第 3 章

CHAPTER 3

MATLAB 数值计算及应用

数值计算指有效使用数字计算机求解数学问题近似解的方法与过程。相比于利用其他程序设计语言进行求解，MATLAB 编程具有效率高、计算方便等特点。

本章介绍多项式计算、数据统计与分析、数据插值、数值微积分及应用等内容，以达到熟练运用 MATLAB 解决简单工程问题的目的。

本章要点：

（1）多项式计算。

（2）数据统计与分析。

（3）数据插值。

（4）数值微积分及应用。

学习目标：

（1）掌握多项式的运算。

（2）掌握常用的数据统计与分析的基本方法。

（3）了解数据插值的基本原理。

（4）掌握数据插值的基本方法。

（5）掌握数值微积分的原理，熟练运用 MATLAB 求解微分方程模型。

（6）掌握运用 MATLAB 解决简单工程问题。

3.1 多项式计算

在数学中，由若干个单项式相加组成的代数式称为多项式，MATLAB 中提供了多项式的创建及各种运算方法，如表 3-1 所示。

表 3-1 多项式计算常用函数

函 数	功 能	函 数	功 能
poly2str	创建一个字符串型多项式	polyval	代数多项式求值
poly2sym	创建一个符号型多项式	polyvalm	矩阵多项式求值
conv	乘法运算	roots	求取多项式的全部根
deconv	除法运算	poly	由多项式的根求多项式系数
polyder	多项式求导	residue	多项式的部分分式展开
polyint	多项式积分		

3.1.1 多项式的创建

定义一个 n 次多项式为

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

在 MATLAB 中多项式 $p(x)$ 各项系数用一个向量表示，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 $p(x)$ 的各项系数可表示为

$$\boldsymbol{P}=[a_n,a_{n-1},\cdots,a_1,a_0]$$

创建一个多项式，可以用 poly2str 和 poly2sym 函数来实现，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
```

注意：两个函数均创建一个系数为 p，变量为 x 的多项式，但数据类型有所区别，一个是字符串型，另一个是符号型。

【例 3-1】 已知多项式系数为 $\boldsymbol{p}=[2,5,-3,-5]$，分别利用上述两个函数创建多项式，并比较两者不同。

程序如下：

```
p = [2,5, - 3, - 5]
f1 = poly2str(p,'x')
f2 = poly2sym(p)
```

程序执行后，运行结果为

```
p =
     2     5    - 3    - 5
f1 =
   2 x^3 + 5 x^2 - 3 x - 5
f2 =
2 * x^3 + 5 * x^2 - 3 * x - 5
```

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，可通过工作区查看。

3.1.2 多项式的四则运算

多项式之间可以进行四则运算，结果仍为多项式。

1. 加减运算

MATLAB 中未提供专门多项式加减运算的函数，事实上多项式的加减运算是多项式的向量系数相加减，在计算过程中，需要保证多项式阶次一致，缺少部分用 0 补足。

2. 乘法运算

两个多项式的乘法运算可以用 conv 函数实现，调用格式如下：

```
P = conv(p1,p2)                        % p1,p2 为两个多项式的系数向量
```

3. 除法运算

两个多项式的除法运算可以用 deconv 函数来实现,调用格式如下:

```
[q,r] = deconv(p1,p2)                    % p1,p2 为两个多项式的系数向量
```

其中,q 为商式,r 为余式,均为多项式系数向量。

deconv 是 conv 的逆函数。


微视频 3-1

【例 3-2】 已知 $f(x)=x^4+4x^3-3x+2$,$g(x)=x^3-2x^2+x$。求解:

(1) $f(x)+g(x)$

(2) $f(x)-g(x)$

(3) $f(x)\times g(x)$

(4) $f(x)/g(x)$

程序如下:

```
p1 = [1 4 0 -3 2];
p2 = [0 1 -2 1 0];
p3 = [1 -2 1 0];
p = p1 + p2
poly2sym(p)
p = p1 - p2
poly2sym(p)
p = conv(p1,p2)
poly2sym(p)
[q,r] = deconv(p1,p3)
p4 = conv(q,p3) + r
```

程序执行后,运行结果为

```
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
```

3.1.3 多项式的值和根

1. 多项式的值

在 MATLAB 中，多项式求值有 polyval 和 polyvalm 两个函数，两者区别在于前者为代数多项式求值，后者为矩阵多项式求值。

代数多项式求值使用函数 polyval，调用格式如下：

```
y = polyval(p,x)                    % p 为多项式系数,x 为自变量
```

若 x 为数值，则求多项式在该点的值；若 x 为向量或矩阵，则对向量或矩阵中每个元素求其多项式的值。

【例 3-3】 已知多项式为 $f(x)=x^3-2x^2+4x+6$，分别求 $x_1=2$ 和 $\boldsymbol{x}=[0,2,4,6,8,10]$ 向量的多项式的值。

程序如下：

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

程序执行后，运行结果为

```
y1 =
    14
y =
     6    14    54   174   422   846
```

矩阵多项式求值使用函数 polyvalm，调用格式如下：

```
y = polyvalm(p,X)                   % p 为多项式系数,X 为自变量
```

与代数求值不同，此处 X 要求为方阵，它以方阵为自变量求多项式的值。若 A 为方阵，p 为多项式 x^3-5x^2+8 的系数，那么 polyvalm(p,A)的含义是：

```
A*A*A-5*A*A+8*eye(size(A))
```

而 polyval(p,A)的含义是：

```
A.*A.*A-5.*A.*A+8*eye(size(A))
```

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

程序如下：

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

程序执行后,运行结果为

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

2. 多项式求根

n 次多项式有 n 个根,可以用 roots 函数求取多项式的全部根,调用格式如下:

```
r = roots(p)                          % p 为多项式系数向量,r 为根向量
```

MATLAB 中还提供了一个由多项式的根求多项式系数的函数,调用格式如下:

```
p = poly(r)                           % p 为多项式系数向量,r 为根向量
```

【例 3-5】 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。

程序如下:

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

程序执行后,运行结果为

```
r =
   -3.7485
   -1.2962
    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。

3.1.4 多项式的微积分运算

1. 多项式的微分

在 MATLAB 中,用 polyder 函数对多项式进行微分运算,可以对单个多项式求导,也可以对两个多项式乘积和商求导,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 积的导数
[p,q] = polyder(p1)                % 求多项式 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 3-6】 已知两个多项式为 $f(x)=x^4+4x^3-3x+2$，$g(x)=x^3-2x^2+x$。


微视频 3-2

(1) 求多项式 $f(x)$的导数。

(2) 求两个多项式乘积 $f(x)\times g(x)$的导数。

(3) 求两个多项式相除 $g(x)/f(x)$的导数。

程序如下：

```
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)
```

程序执行后，运行结果为

```
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

在 MATLAB 中，用 polyint 函数求多项式的积分，其调用格式如下：

```
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,积分常数项默认值 0
```

显然，polyint 是 polyer 的逆函数。

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


微视频 3-3

程序如下：

```
p = [1 4 0 -3 2];
I = polyint(p)
```

```
poly2sym(I)
p = polyder(I)
syms k
I1 = polyint(p,k)
poly2sym(I1)
```

程序执行后,运行结果为

```
I =
    0.2000    1.0000         0   -1.5000    2.0000         0
ans =
    1/5*x^5+x^4-3/2*x^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
```

3.1.5 多项式的部分方式展开

在 MATLAB 中,使用 residue 函数实现多项式部分分式展开,其调用格式如下:

```
[r,p,k] = residue(B,A)        % 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
                              % 为极点列向量,r 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除的分式,其调用格式如下:

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

【例 3-8】 已知分式表达式为

$$f(s)=\frac{B(s)}{A(s)}=\frac{3s^2+1}{s^2-5s+6}$$

(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。

程序如下:

```
a = [1 -5 6];
b = [3 0 1];
[r,p,k] = residue(b,a)
[b1,a1] = residue(r,p,k)
```

程序执行后,运行结果为

```
r =
   82.0000
  -25.0000
```

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

3.2 数据统计与分析

数据统计是科学研究中的常用方法，MATLAB 中提供了多种相关函数，如表 3-2 所示。

表 3-2 数据统计与分析常用函数

函 数	功 能	函 数	功 能
max	求最大元素	mean	求算术平均值
min	求最小元素	median	求中值
sum	求和	cumsum	累加和
prod	求积	cumprod	累乘积
std	标准方差	sort	排序
corrcoef	相关系数		

3.2.1 矩阵的最大元素和最小元素

1. 向量的最大元素和最小元素求解

求向量最大元素可调用函数 max(x)，求向量的最小元素可调用函数 min(x)，具体调用格式如下：

```
y = max(x)              % 返回 x 中最大元素给 y
[y,k] = max(x)          % 返回 x 中最大元素给 y,所在位置为 k
y = min(x)              % 返回 x 中最小元素给 y
[y,k] = min(x)          % 返回 x 中最小元素给 y,所在位置为 k
```

【例 3-9】 求向量 $\boldsymbol{X}=[34,23,-23,4,76,58,10,35]$的最大值和最小值。

程序如下：

```
X = [34,23, - 23,4,76,58,10,35];
y = max(X)
[y,k] = max(X)
y = min(X)
[y,k] = min(X)
```

程序执行后，运行结果为

```
y =
    76
```

```
y =
    76
k =
     5
y =
   -23
y =
   -23
k =
     3
```

2. 矩阵的最大元素和最小元素求解

求矩阵的最大元素及最小元素也可调用 max(x)和 min(x)函数,具体调用格式如下:

```
Y = max(A)              % 返回矩阵 A 每列最大元素给 Y,Y 是一个行向量
[Y,k] = max(A)          % 返回矩阵 A 每列最大元素给 Y,Y 是一个行向量,k 记录每列最大元素的行号
[Y,k] = max(A,[],dim) % dim = 2 时,返回为每行最大元素;dim = 1 时,与 max(A)完全相同
Y = min(A)              % 返回矩阵 A 每列最小元素给 Y,Y 是一个行向量
[Y,k] = min(A)          % 返回矩阵 A 每列最小元素给 Y,k 记录每列最小元素的行号
[Y,k] = min(A,[],dim) % dim = 2 时,返回为每行最小元素;dim = 1 时,与 min (A)完全相同
```

【例 3-10】 已知矩阵 $\boldsymbol{A}$,求每行和每列的最大元素和最小元素,并求整个 $\boldsymbol{A}$ 的最大元素和最小元素。

$$\boldsymbol{A}=\begin{bmatrix} 12 & 1 & 6 & -24 \\ -4 & 23 & 12 & 0 \\ 2 & -3 & 18 & 6 \\ 45 & 13 & 10 & -7 \end{bmatrix}$$

程序如下:

```
A = [12 1 6 -24; -4 23 12 0;2 -3 18 6;45 13 10 -7];
Y1 = max(A,[],2)
[Y2,K] = min(A,[],2)
Y3 = max(A)
[Y4,K1] = min(A)
ymax = max(max(A))
ymin = min(min(A))
```

程序执行后,运行结果为

```
Y1 =
    12
    23
    18
    45
Y2 =
   -24
    -4
    -3
    -7
```

```
K =
     4
     1
     2
     4
Y3 =
    45    23    18     6
Y4 =
    -4    -3     6   -24
K1 =
     2     3     1     1
ymax =
    45
ymin =
   -24
```

3. 同维度向量/矩阵的比较

对于相同维度的向量或矩阵，也可以用 max(x)和 min(x)函数求最大值和最小值，调用格式如下：

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

【例 3-11】 已知矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{B}$，求其对应元素最大值和最小值。

$$\boldsymbol{A}=\begin{bmatrix}12 & 1 & 6 & -24\\ -4 & 23 & 12 & 0\\ 2 & -3 & 18 & 6\\ 45 & 13 & 10 & -7\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}2 & 11 & -6 & 24\\ -14 & 3 & 22 & 7\\ 21 & -13 & 8 & 16\\ 5 & 13 & -10 & 17\end{bmatrix}$$

