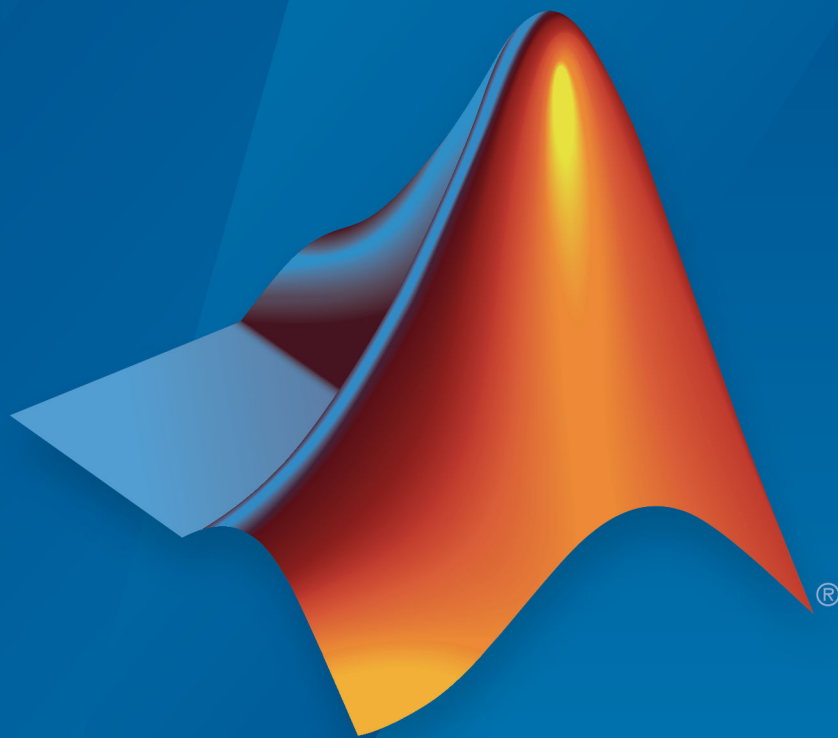


MATLAB®

R2014b 中的图形变化



MATLAB®

R2019a

 **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® R2014b 中的图形变化

© COPYRIGHT 2014–2019 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

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) 中的再发布内容
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

R2014b 中的主要图形变化	1-2
将图形句柄用作对象句柄	1-2
新的视觉外观	1-2
新的默认图形引擎	1-3
兼容性考虑	1-3
图形句柄现在是对象，而不再是双精度值	1-4
将图形句柄用作对象句柄	1-4
访问图形对象的属性	1-5
图形句柄数组	1-5
检验图形句柄的有效性	1-5
通过整数句柄引用图窗	1-6
删除多个图形对象	1-6
使用图形句柄的逻辑表达式	1-6
转换图形句柄的元胞数组	1-7
检验图形句柄的同等性	1-7
从 cellfun 和 arrayfun 函数返回图形对象	1-8
保存图形对象	1-8
编写 MEX 文件	1-9
编写在多个版本中兼容的代码	1-10
对小代码段进行分支	1-10
对大代码段进行分支	1-10
为何绘图线条有不同颜色？	1-11
新的色序	1-11
hold 命令循环遍历颜色	1-11
重置色序	1-13
如何缩小图形标题？	1-15
为何在使用坐标轴命令时坐标轴范围不断更改？	1-18

为何图形的某部分被切除？	1-22
图形对象不再超出坐标区边界	1-22
禁用裁剪	1-24
控制裁剪样式	1-26
为何颜色栏和图例不是有效的坐标区句柄？	1-28
使用支持的颜色栏和图例属性	1-28
颜色栏和图例不能为当前坐标区	1-28
使用新类型属性值查找对象	1-29
颜色栏和图例没有子级	1-29
如何替换 EraseMode 属性？	1-30
创建动画	1-30
显示对象数据的更改	1-30
生成叠加颜色	1-31
提高渲染速度	1-31
为何有些对象的子级属性为空？	1-32
为何同时显示图窗？	1-34
为什么访问刻度标签元素时返回错误消息？	1-35
如何获得对数坐标区的指数值？	1-36
为何未复制回调和应用程序数据？	1-37

更新 GUI 代码

2

为何有些组件缺失或部分遮盖？	2-2
变更说明	2-2
还原编程布局	2-2
还原 GUIDE 布局	2-4
为何 ResizeFcn 的行为发生变化？	2-7
程序启动后 ResizeFcn 返回错误	2-7
ResizeFcn 对不可见组件处于非活动状态	2-8
外部边界或可绘制区域变化时发生意外行为	2-8

为何 <code>handle.listener</code> 返回一个错误?	2-10
---	------

更新图形代码

- “R2014b 中的主要图形变化”（第 1-2 页）
- “图形句柄现在是对象，而不再是双精度值”（第 1-4 页）
- “编写在多个版本中兼容的代码”（第 1-10 页）
- “为何绘图线条有不同颜色？”（第 1-11 页）
- “如何缩小图形标题？”（第 1-15 页）
- “为何在使用坐标轴命令时坐标轴范围不断更改？”（第 1-18 页）
- “为何图形的某部分被切除？”（第 1-22 页）
- “为何颜色栏和图例不是有效的坐标区句柄？”（第 1-28 页）
- “如何替换 EraseMode 属性？”（第 1-30 页）
- “为何有些对象的子级属性为空？”（第 1-32 页）
- “为何同时显示图窗？”（第 1-34 页）
- “为什么访问刻度标签元素时返回错误消息？”（第 1-35 页）
- “如何获得对数坐标区的指数值？”（第 1-36 页）
- “为何未复制回调和应用程序数据？”（第 1-37 页）

R2014b 中的主要图形变化

从 R2014b 开始，MATLAB 图形系统的构建基础架构进行了改进，提供了新的视觉外观、新的图形引擎和许多增强功能，并添加了自定义图表选项。这里描述了此版本引入的一些图形变化。

将图形句柄用作对象句柄

现在，图形对象使用各种类型的对象句柄，而不再是先前版本中使用的数值句柄。图形对象的行为与其他 MATLAB 对象相似，并且支持获取和设置属性的圆点表示法。有关详细信息，请参阅“图形句柄现在是对象，而不再是双精度值”（第 1-4 页）。

新的视觉外观

MATLAB 图形的新视觉外观通过以下技术对图形的清晰度和美观度做了很大的改进：

- 新默认颜色图称为 **parula**。新颜色图从深到浅排序，并且在视觉感受上是均匀的。数据的平滑变化显示为颜色的平滑变化，而数据的急剧变化显示为颜色的急剧变化。新颜色图表示数据更准确，更便于解释数据。
- 绘制线条时的新颜色。颜色具有均等的饱和度，更易于区分多行。
- 更浅的图窗背景色和更浅的网格线可突出绘图数据。
- 消除锯齿字体和线条，使文本和图形更平滑。有关详细信息，请参阅图窗的 **GraphicsSmoothing** 属性以及坐标区和文本对象的 **FontSmoothing** 属性。
- 用于设置网格线条颜色以及控制标题和坐标区标签字体大小的新坐标区属性。有关详细信息，请参阅 **GridColor**、**TitleFontSizeMultiplier** 和 **LabelFontSizeMultiplier** 属性。

其他增强功能和新的自定义选项包括：

- 可旋转的坐标轴刻度标签。使用坐标区的 **XTickLabelRotation**、**YTickLabelRotation** 和 **ZTickLabelRotation** 属性。
- 在坐标轴刻度标签中使用特殊字符，例如上标、下标和希腊字母。默认情况下，坐标区将使用 TeX 标记来解释刻度标签字符。有关详细信息，请参阅坐标区的 **TickLabelInterpreter** 属性。
- 在单个图窗中为每套坐标区使用不同的颜色图。若要更改坐标区的颜色图，请将坐标区作为输入参数传递给 **colormap** 函数。
- 在 **plot** 中使用 **datetime** 和 **duration** 数据类型时，自动更新坐标轴刻度标签。

- **categorical** 数据的饼图可自动添加扇区标签。
- 当 **Clipping** 属性设置为 'on'（这是默认值）时，三维图形不再延伸到坐标区边界之外。

新的默认图形引擎

从 R2014b 开始，MATLAB 采用 OpenGL® 作为图形的默认渲染器。改进的 OpenGL 渲染（如支持向量输出透明度）将使用图表时切换渲染器的需要降到最低。

兼容性考虑

R2014b 引入的图形变化支持以前版本的大多数函数，尽管存在一些差异。有关您可能遇到的与版本变化有关的最常见故障排除主题列表，请参阅“R2014b 中的图形变化”。有关已删除的属性和函数语法的列表，请参阅图形发行说明中的“Save and print functionality being removed or changed”和“Properties and syntaxes being removed or changed”。

如果图形驱动程序过时，有些图形功能则可能无法正常工作或不稳定。这种情况下，请升级到您的图形硬件制造商所提供的最新图形驱动程序。有关详细信息，请参阅“图形的系统要求”。

另请参阅

详细信息

- “为何绘图线条有不同颜色？”（第 1-11 页）

图形句柄现在是对象，而不再是双精度值

本节内容

“将图形句柄用作对象句柄”（第 1-4 页）
“访问图形对象的属性”（第 1-5 页）
“图形句柄数组”（第 1-5 页）
“检验图形句柄的有效性”（第 1-5 页）
“通过整数句柄引用图窗”（第 1-6 页）
“删除多个图形对象”（第 1-6 页）
“使用图形句柄的逻辑表达式”（第 1-6 页）
“转换图形句柄的元胞数组”（第 1-7 页）
“检验图形句柄的同等性”（第 1-7 页）
“从 cellfun 和 arrayfun 函数返回图形对象”（第 1-8 页）
“保存图形对象”（第 1-8 页）
“编写 MEX 文件”（第 1-9 页）

将图形句柄用作对象句柄

在以前版本中，图形句柄是 `double` 类型的数值句柄。从 R2014b 开始，图形句柄是各种类型的对象句柄，具体取决于图形对象的类。现在，图形对象的行为与其他 MATLAB 对象相似。

大多数为数值句柄编写的代码仍适用于对象句柄。例如，您可以访问图形对象属性，可以将图形对象合并到数组中，即使对象属于不同的类也没有关系。然而，您不能执行假设或要求图形句柄是数字值这样的运算，如：

- 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
- 依赖根对象 (0) 的数值或逻辑语句中的图窗句柄（整数）
- 将句柄与数字数组中的数据进行合并
- 使用取决于数值句柄的任何程序逻辑
- 将句柄转换为字符向量，或在字符向量运算中使用句柄

