非线性方程求根上机实习报告

姓名:刘群 学号:2014211591 院系: 地学中心
Email: liu-q14@mails.tsinghua.edu.cn
(本文档由LaTeX编写)

November 15, 2014

1 问题的描述

试用以下方法求方程:

$$x^3 + 2x^2 + 10x - 20 = 0 \tag{1}$$

在$x_0 = 1$附近的根(准确解为$x^* = 1.368808107\cdots$),要求精度达到10^{-7}:

(1) $x_{k+1} = \frac{20-2x_k^2-x_k^3}{10}$;

(2) $x_{k+1} = \sqrt[3]{20 - 10x_k - 2x_k^2}$;

(3) 方法(1)中的Steffensen加速法;

(4) 方法(2)中的Steffensen加速法;

(5) Newton法;

(6) 构造一种三阶收敛的格式.

2 非线性方程的不动点迭代法的相关原理

我们在求解非线性方程组时可以用不动点迭代的方法进行求解. 具体来说我们要求方程$f(x) = 0$的根, 我们可以将原方程变形为$x = \varphi(x)$, 其中x 就叫做$\varphi(x)$的不动点. 从而我们可以构造如下迭代格式:

$$x_{k+1} = \varphi(x_k) \tag{2}$$

当然选择不同的$\varphi(x)$, 可能会产生不同的收敛性和收敛速度. 因此如何选择迭代函数是很重要的.

3 问题(1)和(2)中的迭代算法

3.1 问题(1)中的迭代法

该迭代法是将方程(1)等价变形为

$$x = \frac{20 - 2x^2 - x^3}{10} = \varphi_1(x) \tag{3}$$

我们记方程(1)在$x_0 = 1$附近的根为α, 由题目可知, $\alpha = 1.368808107\cdots$.

对于这个迭代格式, 我们编写了iteration1.m 函数进行求解,我们先按照问题(1)中的迭代公式进行迭代. 如果前后两次迭代产生的误差小于10^{-7} , 我们就停止迭代过程. 但是为了防止出现不收敛的情况, 我们在中间预设了一个迭代步数$N = 1000$, 即当迭代超过1000 步还没有得出结果的话, 迭代过程就会退出. 结果发现, 运行1000 步之后迭代仍然没有收敛, 因此可以说这种方法是不收敛的. 我们可以看一下前10 次迭代的结果, 如Table 1 所示, 可以看出, 问题(1)中给出的迭代法不收敛.

Table 1: 采用问题(1)中的迭代方法时前10步迭代结果

迭代步数	k=0	k=1	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10
$x^{(k)}$	1	1.7	0.9307	1.746142036	0.857796794	1.789718928	0.786117464	1.827823327	0.721147915	1.858485529	0.667291268

3.2 问题(2)中的迭代法

该迭代法是将方程(1)等价变形为

$$x = \sqrt[3]{20 - 10x - 2x^2} = \varphi_2(x) \tag{4}$$

与问题(1)中的方法类似, 我们编写了iteration2.m 函数进行求解,我们先按照问题(2)中的迭代公式进行迭代. 如果前后两次迭代产生的误差小于10^{-7} , 我们就停止迭代过程. 但是为了防止出现不收敛的情况, 我们还是在中间预设了一个迭代步数$N = 1000$, 即当迭代超过1000 步还没有得出结果的话, 迭代过程就会退出. 结果发现, 运行1000 步之后迭代仍然没有收敛, 因此可以说这种方法是不收敛的. 我们可以看一下前10次迭代的结果, 如Table 2 所示, 可以看出, 问题(2)中给出的迭代格式不收敛.

Table 2: 采用问题(2)中的迭代方法时前10步迭代结果

迭代步数	k=0	k=1	k=2	k=3	k=4	k=5	k=6	k=7	k=8	k=9	k=10
$x^{(k)}$	1	1.7	0.9307	1.746142036	0.857796794	1.789718928	0.786117464	1.827823327	0.721147915	1.858485529	0.667291268

4 Steffensen加速迭代法

Steffensen方法是不动点迭代法$x_{k+1} = \varphi(x_k)$的一种加速收敛方法, 其计算公式如下:

$$x_{k+1} = \psi(x_k) \tag{5}$$

其中有

$$\psi(x) = x - \frac{[\varphi(x) - x]^2}{\varphi(\varphi(x)) - 2\varphi(x) + x} = \frac{x\varphi(\varphi(x)) - [\varphi(x)]^2}{\varphi(\varphi(x)) - 2\varphi(x) + x} \tag{6}$$

