第 6 章　MATLAB 解方程与最优化问题求解

在科学研究和工程应用的许多领域，很多问题都常常归结为解方程，包括线性方程、非线性方程和常微分方程。研究方程的解析解固然能使人们更好地掌握问题的规律。但是，在很多情况下无法求出其解析解，这时数值解法就是一个十分重要的手段。MATLAB 为解决这类问题提供了极大的方便。

另外，在日常生活和实际工作中，人们对于同一个问题往往会提出多个解决方案，并通过各方面的论证从中提取最佳方案。最优化方法就是专门研究如何从多个方案中科学、合理地提取出最佳方案的科学。

本章介绍有关解方程和最优化问题的 MATLAB 实现方法。

- MATLAB 线性方程组求解
- MATLAB 非线性方程数值求解
- MATLAB常微分方程初值问题的数值解法
- MATLAB 最优化问题求解

6.1　线性方程组求解

将包含 n 个未知数，由 n 个方程构成的线性方程组表示为

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases}$$

其矩阵表示形式为

$$Ax=b$$

其中

$$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, x=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad b=\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}。$$

在 MATLAB 中，关于线性方程组的解法一般可以分为两类：一类是直接解法，就是在没有舍入误差的情况下，通过有限步的矩阵初等运算来求得方程组的解；另一类是迭代解法，就是先给定一个解的初始值，然后按照一定的迭代算法进行逐步逼近，求出更精确的近似解。

6.1.1　直接解法

1. 利用左除运算符的直接解法

线性方程组的直接解法大多基于高斯消元法、主元素消元法、平方根法和追赶法等。在 MATLAB 中，这些算法已经被编成了现成的库函数或运算符，因此，只需调用相应的函数或运算符即可完成线性方程组的求解。其中，最简单的方法就是使用左除运算符“\”。程序会自动根据输入的系数矩阵判断选用哪种方法进行求解。

对于线性方程组 Ax=b，可以利用左除运算符“\”求解：

x=A\b

当系数矩阵 A 为 N×N 的方阵时，MATLAB 会自行用高斯消元法求解线性方程组。若右端项 b 为 N×1 的列向量，则 x=A\b 可获得方程组的数值解 x（N×1 的列向量）；若右端项 b 为 N×M 的矩阵，则 x=A\b 可同时获得与系数矩阵 A 相同的 M 个线性方程组的数值解 x（为 N×M 的矩阵），即 x(:,j)=A\b(:,j)，j=1,2,…,M。注意，如果矩阵 A 是奇异的或接近奇异的，则 MATLAB 会给出警告信息。

当系数矩阵 A 不是方阵时，称这样的方程组为欠定方程组或超定方程组，MATLAB 将会在最小二乘意义下求解。

例 6-1　用直接解法求解下列线性方程组：

$$\begin{cases} 2x_1 + x_2 - 5x_3 + x_4 = 13 \\ x_1 - 5x_2 + 7x_4 = -9 \\ 2x_2 + x_3 - x_4 = 6 \\ x_1 + 6x_2 - x_3 - 4x_4 = 0 \end{cases}$$

解：程序如下：

```
A=[2,1,-5,1;1,-5,0,7;0,2,1,-1;1,6,-1,-4];
b=[13,-9,6,0]';
x=A\b
x =
  -66.5556
   25.6667
  -18.7778
   26.5556
```

在 2.4.3 节中曾介绍过利用矩阵求逆来解线性方程组，即 $x=A^{-1}b$，其结果与使用左除运算相同。

2. 利用矩阵的分解求解线性方程组

矩阵分解是指根据一定的原理用某种算法将一个矩阵分解成若干个矩阵的乘积。常见的矩阵分解有 LU 分解、QR 分解、Cholesky 分解、Schur 分解、Hessenberg 分解及奇异分解等。这里着重介绍前 3 种常见的分解。通过这些分解方法求解线性方程组的优点是运算速度快，可以节省存储空间。

（1）LU 分解。矩阵的 LU 分解就是将一个矩阵表示为一个交换下三角矩阵和一个上三角矩阵的乘积形式。线性代数中已经证明，只要方阵 A 是非奇异的，LU 分解总是可以进行的。

MATLAB 提供的 lu 函数用于对矩阵进行 LU 分解，其调用格式为：

① [L,U]=lu(X)：产生一个上三角矩阵 U 和一个变换形式的下三角矩阵 L（行交换），使之满足 X=LU。注意，这里的矩阵 X 必须是方阵。

②[L,U,P]=lu(X)：产生一个上三角矩阵 U 和一个下三角矩阵 L 以及一个置换矩阵 P，使之满足 PX=LU。当然矩阵 X 同样必须是方阵。

当使用第①种格式时，矩阵 L 往往不是一个下三角矩阵，但可以通过行交换使其成为一个下三角矩阵。设

$$A=\begin{bmatrix}1 & -1 & 1\\ 5 & -4 & 3\\ 2 & 1 & 1\end{bmatrix}$$

则对矩阵 A 进行 LU 分解的命令如下：

```
A=[1,-1,1;5,-4,3;2,1,1]
A =
     1    -1     1
     5    -4     3
     2     1     1
[L,U]=lu(A)
L =
    0.2000   -0.0769    1.0000
    1.0000         0         0
    0.4000    1.0000         0
U =
    5.0000   -4.0000    3.0000
         0    2.6000   -0.2000
         0         0    0.3846
```

