第九章 句柄图形

句柄图形是对底层图形函数集合的总称，它实际上进行生成图形的工作。这些函数一般隐藏于 M 文件内部，但是它们非常地重要，因为程序员可以利用它对图象或图片的外观进行控制。例如，我们可以利用句柄图形只对 x 轴产生网格线，或选择曲线的颜色为桔黄色，桔黄色 plot 命令中的标准 LineSpec 参数。还有，句柄图形可以帮助程序员为他们的程序创建用户图形界面，用户图形界面，我们将在下一章介绍。

在本章中，我们向大家介绍 **MATLAB** 图形系统的结构，以及如何控制图形对象的属性。

9.1 MATLAB 图形系统

MATLAB 图形系统是建立图形对象的等级系统之上，每一个图形对象都有一个独立的名字，这个名字叫做句柄。每一个图形对象都有它的属性，我们可以通过修改它的属性来修改物体的行为。例如，一条曲线是图形对象的一种。曲线对象有以下的属性:x 数据，y 数据，颜色，线的类型，线宽，符号类型等等。修改其中的一个属性就会改变图象窗口中的一个图象。

由图形命令产生的每一件东西部是图形对象。例如，图形中的每一个曲线，坐标轴和字符串是独立的对象(拥有独立的名字句柄，还有形式)。所有的图象对象按**子对象**和**父对象**的形式管理，如图 9.1 所示。当一个子对象被创建时，它可能继承了父对象的许多属性。

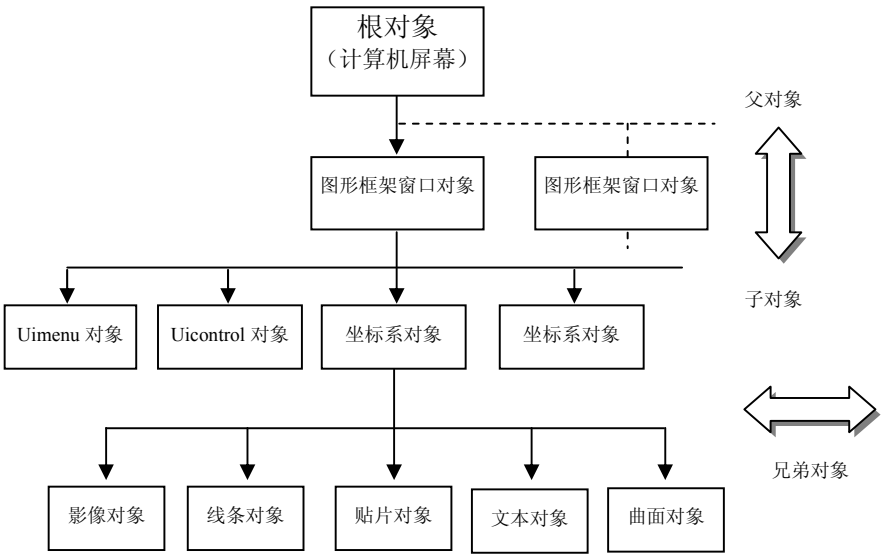


图 9.1 对象的层次结构

在 **MATLAB** 中最高层次的图形对象被根对象，我们可以通过它对整个计算机屏幕进行控制。当 **MATLAB** 启动时，根对象会被自动创建，它一直存在到 **MATLAB** 关闭。与根对象相关的属性是应用于所用 **MATLAB** 窗口的默认属性。

在根对象下，有多个图象窗口，或只有图象。每一个图象在用于显示图象数据的计算机屏

幕上都有一个独立的窗口，每一个图象都有它独立的属性。与图象相关的属性有，颜色，图片底色，纸张大小，纸张排列方向，指针类型等。

每一个图形可包括四个对象:Uimenu 对象，Uicontrol 对象，坐标系对象和 Uicontextmenus 对象。Uimenu 对象，Uicontrol 对象，和 Uicontextmenus 对象是专门地用来创建用户图形界面的对象，它们将在下一章讨论。坐标系对象是指在用于显示图象的图片中的区域。在一个图象窗口中，它可能含有一个或多个坐标系。

每一个坐标系对象可能包括曲线对象，文本对象，贴片对象，还有其他的你所需的图形对象。

9.2 对象句柄

每一个图象对象都有一个独一无二的名字，这个名字叫做句柄。句柄是在 **MATLAB** 中的一个独一无二的整数或实数，用于指定对象的身份。用于创建一个图象对象的任意命令都会自动地返回一个句柄。例如，命令

>>Hnd1 = figure;

创建一个新的图象，并返回这个图象的句柄到变量 Hnd1。根对象句柄一般为 0，图象(图)对象的句柄一般是一个小的正整数，例如 1，2，3……而其他的图形(graphic)对象为任意的浮点数。

我们可以利用 **MATLAB** 函数得到图象，坐标系和其他对象的句柄。例如，函数 gcf 返回当前

图象窗口的句柄，而函数 gca 则返回在当前图象窗口中的当前坐标系对象的句柄，函数 gco 返回当前选择对象的句柄。这些函数将会在后面将会被具体讨论。

为了方便，存储句柄的变量名要在小写字母后面个 H。这样就可以与普通变量(所有的小写变量，大写变量，全局变量)区分开来。

9.3 对象属性的检测和更

对象属性是一些特殊值，它可以控制对象行为的某些方面。每一个属性都有一个属性名和属性值。属性名是用大小写混合格式写成的字符串，属性名中的每一个单词的第一个字母为大写，但是 **MATLAB** 中的变量名的大小不与区分。

9.3.1 在创建对象时改变对象的属性

当一个对象被创建时，所有的属性都会自动初始化为默认值。包含有"propertyname(属性名)"的创建函数创建对象时，默认值会被跳过，而跳过的值在创建函数中有。例如，我们在第二章看到，线宽属性可以通过下面的 plot 命令改变。

plot(x, y, 'LineWidth', 2);

录一个曲线被创建时，函数用值 2 来替代它的默认值。

9.3.2 对象创建后改变对象的属性

我们可以用随时用 get 函数检测任意一个对象的属性，并用 set 函数对它进行修改。get 函数最常见的形式如下

value = get(handle, 'PropertyName');

value = get(handle);

value 是勤句柄指定对象的属性值。如果在调用函数时，只有一个句柄，那么函数将会返回一个结构，域名为这个对象的属性名，域值为属性值。

set 函数的最常用形式为

set(handle,'PropertyName1', value1, ...);

在一个单个的函数中可能有多个"propertyname"和"value"。

例如，假设我们用下面的语句，画出函数 $y(x)= x^2$ 在(0，2)中的图象

```
x = 0:0.1:2;
y = x .^2;
Hnd1 = plot(x, y);
```

图象如图 9.2a 所示。这个曲线的句柄被存储在变量 Hnd1 内，我们可以利用它检测和修改这条曲线的属性。函数 get(Hnd1)在一个结构中返回这条曲线所有的属性，每一个属性名都为结构的一个元素。

