rgbplot(map) 绘制指定颜色图的红色、绿色和蓝色强度。示例。

C. 示例

(1). 绘制预定义颜色图

绘制 parula 颜色图。

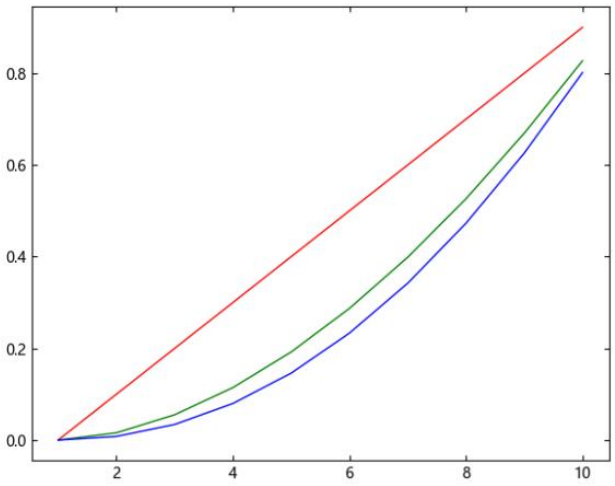
```
c = parula()
rgbplot(c)
```



(2). 绘制自定义颜色图

创建并绘制自定义颜色图。

```
r = (0:0.1:0.9)
g = r .^ 1.8
b = r .^ 2.1
mymap = [r g b]
rgbplot(mymap)
```



3.3.2.3.2 颜色空间

函数名	简介
hsv2rgb	将 HSV 颜色转换为 RGB
rgb2hsv	将 RGB 颜色转换为 HSV

hsv2rgb

将 HSV 颜色转换为 RGB

A. 语法

```
RGB = hsv2rgb(HSV)
rgbmap = hsv2rgb(hsvmap)
```

B. 说明

RGB=hsv2rgb(HSV)将 HSV 图像的色调、饱和度和明度值转换为 RGB 图像的红色、绿色和蓝色值。

rgbmap=hsv2rgb(hsvmap)将 HSV 颜色图转换为 RGB 颜色图。

C. 示例

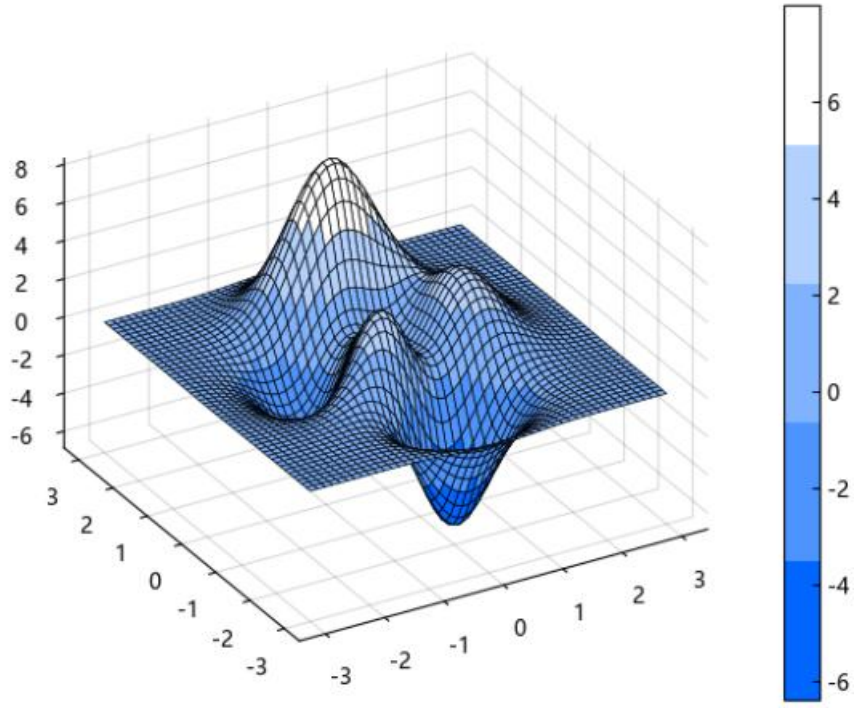
(1). 将 HSV 矩阵转换为颜色图

创建一个三列 HSV 矩阵，用它指定五个蓝色梯度。在本例中，色调和明度不变，饱和度在 1.0 和 0.0 之间变化。

```
hsv1 = [.6 1 1; .6 .7 1; .6 .5 1; .6 .3 1; .6 0 1];
```

通过调用 hsv2rgb 将 HSV 矩阵转换为颜色图。然后在曲面图中使用该颜色图。

```
rgb = hsv2rgb(hsv1);
X,Y,Z = peaks()
s = surf(X,Y,Z);
colormap(s, rgb);
colorbar(s)
```



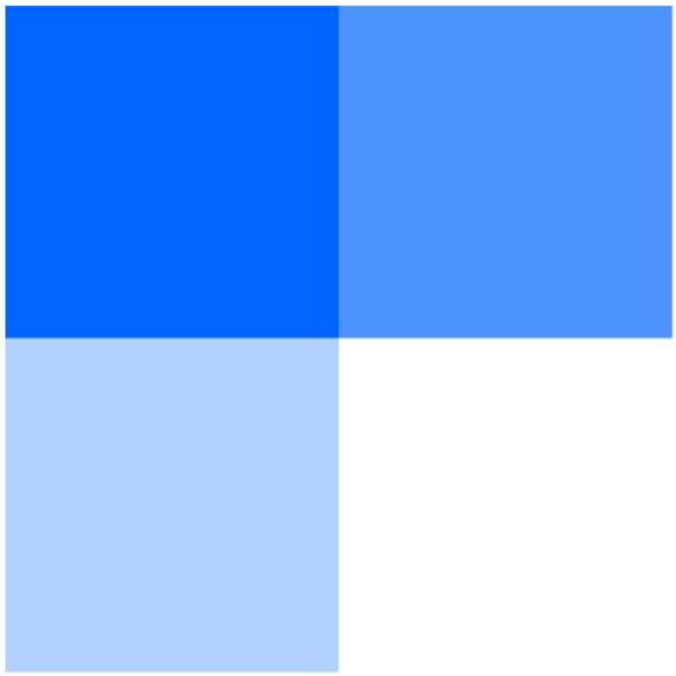
(2). 将三维 HSV 数组转换为真彩色图像

创建一个 2×2×3 的 HSV 数组，用它指定四个蓝色梯度。

```
using TyImages
hsv1 = zeros(2,2,3)
hsv1[:,:,1] = ones(2,2)*.6;
hsv1[:,:,2] = [1 .7; .3 0];
hsv1[:,:,3] = ones(2,2);
```

使用 hsv2rgb 将 HSV 数组转换为真彩色图像。然后显示图像。

```
rgb = hsv2rgb(hsv1);
image(rgb);  # 需要导入图像函数库
```



rgb2hsv

将 RGB 颜色转换为 HSV

A. 语法

```
HSV = rgb2hsv(RGB)
hsvmap = rgb2hsv(rgbmap)
```

B. 说明

HSV=rgb2hsv(RGB)将 RGB 图像的红色、绿色和蓝色值转换为 HSV 图像的色调、饱和度和明度(HSV)值。

hsvmap=rgb2hsv(rgbmap)将 RGB 颜色图转换为 HSV 颜色图。

C. 示例

(1). 将颜色图转换为 HSV

获取下采样 prism 颜色图。

```
rgb = prism(6);
```

```
rgb =
6×3 Matrix{Float64}:
 1.0      0.0  0.0
```

```
 1.0      0.5  0.0
 1.0      1.0  0.0
 0.0      1.0  0.0
 0.0      0.0  1.0
 0.666667  0.0  1.0
```

将颜色图中的 RGB 值转换为 HSV。

```
hsv1 = rgb2hsv(rgb);
```

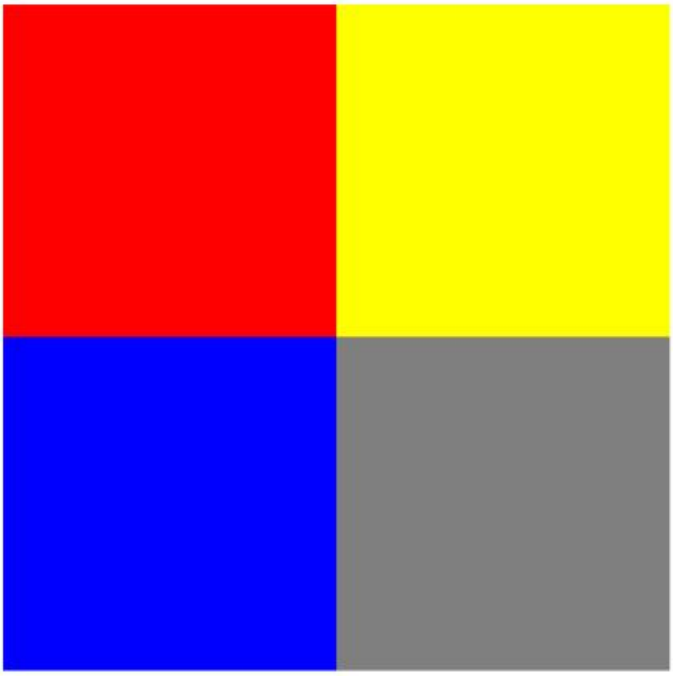
```
hsv1 =
6×3 Matrix{Float64}:
 0.0       1.0  1.0
 0.0833333  1.0  1.0
 0.166667   1.0  1.0
 0.333333   1.0  1.0
 0.666667   1.0  1.0
 0.777778   1.0  1.0
```

(2). 将真彩色图像转换为 HSV

创建一个 2×2 真彩色图像。

```
using TyImages
rgb = zeros(2,2,3)
rgb[:,:,1] = [1 1; 0 .5];
rgb[:,:,2] = [0 1; 0 .5];
rgb[:,:,3] = [0 0; 1 .5];
image(rgb);

hsv1 = rgb2hsv(rgb)
```



3.3.2.3.3 预定义颜色图

函数名	简介
parula	parula 颜色图数组
hsv	HSV 颜色图数组
hot	hot 颜色图数组
cool	冷色颜色图数组
spring	Spring 颜色图数组
summer	Summer 颜色图数组
autumn	Autumn 颜色图数组
winter	Winter 颜色图数组
gray	gray 颜色图数组
bone	bone 颜色图数组
copper	copper 颜色图数组
pink	粉色颜色图数组

lines	线条颜色图数组
jet	Jet 颜色图数组
colorcube	Colorcube 颜色图数组
prism	Prism 颜色图数组
flag	flag 颜色图数组

parula

parula 颜色图数组

A. 语法

```
c = parula()
c = parula(m)
```

B. 说明

c = parula() 以三列数组形式返回 parula 颜色图，其中包含的行数与当前图窗的颜色图相同。如果不存在图窗，则行数等于默认长度 256。数组中的每一行包含一种特定颜色的红、绿、蓝强度。强度介于 [0,1]范围内，颜色方案如下图所示。



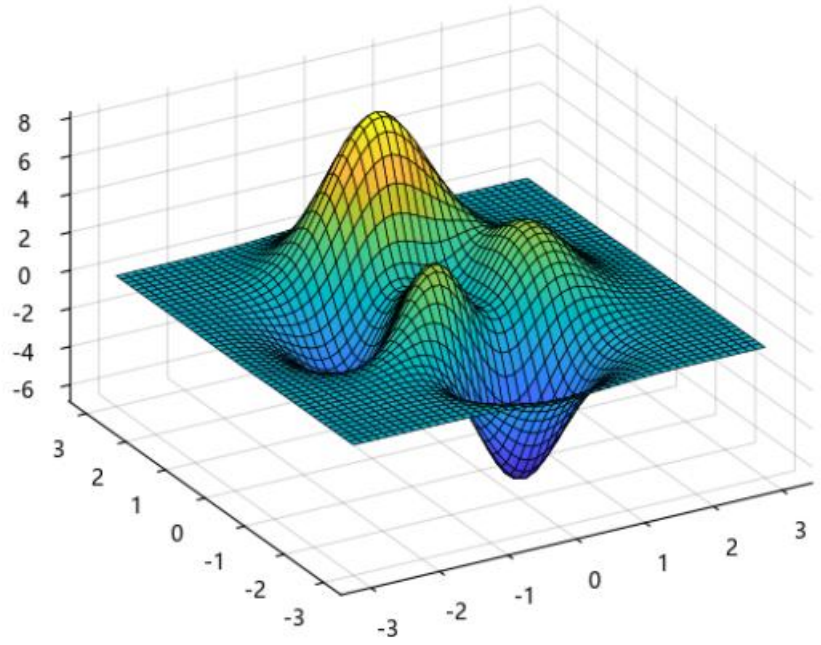
c = parula(m)返回包含 m 种颜色的颜色图。

C. 示例

(1). 反转颜色图

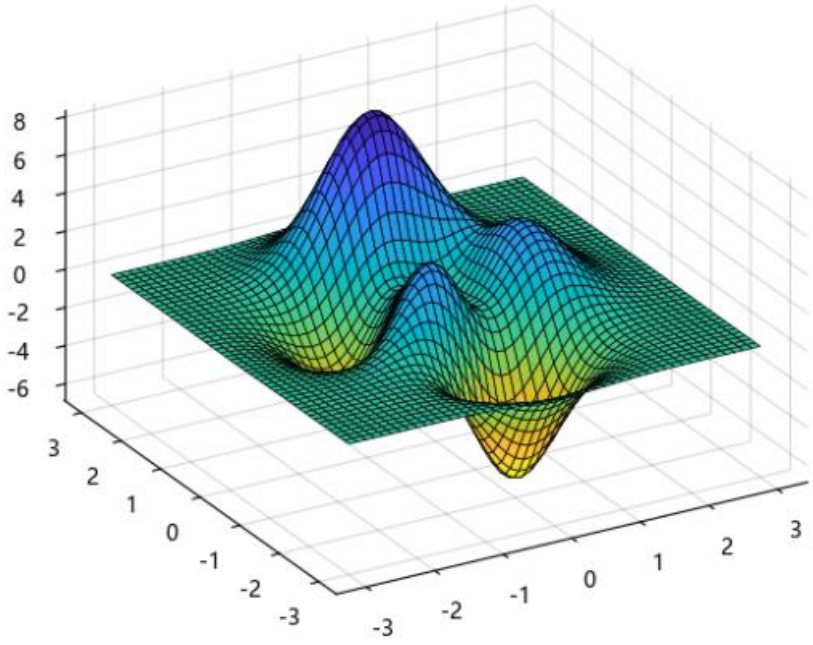
使用默认的 parula 颜色图绘制一个曲面。

```
X,Y,Z = peaks();
s = surf(X,Y,Z)
```



获取 parula 颜色图数组并反转顺序。然后将修改后的颜色图应用于该曲面。

```
c = parula();
c[:,1] = reverse(c[:,1]);
c[:,2] = reverse(c[:,2]);
c[:,3] = reverse(c[:,3]);
colormap(s, c);
```



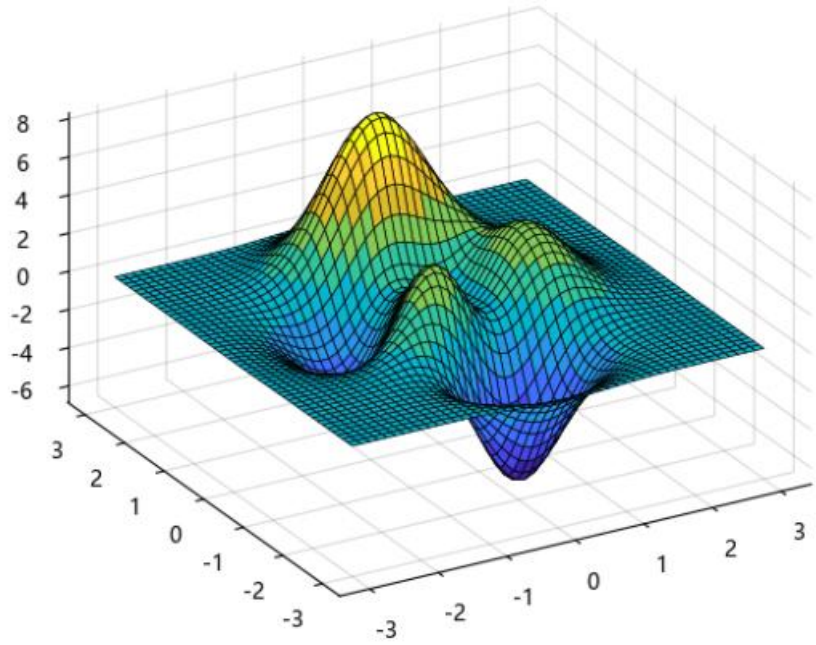
(2). 减采样 parula 颜色图

获取仅包含 parula 颜色图的十种颜色的减采样版本。

```
X,Y,Z = peaks();
```

```
s = surf(X,Y,Z);

c = parula(10)
colormap(s, c);
```



hsv

HSV 颜色图数组

A. 语法

```
c = hsv()
c = hsv(m)
```

B. 说明

c = hsv() 以三列数组形式返回 hsv 颜色图，其中包含的行数与当前图窗的颜色图相同。如果不存在图窗，则行数等于默认长度 256。数组中的每一行包含一种特定颜色的红、绿、蓝强度。强度介于 [0,1]范围内，颜色方案如下图所示。



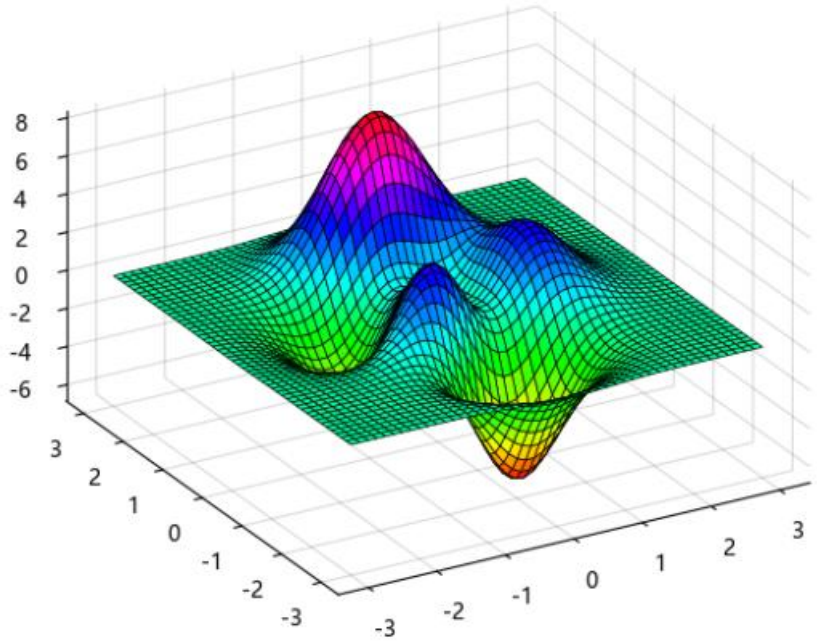
c = hsv(m)返回包含 m 种颜色的颜色图。

C. 示例

(1). 反转颜色图

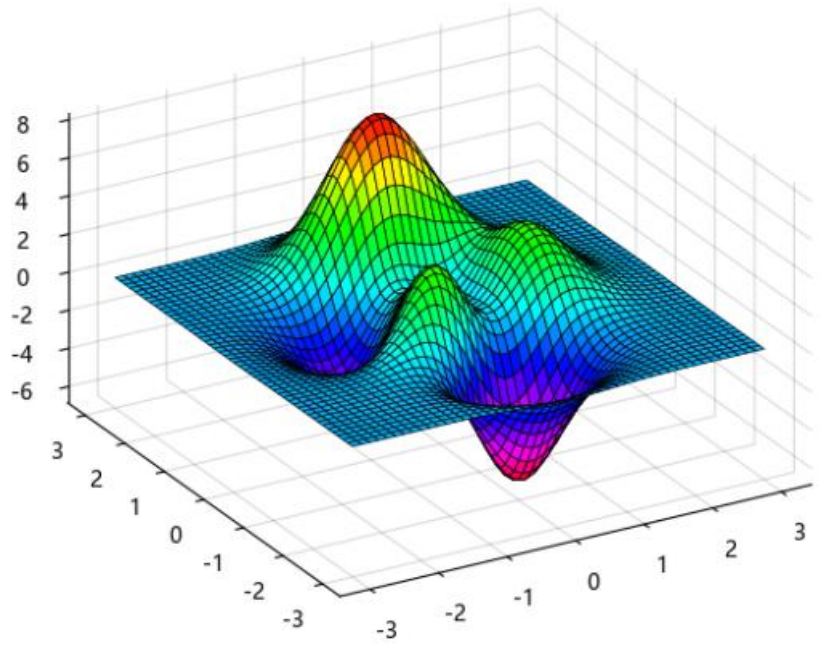
绘制一个曲面并指定 hsv 颜色图。

```
X,Y,Z = peaks();
s = surf(X,Y,Z);
colormap(s, "hsv")
```



获取 hsv 颜色图数组并反转顺序。然后将修改后的颜色图应用于该曲面。

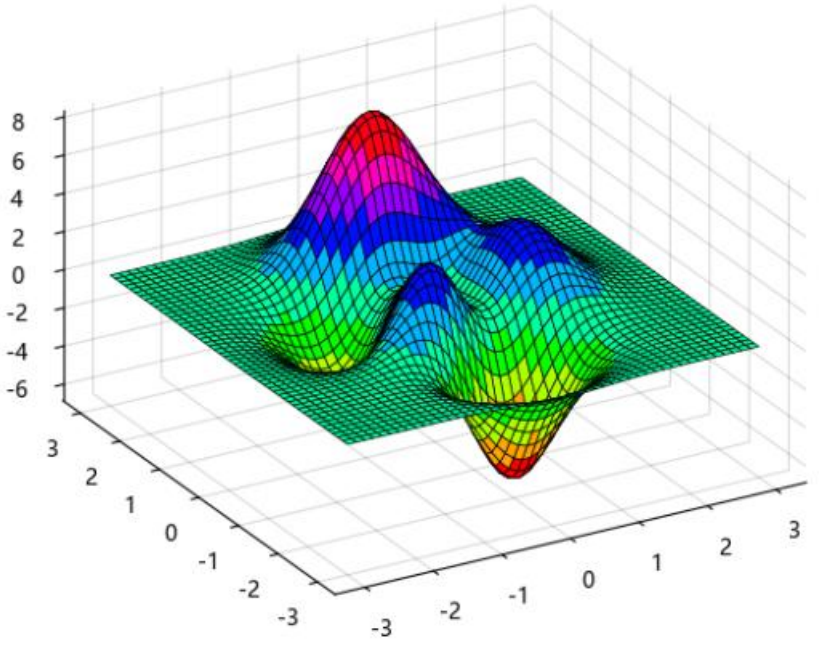
```
c = hsv();
c[:,1] = reverse(c[:,1]);
c[:,2] = reverse(c[:,2]);
c[:,3] = reverse(c[:,3]);
colormap(s, c);
```



(2). 减采样 hsv 颜色图

获取仅包含 hsv 颜色图的十种颜色的减采样版本。

```
X,Y,Z = peaks();
s = surf(X,Y,Z);
c = hsv(10);
colormap(s, c);
```



hot

hot 颜色图数组

A. 语法

```
c = hot()
c = hot(m)
```

B. 说明

c = hot() 以三列数组形式返回 hot 颜色图，其中包含的行数与当前图窗的颜色图相同。如果不存在图窗，则行数等于默认长度 256。数组中的每一行包含一种特定颜色的红、绿、蓝强度。强度介于 [0,1]范围内，颜色方案如下图所示 。



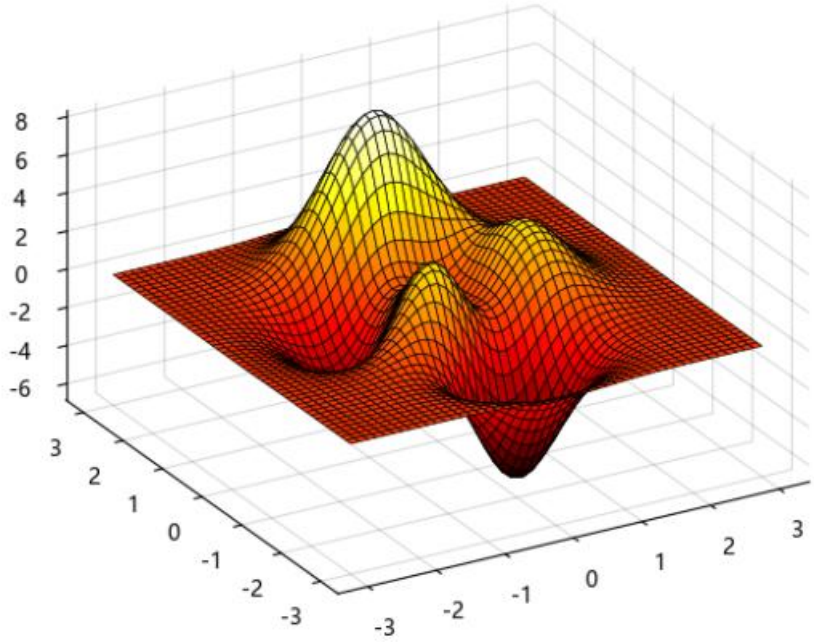
c = hot(m)返回包含 m 种颜色的颜色图。

C. 示例

(1). 反转颜色图

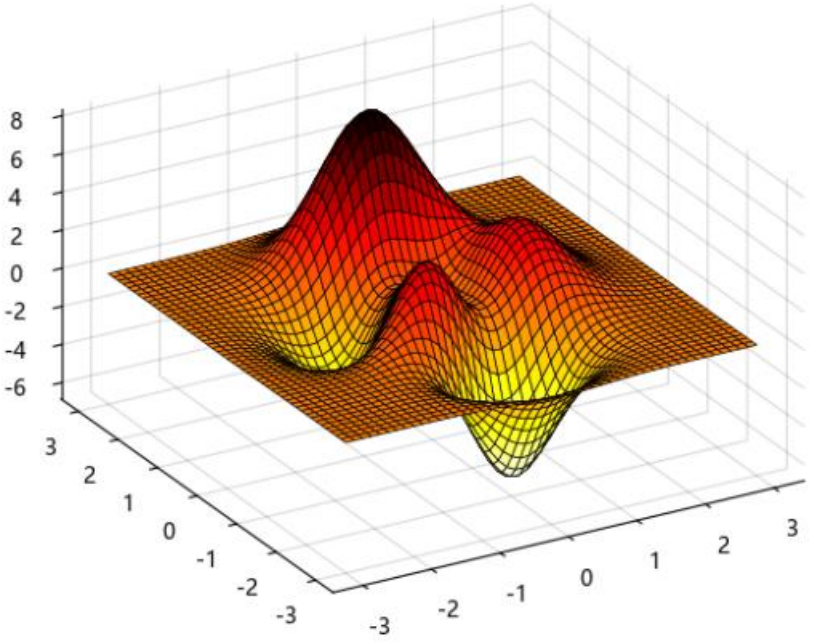
绘制一个曲面并指定 hot 颜色图。

```
X,Y,Z = peaks();
s = surf(X,Y,Z);
colormap(s, "hot")
```



获取 hot 颜色图数组并反转顺序。然后将修改后的颜色图应用于该曲面。

```
c = hot();
c[:,1] = reverse(c[:,1]);
c[:,2] = reverse(c[:,2]);
c[:,3] = reverse(c[:,3]);
colormap(s, c);
```



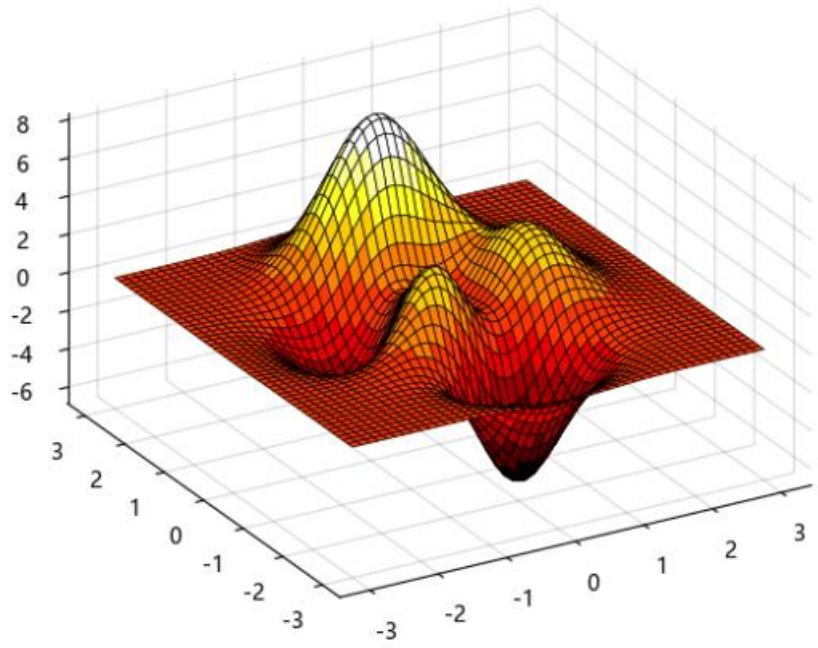
(2). 减采样 hot 颜色图

获取仅包含 hot 颜色图的二十种颜色的减采样版本。

```
X,Y,Z = peaks();
```

```
s = surf(X,Y,Z);

c = hot(20);
colormap(s, c);
```



cool

冷色颜色图数组

A. 语法

```
c = cool()
c = cool(m)
```

B. 说明

c = cool() 以三列数组形式返回冷色颜色图，该数组包含的行数与当前图窗颜色图的行数相同。如果不存在图窗，则行数等于默认长度 256。数组中的每一行包含一种特定颜色的红、绿、蓝强度。强度介于 [0,1]范围内，颜色方案如下图所示。