为检验结果是否正确，输入命令：

```
LU=L*U
LU =
     1    -1     1
     5    -4     3
     2     1     1
```

说明结果是正确的。例中所获得的矩阵 L 并不是一个下三角矩阵，但经过各行互换后，即可获得一个下三角矩阵。

利用第②种格式对矩阵 A 进行 LU 分解的命令如下：

```
[L,U,P]=lu(A)
L =
    1.0000         0         0
    0.4000    1.0000         0
    0.2000   -0.0769    1.0000
U =
    5.0000   -4.0000    3.0000
```

```
         0    2.6000   -0.2000
         0         0    0.3846
P =
     0     1     0
     0     0     1
     1     0     0
LU=L*U           %这种分解其乘积不为 A
LU =
     5    -4     3
     2     1     1
     1    -1     1
inv(P)*L*U       %考虑矩阵 P 后其乘积等于 A
ans =
     1    -1     1
     5    -4     3
     2     1     1
```

实现 LU 分解后，线性方程组 Ax=b 的解 x=U\(L\b)或 x=U\(L\Pb)，这样可以大大提高运算速度。

例 6-2　用 LU 分解求解例 6-1 中的线性方程组。

命令如下：

```
A=[2,1,-5,1;1,-5,0,7;0,2,1,-1;1,6,-1,-4];
b=[13,-9,6,0]';
[L,U]=lu(A);
x=U\(L\b)
x =
  -66.5556
   25.6667
  -18.7778
   26.5556
```

或采用 LU 分解的第②种格式，命令如下：

```
[L,U,P]=lu(A);
x=U\(L\P*b)
```

将得到与上面同样的结果。

（2）QR 分解。对矩阵 X 进行 QR 分解，就是把 X 分解为一个正交矩阵 Q 和一个上三角矩阵 R 的乘积形式。QR 分解只能对方阵进行。MATLAB 的函数 qr 可用于对矩阵进行 QR 分解，其调用格式为：

①[Q,R]=qr(X)：产生一个正交矩阵 Q 和一个上三角矩阵 R，使之满足 X=QR。

②[Q,R,E]=qr(X)：产生一个正交矩阵 Q、一个上三角矩阵 R 及一个置换矩阵 E，使之满足 X E=Q R。

设

$$\mathrm{A}=\begin{bmatrix}1 & -1 & 1\\ 5 & -4 & 3\\ 2 & 7 & 10\end{bmatrix}$$

则对矩阵 A 进行 QR 分解的命令如下：

```
A=[1,-1,1;5,-4,3;2,7,10];
[Q,R]=qr(A)
Q =
   -0.1826   -0.0956   -0.9785
   -0.9129   -0.3532    0.2048
   -0.3651    0.9307   -0.0228
R =
   -5.4772    1.2780   -6.5727
         0    8.0229    8.1517
         0         0   -0.5917
```

为检验结果是否正确，输入命令：

```
QR=Q*R
QR =
    1.0000   -1.0000    1.0000
    5.0000   -4.0000    3.0000
    2.0000    7.0000   10.0000
```

说明结果是正确的。利用第②种格式对矩阵 A 进行 QR 分解：

```
[Q,R,E]=qr(A)
Q =
   -0.0953   -0.2514   -0.9632
   -0.2860   -0.9199    0.2684
   -0.9535    0.3011    0.0158
R =
  -10.4881   -5.4347   -3.4325
         0    6.0385   -4.2485
         0         0    0.4105
E =
     0     0     1
     0     1     0
     1     0     0
Q*R/E          %验证 A=Q*R*inv(E)
ans =
    1.0000   -1.0000    1.0000
    5.0000   -4.0000    3.0000
    2.0000    7.0000   10.0000
```

实现 QR 分解后，线性方程组 Ax=b 的解 x=R\(Q\b)或 x=E(R\(Q\b))。

例 6-3 用 QR 分解求解例 6-1 中的线性方程组。

命令如下：

```
A=[2,1,-5,1;1,-5,0,7;0,2,1,-1;1,6,-1,-4];
b=[13,-9,6,0]';
[Q,R]=qr(A);
x=R\(Q\b)
x =
  -66.5556
   25.6667
```

```
   -18.7778
    26.5556
```

或采用 QR 分解的第②种格式，命令如下：

```
[Q,R,E]=qr(A);
x=E*(R\(Q\b))
```

将得到与上面同样的结果。

（3）Cholesky 分解。如果矩阵 X 是对称正定的，则 Cholesky 分解将矩阵 X 分解成一个下三角矩阵和一个上三角矩阵的乘积。设上三角矩阵为 R，则下三角矩阵为其转置，即 X=R'R。MATLAB 函数 chol(X)用于对矩阵 X 进行 Cholesky 分解，其调用格式为：

① R=chol(X)：产生一个上三角矩阵 R，使 R′R=X。若 X 为非对称正定，则输出一个出错信息。

② [R,p]=chol(X)：这个命令格式将不输出出错信息。若 X 为对称正定的，则 p=0，R 与上述格式得到的结果相同；否则 p 为一个正整数。如果 X 为满秩矩阵，则 R 为一个阶数为 q=p−1 的上三角矩阵，且满足 R′R=X(1:q,1:q)。