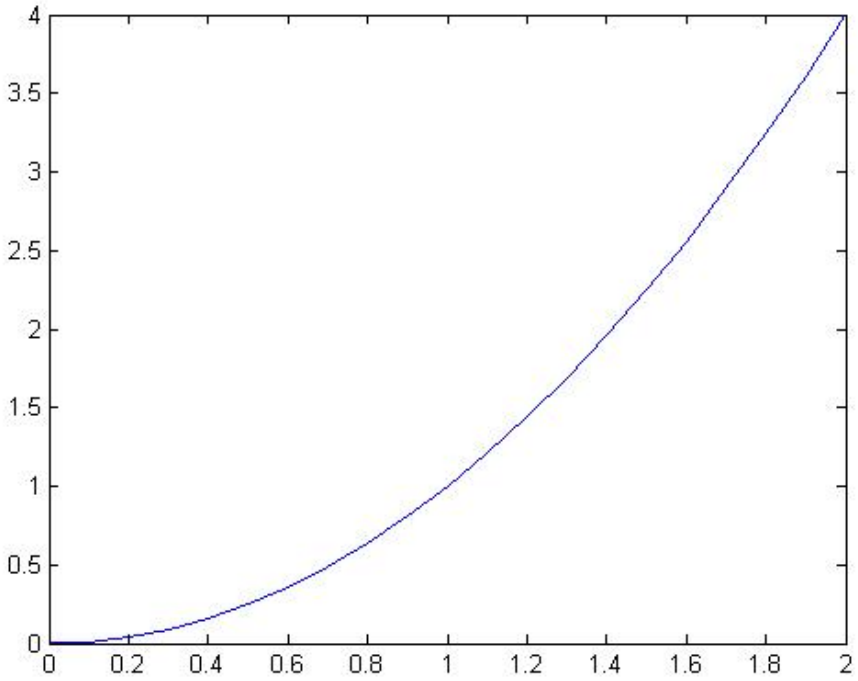
```
>> result=get(Hnd1)

result =

                    Color: [0 0 1]
                EraseMode: 'normal'
                LineStyle: '-'
                LineWidth: 0.5000
                   Marker: 'none'
               MarkerSize: 6
          MarkerEdgeColor: 'auto'
          MarkerFaceColor: 'none'
                    XData: [1x21 double]
                    YData: [1x21 double]
                    ZData: [1x0 double]
            BeingDeleted: 'off'
           ButtonDownFcn: []
                 Children: [0x1 double]
                 Clipping: 'on'
                CreateFcn: []
                DeleteFcn: []
              BusyAction: 'queue'
        HandleVisibility: 'on'
                  HitTest: 'on'
           Interruptible: 'on'
                 Selected: 'off'
       SelectionHighlight: 'on'
                      Tag: ''
                     Type: 'line'
          UIContextMenu: []
                 UserData: []
                  Visible: 'on'
```

```
          Parent: 151.0012
     DisplayName: ''
       XDataMode: 'manual'
     XDataSource: ''
     YDataSource: ''
     ZDataSource: ''
```

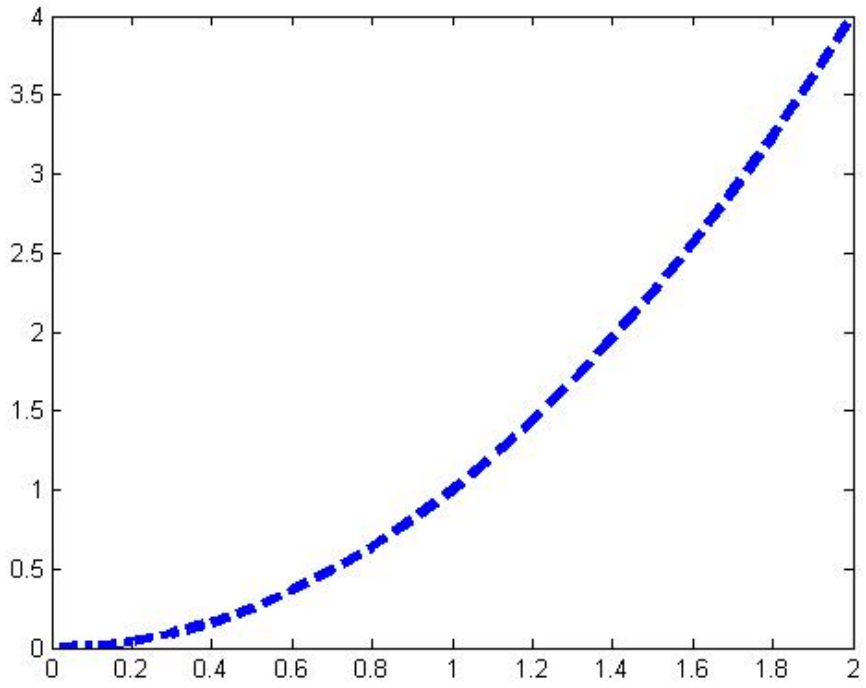
注意当前曲线的线宽为 0.5pixel，线型为虚线。我们能够用这些命令改变线型和线宽。

```
set(Hnd1,'LineWidth',4,'LineStyle','--')
```

产生的结果图象如 9.2b 所示。



(a)



(b)

图 9.1(a)用默认值画出的函数 $y= x^2$ 的图象，(b)修改了线宽和线型的函数图象

函数 get 和 set 对程序员来说非常的有用，因为它们可以直接插入到 **MATLAB** 程序中，根据用户的输入修改图象。在下一章，我们把它用于 GUI 编程。

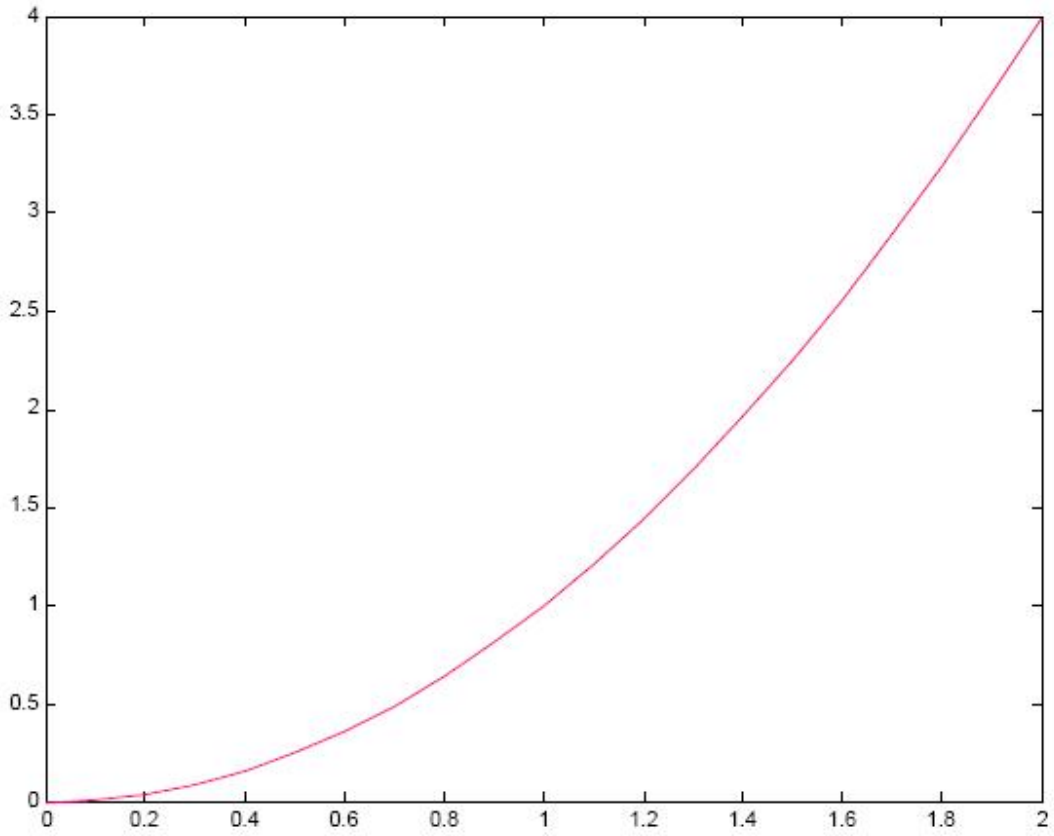
但对于最终的用户，他们要很容易地改变 **MATLAB** 对象的属性。属性编辑器是为了这个目的而设计的工具。启动属性编辑器的命令为

```
propedit(HandleList);
propedit;
```