程序如下：

```
A=[12 1 6 -24;-4 23 12 0;2 -3 18 6;45 13 10 -7];
B=[2 11 -6 24;-14 3 22 7;21 -13 8 16;5 13 -10 17];
Y1=max(A,B)
Y2=min(A,B)
```

程序执行后，运行结果为

```
Y1 =
    12    11     6    24
    -4    23    22     7
    21    -3    18    16
    45    13    10    17
Y2 =
     2     1    -6   -24
   -14     3    12     0
     2   -13     8     6
     5    13   -10    -7
```

3.2.2 求矩阵的平均值和中值

求矩阵的算术平均值可以调用 mean 函数,求中值可以调用 median 函数,调用格式如下:

```
y = mean(X)      % 返回向量的算术平均值
y = median(X)    % 返回向量的中值
Y = mean(A)      % 返回矩阵每列的算术平均值的行向量
Y = median(A)    % 返回矩阵每列的中值的行向量
Y = mean(A,dim) % dim = 2 时,返回矩阵每行的算术平均值的列向量;dim = 1 时,与 mean(A)完全相同
Y = median(A,dim)      % dim = 2 时,返回矩阵每行的中值的列向量;dim = 1 时,与 median(A)完全相同
```

【例 3-12】 求向量 $\boldsymbol{X}$ 和矩阵 $\boldsymbol{A}$ 的平均值和中值。

$$\boldsymbol{X}=[34,23,-23,4,76,58,10,35],\quad \boldsymbol{A}=\begin{bmatrix}12 & 1 & 6 & -24\\ -4 & 23 & 12 & 0\\ 2 & -3 & 18 & 6\\ 45 & 13 & 10 & -7\end{bmatrix}$$

程序如下:

```
X = [34,23, - 23,4,76,58,10,35];
A = [12 1 6  - 24; - 4 23 12 0;2  - 3 18 6;45 13 10  - 7];
y1 = mean(X)
y2 = median(X)
Y1 = mean(A)
Y2 = median(A)
Y3 = mean(A,2)
Y4 = median(A,2)
```

程序执行后,运行结果为

```
y1 =
   27.1250
y2 =
   28.5000
Y1 =
   13.7500    8.5000   11.5000   - 6.2500
Y2 =
    7.0000    7.0000   11.0000   - 3.5000
Y3 =
   - 1.2500
    7.7500
    5.7500
   15.2500
Y4 =
    3.5000
    6.0000
    4.0000
   11.5000
```

3.2.3 矩阵元素求和与求积

向量/矩阵的求和可调用 sum 函数，求积可调用 prod 函数，具体调用格式如下：

```
y = sum(x)          % 返回向量 x 各元素之和
y = prod(x)         % 返回向量 x 各元素之积
Y = sum(A)          % 返回矩阵各列元素之和的行向量
Y = prod(A)         % 返回矩阵各列元素之积的行向量
Y = sum(A,dim)      % dim = 2 时，返回矩阵各行元素之和的列向量；dim = 1 时，与 sum(A)完全相同
Y = prod(A,dim)     % dim = 2 时，返回矩阵各行元素之积的列向量；dim = 1 时，与 prod(A)完全相同
```

【例 3-13】 求向量 $\boldsymbol{X}$ 和矩阵 $\boldsymbol{A}$ 的各元素的和与积。

$$\boldsymbol{X}=[34,23,-23,4,76,58,10,35],\quad \boldsymbol{A}=\begin{bmatrix}12 & 1 & 6 & -24\\ -4 & 23 & 12 & 0\\ 2 & -3 & 18 & 6\\ 45 & 13 & 10 & -7\end{bmatrix}$$

程序如下：

```
X = [34,23, - 23,4,76,58,10,35];
A = [12 1 6  - 24; - 4 23 12 0;2  - 3 18 6;45 13 10  - 7];
y1 = sum(X)
y2 = prod(X)
Y1 = sum(A)
Y2 = prod(A)
Y3 = sum(A,2)
Y4 = prod(A,2)
```

程序执行后，运行结果为

```
y1 =
   217
y2 =
 - 1.1100e + 011
Y1 =
    55    34    46   - 25
Y2 =
        - 4320         - 897        12960            0
Y3 =
    - 5
     31
     23
     61
Y4 =
        - 1728
             0
          - 648
       - 40950
```

3.2.4 矩阵元素累加和与累乘积

向量/矩阵累加和可调用函数 cumsum，累乘积可调用函数 cumprod。具体调用格式如下：

```
y = cumsum(x)       % 返回向量 x 累加和向量
y = cumprod(x)      % 返回向量 x 累乘积向量
Y = cumsum(A)       % 返回矩阵各列元素累加和的矩阵
Y = cumprod(A)      % 返回矩阵各列元素累乘积的矩阵
Y = cumsum(A,dim)   % dim = 2 时,返回矩阵各行元素累加和的矩阵;dim = 1 时,与 cumsum (A)完全相同
Y = cumprod(A,dim)  % dim = 2 时,返回矩阵各行元素累乘积的矩阵;dim = 1 时,与 cumsum (A)完全相同
```

【例 3-14】 求向量 $\boldsymbol{X}$ 和矩阵 $\boldsymbol{A}$ 的各元素的累加和与累乘积。

$$\boldsymbol{X}=[3,2,-2,4,7,9,10,5],\quad \boldsymbol{A}=\begin{bmatrix}12 & 1 & 6 & -24\\ -4 & 23 & 12 & 0\\ 2 & -3 & 18 & 6\\ 45 & 13 & 10 & -7\end{bmatrix}$$

程序如下：

```
X = [3,2, - 2,4,7,9,10,5];
A = [12 1 6  - 24; - 4 23 12 0;2  - 3 18 6;45 13 10  - 7];
y1 = cumsum(X)
y2 = cumprod(X)
Y1 = cumsum(A)
Y2 = cumprod(A)
Y3 = cumsum(A,2)
Y4 = cumprod(A,2)
```

程序执行后，运行结果为

```
y1 =
     3     5     3     7    14    23    33    38
y2 =
     3       6      - 12      - 48       - 336       - 3024      - 30240     - 151200
Y1 =
    12     1     6    - 24
     8    24    18    - 24
    10    21    36    - 18
    55    34    46    - 25
Y2 =
          12           1           6          - 24
         - 48          23          72            0
         - 96         - 69        1296            0
       - 4320         - 897       12960           0
Y3 =
    12    13    19    - 5
   - 4    19    31     31
     2   - 1    17     23
    45    58    68     61
```

```
Y4 =
          12          12          72       -1728
          -4         -92       -1104           0
           2          -6        -108        -648
          45         585        5850      -40950
```

3.2.5 标准方差和相关系数

1. 标准方差

对于具有 N 个元素的数据序列 $x_1, x_2, x_3, \cdots, x_N$，标准方差可以由下列两种公式计算：

$$S_1 = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

或

$$S_2 = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

其中

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N}x_i$$

计算向量/矩阵的标准方差时，可调用 std 函数，具体调用格式如下：

```
d = std(x)              % 求向量 x 的标准方差
D = std(A,flag,dim)     % 求矩阵 A 的标准方差。dim = 1 时求各列元素的标准方差,dim = 2 时求各行
                        % 元素的标准方差;flag = 0 时按公式计算标准方差 S1,flag = 1 时按公式计
                        % 算标准方差 S2。默认 flag = 0,dim = 1
```

【例 3-15】 求向量 $\boldsymbol{X}$ 和矩阵 $\boldsymbol{A}$ 的标准方差。

$$\boldsymbol{X} = [3, 2, -2, 4, 7, 9, 10, 5], \quad \boldsymbol{A} = \begin{bmatrix} 12 & 1 & 6 \\ -4 & 23 & 12 \\ 2 & -3 & 18 \end{bmatrix}$$

程序如下：

```
X = [3,2, - 2,4,7,9,10,5];
A = [12 1 6; - 4 23 12;2  - 3 18];
d = std(X)
D1 = std(A,0,1)
D2 = std(A,0,2)
D3 = std(A,1,1)
D4 = std(A,1,2)
```

程序执行后，运行结果为

```
d =
    3.9188
D1 =
    8.0829   14.0000    6.0000
```

```
D2 =
    5.5076
   13.5769
   10.9697
D3 =
    6.5997   11.4310    4.8990
D4 =
    4.4969
   11.0855
    8.9567
```

2. 相关系数

对于两组数据序列 $x_i, y_i (i=1,2,\cdots,n)$，相关系数可以由下列公式计算：

$$r=\frac{\sqrt{\sum(x_i-\bar{x})(y_i-\bar{y})}}{\sqrt{\sum(x_i-\bar{x})^2}\sqrt{\sum(y_i-\bar{y})^2}}$$

相关系数的计算可以调用 corrcoef 函数，具体调用格式如下：

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


微视频 3-4

【例 3-16】 求向量 $\boldsymbol{X}$ 和 $\boldsymbol{Y}$ 以及正态分布的 1000×5 随机矩阵的 5 列随机数据的相关系数。

$$\boldsymbol{X}=[3,2,-2,4,7,9,10,5],\quad \boldsymbol{Y}=[31,12,-12,24,37,91,18,53]$$

程序如下：

```
X = [3,2, - 2,4,7,9,10,5];
Y = [31,12, - 12,24,37,91,18,53];
r = corrcoef(X,Y)
R = corrcoef(randn(1000,5))
```

程序执行后，运行结果为

```
r =
    1.0000    0.6599
    0.6599    1.0000
R =
    1.0000   - 0.0543    0.0478    0.0083    0.0177
   - 0.0543    1.0000    0.0024    0.0399   - 0.0157
    0.0478    0.0024    1.0000   - 0.0147   - 0.0559
    0.0083    0.0399   - 0.0147    1.0000    0.0069
    0.0177   - 0.0157   - 0.0559    0.0069    1.0000
```

3.2.6 矩阵的排序

对向量/矩阵进行排序可以调用 sort 函数，具体调用格式如下：

```
Y = sort(X)                          % 返回一个按升序排列的向量
[Y,I] = sort(A,dim,mode)             % dim = 1 时按列排序,dim = 2 时按行排序;mode 为'ascend'时
                                     % 为升序,为'descend'时为降序;I 记录 Y 中元素在 A 中的位置
```

【例 3-17】 对向量 $\boldsymbol{X}$ 和矩阵 $\boldsymbol{A}$ 做各种排序。

$$\boldsymbol{X}=[3,2,-2,4,7,9,10,5],\quad \boldsymbol{A}=\begin{bmatrix}12 & 1 & 6\\ -4 & 23 & 12\\ 2 & -3 & 18\end{bmatrix}$$

程序如下：

```
X = [3,2, - 2,4,7,9,10,5];
A = [12 1 6; - 4 23 12;2 - 3 18];
Y = sort(X)
Y1 = sort(A)
Y2 = sort(A,1,'ascend')
Y3 = sort(A,2,'ascend')
[Y4,I] = sort(A,2,'descend')
```

程序执行后，运行结果为

```
Y =
    - 2     2     3     4     5     7     9    10
Y1 =
    - 4    - 3     6
      2     1    12
     12    23    18
Y2 =
    - 4    - 3     6
      2     1    12
     12    23    18
Y3 =
      1     6    12
    - 4    12    23
    - 3     2    18
Y4 =
     12     6     1
     23    12    - 4
     18     2    - 3
I =
      1     3     2
      2     3     1
      3     1     2
```

