第 5 章

创建新图表类型

虽然 MATLAB 定制了很多图表类型，但明显不一定够用，很多图表类型它还没有。这时候需要我们自己动手创建新的图表类型。创建新的图表类型，可以用低级函数创建最基本的点、线、面等图形元素，用它们从零开始创建新图表；也可以在 MATLAB 图表的基础上修改创建新图表；或者用已有的 MATLAB 图表进行组合，生成新图表。

5.1 用低级函数创建基本图元

前面几章使用 plot、bar、surf 等函数创建了 MATLAB 图表，这些函数称为高级绘图函数。本节介绍创建基本图形元素的函数，即低级函数，包括 line、rectangle、patch、surface、image、text 等。使用低级函数绘图的效率更高。基本的图形元素包括直线、多义线、圆、多边形、曲面、图像和文本等。

5.1.1 创建点

MATLAB 中的点常常用点标记表示，实际上是一个组合图形。如圆、三角形、棱形、五角星等，有填充面，有边线。也可以用球面表示三维点。可以用低级函数自定义点，也可以直接用高级函数绘制点。

MATLAB 中可以用 plot 函数和 plot3 函数绘制二维点和三维点。下面的代码可在三维坐标系中绘制一个二维点和一个三维点，标记分别为“*”和“o”。在二维坐标系中创建一个实心圆表示的点，修改实心圆的填充色和边线的颜色：

```
tiledlayout(2,2);

nexttile
x = 1;
y = 2;
plot(x, y,'*');                          % 用 plot 函数绘制二维点
hold on
z = 3;
plot3(x,y,z,'o');                        % 用 plot3 函数绘制三维点
view(3)
```

```
hold off

nexttile
h = plot(1,2,'o');                          % 绘制点
h.LineWidth = 2;                            % 设置点的属性,点标记的边线宽度
h.MarkerSize = 10;                          % 点标记的大小
h.MarkerEdgeColor = 'r';                    % 点标记边线的颜色
h.MarkerFaceColor = [0,0.8,0];              % 点标记的填充色
```

运行代码,生成图 5-1。

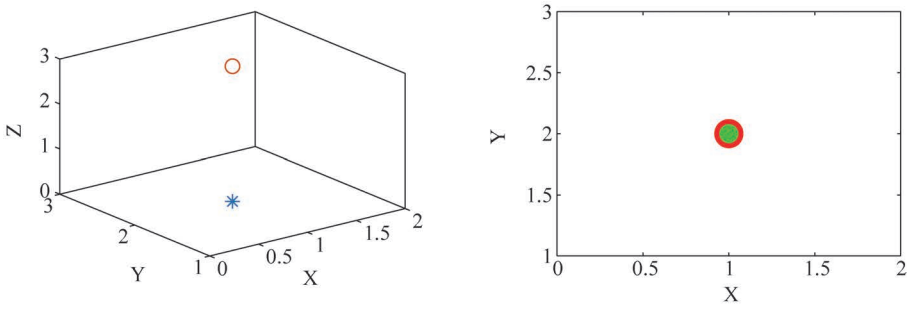

图 5-1　生成点

也可以用 scatter 函数和 scatter3 函数创建二维点和三维点。5.2.1 节会介绍用球面表示三维点。

5.1.2　创建线段、多义线和曲线

MATLAB 中的线段和多义线用 Line 对象表示。用 line 函数创建 Line 对象。按照直线逼近的思路,还可以用该函数创建曲线。线段的属性包括线型、颜色、线宽、点标记等,可参照 4.1 节进行设置。下面结合几个例子介绍 line 函数的使用。

首先创建一条线段,指定起点和终点,设置颜色、线型和线宽。再创建一条多义线,指定多义线各顶点的坐标,设置多义线的颜色、线宽、标记类型、标记面的颜色和边线的颜色,以及标记的大小:

```
tiledlayout(2,2);

nexttile
X = [1 12];
Y = [3 9];
line(X,Y,'Color','r','LineStyle', '-.','LineWidth',1.5)   % 绘制线段
box on
grid on

nexttile
X2 = [1 9 20 28];
Y2 = [1 25 10 32];
 % 绘制折线
line(X2,Y2,'Color','b','LineWidth',1.5 ,'Marker','d',…
    'MarkerFaceColor','y','MarkerEdgeColor','r','MarkerSize',9)
```

```
box on
grid on
```

运行代码，生成图 5-2。

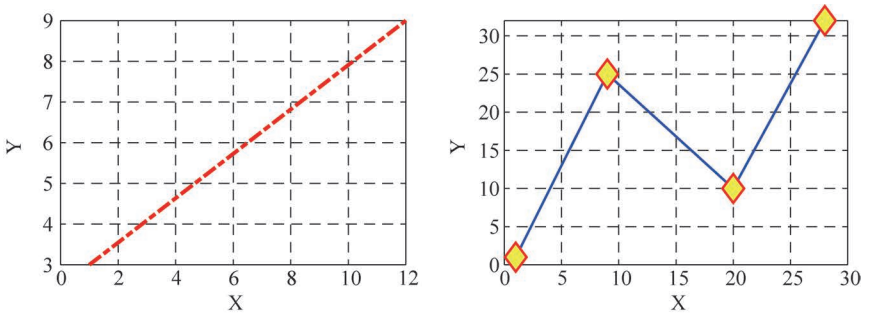

图 5-2　绘制线段

如果 line 函数中的位置参数为矩阵，则可以同时生成多条线段或多义线。用下面的代码在两个坐标系中分别绘制复合线形图，再用颜色和线型区分各自坐标系中的序列：

```
tiledlayout(2,2);
X3 = [1 1 1;10 10 10;20 20 20];                    % 矩阵数据
Y3 = [2 5 8;12 15 18;22 25 28];

ax1 = nexttile;
colororder(ax1,'meadow')                           % 定义颜色序列
line(X3,Y3,'LineWidth',1.5)                        % 绘制复合线形图
box on
grid on

ax2 = nexttile;
linestyleorder(ax2,["-","--",":"])                 % 定义线型序列
line(X3,Y3,'LineWidth',1.5)                        % 绘制复合线形图
box on
grid on
```

运行代码，生成图 5-3。

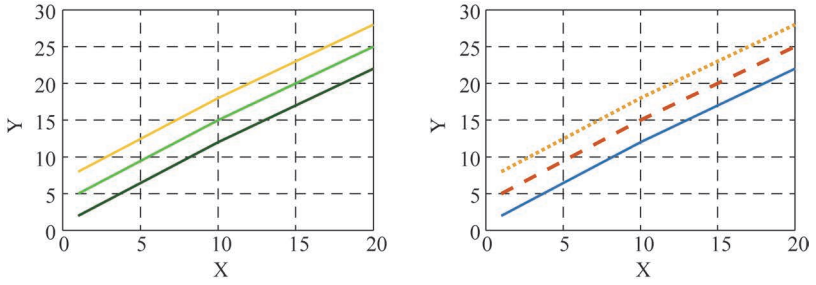

图 5-3　绘制多条多义线

下面用多条线段逼近余弦曲线，再用下面的代码将[0，2π]范围内的余弦曲线用包含 11 个顶点的多义线表示：

```
tiledlayout(2,2);
```

```
ax1 = nexttile;
t = 0:pi/5:2 * pi;
line(t,cos(t),'LineWidth',1.5)                    % 用 10 条线段逼近余弦曲线
box on
grid on

ax2 = nexttile;
t = 0:pi/20:2 * pi;
line(t,cos(t),'LineWidth',1.5)                    % 用 40 条线段逼近余弦曲线
box on
grid on
```

运行代码，生成图 5-4。左图和右图分别为 10 等分和 40 等分区间生成的余弦曲线，右图平滑了很多。

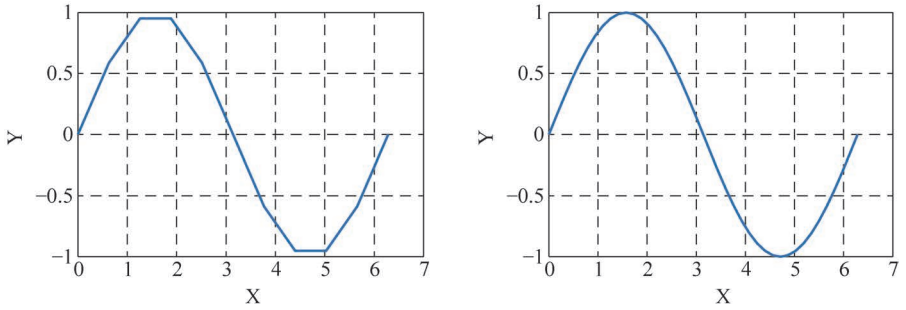

图 5-4　用精度更高的多义线逼近余弦曲线

用 line 函数还可绘制三维线段和曲线。用下面的代码在两个坐标系中分别绘制三维多义线和参数函数定义的三维曲线：

```
tiledlayout(2,2);

ax1 = nexttile;
x = rand(4,3,3) * 10;
colororder(ax1,'gem')
line(x(:,:,1),x(:,:,2),x(:,:,3),'LineWidth',1.5)  % 三维多义线
view(3)
box on
grid on

ax2 = nexttile;
t = linspace(0,10 * pi,200);
x = sin(t);
y = cos(t);
z = t;
line(x,y,z,'LineWidth',1.5)                        % 三维曲线
view(3)
box on
grid on
```

运行代码，生成图 5-5。

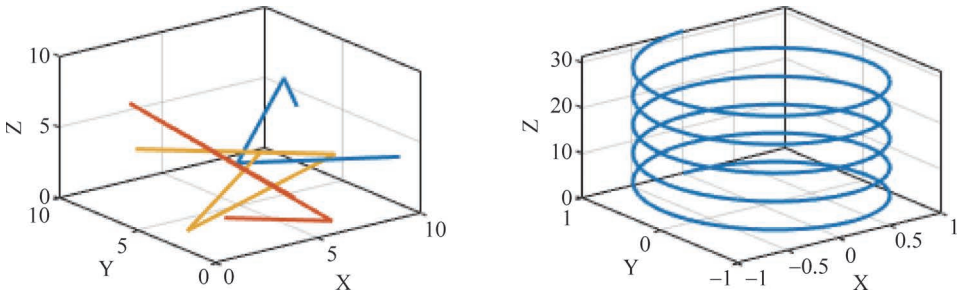

图 5-5　绘制三维多义线和三维曲线

5.1.3　创建矩形、圆角矩形、椭圆、圆及对应的区域图形

MATLAB 中，矩形、圆角矩形、椭圆、圆及对应的区域图形都用 Rectangle 对象表示。用 rectangle 函数可以创建 Rectangle 对象。

矩形或矩形区域的属性包括线型（LineStyle）、线宽（LineWidth）、内部颜色（ForeColor）、边线颜色（EdgeColor）等，可参照 5.1 节进行设置。下面结合两个实例进行介绍。

