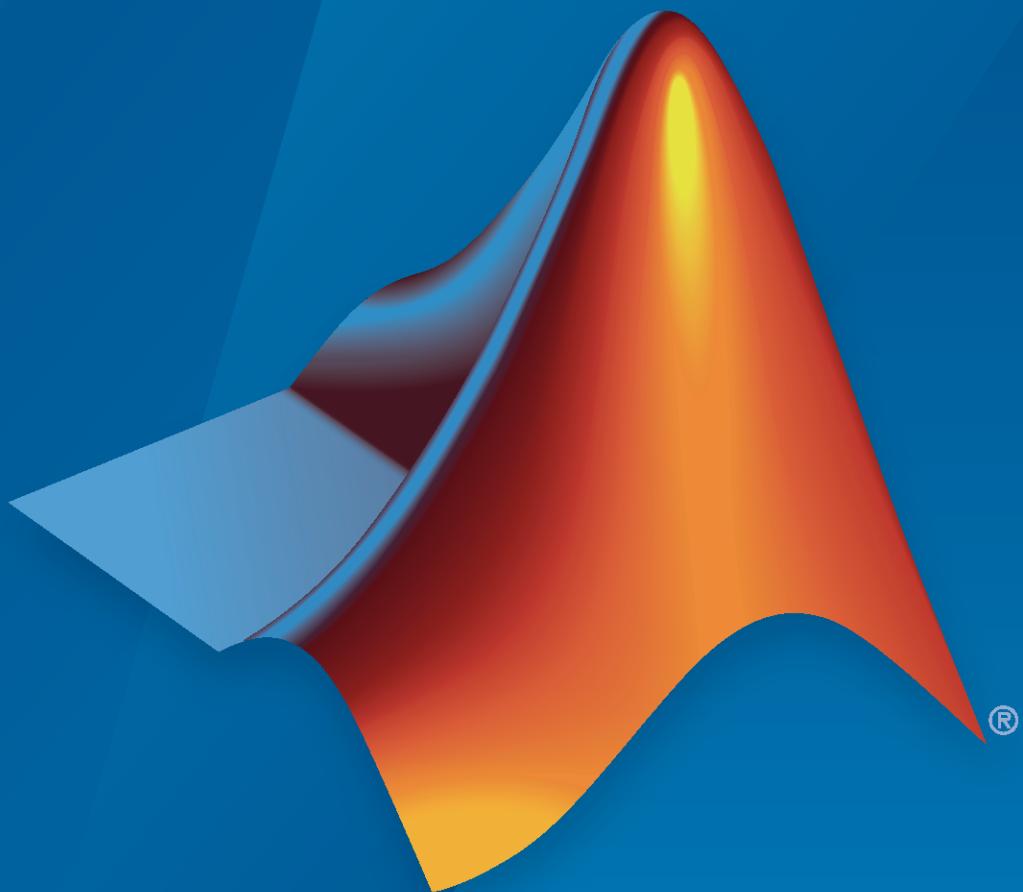


MATLAB®

图形



®

MATLAB®

R2021b

 MathWorks®

如何联系 MathWorks



最新动态: www.mathworks.com



销售和服务: www.mathworks.com/sales_and_services



用户社区: www.mathworks.com/matlabcentral



技术支持: www.mathworks.com/support/contact_us



电话: 010-59827000



迈斯沃克软件(北京)有限公司
北京市朝阳区望京东园四区6号楼
北望金辉大厦16层1604

MATLAB® 图形

© COPYRIGHT 1984–2021 by The MathWorks, Inc.

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from The MathWorks, Inc.

FEDERAL ACQUISITION: This provision applies to all acquisitions of the Program and Documentation by, for, or through the federal government of the United States. By accepting delivery of the Program or Documentation, the government hereby agrees that this software or documentation qualifies as commercial computer software or commercial computer software documentation as such terms are used or defined in FAR 12.212, DFARS Part 227.72, and DFARS 252.227-7014. Accordingly, the terms and conditions of this Agreement and only those rights specified in this Agreement, shall pertain to and govern the use, modification, reproduction, release, performance, display, and disclosure of the Program and Documentation by the federal government (or other entity acquiring for or through the federal government) and shall supersede any conflicting contractual terms or conditions. If this License fails to meet the government's needs or is inconsistent in any respect with federal procurement law, the government agrees to return the Program and Documentation, unused, to The MathWorks, Inc.

商标

MATLAB and Simulink are registered trademarks of The MathWorks, Inc. See www.mathworks.com/trademarks for a list of additional trademarks. Other product or brand names may be trademarks or registered trademarks of their respective holders.

专利

MathWorks products are protected by one or more U.S. patents. Please see www.mathworks.com/patents for more information.

修订历史记录

2006 年 3 月	仅限在线版本	MATLAB® 7.2 (版本 2006a) 中的新内容
2006 年 9 月	仅限在线版本	MATLAB® 7.3 (版本 2006b) 中的修订内容
2007 年 3 月	仅限在线版本	MATLAB® 7.4 (版本 2007a) 中的修订内容
2007 年 9 月	仅限在线版本	MATLAB® 7.5 (版本 2007b) 中的修订内容
2008 年 3 月	仅限在线版本	MATLAB® 7.6 (版本 2008a) 中的修订内容 本出版物以前是“使用 MATLAB® 图形指南”的一部分。
2008 年 10 月	仅限在线版本	MATLAB® 7.7 (版本 2008b) 中的修订内容
2009 年 3 月	仅限在线版本	MATLAB® 7.8 (版本 2009a) 中的修订内容
2009 年 9 月	仅限在线版本	MATLAB® 7.9 (版本 2009b) 中的修订内容
2010 年 3 月	仅限在线版本	MATLAB® 7.10 (版本 2010a) 中的修订内容
2010 年 9 月	仅限在线版本	MATLAB® 7.11 (版本 2010b) 中的修订内容
2011 年 4 月	仅限在线版本	MATLAB® 7.12 (版本 2011a) 中的修订内容
2011 年 9 月	仅限在线版本	MATLAB® 7.13 (版本 2011b) 中的修订内容
2012 年 3 月	仅限在线版本	MATLAB® 7.14 (版本 2012a) 中的修订内容
2012 年 9 月	仅限在线版本	MATLAB 8.0 (版本 2012b) 中的修订内容
2013 年 3 月	仅限在线版本	MATLAB 8.1 (版本 2013a) 中的修订内容
2013 年 9 月	仅限在线版本	MATLAB 8.2 (版本 2013b) 中的修订内容
2014 年 3 月	仅限在线版本	MATLAB 8.3 (版本 2014a) 中的修订内容
2014 年 10 月	仅限在线版本	MATLAB 8.4 (版本 2014b) 中的修订内容
2015 年 3 月	仅限在线版本	MATLAB 8.5 (版本 2015a) 中的修订内容
2015 年 9 月	仅限在线版本	MATLAB 8.6 (版本 2015b) 中的修订内容
2016 年 3 月	仅限在线版本	MATLAB 9.0 (版本 2016a) 中的修订内容
2016 年 9 月	仅限在线版本	MATLAB 9.1 (版本 2016b) 中的修订内容
2017 年 3 月	仅限在线版本	MATLAB 9.2 (版本 2017a) 中的修订内容
2017 年 9 月	仅限在线版本	MATLAB 9.3 (版本 2017b) 中的修订内容
2018 年 3 月	仅限在线版本	MATLAB 9.4 (版本 2018a) 中的修订内容
2018 年 9 月	仅限在线版本	MATLAB 9.5 (版本 2018b) 中的修订内容
2019 年 3 月	仅限在线版本	MATLAB 9.6 (版本 2019a) 中的修订内容
2019 年 9 月	仅限在线版本	MATLAB 9.7 (版本 2019b) 中的修订内容
2020 年 3 月	仅限在线版本	MATLAB 9.8 (版本 2020a) 中的修订内容
2020 年 9 月	仅限在线版本	MATLAB 9.9 (版本 2020b) 中的修订内容
2021 年 3 月	仅限在线版本	MATLAB 9.10 (版本 2021a) 中的修订内容
2021 年 9 月	仅限在线版本	MATLAB 9.11 (版本 2021b) 中的修订内容

线图

1

MATLAB 绘图类型	1-2
创建常见的二维图	1-4
创建二维线图	1-13
创建带标记的线图	1-18
在线图中添加标记	1-18
指定标记大小和颜色	1-19
控制沿线条的标记放置	1-20
在最大数据点和最小数据点处显示标记	1-21
恢复为默认标记位置	1-22
支持的标记符号	1-24
使用两个 y 轴合并线图和条形图	1-26
合并线图和针状图	1-29
叠加阶梯图和线图	1-32
带有置信边界的线图	1-34
绘制虚数和复数数据图	1-35

饼图、条形图和直方图

2

条形图种类	2-2
修改条形图的基线	2-8
叠加条形图	2-11
带有误差条的条形图	2-15
按高度为三维条形着色	2-16
使用叠加区域图对比数据集	2-18
偏移占比最大的饼图扇区	2-22

向饼图添加图例	2-24
为饼图添加文本和百分比标签	2-26
使用二元直方图进行颜色分析	2-28
控制分类直方图的显示	2-34
替换不建议使用的 hist 和 histc 实例	2-41
旧直方图函数 (hist 和 histc)	2-41
推荐的直方图函数	2-41
要求代码更新的差异	2-41

极坐标图

3

在极坐标中绘图	3-2
自定义极坐标区	3-14
极坐标区上的罗盘标签	3-22

等高线图

4

为等高线图添加层级标签	4-2
改变等高线图的填充颜色	4-3
突出显示特定等高线层级	4-5
合并等高线图和箭头图	4-7
带有主网格线和次网格线的等高线图	4-9

专用图

5

基于表格数据创建热图	5-2
使用字符串数组创建文字云	5-11
使用平行坐标图探索表数据	5-14

6

使用纬度和经度数据创建地图	6-2
地理坐标区和地理图中的平移和缩放行为	6-6
地理气泡图概述	6-7
地理气泡图图例	6-9
在地理密度图上查看飓风轨迹数据	6-10
查看蜂窝塔位置的密度	6-14
自定义地理坐标区的布局	6-20
部署地理坐标区和地理图	6-22
使用地理气泡图属性	6-23
控制气泡大小	6-23
控制气泡颜色	6-25
使用地理坐标区指定地图范围	6-27
在指定的范围内居中显示几个地理气泡图	6-27
访问用于地理坐标区和地理图的底图	6-31
在地理图上显示 'darkwater'	6-31
在地理气泡图上显示 'darkwater'	6-33
下载底图	6-35
底图缓存行为	6-35
基于表格数据创建地理气泡图	6-37

动画**7**

动画技术	7-2
更新屏幕	7-2
优化性能	7-2
沿线条跟踪标记	7-3
沿着线条移动一组对象	7-5
对图形对象进行动画处理	7-8
线条动画	7-10

录制动画用于播放	7-12
录制和播放影片	7-12
为影片捕获整个图窗	7-12
为曲面添加动画效果	7-15

标题和标签

8

为图添加标题和轴标签	8-2
将图例添加到图	8-8
向图中添加文本	8-14
向图中添加注释	8-21
图文本中的希腊字母和特殊字符	8-25
包含希腊字母	8-25
包含上标和注释	8-25
TeX 标记选项	8-27
用 LaTeX 创建文本	8-29
使用 LaTeX 创建绘图标题、刻度标签和图例	8-30
缩小图形标题	8-34

坐标区外观

9

指定坐标轴范围	9-2
指定坐标轴刻度值和标签	9-9
添加网格线和编辑布局	9-16
合并多个绘图	9-24
创建包含双 y 轴的图。	9-32
修改包含两个 y 轴的图形的属性	9-40
更改坐标区属性	9-40
更改标尺属性	9-42
使用默认色序指定颜色	9-44
使用多个刻度和坐标轴范围显示数据	9-46
用两个 y 轴显示数据	9-46
用多个 x 轴和 y 轴显示数据	9-47
在不连续的 x 轴上绘制数据	9-49

用不同的颜色栏显示两组数据	9-53
控制坐标轴长度比率和数据单位长度	9-56
图框纵横比	9-56
数据纵横比	9-58
还原为默认比率	9-61
控制坐标区布局	9-63
与坐标区位置相关的属性	9-63
位置和边距的边界	9-63
控制自动调整大小行为	9-64
伸展填充行为	9-65
控制坐标区纵横比	9-67
坐标区纵横比属性	9-67
默认纵横比选择	9-68
通过调整图窗大小保持坐标区比例	9-69
纵横比属性	9-71
显示真实对象	9-75
控制绘图函数如何选择颜色和线型	9-78
自动分配的工作原理	9-78
更改颜色方案和线型	9-79
更改 ColorOrder 和 LineStyleOrder 数组的索引	9-81
在绘图和图表中裁剪	9-83
使用图形平滑处理	9-88

标记图形颜色

10

创建颜色栏	10-2
使用颜色图更改颜色方案	10-10
曲面绘图数据与颜色图的关系	10-16
曲面与颜色图之间的关系	10-16
更改颜色的方向或模式	10-16
图像数据与颜色图的关系	10-21
补片数据与颜色图的关系	10-26
颜色图与 x、y、z 坐标数组的关系	10-26
颜色图与面-顶点数据的关系	10-29
控制颜色图范围	10-34
颜色图和真彩色之间的差异	10-38
工作流差异	10-38
视觉表示的差异	10-39

11

光照概述	11-2
光照命令	11-2
光源对象	11-2
影响光照的属性	11-2
光照控制示例	11-3
图形对象的反射特性	11-7
镜面反射和漫反射	11-7
环境光	11-7
镜面反射指数	11-8
镜面颜色反射	11-8
背面光照	11-9
在数据空间放置光源	11-10

12

为图形对象添加透明度	12-2
什么是透明度?	12-2
支持透明度的图形对象	12-2
创建具有透明度的区域图	12-3
创建具有透明度的条形图	12-4
创建具有透明度的散点图	12-5
使用 Alpha 数据更改透明度	12-6
更改曲面图透明度	12-7
更改补片对象透明度	12-7
更改图像、填充或曲面的透明度	12-9
修改 alphamap	12-16
默认的 alpha 映射	12-16
示例 - 修改 alphamap	12-18

13

交互式探查绘图数据	13-2
缩放、平移和旋转数据	13-2
使用数据提示显示数据值	13-2
使用数据刷亮功能选择和修改数据值	13-3
使用属性检查器自定义图	13-4
创建自定义数据提示	13-6
更改标签和添加行	13-6
在数据提示中显示表值	13-7

更改数据后自动刷新图	13-9
使用数据链接功能更新图	13-9
使用数据源属性更新图	13-10
对图的交互进行控制	13-12
显示或隐藏坐标区工具栏	13-12
自定义坐标区工具栏	13-12
启用或禁用内置交互	13-13
自定义内置交互	13-14

相机视图

14

视图概述	14-2
观察三维图形和场景	14-2
定位视点	14-2
设置纵横比	14-2
默认视图	14-2
利用方位角和仰角设置视点	14-3
方位角和仰角	14-3
相机图形术语	14-7
使用相机工具栏控制视图	14-8
相机工具栏	14-8
相机移动控件	14-10
环移相机	14-10
环移场景灯光	14-11
平转/纵转相机	14-11
水平/垂直移动相机	14-12
向前和向后移动相机	14-13
缩放相机	14-14
相机滚转	14-15
推移相机	14-17
方法简介	14-17
实现	14-17
移动相机穿过场景	14-18
方法简介	14-18
绘制体数据	14-18
设置视图	14-19
指定光源	14-19
选择光照方法	14-19
将相机路径定义为流线	14-19
实现漫游	14-20
低级相机属性	14-21
可以设置的相机属性	14-21
默认视点选择	14-21
移入和移出场景	14-22

使场景变大或变小	14-23
围绕场景旋转	14-23
旋转但不调整大小	14-24
围绕观察轴旋转	14-24
了解视图投影	14-26
两种投影类型	14-26
投影类型和相机位置	14-27

显示位图图像

15

处理 MATLAB 图形中的图像	15-2
什么是图像数据?	15-2
支持的图像格式	15-3
图像类型	15-4
索引图像	15-4
灰度 (强度) 图像	15-5
RGB (真彩色) 图像	15-5
8 位和 16 位图像	15-7
索引图像	15-7
强度图像	15-7
RGB 图像	15-8
uint8 和 uint16 的数学运算支持	15-8
其他 8 位和 16 位数组支持	15-8
将 8 位 RGB 图像转换为灰度图像	15-9
图像类型和数值类摘要	15-11
读取、写入和查询图像文件	15-12
使用图像格式	15-12
读取图形图像	15-12
写入图形图像	15-13
划分图形图像的分集 (裁剪)	15-13
获取有关图形文件的信息	15-14
显示图形图像	15-15
图像类型和显示方法	15-15
控制纵横比和显示尺寸	15-16
图像对象及其属性	15-18
图像 CData	15-18
图像 CDataMapping	15-18
XData 和 YData	15-19
在图像数据上添加文本	15-20
快速更新图像的其他技术	15-21
打印图像	15-23
转换图像图形或数据类型	15-24

打印和保存**16**

从“文件”菜单打印图窗	16-2
直接打印输出	16-2
保留背景色和刻度值	16-2
图窗大小和布局	16-2
线宽和字体大小	16-3
通过“编辑”菜单将图窗复制到剪贴板	16-5
将图窗复制到剪贴板	16-5
指定格式、背景色和大小选项	16-6
保存前自定义图窗	16-8
设置图窗大小	16-8
设置图窗背景色	16-9
设置图窗字体大小和线宽	16-10
将图窗保存到文件	16-11
保存图窗设置以供将来使用	16-12
将设置应用于另一个图窗	16-12
将图窗还原为原始设置	16-12
以编程方式自定义图窗	16-13
将绘图保存为图像或向量图形文件	16-14
以交互方式保存绘图	16-14
以编程方式保存绘图	16-15
在其他应用程序中打开保存的绘图	16-17
使用特定大小、分辨率或背景色保存图窗	16-18
指定分辨率	16-18
指定大小	16-19
指定背景色	16-20
保留坐标轴范围和刻度值	16-20
保存图窗以供以后在 MATLAB 中重新打开	16-22
将图窗保存为 FIG 文件	16-22
生成代码以重新创建图窗	16-23
保存和复制绘图时保留最少的空白	16-24
保存或复制单一绘图	16-24
保存或复制图窗中的多个绘图	16-25

图形属性**17**

修改图形对象	17-2
-------------------------	-------------

图形对象层次结构	17-9
MATLAB 图形对象	17-9
图形由具体对象组成	17-9
图形对象的组织	17-9
访问属性值	17-12
对象属性和圆点表示法	17-12
图形对象变量是句柄	17-13
列出对象属性	17-14
使用 set 和 get 修改属性	17-15
多对象/属性操作	17-15
图形对象控制的功能	17-17
图形对象的用途	17-17
图窗	17-17
坐标区	17-18
表示数据的对象	17-18
组对象	17-19
注释对象	17-19
默认属性值	17-21
属性预定义值	17-21
指定默认值	17-21
在层次结构中什么位置定义默认值	17-22
列出默认值	17-22
将属性设置为当前默认值	17-22
删除默认值	17-22
将属性设置为出厂定义值	17-22
列出出厂定义属性值	17-23
保留字	17-23
自动计算的属性的默认值	17-24
什么是自动计算的属性	17-24
自动计算的属性的默认值	17-24
MATLAB 如何找到默认值	17-26
出厂定义属性值	17-27
多级默认值	17-28

对象标识

18

特殊对象标识符	18-2
获取特殊对象的句柄	18-2
当前图窗、坐标区和对象	18-2
回调对象和回调图窗	18-3
查找对象	18-4
查找具有特定属性值的对象	18-4
通过字符串属性查找文本	18-4

使用含有 findobj 的正则表达式	18-5
限制搜索范围	18-6
复制对象	18-8
使用 copyobj 复制对象	18-8
将单个对象复制到多个目的地。	18-8
复制多个对象	18-8
删除图形对象	18-10
如何删除图形对象	18-10
已删除对象的句柄	18-11

处理图形对象

19

图形对象句柄	19-2
您可以使用句柄做什么	19-2
句柄不具有的功能	19-2
预分配图形对象数组	19-4
检验有效句柄	19-5
逻辑表达式中的句柄	19-6
如果句柄有效	19-6
如果结果为空	19-6
如果句柄相等	19-7
图形数组	19-8

图形对象回调

20

回调 - 对用户操作的程序化响应	20-2
什么是回调?	20-2
窗口回调	20-2
回调定义	20-3
指定回调的方法	20-3
回调函数语法	20-3
相关信息	20-4
将回调定义为默认值	20-4
按钮按下回调函数	20-5
何时使用按钮按下回调	20-5
如何定义按钮按下回调	20-5
定义上下文菜单	20-6
何时使用上下文菜单	20-6

如何定义上下文菜单	20-6
定义对象创建回调	20-7
相关信息	20-7
定义对象删除回调	20-8
捕获鼠标点击	20-9
控制对鼠标点击进行响应的属性	20-9
组合使用 PickablePart/HitTest 值	20-9
沿层次结构向上传递鼠标点击	20-10
将鼠标点击传递给组的父级	20-12
目标和设计	20-12
对象层次结构和关键属性	20-12
MATLAB 代码	20-12
将鼠标点击传递给被遮盖对象	20-14

组对象

21

对象组	21-2
创建对象组	21-3
父级设定	21-3
组的子对象的可见性和选中属性	21-4
hgtransform 支持的变换	21-5
变换对象	21-5
旋转	21-5
转换	21-5
缩放	21-6
默认变换	21-6
不允许的变换：透视	21-6
不允许的变换：剪切	21-6
绝对变换与相对变换	21-7
将变换合并到一个矩阵	21-7
撤消变换操作	21-7
绕任意轴旋转	21-9
旋转前转换到原点	21-9
旋转曲面	21-9
嵌套变换，执行复杂移动	21-12

22

控制图形显示	22-2
您可以控制什么	22-2
以特定图窗和坐标区为目标	22-2
为绘图准备图窗和坐标区	22-4
MATLAB 绘图函数的行为	22-4
NextPlot 属性如何控制行为	22-4
用户编写绘图函数的控制行为	22-5
使用 newplot 控制绘图	22-7
对保留状态进行响应	22-9
防止对图窗和坐标区访问	22-11
为什么防止访问	22-11
如何防止访问	22-11

开发图对象的类**23**

图开发概述	23-2
图类的结构	23-2
隐式构造函数方法	23-2
公共和私有属性块	23-3
setup 方法	23-3
update 方法	23-4
示例：置信边界图	23-5
支持常用图形功能	23-7
为图类编写构造函数	23-9
示例：具有自定义构造函数的置信边界图	23-9
创建包含极坐标区、地理坐标区或多个坐标区的图	23-13
创建单个极坐标区或地理坐标区对象	23-13
创建多个坐标区对象的分块	23-13
示例：包含地理坐标区和笛卡尔坐标区的图	23-14
管理图类的属性	23-17
初始化属性值	23-17
验证属性值	23-17
自定义属性显示	23-18
优化 update 方法	23-19
示例：具有自定义属性显示的优化的等值面图	23-20
启用便利函数以设置坐标区属性	23-25
支持不同类型的属性	23-25
为非计算的属性启用函数	23-25
为计算的属性启用函数	23-25

支持 title、xlim 和 ylim 函数的图类	23-26
保存和加载图类的实例	23-31
保存和加载坐标区更改的编码模式	23-31
定义用于存储图状态的受保护属性	23-31
定义检索图状态的 get 方法	23-31
定义更新坐标区的受保护方法	23-32
示例：用于存储轴范围和视图的三维绘图	23-33
具有自定义属性显示的图类	23-37
具有可变数量的线条的图类	23-40
用于显示不定数量的线条的优化图类	23-43
用于显示可变大小绘图分块的图类	23-47
包含两个交互式绘图的图类	23-50

优化图形程序的性能

24

找到代码瓶颈	24-2
什么影响代码执行速度	24-4
潜在瓶颈	24-4
如何提升性能	24-4
明智的对象创建	24-5
对象开销	24-5
不要创建不需要的对象	24-5
使用 NaN 模拟多个线条	24-5
修改数据而不是创建新对象	24-5
避免重复搜索对象	24-7
限制搜索范围	24-7
屏幕更新	24-8
MATLAB 图形系统	24-8
管理更新	24-8
优化代码以获取和设置图形属性	24-10
自动计算属性	24-10
效率不高的设置和获取周期	24-10
更改文本 Extent 以旋转标签	24-11
避免更新静态数据	24-12
分割数据以减少更新次数	24-12
高效变换对象	24-13

使用低级函数提升速度	24-14
图形的系统要求	24-15
最低系统要求	24-15
推荐系统要求	24-15
升级图形驱动程序	24-15
具有特定要求的图形功能	24-15
解决低级图形问题	24-17
升级图形硬件驱动程序	24-17
选择适合您的系统的渲染器实现	24-17
解决内存不足的问题	24-18
联系技术支持	24-18

线图

- “MATLAB 绘图类型” (第 1-2 页)
- “创建常见的二维图” (第 1-4 页)
- “创建二维线图” (第 1-13 页)
- “创建带标记的线图” (第 1-18 页)
- “使用两个 y 轴合并线图和条形图” (第 1-26 页)
- “合并线图和针状图” (第 1-29 页)
- “叠加阶梯图和线图” (第 1-32 页)
- “带有置信边界的线图” (第 1-34 页)
- “绘制虚数和复数数据图” (第 1-35 页)

MATLAB 绘图类型

MATLAB 提供了各种可用来绘制数据图的函数。下表对常见的图形函数进行了分类和说明。

“线图”	散点图和气泡图	“数据分布图”	“离散数据图”	“地理图”	“极坐标图”	“等高线图”	“向量场”	“曲面图和网格图”	“三维可视化”	“动画”	“图像”
plot	scatter r	histogram	bar	geoplot	polar plot	contour	quiver	surf	streamline	animate	image
plot3	scatter3	histogram2	barh	geosatellite	polar histogram	contourf	quiver3	surf	streamslice	comet	imagesc
stairs	bubblechar	pie	bar3	geobubble	polarscatter	contour3	feather	surfl	streamparticles	comet3	
errorbar	bubblechart3	pie3	bar3h		polarbubblechart	contourslice		ribbon	streamribbon		
area	swarmchart	scatterhistogram	pareto		compass	contourf		pcolor	streamtube		
stackedplot	swarmchart3	swarmchart	stem		ezpolar			fsurf	coneplot		

“线图”	散点图和气泡图	“数据分布图”	“离散数据图”	“地理图”	“极坐标图”	“等高线图”	“向量场”	“曲面图和网格图”	“三维可视化”	“动画”	“图像”
loglog 	spy 	swarm 	stem3 					fimplicit 	slice 		
semilogx 		wordcloud 	stairs 					mesh 			
semilogy 		bubblecloud 						meshc 			
fplot 		heatmap 						meshz 			
fplot3 		parallelplot 						waterfall 			
fimplicit 		plotmatrix 						fmesh 			

另请参阅

相关示例

- “创建二维线图” (第 1-13 页)
- MATLAB 图库

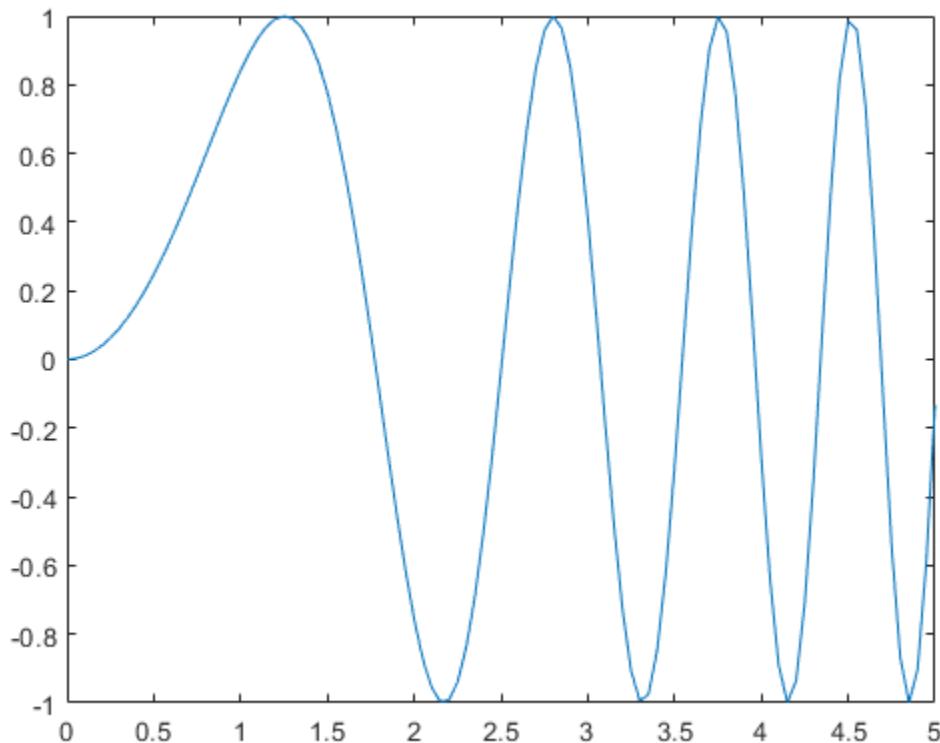
创建常见的二维图

以下示例演示如何在 MATLAB® 中创建各种二维图。

线图

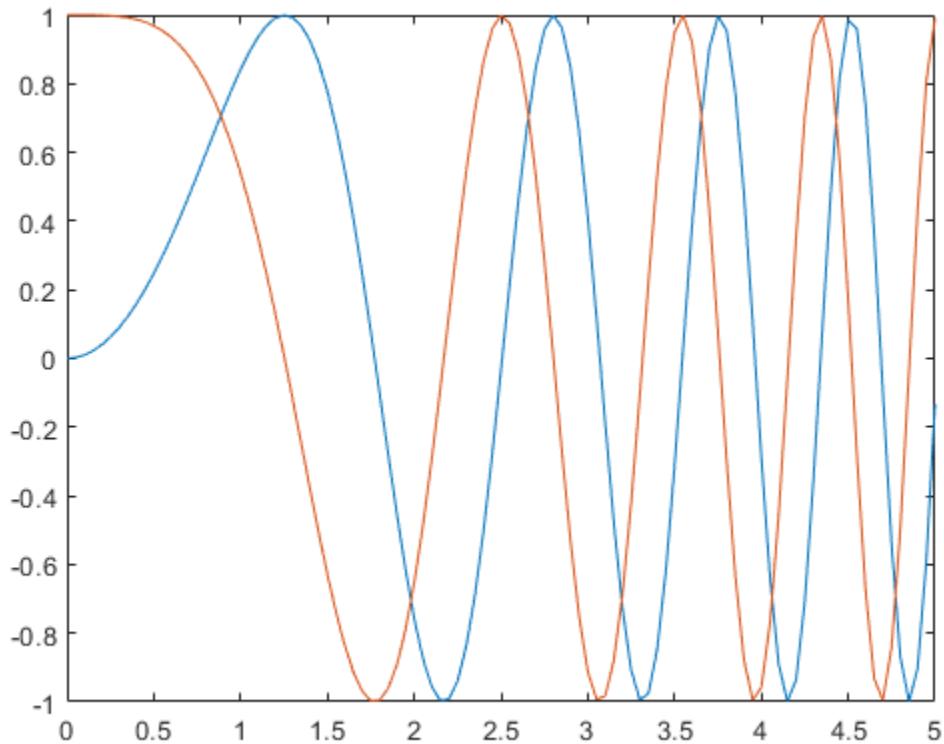
`plot` 函数用来创建 x 和 y 值的简单线图。

```
x = 0:0.05:5;
y = sin(x.^2);
figure
plot(x,y)
```



线图可显示多组 x 和 y 数据。

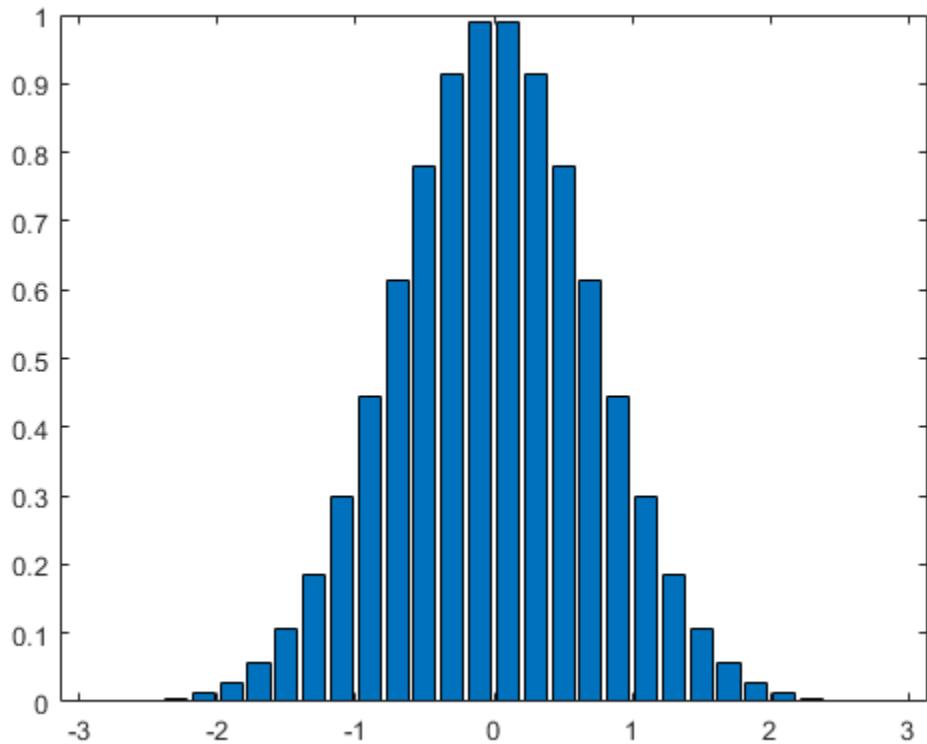
```
y1 = sin(x.^2);
y2 = cos(x.^2);
plot(x,y1,x,y2)
```



条形图

bar 函数用来创建垂直条形图。 **barh** 函数用来创建水平条形图。

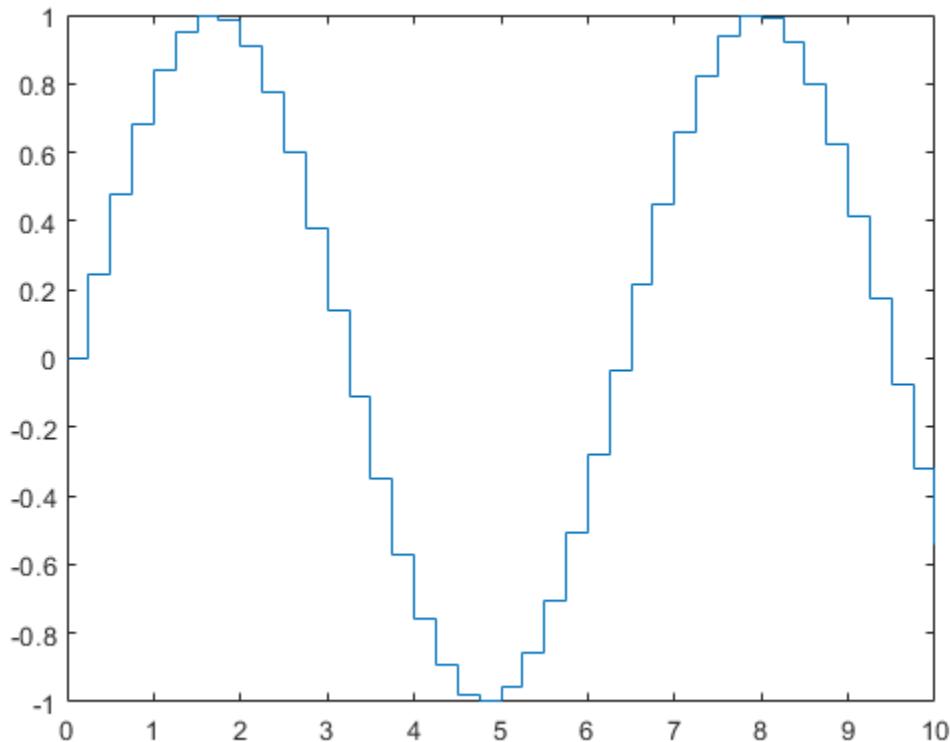
```
x = -2.9:0.2:2.9;  
y = exp(-x.*x);  
bar(x,y)
```



阶梯图

`stairs` 函数用来创建阶梯图。它可以创建仅含 Y 值的阶梯图，或同时包含 x 和 y 值的阶梯图。

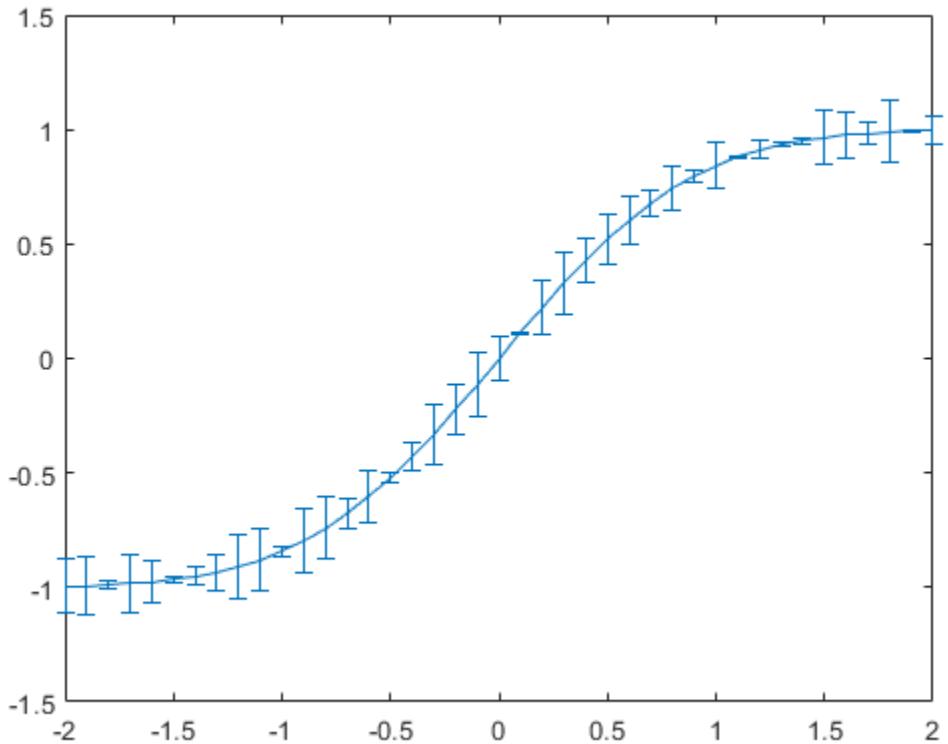
```
x = 0:0.25:10;  
y = sin(x);  
stairs(x,y)
```



误差条形图

`errorbar` 函数用来绘制 x 和 y 值的线图并在每个观察点上叠加垂直误差条。若要指定误差条的大小，需要向 `errorbar` 函数传递一个额外的输入参数。

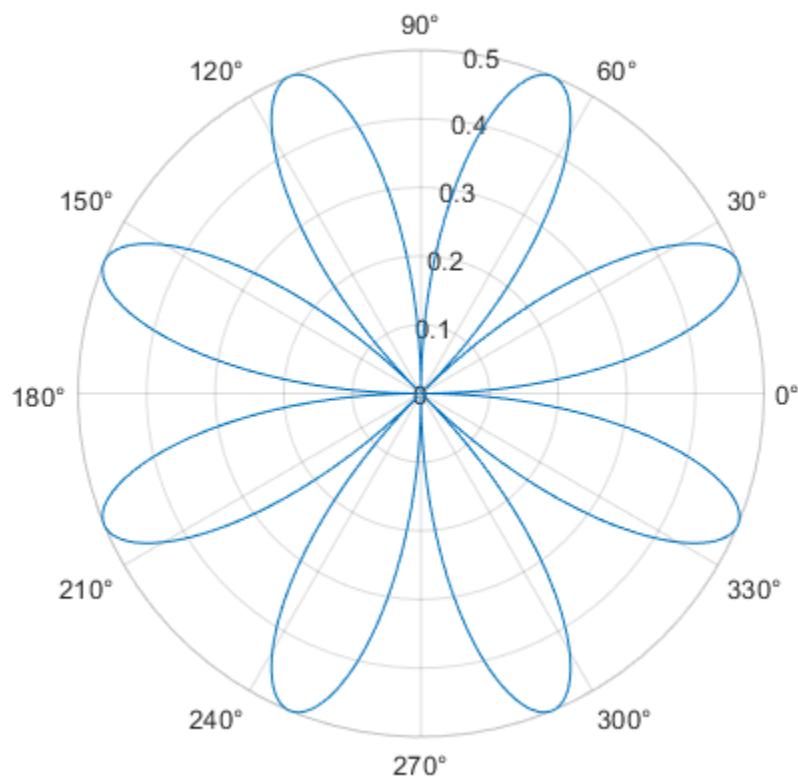
```
x = -2:0.1:2;
y = erf(x);
eb = rand(size(x))/7;
errorbar(x,y,eb)
```



极坐标图

polarplot 函数用来绘制 theta 中的角度值（以弧度为单位）对 rho 中的半径值的极坐标图。

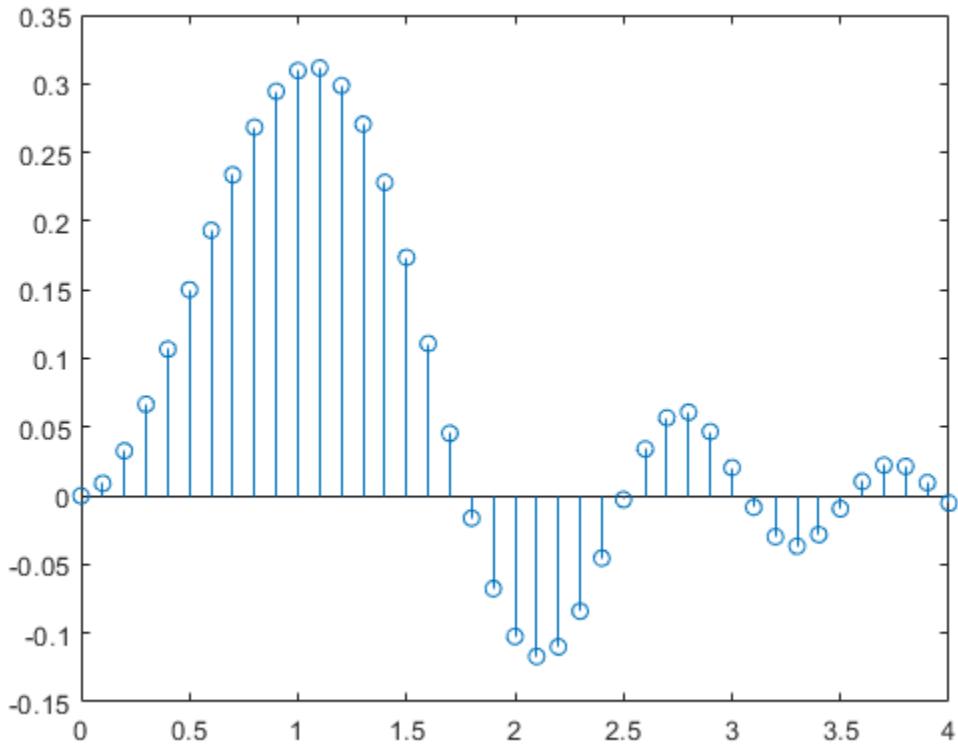
```
theta = 0:0.01:2*pi;
rho = abs(sin(2*theta).*cos(2*theta));
polarplot(theta,rho)
```



针状图

`stem` 函数为每个通过竖线连接到一条公共基线的 x 和 y 值绘制一个标记。

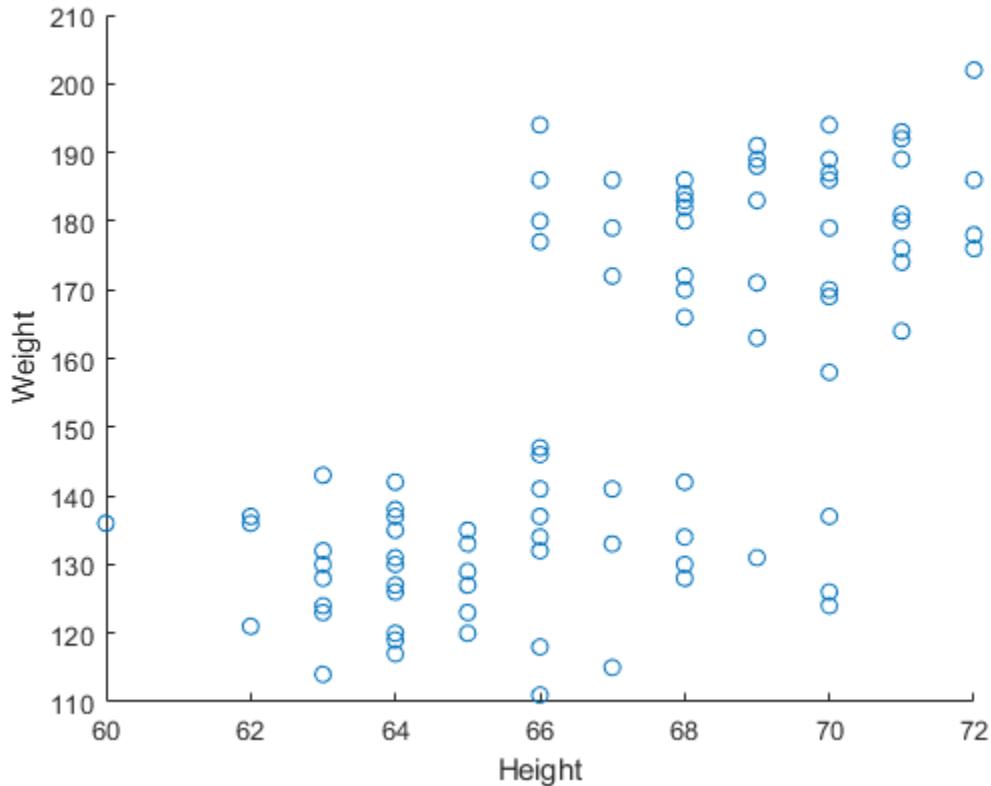
```
x = 0:0.1:4;
y = sin(x.^2).*exp(-x);
stem(x,y)
```



散点图

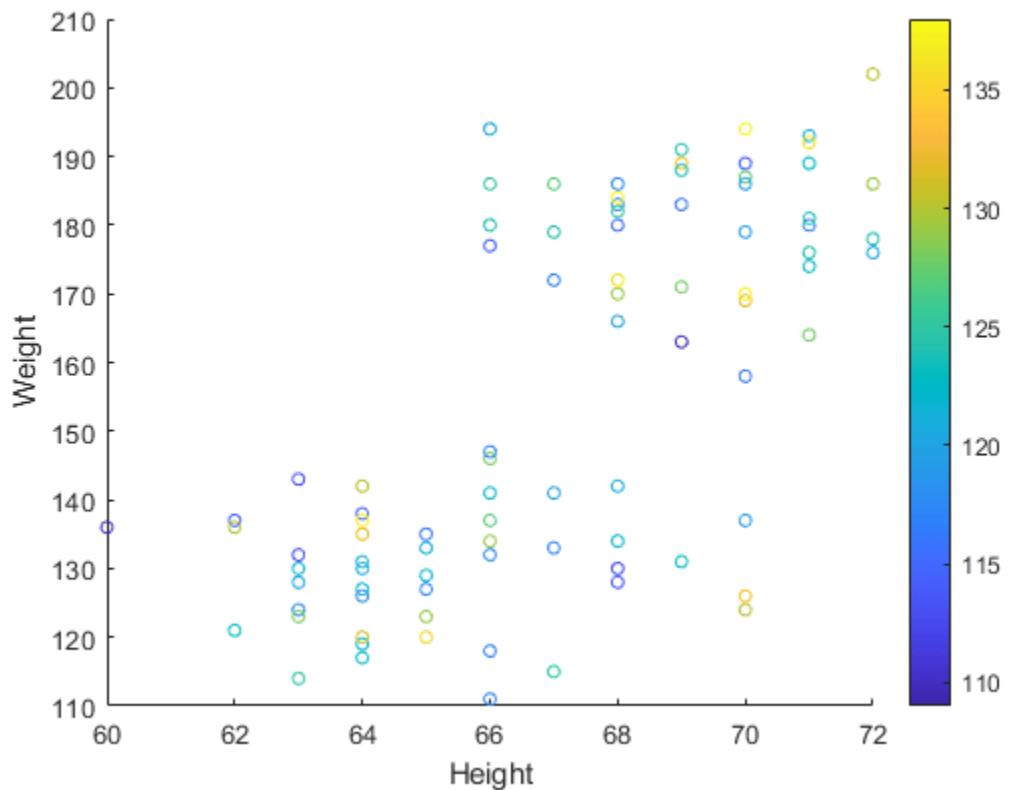
`scatter` 函数用来绘制 x 和 y 值的散点图。

```
load patients Height Weight Systolic  
scatter(Height,Weight)  
xlabel('Height')  
ylabel('Weight')
```



使用 `scatter` 函数的可选参数指定标记的大小和颜色。使用 `colorbar` 函数显示当前坐标区上的色阶。

```
scatter(Height,Weight,20,Systolic)
xlabel('Height')
ylabel('Weight')
colorbar
```



另请参阅

相关示例

- “创建二维线图”（第 1-13 页）

创建二维线图

创建一个简单的线图并标记坐标区。通过更改线条颜色、线型和添加标记来自定义线图的外观。

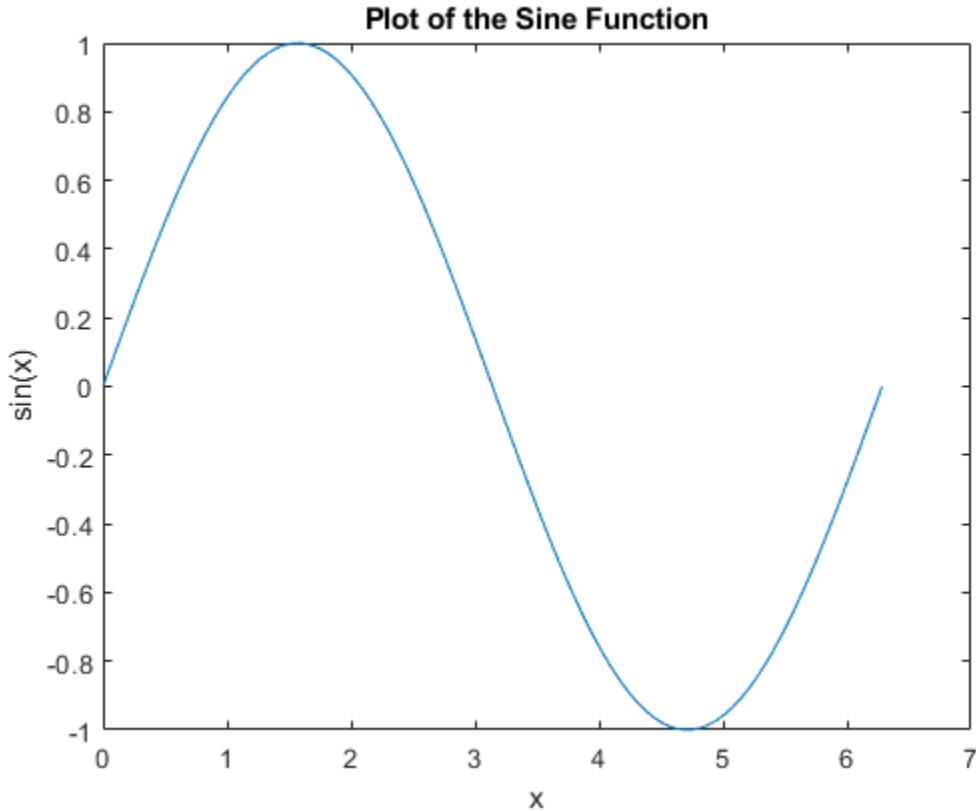
创建线图

使用 `plot` 函数创建二维线图。例如，绘制从 0 到 2π 之间的正弦函数值。

```
x = linspace(0,2*pi,100);
y = sin(x);
plot(x,y)
```

标记坐标区并添加标题。

```
xlabel('x')
ylabel('sin(x)')
title('Plot of the Sine Function')
```



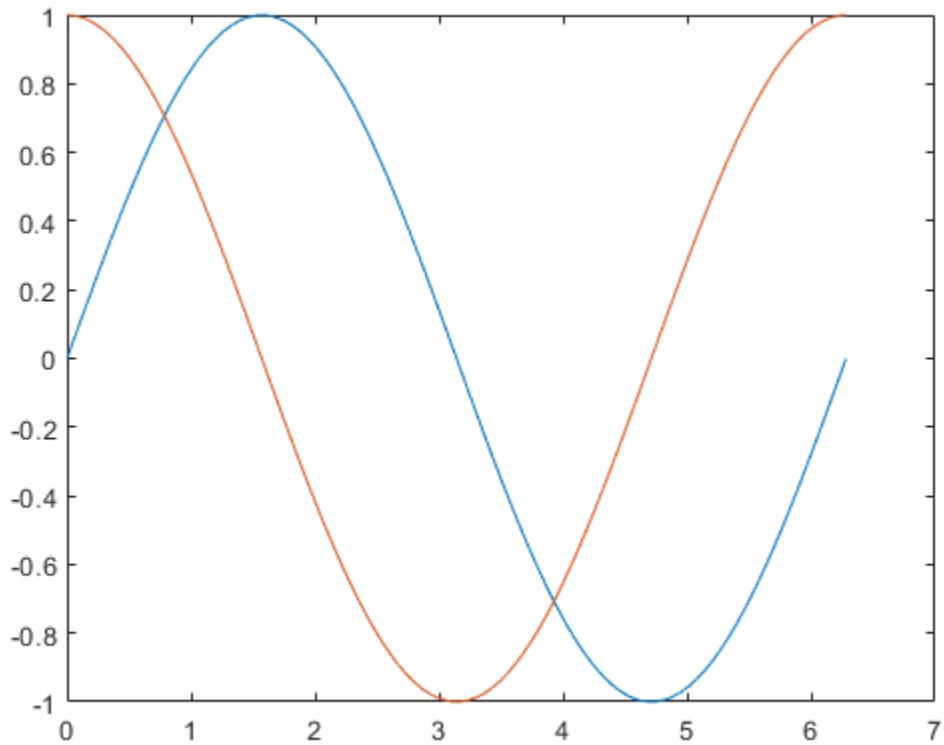
绘制多个线条

默认情况下，MATLAB 会在执行每个绘图命令之前清空图窗。使用 `figure` 命令打开一个新的图窗窗口。可以使用 `hold on` 命令绘制多个线条。在使用 `hold off` 或关闭窗口之前，当前图窗窗口中会显示所有绘图。

```
figure
x = linspace(0,2*pi,100);
y = sin(x);
```

```
plot(x,y)

hold on
y2 = cos(x);
plot(x,y2)
hold off
```



更改线条外观

通过在调用 **plot** 函数时包含可选的线条设定，可以更改线条颜色、线型或添加标记。例如：

- ':' 绘制点线。
- 'g:' 绘制绿色点线。
- 'g:.*' 绘制带有星号标记的绿色点线。
- '.*' 绘制不带线条的星号标记。

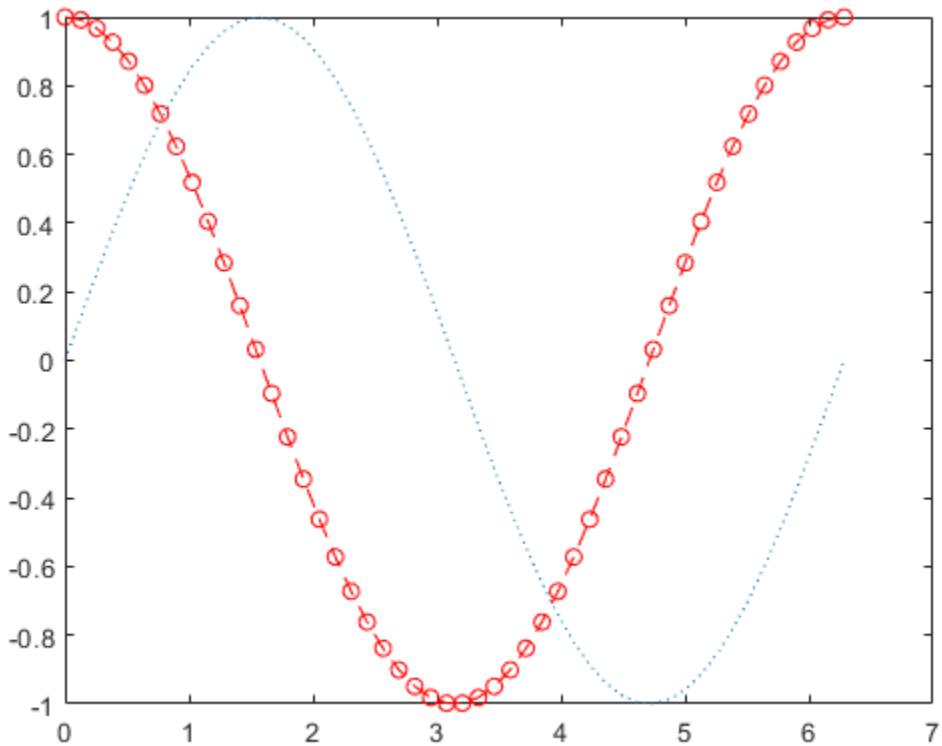
符号可以按任意顺序显示。不需要同时指定所有三个特征（线条颜色、线型和标记）。有关不同样式选项的详细信息，请参阅 **plot** 函数页。

例如，绘制一条点线。添加第二个图，该图使用带有圆形标记的红色虚线。

```
x = linspace(0,2*pi,50);
y = sin(x);
plot(x,y,':r')
```

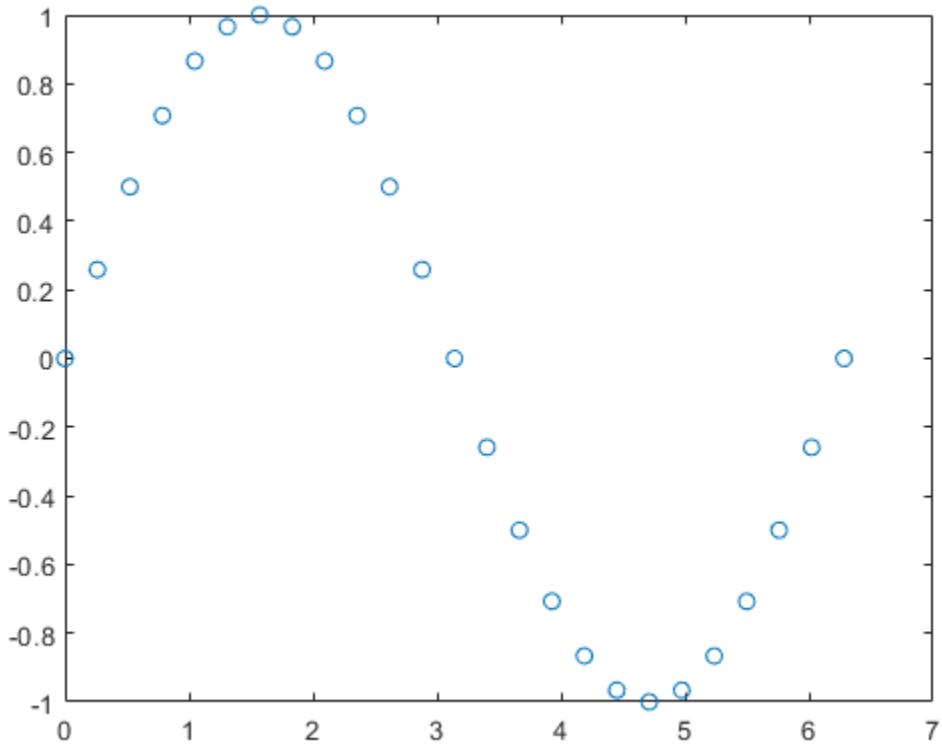
```
hold on
```

```
y2 = cos(x);
plot(x,y2,'-ro')
hold off
```



通过忽略线条设定中的线型选项，仅绘制数据点。

```
x = linspace(0,2*pi,25);
y = sin(x);
plot(x,y,'o')
```



更改线条对象的属性

通过更改用来创建绘图的 Line 对象的属性，还可以自定义绘图的外观。

创建一个线图。将创建的 Line 对象赋给变量 ln。显示画面上显示常用属性，例如 Color、LineStyle 和 LineWidth。

```
x = linspace(0,2*pi,25);
y = sin(x);
ln = plot(x,y)
```

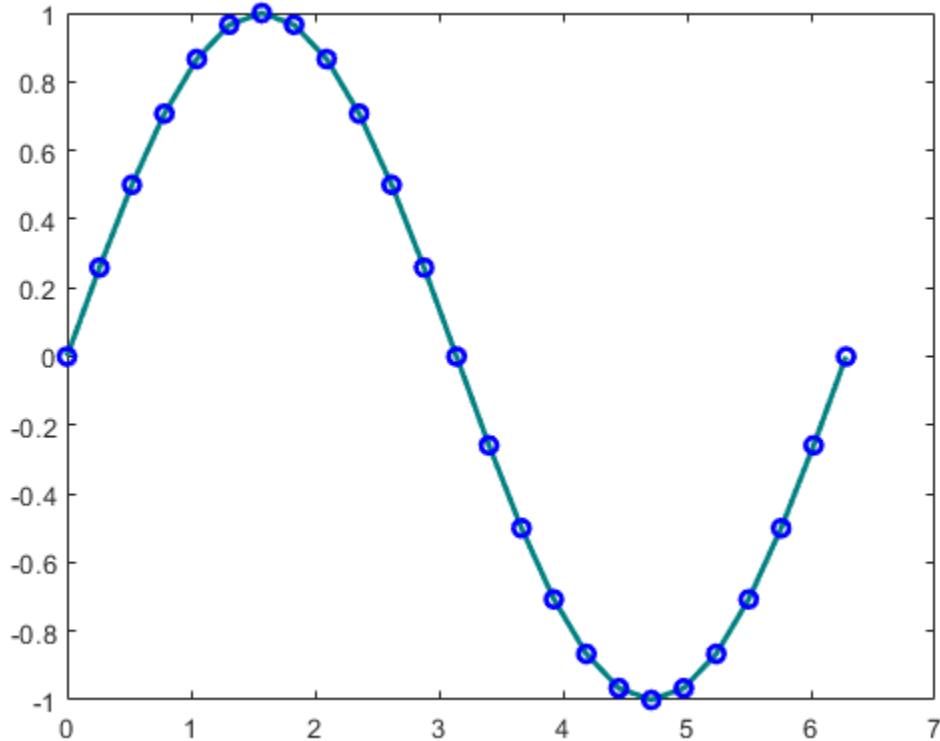
```
ln =
Line with properties:

    Color: [0 0.4470 0.7410]
    LineStyle: '-'
    LineWidth: 0.5000
    Marker: 'none'
    MarkerSize: 6
    MarkerFaceColor: 'none'
    XData: [0 0.2618 0.5236 0.7854 1.0472 1.3090 1.5708 1.8326 ... ]
    YData: [0 0.2588 0.5000 0.7071 0.8660 0.9659 1 0.9659 ... ]
    ZData: [1x0 double]
```

Show all properties

要访问各个属性，请使用圆点表示法。例如，将线宽更改为 2 磅并将线条颜色设置为 RGB 三元组颜色值，在本例中为 [0 0.5 0.5]。添加蓝色圆形标记。

```
ln.LineWidth = 2;  
ln.Color = [0 0.5 0.5];  
ln.Marker = 'o';  
ln.MarkerEdgeColor = 'b';
```



另请参阅

[plot](#) | [loglog](#) | [scatter](#) | [Line Properties](#)

相关示例

- “为图添加标题和轴标签”（第 8-2 页）
- “指定坐标轴范围”（第 9-2 页）
- “指定坐标轴刻度值和标签”（第 9-9 页）
- 创建绘图
- MATLAB 图库

创建带标记的线图

在线图中添加标记是区分多个线条或突出显示特定数据点的有用方法。使用下面的一种方式添加标记：

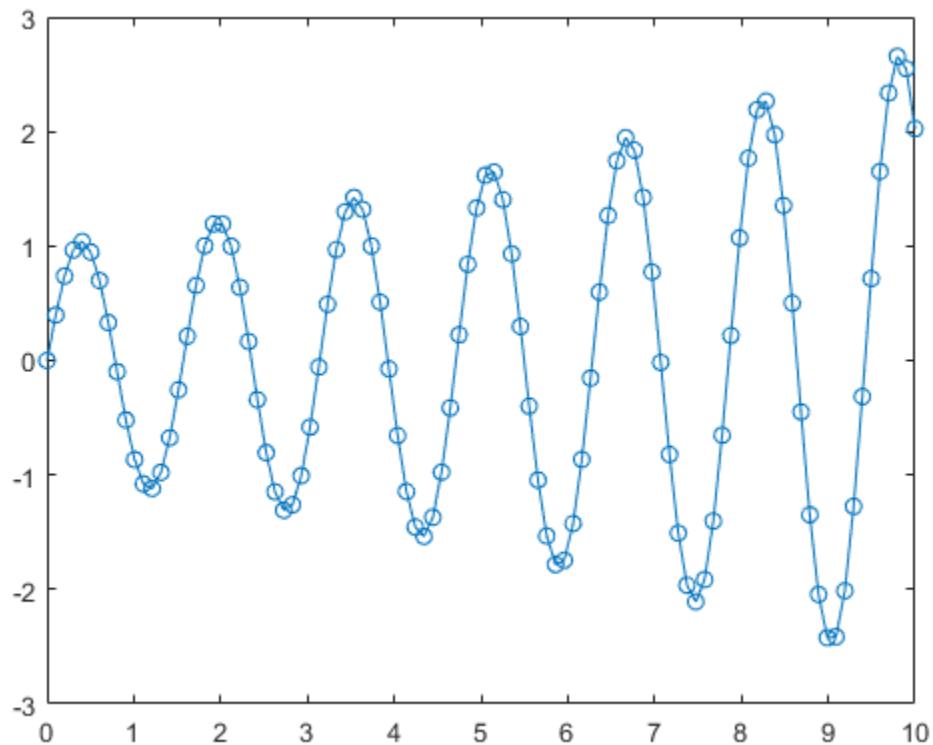
- 在线条设定输入参数（例如 `plot(x,y,'-s')`）中包含标记符号。
- 将 `Marker` 属性指定为一个名称-值对组，例如 `plot(x,y,'Marker','s')`。

有关标记选项列表，请参阅“支持的标记符号”（第 1-24 页）。

在线图中添加标记

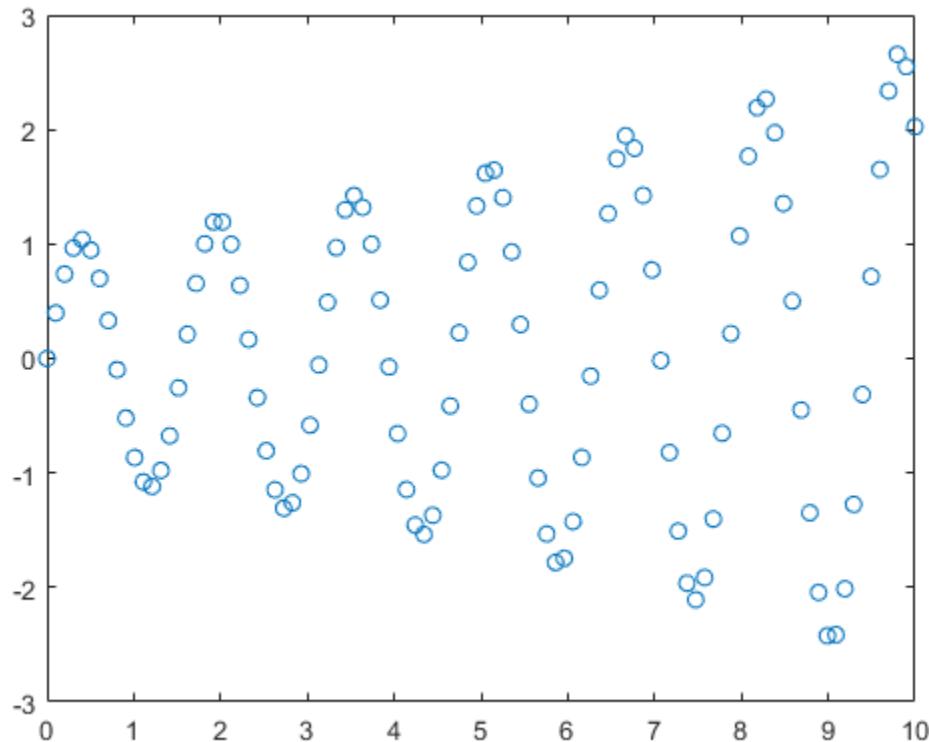
创建一个线图。通过在调用 `plot` 函数时包含线条设定输入参数，在每个数据点处显示一个标记。例如，使用 '`-o`' 可得到一条带圆形标记的实线。

```
x = linspace(0,10,100);
y = exp(x/10).*sin(4*x);
plot(x,y,'-o')
```



如果指定了标记符号但未指定线型，则 `plot` 仅显示无线条连接的标记。

```
plot(x,y,'o')
```



也可以通过将 **Marker** 属性设置为名称-值对组，在线条中添加标记。例如，`plot(x,y,'Marker','o')` 将绘制一个带圆形标记的线条。

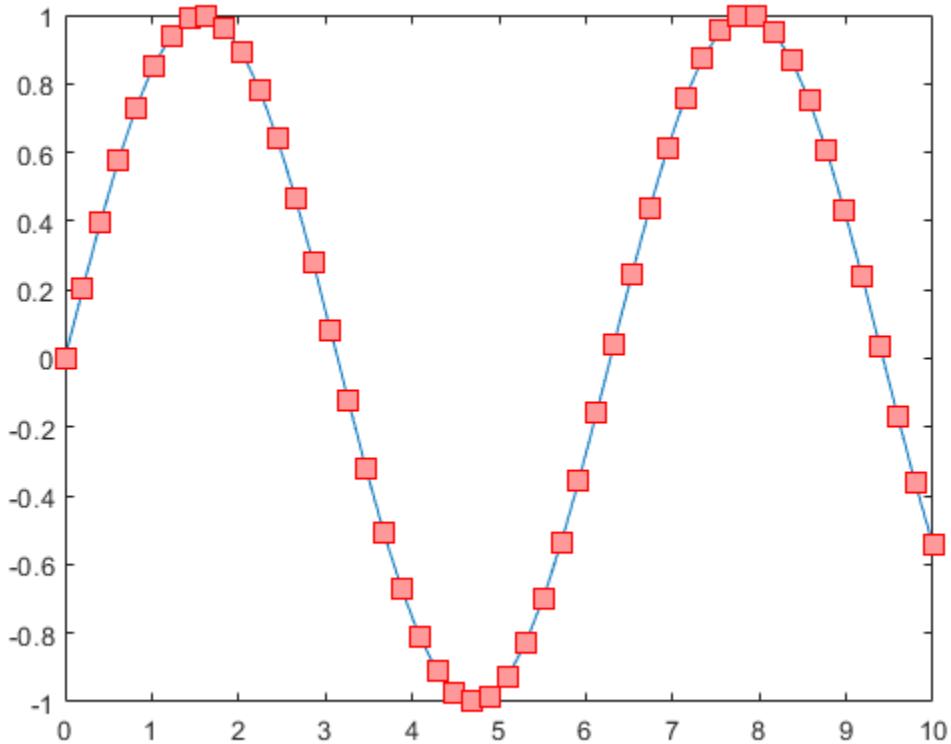
指定标记大小和颜色

创建带标记的线图。通过结合使用 **plot** 函数和名称-值对组参数来设置这些属性，即可自定义标记：

- **MarkerSize** - 标记大小，指定为正值。
- **MarkerEdgeColor** - 标记轮廓颜色，指定为颜色名称或 RGB 三元组。
- **MarkerFaceColor** - 标记内部颜色，指定为颜色名称或 RGB 三元组。

使用颜色名称的字符向量（例如 `'red'`）或 RGB 三元组（例如 `[0.4 0.6 0.7]`）指定颜色。RGB 三元组是包含三个元素的行向量，其元素分别指定颜色中红、绿、蓝分量的强度。强度必须处于范围 [0,1] 中。

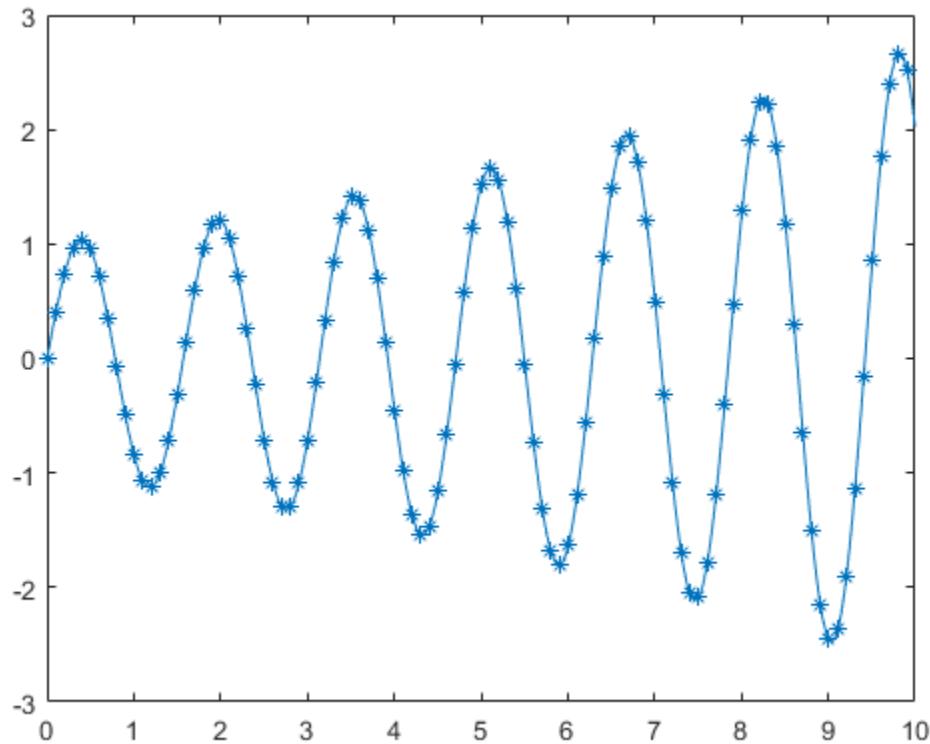
```
x = linspace(0,10,50);
y = sin(x);
plot(x,y,'s','MarkerSize',10,...
      'MarkerEdgeColor','red',...
      'MarkerFaceColor',[1 .6 .6])
```



控制沿线条的标记放置

创建包含 1000 个数据点的线图，添加星号标记，并使用 **MarkerIndices** 属性控制标记位置。将此属性设置为要显示标记的数据点的索引。从第一个数据点开始，每隔十个数据点显示一个标记。

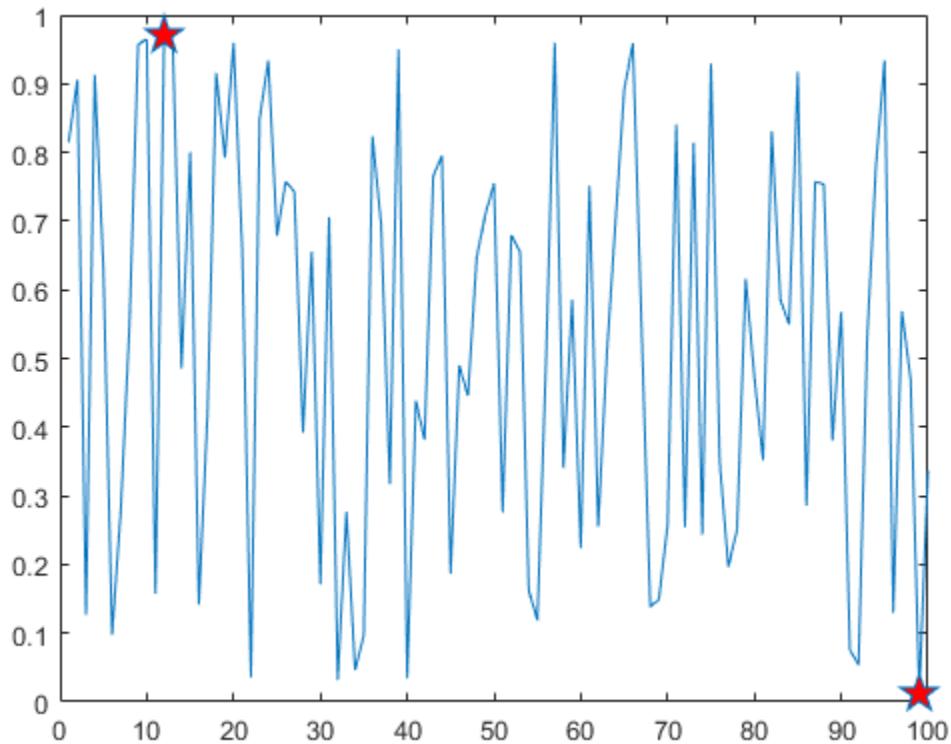
```
x = linspace(0,10,1000);
y = exp(x/10).*sin(4*x);
plot(x,y,'-*','MarkerIndices',1:10:length(y))
```



在最大数据点和最小数据点处显示标记

创建一个随机数据向量，并查找最小值和最大值的索引。然后创建数据的线图。通过将 `MarkerIndices` 属性设置为索引值向量，在最小数据值和最大数据值处显示红色标记。

```
x = 1:100;
y = rand(100,1);
idxmin = find(y == min(y));
idxmax = find(y == max(y));
plot(x,y,'-p','MarkerIndices',[idxmin idxmax],...
'MarkerFaceColor','red',...
'MarkerSize',15)
```

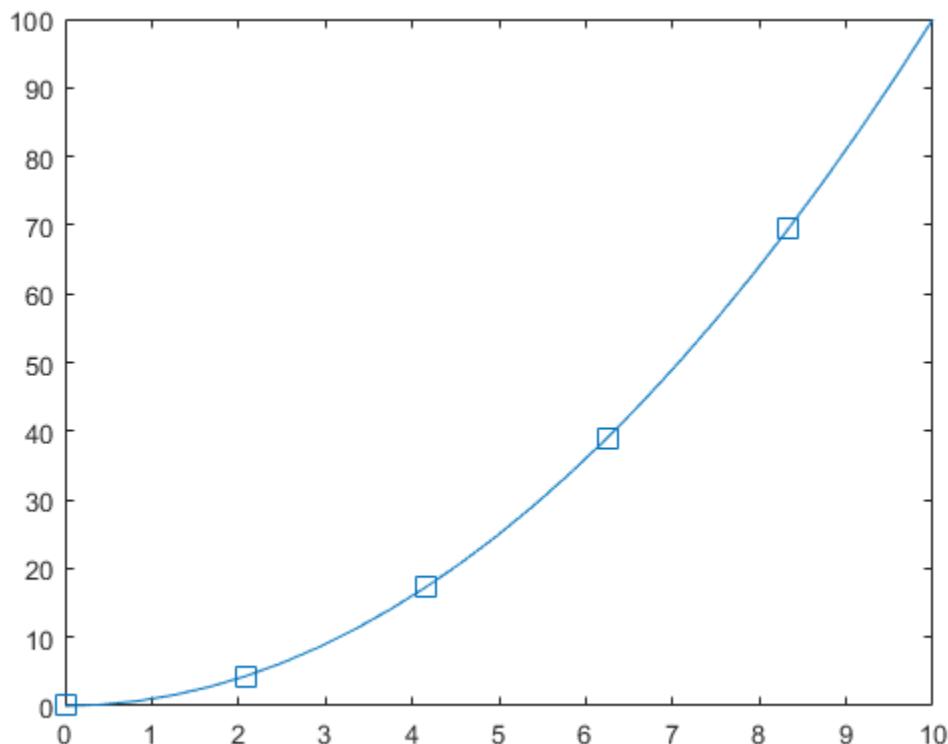


恢复为默认标记位置

修改标记位置，然后恢复为默认位置。

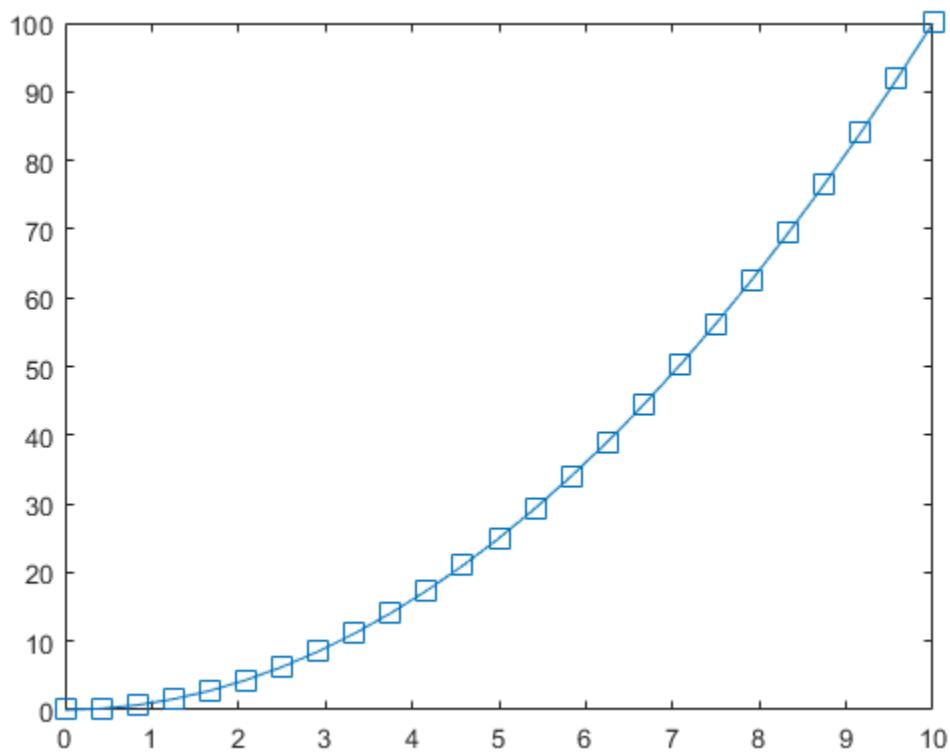
创建一个线图，并每隔五个数据点显示大的方形标记。将图形线条对象赋给变量 **p**，以便在创建后访问其属性。

```
x = linspace(0,10,25);
y = x.^2;
p = plot(x,y,'-s');
p.MarkerSize = 10;
p.MarkerIndices = 1:5:length(y);
```



将 `MarkerIndices` 属性重置为默认值，即从 1 到数据点数量之间的所有索引值组成的一个向量。

```
p.MarkerIndices = 1:length(y);
```



支持的标记符号

标记	说明	生成的标记
'o'	圆圈	○
'+'	加号	+
'*''	星号	*
'.'	点	•
'x'	叉号	×
'_'	水平线条	—
' '	垂直线条	
's'	方形	□
'd'	菱形	◇

标记	说明	生成的标记
'^'	上三角	△
'v'	下三角	▽
右三角	▷	
'<'	左三角	◁
'p'	五角形	☆
'h'	六角形	★
'none'	无标记	不适用

线条设定输入参数不支持有多个字符的标记选项。请改用一个字符的选项或设置 **Marker** 属性。

另请参阅

函数

`plot` | `scatter` | `loglog` | `plot3`

属性

`Line`

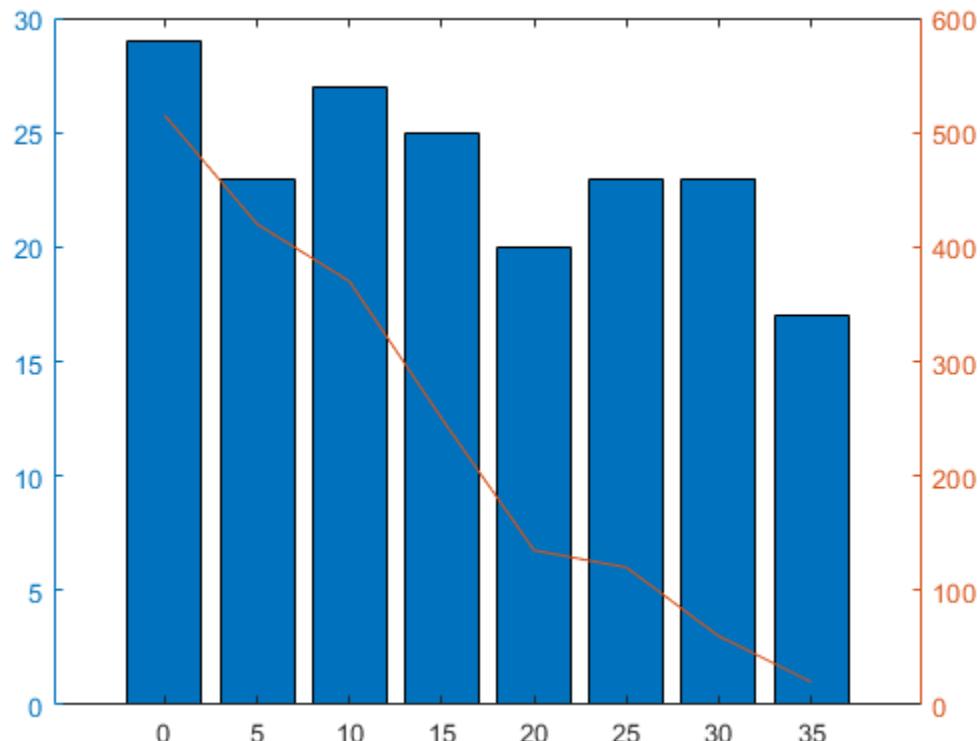
使用两个 y 轴合并线图和条形图

此示例说明如何使用两个不同的 y 轴合并线图和条形图。此外，还演示如何自定义线条和条形。

使用 `yyaxis` 创建包含两个 y 轴的图表。图形函数以图表的活动侧为目标。使用 `yyaxis` 控制活动侧。使用左侧的 y 轴绘制条形图。使用右侧的 y 轴绘制线图。将条形序列对象和图形线条对象赋给变量。

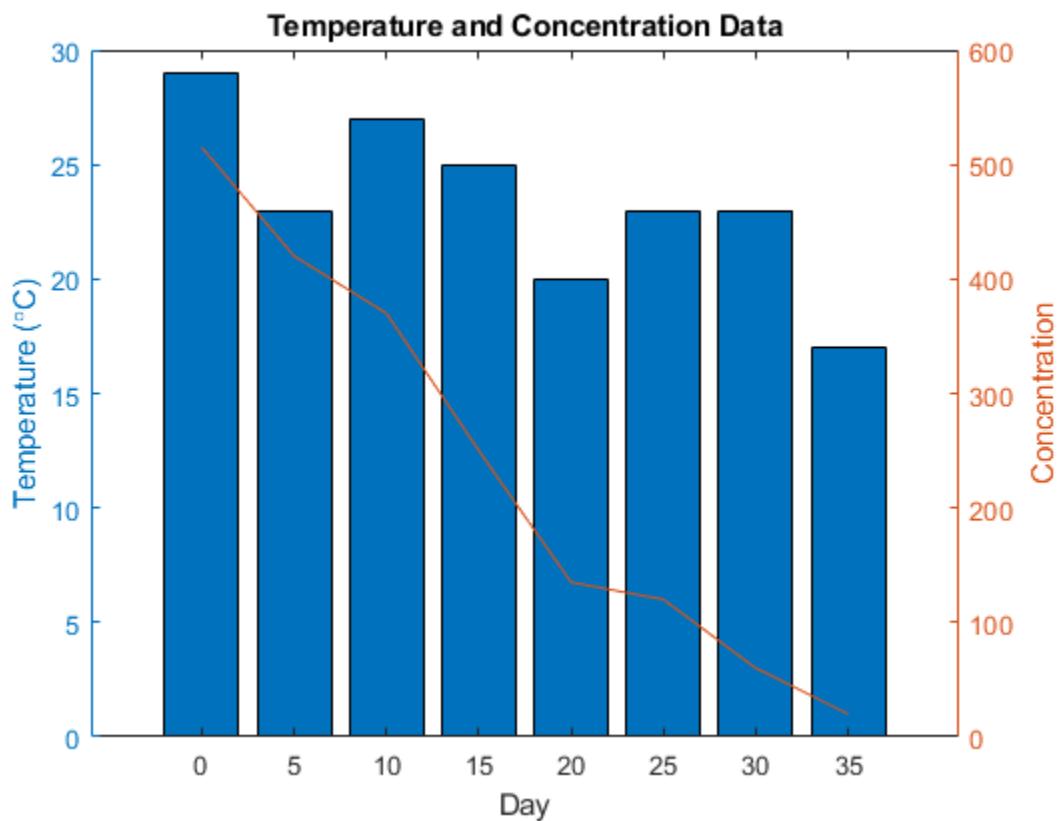
```
days = 0:5:35;
conc = [515 420 370 250 135 120 60 20];
temp = [29 23 27 25 20 23 23 17];
```

```
yyaxis left
b = bar(days,temp);
yyaxis right
p = plot(days,conc);
```



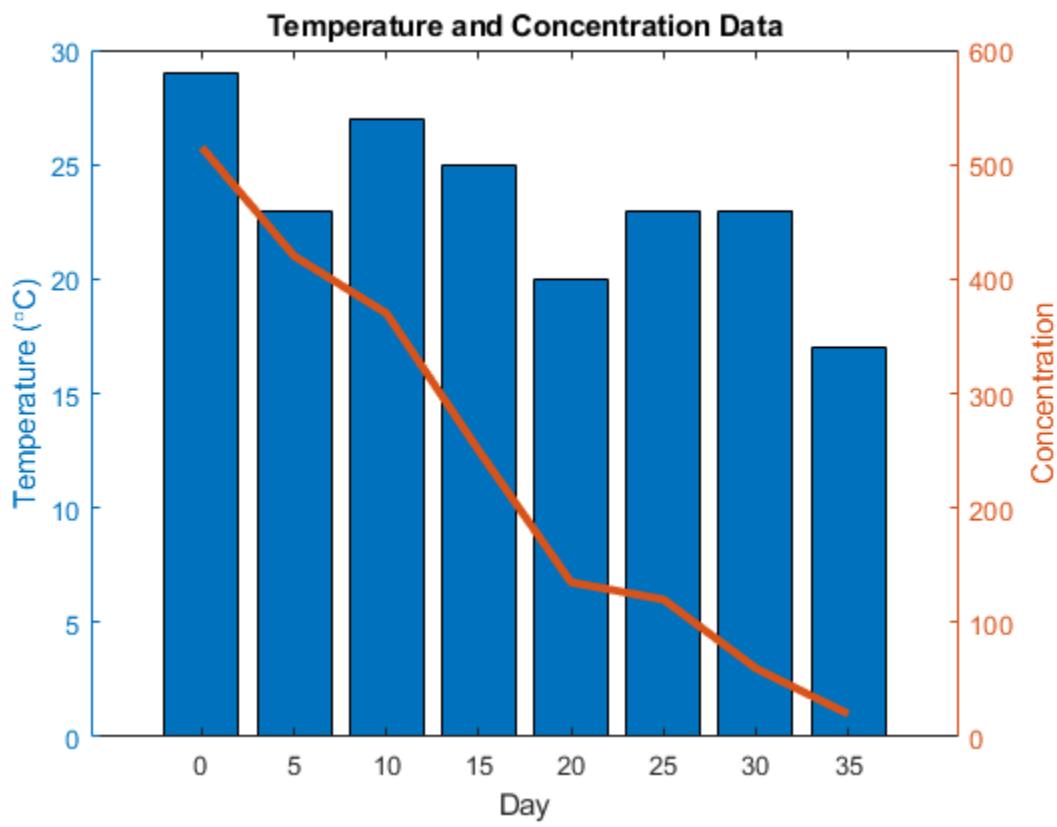
向图形添加标题和轴标签。

```
title('Temperature and Concentration Data')
xlabel('Day')
yyaxis left
ylabel('Temperature (\circC)')
yyaxis right
ylabel('Concentration')
```



更改图形线条的宽度以及更改条形颜色。

```
p.LineWidth = 3;  
b.FaceColor = [ 0 0.447 0.741];
```



另请参阅

函数

bar | plot | yyaxis | xlabel | ylabel | title | hold

属性

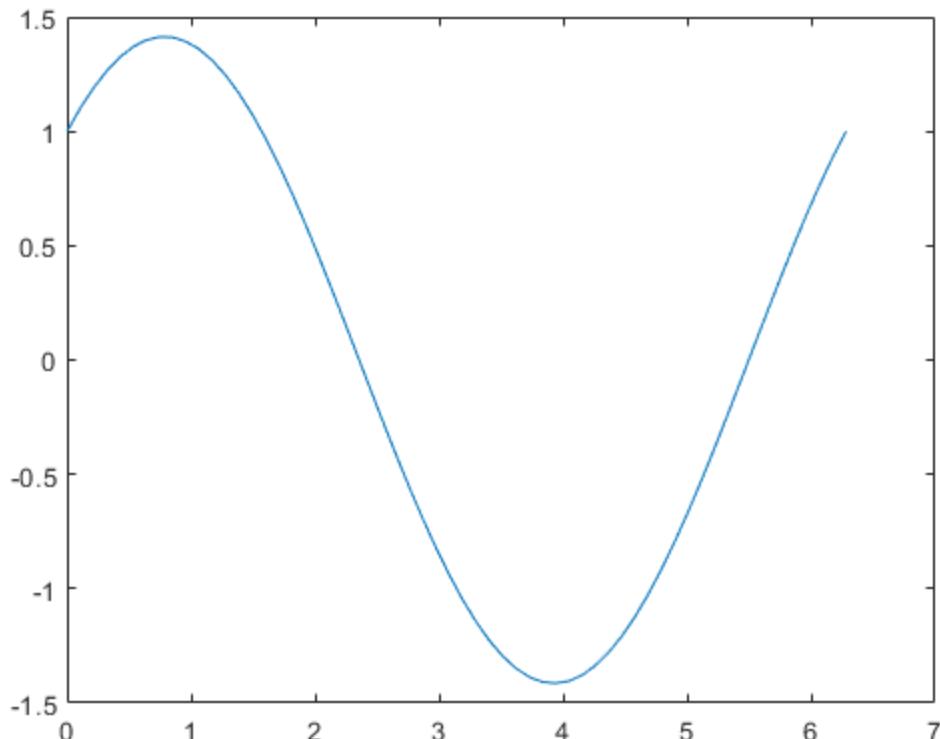
Line | Bar

合并线图和针状图

此示例演示如何合并一个线图和两个针状图。然后，显示如何添加标题、坐标轴标签和图例。

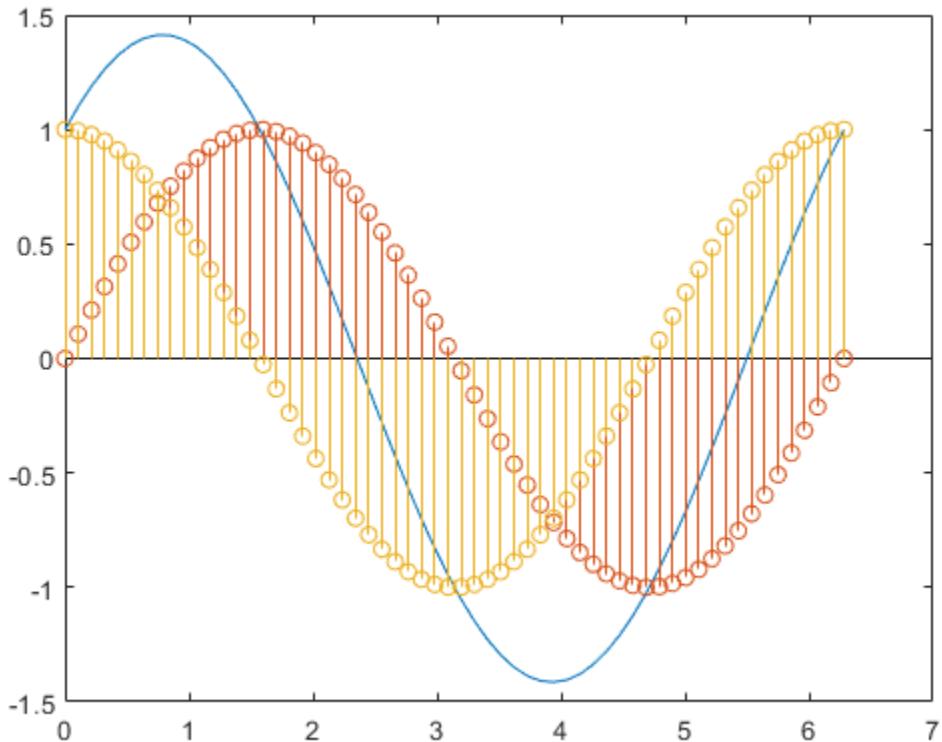
创建数据并绘制一条线条。

```
x = linspace(0,2*pi,60);
a = sin(x);
b = cos(x);
plot(x,a+b)
```



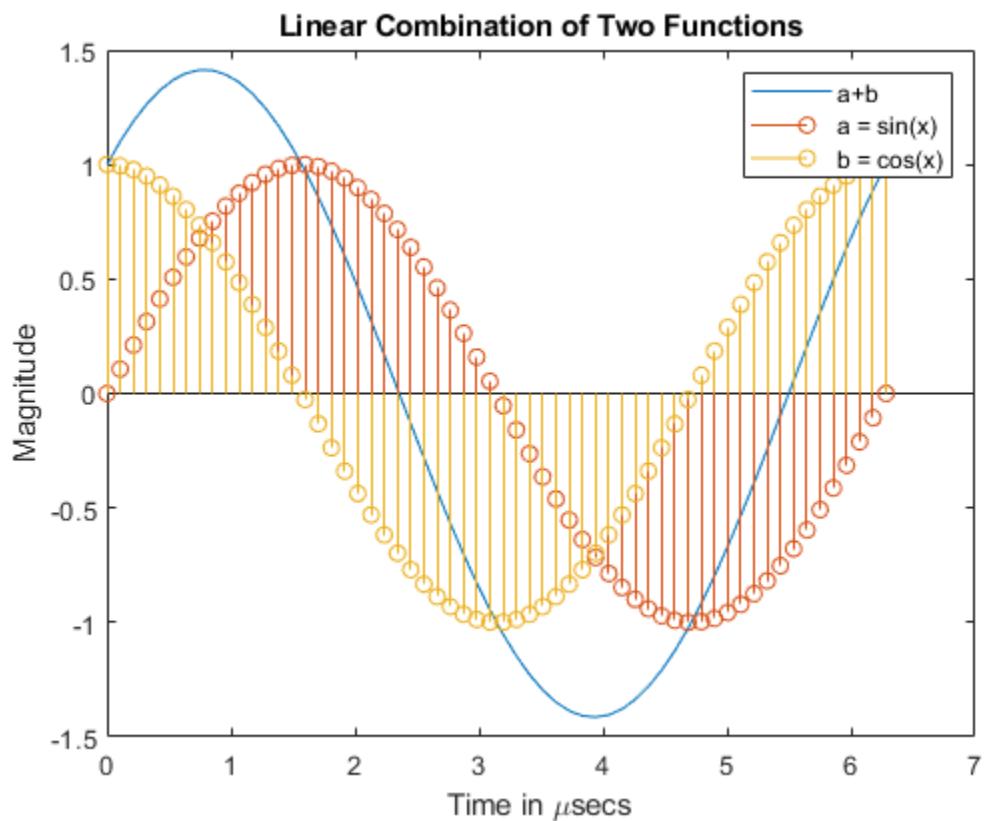
在坐标区中添加两个针状图。使用 `hold on` 防止新绘图替换现有的绘图。

```
hold on
stem(x,a)
stem(x,b)
hold off
```



添加标题、坐标轴标签和图例。按照创建绘图的顺序指定图例说明。

```
title('Linear Combination of Two Functions')
xlabel('Time in \musecs')
ylabel('Magnitude')
legend('a+b','a = sin(x)','b = cos(x)')
```



另请参阅

`plot | stem | hold`

叠加阶梯图和线图

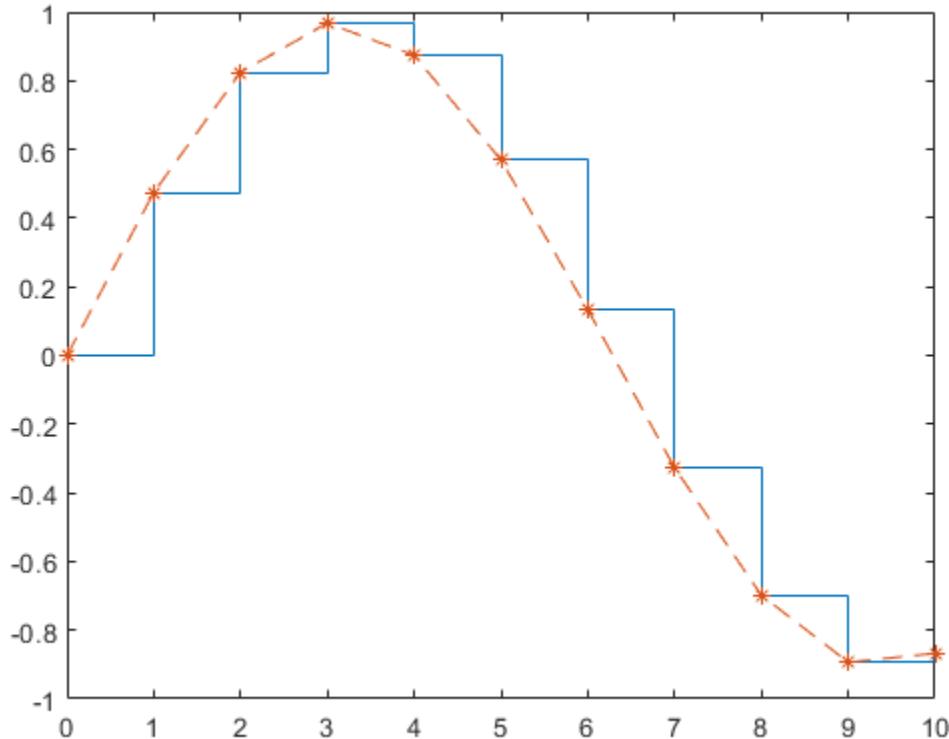
此示例演示如何在阶梯图上叠加线图。

定义要绘图的数据。

```
alpha = 0.01;
beta = 0.5;
t = 0:10;
f = exp(-alpha*t).*sin(beta*t);
```

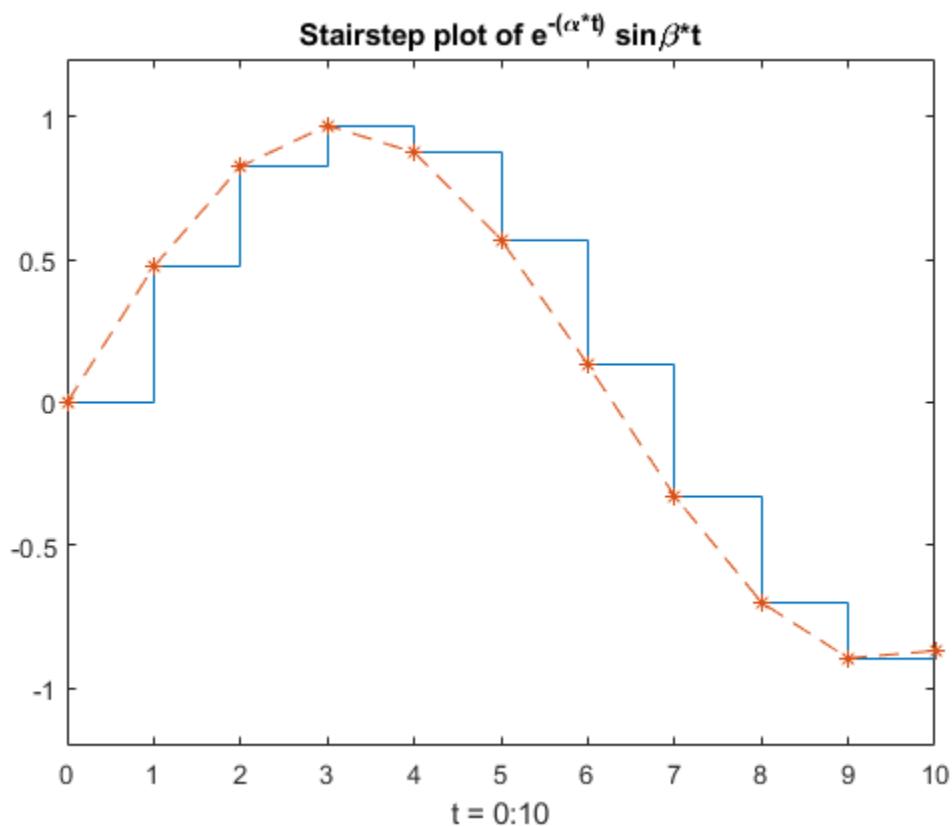
将 f 显示为阶梯图。使用 **hold** 函数保留阶梯图。使用带有星形标记的虚线添加 f 线图。

```
stairs(t,f)
hold on
plot(t,f,'--*')
hold off
```



使用 **axis** 函数设置坐标轴范围。标记 x 轴并向图形添加一个标题。

```
axis([0,10,-1.2,1.2])
xlabel('t = 0:10')
title('Stairstep plot of e^{-(\alpha*t)} sin\beta*t')
```



另请参阅

`stairs | plot | axis`

带有置信边界的线图

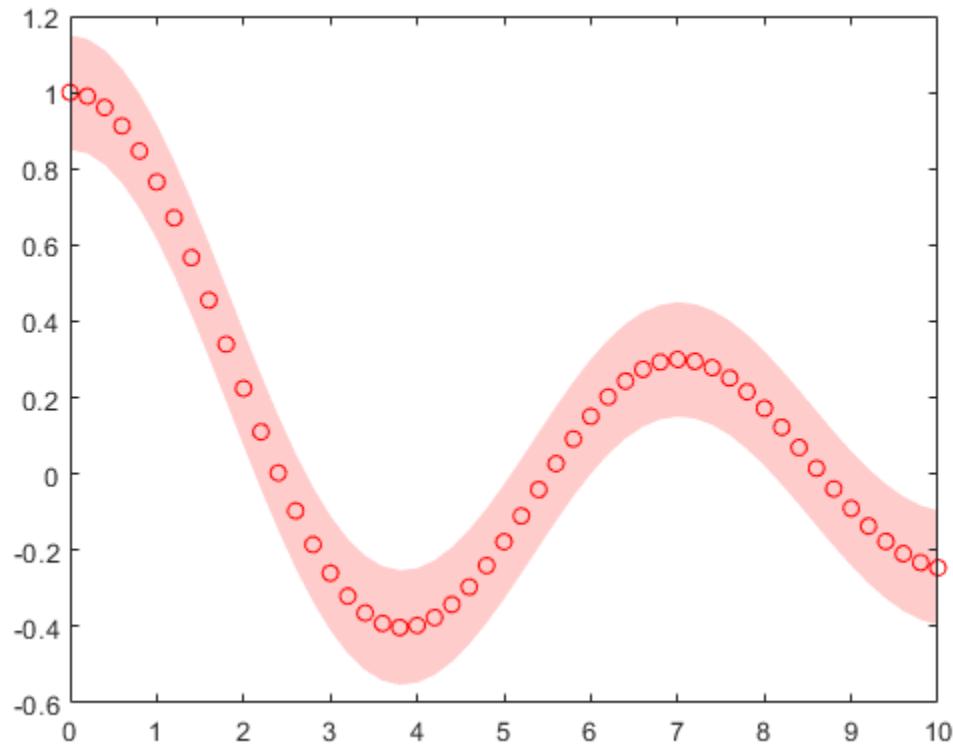
使用 `fill` 函数绘制置信边界，同时使用 `plot` 函数绘制数据点，以此方式创建含有置信边界的绘图。使用圆点表示法语法 `object.PropertyName` 自定义绘图的外观。

```
x = 0:0.2:10;
y = besselj(0, x);

xconf = [x x(end:-1:1)] ;
yconf = [y+0.15 y(end:-1:1)-0.15];

figure
p = fill(xconf,yconf,'red');
p.FaceColor = [1 0.8 0.8];
p.EdgeColor = 'none';

hold on
plot(x,y,'ro')
hold off
```



绘制虚数和复数数据图

绘制一个复数输入

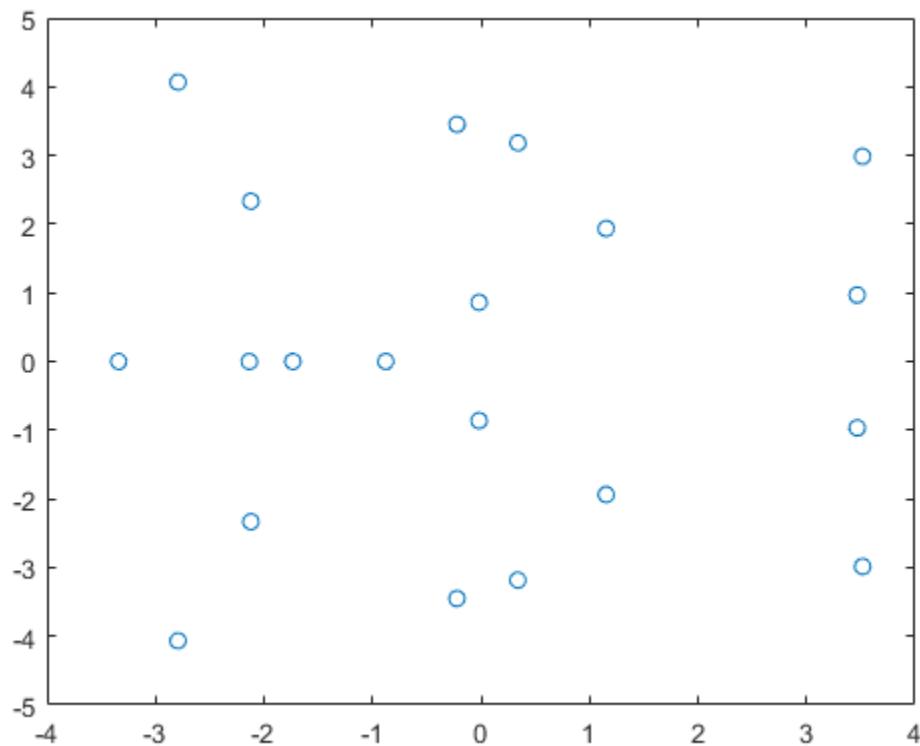
此示例演示如何绘制复数向量 z 的虚部与实部。在此复数输入中，**plot(z)** 等同于 **plot(real(z),imag(z))**，其中 **real(z)** 是 z 的实部，**imag(z)** 是 z 的虚部。

将 z 定义为随机矩阵的特征值向量。

```
z = eig(randn(20));
```

绘制 z 的虚部与 z 的实部。在每个数据点处显示一个圆圈。

```
figure  
plot(z,'o')
```



绘制多个复数输入

此示例演示如何绘制两个复数向量 z_1 和 z_2 的虚部与实部。如果将多个复数参数传递给 **plot**，例如 **plot(z1,z2)**，则 MATLAB® 会忽略输入的虚部，仅绘制实部。要绘制多个复数输入的实部与虚部，必须将实部和虚部显式传递给 **plot**。

定义复数数据。

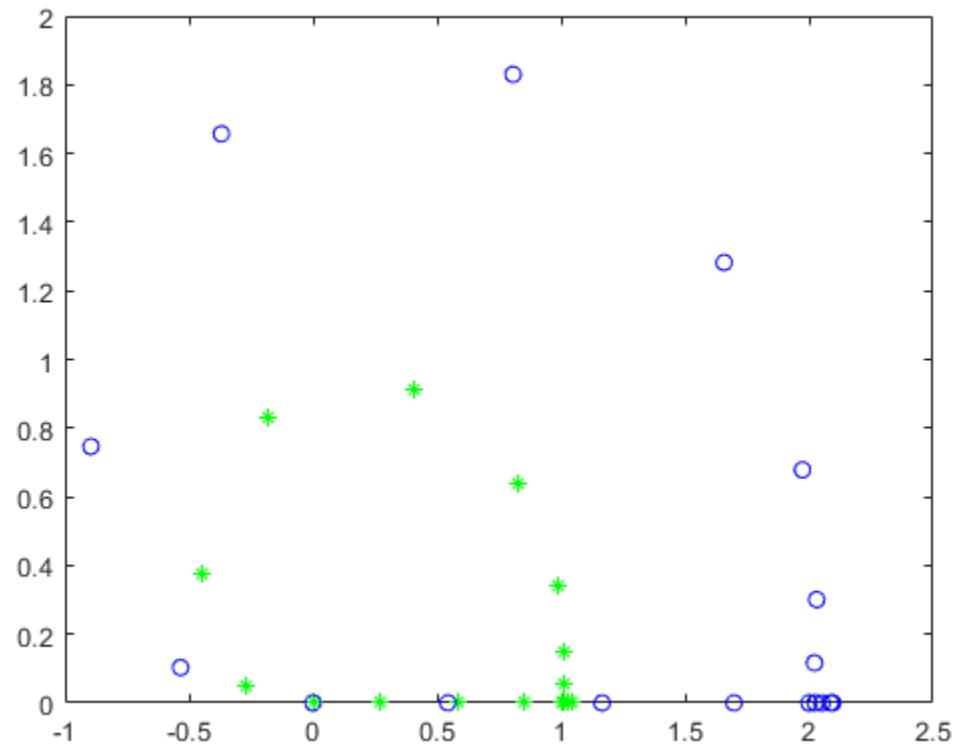
```
x = -2:0.25:2;  
z1 = x.^exp(-x.^2);  
z2 = 2*x.^exp(-x.^2);
```

使用 `real` 和 `imag` 函数求出每个向量的实部和虚部。然后绘制数据图。

```
real_z1 = real(z1);
imag_z1 = imag(z1);

real_z2 = real(z2);
imag_z2 = imag(z2);

plot(real_z1,imag_z1,'g*',real_z2,imag_z2,'bo')
```



另请参阅

`plot` | `real` | `imag`

饼图、条形图和直方图

- “条形图种类” (第 2-2 页)
- “修改条形图的基线” (第 2-8 页)
- “叠加条形图” (第 2-11 页)
- “带有误差条的条形图” (第 2-15 页)
- “按高度为三维条形着色” (第 2-16 页)
- “使用叠加区域图对比数据集” (第 2-18 页)
- “偏移占比最大的饼图扇区” (第 2-22 页)
- “向饼图添加图例” (第 2-24 页)
- “为饼图添加文本和百分比标签” (第 2-26 页)
- “使用二元直方图进行颜色分析” (第 2-28 页)
- “控制分类直方图的显示” (第 2-34 页)
- “替换不建议使用的 hist 和 histc 实例” (第 2-41 页)

条形图种类

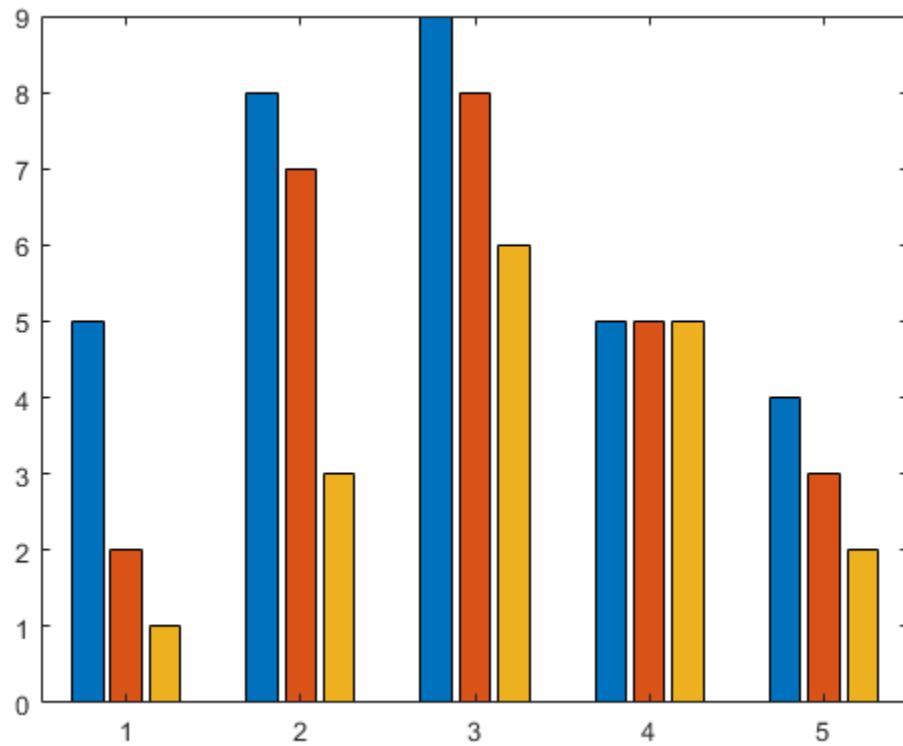
如果需要查看一段时间内的结果、对比不同数据集的结果，或展示单个元素对汇总量的贡献和影响，则条形图会很有用处。

默认情况下，条形图会将一个向量或矩阵中的每个元素表现为一个条形，条形的高度与元素的值成比例。

二维条形图

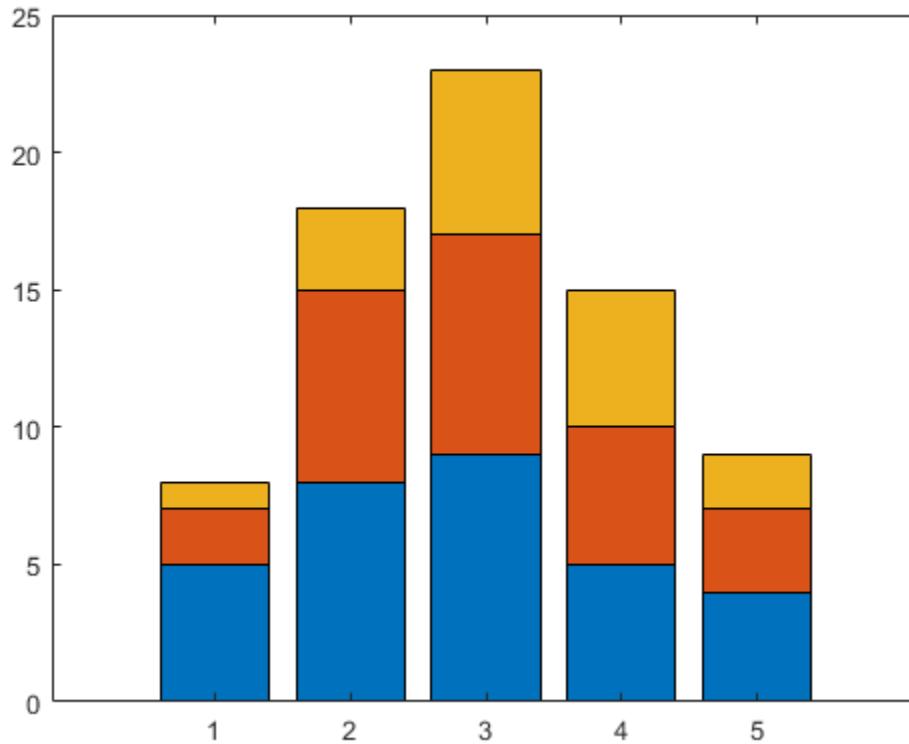
`bar` 函数沿着 x 轴分布条形。同一行的矩阵元素分在同一组。例如，如果矩阵包含五行三列数据，则 `bar` 将沿着 x 轴显示五组条形，每一组中包括三个条形。第一组条形表示 Y 中第一行的元素。

```
Y = [5,2,1
      8,7,3
      9,8,6
      5,5,5
      4,3,2];
figure
bar(Y)
```



要堆叠一行中的元素，请指定 `bar` 函数的 `stacked` 选项。

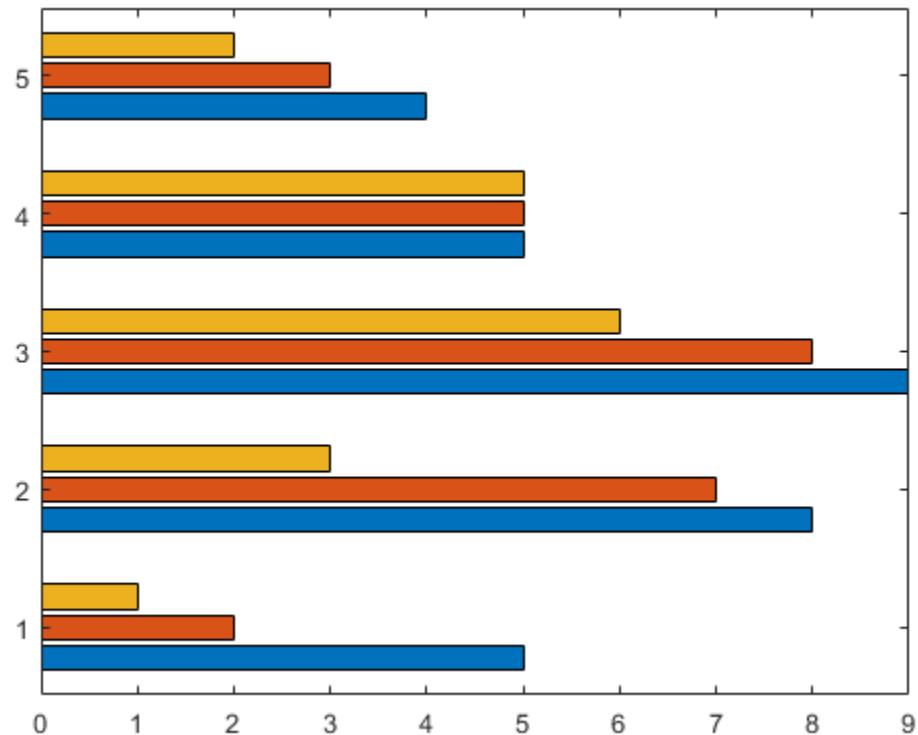
```
figure
bar(Y,'stacked')
```



二维水平条形图

`barh` 函数沿着 y 轴分布条形。同一行的矩阵元素分在同一组。

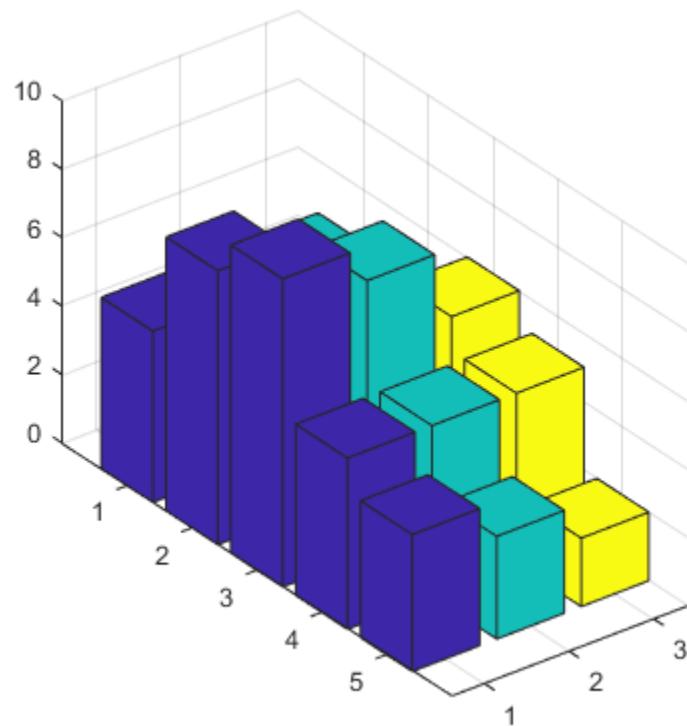
```
Y = [5,2,1  
     8,7,3  
     9,8,6  
     5,5,5  
     4,3,2];  
figure  
barh(Y)
```



三维条形图

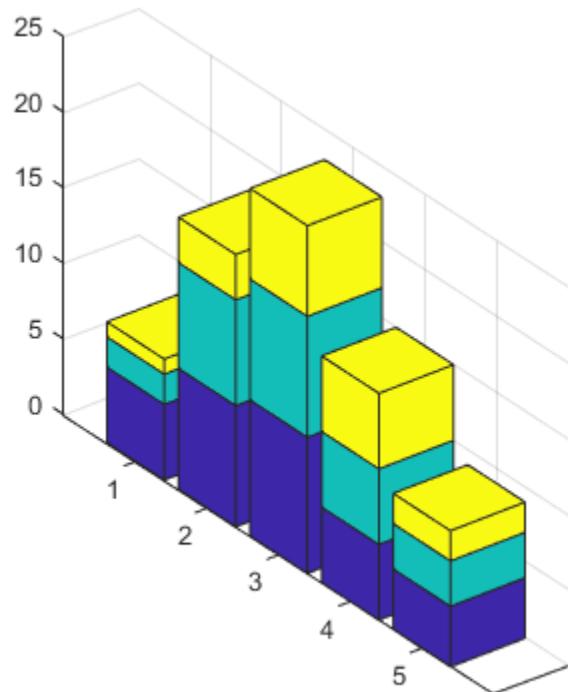
bar3 函数将每个元素绘制为一个单独的三维块，沿着 y 轴分布每列元素。

```
Y = [5,2,1  
     8,7,3  
     9,8,6  
     5,5,5  
     4,3,2];  
figure  
bar3(Y)
```



要堆叠一行中的元素，请指定 `bar3` 函数的 `stacked` 选项。

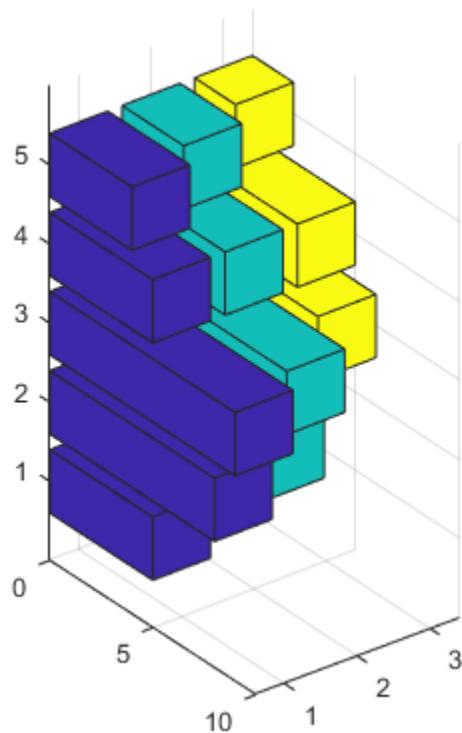
```
figure  
bar3(Y,'stacked')
```



三维水平条形图

bar3h 函数将每个元素绘制为一个单独的三维块，沿着 z 轴分布每列元素。

```
Y = [5,2,1  
     8,7,3  
     9,8,6  
     5,5,5  
     4,3,2];  
figure  
bar3h(Y)
```



另请参阅

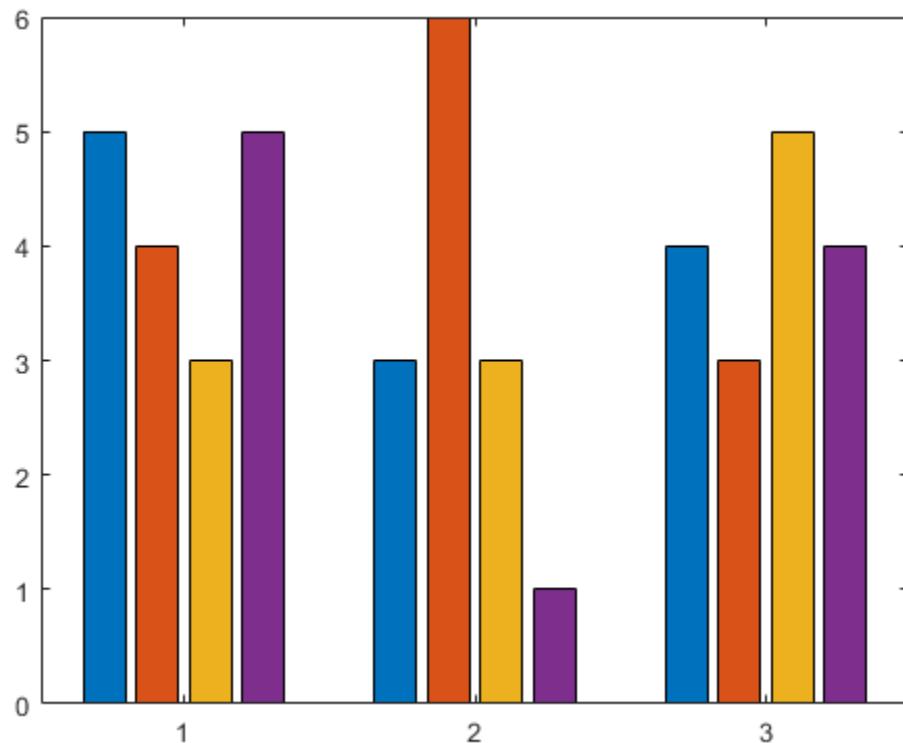
[bar](#) | [barh](#) | [bar3](#) | [bar3h](#)

修改条形图的基线

此示例演示如何修改条形图基线的属性。

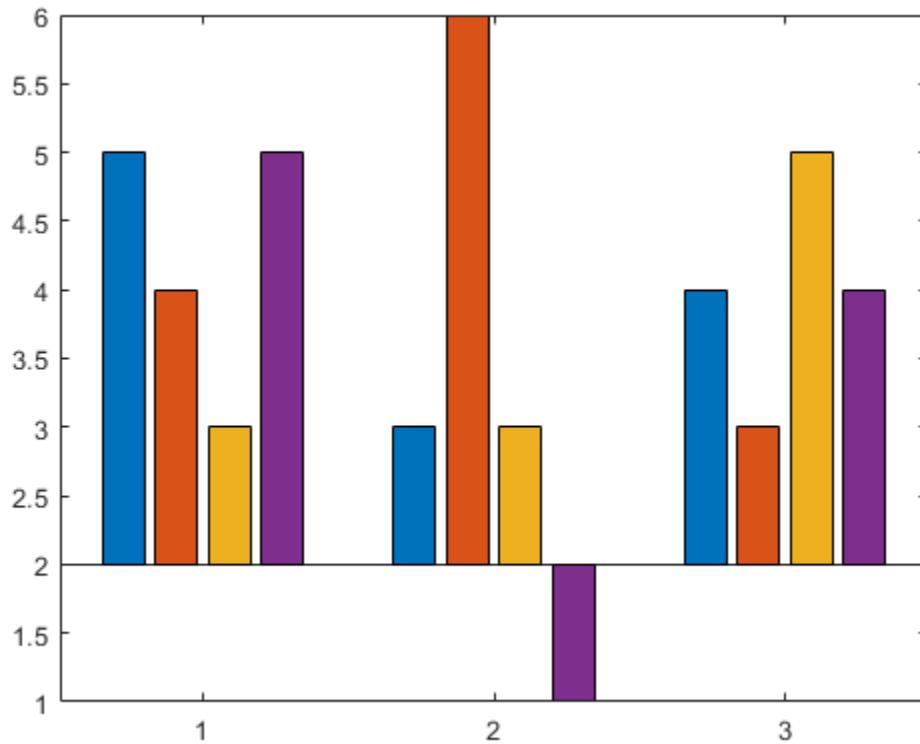
创建一个四列矩阵的条形图。 `bar` 函数为矩阵的每一列创建一个条形序列。将这四个条形序列返回为 `b`。

```
Y = [5, 4, 3, 5;
      3, 6, 3, 1;
      4, 3, 5, 4];
b = bar(Y);
```



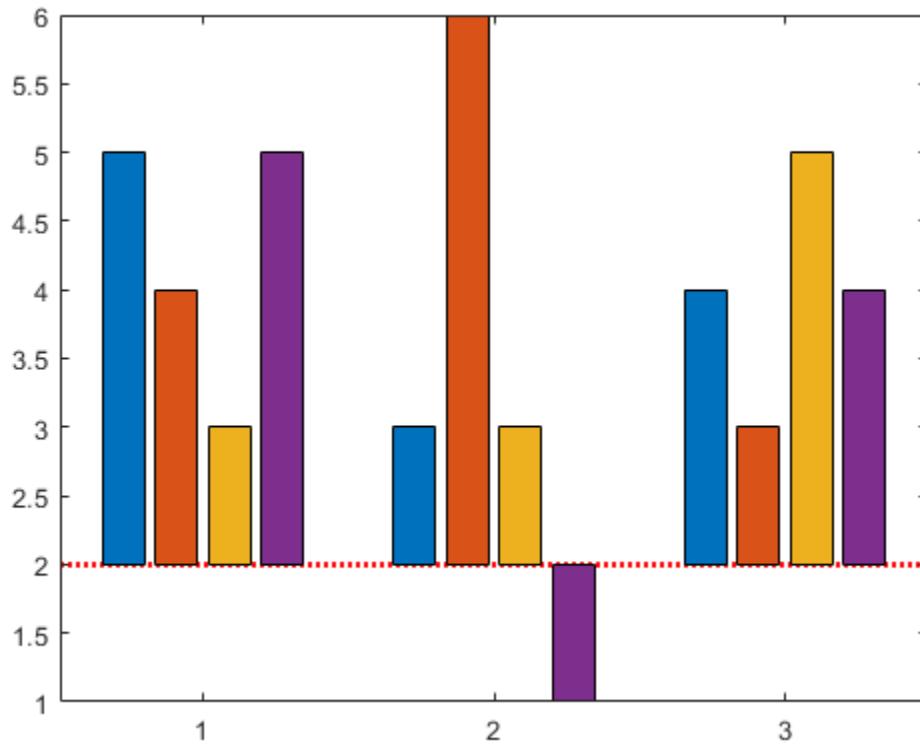
图表中的所有条形序列具有相同的基线。通过设置任一条形序列的 `BaseValue` 属性将基线的值更改为 2。使用圆点表示法设置属性。

```
b(1).BaseValue = 2;
```



将基线更改为粗的红色点线。

```
b(1).BaseLine.LineStyle = '·';  
b(1).BaseLine.Color = 'red';  
b(1).BaseLine.LineWidth = 2;
```



另请参阅

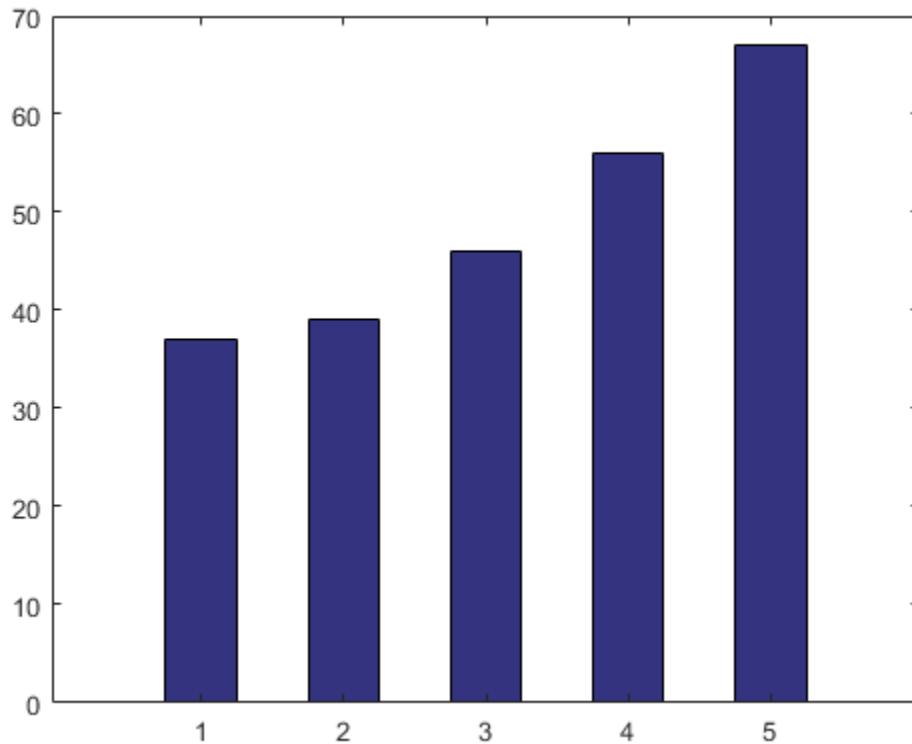
bar | barh

叠加条形图

此示例演示如何叠加两个条形图并指定条形的颜色和宽度。然后演示如何添加图例、显示网格线和指定刻度标签。

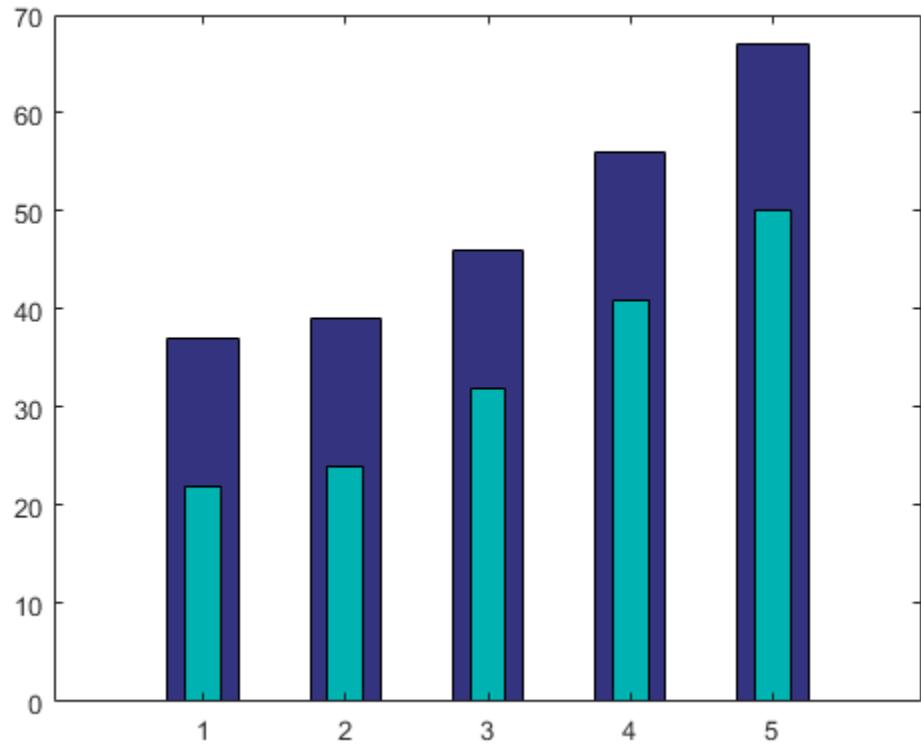
创建一个条形图。将条形宽度设置为 0.5，使条形使用 50% 的可用空间。通过将 FaceColor 属性设置为一个 RGB 颜色值来指定条形的颜色。

```
x = [1 2 3 4 5];
temp_high = [37 39 46 56 67];
w1 = 0.5;
bar(x,temp_high,w1,'FaceColor',[0.2 0.2 0.5])
```



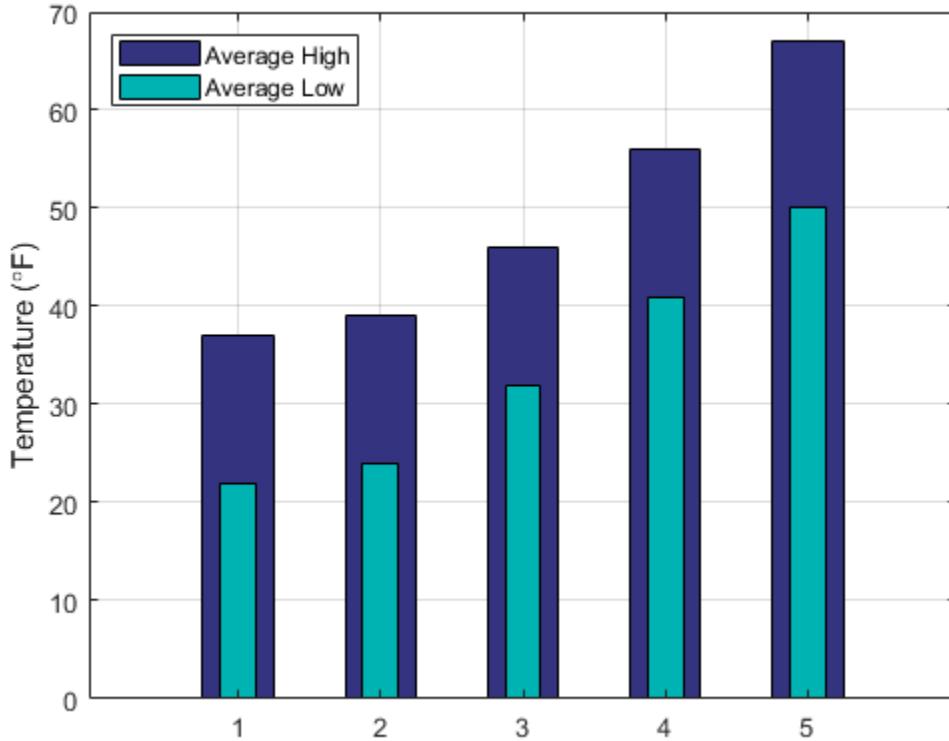
在第一个条形图上绘制第二个条形图。使用 hold 函数保留第一个图形。将条形宽度设置为 .25，使条形使用 25% 的可用空间。为该条形颜色指定一个不同的 RGB 颜色值。

```
temp_low = [22 24 32 41 50];
w2 = .25;
hold on
bar(x,temp_low,w2,'FaceColor',[0 0.7 0.7])
hold off
```



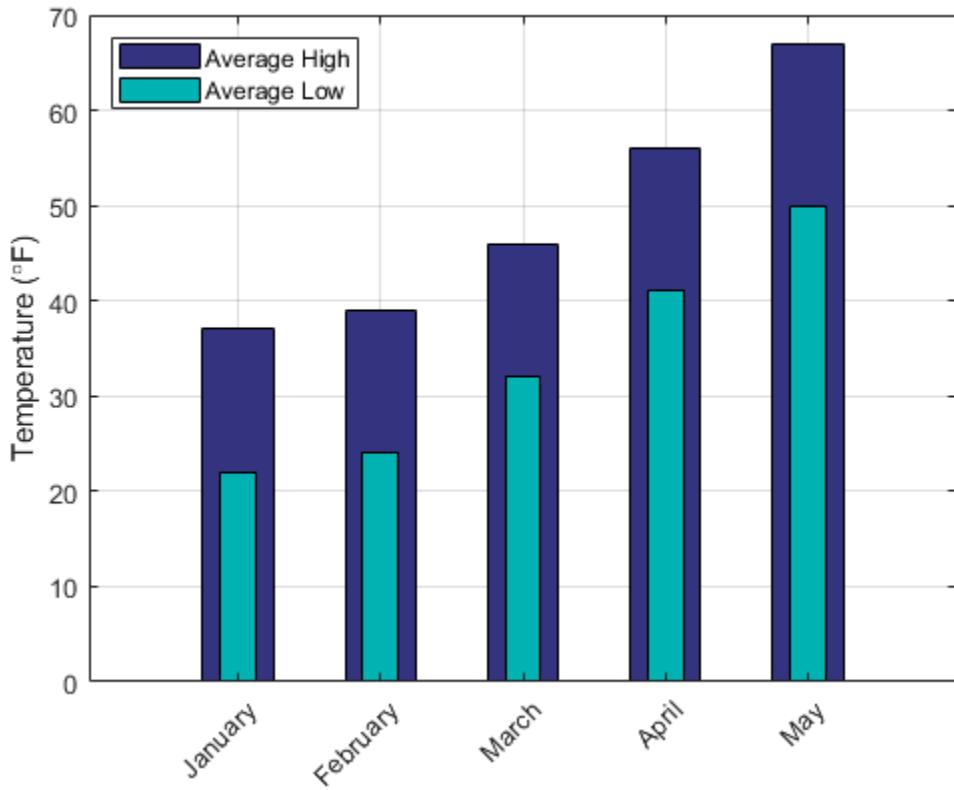
添加网格线、y 轴标签，并在左上角添加图例。按照创建图表的顺序指定图例说明。

```
grid on  
ylabel('Temperature (\circF)')  
legend({'Average High','Average Low'},'Location','northwest')
```



通过设置坐标区对象的 `XTick` 和 `XTickLabel` 属性，指定 x 轴刻度标签。`XTick` 属性用于指定沿 x 轴的刻度值位置。`XTickLabel` 属性用于指定每个刻度值要使用的文本。使用 `XTickLabelRotation` 属性旋转标签。使用圆点表示法设置属性。

```
ax = gca;
ax.XTick = [1 2 3 4 5];
ax.XTickLabels = {'January','February','March','April','May'};
ax.XTickLabelRotation = 45;
```



另请参阅

bar | barh | hold

带有误差条的条形图

结合使用 `bar` 和 `errorbar` 函数创建带有误差条的条形图。

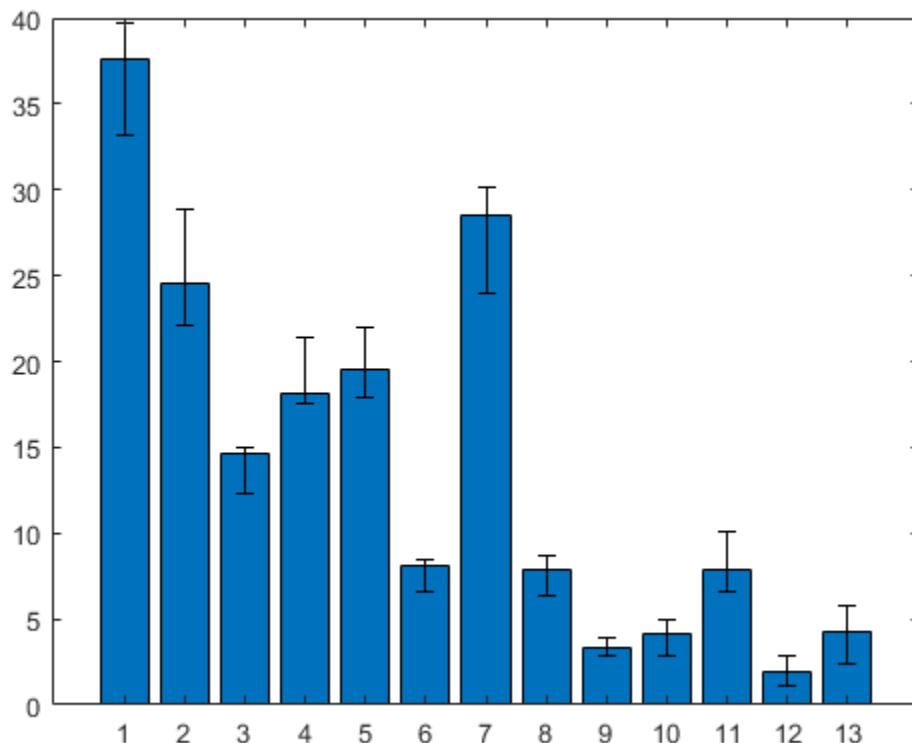
```
x = 1:13;
data = [37.6 24.5 14.6 18.1 19.5 8.1 28.5 7.9 3.3 4.1 7.9 1.9 4.3]';
errhigh = [2.1 4.4 0.4 3.3 2.5 0.4 1.6 0.8 0.6 0.8 2.2 0.9 1.5];
errlow = [4.4 2.4 2.3 0.5 1.6 1.5 4.5 1.5 0.4 1.2 1.3 0.8 1.9];

bar(x,data)

hold on

er = errorbar(x,data,errlow,errhigh);
er.Color = [0 0 0];
er.LineStyle = 'none';

hold off
```



另请参阅

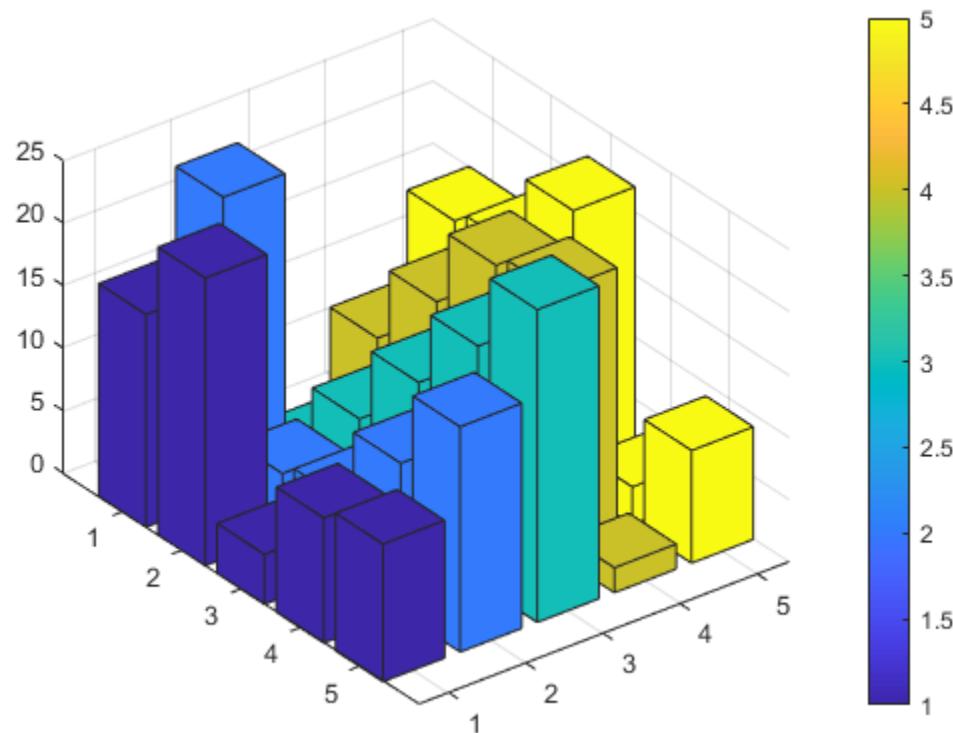
`bar` | `hold` | `errorbar`

按高度为三维条形着色

此示例演示如何根据条形高度为条形着色，以此方式来修改三维条形图。

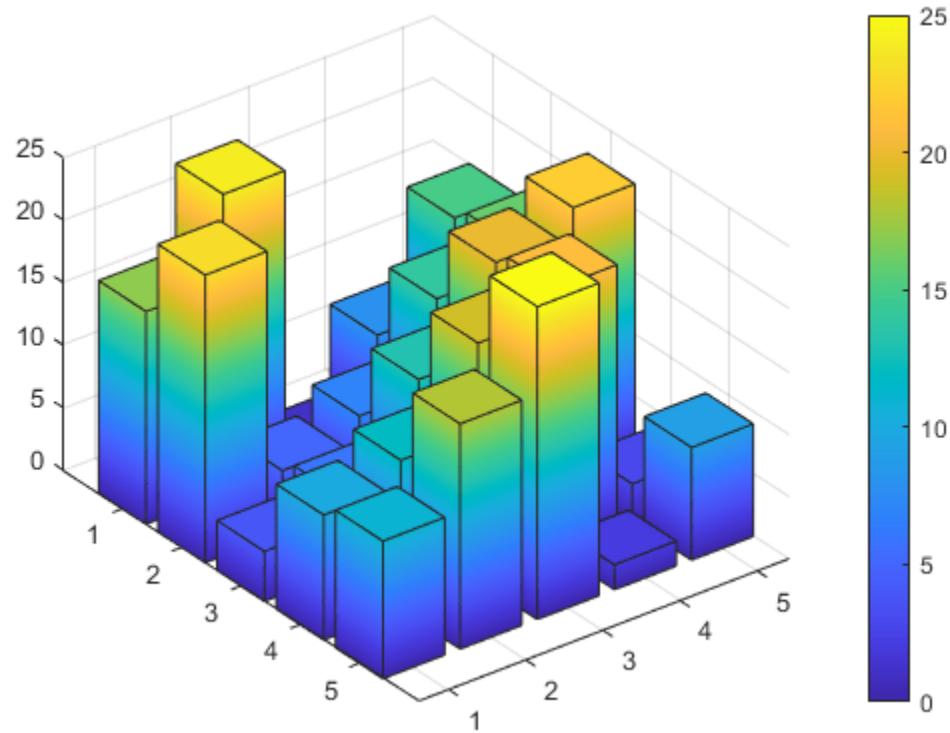
创建使用 `magic` 函数得到的数据的三维条形图。在数组 `b` 中返回用于创建条形图的曲面对象。向图形添加颜色栏。

```
Z = magic(5);
b = bar3(Z);
colorbar
```



对每个曲面对象，从 `ZData` 属性取得 `z` 坐标数组。使用该数组设置 `CData` 属性，该属性用于定义顶点颜色。通过将曲面对象的 `FaceColor` 属性设置为 '`interp`' 来插入面颜色。使用圆点表示法查询和设置属性。

```
for k = 1:length(b)
    zdata = b(k).ZData;
    b(k).CData = zdata;
    b(k).FaceColor = 'interp';
end
```



每个条形的高度决定了它的颜色。您可以通过对比条形颜色和颜色栏来估算条形的高度。

另请参阅

`bar3 | colorbar`

使用叠加区域图对比数据集

此示例演示如何通过叠加数据集区域图对比数据集。

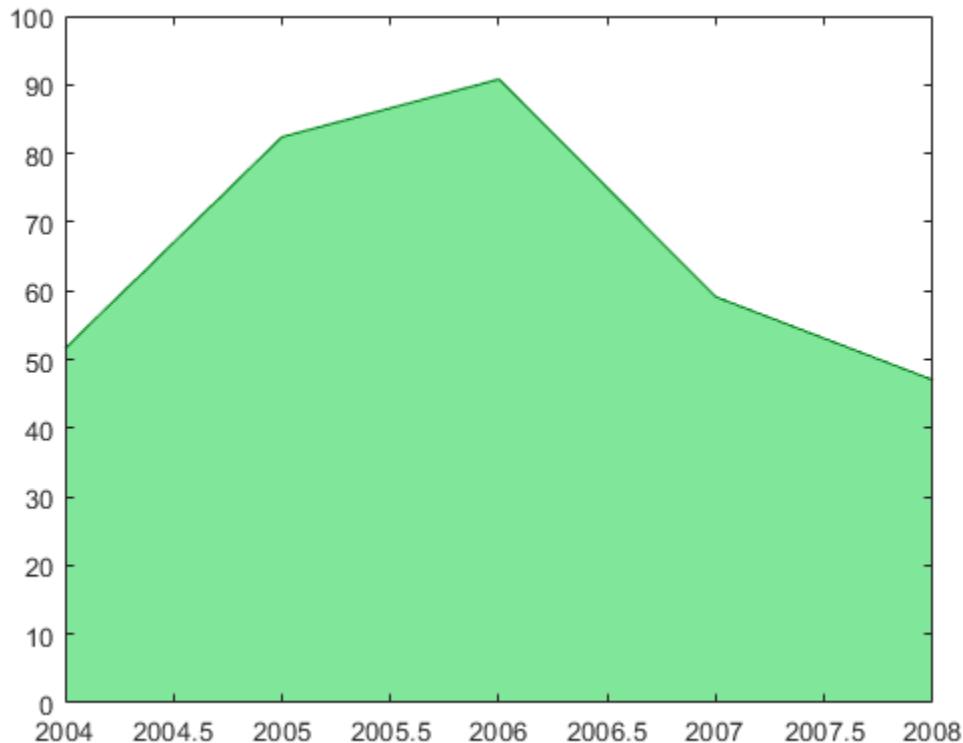
叠加两个区域图

创建从 2004 到 2008 年的销售和支出数据。

```
years = 2004:2008;
sales = [51.6 82.4 90.8 59.1 47.0];
expenses = [19.3 34.2 61.4 50.5 29.4];
```

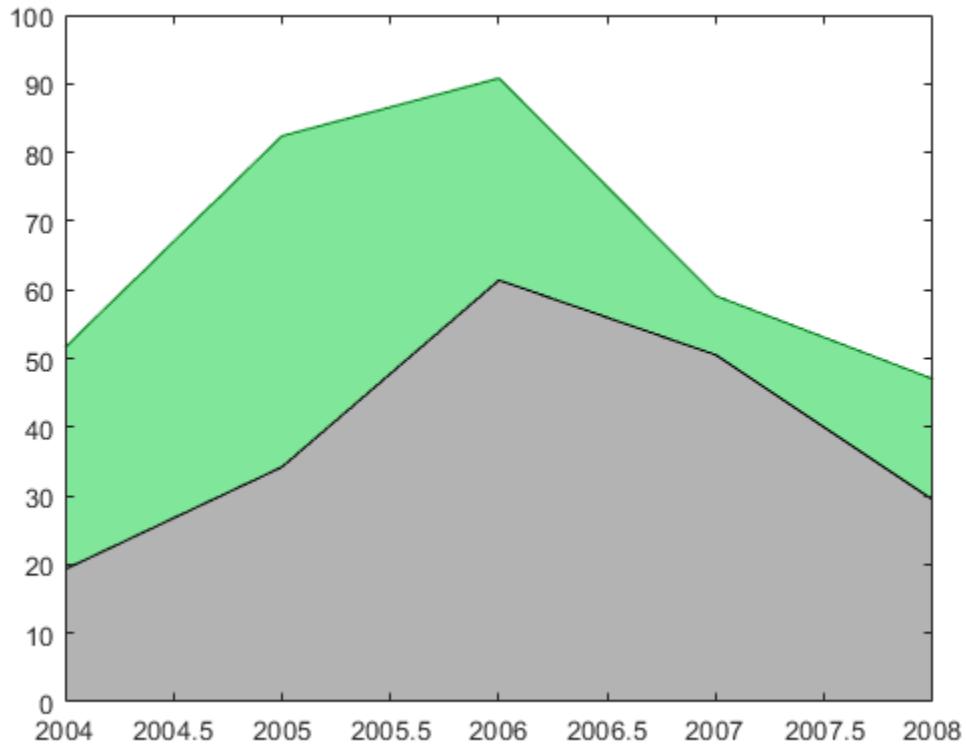
将销售和支出显示为同一套坐标区下两个单独的区域图。首先，绘制 `sales` 的区域图。通过使用 RGB 三元组颜色值设置 `FaceColor` 和 `EdgeColor` 属性，更改区域图的颜色。

```
area(years,sales,'FaceColor',[0.5 0.9 0.6],'EdgeColor',[0 0.5 0.1])
```



使用 `hold` 命令以防止新图形替换现有图形。绘制 `expenses` 的另一个区域图。然后，将 `hold` 状态重新设置为 `off`。

```
hold on
area(years,expenses,'FaceColor',[0.7 0.7 0.7],'EdgeColor','k')
hold off
```



添加网格线

沿 x 轴设置对应各个年份的刻度线。为每个刻度线绘制一条网格线。通过设置 Layer 属性，在区域图上显示网格线。使用圆点表示法设置属性。

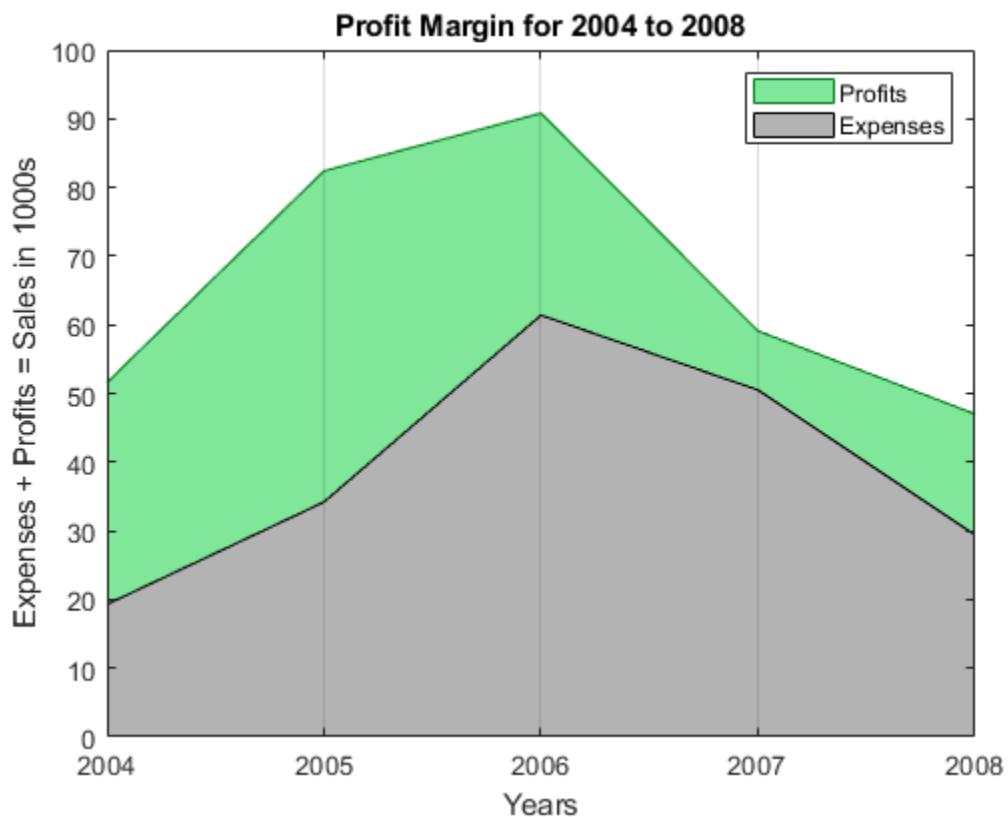
```
ax = gca; % current axes  
ax.XTick = years;  
ax.XGrid = 'on';  
ax.Layer = 'top';
```



添加标题、坐标轴标签和图例

为图形提供标题，并添加轴标签。向图形添加图例以指示利润和支出区域。

```
title('Profit Margin for 2004 to 2008')
xlabel('Years')
ylabel('Expenses + Profits = Sales in 1000s')
legend('Profits','Expenses')
```



另请参阅

[area](#) | [hold](#) | [legend](#)

偏移占比最大的饼图扇区

此示例演示如何创建饼图并自动偏移占比最大的饼图扇区。

建立一个三列数组 X，每一列包含一个特定产品为期 5 年的年销售数据。

```
X = [19.3, 22.1, 51.6  
      34.2, 70.3, 82.4  
      61.4, 82.9, 90.8  
      50.5, 54.9, 59.1  
      29.4, 36.3, 47.0];
```

通过对每列求和计算出每个产品 5 年的总销售额。将结果保存到 `product_totals`。

```
product_totals = sum(X);
```

使用 `max` 函数找出 `product_totals` 中的最大元素并返回该元素的索引 `ind`。

```
[c,ind] = max(product_totals);
```

使用 `pie` 函数的输入参数 `explode` 偏移一个饼图扇区。`explode` 参数是一个由零和非零值组成的向量，其中非零值表示要偏移的扇区。将 `explode` 初始化为一个由零组成的三元素向量。

```
explode = zeros(1,3);
```

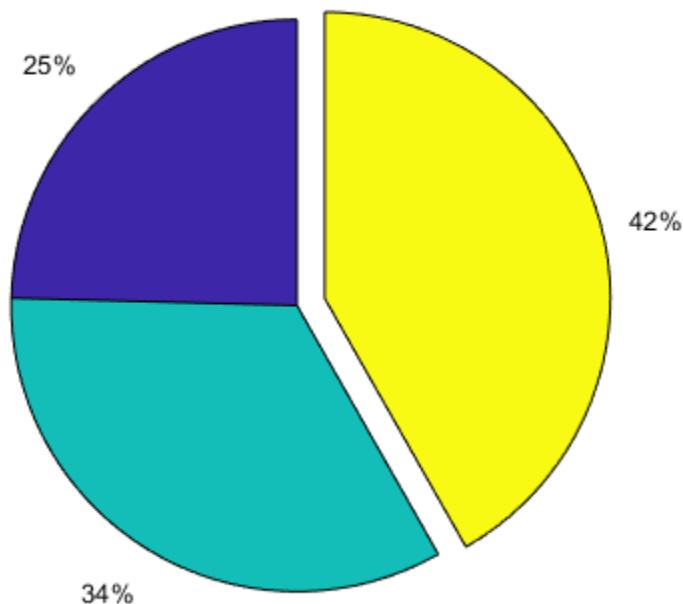
使用 `product_totals` 中最大元素的索引将对应的 `explode` 元素设为 1。

```
explode(ind) = 1;
```

创建包含每个产品销售总额的饼图，并偏移具有最大销售总额的产品所在的饼图扇区。

```
figure  
pie(product_totals=explode)  
title('Sales Contributions of Three Products')
```

Sales Contributions of Three Products



另请参阅

pie | max | zeros

相关示例

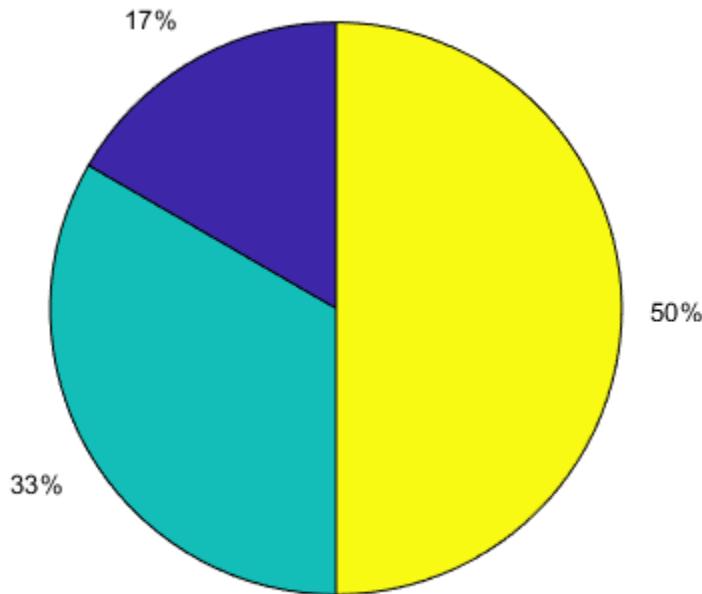
- “向饼图添加图例” (第 2-24 页)

向饼图添加图例

此示例演示如何向饼图添加显示每个扇区说明的图例。

定义 `x` 并创建一个饼图。

```
x = [1,2,3];
figure
pie(x)
```

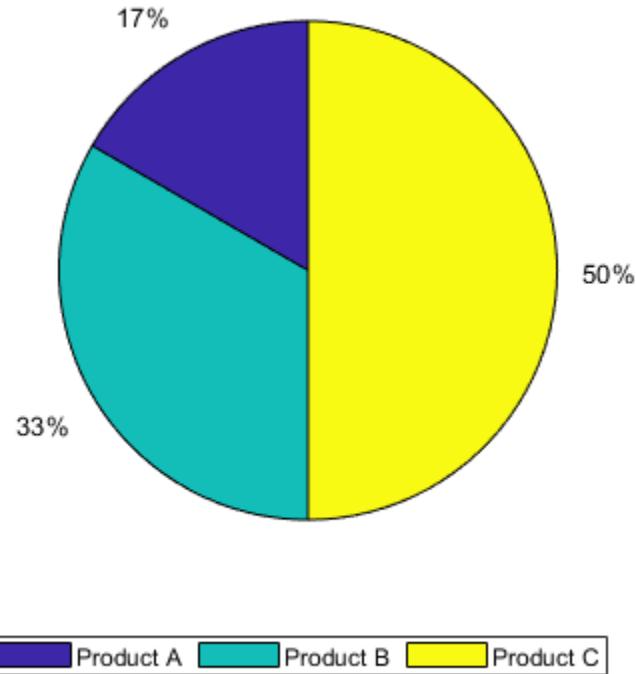


在元胞数组 `labels` 中指定每个饼图扇区的说明。按照您在 `x` 中指定数据的顺序指定说明。

```
labels = {'Product A','Product B','Product C'};
```

在饼图下方显示水平图例。将包含在 `labels` 中的说明传递给 `legend` 函数。将图例的 `Location` 属性设置为 '`southoutside`'，并将其 `Orientation` 属性设置为 '`horizontal`'。

```
legend(labels,'Location','southoutside','Orientation','horizontal')
```



另请参阅

pie | legend

相关示例

- “偏移占比最大的饼图扇区” (第 2-22 页)

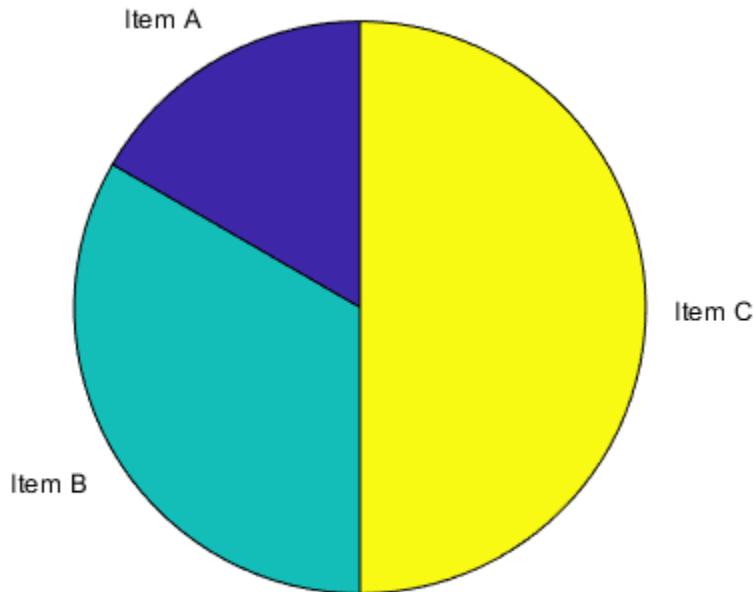
为饼图添加文本和百分比标签

创建饼图时，MATLAB 会用各个扇区在整个饼图中所占的百分比来标记每个扇区。您可以更改标签以显示不同的文本。

简单的文本标签

创建带有简单文本标签的饼图。

```
x = [1,2,3];
pie(x,['Item A','Item B','Item C'])
```



带百分比和文本的标签

创建带有标签的饼图，标签上包含每个扇区的自定义文本和预先计算的百分比值。

创建饼图并指定输出参数 `p`，以包含由 `pie` 函数创建的文本和补片对象。`pie` 函数为每个饼图扇区创建一个文本对象和一个补片对象。

```
x = [1,2,3];
p = pie(x);
```

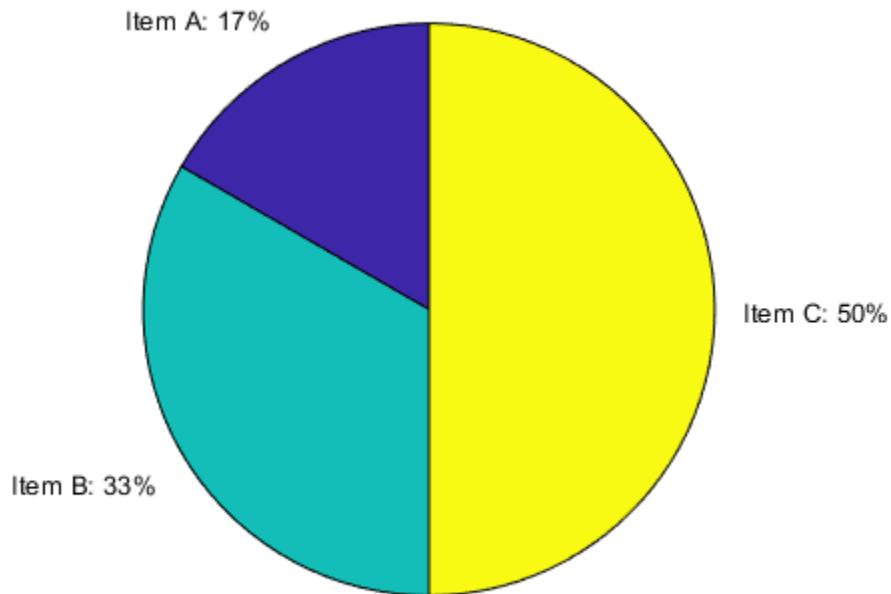
从文本对象的 `String` 属性获取每个饼图扇区的占比百分比值。然后，在元胞数组 `txt` 中指定所需文本。将此文本与元胞数组 `combinedtxt` 中的相应百分比值串联起来。

```
pText = findobj(p,'Type','text');
percentValues = get(pText,'String');
```

```
txt = {'Item A: ','Item B: ','Item C: '};  
combinedtxt = strcat(txt,percentValues);
```

通过将文本对象的 `String` 属性设置为 `combinedtxt` 来更改标签。

```
pText(1).String = combinedtxt(1);  
pText(2).String = combinedtxt(2);  
pText(3).String = combinedtxt(3);
```



另请参阅

`pie` | `findobj` | `cell2mat`

相关示例

- “向饼图添加图例” (第 2-24 页)

使用二元直方图进行颜色分析

此示例说明如何调整二元直方图的色阶，以显示与 bin 有关的更多详细信息。

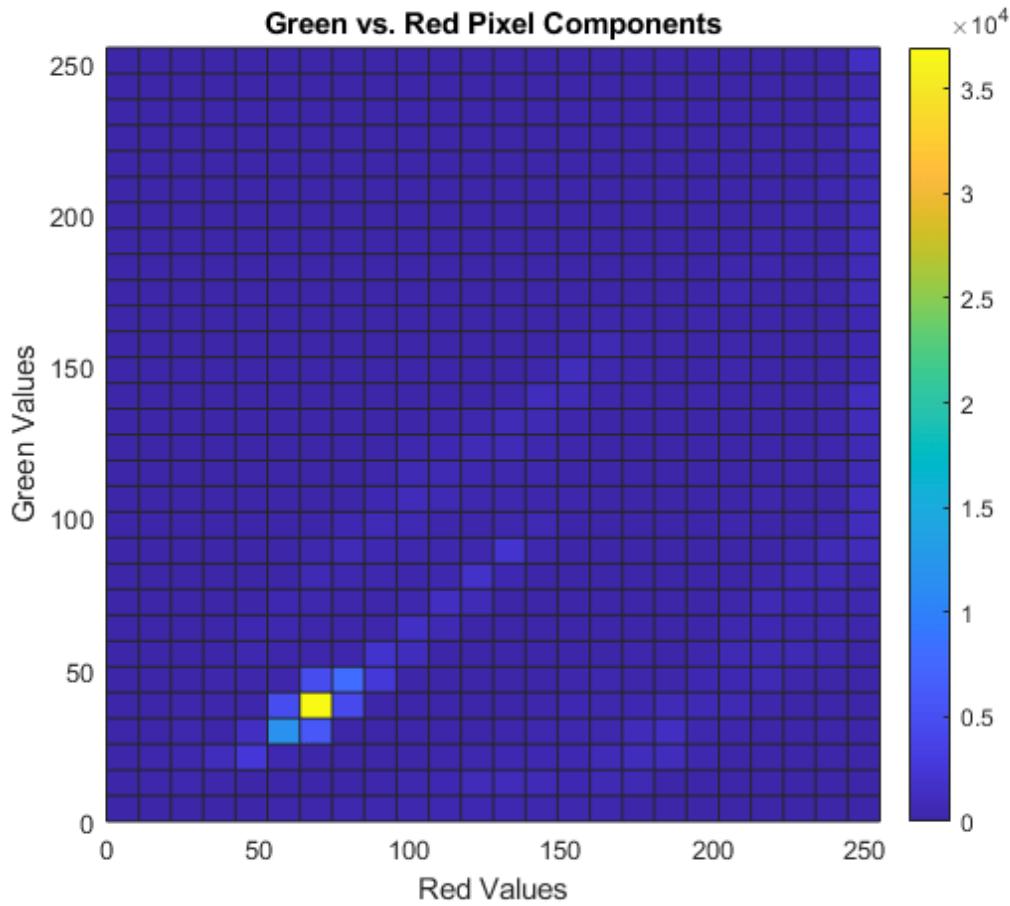
加载图像 `peppers.png`，这张彩色照片显示了几种辣椒和其他一些蔬菜。8 位无符号整数数组 `rgb` 包含图像数据。

```
rgb = imread('peppers.png');
imshow(rgb)
```



为每个像素的红、绿 RGB 值绘制二元直方图，以可视化形式呈现颜色分布。

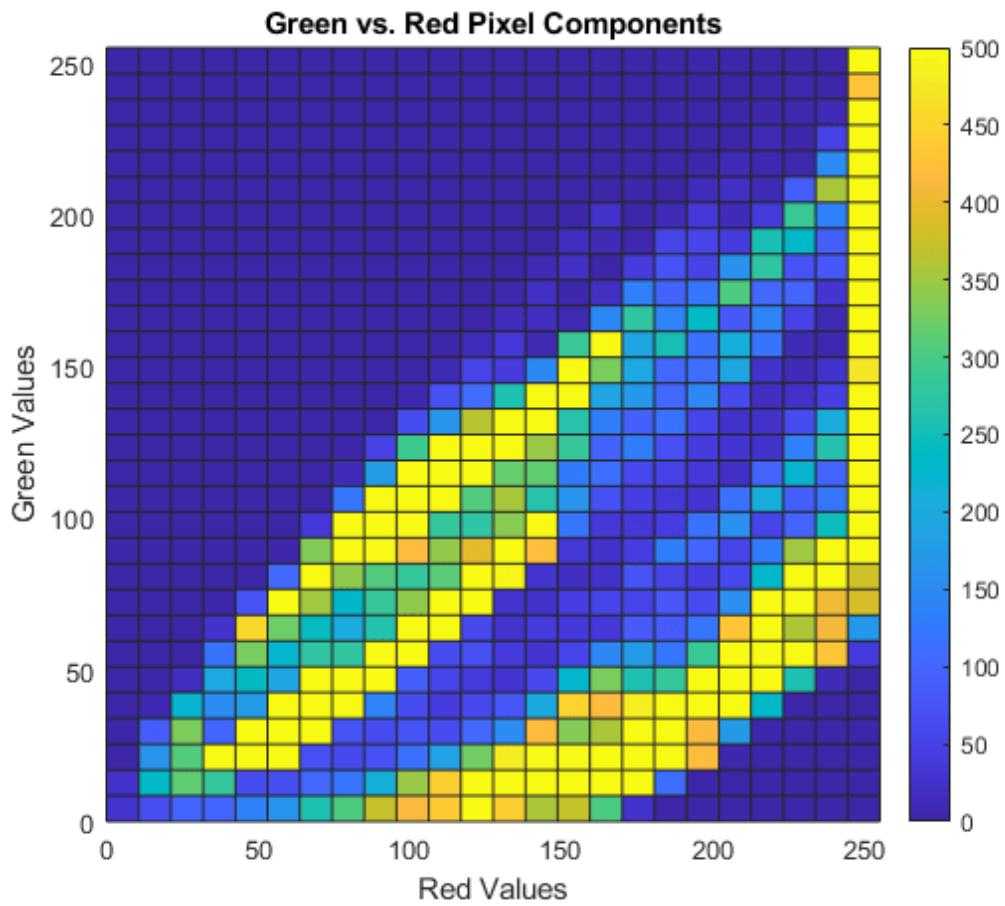
```
r = rgb(:,:,1);
g = rgb(:,:,2);
b = rgb(:,:,3);
histogram2(r,g,'DisplayStyle','tile','ShowEmptyBins','on',...
    'XBinLimits',[0 255],'YBinLimits',[0 255]);
axis equal
colorbar
xlabel('Red Values')
ylabel('Green Values')
title('Green vs. Red Pixel Components')
```



此直方图明显趋向于色阶底部，原因是有些 bin 的计数很大。这导致大部分 bin 在颜色图中显示为第一种颜色，即蓝色。如果没有提供更多详细信息，很难得出关于哪种颜色更占主导性的结论。

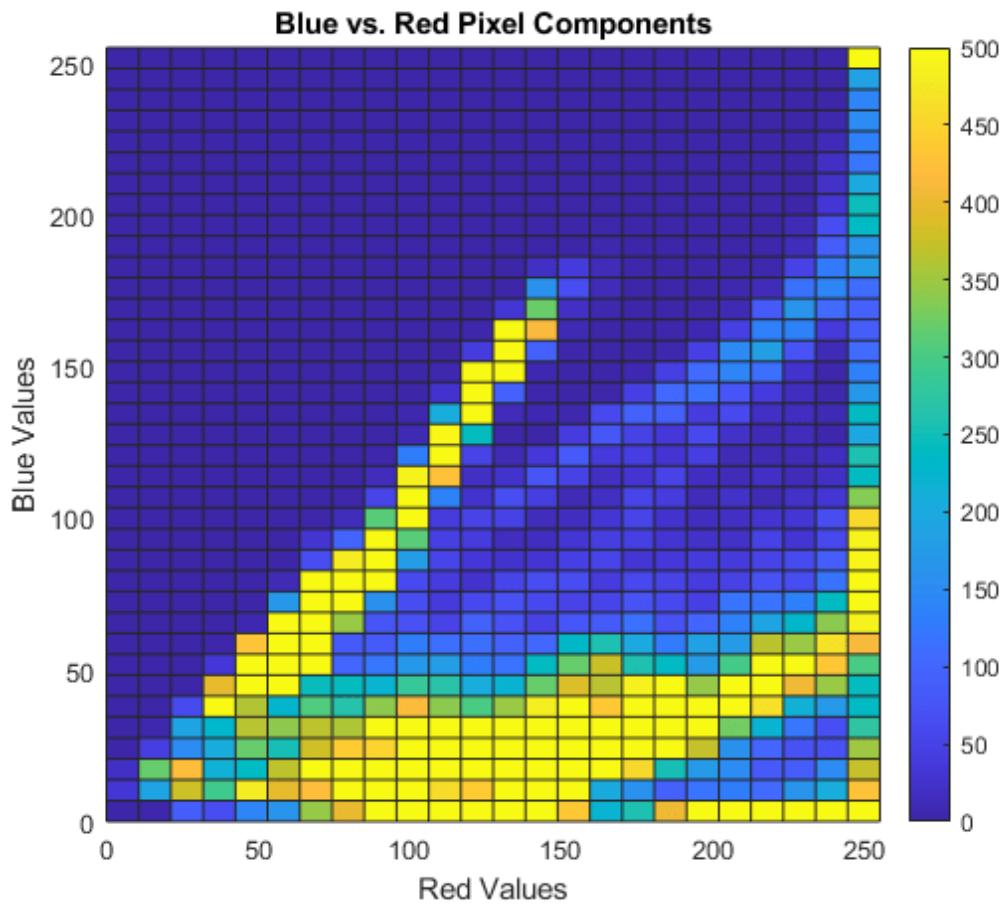
要查看更多详细信息，请通过将坐标区的 CLim 属性设置为介于 0 和 500 之间的范围，重新调整直方图的色阶。其结果就是直方图中计数达到 500 或以上的 bin 在颜色图中显示为最后一种颜色，即黄色。由于大部分 bin 的计数在这个较小范围内，因此所显示的 bin 的颜色变化较大。

```
ax = gca;
ax.CLim = [0 500];
```

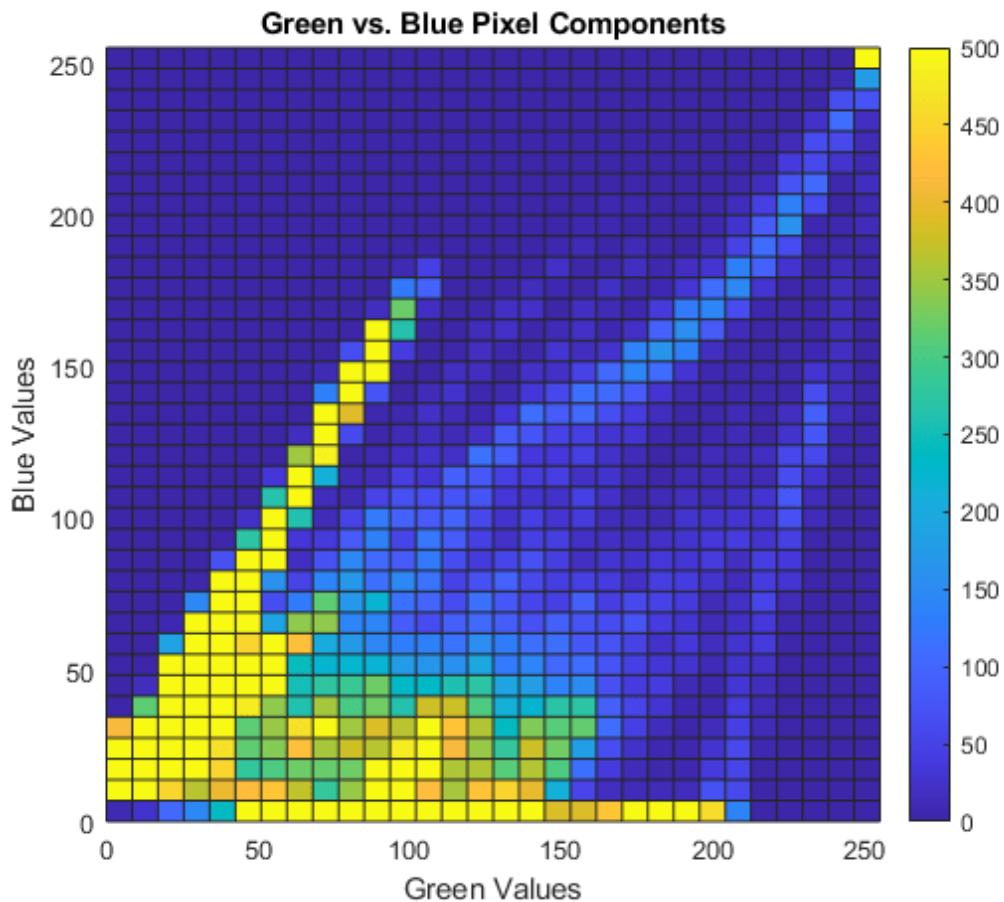


使用类似的方法可以比较红色与蓝色以及绿色与蓝色的主导性。

```
histogram2(r,b,'DisplayStyle','tile','ShowEmptyBins','on',...
    'XBinLimits',[0 255],'YBinLimits',[0 255]);
axis equal
colorbar
xlabel('Red Values')
ylabel('Blue Values')
title('Blue vs. Red Pixel Components')
ax = gca;
ax.CLim = [0 500];
```



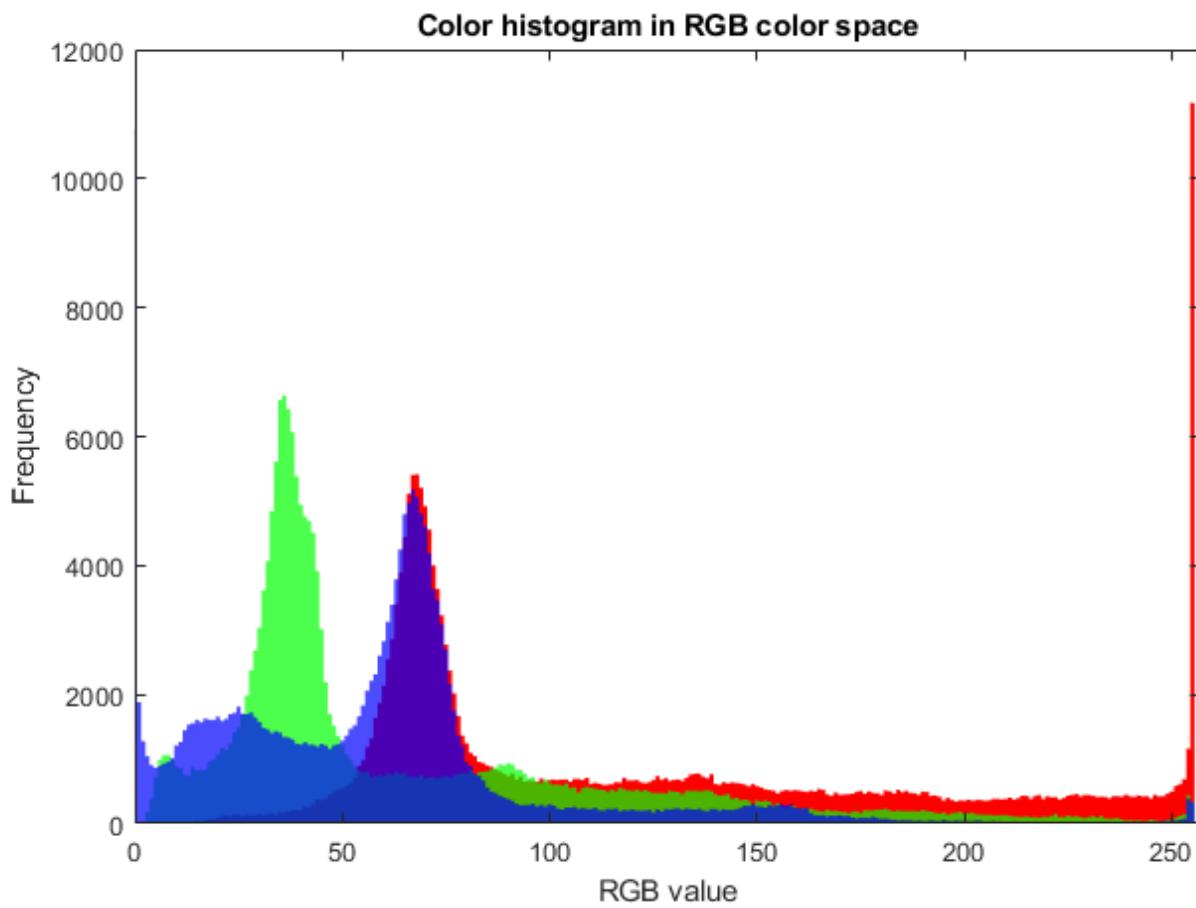
```
histogram2(g,b,'DisplayStyle','tile','ShowEmptyBins','on',...
    'XBinLimits',[0 255],'YBinLimits',[0 255]);
axis equal
colorbar
xlabel('Green Values')
ylabel('Blue Values')
title('Green vs. Blue Pixel Components')
ax = gca;
ax.CLim = [0 500];
```



在每种情况下，蓝色都是最不占主导性的颜色信号。看看所有这三个直方图，红色似乎为主导颜色。

在 RGB 颜色空间中创建一个颜色直方图，对结果进行确认。对于所有这三个颜色分量，较小的 RGB 值都有峰值。但相比其他任何分量，100 以上的值更多出现在红色分量中。

```
histogram(r,'BinMethod','integers','FaceColor','r','EdgeAlpha',0,'FaceAlpha',1)
hold on
histogram(g,'BinMethod','integers','FaceColor','g','EdgeAlpha',0,'FaceAlpha',0.7)
histogram(b,'BinMethod','integers','FaceColor','b','EdgeAlpha',0,'FaceAlpha',0.7)
xlabel('RGB value')
ylabel('Frequency')
title('Color histogram in RGB color space')
xlim([0 257])
```



另请参阅

`histogram | histogram2`

控制分类直方图的显示

此示例说明如何使用 `histogram` 有效查看分类数据。可以使用名称-值对组 'NumDisplayBins'、'DisplayOrder' 和 'ShowOthers' 更改分类直方图的显示。这些选项有助于您更好地整理数据和减少绘图中的噪点。

创建分类直方图

示例文件 `outages.csv` 包含表示美国电力中断情况的数据。该文件包含六个列：**Region**、**OutageTime**、**Loss**、**Customers**、**RestorationTime** 和 **Cause**。

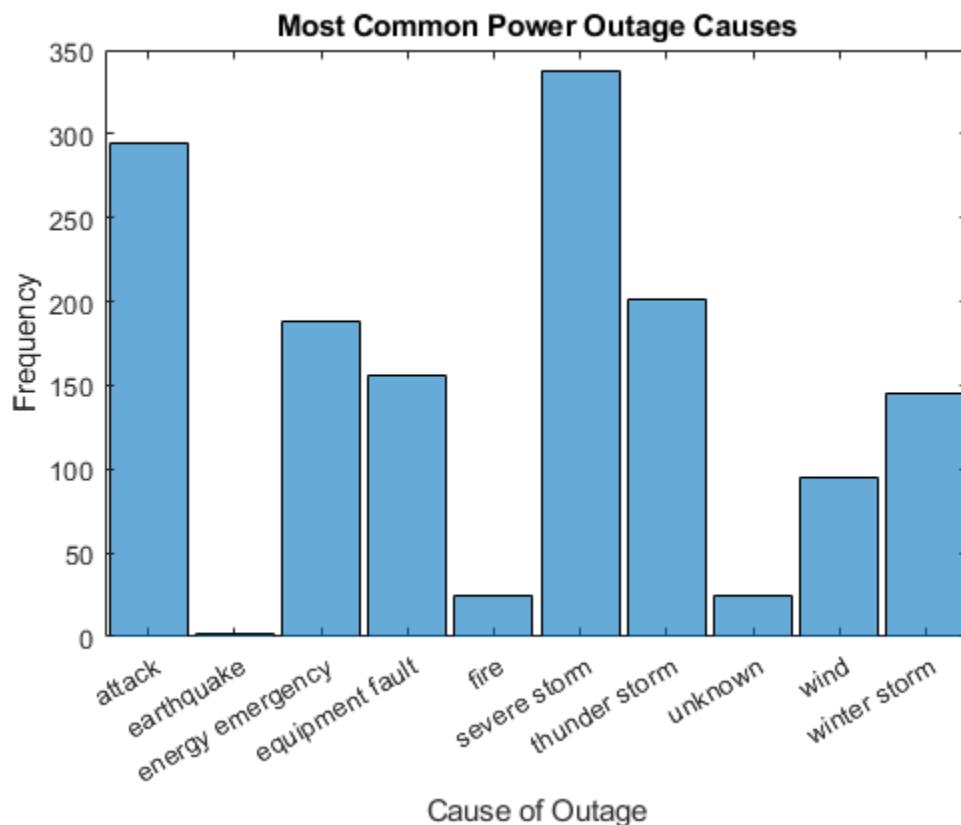
以表的形式读取 `outages.csv` 文件。使用 '`Format`' 选项指定每列包含的数据类型：分类 ('%C')、浮点数 ('%f') 或日期时间 ('%D')。对数据的前几行进行索引以查看变量。

```
data_formats = '%C%D%f%D%C';
C = readtable('outages.csv','Format',data_formats);
first_few_rows = C(1:10,:)

first_few_rows=10×6 table
    Region        OutageTime      Loss    Customers   RestorationTime       Cause
    _____    _____
    SouthWest  2002-02-01 12:18  458.98  1.8202e+06  2002-02-07 16:50  winter storm
    SouthEast  2003-01-23 00:49  530.14  2.1204e+05      NaT  winter storm
    SouthEast  2003-02-07 21:15  289.4   1.4294e+05  2003-02-17 08:14  winter storm
    West      2004-04-06 05:44  434.81  3.4037e+05  2004-04-06 06:10  equipment fault
    MidWest   2002-03-16 06:18  186.44  2.1275e+05  2002-03-18 23:23  severe storm
    West      2003-06-18 02:49      0      0  2003-06-18 10:54  attack
    West      2004-06-20 14:39  231.29      NaN  2004-06-20 19:16  equipment fault
    West      2002-06-06 19:28  311.86      NaN  2002-06-07 00:51  equipment fault
    NorthEast 2003-07-16 16:23  239.93  49434  2003-07-17 01:12  fire
    MidWest   2004-09-27 11:09  286.72  66104  2004-09-27 16:37  equipment fault
```

