基于 Matlab 实现线性方程组的迭代解法

王学彬

（武夷学院　数学与计算机学院，福建　武夷山　354300）

摘　要：本文结合求解线性方程组的迭代法，介绍了如何利用 MatLab 软件求解线性方程组，并给出具体实例。

关键词：Matlab；线性方程组；数值解

中图分类号：O241.6　　**文献标识码**：A　　**文章编号**：1674-2109(2014)05-0006-04

1　Matlab 软件和迭代法简介

Matlab 是由 The Math Works 公司开发的一套强大的数学软件，现已成为国际上最流行的科学计算与工程计算软件工具之一，Matlab 主要面对科学计算、可视化及交互式程序设计的高科技计算环境，这使得它的使用不仅仅局限在控制领域和数值分析领域内，在金融分析、神经网络、优化、虚拟现实等许多领域也都被广泛使用[1-2]。Matlab 在数值计算中有着其它软件无法比拟的优势，文献[3-5]考虑了基于 Matlab 求解常微分方程及在微积分中的一些应用。本文将介绍如何利用 Matlab 实现线性方程组的迭代解法。

设有线性方程组 Ax=b，A 为非奇异矩阵，首先将 A 分裂为 A=M-N。其中 M 一般选择为 A 的某种近似，且为非奇异矩阵，可称其为分裂矩阵，由分裂矩阵选取的不同，得到不同的迭代法，常见的有雅可比迭代法、高斯-塞德尔迭代法、逐次超松弛迭代法[6]。

本文将针对例 1 基于 Matlab 实现上述三种方法，并对运行结果进行分析。（要求计算精度为 10^{-5}）

例 1 利用迭代法解线性方程组 Ax=b，

$$A=\begin{bmatrix}-4 & 1 & 1 & 1\\ 1 & -4 & 1 & 1\\ 1 & 1 & -4 & 1\\ 1 & 1 & 1 & -4\end{bmatrix}\quad b=\begin{bmatrix}1\\1\\1\\1\end{bmatrix}$$

2　迭代法的 Matlab 实现

2.1　雅可比迭代法的 Matlab 实现

步骤一：编写雅可比迭代法的程序。

打开 Editor 编辑器，输入：

```
function [x,k]=Zhujacobi(A,b,x0,tol)
    max1= 300;    D=diag(diag(A));
L=-tril(A,-1);
U=-triu(A,1);
B=D\(L+U);
f=D\b;
x=B*x0+f;
k=1;
while norm(x-x0)>=tol
   x0=x;
    x=B*x0+f;
   k=k+1;
   if(k>=max1)
      disp(' 迭代超过 300 次，方程组可能不收敛 ');
```

收稿日期：2014-07-28

基金项目：武夷学院质量工程项目“数值分析重点课程建设”(项目编号：Jgzk201019)，武夷学院青年教师专项科研基金(项目编号：xq201022)。

作者简介：王学彬(1976-)，男，汉族，副教授，主要研究方向：分数阶微积分，偏微分方程。

```
        return;
    end
    [k vpa(x,10)']
end
```

以文件名 zhujacobi.m 保存

步骤二:编写验证程序

```
a=[-4 1 1 1;1 -4 1 1;1 1 -4 1;1 1 1 -4];
b=[1 1 1 1]';
x0=[0 0 0 0]' ;
 [x,k]=Zhujacobi(a,b,x0,1e-5)
```

以文件名 jacobi.m 保存

步骤三:运行验证程序 jacobi.m,可得到运算结果见表 1。

2.2 高斯-塞德尔迭代法的 Matlab 实现

步骤一:编写高斯-塞德尔迭代法的程序。

打开 Editor 编辑器,输入:

```
function [x,k]=Zhugseid(A,b,x0,tol)
    max1= 300;
      D=diag(diag(A));
L=-tril(A,-1);
U=-triu(A,1);
G=(D-L)\U;
f=(D-L)\b;
x=G*x0+f;
k=1;
while norm(x-x0)>=tol
    x0=x;
    x=G*x0+f;
    k=k+1;
    if(k>=max1)
        disp(' 迭代超过 300 次,方程组可能不收敛 ');
        return;
```