用下面的代码在同一个坐标系中绘制线形图形矩形、圆角矩形、椭圆和圆，使用不同的线型和线宽。在另一个坐标系中绘制圆角矩形区域和圆形区域：

```
tiledlayout(2,2);

ax1 = nexttile;
%绘制矩形
rectangle('Position',[1,1,20,10], 'LineWidth',3,'EdgeColor','m');
%绘制椭圆形
rectangle('Position',[5,3,10,15], 'Curvature',[1 1],'EdgeColor','g');
%绘制圆
rectangle('Position',[5,3,10,10], 'Curvature',[1 1], 'LineWidth',3,'EdgeColor','b');
%绘制圆角矩形
rectangle('Position',[4,5,12,8], 'Curvature',.4,'LineStyle', '-- ','EdgeColor','c');
axis equal
box on

ax2 = nexttile;
%绘制圆角矩形区域
rectangle('Position',[3,7,20,8], 'Curvature',[.3 .4], 'FaceColor', 'g');
%绘制圆形区域
rectangle('Position',[5,3,10,10], 'Curvature',[1 1], 'LineWidth',3, 'FaceColor', 'r');
axis equal
box on
```

运行代码，生成图 5-6。

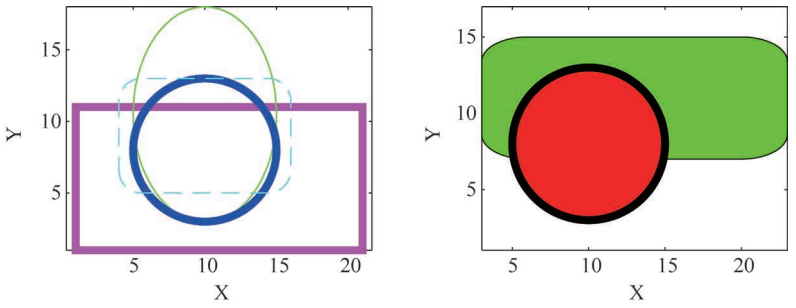

图 5-6　绘制矩形、圆角矩形、椭圆、圆和对应区域

5.1.4　创建面片

4.2.7 节详细介绍了 MATLAB 中面片的着色，这里不再赘述。下面的代码用面片生成颜色渐变的曲线：

```
x = linspace(1,10,15);
y = cos(x);
y(end) = NaN;
c = y;                                          % 用 y 值着色
patch(x,y,c,'EdgeColor','interp','LineWidth',2,…
    'Marker','o','MarkerFaceColor','flat');
colormap jet
colorbar
```

运行代码，生成图 5-7。

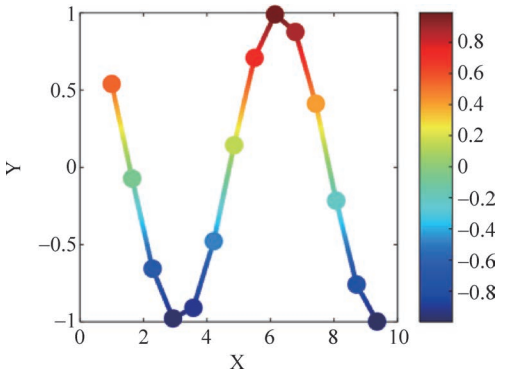

图 5-7　渐变色曲线

5.1.5　创建曲面

MATLAB 中，Surface 对象是由四边形小面组成的规则网格模型。网格模型包括线框模型、刻面模型和曲面模型等。使用 surface 函数可创建 Surface 对象。

下面用 surface 函数绘制规则网格曲面，不显示边线，添加光照：

```
[x,y,z] = peaks(30);
h = surface(x,y,z);                             % 绘制曲面
```

```
h.EdgeColor = 'none';                    % 删除网格线
colormap jet                             % 设置颜色查找表为 jet 表
view(3)                                  % 三维视图
camlight                                 % 添加光照
grid on
box on
```

运行代码，生成图 5-8。

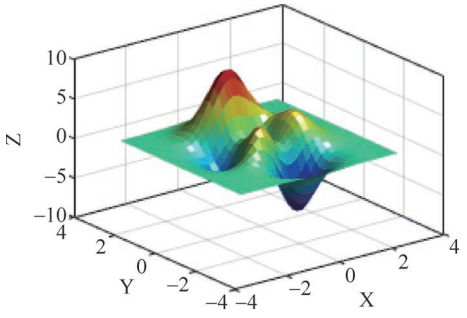

图 5-8　用 surface 函数绘制曲面图

5.1.6　创建图像

用 image 函数处理图像。下面的代码在两个坐标系中创建图像。第一个坐标系中直接用矩阵数据定义颜色创建图像，第二个坐标系中通过导入图像文件创建图像：

```
tiledlayout(2,2);

ax1 = nexttile;
x = [1 4];
y = [6 9];
C = [1 25 50 75; 100 125 150 175; 200 225 250 300];    % 定义矩阵
image(x,y,C)                                           % 用矩阵元素的值着色

ax2 = nexttile;
C = imread('d:\pic.jpg');                              % 读取图片文件，获取像素颜色矩阵
image(C)                                               % 用矩阵元素的值着色
```

运行代码，生成图 5-9。

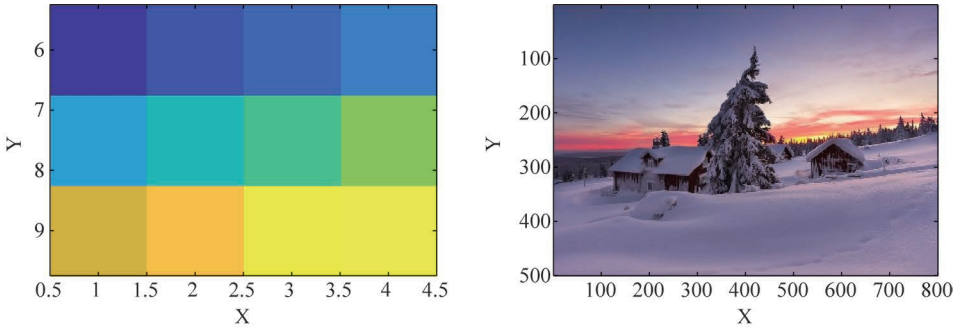

图 5-9　创建图像

5.1.7　添加文本

文本用 Text 对象表示，该对象由 text 函数创建。text 函数是创建文本图形对象的低级函数，它将文本字符串置于指定的位置。使用 text 函数还可以在三维图形中添加文本标注，利用参数-值匹配对的方式在创建文本标注时设置文本属性，如字体名称、字体大小等。也可以用 text 函数创建文本时返回句柄，用句柄对象点引用的方式设置文本属性。

用下面的代码在两个坐标系中分别创建二维线形图和三维曲面图，并分别添加文本标注：

```
tiledlayout(2,2);
ax1 = nexttile;
ax1.XLim = [0 2 * pi];
ax1.YLim = [ - 1 4];
x = 0:pi/20:2 * pi;
y = sin(x);
line(x,y,'LineWidth',1.5)
text(pi,0,'\leftarrow sin(\pi)')                    % 添加文本标注
text(1,2, '添加文本 1');
text(1,3, '添加文本 2', 'FontSize',20, 'FontAngle', 'Italic');
box on

nexttile;
h3 = surfl(peaks);
h3.EdgeColor = 'none';
shading interp
text(25,37,8,'Welcome to 8848')                    % 在三维图中添加文本标注
```