这个函数第一个形式用于编辑所列出的句柄的属性，而这个函数的第二种形式用于编辑当前图象的属性。例如下面的语句创建函数 y(x)=在(0，2)x2 中的图象并打开属性编辑器，让用户间接地改变曲线的属性。

```
figure(2);
x = 0:0.1:2;
y = x .^2;
Hnd1 = plot(x, y);
propedit(Hnd1);
```

我们用这些语句调用属性编辑器，如图 9.3。属性编辑器包含了许多的窗格，用户可以根据对象的类型改变对象的属性。在例子中讨论的曲线对象，它窗格包括"Date"，"Style"，和"Info"。"Date"窗格允许用户选择和修改所要显示的数据它可以修改 X 数据，Y 数据和 Z 数据的属性。"Style"窗格用来线型和符号属性，"Info"用来设置曲线对象的其它信息。


4
3.5
3
2.5
2
1.5
1
0.5
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

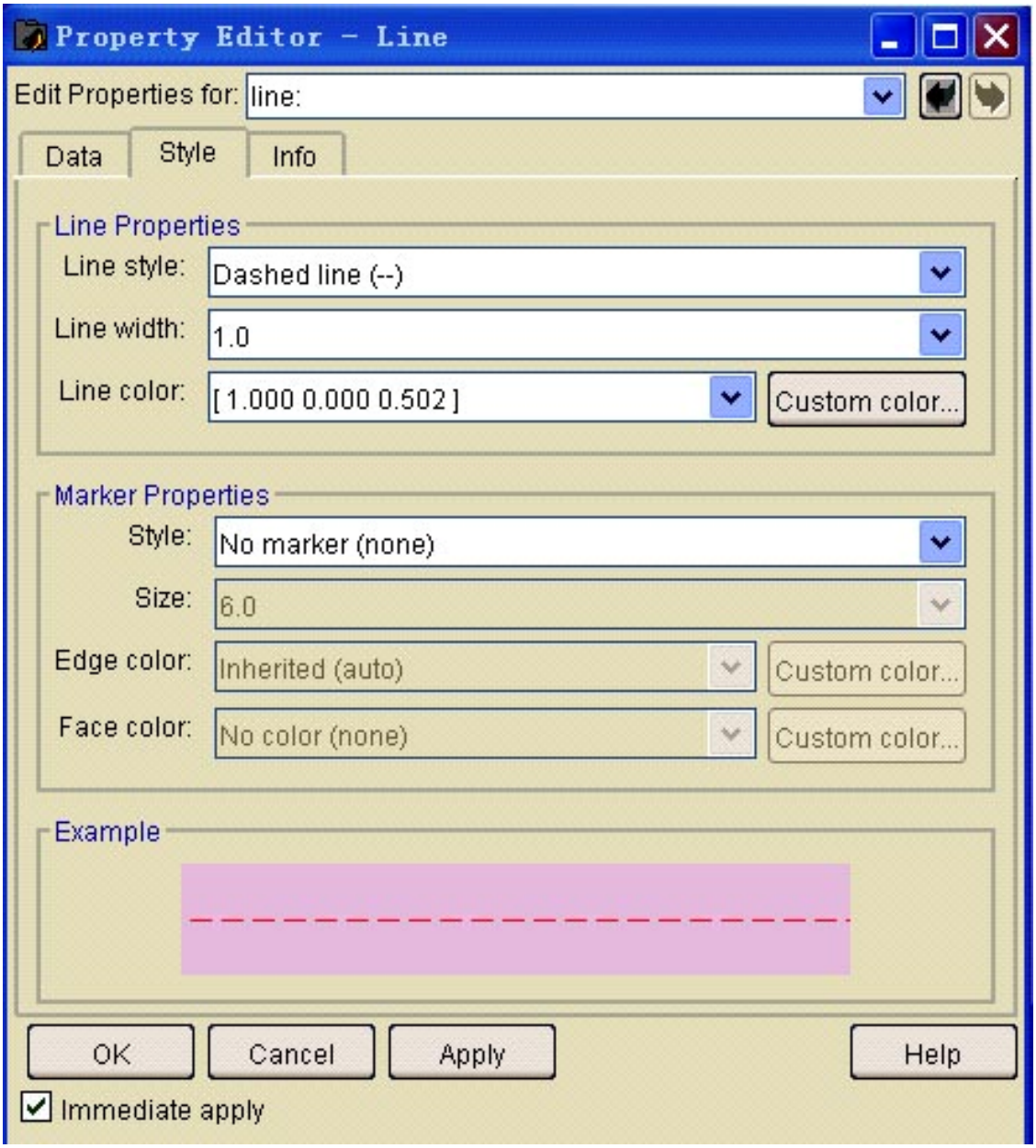


图 9.3 编辑曲线对象的属性编辑器。

属性编辑器可以用图象工具条上的按钮调用，然后双击你所要编辑的对象。

例 9.1

底层图形命令的应用

函数 sinc(x)的定义如下

$$\operatorname{sinc} x = \begin{cases} \dfrac{\sin x}{x} & x \neq 0 \\ 1 & x = 0 \end{cases}$$

x 的取值从-3π 到 3π，画出这个函数的图象，用句柄图形函数画出图象，满足下面的要求

1.使图象背景为粉红色

2.只在 y 轴上有网格线

3.曲线为 3point 宽，桔黄色的实线

答案

为了创建图象，我们需要计算 x 从-3π 到 3π 之间的函数 sincx，然后用 plot 命令画出它的图象。plot 命令这条直线的句柄。

画完直线后，我们需要修改 figure 对象的颜色和 axes 对象的网格状态，以及 line 对象的颜色与线宽。这些修改需要我们访问图对象，axes 对象，line 对象的句柄。图对象的句柄由函数 gcf 返回，axes 对象的句柄由函数 gca 返回，line 对象由 plot 函数返回。

需要修改的底层图形属性可以在 **MATLAB** 在线帮助工作台文件中找到，在主题"Handle Graphics Objects"目录下。它们包括当前图象的"color"属性，当前的坐标系的"YGrid"属性，以及曲线的"LineWidth"属性和"color"属性。

1.陈述问题

画出函数 sincx 的图象，x 的取值从-3π 到 3π，图象背景为粉红色，只在 y 轴上有网格线，曲线为 3point 宽，桔黄色的实线

2.定义的输入与输出

这个程序无输入，输出为指定的图象。

3.设计算法

这个问题可分为三大步

Calculate sinc(x)

Plot sinc(x)

Modify the required graphics object properties

这个程序的第一大步是计算 x 从-3π 到 3π 之间的函数 sincx。这个工作可以用向量化语句完成，但向量化语句在 x=0 时会产生一个 NaN，因为 0/0 没有意义。在画这个函数之前我们必须用 1.0 替换 NaN。这个步骤的伪代码为

```
% Calculate sinc(x)
x = -3*pi:pi/10:3*pi
y = sin(x) ./ x
%Find the zero value and fix it up. The zero is
%located in the middle of the x array.
index = fix(length(y)/2) + 1
y(index) = 1
```

下一步，我们必须画出这个函数的图象，并把所要修改的曲线的句柄存入一个变量。这个步骤的伪代码为

```
Hndl = plot(x, y);
```

现在，我们必须用句柄图形修改图象背景，y 轴上的网格线，线宽和线色。图对象的句柄由函数 gcf 返回，axes 对象的句柄由函数 gca 返回，line 对象由 plot 函数返回。

粉红色背景可由 RGB 向量[1 0.8 0.8]创建，桔黄色曲线可以由 RGB 向量[1 0.5 0]创建。

伪代码如下

```
set(gcf, 'Color', [1 0.8 0.8])
set(gca, 'YGrid', 'on')
set(Hndl, 'Color', [1 0.5 0], 'LineWidth', 3)
```

