Group Members Name	Urwa Maryam, Umar Hayyat
Reg. #	2019-EE-352, 2019-EE-360
Marks	4.5 28-09-21

Experiment #2

Polynomials in MATLAB

Introduction:

Objective: The objective of this session is to learn how to represent polynomials in MATLAB, find roots of polynomials, create polynomials when roots are known and obtain partial fractions.

Polynomial Overview:

MATLAB provides functions for standard polynomial operations, such as polynomial roots, evaluation, and differentiation. In addition, there are functions for more advanced applications, such as curve fitting and partial fraction expansion.

Polynomial Function Summary

Function	Description
Conv	Multiply polynomials
Deconv	Divide polynomials
Poly	Polynomial with specified roots
Polyder	Polynomial derivative
Polyfit	Polynomial curve fitting
Polyval	Polynomial evaluation
Polyvalm	Matrix polynomial evaluation
Residue	Partial-fraction expansion (residues)
Roots	Find polynomial roots

MATLAB represents polynomials as row vectors containing coefficients ordered by descending powers.
For example: $p(x)= ax^2 + bx + c$ is represented by p=[a b c] in MATLAB.

Objective:

Using MATLAB to solve polynomials and partial Fraction.

Polynomial ROOTS:

Roots are calculated by roots(g) command, where g is a polynomial equation

```
>>p = [1 0 -2 -5];

>>r = roots(p)

r =
    2.0946
   -1.0473 +    1.1359i
   -1.0473 -    1.1359i
```

```
Command Window
>> x=[2 4 2]

x =

     2     4     2

>> roots(x)

ans =

    -1
    -1
```

- **Polynomial Evaluation:**
  It is used to find the value of a polynomial value at a specific. Point by command polyval(s,p), where p is a number and s polynomial as shown below.

```
>>polyval(p,5)

ans =
   110
```

```
Command Window
>> x=[2 4 2];
>> c=polyval(x,2)

c =

    18
```

Convolution and Deconvolution:

Polynomial multiplication and division correspond to the operations convolution and deconvolution. The functions conv and deconv implement these operations. Consider the polynomials a(s)= $s^2 + 2s + 3$ and b(s)=4$s^2 + 5s + 6$. To compute their product,

```
>>a = [1 2 3]; b = [4 5 6];
>>c = conv(a,b)

c =
    4    13    28    27    18
```

Use deconvolution to divide back out of the product:

```
>>[q,r] = deconv(c,a)

q =
    4    5    6

r =
    0    0    0    0    0
```

Polynomial Derivatives:

The polyder function computes the derivative of any polynomial. To obtain the derivative of the polynomial

```
>>p= [1 0 -2 -5]
>>q = polyder(p)
```

```
q =
   3   0   -2
```

polyder also computes the derivative of the product or quotient of two polynomials. For example, create two polynomials a and b:

```
>>a = [1 3 5];
>>b = [2 4 6];
```

Calculate the derivative of the product a*b by calling polyder with a single output argument:

```
>>c = polyder(a,b)
```

```
c =
   8   30   56   38
```

Partial Fraction Expansion

'residue' finds the partial fraction expansion of the ratio of two polynomials. This is particularly useful for applications that represent systems in transfer function form. For polynomials b and a,

$$\frac{b(s)}{a(s)} = \frac{r_1}{s-p_1} + \frac{r_2}{s-p_2} + \cdots + \frac{r_n}{s-p_n} + k_s$$

if there are no multiple roots, where r is a column vector of residues, p is a column vector of pole locations, and k is a row vector of direct terms.

Consider the transfer function

```
>>b = [-4 8];
>>a = [1 6 8];
>>[r,p,k] = residue(b,a)
```

Residue: Residue is used for solving partial fraction in MATLAB. We use residue as [r,p,k]=residue(b,a).

```
r =
  -12
    8

p =
   -4
   -2

k =
    []
```

Exercise 1:

Consider the two polynomials $p(s) = s^2 + 2s + 1$ and $q(s) = s + 1$. Using MATLAB compute

a. $p(s) * q(s)$
b. Roots of $p(s)$ and $q(s)$
c. $p(-1)$ and $q(6)$

Exercise 2:

Use MATLAB command to find the partial fraction of the following

a. $\frac{B(s)}{A(s)} = \frac{2s^3+5s^2+3s+6}{s^3+6s^2+11s+6}$
b. $\frac{B(s)}{A(s)} = \frac{s^2+2s+3}{(s+1)^3}$

