Lab Report On

Numerical Analysis Lab

Course Title: Numerical Analysis Lab

Course Code: CSE-232

Submitted To:

Emon Islam Rabbi
Senior Lecturer (CSE Department)

City University, Dhaka.

Submitted By:

MD Faisal Ahmed

ID:152392326

Dept:CSE

Batch:39

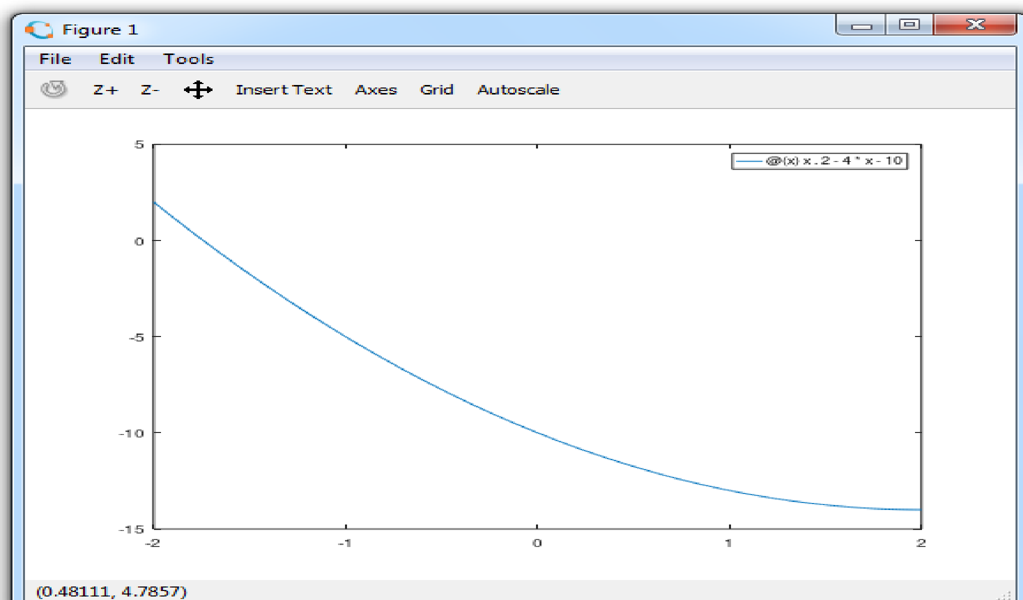
Submission Date:14/1/2020

Bisection Method

bisection.m

```
clc;
close all;
fprintf('Bisection Method\n');
f=@(x)x.^2-4*x-10;
%f=@(x)x.^4-x-10;
%f=@(x) x*log(10^x)-1.2;
fplot(f,[-2,2])
a=5;
b=6;
for i=0:100
  c=(a+b)/2;
  if(f(a)*f(c)>0)
  a=c;
else
  b=c;
end
end
fprintf("The root is %f",c);
fprintf('\nMd Faisal Ahmed \nID:152392326');
```

