北京航空航天大学数学科学学院实验报告

课程名称：科学计算通识实验课		实验名称： 非线性方程的迭代求解
实验类型： 演示性实验□ 验证性实验□ 综合性实验☑ 设计性实验□		
班级：18377475	姓名：陈博胆	学号：18377475
实验日期： 2020.07.13	指导教师：冯成亮	实验成绩：

实验环境：（所用仪器设备及软件）

Windows + Visual Studio 2019, Ubuntu 18.04.1 + g++

实验目的与实验内容：

【目的要求】

通过本实验使学生进一步熟悉个人电脑上 C++代码的编写与调试，服务器上的代码编译与运行；熟悉求解非线性方程的区间逼近法（二分法、试值法），不动点迭代法（简单迭代法、加速迭代法），和牛顿类迭代法（牛顿迭代法、割线法）；了解以上方法的算法的稳定性与收敛速度特点；熟悉高阶迭代法在处理特殊病态问题时的收敛性问题，体会二分法作为外部嵌套迭代算法的必要性。

【实验内容】

实验要求：最大迭代步数：100；

收敛要求：|f(x)|<10E-4 或 $\| x_{k+1}-x_k|$<10E-5；

输出每步 x 值与 f(x)或(x-φ(x))的值；

实验 1.1：（分别用二分法与试值法求解非线性方程 1）

用二分法与试值法求方程 $f(x)=xsin(x)-1=0$ 在（0,2）区间的根.

$(x^*=1.114157140871\text{93})$

实验 1.2：（分别用二分法与试值法求解非线性方程 2）

用二分法与试值法求方程 $f(x)=e^{-100x}-1=0$ 在（−0.51,0.49）区间的根.

$(x^*=0.0)$

实验 1.3：（分别用二分法与试值法求解非线性方程 3）

用二分法与试值法求方程$x^3+4x^2-10=0$在（1,2）区间的根.

$(x^*=1.3652300134141)$

实验 1.4：（分别用二分法与试值法求解非线性方程 4）

用二分法与试值法求方程 $x=1.6+0.99cosx$在（1,2）区间的根.

$(x^*=1.58547180152194)$

实验 1.5：（分别用二分法与试值法求解非线性方程 5）

用二分法与试值法求方程 $f(x)=x^3-x-1=0$ 在（1,2）区间的根.

（$x^* = 1.32471795724475$）

实验 2.1：（用简单迭代法求解非线性方程 3）

用简单迭代法求方程$x^3 + 4x^2 - 10 = 0$在（1,2）区间的根.

$$x_0 = 1.5 \;（x^* = 1.3652300134141）$$

其中迭代公式分别取：

$$x_{k+1} = \frac{1}{2}\sqrt{10 - x_k^3} \#(1)$$

$$x_{k+1} = \sqrt{\frac{10}{x_k} - 4x_k^2} \#(2)$$

$$x_{k+1} = x_k - x_k^3 - 4x_k^2 + 10 \#(3)$$

比较其收敛性差别。

实验 2.2：（分别用简单迭代法与加速迭代法求解非线性方程 4）

用简单迭代法与加速迭代法求方程 $x = 1.6 + 0.99\cos x$ 在（1,2）区间的根.

$$x_0 = \pi/2, \;（x^* = 1.58547180152194）$$

实验 2.3：（用简单迭代法与加速迭代法求解非线性方程 5）

用加速迭代法求方程 $f(x) = x^3 - x - 1 = 0$ 在（1,2）区间的根.

$$x_0 = 1.5 \;（x^* = 1.32471795724475）$$

其中简单迭代公式取：$x_{k+1} = x_k^3 - 1$,比较其收敛性差别。

实验 3.1：（用牛顿迭代法与割线法求解非线性方程 3）

用简单迭代法求方程$x^3 + 4x^2 - 10 = 0$在（1,2）区间的根.

