**数值分析计算实习题（6、7、8、9章）**

信计2002 窦健文 学号：2020005055

# 一、第六章

## 1.1实习题题干

考虑泊松方程边值问题

该问题的解是.

1. 用N=10的正方形网格离散化，得到n=100的线性方程组。列出五点差分格式的线性方程组。
2. 用雅可比迭代法和SOR迭代法（w=1，1.25，1.5，1.75），迭代初值（i , j = 1 , 2 , ... , N）。计算到时停止。给出迭代次数k，和，是解函数在点()上的分量生成的向量。
3. 用CG方法（共轭梯度法）解（1）的线性方程组，要求同（2），比较计算结果。

## 1.2第一问

由题干条件可知需对矩形区域进行分割，得到均匀分布的100个数值点。由五点差分格式：

可得到泊松方程的差分格式：

即

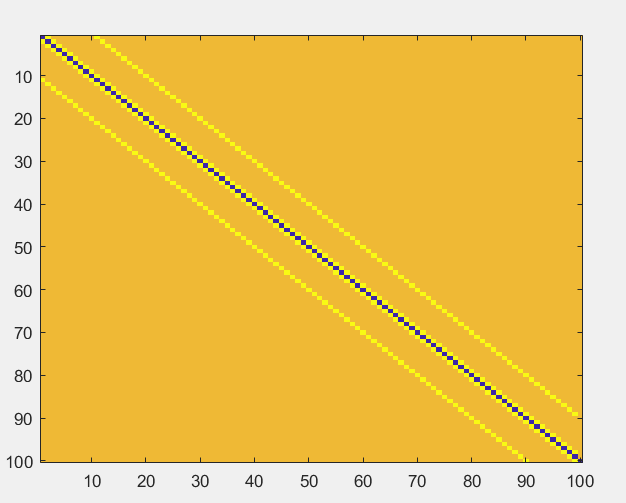
故可得第一问所需的五点差分线性方程组：

记上述方程为 ，其中为划分的坐标点数值解，为划分的坐标点解析解.

以下为第一问的Matlab源代码：

1. %泊松方程列出五点差分格式方程组
2. clear; close all; clc;
3. %默认条件
4. h1 = 1/10;  j1 = 1/10;  L1 = 1;
5. h1\_f = input('请输入横向步长:');
6. j1\_f = input('请输入纵向步长:');
7. if h1\_f
8. h1 = h1\_f;
9. end
10. if j1\_f
11. j1 = j1\_f;
12. end
13. L\_x1 = 1 / h1;  L\_t1 = L1 / j1;
14. %创建右端向量
15. L\_A = L\_x1\*L\_t1;
16. b = zeros(L\_A, 1);
17. for i = 1:L\_x1
18. for j = 1:L\_t1
19. x\_b = i \* h1;
20. y\_b = j \* j1;
21. b((i-1)\*10+j,1) = (x\_b^2 + y\_b^2)\*exp(x\_b\*y\_b);
22. end
23. end
24. b = h1^2 \* b;
25. %如需可视化需定义坐标点（目前不需可视化）
26. %a1 = h1:h1:1;  b1 = j1:j1:L1;
27. %[x\_1, t\_1] = meshgrid(a1, b1);
28. M = zeros(L\_A, 1);
29. %添加初值条件（边界条件）
30. for i = 1:L\_t1
31. M(i,1) = 1;
32. M(90+i,1) = exp(i \* j1);
33. M((i-1)\*10 + 1,1) = 1;
34. M(i\*10,1) = exp(i \* h1);
35. end
36. %通过推导求得系数矩阵
37. A = zeros(L\_A);
38. for i = 1:L\_A
39. A(i,i) = -4;
40. if i-1 > 0
41. A(i,i-1) = 1;
42. end
43. if i+1 < 100
44. A(i,i+1) = 1;
45. end
46. if i-10 > 0
47. A(i,i-10) = 1;
48. end
49. if i+10 < 100
50. A(i,i+10) = 1;
51. end
52. end
53. % 显示矩阵A
54. imagesc(A);

