Finalwork：一维非线性方程的求解

Janice_zh　信息与计算科学 3200103473

July 5, 2022

摘要

本文介绍求解一维非线性方程 $f(x) = 0$ 的算法实现和技术要点。总体来说可将求解方法分为 “root bracketing” 和 “root polishing” 两类。我们将详细介绍二分法、牛顿法和 Brent 法的算法实现，并对其收敛性和收敛速度进行比较。然后我们将对不同的函数进行算例测试，探究求解一维非线性方程的技术要点。最后得出结论。[1]

1　引言

对于一维低阶线性方程，我们易由公式法或其他方法很快得出方程的根。然而当阶数较高时，很难直接利用公式法求解，因此我们考虑利用迭代法实现对一维非线性方程的求解。

2　数学理论与技术要点

目前求解一维非线性方程的方法可分为两类：第一种不断对解的可能区间进行迭代，称为 “root bracketing”，如二分法，Brent 法；第二种不断对一个近似解进行迭代，称为 “root poshing”，如牛顿法。

不同的迭代方式在迭代算法实现上有所不同，但总体的迭代步骤一致，如下所示：

· 对于算法T,初始化状态s
· 利用T对s进行迭代
· 判断s的收敛性，若未达到要求则重复迭代过程

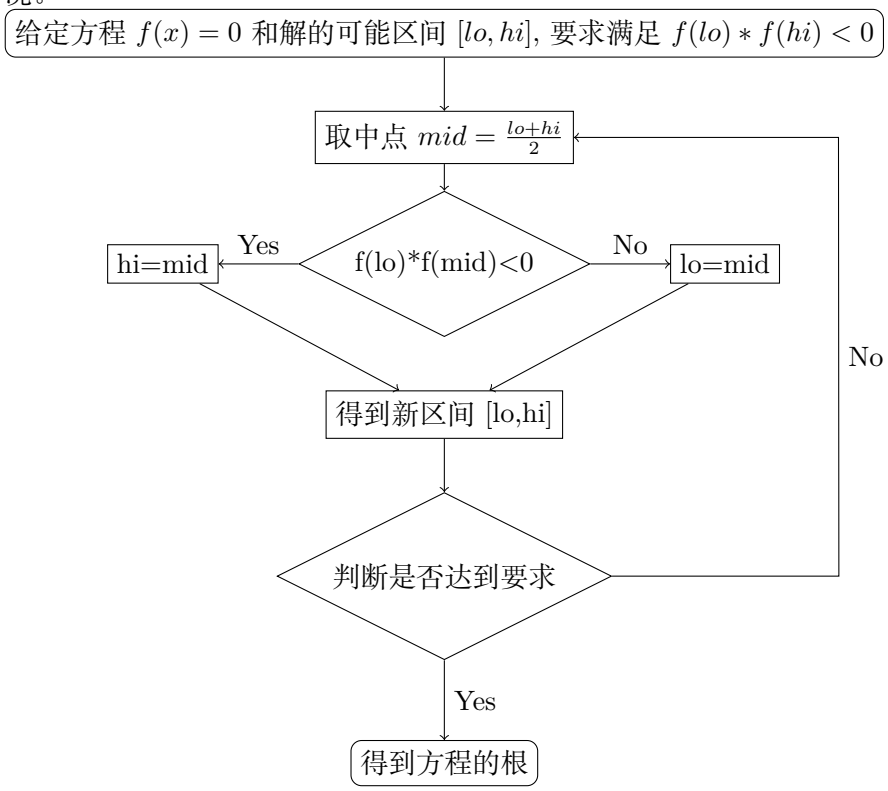
不同的迭代方法对于初始状态的要求不同，但总体来说，初始可能区间、初始近似解、最大迭代次数、精度要求以及函数本身的性质都会影响迭代的效果。因此，对于一个给定的方程，可以利用图像等对方程的根进行大致预估，选取合适的解区间、近似解、求解方法以及上述的其他因素就成为了求解一维非线性方程的技术要点。

3 算法实现与收敛性比较

3.1 算法实现

3.1.1 二分法

给定可能解区间，每次取中点二分进行迭代。是最简单的一种迭代方式，线性收敛。同时要求给定区间端点的函数值异号，且不能处理偶数重根的情况。



3.1.2 牛顿法

不断迭代近似解，切线与 x 轴的交点迭代成为新的近似解。单根二次收敛，复根线性收敛。求解效率与准确度比较依赖初始近似解的选取。(篇幅原因流程图在此不再赘述)

3.1.3 secant 法

牛顿法的简化，从第二步开始用数值估计替代导数。多重根线性收敛。适合在根附近导数无明显变化的函数方程，依赖原方程的特点与初始近似解的选取。(篇幅原因无流程图)

