ME 594 – Numerical Methods - HW02

Viral Panchal | Due date: 09/28

'I pledge my honor that I have abided by the stevens honor system'

Numerical methods - HW02
VIRAL PANCHAL

Q.1)

$\sqrt{x} + x^2 = 7$

Using Newton's method:

$f(x) = \sqrt{x} + x^2 - 7$, $x_0 = 7$

$f'(x) = \frac{1}{2\sqrt{x}} + 2x$

$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} = 7 - \frac{(44.6458)}{(14.189)} = 3.85349$

$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 3.85349 - \frac{(9.81243)}{(7.96169)} = 2.62103$

$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)} = 2.62103 - \frac{(1.48879)}{(5.55091)} = 2.35283$

$x_4 = x_3 - \frac{f(x_3)}{f'(x_3)} = 2.35283 - \frac{(0.0697)}{(5.031627)} = 2.3389$

$x_5 = x_4 - \frac{f(x_4)}{f'(x_4)} = (2.3389) - \left(\frac{1.852 \times 10^{-4}}{5.004886}\right) =$

2.33894 //

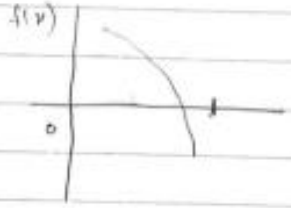
$\therefore x_5 = 2.33894$ // Ans

Q 2) $\cos x - 0.8x^2 = 0$

Positive root by fixed point method:

for fixed point method: $x = g(x)$

Assuming $x_0 = 1$



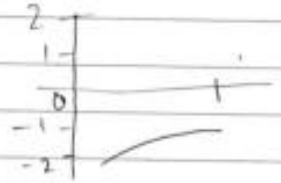
$\therefore 0.8x^2 = \cos x$

$$x = \frac{\cos x}{0.8x}$$

$$\therefore x = 1.25\,\frac{\cos x}{x} = g_1(x)$$

$$g_1'(x) = 1.25\left(\frac{x(-\sin x) - (\cos x \cdot 1)}{x^2}\right)$$

$$= \frac{-1.25}{x^2}\{x\sin x + \cos x\}$$



$|g_1'(x)| > 1$

$\downarrow$

diverging

Alternate solution;

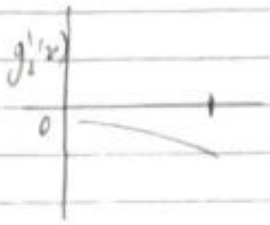
$$x = \sqrt{1.25\cos x} = g_2(x)$$

$$g_2'(x) = \frac{1}{2\sqrt{1.25\cos x}} \cdot (-1.25\sin x) = \frac{-0.625\sin x}{\sqrt{1.25\cos x}}$$

$|g_1'(x)| < 1$

↓

may converge



$x_0 = 1$

$x_1 = \sqrt{1.25\cos(1)} = 0.92181$

$x_2 = \sqrt{1.25\cos(0.92181)} = 0.92256$

$x_3 = \sqrt{1.25\cos(0.92256)} = 0.86875$

$x_4 = \sqrt{1.25\cos(0.86875)} = 0.89846$

$x_5 = \sqrt{1.25\cos(0.89846)} = 0.88234$ //

$x_5 = 0.88234$ // Ans

Q 3) Van der waals equation.

$$P = \frac{nRT}{V - nb} - \frac{n^2 a}{V^2}$$

$R = 0.08206\ L/(mole\ K)$

$n = 1.5\ mol$

$a = 1.49\ L^2\ atm/mol^2$ $\qquad b = 0.03913\ L/mol$

$T = 25^\circ C = 298\ K$

$P = 13.5\ atm$

Based on the Matlab plot:

roots $\in (2, 3)$ ... {plot attached below}

i) Bisection method
ii) Secant metho. } {Matlab program attached}

Q3. Programs

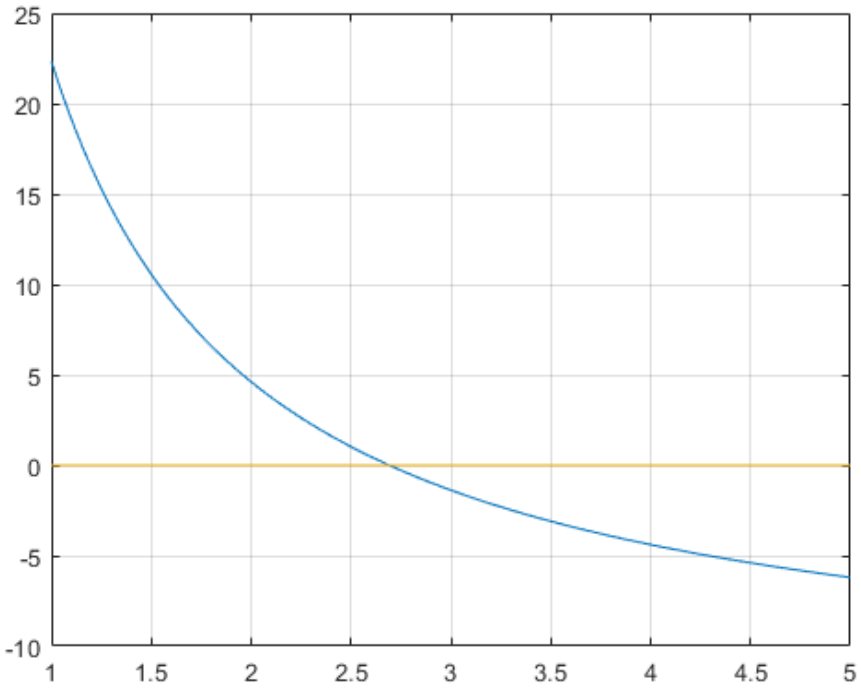
- *Van der waal's function:*

```
% Q3 - Van Der Waal's equation

function f = VDW_f(V)

R = 0.08206;
n = 1.5;
a = 1.39;
b = 0.03913;
T = 298;
p = 13.5;

f = ((n*R*T)./(V-(n*b)))-(((n^2)*a)./(V.^2)) - p;
```

- *Driver for plotting the above function:*

```
% plotting van der waal's function
% To determine the range in which the roots belong to.

x = linspace(1,5,100);
y = VDW_f(x);
z = zeros(100);
plot(x,y,x,z)
grid on
```

- *Matlab Output:*



Published with MATLAB® R2021a

- *Function for Bisection method:*

```
% Making function for bisection method

function [c,error,f_c] = VDW_bisection(f,a,b,delta)

% f = function
% delta = tolerance
% c = root
% f_c = f(c)

f_a = feval(f,a);
f_b = feval(f,b);

while(f_a*f_b) > 0
    break
end
fprintf('  i          a           c           b           f(c) \n')
N = floor((log(b - a) - log(delta))/(log(2)));
for i = 1:N
    c = (a+b)/2;
    f_c = feval(f,c);
    fprintf('%3i %11.6f %11.6f %11.6f % 11.6f\n', i,a,c,b,f_c)
    if f_c == 0
        a = c;
        b = c;
    elseif f_c*f_b>0
        b = c;
        f_b = f_c;
    else
        a = c;
        f_a = f_c;
    end
end

fprintf('\n');
c = (a+b)/2;
error = abs(b-a);
f_c = feval(f,c);
```

- *Driver for the above function:*

```
% Q3.1
% driver for bisection method

a = 2;
b = 3;

% delta = tolerance

delta = 10^(-6);
format long
[V,error,f_V] = VDW_bisection('VDW_f',a,b,delta)
```

- *Matlab Output:*

```
  i         a          c          b         f(c)
  1    2.000000   2.500000   3.000000    1.024687
  2    2.500000   2.750000   3.000000   -0.284173
  3    2.500000   2.625000   2.750000    0.339365
  4    2.625000   2.687500   2.750000    0.020407
  5    2.687500   2.718750   2.750000   -0.133619
  6    2.687500   2.703125   2.718750   -0.057047
  7    2.687500   2.695312   2.703125   -0.018431
  8    2.687500   2.691406   2.695312    0.000960
  9    2.691406   2.693359   2.695312   -0.008742
 10    2.691406   2.692383   2.693359   -0.003893
 11    2.691406   2.691895   2.692383   -0.001467
 12    2.691406   2.691650   2.691895   -0.000253
 13    2.691406   2.691528   2.691650    0.000353
 14    2.691528   2.691589   2.691650    0.000050
 15    2.691589   2.691620   2.691650   -0.000102
 16    2.691589   2.691605   2.691620   -0.000026
 17    2.691589   2.691597   2.691605    0.000012
 18    2.691597   2.691601   2.691605   -0.000007
 19    2.691597   2.691599   2.691601    0.000003


v =

   2.691599845886230


error =

     1.907348632812500e-06


f_v =

    -2.078221672974223e-06
```