该代码最后将得到的系数矩阵结构显示出来：



可以直观地看到，系数矩阵结构即为的结构。

## 1.3第二问

首先对（1）中使用雅可比迭代法，迭代公式如下：

其中

以下为雅可比迭代法的Matlab源代码（应用第一问的初始条件）：

1. %泊松方程使用雅可比迭代
2. function q2\_Jacobi(A,b,x0,eps,N)
3. %功能：用Jacobi迭代法解n阶线性方程组Ax=b
4. n=length(b);
5. x=ones(n,1);
6. h1 = 1/10;  j1 = 1/10;  L1 = 1;
7. L\_t1 = L1 / j1;
8. for i = 1:L\_t1
9. x(i,1) = 1;
10. x(90+i,1) = exp(i \* j1);
11. x((i-1)\*10 + 1,1) = 1;
12. x(i\*10,1) = exp(i \* h1);
13. end
14. k=0;
15. %输入系数矩阵A，右端向量b,以及初始向量x0，精度eps,以及最大迭代次数N
16. %默认条件
17. h1 = 1/10;  j1 = 1/10;  L1 = 1;
18. L\_x1 = 1 / h1;  L\_t1 = L1 / j1;
19. L\_A = L\_x1\*L\_t1;
20. %创建解函数向量
21. p\_x = zeros(L\_A,1);
22. for i = 1:L\_x1
23. for j = 1:L\_t1
24. p = (i-1)\*10+j;
25. p\_x(p) = exp(i \* h1 \* j \* j1);
26. end
27. end
28. %当k≤N时，执行迭代步骤
29. while k<=N
30. %算出第k次迭代递推式
31. for i = 2:9
32. for j = 2:9
33. p = (i-1)\*10+j;
34. x(p)=(b(p)-A(p,[1:p-1,p+1:n])\*x0([1:p-1,p+1:n]))/A(p,p);
35. end
36. end
37. k=k+1;
38. %若||x\_k+1-x\_k||＜eps，则算法停止，输出方程组近似解x\_k+1，否则继续迭代
39. min = norm(x-x0,inf);
40. if min<eps, break;end
41. x0 = x;
42. end
43. %输出方程组的数值解和迭代信息。
44. if k>N
45. disp(['迭代次数=  ，算法超出最大迭代次数！',num2str(k)]);
46. else
47. disp(['迭代次数= ',num2str(k)]);
48. disp('-------------------------');
49. disp('最终向量为= ');
50. disp(x);
51. disp('-------------------------');
52. disp(['与解函数的误差= ',num2str(norm(x-p\_x,inf))]);
53. end

输出结果为：

1. >> q2\_Jacobi(A,b,M,10^-5,999)
2. 迭代次数= 153
3. -------------------------
4. 最终向量为=
5. 1.0000
6. 1.0000
7. 1.0000
8. 1.0000
9. 1.0000
10. 1.0000
11. 1.0000
12. 1.0000
13. 1.0000
14. 1.0000
15. 1.0000
16. 1.0142
17. 1.0289
18. 1.0440
19. 1.0597
20. 1.0766
21. 1.0957
22. 1.1194
23. 1.1543
24. 1.2214
25. 1.0000
26. 1.0289
27. 1.0587
28. 1.0895
29. 1.1216
30. 1.1556
31. 1.1929
32. 1.2355
33. 1.2867
34. 1.3499
35. 1.0000
36. 1.0440
37. 1.0895
38. 1.1365
39. 1.1854
40. 1.2369
41. 1.2920
42. 1.3522
43. 1.4188
44. 1.4918
45. 1.0000
46. 1.0597
47. 1.1216
48. 1.1854
49. 1.2516
50. 1.3211
51. 1.3947
52. 1.4736
53. 1.5583
54. 1.6487
55. 1.0000
56. 1.0766
57. 1.1556
58. 1.2369
59. 1.3211
60. 1.4094
61. 1.5028
62. 1.6024
63. 1.7088
64. 1.8221
65. 1.0000
66. 1.0957
67. 1.1929
68. 1.2920
69. 1.3947
70. 1.5028
71. 1.6177
72. 1.7406
73. 1.8725
74. 2.0138
75. 1.0000
76. 1.1194
77. 1.2355
78. 1.3522
79. 1.4736
80. 1.6024
81. 1.7406
82. 1.8898
83. 2.0510
84. 2.2255
85. 1.0000
86. 1.1543
87. 1.2867
88. 1.4188
89. 1.5583
90. 1.7088
91. 1.8725
92. 2.0510
93. 2.2462
94. 2.4596
95. 1.0000
96. 1.2214
97. 1.3499
98. 1.4918
99. 1.6487
100. 1.8221
101. 2.0138
102. 2.2255
103. 2.4596
104. 2.7183
105. -------------------------
106. 与解函数的误差= 0.10517

可以看到，该迭代算法在迭代153次后无穷范数小于要求值，达到标准，且数值解与解析解最大误差为0.10517.

然后对（1）中使用SOR迭代法，迭代公式如下：

其中,且w为松弛因子

以下为雅可比迭代法的Matlab源代码（应用第一问的初始条件）：

1. %泊松方程使用SOR迭代
2. function q2\_SOR(A,b,x0,eps,w,N)
3. %功能：用SOR迭代法解n阶线性方程组Ax=b
4. n=length(b);
5. x=ones(n,1);
6. h1 = 1/10;  j1 = 1/10;  L1 = 1;
7. L\_t1 = L1 / j1;
8. for i = 1:L\_t1
9. x(i,1) = 1;
10. x(90+i,1) = exp(i \* j1);
11. x((i-1)\*10 + 1,1) = 1;
12. x(i\*10,1) = exp(i \* h1);
13. end
14. k=0;
15. %输入系数矩阵A，右端向量b,以及初始向量x0，精度eps,以及最大迭代次数N
16. %默认条件
17. h1 = 1/10;  j1 = 1/10;  L1 = 1;
18. L\_x1 = 1 / h1;  L\_t1 = L1 / j1;
19. L\_A = L\_x1\*L\_t1;
20. %创建解函数向量
21. p\_x = zeros(L\_A,1);
22. for i = 1:L\_x1
23. for j = 1:L\_t1
24. p = (i-1)\*10+j;
25. p\_x(p) = exp(i \* h1 \* j \* j1);
26. end
27. end
28. %当k≤N时，执行迭代步骤
29. while k<=N
30. %算出第k次迭代递推式
31. for i = 2:9
32. for j = 2:9
33. p = (i-1)\*10+j;
34. x(p)=x0(p) + ( w \* (b(p)-A(p,1:p-1)\*x(1:p-1)-A(p,p:n)\*x0(p:n))/A(p,p) );
35. end
36. end
37. k=k+1;
38. %若||x\_k+1-x\_k||＜eps，则算法停止，输出方程组近似解x\_k+1，否则继续迭代
39. min = norm(x-x0,inf);
40. if min<eps, break;end
41. x0 = x;
42. end
43. %输出方程组的数值解和迭代信息。
44. if k>N
45. disp(['迭代次数=  ，算法超出最大迭代次数！',num2str(k)]);
46. else
47. disp(['迭代次数= ',num2str(k)]);
48. disp('-------------------------');
49. disp('最终向量为= ');
50. disp(x);
51. disp('-------------------------');
52. disp(['与解函数的误差= ',num2str(norm(x-p\_x,inf))]);
53. end