设

$$A=\begin{bmatrix}2 & 1 & 1\\ 1 & 2 & -1\\ 1 & -1 & 3\end{bmatrix}$$

则对矩阵 A 进行 Cholesky 分解的命令如下：

```
A=[2,1,1;1,2,-1;1,-1,3];
R=chol(A)
R =
   1.4142    0.7071    0.7071
        0    1.2247   -1.2247
        0         0    1.0000
```

可以验证 R'R=A：

```
R'*R
ans =
   2.0000    1.0000    1.0000
   1.0000    2.0000   -1.0000
   1.0000   -1.0000    3.0000
```

利用第②种格式对矩阵 A 进行 Cholesky 分解：

```
[R,p]=chol(A)
R =
   1.4142    0.7071    0.7071
        0    1.2247   -1.2247
        0         0    1.0000
p =
     0
```

结果中 p=0，这表示矩阵 A 是一个正定矩阵。如果试图对一个非正定矩阵进行 Cholesky 分解，则将得出错误信息，所以，chol 函数还可以用来判定矩阵是否为正定矩阵。

实现 Cholesky 分解后，线性方程组 Ax=b 变成 R'Rx=b，所以 x=R\(R'\b)。

例 6-4 用 Cholesky 分解求解例 6-1 中的线性方程组。

解：命令如下：

```
A=[2,1,-5,1;1,-5,0,7;0,2,1,-1;1,6,-1,-4];
b=[13,-9,6,0]';
R=chol(A)
??? Error using ==> chol
Matrix must be positive definite.
```

命令执行时，出现错误信息，说明 A 为非正定矩阵。

6.1.2 迭代解法

迭代解法非常适合求解大型系数矩阵的方程组。在数值分析中，迭代解法主要包括 Jacobi 迭代法、Gauss-Serdel 迭代法、超松弛迭代法和两步迭代法。首先用一个例子说明迭代法的思想。

为了求解线性方程组：

$$\begin{cases}10x_1 - x_2 = 9\\ -x_1 + 10x_2 - 2x_3 = 7\\ -2x_2 + 10x_3 = 6\end{cases}$$

将方程改写为

$$\begin{cases}x_1 = 10x_2 - 2x_3 - 7\\ x_2 = 10x_1 - 9\\ x_3 = \dfrac{1}{10}(6 + 2x_2)\end{cases}$$

这种形式的好处是将一组 x 代入右端，可以立即得到另一组 x。如果两组 x 相等，那么它就是方程组的解，不等时可以继续迭代。例如，选取初值 $x_1=x_2=x_3=0$，则经过一次迭代后，得到 $x_1=-7$，$x_2=-9$，$x_3=0.6$，然后再继续迭代。可以构造方程的迭代公式为

$$\begin{cases}x_1^{(k+1)} = 10x_2^{(k)} - 2x_3^{(k)} - 7\\ x_2^{(k+1)} = 10x_1^{(k)} - 9\\ x_3^{(k+1)} = 0.6 + 0.2x_2^{(k)}\end{cases}$$

1. Jacobi 迭代法

对于线性方程组 Ax=b，如果 A 为非奇异方阵，即 $a_{ii}\neq 0$（i=1,2,…,n），则可将 A 分解为 A=D−L−U，其中 D 为对角矩阵，其元素为 A 的对角元素，L 与 U 为 A 的下三角矩阵和上三角矩阵：

$$L = -\begin{bmatrix}0 & & & \\ a_{21} & 0 & & \\ \cdots & \ddots & \ddots & \\ a_{n1} & \cdots & a_{n,n-1} & 0\end{bmatrix},\quad U = -\begin{bmatrix}0 & a_{12} & \cdots & a_{1n}\\ & 0 & \ddots & \vdots\\ & & \ddots & a_{n-1,n}\\ & & & 0\end{bmatrix}$$

于是 Ax=b 化为

$$x=D^{-1}(L+U)x+D^{-1}b$$

与之对应的迭代公式为

$$x^{(k+1)}=D^{-1}(L+U)x^{(k)}+D^{-1}b$$

这就是 Jacobi 迭代公式。如果序列$\{x^{(k+1)}\}$收敛于 x，则 x 必是方程 Ax=b 的解。

Jacobi 迭代法的 MATLAB 函数文件 Jacobi.m 如下：

```
function [y,n]=jacobi(A,b,x0,ep)
if nargin==3
   ep=1.0e-6;
elseif nargin<3
   error
   return
end
D=diag(diag(A));     %求 A 的对角矩阵
L=-tril(A,-1);       %求 A 的下三角矩阵
U=-triu(A,1);        %求 A 的上三角矩阵
B=D\(L+U);
f=D\b;
y=B*x0+f;
n=1;                 %迭代次数
while norm(y-x0)>=ep
   x0=y;
   y=B*x0+f;
   n=n+1;
end
```

例 6-5　用 Jacobi 迭代法求解下列线性方程组。设迭代初值为 0，迭代精度为 10^{-6}。

$$\begin{cases}10x_1-x_2=9\\-x_1+10x_2-2x_3=7\\-2x_2+10x_3=6\end{cases}$$

解：在命令中调用函数文件 Jacobi.m，命令如下：

```
A=[10,-1,0;-1,10,-2;0,-2,10];
b=[9,7,6]';
[x,n]=jacobi(A,b,[0,0,0]',1.0e-6)
x =
   0.9958
   0.9579
   0.7916
n =
   11
```

