第五章 数值计算

5.1 线性代数

5.1.1 LU 分解

1. LU 分解

一个矩阵可以分解为一个上三角矩阵和一个下三角矩阵的乘积，称之为 LU 分解。LU 分解是用高斯主元消去法实现的，通常要对主元位置进行交换，主元交换的方法是将被分解矩阵左乘一个由 0-1 构成的行交换阵。

【调用格式】

[L, U, P] = lu(X) 对矩阵 X 进行 LU 分解，并进行主元交换，

[L, U] = lu(X) 对矩阵 X 进行 LU 分解，无主元交换

【说明】L 为主对角元素为 1 的下三角矩阵，U 为上三角矩阵，P 为行交换矩阵。

2. 行列式和求逆

矩阵的行列式和求逆可以通过 LU 分解的方法求解。

【调用格式】

d = det(X) 求矩阵 X 的行列式

Y = inv(X) 求矩阵 X 的逆矩阵

例 5.1.1 对矩阵进行 LU 分解、行列式和求逆操作。

```
>> A=[1,2,3; 2,2,3;9,7,5];
>> [L1,U1]=lu(A);          %不带主元交换的 LU 分解，L1 通常不是下三角阵
>> [L2,U2,P]=lu(A)         %带主元交换的 LU 分解，L2 为下三角阵
L2 =
    1.0000         0         0
    0.1111    1.0000         0
    0.2222    0.3636    1.0000
U2 =
    9.0000    7.0000    5.0000
         0    1.2222    2.4444
         0         0    1.0000
P =
     0     0     1
     1     0     0
     0     1     0
>> det(A)                          %行列式值
ans =    11
>> Y=inv(A)                        %矩阵求逆
Y =
   -1.0000    1.0000   -0.0000
    1.5455   -2.0000    0.2727
   -0.3636    1.0000   -0.1818
```

5.1.2 特征值和特征向量

eig 函数用于求解矩阵的特征值和特征向量。

【调用格式】

D=eig(A) 计算矩阵 A 的特征值，D 为特征值构成的向量

[V , D]=eig(A) 计算矩阵 A 的特征值对角阵 D 和特征向量矩阵 V

[V , D]=eig(A , 'nobalance') 当矩阵 A 中有与截断误差近似的数值，用本指令

例 5.1.2 特征值分解的精度

```
>> B = [3, -2, -0.9, 2*eps; -2, 4, 1, -eps; -eps/4, eps/2, -1, 0;   -0.5,   -.5,   0.1, 1 ];
>> [VB,DB] = eig(B);
>> B*VB - VB*DB;                          %本例不采用 nobalance 参数会产生很大的计算误差
>> [VN,DN] = eig(B,'nobalance');          %计算误差为 1.0e-014 数量级
>> B*VN - VN*DN
ans =
   1.0e-014 *
    -0.2665         0   -0.0323   -0.0028
     0.4441    0.1110    0.0042   -0.0250
     0.0022    0.0002    0.0007         0
     0.0056   -0.0444    0.0444    0.0083
```

5.1.3 奇异值分解

1.矩阵的奇异值分解

任意矩阵 A 可进行奇异值分解，即存在酉矩阵 U 和 V，使下面等式成立

$$U^T AV = S = diag(\sigma_1, \sigma_2, ..., \sigma_p)$$

其中 $\sigma_1, \sigma_2, ..., \sigma_p$ 称为矩阵的奇异值。

【调用格式】

s = svd(A) 求矩阵 A 的奇异值，s 为由奇异值构成的向量

[U , S , V] = svd(A) 矩阵 A 的奇异值分解

2.矩阵结构特征的奇异值描述

矩阵的奇异值可以描述矩阵的结构特征。有关矩阵结构特征的 MATLAB 函数有如下几种。

r = rank(A, tol) 在指定容差 tol 下，求矩阵 A 的秩。tol 可以省略。

Z = null(A) 求矩阵 A 的零空间。

V = orth(A) 求矩阵 A 的值空间。

n = norm(A) 求矩阵 A 的 2 范数。

n = norm(A ,p) 求矩阵 A 的各种范数。

c = cond(X, p) 求矩阵 A 的条件数，p 可以省略。

theta = subspace(A, B) 求 A 和 B 矩阵所张子空间的夹角。

B = pinv(A, tol) 在指定容差 tol 下，求矩阵 A 的广义逆，tol 可以省略。

5.1.4 线性方程组的解

在线性方程组中，独立方程的个数等于独立未知参数的个数，称为恰定方程；独立方程的个数大于独立未知参数的个数，称为超定方程；独立方程的个数小于独立未知参数的个数，称为欠定方程。形如 $Ax=b$ 的线性方程组可以用以下方法求解：

