EM314 - NUMERICAL METHODS ASSIGNMENT - 2

Hisni Mohammed M.H.

E/15/131

Theory.

Q1. From the error estimate of bisection methode,

$$e_K = |x_K - x_*| \leqslant \frac{b-a}{2^{K+1}}$$

but in bisection method wee need error to be less than tolerance.

$$e_K < \tau$$

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

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

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

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

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

Q2. (a) $g(x) = e^{-x}$

$$g'(x) = -e^{-x} < 0 \quad \forall x.$$

$\forall(x)$ $g'(x)$ is negative.

therefore $g(x)$ is is monotonically decreasing function.

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

when $x = \ln 1.1$.

$$\Rightarrow |g'(x)|_{max}$$

$$e^{-\ln 1.1} = \frac{1}{1.1} < 1$$

$\therefore |g'(x)| < 1$ on G.

$\therefore$ $g(x)$ is a contraction on G

Q2. (b). $g(x)$ is a continuous monotonically decreasing function on G.

and, $g(\ln 1.1) = e^{-\ln(1.1)} = 1/1.1 = 0.9091$

$g(\ln 3) = e^{-\ln(3)} = 1/3 = 0.3333$

$G = [\ln 1.1, \ln 3] = [0.0953, 1.0986]$

$[0.3333, 0.9091] \subseteq [\ln 1.1, \ln 3]$

$\therefore g: G \to G$.

(c). g is a contraction an $g: G \to G$.

therefore from banach fixed point theorem,

then G contains a unique solution x_*

moreover starting with any $x_0 \in G$, the sequence generated by the iterative procedure

$x_{n+1} = g(x_n)$ converges to x_*

Q3. (a) $g(x) = \tan^{-1}(2x)$

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

let $x_k \in [-1/2, 1/2]$

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

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

$$= g'(\xi) \cdot e_k \quad \text{for some } \xi \in (-1/2, 1/2)$$

$$|g'(x)|_{max} = \frac{2}{1+4 \cdot 0} = 2$$

but $g'(x) > 1 \quad \forall x \in (-1/2, 1/2)$ and $e_{k+1} > e_k$

$\therefore$ Therefore the fixed point iteration will not converge to $x_* = 0$.

Q3. (b) (i) $x_0 = 2$. , Fixed point iteration.

1^{st} iteration $\quad x_1 = \tan^{-1}(2 \cdot 2)$

$= 1.3258$.

$e_1 = |e_1 - e_0|$

$= |1.3258 - 2|$

$= 0.6742$.

2^{n} iteration $\quad x_2 = \tan^{-1}(2 \times 1.3258)$

$= 1.2102$.

$e_2 = |e_2 - e_1|$

$= |1.2102 - 1.3258|$

$= 0.1156$

(b) (ii) $x_0 = 2$, Newton's method.

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

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

$$= \frac{1-4x^2}{1+4x^2}$$

Newton's method sequence.

$$x_{n+1} = x_n - \frac{(\tan^{-1} 2x_n - x_n)(1+4x_n^2)}{1-4x_n^2}$$

1^{st} iteration.

$$x_1 = 2 - \frac{(\tan^{-1} 4 - 2)(1+4\times4)}{(1-4\times4)}$$

$= 1.2359$

$e_1 = |2 - 1.2359| = 0.7641$

2^{n} iteration $\quad x_2 = 1.2359 - \dfrac{[\tan^{-1}(2\times1.2359) - 1.2359][1+4\times1.2359^2]}{[1-4\times1.2359^2]}$

$= 1.1670$

$e_2 = |1.1670 - 1.2359| = 0.0689$.

Implementation

Q4. (a), (b), (c)

OCTAVE code (newtons.m)

```
function [zero, res, itr] = newtons(f, df, x0, tol, nmax)
    itr = 0;
    res = abs(x0);
    xor = 2*(sqrt(2)-1);
    fprintf("K\t Xk\t\t ek\t\t ek/(ek-1)^2\t\t\n");
    ekold = abs(x0-xor);
    while ( res >= tol && itr <= nmax)
        x = x0 - (f(x0)/df(x0));
        res = abs(x0-x);
        ek = abs(x-xor);
        con = ek/(ekold^2);
        fprintf("%d\t %d\t %d\t %d\n",itr,x,ek,con);
        x0 = x;
        itr = itr +1;
        ekold = ek;
    end
    if itr > nmax
        fprintf('Newtons method stopped without convergence');
    end
    zero = x;
    endfunction
```

OCTAVE Code (f.m)

```
function y = f(x)
        y = x^2 + 4*x -4;
endfunction
```

OCTAVE Code (df.m)

```
function y = df(x)
        y = 2*x + 4;
endfunction
```

OCTAVE Code (testNewtons.m)

```
x0 = 100;
tol = 10^(-5);
nmax = 100;
[zero, res, itr] = newtons(@f,@df,x0,tol,nmax)
```

Output of the code

```
>> testNewtons
K     Xk          ek          ek/(ek-1)^2
0     49.0392     99.1716     0.0100835
1     23.598      48.2108     0.00490196
2     10.9553     22.7696     0.00979639
3     4.78638     10.1268     0.0195328
4     1.98261     3.95795     0.0385944
5     0.995671    1.15418     0.073677
6     0.833096    0.167243    0.125546
```

7	0.828431	0.00466847	0.166908
8	0.828427	3.84642e-06	0.176485

zero = 0.82843

res = 0.0000038464

itr = 9

(b) Yes**.** The expected solution was accurate up to 5 decimal places.

