清華大學
Tsinghua University

数值分析实验报告

实验二

姓名	王琛
学号	2016011360
班级	计 65
实验日期	2019 年 3 月 20 日
报告日期	2019 年 3 月 20 日

1 第二章第 2 题

问题描述

编程实现阻尼牛顿法。要求:(a) 设定阻尼因子的初始值 λ_0 及解的误差阈值 ϵ; (b) 阻尼因子 λ 用逐次折半法更新; (c) 打印每个迭代步的最终 λ 值及近似解。用所编程序求解:

(1) $x^3 - x - 1 = 0$，取 $x_0 = 0.6$;
(2) $-x^3 + 5x = 0$，取 $x_0 = 1.2$.

解题思路

通过引入一个阻尼因子缩小解的改变量，改进牛顿法当初始值偏离准确解较远时发散的情况，加快收敛的速度。迭代公式为:$x_{k+1} = x_k - \lambda \frac{f(x_k}{f'(x_k)}$。为了保证超线性的收敛速度，阻尼因子的值尽量接近 1。按照书上的伪代码不难写出 matlab 程序。λ 采用每次折半的方法进行更新，初始值为 1.0。值得注意的是，由于本课程都是数值计算，为了避免使用 syms 命令，f(x) 的导数需要自己算好，当作参数传入到牛顿法中。

实验结果

实验结果：由于比较直观，一般牛顿法和阻尼牛顿法的代码不再贴出。(分别是 Newton.m 和 damped_Newton.m)。solve_Newton 对两种方法进行了调用。并且还调用了 fzero 函数进行验证。

solve_Newton.m

```
f=@(x) x^3-x-1;
df=@(x) 3*x^2-1;
delta1=1e-7;
delta2=1e-7;
x0=0.6;
```

```
lambda0=0.5;
fprintf("f(x)=x^3-x-1\n");
damped_Newton(f,df,x0,delta1,delta2,lambda0);
Newton(f,df,x0,delta1,delta2);
fprintf("The answer using fzero=%f\n", fzero(f, x0));

f=@(x) -x^3+5*x;
df=@(x) -3*x^2+5;
x0=1.35;
fprintf("f(x)=-3x^2+5x\n");
damped_Newton(f,df,x0,delta1,delta2,lambda0);
Newton(f,df,x0,delta1,delta2);
fprintf("The answer using fzero=%f\n", fzero(f, x0));
```

得到的结果如图所示，

```
f(x)=x^3-x-1
Using Damped Newton method. Initial lambda=1, x=0.600000
The 0th iteration: lambda=0.01562500, x=1.14062500
The 1th iteration: lambda=1.00000000, x=1.36681366
The 2th iteration: lambda=1.00000000, x=1.32627980
The 3th iteration: lambda=1.00000000, x=1.32472023
The 4th iteration: lambda=1.00000000, x=1.32471796
The 5th iteration: lambda=1.00000000, x=1.32471796
The total iterations=6, and the answer=1.32471796
Using basic Newton Method, initial x=0.600000
The 1th iteration: x=17.90000000
The 2th iteration: x=11.94680233
The 3th iteration: x=7.98552035
The 4th iteration: x=5.35690931
The 5th iteration: x=3.62499603
The 6th iteration: x=2.50558919
The 7th iteration: x=1.82012942
The 8th iteration: x=1.46104411
The 9th iteration: x=1.33932322
The 10th iteration: x=1.32491287
The 11th iteration: x=1.32471799
The 12th iteration: x=1.32471796
The total iterations=12, and the answer=1.324718
The answer using fzero=1.324718
```

```
f(x)=-3x^2+5x
Using Damped Newton method. Initial lambda=1, x=1.35000
The 0th iteration: lambda=0.06250000, x=2.49695856
The 1th iteration: lambda=1.00000000, x=2.27197621
The 2th iteration: lambda=1.00000000, x=2.23690171
The 3th iteration: lambda=1.00000000, x=2.23606844
The 4th iteration: lambda=1.00000000, x=2.23606798
The 5th iteration: lambda=1.00000000, x=2.23606798
The total iterations=6, and the answer=2.23606798
Using basic Newton Method, initial x=1.350000
The 1th iteration: x=10.52566845
The 2th iteration: x=7.12428663
The 3th iteration: x=4.91078065
The 4th iteration: x=3.51691131
The 5th iteration: x=2.70974301
The 6th iteration: x=2.33694003
The 7th iteration: x=2.24224425
The 8th iteration: x=2.23609340
The 9th iteration: x=2.23606798
The 10th iteration: x=2.23606798
The total iterations=10, and the answer=2.236068
The answer using fzero=2.236068
```

实验结论

首先无论是基本的牛顿法还是阻尼牛顿法，求解结果都是十分准确的，至少在小数点后 6 位都和 fzero 得出的结果保持一致。改变阻尼因子的大小，发现其越小收敛速度越慢。