1. 左除运算符法

左除运算符法的形式为

x=A\b

对于一般的非奇异矩阵 A，可以求得唯一数值解。欠定方程和超定方程，可以获得最小二乘解。

2. 广义逆法

如果用左除运算符求解的时候出现提示矩阵A为非奇异的警告或者解中出现Nan，则可以采用广义逆法，形式为

x=pinv(A)*b

3. 符号计算法

可以求得方程组的符号解，对于欠定方程可以求得具有自由变量的解。

例 5.1.3 求以下 3 个方程组的解

I：$\begin{cases}4x+6y+3z=13\\2x+3y+4z=9\\5x+2y+3z=10\end{cases}$ II：$\begin{cases}4x+6y+3z=13\\2x+3y+4z=9\end{cases}$ III：$\begin{cases}4x+6y=13\\2x+3y=9\\5x+2y=10\end{cases}$

解：方程 I 为恰定方程，用左除运算符可以求解

```
>> A1=[4,6,3; 2,3,4; 5,2,3];   b1=[13; 9; 10];
>> x1=A1\b1
x1 =
  1
  1
  1
```

方程 II 为欠定方程，用 3 种方法分别求解

```
>> A2=A1; b2=b1;
>> A2(3 , : )=[];   b2(3)=[];
>> x2=A2\b2                    %左除运算符法，求得一组特解，解中非零元素最少
x2 =
         0
    1.6667
    1.0000
>> x2=pinv(A2)*b2              %广义逆法，求得一组特解
x2 =
    0.7692
    1.1538
    1.0000
>> x2=solve('4*x+6*y+3*z=13,2*x+3*y+4*z=9','x,y,z');  %符号运算法求通解
>> x2=[x2.x; x2.y; x2.z]
x2 =
    5/2-3/2*y
        y
```

1

方程 III 为超定方程，用左除运算符法求最小二乘解

```
>> A3=A1; b3=b1;
   >> A3(: , 3 )=[];
   >> x3=A3\b3
   x3 =
        1.4545
        1.3636
```

5.2 函数分析

5.2.1 函数的零点

1. 多项式的根

通过 roots 函数来求取多项式全部的根。

【调用格式】

r = roots(p)　　多项式求根函数

【说明】p 为多项式的系数行向量，r 为多项式所有根构成的列向量。

2. 一元函数零点

fzero 函数求取一元函数的精确零点。

【调用格式】

[x, fval, exitflag, output] = fzero(fun, x0, options)　一元函数零点的完整调用格式

x = fzero(fun, x0)　　一元函数零点的最简调用格式

【说明】

1. fzero 只能求得 x0 附近的单个零点，不能求取函数的所有零点。
2. 输入变量 fun 表示一元函数，可以是字符串、内联函数或者函数句柄。
3. 输入变量 x0 为零点的初始猜测值（自变量值）。如果 x0 为标量，则求距离 x0 最近的那个零点；如果 x0=[a, b]，要求 fun(a)和 fun(b)异号，此时求自变量在[a, b]区间内的零点。
4. 输入变量 options 为优化迭代选项，是一个结构体。
5. 输出变量 x 为零点处的自变量值，输出变量 fval 为零点处的函数值。
6. 输出变量 exitflag 表示函数中止计算的条件。若 exitflag>0 表示找到零点后退出。

输出变量 output 表示程序运行信息，是一个结构体。

例 5.2.1 求函数 $f(t)=1-12.5e^{-t}\sin(2t+3.4)$ 在 $t>0$ 区间内的所有零点。

解：运行下列程序，观察执行结果。

```
>> y=inline('1-12.5*exp(-t)*sin(2*t+3.4)', 't');      %构造内联函数
>> t=0:0.1:10;                                        %定义自变量取值区间
>> yv=feval(vectorize(y), t);                         %计算自变量区间的函数值
>> plot(t,yv);                                        %绘制函数曲线，如图 5.2.1 所示
>> axis tight;
>> grid on;
>> [tt,yy]=ginput                  %在函数曲线上用鼠标拾取所有零点，按回车键结束
tt =
    1.6475
    2.4309
```

yy =

0.0085

0.0085

>> **[t1,fv1,flag]=fzero(y,tt(1));** %求第一个拾取点的精确零点

>> **[t2,fv2,flag]=fzero(y,tt(2));** %求第二个拾取点的精确零点

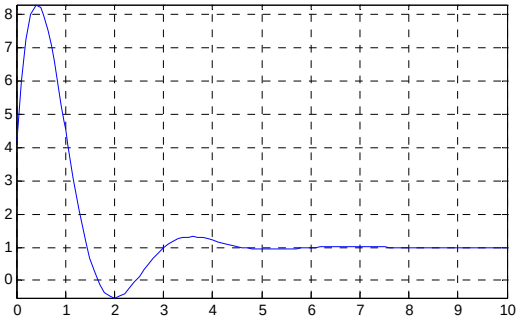

图 5.2.1 函数曲线

3.多元函数的零点

fsolve 函数求取多元函数的精确零点，但是必须提供零点的大致位置才能进行数值搜索。

【调用格式】

[x, fval, exitflag, output] =fsolve(fun, x0, options) 多元非线性方程求解

x = fsolve (fun, x0) 多元非线性方程求解的最简格式

【说明】

1. 输入变量 fun 表示自变量为向量的函数，可以是字符串、内联函数或者函数句柄。
2. 输入变量 x0 为零点的初始猜测向量（自变量值）。
3. 输出变量 x 为零点处的自变量值，输出变量 fval 为零点处的函数值。