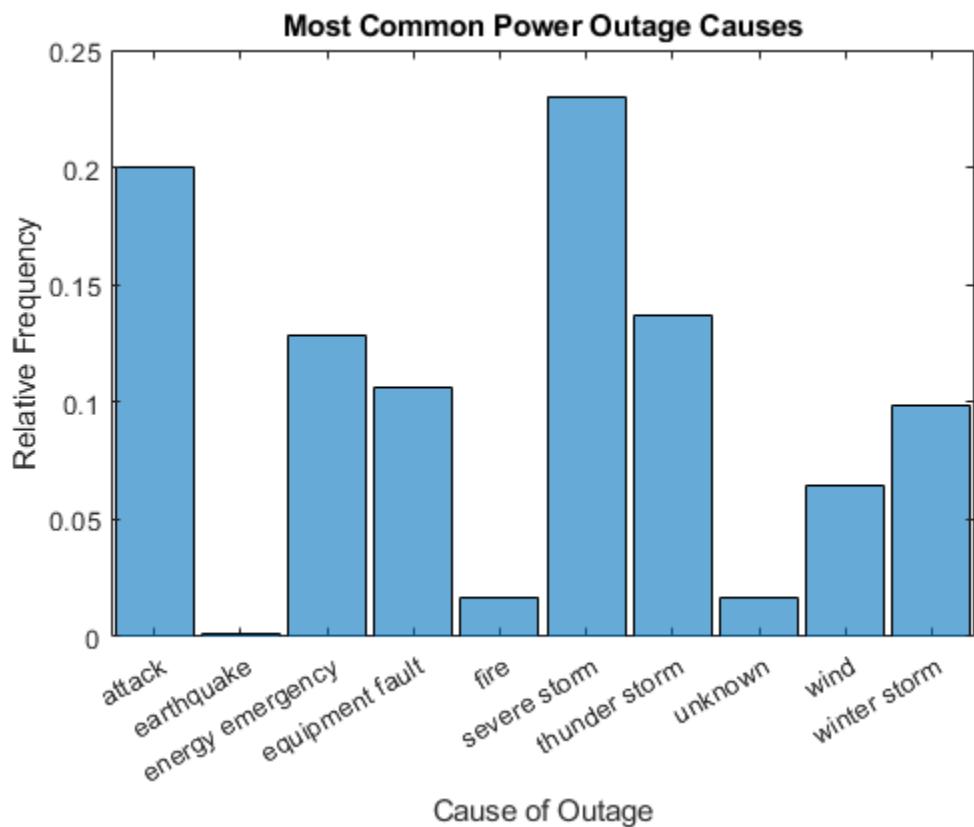
绘制 **Cause** 变量的分类直方图。指定一个输出参数以返回直方图对象的句柄。

```
h = histogram(C.Cause);
xlabel('Cause of Outage')
ylabel('Frequency')
title('Most Common Power Outage Causes')
```



将直方图的归一化方式更改为使用 'probability' 归一化，它显示每种断电原因的相对频率。

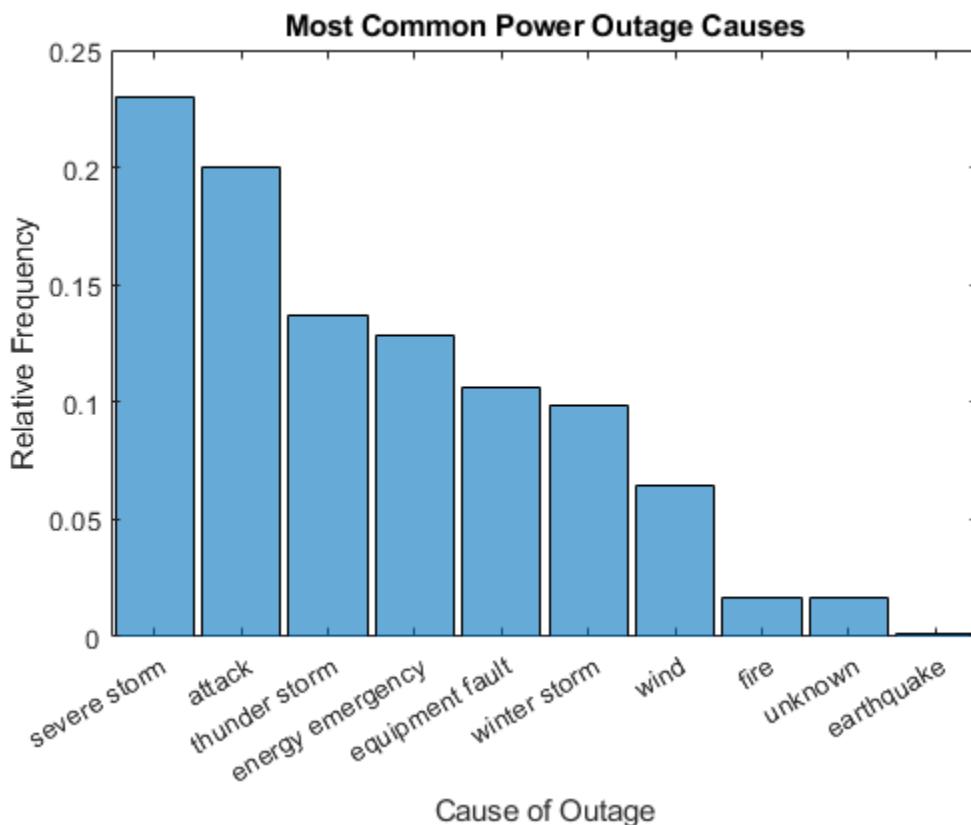
```
h.Normalization = 'probability';
ylabel('Relative Frequency')
```



更改显示顺序

使用 'DisplayOrder' 选项按从大到小的顺序对 bin 排序。

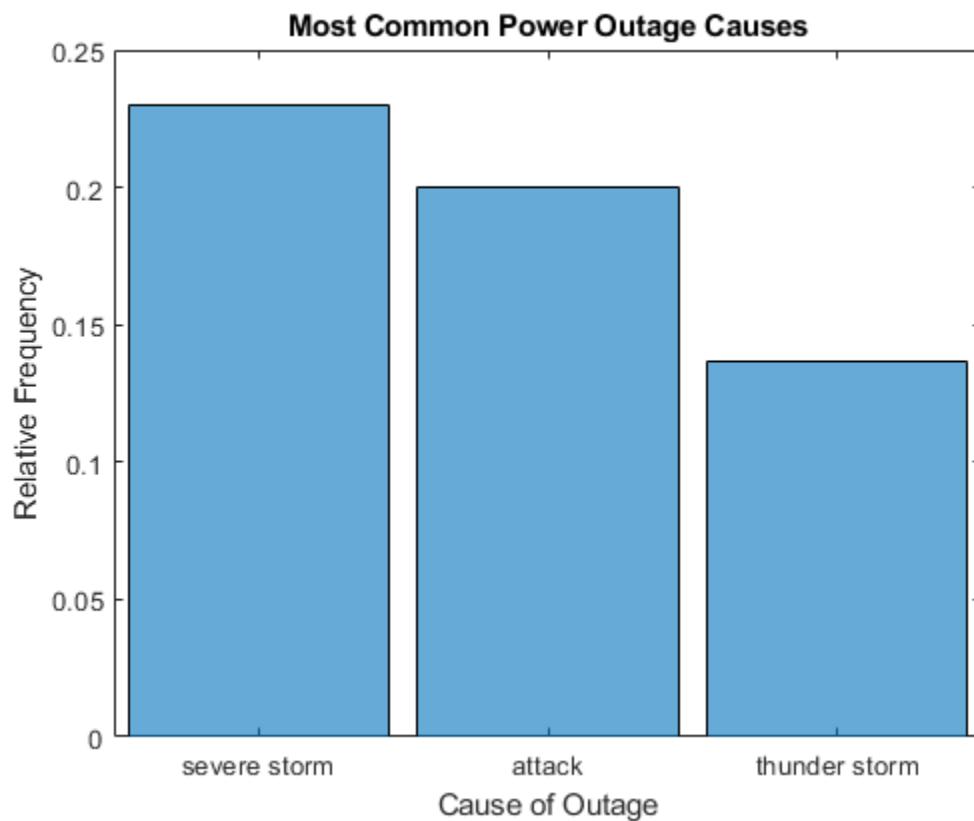
```
h.DisplayOrder = 'descend';
```



截断显示的条形数

使用 'NumDisplayBins' 选项仅在绘图中显示三个条形。由于对未显示的数据仍考虑了归一化，因此所显示的概率相加不再为 1。

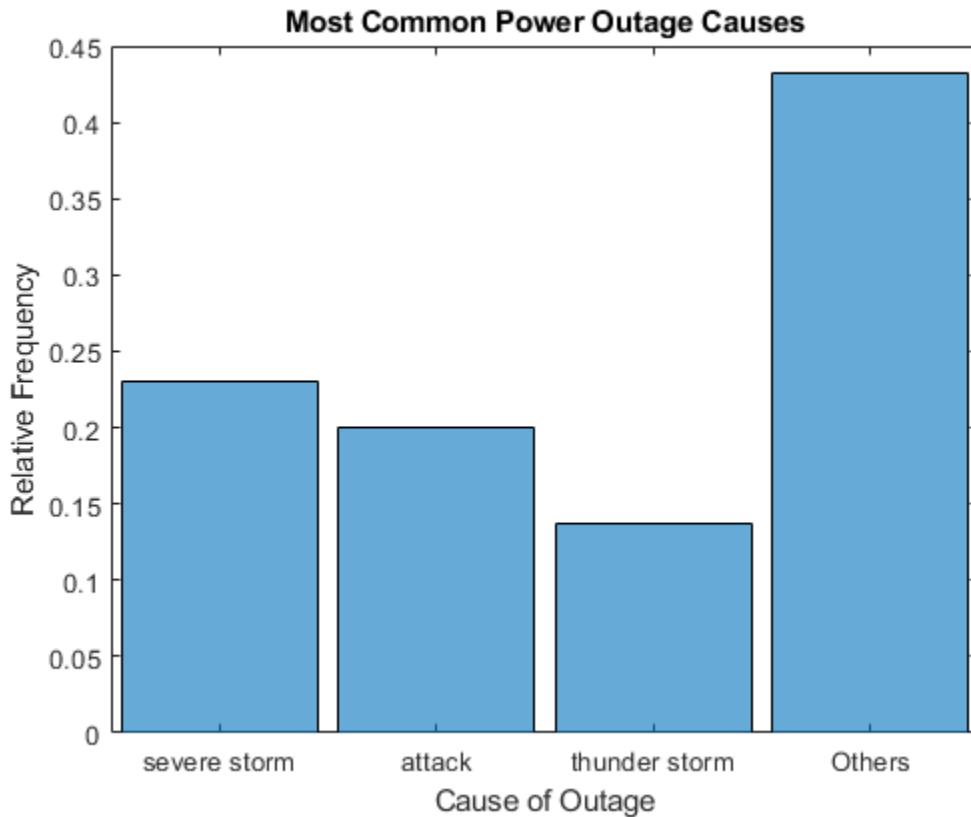
```
h.NumDisplayBins = 3;
```



将被排除的数据相加

使用 'ShowOthers' 选项合并显示所有被排除的条形，以使所显示的概率相加后再次等于 1。

```
h>ShowOthers = 'on';
```



限制只对显示数据进行归一化

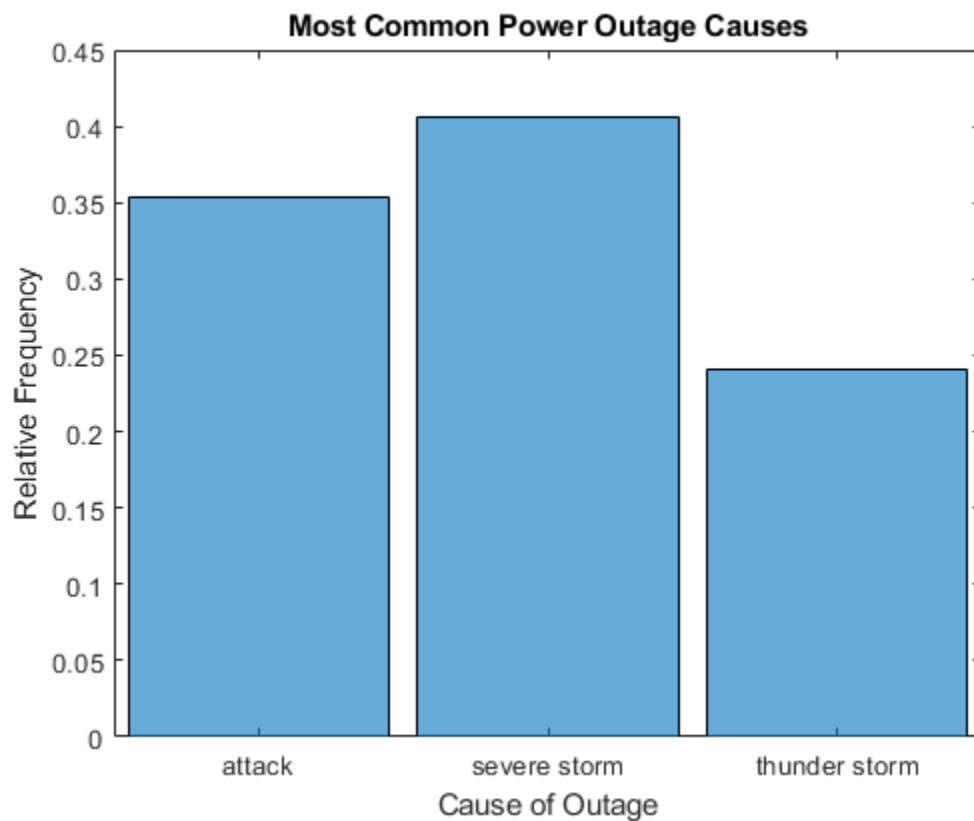
在 R2017a 之前的版本中，`histogram` 和 `histcounts` 函数仅使用经过 bin 处理的数据来计算归一化。这种行为意味着如果部分数据最终位于 bin 外面，则在归一化时会被忽略。但在 MATLAB® R2017a 中，该行为改为始终使用输入数据中的总元素数目来进行归一化。这种新行为更为直观，但如果更愿意采用旧行为，则需要采取几个特殊步骤来限制仅对经过分 bin 处理的数据进行归一化。

您可以限制只对直方图中显示的数据实施概率归一化，而不是对所有输入数据均进行归一化。只需更新直方图对象的 `Data` 属性，删除其他类别即可。`Categories` 属性将反映直方图中显示的类别。使用 `setdiff` 比较这两个属性值，并从 `Data` 中删除不在 `Categories` 中的所有类别。然后从数据中删除得到的所有 `undefined` 分类元素，仅保留所显示类别中的元素。

```

h>ShowOthers = 'off';
cats_to_remove = setdiff(categories(h.Data),h.Categories);
h.Data = removecats(h.Data,cats_to_remove);
h.Data = rmmissing(h.Data);

```



归一化处理现在仅基于剩余的三个类别，因此这三个条形相加等于 1。

另请参阅

[histogram](#) | [categorical](#) | [histogram](#)

替换不建议使用的 hist 和 histc 实例

本节内容

“旧直方图函数 (hist 和 histc) ” (第 2-41 页)

“推荐的直方图函数” (第 2-41 页)

“要求代码更新的差异” (第 2-41 页)

旧直方图函数 (hist 和 histc)

MATLAB 早期版本使用 `hist` 和 `histc` 函数作为创建直方图和计算 bin 计数的主要方法。这些函数适用于某些常规用途，但总体能力有限。基于这些原因（及其他原因），不建议在新代码中使用 `hist` 和 `histc`:

- 使用 `hist` 创建直方图后，修改该直方图的属性会有难度并要求重新计算整个直方图。
- `hist` 的默认行为是使用 10 个 bin，这对很多数据集都不适用。
- 绘制归一化直方图需要手动计算。
- `hist` 和 `histc` 的行为并不一致。

推荐的直方图函数

`histogram`、`histcounts` 和 `discretize` 函数显著提高了 MATLAB 中创建和计算直方图的能力，同时提升了一致性和易用性。要为新代码创建和计算直方图，推荐使用 `histogram`、`histcounts` 和 `discretize` 函数。

请特别注意以下更改，它们体现了相对于 `hist` 和 `histc` 的改进:

- `histogram` 可以返回一个直方图对象。您可以使用该对象修改直方图的属性。
- `histogram` 和 `histcounts` 都具有自动分 bin 和归一化功能，内置了几个常用选项。
- `histcounts` 是 `histogram` 的主要计算函数。这样各函数就具有一致的行为。
- `discretize` 为确定每个元素的 bin 位置提供了其他选择和灵活性。

要求代码更新的差异

尽管做出了上述改进，但旧函数和当前建议的函数之间存在一些重要差异，这可能需要更新您的代码。下表汇总了这些函数之间的差异并提供了代码更新建议。

hist 的代码更新

差异	使用 hist 时的旧行为	使用 histogram 时的新行为
输入矩阵	<p>hist 为输入矩阵的每一列创建直方图并在同一图窗中并排绘制直方图。</p> <pre>A = randn(100,2); hist(A)</pre>	<p>histogram 将整个输入矩阵视为一个高向量并创建一个直方图。要绘制多个直方图，请为每列数据创建一个不同的直方图对象。使用 hold on 命令在同一图窗中绘制直方图。</p> <pre>A = randn(100,2); h1 = histogram(A(:,1),10) edges = h1.BinEdges; hold on h2 = histogram(A(:,2),edges)</pre> <p>以上代码示例对每个直方图使用相同的 bin 边界，但在某些情况下，最好将每个直方图的 BinWidth 设置为相同。此外，出于显示目的，设置每个直方图的 FaceAlpha 属性可能会有所帮助，因为这会影响重叠条形的透明度。</p>
Bin 设定	<p>hist 接受 bin 中心作为另一个输入项。</p>	<p>histogram 接受 bin 边界作为另一个输入项。</p> <p>要将 bin 中心转换为 bin 边界以用于 histogram，请参阅“将 bin 中心转换为 bin 边”（第 2-45 页）。</p> <p>注意 若用于 hist 的 bin 中心是整数，例如 <code>hist(A,-3:3)</code>，请使用 histogram 内置的用于整数的新分 bin 方法。</p> <pre>histogram(A,'BinLimits',[-3,3],'BinMethod','in</pre>

差异	使用 hist 时的旧行为	使用 histogram 时的新行为
输出参数	<p>hist 返回 bin 计数作为一个输出参数，也可以选择返回 bin 中心作为另一个输出参数。</p> <pre>A = randn(100,1); [N, Centers] = hist(A)</pre>	<p>histogram 返回一个直方图对象作为输出参数。该对象包含许多相关属性（bin 计数、bin 边界等）。可通过更改直方图的属性值修改它的各个方面。有关详细信息，请参阅 histogram。</p> <pre>A = randn(100,1); h = histogram(A); N = h.Values Edges = h.BinEdges</pre> <p>注意 要计算 bin 计数（而不绘制直方图），请将 <code>[N, Centers] = hist(A)</code> 替换为 <code>[N,edges] = histcounts(A,nbins)</code>。</p>
默认 bin 数量	默认情况下，hist 使用 10 个 bin。	<p>默认情况下，histogram 和 histcounts 都使用自动分 bin 算法。bin 数量由输入数据的数量和范围程度确定。</p> <pre>A = randn(100,1); histogram(A) histcounts(A)</pre>
bin 范围	hist 使用最小和最大有限数据值来确定绘图中第一个和最后一个条形的左边缘与右边缘。 <code>-Inf</code> 和 <code>Inf</code> 分别包含在第一个和最后一个 bin 中。	<p>如果未设置 BinLimits，则 histogram 会基于（但不完全等于）最小和最大的有限数据值来确定 bin 的有理数范围。histogram 将会忽略 Inf 值，除非其中一个 bin 边界将 Inf 或 <code>-Inf</code> 显式指定为 bin 边界。</p> <p>要重新生成 <code>hist(A)</code> 对有限数据（无 Inf 值）的结果，请使用 10 个 bin 并将 BinLimits 显式设置为最小和最大数据值。</p> <pre>A = randi(5,100,1); histogram(A,10,'BinLimits',[min(A) max(A)])</pre>

histc 的代码更新

差异	使用 histc 时的旧行为	使用 histcounts 时的新行为
输入矩阵	<p>histc 为输入数据的每列计算 bin 计数。对于 $m \times n$ 输入矩阵，histc 返回大小为 $\text{length(edges)} \times n$ 的 bin 计数矩阵。</p> <pre>A = randn(100,10); edges = -4:4; N = histc(A,edges)</pre>	<p>histcounts 将整个输入矩阵视为一个高向量并计算整个矩阵的 bin 计数。</p> <pre>A = randn(100,10); edges = -4:4; N = histcounts(A,edges)</pre> <p>对每列使用 for 循环计算 bin 计数。</p> <pre>A = randn(100,10); nbins = 10; N = zeros(nbins, size(A,2)); for k = 1:size(A,2) N(:,k) = histcounts(A(:,k),nbins); end</pre> <p>如果由于矩阵中的列数很多而造成性能问题，请考虑继续使用 histc 计算各列的 bin 计数。</p>
最后一个 bin 中包含的值	<p>如果 $A(i) == \text{edges}(\text{end})$，histc 会在最后一个 bin 中包含元素 $A(i)$。输出 N 是一个具有 length(edges) 个元素的向量，其中包含 bin 计数。落入 bin 范围外的值不会计算在内。</p>	<p>如果 $\text{edges}(\text{end}-1) <= A(i) <= \text{edges}(\text{end})$，histcounts 会在最后一个 bin 中包含元素 $A(i)$。换言之，histcounts 将来自 histc 的最后两个 bin 合并为最后一个 bin。输出 N 是一个具有 $\text{length(edges)}-1$ 个元素的向量，其中包含 bin 计数。如果指定了 bin 边界，则落入 bin 外部的值不会计算在内。否则，histcounts 会自动确定要使用的适当 bin 边界以包含所有数据。</p> <pre>A = 1:4; edges = [1 2 2.5 3] N = histcounts(A) N = histcounts(A,edges)</pre> <p>histc 中最后的 bin 主要用于整数计数。要使用 histcounts 执行这种整数计数，可以使用 'integers' bin 方法：</p> <pre>N = histcounts(A,'BinMethod','integers');</pre>

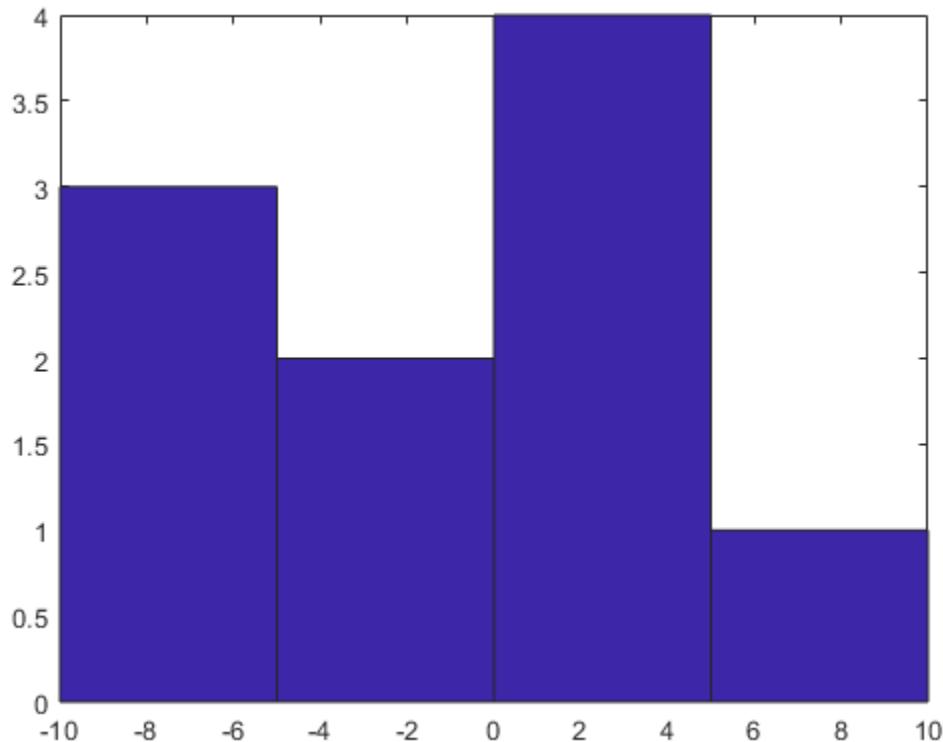
差异	使用 histc 时的旧行为	使用 histcounts 时的新行为
输出参数	<p>histc 返回 bin 计数作为一个输出参数，也可以选择返回 bin 索引作为另一个输出参数。</p> <pre>A = randn(15,1); edges = -4:4; [N,Bin] = histc(A,edges)</pre>	<ul style="list-style-type: none"> 对于 $N = \text{histc}(A, \text{edges})$ 或 $[N, \text{bin}] = \text{histc}(A, \text{edges})$ 之类的 bin 计数计算，请使用 histcounts。histcounts 函数返回 bin 计数作为一个输出参数，也可以选择返回 bin 边界作为另一个输出参数，或返回 bin 索引作为第三个输出参数。 <pre>A = randn(15,1); [N,Edges,Bin] = histcounts(A)</pre> 对于 $[\sim, \text{Bin}] = \text{histc}(A, \text{edges})$ 之类的 bin 位置计算，请使用 discretize。discretize 函数提供用于确定每个元素的 bin 位置的其他选项。 <pre>A = randn(15,1); edges = -4:4; Bin = discretize(A,edges)</pre>

将 bin 中心转换为 bin 边

hist 函数接受 bin 中心，而 histogram 函数接受 bin 边界。要更新代码以使用 histogram，您可能需要将 bin 中心转换为 bin 边界，以重现使用 hist 实现的结果。

例如，指定 bin 中心以用于 hist。这些 bin 具有均匀的宽度。

```
A = [-9 -6 -5 -2 0 1 3 3 4 7];
centers = [-7.5 -2.5 2.5 7.5];
hist(A,centers)
```



要将 bin 中心转换为 bin 边界，请计算 `centers` 中各连续值之间的中点。此方法会重现均匀和非均匀 bin 宽度对应的 `hist` 结果。

```
d = diff(centers)/2;  
edges = [centers(1)-d(1), centers(1:end-1)+d, centers(end)+d(end)];
```

`hist` 函数包括落入每个 bin 右边界上的值（第一个 bin 包含两个边界），而 `histogram` 包括落入每个 bin 左边界上的值（最后一个 bin 包含两个边界）。稍微移动 bin 边界以获得与 `hist` 相同的 bin 计数。

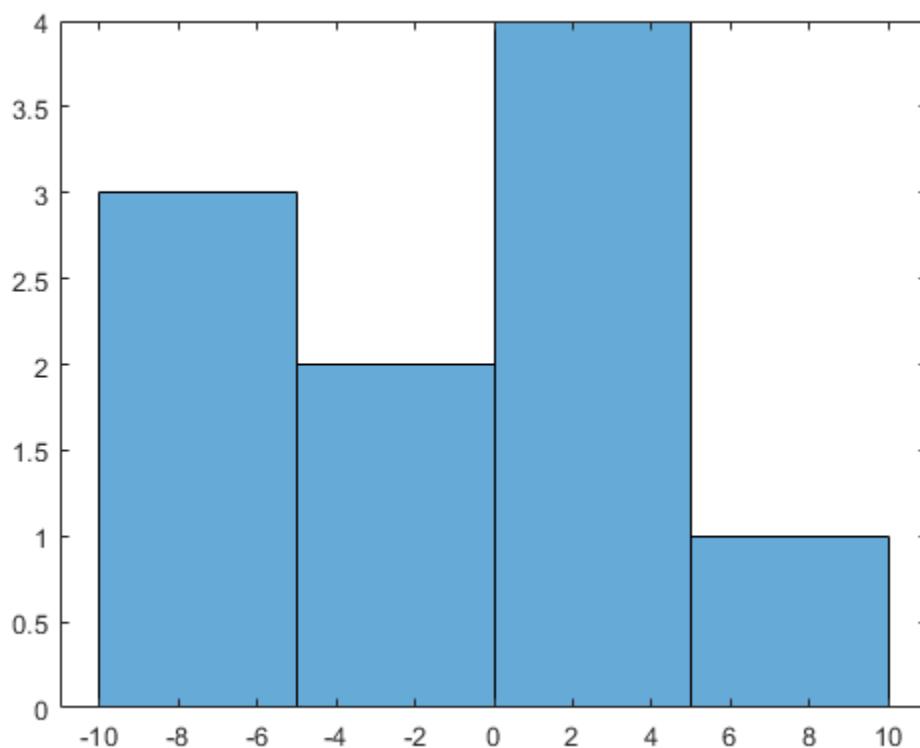
```
edges(2:end) = edges(2:end)+eps(edges(2:end))
```

```
edges = 1×5
```

```
-10.0000  -5.0000   0.0000   5.0000  10.0000
```

现在，将 `histogram` 与 bin 边界结合使用。

```
histogram(A,edges)
```



极坐标图

- “在极坐标中绘图” (第 3-2 页)
- “自定义极坐标区” (第 3-14 页)
- “极坐标区上的罗盘标签” (第 3-22 页)

在极坐标中绘图

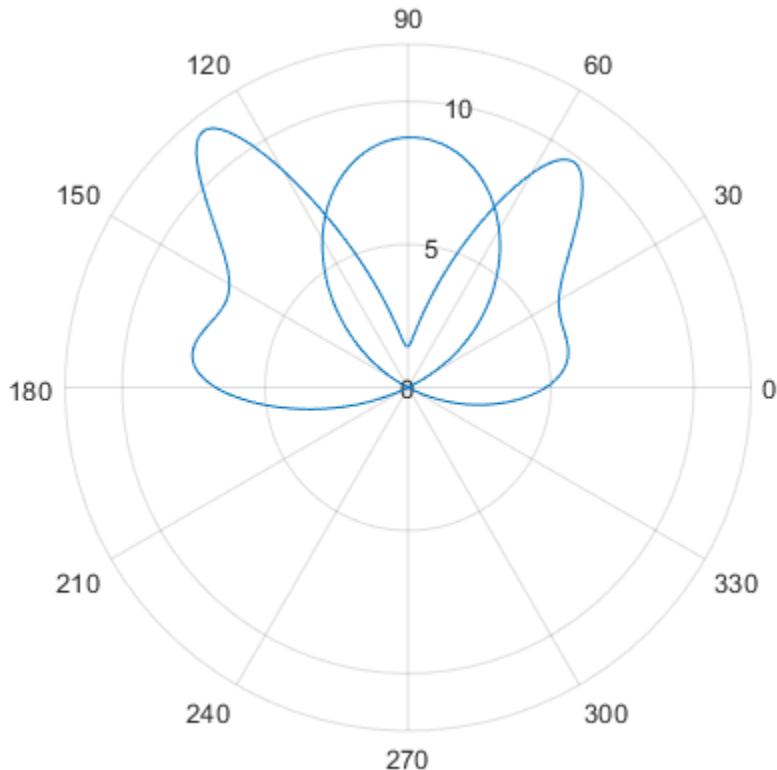
以下示例演示如何在极坐标中创建线图、散点图和直方图。此外，还演示了如何对极坐标图添加注释和更改轴范围。

创建极坐标线图

通过极坐标中的天线以可视化方式呈现辐射图。加载文件 `antennaData.mat`，该文件包含变量 `theta` 和 `rho`。变量 `rho` 用于测量天线对 `theta` 的每个值的辐射强度。通过使用 `polarplot` 函数在极坐标中绘制数据图，以可视化方式呈现该辐射图。

```
load('antennaData.mat')

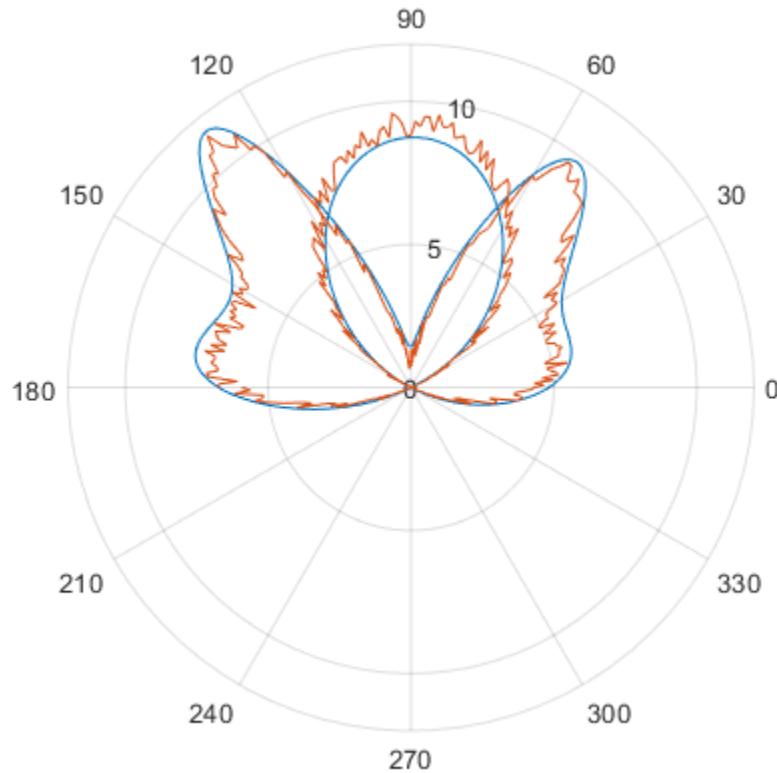
figure
polarplot(theta,rho)
```



多个极坐标线图

使用 `hold on` 保留当前极坐标区，然后通过 `polarplot` 绘制其他数据图。

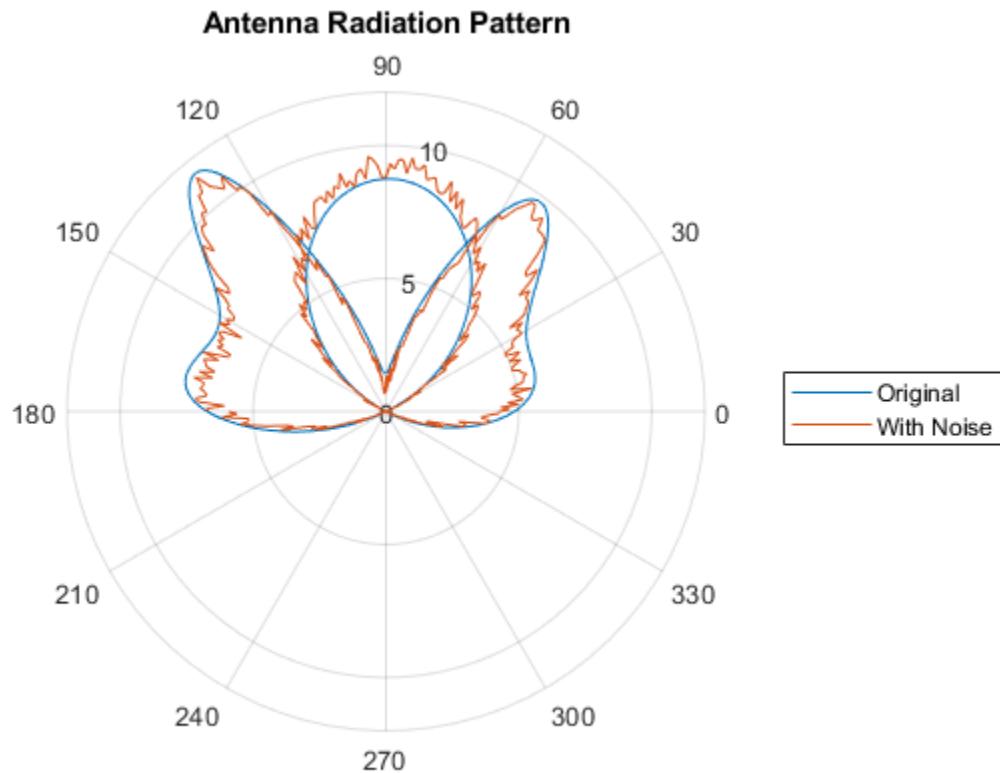
```
rng('default')
noisy = rho + rand(size(rho));
hold on
polarplot(theta,noisy)
hold off
```



为极坐标图添加注释

使用 `legend` 和 `title` 之类的注释函数，标记与其他可视化类型类似的极坐标图。

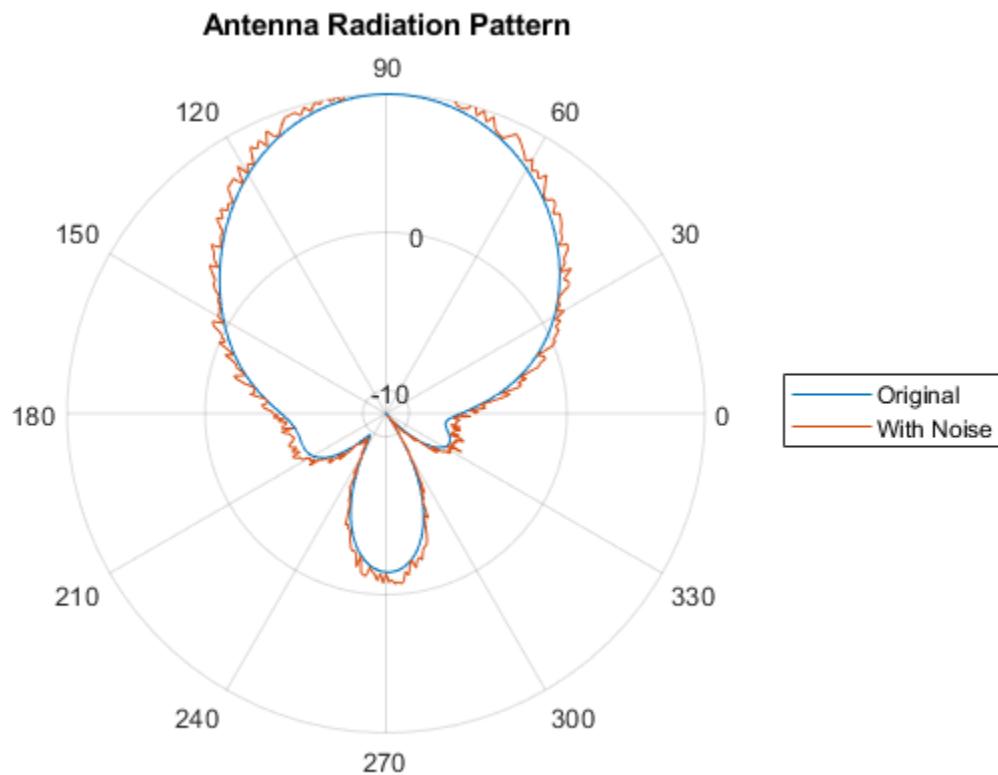
```
legend('Original','With Noise')
title('Antenna Radiation Pattern')
```



更改极坐标区范围

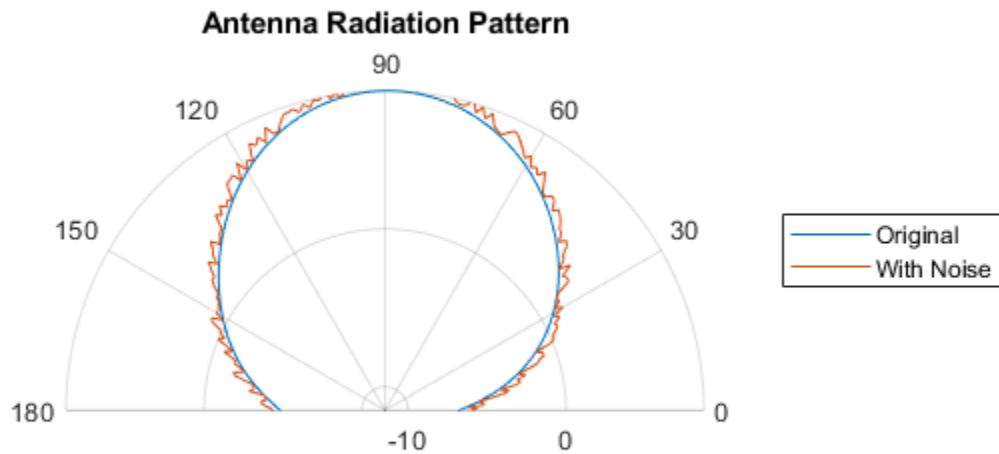
默认情况下，在极坐标图中，半径的负值将被绘制为正值。使用 `rlim` 将 r 坐标轴范围调整为包含负值。

```
rmin = min(rho);
rmax = max(rho);
rlim([rmin rmax])
```



使用 `thetalim` 将 theta 坐标轴范围更改为 0 到 180。

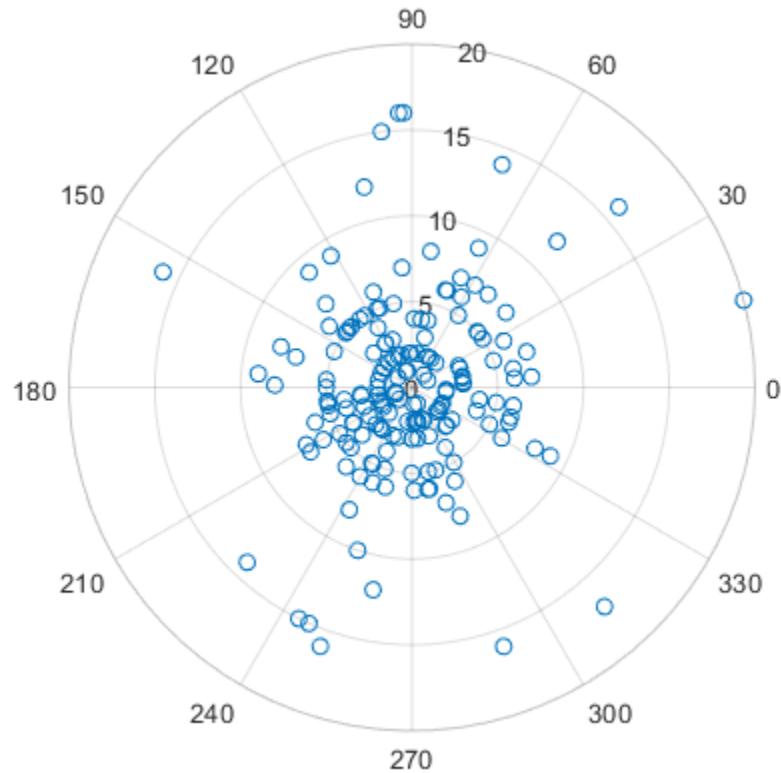
```
thetalim([0 180])
```



创建极坐标散点图

在极坐标中绘制风速数据图。加载文件 `windData.dat`, 该文件包含变量 `direction`、`speed`、`humidity` 和 `C`。通过使用 `polarscatter` 函数在极坐标中绘制数据图，以可视化方式呈现风速图。

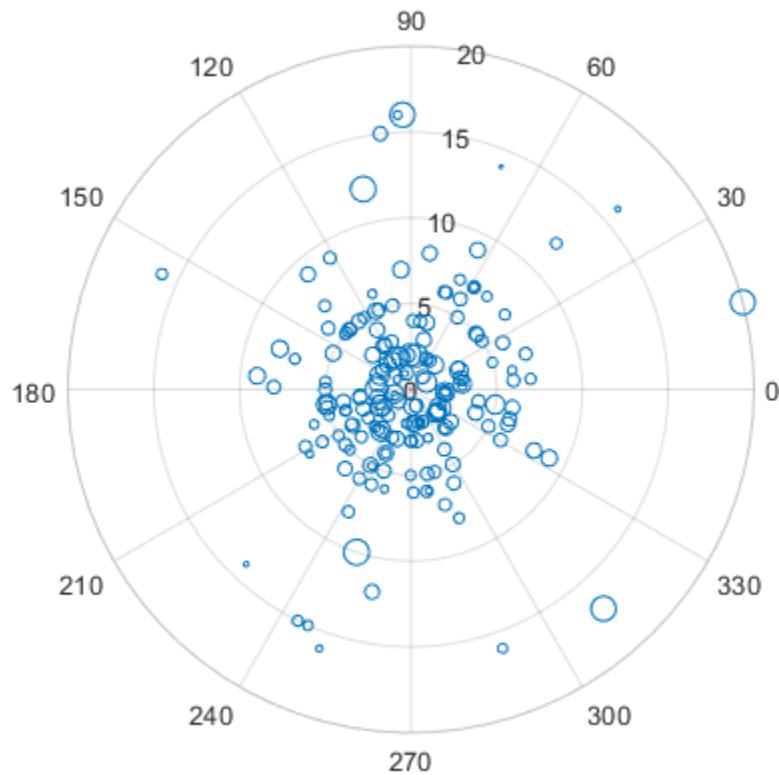
```
load('windData.mat')
polarscatter(direction,speed)
```



包括第三个数据输入以改变标记大小并表示第三个维度。

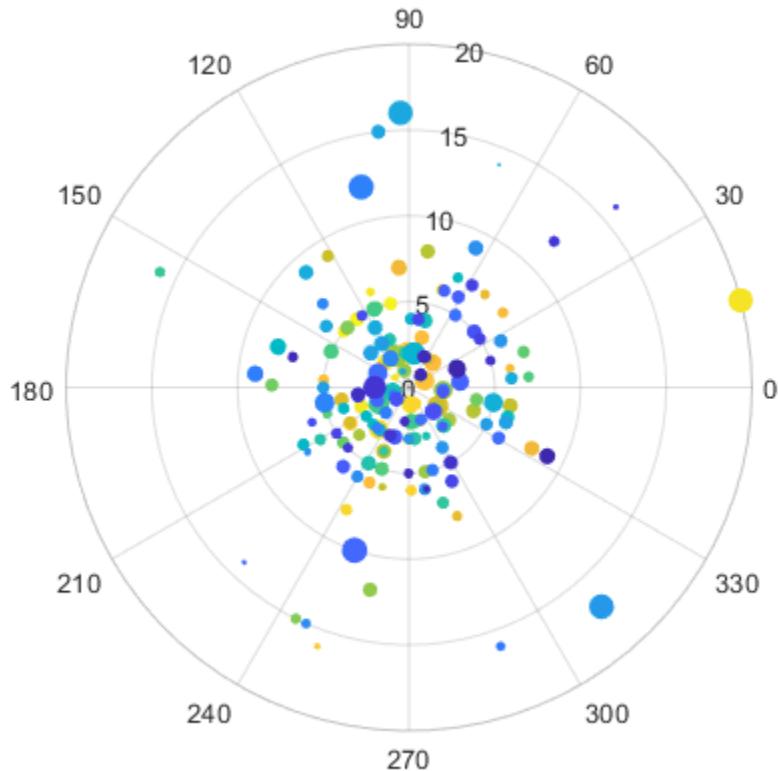
polarscatter(direction,speed,humidity)

3 极坐标图



使用格式化输入调整标记显示属性。

```
polarscatter(direction,speed,humidity,C,'filled')
```

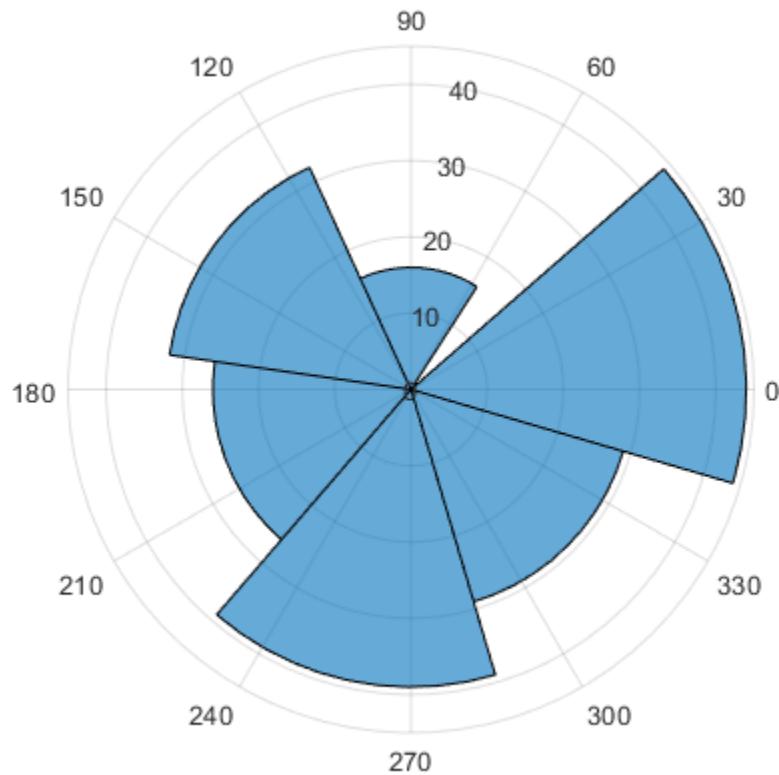


创建极坐标直方图

使用 `polarhistogram` 函数以可视化方式呈现数据，这将会生成称为风向图的可视化表示形式。

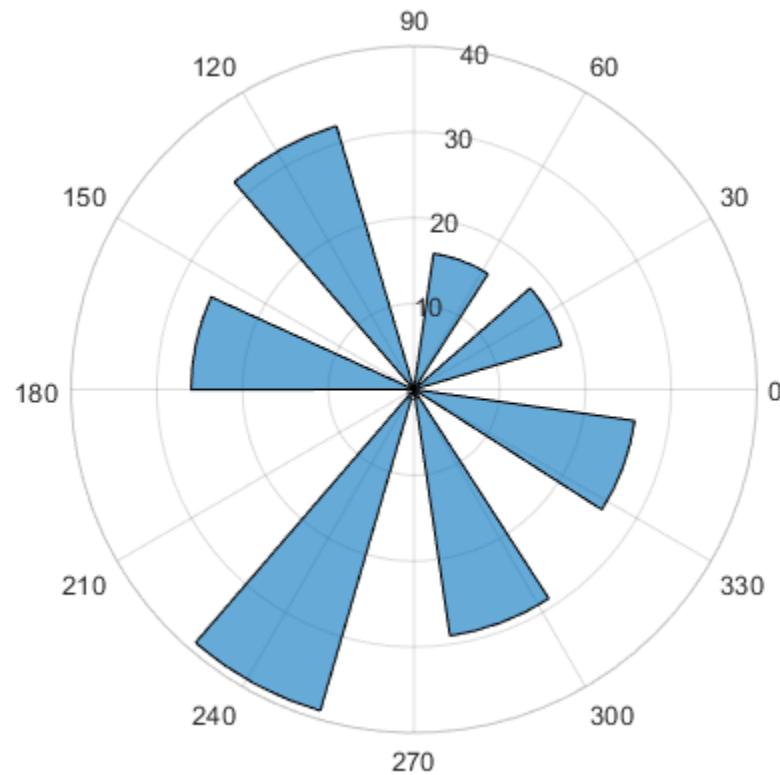
`polarhistogram(direction)`

3 极坐标图



指定 bin 确定算法。 **polarhistogram** 函数具有各种确定 bin 数量和 bin 宽度的算法，可从 **BinMethod** 字段中选择。

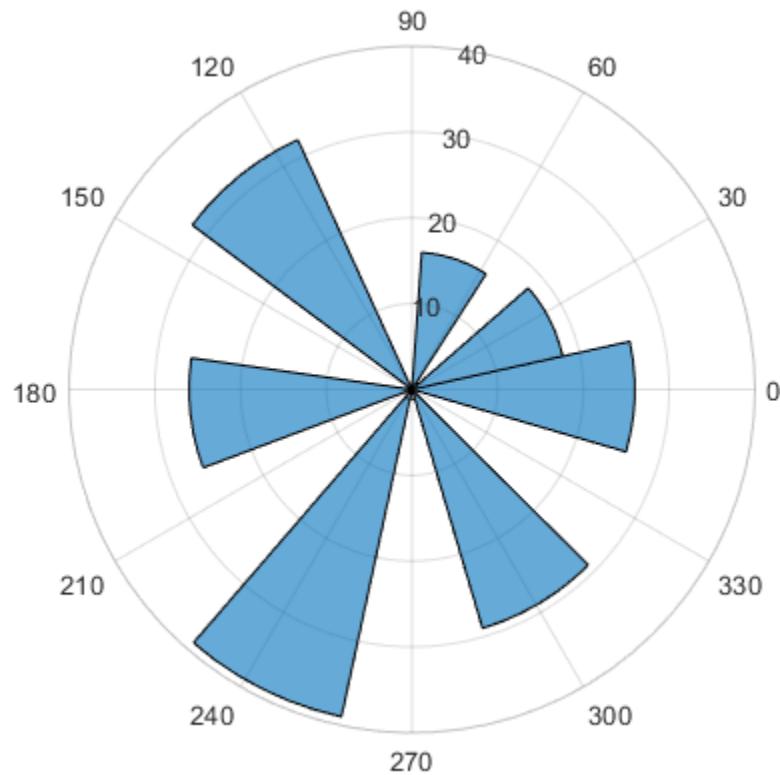
polarhistogram(direction,'BinMethod','sqrt')



指定 bin 数量和 bin 宽度。

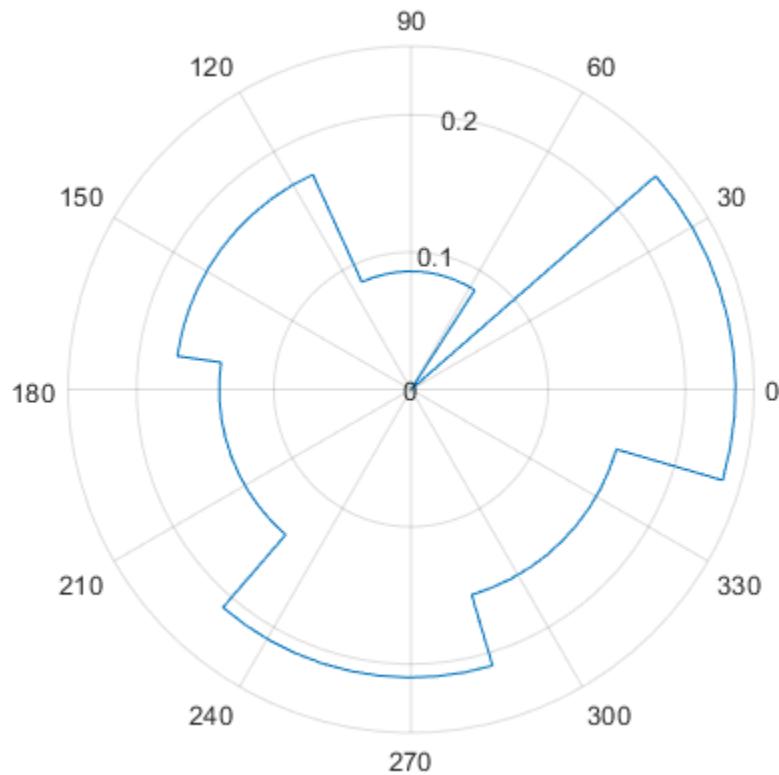
```
polarhistogram(direction,24,'BinWidth',.5)
```

3 极坐标图



指定归一化方法并调整显示样式以排除任何填充。

```
polarhistogram(direction,'Normalization','pdf','DisplayStyle','stairs')
```



另请参阅

[polarplot](#) | [thetaticks](#) | [rticks](#) | [rticklabels](#) | [thetaticklabels](#) | [PolarAxes](#)

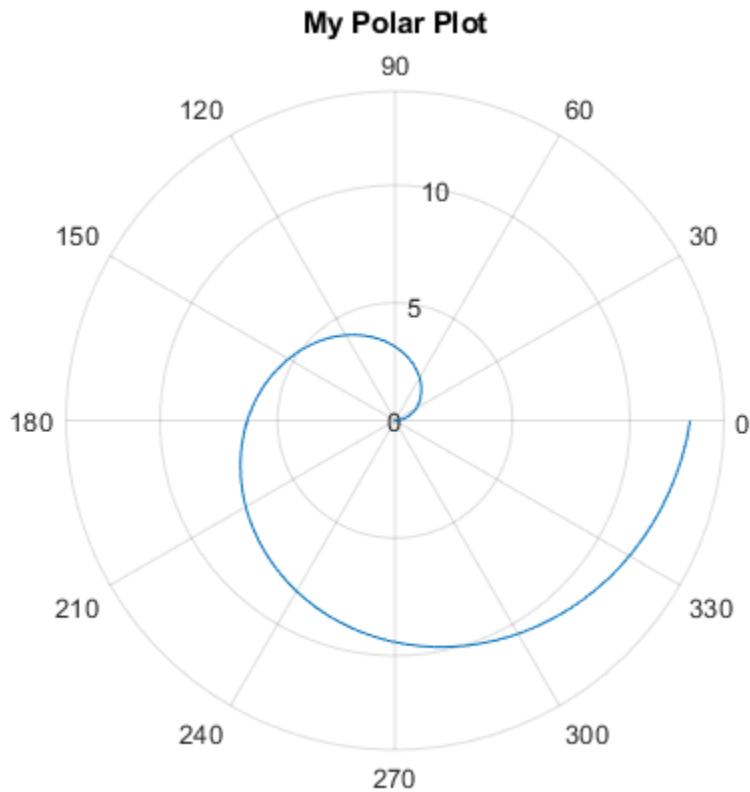
自定义极坐标区

为了便于查看图形，可以修改极坐标区的特定方面。例如，可以更改网格线位置和关联的标签。也可以更改网格线颜色和标签字体大小。

创建极坐标图

在极坐标中绘制线条和添加标题。

```
theta = linspace(0,2*pi);
rho = 2*theta;
figure
polarplot(theta,rho)
title('My Polar Plot')
```



使用属性自定义极坐标区

在创建极坐标图时，MATLAB 会创建一个 **PolarAxes** 对象。**PolarAxes** 对象具有可用于自定义极坐标区外观的属性，例如字体大小、颜色或刻度。有关完整列表，请参阅 **PolarAxes** 属性。

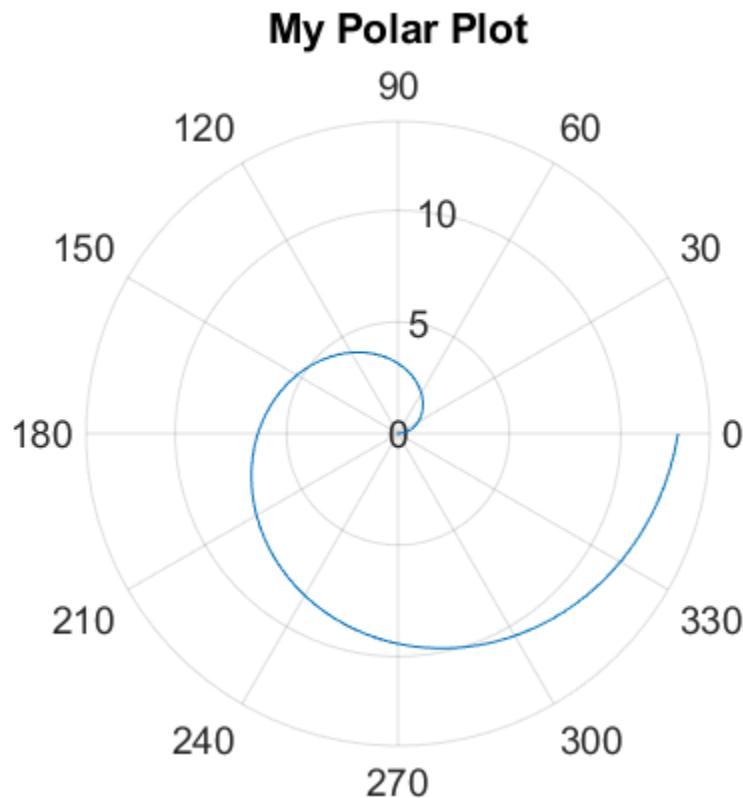
使用 **gca** 函数访问 **PolarAxes** 对象，例如 **pax = gca**。然后，结合使用 **pax** 和圆点表示法来设置属性，例如 **pax.FontSize = 14**。

```
pax = gca
pax =
PolarAxes (My Polar Plot) with properties:
```

```
ThetaLim: [0 360]
RLim: [0 14]
ThetaAxisUnits: 'degrees'
ThetaDir: 'counterclockwise'
ThetaZeroLocation: 'right'
```

Show all properties

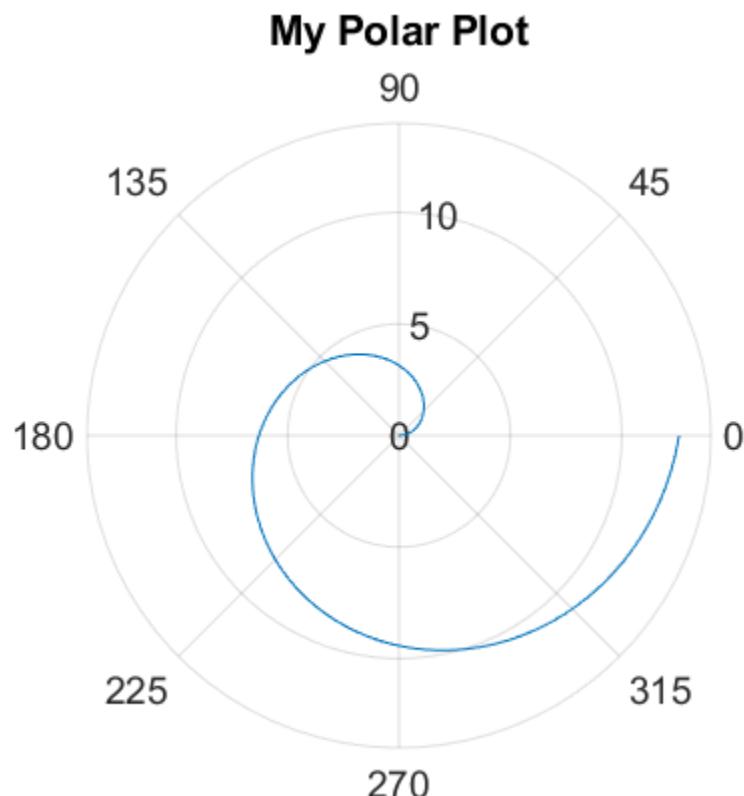
```
pax.FontSize = 14;
```



theta 轴刻度值

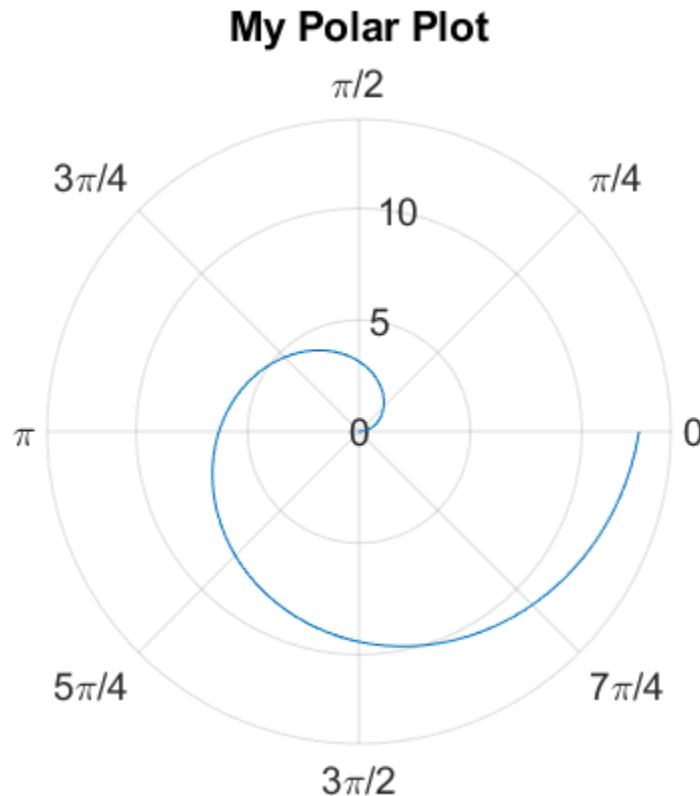
沿 theta 轴每隔 45 度显示刻度线。将这些位置指定为一个由递增值组成的向量。

```
thetaticks(0:45:315)
```



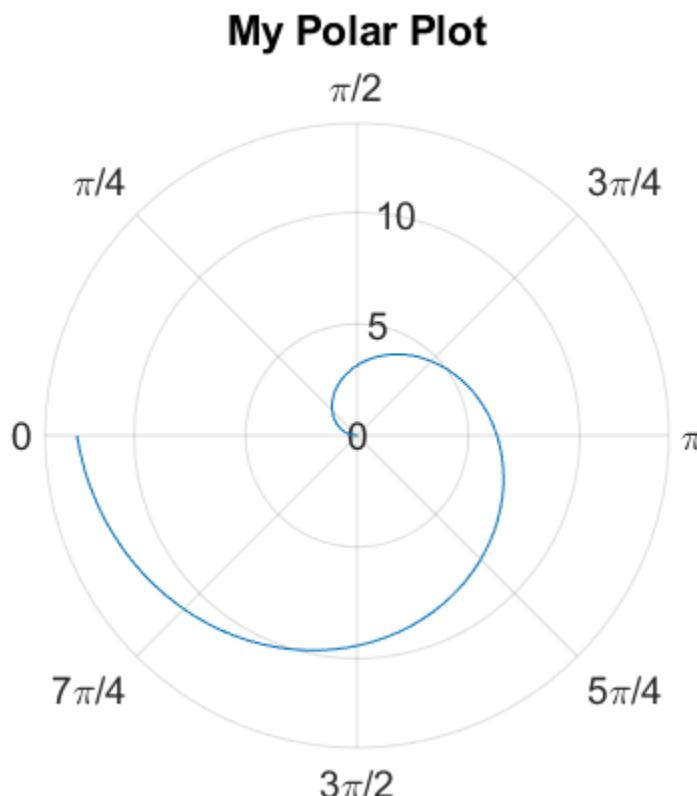
通过设置 `ThetaAxisUnits` 属性，以弧度（而不是度）为单位显示 theta 轴上的值。

```
pax = gca;  
pax.ThetaAxisUnits = 'radians';
```



修改 theta 轴，使其按顺时针方向增加。此外，还要旋转 theta 轴，以使基准角 0 位于左侧。

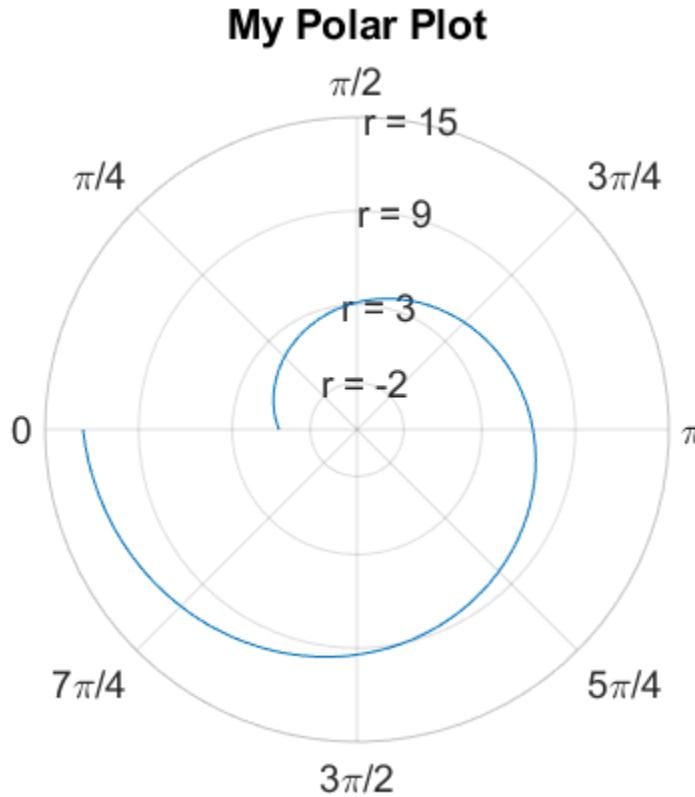
```
pax = gca;  
pax.ThetaDir = 'clockwise';  
pax.ThetaZeroLocation = 'left';
```



r 坐标轴范围、刻度值和标签

将 r 轴范围更改为值介于 -5 和 15 之间。在值 -2、3、9 和 15 处显示刻度线。然后，更改每个刻度线旁边显示的标签。将标签指定为字符向量元胞数组。

```
rlim([-5 15])
rticks([-2 3 9 15])
rticklabels({'r = -2','r = 3','r = 9','r = 15'})
```

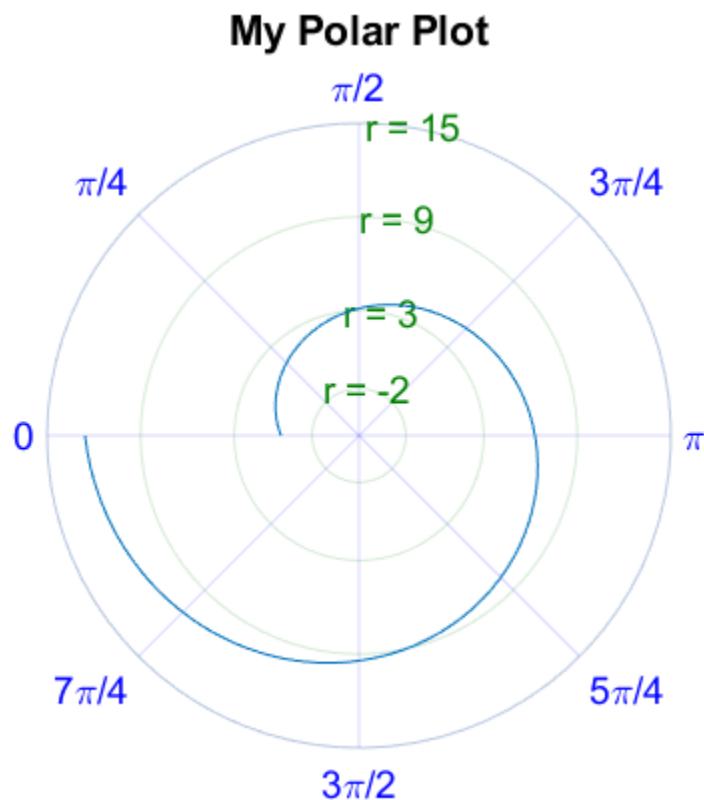


网格线和标签颜色

通过设置 `ThetaColor` 和 `RColor` 属性，对 theta 轴和 r 轴网格线及关联的标签使用不同的颜色。通过设置 `LineWidth` 属性，更改网格线的线宽。

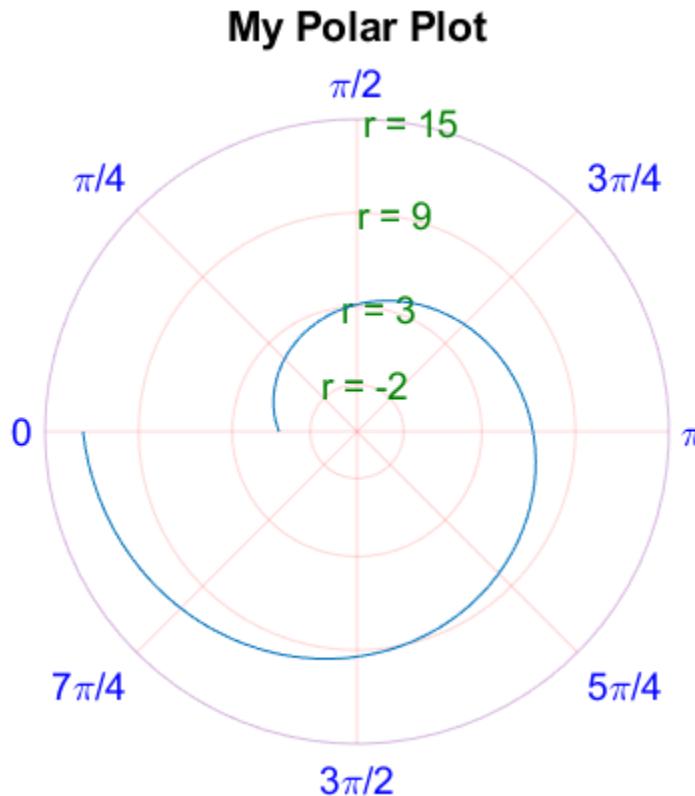
使用某个颜色名称（例如 `'blue'`）的字符向量或 RGB 三元组指定颜色。RGB 三元组是包含三个元素的行向量，其元素分别指定颜色中红、绿、蓝分量的强度。强度必须处于范围 [0,1] 中，例如 `[0.4 0.6 0.7]`。

```
pax = gca;
pax.ThetaColor = 'blue';
pax.RColor = [0 .5 0];
```



通过设置 `GridColor` 属性，在不影响标签的情况下更改所有网格线的颜色。

```
pax.GridColor = 'red';
```



指定 GridColor 属性后，ThetaColor 和 RColor 属性将不再影响网格线。如果希望 ThetaColor 和 RColor 属性影响网格线，则可将 GridColorMode 属性设置回 'auto'。

另请参阅

[polarplot](#) | [thetaticks](#) | [rticks](#) | [rticklabels](#) | [thetaticklabels](#) | [PolarAxes](#)

相关示例

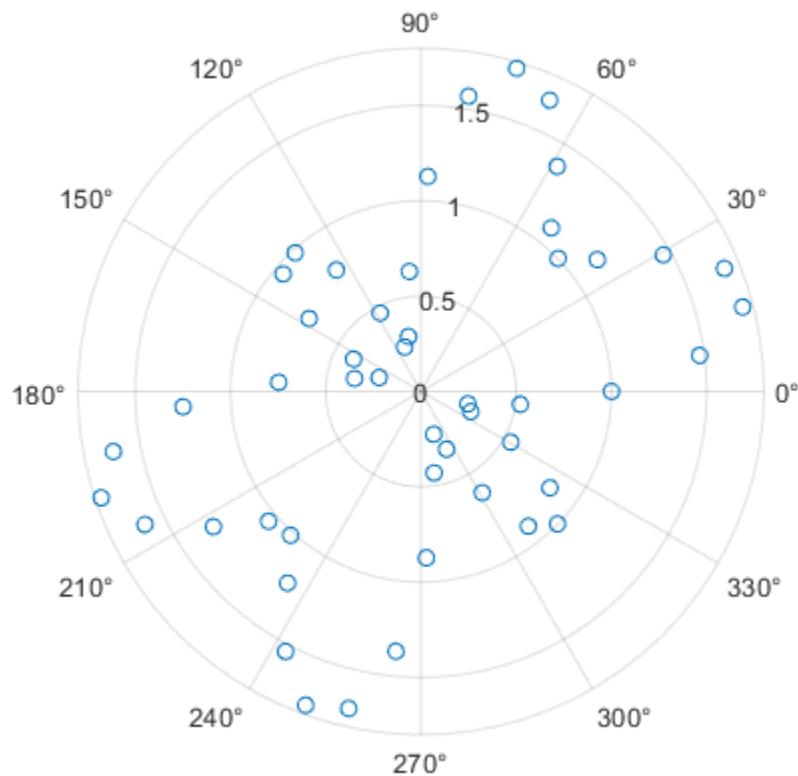
- “极坐标区上的罗盘标签” (第 3-22 页)

极坐标区上的罗盘标签

以下示例演示如何在极坐标中绘制数据图。此外，还演示如何指定要绘制网格线的角度以及如何指定标签。

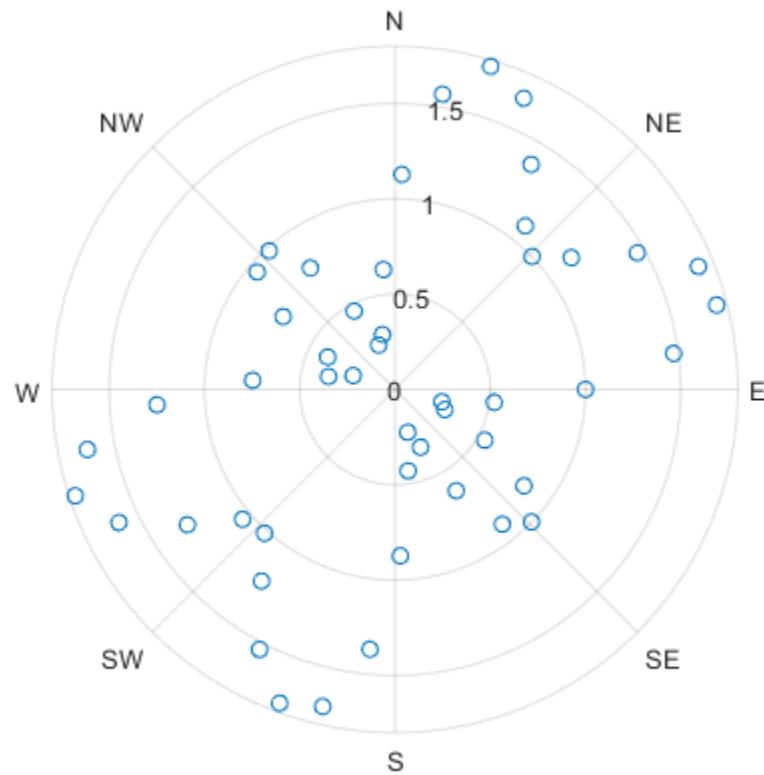
在极坐标中绘制数据图，并在每个数据点处显示一个圆形标记。

```
theta = linspace(0,2*pi,50);
rho = 1 + sin(4*theta).*cos(2*theta);
polarplot(theta,rho,'o')
```



使用 `gca` 访问极坐标区对象。通过设置 `ThetaTick` 属性，指定绘制网格线的角度。然后，通过设置 `ThetaTickLabel` 属性，指定每条网格线的标签。

```
pax = gca;
angles = 0:45:360;
pax.ThetaTick = angles;
labels = {'E','NE','N','NW','W','SW','S','SE'};
pax.ThetaTickLabel = labels;
```



另请参阅

[polarplot](#) | [thetaticks](#) | [rticks](#) | [rticklabels](#) | [thetaticklabels](#) | [PolarAxes](#)

相关示例

- “自定义极坐标区” (第 3-14 页)

等高线图

- “为等高线图添加层级标签” (第 4-2 页)
- “改变等高线图的填充颜色” (第 4-3 页)
- “突出显示特定等高线层级” (第 4-5 页)
- “合并等高线图和箭头图” (第 4-7 页)
- “带有主网格线和次网格线的等高线图” (第 4-9 页)

为等高线图添加层级标签

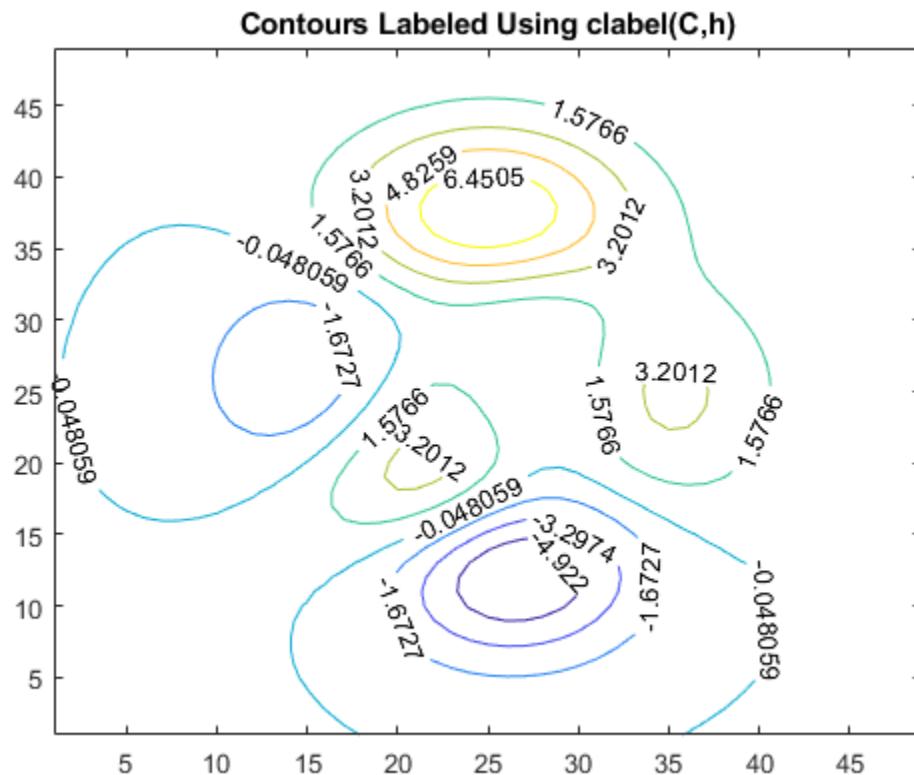
此示例如何用相关值标记等高线。

等高线矩阵 C 是由 `contour`、`contour3` 和 `contourf` 返回的一个可选输出参数。`clabel` 函数使用来自 C 的值显示二维等高线的标签。

显示 `peaks` 函数的八个等高线层级。`clabel` 只标记其大小足以容纳内联标签的等高线。

```
Z = peaks;
figure
[C,h] = contour(Z,8);

clabel(C,h)
title('Contours Labeled Using clabel(C,h)')
```



要以交互方式用鼠标选择要标记的等高线，请将 `manual` 选项传递给 `clabel`，例如 `clabel(C,h,'manual')`。当鼠标处于图窗中时，该命令会显示一个十字准线光标。点击鼠标，会标记离光标最近的等高线。

另请参阅

`contour` | `contour3` | `contourf` | `clabel`

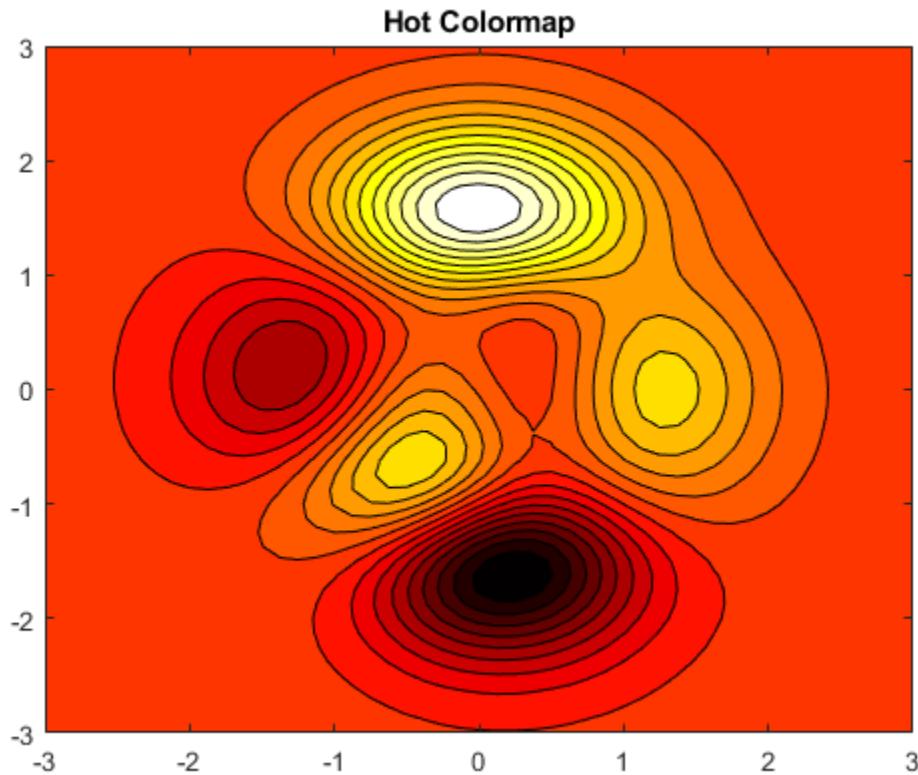
改变等高线图的填充颜色

此示例演示如何更改已填充等高线图的颜色。

更改颜色图

通过改变颜色图设置已填充等高线图的颜色。将预定义的颜色图名称 `hot` 传递给 `colormap` 函数。

```
[X,Y,Z] = peaks;
figure
contourf(X,Y,Z,20)
colormap(hot)
title('Hot Colormap')
```

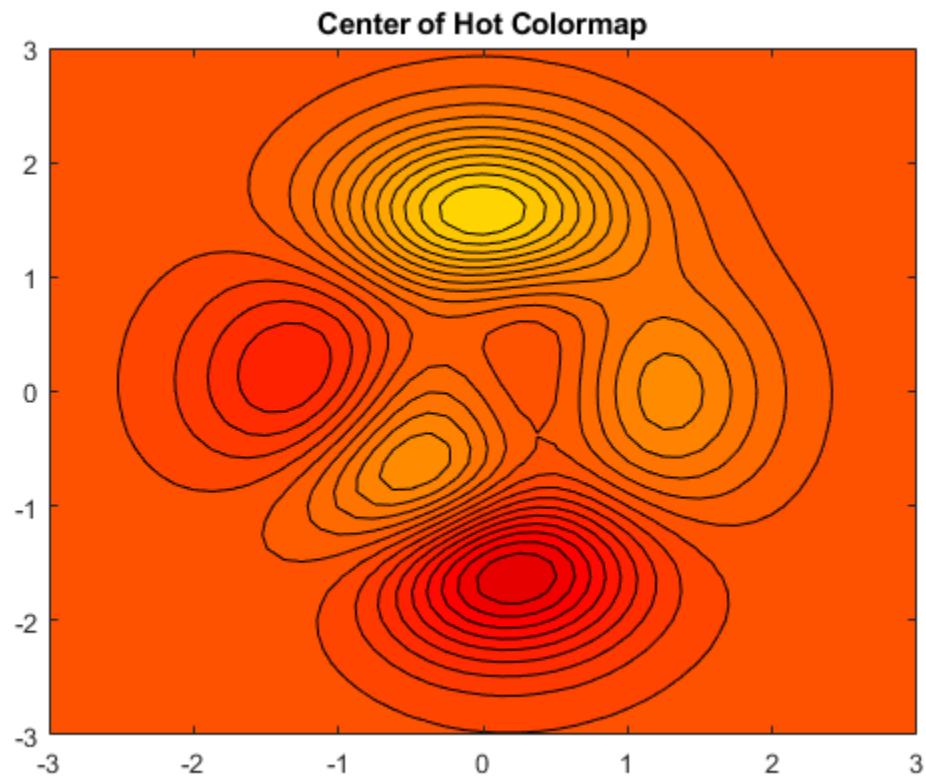


控制颜色值到颜色图的映射

通过将颜色轴缩放设置为比矩阵 Z 中的值大得多的范围值，保证只使用 `hot` 颜色图中间的颜色。`caxis` 函数用于控制数据值到颜色图的映射。使用此函数设置颜色轴缩放。

```
caxis([-20,20])
title('Center of Hot Colormap')
```

4 等高线图



另请参阅

[colormap](#) | [caxis](#) | [contourf](#)

突出显示特定等高线层级

此示例演示如何突出显示特定层级的等高线。

将 Z 定义为从 `peaks` 函数返回的矩阵。

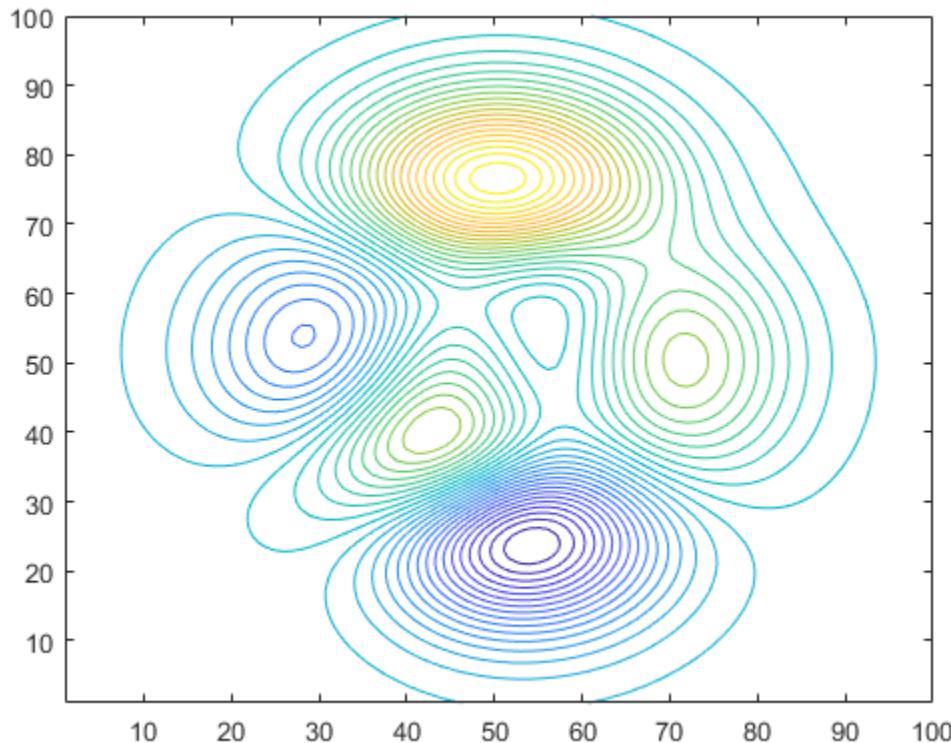
```
Z = peaks(100);
```

对 Z 中的最小和最大数据值四舍五入取整，并分别将这些值存储到 `zmin` 和 `zmax` 中。将 `zlevs` 定义为 40 个介于 `zmin` 和 `zmax` 之间的值。

```
zmin = floor(min(Z(:)));
zmax = ceil(max(Z(:)));
zinc = (zmax - zmin) / 40;
zlevs = zmin:zinc:zmax;
```

绘制等高线图。

```
figure
contour(Z,zlevs)
```



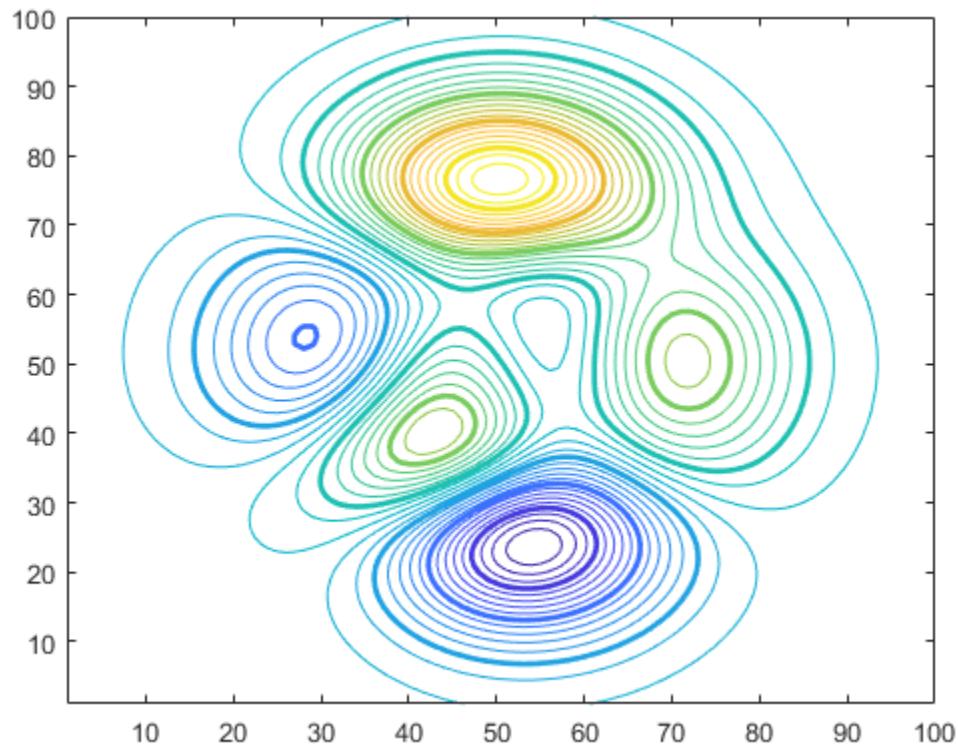
将 `zindex` 定义为 `zmin` 与 `zmax` 之间索引值为 2 的整数值向量。

```
zindex = zmin:2:zmax;
```

保留之前的等高线图。创建第二个等高线图并使用 `zindex` 每隔一个整数值突出显示等高线。将线宽设置为 2。

4 等高线图

```
hold on  
contour(Z,zindex,'LineWidth',2)  
hold off
```



另请参阅

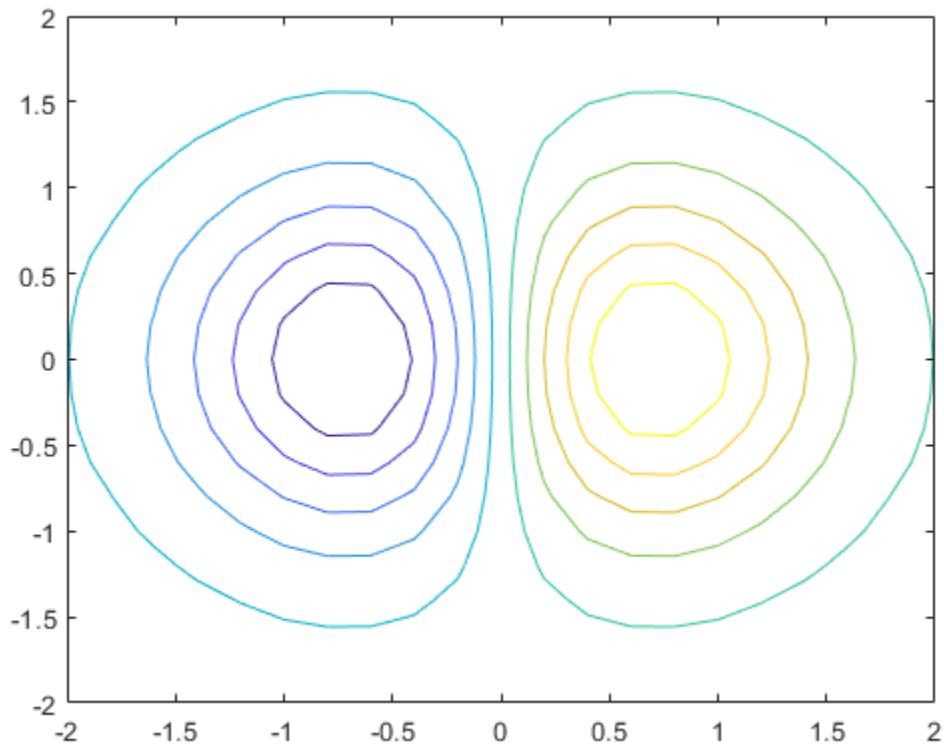
[contour](#) | [floor](#) | [ceil](#) | [min](#) | [max](#) | [hold](#)

合并等高线图和箭头图

在同一绘图上显示等高线和梯度向量。

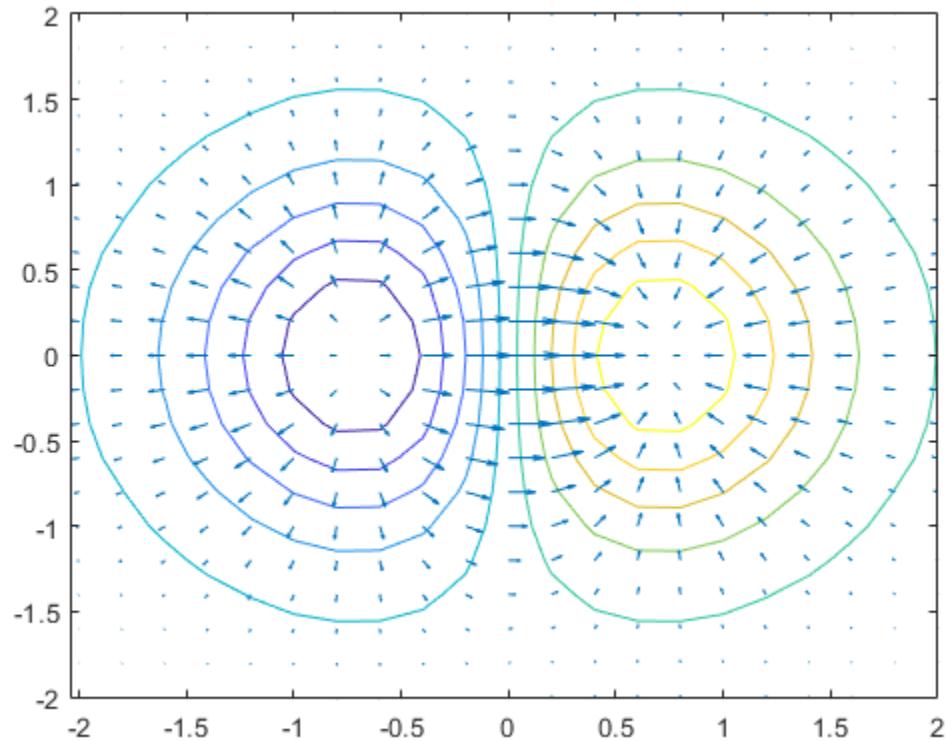
沿 x 和 y 方向在从 -2 到 2 的网格上绘制 10 条 $xe^{-x^2-y^2}$ 等高线。

```
[X,Y] = meshgrid(-2:0.2:2);
Z = X .* exp(-X.^2 - Y.^2);
contour(X,Y,Z,10)
```



使用 `gradient` 函数计算 Z 的二维梯度。`gradient` 函数会返回 U 作为 x 方向上的梯度，返回 V 作为 y 方向上的梯度。使用 `quiver` 函数显示指示梯度值的箭头。

```
[U,V] = gradient(Z,0.2,0.2);
hold on
quiver(X,Y,U,V)
hold off
```



另请参阅

contour | hold

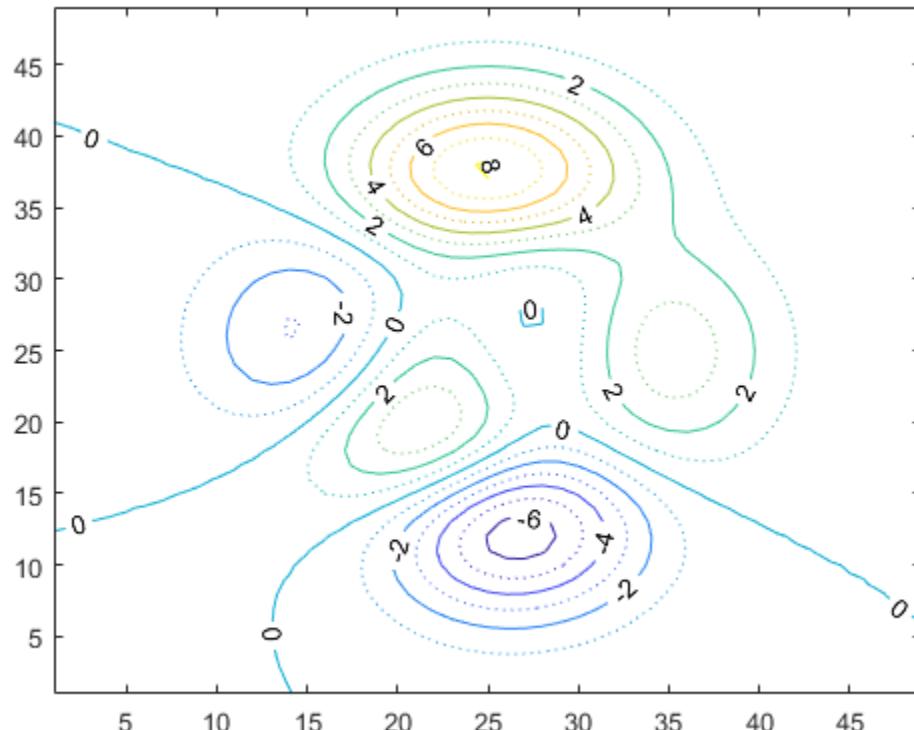
带有主网格线和次网格线的等高线图

您可以通过拆分数据并创建两个重叠的等高线图来创建突出所选等高线的等高线图。

例如，创建 `peaks` 函数的等高线图，其中偶数编号的等高线为实线，奇数编号的等高线为虚线。为偶数编号的水平绘制一个等高线。然后，在其上覆盖以虚线为奇数编号的水平绘制的第二个等高线图。

```
major = -6:2:8;
minor = -5:2:7;
[cmajor,hmajor] = contour(peaks,'LevelList',major);
clabel(cmajor,hmajor)

hold on
[cminor,hminor] = contour(peaks,'LevelList',minor);
hminor.LineStyle = ':';
hold off
```



另请参阅

`contour` | `contourf` | `clabel` | `hold`

专用图

- “基于表格数据创建热图” (第 5-2 页)
- “使用字符串数组创建文字云” (第 5-11 页)
- “使用平行坐标图探索表数据” (第 5-14 页)

基于表格数据创建热图

热图是一种使用颜色实现数据可视化的方式。此示例说明如何将文件作为表导入 MATLAB® 并根据表列创建热图。它还说明如何修改热图的外观，例如设置标题和轴标签。

以表的形式导入文件

加载示例文件 **TemperatureData.csv**，此文件包含 2015 年 1 月至 2016 年 7 月间的日均温度。将此文
件读取到一个表中并显示前五行。

```
tbl = readtable('TemperatureData.csv');
head(tbl,5)

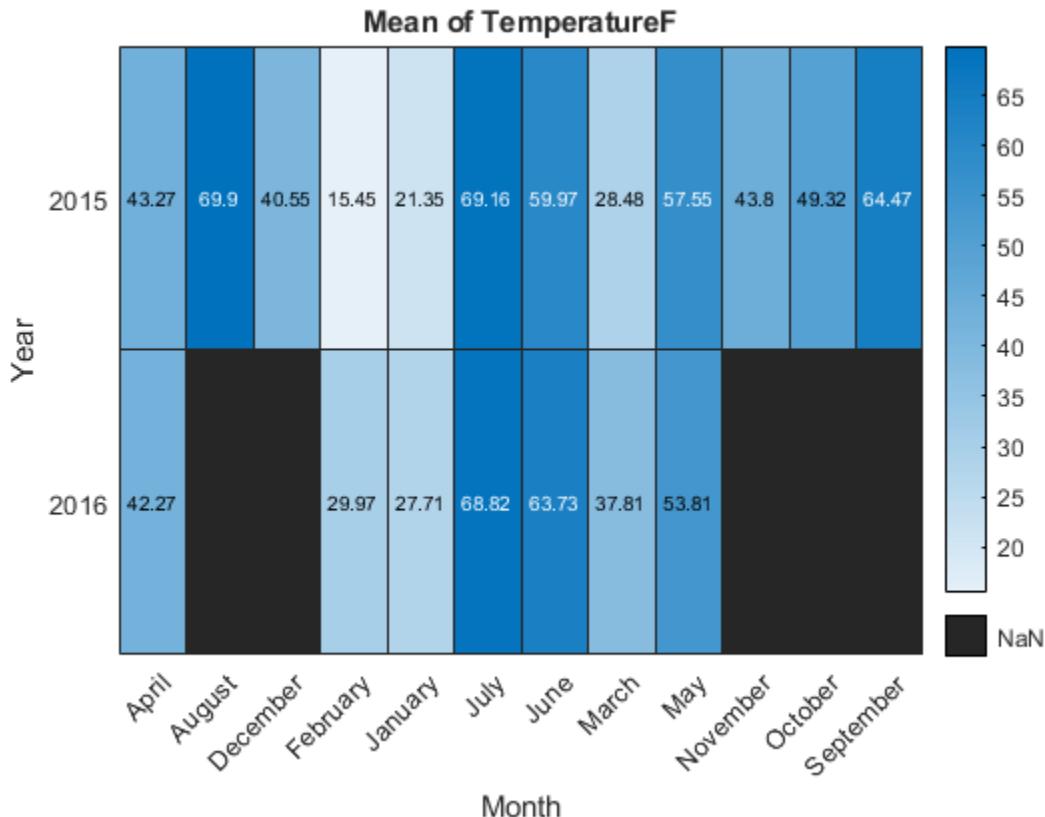
ans=5×4 table
  Year    Month    Day    TemperatureF
  ____   _____  ____    _____

  2015  {'January'}  1      23
  2015  {'January'}  2      31
  2015  {'January'}  3      25
  2015  {'January'}  4      39
  2015  {'January'}  5      29
```

创建基本热图

创建一个热图，其中 x 轴表示月份，y 轴表示年份。通过设置 **ColorVariable** 属性，使用温度数据为热
图单元格着色。将 **HeatmapChart** 对象赋给变量 **h**。在创建图后，可使用 **h** 对其进行修改。

```
h = heatmap(tbl,'Month','Year','ColorVariable','TemperatureF');
```



默认情况下，MATLAB 会将颜色数据作为每月的平均温度进行计算。但是，可通过设置 **ColorMethod** 属性来更改计算方法。

对轴值重新排序

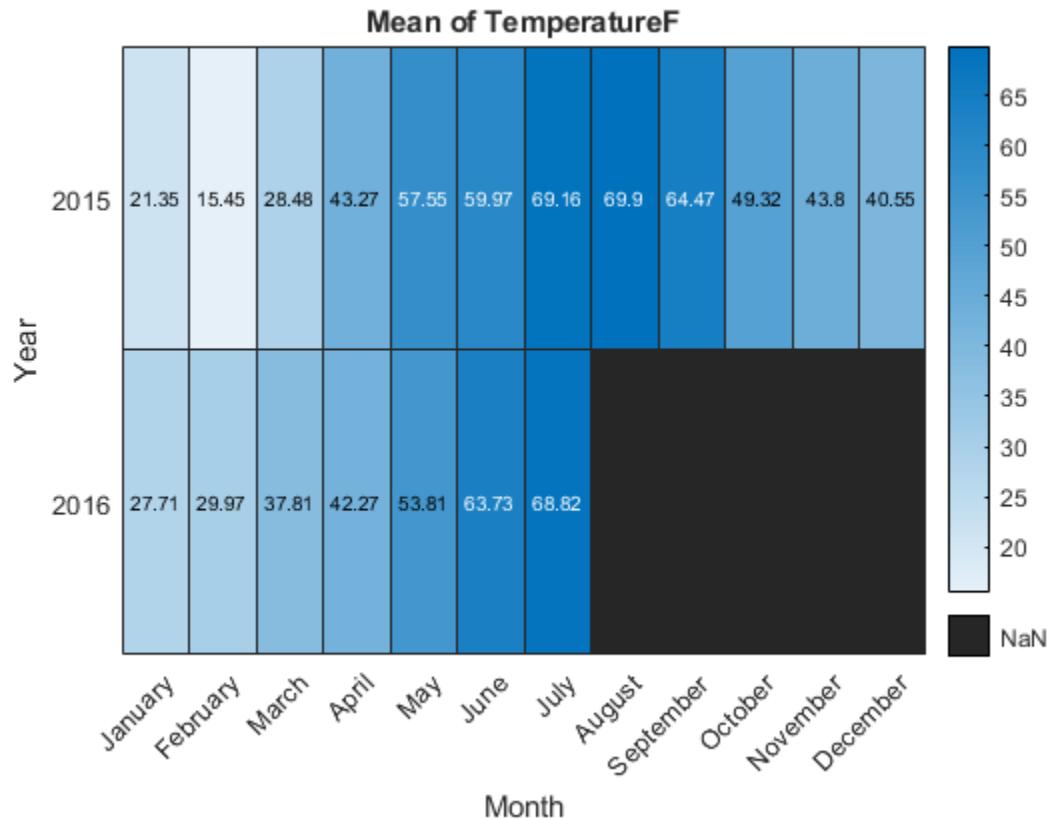
轴值按字母顺序显示。对月份重新排序，以使其按年月顺序显示。可以使用分类数组或通过设置 **HeatmapChart** 属性来自定义标签。

要使用分类数组，首先将该表的 **Month** 列中的数据从元胞数组更改为分类数组。然后使用 **reordercats** 函数对类别重新排序。可以将这些函数应用于工作区中的表 (tbl) 或 **HeatmapChart** 对象的 **SourceTable** 属性中存储的表 (h.SourceTable)。将这些函数应用于 **HeatmapChart** 对象中存储的表可避免影响原始数据。

```

h.SourceTable.Month = categorical(h.SourceTable.Month);
neworder = {'January','February','March','April','May','June','July',...
    'August','September','October','November','December'};
h.SourceTable.Month = reordercats(h.SourceTable.Month,neworder);

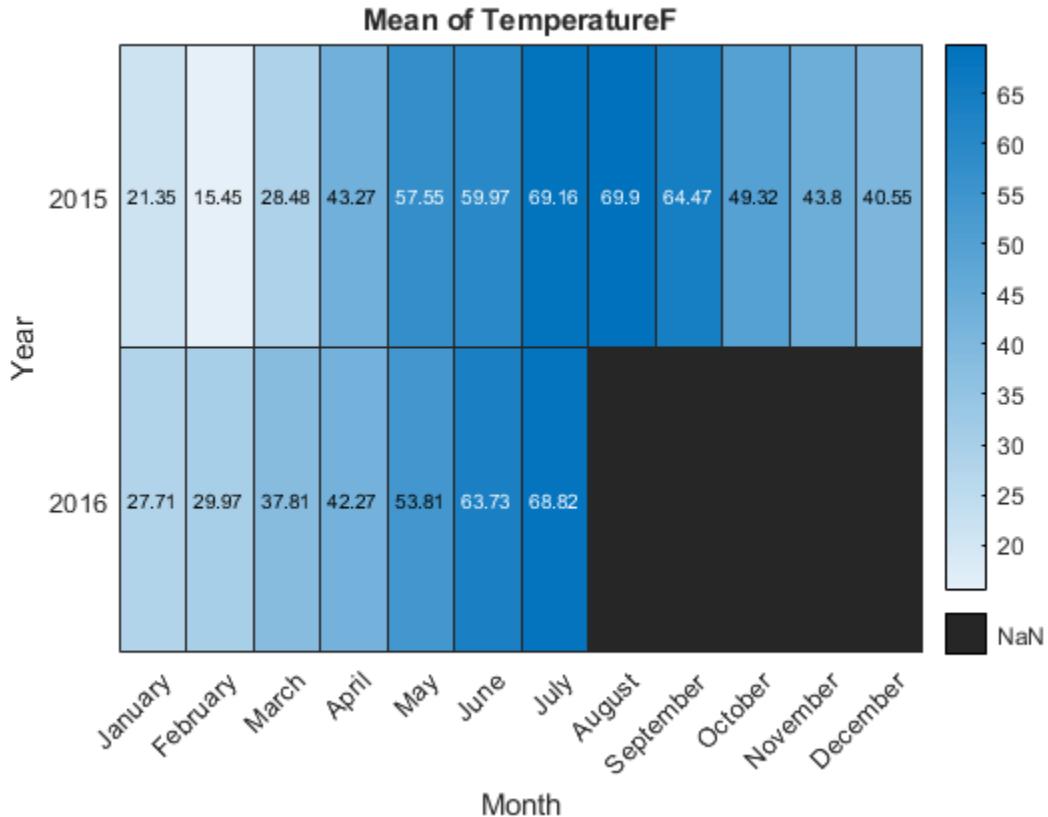
```



对于分类数组，同样可以使用 `addcats`、`removecats` 或 `renamecats` 函数添加、删除或重命名热图标签。

也可以使用 `HeatmapChart` 对象的 `XDisplayData` 和 `YDisplayData` 属性对轴值重新排序。

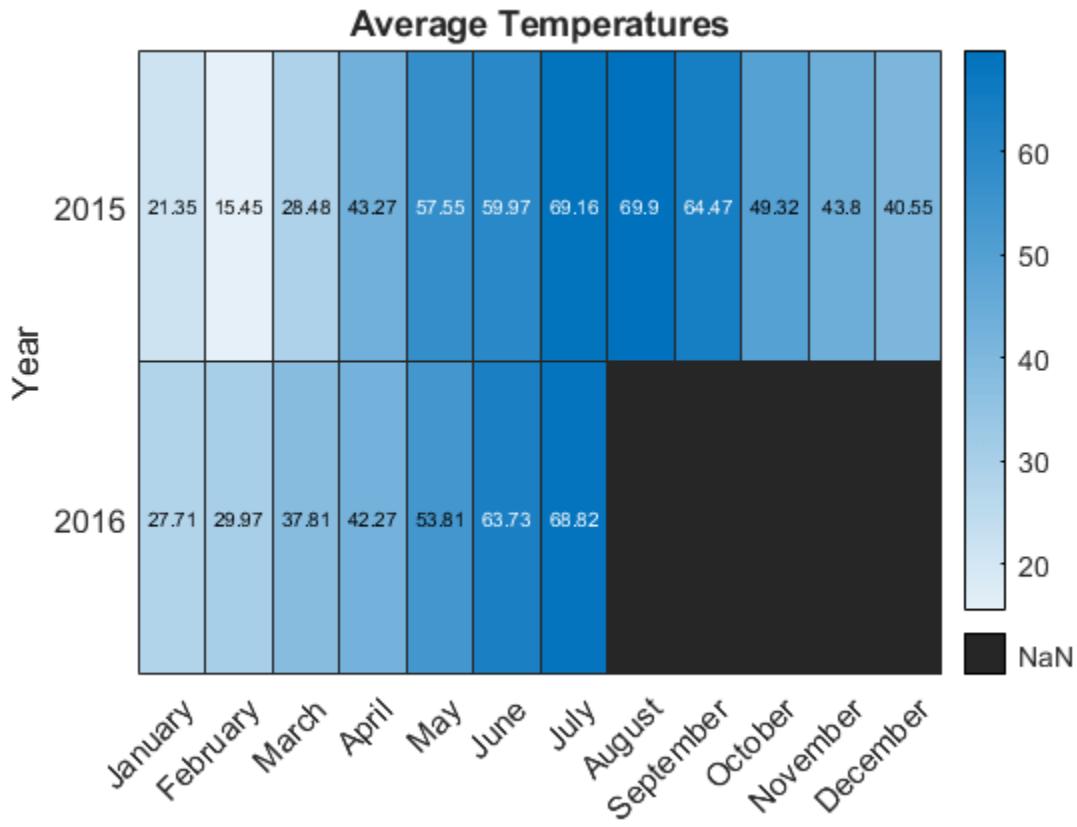
```
h.XDisplayData = {'January','February','March','April','May','June',...
    'July','August','September','October','November','December'};
```



修改标题和轴标签

使用表格数据创建热图时，会自动生成热图的标题和轴标签。通过设置 **HeatmapChart** 对象的 **Title**、**XLabel** 和 **YLabel** 属性，自定义标题和轴标签。例如更改标题和删除 x 轴标签。另外还可以更改字体大小。

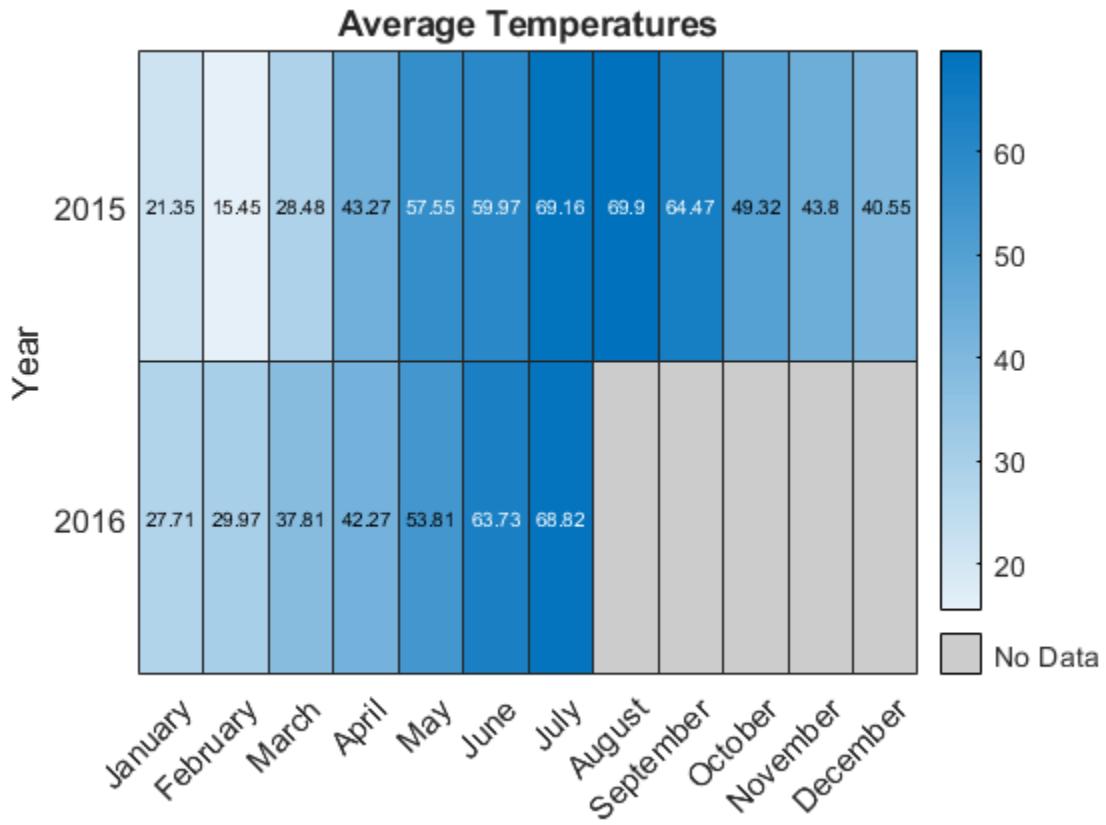
```
h.Title = 'Average Temperatures';  
h.XLabel = " ";  
h.FontSize = 12;
```



修改缺失数据元胞的外观

由于缺少 2016 年 8 月至 2016 年 12 月的数据，因此这些元胞显示为缺失数据。使用 `MissingDataColor` 和 `MissingDataLabel` 属性修改缺失数据元胞的外观。

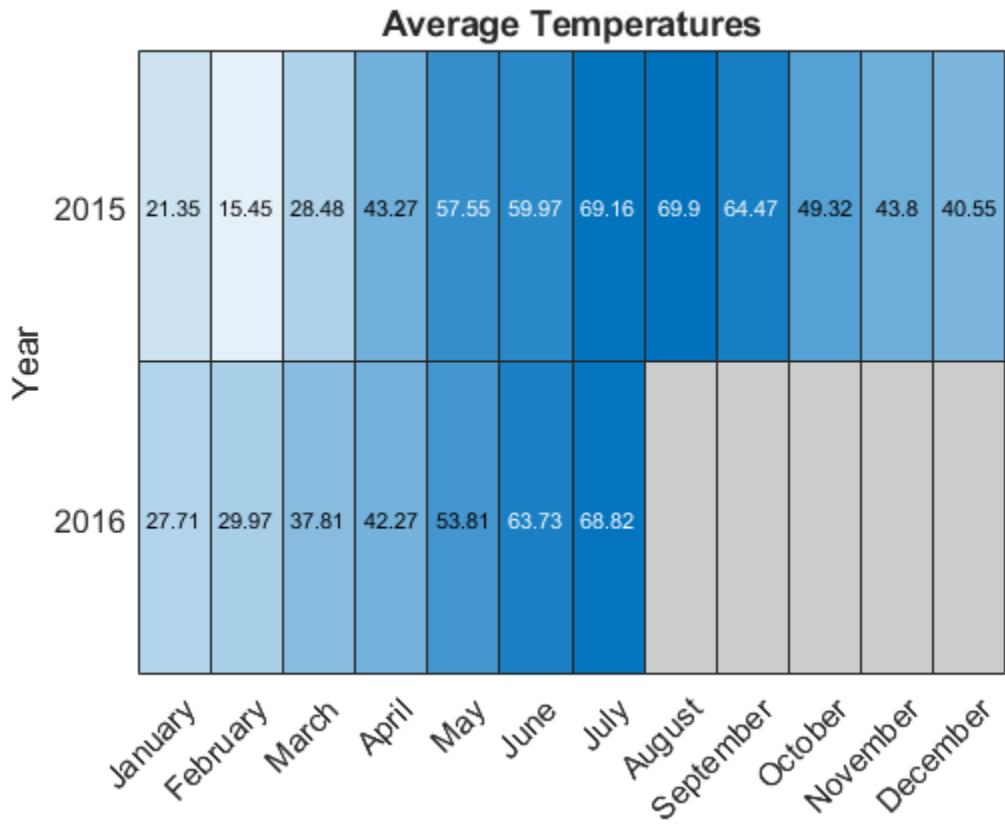
```
h.MissingDataColor = [0.8 0.8 0.8];  
h.MissingDataLabel = 'No Data';
```



删除颜色栏

通过设置 `ColorbarVisible` 属性删除颜色栏。

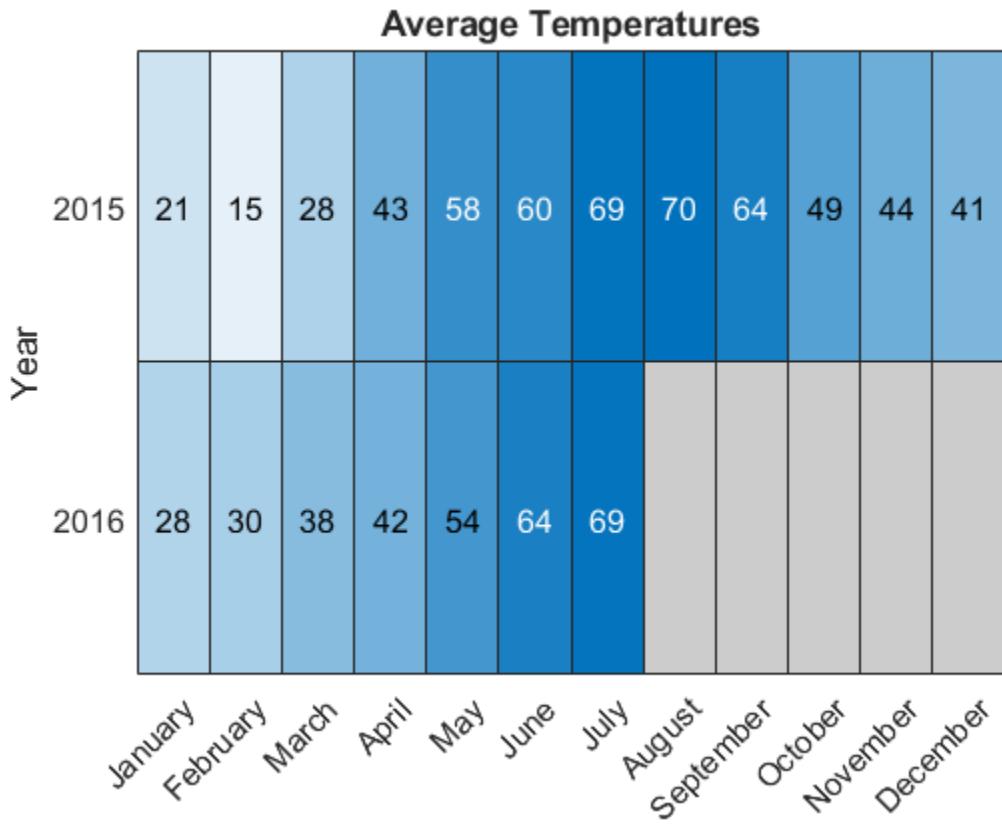
```
h.ColorbarVisible = 'off';
```



设置元胞文本格式

通过设置 `CellLabelFormat` 属性自定义每个元胞中显示的文本的格式。例如，显示不带小数点的值文本。

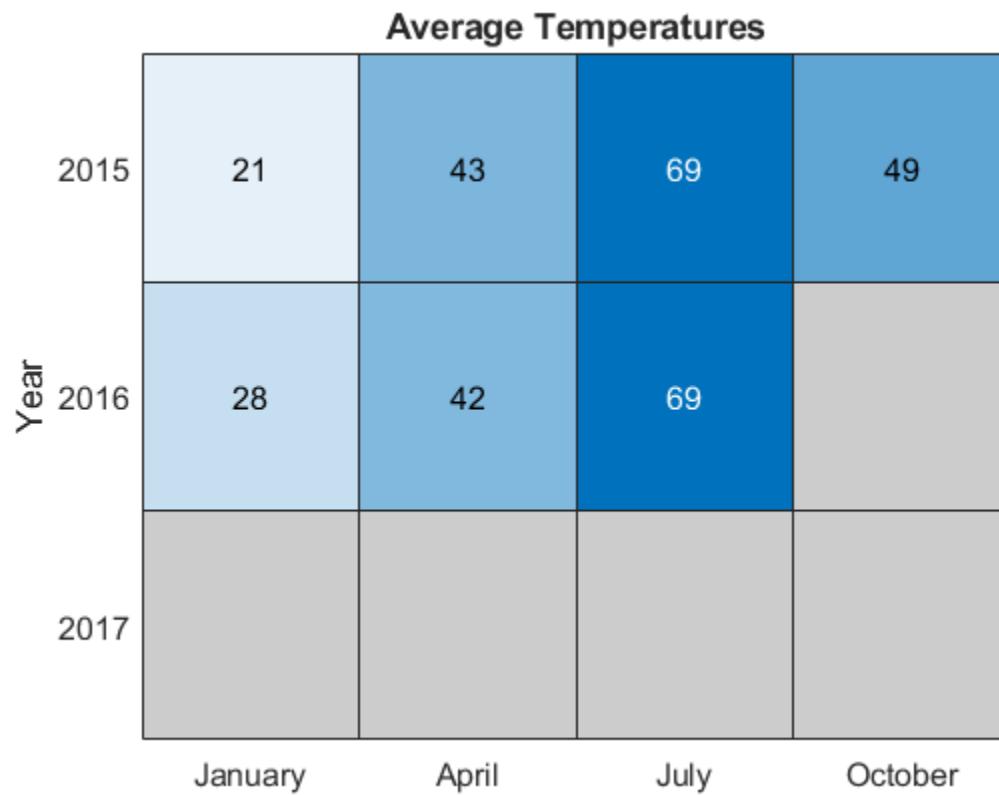
```
h.CellLabelFormat = '%.0f';
```



添加或删除轴值

通过设置 XDisplayData 属性，仅显示每个季度的第一个月。通过设置 YDisplayData 属性，沿 y 轴添加年份 2017。将这些属性分别设置为 XData 或 YData 中的值的子集、超集或置换。

```
h.XDisplayData = {'January','April','July','October'};
h.YDisplayData = {'2015','2016','2017'};
```



由于没有与 2017 年关联的数据，因此热图单元格使用缺失数据颜色。

另请参阅

函数

[heatmap](#) | [table](#) | [readtable](#) | [addcats](#) | [removecats](#) | [renamecats](#) | [reordercats](#) | [categorical](#)

属性

[HeatmapChart](#)

使用字符串数组创建文字云

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。如果您安装了 Text Analytics Toolbox™，则可以直接使用字符串数组创建文字云。有关详细信息，请参阅 `wordcloud` (Text Analytics Toolbox) (Text Analytics Toolbox)。

使用 `fileread` 函数从莎士比亚的十四行诗中读取文本。

```
sonnets = fileread('sonnets.txt');
sonnets(1:135)
```

```
ans =
'THE SONNETS
```

```
by William Shakespeare
```

I

From fairest creatures we desire increase,
That thereby beauty's rose might never die,'

使用 `string` 函数将文本转换为字符串。然后，使用 `splitlines` 函数按换行符对其进行拆分。

```
sonnets = string(sonnets);
sonnets = splitlines(sonnets);
sonnets(10:14)
```

```
ans = 5x1 string
" From fairest creatures we desire increase,"
" That thereby beauty's rose might never die,"
" But as the riper should by time decease,"
" His tender heir might bear his memory."
" But thou, contracted to thine own bright eyes,"
```

用空格替换一些标点字符。

```
p = [".," "?","!",";",":"];
sonnets = replace(sonnets,p," ");
sonnets(10:14)
```

```
ans = 5x1 string
" From fairest creatures we desire increase "
" That thereby beauty's rose might never die "
" But as the riper should by time decease "
" His tender heir might bear his memory "
" But thou contracted to thine own bright eyes "
```

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。

```
sonnets = join(sonnets);
sonnets = split(sonnets);
sonnets(7:12)
```

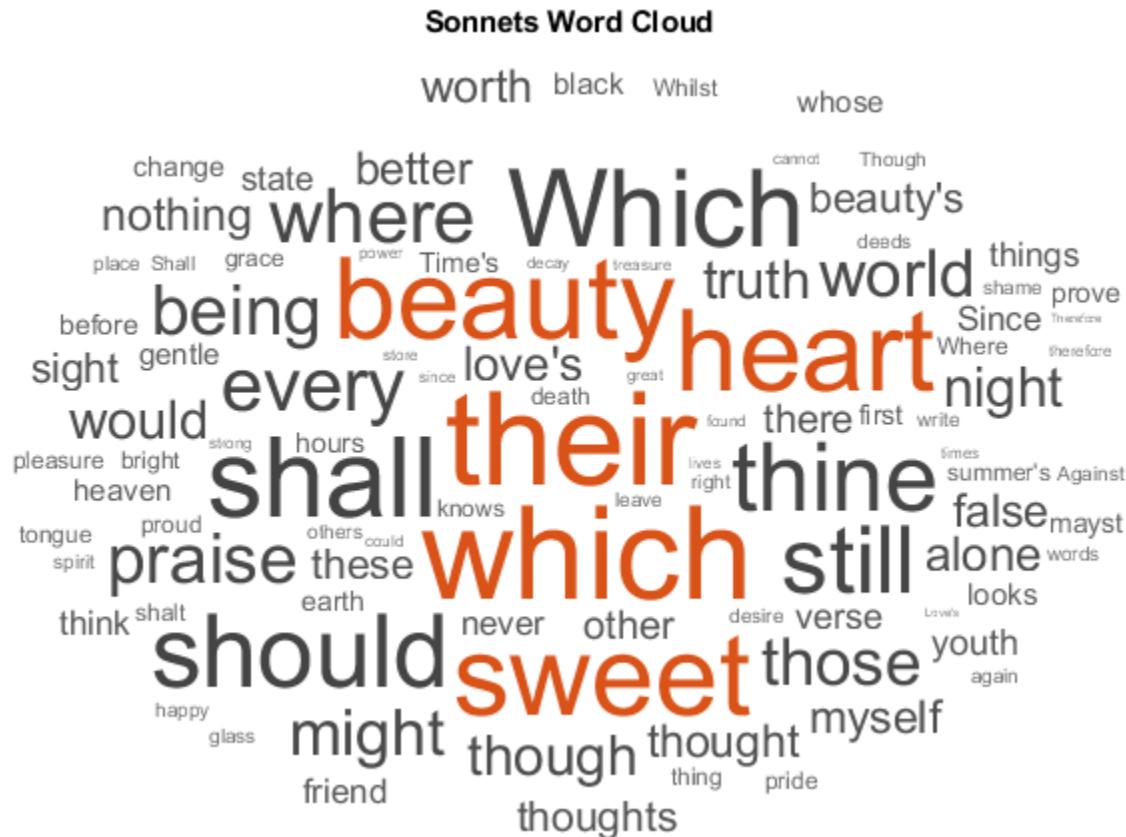
```
ans = 6x1 string  
"From"  
"fairest"  
"creatures"  
"we"  
"desire"  
"increase"
```

删除少于五个字符的单词。

```
sonnets(strlength(sonnets)<5) = [];
```

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。

```
C = categorical(sonnets);
figure
wordcloud(C);
title("Sonnets Word Cloud")
```



另请参阅

[wordcloud | WordCloudChart Properties](#)

使用平行坐标图探索表数据

此示例显示如何将文件以表的形式导入 MATLAB®，从表格数据创建平行坐标图，以及修改绘图的外观。

平行坐标图适用于可视化具有多个列的表格或矩阵数据。输入数据的行对应于绘图中的线条，输入数据的列对应于绘图中的坐标轴。您可以将绘图中的线条进行分组，以便更好地查看数据中的趋势。

以表的形式导入文件

加载示例文件 **TemperatureData.csv**，此文件包含 2015 年 1 月至 2016 年 7 月间的日均温度。将此文
件读取到一个表中，并显示前几行。

```
tbl = readtable("TemperatureData.csv");
head(tbl)

ans=8×4 table
  Year    Month    Day    TemperatureF
  ____    _____   ____    _____

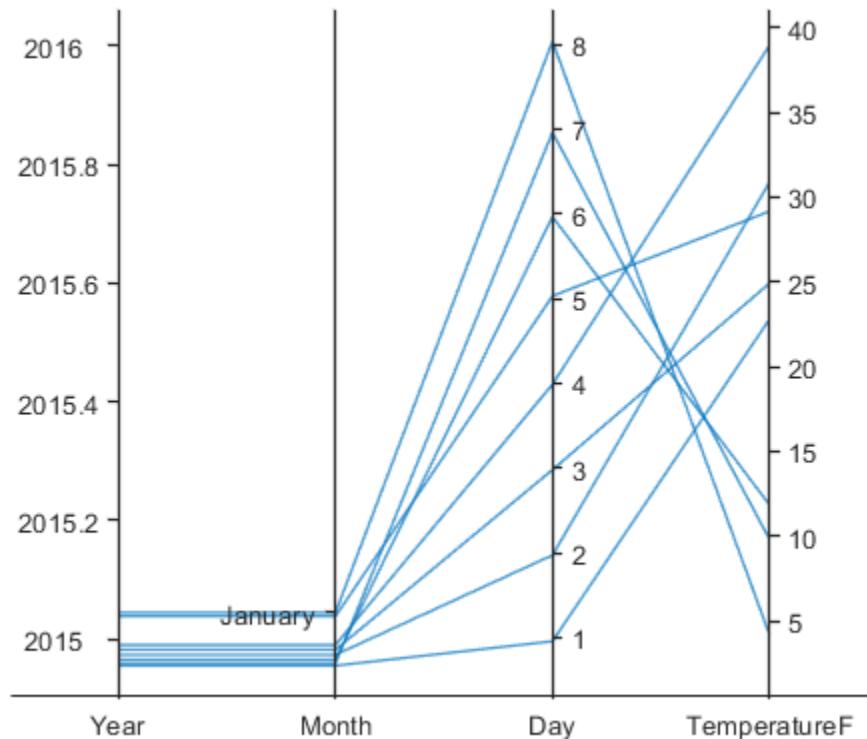
  2015    {'January'}    1    23
  2015    {'January'}    2    31
  2015    {'January'}    3    25
  2015    {'January'}    4    39
  2015    {'January'}    5    29
  2015    {'January'}    6    12
  2015    {'January'}    7    10
  2015    {'January'}    8     4
```

创建基本的平行坐标图

用表的前几行数据创建一个平行坐标图。绘图中的每个线条对应于表中的一行。默认情况下，**parallelplot** 会按照所有坐标变量在表中出现的顺序显示这些坐标变量。软件将在坐标标尺下方显示相应的坐标变量名称。

该绘图显示，表的前 8 行提供了 2015 年 1 月前 8 天的温度数据。例如，就日均温度而言，第 8 天是这 8 天中最寒冷的一天。

```
parallelplot(head(tbl))
```



为了帮助您解读绘图，默认情况下，MATLAB 会在绘图中沿坐标标尺随机抖动线条，使其不会完全重叠。例如，虽然前 8 个观测值具有相同的 Year 和 Month 值，但图中的线并没有完全对齐到 Year 坐标标尺的 2015 刻度线处，也没有完全对齐到 Month 坐标标尺的 January 刻度线处。虽然抖动会影响所有坐标变量，但它通常在分类坐标标尺上更加明显，因为抖动量取决于刻度线之间的距离。您可以设置 Jitter 属性来控制绘图中的抖动量。

请注意，Year 坐标标尺上的一些刻度线是无意义的十进制值。为确保坐标标尺上的刻度线仅对应有意义的值，您可以使用 categorical 函数将变量转换为分类变量。

```
tbl.Year = categorical(tbl.Year);
```

现在用整个表的数据创建一个平行坐标图。将 ParallelCoordinatesPlot 对象赋给变量 p，以便在创建绘图后使用 p 进行修改。例如，使用 Title 属性为绘图添加标题。

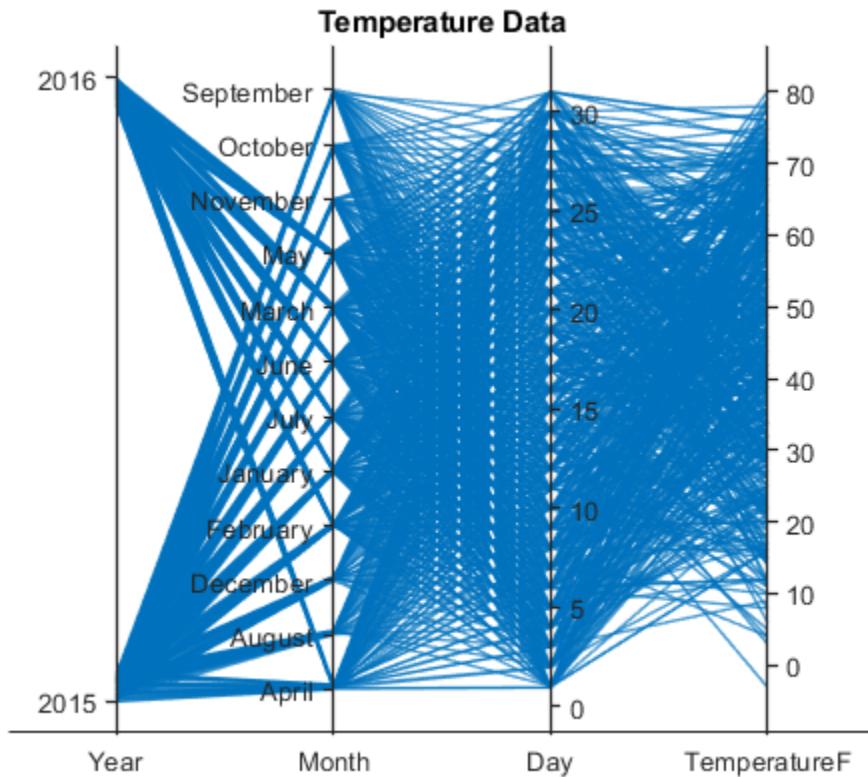
```
p = parallelplot(tbl)
```

```
p =
ParallelCoordinatesPlot with properties:
```

```
SourceTable: [565x4 table]
CoordinateVariables: {'Year' 'Month' 'Day' 'TemperatureF'}
GroupVariable: "
```

Show all properties

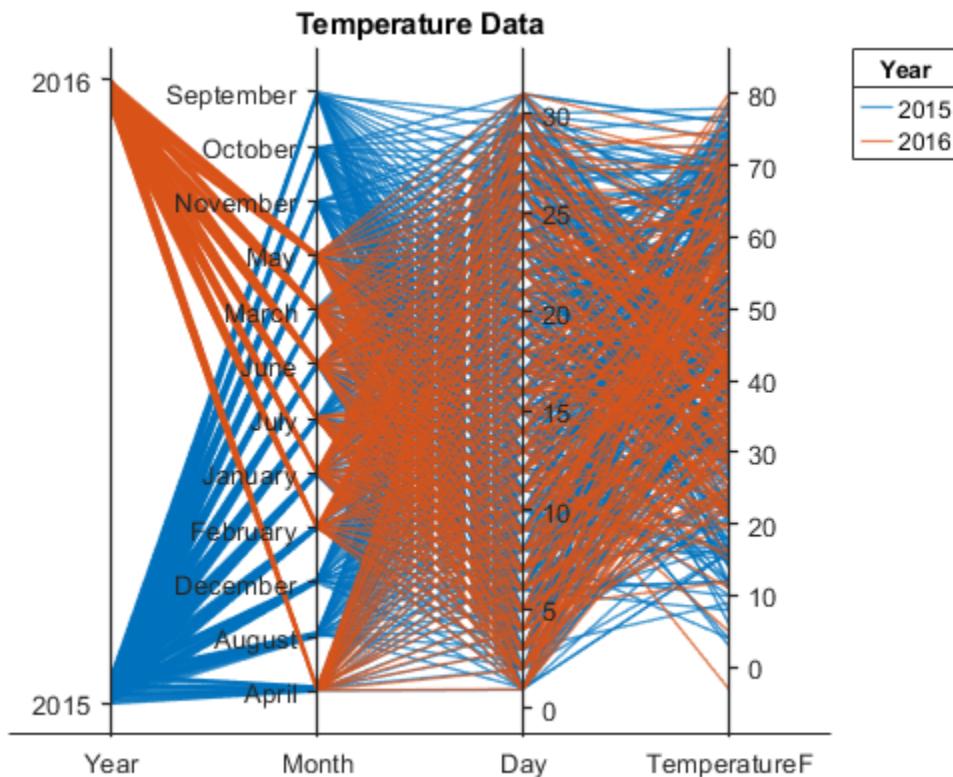
```
p.Title = 'Temperature Data';
```



对绘图线条分组

通过设置 **GroupVariable** 属性，根据 Year 值对绘图中的线条进行分组。默认情况下，MATLAB 会在绘图中添加一个图例。可以通过将 **LegendVisible** 属性设置为 'off' 来删除该图例。

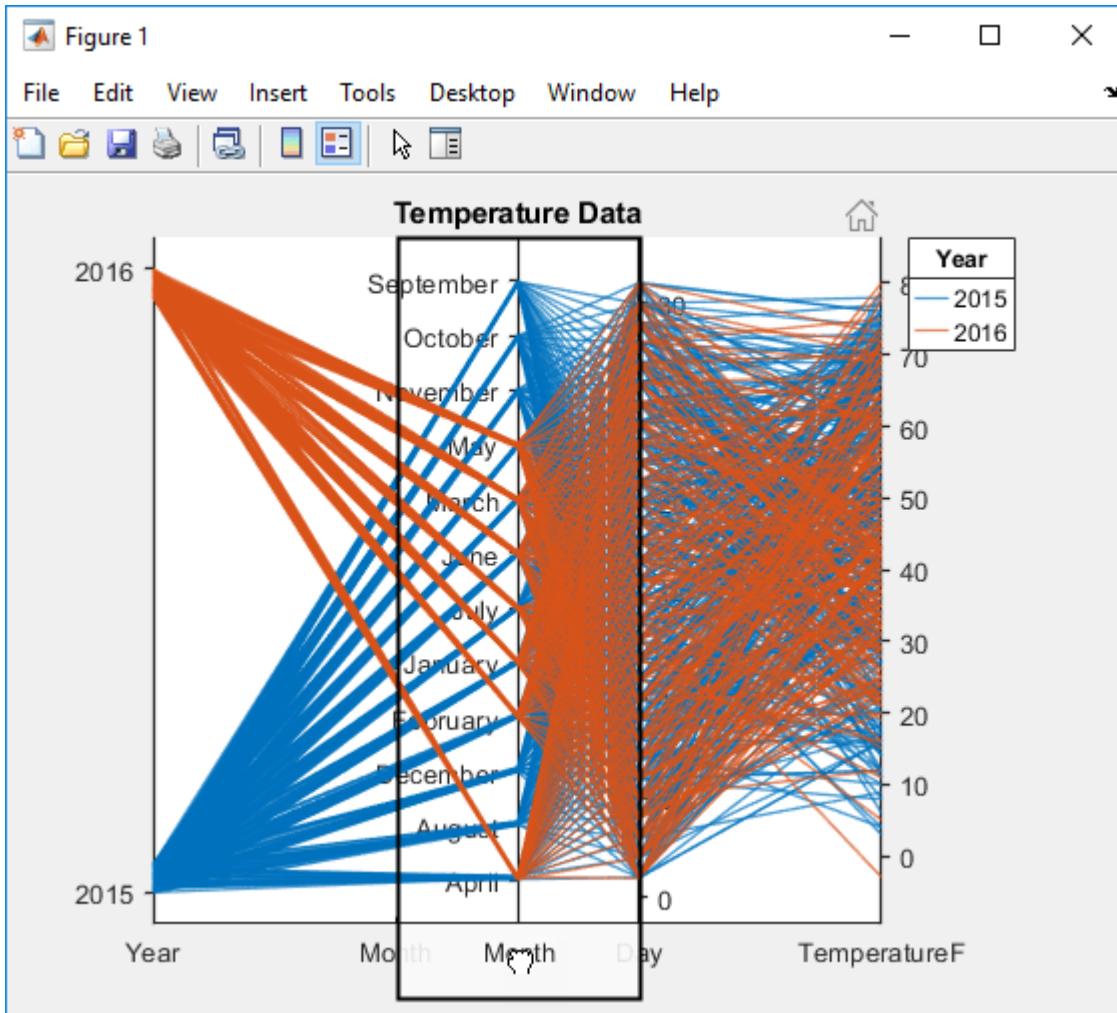
```
p.GroupVariable = 'Year';
```



交互式重排坐标变量

以交互方式重新排列坐标变量，以便更轻松地比较它们，并决定在绘图中保留哪些变量。

在图窗口中打开您的绘图。点击坐标刻度标签，并将关联的坐标标尺拖到您选择的位置。软件用一个黑色矩形来突出显示所选坐标标尺。例如，您可以点击 Month 坐标刻度标签，然后向右拖动坐标标尺。然后，您可以轻松比较 Month 和 TemperatureF 的值。



当您以交互方式重新排列坐标变量时，软件会更新绘图的相关 **CoordinateTickLabels**、**CoordinateVariables** 和 **CoordinateData** 属性。

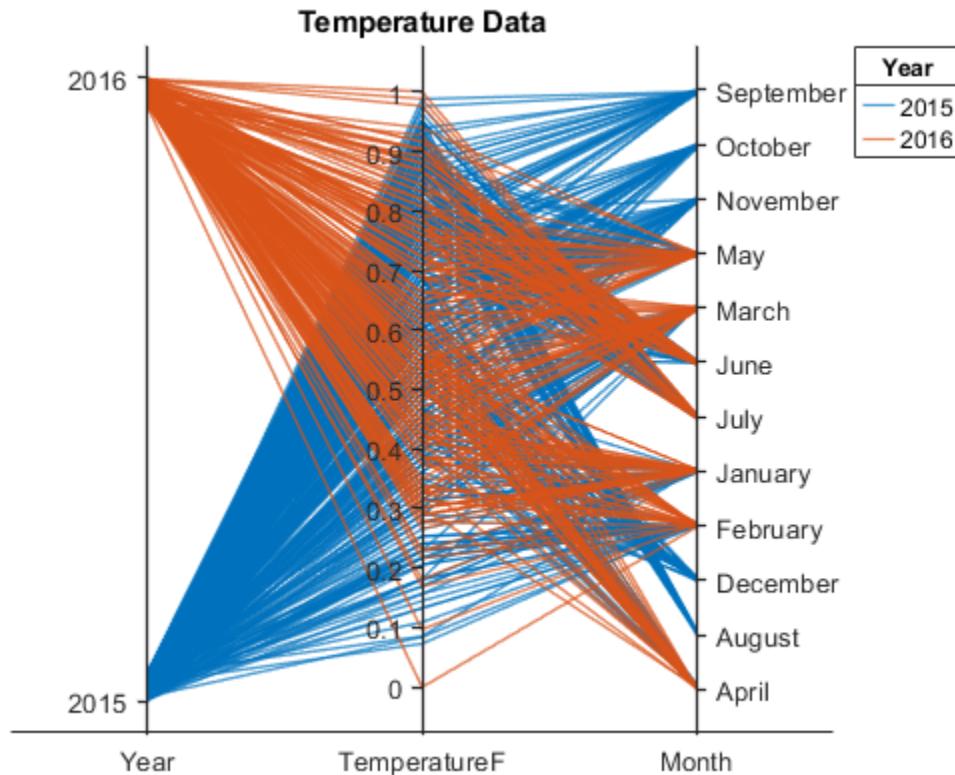
有关更多交互选项，请参阅“提示”。

选择部分坐标变量

显示 **p.SourceTable** 中的部分坐标变量，并通过设置 **p** 的 **CoordinateVariables** 属性来指定它们在绘图中的顺序。

具体操作是从绘图中删除 **Day** 变量，并将 **TemperatureF** 变量（源表中的第四列）显示为绘图中的第二个坐标轴。

```
p.CoordinateVariables = [1 4 2];
```



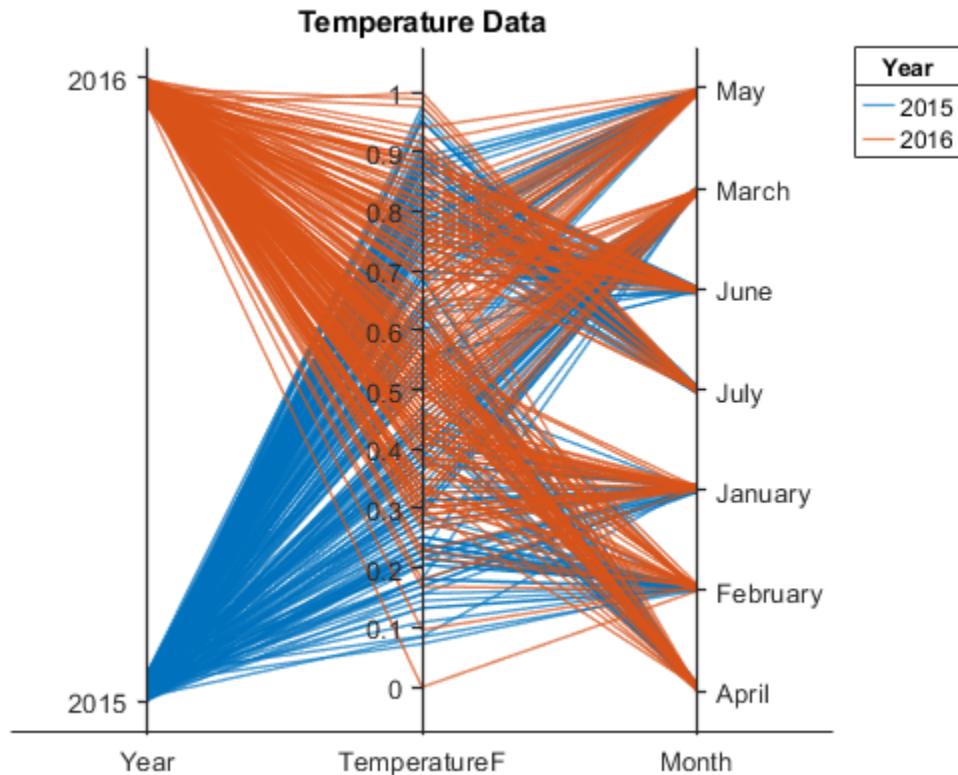
您还可以通过下列方法设置 **CoordinateVariables** 属性：变量名称的字符串或元胞数组，或者以 **true** 元素表示选定变量的逻辑向量。

修改坐标变量中的类别

在绘图中显示 Month 中类别的一个子集，并更改坐标标尺上的类别顺序。

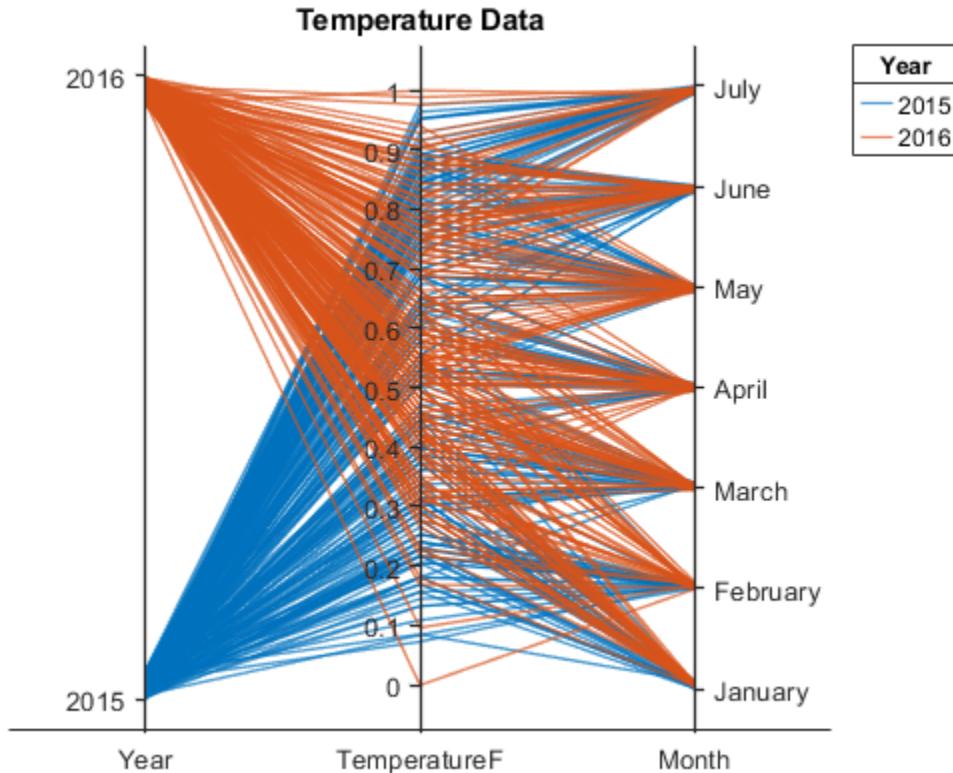
由于一些月份仅包含两年中的一年的数据，因此需要删除源表中与这些月份对应的行。MATLAB 会在源表更改后立即更新绘图。

```
uniqueMonth = {'September','October','November','December','August'};
uniqueMonthIdx = ismember(p.SourceTable.Month,uniqueMonth);
p.SourceTable(uniqueMonthIdx,:) = [];
```



通过更新源表，按时间顺序在 Month 坐标标尺上排列月份。

```
categoricalMonth = categorical(p.SourceTable.Month);
newOrder = {'January','February','March','April','May','June','July'};
orderMonth = reordercats(categoricalMonth,newOrder);
p.SourceTable.Month = orderMonth;
```



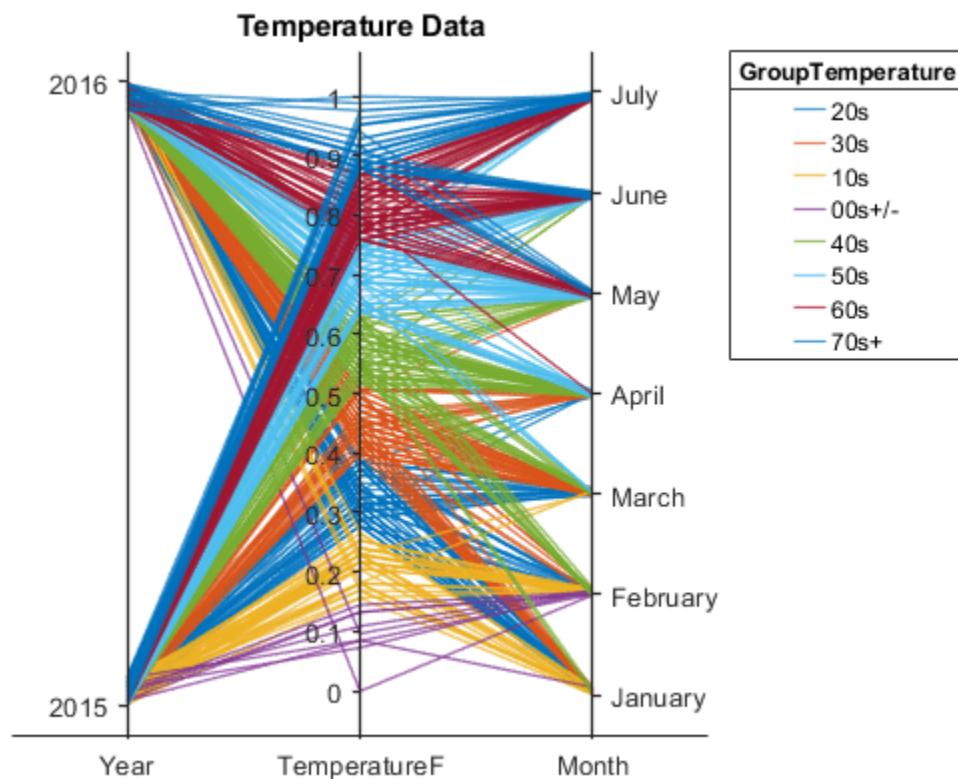
基于分 bin 后的值对绘图线条分组

为了更好地可视化每个月内的温度范围，您可以使用 `discretize` 来对温度数据分 bin，并使用分 bin 后的值对绘图中的线条进行分组。查看源表中的最低和最高温度。设置 bin 边界，将这些值包含在内。

```
min(p.SourceTable.TemperatureF)
ans = -3
max(p.SourceTable.TemperatureF)
ans = 80
binEdges = [-3 10:10:80];
bins = {'00s+/-','10s','20s','30s','40s','50s','60s','70s+'};
groupTemperature = discretize(p.SourceTable.TemperatureF, ...
    binEdges,'categorical',bins);
```

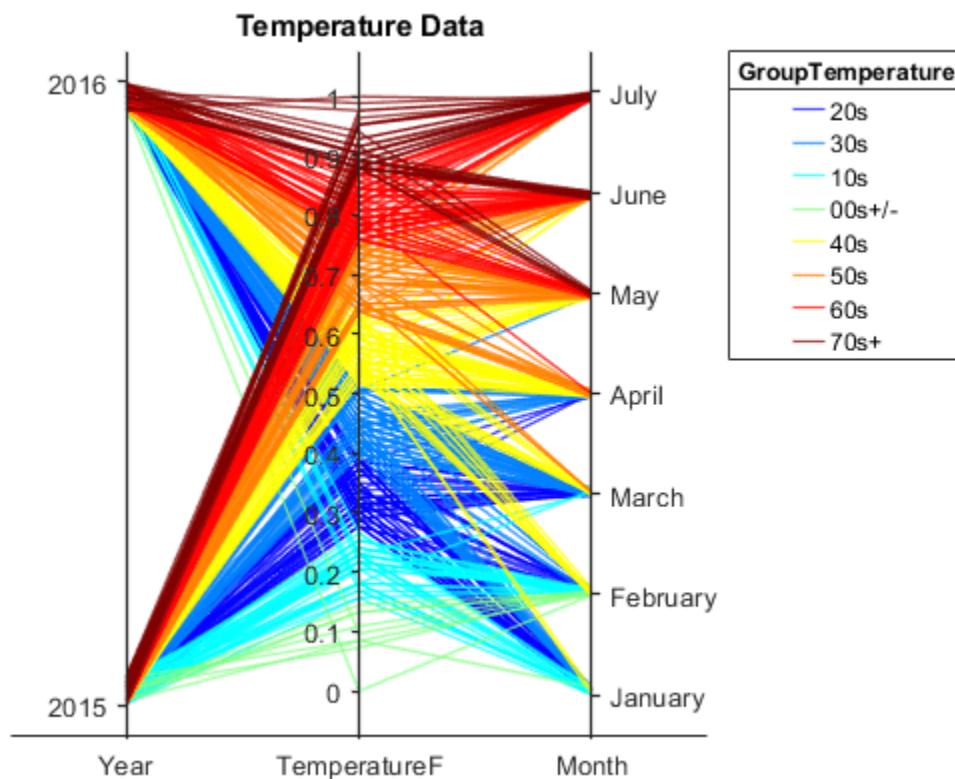
将分 bin 后的温度添加到源表中。根据分 bin 后的温度数据，对绘图中的线条进行分组。

```
p.SourceTable.GroupTemperature = groupTemperature;
p.GroupVariable = 'GroupTemperature';
```



由于 GroupTemperature 包含的类别超过 7 个，因此绘图中的一些组具有相同的颜色。通过设置 Color 属性，为每个组指定不同的颜色。

```
p.Color = jet(8);
```



另请参阅

函数

[parallelplot](#) | [table](#) | [readtable](#) | [reordercats](#) | [categorical](#) | [discretize](#)

属性

[ParallelCoordinatesPlot](#)

地理坐标区和地理图

- “使用纬度和经度数据创建地图” (第 6-2 页)
- “地理坐标区和地理图中的平移和缩放行为” (第 6-6 页)
- “地理气泡图概述” (第 6-7 页)
- “地理气泡图图例” (第 6-9 页)
- “在地理密度图上查看飓风轨迹数据” (第 6-10 页)
- “查看蜂窝塔位置的密度” (第 6-14 页)
- “自定义地理坐标区的布局” (第 6-20 页)
- “部署地理坐标区和地理图” (第 6-22 页)
- “使用地理气泡图属性” (第 6-23 页)
- “使用地理坐标区指定地图范围” (第 6-27 页)
- “访问用于地理坐标区和地理图的底图” (第 6-31 页)
- “基于表格数据创建地理气泡图” (第 6-37 页)

使用纬度和经度数据创建地图

如果您拥有与特定地理位置有关的数据，请使用地理坐标区或地理图，以在地图上可视化您的数据并提供视觉环境。例如，如果您拥有描述世界各地的海啸事件的数据，请在地理坐标区上绘制数据，并通过标记指示每个事件在地图上的位置。下面这些示例说明如何在地理坐标中创建线图、散点图、气泡图和密度图。

创建地理线图

在地图上，在西雅图和安克雷奇之间绘制一条线。指定每个城市的纬度和经度，然后使用 `geoplot` 函数绘制数据。使用线条设定 `'-*'` 自定义线条外观。使用 `geolimits` 调整地图的纬度和经度范围。使用 `geobasemap` 函数更改底图。

```
latSeattle = 47.62;
lonSeattle = -122.33;
latAnchorage = 61.20;
lonAnchorage = -149.9;

geoplot([latSeattle latAnchorage],[lonSeattle lonAnchorage],'*-*')
geolimits([45 62],[-149 -123])
geobasemap streets
```



创建地理散点图

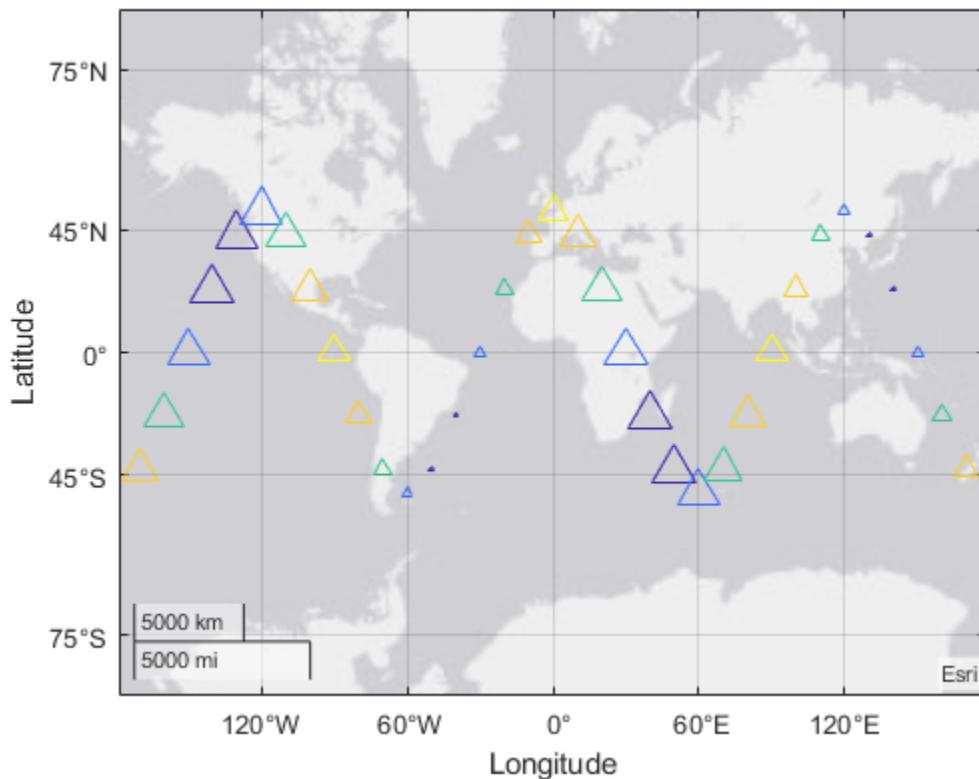
创建经纬度位置，并定义每个点处的值。使用 `geoscatter` 函数在地图上绘制值。该示例指定三角形作为标记，以大小和颜色表示值的变化。

```

lon = (-170:10:170);
lat = 50 * cosd(3*lon);
A = 101 + 100*(sind(2*lon));
C = cosd(4*lon);

geoscatte(lat,lon,A,C,'^')

```



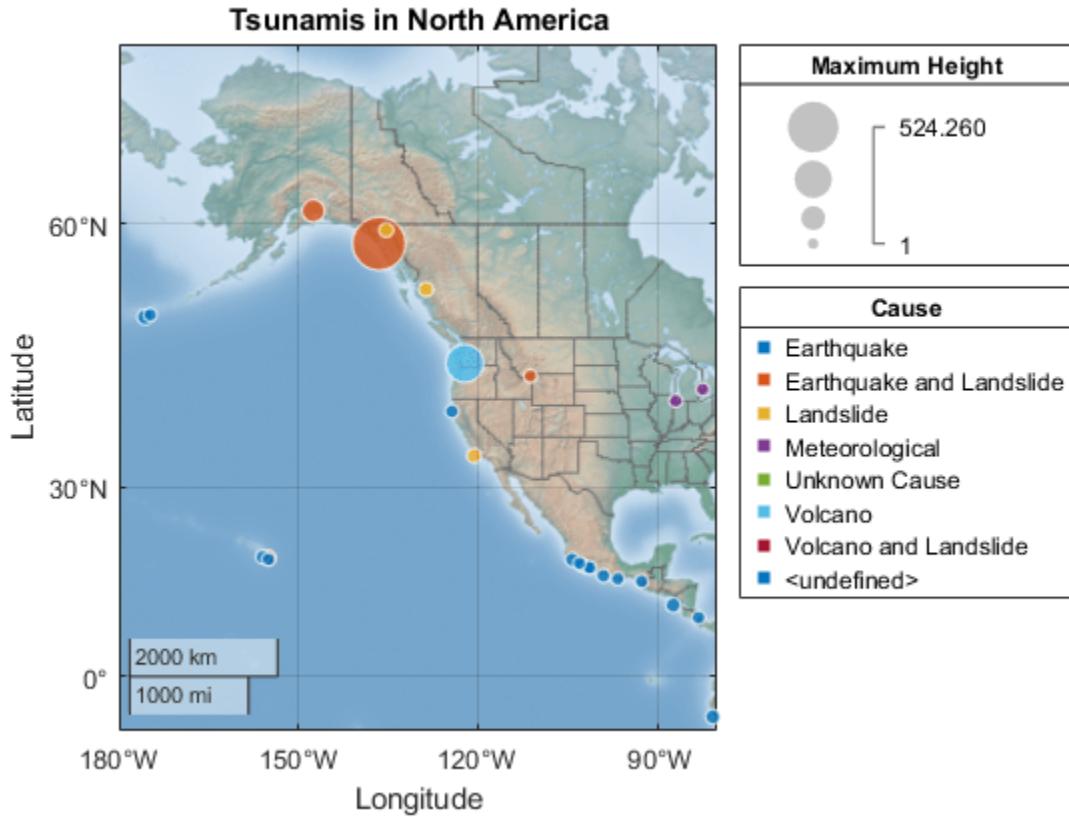
创建地理气泡图

根据海啸数据创建表。将一个值定义为一个分类值。使用 `geobubble` 函数在地图上绘制数据。该示例使用气泡大小指示海啸波浪的高度，使用颜色指示海啸的成因。

```

tsunamis = readtable('tsunamis.xlsx');
tsunamis.Cause = categorical(tsunamis.Cause);
figure
gb = geobubble(tsunamis,'Latitude','Longitude',...
    'SizeVariable','MaxHeight','ColorVariable','Cause');
geolimits([10 65],[-180 -80])
title 'Tsunamis in North America';
gb.SizeLegendTitle = 'Maximum Height';
geobasemap colorterrain

```

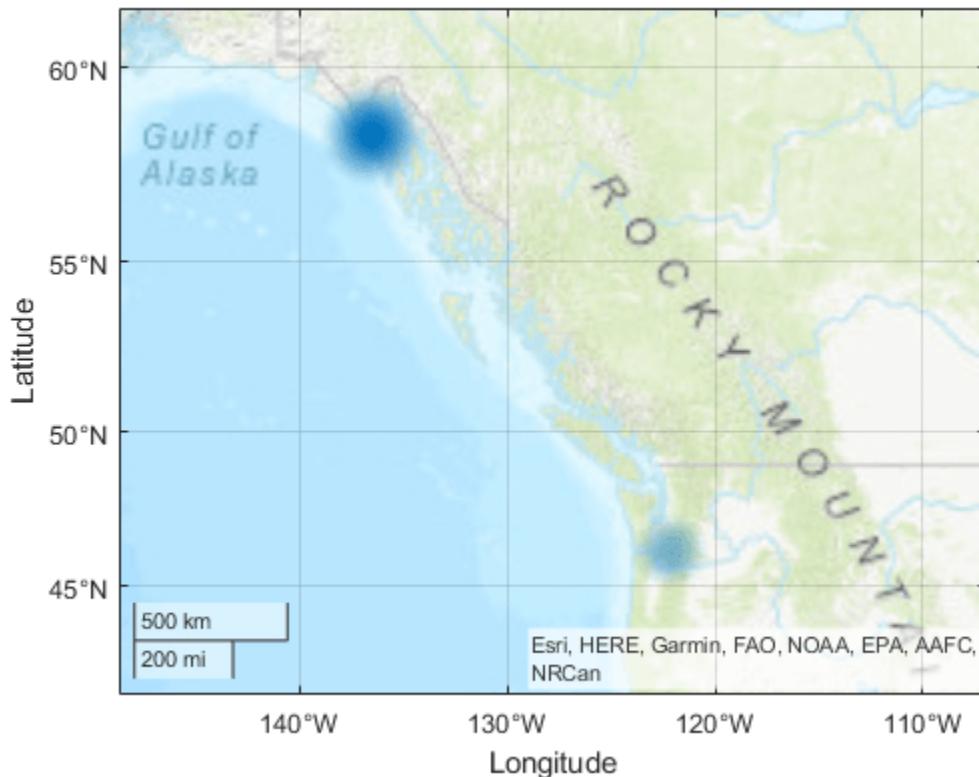


创建地理密度图

根据海啸数据创建表。使用 `geodensityplot` 函数绘制数据。

```
tsunamis = readtable('tsunamis.xlsx');
lat = tsunamis.Latitude;
lon = tsunamis.Longitude;
weights = tsunamis.MaxHeight;

geodensityplot(lat,lon,weights)
geolimits([41.2 61.4],[-148.6 -107.0])
geobasemap topographic
```



另请参阅

函数

[geoaxes](#) | [geoscatte](#)r | [geoplot](#) | [geodensityplot](#) | [geobubble](#)

属性

[GeographicBubbleChart Properties](#)

相关示例

- “使用地理气泡图属性” (第 6-23 页)
- “访问用于地理坐标区和地理图的底图” (第 6-31 页)
- “基于表格数据创建地理气泡图” (第 6-37 页)

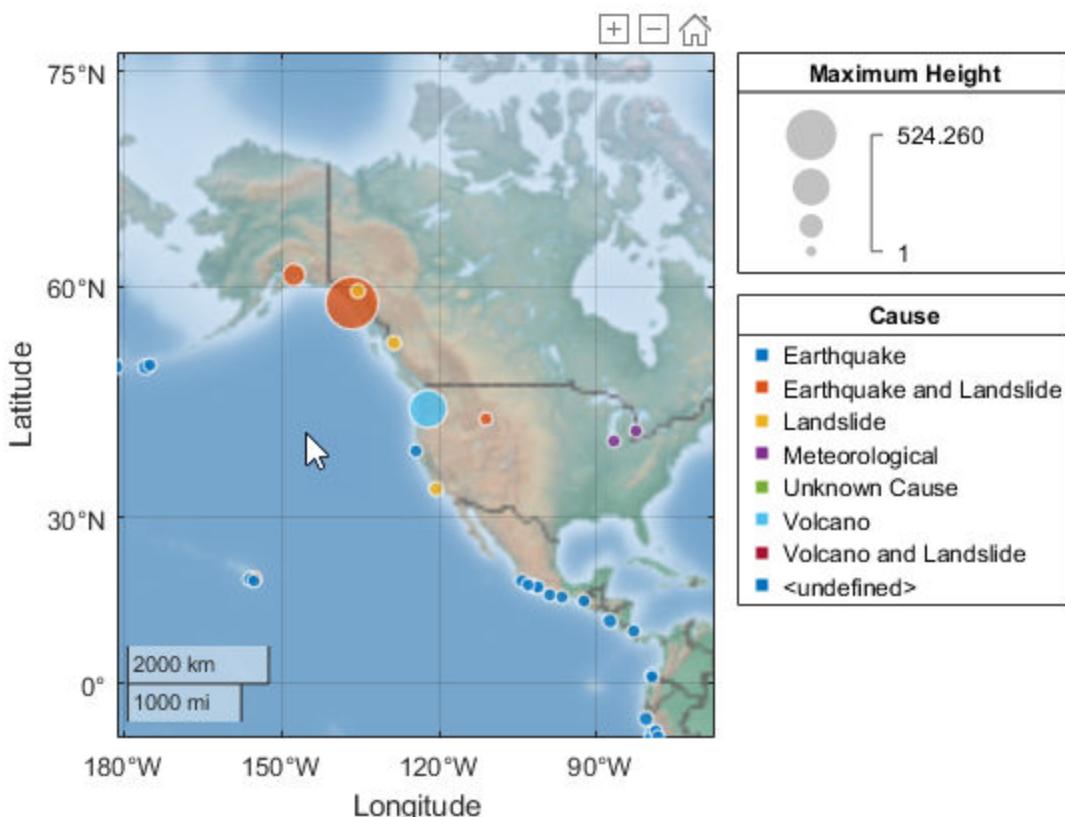
地理坐标区和地理图中的平移和缩放行为

地理坐标区或地理图中的底图是实时的，也就是说，您可以平移底图以查看其他地理位置，或者在地图上进行放大和缩小以更详细地查看某些区域。当您平移和缩放时，地图会更新。在地理坐标区和地理图上，平移和缩放功能默认情况下处于启用状态。

要平移地理坐标区或地理图中的底图，请使用箭头键，或者在地图上移动光标然后点击并拖动底图。可以在整个经度范围内横向连续平移地图。在纵向平移时，最多只能到南北纬 85 度。

要在地理坐标区或地理图中的地图上进行放大和缩小，您可以使用鼠标滚轮、触控板或键盘上的加号和减号键。

您还可以使用坐标区工具栏来放大、缩小或恢复地图的原始视图。将光标移到地图上时，将显示坐标区工具栏。将光标从地图上移开时，坐标区工具栏随即消失。



另请参阅

[geoaxes](#) | [geoscatter](#) | [geoplot](#) | [geodensityplot](#) | [geobubble](#)

相关示例

- “使用地理气泡图属性”（第 6-23 页）
- “访问用于地理坐标区和地理图的底图”（第 6-31 页）
- “基于表格数据创建地理气泡图”（第 6-37 页）

地理气泡图概述

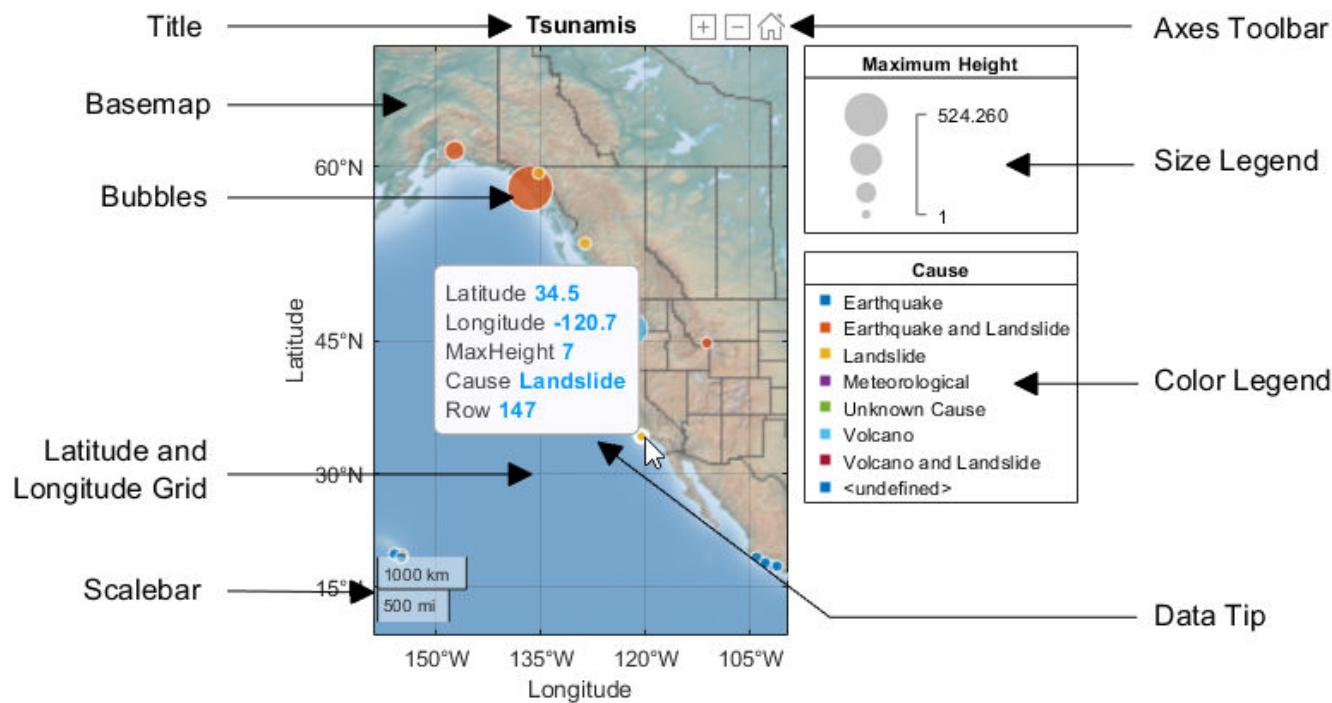
使用地图作为背景时，地理气泡图会在地图上由经度和纬度指定的位置将您的数据绘制为填充了颜色的圆（称为气泡）。您可以使用气泡的大小和颜色来指示在这些位置的数据值。

假设有描述世界各地的海啸事件的数据。将数据绘制在地理气泡图中，在称为底图的地图上用气泡标记每一次事件。您可以使用气泡大小指示海浪高度，使用气泡颜色指示原因。以地图为背景，可以立即看到海啸事件及其严重性。在地图上绘制数据是可视化数据的有效方式。

地理气泡图包括下列组成部分（如以下的图窗所示）：

地理气泡图组成部分

组成部分	说明
底图	指地理气泡图在其上绘制数据的地图。有关详细信息，请参阅“访问用于地理坐标区和地理图的底图”（第 6-31 页）。
气泡	标记地图位置并通过大小和颜色表达其他信息的符号。
数据提示	弹出的小窗口，其中包含有关气泡的信息，例如纬度和经度。
装饰元素	地理气泡图的描述性可视元素，如纬度和经度网格，以及说明在地图上如何表示距离的比例尺。当您在地图上放大和缩小时，地理气泡图会更新这些元素。使用地理气泡图属性来控制这些元素的可见性，如 <code>ScalebarVisible</code> 属性。
图例	解释气泡大小和气泡颜色含义的表格信息。有关详细信息，请参阅“地理气泡图图例”（第 6-9 页）。
标题	显示在图的顶部，与其他 MATLAB 图窗类似。您可以使用地理气泡图 <code>Title</code> 属性或 <code>title</code> 命令来指定标题。
坐标区工具栏	一组控件，可让您在地图上进行放大或缩小，或者返回到地图的原始视图。有关详细信息，请参阅“地理坐标区和地理图中的平移和缩放行为”（第 6-6 页）。



另请参阅

[geobubble | GeographicBubbleChart Properties](#)

相关示例

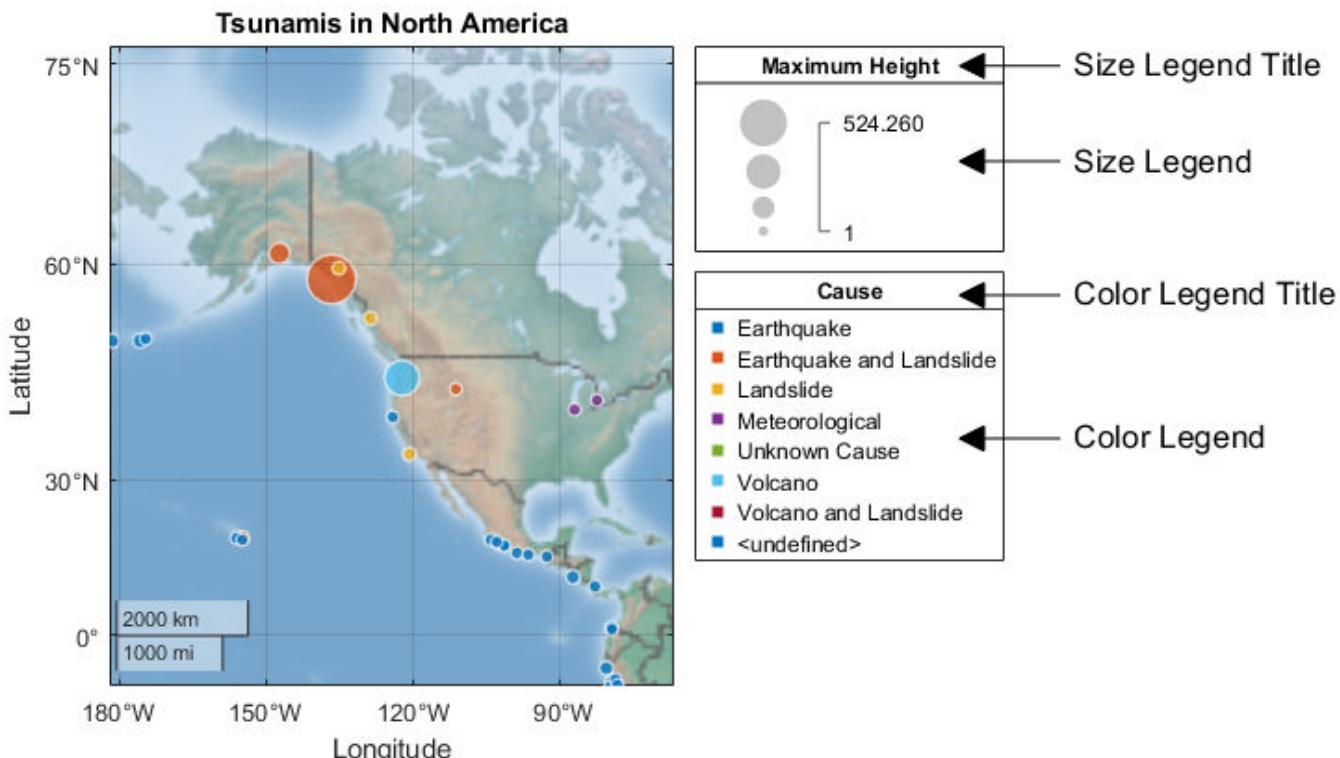
- “使用地理气泡图属性” (第 6-23 页)
- “访问用于地理坐标区和地理图的底图” (第 6-31 页)
- “基于表格数据创建地理气泡图” (第 6-37 页)

地理气泡图图例

当您使用 **SizeData** 创建地理气泡图时，图中会包含一个气泡大小图例，用来说明如何用气泡的大小来表示数据值的大小。图例包括四个从小到大的不同气泡大小采样。您可以使用 **BubbleWidthRange** 属性指定最小和最大气泡的宽度。图例用关联的数值在图例中标记最小和最大的气泡。图例从 **SizeLimits** 属性中获取这些值。如果您直接指定 **SizeData**，则图例无标题。您可以使用 **SizeLegendTitle** 属性指定图例的标题。如果要为大小数据指定表变量，则图例将使用变量的名称作为大小图例的标题。图例包括四个从小到大的不同气泡大小采样。可以使用 **BubbleWidthRange** 属性指定最小和最大气泡的宽度。图例用关联的数值在图例中标记最小和最大的气泡。

同样，如果您使用 **ColorData** 创建地理气泡图，则图中会包含一个颜色图例，用来显示如何用不同的气泡颜色表示分类数据。图例包括各种颜色，每种颜色用与之相关联的类别进行标记。如果您直接指定 **ColorData**，则图例无标题。您可以使用 **ColorLegendTitle** 属性指定图例的标题。如果要为颜色数据指定表变量，则图例将使用变量的名称作为颜色图例的标题。

下图显示了地理气泡图的大小和颜色图例。



另请参阅

[geobubble | GeographicBubbleChart Properties](#)

相关示例

- “使用地理气泡图属性”（第 6-23 页）
- “访问用于地理坐标区和地理图的底图”（第 6-31 页）
- “基于表格数据创建地理气泡图”（第 6-37 页）

在地理密度图上查看飓风轨迹数据

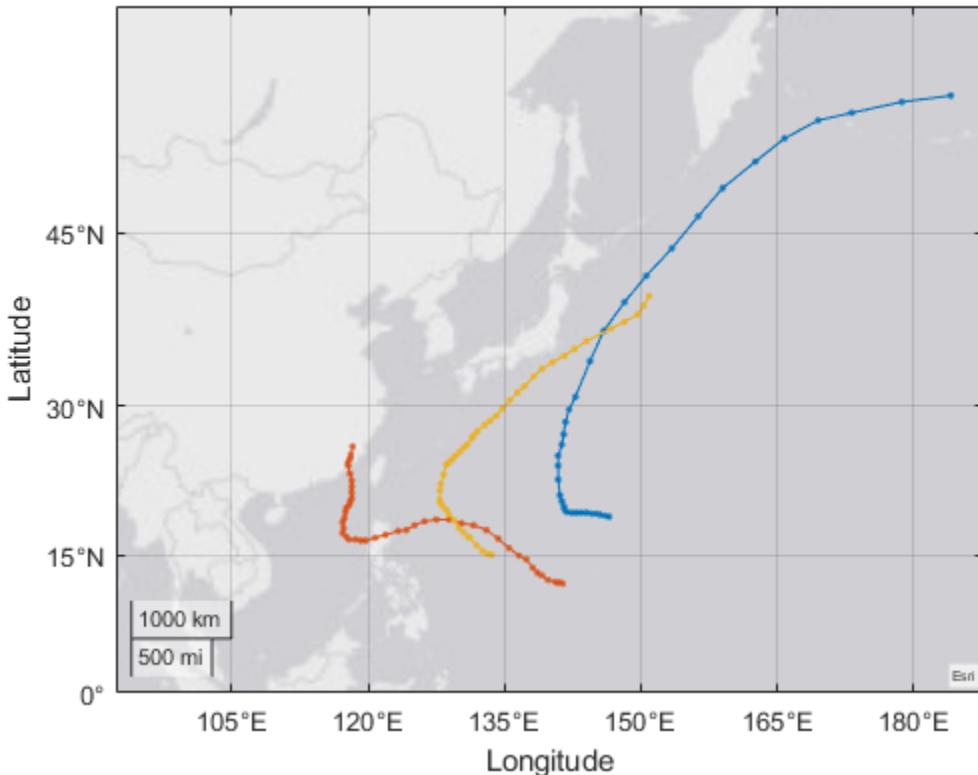
此示例说明如何在地理密度图上查看飓风轨迹数据。该数据记录 2007-2017 之间 11 年的飓风观测结果。

加载飓风轨迹数据。该数据由日本气象厅制作，它以 6 小时为间隔记录了飓风的位置、气压（以百帕为单位）和风速（以节为单位）。表中的每一行代表特定飓风的一次观察记录，由名称和 ID 号进行标识。

```
load cycloneTracks
```

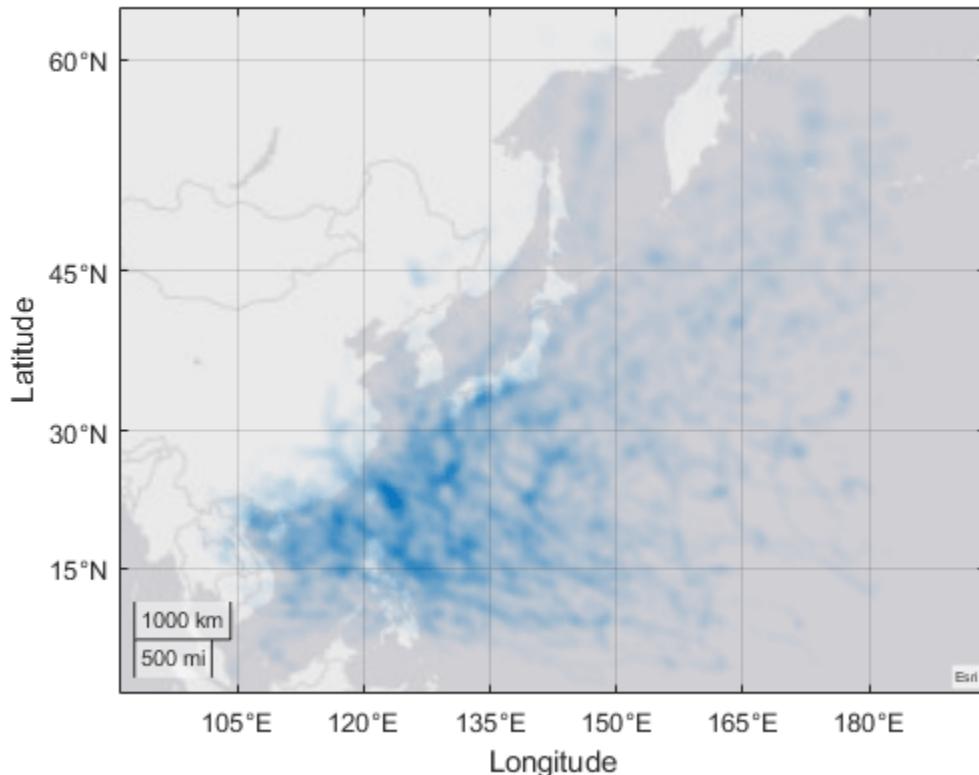
要理解这些数据，请使用 `geoplot` 函数绘制三个飓风的轨迹。获取三个飓风的数据记录，由 ID 号和名称进行标识。每一条观察记录都提供了纬度和经度。通过启用 `hold`，在一个地图上绘制所有三个飓风的轨迹。

```
figure
latMalakas = cycloneTracks.Latitude(cycloneTracks.ID == 1012);
lonMalakas = cycloneTracks.Longitude(cycloneTracks.ID == 1012);
geoplot(latMalakas,lonMalakas,'.-')
geolimits([0 60],[100 180])
hold on
latMegi = cycloneTracks.Latitude(cycloneTracks.ID == 1013);
lonMegi = cycloneTracks.Longitude(cycloneTracks.ID == 1013);
geoplot(latMegi,lonMegi,'.-')
latChaba = cycloneTracks.Latitude(cycloneTracks.ID == 1014);
lonChaba = cycloneTracks.Longitude(cycloneTracks.ID == 1014);
geoplot(latChaba,lonChaba,'.-')
```



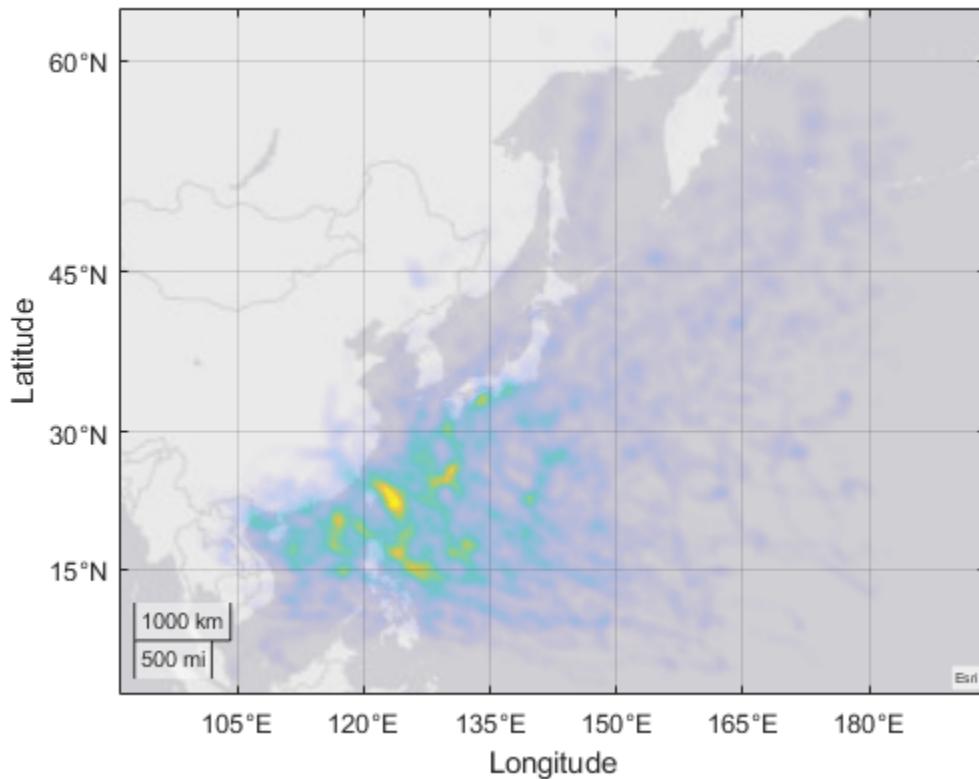
使用 `geodensityplot` 查看在这 11 年期间跟踪的所有飓风的密度。此图查看的不是特定飓风的轨迹，而是所有飓风在每个点的所有记录。`geodensityplot` 使用各个位置的占比来计算累积的概率分布曲面。曲面透明度随密度而变化。

```
figure  
latAll = cycloneTracks.Latitude;  
lonAll = cycloneTracks.Longitude;  
geodensityplot(latAll,lonAll)
```



