Report5

> 计算方法--第五次程序作业
>
> PB20020480 王润泽

牛顿迭代法

“`hw_newton.m`”是利用牛顿迭代法求根，输出结果包括：初值，根，迭代次数。并且利用最后一次迭代检验二阶收敛性。
“`Newton.m`”是牛顿迭代法的函数模块

结果：

```
牛顿迭代法
初值:0.100000, 根: -0.000000000000000, 迭代次数: 3
初值:0.200000, 根: 0.000000000000000, 迭代次数: 4
初值:0.900000, 根: -1.732050807568877, 迭代次数: 7
初值:9.000000, 根: 1.732050807568877, 迭代次数: 9

取最后一次的迭代结果来验证是否为2阶收敛的
k = 1, xk = 6.075000000000000, ek = 4.342949192431123, c = NULL
k = 2, xk = 4.162795697053038, ek = 2.430744889484161, c = 0.128875357807854
k = 3, xk = 2.945153623683457, ek = 1.213102816114580, c = 0.205314134292457
k = 4, xk = 2.219293671090175, ek = 0.487242863521297, c = 0.331093202872011
k = 5, xk = 1.856453816154091, ek = 0.124403008585214, c = 0.524010403990703
k = 6, xk = 1.743532447579469, ek = 0.011481640010591, c = 0.741894512898189
k = 7, xk = 1.732163235312751, ek = 0.000112427743874, c = 0.852836233413889
k = 8, xk = 1.732050818513779, ek = 0.000000010944901, c = 0.865894265361425
```

分析收敛阶

如程序输出结果所示，我选择了**初值为9**的迭代序列，因为这个迭代次数比较多，可以更加清楚的看到收敛的结果。

我先预设了收敛阶次数为2，然后利用

```
 fprintf("k = %d, xk = %.15f, ek = %.15f, c = %.15f\n",ii, x1, error, error/e^2);%c是渐近误差常数
```

来输出渐近误差常数的结果，最后可以看到，渐近误差常数明显在收敛，收敛值大致为 **0.86**左右

弦截迭代法

“`hw_tang.m`”是利用弦截迭代法求根，输出结果包括：初值，根，迭代次数。并且利用最后一次迭代检验收敛性满足$(1+\sqrt{5})/2$
“`Tangent.m`”是弦截迭代法的函数模块

结果：

```
弦截法
初值: x0=-0.100000, x1=0.100000, 根: 0.000000000000000, 迭代次数: 1
初值: x0=-0.200000, x1=0.200000, 根: 0.000000000000000, 迭代次数: 1
初值: x0=-2.000000, x1=0.900000, 根: 1.732050807568877, 迭代次数: 12
初值: x0=0.900000, x1=9.000000, 根: 1.732050807568877, 迭代次数: 14

这里取最后一次迭代，验证弦截法迭代收敛阶是否为(1+sqrt(5))/2
k = 1, xk = 0.922678633068692, ek = 8.077321366931308, C = NULL
k = 2, xk = 0.945425651622759, ek = 0.022747018554067, C = 0.000774345988665
k = 3, xk = -4.260255841874963, ek = 5.205681493497723, C = 2371.535483065387325
k = 4, xk = 1.111142329459913, ek = 5.371398171334876, C = 0.372222181489314
k = 5, xk = 1.279507739861510, ek = 0.168365410401598, C = 0.011090387765539
k = 6, xk = 2.627632903195471, ek = 1.348125163333961, C = 24.081268764321550
k = 7, xk = 1.475358392278638, ek = 1.152274510916833, C = 0.710637396087242
k = 8, xk = 1.597341561251019, ek = 0.121983168972381, C = 0.096984042848802
k = 9, xk = 1.772723874716548, ek = 0.175382313465529, C = 5.276957459490521
k = 10, xk = 1.726991772050257, ek = 0.045732102666291, C = 0.764676687530142
k = 11, xk = 1.731876796901183, ek = 0.004885024850926, C = 0.718909896388291
k = 12, xk = 1.732051572649817, ek = 0.000174775748634, C = 0.959341601223050
k = 13, xk = 1.732050807453568, ek = 0.000000765196249, C = 0.919530229430167
k = 14, xk = 1.732050807568877, ek = 0.000000000115309, C = 0.908087801932059
```

分析收敛阶

如程序输出结果所示，我选择了**初值为x0=0.9, x1=9**的迭代序列，因为这个迭代次数比较多，可以更加清楚的看到收敛的结果。

我先预设了收敛阶次数为 $(1+\sqrt{5})/2$ ，然后利用

```
a = (1+sqrt(5))/2;

fprintf("k = %d, xk = %.15f, ek = %.15f, C = %.15f\n",k, x2, error,
error/e^(a));%C是渐近误差常数
```

来输出渐近误差常数的结果，最后可以看到，渐近误差常数明显在收敛，收敛值大致为 **0.91**左右