2. Gauss-Serdel 迭代法

在 Jacobi 迭代过程中，计算$x_i^{(k+1)}$时，$x_1^{(k+1)},\cdots,x_{i-1}^{(k+1)}$已经得到，不必再用$x_1^{(k)},\cdots,x_{i-1}^{(k)}$，即原来的迭代公式 $Dx^{(k+1)}=(L+U)x^{(k)}+b$ 可以改进为 $Dx^{(k+1)}=Lx^{(k+1)}+Ux^{(k)}+b$，于是得到

$$x^{(k+1)}=(D-L)^{-1}Ux^{(k)}+(D-L)^{-1}b$$

该式即为 Gauss-Serdel 迭代公式。和 Jacobi 迭代相比，Gauss-Serdel 迭代用新分量代替旧分量，精度会高些。

Gauss-Serdel 迭代法的 MATLAB 函数文件 gauseidel.m 如下：

```
function [y,n]=gauseidel(A,b,x0,ep)
if nargin==3
   ep=1.0e-6;
elseif nargin<3
   error
   return
end
D=diag(diag(A));     %求 A 的对角矩阵
L=-tril(A,-1);       %求 A 的下三角矩阵
U=-triu(A,1);        %求 A 的上三角矩阵
G=(D-L)\U;
f=(D-L)\b;
y=G*x0+f;
n=1;                 %迭代次数
while norm(y-x0)>=ep
   x0=y;
   y=G*x0+f;
   n=n+1;
end
```

例 6-6 用 Gauss-Serdel 迭代法求解下列线性方程组。设迭代初值为 0，迭代精度为 10^{-6}。

$$\begin{cases} 10x_1 - x_2 = 9 \\ -x_1 + 10x_2 - 2x_3 = 7 \\ -2x_2 + 10x_3 = 6 \end{cases}$$

解：在命令中调用函数文件 gauseidel.m，命令如下：

```
A=[10,-1,0;-1,10,-2;0,-2,10];
b=[9,7,6]';
[x,n]=gauseidel(A,b,[0,0,0]',1.0e-6)
x =
   0.9958
   0.9579
   0.7916
n =
    7
```

由此可见，一般情况下 Gauss-Serdel 迭代比 Jacobi 迭代要收敛得快一些。但这也不是绝对的，在某些情况下，Jacobi 迭代收敛而 Gauss-Serdel 迭代却可能不收敛，看下面的例子。

例 6-7 分别用 Jacobi 迭代和 Gauss-Serdel 迭代法求解下列线性方程组，看是否收敛。

$$\begin{bmatrix} 1 & 2 & -2 \\ 1 & 1 & 1 \\ 2 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 9 \\ 7 \\ 6 \end{bmatrix}$$

解：命令如下：

```
a=[1,2,-2;1,1,1;2,2,1];
b=[9;7;6];
[x,n]=jacobi(a,b,[0;0;0])
x =
```

```
   -27
    26
     8
n =
     4
[x,n]=gauseidel(a,b,[0;0;0])
x =
   NaN
   NaN
   NaN
n =
    1012
```

可见对此方程，用 Jacobi 迭代收敛，而 Gauss-Serdel 迭代不收敛。因此，在使用迭代法时要考虑算法的收敛性。

6.2　非线性方程数值求解

5.5.4 节曾介绍过多项式方程的求根，这里介绍更一般的非线性方程的求根方法。非线性方程的求根方法有很多，常用的有牛顿迭代法，但该方法需要求原方程的导数，而在实际运算中这一条件有时是不能满足的，所以又出现了弦截法、二分法等其他方法。MATLAB 提供了有关的函数用于非线性方程求解。

6.2.1　单变量非线性方程求解

在 MATLAB 中提供了一个 fzero 函数，可以用来求单变量非线性方程的根。该函数的调用格式为：

z=fzero(@fname,x0,tol,trace)

其中，fname 是待求根的函数文件名，x0 为搜索的起点。一个函数可能有多个根，但 fzero 函数只给出离 x0 最近的那个根。tol 控制结果的相对精度，默认时 tol=eps，trace 指定迭代信息是否在运算中显示，为 1 时显示，为 0 时不显示，默认时 trace=0。

例 6-8　求 $f(x)=x-10^x+2=0$ 在 $x_0=0.5$ 附近的根。

解：（1）建立函数文件 funx.m，命令如下：

```
function fx=funx(x)
fx=x-10.^x+2;
```

（2）调用 fzero 函数求根，命令如下：

```
z=fzero(@funx,0.5)
z =
   0.3758
```

6.2.2　非线性方程组的求解

非线性方程组是经常遇到的一类数学问题，在此作一简单介绍。这部分内容已超出 MATLAB 的基本部分，它需要用到 MATLAB 的优化工具箱（Optimization Toolbox）。

对于非线性方程组 F(X)=0，用 fsolve 函数求其数值解，该函数的调用格式为：

X=fsolve(@fname,X0,options)

