云南大学数学与统计学院实验教学中心

实验报告

|  |  |  |
| --- | --- | --- |
| **课程名称**：数学建模实验 | **学期：**2016~2017学年下学期 | |
| **指导教师**：李朝迁 | | |
| **学生：**刘鹏 20151910042 信计 | **学生：**王泽坤 20151910011 应数 | **学生：**段奕臣 20151910002 应数 |
| **实验名称**：**非线性方程求解** & MATLAB实现 | | **成绩**： |
| **实验编号**：NO.5 | **实验日期**：2017年7月1日 | **实验学时**：2 |
| **学院：**数学与统计学院 | **专业：**信息与计算科学 | **年级**：2015级 |

# 一、实验目的

1. 学习有关非线性方程求解的数学方法；

2. 学会用MATLAB进行非线性方程的求解；

# 二、实验内容

1. 对课后题进行编程求解；

2. 学会相关的MATLAB命令函数。

# 三、实验平台

Windows10 Enterprise 1703中文版操作系统；

MATLAB R2017a 中文版。

# 四、实验记录与实验结果分析

1题

利用help或者document学习solve命令。

**Solution**:

**solve**

Equations and systems solver

**Syntax**

S = solve (eqn, var)

S = solve (eqn, var, Name, Value)

Y = solve (eqns, vars)

Y = solve (eqns, vars, Name, Value)

[y1, ..., yN] = solve (eqns, vars)

[y1, ..., yN] = solve(eqns, vars, Name, Value)

[y1, ..., yN, parameters, conditions] = solve (eqns, vars, 'ReturnConditions', true)

**Description**

S = solve (eqn, var) solves the equation eqn for the variable var. If you do not specify var, the symvar function determines the variable to solve for. For example, solve (x + 1 == 2, x) solves the equation x + 1 = 2 for x.

S = solve(eqn, var, Name, Value) uses additional options specified by one or more Name,Value pair arguments.

Y = solve(eqns, vars) solves the system of equations eqns for the variables vars and returns a structure that contains the solutions. If you do not specify vars, solve uses symvar to find the variables to solve for. In this case, the number of variables that symvar finds is equal to the number of equations eqns.

Y = solve (eqns, vars, Name, Value) uses additional options specified by one or more Name,Value pair arguments.

[y1, ... , yN] = solve (eqns, vars) solves the system of equations eqns for the variables vars. The solutions are assigned to the variables y1, ... ,yN. If you do not specify the variables, solve uses symvar to find the variables to solve for. In this case, the number of variables that symvar finds is equal to the number of output arguments N.

[y1, ... , yN] = solve (eqns, vars, Name, Value) uses additional options specified by one or more Name,Value pair arguments.

[y1 , ..., yN, parameters, conditions] = solve (eqns, vars, 'ReturnConditions', true) returns the additional arguments parameters and conditions that specify the parameters in the solution and the conditions on the solution.

2题

对方程求根，步骤如下：

Step 1: 利用图形法确定根所在的位置；

Step 2: 构造迭代公式



初值由step1给出；

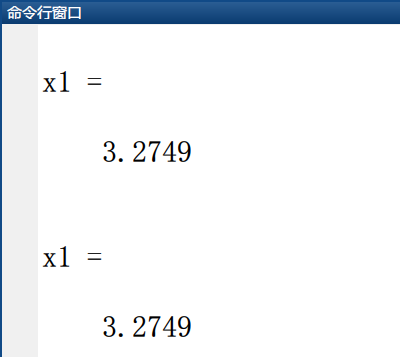
Step 3: 分别用迭代次数和作为终止条件，给出根的近似值

**Solution**:

程序代码：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | % filename: Test\_iteration  %- 方程求根 -%  clc**;**clear all**;**  x0 **=** 3**;**  x1 **=** 14 **/** **(**x0 **+** 1**);**  %- 迭代次数限制 -%  **while** i **<** 50  x0 **=** x1**;**  x1 **=** 14 **/** **(**x0 **+** 1**);**  i **=** i **+** 1**;**  **end**  x1  %- 差的绝对值限制 -%  clear all**;**  x0 **=** 3**;**  x1 **=** 14 **/** **(**x0 **+** 1**);**  **while** abs**(**x0 **-** x1**)** **>** 1e-6  x0 **=** x1**;**  x1 **=** 14 **/** **(**x0 **+** 1**);**  **end**  x1 |