例 5.2.2 求方程组 $\begin{cases} 2x_1 - x_2 - e^{-x_1} = 0 \\ -x_1 + 2x_2 - e^{-x_2} = 0 \end{cases}$ 在 $x_1 = -5, x_2 = -5$ 附近的解。

解：（1）采用字符串表达函数

>> **fun='[2*x(1)-x(2)-exp(-x(1)), -x(1)+2*x(2)-exp(-x(2))]';** %字符串方式表达函数

>> **x0=[-5;5];**

>> **[x, fval] = fsolve(fun,x0)**

（2）采用内联函数表达函数

>> **fun=inline('[2*x(1)-x(2)-exp(-x(1)), -x(1)+2*x(2)-exp(-x(2))]', 'x');**

>> **x0=[-5;5];**

>> **[x,fval] = fsolve(fun,x0)**

（3）采用 M 文件的函数句柄表达函数

编写 f1.m 的文件内容如下：

```
function fv=fun(x)
fv(1)= 2*x(1)-x(2)-exp(-x(1));
fv(2)= -x(1)+2*x(2)-exp(-x(2));
```

[x,fval] = fsolve(@f1, x0) %求零点

5.2.2 函数的极值点

MATLAB 提供了 3 个求极值点的函数，其输入输出参数的定义和 fsolve 函数基本相同。

【调用格式】

[x, fval, exitflag, output] = fminbnd(fun, x1, x2, options)

【说明】求一元函数 fun 在自变量（x1，x2）区间的最小值

【调用格式】

[x, fval, exitflag, output] = fminsearch (fun, x0, options)

【说明】：用单纯形法求多元函数 fun 在自变量向量 x0 附近的极小值点

【调用格式】

[x, fval, exitflag, output] = fminunc (fun, x0, options)

【说明】：用拟牛顿法求多元函数 fun 在自变量向量 x0 附近的极小值点

例 5.2.3 求二元函数 $f(x,y)=100(y-x^2)^2+(1-x)^2$ 在 $x=-1.2, y=1$ 附近的极小值。

解：运行下列程序，观察运行结果。

```
>> fun=inline('100*(x(2)-x(1)^2)^2+(1-x(1))^2', 'x');
>> [x,fval] = fminsearch(fun, [-1.2, 1])
x =
   1.0000    1.0000
fval =
   8.1777e-010
```

5.2.3 数值微分

数值导数和微分是基于差分定义的，对原始数据的采样间隔依赖很大，如果原始数据含有噪声，则数值导数结果没有什么价值，因此尽量避免求数值导数，diff 函数用于求相邻数据间的数值差分。

【调用格式】

DX = diff(X) 求 X 相邻元素的一阶差分，即用后一个元素值减去当前元素值

DX= diff(X, n) 求 X 相邻元素的 n 阶差分

DX = diff(X, n, dim) 在 dim 指定的维上，求 X 相邻元素的 n 阶差分

【说明】

1. 如果 X 是向量，差分是相邻元素相减；如果 X 是矩阵，差分是相邻行间对应元素相减。
2. 差分数据 DX 元素的个数要比被操作数 X 少一个。
3. DX 只是差分数据，如果相邻数据点之间的步长为 DH，则 DX./DH 为导数数据。

例 5.2.4 绘制函数 $f(x)=\sin x$ 的导函数在 $(0,2\pi)$ 区间的曲线。

解：运行下列程序， $f'(x)$ 的曲线如图 5.2.2 中的实线所示，虚线为 $f(x)$ 的曲线。

```
>> dx=2*pi/50;   x=0:dx:2*pi;   y=sin(x);
>> dy=diff(y);                  %求差分数据
>> df=dy./dx;                   %求导数数据
>> y=y(1:end-1); x=x(1:end-1);        %差分数据少 1 个，裁剪原始函数的相关数据
>> plot(x,y,'r--',x,df,'b'),   axis tight,   grid on
```

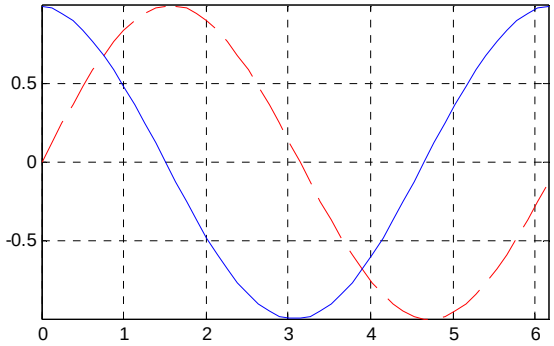

图 5.2.2 函数的导函数曲线

5.2.4 数值积分

数值积分分为开型积分和闭型积分，二者的区别在于是否计算积分区间端点处的函数值。MATLAB提供的数值积分函数有些适用于开型积分和闭型积分，有些只能用于闭型积分。积分计算可以采用本节介绍的相关函数，也可以采用样条积分法，还可以采用符号积分法，后两种方法在后面章节介绍。

