Assignment 2 Solution

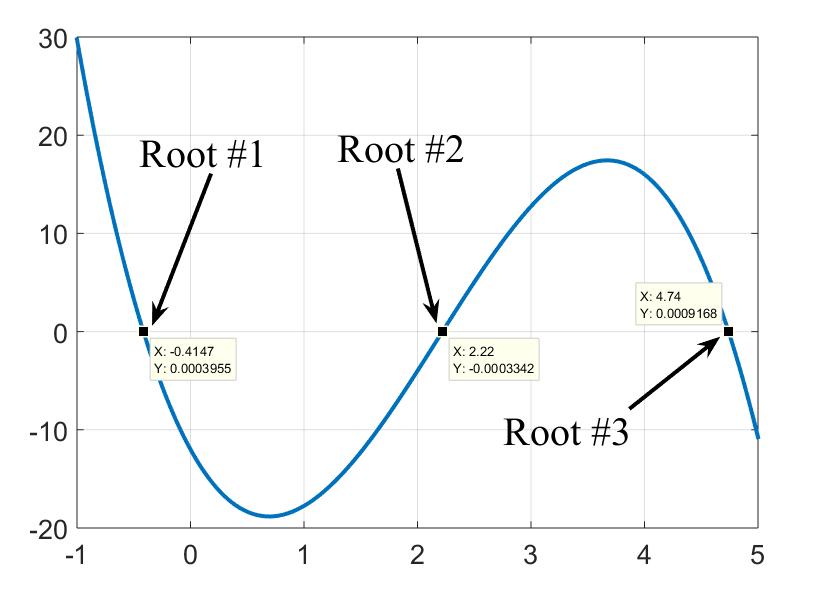
ECE 2412 - Simulation and Engineering Analysis

Haider Mohomad AR

Problem 5.5:

1. Graphical method

|  |
| --- |
| %% Problem 5.5  clear;clc;close all    % Part (a)  x=-1:0.0001:5;  f\_x = -12 - 21\*x + 18\*x.^2 - 2.75\*x.^3;  plot(x,f\_x)  grid on |



(b) Bisection Method

|  |
| --- |
| %% Part (b) Bisection Method  clear;clc;close all  a=-1;  b=0;  E\_s = 1/100; % The tolerance  E\_a = 1 + E\_s; % initial value of the approximated error must be greater than E\_s  x\_r\_old = a; % initial value of x\_r\_old = a or b  count=0; %count will be used to count the number of iterations (optional)    f\_a = -12 - 21\*a + 18\*a.^2 - 2.75\*a.^3;  f\_b = -12 - 21\*b + 18\*b.^2 - 2.75\*b.^3;  % More efficent way is to replace f\_a and f\_b by MATALB function that takes  % the variable x as input and return f(x) as output.    % To check for change in sign in the interval (a,b)  if ( f\_a\*f\_b <0) % There is change in sign    while(E\_a>E\_s)  x\_r = (a + b) / 2; % new estimate of the root  f\_x\_r = -12 - 21\*x\_r + 18\*x\_r.^2 - 2.75\*x\_r.^3; % f(xr)    if(f\_a\*f\_x\_r < 0) % the root is located between a and x\_r  b=x\_r;  f\_b=f\_x\_r;    else % the root is located between x\_r and b  a=x\_r;  f\_a=f\_x\_r;    end  %calculating the approximated error  E\_a = abs((x\_r - x\_r\_old) / x\_r) \*100;  % updating x\_r\_old  x\_r\_old=x\_r;  count=count+1;    end    else % no change in sign between (a,b)  error('F(a) and f(b) must have different sign')    end    fprintf(' The estimated root is %0.8f with relative approximated error of %0.8f%% using the bisection method (%0.0f iterations)\n',x\_r,E\_a,count) |

>> The estimated root is -0.41470337 with relative approximated error of 0.00735889% using the bisection method (15 iterations)

