计　算　方　法

实验二 Newton 迭代法

姓名　孙骁

学号　1180300811

院系　计算机学院

专业　计算机系

哈尔滨工业大学

题目

Newton 迭代法

摘要

求非线性方程 $f(x)=0$ 的根 x^*，Newton 迭代法如下，选取初值 $x_0=\alpha$，通过迭代公式

$$x_k = x_k - \frac{f(x_k)}{f'(x_k)}$$
$$k=0,1,...$$

产生逼近解 x^* 的迭代序列 $\{x_k\}$. 当 x_0 距 x^* 较近时，$\{x_k\}$ 很快收敛于 x^*。但当 x_0 选择不当时，会导致 $\{x_k\}$ 发散。故事先规定迭代的最多次数. 若超过这个次数仍不收敛，则停止迭代另选初值.

一般地，牛顿迭代法具有局部收敛性，为保证迭代收敛，要求，对充分小的 $\delta>0$，$\alpha\in O(x^*,\delta)$. 如果 $f(x)\in C^2[a,b]$，$f(x^*)=0$，$f'(x^*)=0$，那么，对充分小的 $\delta>0$，当 $\alpha\in O(x^*,\delta)$ 时，由牛顿迭代法计算出的 $\{x_k\}$ 收敛于 x^*，且收敛速度是 2 阶的；如果 $f(x)\in C^m[a,b]$，$f(x^*)=f'(x^*)=\cdots=f^{(m-1)}(x^*)=0$，$f^{(m)}(x^*)\neq 0 (m>1)$，那么，对充分小的 $\delta>0$，当 $\alpha\in O(x^*,\delta)$ 时，由牛顿迭代法计算出的 $\{x_k\}$ 收敛于 x^*，且收敛速度是 1 阶的.

前言（目的和意义）

目的：

利用 Newton 迭代法求 $f(x)=0$ 的根.

意义：

通过此次实验，使用编程语言实现 Newton 迭代法，学会使用 Newton 迭代法求 $f(x)=0$ 的根，以解决其他科学实验中的函数求根计算问题.

数学原理

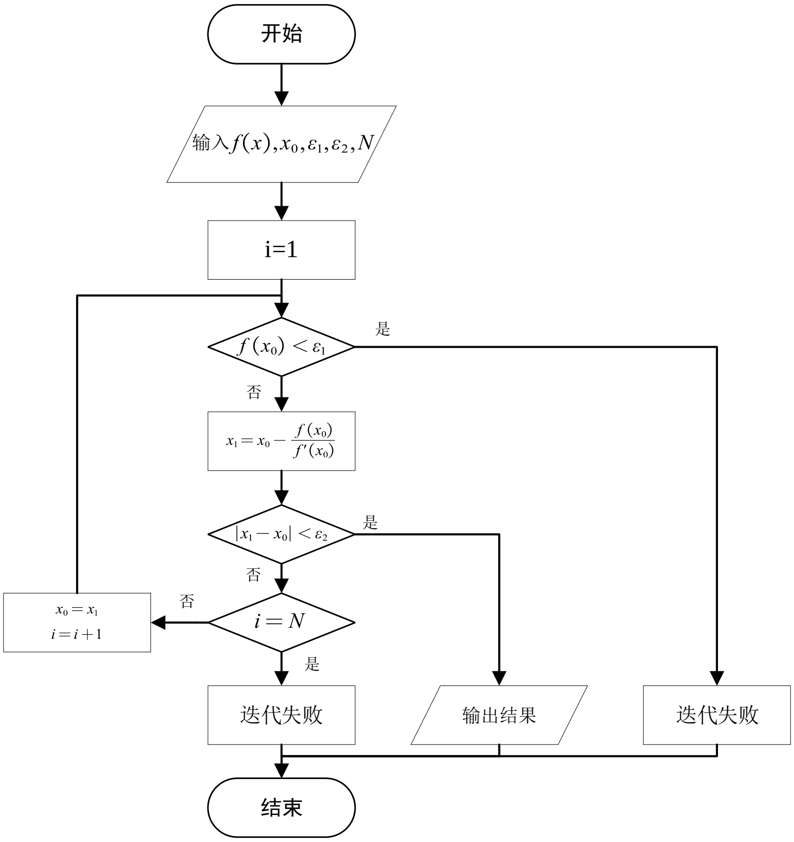
求非线性方程$f(x)=0$的根x^*，Newton 迭代法如下，选取初值$x_0=\alpha$，通过迭代公式

$$x_k = x_k - \frac{f(x_k)}{f'(x_k)}$$
$$k=0,1,...$$

产生逼近解x^*的迭代序列$\{x_k\}$. 当x_0距x^*较近时，$\{x_k\}$很快收敛于x^*。但当x_0选择不当时，会导致$\{x_k\}$发散。故事先规定迭代的最多次数. 若超过这个次数仍不收敛，则停止迭代另选初值.

