最终项目作业: 一维非线性方程的求根

褚朱钇恒
信息与计算科学 3200104144

2022 年 7 月 4 日

摘要

一维非线性方程 $f(x)=0$ 不一定有精确的代数解，但可以求出数值解。本文主要介绍了二分法和牛顿法两种求解一维非线性方程方法，并比较两者在求解不同方程的收敛性和收敛速度。

1 引言

求解一维非线性方程

$$f(x)=0; \tag{1}$$

即确定 $x^*\in[a,b]$ 使 $f(x^*)=0$，其中 f 是定义在 [a,b] 上的连续非线性函数，满足 $f(x^*)=0$ 的 x^* 成为方程 (1) 的根，也称为 $f(x)$ 的零点。

2 数学理论

定理 1 *介值定理 如果 $f(x)$ 在区间 $[a,b]$ 上连续且 $f(a)f(b)<0$，则至少存在一点 $\eta\in[a,b]$ 使 $f(\eta)=0$*

定理 2 *泰勒公式 把 $f(x)$ 在 x_0 点附近展开成 Taylor 级数，得到*

$$f(x)=f(x_0)+(x-x_0)f^{'}(x_0)+(x-x_0)^2f^{''}(x_0)+\cdots$$

3 问题分析

3.1 二分法

由定理1可以得出二分法求解一维非线性方程的想法，即每次将区间中点视为近似根，如果误差超过了要求，则考虑将原左右端点中与中点的 f 值异号的点作为新的端点，然后在子区间上继续迭代。

3.2 Newton 法

由定理2，取其线性部分则有 $f(x_0)-f^{'}x_0(x-x_0)=0$，其解为 $x_1=x_0-\frac{f(x_0)}{f^{'}(x_0)}$，故得到迭代序列 $x_{n+1}=x_n-\frac{f(x_n)}{f^{'}(x_n)}$

4 算法

根据上一节的分析，我们不难得出二分法和 Newton 法的具体流程。
判断停止迭代的条件有：

- 已找到符合精度要求的根
- 以达到给定的最大迭代次数
- 出现错误

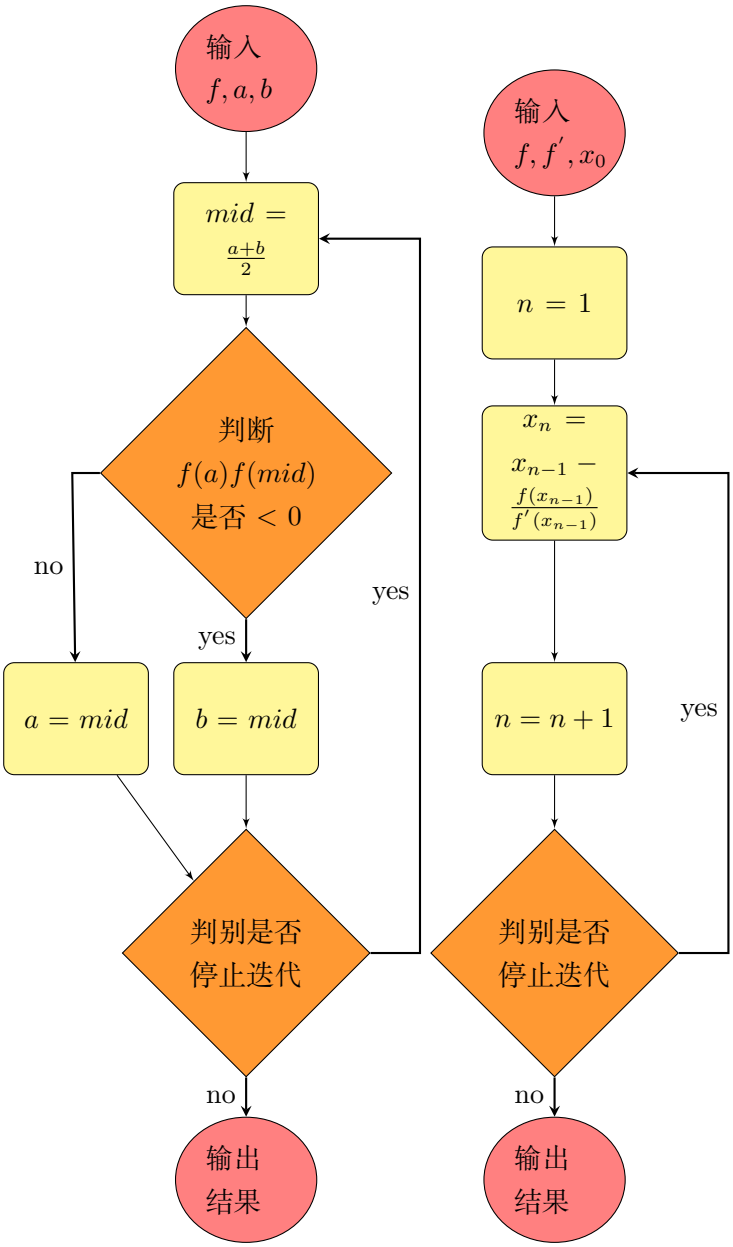

图 1: 二分法和 Newton 法算法流程图

5 数值算例和收敛性测试

为了方便随机生成和求导,我们只考虑形如 $f(x) = ax^5+bx^4+cx^3+dx^2+ex+f+si*\sin(x)+co*\cos(x) = 0$ 的方程。