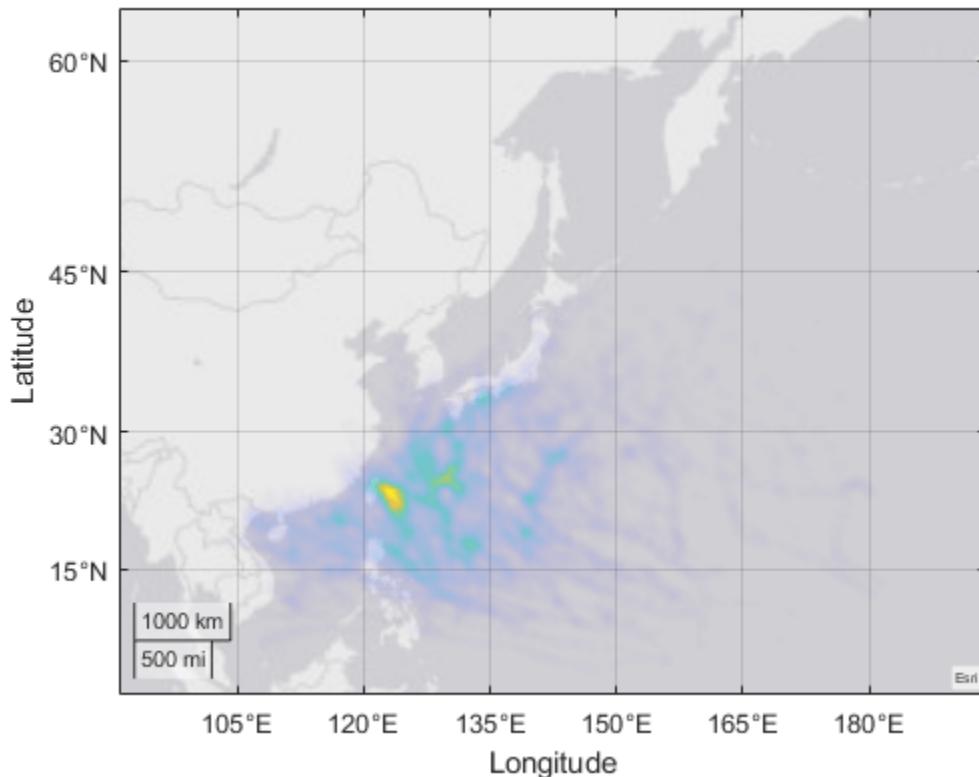
默认情况下，`geodensityplot` 使用一种颜色表示所有密度值，使用透明度表示密度变化。也可以在 `geodensityplot` 中使用多种颜色来表示不同密度的区域。要实现此目的，请设置 '`FaceColor`' 属性。

```
geodensityplot(latAll,lonAll,'FaceColor','interp')
```



密度图可以对各个数据点应用权重。权重乘以各个点在密度曲面上的占比。

```
windspeedAll = cycloneTracks.WindSpeed;  
geodensityplot(latAll,lonAll,windspeedAll,'FaceColor','interp')
```



参考资料：此飓风轨迹数据来自日本气象厅制作的 RSMC 最佳轨迹数据 (https://www.jma.go.jp/jma/jma-eng/jma-center/rsmc-hp-pub-eg/RSMC_HP.htm)，由 MathWorks 修改后用于此示例。

另请参阅

[geodensityplot](#) | [DensityPlot Properties](#)

相关示例

- “访问用于地理坐标区和地理图的底图” (第 6-31 页)

查看蜂窝塔位置的密度

此示例说明如何使用地理密度图来查看加利福尼亚地区蜂窝塔位置的密度。

加载蜂窝塔位置数据

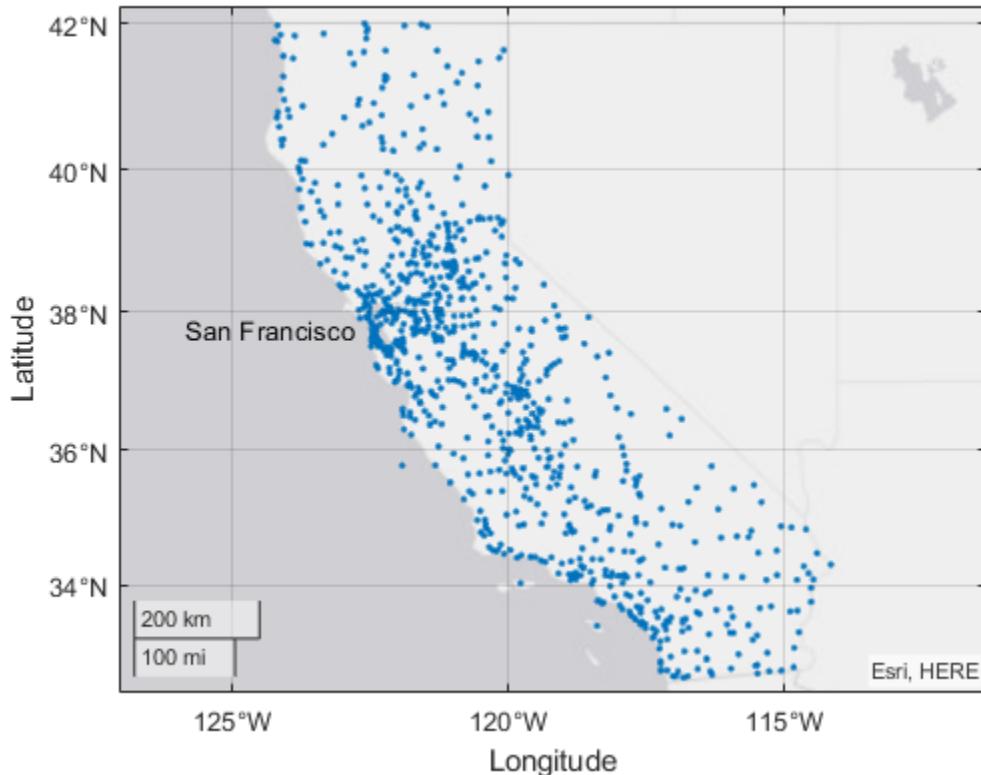
将一个包含蜂窝塔位置数据的表加载到工作区中。下表中包含的字段通过纬度和经度来标识蜂窝塔的位置，并标识塔的类型。

```
load cellularTowers
```

以地理散点图的方式查看数据

使用 `geoscatter` 函数绘制蜂窝塔数据。在下图中，旧金山附近有一些明显的区域因为塔的数量太密集而不能用散点图表示。

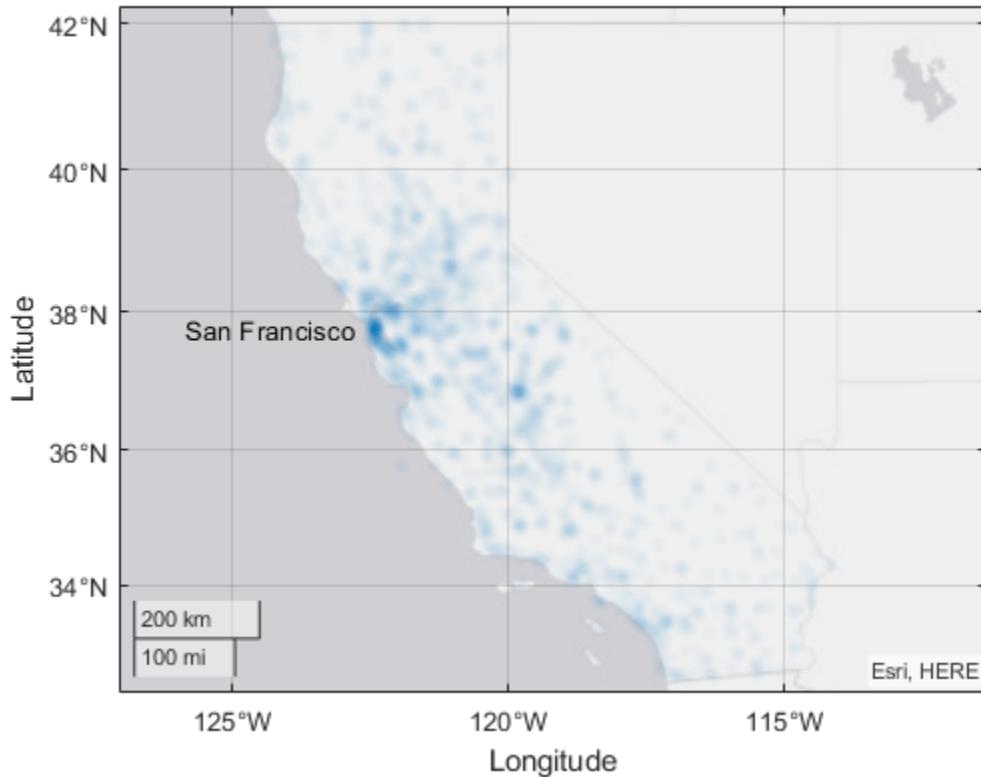
```
lat = cellularTowers.Latitude;
lon = cellularTowers.Longitude;
geoscatter(lat,lon,'.')
text(gca,37.75,-122.75,'San Francisco','HorizontalAlignment','right')
```



以地理密度图的方式查看数据

旧金山地区的蜂窝塔密集区域可以使用 `geodensityplot` 来显示。

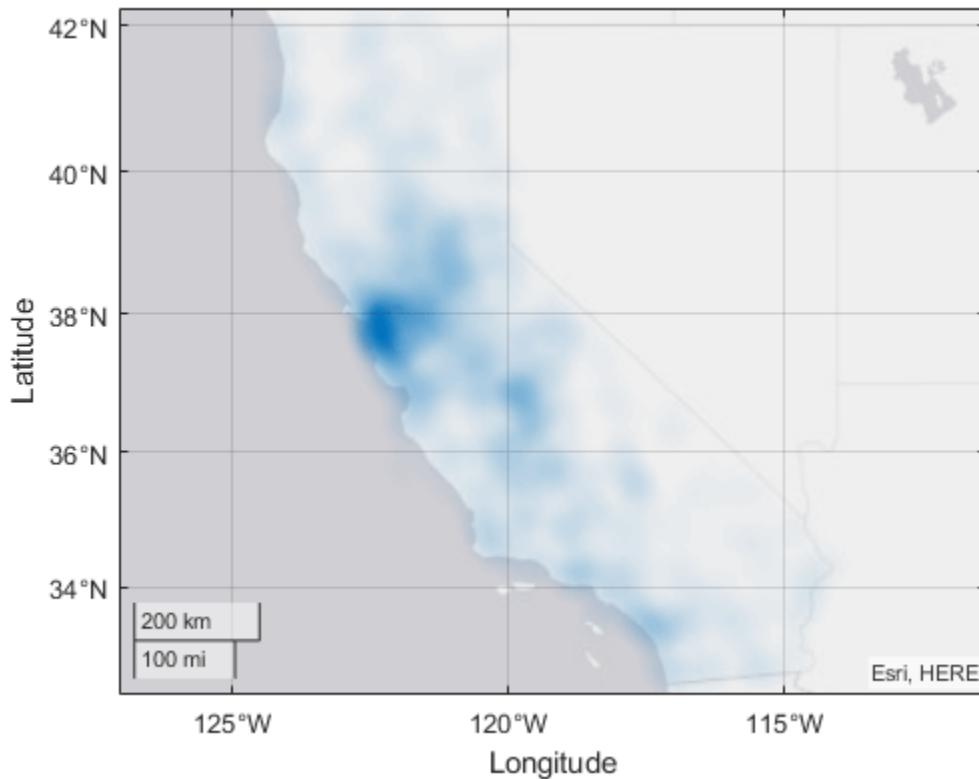
```
geodensityplot(lat,lon)
text(gca,37.75,-122.75,'San Francisco','HorizontalAlignment','right')
```



创建指定半径的密度图

创建地理密度图时，默认情况下，密度图使用纬度和经度数据自动选择半径值。可以使用 **Radius** 属性手动选择以米为单位的半径。

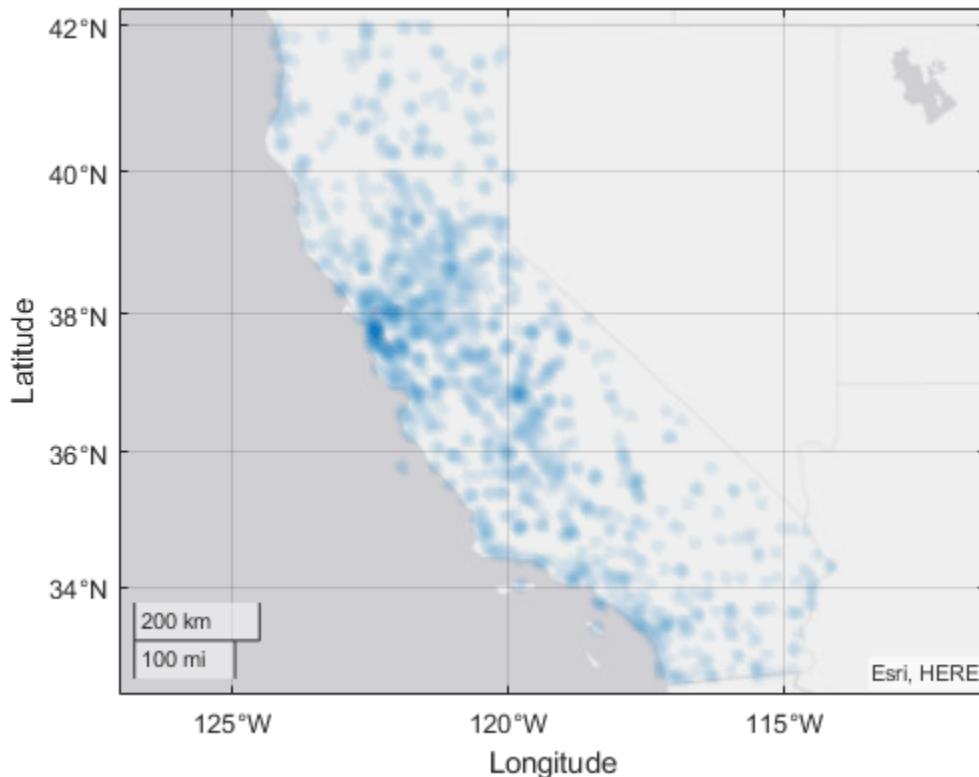
```
radiusInMeters = 50e3; % 50 km  
geodensityplot(lat,lon,'Radius',radiusInMeters)
```



使用坐标区属性调整透明度

设置为 'interp' 时，密度图的 FaceAlpha 和 FaceColor 属性分别使用底层地理坐标区的 Alphamap 和 Colormap 属性。更改 Alphamap 将更改密度值与颜色强度之间的映射。

```
geodensityplot(lat,lon)  
alphamap(normalize((1:64).^0.5,'range'))
```



地理坐标区上的 `AlphaScale` 属性也可用于更改透明度。如果要显示所有有密度的位置，而不是只突出显示最密集的区域，此属性尤其有用。

```
figure
dp = geodensityplot(lat,lon)

dp = DensityPlot with properties:
  FaceColor: [0 0.4470 0.7410]
  FaceAlpha: 'interp'
  LatitudeData: [1×1193 double]
  LongitudeData: [1×1193 double]
  WeightData: [1×0 double]
  Radius: 1.8291e+04
```

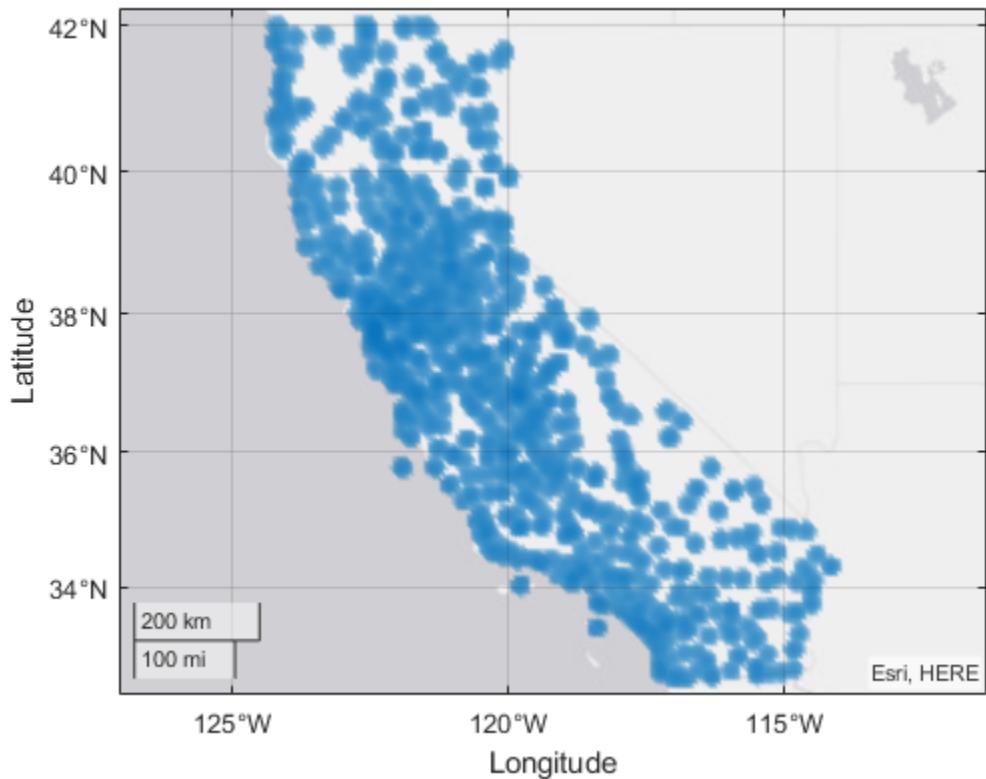
Show all properties

```
gx = gca

gx = GeographicAxes with properties:
  Basemap: 'streets-light'
  Position: [0.1300 0.1100 0.7750 0.8150]
  Units: 'normalized'
```

Show all properties

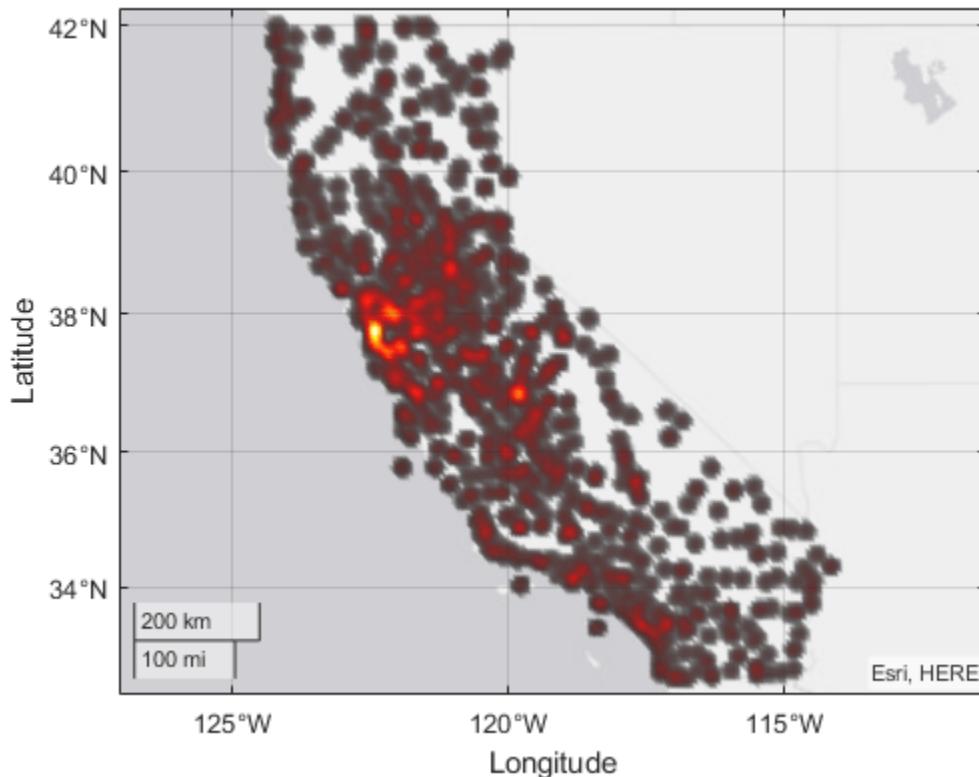
```
gx.AlphaScale = 'log';
```



使用 DensityPlot 对象属性指定颜色

添加颜色。

```
dp.FaceColor = 'interp';
colormap hot
```



另请参阅

[geodensityplot](#) | DensityPlot Properties

相关示例

- “访问用于地理坐标区和地理图的底图” (第 6-31 页)

自定义地理坐标区的布局

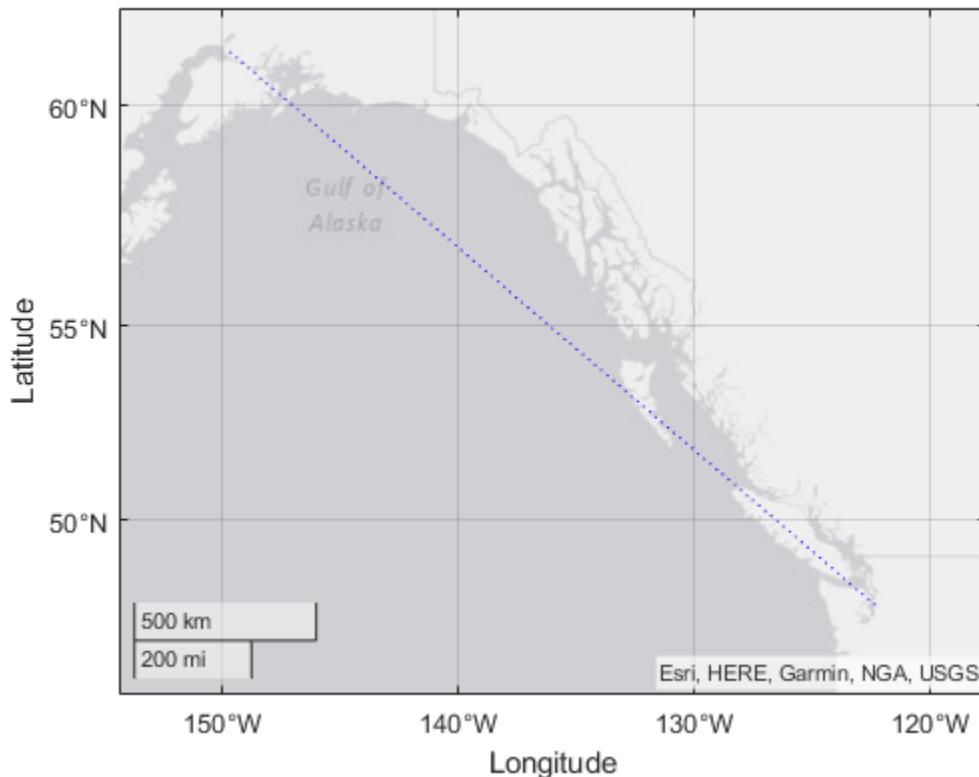
通过修改地理坐标区的属性来自定义地理坐标区的布局。

在地图上的两个点之间绘制一条直线。使用西雅图和安克雷奇的坐标指定线的端点。以度为单位指定纬度和经度。

```
latSeattle = 47.62;
lonSeattle = -122.33;
latAnchorage = 61.20;
lonAnchorage = -149.9;
```

使用 `geoplot` 绘制数据。使用线条设定 '`b:`' 自定义线条外观。使用 `geolimits` 调整地图的纬度和经度范围。

```
geoplot([latSeattle latAnchorage],[lonSeattle lonAnchorage],'b:')
geolimits([45 62],[-149 -123])
```



自定义坐标区的布局。通过修改 `Grid`、`Position` 和 `TickDir` 属性来关闭网格，拉伸网格以占据整个图窗并关闭刻度线。

```
gx = gca;
gx.Grid = 'off';
gx.TickDir = 'out';
gx.Position = gx.OuterPosition;
```



另请参阅

[geoplot](#)

相关示例

- “访问用于地理坐标区和地理图的底图”（第 6-31 页）

部署地理坐标区和地理图

您可以使用 MATLAB Compiler™ 部署使用地理坐标区或地理图的 MATLAB 应用程序。根据所用的底图，所部署的应用程序中的地图交互与 MATLAB 会话中的地图交互相同，也就是说，该地图是“实时”的，可以平移和缩放。

App 用法	行为
部署的应用程序仅使用 'darkwater' 底图。	'darkwater' 底图包含在 MATLAB 中，不需要访问 Internet。
部署的应用程序提供底图选择。	该应用程序通过 Internet 访问底图。
部署的应用程序提供底图选择，不需要 Internet 连接。	您必须下载 MATLAB 底图数据附加功能，并将其包括在部署的应用程序包中。有关下载底图的详细信息，请参阅“访问用于地理坐标区和地理图的底图”（第 6-31 页）。

注意 默认情况下，部署工具会预先选择所有已下载的 MATLAB Basemap Data 附加功能，以包含在部署的应用程序包中。不要选中所有底图。只选择您希望应用程序用户看到的底图。如果在部署的应用程序包中包括所有 MATLAB 底图数据附加功能，创建的文件可能会超过文件系统限制。

另请参阅

[geoaxes](#) | [geoscatte](#)r | [geoplot](#) | [geodensityplot](#) | [geobubble](#)

相关示例

- “访问用于地理坐标区和地理图的底图”（第 6-31 页）

使用地理气泡图属性

本节内容

- “控制气泡大小”（第 6-23 页）
- “控制气泡颜色”（第 6-25 页）

本主题介绍可使用地理气泡图属性执行的一些常见任务。

控制气泡大小

您可以在地理气泡图中使用气泡大小来表达数据的可量化特征。例如，对于莱姆病样本数据，您可以使用气泡大小来可视化显示新英格兰各郡的病例数。地理气泡图的以下属性共同控制图上气泡的大小：

- **SizeData**
- **SizeVariable**
- **SizeLimits**
- **BubbleWidthRange**

SizeData 属性指定要在图上绘制的数据。**SizeData** 必须是与纬度和经度向量大小相同的数值数据向量，或是标量。指定大小数据的另一种方法是向 **geobubble** 函数传递一个表作为它的第一个参数，并指定用作大小数据的表变量的名称。您可以使用 **SizeVariable** 属性来指定此表变量。当您使用表变量指定大小数据时，**geobubble** 将此变量的值存储在 **SizeData** 属性中，并将该属性设置为只读。如果您没有指定 **SizeData**，**geobubble** 将使用大小完全相同的气泡在地图上的地理位置绘图。

geobubble 对 **BubbleWidthRange** 属性设置的范围内的 **SizeData** 值进行线性缩放，从而确定每个气泡的大小（直径）。**BubbleWidthRange** 是一个二元素向量，它指定以磅为单位的最小气泡直径和最大气泡直径。默认情况下，**BubbleWidthRange** 将气泡直径的范围设置在 5 磅和 20 磅之间。您可以指定的气泡直径最小为 1 磅，最大为 100 磅。

使用 **SizeLimits** 属性来控制 **SizeData** 和 **BubbleWidthRange** 之间的映射。默认情况下，**SizeLimits** 属性指定数据范围的极值。例如，当使用 **Cases2010** 变量作为 **SizeVariable** 时，莱姆病样本数据的 **SizeLimits** 默认值为 [0 514]。

指定大小数据时，地理气泡图包含描述气泡大小与数据的映射关系的图例。**geobubble** 将 **SizeLimits** 属性中的值用作图例的上界和下界。指定表变量时，**geobubble** 使用变量名称作为大小图例的标题。

缩小地理气泡图中的气泡

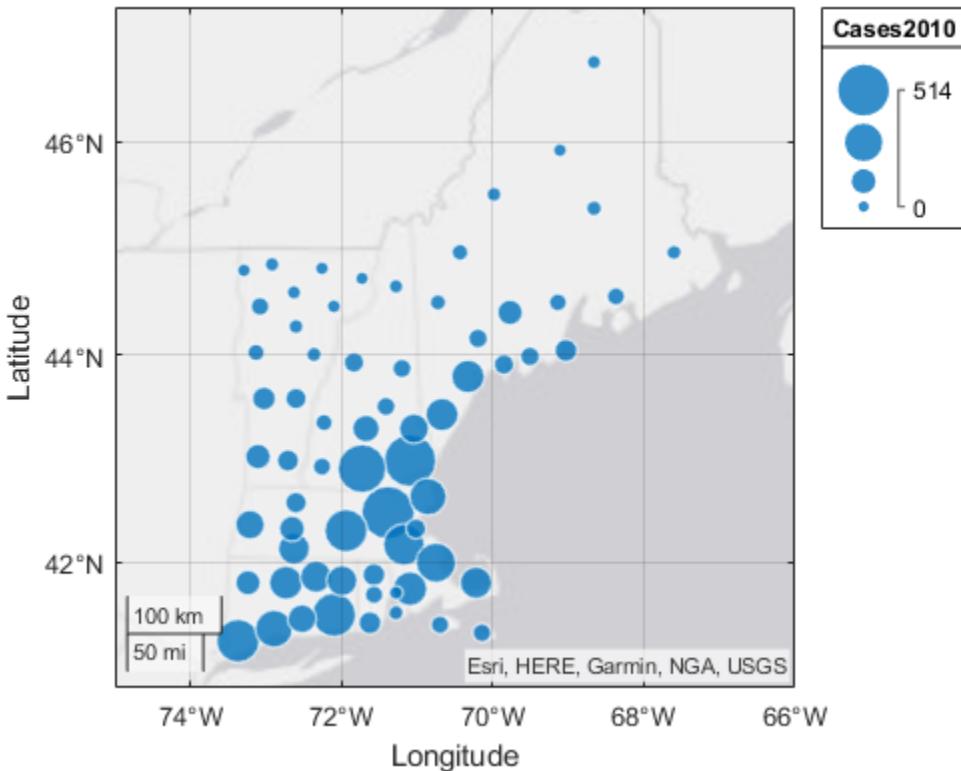
此示例说明如何使用 **BubbleWidthRange** 属性缩小地理气泡图中气泡的大小。（您还可以通过调整地理气泡图图窗的大小来减少重叠。）

将莱姆病样本数据读入工作区中。

```
counties = readtable('counties.xlsx');
```

使用表中的纬度、经度和病例数据创建地理气泡图。使用 **geolimits** 函数调整图的范围。

```
gb = geobubble(counties,'Latitude','Longitude','SizeVariable','Cases2010');
geolimits(gb,[41 47],[-75 -66])
```



查看地理气泡图的 `SizeData` 和 `SizeLimits` 属性的值。

```
size_data_values = gb.SizeData;
size_data_values(1:15)
```

```
ans = 15×1
```

```
331
187
88
125
240
340
161
148
38
4
:
```

```
size_limits = gb.SizeLimits
```

```
size_limits = 1×2
```

```
0 514
```

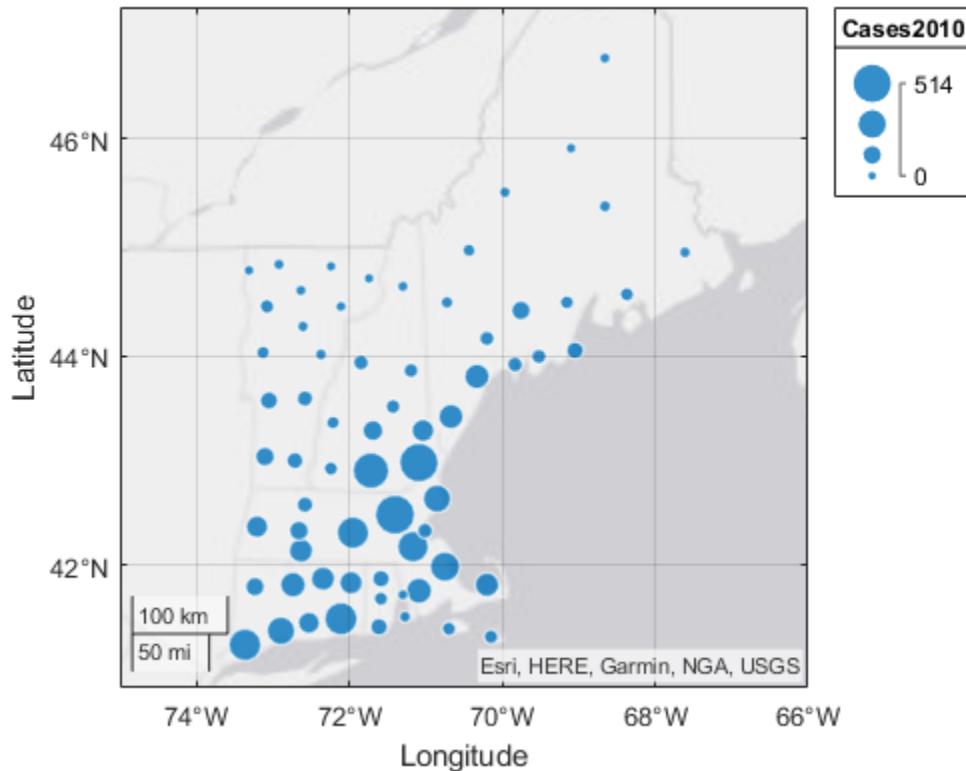
使用 `BubbleWidthRange` 属性缩小气泡以避免重叠。首先查看属性的初始设置。

```
default_width_range = gb.BubbleWidthRange
```

```
default_width_range = 1×2
```

```
5 20
```

```
gb.BubbleWidthRange = [4 15];
```



控制气泡颜色

您可以使用地理气泡图中的气泡颜色，根据数据类别对气泡着色。例如，在莱姆病样本数据中，您可以将新英格兰各郡的莱姆病的严重程度分为高、中、低三个等级。地理气泡图的以下属性共同控制图上气泡的颜色：

- **ColorData**
- **ColorVariable**
- **BubbleColorList**

ColorData 属性指定用来控制图中气泡颜色的数据。**ColorData** 必须是大小与纬度和经度相同的分类数据的向量。指定颜色数据的另一种方法是向 `geobubble` 传递一个表作为函数的第一个参数，并为该表指定用作颜色数据的表变量的名称。您可以使用 **ColorVariable** 属性来指定此表变量。`geobubble` 将该表变量的值存储在 **ColorData** 属性中，并将该属性设置为只读。

如果您的数据最初不包含 `categorical` 变量，则您可以创建该变量。例如，莱姆病样本数据不包括分类变量。创建此类型的变量的一种方法是使用 `discretize` 函数。使用病例发生数据 `cases2010`，根据病例发

生次数创建三个类别，分别将其描述为低、中、高三个严重级别。以下代码根据病例发生数据创建名为 **Severity** 的分类变量。

```
Severity = discretize(counties.Cases2010,[0 50 100 550],...  
'categorical', {'Low', 'Medium', 'High'});
```

BubbleColorList 属性控制地理气泡图中气泡所用的颜色。该值是一个 $m \times 3$ 数组，其中每行是一个 RGB 颜色三元组。默认情况下，**geobubble** 使用包含七种颜色的颜色集。如果您有七个以上的类别，会循环重复使用这些颜色。要更改使用的颜色，请使用其他 MATLAB 颜色图函数之一，例如 **parula** 或 **jet**，或指定自定义颜色列表。

另请参阅

[discretize](#) | [geolimits](#) | [geobubble](#) | [GeographicBubbleChart Properties](#)

相关示例

- “部署地理坐标区和地理图”（第 6-22 页）
- “访问用于地理坐标区和地理图的底图”（第 6-31 页）
- “地理气泡图概述”（第 6-7 页）
- “基于表格数据创建地理气泡图”（第 6-37 页）

使用地理坐标区指定地图范围

地理坐标区或地理图将底图的纬度和经度范围设置为包含您数据中的所有点。当您通过调整图窗窗口大小来调整图大小时，这些地图范围不会改变，但在适应坐标区或图本身的变化时，地图范围会改变。缩放或平移时，地图范围会发生变化。地理坐标区和地理图支持与地图范围相关的属性。一些是只读属性，用于提供信息。

- **LatitudeLimits** - 返回当前纬度范围（只读）。
- **LongitudeLimits** - 返回当前经度范围（只读）。
- **MapCenter** - 返回或设置当前地图中心点。
- **ZoomLevel** - 返回或设置当前地图缩放级别。

获取当前纬度和经度范围的一条捷径是调用 **geolimits** 函数。您还可以使用 **geolimits** 函数来设置纬度和经度范围。当您要创建与现有坐标区或图具有相同地图范围的地理坐标区或地理图时，请使用 **geolimits** 函数。检索现有坐标区或图的范围，并使用 **geolimits** 设置新坐标区或图的范围。

注意 您可以指定超出大致范围 [-85 85] 的纬度，但底图图块不会延伸到此范围之外。不过，这些值通常不可见的，除非您使用 **MapCenter** 和 **ZoomLevel** 属性控制地图范围。此外，非常接近 90 度和 -90 度的数据点也永远不可见，因为它们对应于垂直方向上无穷或接近无穷的值。

在指定的范围内居中显示几个地理气泡图

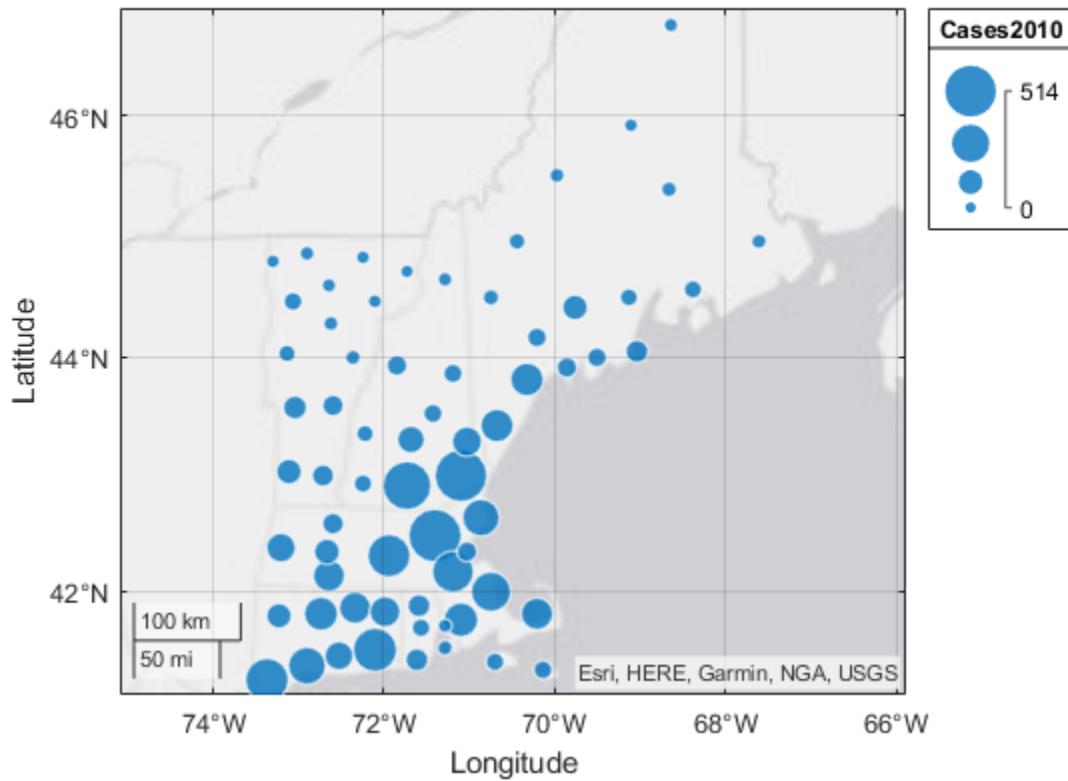
此示例说明如何使用相同的地图范围创建两个地理气泡图。

将莱姆病样本数据读入工作区中。

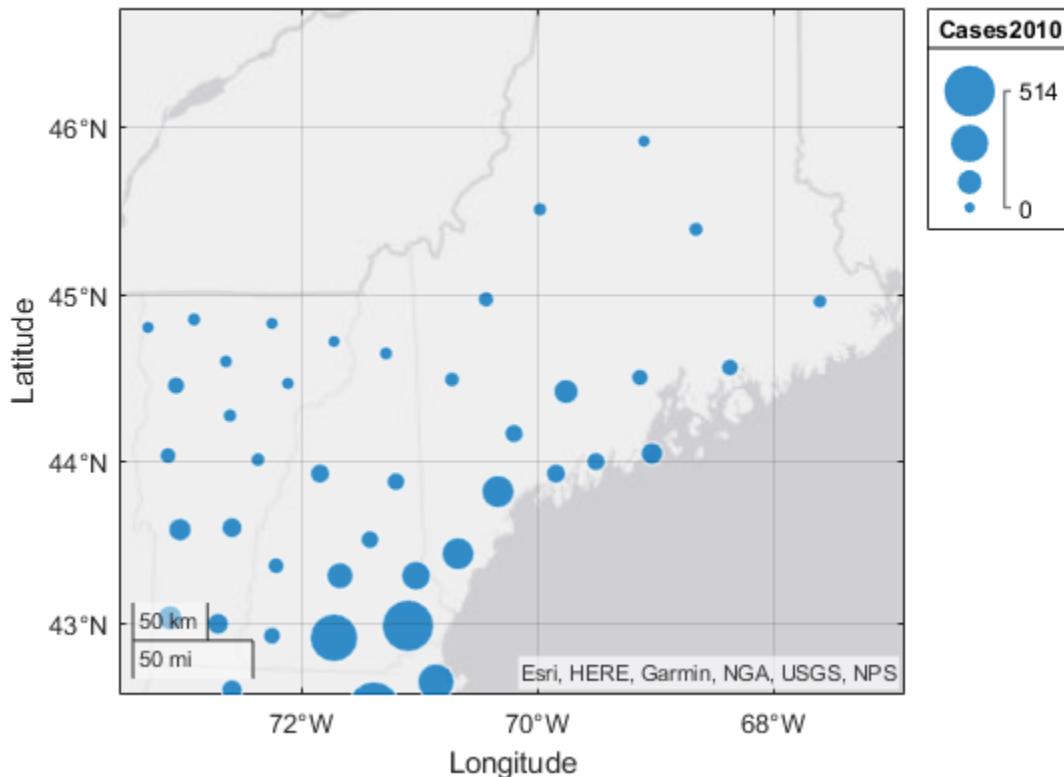
```
counties = readtable('counties.xlsx');
```

创建一个地理气泡图，绘制新英格兰各郡莱姆病的发生情况。

```
gb = geobubble(counties,'Latitude','Longitude','SizeVariable','Cases2010');
```



平移和缩放地图，直到只能看到新英格兰北部的州：佛蒙特州、新罕布什尔州和缅因州。

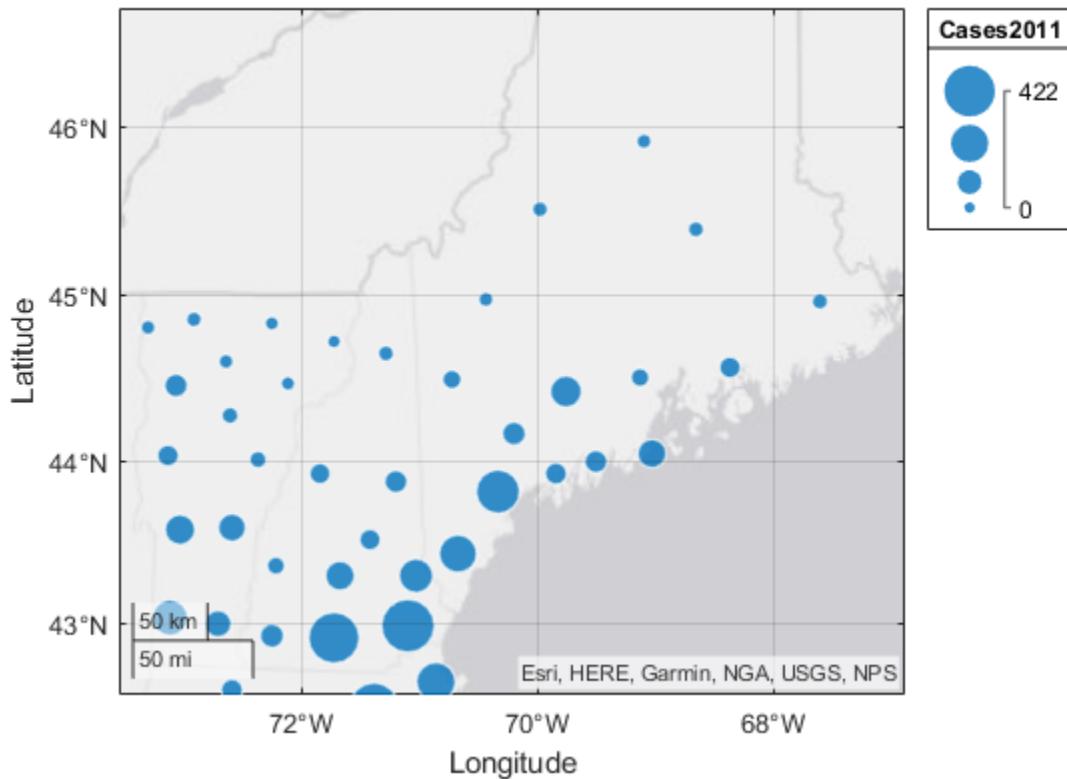


使用命令 `[nlat nlon] = geolimits(gb)` 获取新的地图范围。使用命令 `nzoom = gb.ZoomLevel` 获取新的缩放级别。存储新地图范围的经度、纬度和缩放级别。

```
nlat = [42.5577 46.6921];
nlon = [-73.5500 -66.8900];
nzoom = 6.3747;
```

使用 2011 年的莱姆病病例发生数据创建另一张地图，并设置与第一张图相匹配的地图范围和缩放级别。

```
figure
gb2 = geobubble(counties,'Latitude','Longitude','SizeVariable','Cases2011');
[n2lat n2lon] = geolimits(gb2,nlat,nlon);
gb2.ZoomLevel = nzoom;
```



另请参阅

[geolimits](#) | [GeographicAxes Properties](#) | [GeographicBubbleChart Properties](#) | [DensityPlot Properties](#) | [geoaxes](#) | [geobubble](#) | [geodensityplot](#) | [geoplot](#) | [geoscatte](#)

相关示例

- “地理气泡图概述” (第 6-7 页)
- “基于表格数据创建地理气泡图” (第 6-37 页)

访问用于地理坐标区和地理图的底图

MathWorks® 提供了底图供您选择，可与地理坐标区和地理图结合使用。底图提供了多种地图选项，包括双色调、彩色地形和高缩放级别显示等。有六幅底图是使用 Natural Earth 创建的图块化数据集。其中五个底图是由 Esri® 托管的高缩放级别地图。有关底图选项的详细信息，请参阅 [geobasemap](#)。

要为地理坐标区或地理图指定底图，您可以采用以下方式之一：

- 使用 [geobasemap](#) 函数。
- 设置 [GeographicAxes](#) 或 [GeographicBubbleChart](#) 对象的 [Basemap](#) 属性。

您也可以使用 Mapping Toolbox™ 函数 [addToolbarMapButton](#) 将底图选择器添加到坐标区工具栏。

MATLAB 包括一个已安装的底图，一个名为 'darkwater' 的双色地图。使用此底图不需要访问 Internet。使用其他底图，包括默认底图 'streets-light'，确实需要访问 Internet。

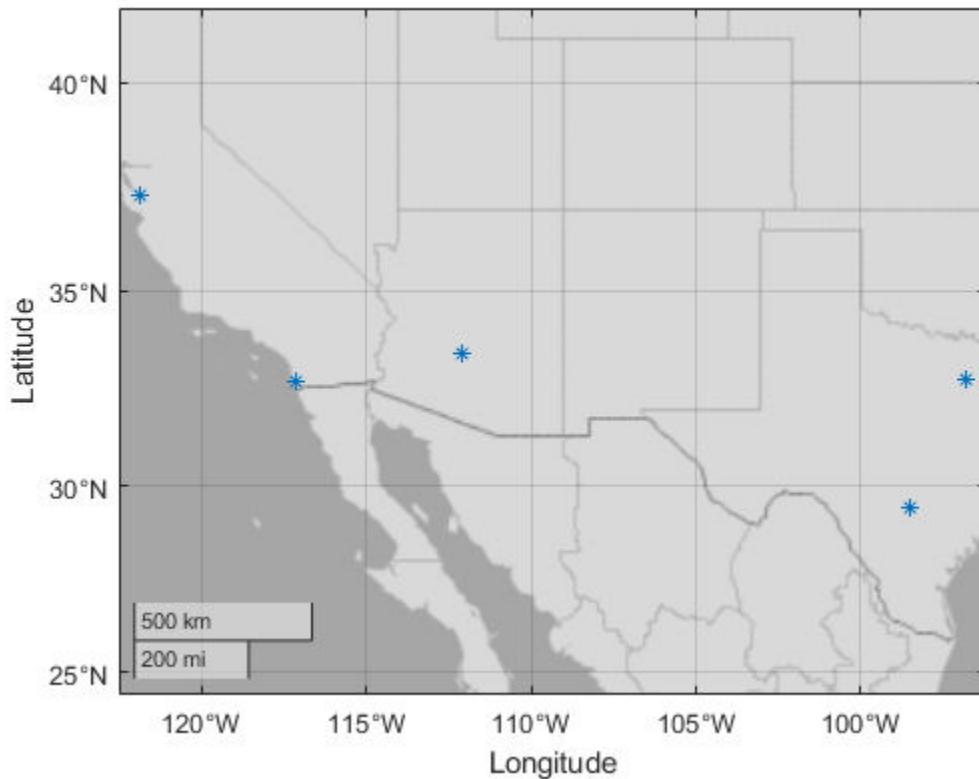
如果不能通过 Internet 访问底图，请检查代理服务器设置。有关指定代理服务器设置的详细信息，请参阅 “[Use MATLAB Web Preferences For Proxy Server Settings](#)”。

如果您无法稳定地访问 Internet，或希望改善地图的响应性能，您可以使用 'darkwater' 底图进行绘图，或将选择的底图下载到本地系统。

在地理图上显示 'darkwater'

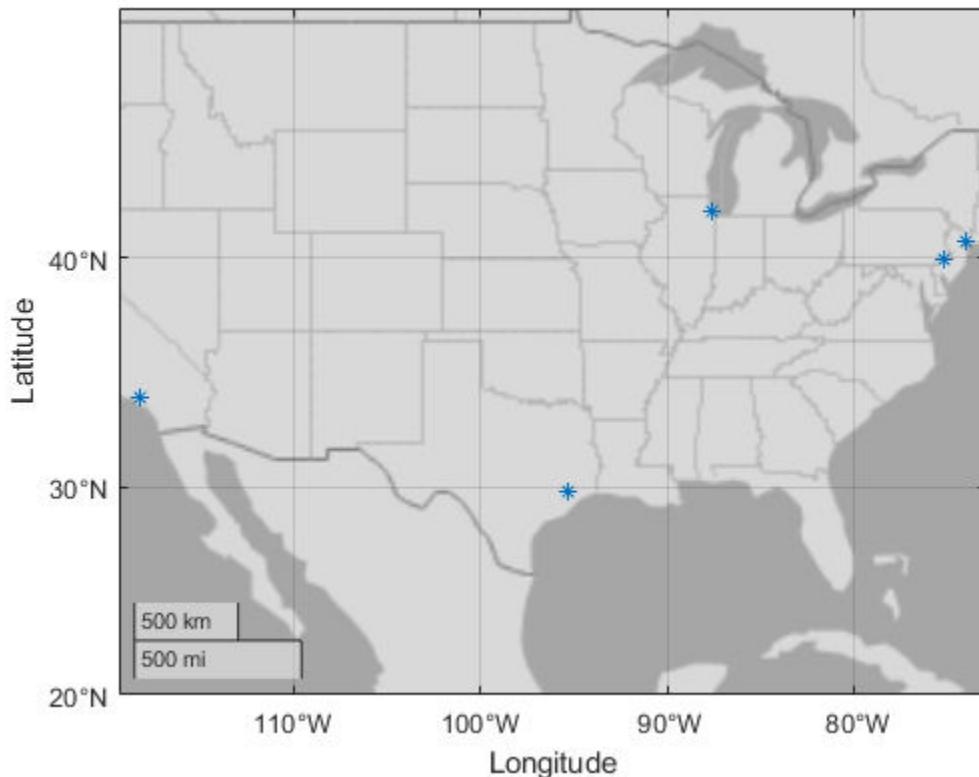
要在使用 [geoplot](#) 和 [geoscatter](#) 等函数绘图时显示 'darkwater' 底图，请调用 [geobasemap](#)。

```
lat1 = [33.448 29.424 32.716 32.777 37.338];
lon1 = [-112.074 -98.494 -117.161 -96.797 -121.886];
geoscatter(lat1,lon1,'*')
geobasemap darkwater
```



您也可以创建一组地理坐标区，并指定 Basemap 名称-值对组。要保留底图，请在绘图前使用 **hold on** 命令。

```
figure
lat2 = [40.713 34.052 41.878 29.760 39.952];
lon2 = [-74.006 -118.244 -87.630 -95.370 -75.165];
geoaxes('Basemap','darkwater')
hold on
geosscatter(lat2,lon2,'*')
```



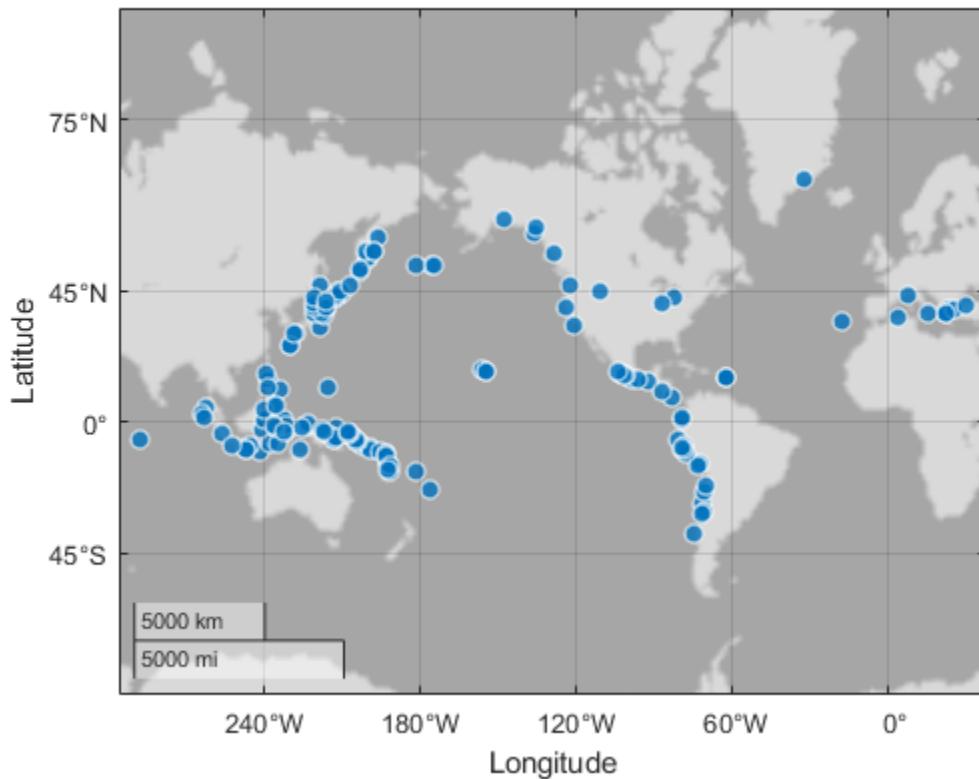
在您的 MATLAB 会话期间，您还可以更改使用 `geoplot`、`geoscatte` 和 `geodensityplot` 创建的所有绘图的默认底图。

```
set(groot,'defaultGeoaxesBasemap','darkwater')
```

在地理气泡图上显示 'darkwater'

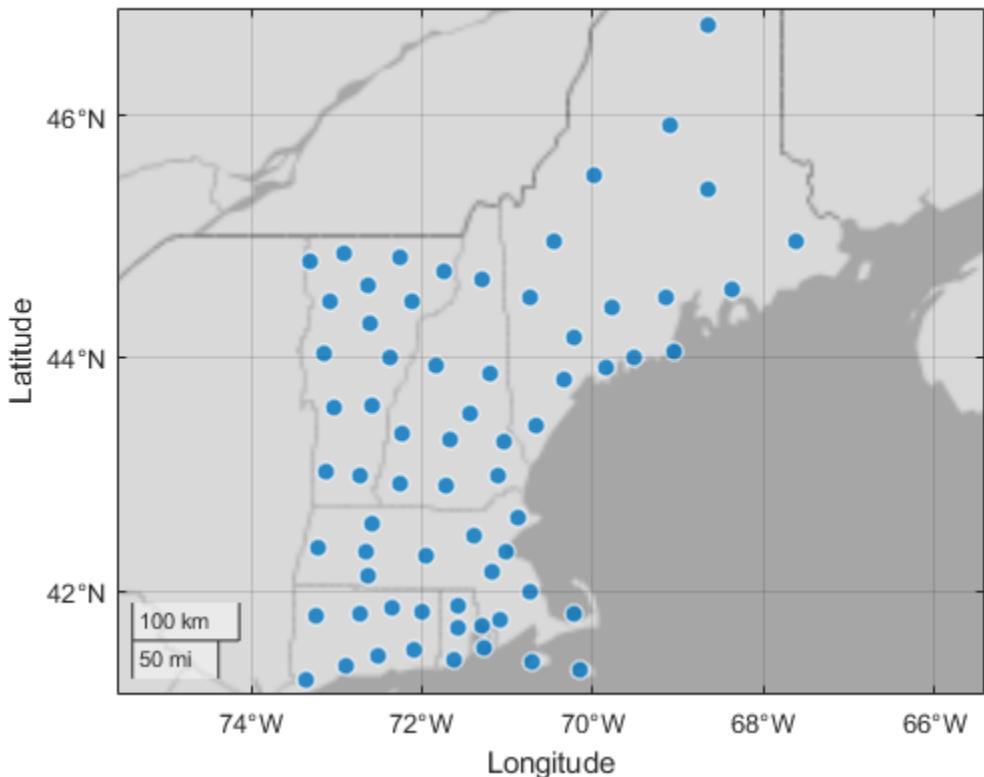
要在地理气泡图上显示 'darkwater'，请使用 'Basemap' 名称-值对组调用 `geobubble`。

```
tsunamis = readtable('tsunamis.xlsx');
geobubble(tsunamis,'Latitude','Longitude','Basemap','darkwater')
```



您也可以使用 `geobasemap` 指定底图。

```
counties = readtable('counties.xlsx');
geobubble(counties,'Latitude','Longitude')
geobasemap darkwater
```



下载底图

使用附加功能资源管理器将底图下载到您的本地系统。Esri 提供的五幅高缩放级别底图不可下载。

- 1 在 MATLAB 主页选项卡的环境部分，点击附加功能 > 获取附加功能。
- 2 在“附加功能资源管理器”中，滚动到 MathWorks 可选功能部分，然后点击显示所有以找到底图包。您还可以按名称搜索底图附加功能（如下表所列），或点击按类型筛选中的可选功能。
- 3 选择要下载的底图数据包。

底图名称	底图数据包名称
'bluegreen'	MATLAB Basemap Data - bluegreen
'grayland'	MATLAB Basemap Data - grayland
'colorterrain'	MATLAB Basemap Data - colorterrain
'grayterrain'	MATLAB Basemap Data - grayterrain
'landcover'	MATLAB Basemap Data - landcover

底图缓存行为

通过 Internet 访问底图时，MATLAB 会通过临时缓存底图图块来提高性能。有了这种缓存行为，当您在地图内平移和缩放时，程序只需下载每个图块一次即可。即使断开 Internet 连接，您仍然可以查看已查看过的地图部分，因为地图图块已存储在本地。

在未连接到 Internet 的情况下尝试查看之前未查看过的地图部分时，这些区域的图块不在缓存中。对于使用 Natural Earth 创建的底图，该程序使用 'darkwater' 底图中的图块替换缺失图块。

对于 Esri 提供的高缩放级别底图，程序可缓存有限数量的图块，缓存的图块在有限时间后过期。如果您尝试查看未缓存的高缩放级别底图的区域，您将看到空白地图图块。地理图不会使用来自 'darkwater' 的图块来显示这些缺失图块。

另请参阅

函数

[geobubble](#) | [geoaxes](#) | [geoplot](#) | [geobasemap](#) | [geoscatte](#)

属性

[GeographicAxes Properties](#) | [GeographicBubbleChart Properties](#)

相关示例

- “地理气泡图概述”（第 6-7 页）

基于表格数据创建地理气泡图

地理气泡图是一种以可视化方式在地图上叠加显示数据的图形。对于具有地理特征的数据，这些图可以提供重要的上下文信息。在此示例中，您将文件以表的形式导入 MATLAB® 中，并基于表变量（列）创建地理气泡图。然后，再使用表中的数据来可视化显示数据的各个方面，例如人口数量大小。

以表的形式导入文件

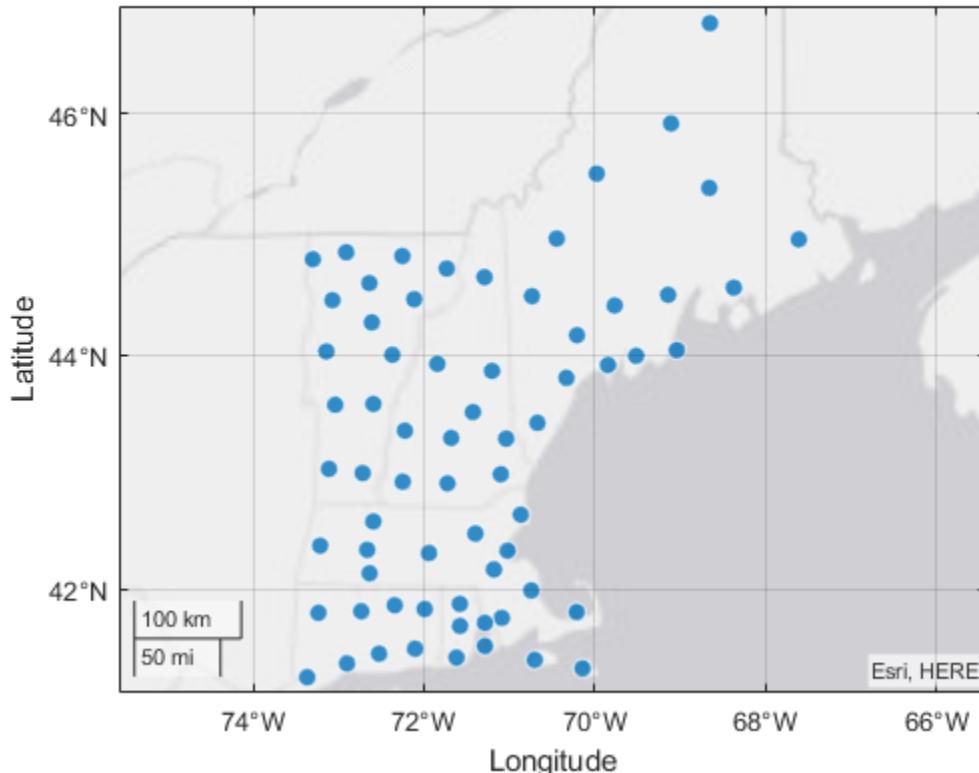
加载样本文件 `counties.xlsx`，其中包含新英格兰各郡的人口数据和莱姆病病例数据。使用 `readtable` 将数据读入一个表中。

```
counties = readtable('counties.xlsx');
```

创建基本地理气泡图

创建一个地理气泡图，显示新英格兰各郡的位置。将该表指定为第一个参数 `counties`。地理气泡图将该表存储在其 `SourceTable` 属性中。使用表的 '`Latitude`' 和 '`Longitude`' 列指定位置。该图自动设置底层地图（称为底图）的纬度和经度范围，以仅包括由这些数据表示的区域。将 `GeographicBubbleChart` 对象赋给变量 `gb`。在创建图后，可使用 `gb` 对其进行修改。

```
figure  
gb = geobubble(counties,'Latitude','Longitude');
```

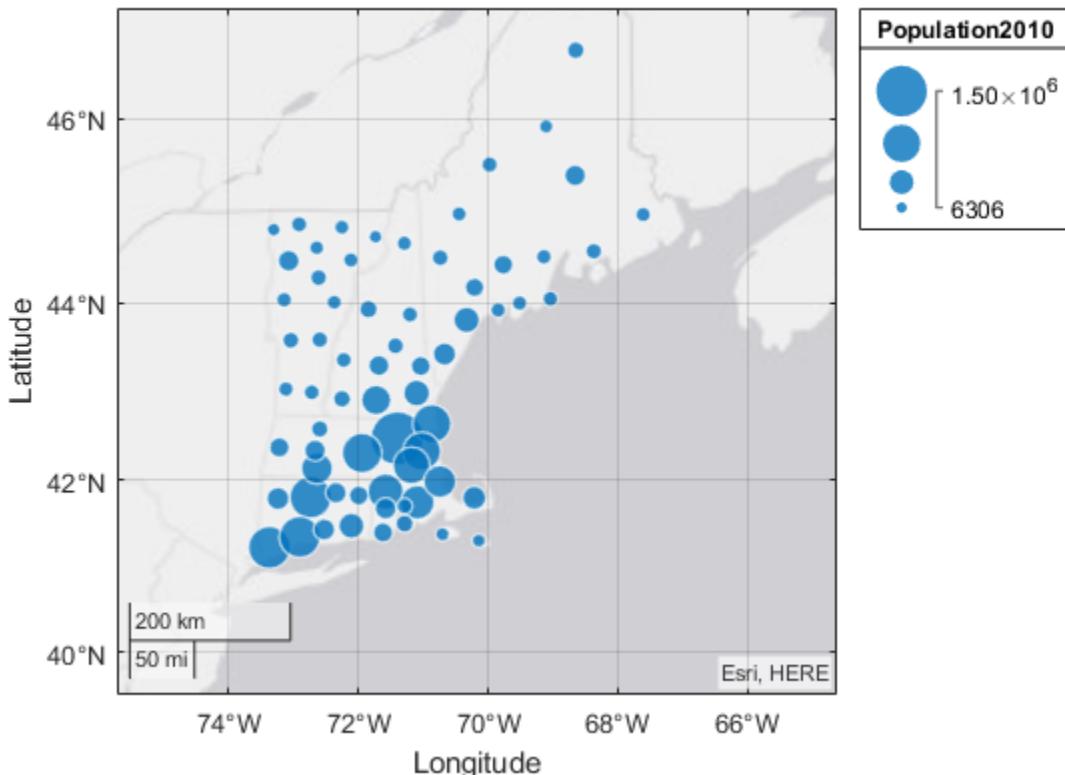


可以在 `geobubble` 函数显示的底图上平移和缩放。

在图上可视化显示各郡人口

用气泡大小（直径）表示不同郡的相对人口。将表中的 **Population2010** 变量指定为 **SizeVariable** 参数的值。在所得到的地理气泡图中，气泡的不同大小代表着人口的多少。该图包括了一个描述气泡直径与人口数量对应关系的图例。使用 **geolimits** 调整图的范围。

```
gb = geobubble(counties,'Latitude','Longitude',...
    'SizeVariable','Population2010');
geolimits([39.50 47.17],[-74.94 -65.40])
```

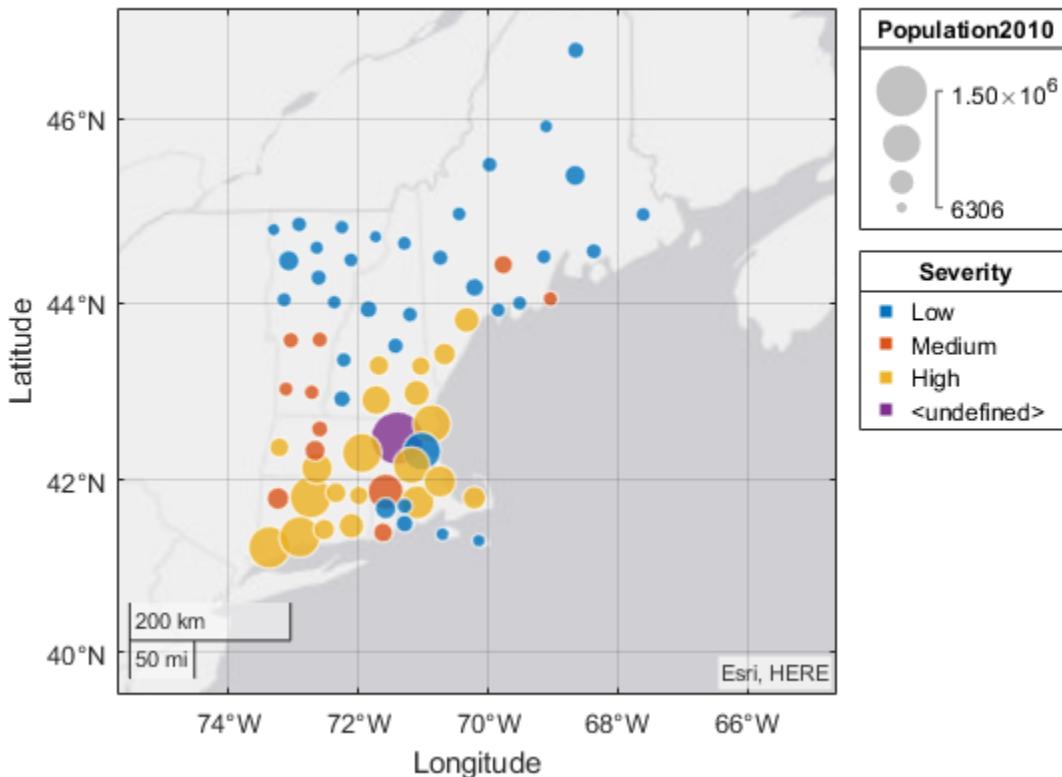


geobubble 在 **SizeLimits** 属性指定的值的范围内对气泡直径进行线性缩放。

可视化显示各郡莱姆病病例

使用气泡颜色显示某郡在给定年份的莱姆病病例数。要显示此类型的数据，**geobubble** 函数要求所提供的数据为 **categorical** 值。原始表中的各列数据都不是分类数据，但您可以创建一个分类数据。例如，您可以使用 **discretize** 函数基于 **Cases2010** 变量中的数据创建一个分类变量。名为 **Severity** 的新变量将数据分为三个类别：Low、Medium 和 High。使用这个新变量作为 **ColorVariable** 参数。这些更改会修改存储在 **SourceTable** 属性中的表，该表是工作区中原始表 **counties** 的副本。对 **GeographicBubbleChart** 对象中存储的表进行更改可避免原始数据受到影响。

```
gb.SourceTable.Severity = discretize(counties.Cases2010,[0 50 100 500],...
    'categorical',{'Low','Medium','High'});
gb.ColorVariable = 'Severity';
```



处理未定义的数据

对严重性信息绘图时，颜色图例中将出现第四个类别：`undefined`。当您转换为 `categorical` 的数据包含空值或超出您所定义类别的范围的值时，会显示此类别。将光标悬停在未定义的气泡上，确定未定义 Severity 值的原因。数据提示显示，气泡代表莱姆病表第 33 行中的值。

检查用于 Severity 的变量 Cases2010 的值，即莱姆病表第 33 行中的第 12 个变量。

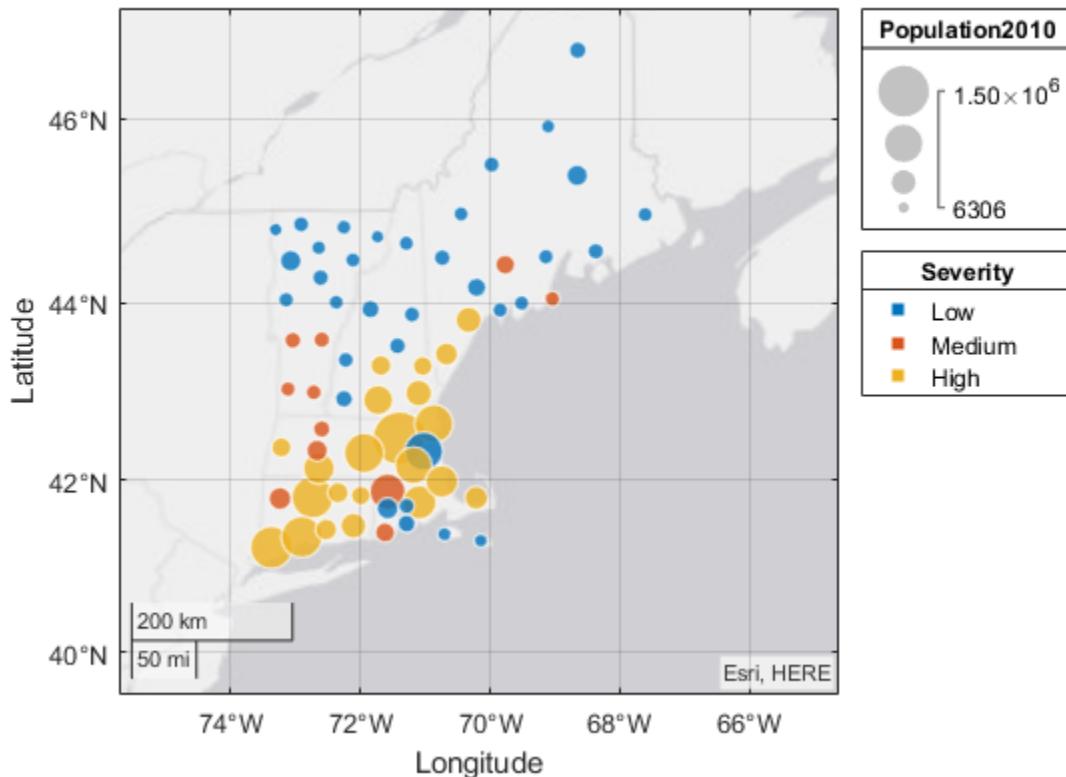
```
gb.SourceTable(33,12)
```

```
ans=table
Cases2010
```

```
514
```

High 类别定义为介于 100 和 500 之间的值。但是，Cases2010 变量的值是 514。要消除此未定义的值，请重新设置 High 类别的上限以包含此值。例如，使用 5000。

```
gb.SourceTable.Severity = discretize(counties.Cases2010,[0 50 100 5000],...
    'categorical',{'Low','Medium','High'});
```

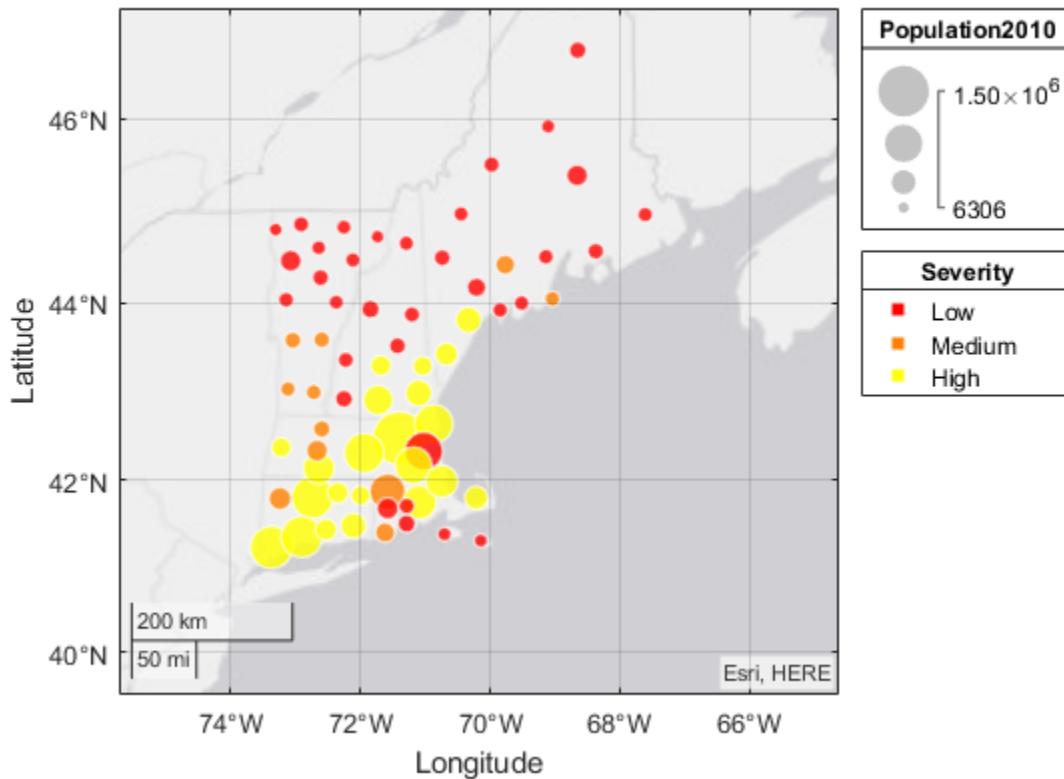


与颜色变量不同，当 `geobubble` 在大小、纬度或经度变量中遇到未定义的数字 (NaN) 时，它会忽略该值。

选择气泡颜色

使用颜色梯度来表示 Low-Medium-High 分类。`geobubble` 在 `BubbleColorList` 属性中将颜色存储为 $m \times 3$ RGB 值列表。

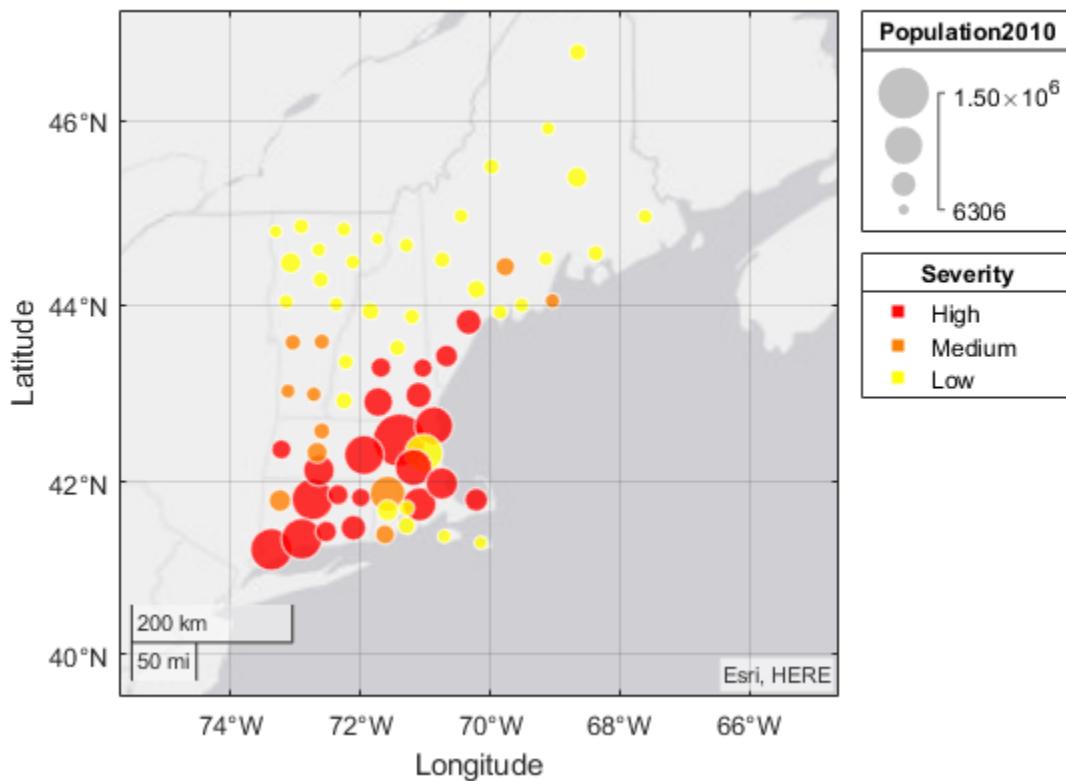
```
gb.BubbleColorList = autumn(3);
```



对气泡颜色重新排序

将指示高严重性的颜色更改为红色而不是黄色。要更改色序，您可以更改类别的顺序，也可以更改 BubbleColorList 属性中列出的颜色的顺序。例如，最初类别顺序为 Low-Medium-High。使用 reordercats 函数将类别顺序更改为 High-Medium-Low。颜色图例中的类别随之更改。

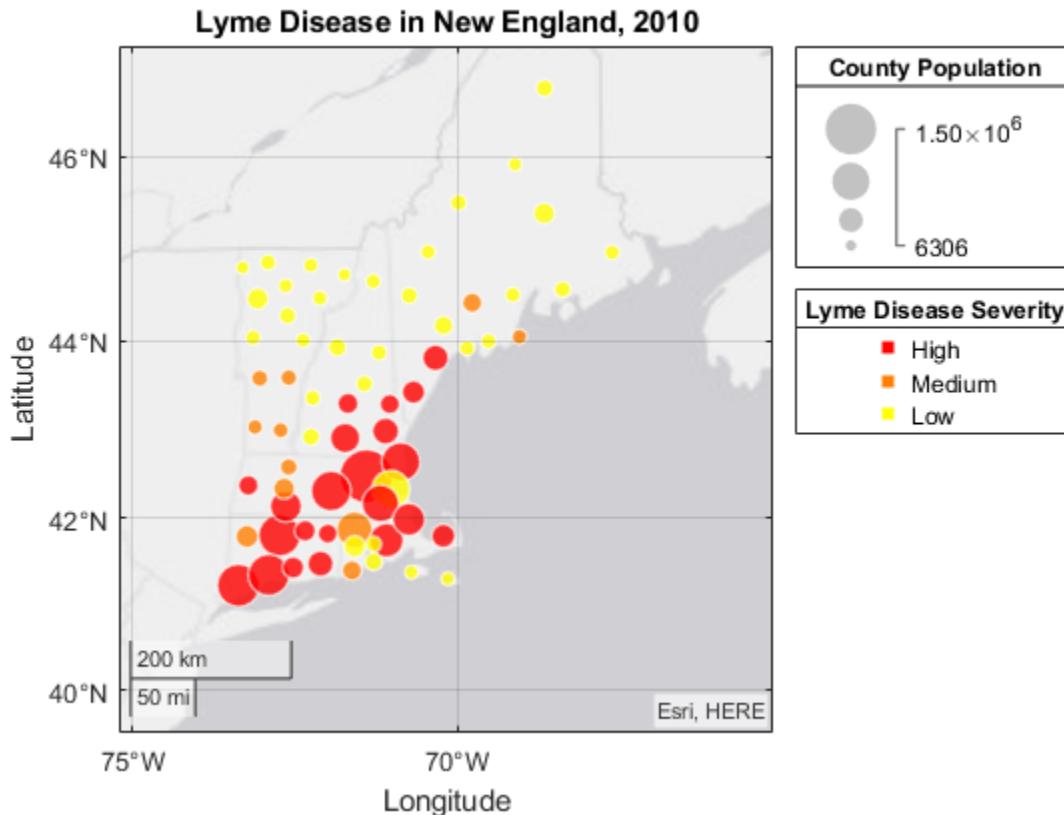
```
neworder = {'High','Medium','Low'};
gb.SourceTable.Severity = reordercats(gb.SourceTable.Severity,neworder);
```



添加标题

当您使用大小和颜色变量显示地理气泡图时，该图会显示大小图例和颜色图例以指示相对大小和颜色的含义。当您指定表作为参数时，`geobubble` 会自动使用表变量名称作为图例标题，但您可以使用属性来指定其他标题。

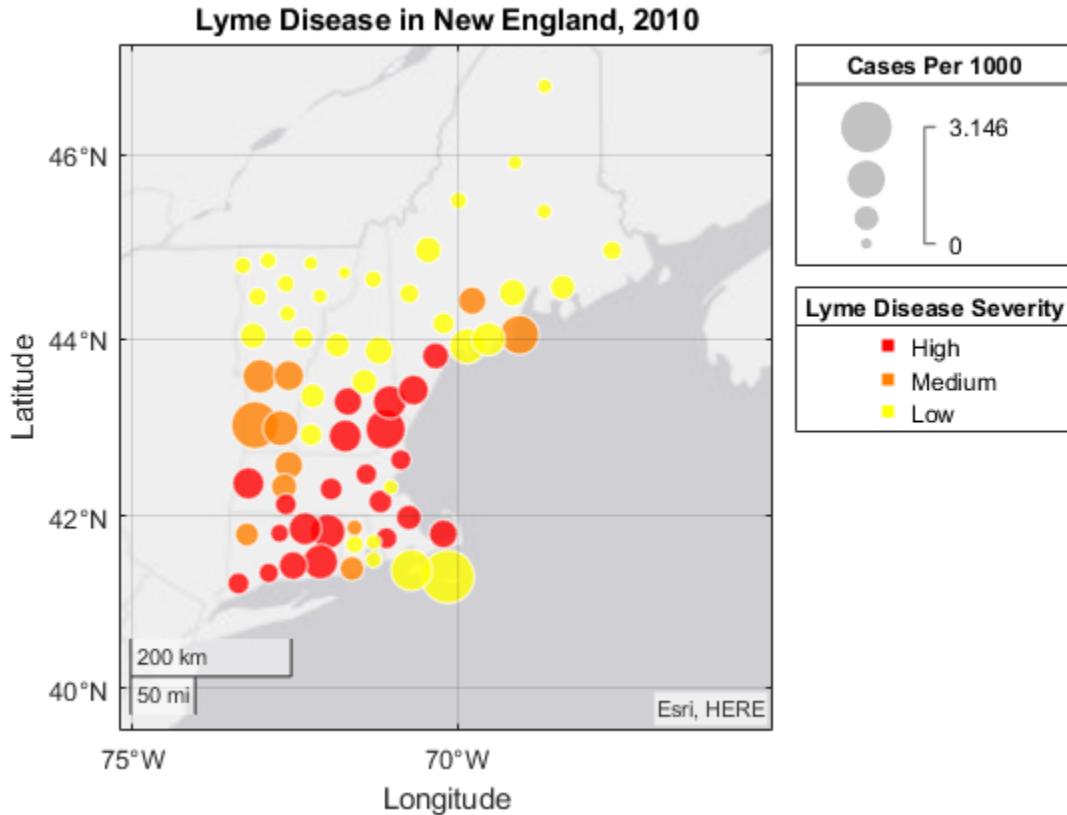
```
title 'Lyme Disease in New England, 2010'
gb.SizeLegendTitle = 'County Population';
gb.ColorLegendTitle = 'Lyme Disease Severity';
```



细化处理图数据

观察莱姆病数据，呈现的趋势似乎是更多病例发生在人口密度较大的地区。查看人口患病率最高的地点可能更有意义。计算每 1000 人中出现的病例数并显示在图上。

```
gb.SourceTable.CasesPer1000 = gb.SourceTable.Cases2010 ./ ...
  gb.SourceTable.Population2010 * 1000;
gb.SizeVariable = 'CasesPer1000';
gb.SizeLegendTitle = 'Cases Per 1000';
```



现在，气泡大小与之前有不同的意义。人口最多的地区与病例严重程度相对吻合。然而，如果按人口对病例数进行归一化处理，则人口患病风险最高的地点呈现不同的地理分布。

另请参阅

[geobubble](#) | [table](#) | [readtable](#) | [reordercats](#) | [categorical](#) | [discretize](#) | [GeographicBubbleChart](#)
Properties

相关示例

- “使用地理气泡图属性”（第 6-23 页）
- “部署地理坐标区和地理图”（第 6-22 页）
- “访问用于地理坐标区和地理图的底图”（第 6-31 页）
- “地理气泡图概述”（第 6-7 页）

动画

- “动画技术” (第 7-2 页)
- “沿线条跟踪标记” (第 7-3 页)
- “沿着线条移动一组对象” (第 7-5 页)
- “对图形对象进行动画处理” (第 7-8 页)
- “线条动画” (第 7-10 页)
- “录制动画用于播放” (第 7-12 页)
- “为曲面添加动画效果” (第 7-15 页)

动画技术

本节内容

- “更新屏幕” (第 7-2 页)
- “优化性能” (第 7-2 页)

您可以使用三种基本方法在 MATLAB 中创建动画：

- 更新图形对象的属性并在屏幕上显示更新。这一技术对于在图形大部分保持不变的情况下创建动画非常有用。例如，重复设置 **XData** 和 **YData** 属性以在图形中移动对象。
- 将变换应用于对象。当您想要同时对一组对象的位置和方向进行操作时，此技术非常有用。将一些对象归组为某一变换对象的子级。使用 **hgtransform** 创建变换对象。设置变换对象的 **Matrix** 属性，调整其所有子级的位置。
- 创建一个影片。如果您有一个复杂的动画无法实时快速绘制，或想要存储动画以再次播放时，影片就非常有用。使用 **getframe** 和 **movie** 函数创建一个影片。

更新屏幕

在某些情况下，直到代码完成后 MATLAB 才会更新屏幕。使用 **drawnow** 命令中的一个命令，在动画过程中显示整个屏幕的更新。

优化性能

要优化性能，可考虑以下技术：

- 使用 **animatedline** 函数创建流化数据的线条动画。
- 更新现有对象的属性，不创建新的图形对象。
- 设置坐标轴范围 (**XLim**、**YLim**、**ZLim**) 或将与之关联的模式属性改为手动模式 (**XLimMode**、**YLimMode**、**ZLimMode**)，以避免 MATLAB 在每次屏幕更新时重新计算值。当您设置坐标轴范围时，与之关联的模式属性会更改成手动模式。
- 尽量避免在循环中创建图例或其他注释。可以在循环之后添加注释。

有关优化性能的详细信息，请参阅 “图形性能”。

另请参阅

相关示例

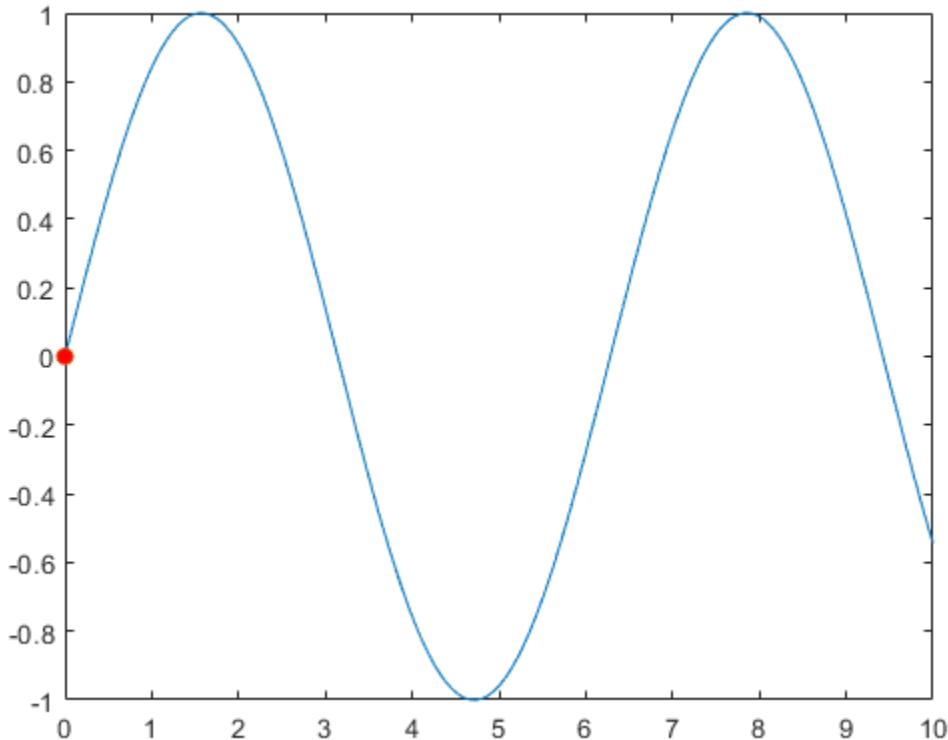
- “沿线条跟踪标记” (第 7-3 页)
- “沿着线条移动一组对象” (第 7-5 页)
- “线条动画” (第 7-10 页)
- “录制动画用于播放” (第 7-12 页)

沿线条跟踪标记

此示例演示如何通过更新标记属性以沿着线条跟踪标记。

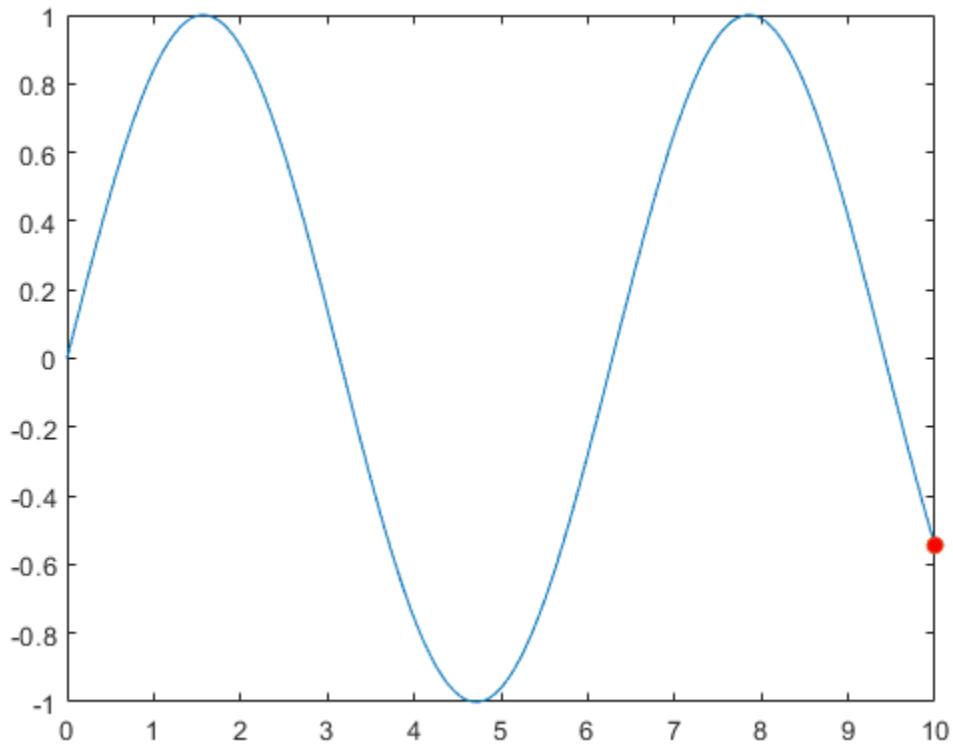
绘制一个正弦波并在线条开始处绘制红色标记。将坐标轴范围模式设置为手动，以避免在动画循环中重新计算范围。

```
x = linspace(0,10,1000);
y = sin(x);
plot(x,y)
hold on
p = plot(x(1),y(1),'o','MarkerFaceColor','red');
hold off
axis manual
```



通过在循环中更新 XData 和 YData 属性，实现沿着线条移动标记。使用 `drawnow` 或 `drawnow limitrate` 命令在屏幕上显示更新。`drawnow limitrate` 的速度最快，但它可能不会在屏幕上绘制每一帧。使用圆点表示法设置属性。

```
for k = 2:length(x)
    p.XData = x(k);
    p.YData = y(k);
    drawnow
end
```



动画演示标记沿着线条移动。

另请参阅

`plot` | `drawnow` | `linspace`

相关示例

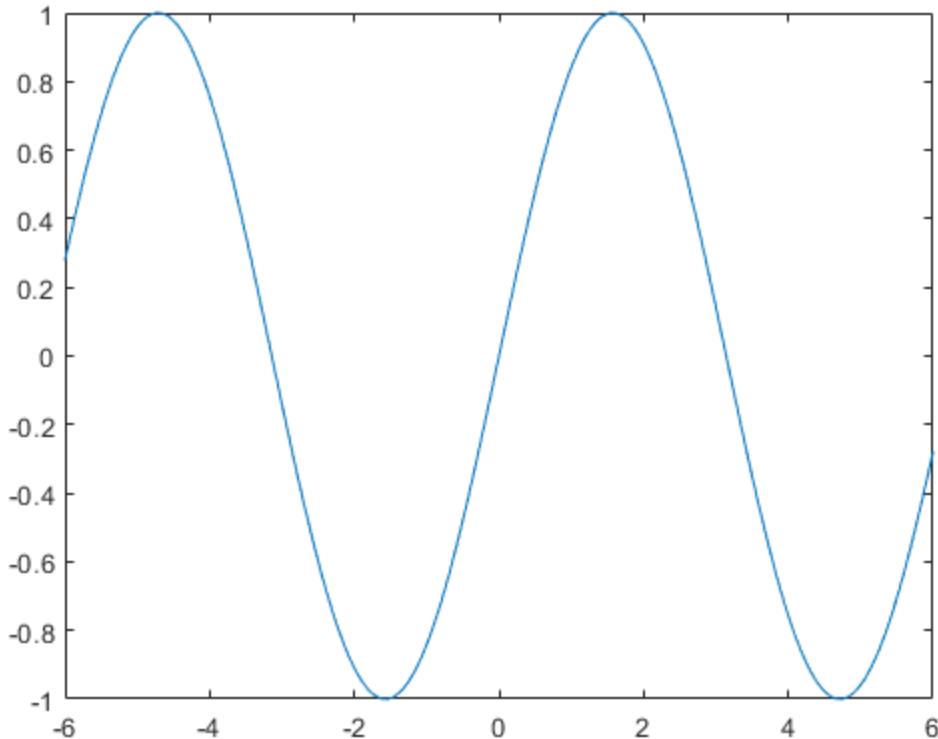
- “沿着线条移动一组对象” (第 7-5 页)
- “对图形对象进行动画处理” (第 7-8 页)
- “录制动画用于播放” (第 7-12 页)
- “线条动画” (第 7-10 页)

沿着线条移动一组对象

此示例演示如何使用变换沿着线条移动一组对象。

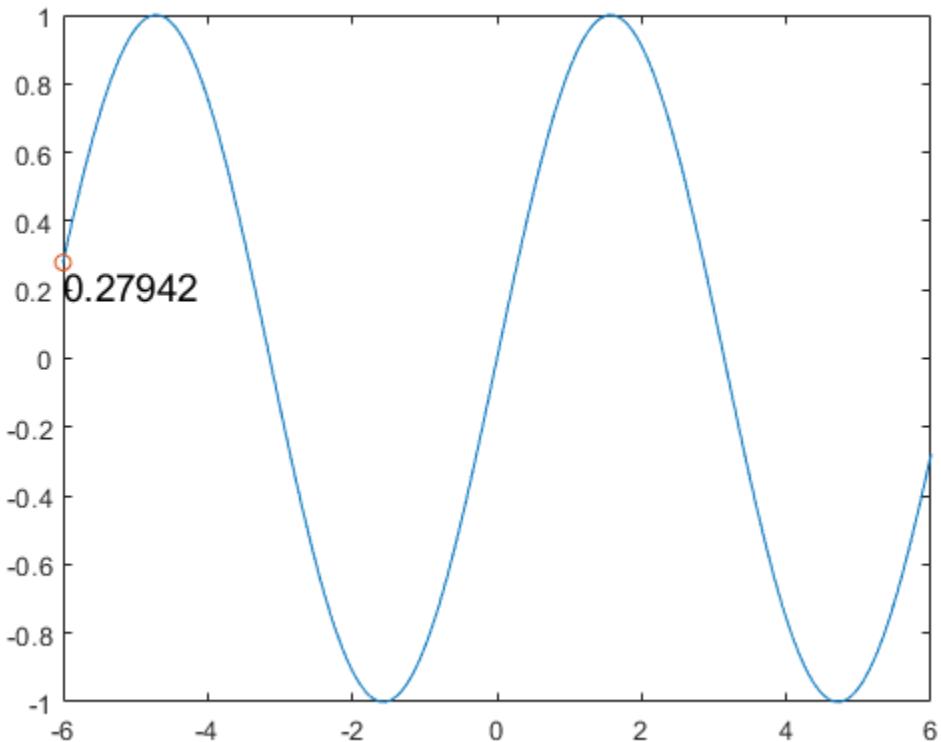
绘制一条正弦波，将坐标轴范围模式设置为手动，以避免在动画循环中重新计算范围。

```
x = linspace(-6,6,1000);
y = sin(x);
plot(x,y)
axis manual
```



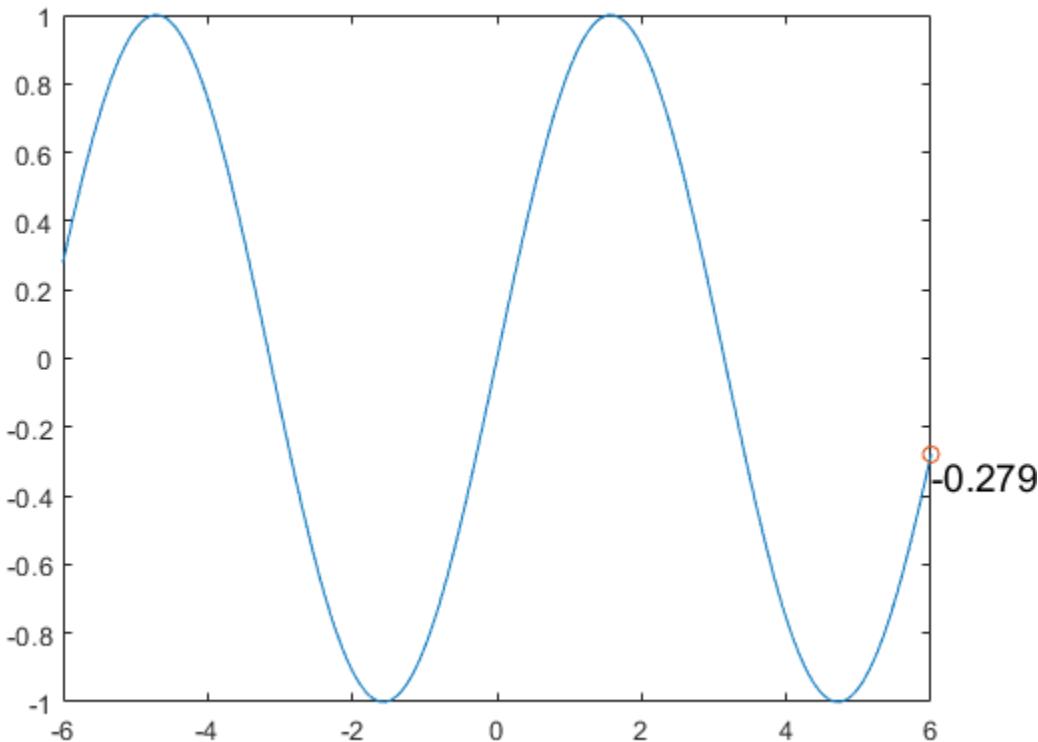
创建一个变换对象并将父级设置为当前坐标区。在线条初始处绘制标记和文本注释。使用 `num2str` 函数将该点处的 y 值转换为文本。通过将两个对象的父级设置为变换对象，将两个对象归为一组。

```
ax = gca;
h = hgtransform('Parent',ax);
hold on
plot(x(1),y(1),'o','Parent',h);
hold off
t = text(x(1),y(1),num2str(y(1)),'Parent',h,...
'VerticalAlignment','top','FontSize',14);
```



通过更新变换对象的 **Matrix** 属性，将标记和文本沿线条移动到每个后续点。使用线条中后接点和第一个点的 **x** 和 **y** 值确定变换矩阵。更新文本以匹配随着线条移动时的 **y** 值。使用 **drawnow** 显示每次迭代后的屏幕更新。

```
for k = 2:length(x)
    m = makehgform('translate',x(k)-x(1),y(k)-y(1),0);
    h.Matrix = m;
    t.String = num2str(y(k));
    drawnow
end
```



动画显示标记和文本沿着线条一起移动。

如果有很多数据，您可以使用 `drawnow limitrate` 代替 `drawnow` 以实现更快的动画。但是，`drawnow limitrate` 可能不会在屏幕上绘制每个更新。

另请参阅

`hgtransform` | `makehgform` | `plot` | `drawnow` | `axis` | `text`

相关示例

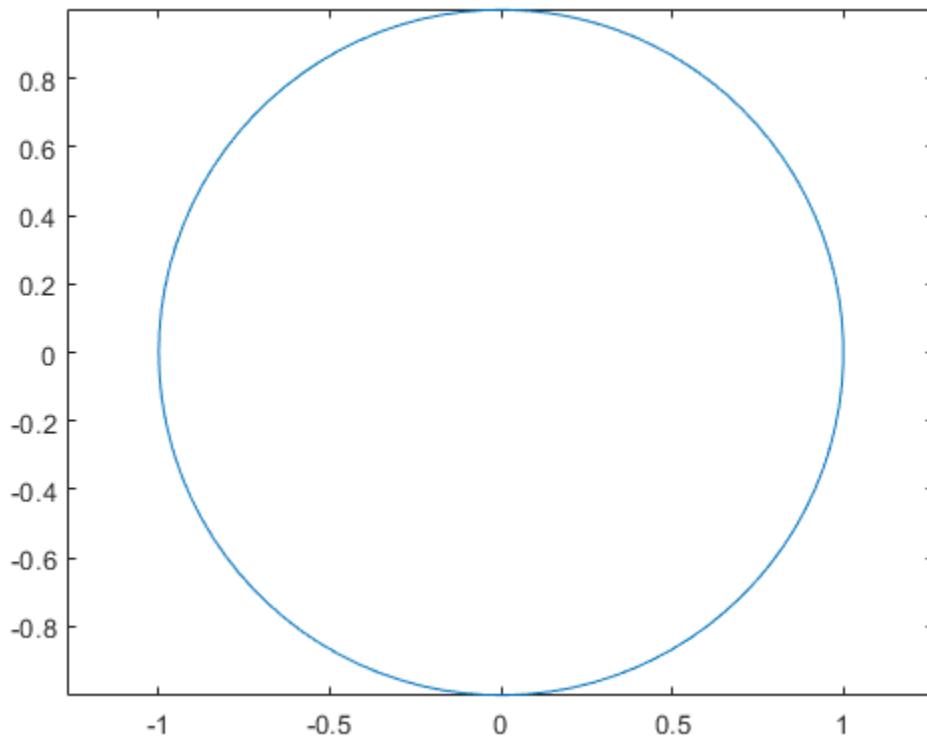
- “对图形对象进行动画处理”（第 7-8 页）
- “录制动画用于播放”（第 7-12 页）
- “线条动画”（第 7-10 页）

对图形对象进行动画处理

此示例演示如何通过更新三角形的数据属性实现三角形沿着圆形内部循环的动画效果。

绘制圆形并设置坐标轴范围，使数据单位在两个方向上保持一致。

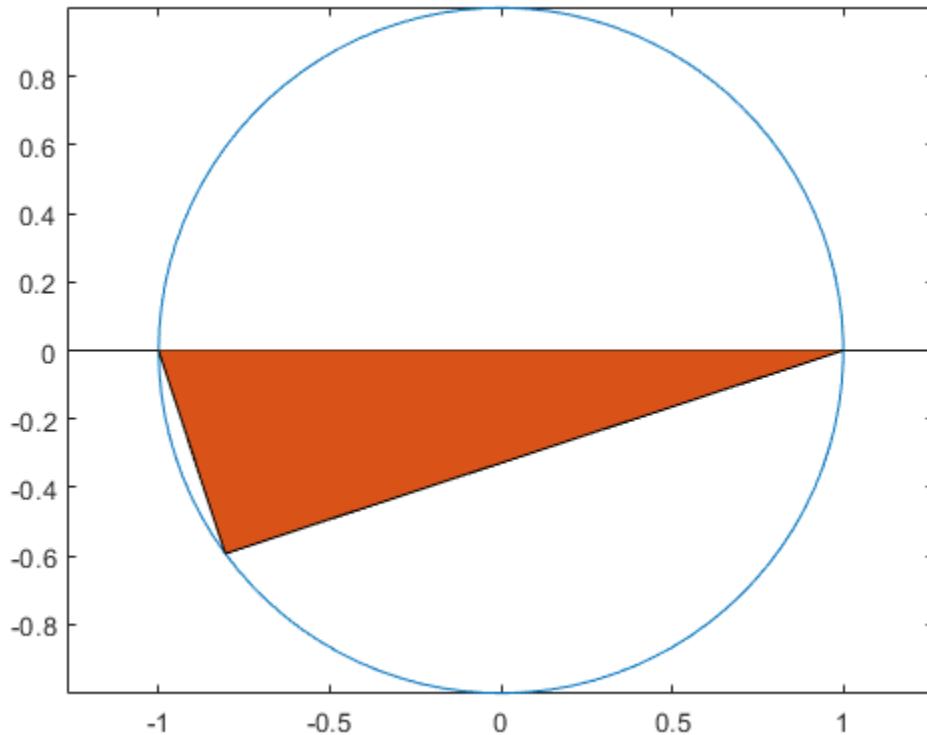
```
theta = linspace(-pi,pi);
xc = cos(theta);
yc = -sin(theta);
plot(xc,yc);
axis equal
```



使用 `area` 函数绘制一个平坦三角形。然后使用圆形的 (x,y) 坐标改变三角形其中一个顶点的值。改变循环中的值，创建一个动画。使用 `drawnow` 或 `drawnow limitrate` 命令在每次迭代后显示更新。
`drawnow limitrate` 的速度最快，但它可能不会在屏幕上绘制每一帧。

```
xt = [-1 0 1 -1];
yt = [0 0 0 0];
hold on
t = area(xt,yt); % initial flat triangle
hold off
for j = 1:length(theta)-10
    xt(2) = xc(j); % determine new vertex value
    yt(2) = yc(j);
    t.XData = xt; % update data properties
    t.YData = yt;
```

```
drawnow limitrate % display updates  
end
```



该动画演示三角形沿着圆形内部循环。

另请参阅

[area](#) | [plot](#) | [hold](#) | [drawnow](#) | [axis](#)

相关示例

- “沿线条跟踪标记” (第 7-3 页)
- “线条动画” (第 7-10 页)
- “录制动画用于播放” (第 7-12 页)

详细信息

- “动画技术” (第 7-2 页)

线条动画

此示例演示如何创建由两条不断变长的线条组成的动画。`animatedline` 函数帮助您优化线条动画。它可以向线条添加新的点而不用重新定义现有点。

创建线条并添加点

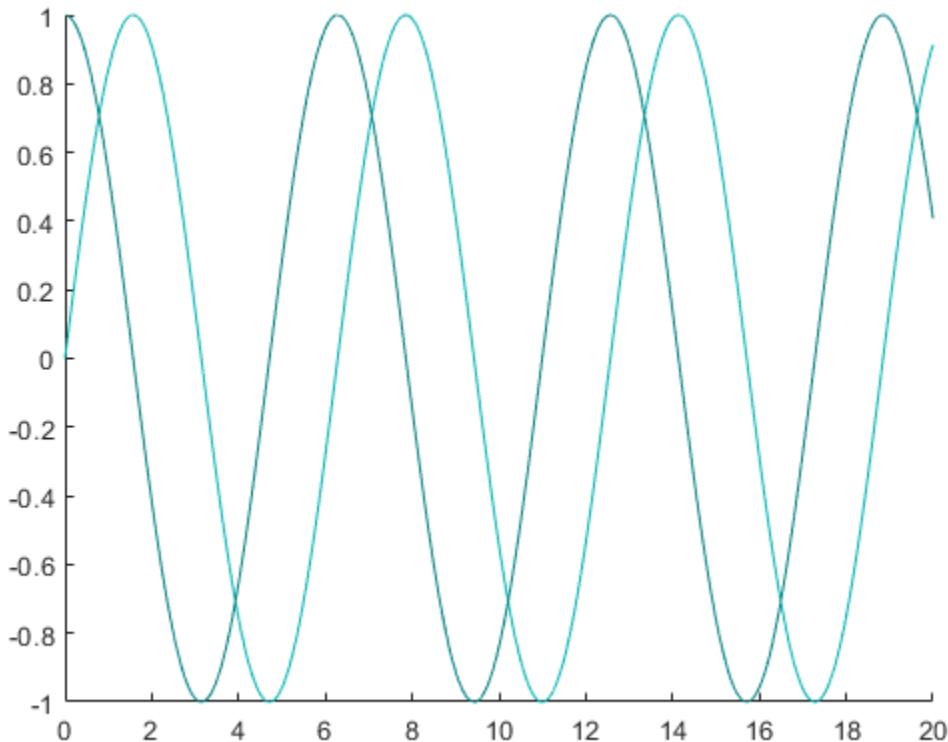
创建两条不同颜色的动画线条。然后在循环中向线条添加点。在循环之前设置坐标轴范围，从而避免每次循环时重新计算范围。使用 `drawnow` 或 `drawnow limitrate` 命令在添加新点之后在屏幕上显示更新。

```
a1 = animatedline('Color',[0 .7 .7]);
a2 = animatedline('Color',[0 .5 .5]);

axis([0 20 -1 1])
x = linspace(0,20,10000);
for k = 1:length(x)
    % first line
    xk = x(k);
    ysin = sin(xk);
    addpoints(a1,xk,ysin);

    % second line
    ycos = cos(xk);
    addpoints(a2,xk,ycos);

    % update screen
    drawnow limitrate
end
```



动画显示两个线条随着数据增加不断变长。

查询线条上的点

查询第一个动画线条上的点。

```
[x,y] = getpoints(a1);
```

x 和 y 是包含定义正弦波的值的向量。

另请参阅

`animatedline | addpoints | getpoints | clearpoints | drawnow`

相关示例

- “沿线条跟踪标记” (第 7-3 页)
- “沿着线条移动一组对象” (第 7-5 页)
- “录制动画用于播放” (第 7-12 页)

详细信息

- “动画技术” (第 7-2 页)

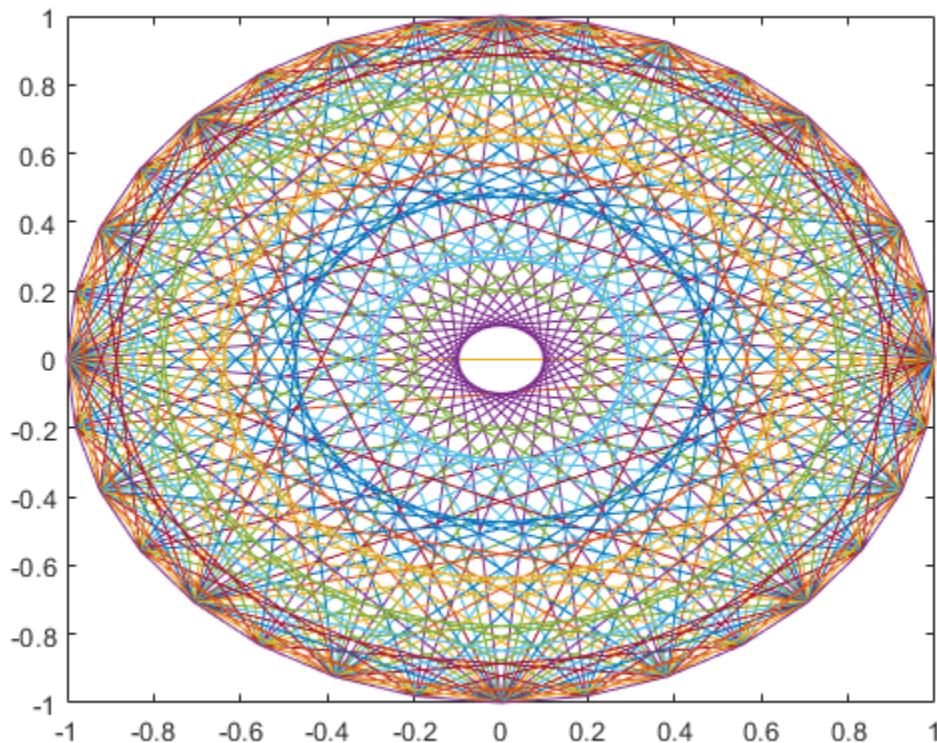
录制动画用于播放

这些示例演示如何录制可播放的动画。

录制和播放影片

在循环中创建绘图系列并将每个绘图捕获为一帧。通过每次在循环中进行设置确保坐标轴范围为常量。将帧存储到 `M` 中。

```
for k = 1:16
    plot(fft(eye(k+16)))
    axis([-1 1 -1 1])
    M(k) = getframe;
end
```



使用 `movie` 函数播放影片。

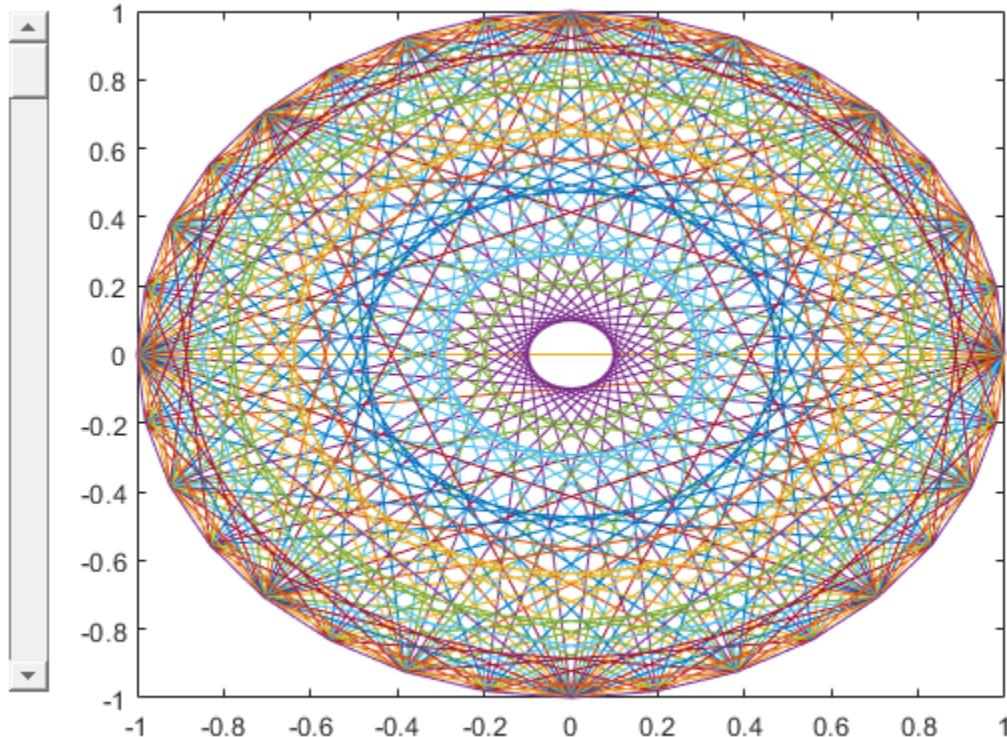
```
figure
movie(M,5)
```

为影片捕获整个图窗

在图窗左侧包含一个滑块。通过将图窗指定为 `getframe` 函数的输入参数捕获整个图窗窗口。

```
figure
u = uicontrol('Style','slider','Position',[10 50 20 340],...
```

```
'Min',1,'Max',16,'Value',1);
for k = 1:16
    plot(fft(eye(k+16)))
    axis([-1 1 -1 1])
    u.Value = k;
    M(k) = getframe(gcf);
end
```



播放影片五次。在当前坐标区中播放影片。创建一个新的图窗和坐标区来填充图窗窗口，从而让影片看上去像原始动画。

```
figure
axes('Position',[0 0 1 1])
movie(M,5)
```

另请参阅

[getframe](#) | [movie](#) | [fft](#) | [eye](#) | [plot](#) | [axes](#) | [axis](#)

相关示例

- “对图形对象进行动画处理” (第 7-8 页)
- “线条动画” (第 7-10 页)

详细信息

- “动画技术” (第 7-2 页)

为曲面添加动画效果

此示例说明如何对曲面进行动画处理。具体而言，此示例是对球谐函数进行动画处理。球谐函数是傅里叶级数的球面版本，可用于构建地球自由振动的模型。

定义球面网格

定义球面网格上的一组点以计算谐波。

```
theta = 0:pi/40:pi;
phi = 0:pi/20:2*pi;

[phi,theta] = meshgrid(phi,theta);
```

计算球谐函数

在半径为 5 的球面上计算一个次数为 6、阶数为 1、幅值为 0.5 的球谐函数。然后，将值转换为笛卡尔坐标。

```
degree = 6;
order = 1;
amplitude = 0.5;
radius = 5;

Ymn = legendre(degree,cos(theta(:,1)));
Ymn = Ymn(order+1,:)';
yy = Ymn;

for kk = 2: size(theta,1)
    yy = [yy Ymn];
end

yy = yy.*cos(order*phi);

order = max(max(abs(yy)));
rho = radius + amplitude*yy/order;

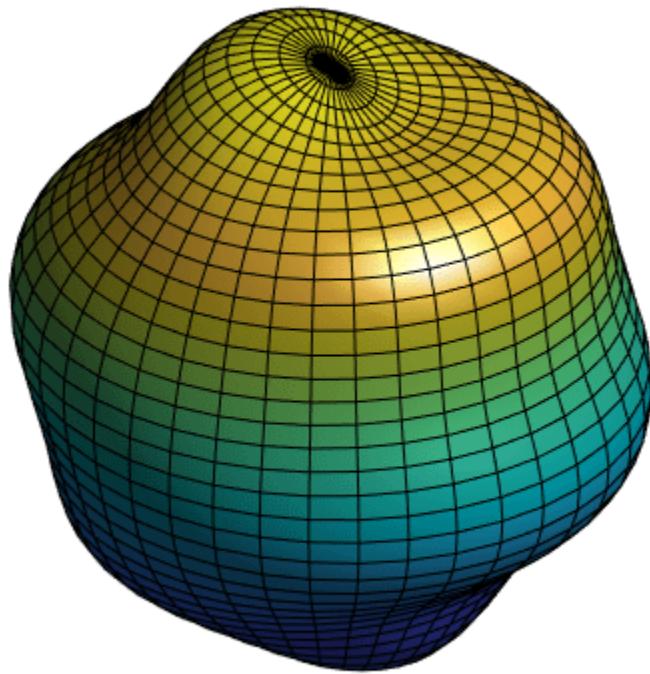
r = rho.*sin(theta);
x = r.*cos(phi);
y = r.*sin(phi);
z = rho.*cos(theta);
```

在球面上绘制球谐函数

使用 `surf` 函数在球面上绘制球谐函数。

```
figure
s = surf(x,y,z);

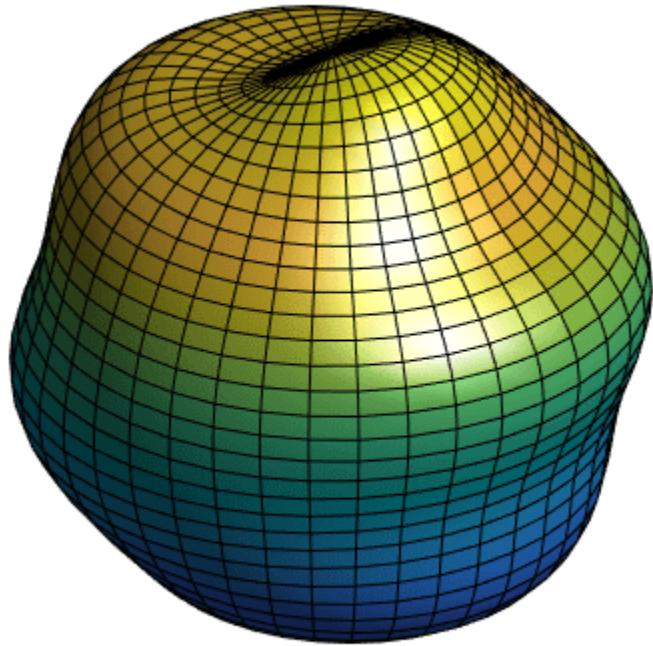
light
lighting gouraud
axis equal off
view(40,30)
camzoom(1.5)
```



为曲面添加动画效果

若要为曲面添加动画效果，请使用 `for` 循环更改绘图中的数据。若要替换曲面数据，请将曲面的 `XData`、`YData` 和 `ZData` 属性设置为新值。若要控制动画的速度，请在更新曲面数据后使用 `pause`。

```
scale = [linspace(0,1,20) linspace(1,-1,40)];  
  
for ii = 1:length(scale)  
  
    rho = radius + scale(ii)*amplitude*yy/order;  
  
    r = rho.*sin(theta);  
    x = r.*cos(phi);  
    y = r.*sin(phi);  
    z = rho.*cos(theta);  
  
    s.XData = x;  
    s.YData = y;  
    s.ZData = z;  
  
    pause(0.05)  
end
```



另请参阅

surf | lighting

标题和标签

- “为图添加标题和轴标签” (第 8-2 页)
- “将图例添加到图” (第 8-8 页)
- “向图中添加文本” (第 8-14 页)
- “向图中添加注释” (第 8-21 页)
- “图文本中的希腊字母和特殊字符” (第 8-25 页)
- “缩小图形标题” (第 8-34 页)

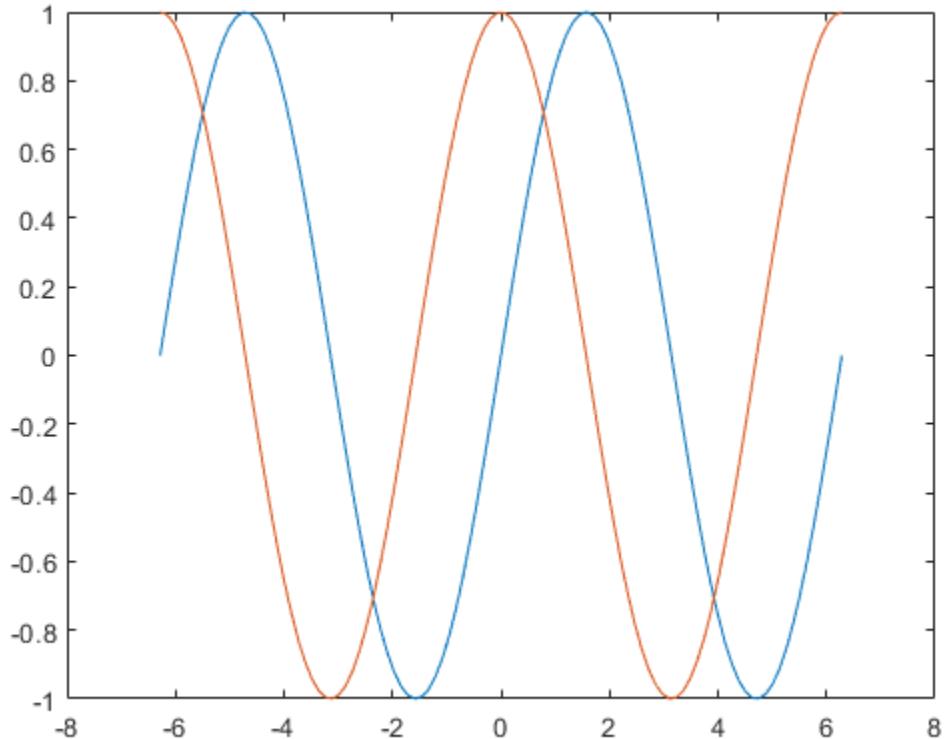
为图添加标题和轴标签

此示例说明如何使用 `title`、`xlabel` 和 `ylabel` 函数向图中添加标题和轴标签。它还说明如何通过更改字体大小来自定义坐标区文本的外观。

创建简单的线图

创建 `x`，它是 100 个介于 -2π 和 2π 之间的线性间隔值。将 `y1` 和 `y2` 创建为 `x` 的正弦和余弦值。绘制两组数据。

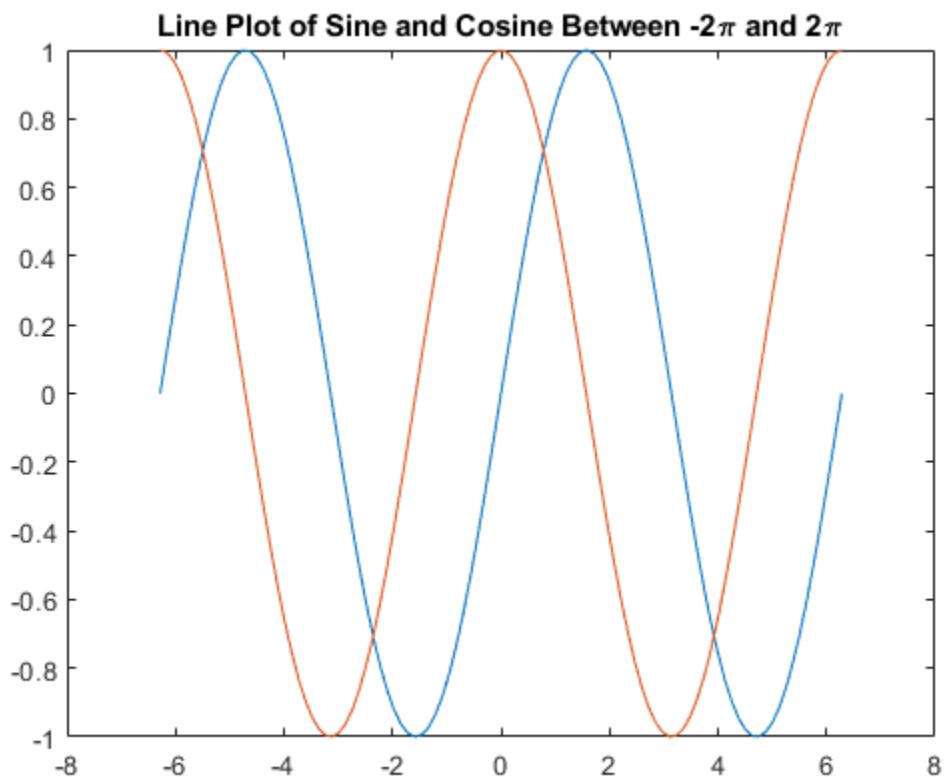
```
x = linspace(-2*pi,2*pi,100);
y1 = sin(x);
y2 = cos(x);
figure
plot(x,y1,x,y2)
```



添加标题

使用 `title` 函数向图中添加标题。要显示希腊符号 π ，请使用 TeX 标记 `\pi`。

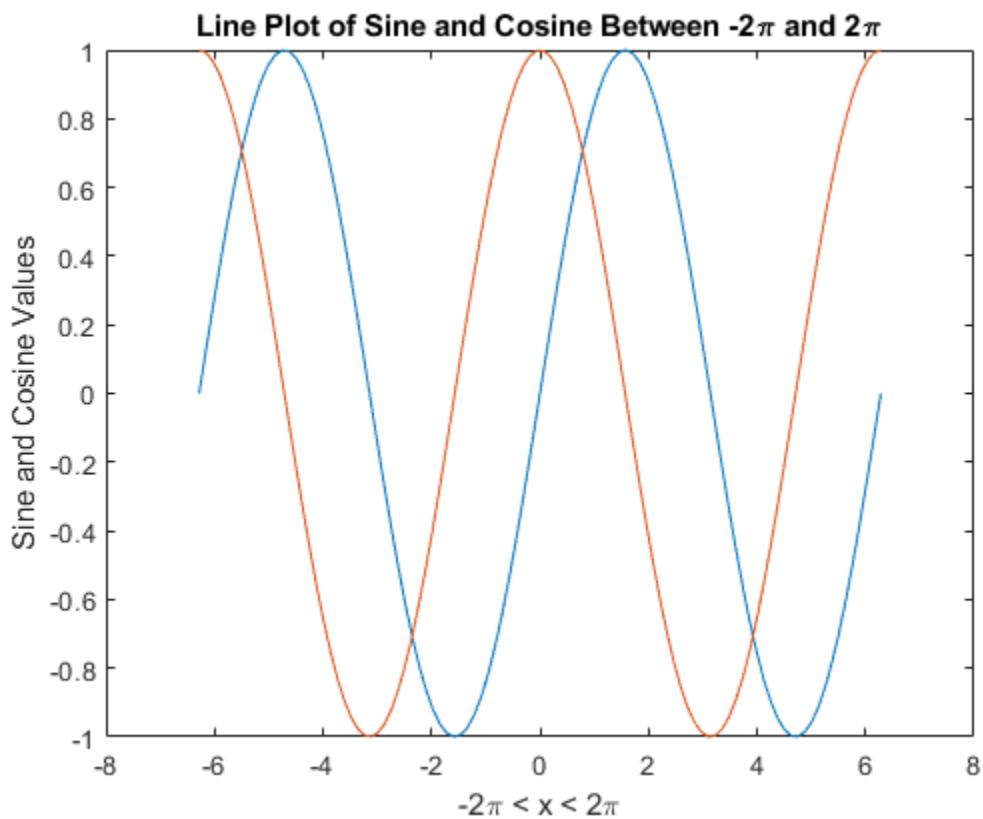
```
title('Line Plot of Sine and Cosine Between -2\pi and 2\pi')
```



添加坐标轴标签

使用 `xlabel` 和 `ylabel` 函数向图中添加轴标签。

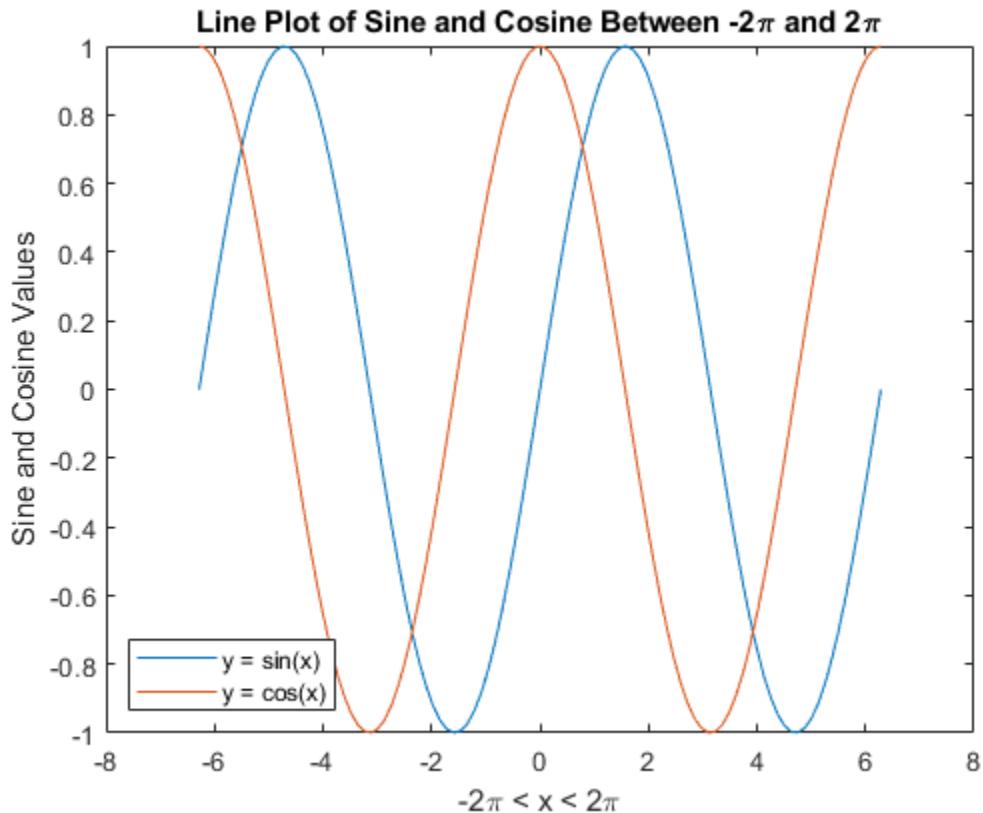
```
xlabel('-2\pi < x < 2\pi')
ylabel('Sine and Cosine Values')
```



添加图例

使用 `legend` 函数向图中添加标识每个数据集的图例。按照绘制线条的顺序指定图例说明。（可选）使用八个基本或斜角方位之一指定图例位置，在本例中为 '`southwest`'。

```
legend({'y = sin(x)', 'y = cos(x)'}, 'Location', 'southwest')
```

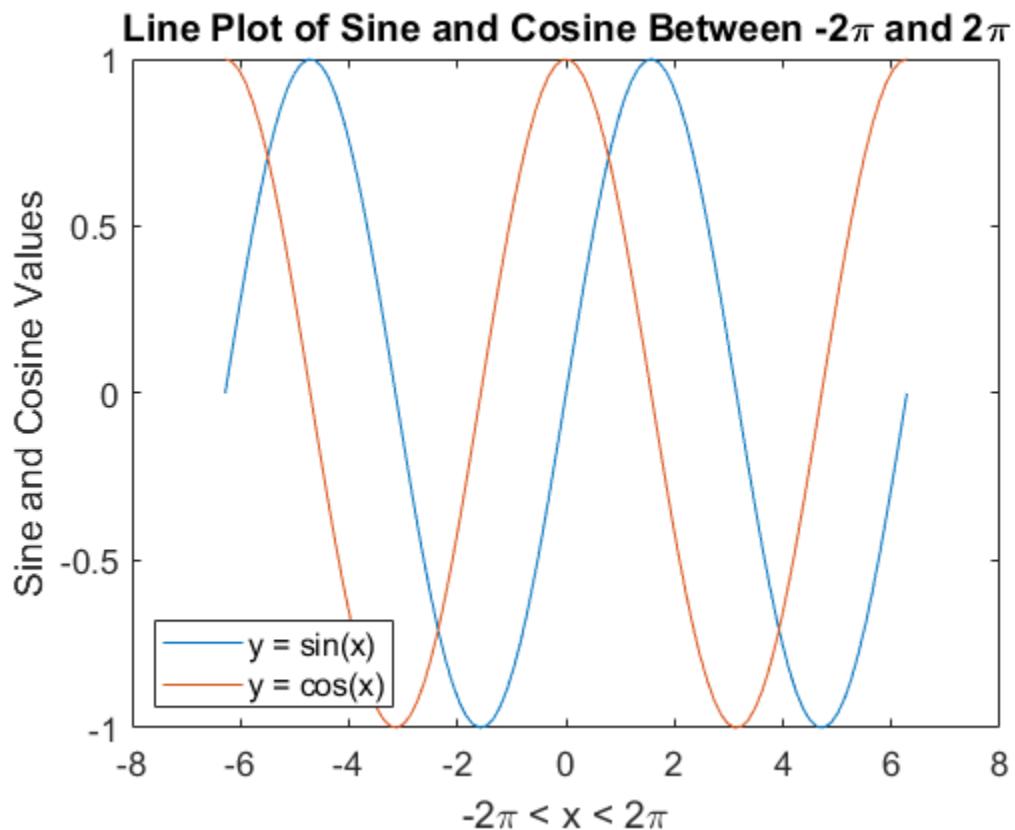


更改字体大小

Axes 对象具有可用来自定义坐标区外观的属性。例如，**FontSize** 属性控制标题、标签和图例的字体大小。

使用 **gca** 函数访问当前 Axes 对象。然后使用圆点表示法设置 **FontSize** 属性。

```
ax = gca;
ax.FontSize = 13;
```

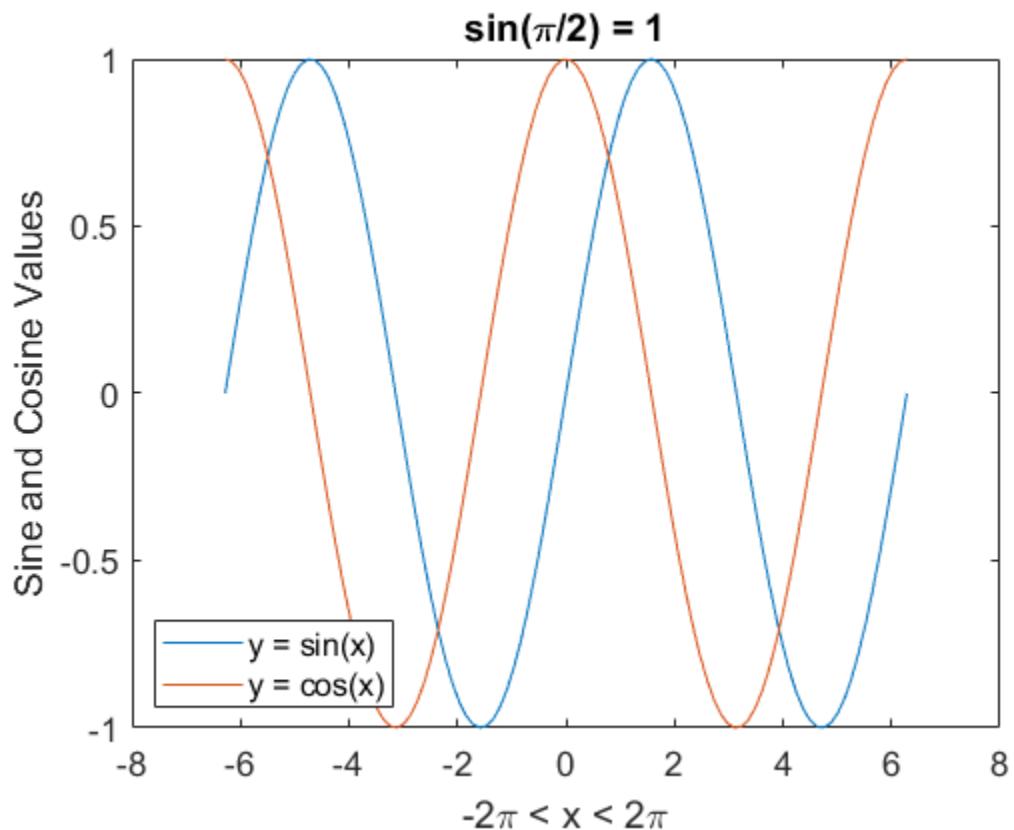


带有变量值的标题

通过使用 `num2str` 函数将值转换为文本，可在标题文本中包含变量值。您可以使用类似的方法为轴标签或图例条目添加变量值。

添加带有 $\sin(\pi)/2$ 值的标题。

```
k = sin(pi/2);  
title(['sin(\pi/2)' num2str(k)])
```



另请参阅

`title | xlabel | ylabel | legend | linspace`

相关示例

- “指定坐标轴范围” (第 9-2 页)
- “指定坐标轴刻度值和标签” (第 9-9 页)

将图例添加到图

图例是标记绘制在图上的数据序列的有用方法。下列示例说明如何创建图例并进行一些常见修改，例如更改位置、设置字体大小以及添加标题。您还可以创建具有多列的图例或为所绘制数据的子集创建图例。

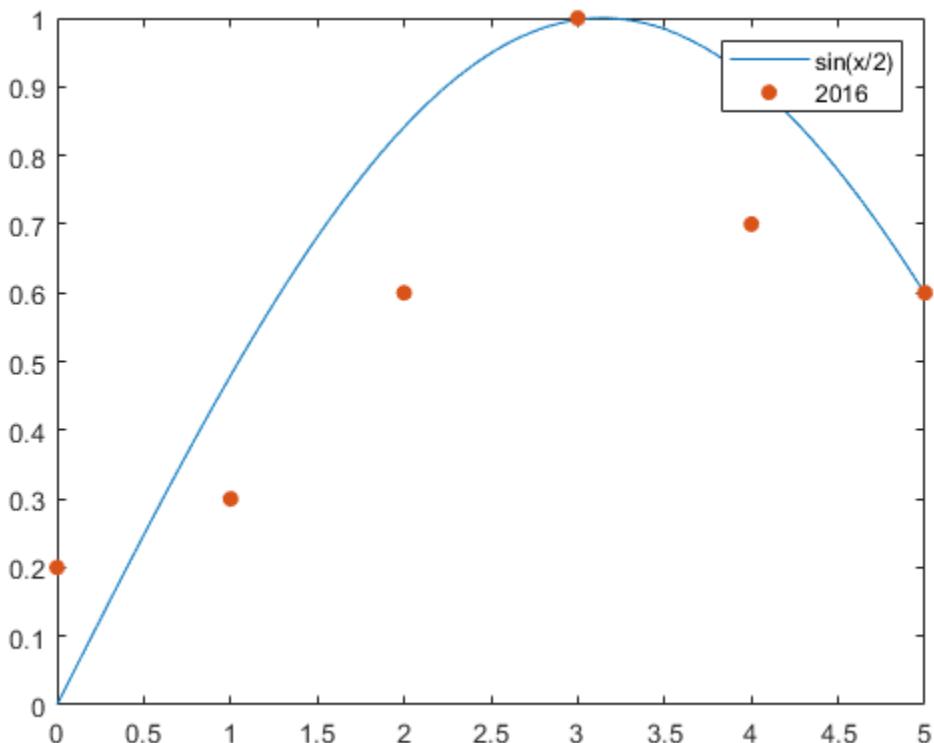
创建简单的图例

创建一个具有线图和散点图的图窗。为每个图添加具有说明的图例。将图例标签指定为 `legend` 函数的输入。

```
figure
x1 = linspace(0,5);
y1 = sin(x1/2);
plot(x1,y1)

hold on
x2 = [0 1 2 3 4 5];
y2 = [0.2 0.3 0.6 1 0.7 0.6];
scatter(x2,y2,'filled')
hold off

legend('sin(x/2)', '2016')
```



使用 `DisplayName` 指定标签

您也可以使用 `DisplayName` 属性指定图例标签。调用绘图函数时，将 `DisplayName` 属性设置为名称-值对组。然后，调用 `legend` 命令创建图例。

```

x1 = linspace(0,5);
y1 = sin(x1/2);
plot(x1,y1,'DisplayName','sin(x/2)')

hold on
x2 = [0 1 2 3 4 5];
y2 = [0.2 0.3 0.6 1 0.7 0.6];
scatter(x2,y2,'filled','DisplayName','2016')

legend

```

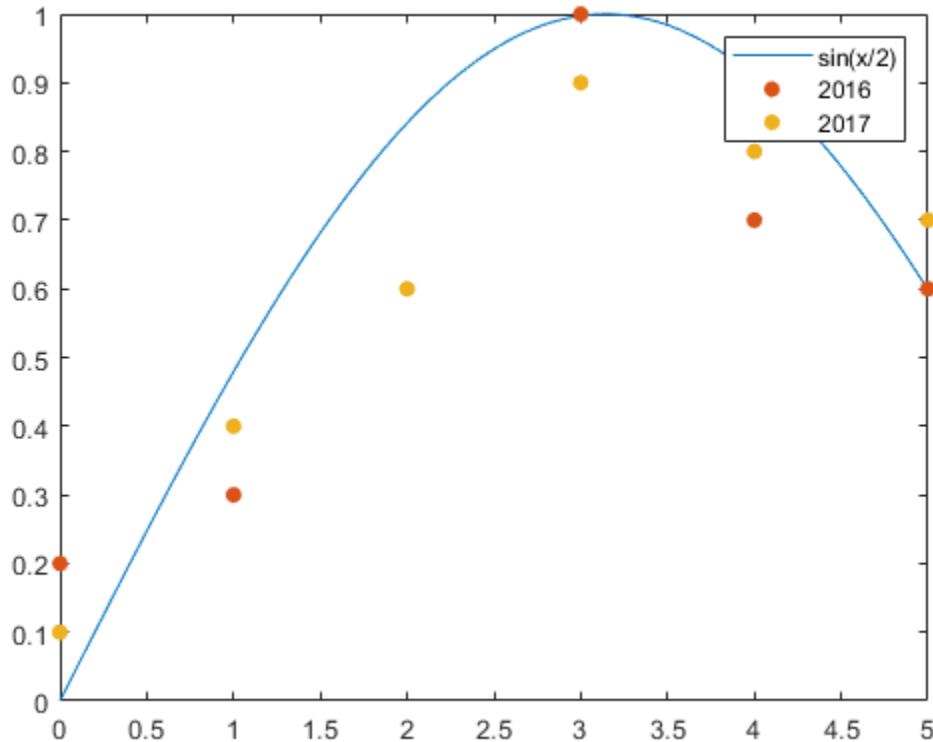
添加或删除数据序列时，图例会自动更新。如果您向坐标区添加更多数据，请使用 `DisplayName` 属性指定标签。如果不设置 `DisplayName` 属性，则图例使用 '`dataN`' 形式的标签。

添加 2017 年数据的散点图。

```

x3 = [0 1 2 3 4 5];
y3 = [0.1 0.4 0.6 0.9 0.8 0.7];
scatter(x3,y3,'filled','DisplayName','2017')
drawnow
hold off

```



自定义图例外观

`legend` 函数会创建一个 `Legend` 对象。`Legend` 对象具有可用于自定义图例外观的属性，如 `Location`、`Orientation`、`FontSize` 和 `Title` 属性。有关完整列表，请参阅 `Legend` 属性。

您可以通过两种方式设置属性：

- 在 `legend` 命令中使用名称-值对组。在大多数情况下，当您使用名称-值对组时，必须在元胞数组中指定标签，例如 `legend({'label1','label2'},'FontSize',14)`。
- 使用 `Legend` 对象。您可以将 `Legend` 对象作为 `legend` 函数的输出参数返回，例如 `lgd = legend`。然后，通过圆点表示法使用 `lgd` 来设置属性，如 `lgd.FontSize = 14`。

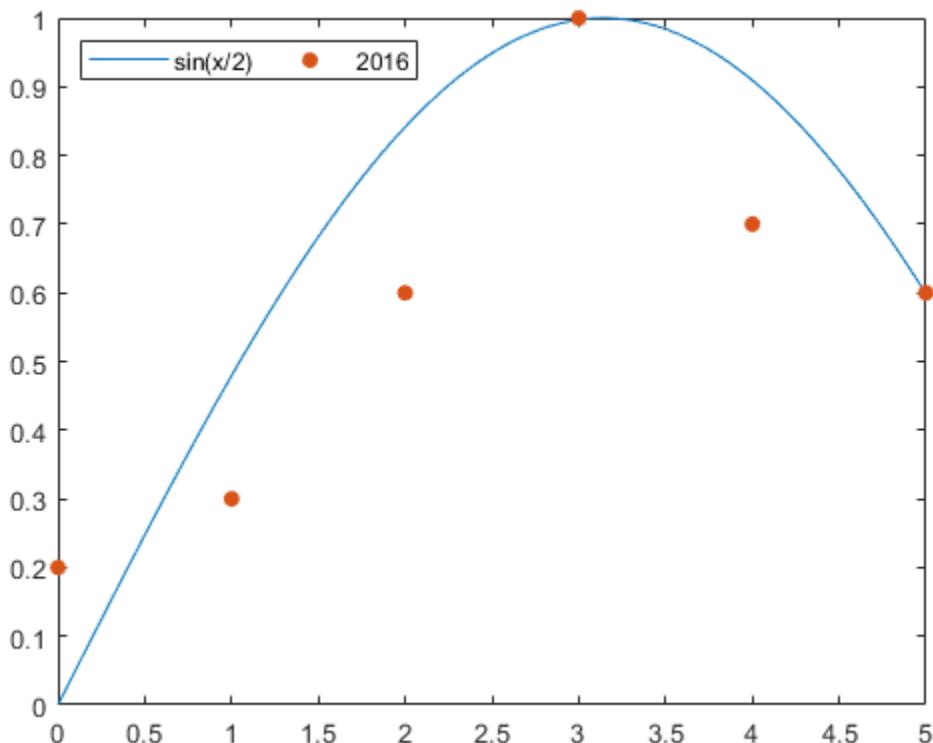
图例位置和方向

通过将 `Location` 和 `Orientation` 属性设置为名称-值对组来指定图例的位置和方向。将位置设置为八个基本及斜角方位之一，在本例中为 '`northwest`'。将方向设置为 '`vertical`'（默认）或 '`horizontal`'（在本例中为后者）。以元胞数组形式指定标签。

```
x1 = linspace(0,5);
y1 = sin(x1/2);
plot(x1,y1)

hold on
x2 = [0 1 2 3 4 5];
y2 = [0.2 0.3 0.6 1 0.7 0.6];
scatter(x2,y2,'filled')
hold off

legend({'sin(x/2)','2016'},'Location','northwest','Orientation','horizontal')
```



图例字体大小和标题

通过设置 `FontSize` 和 `Title` 属性来指定图例字体大小和标题。将 `Legend` 对象赋给变量 `lgd`。然后，通过圆点表示法使用 `lgd` 更改属性。

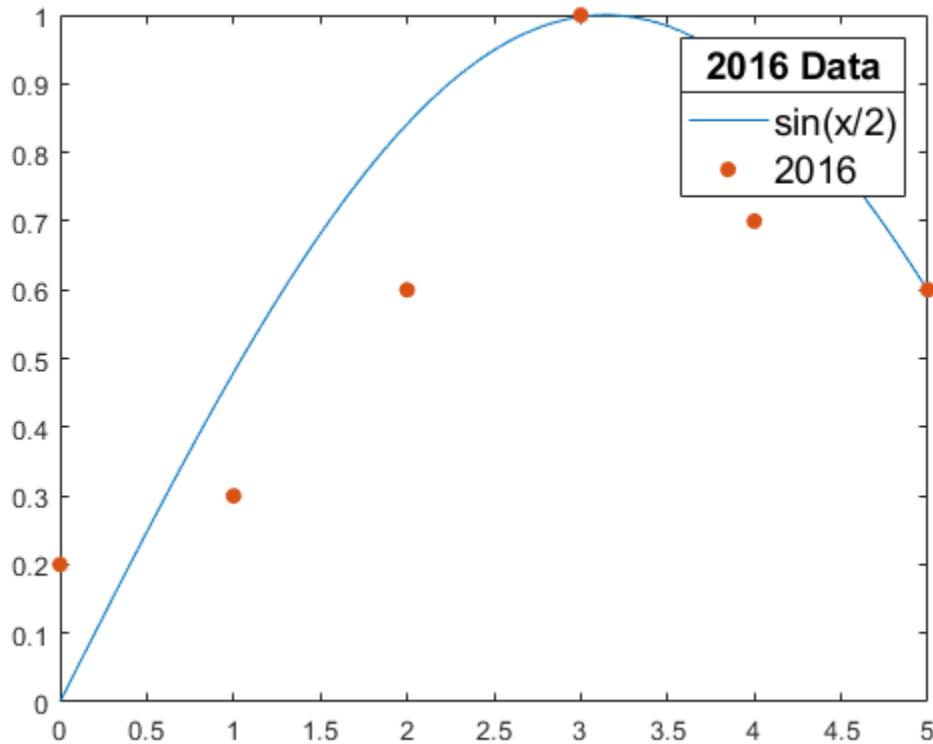
```

x1 = linspace(0,5);
y1 = sin(x1/2);
plot(x1,y1,'DisplayName','sin(x/2)')

hold on
x2 = [0 1 2 3 4 5];
y2 = [0.2 0.3 0.6 1 0.7 0.6];
scatter(x2,y2,'filled','DisplayName','2016')
hold off

lgd = legend;
lgd.FontSize = 14;
lgd.Title.String = '2016 Data';

```



具有多列的图例

创建一个包含六个线图的图。通过将 NumColumns 属性设置为 2 来添加一个具有两列的图例。

```

x = linspace(0,10);
y1 = sin(x);
y2 = sin(0.9*x);
y3 = sin(0.8*x);
y4 = sin(0.7*x);
y5 = sin(0.6*x);
y6 = sin(0.5*x);

plot(x,y1,'DisplayName','sin(x)')
hold on

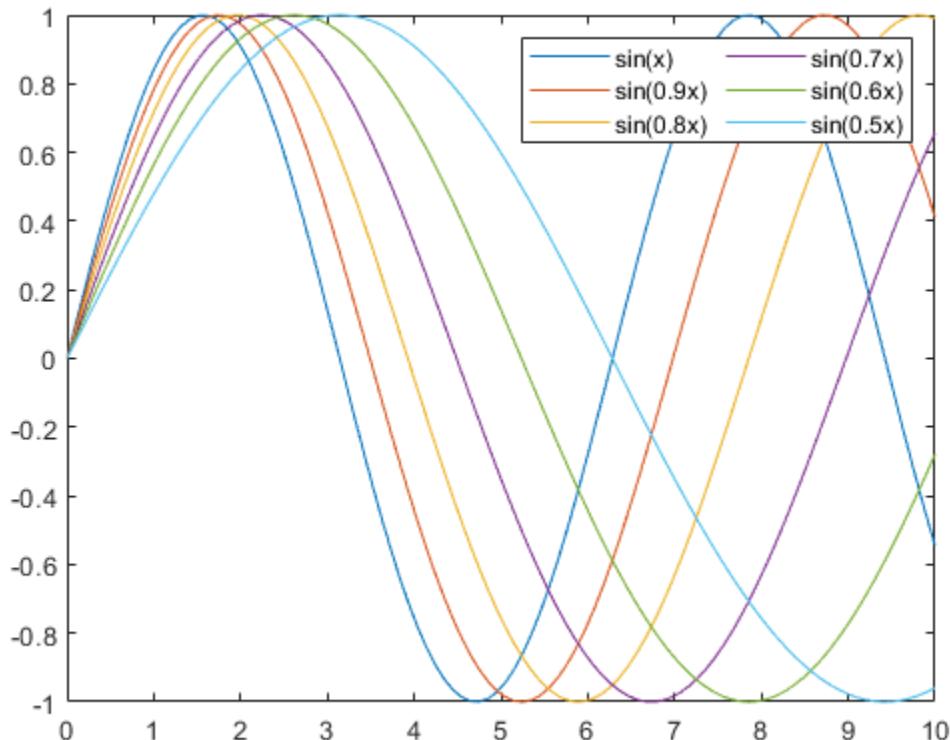
```

```

plot(x,y2,'DisplayName','sin(0.9x)')
plot(x,y3,'DisplayName','sin(0.8x)')
plot(x,y4,'DisplayName','sin(0.7x)')
plot(x,y5,'DisplayName','sin(0.6x)')
plot(x,y6,'DisplayName','sin(0.5x)')
hold off

lgd = legend;
lgd.NumColumns = 2;

```



在图例中包含图的子集

合并两个条形图和一个散点图。将 Bar 对象 b1 和 b2 指定为 legend 函数的第一个输入参数，创建一个仅包含条形图的图例。以向量形式指定对象。

```

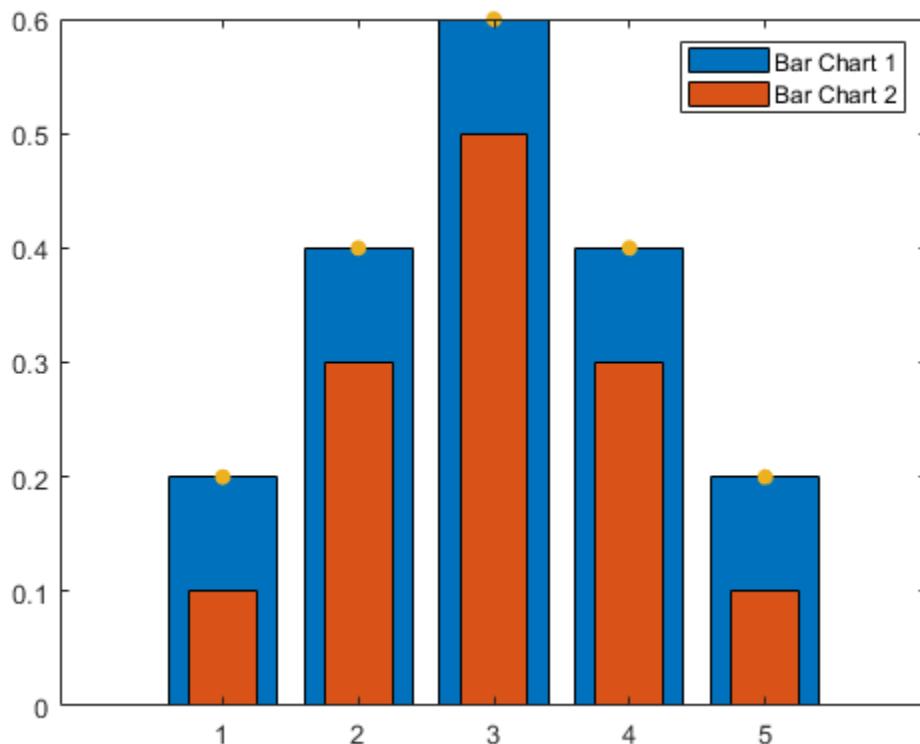
x = [1 2 3 4 5];
y1 = [.2 .4 .6 .4 .2];
b1 = bar(x,y1);

hold on
y2 = [.1 .3 .5 .3 .1];
b2 = bar(x,y2,'BarWidth',0.5);

y3 = [.2 .4 .6 .4 .2];
s = scatter(x,y3,'filled');
hold off

legend([b1 b2],'Bar Chart 1','Bar Chart 2')

```



另请参阅

legend | Legend 属性

向图中添加文本

此示例说明如何向图中添加文本、控制文本位置和大小以及创建多行文本。

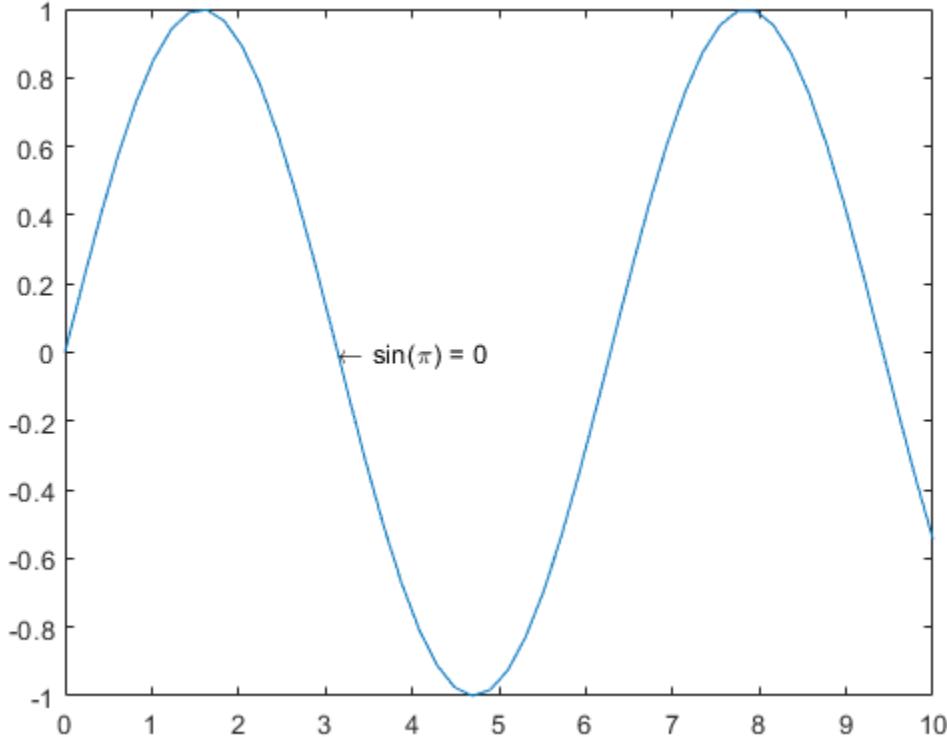
文本位置

使用 `text` 函数在特定数据点旁边添加文本。在本例中，为点 $(\pi, \sin(\pi))$ 添加文本。`text` 函数的前两个输入参数指定位置。第三个参数指定了文本。

默认情况下，`text` 支持一部分 TeX 标记。使用 TeX 标记 `\pi` 表示希腊字母 π 。通过包含 TeX 标记 `\leftarrow`，显示一个指向左侧的箭头。有关标记的完整列表，请参阅“图文本中的希腊字母和特殊字符”（第 8-25 页）。

```
x = linspace(0,10,50);
y = sin(x);
plot(x,y)

txt = '\leftarrow sin(\pi) = 0';
text(pi,sin(pi),txt)
```



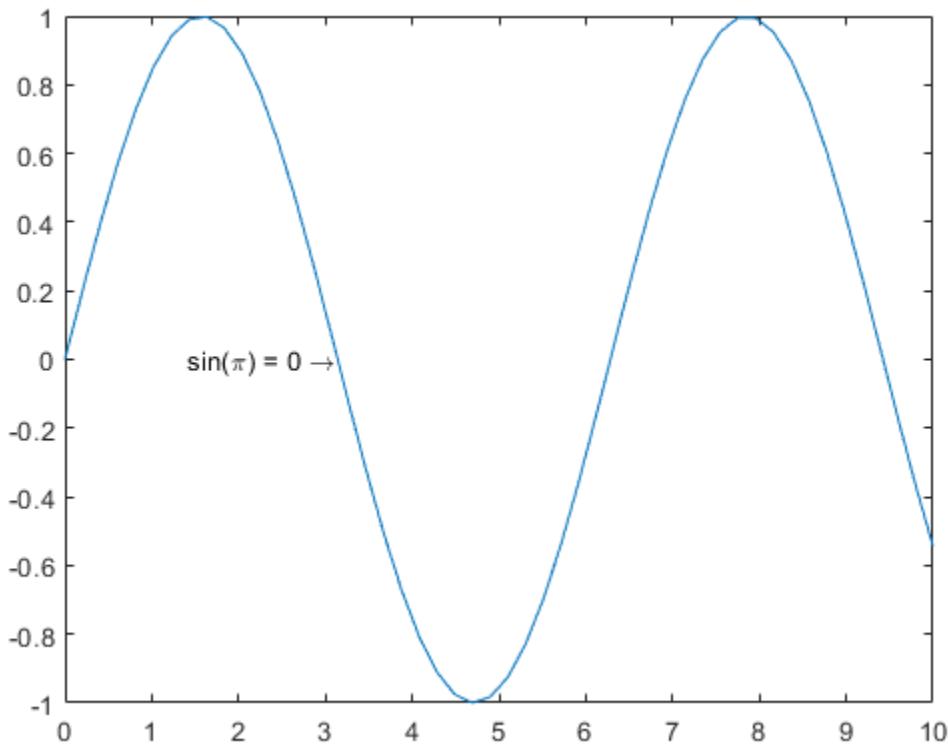
文本对齐方式

默认情况下，指定的数据点位于文本的左侧。通过将 `HorizontalAlignment` 属性指定为 `'right'`，使数据点出现在文本右侧。使用指向右侧而不是左侧的箭头。

```
x = linspace(0,10,50);
y = sin(x);
```

```
plot(x,y)

txt = 'sin(\pi) = 0 \rightarrow';
text(pi,sin(pi),txt,'HorizontalAlignment','right')
```

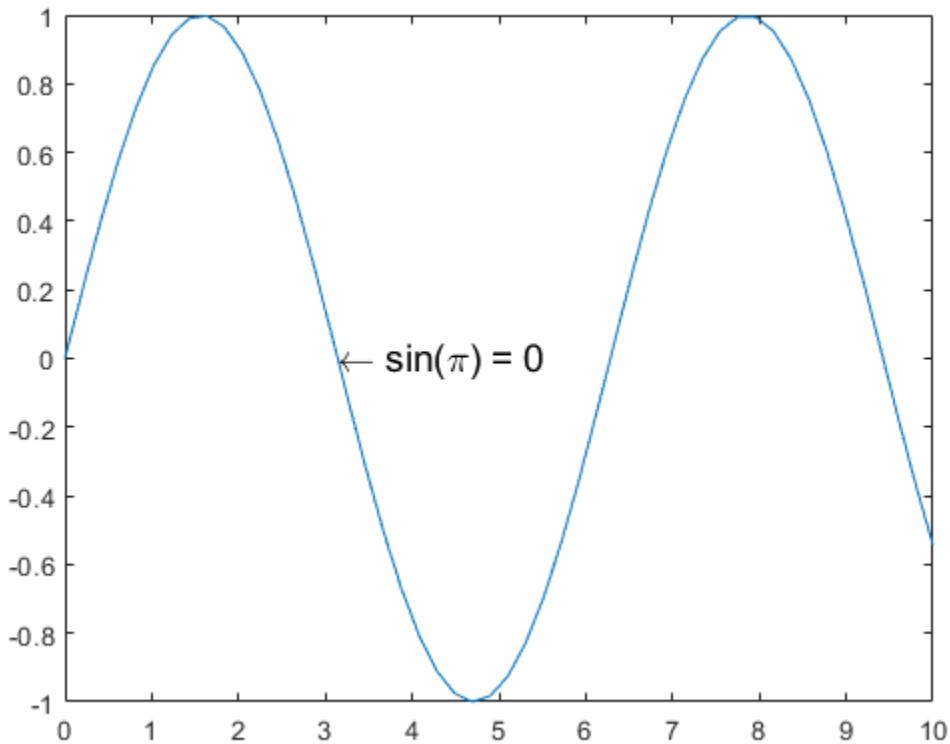


字体大小

通过将 `FontSize` 属性设置为 `text` 函数的名称-值对组参数，指定文本的字体大小。使用 `title`、`xlabel`、`ylabel` 或 `legend` 函数时，可以通过类似的方法更改字体大小。

```
x = linspace(0,10,50);
y = sin(x);
plot(x,y)

txt = '\leftarrow sin(\pi) = 0';
text(pi,sin(pi),txt,'FontSize',14)
```



设置文本属性

`text` 函数用于创建 `Text` 对象。`Text` 对象具有可用来自定义文本外观的属性，例如 `HorizontalAlignment` 或 `FontSize`。

您可以通过两种方式设置属性：

- 在 `text` 命令中使用名称-值对组，例如 `'FontSize',14`。
- 使用 `Text` 对象。您可以将 `Text` 对象作为 `text` 函数的输出参数返回，并将其赋给某个变量，例如 `t`。然后，使用圆点表示法设置属性，例如 `t.FontSize = 14`。

对于此示例，使用圆点表示法而不是名称-值对组来更改字体大小。

```

x = linspace(0,10,50);
y = sin(x);
plot(x,y)

txt = '\leftarrow sin(\pi) = 0';
t = text(pi,sin(pi),txt)

t =
Text (\leftarrow sin(\pi) = 0) with properties:

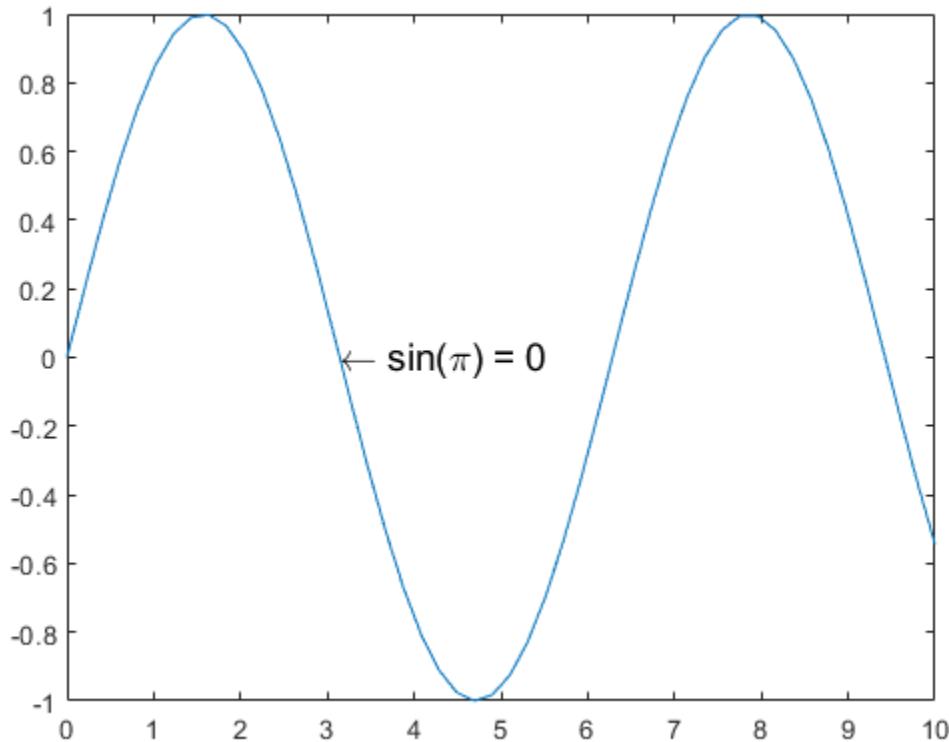
    String: '\leftarrow sin(\pi) = 0'
    FontSize: 10
    FontWeight: 'normal'
    FontName: 'Helvetica'

```

Color: [0 0 0]
 HorizontalAlignment: 'left'
 Position: [3.1416 1.2246e-16 0]
 Units: 'data'

Show all properties

```
t.FontSize = 14;
```

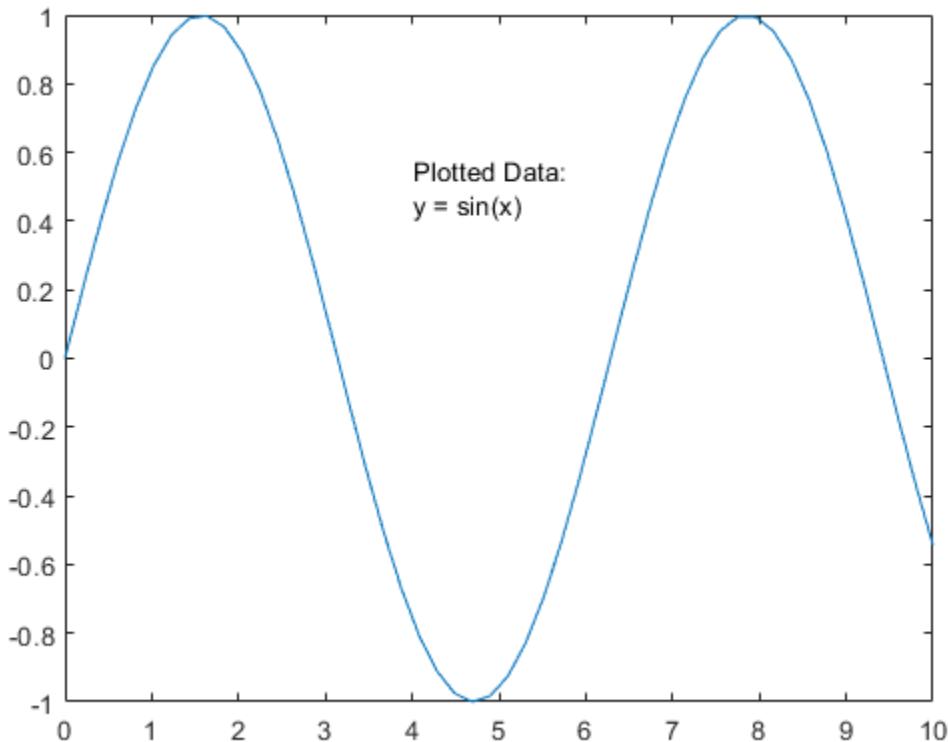


多行文本

使用字符串向量元胞数组显示跨越多行的文本。元胞数组的每个元素代表一行文本。对于此示例，显示包含两行的标题。使用 `title`、`xlabel`、`ylabel` 或 `legend` 函数时，可以通过类似的方法显示多行文本。

```
x = linspace(0,10,50);
y = sin(x);
plot(x,y)

txt = {'Plotted Data:','y = sin(x)'};
text(4,0.5,txt)
```

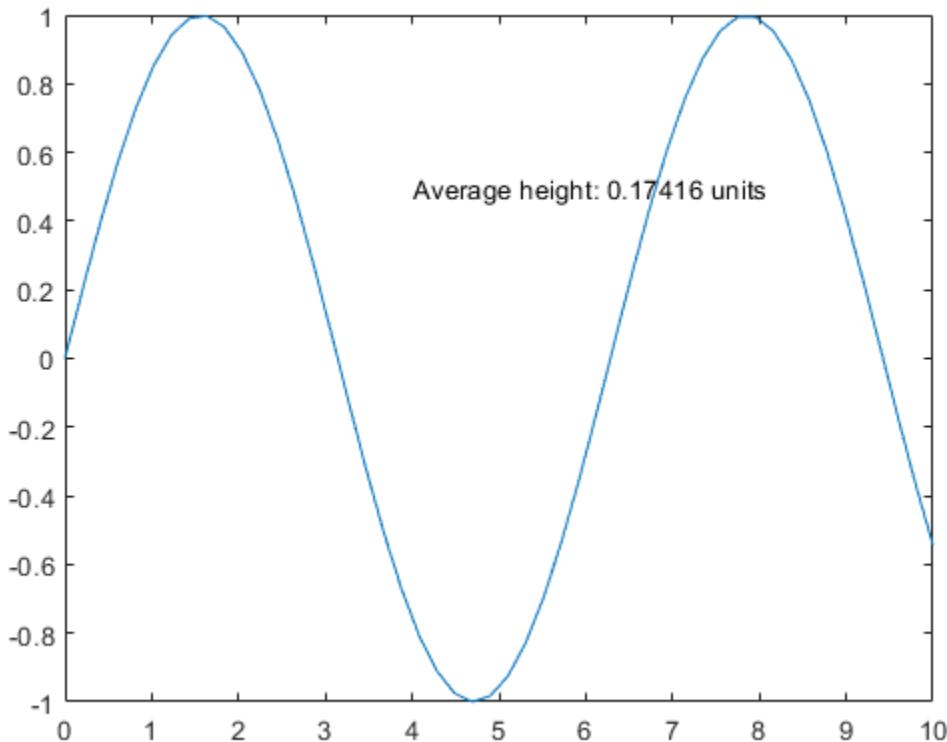


带有变量值的文本

通过使用 `num2str` 函数将数字转换为文本，可在文本中包含变量值。对于此示例，计算均值 y 并在标题中包含该值。使用 `title`、`xlabel`、`ylabel` 或 `legend` 函数时，可以通过类似的方法包含变量值。

```
x = linspace(0,10,50);
y = sin(x);
plot(x,y)

avg = mean(y);
txt = ['Average height: ' num2str(avg) ' units'];
text(4,0.5,txt)
```

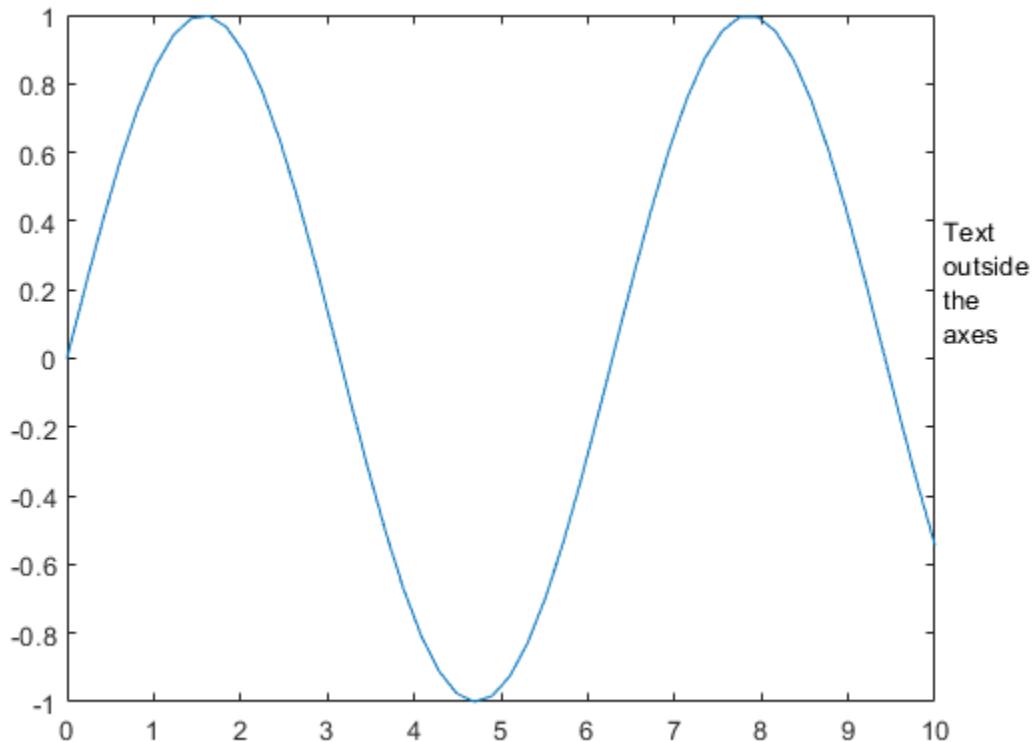


坐标区外部的文本

使用 `annotation` 函数而不是 `text` 函数，可在图窗内的任何位置添加文本。第一个输入参数指定注释的类型。第二个输入参数以归一化的图窗单位指定注释的位置。通过将 `EdgeColor` 属性设置为 `'none'`，删除文本框边框。有关文本框注释的详细信息，请参阅 `annotation` 函数。

```
x = linspace(0,10,50);
y = sin(x);
plot(x,y)

annotation('textbox',[.9 .5 .1 .2], ...
    'String','Text outside the axes','EdgeColor','none')
```



另请参阅

[text](#) | [title](#) | [xlabel](#) | [ylabel](#) | [annotation](#)

相关示例

- “图文本中的希腊字母和特殊字符” (第 8-25 页)

向图中添加注释

注释是添加到图中的额外信息，用来帮助标识一些重要信息。此示例首先解释不同类型的注释，然后说明如何向图中添加圆圈和文本箭头。

注释的类型

使用 **annotation** 函数可以向图中添加注释。函数的第一个输入指定要创建的注释的类型。

- 如果将类型指定为 'line'、'arrow'、'doublearrow' 或 'textarrow'，则第二个输入是注释的起点和终点 x 位置。第三个输入是注释的起点和终点 y 位置。例如，`annotation('line',[x_begin x_end], [y_begin y_end])`。
- 如果将类型指定为 'rectangle'、'ellipse' 或 'textbox'，则第二个参数是位置和大小。例如，`annotation('rectangle',[x y w h])`。

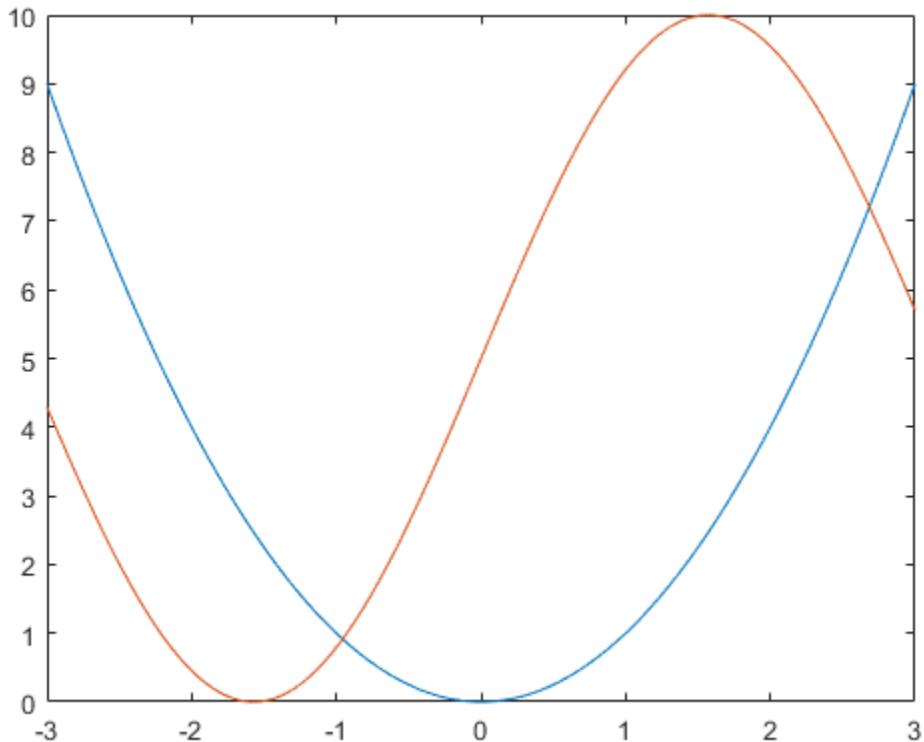
注释使用归一化的图窗单位，并且可以在一个图窗中跨越多个坐标区。

创建简单的图

定义和绘制函数 $f(x)$ 和 $g(x)$ 。

```
x = -3.0:0.01:3.0;
f = x.^2;
g = 5*sin(x) + 5;
```

```
figure
plot(x,f)
hold on
plot(x,g)
hold off
```



圆圈注释

在图中添加一个圆圈，以突出显示 $f(x)$ 等于 $g(x)$ 的位置。要创建圆圈，请使用 'ellipse' 选项作为注释类型。

通过设置底层对象的属性来自定义圆圈。将 **Ellipse** 对象作为 **annotation** 函数的输出参数返回。然后，使用圆点表示法访问对象的属性。例如，设置 **Color** 属性。

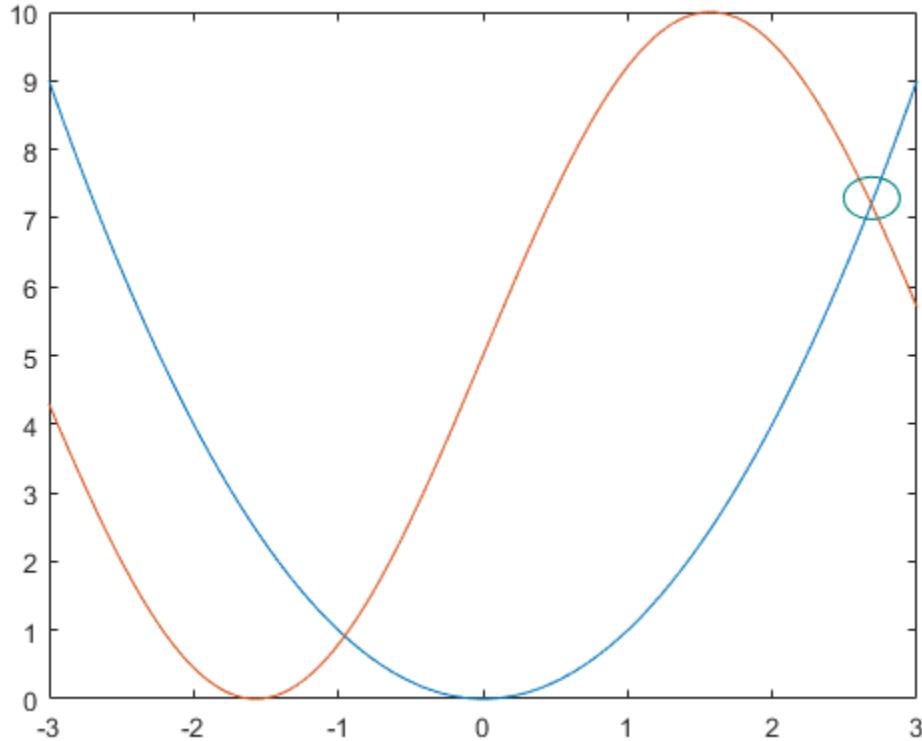
```
elps = annotation('ellipse',[.84 .68 .05 .05])
```

```
elps =
  Ellipse with properties:
```

```
  Color: [0 0 0]
  FaceColor: 'none'
  LineStyle: '-'
  LineWidth: 0.5000
  Position: [0.8400 0.6800 0.0500 0.0500]
  Units: 'normalized'
```

[Show all properties](#)

```
elps.Color = [0 0.5 0.5];
```



文本箭头注释

使用 'textarrow' 选项作为注释类型，向图中添加一个文本箭头。

您可以通过设置底层对象的属性来自定义文本箭头。将 TextArrow 对象作为 annotation 函数的输出参数返回。然后，使用圆点表示法访问对象的属性。例如，将 String 属性设置为所需的文本，将 Color 属性设置为颜色值。

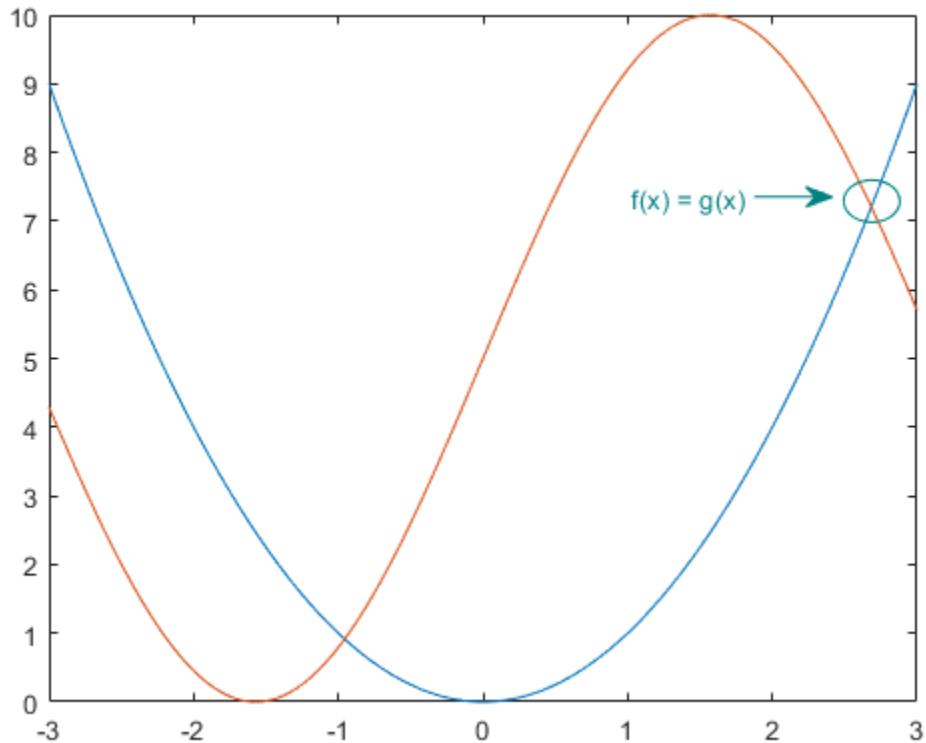
```
ta = annotation('textarrow', [0.76 0.83], [0.71 0.71])
```

```
ta =
TextArrow with properties:
```

```
String: {}
FontName: 'Helvetica'
FontSize: 10
Color: [0 0 0]
TextColor: [0 0 0]
LineStyle: '-'
LineWidth: 0.5000
HeadStyle: 'vback2'
Position: [0.7600 0.7100 0.0700 0]
Units: 'normalized'
X: [0.7600 0.8300]
Y: [0.7100 0.7100]
```

Show all properties

```
ta.String = 'f(x) = g(x)';  
ta.Color = [0 0.5 0.5];
```



另请参阅

[text | annotation](#)

相关示例

- “图文本中的希腊字母和特殊字符” (第 8-25 页)

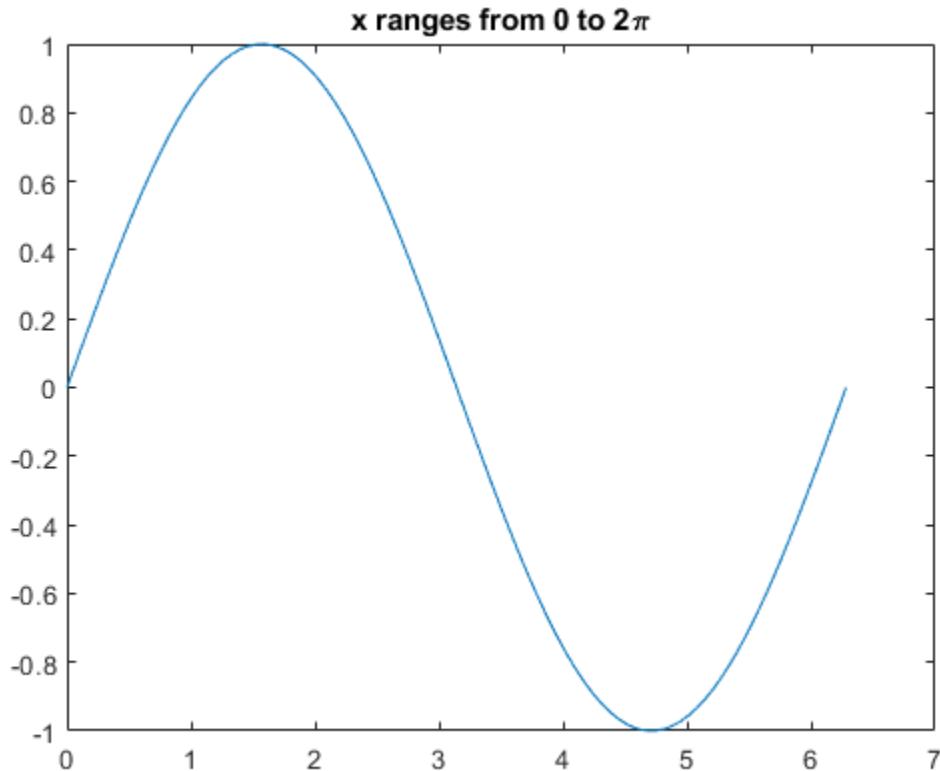
图文本中的希腊字母和特殊字符

您可以使用 TeX 标记向图中添加包含希腊字母和特殊字符的文本。此外，还可以使用 TeX 标记添加上标、下标以及修改文本类型和颜色。默认情况下，MATLAB 支持一部分 TeX 标记。要使用其他特殊字符，如积分和求和符号，可以改用 LaTeX 标记。此示例说明如何向图文本中插入希腊字母、上标和注释，并解释其他可用的 TeX 选项。

包含希腊字母

创建一个简单的线图并添加标题。使用 TeX 标记 `\pi` 在标题中包含希腊字母 π 。

```
x = linspace(0,2*pi);
y = sin(x);
plot(x,y)
title('x ranges from 0 to 2\pi')
```

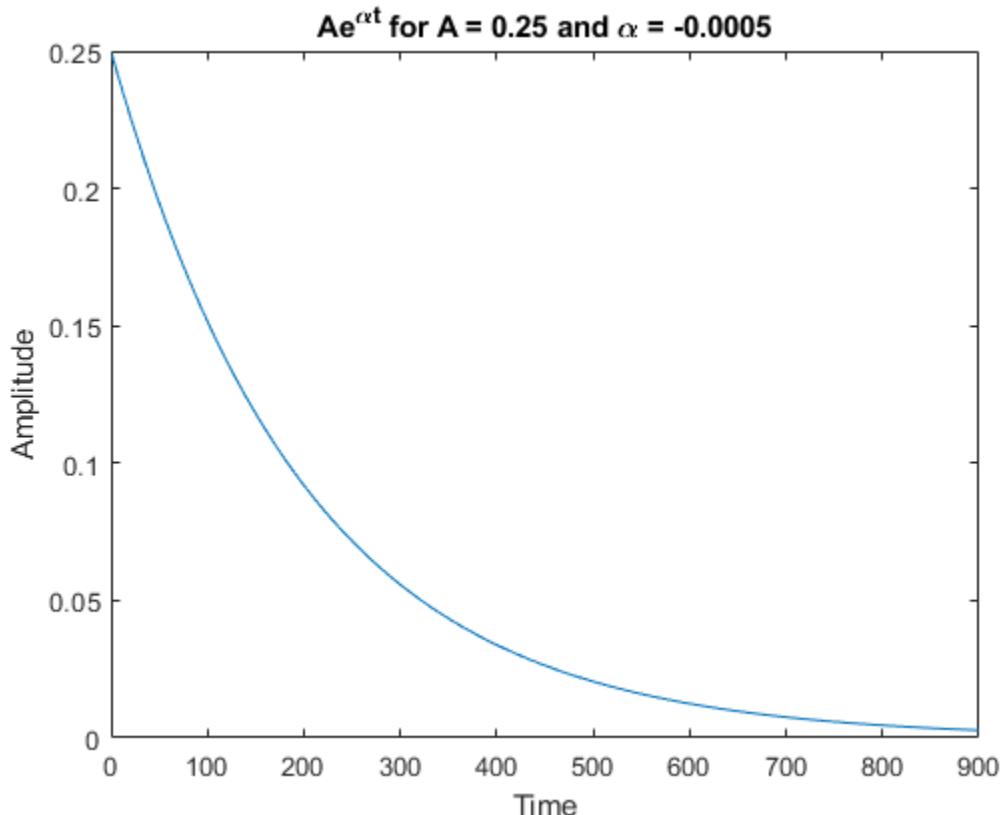


包含上标和注释

创建一个线图并向图中添加标题和轴标签。使用 `^` 字符在标题上显示上标。`^` 字符会修改紧随其后的字符。用花括号 {} 包含多个字符以将这些字符放入上标中。分别使用 TeX 标记 `\alpha` 和 `\mu` 在文本中包含希腊字母 α 和 μ 。