Exercise#1:

```
format compact
disp('Given Polynomials')
p = [ 1 2 1 ]
q = [ 1 1 ]
disp('Ex#1:(a) p(s) * q(s)')
conv(p,q)
disp('Ex#1:(b) Roots of p and q')
roots_p = roots(p)
roots_q = roots(q)
disp('Ex#1:(c) Value of polynomial at specific value p(-1) & q(6)')
p_at_1 = polyval(p,-1)
q_at_6 = polyval(q,6)
```

```
Command Window
Given Polynomials
p =
     1     2     1
q =
     1     1
Ex#1:(a) p(s) * q(s)
ans =
     1     3     3     1
Ex#1:(b) Roots of p and q
roots_p =
    -1
    -1
roots_q =
    -1
Ex#1:(c) Value of polynomial at specific value p(-1) & q(6)
p_at_1 =
     0
q_at_6 =
     7
fx >>
```

Exercise#2:

```
format compact
disp('Ex#2:(a)')
disp('Given polynomial')
B = [ 2 5 3 6 ]
A = [ 1 6 11 6 ]
disp('r: Column vector of residues, p: Column vector of pole location, k:
Row vector')
[r, p, k] = residue(B, A)
disp('Ex#2:(b)')
disp('Given polynomial')
B = [ 1 2 3 ]
A = [ 1 3 3 1 ]         %(a+b)^3 = a^3 + 3(a^2)(b) + 3(a)(b^2) + b^3
disp('r: Column vector of residues, p: Column vector of pole location, k:
Row vector')
[r, p, k] = residue(B, A)
```

Command Window

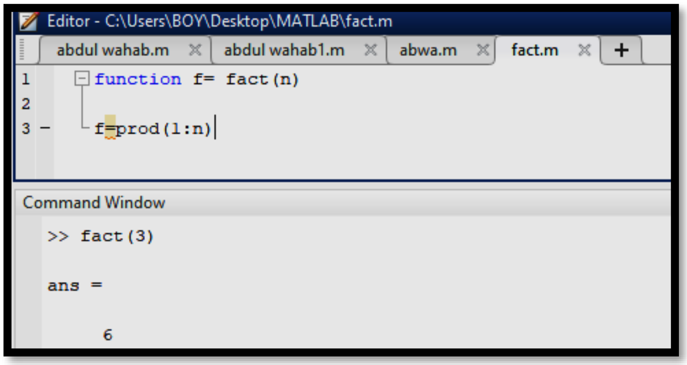
```
Ex#2:(a)
Given polynomial
B =
     2     5     3     6
A =
     1     6    11     6
r: Column vector of residues, p: Column vector of pole location, k: Row vector
r =
   -6.0000
   -4.0000
    3.0000
p =
   -3.0000
   -2.0000
   -1.0000
k =
     2
Ex#2:(b)
Given polynomial
B =
     1     2     3
A =
     1     3     3     1
r: Column vector of residues, p: Column vector of pole location, k: Row vector
r =
    1.0000
    0.0000
    2.0000
p =
   -1.0000
   -1.0000
   -1.0000
k =
     []
fx >>
```

- ***Flow Control:***

  Flow control is the part of MATLAB which controls the if-else statements, for loop and switch statements.

- ***Functions (m-Files):***

M.file is just like a command window in which we can write the input argument, the name of the M-file and the function should be the same. Functions are M-files that can accept input arguments and return output arguments. The names of the M-file and of the function should be the same. Functions operate on variables within their own workspace, separate from the workspace you access at the MATLAB command prompt. An example is provided below:



Flow Control:

Conditional Control – if, else, switch:

This section covers those MATLAB functions that provide conditional program control. if, else, and elseif. The if statement evaluates a logical expression and executes a group of statements when the expression is true. The optional elseif and else keywords provide for the execution of alternate groups of statements. An end keyword, which matches the if, terminates the last group of statements. The groups of statements are delineated by the four keywords—no braces or brackets are involved as given below.

```
if <condition>
        <statements>;
elseif <condition>
        <statements>;
else
        <statements>;
end
```

It is important to understand how relational operators and if statements work with matrices. When you want to check for equality between two variables, you might use