对w取1,1.25,1.5,1.75四个松弛系数，得到的数值解向量与雅可比迭代基本相同，以下为四个系数对应的迭代次数以及无穷范数：

1. >> q2\_SOR(A,b,M,10^-5,1,999)
2. 迭代次数= 84
3. -------------------------
4. 与解函数的误差= 0.10517
5. >> q2\_SOR(A,b,M,10^-5,1.25,999)
6. 迭代次数= 52
7. -------------------------
8. 与解函数的误差= 0.10517
9. >> q2\_SOR(A,b,M,10^-5,1.5,999)
10. 迭代次数= 22
11. -------------------------
12. 与解函数的误差= 0.10517
13. >> q2\_SOR(A,b,M,10^-5,1.75,999)
14. 迭代次数= 46
15. -------------------------
16. 与解函数的误差= 0.10517

可以看到，并不是提高松弛系数，迭代次数就会下降。

## 1.4第三问

第三问使用共轭梯度法（CG法）对（1）中的线性方程组进行迭代，因泊松方程的边界条件涉及到的数值点不应被算法更新，因此需要针对特定的序列下的数值点进行逐次更新，即将书中矩阵运算拆解成单个元素运算。

该算法步骤如下：

第一步，任取，计算，取。

第二步，创建各个迭代量并进行迭代：

显然，这种拆解方式十分繁琐且易错，但是通过这种方式我们可以确保应该保留的边界值能够得到保留。

以下为Matlab的CG方法源代码：

1. %使用CG方法解方程组(Matlab自带的pcg方法暂不清楚如何引入边界条件，故对书中步骤进行单元素拆解，引入边值条件)
2. function q3\_CG(A,b,x0,eps,N)
3. %默认条件
4. h1 = 1/10;  j1 = 1/10;  L1 = 1;
5. L\_x1 = 1 / h1;  L\_t1 = L1 / j1;
6. L\_A = L\_x1\*L\_t1;
7. %如需可视化需定义坐标点（目前不需可视化）
8. %a1 = h1:h1:1;  b1 = j1:j1:L1;
9. %[x\_1, t\_1] = meshgrid(a1, b1);
10. %创建解函数向量
11. p\_x = zeros(L\_A,1);
12. for i = 1:L\_x1
13. for j = 1:L\_t1
14. p = (i-1)\*10+j;
15. p\_x(p) = exp(i \* h1 \* j \* j1);
16. end
17. end
18. %创建x坐标向量
19. x = zeros(L\_A, 1);
20. for i = 1:L\_t1
21. x(i,1) = 1;
22. x(90+i,1) = exp(i \* j1);
23. x((i-1)\*10 + 1,1) = 1;
24. x(i\*10,1) = exp(i \* h1);
25. end
26. %创建r迭代向量
27. r = zeros(L\_A, N);
28. %创建α迭代向量
29. a = zeros(L\_A, 1);
30. %创建beta迭代向量
31. beta = zeros(L\_A, 1);
32. %创建p迭代向量
33. p = zeros(L\_A, 1);
34. %将p r迭代向量初始化
35. for i = 1:L\_A
36. p(i) = b(i) - A(i,:)\*x0;
37. r(i,1) = b(i) - A(i,:)\*x0;
38. end
39. %设置迭代次数
40. count = 0;
41. %开始CG算法
42. for i = 1:N
43. a(i) = (r(:,i)' \* r(:,i)) / (p' \* (A \* p));
44. for i1 = 2:9
45. for j1 = 2:9
46. pk = (i1-1)\*10+j1;
47. x(pk) = x0(pk) + a(i) \* p(pk);
48. end
49. end
50. for j = 1:L\_A
51. r(j,i+1) = r(j,i) - a(i) \*  (A(j,:) \* p);
52. end
53. beta(i) = (r(:,i+1)'\* r(:,i+1)) / (r(:,i)' \* r(:,i));
55. for j = 1:L\_A
56. p(j) = r(j,i+1) + beta(i) \* p(j);
57. end
58. count = count + 1;
59. %若||x\_k+1-x\_k||＜eps，则算法停止，输出方程组近似解x\_k+1，否则继续迭代
60. min = norm(x-x0,inf);
61. if min<eps, break;end
62. x0 = x;
63. end
64. %输出信息
65. if count>N
66. disp(['迭代次数=  ，算法超出最大迭代次数！',num2str(count)]);
67. else
68. disp(['迭代次数= ',num2str(count)]);
69. disp('-------------------------');
70. disp(['与解函数的误差= ',num2str(norm(x-p\_x,inf))]);
71. end

