HW_2
Vibhanshu Jain,
CS19B1027

Q1:

The LU decomposition method in linear algebra has used the solve the linear equation. In this step, we will first represent the linear equation in the form of a matrix. Now, we factorize the given square matrix into two triangular matrices, one upper triangular and one lower triangular matrix, such that the product of these two matrices gives the original matrix.

A X = C
Where A is the coefficient matrix, X is the variable, and C is the matrix of the numbers on the right

A = L U
A: given matrix, L: lower triangular matrix, U: upper triangular matrix

Now substitute A = L U in the first equation, which will result in L U X = C
Now put, Z = U X, after which it will be L Z = C, from here we can find the matrix Z, and later we can solve for Z = U X which will give the matrix and the solution X.

The methods to get LU Decomposition are:

- Gauss Elimination
- Doolittle’s method
- Crout's method
- Cholesky method

Gauss Elimination:
In this method, we try to get the upper or lower triangular matrix using row reduction using the elementary row operations, which are:

- Swapping two rows Ri ↔ Rj
- Multiplying a row by a nonzero number Ri ↔ kRi
- Adding a multiple of one row to another row Ri ↔ Ri + kRj

Q2:

The function to find the root “roots(p)”

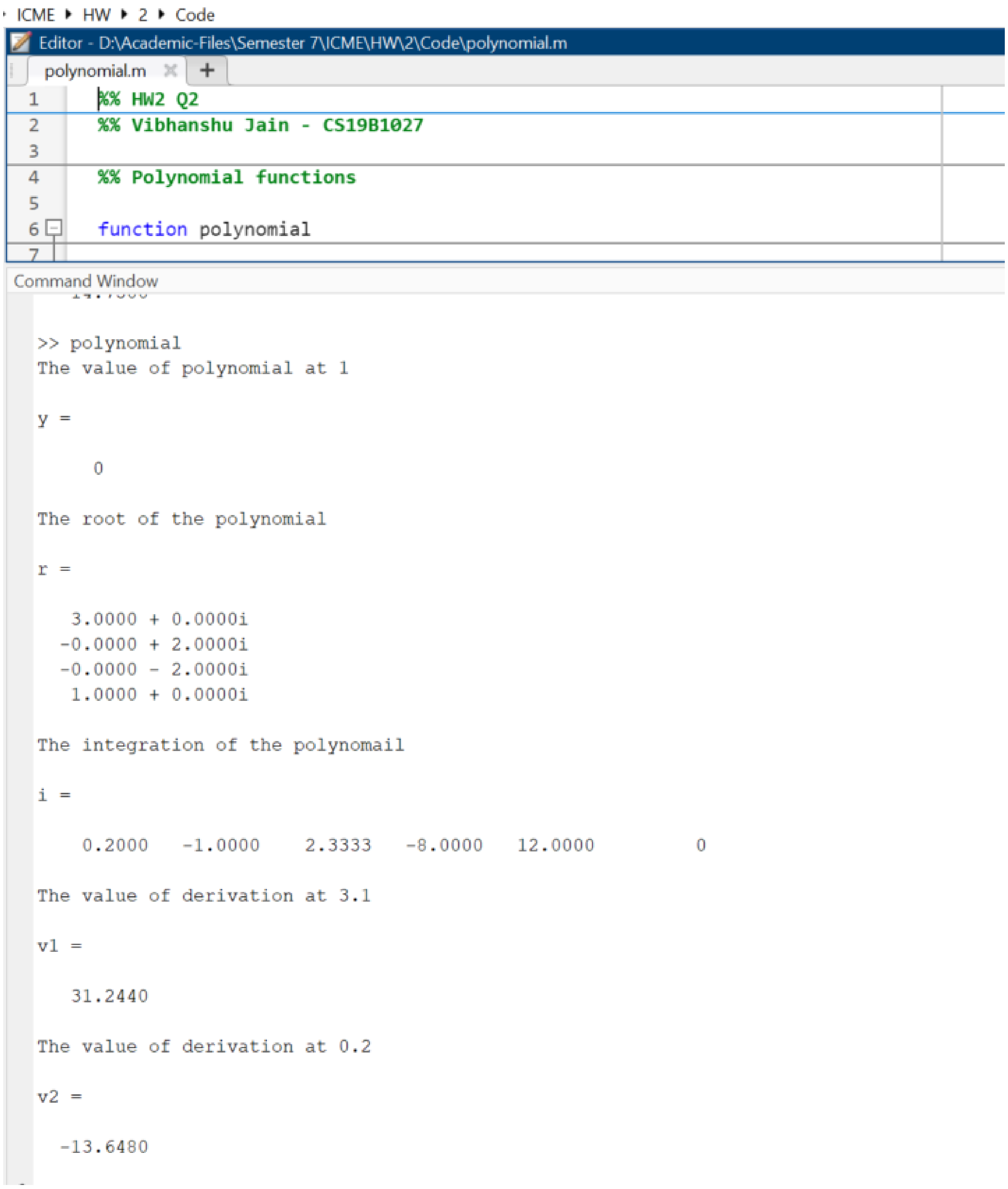
(a) Integration function polyint(p)

