

EM314 – NUMERICAL METHODS  
ASSIGNMENT 02

DE SILVA K.G.P.M.

E/15/065

SEMESTER 04

12/11/2018

(1)

①  $f(a) \neq 0$

By convergence theorem, error estimate =

$$\underbrace{|x_k - x^*|}_{= e_k} \leq \frac{b-a}{2^{k+1}}$$

$$e_k \leq \frac{b-a}{2^{k+1}}$$

When  $e^k < \tau$

$$\frac{b-a}{2^{k+1}} < \tau$$

$$\frac{b-a}{\tau} < 2^{k+1}$$

$$\log_2 \left( \frac{b-a}{\tau} \right) < \log_2 2^{k+1}$$

$$\log_2 \left( \frac{b-a}{\tau} \right) < (k+1) \times 1$$

$$\log_2 \left( \frac{b-a}{\tau} \right) < k+1$$

$$k > \log_2 \left( \frac{b-a}{\tau} \right) - 1$$

$$\underline{\underline{k > \log_2 \left( \frac{b-a}{\tau} \right) - 1}}$$

(2)

a)  $g(x) = e^{-x}$   $G = [\ln(1.1), \ln(3)]$

$$L = |g'(x)| = |-e^{-x}|$$

$$= |e^{-x}|.$$

We get the largest value for  $L$  when  $x = \ln(1.1)$ .  
Because this is a decreasing function.

$$L = |e^{-\ln(1.1)}| = 0.9090 < 1.$$

Then  $g$  is a contraction on  $G = [\ln(1.1), \ln(3)]$ .

b)  $G = [\ln(1.1), \ln(3)]$   
 $G = [0.09531, 1.0986]$

$$g(x) = e^{-x}$$

$$g(\ln(1.1)) = e^{-\ln(1.1)} = 0.9090 \quad \text{--- ①}$$

$$g(\ln(3)) = e^{-\ln(3)} = 0.3334 \quad \text{--- ②}$$

① and ② falls on  $G = [\ln(1.1), \ln(3)]$  and this is a decreasing function as in the previous one.

So,  $g: G \rightarrow G$ .

c)  $x_{k+1} = g(x_k)$

Let us use this result  $|x_{k+1} - x_k| \leq L^k |x_1 - x_0|$

We have to prove it first.

$$x_{k+1} = g(x_k), \quad x_k = g(x_{k-1})$$

$$|f(x_k) - f(x_{k-1})| \leq L |x_k - x_{k-1}| \quad \text{--- ①}$$

$$|x_k - x_{k-1}| \leq |f(x_k) - f(x_{k-1})|$$

$$\text{then } |f(x_{k+1}) - f(x_k)| \leq L |x_{k+1} - x_k| \quad \text{--- ②}$$

by ①, ②

$$|f(x_k) - f(x_{k-1})| \leq L^2 |x_{k-1} - x_{k-2}|$$

$$\leq L^3 |x_{k-2} - x_{k-3}|$$

$$\vdots$$

(2) C)

$$|f(x_k) - f(x_{k+1})| \leq L^k |x_1 - x_0|$$

$$|x_{k+1} - x_k| \leq L^k |x_1 - x_0| \quad \text{--- (3)}$$

by (3),

$$|g(x_k) - x_k| \leq L^k |x_1 - x_0|$$

$$L \in [0, 1)$$

then,  $k$  is large,

$$L^k \rightarrow 0.$$

$$|g(x_k) - x_k| \rightarrow 0$$

$$g(x_k) \rightarrow x_k$$

$x_k$  is the fixed point.

$\therefore x_{k+1} = g(x_k)$  converges to the fixed point  $x_* \in G$   
for any  $x_0 \in G$  //



(3)

③ a). Let us take  $x_k$ .

$$x_k \in [-0.5, 0.5]$$

$$g(x) = \tan^{-1}(2x)$$

$$g'(x) = \frac{2}{1+4x^2}$$

Consider,

$$e_{k+1} = |x_{k+1} - 0|$$

$$x_{k+1} = g(x_k)$$

$$a = g(0)$$

$$e_{k+1} = |g(x_k) - 0|$$

$$= |g'(\xi)| |x_k - 0|$$

$$= |g'(\xi)| e_k$$

But  $|g'(x)| > 1$  for some values in the interval  $(-0.5, 0.5)$  and  $e_{k+1} > e_k$ .