访问图形对象的属性

有两种方式可访问带有对象句柄的图形对象的属性：

- 使用圆点表示法引用特定的对象和属性。当使用圆点表示法时，属性名称区分大小写。例如，此代码将线条的 **Color** 属性设置为 'red'。

```
h = plot(1:10);
h.Color = 'red';
```

- 使用 **set** 和 **get** 函数访问对象数组的属性。例如，此代码设置多个线条的 **LineWidth** 属性。

```
h = plot(rand(4));
set(h,'LineWidth',2);
```

图形句柄数组

从 R2014b 开始，使用 **gobjects** 函数（而不是 **zeros** 或 **ones** 函数）预分配图形句柄数组。**zeros** 或 **ones** 预分配仍可运行不出现错误，但执行速度可能会很慢。

gobjects 的语法与 **ones** 和 **zeros** 的语法相同。

```
h = gobjects(3,1); % preallocate
h(1) = figure;
h(2) = plot(1:10);
h(3) = gca;
```

您可以将图形句柄合并到数组中，即便句柄是不同的类也没关系。MATLAB 会将数组转换为一个公共基类。

```
class(h)
```

```
ans =
```

```
matlab.graphics.Graphics
```

检验图形句柄的有效性

从 R2014b 开始，使用 **isgraphics** 函数（而不是 **ishandle** 函数）检验图形句柄的有效性。

```
x = 1:10;
y = sin(x);
```

```
p = plot(x,y);  
ax = gca;  
isgraphics([p,ax])  
  
ans =  
  
1 1
```

通过整数句柄引用图窗

从 R2014b 开始，您可以通过对象句柄或整数句柄引用图窗。整数句柄是图窗的新 **Number** 属性中的值。

```
h = figure; % object handle  
fignum = h.Number; % integer handle  
  
整数句柄 fignum 是有效的图窗句柄。  
  
isgraphics(fignum) % test handle validity  
  
ans =  
  
1
```

删除多个图形对象

从 R2014b 开始，**delete** 函数仅接受一个输入参数。若要删除多个图形对象，请将单个句柄数组传递给函数，而不要使用多个参数。

```
h1 = annotation('line');  
h2 = annotation('ellipse');  
h3 = annotation('rectangle');  
delete([h1,h2,h3])
```

使用图形句柄的逻辑表达式

从 R2014b 开始，您不能在逻辑表达式中使用图形句柄或依靠 MATLAB 返回非零值或空的双精度值 `[]`。相反，您应该使用函数来实现目的，如 **isempty**、**isgraphics** 和 **isequal**。

- 若要确定是否存在现有图窗，请使用 **isempty**。新的 **groot** 命令可以引用根对象。

```
if ~isempty(get(groot,'CurrentFigure'))  
    disp('There are existing figures.')
```

```
else
    disp('There are no existing figures.')
end
```

- 若要确定是否存在带特定标签的图形对象，请使用 `isempty`。

```
if ~isempty(findobj('Tag','myFigures'))
    disp('There are objects with this tag.')
else
    disp('There are no objects with this tag.')
end
```

- 若要确定句柄是否为有效的图窗句柄，请使用 `isgraphics` 和对对象 `Type`。

```
if isgraphics(h,'figure')
    disp('h is a valid figure handle.')
else
    disp('h is not a valid figure handle.')
end
```

- 若要确定句柄是否为根句柄，请使用新的 `groot` 命令。

```
if isequal(h,groot)
    disp('h is the root handle')
else
    disp('h is not the root handle')
end
```

转换图形句柄的元胞数组

从 R2014b 开始，您不能再对图形句柄的元胞数组使用 `cell2mat` 来创建数值数组。相反，应从元胞数组创建对象数组。

```
p = plot(magic(3));
par = get(p,'Parent');
objarray = [par{:}];
whos objarray
```

Name	Size	Bytes	Class	Attributes
objarray	3x1	128	matlab.graphics.axis.Axes	

检验图形句柄的同等性

从 R2014b 开始，使用 `==` 或 `isequal` 检验图形句柄的同等性。

- 若要确定句柄引用的是否为同一对象并以此来检验是否为相同句柄)，请使用 `==`。

```
p1 = plot(1:10);  
p2 = p1;  
p2 == p1
```

```
ans =
```

```
1
```

- 若要确定句柄是否在引用具有相同属性值的同类对象（但不一定是相同对象），请使用 `isequal`。

```
l1 = line;  
l2 = line;  
isequal(l1,l2)
```

```
ans =
```

```
1
```

从 `cellfun` 和 `arrayfun` 函数返回图形对象

若要使用 `cellfun` 和 `arrayfun` 函数返回图形对象，请将 `UniformOutput` 设置为 `false`。

例如：

```
t = num2str(rand);  
fh = @(t) text(1,1,t);  
th = cellfun(fh,{t},'UniformOutput',false);
```

保存图形对象

从 R2014b 开始，如果使用 `save` 函数将图形对象保存到 MAT 文件中，则该 MAT 文件将包含重新生成该对象所需的全部信息。在以前的版本中，`save` 函数以双精度值的形式保存对象，并且当您加载 MAT 文件时，无法重新生成该对象。

请避免使用 `save` 函数保存图窗。如果在 R2014b 或更高版本中使用 `save` 保存图窗，将导致无法在早期版本的 MATLAB 中访问该 MAT 文件。如果使用 `save` 保存图窗，该函数会显示警告消息。请改用 `savefig` 函数保存图窗。

编写 MEX 文件

如果您编写 MEX 文件或编译引擎应用程序，则 `mexGet` 和 `mexSet` 函数将无法用在图形对象句柄上。相反，应使用 C/C++ 或 Fortran Matrix Library 中的 `mxGetProperty` 和 `mxSetProperty` 函数。

编写在多个版本中兼容的代码

以前版本中编写的大多数图形代码都可兼容 R2014b 中引入的图形变化。当然，在某些情况下，代码可以在一个版本中运行，却无法在另一个版本中运行。如有可能，请尽量实施在不同版本中通用的替代方案。如果不存在替代方案，则您可以将您的代码进行分支以执行不同的代码路径。

对小代码段进行分支

若要使用一个小功能，例如一个属性，请根据是否存在某特定功能对代码进行分支。例如，`SortMethod` 是 R2014b 中引入的一个坐标区属性。此代码检查属性是否存在，然后才设置其值。

```
ax = gca;  
if isprop(ax,'SortMethod')  
    set(ax,'SortMethod','childorder')  
end
```

对大代码段进行分支

若在没有要测试的特定功能时对大代码段进行分支，请使用 `verLessThan('matlab','8.4.0')` 命令。如果您正在运行 R2014b 或更高版本，此命令返回 `0`；如果您正在运行较低版本，此命令返回 `1`。例如，使用此编码模式对您的代码进行分支。

```
if verLessThan('matlab','8.4.0')  
    % execute code for R2014a or earlier  
else  
    % execute code for R2014b or later  
end
```

另请参阅



`class` | `iscell` | `ischar` | `ishghandle` | `ismethod` | `isprop` | `verLessThan` | `whos`

为何绘图线条有不同颜色？

本节内容
“新的色序” （第 1-11 页）
“hold 命令循环遍历颜色” （第 1-11 页）
“重置色序” （第 1-13 页）

新的色序

从 R2014b 开始，MATLAB 图形采用新的色序，该顺序决定绘图中使用的颜色。此表显示了 R2014b 引入的色序与以前版本的对比。它还列出了定义颜色的 RGB 三元组值。

从 R2014b 开始	R2014a 和较早版本
	
0 0.4470 0.7410	0 0 1.0000
0.8500 0.3250 0.0980	0 0.5000 0
0.9290 0.6940 0.1250	1.0000 0 0
0.4940 0.1840 0.5560	0 0.7500 0.7500
0.4660 0.6740 0.1880	0.7500 0 0.7500
0.3010 0.7450 0.9330	0.7500 0.7500 0
0.6350 0.0780 0.1840	0.2500 0.2500 0.2500

坐标区的 **ColorOrder** 属性包含色序。要更改色序，请为 **ColorOrder** 属性设置不同的默认值。例如，此代码将默认色序设置为以前版本中使用的颜色。

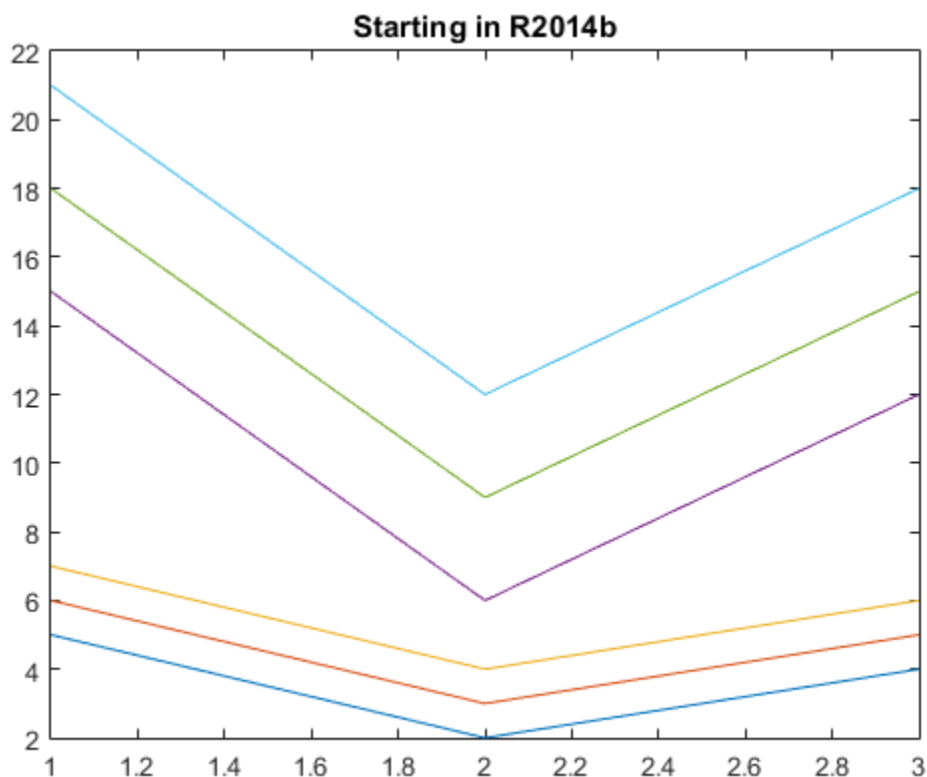
```
co = [0 0 1;  
      0 0.5 0;  
      1 0 0;  
      0 0.75 0.75;  
      0.75 0 0.75;  
      0.75 0.75 0;  
      0.25 0.25 0.25];  
set(groot,'defaultAxesColorOrder',co)
```