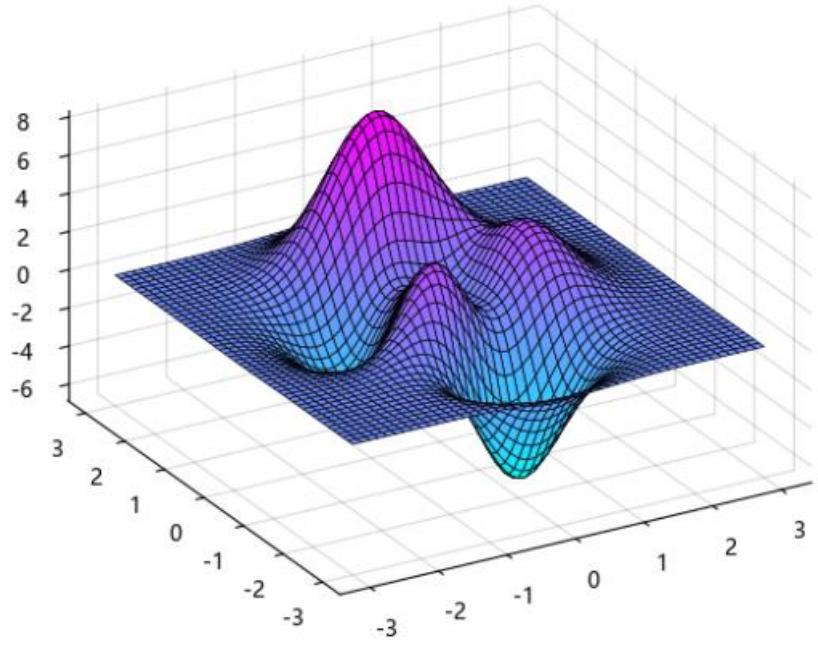
c = cool(m)返回包含 m 种颜色的颜色图。

C. 示例

(1). 反转颜色图

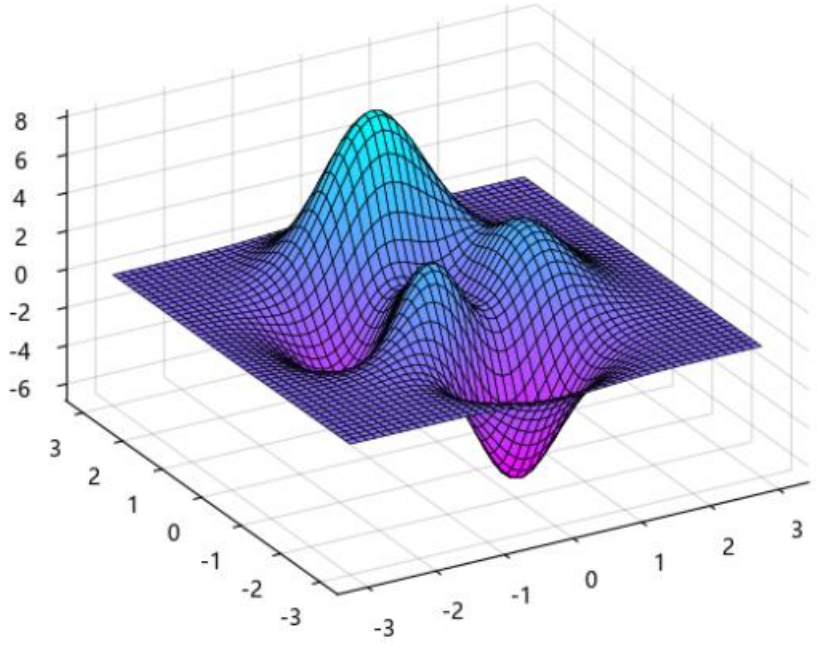
绘制曲面图，并分配冷色颜色图。

```
X,Y,Z = peaks();
s = surf(X,Y,Z);
colormap(s, "cool")
```



获取冷色颜色图数组，并反转顺序。然后将修改后的颜色图应用于该曲面。

```
c = cool();
c[:,1] = reverse(c[:,1]);
c[:,2] = reverse(c[:,2]);
c[:,3] = reverse(c[:,3]);
colormap(s, c);
```

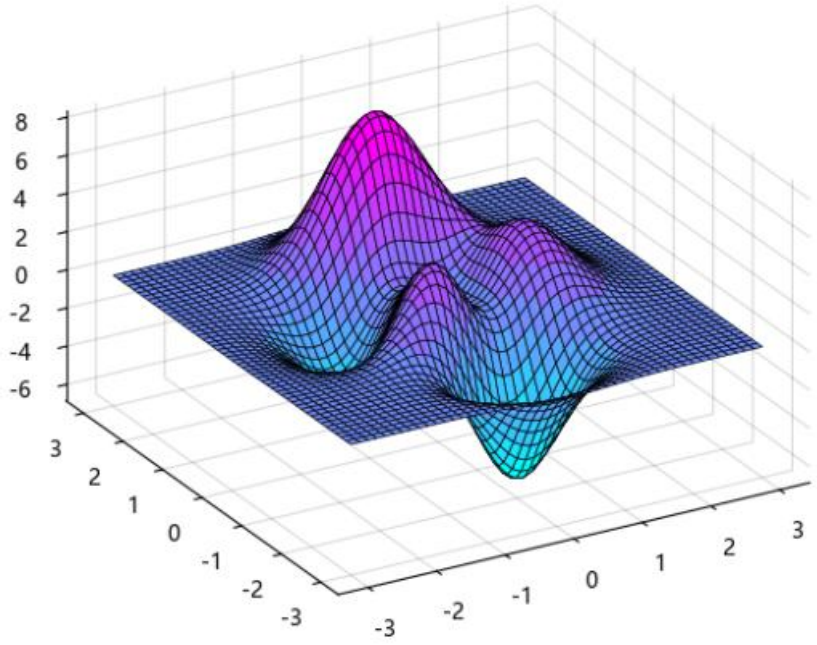


(2). 对冷色颜色图进行减采样

获取仅包含冷色颜色图的十种颜色的减采样版本。

```
X,Y,Z = peaks();
s = surf(X,Y,Z);

c = cool(10);
colormap(s, c);
```



spring

Spring 颜色图数组

A. 语法

```
c = spring()
c = spring(m)
```

B. 说明

c = spring() 以三列数组形式返回 spring 颜色图，该数组包含的行数与当前图窗的颜色图相同。如果不存在图窗，则行数等于默认长度 256。数组中的每一行包含一种特定颜色的红、绿、蓝强度。强度介于 [0,1]范围内，颜色方案如下图所示。



c = spring(m)返回包含 m 种颜色的颜色图。

C. 示例

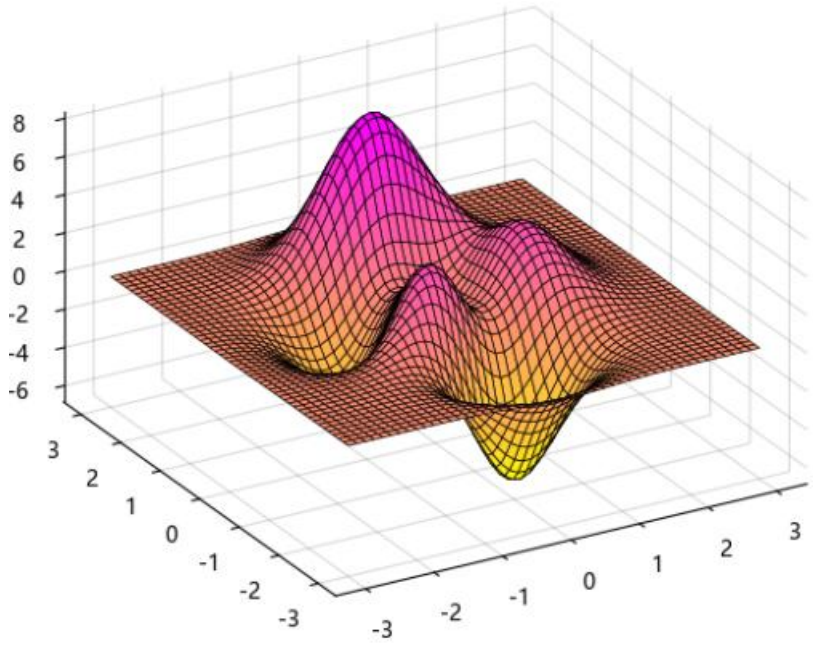
(1). 反转颜色图

绘制曲面图，并分配 spring 颜色图。

```
X,Y,Z = peaks();
s = surf(X,Y,Z);
colormap(s, "spring")
```

获取 spring 颜色图数组，并反转顺序。然后将修改后的颜色图应用于该曲面。

```
c = spring();
c[:,1] = reverse(c[:,1]);
c[:,2] = reverse(c[:,2]);
c[:,3] = reverse(c[:,3]);
colormap(s, c);
```

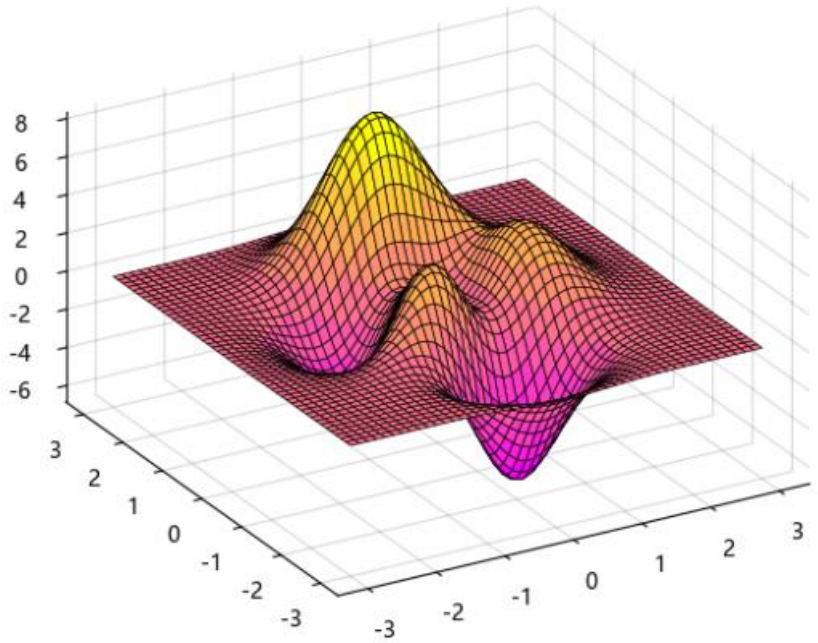


(2). 对 spring 颜色图进行缩减像素采样

获取 spring 颜色图的缩减像素采样版本（仅包含 10 种颜色）。

```
X,Y,Z = peaks();
s = surf(X,Y,Z);

c = spring(10);
colormap(s, c);
```



summer

Summer 颜色图数组

A. 语法

```
c = summer()
c = summer(m)
```

B. 说明

c = summer() 以三列数组形式返回 summer 颜色图，该数组包含的行数与当前图窗的颜色图相同。如果不存在图窗，则行数等于默认长度 256。数组中的每一行包含一种特定颜色的红、绿、蓝强度。强度介于 [0,1]范围内，颜色方案如下图所示。



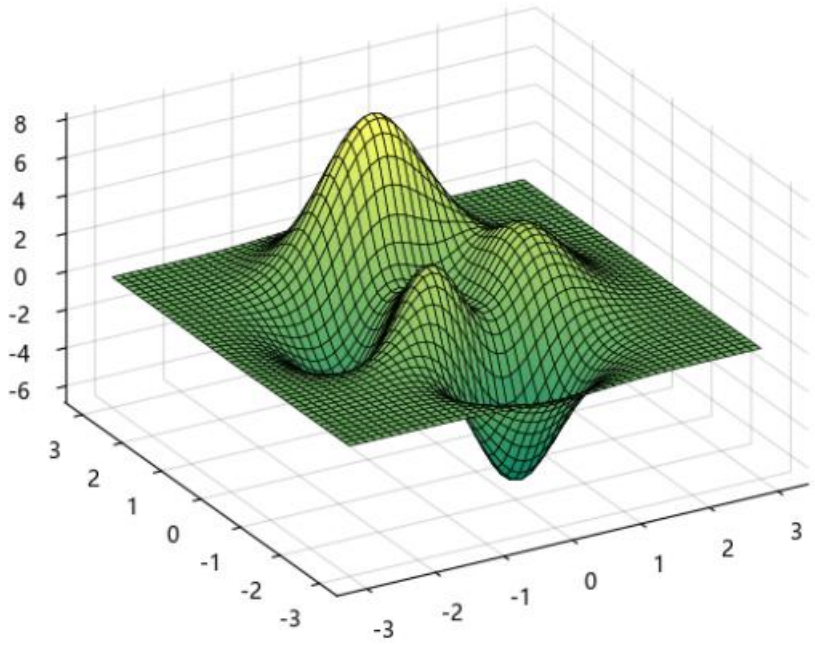
c = summer(m)返回包含 m 种颜色的颜色图。

C. 示例

(1). 反转颜色图

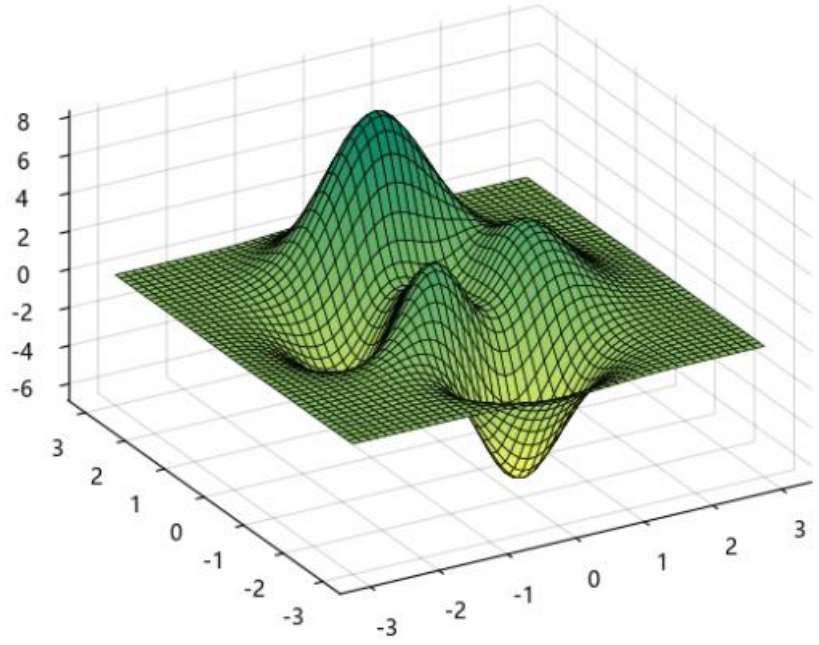
绘制曲面图，并分配 summer 颜色图。

```
X,Y,Z = peaks();
s = surf(X,Y,Z);
colormap(s, "summer")
```



获取 summer 颜色图数组，并反转顺序。然后将修改后的颜色图应用于该曲面。

```
c = summer();
c[:,1] = reverse(c[:,1]);
c[:,2] = reverse(c[:,2]);
c[:,3] = reverse(c[:,3]);
colormap(s, c);
```

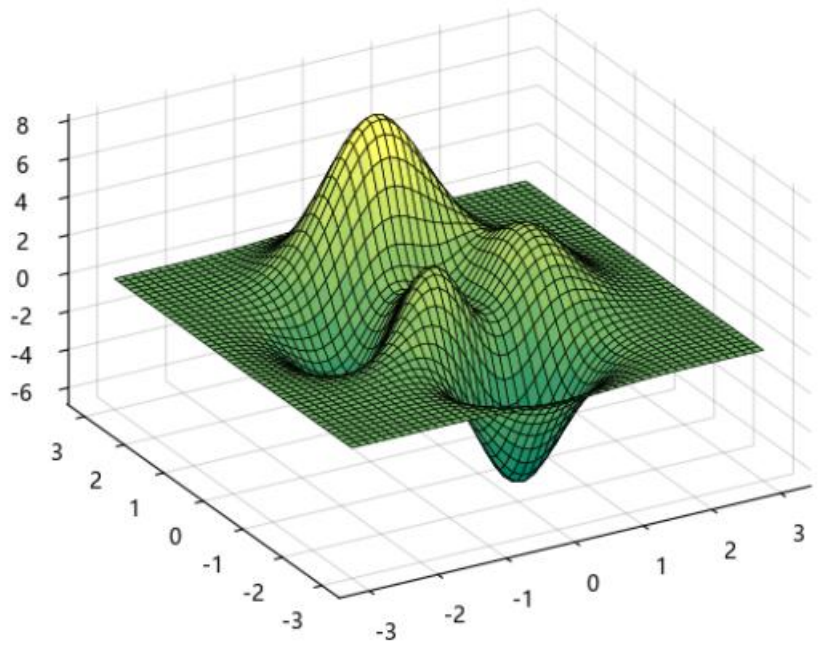


(2). 对 summer 颜色图进行减采样

获取 summer 颜色图的减采样版本（仅包含 10 种颜色）。

```
X,Y,Z = peaks();
s = surf(X,Y,Z);

c = summer(10);
colormap(s, c);
```



autumn

autumn 颜色图数组

A. 语法

```
c = autumn()
c = autumn(m)
```

B. 说明

c = autumn() 以三列数组形式返回 autumn 颜色图，其中包含的行数与当前图窗的颜色图相同。如果不存在图窗，则行数等于默认长度 256。数组中的每一行包含一种特定颜色的红、绿、蓝强度。强度介于 [0,1]范围内，颜色方案如下图所示。



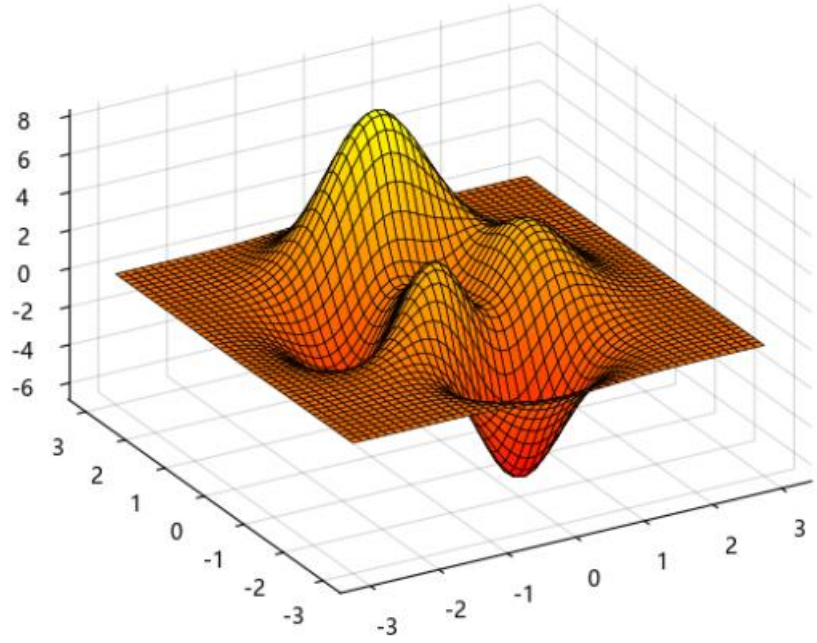
c = autumn(m)返回包含 m 种颜色的颜色图。

C. 示例

(1). 反转颜色图

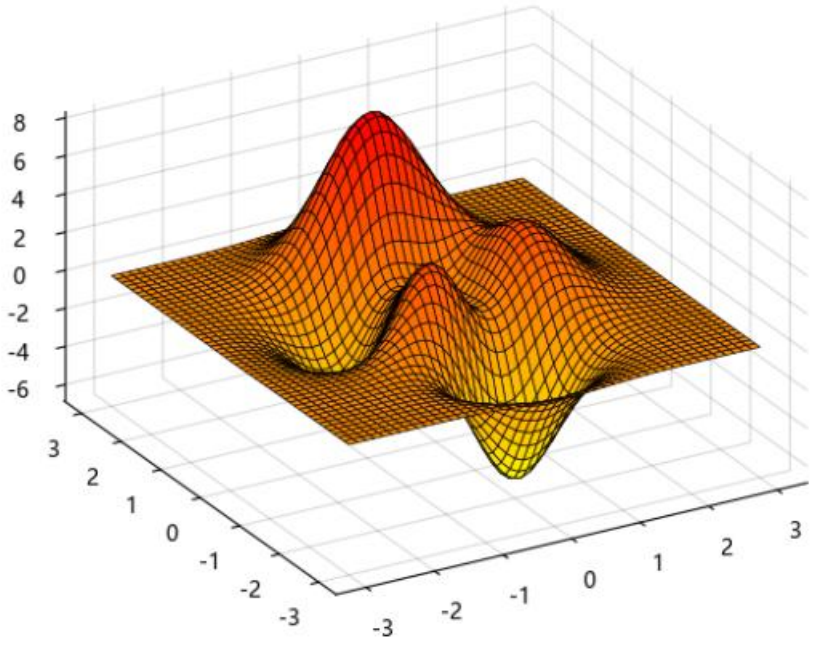
绘制一个曲面并指定 autumn 颜色图。

```
X,Y,Z = peaks();
s = surf(X,Y,Z);
colormap(s, "autumn");
```



获取 autumn 颜色图数组并反转顺序。然后将修改后的颜色图应用于该曲面。

```
c = autumn();
c[:,1] = reverse(c[:,1]);
c[:,2] = reverse(c[:,2]);
c[:,3] = reverse(c[:,3]);
colormap(s, c);
```

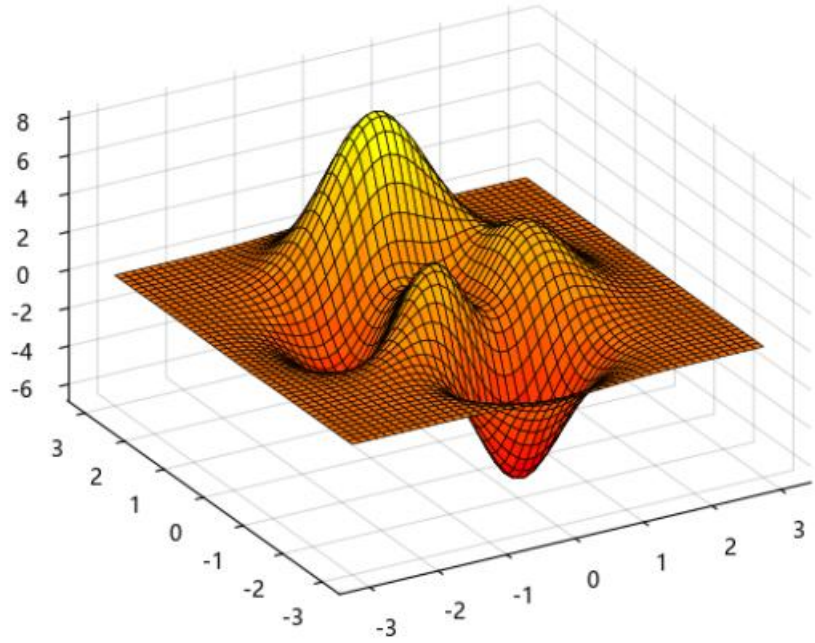


(2). 减下采样 autumn 颜色图

获取仅包含 autumn 颜色图的十种颜色的减采样版本。

```
X,Y,Z = peaks();
s = surf(X,Y,Z);

c = autumn(10);
colormap(s, c);
```



winter

Winter 颜色图数组

A. 语法

```
c = winter()
c = winter(m)
```

B. 说明

c = winter() 以三列数组形式返回 winter 颜色图，该数组包含的行数与当前图窗的颜色图相同。如果不存在图窗，则行数等于默认长度 256。数组中的每一行包含一种特定颜色的红、绿、蓝强度。强度介于 [0,1]范围内，颜色方案如下图所示。



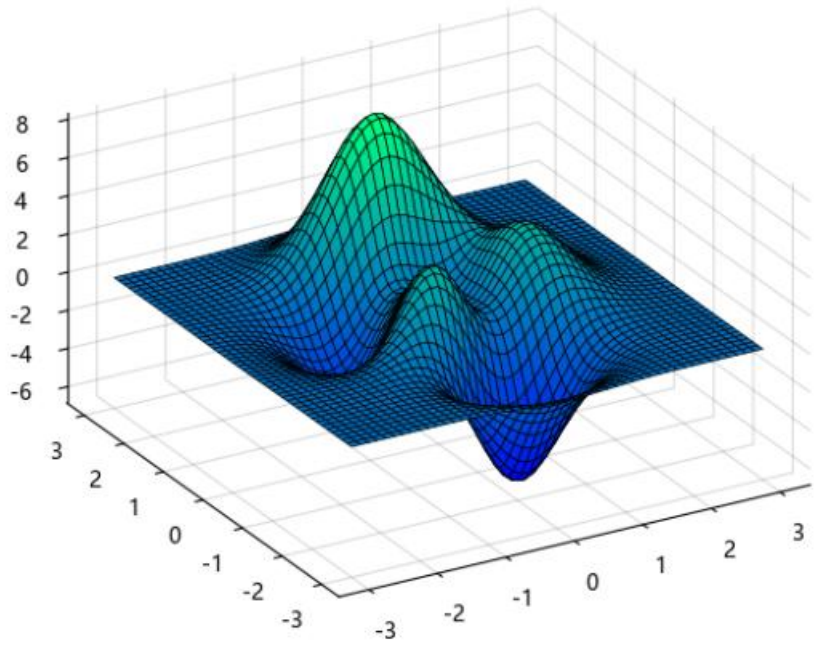
c = winter(m)返回包含 m 种颜色的颜色图。

C. 示例

(1). 反转颜色图

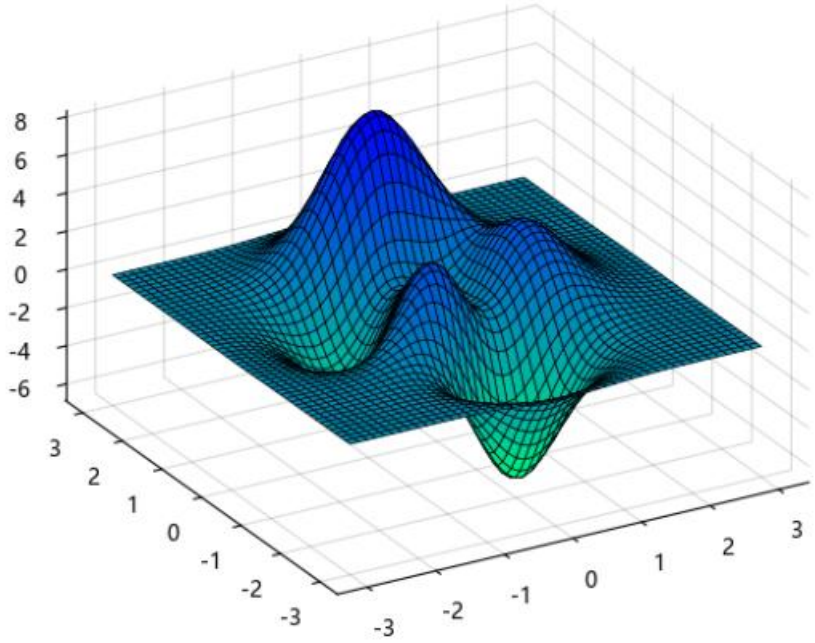
绘制曲面图，并分配 winter 颜色图。

```
X,Y,Z = peaks();
s = surf(X,Y,Z);
colormap(s, "winter")
```



获取 winter 颜色图数组，并反转顺序。然后将修改后的颜色图应用于该曲面。

```
c = winter();
c[:,1] = reverse(c[:,1]);
c[:,2] = reverse(c[:,2]);
c[:,3] = reverse(c[:,3]);
colormap(s, c);
```

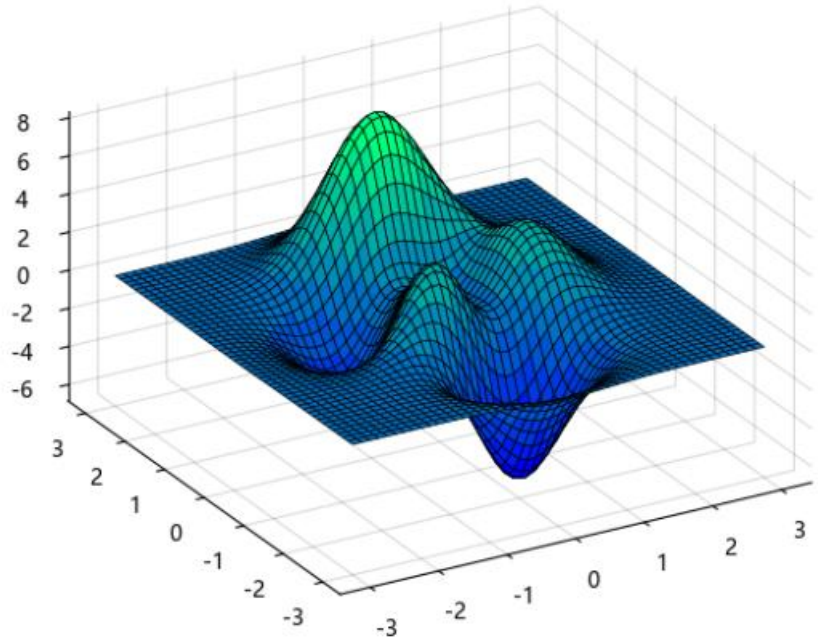


(2). 对 winter 颜色图进行减采样

获取仅包含 winter 颜色图的十种颜色的减采样版本。

```
X,Y,Z = peaks();
s = surf(X,Y,Z);

c = winter(10);
colormap(s, c);
```



gray

gray 颜色图数组

A. 语法

```
c = gray()
c = gray(m)
```

B. 说明

c = gray() 以三列数组形式返回 gray 颜色图，其中包含的行数与当前图窗的颜色图相同。如果不存在图窗，则行数等于默认长度 256。数组中的每一行包含一种特定颜色的红、绿、蓝强度。强度介于 [0,1]范围内，颜色方案如下图所示。