其中，X 为返回的解，fname 是用于定义需求解的非线性方程组的函数文件名，X0 是求解过程的初值，options 用于设置优化工具箱的优化参数。

优化工具箱提供了许多优化参数选项，用户在命令窗口输入下列命令，可以将优化参数全部显示出来：

```
optimset
```

如果希望得到某个优化函数（如 fsolve 函数）当前的默认参数值，则可在命令窗口输入命令：

```
optimset fsolve
```

如果想改变其中某个参数选项，则可以调用 optimset 函数来完成。例如，Display 参数选项决定函数调用时中间结果的显示方式，其中'off'为不显示，'iter'表示每步都显示，'final'只显示最终结果。如果要将 Display 选项设定为 off，可以使用命令：

```
options=optimset('Display','off')。
```

除了 fsolve 函数外，在使用后面 6.4 节中介绍的其他优化函数时，也可以用 optimset 函数来设置优化参数。

例 6-9 求下列非线性方程组在(0.5,0.5)附近的数值解。

$$\begin{cases} x - 0.6\sin x - 0.3\cos y = 0 \\ y - 0.6\cos x + 0.3\sin y = 0 \end{cases}$$

解：（1）建立函数文件 myfun.m，程序如下：

```
function q=myfun(p)
x=p(1);
y=p(2);
q(1)=x-0.6*sin(x)-0.3*cos(y);
q(2)=y-0.6*cos(x)+0.3*sin(y);
```

（2）在给定的初值 x_0=0.5、y_0=0.5 下，调用 fsolve 函数求方程的根，程序如下：

```
options=optimset('Display','off');
x=fsolve(@myfun,[0.5,0.5]',options)
x =
    0.6354
    0.3734
```

将求得的解代回原方程，可以检验结果是否正确，命令如下：

```
q=myfun(x)
q =
     1.0e-009 *
    0.2375    0.2957
```

可见得到了较高精度的结果。

6.3 常微分方程初值问题的数值解法

众所周知，只有对一些典型的常微分方程，才能求出它们的一般解表达式并用初始条件确定表达式中的任意常数。然而在实际问题中遇到的常微分方程往往很复杂，在许多情况下

得不出一般解，所以，一般是要求获得解在若干个点上的近似值。

考虑常微分方程的初值问题

$$y'=f(t,y)，\ t_0\leqslant t\leqslant T$$
$$y(t_0)=y_0$$

所谓其数值解法，就是求它的解 y(t)在节点 $t_0<t_1<\cdots<t_m$ 处的近似值 $y_0,y_1,\cdots,y_m$ 的方法。所求得的 $y_0,y_1,\cdots,y_m$ 称为常微分方程初值问题的数值解。一般采用等距节点 $t_n=t_0+nh$，$n=0,1,\cdots,m$，其中 h 为相邻两个节点间的距离，叫做步长。

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（Euler）法、龙格—库塔（Runge-Kutta）法、线性多步法、预报校正法等。本节简单介绍龙格—库塔法及其 MATLAB 实现。

6.3.1　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

$y(t_0+h)=y_1\approx y_0+hf(t_0,y_0)$，$h>0$，称为步长

$y(t_0+2h)=y_2\approx y_1+hf(t_1,y_1)$

一般地，在任意点 $t_i=t_0+ih$，有

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

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：

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

当然，递推过程中有一个误差累计的问题。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—库塔公式是

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

其中

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

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

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

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

6.3.2　龙格—库塔法的实现

MATLAB 提供了多个求常微分方程数值解的函数，一般调用格式为

```
[t,y]=solver(@fname,tspan,y0)
```

其中，t 和 y 分别给出时间向量和相应的状态向量。solver 为求常微分方程数值解的函数，表 6-1 列出了各函数采用的方法和适用的场合。fname 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。

表 6-1　求常微分方程数值解的函数

求解函数	采用方法	适用场合
ode23	2-3 阶 Runge-Kutta 算法，低精度	非刚性
ode45	4-5 阶 Runge-Kutta 算法，中精度	非刚性
ode113	Adams 算法，精度可达到 10^{-3}～10^{-6}	非刚性，计算时间比 ode45 短
ode23t	梯形算法	适度刚性
ode15s	Gear's 反向数值微分算法，中精度	刚性
ode23s	2 阶 Rosebrock 算法，低精度	刚性，当精度较低时，计算时间比 ode15s 短
ode23tb	梯形算法，低精度	刚性，当精度较低时，计算时间比 ode15s 短
ode15i	可变秩方法	完全隐式微分方程

选取方法时，可综合考虑精度要求和复杂度控制要求等实际需要，选择适当的方法求解。

若微分方程描述的一个变化过程包含着多个相互作用但变化速度相差十分悬殊的子过程，这样一类过程就认为具有“刚性”，这类方程具有非常分散的特征值。求解刚性方程的初值问题的解析解是困难的，常采用表 6-1 中的函数 ode15s、ode23s 和 ode23tb 求其数值解。求解非刚性的一阶常微分方程或方程组的初值问题的数值解常采用函数 ode23 和 ode45，其中 ode23 采用 2 阶龙格—库塔算法，用 3 阶公式作误差估计来调节步长，具有低精度；ode45 则采用 4 阶龙格—库塔算法，用 5 阶公式作误差估计来调节步长，具有中精度。