Exersice 1: MATLAB M-file Script

Use MATLAB to generate the first 100 terms in the sequence **a(n)** define recursively by

$$a(n+1) = p * a(n) * (1 - a(n))$$

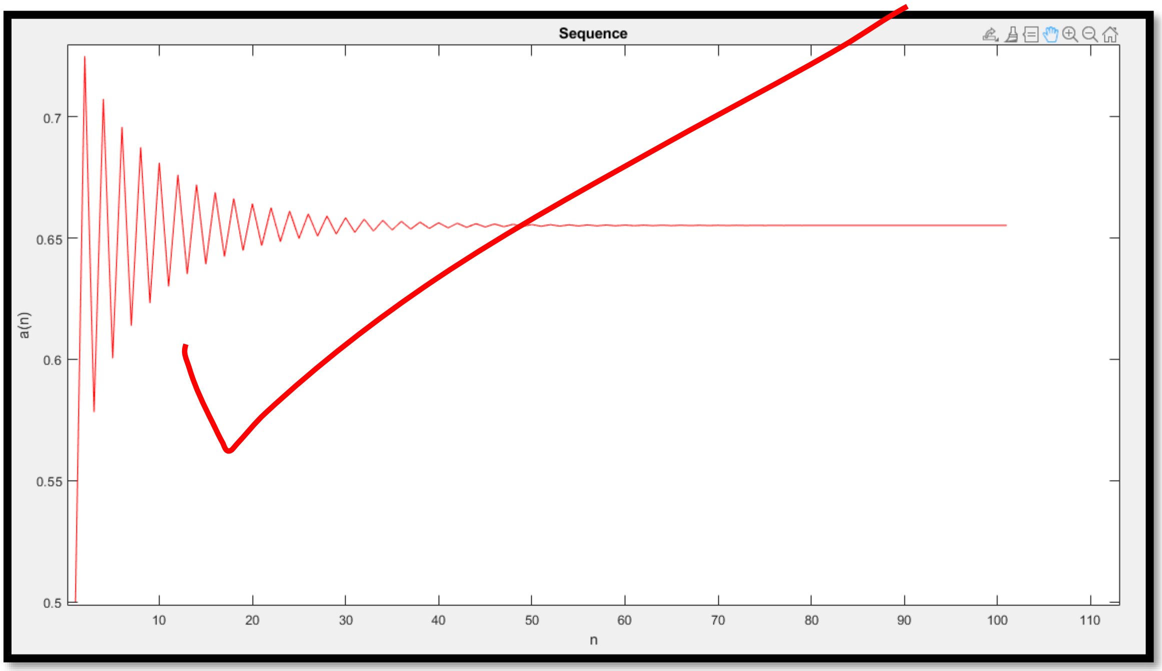
with p=2.9 and a(1) = 0.5.

After you obtain the sequence, plot the sequence.

Exercise#1:

```
function sequence
p=2.9;
a(1)=0.5;
for n=1:100
```

```
     a(n+1)= p*a(n)*(1-a(n));
end
plot(a,'r')
title('Sequence');              %title of Graph
xlabel('n');               %label of x-axis
ylabel('a(n)');               %label of y-axis
end
```



Exersice 2: MATLAB M-file Function

Consider the following equation

$$y(t) = \frac{y0}{\sqrt{1-\zeta}} e^{-\zeta\omega_n t} \sin(\omega_n \sqrt{1-\zeta^2} * t + \theta)$$

a) Write a MATLAB M-file function to obtain numerical values of y(t). Your function must take y(0), ζ, ω_n, t and θ as function inputs and y(t) as output argument.

b) Write a second script m-file to obtain the plot for y(t) for 0<t<10 with an increment of 0.1, by considering the following two cases
Case 1: y0=0.15 m, $\omega_n = \sqrt{2}$ rad/sec, $\zeta = 3/(2\sqrt{2})$ and $\theta = 0$;
Case 2: y0=0.15 m, $\omega_n = \sqrt{2}$ rad/sec, $\zeta = 1/(2\sqrt{2})$ and $\theta = 0$;

Hint: When you write the function you would require element-by-element operator

Exercise#2:

(a)

