New York City College of Technology

Department of Computer Engineering Technology

Feedback Control System

CET 4864L

Spring 2015

LAB Experiment # 1

Introduction to MATLAB

Yeraldina Estrella

February 15, 2015

Professor Bustamante

Objective

The purpose of this lab is to introduce MATLAB. This lab consists of tutorials of polynomials, script writing and programming aspects of MATLAB from control systems view point. This lab will introduce you to the concept of mathematical programming using the software called MATLAB. We shall study how to define variables, matrices etc, see how we can plot results and write simple MATLAB codes. In this lab will teach how to represent polynomials in MATLAB, find roots of polynomials, create polynomials when roots are known and obtain partial fractions.

Experimental

I) Introduction to MATLAB

Exercise #1

Use Matlab command to obtain the following
a) Extract the fourth row of the matrix generated by magic(6)
b) Show the results of 'x' multiply by 'y' and 'y' divides by 'x'.
Given x = [0:0.1:1.1] and y = [10:21]
c) Generate random matrix 'r' of size 4 by 5 with number varying between -8 and 9

A) Extract the fourth row of the matrix generated by magic(6)

Output

```
>> Lab1Exc1aMatrices
This is Magic(6):
    35     1     6    26    19    24
     3    32     7    21    23    25
    31     9     2    22    27    20
     8    28    33    17    10    15
    30     5    34    12    14    16
     4    36    29    13    18    11

This is the fourth Row of Magic 6:
     8    28    33    17    10    15
```

MathLab allows creating random matrices of users defined dimentions. MatLab also allows to extract rows and columns of matrices. The output above clearly shows a random matrix and the extracted sixth row of a matrix.

Source Code

```
%Yeraldina Estrella
%CET 4873
%Lab #1
%Exercise #1

%Extract the fourth row of the matrix generated by magic(6)

%generate matrix magic(6)
magic6= magic(6);
disp('This is Magic(6): ');
disp(magic6);

%Obtain the fourth row elements of magic6
fourthRowMagic6=magic6(4,:);
disp('This is the fourth Row of Magic 6: ');
disp(fourthRowMagic6);
```

B) Show the results of 'x' multiply by 'y' and 'y' divides by 'x'

Output

```
>> Lab1Exc1bMatrices
x:
         0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000    0.7000    0.8000    0.9000    1.0000    1.1000

y:
    10    11    12    13    14    15    16    17    18    19    20    21

Multiplication of 'x' and 'y' is:
         0    1.1000    2.4000    3.9000    5.6000    7.5000    9.6000   11.9000   14.4000   17.1000   20.0000   23.1000

Division of 'y' and 'x' is:
       Inf  110.0000   60.0000   43.3333   35.0000   30.0000   26.6667   24.2857   22.5000   21.1111   20.0000   19.0909
```

The output above clearly demonstrate that matrices can be multiply and divided.

Source Code

```
%Yeraldina Estrella
%CET 4873
%Lab #1
%Exercise #1 Part B. Matrices

%Show the results of 'x' multiply by 'y' and 'y' divided by 'x'.
%Given x=[0:0.1:1.1] and y=[10:21]

%creating and array x
x=[0:0.1:1.1];
y=[10:21];
%x multiply by y
multip_xy= x.*y;
```

```
%y divide by x
div_yx=y./x;

%display Text
disp('x: ');
disp(x);
disp('y: ');
disp(y);
disp('Multiplication of ''x'' and ''y'' is: ');
disp(multip_xy);

disp('Division of ''y'' and ''x'' is: ');
disp(div_yx);
```

C) Generate random matrix 'r' of size 4 by 5 with number varying between -8 and 9

Output

```
>> Lab1Exc1cMatrices
Here is a ramdom 4 by 5 matrix varying between -8 and 9
     3     4     3    -4     4
    -8     5    -5    -8    -3
     7     5     4    -7     9
     8    -1    -8     6    -8
```

The above output demonstrate that MatLab can be use to create random matrices of specified ranges of numbers.

