

## 第13章 图形和声音

MATLAB拥有大量简单、灵活、易用的二维和三维图形命令，并且用户可以在 MATLAB 程序中加入声音效果。许多图形命令都在 MATLAB 所带的演示程序中给出。还未用过 demo 命令的用户不妨马上试一试。

图形命令分两章来介绍。本章只介绍基本的高级命令，在第 14 章中将详细介绍高级图形，并着重介绍具体的低级控制。

### 13.1 二维图形

将数对排序的一种方法是使用 plot 命令。该命令可以带有不同数目的参数。最简单的形式就是将数据传递给 plot，但是线条的类型和颜色可以通过使用字符串来指定，这里用 str 表示。表 13-1 列出了在这个字符串中允许使用的线条类型和颜色。线条的缺省类型是实线型。注意：下面的命令列表中 A 表示一个  $m \times n$  的矩阵。

#### 命令集123 绘图命令

plot(x,y)	对向量 x 绘制向量 y。以 x 为横坐标，y 为纵坐标，按照坐标 $(x_j, y_j)$ 的有序排列绘制曲线。
plot(y)	以 j 为横坐标， $y_j$ 为纵坐标，绘制 $(j, y_j)$ 的有序集合的图形。
plot(z)	以横轴为实轴，纵轴为虚轴，绘制 $(\text{real}(z_k), \text{imag}(z_k))$ 的有序集合的图形。这样，复数 $z_k$ 就在复平面上。
plot(A)	绘制矩阵 A 的列对它下标的图形。对于 $m \times n$ 的矩阵 A，有 n 个含有 m 个元素的数对，或是 n 条有 m 个点曲线，且这 n 条曲线均采用颜色监视器上不同的颜色绘制而成。
plot(x,A)	绘制矩阵 A 对向量 x 的图形。对 $m \times n$ 的矩阵 A 和长度为 m 的向量 x，绘制矩阵 A 的列对向量 x 的图形。如果 x 的长度为 n，则绘制矩阵 A 的行对向量 x 的图形。向量 x 可以是行向量也可以是列向量。
plot(A,x)	对矩阵 A 绘制向量 x 的图形。对于一个 $m \times n$ 的矩阵 A 和一个长度为 m 的向量 x，对矩阵 A 的列绘制向量 x 的图形。如果 x 的长度为 n，则对矩阵 A 的行绘制向量 x 的图形。向量 x 可以是行向量也可以是列向量。
plot(A,B)	对矩阵 A 的行绘制矩阵 B 的列的图形。如果 A 和 B 都是 $m \times n$ 的矩阵，将绘制 n 条由 m 个有序对连成的曲线。
plot(...,str)	使用字符串 str 指定的颜色和线型进行绘图。表 13-1 列出了 str 可以取的值。
plot(x1,y1,str1, ...)	用字符串 str1 指定的颜色和线型对 y1 绘制 x1 的图形，用字符串

`x2,y2,str2,...)` `str2`指定的颜色和线型对 `y2`绘制`x2`的图形……。每组参数值可以采用上述除复数值以外的任何一种形式。`str1, str2...`可以省略,此时, MATLAB自动为每条曲线选择颜色和线型。

`[l,f,p,error]=colstyle(str)` 返回`str`中不同部分的值。其中`l`代表线型, `f`代表颜色, `p`代表点的类型, `error`用来保存系统错误信息。

表13-1 点类型、线类型与颜色

点 类 型		线 类 型	
.	点	-	实线
*	星号	--	虚线
square	正方形	-	点划线
diamond	菱形	:	点线
pentagram	五角星形	none	无线
hexagram	六角星形		
		颜色	
none	无点	g	绿色
o	o	m	品红色
+	+	b	蓝色
x	x	c	灰色
<	顶点指向左边的三角	w	白色
>	顶点指向右边的三角	r	红色
^	正三角	k	黑色
v	倒三角	y	黄色

通过将字符串`str`作为一个参数传递给`plot`,可以指定图形的颜色和线型。表 13-1列出了允许的值和它们代表的意义。这些参数可以组合起来使用,例如,‘`y+`’表示一个黄色的加号,而‘`b - -`’表示一个蓝色的虚线。如果将要画的是几组数据,但是没有指定线型,系统将会自动按照表 13-1赋予它们从黄到黑各种不同的颜色线型。

符号的大小、线条的粗细等也同样可以更改;可参见例 13.1(g)或14.2节。

### 例13.1

(a) 用下列数据来绘制图形:

```
x = [-4 -2 0 1 3 5];
y = [16 4 0 1 9 25];
```

命令`plot(x,y)`产生的结果如图 13-1所示。

(b) 在MATLAB中,能很容易地画出点:

```
x = -pi:0.05:pi;
plot(x,sin(x)*cos(x),' o');
```

产生的结果如图 13-2所示。

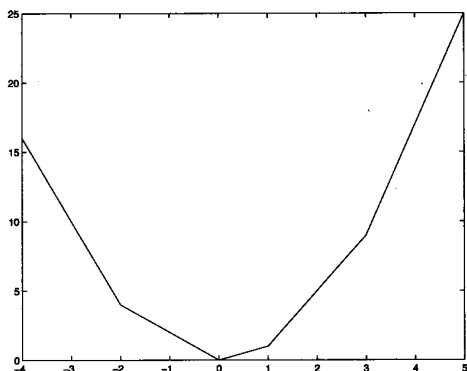
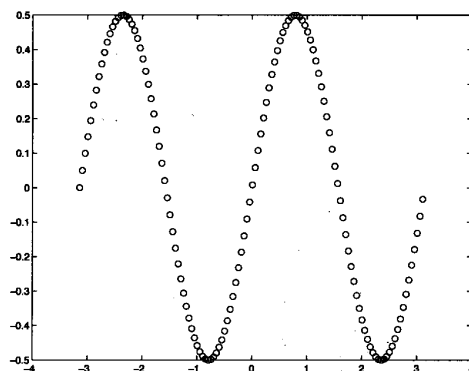


图13-1 向量y对向量x绘图

图13-2 用小圆圈绘制函数  $\sin x \times \cos x$ 

(c) 在同一幅图中可以同时绘制多个函数：

```
x = 0:0.1:2;
A = [sin(pi*x); 0.5+0.5*x];
plot(x, A);
```

产生的结果如图 13-3 所示。

(d) 可以通过交换参数位置来交换坐标轴。对图 13-3 和图 13-4 进行比较：

```
x = 0:0.1:2;
A = [sin(pi*x); 0.5+0.5*x];
plot(A, x);
```

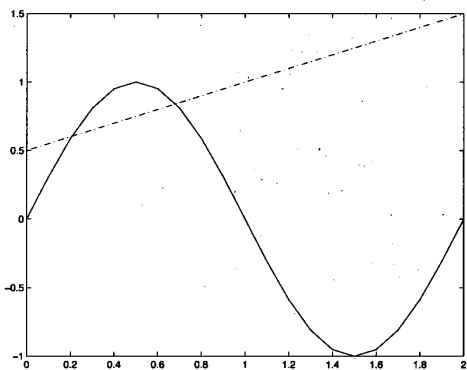


图13-3 矩阵A对向量x绘图

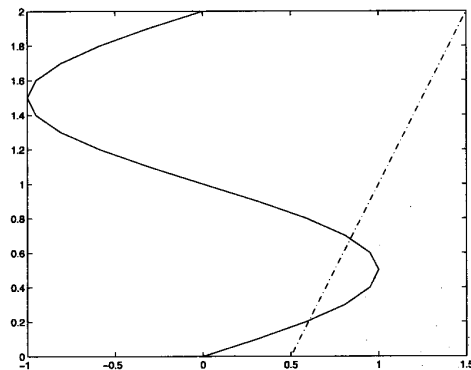
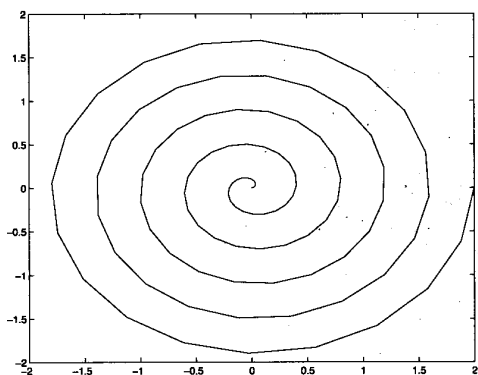


图13-4 向量x对矩阵A绘图

(e) plot命令对复数矩阵同样适用。

```
clear i; % 保证i是复数
r = linspace(0,2); % 创建向量r
theta = linspace(0,2*pi); % 创建角向量
[x,y] = pol2cart(theta,r); % 将弧度坐标
z = x+*y; % 转化成复数向量
plot(z) % 对z绘图
```

结果如图 13-5 所示。注意：还可以用命令 polar、quiver、feather、compass 和 rose 来对复数绘图；参见 13.2 节。

图13-5 复向量 $z$ 代表一个螺旋线。

(f) 下列命令形成文件 expotest.m :

```
% 程序执行前应先设定下列
% 参数 : n, a, b
% 点数 : n.
% 区间 : [a, b]

x = [];
e1 = []; e2 = []; e3 = []; e4 = []; % 清除 e1-e4

for i = 1:n
    xx = a + (b-a)*(i-1)/(n-1);
    x(i) = xx;
    e1(i) = exp(-(xx^2));
    e2(i) = xx^2*exp(-(xx^2));
    e3(i) = xx*exp(-(xx^2));
    e4(i) = exp(-xx);
end
```

尽管下列代码将会产生同样的结果，但它的效率更高，易读且不易产生错误。

```
x = linspace(a,b,n);

e1 = exp(-x.^2);
e2 = (x.^2).*exp(-x.^2);
e3 = x.*exp(-x.^2);
e4 = exp(-x);

下列语句：

n = 50;
a = 0;
b = 3;
expotest
plot(x,e1,x,e2,x,e3,x,e4);
```

将产生图13-6(左)所示的图形。而

```
plot(x,e1,'+',x,e2,'*',x,e3,'o',x,e4,'x');
```

将产生图13-6(右)所示的图形。

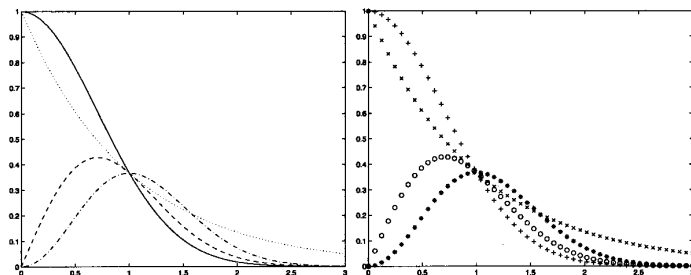


图13-6 用标准符号(左)和用户指定的符号(右)对指数函数绘图

(g) 假设已定义了与(f)中相同的变量。现在要来改变线条的粗细：

```
hold on;
plot(x,e1,'LineWidth',1);
plot(x,e2,'LineWidth',2);
plot(x,e3,'LineWidth',3);
plot(x,e4,'LineWidth',4);
hold off;
```

命令hold on用来保持当前图形，使得可以在同一幅图中绘制多个图形，而hold off用来关闭图形的；可参见命令集130。其中，曲线e1线条最细，e4线条最粗，如图13-7所示。

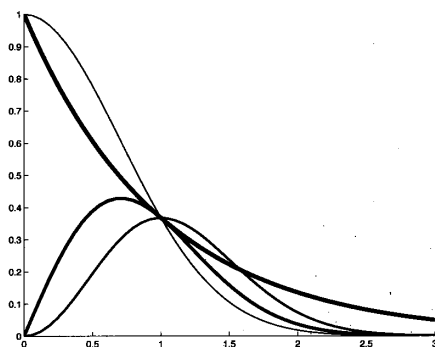


图13-7 用不同的粗细线条绘制的指数函数图形

在MATLAB中可以使用errorbar命令绘制数据的误差条形图。该命令的用法与plot命令完全类似，只是要同时赋予每个点一个误差限。

#### 命令集124 误差条形图

<pre>errorbar(x,y,e,str)</pre>	<p>绘制向量y对x的误差条形图。误差条对称地分布在<math>y_i</math>的上方和下方，长度为<math>e_i</math>。字符串str决定其颜色和线型，参见表13-1。参考命令集123中的命令plot。</p>
<pre>errorbar(x,y,l,u, str)</pre>	<p>绘制向量y对x的误差条形图，误差条分布在<math>y_i</math>上方的长度为<math>u_i</math>，下方的长度为<math>l_i</math>。字符串str选项决定其颜色和风格。</p>

## 例13.2

假定误差限为15%，下面的程序将产生一系列数字，并生成该列数据的误差条形图。

```
x = linspace(0,10,50);      % 创建一系列值
y = exp(sin(x));             % 创建数据

delta = 0.15*y;              % 计算15%的误差限
errorbar(x,y,delta);         % 绘出误差条形图
```

运行后可给出图13-8所示的图形。

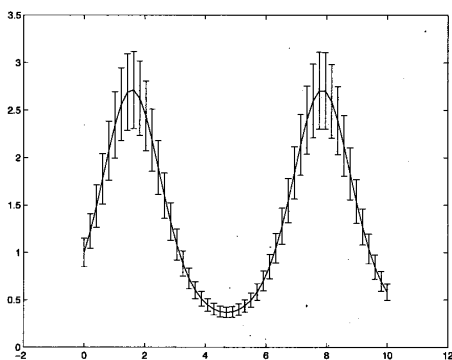


图13-8 函数  $e^{\sin x}$  的误差条形图

使用comet命令可以绘制彗星图形。彗星图形是一个动态的绘图过程。comet3命令可用于绘制三维彗星图形；参见13.5节。

## 命令集125 彗星图形

comet(x,y)	绘制向量y对向量x的彗星轨线。如果只给出一个向量，则用该向量对其下标值绘图
comet(x,y,l)	绘制慧长为 $l * \text{length}(y)$ 的彗星轨线，l的缺省值为0.1
comet	绘出一个彗星图形的例子

输入doc comet可得更多的信息。

MATLAB中有些函数可以用来改变图形的外观。

## 命令集126 其他绘图命令

area(x,y)	和plot命令一样，但是将所得的曲线下方即曲线与横轴之间的区域填充颜色。
area(x,A)	矩阵A的第一行对向量x绘图，然后依次是下一行与前面所有行值的和对向量x绘图。每个区域有各自的颜色。
area(y)	等价于 $x=1:\text{size}(y,1)$
area(...,'Property',	为area创建的带下划线的补片对象设定属性；参见第4章。

