

MATLAB 不仅具有强大的数值计算功能,还有强大的绘图功能,能够将数据方便地以二维、三维乃至四维的图形可视化,并且能够设置图形的颜色、线性、视觉角度等。应用 MATLAB 除了能作一般的曲线图、条形图、散点图等统计图形外,还能绘制流线图、三维向量图等工程实用图形。由于系统采用面向对象的技术和丰富的矩阵计算,所以在图形处理方面既方便又高效。

### 5.1 图形绘制基础

#### 5.1.1 离散函数

一个二元实数标量对 $(x_0,y_0)$ 可以用平面上的点来表示,一个二元实数标量数组 $[(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)]$ 可以用平面上的一组点来表示。对于离散函数 $Y=f(X)$ ,当 $X$ 为一维标量数组 $[x_1,x_2,\cdots,x_n]$ 时,根据函数关系可以求出 $Y$ 相应的一维标量数组 $[y_1,y_2,\cdots,y_n]$ 。当把这两个数组在直角坐标系中用点序列表示时,就实现了离散函数的可视化。当然,这些图形上的离散序列反映的只是 $X$ 所限定的有限点上或是有限区间内的函数关系,应当注意的是,MATLAB 无法实现对无限区间上的数据可视化。

**【例 5-1】** 在某次工程实验中,测得时间  $t$  与温度  $T$  的数据如表 5-1 所示。

表 5-1 时间与温度的关系

|                         |   |      |      |      |      |      |       |       |       |     |       |       |     |
|-------------------------|---|------|------|------|------|------|-------|-------|-------|-----|-------|-------|-----|
| 时间 $t/s$                | 0 | 1    | 2    | 3    | 4    | 5    | 6     | 7     | 8     | 9   | 10    | 11    | 12  |
| 温度 $T/^{\circ}\text{C}$ | 0 | 32.5 | 45.8 | 79.0 | 86.5 | 96.5 | 107.3 | 110.4 | 115.7 | 118 | 119.2 | 119.8 | 120 |

描绘出这些点,以观察温度随时间变化的关系。

其实现的 MATLAB 代码如下:

```
>> clear all;
t = 0:12; % 输入时间 t 的数据
% 输入温度 T 的数据
T = [0 32.5 45.8 79.0 86.5 96.5 107.3 110.4 115.7 118 119.2 119.8 120];
plot(t,T,'ro') % 用红色描绘出相应的数据点
xlabel('时间');ylabel('温度');
```

运行程序,效果如图 5-1 所示。

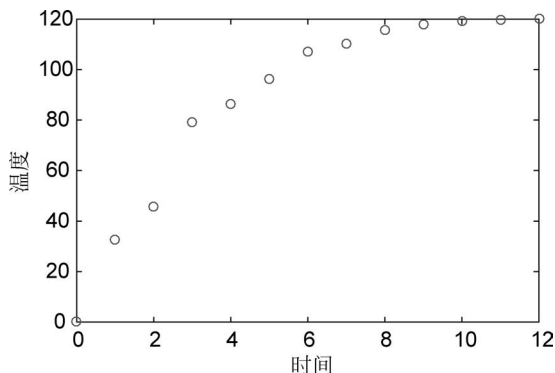


图 5-1 离散数据散点图

### 5.1.2 连续函数

在 MATLAB 中无法画出真正的连续函数,因此在实现连续函数的可视化时,首先必须将连续函数用在一组离散自变量上来计算函数结果,然后将自变量数组和结果数组在图形中表示出来。

当然,这些离散的点还是不能表现函数的连续性。为了更形象地表现函数的规律及其连续变化,通常采用以下两种方法。

(1) 对离散区间进行更细的划分,逐步趋近函数的连续变化特性,直到达到视觉上的连续效果。

(2) 把每两个离散点用直线连接,以每两个离散点之间的直线来近似表示两点间的函数特性。

**【例 5-2】** 绘制方程 
$$\begin{cases} x = 2(\cos t - t \sin t) \\ y = 2(\sin t + t \cos t) \end{cases}$$
 在  $t \in [-2\pi, 2\pi]$  上的图像。

```
>> clear all;
t1 = -2 * pi:pi/5:2 * pi;
t2 = -2 * pi:pi/20:2 * pi;
x1 = 2 * (cos(t1) - t1. * sin(t1));
y1 = 2 * (sin(t1) + t1. * cos(t1));
x2 = 2 * (cos(t2) - t2. * sin(t2));
y2 = 2 * (sin(t2) + t2. * cos(t2));
subplot(221);plot(x1,y1,'m. ');
title('图 1');axis square;
subplot(222);plot(x2,y2,'m. ');
title('图 2');axis square;
subplot(223);plot(x1,y1);
title('图 3');axis square;
subplot(224);plot(x2,y2);
title('图 4');axis square;
```

运行程序,效果如图 5-2 所示。

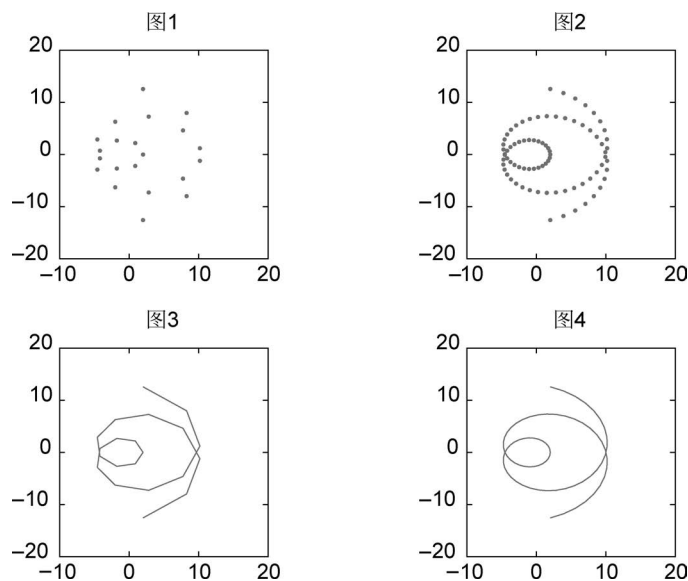


图 5-2 连续函数绘图

## 5.2 二维基本绘图

二维图形是 MATLAB 图形的基础,也是应用最广泛的图形类型之一。MATLAB 提供了许多二维图形绘制函数。

### 5.2.1 基本绘图函数

MATLAB 的基本绘图函数包括 line 函数、plot 函数和 polar 函数, line 函数是直角坐标系中的简单绘图函数, plot 函数是直角坐标系中常用的绘图函数,而 polar 函数是极坐标中的绘图函数。

在 MATLAB 中绘制曲线的基本函数有很多,表 5-2 列出了常用的基本绘图函数。

表 5-2 常用的基本绘图函数

| 函 数      | 功 能                   |
|----------|-----------------------|
| plot     | 建立向量或矩阵的图形            |
| line     | 将数组中的各点用线段连接起来        |
| semilogx | x 轴用于对数标度,y 轴线性标度绘制图形 |
| semilogy | y 轴用于对数标度,x 轴线性标度绘制图形 |
| loglog   | x、y 轴都取对数标度建立图形       |
| plotyy   | 在图的左、右两侧分别建立纵坐标轴      |
| subplot  | 同一个图形窗口中同时显示多个坐标轴的图形  |