综上, 我们写出Steffensen加速迭代法的具体格式:

$$x_{k+1} = x_k - \frac{[\varphi(x_k) - x_k]^2}{\varphi(\varphi(x_k)) - 2\varphi(x_k) + x_k} = \frac{x_k\varphi(\varphi(x_k)) - [\varphi(x_k)]^2}{\varphi(\varphi(x_k)) - 2\varphi(x_k) + x_k} \tag{7}$$

对于Steffensen加速法来说, 它有很好的收敛性, 大多数情况下有至少二阶的收敛性. 如果$\varphi(x)$在其不动点α的邻域内二阶导数连续, 而且有$\varphi'(\alpha) \neq 1$, 则Steffensen方法是二阶收敛的.

4.1 问题(1)中迭代法的Steffensen加速迭代法

对于问题(1)中的迭代函数$\varphi_1(x) = \frac{20-2x^2-x^3}{10}$, 我们编写了phi1.m函数, 在iteration3.m中的Steffensen加速迭代算法中进行调用. 我们还是以前后两次迭代的结果的差是否小于10^{-7}作为迭代中止的条件, 同时我们在循环中加入了一个计数器, 用以统计迭代的次数. 另外, 我们将迭代的结果写入了Excel 表格(如程序iteration3.m 所示), 最终只需4次迭代就可以得到精度高于10^{-7}的解, 结果如Table 3 所示, 与真解$x^* = 1.368808107\cdots$已经很接近, 相差在10^{-8}数量级.

Table 3: 采用问题(1)中的迭代方法采用Steffensen加速法后的迭代结果

迭代步数	k=0	k=1	k=2	k=3	k=4
x_k	1	1.333492139	1.368415439	1.368808058	1.368808108

可以看到, 原来问题(1)中的迭代方法是不收敛的, 但是经过Steffensen加速方法改进后, 只需四次迭代, 就可以达到10^{-7}的精度, 收敛速度非常快.

4.2 问题(2)中迭代法的Steffensen加速迭代法

对于问题(1)中的迭代函数$\varphi_2(x) = \sqrt[3]{20 - 10x - 2x^2}$, 我们编写了phi2.m函数, 在iteration4.m中的Steffensen加速迭代算法中进行调用. 我们还是以前后两次迭代的结果的差是否小于10^{-7}作为迭代中止的条件, 同时我们在循环中加入了一个计数器, 用以统计迭代的次数. 另外, 我们将迭代的结果写入了Excel 表格(如程序iteration4.m 所示), 最终只需8次迭代就可以得到精度高于10^{-7}的解, 结果如Table 4 所示, 与真解$x^* = 1.368808107\cdots$已经很接近, 相差在10^{-8}数量级.可以看到, 原来问题(2)中的迭代方法是不收敛的, 但是经过Steffensen加速方法改进后, 需要8次迭代, 就可以达到10^{-7}的精度, 收敛速度非常快.

Table 4: 采用问题(2)中的迭代方法采用Steffensen加速法后的迭代结果

迭代步数	k=0	k=1	k=2	k=3	k=4	k=5	k=6	k=7	k=8
x_k	1	1.285714286	1.495934157	1.347466215	1.399403509	1.365367836	1.368795737	1.368808142	1.368808108

5 Newton迭代法

Newton迭代法是一种特殊的不动点迭代法, 一般来说,Newton迭代法满足一定条件时具有局部二阶收敛性.它的计算公式为:

$$x_{k+1} = x_k - \frac{x_k}{f'(x_k)}, k = 0, 1, 2, \cdots \tag{8}$$

它的几何解释是:$f(x) = 0$的解x^*是曲线$y = f(x)$与x轴交点的横坐标, 在曲线上的点$(x_k, f(x_k))$处做曲线的切线, 切线与x轴的交点的横坐标即为x_{k+1}.