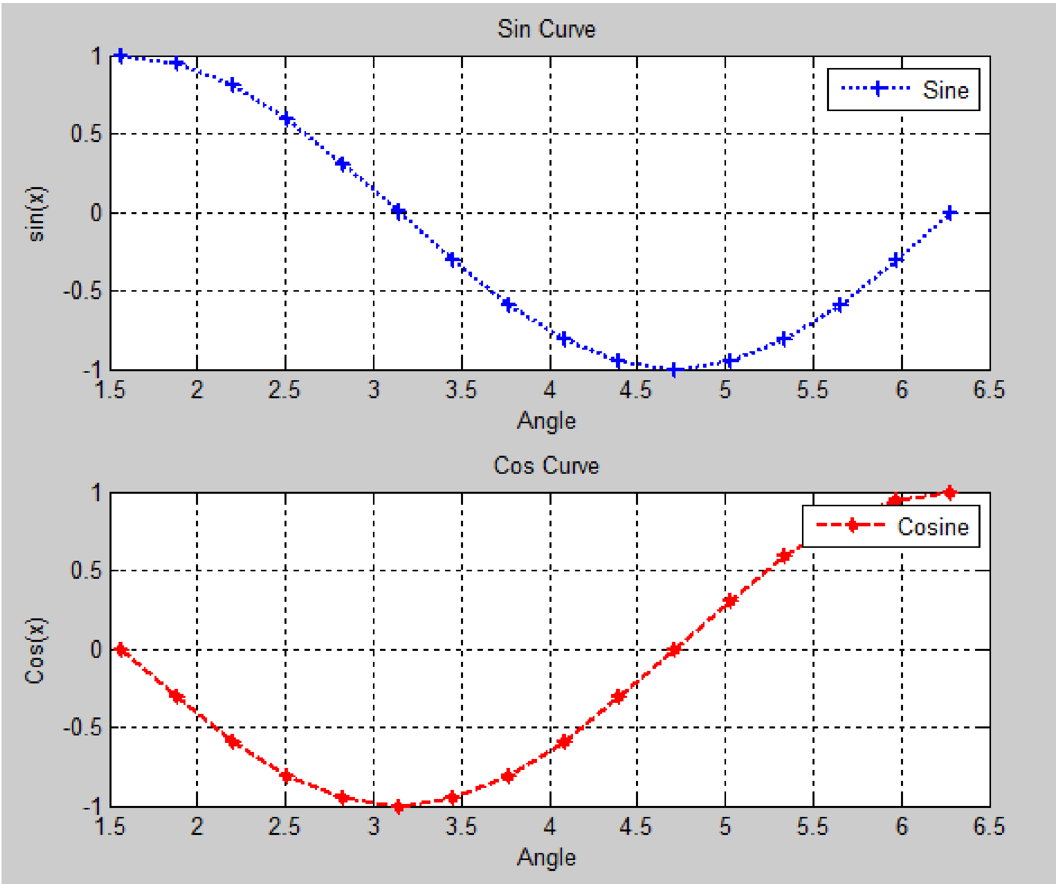
Source Code

```
%Yeraldina Estrella
%CET 4873
%Lab #1
%Exercise #1 Part C

%Generate random matrix 'r' of size 4 by 5 with numbers varying between
%-8 and 9

%creates a ramdom matrix of 4 rows and 5 colunms varying between -8 and 9
r=randi([-8 9], 4,5);
disp('Here is a ramdom 4 by 5 matrix varying between -8 and 9 ');
disp(r);
```

Exercise #2 Plotting

Use MATLAB commands to get exactly as the figure shown below x=pi/2:pi/10:2*pi; y=sin(x); z=cos(x); Lab Experiment 1: Using MATLAB for Control System

Output



The above graph represent the sine function (in Blue) and the cosine function (in Red). This graph demonstrate that MatLab can be use to generate graphs such as that of the trigonometric functions. MatLab allows formatting graphs by setting the line color, texture of the line and width of the line.

Source Code

```
clc
clear

%Yeraldina Estrella
%CET 4873
%Lab #1
%Exercise #2 Part A

%Use the MATLAB commands to get exactly as the figure shown below

x=[(pi/2):(pi/10):(2*pi)];

y=sin(x);

z=cos(x);

figure;
subplot(2,1,1)
plot(x,y, 'b:+','linewidth', 2);
legend('Sine');
title('Sin Curve', 'fontsize', 10)
ylabel('sin(x)')
xlabel('Angle')

grid;


subplot(2,1,2)
plot(x,z, 'r--*','linewidth', 2);
title('Cos Curve', 'fontsize', 10);
legend('Cosine');
ylabel('Cos(x)')
xlabel('Angle')
grid;
```

Part II. Polynomials in MATLAB

Exercise #1

Consider the two polynomials ( ) and ( ) . Using MATLAB compute
a. p(s) * q(s)
b. Roots of p(s) and q(s)
c. p(-1) and q(6)

Output

```
>> lab1Ex1aPolynomials
The product of p(s) and q(s) is:
     0     1     3     3     1

The roots of p(s) are:
    -1
    -1

The roots of q(s) are:
    -1

The evaluation of p(-1):
     0

The evaluation of q(6):
     7

>> |
```

The above output shows that MatLab is useful at multiplying polynomials, finding the roots of polynomials and evaluating polynomials at a certain value.