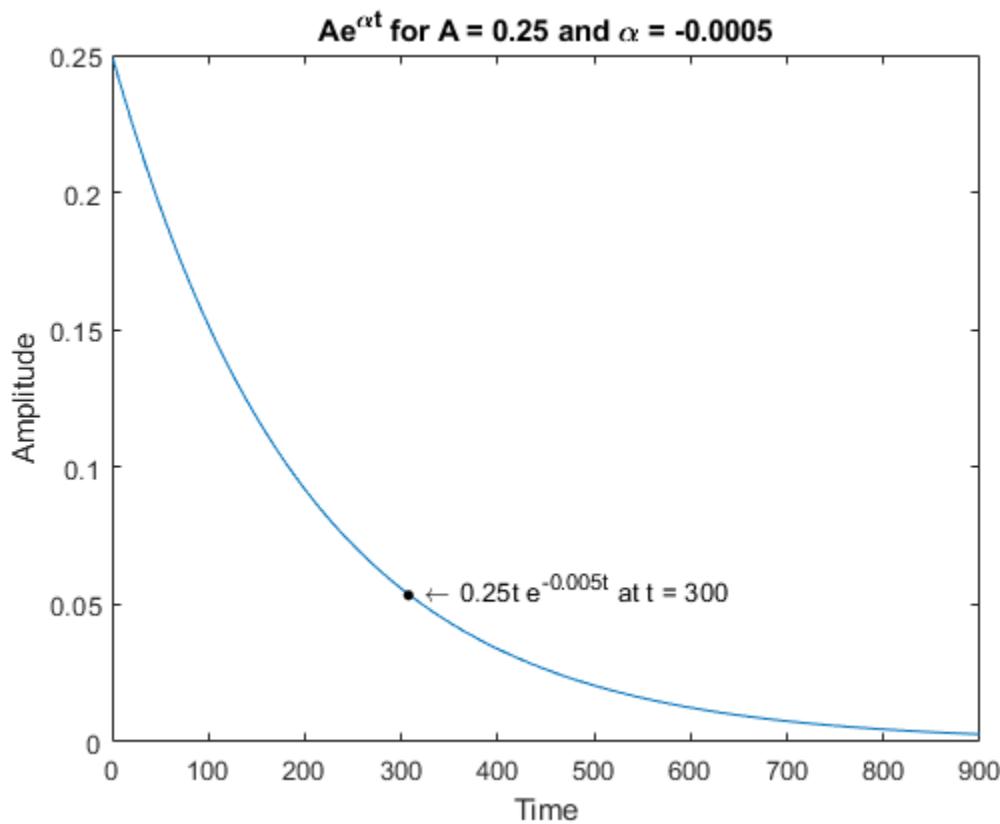
```
t = 1:900;
y = 0.25*exp(-0.005*t);
```

```
figure  
plot(t,y)  
title('Ae^{\hat{\alpha}} for A = 0.25 and \alpha = -0.0005')  
xlabel('Time')  
ylabel('Amplitude')
```



向 $t = 300$ 处的数据点添加文本。使用 TeX 标记 `\bullet` 向指定点添加标记，并使用 `\leftarrow` 包含一个指向左侧的箭头。默认情况下，指定的数据点位于文本的左侧。

```
txt = '\bullet \leftarrow 0.25t e^{-0.005t} at t = 300';  
text(t(300),y(300),txt)
```



TeX 标记选项

MATLAB 支持部分 TeX 标记。使用 TeX 标记可添加上标和下标、修改文本类型和颜色以及包含特殊字符。只要文本对象的 `Interpreter` 属性设置为 'tex' (默认值)，MATLAB 便会解释 TeX 标记。

修饰符会一直作用到文本结尾，但上标和下标除外，因为它们仅修饰下一个字符或花括号中的字符。当您将解释器设置为 'tex' 时，支持的修饰符如下所示。

修饰符	说明	示例
<code>^{ } </code>	上标	'text^{superscript}'
<code>_{} </code>	下标	'text_{subscript}'
<code>\bf </code>	粗体	'\bf text'
<code>\it </code>	斜体	'\it text'
<code>\sl </code>	伪斜体 (通常与斜体相同)	'\sl text'
<code>\rm </code>	常规字体	'\rm text'
<code>\fontname{specifier} </code>	字体名称 - 将 <code>specifier</code> 替换为字体系列的名称。您可以将此说明符与其他修饰符结合使用。	'\fontname{Courier} text'
<code>\fontsize{specifier} </code>	字体大小 - 将 <code>specifier</code> 替换为以磅为单位的数值标量值。	'\fontsize{15} text'

修饰符	说明	示例
<code>\color{specifier}</code>	字体颜色 - 将 specifier 替换为以下颜色之一: red、green、yellow、magenta、blue、black、white、gray、darkGreen、orange 或 lightBlue。	'\color{magenta} text'
<code>\color[rgb]{specifier}</code>	自定义字体颜色 - 将 specifier 替换为三元素 RGB 三元组。	'\color[rgb]{0,0.5,0.5} text'

下表列出了 'tex' 解释器所支持的特殊字符。

字符序列	符号	字符序列	符号	字符序列	符号
<code>\alpha</code>	α	<code>\upsilon</code>	υ	<code>\sim</code>	\sim
<code>\angle</code>	\angle	<code>\phi</code>	ϕ	<code>\leq</code>	\leq
<code>\ast</code>	$*$	<code>\chi</code>	χ	<code>\infty</code>	∞
<code>\beta</code>	β	<code>\psi</code>	ψ	<code>\clubsuit</code>	\clubsuit
<code>\gamma</code>	γ	<code>\omega</code>	ω	<code>\diamondsuit</code>	\diamondsuit
<code>\delta</code>	δ	<code>\Gamma</code>	Γ	<code>\heartsuit</code>	\heartsuit
<code>\epsilon</code>	ϵ	<code>\Delta</code>	Δ	<code>\spadesuit</code>	\spadesuit
<code>\zeta</code>	ζ	<code>\Theta</code>	Θ	<code>\leftrightarrow</code>	\leftrightarrow
<code>\eta</code>	η	<code>\Lambda</code>	Λ	<code>\leftarrow</code>	\leftarrow
<code>\theta</code>	θ	<code>\Xi</code>	Ξ	<code>\Leftarrow</code>	\Leftarrow
<code>\vartheta</code>	ϑ	<code>\Pi</code>	Π	<code>\uparrow</code>	\uparrow
<code>\iota</code>	ι	<code>\Sigma</code>	Σ	<code>\rightarrow</code>	\rightarrow
<code>\kappa</code>	κ	<code>\Upsilon</code>	Υ	<code>\Rightarrow</code>	\Rightarrow
<code>\lambda</code>	λ	<code>\Phi</code>	Φ	<code>\downarrow</code>	\downarrow
<code>\mu</code>	μ	<code>\Psi</code>	Ψ	<code>\circ</code>	\circ
<code>\nu</code>	ν	<code>\Omega</code>	Ω	<code>\pm</code>	\pm
<code>\xi</code>	ξ	<code>\forall</code>	\forall	<code>\geq</code>	\geq
<code>\pi</code>	π	<code>\exists</code>	\exists	<code>\propto</code>	\propto
<code>\rho</code>	ρ	<code>\ni</code>	\ni	<code>\partial</code>	∂
<code>\sigma</code>	σ	<code>\cong</code>	\cong	<code>\bullet</code>	\bullet
<code>\varsigma</code>	ς	<code>\approx</code>	\approx	<code>\div</code>	\div
<code>\tau</code>	τ	<code>\Re</code>	\Re	<code>\neq</code>	\neq
<code>\equiv</code>	\equiv	<code>\oplus</code>	\oplus	<code>\aleph</code>	\aleph
<code>\Im</code>	\Im	<code>\cup</code>	\cup	<code>\wp</code>	\wp
<code>\otimes</code>	\otimes	<code>\subseteq</code>	\subseteq	<code>\oslash</code>	\oslash
<code>\cap</code>	\cap	<code>\in</code>	\in	<code>\supseteq</code>	\supseteq

字符序列	符号	字符序列	符号	字符序列	符号
\supset	\supset	\lceil	\lceil	\subset	\subset
\int	\int	\cdot	\cdot	\circ	\circ
\rfloor	\rfloor	\neg	\neg	\nabla	∇
\lfloor	\lfloor	\times	\times	\ldots	\ldots
\perp	\perp	\surd	\surd	\prime	\prime
\wedge	\wedge	\varpi	ϖ	\emptyset	\emptyset
\rceil	\rceil	\rangle	\rangle	\mid	\mid
\vee	\vee	\langle	\langle	\copyright	\circledC

用 LaTeX 创建文本

默认情况下，MATLAB 可以解析使用 TeX 标记的文本。但是，要获得更多格式设置选项，您可以改用 LaTeX 标记。

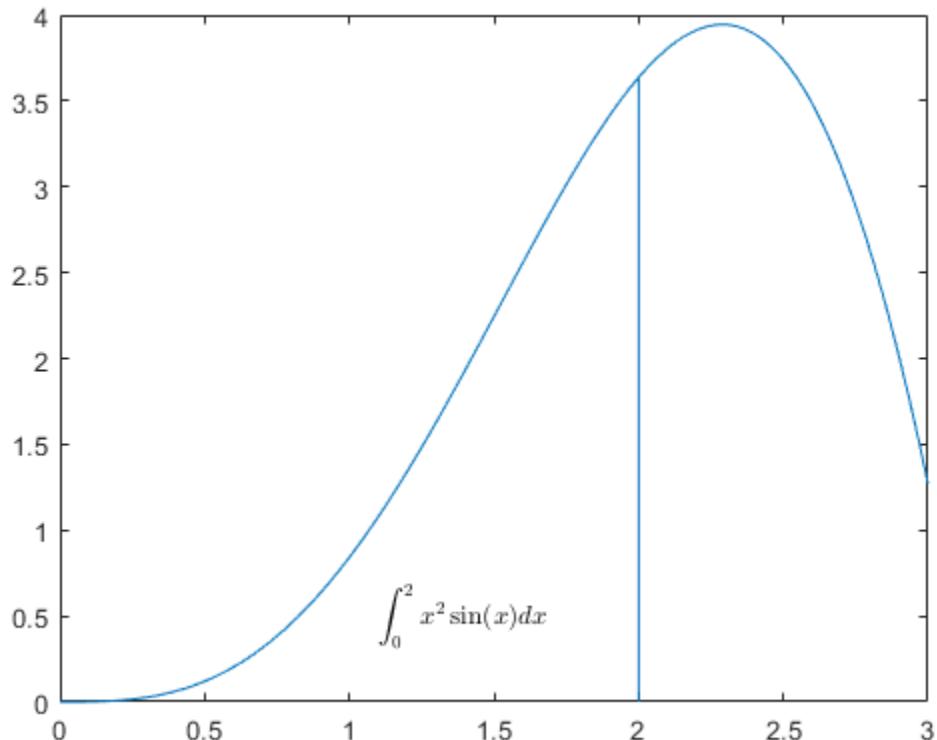
例如，绘制 $y = x^2 \sin(x)$ 并在 $x = 2$ 处绘制一条垂直线。使用 LaTeX 标记向图中添加包含积分表达式的文本。要在显示模式下显示表达式，请用双美元符号 (\$\$) 将标记括起来。调用 `text` 函数时，请将 `Interpreter` 属性设置为 'latex'。

```

x = linspace(0,3);
y = x.^2.*sin(x);
plot(x,y)
line([2,2],[0,2^2*sin(2)])  
  

str = '$$ \int_{0}^{2} x^2 \sin(x) dx $$';
text(1.1,0.5,str,'Interpreter','latex')

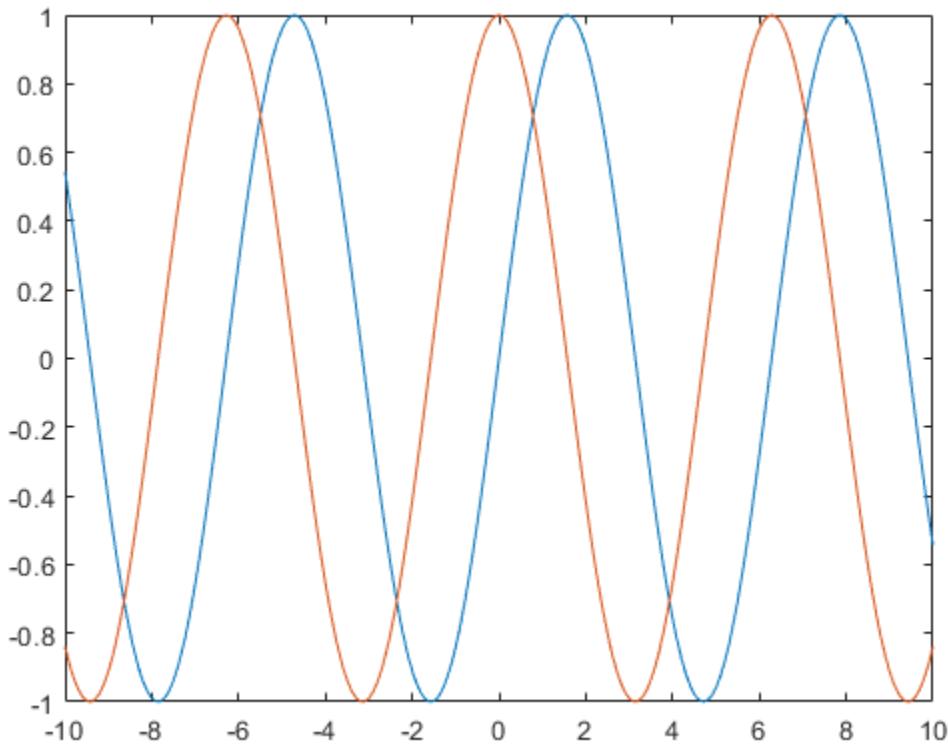
```



使用 LaTeX 创建绘图标题、刻度标签和图例

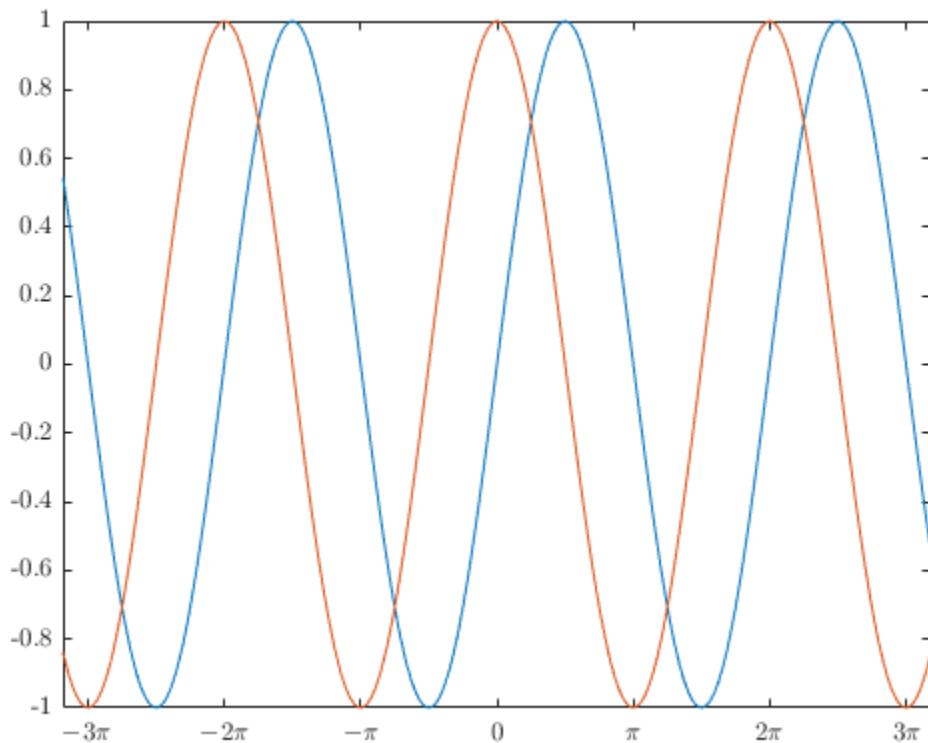
您可以在绘图标题、刻度标签和图例中使用 LaTeX 标记。例如，创建正弦波和余弦波的绘图。

```
x = -10:0.1:10;  
y = [sin(x); cos(x)];  
plot(x,y)
```



通过调用 `xticks` 函数，将 x 轴刻度值设置为 `pi` 的倍数。然后，调用 `gca` 函数以获取当前坐标区，并将 `TickLabelInterpreter` 属性设置为 `'latex'`。使用 LaTeX 标记指定刻度标签。对于行内表达式，请用单个美元符号 (\$) 括起标记。

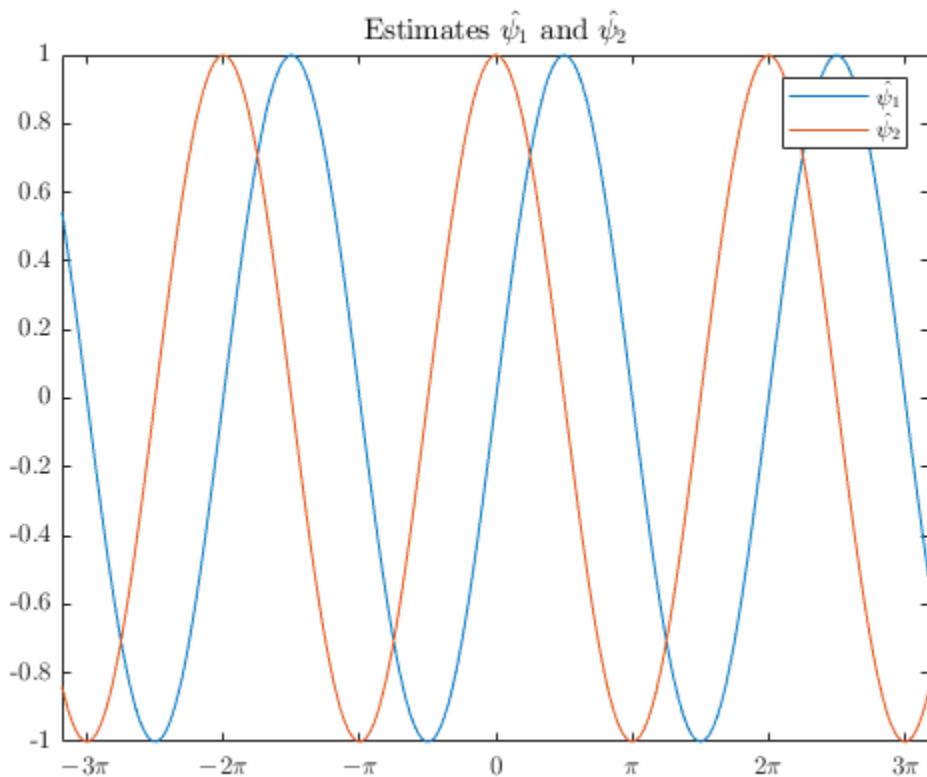
```
xticks([-3*pi -2*pi -pi 0 pi 2*pi 3*pi])
ax = gca;
ax.TickLabelInterpreter = 'latex';
xticklabels({'$-3\pi$', '$-2\pi$', '$-\pi$', '0', '$\pi$', '$2\pi$', '$3\pi$'});
```



通过调用 `title` 函数并将 `Interpreter` 属性设置为 `'latex'`, 添加包含 LaTeX 标记的标题。同样, 创建一个标签包含 LaTeX 标记的图例。

```
% Add title
str = 'Estimates $\hat{\psi}_1$ and $\hat{\psi}_2$';
title(str,'Interpreter','latex')

% Add legend
label1 = '$\hat{\psi}_1$';
label2 = '$\hat{\psi}_2$';
legend(label1,label2,'Interpreter','latex')
```



另请参阅

[text](#) | [plot](#) | [title](#) | [xlabel](#) | [ylabel](#)

详细信息

- “为图添加标题和轴标签” (第 8-2 页)
- “向图中添加文本” (第 8-14 页)

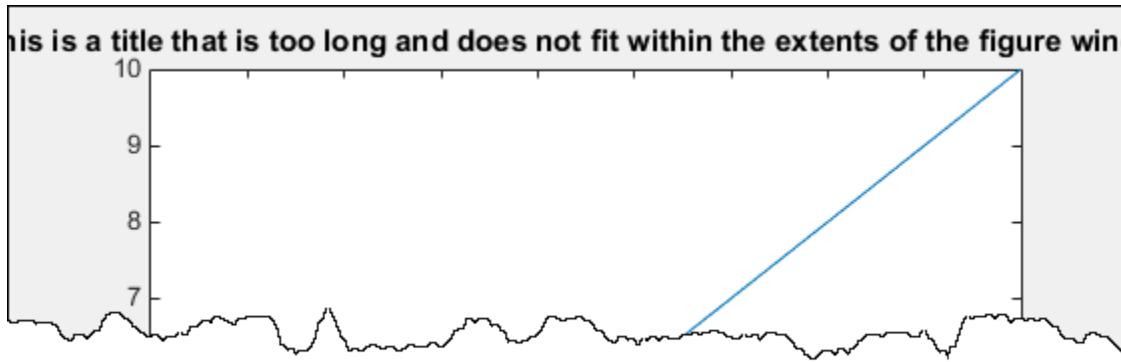
外部网站

- <https://www.latex-project.org/>

缩小图形标题

MATLAB 图形标题使用加粗的微大字体，以便更醒目。因此，一些文本可能在图窗窗口中无法容纳。例如，此代码创建一个图形，其中的长标题无法容纳在图窗窗口范围内。

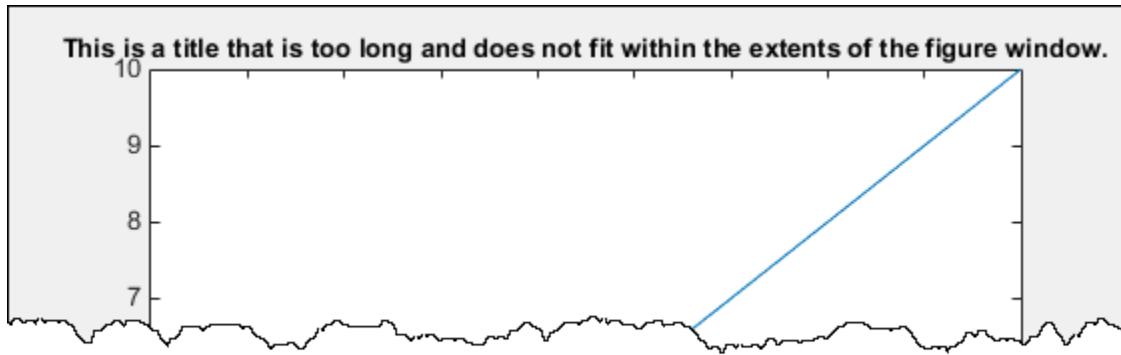
```
plot(1:10);
title(['This is a title that is too long and does not fit',...
    'within the extents of the figure window.'])
```



标题字体大小取决于坐标区的 `TitleFontSizeMultiplier` 和 `FontSize` 属性。默认情况下，`FontSize` 属性为 10 磅，而 `TitleFontSizeMultiplier` 为 1.100，这意味着标题字体大小为 11 磅。

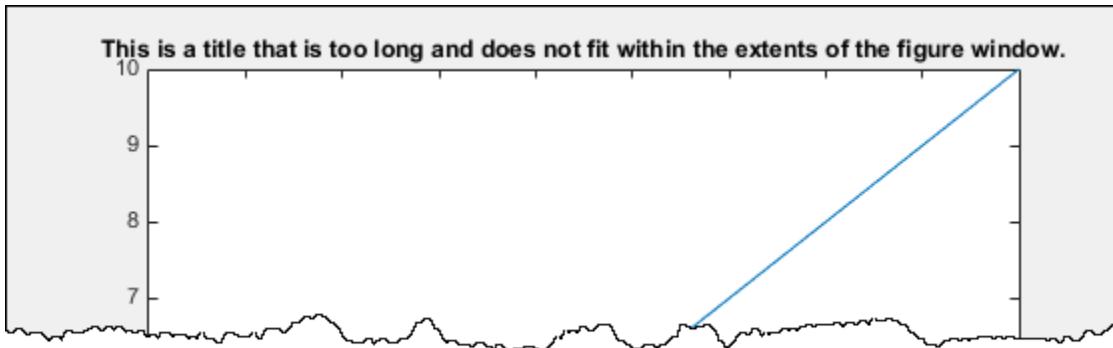
若要更改标题字体大小而不影响坐标区的其他部分字体，请设置坐标区的 `TitleFontSizeMultiplier` 属性。

```
plot(1:10);
title(['This is a title that is too long and does not fit',...
    'within the extents of the figure window.'])
ax = gca;
ax.TitleFontSizeMultiplier = 1;
```



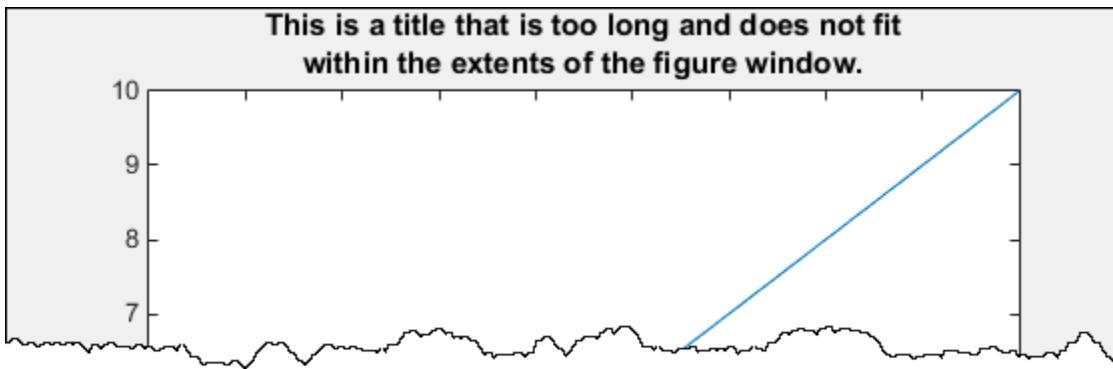
若要使整个坐标区的字体变小，请设置 `FontSize` 属性。更改此属性会影响标题、刻度标签和坐标轴标签（如果存在）的字体。

```
plot(1:10);
title(['This is a title that is too long and does not fit',...
    'within the extents of the figure window.'])
ax = gca;
ax.FontSize = 8;
```



若要保持字体大小相同并跨行显示标题，请使用带有花括号 {} 的元胞数组定义多行标题。

```
plot(1:10);
title({'This is a title that is too long and does not fit',...
    'within the extents of the figure window.'})
```



另请参阅

函数
title

属性
Axes

坐标区外观

- “指定坐标轴范围” (第 9-2 页)
- “指定坐标轴刻度值和标签” (第 9-9 页)
- “添加网格线和编辑布局” (第 9-16 页)
- “合并多个绘图” (第 9-24 页)
- “创建包含双 y 轴的图。” (第 9-32 页)
- “修改包含两个 y 轴的图形的属性” (第 9-40 页)
- “使用多个刻度和坐标轴范围显示数据” (第 9-46 页)
- “控制坐标轴长度比率和数据单位长度” (第 9-56 页)
- “控制坐标区布局” (第 9-63 页)
- “控制坐标区纵横比” (第 9-67 页)
- “控制绘图函数如何选择颜色和线型” (第 9-78 页)
- “在绘图和图表中裁剪” (第 9-83 页)
- “使用图形平滑处理” (第 9-88 页)

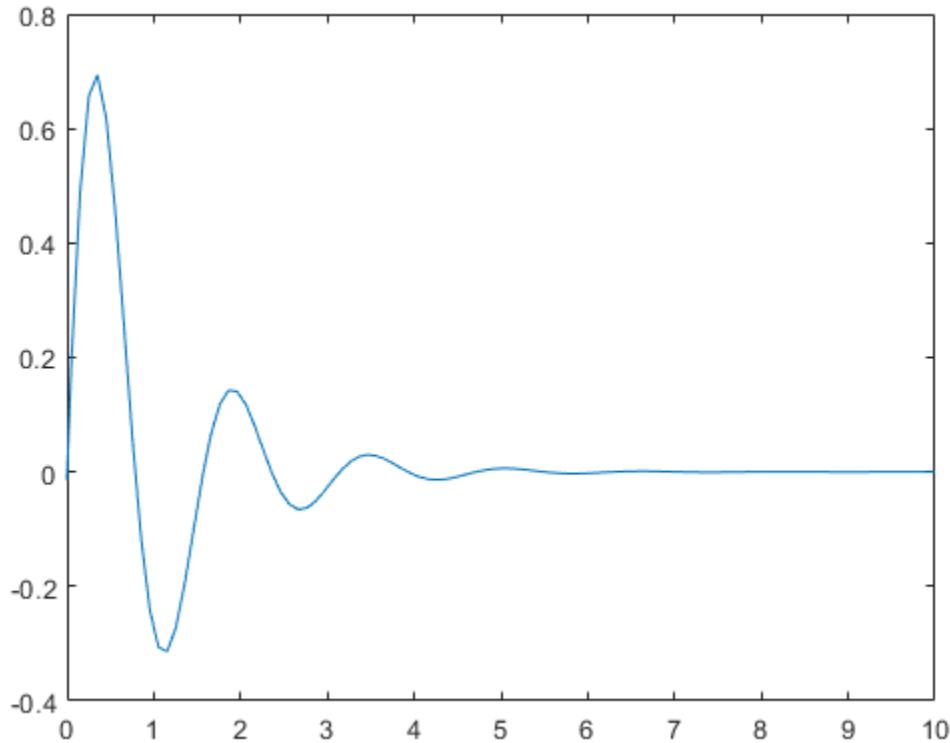
指定坐标轴范围

您可以设置 x 轴、y 轴和 z 坐标轴范围，以控制数据在坐标区上的显示位置。您也可以更改 x 轴线和 y 轴线的显示位置（仅适用于二维绘图），或反转值沿每条轴递增的方向。

更改坐标轴范围

创建一个线图。使用 `xlim` 和 `ylim` 函数指定坐标轴范围。对于三维绘图，请使用 `zlim` 函数。将 `[min max]` 形式的二元素向量传递给函数。

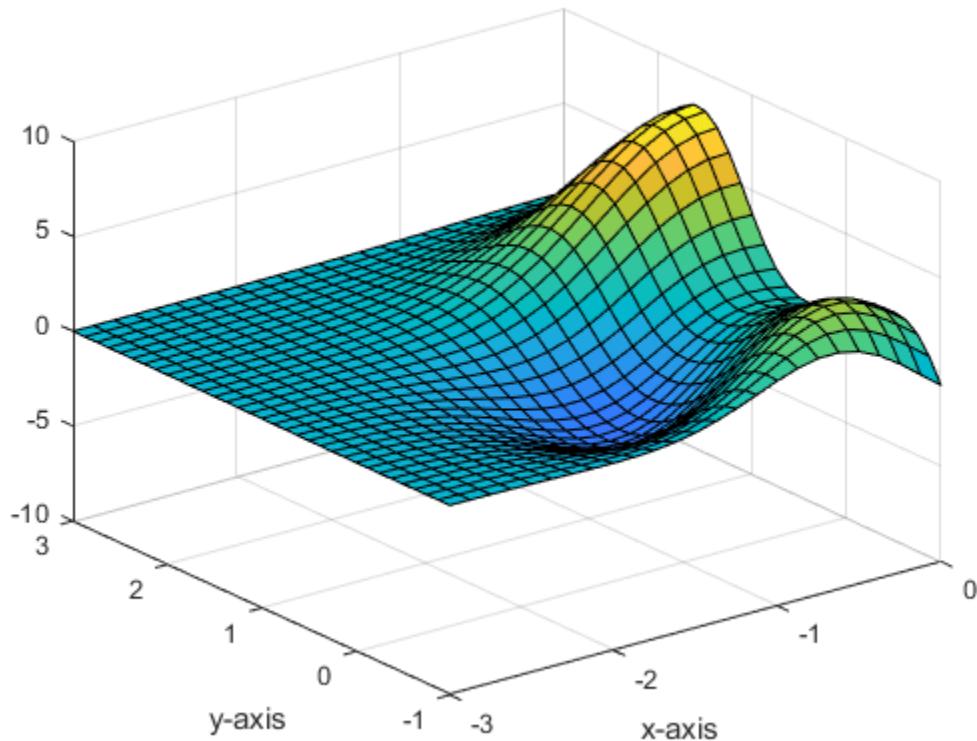
```
x = linspace(-10,10,200);
y = sin(4*x)./exp(x);
plot(x,y)
xlim([0 10])
ylim([-0.4 0.8])
```



使用半自动坐标轴范围

将 x 轴范围最大值设为 0，y 轴范围最小值设为 -1。其他范围则由 MATLAB 选择。对于自动计算的最小值或最大值范围，分别使用 `-inf` 或 `inf` 来表示。

```
[X,Y,Z] = peaks;
surf(X,Y,Z)
xlabel('x-axis')
ylabel('y-axis')
xlim([-inf 0])
ylim([-1 inf])
```

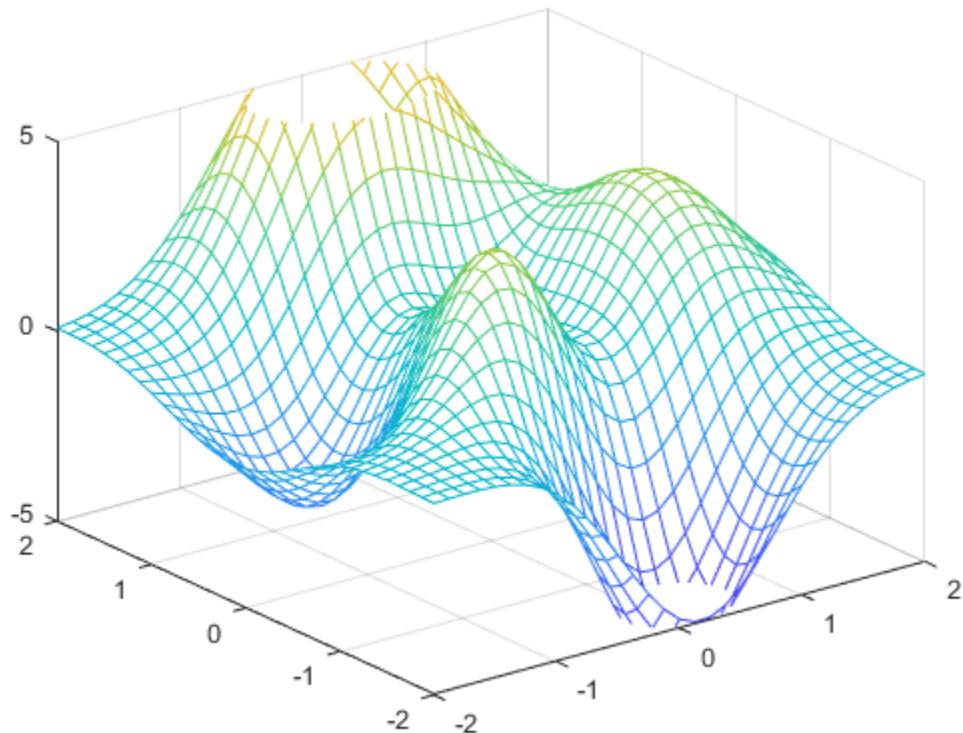


还原为默认范围

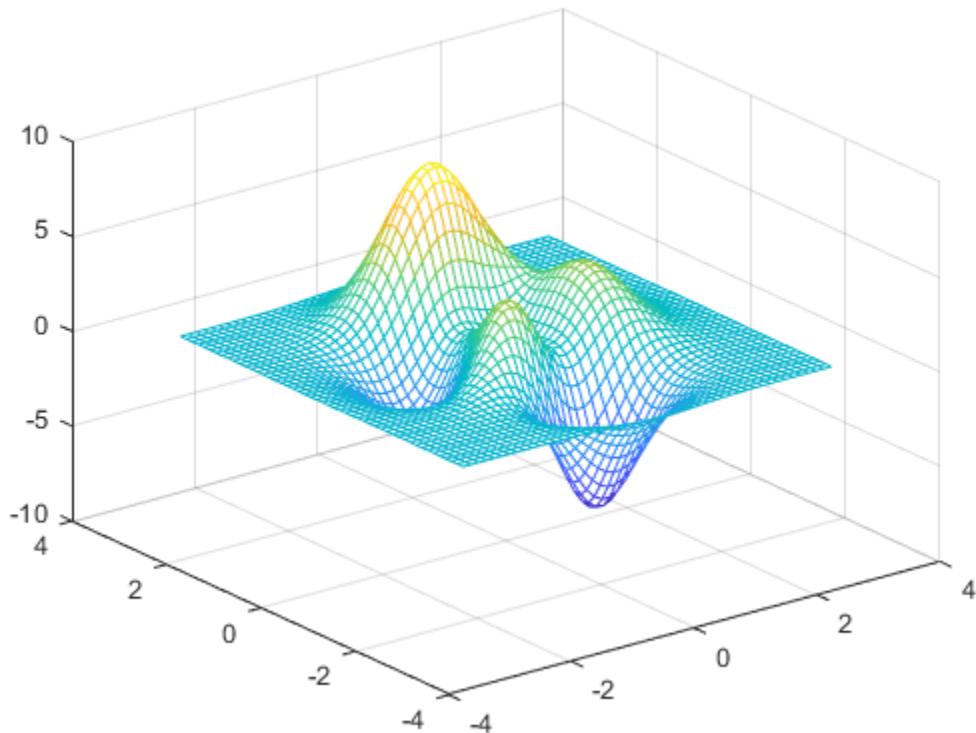
创建一个网格图并更改坐标轴范围，然后还原为默认范围。

```
[X,Y,Z] = peaks;
mesh(X,Y,Z)
xlim([-2 2])
ylim([-2 2])
zlim([-5 5])
```

9 坐标区外观



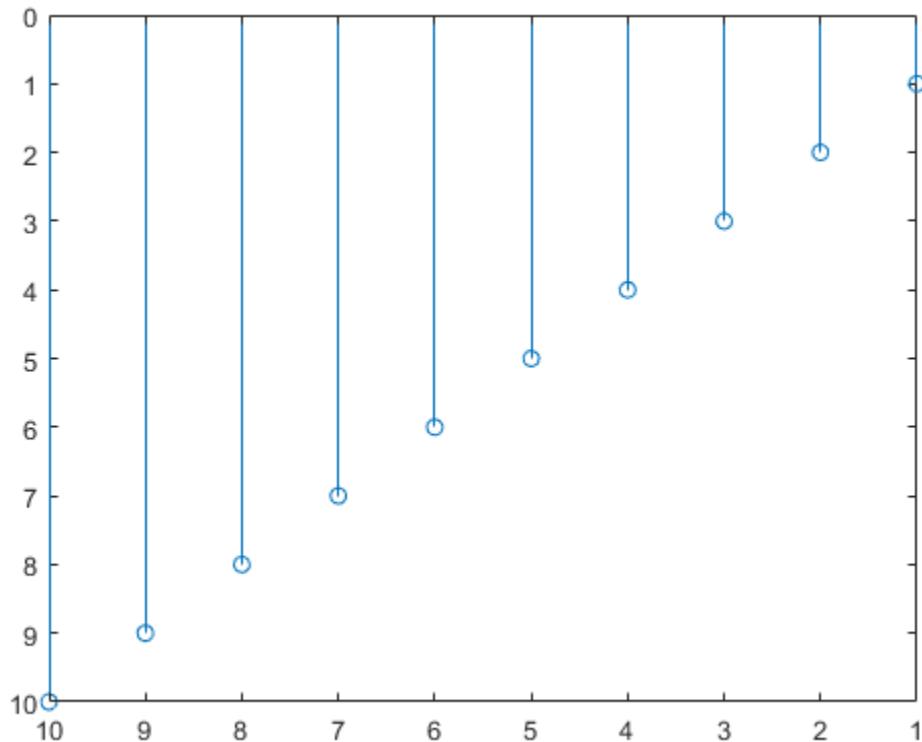
```
xlim auto  
ylim auto  
zlim auto
```



反转坐标轴方向

通过设置 Axes 对象的 XDir 和 YDir 属性，可控制 x 轴和 y 轴值递增的方向。这些属性可以设置为 'reverse' 或 'normal'（默认值）。使用 gca 命令可访问 Axes 对象。

```
stem(1:10)
ax = gca;
ax.XDir = 'reverse';
ax.YDir = 'reverse';
```

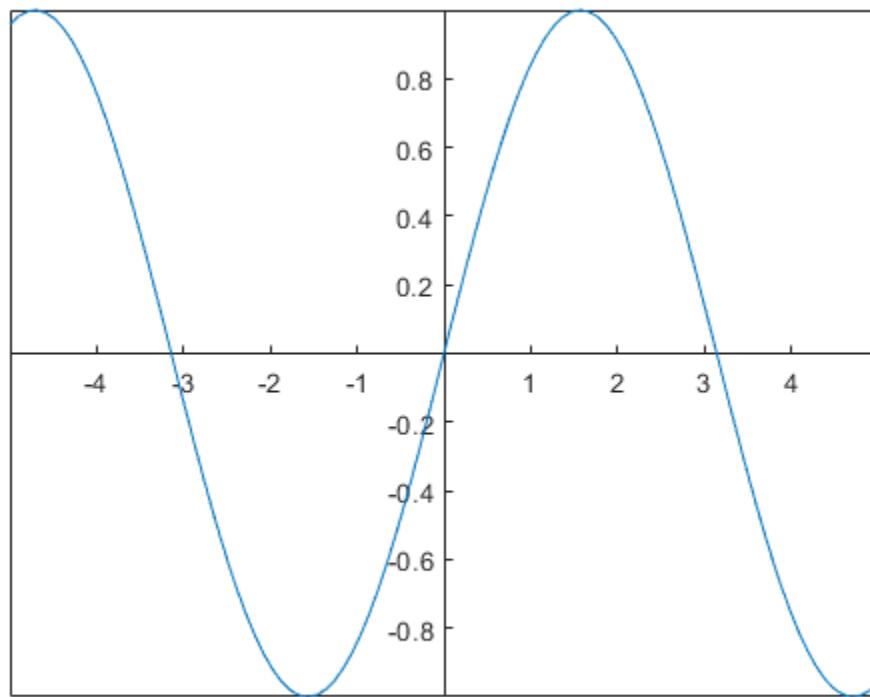


显示通过原点的轴线

默认情况下，x 轴和 y 轴沿坐标区的外边界显示。通过设置 Axes 对象的 XAxisLocation 和 YAxisLocation 属性来更改轴线位置，以使轴线在原点 (0,0) 处交叉。将 XAxisLocation 设置为 'top'、'bottom' 或 'origin'。将 YAxisLocation 设置为 'left'、'right' 或 'origin'。这些属性仅适用于二维视图中的坐标区。

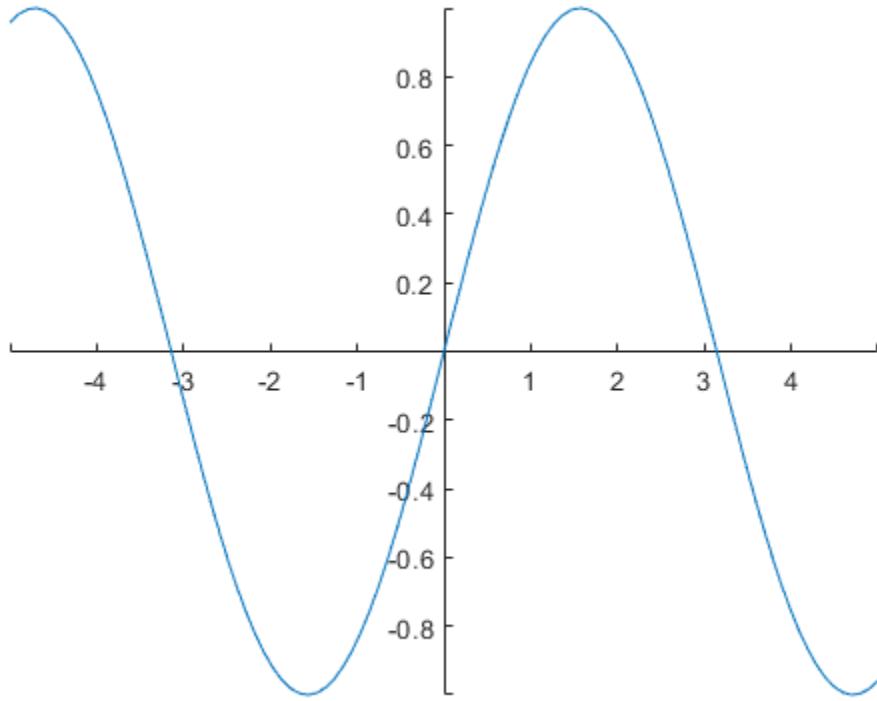
```
x = linspace(-5,5);
y = sin(x);
plot(x,y)

ax = gca;
ax.XAxisLocation = 'origin';
ax.YAxisLocation = 'origin';
```



删除坐标区框轮廓。

box off



另请参阅

函数

`axis | xlim | ylim | zlim | xticks | yticks | zticks | grid`

属性

`Axes`

相关示例

- “指定坐标轴刻度值和标签” (第 9-9 页)
- “添加网格线和编辑布局” (第 9-16 页)
- “为图添加标题和轴标签” (第 8-2 页)

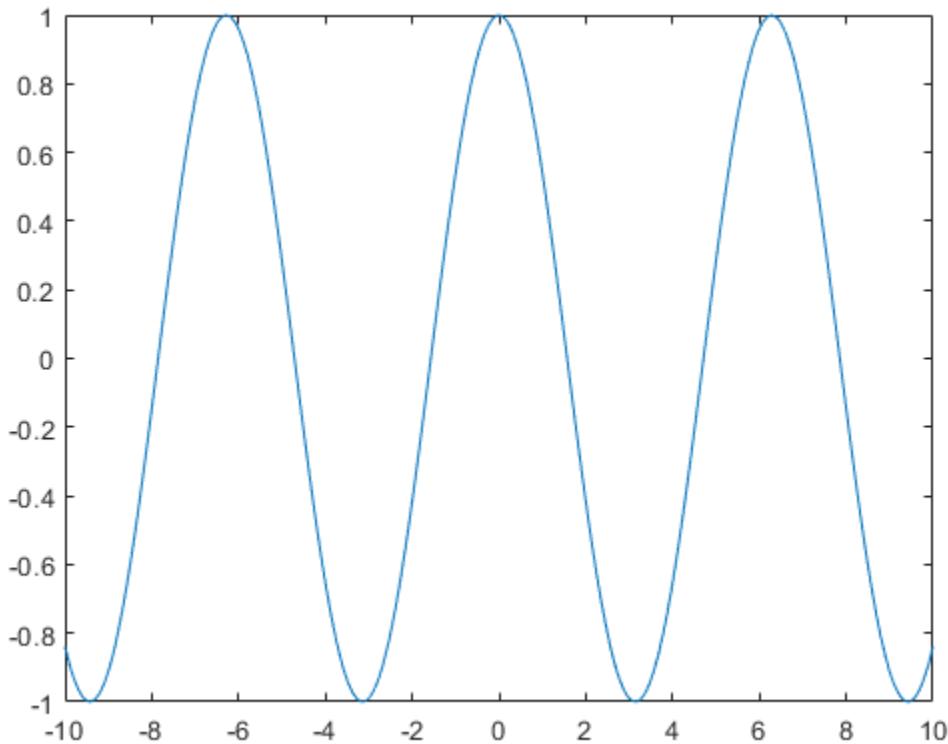
指定坐标轴刻度值和标签

自定义沿坐标轴的刻度值和标签有助于突出显示数据的特定方面。以下示例说明一些常见的自定义，例如修改刻度值的放置位置、更改刻度标签的文本和格式，以及旋转刻度标签。

更改刻度值位置和标签

创建 x ，将其指定为 200 个介于 -10 和 10 之间的线性间隔值。创建 x 的余弦函数 y 。绘制数据图。

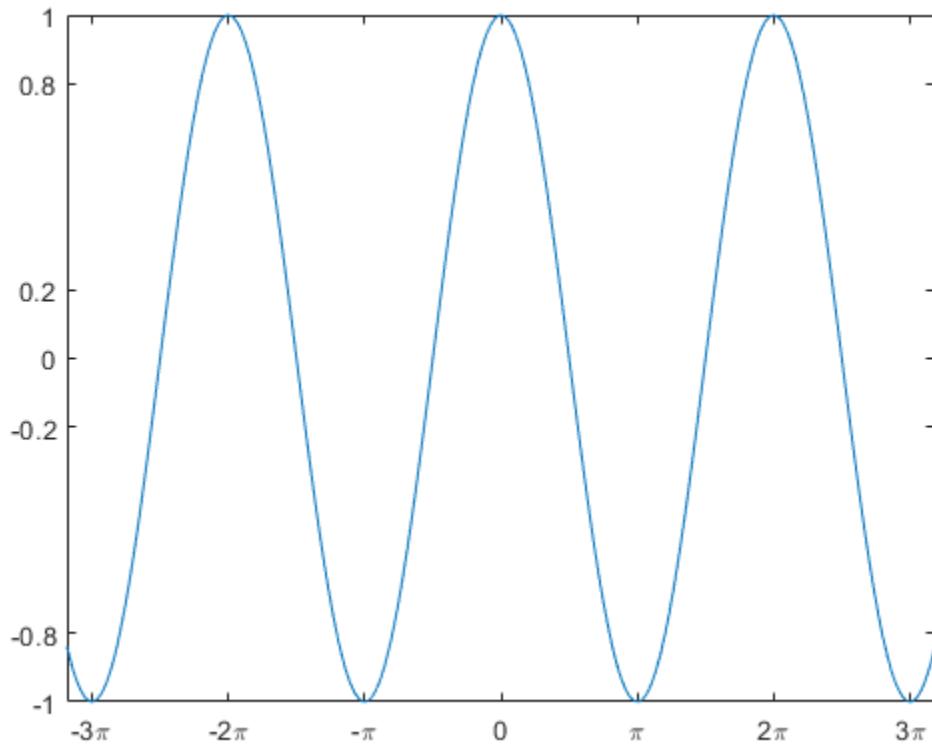
```
x = linspace(-10,10,200);
y = cos(x);
plot(x,y)
```



更改沿 x 轴和 y 轴的刻度值位置。将这些位置指定为一个由递增值组成的向量。这些值无需等距。

此外，还要更改沿 x 轴的每个刻度值关联的标签。并用一个字符向量元胞数组来指定刻度标签。要在标签中包含特殊字符或希腊字母，可使用 TeX 标记，例如用 $\backslash\pi$ 表示 π 符号。

```
xticks([-3*pi -2*pi -pi 0 pi 2*pi 3*pi])
xticklabels({'-3\pi','-2\pi','-\pi','0','\pi','2\pi','3\pi'})
yticks([-1 -0.8 -0.2 0 0.2 0.8 1])
```

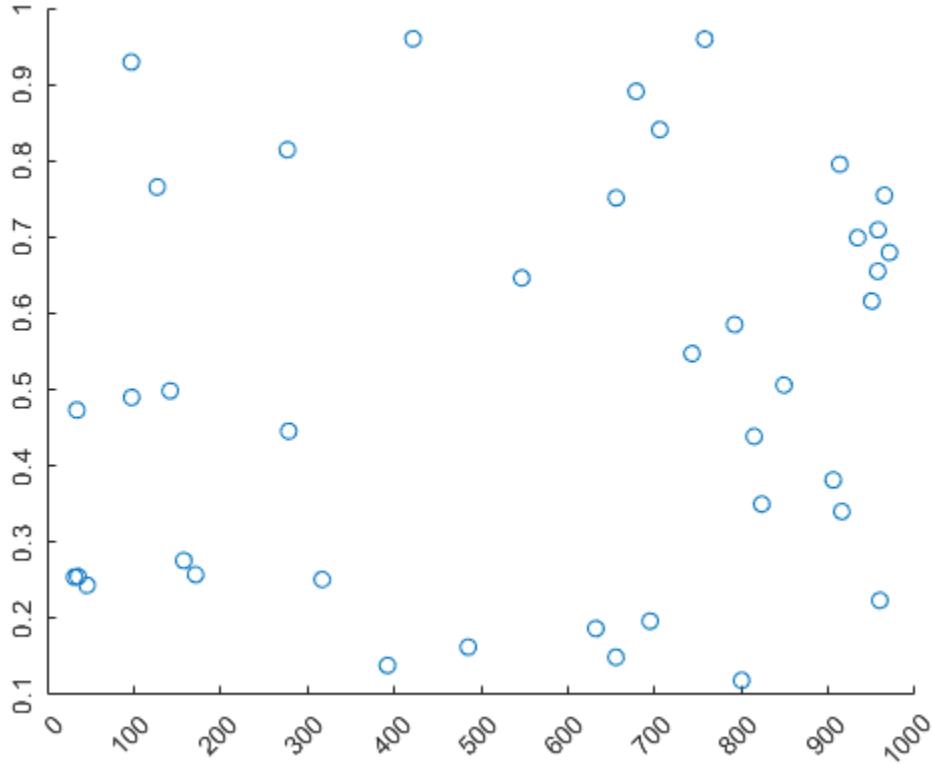


对于 R2016b 之前的版本，应使用 Axes 对象的 XTick、XTickLabel、YTick 和 YTickLabel 属性设置刻度值和标签。例如，将 Axes 对象赋予一个变量（如 `ax = gca`）。然后使用圆点表示法设置 XTick 属性，例如 `ax.XTick = [-3*pi -2*pi -pi 0 pi 2*pi 3*pi]`。对于 R2014b 之前的版本，应使用 `set` 函数设置此属性。

旋转刻度标签

创建散点图并沿每条轴旋转刻度标签。将此旋转指定为一个标量值。正值表示逆时针旋转。负值表示顺时针旋转。

```
x = 1000*rand(40,1);
y = rand(40,1);
scatter(x,y)
xtickangle(45)
ytickangle(90)
```

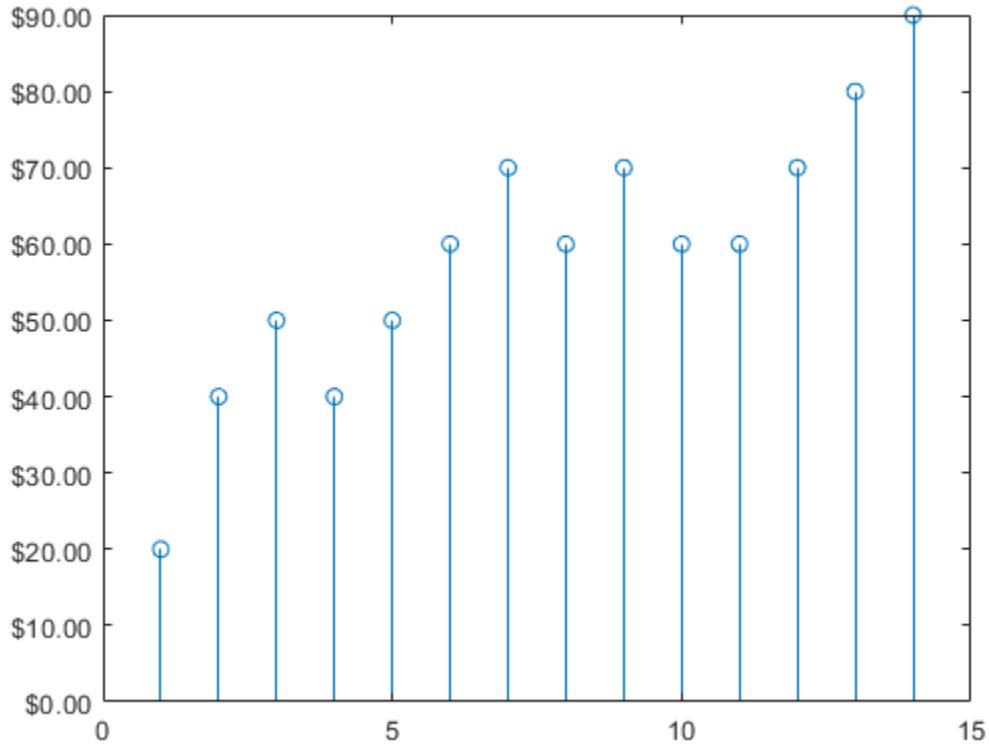


对于 R2016b 之前的版本，使用 `Axes` 对象的 `XTickLabelRotation` 和 `YTickLabelRotation` 属性指定旋转。例如，将 `Axes` 对象赋予一个变量（如 `ax = gca`）。然后使用圆点表示法设置 `XTickLabelRotation` 属性，例如 `ax.XTickLabelRotation = 45`。

更改刻度标签格式

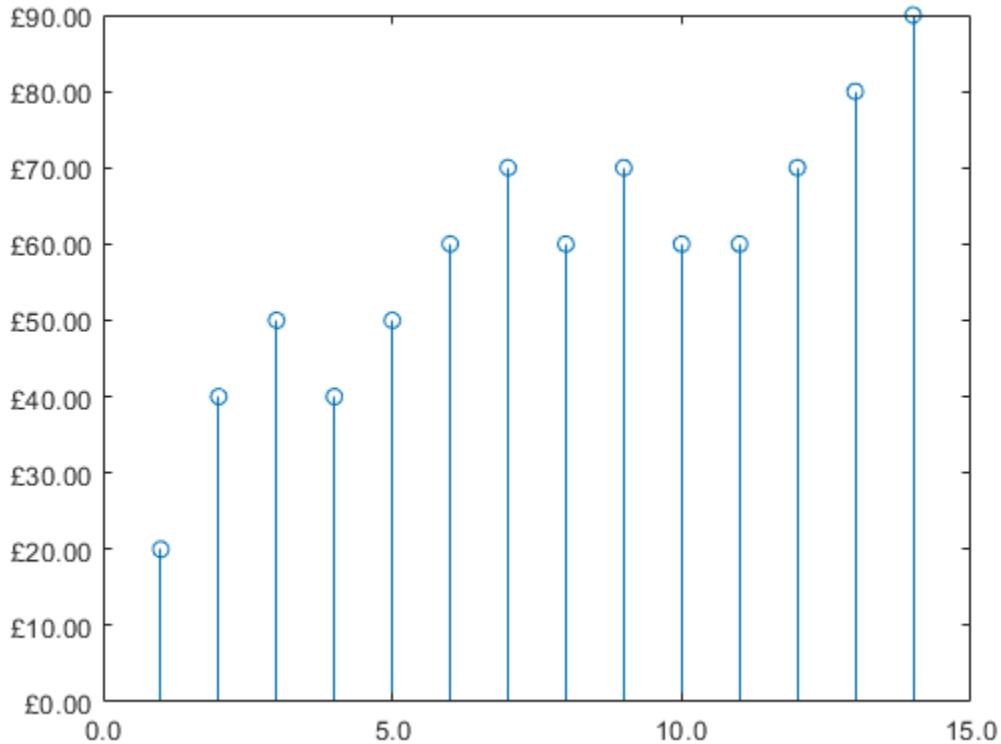
创建针状图并将沿 y 轴的刻度标签值显示为美元值。

```
profit = [20 40 50 40 50 60 70 60 70 60 60 70 80 90];
stem(profit)
xlim([0 15])
ytickformat('usd')
```



若要进一步控制格式，请指定一种自定义格式。例如，使用 `'%.1f'` 在 x 轴刻度标签中显示一个十进制值。使用 `'\xA3%.2f'` 将 y 轴刻度标签显示为英镑。选项 `\xA3` 表示英镑符号的 Unicode 字符。有关指定自定义格式的详细信息，请参阅 `xtickformat` 函数。

```
xtickformat('%.1f')
ytickformat('\xA3%.2f')
```



用于分别控制各个坐标轴的标尺对象

MATLAB 为每个坐标轴创建一个标尺对象。与所有图形对象一样，标尺对象也具有您可以查看和修改的属性。标尺对象允许您进一步分别控制 x 轴、y 轴或 z 轴的格式设置。可以通过 `Axes` 对象的 `XAxis`、`YAxis` 或 `ZAxis` 属性访问与特定坐标轴关联的标尺对象。标尺的类型取决于坐标轴上的数据类型。对于数值数据，MATLAB 创建 `NumericRuler` 对象。

```
ax = gca;
ax.XAxis

ans =
  NumericRuler with properties:

    Limits: [0 15]
    Scale: 'linear'
    Exponent: 0
    TickValues: [0 5 10 15]
    TickLabelFormat: '%.1f'

Show all properties
```

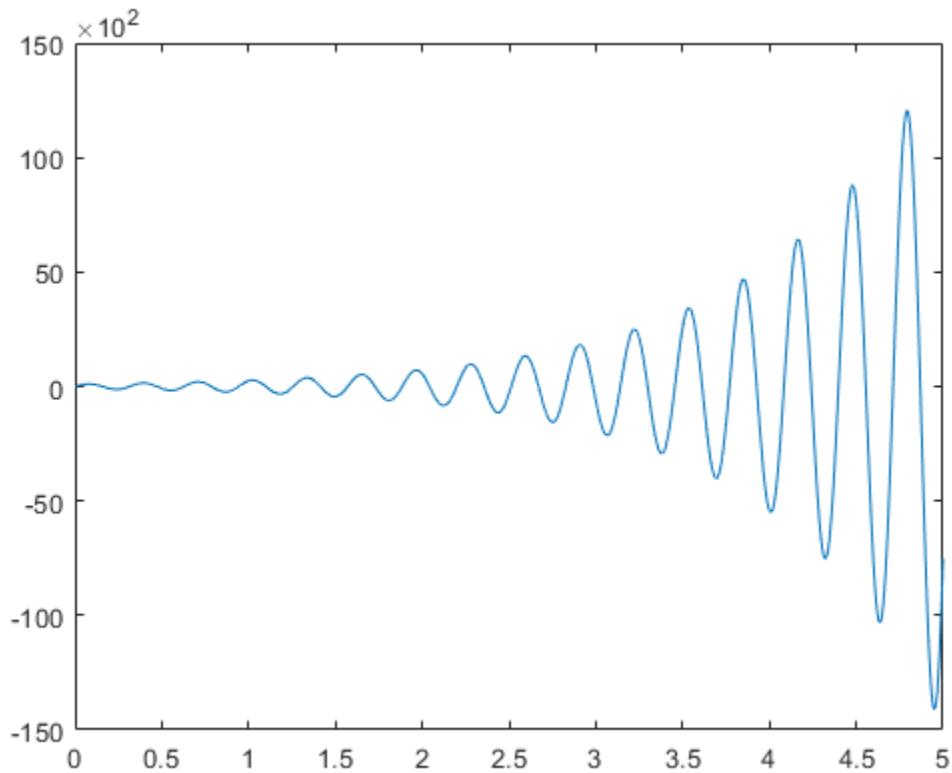
使用标尺对象控制指数标签中的值

使用介于 -15,000 和 15,000 之间的 y 值绘制数据图。默认情况下，y 轴刻度标签使用指数记数法（指数值为 4，底数为 10）。将指数值更改为 2。设置与 y 轴关联的标尺对象的 `Exponent` 属性。通过 `Axes` 对象的 `YAxis` 属性访问标尺对象。指数标签和刻度标签会相应地进行更改。

9 坐标区外观

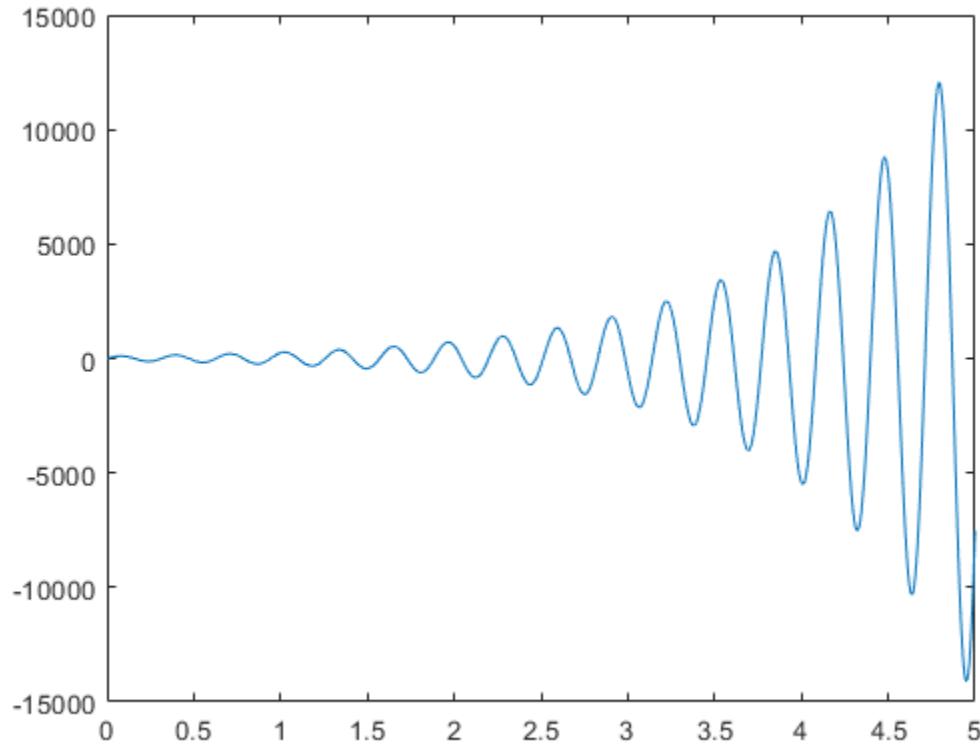
```
x = linspace(0,5,1000);
y = 100*exp(x).*sin(20*x);
plot(x,y)
```

```
ax = gca;
ax.YAxis.Exponent = 2;
```



将指数值更改为 0，使刻度标签不使用指数记数法。

```
ax.YAxis.Exponent = 0;
```



另请参阅

函数

`xlim` | `xticks` | `yticks` | `zticks` | `xtickformat` | `xtickangle`

属性

`Axes` | `NumericRuler`

相关示例

- “添加网格线和编辑布局” (第 9-16 页)
- “指定坐标轴范围” (第 9-2 页)
- “为图添加标题和轴标签” (第 8-2 页)

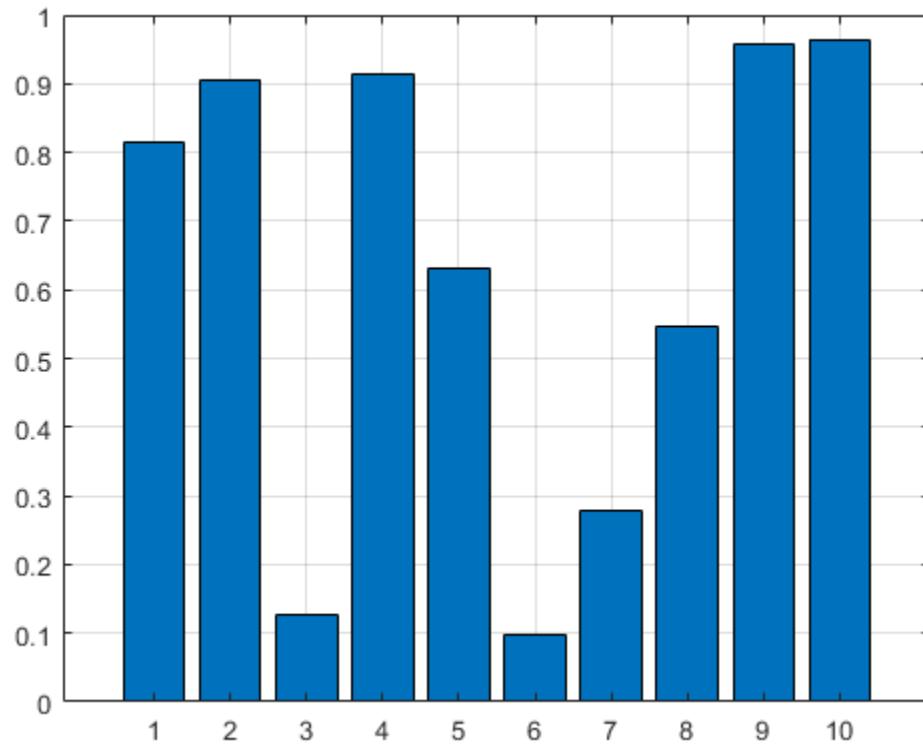
添加网格线和编辑布局

此示例说明如何在图形中添加网格线。它还说明了如何编辑网格线布局和修改网格线外观。

显示网格线

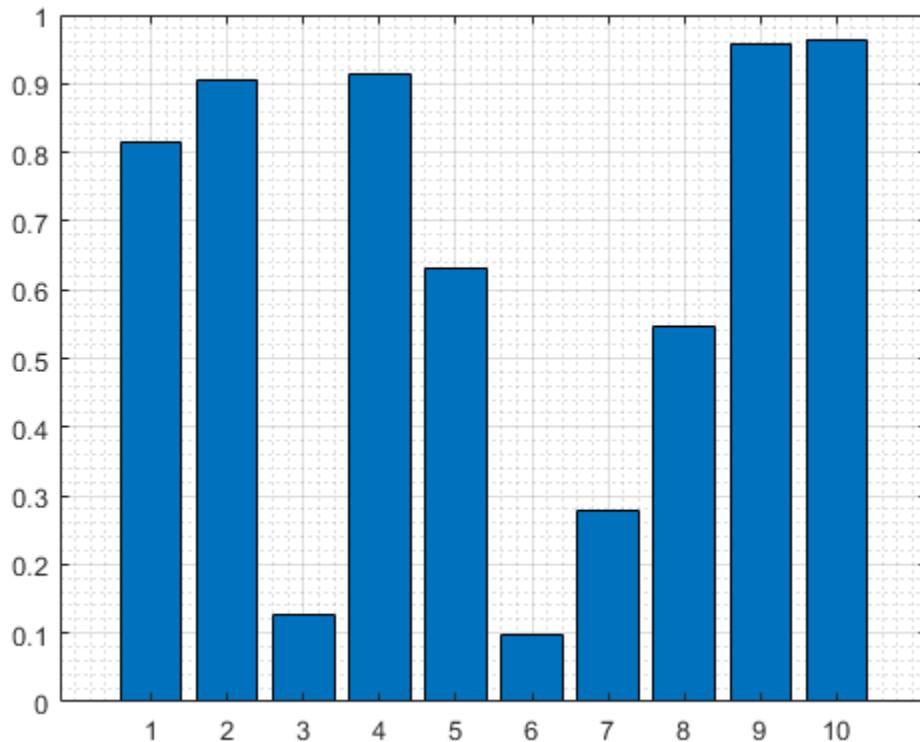
创建条形图并显示网格线。网格线显示在刻度线处。

```
y = rand(10,1);
bar(y)
grid on
```



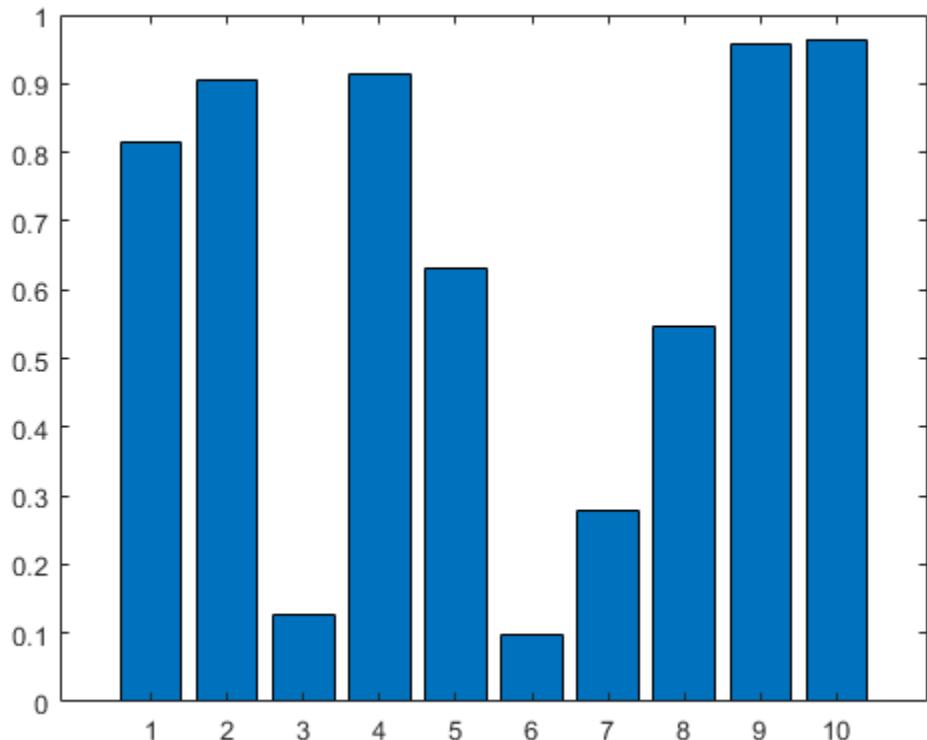
在刻度线之间添加次网格线。

```
grid minor
```



关闭所有网格线。

```
grid off
```

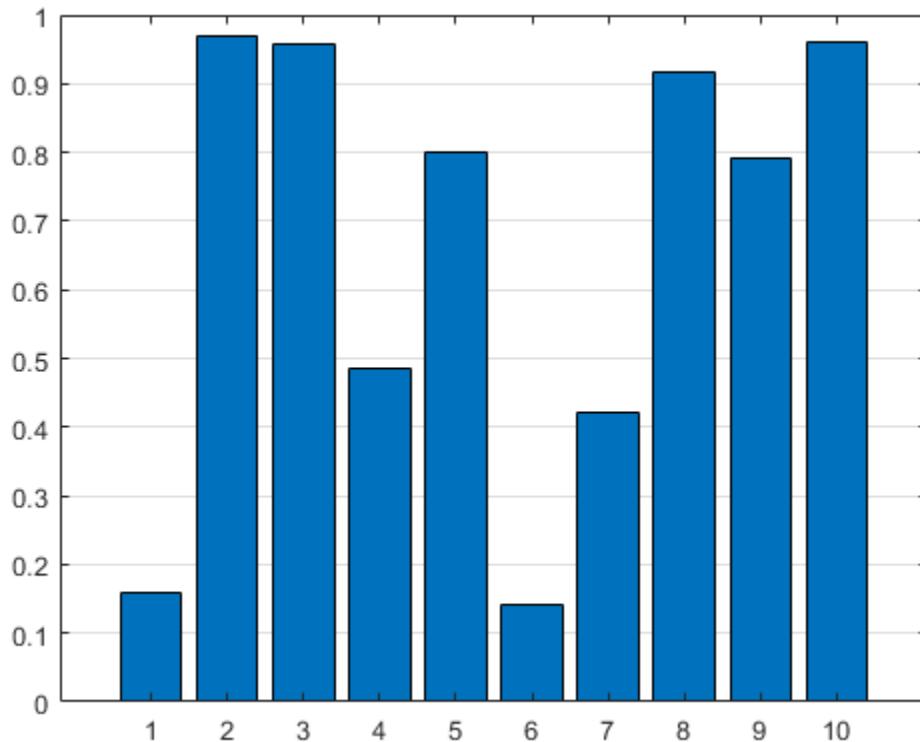


在特定方向显示网格线

通过访问 Axes 对象并设置 XGrid、YGrid 和 ZGrid 属性，可在特定方向显示网格线。这些属性可以设置为 'on' 或 'off'。

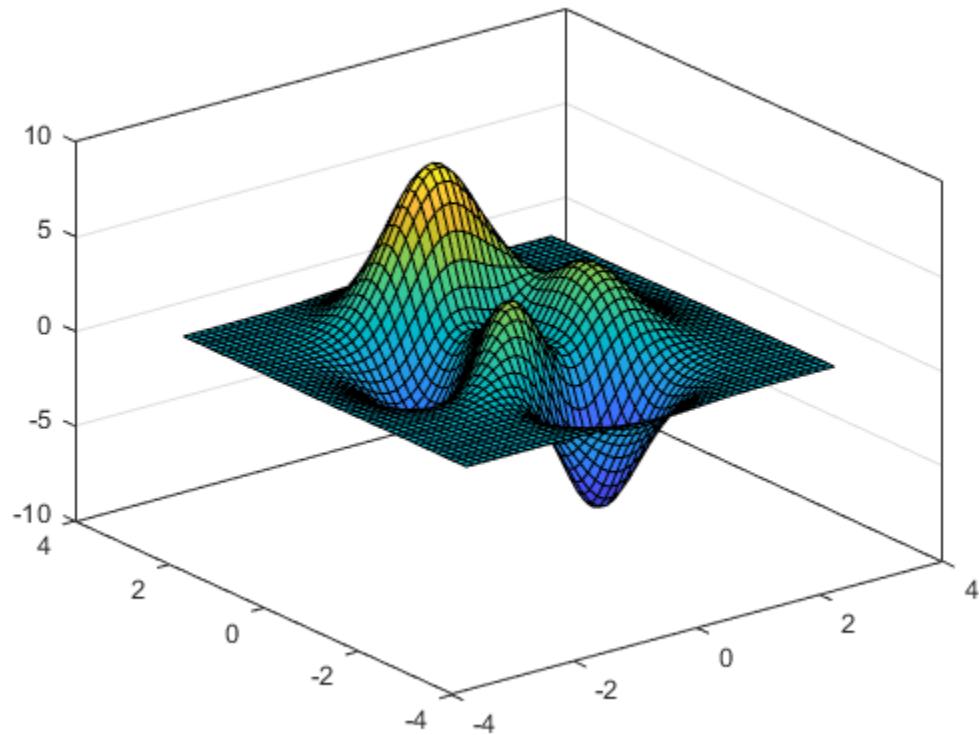
创建二维绘图且仅在 y 方向显示网格线。

```
y = rand(10,1);
bar(y)
ax = gca;
ax.XGrid = 'off';
ax.YGrid = 'on';
```



创建三维绘图且仅在 z 方向显示网格线。使用 **box on** 命令可显示坐标区框轮廓。

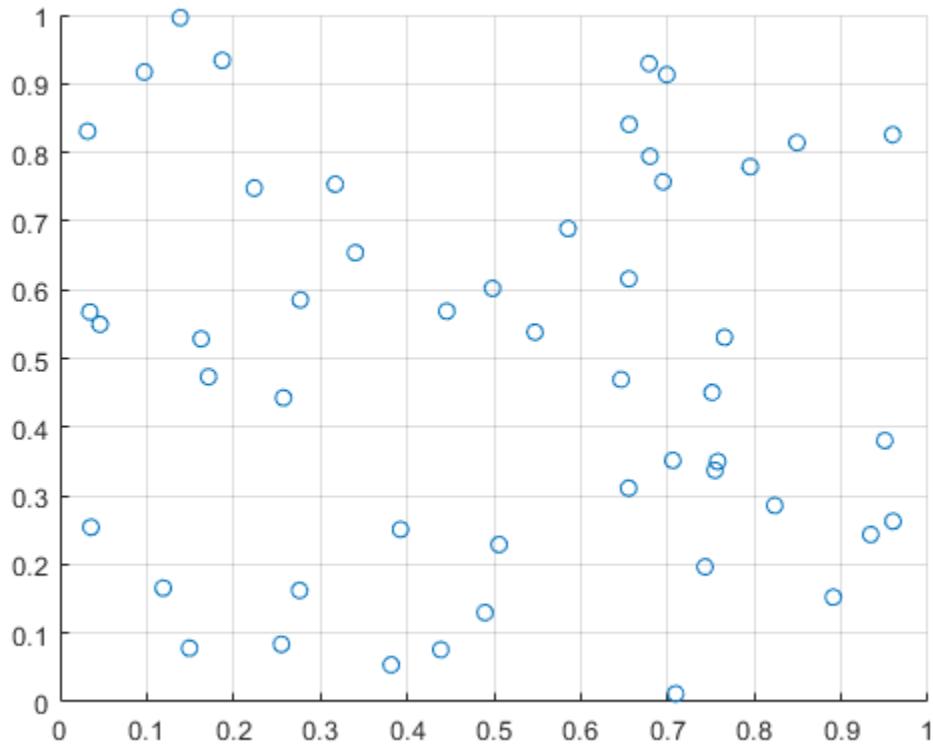
```
[X,Y,Z] = peaks;
surf(X,Y,Z)
box on
ax = gca;
ax.ZGrid = 'on';
ax.XGrid = 'off';
ax.YGrid = 'off';
```



编辑网格线布局

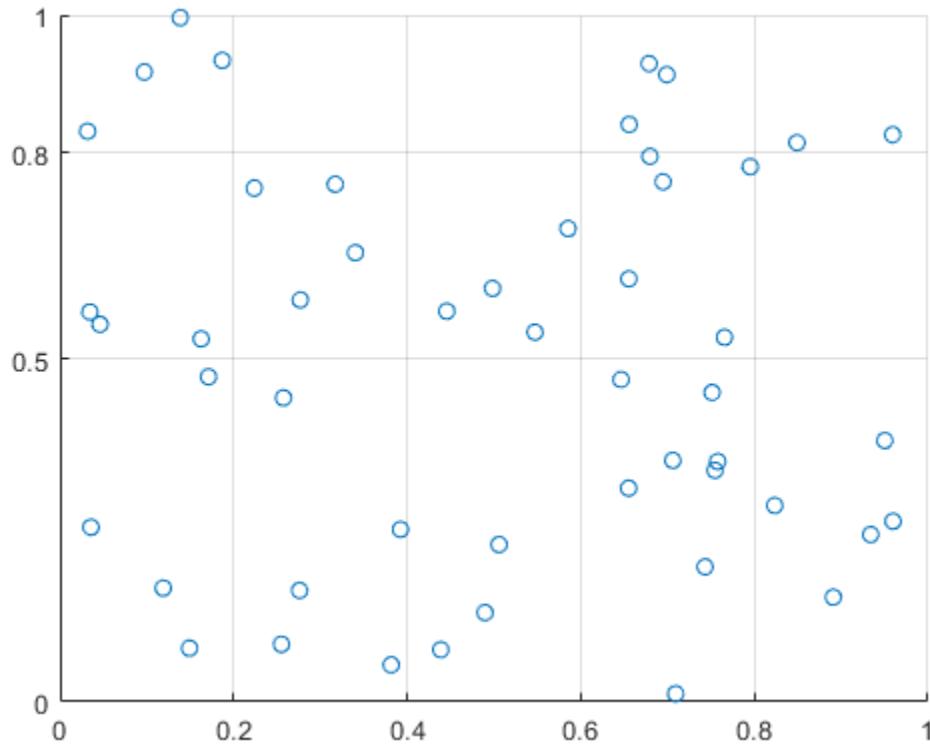
创建一个由随机数据组成的散点图并显示网格线。

```
x = rand(50,1);
y = rand(50,1);
scatter(x,y)
grid on
```



网格线显示在刻度线位置。通过更改刻度线位置可编辑网格线的布局。

```
xticks(0:0.2:1)  
yticks([0 0.5 0.8 1])
```

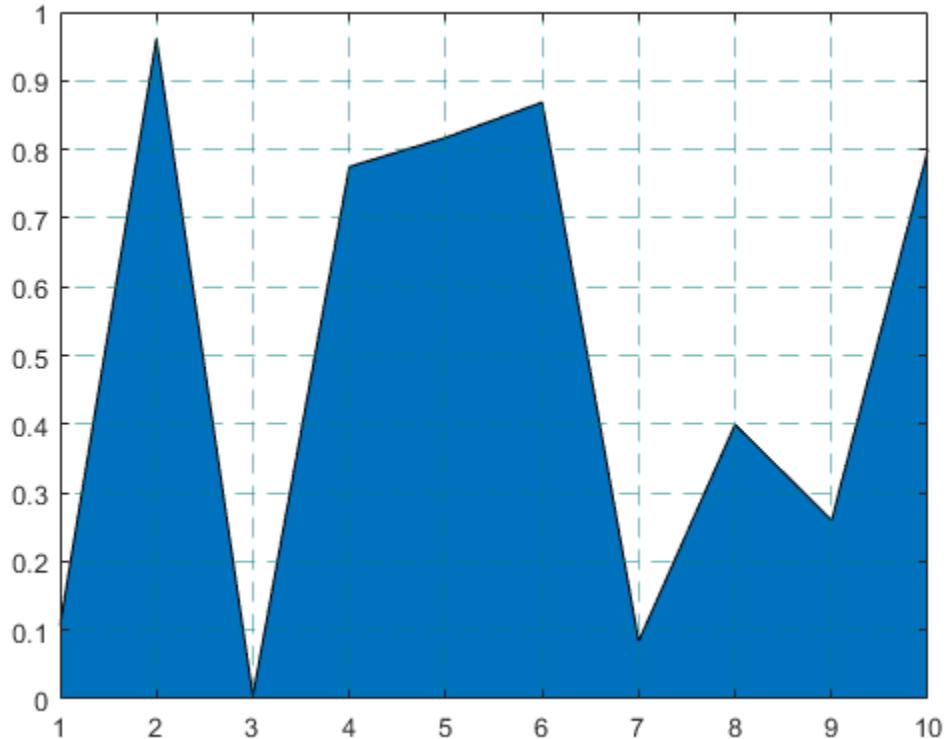


修改网格线的可视外观

更改区域图网格线的颜色、线型和透明度。通过访问 Axes 对象修改网格线的外观。然后设置与网格相关的属性，例如 GridColor、GridLineStyle 和 GridAlpha 属性。通过设置 Layer 属性可在绘图上显示网格线。

```
y = rand(10,1);
area(y)
grid on

ax = gca;
ax.GridColor = [0 .5 .5];
ax.GridLineStyle = '-';
ax.GridAlpha = 0.5;
ax.Layer = 'top';
```



另请参阅

函数

`grid` | `xticks` | `xlim` | `yticks` | `zticks`

属性

`Axes`

相关示例

- “指定坐标轴刻度值和标签”（第 9-9 页）
- “为图添加标题和轴标签”（第 8-2 页）
- “指定坐标轴范围”（第 9-2 页）

合并多个绘图

自 R2019b 开始提供。取代合并多个绘图 (R2019a)。

以下示例说明了如何使用 **hold** 函数合并同一坐标区中的绘图，以及如何使用 **tiledlayout** 函数在图窗中创建多个坐标区。

在同一坐标区中合并绘图

默认情况下，新图将清除现有图，并重置标题等坐标区属性。但是，您可以使用 **hold on** 命令在同一坐标区中合并多个图。例如，绘制两条直线和一个散点图，然后将 **hold** 状态重置为 **off**。

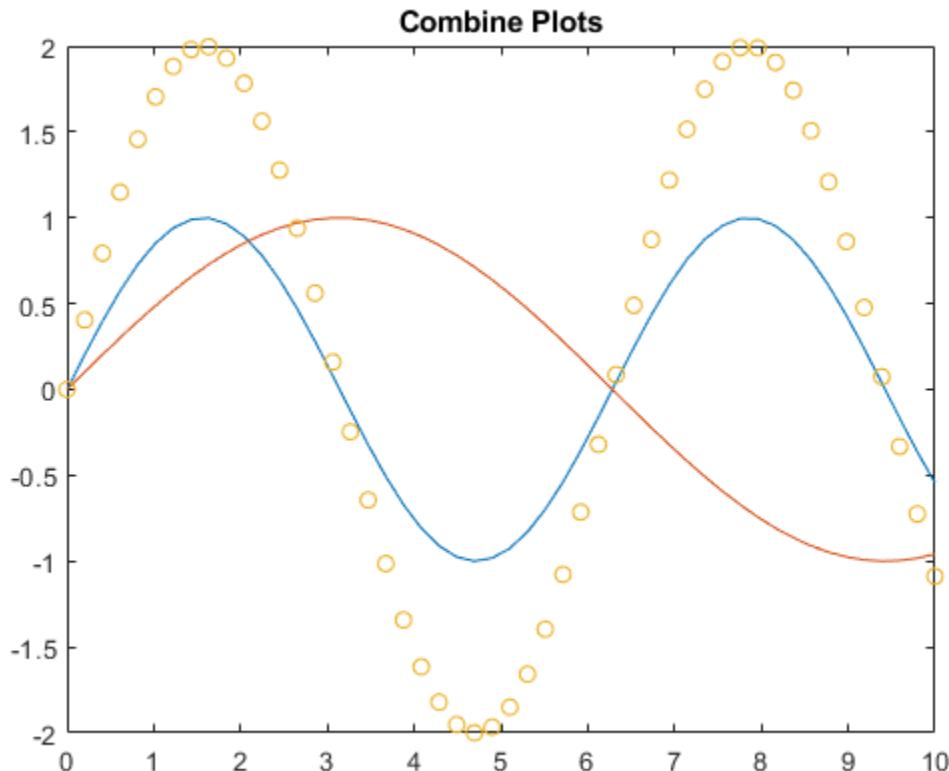
```
x = linspace(0,10,50);
y1 = sin(x);
plot(x,y1)
title('Combine Plots')

hold on

y2 = sin(x/2);
plot(x,y2)

y3 = 2*sin(x);
scatter(x,y3)

hold off
```



启用 `hold` 状态后，新图不会清除现有图，也不会重置标题或轴标签等坐标区属性。新图将根据坐标区的 `ColorOrder` 和 `LineStyleOrder` 属性循环使用颜色和线型。坐标区范围和刻度值可能会进行调整以适应新数据。

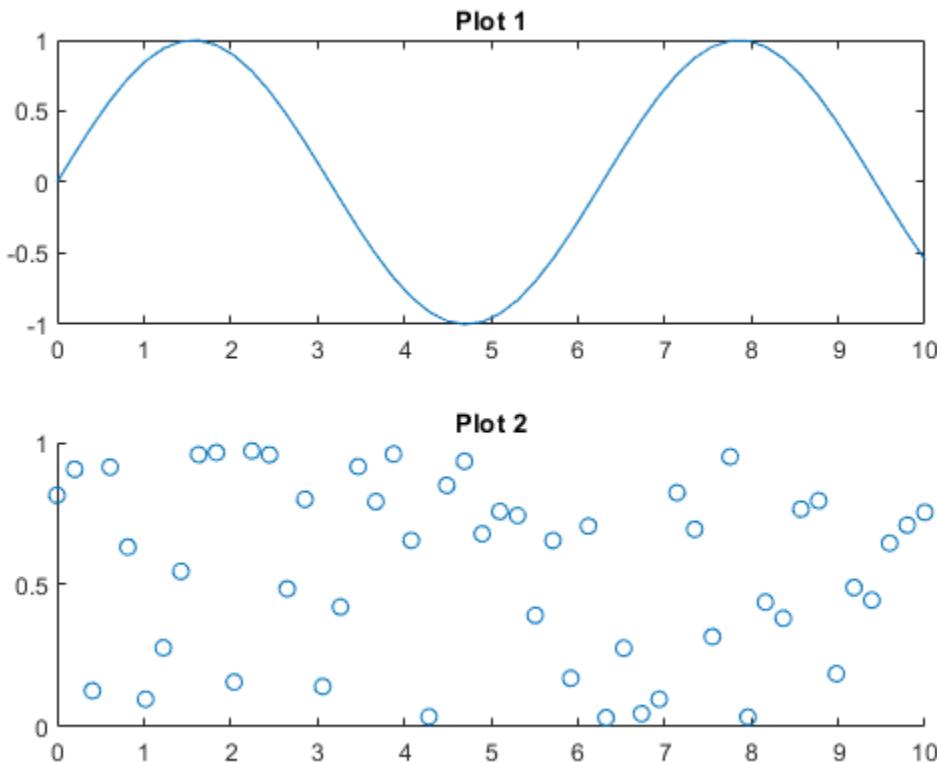
在图窗中显示多个坐标区

您可以使用 `tiledlayout` 函数在单个图窗中显示多个坐标区。此函数将创建一个分块图布局，该布局将图窗分为一系列不可见的图块网格。每个图块可以包含一个用于显示绘图的坐标区。创建布局后，调用 `nexttile` 函数将坐标区对象放置到布局中。然后调用绘图函数在该坐标区中绘图。例如，在一个 2×1 布局中创建两个绘图。为每个绘图添加标题。

```
x = linspace(0,10,50);
y1 = sin(x);
y2 = rand(50,1);
tiledlayout(2,1)

% Top plot
nexttile
plot(x,y1)
title('Plot 1')

% Bottom plot
nexttile
scatter(x,y2)
title('Plot 2')
```



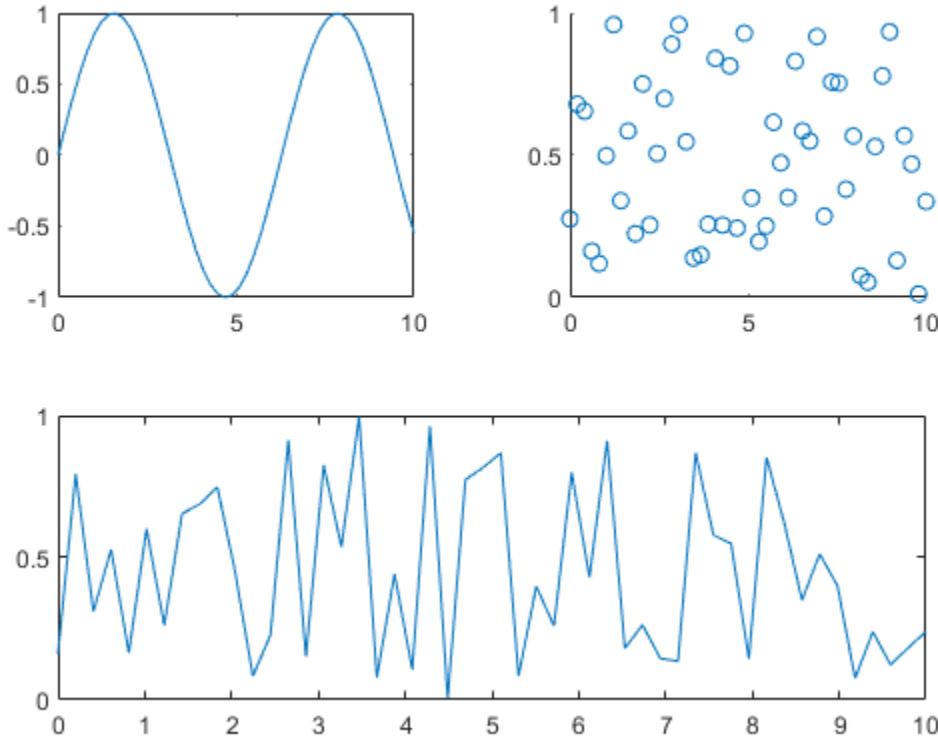
创建跨多行或多列的绘图

要创建跨多行或多列的绘图，请在调用 `nexttile` 时指定 `span` 参数。例如，创建一个 2×2 布局。绘制前两个图块。然后创建一个跨一行两列的图。

```
x = linspace(0,10,50);
y1 = sin(x);
y2 = rand(50,1);
```

```
% Top two plots
tiledlayout(2,2)
nexttile
plot(x,y1)
nexttile
scatter(x,y2)
```

```
% Plot that spans
nexttile([1 2])
y2 = rand(50,1);
plot(x,y2)
```



修改坐标区外观

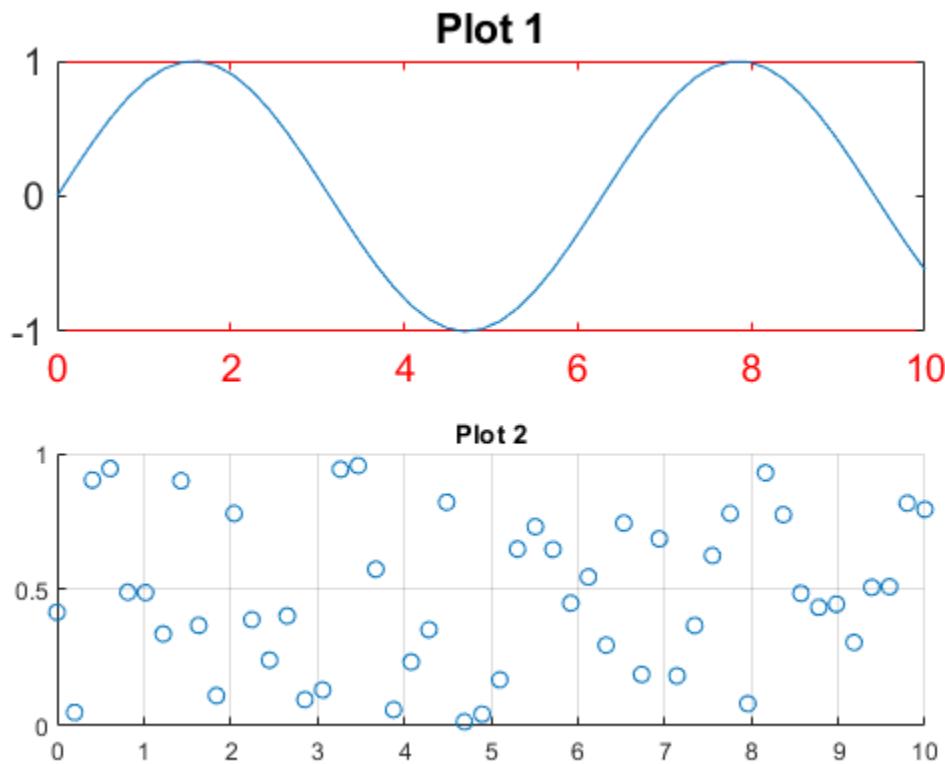
通过在每个坐标区对象上设置属性来修改坐标区外观。您可以通过带输出参数调用 `nexttile` 函数来获取坐标区对象。您也可以将坐标区对象指定为图形函数的第一个输入参数，以确保该函数作用于正确的坐标区。

例如，创建两个绘图，并将坐标区对象赋给变量 `ax1` 和 `ax2`。更改第一个绘图的坐标区字体大小和 x 轴颜色。为第二个绘图添加网格线。

```
x = linspace(0,10,50);
y1 = sin(x);
y2 = rand(50,1);
tiledlayout(2,1)

% Top plot
ax1 = nexttile;
plot(ax1,x,y1)
title(ax1,'Plot 1')
ax1.FontSize = 14;
ax1.XColor = 'red';

% Bottom plot
ax2 = nexttile;
scatter(ax2,x,y2)
title(ax2,'Plot 2')
grid(ax2,'on')
```

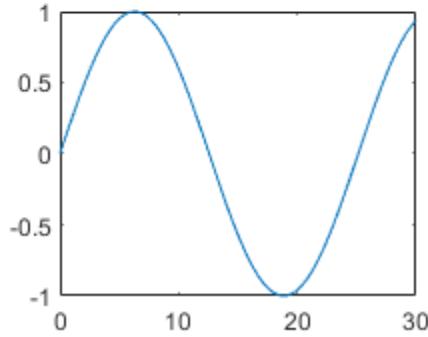
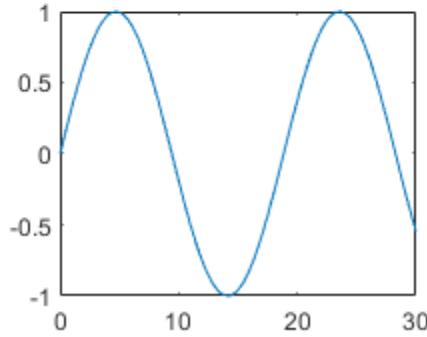
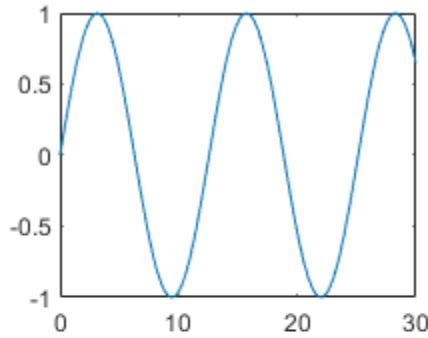
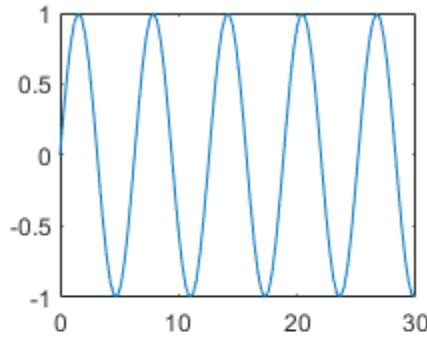


控制图块周围的间距

您可以通过指定 `Padding` 和 `TileSpacing` 属性来控制布局中图块周围的间距。例如，在一个 2×2 布局中显示四个绘图。

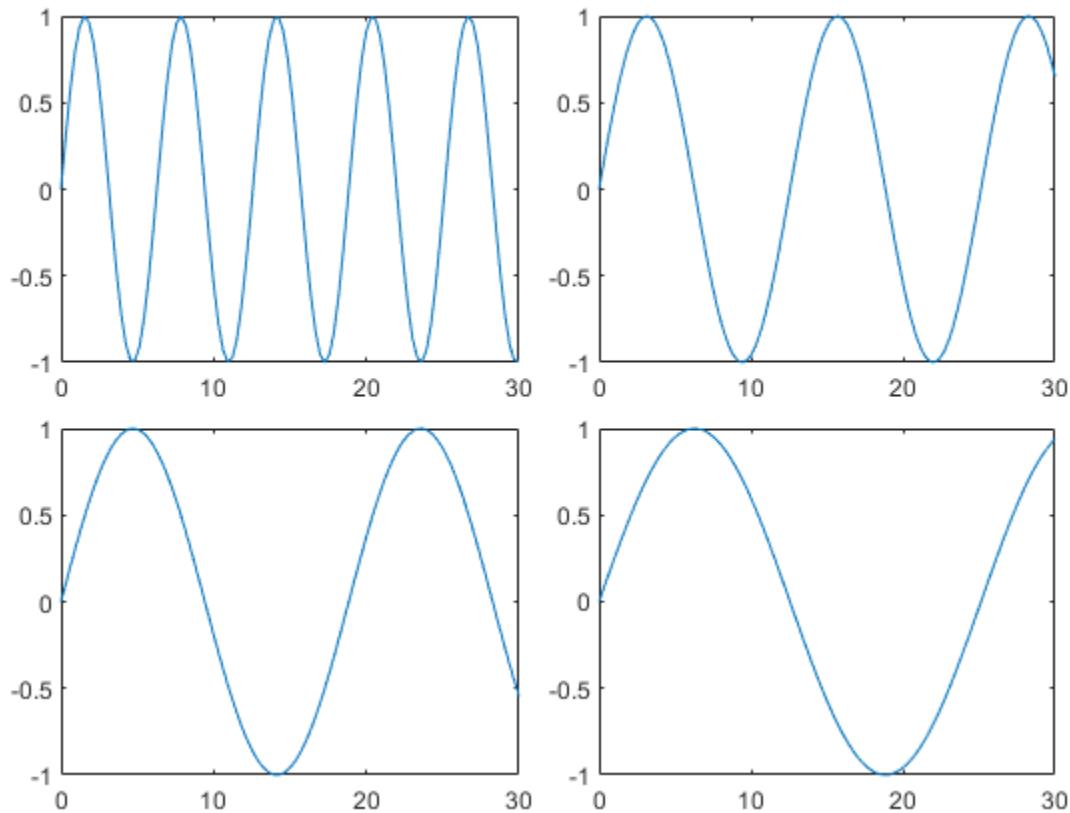
```
x = linspace(0,30);
y1 = sin(x);
```

```
y2 = sin(x/2);  
y3 = sin(x/3);  
y4 = sin(x/4);  
  
% Create plots  
t = tiledlayout(2,2);  
nexttile  
plot(x,y1)  
nexttile  
plot(x,y2)  
nexttile  
plot(x,y3)  
nexttile  
plot(x,y4)
```



通过将 Padding 和 TileSpacing 属性设置为 'compact'，减小布局四周和每个图块周围的间距。

```
t.Padding = 'compact';  
t.TileSpacing = 'compact';
```



显示共享标题和轴标签

您可以在布局中显示共享标题和共享轴标签。创建一个 2×1 布局 t 。然后显示一个线图和一个针状图。通过调用 `linkaxes` 函数来同步 x 轴范围。

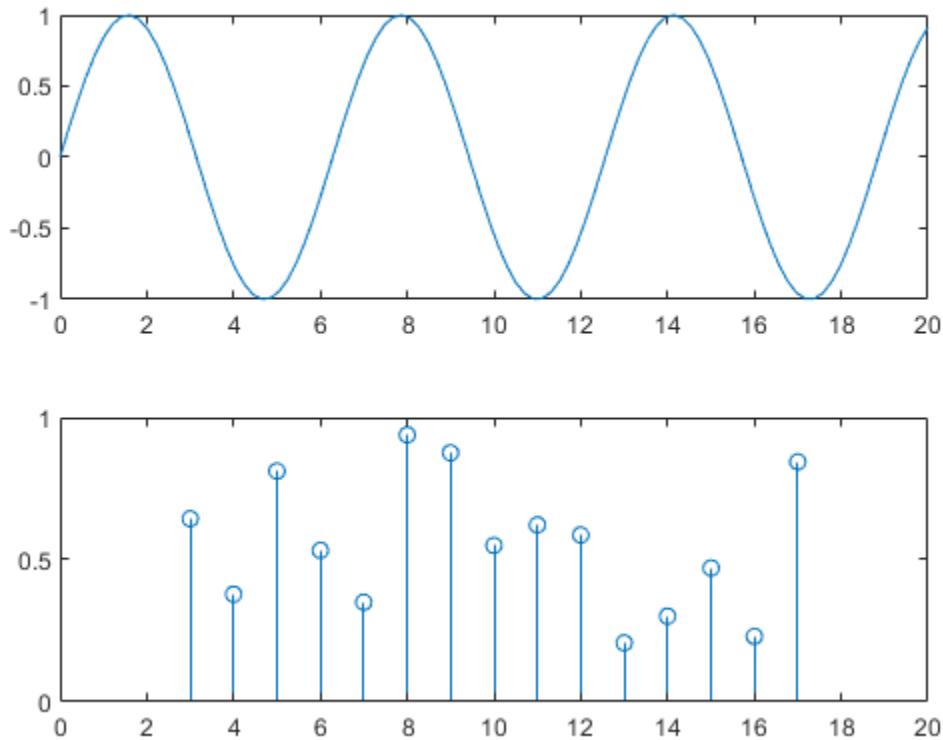
```

x1 = linspace(0,20,100);
y1 = sin(x1);
x2 = 3:17;
y2 = rand(1,15);

% Create plots.
t = tiledlayout(2,1);
ax1 = nexttile;
plot(ax1,x1,y1)
ax2 = nexttile;
stem(ax2,x2,y2)

% Link the axes
linkaxes([ax1,ax2],'x');

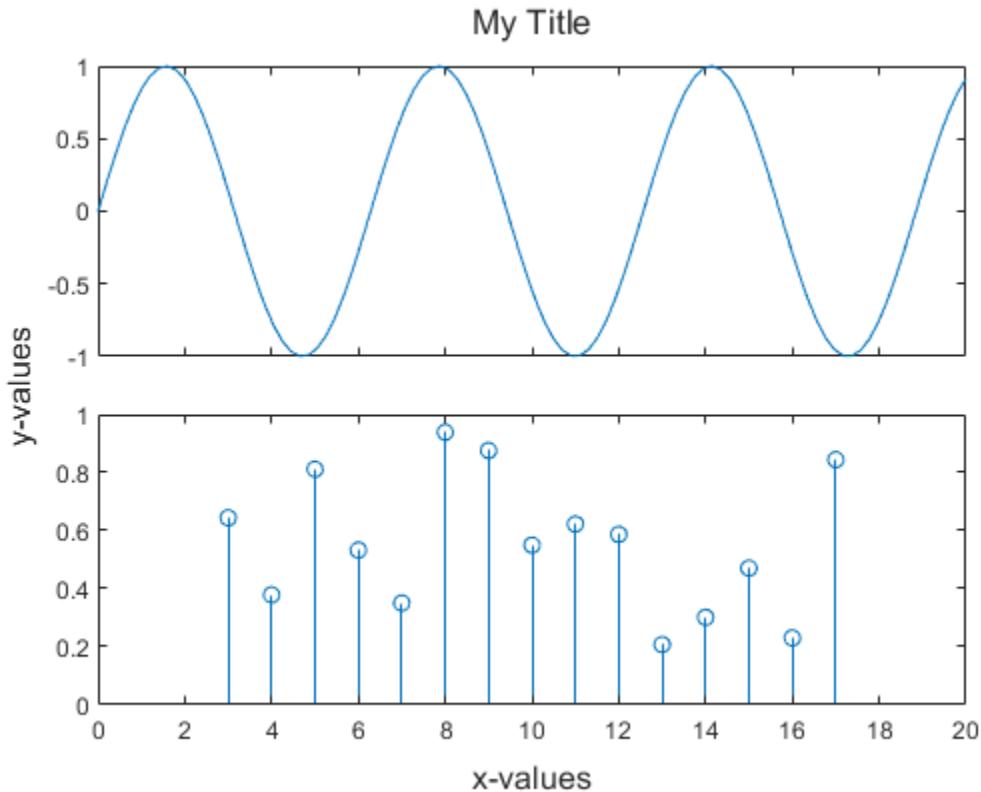
```



通过将 `t` 传递给 `title`、`xlabel` 和 `ylabel` 函数，添加共享标题和共享轴标签。通过从顶部图中删除 `x` 轴刻度标签，并将 `t` 的 `TileSpacing` 属性设置为 '`compact`'，让图彼此更靠近。

```
% Add shared title and axis labels
title(t,'My Title')
xlabel(t,'x-values')
ylabel(t,'y-values')

% Move plots closer together
xticklabels(ax1,{})
t.TileSpacing = 'compact';
```



另请参阅

函数

tiledlayout | nexttile | title | hold

相关示例

- “创建包含双 y 轴的图。” (第 9-32 页)
- “指定坐标轴刻度值和标签” (第 9-9 页)

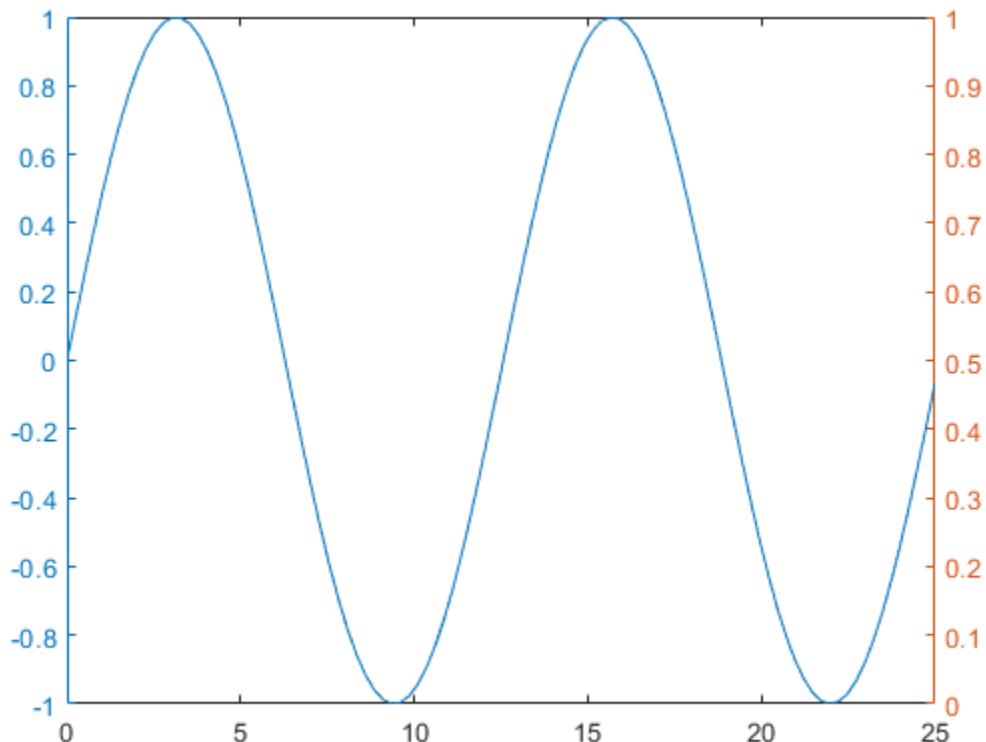
创建包含双 y 轴的图。

以下示例演示如何使用 `yyaxis` 函数创建左侧和右侧带有 y 轴的图表。此外，它还演示如何标记每个轴、合并多个绘图以及清除与两侧或其中一侧关联的绘图。

绘制数据对左侧 y 轴的图。

创建左右两侧都有 y 轴的坐标区。`yyaxis left` 命令用于创建坐标区并激活左侧。后续图形函数（例如 `plot`）的目标为活动侧。绘制数据对左侧 y 轴的图。

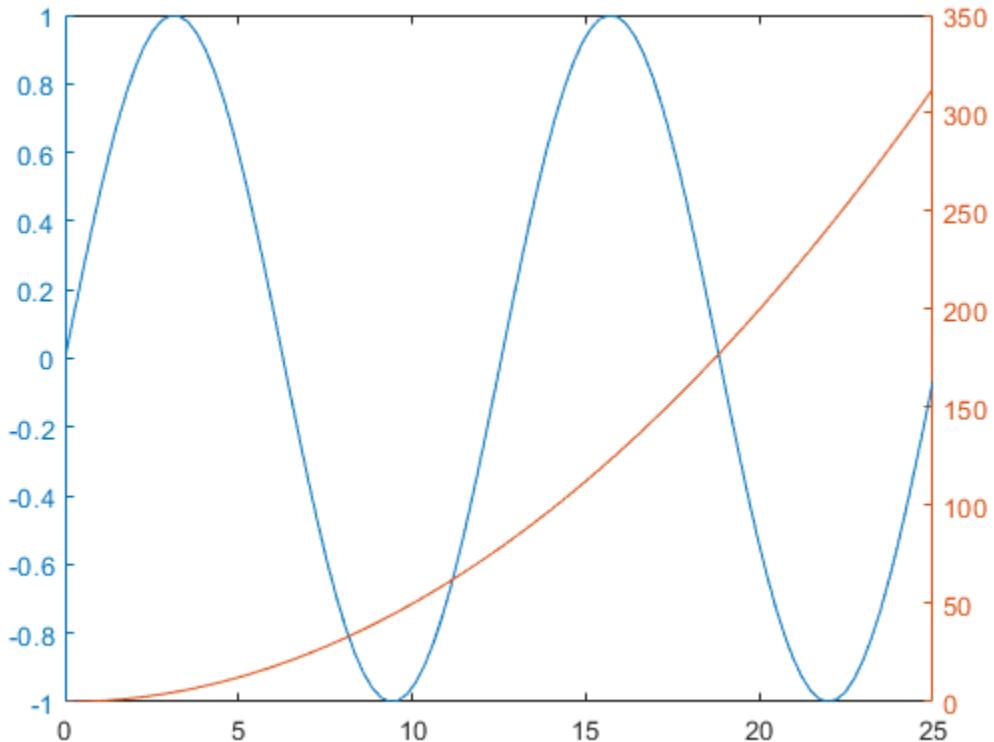
```
x = linspace(0,25);
y = sin(x/2);
yyaxis left
plot(x,y);
```



绘制数据对右侧 y 轴的图。

使用 `yyaxis right` 激活右侧。然后，绘制一组数据对右侧 y 轴的图。

```
r = x.^2/2;
yyaxis right
plot(x,r);
```

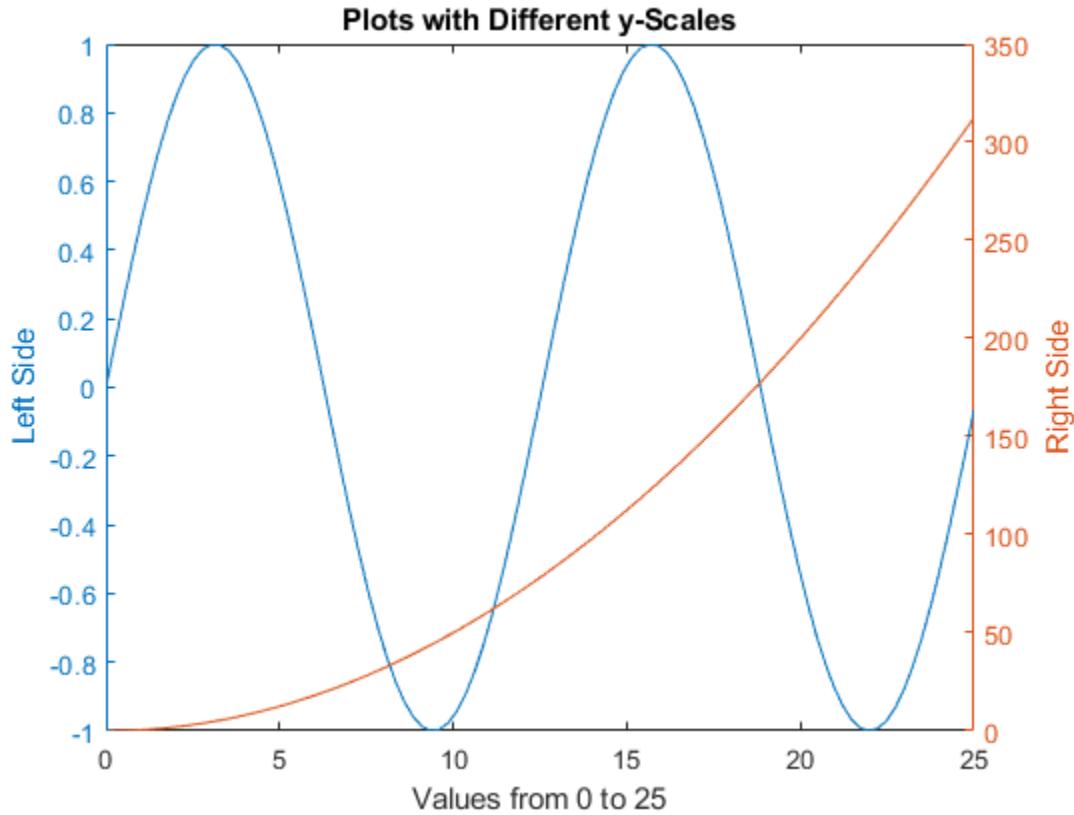


添加标题和轴标签

使用 `yyaxis left` 和 `yyaxis right` 命令控制坐标区的哪一侧为活动侧。然后添加标题和轴标签。

```
yyaxis left
title('Plots with Different y-Scales')
xlabel('Values from 0 to 25')
ylabel('Left Side')

yyaxis right
ylabel('Right Side')
```



基于每一侧绘制其他数据图

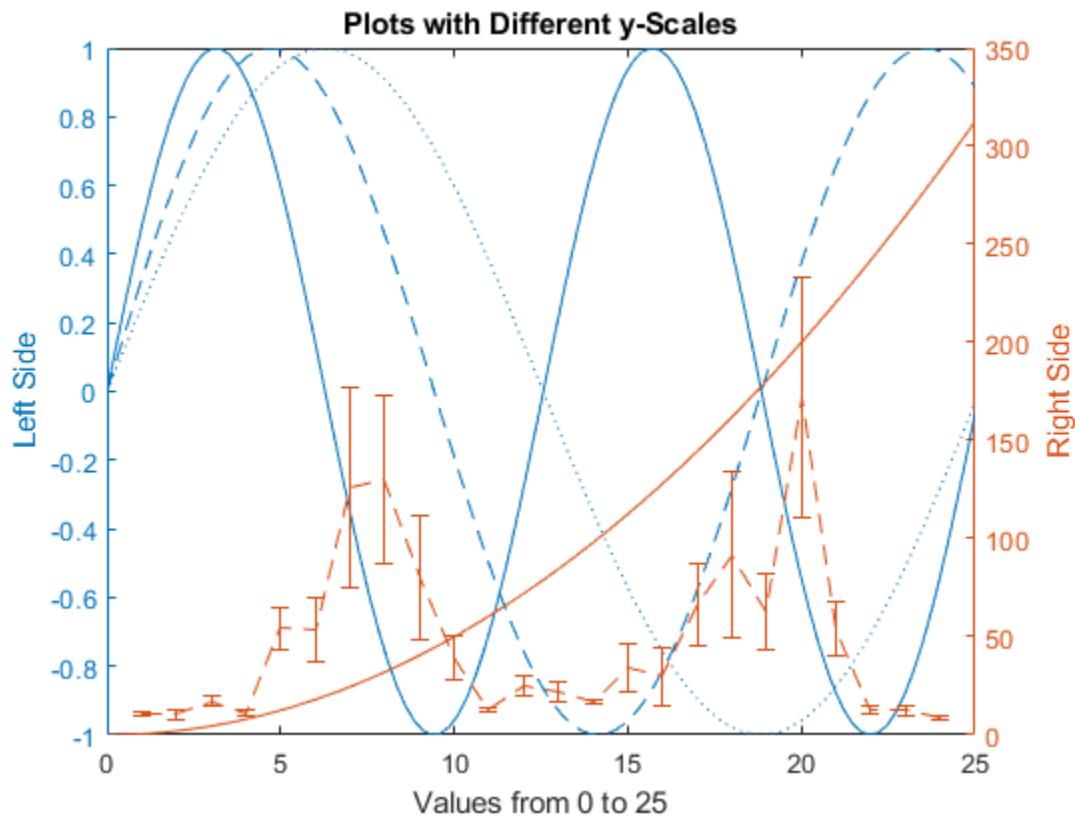
使用 `hold on` 命令再向左侧添加两个线条。向右侧添加一个误差条。新图与对应的 y 轴使用相同的颜色，并循环使用线型序列。`hold on` 命令同时影响左右两侧。

`hold on`

```
yyaxis left
y2 = sin(x/3);
plot(x,y2);
y3 = sin(x/4);
plot(x,y3);

yyaxis right
load count.dat;
m = mean(count,2);
e = std(count,1,2);
errorbar(m,e)
```

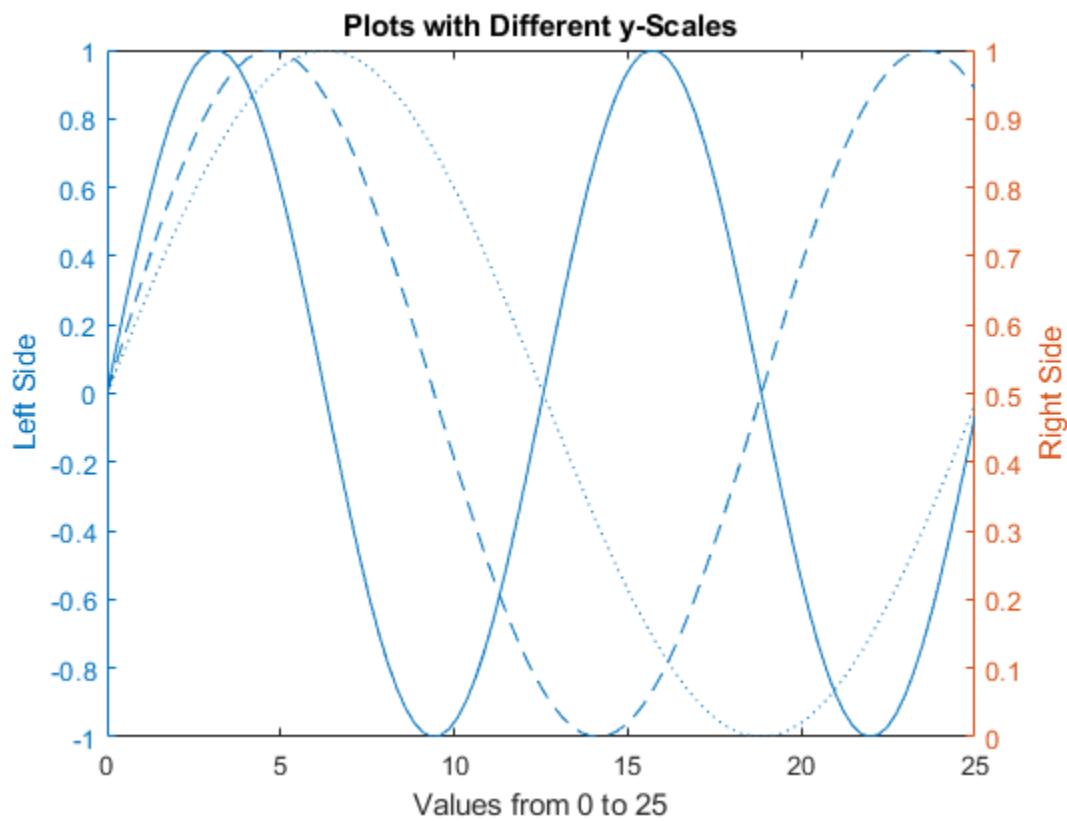
`hold off`



清除坐标区的一侧

首先激活右侧 y 轴，然后使用 `cla` 命令清除右侧 y 轴的数据。

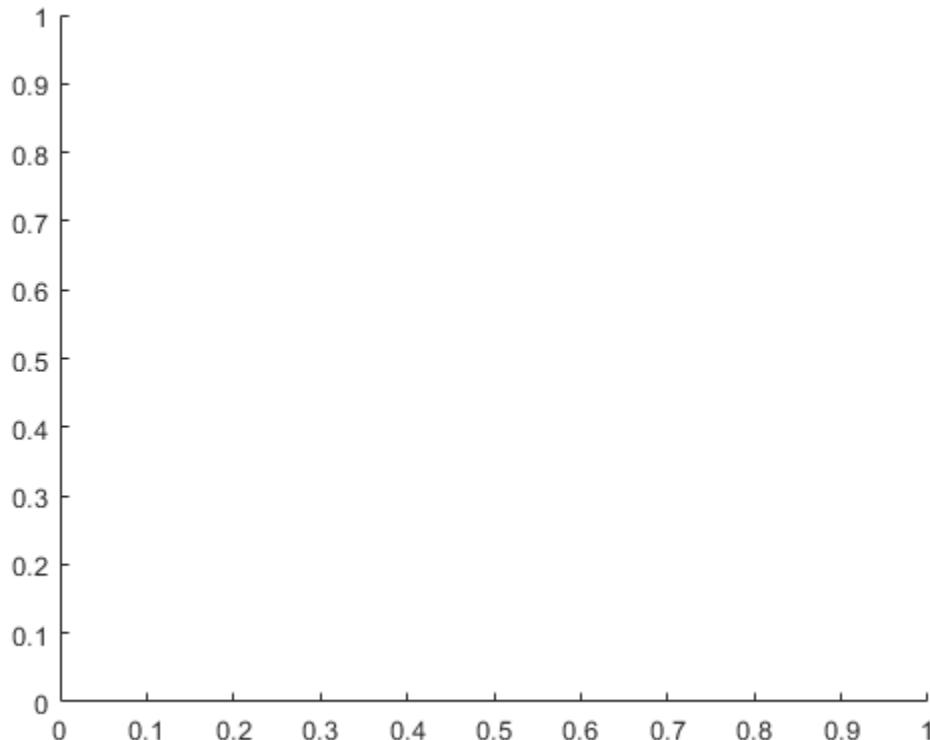
```
yyaxis right  
cla
```



清除坐标区并删除右侧 y 轴

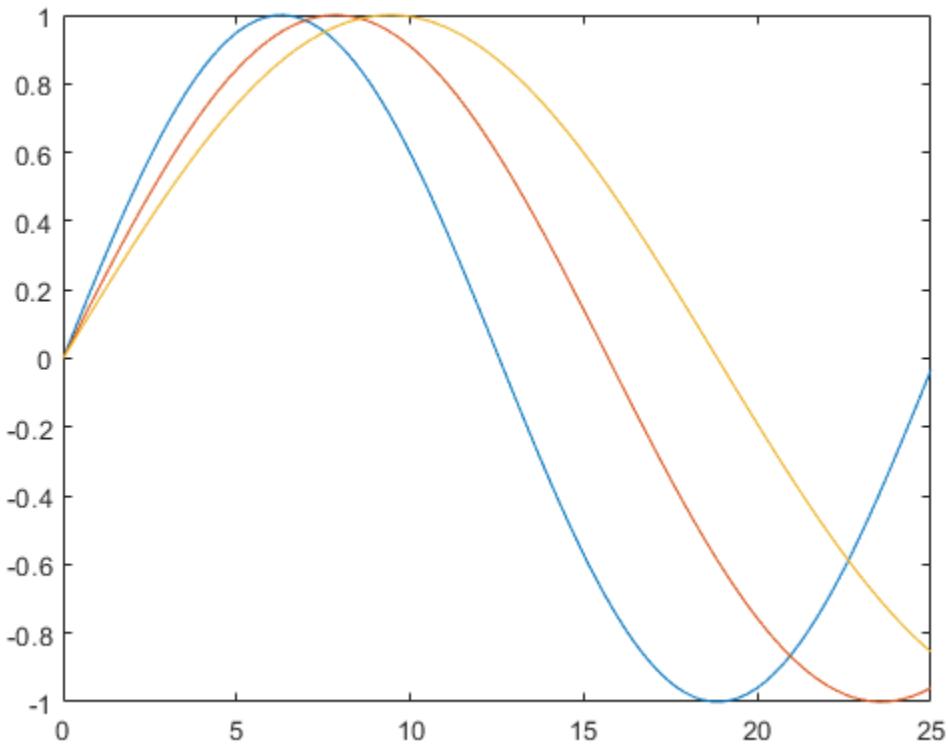
使用 `cla reset` 清除整个坐标区数据并删除右侧的 y 轴。

```
cla reset
```



现在，当您创建绘图时，绘图将仅包含一个 y 轴。例如，基于单个 y 轴绘制三条线条。

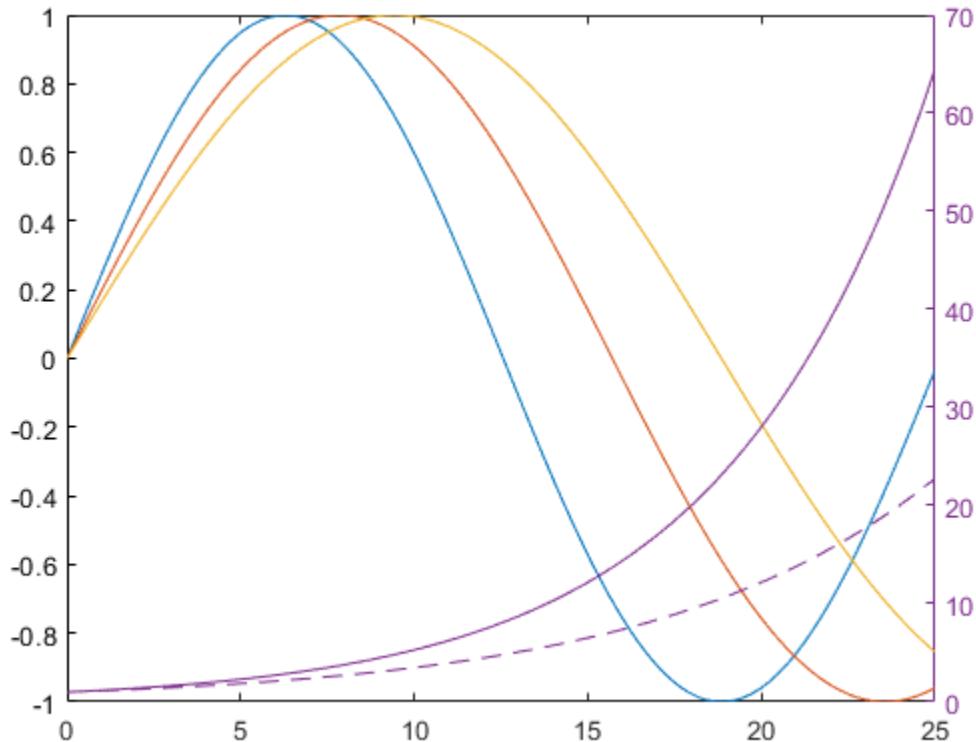
```
xx = linspace(0,25);
yy1 = sin(xx/4);
yy2 = sin(xx/5);
yy3 = sin(xx/6);
plot(xx,yy1,xx,yy2,xx,yy3)
```



将第二个 y 轴添加到现有图形

使用 `yyaxis right` 向现有图表添加第二个 y 轴。现有绘图和左侧的 y 轴不会更改颜色。右侧 y 轴将使用坐标区色序中的下一种颜色。添加到坐标区中的新绘图使用与对应的 y 轴相同颜色。

```
yyaxis right  
rr1 = exp(xx/6);  
rr2 = exp(xx/8);  
plot(xx,rr1,xx,rr2)
```



另请参阅

函数

`yyaxis | ylabel | xlabel | title | hold | cla | plot`

相关示例

- “修改包含两个 y 轴的图形的属性” (第 9-40 页)
- “合并多个绘图” (第 9-24 页)

修改包含两个 y 轴的图形的属性

本节内容

- “更改坐标区属性”（第 9-40 页）
- “更改标尺属性”（第 9-42 页）
- “使用默认色序指定颜色”（第 9-44 页）

使用 `yyaxis` 函数可创建一个左右两侧都有 y 轴的 `Axes` 对象。与 y 轴有关的坐标区属性有两个值。但是，MATLAB 仅允许访问活动侧的值。例如，如果左侧处于活动状态，则 `Axes` 对象的 `YDir` 属性包含左侧 y 轴的方向。同样，如果右侧处于活动状态，则 `YDir` 属性包含右侧的 y 轴的方向。有一个例外是 `YAxis` 属性，它包含由两个标尺对象（每个 y 轴各一个）组成的数组。

您可以按照下面两种方式中的任何一种，来更改特定 y 轴的外观和行为：

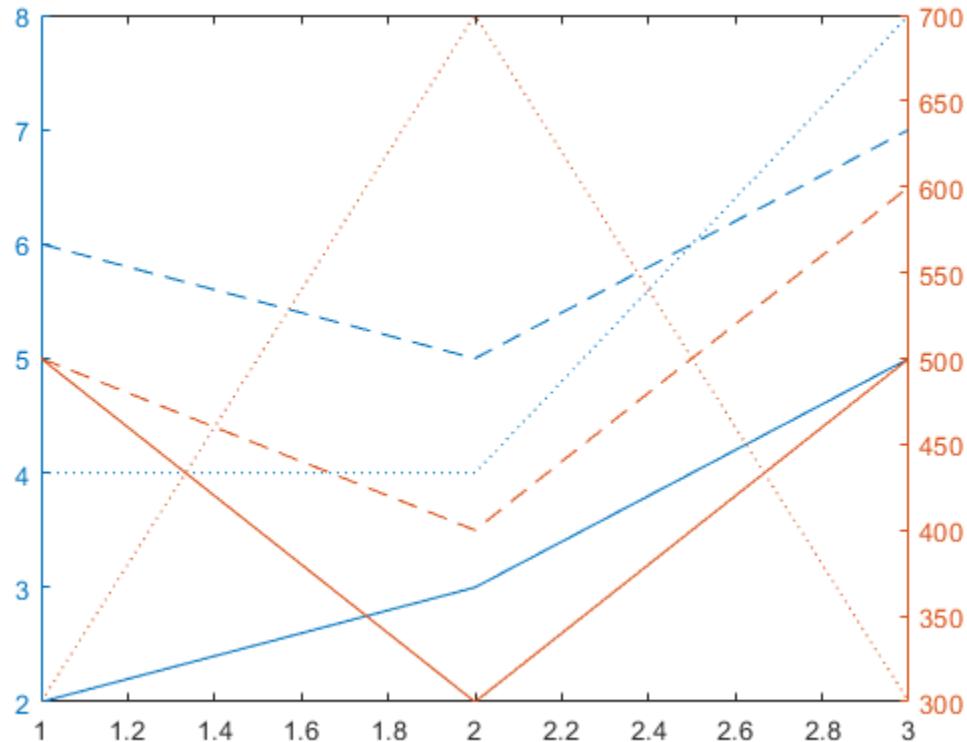
- 设置活动侧，然后更改 `Axes` 对象的属性值。
- 通过 `Axes` 对象的 `YAxis` 属性访问标尺对象，然后更改该标尺对象的属性值。

更改坐标区属性

通过设置 `Axes` 属性修改包含双 y 轴的图形的属性。

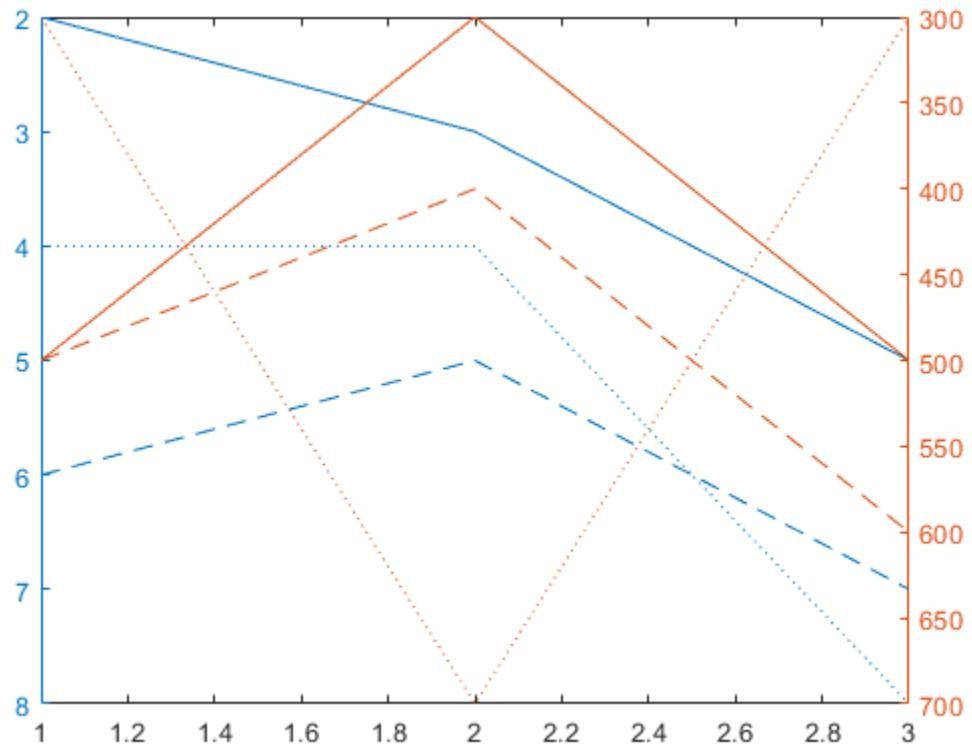
创建一个包含两个 y 轴的图表，然后绘制数据图。

```
x = [1 2 3];
y1 = [2 6 4; 3 5 4; 5 7 8];
y2 = 100*[5 5 3; 3 4 7; 5 6 3];
figure
yyaxis left
plot(x,y1)
yyaxis right
plot(x,y2)
```



反转每个 y 轴值递增的方向。使用 `yyaxis left` 激活左侧，并设置左侧 y 轴的方向。同样，使用 `yyaxis right` 激活右侧。然后，设置右侧 y 轴的方向。

```
ax = gca;
yyaxis left
ax.YDir = 'reverse';
yyaxis right
ax.YDir = 'reverse';
```



更改标尺属性

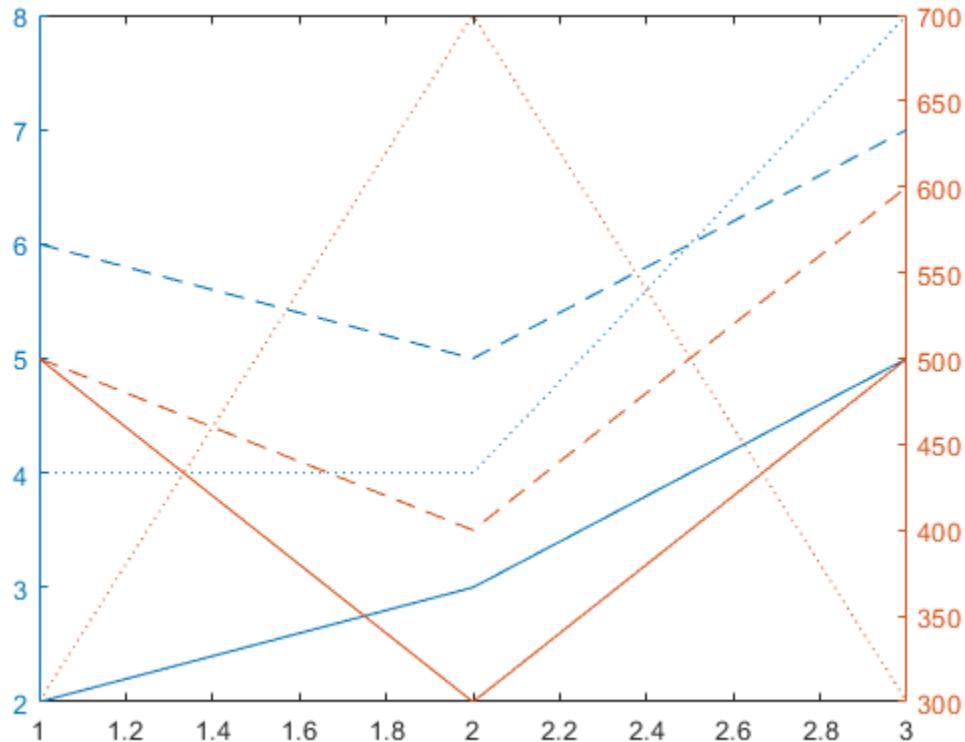
通过设置标尺属性修改包含双 y 轴的图形的属性。

创建一个包含两个 y 轴的图表，然后绘制数据图。

```

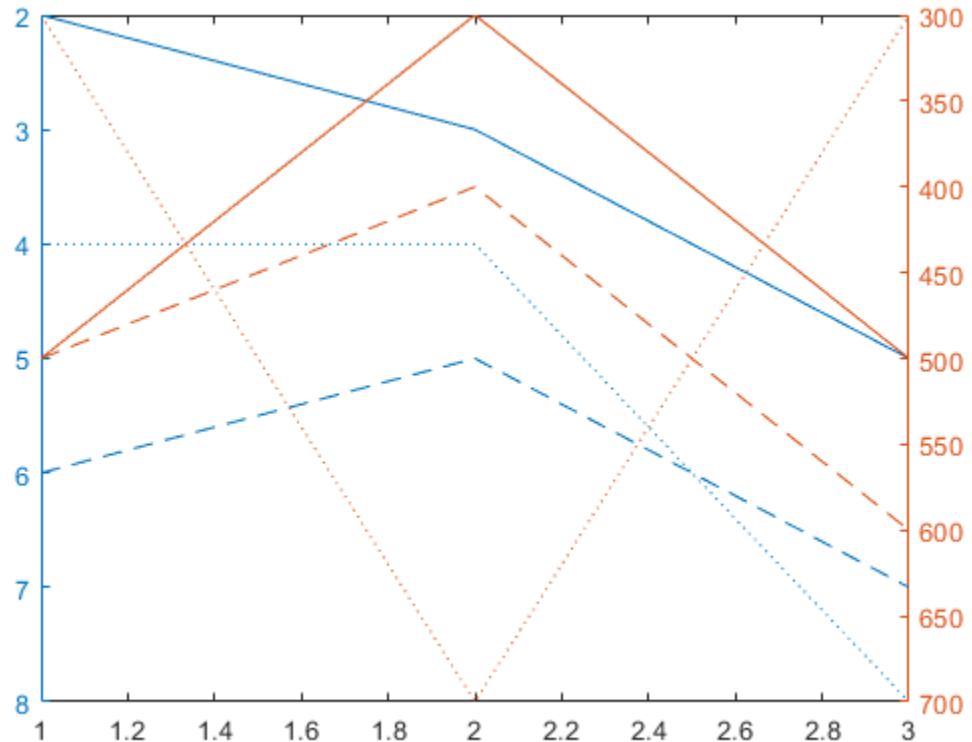
x = [1 2 3];
y1 = [2 6 4; 3 5 4; 5 7 8];
y2 = 100*[5 5 3; 3 4 7; 5 6 3];
figure
yyaxis left
plot(x,y1)
yyaxis right
plot(x,y2)

```



通过设置与每个轴相关联的标尺对象的属性，反转 y 轴值递增的方向。使用 `ax.YAxis(1)` 表示左侧标尺，`ax.YAxis(2)` 表示右侧标尺。

```
ax = gca;
ax.YAxis(1).Direction = 'reverse';
ax.YAxis(2).Direction = 'reverse';
```



使用默认色序指定颜色

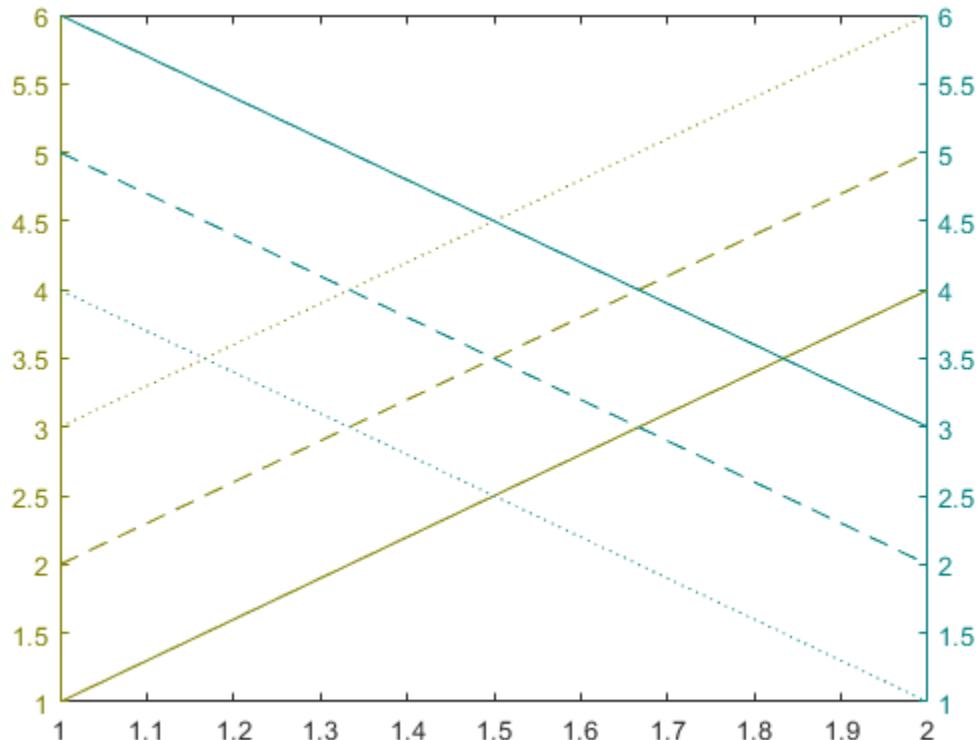
通过更改默认的坐标区色序，为带有两个 y 轴的图表指定颜色。

创建一个图窗。定义两个 RGB 颜色值，一个值对应左侧，一个值对应右侧。在创建坐标区前，先将默认的坐标区色序更改为这两种颜色。在图窗级别设置默认值，以使新颜色仅影响属于图窗 `fig` 子级的坐标区。新颜色不影响其他图窗中的坐标区。然后创建图形。

```
fig = figure;
left_color = [.5 .5 0];
right_color = [0 .5 .5];
set(fig,'defaultAxesColorOrder',[left_color; right_color]);

y = [1 2 3; 4 5 6];
yyaxis left
plot(y)

z = [6 5 4; 3 2 1];
yyaxis right
plot(z)
```



另请参阅

函数

`yyaxis` | `plot`

属性

`Axes` | `Numeric Ruler`

相关示例

- “创建包含双 y 轴的图。” (第 9-32 页)
- “默认属性值” (第 17-21 页)

使用多个刻度和坐标轴范围显示数据

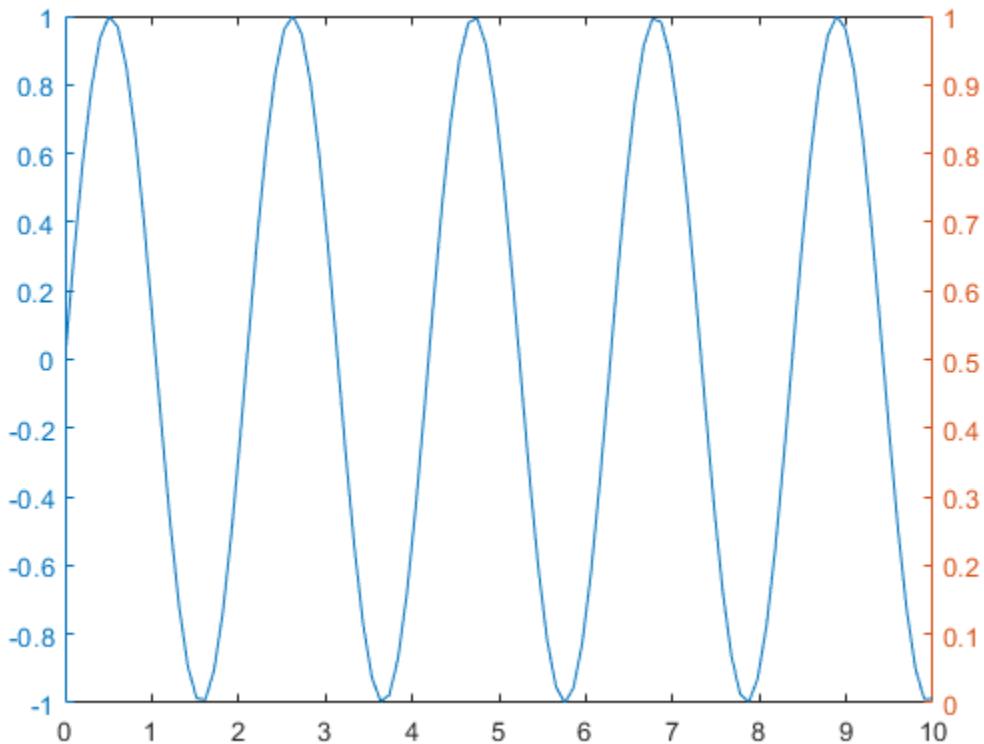
您可以使用多个刻度和坐标轴范围以多种方法来可视化数据。例如，您可以使用 `yyaxis` 函数创建具有两个 y 轴的绘图。要创建具有多个 x 轴和 y 轴、多个颜色栏的绘图，或创建具有分成若干区间的不连续轴的绘图，请使用 `tiledlayout` 函数。

用两个 y 轴显示数据

使用 `yyaxis` 函数创建一个具有两个 y 轴的绘图。例如，您可以使用两个 y 轴以不同刻度绘制两条线。

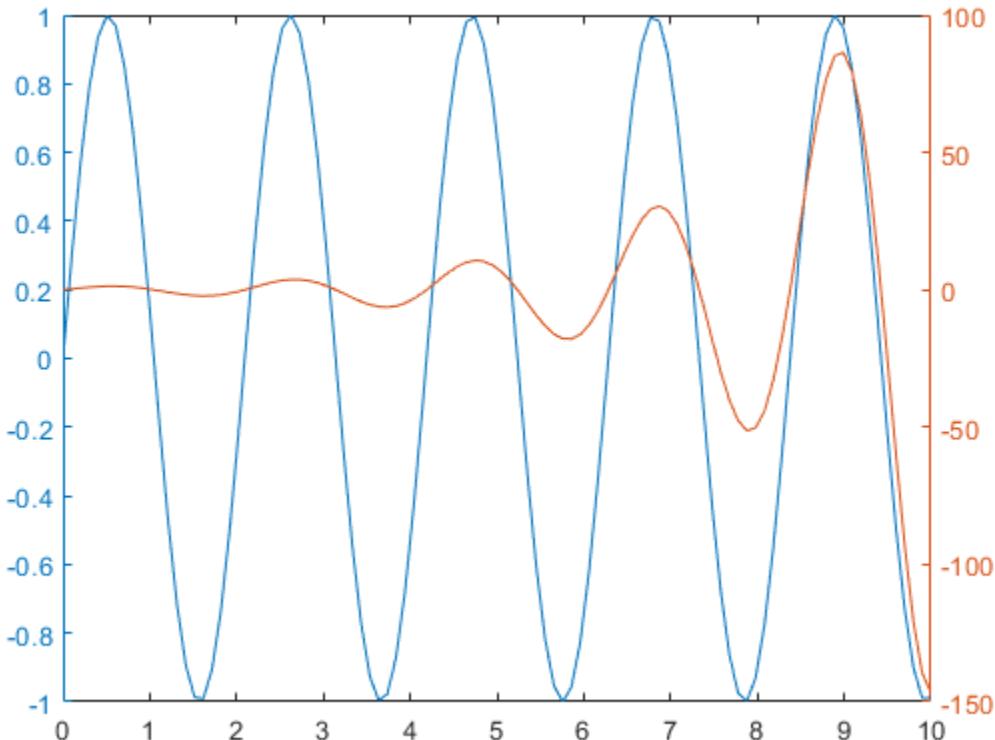
创建一个坐标区对象，并通过调用 `yyaxis left` 来激活左侧 y 轴。然后绘制一个正弦波。

```
figure
yyaxis left
x = linspace(0,10);
y = sin(3*x);
plot(x,y)
```



通过调用 `yyaxis right` 激活右侧 y 轴。然后绘制一个放大的正弦波。

```
yyaxis right
y2 = sin(3*x).*exp(0.5*x);
plot(x,y2)
```



用多个 x 轴和 y 轴显示数据

自 R2019b 开始提供

要用不同的 x 轴和 y 轴绘制两组数据，请在分块图布局中创建两个不同的坐标区对象。在其中一个坐标区对象中，将 x 轴移至图框的顶部，并将 y 轴移至图框的右侧。

例如，您可以创建两个具有不同 x 轴和 y 轴范围的图。

首先，创建两组 x 和 y 坐标。

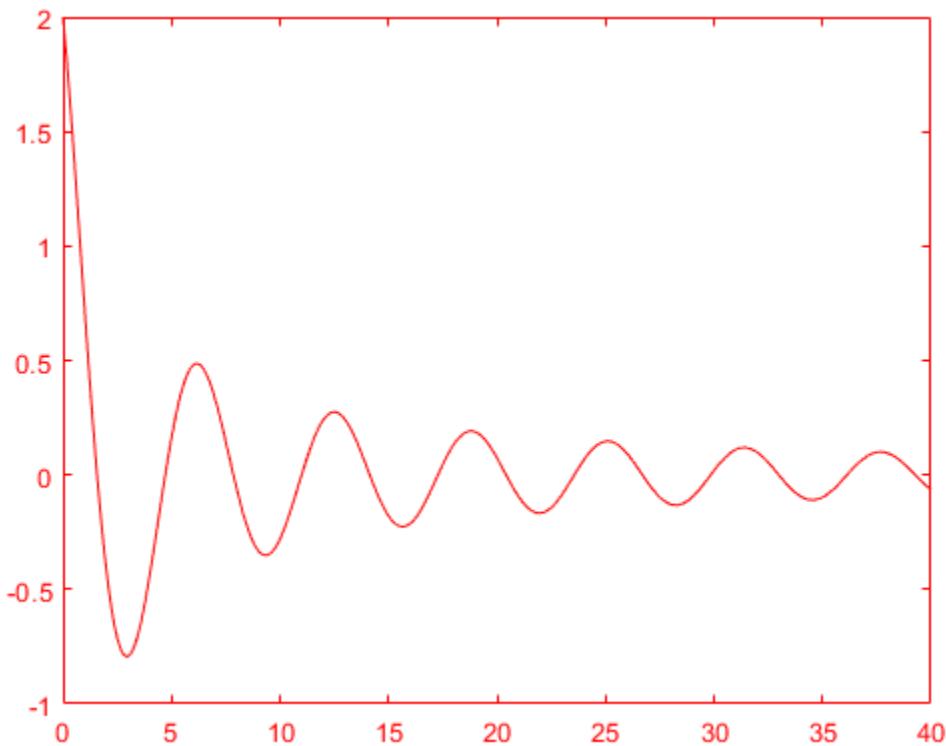
```
x1 = 0:0.1:40;
y1 = 4.*cos(x1)./(x1+2);
x2 = 1:0.2:20;
y2 = x2.^2./x2.^3;
```

创建一个分块图布局和一个坐标区对象。然后在坐标区中绘图：

- 创建一个 1×1 分块图布局 `t`。
- 通过调用 `axes` 函数并将 `t` 指定为父对象，创建坐标区对象 `ax1`。
- 将 `x1` 和 `y1` 绘制为一条红线，并将 `ax1` 指定为目标坐标区。
- 更改 x 轴和 y 轴的颜色以匹配绘制的线条。在绘制后设置坐标区属性可确保设置保持不变。

```
t = tiledlayout(1,1);
ax1 = axes(t);
```

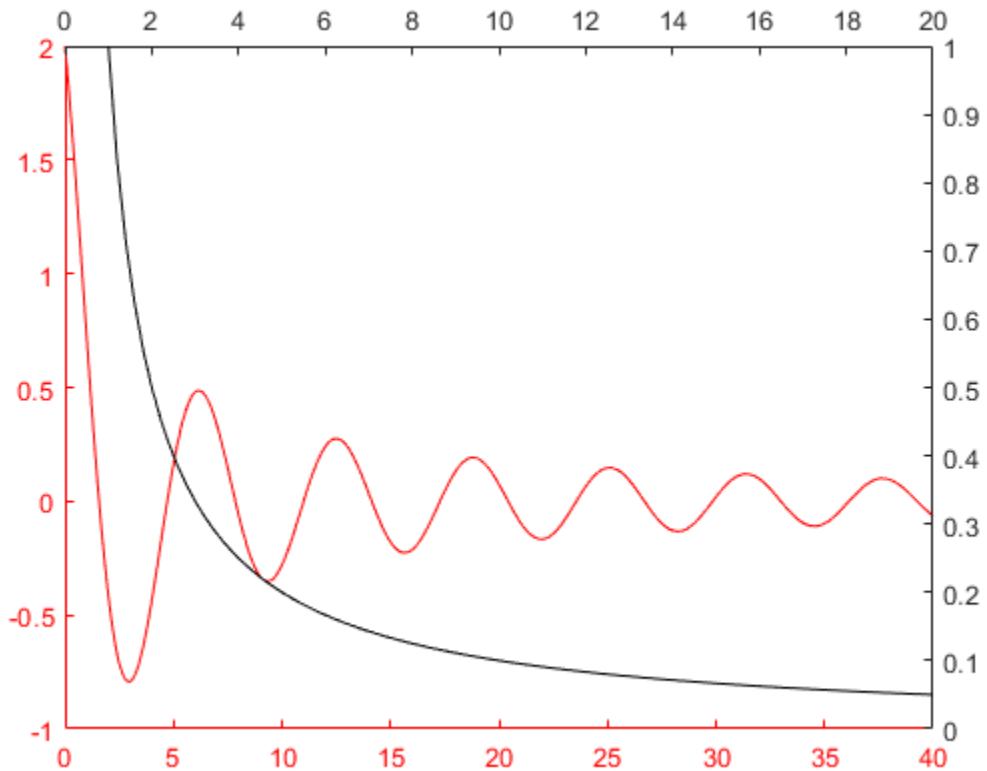
```
plot(ax1,x1,y1,'r')
ax1.XColor = 'r';
ax1.YColor = 'r';
```



创建第二个坐标区对象，并用黑色而不是红色绘制第二组数据。然后，设置第二个坐标区对象的设置属性以移动 x 轴和 y 轴，确保两个图不彼此遮挡。

- 通过调用 `axes` 函数并将 `t` 指定为父对象，创建坐标区对象 `ax2`。
- 将 `x2` 和 `y2` 绘制为一条黑线，并将 `ax2` 指定为目标坐标区。
- 将 `x` 轴移至顶部，并将 `y` 轴移至右侧。
- 将坐标区对象的颜色设置为 '`none`'，以使底层图可见。
- 关闭图框，以防止框边缘遮挡 `x` 和 `y` 轴。

```
ax2 = axes(t);
plot(ax2,x2,y2,'k')
ax2.XAxisLocation = 'top';
ax2.YAxisLocation = 'right';
ax2.Color = 'none';
ax1.Box = 'off';
ax2.Box = 'off';
```



在不连续的 x 轴上绘制数据

自 R2019b 开始提供

您可以使用分块图布局将绘图显示为沿一个轴分成若干区间。例如，您可能希望排除 x 轴的某个部分，以关注其他感兴趣的区域。

创建坐标向量 x 和 y。

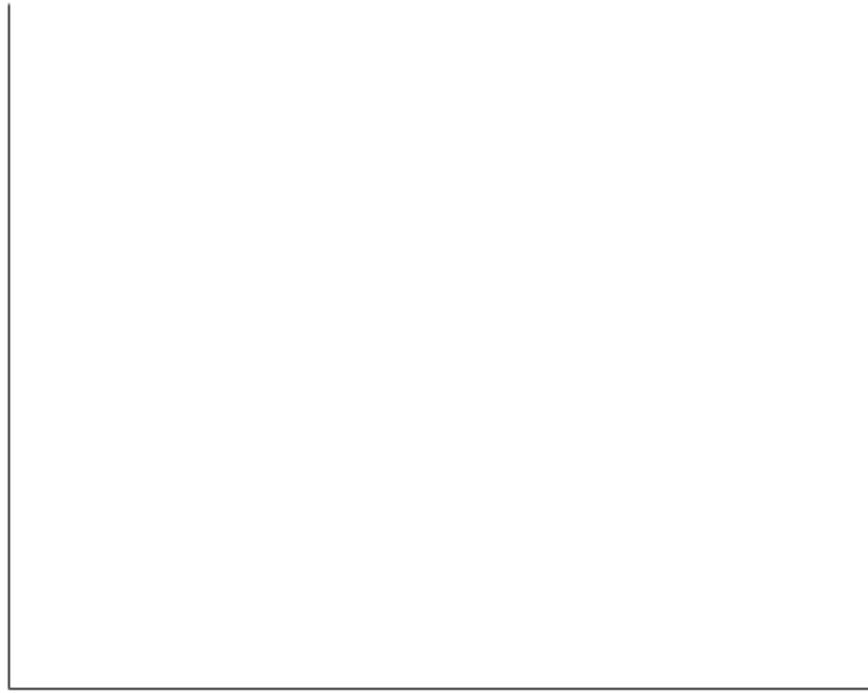
```
x = 0:0.1:60;
y = 4.*cos(x)./(x+2);
```

创建一个包含两个图块的分块图布局，并放置一个跨这两个图块的坐标区对象。在最后的演示中，此坐标区对象将出现在背景中，即显示在另外两个坐标区对象的后面。其 x 轴的一部分将可见，以显示一个长 x 轴。

- 创建一个 1×2 分块图布局 t，并指定紧凑的图块间距。通过设置图块间距，您可以控制 x 轴上各区间之间的间隙大小。
- 通过调用 axes 函数并将 t 指定为父对象，创建背景坐标区 bgAx。指定名称-值参数以删除所有刻度并关闭图框。
- 通过将 bgAx 的 Layout.TileSpan 属性设置为 [1 2]，使背景坐标区跨两个图块。

```
figure
t = tiledlayout(1,2,'TileSpacing','compact');
```

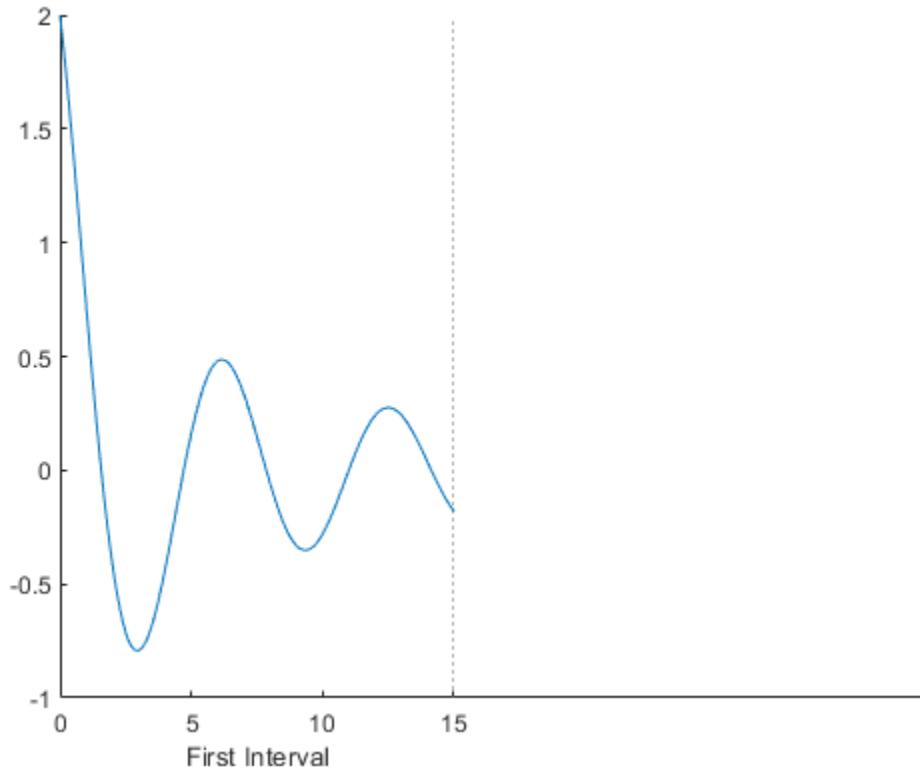
```
bgAx = axes(t,'XTick',[],'YTick',[],'Box','off');
bgAx.Layout.TileSpan = [1 2];
```



在第一个图块中的 **bgAx** 前面创建一个坐标区对象。绘制 x 和 y ，并将 x 轴范围设置为第一个区间：

- 通过调用 **axes** 函数并将 **t** 指定为父对象，创建 **ax1**。默认情况下，坐标区位于第一个图块中。
- 在 **ax1** 中绘制 x 和 y 。
- 调用 **xline** 函数，以在第一个区间的上限处显示一条垂直虚线。
- 将 x 轴范围设置为第一个区间 **[0 15]**。
- 添加轴标签以标识第一个区间。

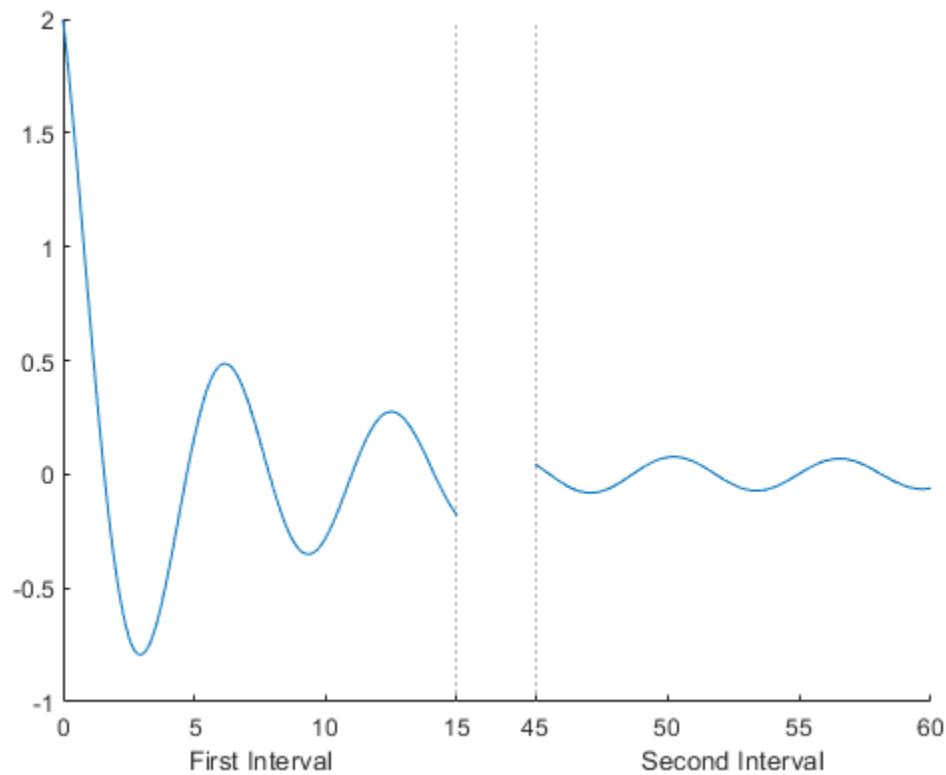
```
ax1 = axes(t);
plot(ax1,x,y)
xline(ax1,15,'r');
ax1.Box = 'off';
xlim(ax1,[0 15])
xlabel(ax1, 'First Interval')
```



重复该过程，以创建另一个坐标区对象并针对第二个区间绘图。默认情况下，坐标区出现在第一个图块中。通过将坐标区的 `Layout.Tile` 属性设置为 2，将其移至第二个图块。然后，链接这两个坐标区，使两个 y 轴的范围匹配。

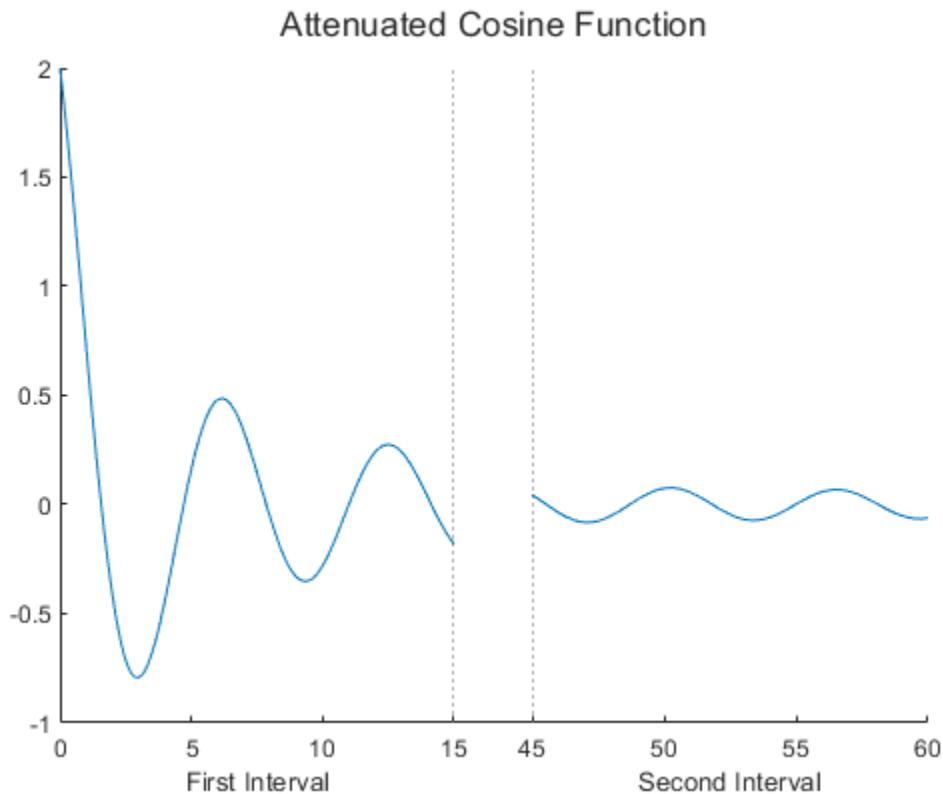
```
% Create second plot
ax2 = axes(t);
ax2.Layout.Tile = 2;
plot(ax2,x,y)
xline(ax2,45,'');
ax2.YAxis.Visible = 'off';
ax2.Box = 'off';
xlim(ax2,[45 60])
xlabel(ax2,'Second Interval')

% Link the axes
linkaxes([ax1 ax2], 'y')
```



要添加标题，请将分块图布局传递给 `title` 函数。

```
title(t,'Attenuated Cosine Function')
```



用不同的颜色栏显示两组数据

自 R2020b 开始提供

一个坐标区对象只能容纳一个颜色栏。要创建具有多个颜色栏的可视化效果，请在一个分块图布局中堆叠多个坐标区对象。仅使其中一个坐标区可见，但在布局的外侧图块中每个坐标区旁边显示一个颜色栏。

为两个气泡图创建坐标向量、大小数据和颜色数据。

```
x = 1:15;
n = 70 * randn(1,15) + 50;
y1 = n + x.^2;
y2 = n - linspace(1,225,15);
sz1 = rand(1,15);
sz2 = rand(1,15);
c = linspace(1,10,15);
```

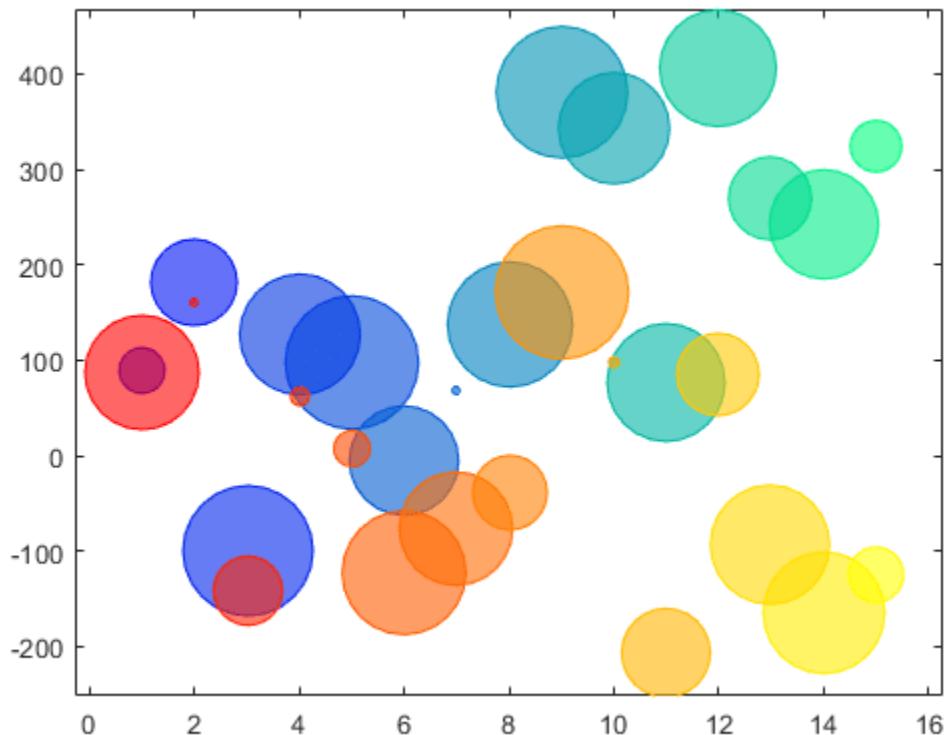
在分块图布局中堆叠两个坐标区对象，每个对象包含一个气泡图。

- 创建一个 1×1 分块图布局 `t`。
- 创建坐标区对象 `ax1` 并使用 `winter` 颜色图创建一个气泡图。
- 创建坐标区对象 `ax2` 并使用 `autumn` 颜色图创建一个气泡图。将 `Visible` 属性设置为 `'off'`，使此坐标区对象不可见。
- 链接各坐标区对象以使它们保持同步。在本例中，您可以将 `t` 的子代传递给 `linkaxes` 函数。您也可以将单个坐标区对象组成的向量传递给该函数。

```
% create first bubble chart with winter colormap
t = tiledlayout(1,1);
ax1 = axes(t);
bubblechart(ax1,x,y1,sz1,c)
colormap(ax1,'winter')

% create second bubble chart with autumn colormap
ax2 = axes(t);
bubblechart(ax2,x,y2,sz2,c)
colormap(ax2,'autumn')
ax2.Visible = 'off';

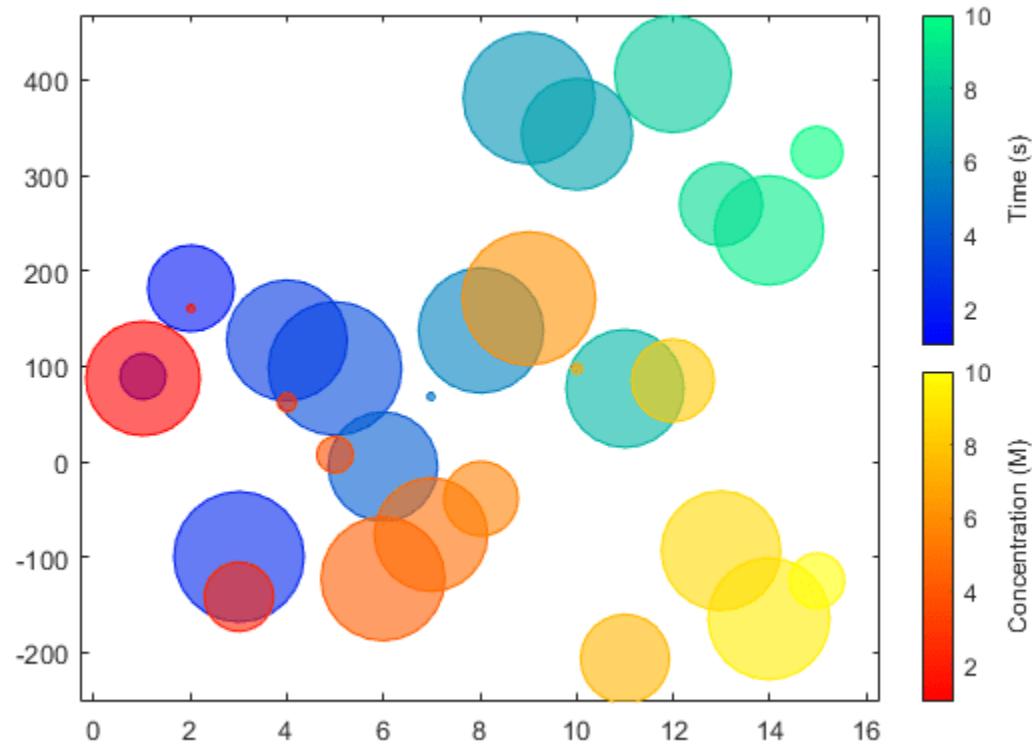
% link the limits of axes
linkaxes(t.Children)
```



在布局的东侧图块中为每个坐标区对象显示一个带标签的颜色栏。布局会排列这些颜色栏并使其保持对齐。

```
cb1 = colorbar(ax1);
cb1.Layout.Tile = 'east';
cb1.Label.String = 'Time (s)';

cb2 = colorbar(ax2);
cb2.Layout.Tile = 'east';
cb2.Label.String = 'Concentration (M)';
```



另请参阅

函数

[tiledlayout](#) | [yyaxis](#) | [axes](#)

相关示例

- “合并多个绘图” (第 9-24 页)
- “创建颜色栏” (第 10-2 页)

控制坐标轴长度比率和数据单位长度

本节内容

“图框纵横比”（第 9-56 页）

“数据纵横比”（第 9-58 页）

“还原为默认比率”（第 9-61 页）

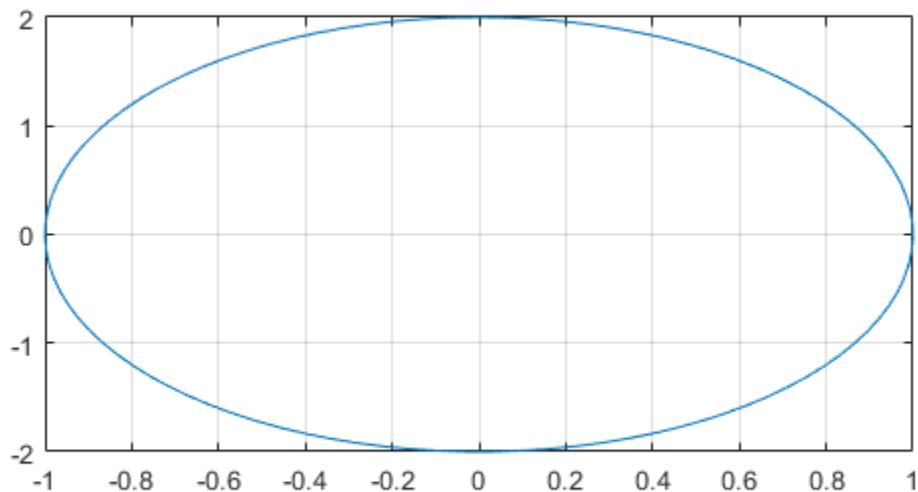
您可以控制 x 轴、y 轴和 z 轴的相对长度（图框纵横比），也可以控制一个数据单位沿每个轴的相对长度（数据纵横比）。

图框纵横比

图框纵横比是 x 轴、y 轴和 z 轴的相对长度。默认情况下，图框纵横比基于图窗大小。您可以使用 `pbaspect` 函数更改纵横比。将纵横比设置为一个由正值组成的三元素向量，这些正值表示相对坐标轴长度。

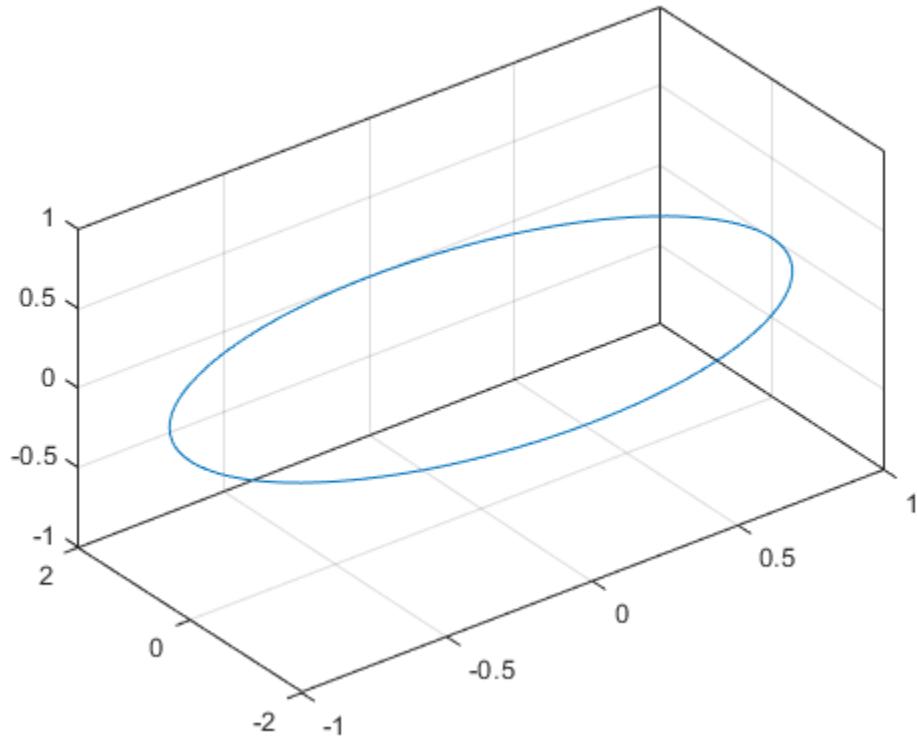
例如，绘制一个拉长的圆的图。然后设置图框纵横比，以使 x 轴是 y 轴和 z 轴（未显示）长度的两倍。

```
t = linspace(0,2*pi);
plot(sin(t),2*cos(t))
grid on
pbaspect([2 1 1])
```



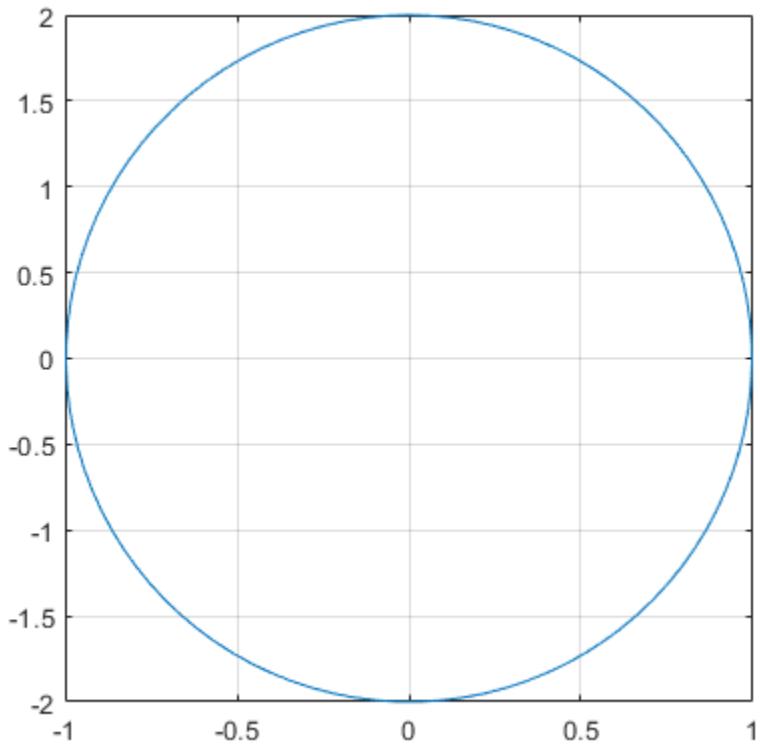
在三维视图中显示坐标区可查看 z 轴。

```
view(3)
```



对于方形坐标区，请使用 [1 1 1]。此值类似于使用 **axis square** 命令。

```
t = linspace(0,2*pi);
plot(sin(t),2*cos(t))
grid on
pbaspect([1 1 1])
```

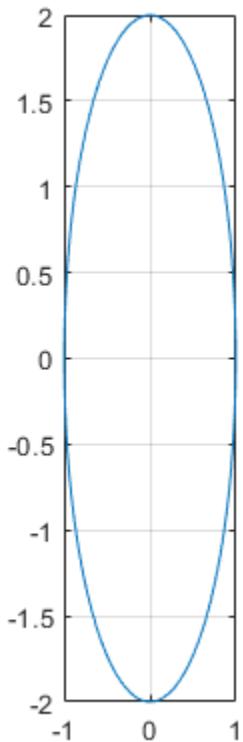


数据纵横比

数据纵横比是沿 x 轴、 y 轴和 z 轴的数据单位的相对长度。使用 `daspect` 函数可以更改数据纵横比。将数据纵横比设置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。

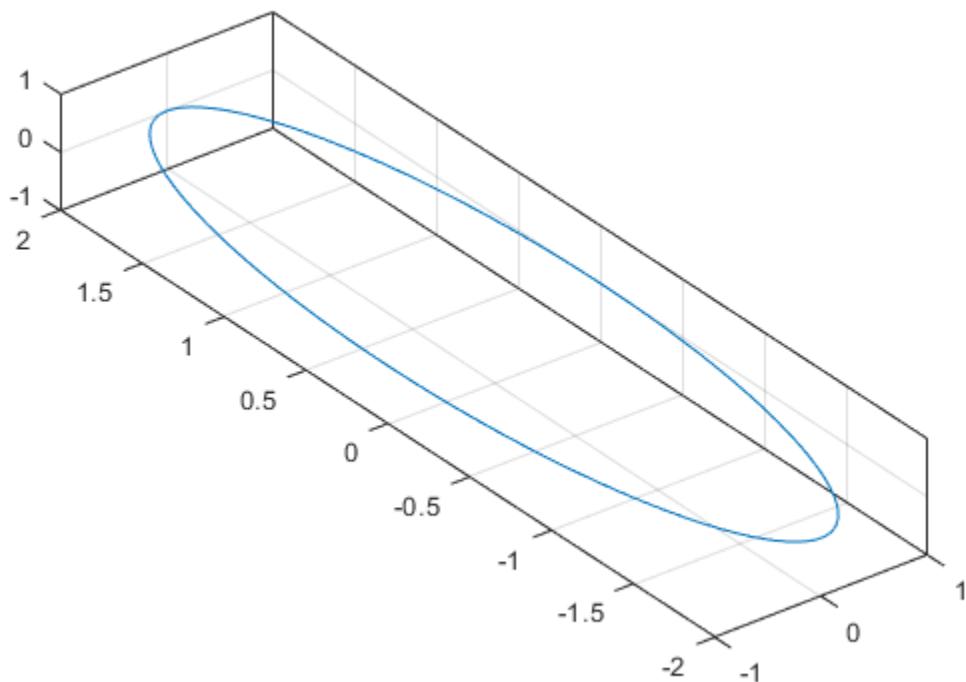
例如，设置此纵横比，以使沿 x 轴从 0 到 1 的长度等于沿 y 轴从 0 到 0.5 的长度和沿 z 轴（未显示） 0 到 2 的长度。

```
t = linspace(0,2*pi);
plot(sin(t),2*cos(t))
grid on
daspect([1 0.5 2])
```



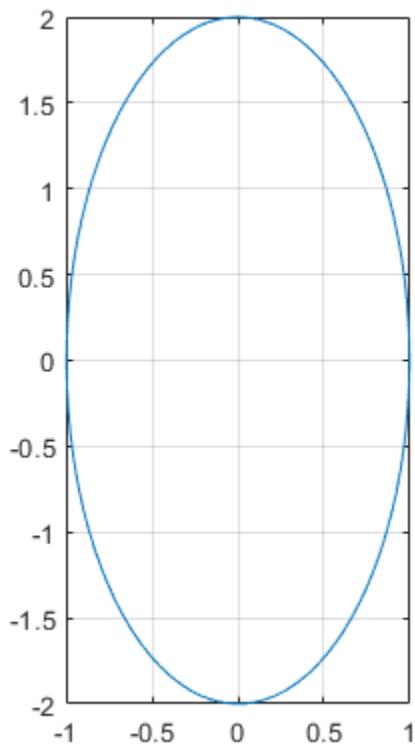
在三维视图中显示坐标区可查看 z 轴。

`view(3)`



若要在所有方向采用相等的数据单位，请使用 [1 1 1]。此值类似于使用 `axis equal` 命令。`x` 方向的一个数据单位与 `y` 和 `z` 方向的一个数据单位长度相同。

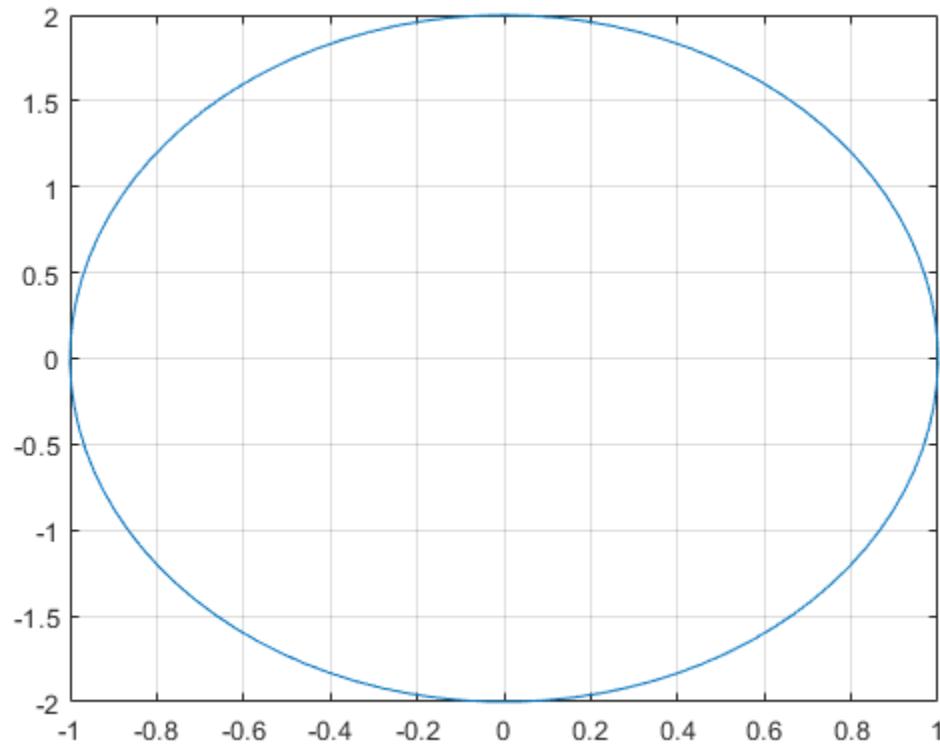
```
t = linspace(0,2*pi);
plot(sin(t),2*cos(t))
grid on
daspect([1 1 1])
```



还原为默认比率

更改数据纵横比。然后使用 `axis normal` 命令还原为默认图框和数据纵横比。

```
t = linspace(0,2*pi);
plot(sin(t),2*cos(t))
grid on
daspect([1 1 1])
axis normal
```



另请参阅

函数

`pbaspect` | `daspect` | `axis`

相关示例

- “指定坐标轴范围” (第 9-2 页)
- “控制坐标区布局” (第 9-63 页)

控制坐标区布局

本节内容

- “与坐标区位置相关的属性”（第 9-63 页）
- “位置和边距的边界”（第 9-63 页）
- “控制自动调整大小行为”（第 9-64 页）
- “伸展填充行为”（第 9-65 页）

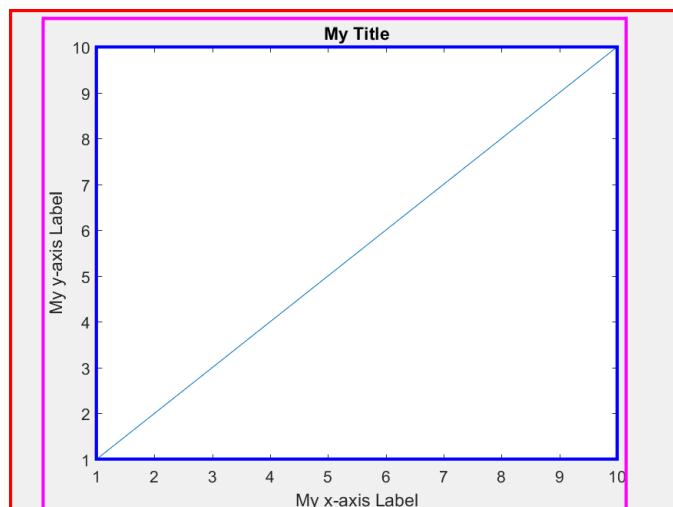
与坐标区位置相关的属性

Axes 对象拥有多项属性，可用于控制坐标区大小以及标题和轴标签在图窗中的布局。

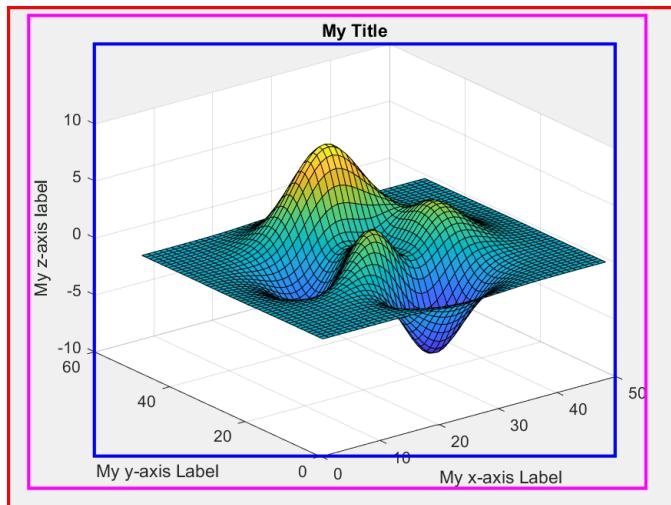
- **OuterPosition** - 坐标区的外边界，包括标题、标签和边距。以 [left bottom width height] 形式的向量指定此属性。**left** 和 **bottom** 值指示从图窗左下角到外边界左下角的距离。**width** 和 **height** 值指示外边界尺寸。
- **Position** - 绘图所在的内坐标区的边界，不包括标题、标签和边距。以 [left bottom width height] 形式的向量指定此属性。
- **TightInset** - 为 **Position** 属性值中的宽度和高度所添加的边距，指定为 [left bottom right top] 形式的向量。此属性是只读的。在添加轴标签和标题时，MATLAB 会更新这些值以适应文本。**Position** 和 **TightInset** 属性所定义的边界大小包含所有图形文本。
- **PositionConstraint** - Axes 对象的大小发生改变时保留下来的位置属性，指定为 '**outerposition**' 或 '**innerposition**'。
- **Units** - 位置单位。单位必须设置为 '**normalized**'（默认值）以启用自动调整坐标区大小。当位置单位为长度单位（例如英寸或厘米）时，Axes 对象为固定大小。

位置和边距的边界

下图显示了一个二维视图，其中 **OuterPosition** 值定义红色区域、**Position** 值定义蓝色区域，以及基于 **Position** 外扩 **TightInset** 值所定义的品红色区域。