1.一元函数的数值积分

求一元函数数值积分的函数有很多，每个函数采用不同的数值算法，各有优缺点，精度和运算速度也不尽相同，quadl 函数为最常用的数值积分函数。

【调用格式】

q = quadl(fun, a, b, tol, trace)　　采用递推自适应 Lobatto 法计算一元函数的积分

【说明】

(1) fun 为被积函数，可以用字符串、内联函数和函数 M 文件的函数句柄表示，且被积函数表达式要按照数组运算规则来编写。通常积分变量采用字母 x。

(2) a 和 b 为积分变量的积分上下限，为常数数值。

(3) tol 为绝对误差，是一个标量，可以省略。

(4) trace 为跟踪标志，当 trace 为非零值时，随积分进程会逐点画出被积函数。

(5) 返回值 q 为数值积分的结果。

例 5.2.5 求积分 $\int_0^2 \frac{1}{x^3-2x+5}dx$

解：运行下列程序，观察运行结果。

```
>> fun=inline('1./(x.^3-2.*x+5)','x');
>> q=quadl(fun,0,2)
q =
    0.4213
```

2.二重数值积分

形如 $\int_{y1}^{y2}\left[\int_{x1}^{x2} f(x,y)dx\right]dy$ 的闭型二重数值积分的函数为 dblquad。

【调用格式】

q = dblquad(fun, x1 ,x2, y1, y2, tol, method)

【说明】

(1) fun 为被积函数 $f(x,y)$，可以用字符串、内联函数和函数 M 文件的函数句柄表示。积分变量用 x，y 表示， x 为内重积分变量，y 为外重积分变量。

(2) x2 和 x1 是变量 x 的积分上下限，y2 和 y1 是变量 y 的积分上下限。

(3) method 表示选用的积分算法，可以省略，缺省算法是@quad，还可以选则@quadl 或者用户自定义的积分算法 M 函数文件的函数句柄。

(4) 本函数无法计算内重积分上下限为函数表达式的情况。

3. 三重数值积分

求形如 $\int_{z1}^{z2}\left[\int_{y1}^{y2}\left[\int_{x1}^{x2} f(x,y,z)dx\right]dy\right]dz$ 的闭型三重数值积分的函数为 triplequad。

【调用格式】

q = triplequad(fun, x1 ,x2, y1, y2, z1, z2, tol, method))

【说明】

(1) fun 为被积函数 $f(x,y,z)$，可以用字符串、内联函数和函数 M 文件的函数句柄表示。积分变量用 x，y，z 表示， 从内到外积分变量依次为 x，y，z。

(2) x2 和 x1 是变量 x 的积分上下限，y2 和 y1 是变量 y 的积分上下限，z2 和 z1 是变量 z 的积分上下限。

(3) 其他事项同函数 dblquad。

例 5.2.6 计算定积分 $\int_{-1}^{1}\left[\int_{0}^{1}\left[\int_{0}^{\pi}(y\cdot\sin(x)+z\cdot\cos(x))dx\right]dy\right]dz$

解：运行下列程序，观察运行结果。

```
>> fun=inline('y*sin(x)+z*cos(x)','x','y','z');
>> q = triplequad(fun,0,pi,0,1,-1,1)
q =
    2.0000
```

5.3 数据拟合

已知某些离散的原始数据，建立一个曲线方程，让它以最佳的方式反映原始数据的变化趋势，尽量避免出现局部的波动，这种方法称为拟合。不要求拟合曲线经过所有的原始数据，最佳方式通常是指用拟合得到的数学模型计算出来的计算数据和原始数据之间的误差的平方和最小，这种方式称为最小二乘。通常是已知曲线方程的形式（数学模型的结构），根据原始数据来计算曲线方程的参数（数学模型的参数）。本节介绍 MATLAB 中如何实现最小二乘拟合。