<code>Value,...)</code>	
<code>barh(x,A,format)</code>	对 $m \times n$ 矩阵绘制含有 $m$ 组、每组 $n$ 个柱形条的直方图。字符串 <code>format</code> 可以是颜色类型或字符串 'stacked'。'stacked' 表示将 $n$ 个柱形条的值叠加在另一条上。
<code>barh(A)</code>	和 <code>barh</code> 命令一样，但是令 $x=1:m$ 。
<code>ezplot(f,xmin,xmax)</code>	绘制函数 $f$ 在区间 $[xmin, xmax]$ 上的图形。如果省略 $xmin$ 和 $xmax$ 参数，区间将大概取在 $-2 \sim 2$ 之间。由于 <code>ezplot</code> 命令使用算法来判断该函数变化显著的区间，因此区间的选取是不固定的。
<code>pareto(y,x)</code>	按降序绘制 $y$ 中各分量的柱形图。可以给定向量 $x$ 并且应该包含 $x$ 轴的下标。如不给定，则将使用向量 $y$ 中各元素的下标，同时， <code>pareto</code> 命令还能对由各元素累积和形成的向量绘制曲线。
<code>pie(x,explode)</code>	绘制向量 $x$ 的饼图。如果 $\text{sum}(x) \leq 1$ ，则将给出一个不完全的饼图。向量 <code>explode</code> 与向量 $x$ 的维数相同，并且 <code>explode</code> 中不为零的元素所对应的相应部分将从饼图中独立出来。
<code>scatter(x,y,size,color)</code>	以具有相同维数的向量 $x$ 、 $y$ 所确定的点为圆心， <code>size</code> (以点为单位) 为半径绘制圆。圆的颜色由 <code>color</code> 确定，可以是向量、矩阵或颜色字符串。参见 <code>helpdesk</code> 可得更多信息。
<code>plotmatrix(X,Y)</code>	绘制 $X$ 的列对 $Y$ 的列的分散矩阵图形。
<code>plotmatrix(X)</code>	和 <code>plotmatrix(X,X)</code> 一样，但是在对角线上画出柱状图。
<code>[H, AX, BigAx, p] = plotmatrix(...)</code>	返回整个图形的句柄 $H$ 矩阵。矩阵 $AX$ 包含单个子坐标系的句柄， $BigAx$ 包含的是大坐标系的句柄。柱状图的句柄保存在 $P$ 中。留下 $BigAx$ 作为当前句柄如被 <code>axes</code> 使用。
<code>plotyy(x1,y1,x2,y2,fun1,fun2)</code>	$y1$ 按左侧轴的刻度对 $x1$ 绘图， $y2$ 按右侧轴的刻度对 $x2$ 绘图。若缺省参数 <code>fun1</code> 和 <code>fun2</code> ，则结果与使用 <code>plot</code> 命令相同。参数 <code>fun1</code> 和参数 <code>fun2</code> 可以是类似 'semilogx'、'loglog' 等的字符串。不同的函数绘图可参见第 3.2 节。

### 例13.3

用如下的方法可以在同一副图中绘制不同尺寸的图形。

```
% plotyy 演示
% 定义数据
x = 0:0.25:4;
y = exp(x);

clf reset;           % Plotyy对坐标轴上定义的数据很敏感
```

```

plotyy(x,y,x,y,'plot','semilogy')
hold on;
title('Plotyy')
ylabel('Linear scale')

```

产生结果如图 13-9 所示，不幸的是 plotyy 与其他命令一样会产生一些问题。例如：legend 只适用于一个坐标轴。通过 <ftp://ftp.mathworks.com/pub/tech-support/library/graphics/plotyy.m> 可以获得 plotyy.m 文件。该文件给出了为所有坐标轴定义标识符的可能情况。

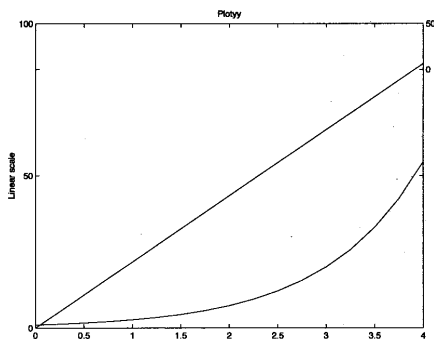


图13-9 借助plotyy 命令在同一幅图中用plot 和semilogy 绘制数据图形

#### 例13.4

使用area命令，MATLAB可以绘制点的累积图形。

% 命令的演示

```
x = 0:10;
```

```
A = [ sin(x);x;(x/3).^2 ]';
```

```
clf;
```

```
areahandle = area(x,A)
```

```
hold on
```

```
title('Area plot')
```

```
legend(areahandle,'sin(x)', 'sin(x)+x', 'sin(x)+x+(x/3)^2',2)
```

结果如图 13-10 所示。

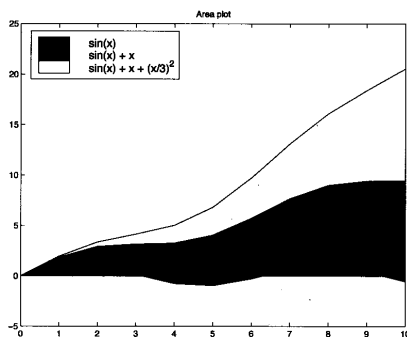


图13-10 使用area 命令绘制三条曲线

命令 fplot 可以绘制出标准的 MATLAB 和用户自定义的函数图形。区间范围和函数名字



可以作为参数给出。

### 命令集127 函数图形

`fplot(fcn,lim,str)` 绘制由字符串 `fcn` 指定的函数图形。这可以是标准函数，也可以是用户在 M 文件 `fcn.m` 中自定义的函数，但不允许是内联函数。向量 `lim=[xmin xmax]` 给出绘图区间范围。该向量也可以包含四个元素，后两个参数用来表示 `y` 轴的区间，即 `lim=[ xmin xmax ymin ymax]`。如果字符串 `str` 传递给 `fplot`，则可以根据表 13-1 来改变图形的线型和颜色。

`fplot(fcn,lim,str,tol)` 同上所述进行绘图，但是带有一个小于 `tol` 的相对误差

注意，使用 `fplot` 绘制所谓的内联函数是不可能的。

#### 例13.5

用下面的语句来绘制  $\sin x^2$  图形

```
fplot('sin(x.^2)',[0,10]);
```

将得到如图 13-11 所示的图形。

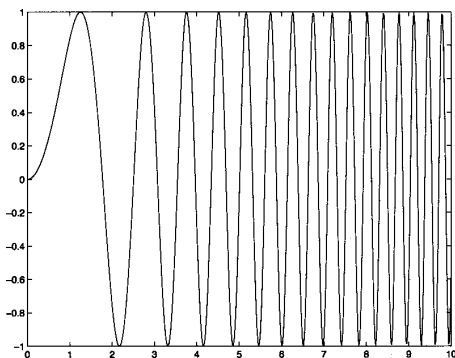


图13-11 使用 `fplot` 命令绘制函数  $\sin x^2$  的图形

## 13.2 在其他坐标系和复平面上绘图

`plot` 命令使用的是笛卡儿坐标系，其实，使用其他的坐标系也是可以的。字符串参数 `str` 可以传递给下列所有的命令，以确定绘图的颜色和线型（参见表 13-1）。

### 命令集128 在其他坐标系中绘图

`polar(theta,r)` 在极坐标中绘图。向量 `theta` 的元素代表弧度参数，向量 `r` 代表从极点开始的长度。

`semilogx(x,y)` 在半对数坐标系中绘图，`x` 轴用以 10 为底的对数刻度标定。这类

`semilogy(x,y)` 似于`plot(log10(x),y)`,但是对于`log10(0)`不能给出警告信息。在半对数坐标系中绘图, y轴用以10为底的对数刻度标定。这类似于`plot(x,log10(y))`,但是对于`log10(0)`不能给出警告信息。  
`loglog(x,y)` 在对数坐标系中绘图。两个坐标轴均用以10为底的对数刻度标定。这类似于`plot(log10(x),log10(y))`,但是对于`log10(0)`不能给出警告信息。

参见第2.4节中关于更改坐标系的命令。

### 例13.6

(a) 在半对数刻度坐标系中绘图与在通常的笛卡儿坐标系中用 `plot` 命令绘图一样容易。

```

x=linspace(0,7);           % 创建x值
y=exp(x);                  % 创建y值
subplot(2,1,1);plot(x,y);  % 绘制通常图形
subplot(2,1,2);semilogy(x,y); % 绘制半对数刻度曲线
  
```

通过使用 `subplot` 命令可以在一个图形窗口中绘制多个小图形; 见第 13.3 节。执行上述命令, 可以得到图 13-12 所示的图形。

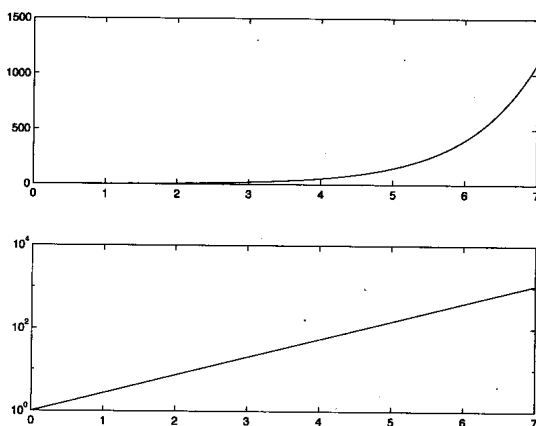


图13-12 在普通坐标系(上图)和y轴对数刻度坐标系(下图)中绘制指数函数

(b) 为了在极坐标系中绘制曲线, 可以使用 `polar` 命令。函数

$$r = e^{\cos t} - 2 \cos 4t + \left( \sin \frac{t}{12} \right)^5$$

描绘的是一条复平面上的曲线。这里介绍绘制这条曲线的两种方法。

% 定义函数

```

t=linspace(0,22*pi,1100);
r=exp(cos(t))-2*cos(4*t)+sin(t./12).^5;
subplot(2,1,1)
p=polar(t,r);           % 在极坐标系中绘图
subplot(2,1,2)
[x,y]=pol2cart(t,r);    % 找到笛卡儿坐标
  
```

```
plot(x,y) % 在x-y平面上绘图
```

可以得到两个不同的图形如图 13-13所示(上图)和(下图)。

```
[x,y] = pol2cart(t,r); % Find the cartesian coordinates.
```

```
plot(x,y); % Plot in x-y plane.
```

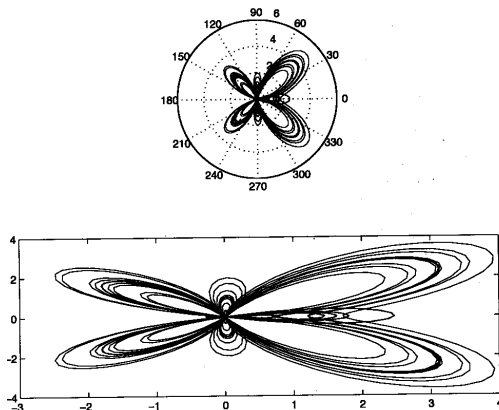


图13-13 在极坐标系中的图形(上图)和笛卡儿坐标系中的图形(下图)

可以使用quiver、feather、compass以及rose命令来绘制复数图形。这些命令同样可以用于实数矩阵；参见例 13.16(b)。

#### 命令集129 复平面图形

quiver(X,Y)	对每对坐标 $(X_{ij}, Y_{ij})$ 绘制一个箭头。命令 $(\text{real}(Z), \text{imag}(Z))$ 可以看成绘制矩阵 $Z$ 中复数元素方向与大小的图形。
quiver(x,y,dx,dy)	在坐标 $(x_i, y_i)$ 处画一个箭头，由 $(dx_i, dy_i)$ 指定方向和大小。
quiver(X,Y,Dx,Dy)	在坐标 $(X_{ij}, Y_{ij})$ 处画一个箭头，由 $(DX_{ij}, DY_{ij})$ 指定方向和大小。
quiver(X,Y,...,s)	同上所述绘制箭头，但是用 $s$ 进行标记。如果省略缺省值为1。
quiver(X,Y,...,str)	使用 $str$ 指定的线型绘制箭头；参见表 13-1。
feather(Z)	把复数矩阵 $Z$ 中元素的相角和幅值显示成沿横轴等间隔辐射的箭头。
feather(X,Y)	等价于 $\text{feather}(X+Y*i)$ 。
feather(Z,str)	使用 $str$ 确定的线型绘制箭头；参见表 13-1。
compass(Z)	把复数矩阵 $Z$ 中元素的相角和幅值显示成从原点辐射的箭头。
compass(X,Y)	等价于 $\text{compass}(X+Y*i)$ 。
compass(Z,str)	使用 $str$ 确定的线型绘制箭头；参见表 13-1。
rose(v)	绘制相角直方图，也就是向量 $v$ 中相角频率的角度直方图，间隔数为36。
rose(v,n)	同上，但是由 $n$ 指定间距数。
rose(v,x)	绘制相角直方图，使用向量 $x$ 确定的间隔。

调色板部分的图P-5用四条极值曲线展示了对于函数  $z=f(x, y)$  的箭图。

### 例13.7

定义Z：

$$Z = \begin{pmatrix} 1+i & 2-i & 3-5i \\ -4+3i & 5-3i & i \\ -1-i & 3+3i & -1 \end{pmatrix}$$

下列命令产生图13-14的结果。

```
clear i; % 确保i为复数。
```

```
Z = [1+i 2-i 3-5i; -4+3i 5-3i i; -1-i 3+3i -1];
```

```
clf;
```

```
subplot(2,2,1); quiver(real(Z),imag(Z)); title('quiver');
```

```
subplot(2,2,2); feather(Z); title('feather');
```

```
subplot(2,2,3); compass(Z); title('compass');
```

```
subplot(2,2,4); rose(angle(Z(:))); title('rose');
```

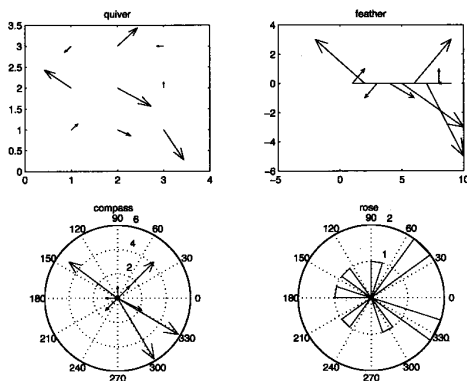


图13-14 图形表示的复数

使用subplot命令可以在一个图形窗口中绘制多个图形，参见 13-3节。这些命令给出如图13-14所示的图形。

## 13.3 图形控制

MATLAB中的图形是面向对象的。本章中列出的命令通常用来设定和创建对象，这些对象用来绘制和改变图形。本节将讨论几个用来改变这些对象的高级命令，其中，许多改变是通过单一的命令来实现的。当然，还可以对单一图形对象的属性进行操作，这些将在第14章中介绍高级绘图命令时进行详细介绍。

MATLAB中有两类窗口：命令窗口和图形窗口。给出 MATLAB命令要使用命令窗口，而图形窗口是用来展示图形的。图形窗口可以用本节所介绍的命令进行控制，也可以用另一种更直接的方式进行控制；参见第14章。

能否在屏幕上同时显示图形窗口和命令窗口由硬件来决定。但是有些命令可以用来在各

个窗口之间进行切换、清除窗口内容或保持当前图形。

### 命令集130 窗口命令

<code>figure(gcf)</code>	显示当前图形窗口。 <code>figure</code> 命令还可以用来在两个图形窗口之间进行切换和创建新的图形窗口；参见14.2节。
<code>shg</code>	显示当前图形窗口，等价于 <code>figure(gcf)</code> 。
<code>clf</code>	清除当前图形窗口。警告：如果设置 <code>hold on</code> 状态，窗口内容也将被清除。
<code>clg</code>	早期版本中等价于 <code>clf</code> 命令。在MATLAB以后的版本中可能会被淘汰。
<code>clc</code>	清除命令窗口。
<code>home</code>	移动光标到命令窗口的左上角。
<code>hold on</code>	保持当前图形。允许在当前图形状态下，使用同样的缩放比例加入另一个图形。
<code>hold off</code>	释放图形窗口，这样下一个图形将称为当前图形。这是缺省状态。
<code>hold</code>	在 <code>hold on</code> 和 <code>hold off</code> 之间进行切换。
<code>ishold</code>	如果当前图形处于 <code>hold on</code> 状态，则返回1；否则，返回0。

命令`subplot`用于在同一个图形窗口中绘制几个图形。`subplot`本身并不绘制任何图形，但是，它决定了如何分割图形窗口以及下一幅图将被画在哪个子窗口中。