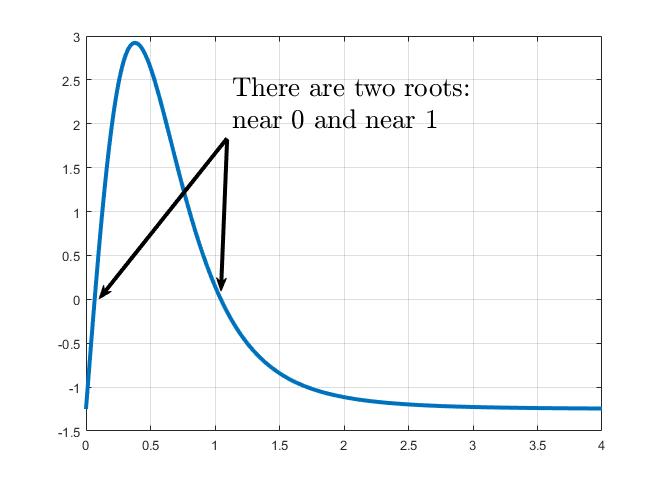
(b) False Position Method

|  |
| --- |
| %% (c) False Position  clear;clc;close all;  a=-1;  b=0;  E\_s = 1/100; % The tolerance  E\_a = 1 + E\_s; % initial value of the approximated error must be greater than E\_s  x\_r\_old = a; % initial value of x\_r\_old = a or b  count=0; %count will be used to count the number of iterations (optional)    f\_a = -12 - 21\*a + 18\*a.^2 - 2.75\*a.^3;  f\_b = -12 - 21\*b + 18\*b.^2 - 2.75\*b.^3;  % More efficent way is to replace f\_a and f\_b by MATALB function that takes  % the variable x as input and return f(x) as output.    % To check for change in sign in the interval (a,b)  if ( f\_a\*f\_b <0) % There is change in sign    while(E\_a>E\_s)  x\_r = b - ( (f\_b\*(a-b))/(f\_a-f\_b)); % new estimate of the root  f\_x\_r = -12 - 21\*x\_r + 18\*x\_r.^2 - 2.75\*x\_r.^3; % f(xr)    if(f\_a\*f\_x\_r < 0) % the root is located between a and x\_r  b=x\_r;  f\_b=f\_x\_r;    else % the root is located between x\_r and b  a=x\_r;  f\_a=f\_x\_r;    end    E\_a = abs((x\_r - x\_r\_old) / x\_r) \*100; %calculating the approximated error  x\_r\_old=x\_r; % updating x\_r\_old  count=count+1;  end    else % no change in sign between (a,b)  error('F(a) and f(b) must have different sign')    end    fprintf(' The estimated root is %0.8f with relative approximated error of %0.8f%% using the false position method (%0.0f iterations)\n',x\_r,E\_a,count) |

>> The estimated root is -0.41467695 with relative approximated error of 0.00832633% using the false position method (8 iterations)

Problem 5.17

|  |
| --- |
| %% Problem 5\_17  clear;clc;close all    % values of the constants  e0=8.9e-12;  Q=2e-5;  q=2e-5;  F=1.25;  a=0.85;    % f\_x = (1/(4\*pi\*e0)\*((q\*Q\*x)/((x^2+a^2)^3/2))-F    % Graphical method  x=0:0.01:4  % calculate f\_x as a vector of values to be plotted  f\_x = (1/(4\*pi\*e0)).\*((q.\*Q.\*x)./((x.^2+a.^2).^(3/2)))-F;  plot(x,f\_x)  grid on    % You can use any method to solve for the root, I will use the MATLAB  % function fzero()  % defined the function to be used in the fzero function  fx = @(x) (1/(4\*pi\*e0))\*((q\*Q\*x)./((x^2+a^2)^3/2))-F;    % solve for the root xr and its value fval using initial guess of xi  % NOTE that the initial values of xi were determined from the graph  xi=0;  [xr1,fval1]=fzero(fx,xi);    xi=1;  [xr2,fval2]=fzero(fx,xi);    fprintf('The distance where the force is 1.25N = %0.8f m and %0.8f m \n',xr1,xr2) |



>> The distance where the force is 1.25N = 0.24104274 m and 1.29127956 m

Problem 6.1

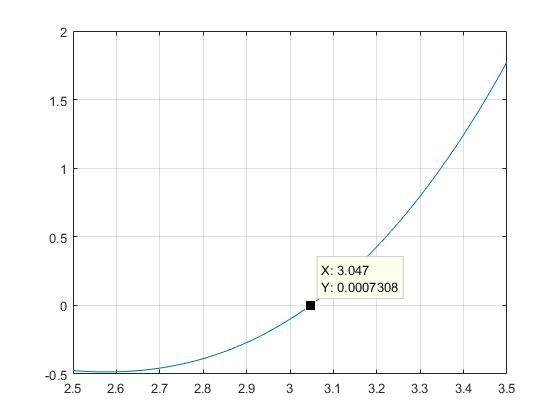
|  |
| --- |
| %% Problem 6.1  clear;clc;close all;    x\_old=0.5; % initial guess  E\_s = 0.01; % The tolerance  E\_a = 1 + E\_s; % initial value of the approximated error must be greater than E\_s  count=0; %count will be used to count the number of iterations (optional)    % Modified the function f(x) ------> x = f(x)+ x ----> x = sin (sqrt(x))      while(E\_a>E\_s)    x\_new = sin ( sqrt(x\_old)); % estimate the new root    E\_a = abs((x\_new - x\_old) / x\_new) \*100; %calculating the approximated error  x\_old=x\_new; % updating x\_old  count=count+1;  end      fprintf(' The estimated root is %0.8f with relative approximated error of %0.8f%% using the fixed point method (%0.0f iterations)\n',x\_new,E\_a,count) |