hold 命令循环遍历颜色

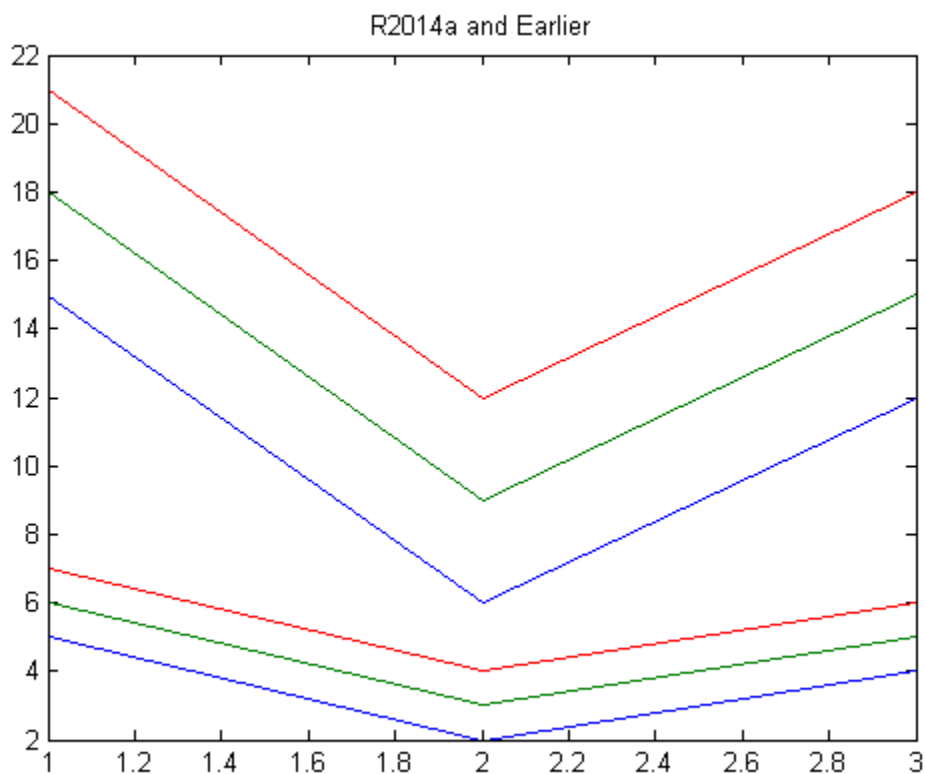
在 R2014a 和早期版本中，**hold on** 命令并不保留当前颜色，因此向坐标区添加的新绘图会从色序的开头开始。视觉上，这意味着新绘图使用相同的初始颜色。从 R2014b 开始，**hold on** 命令将保留当前颜色，以便向坐标区添加的新绘图使用色序中的下一种颜色。

例如，此代码使用 `hold on` 命令显示六个线条。从 R2014b 开始，线条会按色序循环着色，生成的绘图使用色序的前六种颜色。

```
data = [5 6 7; 2 3 4; 4 5 6];  
plot(data);  
hold on  
plot(3*data);  
hold off
```



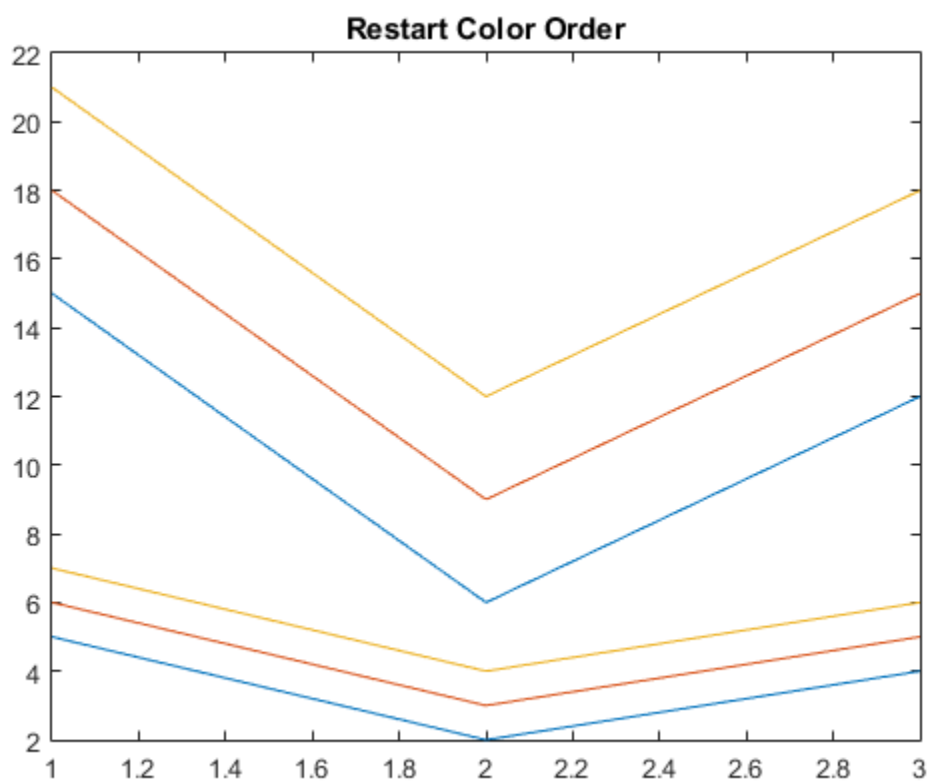
如果您在早期版本中运行相同的代码，则每次运行绘图命令时会重置色序。生成的绘图会使用色序中前三种颜色两次。



重置色序

从 R2014b 开始，如果希望在每个绘图命令之前按重置色序，则将坐标区的 `ColorOrderIndex` 属性设置为 1。

```
data = [5 6 7; 2 3 4; 4 5 6];  
plot(data);  
hold on  
ax = gca;  
ax.ColorOrderIndex = 1;  
plot(3*data);  
hold off
```

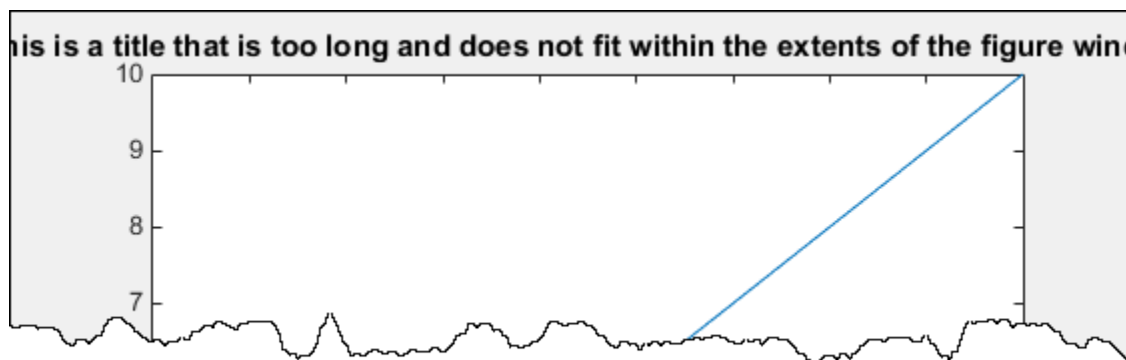


另请参阅
hold

如何缩小图形标题？

从 R2014b 开始，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。

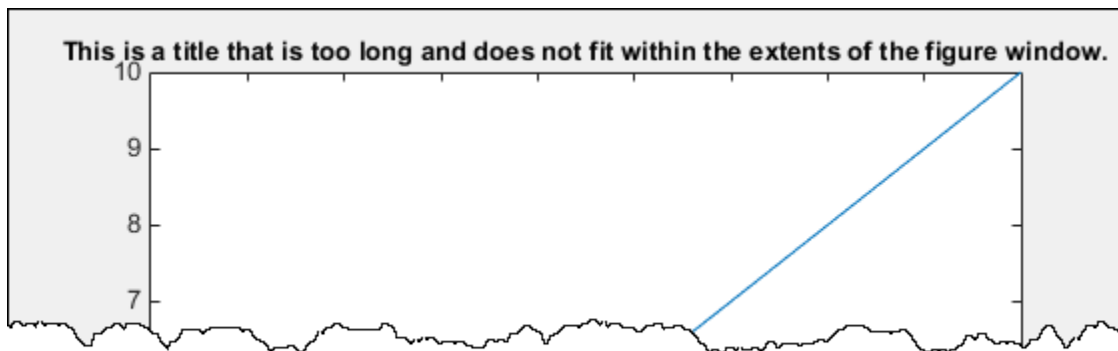
```
plot(1:10);
title(['This is a title that is too long and does not fit',...
      'within the extents of the figure window.'])
```



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。

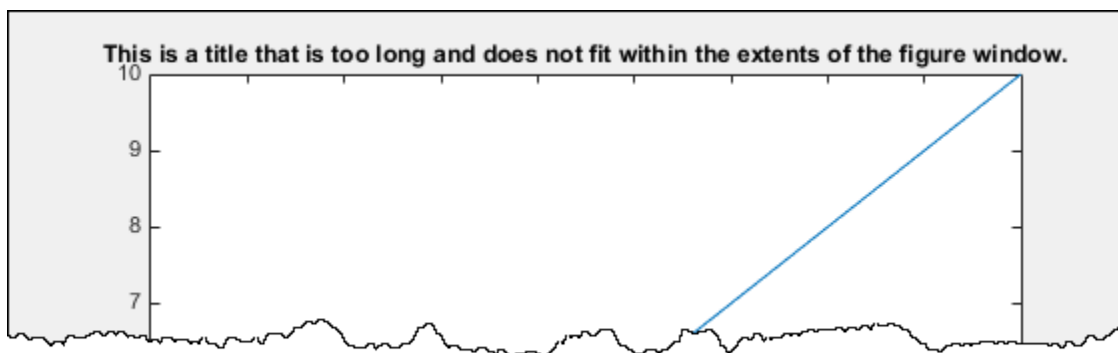
若要更改标题字体大小而不影响坐标区的其他部分字体，请设置坐标区的 **TitleFontSizeMultiplier** 属性。

```
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;
```