### 命令集131 子图

<code>subplot(m,n,p)</code>	将图形窗口分割成 $m$ 行 $n$ 列，并设置 $p$ 所指定的子窗口为当前窗口。子窗口按行由左至右，由上至下进行编号。这一命令在MATLAB的当前版本中也被写作 <code>subplot(mnp)</code> 。
<code>subplot</code>	设置图形窗口为缺省模式，即单窗口模式。等价于 <code>subplot(1,1,1)</code> 。

#### 例13.8

(a) 在命令窗口的左上角显示一个由变化的随机数组成的矩阵。

```
clc                % 清除命令窗口的内容
for I=1:10
    home           % 移动光标至左上角
    A=rand(5)      % 创建并输出矩阵
    pause(1);      % 延迟一秒钟
end
```

(b) 下列的MATLAB命令在左上角的子窗口中绘制出函数 $f(x) = -x\sin x$ 的图形，在右上角的子窗口中绘制其导函数 $f'(x) = -x\cos x - \sin x$ 的图形，在左下角的子窗口中绘制近似导函数的图形，在右下角的子窗口中绘制精确导函数和近似导函数的相对误差图形。

```
% 创建x的值。生成f的值y11，导函数y12，近似导函数y21和它的误差y22。
n = 1000;
x = linspace(-10,10,n);
```

```

y11 = (-x).*sin(x);
y12 = (-x).*cos(x) - sin(x);
y21 = diff(y11)./(x(2)-x(1));
y22 = (y21 - y12(1:n-1));
% 在左上角绘图
subplot(2,2,1); plot(x,y11);
title('The function')
% 在右上角绘图
subplot(2,2,2); plot(x,y12);
title('The derivative')
% 在左下角绘图
subplot(2,2,3); plot(x(1:n-1),y21);
title('The approximated derivative')
% 在右下角绘图
subplot(2,2,4); plot(x(1:n-1),y22);
title('The difference')

```

上述命令给出如图 13-15 的图形。

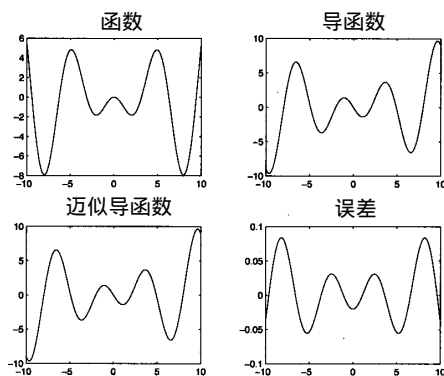


图13-15 函数 $x\sin x$ 及其导函数、近似导函数和误差图形

命令subplot也可用于三维图形，并且子图在同一窗口中可以有不同的大小，如下面的例子所示。

### 例13.9

本例包含的M文件用来计算Mandelbrot不规则碎片，并将其用三种方式显示出来。该程序在由用户定义在复平面的网格中画点，并且根据下面的算法迭代所定义的网格中的每个数字  $c$ 。

$$z_0 = 0$$

$$z_{i+1} = z_i^2 + c$$

如果  $z_i$  是发散的，则当前  $c$  将不作为Mandelbrot集合的一部分。复平面上每一点  $c$  的迭代次数将以与网格中相同的大小保存在矩阵Mandelbrot中。算法将只迭代100次，因此分散点在Mandelbrot矩阵中的值为100。

% Mandelbro程序MandelbrotProg.m。

```

clear;
renum=input('renum:');      % 读实数点的个数
imnum=input('imnum:');      % 读虚数点的个数
remin=-2;remax=1;          % 定义要计算的数字
immin=-1.5;imax=1.5;
% 大小合适的向量

reval1 = linspace(remin,remax,renum);
imval1 = linspace(immin,imax,imnum);

% 虚平面上的网格
[Reval, Imval] = meshgrid(reval1,imval1);

Imvalreal = Imval; Imval = Imval*i;
Cgrid = Reval + Imval;

for reind = 1:renum
    disp(['reind = ',int2str(reind)]);      % 写循环状态

    % 迭代循环          z(i+1) = (z(i))^2 + c.
    for imind = 1:imnum
        c = Cgrid(reind,imind);
        numc = 0;                      % 初始化
        zold = 0.0 + i*0.0;
        z = zold^2 + c;                  % z(0) = c.

        while (abs(z) <= 2) & ...
            (numc < 100)
            numc = numc + 1;
            zold = z;
            z = zold^2 + c;              % 新z!
        end

        % 在矩阵mandelbrot中mandelbrot(n,m)位置上写入点
        % Cgrid(n, m)的迭代次数
        Mandelbrot(reind,imind) = numc;
    end
end

% 用三种方式显示mandelbrot
% 矩阵图形：

clf;                      % 清除图形

subplot(2,2,1);           % 左上角
mesh(reval1,imval1,Mandelbrot); % 绘三维网格表面图
axis([-2 1 -1.5 1.5 0 100]) % 改变坐标轴区间

subplot(2,2,2);           % 右上角
contour(reval1,imval1,Mandelbrot,100); % 等高线图

```

```

grid;                                % 添加网格

subplot(2,1,2);                      % 降低图的位置(仅一个)
surf(Reval,Imvalreal,Mandelbrot);    % Mandelbrot的表面图

view(2);                             % 俯视图

% 每个元素只有一种颜色, 并且使用
% 反白的黑色条
shading flat;
colormap(flipud(jet));

% 显示颜色条。改变坐标轴
colorbar; axis([-2 1 -1.5 1.5]);

```

该程序的结果如图 13-16 所示。

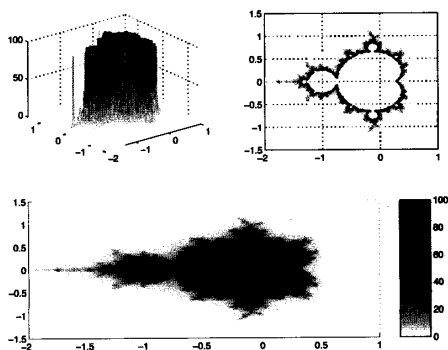


图13-16 以三种方式显示的Mandelbrot fractal:左上角：三维表面图；  
右上角：等高线图；下方：俯视图

图形几乎可以被放置在图形窗口的任何位置, 还有许多的图形控制; 参见第 14 章。

每幅图的坐标轴通常都自动标上适合窗口中所有点的缩放比例。因此图的各个角就被定义成如下的形式：

$(\min(x), \min(y)), (\max(x), \min(y)), (\min(x), \max(y)), (\max(x), \max(y))$ 。

有时, 一些点由于缩放比例原因会与坐标轴重合, 因此看到这些点比较困难。幸运的是, MATLAB 中的命令 `axis` 可以用来改变缩放比例。

同样可以使用鼠标或 `zoom` 命令来改变缩放比例。

#### 命令集132 坐标轴, 刻度和窗体缩放

`axis` 用行向量中给出的值, 设置坐标轴的最大和最小值。对于二维图形, 该向量中含有元素:  $[x_{\min}, x_{\max}, y_{\min}, y_{\max}]$ 。对于三维图形(见13.5节), 是  $[x_{\min}, x_{\max}, y_{\min}, y_{\max}, z_{\min}, z_{\max}]$ 。

`axis(str)` 字符串str的不同将给出不同的结果：

‘manual’ 固定坐标轴刻度。如果当前图形窗口为hold打开状态, 则后面的图形将采用同样的刻度。



- 'auto' 把坐标轴刻度重新设置为缺省状态值。
- 'equal' 设置x轴和y轴为同样的刻度增量。
- 'tight' 采用与x轴方向和y轴方向相同比例的坐标轴，从而只绘制包含数据的部分坐标。
- 'fill' 设定坐标轴的边界，以使其能够匹配数据集的范围。
- 'ij' 翻转y轴，使得正数在下，负数在上。
- 'xy' 复位y轴，使正数在上。
- 'image' 重新设置图形窗口的大小，使得各像素有与宽度相同的高度以适应于计算机。
- 'square' 重新定义图形窗口的大小，使窗口为正方形。
- 'vis3d' 锁定坐标轴之间的关系。比如用在旋转D对象时。
- 'normal' 复位图形窗口至标准大小。
- 'off' 不显示坐标轴或刻度。
- 'on' 显示坐标轴和刻度。

这一命令也可以写成 `axis norma` 等的形式。

`axis(v)`

根据向量v设置坐标轴刻度，使  $x_{\min}=v_1$ ,  $x_{\max}=v_2$ ,  $y_{\min}=v_3$ ,  $y_{\max}=v_4$ 。对于三维图形还会设置  $z_{\min}=v_5$ ,  $z_{\max}=v_6$ 。对于本章前面所讨论的对数图形，使用原数值，而不是对数值。通常还可以使用 `axlimdlg` 命令设置缩放比例；参见命令集169。

`axis(axis)`

固定坐标轴刻度。使得 MATLAB 在向原图上增加图形时不能改变刻度；参见 13.3 节命令 `hold`。

`xlim([xmin xmax])` 设置  $x_{\min}=xmin$ ,  $x_{\max}=xmax$ 。

`xlim` 返回  $[x_{\min}, x_{\max}]$ 。

`ylim([ymin ymax])` 设置  $y_{\min}=ymin$ ,  $y_{\max}=ymax$ 。

`ylim` 返回  $[y_{\min}, y_{\max}]$ 。

`zlim([zmin zmax])` 设置  $z_{\min}=zmin$ ,  $z_{\max}=zmax$ 。

`zlim` 返回  $[z_{\min}, z_{\max}]$ 。

`box`

控制是否将图形用坐标轴从各个边包围。命令 `box on` 打开该功能，而 `box off` 关闭该功能。只改变 `box` 就能在这两种状态之间进行切换。这一命令也同样适用于 3D 图形。

`datetick(axis, format)`

根据日期格式 `format` 格式化在坐标轴 `axis` 上的文本。参数 `axis` 可以是 'x' (缺省值)，'y' 或者 'z'。可参见 2.5 节可获得关于日期格式的更多信息。

`dragrect(X, step)`

允许用户在屏幕上拖动矩形。这些矩形是由  $n \times 4$  的矩阵 `X` 中每一行确定的。如果给定 `step`，则只能在所给大小的偶数次内拖动矩形。

`grid on`

在图形窗口中画出网格。如果前面的图形是比如用极坐标 (参见 13.2 节) 绘制的，则网格也将采用极坐标绘制。

<code>grid off</code>	从图形窗口中清除网格。
<code>grid</code>	在 <code>grid on</code> 和 <code>grid off</code> 之间切换。
<code>zoom on</code>	使得用户可以在图形窗口中通过点击鼠标左键来放大二维图形，点击右键就缩小二维图形。还可以通过“点击和拖动”来选定一个区域。调整坐标轴刻度使得选中的区域占满整个图形窗口。
<code>zoom off</code>	关闭 <code>zoom</code> 功能。
<code>zoom out</code>	复位为满刻度。
<code>zoom</code>	在 <code>zoom on</code> 和 <code>zoom off</code> 之间切换。
<code>zoom(factor)</code>	用 <code>factor</code> 缩放当前坐标轴。
<code>zoom axis</code>	标定坐标轴。如果 <code>axis</code> 是 <code>xon</code> 或 <code>yon</code> ，则仅标定坐标轴。

关于坐标轴和网格的控制更详细的内容可参考第 14 章。与 `axis` 命令关系最为密切的是 `caxis` 和 `saxis` 命令，分别用来设定颜色和声音的比例；参见 13.6 节和 13.8 节。

正如 2.3 节所描述的那样，命令可以看作是带有字符串参数的函数。因此，`axis('square')` 也可以写成 `axis square`，而 `grid off` 也等价于 `grid('off')`。

#### 例 13.10

(a) 定义一个值的集合来表示单位圆：

```
t=0:0.2:2*pi+0.2;    % 角度参数。
x=sin(t);             % x 的值。
y=cos(t);             % y 的值。
```

下面的命令给出如图 13-17 所示的图形：

```
plot(x, y, '-');
```

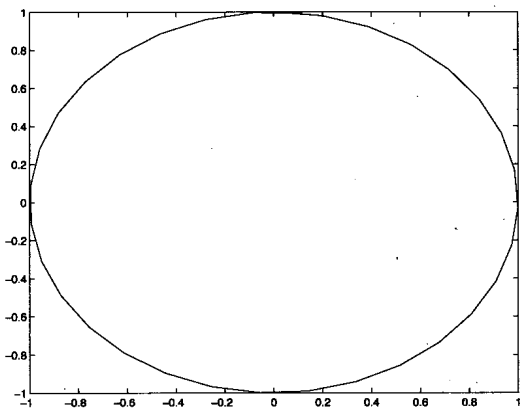


图 13-17 如果比例不调整，圆看上去就象椭圆

(b) 可以重新定义坐标轴的刻度使得圆看上去象圆，并且这次画上网格。

```
axis('square');      % 调整刻度
grid on;             % 绘制网格
```

这些命令可以得到图 13-18 上半部分的图形。下半部的图形是通过下列命令得到的：

```
axis('normal');           % 坐标轴复位
grid off;                 % 关闭网格
axis([-2 2 -3 3]);       % 改变坐标轴刻度
```

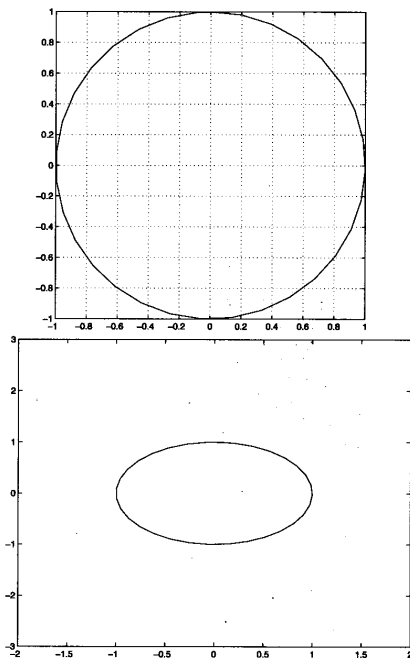


图13-18 上图:坐标轴设为正方形;下图:手动设定坐标轴刻度

有若干个命令可以用来在图形窗口中输出文本。命令 `title`、`xlabel`、`ylabel` 和 `zlabel` 都可以输出标准文本项。MATLAB 拥有大量的特殊字符和一些格式化函数，这些内容都在 MATLAB Help Desk 中关于文本部分做了详细的介绍。所有这些命令都可以用在当前子窗口中，并且通常应该在窗口中画完图形之后给出。要改变字体和其他属性，参见 14.2 节。

#### 命令集 133 图形窗口中的文本

<code>title(txt)</code>	在图形窗口顶端的中间位置输出字符串 <code>txt</code> 作为标题。
<code>xlabel(txt)</code>	在 <code>x</code> 轴下的中间位置输出字符串 <code>txt</code> 作为标注。
<code>ylabel(txt)</code>	在 <code>y</code> 轴边上的中间位置输出字符串 <code>txt</code> 作为标注。
<code>zlabel(txt)</code>	在 <code>z</code> 轴边上的中间位置输出字符串 <code>txt</code> 作为标注。
<code>text(x,y,txt)</code>	在图形窗口的 $(x, y)$ 处写字符串 <code>txt</code> 。坐标 <code>x</code> 和 <code>y</code> 按照与所绘制图形相同的刻度给出。对于向量 <code>x</code> 和 <code>y</code> ，字符串 <code>txt</code> 写在 $(x_i, y_i)$ 的位置上。如果 <code>txt</code> 是一个字符串向量，即一个字符矩阵，且与 <code>y</code> 有相同的行数，则第 <code>i</code> 行的字符串将写在图形窗口的 $(x_i, y_i)$ 的位置上。
<code>text(x,y,txt,'sc')</code>	在图形窗口的 $(x, y)$ 处输出字符串 <code>txt</code> ，给定左下角的坐标为 $(0.0, 0.0)$ ，右上角的坐标则为 $(1.0, 1.0)$ 。
<code>gtext(txt)</code>	通过使用鼠标或方向键，移动图形窗口中的十字光标，

```
legend(str1,str2,
...pos)
```

```
legend(H, str1,
str2,...)
legend off
```

