第 5 章

CHAPTER 5

MATLAB 数值计算

本章要点：

- 多项式运算；
- 数据插值；
- 多项式拟合；
- 数据统计；
- 数值计算。

5.1 多项式

多项式在代数中占有重要的地位，广泛用于数据插值、数据拟合和信号与系统等应用领域。MATLAB 提供了各种多项式的创建以及运算方法，应用起来简单方便。

5.1.1 多项式的创建

一个多项式按降幂排列为

$$p(x)=a_n x^n+a_{n-1}x^{n-1}+\cdots+a_1 x+a_0 \tag{5-1}$$

在 MATLAB 中多项式的各项系数用一个行向量表示，使用长度为 $n+1$ 的行向量按降幂排列，多项式中某次幂的缺项用 0 表示，则表示为

$$p=[a_n,a_{n-1},\cdots,a_1,a_0] \tag{5-2}$$

例如，多项式 $p_1(x)=x^3-2x^2+4x+6$，在 MATLAB 可以表示为 $p_1=[1,-2,4,6]$；$p_2(x)=x^3+3x+6$ 可表示为 $p_2=[1,0,3,6]$。

在 MATLAB 中，创建一个多项式，可以用 poly2str、poly2sym 函数实现，其调用格式如下：

```
f = poly2str(p,'x')        % p 为多项式的系数,x 为多项式的变量
f = poly2sym(p)            % p 为多项式的系数
```

其中，f=poly2str(p,'x')表示创建一个系数为 p，变量为 x 的字符串型多项式；f=poly2sym(p)表示创建一个系数为 p，默认变量为 x 的符号型多项式。两者在命令窗口显示形式类似，但数据类型是不一样的，一个是字符串型，另一个是符号型。

【例 5-1】 已知多项式系数为 $p=[1,-2,4,6]$，分别用 poly2str(p,'x')和 poly2sym(p)创建多项式，比较它们有什么不同。

微课视频

程序代码如下：

```
>> p = [1 -2 4 6]
p =
     1    -2     4     6
>> f1 = poly2str(p,'x')
f1 =
    '   x^3 - 2 x^2 + 4 x + 6'
>> f2 = poly2sym(p)
f2 =
x^3 - 2*x^2 + 4*x + 6
```

显然，两种函数创建的多项式 f1 和 f2 显示形式类似，但数据类型和大小都不一样，如图 5-1 所示。

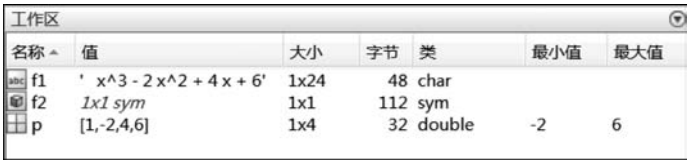

图 5-1　两种多项式的比较

5.1.2　多项式的值和根

1. 多项式的值

在 MATLAB 里，求多项式的值可以用 polyval 和 polyvalm 函数。它们的输入参数都是多项式系数和自变量，二者的区别是前者为代数多项式求值，后者为矩阵多项式求值。

(1) 代数多项式求值

polyval 函数可以求解代数多项式的值，其调用格式如下：

```
y = polyval(p,x)
```

其中，p 为多项式的系数，x 为自变量。当 x 为一个数值时，求解的是多项式在该点的值；若 x 为向量或矩阵，则是对向量或矩阵每个元素求多项式的值。


微课视频

【例 5-2】 已知多项式 $f(x)=x^3-2x^2+4x+6$，分别求 $x1=2$ 和 $\boldsymbol{x}=[0,2,4,6,8,10]$ 向量的多项式的值。

在文件编辑窗口编写命令文件，保存为 exam_5_2.m 脚本文件。程序代码如下：

```
x1 = 2;
x = [0:2:10];
p = [1 -2 4 6];
y1 = polyval(p,x1)
y = polyval(p,x)
```

在命令空间输入文件名 exam_5_2.m，就能直接运行该脚本文件。结果如下：

```
>> exam_5_2
y1 =
    14
```

```
y =
     6    14    54   174   422   846
```

(2) 矩阵多项式求值

polyvalm 函数是以矩阵为自变量求解多项式的值,其调用格式如下:

```
Y = polyvalm(p,X)
```

其中,p 为多项式系数,X 为自变量,自变量要求为方阵。

因为运算规则不一样,所以 MATLAB 用 polyvalm 和 polyval 函数求解多项式的值是不一样的。例如,假设 A 为方阵,p 为多项式 x^2-5x+6 的系数,则 polyvalm(p,A)表示 A*A−5*A+6*eye(size(A)),而 polyval(p,A)表示 A.*A−5*A+6*ones(size(A))。

【例 5-3】 已知多项式为 $f(x)=x^2-3x+2$,分别用 polyvalm 和 polyval 函数,求 $\boldsymbol{X}=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$ 的多项式值。


微课视频

在文件编辑窗口编写命令文件,保存为 exam_5_3.m 脚本文件。程序代码如下:

```
X = [1 2;3 4];
p = [1 -3 2];
Y = polyvalm(p,X)
Y1 = polyval(p,X)
```

程序运行结果:

```
>> exam_5_3
Y =
     6     4
     6    12
Y1 =
     0     0
     2     6
```

2. 多项式的根

一个 n 次多项式有 n 个根,这些根有可能是实根,也有可能包含若干对共轭复根。MATLAB 提供了 roots 函数用于求解多项式的全部根,其调用格式为

```
r = roots(p)
```

其中,p 为多项式的系数向量,r 为多项式的根向量,r(1),r(2),…,r(n)分别表示多项式的 n 个根。

MATLAB 还提供一个由多项式的根求多项式的系数的函数 poly,其调用格式如下:

```
p = poly(r)
```

其中,r 为多项式的根向量,p 为由根 r 构造的多项式系数向量。

【例 5-4】 已知多项式为 $f(x)=x^4+4x^3-3x+2$。

(1) 用 roots 函数求该多项式的根 r。

(2) 用 poly 函数求根为 r 的多项式系数。


微课视频

在文件编辑窗口编写命令文件,保存为 exam_5_4.m 脚本文件。程序代码如下:

```
p = [1 4 0 -3 2];
r = roots(p)
p1 = poly(r)
```

程序运行结果：

```
>> exam_5_4
r =
  -3.7485 + 0.0000i
  -1.2962 + 0.0000i
   0.5224 + 0.3725i
   0.5224 - 0.3725i
p1 =
    1.0000    4.0000   -0.0000   -3.0000    2.0000
```

显然，roots 和 poly 函数的功能正好相反。

5.1.3 多项式的四则运算

多项式之间可以进行四则运算，其结果仍为多项式。在 MATLAB 中，用多项式系数向量进行四则运算，得到的结果也显示为多项式系数向量。

1. 多项式的加减运算

MATLAB 没有提供多项式加减运算的函数。事实上多项式的加减运算，是合并同类项，可以用多项式系数向量进行加减运算。如果多项式阶次不同，则把低次多项式系数缺少的高次项用 0 补足，使得多项式系数矩阵具有相同维度，以便实现加减运算。

2. 多项式乘法运算

在 MATLAB 中，两个多项式的乘积可以用 conv 函数实现。其调用格式为

```
p = conv(p1,p2)
```

其中，p1 和 p2 是两个多项式的系数向量；p 是两个多项式乘积的系数向量。

3. 多项式除法运算

MATLAB 用 deconv 函数实现两个多项式的除法运算。其调用格式为

```
[q,r] = deconv(p1,p2)
```

其中，q 为多项式 p1 除以 p2 的商式；r 为多项式 p1 除以 p2 的余式。q 和 r 都是多项式系数向量。

deconv 是 conv 的逆函数，因此满足 p1＝conv(p2,q)＋r。

微课视频

【例 5-5】 已知两个多项式为 $f(x)=x^4+4x^3-3x+2$，$g(x)=x^3-2x^2+x$。

(1) 求两个多项式相加 $f(x)+g(x)$，两个多项式相减 $f(x)-g(x)$的结果。

(2) 求两个多项式相乘 $f(x)*g(x)$，两个多项式相除 $f(x)/g(x)$的结果。

在文件编辑窗口编写命令文件，保存为 exam_5_5.m 脚本文件。程序代码如下：

```
p1 = [1 4 0 -3 2];
p2 = [0 1 -2 1 0];
p3 = [1 -2 1 0];
p = p1 + p2                    % f(x) + g(x)
```

```
poly2sym(p)
p = p1 - p2                    % f(x) - g(x)
poly2sym(p)
p = conv(p1,p2)                % f(x) * g(x)
poly2sym(p)
[q,r] = deconv(p1,p3)          % f(x)/g(x)
p4 = conv(q,p3) + r            % 验证 deconv 是 conv 的逆函数
```

程序运行结果：

```
>> exam_5_5
p =
     1     5    -2    -2     2
ans =
x^4 + 5*x^3 - 2*x^2 - 2*x + 2
p =
     1     3     2    -4     2
ans =
x^4 + 3*x^3 + 2*x^2 - 4*x + 2
p =
     0     1     2    -7     1     8    -7     2     0
ans =
x^7 + 2*x^6 - 7*x^5 + x^4 + 8*x^3 - 7*x^2 + 2*x
q =
     1     6
r =
     0     0    11    -9     2
p4 =
     1     4     0    -3     2
```