运算结果为：

1. >> q3\_CG(A,b,M,10^-5,999)
2. 迭代次数= 72
3. -------------------------
4. 与解函数的误差= 2.3672

可以看到误差十分地大，且经过多次检查后误差仍然无法改进，在排除代码内部逻辑问题以及语法问题后，通过推断与观察我们可以发现梯度共轭法迭代主要因素由系数矩阵所确定，且运算中尚未涉及到一个坐标值与相邻坐标值的数值关系，因此边值条件提供的信息尚且无法加入到梯度共轭方法中，也就导致了较大的误差。

# 第七章

## 2.1实习题题干

给出方程组

1. 建立一个在域上满足压缩映射定理的不动点迭代法，取计算方程的根。
2. 用牛顿法求解方程，至少用三个不同初值计算，计算到停止

## 2.2第一问

首先我们需要对找到的迭代公式进行收敛性验证，（其中）即先证明：

为此我们先对推导出的迭代公式

进行在区间上最大最小值的计算，不难发现：

可见均能满足该条件。

然后我们对迭代公式进行压缩映射原理的证明：

首先对以上各个迭代公式进行利普希茨条件的证明：

证明如下：

即满足利普希茨条件。

即满足利普希茨条件。

即满足利普希茨条件。

且由

可知迭代方程组满足压缩映射定理中的压缩条件。

故取，使用不动点迭代法计算方程的根。

以下为迭代法计算的Matlab源代码：

1. %建立满足压缩映射原理的不动点迭代法，并计算方程的根
2. function q1(N,eps)
3. %创建迭代向量并将其作为迭代初始点
4. x=zeros(3,1);
5. x0 = input('请输入迭代向量初始点（默认为0）:');
6. if x0
7. x = x0;
8. end
9. %创建中间向量
10. temp = zeros(3,1);
11. %开始迭代
12. k = 0;
13. while k < N
14. %为保证同步性使用中间向量进行传递
15. temp(1) = cos(x(2)\*x(3)+0.5) / 3;
16. temp(2) = ((x(1)^2+sin(x(3))+1.06)/81)^0.5 - 0.1;
17. temp(3) = (1 - 10\*pi/3 - exp(-x(1)\*x(2))) / 20;
18. %提前判断循环跳出条件
19. min = norm(x-temp,inf);
20. %将中间值传入迭代向量
21. for j = 1:3
22. x(j) = temp(j);
23. end
24. %是否跳出循环
25. if min<eps, break;end
26. k = k + 1;
27. end
28. %展示循环次数，迭代结果
29. if k>N
30. disp(['迭代次数=  ，算法超出最大迭代次数！',num2str(k)]);
31. else
32. disp(['迭代次数= ',num2str(k)]);
33. disp('-------------------------');
34. disp('求得方程组的根为= ');
35. disp(x);
36. end

算法运行结果为：

1. >> q1(999,10^-5)
2. 请输入迭代向量初始点（默认为0）:
3. 迭代次数= 4
4. -------------------------
5. 求得方程组的根为=
6. 0.2916
7. -0.0108
8. -0.5238
9. >> q1(999,10^-10)
10. 请输入迭代向量初始点（默认为0）:
11. 迭代次数= 9
12. -------------------------
13. 求得方程组的根为=
14. 0.2916
15. -0.0108
16. -0.5238
17. >> q1(999,10^-100)
18. 请输入迭代向量初始点（默认为0）:
19. 迭代次数= 14
20. -------------------------
21. 求得方程组的根为=
22. 0.2916
23. -0.0108
24. -0.5238