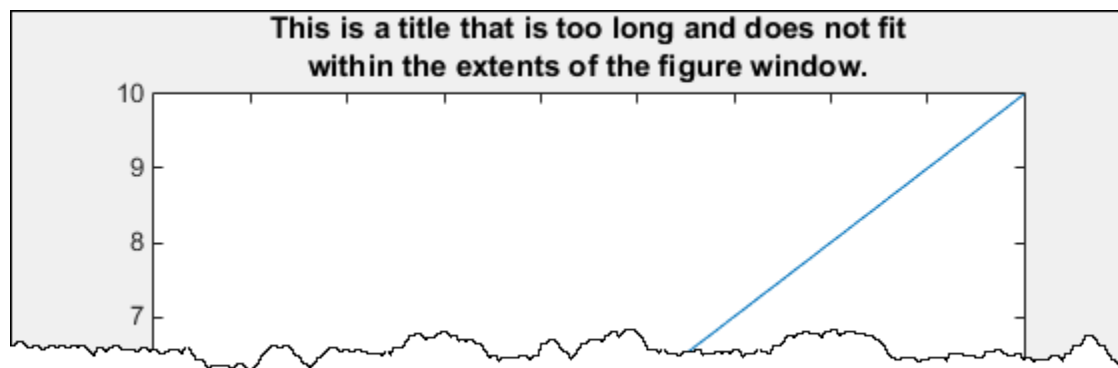
若要使整个坐标区的字体变小，请设置 `FontSize` 属性。更改此属性会影响标题、刻度标签和坐标轴标签（如果存在）的字体。

```
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

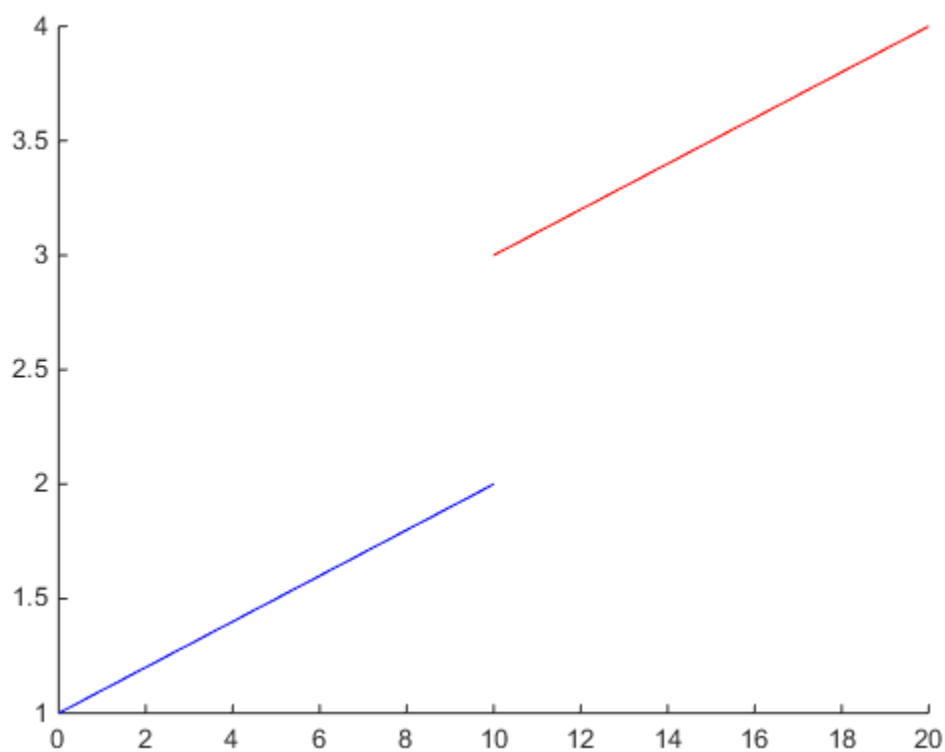
为何在使用坐标轴命令时坐标轴范围不断更改？

在 R2014a 及更早版本中，如果您使用 `axis tight` 或 `axis image` 命令设置坐标轴范围，则计算的坐标轴范围不会更改。如果您将新数据添加到当前坐标轴范围之外，则范围不会自动更新以包括数据。

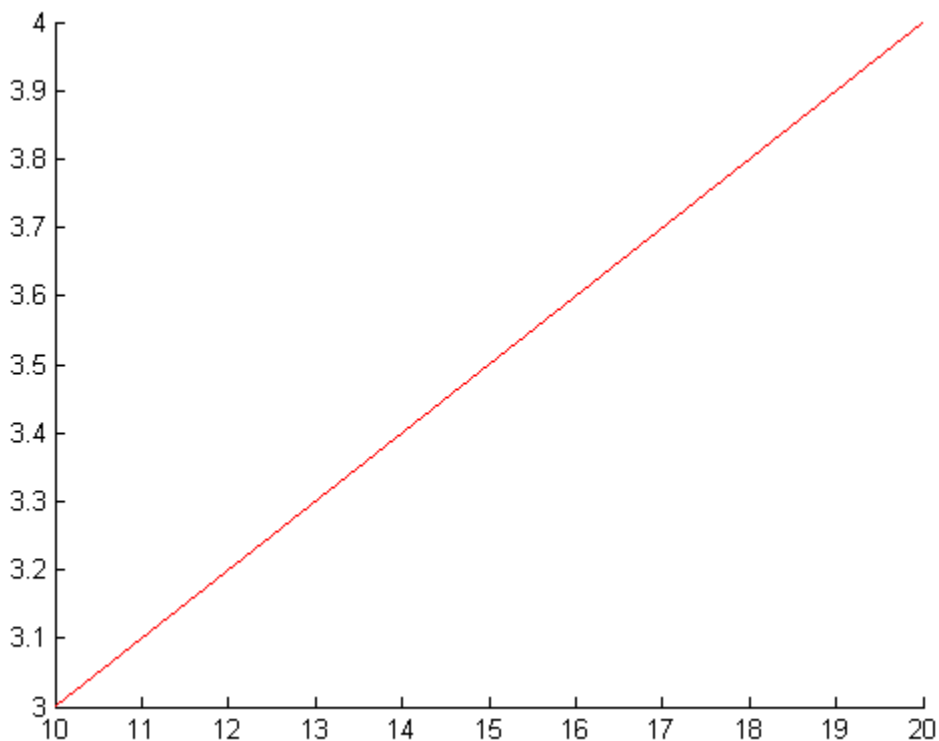
从 R2014b 开始，如果您使用这些命令，而后将新数据添加到坐标区上，则坐标轴范围会自动更新以包括这些数据。

例如，此代码使用 `axis tight` 命令设置坐标轴范围，然后将新数据添加到图形。从 R2014b 开始，坐标轴范围会更新以容纳两个线条。

```
line([10 20],[3 4],'Color','red')  
axis tight  
line([0 10],[1 2],'Color','blue')
```



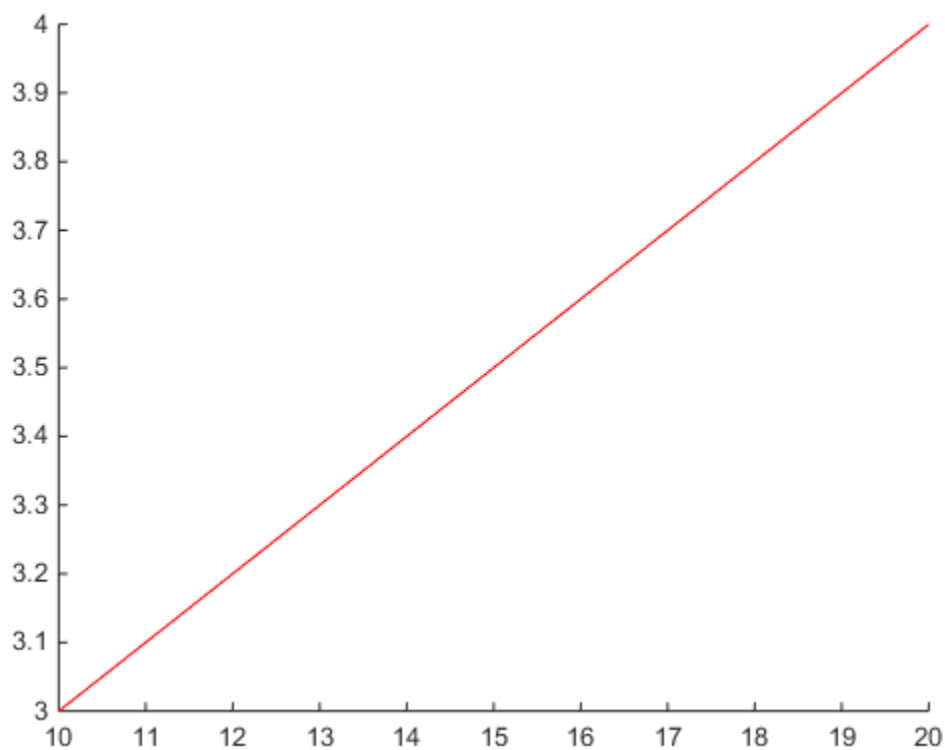
如果您在以前版本中运行相同代码，则范围不会更新，因此蓝色线条在坐标区中不可见。



在 R2014a 和早期版本中，这些 `axis` 命令将坐标轴范围模式 (`XLimMode`、`YLimMode` 和 `ZLimMode`) 设置为 `'manual'`。当范围模式为手动时，范围不会更新以反映数据的变化。从 R2014b 开始，这些 `axis` 命令将坐标轴范围模式设置为 `'auto'`。当范围模式为自动时，范围会更新以反映数据的变化。

从 R2014b 开始，要防止坐标轴范围自动更新，请将 `manual` 追加到 `axis` 命令的末尾处。例如，`axis tight manual`。`manual` 选项将范围模式设置为手动，如以前版本一样，因此范围不会自动更新。例如，此代码不会更新范围以包括第二个蓝色线条。