5.1.4　多项式的微积分运算

1. 多项式的微分

对 n 阶多项式 $p(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$ 求导，其导数为 $n-1$ 阶多项式 $dp(x)=na_nx^{n-1}+(n-1)a_{n-1}x^{n-2}+\cdots+a_1$。原多项式及其导数多项式的系数分别为 $p=[a_n,a_{n-1},\cdots,a_1,a_0]$，$d=[na_n,(n-1)a_{n-1},\cdots,a_1]$。

在 MATLAB 中，可以用 polyder 函数来求多项式的微分运算，polyder 函数可以对单个多项式求导，也可以对两个多项式的乘积或商求导，其调用格式如下：

```
p = polyder(p1)                % 求多项式 p1 的导数
p = polyder(p1,p2)             % 求多项式 p1 × p2 的积的导数
[p,q] = polyder(p1,p2)         % p1 ÷ p2 的导数，p 为导数的分子多项式系数，q 为导数的分母多
                               % 项式系数
```

【例 5-6】 已知两个多项式为 $f(x)=x^4+4x^3-3x+2$，$g(x)=x^3-2x^2+x$。

(1) 求多项式 $f(x)$的导数。

(2) 求两个多项式乘积 $f(x)*g(x)$的导数。

(3) 求两个多项式相除 $g(x)/f(x)$的导数。

微课视频

在文件编辑窗口编写命令文件，保存为 exam_5_6.m 脚本文件。程序代码如下：

```
p1 = [1 4 0 -3 2];
p2 = [1 -2 1 0];
p = polyder(p1)
poly2sym(p)
p = polyder(p1,p2)
poly2sym(p)
[p,q] = polyder(p2,p1)
```

程序运行结果：

```
>> exam_5_6
p =
     4    12     0    -3
ans =
4*x^3 + 12*x^2 - 3
p =
     7    12   -35     4    24   -14     2
ans =
7*x^6 + 12*x^5 - 35*x^4 + 4*x^3 + 24*x^2 - 14*x + 2
p =
    -1     4     5   -14    12    -8     2
q =
     1     8    16    -6   -20    16     9   -12     4
```

2. 多项式的积分

对于 n 阶多项式 $p(x)=a_n x^n+a_{n-1}x^{n-1}+\cdots+a_1 x+a_0$，其不定积分为 $n+1$ 阶多项式 $i(x)=\frac{1}{n+1}a_n x^{n+1}+\frac{1}{n}a_{n-1}x^n+\cdots+\frac{1}{2}a_1 x^2+a_0 x+k$，其中 k 为常数项。原多项式和积分多项式分别可以表示为系数向量 $\boldsymbol{p}=[a_n,a_{n-1},\cdots,a_1,a_0]$，$\boldsymbol{I}=\left[\frac{1}{n+1}a_n,\frac{1}{n}a_{n-1},\cdots,\frac{1}{2}a_1,k\right]$。

在 MATLAB 中，提供了 polyint 函数用于多项式的积分。其调用格式为

```
I = polyint(p,k)                    % 求以 p 为系数的多项式的积分，k 为积分常数项
I = polyint(p)                      % 求以 p 为系数的多项式的积分，积分常数项为默认值 0
```

显然 polyint 是 polyer 的逆函数，因此有 p=polyder(I)。

微课视频

【例 5-7】 求多项式的积分 $I=\int(x^4+4x^3-3x+2)\mathrm{d}x$。

在文件编辑窗口编写命令文件，保存为 exam_5_7.m 脚本文件。程序代码如下：

```
p = [1 4 0 -3 2];
I = polyint(p)                      % 求多项式的积分，常数项为默认的 0
poly2sym(I)                         % 显示多项式积分的多项式
p = polyder(I)                      % 验证 polyint 是 polyder 的逆函数
syms k                              % 定义常数项 k
I1 = polyint(p,k)                   % 求多项式的积分，常数项为 k
poly2sym(I1)
```

程序运行结果：

```
>> exam_5_7
I =
    0.2000    1.0000         0   -1.5000    2.0000         0
ans =
x^5/5 + x^4 - (3*x^2)/2 + 2*x
p =
     1     4     0    -3     2
I1 =
[ 1/5, 1, 0, -3/2, 2, k]
ans =
x^5/5 + x^4 - (3*x^2)/2 + 2*x + k
```

5.1.5 多项式的部分分式展开

由分子多项式 $B(s)$和分母多项式 $A(s)$构成的分式表达式进行多项式的部分分式展开，表达式如下：

$$\frac{B(s)}{A(s)}=\frac{r_1}{s-p_1}+\frac{r_2}{s-p_2}+\cdots+\frac{r_n}{s-p_n}+k(s) \tag{5-3}$$

MATLAB 可以用 residue 函数实现多项式的部分分式展开，residue 函数的调用格式如下：

```
[r,p,k] = residue(B,A)
```

其中，B 为分子多项式系数行向量；A 为分母多项式系数行向量；[p_1；p_2；…；p_n]为极点列向量；[r_1；r_2；…；r_n]为零点列向量；k 为余式多项式行向量。

residue 函数还可以将部分分式展开式转换为两个多项式的分式表达式，其调用格式为：

```
[B,A] = residue(r,p,k)
```

【例 5-8】 已知分式表达式为 $f(s)=\dfrac{B(s)}{A(s)}=\dfrac{3s^3+1}{s^2-5s+6}$。


微课视频

(1) 求 $f(s)$的部分分式展开式。

(2) 将部分分式展开式转换为分式表达式。

在文件编辑窗口编写命令文件，保存为 exam_5_8.m 脚本文件。程序代码如下：

```
a = [1 -5 6];
b = [3 0 0 1];
[r,p,k] = residue(b,a)                    % 部分分式展开
[b1,a1] = residue(r,p,k)                  % 将部分分式展开转换为分式表达式
```

程序运行结果：

```
>> exam_5_8
r =
    82.0000
   -25.0000
```

```
p =
    3.0000
    2.0000
k =
     3     15
b1 =
     3      0      0      1
a1 =
     1     -5      6
```

5.2 数据插值

在工程测量与科学实验中，得到的数据通常都是离散的。如果要得到这些离散数据点以外的其他数据值，就需要根据这些已知数据进行插值。假设已测量得到 n 个点数据，(x_1,y_1)、(x_2y_2)、…、(x_ny_n)，且这些测量值满足某一个未知的函数关系 $y=f(x)$。数据插值的任务就是根据这 n 个测量数据，构造一个函数 $y=p(x)$，使得 $y_i=p(x_i)(i=1,2,\cdots,n)$成立，称 $p(x)$为 $f(x)$关于点 $x_1,x_2,\cdots,x_n$ 的插值函数。求插值函数 $p(x)$的方法为插值法。插值函数 $p(x)$一般可以用线性函数、多项式或样条函数实现。

根据插值函数的自变量个数，数据插值可以分为一维插值、二维插值和多维插值等；根据不同的插值函数，又可以分为线性插值、多项式插值和样条函数插值等。MATLAB 提供了一维插值 interp1、二维插值 interp2、三维插值 interp3 和 N 维插值 interpn 函数，以及三次样条插值 spline 函数等。

5.2.1 一维插值

所谓一维插值是指被插值函数的自变量是一个单变量的函数。一维插值采用的方法一般有一维多项式插值、一维快速插值和三次样条插值。

1. 一维多项式插值

MATLAB 中提供了 interp1 函数进行一维多项式插值。interp1 函数使用了多项式函数，通过已知数据点，计算目标插值点的数据。interp1 函数调用格式如下：

```
yi = interp1(Y,xi)
```

其中，Y 是在默认自变量 x 选为 1：n 的值。

```
yi = interp1(X,Y,xi)
```

其中 X 和 Y 是长度一样的已知向量数据，x_i 可以是一个标量，也可以是向量。

```
yi = interp1(X,Y,xi,'method')
```

其中，method 是插值方法，其取值有下面几种：

(1) linear 线性插值：这是默认插值方法，它将与插值点靠近的两个数据点直线连接，在直线上选取对应插值点的数据。这种插值方法兼顾速度和误差，插值函数具有连续性，但平滑性不好。

（2）nearest 最邻近点插值：根据插值点和最接近的已知数据点进行插值，这种插值方法速度快，占用内存小，但一般误差最大，插值结果最不平滑。

（3）next 下一点插值：根据插值点和下一点的已知数据点插值，这种插值方法的优缺点与最邻近点插值一样。

（4）previous 前一点插值：根据插值点和前一点的已知数据点插值，这种插值方法的优缺点与最邻近点插值一样。

（5）spline 三次样条插值：采用三次样条函数获得插值点数据，要求在各点处具有光滑条件。这种插值方法连续性好，插值结果最光滑，缺点为运行时间长。

（6）cubic 三次多项式插值：根据已知数据求出一个 3 次多项式进行插值。这种插值方法连续性好，光滑性较好，缺点是占用内存多，速度较慢。

需要注意，x_i 的取值如果超出已知数据 X 的范围，就会返回 NaN 错误信息。

MATLAB 还提供 interp1q 函数用于一维插值。它与 interp1 函数的主要区别是，当已知数据不是等间距分布时，interp1q 插值速度比 interp1 快。需要注意，interp1q 执行的插值数据 x 必须是单调递增的。

【例 5-9】 某气象台对当地气温进行测量，实测数据见表 5-1 所示，用不同插值方法计算 $t=12$ 时的气温。


微课视频

表 5-1 某地不同时间的气温

测量时间 t（小时）	6	8	10	14	16	18	20
温度 T（度）	16	17.5	19.3	22	21.2	19.5	18

在文件编辑窗口编写命令文件，保存为 exam_5_9.m 脚本文件。程序代码如下：

```
t = [6 8  10  14  16  18  20];              % 测量时间 t
T = [16 17.5 19.3 22  21.2 19.5 18];        % 测量的温度 T
t1 = 12;                                    % 插值点时间 t1
T1 = interp1(t,T,t1,'nearest')              % 最接近点插值
T2 = interp1(t,T,t1,'linear')               % 线性插值
T3 = interp1(t,T,t1,'next')                 % 下一点插值
T4 = interp1(t,T,t1,'previous')             % 前一点插值
T5 = interp1(t,T,t1,'pchip')                % 三次多项式插值
T6 = interp1(t,T,t1,'spline')               % 三次样条插值
```