3.3 数据插值

在工程实践应用中，通常获取到的数据是离散的，若想得到离散值以外的其他数据，就需要对其进行插值操作。MATLAB 中提供了多种插值函数，如表 3-3 所示。

表 3-3 常用插值函数

函 数	功 能	函 数	功 能
interp1	一维插值	linear	线性插值
interpft	一维快速傅里叶插值	nearest	最邻近点插值
spline	三次样条插值	next	下一点插值
interp2	二维插值	previous	前一点插值
interp3	三维插值	cubic	三次多项式插值
interpn	n 维插值	interp1q	一维插值

3.3.1 一维插值

一维插值指的是被插值函数自变量是一个单变量函数。插值方法分为一维多项式插值、一维快速插值和三次样条插值。

1. 一维多项式插值

一维多项式插值可调用 interp1 函数实现，通过已知数据点计算目标插值点的数据，格式如下：

```
yi = interp1(Y,xi)                  % Y 为默认自变量 x = 1:n 对应的值
yi = interp1(X,Y,xi)                % X,Y 为长度一样的已知向量,xi 可以是标量,也可以是向量
yi = interp1(X,Y,xi,'method')       % method 是插值方法
```

method 插值方法选取如下：

(1) 线性(linear)插值——这是默认插值方法，它是把与插值点靠近的两个数据点以直线连接，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) 最邻近点(nearest)插值——根据插值点和最接近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) 下一点(next)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和最邻近点插值一样。

(4) 前一点(previous)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和最邻近插值点一样。

(5) 三次样条(spline)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) 三次多项式(cubic)插值——根据已知数据，求出一个三次多项式进行插值，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 xi 的取值，如果超出已知数据 X 的范围，则会返回 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，interp1 插值速度比 interp1q 快。需要注意，interp1q 执行的插值数据，X 必须是单调递增的。

【例 3-18】 给出概率积分：

$$f(x)=\frac{2}{\sqrt{\pi}}\int_0^x \mathrm{e}^{-x^2}\,\mathrm{d}x$$

其数据表如表 3-4 所示，用不同的插值方法计算 $f(0.472)$。

表 3-4 概率积分数据表

x	0.46	0.47	0.48	0.49
$f(x)$	0.484 655 5	0.493 754 2	0.502 749 8	0.511 668 3

程序如下：

```
x = 0.46:0.01:0.49;                    % 给出 x,f(x)
f = [0.4846555,0.4937542,0.5027498,0.5116683];
format long
F1 = interp1(x,f,0.472)                % 用默认方法,即线性插值方法计算 f(x)
F2 = interp1(x,f,0.472,'nearest')      % 用最近点插值方法计算 f(x)
F3 = interp1(x,f,0.472,'next')         % 用下一点插值方法计算 f(x)
F4 = interp1(x,f,0.472,'previous')     % 用前一点插值方法计算 f(x)
F5 = interp1(x,f,0.472,'spline')       % 用 3 次样条插值方法计算 f(x)
F6 = interp1(x,f,0.472,'cubic')        % 用 3 次多项式插值方法计算 f(x)
format short
```

程序执行后，运行结果为

```
F1 =
   0.495553320000000
F2 =
   0.493754200000000
F3 =
   0.493754200000000
F4 =
   0.495561119712056
F5 =
   0.495560736000000
F6 =
   0.495561119712056
```

插值方法的好坏依赖于被插值函数，没有一种对所有函数都是最好的插值方法。

2. 一维快速傅里叶插值

一维快速傅里叶插值可以调用 interpft 函数实现，调用格式如下：

```
y = interpft(x,n)          % 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
```

【例 3-19】 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，比较插值结果。

程序如下：

```
clear
x = 0:0.4:2 * pi;
y = sin(x);
N = length(y);
M = N * 4;
x1 = 0:0.1:2 * pi;
y1 = interpft(y,M - 1);
```

```
y2 = sin(x1);
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))
```

程序执行后,插值结果如图 3-1 所示,运行结果为

```
ans =
    0.0980
```

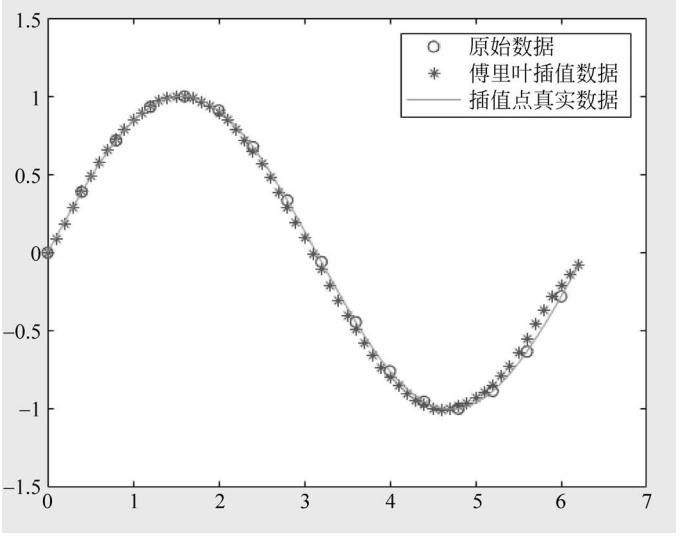


图 3-1　一维快速傅里叶插值及比较

3. 三次样条插值

三次样条插值可调用 spline 函数,调用格式如下:

```
Yi = spline(x,y,xi)                    % 相当于 yi = interp1(x,y,xi,'spline')
```

【例 3-20】 某检测参数 f 随时间 t 的采样结果如表 3-5 所示,用数据插值法计算 $t=2,7,12,17,22,27,32,37,42,47,52,57$ 时的 f 值,比较插值结果。

表 3-5　检测参数 f 随时间 t 的采样结果

t	0	5	10	15	20	25	30
f	3.1025	2.256	879.5	1835.9	2968.8	4136.2	5237.9
t	35	40	45	50	55	60	65
f	6152.7	6725.3	6848.3	6403.5	6824.7	7328.5	7857.6

程序如下:

```
T = 0:5:65;
X = 2:5:57;
F = [3.2015,2.2560,879.5,1835.9,2968.8,4136.2,5237.9,6152.7,...
6725.3,6848.3,6403.5,6824.7,7328.5,7857.6];
F1 = interp1(T,F,X)                    % 用线性插值方法插值
```

```
F2 = interp1(T,F,X,'nearest')        % 用最近点插值方法插值
F3 = interp1(T,F,X,'spline')         % 用三次样条插值方法插值
F4 = interp1(T,F,X,'cubic')          % 用三次多项式插值方法插值
plot(T,F,X,F1,'O',X,F2,'+',X,F3,'*',X,F4,'s')
legend('原始数据','线性插值','nearest 插值','spline 插值','cubic 插值')
max(abs(F1 - F4))
```

程序执行后，插值结果如图 3-2 所示，运行结果为

```
F1 =
  1.0e+003 *
    0.0028    0.3532    1.2621    2.2891    3.4358    4.5769    5.6038    6.3817
6.7745    6.6704    6.5720    7.0262
F2 =
  1.0e+003 *
    0.0032    0.0023    0.8795    1.8359    2.9688    4.1362    5.2379    6.1527
6.7253    6.8483    6.4035    6.8247
F3 =
  1.0e+003 *
   -0.1702    0.3070    1.2560    2.2698    3.4396    4.5896    5.6370    6.4229
6.8593    6.6535    6.4817    7.0441
F4 =
  1.0e+003 *
    0.0025    0.2232    1.2484    2.2736    3.4365    4.5913    5.6362    6.4362
6.7978    6.6917    6.5077    7.0186
ans =
  129.9586
```

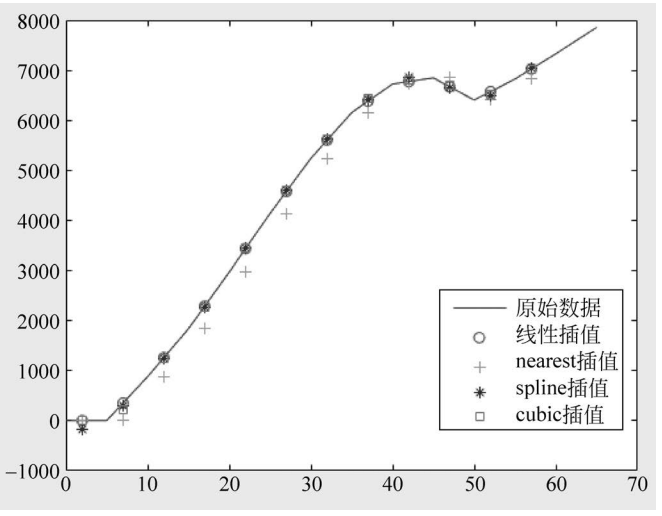


图 3-2 三次样条插值及比较

3.3.2 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 (x_1,y_1)处的 z_1 的值。在 MATLAB 中提供了 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得到的插值结果。插值方法和一维插值函数相同。


微视频 3-5

【例 3-21】 某实验对计算机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 3-6 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间隔 1cm，深度每间隔 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 3-6 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

程序如下：

```
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;
T1 = interp2(x,y,T,x1,y1,'nearest')
T2 = interp2(x,y,T,x1,y1,'linear')
xi = [0:1:25];yi = [0:1:15]';
Ti = interp2(x,y,T,xi,yi,'cubic');
subplot(1,2,1)
mesh(x,y,T)
xlabel('主板宽度(cm)');ylabel('主板深度(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('主板宽度(cm)');ylabel('主板深度(cm)');zlabel('温度(摄氏度)')
title('插值后主板温度分布图')
```

程序执行后，插值结果如图 3-3 所示，运行结果为

```
T1 =
    38
```

```
T2 =
   41.2176
```

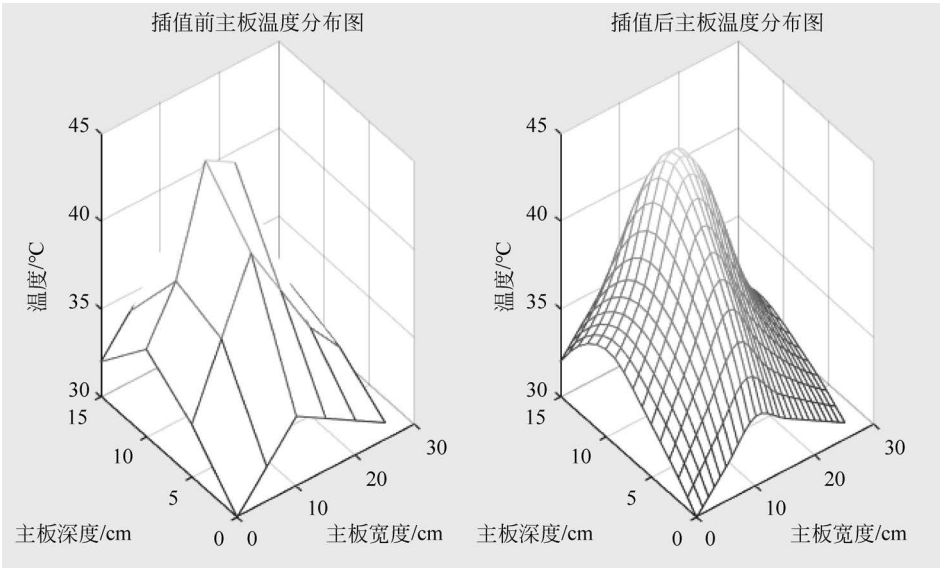


图 3-3 插值前后主板温度分布图

3.3.3 多维插值

1. 三维插值

三维插值可调用函数 interp3，其调用格式为：

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

其中，X、Y、Z 是 3 个参数的采样点，一般是向量，U 是采样点对应的函数值。X1、Y1 和 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得到的插值结果。插值方法和一维插值函数相同。

2. *n* 维插值

若需要对更高维进行插值运算，可以调用 interpn 函数实现，调用格式如下：

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

其中，X1，X2，…，Xn 是 n 个参数的采样点，一般是向量，U 是采样点对应的函数值。Y1，Y2，…，Yn 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得到的插值结果。插值方法和一维插值函数相同。