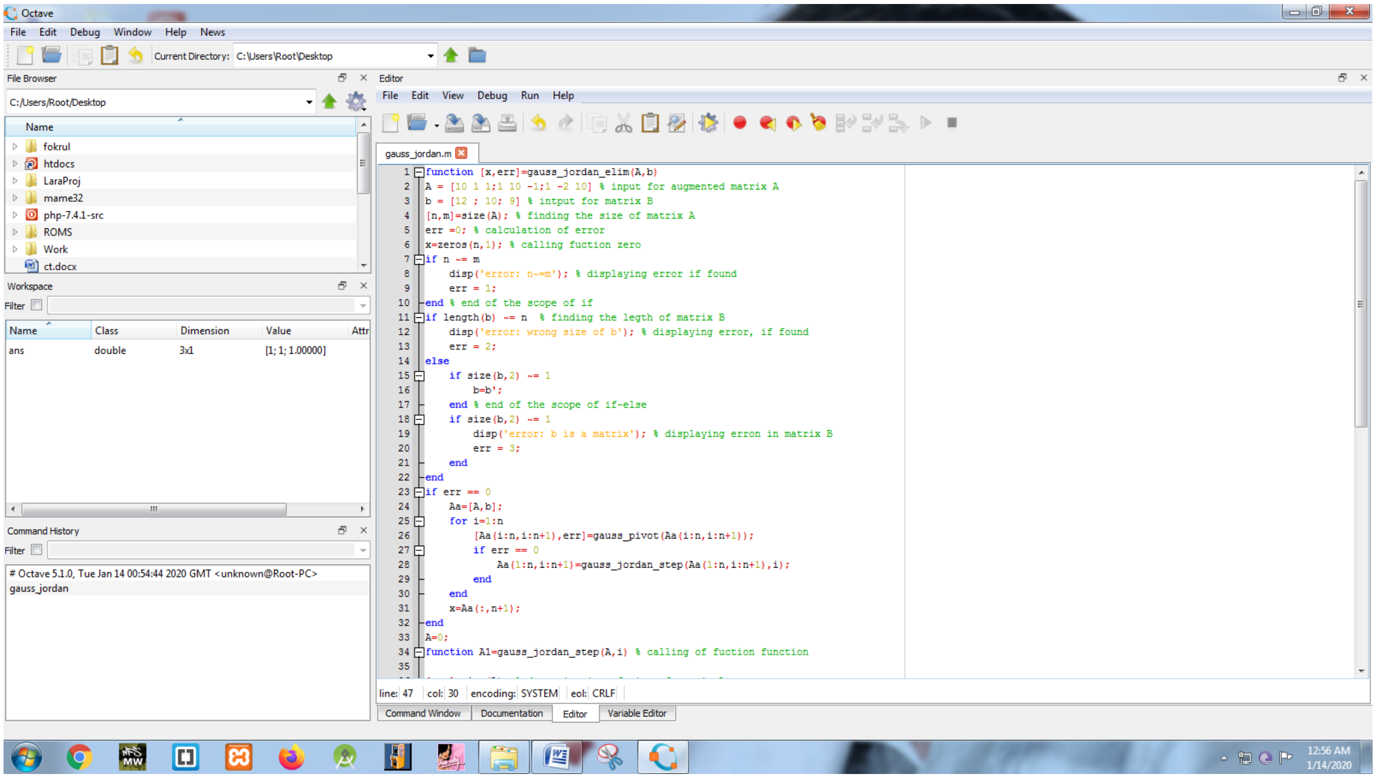
```
Bisection Method
The root is 5.741657
Md Faisal Ahmed
ID:152392326>>
```



False Position Method

```
clc
close all;
fprintf('Falsi Method \n');
f=@(x)x.^2-4*x-10;
%f=&(x)x.^4-x-10;
%f=@(x)3*x-cos(x)-1;
fplot(f,[-2,2])
a=5;
b=6;
for i=0:20
  c=(a*f(b)-b*f(a))/(f(b)-f(a));
  if(f(a)*f(c)>0)
  a=c;
else
  b=c;
  end
end
fprintf("The root is %f",c);
fprintf('\nMd Faisal Ahmed\n ID:152392326');
```

Gauss Jordan Method




```
function A1=gauss_jordan_step(A,i) % calling of fuction function

 [n,m]=size(A); % determination of size of matrix A
A1=A; % assigning A to A1
s=A1(i,1);
A1(i,:) = A(i,:)/s;
k=[[1:i-1],[i+1:n]];
for j=k
    s=A1(j,1);
    A1(j,:)=A1(j,:)-A1(i,:)*s;
end % end of for loop
function [A1,err]=gauss_pivot(A) % calling of fucntion
 [n,m]=size(A); % finding the size of matrix A
A1=A; % process of assigning
err = 0; % error flag
if A1(1,1) == 0
    check = logical(1); % logical(1) - TRUE
    i = 1;
    while check
        i = i + 1;
        if i > n
            disp('error: matrix is singular');
            err = 1;
            check = logical(0);
        else
            if A(i,1) ~= 0 & check
                check = logical(0);
                b=A1(i,:);      % process to change row 1 to i
                A1(i,:)=A1(1,:);
                A1(1,:)=b;
            end
        end
    end
end
```

```
>> gauss_jordan

warning: function name 'gauss_jordan_elim' does not agree with function filename '
esktop\gauss_jordan.m'
A =

   10    1    1
    1   10   -1
    1   -2   10

b =

   12
   10
    9

ans =

   1.00000
   1.00000
   1.00000

>> |
```