让用户将字符串 `txt` 放置在图形窗口中。当十字光标走到所期望的位置时，用户按下任意键或鼠标上的任意按钮，字符串将会写入在窗口中。

在当前图上输出图例，并用说明性字符串 `str1`, `str2` 等作为标注。如果指定参数 `pos`，则图例将按下面所述放置：

- 1： 将图例框放在坐标轴外的右侧。
- 0： 将图例框放在坐标轴内侧，以便最少的点被覆盖。
- 1： 将图例框放在右上角。
- 2： 将图例框放在左上角。
- 3： 将图例框放在左下角。
- 4： 将图例框放在右下角。

`[x,y]` 将图例框的左下角移动到坐标 `(x,y)` 指定的位置。

返回句柄 `H` 以得到适当曲线的合适字符串。这对于用矩阵作为输入来画图是必要的。

从当前图形中清除图例。

命令 `num2str`、`int2str`、`sprintf` 等用于将数字转化成字符串的命令（参见5.1.2节）是十分有用的，甚至有时还可以和文本命令一起使用。

### 例13.11

(a)下面这个简单程序是用来完成随机路径的，可以看作是模拟空气粒子的运动。

该程序保存为 `particle.m`：

```
% 随机路径。一个粒子从原点出发，随机地向任意方向移动，每步移动半个单位。
disp('Give the number of steps') % 步数。
n=input(' >>> ');
x=cumsum(rand(n, 1)/0.5); % 随机x值。
y=cumsum(rand(n, 1)/0.5); % 随机y值。
clf; % 清除图形窗口。
plot(x,y); % 绘制路径。
hold on; % 保持当前图形。
plot(x(1), y(1),'o', x(n), y(n),'o'); % 标记起点/终点。
axis=axis; % 获取最小值和最大值。
scale=axis(2) - axis(1); % 计算比例。
text(x(1)+scale/30, y(1),'Start'); % 在起点和终点右侧。
text(x(n)+scale/30, y(n),'Finish'); % 标注文本。
hold off; % 将保持状态开关恢复到标准状态。

xlabel('x'); ylabel('y');
title('Random walk')
```

键入命令 `particle` 来运行程序，将有：

```
Give the number of steps
>>> 100
```

现在，得到图13-19。

(b) 如果对程序中写文本的几行命令加以改动，比如，改为：

```
text(x(1)+scale/30,y(1),'Start','FontSize',20,'FontName',...
                                     'Times');
text(x(n)+scale/30,y(n),'Finish','FontSize',20,'FontName',...
                                     'Times');

xlabel('x','FontSize',18,'FontWeight','bold');
ylabel('y','FontSize',18,'FontWeight','bold');
title('Random walk','FontSize',18,'FontWeight','bold');
```

得到图13-20：

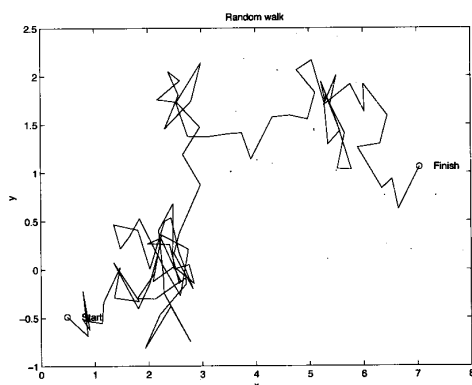


图13-19 随机路径

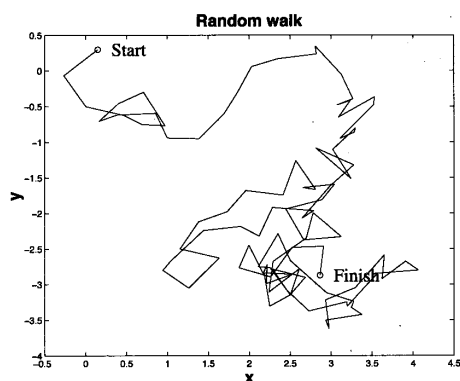


图13-20 改变文本格式后的随机路径

命令 `ginput` 通常用于从图形窗口中获取信息。该命令在图形窗口中放置一个光标，用户可以通过键盘或鼠标移动此光标。当移动到预定位置时，按下任意键或鼠标按钮，坐标值将被传递到 MATLAB 中。如果没有指定读取的坐标数目，MATLAB 将读取每一个键盘或鼠标按钮按下时的值，直到按‘回车’键停止。

#### 命令集134 从图形窗口中读取数据

`[x,y]=ginput`

从图形窗口中读取坐标值。在图形窗口中放置一个光标，用户可以通过鼠标或方向键对光标进行定位，并且通过按下鼠标按钮或键盘上任意键，将坐标值传递到 MATLAB 中。这些坐标值保存在向量 `x` 和 `y` 中。这一过程直到按下‘回车’键才终止。

`[x,y]=ginput(n)`

从图形窗口中读取  $n$  个坐标值。

`[x,y,t]=ginput(...)`

按下鼠标键以整数值的形式返回到向量中。按下左键返回，中键返回，右键返回。如果使用键盘，将返回按下键的 ASCII 码。

`waitforbuttonpress`

停止运行 MATLAB 直到在当前图中按下鼠标或键盘上的任意键。如果按下鼠标，则 `waitforbuttonpress` 返回 0；如果按下键盘上的任意键，则返回 1。

`rbbox`

在当前图的橡皮圈框周围画虚线。可以和 `waitforbuttonpress` 一起使用来进行动态控制。比如：在 `oom` 中使用该命令。

## 例13.12

在MATLAB4中可以使用`ginput`命令重新定义图形窗口，以使得(0,0)为左下角而(1,1)为右上角。而在MATLAB现在的版本中，不再这样使用。但是，使用下面用户所定义的函数`ginput01`可以很容易地创建一个这样的窗口。

```
function [x,y,button] = ginput01(N);

if (nargin == 0), N = inf; end

[x,y,button] = ginput(N);
xlim = get(gca,{'xlim','ylim'}); % 获取坐标轴的范围
x = (x-xylim{1}(1))/diff(xylim{1});
y = (y-xylim{2}(1))/diff(xylim{2}); % 用坐标轴范围重新定义x,y 的大小
```

`get`命令在将14章中介绍。

## 例13.13

`ginput`和`waitforbuttonpress`命令提供给MATLAB程序员用来建立简单的交互式程序。下面的M文件就使用这两个命令来绘制由用户确定的点连成图形。绘制完成后，程序等待，当用户点击图形时，删除图形。

% 交互式绘图的M文件

```
n = figure; % 创建新的图形

disp('To draw a line in the figure:')
disp('Press the left mouse button in the figure for start,')
disp('each bend and stop of the line. ')
disp('Press right mouse button when finished.')

[x,y,t] = ginput(1); % 读取第一次鼠标键按下时的坐标
plot(x,y,'o') % 用圆圈作一标记
xx = x; yy = y; % 保存坐标
hold on; axis([0 1 0 1]); % 保持图形锁定坐标轴

while t ~= 3 % 如果不是右键按下
    [x,y,t] = ginput(1); % 则读新的坐标
    plot(x,y,'o') % 并用圆圈作一标记
    xx = [xx x]; % 保存坐标
    yy = [yy y];
end

clf; line(xx,yy); % 清除图形并画一条线

disp('Click on the figure when you are done')

waitforbuttonpress; % 等待直到用户在图形中按下鼠标注意键
delete(n); % 清除图形
```

命令`figure`、`delete`和`line`将在14.2节中介绍。其他的交互式命令在14.3节中介绍。

上面的M文件运行后给出如图13-21所示的图形。

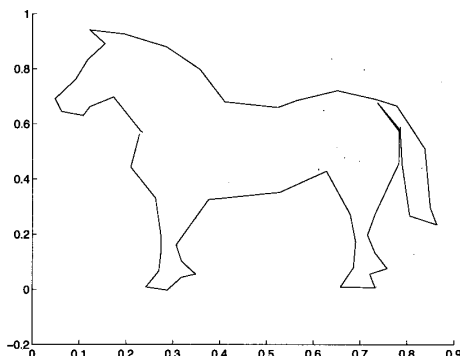


图13-21 例13.13中交互式程序运行的结果

### 13.4 生成网格和绘制等高线图

可以在二维和三维空间中画出带有两个变量的函数如 $z=f(x, y)$ 的等高线图形。前者用`contour`命令完成，而后者可用命令`contour3`完成。值得注意的是，MATLAB5可以处理不统一的网格。

#### 命令集135 等高线图形

<code>contour(Z)</code>	绘制矩阵 $Z$ 的等高线图形。将 $Z$ 中各元素看成是高于 $(x, y)$ 平面的高度。如果 $Z$ 是一个 $m \times n$ 的矩阵，则横轴的刻度设为 1 到 $n$ ，而纵轴的刻度设为 1 到 $m$ 。命令 $c=\text{contour}(Z)$ 返回将被如命令 <code>clabel</code> 使用的等高线矩阵 $C$ ，参见 <code>contourc</code> 。
<code>contour(Z,n)</code>	绘制 $n$ 条等高线。如果不指定 $n$ ，则绘制 10 条。
<code>contour(Z,v)</code>	绘制向量 $v$ 中的值所确定高度的等高线。
<code>contour(x,y,Z)</code>	用向量 $x$ 和 $y$ 作为矩阵 $Z$ 的坐标绘制 $Z$ 的等高线图形，即用 $x$ 和 $y$ 设置坐标轴的刻度。
<code>contour(x,y,Z,n)</code>	用 $x$ 和 $y$ 设置坐标轴的刻度，绘制 $n$ 条等高线。
<code>contour(x,y,Z,v)</code>	用 $x$ 和 $y$ 设置坐标轴的刻度，绘制向量 $v$ 中的值所确定的高度的等高线。
<code>contour(...,str)</code>	用字符串 $str$ 指定的线型和颜色绘制等高线。参见 13.1 节的 <code>plot</code> 指令和表 13-1。
$C=\text{contourc}(...)$	不画出等高线，而使用 <code>contour</code> 和 <code>clabel</code> 计算出等高线矩阵 $C$ 。 $C$ 是一个两行矩阵，对于每条等高曲线都是连续保存它们的图段。使用 <code>type help contour</code> 可以获得更多的信息。
$C=\text{contours}(...)$	计算等高线矩阵 $C$ 。当边界为非矩形区域时， <code>contour</code> 使用该矩阵。 <code>type help conto</code> 可以获得更多的信息。
<code>contourf(Z)</code>	绘制矩阵 $Z$ 的填充等高线。与 <code>contour</code> 使用相同的参数。
<code>contour3(x,y,z,n)</code>	绘制 $n$ 条三维等高线，即不将等高线投影到 $(x, y)$ 平面上。返回 <code>clabel</code> 使用的等高线矩阵。

<code>clabel(C)</code>	在等高线图形上增加高度标记。标记的位置是任意选择的。矩阵C是用命令 <code>contour</code> 和 <code>contourc</code> 返回的等高线矩阵。
<code>clabel(C,v)</code>	用向量v中给出的等高线水平进行标记。矩阵C是用命令 <code>contour</code> 和 <code>contourc</code> 返回的等高线矩阵。
<code>clabel(C,'manual')</code>	在鼠标指定的位置放置等高标记。用户可以通过鼠标或键盘上的方向键来移动等高线图形中的光标,当按下某键时,写入数字进行标记。按‘回车’终止整个操作。

如果要画出已定义矩阵Z的等高线图形,可以使用`contour(Z)`或`contour3(Z)`命令。调色板一节的图P-4用四条极值曲线展示了对函数 $z=f(x,y)$ 使用`contourf`命令的结果。

#### 例13.14

(a) 假定已定义了图13-35中二维函数表面图的矩阵Z。那么,下列命令将得到图3-22所示的图形:

```
[X,Y] = meshgrid(-3:1/8:3);
Z      = peaks(X,Y).*sin(X);

v1 = -4:-1;
v2 = 0:4;

clf;

subplot(2,1,1);          % 上方子图
contour(Z,v1,'k-');      % 对z的负数绘制实线
hold on;

contour(Z,v2,'k--');     % 对z的非负数绘制实线
hold off;

subplot(2,1,2);          % 下方子图
C = contour(Z);           % 绘制等高线
clabel(C);                % 加入等高标记
```

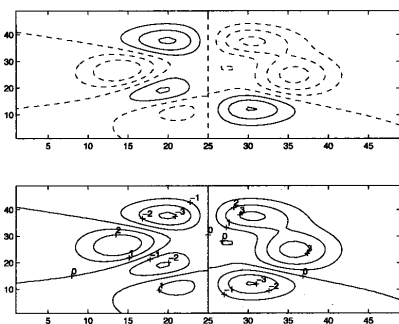


图13-22 等高线图形

(b) 也可以将`contour`用于非矩形网格。运行例 13.23中的程序可以得到图 13-23。在图 13-23中重新设置了文本字体来说明每个等高线可以有自己的图形句柄。参见第 14章中关于图



形对象和句柄部分以获得更多的信息。

`contourf`命令填充线与线之间的区域；如图 13-24所示。该图是用白线代替黑线来绘制的；参见例 13.23。

尽管如此，计算 $Z$ 还是很有必要的。这可以用两步来完成。首先，在希望绘制等高线的区域定义一个网格。该区域由长度分别为  $n$ 和 $m$ 的向量 $x$ 和 $y$ 来定义，这样 $x$ 和 $y$ 的值均在网格中。

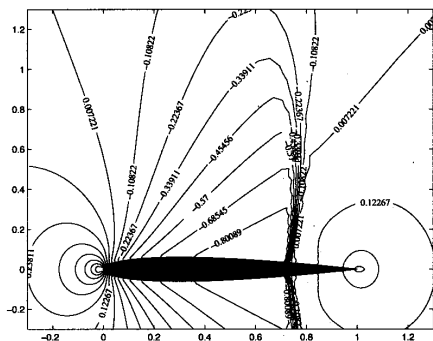


图13-23 气压的等高线、等压线和15个高度水平的剖面图

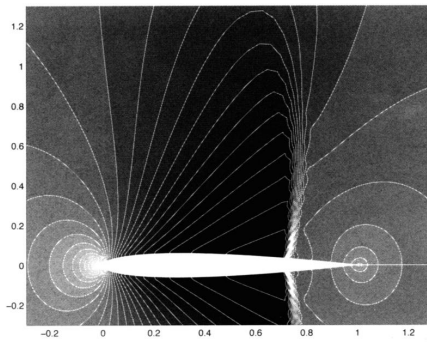


图13-24 `contourf`创建的等高线剖面图。高度水平数为30，且均匀分布在 - 1 ~ 1之间

注意：这里 $x$ 和 $y$ 的元素不一定是等距的。然后，用命令 `[U,V]=meshgrid(x,y)`来形成网格。实际上，网格就是两个矩阵 $U$ 和 $V$ ，包含着它的 $x$ 和 $y$ 坐标。矩阵 $U$ 由复制 $m$ 行的向量 $x$ 组成，而 $V$ 由复制 $n$ 列的向量 $y$ 组成。如图 13-25所示， $y$ 轴方向向下强调网格点和矩阵元素之间的一致性。