可以看到，该迭代法很有效的找到了不动点，且在有限的迭代次数内能够达到极高的精度。

最后，我们证明迭代矩阵能够将D=映入自身，从而完成压缩映射原理的完整证明。

记

因为

且对一切都有

容易发现主对角线元素皆为零，故有

从而有

因此迭代矩阵满足压缩映射原理，第一问解答完毕。

## 2.3第二问

牛顿法需要对方程组矩阵进行求导运算，因此我们先将导矩阵求解出来。

记

则有

通过迭代式

即可得到方程组的根。

以下为牛顿法的Matlab源代码：

1. %使用牛顿法进行迭代
2. function q2(N)
3. %创建迭代向量并将其作为迭代初始点
4. x=zeros(3,1);
5. x0 = input('请输入迭代向量初始点（默认为0）:');
6. if x0
7. x = x0;
8. end
9. %创建最小误差值
10. eps=10^-8;
11. eps0 = input('请输入临界误差值（默认为10^-8）:');
12. if eps0
13. x = eps0;
14. end
15. %创建导矩阵
16. F = zeros(3,3);
17. %创建中间矩阵
18. b = zeros(3,1);
19. %开始迭代
20. k = 0;
21. while k < N
22. %更新导矩阵
23. F(1,1) = 3;
24. F(1,2) = x(3)\*sin(x(2)\*x(3));
25. F(1,3) = x(2)\*sin(x(2)\*x(3));
26. F(2,1) = 2\*x(1);
27. F(2,2) = -162\*(x(2)+0.1);
28. F(2,3) = cos(x(3));
29. F(3,1) = -exp(-x(1)\*x(2));
30. F(3,2) = -exp(-x(1)\*x(2));
31. F(3,3) = 20;
32. %更新中间矩阵
33. b(1) = 3\*x(1)-cos(x(2)\*x(3))-0.5;
34. b(2) = x(1)^2-81\*(x(2)+0.1)^2+sin(x(3))+1.06;
35. b(3) = exp(-x(1)\*x(2))+20\*x(3)+10\*pi/3-1;
36. %更新迭代步伐
37. temp = (F^-1) \* b;
38. %提前判断循环跳出条件
39. min = norm(-temp,inf);
40. %将中间值传入迭代向量
41. for j = 1:3
42. x(j) = x(j) - temp(j);
43. end
44. %是否跳出循环
45. if min<eps, break;end
46. k = k + 1;
47. end
48. %展示循环次数，迭代结果
49. if k>N
50. disp(['迭代次数=  ，算法超出最大迭代次数！',num2str(k)]);
51. else
52. disp(['迭代次数= ',num2str(k)]);
53. disp('-------------------------');
54. disp('求得方程组的根为= ');
55. disp(x);
56. end

采取三个不同的初值点进行迭代，运算结果为：

1. >> q2(10)
2. 请输入迭代向量初始点（默认为0）:
3. 请输入临界误差值（默认为10^-8）:
4. 迭代次数= 5
5. -------------------------
6. 求得方程组的根为=
7. 0.5000
8. -0.0000
9. -0.5236
10. >> q2(10)
11. 请输入迭代向量初始点（默认为0）:[1 2 3]
12. 请输入临界误差值（默认为10^-8）:
13. 迭代次数= 8
14. -------------------------
15. 求得方程组的根为=
16. 0.5000    0.0000   -0.5236
17. >> q2(10)
18. 请输入迭代向量初始点（默认为0）:[-0.2 0.4 0.6]
19. 请输入临界误差值（默认为10^-8）:
20. 迭代次数= 6
21. -------------------------
22. 求得方程组的根为=
23. 0.5000   -0.0000   -0.5236

可以看到，初值点的选择能够较大程度上影响迭代的次数，通过浏览书籍可以发现基于牛顿法有许多改进算法，例如引入迭代因子等，可以一定程度上减弱处置点选取产生的影响，不过针对此题仅用牛顿法足以较好的完成。至此第二问已经完整解答。

# 第八章

## 3.1实习题题干

已知矩阵

1. 用MATLAB函数“eig”求矩阵全部特征值。
2. 用基本QR方法求全部特征值（可用MATLAB函数“qr”实现矩阵的QR分解）

## 3.2第一问

对于矩阵而言，如果找到一个常数和非零向量，使得：

那么我们将C称作A矩阵的特征值，且x是矩阵A属于特征值C的特征向量。

通过MATLAB的函数eig，我们能够便捷地得到矩阵的特征值，以下为使用eig函数求解特征值的Matlab源代码：

1. %使用eig求解矩阵特征值
2. %创建矩阵
3. A = [10 7 8 7;
4. 7 5 6 5;
5. 8 6 10 9;
6. 7 5 9 10];
7. B = [2 3 4 5 6;
8. 4 4 5 6 7;
9. 0 3 6 7 8;
10. 0 0 2 8 9;
11. 0 0 0 1 0];
12. C = ones(6);
13. for i = 1:6
14. for j = 1:6
15. C(i,j) = 1/(i+j-1);
16. end
17. end
18. %求解矩阵特征值
19. a = eig(A);
20. b = eig(B);
21. c = eig(C);
22. %输出信息
23. disp(A);disp("的特征值为:");disp(a);
24. disp("----------------------")
25. disp(B);disp("的特征值为:");disp(b);
26. disp("----------------------")
27. disp(C);disp("的特征值为:");disp(c);

运算结果展示如下：

1. >> q1
2. 10     7     8     7
3. 7     5     6     5
4. 8     6    10     9
5. 7     5     9    10
6. 的特征值为:
7. 0.0102
8. 0.8431
9. 3.8581
10. 30.2887
11. ----------------------
12. 2     3     4     5     6
13. 4     4     5     6     7
14. 0     3     6     7     8
15. 0     0     2     8     9
16. 0     0     0     1     0
17. 的特征值为:
18. 13.1724
19. 6.5519
20. 1.5957
21. -0.3908
22. -0.9291
23. ----------------------
24. 1 至 3 列
25. 1.0000    0.5000    0.3333
26. 0.5000    0.3333    0.2500
27. 0.3333    0.2500    0.2000
28. 0.2500    0.2000    0.1667
29. 0.2000    0.1667    0.1429
30. 0.1667    0.1429    0.1250
31. 4 至 6 列
32. 0.2500    0.2000    0.1667
33. 0.2000    0.1667    0.1429
34. 0.1667    0.1429    0.1250
35. 0.1429    0.1250    0.1111
36. 0.1250    0.1111    0.1000
37. 0.1111    0.1000    0.0909
38. 的特征值为:
39. 0.0000
40. 0.0000
41. 0.0006
42. 0.0163
43. 0.2424
44. 1.6189

