Assignment :

“Scientific Computation”

Name: Raheel Javed

Reg. No. : 2009-EE-105

Section : B

Naive Gaussian Elimination:

Given below is the Matlab code for 3x3 system of equations using Naive Gaussian Elimination.

Matlab Code:

```
factor=A(2, 1)/A(1, 1);           % Starting of Forward Elimination
A(2, :)=A(2, :)-factor*A(1, :); % R2=R2-factor*R1
B(2)=B(2)-factor*B(1);

factor=A(3, 1)/A(1, 1);
A(3, :)=A(3, :)-factor*A(1, :); % R3=R3-factor*R1
B(3)=B(3)-factor*B(1);

factor=A(3, 2)/A(2, 2);
A(3, 2)=A(3, 2)-factor*A(2, 2);
A(3, 3)=A(3, 3)-factor*A(2, 3); % R3=R3-factor*R2
B(3)=B(3)-factor*B(2);

x(3)=B(3)/A(3, 3);                %Back Substitution

sum=B(2);
sum=sum-A(2, 3)*X(3);
x(2)=sum/A(2, 2);

sum=B(1);
sum=sum-A(1, 3)*X(3);
sum=sum-A(1, 2)*X(2);
x(1)=sum/A(1, 1);

X=[x(1); x(2); x(3)]              %Output Matrix
```

Problem With Its Output:

$10x_1 + 5x_2 + 6x_3 = 7$

$2x_1 + 5x_2 + 4x_3 = 8$

$x_1 + 3x_2 + 5x_3 = 10$

Define,

```
A=[10, 5, 6; 2, 5, 4; 1, 3, 5];
B=[7, 8, 10];
```

Output of Matlab Code is,

```
X =

    -0.6132

     0.2830
```

```
          1.9528
```

Partial Pivoting Gaussian Elimination:

Given below is the Matlab code for 3x3 system of equations using Partial Pivoting Gaussian Elimination.

Matlab Code :

```
if(abs(A(1, 1))<abs(A(2, 1)))    % Comparing Elements in 1st Column Before
                                   starting operations
    temp=A(1, :);
    temp2=B(1);
    A(1, :)=A(2, :);
    B(1)=B(2);
    A(2, :)=temp;
    B(2)=temp2;
end

if(abs(A(2, 1))<abs(A(3, 1)))
    temp=A(2, :);
    temp2=B(2);
    A(2, :)=A(3, :);
    B(2)=B(3);
    A(3, :)=temp;
    B(3)=temp2;
end

if(abs(A(1, 1))<abs(A(3 , 1)))
    temp=A(1, :);
    temp2=B(1);
    A(1, :)=A(3, :);
    B(1)=B(3);
    A(3, :)=temp;
    B(3)=temp2;
end

factor=A(2, 1)/A(1, 1);
A(2, :)=A(2, :)-factor*A(1, :);
B(2)=B(2)-factor*B(1);

factor=A(3, 1)/A(1, 1);
A(3, :)=A(3, :)-factor*A(1, :);
B(3)=B(3)-factor*B(1);

if(abs(A(2, 2))<abs(A(3, 2)))    % Comparison in 2nd Column after 1st step of
                                   elimination
    temp=A(2, :);
    temp2=B(2);
    A(2, :)=A(3, :);
    B(2)=B(3);
    A(3, :)=temp;
```

```
    B(3)=temp2;
end

factor=A(3, 2)/A(2, 2);
A(3, 2)=A(3, 2)-factor*A(2, 2);
A(3, 3)=A(3, 3)-factor*A(2, 3)
B(3)=B(3)-factor*B(2)

x(3)=B(3)/A(3, 3);

sum=B(2);
sum=sum-A(2, 3)*x(3);
x(2)=sum/A(2, 2);

sum=B(1);
sum=sum-A(1, 3)*x(3);
sum=sum-A(1, 2)*x(2);
x(1)=sum/A(1, 1);

X=[x(1); x(2); x(3)]
```

Problem With Its Output:

$12x_1 + 10x_2 - 7x_3 = 15$

$6x_1 + 5x_2 + 3x_3 = 14$

$5x_1 - x_2 + 5x_3 = 9$

Define,

```
A=[12, 10, -7; 6, 5, 3; 5, -1, 5];
B=[15, 14, 9];
```

Output of Matlab Code is,

```
X =

    1

    1

    1
```

Lagrange's Interpolation:

Only the data points in which interpolation is to be performed should be provided. The Matlab codes are provided blow.