运行代码，生成图 5-10。

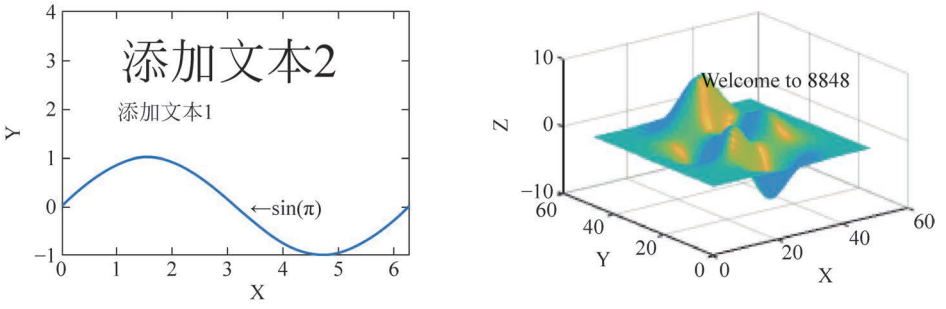

图 5-10　在图中添加文本标注

5.2　用基本图元创建新图表

上一节介绍了点、线、面和文本等基本图形元素的创建，有了基本图形元素后，就可以用它们创建新的图表。本节结合若干实例进行介绍。

5.2.1 用球面创建三维散点图

MATLAB 中可以方便地使用 plot3 函数和 scatter3 函数绘制三维点，但美中不足的是，它们绘制的三维点看起来更像二维点，缺少三维效果。MATLAB 中可以用 sphere 函数绘制单元球面，本例使用该函数生成若干单位球面并进行平移、缩放和着色，获得更美观的三维散点图效果。本例实现了简单三维散点图、复合三维散点图，以及基于球面大小和颜色的度量表示：

```
tiledlayout(2,2);

nexttile
x = rand(20,1) * 20;                                    % 20 组均匀分布的随机数，值介于 0～20
y = rand(20,1) * 20;
z = rand(20,1) * 20;
 % 用 20 组数据绘制球面表示的三维散点
for i = 1:20
  [X,Y,Z] = sphere;                                     % 单位球面上各节点的坐标
  X = X + x(i,1);                                       % 平移
  Y = Y + y(i,1);
  Z = Z + z(i,1);
  surface(X,Y,Z,'EdgeColor','none','FaceColor','y')     % 绘制球面
end
axis([0 20 0 20 0 20])
axis equal
set(gca,'Color',[0.9 0.9 0.9])
xlabel('X')
ylabel('Y')
zlabel('Z')
grid on
box on
view(3)
camlight
title("Simple Plot")

nexttile
x = rand(20,2) * 20;                                    % 两列随机数
y = rand(20,2) * 20;
z = rand(20,2) * 20;
c = ['g' 'y'];                                          % 颜色数组，绿色和黄色
 % 用两列各组数据绘制两种颜色的球面散点图
for i = 1:20
    for j = 1:2
      [X,Y,Z] = sphere;
      X = X + x(i,j);                                   % 平移
      Y = Y + y(i,j);
      Z = Z + z(i,j);
      surface(X,Y,Z,'EdgeColor','none','FaceColor',c(j))  % 绘制球面，两种颜色
    end
end
axis([0 20 0 20 0 20])
axis equal
```

```
set(gca,'Color',[0.9 0.9 0.9])
xlabel('X')
ylabel('Y')
zlabel('Z')
grid on
box on
view(3)
camlight
title("Complex Plot")

nexttile
x = rand(20,1) * 20;                                    % 球面位置
y = rand(20,1) * 20;
z = rand(20,1) * 20;
s = rand(20,1) * 2;                                     % 定义球面的半径
 % 绘制各球面
for i = 1:20
  [X,Y,Z] = sphere;                                     % 单位球面各节点的坐标
  X = s(i,1) * X + x(i,1);                              % 用 s 缩放,用 x、y、z 平移
  Y = s(i,1) * Y + y(i,1);
  Z = s(i,1) * Z + z(i,1);
  surface(X,Y,Z,'EdgeColor','none','FaceColor','r')     % 绘制
end
axis([0 20 0 20 0 20])
axis equal
set(gca,'Color',[0.9 0.9 0.9])
xlabel('X')
ylabel('Y')
zlabel('Z')
grid on
box on
view(3)
camlight
title("Different Size")

nexttile
x = rand(20,1) * 20;                                    % 定义每个球面的位置
y = rand(20,1) * 20;
z = rand(20,1) * 20;
cr = rand(20,3);                                        % 定义每个球面的颜色
 % 绘制各球面
for i = 1:20
  [X,Y,Z] = sphere;                                     % 单位球面各节点的坐标
  X = X + x(i,1);                                       % 平移
  Y = Y + y(i,1);
  Z = Z + z(i,1);
  surface(X,Y,Z,'EdgeColor','none','FaceColor',cr(i,1:3))  % 用指定颜色绘制球面
end
axis([0 20 0 20 0 20])
axis equal
set(gca,'Color',[0.9 0.9 0.9])
xlabel('X')
ylabel('Y')
zlabel('Z')
```

```
grid on
box on
view(3)
camlight
title("Different Color")
```

运行代码，生成图 5-11。图 5-11(a)为简单三维散点图，图 5-11(b)为区分颜色的复合三维散点图，图 5-11(c)用一个变量的值定义球面的大小，图 5-11(d)用一个变量的值定义球面的颜色。

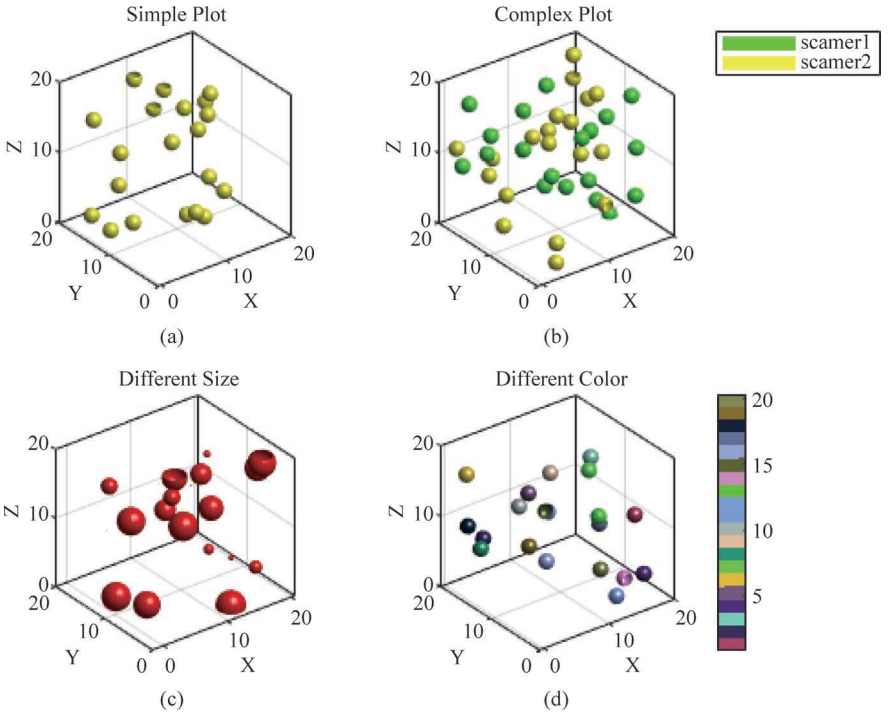

图 5-11　用球面实现三维散点图

5.2.2　创建三元散点图

三元图经常用于表示三组数据的占比关系，如图 5-12 所示，3 个坐标轴围成一个等边三角形，它们的取值范围都是 0～1，即 0%～100%。坐标系中任一点的坐标值，可以通过该点绘制 3 个轴的平行线读取，3 个坐标值的和应该等于 100%。

用下面的代码绘制三元坐标系。先用 fill 函数绘制等边三角形区域，再绘制网格线。注意，网格线的起点和终点需要进行直角坐标系到三元坐标系的转换。最后添加刻度标签和坐标轴标签：

```
% 坐标轴和等边三角形绘图区
h = fill([0 1 0.5 0],[0 0 0.866 0],'w','linewidth',2);
% 网格线绘制，起点和终点由直角坐标系向三元坐标系转换
d1 = cos(pi/3);
d2 = sin(pi/3);
l = linspace(0,1,11);                                    % 直角坐标系中的值
hold on
for i = 2:length(l) - 1
```

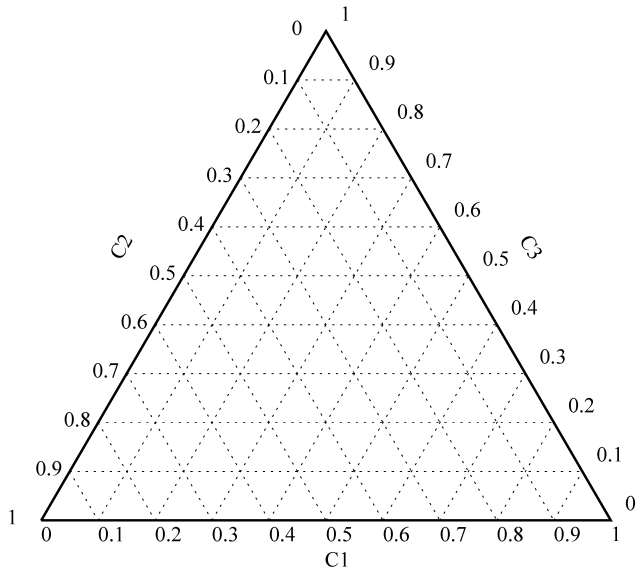

图 5-12　三元坐标系

```
    %斜边上的点有转换,底边上的点没有转换
    plot([l(i)*d1 1-l(i)*d1],[l(i)*d2 l(i)*d2],':k','linewidth',0.25);
    plot([l(i) l(i)+(1-l(i))*d1],[0 (1-l(i))*d2],':k','linewidth',0.25);
    plot([(1-l(i))*d1 1-l(i)],[(1-l(i))*d2 0],':k','linewidth',0.25);
end
axis image
axis off
%绘制刻度标签
for i=1:11
     text(l(i),-0.025,num2str(l(i)));
     text(1-l(i)*cos(pi/3)+0.025,l(i)*sin(pi/3)+0.025,num2str(l(i)));
     text(0.5-l(i)*cos(pi/3)-0.06,sin(pi/3)*(1-l(i)),num2str(l(i)));
end
%绘制坐标轴标签
text(0.5,-0.05,'C1','HorizontalAlignment','center');
text(0.15,sqrt(3)/4+0.05,'C2','HorizontalAlignment','center','Rotation',60);
text(0.85,sqrt(3)/4+0.05,'C3','HorizontalAlignment','center','Rotation',-60);
hold off
```

用下面的代码在三元坐标系上绘制散点图。与处理网格线的起点和终点一样，需要将散点的坐标由直角坐标系转换到三元坐标系：

```
%散点数据
load terplot_data.mat
l=length(B);
B(l+1,:)=[1 0 0 6];
B(l+2,:)=[0 1 0 30];
B(l+3,:)=[0 0 1 1];
d=0.29./sqrt(B(:,4));
c1=B(:,1);c2=B(:,2);c3=B(:,3);

……                                                    %省略部分代码

%散点坐标转换和绘制
```

```
for i = 1:length(c1)
    x = 0.5 - c1(i) * cos(pi/3) + c2(i)/2;
    y = 0.866 - c1(i) * sin(pi/3) - c2(i) * cot(pi/6)/2;
    plot(x,y,'ob','MarkerSize',4);
end
……                                                    % 省略部分代码
```

