第6、7章　线性方程组求解

陈路 CHENLU.SCIEN@GMAIL.COM

201800301206

1. 上机实验题

问题 1

求解线性方程组

$$\begin{cases} 4x - y + z = 7 \\ 4x - 8y + z = -21 \\ -2x + y + 5z = 15 \end{cases} \tag{1}$$

(1)试用LU分解求解此方程组

(2)分别用Jacobi, Gauss-Seidel方法求解此方程组

解:

(1)利用高斯消去法构造方程组1的系数矩阵**A**的三角分解

通过将单位矩阵放在**A**的左边来构造矩阵**L**.

$$\begin{aligned} A &= \begin{pmatrix} 4 & -1 & 1 \\ 4 & -8 & 1 \\ -2 & 1 & 5 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 4 & -1 & 1 \\ 4 & -8 & 1 \\ -2 & 1 & 5 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ -0.5 & 0 & 1 \end{pmatrix} \begin{pmatrix} 4 & -1 & 1 \\ 0 & -7 & 0 \\ 0 & 0.5 & 5.5 \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ -0.5 & -1/14 & 1 \end{pmatrix} \begin{pmatrix} 4 & -1 & 1 \\ 0 & -7 & 0 \\ 0 & 0 & 5.5 \end{pmatrix} \end{aligned}$$

首先利用前向替换法对方程组**LY=B**求解**Y**

$$\begin{cases} y_1 = 7 \\ y_1 + y_2 = -21 \\ -\frac{1}{2}y_1 - \frac{1}{14}y_2 + y_3 = 15 \end{cases}$$

得到$\mathbf{Y} = [7 \quad -28 \quad \frac{33}{2}]$.

接下来表示方程组**UX=Y**为

$$\begin{cases} 4x_1 - x_2 + x_3 = 7 \\ -7x_2 = -28 \\ 5.5x_3 = \frac{33}{2} \end{cases}$$

得到$\mathbf{X} = [2 \quad 4 \quad 3]$.

(2)方程组1的Jacobi迭代过程:

$$x_{k+1} = \frac{7 + y_k - z_k}{4}$$
$$y_{k+1} = \frac{21 + 4x_k + z_k}{8}$$
$$z_{k+1} = \frac{15 + 2x_k - y_k}{5}$$

针对本题，我们可以简单编制如下程序求解：

```
syms x y z;
px(1)=1;
py(1)=2;
pz(1)=2;
format long;
for i=2:20
  px(i)=(7+py(i-1)-pz(i-1))/4
  py(i)=(21+4*px(i-1)+pz(i-1))/8
  pz(i)=(15+2*px(i-1)-py(i-1))/5
end
```

本题的迭代过程如表格1所示.

更一般的Jacobi迭代算法实现如下：

```
function X = jacobi(A, B, P ,delta , max1)
% Input - A is an NxN nonsingular matrix
%       - B is an Nx1 matrix
%       - P is an Nx1 matrix; the initial guess
%       - delta is the tolerance for P
%       - max1 is the maximum number of iterations
```

Table 1: Jacobi迭代过程

k	x_k	y_k	z_k
0	1.0000000000	2.0000000000	2.0000000000
1	1.7500000000	3.3750000000	3.0000000000
2	1.8437500000	3.8750000000	3.0250000000
3	1.9625000000	3.9250000000	2.9625000000
4	1.9906250000	3.9765625000	3.0000000000
5	1.9941406250	3.9953125000	3.0009375000
6	1.9985937500	3.9971875000	2.9985937500
7	1.9996484375	3.9991210938	3.0000000000
8	1.9997802734	3.9998242188	3.0000351563
9	1.9999472656	3.9998945313	2.9999472656
$\vdots$	$\vdots$	$\vdots$	$\vdots$
19	1.9999999993	3.9999999983	3.0000000018
20	2.00000000	4.00000000	3.00000000

```
% Output - X is an N x 1 matrix: the jacobi approximation to
%            the solution of AX = B

N = length(B);
for k = 1:max1
  for j = 1:N
    X(j)=(B(j)-A(j,[1:j-1,j+1:N])*P([1:j-1,j+1:N]))/A(j,j);
  end
  err = abs (norm(X'-P));
  relerr = err/(norm(X) + eps);
  P = X';
  if (err < delta) || (relerr < delta)
    break,end
end
X=X';
end
```

方程组1的Gauss-Seidel迭代过程:

$$x_{k+1} = \frac{7 + y_k - z_k}{4}$$
$$y_{k+1} = \frac{21 + 4x_{k+1} + z_k}{8}$$
$$z_{k+1} = \frac{15 + 2x_{k+1} - y_{k+1}}{5}$$

针对本题，我们可以简单编制如下程序求解：

```
syms x y z;
px(1)=1;
py(1)=2;
pz(1)=2;
format long;
for i=2:15
  px(i)=(7+py(i-1)-pz(i-1))/4
  py(i)=(21+4*px(i)+pz(i-1))/8
  pz(i)=(15+2*px(i)-py(i))/5
end
```

本题的迭代过程如表格2所示.

Table 2: Gauss-Seidel迭代过程

k	x_k	y_k	z_k
0	1.0000000000	2.0000000000	2.0000000000
1	1.7500000000	3.3750000000	2.9500000000
2	1.9500000000	3.9687500000	2.9862500000
3	1.9956250000	3.9960937500	2.9990312500
4	1.9992656250	3.9995117188	2.9998039063
5	1.9999269531	3.9999389648	2.9999829883
6	1.9999889941	3.9999923706	2.9999971235
7	1.9999988118	3.9999990463	2.9999997154
8	1.9999998327	3.9999998808	2.9999999569
9	1.9999999809	3.9999999851	2.9999999954
⋮	⋮	⋮	⋮
13	2.0000000000	4.0000000000	3.0000000000

更一般的Jacobi迭代算法实现如下：

```
function X = gseid(A, B, P ,delta, max1)
% Input - A  is an NxN nonsingular matrix
%       - B  is an Nx1 matrix
%       - P  is an Nx1 matrix; the initial guess
%       - delta is the tolerance for P
%       - max1 is the maximum number of iterations
% Output - X is an N x 1 matrix: the gauss-seidel
%           approximation to the solution of AX = B

N = length(B);
for k = 1:max1
  for j = 1:N
  if j == 1
    X(1) = (B(1)-A(1,2:N) *P(2:N))/A(1,1);
```

```
    elseif j == N
      X(N) = (B(N)-A(N,1:N-1)*(X(1:N-1))')/A(N,N);
    else
      % X contains the kth approximations and P the (k-1)st
      X(j) = (B(j)-A(j,1:j-1)*X(1:j-1)'...
          -A(j, j+1:N)*P(j+1:N))/A(j,j);
    end
  end
  err = abs(norm(X'-P));
  relerr = err/(norm(X) + eps);
  P = X';
  if (err<de1ta) || (relerr<delta)
    break, end
  end
  X = X';
end
```