```
line([10 20],[3 4], 'Color', 'red');  
axis tight manual;  
line([0 10],[1 2], 'Color', 'blue');
```



另请参阅

函数
axis

属性
Axes

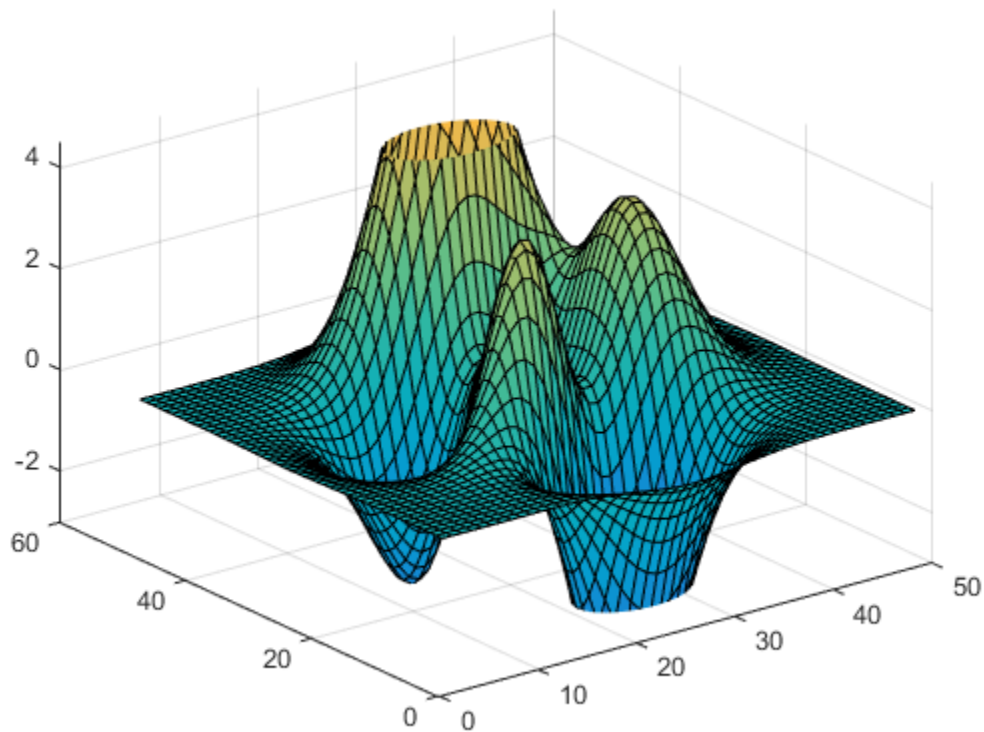
为何图形的某部分被切除？

本节内容
“图形对象不再超出坐标区边界”（第 1-22 页）
“禁用裁剪”（第 1-24 页）
“控制裁剪样式”（第 1-26 页）

图形对象不再超出坐标区边界

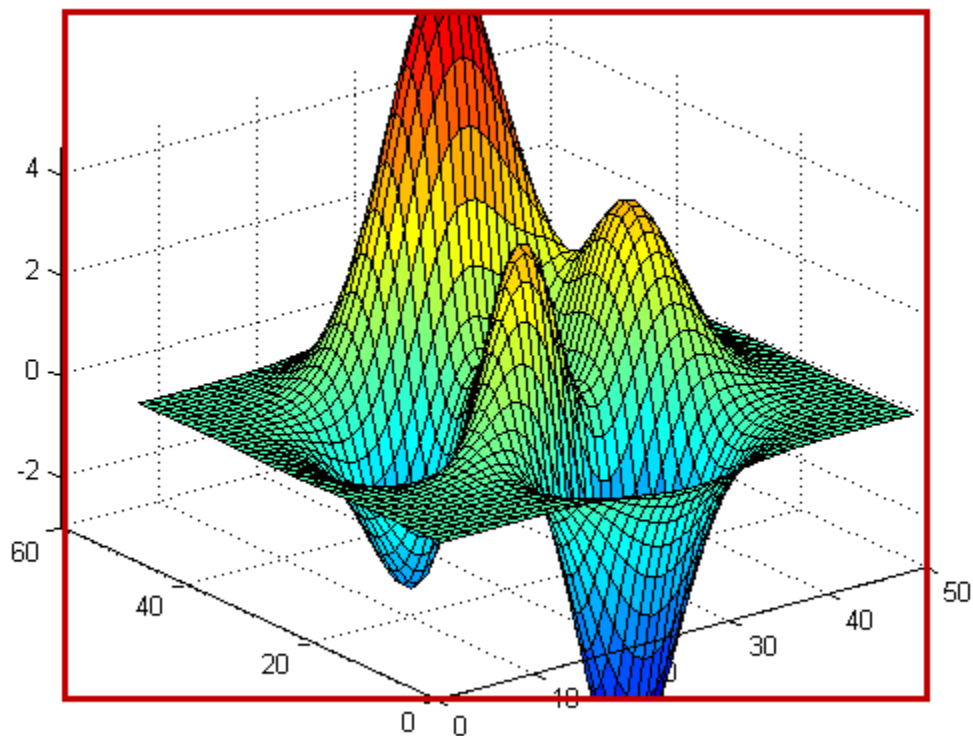
从 R2014b 开始，图形对象不再超出坐标区边界。对象被裁剪到坐标区范围定义的坐标区框的六个边。例如，MATLAB 不会显示超出指定的 z 范围的曲面峰值。

```
surf(peaks);  
zlim([-3,4.5]);
```



在 R2014a 和早期版本中，MATLAB 使用一种不同的技术来裁剪对象。MATLAB 并不裁剪到坐标区范围，而是裁剪到包围坐标区的最小二维矩形。例如，在以前版本中，相同的曲面绘图便会超出指定的 z 范围。红色矩形表示用于裁剪的边界。

```
surf(peaks);  
zlim([-3,4.5]);
```



禁用裁剪

坐标区以及坐标区中的个别对象有控制裁剪行为的 **Clipping** 属性。默认情况下，此属性设置为 'on'。若要禁用裁剪，请将 **Clipping** 属性设置为 'off'。

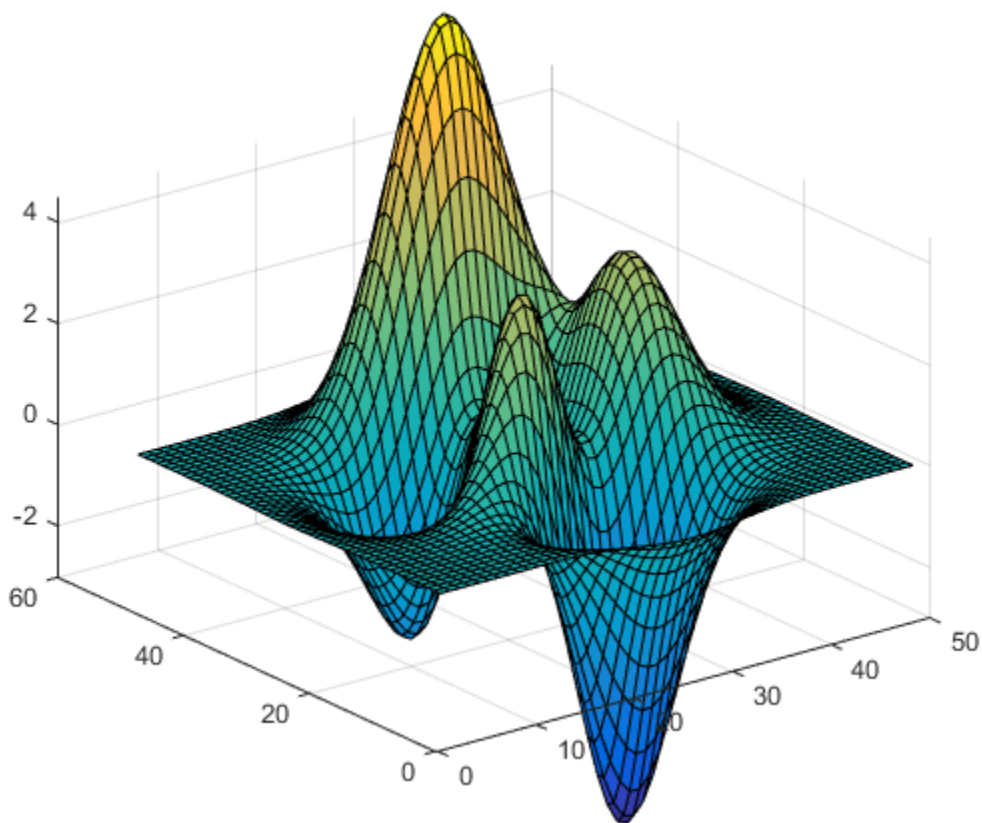
如果坐标区的 **Clipping** 属性为 'on'，则由坐标区的每个对象控制其自身的裁剪行为。要对坐标区中的所有对象禁用裁剪，请将坐标区的 **Clipping** 属性设置为 'off'。下表列出了 **Clipping** 属性值的不同组合结果。

坐标区的裁剪属性	单个对象的裁剪属性	结果
'on'	'on'	裁剪单个对象（默认值）

坐标区的裁剪属性	单个对象的裁剪属性	结果
'on'	'off'	不裁剪单个对象
'off'	'on'	不裁剪坐标区中的对象
'off'	'off'	不裁剪坐标区中的对象

例如，通过将坐标区的 Clipping 属性设置为 'off' 对坐标区中的所有对象禁用裁剪。

```
surf(peaks);  
zlim([-3,4.5]);  
ax = gca;  
ax.Clipping = 'off';
```



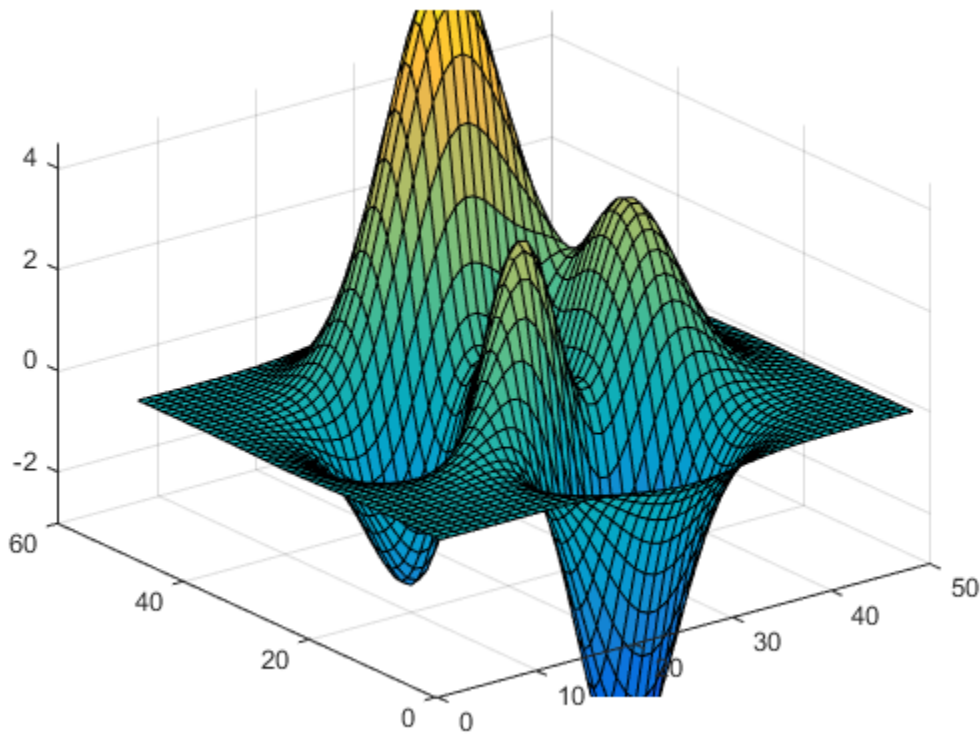
控制裁剪样式

新的 `ClippingStyle` 坐标区属性控制用于裁剪对象的技术。将此属性设置为以下值之一：

- `'3dbox'` - 按照坐标区范围定义的坐标区框的六个边对绘制的对象进行裁剪。这是默认值。
- `'rectangle'` - 按照任何给定视图中坐标区围成的矩形边界对绘制的对象进行裁剪。

若要在 R2014a 和早期版本中获得相同的裁剪样式，请将 `ClippingStyle` 设置为 `'rectangle'`。