运行代码，生成图 5-13。

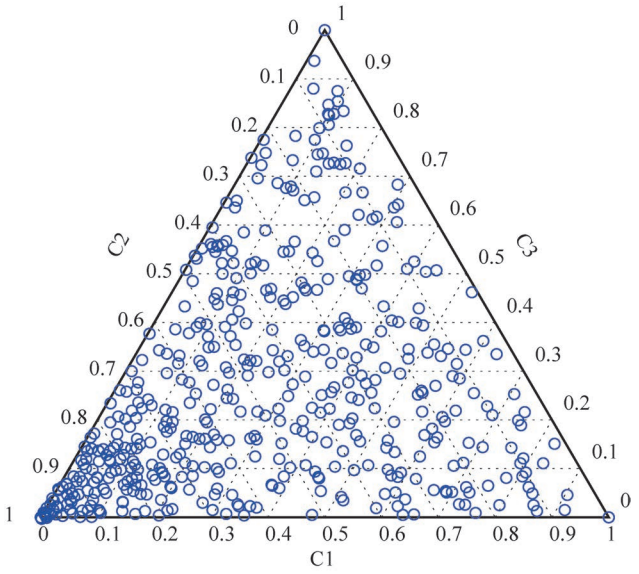

图 5-13　三元散点图

5.2.3　创建三元色谱图

三元色谱图的绘制是在三元散点的基础上，用散点数据生成 Delaunay 三角网，并以插值方式用渐变色绘制 Delaunay 三角曲面。

用下面的代码创建三元色谱图：

```
% 绘图数据
load terplot_data.mat
l = length(B);
B(l + 1,:) = [1 0 0 6];
B(l + 2,:) = [0 1 0 30];
B(l + 3,:) = [0 0 1 1];
d = 0.29./sqrt(B(:,4));
c1 = B(:,1);c2 = B(:,2);c3 = B(:,3);

% 绘制三元坐标系
h = fill([0 1 0.5 0],[0 0 0.866 0],'w','linewidth',2);
d1 = cos(pi/3);
d2 = sin(pi/3);
l = linspace(0,1,11);
hold on
axis image
axis off
```

```
for i = 1:number
     text(l(i), - 0.025,num2str(l(i)));
     text(1 - l(i) * cos(pi/3) + 0.025,l(i) * sin(pi/3) + 0.025,num2str(l(i)));
     text(0.5 - l(i) * cos(pi/3) - 0.06,sin(pi/3) * (1 - l(i)),num2str(l(i)));
end

% 计算数据点在三元坐标系中的新坐标
x = 0.5 - c1 * cos(pi/3) + c2/2;
y = 0.866 - c1 * sin(pi/3) - c2 * cot(pi/6)/2;

% 根据散点计算 Delaunay 三角网关键矩阵
tri = delaunay(x,y);
% 用 trisurf 函数绘制 Delaunay 三角曲面
trisurf(tri,x,y,d);
% 插值着色
shading interp

% 在色谱图上绘制网格线
zmax = max(d);
for i = 2:length(l) - 1
     plot3([l(i) * d1 1 - l(i) * d1],[l(i) * d2 l(i) * d2],[zmax zmax] * 1.1,':k','linewidth',0.25);
     plot3([l(i) l(i) + (1 - l(i)) * d1],[0 (1 - l(i)) * d2],[zmax zmax] * 1.1,':k','linewidth',0.25);
     plot3([(1 - l(i)) * d1 1 - l(i)],[(1 - l(i)) * d2 0],[zmax zmax] * 1.1,':k','linewidth',0.25);
end
plot([0 1 0.5 0],[0 0 sqrt(3)/2 0],'k','linewidth',1)

% 绘制坐标轴标签
text(0.5, - 0.05,'C1','HorizontalAlignment','center');
text(0.15,sqrt(3)/4 + 0.05,'C2','HorizontalAlignment','center','Rotation',60);
text(0.85,sqrt(3)/4 + 0.05,'C3','HorizontalAlignment','center','Rotation', - 60);

hold off                                          % 取消叠加绘图
colorbar                                          % 色条
```

运行代码，生成图 5-14。

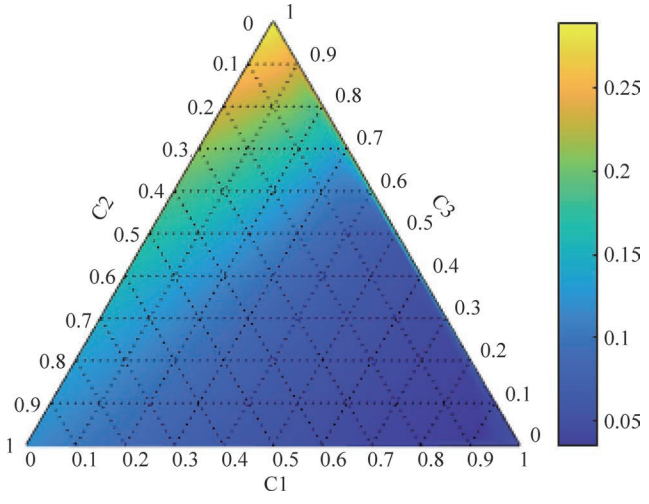

图 5-14　三元色谱图

5.3 修改已有图表创建新图表

也可以在已有 MATLAB 图表的基础上重新渲染图形元素或使用新的图形元素进行表现，从而创建新的图表。例如本节将要介绍的将二维柱状图中的柱形替换为三角形，将三维柱状图中的柱体替换为圆锥或圆柱等。

5.3.1 颜色渐变填充柱状图中的柱形

MATLAB 中用 bar 函数创建的柱状图柱形是单色的，没有太多的变化。本例在 bar 函数创建的柱状图的基础上对柱形进行修改，即隐藏图表中原有的柱形，在原来的位置用 patch 函数创建面片，重新绘制柱形。Patch 对象是可以渐变着色的，从而实现对柱状图中的柱形渐变着色的效果。

用下面的代码按照上面的思路对柱状图中的柱形进行水平渐变着色：

```
data = [1 5 3 7 6];                                    % 绘图数据
b = bar(data);                                         % 用 bar 函数绘柱状图
b(1).FaceColor = 'none';                               % 隐藏柱形的面
b(1).EdgeColor = 'none';                               % 隐藏柱形的边
for i = 1:length(data)                                 % 对于每个柱形
   x = b(1).XData(i);                                  % 柱形位置,底边中心 x 坐标
   w = b(1).BarWidth;                                  % 柱形的宽度
   xb = x - w/2;                                       % 柱形左边 x 坐标
   xe = x + w/2;                                       % 柱形右边 x 坐标
   h = b(1).YData(i);                                  % 柱形的高
   vert = [xb 0;x 0;xe 0;xe h;x h;xb h];               % 面片的顶点坐标数组,顶点从前向后编号
   face = [1 2 3 4 5 6];                               % 顶点围成面,编号逆时针方向
   vc = [0.42 0.75 0.75;1 1 1;0.42 0.75 0.75;0.42 0.75 0.75;1 1 1;0.42 0.75 0.75];   % 顶点的颜色
   % 创建面片,即新的柱形,渐变填充
   p = patch('Faces',face,'Vertices',vert,'FaceVertexCData',vc,'FaceColor','interp');
   p.EdgeColor = 'none';                               % 无边线
end
ylim([0 max(data) * 1.05])
xlabel('X')
ylabel('Y')
grid on
box on
```

运行代码，生成图 5-15(a)。

上面的代码中，每个柱形用面片重建时用了 6 个顶点，柱形中间上下两点的颜色为白色。下面的代码用 3 种颜色从左到右渐变填充柱状图中的矩形，如图 5-15(b)所示。

```
……
for i = 1:length(data)
   ……                                                 % 省略部分代码
  vc = [0.54 0.61 0.8;0.98 0.91 0.75;0.95 0.75 0.77;0.95 0.75 0.77;0.98 0.91 0.75;0.54 0.61
0.8];
  …                                                    % 省略部分代码
end
……
```

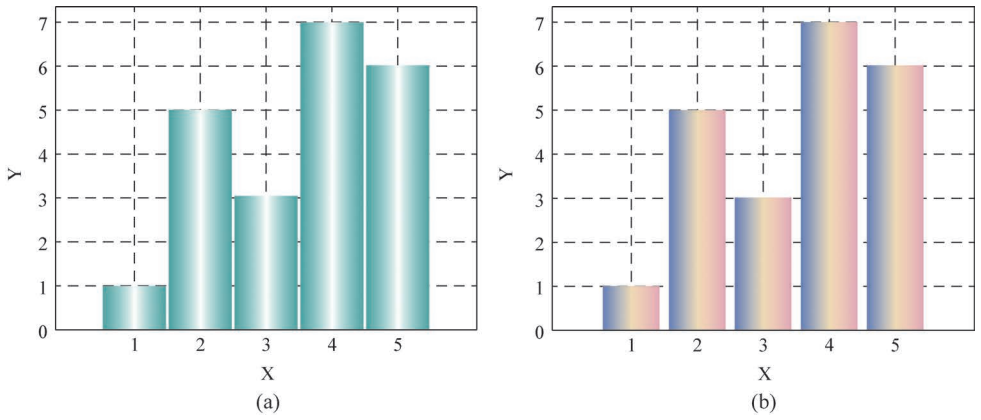

图 5-15　水平渐变填充柱状图

下面的代码中，每个柱形用面片重建时用了矩形的 4 个顶点，左下角和右上角的颜色为白色，右下角和左上角的颜色为蓝色：

```
……                                                    % 省略部分代码
for i = 1:length(data)
   ……                                                 % 省略部分代码
   vert = [xb 0;xe 0;xe h;xb h];                       % 4 个顶点
   face = [1 2 3 4];
   vc = [1 1 1;0.42 0.75 0.75;1 1 1;0.42 0.75 0.75];% 4 个顶点的颜色
   p = patch('Faces',face,'Vertices',vert,'FaceVertexCData',vc,'FaceColor','interp');
   p.EdgeColor = 'none';
end
……                                                    % 省略部分代码
```

运行代码，生成图 5-16(a)，即对角颜色渐变的效果。

下面的代码中，每个柱形用面片重建时用了矩形的 4 个顶点，左下角和左上角的颜色为蓝色，右下角和右上角的颜色为白色：

```
……                                                    % 省略部分代码
for i = 1:length(data)
   ……                                                 % 省略部分代码
   vert = [xb 0;xe 0;xe h;xb h];                       % 4 个顶点
   face = [1 2 3 4];
   vc = [0.42 0.75 0.75;1 1 1;1 1 1;0.42 0.75 0.75];% 4 个顶点的颜色
   ……                                                 % 省略部分代码
end
……                                                    % 省略部分代码
```

运行代码，生成图 5-16(b)。

5.3.2　用图片填充柱状图中的柱形

5.3.1 节使用 Patch 对象重建柱状图中的柱形，实现了柱形的颜色渐变填充。本节介绍用图片填充柱状图中的柱形，使用的是第 4 章介绍的纹理映射方法。这里使用 Surface 对象重建柱形。

