Chap2 非线性方程求根

李晨昊 2017011466

2019-3-4

目录

1 运行代码

环境要求：rust，gnuplot

运行代码

```
cargo run
```

2 q2.2

2.1 任务

实现阻尼牛顿法，利用其求解两个方程

2.2 解题思路

按照课本上描述的算法的思路编写代码即可，编写时无需显式地维护 i 和 k；也无需使用 x 数组，因为只需要最近的两个 x

我选择了初始的 $\lambda_0 = 0.95$，在 $|f(x)| < 10^{-10}$ 时退出

除了阻尼牛顿法之外，我还实现了普通的牛顿法，以比较二者的收敛速度

2.3 实验结果

输出如下

```
solving 3x^3-x-1=0, x0=0.6
damped newton method
iter0: lambda=0.01484375, x=1.113593750000001
iter1: lambda unused, x=1.3829182207538546
iter2: lambda unused, x=1.3276427364127026
iter3: lambda unused, x=1.3247258973156648
iter4: lambda unused, x=1.324717957303496
iter5: lambda unused, x=1.324717957244746

newton method
iter0: x=17.900000000000034
iter1: x=11.946802328608793
iter2: x=7.985520351936227
iter3: x=5.3569093147954705
iter4: x=3.624996032946104
```

```
iter5: x=2.5055891901066367
iter6: x=1.820129422319472
iter7: x=1.4610441098876834
iter8: x=1.339323224262526
iter9: x=1.324912867718656
iter10: x=1.3247179926378148
iter11: x=1.3247179572447472

fzero says x=1.324717957244774

solving -x^3+5x=0, x0=1.35
damped newton method
iter0: lambda=0.059375, x=2.439610628342243
iter1: lambda unused, x=2.2589992391657847
iter2: lambda unused, x=2.2364124819905435
iter3: lambda unused, x=2.2360680570863867
iter4: lambda unused, x=2.236067977499794

newton method
iter0: x=10.525668449197836
iter1: x=7.124286625588786
iter2: x=4.9107806530193825
iter3: x=3.5169113058921715
iter4: x=2.7097430061997922
iter5: x=2.336940031468776
iter6: x=2.2422442539928538
iter7: x=2.236093403021945
iter8: x=2.236067977933435
iter9: x=2.23606797749979

fzero says x=2.236067977498973
```

二者比较发现，阻尼牛顿法的确可以达到比普通的牛顿法更快的收敛速度，主要原因是前几步迭代时可以通过阻尼因子的调节避免“走过头”，而后面几步其实没有用到提供的 $\{\lambda_i\}$ 序列，直接取了系数 1。这基本符合课本上的说法：阻尼牛顿法解决了牛顿法当初始值 x_0 偏离准确解 x^* 较远时可能发散的问题。

2.4 心得体会

经过实验，我认为阻尼牛顿法实现简单，效果较好，是一种很适合平时使用的方程求解方法。

3 q1.3

3.1 任务

利用 fzerotx 求解方法来求解 $J_0(x) = besselj(0, x)$ 的前十个正零点，并作图

3.2 解题思路

首先需要将课本上的 matlab 代码翻译为 rust 代码…

$besselj$ 函数由 rust 函数库 special-fun 提供了，因此不需要再重复实现一次

绘制出 $J_0(x)$ 函数的图像可以发现前十个正零点都 < 50，且任何两个零点之间的距离都 > 1，因此为了确定求根的范围，可以逐一检查 $signum(J_0(x))$ 是否等于 $signum(J_0(x+1))$，二者不等时即证明 $(x, x+1)$ 上有唯一零点

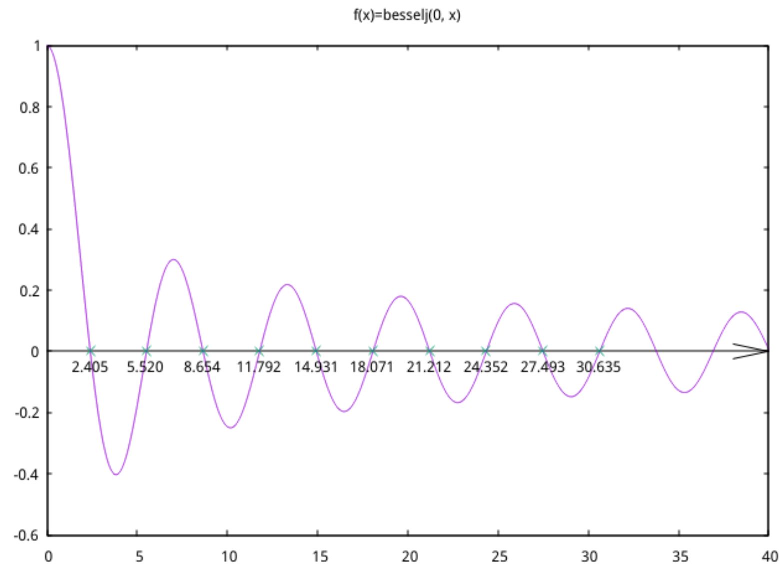
选择精度 $\epsilon = 10^{-15}$

3.3 实验结果

输出为

```
# roots
2.404825557695773, 5.520078110286311, 8.65372791291101,
11.79153443901429, 14.930917708487785, 18.071063967910924,
21.211636629879276, 24.352471530749302, 27.49347913204025,
30.634606468431972
# values at roots
-0.00000000000000009586882554916807, -0.00000000000000001767335334625920,
0.000000000000000039428723571546593, 0.00000000000000002212475072100211,
0.000000000000000031521707773446503, 0.00000000000000002018692387036633,
-0.000000000000000028432841916769668, -0.000000000000000028692898951593197,
0.0000000000000000917693595287399, -0.00000000000000005581595433928796
```

图形为



3.4 心得体会

经过实验，我认为 fzerotx 求解方法效果较好，且不要求求函数的导数值，虽然实现比较复杂，但是是一种很适合写成库函数的方程求解方法。