$x_0 = 1.5$,（$x_0 = 1$, $x_1 = 2$ for 割线法），

（$x^* = 1.3652300134141$）。

实验 3.2：（分别用牛顿迭代法与割线法求解非线性方程 4）

用简单迭代法与加速迭代法求方程 $x = 1.6 + 0.99\cos x$ 在（1,2）区间的根.

$x_0 = \pi/2$,（$x_0 = 1$, $x_1 = 2$ for 割线法），

（$x^* = 1.58547180152194$）。

实验 3.3：（分别用牛顿迭代法与割线法求解非线性方程 5）

用加速迭代法求方程 $f(x) = x^3 - x - 1 = 0$ 在（1,2）区间的根.

$x_0 = 1.5$,（$x_0 = 1$, $x_1 = 2$ for 割线法），

（$x^* = 1.32471795724475$）。

实验 3.4：（分别用牛顿迭代法与割线法求解非线性方程 2）

用加速迭代法求方程 $f(x) = e^{-100x} - 1 = 0$ 在 $(-0.51,0.49)$ 区间的根.

$x_0 = -0.51$，（$x_0 = -0.51$，$x_1 = 0.49$ for 割线法），（$x^* = 0.0$）。

实验过程与结果：

【1】实验 1 中 5 个实验分别用二分法和试值法求解一元非线性方程的根，比较二者的求解效率以及稳定性等；可以发现，试值法在大部分情况下效率比二分法要好，但是在某些特殊情况下也可能不收敛，实验结果如下：

实验 1.1：（分别用二分法与试值法求解非线性方程 1）

实验 1.2：（分别用二分法与试值法求解非线性方程 2）

实验 1.3：（分别用二分法与试值法求解非线性方程 3）

实验 1.4：（分别用二分法与试值法求解非线性方程 4）

实验 1.5：（分别用二分法与试值法求解非线性方程 5）

```
work1@ws1:~/ChenBodan/test$ cd "/home/work1/ChenBodan/class4/" && g++ -std=c++11 1.cpp
-larmadillo -o 1 && "/home/work1/ChenBodan/class4/"1
                          f(x) = xsin(x) - 1
Dichotomy Method
cnt          l               mid             r               f(x)
0            0.00000000      1.00000000      2.00000000      -0.15852902
1            1.00000000      1.50000000      2.00000000      0.49624248
2            1.00000000      1.25000000      1.50000000      0.18623077
3            1.00000000      1.12500000      1.25000000      0.01505104
4            1.00000000      1.06250000      1.12500000      -0.07182663
5            1.06250000      1.09375000      1.12500000      -0.02836172
6            1.09375000      1.10937500      1.12500000      -0.00664277
7            1.10937500      1.11718750      1.12500000      0.00420803
8            1.10937500      1.11328125      1.11718750      -0.00121649
9            1.11328125      1.11523438      1.11718750      0.00149600
10           1.11328125      1.11425781      1.11523438      0.00013981
11           1.11328125      1.11376953      1.11425781      -0.00053832
12           1.11376953      1.11401367      1.11425781      -0.00019925
13           1.11401367      1.11413574      1.11425781      -0.00002972
14           1.11413574      1.11419678      1.11425781      0.00005505
15           1.11413574      1.11416626      1.11419678      0.00001266
16           1.11413574      1.11415100      1.11416626      -0.00000853

Try Method
cnt          l               mid             r               f(x)
0            0.00000000      1.09975017      2.00000000      -0.02001921
1            1.09975017      1.12124074      2.00000000      0.00983461
2            1.09975017      1.11416119      1.12124074      0.00000563
```