So, fixed point iteration will not converge to this fixed point.

b)

$$x_0 = 2$$

If there is a fixed point,  $\tan^{-1}(2x) = x$

$$g(x) = x$$

Then,  $x_{k+1} = g(x_k) \rightarrow x_1 = g(x_0)$

$$x_0 = 2$$

$$x_1 = g(2) = \tan^{-1}(2 \times 2) = 1.325$$

$$\text{error}_1 = |x_1 - x_0| = |1.325 - 2|$$

$$= 0.675$$

$$x_2 = g(x_1) = \tan^{-1}(2 \times 1.325) = 1.209$$

$$\text{error}_2 = |x_2 - x_1| = |1.209 - 1.325|$$

$$= 0.116$$

(3) b)

b)

ii).

$$f(x) = g(x) - x = \tan^{-1}(2x) - x$$

$$f'(x) = \frac{2}{1+4x^2} - 1 = \frac{2 - (1+4x^2)}{1+4x^2} = \frac{1-4x^2}{1+4x^2}$$

b) Newton's method

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)} = x_k - \frac{[\tan^{-1}(2x_k) - x_k](1+4x_k^2)}{(1-4x_k^2)}$$

Let  $x_0 = 2$ ,

$$x_1 = x_0 - \frac{[\tan^{-1}(2x_0) - x_0](1+4x_0^2)}{(1-4x_0^2)} = 1.2359 //$$

$$e_1 = |x_1 - x_0| = \frac{|1.2359 - 2|}{2} = 0.42 //$$

$$x_2 = x_1 - \frac{[\tan^{-1}(2x_1) - x_1](1+4x_1^2)}{(1-4x_1^2)} = 1.167 //$$

$$e_2 = |x_2 - x_1| = \frac{|1.167 - 1.2359|}{2} = 0.49 //$$

(4) a)

```
Editor - C:\Users\Prasad-PC\newtons.m*
function [sol,res,niter]=newtons(f,Fd,Tol,x0,nmax)
1
2     niter=1;
3     x=x0-f(x0)/Fd(x0);
4     disp(x);
5     while abs(x0-x)>=Tol && niter<=nmax
6         niter=niter+1;
7         x0=x;
8         x=x0-f(x0)/Fd(x0);
9     end
10
11     if niter>nmax
12         disp('Newtons method stop without convergence');
13     end
14     res=abs(x0-x);
15     sol=x;
16 end
```