4.把算法转化为 **MATLAB** 语言

```
% Script file: plotsinc.m
%
% Purpose:
% This program illustrates the use of handle graphics
% commands by creating a plot of sinc(x) from -3*pi to
```

```
% 3*pi, and modifying the characteristics of the figure,
% axes, and line using the "set" function.
%
% Record of revisions:
% Date    Programmer    Description of change
% ====    ==========    =====================
% 11/22/97 S. J. Chapman Original code
%
% Define variables:
% Hndl -- Handle of line
% x -- Independent variable
% y -- sinc(x)
% Calculate sinc(x)
x = -3*pi:pi/10:3*pi;
y = sin(x) ./ x;
% Find the zero value and fix it up. The zero is
% located in the middle of the x array.
index = fix(length(y)/2) + 1;
y(index) = 1;
% Plot the function.
Hndl = plot(x,y);
% Now modify the figure to create a pink background,
% modify the axis to turn on y-axis grid lines, and
% modify the line to be a 2-point wide orange line.
set(gcf,'Color',[1 0.8 0.8]);
set(gca,'YGrid','on');
set(Hndl,'Color',[1 0.5 0],'LineWidth',3);
```

5.检测程序

这个程序的检测是非常简单的，我们只要运行这个程序，并检查产生的图象就可以了。产生的图象如图 9.4 所示，它就是我们需要的样式。

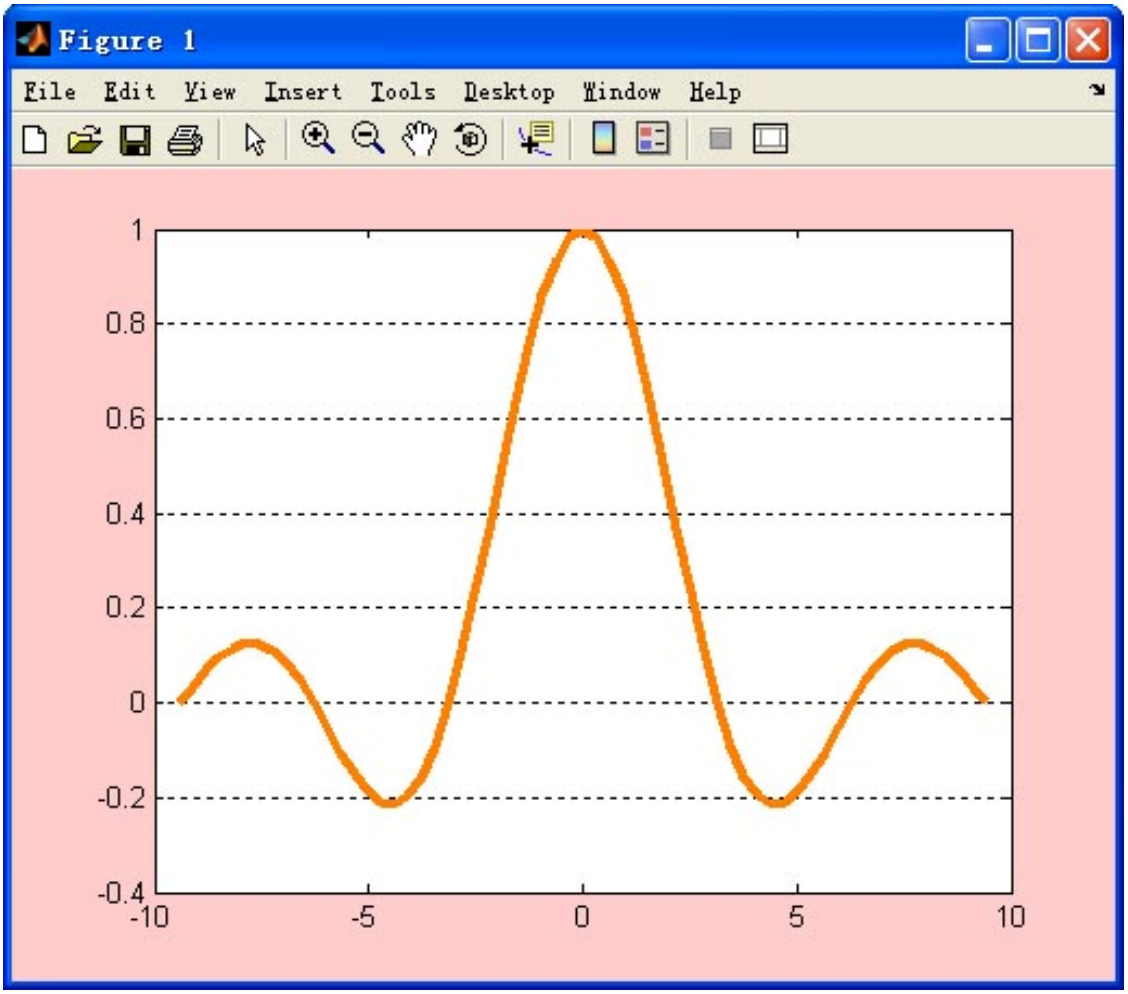


图 9.4 sincx 的图象

9.4 用 set 函数列出可能属性值

函数用于提供所有可能的属性值列表。如果在调用函数 set 时，只包括属性名而不包括相应的属性值，那么函数 set 就会返回所有的合法属性值。例如，命令 set(Hnd1，"LineStyle")将返回所有可能的线型，大括号中是默认的线型。

```
>> set(Hndl,'LineStyle')
[ {-} | -- | : | -. | none ]
```

这个函数的合法包括和"none"，第一个是默认的类型。

```
>> set(Hndl,'LineWidth')
A line's "LineWidth" property does not have a fixed set of property values.
```

函数 set(Hnd1)返回一个对象的所有属性的所有可能的属性值。

```
>> set(Hndl)

ans =

                    Color: {}
               EraseMode: {4x1 cell}
```

```
                LineStyle: {5x1 cell}
                LineWidth: {}
                   Marker: {14x1 cell}
              MarkerSize: {}
         MarkerEdgeColor: {2x1 cell}
         MarkerFaceColor: {2x1 cell}
                   XData: {}
                   YData: {}
                   ZData: {}
           ButtonDownFcn: {}
                Children: {}
                Clipping: {2x1 cell}
               CreateFcn: {}
               DeleteFcn: {}
             BusyAction: {2x1 cell}
       HandleVisibility: {3x1 cell}
                HitTest: {2x1 cell}
          Interruptible: {2x1 cell}
                Selected: {2x1 cell}
    SelectionHighlight: {2x1 cell}
                    Tag: {}
          UIContextMenu: {}
               UserData: {}
                Visible: {2x1 cell}
                 Parent: {}
            DisplayName: {}
              XDataMode: {2x1 cell}
             XDataSource: {}
             YDataSource: {}
             ZDataSource: {}
```

9.5 自定义数据

除了一个 GUI 对象定义的标准属性以外，程序可以定义所要控制的数据的特殊属性。程序员可以用附加属性把任意类型的数据添加到 GUI 对象中。任意数量的数据可以被存储，并应用于各种目的。

自定义数据可以用近似标准属性的形式存储。每一个数据条目都有一个名字和值。数据变量可以用函数 setappdate 存储在一个对象，并用函数 getappdata 接收。

setappdate 函数的基本形式如下

setappdata(Hndl, 'DataName', DataValue);

其中 Hndl 是数据存入的对象的句柄，"DateName"是这个数据的名字，而 DateValue 是赋于是这个名字的值。注意数据值可以是数字，也可以是字符串。

例如，假设我们要定义两个特殊的数据值，其中一个用于存储发在指定图象中的错误数，另一个是用于描述最后发现的错误的字符串。这两个数据值的名字是"ErrorCount"和"LastError"。我们假设 H1 为这个图象的句柄，创建这些数据条目和初始化的命令为

setappdata(Hl,'ErrorCount',0);

setappdata(H1,'LastError','No error');