程序运行结果：

```
>> exam_5_9
T1 =
    22
T2 =
   20.6500
T3 =
    22
T4 =
   19.3000
T5 =
```

```
   21.0419
T6 =
   21.1193
```


微课视频

【例 5-10】 假设测量的数据来自函数 $f(x)=e^{-0.5x}\sin x$，试根据生成的数据，用不同方法进行插值，比较插值结果。

在文件编辑窗口编写命令文件，保存为 exam_5_10.m 脚本文件。程序代码如下：

```
clear
x = (0:0.4:2 * pi)';
y = exp( - 0.5 * x). * sin(x);                    % 生成测试数据
x1 = (0:0.1:2 * pi)';                             % 插值点
y0 = exp( - 0.5 * x1). * sin(x1);                 % 插值点真实值
y1 = interp1(x,y,x1,'nearest');                   % 最接近点插值
disp('interp1 函数插值时间');tic
y2 = interp1(x,y,x1); toc;                        % interp1 插值时间
y3 = interp1(x,y,x1,'spline') ;                   % 三次样条插值
disp('interp1q 函数插值时间');tic
yq = interp1q(x,y,x1);toc;                        % interp1q 插值时间
plot(x1,y1,'--',x1,y2,'-',x1,y3,'-.',x,y,'*',x1,y0,':')
legend('nearest 插值数据','linear 插值数据','spline 插值数据',...
'样本数据点','插值点真实数据')
max(abs(y0 - y3))
```

程序运行结果如下，插值效果如图 5-2 所示。

```
>> exam_5_10
interp1 函数插值时间
历时 0.001270 秒。
interp1q 函数插值时间
历时 0.001472 秒。
ans =
   6.5673e - 04
```

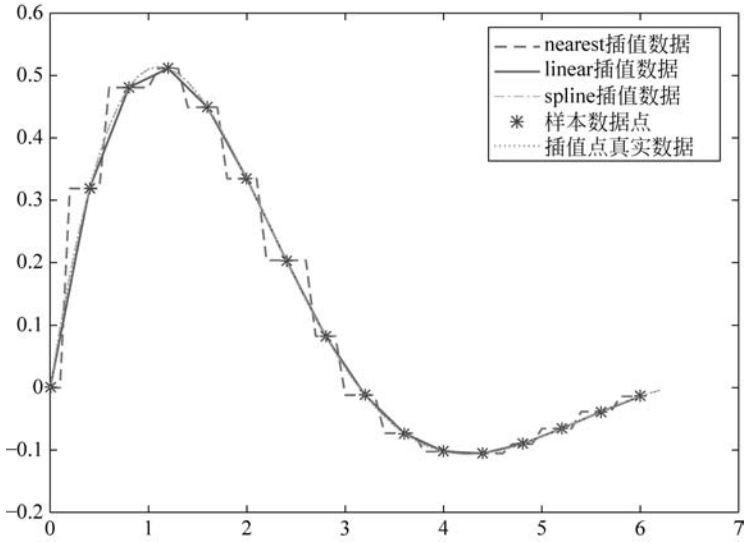

图 5-2 各种插值结果比较

由上面结果可知，最接近点拟合误差大，直线拟合得到的曲线不平滑；三次样条插值的效果最好，曲线平滑，误差很小，基本逼近真实值。

2. 一维快速傅里叶插值

在 MATLAB 中，一维快速傅里叶插值可以用 interpft 函数实现。该函数利用傅里叶变换将输入数据变换到频率域，然后用更多点进行傅里叶逆变换，实现对数据的插值。函数调用格式为

```
y = interpft(x,n)    % 表示对 x 进行傅里叶变换，然后采用 n 点傅里叶逆变换，得到插值后的数据
y = interpft(x,n,dim)    % 表示在 dim 维上进行傅里叶插值
```


微课视频

【例 5-11】 假设测量的数据来自函数 $f(x)=\sin x$，试根据生成的数据，用一维快速傅里叶插值，并比较插值结果。

在文件编辑窗口编写命令文件，保存为 exam_5_11.m 脚本文件。程序代码如下：

```
clear
x = 0:0.4:2 * pi;
y = sin(x);                                          % 原始数据
N = length(y);
M = N * 4;
x1 = 0:0.1:2 * pi;
y1 = interpft(y,M - 1);                              % 傅里叶插值
y2 = sin(x1);                                        % 插值点真实数据
plot(x,y,'O',x1,y1,' * ',x1,y2,' - ')
legend('原始数据','傅里叶插值数据','插值点真实数据')
max(abs(y1 - y2))
```

程序运行结果如下，插值效果如图 5-3 所示。

```
>> exam_5_11
ans =
    0.0980
```

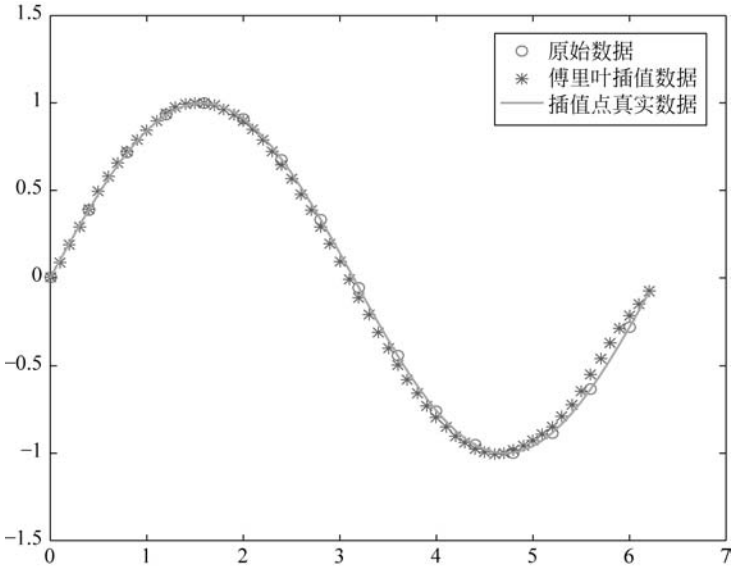

图 5-3　一维快速傅里叶插值及比较

由以上结果可知，一维快速傅里叶插值实现插值的速度比较快，曲线平滑，误差很小，基本逼近真实值。

3. 三次样条插值

三次样条插值是利用多段多项式逼近插值，降低了插值多项式的阶数，使得曲线更为光滑。在 MATLAB 中，interp1 插值函数的 method 选为 spline 样条插值选项，就可以实现三次样条插值。另外，MATLAB 专门提供三次样条插值函数 spline，其格式如下：

```
yi = spline(x,y,xi)    % 利用初始值 x,y,对插值点数据 xi 进行三次样条插值.采用这种调
                       % 用方式,相当于 yi = interp1(x,y,xi,'spline')
```

微课视频

【例 5-12】 已知数据 x=[−5 −4 −3 −2 −1 0 1 2 3 4 5]，y=[25 16 9 4 1 0 1 4 9 16 25]，对 xi=−5:0.5:5，用 spline 进行三次样条插值，并比较用 interp1 实现三次样条插值。

在文件编辑窗口编写命令文件，保存为 exam_5_12.m 脚本文件。程序代码如下：

```
x = -5:5;
y = x. * x;
xi = -5:0.5:5;
y0 = xi. * xi;
y1 = spline(x,y,xi);
y2 = interp1(x,y,xi,'spline');
plot(x,y,'O',xi,y0,xi,y1,'+',xi,y2,'*')
legend('原始数据','插值点真实数据','spline 插值','interp1 样条插值')
max(abs(y1 - y2))
```

程序运行结果如下，插值效果如图 5-4 所示。

```
>> exam_5_12
ans =
     0
```

由程序结果可知，三次样条插值 spline 函数实现插值的效果和 interp1(x,y,xi,'spline') 一样。

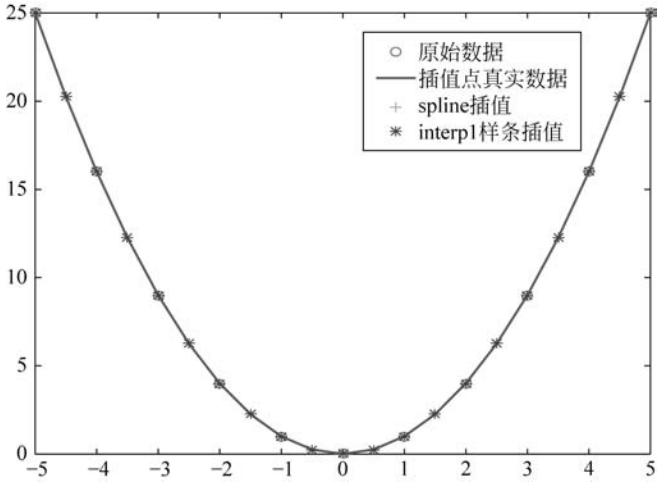

图 5-4 三次样条插值及比较

5.2.2 二维插值

二维插值是指已知一个二元函数的若干个采样数据点 x,y 和 z(x,y),求插值点(x1,y1)处的 z1 的值。在 MATLAB 中,提供 interp2 函数用于实现二维插值,其调用格式为

```
Z1 = interp2(X,Y,Z,X1,Y1,'method')
```