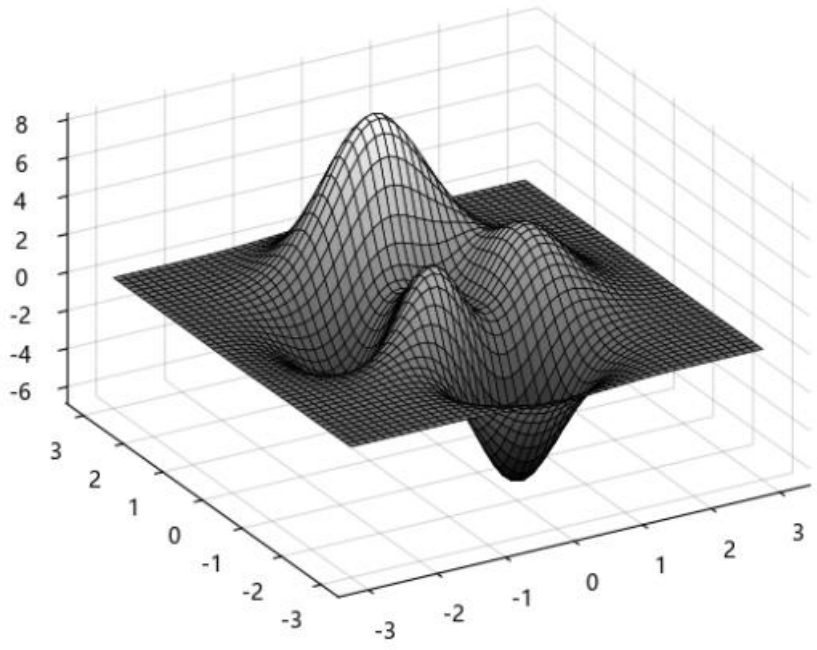
c = gray(m)返回包含 m 种颜色的颜色图。

C. 示例

(1). 反转颜色图

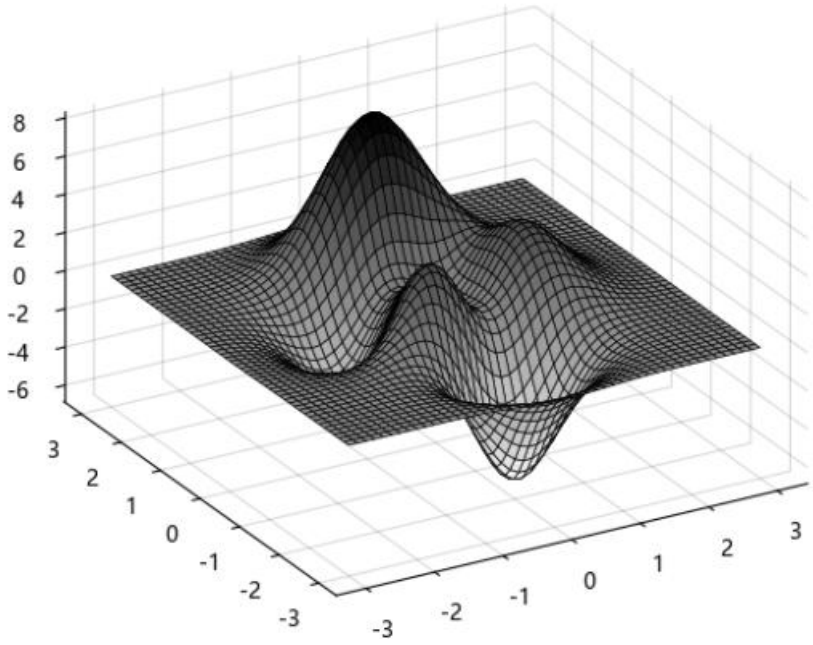
绘制一个曲面并指定 gray 颜色图。

```
X,Y,Z = peaks();
s = surf(X,Y,Z);
colormap(s, "gray")
```



获取 gray 颜色图数组并反转顺序。然后将修改后的颜色图应用于该曲面。

```
c = gray();
c[:,1] = reverse(c[:,1]);
c[:,2] = reverse(c[:,2]);
c[:,3] = reverse(c[:,3]);
colormap(s, c);
```

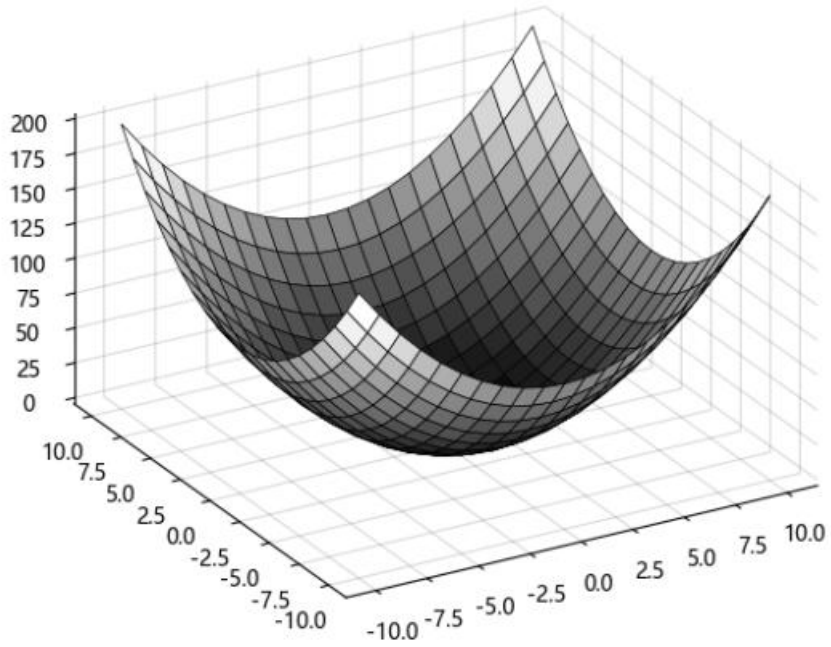


(2). 减采样 gray 颜色图

获取仅包含 gray 颜色图的二十种颜色的减采样版本。

```
c = gray(20);
X, Y = meshgrid2(-10:1:10, -10:1:10);
```

```
Z = X .^ 2 + Y .^ 2;
s = surf(X, Y, Z);
colormap(s, c);
```



bone

bone 颜色图数组

A. 语法

```
c = bone()
c = bone(m)
```

B. 说明

c = bone() 以三列数组形式返回 bone 颜色图，其中包含的行数与当前图窗的颜色图相同。如果不存在图窗，则行数等于默认长度 256。数组中的每一行包含一种特定颜色的红、绿、蓝强度。强度介于 [0,1]范围内，颜色方案如下图所示。



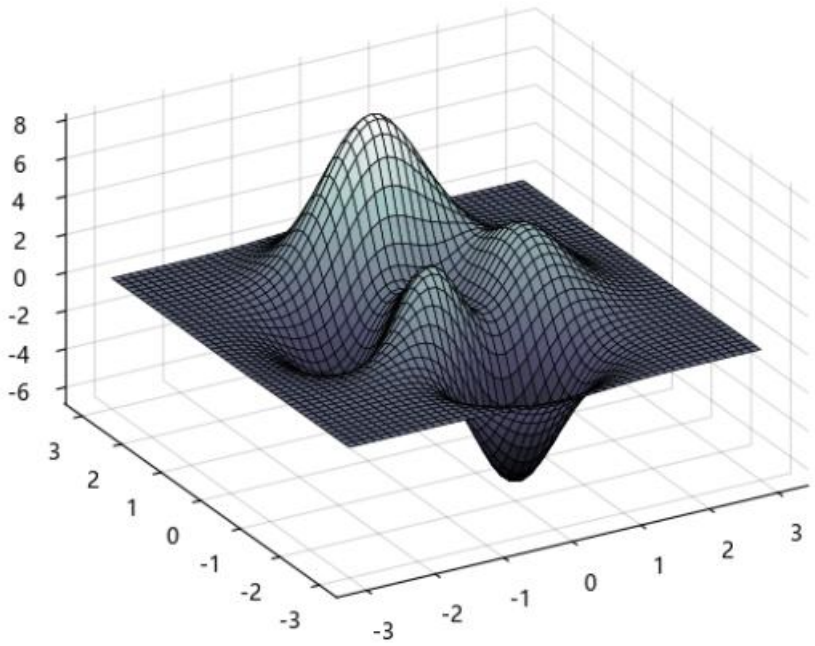
c = bone(m)返回包含 m 种颜色的颜色图。

C. 示例

(1). 反转颜色图

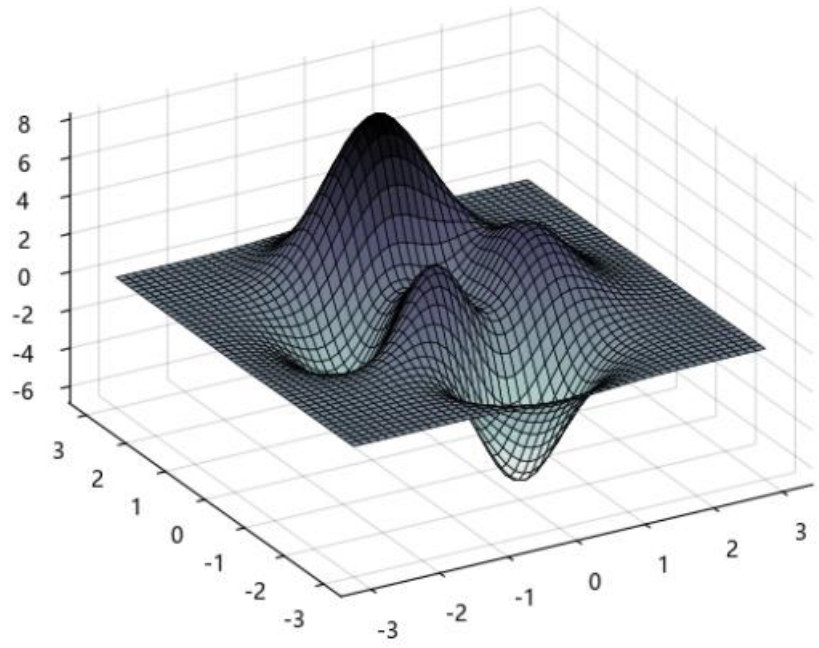
绘制一个曲面并指定 bone 颜色图。

```
X,Y,Z = peaks();
s = surf(X,Y,Z);
colormap(s, "bone")
```



获取 bone 颜色图数组并反转顺序。然后将修改后的颜色图应用于该曲面。

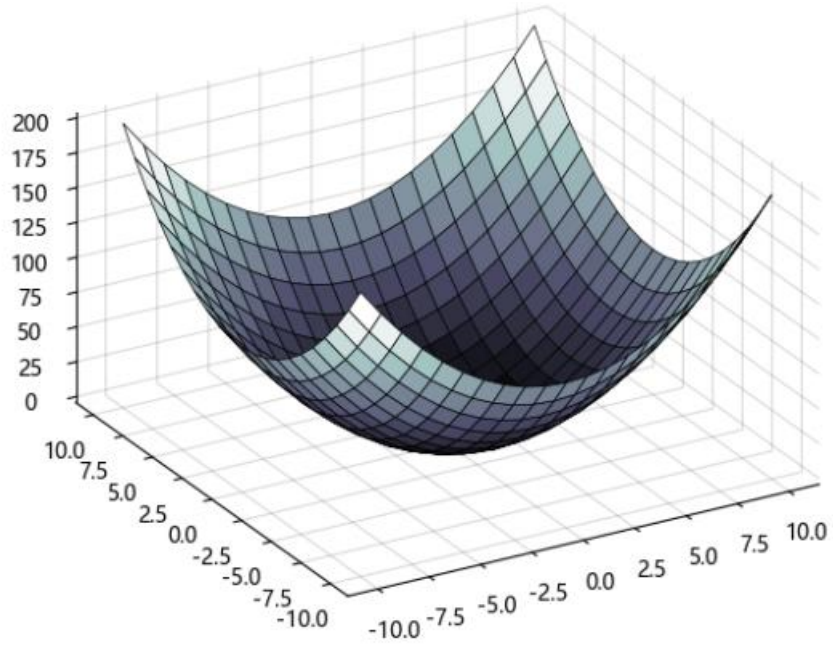
```
c = bone();
c[:,1] = reverse(c[:,1]);
c[:,2] = reverse(c[:,2]);
c[:,3] = reverse(c[:,3]);
colormap(s, c);
```



(2). 减采样 bone 颜色图

获取仅包含 bone 颜色图的二十种颜色的减采样版本。

```
c = bone(20);
X, Y = meshgrid2(-10:1:10, -10:1:10);
Z = X .^ 2 + Y .^ 2;
s = surf(X, Y, Z);
colormap(s, c);
```



copper

copper 颜色图数组

A. 语法

```
c = copper()
c = copper(m)
```

B. 说明

c = copper() 以三列数组形式返回 copper 颜色图，其中包含的行数与当前图窗的颜色图相同。如果不存在图窗，则行数等于默认长度 256。数组中的每一行包含一种特定颜色的红、绿、蓝强度。强度介于 [0,1]范围内，颜色方案如下图所示。



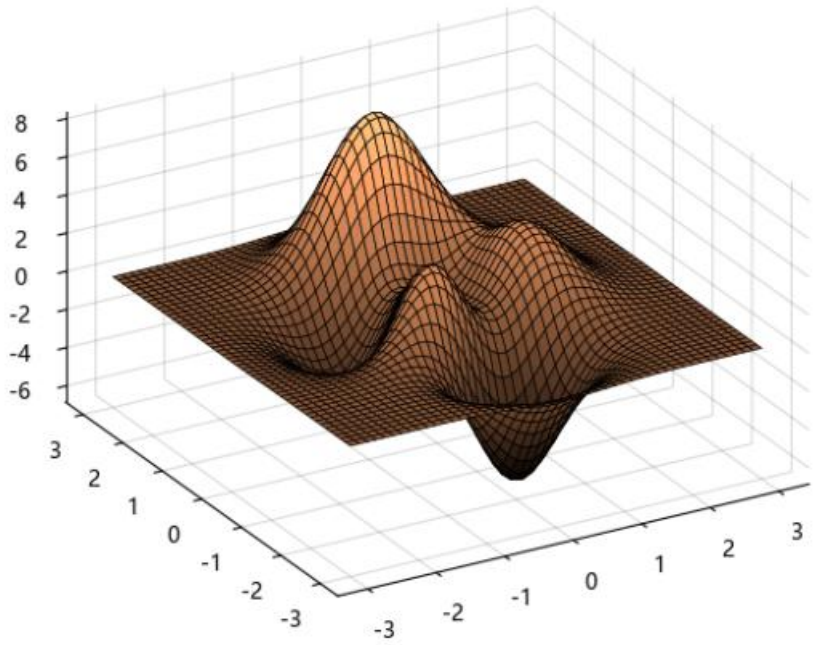
c = copper(m)返回包含 m 种颜色的颜色图。

C. 示例

(1). 反转颜色图

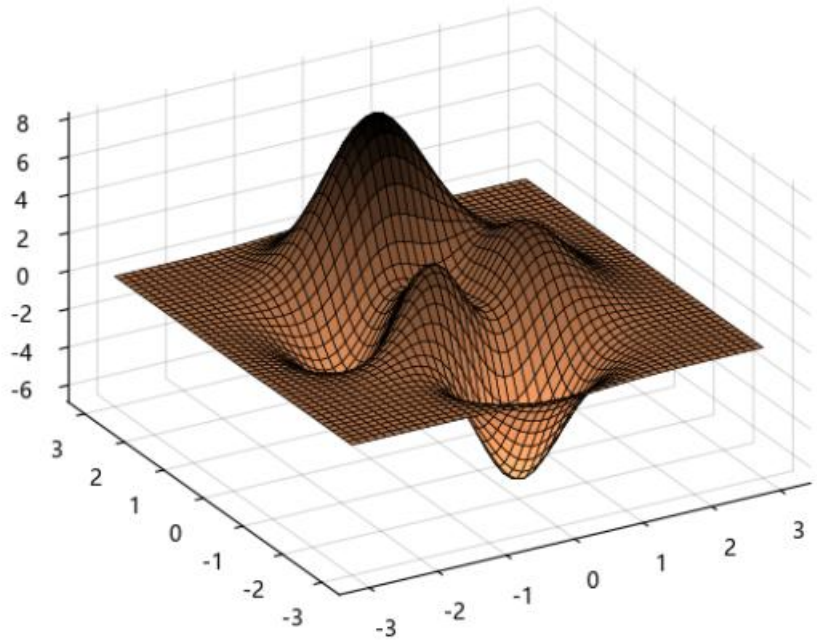
绘制一个曲面并指定 copper 颜色图。

```
X,Y,Z = peaks();
s = surf(X,Y,Z);
colormap(s, "copper")
```



获取 copper 颜色图数组并反转顺序。然后将修改后的颜色图应用于该曲面。

```
c = copper();
c[:,1] = reverse(c[:,1]);
c[:,2] = reverse(c[:,2]);
c[:,3] = reverse(c[:,3]);
colormap(s, c);
```



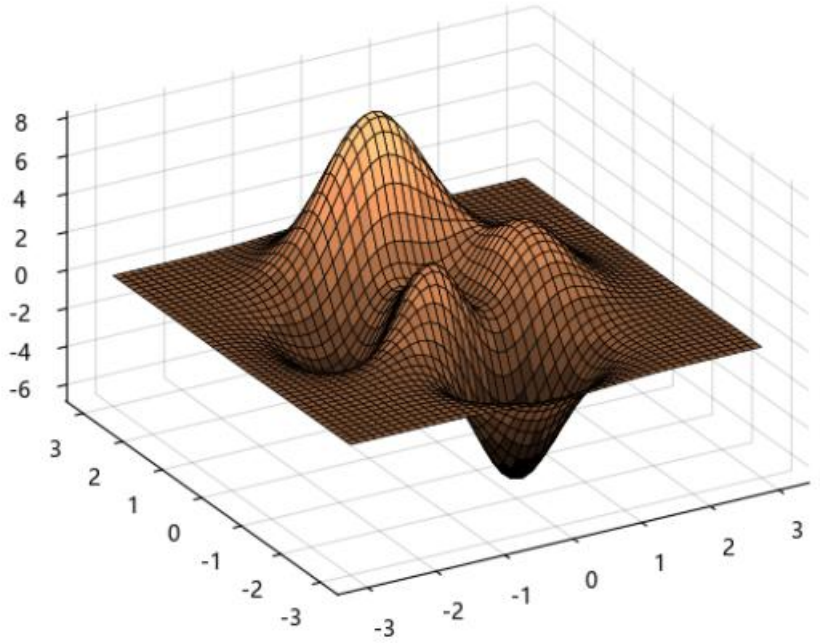
(2). 减采样 copper 颜色图

获取仅包含 copper 颜色图的十种颜色的减采样版本。

```
X,Y,Z = peaks();
```

```
s = surf(X,Y,Z);

c = copper(10);
colormap(s, c);
```



pink

粉色颜色图数组

A. 语法

```
c = pink()
c = pink(m)
```

B. 说明

c = pink() 以三列数组形式返回粉色颜色图，该数组包含的行数与当前图窗的颜色图相同。如果不存在图窗，则行数等于默认长度 256。数组中的每一行包含一种特定颜色的红、绿、蓝强度。强度介于 [0,1]范围内，颜色方案如下图所示。



c = pink(m)返回包含 m 种颜色的颜色图。

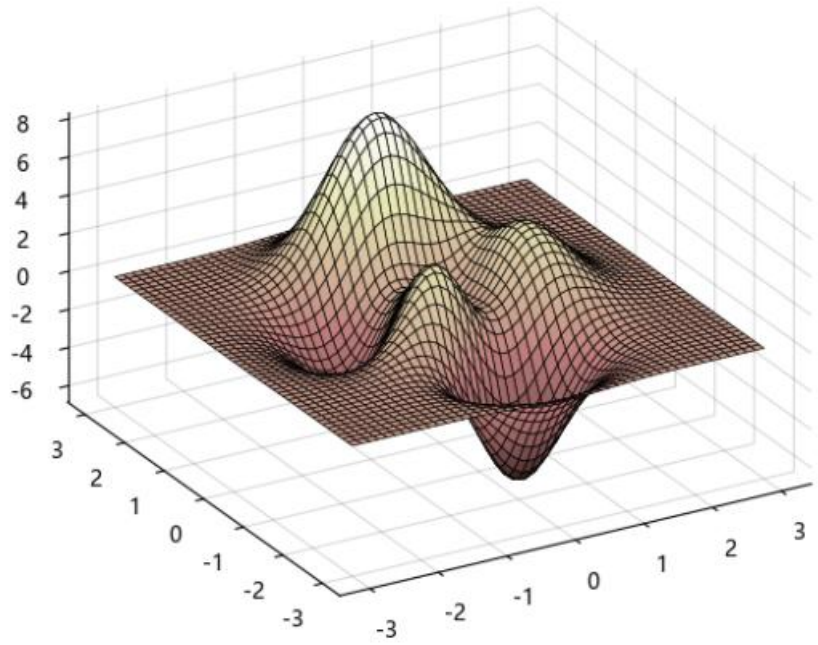
C. 示例

(1). 反转颜色图

绘制曲面图，并分配粉色颜色图。

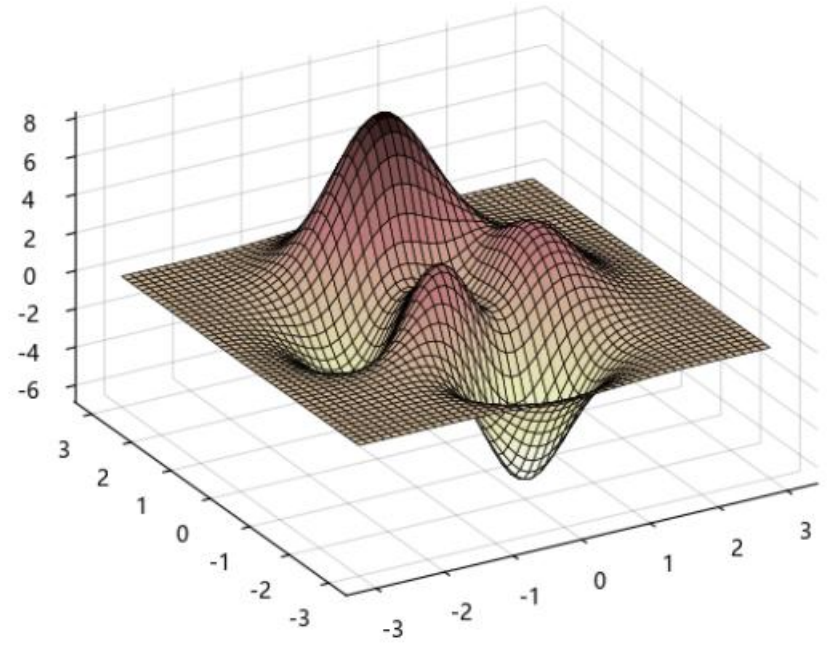
```
X,Y,Z = peaks();
s = surf(X,Y,Z)

colormap(s, "pink")
```



获取粉色颜色图数组，并反转顺序。然后将修改后的颜色图应用于该曲面。

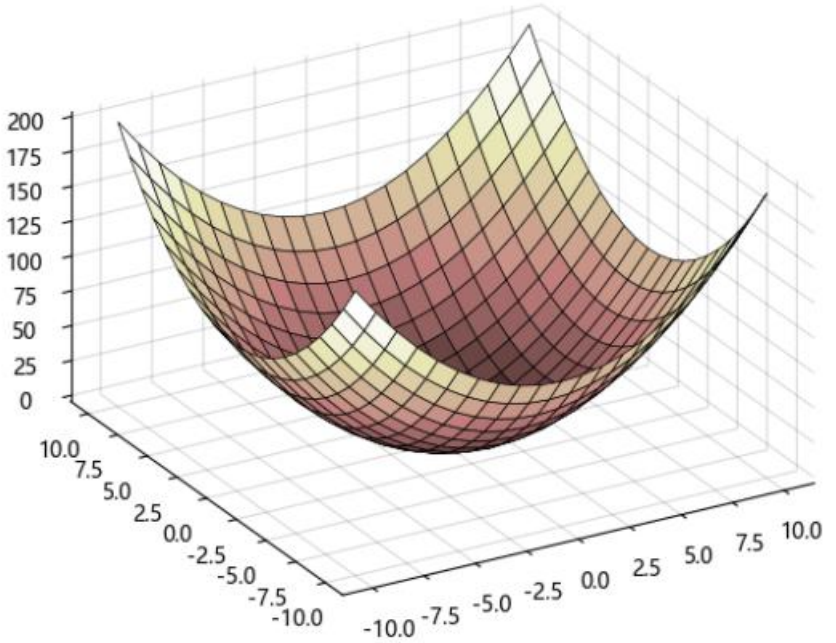
```
c = pink();
c[:,1] = reverse(c[:,1]);
c[:,2] = reverse(c[:,2]);
c[:,3] = reverse(c[:,3]);
colormap(s, c);
```



(2). 对粉色颜色图进行缩减像素采样

获取粉色颜色图的缩减像素采样版本（仅包含 20 种颜色）。

```
c = pink(20);
X, Y = meshgrid2(-10:1:10, -10:1:10);
Z = X .^ 2 + Y .^ 2;
s = surf(X, Y, Z);
colormap(s, c);
```



lines

线条颜色图数组

A. 语法

```
c = lines()
c = lines(m)
```

B. 说明

c = lines() 以一个三列数组的形式返回线条颜色图，该数组包含的行数与当前图窗的颜色图相同。如果不存在图窗，则数组包含 256 行。数组中的每一行包含一种特定颜色的红、绿、蓝强度。强度在 [0,1] 范围内，颜色方案与 Axes 的默认 ColorOrder 属性匹配。因此，您可以使用此颜色图使图形颜色与 plot 函数的输出匹配。



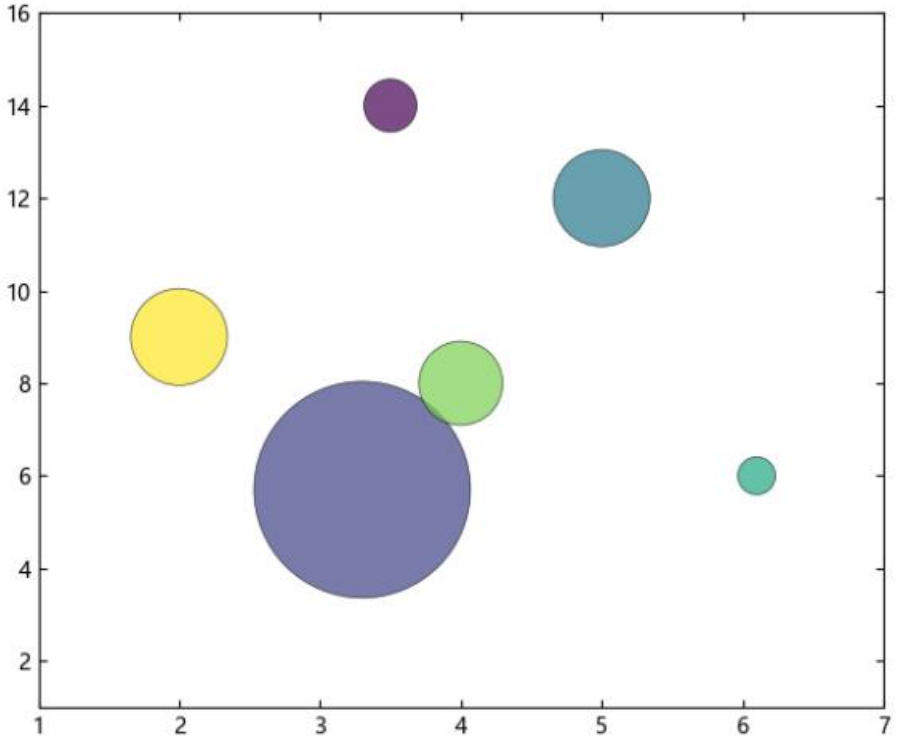
c = lines(m)返回包含 m 种颜色的颜色图。

C. 示例

(1). 对线条颜色图进行缩减像素采样

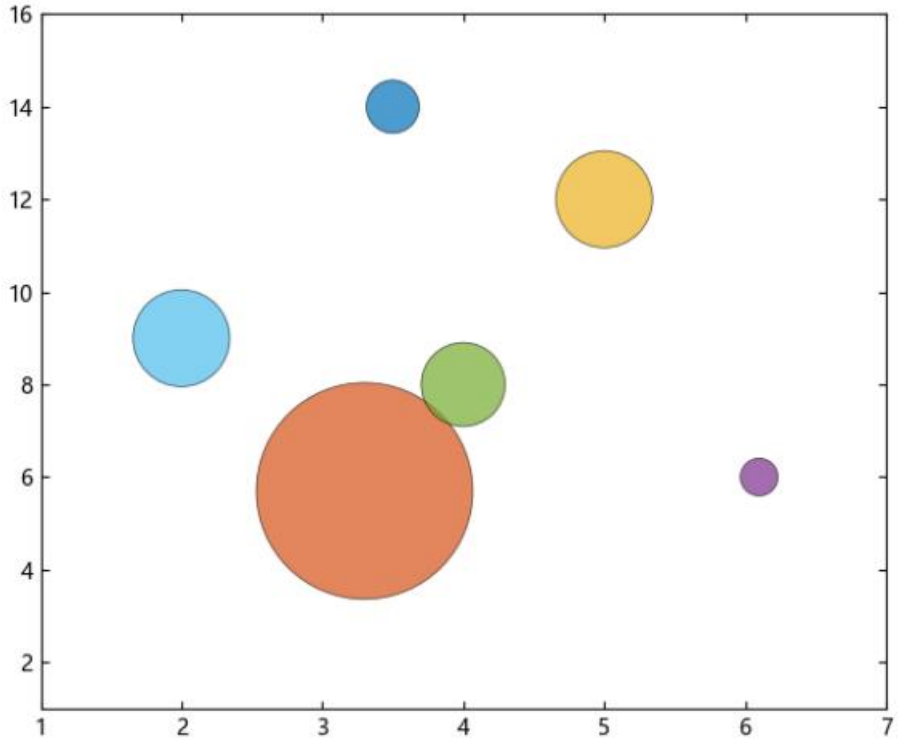
使用默认颜色创建散点图。

```
x = [3.5 3.3 5 6.1 4 2];
y = [14 5.7 12 6 8 9];
sz = 100 .* [6 100 20 3 15 20];
c = [1 2 3 4 5 6];
s = scatter(x, y, sz, c, filled=true, edgecolors="k", alpha=.7);
xlim([1 7]);
ylim([1 16]);
```