$f(x) = \exp(-10x) - 1$

Dichotomy Method

cnt	l	mid	r	f(x)
0	-0.51000000	-0.01000000	0.49000000	0.10517092
1	-0.01000000	0.24000000	0.49000000	-0.90928205
2	-0.01000000	0.11500000	0.24000000	-0.68336323
3	-0.01000000	0.05250000	0.11500000	-0.40844464
4	-0.01000000	0.02125000	0.05250000	-0.19143968
5	-0.01000000	0.00562500	0.02125000	-0.05469722
6	-0.01000000	-0.00218750	0.00562500	0.02211601
7	-0.00218750	0.00171875	0.00562500	-0.01704064
8	-0.00218750	-0.00023438	0.00171875	0.00234650
9	-0.00023438	0.00074219	0.00171875	-0.00739440
10	-0.00023438	0.00025391	0.00074219	-0.00253584
11	-0.00023438	0.00000977	0.00025391	-0.00009765
12	-0.00023438	-0.00011230	0.00000977	0.00112368
13	-0.00011230	-0.00005127	0.00000977	0.00051283
14	-0.00005127	-0.00002075	0.00000977	0.00020754
15	-0.00002075	-0.00000549	0.00000977	0.00005493
16	-0.00000549	0.00000214	0.00000977	-0.00002136
17	-0.00000549	-0.00000168	0.00000214	0.00001678
18	-0.00000168	0.00000023	0.00000214	-0.00000229

Try Method

cnt	l	mid	r	f(x)
0	-0.51000000	0.48394838	0.49000000	-0.99208886
1	-0.51000000	0.47793618	0.48394838	-0.99159864
2	-0.51000000	0.47196328	0.47793618	-0.99108155
3	-0.51000000	0.46602957	0.47196328	-0.99053634
4	-0.51000000	0.46013494	0.46602957	-0.98996172
5	-0.51000000	0.45427928	0.46013494	-0.98935636
6	-0.51000000	0.44846251	0.45427928	-0.98871888
7	-0.51000000	0.44268454	0.44846251	-0.98804786
8	-0.51000000	0.43694526	0.44268454	-0.98734183
9	-0.51000000	0.43124462	0.43694526	-0.98659927
10	-0.51000000	0.42558252	0.43124462	-0.98581862

40	-0.51000000	0.27354143	0.27804976	-0.93513287
41	-0.51000000	0.26907249	0.27354143	-0.93216825
42	-0.51000000	0.26464303	0.26907249	-0.92909614
43	-0.51000000	0.26025320	0.26464303	-0.92591424
44	-0.51000000	0.25590311	0.26025320	-0.92262032
45	-0.51000000	0.25159288	0.25590311	-0.91921216
46	-0.51000000	0.24732266	0.25159288	-0.91568762
47	-0.51000000	0.24309256	0.24732266	-0.91204462
48	-0.51000000	0.23890274	0.24309256	-0.90828116
49	-0.51000000	0.23475333	0.23890274	-0.90439530
50	-0.51000000	0.23064446	0.23475333	-0.90038521

Failed

f(x) = x^3 + 4x^2 - 10

Dichotomy Method

cnt	l	mid	r	f(x)
0	1.00000000	1.50000000	2.00000000	2.37500000
1	1.00000000	1.25000000	1.50000000	-1.79687500
2	1.25000000	1.37500000	1.50000000	0.16210938
3	1.25000000	1.31250000	1.37500000	-0.84838867
4	1.31250000	1.34375000	1.37500000	-0.35098267
5	1.34375000	1.35937500	1.37500000	-0.09640884
6	1.35937500	1.36718750	1.37500000	0.03235579
7	1.35937500	1.36328125	1.36718750	-0.03214997
8	1.36328125	1.36523438	1.36718750	0.00007202
9	1.36328125	1.36425781	1.36523438	-0.01604669
10	1.36425781	1.36474609	1.36523438	-0.00798926
11	1.36474609	1.36499023	1.36523438	-0.00395910
12	1.36499023	1.36511230	1.36523438	-0.00194366
13	1.36511230	1.36517334	1.36523438	-0.00093585
14	1.36517334	1.36520386	1.36523438	-0.00043192
15	1.36520386	1.36521912	1.36523438	-0.00017995
16	1.36521912	1.36522675	1.36523438	-0.00005396
17	1.36522675	1.36523056	1.36523438	0.00000903