5.3.1 多项式拟合

多项式拟合是用一个 n 阶多项式模型去拟合原始数据的方法，需要计算的模型参数包括多项式的阶次和多项式的系数，polyfit 函数实现多项式拟合。

【调用格式】

p = polyfit (x, y, n) 根据给定的数据(x,y)，计算 n 阶拟合多项式的系数向量 p

ye = polyval (p, x) 计算自变量为 x 时多项式 p 的值（估计值）

【说明】多项式拟合一般不要超过 5 阶，否则计算误差变大；拟合多项式只在原始数据范围内可以保证精度，超出范围使用拟合多项式无法保证预报的精度。

例 5.3.1 已知五个数据点：[1,5.5],[2,43],[3,128],[4,290],[5,498]，试画出这五个点拟合的三次曲线。

解：运行下列程序，得到相应的拟合曲线，如图 5.3.1 所示，图中圆圈表示原始数据。

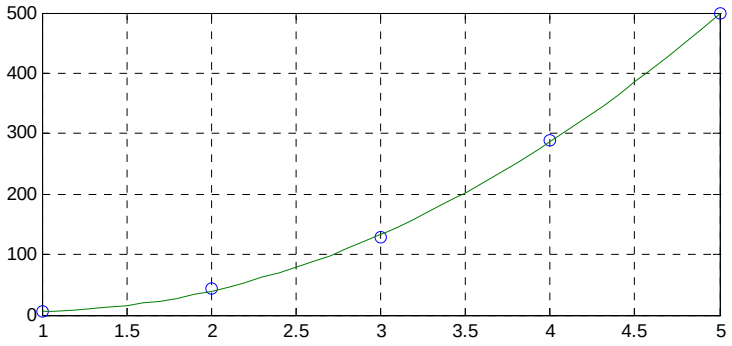

图 5.3.1 多项式曲线拟合

```
>> x=[1,2,3,4,5]; y=[5.5,43,128,290,498];
>> p=polyfit(x,y,3)
p=
    -0.1250
     30.9821
    -58.8929
     34.4000
>> x2=1:0.1:5;   y2=polyval(p,x2);
>> plot(x,y,'o',x2,y2) , grid on
```

5.3.2 最小二乘拟合

1.线性最小二乘拟合

对于线性数学模型的参数估计，可以用形如 Y=Ax+b 的一阶多项式拟合来估计参数。某些非线性模型经过变量替换也可以转换为线性模型，也可以采用线性估计方法，lsqlin 函数实现线性最小二乘估计。

【调用格式】

a= lsqlin(Fun, a0, lb, ub, options, p1, p2, ...)

例 5.3.2 对于非线性数学模型 $y = ae^{bx}$ ， a,b 为模型参数，采用变量替换将其转换为线性数学模型。

解：对已知数学模型取自然对数有　　$\ln y = \ln a + bx$

令 $Y = \ln y, X = x, a1 = \ln a, a2 = b$ ，则有 $Y = a_2 X + a_1$ 的线性数学模型

2. 非线性最小二乘估计

可以使用 MATLAB 提供的 lsqnonlin 函数实现非线性最小二乘估计。

【调用格式】

[a, resnorm, residual, exitflag, output] = lsqnonlin(Fun, a0, lb, ub, options, p1, p2, ...)

a= lsqnonlin(Fun, a0)

【说明】

(1) 函数的功能是求取向量 a 的值，使向量函数 $Fun(a) = \begin{bmatrix} f_1(a) \\ \cdots \\ f_k(a) \end{bmatrix}$ 满足

$$\min_a \frac{1}{2}\left\|Fun(a)\right\|_2^2 = \frac{1}{2}\min_a \sum_{i=1}^{k} f_i^{\,2}(a)$$

(2) 输入参数 Fun 是被解函数，Fun 是向量函数，其自变量 a 是向量。Fun 可以用字符串、内联函数或者函数 M 文件的函数句柄表示。

(3) 输入变量 a0 为向量，表示所求变量 a 的初始猜测值。

(4) 输入变量 lb 表示所求变量 a 的下限，输入变量 ub 表示所求变量 a 的上限。如果没有上下限限制，可以用空矩阵[]表示，如果没有下限，可以用-inf 表示。

(5) 从第六个输入变量开始的参数 p1,p2 等是向 Fun 函数传递的参数，其名字应该和定义 Fun 函数的输入变量名相一致。

(6) 输出变量 a 表示所求变量的值。

(7) 如果将 $Fun(a)$ 构造为残差函数，且 a 为被估计参数向量，则 lsqnonlin 就变为求解非线性最小二乘估计的函数。

例 5.3.3 混凝土的抗压强度 y 随养护时间 x 的延长而增加，其数学模型为

$$y = a_1 + a_2 e^{a_4 x} + a_3 e^{-a_4 x}$$

现将一批混凝土作成 12 个测试块，记录了养护时间 x（日）及抗压强度 y（kg/cm2）的数据，如表 5.3.1 所示。试估计该数学模型的参数 a1，a2，a3，a4。

表 5.3.1 混凝土的抗压强度 y 和养护时间 x 的实测数据

x	2	3	4	5	7	9	12	14	17	21	28	56
y	35	42	47	53	59	65	68	73	76	82	86	99

解：（1）编写残差计算函数文件 Residuals.m

```
function E=Residuals(a,x,y)
x=x(:);  y=y(:);                          %将原始数据变为列向量
Y=a(1)+a(2)*exp(a(4)*x)+a(3)*exp(-a(4)*x);  %估计参数下的计算值
E=Y-y;                                    %计算值减去原始值等于残差
```