Source Code

```
%Yeraldina Estrella
%CET 4873
%Lab #1
%Exercise #1a
%Polynomials

%Consider the two polynomials p(s)=s2 + 2s +1 and q(s) = s + 1
% compute p(s) * q(s)

p=[1 2 1];

q=[0 1 1];
```

```
product= conv(p,q);

disp('The product of p(s) and q(s) is: ');
disp(product);

%roots of p(s0 and q(s)

pRoots = roots(p);
disp('The roots of p(s) are: ');
disp(pRoots);

qRoots = roots(q);
disp('The roots of q(s) are: ');
disp(qRoots);


%Polynomial evaluation

pNeg= polyval(p,-1);

disp('The evaluation of p(-1): ');
disp(pNeg);

q(6)= polyval(q,6);
disp('The evaluation of q(6): ');
disp(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. $\frac{B(s)}{A(s)} = \frac{s^2+2s+3}{(s+1)^3}$

Output

```
>> lab1Ex2aPolynomials
r =
   -6.0000
   -4.0000
    3.0000

p =
   -3.0000
   -2.0000
   -1.0000

k =
     2

this is (s+1)^3:
     1     3     3     1


a =
    1.0000
   -0.0000
    2.0000

b =
   -1.0000
   -1.0000
   -1.0000

c =
>> 
```

From MathWorks Tutorials:

For polynomials b and a, if there are no multiple roots,

$$\frac{b(s)}{a(s)} = \frac{r_1}{s-p_1} + \frac{r_2}{s-p_2} + \frac{r_n}{s-p_n} + k_2$$

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.

According to the output, the partial fractions are as follow:

a)

. $\frac{B(s)}{A(s)} = \frac{2s^3+5s^2+3s+6}{s^3+6s^2+11s+6} = \frac{-6}{s+3} + \frac{-4}{s+2} + \frac{3}{s+1} + 2$

b)

$\frac{B(s)}{A(s)} = \frac{s^2+2s+3}{(s+1)^3} = \frac{1}{s+1} + \frac{0}{s-2} + \frac{2}{s+1}$

Source Code

```
%Yeraldina Estrella
%CET 4873
%Lab #1
%Exercise #2
%Polynomials

%find the partial fractions of a. and b.
%a. (B(s))/(A(s)) where B(s)= 2s3 + 5s2 + 3s + 6
% and A(s)= s3 + 6s2 + 11s + 6

B1_s= [2 5 3 6];

A1_s= [1 6 11 6];
[r,p,k]= residue(B1_s, A1_s);
disp('r = ');
disp(r);
disp('p = ');
disp(p);
disp('k = ');
disp(k);

%find the partial fractions of a. and b.
%a. (B(s))/(A(s)) where B(s)= s2 + 2s + 3
% and A(s)= (s+1)^3
```

```
B2_s= [1 2 3];
a2=[1 1];
A2= conv(a2,a2);
A2_s=conv(a2,A2);
disp('this is (s+1)^3: ');
disp(A2_s);