其中,X 和 Y 是两个参数的采样点,一般是向量,Z 是参数采样点对应的函数值。X1 和 Y1 是插值点,可以是标量也可以是向量。Z1 是根据选定的插值方法(method)得到的插值结果。插值方法 method 和一维插值函数相同,linear 为线性插值(默认算法),nearest 为最近点插值,spline 为三次样条插值,cubic 为三次多项式插值。需要注意,X1 和 Y1 不能超出 X 和 Y 的取值范围,否则会得到 NaN 错误信息。


微课视频

【例 5-13】 某实验对电脑主板的温度分布做测试。用 x 表示主板的宽度(cm),y 表示主板的深度(cm),用 T 表示测得的各点温度(℃),测量结果如表 5-2 所示。

(1) 分别用最近点二维插值和线性二维插值法求(12.6,7.2)点的温度。

(2) 用三次多项式插值求主板宽度每 1cm,深度每 1cm 处各点的温度,并用图形显示插值前后主板的温度分布图。

表 5-2 主板各点温度测量值

y	x					
	0	5	10	15	20	25
0	30	32	34	33	32	31
5	33	37	41	38	35	33
10	35	38	44	43	37	34
15	32	34	36	35	33	32

在文件编辑窗口编写命令文件,保存为 exam_5_13.m 脚本文件。程序代码如下:

```
clear
x = [0:5:25];
y = [0:5:15]';
T = [30   32  34  33  32  31;
33  37  41  38  35  33;
35  38  44  43  37  34;
32  34  36  35  33  32];
x1 = 12.6;y1 = 7.2;                                  % 插值点(12.6,7,2)
T1 = interp2(x,y,T,x1,y1,'nearest')                  % 最近点二维插值
T2 = interp2(x,y,T,x1,y1,'linear')                   % 线性二维插值
xi = [0:1:25];
yi = [0:1:15]';
Ti = interp2(x,y,T,xi,yi,'cubic');                   % 三次多项式二维插值
subplot(1,2,1)
mesh(x,y,T)
xlabel('板宽度(cm)');ylabel('板深度(cm)');zlabel('温度(摄氏度)')
title('插值前主板温度分布图')
subplot(1,2,2)
```

```
mesh(xi,yi,Ti)
xlabel('板宽度(cm)');ylabel('板深度(cm)');zlabel('温度(摄氏度)')
title('插值后主板温度分布图')
```

运行程序，结果如下，图 5-5 是插值前后主板温度分布图。可知，用插值技术处理数据，可以使得温度分布图更加光滑。

```
>> exam_5_13
T1 =
    38
T2 =
   41.2176
```

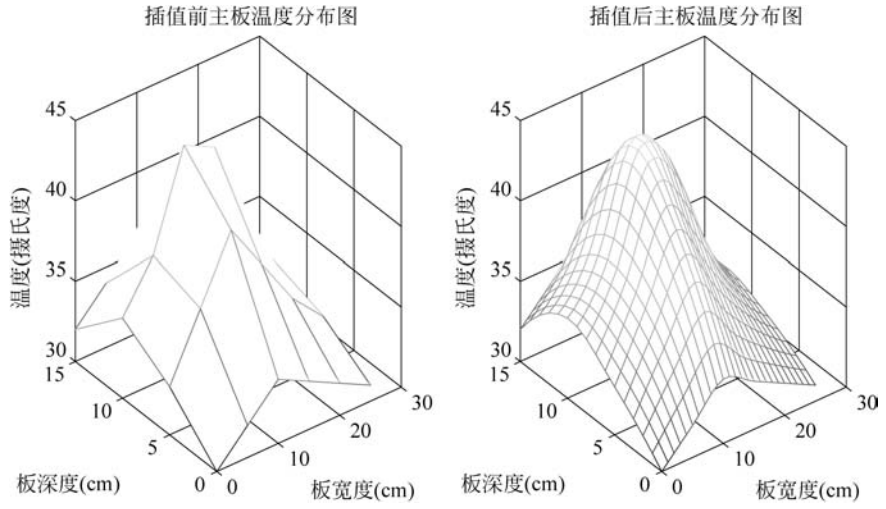

图 5-5　插值前后主板温度分布图

5.2.3　多维插值

1. 三维插值

在 MATLAB 中，还提供了三维插值的函数 interp3，其调用格式为：

```
U1 = interp3(X,Y,Z,U,X1,Y1,Z1,'method')
```

其中，X、Y、Z 是三个参数的采样点，一般是向量，U 是参数采样点对应的函数值。X1、Y1、Z1 是插值点，可以是标量也可以是向量。U1 是根据选定的插值方法(method)得到的插值结果。插值方法 method 和一维插值函数相同，linear 为线性插值(默认算法)，nearest 为最近点插值，spline 为三次样条插值，cubic 为三次多项式插值。需要注意，X1、Y1 和 Z1 不能超出 X、Y 和 Z 的取值范围，否则会得到 NaN 错误信息。

2. *n* 维插值

在 MATLAB 中，还可以实现更高维的插值，interpn 函数用于实现 *n* 维插值。其调用格式为：

```
U1 = interpn(X1,X2,…,Xn,U,Y1,Y2,…,Yn,'method')
```

其中，$X_1,X_2,\cdots,X_n$ 是 n 个参数的采用点，一般是向量，U 是参数采样点对应的函数值。$Y_1,Y_2,\cdots,Y_n$ 是插值点，可以是标量也可以是向量。U1 是根据选定的插值方法(method)得到的插值结果。插值方法 method 和一维插值函数相同，linear 为线性插值(默认算法)，nearest 为最近点插值，spline 为三次样条插值，cubic 为三次多项式插值。需要注意，$Y_1,Y_2,\cdots,Y_n$ 不能超出 $X_1,X_2,\cdots,X_n$ 的取值范围，否则会得到 NaN 错误信息。

5.3 数据拟合

与数据插值类似，数据拟合的目的也是用一个较为简单的函数 $g(x)$ 去逼近一个未知的函数 $f(x)$。利用已知的测量数据 $(x_i,y_i)(i=1,2,\cdots,n)$，构造函数 $y=g(x)$，使得误差 $\delta_i=g(x_i)-f(x_i)(i=1,2,\cdots,n)$ 在某种意义上达到最小。

一般用得比较多的是多项式拟合，所谓多项式拟合是利用已知的测量数据 $(x_i,y_i)(i=1,2,\cdots,n)$，构造一个 $m(m<n)$ 次多项式 $p(x)$：

$$p(x)=a_m x^m+a_{m-1}x^{m-1}+\cdots+a_1 x+a_0 \tag{5-4}$$

使得拟合多项式在各采用点处的偏差的平方和 $\sum_{i=1}^{n}(p(x_i)-y_i)^2$ 最小。

在 MATLAB 中，用 polyfit 函数可以实现最小二乘意义的多项式拟合。polyfit 拟合函数求的是多项式的系数向量。该函数的调用格式为：

```
p = polyfit(x,y,n)
[p,S] = polyfit(x,y,n)
```

其中，p 为最小二乘意义上的 n 阶多项式系数向量，长度为 n+1，x,y 为数据点向量，要求为等长向量，S 为采样点的误差结构体，包括 R，df 和 normr 分量，分别表示对 x 进行 QR 分解为三角元素、自由度和残差。


微课视频

【例 5-14】 在 MATLAB 中，用 polyfit 函数实现一个 4 阶和 5 阶多项式在区间 $[0,3\pi]$ 内逼近函数 $f(x)=e^{-0.5x}\sin x$，利用绘图的方法，比较拟合的 4 阶多项式、5 阶多项式和 $f(x)$ 的区别。

在文件编辑窗口编写命令文件，保存为 exam_5_14.m 脚本文件。程序代码如下：

```
clear
x = linspace(0,3 * pi,30);                    % 在给定区间,均匀选取 30 个采样点
y = exp( - 0.5 * x). * sin(x);
[p1,s1] = polyfit(x,y,4)                      % 4 阶多项式拟合
g1 = poly2str(p1,'x')                         % 显示拟合的 4 阶多项式
[p2,s2] = polyfit(x,y,5)                      % 5 阶多项式拟合
g2 = poly2str(p2,'x')                         % 显示拟合的 5 阶多项式
y1 = polyval(p1,x);                           % 用 4 阶多项式求采样的值
y2 = polyval(p2,x);                           % 用 5 阶多项式求采样的值
plot(x,y,' - * ',x,y1,':O',x,y2,': + ')       % 4 阶多项式,5 阶多项式和 f(x)绘图比较
legend('f(x)','4 阶多项式','5 阶多项式')
```

程序运行结果如下，图 5-6 是 4 阶多项式和 5 阶多项式拟合 f(x)函数的比较结果。

```
>> exam_5_14
p1 =
   -0.0024    0.0462   -0.2782    0.4760    0.1505
s1 =
  包含以下字段的 struct:
        R: [5×5 double]
       df: 25
    normr: 0.4086
g1 =
    '    -0.002378 x^4 + 0.04625 x^3 - 0.27815 x^2 + 0.476 x + 0.15048'
p2 =
    0.0007   -0.0191    0.1856   -0.7593    1.0826    0.0046
s2 =
  包含以下字段的 struct:
        R: [6×6 double]
       df: 24
    normr: 0.0909
g2 =
    '   0.00071166 x^5 - 0.019146 x^4 + 0.18564 x^3 - 0.75929 x^2 + 1.0826 x
        + 0.0045771'
```

