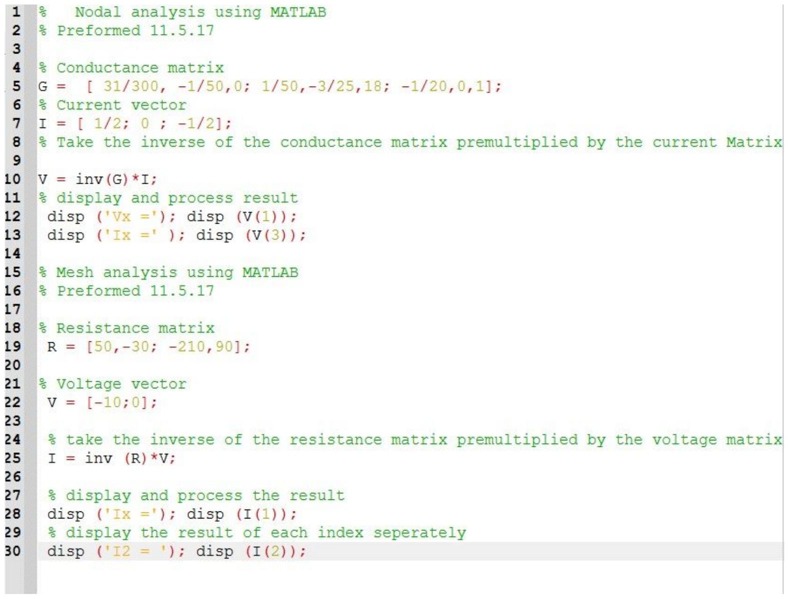
Name: **Johnathan Machler**
Partner: **Jake Wobbrock**
Course # 2006.005
Exp: **Using MATLAB**
Performed: 10.28.17
Due:11.6.17
Lab TA: Dinesh

Introduction:

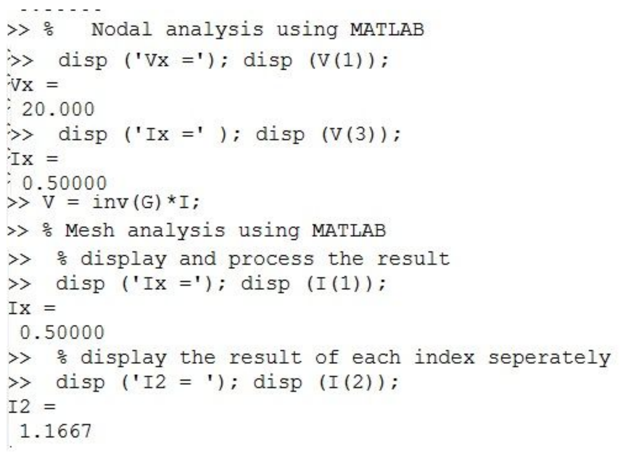
In this lab we will cover various types of basic calculations which can be performed within MATLAB. Matlab is a high level programming language which has a large library of functions any type of manipulation you can think of on a matrix hence why it's short for Matrix laboratory .

Mesh and Nodal analysis in MATLAB:



```
%   Nodal analysis using MATLAB
% Preformed 11.5.17

% Conductance matrix
G =  [ 31/300, -1/50,0; 1/50,-3/25,18; -1/20,0,1];
% Current vector
I = [ 1/2; 0 ; -1/2];
% Take the inverse of the conductance matrix premultiplied by the current Matrix

V = inv(G)*I;
% display and process result
 disp ('Vx ='); disp (V(1));
 disp ('Ix =' ); disp (V(3));

% Mesh analysis using MATLAB
% Preformed 11.5.17

% Resistance matrix
 R = [50,-30; -210,90];

% Voltage vector
 V = [-10;0];

 % take the inverse of the resistance matrix premultiplied by the voltage matrix
 I = inv (R)*V;

 % display and process the result
 disp ('Ix ='); disp (I(1));
 % display the result of each index seperately
 disp ('I2 = '); disp (I(2));
```

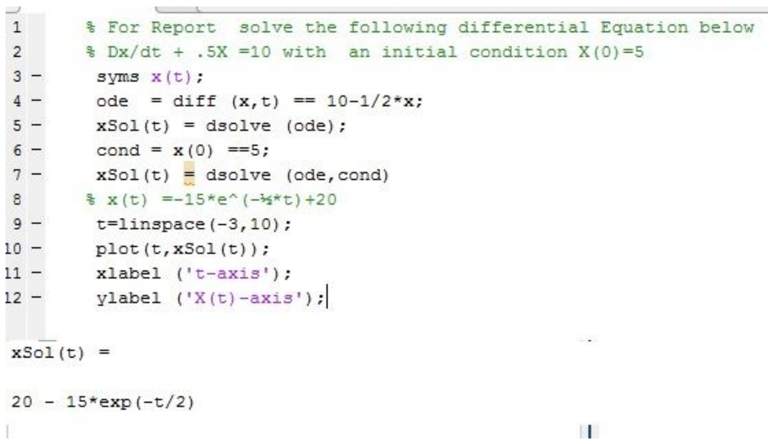
Resultant computation from the script:



```
>> %   Nodal analysis using MATLAB
>>  disp ('Vx ='); disp (V(1));
Vx =
 20.000
>>  disp ('Ix =' ); disp (V(3));
Ix =
 0.50000
>> V = inv(G)*I;
>> % Mesh analysis using MATLAB
>>  % display and process the result
>>  disp ('Ix ='); disp (I(1));
Ix =
 0.50000
>>  % display the result of each index seperately
>>  disp ('I2 = '); disp (I(2));
I2 =
 1.1667
```