#### 1. plot 函数

MATLAB 中最常用的二维曲线的绘图函数为 plot,使用该函数 MATLAB 将开辟一个图形窗口,并画出坐标上的一条二维曲线。函数的调用格式如下。

plot(Y): 输入参数 Y 就是 Y 轴的数据, 一般习惯输入向量, 则 plot(Y) 可以用于绘制索引值所对应的行向量 Y, 若 Y 为复数, 则 plot(Y) 等于 plot(real(Y), image(Y)); 在其他几种使用方式中, 如果有复数出现, 则复数的虚数部分将不被考虑。

plot(X1, Y1, ..., Xn, Yn): 若 Xi、Yi 均为实数向量, 且为同维向量(可以不是同型向量), 则 plot 先描出点(Xi, Yi), 然后用直线依次相连; 若 Xi、Yi 为复数向量, 则不考虑虚数部分; 若 Xi、Yi 均为同型实数矩阵, 则 plot(Xi, Yi) 依次画出矩阵的几条线段; 若 Xi、Yi 一个为向量, 另一个为矩阵, 且向量的维数等于矩阵的行数或列数, 则矩阵按向量的方向分解成几个向量, 再与向量配对分别画出, 矩阵可分解成几个向量就有几条线。在上述的几种使用形式中, 若有复数出现, 则复数的虚数部分将不被考虑。

plot(X1, Y1, LineSpec, ..., Xn, Yn, LineSpec): LineSpec 为选项(开关量)字符串, 用于设置曲线颜色、线型、数据点等, 如表 5-3 所示; LineSpec 的标准设定值的前 7 种颜色(蓝、绿、红、青、品红、黄、黑)依次自动着色。

表 5-3 常用的绘图选项

| 选 项 | 含 义      | 选 项 | 含 义        |
|-----|----------|-----|------------|
| —   | 实线       | .   | 用点号标出数据点   |
| --  | 虚线       | O   | 用圆圈标出数据点   |
| :   | 点线       | x   | 用叉号标出数据点   |
| -.  | 点画线      | +   | 用加号标出数据点   |
| r   | 红色       | s   | 用小正方形标出数据点 |
| g   | 绿色       | D   | 用菱形标出数据点   |
| b   | 蓝色       | V   | 用下三角标出数据点  |
| y   | 黄色       | ^   | 用上三角标出数据点  |
| m   | 洋红       | <   | 用左三角标出数据点  |
| c   | 青色       | >   | 用右三角标出数据点  |
| w   | 白色       | H   | 用六角形标出数据点  |
| k   | 黑色       | P   | 用五角形标出数据点  |
| *   | 用星号标出数据点 |     |            |

plot(X1, Y1, LineSpec, 'PropertyName', PropertyValue): 对所有用 plot 函数创建的图形进行属性值设置, 常用属性如表 5-4 所示。

表 5-4 常用属性

| 属 性 名      | 含 义      | 属 性 名           | 含 义        |
|------------|----------|-----------------|------------|
| LineWidth  | 设置线的宽度   | MarkerEdgeColor | 设置标记点的边缘颜色 |
| MarkerSize | 设置标记点的大小 | MarkerFaceColor | 设置标记点的填充颜色 |

h = plot(X1, Y1, LineSpec, 'PropertyName', PropertyValue): 返回绘制函数的句柄值 h。

**【例 5-3】** 在  $[0, 2\pi]$  上同时绘制 3 条曲线  $y_1 = \sin(x)$ 、 $y_2 = \sin(x - 0.25)$  及  $y_3 = \sin(x - 0.5)$ , 并设置曲线线型和颜色。

```
>> clear all;
x = 0:pi/100:2*pi;
y1 = sin(x);
y2 = sin(x - 0.25);
```

```
y3 = sin(x-0.5);
figure
plot(x,y1,x,y2,'r--',x,y3,'k:')
```

运行程序,效果如图 5-3 所示。

## 2. line 函数

line 函数用于在直角坐标系下简单画线,调用格式如下。

line(X,Y): X 和 Y 都是一维数组,line 函数将数组中(Xi,Yi)代表的各点用线段连接起来,形成一条折线。

line(X,Y,Z): 创建的三维坐标中的线段。

line(X,Y,Z,'PropertyName',propertyvalue,...): 设置对应的属性名 PropertyName 及属性值 propertyvalue。

h = line(...): 返回绘制线段的句柄值 h。

**【例 5-4】** 利用 line 函数绘制线型。

```
>> clear all;
t = 0:pi/20:2*pi;
hline1 = plot(t,sin(t),'k');
ax = gca;
hline2 = line(t+.06,sin(t),...
    'LineWidth',4,...
    'Color',[.8 .8 .8],...
    'Parent',ax);
set(gca,'Children',[hline1 hline2])
```

运行程序,效果如图 5-4 所示。

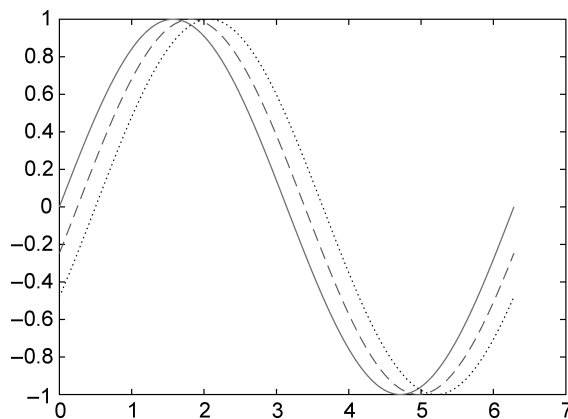


图 5-3 3 条正弦曲线的绘制

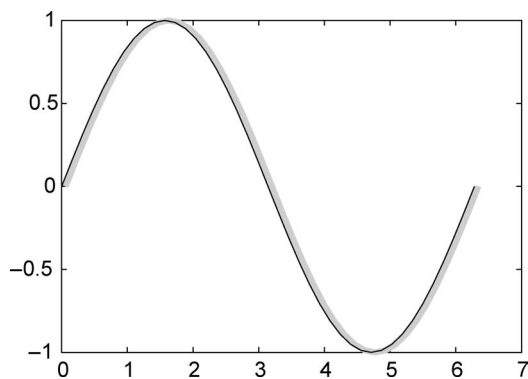


图 5-4 line 函数绘图

## 3. 轴对数坐标

利用 semilogx、semilogy 函数绘制图形时,x、y 轴采用对数坐标。如果没有指定使用的颜色,当绘制的线条较多时,semilogx、semilogy 函数将自动使用当前轴的 ColorOrder 与 LineStyleOrder 属性指定的颜色顺序与线型顺序来绘制。函数的调用格式如下。

`semilogx(Y)`: 对  $x$  轴的刻度求常用对数(以 10 为底), 而  $y$  轴为线性刻度。若  $Y$  为实数向量或矩阵, 则结合  $Y$  的列向量下标与  $Y$  的列向量绘制出线条; 若  $Y$  为复数向量或矩阵, 则 `semilogx(Y)` 等价于 `semilogx(real(Y), imag(Y))`。在 `semilogx` 函数的其他调用格式中,  $Y$  的虚数部分将被忽略。

`semilogx(X1, Y1, ...)`: 结合  $X_i$  与  $Y_i$  绘制出线条, 若其中只有  $X_i$  或  $Y_i$  为矩阵, 另一个为向量, 行向量的维数等于矩阵的列数, 列向量的维数等于矩阵的行数, 则按向量的方向分解矩阵, 再与向量结合, 分别绘制出线条。

`semilogx(X1, Y1, LineSpec, ...)`: 按顺序取 3 个参数  $X_i$ 、 $Y_i$ 、`LineSpec` 绘制线条, 参数 `LineSpec` 指定使用的线型、标记符号及颜色。用户可以混合使用二参数与三参数形式。

`semilogx(..., 'PropertyName', PropertyValue, ...)`: 对所有由 `semilogx` 函数生成的图形对象句柄的属性进行设置。

`h=semilogx(...)`: 返回 line 图形句柄向量, 每条线对应一个句柄, 相对于  $x$  轴采用对数坐标形式。

`h=semilogy(...)`: 返回 line 图形句柄向量, 每条线对应一个句柄, 相对于  $y$  轴采用对数坐标形式。

利用 `loglog` 函数绘制图形时,  $x$  轴与  $y$  轴均采用对数坐标, 其调用格式与 `semilogx` 函数相同。

**【例 5-5】** 绘制双对数图形、 $x$  轴半对数图形及  $y$  轴半对数图形。

```
>> clear all;
x = 0.1:10;
figure; loglog(x, exp(x), 's');           % 双对数图形
figure; semilogy(x, 10.^x);              % y 轴半对数图形
figure; semilogx(x, 10.^x);              % x 轴半对数图形
```