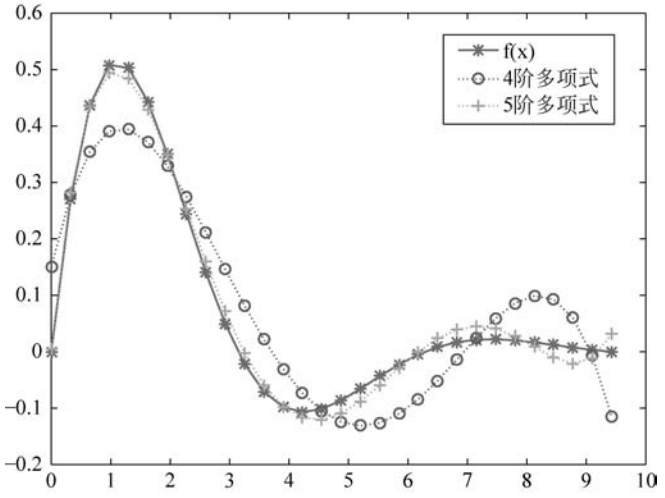

图 5-6　4 阶多项式和 5 阶多项式拟合 f(x)函数

由上述例题结果可知,用高阶多项式拟合 $f(x)$函数的效果更好,误差小,更加逼近实际函数 $f(x)$。

5.4　数据统计

在科学研究和生产实际中经常需要对数据进行统计，MATLAB 语言提供了很多数据统计方面的函数。

5.4.1 矩阵元素的最大值和最小值

1. 求向量的最大元素和最小元素

1) 求向量的最大元素

在 MATLAB 中,可以用函数 max(X)求一个向量 $\boldsymbol{X}$ 的最大元素,其调用格式为

```
y = max(X)          % 返回向量 X 的最大元素给 y,如果 X 中包括复数元素,则按模取最大元素
[y,k] = max(X)      % 返回向量 X 的最大元素给 y,最大元素所在的位置序号给 k,如果 X 中包括复数
                    % 元素,则按模取最大元素
```

例如,求向量 $\boldsymbol{X}=[34,23,-23,6,76,56,14,35]$的最大值。

```
>> X = [34,23, - 23,6,76,56,14,35];
>> y = max(X)
y =
    76
>> [y,k] = max(X)
y =
    76
k =
     5
```

2) 求向量的最小元素

在 MATLAB 中,可以用函数 min(X)求一个向量 $\boldsymbol{X}$ 的最小元素,其调用格式及用法与 max(X)函数一样。

例如,求向量 $\boldsymbol{X}=[34,10,-23,6,76,0,14,35]$的最小值。

```
>> X = [34,10, - 23,6,76,0,14,35];
>> y = min(X)
y =
   - 23
>> [y,k] = min(X)
y =
   - 23
k =
     3
```

2. 求矩阵的最大元素和最小元素

1) 求矩阵的最大元素

在 MATLAB 中,可以用函数 max 求一个矩阵 $\boldsymbol{A}$ 的最大元素,其调用格式为:

```
Y = max(A)          % 返回矩阵 A 的每列上最大元素给 Y,Y 是一个行向量
[Y, K] = max(A)     % 返回矩阵 A 的每列上最大元素给 Y,K 向量记录每列最大元素所在的行号
                    % 如果 X 中包括复数元素,则按模取最大元素
[Y, K] = max(A, [], dim)
```

其中 dim 为 1 时,该函数和 max(A)完全相同。当 dim 为 2 时,该函数返回一个每行上最大元素的列向量。

2）求矩阵的最小元素

在 MATLAB 中，可以用函数 min 求一个矩阵 $\boldsymbol{A}$ 的最小元素，其调用格式及用法和 max 函数一样。


微课视频

【例 5-15】 在 MATLAB 中，用 max 和 min 函数，求矩阵 $\boldsymbol{A}$ 的每行和每列的最大和最小元素，并求整个 $\boldsymbol{A}$ 的最大和最小元素。

$$\boldsymbol{A}=\begin{bmatrix} 12 & 1 & -6 & 24 \\ -4 & 23 & 12 & 0 \\ 2 & -3 & 18 & 6 \\ 45 & 3 & 16 & -7 \end{bmatrix}$$

程序代码如下：

```
>> A = [12 1 -6 24; -4 23 12 0;2 -3 18 6;45 3 16 -7];
>> Y1 = max(A,[],2)          % 求每行最大元素
Y1 =
    24
    23
    18
    45
>> [Y2,K] = min(A,[],2)      % 求每行最小元素,及每行最小值的列数
Y2 =
    -6
    -4
    -3
    -7
K =
     3
     1
     2
     4
>> Y3 = max(A)               % 求每列的最大元素
Y3 =
    45    23    18    24
>> [Y4,K1] = min(A)          % 求每列的最小元素,及最小元素所在的行数
Y4 =
    -4    -3    -6    -7
K1 =
     2     3     1     4
>> ymax = max(max(A))        % 求矩阵 A 的最大元素
ymax =
    45
>> ymin = min(min(A))        % 求矩阵 A 的最小元素
ymin =
    -7
```

3. 两个维度一样的向量或矩阵对应元素比较

max 和 min 函数还能对两个维度一样的向量或矩阵对应元素求较大值和较小值，其调用格式为：

```
Y = max(A, B)
```

其中，A 和 B 是同维度的向量或矩阵，Y 的每个元素为 A 和 B 对应元素的较大者，与 A 和 B 同维。

min 函数的用法和 max 一样。

例如，求 ***A*** 和 ***B*** 矩阵对应元素的较大元素 Y1 和较小元素 Y2。

程序代码如下：

```
>> A = [1 5 6;7 3 1;3 7 4]
A =
      1     5     6
      7     3     1
      3     7     4
>> B = [2 9 4;9 1 3; - 1 0 3]
B =
      2     9     4
      9     1     3
    - 1     0     3
>> Y1 = max(A,B)
Y1 =
      2     9     6
      9     3     3
      3     7     4
>> Y2 = min(A,B)
Y2 =
      1     5     4
      7     1     1
    - 1     0     3
```

5.4.2 矩阵元素的平均值和中值

数据序列的平均值指的是算术平均，中值是指数据序列中其值位于中间的元素，如果数据序列个数为偶数，中值等于中间两项的平均值。

在 MATLAB 中，求矩阵或向量元素的平均值用 mean 函数，求中值用 median 函数。它们的调用方法如下：

```
(1) y = mean(X)          % 返回向量 X 的算术平均值
(2) Y = mean(A)          % 返回一个矩阵 A 每列的算术平均值的行向量
(3) y = median(X)        % 返回向量 X 的中值
(4) Y = median(A)        % 返回一个矩阵 A 每列的中值的行向量
(5) Y = mean(A,dim)      % 当 dim 为 1 时，等同于 mean(A)；当 dim 为 2 时，返回一个矩阵 A
                         % 每行的算术平均值的列向量
(6) Y = median(A,dim)    % 当 dim 为 1 时，等同于 median(A)；当 dim 为 2 时，返回一个
                         % 矩阵 A 每行的中值的列向量
```

例如，求向量 ***X*** 和矩阵 ***A*** 的平均值和中值。

程序代码如下：

```
>> X = [1,12,23,7,9, - 5,30];
```

```
>> y1 = mean(X)
y1 =
     11
>> y2 = median(X)
y2 =
      9
>> A = [0 9 2;7 3 3; - 1 0 3]
A =
      0     9     2
      7     3     3
    - 1     0     3
>> Y1 = mean(A)
Y1 =
      2.0000    4.0000    2.6667
>> Y2 = median(A)
Y2 =
      0     3     3
>> Y3 = mean(A,2)
Y3 =
      3.6667
      4.3333
      0.6667
>> Y4 = median(A,2)
Y4 =
      2
      3
      0
```

5.4.3 矩阵元素的排序

在 MATLAB 中,可以用函数 sort 实现数据序列的排序。对于向量 $\boldsymbol{X}$ 的排序,可以用函数 sort(X),函数返回一个对向量 $\boldsymbol{X}$ 的元素按升序排列的向量。

sort 函数还可以对矩阵 $\boldsymbol{A}$ 的各行或各列的元素重新排序,其调用格式为

```
[Y,I] = sort(A, dim, mode)
```

其中,当 dim 为 1 时,矩阵元素按列排序;当 dim 为 2 时,矩阵元素按行排序。dim 默认为 1。当 mode 为'ascend',则按升序排序;当 mode 为'descend',则按降序排序。mode 默认取'ascend'。Y 为排序后的矩阵,而 I 记录 Y 中元素在 A 中的位置。

例如,对一个向量 $\boldsymbol{X}$ 和一个矩阵 $\boldsymbol{A}$ 做各种排序。

程序代码如下:

```
>> X = [1,12,23,7,9, - 5,30];
>> Y = sort(X)
Y =
    - 5     1     7     9    12    23    30
>> A = [0 9 2;7 3 1; - 1 0 3]
A =
      0     9     2
```

```
       7     3     1
     - 1     0     3
>> Y1 = sort(A)
Y1 =
     - 1     0     1
       0     3     2
       7     9     3
>> Y2 = sort(A,1,'descend')
Y2 =
       7     9     3
       0     3     2
     - 1     0     1
>> Y3 = sort(A,2,'ascend')
Y3 =
       0     2     9
       1     3     7
     - 1     0     3
>> [Y4,I] = sort(A,2,'descend')
Y4 =
       9     2     0
       7     3     1
       3     0   - 1
I =
       2     3     1
       1     2     3
       3     2     1
```