表 1 雅可比迭代法结果

Tab.1 Results of the Jacobi iterative method

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$x_4^{(k)}$
1	-.4375000000	-.4375000000	-.4375000000	-.4375000000
2	-.5781250000	-.5781250000	-.5781250000	-.5781250000
3	-.6835937500	-.6835937500	-.6835937500	-.6835937500
4	-.7626953125	-.7626953125	-.7626953125	-.7626953125
5	-.8220214844	-.8220214844	-.8220214844	-.8220214844
6	-.8665161133	-.8665161133	-.8665161133	-.8665161133
7	-.8998870850	-.8998870850	-.8998870850	-.8998870850
8	-.9249153137	-.9249153137	-.9249153137	-.9249153137
9	-.9436864853	-.9436864853	-.9436864853	-.9436864853
10	-.9577648640	-.9577648640	-.9577648640	-.9577648640
11	-.9683236480	-.9683236480	-.9683236480	-.9683236480
12	-.9762427360	-.9762427360	-.9762427360	-.9762427360
13	-.9821820520	-.9821820520	-.9821820520	-.9821820520
14	-.9866365390	-.9866365390	-.9866365390	-.9866365390
	.....			
36	-.9999761622	-.9999761622	-.9999761622	-.9999761622
37	-.9999821216	-.9999821216	-.9999821216	-.9999821216
38	-.9999865912	-.9999865912	-.9999865912	-.9999865912

```
    end
    [k,vpa(x,10)']
  end
```

以文件名 Zhugseid.m 保存

步骤二:编写验证程序

```
a=[-4 1 1 1;1 -4 1 1;1 1 -4 1;1 1 1 -4];
b=[1 1 1 1]';
x0=[0 0 0 0]' ;
[x,k]=Zhugseid(a,b,x0,1e-5)
```

以文件名 gseid.m 保存

步骤三:运行验证程序 gseid.m,可得到运算结果见表 2。

2.3 逐次超松弛迭代法的 Matlab 实现

步骤一:编写逐次超松弛迭代法的程序。

打开 Editor 编辑器,输入:

```
function [x,k]=Zhusor(A,b,x0,w,tol)
    max = 300;
if(w<=0 || w>=2)
    error;
    return;
end
 D=diag(diag(A));
L=-tril(A,-1);
U=-triu(A,1);
B=inv(D-L*w)*((1-w)*D+w*U);
f=w*inv((D-L*w))*b;
x=B*x0+f;
k=1;
while norm(x-x0)>=tol
    x0=x;
    x =B*x0+f;
    k=k+1;
    if(k>=max)
        disp(' 迭代超过 300 次,方程组可能不收敛 ');
        return;
    end
    [k,vpa(x,10)']
  end
```

表 2 高斯-塞德尔迭代法结果

Tab.2 Results of the Gauss Seidel iteration method

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$x_4^{(k)}$
1	-.5478515625	-.6066894531	-.6607055664	-.7038116455
2	-.7428016663	-.7768297195	-.8058607578	-.8313730359
3	-.8535158783	-.8726874180	-.8893940831	-.9038993448
4	-.9164952115	-.9274471598	-.9369604290	-.9452257001
5	-.9524083222	-.9586486128	-.9640706588	-.9687818985
6	-.9728752925	-.9764319624	-.9795222884	-.9822073858
7	-.9845404092	-.9865675208	-.9883288290	-.9898591897
8	-.9911888849	-.9923442259	-.9933480751	-.9942202965
9	-.9949781494	-.9956366302	-.9962087690	-.9967058872
10	-.9971378216	-.9975131194	-.9978392071	-.9981225370
11	-.9983687159	-.9985826150	-.9987684670	-.9989299495
12	-.9990702579	-.9991921686	-.9992980940	-.9993901301
13	-.9994700982	-.9995395806	-.9995999522	-.9996524077
14	-.9996979851	-.9997375863	-.9997719948	-.9998018915
15	-.9998278681	-.9998504386	-.9998700496	-.9998870891
16	-.9999018943	-.9999147582	-.9999259354	-.9999356470
17	-.9999440852	-.9999514169	-.9999577873	-.9999633223
18	-.9999681316	-.9999723103	-.9999759411	-.9999790957
19	-.9999818368	-.9999842184	-.9999862877	-.9999880857
20	-.9999896480	-.9999910054	-.9999921848	-.9999932095
21	-.9999940999	-.9999948735	-.9999955457	-.9999961298

