**Problem Set 5**

**CHE 310 Spring 2019**

**Problem 1**

**Code:**

% Coded by Nigel F. Reuel on 10.6.2016

% This code solves question 5.8 from the text.

%

% Part (a) from pg. 146 of text

display('Number of iterations needed:')

n = ceil(log2(35/0.05)) % NOTE: Ceil is needed as you should round UP

%

% Part (b) call bisect method embedded below for solution

% Copy the monster equation:

F1 = @(T,OxyConc) exp(-139.34411 + 1.575701\*10^5/(T+273.15)...

 -6.642308\*10^7/(T+273.15)^2+1.243800\*10^10/(T+273.15)^3 ...

 -8.621949\*10^11/(T+273.15)^4)-OxyConc;

xl = 0;

xu = 35;

O\_vec = [8 10 14];

NS = length(O\_vec);

T\_vec = zeros(1,NS);

maxit = n;

for i = 1:NS

 OxyConc = O\_vec(i);

 [root,~,~,~]=bisect(F1,xl,xu,[],maxit,OxyConc);

 T\_vec(1,i) = root;

end

% Plot the function first

F2 = @(T) exp(-139.34411 + 1.575701\*10^5/(T+273.15)...

 -6.642308\*10^7/(T+273.15)^2+1.243800\*10^10/(T+273.15)^3 ...

 -8.621949\*10^11/(T+273.15)^4);

fplot(F2,[0 35])

hold on

% Plot the fit points on top of the full curve

plot(T\_vec,O\_vec,'o')

% Label the plot

xlabel('Temperature (\circC)')

ylabel('Dissolved Oxygen Conc (mg/L)')

function [root,fx,ea,iter]=bisect(func,xl,xu,es,maxit,varargin)

% bisect: root location zeroes

% [root,fx,ea,iter]=bisect(func,xl,xu,es,maxit,p1,p2,...):

% uses bisection method to find the root of func

% input:

% func = name of function

% xl, xu = lower and upper guesses

% es = desired relative error (default = 0.0001%)

% maxit = maximum allowable iterations (default = 50)

% p1,p2,... = additional parameters used by func

% output:

% root = real root

% fx = function value at root

% ea = approximate relative error (%)

% iter = number of iterations

if nargin<3,error('at least 3 input arguments required'),end

test = func(xl,varargin{:})\*func(xu,varargin{:});

if test>0,error('no sign change'),end

if nargin<4|isempty(es), es=0.0001;end

if nargin<5|isempty(maxit), maxit=50;end

iter = 0; xr = xl; ea = 100;

while (1)

 xrold = xr;

 xr = (xl + xu)/2;

 iter = iter + 1;

 if xr ~= 0,ea = abs((xr - xrold)/xr) \* 100;end

 test = func(xl,varargin{:})\*func(xr,varargin{:});

 if test < 0

 xu = xr;

 elseif test > 0

 xl = xr;

 else

 ea = 0;

 end

 if ea <= es | iter >= maxit,break,end

end

root = xr; fx = func(xr, varargin{:});

end

**Output:**

1. **Number of iterations n = 10**

****

**Problem 2**

**Code:**

% Coded by Nigel F. Reuel on 10.6.2016

% This solves problem 5.10 from the text

%

% Function as written in book:

%F1 = @(y) 1-Q^2/(g\*Ac^3)\*B

% Function with subsitution:

g = 9.81;

Q = 20;

F1 = @(y) 1-Q^2/(g\*(3\*y+y^2/2)^3)\*(3+y);

% PART(a) Solve by the graphical method

fplot(F1,[1 2.5])

xlabel('Critical Depth')

ylabel('f(y)')

% From the plot the root is about 1.514m

disp('From the plot the root is about 1.514m')

% PART(b) use bisection

es = 1;

maxit = 10;

display('Results from bisect method')

[root,~,ea\_b,iter\_b]=bisect(F1,0.5,2.5,es,maxit)

% Part (c) use false position

display('Results from false position method')

[root,~,ea\_c,iter\_c]=falsepos(F1,0.5,2.5,es,maxit)

%

display('The bisection method is more efficeint for this problem as it converged before the maxium iterations.')

**Output:**

A) From the plot the root is about 1.514 m



B) Results from bisect method

root = 1.5078

ea\_b = 0.5181 (approximate error)

iter\_b = 8 (iterations)

C) Results from false position method

 root = 2.0908

ea\_c = 1.5914 (approximate error)

iter\_c = 10 (iterations)

The bisection method is more efficeint for this problem as it converged before the maxium iterations.