3.4 数值微积分

在 MATLAB 中，提供了多种函数实现有关的数值微积分运算，如表 3-7 所示。

表 3-7 数值微积分常用函数

函 数	功 能	函 数	功 能
diff	求微分、差分、导数	quad	一重积分
fminbnd	求一元函数的最小值	quadl	自适应一重积分
fminsearch	求多元函数的最小值	dblquad	二重积分
fzero	求函数零点	triplequad	三重积分
ode23	二阶、三阶龙格-库塔法	ode45	四阶、五阶龙格-库塔法

3.4.1 数值微分

在 MATLAB 中,没有可直接提供求数值导数或微分的函数,只有计算向前差分的函数 diff,具体调用格式如下:

```
DX = diff(X)                    % 计算向量 X 的向前差分
DX = diff(X,n)                  % 计算向量 X 的 n 阶向前差分
DX = diff(A,n,dim)              % 计算矩阵 A 的 n 阶向前差分,dim = 1,按列计算;dim = 2,按行计算
```

【例 3-22】 设 x 由在[0,2π]区间均匀分布的 10 个点组成,求 $\sin x$ 的 1～3 阶差分。程序如下:

```
X = linspace(0,2 * pi,10)
Y = sin(X)
DY = diff(Y)                    % 计算 Y 的一阶差分
D2Y = diff(Y,2)                 % 计算 Y 的二阶差分,也可用命令 diff(DY)计算
D3Y = diff(Y,3)                 % 计算 Y 的三阶差分,也可用 diff(D2Y)或 diff(DY,2)
```

程序执行后,运行结果为

```
X =
   0   0.6981   1.3963   2.0944   2.7925   3.4907   4.1888   4.8869   5.5851   6.2832
Y =
   0   0.6428   0.9848   0.8660   0.3420   - 0.3420   - 0.8660   - 0.9848   - 0.6428   - 0.0000
DY =
   0.6428   0.3420   - 0.1188   - 0.5240   - 0.6840   - 0.5240   - 0.1188   0.3420   0.6428
D2Y =
   - 0.3008   - 0.4608   - 0.4052   - 0.1600   0.1600   0.4052   0.4608   0.3008
D3Y =
   - 0.1600   0.0556   0.2452   0.3201   0.2452   0.0556   - 0.1600
```

3.4.2 函数极值与零点

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值点和最小值点,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 只有处理极小值的函数,没有专门求极大值的函数。因为 $f(x)$ 极大值问题等价于一阶 $f(x)$ 极小值问题。MATLAB 求函数的极小值使用 fminbnd 和 fminsearch 函数。具体调用格式如下:

```
x = fminbnd(fun,x1,x2)                    % 一元函数求给定区间内的最小值,x 为极值所在横坐标
[x,y] = fminbnd(fun,x1,x2)                % y 为极值点的函数值
x = fminsearch(fun,x0)                    % 多元函数最小值,x0 为初始值,x 为极值所在横坐标
[x,y] = fminsearch(fun,x0)                % y 为极值点的函数值
```

【例 3-23】 已知 $y=e^{-0.2x}\sin x$，在 $0\leqslant x\leqslant 5\pi$ 范围内，使用 fminbnd 函数获取 y 函数的极小值。


微视频 3-6

程序如下：

```
x1 = 0;x2 = 5 * pi;
fun = @(x)(exp( - 0.2 * x) * sin(x));
[x,y1] = fminbnd(fun,x1,x2)
x = 0:0.1:5 * pi;
y = exp( - 0.2 * x). * sin(x);
plot(x,y)
grid on
```

程序执行后，y 函数曲线如图 3-4 所示，运行结果为

```
x =
    4.5150
y1 =
   - 0.3975
```

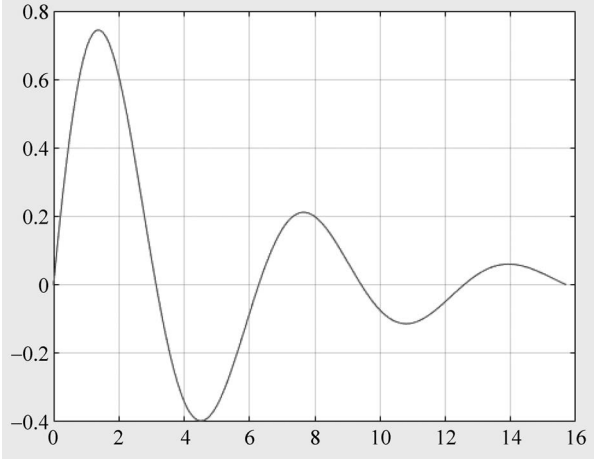


图 3-4 y 函数曲线

【例 3-24】 使用 fminsearch()函数获取二元函数 $f(x,y)=100(y-x^2)^2+(1-x)^2$ 在初始值(0,0)附近的极小值。

程序如下：

```
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
```

程序执行后，运行结果为

```
x =
    1.0000    1.0000
y1 =
  3.6862e-010
```

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 $y_1=$ 3.6862e-10，和理论结果是一致的。

2. 函数过零点

一元函数 $f(x)$的过零点的求解相当于求解 $f(x)=0$ 方程的根。MATLAB 可以使用 fzero 函数实现，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。该函数的调用格式如下：

```
x = fzero(fun,x0)        % x 为过零点的位置,fun 是函数句柄或匿名函数,x0 是初始值或初始区间
[x,y] = fzero(fun,x0)    % y 为函数在过零点处的函数值
```

【例 3-25】 使用 fzero()函数求 $f(x)=x^2-5x+4$ 分别在初始值 $x=0$ 和 $x=5$ 附近的过零点，并求出过零点函数的值。

程序如下：

```
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 附近时函数的过零点
x0 = 5;
[x,y1] = fzero(fun,x0)                    % 求初始值 x0 为 5 附近时函数的过零点
x0 = [0,3];
[x,y1] = fzero(fun,x0)                    % 求初始值 x0 在区间内函数的过零点
```

程序执行后，运行结果为

```
x =
     1
y1 =
     0
x =
    4.0000
y1 =
 -3.5527e-015
x =
     1
y1 =
     0
```

3.4.3 常微分方程的数值求解

常微分方程初值问题的数值解法多种多样，比较常用的有欧拉法、龙格-库塔法、线性多步法、预报校正法等。下面简要介绍龙格-库塔法及其实现。

1. 龙格-库塔法简介

对于一阶常微分方程的初值问题，在求解未知函数 y 时，y 在 t_0 点的值 $y(t_0)=y_0$ 是

已知的，并且根据高等数学中的中值定理，应有：

$$\begin{cases} y(t_0+h)=y_1 \approx y_0+hf(t_0,y_0) \\ y(t_0+2h)=y_2 \approx y_1+hf(t_1,y_1) \end{cases}, \quad h>0$$

一般地，在任意点 $t_i=t_0+ih$，有：

$$y(t_0+ih)=y_i \approx y_{i-1}+hf(t_{i-1},y_{i-1}), \quad i=1,2,\cdots,n$$

当(t_0,y_0)确定后，根据上述递推式能计算出未知函数 y 在点 $t_i=t_0+ih$，$i=0,1,\cdots,n$ 的一列数值解

$$y_i=y_0,y_1,y_2,\cdots,y_n, \quad i=0,1,2,\cdots,n$$

当然，在递推过程中有一个误差累积的问题。在实际计算过程中使用的递推公式一般进行过改造，著名的龙格-库塔公式为：

$$y(t_0+ih)=y_i \approx y_{i-1}+\frac{h}{6}(k_1+2k_2+3k_3+4k_4)$$

其中

$$k_1=f(t_{i-1},y_{i-1})$$

$$k_2=f\left(t_{i-1}+\frac{h}{2},y_{i-1}+\frac{h}{2}k_1\right)$$

$$k_3=f\left(t_{i-1}+\frac{h}{2},y_{i-1}+\frac{h}{2}k_2\right)$$

$$k_4=f(t_{i-1}+h,y_{i-1}+hk_3)$$

2. 龙格-库塔法的实现

基于龙格-库塔法，MATLAB 提供了求常微分方程数值解的函数，一般调用格式如下：

```
[t,y] = ode23(filename,tspan,y0)          % 二阶、三阶龙格 - 库塔法
[t,y] = ode45(filename,tspan,y0)          % 四阶、五阶龙格 - 库塔法
```

其中，filename 是定义 f(t,y)的函数文件名，该函数文件必须返回一个列向量。tspan 形式为[t0,tf]，表示求解区间。y0 是初始状态列向量。t 和 y 分别给出时间向量和相应的状态向量。

【例 3-26】 设有初值问题

$$\begin{cases} y'=\dfrac{y^2-t-2}{4(t+1)} \\ y(0)=2 \end{cases}$$

试求其数值解，并与精确解相比较(精确解为 $y(t)=\sqrt{t+1}+1$)。

程序如下：

(1) 建立函数文件 funt.m。

```
function y = funt(t,y)
y = (y^2 - t - 2)/4/(t + 1);
```

(2) 求解微分方程。

```
t0 = 0;tf = 10;
```

```
y0 = 2;
[t,y] = ode23('funt',[t0,tf],y0);          % 求数值解
y1 = sqrt(t + 1) + 1;                      % 求精确解
plot(t,y,'b.',t,y1,'r - ');                % 通过图形来比较
legend('数值解','精确解');                  % 加图例
```

程序执行后,运行结果如图 3-5 所示,可以看出,两种结果近似。

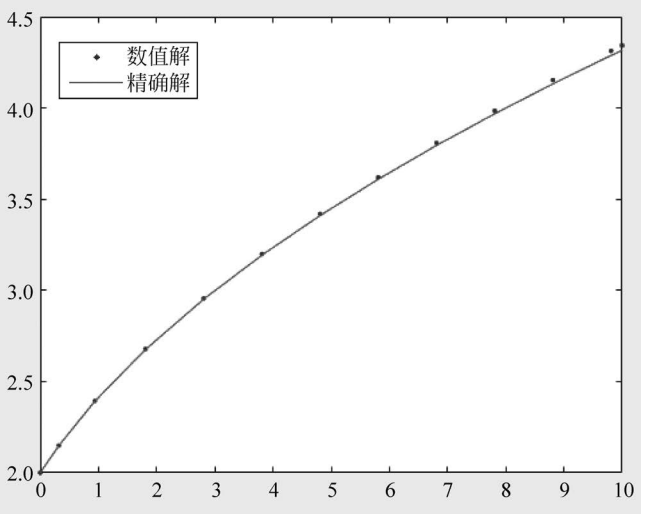


图 3-5　微分方程数值解和精确解的比较

3.4.4　数值积分

数值积分研究定积分的数值求解方法。MATLAB 提供了多种求数值积分的函数,主要包括一重积分和多重积分函数。

1. 一重数值积分

MATLAB 提供了 quad 函数和 quadl 函数,其调用格式如下:

```
quad(filename,a,b,tol,trace)
quadl(filename,a,b,tol,trace)                % 是一种采用自适应的方法
```

其中,filename 是被积函数名;a 和 b 分别是定积分的下限和上限;tol 用来控制积分精度,默认时取 tol=10^{-6};trace 控制是否展现积分过程,若取非 0 则展现积分过程,取 0 不展现,默认时取 trace=0。

【例 3-27】　用两种不同的方法求积分。

$$I=\int_0^1 \mathrm{e}^{-x^2}\,\mathrm{d}x$$

程序如下:

```
g = inline('exp( - x.^2)');          % 定义一个语句函数(内联函数)g(x) = exp( - x.^2)
I = quad(g,0,1)
I = quadl(g,0,1)
```

程序执行后，运行结果为

```
I =
   0.746824180726425
I =
   0.746824133988447
```

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，其调用格式如下：

```
dblquad(filename,xmin,xmax,ymin,ymax,tol,trace)
triplequad(filename,xmin,xmax,ymin,ymax,zmin,zmax,tol,trace)
```