运行程序, 效果如图 5-5 所示。

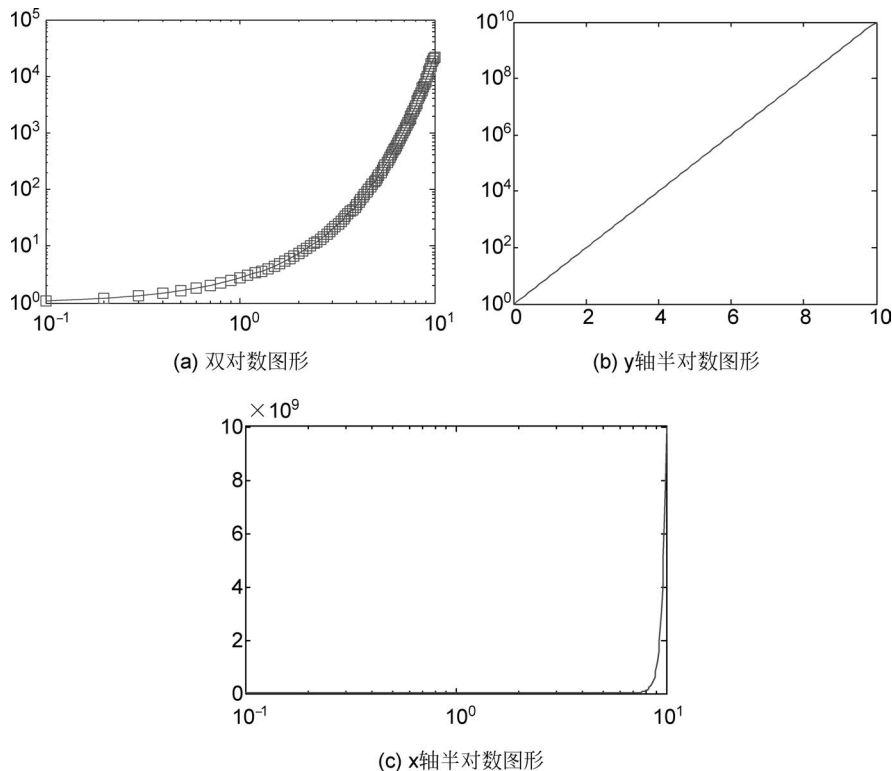


图 5-5 轴对数坐标图

#### 4. plotyy 函数

实际中,如果两组数据的数据范围相差较大,而又希望放在同一个图形中比较分析,可以绘制双 y 轴图形。在 MATLAB 中,提供了 plotyy 函数实现双 y 轴图形的绘制。函数的调用格式如下。

plotyy(x1,y1,x2,y2): 在一个图形窗口同时绘制两条曲线(x1,y1)和(x2,y2),曲线(x1,y1)用左侧的 y 轴,曲线(x2,y2)用右侧的 y 轴。

plotyy(x1,y1,x2,y2,fun): fun 是字符串格式,用来指定绘图的函数名,如 plot、semilogx 等。例如,plotyy(x1,y1,x2,y2,'semilogx')就是用函数 semilogx 来绘制曲线(x1,y1)和(x2,y2)。

plotyy(x1,y1,x2,y2,fun1,fun2): 和第二种形式类似,只是用 fun1 和 fun2 可以指定不同的绘图函数分别绘制这两种曲线。

**【例 5-6】** 用不同标度在同一坐标内绘制曲线  $y_1 = 200e^{-0.05x} \sin(x)$  及曲线  $y_2 = 0.8e^{-0.5x} \sin(10x)$ 。

```
>> clear all;
x = 0:0.01:20;
y1 = 200 * exp(-0.05 * x) .* sin(x);
y2 = 0.8 * exp(-0.5 * x) .* sin(10 * x);
plotyy(x,y1,x,y2,'plot')
```

运行程序,效果如图 5-6 所示。

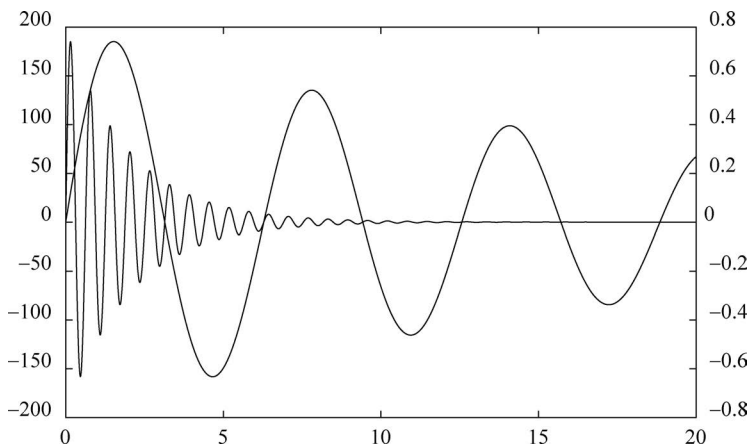


图 5-6 双 y 轴效果图

#### 5. subplot 函数

有的时候,为了便于对比或节省绘图空间,需在同一个图形窗口中同时显示多个坐标轴的图形,即建立多个坐标系并在坐标系中分别绘图,这时就需要用到 subplot 函数。函数的调用格式如下。

subplot(m,n,p): 将当前图形窗口分成  $m \times n$  个绘图区,即共 m 行,每行 n 个,子绘图区的编号按行优先从左到右编号。该函数选定第 p 个区为当前活动区,在每一个子绘图区允许以不同的坐标系单独绘制图形。

subplot(m,n,p,'replace'): 如果定义的坐标轴已经存在,则删除已有的,并创建一个新的坐标轴。

subplot(m,n,p,'align'): 对齐坐标轴。

subplot(h): 使句柄 h 对应的坐标轴为当前显示,用于后面图形的输出显示。

subplot('Position',[left bottom widthheight]): 在指定的位置上建立坐标轴。

subplot(...,prop1,value1,prop2,value2,...): 设置坐标轴的属性名及属性值。

h=subplot(...): 返回坐标轴的句柄值 h。

**【例 5-7】** 利用 subplot 绘制多个子图。

```
>> clear all;
figure
y = zeros(4,15);
for k = 1:4
    y(k,:) = rand(1,15);
    subplot(2, 2, k)
    plot(y(k,:));
end
hax = axes('Position', [.35, .35, .3, .3]);
bar(hax,y,'EdgeColor','none')
set(hax,'XTick',[])
```

运行程序,效果如图 5-7 所示。

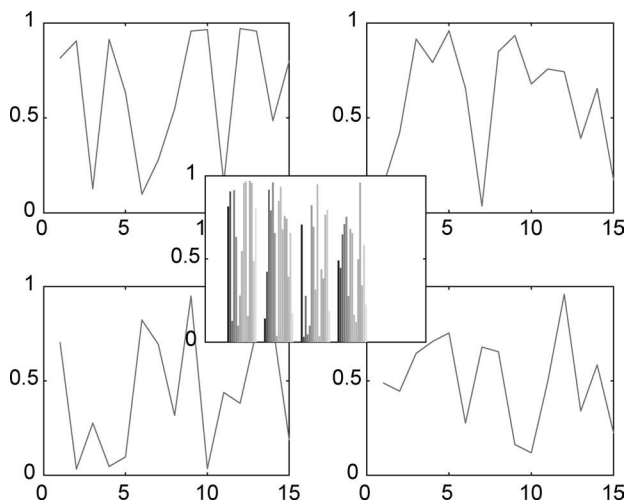


图 5-7 绘制多个子图

## 5.2.2 图形注释

为了让所绘制的图形让人看起来舒服、易懂,MATLAB 提供了许多图形控制函数,下面给予介绍。

### 1. 坐标轴注释

在 MATLAB 中,提供了 xlabel 函数给图形对象的 x 轴加注释;提供了 ylabel 函数给图形对象的 y 轴加注释;提供了 zlabel 函数给图形对象的 z 轴加注释。函数的调用格式如下。

