数值分析实验报告

张钰晖
2015011372, yuhui-zh15@mails.tsinghua.edu.cn, 185-3888-2881

2017 年 6 月 15 日

1 题目描述

2-2 编程实现阻尼牛顿法. 要求：①设定阻尼因子的初始值 λ_0 及解的误差阈值 ϵ；②阻尼因子 λ 用逐次折半法更新；③打印每个迭代步的最终 λ 值及初始解. 用所编程序求解：

(1) $x^3 - x - 1 = 0$, 取 $x_0 = 0.6$.

(2) $-x^3 + 5x = 0$, 取 $x_0 = 1.2$.

2 解题思路

本道题目要求我们使用阻尼牛顿法完成非线性方程组的求解，根据阻尼牛顿法的伪代码，来实现本算法，阻尼引子 λ 用折半法更新。

3 实验结果

设置阻尼因子的初始值 $\lambda_0 = 1.0$, 解的误差阈值 $\epsilon = 10 * -3$。算法运行结果如下：

(1) function: $x^3 - x - 1$, derivative function: $3 * x^2 - 1$

$iteration = 1, \lambda = 0.03125, x = 1.140625$

$iteration = 2, \lambda = 1, x = 1.366814$

$iteration = 3, \lambda = 1, x = 1.326280$

$iteration = 4, \lambda = 1, x = 1.324720$

$iteration = 5, \lambda = 1, x = 1.324718$

$result : x = 1.324718$

最终结果为 $x = 1.324718$。

(2) function: $-x^3+5*x$, derivative function: $-3*x^2+5$

$iteration=1,\lambda=0.5,x=-1.941176$

$iteration=2,\lambda=1,x=-2.320462$

$iteration=3,\lambda=1,x=-2.240459$

$iteration=4,\lambda=1,x=-2.236081$

$iteration=5,\lambda=1,x=-2.236068$

$result:x=-2.236068$

最终结果为 $x=-2.236068$。

4 实验心得

通过这次实验，我重新复习了牛顿法和阻尼牛顿法的原理，更加深入的理解了引入阻尼因子的必要性，也编写了阻尼牛顿法的程序。

通过调整阻尼因子的初值和误差阈值，我发现阻尼牛顿法有很好的自适应性，很好的解决了牛顿法中可能出现的不收敛等问题。初始阻尼因子应取 1 附近的常数，以便能达到较快的收敛速度。

阻尼牛顿法的收敛速度也很快，根据理论至少是二阶收敛的。由于所需求解方程的特殊性，这里的阻尼牛顿法的结果均符合我们所需的解。

然而，阻尼牛顿法依旧有不足，比如难以确定的迭代初值，只能求解出一个根，以及条件数很大时结束条件不能很好地确定。此时，我们可以考虑使用同伦算法进行求解。

5 源代码

```
% 阻尼牛顿法，输入函数f、导数df、迭代初始值x0、阻尼初始值lambda0、迭代误差限epsilon，输出结果x
function [x] = damped_newton(f, df, x0, lambda0, epsilon)
last_x = inf;
x = x0;
k = 0;
% 当不满足终止条件
while (abs(f(x)) > epsilon || abs(x - last_x) > epsilon)
   last_x = x;
   s = f(x) / df(x);
   x = last_x - s;
   lambda = lambda0;
   % 折半更新lambda
   while abs(f(x)) >= abs(f(last_x))
      lambda = lambda / 2;
      x = last_x - lambda * s;
```

```
    end
    % 迭代次数加1，打印结果
    k = k + 1;
    fprintf('iteration = %i, lambda = %.4e, x = %.6e\n', k, lambda, x);
end
```

```
% 函数和导函数
f = @(x) x^3-x-1;
df = @(x) 3*x^2-1;
fprintf('function: %s, derivative function: %s\n', func2str(f), func2str(df));
% 阻尼牛顿法
res = damped_newton(f, df, 0.6, 1.0, 1e-3);
fprintf('result x = %.6e\n', res);

% 函数和导函数
f = @(x) -x^3+5*x;
df = @(x) -3*x^2+5;
fprintf('function: %s, derivative function: %s\n', func2str(f), func2str(df));
% 阻尼牛顿法
res = damped_newton(f, df, 1.2, 1.0, 1e-3);
fprintf('result x = %.6e\n', res);
```

参考文献

[1] 数值分析与算法（第二版）, 喻文健, 2015.