Published with MATLAB® R2021a

- *Function for the secant method:*

```
% Making function for secant method

function [P,i,error, rel_error, f_p] = VDW_secant(f,p0,p1,delta,eps,max_i)

% f = function
% p0, p1 = Initial approximated roots
% delta = tolerance within iterations
% eps = tolerance in function
% max_i = maximum iterations
% P = approximated root
% i = iterations
% f_p = f(p)


fprintf('  i        p0          p1        p2           f(c) \n');
for i = 1:max_i
    p2  = p1 - (feval(f,p1)*(p1-p0))/(feval(f,p1) - feval(f,p0));
    fprintf('%3i %11.6f %11.6f %11.6f %11.6f \n',i,p0,p1,p2,feval(f,p2))
    error = abs(p2-p1);
    rel_error = (2*error)/(abs(p2)+abs(p1));
    p0 = p1;
    p1 = p2;
    f_p = feval(f,p1);
    if (error<delta) | (rel_error<delta) | (abs(f_p)<eps)
        break;
    end
end

fprintf('\n')

P = p1;
```

- *Driver for the above function:*

```
%Q 3.2
% driver for secant method

p0 = 2;
p1 = 2.1;
delta = 10^(-6);
eps = 10^(-6);
max_i = 100;

format long
[v,i,error,rel_error,f_v] = VDW_secant('VDW_f',p0,p1,delta,eps,max_i)
```

- *Matlab Output:*

```
  i        p0          p1          p2          f(c)
  1    2.000000    2.100000    2.540842    0.793415
  2    2.100000    2.540842    2.658741    0.165325
  3    2.540842    2.658741    2.689775    0.009076
  4    2.658741    2.689775    2.691577    0.000110
  5    2.689775    2.691577    2.691599    0.000000


v =

   2.691599412939109


i =

     5


error =

     2.207079108806909e-05


rel_error =

     8.199913208204459e-06


f_v =

     7.378801214485975e-08
```

- *Published with MATLAB® R2021a*

Q.4) Quasi - one - dimensional isentropic flow

$$\varepsilon = \frac{1}{M}\left[\frac{2}{\gamma+1}\left(1+\frac{\gamma-1}{2}M^2\right)\right]^{(\gamma+1)/(2(\gamma-1))}$$

$$f(M) = \frac{1}{M}\left[\frac{2}{(\gamma+1)}\left(1+\frac{(\gamma-1)M^2}{2}\right)\right]^{\frac{\gamma+1}{2(\gamma-1)}} - \varepsilon = 0$$

$\varepsilon = 10.0$

$\gamma = 1.4$

To find : $M_1 < 1$ ... Matlab program attached.
& $M_2 > 1$

Q4 – Program

- *Function for Mach number:*

```
% Making a function for Mach number 'M'

function f = Mach_f(M)

eps = 10;
gamma= 1.4;
f = (1/M)*(2/(gamma+1)*(1+(gamma-1)/2*M^2))^((gamma+1)/2/(gamma-1))-eps;
```

- *Function for Regula Falsi Method:*

```
% function for regula falsi method

function [c,error,f_c] = Regula_falsi(f,a,b,delta,eps,max_i)

f_a = feval(f,a);
f_b = feval(f,b);

while f_a*f_b > 0
    fprintf('Breaking since f(a)*f(b)>0 \n')
    break;
end
for i = 1:max_i
    dx = f_b*(b-a)/(f_b - f_a);
    c = b - dx;
    a_c = c - a;
    f_c = feval(f,c);
    fprintf('%3i %11.6f % 11.6f % 11.6f % 11.6f \n', i,a,c,b,f_c)
    if f_c == 0
        break;
    elseif f_b*f_c > 0
        b = c;
        f_b = f_c;
    else
        a = c;
        f_a = f_c;
    end

    dx = min(abs(dx),a_c);
    if (abs(dx) < delta)
        fprintf('Difference between the iterates is less than the tolerance
\n')
        break;
    end
end
error = abs(b-a/2);
f_c = feval(f,c);
end
```

- *Driver for the above function:*

```
% Q4
% Driver for Regula falsi fuunction

a = 10^(-2);
b = 1;
delta = 10^(-6);
eps = 10^(-8);
max_i = 1000;

[M,error,f_M] = Regula_falsi('Mach_f',a,b,delta,eps,max_i)

fprintf('\n ********************************************** \n\n');
a = 1;
b = 5;

[M,error,f_M] = Regula_falsi('Mach_f',a,b,delta,eps,max_i)
```