xlabel('string'): 在当前轴对象中的 x 轴上标注说明语句 string。

xlabel(fname): 先执行函数 fname,返回一个字符串,然后在 x 轴旁边显示出来。

xlabel(...,'PropertyName',PropertyValue,...): 指定轴对象中要控制的属性名及要改变的属性值。

h=xlabel(...): 返回作为 x 轴标注的 text 对象句柄值 h。

ylabel(...): 在当前轴对象中的 y 轴上标注说明语句 string。



`h=ylabel(...)`: 返回作为 y 轴标注的 text 对象句柄值 h。

`zlabel(...)`: 返回作为 z 轴标注的 text 对象句柄值 h。

`h=zlabel(...)`: 返回作为 z 轴标注的 text 对象句柄值 h。

## 2. 图形注释

在 MATLAB 中,提供了 title 函数用于给图形对象加标题。函数的调用格式如下。

`title('string')`: 在当前坐标轴上方正中央放置字符串 string 作为图形标题。

`title(fname)`: 先执行能返回字符串的函数 fname,然后在当前轴上方正中央放置返回的字符串作为标题。

`title(...,'PropertyName',PropertyValue,...)`: 对由 title 函数生成的 text 图形对象的属性进行设置。

`h=title(...)`: 返回作为标题的 text 对象句柄值 h。

## 3. 标志文本

在对所绘制的图形进行详细的标注时,最常见的两个函数为 text 与 gtext,它们均可以在图形的具体部分进行标注。函数的调用格式如下。

`text(x,y,'string')`: 在图形中指定的位置(x,y)上显示字符串 string。

`text(x,y,z,'string')`: 在三维图形空间中的指定位置(x,y,z)上显示字符串 string。

`text(x,y,z,'string','PropertyName',PropertyValue,...)`: 在三维图形空间中的指定位置(x,y,z)上显示字符串 string,并且对指定的属性进行设置。

`h=text(...)`: 返回 text 对象的句柄值 h。

`gtext('string')`: 当光标位于一个图形窗口内时,等待用户单击或按下键盘。若单击或按下键盘,则在光标的位置放置给定的文字 string。

`gtext({'string1','string2','string3',...})`: 当光标位于一个图形窗口内时,等待用户单击或按下键盘。若单击或按下键盘,则在光标位置的每个单独行上放置所给定的字符 string1,string2,string3,...。

`gtext({'string1';'string2';'string3';...})`: 当光标位于一个图形窗口内时,等待用户单击或按下键盘。每单击一次或按一次键盘,即放置一个 string,直到放置完为止。

`h=gtext(...)`: 当用户在鼠标指定的位置放置文字 string 后,返回一个 text 图形对象句柄给 h。

## 4. 添加图例

除了给图形添加标题、标注和文本,利用 legend 函数给图形添加注释,它用文本确认每一个数据集,为图形添加图例便于图形的观察和分析。函数的调用格式如下。

`legend`: 对当前图形中所有的图例进行刷新。

`legend('string1','string2',...)`: 用指定的文字 string1,string2,...在当前坐标轴中对所给数据的每一部分显示一个图例。

`legend(h,'string1','string2',...)`: 用指定的文字 string 在一个包含于句柄向量 h 中的图形中显示图例。

`legend(M)`: 用字符矩阵参量 M 的每一行字符串作为标签。

`legend(h,M)`: 用字符矩阵参量 M 的每一行字符串作为标签,给包含于句柄向量 h 中的相应的图形对象加标签。

`legend(M, 'parameter_name', 'parameter_value', ...)`: 用字符矩阵参量 `M` 的每一行字符串作为标签, 并指定其属性值。

`legend(h, M, 'parameter_name', 'parameter_value', ...)`: 用字符矩阵参量 `M` 的每一行字符串作为标签, 给包含于句柄向量 `h` 中的相应的图形对象加标签, 并指定其属性值。

`legend(axes_handle, ...)`: 给由句柄 `axes_handle` 指定的坐标轴显示图例。

`legend('off')` 或 `legend(axes_handle, 'off')`: 给由句柄 `axes_handle` 指定的坐标轴显示图例, 并关闭标注的内容。

`legend('toggle')`, `legend(axes_handle, 'toggle')`: 用双位按钮使图例在关闭与显示之间进行切换。

`legend('hide')`, `legend(axes_handle, 'hide')`: 给由句柄 `axes_handle` 指定的坐标轴显示图例, 并隐藏标注内容。

`legend('show')`, `legend(axes_handle, 'show')`: 给由句柄 `axes_handle` 指定的坐标轴显示图例, 并显示标注内容。

`legend('boxoff')`, `legend(axes_handle, 'boxoff')`: 给由句柄 `axes_handle` 指定的坐标轴显示图例, 并关闭标注图例之外的边框。

`legend('boxon')`, `legend(axes_handle, 'boxon')`: 给由句柄 `axes_handle` 指定的坐标轴显示图例, 并显示标注图例之外的边框。

`legend_handle = legend(...)`: 返回图例的句柄值向量 `legend_handle`。

`legend(..., 'Location', 'location')`: 在指定的位置 `location` 放置图例。

`legend(..., 'Orientation', 'orientation')`: 在指定的方向 `orientation` 放置图例。

**【例 5-8】** 为图形添加修改。

```
>> clear all;
x = 0:0.05 * pi:2 * pi;
plot(x, sin(x), 'r + ', x, cos(x), 'b:');
xlabel('x'); ylabel('y');
title('sinandcos');
legend('sin', 'cos');
gtext('sin(x)');
gtext('cos(x)');
```

运行程序, 效果如图 5-8 所示。

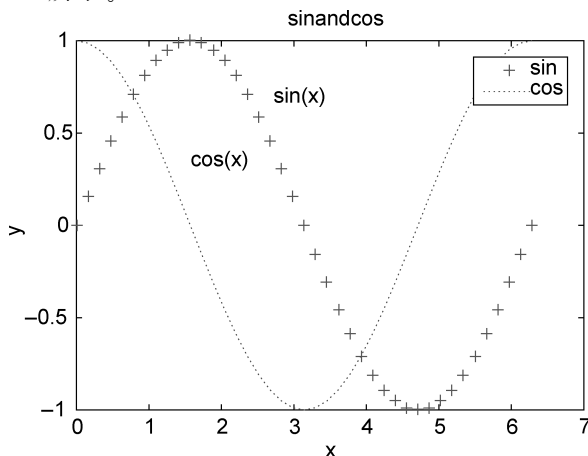


图 5-8 为图形添加修改

5.2.3 特殊二维图形

二维统计分析时常常需要用不同的图形来表示统计结果,如条形图、阶梯图、杆图和填充图等。这时 plot 函数绘制的图形不能满足这些要求,MATLAB 提供了绘制特殊图形的函数,表 5-5 列出了一些常用的绘制特殊二维图形的函数。

表 5-5 特殊二维图形

| 函 数      | 功 能         |
|----------|-------------|
| bar      | 绘制垂直条形图     |
| barh     | 绘制水平条形图     |
| pie      | 绘制饼图        |
| polar    | 绘制极坐标图      |
| errorbar | 绘制沿曲线的误差柱状图 |
| stem     | 绘制火柴杆图      |
| stairs   | 绘制阶梯图       |
| contour  | 绘制等高线图      |