具体来说, 迭代格式为

$$x_{k+1} = x_k - \frac{x_k^3 + 2x_k^2 + 10x_k - 20}{3x_k^2 + 4x_k + 10}, k = 0, 1, 2, \cdots \tag{9}$$

针对问题(5), 我编写了函数newton_method.m进行求解, 在函数中, 利用了公式(8)中的迭代格式, 并且把当前后两次的迭代结果的差是否小于10^{-7}作为迭代中止的条件. 与前面的类似, 也将迭代的结果写在了Excel表格中, 最终的迭代结果如Table 5 所示. 从中可以看出, Newton迭代法只需四次迭代就可以使精度达到10^{-8}.

Table 5: 采用Newton迭代方法后的迭代结果

迭代步数	k=0	k=1	k=2	k=3	k=4
x_k	1	1.411764706	1.369336471	1.368808189	1.368808108

6 三阶收敛的迭代格式

根据上课讲授的构造方法, 我构造三阶收敛格式如下:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} - \frac{f''(x_k)f^2(x)}{2[f'(x_k)]^3}, k = 0, 1, 2, \cdots \tag{10}$$

具体来说,迭代格式为:

$$x_{k+1} = x_k - \frac{x_k^3 + 2x_k^2 + 10x_k - 20}{3x_k^2 + 4x_k + 10} - \frac{(6x_k + 4)(x_k^3 + 2x_k^2 + 10x_k - 20)^2}{2(3x_k^2 + 4x_k + 10)^3} \tag{11}$$

然后我编写了函数three_order_converge.m来进行计算,与前面类似,把当前后两次的迭代结果的差是否小于10^{-7}作为迭代中止的条件. 与前面的类似, 也将迭代的结果写在了Excel表格中, 最终的迭代结果如Table 6 所示. 从中可以看出, 三阶收敛的迭代法只需3次迭代就可以使精度达到10^{-8}.

Table 6: 采用三阶收敛迭代方法后的迭代结果

迭代步数	k=0	k=1	k=2	k=3
x_k	1	1.361897008	1.368808068	1.368808108

7 实验小结

这次上机实验主要是练习非线性方程的解法, 这里我们重点讨论了不动点迭代法. 对于一般的迭代形式, 可能出现不收敛或者收敛速度很慢的情况, 对于这种情况, 我们可以采用Steffensen加速的方法对迭代函数加以改进. 同时, 我们可以采用一般具有二阶精度的Newton迭代法, 可以看到Newton迭代法的收敛速度很快. 当然, 我们也可以采用老师上课介绍的关于如何构造任意阶精度的迭代函数的方法进行构造, 达到我们预期的收敛速度.

8 Matlab源程序

(1) **iteration1.m**

```
% 采用第一种迭代方法进行迭代
% Author: Qun LIU
% Email: liu-q14@mails.tsinghua.edu.cn
% Time: 2014-11-02
clear;
clc;
% 初值
x0 = 1;
% 精度要求
err = 1;
% 迭代次数初值
i = 0;
% 迭代次数最大限制
N = 1000;
while(err > 1e-7)
    if i > N
        break;
    end
    x1 = (20 - 2*x0^2 - x0^3)/10;
    err = abs(x1 - x0);
    x0 = x1;
    i = i + 1;
end
if(err < 1e-7)
    disp(['第一种方法迭代次数为: ' num2str(i) ' 次'])
    disp(['结果为x= ' num2str(x0, 15)])
else
    disp(['迭代 ' num2str(N) ' 次后不收敛.'])
    x0 = 1;
    % 显示前个迭代结果n
    n = 10;
    x = zeros(1,n);
    for i = 1:n
        x(i)=(20 - 2*x0^2 - x0^3)/10;
        x0 = x(i);
    end
    disp(['前 ' num2str(n) ' 步迭代结果为:'])
    x
    xlswrite('iteration1.xlsx', x)
end
```

(2) **iteration2.m**

```
% 采用第二种迭代法
% Author: Qun LIU
% Email: liu-q14@mails.tsinghua.edu.cn
% Time: 2014-11-02

clear;
clc;
%初值
x0 = 1;
% 精度要求
err = 1;
```

```
% 迭代次数初值
i = 0;
% 迭代次数最大限制
N = 1000;
while(err > 1e-7)
    if i > N
        break;
    end
    x1 = (20 - 10*x0 - 2*x0^2)^(1/3);
    err = abs(x1 - x0);
    x0 = x1;
    i = i + 1;
end
if(err < 1e-7)
     disp(['第二种方法迭代次数为: ' num2str(i) ' 次'])
     disp(['结果为x= ' num2str(x0, 15)])
else
     disp(['迭代 ' num2str(N) ' 次后不收敛.'])
     x0 = 1;
     % 显示前个迭代结果n
     n = 10;
     x = zeros(1,n);
     for i = 1:n
          x(i)=(20 - 2*x0^2 - x0^3)/10;
          x0 = x(i);
     end
     disp(['前 ' num2str(n) ' 步迭代结果为:'])
     x
     xlswrite('iteration2.xlsx', x)
end
```