一般地，牛顿迭代法具有局部收敛性，为保证迭代收敛，要求，对充分小的$\delta>0$，$\alpha\in O(x^*,\delta)$. 如果$f(x)\in C^2[a,b]$，$f(x^*)=0$，$f'(x^*)=0$，那么，对充分小的$\delta>0$，当$\alpha\in O(x^*,\delta)$时，由牛顿迭代法计算出的$\{x_k\}$收敛于x^*，且收敛速度是 2 阶的；如果$f(x)\in C^m[a,b]$，$f(x^*)=f'(x^*)=\cdots=f^{(m-1)}(x^*)=0$，$f^{(m)}(x^*)\neq 0(m>1)$，那么，对充分小的$\delta>0$，当$\alpha\in O(x^*,\delta)$时，由牛顿迭代法计算出的$\{x_k\}$收敛于$x^*$，且收敛速度是 1 阶的.

程序设计流程



开始
输入$f(x),x_0,\varepsilon_1,\varepsilon_2,N$
i=1
$f(x_0)<\varepsilon_1$
是
否
$x_1=x_0-\frac{f(x_0)}{f'(x_0)}$
$|x_1-x_0|<\varepsilon_2$
是
否
$i=N$
否
$x_0=x_1$
$i=i+1$
是
迭代失败
输出结果
迭代失败
结束

实验结果、结论与讨论

问题 1：

（1）$\cos x - x = 0$，$\varepsilon_1 = 10^{-6}$，$\varepsilon_2 = 10^{-4}$，$N = 10$，$x_0 = \dfrac{\pi}{4} \approx 0.785398163$

```
>> syms x;
>> f(x) = cos(x) - x;
>>  fprintf("%f\n", Newton(f,pi/4,1e-6,1e-4,10));
0.739085
```

（2）$e^{-x} - \sin x = 0$，$\varepsilon_1 = 10^{-6}$，$\varepsilon_2 = 10^{-4}$，$N = 10$，$x_0 = 0.6$

```
>> syms x;
>> f(x) = exp(-x) - sin(x);
>> fprintf("%f\n", Newton(f,0.6,1e-6,1e-4,10));
0.588533
```

问题 2：

（1）$x - e^{-x} = 0$，$\varepsilon_1 = 10^{-6}$，$\varepsilon_2 = 10^{-4}$，$N = 10$，$x_0 = 0.5$

```
>> syms x;
>> f(x) = x-exp(-x);
>> fprintf("%f\n", Newton(f,0.5,1e-6,1e-4,10));
0.567143
```

（2）$x^2 - 2xe^{-x} + e^{-2x} = 0$，$\varepsilon_1 = 10^{-6}$，$\varepsilon_2 = 10^{-4}$，$N = 20$，$x_0 = 0.5$

```
>> syms x;
>> f(x) = x^2 - 2 * x * exp(-x) + exp(-2 * x);
>> fprintf("%f\n", Newton(f,0.5,1e-6,1e-4,20));
0.566942
```

问题 3：

（1）

①

```
>> p_2 = Legendre(2)
p_2 = (3*x^2)/2 - 1/2
```

即 $P_2(x) = \dfrac{3}{2}x^2 - \dfrac{1}{2}$

```
>> p_3 = Legendre(3)
p_3 = (5*x*((3*x^2)/2 - 1/2))/3 - (2*x)/3
>> p_3 = simplify(p_3)
```

```
p_3 = (x*(5*x^2 - 3))/2
```
即 $P_3(x)=\dfrac{5}{2}x^3-\dfrac{3}{2}x$

```
>> p_4 = Legendre(4)
p_4 =3/8 - (9*x^2)/8 - (7*x*((2*x)/3 - (5*x*((3*x^2)/2 - 1/2))/3))/4
>> p_4 = simplify(p_4)
p_4 = (35*x^4)/8 - (15*x^2)/4 + 3/8
```
即 $P_4(x)=\dfrac{35}{8}x^4-\dfrac{15}{4}x^2+\dfrac{3}{8}$

```
>> p_5 = Legendre(5)
p_5 =(8*x)/15 - (4*x*((3*x^2)/2 - 1/2))/3 - (9*x*((7*x*((2*x)/3 - (5*x*((3*x^2)/2 - 1/2))/3))/4 + (9*x^2)/8 - 3/8))/5
>> p_5 = simplify(p_5)
p_5 = (x*(63*x^4 - 70*x^2 + 15))/8
```
即 $P_5(x)=\dfrac{63}{8}x^5-\dfrac{35}{4}x^3+\dfrac{15}{8}x$