5.4.4 矩阵元素求和与求积

在 MATLAB 中,向量和矩阵求和与求积的基本函数是 sum 和 prod,它们的使用方法类似,调用格式为:

```
(1) y = sum(X)            % 返回向量 X 各元素的和
(2) y = prod(X)           % 返回向量 X 各元素的乘积
(3) Y = sum(A)            % 返回一个矩阵 A 各列元素的和的行向量
(4) Y = prod(A)           % 返回一个矩阵 A 各列元素的乘积的行向量
(5) Y = sum(A,dim)        % 当 dim 为 1 时,该函数等同于 sum(A); 当 dim 为 2 时,返回一个
                          % 矩阵 A 各行元素的和的列向量
(6) Y = prod(A,dim)       % 当 dim 为 1 时,该函数等同于 prod(A); 当 dim 为 2 时,返回一个
                          % 矩阵 A 各行元素的乘积的列向量
```

例如,求一个向量 $\boldsymbol{X}$ 和一个矩阵 $\boldsymbol{A}$ 的各元素的和与乘积。

程序代码如下:

```
>> X = [1,3,9, - 2,7];
>> y = sum(X)              % 求向量 X 的各元素的和
y =
    18
>> y = prod(X)             % 求向量 X 的各元素的乘积
y =
```

```
   -378
>> A=[1 9 2;7 3 1;-1 1 3]
A =
      1     9     2
      7     3     1
     -1     1     3
>> Y1=sum(A)              %求矩阵A的各列元素的和
Y1 =
      7    13     6
>> Y2=sum(A,2)            %求矩阵A的各行元素的和
Y2 =
     12
     11
      3
>> Y3=prod(A)             %求矩阵A的各列元素的乘积
Y3 =
     -7    27     6
>> Y4=prod(A,2)           %求矩阵A的各行元素的乘积
Y4 =
     18
     21
     -3
>> y5=sum(Y1)             %求矩阵A所有元素的和
y5 =
     26
>> y6=prod(Y3)            %求矩阵A所有元素的乘积
y6 =
   -1134
```

5.4.5 矩阵元素的累加和与累乘积

在MATLAB中,向量和矩阵的累加和与累乘积的基本函数是cumsum和cumprod,它们的使用方法类似,调用格式为

```
(1) y=cumsum(X)              %返回向量X累加和向量
(2) y=cumprod(X)             %返回向量X累乘积向量
(3) Y=cumsum(A)              %返回一个矩阵A各列元素的累加和的矩阵
(4) Y=cumprod(A)             %返回一个矩阵A各列元素的累乘积的矩阵
(5) Y=cumsum(A,dim)          %当dim为1时,该函数等同于cumsum(A);当dim为2时,
                             %返回一个矩阵A各行元素的累加和矩阵
(6) Y=cumprod(A,dim)         %当dim为1时,该函数等同于cumprod(A);当dim为2时,
                             %返回一个矩阵A各行元素的累乘积矩阵
```

例如,求一个向量 $\boldsymbol{X}$ 和一个矩阵 $\boldsymbol{A}$ 的各元素的累加和与累乘积。

程序代码如下:

```
>> X=[1,3,9,-2,7];
>> Y=cumsum(X)
Y =
     1     4    13    11    18
```

```
>> Y = cumprod(X)
Y =
     1     3    27   - 54   - 378
>> A = [1 9 2;7 3 1; - 1 1 3]
A =
      1     9     2
      7     3     1
    - 1     1     3
>> Y1 = cumsum(A)
Y1 =
      1     9     2
      8    12     3
      7    13     6
>> Y2 = cumsum(A,2)
Y2 =
      1    10    12
      7    10    11
    - 1     0     3
>> Y3 = cumprod(A)
Y3 =
      1     9     2
      7    27     2
    - 7    27     6
>> Y4 = cumprod(A,2)
Y4 =
      1     9    18
      7    21    21
    - 1   - 1   - 3
```

5.4.6 标准方差和相关系数

1. 标准方差

对于具有 N 个元素的向量数据 $x_1, x_2, \cdots, x_N$，有如下两种标准方差的公式：

$$D_1 = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{5-5}$$

或

$$D_2 = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2} \tag{5-6}$$

其中，

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N}x_i \tag{5-7}$$

在 MATLAB 中，可以用函数 std 计算向量和矩阵的标准方差。对于向量 $\boldsymbol{X}$，std(X)返回一个标准方差；对于矩阵 $\boldsymbol{A}$，std(A)返回一个矩阵 $\boldsymbol{A}$ 各列或者各行的标准方差向量。std 函数的调用格式为：

```
(1) d = std(X)                    % 求向量 X 的标准方差
(2) D = std(A,flag,dim)
```

其中，当 dim 为 1 时，求矩阵 $\boldsymbol{A}$ 的各列元素的标准方差；当 dim 为 2 时，则求矩阵 $\boldsymbol{A}$ 的各行元素的标准方差。当 flag 为 0 时，按公式 D_1 计算标准方差；当 flag 为 1 时，按 D_2 计算标准方差。默认 flag＝0，dim＝1。

例如，求一个向量 $\boldsymbol{X}$ 和一个矩阵 $\boldsymbol{A}$ 的标准方差。

程序代码如下：

```
>> X = [1,3,9, - 2,7];
>> d = std(X)                    % 求向量 X 的标准方差
d =
    4.4497
>> A = [1 9 2;7 3 1; - 1 1 3]
A =
      1     9     2
      7     3     1
    - 1     1     3
>> D1 = std(A,0,1)               % 按 D1 标准方差公式,求矩阵 A 的列元素标准方差
D1 =
     4.1633    4.1633    1.0000
>> D2 = std(A,0,2)               % 按 D1 标准方差公式,求矩阵 A 的行元素标准方差
D2 =
     4.3589
     3.0551
     2.0000
>> D3 = std(A,1,1)               % 按 D2 标准方差公式,求矩阵 A 的列元素标准方差
D3 =
     3.3993    3.3993    0.8165
>> D4 = std(A,1,2)               % 按 D2 标准方差公式,求矩阵 A 的行元素标准方差
D4 =
     3.5590
     2.4944
     1.6330
```

2. 相关系数

对于两组数据序列 $x_i, y_i (i=1,2,\cdots,N)$，可以用下列式子定义两组数据的相关系数：

$$\rho = \frac{\sum_{i=1}^{N}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{N}(x_i - \bar{x})^2 \sum_{i=1}^{N}(y_i - \bar{y})^2}} \tag{5-8}$$

其中，

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i, \quad \bar{y} = \frac{1}{N}\sum_{i=1}^{N} y_i \tag{5-9}$$

在 MATLAB 中，可以用 corrcoef 函数计算数据的相关系数。corrcoef 函数的调用格式为：

```
(1) R = corrcoef(X,Y)          % 返回相关系数,其中 X 和 Y 是长度相等的向量
(2) R = corrcoef(A)            % 返回矩阵 A 的每列之间计算相关系数形成的相关系数矩阵
```

例如，求两个向量 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 的相关系数，并求正态分布随机矩阵 $\boldsymbol{A}$ 的均值、标准方差和相关系数。

程序代码如下：

```
>> X = [1,3,9, - 2,7];
>> Y = [2,3,7,0,6];
>> r = corrcoef(X,Y)            % 求 X 和 Y 向量的相关系数
r =
     1.0000    0.9985
     0.9985    1.0000
>> A = randn(1000,3);           % 产生一个均值为 0,方差为 1 的正态分布随机矩阵
>> y = mean(A)                  % 计算矩阵 A 的列均值
y =
     0.0253    0.0042    0.0427
>> D = std(A)                   % 计算矩阵 A 的列标准方差
D =
     0.9902    0.9919    1.0014
>> R = corrcoef(A)              % 计算 A 矩阵列的相关系数
R =
     1.0000    0.0023   - 0.0028
     0.0023    1.0000     0.0454
   - 0.0028    0.0454     1.0000
```

由上述结果可知，每列的均值接近 0，每列的标准方差接近 1，验证了 $\boldsymbol{A}$ 为标准正态分布随机矩阵。

5.5 数值计算

数值计算是指利用计算机求解数学问题（比如，函数的零点、极值、积分和微分以及微分方程）近似解的方法。常用的数值分析有求函数的最小值、求过零点、数值微分、数值积分和解微分方程等。

5.5.1 函数极值

数学上利用计算函数的导数来确定函数的最大和最小值点，然而，很多函数很难找到导数为零的点。为此，可以通过数值分析来确定函数的极值点。MATLAB 只有求极小值的函数，没有专门求极大值的函数，因为 $f(x)$ 的极大值问题等价于 $-f(x)$ 的极小值问题。MATLAB 求函数的极小值使用 fminbnd 和 fminsearch 函数。

1. 一元函数的极值

fminbnd 函数可以获得一元函数在给定区间内的最小值，函数调用格式如下：

```
(1) x = fminbnd(fun,x1,x2)
```

其中，fun 是函数的句柄或匿名函数；x1 和 x2 是寻找函数最小值的区间范围为 x1 < x < x2；x 为在给定区间内，极值所在的横坐标。

```
(2) [x,y] = fminbnd(fun,x1,x2)
```

其中，y 为求得的函数极值点处的函数值。


微课视频

【例 5-16】 已知 $y=e^{-0.2x}\sin(x)$，在 $0\leqslant x\leqslant 5\pi$ 区间内，使用 fminbnd 函数获取 y 函数的极小值。