以 $f(x) = x^5 - x^4 + x^3 - x^2 + x - 1 + 0.5 * \sin(x) + 0.5 * \cos(x) = 0$ 为例，分别使用牛顿法和二分法得到的结果为：`iter`表示迭代次数，`root`表示当前迭代得到的解，`err`表示当前的解代入 $f(x)$ 的值，`err(est)`表示当前迭代的解的可能区间的大小。

```
using newton method
iter        root        err        del
    1  0.3333333 +0.1128954  0.3333333
    2  0.4603065 +0.0082044  0.1269731
    3  0.4708896 +0.0000265  0.0105832
Converged:
    4  0.4709241 +0.0000000  0.0000344


using bisection method
 iter [    lower,     upper]      root        err  err(est)
    1 [-4.5727181, 7.5818954] 1.5045887 +4.7647356 12.1546134
    2 [-4.5727181, 1.5045887] -1.5340647 -23.0132715 6.0773067
    3 [-1.5340647, 1.5045887] -0.0147380 -0.5223815 3.0386534
    4 [-0.0147380, 1.5045887] 0.7449253 +0.2313636 1.5193267
    5 [-0.0147380, 0.7449253] 0.3650936 -0.0852521 0.7596633
    6 [0.3650936, 0.7449253] 0.5550095 +0.0641367 0.3798317
    7 [0.3650936, 0.5550095] 0.4600516 -0.0084020 0.1899158
    8 [0.4600516, 0.5550095] 0.5075305 +0.0279847 0.0949579
    9 [0.4600516, 0.5075305] 0.4837910 +0.0098801 0.0474790
   10 [0.4600516, 0.4837910] 0.4719213 +0.0007680 0.0237395
   11 [0.4600516, 0.4719213] 0.4659864 -0.0038089 0.0118697
   12 [0.4659864, 0.4719213] 0.4689539 -0.0015185 0.0059349
   13 [0.4689539, 0.4719213] 0.4704376 -0.0003748 0.0029674
   14 [0.4704376, 0.4719213] 0.4711794 +0.0001967 0.0014837
   15 [0.4704376, 0.4711794] 0.4708085 -0.0000890 0.0007419
Converged:
   16 [0.4708085, 0.4711794] 0.4709940 +0.0000539 0.0003709
```

在此算例中，两种方法都得到了一个近似于 0.4708 的解，根据此算例，容易得出猜想：Newton 法的收敛速度较快，二分法每次只将解的可能区间减半，获得相同精度的解需要的迭代次数较多。为了进一步验证此猜想，我生成了 100000 个一维非线性方程，分别使用 gsl 库提供的 Newton 法和二分法求一个解 [1]，记录成功求解的次数和平均迭代次数，再对比每次迭代后未收敛到最终解的方程占全部方程的比例，结果如下。

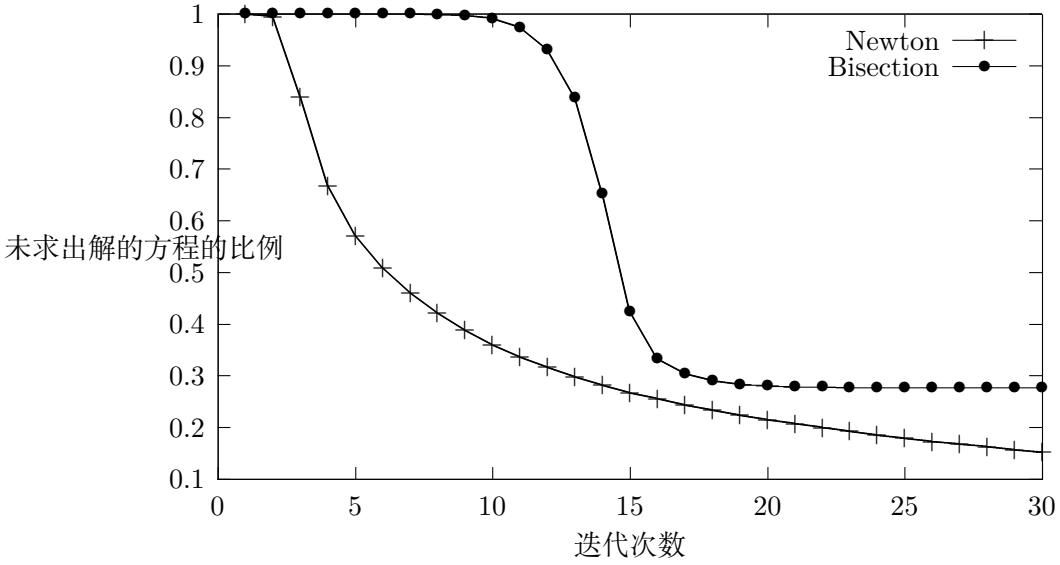

图 2: 两种迭代方式的收敛速度

这 100000 个方程中，牛顿法在给定的迭代次数内得到一个解的方程有 84766 个，平均迭代次数 8.190147；二分法在在给定的初始范围选取次数内得到一个解的可能区间并得出最终解的方程有 72312 个，平均迭代次数 14.506403。

6 分析与结论

根据基于随机生成的方程的测试结果，我们发现牛顿法收敛到一个解的概率高于二分法，由于测试基于的方程数据集相同，排除相同数量的无解方程后，我们仍可以得出结论：牛顿法的收敛性强于二分法；同理，根据图 (5) 和平均迭代次数可得 Newton 法的收敛速度快于二分法。

从图中也可以看出，二分法的稳定性更强，在找到合法的初始区间后，能在有限次迭代后得到符合精度要求的解。同时，二分法避免了对原方程求微分，更适合用于某些难以求导的函数求零点。故二分法也有其可取性。

参考文献

[1] GSL. One dimensional root-finding. `https://www.gnu.org/software/gsl/doc/html/roots.html`.