（2）估计模型参数，并绘制模型曲线，如图 5.3.2 所示

```
>> x=[2 3 4 5 7 9 12 14 17 21 28 56];
>> y=[35 42 47 53 59 65 68 73 76 82 86 99];
>> a0=[50,0,-50,0];
>> options=optimset('lsqnonlin');
>> options.TolX=0.01; options.Display='off';
>> a=lsqnonlin(@Residuals,a0,[],[],options,x,y)
a =
    92.3627    0.0500   -65.2850    0.0878
>> xx=min(x):max(x);
>> yy=a(1)+a(2)*exp(a(4)*xx)+a(3)*exp(-a(4)*xx);   %估计参数下的计算值
>> plot(x,y,'o',xx,yy,'r')
```

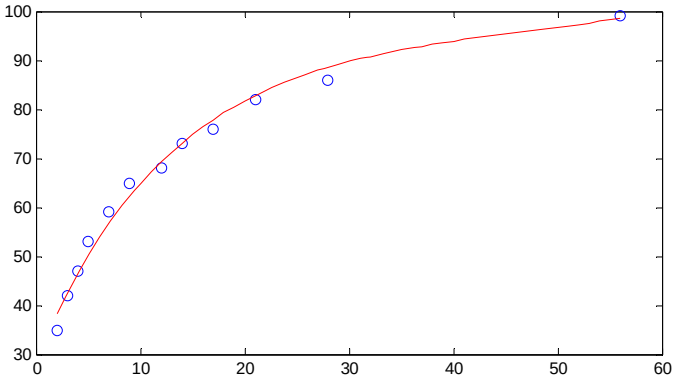

图 5.3.2 非线性最小二乘曲线拟合

5.4 插值和样条

曲线拟合是用带有噪声的“测量数据”构造出以某种方式最接近“真实数据”的曲线方程，因为测量数据包含噪声，所以不要求拟合曲线穿越所有测量数据。插值和拟合不同，插值认为原始数据是完全准确的，目的是用某种算法平滑的计算出这些原始数据之间的数据值。

5.4.1 插值

MATLAB 提供了很多插值函数，这些函数的使用方法大同小异，本节只介绍 interp1 一维插值函数。

【调用格式】

yi = interp1(x, Y, xi, method)　　计算插值点自变量为 xi 时的值 yi

pp = interp1(x, Y, method, 'pp')　　用原始数据获取插值函数数据

yi = ppval(pp, xi)　　计算插值函数数据 pp 函数关系下的函数值

【说明】

1. 输入变量 x，Y 是原始数据向量对。x 的数据必须以单调方式排列。
2. 输入变量 xi 是插值点的自变量坐标向量。
3. 输入变量 method 是插值方法，MATLAB 可以选择的插值方法包括：

   linear　　线性插值，缺省值。

   cublic　　三次多项式插值

   spline　　三次样条插值

   nearst　　最临近插值
4. 输出变量 yi 为插值点自变量为 xi 时的计算值。
5. 输出变量 pp 为插值函数数据，里面保存着计算插值的表达式参数，用于描述相邻原始数据之间的函数关系。可以通过 ppval 函数计算 pp 函数关系下的自变量 xi 的插值结果 yi。

例 5.3.4 已知 1900 年到 1990 年间，每隔 10 年美国的人口数量的统计数据（单位：百万）依次为 75.995，91.972， 105.711，123.203，131.669，150.697，179.323，203.212，226.505，249.633，求在 1975 年美国人口的数量，并绘制 1900 到 1990 年间每年的人口数量趋势图。

解：运行下列程序，观察运行结果。

```
>> t = 1900:10:1990;
>> p = [75.995   91.972   105.711   123.203   131.669...
       150.697   179.323   203.212   226.505   249.633];
>> interp1(t,p,1975)
ans =
   214.8585
>> interp1(t,p,2000)              %无法计算自变量超出范围的插值，返回值为 NaN
>> pp=interp1(t,p,'spline','pp');  %求取人口的插值函数数据
>> tt=1900:1990;
>> yy=ppval(pp,tt);               %根据插值函数数据计算插值结果
>> plot(t,p,'o',tt,yy,'b')        %绘制插值函数的曲线，如图 5.4.1 所示
```