【例 5-9】 绘制各种二维特殊图形。

```
>> clear all;
"绘制条形图"
x1 = -2.9:0.2:2.9;
subplot(2,4,1);bar(x1,exp(-x1.*x1),'r');
title('垂直条形图');

"绘制饼图"
X = [19.3 22.1 51.6; 34.2 70.3 82.4; 61.4 82.9 90.8; 50.5 54.9 59.1;29.4 36.3 47.0];
x2 = sum(X);
explode = zeros(size(x2));
[c,offset] = max(x2);
explode(offset) = 1;           % 为 1 时表示分离扇形,为 0 时表示不分离
subplot(2,4,2);
h = pie(x2,explode);
colormap summer               % 添加颜色效果

"绘制极坐标图"
theta = 0:0.01:2*pi;
rho = sin(2*theta). * cos(2*theta);
subplot(2,4,3);
polar(theta,rho,'--r')
title('极坐标图')

"绘制误差图"
x3 = 0:pi/10:pi;
y1 = sin(x3);
e = std(y1) * ones(size(x3));
subplot(2,4,4);
errorbar(x3,y1,e)
title('误差图')

"绘制火柴杆图"
```

```

x4 = 0:25;
y2 = [exp(-.07 * x4) .* cos(x4); exp(.05 * x4) .* cos(x4)]';
subplot(2,4,5);
h = stem(x4,y2);
title('火柴杆图')

"绘制阶梯图"
X5 = linspace(0,1,30)';
Y3 = [cos(10 * X5), exp(X5) .* sin(10 * X5)];
subplot(2,4,6);
h = stairs(X5,Y3);
title('阶梯图')

"绘制等高线图"
[x,y,z] = peaks(75); % 生成坐标刻度数据和高度数据
subplot(2,4,7);
contour(z);
title('等高线');

```

运行程序,效果如图 5-9 所示。

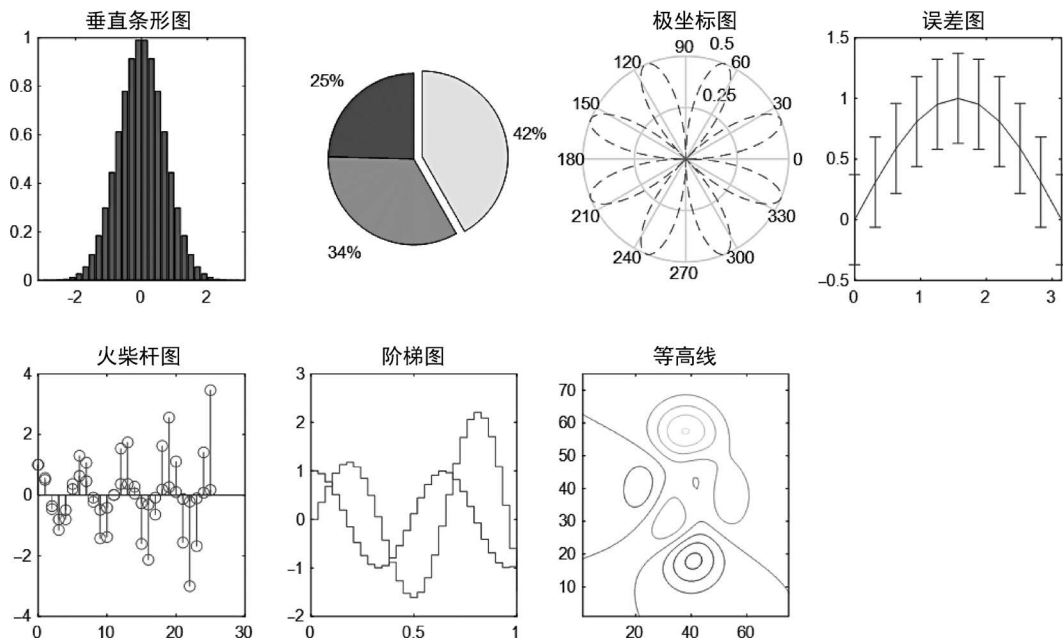


图 5-9 特殊二维图形

### 5.3 函数绘图

如果只知道某个函数的表达式,也可以绘制该函数的图形。函数 `fplot` 用于绘制一元函数的图形,函数 `ezplot` 用于绘制二元函数的图形,函数 `ezpolar` 用于绘制极坐标图形。下面分别给予介绍。

#### 1. `fplot` 函数

`fplot` 函数根据函数的表达式自动调整自变量的范围,无须给函数赋值,直接生成反映函数变化

规律的图形,在函数变化快的区域,采用小的间隔,否则采用大的坐标间隔,使绘制的图形计算量与时间最小,而又能尽可能精确地反映图形的变化。fplot 函数一般在对横坐标的取值间隔没有明确要求,仅在查看函数大致变化规律的情况下使用。函数的调用格式如下。

fplot(fun,limits): 在指定范围 limits=[a,b,c,d]内画出函数名为 fun 的一元函数图形,a、b 为设定的绘图横轴的下限及上限,c、d 是纵轴的下限及上限。

fplot(fun,limits,LineStyle): 用指定的线型 LineSpec 画出函数 fun 的图形。

fplot(fun,limits,tol): 用相对误差值 tol 画出函数 fun 的画形,相对误差的默认值为  $2e-3$ 。

fplot(fun,limits,tol,LineStyle): 用指定的相对误差值 tol 和指定的线型 LineSpec 画出函数 fun 的图形。

fplot(fun,limits,n): 当  $n \geq 1$  时,至少画出  $n+1$  个点(即至少把范围 limits 分成  $n$  个小区间),最大步长不超过  $(x_{\max} - x_{\min})/n$ 。

fplot(fun,limits,tol,n,LineStyle): 允许可选参数 tol、n 和 LineSpec 以任意组合方式输入。

[X,Y]=fplot(fun,limits,...): 返回横坐标与纵坐标的值给变量 X 与 Y,此时 fplot 不画出图形。如需画出图形,可使用 plot(X,Y)语句。

**【例 5-10】** 利用 fplot 函数绘制参数化曲线  $x = \cos(3t)$  和  $y = \sin(2t)$ 。

```
>> clear all;
xt = @(t) cos(3*t);
yt = @(t) sin(2*t);
fplot(xt,yt)
```

运行程序,效果如图 5-10 所示。

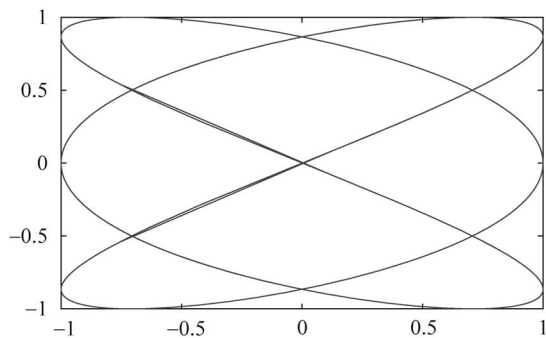


图 5-10 fplot 函数绘图

**注意:** fplot 与 plot 的区别在于,fplot 的指令可以自动地画出一个已定义的函数图形,而无须产生绘图所需要的一组数据作为参数;fplot 采用自适应步长控制来画出函数 fun 的示意图,在函数变化激烈的区间,自动采用小的步长,反之采用大的步长,这样能使计算量与时间最小,并且图形尽可能精确。

## 2. ezpolar 函数

在 MATLAB 中,利用 ezpolar 函数绘制极坐标系下的图形。函数的调用格式如下。

ezpolar(fun): 函数在极坐标系中绘制函数  $\rho = \text{fun}(\theta)$  的图形,变量  $\theta$  默认为开区间  $(0, 2\pi)$ 。

ezpolar(fun,[a,b]): 该函数的变量 theta 在开区间(a,b)上绘制函数  $\rho = \text{fun}(\text{theta})$  的图形。  
 ezpolar(axes\_handle,...): 在指定的图形窗口坐标系 axes\_handle 中绘制极坐标图形。

**【例 5-11】** 利用 ezpolar 函数绘制极坐标系下的图形。

```
>> clear all;
subplot(121);
ezpolar('sin(2 * t) . * cos(3 * t)',[0 pi]);           % 极坐标绘图
subplot(122);
ezpolar(@(t)sin(3 * t),[0 2 * pi]);                 % 极坐标绘图
```