Try Method

cnt	l	mid	r	f(x)
0	1.00000000	1.26315789	2.00000000	-1.60227438
1	1.26315789	1.33882784	2.00000000	-0.43036475
2	1.33882784	1.35854634	2.00000000	-0.11000879
3	1.35854634	1.36354744	2.00000000	-0.02776209
4	1.36354744	1.36480703	2.00000000	-0.00698342
5	1.36480703	1.36512372	2.00000000	-0.00175521
6	1.36512372	1.36520330	2.00000000	-0.00044106
7	1.36520330	1.36522330	2.00000000	-0.00011083
8	1.36522330	1.36522833	2.00000000	-0.00002785
9	1.36522833	1.36522959	2.00000000	-0.00000700

f(x) = 1.6 + 0.99cos(x) - x

Dichotomy Method

cnt	l	mid	r	f(x)
0	1.00000000	1.50000000	2.00000000	0.17002983
1	1.50000000	1.75000000	2.00000000	-0.32646360
2	1.50000000	1.62500000	1.75000000	-0.07863536
3	1.50000000	1.56250000	1.62500000	0.04571327
4	1.56250000	1.59375000	1.62500000	-0.01647214
5	1.56250000	1.57812500	1.59375000	0.01461968
6	1.57812500	1.58593750	1.59375000	-0.00092669
7	1.57812500	1.58203125	1.58593750	0.00684641
8	1.58203125	1.58398438	1.58593750	0.00295984
9	1.58398438	1.58496094	1.58593750	0.00101657
10	1.58496094	1.58544922	1.58593750	0.00004494
11	1.58544922	1.58569336	1.58593750	-0.00044088
12	1.58544922	1.58557129	1.58569336	-0.00019797
13	1.58544922	1.58551025	1.58557129	-0.00007652
14	1.58544922	1.58547974	1.58551025	-0.00001579
15	1.58544922	1.58546448	1.58547974	0.00001457
16	1.58546448	1.58547211	1.58547974	-0.00000061

Try Method

cnt	l	mid	r	f(x)
0	1.00000000	1.58293093	2.00000000	0.00505610
1	1.58293093	1.58551188	2.00000000	-0.00007976
2	1.58293093	1.58547180	1.58551188	-0.00000000

f(x) = x^3 - x - 1

Dichotomy Method

cnt	l	mid	r	f(x)
0	1.00000000	1.50000000	2.00000000	0.87500000
1	1.00000000	1.25000000	1.50000000	-0.29687500
2	1.25000000	1.37500000	1.50000000	0.22460938
3	1.25000000	1.31250000	1.37500000	-0.05151367
4	1.31250000	1.34375000	1.37500000	0.08261108
5	1.31250000	1.32812500	1.34375000	0.01457596
6	1.31250000	1.32031250	1.32812500	-0.01871061
7	1.32031250	1.32421875	1.32812500	-0.00212795
8	1.32421875	1.32617188	1.32812500	0.00620883
9	1.32421875	1.32519531	1.32617188	0.00203665
10	1.32421875	1.32470703	1.32519531	-0.00004659
11	1.32470703	1.32495117	1.32519531	0.00099479
12	1.32470703	1.32482910	1.32495117	0.00047404
13	1.32470703	1.32476807	1.32482910	0.00021371
14	1.32470703	1.32473755	1.32476807	0.00008355
15	1.32470703	1.32472229	1.32473755	0.00001848
16	1.32470703	1.32471466	1.32472229	-0.00001406
17	1.32471466	1.32471848	1.32472229	0.00000221