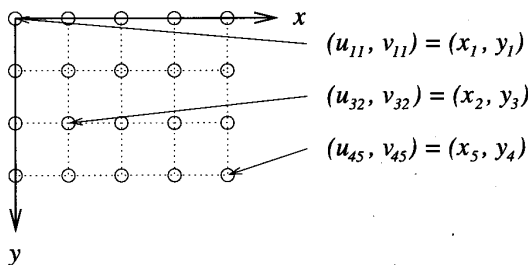


图13-25  $x$ 的五个值和 $y$ 的四个值形成的网格定义两个  $4 \times 5$  矩阵 $U$ 和 $V$ ，矩阵的元素来自向量 $x$ 和 $y$

使用`meshgrid`命令同样可以生成柱形网格和球形网格。

### 命令集136 网格的生成

`[U,V]=meshgrid`  
`(x,y)`

用来自向量 $x$ 和 $y$ 的 $x$ 坐标和 $y$ 坐标形成网格，并生成矩阵。长度为 $n$ 的向量 $x$ 包含升序排列的 $x$ 坐标，而长度为 $m$ 的向量 $y$ 包含升序排列的 $y$ 坐标，分别复制 $m$ 和 $n$ 次形成两个  $m \times n$ 的矩阵 $U$ 和 $V$ 。这些矩阵表示整个矩形区域内的  $x$ 和 $y$ 坐标。坐标对  $(u_{ij}, v_{ij})$ ,  $i=1, \dots, m$ ,  $j=1, \dots, n$ ，可通过使用命令  $Z=f(U,V)$ 用来计算  $z_{ij}=f(u_{ij}, v_{ij})$ ，参见图 13-25。

<code>[U,V]=meshgrid(x)</code>	等价于 <code>[U,V]=meshgrid(x,x)</code> 。
<code>[U,V,W]=meshgrid(x,y,z)</code>	以同样的方式产生三维网格, 可用来对含有三个变量的函数进行求值。
<code>[X,Y,Z]=cylinder(r,n)</code>	象 <code>meshgrid</code> 一样返回坐标矩阵, 返回的坐标形成圆柱体或圆锥体表面。圆柱体半径来自向量 $r$ , 包含沿圆柱周围 $n$ 个等距离的点。如果 $n$ 不确定, 缺省值为 20。如果 $r$ 和 $n$ 都不确定, 则令 $r=(1\ 1)$ , $n=20$ 。
<code>cylinder(r,n)</code>	同上画圆锥体, 但不返回坐标。
<code>[X,Y,Z]=sphere(n)</code>	返回在矩阵 $X$ 、 $Y$ 和 $Z$ 总共 $(n+1) \times (n+1)$ 个矩阵中的单位球体上的 $n$ 个等距坐标。
<code>sphere(n)</code>	同上绘制球体图形, 但不返回坐标。

在调色板一节所创建图 P-2 时多次用到 `sphere` 命令。

#### 例13.15

假定在单位正方形上定义一个网格  $U, V$ , 并且要求沿  $x$  轴有 5 个网格点, 沿  $y$  轴有 4 个网格点, 如图 13-25 所示的那样。首先定义向量  $x$  和  $y$ , 然后形成网格:

```
x = linspace(0,1,5);    % 定义x值
y = linspace(0,1,4);    % 定义y值

[U,V] = meshgrid(x,y)    % 形成网格

U =
    0    0.2500    0.5000    0.7500    1.0000
    0    0.2500    0.5000    0.7500    1.0000
    0    0.2500    0.5000    0.7500    1.0000
    0    0.2500    0.5000    0.7500    1.0000

V =
    0    0    0    0    0
    0.3333    0.3333    0.3333    0.3333    0.3333
    0.6667    0.6667    0.6667    0.6667    0.6667
    1.0000    1.0000    1.0000    1.0000    1.0000
```

第二步是在网格上对函数  $z=f(x, y)$  求值。在所定义的网格中, 即  $Z=f(U, V)$ 。这要求函数  $f$  用元素操作运算符来定义; 参见 3.5 节。

#### 例13.16

(a) 绘制下列三个函数的等高线图形。

$$\begin{cases} Z_1 = f_1(x, y) = \sin x \cdot \sin y & x, y \in [0, \pi] \times [0, \pi] \\ Z_2 = f_2(x, y) = x - 0.5x^3 + 0.2y^2 + 1 & x, y \in [-3, 3] \times [-3, 3] \\ Z_3 = f_3(x, y) = \sin(\sqrt{x^2 + y^2})/\sqrt{x^2 + y^2} & x, y \in [-8, 8] \times [-8, 8] \end{cases}$$

程序的第一部分绘制网格并对函数求值。程序的最后部分绘制图形。这个程序为 `contours.m`:

```
x = 0:0.2:3*pi;          % 生成坐标
y = 0:0.25:5*pi;
```

```

[XX,YY] = meshgrid(x,y);
Z1 = sin(XX).*sin(YY);           % 对Z1求值

x = -3:0.25:3;                   % 生成坐标
y = x;
[XX,YY] = meshgrid(x,y);
Z2 = XX - 0.5*XX.^3 + 0.2*YY.^2 + 1; % 对Z2求值

x = -8:0.5:8;                   % 生成坐标
y = x;
[XX,YY] = meshgrid(x,y);
r = sqrt(XX.^2+YY.^2) + eps;
Z3 = sin(r)./r;                 % 对Z3求值

clf;

subplot(2,2,1); contour(Z1);
title('sin(x)*sin(y)');

subplot(2,2,2); contour(x,y,Z3);
title('sin(r)/r');
subplot(2,2,3); contour3(Z2,15);
title('x-0.5x^3 + 0.2y^2 + 1');

subplot(2,2,4); contour3(x,y,Z3);
title('sin(r)/r');

subplot(2,2,3); rotate3d;

```

最后一行程序允许用户通过使用鼠标对左下角的图形进行旋转以获得更好的视图；  
`rotate3d`将在13.5节中介绍。运行该程序，经过旋转后，将得到图 13-26所示的结果。

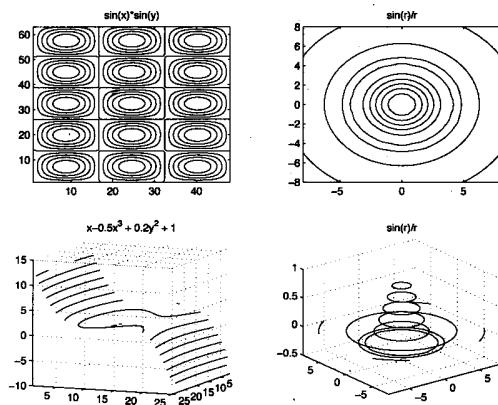


图13-26 一些函数的等高线图形

(b) 为了得到一个真实的函数侧面图，在等高线的图形中绘制梯度。梯度可以用命令 `gradient` (参见6.2节) 计算，并且可以用命令 `quiver` 绘图 (参见13.2节)。这个有趣的图形可以通过下列语句得到：

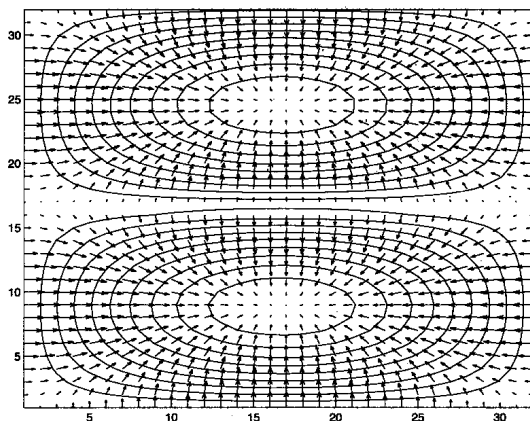


图13-27 带有梯度的等高线图形

```
[X,Y]      = meshgrid(-pi/2:0.1:pi/2,-pi:0.2:pi);
Z          = abs(sin(Y).*cos(X));
[DZDX,DZDY] = gradient(Z,.1,0.2);

contour(Z);      hold on;
quiver(DZDX,DZDY); hold off;
```

结果如图 13-27 所示。

### 13.5 三维图形

三维图形可以用命令 `plot3` 来绘制。该命令与 `plot` 类似，但是 `plot3` 需要 3 个向量或矩阵参数。与 `plot` 一样，线型和颜色可以用一个字符串来确定；参见表 13-1。

#### 命令集 137 三维图形

<code>plot3(x,y,z)</code>	用 $(x_i, y_i, z_i)$ 所定义的点绘制图形。向量 <code>x</code> 、 <code>y</code> 和 <code>z</code> 必须为等长度的。
<code>plot3(X,Y,Z)</code>	对矩阵 <code>X</code> 、 <code>Y</code> 和 <code>Z</code> 的每一列绘图。这些矩阵必须大小相等。或者，也可以是长度与矩阵列向量相等的向量。
<code>plot3(x,y,z,str)</code>	使用字符串 <code>str</code> 确定的线型和颜色按照上面所述的方法绘制图形。参见表 13-1 以获得允许使用的字符串值。
<code>plot3(x1,y1,z1, str1,x2,y2,z2, str2,...)</code>	用字符串 <code>str1</code> 确定的线型和颜色对 <code>x1,y1,z1</code> 绘图，用字符串 <code>str2</code> 确定的线型和颜色对 <code>x2,y2,z2</code> 绘图…。如果省略 <code>str1, str2, …</code> ，MATLAB 将自动选择线型和颜色。

#### 例 13.17

受例 13.11 的启发，现在可以编写一个程序来模拟三维空间的随机路径。程序 `particle3.m` 如下：

```
% 三维空间的随机路径
% 用随机数创建向量 x、y 和 z。绘制“路径”。
```

```

n = input('Give the number of steps : ');
x = cumsum(rand(1,n)-0.5);
y = cumsum(rand(1,n)-0.5);
z = cumsum(rand(1,n)-0.5);

plot3(x,y,z); % 绘制路径
grid on; % 显示网格
text(x(1),y(1),z(1),'Start','FontSize',20); % 标记起点
text(x(n),y(n),z(n),'Finish','FontSize',20); % 标记终点

```

运行程序，将得到如图 13-28 的结果。

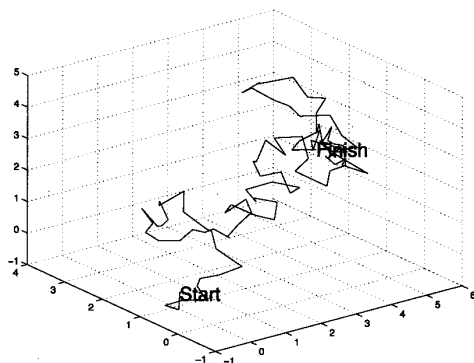


图13-28 三维空间中的随机路径

三维图形中的写文本命令与二维中相同，即 `title`、`text`、`xlabel`、`ylabel` 和 `zlabel`；参见 13.3 节。

#### 命令集 138 三维绘图函数

<pre>bar3(x,A,width,format)</pre>	<p>矩阵 <b>A</b> 的各列对向量 <b>x</b> 绘制竖直接条图。如果给定 <i>width</i>，则每个竖直接条的宽度为 <i>width</i>。如果给定字符串 <b>format</b>，<b>format</b> 可以取下列字符串中的某个。</p> <p>‘detached’ 给出默认的分离的饼图。</p> <p>‘grouped’ 给出一组饼图。</p> <p>‘stacked’ 给出堆叠式饼图。</p> <p><code>linespec</code> 使用指定的线条颜色；参见表 13-1。</p>
<pre>bar3(A)</pre>	<p>等价于 <code>bar3(1:size(A,1),A)</code>。</p>
<pre>bar3h(x,A,format,width)</pre>	<p>矩阵 <b>A</b> 的各列对向量 <b>x</b> 绘制水平条形图，如果给定 <i>width</i>，则每个水平条的宽度为 <i>width</i>。如果给定字符串 <b>format</b>，<b>format</b> 的取值与 <code>bar3</code> 相同；参见 <code>bar3</code> 所述。</p>
<pre>pie3(x,explode)</pre>	<p>类似饼图的三维饼图。</p>
<pre>quiver3(x,y,z,u,v,w,s,format)</pre>	<p>在由三个向量 <b>x</b>、<b>y</b> 和 <b>z</b> 所确定的坐标处绘制箭头。箭头的长度和方向由向量 <b>u</b>、<b>v</b> 和 <b>w</b> 确定，坐标刻度由 <i>s</i> 确定，<i>s</i> 的默认值为 1。字符串 <b>format</b> 是可选项，用来确定线条的格式。</p>

<code>ribbon(x,y,width)</code>	向量 $y$ 对 $x$ 绘制三维丝带图,而不再是普通曲线。 $width$ 指定丝带的宽度,缺省值为0.75。
<code>scatter3(x,y,z, size,color)</code>	在 $x$ , $y$ 和 $z$ 所确定的点处绘制着色的圆圈图形, $x$ , $y$ 和 $z$ 的大小必须相等。可参考 <code>scatter</code> 以获得更多的信息。
<code>stem3(x,y,A)</code>	矩阵 $A$ 对向量 $x$ 和 $y$ 绘制离散序列数据图。
<code>stem3(A)</code>	矩阵 $A$ 对规格化的 $xy$ 平面绘制离散序列数据图。
<code>stem3(...,format)</code>	按照 <code>format</code> 所指定的格式绘制离散序列数据图。 <code>format</code> 除了可以是标准线条格式外,还可以是字符串'filled',它表示将填充上方的圆圈。
<code>trimesh(Tri,x,y,z)</code>	按矩阵 $Tri$ 绘制三角网格表面图。向量 $x$ , $y$ , $z$ 定义三角形。参见第6章关于三角形的介绍。
<code>tirsurf(Tri,x,y,z)</code>	按矩阵 $Tri$ 绘制三角区域的表面图。向量 $x$ , $y$ 和 $z$ 定义三角形的各个角。

调色板一节的图P-6演示了命令`bar3`和`pie3`的使用。

#### 例13.18

现有一向量 $x = [1 \ 2 \ 3 \ 4 \ 5]$ ,要绘制一个三维饼形图,并且将其中最大的部分分离出来。为此,需要向量`explode`=[0 0 0 0 1],除与 $x$ 中最大元素值相对应的元素为1外,其他元素均为0。如果输入`pie3(x,explode)`,将得到如图13-29所示的结果。

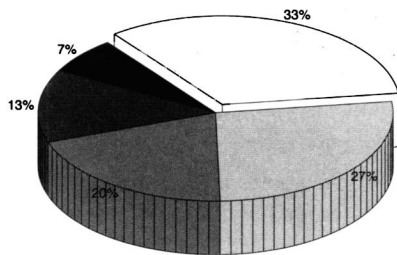


图13-29 最大部分分离出来的三维饼形图

二维图形中,可以用`comet`命令动态显示。同样,三维图形中可以用`comet3`命令动态显示。

#### 命令集139 三维彗星图

<code>comet3(x,y,z)</code>	显示函数 $z=f(x,y)$ 的一个三维彗星轨线动画,即一个带有位于 $(x_i, y_i, z_i)$ 坐标之间的慧尾的星状图。
<code>comet3(x,y,z,p)</code>	显示同上所述的彗星轨线动画,慧长为 $p * \text{length}(y)$ $p$ , 的缺省值为0.1。

在MATLAB中,可以通过下面的方法画出函数 $z=f(x,y)$ 的表面图:

- 1) 如13.4节所描述的那样形成网格;
- 2) 估计 $Z=f(U,V)$ , 矩阵 $U$ 和 $V$ 分别是来自 $x$ 和 $y$ 的坐标形成的矩阵;
- 3) 用MATLAB中的某一表面图命令绘制表面图。注意: 网格不一定是矩形。如果不是, 则网格坐标必须包含在所调用的函数中。

命令集140中的命令用来绘制三维网格表面图。

## 命令集140 三维网格表面图

<code>mesh(Z)</code>	将矩阵 <b>Z</b> 中的各个元素作为矩形网格上的高度，对这些值绘图，并且将相邻的点连接形成三维网格表面图。颜色由高度，即 <b>Z</b> 中的元素指定。
<code>mesh(Z,C)</code>	将矩阵 <b>Z</b> 中的各个元素看成矩形网格上的高度，对这些值绘图。各点的颜色由矩阵 <b>C</b> 中各元素的值确定。
<code>mesh(U,V,Z,C)</code>	绘制矩阵 <b>Z</b> 中元素的函数网格表面图，相邻点用线连接。该图画在三维视图中，元素 $z_{ij}$ 代表网格点 $(x_i, y_j)$ 上的高度。观察点是自动设定的，可以使用 <code>view</code> 命令来更改。参数如下： <b>U</b> $x$ 坐标矩阵 <b>V</b> $y$ 坐标矩阵 <b>Z</b> $z$ 坐标的矩阵，通常 $z_{ij} = f(u_{ij}, v_{ij})$ <b>C</b> 各点的颜色矩阵。如果 <b>C</b> 省略，则 <b>C</b> = <b>Z</b> 。
<code>meshc(...)</code>	与 <code>mesh</code> 一样绘制网格表面图，并在该图的下面绘制等高线图。
<code>meshz(...)</code>	与 <code>mesh</code> 一样绘制网格表面图，并在 $(x, y)$ 平面上绘制参考平面。
<code>waterfall(...)</code>	类似 <code>meshz</code> ，但是参考平面只在一个方向上绘制。
<code>hidden val</code>	切换消隐线的移动状态。变量 <code>val</code> 可以为 <code>on</code> ，表示不绘制消隐线；也可以为 <code>off</code> ，表示绘制消隐线。这一命令只适用于命令 <code>mesh</code> 。

可以使用命令 `rot90` 对由矩阵定义的图形进行旋转。此外，在本节后面，还将看到 `view` 命令。使用 `rotate3d` 命令很容易决定一个观察点。

## 命令集141 矩阵旋转

<code>rotate3d val</code>	允许用户使用鼠标旋转一个三维图形。如果指定 <code>val</code> ，则有两种可能， <code>on</code> 表示打开该功能，或者 <code>off</code> 表示关闭该功能。
<code>rot90(A)</code>	返回逆时针旋转90度后的矩阵 <b>A</b> 。常常和命令 <code>mesh</code> 一起使用。
<code>rot90(A,k)</code>	返回逆时针旋转 $k \times 90$ 度后的矩阵 <b>A</b> 。

必须强调的是命令 `mesh` 对获得一个矩阵的图像非常有用。另外，它对理解数值方法也是十分有帮助的。

## 例13.19

(a) 编写一个 MATLAB 程序来绘制与例 13.16 中相同函数的三维网格表面图。假定定义了例 13.16 中相同的矩阵 **Z1**、**Z2** 和 **Z3**。给出下列语句：

```
% 矩阵Z1、Z2和Z3将在contours中定义
..

% 绘制四个函数的图形

clear;
contoursProg;                                % 运行contours.m
```

```
disp('Matrices defined!');
clf;

subplot(2,2,1), mesh(Z1)
title('sin(x)*sin(y)');

subplot(2,2,2), meshz(Z2)
title('x - 0.5*x^3 + 0.2*y^2 + 1');

subplot(2,2,3), waterfall(Z2)
title('x - 0.5*x^3 + 0.2*y^2 + 1');

subplot(2,2,4), meshc(Z3)
title('sin(r)/r')
```

运行上面的程序，将得到如图 13-30所示的结果。

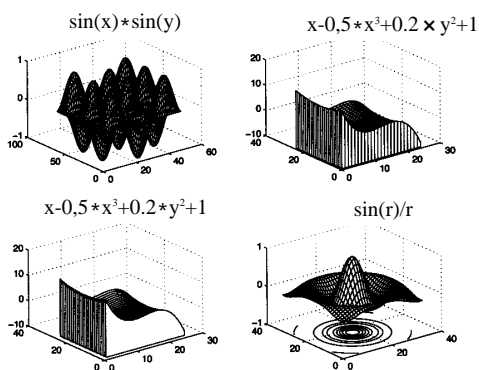


图13-30 用各种mesh命令绘制的几个函数图形

(b) 下面的MATLAB程序用来对给定的矩阵A进行LU和QR分解，并在图形窗口中绘制矩阵L、U、Q和R的四个子图。这些命令保存在文件 luqrmesh.m中：

```
if ~exist('A')
    A = input('Give a matrix A: ');
else
    disp('The following matrix exists:');
    A
end

[L,U] = lu(A);
[Q,R] = qr(A);
subplot; mesh(A); title('The matrix A');
disp('Press any key when you are ready!');
pause; clf;

subplot(2,2,1); mesh(L); title('The matrix L');
subplot(2,2,2); mesh(U); title('The matrix U');
subplot(2,2,3); mesh(Q); title('The matrix Q');
subplot(2,2,4); mesh(R); title('The matrix R');
```



假定已定义了矩阵A，输入命令luqrmesh运行程序：

The following matrix exists:

A =

30	1	7	2	6	5
4	24	9	4	0	1
9	9	21	6	1	7
8	3	7	21	3	9
4	9	1	6	27	9
1	6	4	2	3	9

Press any key when you are ready!

开始得到图13-31，按任意键后，将得到图13-32。

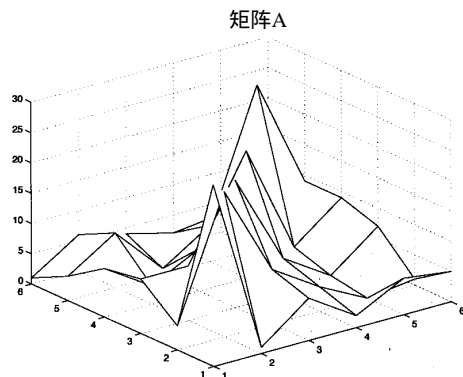


图13-31 矩阵A的三维网格表面图

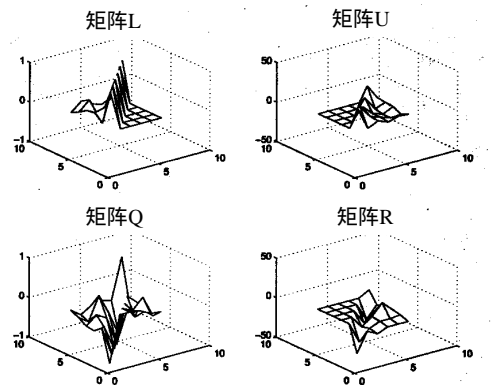


图13-32 矩阵L、U、Q和R的三维网格表面图

在MATLAB中有大量特殊的三维图形函数。

在本节开头详细讨论了如何绘制三维网格表面图。用同样的方式，还可以绘制阴影表面图：用各点的x和y坐标来创建两个矩阵，函数求值，用适当的命令绘制出图形。

用命令fill、fill3、surf、surfc和surfl也可以绘制出表面图形。命令surfl给出的带光照的阴影表面图。它是建立在漫反射、单向反射和周围的亮度模式的组合图基础之上的。这种表面图是由命令集146中定义的shading inter命令得到的，并且以灰度图的形式展现出来。使用这些数据可以创建出正常的表面向量，并且获得漫反射或单向反射的反射比。

#### 命令集142 表面图和亮度

surf(X,Y,Z,C)	绘制出由坐标 $(X_{ij}, Y_{ij}, Z_{ij})$ 确定的表面图形。如果X和Y分别是长度为m和n的向量，那么，Z必须为 $m \times n$ 的矩阵，并且表面是由 $(X_{ij}, Y_{ij}, Z_{ij})$ 来定义的。如果省略参数X和Y，MATLAB将使用统一的矩形网格。图形的颜色由矩阵C中的元素定义。如果省略参数C，则缺省值为 $C=Z$ 。
surfc(X,Y,Z,C)	除完成surf(... )所完成的功能外，还将在表面下方绘制等高线图。
surfl(X,Y,Z,ls)	与surf(... )功能类似，同时还在 $ls=[v, h]$ 或 $ls=[X, Y, Z]$ 方向上有一束光源入射，参数与命令view的参数相同。

<code>surfl(X,Y,Z, ls,r)</code>	与上面的命令类似，使用向量 $\mathbf{r}=[ambient, diffuse, specular, spread]$ 可以设置周围的亮度、漫反射、单向反射以及散射系数所产生的影响
<code>surfnorm(X,Y,Z) [Nx,Ny,Nz]= surfnorm(X,Y,Z)</code>	与 <code>surf</code> 相似，同时还画出法线。 画出矩阵 $\mathbf{X}$ 、 $\mathbf{Y}$ 和 $\mathbf{Z}$ 定义的表面法线，但不绘制图形。因此， $(nx_{ij}, ny_{ij}, nz_{ij})$ 是定义点 $(X_{ij}, Y_{ij}, Z_{ij})$ 处法线的向量。法线长度为1。
<code>diffuse(Nx,Ny,Nz, ls)</code>	使用兰伯特定律，返回法向量分量为 $N_x, N_y$ 和 $N_z$ 的表面漫反射率。 $\mathbf{ls}$ 是定义光源位置的三组分向量。
<code>specular(Nx,Ny, Nz,ls,v)</code>	使用光源位置 $\mathbf{ls}$ 和观察点 $\mathbf{v}$ ，返回法向量分量为 $N_x, N_y$ 和 $N_z$ 的表面反射率。
<code>light</code>	创建带有标准值的光源。
<code>light(propstr, val,...)</code>	创建光源，同时将它的属性 <code>propstr</code> 设置为 <code>val</code> 。可以同时多个属性进行设置。设置属性需要一些关于图形对象的知识；参见第14章，特别是关于光源对象的表14-26。
<code>lightangle (azimuth,height)</code>	创建无穷远处的光源对象，为了定位而使用球面系统， $azimuth$ 是水平的旋转角度，而 $height$ 表示“高度”。
<code>camlight (azimuth,height)</code>	创建观察点坐标系统的光源。为了定位而使用球面系统， $azimuth$ 是水平的旋转角度，而 $height$ 表示“高度”。
<code>camlight headlight</code>	在观察点位置创建光源。
<code>camlight right</code>	在观察点的右上方创建光源。
<code>camlight left</code>	在观察点的左上方创建光源。
<code>camlight(..., type)</code>	设置光源类型。字符串 <code>type</code> 可以是‘local’(缺省值)或‘infinite’。
<code>lighting mode</code>	改变多边形或表面对象，如用 <code>surf</code> 或 <code>patch</code> 等绘制的图形亮度模式。模式值可以是： <code>flat</code> 、 <code>gourand</code> 、 <code>phong</code> 和 <code>none</code> 。关于这些模式的更多的信息可在 <code>helpdesk</code> 中找到。
<code>material mode</code>	改变多边形或表面对象，如用 <code>surf</code> 或 <code>patch</code> 等绘制的图形反射比模式。模式值可以是： <code>skiny</code> 、 <code>dull</code> 和 <code>metal</code> 。更多的信息可参见 <code>helpdesk</code> 。
<code>material({ka kd kn n sc})</code>	为表面的反射比设置不同的值。参见 <code>helpdesk</code> 中关于不同属性的含义。  <code>ka</code> 改变AmbientStrength的属性。 <code>kd</code> 改DiffuseStrength的属性。 <code>kn</code> 改变SpecularStrength的属性。 <code>n</code> 改变SpecularExponent的属性。 <code>sc</code> 改变SpecularColorReflectance的属性。
<code>pcolor(Z)</code>	绘制矩阵 $\mathbf{Z}$ 的伪色图为带有颜色的细胞矩形阵列。图形颜色

	由矩阵 $Z$ 的元素值决定。
<code>pcolor(X,Y,Z)</code>	与 <code>surf(X,Y,Z); view(2)</code> 相同；参见命令集 143。
<code>fill(x,y,c)</code>	绘制出坐标向量 $x$ 和 $y$ 确定边角的多边形。多边形用字符串 $c$ 或与 $x, y$ 长度相等的向量 $c$ 中的值指定的颜色 (参见表 13-1) 填充。如果 $x, y$ 为矩阵，将对每一列绘制多边形。
<code>fill3(x,y,z,c)</code>	绘制 $x, y, z$ 确定的多边形。绘制出坐标向量 $x$ 和 $y$ 确定边角的多边形。多边形用字符串 $c$ 或向量 $c$ 中的值指定的颜色 (参见表 13-1) 填充。如果参数为矩阵，将对每一列绘制多边形。

这些命令中使用的颜色刻度可以进行调整；参见 13.6 节。

在调色板一节的图 P-2 中，使用了命令 `light` 来创建光源。

### 例 13.20

(a) 绘制带有等高线的  $(\sin r)/r$  函数图形。

```
x = -8:0.5:8;
[Xx Yy] = meshgrid(x);

R = sqrt(Xx.^2+Yy.^2) + eps;
Z = sin(R)./R;
clf;
surf(Xx,Yy,Z); title('(sin r)/r');
```

得到图 13-33。

(b) 现在我们绘制同一个函数的带有法线的图形。假设  $Xx$ 、 $Yy$  和  $Z$  与 (a) 中一样已经计算出来。下面的语句将得到图 13-34 的结果。

```
surfnorm(Xx, Yy, Z)
```

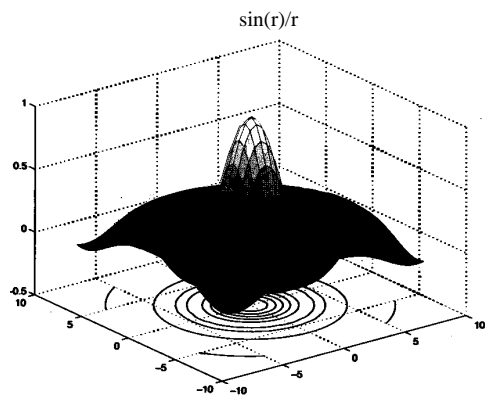


图 13-33 下方有等高线的钟形表面图

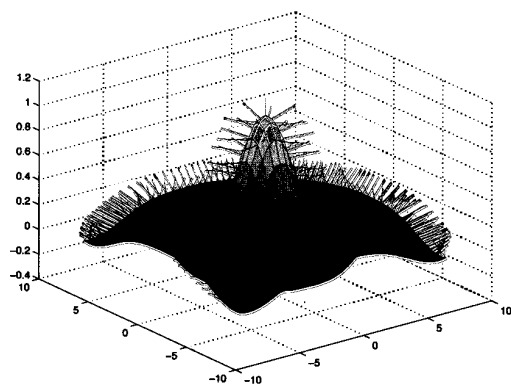


图 13-34 函数的带有法线的钟形图

## 例13.21

下面研究一下对周围亮度、漫反射和单向反射使用不同权值的命令 `surfl` 和 `surf`。将使用常用于三维图形命令演示的内建函数 `peaks`。为了达到更好的视觉效果，按照第 14 章所讲的对坐标轴进行处理，但是，在这里将省略一些专门的命令，而仅给出部分程序。给出如图 13-35 所示图形的程序如下：

```
[X,Y]      = meshgrid(-3:1/8:3);
Z          = peaks(X,Y).*sin(X);
[Nx,Ny,Nz] = surfnorm(Z);

s = [-3 -3 2];           % 光源位置
k1 = [0,1,0,0];          % 漫反射
k2 = [0,0,1,1];          % 单向反射
DD = diffuse(Nx,Ny,Nz,s);

disp('Press a key after each plot.');
```

```
clf;
```

```
colormap(gray);
axis([-3 3 -3 3 min(min(Z)) max(max(Z))]);
```

```
surfl(X,Y,Z,s);          shading interp; % 左上角
axis off; pause;
```

```
surfl(X,Y,Z,s,k1);       shading interp; % 右上角
axis off; pause;
```

```
surfl(X,Y,Z,s,k2);       shading interp; % 左下角
axis off; pause;
```

```
surf(X,Y,Z,DD);          shading interp; % 右下角
axis off; pause;
```