3.2 收敛性比较

分别用二分法，牛顿法，secant 法求解 $x^2 - 5 = 0$。[2]

```
using bisection method
 iter [    lower,     upper]      root        err  err(est)
    1 [0.0000000, 2.5000000] 1.2500000 -0.9860680 2.5000000
    2 [1.2500000, 2.5000000] 1.8750000 -0.3610680 1.2500000
    3 [1.8750000, 2.5000000] 2.1875000 -0.0485680 0.6250000
    4 [2.1875000, 2.5000000] 2.3437500 +0.1076820 0.3125000
    5 [2.1875000, 2.3437500] 2.2656250 +0.0295570 0.1562500
    6 [2.1875000, 2.2656250] 2.2265625 -0.0095055 0.0781250
    7 [2.2265625, 2.2656250] 2.2460938 +0.0100258 0.0390625
    8 [2.2265625, 2.2460938] 2.2363281 +0.0002601 0.0195312
    9 [2.2265625, 2.2363281] 2.2314453 -0.0046227 0.0097656
   10 [2.2314453, 2.2363281] 2.2338867 -0.0021813 0.0048828
   11 [2.2338867, 2.2363281] 2.2351074 -0.0009606 0.0024414
Converged:
   12 [2.2351074, 2.2363281] 2.2357178 -0.0003502 0.0012207
```

```
using newton method
iter        root        err   err(est)
    1  3.0000000 +0.7639320 -2.0000000
    2  2.3333333 +0.0972654 -0.6666667
    3  2.2380952 +0.0020273 -0.0952381
Converged:
    4  2.2360689 +0.0000009 -0.0020263
```

```
using secant method
iter         root          err    err(est)
    1  3.0000000 +0.7639320 -2.0000000
    2  2.5000000 +0.2639320 -0.5000000
    3  2.2727273 +0.0366593 -0.2272727
    4  2.2380952 +0.0020273 -0.0346320
Converged:
    5  2.2360845 +0.0000165 -0.0020108
```

由图像可得，牛顿法与 secant 法相对于二分法，收敛性更强，收敛速度更快，所得解精度更高。

4 数值算例

下用牛顿法实现不同式子的求解。求解时应选取合适的初始近似解。

Example1: 求解 $2x^2 + 4x - 3 = 0$

```
using newton method
iter         root          err    err(est)
    1  2.2083333 -0.0277346 -2.7916667
    2  0.9937771 -1.2422909 -1.2145563
    3  0.6238393 -1.6122287 -0.3699378
    4  0.5817003 -1.6543677 -0.0421390
Converged:
    5  0.5811389 -1.6549290 -0.0005613
```

Figure 1: $2x^2 + 4x - 3 = 0$

Example2: 求解 $x^2 + 5 = 0$
选取不当的初始估计解会造成无法求解：

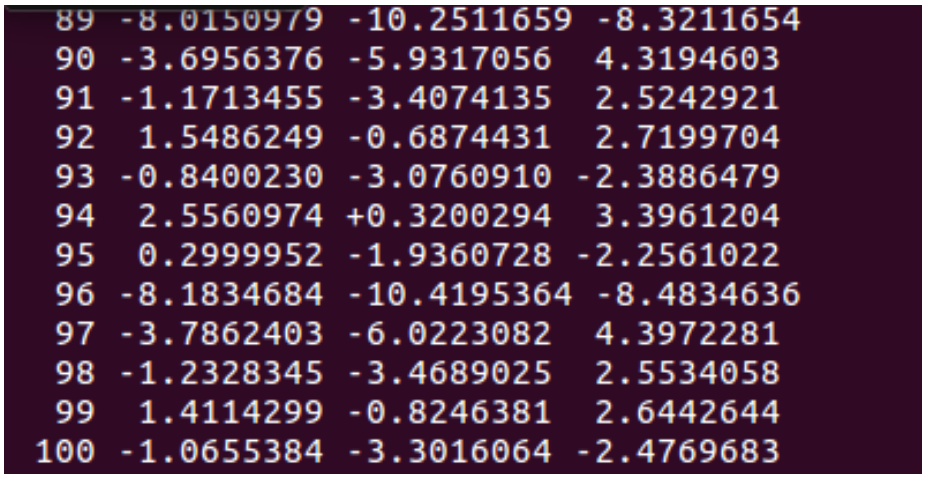


Figure 2: $x^2 + 5 = 0$

5 结论

利用二分法、牛顿法、Brent 法等我们可以实现对一维非线性方程的求解。在求解过程中需主意选取合适的解区间或初始近似解。同时，给定同一方程时，不同的方法对应不同的迭代速度，我们可以结合给定方程的性质，选取合适的方法进行求解，从而获得更高的效率与更精确的结果。

References

[1] Richard, P., and Brent, "An algorithm with guaranteed convergence for finding a zero of a function," *Comput. J.*, vol. 14, 1971.

[2] xxxjrr, "二分法求方程近似解," https://blog.csdn.net/xxxjrr/article/details/120210859, 2022.7.4.