函数的参数定义和一重积分一样。

【例 3-28】 计算二重定积分。

$$I=\int_{-1}^{1}\int_{-2}^{2}e^{-x^2/2}\sin(x^2+y)\,dx\,dy$$


微视频 3-7

程序如下：

```
fxy = inline('exp( - x.^2/2). * sin(x.^2 + y)');
I = dblquad(fxy, - 2,2, - 1,1)
```

程序执行后，运行结果为

```
I =
   1.574493189744944
```

3.5 应用实例

3.5.1 Van Der Pol 方程

【例 3-29】 1928 年，荷兰科学家范·德·波尔(Van Der Pol)为了描述 LC 电子管振荡电路，提出并建立了著名的 Van Der Pol 方程式

$$\frac{d^2y}{dt^2}-\mu(1-y^2)y'+y=0$$

它是一个具有可变非线性阻尼的微分方程，在自激振荡理论中具有重要意义。

用 MATLAB 的 ode45 函数求在 $\mu=10$，初始条件 $y(0)=1$，$\frac{dy(0)}{dt}=0$ 情况下该方程的解，并作出 $y\sim t$ 的关系曲线图和 $y\sim y'$ 相平面图。

(1) 把高阶微分方程改写为一阶微分方程组。

令 $y_1=y$，$y_2=y'$，则

$$\begin{bmatrix}\frac{dy_1}{dt}\\ \frac{dy_2}{dt}\end{bmatrix}=\begin{bmatrix}y_1\\ 10(1-y_1^2)y_2-y_1\end{bmatrix},\quad\begin{bmatrix}y_1(0)\\ y_2(0)\end{bmatrix}=\begin{bmatrix}0\\ 1\end{bmatrix}$$

(2) 定义 Van Der Pol 方程。

```
function y = VDP(t,y)
mu = 10;
y = [y(2);mu * (1 - y(1)^2) * y(2) - y(1)];
end
```

(3) 编写程序。

```
clear
t0 = [0,40];
y0 = [1;0];
[t,y] = ode45(@VDP,t0,y0);
subplot(1,2,1)
plot(t,y(:,1))
xlabel('t'),ylabel('y')
title('y(t) - t')
grid on
subplot(1,2,2)
plot(y(:,1),y(:,2))
xlabel('y(t)'),ylabel('y''(t)')
title('y''(t) - y(t)')
grid on
```

程序执行后,运行结果如图 3-6 所示。

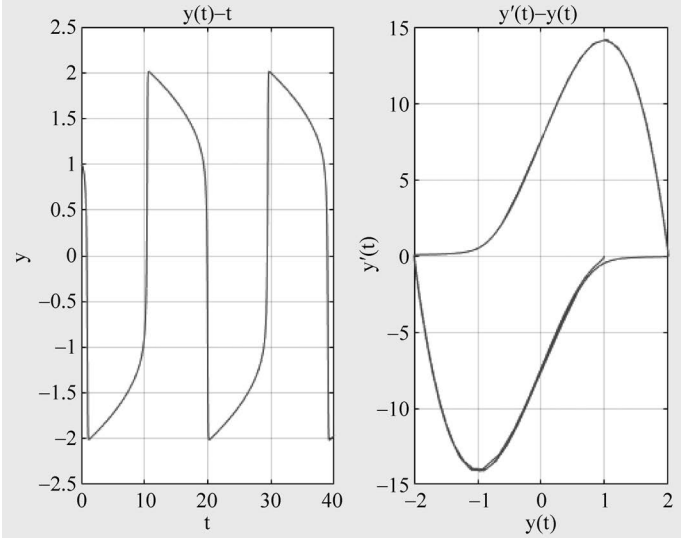


图 3-6 Van Der Pol 微分方程解

3.5.2 电荷在磁场中的运动

【例 3-30】 设一质量为 m、电荷量为 q 的粒子以速度 $\boldsymbol{v}_0$ 进入非均匀磁场 B 中,若速度的方向在 xy 平面内与 x 轴成 θ 角,磁感应强度的大小与 x 成正比,$\boldsymbol{B}=B_0x$,方向沿 x 正方向,如图 3-7 所示。试写出该带电粒子的运动方程,并描绘它在这一非均匀磁场中运动的轨道,重力忽略不计。

（1）该带电粒子在非均匀磁场中仅受到洛仑兹力的作用，因此运动方程为

$$m\frac{\mathrm{d}\boldsymbol{v}}{\mathrm{d}t}=q\,\boldsymbol{v}\times\boldsymbol{B}$$

将速度和磁场的向量 $\boldsymbol{v}=v_x i+v_y j+v_z k$ 及 $B=B_0 xi$ 代入上式，展开后得到

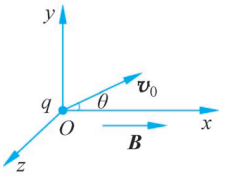


图 3-7 例 3-30 图

$$m\frac{\mathrm{d}v_x}{\mathrm{d}t}=0$$

$$m\frac{\mathrm{d}v_y}{\mathrm{d}t}=qB_0v_zx$$

$$m\frac{\mathrm{d}v_z}{\mathrm{d}t}=-qB_0v_yx$$

加上速度与位置的关系

$$\frac{\mathrm{d}x}{\mathrm{d}t}=v_x,\quad \frac{\mathrm{d}y}{\mathrm{d}t}=v_y,\quad \frac{\mathrm{d}z}{\mathrm{d}t}=v_z$$

在初始条件 $t=0$ 时，$x=y=z=0$ 以及 $v_x=v_0\cos\theta$，$v_y=v_0\sin\theta$，$v_z=0$ 下联立求解以上 6 个方程组成的一阶微分方程组，可以得到 3 个位置(x,y,z)及 3 个分速度(v_x,v_y,v_z)。为便于解方程，假设 $v_0=1000\mathrm{m/s}$，$\theta=30°$，$qB_0/m=100$。

（2）定义电荷在非均匀磁场中运动的微分方程。

```
% 电荷在非均匀磁场中运动的微分方程
function yp = dzc(t,y)
global A;                              % 定义全局变量
yp = [y(2),0,y(4),A * y(6) * y(1),y(6), - A * y(4) * y(1)]';   % 写入微分方程
```

（3）编写程序。

```
% 电荷在非均匀磁场中的运动
clear
global A;                              % 定义全局变量
A = 100;v = 1000;sita = pi/6;          % 设定 qBo/m,带电粒子的初速度及入射角
vx = v * cos(sita);vy = v * sin(sita); % 计算 x,y 方向的初速度
y0 = [0 vx 0 vy 0 0]';                 % 设定 t = 0 的初始条件
tspan = [0 0.1];                       % 设定积分时间
[t,y] = ode23('dzc',tspan,y0);         % 求解名为 dzc 的微分方程组
subplot(2,2,1)                         % 以下为描绘各方向的运动轨道
% plot(t,y(:,1));                      % 绘制一般二维曲线
comet(t,y(:,1));                       % 绘制二维动态轨线
xlabel('t');ylabel('x');
subplot(2,2,2)
plot(t,y(:,3));
comet(t,y(:,3));
xlabel('t');ylabel('y');
subplot(2,2,3)
% plot(t,y(:,5));
comet(t,y(:,5));
```

```
xlabel('t');ylabel('z');
subplot(2,2,4)
plot(y(:,3),y(:,5));
comet(y(:,3),y(:,5));
xlabel('y');ylabel('z');
figure                                          %新开图形窗口
 %plot3(y(:,1),y(:,3),y(:,5))                   %绘制一般三维曲线
comet3(y(:,1),y(:,3),y(:,5));                   %绘制三维动态轨线
xlabel('x');ylabel('y');zlabel('z');
```

程序执行后,运行结果如图 3-8 所示。

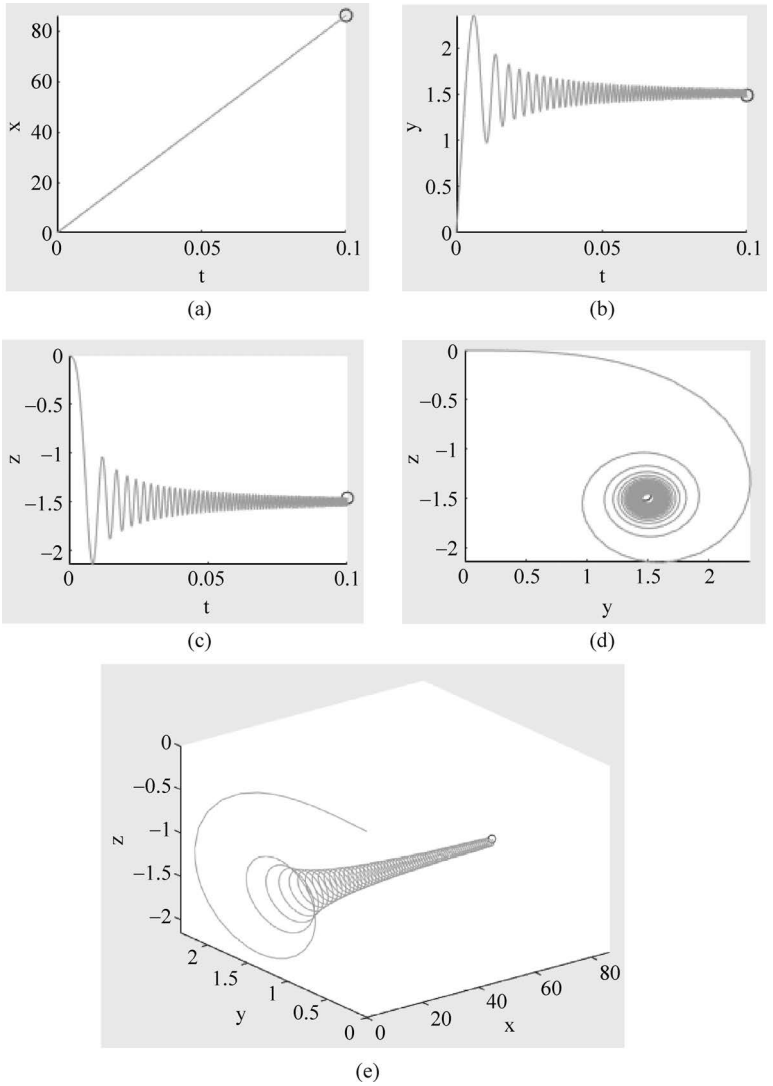


图 3-8　带电粒子在非均匀磁场中的运动轨迹

从程序的运行结果可以看出,在 x 方向带电粒子没有受到任何力,做匀速直线运动,如图 3-8(a)所示;在磁场与 x 成正比增强的洛伦兹力作用下,y 和 z 方向的轨迹呈半径逐步变小的螺旋形,如图 3-8(b)、图 3-8(c)、图 3-8(d)所示;图 3-8(e)则是在三维空间绘出的螺

旋形运动轨迹。

3.5.3 小球在空气中竖直上抛运动的规律

【例 3-31】 设一质量为 m 的小球以速率 v_0 从地面开始上抛。在运动过程中，小球所受空气阻尼的大小与速率成正比，比例系数为 k。

(1) 求小球上抛过程中的速度和高度与时间的关系。速度与高度有什么关系？小球上升的最大高度和到达最大高度的时间是多少？

(2) 求小球落回原处的时间和速度。

【解析】 (1) 小球竖直上升时受到重力和空气阻力，两者方向竖直向下，取竖直向上的方向为正，根据牛顿第二定律得到方程

$$f=-mg-kv=m\frac{\mathrm{d}v}{\mathrm{d}t}$$

分离变量，得

$$\mathrm{d}t=-m\frac{\mathrm{d}v}{mg+kv}=-\frac{m}{k}\frac{\mathrm{d}(mg+kv)}{mg+kv}$$

