MATLAB用二分法、不动点迭代法及Newton迭代（切线）法求非线性方程的根

作者：凯鲁嘎吉 - 博客园
http://www.cnblogs.com/kailugaji/

一、实验原理

1. 二分法：(略)

2. 不动点迭代法：　给x_0，作迭代　$x_{n+1}=\varphi(x_n), n=0,1,\cdots$

TH. 设$\varphi(x)$满足条件

(1)当$x\in[a,b]$时，$\varphi(x)\in[a,b]$；(2)存在正数$L<1$，使对任意$x\in[a,b]$有

$|\varphi'(x)|\le L$，$(or\ \forall x,y\in[a,b]$，有$|\varphi(x)-\varphi(y)|\le L|x-y|)$

则$x=\varphi(x)$在$[a,b]$上有惟一的根x^*，且对任意初值$x_0\in[a,b]$，迭代序列

$$x_{n+1}=\varphi(x_n),\ n=0,1,2,\cdots$$

收敛于x^*。误差估计式为　$|x^*-x_n|\le\dfrac{L^n}{1-L}|x_1-x_0|$。

3. Newton迭代法：设$f(x)$在$[a,b]$上有二阶导数，且满足：

(1)$f(a)f(b)<0$；　(2)$f'(x)\ne 0, x\in[a,b]$；　(3)$f''(x)$不变号，$x\in[a,b]$；

(4)初始值$x_0\in[a,b]$，使$f''(x_0)f(x_0)>0$。

则Newton迭代公式 $x_{n+1}=x_n-\dfrac{f(x_n)}{f'(x_n)}$ $(n=0,1,2,\cdots)$产生的序列$\{x_n\}$收敛于$f(x)=0$在$[a,b]$上的唯一实根。

二、实验步骤

1. 二分法：输入 端点a,b，误差容限$TOL1,TOL2$，最大迭代次数m $(\ge\dfrac{\lg[(b-a)/TOL2]}{\lg 2})$

step1　$a_0\leftarrow a, b_0\leftarrow b$，

step2　对$k=1,2,\cdots,m$，作step3－5。

step3　$p\leftarrow(a+b)/2$。

step4　若$|f(p)|<TOL1$或$(b-a)/2<TOL2$，则输出，停机。

step5　若$f(p)f(b)<0$，则$a\leftarrow p$，否则$b\leftarrow p$。

step6　输出('Method failed')；　停机。

2. 不动点迭代法：输入 初始值x_0;误差容限TOL,最大迭代次数m.

step 1 $p \leftarrow x_0$(初始值)

step 2 对$k=1,2,\cdots,m$,作step3－4.

step 3 $p \leftarrow \varphi(x_0)$.

step 4 若$|p-x_0|<TOL$,则输出p,停机。 否则$x_0 \leftarrow p$.

step 5 输出('Method failed'); 停机.

3. Newton迭代法：输入 初始值x_0;误差容限TOL,最大迭代次数m.

step 1 $p_0 \leftarrow x_0$.

step 2 对$k=1,2,\cdots,m$,作$step$3－4.

step 3 $p \leftarrow p_0-\dfrac{f(p_0)}{f'(p_0)}$.

step 4 若$|p-p_0|<TOL$,则输出p,停机。 否则$p_0 \leftarrow p$.

step 5 输出('Method failed'); 停机.

注：以上三法的迭代终止准则均可用$\dfrac{|p-x_0|}{p}<TOL$,这里相当于$x_0=p_k$,p即p_{k+1}.

三、实验过程

1.(程序)

(1) 二分法：求 $f(x)=x^3+10x-20=0$ 在区间（1，2）之间的根，取$\varepsilon=10^{-4}$.

（a）bipart.m:

```
function [x,m]=bipart(fun,a0,b0,tol)
a=a0;b=b0;
m=1+round(round(log((b-a)/tol))/log(2));
for k=1:m
    p=(a+b)/2;
    if fun(p)*fun(b)<0
            a=p;
    else
            b=p;
    end
```

```
    x=p;
end
```

(b)fun1.m:

```
function f=fun1(x)
f=x^3+10*x-20;
```

（2）不动点迭代法：求方程$f(x)=x^3-2x-5=0$在$x_0=2.0$附近的根，取$\varepsilon=10^{-5}$.