Problem:

Time t (sec)	Current i (Amp)
1	4
2	6
3	9
5	15
6	21

Find i(4) using Matlab.

1st Order:

Matlab Code:

```
function out=Lagrange_Lin(X, Y, xx)
L(1)=(xx-X(2))/(X(1)-X(2));
L(2)=(xx-X(1))/(X(2)-X(1));
out=Y(1)*L(1)+Y(2)*L(2);
end
```

Output:

Define,

```
X=[3, 5];
Y=[9, 15];
```

When the function is called,

```
out = Lagrange_Lin(X, Y, 4)

out =

    12
```

2nd Order:

Matlab Code:

```
function out=Lagrange_2nd(X, Y, xx)
L(1)=(xx-X(2))*(xx-X(3))/((X(1)-X(2))*(X(1)-X(3)));
L(2)=(xx-X(3))*(xx-X(1))/((X(2)-X(3))*(X(2)-X(1)));
L(3)=(xx-X(1))*(xx-X(2))/((X(3)-X(1))*(X(3)-X(2)));
out=Y(1)*L(1)+Y(2)*L(2)+Y(3)*L(3);
end
```

Output:

Define,

```
X=[2, 3, 5];
Y=[6, 9, 15];
```

When the function is called,

```
out = Lagrange_2nd(X, Y, 4)

out =

    12
```

3rd Order:

Matlab Code:

```
function out=Lagrange_3rd(X, Y, xx)
L(1)=(xx-X(2))*(xx-X(3))*(xx-X(4))/((X(1)-X(2))*(X(1)-X(3))*(X(1)-X(4)));
L(2)=(xx-X(3))*(xx-X(1))*(xx-X(4))/((X(2)-X(3))*(X(2)-X(1))*(X(2)-X(4)));
L(3)=(xx-X(1))*(xx-X(2))*(xx-X(4))/((X(3)-X(1))*(X(3)-X(2))*(X(3)-X(4)));
L(4)=(xx-X(1))*(xx-X(2))*(xx-X(3))/((X(4)-X(1))*(X(4)-X(2))*(X(4)-X(3)));
out =Y(1)*L(1)+Y(2)*L(2)+Y(3)*L(3)+Y(4)*L(4);
end
```

Output:

Define,

```
Y=[4, 6, 9, 15];
X=[1, 2, 3, 5];
```

When the function is called,

```
out = Lagrange_3rd(X, Y, 4)

out =

     12.2500
```

4^{th} Order:

Matlab Code:

```
function out=Lagrange_4th(X, Y, xx)
L(1)=(xx-X(2))*(xx-X(3))*(xx-X(4))*(xx-X(5))/((X(1)-X(2))*(X(1)-X(3))*(X(1)-
X(4))*(X(1)-X(5)));

L(2)=(xx-X(3))*(xx-X(1))*(xx-X(4))*(xx-X(5))/((X(2)-X(3))*(X(2)-X(1))*(X(2)-
X(4))*(X(2)-X(5)));

L(3)=(xx-X(1))*(xx-X(2))*(xx-X(4))*(xx-X(5))/((X(3)-X(1))*(X(3)-X(2))*(X(3)-
X(4))*(X(3)-X(5)));

L(4)=(xx-X(1))*(xx-X(2))*(xx-X(3))*(xx-X(5))/((X(4)-X(1))*(X(4)-X(2))*(X(4)-
X(3))*(X(4)-X(5)));

L(5)=(xx-X(1))*(xx-X(2))*(xx-X(3))*(xx-X(4))/((X(5)-X(1))*(X(5)-X(2))*(X(5)-
X(3))*(X(5)-X(4)));
out =Y(1)*L(1)+Y(2)*L(2)+Y(3)*L(3)+Y(4)*L(4)+Y(5)*L(5);
end
```

Output:

Define,

```
X=[1, 2, 3, 5, 6];
Y=[4, 6, 9, 15, 21];
```

When the function is called,

```
out = Lagrange_4th(X, Y, 4)

out =

      11.8000
```