(4) b)

```
Editor - C:\Users\Prasad-PC\RunNewton.m
1
2     x0=100;
3     Tol=10^(-5);
4     namx=110;
5     f=@(x) x^2+4*x-4;
6     Fd=@(x) x*2+4;
7
8     [sol,res,niter]=newtons(f,Fd,Tol,x0,namx);
9
10    if niter>0
11        fprintf('solution is %f\n',sol);
12        fprintf('Residual is %f\n',res);
13        fprintf('Iterations =%d\n',niter);
14    end
15
```

```
Command Window
>> roots([1,4,-4])

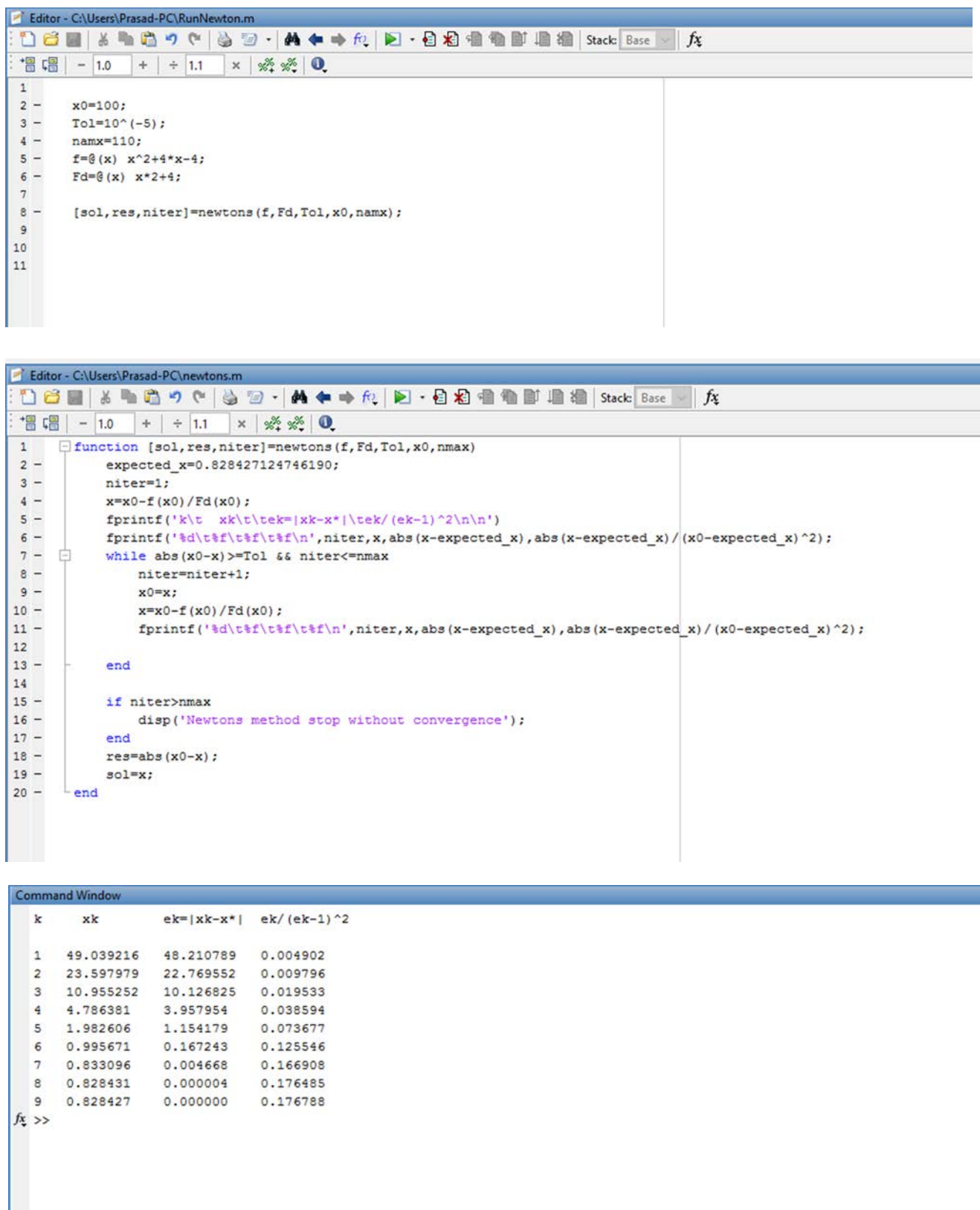
ans =

    -4.828427124746190
     0.828427124746190

solution is 0.828427
Residual is 0.000004
Iterations =9
fx >>
```

Yes. The expected value is 0.828427. And it is obtained.

(4) c)



The image displays two MATLAB windows. The top window, titled 'Editor - C:\Users\Prasad-PC\RunNewton.m', contains a script that defines a function  $f(x) = x^2 + 4x - 4$  and its derivative  $Fd(x) = 2x + 4$ . It then calls a function `newtons(f, Fd, Tol, x0, namx)` with initial values  $x_0 = 100$ ,  $Tol = 10^{-5}$ , and  $namx = 110$ . The bottom window, titled 'Editor - C:\Users\Prasad-PC\newtons.m', shows the implementation of the Newton-Raphson method. It initializes `expected_x` to 0.828427124746190, sets `niter = 1`, and enters a loop that iteratively updates  $x$  until the error is within the tolerance. The loop includes a `while` condition `abs(x0-x) > Tol && niter <= nmax`. After the loop, it checks if `niter > nmax` and displays a message if convergence fails. The `Command Window` at the bottom shows the output of the script, displaying a table of iteration results.

```
1
2 - x0=100;
3 - Tol=10^(-5);
4 - namx=110;
5 - f=@(x) x^2+4*x-4;
6 - Fd=@(x) x*2+4;
7
8 - [sol,res,niter]=newtons(f,Fd,Tol,x0,namx);
9
10
11
```

```
1 function [sol,res,niter]=newtons(f,Fd,Tol,x0,nmax)
2     expected_x=0.828427124746190;
3     niter=1;
4     x=x0-f(x0)/Fd(x0);
5     fprintf('k\t xk\t tek=|xk-x*|\t tek/(ek-1)^2\n\n')
6     fprintf('%d\t%f\t%f\t%f\n',niter,x,abs(x-expected_x),abs(x-expected_x)/(x0-expected_x)^2);
7     while abs(x0-x)>Tol && niter<=nmax
8         niter=niter+1;
9         x0=x;
10        x=x0-f(x0)/Fd(x0);
11        fprintf('%d\t%f\t%f\t%f\n',niter,x,abs(x-expected_x),abs(x-expected_x)/(x0-expected_x)^2);
12
13    end
14
15    if niter>nmax
16        disp('Newtons method stop without convergence');
17    end
18    res=abs(x0-x);
19    sol=x;
20 end
```