以文件名 Zhusor.m 保存

步骤二:编写验证程序

```
a=[-4 1 1 1;1 -4 1 1;1 1 -4 1;1 1 1 -4];
b=[1 1 1 1]';
x0=[0 0 0 0]' ;
[x,k]=Zhusor(a,b,x0,1.3,1e-5)
```

以文件名 sor.m 保存

步骤三:运行验证程序 sor.m,可得到运算结果见表 3。

说明:这里 w=1.3,通过 w 取不同值,可验证其为

表 3 逐次超松弛迭代法结果

Tab.3 Results of the Successive over Relaxation iteration method

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$x_4^{(k)}$
1	-.7985962207	-.8864993674	-.9471878337	-.9536873074
2	-.9913178491	-.9990129116	-.9976495220	-1.009987400
3	-1.004765841	-1.004327024	-1.006906230	-1.002203486
4	-1.002937188	-1.002617136	-1.000449420	-1.001290171
5	-1.000534780	-.9999540295	-1.000443343	-.9999158981
6	-.9999413789	-1.000111493	-.9998568473	-.9999958892
7	-1.000005961	-.9999206288	-1.000017751	-.9999831441
8	-.9999727072	-1.000015232	-.9999852767	-.9999963520
9	-1.000007168	-.9999917892	-1.000002892	-1.000001695
10	-.9999966722	-1.000002873	-.9999995354	-.9999991925
11	-1.000001519	-.9999992183	-1.000000116	-1.000000520

最佳松弛因子。相对其它松弛因子,此时逐次超松弛迭代法的收敛速度最快。当 w=1 时,结果同表 2,验证了"当 w=1 时,逐次超松弛迭代法即为高斯-塞德尔迭代法"这一结论。

3 小结

由表 1-表 3 即可对比三中迭代法解线性方程组的收敛速度,为方便起见,列表如下:

表 4 例 1 中三种迭代方法的收敛速度比较

Tab.4 Comparison of the three iterative methods' convergence speed in example 1

	n	m
雅可比迭代法	12	38
高斯-塞德尔迭代法	7	21
逐次超松弛迭代法	2	11

n:迭代过程中,使方程的根不再有大幅的振荡,而是逐渐趋于平稳,接近收敛值所需的迭代次数

m:迭代法达到要求的精度所需的迭代次数,

由三种迭代法的计算公式,我们知道,高斯-塞德尔迭代法可看作是雅可比迭代法的一种改进,逐次超松弛迭代法是高斯-塞德尔迭代法的一种修正,这一理论结果也由表 4 中三种方法的收敛速度差异得到了验证。

参考文献:

[1] 宋叶志,贾东永. Matlab 数值分析与应用[M].北京:机械工业出版社,2009.

[2] 占海明.基于 Matlab 的高等数学问题求解[M].北京:清华大学出版社,2013.

[3] 王学彬. 基于 Matlab 求解常微分方程 [J]. 武夷学院学报,2010,29(5).

[4] 王学彬.利用 Matlab 求解高等数学中的一些问题[J].武夷学院学报,2009,28(2).

[5] 王学彬.利用 Madab 求解导数的一些问题[J].武夷学院学报,2009,28(5).

[6] 李庆扬,王能超,易大义等.数值分析(第五版)[M].北京:清华大学出版社,2012.

(下转第 38 页)