```
surf(peaks)  
zlim([-3,4.5]);  
ax = gca;  
ax.ClippingStyle = 'rectangle';
```



另请参阅
Axes

为何颜色栏和图例不是有效的坐标区句柄？

从 R2014b 开始，颜色栏和图例不再是坐标区对象。它们是具有支持的自有属性集的新对象类型。在以前版本中，它们是可使用坐标区属性修改的坐标区对象。但是，很多坐标区属性与颜色栏和图例无关。

您不应执行认定或要求颜色栏和图例是坐标区对象的运算。

使用支持的颜色栏和图例属性

请勿使用坐标区属性修改颜色栏和图例。使用其支持的属性。有关列表，请参阅 Colorbar 或 Legend。

例如，若要反转色阶沿颜色栏的方向，请对颜色栏使用新的 **Direction** 属性，而不是设置 **XDir** 或 **YDir** 坐标区属性。

```
c = colorbar;  
c.Direction = 'reverse';
```

颜色栏和图例不能为当前坐标区

请勿将颜色栏对象或图例对象传递给需要以坐标区对象作为输入参数的函数。

例如，若将图例对象传递到 **axes** 函数使其成为当前坐标区，会返回一条错误消息：

```
plot(1:10)  
l = legend('line plot');  
axes(l)
```

```
Error using axes  
Handles of type Legend cannot be made the current Axes.
```

类似地，将颜色栏对象传递到 **axes** 函数会返回一条错误消息：

```
c = colorbar;  
axes(c)
```

```
Error using axes  
Handles of type ColorBar cannot be made the current Axes.
```

在以前版本中，您可能会将颜色栏作为当前坐标区，然后再为其定义标题。现在，改用颜色栏的新 **Label** 属性。

```
c = colorbar;  
c.Label.String = 'Colorbar Label';
```

使用新类型属性值查找对象

颜色栏和图例不再具有 'axes' 的 `Type` 属性。请勿使用 `findall` 或 `findobj` 查找带有 'axes' 的 `Type` 属性的对象并预期其返回颜色栏和图例。

若要查找图例，请搜索带有 'legend' 的 `Type` 属性的对象。

```
findall(groot,'Type','legend')
```

若要查找颜色栏，请搜索带有 'colorbar' 的 `Type` 属性的对象。

```
findall(groot,'Type','colorbar')
```

颜色栏和图例没有子级

颜色栏和图例在其 `Children` 属性不再包含底层对象的句柄。它们的 `Children` 属性包括一个空图形占位对象数组。对于图例，改用 `legend` 函数的输出参数访问这些底层对象。

另请参阅

函数

`colorbar` | `legend`

属性

`Colorbar` | `Legend`

如何替换 EraseMode 属性？

从 R2014b 开始，**EraseMode** 属性已从所有图形对象中删除。您仍然可以使用此处所述的技术实现 **EraseMode** 生成的大多数效果，如创建动画或生成叠加颜色。

创建动画

若要通过将数据添加到每个帧以累积形成一张图片，请使用以下方法之一，而不是将 **EraseMode** 属性设置为 'none'：

- 使用 **hold on** 保留当前数据并将新数据添加到图表。
- 使用新的 **animatedline** 函数创建线条动画。
- 使用 **movie** 函数播放录制的影片帧。

例如，使用新的 **animatedline** 函数创建线条动画。

```
theta = linspace(0,2*pi,1000);  
h = animatedline();  
axis([0,2*pi,-1,1])
```

```
for t = theta  
    addpoints(h,t,sin(t));  
    drawnow;  
end
```

有关创建线条动画的详细信息，请参阅 **animatedline** 参考页和 **drawnow** 函数。

显示对象数据的更改

要立即显示对象数据的更改，请调用 **drawnow** 函数，而不是将 **EraseMode** 设置为 'xor'。

例如，更改线条的 **YData** 并显示更新。

```
t = linspace(0,2*pi,10000);  
y = exp(sin(t));  
h = plot(t,y);  
for k = 1:0.01:10  
    y = exp(sin(t.*k));  
    h.YData = y;  
    drawnow  
end
```

生成叠加颜色

若要生成叠加颜色，请使用透明度，而不是将 **Erasemode** 设置为 'xor'。

```
p1 = patch([0,2,2,0],[0,0,2,2],[1,1,1,1]);  
p2 = patch([1,3,3,1],[1,1,3,3],[2,2,2,2]);  
p2.FaceAlpha = 0.5;
```

提高渲染速度

在以前版本中，将 **EraseMode** 属性设置为 'xor' 可提高渲染速度。删除用于设置 **EraseMode** 属性的代码可获得相似的渲染速度。

另请参阅

函数

drawnow | **hold** | **movie**

为何有些对象的子级属性为空？

在 R2014a 和早期版本中，图表对象、图例及颜色栏在其 **Children** 属性包含底层对象的句柄。例如，散点序列在其 **Children** 属性中包含修补对象。更改补片对象的属性值会更改散点序列的外观。

从 R2014b 开始，这些对象在其 **Children** 属性中不包含底层对象的句柄。若要自定义图形，请使用实际对象的属性。此表列出受影响的对象。

对象	R2014a 和较早版本中的子级	R2014b 开始的子级	替代选项
颜色栏	图像对象	0x0 empty GraphicsPlaceholder array	使用颜色栏属性修改颜色栏。
图例	线条、补片和文本对象	0x0 empty GraphicsPlaceholder array	使用 legend 函数的输出参数获得这些对象的句柄。
区域	补片对象	0x0 empty GraphicsPlaceholder array	使用区域属性修改区域。
条形序列	补片对象	0x0 empty GraphicsPlaceholder array	使用条形序列属性修改条形。
等高线	补片对象	0x0 empty GraphicsPlaceholder array	使用等高线属性修改等高线条。
误差条序列	线条对象	0x0 empty GraphicsPlaceholder array	使用误差条序列属性修改误差条。
散点序列	补片对象	0x0 empty GraphicsPlaceholder array	使用散点序列属性修改标记。
阶梯图	线条对象	0x0 empty GraphicsPlaceholder array	使用阶梯图属性修改阶梯。

对象	R2014a 和较早版本中的子级	R2014b 开始的子级	替代选项
针状序列	线条对象	0x0 empty GraphicsPlaceholder array	使用针状序列属性修改针状线条和标记。
箭头序列	线条对象	0x0 empty GraphicsPlaceholder array	使用箭头序列属性修改箭头。

另请参阅

area | bar | colorbar | contour | errorbar | legend | quiver | scatter | stairs | stem

为何同时显示图窗？

如果您在 R2014a 和较早版本的脚本中创建多个图窗，则 MATLAB 会尝试等待每个图窗都显示在屏幕上，然后继续执行脚本。从 R2014b 开始，MATLAB 不用等待图窗显示在屏幕上，即可继续执行脚本。因此，在图窗显示之前，脚本可能已运行结束。对创建多个图窗和执行长时间计算的脚本来说，这些变化是最明显的。

若要强制图窗在创建时即显示，请使用 **drawnow**。

```
figure  
plot(1:10);  
drawnow
```

另请参阅

drawnow

为什么访问刻度标签元素时返回错误消息？

在 R2014a 和较早版本中，坐标区的 `XTickLabel`、`YTickLabel` 和 `ZTickLabel` 属性在字符数组或元胞数组中包含刻度标签值。从 R2014b 开始，这些属性始终在元胞数组中包含刻度标签。使用带花括号 `{}` 的元胞数组索引访问数组元素。

使用矩阵索引访问元素会返回一条错误消息：

```
plot(0:10,0:10);  
ax = gca;  
xticks = get(ax,'XTickLabel');  
xticks(1) = 'start';
```

Conversion to cell from char is not possible.

使用带花括号 `{}` 的元胞数组索引访问刻度标签元素。

```
plot(0:10,0:10);  
ax = gca;  
xticks = get(ax,'XTickLabel');  
xticks{1} = 'start';  
set(ax,'XTickLabel',xticks) % set tick labels to updated values
```

另请参阅

Axes

如何获得对数坐标区的指数值？

从 R2014b 开始，对数轴的 `XTickLabel`、`YTickLabel` 或 `ZTickLabel` 属性包含带有用于刻度标签的完全数 TeX 标记的元胞数组。在 R2014a 和较早版本中，这些属性包含仅含有用于刻度线的指数值的字符数组。

从 R2014b 开始	R2014a 和较早版本
<pre>semilogx(1:10000); ax = gca; ticks = ax.XTickLabel class(ticks)</pre>	<pre>semilogx(1:10000); ax = gca; ticks = get(ax,'XTickLabel') class(ticks)</pre>
<pre>ticks = '10^{0}' '10^{1}' '10^{2}' '10^{3}' '10^{4}' ans = cell</pre>	<pre>ticks = 0 1 2 3 4 ans = char</pre>

若要从刻度标签属性仅提取指数值，请使用 `regexprep` 函数。

```
expression = 'd*^\{(-?d*)\}';  
replace = '$1';  
exponents = regexprep(ticks,expression,replace)
```