- *Matlab Outout:*

1	0.010000	0.843337	1.000000	-8.977340
2	0.010000	0.711746	0.843337	-8.913905
3	0.010000	0.601593	0.711746	-8.813677
4	0.010000	0.509614	0.601593	-8.678127
5	0.010000	0.432946	0.509614	-8.507300
6	0.010000	0.369128	0.432946	-8.300547
7	0.010000	0.316062	0.369128	-8.057066
8	0.010000	0.271972	0.316062	-7.776358
9	0.010000	0.235365	0.271972	-7.458620
10	0.010000	0.204987	0.235365	-7.105098
11	0.010000	0.179788	0.204987	-6.718359
12	0.010000	0.158893	0.179788	-6.302458
13	0.010000	0.141572	0.158893	-5.862950
14	0.010000	0.127217	0.141572	-5.406733
15	0.010000	0.115322	0.127217	-4.941703
16	0.010000	0.105468	0.115322	-4.476273
17	0.010000	0.097305	0.105468	-4.018806
18	0.010000	0.090543	0.097305	-3.577051
19	0.010000	0.084944	0.090543	-3.157667
20	0.010000	0.080306	0.084944	-2.765880
21	0.010000	0.076466	0.080306	-2.405331
22	0.010000	0.073287	0.076466	-2.078079
23	0.010000	0.070654	0.073287	-1.784745
24	0.010000	0.068474	0.070654	-1.524748
25	0.010000	0.066669	0.068474	-1.296578
26	0.010000	0.065175	0.066669	-1.098079
27	0.010000	0.063938	0.065175	-0.926700
28	0.010000	0.062913	0.063938	-0.779702
29	0.010000	0.062065	0.062913	-0.654324
30	0.010000	0.061363	0.062065	-0.547897
31	0.010000	0.060782	0.061363	-0.457925
32	0.010000	0.060301	0.060782	-0.382125
33	0.010000	0.059903	0.060301	-0.318449
34	0.010000	0.059573	0.059903	-0.265089
35	0.010000	0.059300	0.059573	-0.220465
36	0.010000	0.059074	0.059300	-0.183210
37	0.010000	0.058887	0.059074	-0.152152
38	0.010000	0.058732	0.058887	-0.126291
39	0.010000	0.058604	0.058732	-0.104778
40	0.010000	0.058498	0.058604	-0.086897
41	0.010000	0.058410	0.058498	-0.072045
42	0.010000	0.058337	0.058410	-0.059716
43	0.010000	0.058277	0.058337	-0.049487
44	0.010000	0.058227	0.058277	-0.041002
45	0.010000	0.058186	0.058227	-0.033967
46	0.010000	0.058151	0.058186	-0.028136
47	0.010000	0.058123	0.058151	-0.023303
48	0.010000	0.058100	0.058123	-0.019299
49	0.010000	0.058080	0.058100	-0.015982
50	0.010000	0.058064	0.058080	-0.013234
51	0.010000	0.058051	0.058064	-0.010958

```
 52    0.010000    0.058040    0.058051   -0.009073
 53    0.010000    0.058031    0.058040   -0.007512
 54    0.010000    0.058023    0.058031   -0.006220
 55    0.010000    0.058017    0.058023   -0.005150
 56    0.010000    0.058012    0.058017   -0.004263
 57    0.010000    0.058008    0.058012   -0.003530
 58    0.010000    0.058004    0.058008   -0.002922
 59    0.010000    0.058001    0.058004   -0.002419
 60    0.010000    0.057999    0.058001   -0.002003
 61    0.010000    0.057997    0.057999   -0.001658
 62    0.010000    0.057995    0.057997   -0.001373
 63    0.010000    0.057994    0.057995   -0.001136
 64    0.010000    0.057993    0.057994   -0.000941
 65    0.010000    0.057992    0.057993   -0.000779
Difference between the iterates is less than the tolerance

M =

   0.057991737440885


error =

   0.052991737440885


f_M =

    -7.787717569609498e-04


 ************************************************

  1    1.000000    2.500000    5.000000   -7.363281
  2    2.500000    3.323144    5.000000   -4.247229
  3    3.323144    3.693171    5.000000   -1.881975
  4    3.693171    3.838854    5.000000   -0.727969
  5    3.838854    3.892598    5.000000   -0.266217
  6    3.892598    3.911909    5.000000   -0.095327
  7    3.911909    3.918780    5.000000   -0.033876
  8    3.918780    3.921216    5.000000   -0.012006
  9    3.921216    3.922079    5.000000   -0.004251
 10    3.922079    3.922385    5.000000   -0.001504
 11    3.922385    3.922493    5.000000   -0.000532
 12    3.922493    3.922531    5.000000   -0.000188
 13    3.922531    3.922544    5.000000   -0.000067
 14    3.922544    3.922549    5.000000   -0.000024
 15    3.922549    3.922551    5.000000   -0.000008
 16    3.922551    3.922551    5.000000   -0.000003
Difference between the iterates is less than the tolerance

M =

   3.922551492461081
```

```
error =

   3.038724253769460


f_M =

    -2.954691659340369e-06
```

Published with MATLAB® R2021a