②

```
>> p_6 = Legendre(6)
p_6 = (35*x*((2*x)/3 - (5*x*((3*x^2)/2 - 1/2))/3))/24 - (11*x*((4*x*((3*x^2)/2 - 1/2))/3 - (8*x)/15 + (9*x*((7*x*((2*x)/3 - (5*x*((3*x^2)/2 - 1/2))/3))/4 + (9*x^2)/8 - 3/8))/5))/6 + (15*x^2)/16 - 5/16
>> p_6 = simplify(p_6)
p_6 = (231*x^6)/16 - (315*x^4)/16 + (105*x^2)/16 - 5/16
```
即 $P_6(x)=\dfrac{231}{16}x^6-\dfrac{315}{16}x^4+\dfrac{105}{16}x^2-\dfrac{5}{16}$

```
>> p = sym2poly(p_6);
>> result = roots(p);
>> result

result =

  -0.932469514203153
  -0.661209386466264
   0.932469514203152
   0.661209386466263
  -0.238619186083197
   0.238619186083197
```
与给定的参考值基本一致.

（2）

①

```
>> t2 = Chebyshev(2)
t2 = 2*x^2 - 1
```

即$T_2(x)=2x^2-1$

```
>> t3 = Chebyshev(3)
t3 = 2*x*(2*x^2 - 1) - x
>> t3 = simplify(t3)
t3 = x*(4*x^2 - 3)
```

即$T_3(x)=4x^3-3x$

```
>> t4 = Chebyshev(4)
t4 = 1 - 2*x^2 - 2*x*(x - 2*x*(2*x^2 - 1))
>> t4 = simplify(t4)
t4 = 8*x^4 - 8*x^2 + 1
```

即$T_4(x)=8x^4-8x^2+1$

```
>> t5 = Chebyshev(5)
t5 = x - 2*x*(2*x^2 - 1) - 2*x*(2*x*(x - 2*x*(2*x^2 - 1)) + 2*x^2 - 1)
>> t5 = simplify(t5)
t5 = x*(16*x^4 - 20*x^2 + 5)
```

即$T_5(x)=16x^5-20x^3+5x$

②

```
>> t6 = Chebyshev(6)
t6 = 2*x*(x - 2*x*(2*x^2 - 1)) - 2*x*(2*x*(2*x^2 - 1) - x + 2*x*(2*x*(x - 2*x*(2*x^2 - 1)) + 2*x^2 - 1)) + 2*x^2 - 1
>> t6 = simplify(t6)
t6 = 32*x^6 - 48*x^4 + 18*x^2 - 1
```

即$T_6(x)=32x^6-48x^4+18x^2-1$

```
>> t = sym2poly(t6)
t =
    32     0   -48     0    18     0    -1

>> result = roots(t)
result =

  -0.965925826289068
  -0.707106781186546
   0.965925826289069
   0.707106781186547
  -0.258819045102521
   0.258819045102521
```

按照给出的参考值

```
>> i = 0:5;
>> x = cos((2*i+1)*pi/2/(5+1));
>> x

x =
    0.965925826289068
    0.707106781186548
    0.258819045102521
   -0.258819045102521
   -0.707106781186547
   -0.965925826289068
```

与给定的参考值基本一致.

（3）

①

```
>> l2 = Laguerre(2)
l2 = (x - 1)*(x - 3) - 1
>> l2 = expand(l2)
l2 = x^2 - 4*x + 2
```

即$L_2(x)=x^2-4x+2$

```
>> l3 = Laguerre(3)
l3 = 4*x - ((x - 1)*(x - 3) - 1)*(x - 5) - 4
>> l3 = expand(l3)
l3 = - x^3 + 9*x^2 - 18*x + 6
```

即$L_3(x)=-x^3+9x^2-18x+6$

```
>> l4 = Laguerre(4)
l4 = (x - 7)*(((x - 1)*(x - 3) - 1)*(x - 5) - 4*x + 4) - 9*(x - 1)*(x - 3) + 9
>> l4 = expand(l4)
l4 = x^4 - 16*x^3 + 72*x^2 - 96*x + 24
```

即$L_4(x)=x^4-16x^3+72x^2-96x+24$

```
>> l5 = Laguerre(5)
l5 = 16*((x - 1)*(x - 3) - 1)*(x - 5) - 64*x - (x - 9)*((x - 7)*(((x - 1)*(x - 3) - 1)*(x - 5) - 4*x + 4) - 9*(x - 1)*(x - 3) + 9) + 64
>> l5 = expand(l5)
l5 = - x^5 + 25*x^4 - 200*x^3 + 600*x^2 - 600*x + 120
```

即$L_5(x)=-x^5+25x^4-200x^3+600x^2-600x+120$

②

```
>> l = sym2poly(l5);
>> results = roots(l);
>> results

results =
  12.640800844275811
   7.085810005858809
   3.596425771040735
   1.413403059106519
   0.263560319718141
```

与给定的参考值基本一致.