>> The estimated root is 0.76860623 with relative approximated error of 0.00965506% using the fixed point method (9 iterations)

Problem 6.3

1. Graphical Method

|  |
| --- |
| %(a) Graphically  clear;clc;close all;  x = 2.5:0.001:3.5;  f\_x = (x.^3) - (6.\*x.^2) + (11.\*x) - 6.1;  plot(x,f\_x)  grid on |



1. Newton-Raphson method

|  |
| --- |
| %% (b) Newton-Raphson method  clear;clc;close all;  xi=3.5; % initial guess  n = 3; % 3 iterations    % For Newton-Raphson method, the derivative is required to calculate the  % new estimated root:  % f(x) = x^3 - 6x^2 + 11x - 6.1  % df\_dx(x) = 3x^2 - 12x + 11      for i=1:1:n %start the iteration until i=n    f\_xi = xi^3 - 6\*xi^2 + 11\*xi - 6.1; % calculating f(xi)  df\_dx = 3\*xi^2 - 12\*xi + 11; %calculating f'(xi)  x\_i\_new = xi - ((f\_xi)/(df\_dx)); % estimating the new root  xi= x\_i\_new; %updating xi;    end    fprintf(' The estimated root is %0.8f after %d iterations using the Newton-Raphson method. \n',x\_i\_new,n) |

>> The estimated root is 3.04731674 after 3 iterations using the Newton-Raphson method.

1. Secant method

|  |
| --- |
| %% (c) Secant method  clear;clc;close all;  xi=3.5; % initial guess at x\_i  xi\_1 = 2.5; % initial guess at x\_i-1  n = 3; % 3 iterations    for i=1:1:n %start the iteration until i=n    f\_xi = xi^3 - 6\*xi^2 + 11\*xi - 6.1; % calculating f(xi)  f\_xi\_1 = xi\_1^3 - 6\*xi\_1^2 + 11\*xi\_1 - 6.1; % calculating f(xi-1)  x\_i\_new = xi - ((f\_xi)\*(xi\_1-xi)/(f\_xi\_1-f\_xi)); % estimating the new root  xi\_1=xi; %updating xi-1  xi= x\_i\_new; %updating xi;    end    fprintf(' The estimated root is %0.8f after %d iterations using the Secant method. \n',x\_i\_new,n) |

>> The estimated root is 3.22192345 after 3 iterations using the Secant method.

1. Modified Secant method

|  |
| --- |
| %% (d) Modified Secant method  clear;clc;close all;  xi=3.5; % initial guess at x\_i  delta = 0.01; % initial guess at x\_i-1  n = 3; % 3 iterations    for i=1:1:n %start the iteration until i=n    f\_xi = xi^3 - 6\*xi^2 + 11\*xi - 6.1; % calculating f(xi)  f\_delta\_xi = (xi+delta\*xi)^3 - 6\*(xi+delta\*xi)^2 + 11\*(xi+delta\*xi) - 6.1; % calculating f(xi)  x\_i\_new = xi - ((f\_xi)\*(delta\*xi)/(f\_delta\_xi-f\_xi)); % estimating the new root  xi= x\_i\_new; %updating xi;    end    fprintf(' The estimated root is %0.8f after %d iterations using the Modified Secant method. \n',x\_i\_new,n) |

>> The estimated root is 3.04881823 after 3 iterations using the Modified Secant method.