下面的代码用 Surface 对象重建 MATLAB 柱状图中的柱形，并用图片填充新柱形：

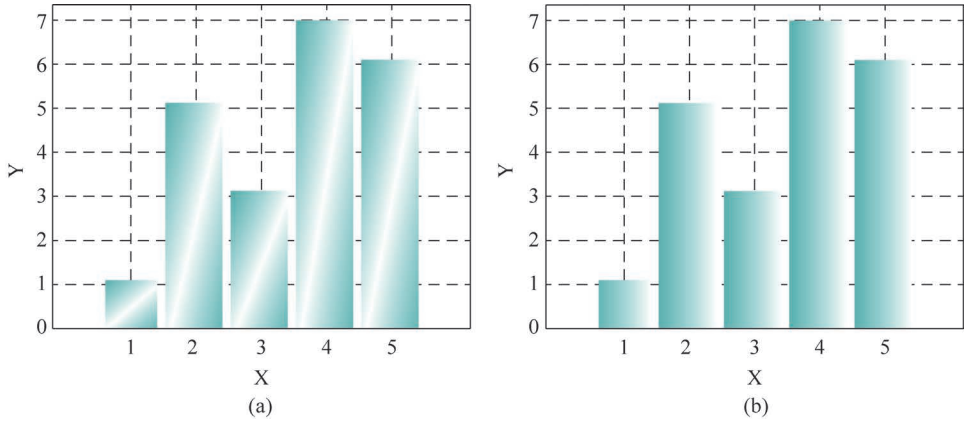

图 5-16　柱状图的更多渐变填充

```
data = [1 5 3 7 6];                                    % 绘图数据
b = bar(data);                                         % 用 bar 函数创建柱状图
b(1).FaceColor = 'none';                               % 隐藏原有柱形
b(1).EdgeColor = 'none';
c = imread('d:\pic.jpg');                              % 从图片文件中读取图像
for i = 1:length(data)                                 % 对于每个柱形
   x = b(1).XData(i);                                  % 柱形底边中心 x 坐标
   w = b(1).BarWidth;                                  % 柱形宽度
   xb = x - w/2;                                       % 柱形左边 x 坐标
   xe = x + w/2;                                       % 柱形右边 x 坐标
   h = b(1).YData(i);                                  % 柱形高度
   x = linspace(xb,xe,20);                             % 准备数据用曲面重建柱形
   y = linspace(0,h,20);
   [X,Y] = meshgrid(x,y);
   Z = zeros(size(X));
    % 用新数据创建曲面,纹理映射着色
   sur = surface(X,Y,Z,'CData',flipud(c),
'FaceColor','texturemap');
   sur.EdgeColor = 'none';                             % 隐藏边线
end
ylim([0 max(data) * 1.05])
xlabel('X')
ylabel('Y')
grid on
box on
```

运行代码，生成图 5-17。也可以为每个柱形填充不同的图片，大家可以自行尝试。

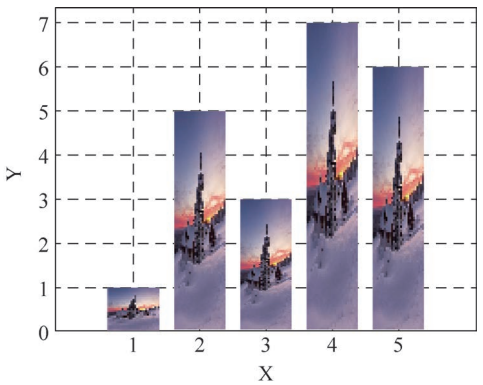

图 5-17　柱状图的图片填充

5.3.3　修改柱状图中的矩形为三角形

5.3.1 节用矩形面片重建了 MATLAB 柱状图中的柱形，这里介绍用三角形面片进行重建。理论上讲，也可以用其他任意图形进行重建，想象空间很大。用三角形面片重建，仍然需要获取原有柱形的位置和大小。

用下面的代码实现 MATLAB 柱状图中柱形的三角形面片重建，并用渐变色进行填充：

```
data = [1 5 3 7 6];
```

```
figure;
set(gcf,'Color',[1 1 1]);
b = bar(data);
b(1).FaceColor = 'none';
b(1).EdgeColor = 'none';
for i = 1:length(data)
  x = b(1).XData(i);
  w = b(1).BarWidth;
  xb = x - w/2;
  xe = x + w/2;
  h = b(1).YData(i);
  vert = [xb 0;xe 0;x h];                          % 3 个顶点
  face = [1 2 3];                                  % 组成面
  vc = [0.94 0.53 0.59;1 1 1;0.94 0.53 0.59];      % 3 个顶点的颜色
  % 创建三角形面片,插值着色
  p = patch('Faces',face,'Vertices',vert,'FaceVertexCData',vc,'FaceColor','interp');
  p.EdgeColor = 'none';
end
title('My Cone Bar')
xlabel('Categorical Variable')
ylabel('Numeric Variable')
grid on
box on
```

运行代码,生成图 5-18。

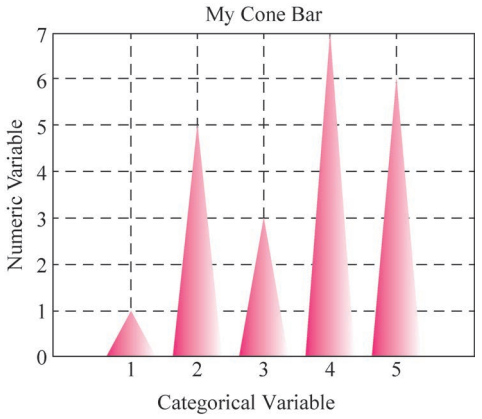

图 5-18　用三角形替换柱状图中的矩形

5.3.4　用高度数据渐变填充柱状图中的柱形

5.3.1 节使用 Patch 对象重建柱状图中的柱形,实现了柱形的颜色渐变填充,当然也可以实现高度方向的颜色渐变填充。本节介绍用 Surface 对象重建柱形,并用 Surface 对象的 CData 属性实现用高度数据渐变填充柱形的效果。

下面的代码用 Surface 对象重建 MATLAB 柱状图中的柱形,并用高度数据实现渐变填充:

```
data = [1 5 3 7 6];
b = bar(data);
for i = 1:length(data)
```

```
    x = b(1).XData(i);
    w = b(1).BarWidth;
    xb = x - w/2;
    xe = x + w/2;
    h = b(1).YData(i);
    x = linspace(xb,xe,20);                    % 用 Surface 对象重建柱形
    y = linspace(0,h,20);
    [X,Y] = meshgrid(x,y);
    Z = zeros(size(X));
    sur = surface(X,Y,Z);
    sur.CData = sur.YData;                     % 用高度数据着色
    sur.FaceColor = 'interp';                  % 插值着色
    sur.EdgeColor = 'none';
end
```

运行代码，生成图 5-19。

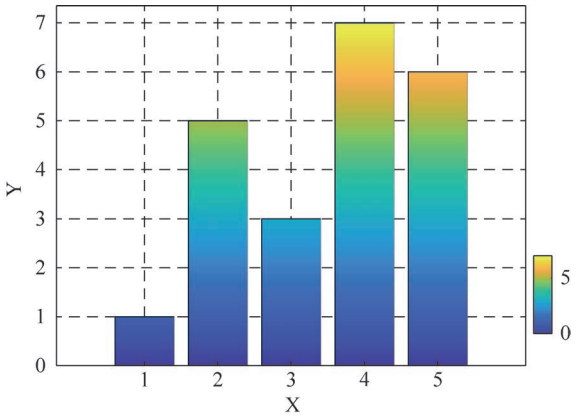

图 5-19　用柱形的高度数据进行渐变填充

5.3.5　替换三维柱状图的长方柱体

对于用 bar3 函数创建的 MATLAB 三维柱状图，按照上面用曲面重建柱形的思路，可以将原有的长方柱体替换为圆锥或圆柱。

下面的代码用圆锥曲面重建 MATLAB 三维柱状图中的长方柱体。圆锥曲面可以用 MATLAB 中的 cylinder 函数快速绘制：

```
data = [1 5 3 7 6];
b = bar3(data);
b(1).FaceColor = 'none';
b(1).EdgeColor = 'none';
y = b(1).YData(:,2);
z = b(1).ZData(:,2);
r = (y(3) - y(2))/2;                           % 底面半径
t = r: - r/40:0;
rc = t;
[X,Y,Z] = cylinder(rc);                        % 用 cylinder 函数获取圆锥曲面的网格数据
for i = 1:length(data)
    y0 = (y(2 + 6 * (i - 1)) + y(3 + 6 * (i - 1)))/2;      % y 中心位置(0,y0,0)
```

```
    z0 = z(2 + 6 * (i - 1));                    % 高度
    h(i) = surf(X,Y + y0,Z * z0);               % 用 surf 函数绘制圆锥,进行平移和缩放
    h(i).EdgeColor = 'none';                    % 隐藏边线
    h(i).FaceColor = 'interp';                  % 渐变着色
    hold on
end
ylabel('Y')
zlabel('Z')
% 删除 x 轴刻度线和刻度标签
ax = gca;
axx = ax.XAxis;
axx.TickLength = [0;0];
axx.TickLabels = '';
% 其他设置
grid on
box on
axis equal
camlight left
camlight right
lighting phong
colorbar('Position',[0.68 0.55 0.03 0.18]);
hold off
```

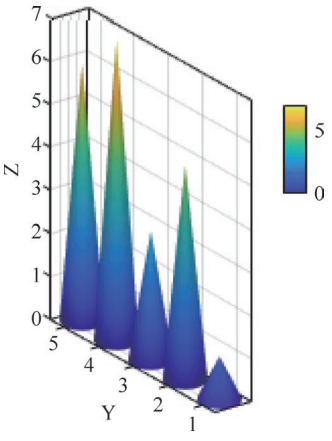

图 5-20　将三维柱状图中的柱体替换为圆锥体

运行代码,生成图 5-20。

按照同样的思路,可以用圆柱重建 MATLAB 三维柱状图中的长方柱体。不同的是,圆柱曲面顶部是空的,需要添加一个圆形面进行封盖。实现代码如下:

```
data = [1 5 3 7 6];
b = bar3(data);
b(1).FaceColor = 'none';
b(1).EdgeColor = 'none';
y = b(1).YData(:,2);
z = b(1).ZData(:,2);
r = (y(3) - y(2))/2;                            % 底面半径
[X,Y,Z] = cylinder(r);
for i = 1:length(data)
    y0 = (y(2 + 6 * (i - 1)) + y(3 + 6 * (i - 1)))/2;    % y 中心位置(0,y0,0)
    z0 = z(2 + 6 * (i - 1));                    % 高度
    % 用圆形面盖住圆柱面
    theta = 0:pi/10:2 * pi;
    rt = 0:r/20:r;
    [TT,RR] = meshgrid(theta,rt);
    XT = RR. * cos(TT);
    YT = y0 + RR. * sin(TT);
    ZT = z0 * ones(size(XT));
    h2 = surf(XT,YT,ZT);
    h2.EdgeColor = 'none';
    h2.FaceColor = 'interp';
    hold on
    % 绘制圆柱,平移和缩放
    h = surf(X,Y + y0,Z * z0);
    h.EdgeColor = 'none';
    h.FaceColor = 'interp';
```

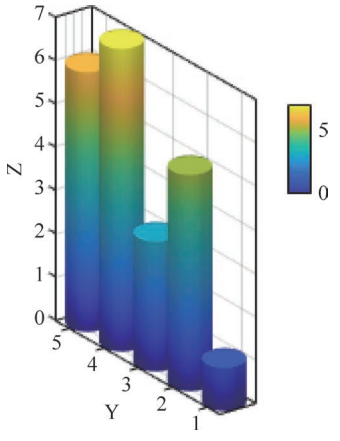

图 5-21　将三维柱状图中的柱体替换为圆柱体

```
end
ylabel('Y')
zlabel('Z')
% 删除 x 轴刻度线和刻度标签
ax = gca;
axx = ax.XAxis;
axx.TickLength = [0;0];
axx.TickLabels = '';
% 其他设置
grid on
box on
axis equal
camlight left
camlight right
lighting phong
colorbar('Position',[0.68 0.55 0.03 0.18]);
hold off
```