求积分得

$$t=-\frac{m}{k}\ln(mg+kv)\Big|_{v_0}^{v}=-\frac{m}{k}\ln\frac{mg+kv}{mg+kv_0}=-\frac{m}{k}\ln\frac{mg/k+v}{mg/k+v_0}$$

小球速度随时间的变化关系为

$$v=\left(v_0+\frac{mg}{k}\right)\exp\left(-\frac{k}{m}t\right)-\frac{mg}{k}$$

由于 $v=\dfrac{\mathrm{d}h}{\mathrm{d}t}$，所以

$$\mathrm{d}h=\left[\left(v_0+\frac{mg}{k}\right)\exp\left(-\frac{k}{m}t\right)-\frac{mg}{k}\right]\mathrm{d}t=-\frac{m(v_0+mg/k)}{k}\mathrm{d}\left[\exp\left(-\frac{k}{m}t\right)\right]-\frac{mg}{k}\mathrm{d}t$$

求积分得

$$h=-\frac{m(v_0+mg/k)}{k}\exp\left(-\frac{k}{m}t\right)-\frac{mg}{k}t\Big|_0^t$$

小球高度随时间的变化关系为

$$h=\frac{m(v_0+mg/k)}{k}\left[1-\exp\left(-\frac{k}{m}t\right)\right]-\frac{mg}{k}t$$

根据高度与时间的关系和速度与时间的关系，可得高度与速度的关系。

当小球上升到最高点时，其速度为零。小球到最高点所需要的时间为

$$T=\frac{m}{k}\ln\frac{mg/k+v_0}{mg/k}=\frac{m}{k}\ln\left(1+\frac{kv_0}{mg}\right)$$

小球上升的最大高度为

$$H=\frac{mv_0}{k}-\frac{m^2g}{k^2}\ln\left(1+\frac{kv_0}{mg}\right)$$

可见，小球上升到最高点的时间和高度由比例系数和初速度决定。在小球质量和空气阻力系数一定的情况下，初速度决定了小球的运动规律。

【讨论】 当 $k\to 0$ 时,利用公式 $e^x\to 1+x$,可得

$$v\to\left(v_0+\frac{mg}{k}\right)\left(1-\frac{k}{m}t\right)-\frac{mg}{k}=v_0\left(1-\frac{k}{m}t\right)-gt\to v_0-gt$$

这是不计空气阻力时竖直上抛运动的速度公式。

利用公式 $e^x\to 1+x+\frac{x^2}{2}$,可得

$$h\to\frac{m(v_0+mg/k)}{k}\left[\frac{kt}{m}-\frac{1}{2}\left(\frac{kt}{m}\right)^2\right]-\frac{mg}{k}t=v_0t-\frac{kv_0}{2m}t^2-\frac{1}{2}gt^2\to v_0t-\frac{1}{2}gt^2$$

这是不计空气阻力时竖直上抛运动的高度公式。

利用公式 $\ln(1+x)\to x$,可得

$$T\to\frac{m}{k}\frac{kv_0}{mg}\to\frac{v_0}{g}$$

这是不计空气阻力时小球竖直上升最大高度的时间。

利用公式 $\ln(1+x)\to x-\frac{x^2}{2}$,可得

$$H\to\frac{mv_0}{k}-\frac{m^2g}{k^2}\left[\frac{kv_0}{mg}-\frac{1}{2}\left(\frac{kv_0}{mg}\right)^2\right]\to\frac{v_0^2}{2g}$$

这是不计空气阻力时小球竖直上抛运动的最大高度。

【算法一】 用解析式。取特征时间 $\tau=\frac{m}{k}$ 为时间单位,取速度单位为 $V_0=\frac{mg}{k}$,则速度可表示为

$$v^*=(v_0^*+1)\exp(-t^*)-1$$

其中,$t^*=\frac{t}{\tau}$,$v_0^*=\frac{v_0}{V_0}=\frac{kv_0}{mg}$,$v^*=\frac{v}{V_0}$。$v_0^*$ 是无量纲的初速度,在小球质量与空气阻力系数一定的情况下,无量纲的初速度就代表了初速度。kv_0 是初始阻力,无量纲的初速度还是初始阻力与小球重力 mg 的比值。

取高度单位 $h_0=V_0\tau=\frac{m^2g}{k^2}$,高度公式可化为

$$h^*=(v_0^*+1)[1-\exp(-t^*)]-t^*$$

其中,$h^*=\frac{h}{h_0}$。小球上升到最高点所需要的时间可表示为

$$T^*=\ln(1+v_0^*)$$

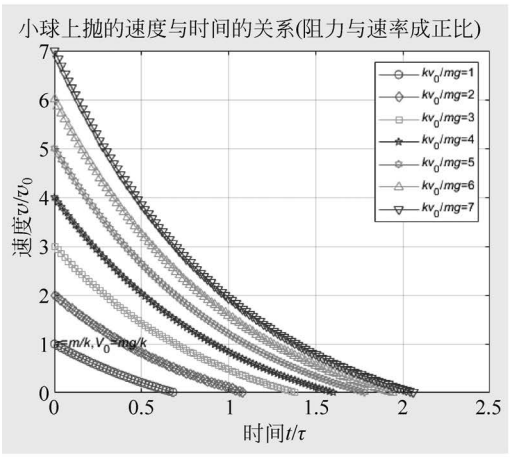
其中,$T^*=\frac{T}{\tau}$。小球上升的最大高度可表示为

$$H^*=v_0^*-\ln(1+v_0^*)$$

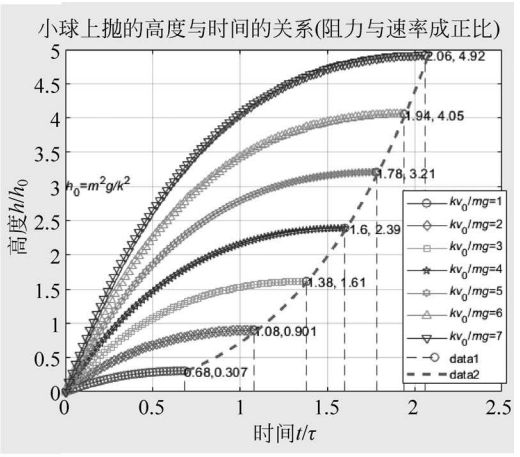
其中,$H^*=\frac{H}{h_0}$。取约化初速度为参数向量,取时间为自变量向量,形成矩阵,计算速度和高度与时间的关系,也确定了速度与高度的关系。用矩阵画线法画出速度和高度与时间的曲线族,并画出速度和高度的曲线族,程序如下:

```
%小球受到与速率成正比的摩擦阻力的上抛运动(用解析式)
clear                                          %清除变量
t = 0:0.02:2.5;                                %时间向量(t0 的倍数或无量纲时间)
v0 = 1:7;                                      %初速度向量
[V0,T] = meshgrid(v0,t);                       %初速度和时间矩阵
V = (V0 + 1). * exp( - T) - 1;                 %速度
H = (V0 + 1). * (1 - exp( - T)) - T;           %高度
n = length(v0);                                %初速度个数
H(V < 0) = nan;                                %速度小于 0 的高度改为非数
V(V < 0) = nan;                                %速度小于 0 的速度改为非数
figure                                         %创建图形窗口
%plot(t,V,'LineWidth',2)                       %画速度曲线族
plot(t,V(:,1),'o - ',t,V(:,2),'d - ',t,V(:,3),'s - ',t,V(:,4),'p - ',…
    t,V(:,5),'h - ',t,V(:,6),'^ - ',t,V(:,7),'v - ')   %画速度曲线族
grid on                                        %加网格
fs = 10;                                       %字体大小
title('小球上抛的速度与时间的关系(阻力与速率成正比)','FontSize',fs)   %显示标题
xlabel('时间\itt/\tau','FontSize',fs)          %显示横坐标标目
ylabel('速度\itv/V\rm_0','FontSize',fs)        %显示纵坐标标目
legend([repmat('\itkv\rm_0/\itmg\rm = ',n,1),num2str(v0')])%图例
text(0,1,'\it\tau\rm = \itm/k\rm,\itV\rm_0 = \itmg/k','FontSize',fs)   %时间和速度单位文本
figure                                         %创建图形窗口
%plot(t,H,'LineWidth',2)                       %画高度曲线族
plot(t,H(:,1),'o - ',t,H(:,2),'d - ',t,H(:,3),'s - ',t,H(:,4),'p - ',…
    t,H(:,5),'h - ',t,H(:,6),'^ - ',t,H(:,7),'v - ')%画高度曲线族
title('小球上抛的高度与时间的关系(阻力与速率成正比)','FontSize',fs)   %显示标题
xlabel('时间\itt/\tau','FontSize',fs)          %显示横坐标标目
ylabel('高度\ith/h\rm_0','FontSize',fs)        %显示纵坐标标目
grid on                                        %加网格
legend([repmat('\itkv\rm_0/\itmg\rm = ',n,1),num2str(v0')])   %图例
text(0,3,'\ith\rm_0 = \itm\rm^2\itg/k\rm^2','FontSize',fs)   %标记高度单位
[hm,im] = max(H);                              %求最大高度及其下标
hold on                                        %保持图像
stem(t(im),hm,' -- ')                          %画最高点的杆图
txt = [num2str(t(im)',3),repmat(',',n,1),num2str(hm',3)];   %运动时间和高度字符串
text(t(im),hm,txt,'FontSize',fs)               %标记时间和最大高度
vm = 1:0.1:7;                                  %较密的初速度向量
tm = log(1 + vm);                              %最长时间
hm = vm - log(1 + vm);                         %最高高度
plot(tm,hm,' -- ','LineWidth',2)               %画峰值线
figure                                         %创建图形窗口
%plot(H,V,'LineWidth',2)                       %画速度和高度曲线族
plot(H(:,1),V(:,1),'o - ',H(:,2),V(:,2),'d - ',H(:,3),V(:,3),'s - ',…
    H(:,4),V(:,4),'p - ',H(:,5),V(:,5),'h - ',H(:,6),V(:,6),'^ - ',…
    H(:,7),V(:,7),'v - ')                      %画速度和高度曲线族
title('小球上抛的速度与高度的关系(阻力与速率成正比)','FontSize',fs)   %显示标题
xlabel('高度\ith/h\rm_0','FontSize',fs)        %显示横坐标标目
ylabel('速度\itv/V\rm_0','FontSize',fs)        %显示纵坐标标目
grid on                                        %加网格
legend([repmat('\itkv\rm_0/\itmg\rm = ',n,1),num2str(v0')])   %图例
```

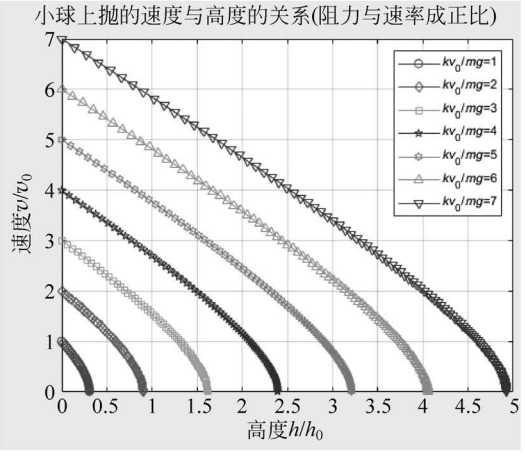
程序执行后,运行结果如图 3-9 所示。



(a)



(b)



(c)

图 3-9　小球在空气中竖直上抛运动的规律

由图 3-9(a)可以看出，不论初速度为多少，小球的速度随时间逐渐减小，直到 0 为止。初速度越大，小球到达最高点所需要的时间越长。

由图 3-9(b)可以看出，不论初速度为多少，小球的高度随时间增加，开始的时候几乎直线增加，在最高点附近缓慢增加，直到最大高度为止。初速度越大，小球达到的最大高度越高。