k	xk	ek= xk-x*	ek/(ek-1)^2
1	49.039216	48.210789	0.004902
2	23.597979	22.769552	0.009796
3	10.955252	10.126825	0.019533
4	4.786381	3.957954	0.038594
5	1.982606	1.154179	0.073677
6	0.995671	0.167243	0.125546
7	0.833096	0.004668	0.166908
8	0.828431	0.000004	0.176485
9	0.828427	0.000000	0.176788

fx >>

When going to higher iterations the error goes to 0. And approximate value convergence to expected value as expected.



(4) d)

```
Command Window

k      xk      ek=|xk-x*|  ek/(ek-1)^2
1      49.039216  48.210789  0.004902
2      23.597979  22.769552  0.009796
3      10.955252  10.126825  0.019533
4      4.786381   3.957954   0.038594
5      1.982606   1.154179   0.073677
6      0.995671   0.167243   0.125546
7      0.833096   0.004668   0.166908
8      0.828431   0.000004   0.176485
9      0.828427   0.000000   0.176788
10     0.828427   0.000000   16228413.531752

fx >> |
```

Same the as the previous case in higher iterations error equals to 0. The approximate answer convergence to expected value until 9<sup>th</sup> step. But in the 10<sup>th</sup> iteration we got a strange value because of computational limits.

(5)

```
Editor - C:\Users\Prasad-PC\testKepler.m
Stack: Base fx

1 - E0=100;
2 - e=0.8;
3 - Tol=10^(-8);
4 - nmax=100;
5 - M=3;
6 - f=@(E) M-E+e*sin(E);
7 - Fd=@(E) -1+e*cos(E);
8
9 - [sol,res,niter]=newtons(f,Fd,Tol,E0,nmax);
10
11
12 - if niter>0
13 -     fprintf('solution is %f\n',sol);
14 -     fprintf('Residual is %f\n',res);
15 -     fprintf('Iterations =%d\n',niter);
16
17 - end

Command Window

solution is 3.062894
Residual is 0.000000
Iterations =16

fx >>
```

Solution is 3.062894.

(6)

```
Editor - C:\Users\Prasad-PC\Gas.m
1 - N=1000;
2 - T=300;
3 - p=3.5*(10^7);
4 - Tol=10^(-12);
5 - a=0.401;
6 - b=42.7*(10^(-6));
7 - k=1.3806503*(10^(-23));
8 - nmax=100;
9
10 - f=@(x) p*(x.^3)-(p*N*b+2*k*N*T)*(x.^2)+a*(N^2)*x-a*b*(N^3);
11
12 - [zero, res, niter]=bisection(f,0,2,Tol,nmax);
13
14 - if niter>0
15 -     fprintf('solution is %f\n',zero);
16 -     fprintf('Residual is %f\n',res);
17 -     fprintf('Iterations =%d\n',niter);
18
19 - end
```

Given code for bisection method,

```
Editor - C:\Users\Prasad-PC\bisection.m
1 - function [zero, res, niter] = bisection(f,a,b,tol,nmax)
2 -     x = [a (a+b)/2 b]; y = f(x); niter = 0; I = (b-a)/2;
3 -     if y(1)*y(3)>0
4 -         error('The signs of the function at the extrema must be opposite');
5 -     elseif y(1) == 0
6 -         zero = a; res = 0; return
7 -     elseif y(3) == 0
8 -         zero = b; res = 0; return
9 -     end
10 -     while ( I >= tol && niter <= nmax )
11 -         if sign(y(1))*sign(y(2))<0
12 -             x(3) = x(2); x(2) = (x(1) + x(3))/2;
13 -             y = f(x); I = (x(3)-x(1))/2;
14 -         elseif sign(y(2))*sign(y(3))<0
15 -             x(1) = x(2); x(2) = (x(1) + x(3))/2;
16 -             y = f(x); I = (x(3)-x(1))/2;
17 -         else
18 -             x(2) = x(find(y ==0)); I = 0;
19 -         end
20 -         niter = niter+1;
21 -     end
22 -     if niter > nmax
23 -         fprintf('bisection method exited without convergence');
24 -     end
25 -     zero = x(2); res = f(x(2));
```

```
Command Window
solution is 0.042700
Residual is 0.000000
Iterations =40
fx >> |
```

Solution is 0.042700.