(b) Finding the derivate

(i) Firstly find the derivative of the function using the function “polyder(p) which will return a polynomial function

(ii) Find the value of the derivate function return by the previous step using, “ployval(p, point)”

```
%% HW2 Q2
%% Vibhanshu Jain - CS19B1027
%% Polynomial functions
function polynomial
%% The polynomial equation
%% y(x) = (1)*x^4 + (-4)*x^3 + 7*x^2 + (-16)*x^1 + 12*x^0
%% the polynomail representation of matlab
p = [1 -4 7 -16 12];
%% Printing the value at x = 1
disp("The value of polynomial at 1");
y = polyval(p,1)
%% The root of the polynomial
disp("The root of the polynomial");
r = roots(p)
%% The intergration of the polynomial
disp("The integration of the polynomail");
i = polyint(p)
%% The derivate
d  = polyder(p);
%% Derivate value at 3.1 & 0.2
disp("The value of derivation at 3.1");
v1 = polyval(d,3.1)
disp("The value of derivation at 0.2");
v2 = polyval(d,0.2)
end
```



ICME ▸ HW ▸ 2 ▸ Code

Editor - D:\Academic-Files\Semester 7\ICME\HW\2\Code\polynomial.m

polynomial.m

```
%% HW2 Q2
%% Vibhanshu Jain - CS19B1027

%% Polynomial functions

function polynomial

```

Command Window

```
>> polynomial
The value of polynomial at 1

y =

     0

The root of the polynomial

r =

   3.0000 + 0.0000i
  -0.0000 + 2.0000i
  -0.0000 - 2.0000i
   1.0000 + 0.0000i

The integration of the polynomail

i =

    0.2000   -1.0000    2.3333   -8.0000   12.0000         0

The value of derivation at 3.1

v1 =

   31.2440

The value of derivation at 0.2

v2 =

  -13.6480
```

Q3: Lagrange polynomial

If we are given n points of a function f(x) then, we can write a polynomial function of degree n-1, and this function is called Lagrange Polynomial.
Interpolation is a technique to estimate the value of a mathematical function. for any intermediate value of the independent variable.
The Lagrange Interpolation function is given by,

$$y = f(x) = \frac{(x-x_1)(x-x_2)\ldots(x-x_n)}{(x_0-x_1)(x_0-x_2)\ldots(x_0-x_n)} y_0 + \frac{(x-x_0)(x-x_2)\ldots(x-x_n)}{(x_1-x_0)(x_1-x_2)\ldots(x_1-x_n)} y_1 + \ldots + \frac{(x-x_0)(x-x_1)\ldots(x-x_{n-1})}{(x_n-x_0)(x_n-x_1)\ldots(x_n-x_{n-1})} y_n$$

Let’s try to understand this using our code. Let’s take a 3-degree polynomial, with the function values (3, -4, 5, -6) at (-1, 0, 1, 2). Let’s try to value our Lagrange created by this function at x. Also, let’s assume that temp is the value of the Lagrange at the given x.
The Matlab code looks like this,