获取具有六个条目的线条颜色图数组。然后替换散点图中的颜色图。

```
c = lines(6);
colormap(s, c);
```



jet

Jet 颜色图数组

A. 语法

```
c = jet()
c = jet(m)
```

B. 说明

c = jet() 以一个三列数组的形式返回 Jet 颜色图，该数组包含的行数与当前图窗的颜色图相同。如果不存在图窗，则行数等于默认长度 256。数组中的每一行包含一种特定颜色的红、绿、蓝强度。强度介于 [0,1]范围内，颜色方案如下图所示。



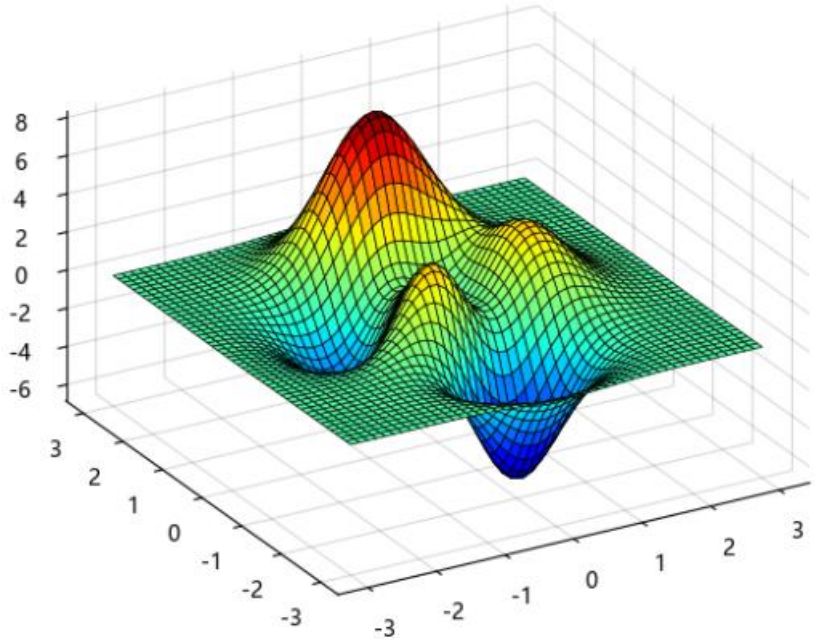
c = jet(m)返回包含 m 种颜色的颜色图。

C. 示例

(1). 反转颜色图

绘制曲面图，并分配 jet 颜色图。

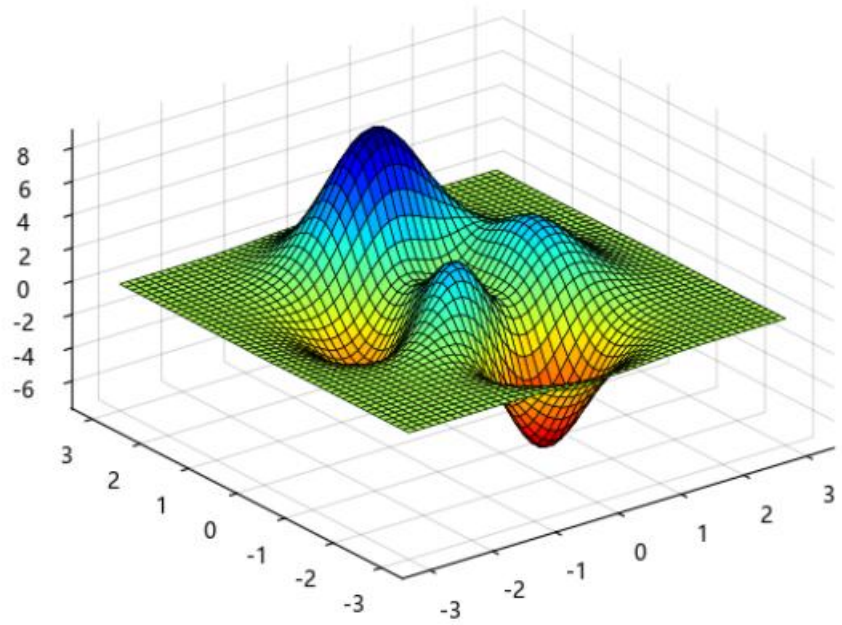
```
X,Y,Z = peaks();
s = surf(X,Y,Z);
colormap(s, "jet")
```



获取 jet 颜色图数组，并反转顺序。然后将修改后的颜色图应用于该曲面。

```
c = jet();
c[:,1] = reverse(c[:,1]);
c[:,2] = reverse(c[:,2]);
c[:,3] = reverse(c[:,3]);

colormap(s, c);
```

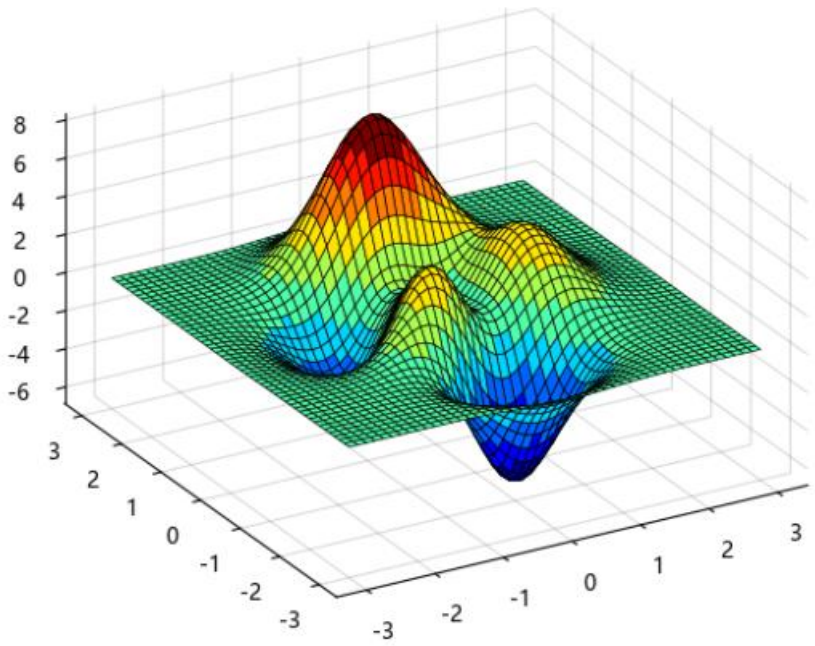


(2). 对 jet 颜色图进行减采样

获取仅包含 jet 颜色图的十种颜色的减采样版本。

```
X,Y,Z = peaks();
s = surf(X,Y,Z);

c = jet(10);
colormap(s, c)
```



colorcube

Colorcube 颜色图数组

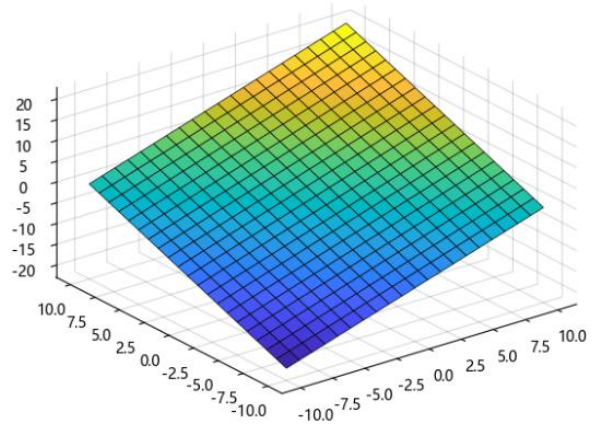
A. 语法

```
c = colorcube(s)
c = colorcube(m)
```

B. 说明

c = colorcube(s) 以三列数组形式返回 colorcube 颜色图，其中包含的行数与当前图窗的颜色图相同。如果不存在图窗，则行数等于默认长度 256。数组中的每一行包含一种特定颜色的红、绿、蓝强度。强度介于 [0,1] 范围内，颜

色方案如下图所示。示例



c = colorcube(m) 返回包含 m 种颜色的颜色图。示例。

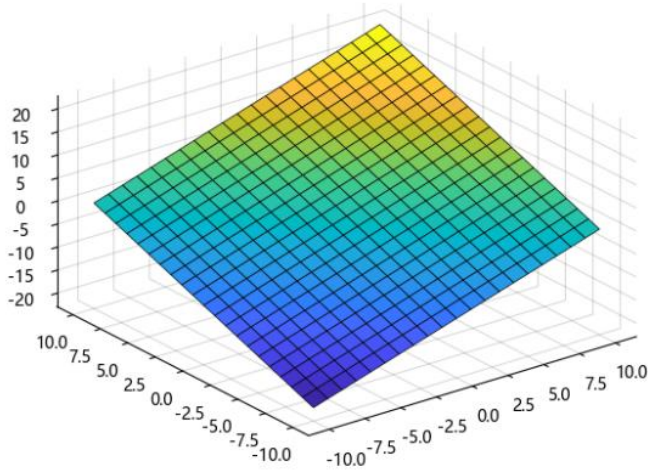
C. 示例

(1). 使用颜色图的一部分

获取颜色图数组后，您可以隔离一部分用作颜色方案。

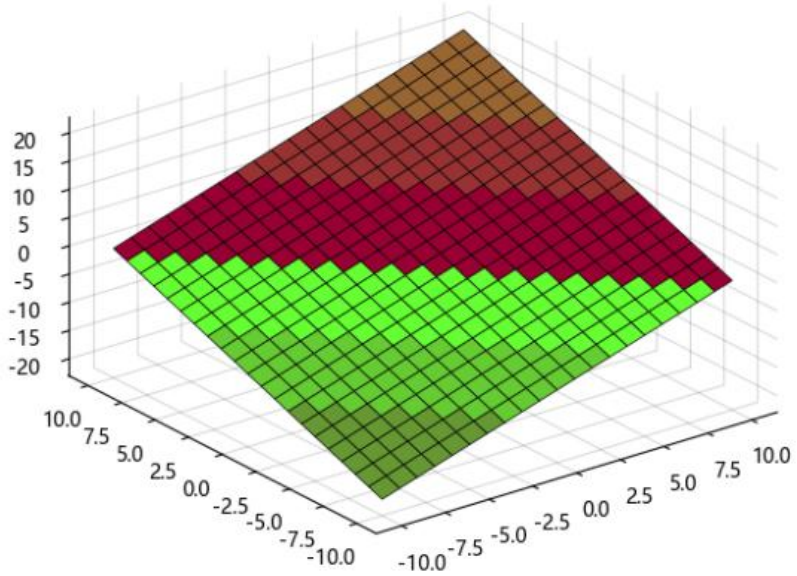
创建平面的曲面图。

```
X, Y = meshgrid2(-10:1:10);
Z = X + Y;
s = surf(X, Y, Z)
```



获取 colorcube 颜色图数组，并隔离红色部分（条目 39 到 44）。然后使用此部分为整个曲面着色。

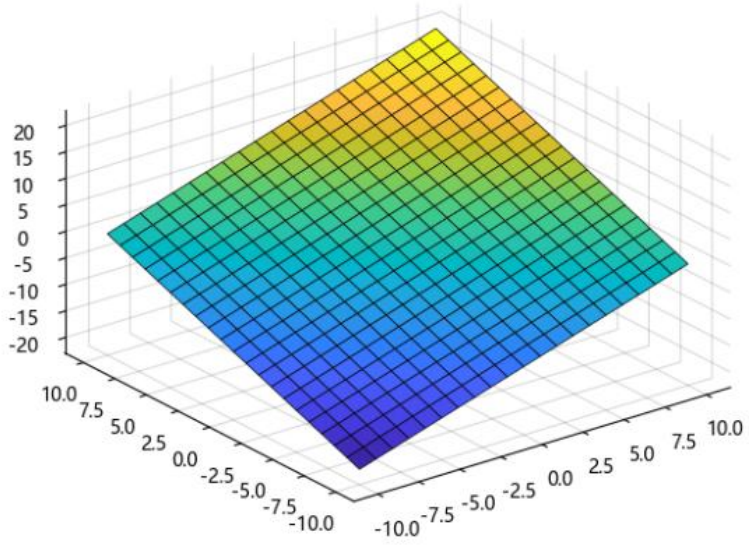
```
c = colorcube(s)
c = c[39:44, :]
colormap(s, c)
```



(2). 使用 colorcube 颜色图为平面着色

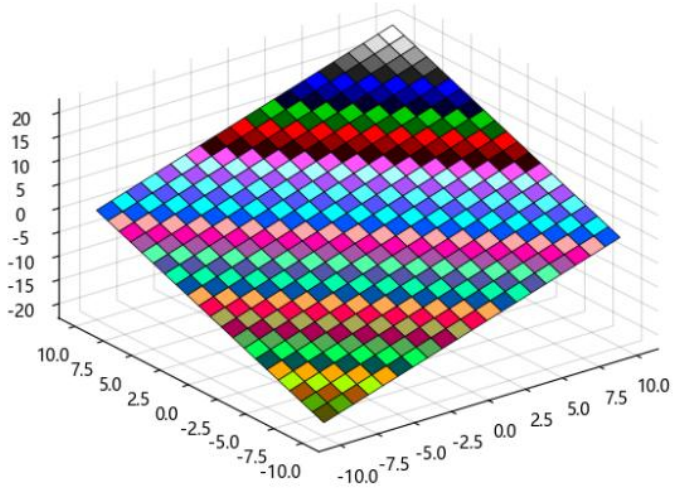
创建平面的曲面图。

```
X, Y = meshgrid2(-10:1:10);
Z = X + Y;
s = surf(X, Y, Z)
```



获取包含 75 个条目的 colorcube 颜色图，然后使用它为平面着色。

```
c = colorcube(75)
colormap(s, c)
```



prism

Prism 颜色图数组

D. 语法

```
c = prism()
c = prism(m)
```

E. 说明

c = prism() 以一个三列数组的形式返回 prism 颜色图，该数组包含的行数与当前图窗的颜色图相同。如果不存在图窗，则行数等于默认长度 256。数组中的每一行包含一种特定颜色的红、绿、蓝强度。强度值范围为 [0,1]，颜色方案是彩虹颜色的重复序列。



c = prism(m)返回包含 m 种颜色的颜色图。

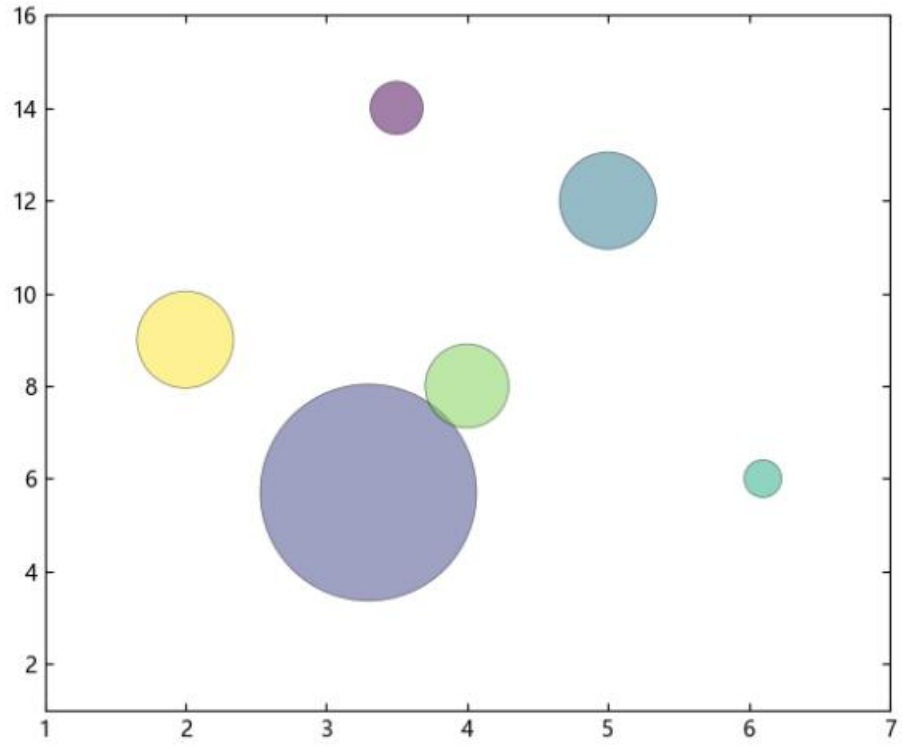
F. 示例

(3). 对 Prism 颜色图进行减采样

使用默认颜色创建一个散点图。

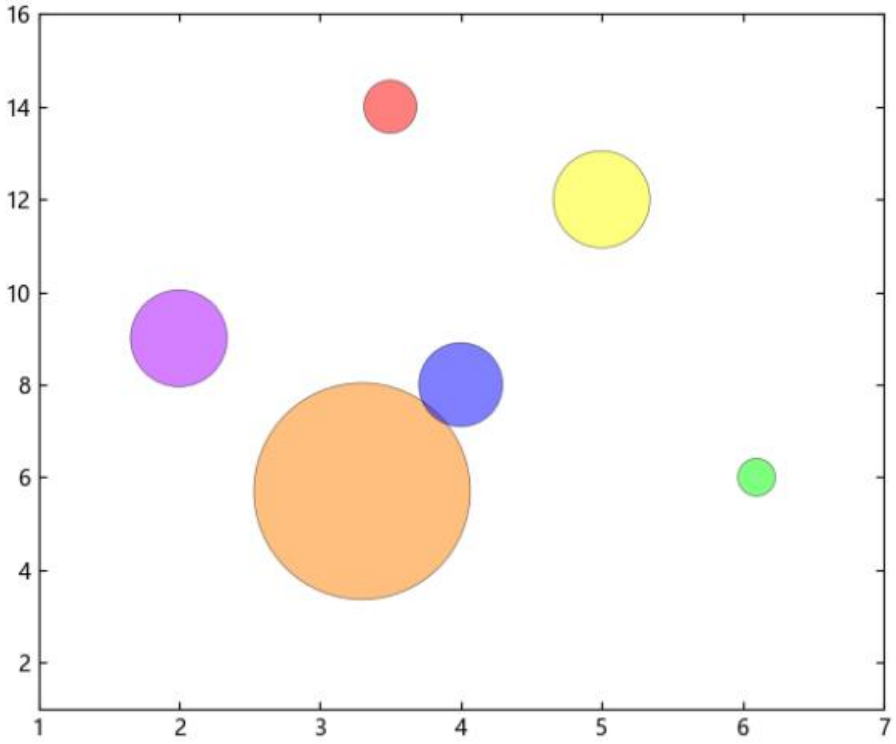
```
x = [3.5 3.3 5 6.1 4 2];
y = [14 5.7 12 6 8 9];
```

```
sz = 100 .* [6 100 20 3 15 20];
c = [1 2 3 4 5 6];
s = scatter(x, y, sz, c, filled=true, edgecolors="k", alpha=.7);
xlim([1 7]);
ylim([1 16]);
```



获取具有六个条目的 prism 颜色图数组。然后替换散点图中的颜色图。

```
c = prism(6);
colormap(s, c);
```



flag

flag 颜色图数组

A. 语法

```
c = flag()
c = flag(m)
```

B. 说明

c = flag() 以一个三列数组的形式返回 flag 颜色图，该数组包含的行数与当前图窗的颜色图相同。如果不存在图窗，则行数等于默认长度 256。数组中的每一行包含一种特定颜色的红、绿、蓝强度。强度值范围为 [0,1]，颜色方案是彩虹颜色的重复序列。



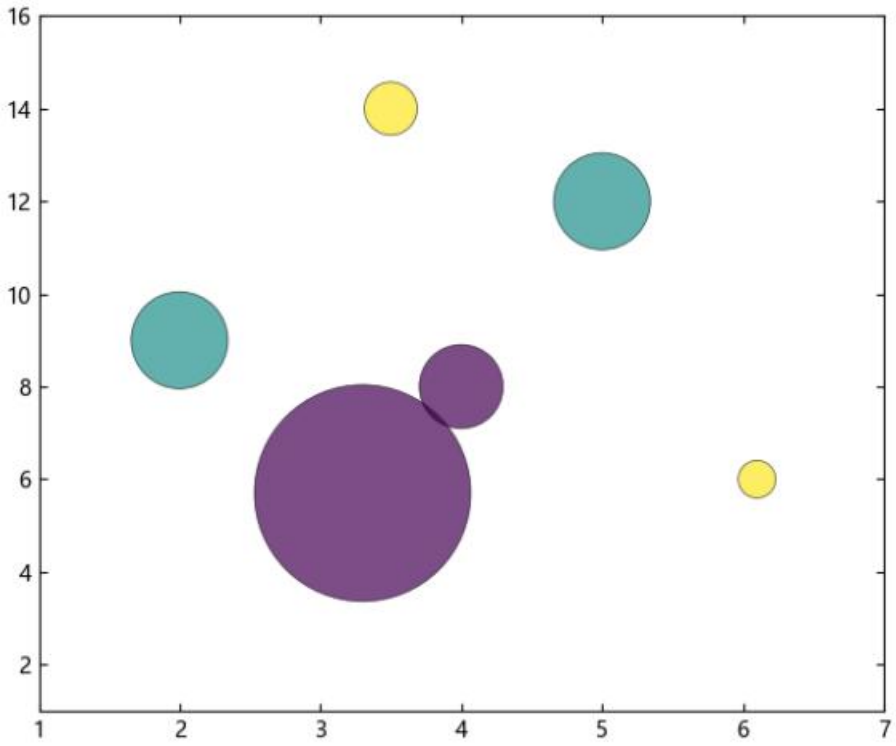
c = flag(m)返回包含 m 种颜色的颜色图。

C. 示例

(1). 减采样 flag 颜色图

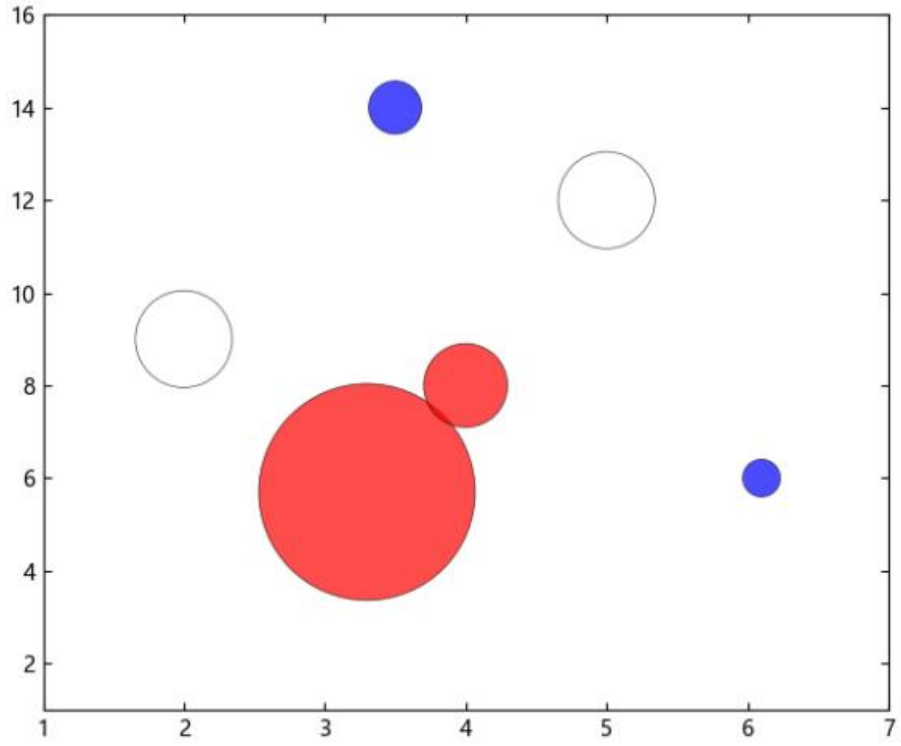
使用默认颜色创建散点图。

```
x = [3.5 3.3 5 6.1 4 2];
y = [14 5.7 12 6 8 9];
sz = 100 .* [6 100 20 3 15 20];
c = [3 1 2 3 1 2];
s = scatter(x, y, sz, c, filled = true, markeredgecolor = "k",
alpha = 0.7);
xlim([1 7]);
ylim([1 16]);
```



获取包含三个条目的 flag 颜色图数组。然后替换散点图中的颜色图。

```
c = flag(3);
colormap(s, c);
```



3.3.2.4 三维场景控制

添加光源、设置对象透明度、控制相机视图

函数名	简介
plt_view	相机视线
rotate	以指定原点和方向旋转对象
makehgtform	创建 4×4 变换矩阵
viewmtx	查看变换矩阵
camproj	设置或查询投影类型
camzoom	放大和缩小场景
alpha	向坐标区中的对象添加透明度

3.3.2.4.1 plt_view

相机视线

A. 语法

```
plt_view(az,el)
plt_view(v)
plt_view(dim)
plt_view(ax,___)
caz, cel = plt_view()
```

B. 说明

plt_view(az,el)为当前坐标区设置相机视线的方位角和仰角。

plt_view(v)根据 v（二元素或三元素数组）设置视线：

二元素数组 - 其值分别是方位角和仰角。

三元素数组 - 其值是从图框中心点到相机位置所形成向量的 x、y 和 z 坐标。Syslab 使用指向同一方向的单位向量计算方位角和仰角。

plt_view(dim)对二维或三维绘图使用默认视线。对默认二维视图，将 dim 指定为 2，对于默认三维视图，指定为 3。

plt_view(ax,___)指定目标坐标区的视线。

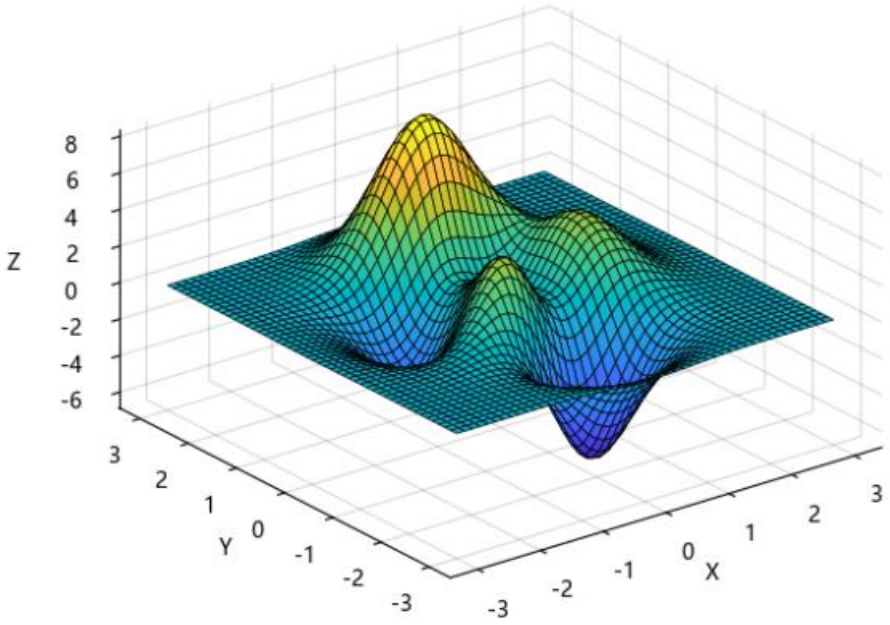
caz, cel = plt_view() 分别将方位角和仰角返回为 caz 和 cel。指定上述任一语法中的输入参数，以获得新视线的角度。或者，不指定输入参数以获得当前视线的角度。

C. 示例

(1). 更改曲面图的视图

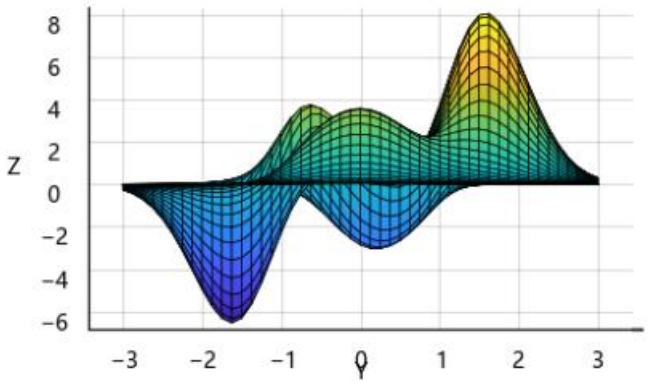
使用 peaks 函数获取曲面的 x、y 和 z 坐标。然后绘制曲面并标记每个轴。

```
X, Y, Z = peaks();
surf(X, Y, Z)
xlabel("X")
ylabel("Y")
zlabel("Z")
```



使用 90 度的方位角和 0 度的仰角查看绘图。新视线与 x 轴同向。

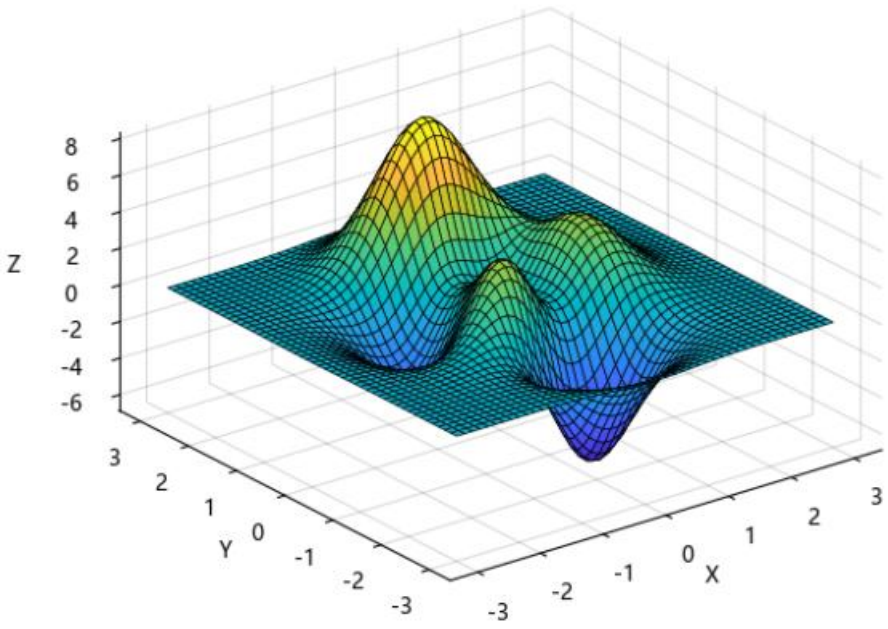
```
plt_view(90, 0)
```



(2). 在二维视图中显示曲面

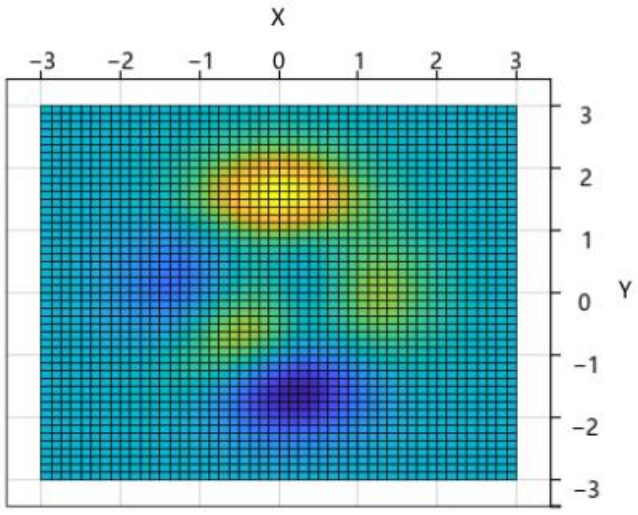
使用 peaks 函数获取曲面的 x、y 和 z 坐标。然后绘制曲面并标记每个轴。

```
X,Y,Z = peaks();
surf(X,Y,Z)
xlabel("X")
ylabel("Y")
zlabel("Z")
```