我们可以用 getappdata 函数随时调用这些数据。getappdata 的两种形式如下

value = getappdata(Hndl, 'DataName');

struct = getappdata(Hndl);

其中，Hnd1 是包含有这个数据的对象句柄，"DateName"是要调用的数据的名字，如果一个"DateName"被指定，那么与"DateName"相关的值就会被返回。如果没有被指定，那么所有与这个对象形字相关的自定义值就会以结构的形式被返回。数据条目名就是结构元素名。

对上面的例子来说，getappdata 将会产生下面的结果

```
>> value = getappdata(Hl, 'ErrorCount')
value =
        0
>> value = getappdata(Hl);
struct =
    ErrorCount:    0
    LastError:    'No error'
```

与自定义数据相关的函数被总结在表 9.1 中。

表 9.1 与自定义数据相关的函数

函数	描述
setappdata(Hndl, 'DataName', DataValue)	把 DataValue 存储在对象中的'DataName'，这个对象以 Hndl 为句柄。
value = getappdata(Hndl, 'DataName') struct = getappdata(Hndl)	从以 Hndl 句柄的对象重新调用程序，第一种形式只读取'DataName'中的数据，第二种形式重新所有的自定义数据。
isappdata(Hndl, 'DataName')	如果'DataName'在以 Hndl 为句柄的对象中有定义，那就会返回 1，否则返回 0。
isappdata(Hndl, 'DataName')	删除'DataName'，'DataName'是在以 Hndl 为句柄的对象中的自定义数据。

9.6 对象查找

每一个新的图象在从创建开始时就有它们自己的句柄，句柄可以由创建函数返回。

好的编程习惯

如果你打算修改你创建的对象的属性，那么请保存对象的句柄，为以后调用函数 get 和 set 做准备。

但是我们有时不能访问句柄。假设我们由于一些原因，丢失了对象的句柄。我们如何检测和图形对象呢?**MATLAB** 提供了四个专门的函数，用来帮助寻找对象的句柄。

- gcf 返回当前图象的句柄
- gca 返回当前图象中当前坐标系的句柄
- gco 返回当前对象的句柄
- findobj 寻找指定属性值的图形对象

函数 gcf 返回当前图象的句柄。如果这个图象不存在，gcf 将会创建一个，并返回它的句柄。函数 gca 返回当前图象中当前坐标系的句柄，如果图象不存在，或当前图象中无坐标系，那么函数 gca 将创建一个坐标系，并返回它的句柄。函数 gco 的形式如下

```
H_obj = gco;
H_obj = gco(H_fig);
```

其中，H_obj 是一个对象的句柄，H_fig 是一个图象的句柄。这个函数的第一种形式返回当

前图象中的当前对象的句柄。它的第二种形式返回一指定图象中的当前对象的句柄。

当前对象是指用鼠标单击的最一个对象。这个对象可以是除了根对象的任意图形对象。直到鼠标在图象内发生了单击事件，在图象内才有一个当前对象。在单击事件发生之后，函数 gco 将返回一个空数组[]，不像函数 gcf 和 gca，gco 如果不存在就自动创建。

一旦我们得知了一个对象的句柄，我们可以通过检测"Type"属性去时来确定对象的类型。"Type"属性是一个字符串，例如"图"，"line"，"text"等等。

```
H_obj = gco;
type = get(H_obj, 'Type')
```

查找任意一个 **MATLAB** 对象最简单的方法是用 findobj 函数。它的基本形式如下

```
Hndls = findobj('PropertyName1',value1, ...)
```

这个命令起始于根对象，并搜索所有的对象，找出含有指定属性，指定值的对象。注意可以指定多个属性/值，findobj 只返回与之匹配的对象句柄。

例如，假设我们已经创建了图 1 和图 3。那么函数 findobj("Type"，"图")将会返回结果

```
>> H_fig = findobj('Type', 'figure')
H_fig =
     3
     1
```

函数 findobj 的这种形式非常的有用，但却比较慢，因为它必须对整个对象树进行搜索。如果你必须多次用到一对象，只调用一次函数 findobj，为了重复利用句柄，句柄应存储下来。

限定搜索对象的数目能够加快函数运行的速度。它的形式为

```
Hndls = findobj(SrchHndls, 'PropertyName1', value1, ...)
```

在这里，只有数组 srchHndls 和它的子数组中的句柄，才在搜索的范围内。例如你想找到图 1 中的虚线。它的命令为

```
Hndls = findobj(1, 'Type', 'line', 'LineStyle', '--');
```

好的编程习惯

如果有可能的话，限定函数 findobj 的搜索范围将能加快函数的运行速度。

9.7 用鼠标选择对象

函数 gco 将返回当前对象，当前对象是指用鼠标最后一次单击的对象。每一个对象都有一个与之相关的**可选择区**，在可选择区内任意一个单击都可以看作对这个对象的单击。对于细小的对象(例如线，点)来说，这种特性是非常重要的。可选择区的宽度和形状由对象的类型确定。例如，一个曲线的可选择区在离直线 5pixel 的范围内，而一个表面，一个小块和文本对象的可选择区是包含这些对象的最小长方形。

对于一个坐标系对象来说，它的可选择区是坐标轴区域加上标题和标签的区域。但是在坐标轴内的曲线对象或其他对象有更高的优先权，你必须在单击坐标内的一点，并且不靠近直线和文本。如果单击坐标外的图象将会选择图象本身。

如果一个用户单击了两个或多个对象的所在点，例如两线的交插点将会有什么事情发生。这取决于每一个对象**堆垛顺序(stacking order)**。堆垛顺序是 **MATLAB** 选择对象的顺序。在一个图象中所有的"子对象"属性句柄顺序就是堆垛顺序。如果单击了两个或多个对象的所在点，在堆垛顺序的优先权高的将会被选择。

当选择图形对象时，我们有时可以调用 **MATLAB** 内建函数 waitforbuttonpress。这个函数的形式为

```
k = waitforbuttonpress
```

当这个函数运行时，它将会暂停程序，直到任意键按下或鼠标单击事件发生后，程序才恢

复运行。如果按下了鼠标键函数将会返回 0，按下任意键，函数将会 1。

函数经常用于暂停程序。当鼠标单击事件发生后，程序将会用 gco 函数恢复选择对象的句柄。

例 9.2

图形对象的选择

在本例中的程序可以探测图形对象的属性，并显示如何用函数 waitforbuttonpress 和 gco 选择对象。程序允许用户可以多次重复选择对象。

```
% Script file: select_object.m
%
% Purpose:
% This program illustrates the use of waitforbuttonpress
% and gco to select graphics objects. It creates a plot
% of sin(x) and cos(x), and then allows a user to select
% any object and examine its properties. The program
% terminates when a key press occurs.
%
% Record of revisions:
% Date Programmer Description of change
% ==== ========== ====================
% 11/23/97 S. J. Chapman Original code
%
% Define variables:
% details -- Object details
% H1 -- Handle of sine line
% H2 -- Handle of cosine line
% Handle -- Handle of current object
% k -- Result of waitforbuttonpress
% type -- Object type
% x -- Independent variable
% y1 -- sin(x)
% y2 -- cos(x)
% yn -- Yes/No
% Calculate sin(x) and cos(x)
x = -3*pi:pi/10:3*pi;
y1 = sin(x);
y2 = cos(x);
% Plot the functions.
H1 = plot(x,y1);
set(H1,'LineWidth',2);
hold on;
H2 = plot(x,y2);
set(H2,'LineWidth',2,'LineStyle',':','Color','r');
title('\bfPlot of sin \itx \rm\bf and cos \itx');
xlabel('\bf\itx');
ylabel('\bfsin \itx \rm\bf and cos \itx');
legend('sine','cosine');
hold off;
% Now set up a loop and wait for a mouse click.
```

```
k = waitforbuttonpress;
while k == 0
    % Get the handle of the object
    Handle = gco;
    % Get the type of this object.
    type = get(Handle,'Type');
    % Display object type
    disp (['Object type = ' type '.']);
    % Do we display the details?
    yn = input('Do you want to display details? (y/n) ','s');
    if yn == 'y'
        details = get(Handle);
        disp(details);
    end
    % Check for another mouse click
    k = waitforbuttonpress;
end
```