阻尼牛顿法引入的原因是当初始值 x_0 偏离解较远时，牛顿法可能会发散。因此增加单调性要求，使得 $|f(x_{k+1})| < |f(x_k)|$。当迭代解充分靠近准确解时，则不再要阻尼因子的调节。由于引入了阻尼因子，不仅避免了发散，也增加了收敛的速度。从实验结果中可以看出，使用阻尼牛顿法，两个方程都迭代了 6 次，而使用一般的牛顿法，则分别迭代了 13 次和 11 次，是两倍的关系。

另外，从结果中可以看出，两个方程都只有在第一次迭代时 λ 的值发生了变化，后面的迭代情况都比较理想。理论上，阻尼牛顿法也会出现情况很糟糕的时候。

实验心得

这次实验是对书上的伪代码进行实现，不算太难。我主要了解了阻尼牛顿法的优点。建议可以增加一个该方法不好的例子。

2 第二章第 3 题

问题描述

利用 2.6.3 节给出的 fzerotx 程序，在 MATLAB 中编程求第一类的零阶贝塞尔函数 $J''(x)$ 的零点，$J''(x)$ 在 MATLAB 中通过命令 besselj(0,x) 得到。试求 $J''(x)$ 的前 10 个正的零点，并绘出函数曲线和零点的位置。

解题思路

根据 fzerotx 的定义，需要传入的参数有 $J_0(x)$ 以及每个零点的所在区间。根据 $J_(0)(x)$ 的性质，第 i 个零点所在区间为 $[(i-1)\pi, i\pi]$。再调用 fzerotx 就能够得到想要的图像。

实验结果

代码如下：

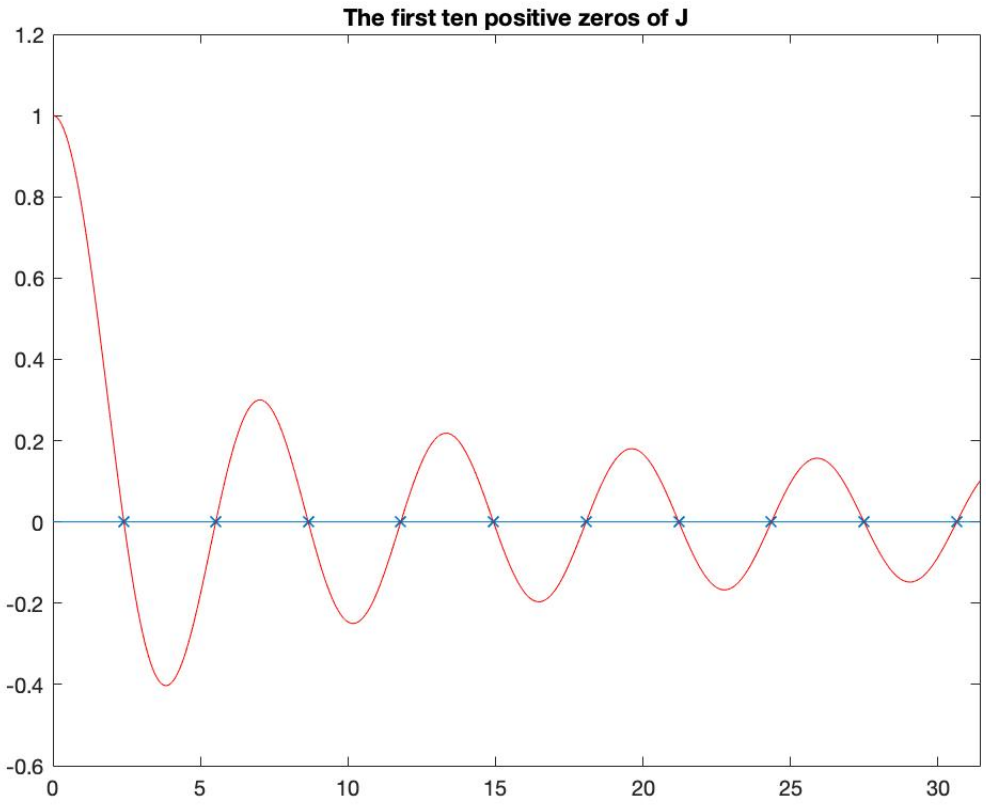
solve_besselj.m

```
J0 = @(x) besselj(0,x);
z=zeros(1,10);
for n = 1:10
    z(n) = fzerotx(J0,[(n-1),n]*pi);
end

disp(z)
x = 0:pi/100:10*pi;

plot(z,zeros(1,10),'x',x,J0(x),'red')
title("The first ten positive zeros of J")
line([0 10*pi],[0 0])
axis([0 10*pi -0.6 1.2])
```

得到的十个零点为：

```
2.4048    5.5201    8.6537   11.7915   14.9309   18.0711   21.2116   24.3525   27.4935   30.6346
```

以及它们在图上的位置：



实验结论

这次实验主要是调用 fzerotx 函数，本身并不困难。但是 zeroin 算法的理解需要花费一定功夫，而且从源码来看，其实现也相对比较复杂。最终求得的零点应当与理论十分契合，曲线相当优美。