[a,b,c]= residue(B2_s, A2_s);
disp('a = ');
disp(a);
disp('b = ');
disp(b);
disp('c = ');
disp(c);
```

Part III. Scripts, Functions & Flow Control in MATLAB

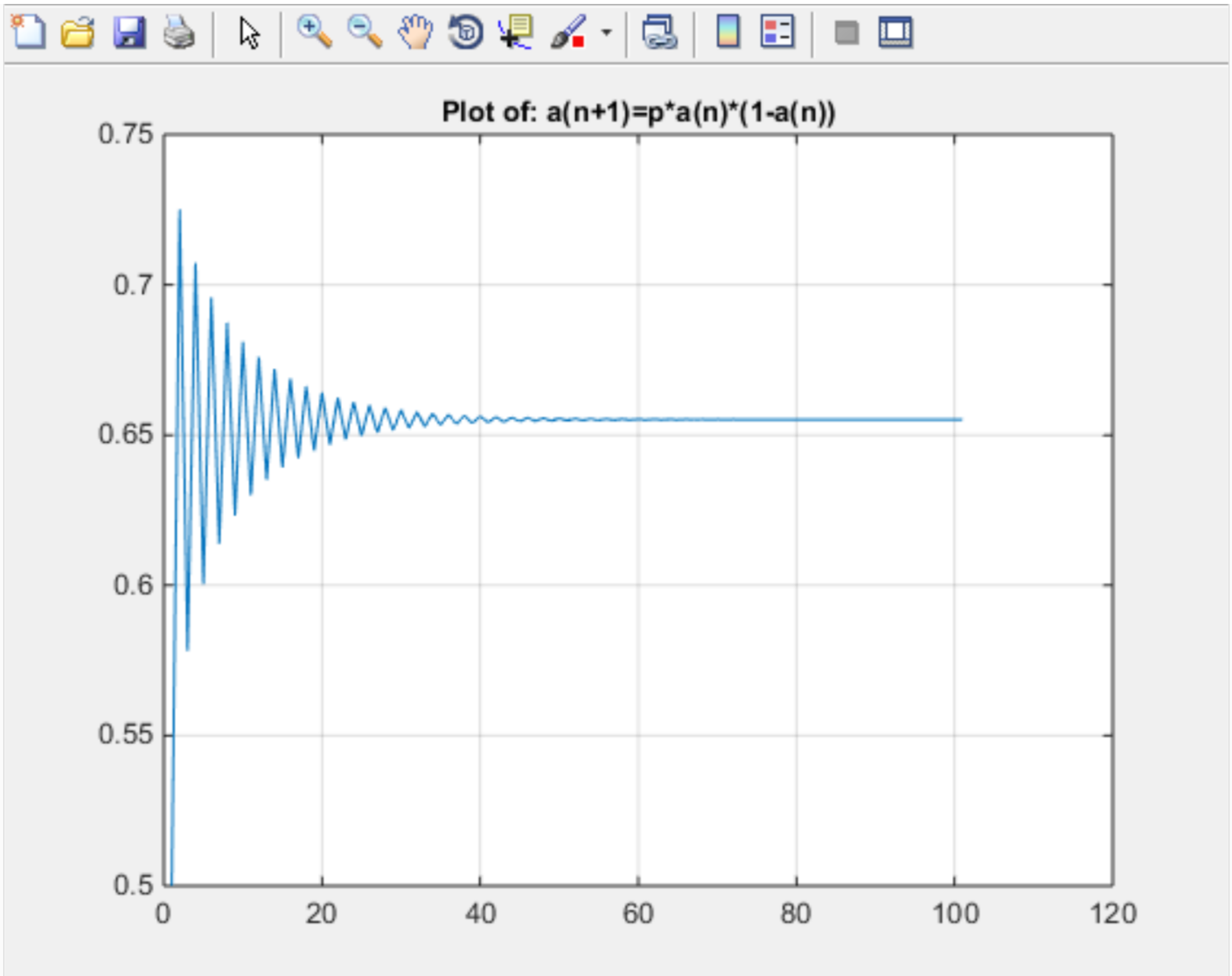
Exercise #1

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.

Source Code

```
clc
clear

%Yeraldina Estrella
%CET 4873
%Lab #1
%Exercise #1. MATLAB M-File script
%{
Generating the first 100 terms in the sequence a(n)
define by the equation a(n+1)=p*a(n)*(1-a(n))
where p = 2.9 and a(1)= .5
%}

p=2.9;
a(1)=0.5;

for n=1:100

    a(n+1)=p*a(n)*(1-a(n));
end;
plot(a)
title('Plot of: a(n+1)=p*a(n)*(1-a(n))', 'fontsize', 10)

grid;
```

Output



The graph above represents the sequence plot of a(n+1) = p * a(n) * (1 - a(n))
with p=2.9 and a(1) = 0.5.

Exercise #2. MatLab M-File Function

Consider the following equation

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

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, ω_n = √ rad/sec, ζ = 3/(2√) and θ = 0;
Case 2: y0=0.15 m, ω_n = √ rad/sec, ζ = 1/(2√) and θ = 0;

Source Code Part A