在二维视图中显示绘图。

```
plt_view(2)
```

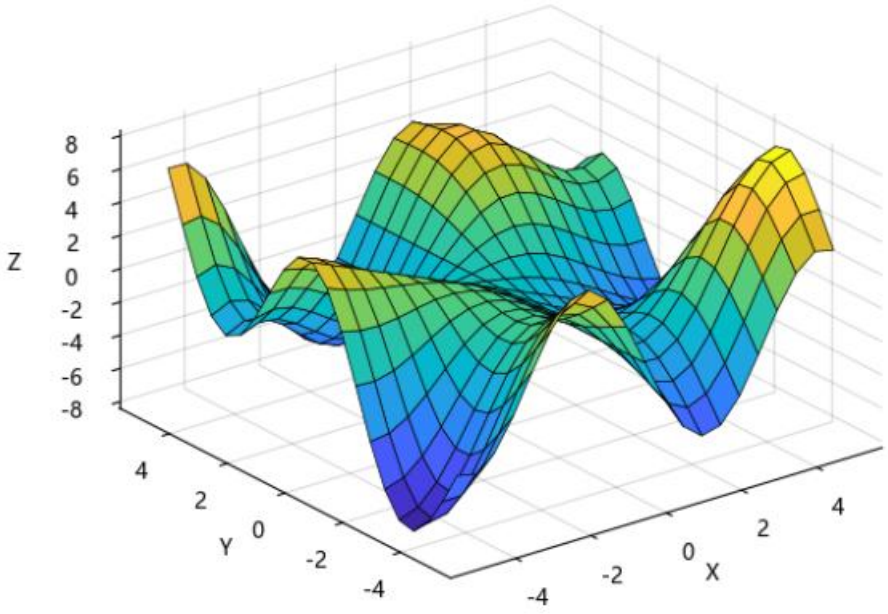


(3). 使用向量更改视图

创建一组 x、y 和 z 坐标，并使用它们绘制一个曲面。然后标记每个轴。

```
X, Y = meshgrid2(-5:0.5:5, -5:0.5:5);
Z = Y .* sin.(X) - X .* cos.(Y);

surf(X, Y, Z)
xlabel("X")
ylabel("Y")
zlabel("Z")
```



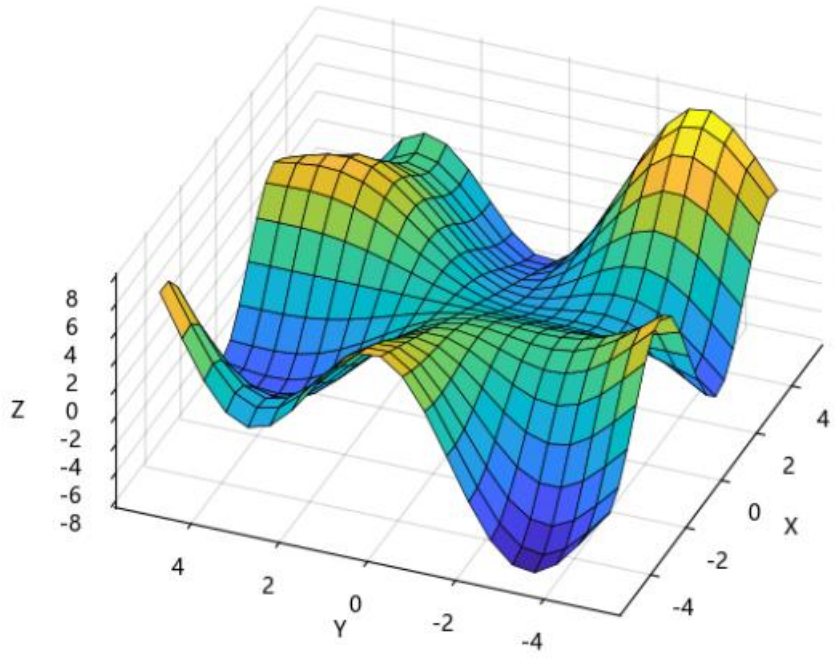
获取此绘图的方位角和仰角。

```
caz, cel = plt_view()
```

```
caz = -37.5
cel = 30.0
```

通过将 v 指定为一个向量的 x、y 和 z 坐标来更改视图，并返回新的方位角和仰角。新角度基于指向与 v 相同方向的单位向量。

```
v = [-5 -2 5];
caz, cel = plt_view(v)
```




```
caz = -68.1986
cel = 42.8760
```

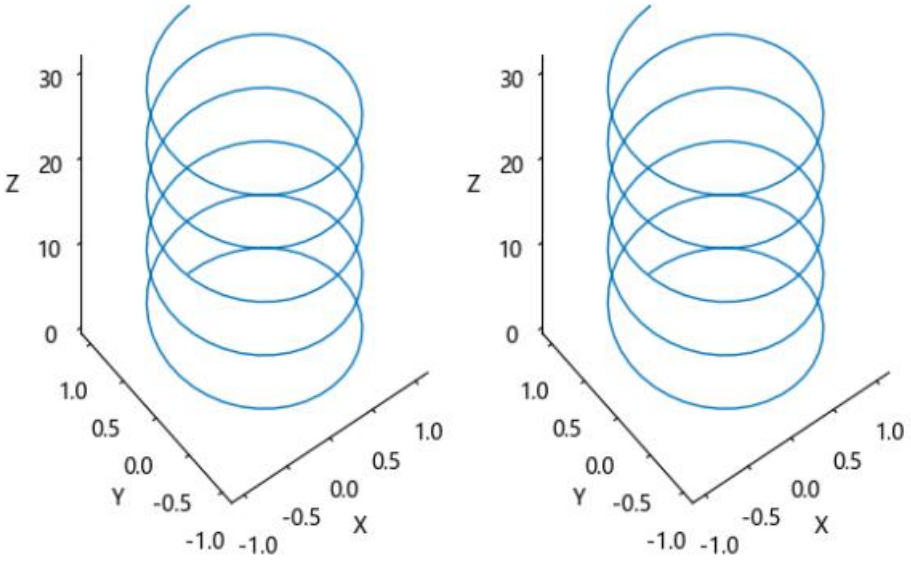
(4). 更改特定坐标的视图

调用 subplot 函数以创建一个 2×1 分块图布局和一个坐标区对象，并将该对象返回为 ax1 和 ax2。在两个坐标区中分别创建一个同样的线图。

```
t = 0:pi/20:10*pi;
xt1 = sin.(t);
yt1 = cos.(t);

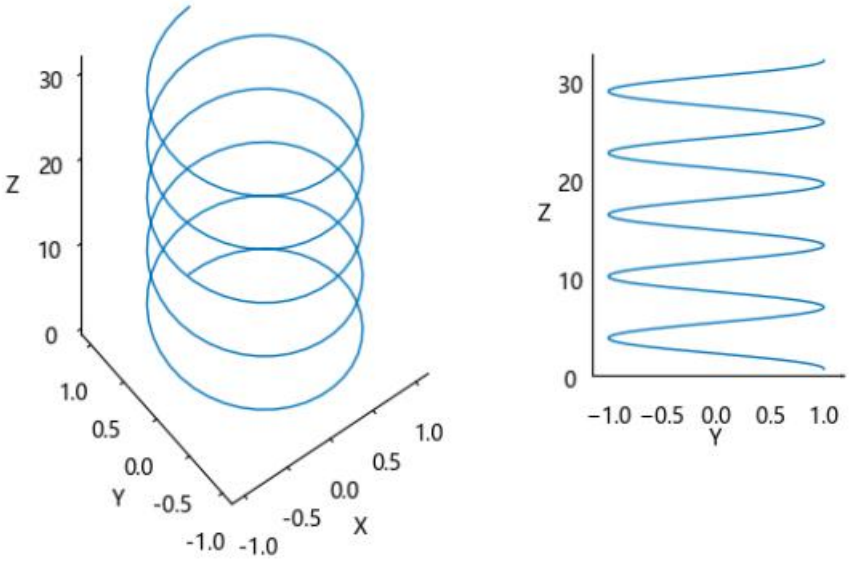
ax1 = subplot(1, 2, 1, projection = "3d")
plot3(ax1, xt1, yt1, t)
xlabel("X")
ylabel("Y")
zlabel("Z")

ax2 = subplot(1, 2, 2, projection = "3d")
plot3(ax2, xt1, yt1, t)
xlabel("X")
ylabel("Y")
zlabel("Z")
```



将右图的视图更改为沿 x 轴的侧视图。

```
plt_view(ax2, [90 0]);
```



3.3.2.4.2 rotate

以指定原点和方向旋转对象

A. 语法

```
rotate(data,direction,alpha)
rotate(...,origin)
```

B. 说明

rotate 函数在三维空间中旋转图形对象。

rotate(data,direction,alpha) 将图形旋转 alpha 度。direction 是一个三元素向量，它与旋转轴原点共同确定旋转轴。旋转轴的默认原点是图框中心。该点未必是坐标区的原点。示例

正 alpha 定义围绕方向向量从旋转原点伸展的右旋角度。

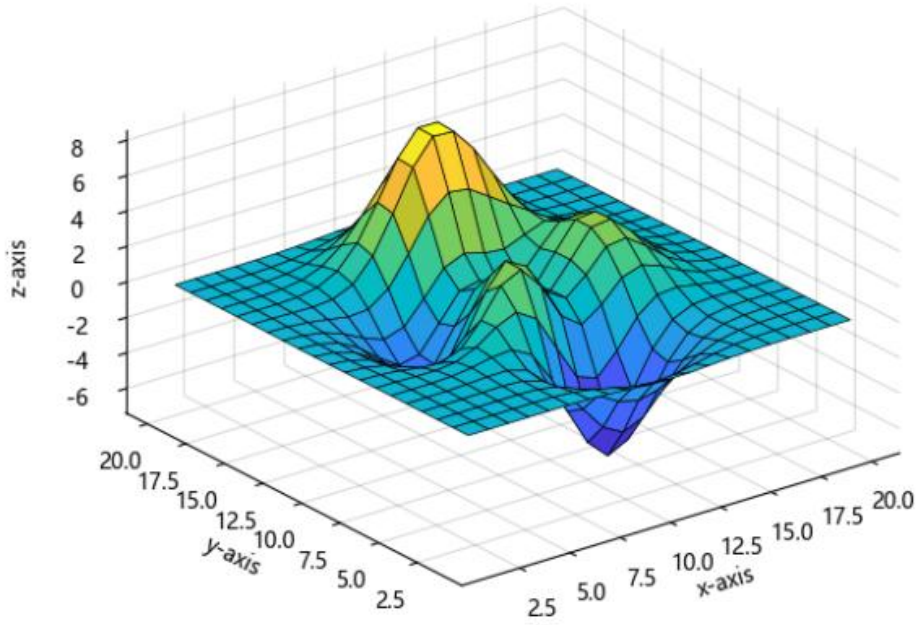
rotate(...,origin) 将旋转轴的原点指定为三元素向量 [x0,y0,z0]。示例。

C. 示例

(1). 绕 x 轴旋转绘图

创建 peaks 函数的曲面图并返回曲面对象。

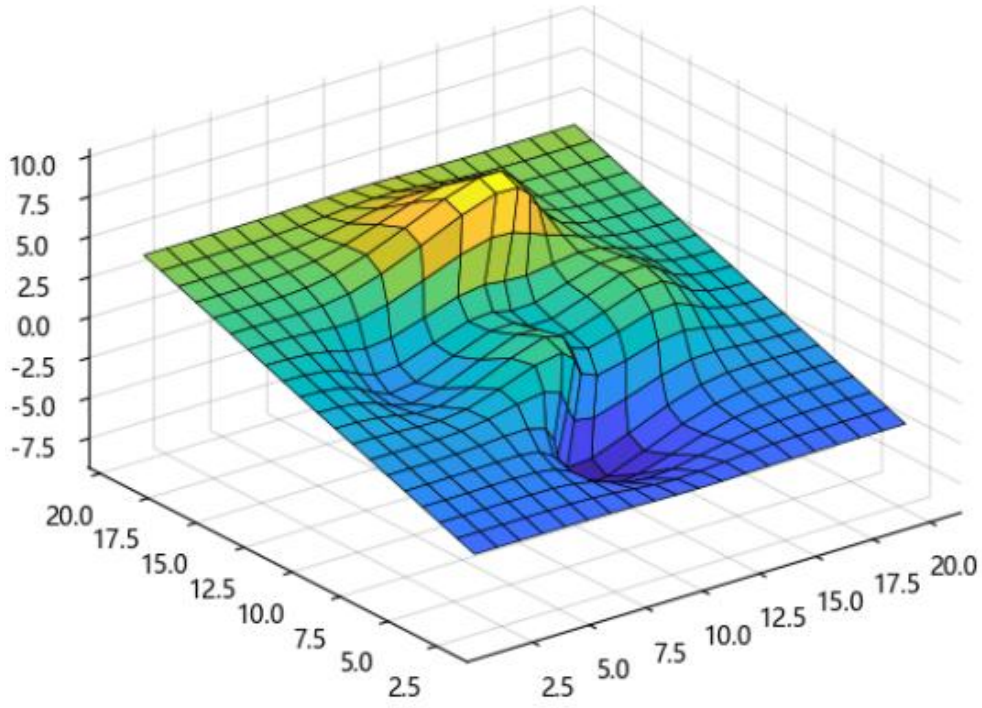
```
x_data, y_data, z_data = peaks(20)
s = surf(z_data)
xlabel("x-axis")
ylabel("y-axis")
zlabel("z-axis")
```



将曲面图绕其 x 轴旋转 25 度。

```
direction = [1, 0, 0]
```

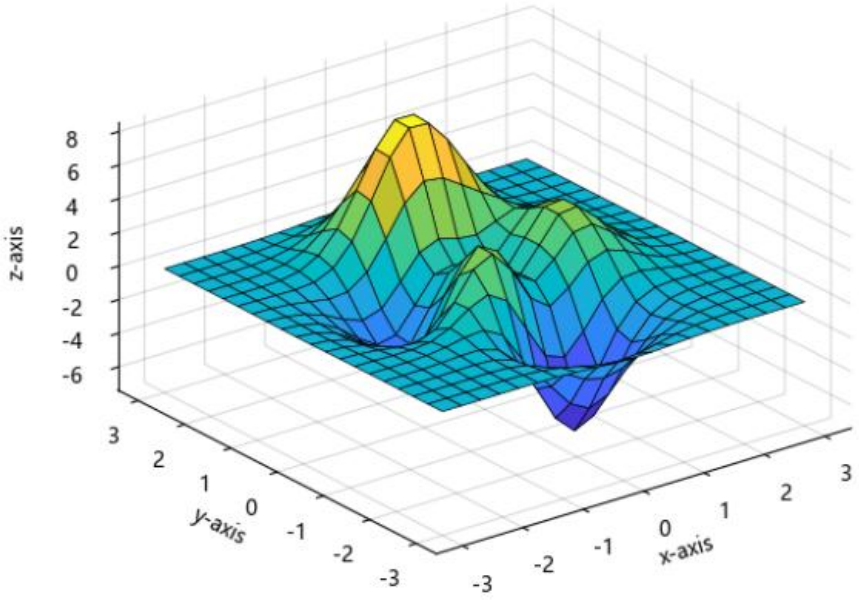
```
rotate(z_data, direction, 25)
```



(2). 绕 y 轴旋转绘图

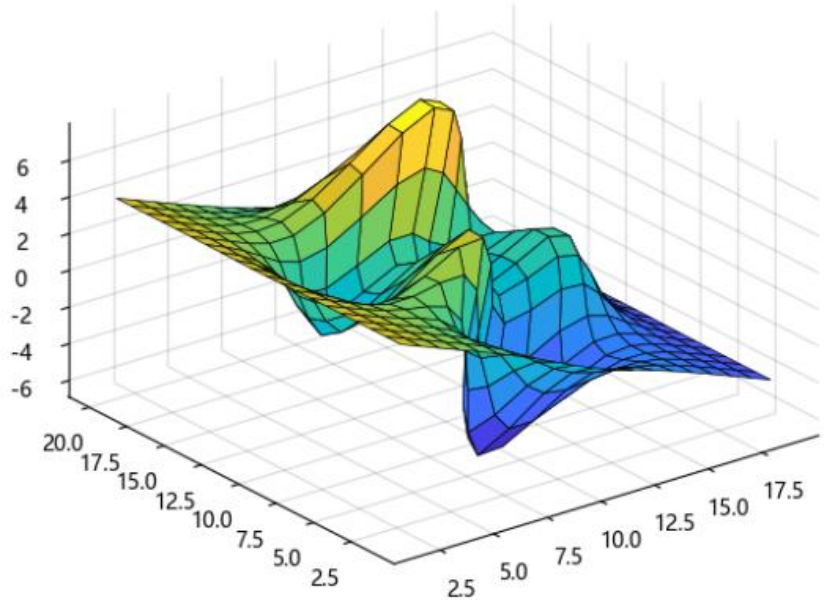
创建 peaks 函数的曲面图并返回曲面对象。

```
x_data, y_data, z_data = peaks(20)
surf(x_data, y_data, z_data)
xlabel("x-axis")
ylabel("y-axis")
zlabel("z-axis")
```



将曲面图绕其 y 轴旋转 25 度。

```
direction = [1, 0, 0]
rotate(z_data, direction, 25)
```

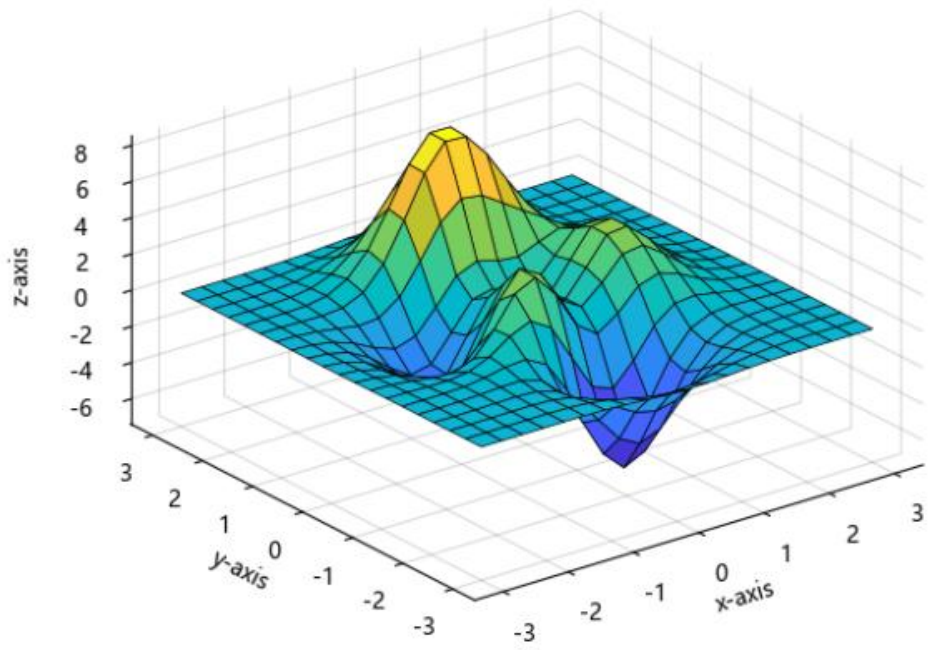


(3). 绕 x 轴和 y 轴旋转绘图

创建 peaks 函数的曲面图并返回曲面对象。

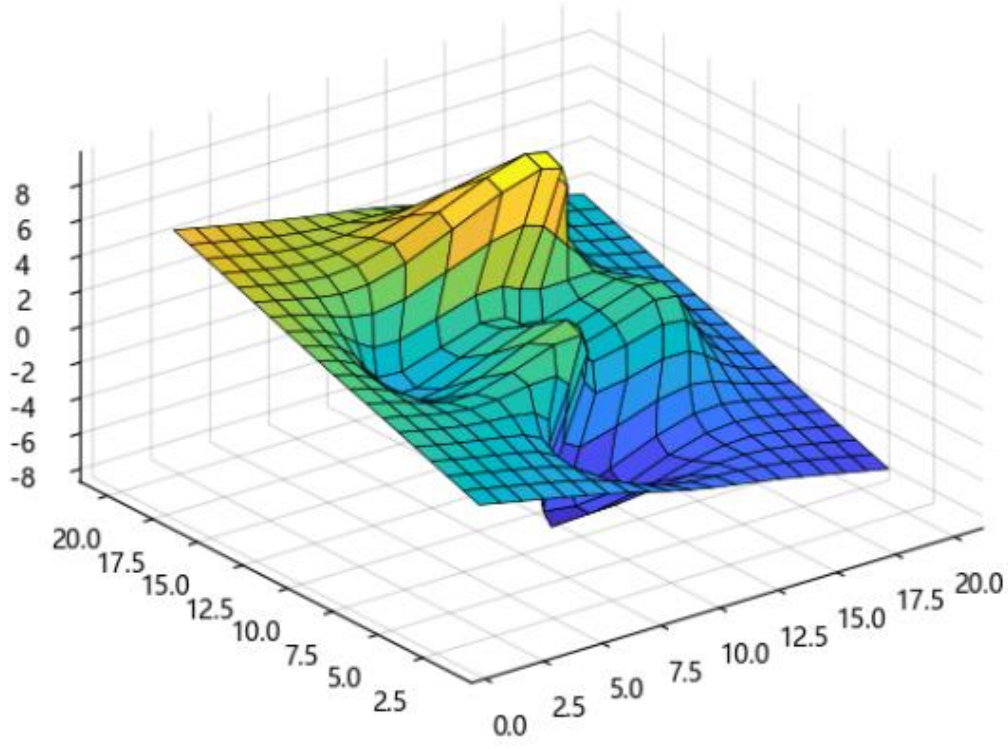
```
x_data, y_data, z_data = peaks(20)
surf(x_data, y_data, z_data)
xlabel("x-axis")
```

```
ylabel("y-axis")
zlabel("z-axis")
```



将曲面图绕其 x 轴和 y 轴旋转 25 度。

```
direction = [1, 1, 0]
rotate(z_data, direction, 25)
```



3.3.2.4.3 makehgtform

创建 4×4 变换矩阵

A. 语法

```
M = makehgtform()
M = makehgtform("translate",[tx ty tz])
M = makehgtform("scale",s)
M = makehgtform("scale",[sx,sy,sz])
M = makehgtform("xrotate",t)
M = makehgtform("yrotate",t)
M = makehgtform("zrotate",t)
M = makehgtform("axisrotate",[ax,ay,az],t)
```

B. 说明

使用 makehgtform 创建用于转换、缩放和旋转图形对象的变换矩阵。通过将变换指定给父变换对象的 Matrix 属性来将变换应用于图形对象。

M = makehgtform() 返回恒等变换矩阵。

M = makehgtform("translate",[tx ty tz]) 返回分别沿 x、y 和 z 轴按 tx、ty 和 tz 进行转换的变换矩阵。

M = makehgtform("scale",s) 返回沿 x、y 和 z 轴均匀缩放的变换矩阵。

M = makehgtform("scale",[sx,sy,sz]) 返回分别沿 x、y 和 z 轴按 sx、sy 和 sz 进行缩放的变换矩阵。

M = makehgtform("xrotate",t) 返回围绕 x 轴旋转 t 弧度的变换矩阵。

M = makehgtform("yrotate",t) 返回围绕 y 轴旋转 t 弧度的变换矩阵。

M = makehgtform("zrotate",t) 返回围绕 z 轴旋转 t 弧度的变换矩阵。

M = makehgtform("axisrotate",[ax,ay,az],t) 围绕轴 [ax ay az] 旋转 t 弧度。示例。

C. 示例

(1). 生成格式变换矩阵

使用 makehgtform 创建用于转换、缩放和旋转图形对象的变换矩阵。

```
makehgtform()
```

```
4×4 Matrix{Float64}:
```

```
 1.0  0.0  0.0  0.0
 0.0  1.0  0.0  0.0
 0.0  0.0  1.0  0.0
 0.0  0.0  0.0  1.0
```

```
makehgtform("translate", [9, 5, 7])
```

```
4×4 Matrix{Float64}:
 1.0  0.0  0.0  9.0
 0.0  1.0  0.0  5.0
 0.0  0.0  1.0  7.0
 0.0  0.0  0.0  1.0
```

```
makehgtform("scale", 9)
```

```
4×4 Matrix{Float64}:
 9.0  0.0  0.0  0.0
 0.0  9.0  0.0  0.0
 0.0  0.0  9.0  0.0
 0.0  0.0  0.0  1.0
```

```
makehgtform("scale", [9, 5, 7])
```

```
4×4 Matrix{Float64}:
 9.0  0.0  0.0  0.0
 0.0  5.0  0.0  0.0
 0.0  0.0  7.0  0.0
 0.0  0.0  0.0  1.0
```

```
makehgtform("xrotate", pi / 6)
```

```
4×4 Matrix{Float64}:
 1.0  0.0       0.0      0.0
 0.0  0.866025  -0.5      0.0
 0.0  0.5       0.866025  0.0
 0.0  0.0       0.0      1.0
```

```
makehgtform("yrotate", pi / 6)
```

```
4×4 Matrix{Float64}:
  0.866025  0.0  0.5      0.0
  0.0       1.0  0.0      0.0
 -0.5       0.0  0.866025  0.0
  0.0       0.0  0.0      1.0
```

```
makehgtform("zrotate", pi / 6)
```

```
4×4 Matrix{Float64}:
 0.866025  -0.5      0.0  0.0
 0.5       0.866025  0.0  0.0
 0.0       0.0       1.0  0.0
 0.0       0.0       0.0  1.0
```

```
makehgtform("axisrotate", [1, 1, 0], pi / 2)
```

```
4×4 Matrix{Float64}:
  0.5       0.5       0.707107     0.0
  0.5       0.5      -0.707107     0.0
 -0.707107  0.707107   6.12323e-17  0.0
  0.0       0.0       0.0          1.0
```

3.3.2.4.4 viewmtx

查看变换矩阵

A. 语法

```
T = viewmtx(az,el)
T = viewmtx(az,el,phi)
T = viewmtx(az,el,phi,xc)
```

B. 说明

T = viewmtx(az,el) 返回对应于方位角 az 和仰角 el 的正交变换矩阵。az 是视点的方位角（即水平旋转，以度为单位）。el 是视点的仰角（以度为单位）。示例

T = viewmtx(az,el,phi) 返回透视变换矩阵。phi 是透视视角（以度为单位）。phi 是归一化绘图立方体（以度为单位）的对向视角，控制透视扭曲量。示例。

Phi	说明
0 度	正交投影
10 度	类似于远摄镜头
25 度	类似于普通镜头
60 度	类似于广角镜头

T = viewmtx(az,el) 返回对应于方位角 az 和仰角 el 的正交变换矩阵。az 是视点的方位角（即水平旋转，以度为单位）。el 是视点的仰角（以度为单位）。示例

T = viewmtx(az,el,phi) 返回透视变换矩阵。phi 是透视视角（以度为单位）。phi 是归一化绘图立方体（以度为单位）的对向视角，控制透视扭曲量。示例

C. 示例

(1). 计算转换矩阵

使用默认视图方向确定对应于三维点 (0.5,0.0,-3.0) 的投射二维向量。请注意，该点为列向量。

```
A = viewmtx(-37.5, 30)
x4d = [0.5 0 -3 1]';
x2d = A * x4d;
x2d = x2d[1:2]
```

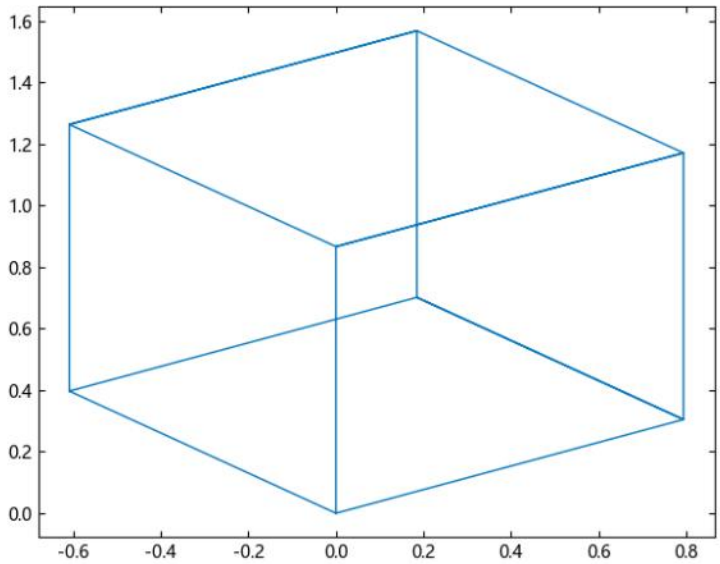
```
2-element Vector{Float64}:
  0.39667667014561747
 -2.4458858541011357
```

创建用于跟踪单位立方体的边的向量。

```
x = [0 1 1 0 0 0 1 1 0 0 1 1 1 1 0 0];
y = [0 0 1 1 0 0 0 1 1 0 0 0 1 1 1 1];
z = [0 0 0 0 0 1 1 1 1 1 1 0 0 1 1 0];
```