运行代码，生成图 5-21。

5.3.6　为线形图添加背景

3.5.2 节介绍了利用坐标系对象的 Backdrop.Face.ColorBinding 属性对二维坐标系的绘图区背景进行颜色渐变填充。这里介绍另一种更通用的方法。思路是用一个 Surface 对象重建绘图区背景，进行渐变色填充或图片填充。

实现代码如下：

```
tiledlayout(2,2);
dx = [1 2 3 4 5];                          % 绘制折线的数据
dy = [1 5 3 7 6];
dxMin = min(dx);
dxMax = max(dx);
dyMin = min(dy);
dyMax = max(dy);

ax1 = nexttile;
ax1.XLim = [dxMin dxMax];
ax1.YLim = [dyMin dyMax];
% 用面片绘制绘图区背景矩形
vert = [dxMin,dyMin;dxMax,dyMin;dxMax,dyMax;dxMin,dyMax];
face = [1,2,3,4];
color = [1 0 0;1 1 0;0 1 0;0 0 1];
% 面片，渐变色填充
p = patch('Vertices',vert,'Faces',face,…
    'FaceVertexCData',color,'FaceColor','interp');
% 绘制折线
line(dx,dy,'LineWidth',1.5,'Color','w')
xlabel('X')
ylabel('Y')
box on

ax2 = nexttile;
ax2.XLim = [dxMin dxMax];
```

```
ax2.YLim = [dyMin dyMax];
vert = [dxMin,dyMin;dxMax,dyMin;dxMax,dyMax;dxMin,dyMax];
face = [1,2,3,4];
color = [0.95 0.66 0.23;0.95 0.66 0.23;1 1 1;1 1 1];
%面片,上下渐变色填充
p = patch('Vertices',vert,'Faces',face,…
    'FaceVertexCData',color,'FaceColor','interp');
%绘制折线
line(dx,dy,'LineWidth',1.5,'Color','b')
xlabel('X')
ylabel('Y')
box on

ax3 = nexttile;
ax3.XLim = [dxMin dxMax];
ax3.YLim = [dyMin dyMax];
x = linspace(dxMin,dxMax,20);
y = linspace(dyMin,dyMax,20);
[X,Y] = meshgrid(x,y);
Z = zeros(size(X));
sur = surface(X,Y,Z);                    %用曲面绘制绘图区背景
sur.CData = sur.YData;                   %用 y 向数据着色,数据与颜色查找表映射得到的颜色
sur.FaceColor = 'interp';
sur.EdgeColor = 'none';
line(dx,dy,'LineWidth',1.5,'Color','w') %绘制折线
xlabel('X')
ylabel('Y')
box on

ax4 = nexttile;
ax4.XLim = [dxMin dxMax];
ax4.YLim = [dyMin dyMax];
x = linspace(dxMin,dxMax,20);
y = linspace(dyMin,dyMax,20);
[X,Y] = meshgrid(x,y);
Z = zeros(size(X));
C = imread('d:\pic.jpg');                %在绘图区加载图片作为背景
sur = surface(X,Y,Z);
sur.FaceColor = 'texturemap';            %注意属性的设置
sur.CData = flipud(C);                   %颜色矩阵上下翻转
sur.EdgeColor = 'none';
line(dx,dy,'LineWidth',1.5,'Color','w')
xlabel('X')
ylabel('Y')
box on
```

运行代码,生成图 5-22。

5.3.7　三维图添加坐标轴面板背景

将 5.3.6 节的问题扩展到三维,可以在三维图中添加坐标轴面板背景。思路与前面相同,可以用面片或曲面重建面板背景,然后进行设置。

用下面的代码在两个坐标系中分别实现三维图形坐标轴面板的渐变色填充和图片填充:

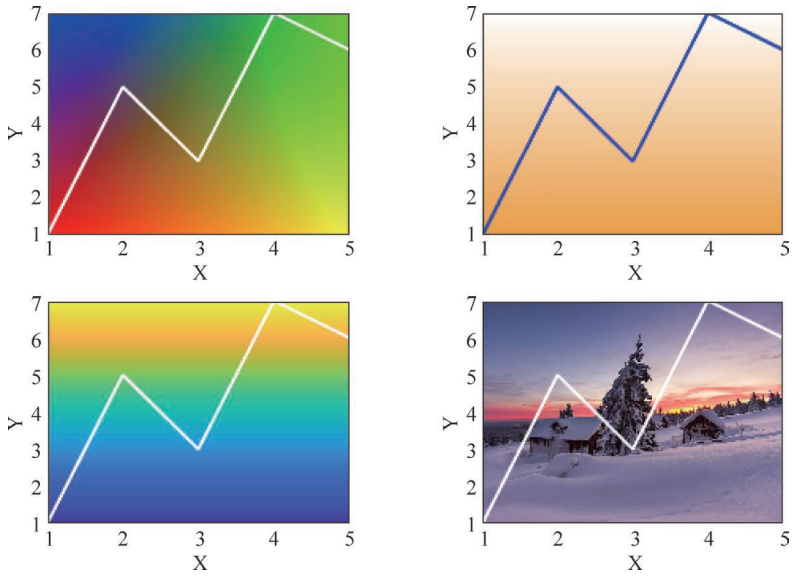

图 5-22　二维坐标系绘图区背景的渐变填充和图片填充

```
tiledlayout(2,2);
[X,Y,Z] = peaks;
xMin = min(min(X));
xMax = max(max(X));
yMin = min(min(Y));
yMax = max(max(Y));
zMin = min(min(Z));
zMax = max(max(Z));

ax1 = nexttile;                          % 在第一个坐标系中用渐变色填充坐标轴面板
ax1.XLim = [xMin xMax];
ax1.YLim = [yMin yMax];
ax1.ZLim = [zMin zMax];

% x - y 面板
vert1 = [xMin,yMin,zMin;xMax,yMin,zMin;xMax,...
    yMax,zMin;xMin,yMax,zMin];
face1 = [1,2,3,4];
color1 = [0.9 0.9 0.9;0.9 0.9 0.9;0.9 0.9 0.9;0.9 0.9 0.9];
p1 = patch('Vertices',vert1,'Faces',face1,...
    'FaceVertexCData',color1,'FaceColor','interp');

% x - z 面板
vert2 = [xMin,yMax,zMin;xMax,yMax,zMin;xMax,...
    yMax,zMax;xMin,yMax,zMax];
face2 = [1,2,3,4];
color2 = [0.95 0.66 0.23;0.95 0.66 0.23;1 1 1;1 1 1];
p2 = patch('Vertices',vert2,'Faces',face2,...
    'FaceVertexCData',color2,'FaceColor','interp');

% y - z 面板
vert3 = [xMax,yMax,zMin;xMax,yMin,zMin;xMax,...
```

```
    yMin,zMax;xMax,yMax,zMax];
face3 = [1,2,3,4];
color3 = [0.95 0.66 0.23;0.95 0.66 0.23;1 1 1;1 1 1];
p3 = patch('Vertices',vert3,'Faces',face3,…
    'FaceVertexCData',color3,'FaceColor','interp');

%绘制曲面
h1 = surface(X,Y,Z);
h1.EdgeColor = 'none';
xlabel('X')
ylabel('Y')
zlabel('Z')
view(3)
camlight

ax2 = nexttile;                          %在第二个坐标系中用图片填充坐标轴面板
ax2.XLim = [xMin xMax];
ax2.YLim = [yMin yMax];
ax2.ZLim = [zMin zMax];

%x-z 面板
vert1 = [xMin,yMin,zMin;xMax,yMin,zMin;xMax,…
    yMax,zMin;xMin,yMax,zMin];
face1 = [1,2,3,4];
color1 = [0.9 0.9 0.9;0.9 0.9 0.9;0.9 0.9 0.9;0.9 0.9 0.9];
p1 = patch('Vertices',vert1,'Faces',face1,…
    'FaceVertexCData',color1,'FaceColor','interp');

C = imread('d:\pic.jpg');

%x-z 面板
x1 = linspace(xMin,xMax,20);
z1 = linspace(zMin,zMax,20);
[X1,Z1] = meshgrid(x1,z1);
Y1 = ones(size(X1)) * yMax;
sur1 = surface(X1,Y1,Z1);
sur1.FaceColor = 'texturemap';
sur1.CData = flipud(C);
sur1.EdgeColor = 'none';

%y-z 面板
y2 = linspace(yMin,yMax,20);
z2 = linspace(zMin,zMax,20);
[Y2,Z2] = meshgrid(y2,z2);
X2 = ones(size(Y2)) * xMax;
sur2 = surface(X2,Y2,Z2);
sur2.FaceColor = 'texturemap';
sur2.CData = flipud(C);
sur2.EdgeColor = 'none';

%绘制曲面
h2 = patch(surf2patch(X,Y,Z,Z));
[f,v,c] = surf2patch(X,Y,Z,Z);
h2.FaceVertexCData = (h2.Vertices(:,3) - …
```

```
    min(h2.Vertices(:,3)))/(max(h2.Vertices(:,3))...
    -min(h2.Vertices(:,3)));
h2.FaceColor = "interp";
h2.EdgeColor = 'none';
xlabel('X')
ylabel('Y')
zlabel('Z')
view(3)
camlight
```