```
exponents =  
  
    '0'  
    '1'  
    '2'  
    '3'  
    '4'
```

另请参阅

Axes

为何未复制回调和应用程序数据？

从 R2014b 开始，`copyobj` 不复制与图形对象相关的回调属性或应用程序数据。复制的对象具有设置为空字符数组的回调和设置为空结构体的应用程序数据集。对象副本的行为可能与预期不一致。例如，点击 `uicontrol` 的副本上的普通按钮没有效果。

如果您想要创建具有回调的对象的一个副本，则重新运行用于创建第一个对象的代码以创建第二个对象。

如果您现有的代码使用 `copyobj` 复制回调，则您可以结合使用 `copyobj` 和 `'legacy'` 选项，例如 `c = copyobj(h,p,'legacy')`。`'legacy'` 选项的行为与 R2014b 之前的 MATLAB 版本一致。

另请参阅

`copyobj`

更新 GUI 代码

- “为何有些组件缺失或部分遮盖？”（第 2-2 页）
- “为何 ResizeFcn 的行为发生变化？”（第 2-7 页）
- “为何 handle.listener 返回一个错误？”（第 2-10 页）

为何有些组件缺失或部分遮盖？

本节内容
“变更说明”（第 2-2 页）
“还原编程布局”（第 2-2 页）
“还原 GUIDE 布局”（第 2-4 页）

变更说明

新的图形系统可能看起来缺少坐标区、uicontrol 或 uitable，因为它们被其他组件遮盖。

在以前版本中，**Children** 属性中所列的组件顺序与它们的创建顺序一致。然而，这种顺序不一定与组件在屏幕上的前端到后端定位（或堆叠顺序）一致。

在以前版本中，uicontrol 始终显示在 uipanel 和 uibuttongroup 上方。以前版本还允许坐标区显示在 uipanel 上方，而不作为 uipanel 的子级。

从 R2014b 开始，**Children** 属性中所列的组件顺序与屏幕上的子级堆叠顺序一致。如果您的 UI 包含一个 uipanel 或 uibuttongroup，您可能需要更新代码。

- 相对定位并不足以将坐标区、uicontrol 或 uitable 显示在 uipanel 或 uibuttongroup 之上。若要将一个组件放在另一组件上方，请将其 **Parent** 设置为要在其下方出现的组件。
- Uipanel 和 uibuttongroup 在屏幕上的堆叠顺序行为与 uicontrol 和 uitable 相同。

新的行为反映了 MATLAB 的变化，以实现更加一致的行为。

还原编程布局

此代码创建一个图窗，其中顶层面板包含一个坐标区，底层面板包含一个普通按钮和弹出菜单。

```
hf = figure;  
hb = uicontrol('Style','PushButton',...  
    'String','Plot',...  
    'Position',[175, 40, 60, 25]);  
hpulabel = uicontrol('Style','text',...  
    'String','Plot Type',...  
    'Position',[300, 65, 60, 20]);  
hpu = uicontrol('Style','popupmenu',...
```

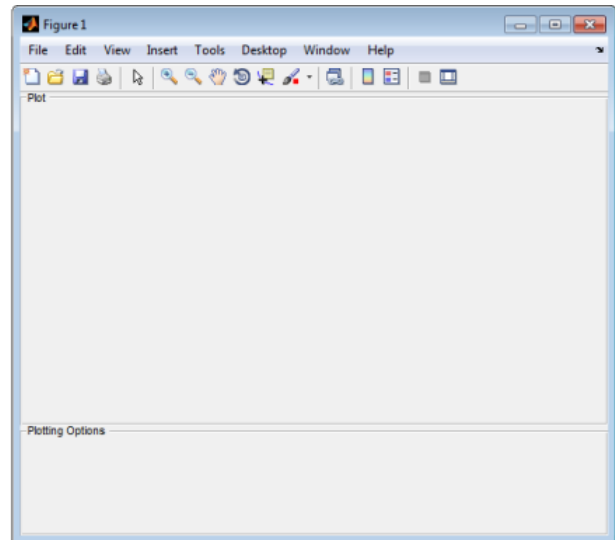
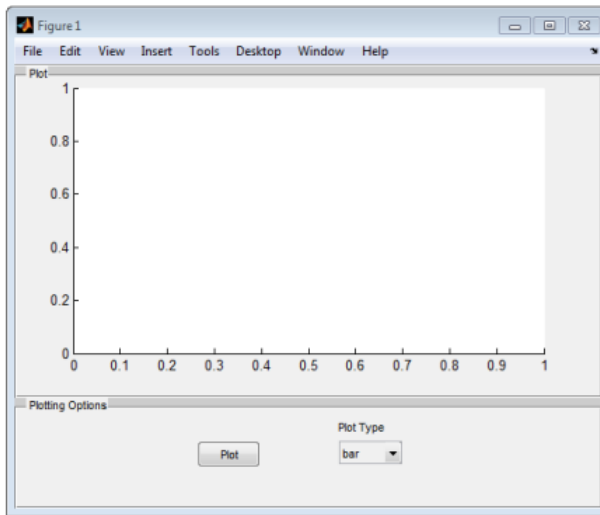


```

        'String', {'bar', 'plot', 'stem'},...
        'Position',[310, 40, 60, 25]);
topp = uipanel('Title', 'Plot',...
        'Position',[0 .25 1 .75]);
ah = axes('Position', [.10, .35 .80 .60]);
bottomp = uipanel('Title','Plotting Options',...
        'Position',[0 0 1 .25]);

```

在以前版本中运行代码会在左侧生成图窗。然而，在新图形系统中运行此函数会在右侧生成图窗。



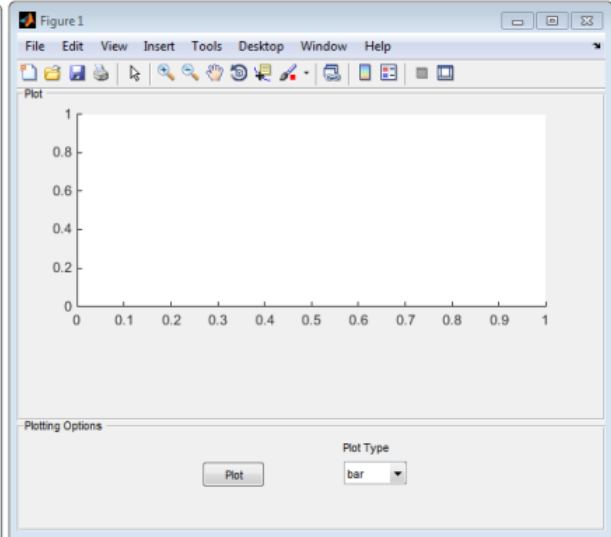
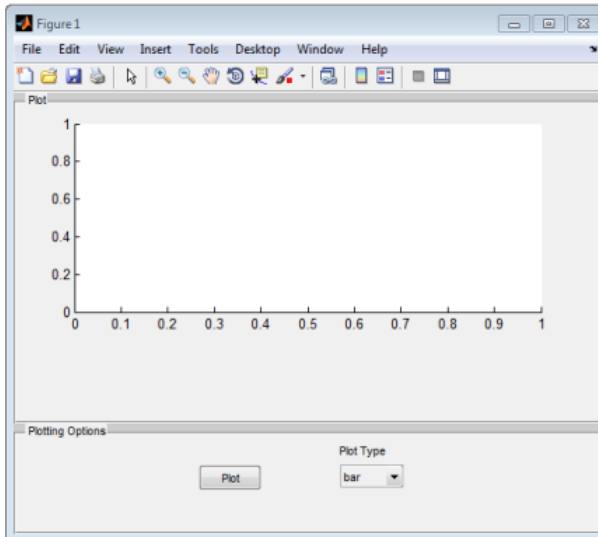
若要确保没有组件隐藏在容器后面，请设置每个组件的 **Parent** 属性使其成为容器的子级。例如，以下代码还原原始 UI。

```

hf = figure;
topph = uipanel('Parent', hf, 'Title', 'Plot',...
        'Position',[0 .25 1 .75]);
axes('Parent', topph, 'Position', [.10, .35 .80 .60]);
bottomp = uipanel('Parent', hf, 'Title','Plotting Options',...
        'Position',[0 0 1 .25])
hpulabel = uicontrol('Parent', bottomp, 'Style','text',...
        'String', 'Plot Type',...
        'Position', [300, 65, 60, 20]);
hb = uicontrol('Parent', bottomp, 'Style','PushButton',...
        'String','Plot',...

```