将这些向量中的点变换到屏幕上，然后绘制对象：

```
A = viewmtx(-37.5, 30)
(m, n) = size(x)
x4d = [x' y' z' ones(m * n, 1)]'
x2d = A * x4d
x2 = zeros(m, n)
y2 = zeros(m, n)
x2[:] = x2d[1, :]
y2[:] = x2d[2, :]
plot(x2, y2)
```



以 25 度视角使用透视变换。

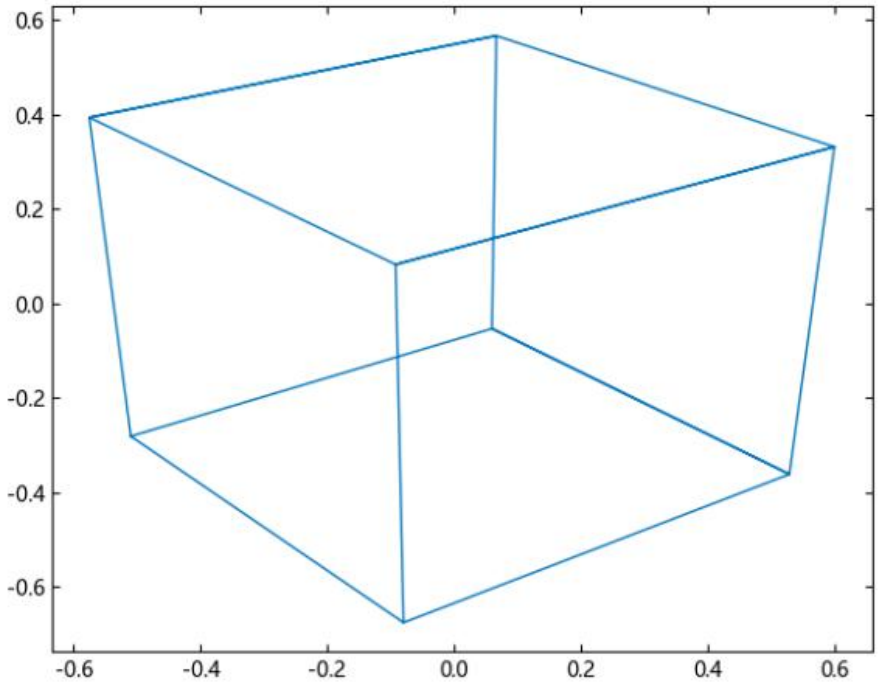
```
A = viewmtx(-37.5, 30, 25);

x4d = [0.5 0 -3 1]';
x2d = A * x4d;
x2d = x2d[1:2] / x2d[4]
```

```
2-element Vector{Float64}:
  0.17774260927705404
 -1.8858186417007579
```

将立方体向量变换到屏幕上，然后绘制对象。

```
A = viewmtx(-37.5, 30, 25)
(m, n) = size(x)
x4d = [x' y' z' ones(m * n, 1)]'
x2d = A * x4d
x2 = zeros(m, n)
y2 = zeros(m, n)
x2[:] = x2d[1, :] ./ x2d[4, :]
y2[:] = x2d[2, :] ./ x2d[4, :]
plot(x2, y2)
```



3.3.2.4.5 camproj

设置或查询投影类型

A. 语法

```
camproj()
camproj("projection_type")
camproj(axes_handle,...)
```

B. 说明

camproj() 返回当前坐标区中的投影类型设置。投影类型确定 Syslab 三维视图是使用透视投影还是正交投影。

camproj("projection_type") 将当前坐标区中的投影类型设置为指定的值。projection_type 的可能值包括 "orthographic" 和 "perspective"。示例

camproj(axes_handle,...) 在第一个参数 axes_handle 标识的坐标区上执行设置或查询。不指定坐标区句柄时，camproj 在当前坐标区上操作。

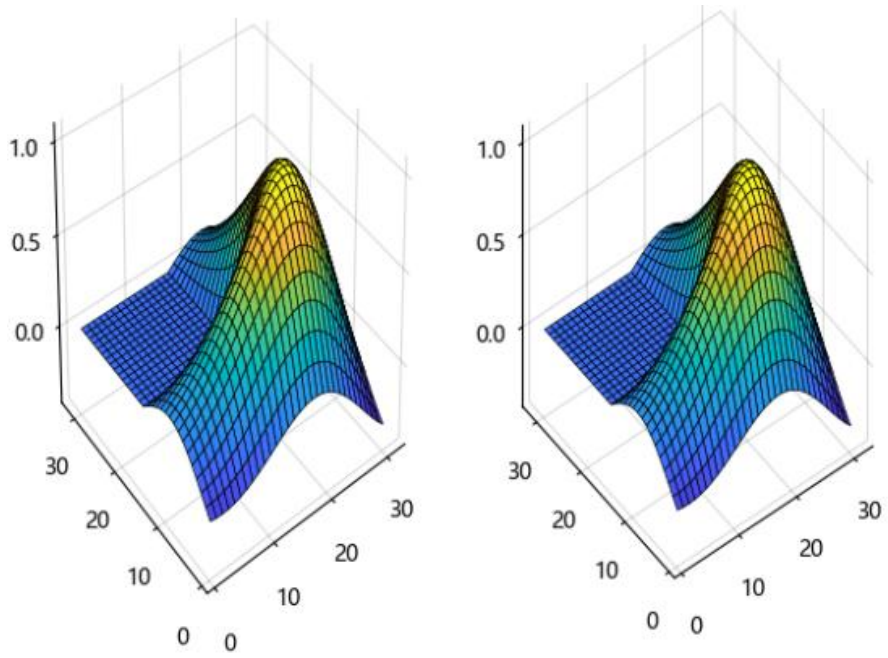
C. 示例

(1). 设置投影类型

绘制两个曲面，并将第一个坐标区的投影设置为 "perspective"。第二个坐标区使用默认投影，即 "orthographic"。

```
using TyPlot
source_path = pkgdir(TyPlot) * "/examples/camproj/0-data.jl"
include(source_path);

ax1 = subplot(1, 2, 1; projection="3d")
ax2 = subplot(1, 2, 2; projection="3d")
surf(ax1, membrane)
camproj("perspective")
surf(ax2, membrane)
```



3.3.2.4.6 camzoom

放大和缩小场景

A. 语法

```
camzoom(zoom_factor)
camzoom(axes_handle,...)
```

B. 说明

camzoom(zoom_factor) 根据 zoom_factor 指定的值放大或缩小场景。如果 zoom_factor 大于 1，则场景将变大；如果 zoom_factor 大于零且小于 1，则场景将变小。示例

camzoom(axes_handle,...) 在第一个参数 axes_handle 标识的坐标区上操作。不指定坐标区句柄时，camzoom 在当前坐标区上操作。示例。

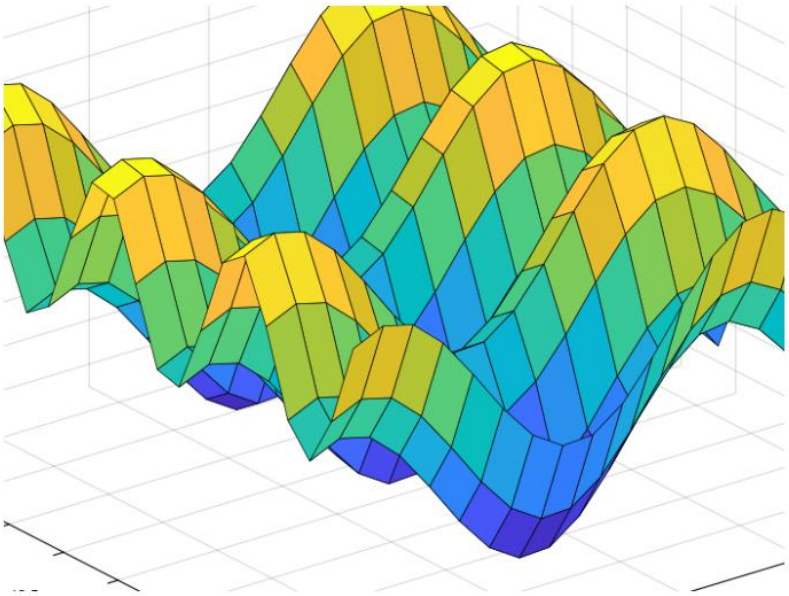
C. 示例

(1). 放大缩小场景

创建三个相同大小的矩阵。然后将它们绘制为一个曲面。将曲面图放大 2 倍。

```
using TyPlot
```

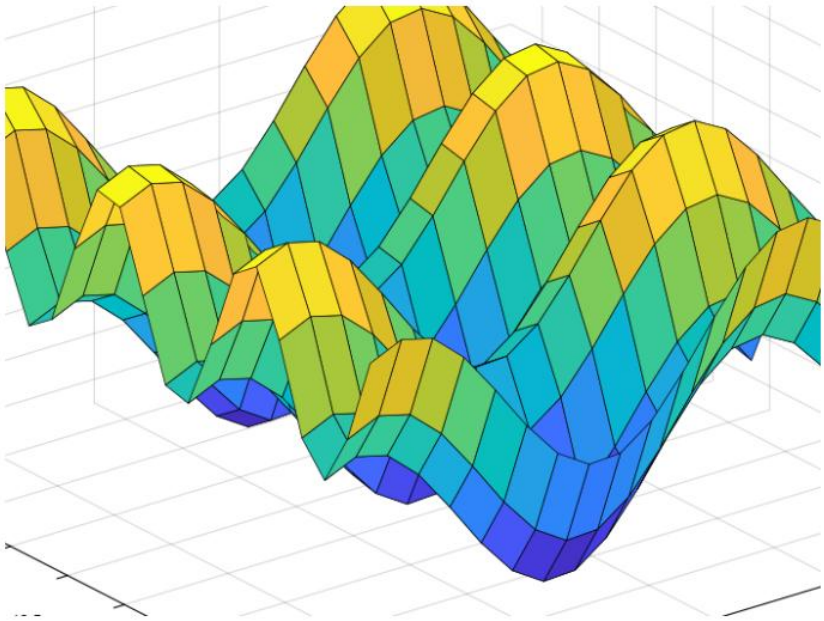
```
X, Y = meshgrid2(1:0.5:10, 1:20)
Z = sin.(X) + cos.(Y)
surf(X, Y, Z)
camzoom(2)
```



(2). ‘指定 axes_handle 标识的坐标区场景缩放

创建三个相同大小的矩阵。然后将它们绘制为一个曲面。将当前图窗放大 2 倍。

```
using TyPlot
X, Y = meshgrid2(1:0.5:10, 1:20)
Z = sin.(X) + cos.(Y)
surf(X, Y, Z)
camzoom(gca(), 2)
```



3.3.2.4.7 alpha

向坐标区中的对象添加透明度

A. 语法

```
alpha(value)
alpha(obj,___)
alpha(ax,___)
```

B. 说明

alpha(value) 为当前坐标区中支持透明度的图形对象设置面透明度。将 value 指定为介于 0（透明）和 1（不透明）之间的标量值（两者分别表示为 "clear" 和 "opaque"）。

alpha(obj,___) 用于为 obj 指定的对象设置面透明度。选项 obj 可以位于上述语法中的任何输入参数之前。例如 alpha(obj,"opaque")。

alpha(ax,___) 设置 ax 指定的坐标区（而非当前坐标区）中图形对象的面透明度。指定 ax 作为第一个输入参数。您不能同时指定 ax 和 obj 输入。

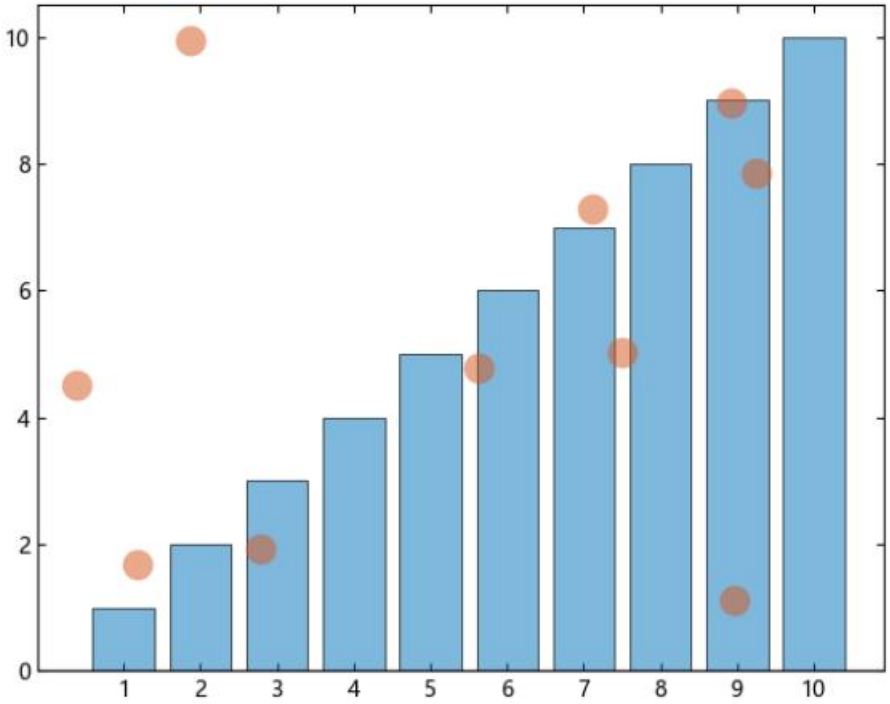
C. 示例

(1). 设置坐标区中所有图形对象的透明度

创建条形图和散点图。将条形序列和散点序列对象的面透明度都设置为 0.5。

```
b = bar(1:10)
```

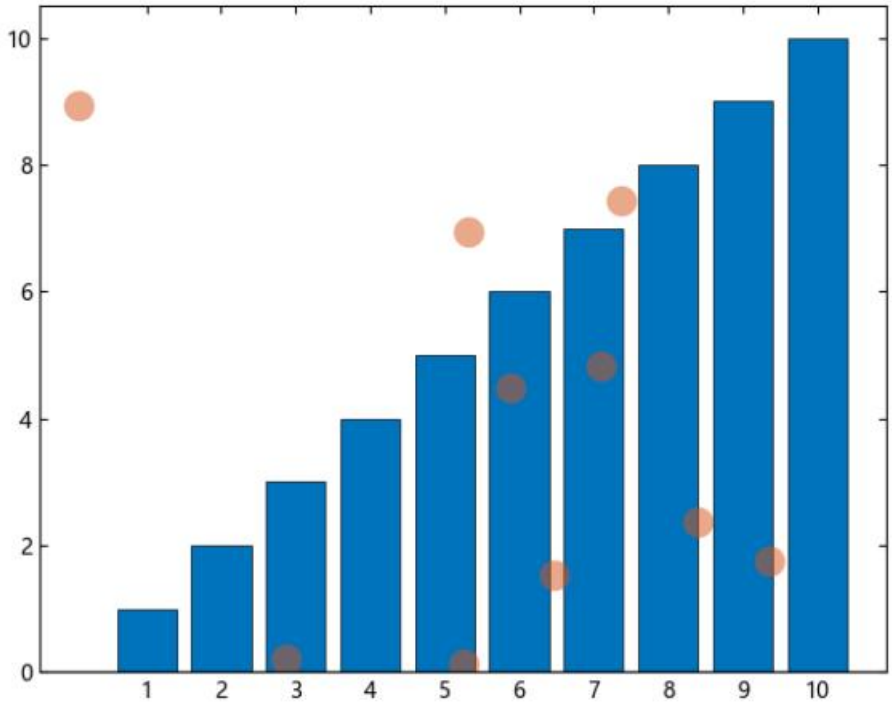
```
hold("on")
s = scatter(10 * rand(10, 1), 10 * rand(10, 1), 200; filled=true)
hold("off")
alpha(0.5)
```



(2). 设置特定图形对象的透明度

创建条形图和散点图。将散点序列对象的透明度设置为 0.5 而不影响条形序列对象。

```
b = bar(1:10);
hold("on")
s = scatter(10 * rand(10, 1), 10 * rand(10, 1), 200; filled=true)
hold("off")
alpha(s, 0.5)
```



3.3.3 打印和保存

打印和导出为标准文件格式

函数名	简介
plt_print	打印图窗或保存为特定文件格式
saveas	将图窗保存为特定文件格式
getframe	捕获坐标区或图窗作为影片帧

3.3.3.1 plt_print

打印图窗或保存为特定文件格式

A. 语法

```
plt_print()

plt_print(filename)

plt_print(printer)
```

```
plt_print(fig,___)
```

B. 说明

plt_print(filename) 使用指定的文件格式将当前图窗保存到文件中，例如 plt_print("BarPlot","-dpng")。如果该文件不包括扩展名，则 plt_print 会附加适用的扩展名。

plt_print 将当前图窗输出到默认打印机。

plt_print(printer) 指定打印机。将打印机指定为字符向量或字符串，其中包含以 -P 开头的打印机名称，例如 "-Pmy printer"。该打印机必须已设置在您的系统上。

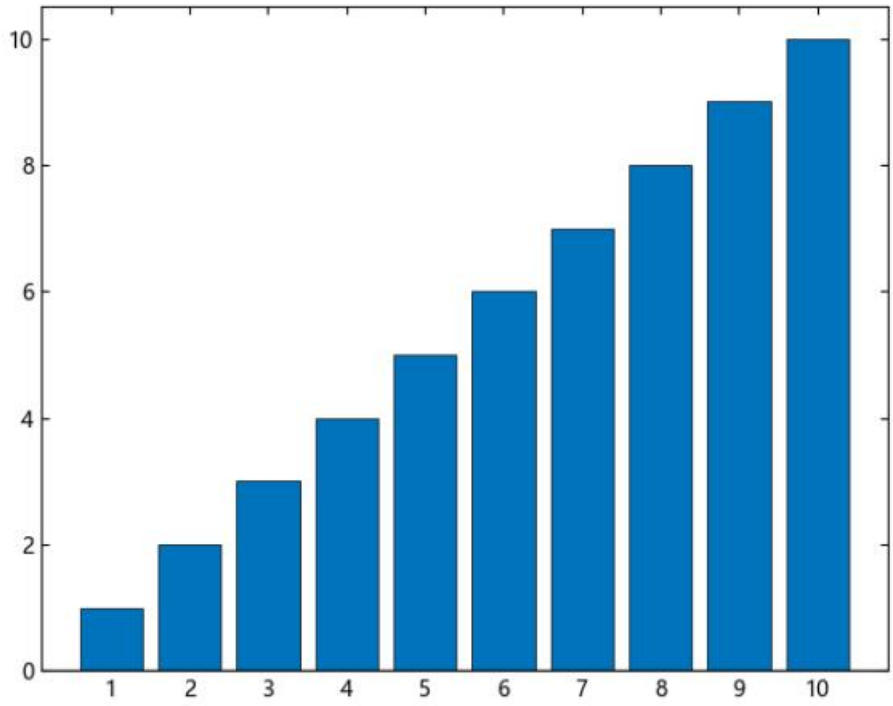
plt_print(fig,___) 保存或打印 fig 指定的图窗。

C. 示例

(1). 打印图窗纸张副本

创建一个条形图并将其输出到系统默认打印机。如果您不指定要打印的图窗，则 print 使用当前图窗。

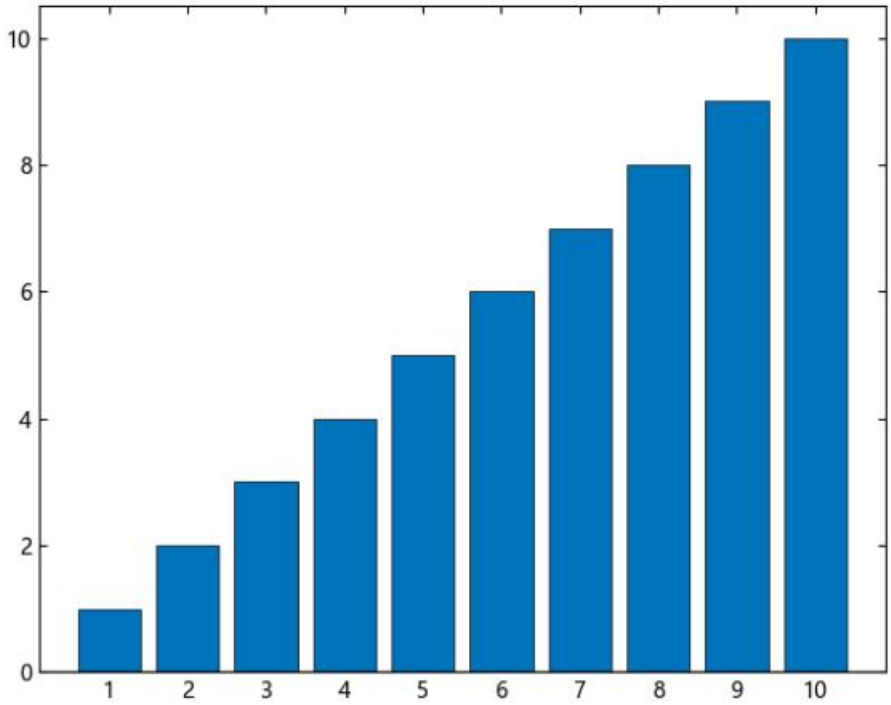
```
bar(1:10)
plt_print()
```



(2). 将图窗另存为图像文件

创建一个绘图并将其另存为 PNG 图像文件。

```
bar(1:10);
plt_print("BarPlot", "-dpng")
```

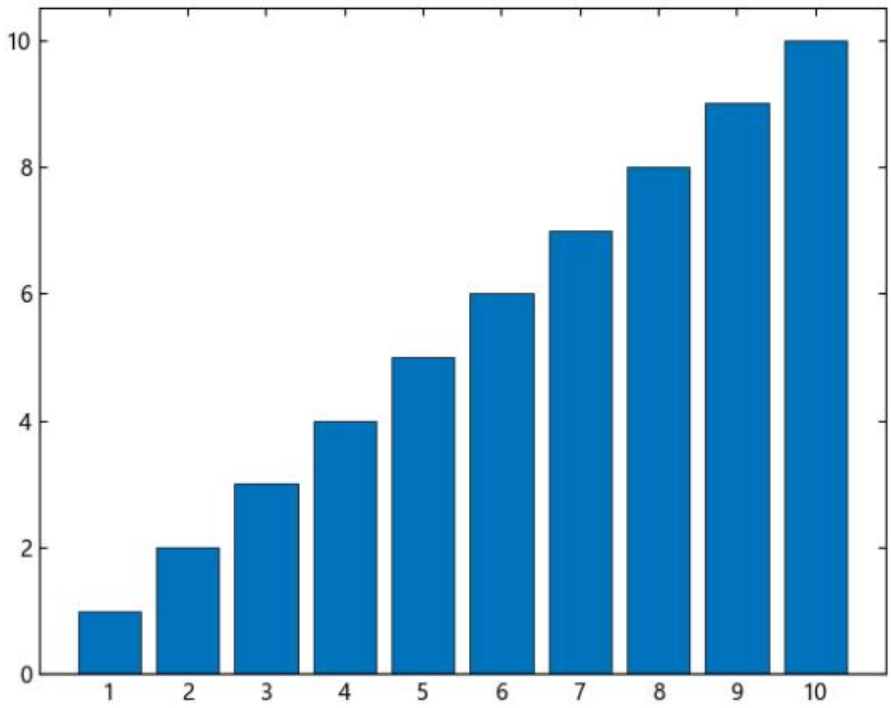


plt_print 将绘图另存为 BarPlot.png。

(3). 将图窗另存为向量图形文件

创建一个绘图并将其另存为封装的 eps 文件。

```
bar(1:10);
plt_print("BarPlot", "-deps")
```

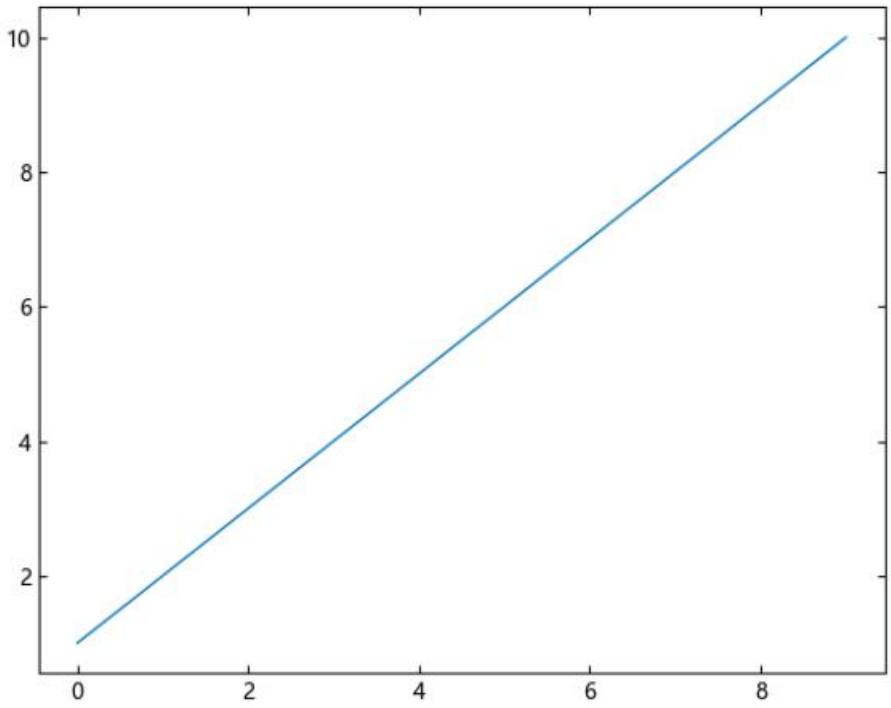


plt_print 将绘图另存为 BarPlot.eps。

(4). 指定要保存的图窗

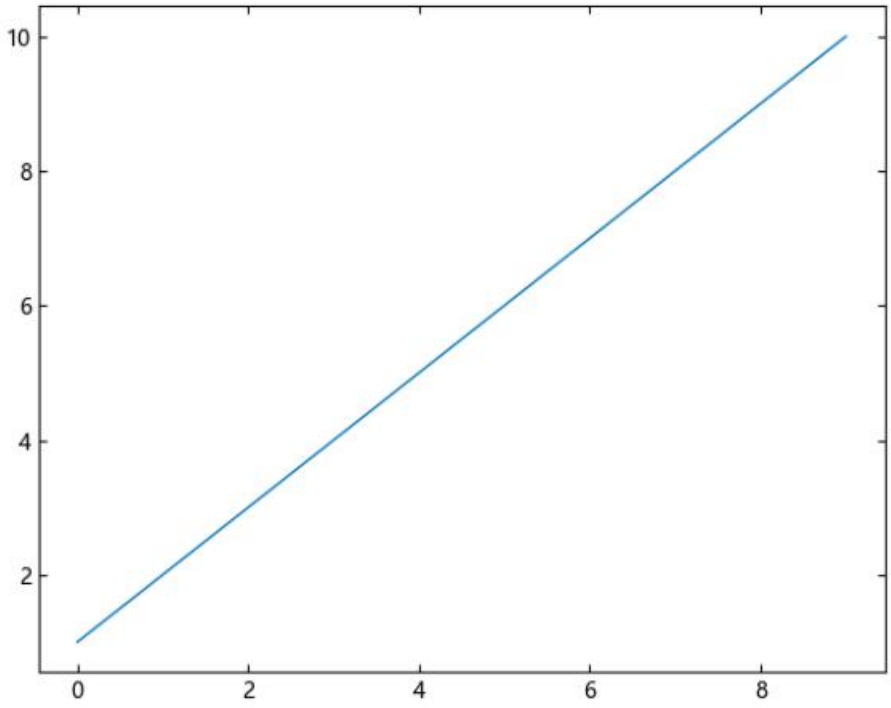
通过将特定图窗的对象变量传递到 plt_print 来保存该图窗。

```
fig = figure();
plot(1:10)
plt_print(fig, "MySavedPlotx", "-dpng")
```



例如，保存图窗并在标题栏中显示 Figure 2。

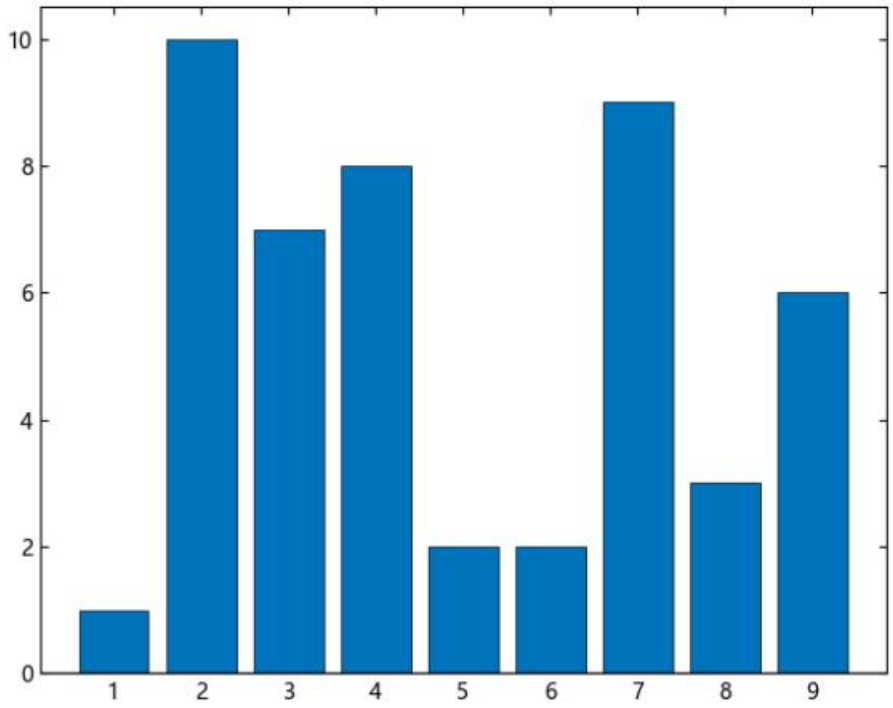
```
f2 = figure(2);
plot(1:10)
plt_print(f2, "MySavedPlot", "-dpng")
```