1. All roots using MATLAB

|  |
| --- |
| %% (e) All roots using matlab  clear;clc;close all;  % represents f(x) as a vector of the coefficients of polynomial  f\_x = [ 1 -6 11 -6.1];  % use matlab function roots()  f\_x\_roots = roots(f\_x) |

>> f\_x\_roots =

3.0467

1.8990

1.0544

Problem 6.24

|  |
| --- |
| %% Problem 6.24  clear; clc; close all    % Since C(s) and N(s) are polynomials, they can be represented in MATLAB as  % a vector of coefficients.    C = [1 9 26 24];  N = [1 15 77 153 90];    % finding the roots of the numerator C(s)  roots\_num = roots(C);    % finding the roots of the denominator N(s)  roots\_den = roots(N);    fprintf('The numerator C(s) factors are (s%0.3f)(s%0.3f)(s%0.3f) \n',roots\_num(1),roots\_num(2),roots\_num(3))  fprintf('The denominator N(s) factors are (s%0.3f)(s%0.3f)(s%0.3f)(s%0.3f) \n',roots\_den(1),roots\_den(2),roots\_den(3),roots\_den(4)) |

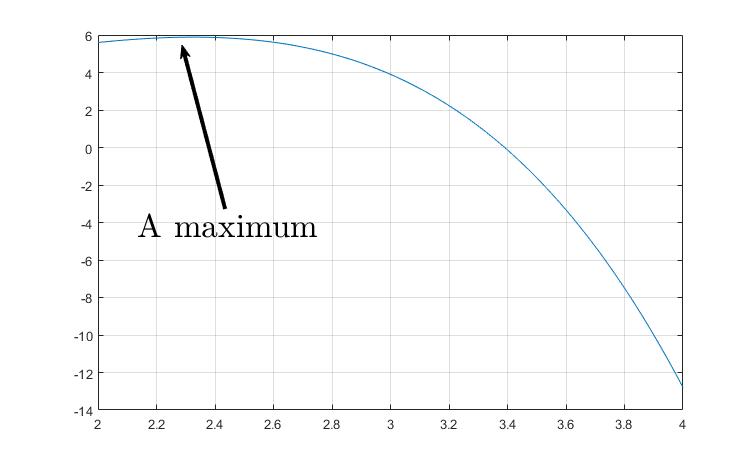
>> The numerator C(s) factors are (s-4.000)(s-3.000)(s-2.000)

The denominator N(s) factors are (s-6.000)(s-5.000)(s-3.000)(s-1.000)

Problem 7.7

Start with graphical solution in order to have an in-sight of the curve

|  |
| --- |
| %% Problem 7.7  clear;clc;close all;  x=2:0.01:4;  fx = 4\*x - 1.8\*x.^2 + 1.2\*x.^3 - 0.3\*x.^4;  plot(x,fx)  grid on |



From the graph, the expected maximum location is near x=2.3 and its value f(x) ≈ 6

1. The golden-section method

|  |
| --- |
| clear;clc;  a=2; b=4;  E\_s = 1/100; % The tolerance  E\_a = 1 + E\_s; % initial value of the approximated error > E\_s  count=0; %count will be used to count the number of iterations (optional)  phai = 1.6180339887; % The golden ratio (constant)  phai\_1 = (phai-1); % calculate the constant outside the loop    % calculate d  d = phai\_1\*(b-a);  % calculate the two points x1 and x2  x1 = b - d;  x2 = a + d;  while(E\_a>E\_s)  %calculate f(x1) and f(x2)  fx1 = -(4\*x1 - 1.8\*x1^2 + 1.2\*x1^3 - 0.3\*x1^4);  fx2 = -(4\*x2 - 1.8\*x2^2 + 1.2\*x2^3 - 0.3\*x2^4);    % select the Xopt based on the comparison of f(x1) and f(x2)  if(fx1 < fx2) % the true root is located between a and x2  xopt=x1; % estimate of the maximum  fxopt=fx1;  b=x2; % update the interval side  x2=x1; %update x2  d= phai\_1\*(b-a); %update d  x1 = b - d; %calculate the new x1    else % the root is located between x\_r and b  xopt=x2; % estimate of the maximum  fxopt=fx2;  a=x1; % update the interval side  x1=x2; %update x1  d= phai\_1\*(b-a); %update d  x2 = a + d; %calculate the new x2  end    E\_a = (2-phai)\*abs((b-a) / xopt) \*100; %calculating the approximated error  count=count+1;    end  fprintf(' Using the golden-section method (%d iterations)\n',count)  fprintf(' The estimated maximum location is %0.8f \n',xopt)  fprintf(' The estimated maximum value is %0.8f \n',-fxopt)  fprintf(' The relative approximated error is %0.8f \n',E\_a) |

>> Using the golden-section method (17 iterations)