**Problem 3**

**Code:**

% Coded by Nigel F. Reuel on 10.6.2016

% This code solves problem 5.11 from the textbook.

%

% Define the given function

So = 8; %moles/L

vm = 0.7; %moles/L/d

ks = 2.5; %moles/L

F1 = @(t,S) So-vm\*t+ks\*log(So/S)-S;

% This questions is open ended as to which solution method to use. I am

% going to demonstrate the utility of the built in matlab function - fzero.

%

% We have a choice of using either S or t as our defined variable and

% solving for the other. In this case, I don't know how long the enzyme

% reaction will go, but I do know that the substrate will be consumed in an

% enzymatic reaction, so let's create a vector spanning from 0.01 of the

% Substrate (99% consumed) to the starting substrate concentration.

%

S\_vec = linspace(0.01\*So,So,100);

t\_vec = zeros(1,100);

to = 1;

for i = 1:100

 S = S\_vec(1,i);

 t\_solve = fzero(F1,to,[],S);

 t\_vec(1,i) = t\_solve;

end

plot(t\_vec,S\_vec)

xlim([0 max(t\_vec)])

xlabel('Time (d)')

ylabel('Substrate Concentration (moles/L)')

text(10,6,['Substrate is 99% consumed at ',num2str(max(t\_vec)),' days'])

**Output:**

****

**Problem 4**

**Code:**

% Coded by Nigel F. Reuel on 10.6.2016

% This code solves problem 6.15 from the text

%

% NOTE: This one time solution is probably most easily solved using the

% Excel solver tool, but I would like to demonstrate how fzero can be used

% on a 'function' that is actually a subfunction. Let's see how this goes:

%

% Define a subfunction for the Redlich Kwong equation and move the pressure

% to right hand side of the equation so we can use a root finding method

% (solve for zero)

%

% Use fzero for simplicity (because I don't know where the answer is)

% pass T and P as extra parameters:

clc

T = 273.15-40; %K

P = 65000\*1000; %Pa

vo = 0.0001; % initial guess for molar volume of gas

Root = fzero(@RK,vo,[],T,P); % <-- Note, when passing extra variables you have to use this @ notation with the function name

% Convert molar volume (m^3/mol) to Vm (m^3/kg) with molar mass of methane (16.04 g/mol):

Vm = Root/16.04\*1000;

Mass = 3/Vm;

% Print the answers to the screen with units:

fprintf('For the given conditions Vm = %fm^3/kg\n',Vm)

fprintf('The mass of fuel in a 3m^3 tank is %5.0fkg\n',Mass)

function OUT = RK(v,T,P)

Tc = 191; %K

Pc = 4600\*1000; %Pa

R = 8.314; %m^3\*Pa/K/mol

a = 0.427\*R^2\*Tc^2.5/Pc;

b = 0.0866\*R\*Tc/Pc;

OUT = R\*T/(v-b)-a/(v\*(v+b)\*(T)^(1/2))-P;

end

**Output:**

**For the given conditions Vm = 0.002810m^3/kg**

**The mass of fuel in a 3m^3 tank is 1068kg**

**Problem 5**

**Code:**

% This code solves problem 6.3 from the textbook

% Coded by Nigel F. Reuel on 10.6.2016

%

% Determine highest real root of following function

F1 = @(x) x^3-6\*x^2+11\*x-6.1;

% Part(a) Graphically

fplot(F1,[0 4])

line([0 4], [0 0])

disp('From inspection the highest root is at ~3.0467')

% Part (b)

D1 = @(x) 3\*x^2-12\*x+11;

xr = 3.5;

[root,~,~]=newtraph(F1,D1,xr,[],3);

disp('Root from newton-raphson method is:')

root

% Part (c) Secant Method

x0 = 2.5;

xr = 3.5;

[root,~,~]=secantmethod(F1,xr,x0,[],3);

disp('Root from secant method is:')

root

% Part (d) Modified Secant Method

xr = 3.5;

per = 0.01;

[root,~,~]=modsecantmethod(F1,xr,per,[],3);

disp('Root from modified secant method is:')

root

% Part (e) Roots function from MATLAB

p = [1 -6 11 -6.1];

R = roots(p);

disp('Maxium Root from built in matlab function:')

max(R)

**Output:**

**A) From inspection the highest root is at ~3.0467**

****

**B) Root from newton-raphson method is 3.0473**

**C) Root from secant method is 3.1687**

**D) Root from modified secant method is 3.0488**

**E) Maximum root from build in matlab function is 3.0467**