2020/10/17 注：这里求解用的牛顿迭代法，不是不动点迭代法，不动点迭代法的MATLAB程序请参看：MATLAB实例：不动点迭代法求一元函数方程的根 - 凯鲁嘎吉 - 博客园

（a）budong.m:

```
function [x,k]=budong(fun,x0,tol,m)
for k=1:m
    x=fun(x0);
    if abs(x-x0)<tol
        break;
    end
    x0=x;
end
x=vpa(x,8);
```

(b)fun.m

```
function t=fun(x1)
syms x;
f=x^3-2*x-5;
s=subs(diff(f,x),x,x1);
x=x1;
f=x^3-2*x-5;
t=x-f/s;
```

（3）牛顿迭代法：求方程$f(x)=x^3-2x-5=0$在$x_0=2.0$附近的根，取$\varepsilon=10^{-5}$.

newton.m:

```
function x1=newton(t1,esp,m)

syms x;
```

```
fun=x^3+2*x-5;
for k=1:m
    if abs(subs(diff(fun,'x'),x,t1))<esp
        x1=t1;
        break;
    else
        if subs(diff(fun,'x',2),x,t1)==0
            break;
            disp('解题失败！')
        else
            t0=t1;
            t1=t0-subs(fun,x,t0)/subs(diff(fun,'x'),x,t0);
            if abs(t1-t0)<esp
                x1=t1;
                break;
            end
        end
    end
end
x1=vpa(x1,8);
```

2.(运算结果)

（1）二分法:

```
>> [x,m]=bipart(@fun1,1,2,0.0001)
x =
    1.5945
m =
    14
```

(2)不动点迭代法:

```
>> [x,k]=budong(@fun,2,1e-5,100)
x =
2.0945515
k =
     4
```

(3)牛顿迭代法:

```
 >> x1=newton(2,1e-4,20)
     x1 =
        1.3282689
```

3.(拓展（方法改进、体会等）)

对于方程的根为重根的情形，newton法求重根只是线性收敛，迭代缓慢，如果对于求重根的情形，对newton法进行改进，取

$$\varphi(x)=x-m\frac{f(x)}{f'(x)},$$

则 $\varphi'(x^*)=0$。用迭代法

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

求m重根，则具有二阶收敛性，但要知道的重数m。

计算方程 $x^4-4x^2+4=0$ 的根 $x^*=\sqrt{2}$ 是二重根，用newton法与改进方法求根。

源程序:

newton_biroot.m:

```
function t=newton_biroot(x1)

syms x;

f=x^4-4*(x^2)+4;

s=subs(diff(f,x),x,x1);

x=x1;

f=x^4-4*(x^2)+4;

t=x-f/s;
```

biroot1.m:

```
function t=biroot1(x1)

syms x;

f=x^4-4*(x^2)+4;

s=subs(diff(f,x),x,x1);

x=x1;

f=x^4-4*(x^2)+4;

t=x-2*f/s;
```

budong.m:

```
function [x,k]=budong(fun,x0,tol,m)

for k=1:m

    x=fun(x0);

    if abs(x-x0)<tol

        break;

    end
```

```
    x0=x;
    x=vpa(x,8)
end
x=vpa(x,8);
```

运行结果：取初值为2

k	xk	newton法	改进方法
1	x1	1.75	1.5
2	x2	1.5982143	1.4166667
3	x3	1.5115099	1.4142157
4	x4	1.4644275	1.4142157

计算4步，改进方法就已经收敛，而newton法只是线性收敛，要达到同样精度需迭代17次。

附结果：

```
>> [x,k]=budong(@biroot1,2,1e-5,3)

x =

1.5

x =

1.4166667

x =
```

```
1.4142157

x =

1.4142157

k =

     3

>> [x,k]=budong(@biroot1,2,1e-5,10)

x =

1.5

x =

1.4166667

x =

1.4142157

x =

1.4142136

k =

     4

>> [x,k]=budong(@newton_biroot,2,1e-5,50)

x =

1.75
```

x =

1.5982143

x =

1.5115099

x =

1.4644275

x =

1.439751

x =

1.4270955

x =

1.4206836

x =

1.4174559

x =

1.4158366

x =

1.4150256

x =

1.4146197

x =

1.4144166

x =

1.4143151

x =

1.4142643

x =

1.414239

x =

1.4142263

x =

1.4142199

k =

    17