可以看到，Matlab对矩阵运算有着很高的效率，至此第一问解答完毕。

## 3.3第二问

QR算法是一种计算矩阵特征值的十分有效的算法，它通过不断将矩阵分解构造矩阵序列，进而使非奇异矩阵最终收敛到一个上三角矩阵，其主对角线元素即为矩阵的特征值。

设，该算法迭代公式如下：

其中为上三角矩阵。

通过Matlab函数的“qr”函数可以方便地进行矩阵的QR分解，以下为QR算法的Matlab源代码：

1. %使用QR分解法求解矩阵特征值
2. function q2\_QR(N)
3. %创建矩阵
4. A = [10 7 8 7;
5. 7 5 6 5;
6. 8 6 10 9;
7. 7 5 9 10];
8. B = [2 3 4 5 6;
9. 4 4 5 6 7;
10. 0 3 6 7 8;
11. 0 0 2 8 9;
12. 0 0 0 1 0];
13. C = ones(6);
14. for i = 1:6
15. for j = 1:6
16. C(i,j) = 1/(i+j-1);
17. end
18. end
19. %进行迭代
20. tempA = A;
21. tempB = B;
22. tempC = C;
23. for i = 1:N
24. [qa,ra] = qr(tempA);
25. tempA = ra \* qa;
27. [qb,rb] = qr(tempB);
28. tempB = rb \* qb;
29. [qc,rc] = qr(tempC);
30. tempC = rc \* qc;
31. end
32. %求解特征值
33. eigenA = zeros(length(A),1);
34. eigenB = zeros(length(B),1);
35. eigenC = zeros(length(C),1);
36. for j = 1:length(A)
37. eigenA(j) = A(j,j);
38. end
39. for j = 1:length(B)
40. eigenB(j) = B(j,j);
41. end
42. for j = 1:length(C)
43. eigenC(j) = C(j,j);
44. end
45. %输出信息
46. disp(A);disp("的特征值为:");disp(eigenA);
47. disp("---------------------------------")
48. disp(B);disp("的特征值为:");disp(eigenB);
49. disp("---------------------------------")
50. disp(C);disp("的特征值为:");disp(eigenC);

为了确定迭代矩阵序列成功收敛，我们采用两次不同的迭代数，结果如下：

1. >> q2\_QR(10)
2. 10     7     8     7
3. 7     5     6     5
4. 8     6    10     9
5. 7     5     9    10
6. 的特征值为:
7. 10
8. 5
9. 10
10. 10
11. ---------------------------------
12. 2     3     4     5     6
13. 4     4     5     6     7
14. 0     3     6     7     8
15. 0     0     2     8     9
16. 0     0     0     1     0
17. 的特征值为:
18. 2
19. 4
20. 6
21. 8
22. 0
23. ---------------------------------
24. 1 至 3 列
25. 1.0000    0.5000    0.3333
26. 0.5000    0.3333    0.2500
27. 0.3333    0.2500    0.2000
28. 0.2500    0.2000    0.1667
29. 0.2000    0.1667    0.1429
30. 0.1667    0.1429    0.1250
31. 4 至 6 列
32. 0.2500    0.2000    0.1667
33. 0.2000    0.1667    0.1429
34. 0.1667    0.1429    0.1250
35. 0.1429    0.1250    0.1111
36. 0.1250    0.1111    0.1000
37. 0.1111    0.1000    0.0909
38. 的特征值为:
39. 1.0000
40. 0.3333
41. 0.2000
42. 0.1429
43. 0.1111
44. 0.0909
45. >> q2\_QR(100)
46. 10     7     8     7
47. 7     5     6     5
48. 8     6    10     9
49. 7     5     9    10
50. 的特征值为:
51. 10
52. 5
53. 10
54. 10
55. ---------------------------------
56. 2     3     4     5     6
57. 4     4     5     6     7
58. 0     3     6     7     8
59. 0     0     2     8     9
60. 0     0     0     1     0
61. 的特征值为:
62. 2
63. 4
64. 6
65. 8
66. 0
67. ---------------------------------
68. 1 至 3 列
69. 1.0000    0.5000    0.3333
70. 0.5000    0.3333    0.2500
71. 0.3333    0.2500    0.2000
72. 0.2500    0.2000    0.1667
73. 0.2000    0.1667    0.1429
74. 0.1667    0.1429    0.1250
75. 4 至 6 列
76. 0.2500    0.2000    0.1667
77. 0.2000    0.1667    0.1429
78. 0.1667    0.1429    0.1250
79. 0.1429    0.1250    0.1111
80. 0.1250    0.1111    0.1000
81. 0.1111    0.1000    0.0909
82. 的特征值为:
83. 1.0000
84. 0.3333
85. 0.2000
86. 0.1429
87. 0.1111
88. 0.0909

可以看到，两次迭代运算均收敛到同一特征值，至此，第二问已解答完毕。

# 第九章

## 4.1实习题题干

考虑常微分方程组初值问题

其中

要求用四阶R-K方法及梯形法计算（可以直接用数学库的软件），根据计算结果画出函数的图形。

## 4.2第一问（四阶R-K方法）

显示龙格-库达方法是基于改进的欧拉法

即

对其中的增量函数通过增加右函数的计算值来提高精度，一般而言，增加的计算值f越多，精度越高，其计算公式如下所示：

其中r的数值即为龙格-库达方法的阶数，在确定r值后，公式有许多常数需要通过局部截断误差，通过控制误差精度，进而确定出常数之间的关系式，有时关系式甚至是非线性的，这就意味着可能存在多个常数组合均满足r阶龙格-库达方法。