（4）

①

```
>> h2 = Hermite(2)
h2 = 4*x^2 – 2
```

即$H_2(x)=4x^2-2$

```
>> h3 = Hermite(3)
h3 = 2*x*(4*x^2 - 2) - 8*x
>> h3 = simplify(h3)
h3 = 4*x*(2*x^2 - 3)
```

即$H_3(x)=8x^3-12x$

```
>> h4 = Hermite(4)
h4 = 12 - 24*x^2 - 2*x*(8*x - 2*x*(4*x^2 - 2))
>> h4 = simplify(h4)
h4 = 16*x^4 - 48*x^2 + 12
```

即$H_4(x)=16x^4-48x^2+12$

```
>> h5 = Hermite(5)
h5 = 64*x - 16*x*(4*x^2 - 2) - 2*x*(2*x*(8*x - 2*x*(4*x^2 - 2)) + 24*x^2 - 12)
>> h5 = simplify(h5)
h5 = 8*x*(4*x^4 - 20*x^2 + 15)
```

即$H_5(x)=32x^5-160x^3+120x$

②

```
>> h6 = Hermite(6)
h6 = 20*x*(8*x - 2*x*(4*x^2 - 2)) - 2*x*(16*x*(4*x^2 - 2) - 64*x + 2*x*(2*x*(8*x - 2*x*(4*x^2 - 2)) + 24*x^2 - 12)) + 240*x^2 - 120
>> h6 = simplify(h6)
h6 = 64*x^6 - 480*x^4 + 720*x^2 – 120
```

即$H_6(x)=64x^6-480x^4+720x^2-120$

```
>> h = sym2poly(h6);
>> results = roots(h);
>> results

results =
  -2.350604973674488
   2.350604973674488
  -1.335849074013696
   1.335849074013698
  -0.436077411927617
   0.436077411927616
```

与给定的参考值基本一致.

思考题

问题 1：

由于 Newton 法具有局部收敛性，所以当实际问题本身能提供接近于根的初始近似值时，就可保证迭代序列收敛，但当初值难以确定时，迭代序列就不一定收敛。

实际计算时应先用比较稳定的算法，如二分法，计算根的近似值，再将该近似值作为牛顿法的初值，以保证迭代序列的收敛性。

问题 2：

实验 2 中两个方程根其实相同，只是第二个方程为重根，通过比较迭代次数，第一个方程迭代了 3 次得出结果，第二个方程迭代了 8 次得出结果，且第二个方程的结果不如第一个准确，这是由于第二个方程在根处导数为 0，在根的领域内导数很小使 Newton 法收敛速度变慢，精度变低。

问题 3：

这些多项式在比较小的区间内有多个根，这就致使其导数也会有多个根，因此如果用 Newton 法寻根的话，初值非常不好估计，所以要用最稳定的二分法找它们的根。

程序代码

Newton.m

```
function result = Newton(fun, x0, ftol, dftol, maxit)
x = x0;
i = 0;
while i <= maxit
    i = i + 1;
    f = feval(fun,x);
    dfdx = diff(fun);
    df = feval(dfdx,x);
    if abs(df) < dftol
        result = [];
        warning('dfdx is too small!');
        return;
    end
    dx = f/df;
    x = x - dx;
    if abs(f) < ftol
        result = x;
        return;
    end
end
result = [];
```

Legendre.m

```
function P = Legendre(n)
syms x
if (n == 0)
    P = 1;
elseif (n == 1)
    P = x;
else
    P = ((2 * n - 1) * x * Legendre(n - 1) - (n - 1) *
Legendre(n - 2)) / (n);
end
end
```

```
Chebyshev.m
function P = Chebyshev(n)
syms x
if (n == 0)
    P = 1;
elseif (n == 1)
    P = x;
else
    P = 2 * x * Chebyshev(n - 1) - Chebyshev(n - 2);
end
end


Laguerre.m
function P = Laguerre(n)
syms x
if (n == 0)
    P = 1;
elseif (n == 1)
    P = 1-x;
else
    P = ((2 * n - 1 - x) * Laguerre(n - 1) - (n - 1)^2 *
Laguerre(n - 2));
end
end


Hermite.m
function P = Hermite(n)
syms x
if (n == 0)
    P = 1;
elseif (n == 1)
    P = 2 * x;
else
    P = (2 * x * Hermite(n - 1) - (n - 1) * 2 * Hermite(n -
2));
end
end
```