运行程序,效果如图 5-11 所示。

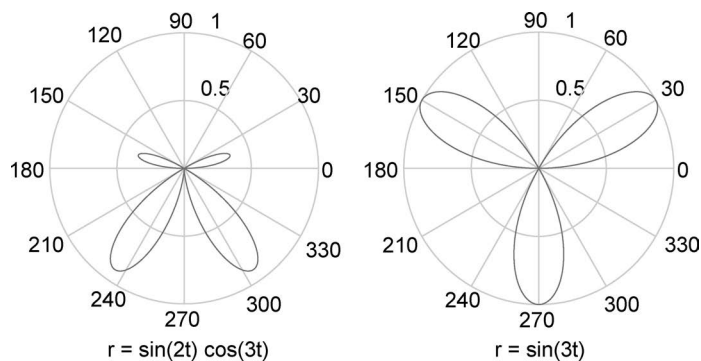


图 5-11 ezpolar 函数绘图

## 5.4 三维基本绘图

本节将介绍 MATLAB 中的各种三维绘图函数。三维绘图函数在很多方面与二维绘图类似,其中曲线的属性设置完全相同。

### 5.4.1 三维曲线图

对应于二维空间的 plot 函数,在三维空间中则是 plot3 函数。函数的调用格式如下。

plot3(X1,Y1,Z1,...): 以默认线型属性绘制三维点集  $(X_i, Y_i, Z_i)$  确定的曲线。 $X_i, Y_i, Z_i$  为相同大小的向量或矩阵。

plot3(X1,Y1,Z1,LineSpec,...): 以参数 LineSpec 确定的线型属性绘制三维点集  $(X_i, Y_i, Z_i)$  确定的曲线, $X_i, Y_i, Z_i$  为相同大小的向量或矩阵。

plot3(...,'PropertyName',PropertyValue,...): 绘制三维曲线,根据指定的属性值设定曲线的属性。

$h = \text{plot3}(\dots)$ : 返回绘制的曲线图的句柄值 h。

**【例 5-12】** 利用 plot3 函数绘制以下参数方程的三维曲线。

$$\begin{cases} x = t \\ y = \cos t \\ z = \sin 2t \end{cases}$$

其实现的 MATLAB 代码如下:

```
>> clear all;
x = 0:0.01:50;
y = cos(x);
z = sin(2 * x);
plot3(x,y,z, 'r-.');
grid on;
title('三维曲线');
```

运行程序,效果如图 5-12 所示。

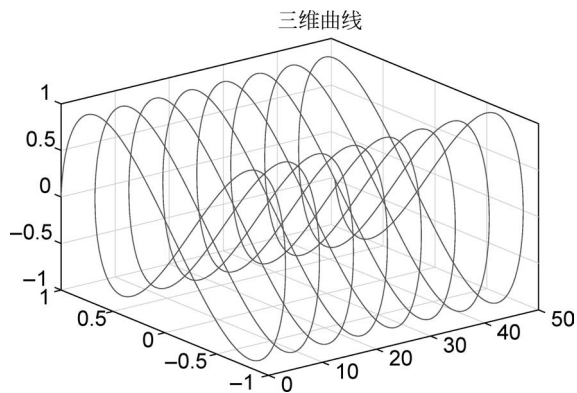


图 5-12 三维曲线图

#### 5.4.2 三维网格图

三维网格图和三维曲面图的绘制比三维曲线图的绘制稍显复杂,主要是因为绘图数据的准备及三维图形的色彩、明暗、光照和视角等的处理。绘制函数  $z=f(x,y)$  的三维网格图的过程如下。

(1) 确定自变量  $x$  和  $y$  的取值范围和取值间隔。

```
x = x1:dx:x2;
y = y1:dy:y2;
```

(2) 构成  $xOy$  平面上的自变量采样“格点”矩阵。

① 利用“格点”矩阵的原理生成矩阵。

```
x = x1:dx:x2;
y = y1:dy:y2;
X = ones(size(y)) * x;
Y = y * ones(size(x));
```

② 利用 meshgrid 函数生成“格点”矩阵。

```
x = x1:dx:x2;
y = y1:dy:y2;
[X,Y] = meshgrid(x,y);
```

(3) 计算在自变量采样“格点”上的函数值:  $z=f(x,y)$ 。

在 MATLAB 中,提供了 mesh 函数用于绘制三维网格图。函数的调用格式如下。

mesh(X,Y,Z): 绘制三维网格图,颜色和曲面的高度相匹配,如果 X 与 Y 为向量,且  $\text{length}(X)=n, \text{length}(Y)=m, [m,n]=\text{size}(Z)$ , 空间中的点  $(X(j), Y(i), Z(i,j))$  为所画曲面网线的交点; 如果 X 与 Y 均为矩阵,则空间中的点  $(X(i,j), Y(i,j), Z(i,j))$  为所画曲面网线的交点。

mesh(Z): 参数 Z 是维数为  $m \times n$  的矩阵,网格曲面的颜色分布与 Z 方向上的高度值成正比。

mesh(...,C): 参数 X、Y、Z 都为矩阵值,参数 C 表示网格曲面的颜色分布情况。

mesh(..., 'PropertyName', PropertyValue, ...): 对指定的属性 PropertyName 设置属性值 PropertyValue,可以在同一语句中对多个属性进行设置。

$h = \text{mesh}(\dots)$ : 返回图形对象的句柄值 h。

另外, MATLAB 中还有两个 mesh 的派生函数。

(1) meshc 在绘制曲面图的同时,在 x-y 平面上绘制函数的等值线。

(2) meshz 在曲面图基础上,在图形的底部外侧绘制平行 z 轴的边框线。

**【例 5-13】** 利用 mesh、meshz 及 meshc 绘制三维网格图。

```
>> clear all;
[X,Y] = meshgrid(-3:1.25:3);
Z = peaks(X,Y);
subplot(1,3,1);mesh(X,Y,Z);
title('mesh 绘图')
axis([-3 3 -3 3 -10 5])
subplot(1,3,2);meshc(X,Y,Z);
title('meshc 绘图')
axis([-3 3 -3 3 -10 5])
subplot(1,3,3);meshz(X,Y,Z);
title('meshz 绘图')
axis([-3 3 -3 3 -10 5])
set(gcf, 'color', 'w');
```

运行程序,效果如图 5-13 所示。

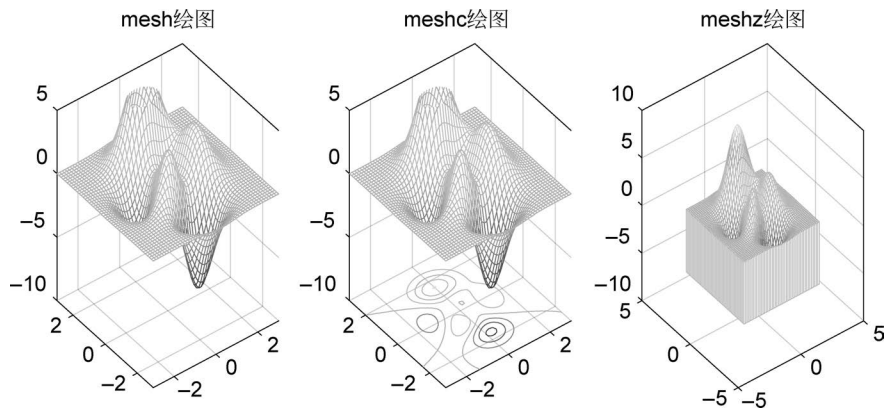


图 5-13 三维网格图

### 5.4.3 三维曲面图

三维曲面图也可以用来表示三维空间内数据的变化规律,与之前讲述的三维网格图的不同之处



在于,对网格的区域填充了不同的色彩。在 MATLAB 中绘制三维曲面图的函数主要有 surf 函数、surfc 函数、surfl 函数。surfl 函数用于产生一个带有阴影效果的曲面,该阴影效果是基于环绕、分散、特殊光照等模型得到的。surfc 函数是在 surf 函数的基础上添加曲面的等高线。surfc 函数和 surfl 函数的用法与 surf 函数类似。surf 函数的调用格式如下。

surf(X,Y,Z): 该函数绘制彩色的三维曲面图,其中矩阵 X 和 Y 控制 x 轴和 y 轴,矩阵 Z 为 z 轴数据。

surf(X,Y,Z,C): 图形的颜色采用参数 C 设置。

surf(Z)或 surf(Z,C): 该函数默认向量 x 为 1:n,向量 y 为 1:m,其中[m,n]=size(Z)。

surf(...,'PropertyName',PropertyValue): 该函数中对图形的一些属性值进行设置。

h=surf(...): 返回三维表面图的句柄值 h。

此外,在 MATLAB 中,采用函数 surfc 绘制带有等高线的三维曲面图,该函数的调用格式和 surf 相同。通过函数 surfl 添加三维曲面的光照效果,该函数和 surf 函数的调用格式相同。