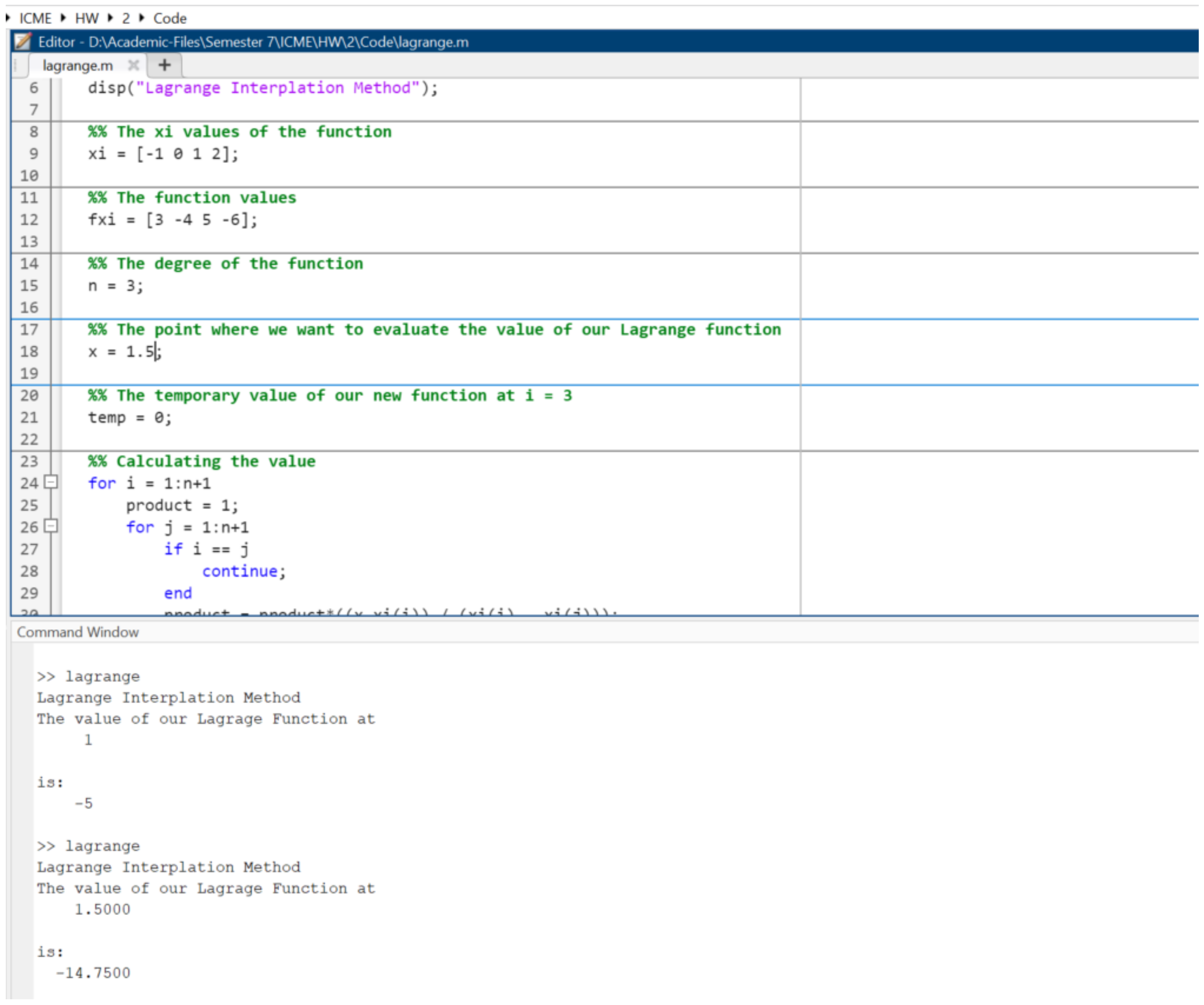
```
%% The xi values of the function
xi = [-1 0 1 2];
%% The function values
fxi = [3 -4 5 -6];
%% The degree of the function
n = 3;
%% The point where we want to evaluate the value of our Lagrange function
x = 0;
%% The temporary value of our new function at i = 3
temp = 0;
```

Now we will be calculating the values of the function by iterating through two loops.

```
%% Calculating the value
for i = 1:n+1
   product = 1;
   for j = 1:n+1
       if i == j
           continue;
       end
       product = product*((x-xi(i)) / (xi(i) - xi(j)));
   end
   temp = temp + fxi(i)*product;
end
```

Here, we are skipping the calculations when i == j, as the value at this point will be zero, which won't change anything in the final product. Later we are also calculating the value of the function at point x.

In the same way can also find the complete function, not only the specific value.


ICME ▸ HW ▸ 2 ▸ Code
Editor - D:\Academic-Files\Semester 7\ICME\HW\2\Code\lagrange.m
lagrange.m
Command Window

```
disp("Lagrange Interplation Method");

%% The xi values of the function
xi = [-1 0 1 2];

%% The function values
fxi = [3 -4 5 -6];

%% The degree of the function
n = 3;

%% The point where we want to evaluate the value of our Lagrange function
x = 1.5;

%% The temporary value of our new function at i = 3
temp = 0;

%% Calculating the value
for i = 1:n+1
    product = 1;
    for j = 1:n+1
        if i == j
            continue;
        end
```

```
>> lagrange
Lagrange Interplation Method
The value of our Lagrage Function at
     1

is:
    -5

>> lagrange
Lagrange Interplation Method
The value of our Lagrage Function at
    1.5000

is:
  -14.7500
```