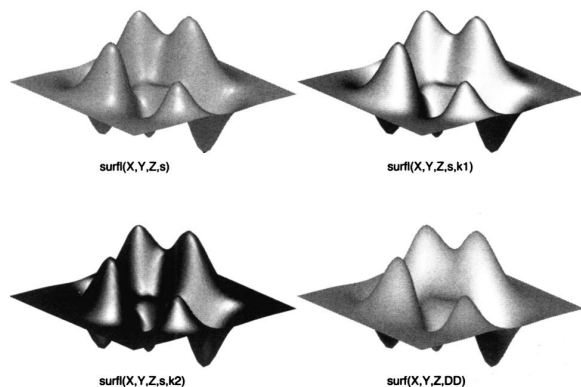


图13-35 不同亮度模式下的三维阴影表面图

从不同的角度观察，更容易观察一个图形。命令 `view` 用于改变图的观察点，同时可以确定观察点、方位角和仰角；参见图 13-36。还可以改变用命令 `viewmtx` 得到的视图。

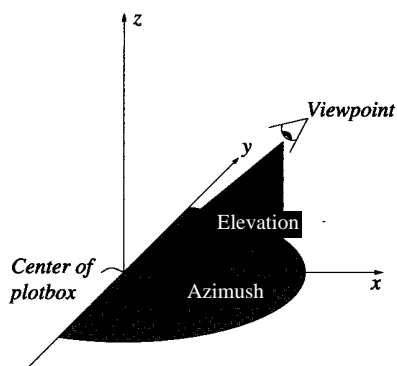


图13-36 命令 `view` 示意图

#### 命令集143 观察点和视图

<code>view(v,h)</code>	设置观察点角度。标量 $v$ 是方位角，即 $xy$ 平面上逆时针得到的角度。平面上方的仰角由标量 $h$ 设定。 $h$ 和 $v$ 都用度来衡量。
<code>[v,h]=view</code>	返回命令 <code>view</code> 当前使用的 $xy$ 平面内的角度 $v$ 和 $xy$ 平面上的角度 $h$ 。
<code>view(r)</code>	设置观察点位置 $r=(x\ y\ z)$ 。
<code>view(n)</code>	按下列值设置观察角度： $n=2$ 标准二维观察点，从上直接向下。 $n=3$ 标准三维观察点。
<code>view</code>	给出绘图时用于传递数据的 $4 \times 4$ 矩阵。
<code>view(T)</code>	MATLAB 在绘图时使用 $4 \times 4$ 矩阵 $T$ 。
<code>viewmtx(v,h,s,r)</code>	返回定义观察点和观察方向的 $4 \times 4$ 矩阵。使用 <code>help viewmtx</code> 可获得更多信息。

#### 例13.22

(a) 假设图形窗口中有图 13-33。命令 `view([1 0 0])` 将得到图 13-37 所示的同一个钟形表面图的侧视图。

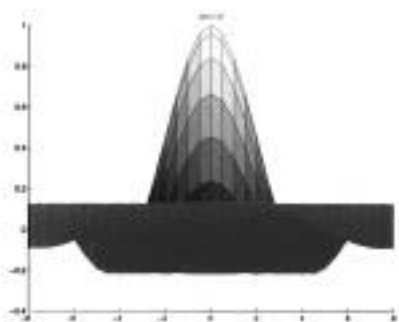


图13-37 从侧面观察的钟形表面图

(b) 矩阵 **Matlabmatrix** 由 1 和 0 组成，其中 1 形成字符 **MATLAB**。由于矩阵太大，就不在这里列出。下列命令用来绘制出该矩阵的三维网格表面图和稀疏结构图。参见矩阵的 9.3 节内容。这些命令保存在文件 **meshplot.m** 中：

```
% Matlabmatrix.m
% 创建一个由1和0组成的矩阵Matlabmatrix，代表MATLAB字符

Matlabmatrix; % 运行Matlabmatrix.m.

clf; % 清除图形

subplot(2,2,1); mesh(Matlabmatrix); % 绘制标准三维网格表面图
title('Standard view'); % 写标题

subplot(2,2,2); mesh(Matlabmatrix); % 绘制三维网格表面图
view([1 -4 2]); % 调整观察点
title('Viewed from position [1,-4,2]');

subplot(2,2,3); mesh(Matlabmatrix); % 等等
view([-1 -2 -7]);
title('Viewed from beneath');

subplot(2,2,4); spy(Matlabmatrix); % 命令spy显示非零元素
title('This is the matrix structure');
```

用 **meshplot** 运行程序；得到图 13-38。

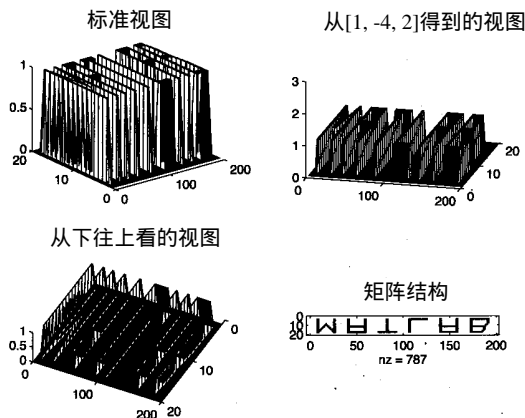


图13-38 用不同的方式给出的 MATLAB 矩阵图形

在使用命令 **spy**(参见 9.3 节)时，要注意的是原点被放在左上角。这是描述矩阵最好的方法，因为，书写和定义矩阵就用的是这种方法。

(c) 尽管命令 **view** 最多的是用在与三维图形相关的操作中，但也可以将其用于二维图形。如果在图形窗口中已有了图 13-37，该图是用二维图形命令 **plot** 创建的，那么，命令 **view([1 0.6 0.35])** 将在三维空间中显示出观察点为 (1, 0.6, 0.35) 的同样的圆；如图 13-39 所示。

```
view([1 0.6 0.35])
```

命令surf和mesh可以用于在非矩形网格中绘制函数。  
在对图形程序的调用中必须包括坐标矩阵。

### 例13.23

假设要研究机翼周围部分的空气气压。这里的计算和网格的生成将远非文字所能描述，但是在MATLAB中很容易将其图形画出。网格存储在X和Y中，矩阵P表示空气气压。用axis命令改变图形大小，并使用view(2)命令从上方观察图形。使用命令fill对图形中的机翼进行染色。

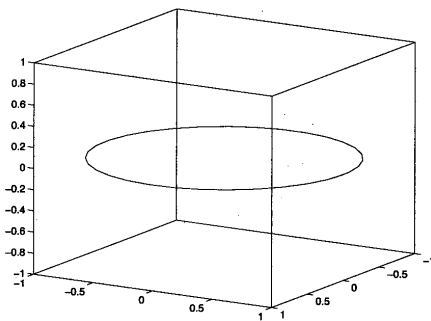


图13-39 空间圆

```
% NACA0012式机翼周围的气流所形成的二维欧拉方程的解
% 攻击角度 alpha=0
% Mach 代码=0.85. Roe的秒序方法
% 网格= 200*80点，保存在FoilXY、mat中
% 计算结果存在Foil Pressure, mat的矩阵P中

if ~exist('X')
    disp('X does not exist.');
```

% 载入求解的数据

```
    load FoilPressure;
    load FoilXY;
    % 载入画网格的数据

else
    disp('X exists');
```

% 定义机翼

```
end;

BodyX = X(:,1);
BodyY = Y(:,1);
BodyU = P(:,1);

maxP = max(max(P)); minP = min(min(P)); % 最大和最小压力

% 机翼周围的网格
clf;
mesh(X,Y,P,ones(size(X))); view(2); % 绘制网格
axis([-0.3 1.3 -0.3 1.3]); % 设定观察点
pause;

% 气压分布的表面图
clf;
surf(X,Y,P); shading interp; view(2); % 绘制表面图
axis([-0.3 1.3 -0.3 1.3]); % 设定观察点
pause;

% 用黑色等高线和标注的等高图
clf;
fill(BodyX,BodyY,'k'); hold on; % 绘制填充的机翼
axis([-0.3 1.3 -0.3 1.3]); % 设定观察点
[C,H] = contour(X,Y,P,15,'k'); % 绘制等高线
```

```

h = clabel(C,H);                                % 等高线仰角标注

for lev = 1:length(h)                            % 改变标注字体
    set(h(lev),'fontname','times');
end;
pause;

% 用contourf和白色等高线填充等高图
% 每两条线做一个标注
clf;
DrawLevel = linspace(-1,1,30);                  % 定义要画的线

[C,H] = contourf(X,Y,P,DrawLevel,'w');          % 绘制等高线
axis([-0.3 1.3 -0.3 1.3]);                      % 设定观察点
pause;

```

结果如图13-40、图13-41、图13-23和图13-24所示。图13-40中的网格是用与等大小的常数矩阵绘制的。

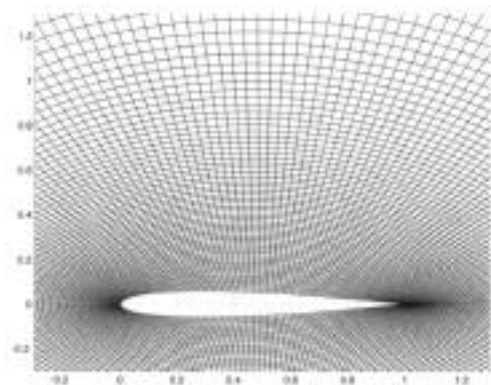


图13-40 机翼剖面周围的网格图形

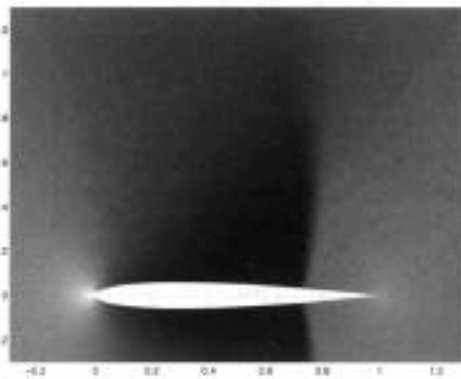


图13-41 机翼剖面周围的气压图形

camera是另一种更具体的控制观察点的方式。camera的属性如图13-42所示。

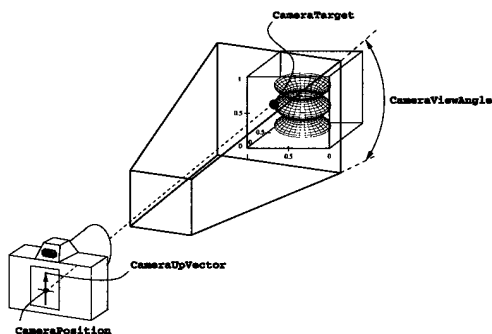


图13-42 观察点和它的属性

下列命令可用于控制观察点。



## 命令集144 观察点设置

`camdolly(dx,dy,dz, 按向量(dx,dy,dz)移动观察点。字符串dirval将决定观察目标dirval, coordsys)`是否随观察点一起移动。有两种选择：

‘movetarget’ 观察目标随观察点一起移动(缺省值)。

‘fixtarget’ 观察目标不随观察点移动。

字符串`coordsys`决定使用什么坐标系统。有三种选择：

‘camera’ 观察点中规格化的坐标系统， $x$ 轴指向右方， $y$ 轴指向上方(缺省值)。例如：  
`camdolly(1, -1, 0)`向右下移动观察点，以便原观察目标移动到右上角的位置。

‘pixels’ 是基于像素的坐标系统， $x$ 轴指向右方， $y$ 轴指向上方。此时，变量 $dz$ 将被忽略。

‘data’ 表示图形对象的坐标系统。

`camlookat` 移动观察点以观察绘制区域的某一特定部分。这需要了解有关各部分之间是如何联系的；参见第14章，特别是表14-12。

`camorbit(dteta,dfi, 绕观察目标旋转观察点。水平方向旋转delta度，竖直面coordsys, dirval)`向旋转 $dfi$ 度。字符串`coordsys`决定所使用的坐标系统，可以是‘camera’(默认)或‘data’(参见`camdolly`)。如果`coordsys`为‘data’，则将绕从观察目标到`dirval`方向的连线旋转。

`campan(dteta,dfi, 绕观察点旋转观察目标。水平方向旋转delta度，竖直面coordsys, dirval)`向旋转 $dfi$ 度。字符串`coordsys`决定所使用的坐标系统，可以是‘camera’(默认)或‘data’(参见`camdolly`)。如果`coordsys`为‘data’，则将绕从观察目标到`dirval`方向的连线旋转。

`campos` 返回观察点位置。

`campos([x y z])` 设置观察点位置。从现在起，观察点的位置将不再由MATLAB计算；见下面命令。

`campos('pos')` 决定是否由MATLAB自动计算观察点的位置。字符串 `pos` 可以是‘auto’或‘manual’。

`campos('mode')` 显示MATLAB是否计算观察点的位置。

`camproj` 返回当前投影。

`camproj(projec)` 设定当前投影。字符串 `projec` 可以是‘ortographic’(默认)或‘perspective’。

`camroll(dteta)` 绕由观察目标到观察点的连线逆时针旋转观察点  $dteta$ 度。

`camtarget` 返回观察目标。

`camtarget([x y z])` 设定观察目标。从现在起，观察目标将不再自动计算；见下面命令。

`camtarget('pos')` 决定是否由MATLAB计算观察目标。字符串 `pos` 可以是‘auto’或‘manual’。

`camtarget('mode')` 显示MATLAB是否计算观察点的位置。

<code>camup</code>	返回向量 <code>up</code> 。
<code>camup(up)</code>	设定观察点的 <code>up</code> 向量，即：图中向上的方向，到向量 <code>up</code> 中。 从现在起， <code>up</code> 向量将不再自动计算；见下面命令
<code>camup('pos')</code>	决定是否由 MATLAB 自动计算 <code>up</code> 向量。字符串 <code>pos</code> 可以是 'auto' 或 'manual'。
<code>camup('mode')</code>	显示 MATLAB 是否计算 <code>up</code> 向量。
<code>camva</code>	返回观察点的观察角度。
<code>camva(val)</code>	设置观察角度为 <code>val</code> 。从现在起，观察点的观察角度将不再自动计算；见下面命令
<code>camva('pos')</code>	决定是否由 MATLAB 自动计算观察点的观察角度。字符串 <code>pos</code> 可以是 'auto' 或 'manual'。
<code>camva('mode')</code>	显示 MATLAB 是否计算观察点的观察角度。
<code>camzoom(zoom)</code>	改变图形大小。 <code>zoom</code> 取大于 1 的数，图形变大；取 0~1 之间的数，图形变小。
<code>daspect</code>	返回当前刻度。
<code>daspect('pos')</code>	决定是否由 MATLAB 自动计算刻度。字符串 <code>pos</code> 可以是 'auto' 或 'manual'。
<code>daspect('mode')</code>	显示 MATLAB 是否计算刻度。
<code>daspect([x y z])</code>	决定绘图区域的 <code>x</code> , <code>y</code> 和 <code>z</code> 方向上刻度标定方式。从现在起， 刻度将不再自动计算；见下面命令。
<code>pbaspect</code>	返回当前刻度。
<code>pbaspect('pos')</code>	决定是否由 MATLAB 自动计算刻度。字符串 <code>pos</code> 可以是 'auto' 或 'manual'。
<code>pbaspect('mode')</code>	显示 MATLAB 是否计算刻度。