由图 3-9(c)可以看出，不论初速度为多少，小球的速度随高度减小，在最高点速度为 0。

说明：当小球运动到最高点之后，其运动方程发生了改变，只考虑小球上升的过程，其他运动时间的速度和高度就改为非数；为了画出最高点的连续曲线，需要设置较密集的初速度向量。

【算法二】 用两个一阶微分方程的数值解。

$$\frac{\mathrm{d}^2h}{\mathrm{d}t^2}=-g-\frac{k}{m}\frac{\mathrm{d}h}{\mathrm{d}t}$$

可化为

$$\frac{\mathrm{d}^2(h/h_0)}{\mathrm{d}(t/t_0)^2}=-g\frac{t_0^2}{h_0}-\frac{k}{m}t_0\frac{\mathrm{d}(h/h_0)}{\mathrm{d}(t/t_0)}=-1-\frac{\mathrm{d}(h/h_0)}{\mathrm{d}(t/t_0)}$$

即

$$\frac{\mathrm{d}^2 h^*}{\mathrm{d}t^{*2}} = -1 - \frac{\mathrm{d}h^*}{\mathrm{d}t^*}$$

设 $h(1)=h^*$ 和 $h(2)=\dfrac{\mathrm{d}h^*}{\mathrm{d}t^*}$，则可得

$$\frac{\mathrm{d}h(1)}{\mathrm{d}t^*} = h(2), \quad \frac{\mathrm{d}h(2)}{\mathrm{d}t^*} = -1 - h(2)$$

当 $t=0$ 时，$h=0$，因此初始条件 $h(1)=0$，而

$$h(2) = \frac{\mathrm{d}h^*}{\mathrm{d}t^*} = \frac{\mathrm{d}(h/h_0)}{\mathrm{d}(t/t_0)} = \frac{t_0}{h_0}\frac{\mathrm{d}h}{\mathrm{d}t} = \frac{\mathrm{d}h}{V_0\,\mathrm{d}t} = \frac{v_0}{V_0}$$

根据初始条件可求得常微分方程的数值解，程序如下：

```
% 小球受到与速率成正比的摩擦阻力的上抛运动(用二阶微分方程的数值解)
clear                                          % 清除变量
t = 0:0.02:2.5;                                % 时间向量(t0 的倍数或无量纲时间)
v0 = 1:7;                                      % 初速度向量
n = length(v0);                                % 初速度个数
H = [];                                        % 高度矩阵置空
V = [];                                        % 速度矩阵置空
for i = 1:n                                    % 按初速度循环
    [tt,HV] = ode45('fun',t,[0,v0(i)]);        % 求高度和速度的数值解
    V = [V,HV(:,2)];                           % 连接速度
    H = [H,HV(:,1)];                           % 连接高度
end                                            % 结束循环
H(V < 0) = nan;                                % 速度小于 0 的高度改为非数
V(V < 0) = nan;                                % 速度小于 0 的速度改为非数
figure                                         % 创建图形窗口
% plot(t,V,'LineWidth',2)                      % 画速度曲线族
plot(t,V(:,1),'o - ',t,V(:,2),'d - ',t,V(:,3),'s - ',t,V(:,4),'p - ',…
    t,V(:,5),'h - ',t,V(:,6),'^ - ',t,V(:,7),'v - ')   % 画速度曲线族
grid on                                        % 加网格
fs = 10;                                       % 字体大小
title('小球上抛的速度与时间的关系(阻力与速率成正比)','FontSize',fs)   % 显示标题
xlabel('时间\itt/\tau','FontSize',fs)          % 显示横坐标标目
ylabel('速度\itv/V\rm_0','FontSize',fs)        % 显示纵坐标标目
legend([repmat('\itkv\rm_0/\itmg\rm = ',n,1),num2str(v0')])   % 图例
text(0,1,'\it\tau\rm = \itm/k\rm,\itV\rm_0 = \itmg/k','FontSize',fs)   % 时间和速度单位文本
figure                                         % 创建图形窗口
% plot(t,H,'LineWidth',2)                      % 画高度曲线族
plot(t,H(:,1),'o - ',t,H(:,2),'d - ',t,H(:,3),'s - ',t,H(:,4),'p - ',…
    t,H(:,5),'h - ',t,H(:,6),'^ - ',t,H(:,7),'v - ') % 画高度曲线族
title('小球上抛的高度与时间的关系(阻力与速率成正比)','FontSize',fs)  % 显示标题
xlabel('时间\itt/\tau','FontSize',fs)          % 显示横坐标标目
ylabel('高度\ith/h\rm_0','FontSize',fs)        % 显示纵坐标标目
grid on                                        % 加网格
legend([repmat('\itkv\rm_0/\itmg\rm = ',n,1),num2str(v0')])   % 图例
text(0,3,'\ith\rm_0 = \itm\rm^2\itg/k\rm^2','FontSize',fs)    % 标记高度单位
[hm,im] = max(H);                              % 求最大高度及其下标
hold on                                        % 保持图像
```

```
stem(t(im),hm,'--')                                          % 画最高点的杆图
txt = [num2str(t(im)',3),repmat(',',n,1),num2str(hm',3)];    % 运动时间和高度字符串
text(t(im),hm,txt,'FontSize',fs)                             % 标记时间和最大高度
vm = 1:0.1:7;                                                % 较密的初速度向量
tm = log(1 + vm);                                            % 最大时间
hm = vm - log(1 + vm);                                       % 最大高度
plot(tm,hm,'--','LineWidth',2)                               % 画峰值线
figure                                                       % 创建图形窗口
% plot(H,V,'LineWidth',2)                                    % 画速度和高度曲线族
plot(H(:,1),V(:,1),'o-',H(:,2),V(:,2),'d-',H(:,3),V(:,3),'s-',…
    H(:,4),V(:,4),'p-',H(:,5),V(:,5),'h-',H(:,6),V(:,6),'^-',…
    H(:,7),V(:,7),'v-')                                      % 画速度和高度曲线族
title('小球上抛的速度与高度的关系(阻力与速率成正比)','FontSize',fs)   % 显示标题
xlabel('高度\ith/h\rm_0','FontSize',fs)                      % 显示横坐标标目
ylabel('速度\itv/V\rm_0','FontSize',fs)                      % 显示纵坐标标目
grid on                                                      % 加网格
legend([repmat('\itkv\rm_0/\itmg\rm = ',n,1),num2str(v0')])  % 图例
```

定义的函数文件为：

```
% 小球的约化速度和加速度函数
function f = fun(t,h)                    % 函数过程
f = [h(2);-1-h(2)];                      % 约化速度表达式和约化加速度表达式
```

程序执行后，运行结果同图 3-9。

【算法三】 用一阶微分方程的数值解。

$$\frac{\mathrm{d}v}{\mathrm{d}t}=-g-\frac{k}{m}v$$

可转化为

$$\frac{\mathrm{d}(v/V_0)}{\mathrm{d}(t/t_0)}=-g\frac{t_0}{V_0}-\frac{kt_0}{mV_0}v=-1-\frac{v}{V_0}$$

利用约化时间 $t^*=\frac{t}{\tau}$ 和约化速度 $v^*=\frac{v}{V_0}$，可得无量纲的微分方程

$$\frac{\mathrm{d}v^*}{\mathrm{d}t^*}=-1-v^*$$

微分方程的初始条件为：$t^*=0$ 时，$v^*=\frac{v_0}{V_0}$，高度为

$$h=\int_0^t v\,\mathrm{d}t=Vt\int_0^{t^*}v^*\,\mathrm{d}t^*=h_0\int_0^{t^*}v^*\,\mathrm{d}t^*$$

通过无量纲速度对无量纲时间的数值积分可求得高度与时间的关系。程序自行编写，结果同图 3-9。

【算法四】 用微分方程的符号解。

$$\frac{\mathrm{d}^2h}{\mathrm{d}t^2}+\frac{\mathrm{d}h}{\mathrm{d}t}+1=0$$

当 $t=0$ 时，$h^*=0$，$\dfrac{\mathrm{d}h^*}{\mathrm{d}t^*}=\dfrac{v_0}{V_0}$，根据初始条件可求得常微分方程的符号解，程序自行编写，结果同图 3-9。

【解析】 （2）当小球从最高点竖直下落时，空气阻力的方向向上，速度方向向下，取竖直向下的方向为正，根据牛顿第二定律得到方程

$$f=mg-kv=m\frac{\mathrm{d}v}{\mathrm{d}t}$$

分离变量，得

$$\mathrm{d}t=m\frac{\mathrm{d}v}{mg-kv}=-\frac{m}{k}\frac{\mathrm{d}(mg-kv)}{mg-kv}$$

积分得

$$t=-\frac{m}{k}\ln(mg-kv)\Big|_{v_0}^{v}=-\frac{m}{k}\ln\frac{mg-kv}{mg}=-\frac{m}{k}\ln\left(1-\frac{kv}{mg}\right)$$

小球速度随时间的变化关系为

$$v=\frac{mg}{k}\left[1-\exp\left(-\frac{k}{m}t\right)\right]$$

由于 $v=\dfrac{\mathrm{d}h}{\mathrm{d}t}$，所以

$$\mathrm{d}h=\frac{mg}{k}\left[1-\exp\left(-\frac{k}{m}t\right)\right]\mathrm{d}t=\frac{mg}{k}\mathrm{d}t+\frac{m^2g}{k^2}\mathrm{d}\left[\exp\left(-\frac{k}{m}t\right)\right]$$

积分得

$$h=\frac{mg}{k}-\frac{m^2g}{k^2}\left[1-\exp\left(-\frac{k}{m}t\right)\right]$$

这是小球下落的高度与时间的关系。

当小球落回原处时，$h=H$，则

$$\frac{k}{m}T'-1+\exp\left(-\frac{k}{m}T'\right)=\frac{kv_0}{mg}-\ln\left(1+\frac{kv_0}{mg}\right)$$

这是关于时间的超越方程。如果求得 T'，就可求得小球落回原处的速度。

【讨论】 当 $k\to0$ 时，利用公式 $\mathrm{e}^x\to1+x$，可得

$$v\to\frac{mg}{k}\frac{k}{m}t=gt$$

这是不计空气阻力时小球自由下落的速度公式。

利用公式 $\mathrm{e}^x\to1+x+\dfrac{x^2}{2}$，可得

$$h\to\frac{mg}{k}t-\frac{m^2g}{k^2}\left\{1-\left[1-\frac{k}{m}t+\frac{1}{2}\left(\frac{k}{m}t\right)^2\right]\right\}\to\frac{1}{2}gt^2$$

这是小球自由下落的高度公式。

利用公式 $\ln(1+x)\to x-\dfrac{x^2}{2}$，可得

$$\frac{k}{m}T'-1+\left[1-\frac{k}{m}T'+\frac{1}{2}\left(\frac{k}{m}T'\right)^2+\cdots\right]=\frac{kv_0}{mg}-\left[\frac{kv_0}{mg}-\frac{1}{2}\left(\frac{kv_0}{mg}\right)+\cdots\right]$$

化简得

$$T' \to \frac{v_0}{g}$$

这是不计空气阻力时小球自由下落到原处的时间。

【算法】 取特征时间 $\tau=\frac{m}{k}$ 为时间单位,取速度单位为 $V_0=\frac{mg}{k}$,则小球下落的速度为

$$v=V_0[1-\exp(-t^*)]$$

其中,$t^*=\frac{t}{\tau}$。当小球落回原处时关于时间的超越方程可表示为

$$t^*+\exp(-t^*)-(1+v_0^*)+\ln(1+v_0^*)=0$$

其中,$v_0=\frac{v}{V_0}$。此方程有一个解

$$t^*=-\ln(1+v_0^*)$$

由于 $t^*<0$,此解没有意义。将初速度和时间形成矩阵,计算高度差,利用等值线函数 contour 取零值线,即可求出小球落回原处的时间与初速度的数值解,进而求出小球落回原处的速度。