**【例 5-14】** 利用 surf 函数,绘制旋转抛物面  $z=x^3+y^3$  的图形。

```
>> clear all;
x = -2:0.05:2;
y = -2:0.05:2;
[X,Y] = meshgrid(x,y);           % 生成数据点矩阵 X 与 Y
z = X.^3 + Y.^3;
surf(X,Y,z)                      % 绘制曲面图
grid on;                         % 绘制网格线
shading flat                      % 阴影效果
```

运行程序,效果如图 5-14 所示。

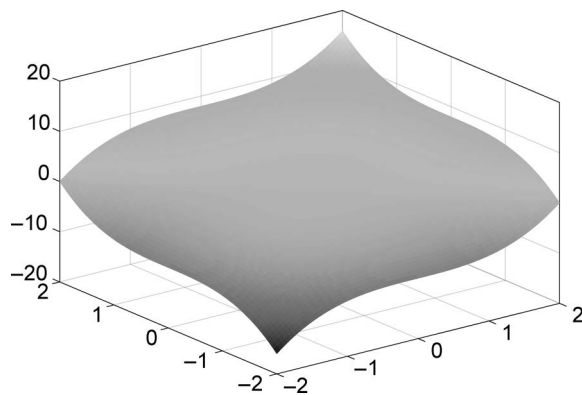


图 5-14 三维曲面图

#### 5.4.4 特殊三维图形

在科学研究中,有时也需要绘制一些特殊的三维图形,如统计学中三维直方图、圆柱体图、饼形图等。MATLAB 中提供了用于绘制这些特殊三维图形的函数,下面通过一个实例来演示特殊三维图形的效果。

**【例 5-15】** 绘制各种特殊三维图形。

```

>> clear all;
t = 0:pi/10:2*pi;
[X1,Y1,Z1] = cylinder(2+cos(t));
subplot(231);surf(X1,Y1,Z1)
axis square;title('三维柱面图');
subplot(232);sphere
axis equal;title('三维球体');
x1 = [1 3 0.5 2.5 2];
explode = [0 1 0 0 0];
subplot(233);pie3(x1,explode)
title('三维饼图');axis equal;
X2 = [0 1 1 2;1 1 2 2;0 0 1 1];
Y2 = [1 1 1 1;1 0 1 0;0 0 0 0];
Z2 = [1 1 1 1;1 0 1 0;0 0 0 0];
C = [0.5000 1.0000 1.0000 0.5000;
     1.0000 0.5000 0.5000 0.1667;
     0.3330 0.3330 0.5000 0.5000];
subplot(234);fill3(X2,Y2,Z2,C);
colormap hsv
title('三维填充图');axis equal;
[x2,y2] = meshgrid(-3:.5:3,-3:.1:3);
z2 = peaks(x2,y2);
subplot(235);ribbon(y2,z2)
colormap hsv
title('三维彩带图');axis equal;
[X3,Y3] = meshgrid(-2:0.25:2,-1:0.2:1);
Z3 = X3.*exp(-X3.^2-Y3.^2);
[U,V,W] = surfnorm(X3,Y3,Z3);
subplot(236);quiver3(X3,Y3,Z3,U,V,W,0.5);
hold on
surf(X3,Y3,Z3);
colormap hsv
view(-35,45);
title('三维向量场图');axis equal;
set(gcf,'color','w'); %设置背景色为白色

```

运行程序,效果如图 5-15 所示。

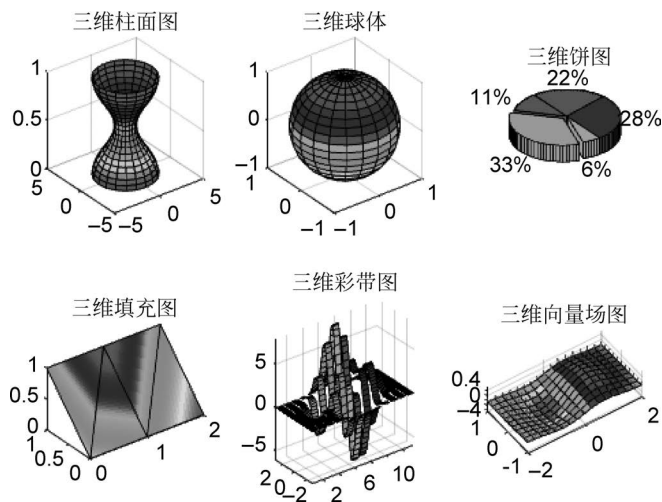


图 5-15 特殊三维图形



彩色图片

## 5.5 四维绘图

人对自然界的理解和思考是多维的。人的感官不仅善于接受一维、二维、三维的几何信息,而且对几何物体的运动、颜色、声音、气味、触感等反应灵敏。从某种意义上来说,MATLAB 色彩控制、动画等指令,为四维或更高维表现提供了手段。

### 5.5.1 用色彩表现函数特征

下面直接通过一个例子来演示色彩表现函数的特征对图形的影响。

**【例 5-16】** 当三维网线图、曲面图的第四个输入变量取一些特殊矩阵时,色彩就能表现或加强函数的某些特征,如梯度、曲率、方向导数等。

```
>> clear all;
x = 3 * pi * (-1:1/15:1); y = x;
[X,Y] = meshgrid(x,y);
r = sqrt(X.^2 + Y.^2) + eps;
Z = sin(r) ./ r;
[dx,dy] = gradient(Z); % 计算对 r 全导数
dz = sqrt(dx.^2 + dy.^2); % 计算曲率
dz2 = del2(Z);
subplot(1,2,1); surf(X,Y,Z);
title('surf(X,Y,Z)效果图');
shading faceted, colorbar('horiz');
brighten(0.3);
subplot(1,2,2); surf(X,Y,Z,r);
title('surf(X,Y,Z,r)效果图');
shading faceted, colorbar('horiz');
set(gcf, 'color', 'w');
```

运行程序,效果如图 5-16 所示。

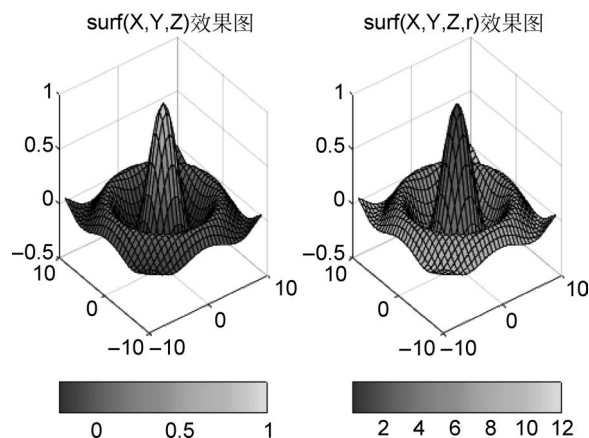


图 5-16 色彩表现函数的高度与半径特征

如果想要改变表现函数 X 方向与 Y 方向的导数特征,其实现的 MATLAB 代码如下:



彩色图片

```
>> subplot(1,2,1);surf(X,Y,Z,dx);
shading faceted, colorbar('horiz');
brighten(0.25);
title('surf(X,Y,Z,dx)X方向特征导数图')
subplot(1,2,2);surf(X,Y,Z,dy);
shading faceted, colorbar('horiz');
title('surf(X,Y,Z,dy)Y方向特征导数图')
```