例 6-10　设有初值问题：

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

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

解：（1）建立函数文件 funt.m，命令如下：

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

（2）求解微分方程，程序如下：

```
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
t'
ans =
  Columns 1 through 7
        0    0.3200    0.9380    1.8105    2.8105    3.8105    4.8105
  Columns 8 through 13
    5.8105    6.8105    7.8105    6.8105    9.8105   10.0000
y'
ans =
  Columns 1 through 7
    2.0000    2.1490    2.3929    2.6786    2.9558    3.1988    3.4181
```

```
  Columns 8 through 13
    3.6198    3.8079    3.9849    4.1529    4.3133    4.3430
y1'
ans =
  Columns 1 through 7
    2.0000    2.1489    2.3921    2.6765    2.9521    3.1933    3.4105
  Columns 8 through 13
    3.6097    3.7947    3.9683    4.1322    4.2879    4.3166
```

y 为数值解，y1 为精确值，显然两者近似。

例 6-11　求著名的 Van der Pol 方程 $\ddot{x}+(x^2-1)\dot{x}+x=0$。

解：函数 ode23 和 ode45 是对一阶常微分方程组设计的，因此对高阶常微分方程，需先将它转化为一阶常微分方程组，即状态方程。选择状态变量 $x_1=\dot{x}$，$x_2=x$，则可写出 Van der Pol 方程的状态方程形式：

$$\dot{x}_1=(1-x_2^2)x_1-x_2$$

$$\dot{x}_2=x_1$$

基于以上状态方程，求解过程如下：

（1）建立函数文件 vdpol.m，命令如下：

```
function xdot=vdpol(t,x)
xdot(1)=(1-x(2)^2)*x(1)-x(2);
xdot(2)=x(1);
xdot=xdot';
```

状态方程的表示形式并不是唯一的，如 Van der Pol 方程还可以表示成“xdot=[(1-x(2)^2)*x(1)-x(2);x(1)];”或“xdot=[(1-x(2)^2),-1;1,0]*x;”。

（2）求解微分方程，命令如下：

```
t0=0;
tf=20;
x0=[0;0.25];
[t,x]=ode45(@vdpol,[t0,tf],x0);
[t,x]
ans =
         0         0    0.2500
    0.0002   -0.0001    0.2500
    0.0004   -0.0001    0.2500
    0.0006   -0.0002    0.2500
    0.0008   -0.0002    0.2500
            ……
   19.6482   -0.6301    1.7403
   19.7362   -0.6703    1.6831
   19.8241   -0.7105    1.6224
   19.9121   -0.7520    1.5581
   20.0000   -0.7961    1.4900
```

结果第 1 列为 t 的采样点，第 2 列和第 3 列分别为 x'和 x 与 t 对应点的值（只列出部分结果）。

（3）得出方程的数值解之后，还可以绘制解的曲线。

```
subplot(1,2,1);plot(t,x);                    %系统时间响应曲线
subplot(1,2,2);plot(x(:,1),x(:,2))           %系统相平面曲线
```

绘制系统的响应曲线及相平面曲线如图 6-1 所示。

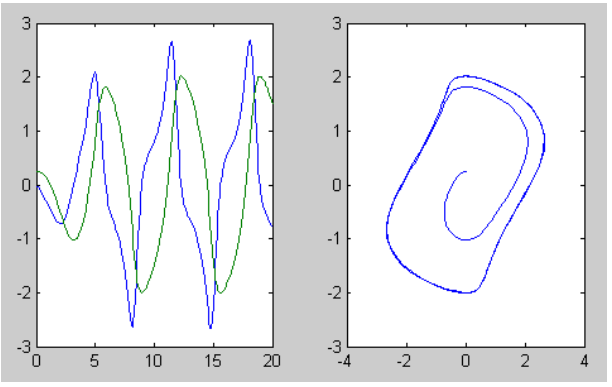

图 6-1　Van der Pol 方程的时间响应曲线及相平面曲线

例 6-12　Lorenz 模型的状态方程表示为

$$\dot{x}_1(t) = -\beta x_1(t) + x_2(t)x_3(t)$$
$$\dot{x}_2(t) = -\sigma x_2(t) + \sigma x_3(t)$$
$$\dot{x}_3(t) = -x_2(t)x_1(t) + \rho x_2(t) - x_3(t)$$

取 σ=10，ρ=28，β=8/3，试绘制系统相平面图。

解：（1）建立 Lorenz 模型的函数文件 lorenz.m，命令如下：

```
function xdot=lorenz(t,x)
xdot=[-8/3,0,x(2);0,-10,10;-x(2),28,-1]*x;
```

（2）解微分方程组，命令如下：

```
x0=[0,0,eps]';
[t,x]=ode23(@lorenz,[0,100],x0);
```

（3）绘制系统相平面图，如图 6-2 所示。

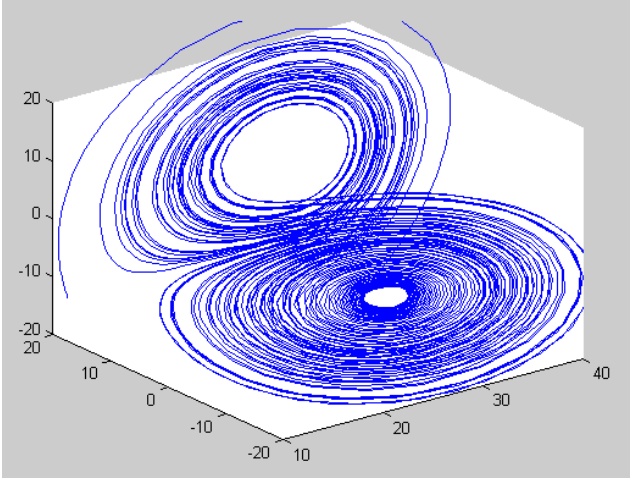

图 6-2　Lorenz 模型相平面图

```
plot3(x(:,1),x(:,2),x(:,3));
axis([10,40,-20,20,-20,20]);
```