下图显示了一个三维视图，同样包含了由 **OuterPosition** 值定义的坐标区外边界区域（红色）、**Position** 值定义的坐标区内边界区域（蓝色），以及在 **Position** 内边界基础上外扩 **TightInset** 值所定义的坐标区区域（品红色）。



控制自动调整大小行为

某些情况可能触发 **Axes** 对象自动调整大小。例如，以交互方式调整图窗大小或添加标题或轴标签将激活自动调整大小。有时，新坐标区的大小无法同时满足 **Position** 和 **OuterPosition** 值，这种情况下需要使用 **PositionConstraint** 属性来指示要保留哪个值。将 **PositionConstraint** 属性指定为下列值之一：

- '**outerposition**' - 保留 **OuterPosition** 值。如果不希望坐标区或任何周围文本超出特定外边界，可使用此选项。MATLAB 会调整坐标区内部区域大小（显示绘图时），以尽力在外边界范围内适应内容。
- '**innerposition**' - 保留 **InnerPosition** 值。如果希望坐标区的内部区域在图窗中保持特定大小，可使用此选项。此选项有时会导致文本溢出图窗。

通常，将 **PositionConstraint** 属性设置为 '**outerposition**' 更可取。但是，坐标区标题或标签过长可能会使坐标区内部区域大大缩小，文字过小而难于阅读。在这种情况下，最好保持特定大小的内部坐标区，即使周围文本溢出了图窗也应如此。

例如，创建一个带有两个坐标区的图窗，并为每个坐标区位置指定相同的宽度和高度。对于上坐标区，将 **PositionConstraint** 属性设置为 '**outerposition**'，对于下坐标区则设置为 '**innerposition**'。注意，在上坐标区中，内部区域会缩小以适应文本，但文本不会溢出图窗。在下坐标区中，会保留内部区域的大小，但部分文本被截断。

注意 以下代码使用了 **PositionConstraint** 属性，该属性是从 R2020a 开始引入的新属性。如果您使用的是较早的版本，请将 **ActivePositionProperty** 设置为 '**outerposition**' 或 '**position**'。

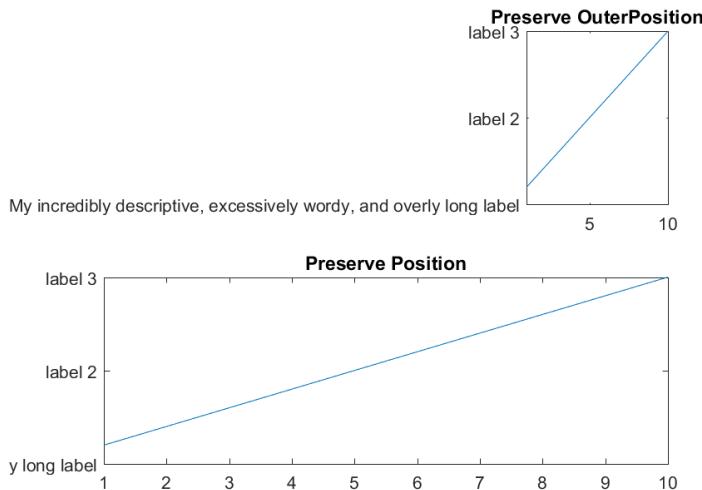
```
figure;
ax1 = axes('Position',[0.13 0.58 0.77 0.34]);
ax1.PositionConstraint = 'outerposition';
% R2019b and earlier: ax1.ActivePositionProperty = 'outerposition';
plot(ax1,1:10)
```

```

title(ax1,'Preserve OuterPosition')
yticklabels(ax1,{'My incredibly descriptive, excessively wordy, and overly long label',...
    'label 2','label 3'})

ax2 = axes('Position',[0.13 0.11 0.77 0.34]);
ax2.PositionConstraint = 'innerposition';
% R2019b and earlier: ax2.ActivePositionProperty = 'position';
plot(ax2,1:10)
title(ax2,'Preserve Position')
yticklabels(ax2,{'My incredibly descriptive, excessively wordy, and overly long label',...
    'label 2','label 3'})

```

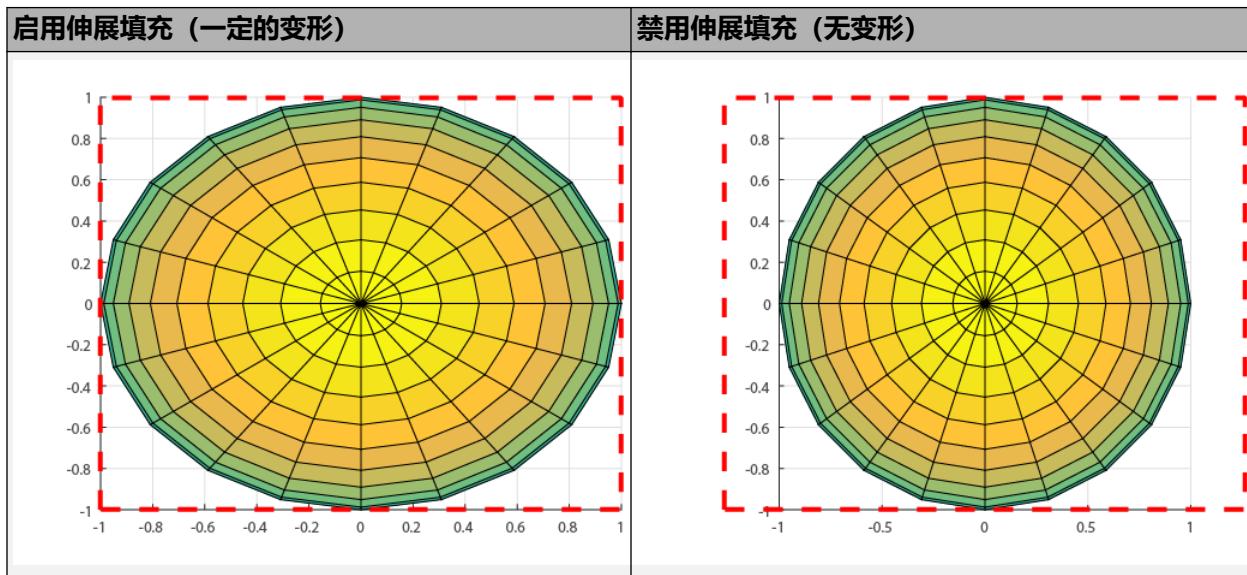


伸展填充行为

默认情况下，MATLAB 会伸展坐标区以填充可用的空间。这种“伸展填充”行为可能会导致部分扭曲。坐标区可能与其 **DataAspectRatio**、**PlotBoxAspectRatio** 和 **CameraViewAngle** 属性中存储的数据纵横比、图框纵横比和相机视角值不完全匹配。Axes 对象的 **DataAspectRatioMode**、**PlotBoxAspectRatioMode** 和 **CameraViewAngleMode** 属性设置为 **'auto'** 时，会启用“伸展填充”行为。

如果您指定数据纵横比、绘图框纵横比或相机视角，则会禁用“伸展填充”行为。如果禁用“伸展填充”行为，MATLAB 会使坐标区在可用空间内尽可能大并严格遵循属性值，不会出现扭曲。

例如，以下图窗分别显示了同一绘图在启用和未启用“伸展填充”行为时的效果。点线显示由 **Position** 属性定义的可用空间。在两种情况中，数据纵横比、绘图框纵横比和相机视角值是相同的。但是，在左侧绘图中，伸展造成了一定的变形。



另请参阅

函数

`axes` | `tiledlayout` | `title` | `daspect` | `pbaspect`

属性

`Axes`

相关示例

- “保存和复制绘图时保留最少的空白” (第 16-24 页)

控制坐标区纵横比

本节内容
“坐标区纵横比属性”（第 9-67 页）
“默认纵横比选择”（第 9-68 页）
“通过调整图窗大小保持坐标区比例”（第 9-69 页）
“纵横比属性”（第 9-71 页）
“显示真实对象”（第 9-75 页）

坐标区纵横比属性

`axis` 命令通过设置各种坐标区对象属性来实现效果。您可以直接设置这些属性，精确实现您想要的效果。

属性	说明
DataAspectRatio	设置各个坐标轴数据值的相对比例。将 <code>DataAspectRatio</code> 设置为 [1 1 1] 可按正确比例显示真实世界的对象。为 <code>DataAspectRatio</code> 指定值将覆盖伸展填充行为。 通过 <code>daspect</code> 进行设置
DataAspectRatioMode	在 <code>auto</code> 模式下，MATLAB 软件将选择能在可用空间提供最高分辨率的轴刻度。
PlotBoxAspectRatio	设置坐标区图框的比例（将 <code>box</code> 设置为 <code>on</code> 可以看到图框）。为 <code>PlotBoxAspectRatio</code> 指定值将覆盖伸展填充行为。 通过 <code>pbaspect</code> 进行设置
PlotBoxAspectRatioMode	在 <code>auto</code> 模式下，MATLAB 将 <code>PlotBoxAspectRatio</code> 设置为 [1 1 1]，除非您显式设置 <code>DataAspectRatio</code> 和/或坐标轴范围。
Position	通过四元素向量（[左偏移、下偏移、宽、高]）定义坐标区的位置和大小。
XLim, YLim, ZLim	设置各个轴的最小范围和最大范围。
XLimMode , YLimMode , ZLimMode	在 <code>auto</code> 模式下，MATLAB 会选择坐标轴范围。

当模式属性设置为 `auto` 时，MATLAB 会自动确定所有这些属性的值，然后伸展坐标区以适合图窗形状。通过指定属性值或将模式属性设置为手动，可以覆盖任何属性的自动操作。

您为特定属性选择的值主要取决于要显示的数据类型。通过 MATLAB 可视化的大部分数据都属于以下两种情况之一：

- 以线图、网格图或其他专业绘图方式显示的数值数据
- 真实世界对象（例如，机动车辆或地球地形剖面图）的表示

在第一种情况下，通常需要选择能够在每个轴方向提供清晰分辨率并能填满可用空间的坐标轴范围。但是，对于真实世界对象，则需要按比例精确表示，而不管视角如何。

MATLAB 的默认属性值用于

- 选择坐标轴范围以涵盖数据范围（当 `XLimMode`、`YLimMode`、`ZLimMode` 设置为 `auto` 时）。

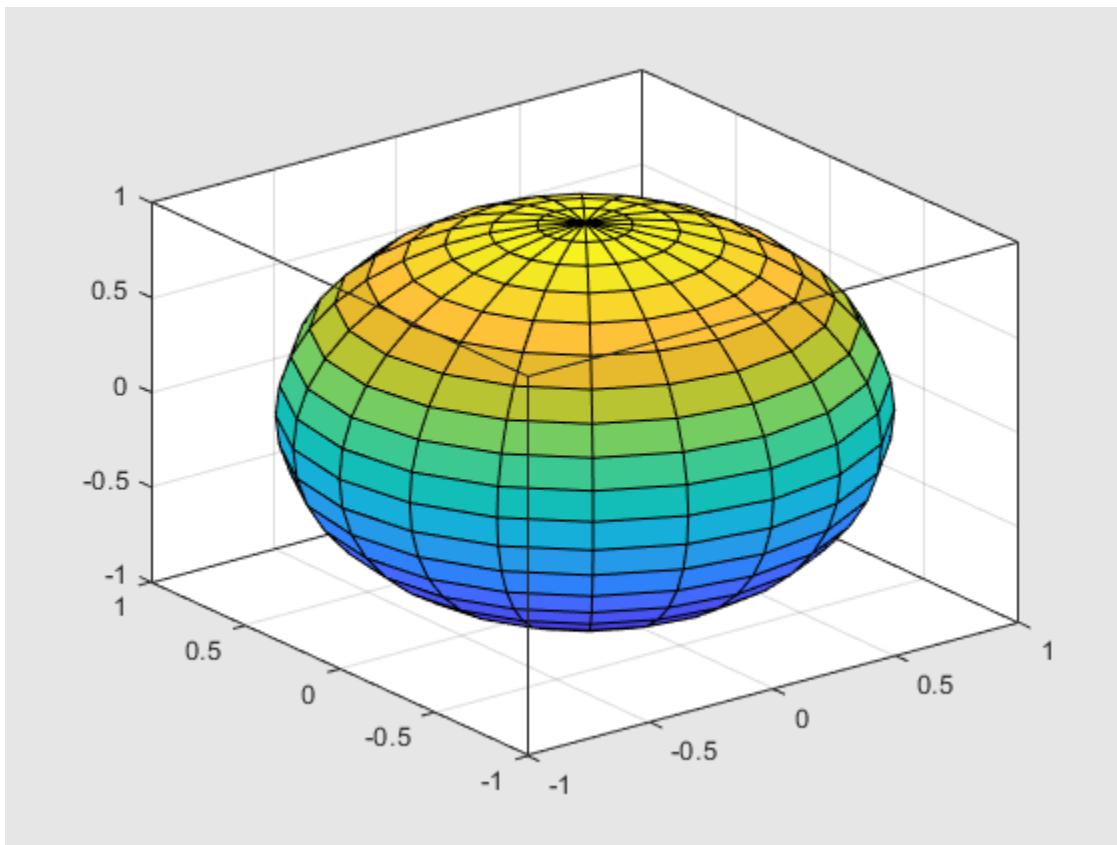
- 通过单独设置每个轴的标度，在可用空间提供最高分辨率（当 `DataAspectRatioMode` 和 `PlotBoxAspectRatioMode` 设置为 `auto` 时）。
- 通过调整 `CameraViewAngle` 来绘制适合位置矩形的坐标区，然后根据需要对坐标区进行伸展填充。

默认纵横比选择

坐标区属性 `Position` 指定坐标区在图窗中的位置和尺寸。`Position` 向量的第三个和第四个元素（宽度和高度）定义了一个矩形，MATLAB 将在其中绘制坐标区。MATLAB 将调整坐标区以适合该矩形。

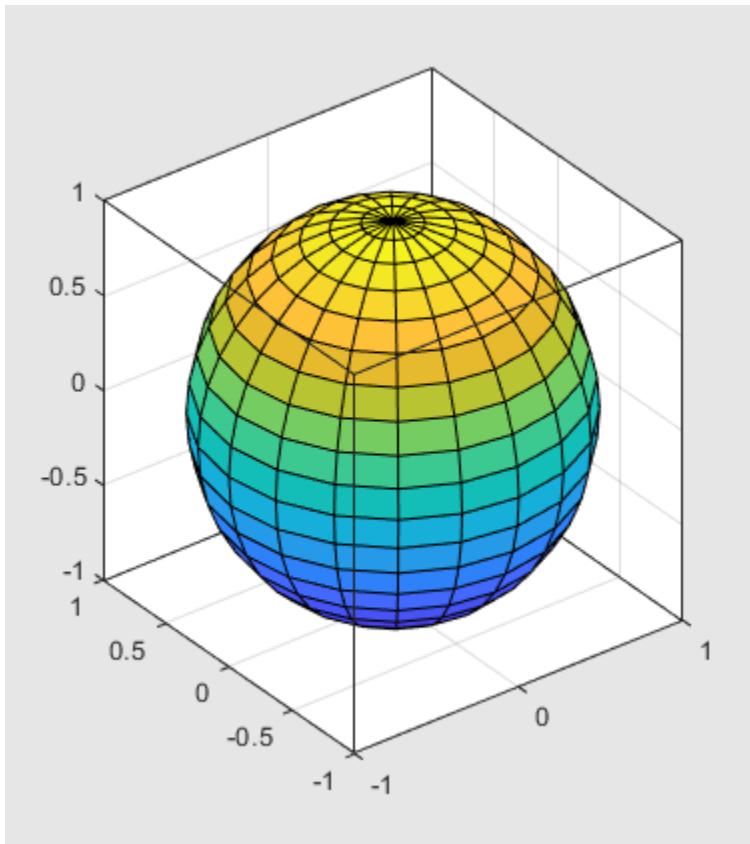
坐标区属性 `Units` 的默认值根据父图窗尺寸进行归一化。这意味着图窗窗口的形状决定位置矩形的形状。当您更改图窗窗口的大小时，MATLAB 会调整位置矩形的形状以适合图窗。

```
sphere
set(gcf,'Color',[0.90 0.90 0.90])
set(gca,'BoxStyle','full','Box','on')
```



更改图窗的大小和形状将导致坐标区的大小和形状发生变化。坐标区也可以选择新的轴刻度线位置。

```
f = gcf;
f.Position(3) = f.Position(3) * 0.67;
```



重新调整坐标区的形状以适合图窗窗口可能会更改图形的纵横比。MATLAB 将调整坐标区以填满位置矩形，在此过程中可能会导致形状变形。这通常适用于数值数据图形，但不适用于真实显示的对象。

通过调整图窗大小保持坐标区比例

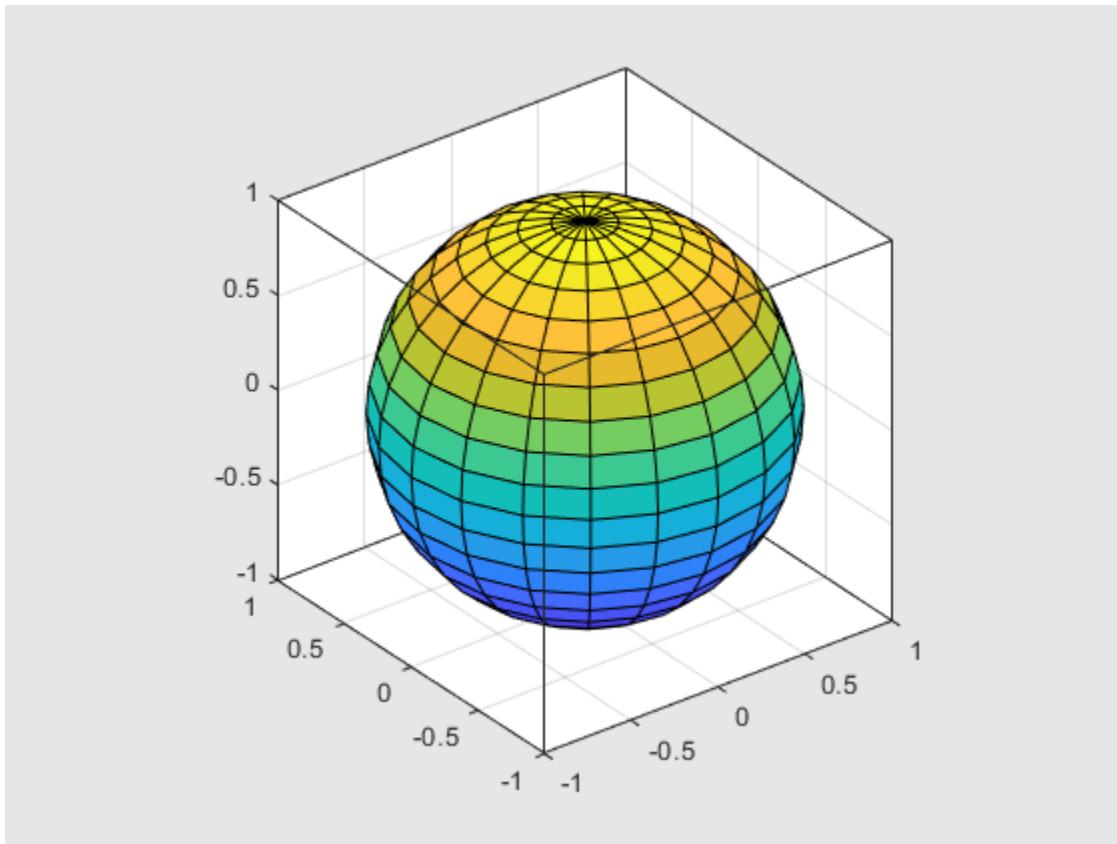
要保持特定的形状，您可以用独立于图窗窗口大小的绝对单位（如英寸）指定坐标区大小。但是，如果您要编写能够适合任意图窗窗口大小的 MATLAB 程序，这并不是一个好方法。更好的做法是指定坐标区的纵横比并覆盖自动伸展填充。

如果您需要特定的纵横比，可以通过指定以下坐标区属性值来覆盖伸展填充：

- **DataAspectRatio** 或 **DataAspectRatioMode**
- **PlotBoxAspectRatio** 或 **PlotBoxAspectRatioMode**
- **CameraViewAngle** 或 **CameraViewAngleMode**

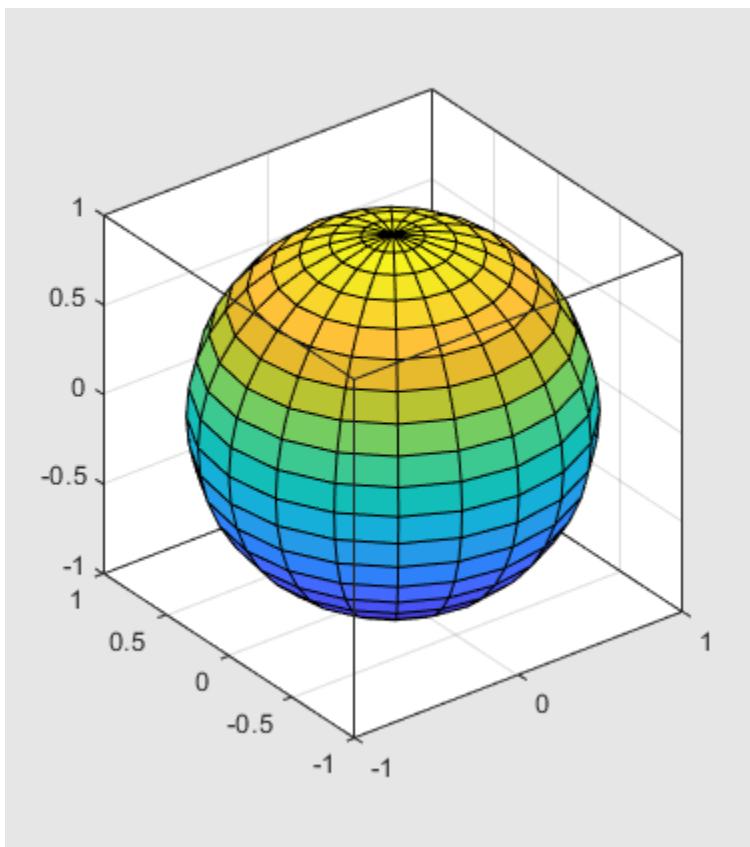
前两组属性直接影响纵横比。将任一模式属性设置为 `manual` 只会禁用伸展填充，而当前所有属性值都保持不变。在这种情况下，MATLAB 将放大坐标区，直到受限于位置矩形的一个维度为止。例如，将 **DataAspectRatio** 设置为 `[1 1 1]`。此外，设置图窗颜色，以查看图窗与坐标区之间的关系。

```
sphere
daspect([1 1 1])
set(gca,'BoxStyle','full','Box','on')
set(gcf,'Color',[0.90 0.90 0.90])
```



更改图窗的大小和形状不会改变坐标区的纵横比。

```
f = gcf;
f.Position(3) = f.Position(3) * 0.67;
```



设置 **CameraViewAngle** 属性将禁用伸展填充，还可以防止 MATLAB 在您更改视图时重新调整坐标区的大小。

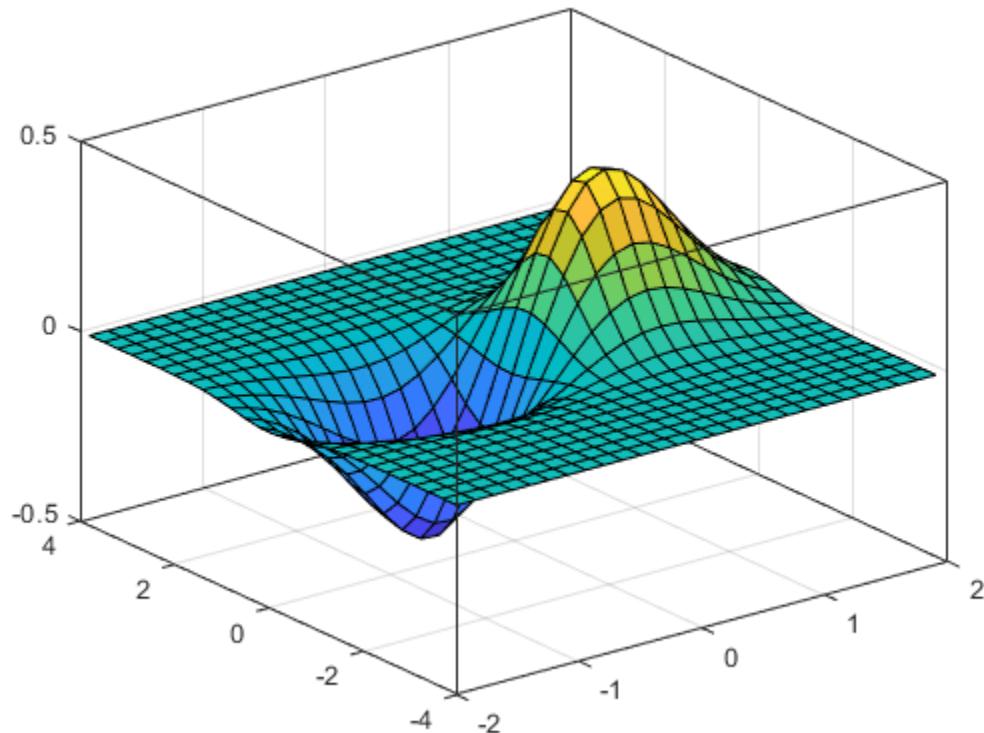
纵横比属性

了解属性之间如何相互影响很重要，这样才能获得您想要的结果。**DataAspectRatio**、**PlotBoxAspectRatio** 以及 x、y 和 z 坐标轴范围 (**XLim**、**YLim** 和 **ZLim** 属性) 都会限制坐标区的形状。

数据纵横比

DataAspectRatio 属性控制轴刻度的比例。例如，要显示数学表达式的曲面图，MATLAB 会选择突出显示函数值的数据纵横比：

```
[X,Y] = meshgrid((-2:.15:2),(-4:.3:4));
Z = X.*exp(-X.^2 - Y.^2);
surf(X,Y,Z)
set(gca,'BoxStyle','full','Box','on')
```



daspect 函数返回 **DataAspectRatio** 属性的实际值。

daspect

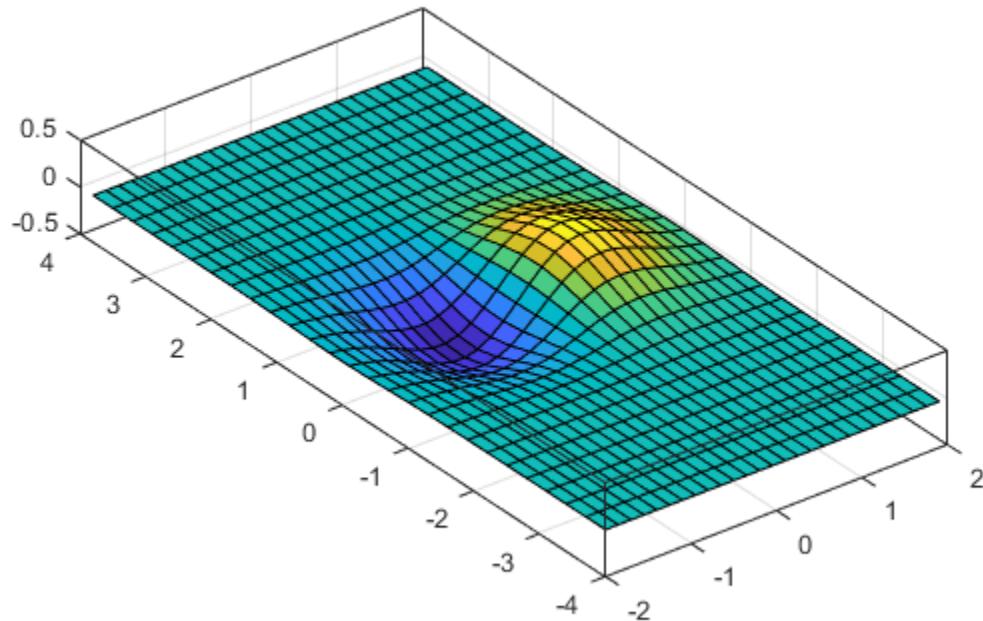
ans = 1×3

4 8 1

这意味着 x 轴上四个单位的长度覆盖的数据值范围与 y 轴上八个单位以及 z 轴上一个单位覆盖的数据值范围相同。坐标区填充的图框的默认纵横比为 [1 1 1]。

如果要查看曲面图以使每个轴上的相对大小彼此相等，可将 **DataAspectRatio** 设置为 [1 1 1]。

daspect([1 1 1])



设置 **DataAspectRatio** 属性值还会将 **DataAspectRatioMode** 设置为 **manual** 并覆盖伸展填充，以实现指定的纵横比。

图框纵横比

观察上一节中图的 **PlotBoxAspectRatio** 值，可以看到它现在采用了之前的 **DataAspectRatio** 值。**pbaspect** 函数返回 **PlotBoxAspectRatio** 的值：

```
pbaspect
```

```
ans = 1×3
```

```
4 8 1
```

请注意，MATLAB 使用指定的 **DataAspectRatio** 重新缩放图框以适应该图形。

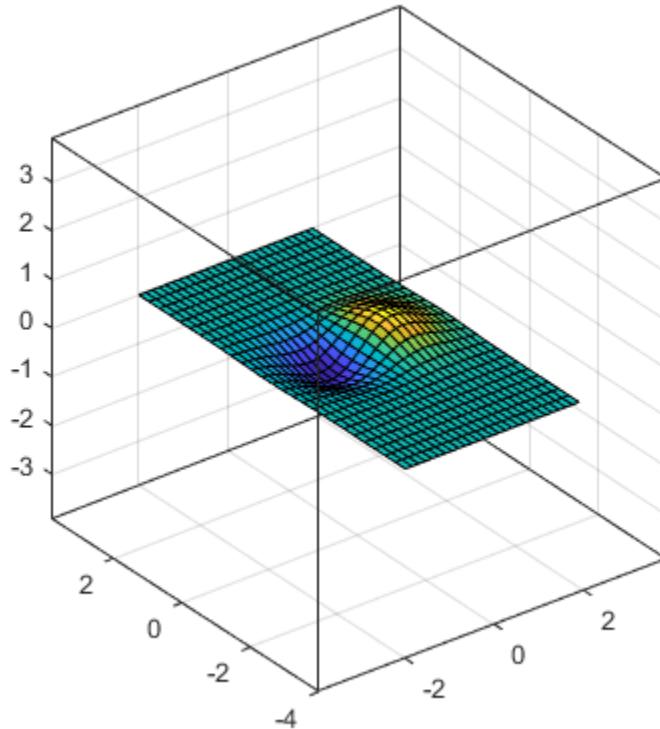
PlotBoxAspectRatio 属性控制坐标区图框的形状。默认情况下，MATLAB 将此属性设置为 [1 1 1] 并调整 **DataAspectRatio** 属性，以使图形填满图框或直至受到约束为止。

如果您设置了 **DataAspectRatio** 值从而防止它改变，则 MATLAB 会更改 **PlotBoxAspectRatio**。

如果您同时指定 **DataAspectRatio** 和 **PlotBoxAspectRatio**，MATLAB 将被迫更改坐标轴范围，以遵守您已定义的两个约束。

继续网格示例，如果您同时设置了这两个属性，MATLAB 将更改坐标轴范围，以满足施加于坐标区的两个约束。

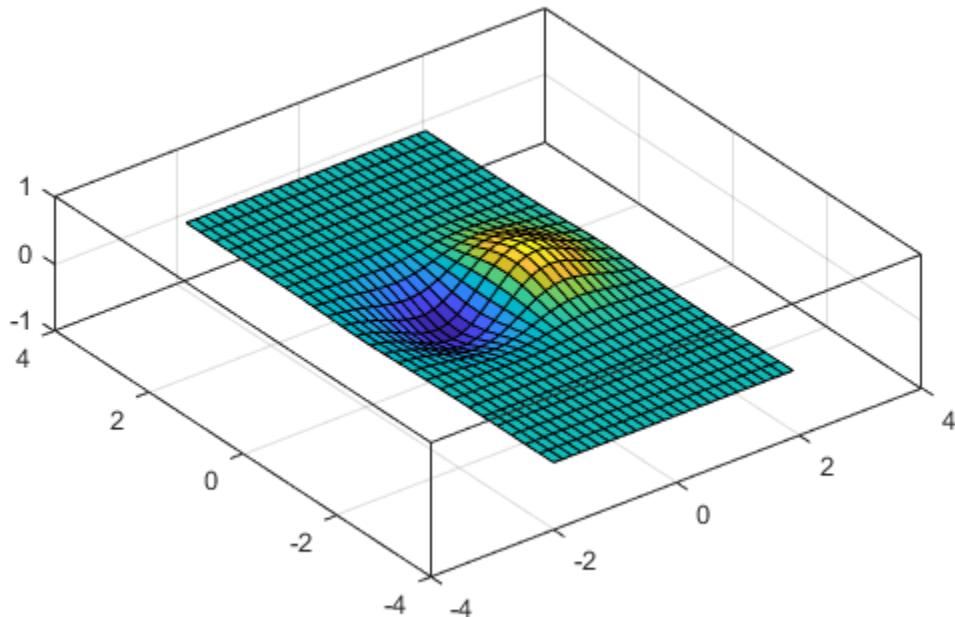
```
daspect([1 1 1])
pbaspect([1 1 1])
```



调整坐标轴范围

坐标区还具有用于设置 x 轴、y 轴和 z 轴范围的属性。但是，使用 **PlotBoxAspectRatio** 和 **DataAspectRatio** 属性指定坐标轴范围会过度约束坐标区。例如，以下命令指定的坐标轴范围将会与 **PlotBoxAspectRatio** 值相冲突。

```
set(gca,'DataAspectRatio',[1 1 1],...
    'PlotBoxAspectRatio',[1 1 1],...
    'XLim',[-4 4],...
    'YLim',[-4 4],...
    'ZLim',[-1 1])
```



如果查询绘图框的纵横比，您会发现 **PlotBoxAspectRatio** 值已更改，以适应坐标轴范围。

```
pbaspect
```

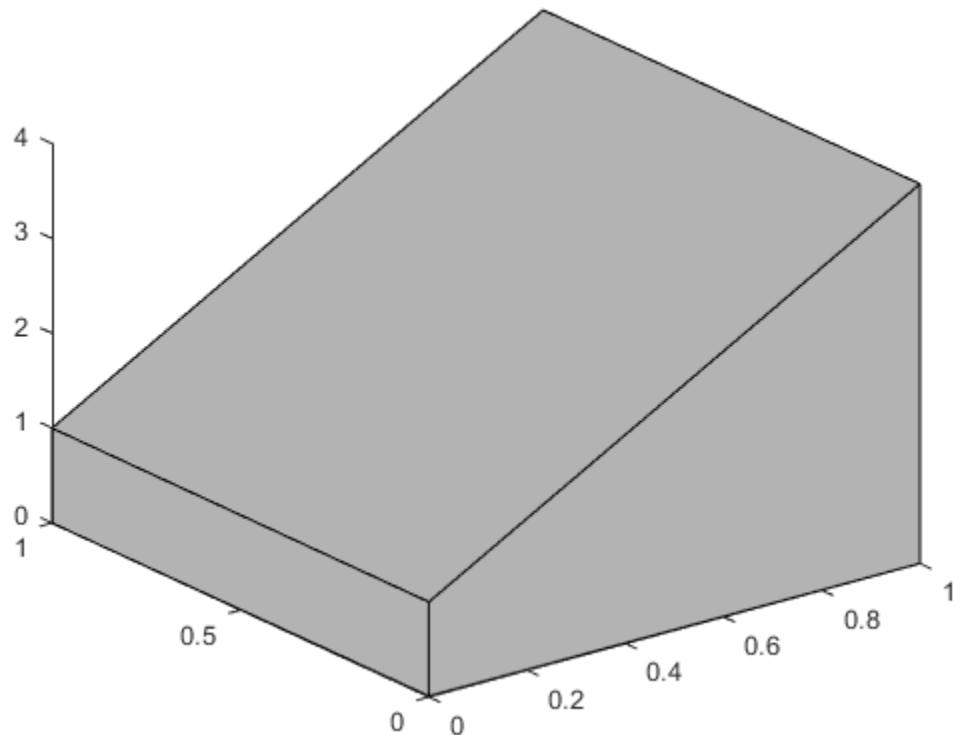
```
ans = 1×3
```

```
4    4    1
```

显示真实对象

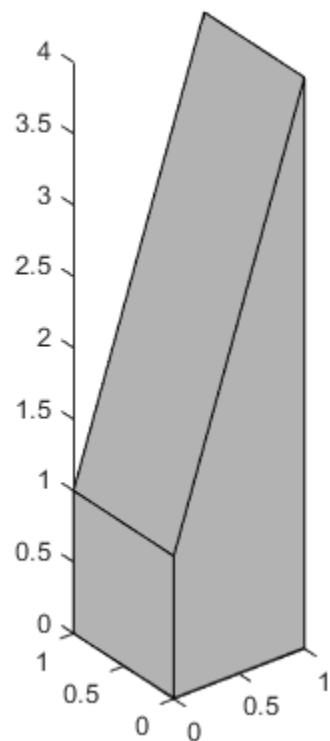
如果要使显示的对象看起来很逼真，您需要更改 MATLAB 的默认值。例如，以下数据定义一个楔形补片对象。

```
vert = [0 0 0; 0 1 0; 1 1 0; 1 0 0; 0 0 1; 0 1 1; 1 1 4; 1 0 4];
fac = [1 2 3 4; 2 6 7 3; 4 3 7 8; 1 5 8 4; 1 2 6 5; 5 6 7 8];
patch('Vertices',vert,'Faces',fac,...
'FaceColor',[0.7 0.7 0.7],'EdgeColor','k')
view(3)
```



但是，这个坐标区使这些数据定义的实体对象的实际形状变得失真。要按正确的比例显示它，需要设置 **DataAspectRatio**。设置此属性将使 x、y 和 z 方向上的单位相等，并且会阻止坐标区伸展以填充位置矩形，从而显示对象的真实形状。

```
set(gca,'DataAspectRatio',[1 1 1])
```



控制绘图函数如何选择颜色和线型

本节内容

- “自动分配的工作原理” (第 9-78 页)
- “更改颜色方案和线型” (第 9-79 页)
- “更改 ColorOrder 和 LineStyleOrder 数组的索引” (第 9-81 页)

当您在同一坐标区内绘制多个数据集时，MATLAB 会自动为绘图对象分配不同的颜色（可能还有不同的线型和标记）。您可以在调用绘图函数时自定义颜色、线型和标记，还可以在调用函数后设置属性。

例如，绘制一条红色实线和一条绿色虚线。然后在红线上添加方形标记，在绿线上添加圆形标记。

```
p1 = plot([0 1 2 3],'-r');
hold on
p2 = plot([1 2 3 4],'-g');
hold off

% Add markers
p1.Marker = 'sq';
p2.Marker = 'o';
```

“Specify Plot Colors” 中说明了这种方法。要自定义几个绘图的特征，此方法很有用。但是，它在其他情况下的灵活性较差，例如在循环中绘制数据图或将矩阵数据传递给绘图函数时。在这类情况下，可以更改用于控制 MATLAB 如何自动分配颜色、线型和标记的属性。

注意 以下示例中的有些功能从 R2019b 开始提供，有些功能从 R2020a 开始提供。要在较早的版本中修改绘图颜色和线型，请参阅为何绘图线条有不同颜色？和用于绘图的线型 - LineStyleOrder。

自动分配的工作原理

MATLAB 通过循环选择坐标区属性 **ColorOrder** 中所列的颜色，为绘图对象（例如 **Line**、**Scatter** 和 **Bar** 对象）分配颜色。**ColorOrder** 属性包含一个 RGB 三元组数组，其中每个 RGB 三元组定义了一种颜色。默认 **ColorOrder** 数组包含七种颜色。如果您创建的对象数量多于颜色数量，则颜色将会重复。

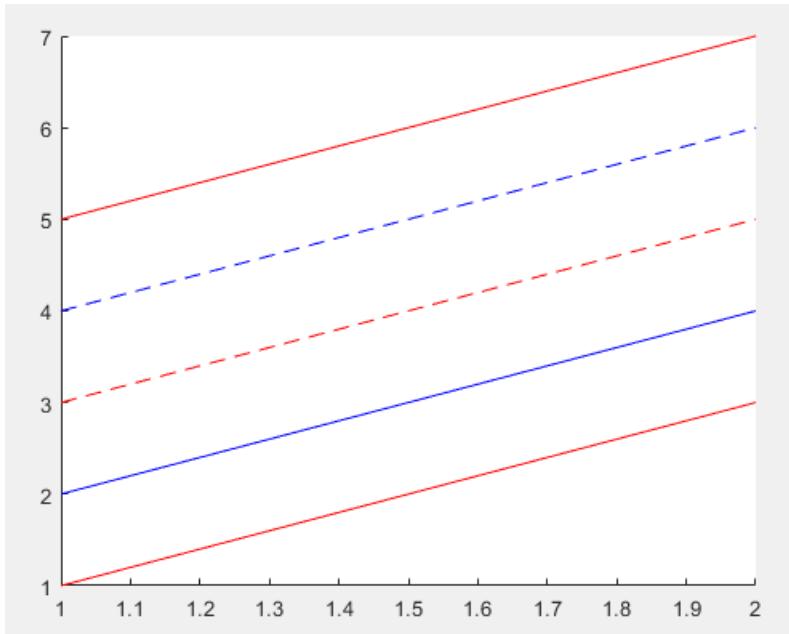
如果绘图对象支持线型和标记，MATLAB 还会循环使用坐标区属性 **LineStyleOrder** 的列表中的线型。**LineStyleOrder** 属性包含一个字符序列元胞数组，其中每个字符序列对应一个线型（或与标记结合的线型）。默认 **LineStyleOrder** 数组仅包含实线线型 ('-')。**ColorOrder** 数组中的所有颜色将与 **LineStyleOrder** 数组中的一个字符序列一起使用，然后再使用下一个序列。MATLAB 对每个新的绘图对象继续使用该循环。如果对象数量多于颜色和字符序列的组合数，则重复该循环。

对于给定的 **ColorOrder** 和 **LineStyleOrder** 数组对，特定绘图对象的颜色、线型和标记取决于对象的 **SeriesIndex** 属性值，该属性是从 R2020a 开始提供的一个新属性。默认情况下，**SeriesIndex** 属性是一个与对象的创建顺序对应的数字，从 1 开始。MATLAB 使用该数字计算 **ColorOrder** 和 **LineStyleOrder** 数组的索引。

例如，使用 **ColorOrder** 数组中的两种颜色（红色和蓝色）以及 **LineStyleOrder** 数组中的两种线型（实线和虚线）创建一个坐标区对象。然后绘制五条线。

```
ax = axes;
ax.ColorOrder = [1 0 0; 0 0 1];
ax.LineStyleOrder = {'-', '--'};
```

```
hold on
for i = 1:5
    plot([i i+2])
end
hold off
```



下表列出了上图中每条线的 SeriesIndex、在 ColorOrder 数组中的索引和在 LineStyleOrder 数组中的索引。

	SeriesIndex	在 ColorOrder 数组中的索引	在 LineStyleOrder 数组中的索引	线外观
第一条线	1	1	1	红色实线
第二条线	2	2	1	蓝色实线
第三条线	3	1	2	红色虚线
第四条线	4	2	2	蓝色虚线
第五条线	5	1	1	红色实线

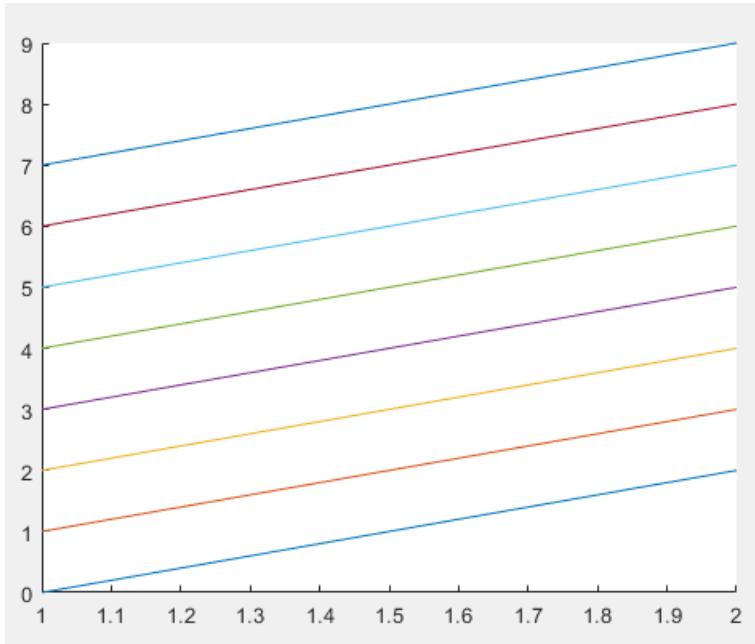
您可以通过修改坐标区的 ColorOrder 或 LineStyleOrder 属性，或通过更改绘图对象的 SeriesIndex 属性，来更改绘图对象的颜色、线型和标记。

更改颜色方案和线型

更改坐标区的 ColorOrder 属性会更改绘图的颜色方案。更改坐标区的 LineStyleOrder 属性会更改绘图中使用的线型（如果使用了标记，则还会更改标记）。例如，使用默认颜色和线型在一个循环中绘制 8 条线。

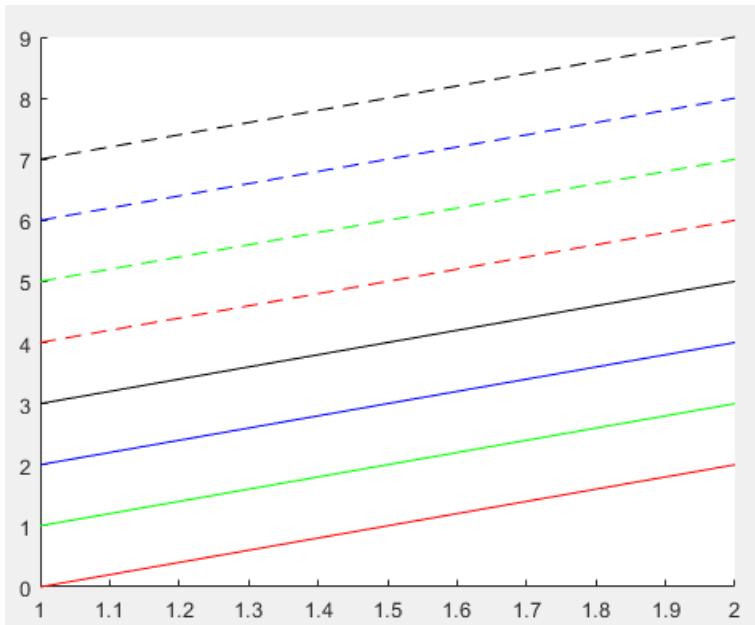
```
ax = axes;
hold on
```

```
for i = 0:7
    plot([i i+2])
end
hold off
```



将 ColorOrder 数组替换为一个包含四种颜色的新数组（也可以使用 colororder 函数替换此数组）。然后将 LineStyleOrder 数组替换为一个包含两种线型的新元胞数组。线条将自动使用新的颜色和线型。

```
% Updates existing plots in R2019b or later
ax.ColorOrder = [1 0 0; 0 1 0; 0 0 1; 0 0 0];
ax.LineStyleOrder = {[1,1],[1,2]};
```

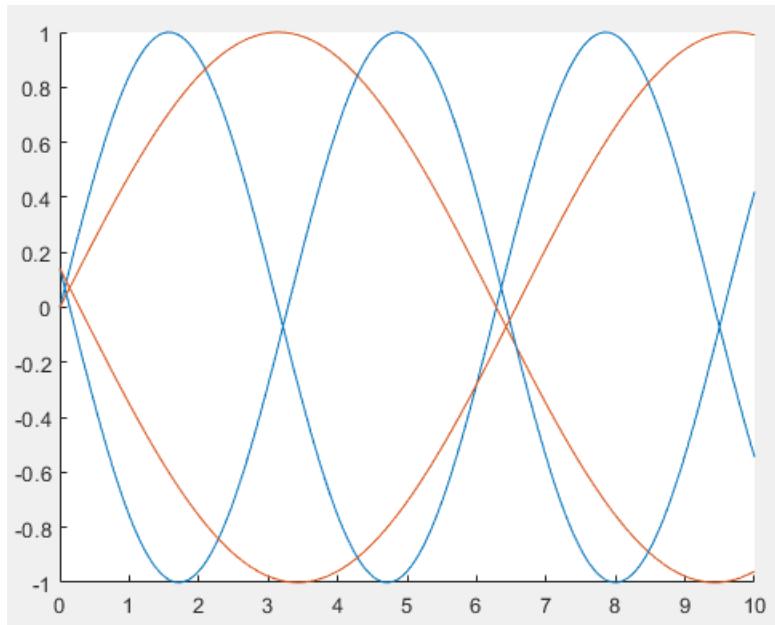


更改 ColorOrder 和 LineStyleOrder 数组的索引

更改绘图对象的 SeriesIndex 属性将会更改 ColorOrder 和 LineStyleOrder 数组的索引。当您希望某个对象的颜色、线型和标记与另一个对象匹配时，更改索引非常有用。

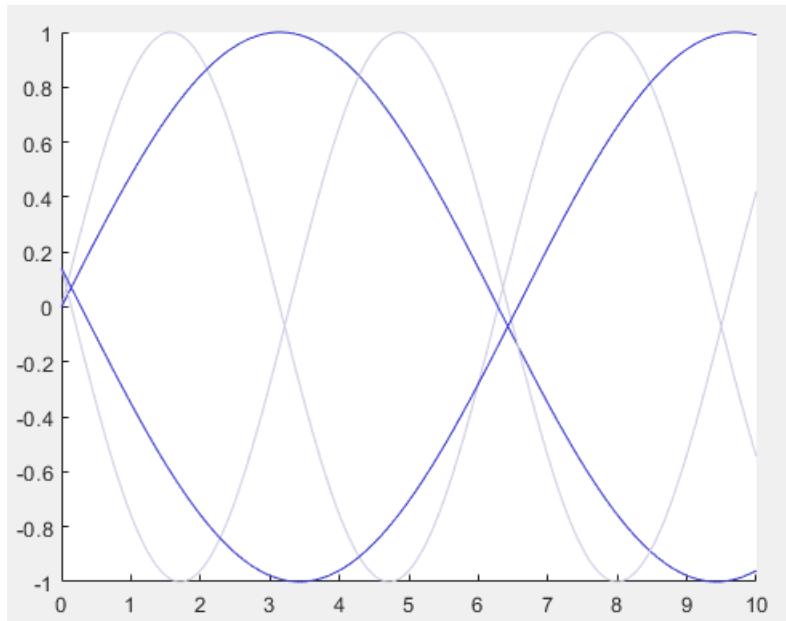
例如，在一个循环中绘制四个具有不同波长和相位的正弦波。对于每个正弦波，根据波长设置 SeriesIndex 属性。在生成的绘图中，具有相同波长的正弦波也具有相同的颜色。

```
x = linspace(0,10,200);
ax = axes;
hold on
for phi = 0:3:3
    for t = 1:2
        plot(x,sin(x/t + phi),'SeriesIndex',t) % Requires R2020a or later
    end
end
hold off
```



要使其中一对正弦波更加突出，请将色序更改为不同的颜色集。

```
ax.ColorOrder = [0.8 0.8 0.9; 0.2 0.2 0.8];
```



另请参阅

函数

[plot](#) | [gca](#) | [colororder](#)

属性

[Axes](#)

详细信息

- “Specify Plot Colors”

在绘图和图表中裁剪

以下示例演示 MATLAB® 如何在绘图中使用裁剪和如何控制裁剪。

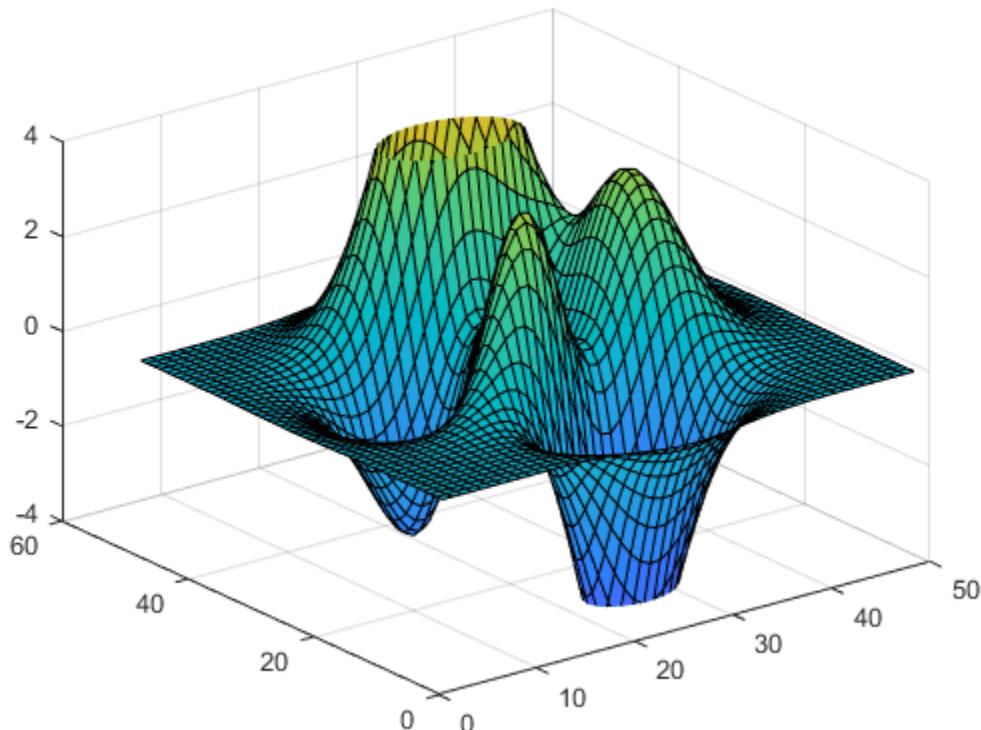
什么是裁剪？

当绘图的一部分超出坐标区的边界时即会出现裁剪的情况。在 MATLAB® 中，被裁剪的绘图部分不会显示在屏幕上或打印输出中。边界由绘图的坐标轴范围确定。

关闭裁剪

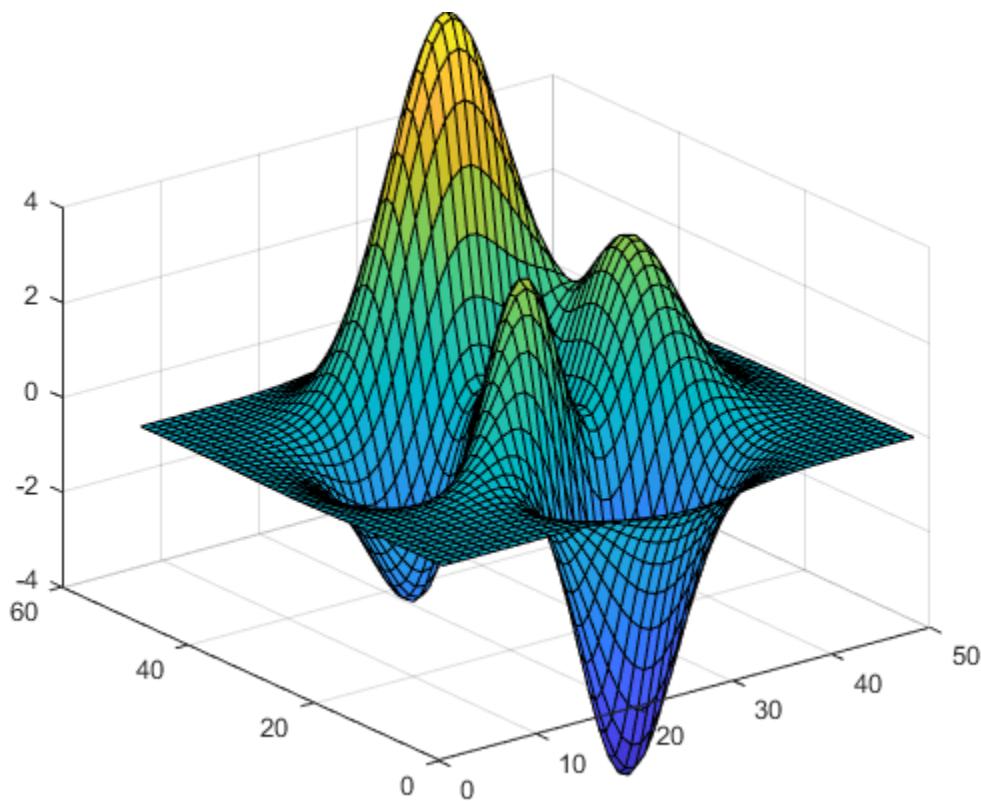
默认情况下，MATLAB 会裁剪掉超出坐标区范围的绘图。

```
figure  
surf(peaks)  
zlim([-4 4])
```



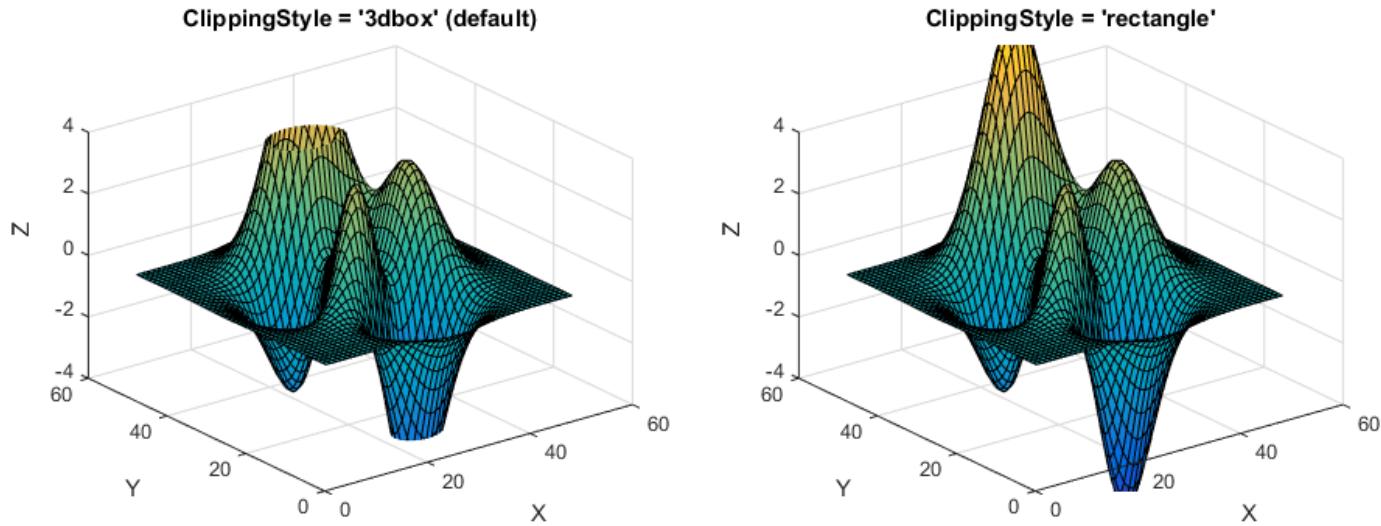
使用坐标区的 **Clipping** 属性控制裁剪行为。

```
ax = gca; % get the current axis  
ax.Clipping = 'off'; % turn clipping off
```



控制裁剪样式

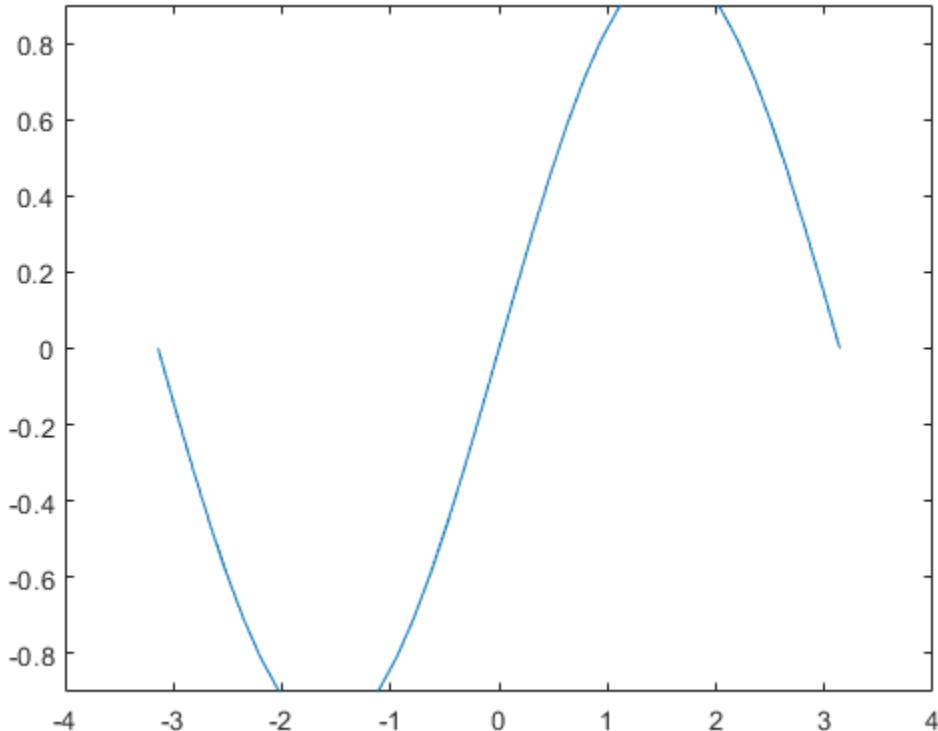
使用 `ClippingStyle` 属性控制裁剪的工作方式。如果 `ClippingStyle` 设置为 '`3dbox`'，则 MATLAB 会将绘图裁剪为由 x、y 和 z 轴范围定义的空间体。如果 `ClippingStyle` 设置为 '`rectangle`'，则 MATLAB 会将绘图裁剪为一个围绕 x、y 和 z 轴外围而成的虚构矩形。以下绘图显示两种裁剪样式之间的差异。



在二维绘图中裁剪

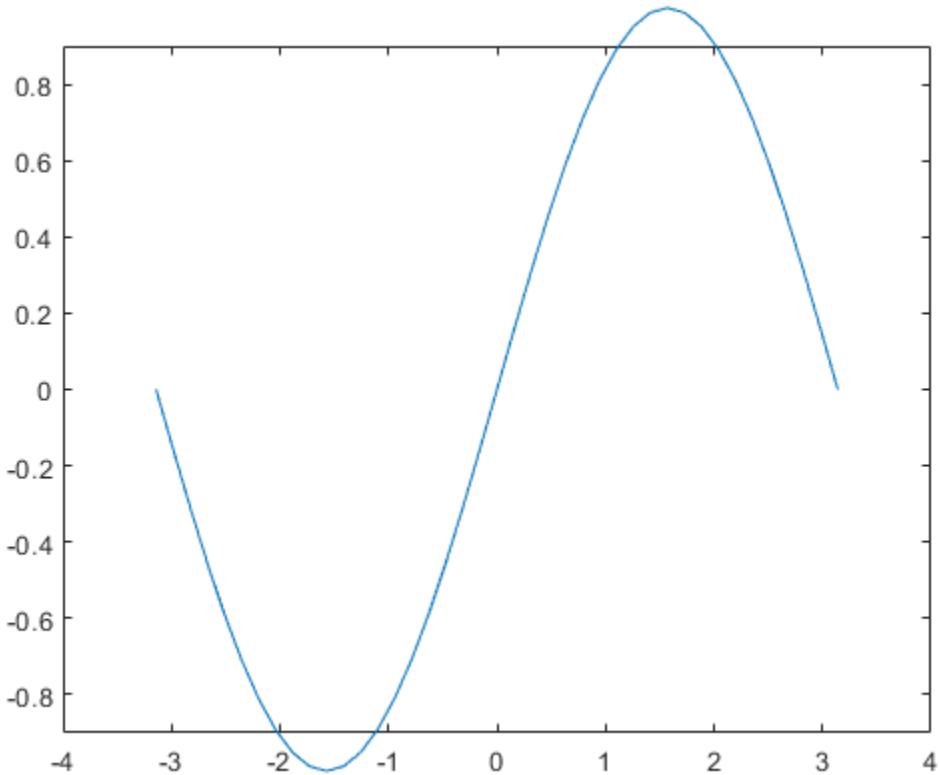
在二维绘图中也可以使用裁剪。例如，MATLAB 会裁剪以下绘图中的正弦波。

```
x = -pi:pi/20:pi;
y = sin(-pi:pi/20:pi);
plot(x,y)
ylim([-0.9 0.9])
```



如果关闭裁剪，则 MATLAB 显示整个正弦波。

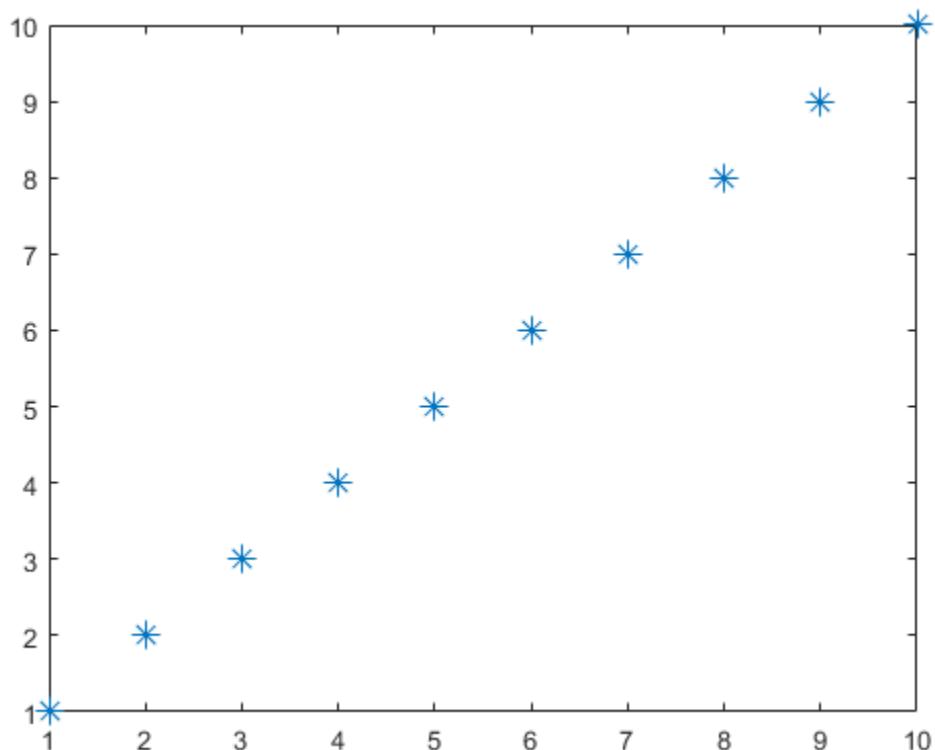
```
ax = gca;
ax.Clipping = 'off';
```



裁剪和标记

只要每个数据点本身在绘图的 x 和 y 坐标轴范围内，裁剪就不影响在每个数据点上绘制的标记。MATLAB 会显示整个标记，即便它稍微超出坐标区的边界，也会显示出来。

```
p = plot(1:10, '*');
p.MarkerSize = 10;
axis([1 10 1 10])
```



使用图形平滑处理

此示例说明如何在 MATLAB 绘图中使用图形和字体平滑处理。

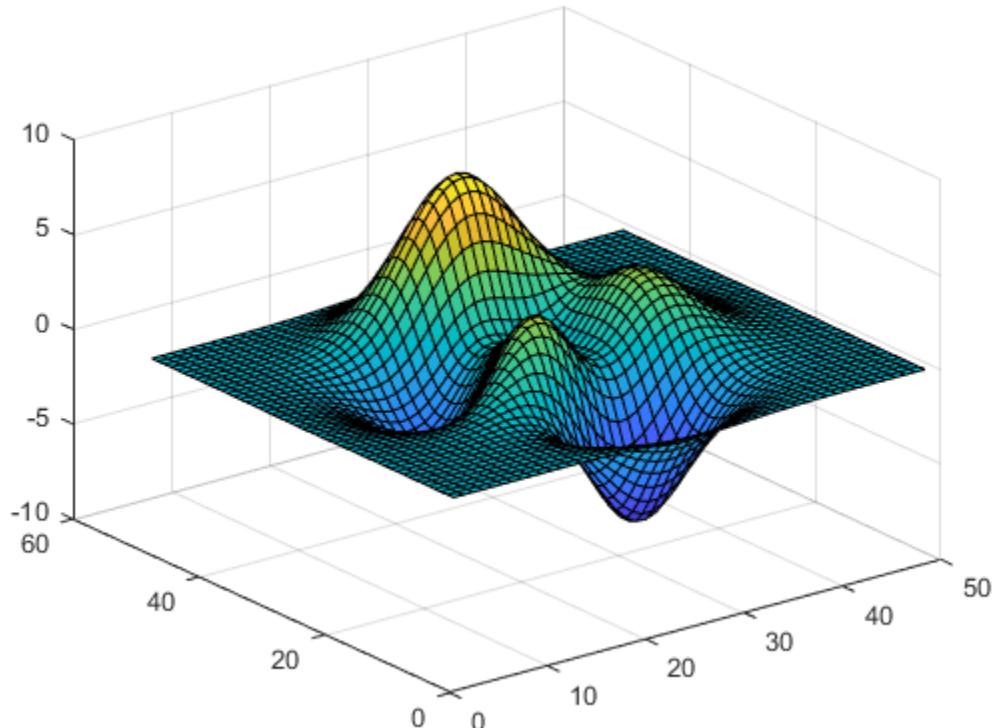
什么是图形平滑处理？

图形平滑处理可改进绘图中图形的外观。平滑处理会消除使用像素或点表示连续对象时产生的锯齿状边缘。用于图形平滑处理的技术包括多采样处理和消除锯齿。

图窗中的图形平滑处理

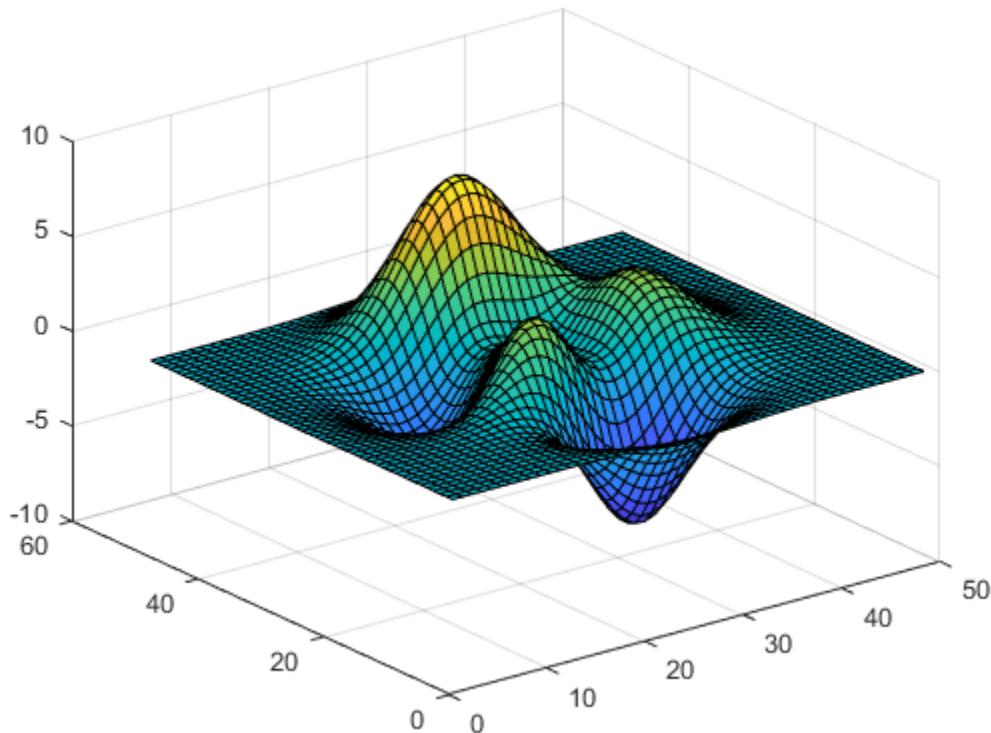
通过使用 **GraphicsSmoothing** 属性在图窗中控制图形平滑处理。默认情况下，**GraphicsSmoothing** 属性设置为 'on'。

```
f = figure;
surf(peaks)
```



您可以通过将 **GraphicsSmoothing** 属性设置为 'off' 以关闭图形平滑处理。

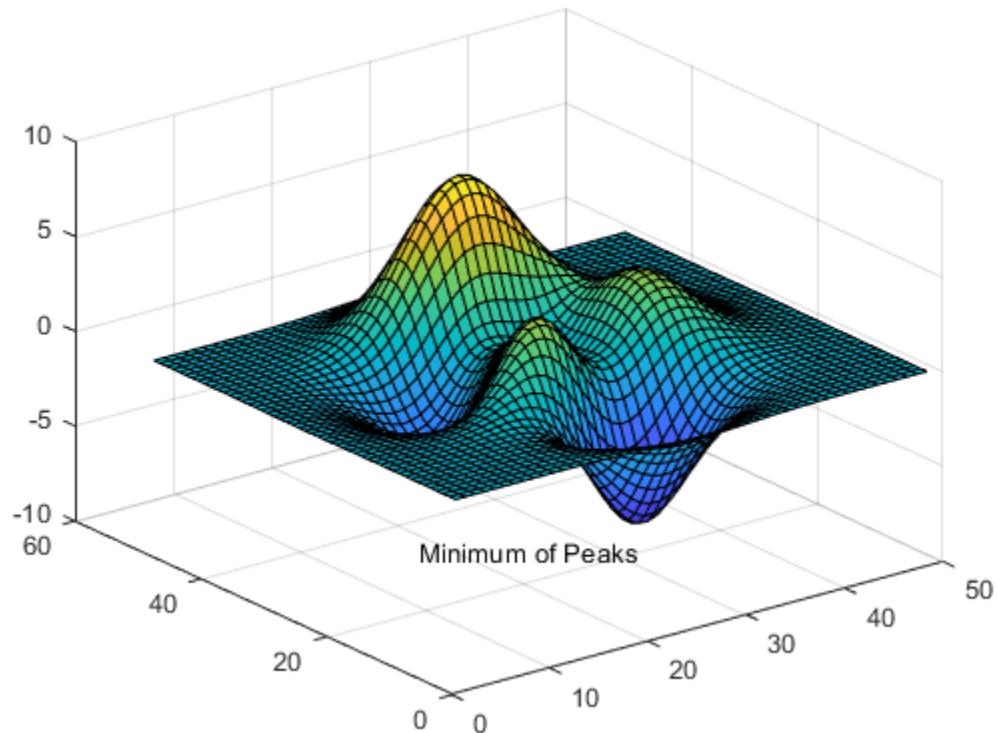
```
f.GraphicsSmoothing = 'off';
```



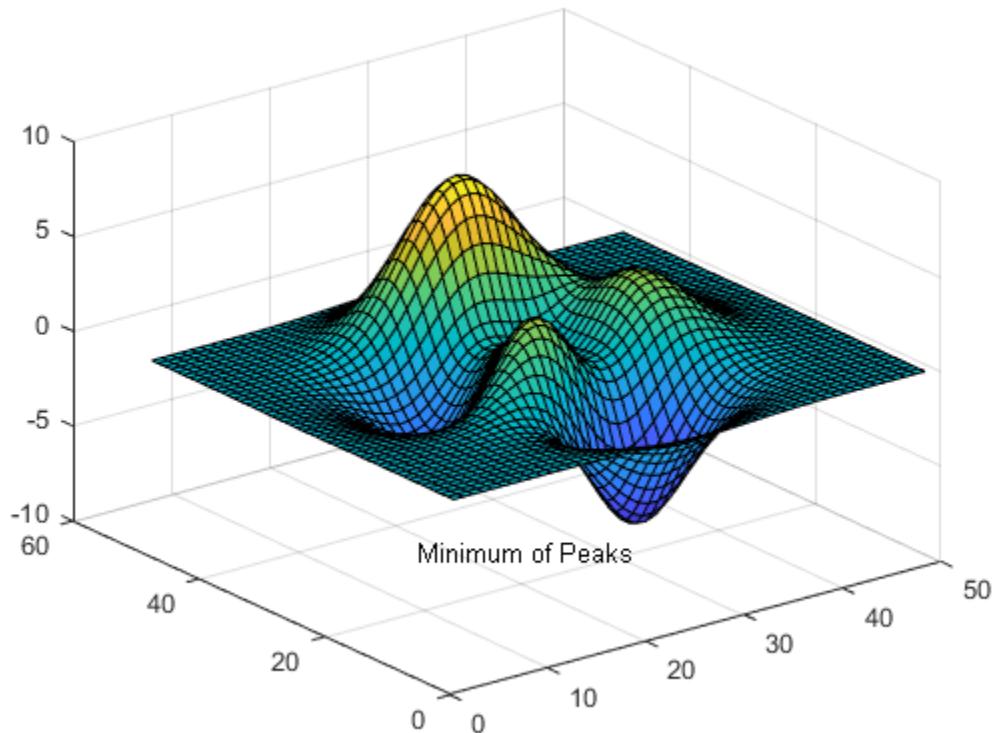
文本和坐标区对象的字体平滑处理

文本和坐标区对象的 **FontSmoothing** 属性控制文本的渲染方式。当 **FontSmoothing** 属性设置为 'on' 时，文本将以平滑的边缘绘制。默认情况下，字体平滑处理为 'on'。

```
t = text(14,27,-8.5, 'Minimum of Peaks');
```



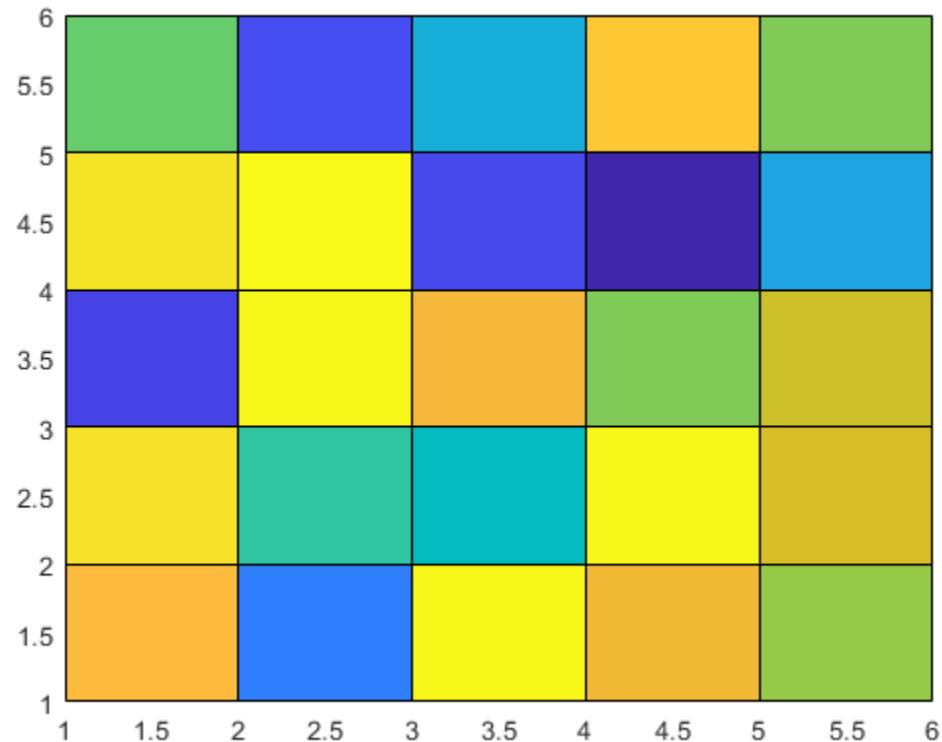
```
t.FontSmoothing = 'off';
```



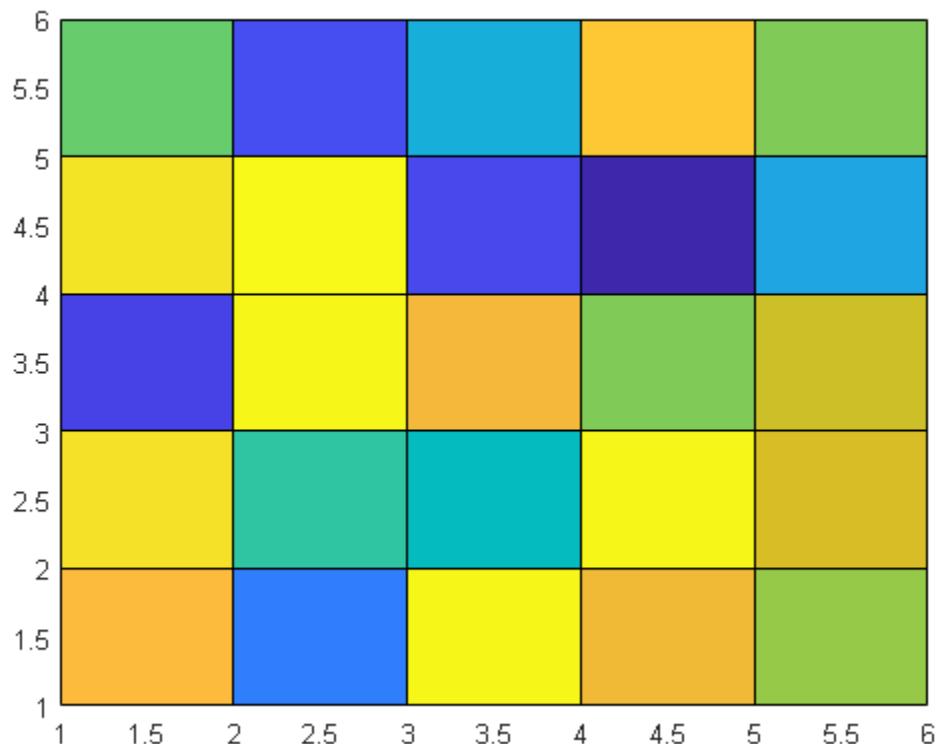
为何关闭图形平滑处理?

如果不做图形平滑处理，水平和垂直线条将显示得更加锐利。当图形平滑处理关闭时，某些图表类型可能显示效果更好。同样，关闭字体平滑处理后，使用小字体的文本看起来会更加清晰。

```
pcolor(rand(6))
```



```
ax = gca;           % get current axes  
ax.FontSmoothing = 'off';    % turn off axes font smoothing
```



```
f.GraphicsSmoothing = 'off'; % turn off figure graphics smoothing
```

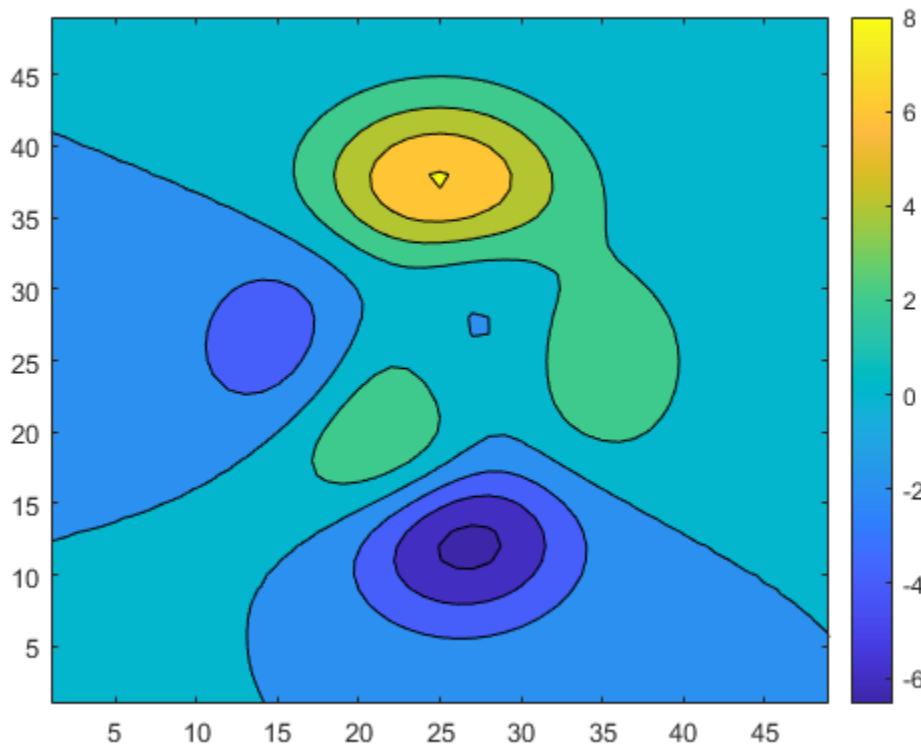

标记图形颜色

- “创建颜色栏” (第 10-2 页)
- “使用颜色图更改颜色方案” (第 10-10 页)
- “曲面绘图数据与颜色图的关系” (第 10-16 页)
- “图像数据与颜色图的关系” (第 10-21 页)
- “补片数据与颜色图的关系” (第 10-26 页)
- “控制颜色图范围” (第 10-34 页)
- “颜色图和真彩色之间的差异” (第 10-38 页)

创建颜色栏

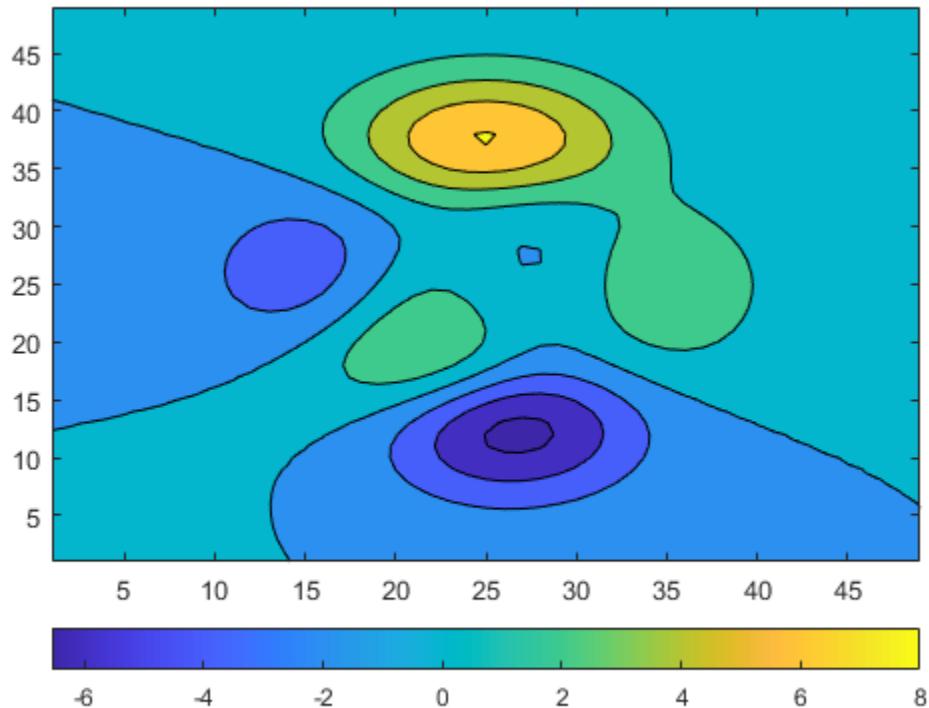
通过颜色栏可以查看数据与图形中所示颜色之间的关系。创建颜色栏后，可以自定义外观的不同方面，例如位置、厚度和刻度标签。例如，此颜色栏显示 `peaks` 函数的值与其旁边图中所示颜色之间的关系。

```
contourf(peaks)  
c = colorbar;
```



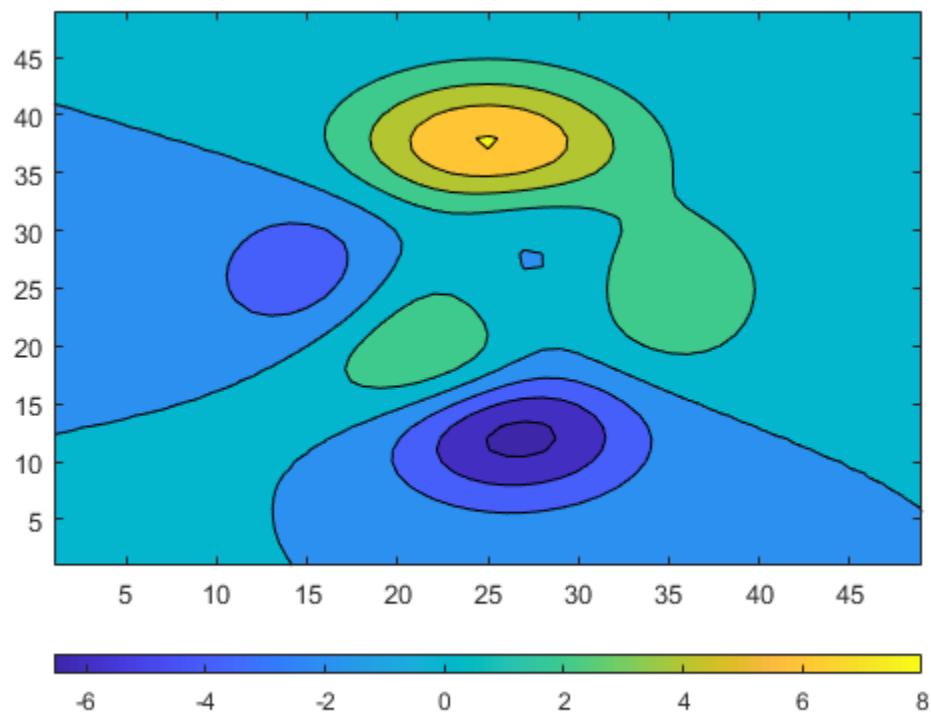
颜色栏的默认位置为坐标区右侧。但是，您可以通过设置 `Location` 属性将颜色栏移至不同位置。本例中使用 `'southoutside'` 选项将颜色栏放在坐标区下方。

```
c.Location = 'southoutside';
```



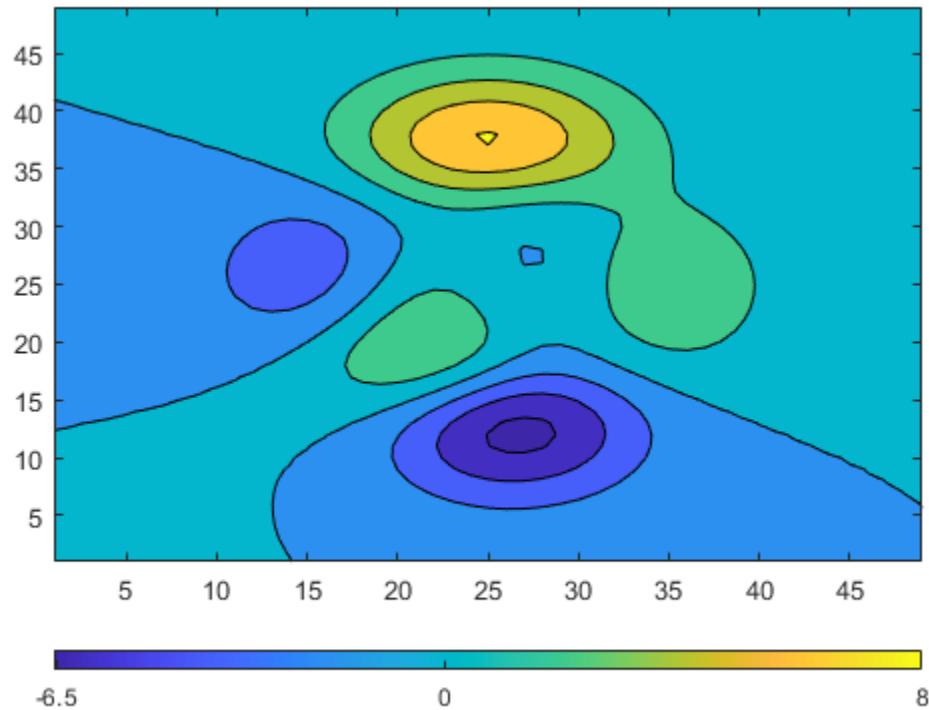
您也可以更改颜色栏的厚度。“Position” 属性控制大部分图形对象（包括坐标区和颜色栏）的位置和大小。由于此颜色栏是水平的，因此 c.Position 中的第四个值（对应于高度）控制其厚度。此处的颜色栏变窄，且坐标区位置进行了重置，因此没有与此颜色栏重叠。

```
ax = gca;
axpos = ax.Position;
c.Position(4) = 0.5*c.Position(4);
ax.Position = axpos;
```



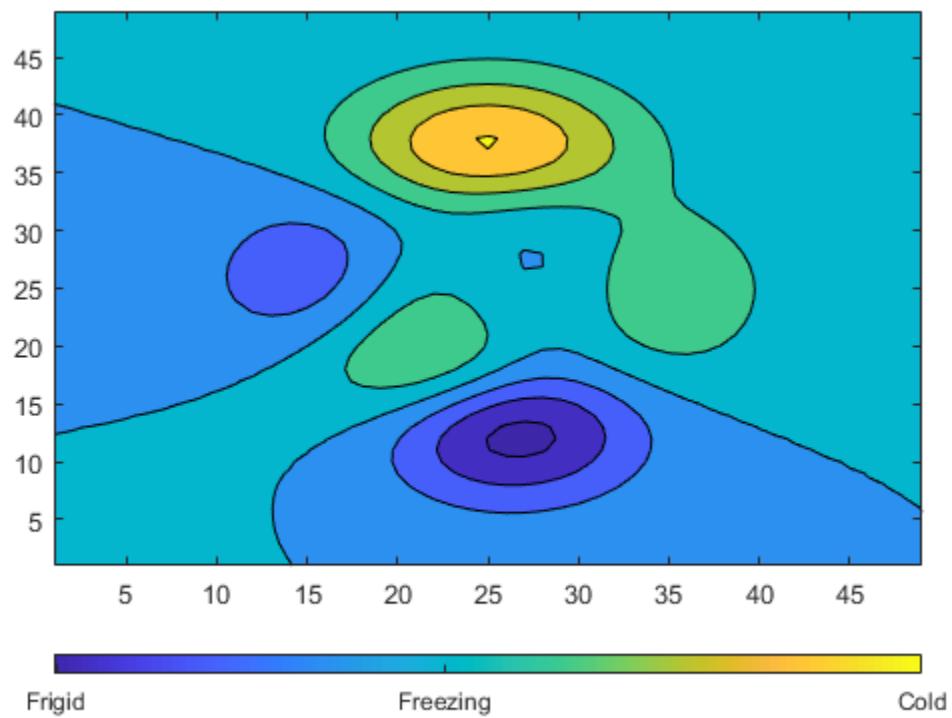
Colorbar 对象具有多项用于修改刻度间距和标签的属性。例如，您可以指定仅在这三处显示刻度：-6.5、0 和 8。

```
c.Ticks = [-6.5 0 8];
```



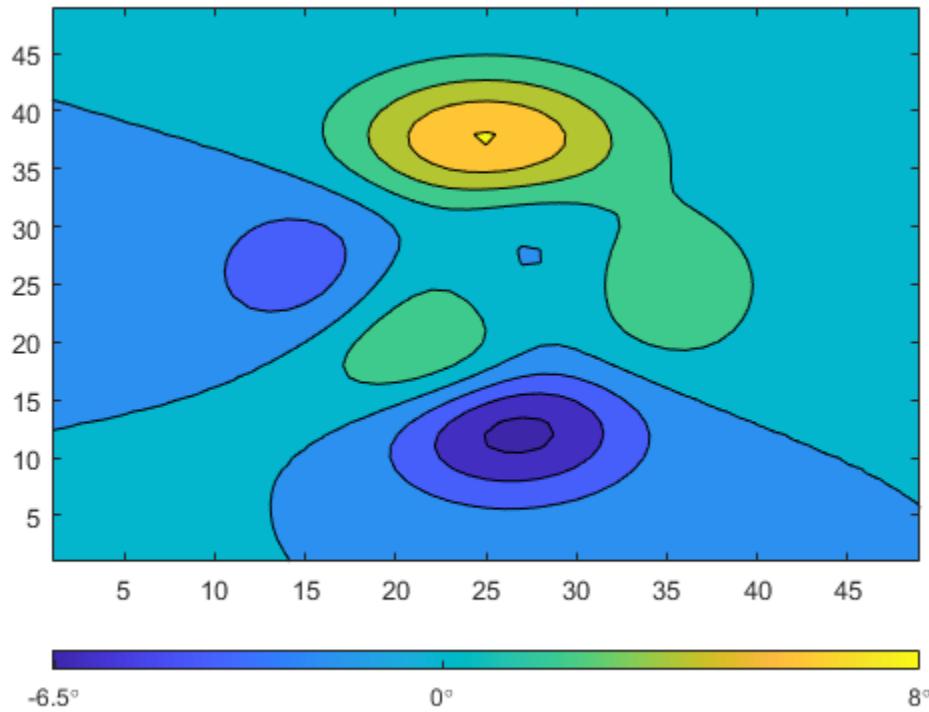
您可以将刻度标签更改为任意值。使用元胞数组指定刻度标签。

```
c.TickLabels = {'Frigid','Freezing','Cold'};
```



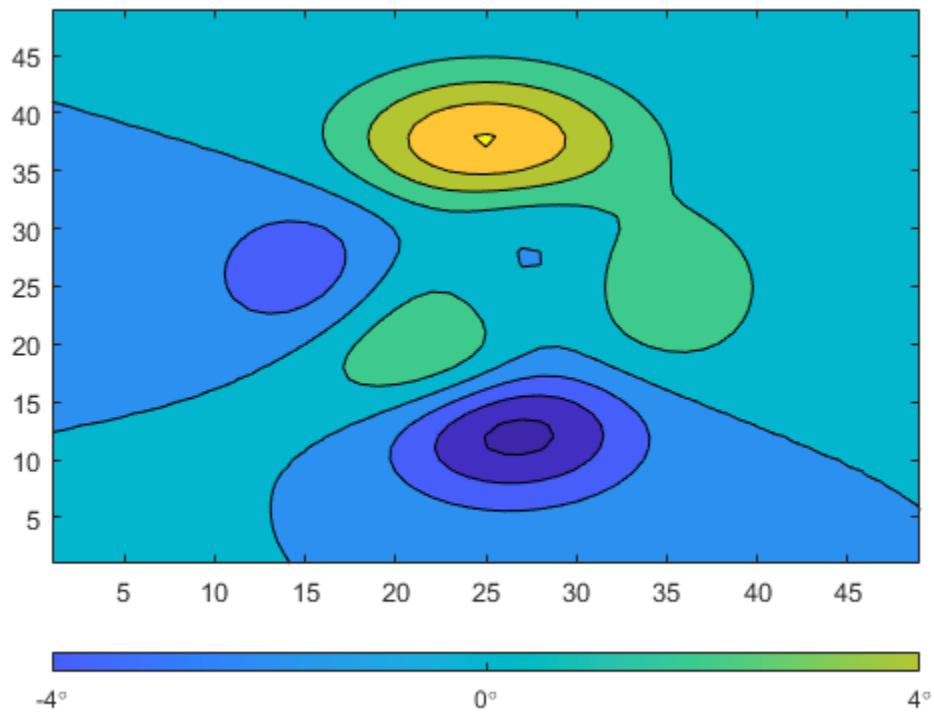
您也可以使用 TeX 或 LaTeX 标记。在使用 TeX 或 LaTeX 时通过 **TickLabelInterpreter** 属性设置解释器。

```
c.TickLabelInterpreter = 'tex';
c.TickLabels = {'-6.5\circ','0\circ','8\circ'};
```



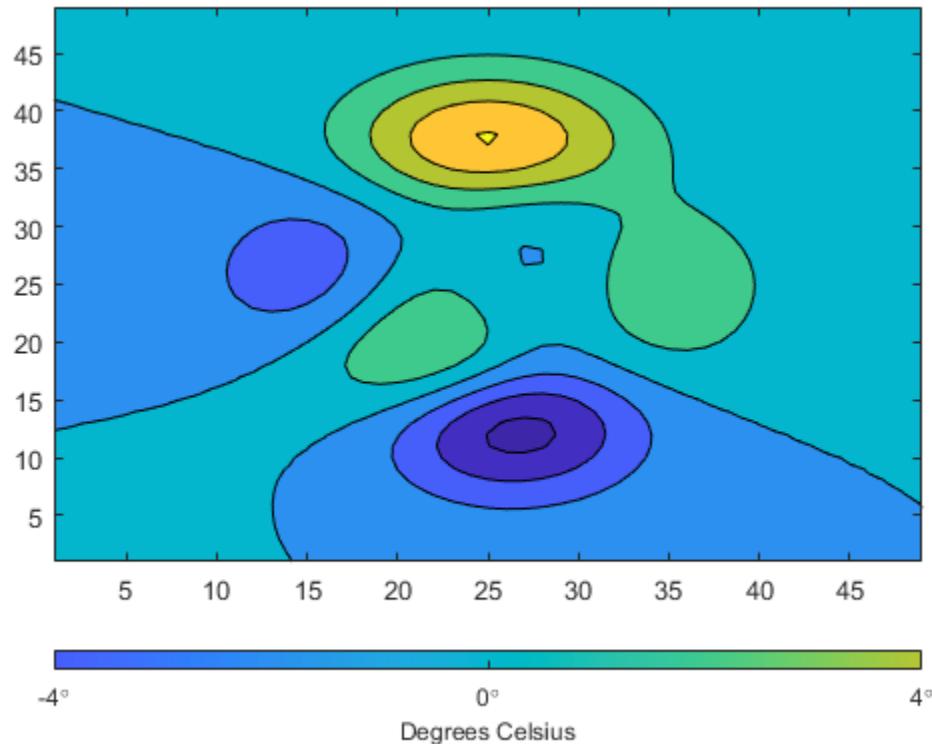
您可以更改颜色栏的范围以突出显示特定颜色区域。例如，您可以缩小范围，并调整刻度标签以反映新范围。得到的颜色栏排除了原本位于左端的深蓝色以及原本位于右端的黄色。

```
c.Limits = [-4 4];  
c.Ticks = [-4 0 4];  
c.TickLabels = {-4\circ,0\circ,4\circ};
```



使用 **Label** 属性向颜色栏添加描述性标签。由于 **Label** 属性必须作为 **Text** 对象指定，因此必须先设置 **Text** 对象的 **String** 属性。然后即可将此 **Text** 对象赋给 **Label** 属性。使用以下命令只需一步即可完成这两项任务。

```
c.Label.String = 'Degrees Celsius';
```



另请参阅

函数

`colorbar | pcolor`

属性

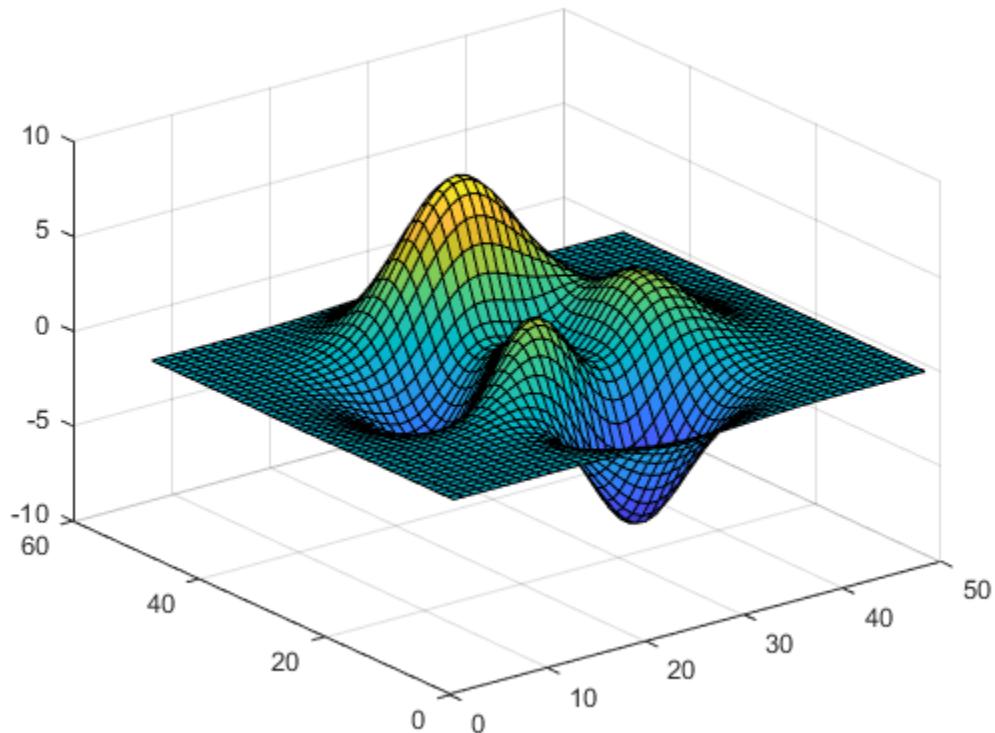
`Colorbar`

使用颜色图更改颜色方案

MATLAB® 在显示曲面绘图等视觉呈现时使用默认的颜色方案。您可以通过指定颜色图来更改颜色方案。颜色图是包含 RGB 三元组的三列数组，其中每一行定义一种不同的颜色。

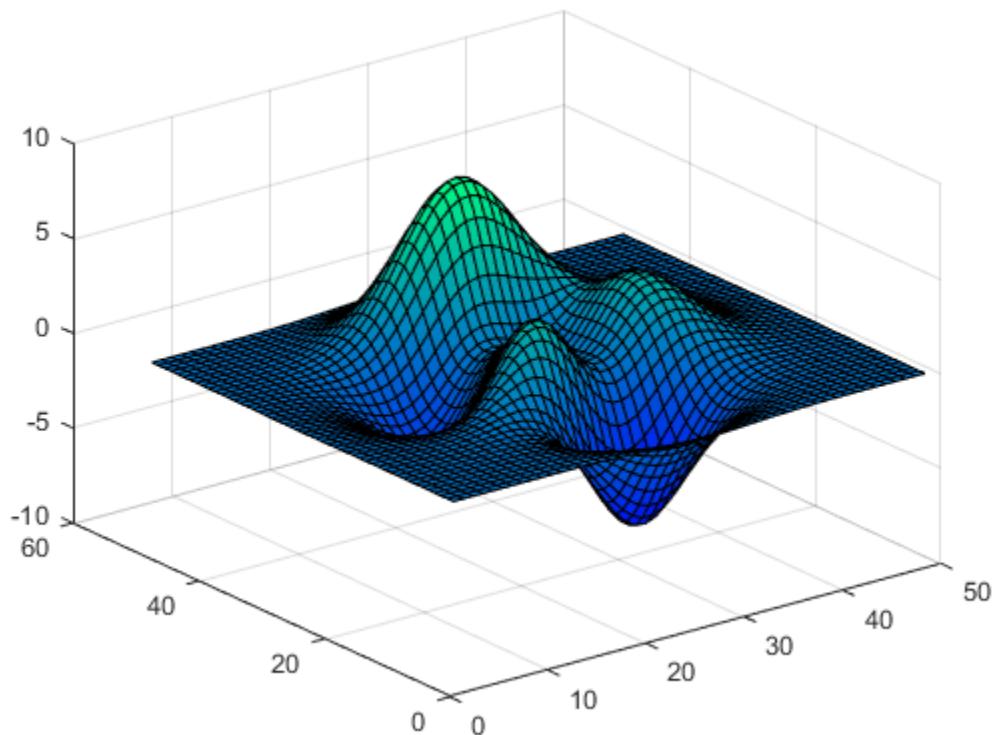
例如，此处为使用默认颜色方案的曲面绘图。

```
f = figure;
surf(peaks);
```



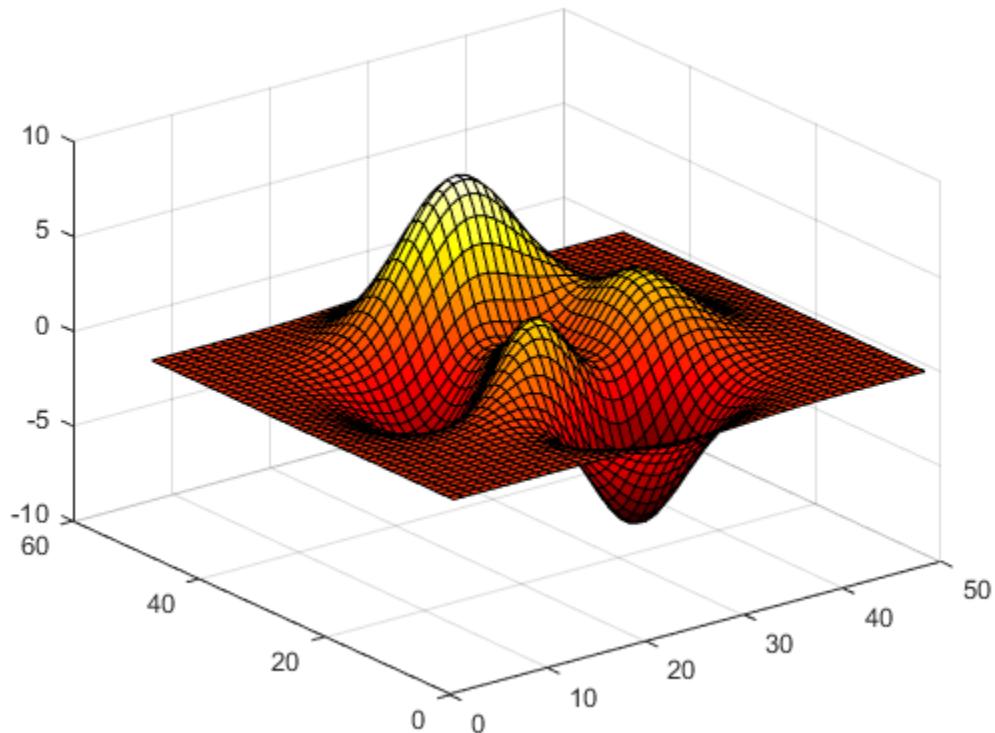
以下命令可将当前图窗的颜色图更改为 `winter`，它是多种预定义颜色图中的一种（有关预定义颜色图的完整列表，请参阅“颜色图”）。

```
colormap winter;
```



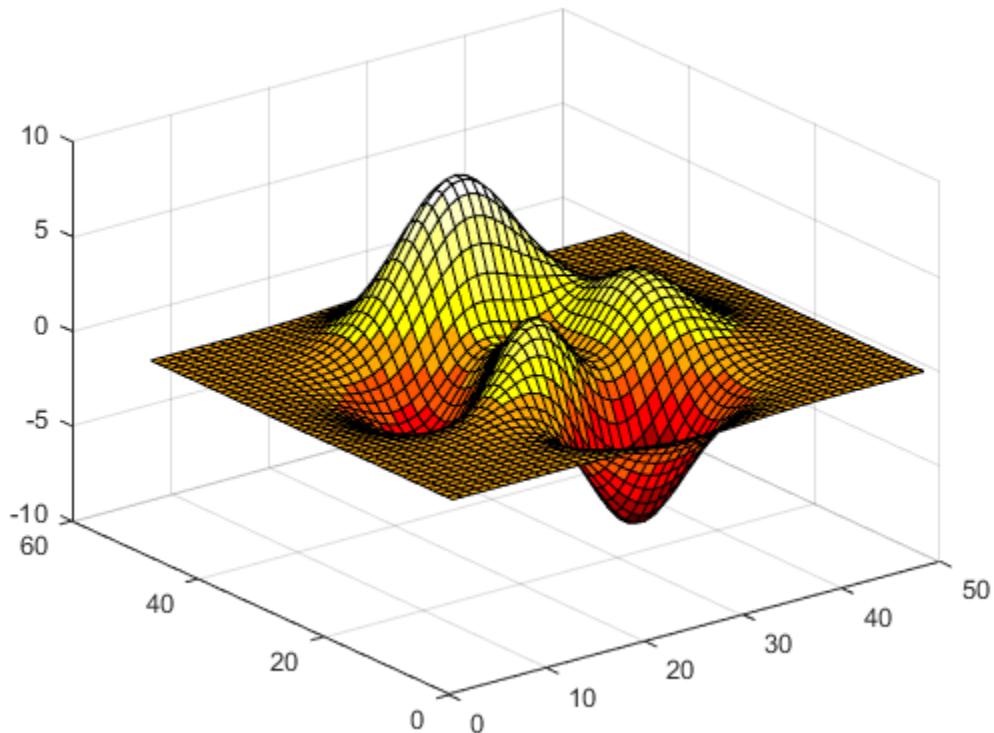
如果您打开了多个图窗，可将 **Figure** 对象作为第一个参数传递给 **colormap** 函数。

```
colormap(f,hot);
```



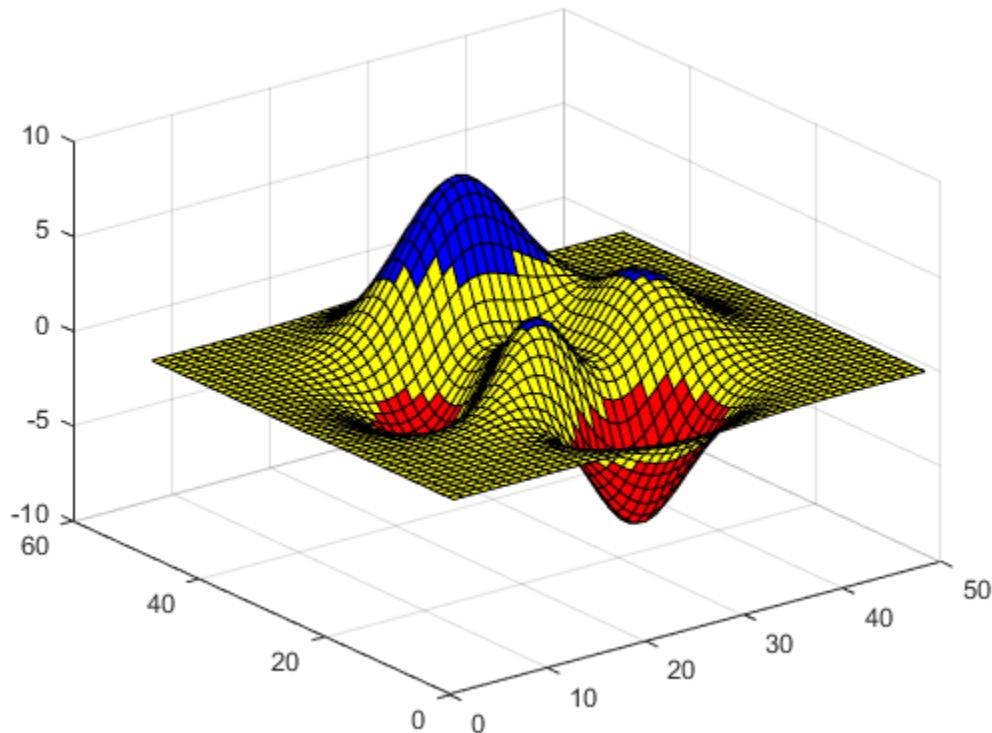
默认情况下，每个预定义的颜色图都会提供一个包含 256 种颜色的调色板。但是，您可以向预定义的颜色图函数传递一个整数来指定所需数量的颜色。例如，此处为包含十个条目的 hot 颜色图。

```
c = hot(10);  
colormap(c);
```



您也可以自行创建 $M \times 3$ 数组形式的颜色图。此数组中的各行包含不同颜色强度的红色、绿色和蓝色。颜色强度介于范围 [0,1] 之间。此处为一个简单的颜色图，其中包含三个条目。

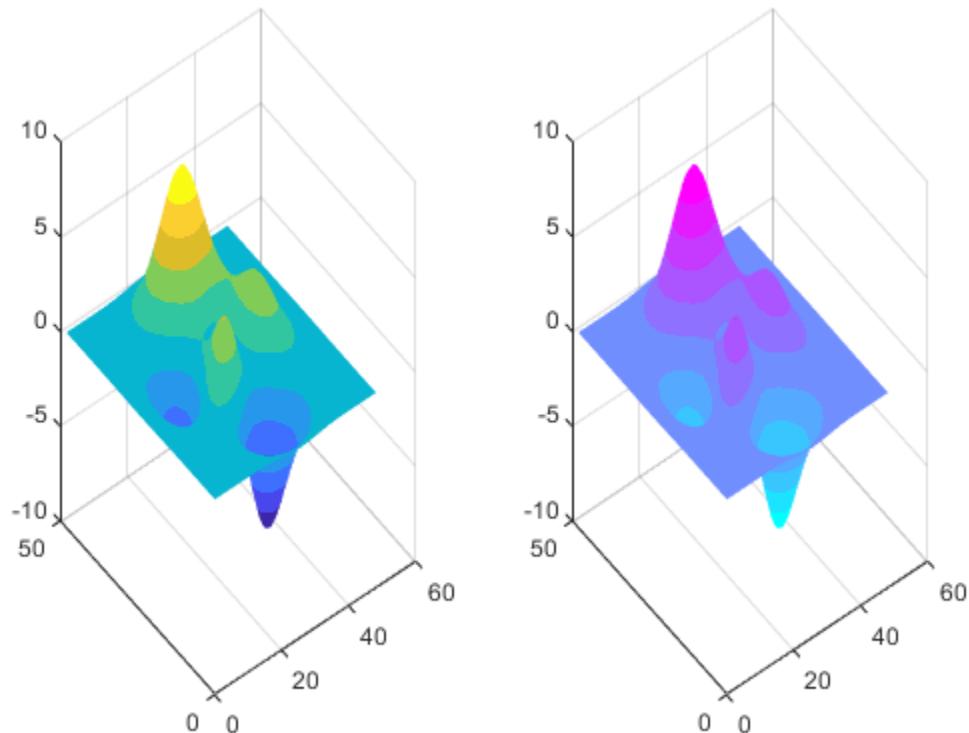
```
mycolors = [1 0 0; 1 1 0; 0 0 1];  
colormap(mycolors);
```



如果使用多个坐标区，可以通过将坐标区对象传递给 `colormap` 函数，为每个坐标区分配不同的颜色图。

```
tiledlayout(1,2)
ax1 = nexttile;
surf(peaks);
shading interp;
colormap(ax1,parula(10));

ax2 = nexttile;
surf(peaks);
shading interp;
colormap(ax2,cool(10));
```



另请参阅

相关示例

- “曲面绘图数据与颜色图的关系” (第 10-16 页)

曲面绘图数据与颜色图的关系

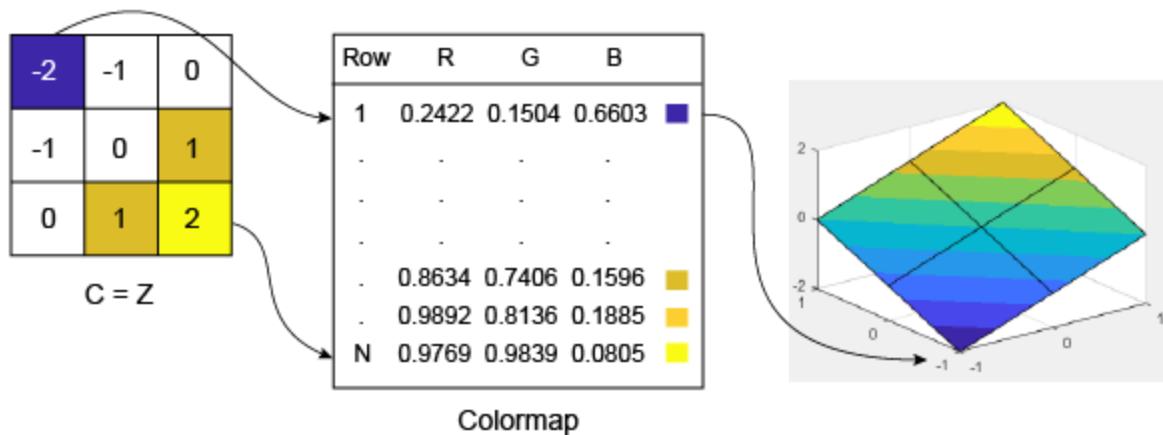
在使用 `surf` 或 `mesh` 等函数创建曲面绘图时，可以通过调用 `colormap` 函数自定义颜色方案。如果要进一步控制外观，可以更改穿过此曲面的颜色的方向或模式。这种自定义需要更改控制曲面和颜色图之间关系的数组中的值。

曲面与颜色图之间的关系

Surface 对象的 `CData` 属性包含一个索引数组 `C`，此数组将绘图中的特定位置与颜色图中的颜色相关联。`C` 与曲面 $z = f(x,y)$ 具有如下关系：

- `C` 的大小与 `Z` 相同，其中 `Z` 是在曲面上每个网格点处都包含 $f(x,y)$ 值的数组。
- $C(i,j)$ 处的值控制曲面上网格位置 (i,j) 的颜色。
- 默认情况下，`C` 等于 `Z`，即颜色随高度而异。
- 默认情况下，`C` 的范围线性映射到颜色图数组中的行数。

举例来说， $Z = X + Y$ 的一个 3×3 抽样与包含 N 个条目的颜色图具有下列关系。



请注意，最小值 (-2) 映射到颜色图中的第一行。最大值 (2) 映射到颜色图中的最后一行。`C` 中的中间值线性映射到颜色图中间的行。

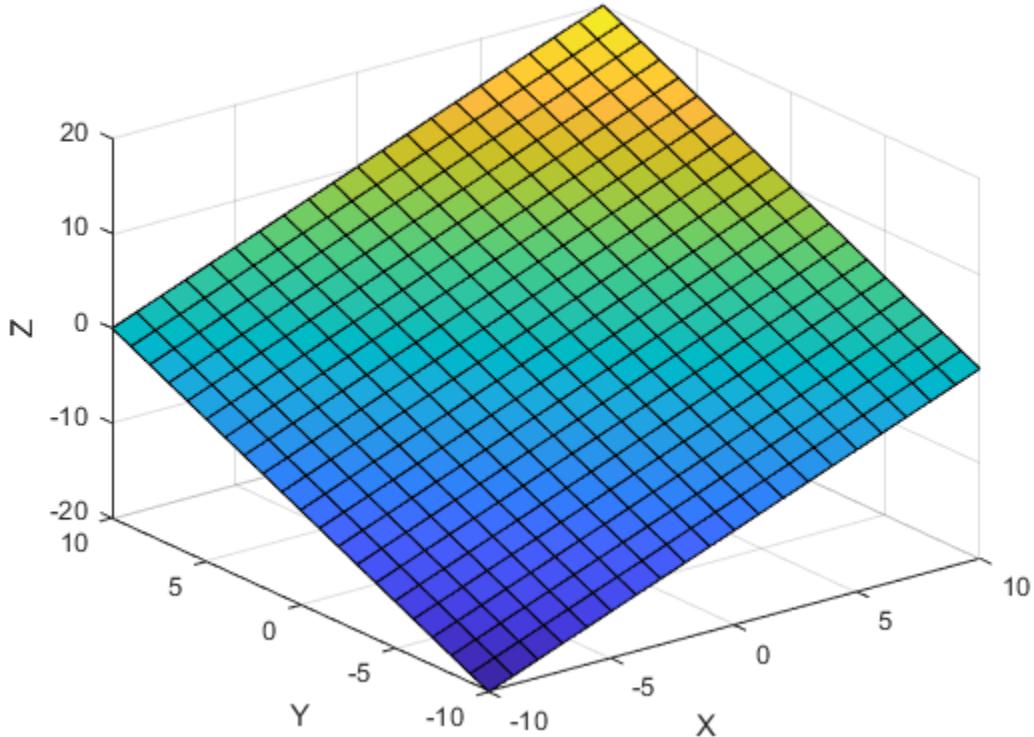
注意 先前的曲面绘图说明颜色如何分配给曲面上的顶点，但默认行为是用纯色填充补片面。该种纯色基于分配给周围顶点的颜色。有关详细信息，请参阅 `FaceColor` 的属性说明。

更改颜色的方向或模式

在使用 `C=Z` 的默认值时，颜色随 `Z` 的变化而异。

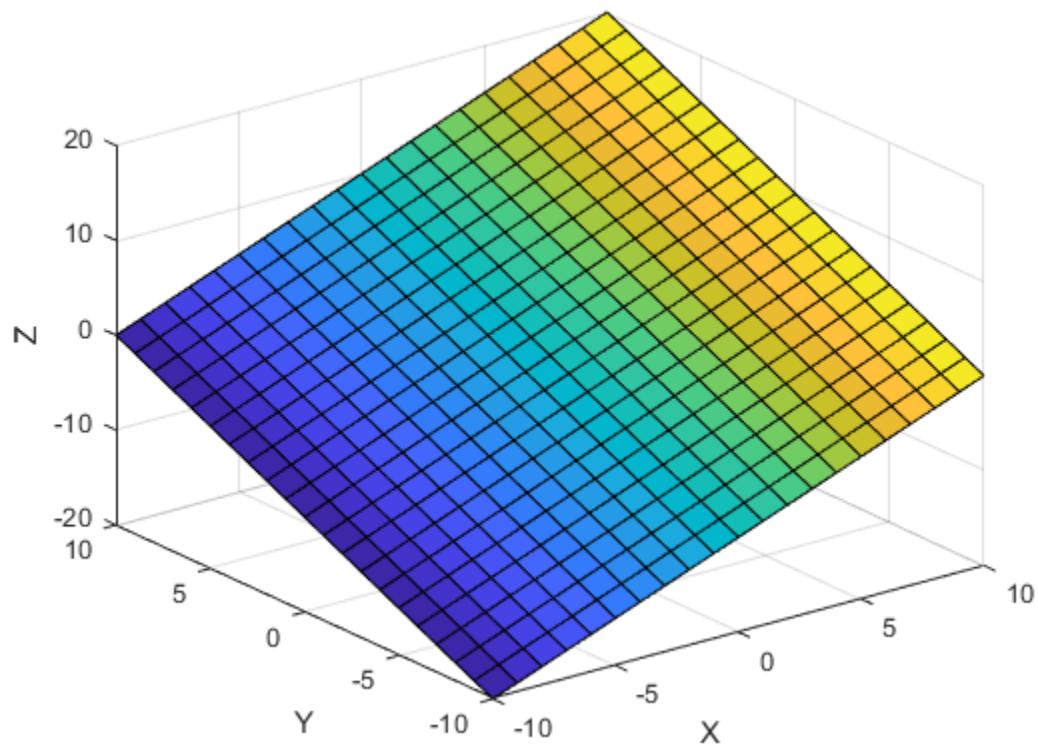
```
[X,Y] = meshgrid(-10:10);
Z = X + Y;
s = surf(X,Y,Z);
xlabel('X');
```

```
ylabel('Y');  
zlabel('Z');
```



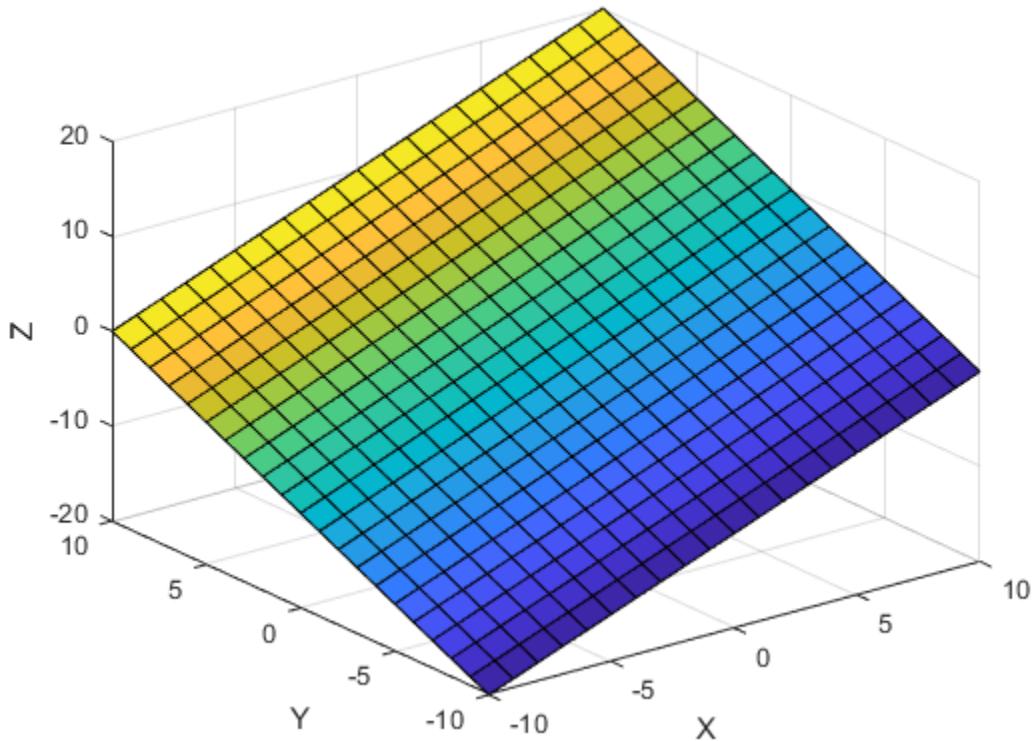
在创建曲面时通过指定 C 可以改变这种行为。例如，以下曲面上的颜色随 X 而异。

```
C = X;  
s = surf(X,Y,Z,C);  
xlabel('X');  
ylabel('Y');  
zlabel('Z');
```



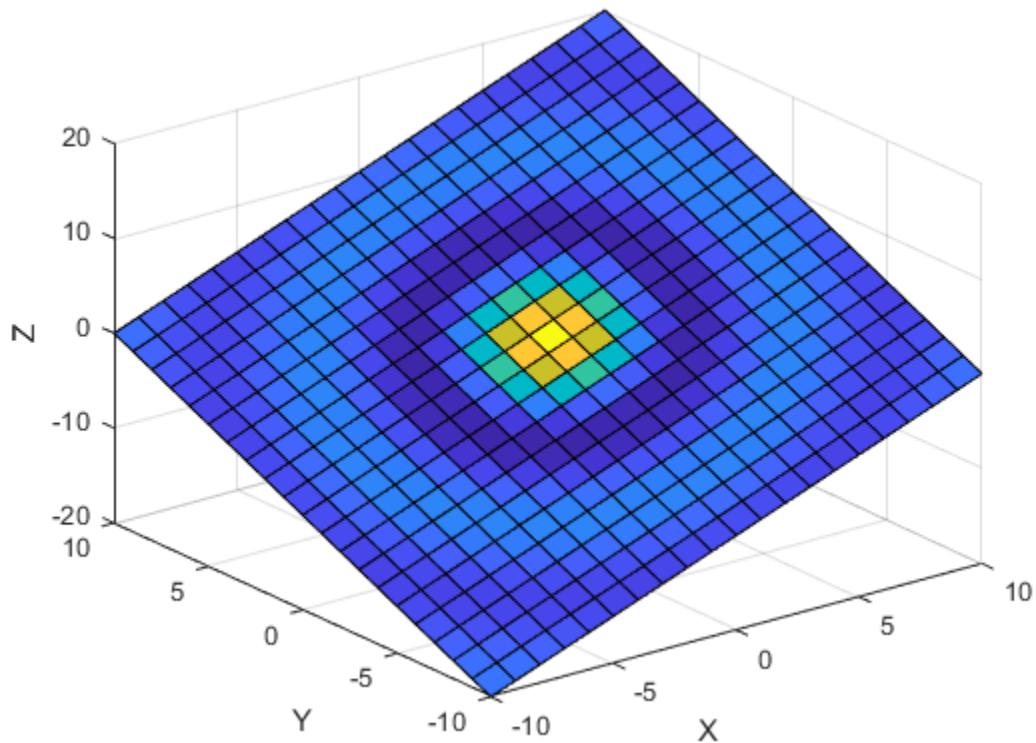
您也可以直接设置 **CData** 属性。以下命令将使得颜色随 Y 而异。

```
s.CData = Y;
```



颜色无需遵循单一维度的变化。实际上，CData 可以是任何与 Z 大小相同的数组。例如，以下平面上的颜色遵循 sinc 函数形状。

```
R = sqrt(X.^2 + Y.^2) + eps;
s.CData = sin(R)./(R);
```



另请参阅

函数

`surf | mesh`

属性

Chart Surface

相关示例

- “使用颜色图更改颜色方案” (第 10-10 页)
- “颜色图和真彩色之间的差异” (第 10-38 页)

图像数据与颜色图的关系

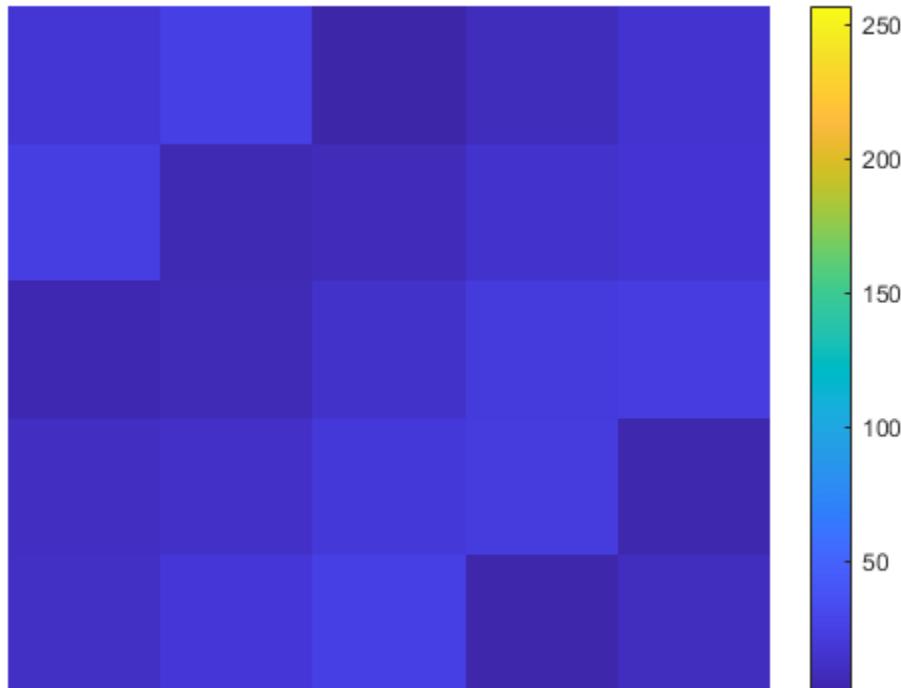
在使用 `image` 函数显示图像时，可以控制像素范围值与颜色图范围的映射关系。例如，此处的 5×5 幻方显示为一个使用默认颜色图的图像。

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

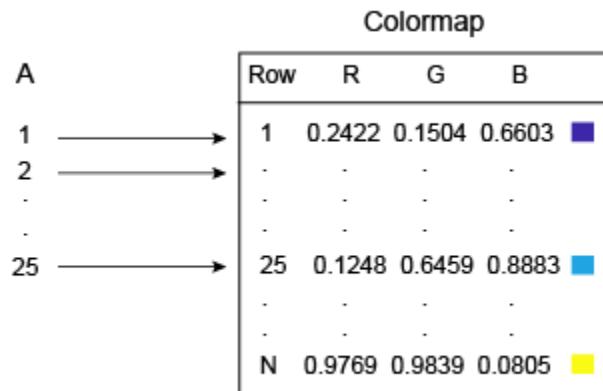
```
A = 5×5
```

```
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
```

```
im = image(A);  
axis off  
colorbar
```

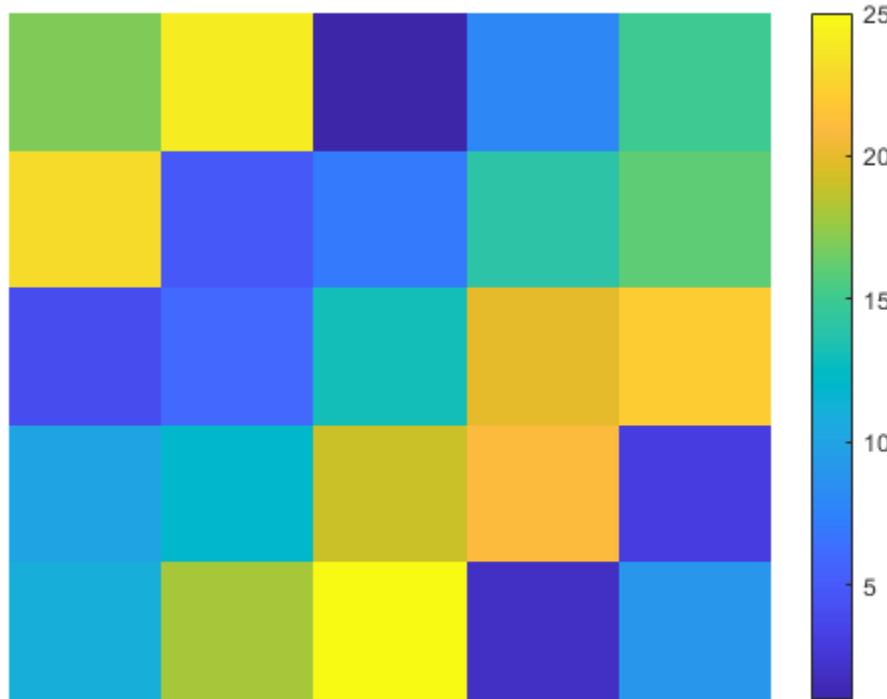


`A` 包含介于 1 和 25 之间的值。MATLAB 将这些值视为指向颜色图（包含 64 个条目）的索引。因此，先前图像中的所有像素映射到该颜色图中的前 25 个条目（大致为颜色栏中的蓝色区域）。

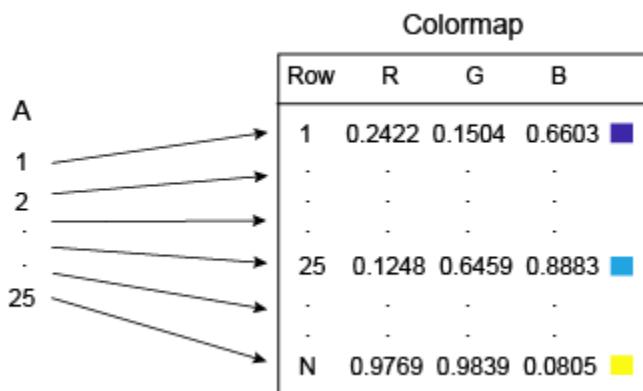


您可以通过 `Image` 对象的 `CDataMapping` 属性控制此映射。先前的图中所示的默认行为对应于此属性的 '`direct`' 选项。在显示包含自带颜色图的图像（例如 GIF 图像）时，直接映射很有用。但是，如果图像表示某些物理单位（例如米或度）的测量值，则可将 `CDataMapping` 属性设置为 '`scaled`'。缩放映射使用完整的颜色范围，它允许您以可视化方式显示数据中的相对差异。

```
im.CDataMapping = 'scaled';
```

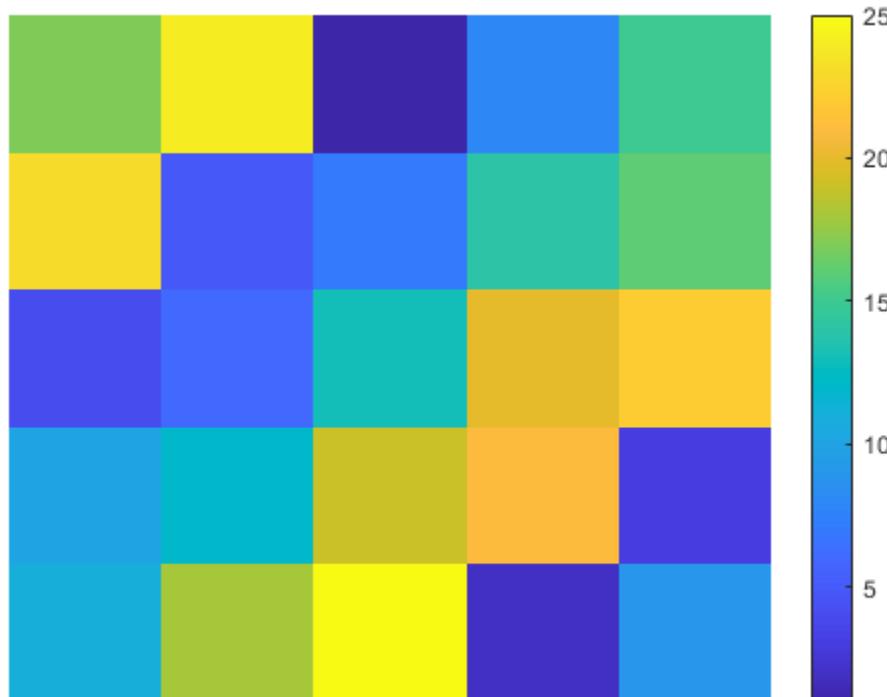


'`scaled`' 选项可将 A 的最小值映射到颜色图中的第一个条目，将 A 的最大值映射到颜色图中的最后一个条目。A 的所有中间值线性缩放映射至颜色图。



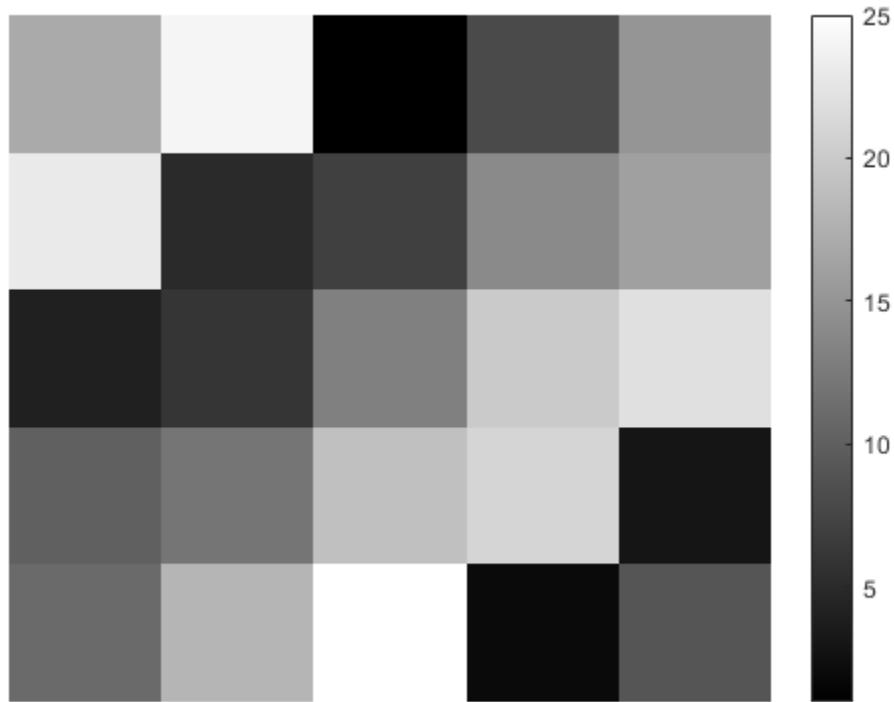
作为设置 `CDataMapping` 属性为 '`scaled`' 的一个替代方法，您还可以调用 `imagesc` 函数以达到相同效果。

```
imagesc(A);
axis off
colorbar
```



如果更改颜色图，A 中的值会缩放映射至新颜色图。

```
colormap(gray);
```



缩放映射也适用于显示不含颜色图的绘图图像，或者您想更改绘图图像的颜色图的情形。以下命令使用 `gray` 颜色图显示图像，此颜色图不同于此图像中存储的原始颜色图。

```
load clown
image(X,'CDataMapping','scaled');
colormap(gray);
axis off
colorbar
```



另请参阅

函数

`image | imagesc`

属性

`Image`

相关示例

- “图像类型” (第 15-4 页)
- “颜色图和真彩色之间的差异” (第 10-38 页)

补片数据与颜色图的关系

在创建使用 Patch 对象的图形时，可以通过调用 `colormap` 函数来控制总体颜色方案。您也可以通过以下方式控制颜色图与补片之间的关系：

- 为面分配特定颜色
- 为各个面周围的顶点分配特定颜色

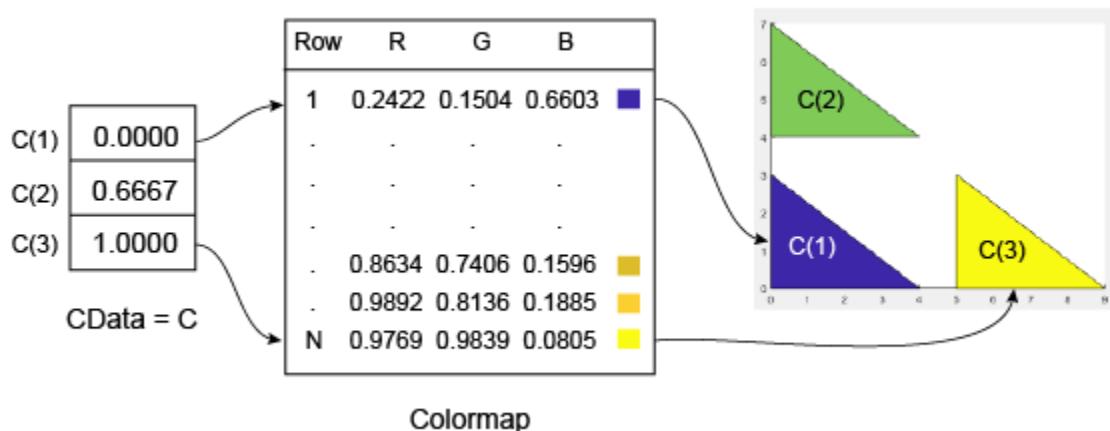
您控制这些关系的方式取决于您指定补片的方式：是指定为 x、y、z 坐标，还是指定为面-顶点数据。

颜色图与 x、y、z 坐标数组的关系

如果您使用 x、y、z 坐标数组创建 Patch 对象，Patch 对象的 `CData` 属性将包含索引数组 C。此数组控制颜色图与补片之间的关系。要为面分配颜色，应将 C 指定为具有以下特征的数组：

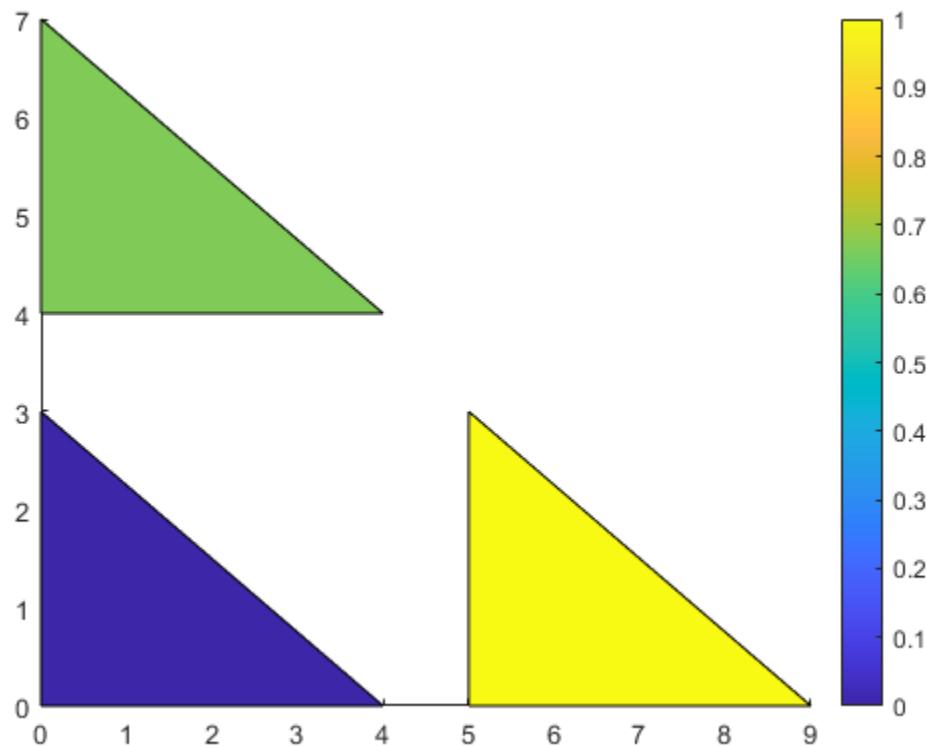
- C 为 $n \times 1$ 数组，其中 n 为面数。
- C(i) 处的值控制面 i 的颜色。

下面的示例演示了 C 及其与颜色图和三个面的关系。C(i) 的值控制顶点 $(X(i,:), Y(i,:))$ 定义的面的颜色。



C 的最小值为 0。它映射到颜色图中的第一行。C 的最大值为 1，它映射到颜色图中的最后一行。C 的中间值线性映射到颜色图的中间行。在本例中，C(2) 映射到距离颜色图起始点约三分之二处的颜色。此代码将创建先前图示中所描述的 Patch 对象。

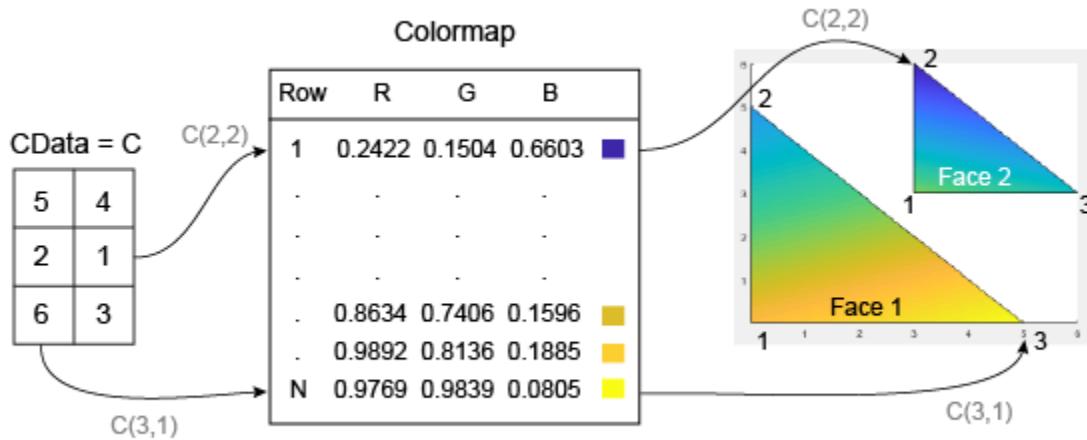
```
X = [0 0 5; 0 0 5; 4 4 9];
Y = [0 4 0; 3 7 3; 0 4 0];
C = [0; .6667; 1];
p = patch(X,Y,C);
colorbar
```



要为顶点分配颜色，应将 C 指定为具有以下特征的数组：

- C 为 $m \times n$ 数组，其中 m 为每个面的顶点数，n 为面数。
- C(i,j) 处的值控制面 j 的顶点 i 处的颜色。

下面的示例演示了 C 及其与颜色图和六个顶点之间的关系。C(i,j) 的值控制 $(X(i,j), Y(i,j))$ 处的顶点的颜色。



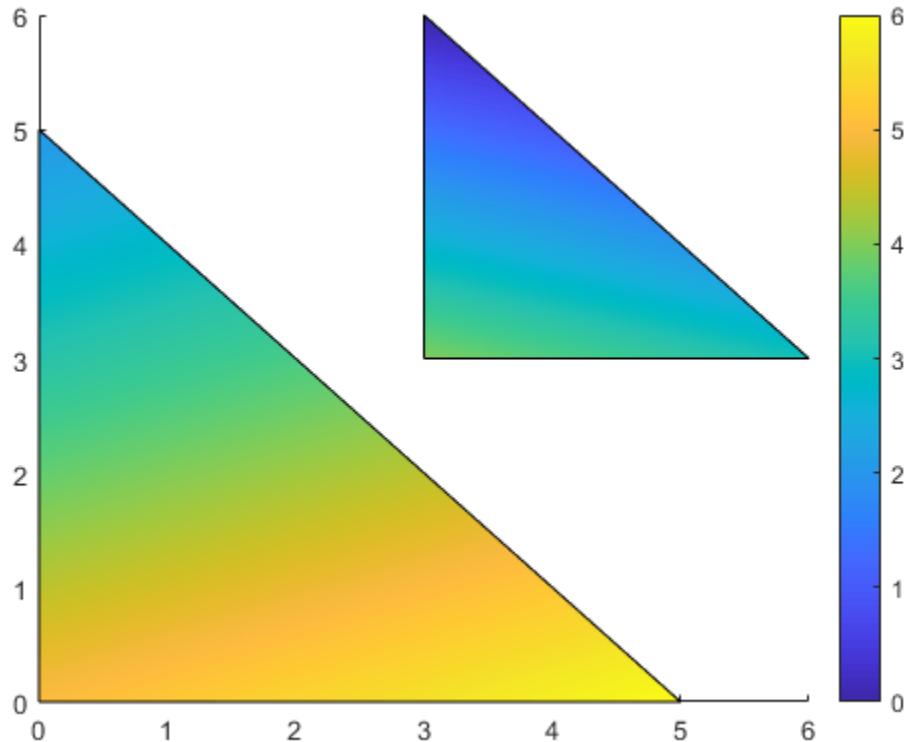
与补片面一样，MATLAB 可将 C 中的值缩放映射到颜色图中的行数。在本例中，最小值为 $C(2,2)=1$ ，它映射到颜色图中的第一行。最大值为 $C(3,1)=6$ ，它映射到颜色图中的最后一行。

此代码将创建先前图示中所描述的 **Patch** 对象。**FaceColor** 属性设置为 '**interp**'，以使顶点颜色在各个面之间混合。

```

clf
X = [0 3; 0 3; 5 6];
Y = [0 3; 5 6; 0 3];
C = [5 4; 2 0; 6 3];
p = patch(X,Y,C,'FaceColor','interp');
colorbar

```



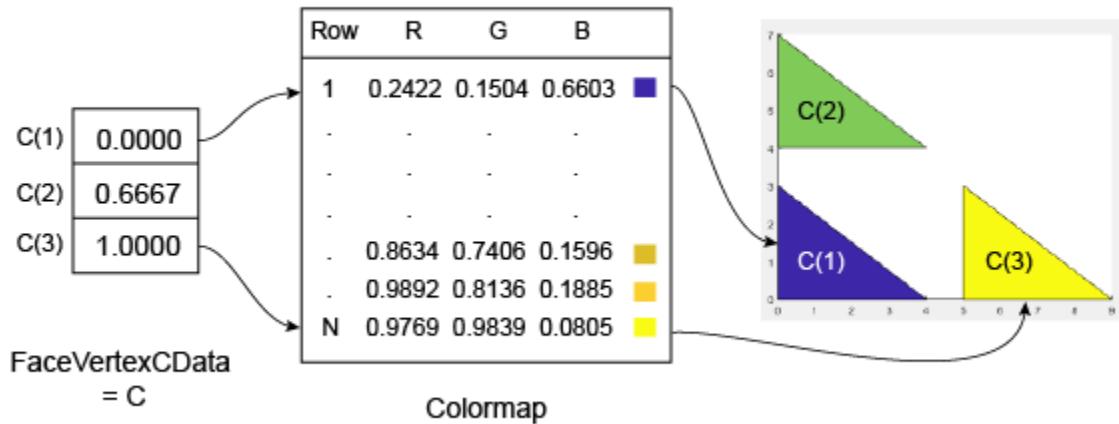
颜色图与面-顶点数据的关系

如果是使用面-顶点数据创建补片，Patch 对象的 FaceVertexCData 属性将包含索引数组 C。此数组控制颜色图与补片之间的关系。

要为面分配颜色，应将 C 指定为具有以下特征的数组：

- C 为 $n \times 1$ 数组，其中 n 为面数。
- C(i) 处的值控制面 i 的颜色。

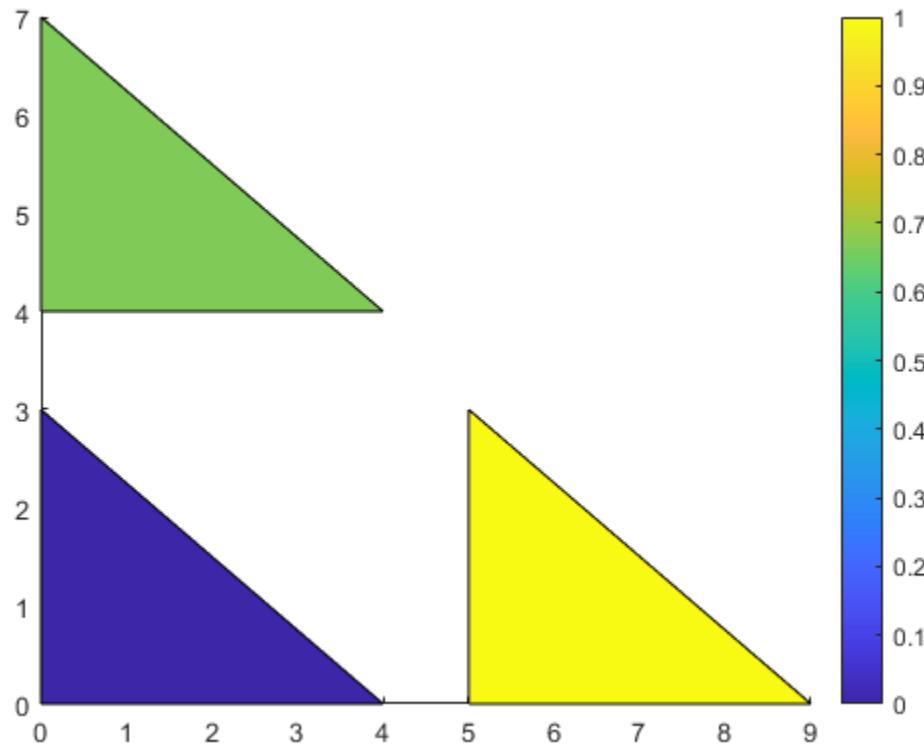
下面的示例演示了 C 及其与颜色图和三个面的关系。



C 的最小值为 0，它映射到颜色图中的第一行。 C 的最大值为 1，它映射到颜色图中的最后一个值。 C 的中间值线性映射到颜色图的中间行。在本例中， $C(2)$ 映射到距离颜色图底部约三分之二处的颜色。

此代码将创建先前图示中所描述的 Patch 对象。FaceColor 属性设置为 'flat'，以显示颜色图的颜色而不显示默认颜色（黑色）。

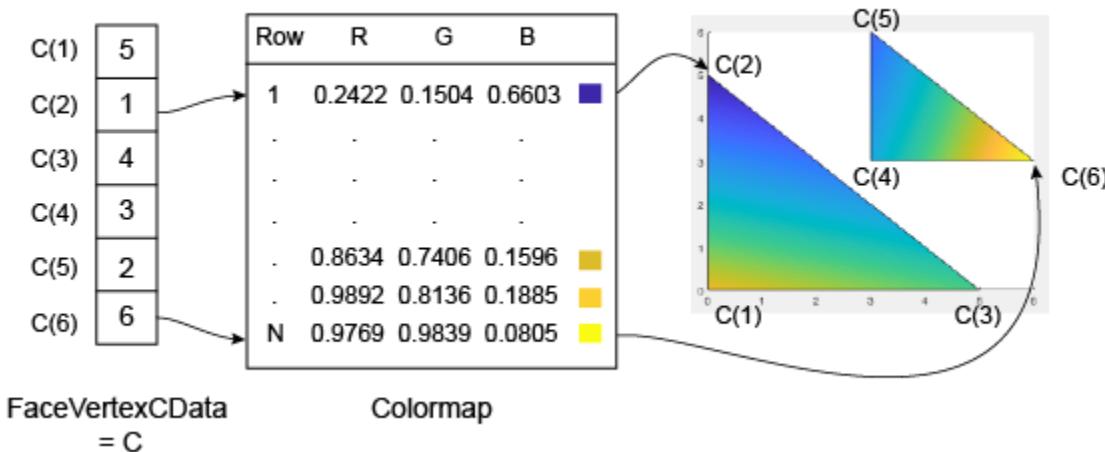
```
clf
vertices = [0 0; 0 3; 4 0; 0 4; 0 7; 4 4; 5 0; 5 3; 9 0];
faces = [1 2 3; 4 5 6; 7 8 9];
C = [0; 0.6667; 1];
p = patch('Faces',faces,'Vertices',vertices,'FaceVertexCData',C);
p.FaceColor = 'flat';
colorbar
```



要为顶点分配颜色，应将 Patch 对象的 FaceVertexCData 属性指定为具有以下特征的数组 C：

- C 为 $n \times 1$ 数组，其中 n 为顶点数。
- C(i) 处的值控制顶点 i 的颜色。

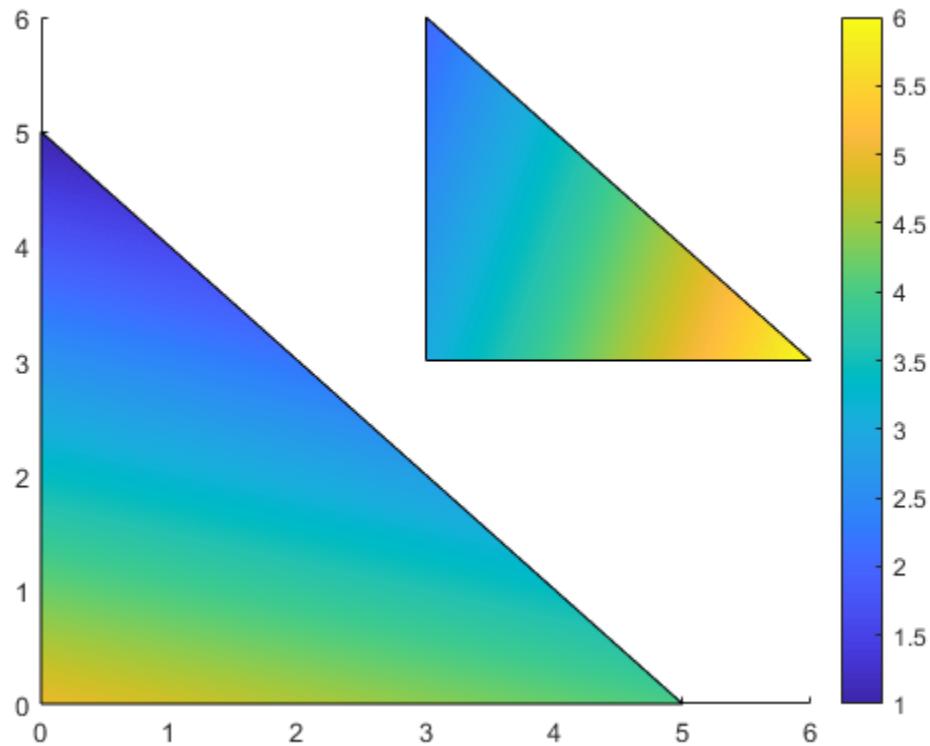
下面的示例演示了 C 及其与颜色图和六个顶点之间的关系。



与补片面一样，MATLAB 可将 C 中的值缩放映射到颜色图中的行数。在本例中，最小值为 $C(2)=1$ ，它映射到颜色图中的第一行。最大值为 $C(6)=6$ ，它映射到颜色图中的最后一行。

此代码将创建先前图示中所描述的 Patch 对象。FaceColor 属性设置为 'interp'，以使顶点颜色在各个面之间混合。

```
clf
vertices = [0 0; 0 5; 5 0; 3 3; 3 6; 6 3];
faces = [1 2 3; 4 5 6];
C = [5; 1; 4; 3; 2; 6];
p = patch('Faces',faces,'Vertices',vertices,'FaceVertexCData',C);
p.FaceColor = 'interp';
colorbar
```



另请参阅

函数

[patch](#)

属性

[Patch](#)

相关示例

- “使用颜色图更改颜色方案”（第 10-10 页）

- “颜色图和真彩色之间的差异” (第 10-38 页)

控制颜色图范围

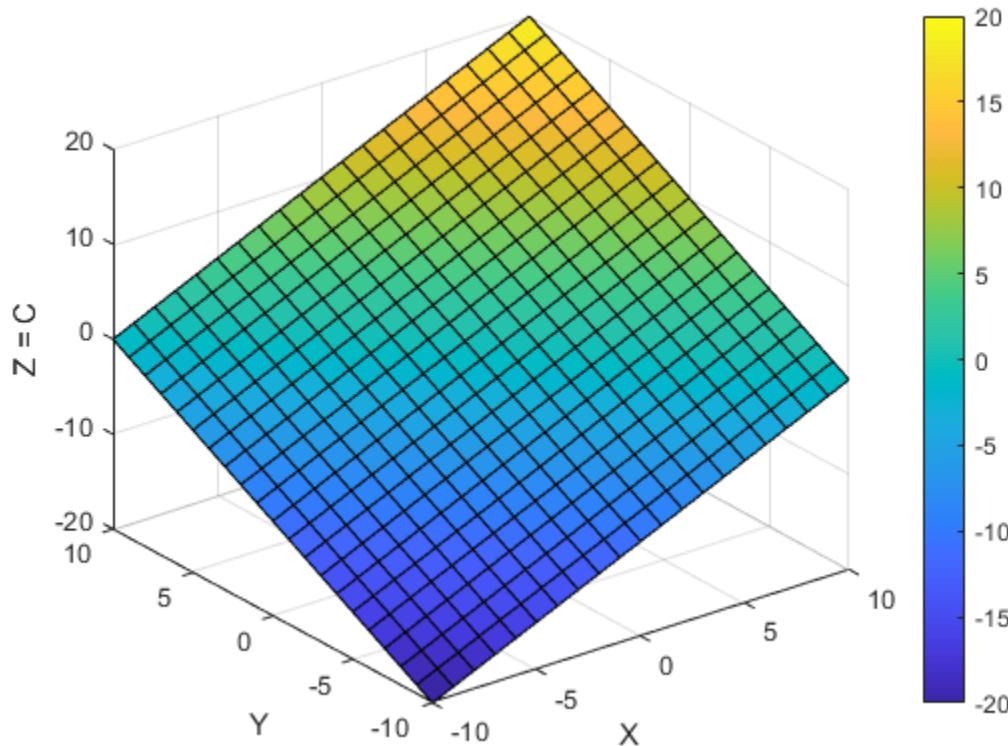
对于您创建的许多类型的可视化图形，MATLAB 默认将完整的数据范围映射到颜色图上。数据中的最小值映射到颜色图中的第一行，最大值映射到颜色图中的最后一行。所有中间值线性映射到颜色图的中间行。

这种默认映射适用于大部分情况，但您也可以对选定的任意范围进行映射，即便您选择的范围不同于数据的范围也可以。通过选择不同的映射范围可以执行以下操作：

- 查看有哪些数据刚好处于指定的范围边界或超出指定的范围。
- 查看您的数据落在指定范围的哪个位置。

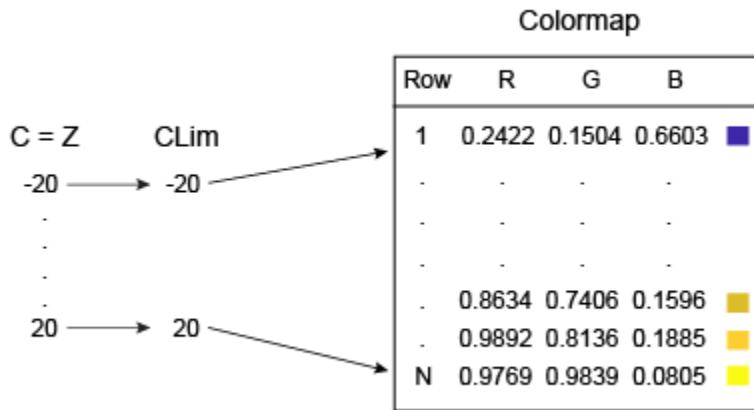
假设曲面 $Z = X + Y$ ，其中 $-10 \leq x \leq 10$, $-10 \leq y \leq 10$ 。

```
[X,Y] = meshgrid(-10:10);
Z = X + Y;
s = surf(X,Y,Z);
xlabel('X');
ylabel('Y');
zlabel('Z = C');
colorbar
```

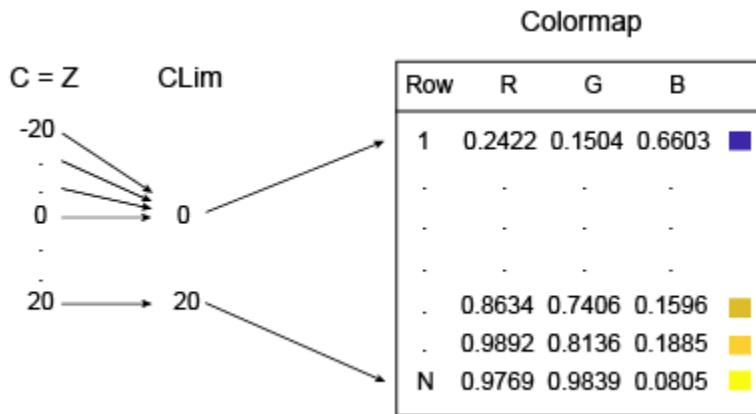


“曲面绘图数据与颜色图的关系”（第 10-16 页）描述了控制此演示中颜色的属性。实质上，Surface 对象的 **CData** 属性包含一个数组 **C**，该数组将曲面上的每个网格点与颜色图中的一种颜色建立关联。默认情况下，**C** 等于 **Z**，其中 **Z** 是在网格点处包含 $z = f(x,y)$ 值的数组。因此，这些颜色随 **Z** 的变化而变化。

映射范围由 Axes 对象的 CLim 属性控制。此属性包含 [cmin cmax] 形式的二元素向量。cmin 的默认值等于 C 的最小值，cmax 的默认值等于 C 的最大值。在本例中，CLim 为 [-20 20]，因为 C 的范围反映 Z 的范围。

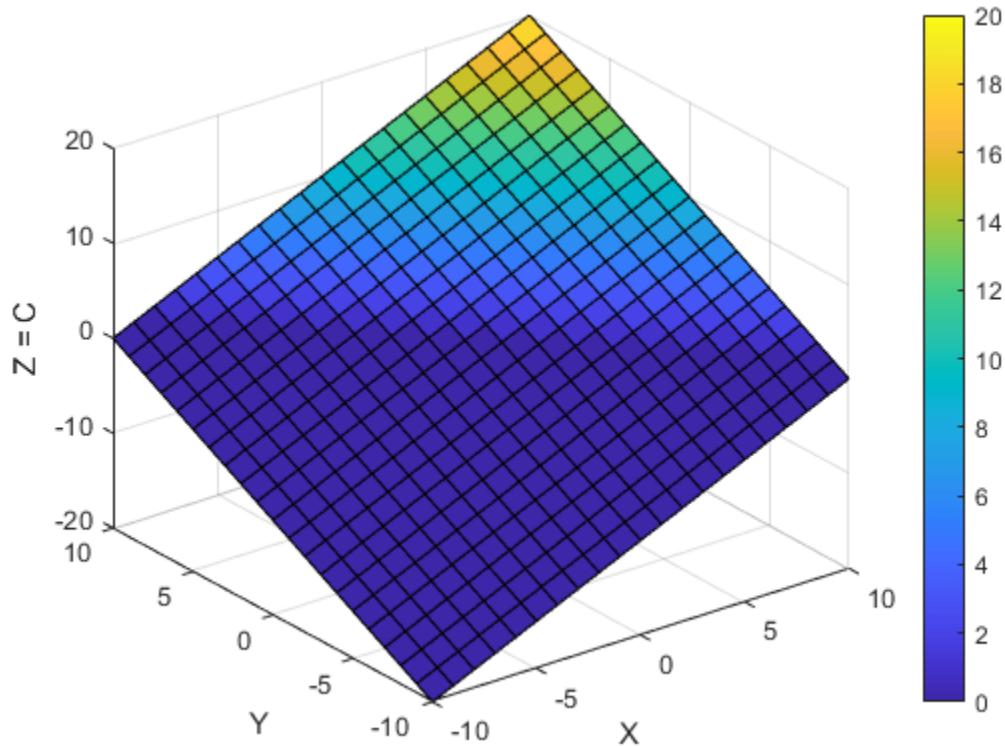


将 CLim 更改为 [0 20] 会将 0 处以及其下的所有值统一映射为颜色图中的第一种颜色。



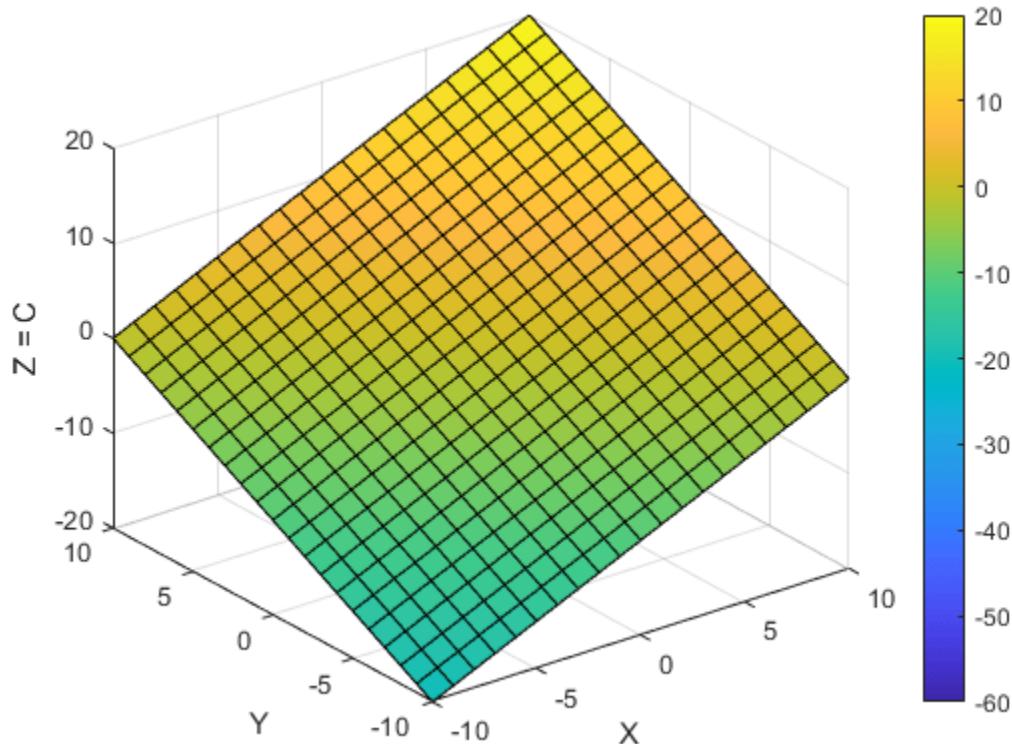
此命令会将 CLim 属性更改为 [0 20]。注意，曲面的下半部分映射到颜色图中的第一种颜色（深蓝色）。之所以进行这种裁剪，是因为 C (等于 Z) 在这些点处的值小于或等于 0。

```
caxis([0 20]);
```



您也可以拓宽映射范围，以查看数据在范围中所处的位置。例如，将范围更改为 [-60 20] 将得到仅使用一半颜色的曲面。颜色图的下半部分对应于 C 范围之外的值，因此曲面上不呈现这些颜色。

```
caxis([-60 20]);
```



注意 您可以为曲面绘图、补片、图像或任何使用颜色图的图形对象设置 `CLim` 属性。但是，此属性仅影响 `CDataMapping` 属性设置为 '`scaled`' 的图形对象。如果 `CDataMapping` 属性设置为 '`direct`'，则 C 索引的所有值将直接映射到颜色图，而不会进行任何缩放。C 的任何小于 1 的值将裁剪映射至颜色图中的第一种颜色。C 的任何大于颜色图长度的值则裁剪映射至颜色图中的最后一种颜色。

另请参阅

[colormap](#) | [colorbar](#) | [caxis](#) | [surf](#)

相关示例

- “使用颜色图更改颜色方案”（第 10-10 页）
- “曲面绘图数据与颜色图的关系”（第 10-16 页）
- “创建颜色栏”（第 10-2 页）

颜色图和真彩色之间的差异

曲面、补片和图像等许多图形对象支持两种不同的指定颜色的方法：颜色图（使用索引颜色）和真彩色。每种方法涉及不同的工作流，对视觉表示有不同的影响。

工作流差异

颜色图是一个 $m \times 3$ 数组，其中每一行指定一个 RGB 三元组。要在图形表示中使用颜色图，应为图形中的每个位置分配一个索引。每个索引对应颜色图中的一行，用以在图形中的指定位置显示一种颜色。相比之下，使用真彩色则是在图形中的每个位置指定一个 RGB 三元组。

确定使用何种方法时要考虑的一些要点如下。

- 真彩色更为直接。如果您想为图形中的特定位置分配具体的红色、绿色、蓝色值，使用真彩色通常更为容易。
- 要在调色板区域中进行更改，则使用颜色图更为容易。例如，如果您想加亮从蓝色到绿色的渐变，则在颜色图中编辑这些行比在图形中的各个位置编辑颜色更为容易。
- 您的数据格式可能更适合其中一个工作流。例如，许多 GIF 压缩图像使用索引颜色进行存储。

这两种标记颜色的方法都是使用颜色数组 C 来指定颜色信息。C 的形状取决于图形对象的类型以及您选取的颜色标记方法。下表归纳了这些差异。

图形对象的类型	包含颜色数组 C 的属性	索引颜色 C 的形状	真彩色 C 的形状
Surface	CData	C 为 $m \times n$ 数组，其大小与 z 坐标数组相同。 C(i,j) 处的值指定 Z(i,j) 的颜色图索引。	C 为 $m \times n \times 3$ 数组，其中 C(:,:,i) 的大小与 z 坐标数组相同。 C(i,j,1) 指定 Z(i,j) 的红色分量。 C(i,j,2) 指定 Z(i,j) 的绿色分量。 C(i,j,3) 指定 Z(i,j) 的蓝色分量。
Image	CData	C 是 $m \times n$ 图像的一个 $m \times n$ 数组。C(i,j) 处的值指定像素 (i,j) 的颜色图索引。	C 是 $m \times n$ 图像的一个 $m \times n \times 3$ 数组。 C(i,j,1) 指定像素 (i,j) 的红色分量。 C(i,j,2) 指定像素 (i,j) 的绿色分量。 C(i,j,3) 指定像素 (i,j) 的蓝色分量。

图形对象的类型	包含颜色数组 C 的属性	索引颜色 C 的形状	真彩色 C 的形状
Patch (x, y, z)	CData	在标记补片面的颜色时，C 是 m 个补片面的一个 $1 \times m$ 数组。C(i) 指定面 i 的颜色图索引。 在标记补片顶点的颜色时，C 是一个 $m \times n$ 数组，其中 m 为每个面的顶点数，n 为面数。 C(i,j) 指定面 i 的顶点 j 的颜色图索引。	在标记补片面的颜色时，C 是 m 个补片面的一个 $m \times 3$ 数组。C(i,:) 指定面 i 的红色、绿色和蓝色值。 在标记补片顶点的颜色时，C 是一个 $n \times 3$ 数组，其中 n 为顶点总数。 C(i,:) 指定顶点 i 的红色、绿色和蓝色值。
Patch (面-顶点数据)	FaceVertexCData	在标记补片面的颜色时，C 是 m 个补片面的 $1 \times m$ 数组。C(i) 指定面 i 的颜色图索引。 在标记补片顶点的颜色时，C 为 $1 \times n$ 数组，其中 n 为顶点总数。C(i) 指定顶点 i 的颜色图索引。	在标记补片面的颜色时，C 是 m 个补片面的一个 $m \times 3$ 数组。C(i,:) 指定面 i 的红色、绿色和蓝色值。 在标记补片顶点的颜色时，C 是一个 $n \times 3$ 数组，其中 n 为顶点总数。 C(i,:) 指定顶点 i 的红色、绿色和蓝色值。

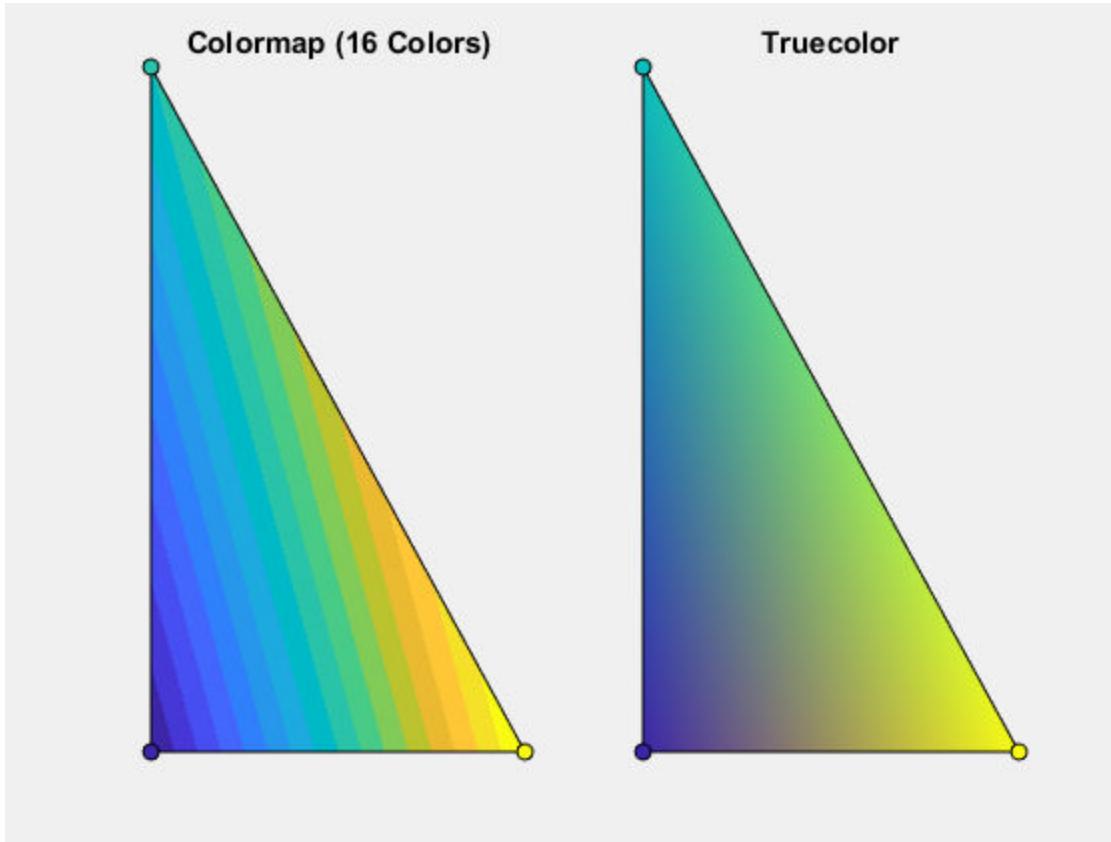
视觉表示的差异

颜色图提供由 m 种颜色组成的调色板，其中 m 为颜色图的长度。相比之下，真彩色则提供涵盖 $256 \times 256 \times 256 \approx 168$ 万种颜色的调色板。

在确定调色板的大小时应考虑以下因素：

- 用纯色填充大块区域时，最经济的方法是使用较小的调色板。它们在可视化曲面等高线时也很有用。
- 较大的调色板显示细微过渡和平滑颜色梯度的效果更好。

通过跨补片面的顶点进行颜色插值能够更明显地看出索引颜色和真彩色之间的差异。下面两个补片说明了一种极端情况。左侧的补片使用了小颜色图，而右侧的补片使用了真彩色。



如果您担心颜色图的调色板有限，可以向调色板中添加更多颜色。“使用颜色图更改颜色方案”（第 10-10 页）说明了如何使用包含特定数量颜色的颜色图。

另请参阅

相关示例

- “图像类型”（第 15-4 页）
- “使用颜色图更改颜色方案”（第 10-10 页）
- “曲面绘图数据与颜色图的关系”（第 10-16 页）
- “图像数据与颜色图的关系”（第 10-21 页）
- “补片数据与颜色图的关系”（第 10-26 页）

光照

- “光照概述” (第 11-2 页)
- “图形对象的反射特性” (第 11-7 页)

光照概述

本节内容

- “光照命令”（第 11-2 页）
- “光源对象”（第 11-2 页）
- “影响光照的属性”（第 11-2 页）
- “光照控制示例”（第 11-3 页）

光照命令

MATLAB 图形环境提供的命令允许您放置光源并调整反射光线的对象的特性。这些命令如下。

命令	用途
<code>camlight</code>	相对于相机位置创建或移动光源
<code>lightangle</code>	在球面坐标中创建或放置光源
<code>light</code>	创建光源对象
<code>lighting</code>	选择光照方法
<code>material</code>	设置被照亮对象的反射属性

您可以设置光源和被照亮对象的属性，以获得特定的结果。除了本主题中的材料外，您还可以通过学习光照示例来了解光照在可视化中的应用。

光源对象

您可以使用 `light` 函数创建光源对象。有三个重要的光源对象属性：

- **Color** - 光源对象投射的光线的颜色
- **Style** - 无限远（默认值）或局部
- **Position** - 方向（无限远光源）或位置（局部光源）

Color 属性决定来自光源的定向光的颜色。场景中对象的颜色由对象和光源的颜色共同决定。

Style 属性决定光源是从指定位置向所有方向发光的点源（**Style** 设置为 `local`），还是放置在无限远处，从指定位置的方向发出平行光线的光源（**Style** 设置为 `infinite`）。

Position 属性以坐标区数据单位指定光源的位置。在光源无限远的情况下，**Position** 指定光源的方向。

光源影响与其在同一坐标区内的曲面和补片对象。这些对象有许多属性，可在被光源照亮时改变对象的外观。

影响光照的属性

您看不到光源对象本身，但可以看到它们照射在位于相同坐标区内的任何补片和曲面对象上的效果。有许多函数可以创建这些对象，包括 `surf`、`mesh`、`pcolor`、`fill` 和 `fill3` 以及 `surface` 和 `patch` 函数。

您可以通过设置各种坐标区、光源、补片和曲面对象属性来控制光照效果。所有属性都具有默认值，它们一般情况下可以产生理想的结果。但是，您可以通过调整这些属性的值来实现想要的特定效果。

属性	作用
AmbientLightColor	坐标区属性，指定场景中背景光源的颜色，没有方向，均匀地影响所有对象。仅当坐标区内存在可见的光源对象时，才会出现环境光效果。
AmbientStrength	补片和曲面属性，决定从对象反射的环境光分量的强度。
DiffuseStrength	补片和曲面属性，决定从对象反射的散射光分量的强度。
SpecularStrength	补片和曲面属性，决定从对象反射的镜面反射光分量的强度。
SpecularExponent	补片和曲面属性，决定镜面反射光的大小。
SpecularColorReflectance	补片和曲面属性，决定镜面反射光按对象颜色或光源颜色进行着色的程度。
FaceLighting	补片和曲面属性，决定用来计算对象面上的光照效果的方法。选项包括无光照、均一光照或 Gouraud 光照算法。
EdgeLighting	补片和曲面属性，决定用来计算对象边上的光照效果的方法。选项包括无光照、均一光照或 Gouraud 光照算法。
BackFaceLighting	补片和曲面属性，决定当顶点法线远离相机时如何照亮对象的面。此属性对于区分对象的内表面和外表面很有用。
FaceColor	补片和曲面属性，指定对象面的颜色。
EdgeColor	补片和曲面属性，指定对象边的颜色。
VertexNormals	补片和曲面属性，包含对象每个顶点的法向量。MATLAB 使用顶点法向量执行光照计算。虽然 MATLAB 会自动生成此数据，但您也可以指定自己的顶点法向。
NormalMode	补片和曲面属性，决定当您更改对象数据后 MATLAB 是重新计算顶点法向 (auto) 还是使用 VertexNormals 属性的当前值 (manual)。如果您为 VertexNormals 指定值，MATLAB 会将此属性设置为 manual。

有关详细信息，请参阅 Axes、Chart Surface 和 Patch。

光照控制示例

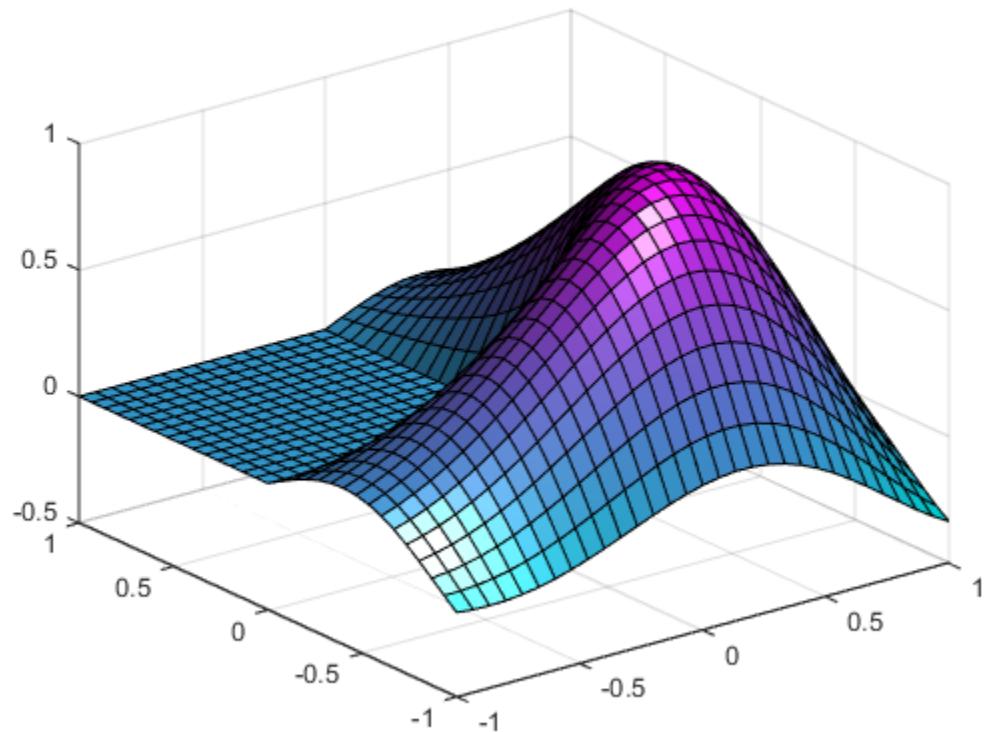
光照是为图形场景增加真实感的一种方式。它通过模拟自然光（例如来自太阳的定向光）照射下对象上出现的高光区和黑暗区来实现。为了产生光照效果，MATLAB 定义了一个图形对象，称为光源。MATLAB 将光照应用于曲面对象和补片对象。

示例 - 在场景中添加光源

此示例显示膜表面，并使用从相机位置右侧发出的光源照亮它。

```
membrane
camlight
```

创建光源将激活许多与光照相关的属性，它们控制诸如环境光和对象反射属性之类的特性。



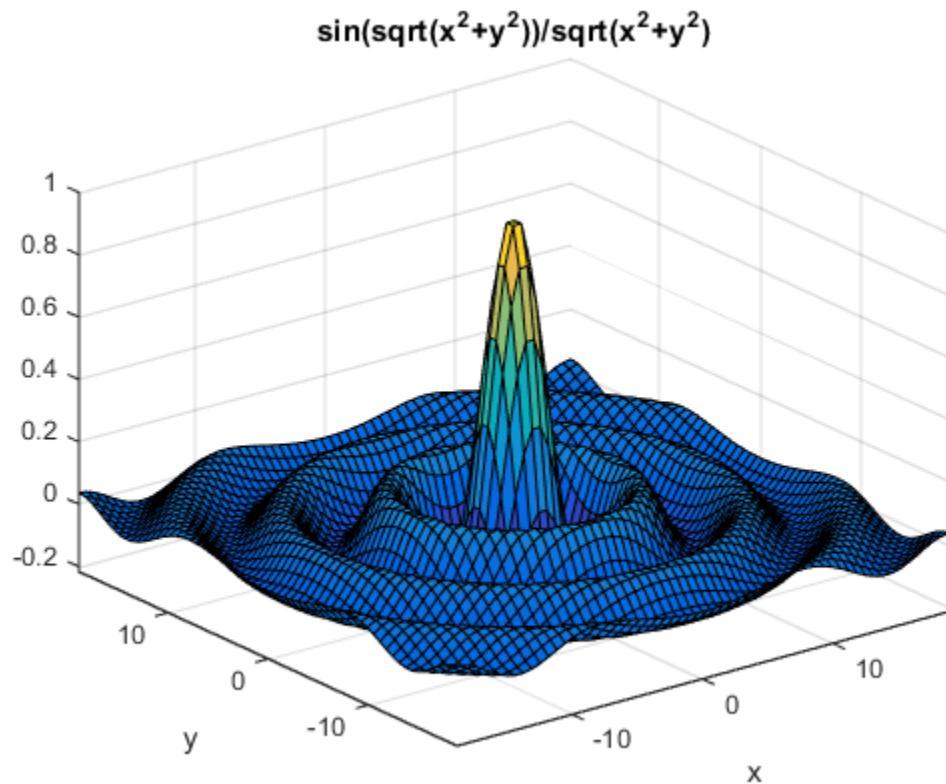
示例 - 数学函数光照效果

光照可以增强数学函数的曲面图的显示。例如，使用 `ezsurf` 命令计算表达式

$$\sin(\sqrt{x^2 + y^2}) / (\sqrt{x^2 + y^2})$$

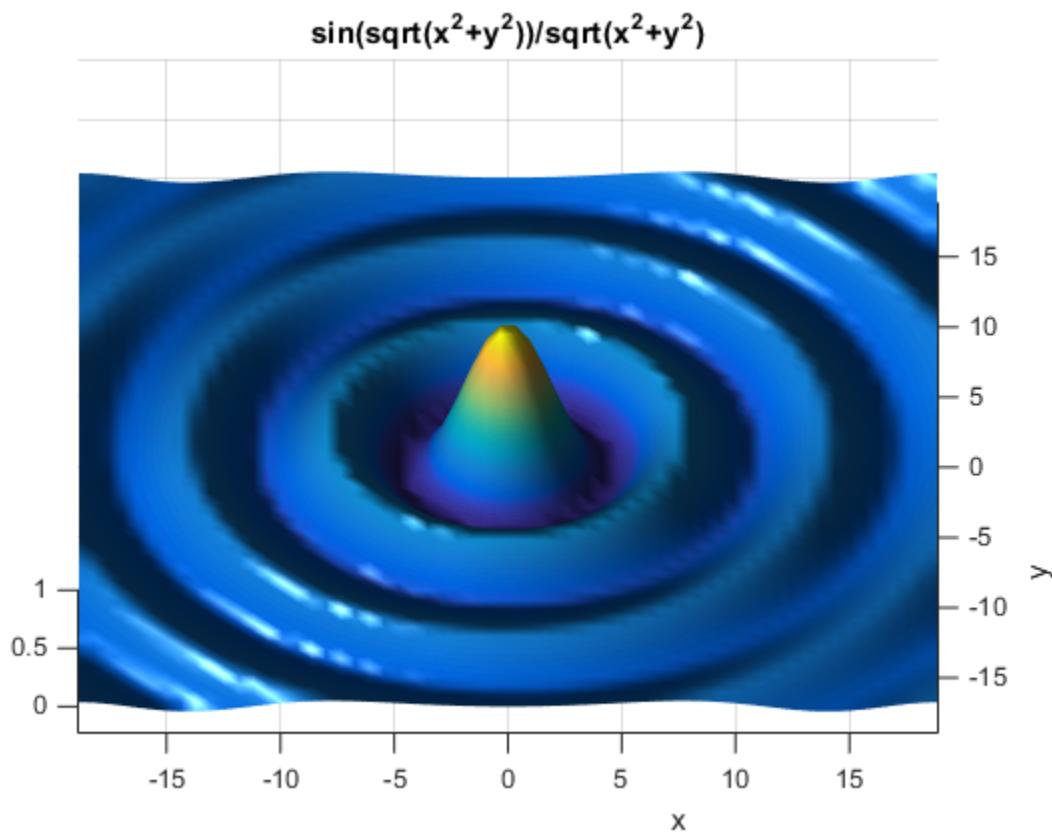
在 -6π 到 6π 区域内的值。

```
h = ezsurf('sin(sqrt(x^2+y^2))/sqrt(x^2+y^2)',[-6*pi,6*pi]);
```