运行结果：



代码分析：

根据迭代公式给出代码。比较简单的循环。

3题

请将Step 2中公式换为



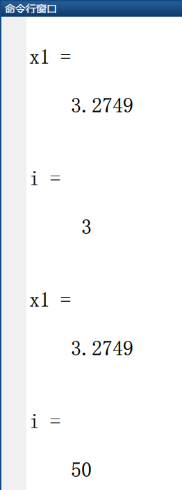
以及牛顿公式。然后再次计算。

**Solution:**

程序代码：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | % filename: Test\_iteration\_optim  %- 方程求根 -%  clc**;**clear all**;**  x0 **=** 3**;**  x1 **=** x0 **-** **(**x0**^**2 **+** x0 **-**14**)** **/** **(**2 **\*** x0 **+** 1**);**  %- 迭代次数限制 -%  i **=** 0**;**  **while** abs**(**x0 **-** x1**)** **>** 1e-6  x0 **=** x1**;**  x1 **=** x0 **-** **(**x0**^**2 **+** x0 **-**14**)** **/** **(**2 **\*** x0 **+** 1**);**  i **=** i **+** 1**;**  **end**  x1  i  %- 差的绝对值限制 -%  clear all**;**  i **=** 0**;**  x0 **=** 3**;**  x1 **=** 14 **/** **(**x0 **+** 1**);**  **while** abs**(**x0 **-** x1**)** **>** 1e-6  x0 **=** x1**;**  x1 **=** 14 **/** **(**x0 **+** 1**);**  i **=** i **+** 1**;**  **end**  x1  i |

运行结果：



代码分析：

可以看到，新的迭代公式下，只用了三次就很棒了，而老的方法却用了50次。差距相当大。

4题

分别用fzero和fsolve程序求方程的所有根，精确到，取不同的初值计算，输出**初值**、**根的近似值**和**迭代次数**，分析不同根的收敛域；自己构造某个迭代公式（如等），用迭代法求解，并自己编写牛顿法的程序进行求解和比较。