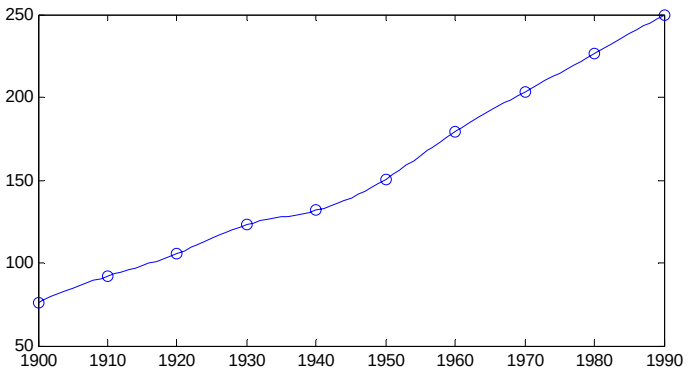

图 5.4.1 人口趋势曲线

5.4.2 样条

样条插值是常用的一种插值方法，其特点是精度高、最平滑，但是运算速度慢。经过样条插值后的曲线，除了在原始数据的端点外的其他数据点上都存在一阶和二阶导数，因此样条插值是非常平滑的。

【调用格式】

yy = spline(x,y,xx) 根据原始数据（x，y）计算 xx 的样条插值 yy

pp = spline(x,y) 根据原始数据（x，y）计算分段样条函数数据 pp

dpp = fnder(pp) 求 PP 形式的样条函数 $f(x)$ 的导数 $f'(x)$

ipp = fnint(pp) 求 PP 形式的样条函数 $f(x)$ 的不定积分 $s(x)=\int f(x)dx$

例 5.4.2 设函数 $f(x)=\sin x\cos x$，用样条函数求 $\int_1^2 f(x)dx$ 和 $f'(2)$。

解：运行下列程序，观察运行结果

```
>> dx=0.05;                          %设定采样时间间隔
>> x=(0:dx:1)*pi; y=sin(x).*cos(x);  %求采样点数据
>> pp=spline(x,y);                   %求 f(x)的样条函数
>> ipp=fnint(pp);                    %求 pp 样条函数的不定积分，近似为 f(x)的不定积分
>> dpp=fnder(pp);                    %求 pp 样条函数的微分，近似为 f(x)的微分
>> ppval(ipp,[1,2])*[-1;1]           %ppval(ipp,2)-ppval(ipp,1)，求定积分
ans =
    0.0594
>> ppval(dpp,2)                      %求微分
ans =
    0.9596
```

5.5 常微分方程的数值解

只含有一个自变量的微分方程称为常微分方程（ODE）。工程上的许多常微分方程或者没有解析解，或者求解析解困难太大，这时可以选择其数值解法。常微分方程分为初值问题和边值问题，本节只介绍初值问题的数值解法。

5.5.1 ODE 文件的编写

MATLAB 中求解常微分方程的数值解是通过将其变形为一阶向量微分方程来实现的。用 MATLAB 的 ODE 解算指令解常微分方程，要编写表示一阶向量微分方程的函数 M 文件，其基本格式为：

```
function    DY = Fun(t, Y)
```

其中输入变量 t 为时间变量，输入变量 Y 为列向量，输出变量 DY 是 Y 的一阶导数。

5.5.2 solver 解算指令

1. solver 解算指令说明

常微分方程化成一阶向量微分方程时，某些向量微分方程的向量解的各个分量的量级差别较大，这对数值求解算法来说是很大的困难，这种问题称之为刚性（stiff）问题。MATLAB 提供了很多常微分方程的解算函数，这些函数有些适用于刚性方程，有些适用于非刚性方程，并且其使用的数值算法和解算精度也

各有不同，这些函数通称为 solver 解算指令。表 5.5.1 中列出了各个解算指令的名称、精度和适用范围。

表 5.5.1 solver 解算指令

solve 指令	解题类型	精度	适用场合
ode45	非刚性	一步法，4、5 阶龙格库塔法，中等精度	大多数场合的首选算法
ode23	非刚性	一步法，2、3 阶龙格库塔法，精度低	较低精度场合，缺省e^{-3}
ode113	非刚性	多步法，Adams 算法，高低精度均可	ode45 计算时间长时的替代
ode23t	适度刚性	梯形法则算法，精度低	适度刚性
ode15s	刚性	多步法，中低精度	ode45 失败时候适用
ode23s	刚性	一步法，2 阶 Rosenbrock 算式，精度低	低精度时，比 ode15s 有效
ode23tb	刚性	梯形法则－反向数值微分两阶段法，精度低	低精度时，比 ode15s 有效

2.solver 解算指令的调用格式

【调用格式】

[t, Y] = solver('ODE_FUN ', tspan, Y0, options)

【说明】

(1) 输入变量 ODE_FUN 是 ODE 函数的文件名。

(2) 输入变量 tspan 为求数值解的时间范围。当 tspan=[t0, tf]时，表示求解 t0 到 tf 区间的数值解；当 tspan=[t0, t1 ,..., tf]时，表示求解 tspan 指定时间序列上的数值解。