Try Method

cnt	l	mid	r	f(x)
0	1.00000000	1.16666667	2.00000000	-0.57870370
1	1.16666667	1.25311203	2.00000000	-0.28536303
2	1.25311203	1.29343740	2.00000000	-0.12954209
3	1.29343740	1.31128102	2.00000000	-0.05658849
4	1.31128102	1.31898850	2.00000000	-0.02430375
5	1.31898850	1.32228272	2.00000000	-0.01036185
6	1.32228272	1.32368429	2.00000000	-0.00440395
7	1.32368429	1.32427946	2.00000000	-0.00186926
8	1.32427946	1.32453199	2.00000000	-0.00079296
9	1.32453199	1.32463909	2.00000000	-0.00033630
10	1.32463909	1.32468452	2.00000000	-0.00014261
11	1.32468452	1.32470378	2.00000000	-0.00006047
12	1.32470378	1.32471194	2.00000000	-0.00002564
13	1.32471194	1.32471541	2.00000000	-0.00001087
14	1.32471541	1.32471688	2.00000000	-0.00000461

【2】实验 2 利用简单迭代法和加速迭代法求解非线性方程的根，需要注意的是选取合适的迭代初值以及收敛的迭代方式，实验结果如下：

实验 2.1：（用简单迭代法求解非线性方程 3）

实验 2.2：（分别用简单迭代法与加速迭代法求解非线性方程 4）

实验 2.3：（用简单迭代法与加速迭代法求解非线性方程 5）

```
work1@ws1:~/ChenBodan/class4$ cd "/home/work1/ChenBodan/class4/" && g++ -std=c++11 2-1.cpp -larmadillo -o 2-1 && "/h
ome/work1/ChenBodan/class4/"2-1
简单迭代法
                x(k+1)=1/2*sqrt(10-x(k)^2)
cnt             x(k)            x(k+1)          |x(k)-x(k+1)|
0           1.50000000      1.28695377      0.21304623
1           1.28695377      1.40254080      0.11558704
2           1.40254080      1.34545837      0.05708243
3           1.34545837      1.37517025      0.02971188
4           1.37517025      1.36009419      0.01507606
5           1.36009419      1.36784697      0.00775277
6           1.36784697      1.36388700      0.00395996
7           1.36388700      1.36591673      0.00202973
8           1.36591673      1.36487822      0.00103852
9           1.36487822      1.36541006      0.00053184
10          1.36541006      1.36513782      0.00027224
11          1.36513782      1.36527721      0.00013939
12          1.36527721      1.36520585      0.00007136
13          1.36520585      1.36524238      0.00003653
14          1.36524238      1.36522368      0.00001870
15          1.36522368      1.36523326      0.00000958

                x(k+1)=sqrt(10/x(k)-4x(k))
cnt             x(k)            x(k+1)          |x(k)-x(k+1)|
0           1.50000000      0.81649658      0.68350342
1           0.81649658      2.99690881      2.18041222
2           2.99690881      -nan            nan

                x(k+1)=x(k)-x(k)^3-4x(k)^2+10
cnt             x(k)            x(k+1)          |x(k)-x(k+1)|
0           1.50000000      -0.87500000       2.37500000
1           -0.87500000      6.73242188       7.60742188
2           6.73242188      -469.72001200       476.45243388
3           -469.72001200       102754555.18738511          102755024.90739712
4           102754555.18738511          -1084933870531746352594944.00000000         1084933870531746486812672.00000000