在文件编辑窗口编写命令文件，保存为 exam_5_16.m 脚本文件。程序代码如下：

```
clear
x1 = 0;x2 = 5 * pi;
fun = @(x)(exp( - 0.2 * x) * sin(x));                    % 创建函数句柄
[x,y1] = fminbnd(fun,x1,x2)                              % 计算句柄函数的极小值
x = 0:0.1:5 * pi;
y = exp( - 0.2 * x). * sin(x);
plot(x,y)
grid on
```

程序运行结果如下，图 5-7 是函数在区间[0,5π]的函数曲线图。

```
>> exam_5_16
x =
     4.5150
y1 =
    - 0.3975
```

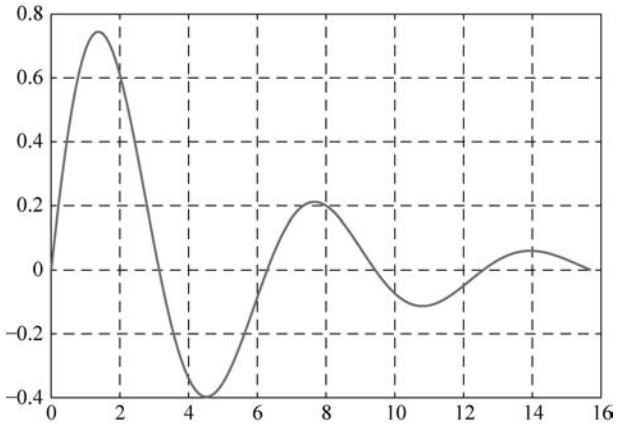

图 5-7 在区间[0,5π]函数曲线

由图 5-7 可知，函数在 $x=4.5$ 附近出现极小值点，极小值约为 -0.4，验证了用极小值 fminbnd 函数求的极小值点和极小值是正确的。

2. 多元函数的极值

fminsearch 函数可以获得多元函数的最小值，使用该函数时需要指定开始的初始值，获得初始值附近的局部最小值。该函数调用格式如下：

```
(1) x = fminsearch(fun, x0)
(2) [x,y] = fminsearch(fun, x0)
```

其中，fun 是多元函数的句柄或匿名函数；x0 是给定的初始值；x 是最小值的取值点；y 是返回的最小值，可以省略。

【例 5-17】 使用 fminsearch 函数获取 $f(x,y)$二元函数在初始值(0, 0)附近的极小值,已知 $f(x,y)=100(y-x^2)^2+(1-x)^2$。

在文件编辑窗口编写命令文件,保存为 exam_5_17.m 脚本文件。程序代码如下:


微课视频

```
clear
fun = @(x)(100 * (x(2) - x(1)^2)^2 + (1 - x(1))^2);        % 创建句柄函数
x0 = [0,0];
[x,y1] = fminsearch(fun,x0)                                 % 计算局部函数的极小值
```

程序运行结果如下:

```
>> exam_5_17
x =
    1.0000    1.0000
y1 =
    3.6862e - 10
```

由结果可知,由函数 fminsearch 计算出局部最小值点是[1, 1],最小值为 y1=3.6862e−10,和理论上是一致的。

5.5.2 函数零点

一元函数 $f(x)$过零点的求解相当于求解 $f(x)=0$ 方程的根,MATLAB 可以用 fzero 函数实现,使用该函数时需要指定一个初始值,在初始值附近查找函数值变号时的过零点,也可以根据指定区间来求过零点。该函数的调用格式为:

```
(1) x = fzero(fun, x0)
(2) [x,y] = fzero(fun, x0)
```

其中,x 为过零点的位置,如果找不到,则返回 NaN; y 是指函数在零点处函数的值; fun 是函数句柄或匿名函数; x0 是一个初始值或初始值区间。

需要指出,fzero 函数只能返回一个局部零点,不能找出所有的零点,因此需要设定零点的范围。

【例 5-18】 使用 fzero 函数求 $f(x)=x^2-5x+4$ 分别在初始值 $x0=0$,$x0=5$ 附近的过零点,并求出过零点函数的值。


微课视频

在文件编辑窗口编写命令文件,保存为 exam_5_18m 脚本文件。程序代码如下:

```
clear
fun = @(x)(x^2 - 5 * x + 4);                % 创建句柄函数
x0 = 0;
[x,y1] = fzero(fun,x0)                      % 求初始值 x0 为 0 附近,函数的过零点
x0 = 5;
[x,y1] = fzero(fun,x0)                      % 求初始值 x0 为 5 附近,函数的过零点
x0 = [0,3];
[x,y1] = fzero(fun,x0)                      % 求初始值 x0 区间内,函数的过零点
```

程序运行结果如下:

```
>> exam_5_18
x =
```

```
     1
y1 =
     0
x =
    4.0000
y1 =
  -3.5527e-15
x =
     1
y1 =
     0
```

由结果可知，用 fzero 函数可以求在初始值 x0 附近的函数过零点。不同的零点，需要设置不同的初始值 x0。

5.5.3 数值差分

任意函数 $f(x)$在 x 点的前向差分定义为

$$\Delta f(x)=f(x+h)-f(x) \tag{5-10}$$

称 $\Delta f(x)$为函数 $f(x)$在 x 点处以 $h(h>0)$为步长的向前差分。

在 MATLAB 中，没有直接求数值导数的函数，只有计算前向差分的函数 diff，其调用格式为：

```
(1) D = diff(X)          % 计算向量 X 的向前差分，即 D = X(i+1) - X(i), i = 1,2,…n-1
(2) D = diff(X, n)       % 计算向量 X 的 n 阶向前差分。即 diff(X, n) = diff(diff(X,n-1))
(3) D = diff(A, n, dim)  % 计算矩阵 A 的 n 阶差分。当 dim = 1(默认)，按行计算矩阵 A 的差分；
                         % 当 dim = 2，按列计算矩阵的差分
```

例如，已知矩阵 $\boldsymbol{A}=\begin{bmatrix}1 & 6 & 3\\ 6 & 2 & 4\\ 5 & 8 & 1\end{bmatrix}$，分别求矩阵 $\boldsymbol{A}$ 行和列的一阶和二阶前向差分。

```
>> A = [1 6 3;6 2 4;5 8 1]
A =
     1     6     3
     6     2     4
     5     8     1
>> D = diff(A,1,1)
D =
     5    -4     1
    -1     6    -3
>> D = diff(A,1,2)
D =
     5    -3
    -4     2
     3    -7
>> D = diff(A,2,1)
D =
    -6    10    -4
>> D = diff(A,2,2)
```

```
D =
    -8
     6
    -10
```

5.5.4 数值积分

数值积分是研究定积分的数值求解的方法。MATLAB 提供了很多种求数值积分的函数，主要包括一重积分和二重积分两类函数。

1. 一重数值积分

MATLAB 提供了 quad 函数和 quadl 函数求一重定积分。调用格式为：

```
(1) q = quad(fun, a, b, tol, trace)
```

它是一种采用自适应的 Simpson 方法的一重数值积分，其中 fun 为被积函数，是函数句柄；a 和 b 为定积分的下限和上限；tol 为绝对误差容限值，默认是 10^{-6}；trace 控制是否展现积分过程，当 trace 取非 0 时，则展现积分过程，默认取 0。

```
(2) q = quadl(fun, a, b, tol, trace)
```

它是一种采用自适应的 Lobatto 方法的一重数值积分，参数定义和 quad 一样。


微课视频

【例 5-19】 分别使用 quad 函数和 quadl 函数求 $q=\int_0^{3\pi} e^{-0.2x}\sin(x)dx$ 的数值积分。

在文件编辑窗口编写命令文件，保存为 exam_5_19.m 脚本文件。程序代码如下：

```
clear
fun = @(x)(exp(-0.2*x).*sin(x));                    %定义一个函数句柄
a = 0;b = 3*pi;
q1 = quad(fun,a,b)                                  %自适应 Simpson 方法的数值积分
q2 = quadl(fun,a,b)                                 %自适应 Lobatto 方法的数值积分
q3 = quad(fun,a,b,1e-3,1)                           %定义积分精度和显示积分过程
```

程序运行结果如下：

```
>> exam_5_19
q1 =
    1.1075
q2 =
    1.1075
     9     0.0000000000     2.55958120e+00      1.3793949196
    11     0.0000000000     1.27979060e+00      0.6053358622
    13     1.2797905993     1.27979060e+00      0.7742537042
    15     2.5595811986     4.30561556e+00     -0.6459997048
    17     2.5595811986     2.15280778e+00     -0.3430614927
    19     4.7123889804     2.15280778e+00     -0.3052258622
    21     6.8651967622     2.55958120e+00      0.3762543321
q3 =
    1.1076
```

其中，迭代过程最后一列的和为数值积分 q3 的值。

2. 多重数值积分

MATLAB 提供了 dblquad 函数和 triplequad 函数求二重积分和三重积分。调用格式如下：

```
(1) q2 = dblquad(fun, xmin, xmax, ymin,ymax, tol)
(2) q3 = triplequad(fun, xmin, xmax, ymin, ymax, zmin, zmax, tol)
```

函数的参数定义和一重积分一样。

例如，求二重数值积分 $q=\int_0^{3\pi}\int_0^{2\pi}\sin(x)y+x\sin(y)\mathrm{d}x\,\mathrm{d}y$。

代码如下：

```
>> q = dblquad('sin(x) * y + x * sin(y)',0,2 * pi,0,3 * pi)
q =
   39.4784
```

5.5.5 常微分方程求解

MATLAB 为解常微分方程提供了多种数值求解方法，包括 ode45、ode23、ode113、ode15s、ode23s、ode23t 和 ode23tb 等函数，用得最多的是 4/5 阶龙格-库塔法 ode45 函数。该函数格式如下：

```
[t,y] = ode45(fun,ts,y0,options)
```