```
%Yeraldina Estrella
%CET 4873
%Lab #1
%MatLab M-file Function Exersice

%{
This program will ask users to input values for the following equation
y(t)=(y0/(sqrt(1-tao)))*exp(-tao*omegaN*t)*sin(omegaN*(sqrt(1-(tao^2))*t +
angleTeta));
%}

y0=input('Please enter the value of y0: ');
tao=input('Please enter the value of tao: ');
omegaN=input('Please enter the value of omega (in rad per second): ');
angleTeta=input('Please enter the value of teta: ');

t=input('Please input the value of ''t'' to evaluate the function y(t): ');

y_t=(y0/(sqrt(1-tao)))*exp(-tao*omegaN*t)*sin(omegaN*(sqrt(1-(tao^2))*t +
angleTeta));

disp(y_t);
```

Output Part A

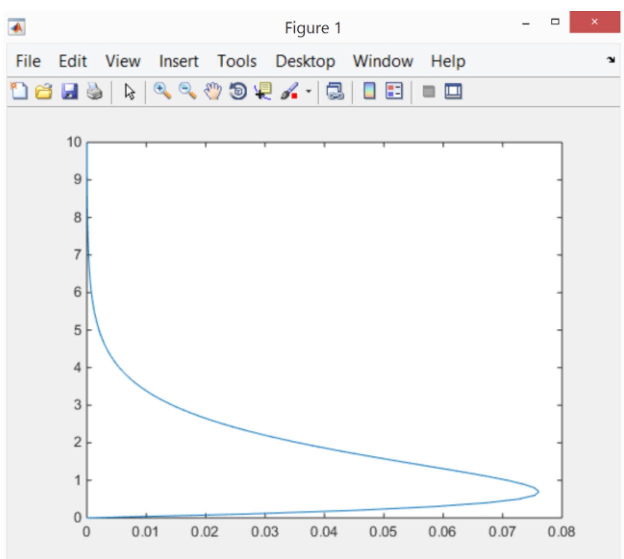
```
>> lab1Ex1MfileExc2a
Please enter the value of y0: 0.3
Please enter the value of tao: 1.5
Please enter the value of omega (in rad per second): .9
Please enter the value of teta: 0
Please input the value of 't' to evaluate the function y(t): 3
    0.0755
```

The above output demonstrate that MatLab can be use to obtain numerical values of an equation such as:

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

evaluated at the users specified values.

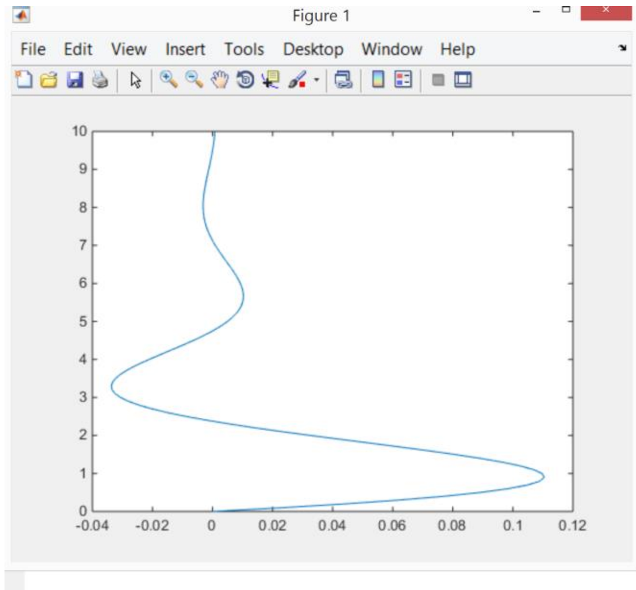
Output Part B



The above graph represent the output plot of the equation $y(t) = \frac{y0}{\sqrt{1-\delta}}\, e^{-\delta(\omega_n)t}\sin(\omega_n\sqrt{1-\ \delta^2} * t + \theta)$

With the conditions of y0=0.15 m, ωn = √ rad/sec, ζ = 3/(2√) and θ = 0

Output Part C



The above graph represent the output plot of the equation $y(t) = \frac{y0}{\sqrt{1-\delta}} e^{-\delta(\omega_n)t} \sin(\omega_n\sqrt{1-\ \delta^2} * t + \theta)$
With the conditions of y0=0.15 m, ω_n = √ rad/sec, ζ = 1/(2√) and θ = 0

Source Code Part B and C

```
%Yeraldina Estrella
%CET 4873
%Lab #1
%MatLab M-file Function Exersice
clc
clear