程序代码：

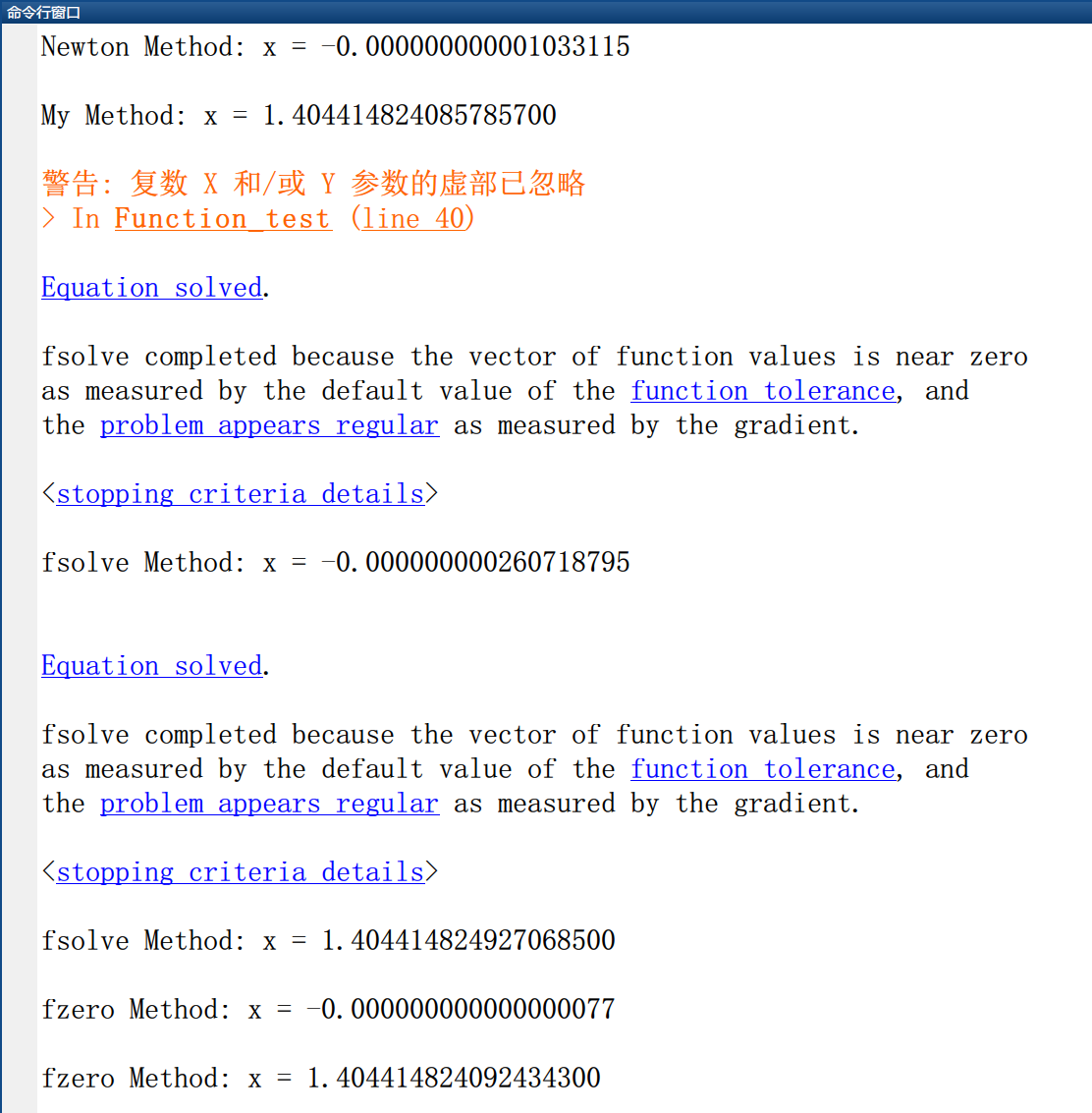
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99  100  101  102  103 | % filename: Function\_test  clc**;**clear all**;**  fun **=** **@(**x**)** sin**(**x**)** **-** x**.^**2 **/** 2**;**  Diff **=** **@(**x**)**cos**(**x**)** **-** x**;**  x **=** **-**0.5 **:** 0.01 **:** 1.5**;**  plot**(**x**,**fun**(**x**));**  %- 可以发现，在（-0.5，0.5）之内函数有零点 -%  x0 **=** x**(**1**);**  x1 **=** x0 **-** fun**(**x0**)** **/** Diff**(**x0**);**  **while** abs**(**x0 **-** x1**)** **>** 1e-10  x0 **=** x1**;**  x1 **=** x0 **-** fun**(**x0**)** **/** Diff**(**x0**);**  **end**  fprintf**(**'Newton Method: x = %3.18f\n\n'**,**x0**);**  hold on  plot**(**x0**,**fun**(**x0**),**'o'**)**  grid on  xlabel**(**'X'**);**  ylabel**(**'Y'**);**  %- 自行构造迭代公式 -%  clear all  Phi **=** **@(**x**)** sqrt**(**2 **\*** sin**(**x**));**  x0 **=** **-**1**;**  x1 **=** Phi**(**x0**);**  **while** abs**(**x0 **-** x1**)** **>** 1e-10  x0 **=** x1**;**  x1 **=** Phi**(**x0**);**  **end**  fprintf**(**'My Method: x = %3.18f\n\n'**,**x0**);**  figure**;**  fun **=** **@(**x**)** sin**(**x**)** **-** x**.^**2 **/** 2**;**  x **=** **-**0.5 **:** 0.01 **:** 1.5**;**  plot**(**x**,**fun**(**x**));**  hold on  plot**(**x0**,**fun**(**x0**),**'o'**)**  grid on  xlabel**(**'X'**);**  ylabel**(**'Y'**);**  %- fsolve: x0 = -1 -%  clear all  fun **=** **@(**x**)** sin**(**x**)** **-** x**.^**2 **/** 2**;**  x0 **=** fsolve**(**fun**,-**1**);**  fprintf**(**'fsolve Method: x = %3.18f\n\n'**,**x0**);**  figure  x **=** **-**0.5 **:** 0.01 **:** 1.5**;**  plot**(**x**,**fun**(**x**));**  hold on  plot**(**x0**,**fun**(**x0**),**'o'**)**  grid on  xlabel**(**'X'**);**  ylabel**(**'Y'**);**  %- fsolve: x0 = 1.5 -%  clear all  fun **=** **@(**x**)** sin**(**x**)** **-** x**.^**2 **/** 2**;**  x0 **=** fsolve**(**fun**,**1.5**);**  fprintf**(**'fsolve Method: x = %3.18f\n\n'**,**x0**);**  figure  x **=** **-**0.5 **:** 0.01 **:** 1.5**;**  plot**(**x**,**fun**(**x**));**  hold on  plot**(**x0**,**fun**(**x0**),**'o'**)**  grid on  xlabel**(**'X'**);**  ylabel**(**'Y'**);**  %- fzero: x0 = -1 -%  clear all  fun **=** **@(**x**)** sin**(**x**)** **-** x**.^**2 **/** 2**;**  x0 **=** fzero**(**fun**,-**1**);**  fprintf**(**'fzero Method: x = %3.18f\n\n'**,**x0**);**  figure  x **=** **-**0.5 **:** 0.01 **:** 1.5**;**  plot**(**x**,**fun**(**x**));**  hold on  plot**(**x0**,**fun**(**x0**),**'o'**)**  grid on  xlabel**(**'X'**);**  ylabel**(**'Y'**);**  %- fsolve: x0 = 1.5 -%  clear all  fun **=** **@(**x**)** sin**(**x**)** **-** x**.^**2 **/** 2**;**  x0 **=** fzero**(**fun**,**1.5**);**  fprintf**(**'fzero Method: x = %3.18f\n\n'**,**x0**);**  figure  x **=** **-**0.5 **:** 0.01 **:** 1.5**;**  plot**(**x**,**fun**(**x**));**  hold on  plot**(**x0**,**fun**(**x0**),**'o'**)**  grid on  xlabel**(**'X'**);**  ylabel**(**'Y'**);** |