## 例13.24

再一次对图 13-33 进行观察。现在输入：

```
axis vis3d off      % 不绘制坐标轴
for x = 1:10
    camorbit(5,10); % 绕观察目标旋转10次
    drawnow         % 画出图形
end
for x = 1:10
    camroll(5);      % 绕观察目标到观察点的连线旋转10次
    drawnow
end
```

首先，绕观察目标旋转观察点，接着绕由观察目标到观察点的连线逆时针旋转。最后的结果如图 13-43 所示。

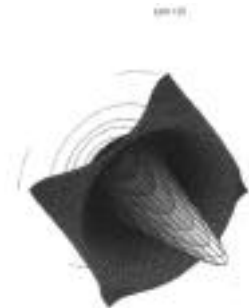


图13-43 旋转后的钟形表面图

MATLAB使用命令`slice`来以图形的方式研究三变量函数。该命令绘制三维表面图，并且图中各点的颜色与这些点的函数值是一致的。

#### 命令集145 三维空间的部分图

```
slice(V,xs,ys,
      zs,nx)
```

绘制矩阵 $V$ 定义三变量函数的部分图。矩阵 $V$ 本身是一个 $n \times$ 层的集合，对由带有三个参数的`meshgrid`得到的三个矩阵进行求值。向量 $xs,ys$ 和 $zs$ 规定要绘制的部分图形。

```
slice(x,y,z,V,xs
      ,ys,zs)
```

以矩阵 $V$ 确定的颜色绘制三维平面。`slice`的旧语法仍然可用；这里是它的一个延伸。向量 $x,y$ 和 $z$ 用作坐标轴。参数 $xs,ys$ 和 $zs$ 确定绘图平面。

#### 例13.25

下面来观察函数 $f(x,y,z)=x^2+y^2+z^2$ 在立方体中的形状：

$[-1 \ 1] \times [-1 \ 1] \times [-1 \ 1]$

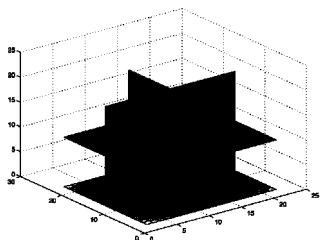
首先要用`meshgrid`定义一个三维网格并对函数求值：

```
[X,Y,Z] = meshgrid(-1:0.1:1,-1:0.1:1,-1:0.1:1);
V = X.^2 + Y.^2 + Z.^2;
```

注意，在 $21^3$ 个点上对函数进行求值。现在要决定绘制与坐标轴平行的那一部分图形。比如：向量 $(1 \ 3 \ 21)$ 表示我们要绘制1、3和21部分。命令：

```
slice(V,[11],[11],[1 11]);
```

得到如图13-44的结果，它的部分图是由平面 $x=11, y=11, z=1$ 和 $z=11$ 定义的。

图13-44 用命令`slice`得到的三变量函数图解

正如所预想的，函数值沿球形恒定。这在彩色图中比在黑白图中效果更加明显。

### 13.6 颜色控制

在MATLAB中，用户可以控制颜色和三维表面图的光照效果。

命令`shading`可以用来配置表面图的绘制方式。表面图可以在有网格或无网格情况下绘制，也可以在平面设置当前图形的阴影，或设置内插阴影。

#### 命令集146 表面图属性

<code>shading type</code>	用下列属性重新绘制表面图：
<code>faceted</code>	设置表面阴影；这是缺省值。
<code>interp</code>	设置内插阴影。
<code>flat</code>	对平面设置当前图形的阴影。

在调色板一节的图P-3的Riemann表面图中给出了两种不同的阴影类型。

#### 例13.26

在例13.20中，绘制了一个网格可见的钟形表面图。用下面的命令可以在图形窗口中得到用内插值颜色着色的图13-33。

```
shading interp
```

得到如图13-45所示的结果。

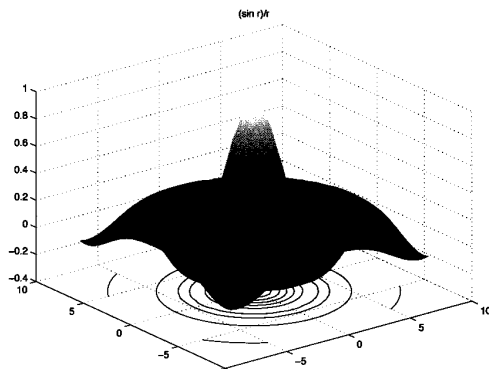


图13-45 用内插值颜色着色的钟形图

MATLAB使用色图来绘制表面图形。色图是一个 $m \times 3$ 的矩阵，其中，行代表颜色，第1列给出红颜色的数量，第2列给出绿颜色的数量，第3列给出蓝颜色的数量，因此，该色图能给出 $m$ 种颜色。

表面图的颜色由色图的下标来确定。下标通常与表面图的最大值和最小值有关。命令`colormap`常用于设置MATLAB使用的色图。

#### 命令集147 色图

<code>colordef definition</code>	改变图形的颜色设置。 <b>definition</b> 的有效值为：
----------------------------------	-------------------------------------

	white	采用亮度背景和黑色坐标轴。
	black	采用黑色背景和亮度坐标轴。
	none	采用MATLAB的颜色设置。
colormap(Cm)		设定当前颜色表为 Cm。矩阵 Cm 可以是 MATLAB 自身的色表，或用户定义的色表。
colormap		返回 $m \times 3$ 矩阵的当前图形色表。
colorbar		在当前窗口中绘制竖直方向的条形颜色刻度。参阅 <a href="#">colorbar</a> 。
colorbar('horiz')		在当前窗口中绘制水平方向的条形颜色刻度。

为了使用预先定义の色图，可以使用命令 `colormap(winter(m))`，其中  $m$  为色图中颜色的数目。MATLAB 中共有 17 个预定义の色图(见表 13-2 和图 P-1)。

表13-2 MATLAB中的色图

colorcube	返回RGB调色板中的均匀间距的颜色
lines	按照坐标轴的颜色顺序返回色图
autum	返回红色和黄色图
spring	返回红紫色和黄色色图
summer	返回绿色和黄色图
winter	返回蓝色和绿色图
gray	返回线形灰色刻度
hsv	返回从红到蓝再到红的深度颜色
hot	返回由黑到红到黄和白的混合的暖色
cool	返回青色和深红色的冷色
bone	返回带蓝色的灰色刻度
copper	返回铜色刻度
pink	返回粉红色的变化
flag	返回英制和美制标志的颜色，红色、白色、蓝色和黑色循环重复
prism	循环重复返回六种颜色：红色、橘黄色、黄色、绿色、蓝色和紫罗兰色
jet	返回一个交替的从红到蓝的色彩模式颜色表
white	返回一个全白的色图

例13.27

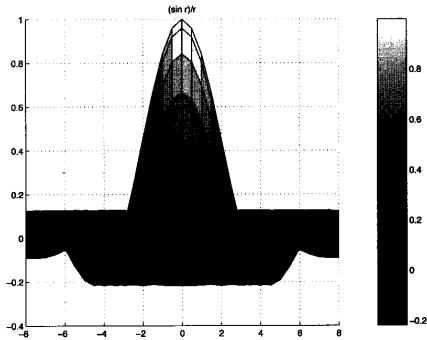


图13-46 带有色图条的钟形曲线图

假设图形窗口中已有了图 13-37。命令：

colorbar

给出如图13-46所示的结果。

此外，还有几个对色图进行处理的命令。

#### 命令集148 颜色处理

<code>rgb2hsv(Cm)</code>	返回 $m \times 3$ 的矩阵 $Cm$ 中的rgb色图的色彩模型图。该色彩模型图包括与rgb图同样的颜色，但是颜色更深。
<code>hsv2rgb(Cm)</code>	返回色彩模型色图 $Cm$ 的 $m \times 3$ 的rbg色图。
<code>rgbplot(Cm)</code>	绘制色图 $Cm$ 各列的图形。线条分别由红色、绿色和蓝色绘制。
<code>caxis(v)</code>	设置色图的当前区间为 $v=[v_{\min}, v_{\max}]$ ，其中 $v_{\min}$ 和 $v_{\max}$ 代表色图的较低和较高下标边界。
<code>caxis</code>	返回色图的当前区间。
<code>caxis('auto')</code>	恢复刻度为MATLAB自动标记的刻度。
<code>spinmap(t,s)</code>	用 $s$ 步旋转色图 $t$ 秒。如果 $s$ 不确定，则令 $s=2$ ，如果 $t$ 不给出，则令 $t=3$ 对色图做永久的旋转。
<code>spinmap(inf)</code>	
<code>brighten(s)</code>	如果 $0 < s < 1$ ，则加亮当前色图。如果 $-1 < s < 0$ ，则加暗色图，重画图形。
<code>nt=brighten(Cm,s)</code>	返回变亮或变暗的 $Cm$ 色图，但是不重画当前图形。
<code>contrast(Cm,m)</code>	返回从颜色表 $Cm$ 中长度为 $m$ ，并且增加黑白监视器对比度的颜色表。如果省略 $m$ ，将返回与 $Cm$ 同样大小的色表。
<code>whitebg</code>	在黑白之间切换图形窗口的背景颜色。如果有必要，刻度颜色等将改变为可见的。
<code>whitebg(str)</code>	根据 $str$ 设置背景颜色，可以是字符串(参见表13-1)或rgb向量。
<code>graymon</code>	对黑白监视器设置参数。

#### 例13.28

用命令`rgbplot`对色度模型色图做一下研究：

```
clf;
rgbplot(hsv);
title('rgbplot of hsv');
axis([0 70 -0.1 1.1]);
```

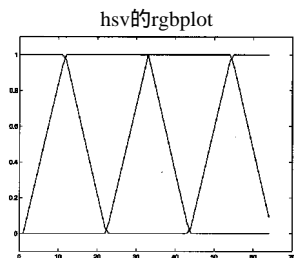


图13-47 一个色度模型色图上的rgbplot图形

得到图 13-47 的结果。

## 13.7 图形窗口的硬拷贝

假设已安装了必要的硬件和软件，就可以从 MATLAB 的图形窗口中得到图形的一个硬拷贝。命令 `print` 可用于硬拷贝到文件中或送打印机，应用于当前图形。在 PC 或 Macintosh 系统中，文件菜单下的打印选项是打印图形的最简单方式。

### 命令集 149 打印硬拷贝图形

<code>print</code>	将当前图形窗口的一个高分辨率拷贝发送至打印机。这要求将 <code>print</code> 命令分配给打印机，输入 <code>help print</code> 可获得更多信息。
<code>print filename</code>	将当前图形窗口的拷贝保存到文件 <code>filename</code> 中。
<code>print filename -deps</code>	将副本以 eps 格式，即压缩的附录，保存到文件 <code>filename</code> 中。 键入 <code>help print</code> 会有更多的选项。
<code>[str,dev]=printopt</code>	给定 <code>print</code> 使用的命令字符串和设备。这有可能要修改这个 M 文件；输入 <code>help printop</code> 可获得更多信息。

如果打印机在安装时并未指定给 MATLAB，那么，可以先将图形存成一个文件，然后使用系统命令将该文件发送至打印机。比如：在 SUN 公司的 Solaris 2.x 网络操作系统上的一种类型为：

```
!lp -dskrivarnamn filename.ps
```

命令 `orient` 可以用来设定硬拷贝图形的打印方向。如果要改变方向，该命令设定的方向将优先于用 `print` 命令的设定方向。

### 命令集 150 纸张方向控制

<code>orient landscape</code>	设定下一次打印的方向为 <code>landScape</code> ，即：水平方向。
<code>orient portrait</code>	设定下一次打印的方向为 <code>portrait</code> ，即：竖直方向。
<code>orient tall</code>	设定下一次打印的方向为竖直方向，并将刻度设为全部纸面。
<code>orient</code>	将当前方向返回到一个字符串中。

#### 例 13.29

本书包含许多来自 MATLAB 的图形。这些图形都是由各种 MATLAB 命令创建的，并且可以用类似下面的语句制作硬拷贝：

```
print -deps fig10
```

这些图形直接引入到本书中。这是 MATLAB 结合其他程序编程的例子。

`orient` 命令可以设置当前图形的属性。图形对象的使用（参见第 14 章）给出了硬拷贝图形属性的更具体的控制。纸张的大小、纸张中的位置、背景颜色和一些其他属性都可以设定。

## 13.8 声音

MATLAB 可以使用 `sound` 命令制作声音向量。

## 命令集151 声音

<code>sound(y)</code>	将向量 <code>y</code> 传送给扬声器。向量确定了最大振幅。
<code>sound(y,f)</code>	与上面的命令相似，而且还设定采样频率为 $f$ Hz。该命令不能用于SUN公司的SPARC工作站。
<code>soundsc(x, f,slim)</code>	采用与 <code>sound</code> 相同的方式播放向量 <code>x</code> ，除了 <code>soundsc</code> 给出的声音向量，声音可以尽可能地大。如果给出 $f$ ， $f$ 就表示采样频率。这里 <code>slim</code> 设定满音量范围，缺省值为 $[\min(x) \max(x)]$ 。

## 例13.30

(a) 正弦波听起来是这样的：

```
x=sin(linspace(0, 10000, 10000)); % 一个纯正弦波。
sound(x); % 产生声音。
```

(b) 可以用`load`命令载入几个预定义的声音。这里，就试试其中的两个。

```
load train; % 装入产生火车汽笛的声音数据。
whos; % 显示变量y，Fs：向量y：表示创建的信号；标量Fs：表示以Hz为单位%的频率。

sound(y); % 产生声音。
load chirp; % 装入产生鸟儿唧唧喳喳声音的数据。
sound(y); % 产生新的声音。
```

在某些系统中还有一些附加的声音命令；用`help sound`可以了解这些特殊的系统。

SUN公司的SPARC工作站使用存成mu-law编码数据的声音向量作为声音文件。

## 命令集152 SPARC工作站上的声音命令

<code>auread(fstr)</code>	从文件 <code>fstr</code> 中读取并返回一个向量。
<code>auwrite(sv,fstr)</code>	将向量 <code>sv</code> 写入Sun的音频文件 <code>fstr</code> 中。
<code>lin2mu(sv)</code>	将线性声音向量 <code>sv</code> 转化为mu-law编码的向量。
<code>mu2lin(sv)</code>	将mu-law编码的向量 <code>sv</code> 转化为线性声音向量。

微软Windows操作系统中使用的声音文件为.wav格式。

## 命令集153 专用于微软Windows操作系统的声音命令

<code>wavread(fstr)</code>	在文件 <code>fstr</code> 中返回采样数据。用 <code>help wavread</code> 可获得更多信息。
<code>wavwrite(sv,f,fstr)</code>	以采样频率 $f$ 将采样声音向量 <code>sv</code> 写入文件 <code>fstr</code> 中。