程序运行后，得到的结果如图 9.5 所示

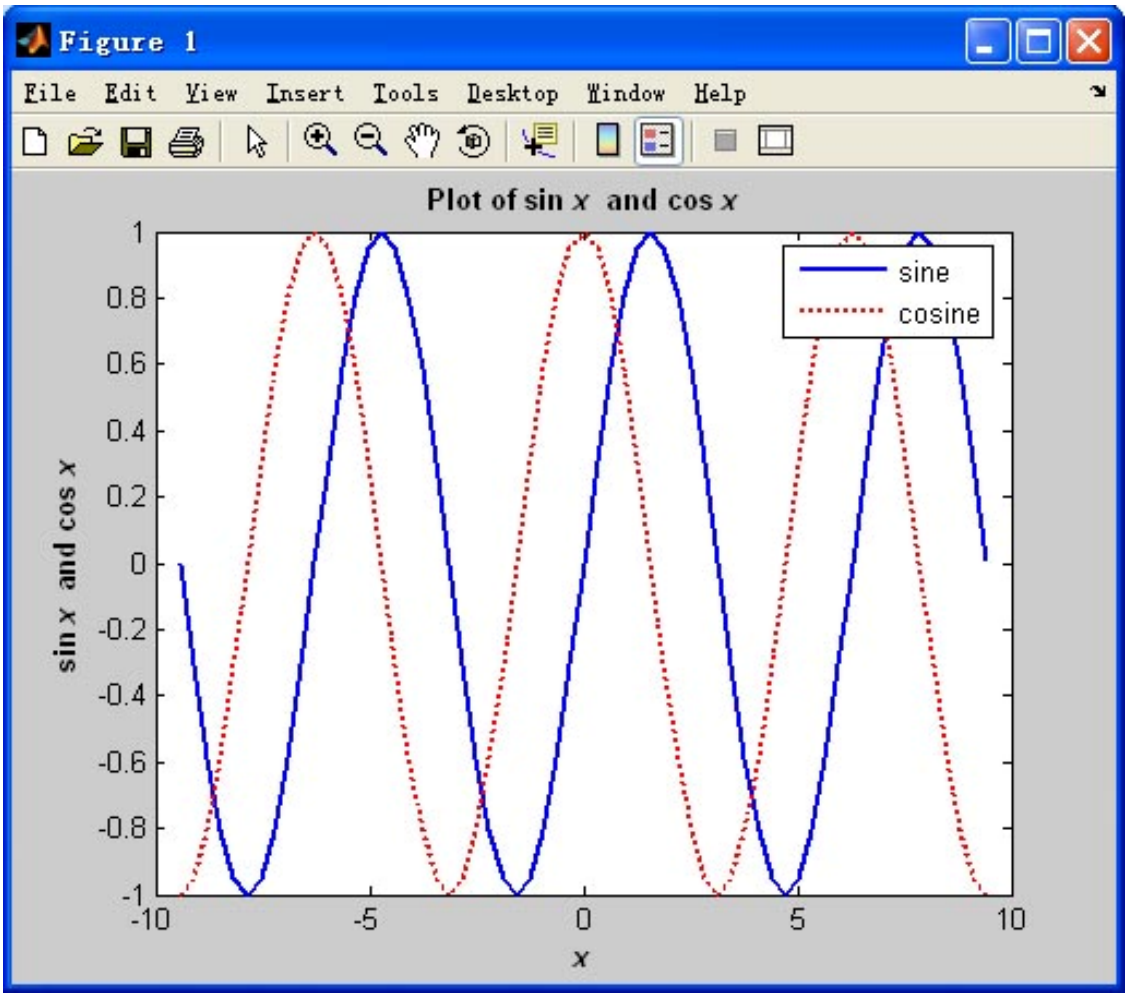


图 9.5 sinx 和 cosx 的图象。

我们可以在图象内单击各种对象，并查看它的类型。

9.8 位置和单位

许多的 **MATLAB** 对象都包括位置("position")属性，它用来指定对象在计算机屏幕的位置和大小。这个属性在不同类型的对象中有细节的差别，这一点将在本节中描述。

9.8.1 图象(figure)对象的位置

一个图象(图)的位置("position")用一个 4 元素行向量指定在计算机屏幕内的位置。在这个向量中的值为[left bottom width height]，其中 left 是指图象的左边界，bottom 是指图象的底边界，width 是指图象的宽度，height 是指图象的高度。它的这些位置值的单位可以用对象的"Units"属性指定。例如，与当前图象的位置和单位可以用下面的语句得到。

```
>> get(gcf,'Position')
ans =
    128     259     506     373
>> get(gcf,'Units')
ans =
pixels
```

这些信息说明当前图象窗口的左下角距屏幕右边的距离为 176pixel，距屏幕底边的距离为 204pixel。，图象的宽度为 672pixel，上下高度为 504pixel。注意这是图象的可作图区，包括边界，滚动条，菜单，还有图象的标题区。

单位("units")属性的默认值为像素(pixels)，但是它的属性值还可以为英尺(inches)，公分(centimeters)，点(points)，或归一化坐标(normalixed coordinates)。像素代表了屏幕像素，即在屏幕上可表示出来的最小的对象。典型的计算机屏幕最小分辨为 640×480，在屏幕的每一个位置都有超过 1000 的像素。因为像素数因机算机屏幕的不同而不同，所以指定对象的大小也会随之改变。

归一化坐标是在 0 到 1 范围内。在归一化坐标中，屏幕的左下角为[0,0]右上角为[1.0, 1.0]。

如果对象的位置归一化坐标系的形式描述，那么不同分辨率的显示器上对象的相对位置是固定的。例如，下面的语句创建了一个图象，把图象放置在屏幕的上部，而不用考虑显示器的大小。

```
H = figure(1)
set(H,'units', 'normalized','position',[0 .5 .5 .45])
```

好的编程习惯

如果你想把对象放置在窗口的特定位置，最好的方法是用归一化坐标，因为不用考虑显示器的大小。

9.8.2 坐标系对象和 uicontrol 对象的位置

坐标系对象和 uicontrol 对象的位置同样可以用一个 4 元素向量表示，但它是相对于 figure 对象的位置。一般说来，所有子对象的"position"属性都与它的父对象相关。

默认地，坐标系对象在一图象内的位置是有归一化单位指定的，(0，0)代表图象的左下角，(1，1)代表图象的右上角。

9.8.3 文本(text)对象的位置

与其他对象不同，文本(text)对象有一个位置属性，包含两个或三个元素。这些元素为坐标系对象中文本对象的 x，y 和 z 轴。注意都显示在坐标轴上。

放置在某一特定点的文本对象的位置可由这个对象的 HorizontalAlignment 和 VerticalAlignment 属性控制。HorizontalAlignment 的属性可以是{Left}。Center，或 Right。VerticalAlignment 的属性值可以为 Top，cap，{Middle}，Baseline 或 Bottom。