(5). 保存填满页面的图窗

保存填满页面的图窗。

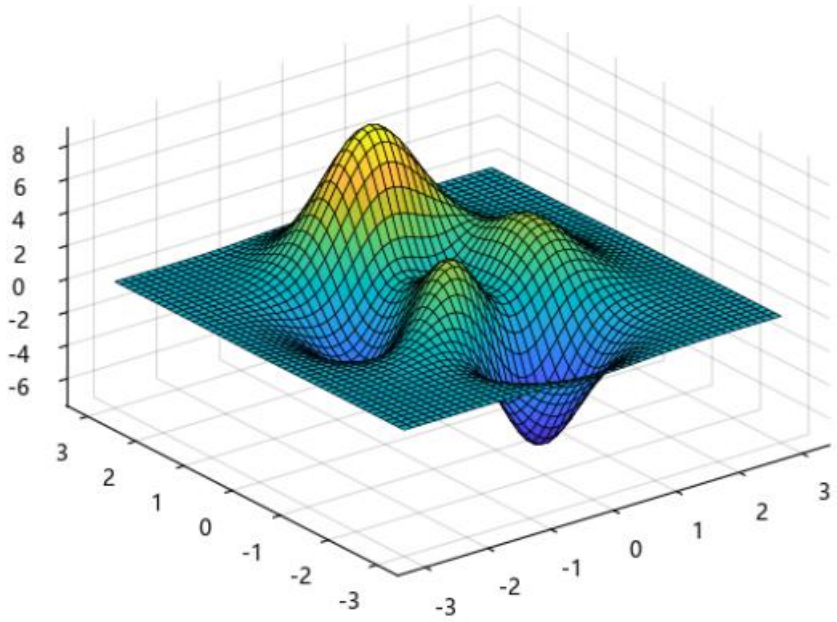
```
bar([1 10 7 8 2 2 9 3 6]);
plt_print("FillPageFigure", "-dpdf")
```



(6). 返回图窗的 RGB 图像数据

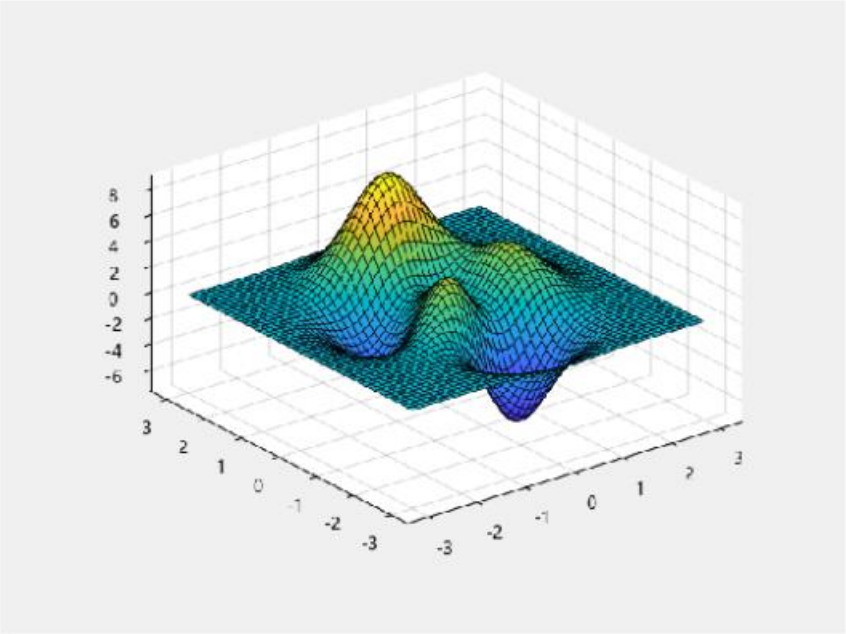
返回图窗的 RGB 图像数据。

```
X, Y, Z = peaks();
surf(X, Y, Z);
cdata = plt_print("-RGBImage");
```



使用 imshow 以完全分辨率显示图像数据。

```
using TyImages
figure();
imshow(cdata)
```

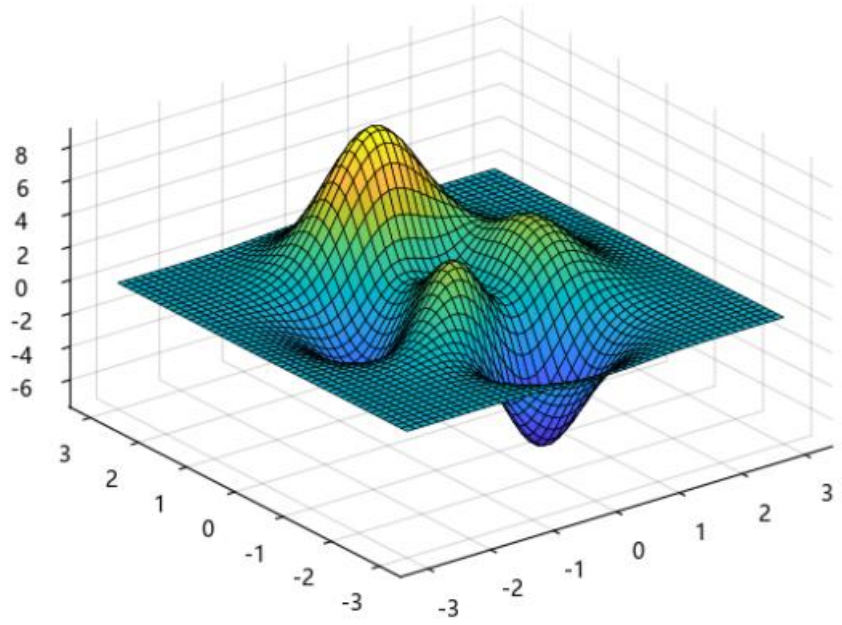


(7). 创建高分辨率影片帧

创建一个曲面图。返回图窗的 RGB 图像数据，并指定图像分辨率。然后，将图像数据转换为影片帧 F。

```
X, Y, Z = peaks();
surf(X, Y, Z);
cdata = plt_print("-RGBImage");

using TyImages
F = im2frame(cdata);
```



3.3.3.2 saveas

将图窗保存为特定文件格式

A. 语法

```
saveas(fig,filename)
```

B. 说明

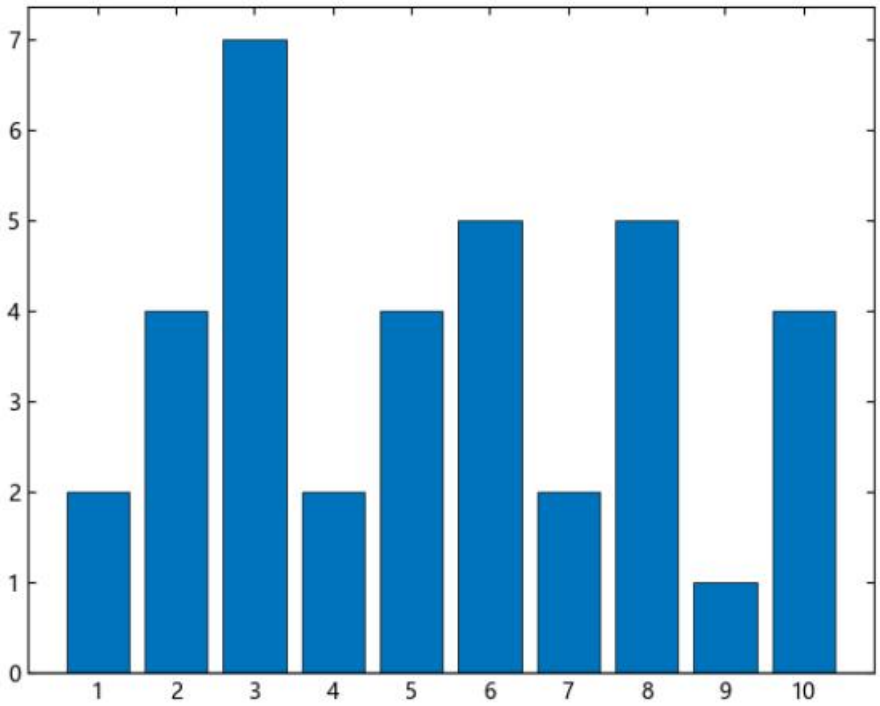
saveas(fig,filename)将 fig 指定的图窗保存到 filename 文件中。将文件名指定为字符向量或字符串，包括文件扩展名，例如"myplot.jpg"。文件扩展名用于定义文件格式。

C. 示例

(1). 将图窗保存为 PNG 文件

创建条形图并将其另存为 PNG 文件。

```
x = [2 4 7 2 4 5 2 5 1 4];
bar(x);
saveas(gcf(), "Barchart.png")
```



3.3.3.3 getframe

捕获坐标区或图窗作为影片帧

A. 语法

```
F = getframe()
F = getframe(fig)
```

B. 说明

F=getframe()捕获显示在屏幕上的当前坐标区作为影片帧。F 是一个包含图像数据的结构体。getframe 按照屏幕上显示的大小捕获这些坐标区。示例

F=getframe(fig)捕获由 fig 标识的图窗。如果您需要捕获图窗窗口的整个内部区域（包括坐标区标题、标签和刻度线），则指定一个图窗。捕获的影片帧不包括图窗菜单和工具栏。示例。

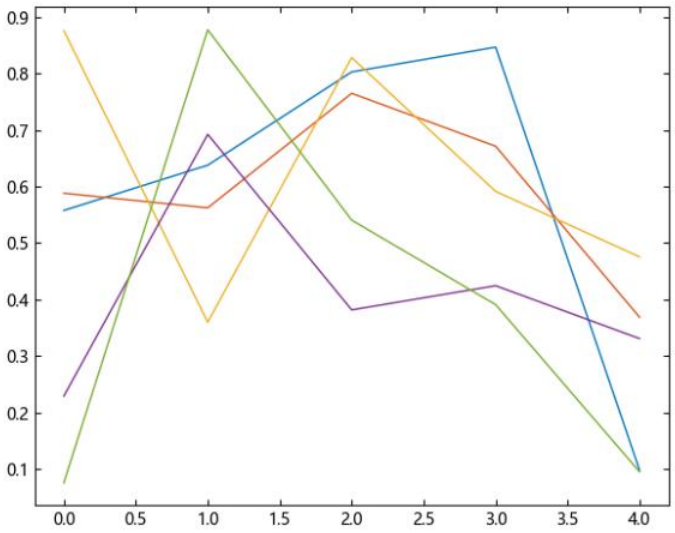
C. 示例

(1). 捕获当前坐标区的内容

创建一个随机数据图。捕获坐标区并返回图像数据。getframe 捕获坐标区的内部区域和坐标区轮廓。它并不捕获延伸到坐标区轮廓之外的内容。

```
using TyImages
```

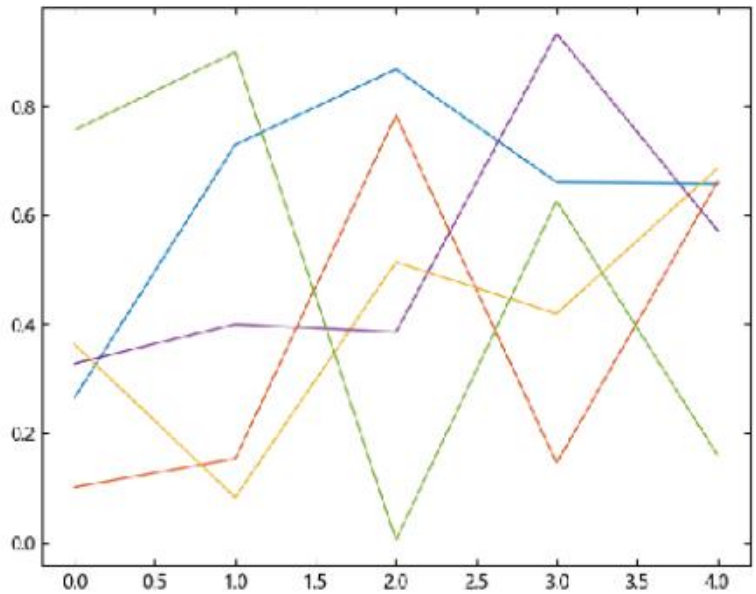
```
plot(rand(5,5))
F = getframe();
```



F 是一个结构体，其中包含的 cdata 字段中包含捕获的图像数据。

使用 imshow 显示捕获的图像数据。

```
figure()
imshow(F.cdata)
```

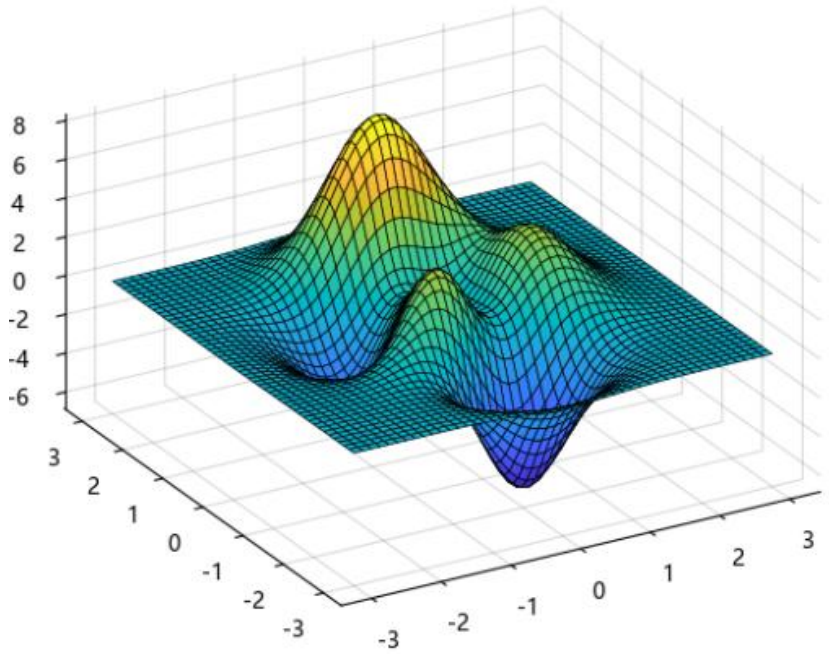


(2). 捕获图窗内容

创建一个曲面图。捕获图窗窗口的内部区域，不包括菜单和工具栏。

```
using TyImages
X,Y,Z = peaks()
```

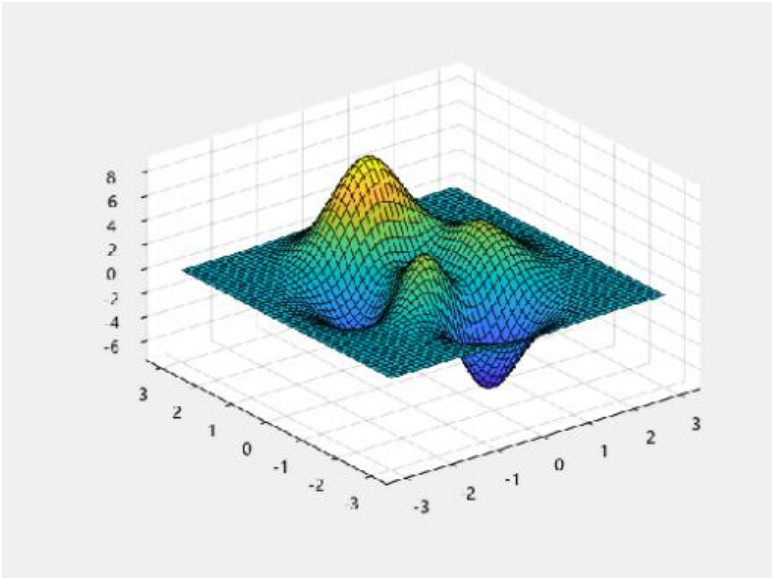
```
surf(X,Y,Z)
F = getframe(gcf());
```



F 是一个结构体，其中包含的 cdata 字段中包含捕获的图像数据。

使用 imshow 显示捕获的图像数据。

```
figure()
imshow(F.cdata)
```



3.3.4 图形对象

通过设置底层对象的属性自定义图形

图形对象是 Syslab 用来创建可视化数据的组件。每个对象在图形显示中都具有特定角色。例如，一个线图包含一个图窗对象、一个坐标区对象和一个图形线条对象。

3.3.4.1 图形对象属性

查看和设置图形对象属性、定义默认值

函数名	简介
get	查询图形对象属性
set	设置图形对象属性

3.3.4.1.1 get

查询图形对象属性

A. 示例

(1). 列出特定对象的所有属性值

创建一个线图并以 p 形式返回图形线条对象。列出线条的所有属性和当前属性值。

```
p, = plot(1:10);
p
```

```
TyPythonCall.PyObject(<py matplotlib.lines.Line2D object at 0x00000000F5671D90>)
```

(2). 查询特定对象的特定属性

创建一个线图并以 p 形式返回图形线条对象。使用 get 可返回 LineWidth 属性的当前值。

```
p, = plot(1:10);
p.get_linewidth()
```

```
1.0
```

(3). 查询特定对象的属性集

使用圆形标记创建一个线图并以 p 形式返回图形线条对象。使用 get 可返

回该对象的 LineWidth、Marker 和 MarkerSize 属性的当前值。

```
p, = plot(1:10,"ro-");
lw = p.get_linewidth()
m = p.get_marker()
ms = p.get_markersize()
```

```
lw = 1.0
m = "o"
ms = 6.0
```

3.3.4.1.2 set

设置图形对象属性

A. 示例

(1). 更改特定线条的颜色

绘制一个线条并以 p 的形式返回图形线条对象。将行的 Color 属性设置为 "red"。

```
p = plot(1:10);
p[1].set_color("red")
```

(2). 更改多个线条的颜色

使用随机数据创建一个包含四个线条的图形，并将这四个线条对象返回至 P。将所有线条的 Color 属性设置为 "red" 。

```
P = plot(rand(4, 4));
for l in P
    l.set_color("red")
end
```

(3). 为多个线条的线型设置为不同的值

将四个图形线条对象的 LineStyle 属性值分别设为不同的值。转置数组值，使其拥有合适的形状。

```
P = plot(rand(4, 4));
ValueArray = ["-", "--", ":", "-."];

for i in 1:length(P)
    P[i].set_linestyle(ValueArray[i])
```

```
end
```

(4). 为多个对象的多个属性设置不同值

将三个不同针状序列对象的 Marker 属性设置为不同值。

```
x = 0:30;
y = [1.5 * cos.(x) 4 * exp.(-.1 * x) .* cos.(x) exp.(0.05 * x) .*
cos.(x)];
y = Matrix(reshape(y, 31, 3))
S = stem(x, y);
ValueArray = ["o"; "s"; "*"];

for i = 1:length(S)
    S[i][1].set_marker(ValueArray[i])
end
```

3.3.4.2 图形对象的标识

查找、复制和删除图形对象

函数名	简介
ancestor	图形对象的父级
plt_delete	删除图形对象
findobj	查找具有特定属性的图形对象
gca	当前坐标区或图
gcf	当前图窗的句柄
isgraphics	对有效的图形对象句柄为 True
ishandle	测试是否有效的图形对象

3.3.4.2.1 ancestor

图形对象的父级

A. 语法

```
p = ancestor(h,type)
p = ancestor(h,type,"toplevel")
```

B. 说明

如果父级是 type 指定的图形对象类型之一，则 p = ancestor(h,type) 返回 h 的最近父级的句柄。type 可以是：

包含单个对象类型的名称的字符串，例如 "figure"。

包含多个对象名称的数组，例如["figure","axes"]。

如果 Syslab 找不到 h 的某种指定类型的父级，则 ancestor 返回空的 p。ancestor 搜索层次结构时，它在搜索中包含对象本身。因此，如果具有句柄 h 的对象属于 type 中列出的类型之一，则 ancestor 将返回对象 h。

如果 h 不是图形对象，则 ancestor 返回空的 p，但不引发错误。

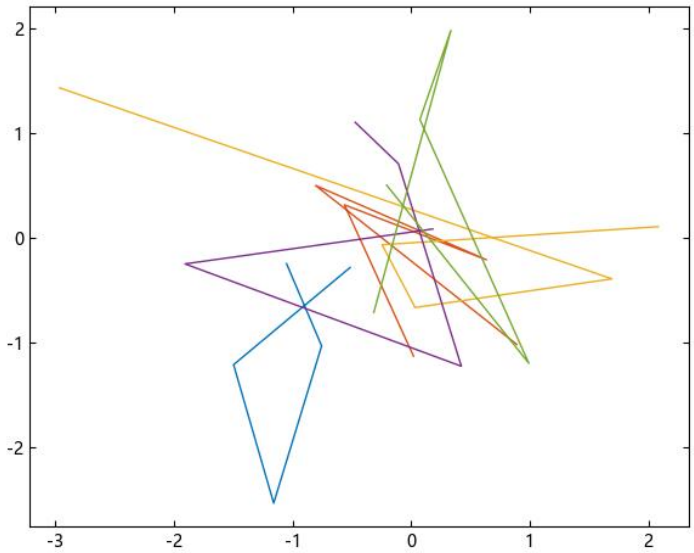
如果此类型出现在 type 参数中，则 p = ancestor(h,type,"toplevel") 返回 h 的最高父级。

C. 示例

(1). 返回顶层父级

创建 Line 对象然后返回顶层父级。

```
ln = line(randn(5,5), randn(5,5));
```




```
tp = ancestor(ln, ["figure","axes"], "toplevel")
```

```
PyObject <Figure size 700x524 with 1 Axes>
```

3.3.4.2.2 lt_delete

删除图形对象

A. 语法

```
plt_delete(obj)
```

B. 说明

plt_delete(obj) 删除指定的对象。如果 obj 是数组，则 plt_delete 将删除数组中的所有对象。obj 会保留在工作区中，但不再有效。示例。

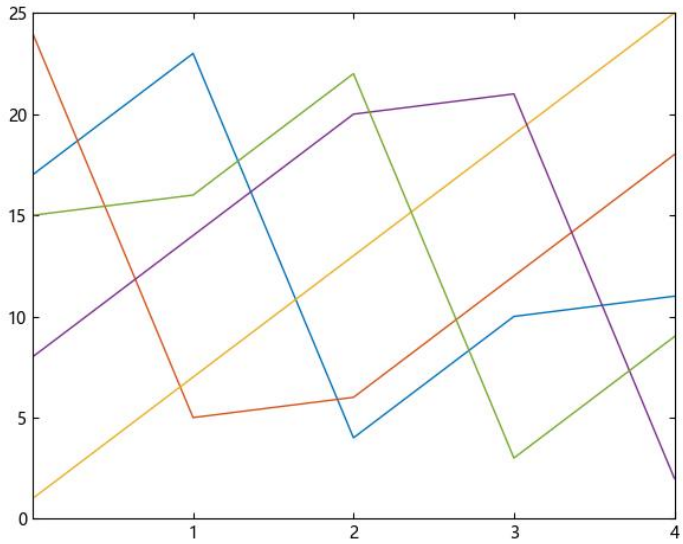
C. 示例

(1). 删除图形对象

删除图形对象和图形对象数组。

创建一个条形图并绘制五条线。然后删除该条形图。

```
using TyMath
b = bar(1:5);
hold("on")
P = plot(magic(5));
plt_delete(b)
```

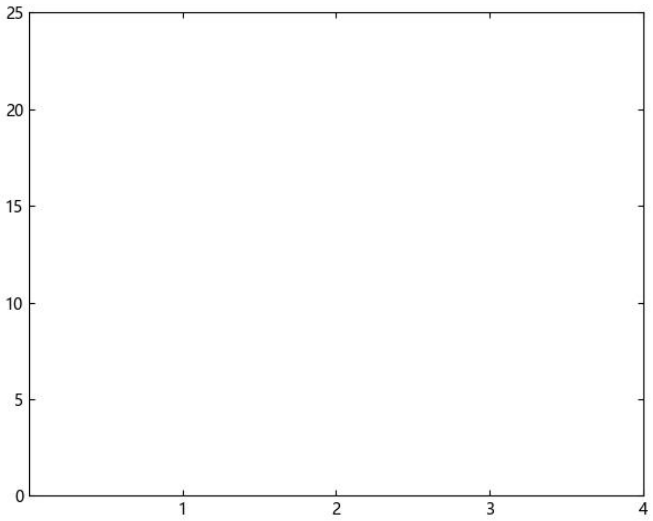


Bar 对象变量 b 仍保留在工作区中，但不再引用对象。

```
display(b)
```

删除 plot 创建的所有 Line 对象。

```
plt_delete(P)
```



3.3.4.2.3 findobj

查找具有特定属性的图形对象

A. 语法

```
h = findobj()
h = findobj(;Key=Value,...)
h = findobj(objhandles,___)
```

B. 说明

h = findobj 返回根对象及其所有子级的句柄。

h = findobj(;Key=Value,...) 返回属性 PropertyName 设置为值 PropertyValue 的所有图形对象的句柄。您可以指定多个属性/值对组，在这种情况下，findobj 将仅返回具有所有指定值的对象。

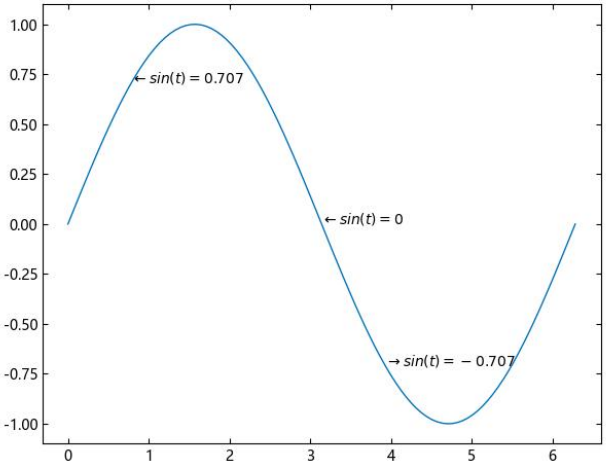
h = findobj(objhandles,...) 将搜索限制为 objhandles 中列出的对象及其子级。

C. 示例

(1). 通过字符串属性查找文本

此示例说明如何使用 String 属性查找文本对象。

以下图形包含用于标记特定函数值的文本对象。



假设要将标记值 sin(t) = .707 的文本从其位于 [pi/4,sin(pi/4)] 的当前位置移动到函数具有相同值的点 [3*pi/4,sin(3*pi/4)]（在图形中呈淡灰色显示）。

要使用 findobj，选取能唯一标识该对象的属性值。

```
x = LinRange(0, 2 * pi, 100)
plot(x, sin.(x))
text(pi / 4, sin(pi / 4), raw"$\leftarrow sin(t) = 0.707$")
text(pi, sin(pi), raw"$\leftarrow sin(t) = 0$")
text(5 * pi / 4, sin(5 * pi / 4), raw"$\rightarrow sin(t) = -
0.707$")
```

使用 findobj 可通过指定层次结构中起始点来限制搜索，而不用从根对象开始。如果对象树中有很多对象，则此功能可以加快搜索速度。在前一示例中，您知道关注的文本对象在当前坐标区上，因此可以键入：

```
findobj(gca(), string = raw"$\leftarrow sin(t) = 0.707$")
```

(2). 限制搜索范围

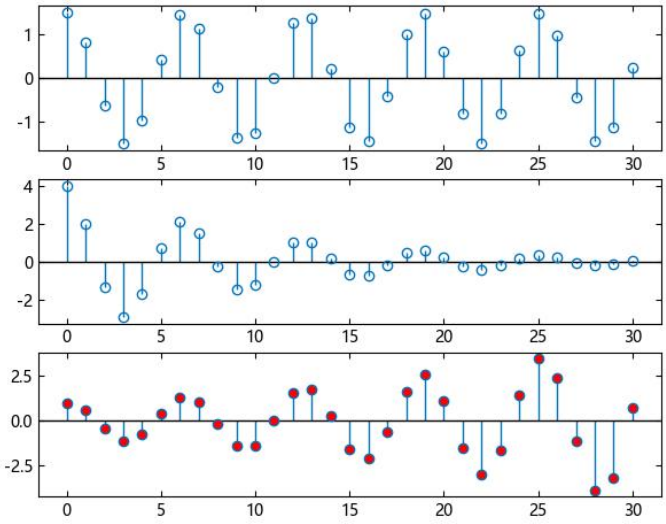
指定对象树中的起始点可以限制搜索范围。起始点可以是图窗、坐标区的句柄或一组对象句柄。

例如，假设您想要更改特定坐标区中针状图的标记面颜色:

```
x = 0:30;
y = [1.5 * cos.(x) 4 * exp.(-.1 * x) .* cos.(x) exp.(0.05 * x) .*
cos.(x)];
ax1 = subplot(3, 1, 1)
stem(x, y[:, 1])
ax2 = subplot(3, 1, 2)
stem(x, y[:, 2])
ax3 = subplot(3, 1, 3)
stem(x, y[:, 3])
```

只设置第三个坐标区上针状图的标记面颜色。

```
h, = findobj(ax3, type="stem");
h[1].set_markerfacecolor("red")
plt_update()
```



3.3.4.2.4 gca

当前坐标区或图

A. 语法

```
ax = gca()
```

B. 说明

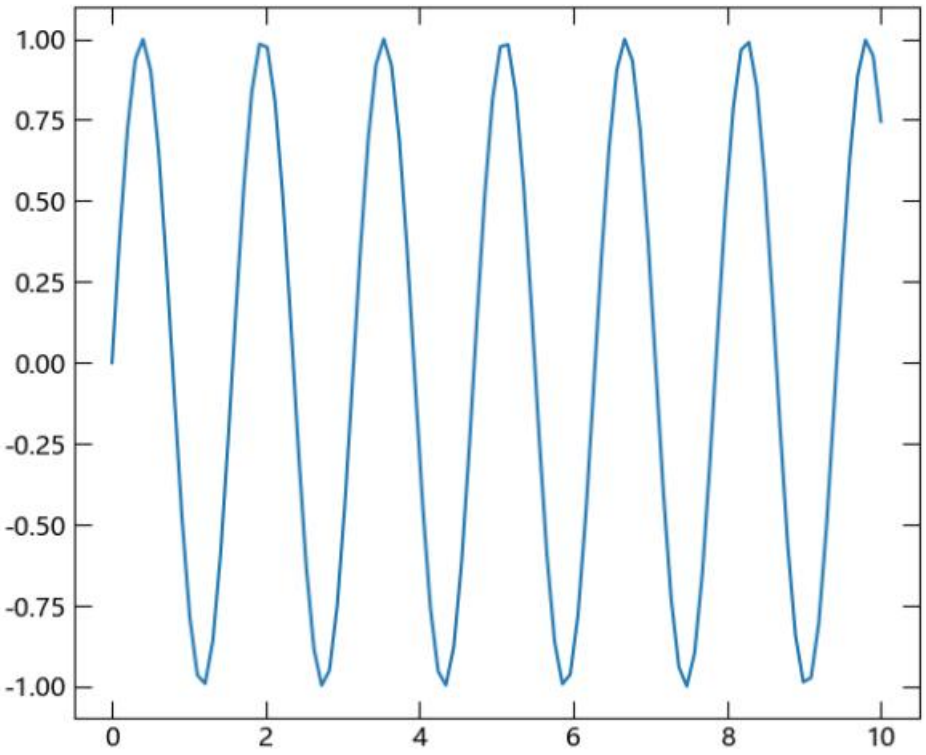
ax=gca()返回当前图窗中的当前坐标区（或独立可视化）。使用 ax 获取和设置当前坐标区的属性。如果当前图窗中没有坐标区或图，则 gca 会创建一个笛卡尔坐标区对象。