6.4　最优化问题求解

6.4.1　无约束最优化问题求解

无约束最优化问题的一般描述为

$$\min_{x} f(x)$$

其中，$x=[x_1,x_2,\cdots,x_n]^T$，该数学表示的含义亦即求取一组 x，使得目标函数 f(x)为最小，故这样的问题又称为最小化问题。

在实际应用中，许多科学研究和工程计算问题都可以归结为一个最小化问题，如能量最小、时间最短等。MATLAB 提供了 3 个求最小值的函数，它们的调用格式为：

（1）[x,fval]=fminbnd(@fname,x1,x2,options)：求一元函数在(xl,x2)区间中的极小值点 x 和最小值 fval。

（2）[x,fval]=fminsearch(@fname,x0,options)：基于单纯形算法求多元函数的极小值点 x 和最小值 fval。

（3）[x,fval]=fminunc(@fname,x0,options)：基于拟牛顿法求多元函数的极小值点 x 和最小值 fval。

确切地讲，这里讨论的也只是局域极值的问题（全域最小问题要复杂得多）。fname 是定义目标函数的 M 文件名。fminbnd 的输入变量 xl、x2 分别表示被研究区间的左、右边界。fminsearch 和 fminunc 的输入变量 x0 是一个向量，表示极值点的初值。options 为优化参数，可以通过 optimset 函数来设置。当目标函数的阶数大于 2 时，使用 fminunc 比 fminsearch 更有效，但当目标函数高度不连续时，使用 fminsearch 效果较好。

MATLAB 没有专门提供求函数最大值的函数，但只要注意到-f(x)在区间（a,b）上的最小值就是 f(x)在（a,b）上的最大值，所以 fminbnd(-f,x1,x2)返回函数 f(x)在区间（x1,x2）上的最大值。

例 6-13　求 $f(x)=x^3-2x-5$ 在[0,5]内的最小值点。

解：（1）建立函数文件 mymin.m，命令如下：

```
function fx=mymin(x)
fx=x.^3-2*x-5;
```

（2）调用 fminbnd 函数求最小值点，命令如下：

```
x=fminbnd(@mymin,0,5)
x=
    0.8165
```

例 6-14　设

$$f(x,y,z)=x+\frac{y^2}{4x}+\frac{z^2}{y}+\frac{2}{z}$$

求函数 f 在（0.5,0.5,0.5）附近的最小值。

解：（1）建立函数文件 fxyz.m，命令如下：

```
function f=fxyz(p)
x=p(1);y=p(2);z=p(3);
f=x+y^2/x/4+z^2/y+2/z;
```

（2）求函数的最小值点和最小值，命令如下：

```
[U,fmin]=fminsearch(@fxyz,[0.5,0.5,0.5])
U =
    0.5000    1.0000    1.0000
fmin =
    4.0000
```

6.4.2 有约束最优化问题求解

有约束最优化问题的一般描述为

$$\min_{x\ \text{s.t.}\ G(x)\leqslant 0} f(x)$$

其中，$x=[x_1,x_2,\cdots,x_n]^T$，该数学表示的含义亦即求取一组 x，使得目标函数 f(x)为最小，且满足约束条件 G(x)≤0。记号 s.t.是英文 subject to 的缩写，表示 x 要满足后面的约束条件。

约束条件可以进一步细化为：

（1）线性不等式约束：Ax≤b。

（2）线性等式约束：$A_{eq}x=b_{eq}$。

（3）非线性不等式约束：Cx≤0。

（4）非线性等式约束：C_{eq} x=0。

（5）x 的下界和上界：$L_{bnd}\leqslant x\leqslant U_{bnd}$。

MATLAB 最优化工具箱提供了一个 fmincon 函数，专门用于求解各种约束下的最优化问题。该函数的调用格式为：

[x,fval]=fmincon(@fname,x0,A,b, Aeq,beq,Lbnd,Ubnd,NonF,options)

其中，x、fval、fname、x0 和 options 的含义与求最小值函数相同。其余参数为约束条件，参数 NonF 为非线性约束函数的 M 文件名。如果某个约束不存在，则用空矩阵来表示。

例 6-15 求解有约束最优化问题。

$$\min_{x\ \text{s.t.}\begin{cases}x_1+0.5x_2\geqslant 0.4\\ 0.5x_1+x_2\geqslant 0.5\\ x_1\geqslant 0,x_2\geqslant 0\end{cases}} f(x)=0.4x_2+x_1^2+x_2^2-x_1x_2+\frac{1}{30}x_1^3$$

解：（1）编写目标函数 M 文件 fop.m，命令如下：

```
function f=fop(x)
f=0.4*x(2)+x(1)^2+x(2)^2-x(1)*x(2)+1/30*x(1)^3;
```