%{
Consider the following equation
y(t) = y0/?(1- ?)  e^(-?(?_n )t) sin?(?_n ?(1-? ??^2 )*t+?)
(a) y0=0.15 m, ?n = ? rad/sec, ? = 3/(2? ) and ? = 0;
(b) y0=0.15 m, ?n = ? rad/sec, ? = 1/(2? ) and ? = 0;
%}

disp('Consider the following equation ');
disp('y(t) = y0/sqrt(1- tao)  e^(-tao(omega_n )t) sin(teta)(omega_n sqrt(1-
(tao)^2 )*t+?)');
disp('(a) y0=0.15 m, omega_n = sqrt(2) rad/sec, tao = 3/(2*(sqrt(2)) ) and
teta = 0;');
disp('(b) y0=0.15 m, omega_n = (sqrt(2))rad/sec, tao = 1/(2*(sqrt(2))) and
teta = 0;');
choice=input('Please enter (1) for choice a) and (2) for choice b) ');

if choice==1
```

```
y0=0.15;
tao=3/(2*sqrt(2));
omegaN=sqrt(2);
angleTeta=0;

t=0:0.1:10;
y_t=(y0/(sqrt(1-tao))).*exp(-tao*omegaN.*t).*sin(omegaN.*(sqrt(1-(tao^2)).*t
+ angleTeta));
plot(y_t,t);
elseif choice==2

y0=0.15;
tao=1/(2*sqrt(2));
omegaN=sqrt(2);
angleTeta=0;
t=0:0.1:10;
y_t=(y0/(sqrt(1-tao))).*exp(-tao*omegaN.*t).*sin(omegaN.*(sqrt(1-(tao^2)).*t
+ angleTeta));
plot(y_t,t);
else
    disp('Invalid input!');

end;
```

Exercise #3

MATLAB Flow Control

Use 'for' or 'while' loop to convert degrees Fahrenheit (Tf) to degrees Celsius using the following equation . Use any starting temperature, increment and ending temperature (example: starting temperature=0, increment=10, ending temperature = 200).

Output

```
Temperature values converted from Fahrenheit to degree Celsius
Celsius:
     0

Fahrenheit:
    32

Celsius:
    10

Fahrenheit:
    50

Celsius:
    20

Fahrenheit:
    68

Celsius:
    30

Fahrenheit:
    86

Celsius:
    40

Fahrenheit:
   104

Celsius:
    50
```

```
Fahrenheit:
   122

Celsius:
    60

Fahrenheit:
   140

Celsius:
    70

Fahrenheit:
   158

Celsius:
    80

Fahrenheit:
   176

Celsius:
    90

Fahrenheit:
   194

Celsius:
   100

Fahrenheit:
   212
```

```
Fahrenheit:
   302

Celsius:
   160

Fahrenheit:
   320

Celsius:
   170

Fahrenheit:
   338

Celsius:
   180

Fahrenheit:
   356

Celsius:
   190

Fahrenheit:
   374

Celsius:
   200

Fahrenheit:
   392

>>
```

The above output shows that MatLab can be use for flow control. In this example, temperature were converted from degree Fahrenheit to degree Celsius at a specified increasing range, starting temperature and ending temperature.

Source Code

```
%Yeraldina Estrella
%CET 4873
%Lab #1
%MatLab Flow Control Exersice

clc
clear

%{
Covert temperature from Celsius values to Fahrenheit
from 0 degree celcius to 200 degree celcius with an increment of 10
%}

disp('Temperature values converted from Fahrenheit to degree Celsius');
%Converting Celsius values to Fahrenheit from 0-200 with increment of 10
for tc = 0:10:200
    tf = (9/5).*tc + 32; %Formula to convert Celsius to Fahrenheit.
    disp('Celsius: ');
    disp(tc);
    disp('Fahrenheit: ');
    disp(tf);
end
```

Conclusion

In conclusion, this laboratory introduced to MatLab programing. I learned to create M-file scripts, MatLab functions and use flow control functions. The lab was very helpful at explaining how to analyze polynomials, obtain partial fractions and matrices mathematical operations as well as plotting of functions. MatLab is a great tool for mathematical programming. Users can easily obtain polynomial solutions, partial fractions and plot of graphs by creating MatLab scripts. The manual provided to perform this lab was very informative and helpful. Nevertheless, I would had like to see how to analyze the data obtained for partial franctions. In overall, the laboratory was very descriptive at introducing the basics functions of MatLab.