如果不计空气阻力,小球来回运动的时间可表示为

$$t=2\frac{v_0}{g}=\tau 2\frac{v_0\tau}{g}=\tau 2v_0^*$$

小球在空气中来回运动的时间可与无空气阻力的情况相比较。程序如下:

```
% 小球受到与速率成正比的摩擦阻力作用时上升的高度和落回原处的时间以及速度
clear                                          % 清除变量
v0 = 0:0.05:7;                                 % 较密的初速度向量
h = v0 - log(1 + v0);                          % 最大高度
figure                                         % 创建图形窗口
plot(v0,h,'LineWidth',2)                       % 画最大高度曲线
grid on                                        % 加网格
fs = 10;                                       % 字体大小
title('小球上升的最大高度与上升初速度的关系','FontSize',fs)      % 显示标题
xlabel('上升初速度\itv\rm_0/\itV\rm_0','FontSize',fs)            % 显示横坐标标目
ylabel('最大高度\itH/h\rm_0','FontSize',fs) % 显示纵坐标标目
txt1 = '\ith\rm_0 = \itm\rm^2\itg/k\rm^2';     % 高度单位文本
txt2 = '\itV\rm_0 = \itmg/k';                  % 速度单位文本
txt3 = '\it\tau\rm = \itm/k\rm';               % 时间单位文本
text(0,3,[txt1,',',txt2],'FontSize',fs)        % 标记高度单位和速度单位
t = 0:0.05:1;                                  % 落回时间向量
[V0,T] = meshgrid(v0,t);                       % 初速度和时间矩阵
H = T + exp( - T) - 1 - V0 + log(1 + V0);      % 下落的高度差函数
figure                                         % 创建图形窗口
h = contour(V0,T,H,[0,0]);                     % 高度差为零的落回时间与初速度等值线
v0 = h(1,2:end);                               % 取初速度
t2 = h(2,2:end);                               % 取落回时间
t1 = log(1 + v0);                              % 上升时间
 % plot(v0,[t1;t2;t1 + t2;2 * v0])              % 画时间曲线
plot(v0,t1,'o - ',v0,t2,'d - ',v0,t1 + t2,'s - ',v0,2 * v0,'^ - ')      % 画时间曲线
```

```
legend('上升到最高点的时间\itT/\tau','落回原处的时间\itT\prime/\tau',…
    '上升和落回的总时间','无空气阻力上升和落回的总时间',num2str(v0'))  % 图例
title('小球运动的时间与上升初速度的关系','FontSize',fs)                  % 显示标题
xlabel('上升初速度\itv\rm_0/\itV\rm_0','FontSize',fs)                    % 显示横坐标标目
ylabel('时间\itT/\it\tau','FontSize',fs)            % 显示纵坐标标目
text(0,1,[txt3,',',txt2],'FontSize',fs)             % 标记时间单位和速度单位
grid on                                             % 加网格
v2 = 1 - exp( - t2);                                % 落回的速度
figure                                              % 创建图形窗口
plot(v0,v2,'LineWidth',2)                           % 画有空气阻力的速度曲线
grid on                                             % 加网格
axis equal                                          % 使坐标间隔相等
title('小球上抛后落回原处的速度与上升初速度的关系','FontSize',fs)          % 显示标题
xlabel('上升初速度\itv\rm_0/\itV\rm_0','FontSize',fs)                    % 显示横坐标标目
ylabel('落回末速度\itv/itV\rm_0','FontSize',fs)                          % 显示纵坐标标目
text(0,0,txt2,'FontSize',fs)                        % 标记速度单位
hold on                                             % 保持图像
plot(v0,v0,'r -- ','LineWidth',2)                   % 画无空气阻力的速度曲线
legend('有空气阻力','无空气阻力',num2str(v0'))      % 图例
```

程序执行后，运行结果如图 3-10 所示。

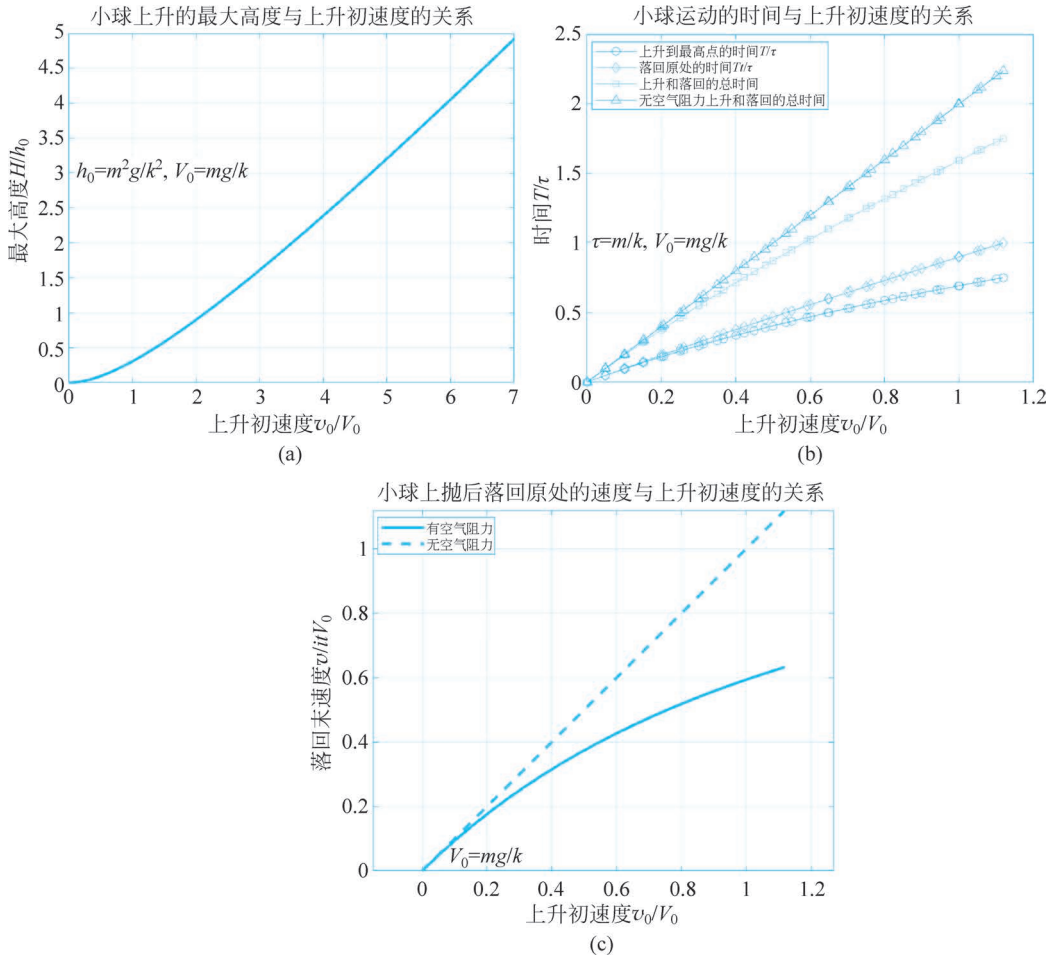


图 3-10 小球在空气中竖直上抛运动的规律

由图 3-10(a)可以看出,上升的初速度越大,小球上升的高度就越大。当初速度比较大时,小球上升的高度随初速度几乎呈直线增加。

由图 3-10(b)可以看出,上升的初速度越大,小球达到最大高度所需要的时间越多,但是少于小球从最高点下落到原处的时间。小球在空气中上升和下落到原处的总时间比不计阻力的时间要短。

由图 3-10(c)可以看出,上升的初速度越大,小球回到原处的速度越大,与小球上升的初速度相差也越大。

说明:画等值线主要是为了取句柄;句柄有两行若干列,第一行的第一个数值表示等值线数据对的数目,第二行的第一个数值表示等值线的值,第一行的其他数值表示速度,第二行的其他数值表示时间;画时间曲线时就抹去了等值线。

实训项目三

本实训项目的目的如下:

- 掌握多项式的常用运算。
- 掌握数值插值的方法及应用。
- 掌握数据统计和分析的方法。
- 掌握求数值导数和数值积分的方法。
- 掌握常微分方程数值求解的方法。

3-1 利用 MATLAB 提供的 rand 函数生成 2000 个符合均匀分布的随机数,然后检验随机数的性质:

(1) 均值和标准方差。

(2) 最大元素和最小元素。

(3) 大于 0.5 的随机数个数占总数的百分比。

3-2 将 100 个学生 5 门课的成绩存入矩阵 $\boldsymbol{P}$ 中,进行如下处理:

(1) 分别求每门课的最高分、最低分及相应学生序号。

(2) 分别求每门课的平均分和标准方差。

(3) 5 门课总分的最高分、最低分及相应学生序号。

(4) 将 5 门课总分按从大到小顺序存入 zongchengji 中,相应学生序号存入 xsxh。

提示:上机时,可用取值范围为[45,95]的随机矩阵来表示学生的成绩。

3-3 某气象站观测的某日 6:00—18:00 每隔 2h 的室内外温度(℃)如表 3-8 所示。

表 3-8 室内外温度观测结果

时间/h	6	8	10	12	14	16	18
室内温度 t_1/℃	18.0	20.0	22.0	25.0	30.0	28.0	24.0
室外温度 t_2/℃	15.0	19.0	24.0	28.0	34.0	32.0	30.0

试用三次样条插值分别求出该室内外 6:30—17:30 每隔 2h 各点的近似温度(℃)。

3-4 某电路元件,测试两端的电压 U 与流过的电流 I 的关系,实测数据如表 3-9 所示。用不同的插值方法(最近点法、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 3-9 实测数据

电流 I/A	0	2	4	6	8	10	12
电压 U/V	0	0	5	8.2	12	16	21

3-5 已知多项式 $P_1(x)=3x+2$，$P_2(x)=5x^2-x+2$，$P_3(x)=x^2-0.5$，求：

(1) $P(x)=P_1(x)P_2(x)P_3(x)$。

(2) $P(x)=0$ 的全部根。

(3) 计算 $x=0.2i(i=0,1,2,\cdots,10)$各点上的 $P(x_i)$。

3-6 求函数在指定点的数值导数。

$$f(x)=\begin{vmatrix} x & x^2 & x^3 \\ 1 & 2x & 3x^2 \\ 0 & 2 & 6x \end{vmatrix}, \quad x=1,2,3$$

3-7 用数值方法求定积分。

(1) $I_1=\int_0^{2\pi}\sqrt{\cos t^2+4\sin(2t)^2+1}\,\mathrm{d}t$ 的近似值。

(2) $I_2=\int_0^1\frac{\ln(1+x)}{1+x^2}\mathrm{d}x$

(3) $I_3=\int_0^{\pi}\int_0^{\pi}|\cos(x+y)|\,\mathrm{d}x\,\mathrm{d}y$

3-8 求函数在指定区间的极值。

(1) $f(x)=\frac{x^3+\cos x+x\log x}{\mathrm{e}^x}$在(0,1)内的最小值。

(2) $f(x_1,x_2)=2x_1^3+4x_1x_2^3-10x_1x_2+x_2^2$ 在[0,0]附近的最小值点和最小值。

3-9 求微分方程组的数值解，并绘制解的曲线。

$$\begin{cases} y_1'=y_2y_3 \\ y_2'=-y_1y_3 \\ y_3'=-y_1y_2 \\ y_1(0)=0,y_2(0)=1,y_3(0)=1 \end{cases}$$

3-10 两个同心导体球面的内半径为 R_0，外半径为 R，构成球形电容器，球面间充满介电常数为 ε 的各向同性的介质。求球形电容器的电容(内球面也可以用同样半径的球体代替)。