文本对象的大小由字体大小和字符数决定，所以没有高度和宽度值与之相连。

例 9.3

设置一个图象内对象的位置

正如我们前面所提到的，坐标系的位置与包含它的图象窗口的左下角有关，而文本对象的位置与坐标系的位置相关。

为了说明如何在一图象窗口中设置图形对象的位置，我们将编写一个程序，用它在单个的图象窗口内创建两个交迭的坐标系。

第一个坐标系将用来显示函数 sinx 的图象，并带有相关文本说明。第二个坐标系用来显示函数 cosx 的图象，并在坐标系的左下角有相关的文本说明。

用来创建图象的程序如下所示。注意我们用图函数来创建一个空图象，然后两个 axes 函数在图象窗口中创建两个坐标系。函数 axes 的位置可以用相对于图象窗口的归一化单位指定，所以第一个坐标系起始于(0.05,0.05)，位于图象窗口的左下角，第二坐标系起始于(0.45,0.45)，位于图象的右上角。每个坐标系都有合适的函数进行作图。

第一个坐标系中的文本对象的位置为(-π, 0)，它是曲线上的一点。当我们选择 HorizontalAlignment 的属性值为 right，那么点(-π, 0)则在文本字符串的右边。所以在最终的图象中，文本就会显示在位置点的左边(这对于新程序员来说很容易迷惑)。

在第二个坐标系中的文本对象的位置为(7.5, 0.9)，它位于坐标轴的左下方。这个字符串用 HorizontalAlignment 属性的默认值"left"，点(7.5，0.9)则在文本字符串的右边。所以在最终的图象中，文本就会显示在位置点的右边。

```
%Script file: position_object.m
%
%     Purpose:
%     This program illustrates the positioning of graphics
%     objects. It creates a figure and then places
%     two overlapping sets of axes on the figure. The first
%     set fo axes is placed in the lower left corner of
%     the figure. and contains a plot of sin(x), The second
%     set of axes is placed in the upper right corner of the
%     figure, and contains a plot of cos(x). Then two
%     text strings are added to the axes, illustrating the
%     positioning of text within axes.
%
%     Record of revisions:
%         Date            Programmer              Description fo change
%         =====           ==========              =====================
%     02/26/99            S.J.Chapman             Original code
%
```

```
%     Define varibles:
%     H1              --Handle of sine line
%     H2              --Handle of sosine line
%     Ha1             --Handle of first axes
%     Ha2             --Handle of second axes
%     x               --Independent variable
%     y1              --sin(x)
%     y2              --cos(x)
%     Calculate sin(x) and cos(x)
x = -2*pi:pi/10:2*pi;
y1 = sin(x);
y2 = cos(x);
%     Create a new figure
figure;
%     Create the first set of axes and plot sin(x).
%     Note that the position of the axes is expressed
%     in normalized units.
Ha1 = axes('Position',[.05 .05 .5 .5]);
H1 = plot(x, y1);
set(H1,'LineWidth',2);
title('\bfPlot of sin \itx');
xlabel('\bf\itx');
ylabel('\bfsin \itx');
axis([-8 8 -1 1]);

%     Create the second set of    axes and plot cos(x).
%     Note that the position of the axes is expressed
%     in    normalized units.
Ha2 = axes('Position',[.45 .45 .5 .5]);
H2 = plot(x, y1);
set(H2,'LineWidth',2,'Color','r','LineStyle','--');
title('\bfPlot of cos \itx');
xlabel('\bf\itx');
ylabel('\bfsin \itx');
axis([-8 8 -1 1]);

%     Create a text string attached to the line on the first
%     set of axes.
axes(Ha1);
text(-pi,0.0,'min(x)\rightarrow','HorizontalAlignment','right');

%     Create a text string in the lower left corner
%     of the second set of axes.
axes(Ha2);
text(-7.5,-0.9,'Text string 2');
```

当这个程序执行后，产生的图象如图 9.6 所示。你就应当在你的计算机上重复地执行这人程序，所要画的对象的大小与位置，观察结果。

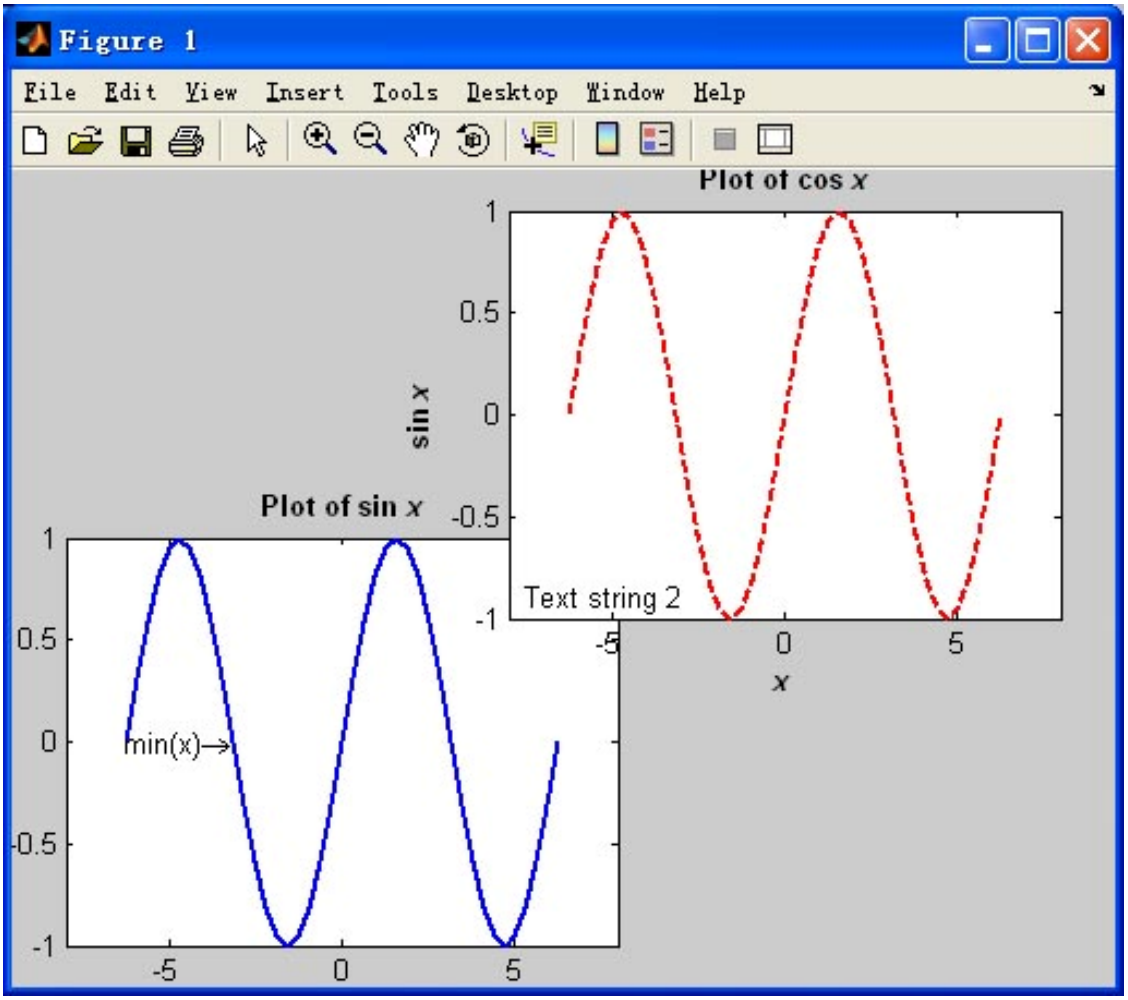


图 9.6 程序 position_object 的结果。

9.9 打印位置

属性"Position"和"Units"用来指定图象在计算机屏幕上的位置。还有其他的五个属性用于指定图象在打印纸上的位置。这些属性被总结在表 9.2 中。

表 9.2 与打印相关的图象属性

参数	描述
PaperUnits	度量纸张的单位 [{inches} | centimeters | normalized | points]
PaperOrientation	[{portrait} | landscape]
PaperPosition	位置向量，形式为[left, bottom, width, height]，单位是 PaperUnits。
PaperSize	包含纸张大小两个元素的向量，例如[8.5 11]
PaperType	设置纸张的类型，注意设置这个属性会自动更新纸张的 PaperSize 属性。[{usletter} | uslegal | a3 | a4letter | a5 | b4| tabloid]