运行结果：





由于生成的图片太多而且基本一样，这里就不全部展示了。总得来看，牛顿方法，自定义迭代方法，以及fsolve与fzero，采用的核心原理基本一致，都与给定的初值有关，所以，所得根都在初值的附近。



代码分析：

5题

给定4种物质对应的参数，，和交互作用矩阵***Q***如下：

*a*1 = 18.607, *a*2 = 15.841, *a*3 = 20.443, *a*4 = 19.293;

*b*1 = 2643.31, *b*2 = 2755.64, *b*3 = 4628.96, *b*4 = 4117.07;

*c*1 = 239.73, *c*2 = 219.16, *c*3 = 252.64, *c*4 = 227.44;



在压强p = 760mmHg下，为了形成均相共沸混合物，温度和组分分别是多少？请尽量找出所有可能的解。

程序代码：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33 | % filename: Mixed  clc**;**clear all  n **=** 4**;**  P **=** 760**;**  a **=** **[**18.607**,** 15.841**,** 20.443**,** 19.293**]';**  b **=** **[**2643.31**,** 2755.64**,** 4628.96**,** 4117.07**]';**  c **=** **[**239.73**,** 219.16**,** 252.64 227.44**]';**  Q **=** **[** 1.0 **,**0.192 **,**2.169 **,**1.611  0.316 **,**1.0 **,**0.477 **,**0.524  0.377 **,**0.360 **,**1.0 **,**0.296  0.524 **,**0.282 **,**2.065 **,**1.0**];**  XT0 **=** **[**0.50 0.10 0.20 30**];**  **[**XT**,**Y**]** **=** fsolve**(@**azeofun**,**XT0**,[],**n**,**P**,**a**,**b**,**c**,**Q**)**  **function** f **=** azeofun**(**XT**,**n**,**P**,**a**,**b**,**c**,**Q**)**  x**(**n**)** **=** 1**;**  **for** i **=** 1 **:** n **-** 1  x**(**i**)** **=** XT**(**i**);**  x**(**n**)** **=** x**(**n**)** **-** x**(**i**);**  **end**  T **=** XT**(**n**);**  p **=** log**(**P**);**  **for** i **=** 1 **:** n  d**(**i**)** **=** x **\*** Q**(**i**,**1**:**n**)';**  dd**(**i**)** **=** x**(**i**)** **/** d**(**i**);**  **end**  **for** i **=** 1 **:** n  f**(**i**)** **=** x**(**i**)** **\*** **(**b**(**i**)** **/** **(**T **+** c**(**i**))** **+** ...  log**(**x **\*** Q**(**i**,** 1**:**n**)')** **+** ...  dd **\*** Q**(**1**:**n**,**i**)** **-** a**(**i**)** **-** 1 **+** p**);**  **end**  **end** |

运行结果：

代码分析：

可以看到，第一种物质的参数都很小，所以最终结果的组分为0。

4. 设国民经济由农业、制造业和服务业三个部门构成，已知某年它们之间的投入产出关系、外部关系、初始投入等如表5.6所示。

表5.6 国民经济三个部门之间的投入产出表 单位：亿元o9

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 产出  投入 | 农业 | 制造业 | 服务业 | 外部需求 | 总产出 |
| 农业 | 15 | 20 | 30 | 35 | 100 |
| 制造业 | 30 | 10 | 45 | 115 | 200 |
| 服务业 | 20 | 60 | 0 | 70 | 150 |
| 初始投入 | 35 | 110 | 75 |  |  |
| 总投入 | 100 | 200 | 150 |  |  |

# 四、实验过程

设计算法，编写程序并调试。

# 五、实验总结

# 六、参考文献

[1] 大学数学实验/姜启源，谢金星，邢文训，张立平，北京：清华大学出版社，2010.12

[2] MATLAB教程/张志涌，杨祖樱，北京：北京航空航天大学出版社，2015.1

# 七、教师评语