其中，fun 是待解微分方程的函数句柄；ts 是自变量范围，可以是范围[t0, tf]，也可以是向量[t0,…,tf]；y0 是初始值，y0 和 y 是具有相同长度的列向量；options 是设定微分方程解法器的参数，可以省略，也可以由 odeset 函数获得。

需要注意，用 ode45 求解时，需要将高阶微分方程 $y^{(n)}=f(t,y,y',\cdots,y^{(n-1)})$，改写为一阶微分方程组，通常解法是，假设 $y_1=y$，从而 $y_1=y,y_2=y',\cdots,y_n=y^{(n-1)}$，于是高阶微分方程可以转换为下述常微分方程组求解：

$$\begin{cases} y_1'=y_2 \\ y_2'=y_3 \\ \vdots \\ y_n'=f(t,y,y',\cdots,y^{(n-1)}) \end{cases} \tag{5-11}$$


微课视频

【例 5-20】 已知二阶微分方程 $\frac{\mathrm{d}^2y}{\mathrm{d}t^2}-3y'+2y=1,y(0)=1,\frac{\mathrm{d}y(0)}{\mathrm{d}t}=0,t\in[0,1]$，试用 ode45 函数解微分方程，作出 $y\sim t$ 的关系曲线图。

(1) 首先把二阶微分方程改写为一阶微分方程组。

令 $y_1=y,y_2=y_1'$，则

$$\begin{bmatrix} \frac{\mathrm{d}y_1}{\mathrm{d}t} \\ \frac{\mathrm{d}y_2}{\mathrm{d}t} \end{bmatrix}=\begin{bmatrix} y_2 \\ 3y_2-2y_1+1 \end{bmatrix},\quad \begin{bmatrix} y_1(0) \\ y_2(0) \end{bmatrix}=\begin{bmatrix} 1 \\ 0 \end{bmatrix} \tag{5-12}$$

(2) 在文件编辑窗口编写命令文件，保存为 exam_5_20.m 脚本文件。程序代码如下：

```
clear
t0 = [0,1];                                  % 求解的时间区域
y0 = [1;0];                                  % 初值条件
[t,y] = ode45(@f05_20,t0,y0);                % 采用 ode45 函数解微分方程
plot(t,y(:,1))
xlabel('t'),ylabel('y')
title('y(t) - t')
grid on
    定义 f05_20 函数文件
function y = f05_20(t,y)
% f05_20 定义微分方程的函数文件
y = [y(2);3 * y(2) - 2 * y(1) + 1];
end
```

程序运行结果如图 5-8 所示。

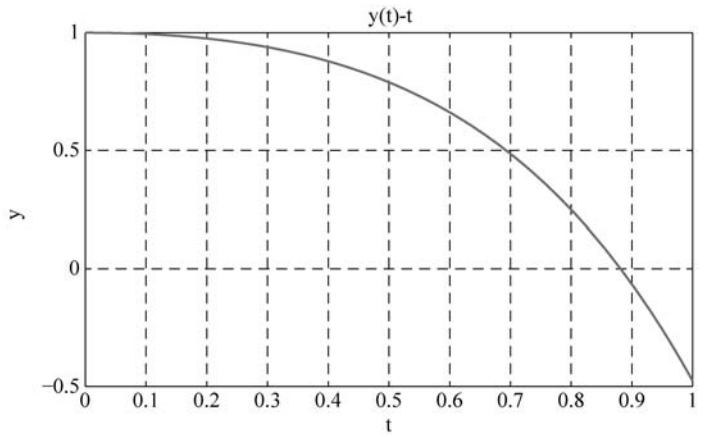

图 5-8　二阶微分方程的数值解

习题

1. 已知多项式 $p_1(x)=x^4-3x^3+5x+1$，$p_2(x)=x^3+2x^2-6$，求：

(1) $p(x)=p_1(x)+p_2(x)$；

(2) $p(x)=p_1(x)-p_2(x)$；

(3) $p(x)=p_1(x)\times p_2(x)$；

(4) $p(x)=p_1(x)/p_2(x)$。

2. 已知多项式为 $p(x)=x^4-2x^2+4x-6$，分别求 $x=3$ 和 $\boldsymbol{x}=[0,2,4,6,8]$向量的多项式的值。

3. 已知多项式为 $p(x)=x^4-2x^2+4x-6$，试求：

(1) 用 roots 函数求该多项式的根 r；

(2) 用 poly 函数求根为 r 的多项式系数。

4. 已知两个多项式为 $p_1(x)=x^4-3x^3+x+2$，$p_2(x)=x^3-2x^2+4$

(1) 求多项式 $p_1(x)$的导数；

(2) 求两个多项式乘积 $p_1(x)*p_2(x)$的导数；

(3) 求两个多项式相除 $p_2(x)/p_1(x)$的导数。

5. 已知分式表达式为 $f(s)=\dfrac{B(s)}{A(s)}=\dfrac{s+1}{s^2-7s+12}$。

(1) 求 $f(s)$的部分分式展开式；

(2) 将部分分式展开式转换为分式表达式。

6. 某电路元件，测试两端的电压 U 与流过的电流 I 的关系，实测数据见表 5-3 所示。用不同的插值方法(最接近点法、线性法、三次样条法和三次多项式法)计算 $I=9$A 处的电压 U。

表 5-3 实测数据

流过的电流 I/A	0	2	4	6	8	10	12
两端的电压 U/V	0	2	5	8.2	12	16	21

7. 某实验对一幅灰度图像灰度分布做测试。用 i 表示图像的宽度(PPI)，j 表示图像的深度(PPI)，用 I 表示测得的各点图像颜色的灰度，测量结果如表 5-4 所示。

(1) 分别用最近点二维插值、三次样条插值、线性二维插值法求(13,12)点的灰度值；

(2) 用三次多项式插值求图像宽度每 1PPI，深度每 1PPI 处各点的灰度值，并用图形显示插值前后图像的灰度分布图。

表 5-4 图像各点颜色灰度测量值

j	i					
	0	5	10	15	20	25
0	130	132	134	133	132	131
5	133	137	141	138	135	133
10	135	138	144	143	137	134
15	132	134	136	135	133	132

8. 用 polyfit 函数实现一个 3 阶和 5 阶多项式在区间[0,2]内逼近函数 $f(x)=e^{-0.5x}+\sin x$，利用绘图的方法，比较拟合的 5 阶多项式、7 阶多项式和 $f(x)$的区别。

9. 已知矩阵 $\boldsymbol{A}=\begin{bmatrix}10 & 4 & 7\\ 9 & 6 & 2\\ 3 & 9 & 4\end{bmatrix}$，试求：

(1) 用 max 和 min 函数，求每行和每列的最大和最小元素，并求整个 $\boldsymbol{A}$ 的最大和最小元素；

(2) 求矩阵 $\boldsymbol{A}$ 的每行和每列的平均值和中值；

(3) 对矩阵 $\boldsymbol{A}$ 做各种排序；

(4) 对矩阵 $\boldsymbol{A}$ 的各列和各行求和与求乘积；

(5) 求矩阵 $\boldsymbol{A}$ 的行和列的标准方差；

(6) 求矩阵 $\boldsymbol{A}$ 列元素的相关系数。

10. 已知 $y=e^{-0.5x}\sin(2*x)$，在 $0\leqslant x\leqslant\pi$ 区间内，使用 fminbnd 函数获取 y 函数的极小值。

11. 使用 fzero 函数求 $f(x)=x^2-8x+12$ 分别在初始值 $x=0$，$x=7$ 附近的过零点，并求出过零点函数的值。

12. 已知矩阵 $\boldsymbol{A}=\begin{bmatrix}10 & 4 & 7\\ 9 & 6 & 2\\ 3 & 9 & 4\end{bmatrix}$，分别求矩阵 $\boldsymbol{A}$ 行和列的一阶和二阶前向差分。

13. 分别使用 quad 函数和 quadl 函数求 $q=\int_0^{2\pi}\frac{\sin(x)}{x+\cos^2 x}\mathrm{d}x$ 的数值积分。

14. 求二重数值积分 $q=\int_0^{2\pi}\int_0^{2\pi}x\cos(y)+y\sin(x)\mathrm{d}x\,\mathrm{d}y$。

15. 已知二阶微分方程 $\frac{\mathrm{d}^2y}{\mathrm{d}t^2}-2y'+y=0$，$y(0)=1$，$\frac{\mathrm{d}y(0)}{\mathrm{d}t}=0$，$t\in[0,2]$，试用 ode45 函数解微分方程，作出 $y\sim t$ 的关系曲线图。

16. 洛伦兹(Lorenz)模型的状态方程表示为：

$$\begin{cases}\dfrac{\mathrm{d}x_1(t)}{\mathrm{d}t}=-\beta x_1(t)+x_2(t)x_3(t)\\ \dfrac{\mathrm{d}x_2(t)}{\mathrm{d}t}=-\delta x_2(t)+\delta x_3(t)\\ \dfrac{\mathrm{d}x_3(t)}{\mathrm{d}t}=-x_2(t)x_1(t)+\rho x_2(t)-x_3(t)\end{cases},\quad \begin{cases}x_1(0)=0\\ x_2(0)=0\\ x_3(0)=10^{-10}\end{cases}$$

取 $\delta=10$，$\rho=28$，$\beta=8/3$，解该微分方程，并绘制出 $x_1(t)\sim t$ 时间曲线和 $x_1(t)\sim x_2(t)$ 相空间曲线。