NAN,迭代不收敛

work1@ws1:~/ChenBodan/class4$ cd "/home/work1/ChenBodan/class4/" && g++ -std=c++11 2-2.cpp -larmadillo -o 2-2 && "/h
ome/work1/ChenBodan/class4/"2-2

                x(k+1)=1/2*sqrt(10-x(k)^2)
简单迭代法
cnt             x(k)            x(k+1)          |x(k)-x(k+1)|
0           1.57079633      1.60000000      0.02920367
1           1.60000000      1.57109247      0.02890753
2           1.57109247      1.59970682      0.02861434
3           1.59970682      1.57138260      0.02832421
4           1.57138260      1.59941959      0.02803698
5           1.59941959      1.57166684      0.02775274
6           1.57166684      1.59913819      0.02747135
7           1.59913819      1.57194531      0.02719288
8           1.57194531      1.59886251      0.02691719
9           1.59886251      1.57221813      0.02664437
...
781         1.58547705      1.58546660      0.00001045
782         1.58546660      1.58547695      0.00001035
783         1.58547695      1.58546671      0.00001024
784         1.58546671      1.58547685      0.00001014
785         1.58547685      1.58546681      0.00001004
786         1.58546681      1.58547674      0.00000994

加速迭代法
cnt             x(k)            y(k)            z(k)            x(k+1)          |x(k)-x(k+1)|
0           1.57079633      1.60000000      1.57109247      1.58547258       0.02920367
1           1.58547258      1.58547103      1.58547256      1.58547180       0.00000154
work1@ws1:~/ChenBodan/class4$ cd "/home/work1/ChenBodan/class4/" && g++ -std=c++11 2-3.cpp -larmadillo -o 2-3 && "/h
ome/work1/ChenBodan/class4/"2-3

                x(k+1)=x^3-1
简单迭代法
cnt             x(k)            x(k+1)          |x(k)-x(k+1)|
0           1.50000000      2.37500000      0.87500000
1           2.37500000      12.39648438      10.02148438
2           12.39648438      1904.00277223       1891.60628786
3           1904.00277223       6902441412.88919163         6902439508.88641930
4           6902441412.88919163         3288578303998010084332740280320.00000000        32885783039980100843327402803
2.00000000
NAN,迭代不收敛

加速迭代法
cnt             x(k)            y(k)            z(k)            x(k+1)          |x(k)-x(k+1)|
0           1.50000000      2.37500000      12.39648438      1.41629297       0.87500000
1           1.41629297      1.84092195      5.23887278      1.35565044       0.42462898
2           1.35565044      1.49139828      2.31727068      1.32894878       0.13574783
3           1.32894878      1.34706288      1.44435123      1.32480449       0.01811411
4           1.32480449      1.32517355      1.32711729      1.32471799       0.00036906
5           1.32471799      1.32471815      1.32471898      1.32471796       0.00000016
```