(3) 输入变量 Y0 是微分方程组的初始条件列向量。

(4) 输入变量 options 是可选择的算法参数，详见 MATLAB 的帮助文件。

(5) 输出变量 t 是数值解的自变量数据列向量。

(6) 输出变量 Y 是数值解。Y 是一个矩阵，每一列代表了向量解的一个分量在自变量 t 上的数值解。

(7) solver 解算指令指的是表 5.5.1 中的任意函数。solver 不仅可以求解常微分方程，还可以求解常微分方程组。

例 5.5.1 求刚性常微分方程 $y''-1000(1-y^2)y'+y=0$ 的解 y。其初始条件为 $y(0)=2, y'(0)=0$。

解：（1）将常微分方程变为一阶微分方程组

将微分方程的高阶导数写成低阶导数的组合，记 $y''=1000(1-y^2)y'-y$，令 $y_1=y$，$y_2=y'$，可以将原来的方程和初始条件转换为下面的一阶微分方程组和初始条件

$$\begin{cases} y_1'=y_2 \\ y_2'=1000(1-y_1^2)y_2-y_1 \end{cases} \qquad \begin{cases} y_1(0)=2 \\ y_2(0)=0 \end{cases}$$

（2）编写 ODE 文件

```
%OdeFun1.m
function DY=OdeFun1(t, Y)
DY = zeros(2,1);          % 预定义列向量
DY(1) = Y(2);
DY(2) = 1000*(1-Y(1)^2)*Y(2)-Y(1);
```

（3）求解常微分方程的数值解，绘制 $y(t)$ 的曲线，如图 5.5.1 所示

```
>> [t, Y] = ode15s('OdeFun1',[0 3000],[2; 0]);        %刚性方程，不能用 ode45
>> plot(t,Y(:,1),'-o')
```

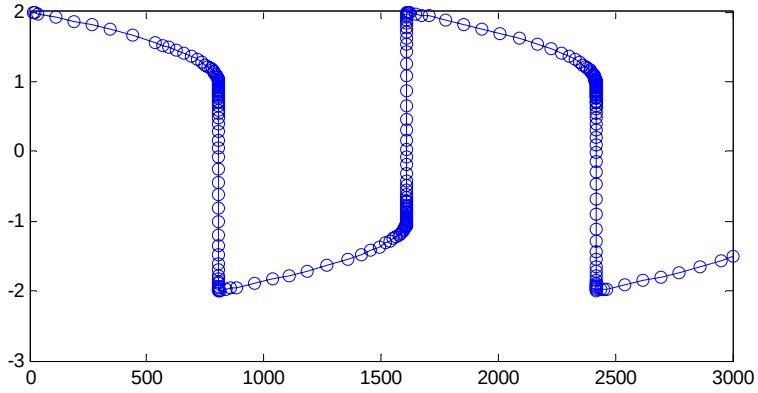

图 5.5.1 微分方程组的数值解曲线

例 5.5.2 求常微分方程组 $\begin{cases} y_1' = y_2 y_3 \\ y_2' = -y_1 y_3 \\ y_3' = -0.51 y_1 y_2 \end{cases}$ 在 [0,12] 区间的数值解。其初始条件为 $y_1(0)=0, y_2(0)=1, y_3(0)=1$。

解：按如下步骤运行程序，方程组 3 个变量的数值解的曲线如图 5.5.2 所示。

（1）编写 ODE 文件

```
%OdeFun2.m
function dy =OdeFun2(t,y)
dy = zeros(3,1);                    % 预定义列向量
dy(1) = y(2) * y(3);
dy(2) = -y(1) * y(3);
dy(3) = -0.51 * y(1) * y(2);
```

（2）求解常微分方程的数值解

```
>> [t, Y] = ode45('OdeFun2', [0 12], [0; 1; 1]);
>> plot(t, Y(:,1),'-', t, Y(:,2),'-.', t, Y(:,3),'.' );
```

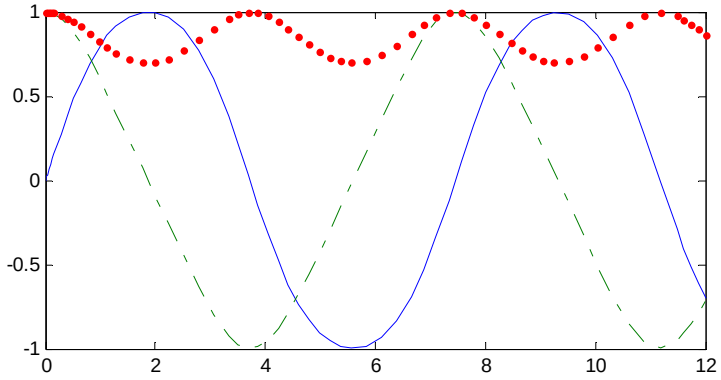

图 5.5.2 微分方程组的数值解曲线

小结

本章介绍常见的数值计算问题，包括线性代数、函数分析、数值微积分、常微分方程等，重点介绍了如何利用 MATLAB 提供的函数来实现数值计算，对于数学理论问题不做详细阐述，使用者很快就能应用 MATLAB 提供的强大函数进行复杂的方程组、微分、积分等运算。MATLAB 数值计算的结果为数值型数据，而不是数学上的解析表达式。