```
'Position',[175, 40, 60, 25]);  
hpu = uicontrol('Parent', bottomph, 'Style', 'popupmenu',...  
    'String',{'bar', 'plot', 'stem'},...  
    'Position',[310, 40, 60, 25]);
```

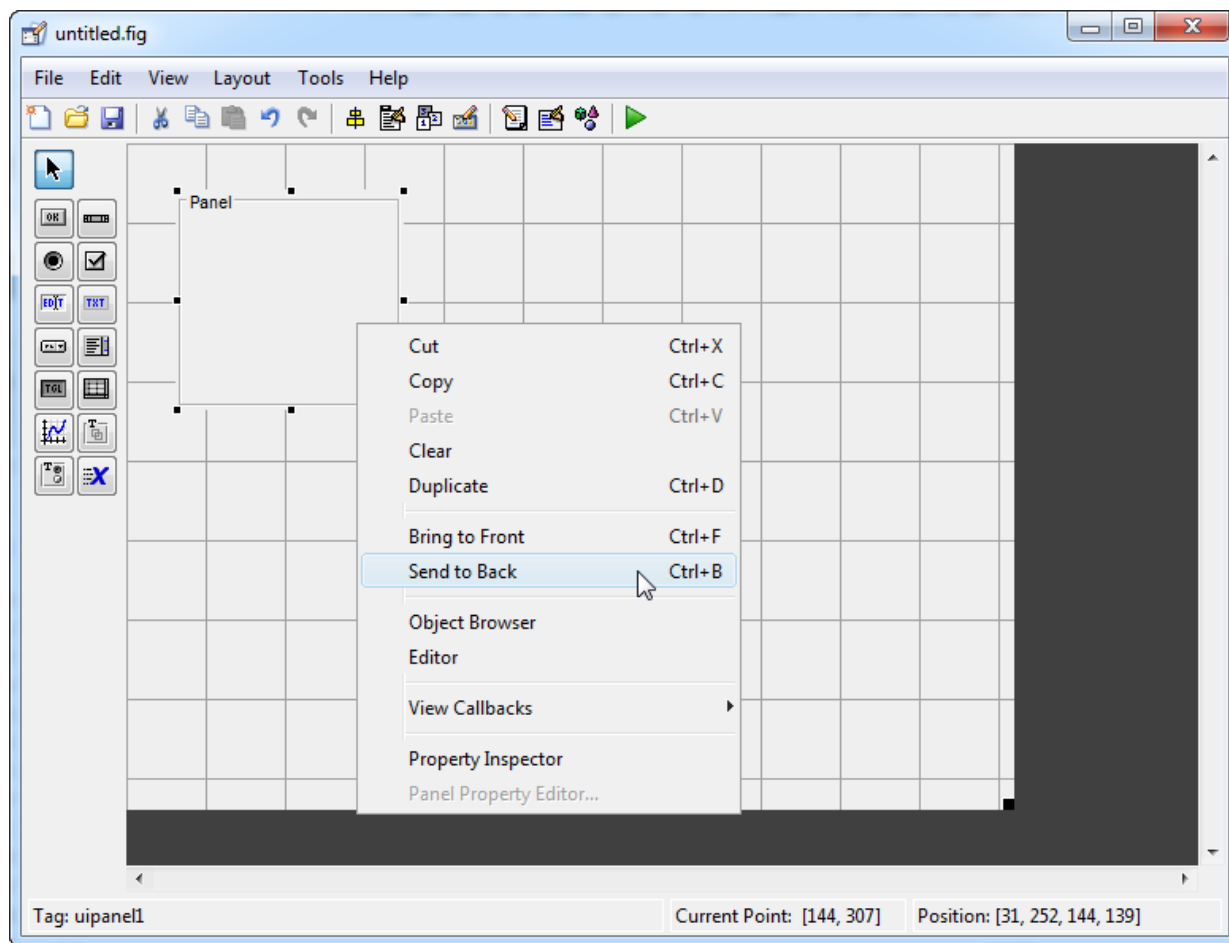


还原 GUIDE 布局

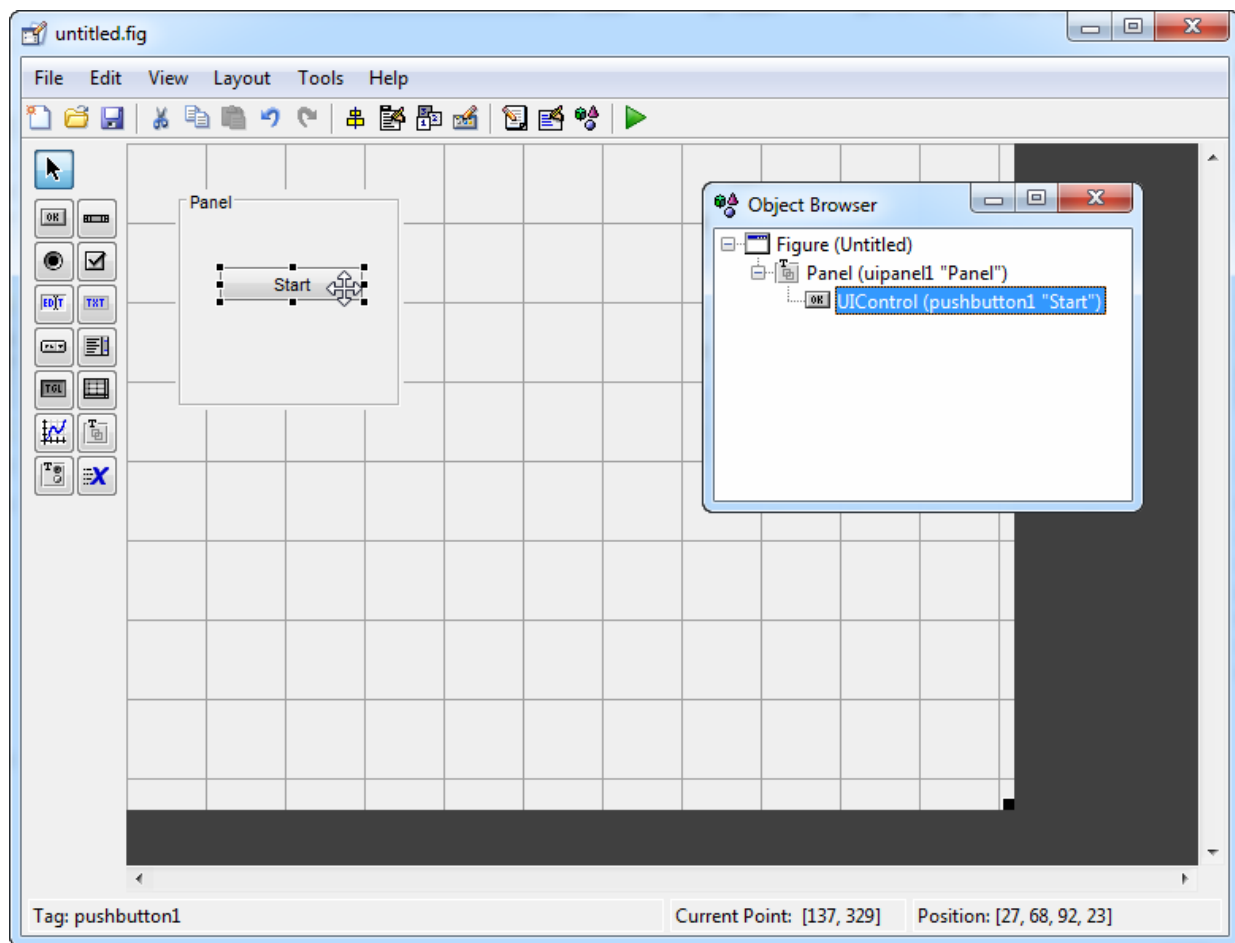
还原 GUIDE UI 的布局需要两个单独的步骤：

- 修复 GUIDE 中的布局。
- 修复组件的子顺序。

若要修复 GUIDE 中的布局，请打开 GUIDE 中的 FIG 文件并使用 **置于底面** 选项以重新安排堆叠。例如，若要将此布局移到所有其他组件后面，右键点击面板并选择 **置于底面**。



若要修复组件的子顺序，使其以与 GUIDE 中相同的方式显示，请选择 **视图 > 对象浏览器**。然后，选择一个组件并将其稍微在面板或按钮组中移动。例如，选择并稍微移动面板中的 **开始按钮** 使其成为面板的子级。



子组件显示在其父级组件上方，所以在程序运行时此 **开始**按钮显示在面板上方。

另请参阅

相关示例

- “以编程方式设置 UI 布局”

为何 ResizeFcn 的行为发生变化？

ResizeFcn 回调的行为的变化反映了图窗和容器的定位和布局的变化，以使其相互之间以及所有平台中的行为更趋一致。

程序启动后 ResizeFcn 返回错误

从 R2014b 开始，ResizeFcn 回调可能在分配程序文件中的所有变量之前执行。在此行为发生时，ResizeFcn 回调会返回错误。

例如，此程序有一个 ResizeFcn 回调，该回调使用 createGUI 函数返回的一个变量。

```
function mygui

    hs = createGUI;

    function handles = createGUI
        % Create figure and its children
        f = figure('Tag','fig',...
            'ResizeFcn',@doResizeFcn,...
            'Visible','off');
        u = uicontrol('Parent',f,'Tag','ctrl');
        handles = guihandles(f);

        % make figure visible
        set(f,'Visible','on');
    end

    function doResizeFcn(varargin)
        length(hs)
    end
end
```

在新的图形系统中运行此程序会导致错误，因为在第一次执行 doResizeFcn 回调时，hs 不存在，而此时图窗变为可见。

```
mygui

Undefined function or variable "hs".
Error in mygui/doResizeFcn (line 18)
    length(hs)

Error using mygui/createGUI (line 14)
Error while evaluating Figure SizeChangedFcn
```

若要纠正此问题，请在分配回调引用的所有变量后使图窗可见。在这一示例中，要在调用 `createGUI` 函数后使图窗可见。

```
function mygui

    hs = createGUI;
    % Make the figure visible
    set(hs.fig, 'Visible','on');

    function handles = createGUI
        % Create figure and its children
        f = figure('Tag','fig',...
            'ResizeFcn',@doResizeFcn,...
            'Visible','off');
        u = uicontrol('Parent',f,'Tag','ctrl');
        handles = guihandles(f);

    end

    function doResizeFcn(varargin)
        length(hs)
    end
end
```

ResizeFcn 对不可见组件处于非活动状态

从 R2014b 开始，更改不可见容器（如图窗、面板、按钮组）的尺寸不会触发 `ResizeFcn` 回调，直到容器变得可见后才触发。

在 MATLAB 的以前版本中，`ResizeFcn` 回调在容器的尺寸发生变化时执行，不管它是否可见。

您可以使用 `Visible` 属性控制图窗和容器的可见性：

- 如果图窗的 `Visible` 属性设置为 `'on'`，则图窗可见。
- 如果 `uipanel` 或 `uibuttongroup` 及其父级的 `Visible` 属性设置为 `'on'`，则 `uipanel` 或 `uibuttongroup` 可见。例如，当 `uibuttongroup`、`uipanel` 和图窗的 `'Visible'` 属性全都设置为 `'on'` 时，父级为 `uipanel` 的 `uibuttongroup` 可见。

外部边界或可绘制区域变化时发生意外行为

从 R2014b 开始，改变图窗或容器的外部边界不会触发 `ResizeFcn` 回调。

例如，在新的图形系统中，当您更改此代码第二行上的 **OuterPosition**（通过删除菜单栏）时，此图窗 **ResizeFcn** 回调不会执行。但是，在以前的 MATLAB 版本中该回调会执行。

```
f = figure('ResizeFcn','display resized');  
set(f,'Menubar','none');
```

从 R2014b 开始，仅当容器的可绘制区域（内部区域）变化时，**ResizeFcn** 回调才执行。当可绘制区域变化时，以前的 MATLAB 版本可能不会执行 **ResizeFcn** 回调。

例如，在新的图形系统中，当您通过增加边框宽度更改 **uipanel** 的可绘制区域时，此 **uipanel** **ResizeFcn** 回调会执行。当您在以前版本中运行此代码时，该回调不执行。

```
p = uipanel('ResizeFcn','display resized');  
set(p,'BorderWidth',3);
```

另请参阅

相关示例

- “以编程方式设置 UI 布局”

为何 `handle.listener` 返回一个错误？

新的图形系统基于 MATLAB 对象，而这些对象不支持使用 `handle.listener` 创建事件侦听程序。若要创建图形对象的侦听程序，请使用 `addlistener`。

注意 版本 2010b 和更高版本支持 `addlistener`，因此您的代码在更新后可在多个版本中运行。