The estimated maximum location is 2.32634488

The estimated maximum value is 5.88534005

The relative approximated error is 0.00919583

1. Parabolic interpolation method

|  |
| --- |
| %% (b) Parabolic interpolation  clear;clc;  x1=1.75; %initial value of x1  x2=2; %initial value of x2  x3=2.5; %initial value of x3  n=5; % number of iterations    %evaluating f(x) at x1, x2, x3  fx1 = -(4\*x1 - 1.8\*x1^2 + 1.2\*x1^3 - 0.3\*x1^4);  fx2 = -(4\*x2 - 1.8\*x2^2 + 1.2\*x2^3 - 0.3\*x2^4);  fx3 = -(4\*x3 - 1.8\*x3^2 + 1.2\*x3^3 - 0.3\*x3^4);    for i=1:1:n    % calculate the forth point  num = (x2-x1)^2\*(fx2-fx3)-( (x2-x3)^2\*(fx2-fx1) );  den = (x2-x1)\*(fx2-fx3)-( (x2-x3)\*(fx2-fx1) );  x4 = x2 - 0.5 \* (num) / (den);    %calculate f(x) at x4  fx4 = -(4\*x4 - 1.8\*x4^2 + 1.2\*x4^3 - 0.3\*x4^4);      % select the Xopt based on the comparision of f(x2) and f(x4)    if(fx2 < fx4) % the true root is located between a and x2  xopt=x2; % estimate of the maximum  fxopt=fx2;  x3=x4; % update the interval side    else % the root is located between x\_r and b  xopt=x4; % estimate of the maximum  fxopt=fx4;  x1=x2; % update the interval side  x2=x4;  end  end  fprintf(' Using the parabolic interpolation method (%d iterations)\n',n)  fprintf(' The estimated maximum location is %0.8f \n',xopt)  fprintf(' The estimated maximum value is %0.8f \n',-fxopt) |

>> Using the parabolic interpolation method (5 iterations)

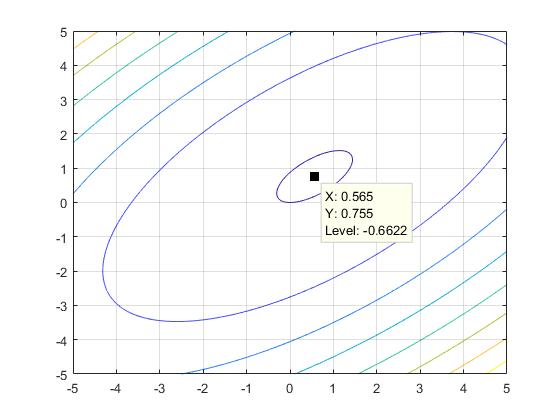
The estimated maximum location is 2.33405797

The estimated maximum value is 5.60000000

Problem 7.23

|  |
| --- |
| %% Problem 7.23  clear; clc;close all    x=-5:0.005:5;  y=-5:0.005:5;  [X,Y]=meshgrid(x,y);  Z=2.\*Y.^2 - 2.25.\*X.\*Y - 1.75.\*Y + 1.5.\*X.^2;  contour(X,Y,Z);  grid on    % from the contour, we can estimate the location of the minimum to be  % around the point (0.5, 0.5) ---> we will use it as initial guess  f=@(x) 2\*x(2)^2 - 2.25\*x(1)\*x(2) - 1.75\*x(2) + 1.5\*x(1)^2;  [x,fval]=fminsearch(f,[0.5 0.5]);    fprintf('The location of the minimum is (%0.4f, %0.4f) with value of %0.8f \n',x(1),x(2),fval) |

>> The location of the minimum is (0.5676, 0.7568) with value of -0.66216216



Problem 7.23

|  |
| --- |
| %% Problem 7.24  clear; clc;close all    x=-3:0.01:3;  y=-3:0.01:3;  [X,Y]=meshgrid(x,y);  Z=-1\*(4\*X + 2\*Y + X.^2 - 2\*X.^4 + 2\*X.\*Y - 3\*Y.^2);  contour(X,Y,Z);  grid on    f=@(x) -1\*(4\*x(1) + 2\*x(2) + x(1)^2 - 2\*x(1)^4 + 2\*x(1)\*x(2) - 3\*x(2)^2);  [x,fval]=fminsearch(f,[1 2]);    fprintf('The location of the maximum is (%0.4f, %0.4f) with value of %0.8f \n',x(1),x(2),-fval) |

>> The location of the maximum is (0.9676, 0.6558) with value of 4.34400579 