（2）设定约束条件，并调用 fmincon 函数求解此约束最优化问题，程序如下：

```
x0=[0.5;0.5];
A=[-1,-0.5;-0.5,-1];
b=[-0.4;-0.5];
lb=[0;0];
options=optimset('Display','off');
[x,f]=fmincon(@fop,x0,A,b,[],[],lb,[],[],options)
```

```
x =
    0.3400
    0.3300
f =
    0.2456
```

6.4.3　线性规划问题求解

线性规划是研究线性约束条件下线性目标函数的极值问题的数学理论和方法。线性规划问题的标准形式为

$$\min_{x \text{ s.t. } \begin{cases} Ax=b \\ A_{eq}x=b_{eq} \\ L_{bnd} \leqslant x \leqslant U_{bnd} \end{cases}} f(x)$$

在 MATLAB 中求解线性规划问题使用函数 linprog，其调用格式为：

[x, fval] = linprog(f, A, b, Aeq, beq, lbnd, ubnd)

其中，x 是最优解，fval 是目标函数的最优值。函数中的各项参数是线性规划问题标准形式中的对应项，x、b、beq、lbnd、ubnd 是向量，A、Aeq 为矩阵，f 为目标函数系数向量。

例 6-16　求解线性规划问题。

$$\min_{x \text{ s.t.} \begin{cases} 3x_1+x_2 \geqslant 3 \\ 4x_1+3x_2 \geqslant 6 \\ x_1+2x_2 \geqslant 2 \\ x_1 \geqslant 0, x_2 \geqslant 0 \end{cases}} f(x) = 2x_1 + x_2$$

解：程序如下：

```
f=[2;1];
A=[-3,-1;-4,-3;-1,-2];
b=[-3;-6;-2];
lb=[0;0];
options=optimset('Display','off');
[x,f]=linprog(f,A,b,[],[],lb,[])
Optimization terminated.
x =
    0.6000
    1.2000
f =
    2.4000
```

实验指导

一、实验目的

1．掌握线性方程组的数值求解方法。
2．掌握常微分方程的数值求解方法。
3．掌握非线性方程以及最优化问题的求解方法。

二、实验内容

1．下面是一个线性病态方程组：

$$\begin{bmatrix} 1/2 & 1/3 & 1/4 \\ 1/3 & 1/4 & 1/5 \\ 1/4 & 1/5 & 1/6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0.95 \\ 0.67 \\ 0.52 \end{bmatrix}$$

（1）求方程的解。

（2）将方程右边向量元素 b_3 改为 0.53，再求解，并比较 b_3 的变化和解的相对变化。

（3）计算系数矩阵 A 的条件数并分析结论。

2．求下列方程的解。

（1）$x^{41}+x^3+1=0$，x_0=-1。

（2）$x-\dfrac{\sin x}{x}=0$，x_0=0.5。

（3）$\begin{cases} \sin x + y^2 + \ln z - 7 = 0 \\ 3x + 2^y - z^3 + 1 = 0 \\ x + y + z - 5 = 0 \end{cases}$，初值 x_0=1，y_0=1，z_0=1。

3．求常微分方程的数值解。

（1）$y'+(1.2+\sin 10t)y=0,\ t_0=0,\ t_f=5,\ y(t_0)=1$。

（2）$y'+\dfrac{1}{1+t^2}y=\cos t,\ t_0=0,\ t_f=5,\ y(t_0)=1$。

4．求函数在指定区间的最大值。

$$f(x)=\frac{1+x^2}{1+x^4},\ x\in(0,2)$$

5．设有 400 万元资金，要求 4 年内使用完，若在一年内使用资金 x 万元，则可得效益 $\sqrt{x}$ 万元（效益不能再使用），当年不用的资金可存入银行，年利率为 10%。试制定出资金的使用计划，以使 4 年效益之和为最大。

思考练习

1．分别用矩阵求逆、矩阵除法以及矩阵分解求线性方程组的解。

（1）$\begin{cases} 2x+3y+5z=10 \\ 3x+7y+4z=3 \\ x-7y+z=5 \end{cases}$

（2）$\begin{cases} 6x_1+5x_2-2x_3+5x_4=-4 \\ 9x_1-x_2+4x_3-x_4=13 \\ 3x_1+4x_2+2x_3-2x_4=1 \\ 3x_1-9x_2+2x_4=11 \end{cases}$

2．求下列方程的解。

（1）$3x+\sin x-e^x=0$ 在 x_0=1.5 附近的解。

（2）$x-\dfrac{1}{x}+5=0$ 在 x_0=1 附近的解。

（3）$\begin{cases} x^2+y^2=9 \\ x+y=1 \end{cases}$，初值 x_0=3，y_0=0。

3．求常微分方程的数值解。

（1）$(1+t^2)y''+2ty'+3y=2,\ t_0=0,\ t_f=5,\ y(t_0)=0,\ y'(t_0)=1$。

（2）$y'''-5\dfrac{\cos 2t}{(t+1)^2}y''+y'+\dfrac{1}{3+\sin t}y=\cos t,$

$t_0=0,\ t_f=5,\ y(t_0)=1,\ y'(t_0)=0,\ y''(t_0)=2$。

4．求函数在指定区间的最大值。

$$f(x)=\sin x+\cos x^2,\ x\in(0,\pi)$$

5．对边长为 3m 的正方形铁板，在 4 个角剪去相等的正方形以制成方形无盖水槽，问何种剪法使水槽的容积最大？