运行代码，生成图 5-23。可见，左图中坐标轴面板的渐变色填充已完美实现，右图中坐标轴面板的图片填充也已实现，但是曲面无法正常着色。

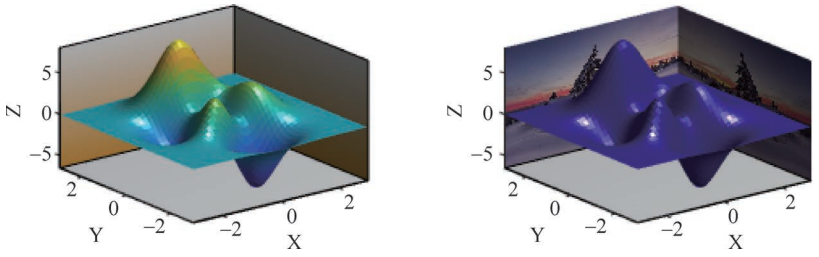

图 5-23　三维坐标系绘图区背景的渐变填充和图片填充

5.4　组合已有类型的图表创建新图表

前面两节介绍了用基本图形元素创建新图表及在 MATLAB 图表基础上重建图元创建新图表。除此之外，还可以通过组合已有类型的图表创建新图表。

5.4.1　创建带误差条的柱状图

带误差条的柱状图如图 5-24 所示，该图在二维柱状图的基础上叠加误差条图，从而提供更多的信息。该图的绘制很简单，组合二维柱状图和误差条图就可以了。

下面的代码实现了带误差条的柱状图的绘制：

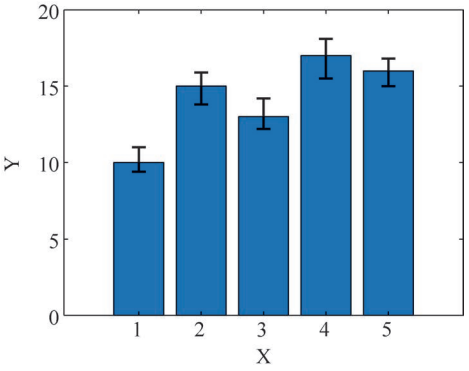

图 5-24　带误差条的柱状图

```
x = 1:5;
data = [10 15 13 17 16];          % 绘制柱状图的数据
eh = [0.6 1.2 0.8 1.5 1.0];       % 误差条上触须数据
el = [1.0 0.9 1.2 1.1 0.8];       % 误差条下触须数据
b = bar(x,data);                  % 绘制柱状图
hold on                           % 叠加绘图
h = errorbar(x,data,eh,el);       % 误差条图
h.Color = 'k';                    % 误差条属性设置
h.LineStyle = 'none';
h.LineWidth = 1;
hold off
xlabel('X')
ylabel('Y')
box on
```

5.4.2　自行创建帕累托图

MATLAB 的统计工具箱提供了绘制帕累托图的函数，这里我们自行创建一下。图 5-25 即为帕累托图，可见，帕累托图是二维柱状图和线形图的组合。而且，该图实际上是一个双轴图，有两个 y 轴，共用一个 x 轴。柱状图表示数据的大小，注意从大到小进行了排序。线形图表示各条形数据当前累加和占总和的百分比。帕累托图能清晰地表明主要组分及其占比情况。

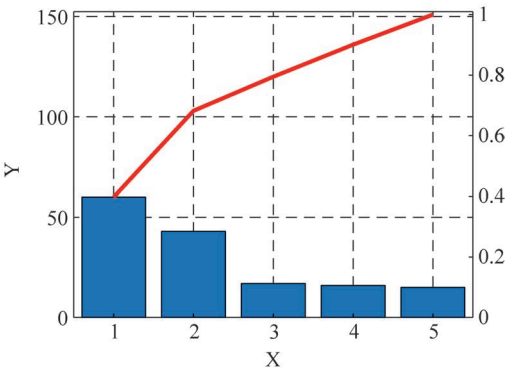

图 5-25　创建帕累托图

下面的代码实现了帕累托图的自行创建：

```
x = 1:5;                                                    % 绘图数据
data0 = [60 15 43 17 16];
y1 = sort(data0,'descend');                                 % 将数据从大到小排序
yp = y1./sum(y1);
y2(1) = yp(1);
for i = 2:length(data0)
   y2(i) = y2(i - 1) + yp(i);                               % 排序后的数据累加求和
end

fig = figure;
ax1 = axes(fig);
b = bar(x,y1);                                              % 绘制二维柱状图
box(ax1,"off")

ax2 = axes('Position',get(ax1,'Position'), …                % 创建第 2 个坐标系
     'XAxisLocation','top', …
     'YAxisLocation','right', …
     'Color','none', …
     'XColor','k','YColor','k');
h = line(x,y2,'Color','r', 'LineWidth',1.5,'Parent',ax2); % 用累加和数组绘制线形图

ylim(ax1,[0 sum(data0) * 1.01])                             % 两个坐标系的取值范围
ylim(ax2,[0 1.01])
xlim(ax1,[0.5 5.5])
xlim(ax2,[0.5 5.5])
ax2.XAxis.TickLabels = '';
ax2.XAxis.TickLength = [0 0];

xlabel(ax1,'X')
ylabel(ax1,'Y')
grid(ax1,"on")
```

5.5　标注

MATLAB 中可以通过创建标注对象对图表进行标注。可选的标注类型包括箭头、双头箭头、文本箭头、直线、椭圆、矩形和文本框等。

5.5.1 标注对象

Arrow 对象类型表示单箭头。可以用 set 和 get 函数及属性编辑器设置和查询标注对象的属性，使用 annotation 函数创建标注对象并获取它们的句柄。箭头头的可选样式如表 5-1 所示，父对象为 Axes 对象。

表 5-1 箭头头的可选样式

箭头头样式字符串	图 形	箭头头样式字符串	图 形
none		cback3	
plain		star4	
ellipse		rectangle	
vback1		diamond	
vback2(Default)		rose	
vback3		hypocycloid	
cback1		astroid	
cback2		deltoid	

Doublearrow 对象类型表示双头箭头，其属性类型与 Arrow 对象的基本相同，不同的是，涉及箭头端部的属性要一分为二。例如，原来的 HeadStyle 属性现在变为 Head1Style 和 Head2Style 属性。

Ellipse 对象类型表示椭圆标注。

Line 对象类型表示直线段标注。它具有 Color、LineStyle、LineWidth、X 和 Y 等属性，分别定义直线段的颜色、线型、线宽和顶点坐标。

Rectangle 对象类型表示矩形标注。

Textarrow 对象类型表示文本箭头标注。

Textbox 对象类型表示文本框标注。

5.5.2 创建标注对象

MATLAB 中使用 annotation 函数创建标注对象。标注对象在隐藏坐标系中创建，该坐标系的宽度和高度与图形窗口的相同，这样可以标准化坐标(左下角点的坐标为(0,0)，右上角的坐标为(1,1)，坐标值表示占窗口宽度或高度的百分比)，在图形窗口中任意一处指定标注对象的位置。

用下面的代码绘制正弦曲线，并添加各种类型的标注：

```
x = -4:.2:4;
y = sin(x);
plot(x,y,'LineWidth',1.5)
ylim([-1.5 1.5])
%带箭头的文本标注
annotation('textarrow',[0.4 0.5],[0.6 0.5],'string','y = sin(x)')
%椭圆形标注
annotation('ellipse',[0.55 0.7 0.25 0.15])
%矩形标注
annotation('rectangle',[0.25 0.2 0.25 0.15],'FaceColor','y','FaceAlpha',0.5)
```

```
% 文本框标注
annotation('textbox',[0.6 0.2 0.15 0.12],'String',{'最大值','最小值'})
xlabel('X')
ylabel('Y')
box on
```

运行代码，生成图 5-26。

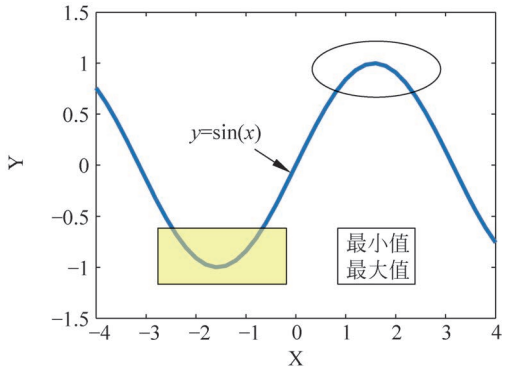

图 5-26　为图表添加标注

5.6　图形几何变换

在绘图过程中，经常需要对图形进行平移、缩小、放大和旋转等操作。MATLAB 提供了专门的函数，实现图形的几何变换。

5.6.1　几何变换的基本原理

旋转变换使对象绕 x、y 或 z 轴旋转，逆时针旋转时角度为正。如果旋转角度为 θ，则下面的矩阵定义绕各轴的旋转。

$$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\theta_x & -\sin\theta_x & 0\\ 0 & \sin\theta_x & \cos\theta_x & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\quad\begin{bmatrix}\cos\theta_y & 0 & \sin\theta_y & 0\\ 0 & 1 & 0 & 0\\ -\sin\theta_y & 0 & \cos\theta_y & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\quad\begin{bmatrix}\cos\theta_z & -\sin\theta_z & 0 & 0\\ \sin\theta_z & \cos\theta_z & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

平移变换相对于当前位置移动对象，用距离 t_x、t_y 和 t_z 指定平移。下面的矩阵显示了这些元素在变换矩阵中的位置。

$$\begin{bmatrix}1 & 0 & 0 & t_x\\ 0 & 1 & 0 & t_y\\ 0 & 0 & 1 & t_z\\ 0 & 0 & 0 & 1\end{bmatrix}$$

缩放（比例）变换改变对象的大小，指定比例因子 s_x、s_y 和 s_z，创建下面的矩阵：

$$\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

默认变换矩阵是单位矩阵，可以用 eye 函数创建。下面是单位矩阵：

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

5.6.2 实现图形变换

使用 hgtransform 对象的 Matrix 属性可使 hgtransform 对象的所有子对象应用一个变换。

变换是绝对意义上的，而不是相对于当前变换进行的。例如，如果应用一个变换，它在 x 方向将 hgtransform 对象平移 5 个单位，然后用另一个变换将对象在 y 方向平移 4 个单位，则对象的最后位置是相对于原来位置在 y 方向平移了 4 个单位，即起作用的是最后一次变换。如果希望做相对变换，必须将单个变换矩阵聚合到一个单一矩阵中。

通常通过聚合单个矩阵，并将结果赋给 Matrix 属性实现矩阵聚合。注意，矩阵相乘是不可逆的，所以矩阵的顺序会影响计算结果。例如，假设要进行一个先缩放，再平移、旋转的操作，可以像下面这样做矩阵相乘的运算：

```
C = R * T * S    % 从右向左计算
```

其中，S 是缩放矩阵，T 是平移矩阵，R 是旋转矩阵，而 C 是 3 种操作的组合。然后设置 hgtransform 对象的 Matrix 属性值为 C：

```
set(hgtransform_handle,'Matrix',C)
```

注意，下面的语句行：

```
set(hgtransform_handle,'Matrix',C)
set(hgtransform_handle,'Matrix',eye(4))
```

与

```
C = eye(4) * R * T * S
set(hgtransform_handle,'Matrix',C)
```

是不等价的。将单位矩阵聚合到其他矩阵对于复合矩阵没有影响。

因为变换操作是在绝对意义上指定的，所以可以通过将当前变换设置为单位矩阵实现一系列变换操作。例如，下面的语句：

```
set(hgtransform_handle,'Matrix',eye(4))
```