现在使用 `lightangle` 函数添加光照，此函数接受以方位角和仰角表示的光照位置。

```
view(0,75)
shading interp
lightangle(-45,30)
h.FaceLighting = 'gouraud';
h.AmbientStrength = 0.3;
h.DiffuseStrength = 0.8;
h.SpecularStrength = 0.9;
h.SpecularExponent = 25;
h.BackFaceLighting = 'unlit';
```



使用 **findobj** 获取曲面对象的句柄后，可以设置影响曲面的反射光的属性。有关这些属性的详细说明，请参阅“影响光照的属性”（第 11-2 页）。

图形对象的反射特性

本节内容

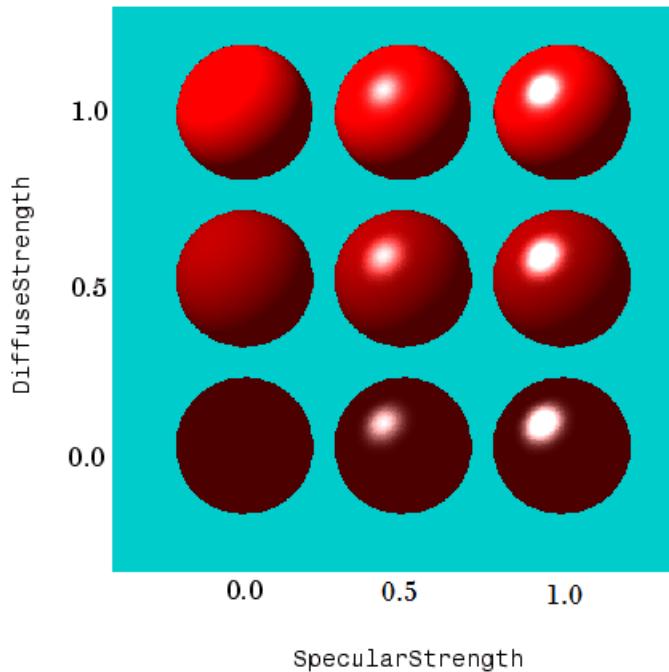
- “镜面反射和漫反射”（第 11-7 页）
- “环境光”（第 11-7 页）
- “镜面反射指数”（第 11-8 页）
- “镜面颜色反射”（第 11-8 页）
- “背面光照”（第 11-9 页）
- “在数据空间放置光源”（第 11-10 页）

镜面反射和漫反射

您可以指定补片和曲面对象的反射特性，使这些对象的外观在对场景应用光照后产生一些变化。您可能需要对这些特性进行综合调整，以产生特定的结果。

另请参阅 **material** 命令，以了解产生某些光照效果的便捷方式。

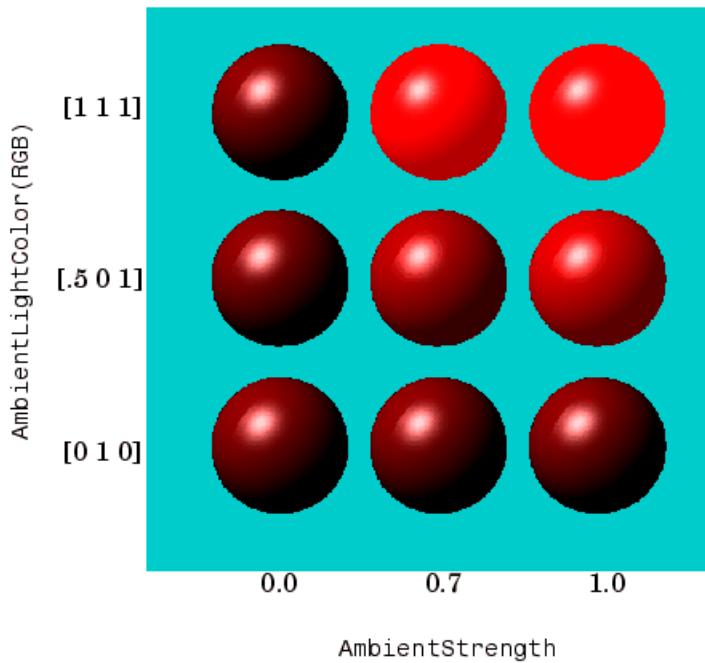
通过设置 **SpecularStrength** 和 **DiffuseStrength** 属性，您可以控制对象表面上的镜面反射和漫反射量。下图说明了这两种属性不同组合设置下的显示效果。



环境光

环境光是一种无向光，均匀地照射在场景中的所有对象上。仅当坐标区内存在光源对象时，环境光才可见。有两个控制环境光的属性 - **AmbientLightColor** 是坐标区属性，用来设置颜色；**AmbientStrength** 是补片和曲面对象的属性，用来确定环境光在特定对象上的照射强度。

下图显示三种不同的环境光在各种强度下的颜色。球体是红色，并且存在一个白光源对象。

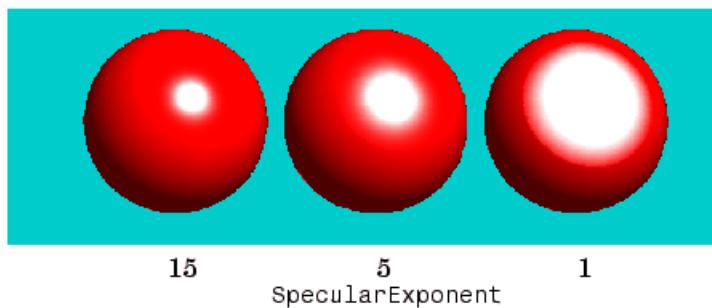


绿色 [0 1 0] 环境光不会影响场景，因为绿光中没有红光分量。而 RGB 值 [.5 0 1] 定义的颜色中由于有红光分量，因此会对球体上的光照产生影响，但其影响小于白色 [1 1 1] 环境光所产生的影响。

镜面反射指数

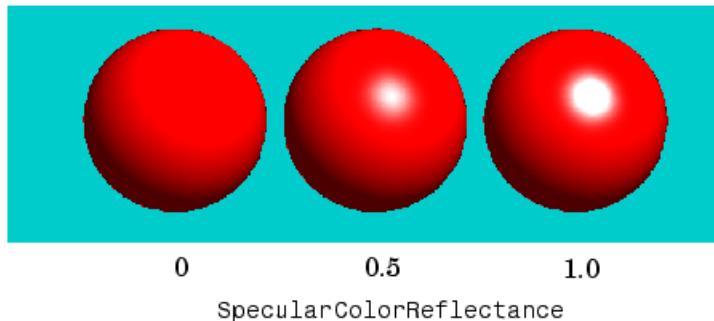
镜面反射光点的大小取决于补片和曲面对象的 **SpecularExponent** 属性值。此属性的取值范围为 1 到 500，对一般对象而言，其值范围在 5 到 20 之间。

下图显示根据三种不同的 **SpecularExponent** 属性值用白光照射红色球体的效果。



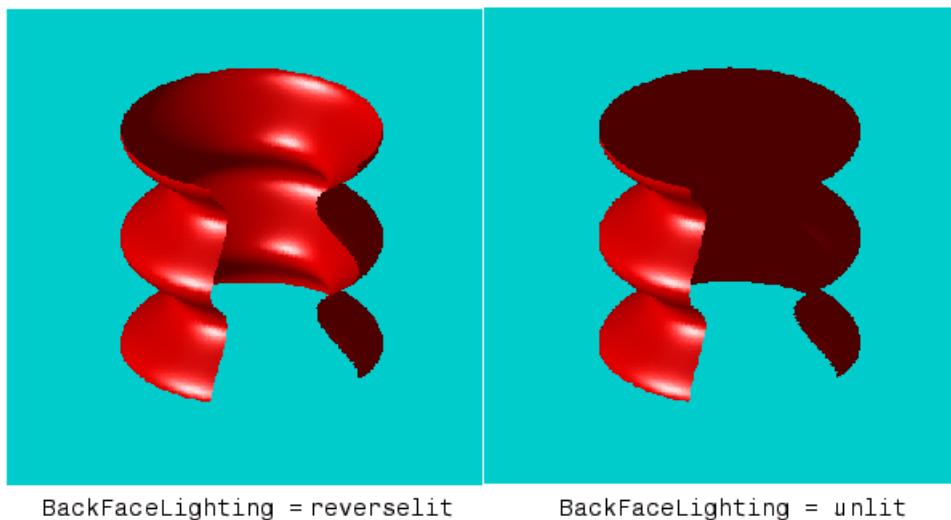
镜面颜色反射

镜面反射光的颜色可以在一定范围内变化，即从光源颜色与对象颜色的组合色到光源颜色本身。补片和曲面的 **SpecularColorReflectance** 属性控制此颜色。下图显示了白光照射下的红色球体。**SpecularColorReflectance** 属性的值从 0 (对象与光源的颜色) 到 1 (光源的颜色)。



背面光照

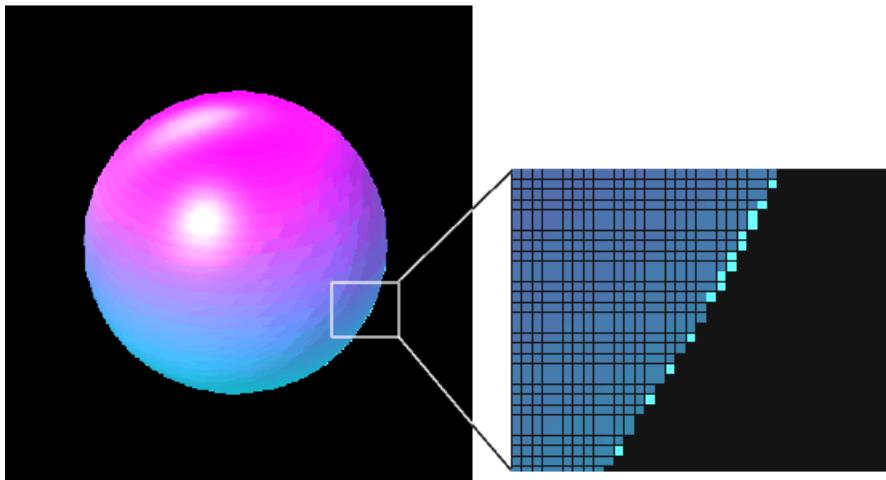
背面光照对于区别显示对象的内外表面非常有用。下面这些柱状曲面切割图说明了背面光照的效果。



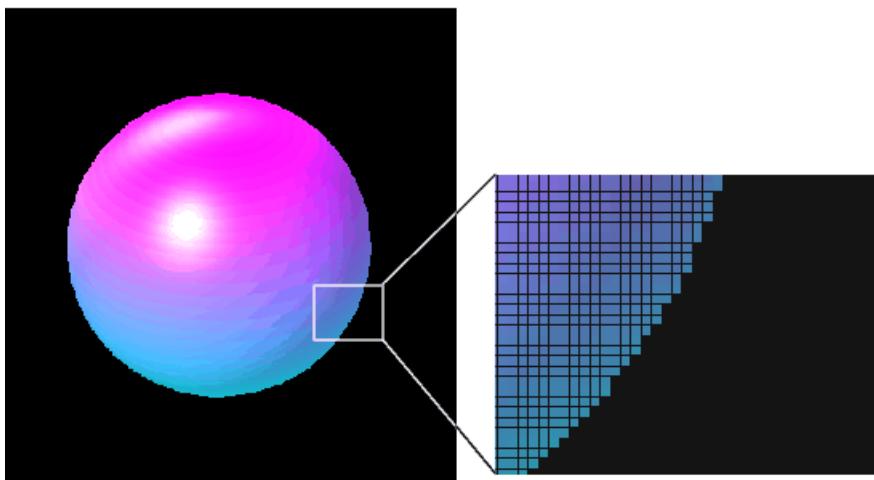
BackFaceLighting 的默认值为 **reverselit**。此设置会将顶点背离相机的法向量的方向反转，使光线可以照射到内曲面，再反射到相机。将 **BackFaceLighting** 设置为 **unlit** 会对法向量背离相机的面禁用光照。

您还可以使用 **BackFaceLighting** 来消除闭合对象上的边缘效果。当 **BackFaceLighting** 设置为 **reverselit** 时会产生这些效果，闭合对象边缘上的像素被照亮，就好像其顶点法线朝向相机一样。这会产生被错误照亮的像素，因为这些可见像素实际上是背离相机的。

为了说明这种效果，下图放大了被照亮的球体的边缘。将 **BackFaceLighting** 设置为 **lit** 可防止出现被错误照亮的像素。



```
BackFaceLighting = reverselit
```



```
BackFaceLighting = lit
```

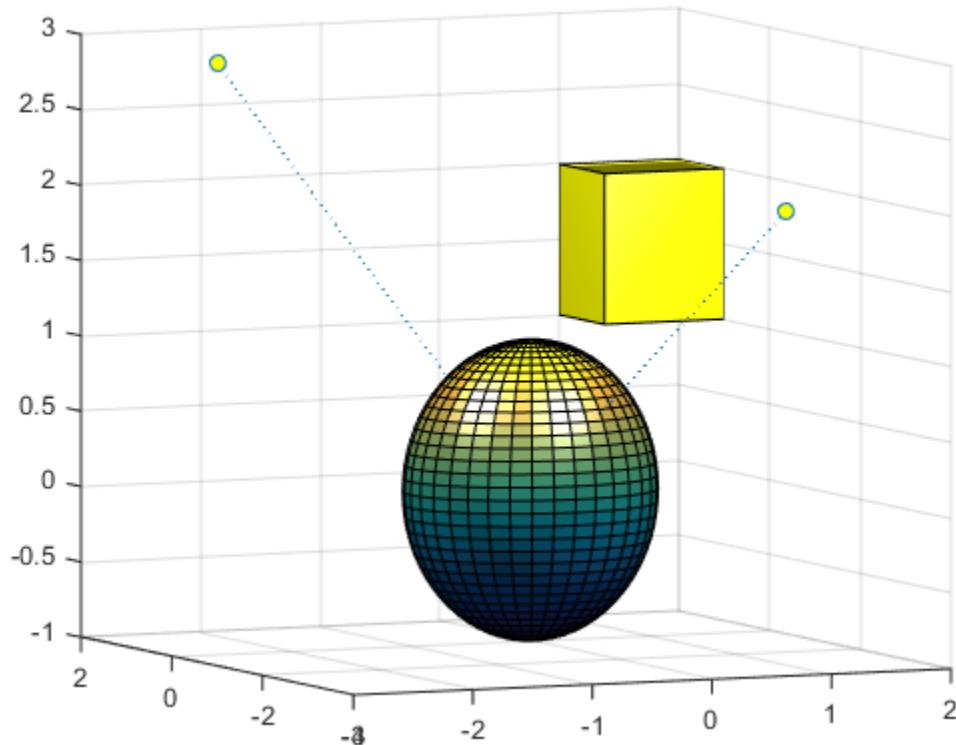
在数据空间放置光源

此示例创建一个球体和一个立方体，然后用两个光源照亮它们。光源对象被放在无穷远处，但方向由它们的位置向量指定。

```
% Create a sphere
sphere(36);
axis([-3 3 -3 3 -3 3])
hold on
% Create a cube
fac = [1 2 3 4;2 6 7 3;4 3 7 8;1 5 8 4;1 2 6 5;5 6 7 8];
vert = [1 1 1;1 2 1;2 2 1;2 1 1;1 1 2;1 2 2;2 2 2;2 1 2];
patch('faces',fac,'vertices',vert,'FaceColor','y');
```

```
% Add lights  
light('Position',[1 3 2]);  
light('Position',[-3 -1 3]);  
hold off
```

light 函数定义两个光源对象，它们位于无限远处，方向由 **Position** 向量指定。这些向量以坐标区坐标 [x, y, z] 方式定义。



透明度

- “为图形对象添加透明度” (第 12-2 页)
- “更改图像、填充或曲面的透明度” (第 12-9 页)
- “修改 alphamap” (第 12-16 页)

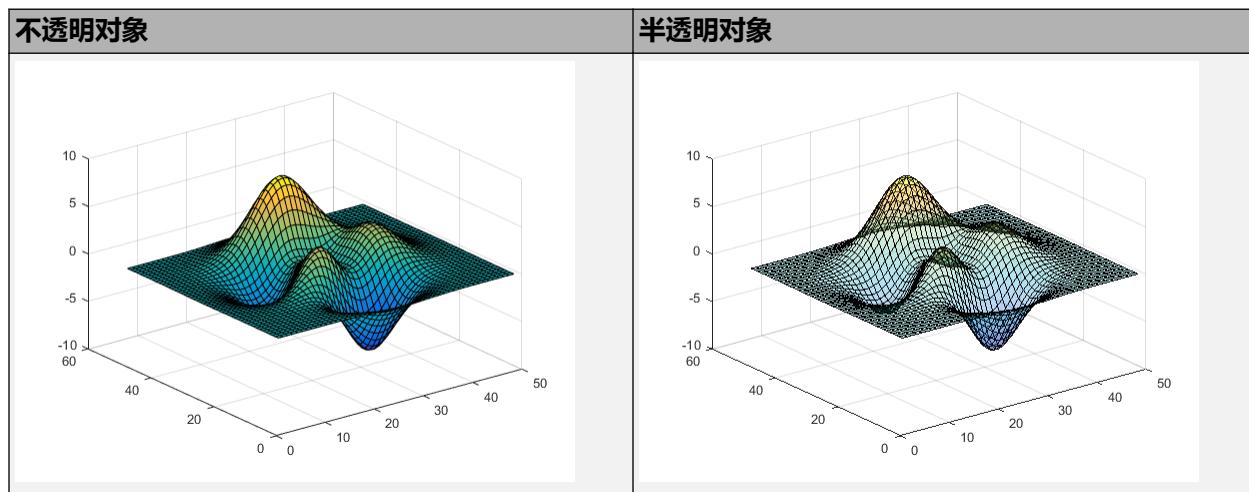
为图形对象添加透明度

本节内容

- “什么是透明度？”（第 12-2 页）
- “支持透明度的图形对象”（第 12-2 页）
- “创建具有透明度的区域图”（第 12-3 页）
- “创建具有透明度的条形图”（第 12-4 页）
- “创建具有透明度的散点图”（第 12-5 页）
- “使用 Alpha 数据更改透明度”（第 12-6 页）
- “更改曲面图透明度”（第 12-7 页）
- “更改补片对象透明度”（第 12-7 页）

什么是透明度？

图形对象的透明度决定您可以透视对象的程度。通过为图形对象添加透明度，您可以自定义图的外观，或者显示对象本来被隐藏的细节。下表显示了不透明曲面和半透明曲面之间的区别。



支持透明度的图形对象

可以使用 `alpha` 函数或通过设置对象的透明度属性来控制对象的透明度。有些图形对象支持对面和边使用不同的透明度值。

下表列出了支持透明度的对象及相应的属性。可将属性设置为 `[0,1]` 范围内的标量值。值 0 表示完全透明，值 1 表示完全不透明，0 和 1 之间的值表示半透明。

支持透明度的图形对象	实现统一透明度的属性
区域	<code>FaceAlpha</code> <code>EdgeAlpha</code>
条形序列	<code>FaceAlpha</code> <code>EdgeAlpha</code>

支持透明度的图形对象	实现统一透明度的属性
散点序列	MarkerFaceAlpha MarkerEdgeAlpha
气泡图序列	MarkerFaceAlpha MarkerEdgeAlpha
直方图	FaceAlpha
二元直方图	FaceAlpha
图曲面	FaceAlpha EdgeAlpha
基本曲面	FaceAlpha EdgeAlpha
补片	FaceAlpha EdgeAlpha
图像	AlphaData

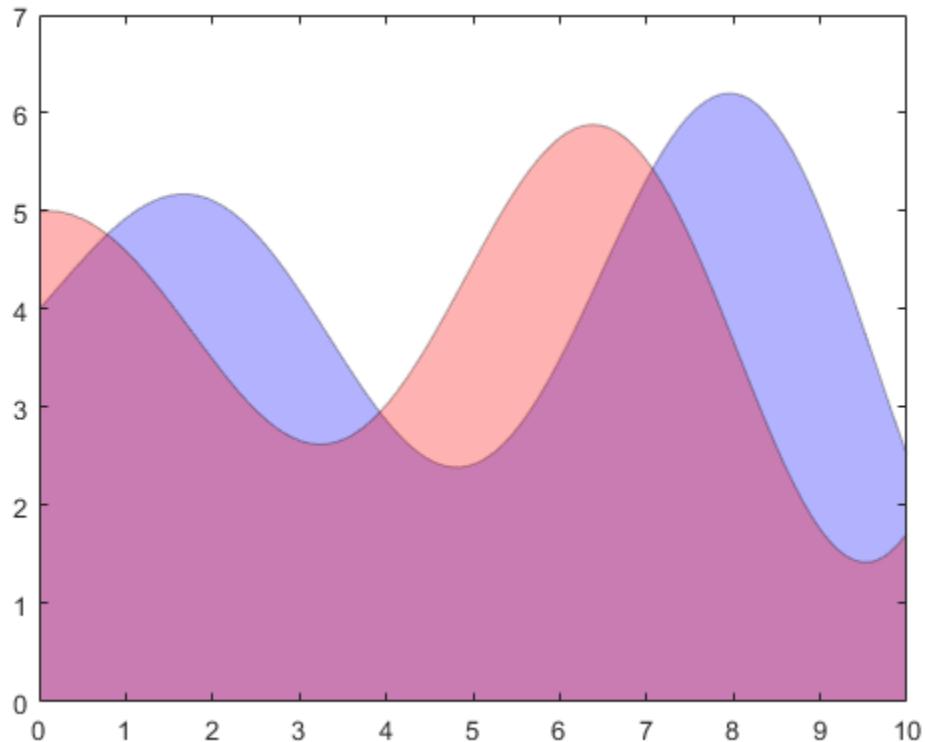
提示 补片、曲面、散点和图像对象支持使用 Alpha 数据来更改整个对象的透明度。有关详细信息，请参阅“使用 Alpha 数据更改透明度”（第 12-6 页）。

创建具有透明度的区域图

通过为每个区域对象设置 FaceAlpha 和 EdgeAlpha 属性，将两个半透明区域图合并在一起。

```
x = linspace(0,10);
y1 = 4 + sin(x).*exp(0.1*x);
area(x,y1,'FaceColor','b','FaceAlpha',.3,'EdgeAlpha',.3)

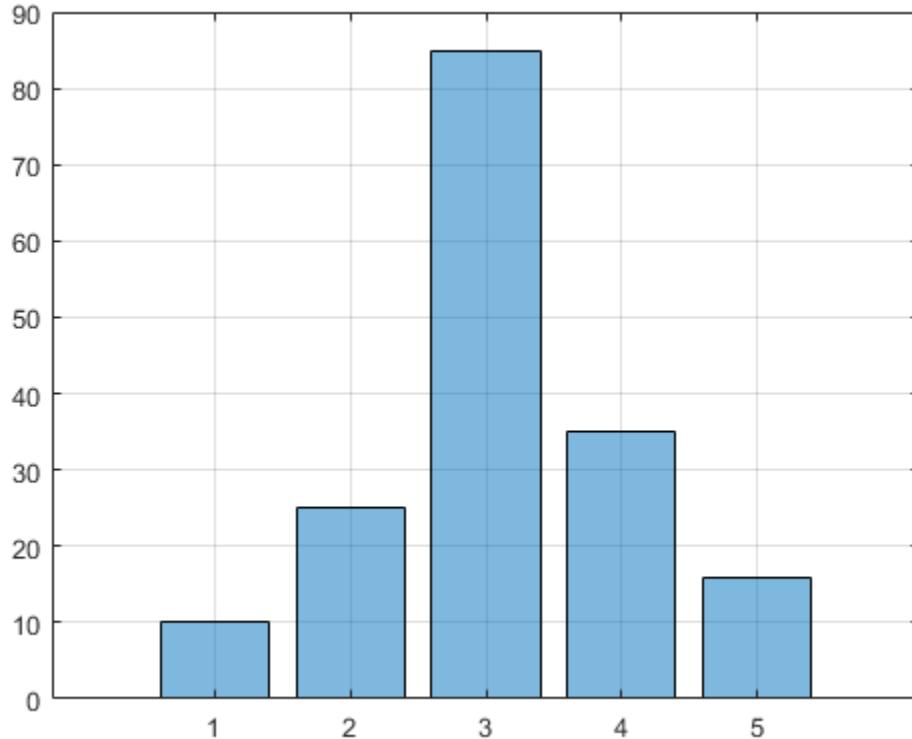
y2 = 4 + cos(x).*exp(0.1*x);
hold on
area(x,y2,'FaceColor','r','FaceAlpha',.3,'EdgeAlpha',.3)
hold off
```



创建具有透明度的条形图

通过将条形序列对象的 FaceAlpha 属性设置为介于 0 和 1 之间的值，创建半透明的条形图。网格线的显示方式。

```
month = 1:5;
sales = [10 25 85 35 16];
bar(month,sales,'FaceAlpha',.5)
grid on
```



创建具有透明度的散点图

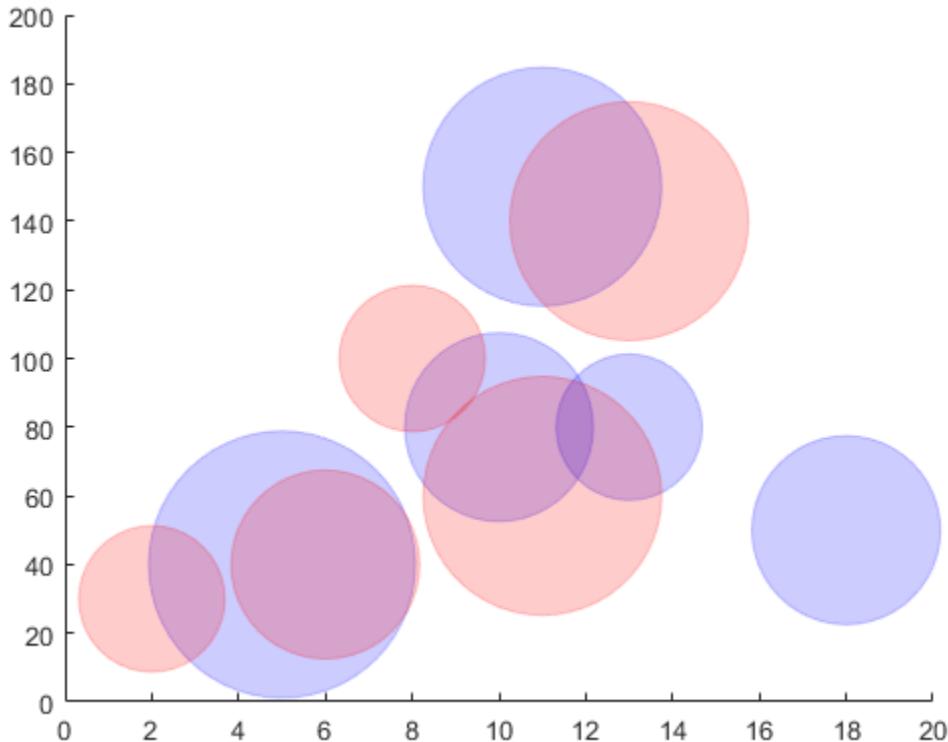
创建使用蓝色半透明标记的散点图。然后添加使用红色半透明标记的第二个散点图。通过设置散点序列对象的 `MarkerFaceColor` 和 `MarkerEdgeColor` 属性来指定标记颜色。通过将 `MarkerFaceAlpha` 和 `MarkerEdgeAlpha` 属性设置为介于 0 和 1 之间的标量值来指定透明度。

```

x = [5 10 11 13 18];
y1 = [40 80 150 80 50];
a1 = 100*[100 50 80 30 50];
scatter(x,y1,a1,'MarkerFaceColor','b','MarkerEdgeColor','b',...
    'MarkerFaceAlpha',.2,'MarkerEdgeAlpha',.2)
axis([0 20 0 200])

x = [2 6 8 11 13];
y2 = [30 40 100 60 140];
a2 = 100*[30 50 30 80 80];
hold on
scatter(x,y2,a2,'MarkerFaceColor','r','MarkerEdgeColor','r',...
    'MarkerFaceAlpha',.2,'MarkerEdgeAlpha',.2)
hold off

```



使用 Alpha 数据更改透明度

补片、曲面和图像对象有几个额外的属性可用来更改对象的透明度。

- 图像 - 为每个图像元素指定不同的透明度值。可通过将 **AlphaData** 属性设置为与 **CData** 属性大小相同的数组来指定其属性值。
- 图曲面和基本曲面 - 为每个面和边指定不同的透明度值。此外，还可以指定对每个面或边是使用单一透明度还是插补透明度。首先，通过将 **AlphaData** 属性设置为与 **ZData** 属性大小相同的数组来指定透明度值。然后，通过将 **FaceAlpha** 和 **EdgeAlpha** 属性设置为 '*flat*' 或 '*interp*' 来指定单一透明度或插补透明度。
- 补片 - 为每个面和边指定不同的透明度值。此外，还可以指定对每个面或边是使用单一透明度还是插补透明度。首先，通过将 **FaceVertexAlphaData** 属性设置为长度等于补片的面数（对于单一透明度）或顶点数（对于插补透明度）的列向量来指定透明度值。然后，通过将 **FaceAlpha** 和 **EdgeAlpha** 属性设置为 '*flat*' 或 '*interp*' 来指定单一透明度或插补透明度。
- 散点图 - 为每个标记指定不同的透明度值。首先，通过将 **AlphaData** 属性设置为与 **XData** 属性大小相同的数组来指定透明度值。然后，通过将 **MarkerFaceAlpha** 或 **MarkerEdgeAlpha** 属性设置为 '*flat*' 来指定单一透明度。

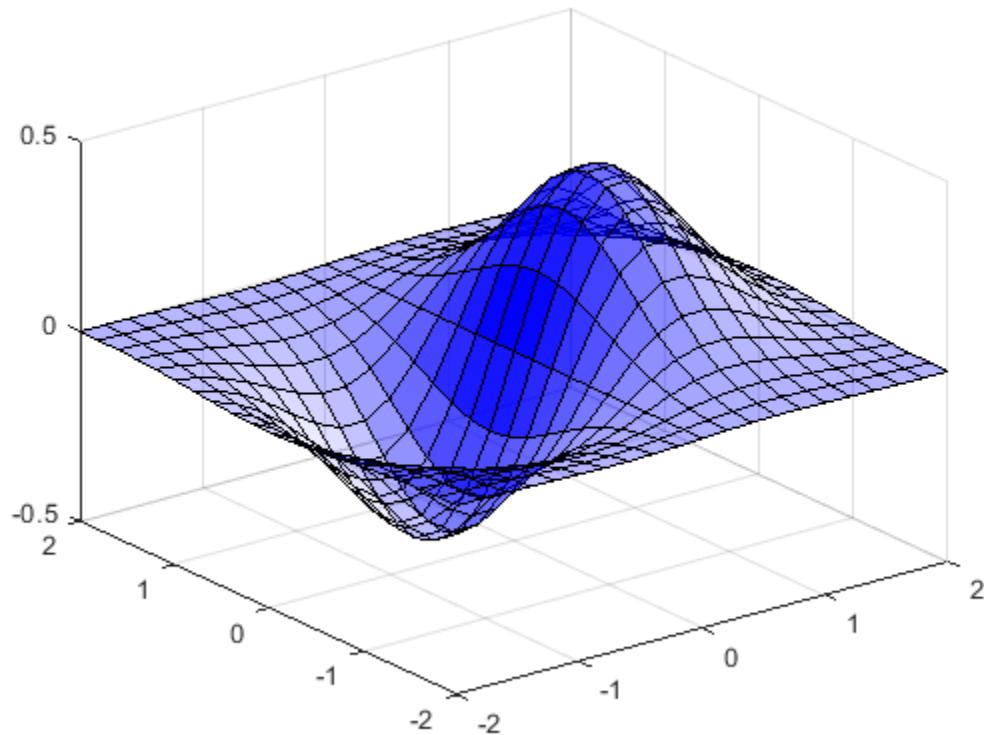
可以使用 **AlphaDataMapping** 属性控制对象如何解释 Alpha 数据值。有关详细信息，请参阅属性说明。

更改曲面图透明度

创建一个曲面并根据 z 数据的梯度更改透明度。通过将 FaceAlpha 设置为 'flat'，对曲面的每个面使用单一透明度。将曲面颜色设置为蓝色，以显示透明度如何变化。

```
[x,y] = meshgrid(-2:.2:2);
z = x.*exp(-x.^2-y.^2);
a = gradient(z);

surf(x,y,z,'AlphaData',a,...  
'FaceAlpha','flat',...  
'FaceColor','blue')
```



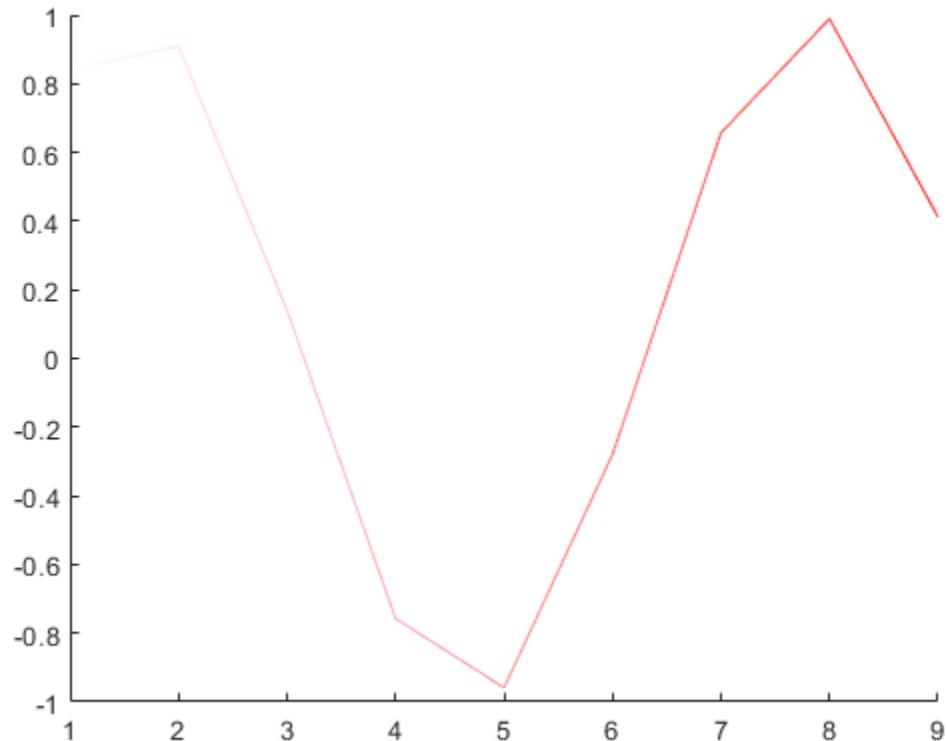
更改补片对象透明度

使用 patch 函数绘制一个线条。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。

通过将 FaceVertexAlphaData 属性设置为一个列向量，为每个顶点定义一个透明度值。通过将 AlphaDataMapping 属性设置为 'none'，将这些值解释为透明度值（0 表示完全透明，1 表示完全不透明）。通过将 EdgeAlpha 属性设置为 'interp'，在顶点之间进行透明度插值。

```
x = linspace(1,10,10);
y = sin(x);
y(end) = NaN;
```

```
figure  
alpha_values = linspace(0,1,10);  
patch(x,y,'red','EdgeColor','red',...
'FaceVertexAlphaData',alpha_values,'AlphaDataMapping','none',...
'EdgeAlpha','interp')
```



另请参阅

[alpha](#) | [alphamap](#) | [alim](#) | [scatter](#) | [bar](#) | [image](#) | [surf](#) | [patch](#) | [area](#)

更改图像、填充或曲面的透明度

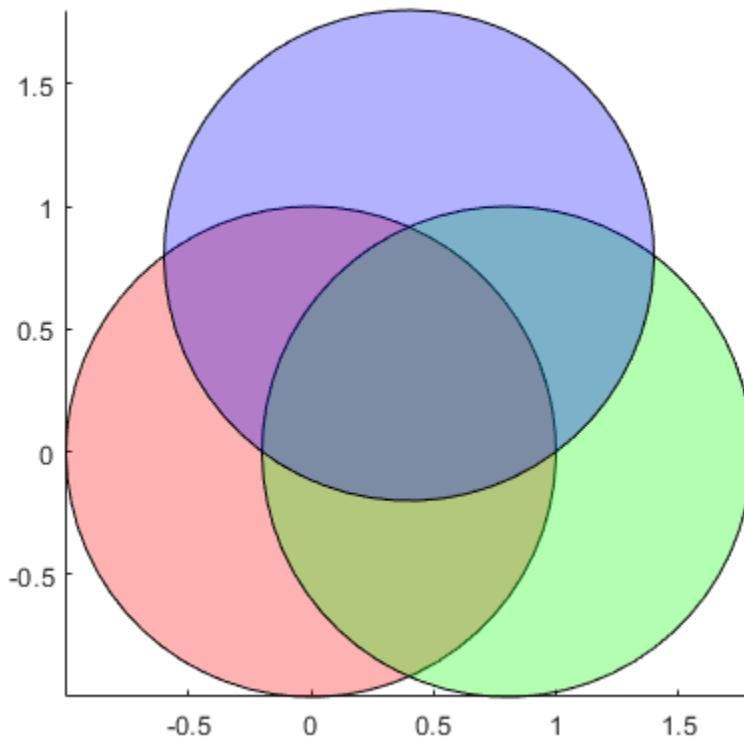
此示例说明如何修改图像、填充或曲面的透明度。

坐标区框中所有对象的透明度

透明度值称为 **alpha** 值。使用 **alpha** 函数设置当前坐标区范围内所有图像、填充或曲面对象的透明度。指定一个介于 0 (完全透明) 和 1 (完全不透明) 之间的透明度值。

```
t = 0:0.1:2*pi;
x = sin(t);
y = cos(t);

figure
patch(x,y,'r')
patch(x+0.8,y,'g')
patch(x+0.4,y+0.8,'b')
axis square tight
alpha(0.3)
```

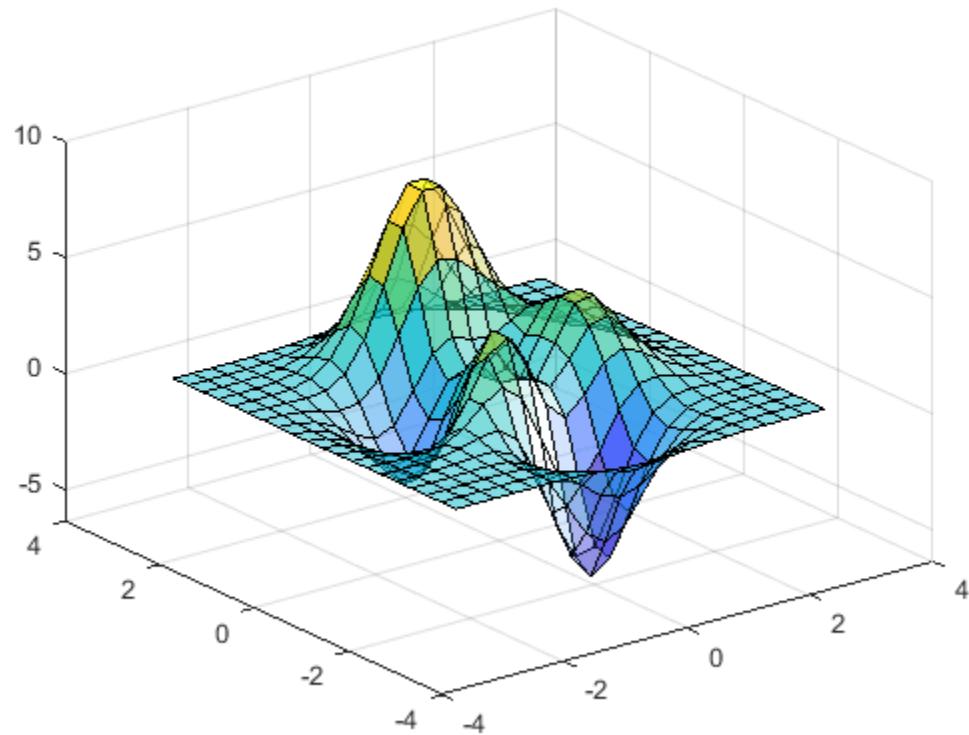


个别曲面的透明度

曲面的透明度由其 **AlphaData** 属性定义。将 **alpha** 数据设置为用于指定曲面的每个顶点透明度的标量值或值矩阵。**FaceAlpha** 属性指示如何从顶点透明度确定曲面透明度。

```
[X,Y,Z] = peaks(20);
s2 = surf(X,Y,Z);
```

```
s2.AlphaData = gradient(Z);
s2.FaceAlpha = 'flat';
```

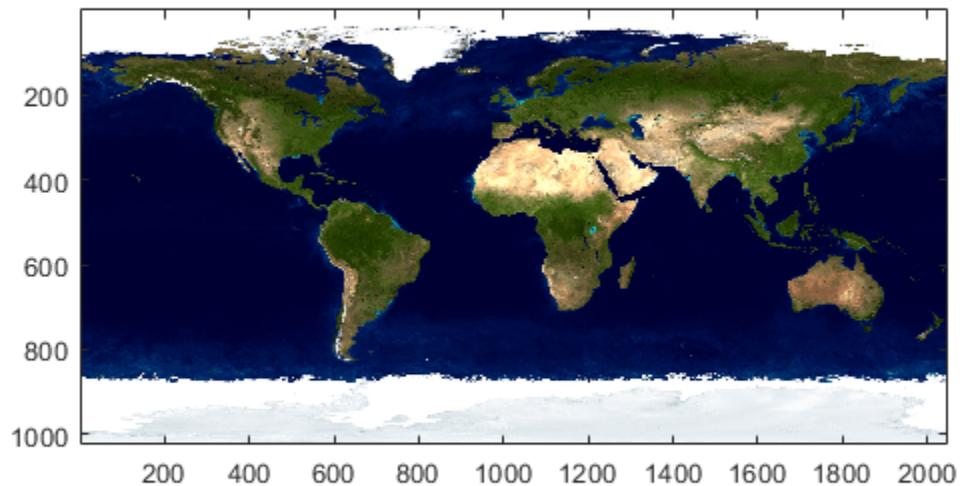


个别图像的透明度

与曲面一样，图像的透明度也由其 `AlphaData` 属性定义。对于图像，将 `alpha` 数据设置为用于指定图像数据的每个元素透明度的标量值或值矩阵。

例如，使用透明度覆盖两个图像。首先，显示地球的图像。

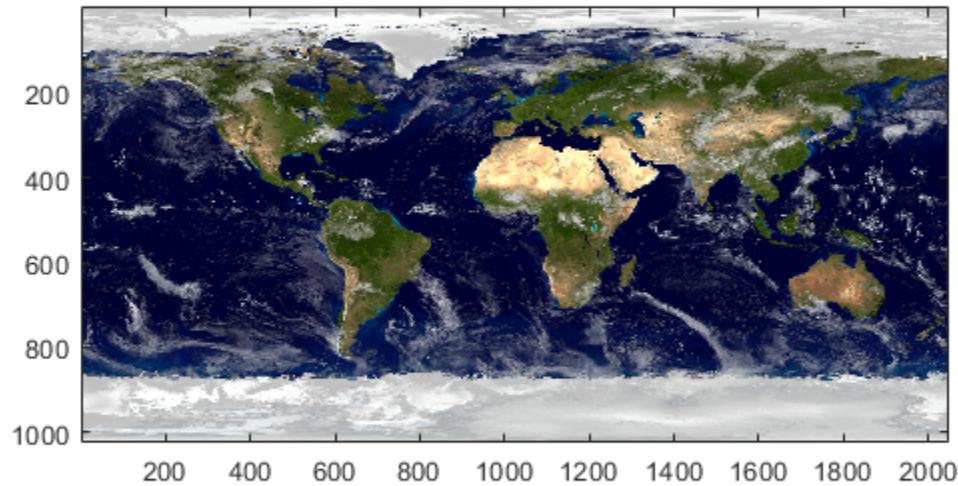
```
earth = imread('landOcean.jpg');
image(earth)
axis image
```



然后，使用透明度将云图层添加到地球图像。

```
clouds = imread('cloudCombined.jpg');
image(earth)
axis image
hold on

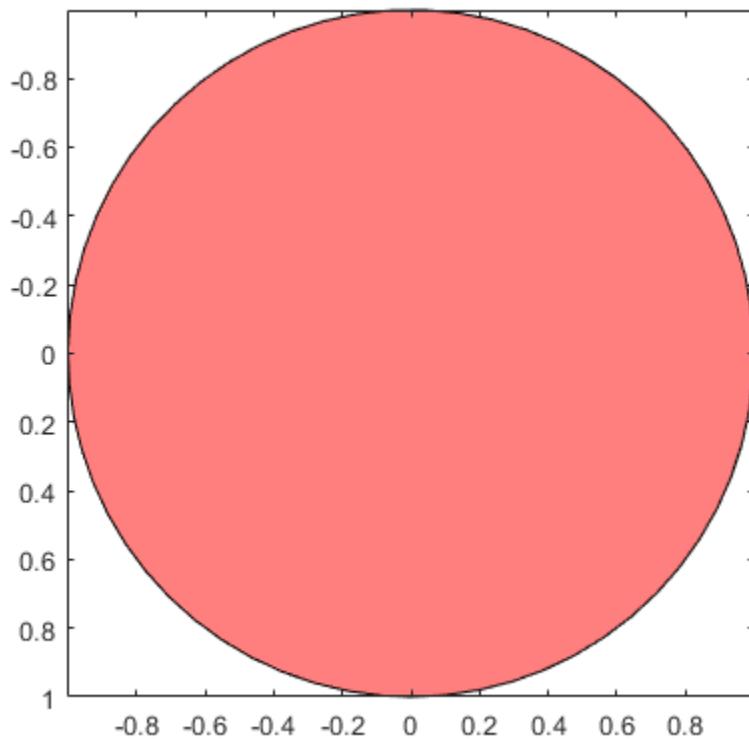
im = image(clouds);
im.AlphaData = max(clouds,[],3);
hold off
```



个别填充的透明度

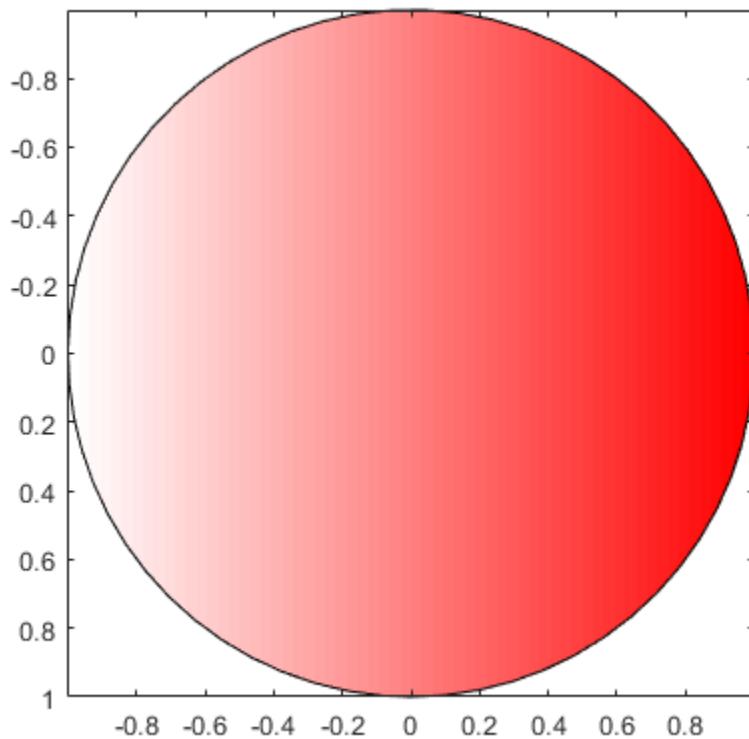
填充的透明度由其 FaceAlpha 和 FaceVertexAlphaData 属性定义。若要在整个填充上实现单一的透明度，请将 FaceVertexAlphaData 设置为一个介于 0 (完全透明) 和 1 (完全不透明) 之间的常量，并将 FaceAlpha 属性设置为 'flat'。

```
cla  
p1 = patch(x,y,'r');  
axis square tight  
p1.FaceVertexAlphaData = 0.2;  
p1.FaceAlpha = 'flat';
```



若要在整个填充上实现可变的透明度，请将 FaceVertexAlphaData 设置为用于指定填充的每个顶点或每个面的透明度的值矩阵。然后，通过 FaceAlpha 属性指示如何使用 FaceVertexAlphaData 确定面的透明度。如果为顶点指定了 alpha 数据，则必须将 FaceAlpha 设置为 'interp'。

```
p1.FaceVertexAlphaData = x';  
p1.FaceAlpha = 'interp';
```



包含纹理映射的透明度

纹理映射将二维图像映射到三维曲面上。通过将 **CData** 属性设置为图像数据并将 **FaceColor** 属性设置为 '**texturemap**'，可将图像映射到曲面上。

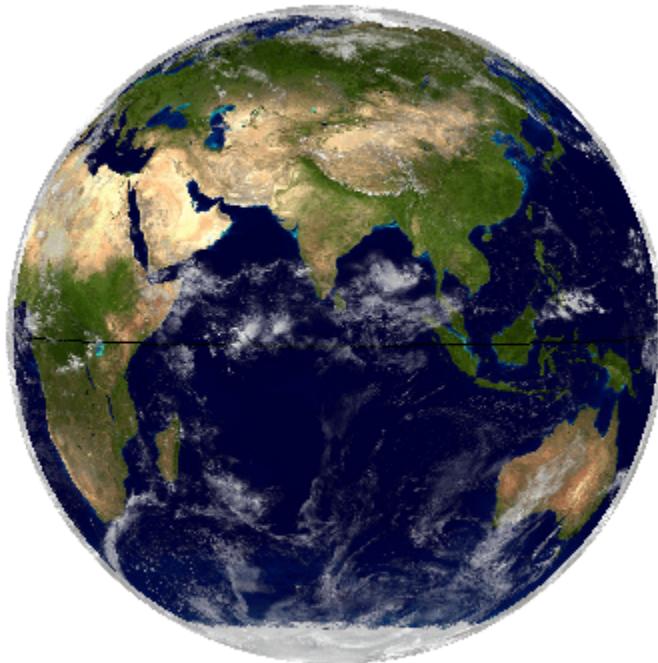
此示例创建地球和云的三维视图。它创建球形表面，并使用纹理映射将地球和云的图像映射到曲面上。

```
[px,py,pz] = sphere(50);

sEarth = surface(py, px ,flip(pz));
sEarth.FaceColor = 'texturemap';
sEarth.EdgeColor = 'none';
sEarth.CData = earth;
hold on
sCloud = surface(px*1.02,py*1.02,flip(pz)*1.02);

sCloud.FaceColor = 'texturemap';
sCloud.EdgeColor = 'none';
sCloud.CData = clouds;

sCloud.FaceAlpha = 'texturemap';
sCloud.AlphaData = max(clouds,[],3);
hold off
view([80 2])
daspect([1 1 1])
axis off tight
```



此示例中使用的图像来自 Visible Earth。

致谢：美国国家航空航天局戈达德太空飞行中心图像，由 Reto Stöckli 拍摄（陆地表面、浅水、云）。
Robert Simmon 提供增强效果（海洋颜色、合成、三维地球仪、动画）。数据和技术支持：MODIS 土地组；MODIS 科学数据支持团队；MODIS 大气组；MODIS 海洋组附加数据：美国地质调查局 EROS 数据中心（地貌）；美国地质勘探局地球遥感弗拉格斯塔夫球场中心（南极洲）；美国国防气象卫星计划（城市灯光）。

另请参阅

[alpha](#) | [alphamap](#) | [alim](#)

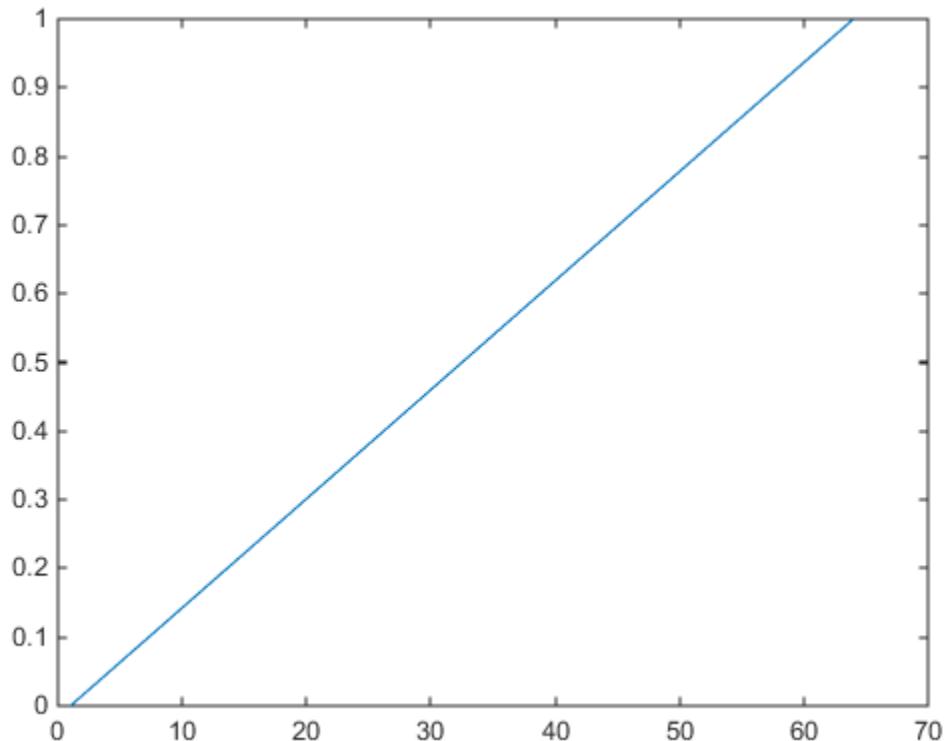
修改 alphamap

每个图窗都有一个关联的 alphamap，它是一组 0 到 1 之间的向量值。默认的 alphamap 包含 0 到 1 范围内的 64 个线性变化值。可以使用图窗的 **Alphamap** 属性或使用 **alphamap** 函数来查看或修改 alphamap。

默认的 alpha 映射

默认的 alphamap 包含 0 到 1 范围内的 64 个线性变化值，如下图所示。

```
am = get(gcf,'Alphamap');
plot(am)
```



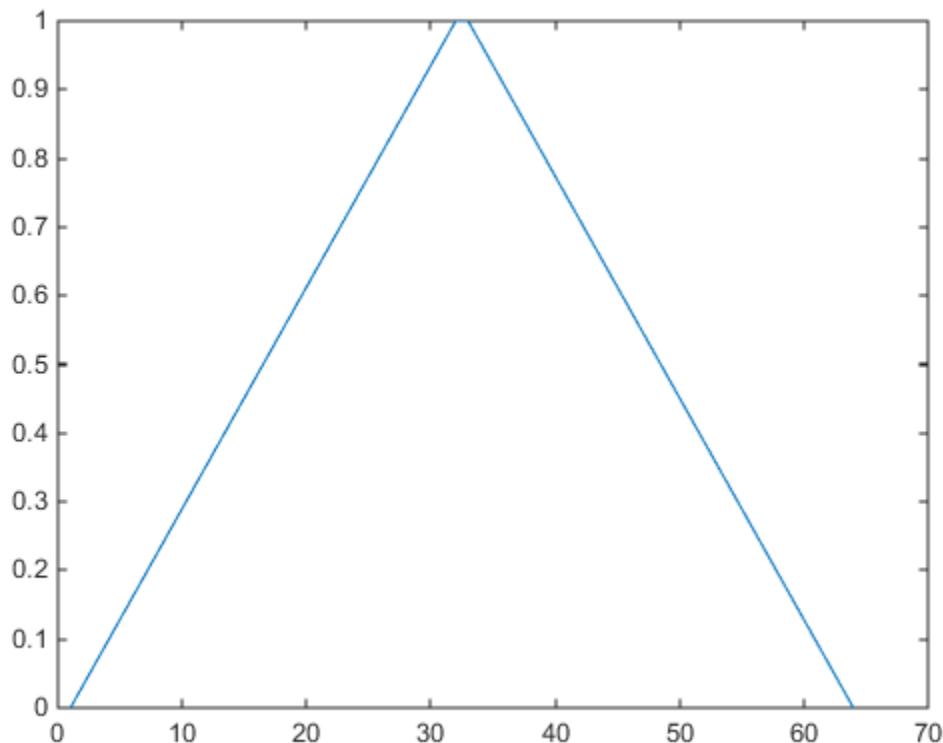
此 alphamap 将最低的 alpha 数据值显示为完全透明，将最高的 alpha 数据值显示为完全不透明。

alphamap 函数可以创建一些有用的预定义 alphamap，还可以修改现有 alphamap。例如，

```
figure;
alphamap('vup')
```

将图窗的 **Alphamap** 属性设置为值先增后减的 alphamap：

```
am = get(gcf,'Alphamap');
plot(am)
```

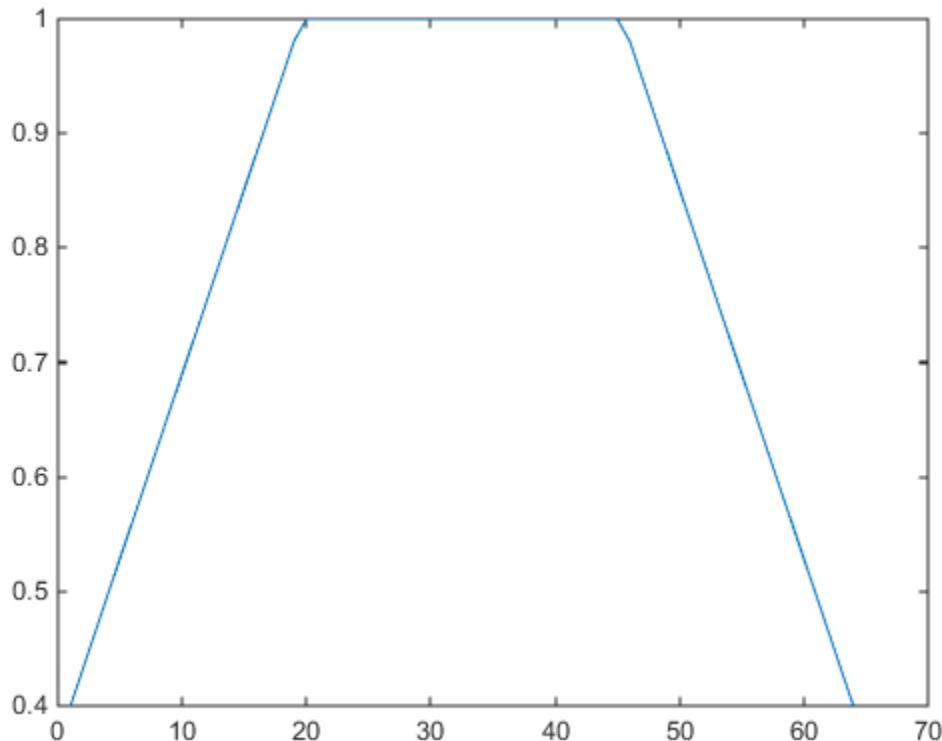


您可以使用 `increase` 或 `decrease` 选项改变这些值。例如，

```
alphamap('increase',.4)
```

将图窗当前 alphamap 中的所有值加上 0.4。重新绘制 'vup' alphamap 可以看出变化。这些值限制在 [0 1] 的范围内。

```
am = get(gcf,'Alphamap');
plot(am)
```



示例 - 修改 alphamap

此示例使用切片平面来查看三维体数据。切片平面使用颜色数据作为 alpha 数据，并使用递减的 alphamap（值范围从 1 到 0）：

- 1 通过计算一个包含三个变量的函数来创建三维体数据。

```
[x,y,z] = meshgrid(-1.25:.1:-.25,-2:.2:2,-2:.1:2);
v = x.*exp(-x.^2-y.^2-z.^2);
```

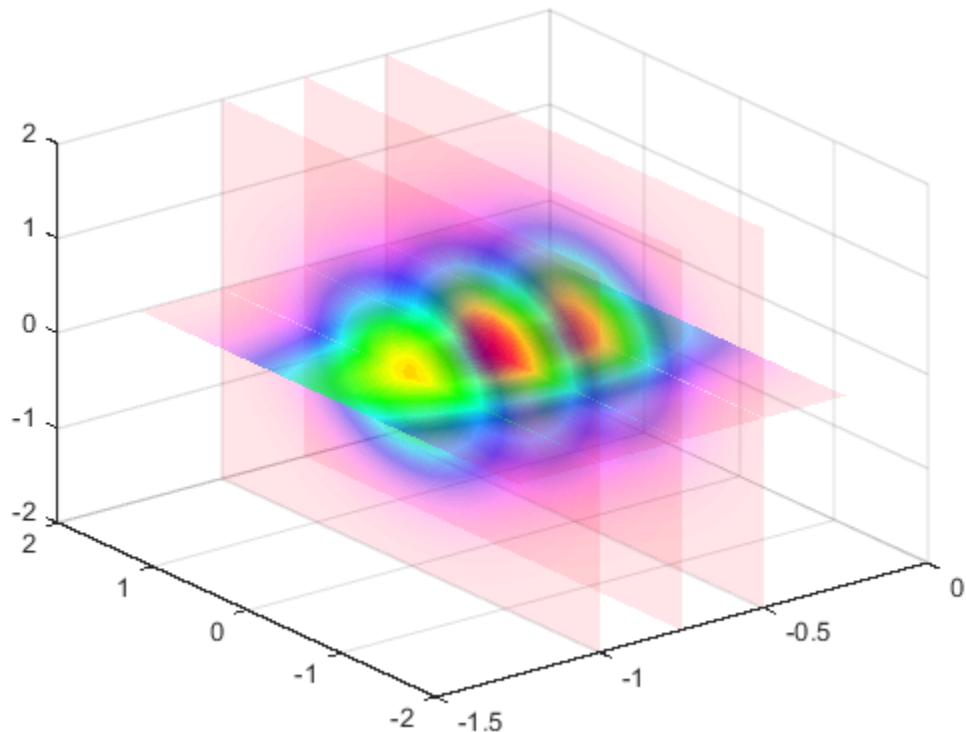
- 2 创建切片平面，将 alpha 数据设置为等于颜色数据，并指定 FaceColor 和 FaceAlpha 插值。

```
h = slice(x,y,z,v,[-1 -.75 -.5],[],[0]);
set(h,'EdgeColor','none',...
'FaceColor','interp',...
'FaceAlpha','interp')
alpha('color')
```

- 3 使用不透明度线性递减的 alphamap 并通过将 alphamap 中的每个值增加 0.1 来实现所需的透明度。指定 hsv 颜色图。

```
alphamap('rampdown')
alphamap('increase',.1)
colormap hsv
```

此 alphamap 用最小的透明度显示函数的最小值（接近零），用最大的透明度显示函数的最大值。这使您能够透视切片平面，同时保留了零附近的数据。



另请参阅

相关示例

- “[为图形对象添加透明度](#)” (第 12-2 页)

数据探查

- “交互式探查绘图数据” (第 13-2 页)
- “创建自定义数据提示” (第 13-6 页)
- “更改数据后自动刷新图” (第 13-9 页)
- “对图的交互进行控制” (第 13-12 页)

交互式探查绘图数据

您可以交互式探查和编辑绘图数据，以改善数据的视觉效果或显示有关数据的其他信息。可用的交互操作取决于坐标区的内容，但通常包括缩放、平移、旋转、数据提示、数据刷亮以及还原原始视图。

有些类型的交互可通过坐标区工具栏使用。将鼠标悬停在图区域上时，工具栏会显示在坐标区的右上角。



其他类型的交互内置于坐标区中并且对应于手势，例如拖动平移或滚动缩放。这些交互与坐标轴工具栏中的交互是分开的。

注意 在 R2018a 和之前的版本中，交互选项显示在图窗工具栏而不是坐标区工具栏上。此外，在以前的版本中，坐标区中没有内置任何基于手势的交互。

缩放、平移和旋转数据

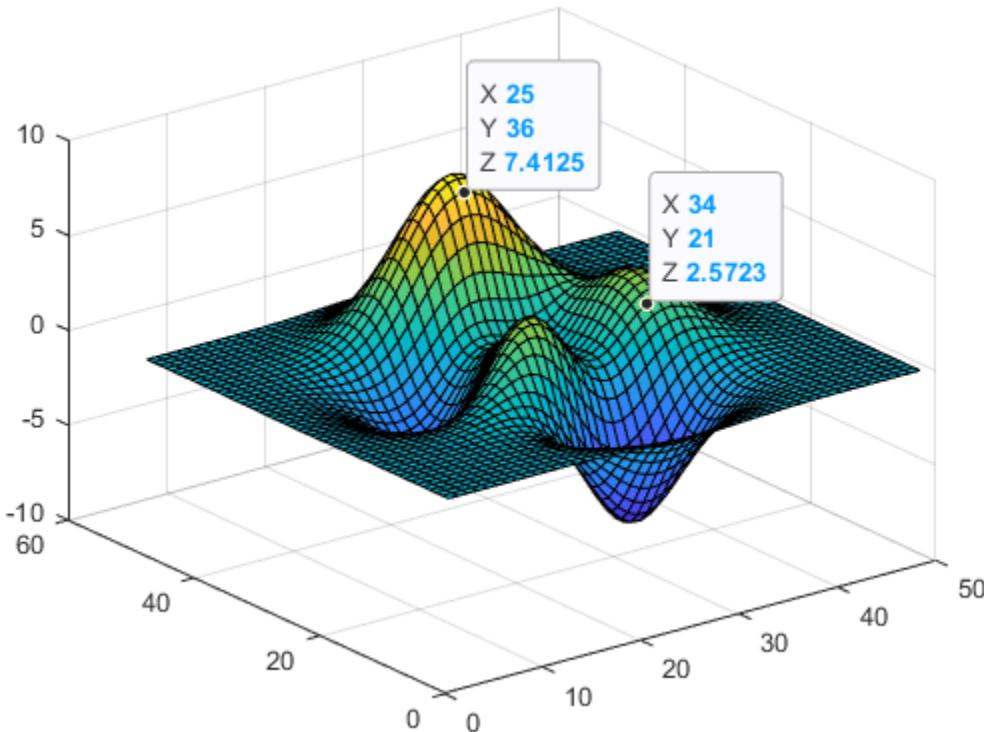
通过缩放、平移和旋转坐标区，可以从不同的角度探查数据。默认情况下，可以通过滚动或手指开合放大和缩小坐标区视图。此外，还可以拖动平移（二维视图）或拖动旋转（三维视图）。

通过点击坐标区工具栏中的放大 、缩小 、平移 和旋转 按钮，可以启用更多交互操作。例如，如果要拖动鼠标绘制一个矩形以放大感兴趣的区域，请点击放大按钮。

使用数据提示显示数据值

要标识图中数据点的值，请创建数据提示。当您将鼠标悬停在图中的数据点上时，数据提示会临时显示。

要使数据提示一直（固定）显示，可以点击数据点。或者，选择坐标区工具栏中的数据提示按钮 ，然后点击一个数据点。要使用数据提示按钮固定显示多个数据提示，请按住 **Shift** 键。要将数据提示放在与其重叠的其他数据提示之前，请点击它。



注意 在 MATLAB Online™ 中，数据提示交互性体验可能会有一些不同。例如，在某些情况下，您无法通过点击数据提示将其放在与其重叠的其他数据提示之前。

使用数据刷亮功能选择和修改数据值

可以使用数据刷亮功能选择、删除或替换单个数据值。要刷亮数据，请从坐标区工具栏上选择数据刷亮按钮 。点击某个数据点以突出显示它，或者拖动鼠标绘制一个矩形，以突出显示该矩形内的所有数据点。要突出显示更多数据点，请使用 **Shift** 键。

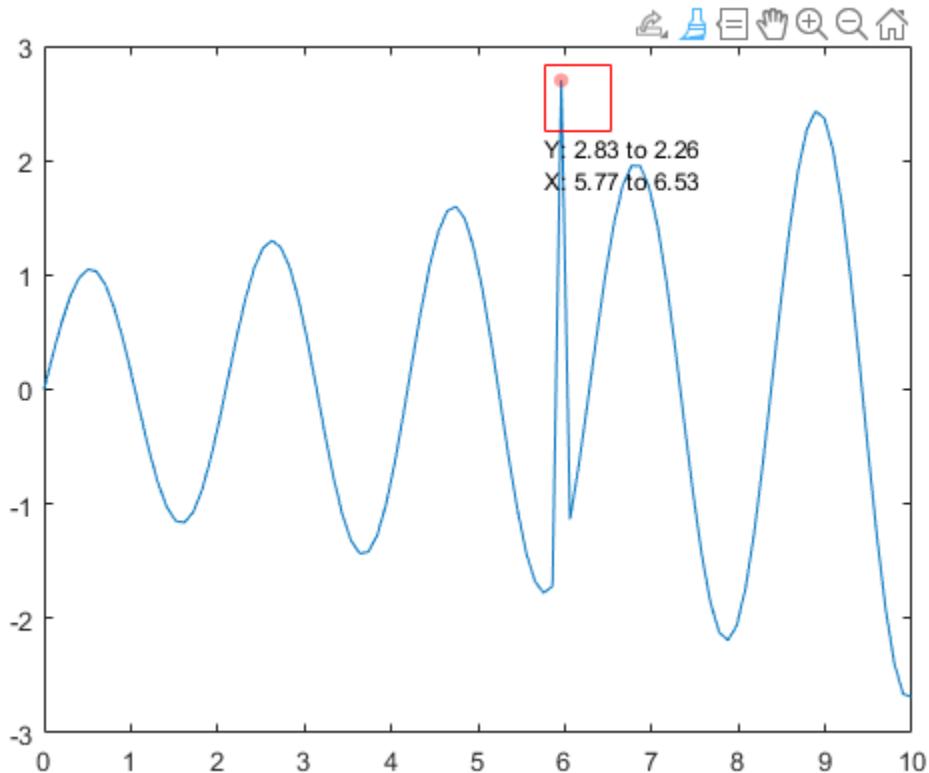
突出显示所需的数据点后，可以使用右键点击上下文菜单中的选项来删除、替换或复制值。在显示的图中，会显示您所做的更改。此外，更新绘制的对象后，您会看到数据属性（例如 **XData**）也相应地更新。但是，原始工作区变量不会更新。这种情况下，如果您还想更新工作区变量，则可以使用图窗的**工具**菜单上的**链接**选项，将变量与图链接起来。

从绘图数据中删除离群值

此示例说明如何使用数据刷亮功能从包含 100 个数据点的图中删除离群值。

首先，绘制包含一个离群值的数据。然后，从坐标区工具栏上选择数据刷亮按钮 ，并在离群值周围拖动鼠标绘制一个矩形。

```
x = linspace(0,10);
y = exp(.1*x).*sin(3*x);
y(60) = 2.7;
plot(x,y)
```

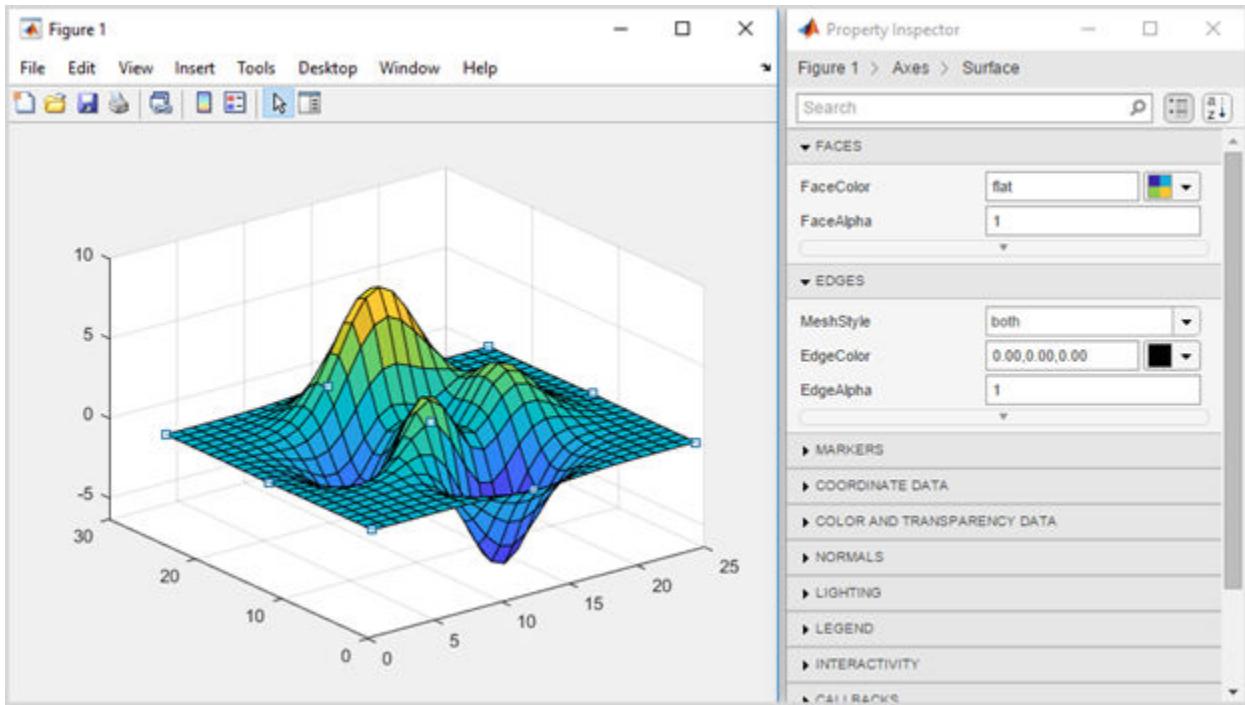


右键点击刷亮的数据点，然后从上下文菜单中选择删除。请注意，图将会更新。但是，工作区变量没有变化。

如果要从工作区变量中删除该点，请在刷亮数据之前从图窗的工具菜单中选择链接选项。

使用属性检查器自定义图

您可以使用**属性检查器**以交互方式修改图。打开属性检查器并选择图后，检查器将显示可以编辑的属性列表。要打开检查器，请使用 `inspect` 函数或点击图窗工具栏上的属性检查器按钮 。



另请参阅

[brush](#) | [datacursormode](#) | [rotate3d](#) | [pan](#) | [zoom](#) | [linkdata](#) | [属性检查器](#)

详细信息

- “更改数据后自动刷新图” (第 13-9 页)
- “创建自定义数据提示” (第 13-6 页)

创建自定义数据提示

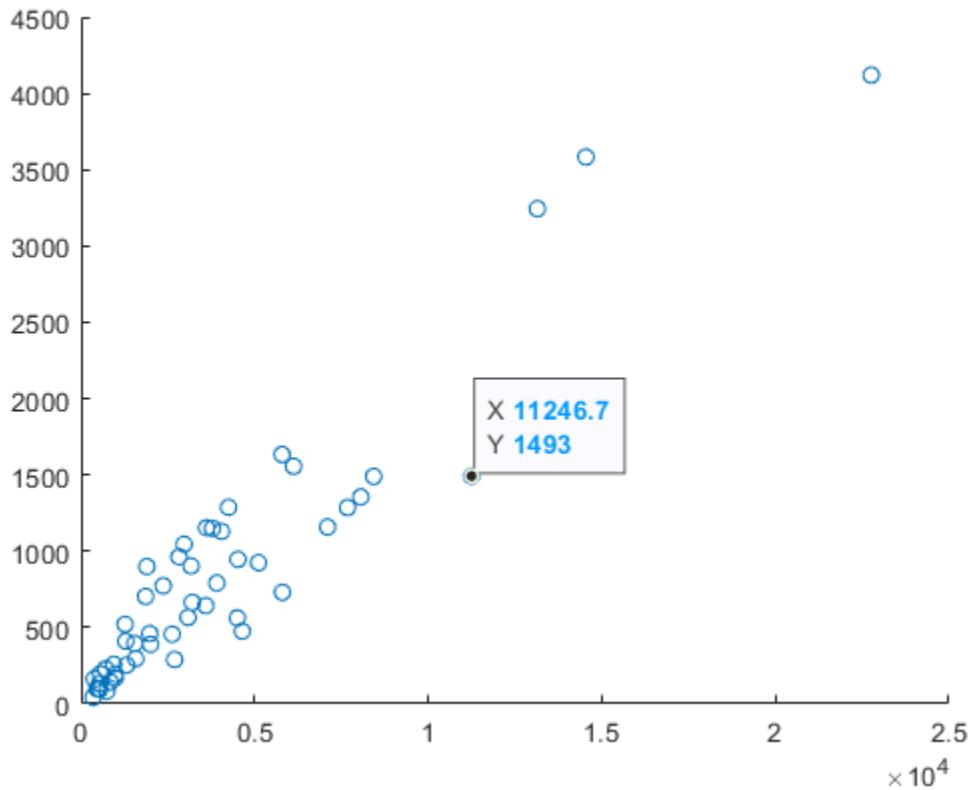
当您将光标悬停在数据点上时，将显示数据提示。默认情况下，数据提示显示所选点的坐标。但对于某些类型的图，您可以自定义数据提示中显示的信息，例如更改数据提示标签或添加新行。

支持这些自定义操作的图具有 **DataTipTemplate** 属性，例如使用 **plot** 函数创建的 **Line** 对象。

更改标签和添加行

修改散点图上的数据提示的内容。首先，加载样本事故数据并创建散点图。然后，以交互方式或使用 **datatip** 函数创建数据提示。默认情况下，数据提示显示数据点的坐标。

```
load('accidents.mat','hwydata','statelabel')
s = scatter(hwydata(:,5),hwydata(:,4));
dt = datatip(s,11246.7,1493);
```

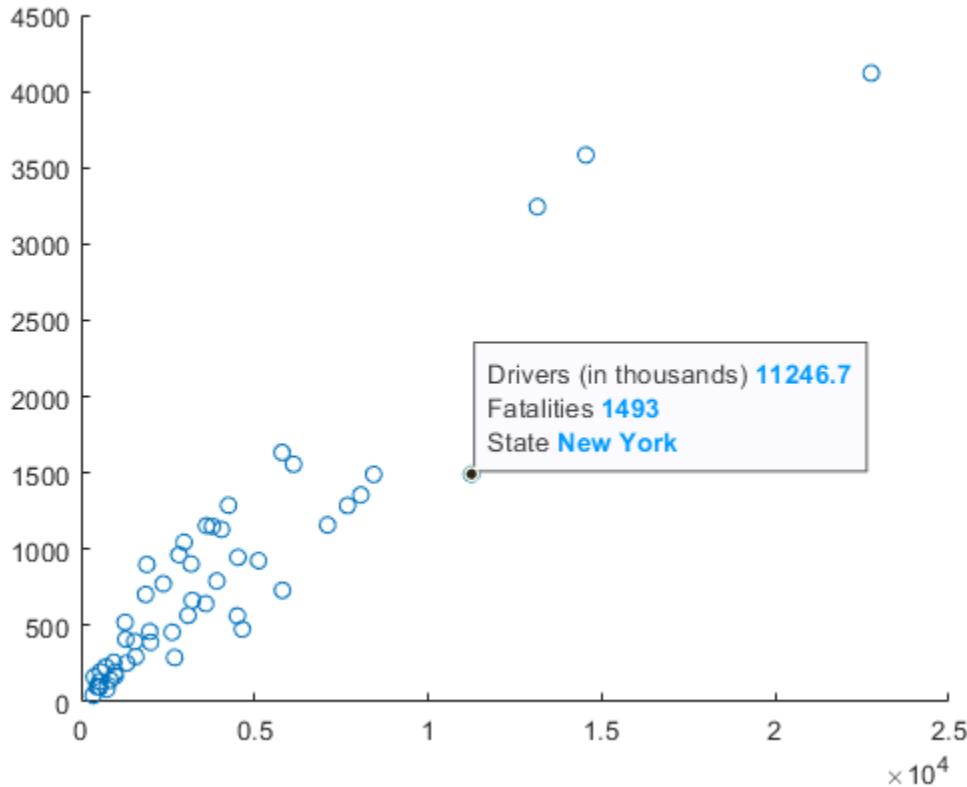


对于每行，通过访问绘制的对象的 **DataTipTemplate** 属性并设置 **Label** 属性，将数据提示标签从 X 和 Y 更改为 **Drivers (in thousands)** 和 **Fatalities**。

```
s.DataTipTemplate.DataTipRows(1).Label = 'Drivers (in thousands)';
s.DataTipTemplate.DataTipRows(2).Label = 'Fatalities';
```

在数据提示中添加一个新行。对于标签，使用 **State**。对于值，使用工作区中 **statelabel** 变量所包含的州名称。

```
row = dataTipTextRow('State',statelabel);
s.DataTipTemplate.DataTipRows(end+1) = row;
```



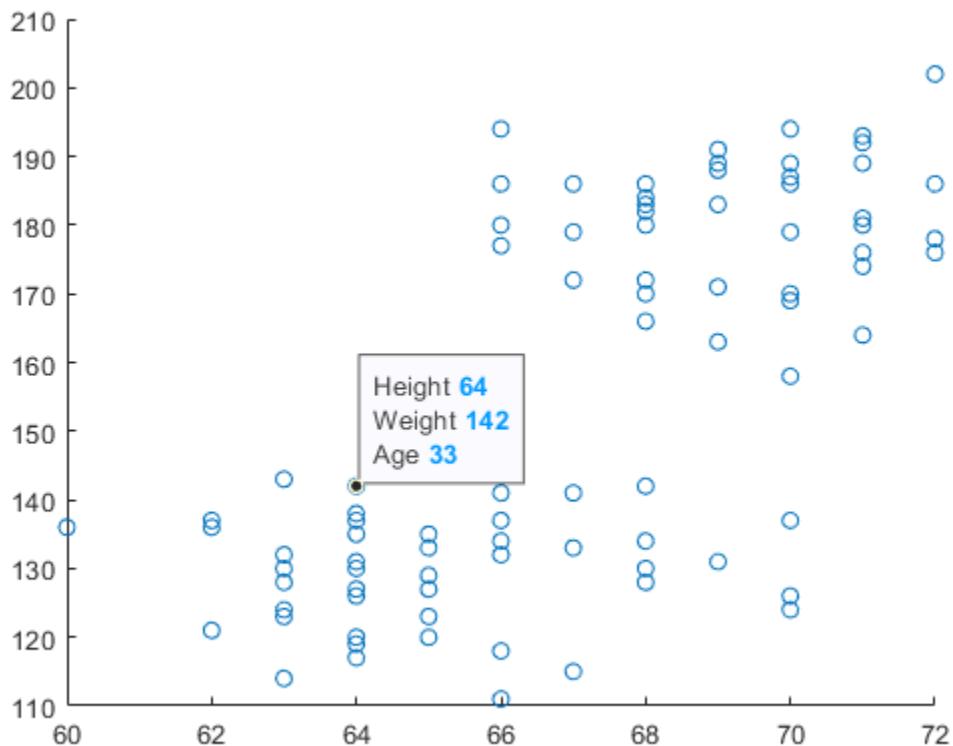
在数据提示中显示表值

修改散点图的数据提示内容，以包含表中的值。首先，从患者数据的示例电子表格创建一张表。绘制数据图。然后，以交互方式或通过使用 `datatip` 函数创建数据提示。

```
tbl = readtable('patients.xls');
s = scatter(tbl.Height,tbl.Weight);
dt = datatip(s,64,142);
```

将数据提示标签从 X 和 Y 更改为 Height 和 Weight。然后在数据提示中添加一个新行，该行使用标签 Age，并显示表中 “Age” 列的值。

```
s.DataTipTemplate.DataTipRows(1).Label = 'Height';
s.DataTipTemplate.DataTipRows(2).Label = 'Weight';
row = dataTipTextRow('Age',tbl.Age);
s.DataTipTemplate.DataTipRows(end+1) = row;
```



如果使用的是 R2018b 或更早版本，请通过设置 `datacursormode` 对象的 `UpdateFcn` 属性（而不是使用 `DataTipTemplate` 对象）来自定义数据提示。

另请参阅

[dataTipTextRow](#) | [DataTipTemplate](#) | [datatip](#)

详细信息

- “交互式探查绘图数据” (第 13-2 页)
- “更改数据后自动刷新图” (第 13-9 页)

更改数据后自动刷新图

绘制工作区变量中的数据时，图中包含的只是这些变量的副本。因此，如果更改工作区变量（例如添加或删除数据），图并不会自动更新。如果要在图中反映这种更改，必须重新绘图。但是，您可以使用下面的方法之一，将图链接到它们所表示的工作区变量。将图与工作区变量链接后，这两个位置会同时反映数据更改。

- 使用数据链接功能将图链接到工作区变量。
- 将绘图对象的数据源属性（例如 `XDataSource` 属性）设置为工作区变量的名称。然后调用 `refreshdata` 函数，间接更新数据属性。可以使用此方法来创建动画。

使用数据链接功能更新图

数据链接功能可使图与它们所表示的工作区变量保持持续同步。

例如，以迭代方式逼近 π 值。创建变量 `x` 表示迭代次数，创建变量 `y` 表示逼近值。绘制 `x` 和 `y` 的初始值。使用 `linkdata on` 打开数据链接功能，以便在变量更改时更新图。然后，通过 `for` 循环更新 `x` 和 `y`。图以半秒为间隔进行更新。

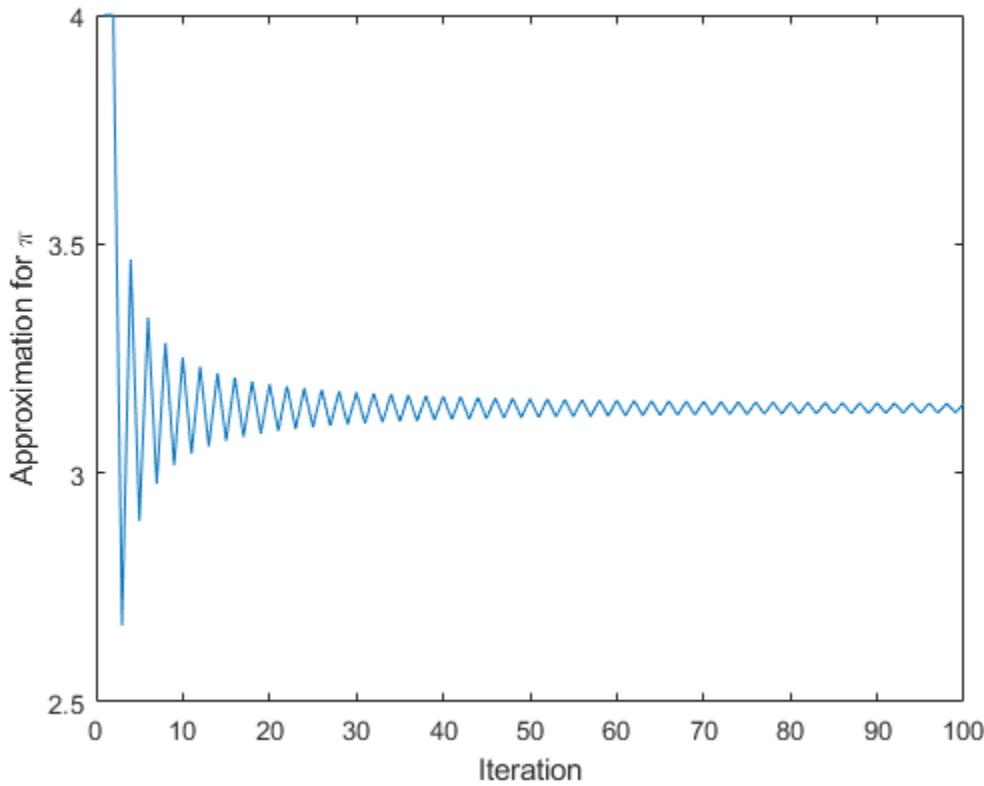
```

x = [1 2];
y = [4 4];
plot(x,y);
xlim([0 100])
ylim([2.5 4])
xlabel('Iteration')
ylabel('Approximation for \pi')

linkdata on

denom = 1;
k = -1;
for t = 3:100
    denom = denom + 2;
    x(t) = t;
    y(t) = 4*(y(t-1)/4 + k/denom);
    k = -k;
end

```



使用数据源属性更新图

除了使用数据链接功能外，还可以通过设置绘图对象的数据源属性使图与工作区变量保持同步。可以使用此方法来创建动画。

例如，以迭代方式逼近 π 值。创建变量 $x2$ 表示迭代次数，创建变量 $y2$ 表示逼近值。绘制 $x2$ 和 $y2$ 的初始值。通过将绘图对象的数据源属性设置为 ' $x2$ ' 和 ' $y2$ '，使图与工作区变量链接起来。然后，通过 `for` 循环更新 $x2$ 和 $y2$ 。在每次迭代中调用 `refreshdata` 和 `drawnow`，根据更新的数据来更新绘图。

```

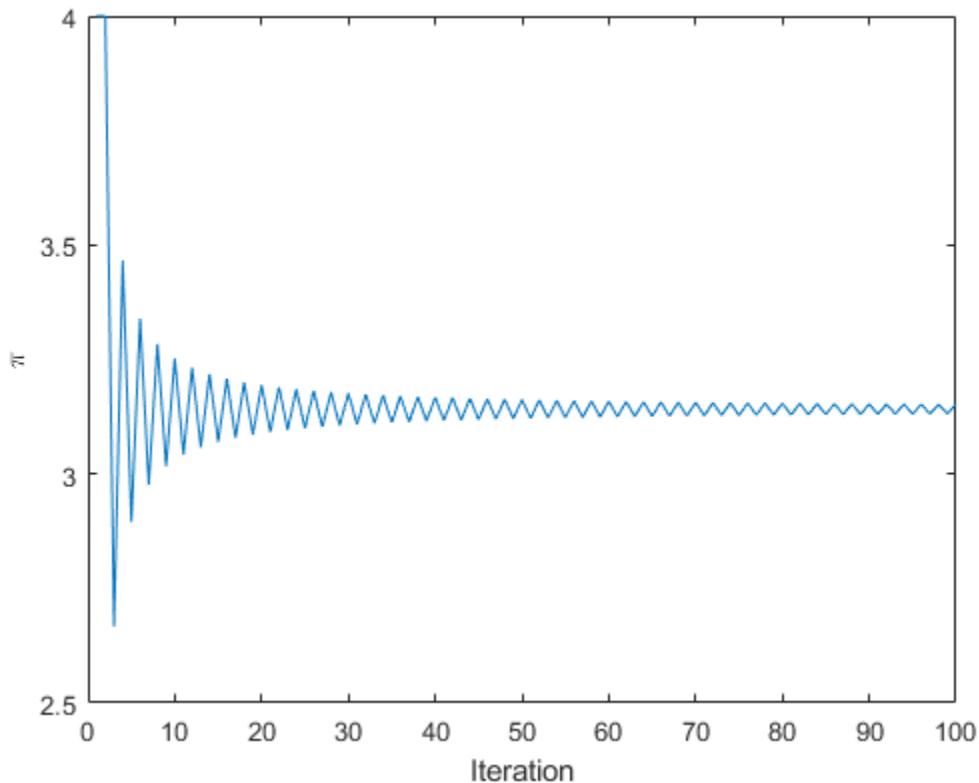
x2 = [1 2];
y2 = [4 4];
p = plot(x2,y2);
xlim([0 100])
ylim([2.5 4])
xlabel('Iteration')
ylabel('Approximation for \pi')

p.XDataSource = 'x2';
p.YDataSource = 'y2';

denom = 1;
k = -1;
for t = 3:100
    denom = denom + 2;
    x2(t) = t;
    y2(t) = 4*(y2(t-1)/4 + k/denom);
end

```

```
refreshdata  
drawnow  
k = -k;  
end
```



另请参阅

[linkdata](#) | [brush](#) | [refreshdata](#) | [linkaxes](#)

详细信息

- “[交互式探查绘图数据](#)” (第 13-2 页)

对图的交互进行控制

您可以交互式探查和编辑绘图数据，以改善数据的视觉效果或显示有关数据的其他信息。可用的交互操作取决于坐标区的内容，但通常包括缩放、平移、旋转、数据提示、数据刷亮以及还原原始视图。

有些类型的交互可通过坐标区工具栏使用。将鼠标悬停在图区域上时，工具栏会显示在坐标区的右上角。



其他交互类型已内置于坐标区中，并可通过手势使用，例如拖动平移或滚动缩放。这些交互的控制与坐标区工具栏中交互的控制是分开的。

创建图时，可以通过多种方式控制可用的交互集：

- 显示或隐藏坐标区工具栏（第 13-12 页）。
- 自定义坐标区工具栏（第 13-12 页）。
- 启用或禁用内置交互（第 13-13 页）。
- 自定义内置交互（第 13-14 页）。

在 R2018a 和之前的版本中，许多交互选项显示在图窗工具栏而不是坐标区工具栏上。此外，在以前的版本中，坐标区中没有内置任何交互。

显示或隐藏坐标区工具栏

要显示或隐藏坐标区工具栏，请将 `AxesToolbar` 对象的 `Visible` 属性相应设置为 `'on'` 或 `'off'`。例如，隐藏当前坐标区的工具栏：

```
ax = gca;
ax.Toolbar.Visible = 'off';
```

自定义坐标区工具栏

您可以使用 `axtoolbar` 和 `axtoolbarbtn` 函数自定义坐标区工具栏上的可用选项。

例如，为坐标区工具栏添加自定义状态按钮，以打开和关闭坐标区网格线。首先，创建一个名为 `mycustomstatebutton.m` 的程序文件。在该程序文件中：

- 绘制随机数据图。
- 使用 `axtoolbar` 函数为坐标区创建一个工具栏，其中包含放大、缩小和还原视图的选项。
- 使用 `axtoolbarbtn` 函数向工具栏添加一个空状态按钮。返回 `ToolbarStateButton` 对象。
- 通过设置 `Icon`、`Tooltip` 和 `ValueChangedFcn` 属性，指定状态按钮的图标、工具提示和回调函数。此示例使用 图标，此图标必须先作为图像文件保存到您的计算机路径中并命名为 `mygridicon.png`。

运行程序文件时，点击该图标即可打开和关闭网格线。

```
function mycustomstatebutton
```

```
plot(rand(5))
ax = gca;
```

```

tb = axtoolbar(ax,{'zoomin','zoomout','restoreview'});
```

btn = axtoolbarbtn(tb,'state');

btn.Icon = 'mygridicon.png';

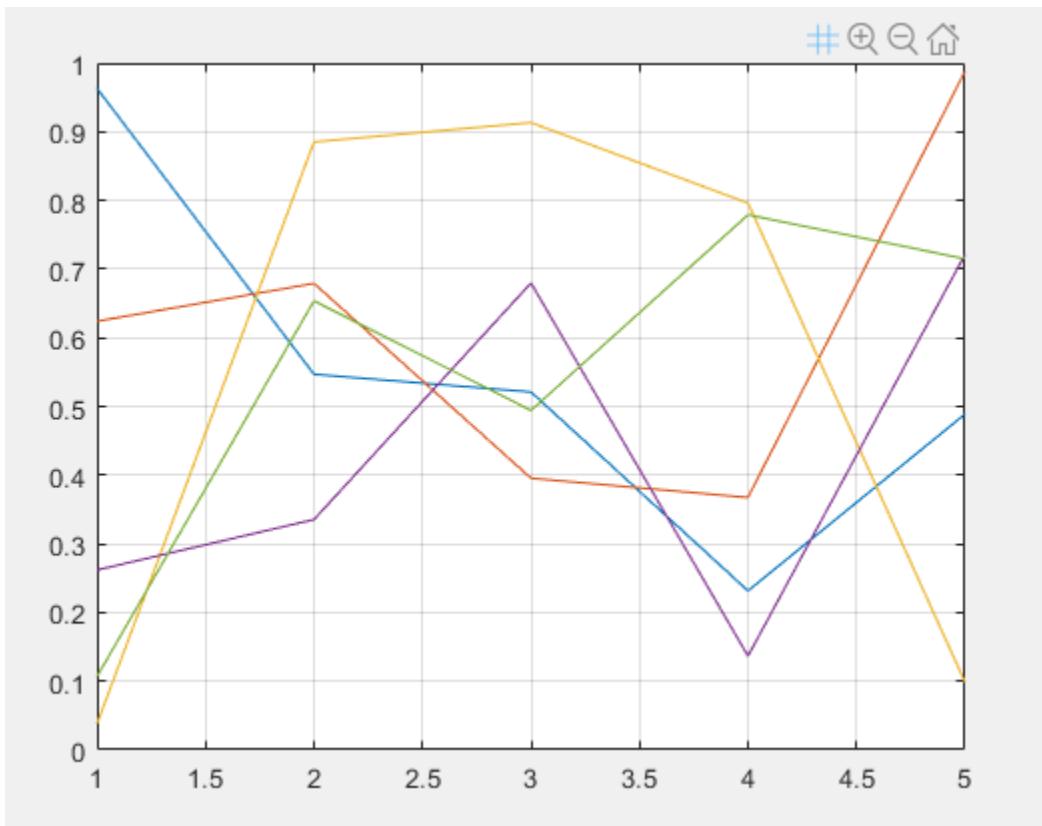
btn.Tooltip = 'Grid Lines';

btn.ValueChangedFcn = @customcallback;


```

function customcallback(src,event)
    switch src.Value
        case 'off'
            event.Axes.XGrid = 'off';
            event.Axes.YGrid = 'off';
            event.Axes.ZGrid = 'off';
        case 'on'
            event.Axes.XGrid = 'on';
            event.Axes.YGrid = 'on';
            event.Axes.ZGrid = 'on';
    end
end
```


end



启用或禁用内置交互

要控制是否在图中启用某内置交互集，请使用 `disableDefaultInteractivity` 和 `enableDefaultInteractivity` 函数。有时 MATLAB 会自动禁用内置交互。例如，对于包含特殊功能的图，或在实现某些回调（例如 `WindowScrollWheelFcn` 回调）时，可能会禁用内置交互。

自定义内置交互

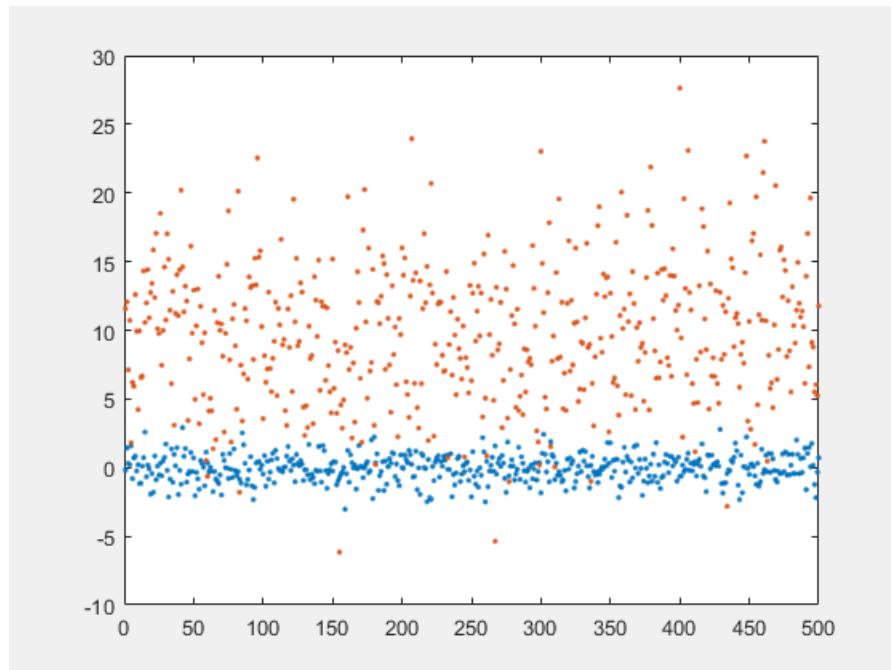
大多数类型的坐标区都包括一组默认的与特定手势对应的内置交互。可用的交互取决于坐标区的内容。大多数笛卡尔坐标区包含以下交互：滚动缩放、悬停或点击显示数据提示，以及拖动平移（在二维视图中）或旋转（在三维视图中）。您可以将默认集替换为新的交互集，但不能访问或修改默认集中的任何交互。

要替换默认交互，请将坐标区的 **Interactions** 属性设为交互对象数组。从下表中选择兼容的交互对象组合。要从坐标区删除所有交互，请将该属性设置为空数组 ([])

交互对象	说明	兼容的交互
panInteraction	通过拖动操作在图中平移。	除 regionZoomInteraction 和 rotateInteraction 以外的全部交互
rulerPanInteraction	通过拖动坐标轴对其进行平移。	全部
zoomInteraction	通过滚动或捏合手指进行缩放。	全部
regionZoomInteraction	通过拖动操作放大矩形区域。 (仅适用于二维笛卡尔图)	除 panInteraction 和 rotateInteraction 以外的全部交互
rotateInteraction	通过拖动图对其进行旋转。	除 panInteraction 和 regionZoomInteraction 以外的全部交互
dataTipInteraction	通过悬停、单击或点击显示数据提示。	全部

例如，创建一幅包含 1000 个散点的绘图。

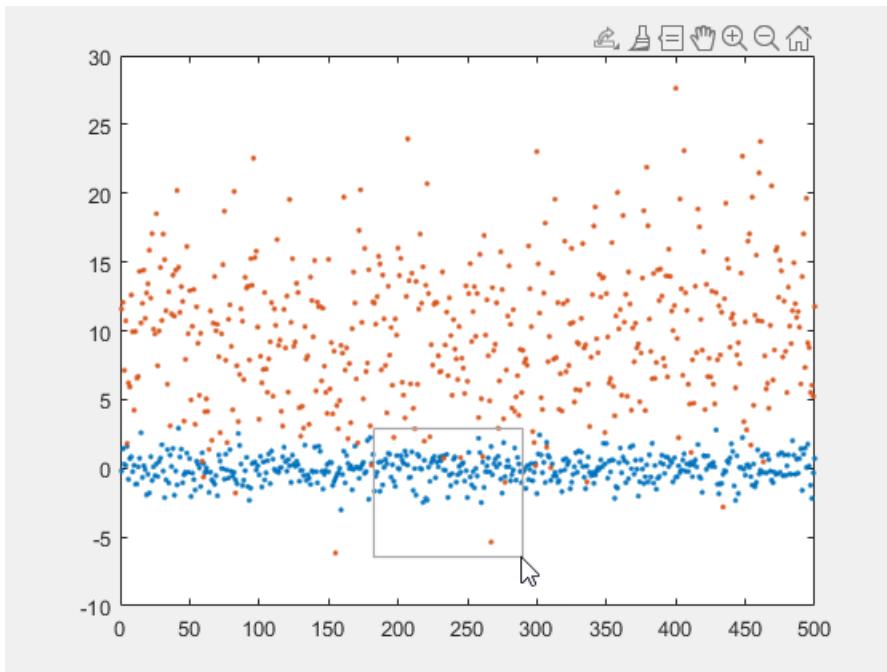
```
x = 1:500;
y = randn(1,500);
y2 = 5*randn(1,500) + 10;
plot(x,y,'.',x,y2,'.')
```



默认情况下，此绘图提供的交互支持图区域内的拖动平移。但由于该绘图包含大量密集的点，支持放大绘图特定区域的交互会更加有用。`regionZoomInteraction` 对象提供此功能。将当前坐标区的默认交互集替换为包括 `regionZoomInteraction` 对象的数组。

```
ax = gca;
ax.Interactions = [zoomInteraction regionZoomInteraction rulerPanInteraction];
```

现在，在绘图区域内拖动鼠标会定义一个需要放大的矩形目标区域。



另请参阅

函数

`axtoolbar`

属性

`AxesToolbar` | `ToolbarPushButton` | `ToolbarStateButton` | `Axes`

详细信息

- “[交互式探查绘图数据](#)”（第 13-2 页）

相机视图

- “视图概述” (第 14-2 页)
- “利用方位角和仰角设置视点” (第 14-3 页)
- “相机图形术语” (第 14-7 页)
- “使用相机工具栏控制视图” (第 14-8 页)
- “推移相机” (第 14-17 页)
- “移动相机穿过场景” (第 14-18 页)
- “低级相机属性” (第 14-21 页)
- “了解视图投影” (第 14-26 页)

视图概述

本节内容

- “观察三维图形和场景”（第 14-2 页）
- “定位视点”（第 14-2 页）
- “设置纵横比”（第 14-2 页）
- “默认视图”（第 14-2 页）

观察三维图形和场景

视图是指显示图形或图形场景时选择的特定方向。观察是指从各方向显示图形场景、放大或缩小、更改透视和纵横比以及执行漫游等操作的过程。

本节介绍如何定义各种观察参数以获得所需的视图。一般而言，“观察”适用于三维图形或模型，尽管您也可能想要调整二维视图的纵横比以实现特定比例或使图形适合特定形状。

MATLAB 观察包括两个基本的方面：

- 定位视点以确定场景方向
- 设置纵横比和相对轴比例以控制所显示对象的形状

定位视点

- “利用方位角和仰角设置视点”（第 14-3 页） - 讨论如何利用方位角和仰角来指定观察图形的视点。此操作从概念上讲很简单，但存在一些限制。
- “使用相机工具栏控制视图”（第 14-8 页） - 说明如何使用 MATLAB 相机观察模型合成复杂场景。
- “移动相机穿过场景”（第 14-18 页） - 介绍围绕和穿过场景移动视图的编程技巧。
- “低级相机属性”（第 14-21 页） - 介绍控制相机的图形属性并说明它们产生的效果。

设置纵横比

- “了解视图投影”（第 14-26 页） - 介绍正交投影和透视投影类型并说明它们的用法。
- “控制坐标区纵横比”（第 9-67 页） - 说明 MATLAB 如何设置坐标区纵横比以及您如何为图形选择最合适设置。

默认视图

当您创建图形时，MATLAB 会自动设置视图。MATLAB 选择的实际视图取决于您创建的是二维图形还是三维图形。有关 MATLAB 如何定义标准视图的说明，请参阅“默认视点选择”（第 14-21 页）和“默认纵横比选择”（第 9-68 页）。

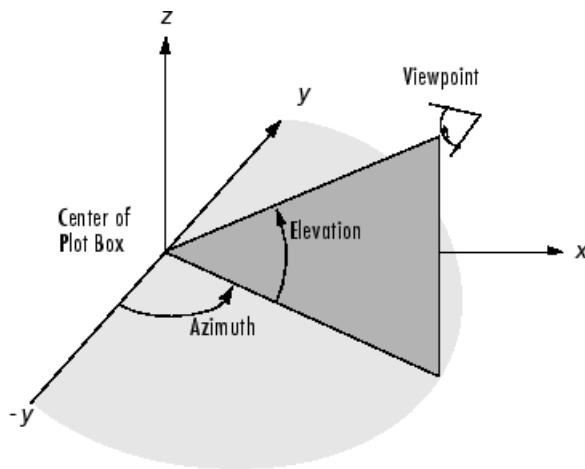
利用方位角和仰角设置视点

方位角和仰角

您可以使用 MATLAB 图形函数控制坐标区内显示的图形的方向。您可以指定图窗窗口中显示的视图的视点、视图目标、方向和范围。这些查看特性由一组图形属性控制。您可以直接指定这些属性的值，也可以使用 `view` 命令并依靠 MATLAB 的自动属性选择功能来定义合理的视图。

`view` 命令通过定义相对于坐标轴原点的方位角和仰角来指定视点。方位角是 x - y 平面上的极坐标角，正值表示按逆时针方向旋转视点。仰角是位于 x - y 平面上方（正）或下方（负）的角度。

下图展示了坐标系。箭头指示正方向。



默认的二维和三维视图

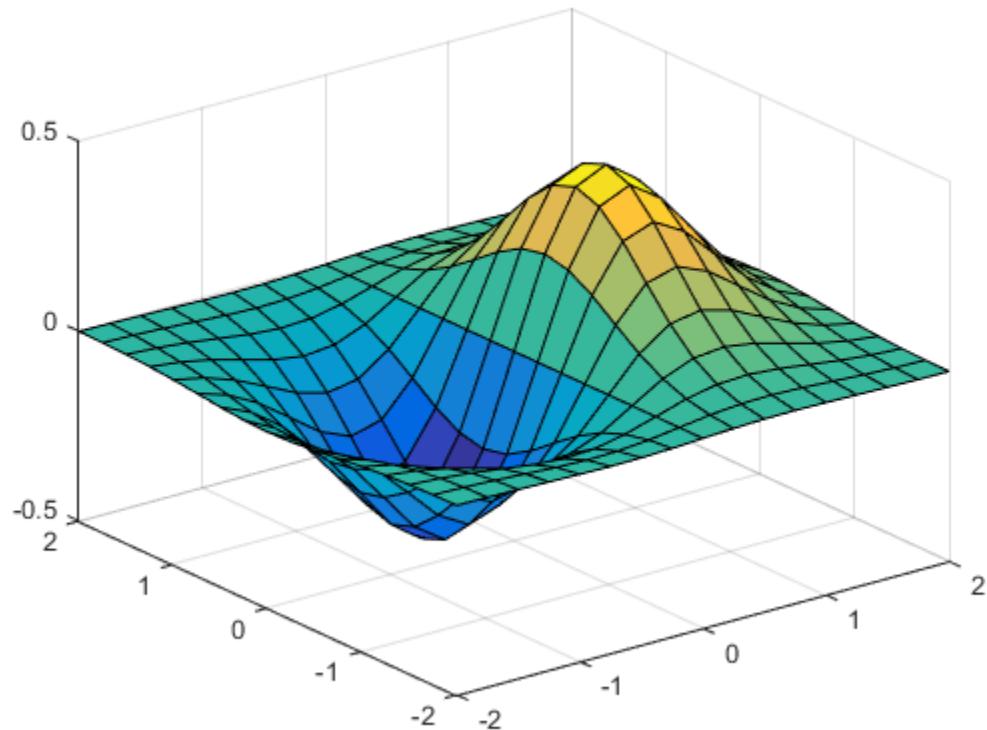
MATLAB 会自动选择视点，该视点由图形是二维还是三维来决定：

- 对于二维绘图，默认值是方位角 = 0° ，仰角 = 90° 。
- 对于三维绘图，默认值是方位角 = -37.5° ，仰角 = 30° 。

利用方位角和仰角指定的视图示例

例如，以下语句将创建一个三维曲面图并将其显示在默认的三维视图中。

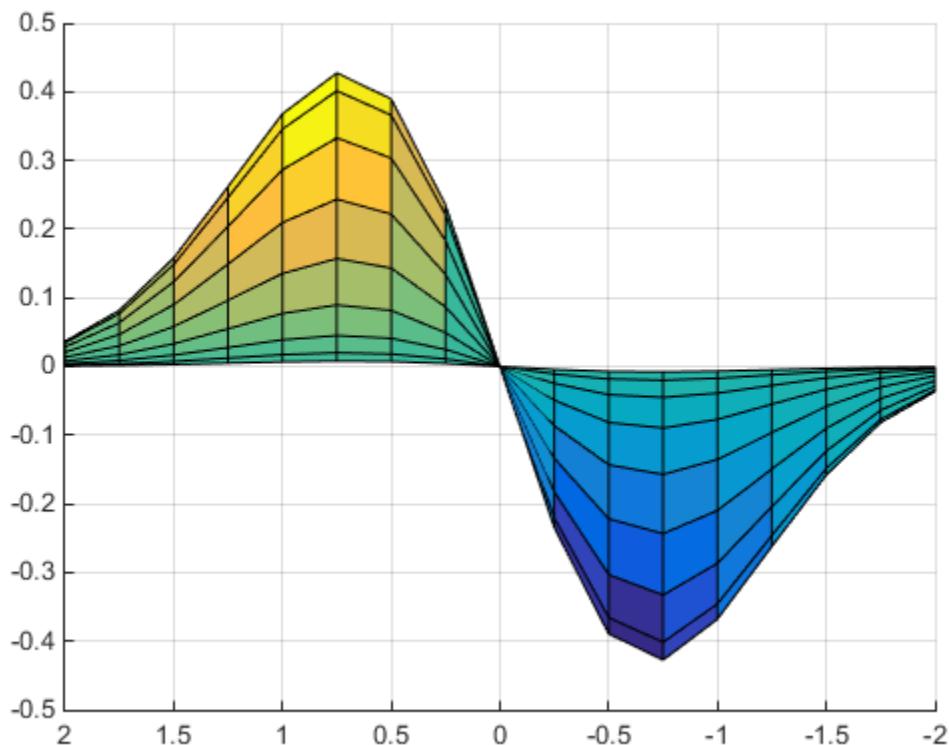
```
[X,Y] = meshgrid([-2:25:2]);
Z = X.*exp(-X.^2 -Y.^2);
surf(X,Y,Z)
```



语句

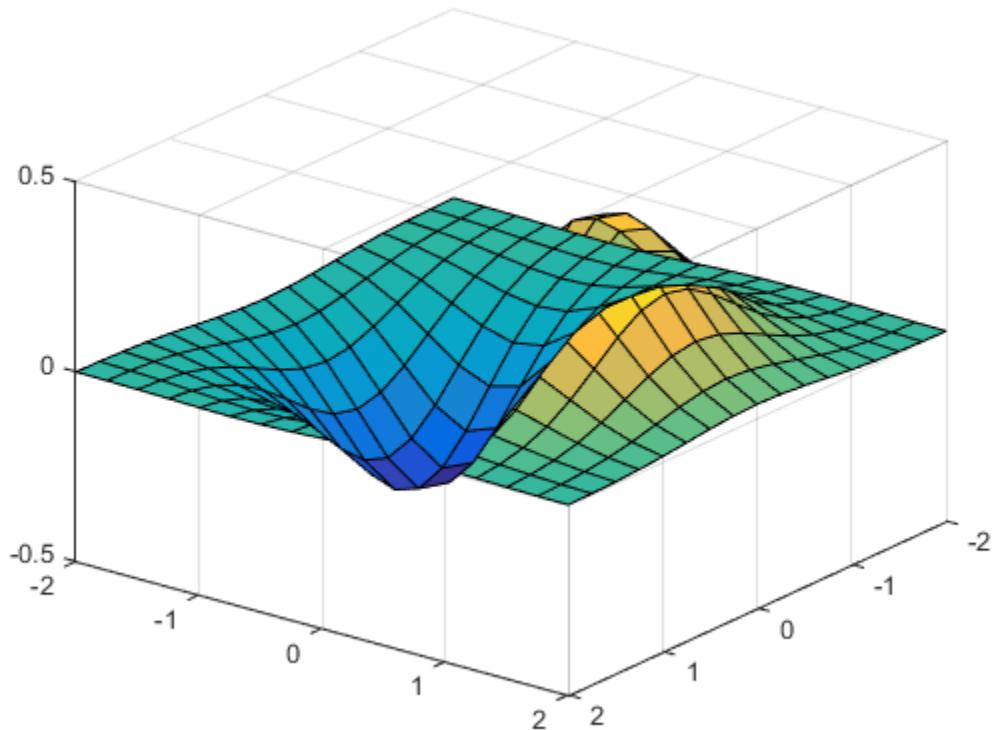
```
view([180 0])
```

设置的视点让您从 y 轴负方向观看，眼睛的仰角为 $z = 0$ 。



您可以使用负的仰角值，将视点移动到坐标轴原点下方的某个位置。

```
view([-37.5 -30])
```



方位角和仰角的限制

利用方位角和仰角指定视点从概念上讲很简单，但存在一些限制。它不允许您指定视点的实际位置，只能指定方向，而且 z 轴始终指向上方。它不允许您放大或缩小场景，也不允许您执行任意旋转和转换。

相对于方位角和仰角允许的简单调整，MATLAB 相机图形提供了更好的控制。

另请参阅

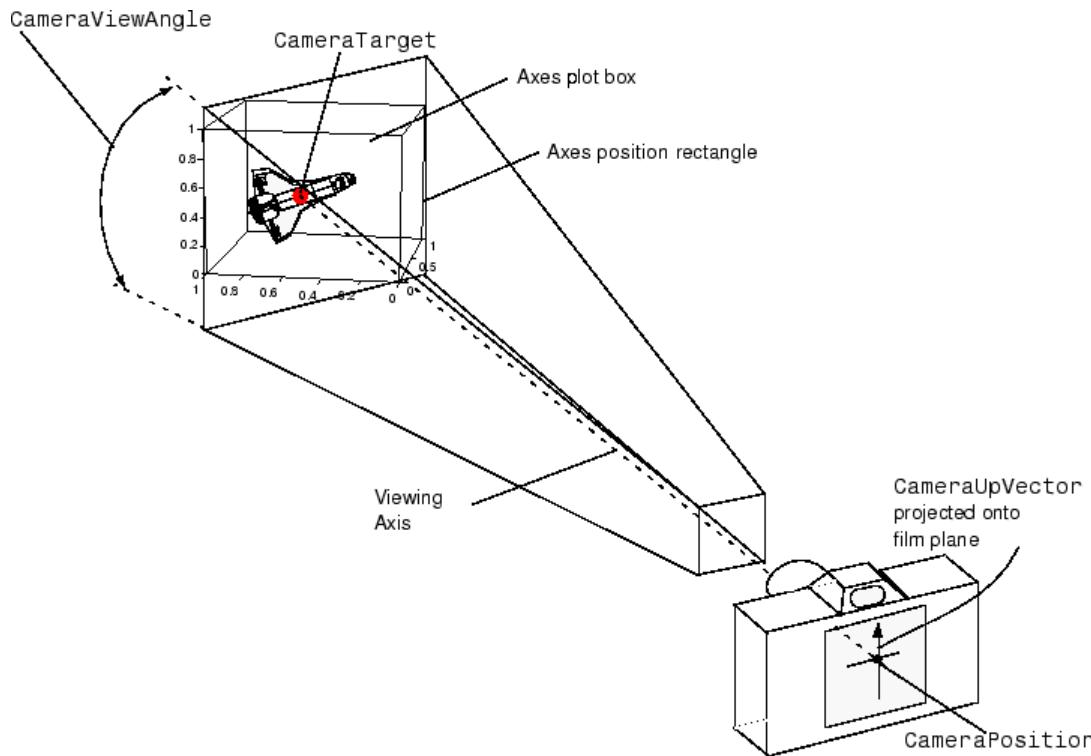
详细信息

- “相机图形术语”（第 14-7 页）
- “使用相机工具栏控制视图”（第 14-8 页）

相机图形术语

当您查看坐标区中显示的图形对象时，您是从空间中相对于场景的特定方向上的特定位置观察场景。MATLAB 图形提供的功能类似于带变焦镜头的相机，可让您控制由 MATLAB 创建的场景视图。

下图说明如何从坐标区属性方面定义相机。该视图是图框在屏幕上的二维投影。



另请参阅

[camdolly](#) | [camlookat](#) | [camorbit](#) | [campan](#) | [camproj](#) | [camroll](#) | [camtarget](#) | [camup](#) | [camva](#) | [camzoom](#)

相关示例

- “使用相机工具栏控制视图”（第 14-8 页）

使用相机工具栏控制视图

本节内容

- “相机工具栏” (第 14-8 页)
- “相机移动控件” (第 14-10 页)
- “环移相机” (第 14-10 页)
- “环移场景灯光” (第 14-11 页)
- “平转/纵转相机” (第 14-11 页)
- “水平/垂直移动相机” (第 14-12 页)
- “向前和向后移动相机” (第 14-13 页)
- “缩放相机” (第 14-14 页)
- “相机滚转” (第 14-15 页)

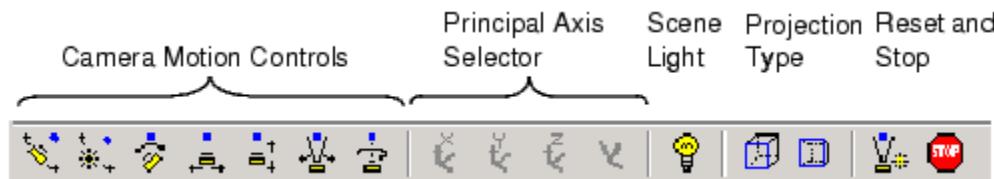
相机工具栏

相机工具栏允许您以交互方式执行多种观察操作。要使用相机工具栏，请执行以下操作：

- 显示工具栏，方法是：从图窗窗口的**视图**菜单中选择**相机工具栏**，或者在命令行窗口中键入 **cameratoolbar**。
- 选择要使用的相机移动控制类型，方法是：点击相应的按钮，或者在命令行窗口更改 **cameratoolbar** 模式。
- 将光标置于图窗窗口上并点击，按住鼠标右键，然后将光标移动到所需的方向。

显示屏将随着鼠标移动即时更新。

工具栏包含以下几部分：



- 相机移动控制 - 这些工具用于选择要启用的相机移动功能。您也可以从**工具**菜单中访问相机移动控制。
- 主轴选择器 - 某些相机控制的工作方式基于特定的坐标轴。这些选择器允许您选择主轴或选择不受坐标轴限制的移动。如果不适用于当前选定的功能，选择器将灰显。您还可以从**工具**菜单中访问主轴选择器。
- 场景灯光 - 场景灯光按钮用于打开或关闭场景中的光源（每个坐标区一个光源）。
- 投影类型 - 您可以选择正交投影或透视投影类型。
- 重置和停止 - 重置可将场景恢复为交互开始时的视图。“停止”将使相机停止移动（如果您应用了太多的光标移动，这会很有用）。您还可以从**工具**菜单中访问更多重置功能。

主轴

场景的主轴定义屏幕上向上的方向。例如，MATLAB 曲面图的向上方向与 z 轴的正方向一致。

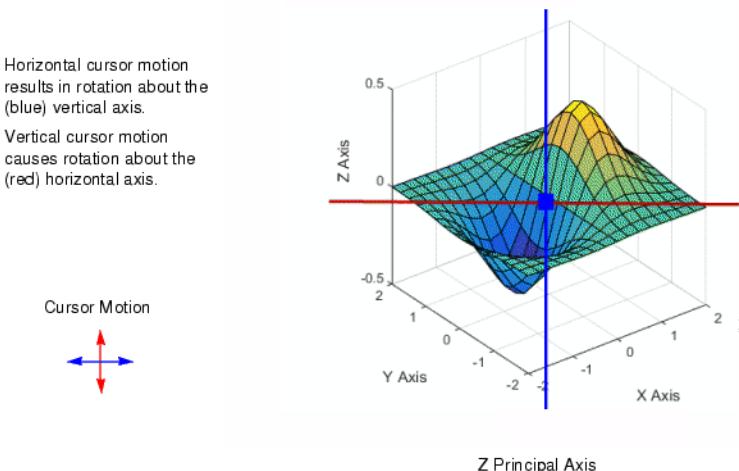
主轴将相机移动限制在与您选择的主轴平行（在屏幕上）和垂直的坐标轴上。如果您的数据是针对特定坐标轴定义的，则指定主轴很有用。Z 轴是默认的主轴，因为这与默认的 MATLAB 三维视图一致。

有两个相机工具（环移和平转/纵转）允许您选择主轴和不受坐标轴限制的移动。在屏幕上，旋转轴由一条与主轴平行的垂直线和一条与主轴垂直的水平线决定，这两条线都穿过 CameraTarget 属性所定义的点。

例如，当主轴是 z 轴时，移动时所围绕的中心轴为

- 穿过相机目标并与 z 轴平行的垂直线
- 穿过相机目标并与 z 轴垂直的水平线

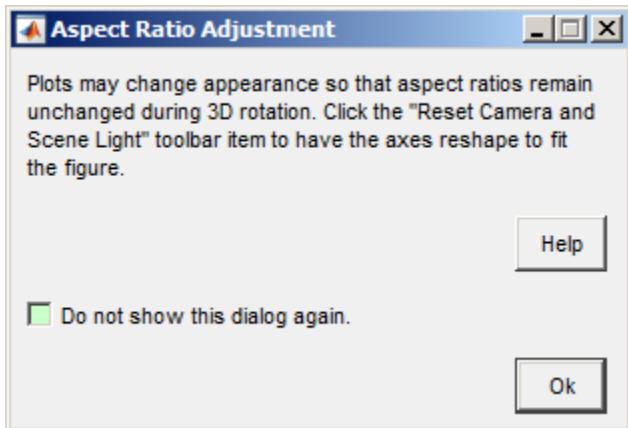
这意味着场景（或相机，视情况而定）以相机目标为中心进行弧线移动。下图显示了主轴为 z 轴时的旋转轴。



旋转轴始终穿过相机目标。

针对三维相机移动的优化

当您创建绘图时，MATLAB 将以适合图窗窗口的纵横比显示它。这一行为可能无法产生最优的三维图形操作环境，当您围绕场景移动相机时，可能会产生变形。为了避免可能的变形，最好切换到三维可视化模式（从命令行中使用命令 `axis vis3d` 来启用）。使用相机工具栏时，MATLAB 会自动切换到三维可视化模式，但会首先通过以下对话框发出警告。



此对话框在每个 MATLAB 会话中仅出现一次。

相机移动控件

本节讨论可从工具栏上选择的各项相机移动功能。

注意 理解下面的图时，请记住，相机始终指向相机目标。有关相机移动中涉及到的图形属性的说明，请参阅“相机图形术语”（第 14-7 页）。

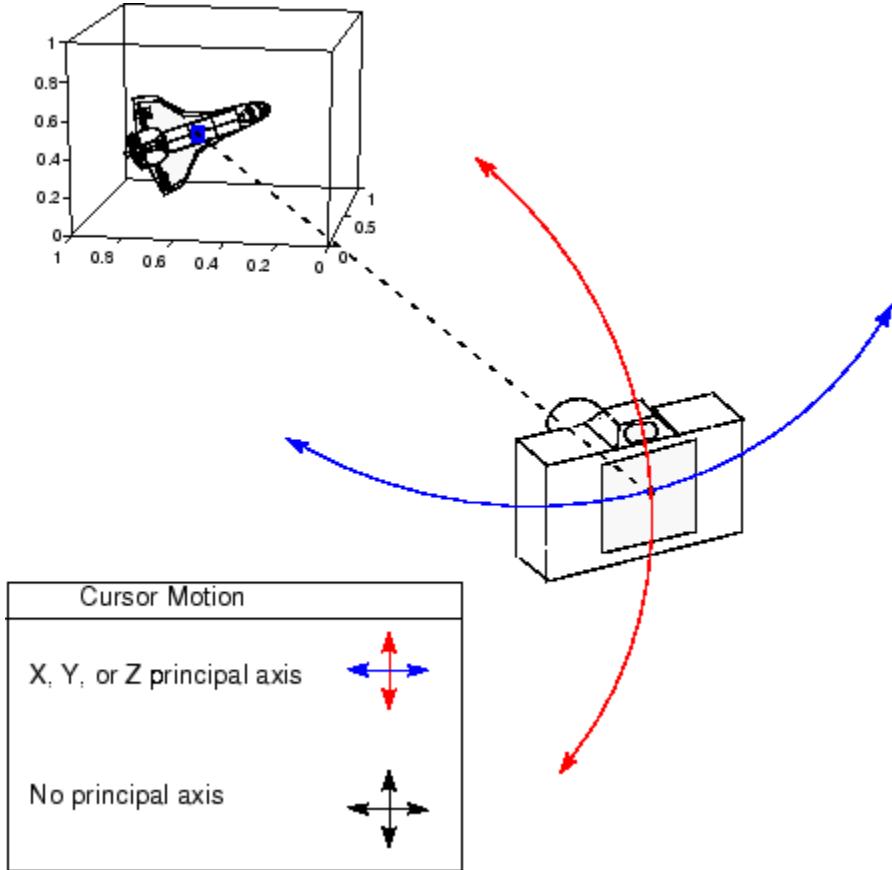
环移相机



环移相机围绕 z 轴（默认值）旋转相机。您可以使用主轴选择器选择围绕 x 轴、y 轴、z 轴旋转或者选择不受坐标轴限制的旋转。不使用主轴时，可以围绕任意轴旋转。

图形属性

环移相机会更改 **CameraPosition** 属性，但 **CameraTarget** 保持不变。



环移场景灯光



场景灯光是相对于相机位置放置的光源。默认情况下，场景灯光位于相机的右侧（即 `camlight right`）。

“环移场景灯光”可改变光源离相机位置的偏移量。场景灯光只有一个，但您可以使用 `light` 命令添加其他光源。

点击黄色灯泡图标可以打开和关闭场景灯光。

图形属性

环移场景灯光通过更改光源的 **Position** 属性来移动场景灯光。

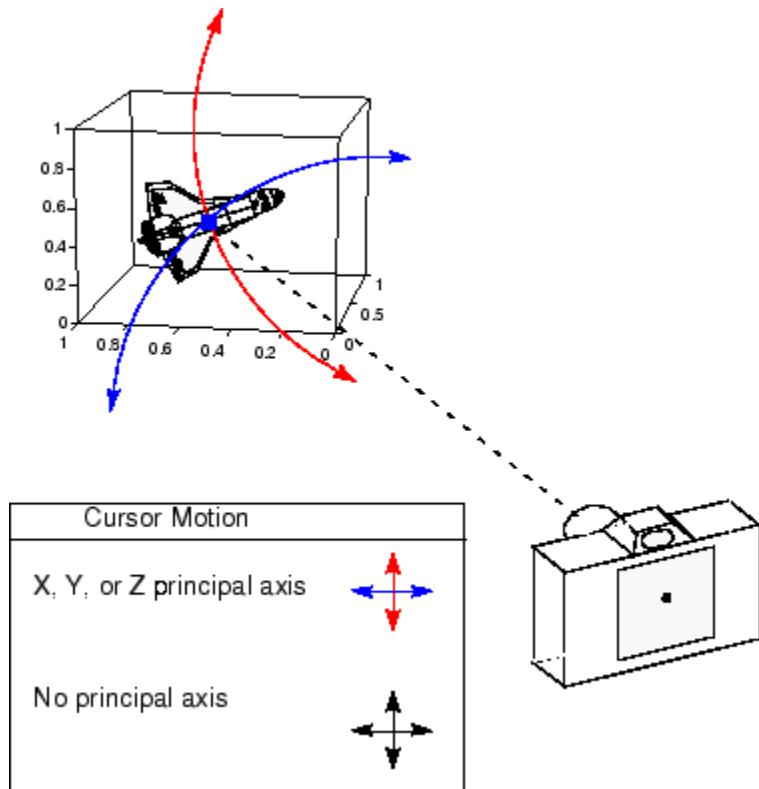
平转/纵转相机



“平转/纵转相机”可移动场景中相机所指向的点，但相机保持固定。默认情况下，移动围绕 z 轴沿弧线进行。您可以使用主轴选择器选择围绕 x 轴、y 轴、z 轴旋转或者选择不受坐标轴限制的旋转。

图形属性

平转/纵转相机通过更改 **CameraTarget** 属性来移动场景中相机所指向的点。



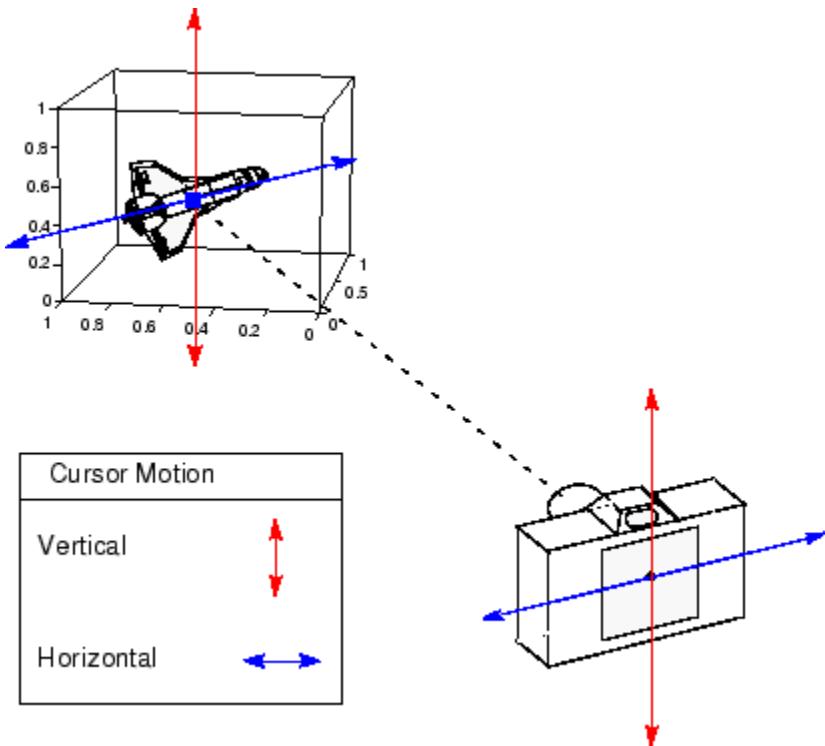
水平/垂直移动相机



水平或垂直移动光标（或两者的任意组合）将按相同方向移动场景。

图形属性

水平和垂直移动可通过沿平行线协调一致地移动 **CameraPosition** 和 **CameraTarget** 来实现。



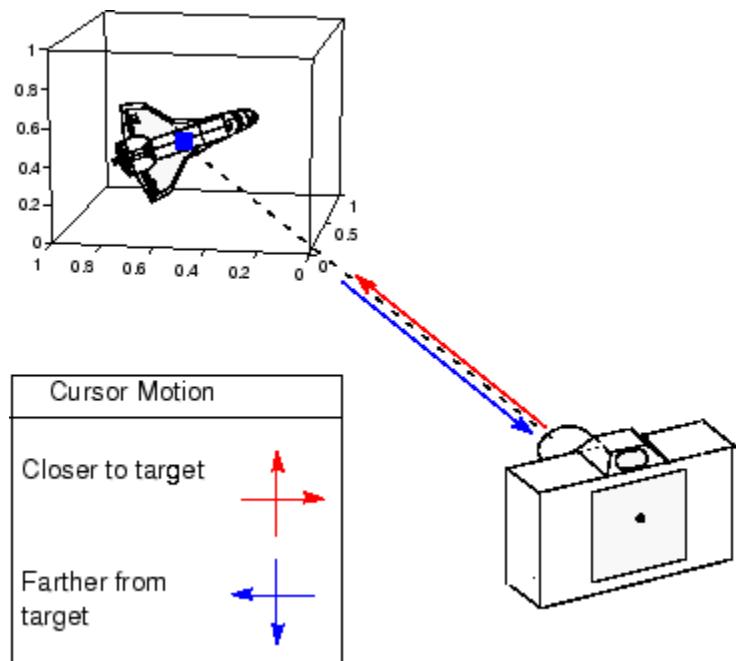
向前和向后移动相机



向上或向右移动光标可将相机移向场景。向下或向左移动光标可将相机移离场景。可以移动相机穿过场景中的对象，到达相机目标的另一侧。

图形属性

此功能沿相机位置与相机目标之间的连线移动 CameraPosition。



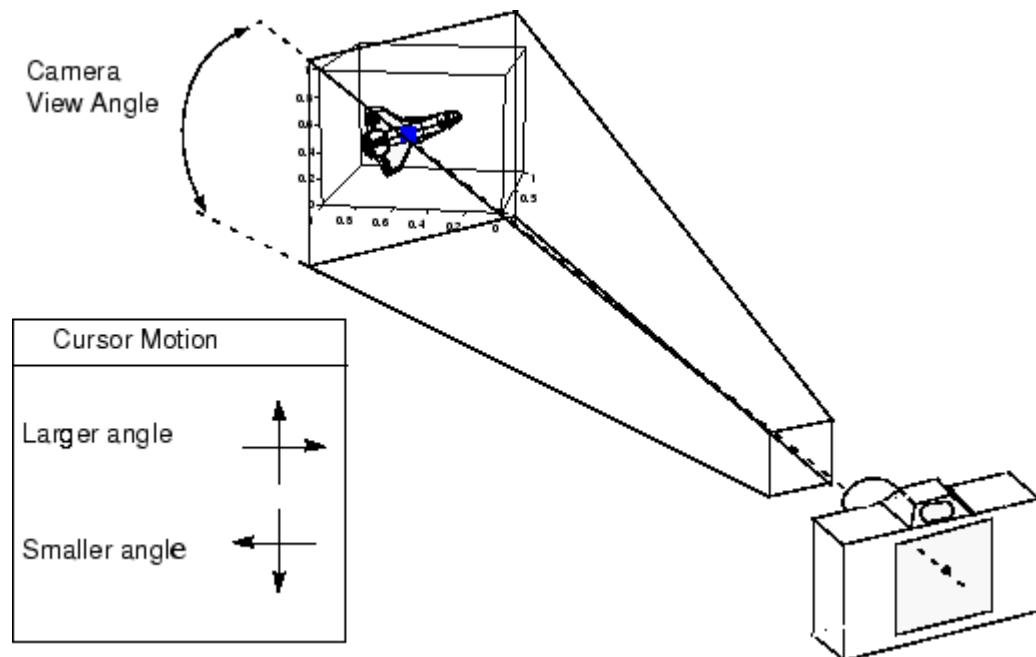
缩放相机



使用“缩放相机”功能，向上或向右移动光标时，可使场景变大；向下或向左移动光标时，可使场景变小。缩放并不会移动相机，因此不能移动视点使之穿过场景中的对象。

图形属性

缩放可通过更改 `CameraViewAngle` 来实现。角度越大，场景越小，反之亦然。



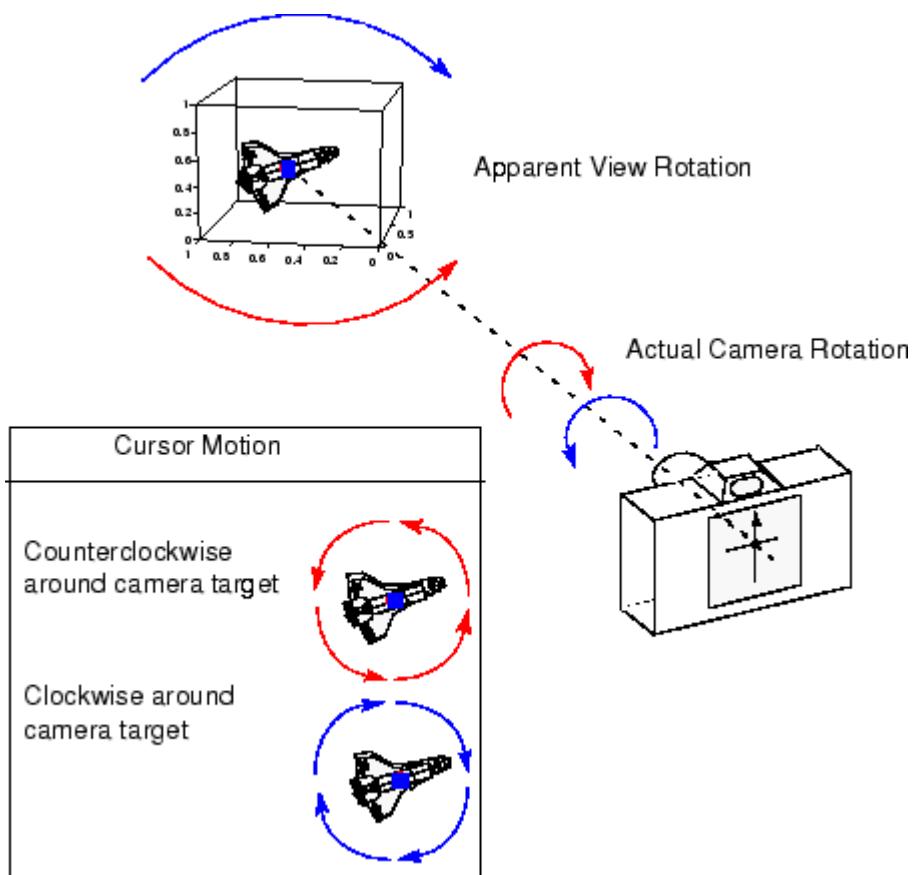
相机滚转



相机滚转是围绕观察轴旋转相机，从而旋转屏幕上的视图。

图形属性

相机滚转将更改 CameraUpVector。



推移相机

本节内容

“方法简介”（第 14-17 页）

“实现”（第 14-17 页）

方法简介

在相机领域，移动摄影车是指在可将相机从场景的一侧移到另一侧的平台。**camdolly** 命令通过同时移动相机的位置和相机目标的位置来实现类似的行为（如果需要，也可以只移动相机的位置）。

此示例说明如何使用 **camdolly** 浏览图像的不同区域。其中介绍了以下函数的用法：

- **ginput** 用于获取图像上各个位置的坐标
- **camdolly data** 坐标选项用于根据从 **ginput** 获得的坐标，将相机和目标移动到新位置
- **camva** 用于放大和固定相机视角，若不设置将采用自动控制模式

实现

首先加载鳕鱼角图像并通过设置相机视角（使用 **camva**）进行放大。

```
load cape
image(X)
colormap(map)
axis image
camva(camva/2.5)
```

然后，使用 **ginput** 选择相机目标和相机位置的 x 和 y 坐标。

```
while 1
    [x,y] = ginput(1);
    if ~strcmp(get(gcf,'SelectionType'),'normal')
        break
    end
    ct = camtarget;
    dx = x - ct(1);
    dy = y - ct(2);
    camdolly(dx,dy,ct(3),'movetarget','data')
    drawnow
end
```

移动相机穿过场景

本节内容

- “方法简介”（第 14-18 页）
- “绘制体数据”（第 14-18 页）
- “设置视图”（第 14-19 页）
- “指定光源”（第 14-19 页）
- “选择光照方法”（第 14-19 页）
- “将相机路径定义为流线”（第 14-19 页）
- “实现漫游”（第 14-20 页）

方法简介

漫游是指移动相机穿过三维空间而产生的一种效果，就好像把相机放在飞机上，而您跟着相机一起飞一样。您可以漫游场景中可能被其他对象挡住的区域，也可以通过将相机定焦在特定点上从场景旁边漫游。

要实现这些效果，可按照一系列步骤沿特定路线（例如沿 x 轴）移动相机。要生成漫游效果，请同时移动相机位置和相机目标。

以下示例利用漫游效果查看在由风速向量场定义的三维体内绘制的等值面的内部。这些数据代表北美地区的气流。

本示例使用了好几种可视化方法。它使用的方法有：

- 使用等值面和圆锥图来说明通过三维体的气流
- 使用光照照亮三维体内的等值面和圆锥体
- 使用流线定义相机穿过三维体的路线
- 协调移动相机位置、相机目标和光源

绘制体数据

第一步是绘制等值面并使用圆锥图绘制气流。

请参阅 `isosurface`、`isonormals`、`reducepatch` 和 `coneplot` 以了解如何使用这些命令的信息。

在绘制圆锥图之前，将数据纵横比 (`daspect`) 设置为 [1,1,1] 可确保 MATLAB 软件正确计算最终视图的圆锥体大小。

```
load wind
wind_speed = sqrt(u.^2 + v.^2 + w.^2);
figure
p = patch(isosurface(x,y,z,wind_speed,35));
isonormals(x,y,z,wind_speed,p)
p.FaceColor = [0.75,0.25,0.25];
p.EdgeColor = [0.6,0.4,0.4];

[f,vt] = reducepatch(isosurface(x,y,z,wind_speed,45),0.05);
daspect([1,1,1]);
hcone = coneplot(x,y,z,u,v,w,vt(:,1),vt(:,2),vt(:,3),2);
```

```
hccone.FaceColor = 'blue';
hccone.EdgeColor = 'none';
```

设置视图

您需要定义查看参数，以确保正确显示场景：

- 选择透视投影可提供相机穿过等值面内部时的深度感 (`camproj`)。
- 将相机视角设置为固定值可防止 MATLAB 自动调整视角以包含整个场景以及放大所需量 (`camva`)。

```
camproj perspective
camva(25)
```

指定光源

将光源定位在相机位置并修改等值面和圆锥体的反射特性可增强场景的真实感：

- 在相机位置创建光源可提供“前灯”效果，与相机一起在等值面内部移动 (`camlight`)。
- 设置等值面的反射属性可产生高反射材料 (`SpecularStrength` 和 `DiffuseStrength` 设置为 1) 的黑暗内景效果 (`AmbientStrength` 设置为 0.1)。
- 将圆锥体的 `SpecularStrength` 设置为 1 可使它们具有高反射性。

```
hlight = camlight('headlight');
p.AmbientStrength = 1;
p.SpecularStrength = 1;
p.DiffuseStrength = 1;
hccone.SpecularStrength = 1;
set(gcf,'Color','k')
set(gca,'Color',[0,0,0.25])
```

选择光照方法

使用 `gouraud` 光照可获得更平滑的光照效果：

```
lighting gouraud
```

将相机路径定义为流线

流线表示向量场中的流向。此示例使用单个流线的 x、y 和 z 坐标数据映射穿过三维体的路径。之后相机将沿着这条路径移动。任务包括

- 从点 $x = 80$ 、 $y = 30$ 、 $z = 11$ 开始创建一条流线。
- 获取流线的 x、y 和 z 坐标数据。
- 删除流线（您也可以使用 `stream3` 计算流线数据，而不用实际绘制流线）。

```
hsline = streamline(x,y,z,u,v,w,80,30,11);
xd = hsline.XData;
yd = hsline.YData;
zd = hsline.ZData;
delete(hsline)
```

实现漫游

要产生漫游效果，请沿相同路径移动相机位置和相机目标。在此示例中，相机目标放置在 x 轴上远离相机五个元素的位置。在相机目标 x 位置数据上加上一个较小的数值，以防在出现 $xd(n) = xd(n+5)$ 的情况时相机和目标的位置在同一点：

- 更新相机位置和相机目标，使它们都沿着流线的坐标移动。
- 与相机一起移动光源。
- 调用 `drawnow` 以显示每次移动的结果。

```
for i=1:length(xd)-5
    campos([xd(i),yd(i),zd(i)])
    camtarget([xd(i+5)+min(xd)/500,yd(i),zd(i)])
    camlight(hlight,'headlight')
    drawnow
end
```

要创建相同数据的固定可视化绘图，请参阅 `coneplot`。

低级相机属性

本节内容
“可以设置的相机属性”（第 14-21 页）
“默认视点选择”（第 14-21 页）
“移入和移出场景”（第 14-22 页）
“使场景变大或变小”（第 14-23 页）
“围绕场景旋转”（第 14-23 页）
“旋转但不调整大小”（第 14-24 页）
“围绕观察轴旋转”（第 14-24 页）

可以设置的相机属性

相机图形基于一组坐标区属性，它们控制着相机的位置和方向。通常，如果使用相机命令（如 `campos`、`camtarget` 和 `camup`），则不需要直接访问这些属性。

属性	说明
<code>CameraPosition</code>	以坐标区单位指定视点的位置。
<code>CameraPositionMode</code>	在 <code>automatic</code> 模式下，由场景决定位置。在 <code>manual</code> 模式下，由您指定视点位置。
<code>CameraTarget</code>	指定坐标区中相机指向的位置。它与 <code>CameraPosition</code> 一起定义观察轴。
<code>CameraTargetMode</code>	在 <code>automatic</code> 模式下，MATLAB 将 <code>CameraTarget</code> 指定为坐标区图框的中心。在 <code>manual</code> 模式下，由您指定位置。
<code>CameraUpVector</code>	相机围绕观察轴的旋转由一个向量定义，指示方向为向上。
<code>CameraUpVectorMode</code>	在 <code>automatic</code> 模式下，MATLAB 将二维视图的 <code>y</code> 轴正方向和三维视图的 <code>z</code> 轴正方向作为向上向量。在 <code>manual</code> 模式下，由您指定方向。
<code>CameraViewAngle</code>	指定“镜头”的视野。如果您为 <code>CameraViewAngle</code> 指定了值，MATLAB 将不会拉伸坐标区以适应图窗。
<code>CameraViewAngleMode</code>	在 <code>automatic</code> 模式下，MATLAB 将视角调整为能够捕捉到整个场景的最小角度。在 <code>manual</code> 模式下，由您指定角度。 将 <code>CameraViewAngleMode</code> 设置为 <code>manual</code> 会覆盖伸展填充行为。
<code>Projection</code>	选择正交投影或透视投影。

默认视点选择

如果所有相机模式属性都设置为 `auto`（默认值），将由 MATLAB 自动控制视图，即，它会基于您希望场景填满位置矩形（由坐标区 `Position` 属性的宽度和高度分量决定）的假设来选择合适的值。

默认情况下，MATLAB

- 按照标准 MATLAB 二维或三维视图的场景方向设置 `CameraPosition`（请参阅 `view` 命令）
- 将 `CameraTarget` 设置为图框的中心
- 以二维视图的 `y` 方向和三维视图的 `z` 方向为向上方向来设置 `CameraUpVector`

- 将 CameraViewAngle 设置为能使场景填满位置矩形（由坐标区 Position 属性定义）的最小角度
- 使用正交投影

此默认行为一般情况下可以产生理想的结果。但是，您可以更改这些属性以产生对您有用的效果。

移入和移出场景

您可以将相机移动到由坐标区定义的三维空间中的任何位置。相机无论在什么位置，都始终指向其目标。当相机移动时，MATLAB 将更改相机视角，确保场景填满位置矩形。

穿过场景

通过移动相机穿过场景，可以产生漫游效果。要实现此目的，需要不断地更改 CameraPosition 属性，使相机朝着目标移动。由于相机是在这个空间内穿过，因此它在经过相机目标之后会拐弯。如果将 CameraViewAngleMode 设置为 manual，则当您每次移动相机时，MATLAB 不再自动调整场景大小。

如果您更新 CameraPosition 和 CameraTarget，效果是穿过场景时仍然面向移动方向。

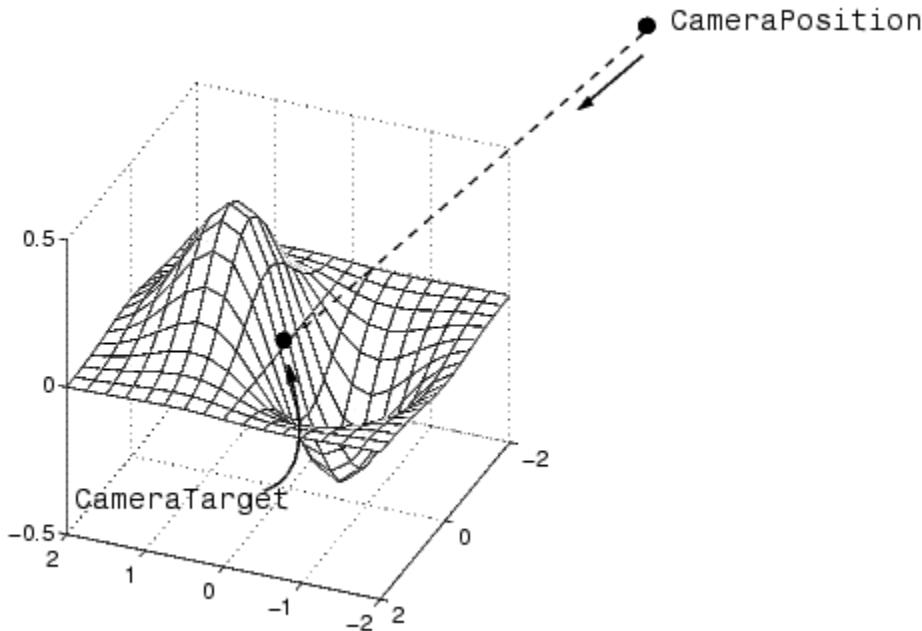
如果 Projection 设置为 perspective，透视失真量将随着相机靠近目标而增加，随着相机远离目标而减少。

示例 - 移近目标或移离目标

要沿着观察轴移动相机，您需要计算 CameraPosition 属性的新坐标。这是通过减去（移近目标）或加上（移离目标）相机位置与相机目标之间总距离的几分之一来实现的。

如果参数 dist 为正，则函数 movecamera 按移入场景计算新的 CameraPosition；如果参数 dist 为负，则按移出场景计算新位置。

```
function movecamera(dist) %dist in the range [-1 1]
set(gca,'CameraViewAngleMode','manual')
newcp = cpos - dist * (cpos - ctarg);
set(gca,'CameraPosition',newcp)
function out = cpos
out = get(gca,'CameraPosition');
function out = ctarg
out = get(gca,'CameraTarget');
```



将 CameraViewAngleMode 设置为 manual 可能会导致纵横比突然变化。

使场景变大或变小

调整 CameraViewAngle 属性可使场景的视图变大或变小。角度越大，视图包含的区域就越大，因此场景中的对象就显得越小。同理，角度越小，对象就显得越大。

更改 CameraViewAngle 会使场景变大或变小，而不会影响相机的位置。如果您希望放大，而不移动视点越过场景中将不再存在的对象（假如您更改了相机位置，就可能会发生这种情况），则可以使用这种方法。此外，更改 CameraViewAngle 不会像图窗属性 Projection 设置为 perspective 时更改 CameraPosition 那样影响应用于场景的透视量。

围绕场景旋转

您可以使用 view 命令通过更改方位角使视点围绕 z 轴旋转，通过更改仰角使视点围绕方位角旋转。这样可以产生围绕球体（半径等于观察轴的长度）曲面上的场景移动相机的效果。您可以通过更改 CameraPosition 实现相同的效果，但需要您执行调用 view 时由 MATLAB 完成的那些计算。

例如，函数 orbit 围绕场景移动相机。

```
function orbit(deg)
[az, el] = view;
rotvec = 0:deg/10:deg;
for i = 1:length(rotvec)
    view([az+rotvec(i) el])
    drawnow
end
```

旋转但不调整大小

当 CameraViewAngleMode 为 `auto` 时, MATLAB 将计算能够使场景大到足以填满坐标区位置矩形的 CameraViewAngle。这会在场景旋转过程中引起明显的大小变化。为了防止旋转时大小改变, 您需要将 CameraViewAngleMode 设置为 `manual` (当您为 CameraViewAngle 属性指定值时, 会自动启用此设置)。要在 `orbit` 函数中执行此操作, 请添加以下语句

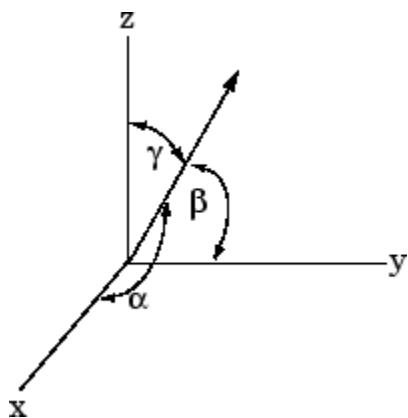
```
set(gca,'CameraViewAngleMode','manual')
```

围绕观察轴旋转

通过指定定义为向上的方向, 可以更改场景的方向。默认情况下, MATLAB 将二维视图的 y 轴 (`CameraUpVector` 为 `[0 1 0]`) 和三维视图的 z 轴 (`CameraUpVector` 为 `[0 0 1]`) 定义为向上。但是, 您可以指定任意方向作为向上的方向。

`CameraUpVector` 属性定义的向量构成相机坐标系的一个坐标轴。在内部, MATLAB 通过将指定的向量投影到垂直于相机方向 (即观察轴) 的平面上来确定相机向上向量的实际方向。这简化了 `CameraUpVector` 属性的设定, 因为它不需要位于此平面上。

在很多情况下, 您可能会发现根据与 x、y 和 z 轴之间形成的角度来可视化所需的向上向量很方便。然后, 可以使用方向余弦将角度转换为向量分量。对于单位向量, 表达式简化为



其中的角 α 、 β 和 γ 以度为单位指定。

```
XComponent = cos(alpha*(pi/180));
YComponent = cos(beta*(pi/180));
ZComponent = cos(gamma*(pi/180));
```

有关方向余弦的详细说明, 请查阅有关向量分析的数学书籍。

计算相机的向上向量

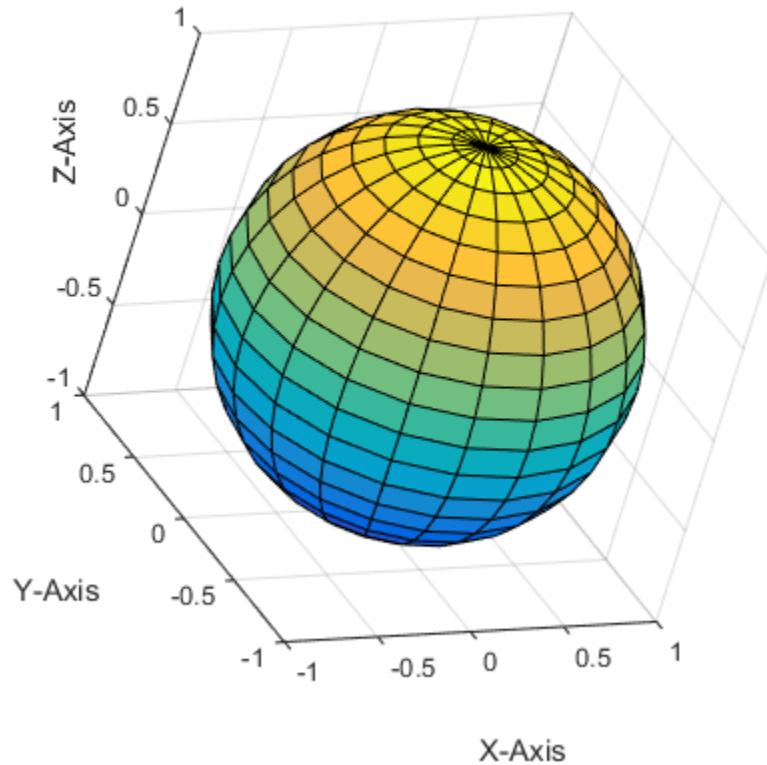
要指定与 z 轴呈 30° 夹角并且位于 y-z 平面上的向上向量, 请使用以下表达式

```
upvec = [cos(90*(pi/180)),cos(60*(pi/180)),cos(30*(pi/180))];
```

然后设置 `CameraUpVector` 属性。

```
set(gca,'CameraUpVector',upvec)
```

按此方向绘制球体将得到



了解视图投影

本节内容

- “两种投影类型” (第 14-26 页)
- “投影类型和相机位置” (第 14-27 页)

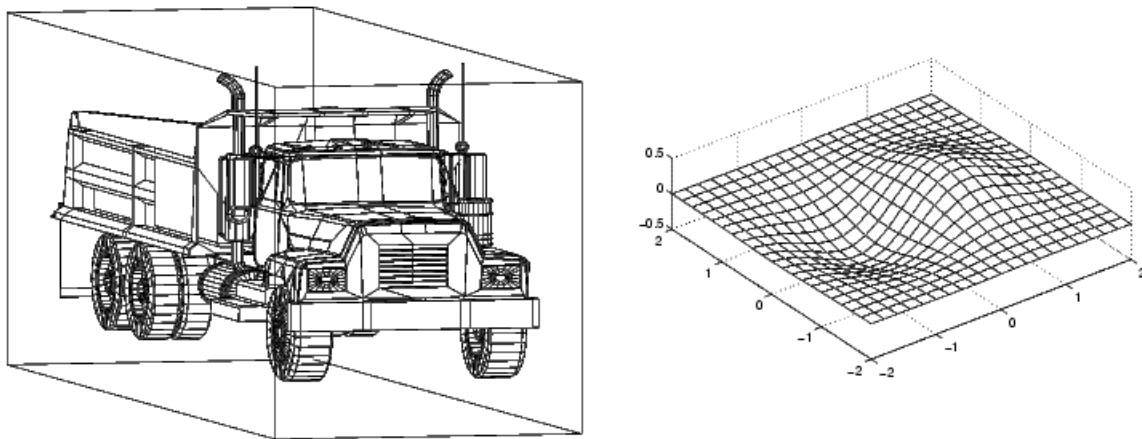
两种投影类型

MATLAB 图形支持通过正交投影和透视投影两种类型来显示三维图形。具体选择哪一种取决于您要显示的图形类型：

- **orthographic** 将观察体投影为矩形平行六面体（即相对的面互相平行的箱体）。与相机之间的相对距离不影响对象的大小。当需要保持对象的实际大小以及对象之间的角度时，这种投影方式非常有用。
- **perspective** 将观察体投影为棱锥的截头锥体（平行于底部截掉头部的棱锥）。距离会产生前缩透视效果，距相机越远的对象越小。当您要显示真实对象的逼真视图时，这种投影方式非常有用。

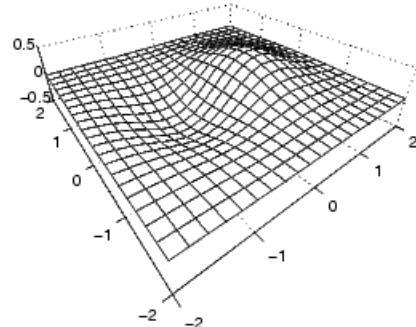
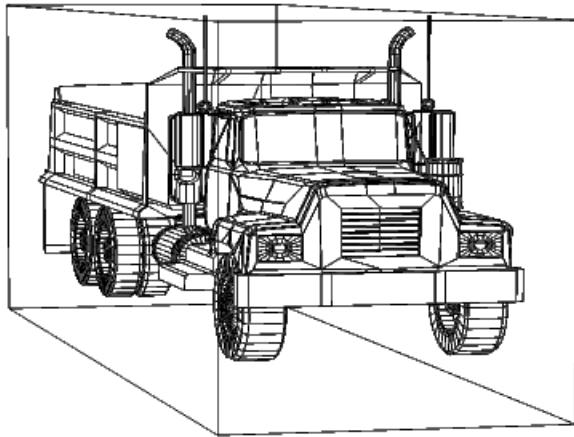
默认情况下，MATLAB 使用正交投影方式来显示对象。您可以使用 **camproj** 命令设置投影类型。

下图显示了自卸车的绘图（使用 **patch** 创建）和数学函数的曲面图，二者都使用正交投影。



如果测量自卸车箱体前后面的宽度，您会发现二者相等。这张图看起来不自然，因为它缺少观察具有景深的真实对象时能够看到的明显透视效果。曲面图则能精确表示矩形空间内的函数值。

现在观察同样的图形对象在添加透视之后的效果。自卸车看起来自然多了，因为距离观察者较远的卡车部分看起来更小。这种投影模拟了人类视觉的工作原理。曲面图看起来则失真了。

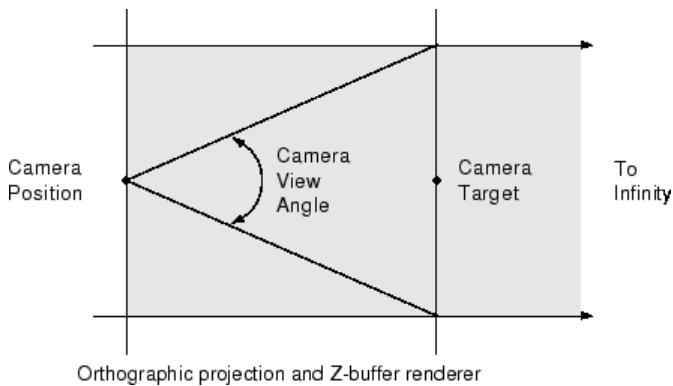


投影类型和相机位置

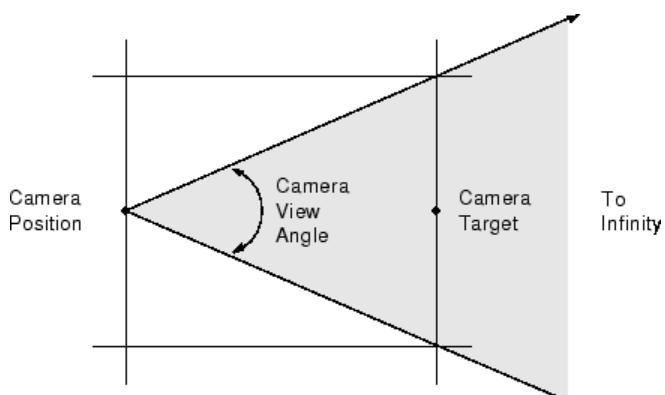
默认情况下，MATLAB 会调整 `CameraPosition`、`CameraTarget` 和 `CameraViewAngle` 属性，使相机指向场景中心并包含坐标区内的所有图形对象。如果您放置的相机位置导致一些图形对象在相机后面，则显示的场景同时受到坐标区属性 `Projection` 和图窗属性 `Renderer` 的影响。下面总结了投影类型与渲染方法之间的相互影响。

	正交	透视
OpenGL®	<code>CameraViewAngle</code> 决定 <code>CameraTarget</code> 处的场景范围。	<code>CameraViewAngle</code> 决定场景范围从 <code>CameraPosition</code> 到无穷远。
Painters	显示所有对象，无论 <code>CameraPosition</code> 如何。	如果图形对象位于 <code>CameraPosition</code> 后面，则不建议使用。

下图说明使用正交投影和 OpenGL 时您所看到的内容（灰色区域）。相机前面的所有对象都可见。

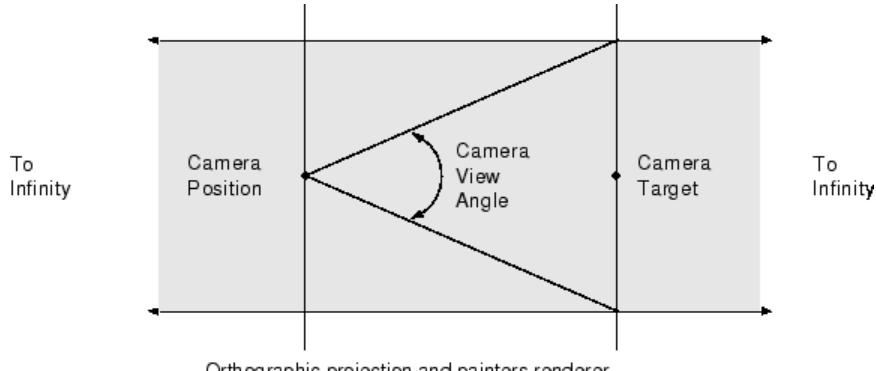


在透视投影中，您只能看到相机视角锥形区域内可见的对象。



Perspective projection and Z-buffer renderer

Painters 渲染方法不太适合在三维空间移动相机，因为 MATLAB 不会沿观察轴进行裁剪。Painters 方法中的正交投影会使场景中包含的所有对象都可见，而不管相机在什么位置。



Orthographic projection and painters renderer

打印三维场景

上一节描述的效果可展现在硬拷贝输出中。您应该显式指定 OpenGL 打印，以获取屏幕上显示的效果（使用 print 命令的 -opengl 选项）。

显示位图图像

- “处理 MATLAB 图形中的图像” (第 15-2 页)
- “图像类型” (第 15-4 页)
- “8 位和 16 位图像” (第 15-7 页)
- “读取、写入和查询图像文件” (第 15-12 页)
- “显示图形图像” (第 15-15 页)
- “图像对象及其属性” (第 15-18 页)
- “打印图像” (第 15-23 页)
- “转换图像图形或数据类型” (第 15-24 页)
- “显示图像数据” (第 15-25 页)

处理 MATLAB 图形中的图像

本节内容

“什么是图像数据？”（第 15-2 页）

“支持的图像格式”（第 15-3 页）

什么是图像数据？

MATLAB 的基本数据结构是数组，即实数或复数元素的有序集合。数组天然适合表示图像、实数值、颜色或强度数据的有序集合。（数组非常适合复数值图像。）

在 MATLAB 工作区中，大多数图像表示为二维数组（矩阵），其中矩阵的每个元素对应所显示图像的一个像素。例如，由 200 行和 300 列不同颜色的点组成的图像保存为一个 200×300 的矩阵。有些图像，如 RGB，需要三维数组，其中三个维度的第一个平面表示红色像素强度，第二个平面表示绿色像素强度，第三个平面表示蓝色像素强度。

这样的转换使用户可以像处理其他类型矩阵数据一样处理图形文件格式图像。例如，您可以使用普通的矩阵下标表示法从图像矩阵中取得一个像素：

I(2,15)

此命令返回图像 I 第 2 行第 15 列的像素值。

以下部分描述了不同数据和图像类型，并详细说明了如何读取、写入、使用和显示图形图像；如何在显示时修改图像的显示属性和纵横比；如何打印图像；如何转换图像的数据类型或图形格式。

数据类型

MATLAB 数学支持三种不同的数值类用于图像显示：

- 双精度浮点数 (**double**)
- 16 位无符号整数 (**uint16**)
- 8 位无符号整数 (**uint8**)

图像显示命令根据数据存入的数值类解释数据值。“8 位和 16 位图像”（第 15-7 页）中介绍了 8 位和 16 位图像存储的内部工作原理的详情。

默认情况下，大多数数据都使用 **double** 类数组。这些数组中的数据存储为双精度（64 位）浮点值。所有的 MATLAB 函数和功能都能使用这些数组。

但是，对于以 MATLAB 函数支持的图形文件格式存储的图像而言，这种数据表现形式却并不理想。这样的图像中的像素数目可能非常巨大；如一个 1000×1000 的图像会有一百万个像素。由于至少要用一个数组元素表示一个像素，因此如果图像以 **double** 类存储，那么它至少需要 8 兆内存。

为了减小内存需求，您可以将图像数据存储在 **uint8** 和 **uint16** 这两类数组中。这些数组中的数据存储为 8 位或 16 位无符号整数。同样的数据，这些数组只需要 **double** 数组的八分之一或四分之一的存储容量。

位深

MATLAB 输入函数读取任何一个支持的图形文件格式的最常使用的位深（每像素位数）。当数据存于内存中时，它可能存储为 **uint8**、**uint16** 或 **double** 类的形式。有关哪种位深适合哪种支持格式的详细信息，请参阅 **imread** 和 **imwrite**。

支持的图像格式

MATLAB 命令可以读取、写入和显示以下几种图像的图形文件格式。对于 MATLAB 生成的图像，一旦显示一个图形文件格式的图像，该图像便会成为一个图像对象。MATLAB 支持如下图形文件格式以及其他格式：

- BMP (Microsoft® Windows® 位图)
- GIF (图形交换文件)
- HDF (分层数据格式)
- JPEG (联合图像专家组)
- PCX (画笔)
- PNG (可移植网络图形)
- TIFF (标记图像文件格式)
- XWD (X 窗口转储)

有关这些格式支持的位深和图像类型的详细信息，请参阅 **imread** 和 **imwrite**。

图像类型

本节内容

- “索引图像”（第 15-4 页）
- “灰度（强度）图像”（第 15-5 页）
- “RGB（真彩色）图像”（第 15-5 页）

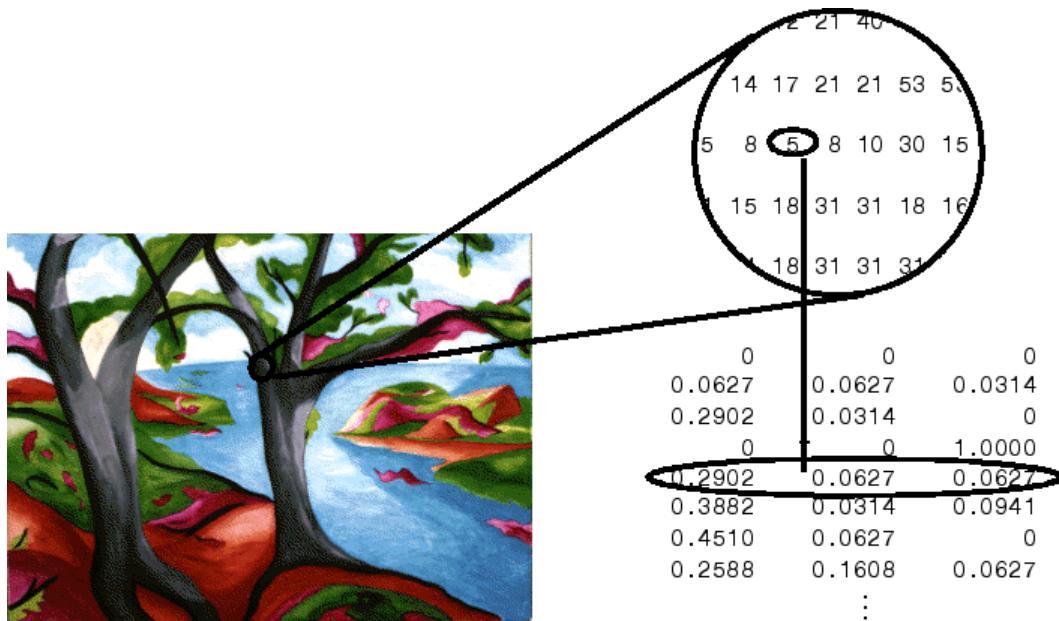
索引图像

索引图像由数据矩阵 **X** 和颜色图矩阵 **map** 组成。**map** 是一个 **double** 类的 $m \times 3$ 数组，由 [0, 1] 范围内的浮点值组成。**map** 的每一行指定单一颜色的红、绿和蓝分量。索引图像使用像素值到颜色图值的“直接映射”。每个图像像素的颜色是以 **X** 中的对应值为索引求得的 **map** 中的值。因此 **X** 值必须是整数。值 1 指向 **map** 中第一行，值 2 指向第二行，以此类推。使用以下语句显示索引图像

```
image(X); colormap(map)
```

颜色图通常与索引图像一起存储，而且当您使用 **imread** 函数时，颜色图会随着索引图像一起加载。但您不仅限于使用默认的颜色图，您可以根据意愿选择任何一个颜色图。属性 **CDataMapping** 的说明描述了如何修改所使用映射的类型。

下一幅图展示了索引图像的结构。图像中的像素以整数表示，该整数是颜色图中存储的颜色值的指针（索引）。

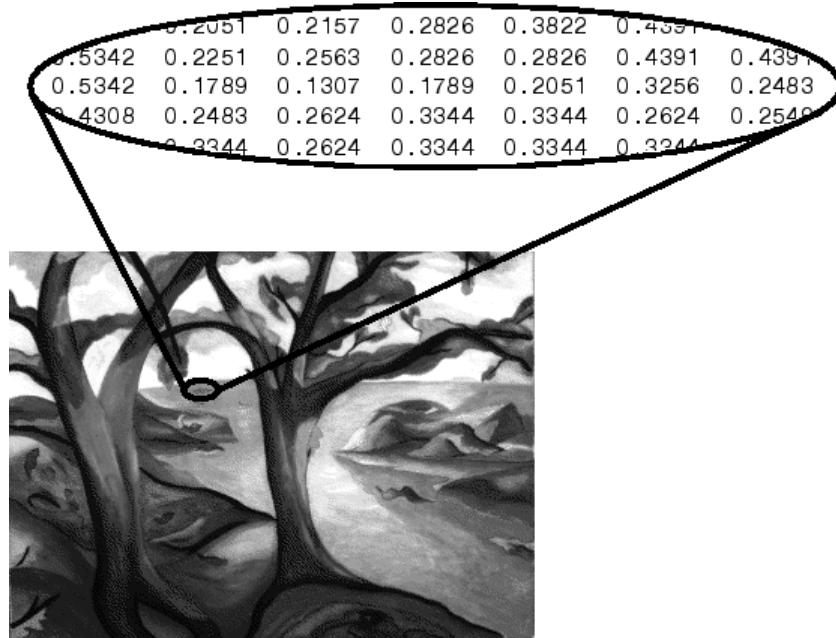


图像矩阵和颜色图中值的关系取决于图像矩阵的类。如果图像矩阵是 **double** 类，那么值 1 指向颜色图中的第一行，值 2 指向第二行，以此类推。如果图像矩阵是 **uint8** 或 **uint16** 类，那么有个偏移 - 值 0 指向颜色图中的第一行，值 1 指向第二行，以此类推。在图形文件格式中使用偏移是为了使可支持的颜色数目达到最大。在上面的图像中，图像矩阵是 **double** 类。由于没有偏移，值 5 指向颜色图的第五行。

灰度（强度）图像

灰度图像有时也称为强度图像，它是一个数据矩阵 I ，其中的值表示某一范围内的强度。灰度图像表示为单个矩阵，矩阵的每个元素对应一个图像像素。矩阵可能是 **double**、**uint8** 或 **uint16** 类。尽管灰度图像很少与颜色图一起保存，但它仍需要用颜色图来显示。实际上，灰度图像被当做索引图像处理。

此图窗描绘了一幅 **double** 类的灰度图像。



要显示灰度图像，使请用 **imagesc**（“图像缩放”）函数，它允许设置强度图像的范围。**imagesc** 会调整图像数据以使用整个颜色图。使用 **imagesc** 的双输入形式显示灰度图像，例如：

```
imagesc(I,[0 1]); colormap(gray);
```

imagesc 的第二个输入参数指定了所需的强度范围。**imagesc** 函数通过将范围内的第一个值（通常是 0）映射到第一个颜色图条目，第二个值（通常是 1）映射到最后一个颜色图条目来显示 I 。这两者之间的颜色值在余下的颜色图颜色中呈线性分布。

尽管常规的做法是使用灰度颜色图显示灰度图像，但您也可以使用其他颜色图。例如，以下语句以不同深浅的蓝色和绿色显示灰度图像 I ：

```
imagesc(I,[0 1]); colormap(winter);
```

要将具有任意值范围的矩阵 A 显示为灰度图像，请使用 **imagesc** 的单参数形式。使用一个输入参数，**imagesc** 会将数据矩阵的最小值映射为第一个颜色图条目，最大值映射为最后一个颜色图条目。例如：这两个线条是等价的：

```
imagesc(A); colormap(gray)
imagesc(A,[min(A(:)) max(A(:))]); colormap(gray)
```

RGB（真彩色）图像

RGB 图像，有时称为真彩色图像，以 $m \times n \times 3$ 数据数组形式存储，该数组定义了对应图像每个像素的红色、绿色和蓝色分量。RGB 图像不使用调色板。每个像素的颜色由存储在每个像素位置的颜色平面的红

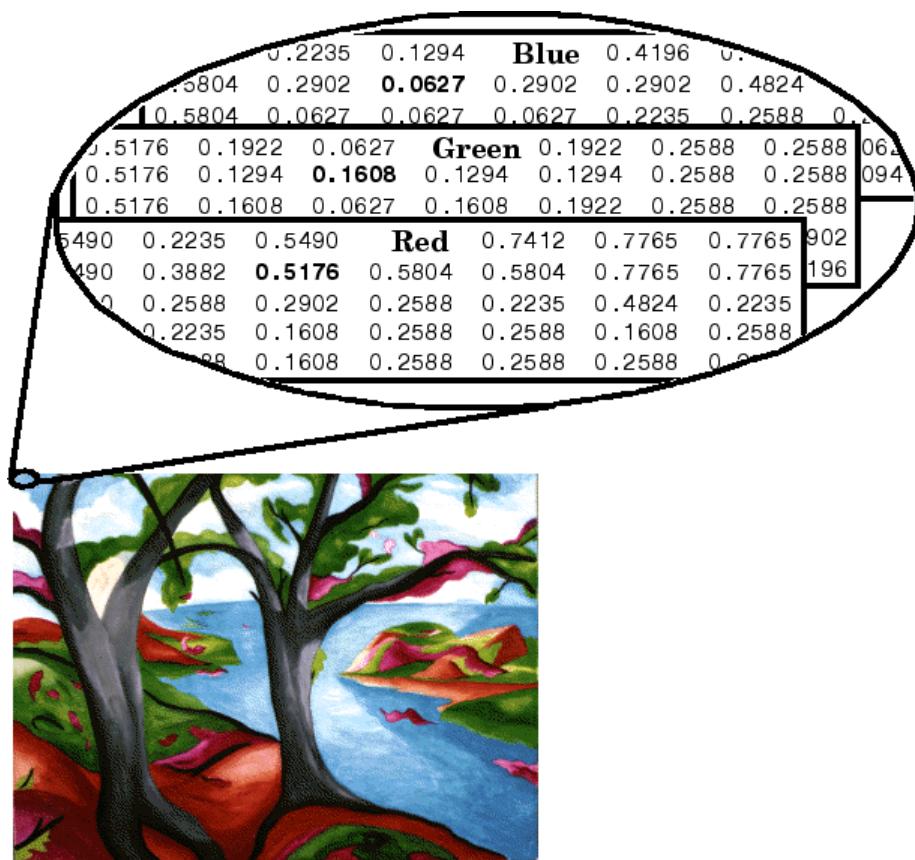
色、绿色和蓝色强度决定。图形文件格式将 RGB 图像存储为 24 位图像，其中红色、绿色和蓝色分量各占 8 位。这样便可能产生 1600 万种颜色。这种可复制现实中图像的精度由此被称为“真彩色图像”。

RGB MATLAB 数组可以是 **double**、**uint8** 或 **uint16** 类。在 **double** 类的 RGB 数组中，每个颜色分量都是 0 到 1 之间的值。颜色分量为 (0,0,0) 的像素显示为黑色，颜色分量为 (1,1,1) 的像素显示为白色。每个像素的三个颜色分量沿数据数组的第三个维度存储。例如像素 (10,5) 的红色、绿色和蓝色分量分别存储在 **RGB(10,5,1)**、**RGB(10,5,2)** 和 **RGB(10,5,3)** 中。

要显示真彩色图像 **RGB**，请使用 **image** 函数：

```
image(RGB)
```

下一幅图显示 **double** 类的 RGB 图像。



要确定 (2,3) 处像素的颜色，请查看存储在 (2,3,1:3) 的 RGB 三元组：假设 (2,3,1) 包含值 **0.5176**，(2,3,2) 包含 **0.1608**，(2,3,3) 包含 **0.0627**。则 (2,3) 处的像素颜色是

0.5176 0.1608 0.0627

8 位和 16 位图像

本节内容

- “索引图像”（第 15-7 页）
- “强度图像”（第 15-7 页）
- “RGB 图像”（第 15-8 页）
- “uint8 和 uint16 的数学运算支持”（第 15-8 页）
- “其他 8 位和 16 位数组支持”（第 15-8 页）
- “将 8 位 RGB 图像转换为灰度图像”（第 15-9 页）
- “图像类型和数值类摘要”（第 15-11 页）

索引图像

双精度（64 位）浮点数是数值数据默认的 MATLAB 表现形式。但为了降低使用图像的内存需求，您可以分别使用数值类 `uint8` 和 `uint16` 将图像保存为 8 位和 16 位无符号整数。数据矩阵为 `uint8` 类的图像称为 8 位图像；数据矩阵为 `uint16` 类的图像称为 16 位图像。

`image` 函数可以显示 8 位或 16 位图像，而无需将其转换为双精度。但如果图像矩阵为 `uint8` 或 `uint16`，则 `image` 在解释矩阵值时会稍微有些不同。具体的解释取决于图像类型。

如果 `X` 的类是 `uint8` 或 `uint16`，那么在被用作颜色图索引之前，它的值会偏移 1。值 0 指向颜色图中的第一行，值 1 指向第二行，以此类推。`image` 命令自动提供合适的偏移，从而无论 `X` 是 `double`、`uint8` 还是 `uint16`，显示方法都一样：

```
image(X); colormap(map);
```

`uint8` 和 `uint16` 数据的颜色图索引偏移是用于支持标准图形文件格式的，这种文件通常以索引形式与包含 256 个颜色的颜色图一起存储。偏移可以让您使用更节约内存的 `uint8` 和 `uint16` 数组操作和显示这种形式的图像。

由于偏移，必须加 1 将 `uint8` 或 `uint16` 索引图像转换为 `double` 类。例如：

```
X64 = double(X8) + 1;
      or
X64 = double(X16) + 1;
```

反过来，减 1 可将 `double` 索引图像转换为 `uint8` 或 `uint16` 类：

```
X8 = uint8(X64 - 1);
      or
X16 = uint16(X64 - 1);
```

强度图像

`double` 图像数组范围通常是 $[0, 1]$ ，而 8 位强度图像的范围通常是 $[0, 255]$ ，16 位强度图像的范围通常是 $[0, 65535]$ 。使用如下命令以灰度颜色图显示 8 位强度图像：

```
imagesc(I,[0 255]); colormap(gray);
```

要将强度图像从 `double` 类转换为 `uint16` 类，首先乘以 65535：

```
I16 = uint16(round(I64*65535));
```

反过来，将 `uint16` 强度图像转换为 `double` 类之后，要除以 65535：

```
I64 = double(I16)/65535;
```

RGB 图像

8 位 RGB 图像的颜色分量是 [0, 255] 范围内的整数值，而非 [0, 1] 范围内的浮点值。颜色分量为 (255,255,255) 的像素显示为白色。`image` 命令会正确显示 RGB 图像，无论它的类是 `double`、`uint8`，还是 `uint16`：

```
image(RGB);
```

要将 RGB 图像从 `double` 类转换为 `uint8` 类，首先乘以 255：

```
RGB8 = uint8(round(RGB64*255));
```

反过来，将 `uint8` RGB 图像转换为 `double` 类之后，要除以 255：

```
RGB64 = double(RGB8)/255
```

要将 RGB 图像从 `double` 转换为 `uint16`，首先乘以 65535：

```
RGB16 = uint16(round(RGB64*65535));
```

反过来，将 `uint16` RGB 图像转换为 `double` 类之后，要除以 65535：

```
RGB64 = double(RGB16)/65535;
```

uint8 和 uint16 的数学运算支持

要在以下 MATLAB 函数中使用 `uint8` 和 `uint16` 数据，需要先将数据转换为 `double` 类型：

- `conv2`
- `convn`
- `fft2`
- `fftn`

例如，如果 `X` 是一个 `uint8` 图像，那要将其数据转换为 `double` 类型便可使用以下语句：

```
fft(double(X))
```

在这种情况下，输出始终为 `double`。

`sum` 函数返回与输入同一类型的结果，但它还提供了一个选项，可以使用双精度计算。

MATLAB 整数数学

有关数学函数如何使用非双精度数据类型的详细信息，请参阅“整数类的算术运算”。

大多数 Image Processing Toolbox™ 函数都接受 `uint8` 和 `uint16` 输入。如果您打算对 `uint8` 或 `uint16` 数据进行复杂的图像处理，请考虑在 MATLAB 计算环境中包含该工具箱。

其他 8 位和 16 位数组支持

您可以对 `uint8` 和 `uint16` 数组执行其他几种操作，包括：

- 使用 `reshape`、`cat`、`permute` 函数以及 `[]` 和 `'` 运算符重构、重新排序和串联数组
- 使用 `save` 和 `load` 将 `uint8` 和 `uint16` 数组保存及加载到 MAT 文件。（请记住，如果要加载或保存图形格式文件图像，则必须使用 `imread` 和 `imwrite` 命令代替。）
- 使用 `find` 定位 `uint8` 和 `uint16` 数组中非零元素的索引。但返回的数组始终为 `double` 类。
- 关系运算符

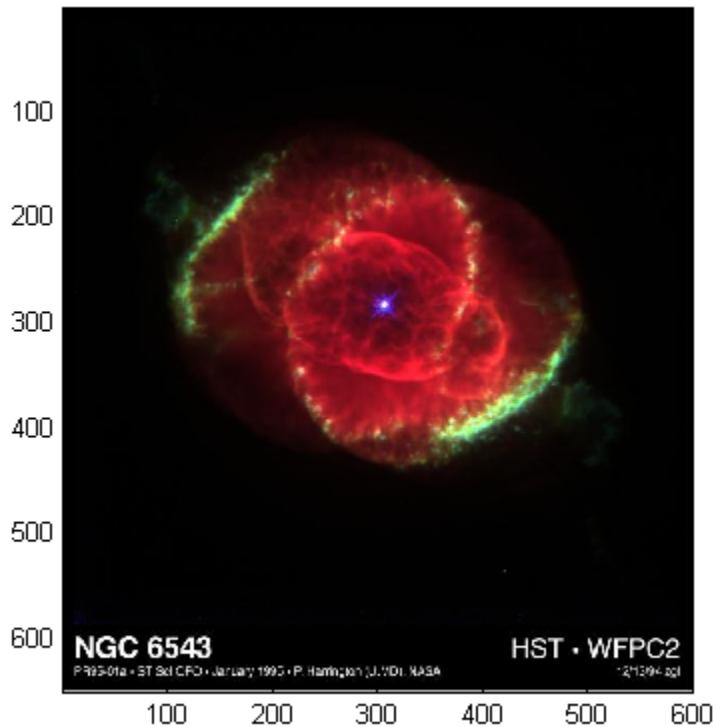
将 8 位 RGB 图像转换为灰度图像

您可以对整型数据执行数学运算，这样可以直接转换图像类型，而无需先转换图像数据的数值类。

此示例将一个 8 位 RGB 图像读入 MATLAB 变量，并将其转换为灰度图像：

```
rgb_img = imread('ngc6543a.jpg'); % Load the image
image(rgb_img) % Display the RGB image

axis image;
```



注意 该图像是在太空望远镜科学研究所（由大学天文研究联合组织管理）支持下根据 NASA 合同 NAs5-26555 创建的，并得到 AURA/STScI 的复制许可。使用者可以免版权费获得由 AURA/ STScI 制作的图像的数字版本。感谢：J.P. Harrington 和 K.J. Orkowsky (马里兰大学) 以及 NASA。

基于 NTSC 标准合并 RGB 值，以此方式来计算单色亮度，这会将与眼睛敏感度相关的系数应用到 RGB 颜色：

```
I = .2989*rgb_img(:,:,1)...
+.5870*rgb_img(:,:,2)...
+.1140*rgb_img(:,:,3);
```

I 是具有整数值的强度图像，整数值范围从最小值零开始：

```
min(I(:))
ans =
0
```

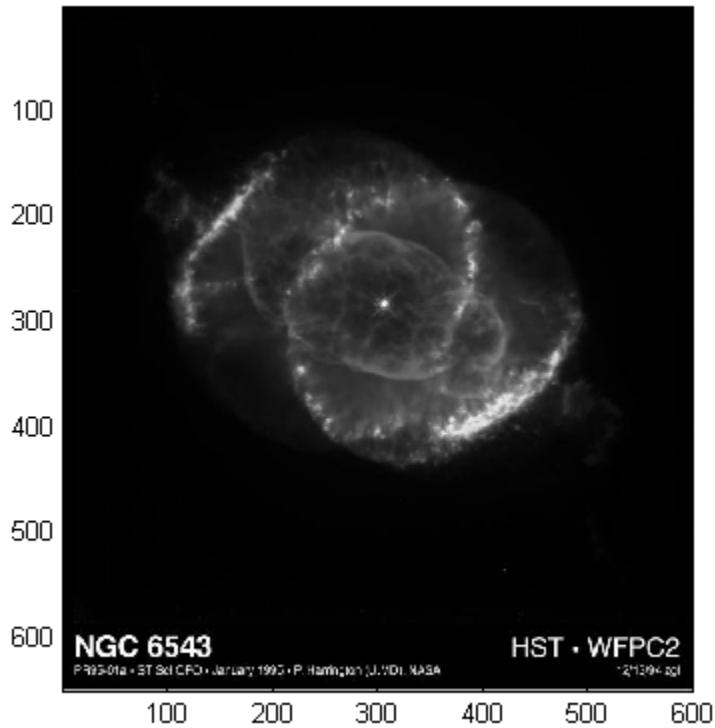
到最大值 255：

```
max(I(:))
ans =
255
```

要显示图像，请使用 256 值的灰度颜色图。这样可以避免调整数据到颜色的映射，当使用不同尺寸的颜色图时通常需要进行此项调整。如果颜色图未包含每个数据值的项，则需使用 `imagesc` 函数。

现在使用灰度颜色图在新图窗中显示图像：

```
figure; colormap(gray(256)); image(I);
axis image;
```



相关信息

其他的颜色范围从深色到浅色连续变化的颜色图也可生成可用的图像。例如，使用 `colormap(summer(256))` 生成经典的示波器样式。有关更多选项，请参阅 `colormap`。

brighten 函数可以让您增加或减少颜色图中的强度以补偿电脑显示差异或增强图像中模糊或明亮区域的可见性（其代价是另一端的颜色会有损失）。

图像类型和数值类摘要

下表总结了如何根据图像类型和数据类将数据矩阵元素解释为像素颜色。

图像类型	双精度数据	uint8 或 uint16 数据
索引	图像是由 $[1, p]$ 范围内的整数组成的 $m \times n$ 数组。 颜色图是由 $[0, 1]$ 范围内的浮点值组成的 $p \times 3$ 数组。	图像是由 $[0, p-1]$ 范围内的整数组成的 $m \times n$ 数组。 颜色图是由 $[0, 1]$ 范围内的浮点值组成的 $p \times 3$ 数组。
强度	图像是由经过线性调整以生成颜色图索引的浮点值组成的 $m \times n$ 数组。这些值通常的范围为 $[0, 1]$ 。 颜色图是由 $[0, 1]$ 范围内的浮点值组成的 $p \times 3$ 数组，通常为灰度颜色图。	图像是由经过线性调整以生成颜色图索引的整数组成的 $m \times n$ 数组。这些值的范围是 $[0, 255]$ 或 $[0, 65535]$ 。 颜色图是由 $[0, 1]$ 范围内的浮点值组成的 $p \times 3$ 数组，通常为灰度颜色图。
RGB (真彩色)	图像是由 $[0, 1]$ 范围内的浮点值组成的 $m \times n \times 3$ 数组。	图像是由 $[0, 255]$ 或 $[0, 65535]$ 范围内的整数组成的 $m \times n \times 3$ 数组。

读取、写入和查询图像文件

本节内容

- “使用图像格式” (第 15-12 页)
- “读取图形图像” (第 15-12 页)
- “写入图形图像” (第 15-13 页)
- “划分图形图像的分集 (裁剪)” (第 15-13 页)
- “获取有关图形文件的信息” (第 15-14 页)

使用图像格式

从本质上说，图形文件格式图像并非以 MATLAB 矩阵的形式存储，甚至不必以矩阵的形式存储。大多数图形文件都以包含特定格式信息标签的标头开头，然后是能够以连续流方式读取的位图数据。因此，不能使用标准的 MATLAB I/O 命令 `load` 和 `save` 来读取和写入图形文件格式图像。

调用专门的 MATLAB 函数从图形文件格式读取和写入图像数据：

- 要读取图形文件格式图像，请使用 `imread`。
- 要写入图形文件格式图像，请使用 `imwrite`。
- 要获取有关图形文件图像的性质信息，请使用 `imfinfo`。