在多r值很高的龙格-库达方法进行常数推导的过程中，需要对初始局部截断误差式进行多元泰勒展开，因此计算过程十分繁琐复杂，在此我们使用书中已经计算出来的4阶龙格-库达方法的常数进行计算，该计算公式如下所示：

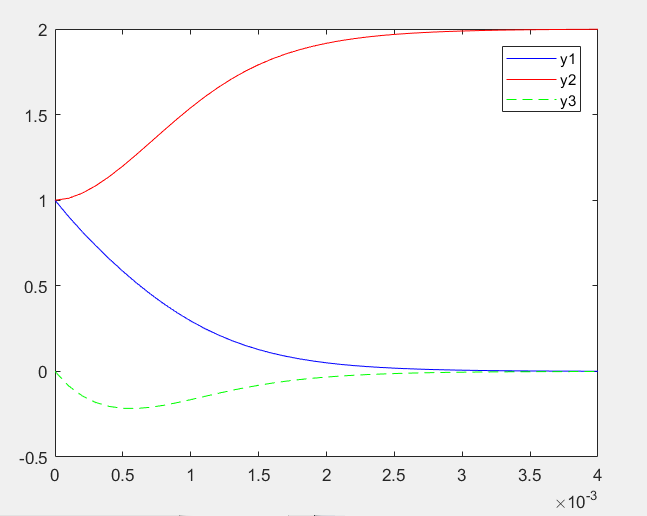
注意到方程组系数数量级差距较大，该方程组为刚性方程组。因此我们需要对步长以及时间长度进行调试，以寻得最佳参数。

以下为使用4阶龙格-库达方法计算常微分方程组初值问题的Matlab源代码：

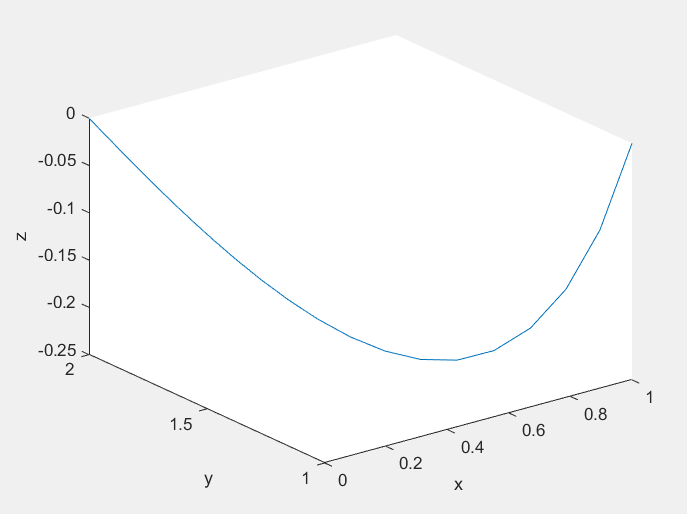
1. %四阶龙格库塔方法
2. function [t,z] = Runge\_Kutta(fun, t0, tf, Za, h)
3. %t0, tn为区间
4. %Za为初值
5. M = floor((tf-t0)/h) ;      %离散点的个数M+1
6. if t0 >= tf
7. printf('左端点必须小于右端点');
8. return;
9. end
10. N = length(Za);           %获得变量个数,N
11. z = zeros(M+1, N);
12. t =(t0 : h : tf)';
13. z(1,:) = Za';            %与微分方程中的变量方向统一，变成行向量
14. for i = 1:M
15. K1 =  feval(fun, t(i) , z(i,:));                    %K是行向量
16. K2 =  feval(fun, t(i)+1/2\*h ,z(i,:)+1/2\* h\*K1);
17. K3 =  feval(fun, t(i)+1/2\*h ,z(i,:)+1/2\* h\*K2);
18. K4 =  feval(fun, t(i)+ h ,z(i,:)+ h\*K3);
19. z(i+1,:) = z(i,:) + h/6 \*(K1 + 2\*K2 + 2\*K3 + K4);
20. end
21. %创建方程组
22. function s = equation(t,z)
23. dy1 = -0.013\*z(1) - 1000\*z(1)\*z(2);
24. dy2 = -2500\*z(2)\*z(3);
25. dy3 = -0.013\*z(1) - 1000\*z(1)\*z(2) - 2500\*z(2)\*z(3);
26. s = [dy1 dy2 dy3];
27. %主函数
28. %设置基础条件
29. format long
30. t0 = 0; tf = 0.004; %t0, tf为区间
31. Za = [1; 1; 0];     %x初值
32. h = 0.0001;
33. %代入龙格库达方法求解
34. [t,z] = Runge\_Kutta(@equation, t0, tf , Za, h);
35. %绘制图形
36. figure(1)
37. plot(t,z(:,1),'b',t,z(:,2), 'r',t,z(:,3), 'g--')
38. legend('y1','y2','y3')
39. figure(2)
40. plot3(z(:,1),z(:,2),z(:,3));
41. xlabel('x');ylabel('y');zlabel('z');