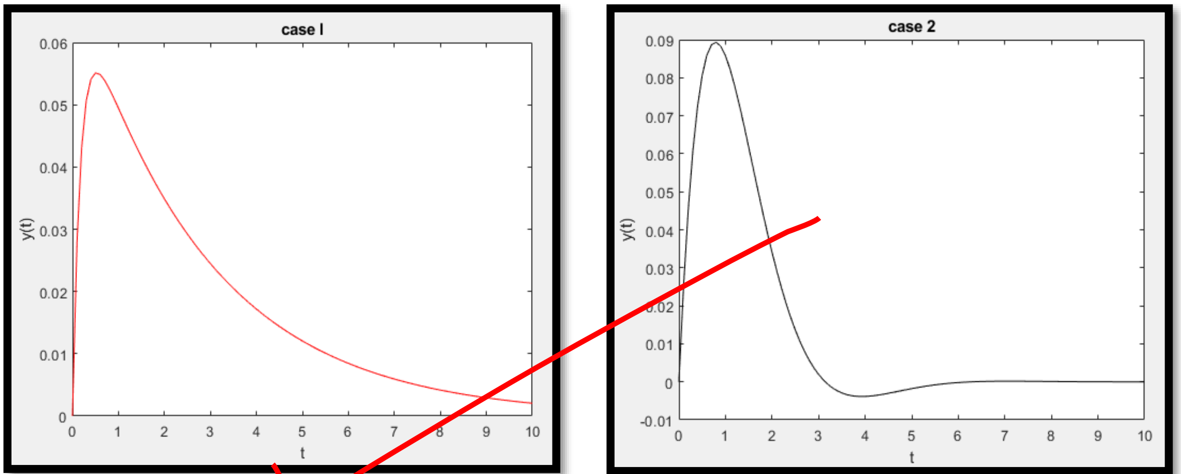
```
function y=fun(yo,z,w,t,x)   %function header
val = (yo.*exp(-z*w.*t).*sin(w.*(sqrt(1-z*z)).*t+x))/sqrt(1-z)    %equation
end
```

```
>> clear all
>> fun(0.2, 2/(4*sqrt(5)), sqrt(5), 0:0.5:5, 2)

val =

    0.2064    0.0092   -0.1186   -0.0910    0.0063    0.0598    0.0392   -0.0080   -0.0295   -0.0165    0.0061
```

(b)

```
clc;
clear all;
disp('Given equation is')
disp('(yo*exp(-z*w*t)*sin(w*(sqrt(1-z*z))*t+x))/sqrt(1-z)')
disp('Case 1 is:   yo = 0.15m, omega = sqrt(2)rad/sec, zitta =
3/(2*(sqrt(2))) , theta = 0;');
disp('Case 2 is:   yo = 0.15m, omega = sqrt(2)rad/sec, zitta =
1/(2*(sqrt(2))) , theta = 0;');
choice = input('Please enter 1 for case1 and 2 for case2 = ');
% if user enters 1 then case 1 will be executed
if choice == 1
yo=0.15;
z=3/2*(sqrt(2));    %zitta
w=sqrt(2);      %omega
t=0:.1:10;
x=0;      %theta
val = (yo.*exp(-z*w.*t).*sin(w.*(sqrt(1-z*z)).*t+x))/sqrt(1-z);
plot(t, val, 'r');
title('case 1');
xlabel('t');
ylabel('y(t)');
% if user enters 2 then case 2 will be executed
elseif choice == 2
yo=0.15;
z=1/2*(sqrt(2));     %zitta
w=sqrt(2);        %omega
t=0:0.1:10;
x=0;         %theta
val = (yo.*exp(-z*w.*t).*sin(w.*(sqrt(1-z*z)).*t+x))/sqrt(1-z);
plot(t, val, 'black');
title('case 2');
xlabel('t');
ylabel('y(t)');
% error statement in case of wrong input entered by the user
else
disp('Invalid input');
end;
```

```
Given equation is
(yo*exp(-z*w*t)*sin(w*(sqrt(1-z*z))*t+x))/sqrt(1-z)
Case 1 is:   yo = 0.15m, omega = sqrt(2)rad/sec, zitta = 3/(2*(sqrt(2))) , theta = 0;
Case 2 is:   yo = 0.15m, omega = sqrt(2)rad/sec, zitta = 1/(2*(sqrt(2))) , theta = 0;
fx Please enter 1 for case1 and 2 for case2 =
```



Exersice 3: MATLAB Flow Control

Use 'for' or 'while' loop to convert degrees Fahrenheit (T_f) to degrees Celsius using the following equation $T_f = \frac{9}{5} * T_c + 32$. Use any starting temperature, increment and ending temperature (example: starting temperature=0, increment=10, ending temperature = 200).

Exercise#3:

```
disp('Ex#3: Celsius to Fahrenheit Conversion')
disp('Celsius    |    Fahrenheit')
for t=0:10:200
    temp_F=5/9*(t-32);           %formula to convert Celsius to Fahrenheit
 fprintf('%d                 %f\n',t,temp_F);       %display pattern
end
```

```
Command Window
Ex#3: Celsius to Fahrenheit Conversion
Celsius    |    Fahrenheit
0               -17.777778
10               -12.222222
20               -6.666667
30               -1.111111
40               4.444444
50               10.000000
60               15.555556
70               21.111111
80               26.666667
90               32.222222
100               37.777778
110               43.333333
120               48.888889
130               54.444444
140               60.000000
150               65.555556
160               71.111111
170               76.666667
180               82.222222
190               87.777778
200               93.333333
```

Conclusion?