下表更加清楚地阐明了具体的 MATLAB 命令所该采用的具体的图像类型。

过程	要使用的函数
以 MAT 文件的形式加载或保存矩阵。	<code>load</code> <code>save</code>
加载或保存图形文件格式图像，如 BMP、TIFF。	<code>imread</code> <code>imwrite</code>
显示加载到 MATLAB 工作区的任何图像。	<code>image</code> <code>imagesc</code>
实用工具	<code>imfinfo</code> <code>ind2rgb</code>

读取图形图像

`imread` 函数能够按照任意受支持的位深读取任意受支持的图形文件中的图像。所读取的大多数图像均为 8 位。将这些图像读入内存后，以 `uint8` 类的形式来存储这些图像。这一规则的主要例外是 MATLAB 支持 16 位数据的 PNG 和 TIFF 图像；如果您读取 16 位的 PNG 或 TIFF 图像，那么该图像以 `uint16` 类的形式保存。

注意 对于索引图像，`imread` 始终将颜色图读入 `double` 类的数组，即使图像数组本身可能属于 `uint8` 或 `uint16` 类，也是如此。

以下命令可将图像 `ngc6543a.jpg` 读入工作区变量 `RGB`, 然后用 `image` 函数来显示该图像:

```
RGB = imread('ngc6543a.jpg');
image(RGB)
```

您可以使用 `imwrite` 函数写入 (保存) 图像数据。以下语句

```
load clown % An image that is included with MATLAB
imwrite(X,map,'clown.bmp')
```

可用于创建包含 `clown` 图像的 BMP 文件。

写入图形图像

当您使用 `imwrite` 保存图像时, 默认的行为是自动将位深减小为 `uint8`。MATLAB 中所使用的图像大多数为 8 位, 而且大多数图形文件格式图像不需要双精度数据。对于以 `uint8` 形式保存图像数据的规则而言, PNG 和 TIFF 图像是例外的, 它们可保存为 `uint16`。因为这两种格式支持 16 位数据, 所以您可以通过将 `uint16` 指定为 `imwrite` 的数据类型以覆盖 MATLAB 的默认行为。以下示例说明了使用 `imwrite` 写入 16 位 PNG 文件的方式。

```
imwrite(I,'clown.png','BitDepth',16);
```

划分图形图像的分集 (裁剪)

有时候您只想使用图像文件的一部分或要将其划分成几个子区域。在命令行中, 指定您想要使用的矩形子区域的内部坐标, 并将它保存到文件中。如果您不知道子区域的各边角点的坐标, 则以交互的方式进行选择, 正如以下示例所示:

```
% Read RGB image from graphics file.
im = imread('street2.jpg');

% Display image with true aspect ratio
image(im); axis image

% Use ginput to select corner points of a rectangular
% region by pointing and clicking the mouse twice
p = ginput(2);

% Get the x and y corner coordinates as integers
sp(1) = min(floor(p(1)), floor(p(2))); %xmin
sp(2) = min(floor(p(3)), floor(p(4))); %ymin
sp(3) = max(ceil(p(1)), ceil(p(2))); %xmax
sp(4) = max(ceil(p(3)), ceil(p(4))); %ymax

% Index into the original image to create the new image
MM = im(sp(2):sp(4), sp(1): sp(3),:);

% Display the subsetted image with appropriate axis ratio
figure; image(MM); axis image

% Write image to graphics file.
imwrite(MM,'street2_cropped.tif')
```

如果您知道图像边角点的坐标, 那么在上例中, 您可以手动定义 `sp` 而不必使用 `ginput`。

您还可以在对图像执行交互操作时，显示一个“橡皮筋框”以划分图像的子集。有关详细信息，请参阅 `rbbox` 代码示例。有关详细信息，请参阅 `ginput` 和 `image` 函数的文档。

获取有关图形文件的信息

`imfinfo` 函数可以让您获取先前列出的所有格式图形文件的相关信息。您所获取的信息取决于文件类型，但至少始终包含以下信息：

- 文件名称，包括文件夹路径（如果文件不在当前文件夹中）
- 文件格式
- 文件格式的版本号
- 文件修改日期
- 文件大小（以字节为单位）
- 图像宽度（以像素为单位）
- 图像高度（以像素为单位）
- 每像素位数
- 图像类型：RGB（真彩色）、强度（灰度）或索引

显示图形图像

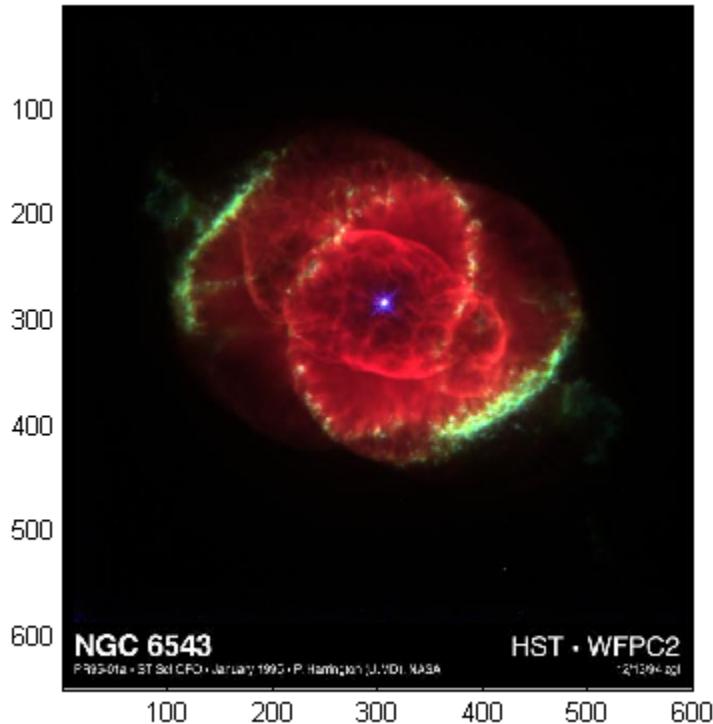
本节内容

- “图像类型和显示方法”（第 15-15 页）
- “控制纵横比和显示尺寸”（第 15-16 页）

图像类型和显示方法

要显示图形文件图像，请使用 `image` 或 `imagesc`。例如，将图像 `ngc6543a.jpg` 读取到变量 `RGB` 并使用 `image` 函数来显示图像。使用 `axis` 命令将坐标区的纵横比改为实际比例。

```
RGB = imread('ngc6543a.jpg');
image(RGB);
axis image;
```



下表总结了三种图像的显示方法。

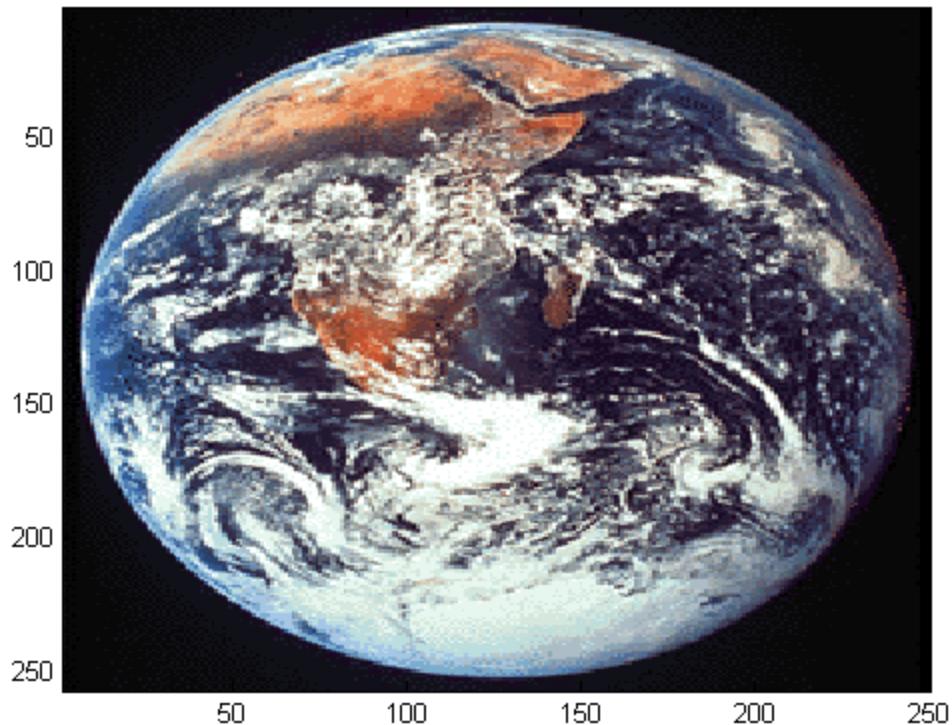
图像类型	显示命令	使用颜色图颜色
索引	<code>image(X); colormap(map)</code>	Yes
强度	<code>imagesc(I,[0 1]); colormap(gray)</code>	Yes
RGB (真彩色)	<code>image(RGB)</code>	No

控制纵横比和显示尺寸

`image` 函数按照默认大小的图窗和坐标区来显示图像。图像会拉伸或收缩以适合显示区域。有时候您可能希望显示的纵横比与图像数据矩阵的纵横比相匹配。最简单的做法是使用 `axis image` 命令。

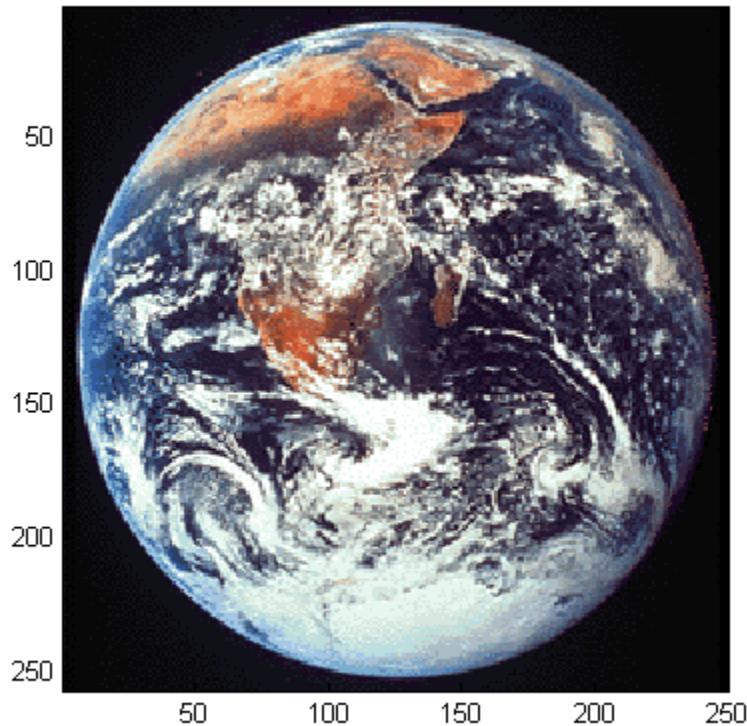
例如，这些命令使用默认的图窗和坐标区位置来显示 `earth` 图像：

```
load earth  
image(X)  
colormap(map)
```



显示由于拉伸图像而形成的细长球体以适合坐标区的位置。使用 `axis image` 命令强制纵横比为一比一。

```
axis image
```



axis image 命令通过将坐标区对象的 **DataAspectRatio** 属性设置为 [1 1 1] 实现效果。有关如何控制坐标区对象外观的详细信息，请参阅 **axis** 和 **axes**。

有时您希望在显示图像中，数据矩阵中的每个元素都与屏幕上的单个像素相对应。要显示这种矩阵元素与屏幕像素一一对应的图像，请使用 **imshow**。例如，以下命令显示的地球图像中，一个数据元素对应一个屏幕像素：

```
imshow(X,map)
```



图像对象及其属性

本节内容

- “图像 CData” (第 15-18 页)
- “图像 CDataMapping” (第 15-18 页)
- “XData 和 YData” (第 15-19 页)
- “在图像数据上添加文本” (第 15-20 页)
- “快速更新图像的其他技术” (第 15-21 页)

图像 CData

注意 `image` 和 `imagesc` 命令可用于创建图像对象。图像对象是坐标区对象的子级，线条、补片、曲面和文本对象也是如此。和所有图形对象一样，您可以设置图像对象的许多属性，从而微调该对象在屏幕上的外观。与图像对象的外观相关的最重要的属性有 **CData**、**CDataMapping**、**XData** 和 **YData**。这些属性将会在这一部分以及下面的部分中加以讨论。有关图像对象的这些以及所有属性的详细信息，请参阅 [image](#)。

图像对象的 **CData** 属性包含了数据数组。在以下命令中，**h** 是由 `image` 创建的图像对象的句柄，而且矩阵 **X** 和 **Y** 是相同的：

```
h = image(X); colormap(map)
Y = get(h,'CData');
```

CData 数组的维数控制图像是使用颜色图颜色还是以 RGB 图像的形式显示。如果 **CData** 数组是二维的，则图像是索引图像或强度图像；在这两种情况下，图像都使用颜色图颜色进行显示。如果另一方面，**CData** 数组是 $m \times n \times 3$ ，则该数组显示为真彩色图像，而忽略颜色图颜色。

图像 CDataMapping

CDataMapping 属性可用于控制图像是 **indexed** 还是 **intensity**。要显示索引图像，请将 **CDataMapping** 属性设置为 '**direct**'，从而使 **CData** 数组的值直接用作图窗颜色图的索引。当 `image` 命令与单个输入参数配合使用时，该命令将 **CDataMapping** 的值设置为 '**direct**'：

```
h = image(X); colormap(map)
get(h,'CDataMapping')
ans =
```

```
direct
```

通过将 **CDataMapping** 属性设置为 '**scaled**' 来显示强度图像。在这种情况下，**CData** 值经过线性缩放来形成颜色图索引。坐标区的 **CLim** 属性可用于控制缩放因子。`imagesc` 函数可创建一个 **CDataMapping** 属性设置为 '**scaled**' 的图像对象，而且该函数可调整父坐标区的 **CLim** 属性。例如：

```
h = imagesc(I,[0 1]); colormap(map)
get(h,'CDataMapping')
ans =
```

```
scaled
```

```
get(gca,'CLim')
```

```
ans =  
[0 1]
```

XData 和 YData

XData 和 YData 属性可用于控制图像的坐标系。对于 $m \times n$ 的图像而言，默认的 XData 是 [1 n]，而默认的 YData 是 [1 m]。这些设置表明以下内容：

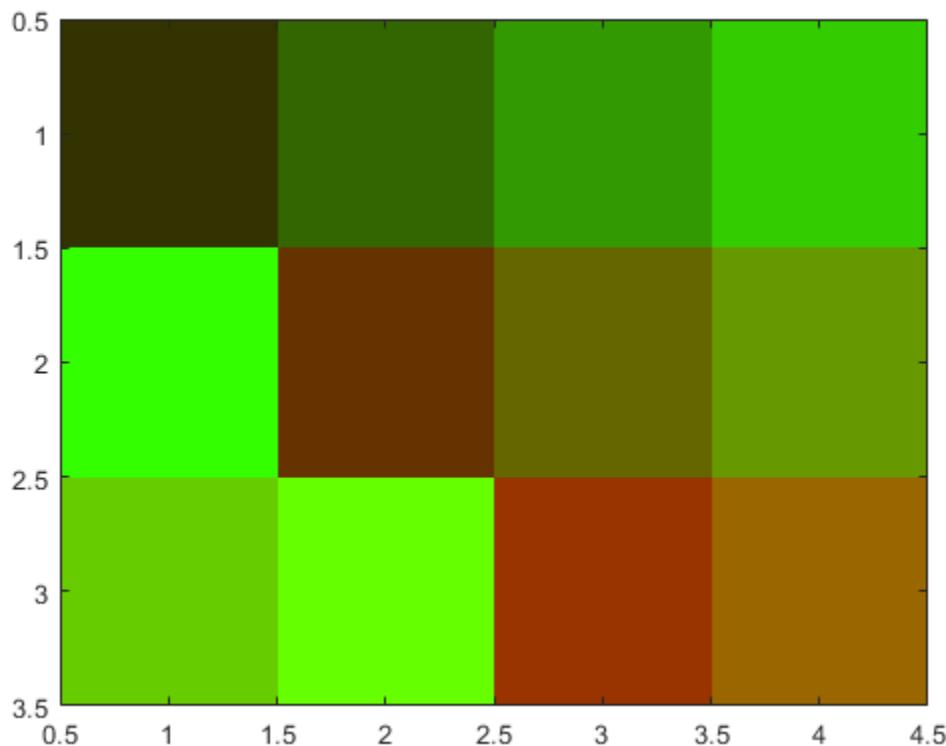
- 图像左列的 x 坐标为 1。
- 图像右列的 x 坐标为 n。
- 图形顶行的 y 坐标为 1。
- 图像底行的 y 坐标为 m。

图像的坐标系

使用默认坐标系

使用默认坐标系显示图像。使用 colrcube 图中的颜色。

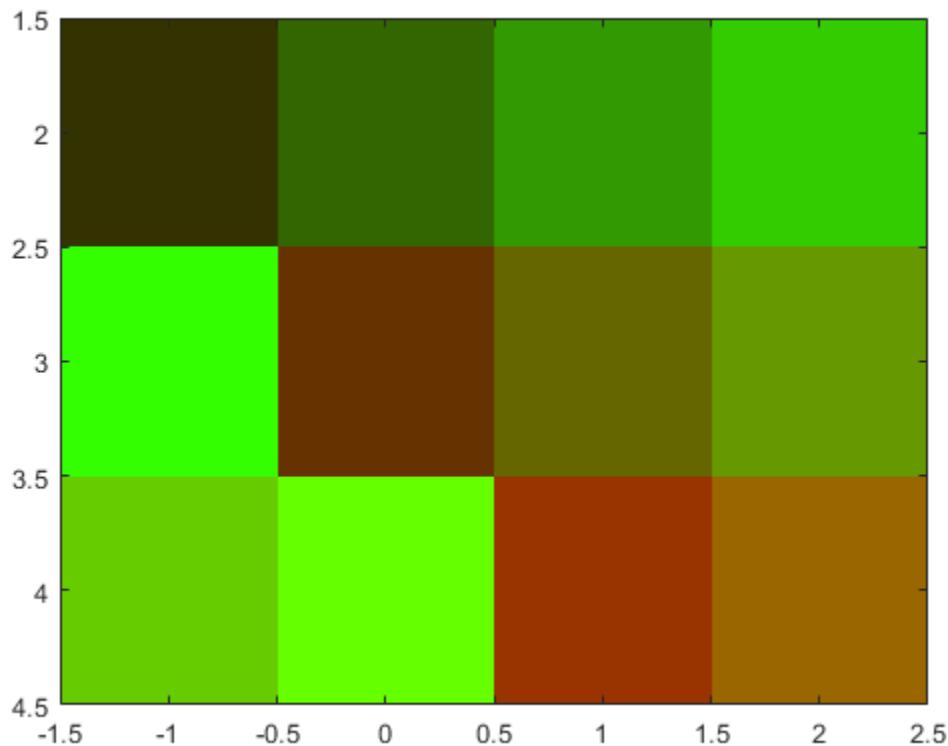
```
C = [1 2 3 4; 5 6 7 8; 9 10 11 12];  
im = image(C);  
colormap(colrcube)
```



指定坐标系

显示图像并指定坐标系。使用 `colorcube` 图中的颜色。

```
C = [1 2 3 4; 5 6 7 8; 9 10 11 12];
x = [-1 2];
y = [2 4];
figure
image(x,y,C)
colormap(colorcube)
```



在图像数据上添加文本

此示例说明如何使用数组索引将文本光栅化到现有图像。

使用 `text` 函数在坐标区中绘制文本。随后，使用 `getframe` 命令从屏幕上捕获文本并关闭图窗。

```
fig = figure;
t = text(.05,.1,'Mandrill Face','FontSize',20,'FontWeight','bold');
F = getframe(gca,[10 10 200 200]);
close(fig)
```

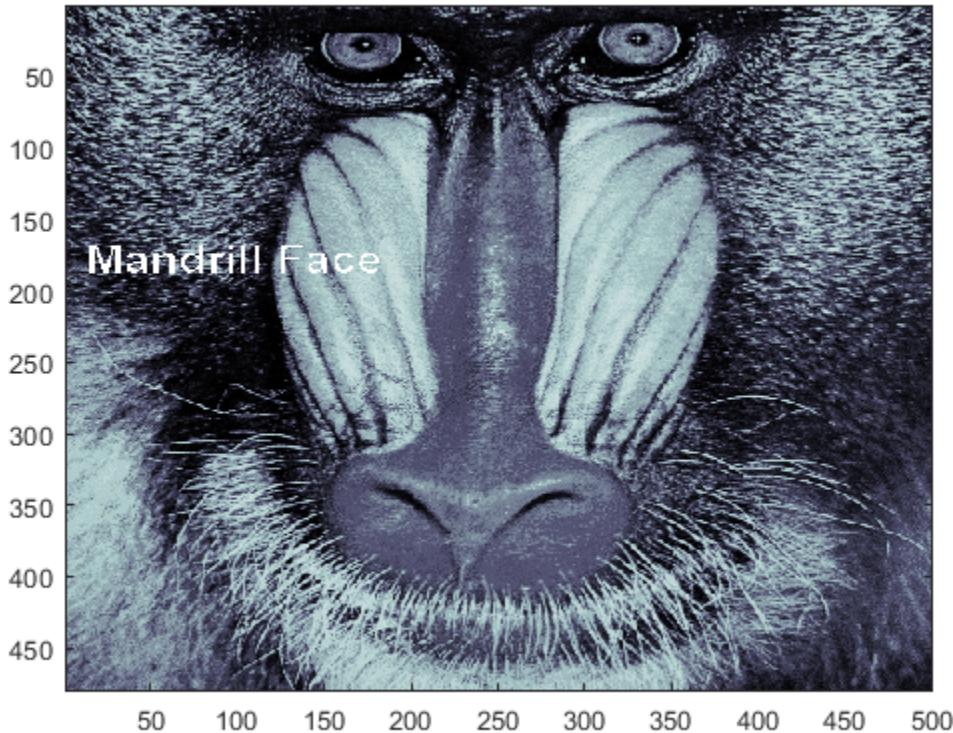
选择由 `getframe` 返回所得的 RGB 图像的任意平面。找到黑色像素（黑色是 0）并使用 `sub2ind` 将它们的下标转换为索引。使用这些下标将文本“刷”到 `mandrill` MAT 文件所包含的图像中。利用该图像的大小，加上文本的行位置和列位置来确定新图像的位置。对新图像建立索引，取代像素。

```
c = F.cdata(:,:,1);
[i,j] = find(c==0);
```

```
load mandrill
ind = sub2ind(size(X),i,j);
X(ind) = uint8(255);
```

使用 bone 颜色图显示新图像。

```
imagesc(X)
colormap bone
```



快速更新图像的其他技术

要提高图像对象的 **CData** 属性的更新速度，则优化 **CData** 并设置一些相关的图窗和坐标区属性：

- 尽可能使用最小的数据类型。对图像使用 **uint8** 数据类型比使用 **double** 数据类型更快。

设置图像 **CData** 属性过程包括复制矩阵用于图像。矩阵的总体大小依赖于单个元素的大小。使用更小的单个元素（即更小的数据类型）将会减小矩阵大小，并减少复制矩阵所需的时间。

- 使用最小可接受的矩阵。

如果图像显示速度处于最高优先级，那么您可能需要牺牲图像的大小和质量。再次强调，减小尺寸会减少复制矩阵所需的时间。

- 将坐标区的范围模式属性 (**XLimMode** 和 **YLimMode**) 设置为 **manual**。

如果它们设置为 **auto**，则每当有对象（例如图像、线条、补片等）改变了自身的某些数据时，坐标区都必须重新计算相关的属性。例如，如果指定

```
image(firstimage);
set(gca, 'xlimmode','manual',...
'ylimmode','manual',...
'zlimmode','manual',...
'climmode','manual',...
'alimmode','manual');
```

重绘图像前，坐标区并不重新计算任何一个范围值。

- 如果您的主要任务只是在屏幕上显示一系列图像，则考虑使用 **movie** 对象。

MATLAB **movie** 对象直接利用基本系统图形资源，而不会执行 MATLAB 对象代码。这比重复设置图像的 **CData** 属性更快，如上所述。

打印图像

如果将坐标区的 **Position** 设置为 [0 0 1 1] 以使其充满整个图窗，则打印时不会保留纵横比，因为 MATLAB 打印软件会在打印时根据图窗的 **PaperPosition** 属性调整图窗大小。要在打印时保留图像的纵横比，请通过命令行将图窗的 **PaperPositionMode** 设置为 'auto'。

```
set(gcf,'PaperPositionMode','auto')
print
```

如果 **PaperPositionMode** 设置为 'auto'，则所打印的图窗的宽度和高度由屏幕上图窗的维度决定，而且图窗位置会调整，以使得图窗在页面上居中。如果您想要 **PaperPositionMode** 的默认值为 'auto'，则在您的 **startup.m** 文件中输入此行内容。

```
set(groot,'defaultFigurePaperPositionMode','auto')
```

转换图像图形或数据类型

在数据类型之间转换会改变图像数据的解释。如果您将生成的数据组正确地解释为图像数据，则需要在转换时重新调整或偏移数据。（有关偏移的详细信息，请参阅先前的部分“图像类型”（第 15-4 页）和“索引图像”（第 15-7 页）。）

对于某些操作，将图像转换为不同图像类型会很有帮助。例如，要过滤以索引图像形式存储彩色图像，首先要将其转换为 RGB 格式。为了有效做到这一点，请使用 **ind2rgb** 函数。如果对该 RGB 图像应用滤波器，也会对图像中的强度值进行过滤，因为这同样适用。如果您试图对索引图像应用滤波器，则会将滤波器应用到索引图像矩阵的索引上，而结果可能毫无意义。

您可以使用 MATLAB 语法执行某些转换。例如，要将灰度图像转换为 RGB，可将原始矩阵的三个副本沿着第三维进行串联：

```
RGB = cat(3,I,I,I);
```

生成的 RGB 图像在红、绿、蓝平面具有完全相同的矩阵，因此图像以不同深浅的灰色显示。

也许为了与其他的软件产品兼容而改变图像的图形格式，这也非常简单。例如，要将图像从 BMP 转换为 PNG，则使用 **imread** 加载 BMP，将数据类型设置为 **uint8**、**uint16** 或 **double**，然后使用 **imwrite** 保存图像，且将 '**PNG**' 指定为目标格式。有关不同图形格式支持哪些位深的详细信息，以及将图像写入文件时如何指定格式类型，请参阅 **imread** 和 **imwrite**。

显示图像数据

此示例说明如何将 RGB 图像读取到工作区并进行显示。然后，示例将 RGB 图像转换为灰度图像并进行显示。最后，示例说明如何将多个单独的图像组合成一个分块图（即蒙太奇）。

读取图像

示例文件 `peppers.png` 包含一个 RGB 图像。使用 `imread` 函数将图像读取到工作区中。

```
RGB = imread('peppers.png');
```

显示颜色图像

使用 `imshow` 函数显示图像数据。

```
imshow(RGB)
```



转换为灰度

使用 `rgb2gray` 函数将 RGB 图像转换为灰度。

```
gray = rgb2gray(RGB);
```

显示灰度图像

使用 `imshow` 函数显示灰度图像。

```
imshow(gray)
```



使用多个图像创建一个分块图

将多个单独的图像组合成一个分块图，并使用 `imshow` 函数显示分块图。

```
out = imtile({'peppers.png', 'ngc6543a.jpg'});
imshow(out);
```



打印和保存

- “从“文件”菜单打印图窗”（第 16-2 页）
- “通过“编辑”菜单将图窗复制到剪贴板”（第 16-5 页）
- “保存前自定义图窗”（第 16-8 页）
- “将绘图保存为图像或向量图形文件”（第 16-14 页）
- “使用特定大小、分辨率或背景色保存图窗”（第 16-18 页）
- “保存图窗以供以后在 MATLAB 中重新打开”（第 16-22 页）
- “保存和复制绘图时保留最少的空白”（第 16-24 页）

从“文件”菜单打印图窗

本节内容

- “直接打印输出”（第 16-2 页）
- “保留背景色和刻度值”（第 16-2 页）
- “图窗大小和布局”（第 16-2 页）
- “线宽和字体大小”（第 16-3 页）

直接打印输出

要打印图窗，请使用**文件** > **打印**。例如，创建一个条形图以进行打印。

```
x = [3 5 2 6 1 8 2 3];
bar(x)
```

点击**文件** > **打印**，选择一个打印机，并点击**确定**。该打印机必须已设置在您的系统上。如果未看到已设置的打印机，请重新启动 MATLAB。

要以编程方式打印图窗，请使用**print** 函数。

保留背景色和刻度值

所打印图窗的某些详细信息的外观可能不同于图窗在屏幕上的显示。默认情况下，打印的图窗使用白色图窗背景色。此外，如果打印的图窗大小不同于原始图窗大小，则坐标轴范围和刻度值可能不同。

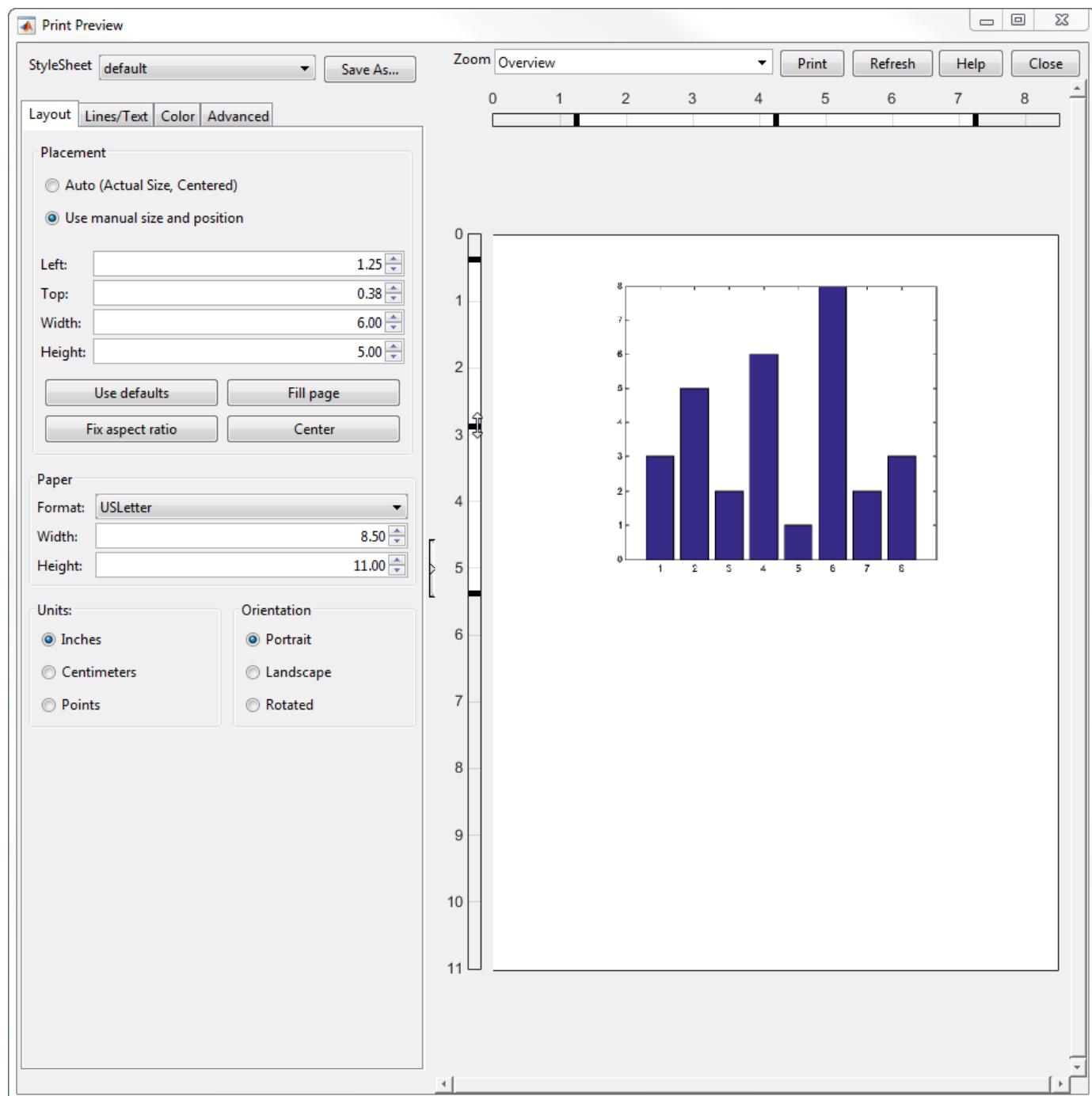
- 通过点击**文件** > **打印预览** > **颜色**选项卡保留图窗背景色。对背景色，选择**与图窗相同**。对色阶，选择**颜色**。
- 通过点击**文件** > **打印预览** > **高级**选项卡保留坐标轴范围和刻度值位置。然后，对于**坐标轴范围和刻度**选项，选择**保持屏幕范围和刻度**。

要以编程方式保留颜色方案，请将图窗的 **InvertHardcopy** 属性设置为 '**off**'。要保持相同的坐标轴范围和刻度线，请将坐标区的 **XTickMode**、**YTickMode** 和 **ZTickMode** 属性设置为 '**manual**'。

图窗大小和布局

要打印具有特定尺寸的图窗，请点击**文件** > **打印预览** > **布局**选项卡。然后，对于**布局**选项，选择**手动确定大小及位置**。在文本框中指定您所需的尺寸。或者，使用图窗预览左侧和顶部的滑块调整大小和布局。

MATLAB 可更改打印预览中的图窗大小，但不会更改实际图窗大小。

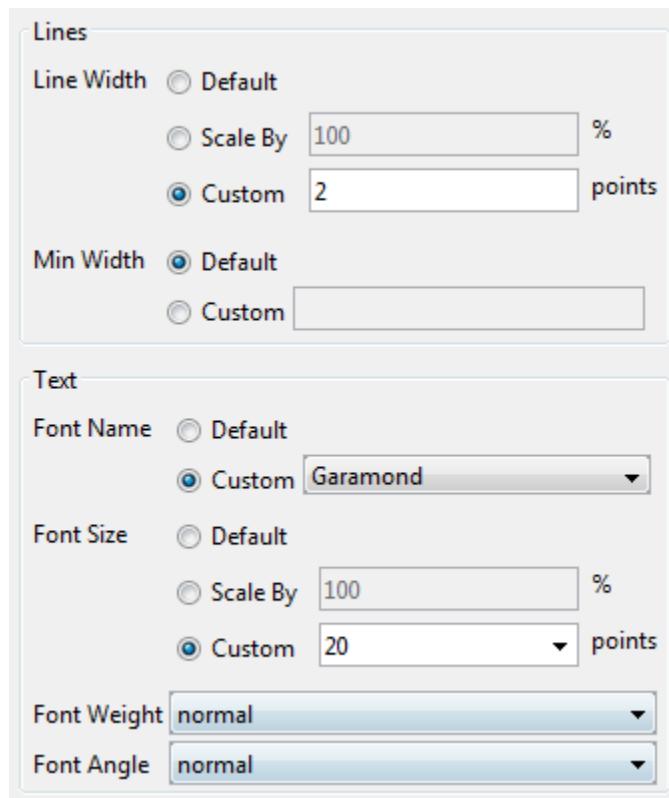


要以编程方式指定打印的图窗大小和布局，请使用图窗的 **PaperPosition** 属性。

线宽和字体大小

要更改打印输出的线宽、字体大小和字体名称，请点击**文件** > **打印预览** > **直线/文本**选项卡。在适当的文本框中指定自定义线宽，例如 2 磅。从字体下拉列表中选择字体名称并指定自定义字体大小。例如，使用 20 磅 Garamond 字体。

MATLAB 可更改打印预览中的线宽和字体，但不会更改实际图窗的外观。



要以编程方式更改线宽和字体大小，请设置图形对象的属性。有关列表，请参阅“图形对象属性”。

另请参阅

[print](#) | [saveas](#)

相关示例

- “通过“编辑”菜单将图窗复制到剪贴板”（第 16-5 页）

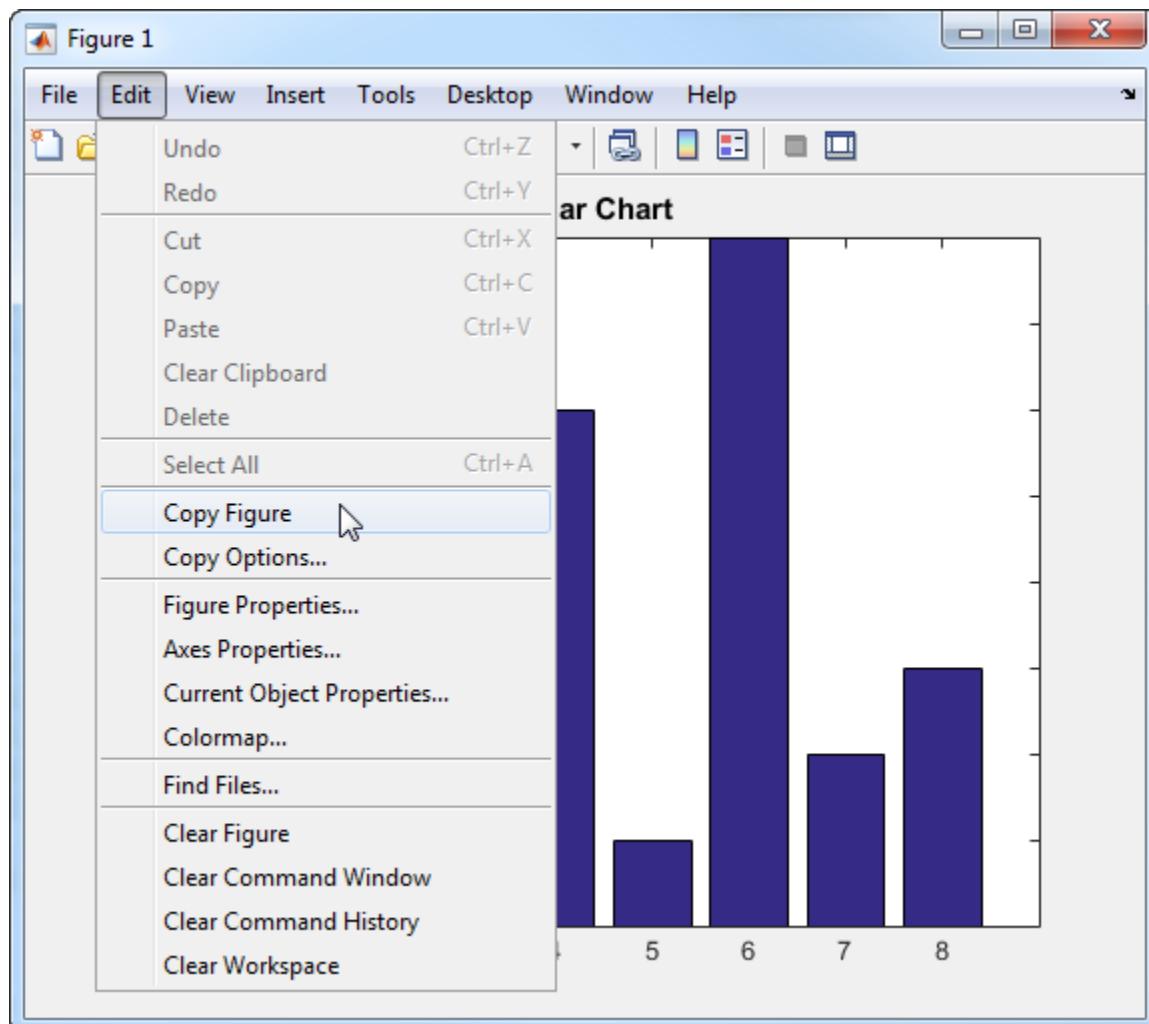
通过“编辑”菜单将图窗复制到剪贴板

以下示例演示如何将图窗复制到剪贴板以及如何设置复制选项。当图窗在剪贴板中时，您可以将其粘贴到其他应用程序（如文档或演示文稿）中。

将图窗复制到剪贴板

创建一个具有标题的条形图。通过点击编辑 > 复制图窗将图窗复制到系统剪贴板。

```
x = [3 5 2 6 1 8 2 3];
bar(x)
title('Bar Chart')
```



将复制的图窗粘贴到其他应用程序中（通常通过右键点击方式）。默认情况下，MATLAB 将所复制图窗的背景色转换为白色。

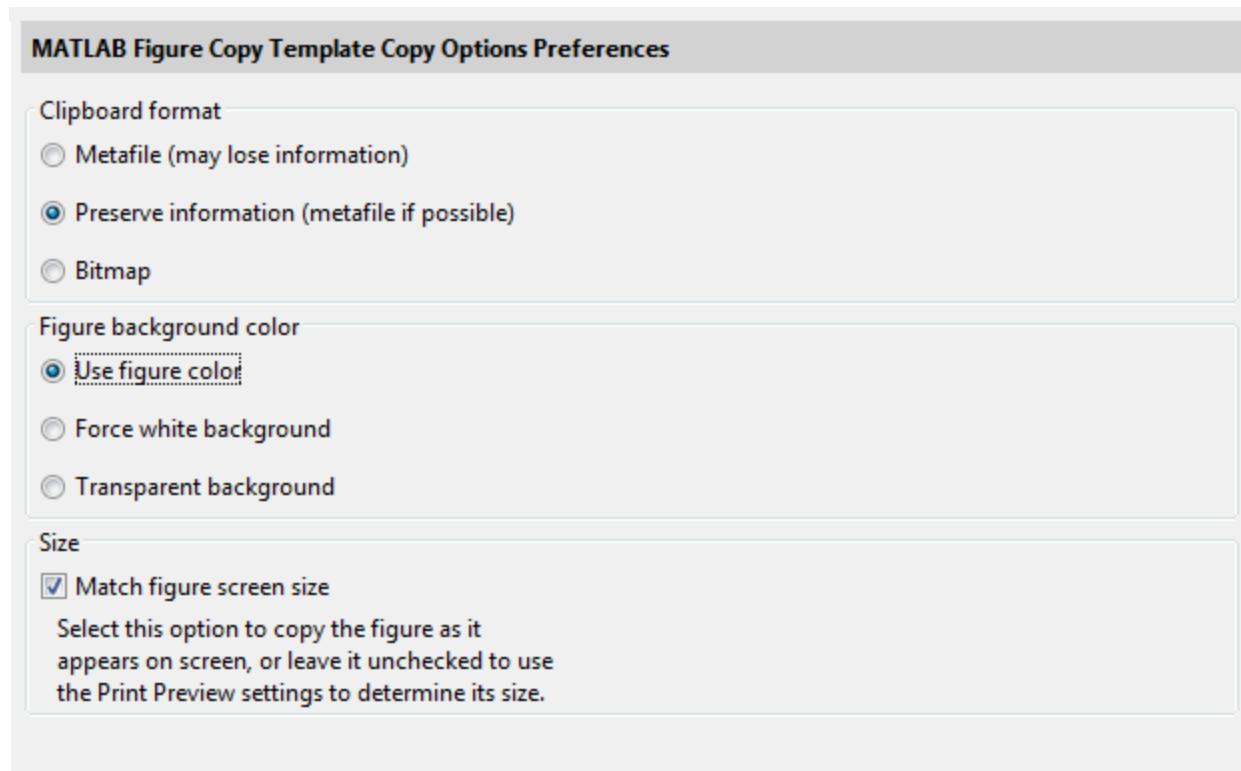
注意 复制图窗选项在 Linux® 系统上不可用。可使用编程方式作为替代。

要以编程方式复制图窗，请结合使用 '`-clipboard`' 选项和 `print`。将格式指定为 '`-dbitmap`'、'`-dpdf`' 或 '`-dmeta`'。仅 Windows 系统支持图元文件格式 '`-dmeta`'。

指定格式、背景色和大小选项

您可以调整复制到剪贴板的图窗的某些设置。在图窗菜单中选择**编辑 > 复制选项**以访问这些选项。这些设置适用于以后复制到剪贴板的所有图窗。它们不会影响图窗在屏幕上的显示方式。

注意 该窗口仅在 Windows 系统上可用。在 Mac 和 Linux 系统上，可使用编程方式替代。



将剪贴板格式设置为以下选项之一：

- 图元文件 - 以 EMF 颜色向量格式复制图窗。
- 保留信息 - 根据图窗的渲染器选择格式。如果渲染器为 Painters，则格式为图元文件。如果渲染器为 OpenGL，则格式为位图图像。
- 位图 - 以位图格式复制图窗。

将图窗背景色设置为以下选项之一：

- 使用图窗颜色 - 保留与屏幕上显示相同的背景色。要使用编程方式替代，请在复制前将图窗的 `InvertHardcopy` 属性设置为 '`'off'`'。
- 强制使用白色背景 - 使用白色背景复制图窗。要使用编程方式替代，请在复制前将图窗的 `InvertHardcopy` 属性设置为 '`'on'`'。

- 透明背景 - 使用透明背景复制图窗。要使用编程方式替代，请在复制前将图窗的 `Color` 属性设置为 '`none`'，将 `InvertHardcopy` 属性设置为 '`off`'。图元文件和 PDF 格式支持透明度。位图格式不支持透明度。

选择与图窗在屏幕上的大小相匹配，以使用与屏幕上显示相同的大小复制图窗。清除此选项可使用“导出设置”对话框中指定的宽度和高度。

另请参阅

`saveas` | `print`

相关示例

- “将绘图保存为图像或向量图形文件” (第 16-14 页)

保存前自定义图窗

以下示例说明如何在保存图窗前使用“导出设置”窗口自定义图窗。它说明如何更改图窗大小、背景色、字体大小和线宽。它还说明如何将设置另存为导出样式，以便能在保存其他图窗前应用于它们。

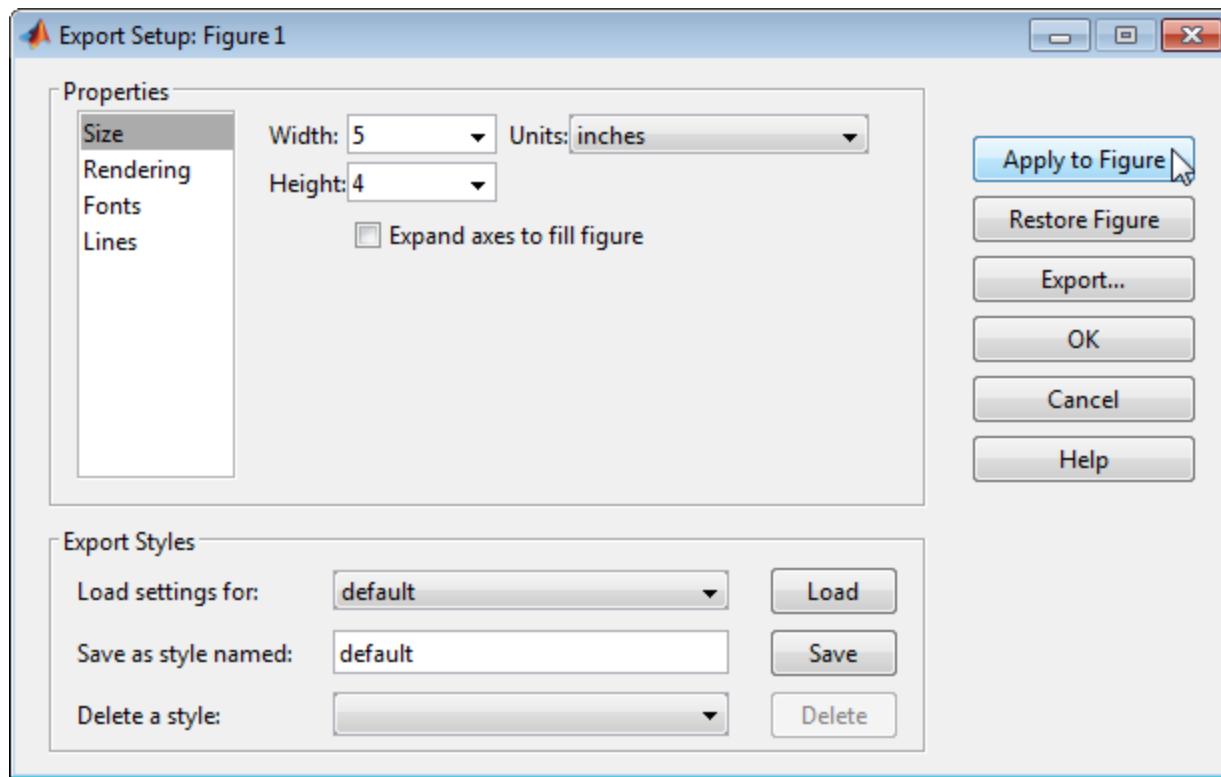
设置图窗大小

创建一个线图。

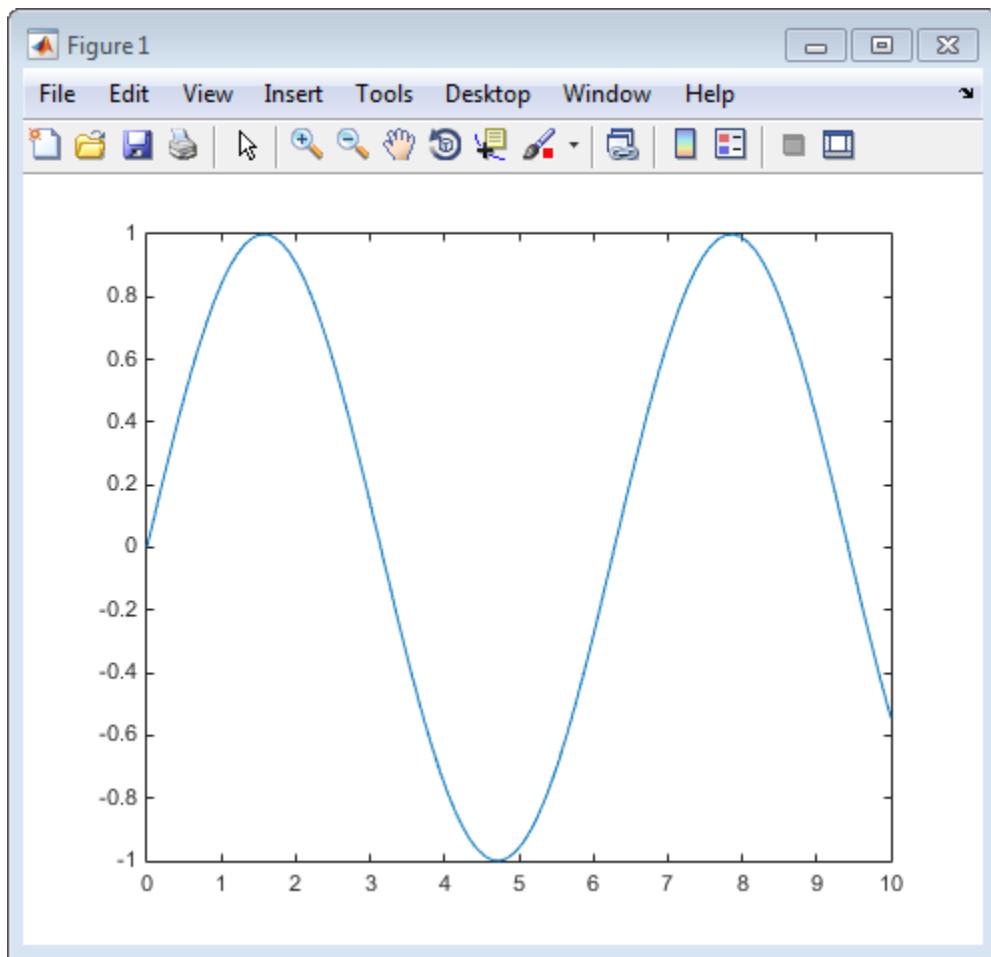
```
x = linspace(0,10);
y = sin(x);
plot(x,y)
```

通过点击文件 > 导出设置设置图窗大小。在宽度和高度字段中指定所需的尺寸，例如 5×4 英寸。尺寸包括除框架、标题栏、菜单栏和任何工具栏外的整个图窗窗口。如果指定的宽度和高度太大，则图窗可能无法达到指定的大小。

要使坐标区填充图窗，请选择放大坐标区至充满图窗。此选项仅影响 PositionConstraint 属性设置为 'outerposition' 的坐标区。

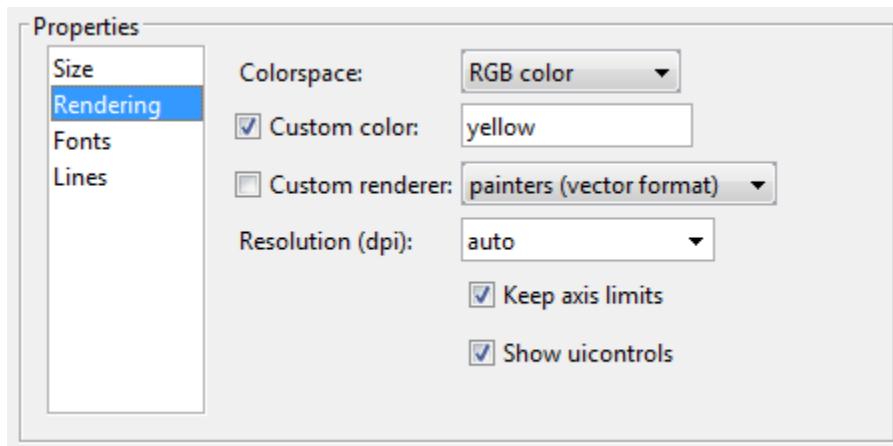


点击应用于图窗。应用这些设置会更改图窗在屏幕上的外观。“导出设置”对话框中的所有设置都将应用于图窗。因此，可更改的项不仅仅是图窗大小。例如，默认情况下，MATLAB 将所保存图窗的背景色转换为白色。



设置图窗背景色

通过点击“导出设置”窗口中的**渲染**属性设置图窗背景色。在“自定义颜色”字段中，从表中指定一种颜色名称或以RGB三元组指定颜色。例如，将背景色设置为黄色。

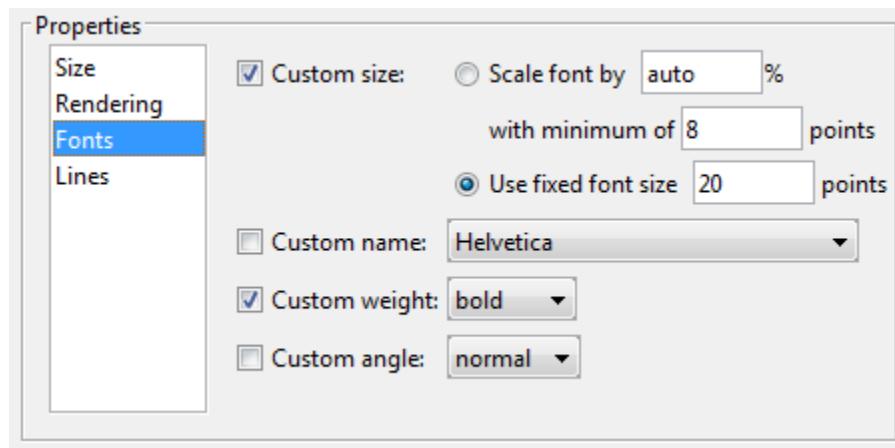


RGB 三元组是包含三个元素的行向量，其元素分别指定颜色中红、绿、蓝分量的强度。强度值必须位于 [0,1] 范围内，例如 [0.4 0.6 0.7]。下表列出了一些常见 RGB 三元组及其对应的颜色名称。要指定默认的灰色背景色，请将“自定义颜色”字段设置为 default。

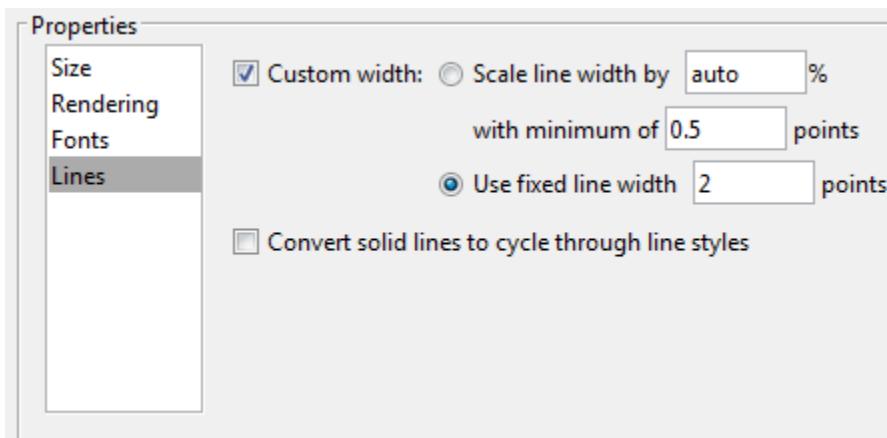
长名称	短名称	对应的 RGB 三元组
white	w	[1 1 1]
yellow	y	[1 1 0]
magenta	m	[1 0 1]
red	r	[1 0 0]
cyan	c	[0 1 1]
green	g	[0 1 0]
blue	b	[0 0 1]
black	k	[0 0 0]

设置图窗字体大小和线宽

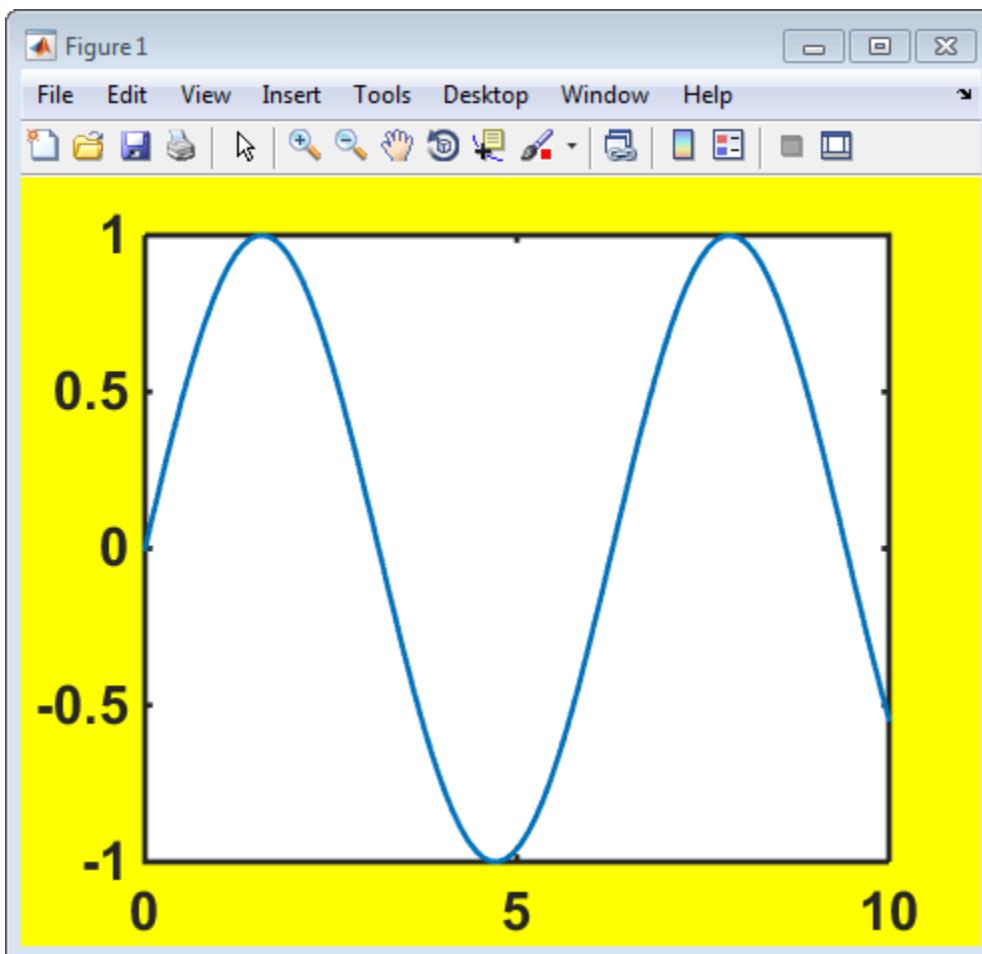
通过点击字体属性更改字体。指定固定字体大小，然后选择字体名称、字体粗细和字体角度。例如，使用 20 磅加粗字体。刻度线位置可能会更改以适应新字体大小。



通过点击线条属性更改线宽。指定固定线宽，例如 2 磅。

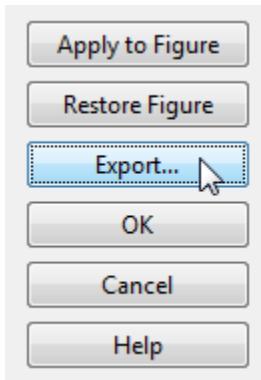


点击“导出设置”对话框右侧的**应用于图窗**。



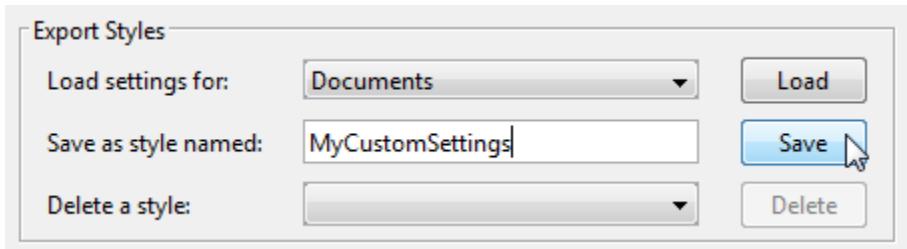
将图窗保存到文件

通过首先点击**导出**，然后指定文件名称、位置和所需的格式，将图窗保存到文件。有关文件格式的详细信息，请参阅 `saveas`。



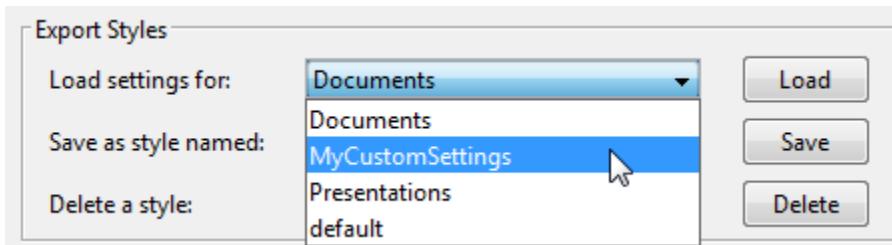
保存图窗设置以供将来使用

通过创建导出样式，保存您的设置以用于以后的图窗。在“导出样式”部分中，键入样式名称，例如 MyCustomSettings。然后，点击“保存”。



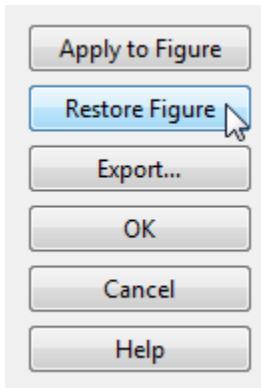
将设置应用于另一个图窗

通过从其图窗菜单打开“导出设置”框，将您的设置应用于另一个图窗。在“导出样式”部分中，选择样式名称并点击加载。接下来，点击“导出设置”对话框右侧的**应用于图窗**。MATLAB会将保存的样式设置应用于图窗。



将图窗还原为原始设置

通过点击**还原图窗**将屏幕上的图窗还原为原始设置。



以编程方式自定义图窗

您也可以采用编程方式自定义图窗。要以编程方式自定义图窗，请设置图形对象的属性。通常，图形函数会返回可用来访问和修改图形对象的输出参数。例如，将 `plot` 函数返回的图形线条对象赋给变量，并设置这些对象的 `LineWidth` 属性。

```
p = plot(rand(5));
set(p,'LineWidth',3)
```

如果不返回图形对象作为输出参数，可以使用 `findobj` 查找具有特定属性的对象。例如，在当前图窗中查找 `Type` 属性设置为 `'line'` 的所有对象。然后，设置其 `LineWidth` 属性。

```
plot(rand(5))
p = findobj(gcf,'Type','line')
set(p,'LineWidth',3);
```

有关所有图形对象及其属性的列表，请参阅“图形对象属性”。

另请参阅

`saveas` | `print` | `属性检查器`

相关示例

- “将绘图保存为图像或向量图形文件”（第 16-14 页）
- “使用特定大小、分辨率或背景色保存图窗”（第 16-18 页）

将绘图保存为图像或向量图形文件

您可以使用坐标区工具栏中的导出按钮 ，或通过调用 **exportgraphics** 函数，将绘图另存为图像，或另存为向量图形文件。在决定要使用的内容类型时，应考虑要将文件放入其中的文档的质量、文件大小和格式要求。

大多数应用程序都支持图像。它们适用于表示绘画图像和复杂的曲面。但是，由于图像由像素组成，因此当您在其他具有不同分辨率的设备上打印或显示它们时，它们不一定能够很好地缩放。在某些情况下，您可能需要以足够的分辨率保存图像，以满足某些质量要求。文件的分辨率越高，大小就越大，这会使它们难以通过电子邮件共享或上传到服务器。而且很难在不引入人为处理痕迹的情况下编辑图像中的线和文本。

向量图形文件包含绘制线、曲线和多边形的说明。它们适用于表示由线、曲线和纯色区域组成的内容。这些文件包含可缩放到任意大小的高质量内容。但是，某些曲面和网格图过于复杂，无法使用向量图形来表示。某些应用程序支持对向量图形文件进行广泛的编辑，但其他应用程序仅支持调整图形大小。

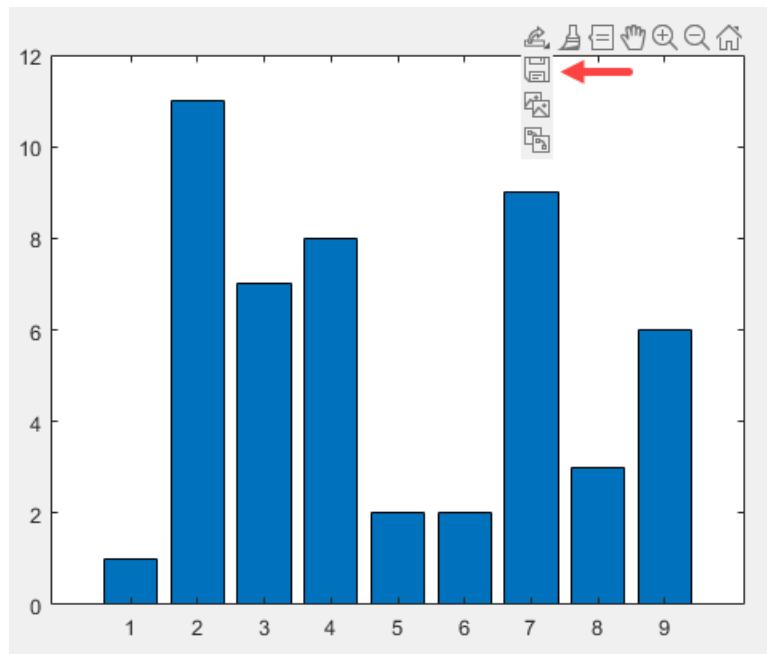
无论将绘图另存为图像还是向量图形文件，在保存文件之前，都可以通过在 MATLAB 图窗中最终确定您的内容，来获得最佳结果。

以交互方式保存绘图

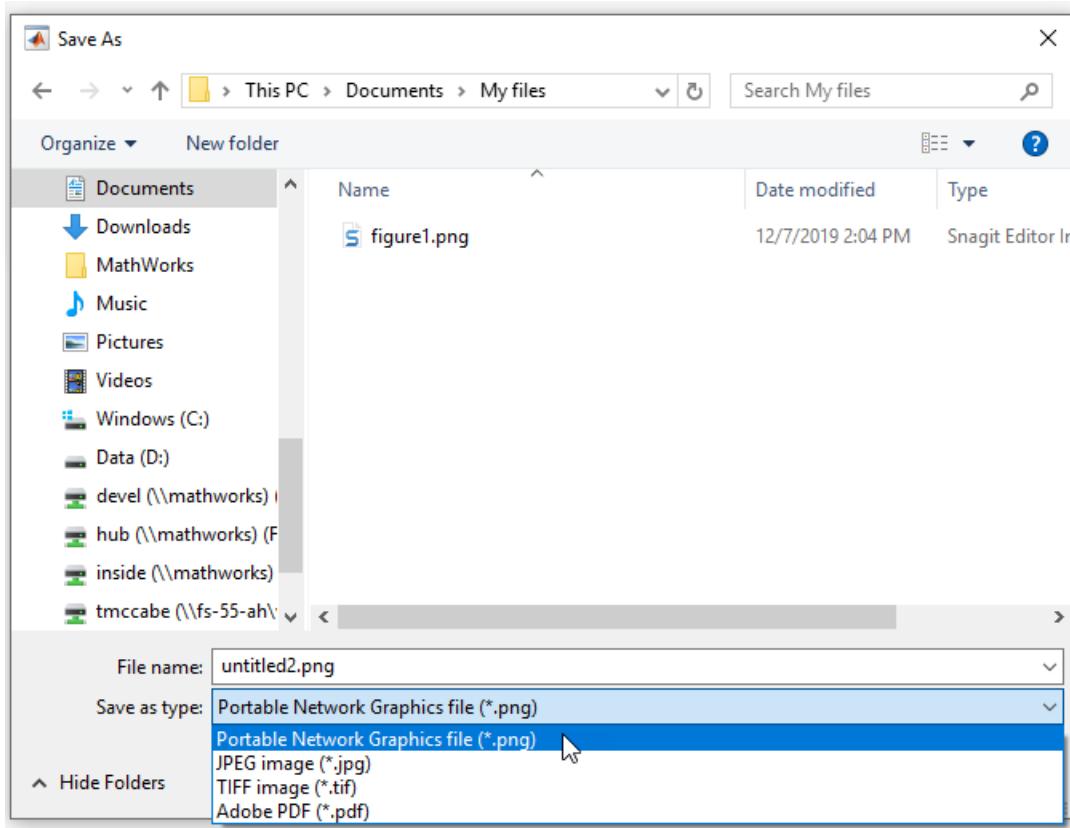
要使用交互式控件保存绘图，请使用坐标区工具栏中的导出按钮 。将鼠标指针悬停在坐标区右上角时会显示该工具栏。该导出按钮支持三种图像格式（PNG、JPEG 和 TIFF）以及 PDF 文件，PDF 文件中可以包含图像或向量图形，具体取决于坐标区中的内容。

例如，创建一个条形图。通过将鼠标悬停在坐标区工具栏中的导出按钮  上，然后在下拉列表中选择第一项，将图保存到文件中。

```
bar([1 11 7 8 2 2 9 3 6])
```



MATLAB 显示包含文件类型选项的“另存为”对话框。



当使用导出按钮保存绘图时，输出将围绕坐标区内容（包括任何图例或颜色栏）精确裁剪。输出不包括坐标区以外的内容，例如图窗中的其他坐标区。

如果图窗以分块图布局方式包含多个绘图，您可以通过将工具栏移到布局上将所有绘图保存在一起。要移动工具栏，请调用 `axtoolbar` 函数并将 `TiledChartLayout` 对象指定为输入参数。然后将鼠标悬停在工具栏中的导出按钮上。将鼠标悬停在布局的右上角时，将会显示工具栏。

以编程方式保存绘图

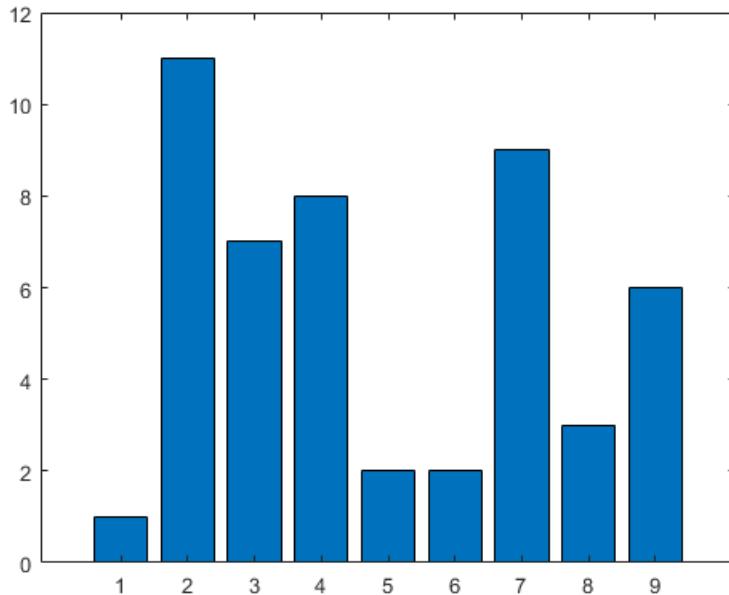
注意 以下示例使用了 `exportgraphics` 函数，该函数从 R2020a 开始提供。如果您使用的是较早的版本，请参阅将绘图保存为图像或向量图形文件 (19b)。

要以编程方式保存绘图，请使用 `exportgraphics` 函数，该函数是 R2020a 中的新函数。保存的内容会围绕坐标区精确裁剪，尽可能减少留白。所有 UI 组件和相邻的容器（例如面板）都不会包括在保存的内容中。`exportgraphics` 函数支持三种图像格式（PNG、JPEG 和 TIFF）和三种同时支持向量和图像内容的格式（PDF、EPS 和 EMF）。PDF 格式支持嵌入字体。

例如，创建一个条形图并获取当前图窗。然后将该图窗另存为 PNG 文件。在本例中，指定每英寸 300 点 (DPI) 的输出分辨率。

```
bar([1 11 7 8 2 2 9 3 6])
f = gcf;
```

```
% Requires R2020a or later
exportgraphics(f,'barchart.png','Resolution',300)
```



如果您指定了扩展名为 .pdf、.eps 或 .emf 的文件名，MATLAB 将根据图窗中的内容存储图像或向量图形。

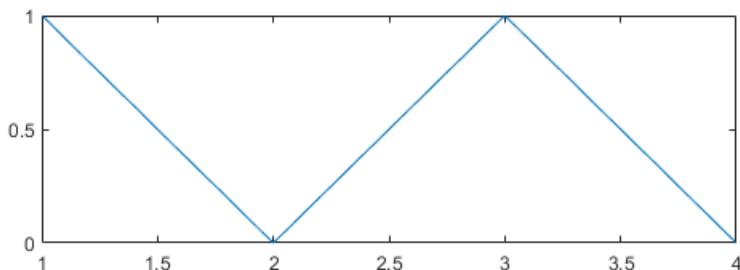
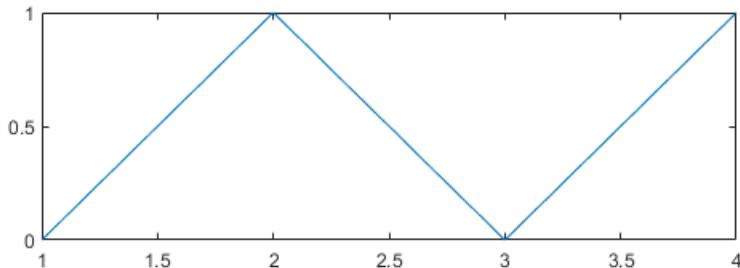
您可以通过指定 'ContentType' 名称-值对组参数，来控制文件中包含图像还是向量图形。例如，将当前图窗中的内容另存为包含向量图形的 PDF。

```
% Requires R2020a or later
exportgraphics(gcf,'vectorfig.pdf','ContentType','vector')
```

要保存图窗中的多个绘图，请创建一个分块图布局，并将 TileChartLayout 对象传递给 **exportgraphics** 函数。例如，创建一个 2×1 分块图布局 **t**。通过调用 **nexttile** 函数在布局中放置两个坐标区，并在坐标区内绘图。然后，使用 **t** 作为第一个参数调用 **exportgraphics** 函数，将两个绘图另存为一个 EPS 文件。

```
t = tiledlayout(2,1);
nexttile
plot([0 1 0 1])
nexttile
plot([1 0 1 0])

% Requires R2020a or later
exportgraphics(t,'twoplots.eps')
```



在其他应用程序中打开保存的绘图

可以在其他应用程序（例如 Microsoft Word 或 LaTeX）中打开您保存的文件。

要将绘图添加到 LaTeX 文档，请先使用 `exportgraphics` 函数将绘图另存为 EPS 文件。然后将 `\includegraphics` 元素添加到 LaTeX 文档中。例如：

```
\documentclass{article}
\usepackage{graphicx}
\begin{document}

\begin{figure}[h]
\centerline{\includegraphics[height=10cm]{twoplots.eps}}
\caption{Plots from MATLAB}
\end{figure}

\end{document}
```

另请参阅

[nexttile](#) | [tiledlayout](#) | [exportgraphics](#) | [copygraphics](#)

相关示例

- “使用特定大小、分辨率或背景色保存图窗”（第 16-18 页）
- “保存和复制绘图时保留最少的空白”（第 16-24 页）
- “保存图窗以供以后在 MATLAB 中重新打开”（第 16-22 页）

使用特定大小、分辨率或背景色保存图窗

本节内容

- “指定分辨率”（第 16-18 页）
- “指定大小”（第 16-19 页）
- “指定背景色”（第 16-20 页）
- “保留坐标轴范围和刻度值”（第 16-20 页）

从 R2020a 开始提供。替换以特定大小和分辨率保存图窗 (R2019b) 和保存图窗时保留背景色 (R2019b)。

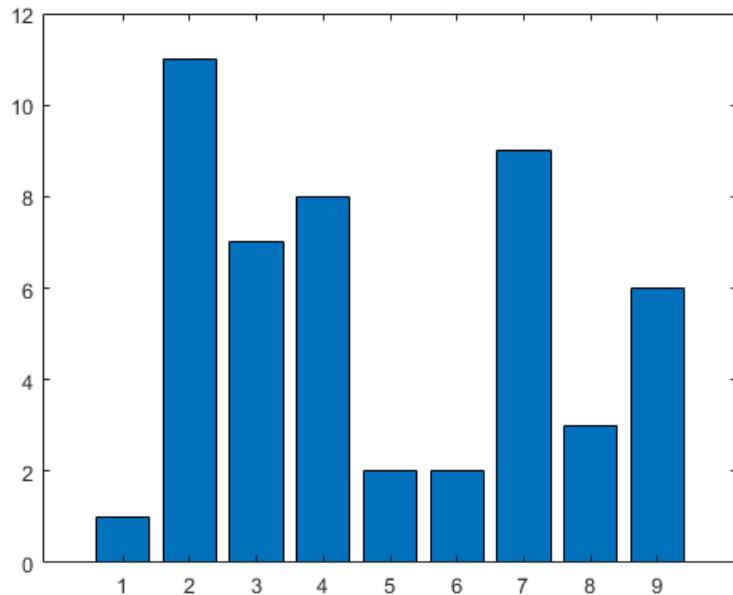
要保存绘图以包含在文档（例如出版物或幻灯片演示文稿）中，请使用 **exportgraphics** 函数。此函数使您能够以适合您文档的大小、分辨率和背景色保存绘图。保存的内容会围绕坐标区精确裁剪，尽可能减少留白。所有 UI 组件和相邻的容器（例如面板）都不会包括在保存的内容中。

指定分辨率

要以特定的分辨率将图窗另存为图像，请调用 **exportgraphics** 函数，并指定 '**Resolution**' 名称-值对组参数。默认情况下，图像以每英寸 150 点 (DPI) 的分辨率保存。

例如，创建一个条形图并获取当前图窗。然后将图窗另存为 300-DPI PNG 文件。

```
bar([1 11 7 8 2 2 9 3 6])
f = gcf;
exportgraphics(f,'barchart.png','Resolution',300)
```



或者，您也可以指定坐标区而不是图窗作为 **exportgraphics** 函数的第一个参数。

```
ax = gca;
exportgraphics(ax,'barchartaxes.png','Resolution',300)
```

指定大小

`exportgraphics` 函数会以您屏幕上显示的相同宽度和高度捕获内容。如果想要更改宽度和高度，请调整图窗中所显示内容的大小。一种方法是使用分块图布局方式以所需大小（没有任何填充）创建绘图。然后将该布局传递给 `exportgraphics` 函数。

例如，要将一个条形图另存为 3×3 英寸的方形图像，请先创建一个 1×1 的分块图布局 `t`，然后将 '`Padding`' 名称-值对组参数设置为 '`tight`'。

```
t = tiledlayout(1,1,'Padding','tight');
```

在 R2021a 之前，需要将 '`Padding`' 设置为 '`none`'。

将 `t` 的 `Units` 属性设置为 `inches`。然后将 `t` 的 `OuterPosition` 属性设置为 `[0.25 0.25 3 3]`。向量中的前两个数字将布局定位在距离图窗左侧和底部边缘 0.25 英寸的位置。后两个数字将布局的宽度和高度设置为 3 英寸。

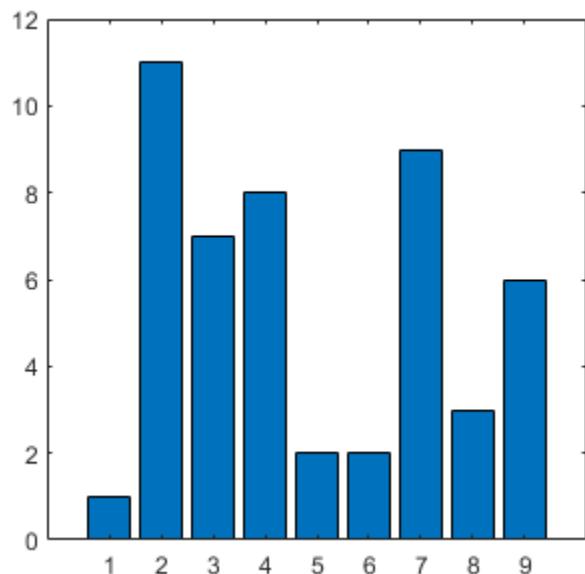
```
t.Units = 'inches';
t.OuterPosition = [0.25 0.25 3 3];
```

下一步，通过调用 `nexttile` 函数创建一个坐标区对象。然后在坐标区中创建一个条形图。

```
nexttile;
bar([1 11 7 8 2 2 9 3 6])
```

通过将 `t` 传递给 `exportgraphics` 函数，将布局另存为一个 300-DPI 的 JPEG 文件。生成的图像是一个约 3 英寸的方形。

```
exportgraphics(t,'bar3x3.jpg','Resolution',300)
```



另一种更改大小的方法是将内容另存为向量图形文件。然后，您可以在文档中调整内容大小。要将内容另存为向量图形文件，请调用 `exportgraphics` 函数，并将 '`ContentType`' 名称-值对组参数设置为 '`vector`'。例如，创建一个条形图，然后将图窗另存为一个包含向量图形的 PDF 文件。所有可嵌入字体均包含在 PDF 中。

```
bar([1 11 7 8 2 2 9 3 6])
f = gcf;
exportgraphics(f,'barscalable.pdf','ContentType','vector')
```

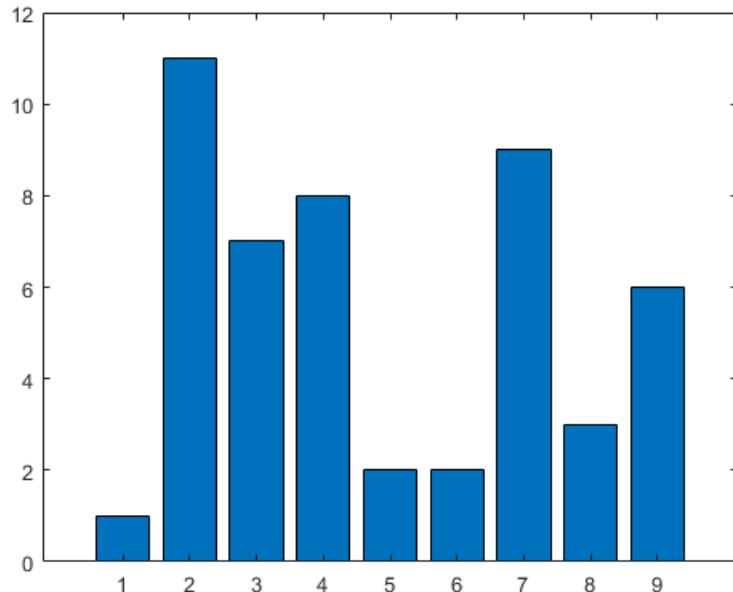
指定背景色

默认情况下，`exportgraphics` 函数使用白色背景保存内容。您可以通过设置 `BackgroundColor` 名称值对组参数来指定其他背景。以下是可能的值：

- '`current`' - 使用坐标区父容器（例如图窗或面板）的颜色。
- '`none`' - 将背景色设为透明或白色，具体取决于文件格式和 `ContentType` 的值：
 - 透明 - 适用于 `ContentType='vector'` 的文件
 - 白色 - 适用于图像文件，或当 `ContentType='image'` 时
- 自定义颜色，指定为 RGB 三元组（如 [1 0 0]）、十六进制颜色代码（如 #FF0000）或指定颜色（例如 '`red`'）。

例如，创建一个条形图，使用透明背景将图窗另存为一个 PDF 文件。

```
bar([1 11 7 8 2 2 9 3 6])
f = gcf;
exportgraphics(f,'bartransparent.pdf','ContentType','vector',...
    'BackgroundColor','none')
```

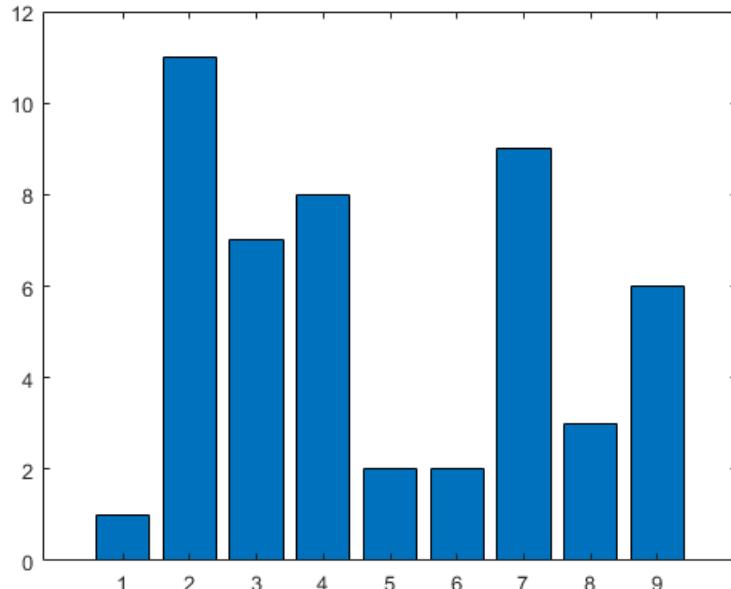


保留坐标轴范围和刻度值

有时，`exportgraphics` 函数使用不同的坐标轴范围或刻度值来保存您的内容，具体取决于字体大小和文件的分辨率。要防止更改坐标轴范围和刻度值，请将坐标区的刻度值模式和范围模式属性设置为 '`manual`'。例如，在笛卡尔坐标区内绘图时，设置 x 轴、y 轴和 z 轴的刻度值和范围模式属性。

```
bar([1 10 7 8 2 2 9 3 6])
ax = gca;
ax.XTickMode = 'manual';
```

```
ax.YTickMode = 'manual';
ax.ZTickMode = 'manual';
ax.XLimMode = 'manual';
ax.YLimMode = 'manual';
ax.ZLimMode = 'manual';
exportgraphics(ax,'barticks.png')
```



对于极坐标图，将极坐标区的 RTickMode、ThetaTickMode、RLimMode 和 ThetaLimMode 属性设置为 'manual'。

另请参阅

函数

[exportgraphics](#) | [copygraphics](#) | [tiledlayout](#) | [nexttile](#)

属性

[TiledChartLayout Properties](#) | [Axes](#) | [PolarAxes](#)

详细信息

- “将绘图保存为图像或向量图形文件” (第 16-14 页)
- “保存和复制绘图时保留最少的空白” (第 16-24 页)

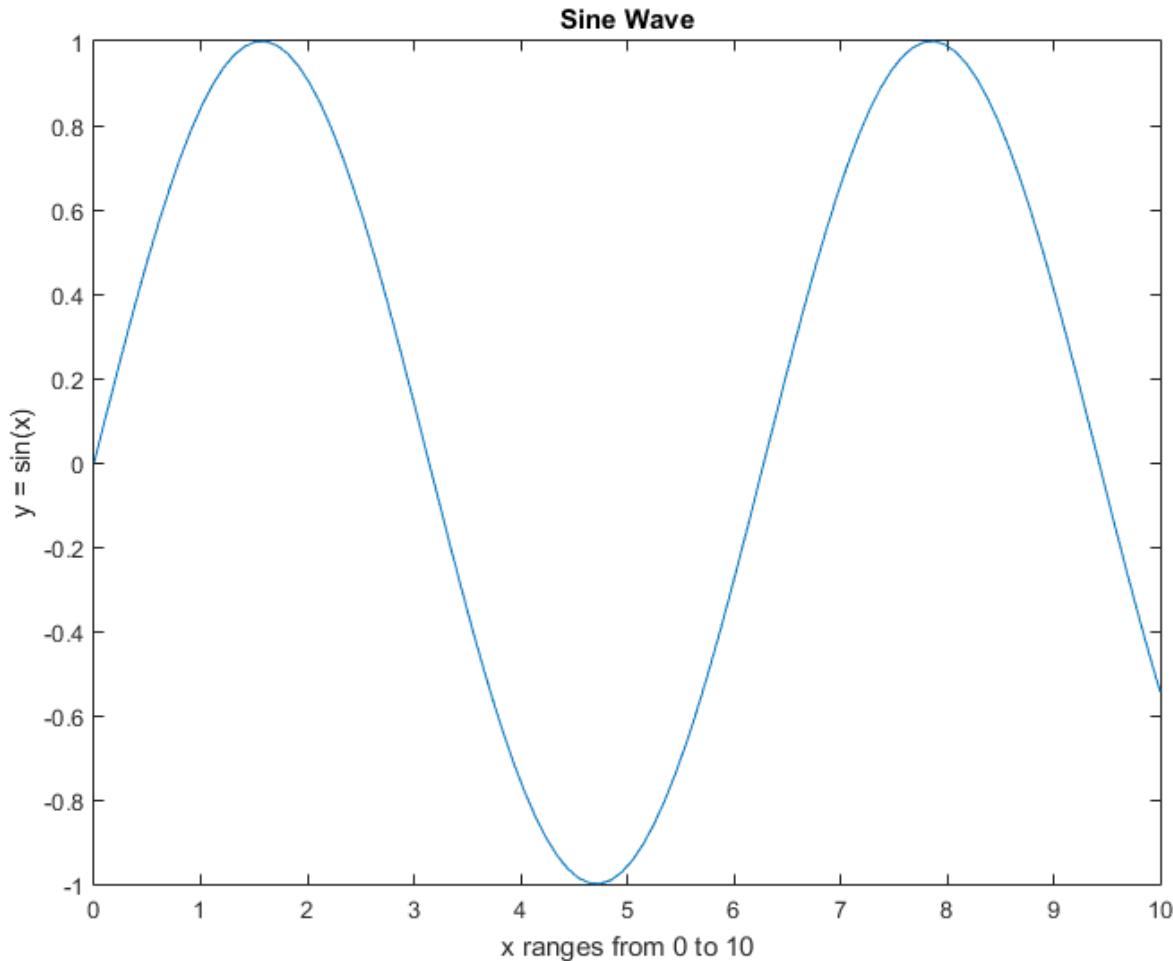
保存图窗以供以后在 MATLAB 中重新打开

以下示例说明如何保存图窗以便以后在 MATLAB 中将其重新打开。您可以将图窗保存为 FIG 文件，也可以生成并保存代码。

将图窗保存为 FIG 文件

创建要保存的绘图。添加标题和轴标签。

```
x = linspace(0,10);
y = sin(x);
plot(x,y)
title('Sine Wave')
xlabel('x ranges from 0 to 10')
ylabel('y = sin(x)')
```



使用 `savefig` 函数将图窗保存为 FIG 文件。FIG 文件存储重新创建图窗所需的信息。

```
savefig('SineWave.fig')
```

关闭图窗，然后使用 **openfig** 函数重新打开保存的图窗。

```
close(gcf)  
openfig('SineWave.fig')
```

openfig 使用与原始对象相同的数据创建新图窗、新坐标区以及新线条对象。新对象的大多数属性值都与原始对象一样。但是，任何当前默认值都会应用于新图窗。您可以与图窗进行交互。例如，您可以平移、缩放和旋转坐标区。

注意 FIG 文件仅可在 MATLAB 中打开。如果您想以可在其他应用程序中打开的格式保存图窗，请参阅“将绘图保存为图像或向量图形文件”（第 16-14 页）。

生成代码以重新创建图窗

或者，生成绘图的 MATLAB 代码，然后使用代码重新生成图形。生成代码时会捕获您使用绘图工具所做的修改。

点击**文件** > **生成代码...**。生成的代码显示在 MATLAB 编辑器中。通过点击**文件** > **另存为**保存代码。

生成的文件不会存储重新创建图形所需的数据，因此您必须提供数据参数。数据参数不必与原始数据相同。文件开头的注释声明了所期望的数据类型。

另请参阅

[saveas](#) | [savefig](#) | [openfig](#)

相关示例

- “将绘图保存为图像或向量图形文件”（第 16-14 页）

保存和复制绘图时保留最少的空白

在保存或复制绘图内容时，使空白最小化的方法之一是使用坐标区工具栏，将鼠标悬停在坐标区右上角时会出现该工具栏。另一种方法是使用 `exportgraphics` 和 `copygraphics` 函数，它们具有更大的灵活性。

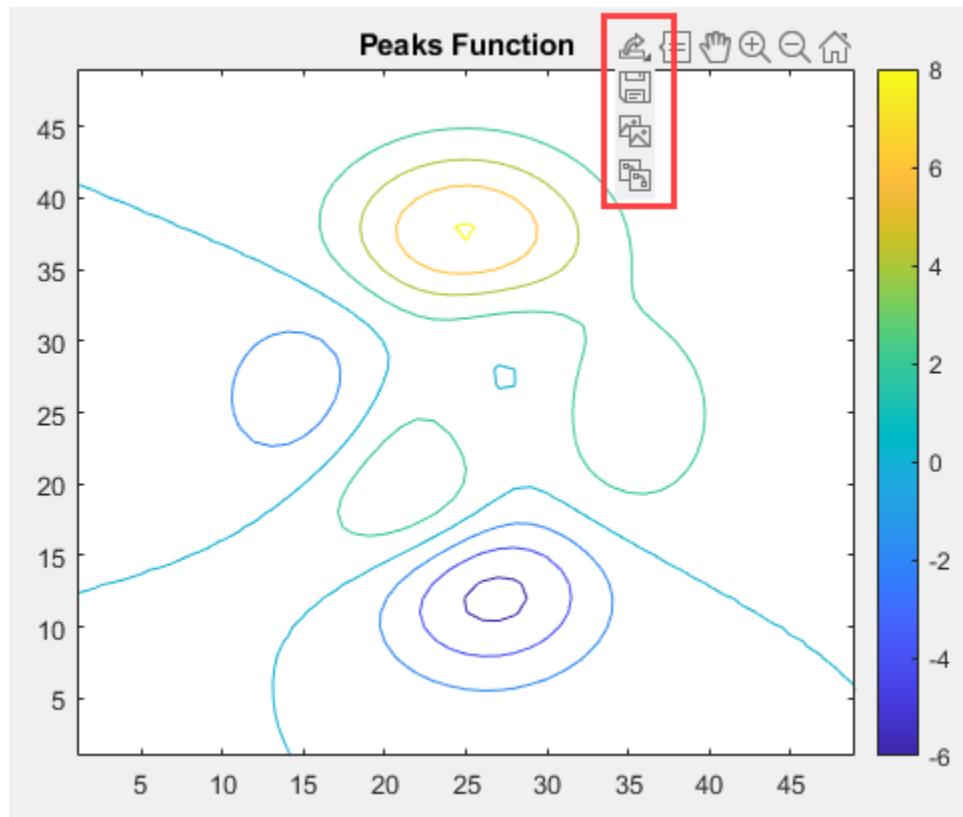
注意 以下示例使用了 R2020a 中新增的 `exportgraphics` 和 `copygraphics` 函数。如果您使用的是较早的版本，请参阅保存绘图时保留最少的空白 (19b)。

保存或复制单一绘图

使用标题和颜色栏创建 `peaks` 函数的等高线图。

```
contour(peaks)
colorbar
title('Peaks Function')
```

将鼠标悬停在坐标区工具栏中的导出按钮  上并选择下拉列表中的第一项，将绘图保存到文件。如果要将绘图的内容复制到剪贴板，请选择下拉列表中的第二项或第三项。第二项将内容复制为图像，第三项将内容复制为向量图形。保存或复制的内容将围绕标题、坐标区和颜色栏精确裁剪。



也可以使用 `exportgraphics` 函数保存内容，此函数从 R2020a 开始提供。此函数围绕您的内容提供相同的精确裁剪，还提供了其他选项。例如，您可以保存图像文件并指定分辨率。

```
ax = gca;
% Requires R2020a or later
exportgraphics(ax,'myplot.png','Resolution',300)
```

copygraphics 函数提供了将内容复制到剪贴板的类似功能。

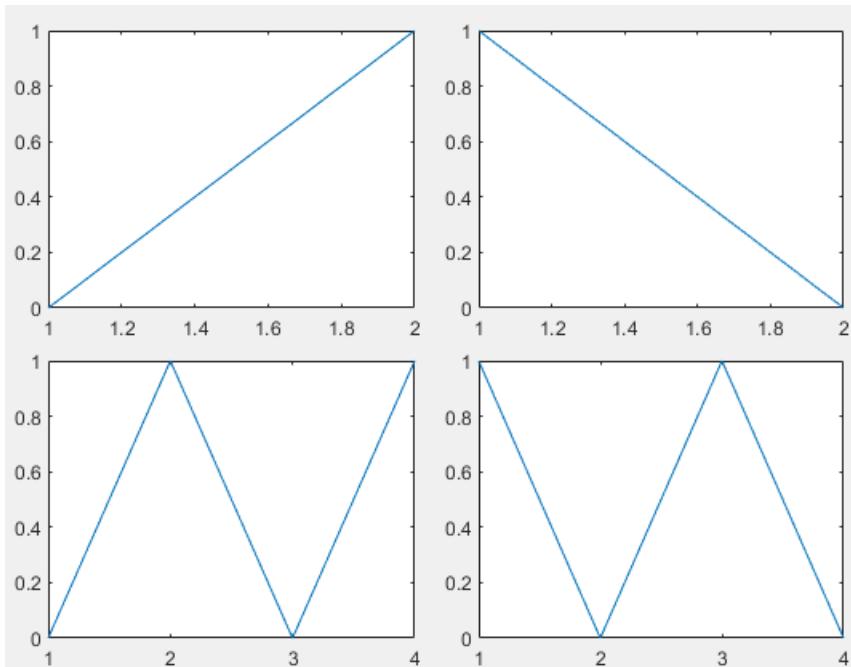
```
ax = gca;
% Requires R2020a or later
copygraphics(ax,'Resolution',300)
```

保存或复制图窗中的多个绘图

从 R2019b 开始，您可以使用 **tiledlayout** 函数在图窗中创建分块图。该函数具有用于最小化绘图周围空间的选项。（如果您使用的是较早的版本，您可以使用 **subplot** 函数来创建分块图。不过，**subplot** 函数没有用于控制绘图周围空间的选项。）

通过调用 **tiledlayout** 函数，创建一个 2×2 分块图布局。要最小化绘图之间的空间，请将 'TileSpacing' 名称-值对组参数设置为 'compact'。要使布局周围的空间最小化，请将 'Padding' 名称-值对组参数设置为 'compact'。下一步，调用 **nexttile** 函数创建第一个坐标区，并调用 **plot** 函数在坐标区中绘图。然后再创建三个坐标区和绘图。

```
% Requires R2019b or later
t = tiledlayout(2,2,'TileSpacing','Compact','Padding','Compact');
nexttile
plot([0 1])
nexttile
plot([1 0])
nexttile
plot([0 1 0 1])
nexttile
plot([1 0 1 0])
```



通过将分块图布局 (*t*) 传递给 `exportgraphics` 函数，将布局另存为 PDF 文件。在本例中，使用透明背景保存 PDF。

```
% Requires R2020a or later  
exportgraphics(t,'fourplots.pdf','BackgroundColor','none')
```

也可以使用 `copygraphics` 函数，将布局复制到剪贴板。

```
% Requires R2020a or later  
copygraphics(t,'BackgroundColor','none')
```

另请参阅

函数

`exportgraphics` | `copygraphics` | `tiledlayout` | `nexttile`

属性

`TiledChartLayout` Properties

详细信息

- “将绘图保存为图像或向量图形文件”（第 16-14 页）

图形属性

- “修改图形对象” (第 17-2 页)
- “图形对象层次结构” (第 17-9 页)
- “访问属性值” (第 17-12 页)
- “图形对象控制的功能” (第 17-17 页)
- “默认属性值” (第 17-21 页)
- “自动计算的属性的默认值” (第 17-24 页)
- “MATLAB 如何找到默认值” (第 17-26 页)
- “出厂定义属性值” (第 17-27 页)
- “多级默认值” (第 17-28 页)

修改图形对象

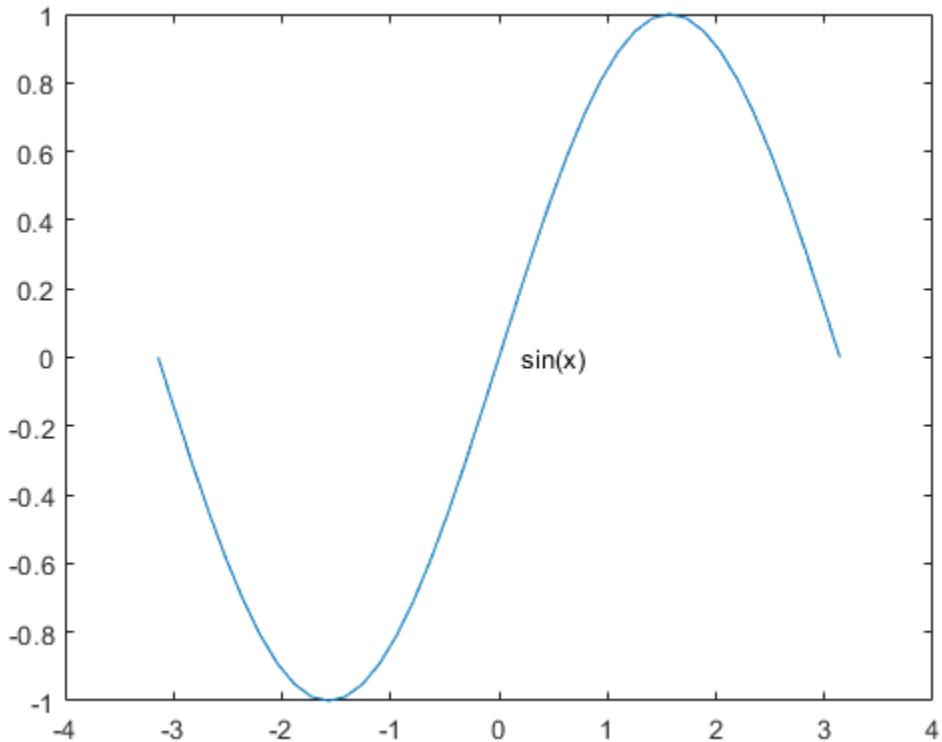
以下示例演示如何在 MATLAB® 中创建、显示和修改图形对象。

图形对象

当 MATLAB 创建一个绘图时，它会创建一系列的图形对象。图形对象的例子有图窗、坐标区、线条、填充和文本。下图有三个图形对象 -- 一个坐标区、一条线条和一个文本对象。使用可选输出参数存储所创建的图形对象。

```
x = -pi:pi/20:pi;
y = sin(x);

f = figure;
p = plot(x,y);
txt1 = text(0.2,0,'sin(x)');
```



所有图形对象均有您可以查看和修改的属性。这些属性具有默认值。下面所示的线条对象 `p` 显示了最常用的线条属性，如 `Color`、`LineStyle` 和 `LineWidth`。

```
p
p =
Line with properties:

    Color: [0 0.4470 0.7410]
    LineStyle: '-'
```

```

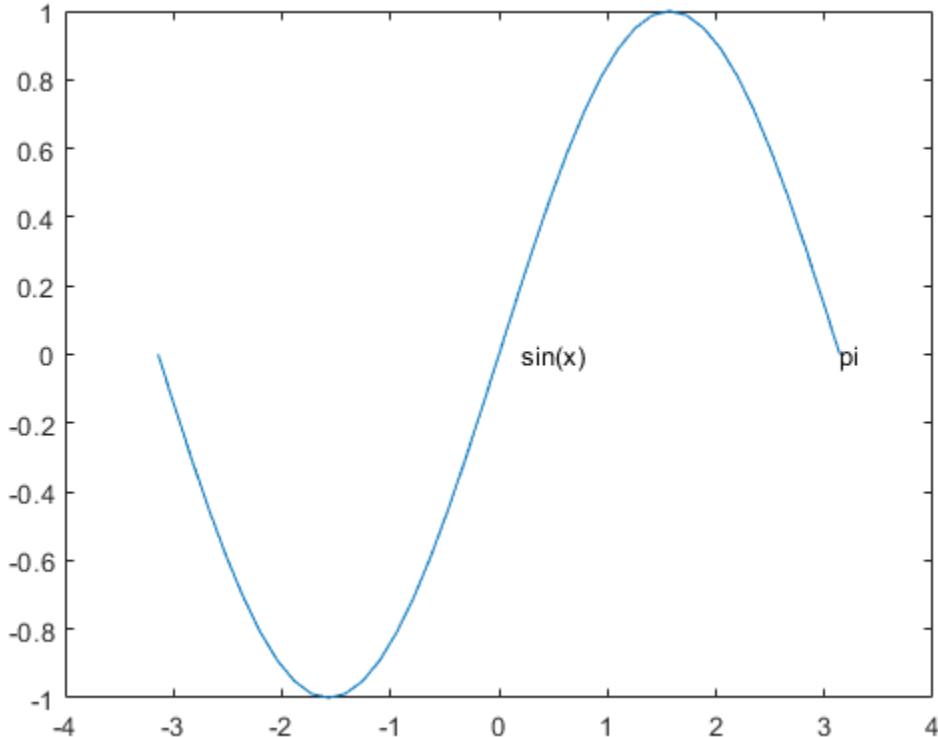
LineWidth: 0.5000
Marker: 'none'
MarkerSize: 6
MarkerFaceColor: 'none'
XData: [-3.1416 -2.9845 -2.8274 -2.6704 -2.5133 -2.3562 ... ]
YData: [-1.2246e-16 -0.1564 -0.3090 -0.4540 -0.5878 ... ]
ZData: [1x0 double]

```

Show all properties

如果用于创建对象的命令缺少了分号，MATLAB 会以相同的方式显示内容。

```
txt2 = text(x(end), y(end), 'pi')
```



```

txt2 =
Text (pi) with properties:

    String: 'pi'
    FontSize: 10
    FontWeight: 'normal'
    FontName: 'Helvetica'
    Color: [0 0 0]
    HorizontalAlignment: 'left'
    Position: [3.1416 1.2246e-16 0]
    Units: 'data'

```

[Show all properties](#)

获取图形对象属性

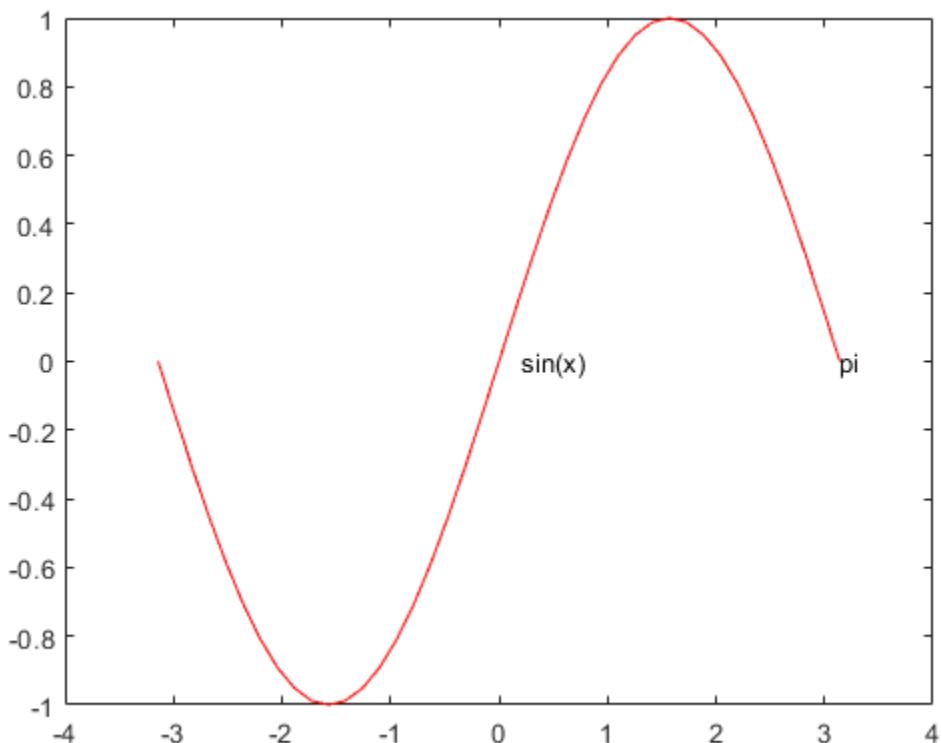
若要访问图形对象的个别属性，请使用圆点表示法语法 `object.PropertyName`。例如，返回线条对象的 `LineWidth` 属性。

```
pcol = p.LineWidth
```

```
pcol = 0.5000
```

通过设置线条的 `Color` 属性将其颜色更改为红色。

```
p.Color = 'red';
```



父级和子级

MATLAB 按一定的层次结构排列图形对象。层次结构的顶部是称为图形根的特殊对象。若要访问图形根，请使用 `groot` 函数。

```
groot
```

```
ans =
Graphics Root with properties:
```

```
    CurrentFigure: [1x1 Figure]
    ScreenPixelsPerInch: 100
```

```
ScreenSize: [1 1 1280 1024]
MonitorPositions: [1 1 1280 1024]
Units: 'pixels'
```

Show all properties

所有图形对象（除了根）均有一个父级。例如，坐标区的父级是一个图窗。

```
ax = gca;
ax.Parent

ans =
Figure (1) with properties:

    Number: 1
    Name: ''
    Color: [1 1 1]
    Position: [348 480 583 437]
    Units: 'pixels'
```

Show all properties

许多对象也有子级。此套坐标区有三个子级 - 两个文本对象和一个线条对象。

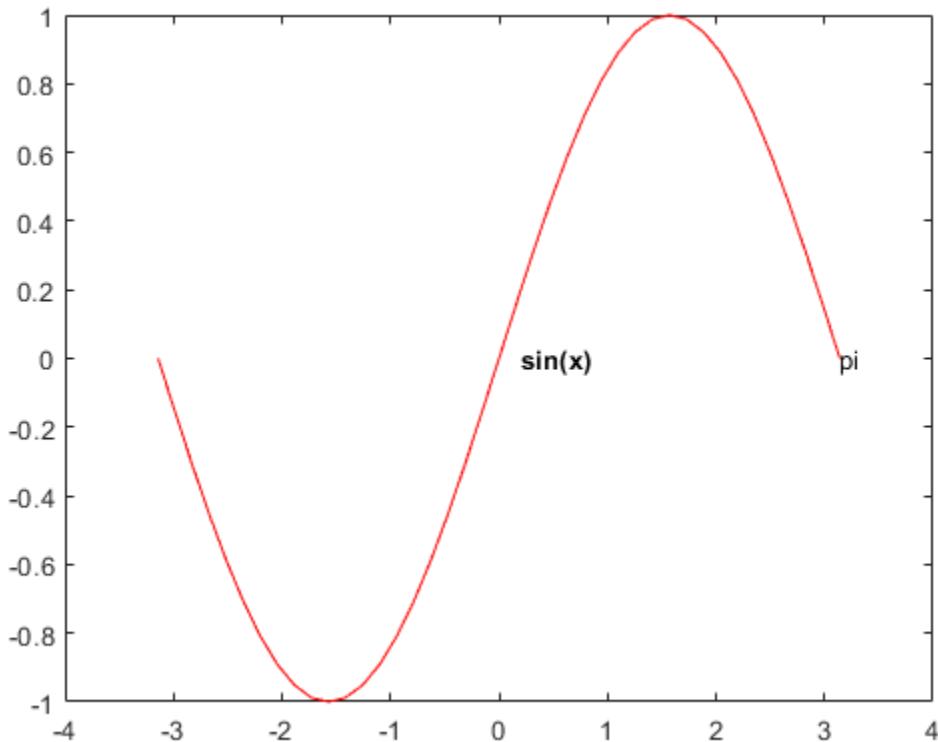
ax.Children

```
ans =
3x1 graphics array:
```

```
Text  (pi)
Text  (sin(x))
Line
```

因为坐标区有多个子级，**Children** 属性的值是一个图形对象数组。要访问坐标区的个别子级，请对数组建立索引。然后，您可以设置子级对象的属性。

```
t = ax.Children(2);
t.FontWeight = 'bold';
```



预分配图形对象数组

在 MATLAB 中有一个最佳做法，是在使用数组前先进行预分配。使用 **gobjects** 命令预分配图形对象数组。然后，您可以将图形对象添加到该数组中。

```
objarray = gobjects(1,5);
objarray(1) = f;
objarray(2) = ax;
objarray(3) = p;
objarray(4) = txt1;
objarray(5) = txt2;
objarray

objarray =
1x5 graphics array:
```

Figure Axes Line Text Text

获取所有对象属性

MATLAB 中的图形对象有许多属性。若要查看对象的所有属性，请使用 **get** 命令。

```
get(f)
```

```
Alphamap: [0 0.0159 0.0317 0.0476 0.0635 0.0794 0.0952 ... ]
BeingDeleted: off
BusyAction: 'queue'
```

```
ButtonDownFcn: "
    Children: [1x1 Axes]
    Clipping: on
CloseRequestFcn: 'closereq'
    Color: [1 1 1]
    Colormap: [256x3 double]
ContextMenu: [0x0 GraphicsPlaceholder]
CreateFcn: "
CurrentAxes: [1x1 Axes]
CurrentCharacter: "
CurrentObject: [0x0 GraphicsPlaceholder]
CurrentPoint: [0 0]
DeleteFcn: "
DockControls: on
FileName: "
GraphicsSmoothing: on
HandleVisibility: 'on'
Icon: "
InnerPosition: [348 480 583 437]
IntegerHandle: on
Interruptible: on
InvertHardcopy: on
KeyPressFcn: "
KeyReleaseFcn: "
MenuBar: 'none'
Name: "
NextPlot: 'add'
Number: 1
NumberTitle: on
OuterPosition: [348 480 583 437]
PaperOrientation: 'portrait'
PaperPosition: [1.3350 3.3150 5.8300 4.3700]
PaperSizeMode: 'auto'
    PaperSize: [8.5000 11]
    PaperType: 'usletter'
    PaperUnits: 'inches'
        Parent: [1x1 Root]
        Pointer: 'arrow'
PointerShapeCData: [16x16 double]
PointerShapeHotSpot: [1 1]
    Position: [348 480 583 437]
    Renderer: 'opengl'
    RendererMode: 'auto'
        Resize: on
        Scrollable: off
SelectionType: 'normal'
SizeChangedFcn: "
    Tag: "
ToolBar: 'none'
    Type: 'figure'
    Units: 'pixels'
UserData: []
Visible: off
WindowButtonDownFcn: "
WindowButtonMotionFcn: "
WindowButtonUpFcn: "
WindowKeyPressFcn: "
WindowKeyReleaseFcn: "
```

```
WindowScrollWheelFcn: "
    WindowState: 'normal'
    WindowStyle: 'normal'
    XDisplay: ':100'
```

图形对象层次结构

本节内容

- “MATLAB 图形对象” (第 17-9 页)
- “图形由具体对象组成” (第 17-9 页)
- “图形对象的组织” (第 17-9 页)

MATLAB 图形对象

图形对象是 MATLAB 用来以图形的形式显示数据的可视化组件。例如，图形可能包含线条、文本和坐标区，所有都显示在图窗窗口中。

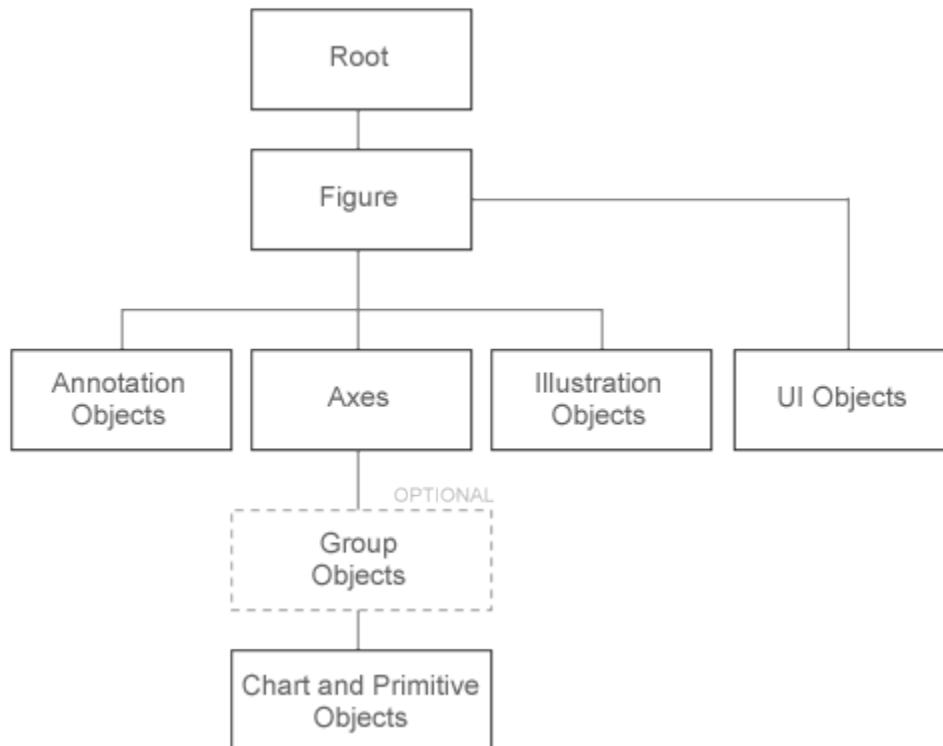
每个对象都有一个名为句柄的唯一标识符。使用该句柄，您可以通过设置对象属性来操作现有图形对象的特征。您还可以在创建图形对象时指定属性值。通常，使用如 `plot`、`bar`、`scatter` 等绘图函数创建图形对象。

图形由具体对象组成

当您创建图形时，例如通过调用 `plot` 函数创建，MATLAB 会自动执行一系列步骤生成图形。这些步骤包括创建对象和将这些对象属性值设置为适合特定图形的值。

图形对象的组织

这些图形对象按层次结构组织，如下图所示。

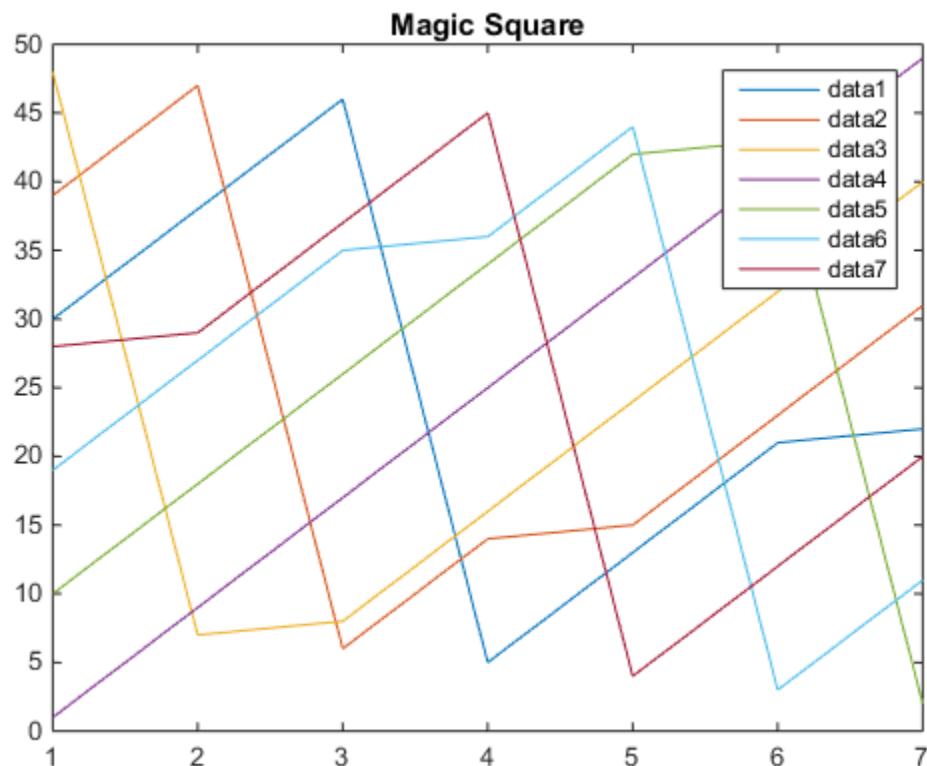


图形对象的层次结构本身反映出对象之间的包含关系。每个对象在图形显示中都具有特定角色。

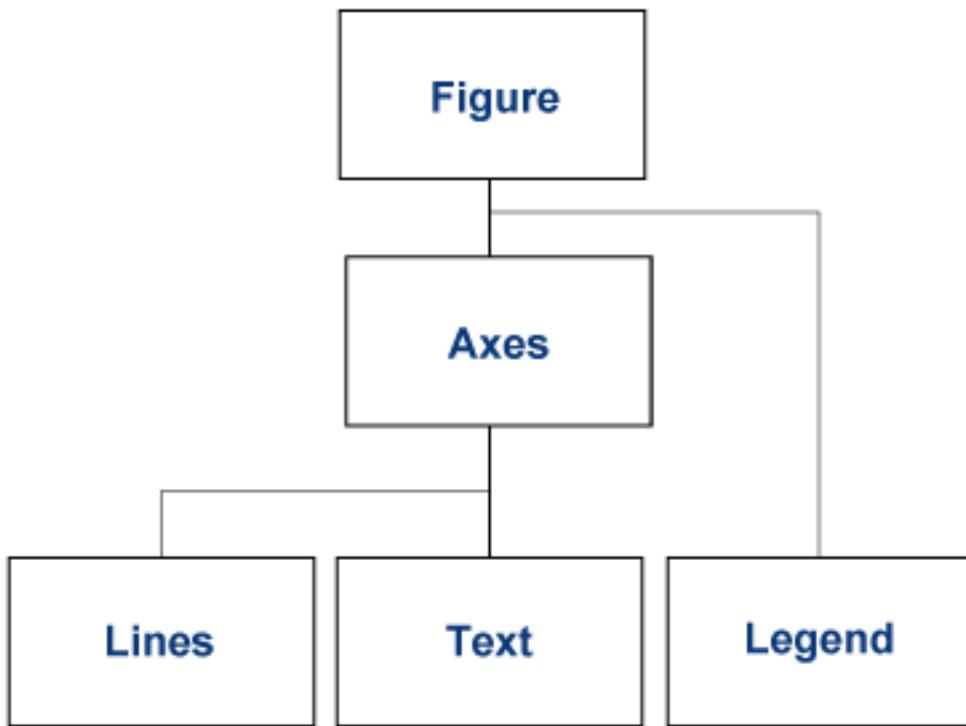
例如，您使用 `plot` 函数创建线图。坐标区对象为表示数据的线定义了参考系。图窗是显示图形的窗口。图窗包含坐标区，坐标区包含线条、文本、图例以及其他用于表示图形的对象。

注意 坐标区是表示 x、y 和 z 坐标区标度、刻度线、刻度标签、坐标区标签等对象的单个对象。

以下是一个简单的图形。



该图形形成了对象层次结构。



父子关系

对象间的关系保存在 **Parent** 和 **Children** 属性中。例如，坐标区的父级是一个图窗。坐标区的 **Parent** 属性包含了该坐标区所在图窗的句柄。

同样，图窗的 **Children** 属性包含了其所含的所有坐标区。图窗 **Children** 属性还包含其所含的其他所有对象，如图例和用户界面对象。

您可以使用父子关系找到其他对象句柄。例如，如果您创建一个绘图，当前坐标区 **Children** 属性包含了所有线条的句柄：

```

plot(rand(5))
ax = gca;
ax.Children

ans =
5x1 Line array:

Line
Line
Line
Line
Line
  
```

您还可以指定对象的父级。例如，创建一个组对象，让该组成为坐标区中线条的父级：

```

hg = hggroup;
plot(rand(5),'Parent',hg)
  
```

访问属性值

本节内容

- “对象属性和圆点表示法”（第 17-12 页）
- “图形对象变量是句柄”（第 17-13 页）
- “列出对象属性”（第 17-14 页）
- “使用 set 和 get 修改属性”（第 17-15 页）
- “多对象/属性操作”（第 17-15 页）

对象属性和圆点表示法

图形函数返回该函数创建的一个或多个对象。例如：

```
h = plot(1:10);
```

h 引用图形中绘制的值从 1 到 10 的线条。

圆点表示法语法使用对象变量和区分大小写的属性名以圆点(.) 相连，组成对象圆点名称表示法：

```
object.PropertyName
```

如果对象变量是非标量，那么使用索引引用单个对象：

```
object(n).PropertyName
```

标量对象变量

如果 **h** 是 **plot** 函数创建的线条，那么 **h.Color** 是特定线条的 **Color** 属性值：

```
h.Color
```

```
ans =
```

```
0 0.4470 0.7410
```

如果将颜色值分配给变量：

```
c = h.Color;
```

变量 **c** 是双精度值。

```
whos
```

Name	Size	Bytes	Class
c	1x3	24	double
h	1x1	112	matlab.graphics.chart.primitive.Line

您可以使用赋值语句更改线条的 **Color** 属性值：

```
h.Color = [0 0 1];
```

在表达式中使用圆点表示法属性引用：

```
meanY = mean(h.YData);
```

或更改属性值：

```
h.LineWidth = h.LineWidth + 0.5;
```

使用多个圆点引用来引用属性中包含的其他对象：

```
h.Annotation.LegendInformation.IconDisplayStyle
```

```
ans =
```

```
on
```

设置属性中包含的对象属性：

```
ax = gca;
ax.Title.FontWeight = 'normal';
```

非标量对象变量

图形函数可返回对象数组。例如：

```
y = rand(5);
h = plot(y);
size(h)
```

```
ans =
```

```
5    1
```

使用数组索引访问表示 y 中第一列的线条：

```
h(1).LineStyle = '-';
```

使用 set 函数设置数组中所有线条的 LineStyle：

```
set(h,'LineStyle','--')
```

将数据追加到属性值

通过圆点表示法，您可以使用 "end" 索引将数据追加到包含数据数组的属性，如线条 XData 和 YData。例如，这段代码同时更新 XData 和 YData，以延长线条。您必须确保线条的 x 和 y 数据的大小相同，然后才能通过调用 drawnow 进行渲染或返回 MATLAB 提示符。

```
h = plot(1:10);
for k = 1:5
    h.XData(end + 1) = h.XData(end) + k;
    h.YData(end + 1) = h.YData(end) + k;
    drawnow
end
```

图形对象变量是句柄

图形函数返回的对象变量是句柄。句柄是对实际对象的引用。对象变量是在复制时以及删除对象时具有特殊行为的句柄。

复制对象变量

例如，创建含有一个线条的图形：

```
h = plot(1:10);
```

现在将对象变量复制到另一个变量，并使用新对象变量设置属性值：

```
h2 = h;
h2.Color = [1,0,0]
```

将对象变量 **h** 分配给 **h2** 可创建句柄副本，而不是变量引用的对象。从变量 **h** 访问的 **Color** 属性值与从变量 **h2** 访问一样。

```
h.Color
```

```
ans =
```

```
1 0 0
```

h 和 **h2** 引用同一个对象。复制句柄对象变量不会复制对象。

删除对象变量

工作区中有两个对象变量引用同一个线条。

```
whos
```

Name	Size	Bytes	Class
h	1x1	112	matlab.graphics.chart.primitive.Line
h2	1x1	112	matlab.graphics.chart.primitive.Line

现在，关闭包含线图的图窗：

```
close gcf
```

线条对象不存在了，但引用此线条的对象变量仍然存在：

```
whos
```

Name	Size	Bytes	Class
h	1x1	112	matlab.graphics.chart.primitive.Line
h2	1x1	112	matlab.graphics.chart.primitive.Line

但此对象变量不再有效：

```
h.Color
```

```
Invalid or deleted object.
```

```
h2.Color = 'blue'
```

```
Invalid or deleted object.
```

要移除无效对象变量，使用 **clear**：

```
clear h h2
```

列出对象属性

要查看一个对象包含哪些属性，使用 **get** 函数：

```
get(h)
```

MATLAB 返回对象属性及其当前值列表：

```
AlignVertexCenters: 'off'
Annotation: [1x1 matlab.graphics.eventdata.Annotation]
BeingDeleted: 'off'
BusyAction: 'queue'
ButtonDownFcn: ''
Children: []
Clipping: 'on'
Color: [0 0.4470 0.7410]
...
LineStyle: '-'
LineWidth: 0.5000
Marker: 'none'
...
```

您可以使用 `set` 函数查看具有一组枚举值的属性值：

```
set(h,'LineStyle')
```

```
'-'  
'--'  
'.'  
'-'  
'none'
```

要显示所有可设置的属性，包括具有一组枚举值的可能属性值，使用 `set` 和对象变量：

```
set(h)
```

使用 `set` 和 `get` 修改属性

您还可以使用 `set` 和 `get` 函数访问以及修改属性。

设置现有对象属性值的基本语法是：

```
set(object,'PropertyName',NewPropertyValue)
```

要查询特定对象属性的当前值，使用以下形式的语句：

```
returned_value = get(object,'PropertyName');
```

属性名始终为字符向量。您可以使用单引号或字符向量形式的变量。属性值取决于特定的属性。

多对象/属性操作

如果对象参数是一个数组，那么 MATLAB 对所有识别出的对象设置特定的值。例如：

```
y = rand(5);
h = plot(y);
```

将所有线条设置为红色：

```
set(h,'Color','red')
```

要对多个对象设置相同属性，使用结构体或元胞数组指定属性名和属性值。例如，定义一个结构体，正确设置坐标区属性以显示特定图形：

```
view1.CameraViewAngleMode = 'manual';
view1.DataAspectRatio = [1 1 1];
view1.Projection = 'Perspective';
```

要在当前坐标区设置这些值，输入：

```
set(gca,view1)
```

查询多个属性

您可以定义属性名元胞数组并用它获取那些属性值。例如，假设您想要查询坐标区的“camera mode”属性值。首先，定义元胞数组：

```
camModes = {'CameraPositionMode','CameraTargetMode',...
'CameraUpVectorMode','CameraViewAngleMode'};
```

使用此元胞数组作为参数获取这些属性的当前值：

```
get(gca,camModes)
```

```
ans =
'auto' 'auto' 'auto' 'auto'
```

图形对象控制的功能

本节内容

- “图形对象的用途”（第 17-17 页）
- “图窗”（第 17-17 页）
- “坐标区”（第 17-18 页）
- “表示数据的对象”（第 17-18 页）
- “组对象”（第 17-19 页）
- “注释对象”（第 17-19 页）

图形对象的用途

图形对象以直观而有意义的方式表示数据，如线图、图像、文本以及这些对象的组合。图形对象作为其他对象的容器或数据的表示。

- 容器 - 图窗显示所有图形对象。通过面板和组可将一组对象视作一个整体进行某些操作。
- 坐标区是为图形中表示实际数据的对象定义坐标系的容器。
- 数据可视化对象 - 实现各种类型图形的线条、文本、图像、曲面和填充。

图窗

图窗是 MATLAB 图形显示于其中的窗口。图窗包含菜单、工具栏、用户界面对象、上下文菜单和坐标区。

图窗在 MATLAB 中扮演两种不同的角色：

- 包含数据图形
- 包含用户界面（界面中可以包含图形）

图窗控制的图形功能

图窗属性控制某些会影响图形的特征：

- 曲面、补片的颜色和透明度 - **Alphamap** 和 **Colormap**
- 绘制的线条和坐标区网格线的外观 - **GraphicsSmoothing**
- 打印和导出图形 - 图窗打印属性
- 绘图速度和渲染功能 - **Renderer**

图窗使用不同的绘图方法，这些方法称为渲染器。有两种渲染器：

- OpenGL - MATLAB 用于大多数应用程序的默认渲染器。有关详细信息，请参阅 **opengl**。
- Painters - 当 OpenGL 在计算机上遇到特定的图形硬件问题时使用，这些硬件可能具有软件缺陷或过时的软件驱动程序。它也可用于将图形导出到特定格式，如 PDF。

注意 为了获得最佳效果，确保您的计算机含有硬件供应商提供的最新图形硬件驱动程序。

有关所有图窗属性列表，请参阅 **Figure**

坐标区

MATLAB 创建坐标区以定义每个图形的坐标系。坐标区通常包含在图窗对象中。坐标区自身也包含表示数据的图形对象。

坐标区控制 MATLAB 如何显示图形信息的多个方面。

坐标区控制的图形功能

在图形中可自定义的大多数内容都由坐标区属性控制。

- 坐标轴范围、方向和刻度放置
- 轴标度（线性或对数）
- 网格控件
- 标题和轴标签的字体特征。
- 定义多线条图形的默认线条颜色和线型
- 轴线条和网格控件
- 根据颜色图调整对象颜色
- 视图和固定纵横比
- 按坐标轴范围裁剪图形
- 控制坐标区调整大小行为
- 光照和透明度控制

有关所有坐标区属性的列表，请参阅 [Axes](#)。

表示数据的对象

数据对象是图形用于表示数据的线条、图像、文本和多边形。例如：

- 线条使用特定的 x 和 y 坐标连接数据点。
- 标记定位某一范围值中的散点数据。
- 矩形条指示直方图中值的分布。

由于存在许多种图形，因此也存在许多数据对象类型。其中一些用于一般用途，例如线条和矩形，还有一些是用于高度专业的用途，例如误差条、颜色栏和图例。

数据对象控制的图形功能

数据对象属性控制对象外观，而且包含定义对象的数据。数据对象属性还可以控制某些行为。

- 数据 - 更改数据，更新图形。很多数据对象可将其数据属性链接到包含数据的工作区变量。
- 颜色数据 - 对象可控制如何通过指定颜色数据将数据映射到颜色。
- 外观 - 指定线条颜色、标记、多边形表面以及线型、标记类型。
- 特定行为 - 属性可以控制对象如何解析或显示其数据。例如，条形对象有一个名为 `BarLayout` 的属性，该属性确定条形是分组还是堆叠。等高线对象有一个 `LevelList` 属性，该属性确定绘制等高线的等高线区间。

高级函数与低级函数

绘图函数通过以下两种方法之一创建对象：

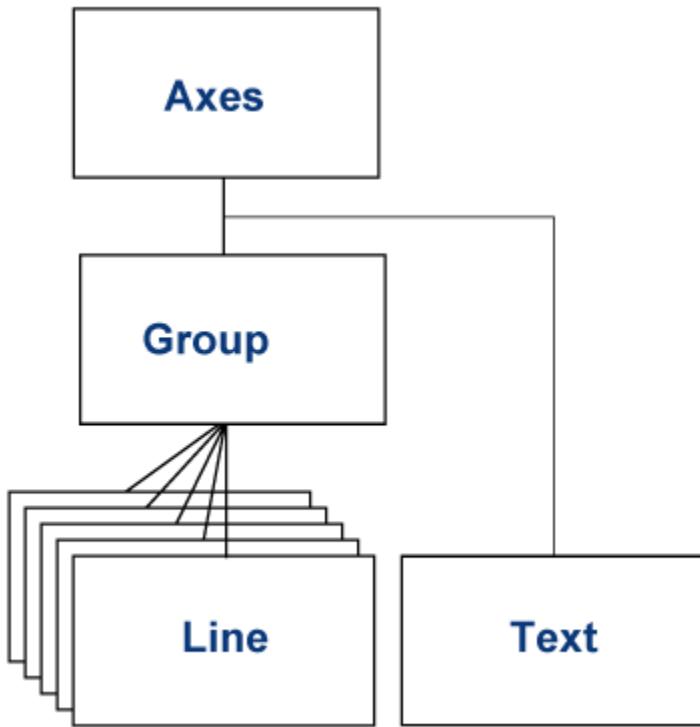
- 高级函数 - 创建整个图形，用新图形替换已有图形。高级函数包括 **plot**、**bar**、**scatter** 等等。有关高级函数摘要，请参阅“MATLAB 绘图类型”（第 1-2 页）。
- 低级函数 - 添加图形对象，对现有图形做最少的更改。低级函数包括 **line**、**patch**、**rectangle**、**surface**、**text**、**image** 和 **light**。

组对象

组对象可以将多个数据对象作为一个整体。例如，您可以将整个组设置为可见或不可见，当点击其中一个时选择所有对象，或应用一个变换矩阵以旋转、转换或缩放组中的所有对象。

以下代码将 **hggroup** 函数返回的组对象设定为所绘线条的父级。文本对象不是组的一部分。

```
y = magic(5);
hg = hggroup;
plot(y,'Parent',hg)
text(2.5,10,'Plot of 5x5 magic square')
```



注释对象

注释对象包含箭头、文本框以及两者组合。注释对象有特殊的功能，这些功能突破数据对象的限制，可用 来注释图形：

- 注释对象是图窗的子级。

- 您可以在图窗中轻松定位注释。
- 以归一化图窗坐标定义注释对象的位置：左下角 = (0,0)，右上角 = (1,1)，这使得其定位独立于用坐标区表示的数据范围。

注意 MATLAB 将特定层设定为注释对象的父级。不要尝试将该层作为对象父级。MATLAB 会自动将注释对象分配到合适的父级。

默认属性值

本节内容
“属性预定义值” (第 17-21 页)
“指定默认值” (第 17-21 页)
“在层次结构中什么位置定义默认值” (第 17-22 页)
“列出默认值” (第 17-22 页)
“将属性设置为当前默认值” (第 17-22 页)
“删除默认值” (第 17-22 页)
“将属性设置为出厂定义值” (第 17-22 页)
“列出出厂定义属性值” (第 17-23 页)
“保留字” (第 17-23 页)

属性预定义值

几乎所有图形对象都有预定义值。预定义值有两个可能的来源：

- 在对象前代中定义的默认值
- 在图形对象层次结构的根级定义的出厂值

用户可以为对象属性创建默认值，且它的优先级比出厂定义值高。以下情形，对象使用默认值：

- 在前代定义默认值的层次结构中创建
- 在前代定义默认值的层次结构中作为父对象

指定默认值

使用包含以下三部分的字符向量定义默认属性值：

```
'default' ObjectType PropertyName
```

- **default** 单词
- 对象类型（例如，**Line**）
- 属性名（例如，**LineWidth**）

指定默认线条 **LineWidth** 的字符向量如下所示：

```
'defaultLineLineWidth'
```

使用此字符向量指定默认值。例如，要将线条 **LineWidth** 属性指定为 2 磅，使用以下语句：

```
set(groot,'defaultLineLineWidth',2)
```

字符向量 **defaultLineLineWidth** 确定此属性为线条属性。要指定图窗颜色，使用 **defaultFigureColor**：

```
set(groot,'defaultFigureColor','b')
```

在层次结构中什么位置定义默认值

通常应该在根级别定义默认值，从而后续绘图函数都可以使用这些默认值。在 `set` 和 `get` 语句中使用 `groot` 函数指定根对象，此函数会返回根对象的句柄。

您可以在三个级别上定义默认属性值：

- 根 - 值应用于在当前 MATLAB 会话中创建的对象
- 图窗 - 用于将默认值应用到定义默认值的图窗的子对象。
- 坐标区 - 用于将默认值应用到定义默认值的坐标区的子对象，而且仅当使用低级函数 (`light`、`line`、`patch`、`rectangle`、`surface`、`text` 和 `image` 低级形式) 时。

例如，只在根级别指定默认图窗颜色。

```
set(groot,'defaultFigureColor','b')
```

列出默认值

使用 `get` 可确定当前在任意给定对象级别设置了哪些默认值：

```
get(groot,'default')
```

返回在当前 MATLAB 会话中设置的所有默认值。

将属性设置为当前默认值

指定属性值为 '`default`' 将会让属性设置为首先遇到的为该属性定义的默认值。例如，这些语句生成绿色曲面 `EdgeColor`：

```
set(groot,'defaultSurfaceEdgeColor','k')
h = surface(peaks);
set(gcf,'defaultSurfaceEdgeColor','g')
set(h,'EdgeColor','default')
```

由于曲面的 `EdgeColor` 的默认值在图窗级别，MATLAB 首先遇到这个值，并使用该值代替根上定义的默认 `EdgeColor`。

删除默认值

指定属性值为 '`remove`' 会去除用户定义的默认值。语句

```
set(groot,'defaultSurfaceEdgeColor','remove')
```

从根移除默认曲面 `EdgeColor` 定义。

将属性设置为出厂定义值

指定属性值为 '`factory`' 将会让属性设置为其出厂定义值。例如，这些语句将曲面 `h` 的 `EdgeColor` 设置为黑色（其出厂设定），无论您定义的默认定义值为何值：

```
set(gcf,'defaultSurfaceEdgeColor','g')
h = surface(peaks);
set(h,'EdgeColor','factory')
```

列出出厂定义属性值

您可以列出出厂值：

- `get(groot,'factory')` - 列出所有图形对象的出厂定义属性值
- `get(groot,'factoryObjectType')` - 列出特定对象的出厂定义属性值
- `get(groot,'factoryObjectTypePropertyName')` - 列出指定属性的出厂定义值。

保留字

将属性值设置为 `default`、`remove` 或 `factory` 会得到前面部分所描述的结果。要将属性设置为这些保留字（例如，文本 `String` 属性设置为保留字 `default`），在该保留字前加上反斜杠：

```
h = text('String','\default');
```

自动计算的属性的默认值

本节内容

“什么是自动计算的属性” (第 17-24 页)

“自动计算的属性的默认值” (第 17-24 页)

什么是自动计算的属性

当您创建图形时，MATLAB 会为这一特定图形适当设置某些属性值。这些属性，如控制坐标轴范围和图窗渲染器的属性，具有相关联的模式属性。

这一模式属性确定 MATLAB 是否计算此属性值（模式为 `auto`）或属性是否使用指定值（模式为 `manual`）。

自动计算的属性的默认值

定义自动计算的属性的默认值需要两步：

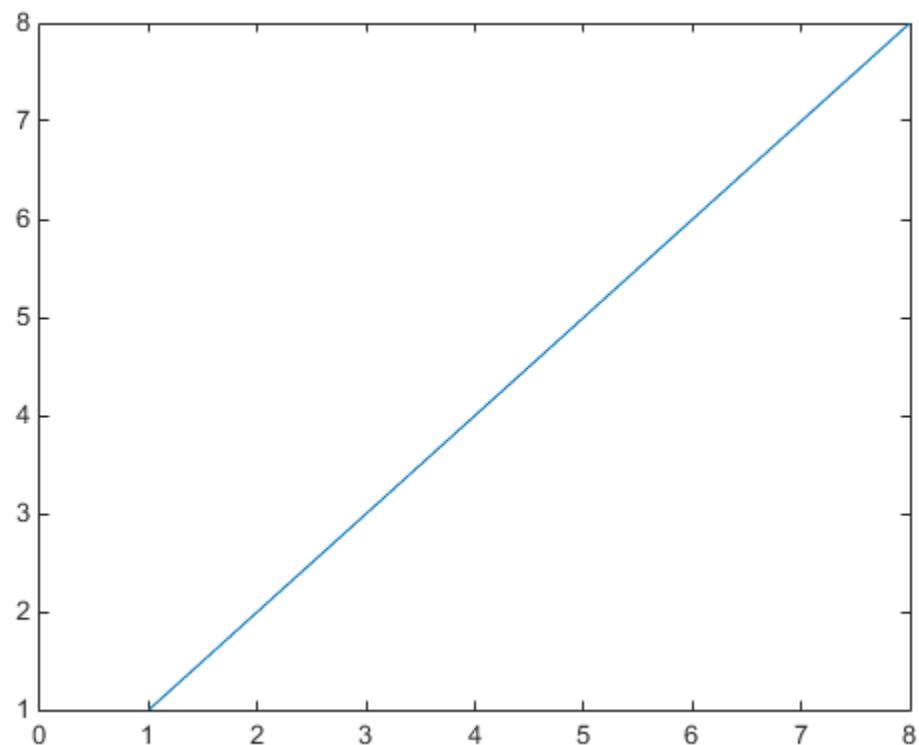
- 定义属性默认值
- 将模式属性的默认值定义为 `manual`

设置 X 坐标轴范围

假设您要定义 x 坐标轴范围默认值。由于坐标区 `XLim` 属性通常为自动计算，您必须将相关联的模式属性 (`XLimMode`) 设置为 `manual`。

```
set(groot,'defaultAxesXLim',[0 8])
set(groot,'defaultAxesXLimMode','manual')
plot(1:20)
```

这些坐标区使用默认 x 坐标轴范围 [0 8]：



MATLAB 如何找到默认值

MATLAB 中的图形对象都有其内置属性值。这些值称为出厂定义值。任何不指定值的属性都使用预定义的值。

您也可以定义自己的默认值。MATLAB 会使用您的默认值，除非您在创建该对象时指定了属性值。

MATLAB 从当前对象开始查找默认值，然后查找对象的前代，直到找到用户定义默认值或找到出厂定义值。因此，总能找到属性值。

根据这些步骤，MATLAB 确定给定的属性使用哪个值：

- 1 属性默认值被指定为绘图函数参数
- 2 如果对象是由高级绘图函数，如 `plot` 所创建的线条，那么坐标区 `ColorOrder` 和 `LineStyleOrder` 定义会覆盖 `Color` 或 `LineStyle` 属性的默认值。
- 3 坐标区定义的属性默认值（默认值可被绘图函数清除）
- 4 图窗定义的属性默认值
- 5 根定义的属性默认值
- 6 如果未定义默认值，使用出厂默认值

设置默认值仅影响您设置默认值之后创建的对象。现有图形对象不受影响。

出厂定义属性值

MATLAB 定义了所有图形对象属性值。如果您未指定参数值或默认值，绘图函数将使用这些值。使用以下语句生成出厂定义值列表

```
a = get(groot,'Factory');
```

get 返回一个结构数组，它的字段名称是对象类型和属性名称相连而成，字段值是指示对象和属性的出厂值。例如，字段

```
factoryAxesVisible: 'on'
```

指示坐标区对象的 **Visible** 属性出厂值是 **on**。

您可以使用以下函数获得单个属性的出厂值

```
get(groot,'factoryObjectTypePropertyName')
```

例如：

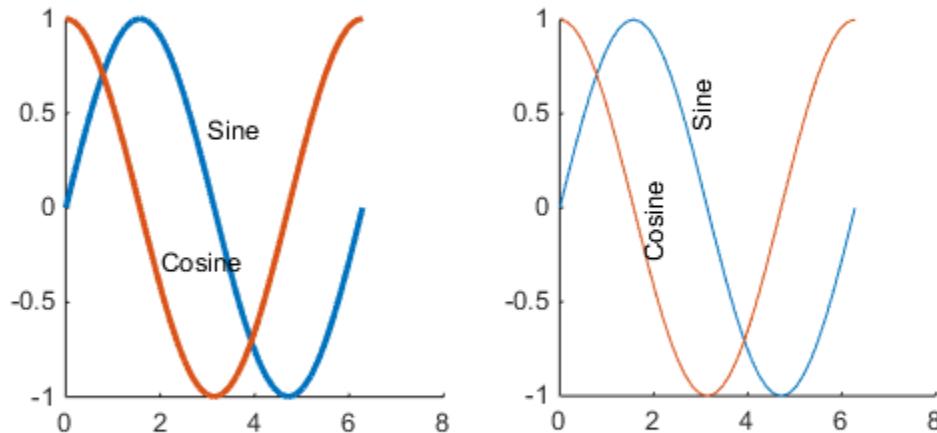
```
get(groot,'factoryTextFontName')
```

多级默认值

此示例在层次结构的多个级别中设置默认值。这些语句在一个图窗窗口中创建两个坐标区，在图窗级和坐标区级设置默认值：

```
t = 0:pi/20:2*pi;
s = sin(t);
c = cos(t);
figure('defaultAxesPlotBoxAspectRatio',[1 1 1],...
    'defaultAxesPlotBoxAspectRatioMode','manual');
subplot(1,2,1,'defaultLineLineWidth',2);
hold on
plot(t,s,t,c)
text('Position',[3 0.4],'String','Sine')
text('Position',[2 -0.3],'String','Cosine')

subplot(1,2,2,'defaultTextRotation',90);
hold on
plot(t,s,t,c)
text('Position',[3 0.4],'String','Sine')
text('Position',[2 -0.3],'String','Cosine')
```



对每个子图使用同样的 `plot` 和 `text` 语句会生成不同的显示，这反映出为坐标区定义的默认值不同。图窗级别定义的默认值应用到两个坐标区。

需要调用 `hold on` 以防止 `plot` 函数重置坐标区属性。

注意 如果属性有关联模式属性（例如，`PlotBoxAspectRatio` 和 `PlotBoxAspectRatioMode`），那么在定义关联属性默认值时，必须定义模式属性的 `manual` 默认值。

对象标识

- “特殊对象标识符”（第 18-2 页）
- “查找对象”（第 18-4 页）
- “复制对象”（第 18-8 页）
- “删除图形对象”（第 18-10 页）

特殊对象标识符

本节内容

- “获取特殊对象的句柄”（第 18-2 页）
- “当前图窗、坐标区和对象”（第 18-2 页）
- “回调对象和回调图窗”（第 18-3 页）

获取特殊对象的句柄

MATLAB 提供了返回重要对象句柄的函数，从而您可以根据需要随时获取它们。

这些对象包括：

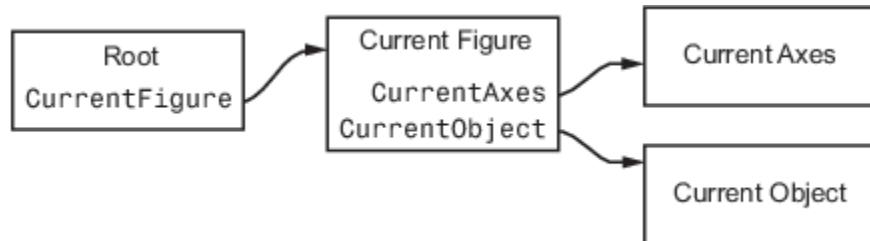
- 当前图窗 - 图形命令当前目标的图窗句柄。
- 当前坐标区 - 作为图形命令目标的当前图窗中的坐标区句柄。
- 当前对象 - 选中对象的句柄
- 回调对象 - 其回调正在执行的对象句柄。
- 回调图窗 - 回调对象的父级图窗的句柄。

当前图窗、坐标区和对象

MATLAB 图形中一个重要概念是当前对象。成为当前对象意味着该对象成为所有会影响该类型对象的操作的目标。有三个对象可在任意时刻指定为当前对象：

- 当前图窗是指定接收图形输出的窗口。
- 当前坐标区是绘图函数显示图形的坐标区。
- 当前对象是最新创建或选中的对象。

MATLAB 将对应于这些对象的句柄存储在前代相应的属性中。



这些属性可以让您获得这些关键对象的句柄：

```

hRoot = groot;
hFigure = hRoot.CurrentFigure;
hAxes = hFigure.CurrentAxes;
hobj = hFigure.CurrentObject;
  
```

便利函数

以下命令是这些属性查询的简写表示。

- **gcf** - 返回根 **CurrentFigure** 属性值；如果没有当前图窗，则创建一个图窗。**HandleVisibility** 属性设置为 **off** 的图窗不能成为当前图窗。
- **gca** - 返回当前图窗的 **CurrentAxes** 属性值，如果没有当前图窗，则创建一个坐标区。**HandleVisibility** 属性设置为 **off** 的坐标区不能成为当前坐标区。
- **gco** - 返回当前图窗的 **CurrentObject** 属性值。

使用这些命令作为需要对象句柄的函数的输入参数。例如，可以点击线条对象然后使用 **gco** 将句柄指定给 **set** 命令，

```
set(gco,'Marker','square')
```

或点击坐标区对象以设置坐标区属性：

```
set(gca,'Color','black')
```

您可以获得当前坐标区中所有图形对象的句柄（隐藏的句柄除外）：

```
h = get(gca,'Children');
```

然后确定对象的类型：

```
get(h,'Type')
```

```
ans =
'text'
'patch'
'surface'
'line'
```

尽管 **gcf** 和 **gca** 提供了简单的方法获取当前图窗和坐标区句柄，它们在代码文件中的用处不大。如果您的代码是 MATLAB 上层应用程序的一部分，您不知道其中哪些用户操作会更改这些值，则上述方法尤其有用。

有关如何防止用户访问您想要保护的图形对象句柄的详细信息，请参阅“防止对图窗和坐标区访问”（第 22-11 页）。

回调对象和回调图窗

回调函数经常需要有关定义回调的对象或包含其回调正在执行的图窗的信息。要获得句柄、这些对象，使用以下便利函数：

- **gcbo** - 返回根 **CallbackObject** 属性值。该属性包含其回调正在执行的对象句柄。**gcbo** 还可以返回包含回调对象的图窗句柄。
- **gcbf** - 返回包含回调对象的图窗句柄。

MATLAB 保持 **CallbackObject** 属性值与当前执行的回调同步。如果有回调中断正在执行的回调，MATLAB 会更新 **CallbackObject** 属性值。

在编写 **CreateFcn** 和 **DeleteFcn** 回调函数时，一定要使用 **gcbo** 引用回调对象。

有关编写回调函数的详细信息，请参阅“回调定义”（第 20-3 页）

查找对象

本节内容

- “查找具有特定属性值的对象” (第 18-4 页)
- “通过字符串属性查找文本” (第 18-4 页)
- “使用含有 `findobj` 的正则表达式” (第 18-5 页)
- “限制搜索范围” (第 18-6 页)

查找具有特定属性值的对象

`findobj` 函数可以扫描对象层次结构以获得具有特定属性值的对象句柄。

为便于标识，所有图形对象均具有可设为任意字符向量的 `Tag` 属性。您随后可以查找特定的属性/值对组。例如，假设创建了一个有时会在 UI 中停用的复选框。通过将唯一的值赋给 `Tag` 属性，您可以找到这个特定对象：

```
uicontrol('Style','checkbox','Tag','save option')
```

使用 `findobj` 找到 `Tag` 属性值设为 'save option' 的对象并禁用它：

```
hCheckbox = findobj('Tag','save option');  
hCheckbox.Enable = 'off'
```

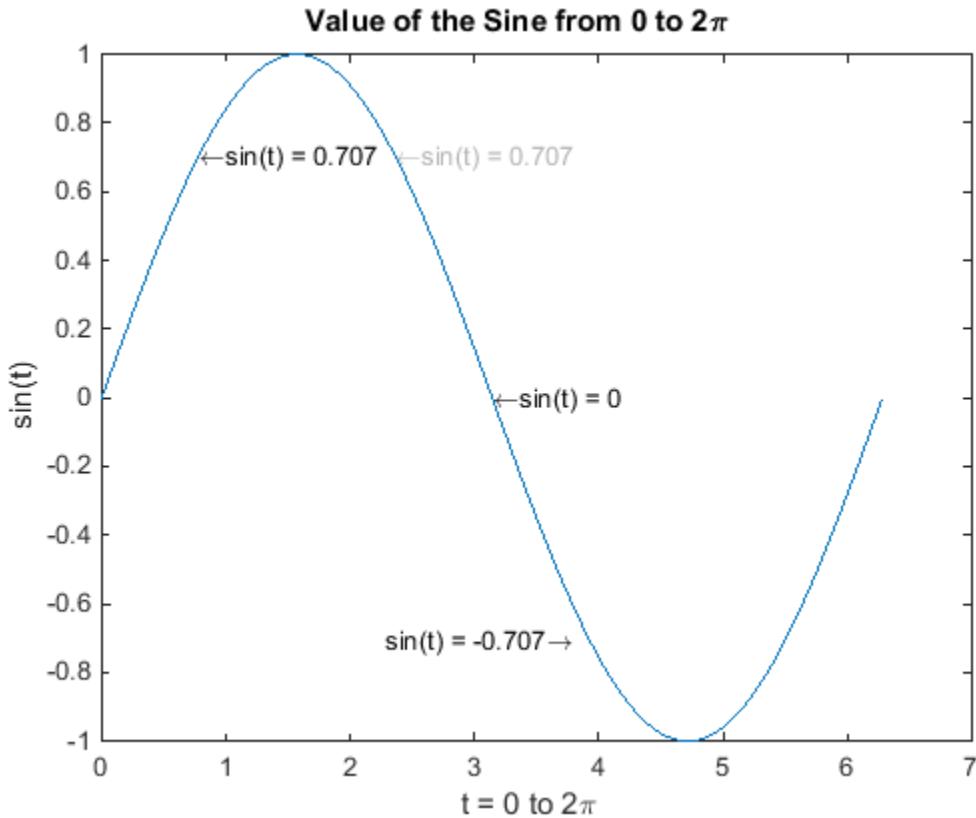
如果未指定起始对象，`findobj` 会从根对象开始，查找所有出现您所指定的名称/属性值对组的对象。

要查找隐藏句柄对象，使用 `findall`。

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)。

确定标记点 $[\pi/4, \sin(\pi/4)]$ 的文本对象句柄并更改其 **Position** 属性。

要使用 **findobj**, 选取能唯一标识该对象的属性值。此例使用文本 **String** 属性:

```
hText = findobj('String','\leftarrow\sin(t) = .707');
```

将对象移动到新位置, 以坐标区单位定义文本 **Position**。

```
hText.Position = [3*pi/4,sin(3*pi/4),0];
```

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:

```
hText = findobj(gca,'String','\leftarrow\sin(t) = .707');
```

使用含有 **findobj** 的正则表达式

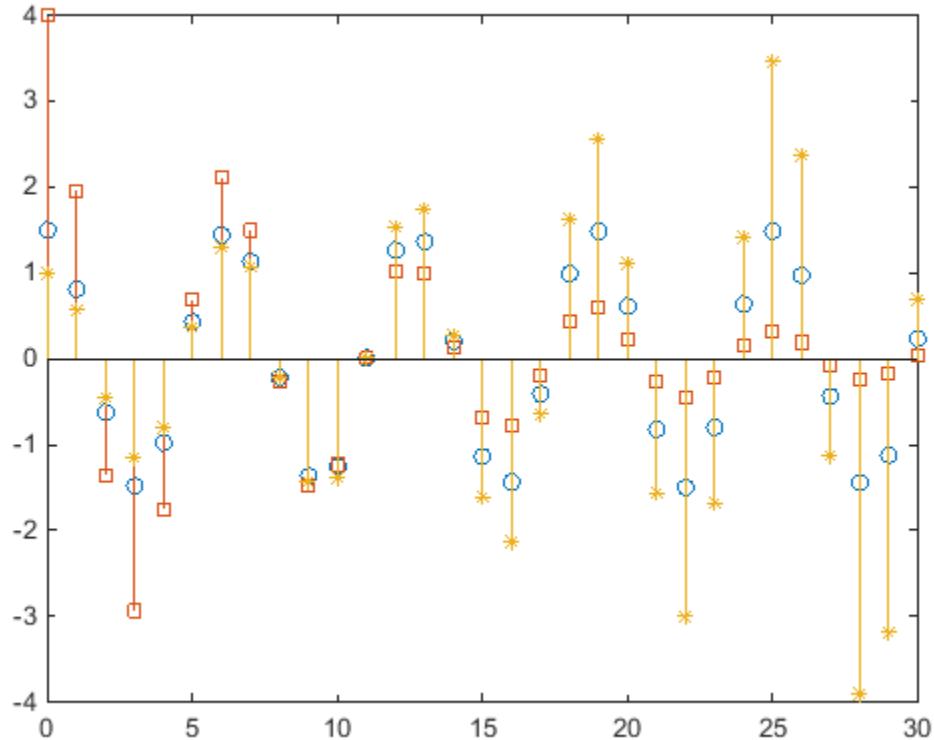
此示例说明如何使用正则表达式找出特定的属性值, 从而找到对象句柄。有关正则表达式的详细信息, 请参阅 **regexp**。

假设您创建了以下图形并想要修改所创建对象的某些属性。

```

x = 0:30;
y = [1.5*cos(x);4*exp(-.1*x).*cos(x);exp(.05*x).*cos(x)]';
h = stem(x,y);
h(1).Marker = 'o';
h(1).Tag = 'Decaying Exponential';
h(2).Marker = 'square';
h(2).Tag = 'Growing Exponential';
h(3).Marker = '*';
h(3).Tag = 'Steady State';

```



将正则表达式传递给 `findobj` 可以匹配特定模式。例如，假设您想要将 Tag 属性未设置成 'Steady State' 的所有针状图对象（即表示衰减和增长指数的针状图）的 `MarkerFaceColor` 属性的值设置成绿色。

```

hStems = findobj(~regexp,'Tag','^(?!Steady State$).');
for k = 1:length(hStems)
    hStems(k).MarkerFaceColor = 'green'
end

```

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：

```

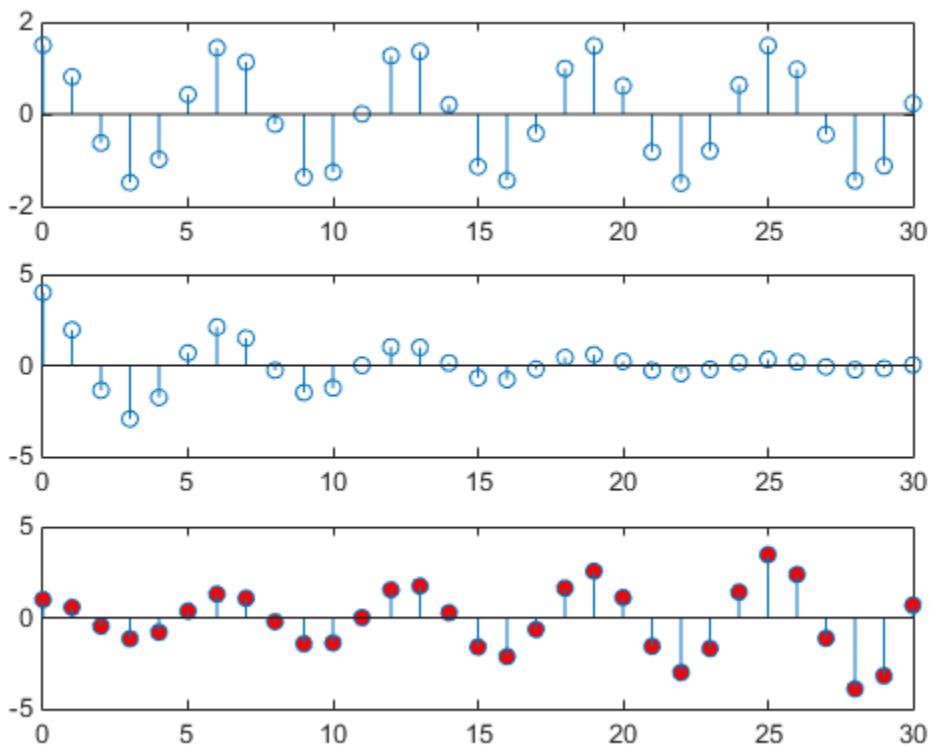
x = 0:30;
y = [1.5*cos(x);4*exp(-.1*x).*cos(x);exp(.05*x).*cos(x)]';

```

```
tiledlayout(3,1)
ax1 = nexttile;
stem(x,y(:,1))
ax2 = nexttile;
stem(x,y(:,2))
ax3 = nexttile;
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.MarkerFaceColor = 'red';
```



有关限制搜索及对象搜索深度的详细信息，请参阅 [findobj](#) 和 [findall](#)。

复制对象

本节内容

- “使用 `copyobj` 复制对象” (第 18-8 页)
- “将单个对象复制到多个目的地。” (第 18-8 页)
- “复制多个对象” (第 18-8 页)

使用 `copyobj` 复制对象

使用 `copyobj` 函数将对象从一个父级复制到另一个。副本与原始对象存在以下不同之处：

- `Parent` 属性是新的父对象。
- 复制对象的句柄与原始句柄不同。
- `copyobj` 不会复制原始对象的回调属性。
- `copyobj` 不会复制与原始对象关联的任何应用程序数据。

因此，在比较原始句柄与新句柄时，`==` 和 `isequal` 会返回 `false`。

您可以将多个不同的对象复制到一个新的父对象，或将一个对象复制到多个新的父对象，只要结果保持正确的父/子关系即可。当复制包含子对象的对象时，MATLAB 也会复制所有子对象。

注意 不能将同一对象多次复制到对 `copyobj` 的单个调用中的同一父级。

将单个对象复制到多个目的地。

在将单个对象复制到多个目的地时，`copyobj` 所返回的新句柄顺序与父句柄一致。

```
h = copyobj(cobj,[newParent1,newParent2,newParent3])
```

返回的数组 `h` 所包含新对象句柄顺序如下所示：

```
h(1) -> newParent1
h(2) -> newParent2
h(3) -> newParent3
```

复制多个对象

此示例演示如何将多个对象复制到单个父对象。

假设您想要创建一组类似的图形，并在每个图形中标记相同的数据点。您可以将在第一个图形中标记点的文本和标记对象复制到每个后续图形中。

创建和标记第一个图形：

```
x = 0:.1:2*pi;
plot(x,sin(x))
hText = text('String','\{5\pi\div4, sin(5\pi\div4)\}\rightarrow',...
    'Position',[5*pi/4,sin(5*pi/4),0],...
    'HorizontalAlignment','right');
hMarker = line(5*pi/4,sin(5*pi/4),0,'Marker','*');
```

创建两个以上没有标签的图形：

```
figure
x = pi/4:1:9*pi/4;
plot(x,sin(x))
hAxes1 = gca;

figure
x = pi/2:1:5*pi/2;
plot(x,sin(x))
hAxes2 = gca;
```

通过将文本和标记（**hText** 和 **hMarker**）的父级设置为图形各自所在的坐标区将文本和标记复制到每个图形。返回文本和标记副本的新句柄：

```
newHandles1 = copyobj([hText,hMarker],hAxes1);
newHandles2 = copyobj([hText,hMarker],hAxes2);
```

由于目的是将两个对象复制到每个坐标区，因此请调用 **copyobj** 两次，每次一个目标坐标区。

将多个对象复制到多个目的地

当使用多个复制对象及多个目的地调用 **copyobj** 时，**copyobj** 会将各个对象复制到各个父级。也就是说，如果 **h** 和 **p** 为长度为 **n** 的句柄数组，那么调用 **copyobj**：

```
copyobj(h,p)
```

会生成按元素的副本：

```
h(1) -> p(1);
h(2) -> p(2);
...
h(n) -> p(n);
```

删除图形对象

本节内容

- “如何删除图形对象”（第 18-10 页）
- “已删除对象的句柄”（第 18-11 页）

如何删除图形对象

使用 **delete** 函数删除图形对象。将对象句柄作为参数传递给 **delete**。例如，使用以下语句可删除当前坐标区以及坐标区中包含的所有对象。

```
delete(gca)
```

如果想要删除多个对象，可将句柄数组传递给 **delete**。例如，如果 **h1**、**h2** 和 **h3** 是您想要删除的图形对象的句柄，请将这些句柄串联成单个数组。

```
h = [h1,h2,h3];
delete(h)
```

关闭图窗会删除图窗中包含的所有对象。例如，创建一个条形图。

```
f = figure;
y = rand(1,5);
bar(y)
```

该图窗现在包含坐标区和条形对象。

```
ax = f.Children;
b = ax.Children;
```

关闭图窗：

```
close(f)
```

MATLAB 会删除每个对象。

```
f
f =
handle to deleted Figure
ax
ax =
handle to deleted Axes
b
b =
handle to deleted Bar
```

已删除对象的句柄

当您删除一个图形对象时，MATLAB 不会删除包含此对象句柄的变量。但是，该变量将变成无效句柄，因为其引用的对象不再存在。

您可以通过显式使用 `delete` 函数或关闭包含图形对象的图窗来删除图形对象。例如，创建一个条形图。

```
f = figure;
y = rand(1,5);
b = bar(y);
```

关闭包含该条形图的图窗。

```
close(f)
```

关闭图窗后，句柄变量仍然存在，但图形对象不再存在。

```
whos
```

Name	Size	Bytes	Class
f	1x1	104	matlab.ui.Figure
b	1x1	104	matlab.graphics.chart.primitive.Bar
y	1x5	40	double

使用 `isgraphics` 确定图形对象句柄的有效性。

```
isgraphics(b)
```

```
ans =
```

```
0
```

您不能通过无效的句柄变量来访问属性。

```
h.FaceColor
```

Invalid or deleted object.