例如，我们用 landscape 模式，用归一化单位在 A4 纸上打印一个图象。我们可以设置下面的属性。

```
set(Hndl, 'PaperType', 'a4letter')
set(Hndl, 'PaperOrientation', 'landscape')
set(\Hndl, 'PaperUnits', 'normalized');
```

9.10 默认和 factory 属性

当一个对象被创建时，**MATLAB** 就会把默认的属性值赋值于每一个对象。如果这些属性值不是你想要的，那么你必须用 set 函数选择你想要的值。如果你想更改你创建的每一个对象的一个属性，这个过程将变得非常麻烦。由于这个原因，**MATLAB** 允许你修改默认值本身，所以当他们被创建时，所有的对象都会继承所有正确的属性值。

当一个图形对象被创建时，**MATLAB** 就会通过检测对象的父对象来寻找每一个属性的默认值。如果父对象设置了默认值，那么这个值就会被应用。如果没有设置默认值，那么 **MATLAB** 就会检测父对象的父对象，看是否有默认值。以此类推，直到根对象。在这个过程中，**MATLAB** 会应用第一次遇到的默认值。

默认属性可以在优先级高的图形对象中的任意一点设置。例如，默认的图的颜色在根对象中设置，而在这之后的所有图象都有一个新的默认颜色。从另一方面说，默认的坐标轴颜色可以在根对象或图象对象设置。如果坐标的默认颜色在根目录中设置，那么它将应用于所有图象的所有新坐标轴，如果默认的坐标轴颜色在图象对象中设置，它将在当前图象窗中的新坐标轴。

默认值的设置要用一个字符串，这个字符串由"Default"，对象类型和属性名组成。所以默认图象颜色可以通过属性"DefaultFigureColor"来设置，默认的坐标轴颜色可以通过属性"DefaultAxesColor"设置。下面是设置默认值的一些例子

```
set(0, 'DefaultFigureColor', 'y')          黄色图象背景
set(0, 'DefaultAxesColor', 'r')            红色坐标系背景——所有图象中的坐标轴
set(gcf, 'DefaultAxesColor', 'r')          红色坐标系背景——当前图象坐标轴
set(gca, 'DefaultLineStyle', ':')          只在当前坐标系中设置默认线型为虚线
```

如果你要对已存在的对象的属性进行修改，那么在用完这个属性之后，最好恢复原来的条件。如果你在一个函数中修改了一个对象的默认属性值，保存它原来的值，并在跳出这个函数之前恢复它们。例如，假充我们用归一化单位创建一系列的图象，我们可以用下面的保存和修复原来的单位。

```
saveunits = get(0, 'DefaultFigureUnits');
set(0, 'DefaultFigureUnits', 'normalized');
...
<MATLAB statements>
...
set(0, 'DefaultFigureUnits', saveunits);
```

如果你想要定制 **MATLAB**，每一次都有不同的默认值，那么每次当 **MATLAB** 启动时你必须对根对象设置默认值。最简单的方法是把默认值存入 startup.m 文件，每次 **MATLAB** 启动时都会自动执行。例如，假设你经常使用 A4 纸，并在图象中经常加入网格线。那么你可以把下面的语句加入到 startup.m 文件中。

```
set(0, 'DefaultFigurePaperType', 'a4letter');
set(0, 'DefaultAxesXGrid', 'on');
set(0, 'DefaultAxesYGrid', 'on');
set(0, 'DefaultAxesZGrid', 'on');
```

有三种特殊值字符串用于句柄图形:"remove"，"factory"和"default"。如果你已经为一个属

性设置了默认值，那么"remove"值将会删除你所设置的默认值。例如，假设你设置默认的图象颜色为黄色。

```
set(0, 'DefaultFigureColor', 'y');
```

调用下面的函数将会取消当前的默认值并恢复先前的默认值。

```
set(0, 'DefaultFigureColor', 'remove');
```

字符串"factory"允许临时跳过当前的默认值，并使用原来的 **MATLAB** 的默认值。例如，尽管当前的默认颜色为黄色，下面的语句将会用 factory 创建下面的图象。

```
set(0, 'DefaultFigureColor', 'y');
figure('Color', 'factory');
```

第三个特殊的属性值字符串是 default，这个属性值迫使 **MATLAB** 搜索对象层次结构，直到查到所需属性的一个默认值。如果找到，它就使用该默认值。如果查到根对象，没有找到用户定义的默认值，**MATLAB** 就使用 factory 默认值。它的应用说明如下

```
%    Set default values
set(0, 'DefaultLineColor', 'k');        %    root default = black
set(gcf, 'DefaultLineColor', 'g');      %    figure default = green
%    Create a line on the current axes. This line is green.
Hndl = plot(randn(1, 10));
set(Hndl, 'Color', 'default');
pause(2);
%    Now clear the figure default and set the line color to the new
%    default. The line is now black.
set(gcf, 'DefaultLineColor', 'remove');
set(Hndl, 'Color', 'default');
```

9.11 图形对象属性

由于有成百上千的图形对象属性，我们在这里不一一讨论了。我们可以通过 **MATLAB** 帮助台得到所有属性。

9.12 总结

好的编程方法总结

1. 如果你打算修改你创建的对象的属性，那么请保存对象的句柄，为以后调用函数 get 和 set 做准备。
2. 如果有可能的话，限定函数 findobj 的搜索范围将能加快函数的运行速度。
3. 如果你想把对象放置在窗口的特定位置，最好的方法是用归一化坐标，因为不用考虑显示器的大小。

MATLAB 总结

- gcf 返回当前图象的句柄
- gca 返回当前图象中当前坐标系的句柄
- gco 返回当前对象的句柄
- findobj 寻找指定属性值的图形对象

9.13 练习

1.

句柄语句什么意思?**MATLAB** 图形对象的优先级是怎样的?

2.

用 **MATLAB** 帮助工作台了解图对象的 Name 和 Number Title 属性。画出函数 $y(x) = e^x(-2 \le x \le 2)$ 的图象。改变上面提到的图象属性。禁止更改图象数，禁止增加"plot window"的标题。

3.

编写一个程序，修改图象的默认颜色为桔黄色，默认线宽为 3.0point。那么用下面的等式创建图象

$$x(t) = 10\cos t$$
$$y(t) = 6\sin t$$

其中 t 的取值从 $t = 0$ 到 $t = 2\pi$。

4.

用 **MATLAB** 帮助工作台了解坐标系对象的 CurrentPoint 属性。用这个属性编写一个程序。创建一个坐标系对象，并在坐标内画出鼠标单击过的点，然后用直线相连。用函数 waitforbuttonpress 等待鼠标单击，并在每一次单击后更新图象。当键盘事件发生后，中止程序。

5.

用 **MATLAB** 程序画出下面函数的图象

$$x(t) = \cos\ \frac{t}{\pi}$$
$$x(t) = 2\ \sin\ \frac{t}{2\pi}$$

其中 $-2 \leqslant t \leqslant 2$。这个程序应当等待鼠标单击，如果鼠标单击在两条直线上的一条，程序就会随机地改变这条直线的颜色。用函数 waitforbuttonpress 等待鼠标单击，并在每一次单击后更新图象。用函数 gco 判断单击的对象，并用 Type 属性判断单击的对象是否为直线。

6.

创建一个 **MATLAB** 图象，并把一些自定义数据变量存储在图象中。然后用 file/save 把图象存储到图象文件中(*.fig)。重启 **MATLAB** 程序，加载图象文件，并用函数 getappdata 查看自定义数据。这些数据被存储了吗?