可使对象 hgtransform_handle 返回至其变换前的方位。

因为旋转是绕原点进行的，常常需要平移 hgtransform 对象，这样旋转的目标坐标暂时位于原点。采用旋转变换矩阵以后，将 hgtransform 对象移回原点。

MATLAB 操作中，用 hgtransform 函数创建 hgtransform 图形对象。变换时，将需要变换的对象作为 hgtransform 对象的子对象，例如：

```
h = hgtransform;
surface('Parent',h,…)
```

将 hgtransform 对象作为父对象的主要优点在于，该对象具有对其子对象进行变换的能力，这些变换包括平移、比例化和旋转等。hgtransform 对象可以是任意个数 axes 对象子对象的父对象，包括其他 hgtransform 对象。

使用 makehgtform 创建用于转换、缩放和旋转图形对象的变换矩阵。通过将变换指定给父变换对象的 Matrix 属性，将变换应用于图形对象。

- M＝makehgtform　返回恒等变换矩阵。
- M＝makehgtform('translate',[tx ty tz])或 M＝makehgtform('translate',tx,ty,tz)　返回分别沿 x、y 和 z 轴按 tx、ty 和 tz 进行转换的变换矩阵。
- M＝makehgtform('scale',s)　返回沿 x、y 和 z 坐标轴均匀缩放的变换矩阵。
- M＝makehgtform('scale',[sx,sy,sz])　返回分别沿 x、y 和 z 轴按 sx、sy 和 sz 进行缩放的变换矩阵。
- M＝makehgtform('xrotate',t)　返回围绕 x 轴旋转 t 弧度的变换矩阵。
- M＝makehgtform('yrotate',t)　返回围绕 y 轴旋转 t 弧度的变换矩阵。
- M＝makehgtform('zrotate',t)　返回围绕 z 轴旋转 t 弧度的变换矩阵。
- M＝makehgtform('axisrotate',[ax,ay,az],t)　围绕轴[ax ay az]旋转 t 弧度。

注意，可以在对 makehgtform 的一个调用中指定多个操作，MATLAB 返回依次执行所有指定操作所得的变换矩阵。

下面生成一个三维曲面，然后对它进行平移、缩小和旋转操作。在命令窗口输入代码。

首先创建一个 axes 对象并设置坐标系范围，调整视图。防止缩放时 MATLAB 自动调整范围：

```
>> ax = axes('XLim',[ - 6 6], 'YLim',[ - 6 6], 'ZLim',[ - 10 10]);
>> view(3)
```

创建希望作为 hgtransform 对象子对象的对象：

```
>> [x,y,z] = peaks;
>> h = surface(x,y,z);
>> h.EdgeColor = 'none';
>> camlight
>> box on
```

运行代码，生成图 5-27。

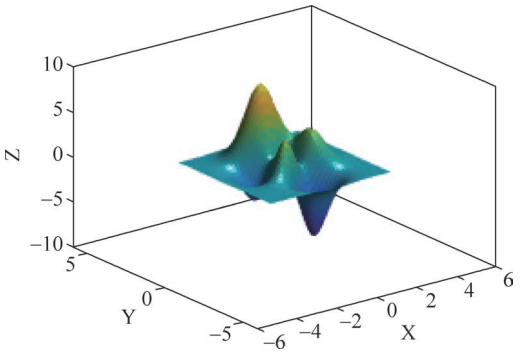

图 5-27　生成图形

创建一个 hgtransform 对象，并作为刚刚创建的一系列曲面对象的父对象：

```
>> t = hgtransform('Parent',ax);
>> set(h,'Parent',t)
```

将平移、缩放和旋转矩阵初始化为单位矩阵：

```
>> Tz = eye(4);
>> Sxy = Tz;
>> Rz = Tz;
```

将图形沿 z 轴向下平移 4 个单位：

```
>> Tz = makehgtform('translate',0,0, - 4);
```

```
>> set(t,'Matrix',Tz)
```

运行代码，生成图 5-28。

将图形放大 1.5 倍：

```
>> Sxy = makehgtform('scale',1.5);
>> set(t, 'Matrix',Sxy)
```

运行代码，生成图 5-29。注意，曲面是在原始位置上放大的，而不是在下移 4 个单位后的位置上放大的。

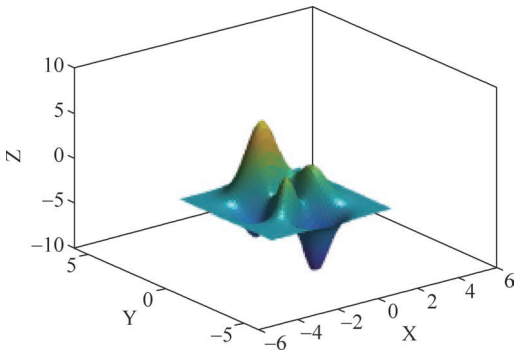

图 5-28　将图形沿 z 轴向下移动 4 个单位

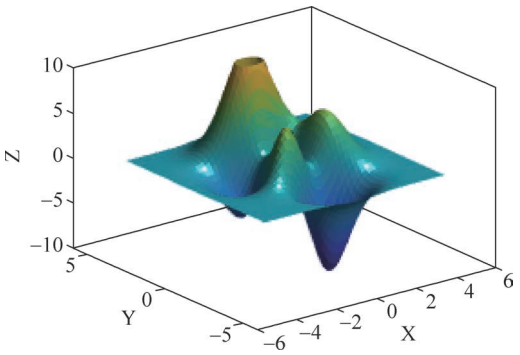

图 5-29　图形放大 1.5 倍

将图形绕 z 轴逆时针方向旋转 90°：

```
>> Rz = makehgtform('zrotate',pi/2);
>> set(t,'Matrix',Rz)
```

运行代码，生成图 5-30。同样，曲面也是在原始位置上旋转的。

将图形先沿 z 轴向下移 2 个单位，然后放大到 1.5 倍并绕 z 轴逆时针方向旋转 90°。注意，后面的变换矩阵是在原来矩阵的基础上左乘的：

```
>> set(t,'Matrix',Rz * Sxy * Tz)
```

运行代码，生成图 5-31。

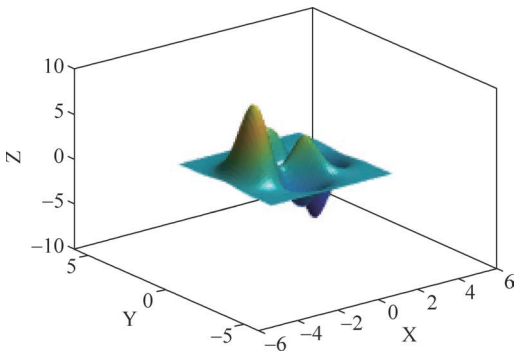

图 5-30　图形绕 z 轴逆时针旋转 90°

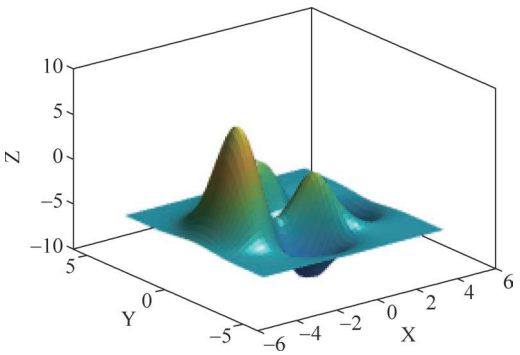

图 5-31　对图形先后进行平移、放大和旋转操作

用单位矩阵重新设置原始方位和大小：

```
>> set(t,'Matrix', eye(4))
```

5.7　图形块

绘图时常见的一种情况是，一个部件由几个或几十个基本的图形单元组成。处理时，一般将这些图形单元作为一个图形块进行处理，例如，一起平移、旋转，一起被选取等。

在 MATLAB 中创建图形块很简单，创建一个图形块就是创建一个 hggroup 对象，然后把组成图形块的各图形单元作为该 hggroup 对象的子对象。

利用 hggroup 函数创建 hggroup 对象，该函数的调用格式为

```
h = hggroup
h = hggroup(…,'PropertyName',propertyvalue)
```

hggroup 对象是任何坐标系子对象和其他 hggroup 对象的父对象。

用下面的代码创建一个圆角矩形和一个圆，其父对象是同一个 hggroup 对象。即用该圆角矩形和圆组成一个图形块进行处理。对该图形块进行平移和缩小变换：

```
>> hg = hggroup;
>> rec = rectangle('Position',[3,7,20,8], 'Curvature',[.3 .4], 'FaceColor', 'g', 'Parent',hg);
>> cir = rectangle('Position',[8,15,10,10], 'Curvature',[1 1], 'LineWidth',3, 'FaceColor', 'r',
'Parent',hg);
>> axis([0 25 0 25]);
>> axis equal
>> box on
```

运行代码，生成图 5-32。

下面对圆和圆角矩形进行平移和缩小变换，因为上面已经将它们绑定到组对象 hg 中，直接针对该组对象进行操作即可：

```
>> t = hgtransform;
>> set(hg, 'Parent',t)
>> Tz = makehgtform('translate',1,2,0);
>> Sxy = makehgtform('scale',0.5);
>> set(t,'matrix', Sxy * Tz)
```

运行代码，生成图 5-33。

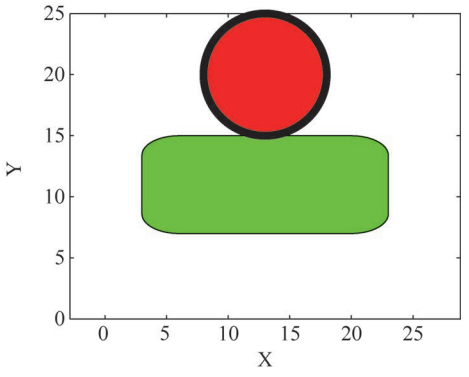

图 5-32　生成图形并组合为图形块

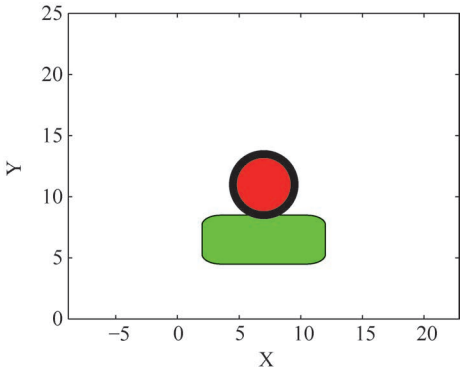

图 5-33　平移和缩小图形块