(3) **phi1.m**

```
function y = phi1(x)
% (1)的迭代函数
% Qun LIU
% Time: 2014-11-02
y=(20 - 2*x^2 - x^3)/10;
end
```

(4) **iteration3.m**

```
% 采用第一种迭代方法的加速法进行迭代Steffensen
% Author: Qun LIU
% Email: liu-q14@mails.tsinghua.edu.cn
% Time: 2014-11-02

clear;
clc;
% 初值
x0 = 1;
% 精度要求
err = 1;
% 迭代次数
i = 0;
while(err > 1e-7)
    x(i+1) = (x0*phi1(phi1(x0))-(phi1(x0))^2)/(phi1(phi1(x0)) - 2*phi1(x0
        ) + x0);
    err = abs(x(i+1) - x0);
```

```
    x0 = x(i+1);
    i = i + 1;
end
disp(['第一种方法经加速后迭代次数为Steffensen: ' num2str(i) ' 次'])
disp(['结果为x= ' num2str(x0, 15)])
xlswrite('iteration3.xlsx', x)
```

(5) **phi2.m**

```
function y = phi2(x)
% (1)的迭代函数
% Qun LIU
% Time: 2014-11-02
y=(20 - 10*x - 2*x^2)^(1/3);
end
```

(6) **iteration4.m**

```
% 采用第二种迭代法的加速法Steffensen
% Author: Qun LIU
% Email: liu-q14@mails.tsinghua.edu.cn
% Time: 2014-11-02

clear;
clc;
%初值
x0 = 1;
% 精度要求
err = 1;
% 迭代次数
i = 0;
while(err > 1e-7)
    x(i+1) = (x0*phi2(phi2(x0))-(phi2(x0))^2)/(phi2(phi2(x0)) - 2*phi2(x0
        ) + x0);
    err = abs(x(i+1) - x0);
    x0 = x(i+1);
    i = i + 1;
end
disp(['第二种方法经加速后迭代次数为Steffensen: ' num2str(i) ' 次'])
disp(['结果为x= ' num2str(x0, 15)])
xlswrite('iteration4.xlsx', x)
```

(7) **newton_method.m**

```
% 采用迭代方法进行迭代Newton
% Author: Qun LIU
% Email: liu-q14@mails.tsinghua.edu.cn
% Time: 2014-11-02

clear;
clc;
% 初值
x0 = 1;
% 精度要求
err = 1;
% 迭代次数
i = 0;
while(err > 1e-7)
    x(i+1) = x0 - (x0^3+2*x0^2+10*x0-20)/(3*x0^2+4*x0+10);
    err = abs(x(i+1) - x0);
```

```
    x0 = x(i+1);
    i = i + 1;
end
disp(['迭代法迭代次数为Newton:␣' num2str(i) '␣次'])
disp(['结果为x=␣' num2str(x0, 15)])
xlswrite('Newton.xlsx', x)
```

(8) **three_order_converge.m**

```
% Three order convergence
% x = x - f(x)/f'(x) - f"(x)f(x)/(2f'(x)^3)
% Author: Qun LIU
% Email: liu-q14@mails.tsinghua.edu.cn

clear;
clc;
% 初值
x0 = 1;
% 精度要求
err = 1;
% 迭代次数
i = 0;
while(err > 1e-7)
    x(i+1) = x0 - (x0^3+2*x0^2+10*x0-20)/(3*x0^2+4*x0+10) - (6*x0+4)*(x0
        ^3+2*x0^2+10*x0-20)^2/2/(3*x0^2+4*x0+10)^3;
    err = abs(x(i+1) - x0);
    x0 = x(i+1);
    i = i + 1;
end
disp(['三阶收敛的迭代法迭代次数为:␣' num2str(i) '␣次'])
disp(['结果为x=␣' num2str(x0, 15)])
xlswrite('three_order.xlsx', x)
```