Gauss Elimination Method

```
function C = gauss_elimination(A,B) % defining the function
A= [ 1 2; 4 5] % Inputting the value of coefficient matrix
B = [-1; 4] % % Inputting the value of coefficient matrix
    i = 1; % loop variable
    X = [ A B ];
    [ nX mX ] = size( X); % determining the size of matrix
    while i <= nX % start of loop
        if X(i,i) == 0 % checking if the diagonal elements are zero or not
            disp('Diagonal element zero') % displaying the result if there exists zero
            return
        end
        X = elimination(X,i,i); % proceeding forward if diagonal elements are non-zero
        i = i +1;
    end
    C = X(:,mX);

function X = elimination(X,i,j)
% Pivoting (i,j) element of matrix X and eliminating other column

% elements to zero

[ nX mX ] = size( X);
a = X(i,j);
X(i,:) = X(i,:)/a;
for k =  1:nX % loop to find triangular form
    if k == i
        continue
    end
    X(k,:) = X(k,:) - X(i,:)*X(k,j); % final result
end
```

```
>> gauss_elimination

A =

     1     2
     4     5

B =

    -1
     4

ans =

    4.3333
   -2.6667

>>
```

Factorization Method

```
function [L,U,P]=LU_pivot(A)
A = [2 -3 10;-1 4 2;5 2 1 ]
[n,n]=size(A);
L=eye(n); P=L; U=A;
for k=1:n
    [pivot m]=max(abs(U(k:n,k)));
    m=m+k-1;
    if m~=k

        temp=U(k,:);
        U(k,:)=U(m,:);
        U(m,:)=temp;

        temp=P(k,:);
        P(k,:)=P(m,:);
        P(m,:)=temp;
        if k >= 2 %
            temp=L(k,1:k-1);
            L(k,1:k-1)=L(m,1:k-1);
            L(m,1:k-1)=temp;
        end
    end
    for j=k+1:n
        L(j,k)=U(j,k)/U(k,k);
        U(j,:)=U(j,:)-L(j,k)*U(k,:);
    end
end
fprintf('\nMd Faisal Ahmed.\nID:142392326');
```

```
warning: function name 'LU_pivot' does not
ctorization.m'
A =

    2   -3   10
   -1    4    2
    5    2    1


Md Faisal Ahmed.
ID:142392326ans =

   1.00000   0.00000   0.00000
  -0.20000   1.00000   0.00000
   0.40000  -0.86364   1.00000
```

Newton Raphson Method

```
clc;
f=@(x) x.^2-5*x+6;
fplot(f,[-2,8])
df=@(x)2*x-5;
x=0;
for i=0:5
  y=x;
  x=y-f(x)./df(x);
  if(x==y)
  break
  end
end
fprintf('the root is %f',x);
fprintf('\nMd Faisal Ahmed\nID:152392326');
```

```
the root is 2.000000
Md Faisal Ahmed
ID:152392326>>
```

Secant Method

```
function secant_method()
    f = @(x) x^2 - 9;
    eps = 1e-6;
    x0 = 1000;  x1 = x0 - 1;
    [solution,no_iterations] = secant(f, x0, x1, eps);
    if no_iterations > 0   % Solution found
        fprintf('Number of function calls: %d\n', 2 + no_iterations);
        fprintf('A solution is: %f\n', solution)
    else
        fprintf('Abort execution.\n')
    end
end

function [solution,no_iterations] = secant(f, x0, x1, eps)
    f_x0 = f(x0);
    f_x1 = f(x1);
    iteration_counter = 0;
    while abs(f_x1) > eps && iteration_counter < 100
        try
            denominator = (f_x1 - f_x0)/(x1 - x0);
            x = x1 - (f_x1)/denominator;
        catch
            fprintf('Error! - denominator zero for x = \n', x1)
            break
        end
        x0 = x1;
        x1 = x;
        f_x0 = f_x1;
        f_x1 = f(x1);
        iteration_counter = iteration_counter + 1;
    end
    % Here, either a solution is found, or too many iterations
    if abs(f_x1) > eps
        iteration_counter = -1;
    end
    solution = x1;
    no_iterations = iteration_counter;
end
```

```
warning: function name 'secant_method' does not agree with function filename 'C:\Users\Root\Deskt
op\secant.m'
Number of function calls: 19
A solution is: 3.000000
>>
```