以下为运算输出图像：

（figure1）



（figure2）



可以发现，通过选择适当的步长与时间长度，该常微分方程组图像能够被良好的展示出来。至此第一问已解答完毕。

## 4.3第二问（梯形方法）

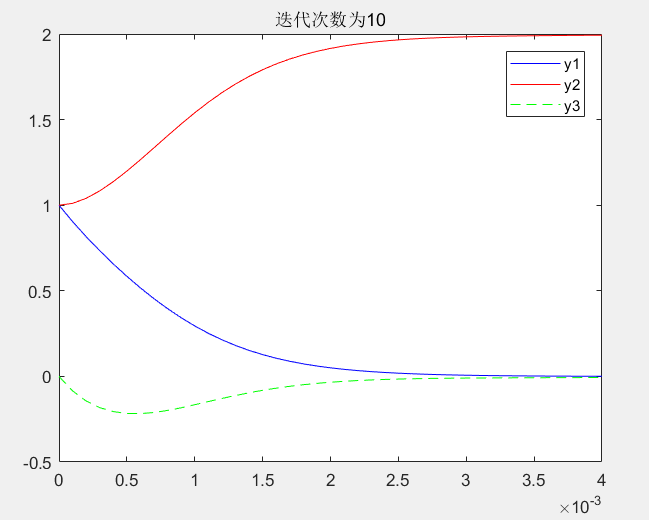
为了得到比欧拉公式精度更高的计算公式，将梯形求积公式代入得到

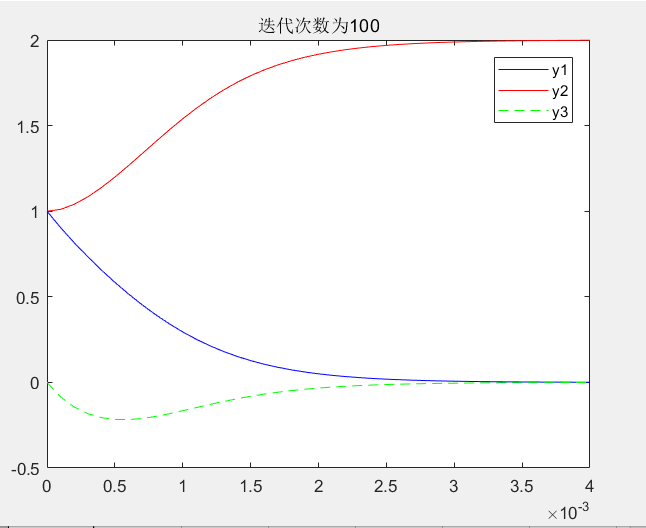
梯形方法为隐式单步法，可用迭代法求解，其迭代公式为：

以下为梯形方法的Matlab源代码：

1. %梯形方法
2. function [t,z] = Trapezoid(fun, t0, tf, Za, h, iter)
3. %t0, tn为区间
4. %Za为初值
5. M = floor((tf-t0)/h) ;      %离散点的个数M+1
6. if t0 >= tf
7. printf('左端点必须小于右端点');
8. return;
9. end
10. N = length(Za);           %获得变量个数,N
11. z = zeros(M+1, N);
12. t =(t0 : h : tf)';
13. z(1,:) = Za';            %与微分方程中的变量方向统一，变成行向量
14. for i = 1:M
15. z(i+1,:) = z(i,:) + h\*feval(fun, t(i) , z(i,:));
16. end
17. for j = 1:iter
18. for i = 1:M
19. z(i+1,:) = z(i,:) + h/2 \* (feval(fun, t(i) , z(i,:))+feval(fun, t(i+1) , z(i+1,:)));
20. end
21. end
22. %创建方程组
23. function s = equation(t,z)
24. dy1 = -0.013\*z(1) - 1000\*z(1)\*z(2);
25. dy2 = -2500\*z(2)\*z(3);
26. dy3 = -0.013\*z(1) - 1000\*z(1)\*z(2) - 2500\*z(2)\*z(3);
27. s = [dy1 dy2 dy3];
28. %主函数(第二问)
29. %设置基础条件
30. format long
31. t0 = 0; tf = 0.004; %t0, tf为区间
32. Za = [1; 1; 0];     %x初值
33. h = 0.0001;
34. iter = 10;   %迭代次数
35. %代入梯形方法求解
36. [t,z] = Trapezoid(@equation, t0, tf , Za, h, iter);
37. %绘制图形
38. figure(1)
39. plot(t,z(:,1),'b',t,z(:,2), 'r',t,z(:,3), 'g--')
40. legend('y1','y2','y3')
41. figure(2)
42. plot3(z(:,1),z(:,2),z(:,3));
43. xlabel('x');ylabel('y');zlabel('z');

为了确保迭代向量序列收敛，选取不同的迭代次数。求得数值解图像为：





函数图像的差别十分之小，可见数值解向量的序列已经完成收敛。经过此题我们知道，在计算常微分初值问题的时候，通过判断方程组的性质和特点选取适当的方法是能否很好的计算出数值解的重中之重，因此在面对类似的问题时，应即时搜寻相关适当方法，确保方程组数值解有足够高的精确度。

至此数值分析计算题已解答完毕。

2022/11/15