运行程序,效果如图 5-17 所示。

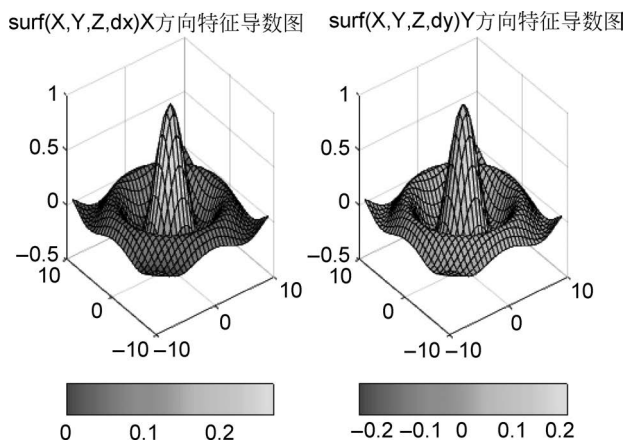


图 5-17 色彩表现 X 与 Y 方向特征导数图

如果想要用色彩表现函数的径向导数与曲率特征,其实现的 MATLAB 代码如下:

```
>> subplot(1,2,1);surf(X,Y,Z,abs(dz));
shading faceted, colorbar('horiz');
brighten(0.5);
title('surf(X,Y,Z,abs(dz))径向导数图')
subplot(1,2,2);surf(X,Y,Z,dz2);
shading faceted, colorbar('horiz');
title('surf(X,Y,Z,dz2)曲率图')
```

运行程序,效果如图 5-18 所示。

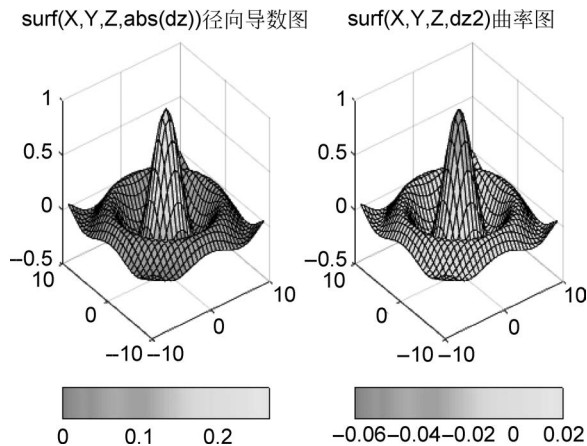


图 5-18 色彩表现函数的径向导数与曲率特征



彩色图片

### 5.5.2 切片图与切片等位线图

前面介绍的用色彩表现函数特征绘制效果,仅仅表现出了三维函数的某些特征,还不能充分表现出四维图形的效果。在 MATLAB 中提供了 meshgrid、slice、contourslice 函数,可充分表现四维图形的效果。meshgrid 函数在 5.4.2 节中已经介绍过,下面只对后面两个函数进行简单的介绍。

#### 1. slice 函数

函数 slice 用于绘制三维切片图。三维切片图可形象地称为“四维图”,可以在三维空间内表达第四维的信息,用颜色来标识第四维数据的大小。函数的调用格式如下。

slice(X,Y,Z,V,xslice,yslice,zslice): 为三维体数据 V 绘制切片。指定 X、Y 和 Z 作为坐标数据。使用以下形式之一指定 xslice、yslice 和 zslice 作为切片位置。

- 要绘制一个或多个与特定轴正交的切片平面,需将切片参数指定为标量或向量。
- 要沿曲面绘制单个切片,需将所有切片参数指定为定义曲面的矩阵。

slice(V,xslice,yslice,zslice): 使用 V 的默认坐标数据。V 中每个元素的(x,y,z)位置分别基于列、行和页面索引。

slice(\_\_\_\_,method): 指定插值方法,其中 method 可以是'linear'(默认值)、'cubic'或'nearest'。可将此选项与上述语法中的任何输入参数一起使用。

slice(ax,\_\_\_\_): 在指定坐标区而不是当前坐标区(gca)中绘图。

s=slice(\_\_\_\_): 返回创建的 Surface 对象。slice 为每个切片返回一个 Surface 对象。

**【例 5-17】** 利用 slice 函数绘制无限大小水体中水下射流速度数据 flow。

```
>> clear all;
[X,Y,Z,V] = flow;                                % 取 4 个 (50×25×25) 的射流数据矩阵,V 是射流速度
x1 = min(min(min(X))); x2 = max(max(max(X)));    % 取 X 坐标上下限
y1 = min(min(min(Y))); y2 = max(max(max(Y)));    % 取 Y 坐标上下限
z1 = min(min(min(Z))); z2 = max(max(max(Z)));    % 取 Z 坐标上下限
sx = linspace(x1+1.5,x2,5);                      % 确定 5 个垂直 X 轴的切面坐标
sy = 0;                                           % 在 y = 0 处,取垂直 y 轴的切面
sz = 0;                                           % 在 z = 0 处,取垂直 z 轴的切面
slice(X,Y,Z,V,sx,sy,sz);
view([-38,38]); shading interp;
colormap jet;
set(gcf,'color','w');
```

运行程序,效果如图 5-19 所示。

#### 2. contourslice 函数

contourslice 函数用来实现三元函数切面等高线的效果图。函数的调用格式如下。

contourslice(X,Y,Z,V,Sx,Sy,Sz): X、Y、Z 是维数为  $m \times n \times p$  的自变量“格点”数组; V 是与 X、Y、Z 同维的函数值数组; Sx、Sy、Sz 是决定切片位置的数值向量。假如取“空阵”,就表示不取切片。

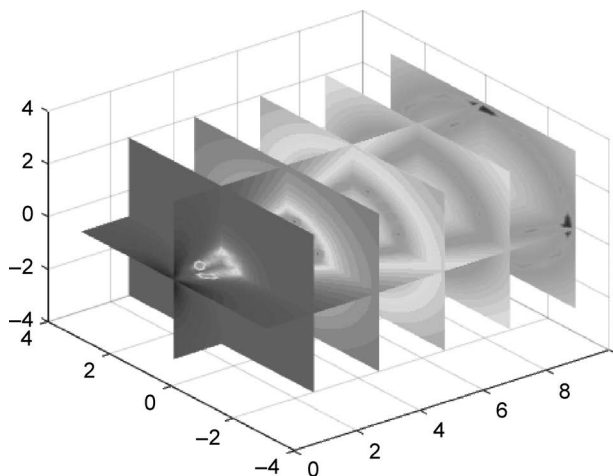


图 5-19 切片效果图

**【例 5-18】** 使用 `contourslice` 函数进行 `flow` 设置,展示飞机上使用切片轮廓数据的效果。

```
>> clear all;
[x y z v] = flow;
h = contourslice(x,y,z,v,[1:9],[],[0],linspace(-8,2,10));
axis([0,10,-3,3,-3,3]); daspect([1,1,1]);
camva(24); camproj perspective;
campos([-3,-15,5]);
set(gcf,'Color',[.5,.5,.5],'Renderer','zbuffer');
set(gca,'Color','black','XColor','white','YColor','white','ZColor','white');
colormap jet; colorbar;
box on;
set(gcf,'color','w');
```

运行程序,效果如图 5-20 所示。

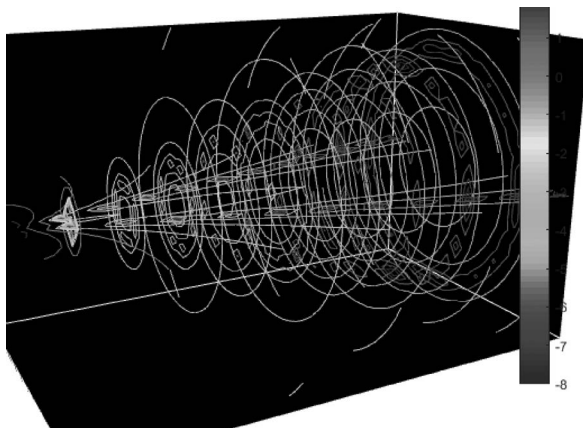


图 5-20 切片等位线效果图