【3】实验三中使用牛顿迭代法以及割线法求解非线性方程，可以看到，牛顿法用到了函数的一阶导数的信息，收敛速度较快，迭代精度较好，实验结果如下：

实验 3.1：（用牛顿迭代法与割线法求解非线性方程 3）

实验 3.2：（分别用牛顿迭代法与割线法求解非线性方程 4）

实验 3.3：（分别用牛顿迭代法与割线法求解非线性方程 5）

实验 3.4：（分别用牛顿迭代法与割线法求解非线性方程 2）

```
work1@ws1:~/ChenBodan/class4$ cd "/home/work1/ChenBodan/class4/" && g++ -std=c++11 3.cpp -larmadillo -o 3 && "/home/
work1/ChenBodan/class4/"3
                    f(x)=x^3+4x^2-10
Newton 迭代法
cnt            x(k)             x(k+1)          |x(k)-x(k+1)|
0           1.5000000         1.3733333           0.1266667
1           1.3733333         1.3652620           0.0080713
2           1.3652620         1.3652300           0.0000320
3           1.3652300         1.3652300           0.0000000

割线法
cnt            x(k)             x(k+1)          x(k+2)            |x(k)-x(k+2)|
0            1.0000000        2.0000000       1.2631579            0.2631579
1            2.0000000        1.2631579       1.3388278            0.6611722
2            1.2631579        1.3388278       1.3666164            0.1034585
3            1.3388278        1.3666164       1.3652119            0.0263841
4            1.3666164        1.3652119       1.3652300            0.0013864
5            1.3652119        1.3652300       1.3652300            0.0000181
6            1.3652300        1.3652300       1.3652300            0.0000000

                    f(x)=1.66+0.99cos(x)
Newton 迭代法
cnt            x(k)             x(k+1)          |x(k)-x(k+1)|
0           1.5707963         1.5854715           0.0146752
1           1.5854715         1.5854718           0.0000003

割线法
cnt            x(k)             x(k+1)          x(k+2)            |x(k)-x(k+2)|
0            1.0000000        2.0000000       1.5829309            0.5829309
1            2.0000000        1.5829309       1.5855119            0.4144881
2            1.5829309        1.5855119       1.5854718            0.0025409
3            1.5855119        1.5854718       1.5854718            0.0000401
4            1.5854718        1.5854718       1.5854718            0.0000000



                    f(x)=x^3-x-1
Newton 迭代法
cnt            x(k)             x(k+1)          |x(k)-x(k+1)|
0           1.5000000         1.3478261           0.1521739
1           1.3478261         1.3252004           0.0226257
2           1.3252004         1.3247182           0.0004822
3           1.3247182         1.3247180           0.0000002

割线法
cnt            x(k)             x(k+1)          x(k+2)            |x(k)-x(k+2)|
0            1.0000000        2.0000000       1.1666667            0.1666667
1            2.0000000        1.1666667       1.2531120            0.7468880
2            1.1666667        1.2531120       1.3372064            0.1705398
3            1.2531120        1.3372064       1.3238501            0.0707381
4            1.3372064        1.3238501       1.3247079            0.0124985
5            1.3238501        1.3247079       1.3247180            0.0008679
6            1.3247079        1.3247180       1.3247180            0.0000100
7            1.3247180        1.3247180       1.3247180            0.0000000

                    f(x)=e^(-10x)-1
Newton 迭代法
cnt            x(k)             x(k+1)          |x(k)-x(k+1)|
0           -0.5100000        -0.4106097          0.0993903
1           -0.4106097        -0.3122569          0.0983528
2           -0.3122569        -0.2166613          0.0955956
3           -0.2166613        -0.1281178          0.0885435
4           -0.1281178        -0.0558888          0.0722290
5           -0.0558888        -0.0130732          0.0428155
6           -0.0130732        -0.0008185          0.0122548
7           -0.0008185        -0.0000033          0.0008152
8           -0.0000033        -0.0000000          0.0000033

割线法
cnt            x(k)             x(k+1)          x(k+2)            |x(k)-x(k+2)|
0            -0.5100000       0.4900000       0.4839484            0.9939484
1            0.4900000        0.4839484       -12.4397359           12.9297359
2            0.4839484        -12.4397359      0.4839484            0.0000000
```

实验分析与总结：

通过本次实验，我进一步熟悉了个人电脑上 C++代码的编写与调试，服务器上的代码编译与运行；熟悉了求解非线性方程的区间逼近法（二分法、试值法），不动点迭代法（简单迭代法、加速迭代法），和牛顿类迭代法（牛顿迭代法、割线法）；了解了以上方法的算法的稳定性与收敛速度特点；熟悉了高阶迭代法在处理特殊病态问题时的收敛性问题，体会二分法作为外部嵌套迭代算法的必要性。

具体来说，实验 1 中 5 个实验分别用二分法和试值法求解一元非线性方程的根，比较二者的求解效率以及稳定性等；可以发现，试值法在大部分情况下效率比二分法要好，但是在某些特殊情况下也可能不收敛；

实验 2 利用简单迭代法和加速迭代法求解非线性方程的根，需要注意的是选取合适的迭代初值以及收敛的迭代方式；

实验三中使用牛顿迭代法以及割线法求解非线性方程，可以看到，牛顿法用到了函数的

一阶导数的信息，收敛速度较快，迭代精度较好。

注：若填写内容较多，可在背面继续填写。