(c) The quadratic convergence was not obtained, as it can be seen that the ek/(ek-1)^2 Does not go to a constant value.

(d) Part (c) redone with tolerance=$10^{(-8)}$

<u>Output of the code (testNewtons.m)</u>

>> testNewtons

K	Xk	ek	ek/(ek-1)^2
0	49.0392	99.1716	0.0100835
1	23.598	48.2108	0.00490196
2	10.9553	22.7696	0.00979639
3	4.78638	10.1268	0.0195328
4	1.98261	3.95795	0.0385944
5	0.995671	1.15418	0.073677
6	0.833096	0.167243	0.125546
7	0.828431	0.00466847	0.166908
8	0.828427	3.84642e-06	0.176485
9	0.828427	2.61524e-12	0.176766

zero = 0.82843

res = 2.6155e-12

itr = 10

It can be said that, the quadratic convergence was obtained, as the ek/(ek-1)^2 term Does go to a constant value 0.176766

Applications

Q5. **Kepler's Equation.**

Kepler's Equation can be rewritten as,

$$f(E) = E – esin(E) – M$$

Implementation of this function in OCTAVE,

OCTAVE Code (f.m)

```
function y = f(x)

        M = 3;

        e = 0.8;

        y = x - e*sin(x)-M;

endfunction
```

OCTAVE Code (df.m)

```
function y = df(x)

        e = 0.8;

        y = 1 - e*cos(x);

endfunction
```

Newton's method is used to solve this equation

OCTAVE code (newtons.m)

```
function [zero, res, itr] = newtons(f, df, x0, tol, nmax)

        itr = 0;

        res = abs(x0);


        while ( res >=  tol  && itr  <= nmax )

                x = x0 - (f(x0)/df(x0));

                res = abs(x0-x);
```

```
        x0 = x;
        itr = itr +1;
    end
    if itr > nmax
        fprintf('Newtons method stopped without convergence');
    end
    zero = x;
    endfunction
```

<u>OCTAVE code (q5.m)</u>

```
x0 = 4;
tol = 10^(-8);
nmax = 100;
[zero, res, itr] = newtons(@f,@df,x0,tol,nmax)
```

<u>Output of the code</u>

```
>> q5
zero =  3.0629
res =  0.0000000040056
itr =  4
```

Therefore, angle E = 3.0629 rad.

Q6. State Equation of a Gas

The Van der Waals equation can be rewritten as,

$$f(V) = pV^3 - (Nbp + kNT)V^2 + (aN^2)V - abN^3$$

Implementation of this function in OCTAVE,

OCTAVE Code (f.m)

```
function y = f(x)
    p=3.5*(10^7);
    a=0.401;
    N=1000;
    b=42.7*10^(-6);
    k=1.3806503*(10^(-23));
    T=300;
    y = p*(x.^3) + a*(N^2)*x - a*b*(N^3) -(N*b*p+k*N*T)*x.^2;
endfunction
```

OCTAVE Code (bisection.m)

```
function [zero, res, niter] = bisection(f,a,b,tol,nmax)
    x = [a (a+b)/2 b];
    y = f(x);
    niter = 0;
    I = (b-a)/2;
    if y(1)*y(3)>0
        error('The signs of the function at the extrema must be opposite');
    elseif y(1) == 0
        zero = a; res = 0; return
```

```
        elseif y(3) == 0
                zero = b; res = 0; return
        end
while ( I >= tol && niter <= nmax )
        if sign(y(1))*sign(y(2))<0
                x(3) = x(2); x(2) = (x(1) + x(3))/2;
                y = f(x); I = (x(3)-x(1))/2;
        elseif sign(y(2))*sign(y(3))<0
                x(1) = x(2); x(2) = (x(1) + x(3))/2;
                y = f(x); I = (x(3)-x(1))/2;
        else
                x(2) = x(find(y ==0)); I = 0;
        end
                niter = niter+1;
        end
        if niter > nmax
                fprintf('bisection method exited without convergence');
        end
        zero = x(2); res = f(x(2));
        endfunction
```

Bisection method is used to solve f(V) = 0.

OCTAVE code (q6.m)

```
a = 0; b = 1;
 tol = 10^(-12); nmax = 100;
 [zero, res, niter] = bisection(@f,a,b,tol,nmax)
```

Output of the code

```
>> q6

zero =  0.042700

res =  0.00000040785

niter =  39
```

Therefore, the volume = 0.0427m^3