C. 示例

(3). 指定当前坐标区的属性

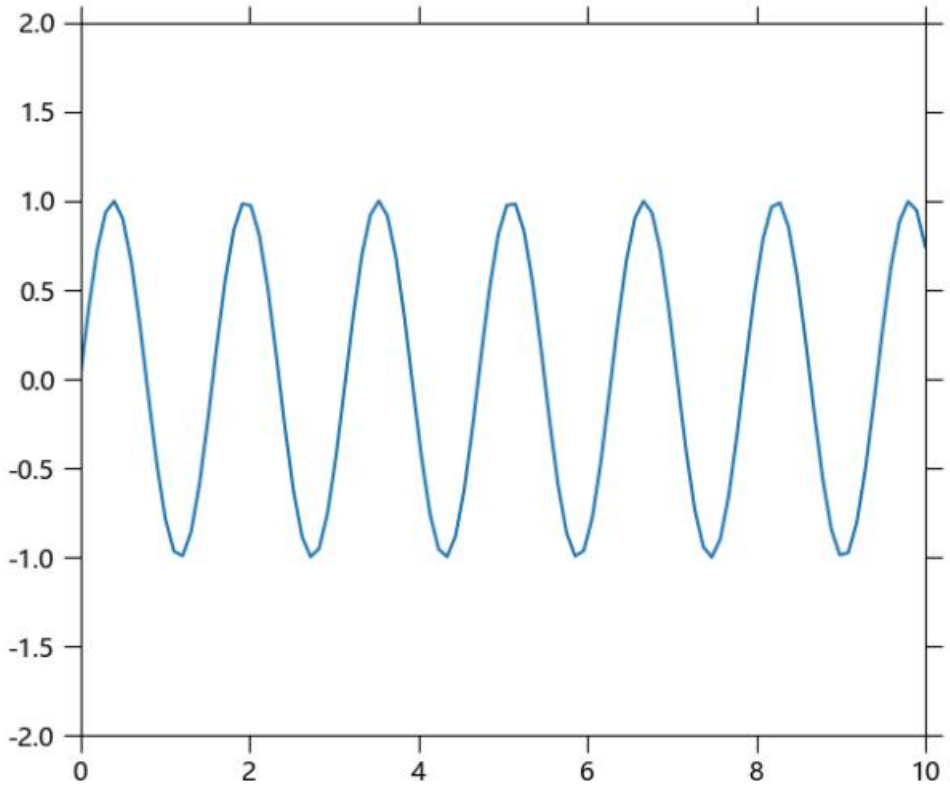
绘制一个正弦波。

```
x = LinRange(0, 10, 100);
y = sin.(4 * x);
plot(x, y)
```



设置当前坐标区的字体大小、刻度方向、刻度长度以及 y 坐标轴范围。使用 gca 指代当前坐标区。

```
ax = gca()
ax.tick_params(labelsize=12)
ax.tick_params(direction="out")
ax.tick_params(length=8)
ax.set_ylim(-2,2)
ax.set_xlim(0,10)
```



3.3.4.2.5 gcf

当前图窗的句柄

A. 语法

```
fig = gcf()
```

B. 说明

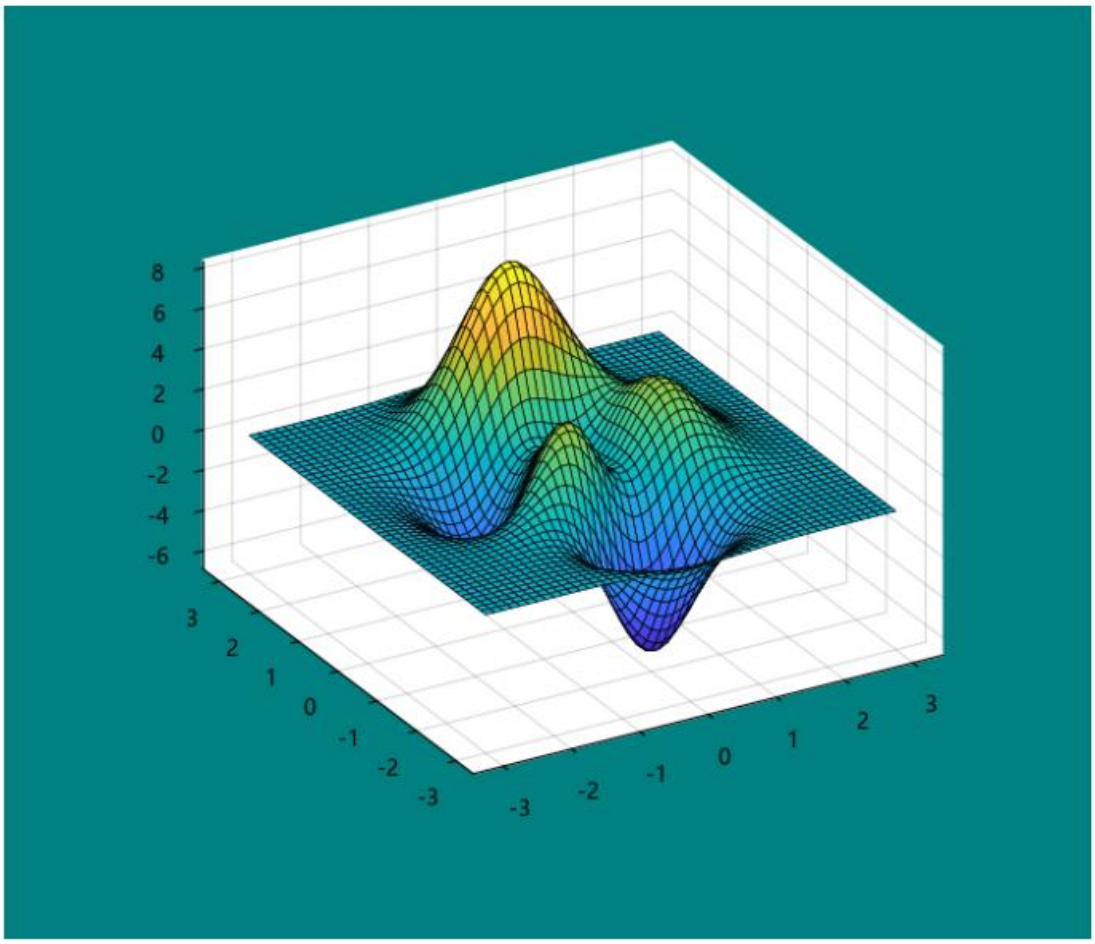
fig = gcf() 返回当前图窗的句柄。如果图窗不存在，则 gcf 将创建一个图窗并返回其句柄。可以使用图窗句柄查询和修改图窗的属性。

C. 示例

(1). 指定当前图窗的属性

设置当前图窗的背景色并删除工具栏。使用 gcf 命令获取当前的图窗句柄。

```
X,Y,Z = peaks()
surf(X,Y,Z)
fig = gcf();
fig.set_facecolor([0,0.5,0.5])
fig.canvas.manager.toolbar.hide()
```



3.3.4.2.6 isgraphics

对有效的图形对象句柄为 True

A. 语法

```
tf = isgraphics(H)
tf = isgraphics(H,type)
```

B. 说明

tf = isgraphics(H) 为 H 中属于有效图形对象的元素返回 true，为不是有效图形对象或是已删除图形对象的元素返回 false。

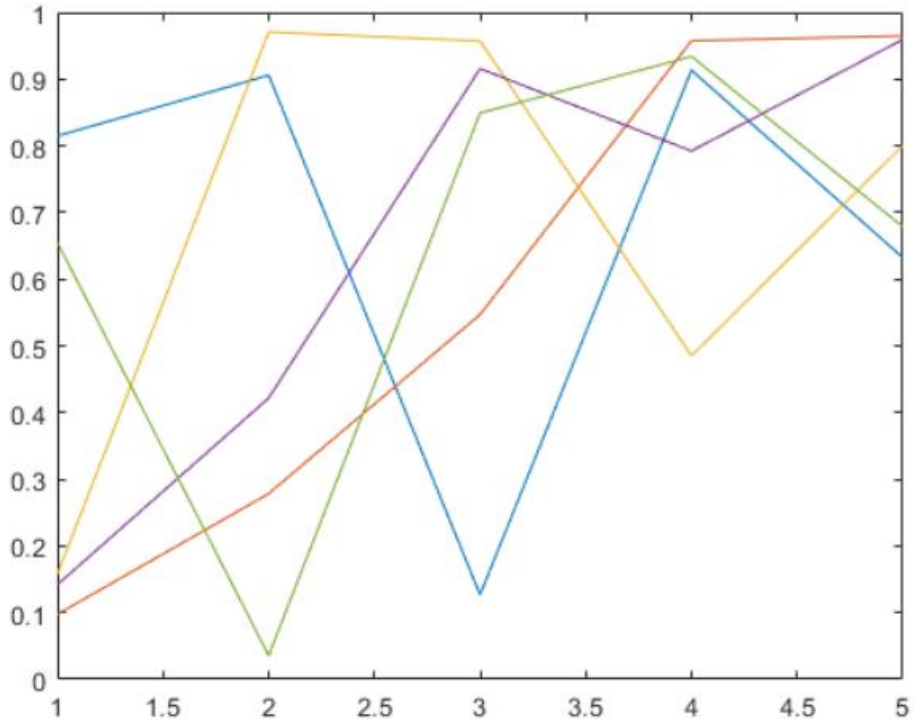
对于属于由 type 参数所指定类型的有效图形对象的 H 元素，tf = isgraphics(H,type) 返回 true。对象类型是对象的 Type 属性中包含的值。

C. 示例

(1). 测试是否为有效句柄

创建一个绘图并返回句柄数组。测试数组是否为有效句柄。

```
H = plot(rand(5,5));
```



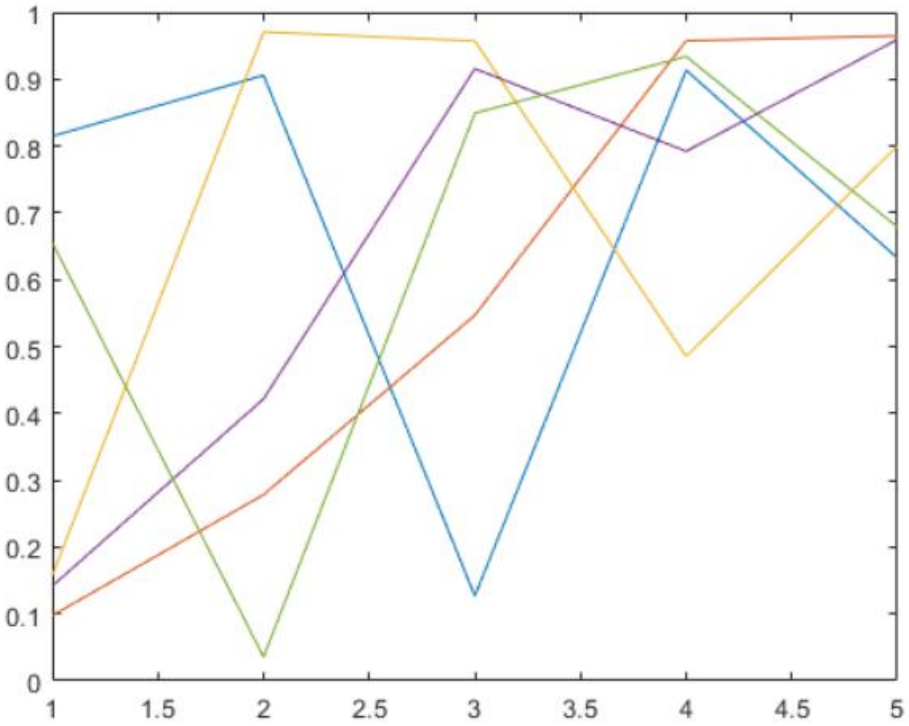

```
isgraphics(H)
```

```
ans =
5-element BitVector:
 1
 1
 1
 1
 1
```

(2). 测试句柄类型

创建一个绘图并返回句柄数组。与其他图形对象串联并测试句柄是否为 Line 类型。

```
H = plot(rand(5, 5));
```




```
a = [H; gca(); gcf()]
isgraphics(a, "line")
```

```
ans =
7-element Vector{Bool}:
 1
 1
 1
 1
 1
 0
 0
```

3.3.4.2.7 ishandle

测试是否有效的图形对象

A. 语法

```
ishandle(H)
```

B. 说明

如果 H 的元素是图形对象，ishandle(H) 将返回一个元素为 1 的数组；否则，将返回 0。

C. 示例

(1). 测试是否为有效句柄

测试是否为有效句柄。

```
H = plot(rand(5, 5));
B = ishandle(H)
```

3.3.4.3 图形对象编程

对对象数组进行比较、有效性测试、预分配和使用

函数名	简介
clf	清空图窗
cla	清除坐标区
plt_close	删除指定图窗

3.3.4.3.1 clf

清空当前图窗窗口

A. 语法

```
clf()
clf("reset")
clf(fig)
clf(fig,"reset")
figure_handle = clf(__)
```

B. 说明

clf() 从当前图窗中删除其句柄未隐藏的所有图形对象。

clf("reset") 从当前图窗中删除所有图形对象，无论它们是否可见。

clf(fig) 或 clf(fig,"reset") 清除其句柄为 fig 的单个图窗。

figure_handle = clf(__) 返回图窗句柄。

3.3.4.3.2 cla

详情见图形 API->格式和注释->坐标区外观->清除或创建坐标系->cla

3.3.4.3.3 plt _close

删除指定图窗

A. 语法

```
plt_close()
plt_close(h)
plt_close(name)
plt_close("all")
status = plt_close(__)
```

B. 说明

plt_close() 删除当前图窗或指定图窗。它可以选择返回关闭操作的状态。

plt_close() 删除当前图窗（等同于 plt_close(gcf())）。

plt_close(h) 删除由 h 标识的图窗。如果 h 是数组，plt_close 会删除 h 标识的所有图窗。h 还可以是图窗 Number。

plt_close(name) 删除具有指定名称的图窗。

plt_close("all") 删除其句柄未隐藏的所有图窗。

如果指定窗口已被删除，status = plt_close(__) 返回 1，否则返回 0。

C. 示例

(1). 删除图窗

删除当前图窗或指定图窗。

```
plot(sin.(1:0.01:2*pi))
plt_close()
```

3.3.4.4 指定图形输出的目标

控制目标图窗和轴以及如何更新这些对象

函数名	简介
hold	添加新绘图时保留当前绘图

ishold	当前保留状态
clf	清空图窗
cla	清除坐标区
newplot	确定图形对象的绘制位置

3.3.4.4.1 hold

详情见图形 API->格式和注释->坐标区外观->多个绘图->cla

3.3.4.4.2 ishold

当前保留状态

A. 语法

```
tf = ishold()
tf = ishold(ax)
```

B. 说明

tf = ishold() 返回当前坐标区的保留状态。如果 hold 是 "on"，返回状态为 1；如果它是 "off"，则返回状态为 0。如果 hold 为 "on"，会保留当前绘图和大多数的坐标区属性，以便将后续绘图命令添加到现有图形中。如果没有图窗，ishold 返回 0 且 Syslab 创建一个新图窗。

tf = ishold(ax) 返回指定坐标区对象的保留状态，该对象可以是 Axes、PolarAxes 对象。

hold on 的状态表示图窗和坐标区的 NextPlot 属性都设置为 "add"。

3.3.4.4.3 clf

详情见图形 API->图像对象->图形对象编程->clf

3.3.4.4.4 cla

详情见图形 API->格式和注释->坐标区外观->清除或创建坐标系->cla

3.3.4.4.5 newplot

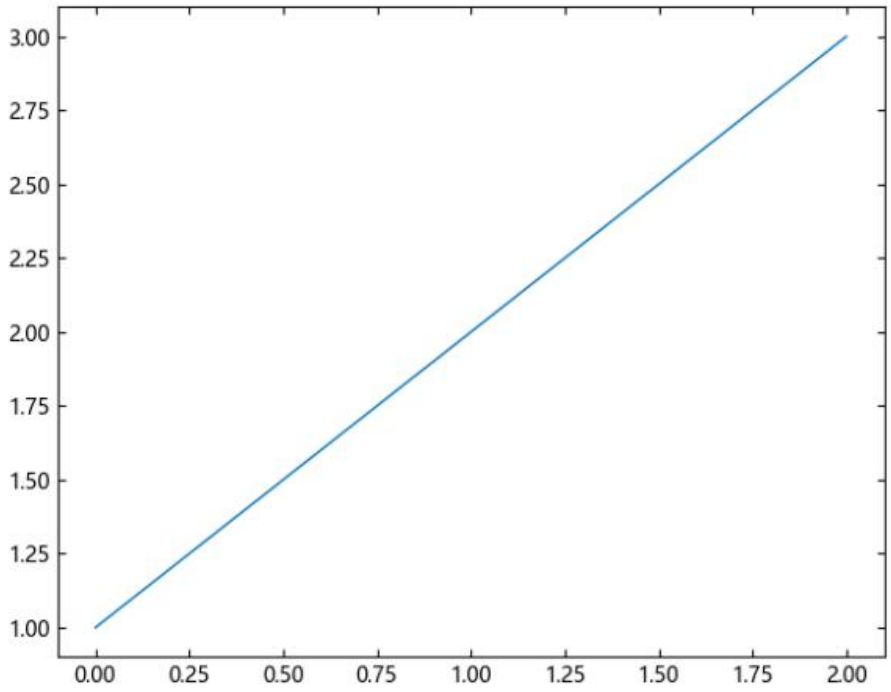
确定图形对象的绘制位置

A. 示例

(1). 清空指定坐标轴

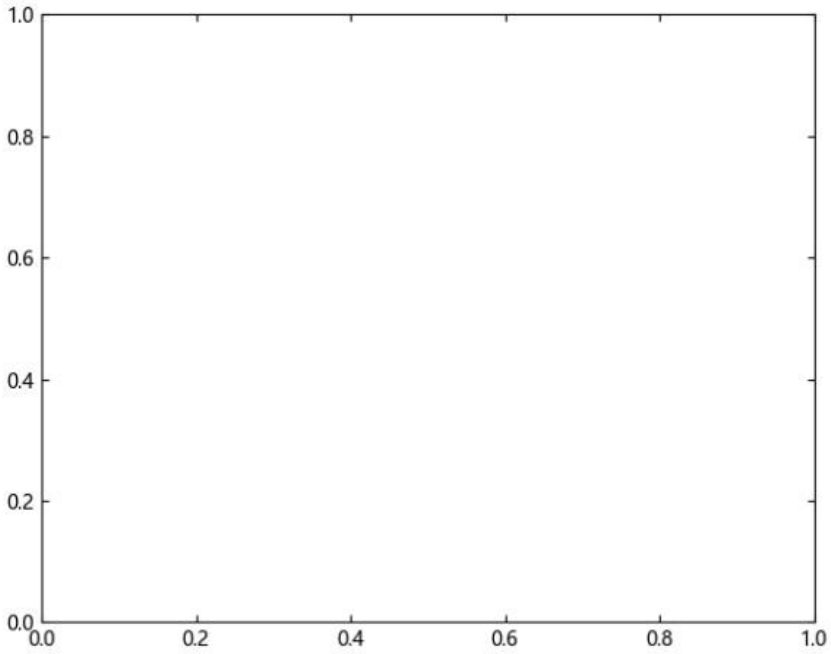
创建一个线图。

```
f = figure()
a = plt_axes()
plot([1, 2, 3])
```



清空图窗 a。

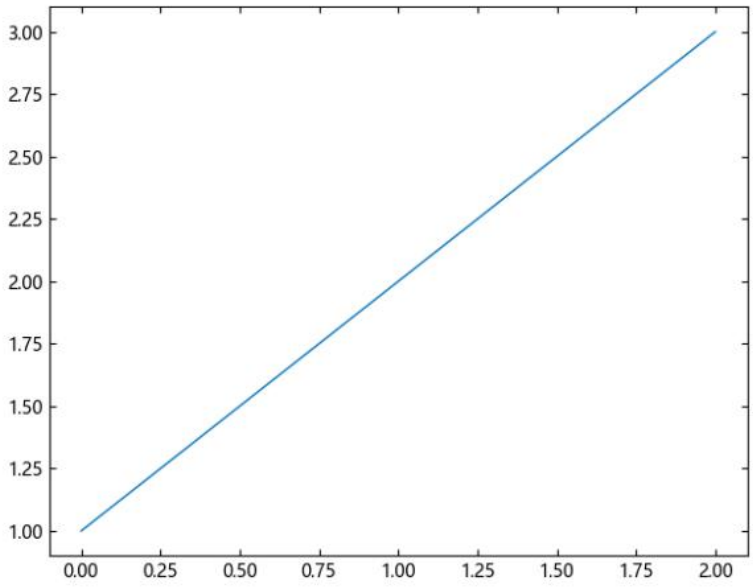
```
cla(a)
```



(2). 清空当前图窗绘制新的坐标轴

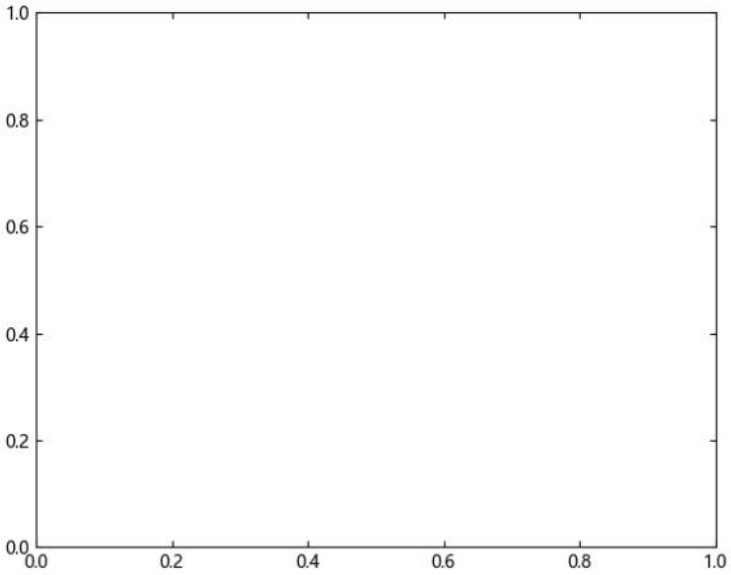
创建一个线图。

```
f = figure()
a = plt_axes()
plot([1, 2, 3])
```



清空整个图窗。

```
clf()
plt_axes()
```



3.4 App 构架 API

3.4.1 使用 App 设计工具

建设中...

3.4.2 以编程方式开发 App

建设中...

3.4.3 App 管理

定义了一套在科学计算环境中安装、卸载、运行 Apps 的 Julia 语言接口。

函数名	简介
init_syslabapp	初始化 App 环境
AppInfo	App 模型定义
install	注册并安装 App
uninstall	卸载名称为 name 的 App
get_apps	查询用户注册的所有 App 列表信息
get_app	查询名称为 name 的 App 的信息

start	启动名称为 name 的 App
disable	禁用名称为 name 的 App
enable	启用名称为 name 的 App

3.4.3.1 init_syslabapp

初始化 App 环境，加载 App 的注册函数等上下文。

A. 语法

```
init_syslabapp(name::AbstractString="all")
```

B. 参数

[in] name　　App 名称，可以为空，为空时将初始化所有的 App

C. 返回值

true-成功，false-失败

D. 示例

```
julia> init_syslabapp(cftool_info)
true
```

3.4.3.2 AppInfo

App 描述信息结构体。

A. 语法

```
struct AppInfo
    name::AbstractString
    path::AbstractString
    registerFunc::AbstractString
    version::AbstractString
    author::AbstractString
    description::AbstractString
```

```
    enable::Bool
end
```

B. 参数

name App 名称，必填项，唯一主键
path App 文件路径，必填项
registerFunc App 启动注册函数的文件路径
version App 版本信息
author App 作者
description App 描述信息
enable::Bool 使能状态，true：激活，false：失效

C. 示例

```
julia> cftool_info = SyslabApp.AppInfo("mycftool",
    raw"${SYSLAB_HOME}/Examples/10 AppDemos/x64/Release/CurveFitTool.exe",
    raw"${SYSLAB_HOME}/Examples/10 AppDemos/x64/Release/mycftool.jl",
    "1.0",
    "Tongyuan",
    "曲线拟合工具箱")
```

3.4.3.3 install

注册并安装 App

A. 语法

```
SyslabApp.install(info::AppInfo)::Bool
```

B. 参数

[in] info App 信息

C. 返回值

true-成功，false-失败

D. 示例

```
julia> SyslabApp.install(cftool_info)
```

```
true
```

3.4.3.4 uninstall

卸载名称为 name 的 App

A. 语法

```
SyslabApp.uninstall(name::AbstractString)::Bool
```

B. 参数

[in] name App 名称

C. 返回值

true-成功，false-失败

D. 示例

```
julia> SyslabApp.uninstall("mycftool")
true
```

3.4.3.5 get_apps

查询用户注册的所有 App 列表信息

A. 语法

```
SyslabApp.get_Apps()::Vector{AppInfo}
```

B. 参数

无

C. 返回值

数组类型的 App 列表，包含所有 App 的详情

D. 示例

```
julia> SyslabApp.get_apps()
Dict{String, Any} with 1 entry:
  "mycftool"                        =>                        Dict{String,
Any}("registerFunc"=>"\${SYSLAB_HOME}/Examples/10
AppDemos/x64/Release/mycftool.jl", "name"=>"mycftool…
```

3.4.3.6 get_app

查询名称为 name 的 App 的信息

A. 语法

```
SyslabApp.get_app(name::AbstractString)::AppInfo
```

B. 参数

name App 名称

C. 返回值

App 的详情

D. 示例

```
julia> SyslabApp.get_app("mycftool")
Dict{String, Any} with 8 entries:
  "registerFunc"            =>            "\${SYSLAB_HOME}/Examples/10
AppDemos/x64/Release/mycftool.jl"
  "name"         => "mycftool"
  "author"       => "Tongyuan"
  "enable"       => true
  "runScript"    => "SyslabApp.start(\"mycftool\")"
  "version"      => "1.0"
  "path"                              =>    "\${SYSLAB_HOME}/Examples/10
AppDemos/x64/Release/CurveFitTool.exe"
  "description"  => "曲线拟合工具箱"
```

3.4.3.7 start

启动名称为 name 的 App

A. 语法

```
SyslabApp.start(name::AbstractString)::Bool
```

B. 参数

name App 名称

C. 返回值

true-成功，false-失败

D. 示例

```
julia> SyslabApp.start("mycftool")
true
```

3.4.3.8 disable

禁用名称为 name 的 App，App 禁用后不可运行

A. 语法

```
SyslabApp.disable(name::AbstractString)::Bool
```

B. 参数

name App 名称

C. 返回值

true-成功，false-失败

D. 示例

```
julia> SyslabApp.disable("mycftool")
true

julia> SyslabApp.start("mycftool")
 Warning: mycftool is disabled!
false
```

3.4.3.9 enable

启用名称为 name 的 App

A. 语法

```
SyslabApp.enable(name::AbstractString)::Bool
```

B. 参数

name App 名称

C. 返回值

true-成功，false-失败

D. 示例

```
julia> SyslabApp.enable("mycftool")
true

julia> SyslabApp.start("mycftool")
true
```

3.4.4 App 通信

平台提供与 Syslab 通信的 Sdk，支持 App 从 Syslab 工作空间中获取数据、将数据写入 Syslab 工作空间、运行 julia 脚本并获取结果。

函数名	简介
MwGetVariables	初始化 App 环境
MwGetValue	查询工作区变量 var 的值
MwRunScript	在工作区运行脚本，生成结果
MwDeleteVar	删除工作区单个变量 var

3.4.4.1 MwGetVariables

查询工作区变量名列表

A. 语法

```
bool MwGetVariables(std::vector<std::string>& variables)
```

B. 参数

[out]variables 变量名列表

C. 返回值

true-成功，false-失败

D. 示例

```
vector<string> variables;
bool res = MwGetVariables(variables);
```

3.4.4.2 MwGetValue

查询工作区变量 var 的值

A. 语法

```
bool MwGetValue(const std::string& var, std::vector<double>& vals)
```

B. 参数

[in] var 变量名
[out]vals 变量值

C. 返回值

true-成功，false-失败

D. 示例

```
vector<double> vals;
bool res = MwGetValue("var1", vals);
```

3.4.4.3 MwRunScript

在工作区运行脚本，生成结果

A. 语法

```
std::string MwRunScript(const std::string& code)
```

B. 参数

[in] code 要运行的脚本

C. 返回值

脚本 code 运行后的结果

D. 示例

```
string script =
    "using TyCurveFitting
    f = TyCurveFitting.fit(\"poly3\", cdate, pop)
    res = f(cdate)";
string res = MwRunScript(script );
```

3.4.4.4 MwDeleteVar

删除工作区单个变量 var

A. 语法

```
bool MwDeleteVar(const std::string& var)
```

B. 参数

[in] var 变量名

C. 返回值

脚本 code 运行后的结果

D. 示例

```
 bool res = MwDeleteVar("var1");
```