要移除该变量，请使用 `clear` 函数。

```
clear h
```

另请参阅

`isValid`

相关示例

- “查找对象”（第 18-4 页）

处理图形对象

- “图形对象句柄” (第 19-2 页)
- “预分配图形对象数组” (第 19-4 页)
- “检验有效句柄” (第 19-5 页)
- “逻辑表达式中的句柄” (第 19-6 页)
- “图形数组” (第 19-8 页)

图形对象句柄

本节内容

- “您可以使用句柄做什么”（第 19-2 页）
- “句柄不具有的功能”（第 19-2 页）

您可以使用句柄做什么

句柄引用图形对象的具体实例。使用对象句柄设置和查询对象属性的值。

当创建图形对象时，可以将对象的句柄保存到变量中。例如：

```
x = 1:10;
y = x.^2;
plot(x,y);
h = text(5,25,'*(5,25)');
```

变量 `h` 引用特定的文本对象 `'*(5,25)'`，它位于点 `5,25`。使用句柄 `h` 查询和设置该文本对象的属性。

设置字号

```
h.FontSize = 12;
```

获取字号

```
h.FontSize
```

```
ans =
```

```
12
```

创建变量 `h` 的副本。此副本引用同一个对象。例如，下列语句创建句柄的副本而不是对象：

```
hNew = h;
hNew.FontAngle = 'italic';
h.FontAngle
```

```
ans =
```

```
italic
```

句柄不具有的功能

句柄变量是对象。不要尝试对句柄执行将句柄转换为数值、字符或其他类型的操作。例如，不能执行以下操作：

- 对句柄执行算术运算。
- 在逻辑语句中直接使用句柄，而不转换为逻辑值。
- 在逻辑语句中根据图窗句柄的数值（整数）做判断。
- 将句柄与数字数组中的数据合并。
- 将句柄转换为字符向量，或在字符向量运算中使用句柄。

另请参阅

详细信息

- “图形数组” (第 19-8 页)
- “Dominant Argument in Overloaded Graphics Functions”

预分配图形对象数组

使用 `gobjects` 函数为图形对象预分配数组。您可以为数组中每个元素填入图形对象句柄。

预分配一个 4×1 数组：

```
h = gobjects(4,1);
```

将坐标区句柄分配给数组元素：

```
tiledlayout(2,2)
for k=1:4
    h(k) = nexttile;
end
```

`gobjects` 返回 `GraphicsPlaceholder` 数组。您可以用任意类型的图形对象替换这些占位符元素。必须使用 `gobjects` 预分配图形对象数组以确保所有分配到数组的图形对象兼容。

检验有效句柄

使用 `isgraphics` 确定变量是否是有效的图形对象句柄。句柄变量（在本例中是 `h`）仍可存在，但如果其引用的对象已删除，则不是有效句柄。

```
h = plot(1:10);
...
close % Close the figure containing the plot
whos
```

Name	Size	Bytes	Class	Attributes
h	1x1	104	matlab.graphics.chart.primitive.Line	

检验 `h` 的有效性：

```
isgraphics(h)
```

```
ans =
```

```
0
```

有关被删除句柄的详细信息，请参阅“[删除的句柄对象](#)”。

逻辑表达式中的句柄

本节内容

- “如果句柄有效” (第 19-6 页)
- “如果结果为空” (第 19-6 页)
- “如果句柄相等” (第 19-7 页)

句柄对象不会计算成逻辑 true 或 false。您必须使用函数检验关注的状态并返回逻辑值。

如果句柄有效

使用 `isgraphics` 确定变量是否包含有效图形对象句柄。例如，假定 `hobj` 是工作区中的变量。对变量进行操作前，检验其有效性：

```
if isgraphics(hobj)
...
end
```

您还可以确定对象的类型：

```
if isgraphics(hobj,'figure')
...% hobj is a figure handle
end
```

如果结果为空

您不能在逻辑语句中直接使用空对象。使用 `isempty` 返回可以在逻辑语句中使用的逻辑值。

有些属性包含其他对象的句柄。如果其他对象不存在，则该属性包含空对象：

```
close all
hRoot = groot;
hRoot.CurrentFigure

ans =
0x0 empty GraphicsPlaceholder array.
```

例如，要通过查询根 `CurrentFigure` 属性确定是否存在当前图窗，使用 `isempty` 函数：

```
hRoot = groot;
if ~isempty(hRoot.CurrentFigure)
... % There is a current figure
end
```

代码可能遇到空对象的另一个例子是搜索句柄的情形。例如，假设将一个图窗的 `Tag` 属性设置为字符串向量 '`myFigure`'，并使用 `findobj` 获取该图窗的句柄：

```
if isempty(findobj('Tag','myFigure'))
... % That figure was NOT found
end
```

如果没有匹配项，则 `findobj` 返回一个空对象。

如果句柄相等

句柄相等有两种状态：

- 两个句柄引用同一个对象（用 `==` 检验）。
- 两个句柄所引用的对象属于同一个类，且所有属性值相等（用 `isequal` 检验）。

假设您想要确定 `h` 是否是 `Tag` 属性为 `myFigure` 的特定图窗的句柄：

```
if h == findobj('Tag','myFigure')
    ...% h is correct figure
end
```

如果您想确定是否有其他对象是同一状态，使用 `isequal`：

```
hLine1 = line;
hLine2 = line;
isequal(hLine1,hLine2)
```

```
ans =
```

```
1
```

图形数组

图形数组可以包含任何图形对象的句柄。例如，调用 `plot` 函数返回包含五个线条对象句柄的数组：

```
y = rand(20,5);
```

```
h = plot(y)
```

```
h =
```

```
5x1 Line array:
```

```
Line  
Line  
Line  
Line  
Line
```

该数组仅包含线条对象的句柄。图形数组可以包含不止一种类型的图形对象。也就是说，图形数组可以是异类的。

例如，可以将图窗、坐标区和线条对象的句柄串联到一个数组 `harray`：

```
hf = figure;
```

```
ha = axes;
```

```
hl = plot(1:10);
```

```
harray = [hf,ha,hl]
```

```
harray =
```

```
1x3 graphics array:
```

```
Figure Axes Line
```

图形数组遵循与所有 MATLAB 数组一样的规则。例如，数组元素维度必须相同。在此代码中，绘图返回一个 5×1 线条数组：

```
hf = figure;
```

```
ha = axes;
```

```
hl = plot(rand(5));
```

```
harray = [hf,ha,hl];
```

```
Error using horzcat
```

```
Dimensions of matrices being concatenated are not consistent.
```

要构造数组，必须转置 `hl`：

```
harray = [hf,ha,hl']
```

```
harray =
```

```
1x7 graphics array:
```

```
Figure Axes Line Line Line Line Line
```

不能将数值数据连结到对象句柄，除非是由图窗 `Number` 属性指定的唯一标识符。例如，如果有一个图窗的 `Number` 属性设置为 1，您可以用此数字引用该图窗：

```
figure(1)
aHandle = axes;
[aHandle,1]

ans =
1x2 graphics array:

Axes    Figure
```

数组构造的同样规则也适用于索引分配。例如，您可以使用 **for** 循环构造一个句柄数组：

```
harray = gobjects(1,7);
hf = figure;
ha = axes;
hl = plot(rand(5));
harray(1) = hf;
harray(2) = ha;
for k = 1:length(hl)
    harray(k+2) = hl(k);
end
```


图形对象回调

- “回调 - 对用户操作的程序化响应” (第 20-2 页)
- “回调定义” (第 20-3 页)
- “按钮按下回调函数” (第 20-5 页)
- “定义上下文菜单” (第 20-6 页)
- “定义对象创建回调” (第 20-7 页)
- “定义对象删除回调” (第 20-8 页)
- “捕获鼠标点击” (第 20-9 页)
- “将鼠标点击传递给组的父级” (第 20-12 页)
- “将鼠标点击传递给被遮盖对象” (第 20-14 页)

回调 - 对用户操作的程序化响应

本节内容

“什么是回调？” (第 20-2 页)

“窗口回调” (第 20-2 页)

什么是回调？

回调是对某些预定义用户操作，如点击图形对象或关闭图窗窗口，做出响应的函数。通过将函数分配给该用户操作的回调属性，从而将回调函数与特定的用户操作关联。

所有图形对象都具有以下属性，您可以用它们来定义回调函数：

- **ButtonDownFcn** - 当光标悬停在对象上或在对象几个像素以内按下鼠标左键时执行。
- **CreateFcn** - 在对象创建过程中，MATLAB 设置所有属性之后执行
- **DeleteFcn** - 就在 MATLAB 删除对象之前执行

注意 当您调用绘图函数（如 **plot** 或 **bar**）时，MATLAB 会创建新图形对象并重置大部分图窗和坐标区属性。因此，您对图形对象定义的回调函数会被 MATLAB 移除。要避免这一问题，请参阅 “将回调定义为默认值” (第 20-4 页)。

窗口回调

图窗还有一些属性，它们针对特定的用户操作执行回调。这些额外属性在 MATLAB Online 中不可用。

- **CloseRequestFcn** - 请求关闭图窗（由 **close** 命令、窗口管理器菜单或退出 MATLAB 发出）时执行。
- **KeyPressFcn** - 当光标位于图窗窗口中并按下某个键时执行。
- **SizeChangedFcn** - 当您重新调整图窗窗口大小时执行。
- **WindowButtonDownFcn** - 当光标悬停在图窗背景、禁用的用户界面控件或坐标区背景，按下鼠标按键时执行。
- **WindowButtonMotionFcn** - 当您在图窗窗口移动鼠标（但不在菜单或标题栏上）时执行。
- **WindowButtonUpFcn** - 当您在图窗按下鼠标按键然后松开鼠标按键时执行。

回调定义

本节内容

- “指定回调的方法” (第 20-3 页)
- “回调函数语法” (第 20-3 页)
- “相关信息” (第 20-4 页)
- “将回调定义为默认值” (第 20-4 页)

指定回调的方法

要使用回调属性，将回调代码赋予该属性。使用以下技术之一：

- 引用函数执行的函数句柄。
- 包含函数句柄和其他参数的元胞数组
- 计算为有效 MATLAB 表达式的字符向量。MATLAB 在基础工作区中计算该字符向量。

不建议将回调定义为字符向量。将函数指定为函数句柄的用法可以让 MATLAB 为回调函数提供重要信息。

有关详细信息，请参阅 “回调函数语法” (第 20-3 页)。

回调函数语法

图形回调函数必须至少接受两个输入参数：

- 正在执行其回调的图形对象的句柄。在您的回调函数中使用该句柄以引用回调对象。
- 事件数据结构，它对于某些回调可能是空的，或包含该对象的属性说明中的具体信息。

无论何时执行作为特定触发操作结果的回调函数，MATLAB 都会调用回调函数并传递两个参数给该函数。

例如，为 `plot` 函数创建的线条定义一个名为 `lineCallback` 的回调函数。通过使用 MATLAB 路径上的 `lineCallback` 函数，使用 `@` 运算符将函数句柄分配给 `plot` 创建的每个线条的 `ButtonDownFcn` 属性。

```
plot(x,y,'ButtonUpFcn',@lineCallback)
```

定义接受两个输入参数的回调。使用第一个参数引用正在执行其回调的特定线条。使用该参数设置线条的 `Color` 属性：

```
function lineCallback(src,~)
    src.Color = 'red';
end
```

第二个参数对 `ButtonDownFcn` 回调为空。`~` 字符表示该参数未使用。

传递额外输入参数

要定义回调函数的额外输入参数，将参数添加到函数定义，同时保持默认参数和其他参数的正确顺序：

```
function lineCallback(src,evt,arg1,arg2)
    src.Color = 'red';
    src.LineStyle = arg1;
```

```
src.Marker = arg2;
end
```

将包含函数句柄和额外参数的元胞数组分配给属性：

```
plot(x,y,'ButtonDownFcn',{@lineCallback,'_','*'})
```

您可以使用匿名函数传递额外参数。例如：

```
plot(x,y,'ButtonDownFcn',...
@(src eventdata)lineCallback(src,eventdata,'_','*'))
```

相关信息

有关使用匿名函数的详细信息，请参阅“[匿名函数](#)”。

有关将类方法用作回调的详细信息，请参阅“[Class Methods for Graphics Callbacks](#)”。

有关 MATLAB 如何解决多回调执行，请参阅对象定义回调的 **BusyAction** 和 **Interruptible** 属性。

将回调定义为默认值

您可以将回调分配给特定对象的属性或定义该类型所有对象的默认回调。

要定义所有线型对象的 **ButtonDownFcn**，在根级别设置默认值。

- 使用 **groot** 函数指定对象层次结构的根级别。
- 定义 MATLAB 路径上的回调函数。
- 将引用该函数的函数句柄分配给 **defaultLineButtonDownFcn**。

```
set(groot,'defaultLineButtonDownFcn',@lineCallback)
```

默认值仍然分配给 MATLAB 会话。您可以在 **startup.m** 文件中进行默认值分配。

按钮按下回调函数

本节内容

- “何时使用按钮按下回调”（第 20-5 页）
- “如何定义按钮按下回调”（第 20-5 页）

何时使用按钮按下回调

当用户在分配回调的图形对象上点击鼠标左键时会执行按钮按下回调。按钮按下回调为用户提供了与对象交互的简单方式，无需对额外的用户界面对象，如普通按钮或弹出式菜单，进行编程。

当您想要用户能够执行以下操作时，对按钮按下回调进行编程：

- 通过左键点击图形对象执行单个操作
- 结合使用修改键和左键点击选择对图形对象执行的不同操作

如何定义按钮按下回调

- 创建用户左键点击图形对象时 MATLAB 执行的回调函数。
- 分配将回调函数引用到对象 **ButtonDownFcn** 属性的函数句柄。

```
...'ButtonDownFcn',@callbackFcn
```

定义回调函数

在此例中，回调函数名为 **lineCallback**。当您将函数句柄分配给 **ButtonDownFcn** 属性时，此函数必须在 MATLAB **path** 上。

回调函数中用到的值包括：

- **src** - 用户点击的线条对象的句柄。MATLAB 传递此句柄。
- **src.Color** - 线条对象 **Color** 属性。

```
function lineCallback(src,~)
    src.Color = rand(1,3);
end
```

使用回调

以下是对绘图函数的调用，该函数创建线图并为创建的每个线条定义按钮按下回调。

```
plot(rand(1,5),'ButtonDownFcn',@lineCallback)
```

要使用回调，创建绘图并左键点击线条。

定义上下文菜单

此示例演示如何定义上下文菜单。

何时使用上下文菜单

当用户右键点击分配了上下文菜单的图形对象时会显示上下文菜单。上下文菜单可让您为用户提供与图形对象交互的选择。

当您想要用户能执行以下操作时，对上下文菜单编程：

- 选择右键点击图形对象的具体选项。
- 通过菜单标签指示每个选项是什么。
- 生成特定结果，而无需知道组合键。

如何定义上下文菜单

- 通过带输出参数调用 `uicontextmenu` 函数，创建一个 `ContextMenu` 对象。
- 使用 `uimenu` 创建每个菜单项。
- 为上下文菜单中的每个菜单项定义回调。
- 使各个菜单项成为上下文菜单的父对象，并分配各个回调。
- 将 `ContextMenu` 对象分配给您为其定义上下文菜单的对象的 `ContextMenu` 属性。

```
function cmHandle = defineCM
    cmHandle = uicontextmenu;
    uimenu(cmHandle,'Label','Wider','Callback',@increaseLW);
    uimenu(cmHandle,'Label','Inspect','Callback',@inspectLine);
end
function increaseLW(~,~)
% Increase line width
    h = gco;
    orgLW = h.LineWidth;
    h.LineWidth = orgLW+1;
end
function inspectLine(~,~)
% Open the property inspector
    h = gco;
    inspect(h)
end
```

`defineCM` 函数将句柄返回给它创建的上下文菜单。将此句柄分配给线条对象的 `ContextMenu` 属性，因为这些线条是 `plot` 函数创建的：

```
plot(rand(1,5),'ContextMenu',defineCM)
```

根据您具体需求使用类似的编程模式。

定义对象创建回调

此示例演示如何定义对象创建回调。

定义指定线条对象 `LineWidth` 和 `Marker` 属性的对象创建回调。

```
function lineCreate(src,~)
    src.LineWidth = 2;
    src.Marker = 'o';
end
```

使用线条 `CreateFcn` 属性将该函数指定为默认线条创建回调：

```
set(groot,'defaultLineCreateFcn',@lineCreate)
```

`groot` 函数指定图形对象层次结构的根级别。因此，在给定 MATLAB 会话中创建的所有线条都获得此回调。创建线条的所有绘图函数都使用这些默认值。

对象创建回调在 MATLAB 创建对象并设置所有属性值后直接执行。因此，创建回调会覆盖绘图函数中指定的属性名称/值对组。例如：

```
set(groot,'defaultLineCreateFcn',@lineCreate)
h = plot(1:10,'LineWidth',5,'Marker','none')
```

创建回调在绘图函数完全执行后再执行。生成线条的 `LineWidth` 和 `Marker` 属性值是在创建回调中指定的值：

```
h =
```

Line with properties:

```
Color: [0 0 1]
LineStyle: '-'
LineWidth: 2
Marker: 'o'
MarkerSize: 6
MarkerFaceColor: 'none'
XData: [1 2 3 4 5 6 7 8 9 10]
YData: [1 2 3 4 5 6 7 8 9 10]
ZData: []
```

相关信息

有关定义回调函数的详细信息，请参阅“回调定义”（第 20-3 页）

定义对象删除回调

您可以创建在删除对象时执行代码的对象删除回调。

例如，为图窗创建一个对象删除回调，以便在删除此图窗时显示一个对话框来询问是否要删除所有图窗。将以下代码复制到一个新的函数文件，然后将其以 `figDelete.m` 的名称保存在当前文件夹或 MATLAB 搜索路径上的某个文件夹中。

```
function figDelete(~,~)
yn = questdlg('Delete all figures?',...
'Figure Menu',...
'Yes','No','No');
switch yn
case 'Yes'
    allfigs = findobj(get(groot,'Children'),'Type','figure');
    set(allfigs,'DeleteFcn',[]);
    delete(allfigs)
case 'No'
    return
end
end
```

然后创建两个图窗，并将 `figDelete` 函数赋给 `DeleteFcn` 属性。删除其中一个图窗，并选择所出现的对话框中的一个选项。

```
figure('DeleteFcn',@figDelete)
figure('DeleteFcn',@figDelete)
```

捕获鼠标点击

本节内容

- “控制对鼠标点击进行响应的属性”（第 20-9 页）
- “组合使用 PickablePart/HitTest 值”（第 20-9 页）
- “沿层次结构向上传递鼠标点击”（第 20-10 页）

控制对鼠标点击进行响应的属性

有两个属性可以确定对象是否及如何响应鼠标点击：

- **PickableParts** - 确定对象是否捕获鼠标点击
- **HitTest** - 确定对象是否响应鼠标点击或将其传递给最近的前代。

对象在层次结构中传递点击，直到有对象对其响应。

对鼠标点击的响应编程

当对象捕获并响应鼠标点击时，该对象执行以下操作：

- 执行鼠标按下函数以响应鼠标左击操作 - 如果对象为 **ButtonDownFcn** 属性定义了回调，那么 MATLAB 执行此回调。
- 显示上下文菜单以响应鼠标右键点击 - 如果对象使用 **ContextMenu** 属性定义了上下文菜单，那么 MATLAB 调用上下文菜单。

注意 图窗没有 **PickableParts** 属性。图窗执行按钮回调函数，无论其 **HitTest** 属性设置如何。

注意 如果坐标区的 **PickableParts** 属性设置为 '**none**'，那么坐标区的子对象无法捕获鼠标点击。这种情况下，所有鼠标点击都会由图窗捕获。

组合使用 PickablePart/HitTest 值

使用 **PickableParts** 和 **HitTest** 属性实现以下行为：

- 被点击的对象捕获鼠标点击，并以按钮按下回调或上下文菜单响应。
- 被点击的对象捕获鼠标点击，并将鼠标点击传递给它的一个前代，该前代以按钮按下回调或上下文菜单响应。
- 被点击的对象未捕获鼠标点击。鼠标点击可能由被点击对象背后的对象捕获。

下表总结了基于属性值的鼠标点击响应。

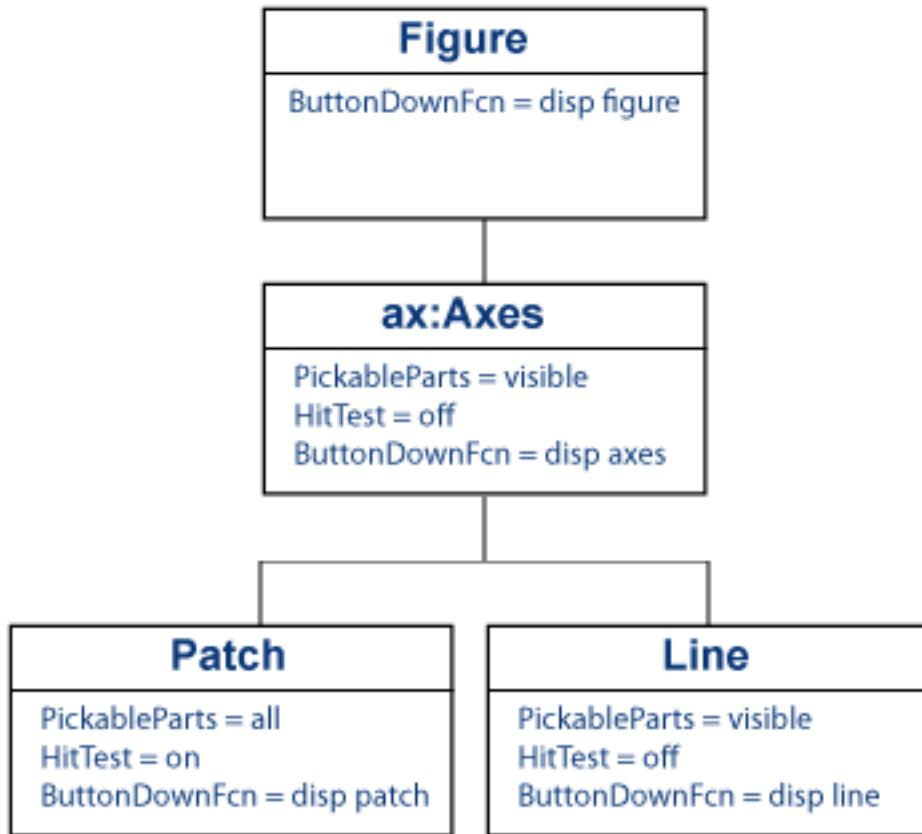
坐标区 PickableParts	PickableParts	HitTest	鼠标点击结果
visible/all	visible (默认值)	on (默认值)	点击对象可见部分会执行按钮按下回调或调用上下文菜单

坐标区 PickableParts	PickableParts	HitTest	鼠标点击结果
visible/all	all	on	点击对象任何部分（即使不可见），也会使该对象成为当前对象并执行按钮按下回调或调用上下文菜单
visible/all/none	none	on/off	点击对象不会使其成为当前对象，而且不会执行按钮按下回调或调用上下文菜单
none	visible/all/none	on/off	点击任何坐标区的子对象都不会执行按钮按下回调或调用上下文菜单

MATLAB 使用每个对象的 `Parent` 属性查找前代，直到找到合适的前代或达到图窗。

沿层次结构向上传递鼠标点击

考虑以下对象层次结构及其 `PickableParts` 和 `HitTest` 属性设置。



以下代码创建层次结构：

```

function pickHit
f = figure;
ax = axes;
p = patch(rand(1,3),rand(1,3),'g');
l = line([1 0],[0 1]);
  
```

```
set(f,'ButtonDownFcn',@(~,~)disp('figure'),...
    'HitTest','off')
set(ax,'ButtonDownFcn',@(~,~)disp('axes'),...
    'HitTest','off')
set(p,'ButtonDownFcn',@(~,~)disp('patch'),...
    'PickableParts','all','FaceColor','none')
set(l,'ButtonDownFcn',@(~,~)disp('line'),...
    'HitTest','off')
end
```

点击线条

左键点击线条：

- 线条成为当前对象，但无法执行其 **ButtonDownFcn** 回调，因为其 **HitTest** 属性是 **off**。
- 该线条将点击传递给最近的前代（父坐标区），但坐标区无法执行其 **ButtonDownFcn** 回调，因此坐标区将点击传递给图窗。
- 图窗可执行其回调，因此 MATLAB 在命令行窗口中显示 **figure**。

点击补片

补片 **FaceColor** 为 **none**。尽管如此，补片的 **PickableParts** 为 **all**，因此可以通过点击空表面和边缘选取补片。

补片 **HitTest** 属性为 **on**，因此该补片成为当前对象。当该补片成为当前对象时，它执行其按钮按下回调。

将鼠标点击传递给组的父级

此示例演示一组对象如何将鼠标点击传递给父对象，该对象会作用于组中所有对象。

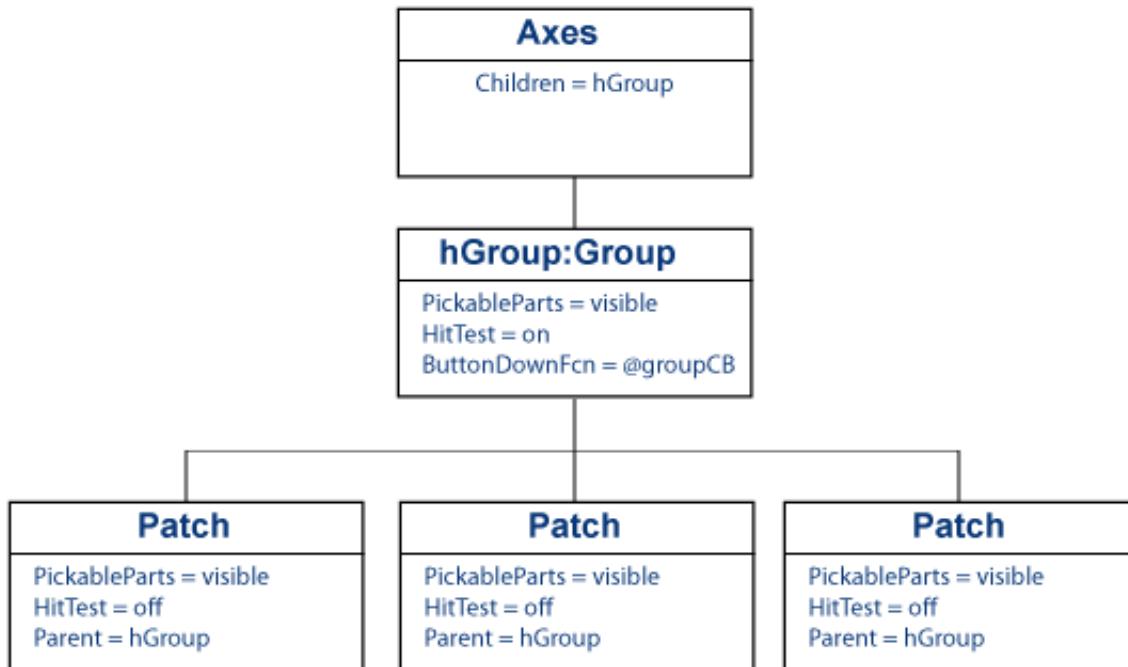
目标和设计

假设您想要对一组对象中任何一个成员点击鼠标，都会执行影响组中所有对象的鼠标按下回调。

- 定义将添加到组中的图形对象。
- 将 **hggroup** 对象分配为图形对象的父级。
- 定义一个任何对象被点击时都会执行的函数。将其函数句柄分配给 **hggroup** 对象的 **ButtonDownFcn** 属性。
- 将组中所有对象的 **HitTest** 属性设置为 **off**，从而让鼠标点击传递到对象父级。

对象层次结构和关键属性

本例使用以下对象层次结构。



MATLAB 代码

创建含有两个函数的文件：

- **pickPatch** - 创建图形对象的主函数。
- **groupCB** - **hggroup** 回调的局部函数。

pickPatch 函数创建三个补片对象，并使 **hggroup** 对象成为其父级。设置每个补片的 **HitTest** 属性，从而将鼠标点击指向其父级。

```
function pickPatch
    figure
    x = [0 1 2];
    y = [0 1 0];
    hGroup = hggroup('ButtonDownFcn',@groupCB);
    patch(x,y,'b',...
        'Parent',hGroup,...
        'HitTest','off')
    patch(x+2,y,'b',...
        'Parent',hGroup,...
        'HitTest','off')
    patch(x+3,y,'b',...
        'Parent',hGroup,...
        'HitTest','off')
end
```

groupCB 回调作用于 **hggroup** 中包含的所有对象。**groupCB** 函数使用传递给回调的 回调源参数 (**src**) 以获得补片对象的句柄。

使用回调源参数 (**hggroup** 对象句柄) 则无需创建全局数据将额外参数传递给回调。

在任何补片上左键点击都会将所有三个补片的面颜色改成随机 RGB 颜色值。

```
function groupCB(src,~)
    s = src.Children;
    set(s,'FaceColor',rand(1,3))
end
```

有关回调函数的详细信息, 请参阅 “回调定义” (第 20-3 页)

将鼠标点击传递给被遮盖对象

本例演示如何将鼠标点击传递给被遮盖对象。

将图形对象的 **PickableParts** 属性设置为 **none** 以防止该对象捕获鼠标点击。此示例：

- 为坐标区定义了一个上下文菜单，它使用值 **on** 或 **off** 调用 **hold**。
- 创建一个图形，该图形中的数据对象都不能捕获鼠标点击，从而让所有的右键点击都传递到坐标区并调用上下文菜单。

axesHoldCM 函数定义了上下文菜单并返回其句柄。

```
function cmHandle = axesHoldCM
    cmHandle = uicontextmenu;
    uimenu(cmHandle,'Label','hold on','Callback',@holdOn);
    uimenu(cmHandle,'Label','hold off','Callback',@holdOff);
end
function holdOn(~,~)
    fig = gcbf;
    ax = fig.CurrentAxes;
    hold(ax,'on')
end
function holdOff(~,~)
    fig = gcbf;
    ax = fig.CurrentAxes;
    hold(ax,'off')
end
```

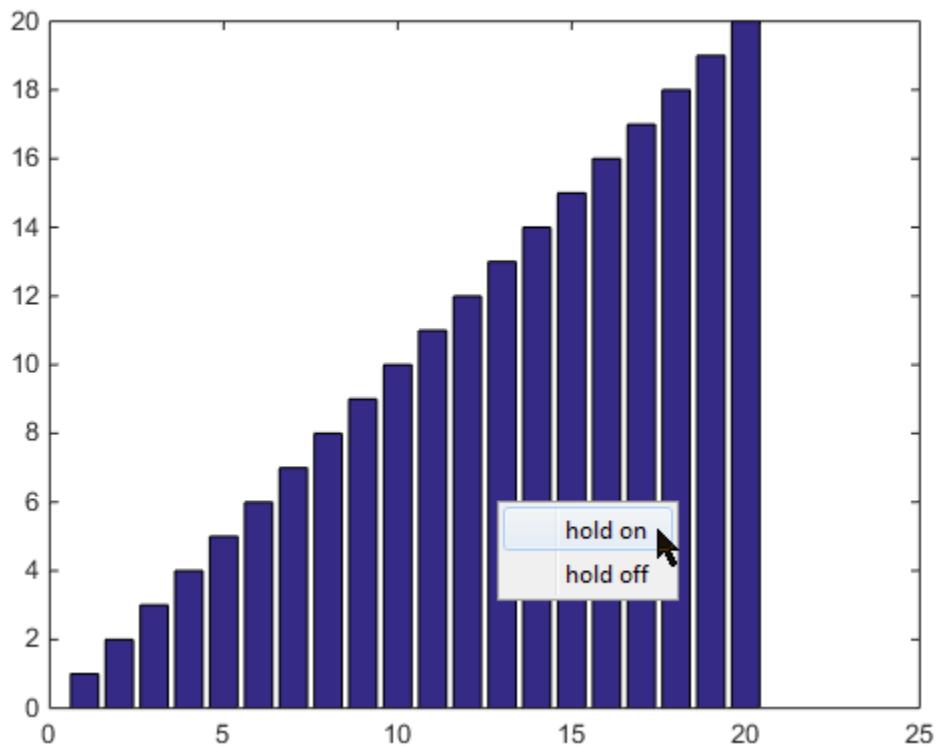
创建一个条形图并设置条形图对象的 **PickableParts** 属性：

```
bar(1:20,'PickableParts','none')
```

创建当前坐标区的上下文菜单：

```
ax = gca;
ax.ContextMenu = axesHoldCM
```

右键点击图形中的条形并显示坐标区上下文菜单：



组对象

- “对象组” (第 21-2 页)
- “创建对象组” (第 21-3 页)
- “hgtransform 支持的变换” (第 21-5 页)
- “绕任意轴旋转” (第 21-9 页)
- “嵌套变换，执行复杂移动” (第 21-12 页)

对象组

组对象是不可见的图形对象容器。使用组对象组成一个对象集合，这些对象在某些方面行为如同一个对象。当您设置组对象的属性时，结果会应用到包含在组中的所有对象。

例如，您可以将整个组设定为可见或不可见，当点击其中一个时选择所有对象，或应用一个变换矩阵以重新定位对象。

组对象可以包含坐标区所包含的任意对象，如线条、曲面、文本等。组对象还可以包含其他组对象。组对象父级通常为坐标区对象或其他组对象。

有两种类型组对象：

- **组** - 当您想要创建一组对象，或根据组中单个对象的事件控制整个组的可见性或可选择性时使用。使用 **hggroup** 函数创建组对象。
- **变换** - 当您想要变换一组对象时使用。变换包括旋转、转换和缩放。有关示例，请参阅“嵌套变换，执行复杂移动”（第 21-12 页）。使用 **hgtransform** 函数创建变换对象。

组对象与变换对象的区别在于变换对象可以将变换矩阵（通过其 **Matrix** 属性）应用到所有子级对象。

创建对象组

本节内容

- “父级设定”（第 21-3 页）
- “组的子对象的可见性和选中属性”（第 21-4 页）

通过将一个组或变换对象作为对象父级来创建对象组。例如，调用 **hggroup** 创建组对象并保存其句柄。将该组对象分配为后续创建对象的父级：

```
hg = hggroup;
plot(rand(5),'Parent',hg)
text(3,0.5,'Random lines','Parent',hg)
```

将该组的可见性设置为关闭，可使其所包含的线条和文本对象都不可见。

```
hg.Visible = 'off';
```

您可以有选择地将对象添加到组中。例如，以下对 **bar** 函数的调用返回五个条形对象的句柄：

```
hb = bar(randn(5))
```

```
hb =
```

```
1x5 Bar array:
```

```
Bar Bar Bar Bar Bar
```

将该组作为第三个、第四个、第五个条形对象的父级：

```
hg = hggroup;
set(hb(3:5),'Parent',hg)
```

组对象可以是任意数目坐标区子对象（包括其他组对象）的父级。有关示例，请参阅“绕任意轴旋转”（第 21-9 页）和“嵌套变换，执行复杂移动”（第 21-12 页）。

父级设定

绘图函数可以在生成其图形之前清除坐标区。但如果在绘图函数中将组或变换分配为 **Parent**，那么此组或变换对象将被清除。

例如：

```
hg = hggroup;
hb = bar(randn(5));
set(hb,'Parent',hg)
```

```
Error using matlab.graphics.chart.primitive.Bar/set
Cannot set property to a deleted object
```

bar 函数清除坐标区。但如果将 **Parent** 属性设置为 **bar** 函数参数中的名称/值对组，那么条形函数不会删除该组：

```
hg = hggroup;
hb = bar(randn(5),'Parent',hg);
```

组的子对象的可见性和选中属性

设置组或变换对象的 **Visible** 属性可以控制组中的对象可见还是不可见。但改变组对象的 **Visible** 属性的状态不会改变单个对象此属性的状态。单个对象的 **Visible** 属性值是预先保留的。

例如，如果组的 **Visible** 属性设置为关闭，以后再设置为打开，则只有初始为可见的对象才会显示。

相同的行为也适用于 **Selected** 和 **SelectionHighlight** 属性。组对象或变换对象的子级显示在其所属对象属性的状态，而不会实际改变其属性值。

hgtransform 支持的变换

本节内容

- “变换对象” (第 21-5 页)
- “旋转” (第 21-5 页)
- “转换” (第 21-5 页)
- “缩放” (第 21-6 页)
- “默认变换” (第 21-6 页)
- “不允许的变换：透视” (第 21-6 页)
- “不允许的变换：剪切” (第 21-6 页)
- “绝对变换与相对变换” (第 21-7 页)
- “将变换合并到一个矩阵” (第 21-7 页)
- “撤消变换操作” (第 21-7 页)

变换对象

变换对象的 **Matrix** 属性将变换同时应用到对象的所有子对象。变换包括旋转、转换和缩放。定义一个 4×4 变换矩阵的变换。

创建变换矩阵

makehgform 函数简化了执行旋转、转换和缩放来构造矩阵的过程。有关创建使用 **makehgform** 创建变换矩阵的详细信息，请参阅 “嵌套变换，执行复杂移动” (第 21-12 页)。

旋转

旋转变换遵守右手定则 - 将对象绕 x 、 y 或 z 轴旋转，以正角度按逆时针旋转，同时沿着各轴指向原点。如果旋转角度是 θ ，以下矩阵定义了绕 x 轴旋转 θ 。

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \theta_x & -\sin \theta_x & 0 \\ 0 & \sin \theta_x & \cos \theta_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

要定义任意轴旋转的变换矩阵，使用 **makehgform** 函数。

转换

转换变换将对象相对于当前位置进行移动。转换可指定为采用数据空间单位的距离 t_x 、 t_y 和 t_z 。以下矩阵显示变换矩阵中这些元素的位置。

$$\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

缩放

缩放变换会改变对象的大小。指定缩放因子 s_x 、 s_y 和 s_z 并构造以下矩阵。

$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

不能使用小于或等于零的缩放因子。

默认变换

默认变换是单位矩阵，您可以使用 `eye` 函数来创建它。以下是单位矩阵。

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

请参阅“撤消变换操作”（第 21-7 页）。

不允许的变换：透视

透视变换会改变您观察对象的距离。以下矩阵为透视变换矩阵示例，这在 MATLAB 图形中是不允许的。

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & p_x & 0 \end{bmatrix}$$

在本例中， p_x 是透视因子。

不允许的变换：剪切

剪切变换保持给定直线（或三维坐标中的平面）上所有点固定，同时将其他所有平行于直线（或平面）的点根据其与直线（或平面）的垂直距离成比例移动。以下矩阵为剪切变换矩阵示例，这在 `hgtransform` 中不允许。

$$\begin{bmatrix} 1 & s_x & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

在此例中， s_x 是剪切因子，可以替换单位矩阵中的任意零元素。

绝对变换与相对变换

变换以绝对形式指定，不相对于当前变换。例如，如果首先应用将变换对象在 x 方向上转换 5 个单位的变换，然后应用将变换对象在 y 方向上转换 4 个单位的变换，那么对象最后的位置在 y 方向上偏离原始位置 4 个单位。

如果您将变换进行累积，您必须将各个变换串联到一个矩阵中。请参阅“将变换合并到一个矩阵”（第 21-7 页）。

将变换合并到一个矩阵

通过将单个矩阵串联（或相乘）并把 **Matrix** 属性设置为结果，来将各种变换合并到一个矩阵中，这样往往更加高效。矩阵乘法不适用交换律，因此矩阵相乘的顺序会影响结果。

例如，您想要先执行缩放，再转换，最后旋转。假设 **R**、**T** 和 **S** 是各个变换矩阵，按以下顺序相乘：

```
C = R*T*S % operations are performed from right to left
```

S 是缩放矩阵，**T** 是转换矩阵，**R** 是旋转矩阵，**C** 是三种操作的复合矩阵。然后将变换对象的 **Matrix** 属性设置为 **C**：

```
hg = hgtransform('Matrix',C);
```

将变换乘以单位矩阵

以下两组语句并不等价。第一组：

```
hg.Matrix = C;
hg.Matrix = eye(4);
```

结果是消除变换 **C**。第二组：

```
I = eye(4);
C = I*R*T*S;
hg.Matrix = C;
```

应用变换 **C**。将单位矩阵串联到其他矩阵对复合矩阵无效。

撤消变换操作

由于变换操作是以绝对形式指定（不相对于当前变换），您可以通过将当前变换设置为单位矩阵来撤消一系列变换。例如：

```
hg = hgtransform('Matrix',C);
...
hg.Matrix = eye(4);
```

将变换对象 **hg** 所包含的对象返回到其方向，然后再应用变换 **C**。

有关单位矩阵的详细信息，请参阅 **eye** 函数。

另请参阅

[hgtransform](#) | [makehgtransform](#) | [eye](#)

详细信息

- “嵌套变换，执行复杂移动”（第 21-12 页）
- “撤消变换操作”（第 21-7 页）
- “将变换合并到一个矩阵”（第 21-7 页）

绕任意轴旋转

此示例演示如何绕任意轴旋转对象。

旋转前转换到原点

旋转是绕原点进行的。因此，您需要执行转换，以将旋转的目标轴暂时定在原点。应用旋转变换矩阵后，将对象转换回原始位置。

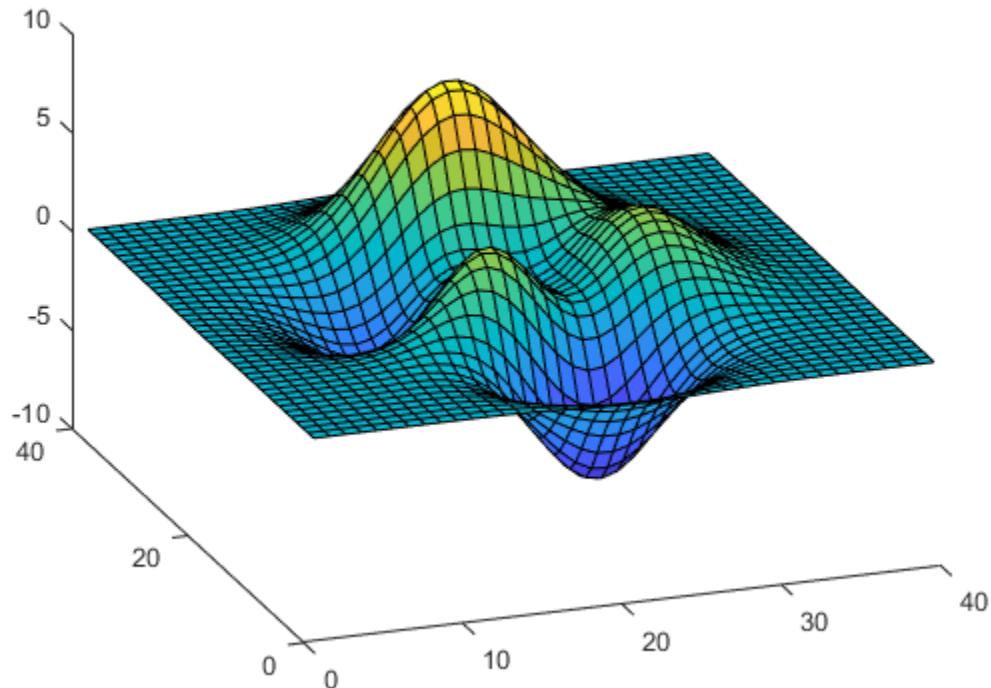
旋转曲面

此示例演示如何让曲面绕 y 轴旋转。

创建曲面和变换

将变换对象设为曲面的父级。

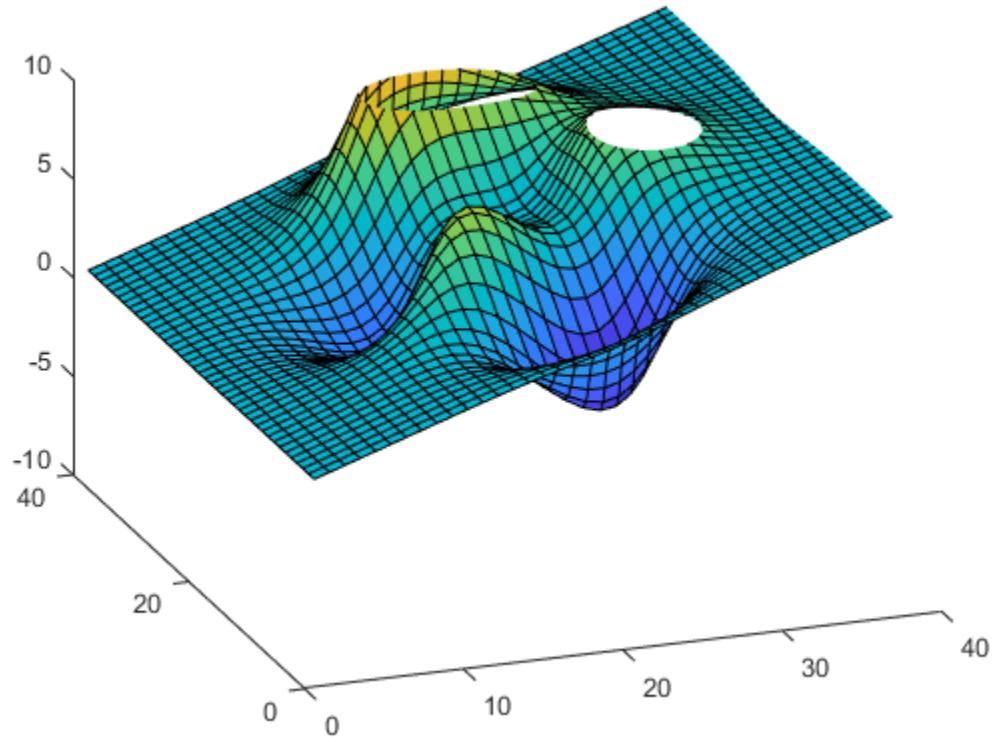
```
t = hgtransform;
surf(peaks(40),'Parent',t)
view(-20,30)
axis manual
```



创建变换

设置一个 y 轴旋转矩阵，将曲面旋转 -15 度。

```
ry_angle = -15*pi/180;
Ry = makehgtform('yrotate',ry_angle);
t.Matrix = Ry;
```



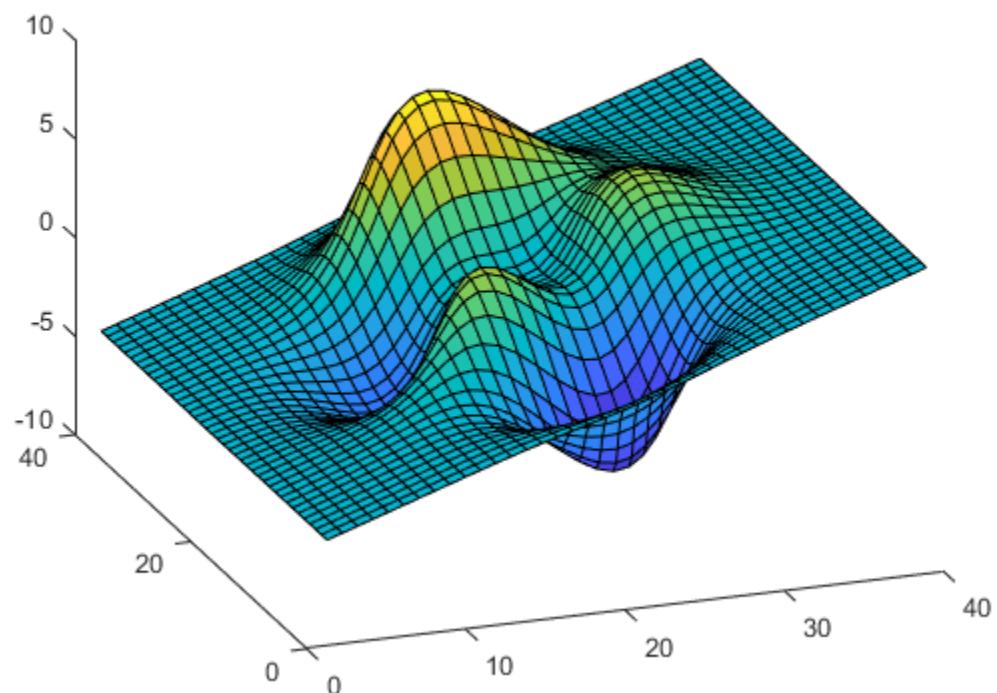
曲面绕 y 轴旋转 -15 度，经过原点。

转换曲面并旋转

现在绕 y 轴旋转曲面，经过点 $x = 20$ 。

创建两个转换矩阵，一个将曲面沿 x 轴转换 -20 单位，另一个往回转换 20 单位。用旋转矩阵按照正确的顺序串联两个转换矩阵，并设置该变换。

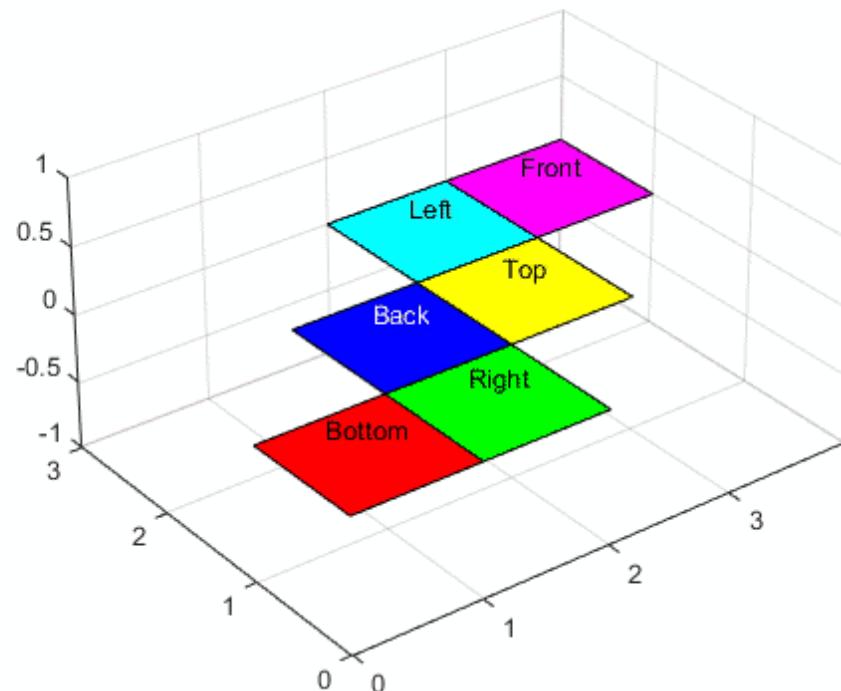
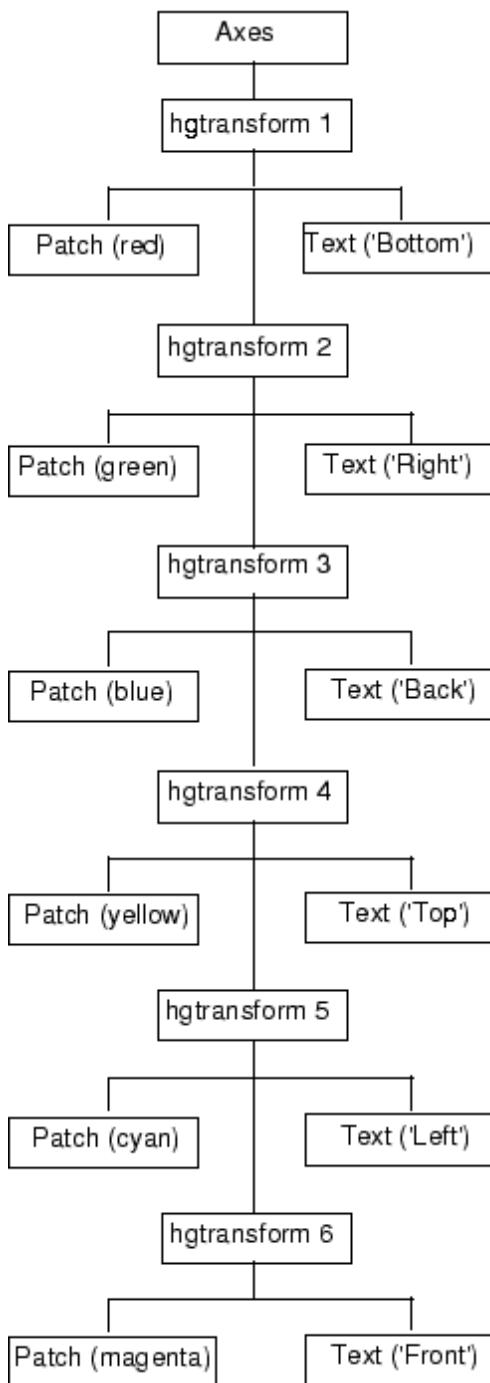
```
Tx1 = makehgtform('translate',[-20 0 0]);
Tx2 = makehgtform('translate',[20 0 0]);
t.Matrix = Tx2*Ry*Tx1;
```



嵌套变换，执行复杂移动

以下示例创建变换对象的嵌套层次结构，然后按顺序对这些对象进行变换，用六个正方形创建一个立方体。此示例演示如何使变换对象作为其他变换对象的父级以创建层次结构，以及变换层次结构的成员如何影响附属成员。

这里展示了层次结构。



`transform_foldbox` 函数实现变换层次结构。`doUpdate` 函数显现每一步。将这两个函数放入一个名为 `transform_foldbox.m` 的文件并执行 `transform_foldbox`。

```

function transform_foldbox
    % Create six square and fold
    % them into a cube

    figure

    % Set axis limits and view
    axes('Projection','perspective',...
        'XLim',[0 4],...
        'YLim',[0 4],...
        'ZLim',[0 3])
    view(3); axis equal; grid on

    % Create a hierarchy of transform objects
    t(1) = hgtransform;
    t(2) = hgtransform('parent',t(1));
    t(3) = hgtransform('parent',t(2));
    t(4) = hgtransform('parent',t(3));
    t(5) = hgtransform('parent',t(4));
    t(6) = hgtransform('parent',t(5));

    % Patch data
    X = [0 0 1 1];
    Y = [0 1 1 0];
    Z = [0 0 0 0];

    % Text data
    Xtext = .5;
    Ytext = .5;
    Ztext = .15;

    % Corresponding pairs of objects (patch and text)
    % are parented into the object hierarchy
    p(1) = patch('FaceColor','red','Parent',t(1));
    txt(1) = text('String','Bottom','Parent',t(1));
    p(2) = patch('FaceColor','green','Parent',t(2));
    txt(2) = text('String','Right','Parent',t(2));
    p(3) = patch('FaceColor','blue','Parent',t(3));
    txt(3) = text('String','Back','Color','white','Parent',t(3));
    p(4) = patch('FaceColor','yellow','Parent',t(4));
    txt(4) = text('String','Top','Parent',t(4));
    p(5) = patch('FaceColor','cyan','Parent',t(5));
    txt(5) = text('String','Left','Parent',t(5));
    p(6) = patch('FaceColor','magenta','Parent',t(6));
    txt(6) = text('String','Front','Parent',t(6));

    % All the patch objects use the same x, y, and z data
    set(p,'XData',X,'YData',Y,'ZData',Z)

    % Set the position and alignment of the text objects
    set(txt,'Position',[Xtext Ytext Ztext],...
        'HorizontalAlignment','center',...
        'VerticalAlignment','middle')

    % Display the objects in their current location

```

```
doUpdate(1)

% Set up initial translation transforms
% Translate 1 unit in x
Tx = makehgtform('translate',[1 0 0]);
% Translate 1 unit in y
Ty = makehgtform('translate',[0 1 0]);

% Translate the unit squares to the desired locations
% The drawnow and pause commands display
% the objects after each translation
set(t(2),'Matrix',Tx);
doUpdate(1)
set(t(3),'Matrix',Ty);
doUpdate(1)
set(t(4),'Matrix',Tx);
doUpdate(1)
set(t(5),'Matrix',Ty);
doUpdate(1)
set(t(6),'Matrix',Tx);
doUpdate(1)

% Specify rotation angle (pi/2 radians = 90 degrees)
fold = pi/2;

% Rotate -y, translate x
Ry = makehgtform('yrotate',-fold);
RyTx = Tx*Ry;

% Rotate x, translate y
Rx = makehgtform('xrotate',fold);
RxTy = Ty*Rx;

% Set the transforms
% Draw after each group transform and pause
set(t(6),'Matrix',RyTx);
doUpdate(1)
set(t(5),'Matrix',RxTy);
doUpdate(1)
set(t(4),'Matrix',RyTx);
doUpdate(1)
set(t(3),'Matrix',RxTy);
doUpdate(1)
set(t(2),'Matrix',RyTx);
doUpdate(1)
end

function doUpdate(delay)
drawnow
pause(delay)
end
```

控制图形输出

- “控制图形显示” (第 22-2 页)
- “为绘图准备图窗和坐标区” (第 22-4 页)
- “使用 newplot 控制绘图” (第 22-7 页)
- “对保留状态进行响应” (第 22-9 页)
- “防止对图窗和坐标区访问” (第 22-11 页)

控制图形显示

本节内容

- “您可以控制什么”（第 22-2 页）
- “以特定图窗和坐标区为目标”（第 22-2 页）

您可以控制什么

MATLAB 允许一个会话中同时打开多个图窗窗口。您可以控制 MATLAB 使用哪些图窗和哪些坐标区来显示绘图函数的结果。您还可以控制 MATLAB 清除和重置目标图窗和坐标区的属性的程度。

您可以修改 MATLAB 绘图函数行为方式，您还可以在您编写的绘图函数中实现特定的行为。

考虑如下方面：

- 您可以防止特定的图窗或坐标区成为显示图形的目标吗？
- 当您向现有图形绘制更多数据时，图形会出现什么情况？现有图形被取代，还是新图形对象添加到现有图形中？

以特定图窗和坐标区为目标

默认情况下，MATLAB 绘图函数在当前图窗和当前坐标区（分别由 `gcf` 和 `gca` 返回的对象）中显示图形。您可以通过以下方式将输出指向其他图窗和坐标区：

- 在绘图函数中显式指定目标坐标区。
- 使目标坐标区成为当前坐标区。

指定目标坐标区

假设您创建一个包含两个坐标区（分别为 `ax1` 和 `ax2`）的图窗。

```
tiledlayout(1,2)
ax1 = nexttile;
ax2 = nexttile;
```

调用 `plot`，以坐标区对象作为第一个参数：

```
plot(ax1,1:10)
```

对于不支持将坐标区作为第一个参数的绘图函数，设置 `Parent` 属性：

```
t = 0:pi/5:2*pi;
patch(sin(t),cos(t),'y','Parent',ax2)
```

使目标成为当前对象

要指定目标，可以让图窗成为当前图窗，让此图窗中的坐标区成为当前坐标区。绘图函数默认使用当前图窗和当前坐标区。如果当前图窗没有当前坐标区，那么 MATLAB 会创建一个。

如果 `fig` 是图窗句柄，则以下语句

```
figure(fig)
```

- 使 **fig** 成为当前图窗。
- 重新叠放 **fig**, 使其成为最前面的图窗。
- 如果 **fig** 不可见, 则使其可见 (将 **Visible** 属性设置为 'on') 。
- 更新图窗显示并处理所有挂起的回调。

同样的行为会应用到坐标区。如果 **ax** 是坐标区的句柄, 则以下语句

```
axes(ax)
```

- 使 **ax** 成为当前坐标区。
- 重新叠放 **ax**, 使其成为最前面的坐标区。
- 如果 **ax** 不可见, 则使其可见。
- 更新包含此坐标区的图窗并处理所有挂起的回调。

使图窗或坐标区成为当前对象, 而不改变其他状态

您可以使图窗或坐标区成为当前对象, 而不改变对象状态其他方面。将根 **CurrentFigure** 属性或图窗对象的 **CurrentAxes** 设置为您要用作目标的图窗或坐标区的句柄。

如果 **fig** 是现有图窗的句柄, 则以下语句

```
r = groot;  
r.CurrentFigure = fig;
```

使 **fig** 成为当前图窗。同样, 如果 **ax** 是坐标区对象的句柄, 则以下语句

```
fig.CurrentAxes = ax;
```

使其成为当前坐标区, 如果 **fig** 是坐标区的父图窗的句柄。

为绘图准备图窗和坐标区

本节内容

- “MATLAB 绘图函数的行为” (第 22-4 页)
- “NextPlot 属性如何控制行为” (第 22-4 页)
- “用户编写绘图函数的控制行为” (第 22-5 页)

MATLAB 绘图函数的行为

如果没有图窗和坐标区，MATLAB 绘图函数会创建新的图窗和坐标区，如果有，该函数会重用现有图窗和坐标区。在重用现有坐标区时，MATLAB

- 从坐标区中清除图形对象。
- 将大多数坐标区属性重置成默认值。
- 根据新数据计算新的坐标区范围。

当绘图函数创建图形时，该函数能够：

- 为绘图创建图窗和坐标区，并为特定图形设置必要的属性（当前图窗不存在时的默认行为）
- 重用现有图窗和坐标区，根据需要清除和重置坐标区属性（图形存在时的默认行为）
- 将新数据对象添加到现有图形，而不会重置属性（如果 `hold` 为 `on`）

`NextPlot` 图窗和坐标区属性控制 MATLAB 绘图函数的行为方式。

NextPlot 属性如何控制行为

MATLAB 绘图函数根据图窗和坐标区 `NextPlot` 属性值确定是否在绘制新图形之前添加、清除或清除并重置图窗和坐标区。低级对象创建函数不检查 `NextPlot` 属性。它们只是将新图形对象添加到当前图窗和坐标区。

下表总结了 `NextPlot` 属性的可能值。

<code>NextPlot</code>	图窗	坐标区
<code>new</code>	创建一个新图窗并将其用作当前图窗。	不是用于坐标区的选项。
<code>add</code>	在不清空或重置当前图窗的前提下添加新的图形对象。（默认值）	添加新图形对象，而不清除或重置当前坐标区。
<code>replacechildren</code>	在添加新对象前移除所有句柄未隐藏的坐标区对象。不重置图窗属性。等效于 <code>clf</code> 。	在添加新图形对象前移除所有未隐藏句柄的坐标区子对象。不要重置坐标区属性。等效于 <code>cla</code> 。
<code>replace</code>	在添加新对象前移除所有坐标区对象并将图窗属性重置为默认值。等效于 <code>clf reset</code> 。	在添加新对象前移除所有子对象并将坐标区属性重置为默认值。等效于 <code>cla reset</code> 。（默认值）

绘图函数调用 `newplot` 函数获取适合坐标区的句柄。

默认场景

考虑默认场景，图窗 NextPlot 属性是 add，坐标区 NextPlot 属性是 replace。当您调用 newplot 时，它将执行以下操作：

- 1 查看当前图窗的 NextPlot 属性值（是 add）。
- 2 确定 MATLAB 可在当前图窗中绘图而不修改图窗。如果没有当前图窗，newplot 会创建一个，但不会重新检查其 NextPlot 属性。
- 3 查看当前坐标区的 NextPlot 属性值（是 replace）、从坐标区删除所有图形对象、将所有坐标区属性（除了 Position 和 Units）重置为其默认值，并返回当前坐标区的句柄。如果没有当前坐标区，newplot 会创建一个，但不会重新检查其 NextPlot 属性。
- 4 从坐标区删除所有图形对象、将所有坐标区属性（除了 Position 和 Units）重置为其默认值并返回当前坐标区的句柄。如果没有当前坐标区，newplot 会创建一个，但不会重新检查其 NextPlot 属性。

hold 函数和 NextPlot 属性

hold 函数可方便地访问 NextPlot 属性。当您想要将对象添加到图形，而同时不移除其他对象或重置属性，可使用 hold on：

- hold on - 将图窗和坐标区 NextPlot 属性设置为 add。线图会继续循环使用 ColorOrder 和 LineStyleOrder 属性值。
- hold off - 将坐标区 NextPlot 属性设置为 replace。

使用 ishold 确定 hold 是 on 还是 off。

用户编写绘图函数的控制行为

MATLAB 提供了 newplot 函数以方便编写符合 NextPlot 属性设置的绘图函数。

newplot 检查 NextPlot 属性的值并根据这些值采取合适的操作。将 newplot 放在所有调用对象创建函数的函数开头。

当您的函数调用 newplot 时，newplot 首先查询图窗 NextPlot 属性。根据属性值，newplot 会采取下表中描述的操作。

图窗 NextPlot 属性值	newplot 函数
不存在图窗	创建图窗，并使该图窗成为当前图窗。
add	使该图窗成为当前图窗。
new	创建新图窗，并使其成为当前图窗。
replacechildren	删除图窗的子对象（坐标区对象及其后代）并使该图窗成为当前图窗。
replace	删除图窗的子对象，将图窗属性重置为其默认值，并使该图窗成为当前图窗。

然后 newplot 检查当前坐标区的 NextPlot 属性。根据属性值，newplot 执行下表中描述的操作。

坐标区 NextPlot 属性值	newplot 函数
当前图窗中没有坐标区	创建坐标区，并使其成为当前坐标区
add	使该坐标区成为当前坐标区，并返回其句柄。

坐标区 NextPlot 属性值	newplot 函数
replacechildren	删除坐标区的子对象并使该坐标区成为当前坐标区。
replace	删除坐标区的子对象、将坐标区的属性重置为其默认值并使该坐标区成为当前坐标区。

使用 newplot 控制绘图

此示例演示如何将图窗和坐标区用于用户编写的绘图函数。使用圆点表示法设置属性。

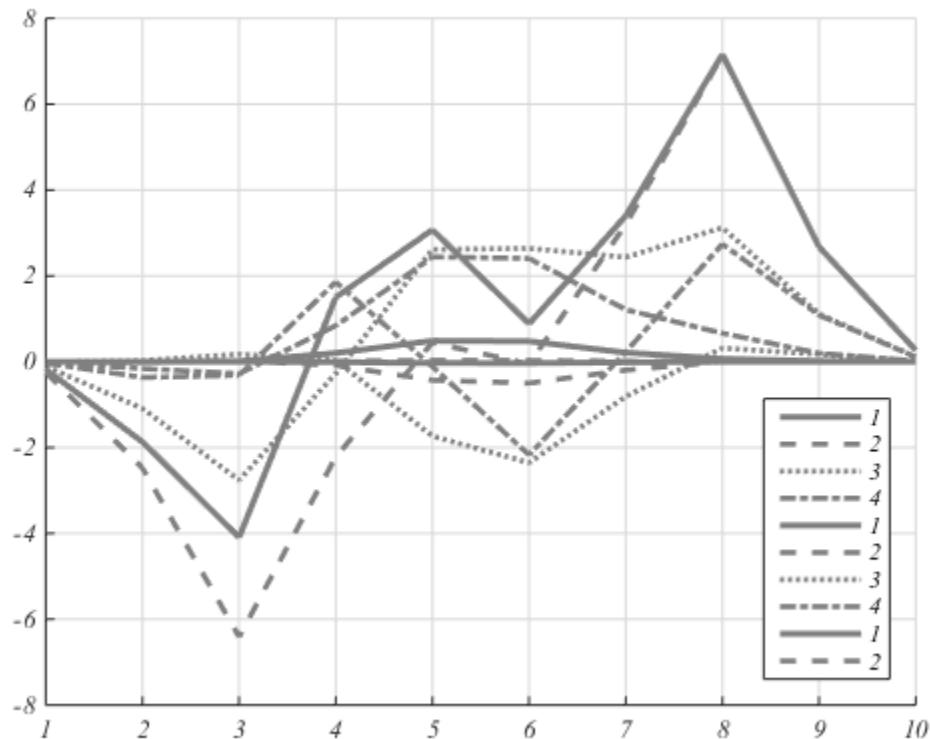
使用 **newplot** 管理指定绘图函数输出。 **myPlot2D** 函数：

- 为特定的出版要求自定义坐标区和图窗外观。
- 针对多线条图形循环使用线型和单一颜色。
- 添加具有指定显示名称的图例。

```
function myPlot2D(x,y)
% Call newplot to get the axes handle
cax = newplot;
% Customize axes
cax.FontName = 'Times';
cax.FontAngle = 'italic';
% Customize figure
fig = cax.Parent;
fig.MenuBar= 'none';
% Call plotting commands to
% produce custom graph
hLines = line(x,y, ...
    'Color',[.5,.5,.5],...
    'LineWidth',2);
lso = [ '-' ';' '-' ';' '-' ];
setLineStyle(hLines)
grid on
legend('show','Location','SouthEast')
function setLineStyle(hLines)
style = 1;
for ii = 1:length(hLines)
    if style > length(lso)
        style = 1;
    end
    hLines(ii).LineStyle = lso(style,:);
    hLines(ii).DisplayName = num2str(style);
    style = style + 1;
end
end
end
```

该图形显示 **myPlot2D** 函数的典型输出：

```
x = 1:10;
y = peaks(10);
myPlot2D(x,y)
```



myPlot2D 函数显示用户编写的绘图函数的基本结构：

- 调用 `newplot` 获得目标坐标区的句柄，并应用坐标区和图窗的 `NextPlot` 属性设置。
- 使用返回的坐标区句柄为该特定绘图函数自定义坐标区或图窗。
- 调用绘图函数（例如，`line` 和 `legend`）以实现指定的图形。

由于 myPlot2D 使用 `newplot` 返回的句柄访问目标图窗和坐标区，该函数：

- 在使用各后续调用清除坐标区时遵循 MATLAB 绘图函数的行为。
- 在 `hold` 设置为 `on` 时，工作正常。

`NextPlot` 属性的默认设置确保您的绘图函数遵循标准 MATLAB 行为 - 重用图窗窗口，但不清除和重置每个新图形的坐标区。

对保留状态进行响应

此示例演示如何在用户定义绘图函数中测试 **hold** 状态并正确响应。

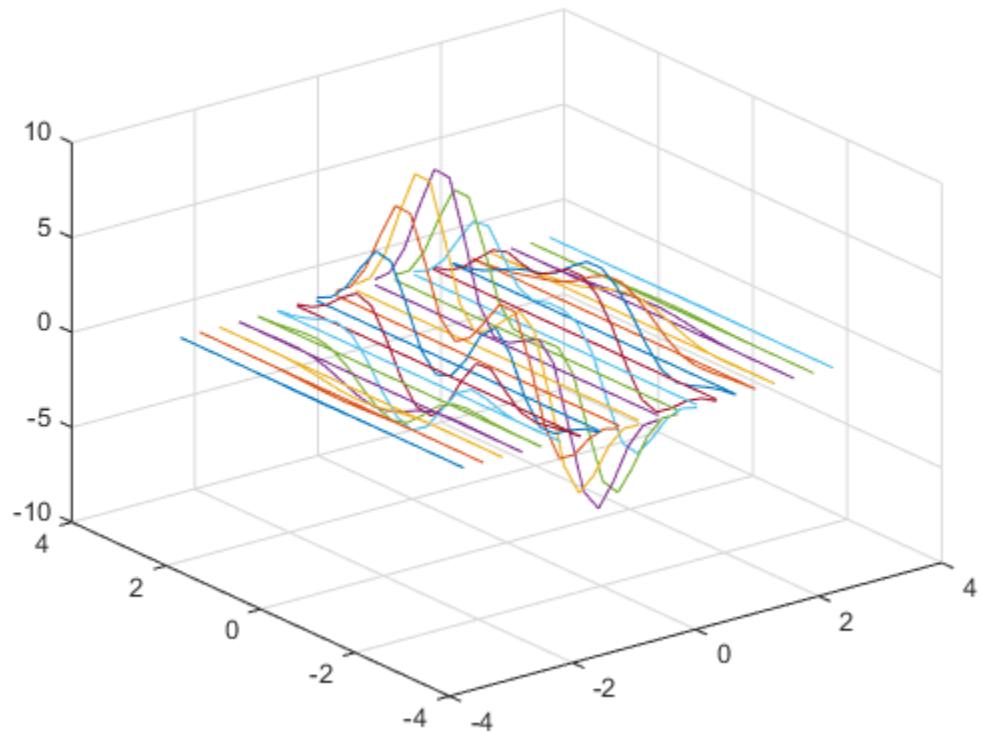
绘图函数通常会改变各种坐标区参数以适应不同的数据。 **myPlot3D** 函数：

- 根据输入数据使用二维或三维视图。
- 遵守当前 **hold** 状态，与 MATLAB 绘图函数行为一致。

```
function myPlot3D(x,y,z)
% Call newplot to get the axes handle
cax = newplot;
% Save current hold state
hold_state = ishold;
% Call plotting commands to
% produce custom graph
if nargin == 2
    line(x,y);
    % Change view only if hold is off
    if ~hold_state
        view(cax,2)
    end
elseif nargin == 3
    line(x,y,z);
    % Change view only if hold is off
    if ~hold_state
        view(cax,3)
    end
end
grid on
end
```

例如，第一次 **myPlot3D** 调用创建一个三维图形。第二次调用 **myPlot3D** 将二维数据添加到三维视图，这是因为 **hold** 为 **on**。

```
[x,y,z] = peaks(20);
myPlot3D(x,y,z)
hold on
myPlot3D(x,y)
```



防止对图窗和坐标区访问

本节内容

[“为什么防止访问” \(第 22-11 页\)](#)

[“如何防止访问” \(第 22-11 页\)](#)

为什么防止访问

某些情况下，防止特定图窗或坐标区成为图形输出的目标非常重要。也就是防止其成为 `gcf` 返回的当前图窗，或 `gca` 返回的当前坐标区。

您可能想要防止访问包含实现用户界面的控件的图窗。或者您想要防止访问作为应用程序一部分且只能被应用程序访问的坐标区。

如何防止访问

通过将特定图窗或坐标区从 MATLAB 函数的可见句柄列表中移除，以防止该函数访问这些图窗或坐标区。

有两个属性控制句柄可见性：`HandleVisibility` 和 `ShowHiddenHandles`

`HandleVisibility` 是所有图形对象的属性。它控制对象句柄的可见性，有三个可能值：

- '`on`' - 您可以通过返回句柄的函数获得对象句柄，如 (`gcf`、`gca`、`gco`、`get` 和 `findobj`)。这是默认行为。
- '`callback`' - 对象的句柄仅在回调函数的工作区中可见。
- '`off`' - 句柄对所有在命令行窗口和回调函数中执行的函数都隐藏。

受句柄可见性影响的属性

当对象的 `HandleVisibility` 设置为 '`callback`' 或 '`off`' 时：

- 对象句柄不会出现在其父对象的 `Children` 属性中。
- 图窗不会出现在根对象的 `CurrentFigure` 属性中。
- 坐标区不会出现在包含图窗的 `CurrentAxes` 属性中。
- 图形对象不会出现在图窗的 `CurrentObject` 属性中。

受句柄可见性影响的函数

当句柄在其父对象的子对象列表中不可见时，通过搜索对象层次结构获取句柄的函数无法返回该句柄。这些函数包括 `get`、`findobj`、`gca`、`gcf`、`gco`、`newplot`、`cla`、`clf` 和 `close`。

gca 和 gcf 返回值

如果隐藏句柄的图窗位于屏幕最顶层，而可见句柄的图窗叠放在其后，那么 `gcf` 会在堆栈中返回最顶层图窗。同样的行为也适用于 `gca`。如果不存在可见句柄的图窗，那么调用 `gcf` 或 `gca` 创建一个。

访问隐藏句柄对象

根 `ShowHiddenHandles` 属性启用或禁用句柄可见性控制。默认情况下，`ShowHiddenHandles` 为 '`off`'，这意味着 MATLAB 遵循每个对象 `HandleVisibility` 属性设置。

将 **ShowHiddenHandles** 设置为 **on** 等效于将图形层次结构中所有对象的 **HandleVisibility** 属性设置为 **on**。

注意 坐标区标题和坐标区标签文本对象不是坐标区的子对象。要访问这些对象的句柄，使用坐标区 **Title**、**XLabel**、**YLabel** 和 **ZLabel** 属性。

close 函数还允许使用 **hidden** 选项访问隐藏句柄的图窗。例如：

```
close('hidden')
```

关闭屏幕上最顶层图窗，即使其句柄是隐藏的。

合并 **all** 和 **hidden** 选项：

```
close('all','hidden')
```

关闭所有图窗。

句柄有效性与句柄可见性

无论句柄的 **HandleVisibility** 属性的状态如何，所有句柄都保持有效。如果您已将一个对象句柄分配给一个变量，那么您可以使用该句柄变量设置或获取其属性。

开发图对象的类

- “图开发概述” (第 23-2 页)
- “为图类编写构造函数” (第 23-9 页)
- “创建包含极坐标区、地理坐标区或多个坐标区的图” (第 23-13 页)
- “管理图类的属性” (第 23-17 页)
- “启用便利函数以设置坐标区属性” (第 23-25 页)
- “保存和加载图类的实例” (第 23-31 页)
- “具有自定义属性显示的图类” (第 23-37 页)
- “具有可变数量的线条的图类” (第 23-40 页)
- “用于显示不定数量的线条的优化图类” (第 23-43 页)
- “用于显示可变大小绘图分块的图类” (第 23-47 页)
- “包含两个交互式绘图的图类” (第 23-50 页)

图开发概述

使用绘图函数（如 `plot`、`scatter` 和 `bar`），您可以通过颜色和线型等方面的基本控制来快速可视化数据。要创建自定义图，您可以组合多个图形对象，设置这些对象的属性，或调用其他函数。在 R2019a 及更早版本中，存储自定义代码并与他人共享的一种常见方法是编写脚本或函数。

从 R2019b 开始，您可以通过定义 `ChartContainer` 基类的子类来为图创建类实现。创建类使您能够：

- 为用户提供方便的接口 - 当用户要自定义图的某个方面时，他们可以设置属性，而不必修改和重新运行图形代码。用户可以在命令行修改属性或在属性检查器中检查它们。
- 封装算法和基本图形对象 - 实现用于执行计算和管理基础图形对象的方法。以这种方式组织您的代码，您可以对用户隐藏实现细节。

定义派生自该基类的图时，图的实例是图形对象层次结构的成员。因此，您的图与图形系统的许多方面兼容。例如，`gca` 和 `findobj` 函数可以返回图的实例。

图类的结构

图类的第一行将 `matlab.graphics.chartcontainer.ChartContainer` 类指定为超类。例如，名为 `ConfidenceChart` 的类的第一行如下所示：

```
classdef ConfidenceChart < matlab.graphics.chartcontainer.ChartContainer
```

除了指定超类以外，还需要/建议在类定义中包括以下组件。

组件	说明
公共属性代码块（第 23-3 页） (推荐)	此代码块定义您允许用户访问的所有属性。这些属性一起构成图的用户界面。
私有属性代码块（第 23-3 页） (推荐)	此代码块存储您不希望用户访问的基础图形对象和其他实现细节。 在此代码块中，设置以下属性值： <ul style="list-style-type: none"> • <code>Access = private</code> • <code>Transient</code> • <code>NonCopyable</code>
<code>setup</code> 方法（第 23-3 页） (必需)	此方法设置图的初始状态。它会在 MATLAB 构造对象时执行一次。 在受保护的代码块中定义此方法，以便只有您的类才能执行它。
<code>update</code> 方法（第 23-4 页） (必需)	此方法更新图中的底层对象。它在用户更改一个或多个属性值后的下一次 <code>drawnow</code> 执行期间执行。 在与 <code>setup</code> 方法相同的受保护模块中定义此方法。

隐式构造函数方法

您不必为您的类编写构造函数方法，因为构造函数是从 `ChartContainer` 基类继承的。构造函数接受可选的输入参数：父容器和任意数量的用于设置图属性的名称-值对组参数。例如，如果您定义名为 `ConfidenceChart` 的类，该类具有公共属性 `XData` 和 `YData`，您可以使用以下命令之一创建该类的实例：

```
c = ConfidenceChart(gcf,'XData',[1 2 3],'YData',[4 5 6])
c = ConfidenceChart('XData',[1 2 3],'YData',[4 5 6])
```

如果您要提供接口，以与典型函数相同的方式接受输入参数，您可以定义自定义构造函数方法。有关详细信息，请参阅“为图类编写构造函数”（第 23-9 页）。

公共和私有属性块

在至少两个属性块之间划分您的类属性：

- 用于存储面向用户界面的组件的公共属性块
- 用于存储要隐藏的实现细节的私有属性块

公共属性块中的属性存储用户提供的输入值。例如，显示线条的图可能会将 x 和 y 坐标向量存储在两个公共属性中。由于属性名称-值对组参数是隐式构造函数方法的可选输入，因此推荐的方法是将公共属性初始化为默认值。如果您定义了存储坐标值的公共属性，如果用户在没有任何输入的情况下调用构造函数，则将这些公共属性初始化为 NaN 值或空数组会构造一个空图。

私有属性块中的属性存储构成图的基础图形对象，以及要存储的任何计算值。最终，您的类将使用公共属性中的数据来配置底层对象。通过为私有属性块设置 `Transient` 和 `NonCopyable` 特性，可以避免在用户复制或保存图实例时存储冗余信息。

例如，以下是显示 `Line` 对象和 `Patch` 对象的图的属性块。公共属性块存储用户可以控制的值：线的 x 和 y 坐标、置信边界值、标记符号和颜色值。私有属性块存储 `Line` 和 `Patch` 对象。

```
properties
    XData = NaN
    YData = NaN
    ConfidenceMargin = 0.15
    MarkerSymbol = 'o'
    Color = [1 0 0]
end

properties(Access = private,Transient,NonCopyable)
    LineObject
    PatchObject
end
```

setup 方法

当 MATLAB 构造图对象时，`setup` 方法会执行一次。执行此方法后，将赋予作为名称-值对组参数传递给构造函数方法的任何属性值。

使用 `setup` 方法：

- 调用绘图函数来创建要在图中使用的基本图形对象。
- 将绘图函数返回的基本对象作为私有属性存储在图对象上。
- 配置基本图形对象。
- 配置坐标区。

许多图形函数都有可选的输入参数来指定目标坐标区对象。这些函数包括绘图函数（例如 `plot`、`scatter` 和 `bar`）和用于修改坐标区的函数（例如 `hold`、`grid` 和 `title`）。从类方法中调用这些类型的函数时，必须指定目标坐标区对象。您可以通过调用 `getAxes` 方法来访问坐标区对象。此方法将返回坐标区对象；如果图尚未包含坐标区对象，则将创建一个笛卡尔坐标区对象。

小心 在未指定目标坐标区的情况下，调用绘图函数或用于修改坐标区的函数可能会产生意外的结果。

在 **setup** 方法中调用绘图函数时，请为坐标数据指定临时值（例如 `Nan`）。另外，还需要为与类的公共属性相对应的其他参数指定临时值。这样做可以避免在 **setup** 和 **update** 方法中设置相同的属性值。

如果要在坐标区上显示多个基本对象，请在绘图命令之间调用 **hold** 函数。在完成最后一个绘图命令后，将 **hold** 状态设置回 `'off'`。

例如，假定有一个图，图中显示了一条线和一个补片。它具有以下属性：

- 两个公共属性，分别名为 **XData** 和 **YData**，用于存储线的 x 和 y 坐标
- 两个私有属性，分别名为 **LineObject** 和 **PatchObject**

setup 方法通过调用 **getAxes** 方法获取坐标区对象。然后，它会调用 **patch** 函数，并将输出存储在 **PatchObject** 属性中。在调用 **plot** 函数创建 **LineObject** 属性之前，下一行代码将坐标区的保留状态设置为 `'on'`。最后一行代码将坐标区 **hold** 状态设置回 `'off'`。

```
function setup(obj)
    % Get the axes
    ax = getAxes(obj);

    % Create Patch and Line objects
    obj.PatchObject = patch(ax,NaN,NaN,'r','FaceAlpha',0.2,...
        'EdgeColor','none');
    hold(ax,'on')
    obj.LineObject = plot(ax,NaN,NaN);

    % Turn hold state off
    hold(ax,'off')
end
```

update 方法

当用户更改图对象的一个或多个属性值时，MATLAB 会标记该图对象以进行更新。在 **setup** 方法运行后，将首次运行 **update** 方法。然后，它将在下次执行 **drawnow** 时运行。根据用户 MATLAB 会话中的图形环境状态，**drawnow** 函数会周期性自动执行。因此，在更改属性值和查看这些更改的结果之间可能会存在延迟。

使用 **update** 方法，根据公共属性的新值重新配置图中的基础图形对象。通常，此方法不会区分哪些公共属性发生了更改。它会重新配置依赖于公共属性的基础图形对象的所有方面。

例如，假设有具有以下属性的图：

- 两个公共属性，分别名为 **XData** 和 **Color**
- 两个名为 **LineObject** 和 **PatchObject** 的私有属性

update 方法会更新 **Line** 和 **Patch** 对象的 **XData** 和 **Color** 属性。

```
function update(obj)

    % Update XData of line object
    obj.LineObject.XData = obj.XData;

    % Update patch XData
```

```

x = obj.XData;
obj.PatchObject.XData = [x x(end:-1:1)];

% Update line object colors
obj.LineObject.Color = obj.Color;
obj.PatchObject.FaceColor = obj.Color;

end

```

示例：置信边界图

此示例说明如何创建一个图来绘制带有置信边界的线条。在位于 MATLAB 路径上的文件夹中创建名为 `ConfidenceChart.m` 的类定义文件。按照以下步骤定义类。

步骤	实现
从 <code>ChartContainer</code> 基类派生而来。	<code>classdef ConfidenceChart < matlab.graphics.chartcontainer.ChartContainer</code>
定义公共属性。	<code>properties</code> <code>XData = NaN</code> <code>YData = NaN</code> <code>ConfidenceMargin = 0.15</code> <code>MarkerSymbol = 'o'</code> <code>Color = [1 0 0]</code> <code>end</code>
定义私有属性。	<code>properties(Access = private, Transient, NonCopyable)</code> <code>LineObject</code> <code>PatchObject</code> <code>end</code>
实现 <code>setup</code> 方法。在本例中，调用 <code>plot</code> 和 <code>patch</code> 函数以分别创建 <code>Patch</code> 和 <code>Line</code> 对象。将这些对象存储在对应的私有属性中。 退出方法之前，将坐标区的 <code>hold</code> 状态设回 <code>'off'</code> 。	<code>methods(Access = protected)</code> <code>function setup(obj)</code> <code>% get the axes</code> <code>ax = getAxes(obj);</code> <code>% Create Patch and Line objects</code> <code>obj.PatchObject = patch(ax,NaN,NaN,'r','FaceAlpha',0.2,...</code> <code>'EdgeColor','none');</code> <code>hold(ax,'on')</code> <code>obj.LineObject = plot(ax,NaN,NaN);</code> <code>% Turn hold state off</code> <code>hold(ax,'off')</code> <code>end</code>

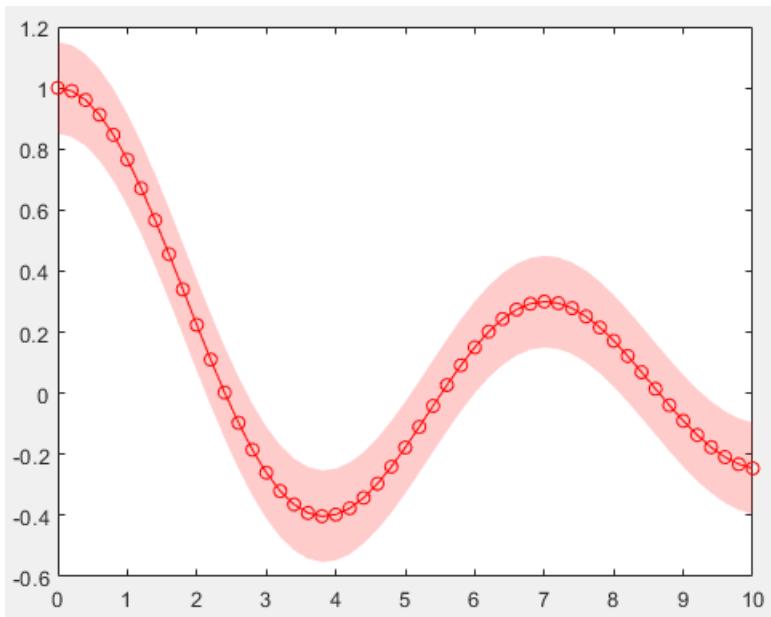
步骤	实现
实现 <code>update</code> 方法。在本例中，更新底层对象的 x 和 y 坐标、颜色和标记符号。	<pre> function update(obj) % Update XData and YData of Line obj.LineObject.XData = obj.XData; obj.LineObject.YData = obj.YData; % Update patch XData and YData x = obj.XData; obj.PatchObject.XData = [x x(end:-1:1)]; y = obj.YData; c = obj.ConfidenceMargin; obj.PatchObject.YData = [y+c y(end:-1:1)-c]; % Update colors obj.LineObject.Color = obj.Color; obj.PatchObject.FaceColor = obj.Color; % Update markers obj.LineObject.Marker = obj.MarkerSymbol; end end end </pre>

接下来，通过调用具有一些公共属性的隐式构造函数方法来创建图的实例：

```

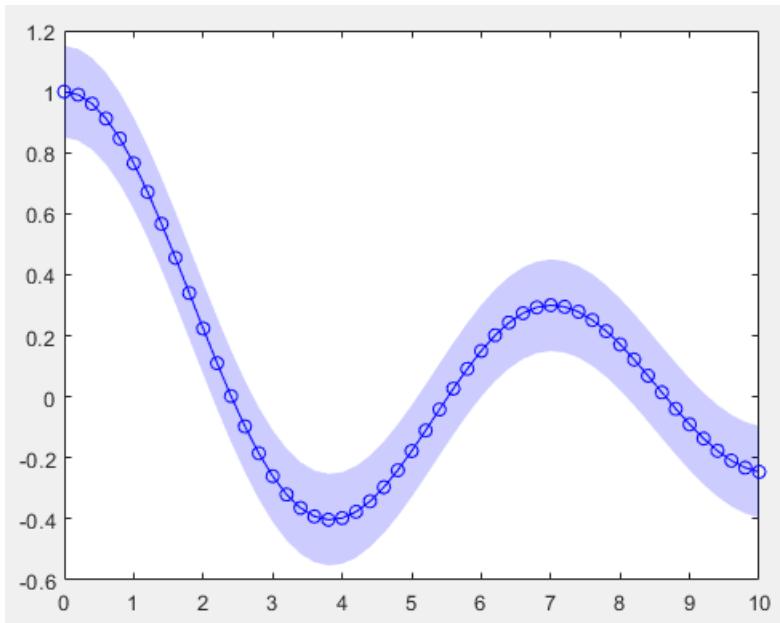
x = 0:0.2:10;
y = besselj(0,x);
c = ConfidenceChart('XData',x,'YData',y,'ConfidenceMargin',0.15);

```



更改颜色。

```
c.Color = [0 0 1];
```



提示 除了文档中提供的示例，您还可以在 MATLAB Central™ 的 File Exchange 上找到社区编写的各种示例。

支持常用图形功能

默认情况下，图实例支持所有 MATLAB 图通用的许多功能。例如，`gca` 和 `findobj` 函数可以返回图的实例。您还可以将图的实例传递给 `set` 和 `get` 函数，并且可以在属性检查器中配置图的属性。

只有当您对图启用了下表中所述的功能时，这些功能才可用。

功能	说明	更多信息
图例	启用图窗工具栏中的 <code>legend</code> 函数和图例工具。	<code>matlab.graphics.chartcontainer.mixin.Legend</code>
颜色栏	启用图窗工具栏中的 <code>colorbar</code> 函数和颜色栏工具。	<code>matlab.graphics.chartcontainer.mixin.Colorbar</code>
不同类型的坐标区或多个坐标区	显示一个或多个笛卡尔图、极坐标图或地理图。	“创建包含极坐标区、地理坐标区或多个坐标区的图”（第 23-13 页）
函数	启用在坐标区上设置属性的函数，如 <code>title</code> 、 <code>xlim</code> 、 <code>ylim</code> 函数。	“启用便利函数以设置坐标区属性”（第 23-25 页）
保存和加载	在用户完成与图的交互后存储更改，以便在将图加载回 MATLAB 时保存图并还原其状态。	“保存和加载图类的实例”（第 23-31 页）

另请参阅

类

`matlab.graphics.chartcontainer.ChartContainer`

函数

`plot | patch`

属性

`Line | Patch`

详细信息

- “[定义类](#)”
- “[创建包含极坐标区、地理坐标区或多个坐标区的图](#)” (第 23-13 页)

为图类编写构造函数

当您将图开发为 `ChartContainer` 基类的子类时，基类提供默认构造函数，该构造函数接受用于设置图属性的可选名称-值对组参数。例如，此命令创建名为 `ConfidenceChart` 的类实例。

```
ConfidenceChart('XData',x,'YData',y,'ConfidenceMargin',0.15,'Color',[1 0 0])
```

通过编写自定义构造函数方法，您可以提供一个接口，该接口接受单个参数值和可选的名称-值对组参数。例如，您可以设计自定义构造函数来更改 `ConfidenceChart` 的调用语法，以便这两个命令均为创建图的有效方法：

```
ConfidenceChart(x,y,0.15)
ConfidenceChart(x,y,0.15,'Color',[1 0 0])
```

当您编写构造函数方法时，请：

- 指定要在函数声明中支持的输入参数。将 `varargin` 作为最后一个输入参数，以捕获用户指定的任何属性名称-值对组参数。
- 在对图对象进行所有其他引用之前调用 `ChartContainer` 构造函数。

例如，`ConfidenceChart` 类的以下构造函数方法将执行下列任务：

- 检查输入参数的数量，如果该数量少于三个，则返回错误。
- 将 `x`、`y` 和 `margin` 值转换为 `ChartContainer` 构造函数接受的名称-值对组参数，并将结果存储在 `args` 中。
- 将用户指定的任何名称-值对组参数追加到 `args` 的末尾。
- 将 `args` 传递给 `ChartContainer` 构造函数方法。

```
methods
    function obj = ConfidenceChart(x,y,margin,varargin)
        % Check for at least three inputs
        if nargin < 3
            error('Not enough inputs');
        end

        % Convert x, y, and margin into name-value pairs
        args = {'XData', x, 'YData', y, 'ConfidenceMargin', margin};

        % Combine args with user-provided name-value pairs
        args = [args varargin];

        % Call superclass constructor method
        obj@matlab.graphics.chartcontainer.ChartContainer(args{:});
    end
end
```

示例：具有自定义构造函数的置信边界图

此示例说明如何开发图，该图具有接受单值输入参数和可选的名称-值对组参数的自定义构造函数。该图绘制一条带标记和置信边界的线。

在位于 MATLAB 路径上的文件夹中创建一个名为 `ConfidenceChart.m` 的程序文件。按照以下步骤定义类。

步骤	实现
从 ChartContainer 基类派生而来。	<pre>classdef ConfidenceChart < matlab.graphics.chartcontainer.ChartContainer</pre>
定义公共属性。	<pre>properties XData (1,:) double = NaN YData (1,:) double = NaN ConfidenceMargin (1,1) double = 0.15 MarkerSymbol (1,:) char = 'o' Color (1,3) double {mustBeGreaterThanOrEqual(Color,0),... mustBeLessThanOrEqual(Color,1)} = [1 0 0] end</pre>
定义私有属性。	<pre>properties(Access = private,Transient,NonCopyable) LineObject (1,1) matlab.graphics.chart.primitive.Line PatchObject (1,1) matlab.graphics.primitive.Patch end</pre>
实现自定义构造函数方法，该方法接受 x、y 和 margin 值以及可选的属性名称-值对组参数。	<pre>methods function obj = ConfidenceChart(x,y,margin,varargin) % Check for at least three inputs if nargin < 3 error('Not enough inputs'); end % Convert x, y, and margin into name-value pairs args = {'XData', x, 'YData', y, 'ConfidenceMargin', margin}; % Combine args with user-provided name-value pairs. args = [args varargin]; % Call superclass constructor method obj@matlab.graphics.chartcontainer.ChartContainer(args{:}); end end</pre>
实现 setup 方法。	<pre>methods(Access = protected) function setup(obj) % get the axes ax = getAxes(obj); % Create Patch and objects obj.PatchObject = patch(ax,NaN,NaN,'r','FaceAlpha',0.2,',... 'EdgeColor','none'); hold(ax,'on') obj.LineObject = plot(ax,NaN,NaN); hold(ax,'off') end</pre>

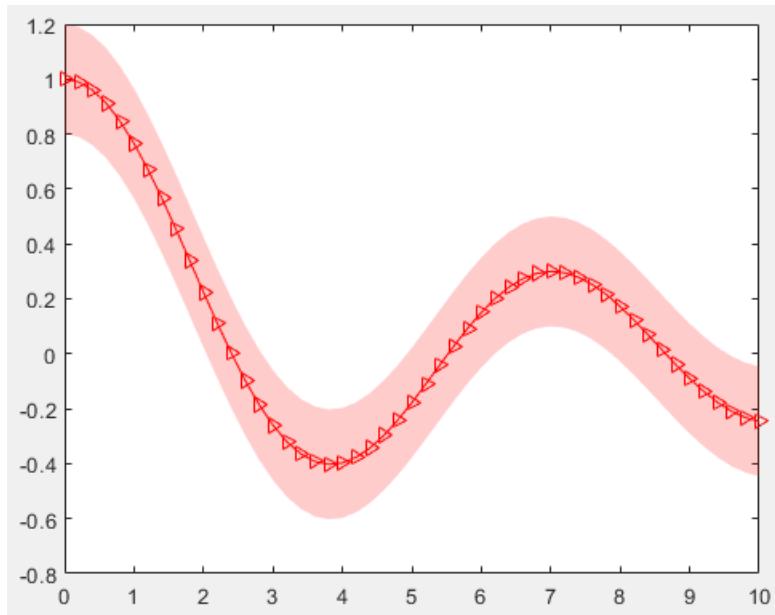
步骤	实现
实现 update 方法。	<pre> function update(obj) % Update XData and YData of Line obj.LineObject.XData = obj.XData; obj.LineObject.YData = obj.YData; % Update patch XData and YData x = obj.XData; obj.PatchObject.XData = [x x(end:-1:1)]; y = obj.YData; c = obj.ConfidenceMargin; obj.PatchObject.YData = [y+c y(end:-1:1)-c]; % Update colors obj.LineObject.Color = obj.Color; obj.PatchObject.FaceColor = obj.Color; % Update markers obj.LineObject.Marker = obj.MarkerSymbol; end end end </pre>

接下来，创建 **ConfidenceChart** 的一个实例。指定 x 和 y 坐标、边距值和标记符号。

```

x = 0:0.2:10;
y = besselj(0,x);
ConfidenceChart(x,y,0.20,'MarkerSymbol','>');

```



另请参阅

类

`matlab.graphics.chartcontainer.ChartContainer`

函数

plot | patch

属性

Line | Patch

详细信息

- “类构造函数方法”
- “对子类对象调用超类方法”
- “图开发概述” (第 23-2 页)

创建包含极坐标区、地理坐标区或多个坐标区的图

对于作为 `ChartContainer` 基类的子类创建的图，`getAxes` 方法提供了一种方法来支持单个笛卡尔坐标区对象。如果要支持极坐标区、地理坐标区或多个坐标区，必须创建这些坐标区并将其配置为 `TiledChartLayout` 对象（该对象存储在图对象中）的子对象。

创建单个极坐标区或地理坐标区对象

要在图中包含单个极坐标区或地理坐标区对象：

- 1 定义一个私有属性来存储坐标区。
- 2 在 `setup` 方法中：
 - 调用 `getLayout` 方法以获取 `TiledChartLayout` 对象。
 - 调用 `polaraxes` 或 `geoaxes` 函数以创建坐标区，并指定 `TiledChartLayout` 对象作为父对象。

例如，下面是一个包含极坐标区对象的基本类。

```
classdef SimplePolar < matlab.graphics.chartcontainer.ChartContainer
  properties(Access = private, Transient, NonCopyable)
    PolarAx matlab.graphics.axis.PolarAxes
  end

  methods(Access = protected)
    function setup(obj)
      % Get the layout and create the axes
      tcl = getLayout(obj);
      obj.PolarAx = polaraxes(tcl);

      % Other setup code
      % ...
    end
    function update(obj)
      % Update the chart
      % ...
    end
  end
end
```

创建多个坐标区对象的分块

要显示多个坐标区的分块，请：

- 1 定义存储坐标区对象的私有属性。您还可以定义一个属性，用于存储坐标区对象数组。
- 2 在 `setup` 方法中：
 - 调用 `getLayout` 方法以获取 `TiledChartLayout` 对象。
 - 设置 `TiledChartLayout` 对象的 `GridSize` 属性，使其针对每个坐标区包含至少一个图块。
 - 调用 `axes`、`polaraxes` 或 `geoaxes` 函数以创建坐标区对象，并指定 `TiledChartLayout` 对象作为父对象。
 - 通过设置每个坐标区对象的 `Layout` 属性，将每个坐标区移到目标图块。默认情况下，坐标区出现在第一个图块中。

例如，以下是一个包含两个笛卡尔坐标区的基本类：

```
classdef TwoAxesChart < matlab.graphics.chartcontainer.ChartContainer
    properties(Access = private,Transient,NonCopyable)
        Ax1 matlab.graphics.axis.Axes
        Ax2 matlab.graphics.axis.Axes
    end

    methods(Access = protected)
        function setup(obj)
            % Get the layout and set the grid size
            tcl = getLayout(obj);
            tcl.GridSize = [2 1];

            % Create the axes
            obj.Ax1 = axes(tcl);
            obj.Ax2 = axes(tcl);

            % Move the second axes to the second tile
            obj.Ax2.Layout.Tile = 2;
        end
        function update(obj)
            % Update the chart
            % ...
        end
    end
end
```

示例：包含地理坐标区和笛卡尔坐标区的图

以下示例说明了如何定义一个图类，以使用两个坐标区可视化地理和分类数据。左侧坐标区包含一幅地图，显示了多个蜂窝塔的位置。右侧坐标区按类别显示了这些塔的分布情况。

以下 TowerChart 类定义显示了如何：

- 定义一个名为 TowerData 的用于存储表的公共属性。
- 使用一个名为 mustHaveRequiredVariables 的局部函数验证表的内容。
- 定义两个私有属性 MapAxes 和 HistogramAxes，用于存储坐标区。
- 实现 setup 方法，用于获取 TiledChartLayout 对象，指定布局的网格大小，以及定位坐标区。

要定义该类，请将以下代码复制到编辑器中，并将其以名称 TowerChart.m 保存在可写文件夹中。

```
classdef TowerChart < matlab.graphics.chartcontainer.ChartContainer
    properties
        TowerData (:,:) table {mustHaveRequiredVariables} = table([],...
            [],[],'VariableNames',{'STRUCTYPE','Latitude','Longitude'})
    end

    properties (Access = private,Transient,NonCopyable)
        MapAxes matlab.graphics.axis.GeographicAxes
        HistogramAxes matlab.graphics.axis.Axes
        ScatterObject matlab.graphics.chart.primitive.Scatter
        HistogramObject matlab.graphics.chart.primitive.categorical.Histogram
    end
```

```

methods (Access = protected)
    function setup(obj)
        % Configure layout and create axes
        tcl = getLayout(obj);
        tcl.GridSize = [1 2];
        obj.MapAxes = geoaxes(tcl);
        obj.HistogramAxes = axes(tcl);

        % Move histogram axes to second tile
        obj.HistogramAxes.Layout.Tile = 2;

        % Create Scatter and Histogram objects
        obj.ScatterObject = geoscatte(r(obj.MapAxes,NaN,NaN,'.'));
        obj.HistogramObject = histogram(obj.HistogramAxes,categorical.empty, ...
            'Orientation','horizontal');

        % Add titles to the axes
        title(obj.MapAxes,"Tower Locations")
        title(obj.HistogramAxes,"Tower Types")
        xlabel(obj.HistogramAxes,"Number of Towers")
    end

    function update(obj)
        % Update Scatter object
        obj.ScatterObject.LatitudeData = obj.TowerData.Latitude;
        obj.ScatterObject.LongitudeData = obj.TowerData.Longitude;

        % Get tower types from STRUCTYPE table variable
        towertypes = obj.TowerData.STRUCTYPE;

        % Check for empty towertypes before updating histogram
        if ~isempty(towertypes)
            obj.HistogramObject.Data = towertypes;
            obj.HistogramObject.Categories = categories(towertypes);
        else
            obj.HistogramObject.Data = categorical.empty;
        end
    end
end

function mustHaveRequiredVariables(tbl)
    % Return error if table does not have required variables
    assert(all(ismember({'STRUCTYPE','Latitude','Longitude'},...
        tbl.Properties.VariableNames)),...
        'MATLAB:TowerChart:InvalidTable',...
        'Table must have STRUCTYPE, Latitude, and Longitude variables.');
end

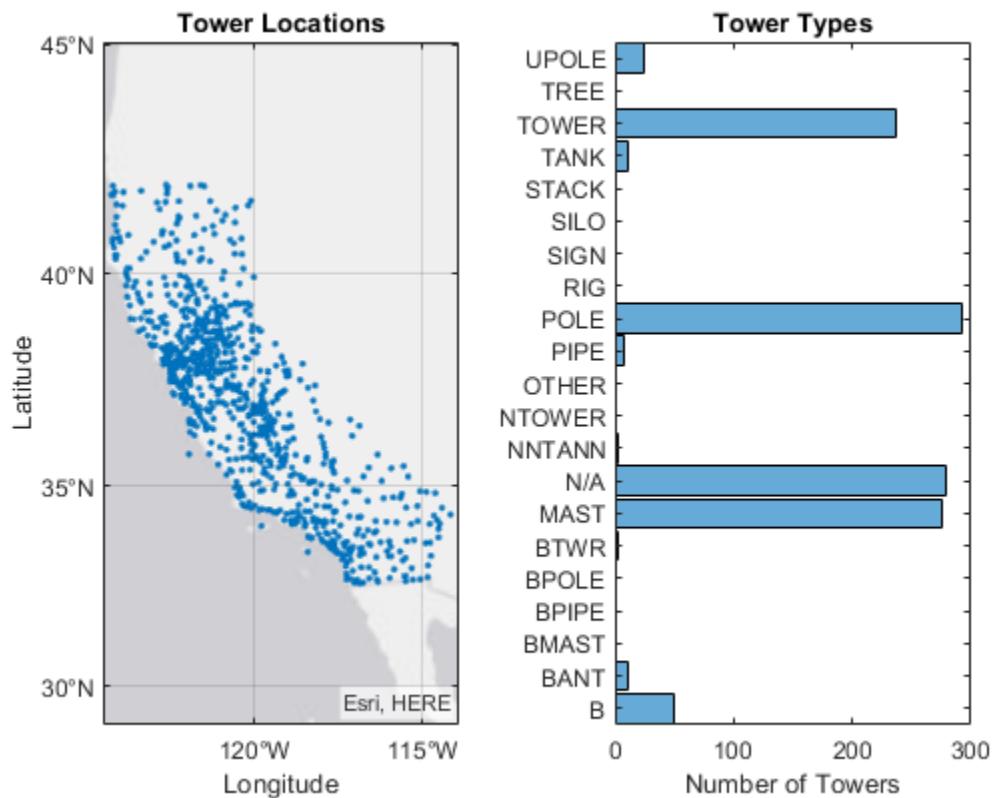
```

保存类文件后，加载 `cellularTowers.mat` 中存储的表。然后通过将该表作为名称-值对组参数传递给 `TowerChart` 构造函数方法来创建图实例。

```

load cellularTowers.mat
TowerChart('TowerData',cellularTowers);

```



另请参阅

函数

`getLayout`

类

`matlab.graphics.chartcontainer.ChartContainer`

属性

`TiledChartLayout` Properties

详细信息

- “图开发概述” (第 23-2 页)

管理图类的属性

当您将自定义图作为 `ChartContainer` 基类的子类进行开发时，您可以使用某些方法来使代码更加稳健、高效，并针对用户的需求而量身定制。这些方法侧重于如何定义和管理类的属性。使用的任何方法都应该有助于您要创建的可视化的类型和您要提供的用户体验。

- 初始化属性值（第 23-17 页） - 设置图的默认状态，以防用户在没有任何输入参数的情况下调用隐式构造函数。
- 验证属性值（第 23-17 页） - 在使用这些值执行计算或配置图中的一个基础图形对象之前，确保这些值有效。
- 自定义属性显示（第 23-18 页） - 当用户不带分号引用图对象时显示自定义属性列表。
- 优化 `update` 方法（第 23-19 页） - 当在某个耗时的计算中只用到部分属性时提高 `update` 方法的性能。

初始化属性值

为您的类的所有公共属性指定默认值。如果用户在调用构造函数方法时忽略了一些名称-值对组参数，则这样做可配置有效的图。

对于存储坐标数据的属性，将初始值设置为 `NaN` 值或空数组，以便当用户未指定坐标时，默认图为空。根据您计划在类方法中调用的绘图函数的要求选择默认坐标。要了解这些要求，请参阅您计划使用的绘图函数的文档。

验证属性值

好的做法是在代码使用类属性值之前先验证这些值。一种方便的方法是在定义属性时验证属性的大小和类。例如，下面的属性块验证四个属性的大小和类。

```
properties
    IsoValue (1,1) double = 0.5
    Enclose {mustBeMember(Enclose,['above','below'])} = 'below'
    CapVisible (1,1) matlab.lang.OnOffSwitchState = 'on'
    Color (1,3) double {mustBeGreaterThanOrEqual(Color,0),...
        mustBeLessThanOrEqual(Color,1)} = [.2 .5 .8]
end
```

- `IsoValue` 必须为 `double` 类的 1×1 数组。
- `Enclose` 的值必须为 '`above`' 或 '`below`'。
- `CapVisible` 必须为 `matlab.lang.OnOffSwitchState` 类的 1×1 数组。
- `Color` 必须为 `double` 类的 1×3 数组，其中每个值都在 $[0,1]$ 范围内。

您还可以验证在您的图中存储基础图形对象的属性。要确定对象的类名，请在命令行调用对应的绘图函数，然后调用 `class` 函数获取类名。例如，如果您计划在 `setup` 方法中调用 `patch` 函数，请在命令行中带输出参数（输入参数与此无关）调用 `patch` 函数。然后将输出传递给 `class` 函数，以获取其类名。

```
x = patch(NaN,NaN,NaN);
class(x)

ans =

'matlab.graphics.primitive.Patch'
```

使用 `class` 函数的输出来验证您的类中对应属性的类。例如，以下每个属性存储一个 `Patch` 对象。

```
properties (Access = private,Transient,NonCopyable)
    IsoPatch (1,1) matlab.graphics.primitive.Patch
    CapPatch (1,1) matlab.graphics.primitive.Patch
end
```

有时，您可能希望定义可以存储不同形状和值类的属性。例如，如果您定义可以存储字符向量、字符向量元胞数组或字符串数组的属性，请省略大小和类验证或使用自定义属性验证方法。

有关验证属性的详细信息，请参阅“[验证属性值](#)”。

自定义属性显示

将图定义为 `ChartContainer` 基类的子类的好处之一是，它还继承 `matlab.mixin.CustomDisplay` 类。因此，当您以不带分号的方式引用图时，可以自定义 MATLAB 在命令行窗口中显示的属性列表。要自定义属性显示，请重载 `getPropertyGroups` 方法。在该方法中，您可以自定义列出哪些属性以及列表的顺序。例如，假设有具有以下公共属性的 `IsoSurfCapChart` 类。

```
properties
    IsoValue (1,1) double = 0.5
    Enclose {mustBeMember(Enclose,['above','below'])} = 'below'
    CapVisible (1,1) matlab.lang.OnOffSwitchState = 'on'
    Color (1,3) double {mustBeGreaterThanOrEqual(Color,0),...
        mustBeLessThanOrEqual(Color,1)} = [.2 .5 .8]
end
```

以下 `getPropertyGroups` 方法将标量对象属性列表指定为 `Color`、`IsoValue`、`Enclose` 和 `CapVisible`。

```
function propgrp = getPropertyGroups(obj)
    if ~isscalar(obj)
        % List for array of objects
        propgrp = getPropertyGroups@matlab.mixin.CustomDisplay(obj);
    else
        % List for scalar object
        propList = {'Color','IsoValue','Enclose','CapVisible'};
        propgrp = matlab.mixin.util.PropertyGroup(propList);
    end
end
```

当用户以不带分号的方式引用此图的实例时，MATLAB 会显示自定义列表。

```
c = IsoSurfCapChart
```

```
c =
```

```
IsoSurfCapChart with properties:
```

```
    Color: [0.2000 0.5000 0.8000]
    IsoValue: 0.5000
    Enclose: 'below'
    CapVisible: on
```

有关自定义属性显示的详细信息，请参阅“[Customize Property Display](#)”。

优化 update 方法

在大多数情况下，您的类的 `update` 方法会重新配置图中依赖于公共属性的所有相关方面。有时，重新配置需要耗费大量时间进行大量计算。如果计算只涉及属性的子集，您可以将您的类设计为仅在必要时执行该代码。

优化 `update` 方法的一种方法是将这些组件添加到您的类中：

- 定义名为 `ExpensivePropChanged` 的私有属性，该属性接受 `logical` 值。此属性指示大量计算中使用的任何属性是否已更改。
- 为耗费大量资源的计算中涉及的每个属性编写一个 `set` 方法。在每个 `set` 方法中，将 `ExpensivePropChanged` 属性设置为 `true`。
- 编写一个执行大量计算的受保护方法。
- 在 `update` 方法中编写一个条件语句，用于检查 `ExpensivePropChanged` 的值。如果该值为 `true`，则会执行需要大量计算的方法。

以下代码提供这种设计的简化实现。

```
classdef OptimizedChart < matlab.graphics.chartcontainer.ChartContainer

    properties
        Prop1
        Prop2
    end
    properties(Access=private,Transient,NonCopyable)
        ExpensivePropChanged (1,1) logical = true
    end

    methods(Access = protected)
        function setup(obj)
            % Configure chart
            % ...
        end
        function update( obj )
            % Perform expensive computation if needed
            if obj.ExpensivePropChanged
                doExpensiveCalculation(obj);
                obj.ExpensivePropChanged = false;
            end

            % Update other aspects of chart
            % ...
        end
        function doExpensiveCalculation(obj)
            % Expensive code
            % ...
        end
    end
end

methods
    function set.Prop2(obj,val)
        obj.Prop2 = val;
        obj.ExpensivePropChanged = true;
    end
end
end
```

在本例中，`Prop2` 参与大量计算。`set.Prop2` 方法设置 `Prop2` 的值，然后将 `ExpensivePropChanged` 设置为 `true`。因此，下次运行 `update` 方法时，仅当 `ExpensivePropChanged` 为 `true` 时，才会调用 `doExpensiveCalculation`。然后，`update` 方法继续更新图的其他方面。

示例：具有自定义属性显示的优化的等值面图

定义一个 `IsoSurfCapChart` 类，用于显示 `isosurface` 和相关联的 `isocaps`。包括以下特征：

- 使用大小和类验证的属性
- 自定义的属性显示
- 优化的 `update` 方法，仅当一个或多个相关属性发生变化时，该方法才会重新计算 `isosurface` 和 `isocaps`

要定义此类，请在位于 MATLAB 路径上的文件夹中创建名为 `IsoSurfCapChart.m` 的程序文件。然后按照表中的步骤实现该类。

步骤	实现
从 <code>ChartContainer</code> 基类派生而来。	<code>classdef IsoSurfCapChart < matlab.graphics.chartcontainer.ChartContainer</code>
使用类和大小验证定义公共属性。 <ul style="list-style-type: none"> • <code>VolumeData</code>、<code>IsoValue</code> 和 <code>Color</code> 是 <code>isosurface</code> 的参数。 • <code>Enclose</code>、<code>WhichCapPlane</code> 和 <code>CapVisible</code> 是 <code>isocaps</code> 的参数。 	<pre>properties VolumeData double = rand(25,25,25) IsoValue (1,1) double = 0.5 Enclose {mustBeMember(Enclose,['above','below'])} = 'below' WhichCapPlane {mustBeMember(WhichCapPlane,['all','xmin',... 'xmax','ymin','ymax','zmin','zmax'])} = 'all' CapVisible (1,1) matlab.lang.OnOffSwitchState = 'on' Color (1,3) double {mustBeGreaterThanOrEqual(Color,0),... mustBeLessThanOrEqual(Color,1)} = [.2 .5 .8] end</pre>
定义私有属性。 <ul style="list-style-type: none"> • <code>IsoPatch</code> 和 <code>CapPatch</code> 存储 <code>isosurface</code> 和 <code>isocaps</code> 的 <code>Patch</code> 对象。 • <code>SmoothData</code> 存储三维体数据的经过平滑处理的版本。 • <code>ExpensivePropChanged</code> 指示更新方法是否需要重新计算 <code>isosurface</code> 和 <code>isocaps</code>。 	<pre>properties(Access = private,Transient,NonCopyable) IsoPatch (1,1) matlab.graphics.primitive.Patch CapPatch (1,1) matlab.graphics.primitive.Patch SmoothData double = []; ExpensivePropChanged (1,1) logical = true end</pre>

步骤	实现
实现 setup 方法。在本例中，调用 patch 函数两次，为 isosurface 和 isocaps 创建 Patch 对象。将对象存储在对应的属性中，并配置坐标区。	<pre> methods(Access = protected) function setup(obj) ax = getAxes(obj); % Create two Patch objects obj.IsoPatch = patch(ax,NaN,NaN,NaN,'EdgeColor','none',... 'FaceColor',[.2 .5 .8],'FaceAlpha',0.9); hold(ax,'on'); obj.CapPatch = patch(ax,NaN,NaN,NaN,'EdgeColor','none',... 'FaceColor','interp'); % Configure the axes view(ax,3) camlight(ax, 'infinite'); camlight(ax,'left'); lighting(ax, 'gouraud'); hold(ax,'off'); end </pre>
实现 update 方法。通过测试 ExpensivePropChanged 的值，决定是否调用 doExpensiveCalculation 方法。然后继续更新图的其他方面（开销较低）。	<pre> function update(obj) % Perform expensive computation if needed if obj.ExpensivePropChanged doExpensiveCalculation(obj); obj.ExpensivePropChanged = false; end % Update visibility of CapPatch and update color obj.CapPatch.Visible = obj.CapVisible; obj.IsoPatch.FaceColor = obj.Color; end </pre>
实现 doExpensiveCalculation 方法，该方法对三维体数据进行平滑处理并重新计算 isosurface 和 isocaps 的面和顶点。	<pre> function doExpensiveCalculation(obj) % Update isosurface obj.SmoothData = smooth3(obj.VolumeData,'box',7); [F,V] = isosurface(obj.SmoothData, obj.IsoValue); set(obj.IsoPatch,'Faces',F,'Vertices',V); isonormals(obj.SmoothData,obj.IsoPatch); % Update isocaps [m,n,p] = size(obj.SmoothData); [Xc,Yc,Zc] = meshgrid(1:n,1:m,1:p); [Fc,Vc,Cc] = isocaps(Xc,Yc,Zc,obj.SmoothData,obj.IsoValue, ... obj.Enclose,obj.WhichCapPlane); set(obj.CapPatch,'Faces',Fc,'Vertices',Vc,'CData',Cc); end </pre>

步骤	实现
实现 <code>getPropertyGroups</code> 方法以自定义属性显示。	<pre> function propgrp = getPropertyGroups(obj) if ~isscalar(obj) % List for array of objects propgrp = getPropertyGroups@matlab.mixin.CustomDisplay(obj); else % List for scalar object propList = {'Color','IsoValue','Enclose','CapVisible',... 'WhichCapPlane','VolumeData'}; propgrp = matlab.mixin.util.PropertyGroup(propList); end end end </pre>
对每个耗费大量资源的属性 (VolumeData、IsoValue、Enclose) 实现 set 方法。在每个方法中，设置对应的属性值，然后将 ExpensivePropChanged 设置为 true。	<pre> methods function set.VolumeData(obj,val) obj.VolumeData = val; obj.ExpensivePropChanged = true; end function set.IsoValue(obj, val) obj.IsoValue = val; obj.ExpensivePropChanged = true; end function set.Enclose(obj, val) obj.Enclose = val; obj.ExpensivePropChanged = true; end end end </pre>

接下来，创建一个三维体数据数组，然后创建 IsoSurfCapChart 的实例。

```

[X,Y,Z] = meshgrid(-2:0.1:2);
v = (1/9)*X.^2 + (1/16)*Y.^2 + Z.^2;
c = IsoSurfCapChart('VolumeData',v,'IsoValue',0.5)

c =

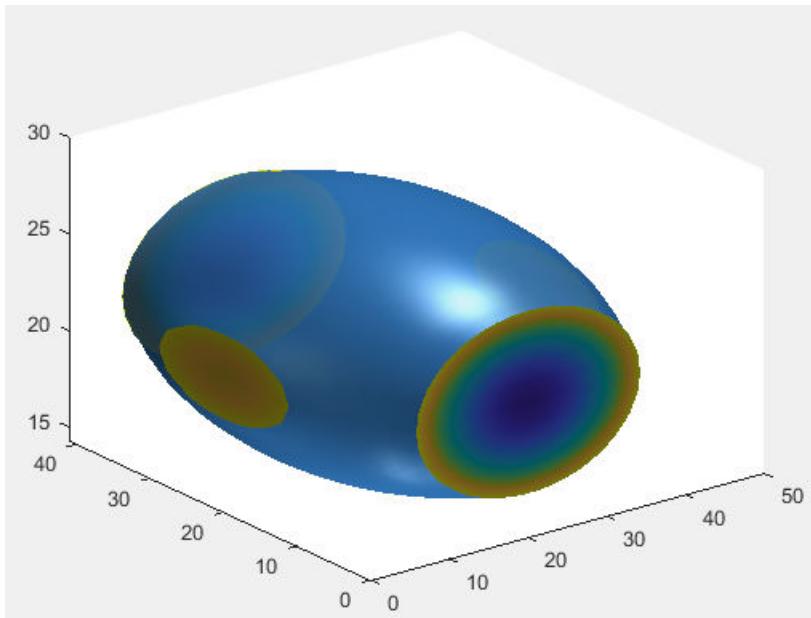
```

IsoSurfCapChart with properties:

```

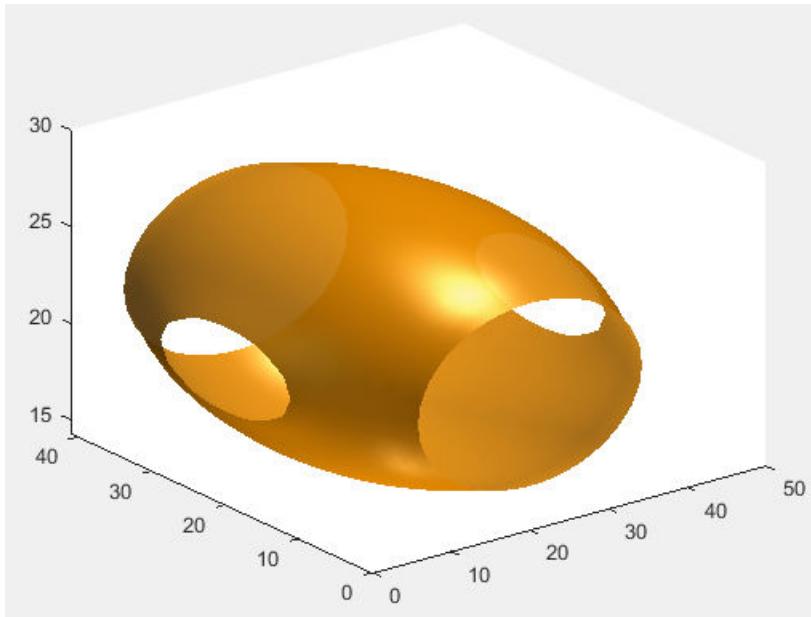
Color: [0.2000 0.5000 0.8000]
IsoValue: 0.5000
Enclose: 'below'
CapVisible: on
WhichCapPlane: 'all'
VolumeData: [41x41x41 double]

```



更改 c 的颜色并隐藏 isocaps。

```
c.Color = [1 0.60 0];  
c.CapVisible = false;
```



另请参阅

类

`matlab.graphics.chartcontainer.ChartContainer`

函数

`isosurface` | `isocaps`

详细信息

- “验证属性值”
- “Customize Property Display”
- “属性 set 方法”
- “图开发概述”（第 23-2 页）

启用便利函数以设置坐标区属性

当您将图开发为 `ChartContainer` 类的子类时，请考虑启用一些 MATLAB 便利函数来设置坐标区上的属性。例如，您可以将您的类设计为支持 `title` 函数。通过启用便利函数，您可以提供与 MATLAB 绘图函数一致的用户体验。

支持不同类型的属性

启用便利函数的方式取决于该函数是控制非计算属性还是计算属性。下表列出了您可以使用的便利函数。

便利函数	关联的坐标区属性	属性类型
<code>title</code> 、 <code>subtitle</code>	<code>Title</code> 、 <code>Subtitle</code>	非计算
<code>xlabel</code> 、 <code>ylabel</code> 、 <code>zlabel</code>	分别为 <code>XLabel</code> 、 <code>YLabel</code> 和 <code>ZLabel</code>	非计算
<code>xlim</code> 、 <code>ylim</code> 、 <code>zlim</code>	分别为 <code>XLim</code> 、 <code>YLim</code> 和 <code>ZLim</code>	计算
<code>xticks</code> 、 <code>yticks</code> 、 <code>zticks</code>	分别为 <code>XTick</code> 、 <code>YTick</code> 和 <code>ZTick</code>	计算
<code>xticklabels</code> 、 <code>yticklabels</code> 、 <code>zticklabels</code>	分别为 <code>XTickLabel</code> 、 <code>YTickLabel</code> 和 <code>ZTickLabel</code>	计算
<code>view</code>	<code>View</code>	计算

为非计算的属性启用函数

非计算的属性是固定值。除非用户或您的代码显式更改它们，否则它们不会变。

要为非计算的属性启用便利函数，请在类中定义公共属性，该属性存储要控制的坐标区属性值。然后定义公共方法，该方法与您要支持的便利函数具有相同的名称并支持相同的调用语法。向设置属性值的方法中添加一行代码。例如，假设有一个类，该类包含一个名为 `TitleText` 的公共属性来存储标题。以下代码显示该类的 `title` 方法。

```
function title(obj,txt)
    obj.TitleText = txt;
end
```

接下来，向 `update` 方法添加一行代码，该方法调用 MATLAB 便利函数来设置对应的坐标区属性。

```
title(getAxes(obj),obj.TitleText);
```

执行上述步骤并保存类文件后，您可以创建图实例，并调用 `title` 函数以显示标题。这样做会触发以下调用序列：

- 1 该类的 `title` 方法设置 `TitleText` 属性，该属性标记要更新的图。
- 2 下次 `drawnow` 执行时，`update` 方法会执行并对坐标区调用 `title` 函数。
- 3 `title` 函数更新坐标区上的 `Title` 属性。

为计算的属性启用函数

计算的属性由坐标区控制。坐标区根据坐标区的内容和基础数据重新计算其值。

要为计算的属性启用便利函数，请定义一个方法，该方法与要启用的便利函数具有相同的名称和调用语法。在该方法中，调用便利函数并将坐标区指定为第一个参数。例如，要启用 `xlim` 函数，请在您的类中

定义名为 `xlim` 的方法。由于 `xlim` 函数接受可变数量的输入参数，您必须指定 `varargin` 作为第二个输入参数。`xlim` 函数还支持可变数量的输出参数，因此您必须指定 `[varargout{1:nargout}]` 来支持这些参数。

```
function varargout = xlim(obj,varargin)
    ax = getAxes(obj);
    [varargout{1:nargout}] = xlim(ax,varargin{:});
end
```

要提供对您的图上对应属性值的访问，请在您的类上定义两个从属属性。第一个属性提供对便利函数控制的值的访问。另一个属性提供对模式属性的访问，该属性指示如何控制第一个属性。模式属性的值可以是 '`auto`' 或 '`manual`'。将这些属性定义为从属属性，以便图不存储这些值。坐标区控制并存储这些值。例如，要提供对坐标区上 `XLim` 和 `XLimMode` 属性的访问，请定义一对从属属性，称为 `XLimits` 和 `XLimitsMode`。

```
properties (Dependent)
    XLimits (1,2) double
    XLimitsMode {mustBeMember(XLimitsMode,['auto','manual'])}
end
```

接下来，为每个从属属性定义 `set` 和 `get` 方法。在每个方法中，设置对应的坐标区属性。以下代码显示 `XLimits` 和 `XLimitsMode` 属性的 `set` 方法和 `get` 方法。

```
function set.XLimits(obj,xlm)
    ax = getAxes(obj);
    ax.XLim = xlm;
end
function xlm = get.XLimits(obj)
    ax = getAxes(obj);
    xlm = ax.XLim;
end
function set.XLimitsMode(obj,xlmmode)
    ax = getAxes(obj);
    ax.XLimMode = xlmmode;
end
function xlmmode = get.XLimitsMode(obj)
    ax = getAxes(obj);
    xlmmode = ax.XLimMode;
end
```

执行上述步骤并保存类文件后，您可以创建图实例，并调用 `xlim` 函数来更改图中的 x 轴范围。会执行 `xlim` 方法，该方法又会调用 `xlim` 函数来更新坐标区上的 `XLim` 属性。

注意 默认情况下，当用户调用 `xlim` 和 `ylim` 函数时，MATLAB 不存储任何更改。要支持在用户保存和加载图时保留这些更改，请参阅“保存和加载图类的实例”（第 23-31 页）。

支持 `title`、`xlim` 和 `ylim` 函数的图类

此示例说明如何定义一个支持 `title`、`xlim` 和 `ylim` 函数的图类。以下代码说明如何：

- 定义一个 `TitleText` 属性并实现一个 `title` 方法，以便图的实例支持 `title` 函数。
- 实现 `xlim` 和 `ylim` 方法，以便图实例支持 `xlim` 和 `ylim` 函数。
- 定义允许用户获取和设置 x 和 y 轴范围的属性。

- 将 Bar 和 ErrorBar 对象合并到单一图中。

要定义该类，请将以下代码复制到编辑器中，并将其以名称 **BarErrorBarChart.m** 保存在可写文件夹中。

```

classdef BarErrorBarChart < matlab.graphics.chartcontainer.ChartContainer
properties
    XData (1,:) double = NaN
    YData (1,:) double = NaN
    EData (1,:) double = NaN
    TitleText (:,:) char = "
end
properties (Dependent)
    % Provide properties to support setting & getting
    XLimits (1,2) double
    XLimitsMode {mustBeMember(XLimitsMode,'auto','manual')}
    YLimits (1,2) double
    YLimitsMode {mustBeMember(YLimitsMode,'auto','manual')}
end
properties (Access = private)
    BarObject (1,1) matlab.graphics.chart.primitive.Bar
    ErrorBarObject (1,1) matlab.graphics.chart.primitive.ErrorBar
end

methods(Access = protected)
    function setup(obj)
        ax = getAxes(obj);
        obj.BarObject = bar(ax,NaN,NaN);
        hold(ax,'on')
        obj.ErrorBarObject = errorbar(ax,NaN,NaN,NaN);
        obj.ErrorBarObject.LineStyle = 'none';
        obj.ErrorBarObject.LineWidth = 2;
        obj.ErrorBarObject.Color = [0.6 0.7 1];
        hold(ax,'off');
    end
    function update(obj)
        % Update Bar and ErrorBar XData and YData
        obj.BarObject.XData = obj.XData;
        obj.BarObject.YData = obj.YData;
        obj.ErrorBarObject.XData = obj.XData;
        obj.ErrorBarObject.YData = obj.YData;

        % Update ErrorBar delta values
        obj.ErrorBarObject.YNegativeDelta = obj.EData;
        obj.ErrorBarObject.YPositiveDelta = obj.EData;

        % Update axes title
        ax = getAxes(obj);
        title(ax,obj.TitleText);
    end
end

methods
    % xlim method
    function varargout = xlim(obj,varargin)
        ax = getAxes(obj);
        [varargout{1:nargout}] = xlim(ax,varargin{:});

```

```

end
% ylim method
function varargout = ylim(obj,varargin)
    ax = getAxes(obj);
    [varargout{1:nargout}] = ylim(ax,varargin{:});
end
% title method
function title(obj,txt)
    obj.TitleText = txt;
end

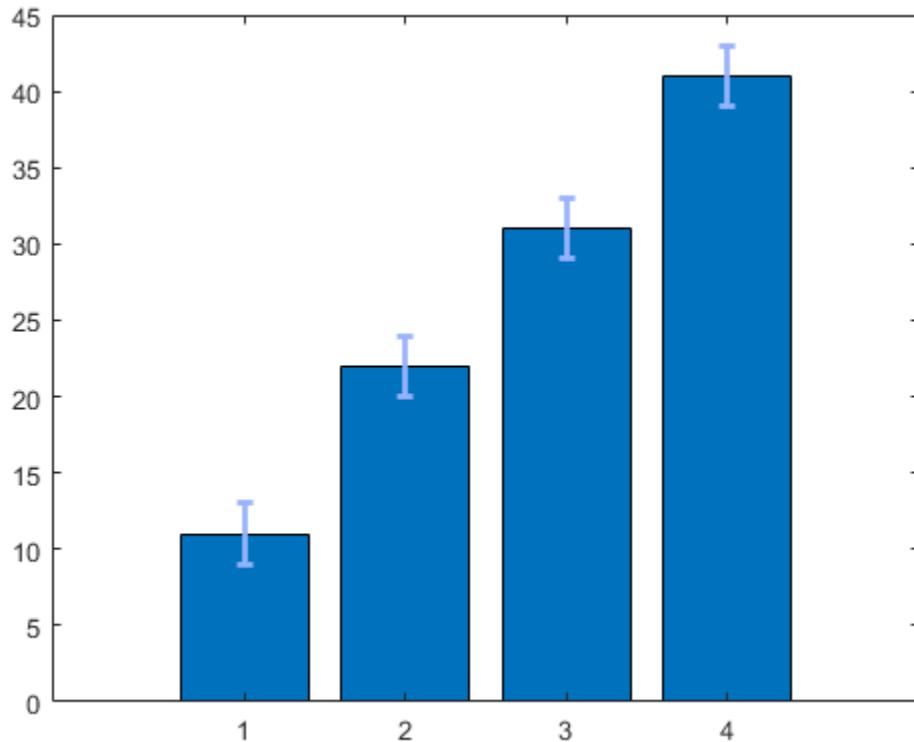
% set and get methods for XLimits and XLimitsMode
function set.XLimits(obj,xlm)
    ax = getAxes(obj);
    ax.XLim = xlm;
end
function xlm = get.XLimits(obj)
    ax = getAxes(obj);
    xlm = ax.XLim;
end
function set.XLimitsMode(obj,xlmmode)
    ax = getAxes(obj);
    ax.XLimMode = xlmmode;
end
function xlmmode = get.XLimitsMode(obj)
    ax = getAxes(obj);
    xlmmode = ax.XLimMode;
end

% set and get methods for YLimits and YLimitsMode
function set.YLimits(obj,ylm)
    ax = getAxes(obj);
    ax.YLim = ylm;
end
function ylm = get.YLimits(obj)
    ax = getAxes(obj);
    ylm = ax.YLim;
end
function set.YLimitsMode(obj,ylmmode)
    ax = getAxes(obj);
    ax.YLimMode = ylmmode;
end
function ylmmode = get.YLimitsMode(obj)
    ax = getAxes(obj);
    ylmmode = ax.YLimMode;
end
end
end

```

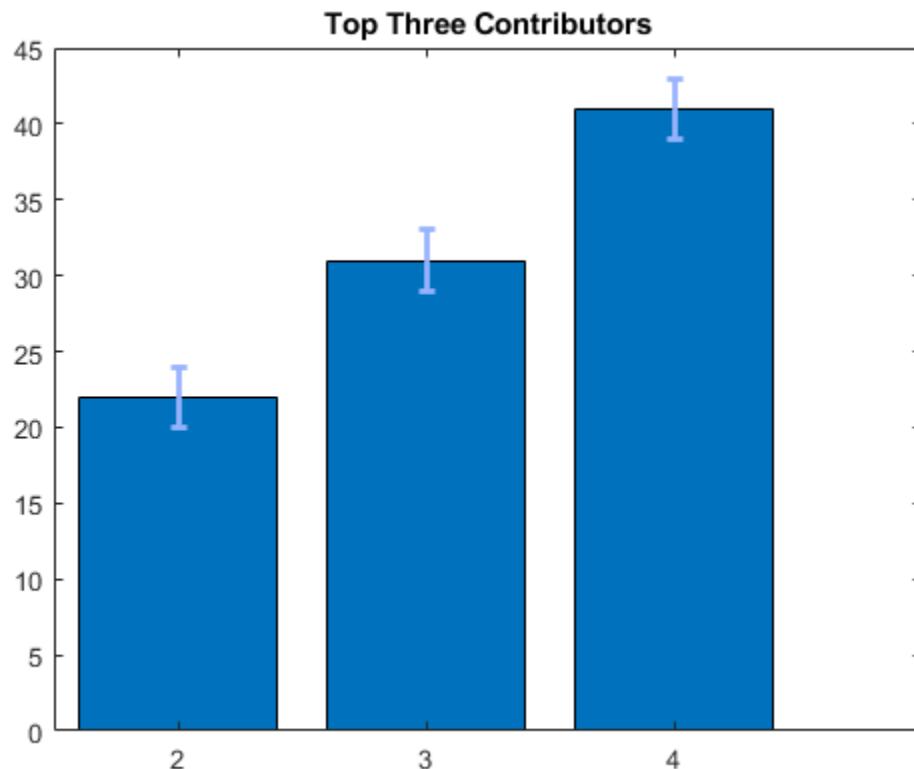
保存 BarErrorBarChart.m 后，创建图实例。

```
BarErrorBarChart('XData',[1 2 3 4],'YData',[11 22 31 41],'EData',[2 2 2 2]);
```



通过调用 `title` 函数指定标题。然后通过调用 `xlim` 函数放大最后三个条形。

```
title('Top Three Contributors')
xlim([1.5 5])
```



另请参阅

类

`matlab.graphics.chartcontainer.ChartContainer`

函数

`bar` | `errorbar` | `title` | `xlim` | `ylim`

属性

`Axes` | `Bar` | `ErrorBar`

详细信息

- “图开发概述” (第 23-2 页)
- “属性 get 方法”
- “属性 set 方法”

保存和加载图类的实例

从 `ChartContainer` 基类继承的图遵循与其他 MATLAB 对象相同的保存和加载规则。但是，在某些情况下，您可能希望您的对象保存和加载其他信息。例如，为了支持保存和加载交互式更改（如旋转或缩放）的结果，必须将坐标区的修改视图存储在类的属性中。通过定义存储和检索这些类型的更改的属性和方法，用户可以保存和重新加载图的实例，同时保留其更改。

保存和加载坐标区更改的编码模式

内置坐标区交互会更改坐标区上的某些属性。例如，拖动以旋转三维图会更改 `View` 属性。同样，在图中滚动缩放会更改坐标区上的 `XLim`、`YLim` 以及可能包含的 `ZLim` 属性。要在用户保存和重新加载图时保留更改，请将这些组件添加到您的类中：

- 定义用于存储图状态的受保护属性（第 23-31 页） - 该属性提供当 MATLAB 保存图对象时用来存储坐标区更改的位置。例如，您可以将该属性命名为 `ChartState`。
- 定义用于检索图状态的 `get` 方法（第 23-31 页） - 根据 MATLAB 是保存还是加载图对象，该方法执行以下两项操作之一。当 MATLAB 保存图对象时，该方法会返回相关坐标区更改，以便保存它们。当 MATLAB 加载图对象时，该方法返回存储在 `ChartState` 属性中的坐标区更改。
- 定义用于更新坐标区的受保护方法（第 23-32 页） - 当图对象加载到 MATLAB 中时，此方法对 `ChartState` 属性调用 `get` 方法，然后更新图的相关坐标区属性。

定义用于存储图状态的受保护属性

定义一个受保护属性来存储相关坐标区信息。此属性为空，除非 MATLAB 在保存过程中设置其值，或 MATLAB 加载图的已保存实例。用易于识别的有意义名称定义属性。例如，定义一个名为 `ChartState` 的属性。

```
properties (Access = protected)
    ChartState = []
end
```

定义检索图状态的 `get` 方法

为 `ChartState` 属性定义公共 `get` 方法。像所有 `set` 和 `get` 方法一样，此方法自动继承 `ChartState` 属性的访问权限。MATLAB 在保存图实例时调用此方法。

在此方法中，创建一个名为 `isLoadedStateAvailable` 的变量，该变量存储一个 `logical` 值。当 `ChartState` 属性不为空时，该值为 `true`。

接下来，编写一个条件语句来检查 `isLoadedStateAvailable` 的值。将语句分成下列子句：

- `if...then` 子句 - `isLoadedStateAvailable` 值为 `true`。返回 `ChartState` 属性的内容。
- `else` 子句 - `isLoadedStateAvailable` 值为 `false`。创建一个结构体并获取坐标区对象。仅当坐标区上的 `XLim`、`YLim` 和 `ZLim` 属性发生变化时，才能将 `XLim`、`YLim` 和 `ZLim` 字段添加到结构体中。要测试坐标区属性是否发生了更改，请检查对应的模式属性是否设置为 '`manual`'。由于没有与坐标区 `View` 属性相关联的模式属性，因此无需检查任何内容即可将 `View` 字段添加到结构体中。

```
methods
    function data = get.ChartState(obj)
        isLoadedStateAvailable = ~isempty(obj.ChartState);
```

```

if isLoadedStateAvailable
    data = obj.ChartState;
else
    data = struct;
    ax = getAxes(obj);

    % Get axis limits only if mode is manual.
    if strcmp(ax.XLimMode,'manual')
        data.XLim = ax.XLim;
    end
    if strcmp(ax.YLimMode,'manual')
        data.YLim = ax.YLim;
    end
    if strcmp(ax.ZLimMode,'manual')
        data.ZLim = ax.ZLim;
    end

    % No ViewMode to check. Store the view anyway.
    data.View = ax.View;
end
end
end

```

定义更新坐标区的受保护方法

定义名为 `loadstate` 的受保护方法。在此方法中，执行以下步骤：

- 查询 `ChartState` 属性，并将返回值存储为 `data`。
- 在更新坐标区上的对应属性之前，请检查是否存在 `XLim`、`YLim`、`ZLim` 和 `View` 字段。
- 清除 `ChartState` 属性的内容。

创建此方法后，在 `setup` 方法的末尾（创建构成图的图形对象后）调用它。当 MATLAB 创建图的新实例或加载图实例时，将执行 `setup` 方法。

```

function loadstate(obj)
    data=obj.ChartState;
    ax = getAxes(obj);

    % Look for states that changed
    if isfield(data, 'XLim')
        ax.XLim=data.XLim;
    end
    if isfield(data, 'YLim')
        ax.YLim=data.YLim;
    end
    if isfield(data, 'ZLim')
        ax.ZLim=data.ZLim;
    end
    if isfield(data, 'View')
        ax.View=data.View;
    end

    % Reset ChartState to empty
    obj.ChartState=[];
end

```

示例：用于存储轴范围和视图的三维绘图

定义一个 `MeshGradientChart` 类，该类显示网格点处有 x 和 y 梯度向量的网格图。将此类设计为当用户保存和重新加载图实例时，坐标区的 `XLim`、`YLim`、`ZLim` 和 `View` 属性会被保留。

要定义此类，请在位于 MATLAB 路径上的文件夹中创建名为 `MeshGradientChart.m` 的程序文件。然后按照表中的步骤实现该类。

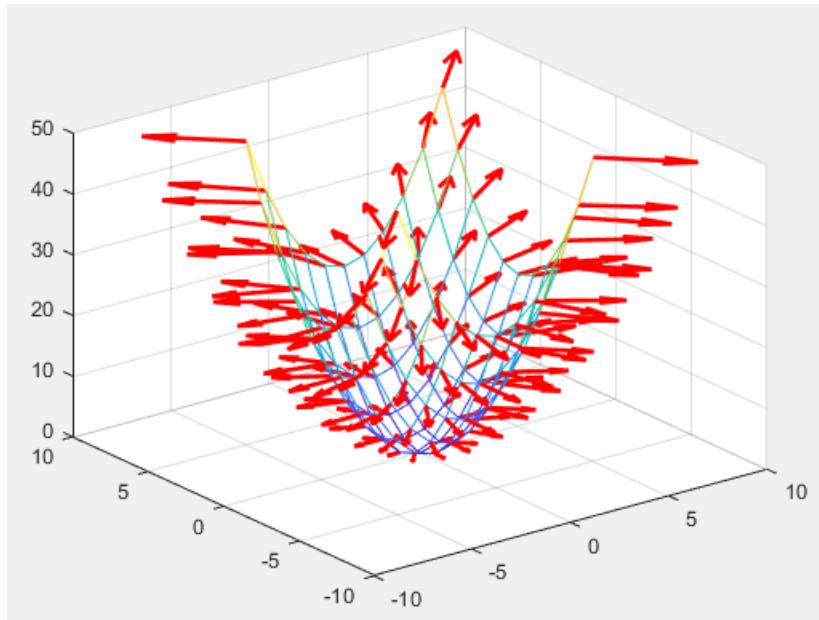
步骤	实现
从 <code>ChartContainer</code> 基类派生而来。	<code>classdef MeshGradientChart < matlab.graphics.chartcontainer.ChartContainer</code>
定义公共属性。	<pre>properties XData (:,:) double = [] YData (:,:) double = [] ZData (:,:) double = [] end</pre>
定义私有属性。一个属性存储 <code>Surface</code> 对象，另一个属性存储 <code>Quiver</code> 对象。	<pre>properties (Access = private,Transient,NonCopyable) SurfaceObject (1,1) matlab.graphics.chart.primitive.Surface QuiverObject (1,1) matlab.graphics.chart.primitive.Quiver end</pre>
定义一个受保护 <code>ChartState</code> 属性以存储坐标区状态。	<pre>properties (Access = protected) ChartState = [] end</pre>
实现 <code>setup</code> 方法。在本例中，调用 <code>mesh</code> 和 <code>quiver3</code> 函数以分别创建 <code>Surface</code> 和 <code>Quiver</code> 对象。将对象存储在对应的属性中，并将坐标区的 <code>hold</code> 状态变为 ' <code>off</code> '。然后调用 <code>loadstate</code> 方法以更新坐标区的状态。	<pre>methods(Access = protected) function setup(obj) ax = getAxes(obj); % Create Mesh and Quiver objects. obj.SurfaceObject=mesh(ax,[],[],[],'FaceColor','none'); hold(ax,'on') obj.QuiverObject=quiver3(ax,[],[],[],[],'Color','r','LineWidth',2); hold(ax,'off') % Load state of the axes. loadstate(obj); end</pre>
实现 <code>update</code> 方法。在本例中，更新网格图和梯度向量尾部的 x 和 y 坐标。然后更新向量的长度和方向。	<pre>function update(obj) % Update Mesh data. obj.SurfaceObject.XData = obj.XData; obj.SurfaceObject.YData = obj.YData; obj.SurfaceObject.ZData = obj.ZData; % Update locations of vector tails. obj.QuiverObject.XData = obj.XData; obj.QuiverObject.YData = obj.YData; obj.QuiverObject.ZData = obj.ZData; % Update lengths and directions of vectors. [gradx,grady] = gradient(obj.ZData); obj.QuiverObject.UData = gradx; obj.QuiverObject.VData = grady; obj.QuiverObject.WData = zeros(size(obj.ZData)); end</pre>

步骤	实现
实现 <code>loadstate</code> 方法，该方法会更新坐标区并将 <code>ChartState</code> 属性重置为空数组。	<pre> function loadstate(obj) data=obj.ChartState; ax = getAxes(obj); % Look for states that changed. if isfield(data, 'XLim') ax.XLim=data.XLim; end if isfield(data, 'YLim') ax.YLim=data.YLim; end if isfield(data, 'ZLim') ax.ZLim=data.ZLim; end if isfield(data, 'View') ax.View=data.View; end % Reset ChartState to empty. obj.ChartState=[]; end end </pre>
实现 <code>ChartState</code> <code>get</code> 方法，该方法返回坐标区状态信息。	<pre> methods function data = get.ChartState(obj) isLoadedStateAvailable = ~isempty(obj.ChartState); % Return ChartState content if loaded state is available. % Otherwise, return current axes state. if isLoadedStateAvailable data = obj.ChartState; else data = struct; ax = getAxes(obj); % Get axis limits only if mode is manual. if strcmp(ax.XLimMode,'manual') data.XLim = ax.XLim; end if strcmp(ax.YLimMode,'manual') data.YLim = ax.YLim; end if strcmp(ax.ZLimMode,'manual') data.ZLim = ax.ZLim; end % No ViewMode to check. Store the view anyway. data.View = ax.View; end end end </pre>

接下来，创建图的一个实例。然后旋转或放大图并保存。当您将图加载回 MATLAB 时，该对象会保留交互式更改。

创建图的实例

```
[X,Y] = meshgrid(-5:5);
Z = X.^2 + Y.^2;
c = MeshGradientChart('XData',X,'YData',Y,'ZData',Z);
```



创建图时：

- **setup** 方法会调用 **loadstate** 方法。
- **loadstate** 方法会执行以下任务，这些任务最终对图对象或底层坐标区对象没有影响。
 - 调用 **get.ChartState** 方法，该方法返回包含坐标区 **View** 属性当前值的结构体。
 - 将坐标区上的 **View** 属性重置为存储在结构体中的值。
 - 清除 **ChartState** 属性的内容。

旋转或放大图并保存。

```
savefig(gcf,'mychart.fig')
```

保存图时，MATLAB 调用 **get.ChartState** 方法，该方法返回包含以下内容的结构体：

- 坐标区上的 **XLim**、**YLim** 或 **ZLim** 属性的值（仅当这些值发生了更改时）
- 坐标区上的 **View** 属性的值

在 MATLAB 检索结构体后，会将该结构体存储在要保存的图对象的 **ChartState** 属性中。

加载保存的图。

```
openfig('mychart.fig')
```

加载图时：

- **setup** 方法调用 **loadstate** 方法。

- `loadstate` 方法执行以下任务：
 - 调用 `get.ChartState` 方法，该方法从 `ChartState` 属性返回结构体。
 - 重置坐标区上的 `XLim`、`YLim`、`ZLim` 和 `View` 属性（仅当结构体包含对应的字段时）。
 - 清除 `ChartState` 属性的内容。

另请参阅

类

`matlab.graphics.chartcontainer.ChartContainer`

函数

`mesh` | `quiver3`

属性

`Axes` | `Chart Surface` | `Quiver`

详细信息

- “对象的保存和加载过程”
- “图开发概述”（第 23-2 页）
- “属性 `get` 方法”

具有自定义属性显示的图类

此示例说明如何定义一个具有自定义属性显示（仅显示部分属性）的图类。以下代码说明如何重载 `matlab.mixin.CustomDisplay` 类的 `getPropertyGroups` 方法。该示例还演示从 `matlab.graphics.chartcontainer.ChartContainer` 基类派生的图的基本编码模式。您可以通过此示例来熟悉图开发的编码方法，或基于此示例来开发您自己的类。

要定义该类，请将以下代码复制到编辑器中，并将其以名称 `SmoothPlotCustomDisplay.m` 保存在可写文件夹中。

```
classdef SmoothPlotCustomDisplay < matlab.graphics.chartcontainer.ChartContainer
    % c = SmoothPlotCustomDisplay('XData',X,'YData',Y,Name,Value,...)
    % plots a dotted line of the coordinates in X and Y with a smoothed
    % version of the line. You can also specify additional name-value
    % arguments, such as 'SmoothColor' and 'SmoothWidth'.
    properties
        XData (1,:) double = NaN
        YData (1,:) double = NaN
        SmoothColor {validatecolor} = [0.9290 0.6940 0.1250]
        SmoothWidth (1,1) double = 2
    end
    properties(Access = private,Transient,NonCopyable)
        OriginalLine (1,1) matlab.graphics.chart.primitive.Line
        SmoothLine (1,1) matlab.graphics.chart.primitive.Line
    end

    methods(Access = protected)
        function setup(obj)
            % Get the axes
            ax = getAxes(obj);

            % Create the original and smooth lines
            obj.OriginalLine = plot(ax,NaN,NaN,'LineStyle',':');
            hold(ax,'on')
            obj.SmoothLine = plot(ax,NaN,NaN,'LineStyle','-',...
                'Color',[0.9290 0.6940 0.1250],'LineWidth',2);
            hold(ax,'off')
        end
        function update(obj)
            % Update line data
            obj.OriginalLine.XData = obj.XData;
            obj.OriginalLine.YData = obj.YData;
            obj.SmoothLine.XData = obj.XData;
            obj.SmoothLine.YData = createSmoothData(obj);

            % Update line color and width
            obj.SmoothLine.Color = obj.SmoothColor;
            obj.SmoothLine.LineWidth = obj.SmoothWidth;
        end
        function propgrp = getPropertyGroups(obj)
            if ~isscalar(obj)
                % List for array of objects
                propgrp = getPropertyGroups@matlab.mixin.CustomDisplay(obj);

            else
                % List for scalar object
            end
        end
    end
end
```

```
propList = {'SmoothColor','XData','YData'};  
propgrp = matlab.mixin.util.PropertyGroup(propList);  
end  
end  
function sm = createSmoothData(obj)  
    % Calculate smoothed data  
    v = ones(1,10)*0.1;  
    sm = conv(obj.YData,v,'same');  
end  
end  
end
```

保存类文件后，可以创建图的实例。如果要在创建图时查看自定义显示，请省略分号。

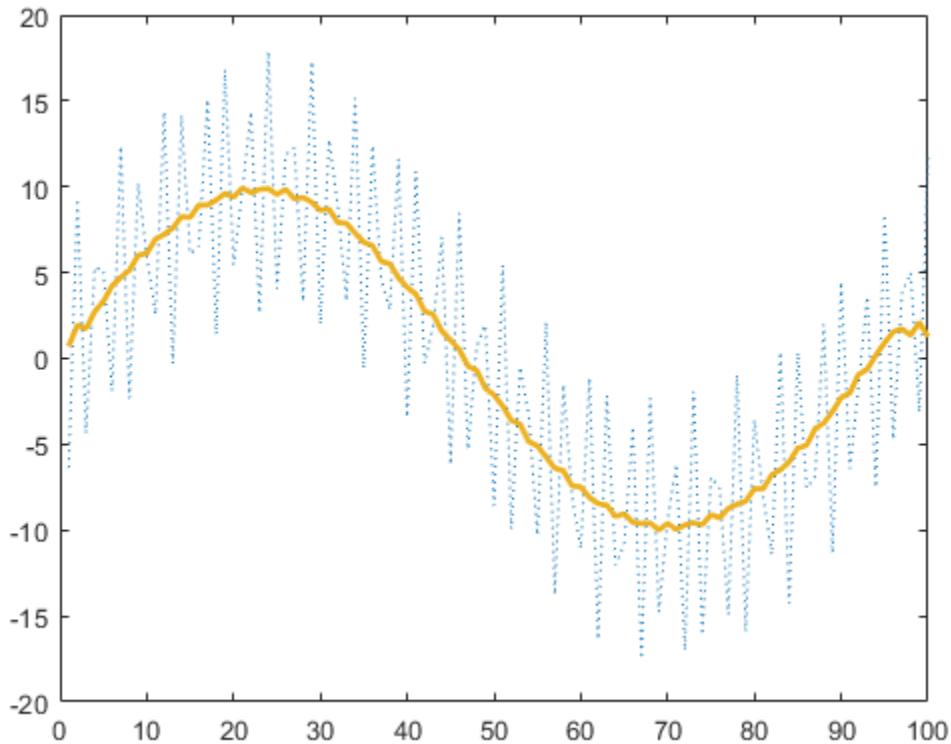
```
x = 1:1:100;  
y = 10*sin(x/15)+8*sin(10*x+0.5);  
c = SmoothPlotCustomDisplay('XData',x,'YData',y)
```

```
c =
```

```
SmoothPlotCustomDisplay with properties:
```

```
SmoothColor: [0.9290 0.6940 0.1250]  
    XData: [1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 ... ]  
    YData: [-6.3714 9.3040 -4.3583 5.3084 5.1309 -1.8987 12.3614 ... ]
```

```
Use GET to show all properties
```



另请参阅

类

`matlab.graphics.chartcontainer.ChartContainer`

详细信息

- “Customize Property Display”
- “管理图类的属性” (第 23-17 页)

具有可变数量的线条的图类

此示例说明如何定义一个图类，该图类可以根据用户数据的大小显示任意数量的线条。图显示的线条数与 **YData** 矩阵中的列数一样多。对于每个线条，图计算局部极值，并用圆形标记指示其位置。以下代码说明如何：

- 定义两个属性，名称分别为 **PlotLineArray** 和 **ExtremaArray**，分别存储线条和标记的对象。
- 实现 **update** 方法，该方法用新对象替换 **PlotLineArray** 和 **ExtremaArray** 属性的内容。由于此方法执行所有绘图和配置命令，因此 **setup** 方法为空。这是创建任意数量线条的简单方法。要了解如何通过重用现有线条对象来更高效地创建此图，请参阅“用于显示不定数量的线条的优化图类”（第 23-43 页）。

要定义该类，请将以下代码复制到编辑器中，并将其以名称 **LocalExtremaChart.m** 保存在可写文件夹中。

```
classdef LocalExtremaChart < matlab.graphics.chartcontainer.ChartContainer
    % c = LocalExtremaChart('XData',X,'YData',Y,Name,Value,...)
    % plots one line with markers at local extrema for every column of matrix Y.
    % You can also specify the additonal name-value arguments, 'MarkerColor'
    % and 'MarkerSize'.

    properties
        XData (1,:) double = NaN
        YData (:,:) double = NaN
        MarkerColor {validatecolor} = [1 0 0]
        MarkerSize (1,1) double = 5
    end
    properties(Access = private,Transient,NonCopyable)
        PlotLineArray (:,1) matlab.graphics.chart.primitive.Line
        ExtremaArray (:,1) matlab.graphics.chart.primitive.Line
    end

    methods(Access = protected)
        function setup(~)
        end
        function update(obj)
            % get the axes
            ax = getAxes(obj);

            % Plot Lines and the local extrema
            obj.PlotLineArray = plot(ax,obj.XData,obj.YData);
            hold(ax,'on')

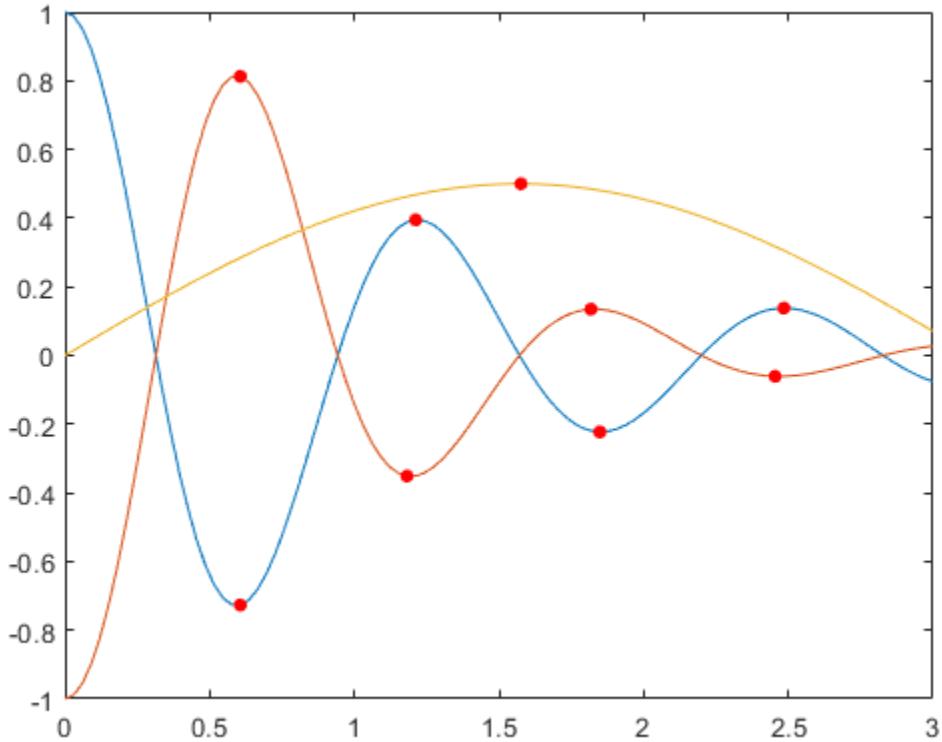
            % Replicate x-coordinate vectors to match size of YData
            newx = repmat(obj.XData(:,1),size(obj.YData,2));

            % Find local minima and maxima and plot markers
            tfmin = islocalmin(obj.YData,1);
            tfmax = islocalmax(obj.YData,1);
            obj.ExtremaArray = plot(ax,newx(tfmin),obj.YData(tfmin),'o',...
                'MarkerEdgeColor','none',...
                'MarkerFaceColor',obj.MarkerColor,...
                'MarkerSize',obj.MarkerSize);
            hold(ax,'off')
        end
    end
end
```

```
    end  
end  
end
```

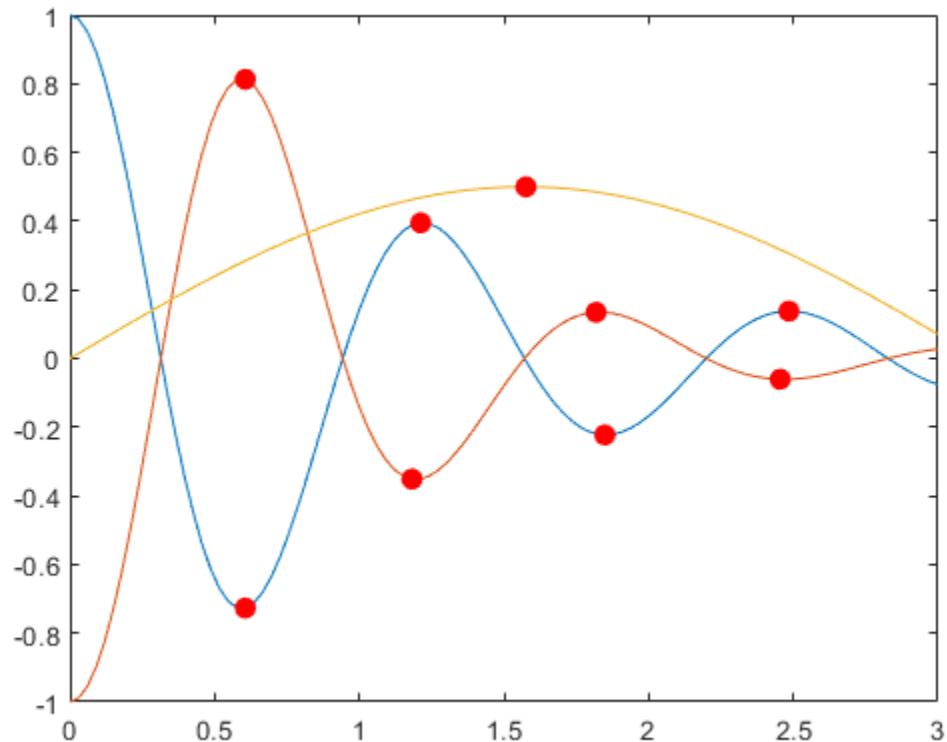
保存类文件后，可以创建图的实例。例如：

```
x = linspace(0,3);  
y1 = cos(5*x)./(1+x.^2);  
y2 = -cos(5*x)./(1+x.^3);  
y3 = sin(x)./2;  
y = [y1' y2' y3'];  
c = LocalExtremaChart('XData',x,'YData',y);
```



将标记大小更改为 8。

```
c.MarkerSize = 8;
```



另请参阅

类

`matlab.graphics.chartcontainer.ChartContainer`

详细信息

- “图开发概述” (第 23-2 页)
- “用于显示不定数量的线条的优化图类” (第 23-43 页)

用于显示不定数量的线条的优化图类

此示例说明如何优化一个图类以显示不定数量的线条。它重用现有线条对象，这可以提高图的性能，尤其是在线条的数量不经常变化的情况下。有关未经优化的此图的更简单版本，请参阅“具有可变数量的线条的图类”（第 23-40 页）。

图显示的线条数与 YData 矩阵中的列数一样多，在局部极值处有圆形标记。以下代码说明如何：

- 定义两个属性，名称分别为 PlotLineArray 和 ExtremaLine，分别用于存储线条和标记的对象。
- 实现用于初始化 ExtremaLine 对象的 setup 方法。
- 实现用于获取 PlotLineArray 的大小的 update 方法，然后根据 YData 中的列数在该数组中加减对象。

要定义该类，请将以下代码复制到编辑器中，并将其以名称 OptimLocalExtremaChart.m 保存在可写文件夹中。

```
classdef OptimLocalExtremaChart < matlab.graphics.chartcontainer.ChartContainer
    % c = OptimLocalExtremaChart('XData',X,'YData',Y,Name,Value,...)
    % plots one line with markers at local extrema for every column of matrix Y.
    % You can also specify the additional name-value arguments, 'MarkerColor'
    % and 'MarkerSize'.

    properties
        XData (:,1) double = NaN
        YData (:,:) double = NaN
        MarkerColor {validatecolor} = [1 0 0]
        MarkerSize (1,1) double = 5
    end
    properties(Access = private,Transient,NonCopyable)
        PlotLineArray (:,1) matlab.graphics.chart.primitive.Line
        ExtremaLine (:,1) matlab.graphics.chart.primitive.Line
    end

    methods(Access = protected)
        function setup(obj)
            obj.ExtremaLine = matlab.graphics.chart.primitive.Line(...%
                'Parent', obj.getAxes(), 'Marker', 'o', ...
                'MarkerEdgeColor', 'none', 'LineStyle', 'none');
        end
        function update(obj)
            % Get the axes
            ax = getAxes(obj);

            % Create extra lines as needed
            p = obj.PlotLineArray;
            nPlotLinesNeeded = size(obj.YData, 2);
            nPlotLinesHave = numel(p);
            for n = nPlotLinesHave+1:nPlotLinesNeeded
                p(n) = matlab.graphics.chart.primitive.Line('Parent', ax, ...
                    'SeriesIndex', n, 'LineWidth', 2);
            end

            % Update the lines
            for n = 1:nPlotLinesNeeded
                p(n).XData = obj.XData;
            end
        end
    end
end
```

```
p(n).YData = obj.YData(:,n);
end

% Delete unneeded lines
delete(p((nPlotLinesNeeded+1):numel(p)))
obj.PlotLineArray = p(1:nPlotLinesNeeded);

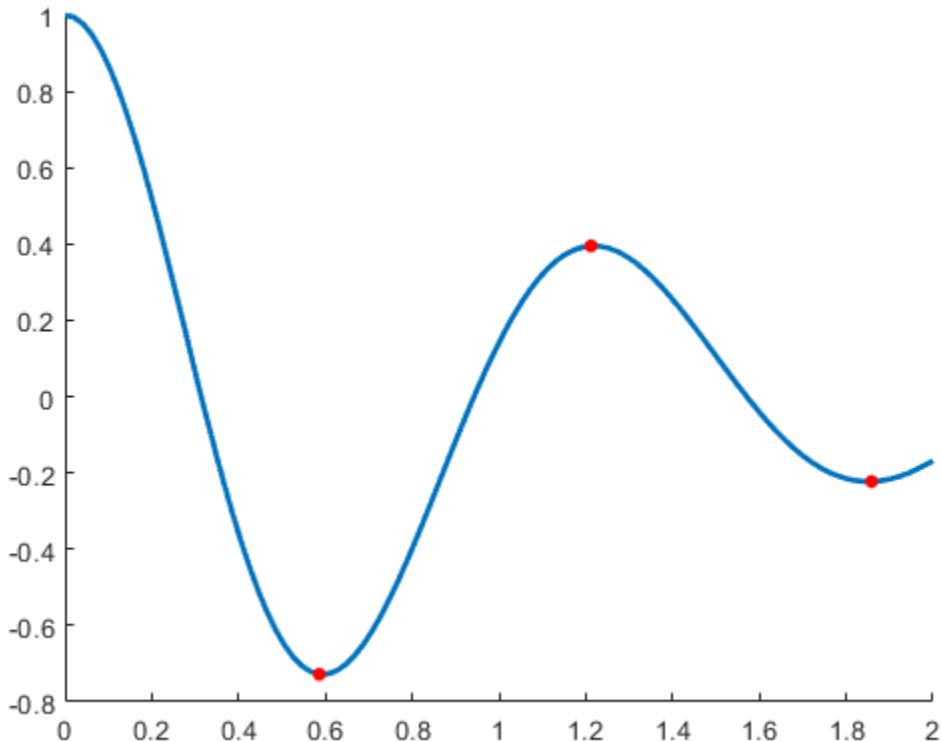
% Replicate x-coordinate vectors to match size of YData
newx = repmat(obj.XData(:,1),size(obj.YData,2));

% Find local minima and maxima and plot markers
tfmin = islocalmin(obj.YData,1);
tfmax = islocalmax(obj.YData,1);
obj.ExtremaLine.XData = [newx(tfmin); newx(tfmax)];
obj.ExtremaLine.YData = [obj.YData(tfmin); obj.YData(tfmax)];
obj.ExtremaLine.MarkerFaceColor = obj.MarkerColor;
obj.ExtremaLine.MarkerSize = obj.MarkerSize;

% Make sure the extrema are on top
uistack(obj.ExtremaLine, 'top');
end
end
end
```

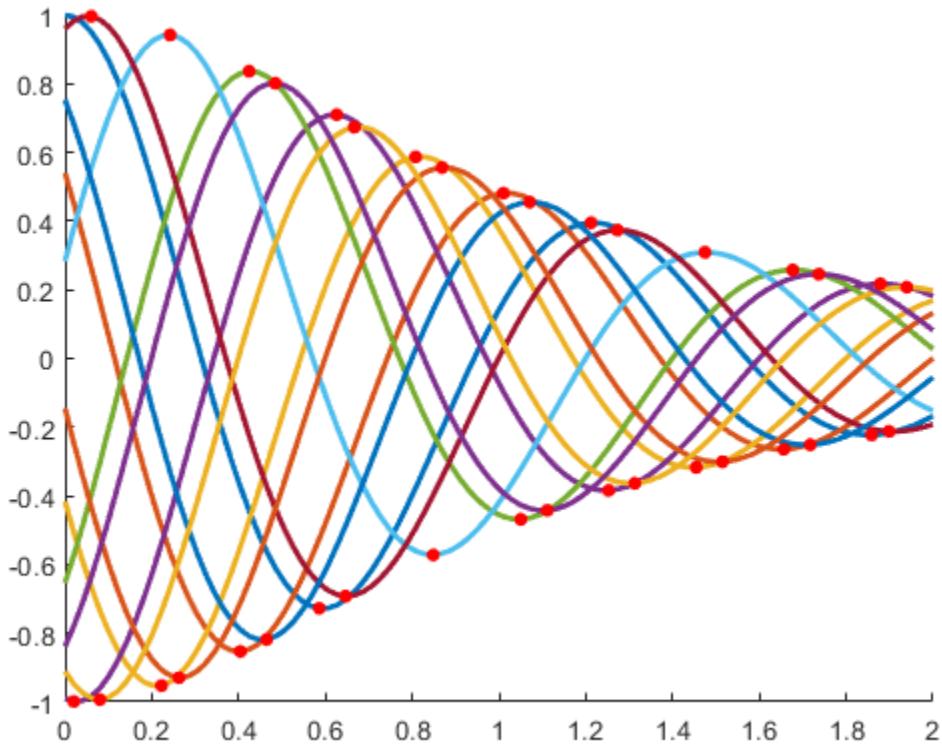
保存类文件后，可以创建图的实例。例如：

```
x = linspace(0,2)';
y = cos(5*x)./(1+x.^2);
c = OptimLocalExtremaChart('XData',x,'YData',y);
```



现在，创建一个 **for** 循环，该循环在每次迭代时向图中添加一个线条。图对象会保留所有现有线条，并为每个 **i** 另外添加一个线条。

```
for i=1:10
    y = cos(5*x+i)./(1+x.^2);
    c.YData = [c.YData y];
end
```



另请参阅

类

`matlab.graphics.chartcontainer.ChartContainer`

相关示例

- “具有可变数量的线条的图类” (第 23-40 页)
- “管理图类的属性” (第 23-17 页)

用于显示可变大小绘图分块的图类

以下示例说明了如何定义一个类，用于创建可根据用户数据大小而定的任意大小的绘图分块。该图具有一个公共属性 **Data**，该属性接受 $m \times n$ 矩阵。该图显示了散点图和直方图的 $n \times n$ 方形分块。散点图显示了相对彼此而绘制的不同数据列。直方图显示了每个数据列中的值分布。

此类中的 **update** 方法会重新创建直方图和散点图，以反映数据中的变化。如果布局的网格大小与数据大小冲突，则删除所有坐标区，并更新 **GridSize** 属性以匹配数据大小。然后创建一组新的坐标区对象。

要定义该类，请将以下代码复制到编辑器中，并将其以名称 **TrellisChart.m** 保存在可写文件夹中。

```
classdef TrellisChart < matlab.graphics.chartcontainer.ChartContainer

    properties
        Data(:, :) {mustBeNumeric}
        ColNames(1, :) string
        TitleText(1, :) string
    end

    methods (Access = protected)
        function setup(obj)
            % Use one toolbar for all of the axes
            axtoolbar(getLayout(obj), 'default');
        end

        function update(obj)
            % Get the layout and store it as tcl
            tcl = getLayout(obj);
            numvars = size(obj.Data, 2);

            % Reconfigure layout if needed
            if numvars ~= tcl.GridSize(1)
                % Delete layout contents to change the grid size
                delete(tcl.Children);
                if numvars > 0
                    tcl.GridSize = [numvars numvars];
                    for i = 1:numvars^2
                        nexttile(tcl, i);
                    end
                end
            end
        end

        % Populate the layout with the axes
        ax = gobjects(numvars, numvars);
        for col = 1:numvars
            for row = 1:numvars
                % Get the axes at the current row/column
                t = col + (row - 1) * numvars;
                ax(row, col) = nexttile(tcl, t);
                if col == row
                    % On the diagonal, draw histograms
                    histogram(ax(row, col), obj.Data(:, col));
                    ylabel(ax(row, col), 'Count')
                else
                    % Off the diagonal, draw scatters
                    scatter(ax(row, col), obj.Data(:, col), ...
                end
            end
        end
    end
end
```

```
    obj.Data(:,row),'filled','MarkerFaceAlpha',0.6)
    if length(obj.ColNames) >= row
        ylabel(ax(row,col),obj.ColNames(row));
    end
end

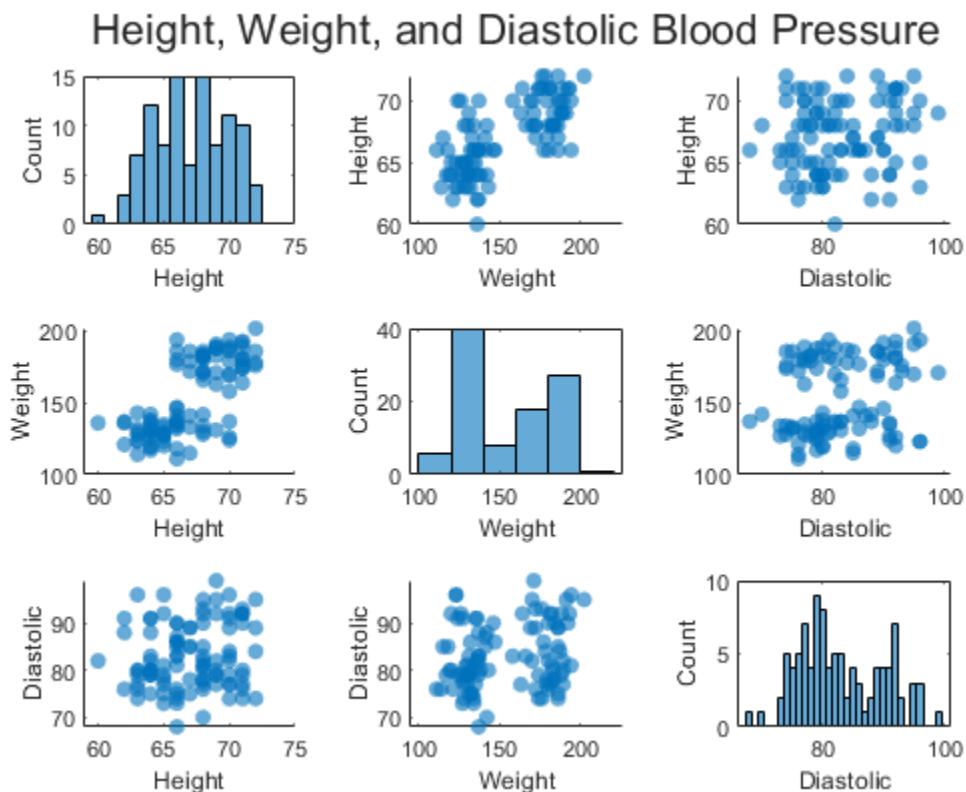
if length(obj.ColNames) >= col
    xlabel(ax(row,col),obj.ColNames(col));
end
end

% Link the x-axis for each column, so that panning or zooming
% affects all axes in the column.
linkaxes(ax(:,col),'x')
end

% Chart title
title(tcl,obj.TitleText,'FontSize',16);
end
end
end
```

保存类文件后，创建图实例。

```
load patients
chartTitle = "Height, Weight, and Diastolic Blood Pressure";
c = TrellisChart('Data',[Height Weight Diastolic], ...
    'colNames', ["Height" "Weight" "Diastolic"],...
    'TitleText',chartTitle);
```



另请参阅

函数

`getLayout`

类

`matlab.graphics.chartcontainer.ChartContainer`

属性

`TiledChartLayout` Properties

详细信息

- “创建包含极坐标区、地理坐标区或多个坐标区的图”（第 23-13 页）

包含两个交互式绘图的图类

以下示例说明如何使用两个具有交互式功能的坐标区来定义一个类，以用于可视化时间表数据。顶部坐标区启用了沿 x 维度的平移和缩放，以便用户检查所关注的区域。底部坐标区在整个时间范围内显示绘图。底部坐标区还显示了一个浅蓝色的时间窗，指示顶部坐标区中的时间范围。该类定义了以下属性、方法和局部函数。

属性：

- **Data** - 公共从属属性，用于存储时间表。
- **TimeLimits** - 公共属性，用于设置顶部坐标区的范围以及底部坐标区中时间窗的宽度。
- **SavedData** - 受保护属性，用户可以用它来保存和加载图实例并保留数据。
- **TopAxes** 和 **BottomAxes** - 私有属性，用于存储坐标区对象。
- **TopLine** 和 **BottomLine** - 私有属性，用于存储线对象。
- **TimeWindow** - 底部坐标区中显示的补片对象，用于指示顶部坐标区的时间范围。

方法：

- **set.Data** 和 **get.Data** - 使用户能够保存和加载图实例并保留数据。
- **setup** - 在创建图时执行一次。它会配置布局和坐标区、线对象以及补片对象。
- **update** - 在 **setup** 方法之后以及用户更改图上一个或多个属性之后执行。
- **panZoom** - 当用户在顶部坐标区内平移或缩放时，更新图的时间范围。这会使时间窗更新，以反映新的范围。
- **click** - 当用户点击底部坐标区时，重新计算时间范围。

局部函数：

- **updateDataTipTemplate** - 从 **update** 方法内部调用。它在数据提示中创建与时间表中的变量相对应的行。
- **mustHaveOneNumericVariable** - 验证 **Data** 属性。此函数可确保用户指定的时间表至少具有一个数值变量。

要定义该类，请将以下代码复制到编辑器中，并将其以名称 **TimeTableChart.m** 保存在可写文件夹中。

```
classdef TimeTableChart < matlab.graphics.chartcontainer.ChartContainer
    properties (Dependent)
        Data timetable {mustHaveOneNumericVariable} = ...
            timetable(datetime.empty(0,1),zeros(0,1))
    end

    properties
        TimeLimits (1,2) datetime = [NaT NaT]
    end

    properties (Access = protected)
        SavedData timetable = timetable(datetime.empty(0,1),zeros(0,1))
    end

    properties (Access = private, Transient, NonCopyable)
        TopAxes matlab.graphics.axis.Axes
    end

```

```

TopLine matlab.graphics.chart.primitive.Line
BottomAxes matlab.graphics.axis.Axes
BottomLine matlab.graphics.chart.primitive.Line
TimeWindow matlab.graphics.primitive.Patch
end

methods
function set.Data(obj, tbl)
    % Reset the time limits if the row times have changed.
    oldTimes = obj.SavedData.Properties.RowTimes;
    newTimes = tbl.Properties.RowTimes;
    if ~isequal(oldTimes, newTimes)
        obj.TimeLimits = [NaN NaN];
    end

    % Store the new table.
    obj.SavedData = tbl;
end

function tbl = get.Data(obj)
    tbl = obj.SavedData;
end
end

methods (Access = protected)
function setup(obj)
    % Create two axes. The top axes is 3x taller than bottom axes.
    tcl = getLayout(obj);
    tcl.GridSize = [4 1];
    obj.TopAxes = nexttile(tcl, 1, [3 1]);
    obj.BottomAxes = nexttile(tcl, 4);

    % Add a shared toolbar on the layout, which removes the
    % toolbar from the individual axes.
    axtoolbar(tcl, 'default');

    % Create one line to show the zoomed-in data.
    obj.TopLine = plot(obj.TopAxes, NaN, NaN);

    % Create one line to show an overview of the data, and disable
    % HitTest so the ButtonDownFcn on the bottom axes works.
    obj.BottomLine = plot(obj.BottomAxes, NaN, NaN, ...
        'HitTest', 'off');

    % Create a patch to show the current time limits.
    obj.TimeWindow = patch(obj.BottomAxes, ...
        'Faces', 1:4, ...
        'Vertices', NaN(4,2), ...
        'FaceColor', obj.TopLine.Color, ...
        'FaceAlpha', 0.3, ...
        'EdgeColor', 'none', ...
        'HitTest', 'off');

    % Constrain axes panning/zooming to only the X-dimension.
    obj.TopAxes.Interactions = [ ...
        dataTipInteraction;
        panInteraction('Dimensions','x');
        rulerPanInteraction('Dimensions','x');

```

```
zoomInteraction('Dimensions','x')];  
  
% Disable pan/zoom on the bottom axes.  
obj.BottomAxes.Interactions = [];  
  
% Add a listener to XLim to respond to zoom events.  
addlistener(obj.TopAxes, 'XLim', 'PostSet', @(~, ~) panZoom(obj));  
  
% Add a callback for clicks on the bottom axes.  
obj.BottomAxes.ButtonDownFcn = @(~, ~) click(obj);  
end  
  
function update(obj)  
    % Extract the time data from the table.  
    tbl = obj.Data;  
    t = tbl.Properties.RowTimes;  
  
    % Extract the numeric variables from the table.  
    S = vartype('numeric');  
    numericTbl = tbl(:,S);  
  
    % Update the data on both lines.  
    set([obj.BottomLine obj.TopLine], 'XData', t, 'YData', numericTbl(:,1));  
  
    % Create a dataTipTextRow for each variable in the timetable.  
    updateDataTipTemplate(obj.TopLine, tbl)  
  
    % Update the top axes limits.  
    obj.TopAxes.YLimMode = 'auto';  
    if obj.TimeLimits(1) < obj.TimeLimits(2)  
        obj.TopAxes.XLim = obj.TimeLimits;  
    else  
        % Current time limits are invalid, so set XLimMode to auto and  
        % let the axes calculate limits based on available data.  
        obj.TopAxes.XLimMode = 'auto';  
        obj.TimeLimits = obj.TopAxes.XLim;  
    end  
  
    % Update time window to reflect the new time limits.  
    xLimits = ruler2num(obj.TimeLimits, obj.BottomAxes.XAxis);  
    yLimits = obj.BottomAxes.YLim;  
    obj.TimeWindow.Vertices = [xLimits([1 1 2 2]); yLimits([1 2 2 1])];  
end  
  
function panZoom(obj)  
    % When XLim on the top axes changes, update the time limits.  
    obj.TimeLimits = obj.TopAxes.XLim;  
end  
  
function click(obj)  
    % When clicking on the bottom axes, recenter the time limits.  
  
    % Find the center of the click using CurrentPoint.  
    center = obj.BottomAxes.CurrentPoint(1,1);  
  
    % Convert from numeric units into datetime using num2ruler.  
    center = num2ruler(center, obj.BottomAxes.XAxis);
```

```

% Find the width of the current time limits.
width = diff(obj.TimeLimits);

% Recenter the current time limits.
obj.TimeLimits = center + [-1 1]*width/2;
end
end
end

function updateDataTipTemplate(obj, tbl)

% Create a dataTipTextRow for each variable in the timetable.
timeVariable = tbl.Properties.DimensionNames{1};
rows = dataTipTextRow(timeVariable, tbl.(timeVariable));
for n = 1:numel(tbl.Properties.VariableNames)
    rows(n+1,1) = dataTipTextRow(...,
        tbl.Properties.VariableNames{n},tbl{:,n});
end
obj.DataTipTemplate.DataTipRows = rows;

end

function mustHaveOneNumericVariable(tbl)

% Validation function for Data property.
S = vartype('numeric');
if width(tbl(:,S)) < 1
    error('TimeTableChart:InvalidTable', ...
        'Table must have at least one numeric variable.')
end
end

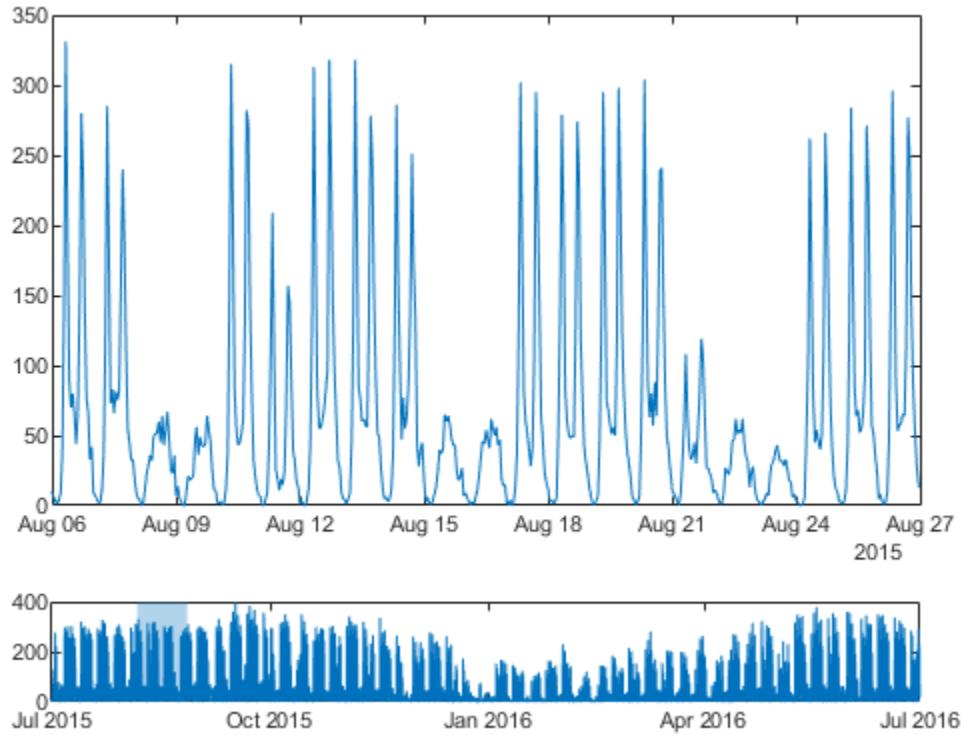
```

保存类文件后，创建图实例。在此例中，请使用该图来查看一年中某几周的自行车交通数据。

```

bikeTbl = readtimetable('BicycleCounts.csv');
bikeTbl = bikeTbl(169:8954,:);
tlimits = [datetime(2015,8,6) datetime(2015,8,27)];
TimeTableChart('Data',bikeTbl,'TimeLimits',tlimits);

```



另请参阅

函数

[getLayout](#) | [timetable](#)

类

[matlab.graphics.chartcontainer.ChartContainer](#)

属性

[TiledChartLayout Properties](#) | [DataTipTemplate Properties](#)

详细信息

- “创建包含极坐标区、地理坐标区或多个坐标区的图” (第 23-13 页)
- “事件和侦听程序概述”
- “创建自定义数据提示” (第 13-6 页)

优化图形程序的性能

- “找到代码瓶颈” (第 24-2 页)
- “什么影响代码执行速度” (第 24-4 页)
- “明智的对象创建” (第 24-5 页)
- “避免重复搜索对象” (第 24-7 页)
- “屏幕更新” (第 24-8 页)
- “优化代码以获取和设置图形属性” (第 24-10 页)
- “避免更新静态数据” (第 24-12 页)
- “高效变换对象” (第 24-13 页)
- “使用低级函数提升速度” (第 24-14 页)
- “图形的系统要求” (第 24-15 页)
- “解决低级图形问题” (第 24-17 页)

找到代码瓶颈

使用探查器确定哪些函数占用最多执行时间。您可以通过尽量减少算法和计算时间来提升性能。

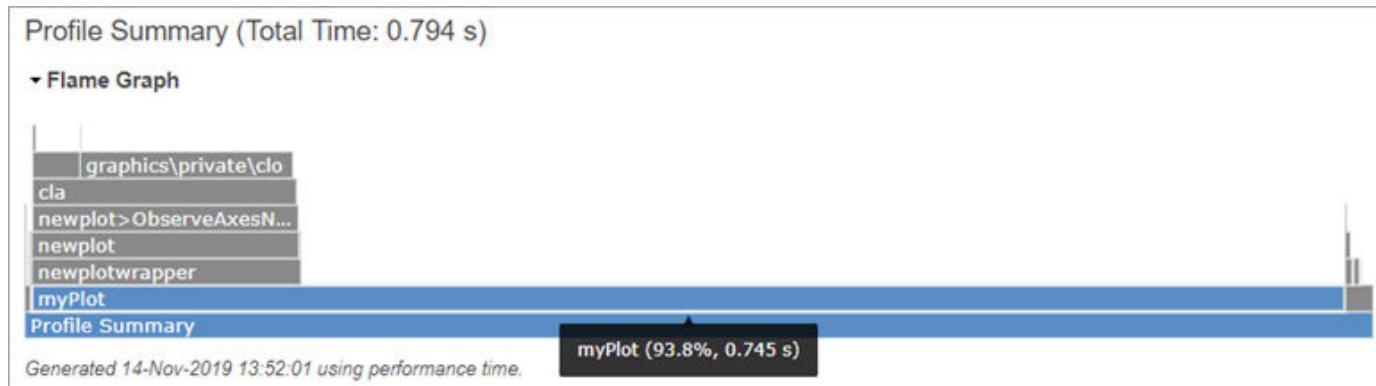
一旦优化了代码，使用以下技术降低对象创建和更新显示的开销量。

例如，假设您使用 `myPlot` 函数绘制一个 10×1000 元素的数组：

```
function myPlot
    x = rand(10,1000);
    y = rand(10,1000);
    plot(x,y,'LineStyle','none','Marker','o','Color','b');
end

profile on
myPlot
profile viewer
```

当您探查此代码时，您发现大多数时间花在 `myPlot` 函数上：

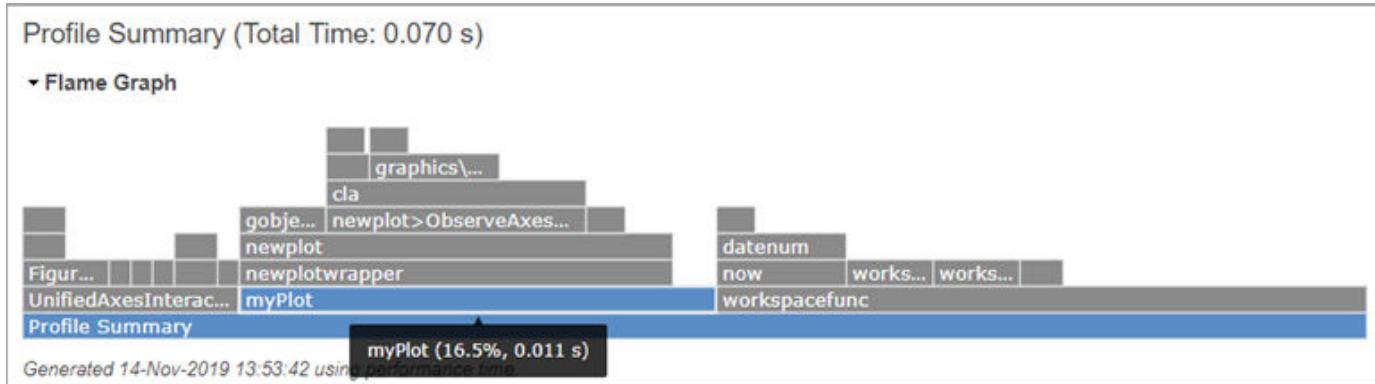


由于 `x` 和 `y` 数组含有 1000 列数据，绘图函数创建 1000 个线条对象。在本例中，您可以通过创建一个含有 10000 个数据点的线条实现同样效果：

```
function myPlot
    x = rand(10,1000);
    y = rand(10,1000);
    % Pass x and y as 1-by-1000 vectors
    plot(x(:,y(:)),'LineStyle','none','Marker','o','Color','b');
end

profile on
myPlot
profile viewer
```

对象创建时间是本例中的主要因素：



您可以通过了解如何从根本上避免或尽量减少耗时的操作，从而在执行速度方面得到提升。有关如何使用该工具提升性能的详细信息，请参阅 **profile** 函数文档。

什么影响代码执行速度

本节内容

“潜在瓶颈” (第 24-4 页)

“如何提升性能” (第 24-4 页)

潜在瓶颈

当处理大量数据或大量对象时，性能会成为瓶颈。在这种情况下，您可以通过尽量减少构成总执行时间的以下两个因素的影响，提高图形代码执行速度：

- 对象创建 - 将新图形对象添加到场景中。
- 屏幕更新 - 更新图形模型并将发送更改以进行渲染。

通过是可以防止这些操作大量占据某一编程模式的总执行时间。考虑执行时间为以下项目总和：

$$T_{\text{执行时间}} = T_{\text{创建对象}} + T_{\text{更新}} + (T_{\text{计算等}})$$

以下示例演示尽量减小对象创建和更新屏幕所耗费时间的方法。在以上表达式中，执行时间不包含实际渲染屏幕的时间。

如何提升性能

探查您的代码并优化算法、计算和其他应用程序具体的瓶颈。然后确定代码是在对象创建函数还是在 `drawnow` (更新) 中花费了更多时间。您可以优化这两项操作，从总时间方程中较大项开始。

您的代码是否：

- 创建新对象而不是更新现有对象？请参阅“明智的对象创建” (第 24-5 页)。
- 更新包含一部分静态数据的对象？请参阅“避免更新静态数据” (第 24-12 页)。
- 搜索对象句柄。请参阅“避免重复搜索对象” (第 24-7 页)。
- 旋转、转换或缩放对象？请参阅“高效变换对象” (第 24-13 页)。
- 在同一循环中查询和设置属性？请参阅“优化代码以获取和设置图形属性” (第 24-10 页)。

明智的对象创建

本节内容

- “对象开销”（第 24-5 页）
- “不要创建不需要的对象”（第 24-5 页）
- “使用 NaN 模拟多个线条”（第 24-5 页）
- “修改数据而不是创建新对象”（第 24-5 页）

对象开销

图形对象是复杂的结构体，它存储信息（数据和对象特征）、监听某些事件发生（回调属性），并能促使其他对象更改以适应其存在（更新坐标区范围等等）。因此，创建对象会耗费资源。

当性能成为重要考虑因素时，需要尝试采用消耗资源最少的方式来实现您的目标。

您可以按照以下原则提升性能：

- 不要创建不需要的对象
- 避免搜索对象层次结构

不要创建不需要的对象

看看您能否创建更少的对象而实现同样的效果。例如，假设您想要绘制 10×1000 点数组，只显示标记。

此代码创建 1000 个线条对象：

```
x = rand(10,1000);
y = rand(10,1000);
plot(x,y,'LineStyle','none','Marker','!','Color','b');
```

将数据从 10×1000 转换为 10000×1 。此代码创建的图形看上去一样，但只创建了一个对象：

```
plot(x(:),y(:),'LineStyle','none','Marker','!','Color','b')
```

使用 NaN 模拟多个线条

如果坐标数据包含 NaN，那么 MATLAB 不会渲染这些点。您可以将 NaN 添加到顶点数据以创建与单独线条看上去一样的线段。将 NaN 放在每个数据向量中同样的元素位置。例如，此代码会创建三个单独线条：

```
x = [0:10,NaN,20:30,NaN,40:50];
y = [0:10,NaN,0:10,NaN,0:10];
line(x,y)
```

修改数据而不是创建新对象

要查看几乎一样的图形上的不同数据，更高效的方法是更新现有对象（线条、文本等）的数据，而不是重新创建整个图形。

例如，假设您想要显示改变某些参数后对数据的影响。

- 1 设置可以预先确定的所有轴的范围，或将坐标轴范围模式设置为 **manual**。
- 2 使用新参数重新计算数据。
- 3 使用新数据更新线条、文本等图形中所用对象的属性。
- 4 调用 **drawnow** 更新图窗（以及图窗中的所有子对象）。

例如，假设您想要当数据变化时更新图形：

```
figure
z = peaks;
h = surf(z);
drawnow
zlim([min(z(:)), max(z(:))]);
for k = 1:50
    h.ZData = (0.01+sin(2*pi*k/20)*z);
    drawnow
end
```

避免重复搜索对象

当您搜索句柄时，MATLAB 必须搜索对象层次结构以找到匹配句柄，这非常耗时。保存您以后要访问的句柄是更快的方法。数组索引通常比使用 `findobj` 或 `findall` 更快。

此对象创建 500 个线条对象然后在循环中调用 `findobj`。

```
figure
ax = axes;
for ix=1:500
    line(rand(1,5),rand(1,5),'Tag',num2str(ix),'Parent',ax);
end
drawnow;
for ix=1:500
    h = findobj(ax,'Tag',num2str(ix));
    set(h,'Color',rand(1,3));
end
drawnow;
```

更好的方法是将句柄保存到数组中，并在第二个 `for` 循环中为数组建立索引。

```
figure
ax = axes;
h = gobjects(1,500);
for ix = 1:500
    h(ix) = line(rand(1,5),rand(1,5),'Tag',num2str(ix),'Parent',ax);
end
drawnow;
% Index into handle array
for ix=1:500
    set(h(ix),'Color',rand(1,3));
end
drawnow
```

限制搜索范围

如果搜索句柄是必需的，那么通过指定对象树的起始点来限制被搜索对象的范围。例如，将起始点指定为包含要搜索的对象的图窗或坐标区。

另一种限制搜索对象花费时间的方法是限制搜索的深度。例如，`'flat'` 搜索可以将搜索限制为特定句柄数组中的对象。

使用 `findobj` 和 `findall` 函数搜索句柄。

有关详细信息，请参阅“查找对象”（第 18-4 页）。

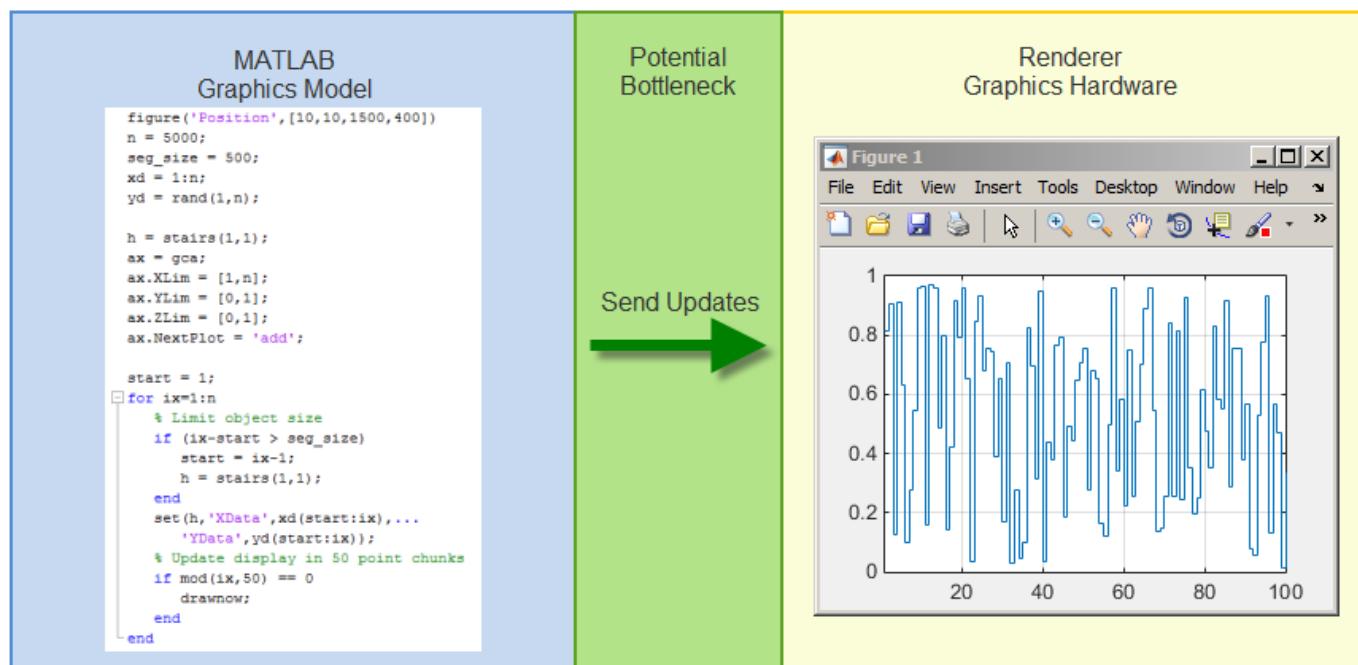
屏幕更新

本节内容

- “MATLAB 图形系统”（第 24-8 页）
- “管理更新”（第 24-8 页）

MATLAB 图形系统

MATLAB 图形是通过执行多个线程实现的。下图演示了主线程和渲染器线程在更新过程中如何交互。MATLAB 端包含图形模型，它描述了由图形硬件渲染的几何图。渲染器端在其内存系统中含有几何图。图形硬件可以渲染屏幕而不会阻止 MATLAB 执行。



当图形模型改变时，这些更新必须传递到图形硬件。发送更新可能是瓶颈，因为图形硬件并非支持所有的 MATLAB 数据类型。更新过程必须将数据转换到正确形式。

当几何图已在图形硬件内存中时，可以通过使用该数据并尽量减少更新的数据传送以实现性能优势。

管理更新

更新包含以下步骤：

- 收集更新到屏幕的变更，如属性变化和添加的对象。
- 更新在图形模型中的依赖关系。
- 将这些更新发送到渲染器。
- 等待渲染器在返回执行 MATLAB 之前接受这些参数。

通过调用 **drawnow** 函数启动更新。当渲染器接受更新时 **drawnow** 完成执行，这可能在渲染器完成更新屏幕之前发生。

显式更新

在函数执行过程中，将图形对象添加到图窗或改变现有对象的属性不一定会让屏幕立即更新。当需要更新的图形发生变化时，会出现更新过程，代码：

- 调用 **drawnow**、**pause**、**figure** 或其他实际上引起更新的函数（请参阅 **drawnow**）。
- 查询其值依赖于其他属性的属性（请参阅“自动计算属性”（第 24-10 页））。
- 完成执行并将控制返回给 MATLAB 提示符或调试器。

优化代码以获取和设置图形属性

本节内容
“自动计算属性”（第 24-10 页）
“效率不高的设置和获取周期”（第 24-10 页）
“更改文本 Extent 以旋转标签”（第 24-11 页）

自动计算属性

某些属性依赖于其他属性的值。MATLAB 会根据当前图形模型自动计算这些属性值并更新值。例如，坐标轴范围影响用于轴刻度的值，轴刻度值会影响轴刻度标签。

当您查询计算属性时，MATLAB 会执行隐式 `drawnow` 以确保返回属性值之前所有属性值都是最新的。查询会引起所有从属属性的更新和屏幕的更新。

MATLAB 计算某些属性值时需要根据该属性所依赖的值。例如，绘图函数会自动创建具有坐标轴范围、刻度标签及尺寸适合绘制数据及图窗尺寸的坐标区。

MATLAB 图形执行完全更新，如果需要，再从计算属性返回值以确保返回值是最新的。

下表列出了一些更常见的计算属性。

对象	属性	MATLAB 何时计算这些属性
坐标区	<code>CameraPosition</code> 、 <code>CameraTarget</code> 、 <code>CameraUpVector</code> 、 <code>CameraViewAngle</code>	始终
	<code>Position</code> 、 <code>OuterPosition</code> 、 <code>TightInset</code>	始终
	<code>XLim</code> 、 <code>YLim</code> 、 <code>ZLim</code>	始终
	<code>XTick</code> 、 <code>YTick</code> 、 <code>ZTick</code> 、 <code>XMinorTick</code> 、 <code>YMinorTick</code> 、 <code>ZMinorTick</code>	始终
	<code>XTickLabel</code> 、 <code>YTickLabel</code> 、 <code>ZTickLabel</code> 、 <code>TickDir</code>	始终
	<code>SortMethod</code>	始终
文本	<code>Extent</code>	始终
	<code>Position</code>	仅当文本对象用作坐标区标题或轴标签时
	<code>FontSize</code> 、 <code>FontWeight</code>	仅当文本对象用作坐标区标题或轴标签时

效率不高的设置和获取周期

当您设置属性值时，会更改图形模型的状态，并将其标记为需要更新。当您查询自动计算属性时，如果图形模型和图形硬件未同步，MATLAB 需要执行更新。

当您在同一个循环中获取和设置属性时，会造成每次遍历循环都要执行更新的情况。

- `get` 会引起更新。

- `set` 将图形模型标记为需要更新。

每次遍历循环都会重复这个周期。最好在一个循环中执行所有属性查询，在另一个循环中执行所有属性设置，如以下示例。

此示例获取和设置文本 `Extent` 属性。

性能不佳的代码	性能更高的代码
<pre> h = gobjects(1,500); p = zeros(500,3); for ix = 1:500 h(ix) = text(ix/500,ix/500,num2str(ix)); end drawnow % Gets and sets in the same loop, % prompting a full update at each pass for ix = 1:500 pos = get(h(ix),'Position'); ext = get(h(ix),'Extent'); p(ix,:) = [pos(1)+(ext(3)+ext(1)), ... pos(2)+ext(2)+ext(4),0]; set(h(ix),'Position',p(ix,:)); end drawnow </pre>	<pre> h = gobjects(1,500); p = zeros(500,3); for ix = 1:500 h(ix) = text(ix/500,ix/500,num2str(ix)); end drawnow % Get and save property values for ix=1:500 pos = get(h(ix),'Position'); ext = get(h(ix),'Extent'); p(ix,:) = [pos(1)+(ext(3)+ext(1)), ... pos(2)+ext(2)+ext(4),0]; end % Set the property values and % call a drawnow after the loop for ix=1:500 set(h(ix),'Position',p(ix,:)); end drawnow </pre>
这段代码性能不佳是因为：	这样的性能更佳，因为此代码： <ul style="list-style-type: none"> • <code>Extent</code> 属性依赖于其他值，如屏幕分辨率、图窗尺寸和坐标轴范围，因此查询此属性会引起全部更新。 • 当下一次获取 <code>Extent</code> 属性发生时，就需要设置 <code>Position</code> 属性。

更改文本 `Extent` 以旋转标签

如果您更改文本 `Extent` 属性以旋转标签，那么使用坐标区属性 `XTickLabelRotation`、`YTickLabelRotation` 和 `ZTickLabelRotation` 更高效。

避免更新静态数据

如果每次更新屏幕时只有一小部分定义图形场景的数据发生变化，那么您可以通过只更新发生变化的数据从而提高性能。下例演示了这一技术。

性能不佳的代码	性能更高的代码
<p>在此示例中，通过每次遍历循环时创建两个对象，让标记沿着曲面移动。</p> <pre>[sx,sy,sz] = peaks(500); nframes = 490; for t = 1:nframes surf(sx,sy,sz,'EdgeColor','none') hold on plot3(sx(t+10,t),sy(t,t+10),... sz(t+10,t+10)+0.5,'o',... 'MarkerFaceColor','red',... 'MarkerSize',14) hold off drawnow end</pre>	<p>创建曲面，然后在循环中更新标记的 XData、YData 和 ZData。每次迭代时仅改变标记数据。</p> <pre>[sx,sy,sz] = peaks(500); nframes = 490; surf(sx,sy,sz,'EdgeColor','none') hold on h = plot3(sx(1,1),sy(1,1),sz(1,1),'o',... 'MarkerFaceColor','red',... 'MarkerSize',14); hold off for t = 1:nframes set(h,'XData',sx(t+10,t),... 'YData',sy(t,t+10)... 'ZData',sz(t+10,t+10)+0.5) drawnow end</pre>

分割数据以减少更新次数

考虑以下情况，当代码在循环中执行时，对象数据变得很大，例如随时间追踪信号的线条。

每次调用 **drawnow**，更新都被传递到渲染器。随着数据数组不断变大，性能下降很快。如果您使用这个模式，采用右侧示例中描述的分段方法。

性能不佳的代码	性能更高的代码
<pre>% Grow data figure('Position',[10,10,1500,400]) n = 5000; h = stairs(1,1); ax = gca; ax.XLim = [1,n]; ax.YLim = [0,1]; ax.ZLim = [0,1]; ax.NextPlot = 'add'; xd = 1:n; yd = rand(1,n); tic for ix = 1:n set(h,'XData',xd(1:ix),'YData',yd(1:ix)); drawnow; end toc</pre>	<pre>% Segment data figure('Position',[10,10,1500,400]) n = 5000; seg_size = 500; xd = 1:n; yd = rand(1,n); h = stairs(1,1); ax = gca; ax.XLim = [1,n]; ax.YLim = [0,1]; ax.ZLim = [0,1]; ax.NextPlot = 'add'; tic start = 1; for ix=1:n % Limit object size if (ix-start > seg_size) start = ix-1; h = stairs(1,1); end set(h,'XData',xd(start:ix),... 'YData',yd(start:ix)); % Update display in 50 point chunks if mod(ix,50) == 0 drawnow; end end toc</pre> <p>这段代码性能更高，因为限制因素是更新时发送的数据量。</p>

高效变换对象

移动对象（例如，通过旋转变换）需要变换定义对象的数据。您可以通过利用图形硬件能将变换应用到数据这一功能来提高性能。这样，您就可以避免将变换后的数据发送给渲染器。您只要发送 4×4 变换矩阵。

要发挥此方法的性能优势，请使用 **hgtransform** 函数对要移动的对象分组。

以下示例定义一个球体，并使用两种方法旋转以比较性能：

- **rotate** 函数变换球体的数据并每次调用 **drawnow** 时将数据发送给渲染器线程。
- **hgtransform** 函数将同样旋转的变换矩阵发送给渲染器线程。

性能不佳的代码	性能更高的代码
<p>当对象数据很大时，更新操作瓶颈就成为限制因素。</p> <pre>% Using rotate figure [x,y,z] = sphere(270); s = surf(x,y,z,'EdgeColor','none'); axis vis3d for ang = 1:360 rotate(s,[1,1,1],1) drawnow end</pre>	<p>使用 hgtransform 会将变换应用到瓶颈的渲染器端。</p> <pre>% Using hgtransform figure ax = axes; [x,y,z] = sphere(270); % Transform object contains the surface grp = hgtransform('Parent',ax); s = surf(ax,x,y,z,'Parent',grp,... 'EdgeColor','none'); view(3) grid on axis vis3d % Apply the transform tic for ang = linspace(0,2*pi,360) tm = makehgform('axisrotate',[1,1,1],ang); grp.Matrix = tm; drawnow end toc</pre>

使用低级函数提升速度

那些使得绘图函数易于使用的功能同时也耗费更多的计算机资源。如果您想要尽量提高绘图性能，那么使用低级函数并禁用某些自动功能。

低级图形函数（如 `line` 与 `plot`、`surface` 与 `surf`）执行的操作更少，因此在创建很多图形对象时更快。

低级图形函数有 `line`、`patch`、`rectangle`、`surface`、`text`、`image`、`axes` 和 `light`

图形的系统要求

本节内容
“最低系统要求”（第 24-15 页）
“推荐系统要求”（第 24-15 页）
“升级图形驱动程序”（第 24-15 页）
“具有特定要求的图形功能”（第 24-15 页）

最低系统要求

所有系统都支持大部分常见 MATLAB 图形功能。

推荐系统要求

为了实现最佳图形效果，系统必须具备：

- 至少 1 GB 的 GPU 内存。
- 支持 OpenGL 2.1 或以上硬件加速实现的图形硬件。2006 年以后发布的大多数图形硬件都带有 OpenGL 2.1 或更高版本。如果您使用早期版本的 OpenGL，大多数图形功能仍然有效，但某些高级图形功能将不可用。有关详细信息，请参阅“具有特定要求的图形功能”（第 24-15 页）。要获得最佳性能，推荐使用 OpenGL 4.0 或更高版本。
- 您的计算机制造商或图形硬件供应商提供的最新版本的图形驱动程序。

有关确定您的图形硬件的详细信息，请参阅 [rendererinfo](#)。

从 R2015b 开始，MATLAB 成为一款能识别 DPI 的应用程序，可利用您的全系统分辨率。MATLAB 图形在所有系统上均可清晰显示和正确缩放，包括与 Apple Retina 显示屏连接的 Macintosh 系统以及高 DPI 的 Windows 系统。

升级图形驱动程序

图形硬件供应商频繁地提供更新的图形驱动程序以改进硬件性能。为了帮助确保您的图形硬件适用于 MATLAB，请将您的图形驱动程序升级到可用的最新版本。

- 在 Windows 系统上，检查您的计算机制造商的网站是否提供了驱动程序更新，例如 Lenovo®、HP® 或 Dell®。如果未提供更新，则请检查您的图形硬件供应商网站，例如 AMD® 网站、NVIDIA® 网站或 Intel® 网站。
- 在 Linux 系统上，使用专门的供应商驱动程序替换开源程序。
- 在 Macintosh 系统上，图形驱动程序是操作系统的一部分。使用最近的更新。

具有特定要求的图形功能

大多数图形功能均适用于所有系统。但对某些图形功能的支持取决于：

- 所用的图形渲染器实现是硬件、基础硬件还是软件。默认情况下，MATLAB 使用硬件加速图形，前提是您的图形硬件支持它。基础硬件和软件 OpenGL 为替代选项，可用来解决低级图形问题。在某些情况下，MATLAB 会自动切换到软件 OpenGL。有关详细信息，请参阅 [rendererinfo](#)。

- 渲染器实现的版本，例如 OpenGL 2.1。

下表列出了高级图形功能，以及支持这些功能的环境。有关这些功能的详细信息，请参阅 [rendererinfo](#)。

图形功能	硬件 OpenGL	基础硬件 OpenGL	Windows 上的软件 OpenGL	Linux 上的软件 OpenGL	WebGL™
图形平滑处理	支持 OpenGL 2.1 或更高版本	支持 OpenGL 2.1 或更高版本	不支持	不支持	支持
深度剥离透明度	支持 OpenGL 2.1 或更高版本	禁用	不支持	支持	支持
对齐顶点中心	支持 OpenGL 2.1 或更高版本	禁用	不支持	不支持	支持
硬件加速标记	支持 OpenGL 4.0 或更高版本	禁用	不支持	不支持	支持

另请参阅

函数

[opengl](#) | [rendererinfo](#)

详细信息

- “解决低级图形问题”（第 24-17 页）

解决低级图形问题

MATLAB 在您的系统上创建图形时可能遇到低级问题。例如，条形图中可能缺少条形边，针状图可能缺少针，或者您的图形硬件的内存可能不足。在使用包含示波器的 Simulink® 模型或使用 MathWorks 工具箱中的 UI 创建 2-D 或 3-D 图表时，您可能遇到这些问题。这些问题通常是图形硬件太旧或图形驱动程序过时而导致的。要解决它们，请尝试此处介绍的方法。

升级图形硬件驱动程序

图形硬件供应商频繁地提供更新的图形驱动程序以改进硬件性能。为了帮助确保您的图形硬件适用于 MATLAB，请将您的图形驱动程序升级到可用的最新版本。

- 在 Windows 系统上，请在您的计算机制造商（例如 Lenovo、HP 或 Dell）的网站中检查是否有驱动程序更新。如果未提供更新，则请检查您的图形硬件供应商网站，例如 AMD、NVIDIA 或 Intel。
- 在 Linux 系统上，使用专门的供应商驱动程序替换开源程序。
- 在 Macintosh 系统上，图形驱动程序是操作系统的一部分。使用最近的更新。

使用支持 OpenGL 2.1 或更高版本的硬件加速实现的图形硬件。2006 年以后发布的大多数图形硬件都带有 OpenGL 2.1 或更高版本。如果您使用早期版本的 OpenGL，大多数图形功能仍然有效，但某些高级图形功能将不可用。要获得最佳性能，推荐使用 OpenGL 4.0 或更高版本。有关确定您的图形硬件的详细信息，请参阅 `rendererinfo`。

选择适合您的系统的渲染器实现

MATLAB 显示所用的图形渲染器实现是硬件加速、基础硬件加速还是软件。默认情况下，MATLAB 会尝试使用硬件加速实现（如果您的图形硬件支持）。您可以通过切换到软件实现或基础硬件加速实现来解决许多图形问题。这些替代实现不支持某些高级图形功能。

在某些情况下，MATLAB 会自动切换到软件实现：

- 如果系统没有必要的图形硬件。
- 如果您使用的图形驱动程序存在已知问题、使用的图形驱动程序的版本较低或在使用图形虚拟化。请将您的图形驱动程序更新为可用的最新版本。
- 如果以前的 MATLAB 会话因图形问题而崩溃。如果先前的会话使用软件 OpenGL 并且崩溃，则后续会话将使用具有较少功能但更稳定的软件 OpenGL 版本。

在 Windows 系统上使用远程桌面时，硬件加速图形不一定总是可用。如果在硬件加速图形不受支持的情况下尝试使用，MATLAB 将返回一条警告消息，并改用软件 OpenGL。将您的图形驱动程序更新为最新版本可能会支持硬件加速图形。

要确定 MATLAB 使用的是哪种实现，请调用 `rendererinfo` 函数。例如，此命令将获取当前坐标区的信息并将其存储在名为 `info` 的结构体中。

```
info = rendererinfo(gca)
```

此结构体还在 `GraphicsRenderer` 字段中提供图形渲染器的名称。例如，如果 MATLAB 使用的是硬件加速 OpenGL，该字段将返回 '`OpenGL Hardware`'。如果使用的是软件 OpenGL，该字段将返回 '`OpenGL Software`'。

为当前会话指定 OpenGL 实现

要为当前 MATLAB 会话指定某一 OpenGL 实现，请按下列相应方法操作。

- 软件 OpenGL - 使用命令 `matlab -softwareopengl` 从系统中的命令提示符启动 MATLAB。此命令仅适用于 Windows 和 Linux 系统。Macintosh 系统不支持 OpenGL 软件。
- 基础硬件加速 OpenGL - 在 MATLAB 命令提示符下键入 `opengl hardwarebasic`。
- 硬件加速 OpenGL - 在 MATLAB 命令提示符下键入 `opengl hardware`。

为以后的会话指定 OpenGL 实现

要设置预设项以使 MATLAB 始终使用指定的 OpenGL 实现启动，请按下列相应方法操作。

- 软件 OpenGL - 在 MATLAB 命令提示符处键入 `opengl('save','software')`。然后，重新启动 MATLAB。
- 基础硬件加速 OpenGL - 在 MATLAB 命令提示符处键入 `opengl('save','hardwarebasic')`。然后，重新启动 MATLAB。
- 硬件加速 OpenGL - 在 MATLAB 命令提示符处键入 `opengl('save','hardware')`。然后，重新启动 MATLAB。
- 撤消预设设置 - 在 MATLAB 命令行处执行 `opengl('save','none')`。然后，重新启动 MATLAB。

解决内存不足的问题

图形内存有限的图形硬件可能导致性能不佳或内存不足问题。可通过以下更改提升性能和解决内存问题：

- 使用较小的图窗窗口。
- 将图窗的 **GraphicsSmoothing** 属性设置为 '`off`' 以关闭消除锯齿功能。
- 不使用透明度。
- 使用软件 OpenGL。

联系技术支持

如果使用上述各方法仍无法解决问题，则您可能遇到了 MATLAB 中的程序错误。请联系 MathWorks 技术支持并提供以下信息：

- `info = rendererinfo(gca)` 所返回的输出。
- 使用软件 OpenGL 时您的代码是否能运行而不出现错误。
- 您的代码是否能在不同计算机上运行而不出现错误。请提供您在其上进行代码测试的所有计算机的 `rendererinfo` 输出。
- 某些错误消息包含指向某个文件的链接，该文件包含有关您遇到的图形错误的详细信息。如果提供了此文件的链接，请将此文件连同您的服务请求一起提交。

在 https://www.mathworks.com/support/contact_us 页面上创建服务请求。

另请参阅

[opengl](#) | [rendererinfo](#)

详细信息

- “[图形的系统要求](#)” (第 24-15 页)