※What was done in the code is self-explanatory with the comments included to guide any other user through what types of operations were done in MATLAB to said processed result. It was easy to cross-reference the result because both independent scripts should yield the same results algebraically.

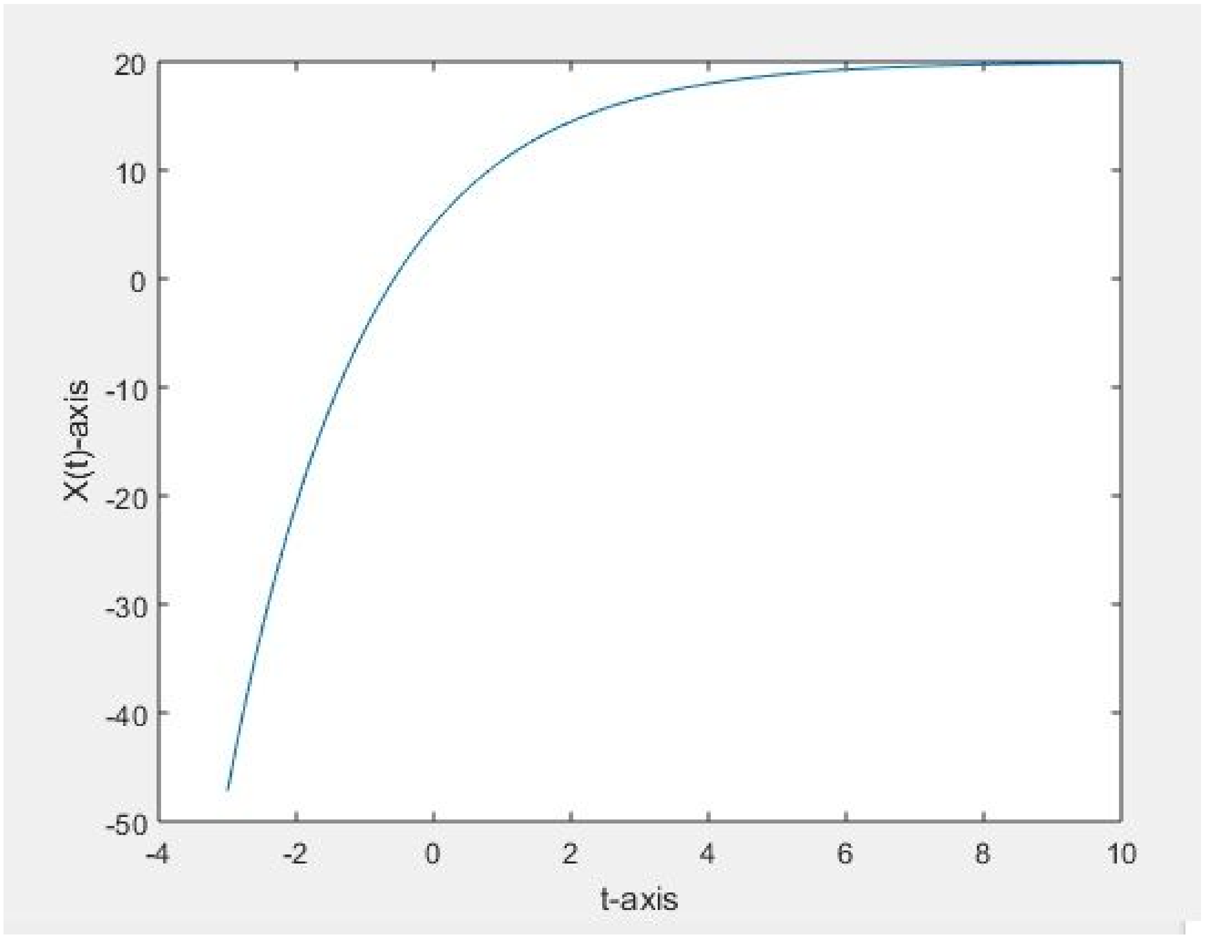
Using matlab to solve for an ODE:



```
% For Report  solve the following differential Equation below
% Dx/dt + .5X =10 with  an initial condition X(0)=5
 syms x(t);
 ode  = diff (x,t) == 10-1/2*x;
 xSol(t) = dsolve (ode);
 cond = x(0) ==5;
 xSol(t) = dsolve (ode,cond)
% x(t) =-15*e^(-½*t)+20
 t=linspace(-3,10);
 plot(t,xSol(t));
 xlabel ('t-axis');
 ylabel ('X(t)-axis');
```

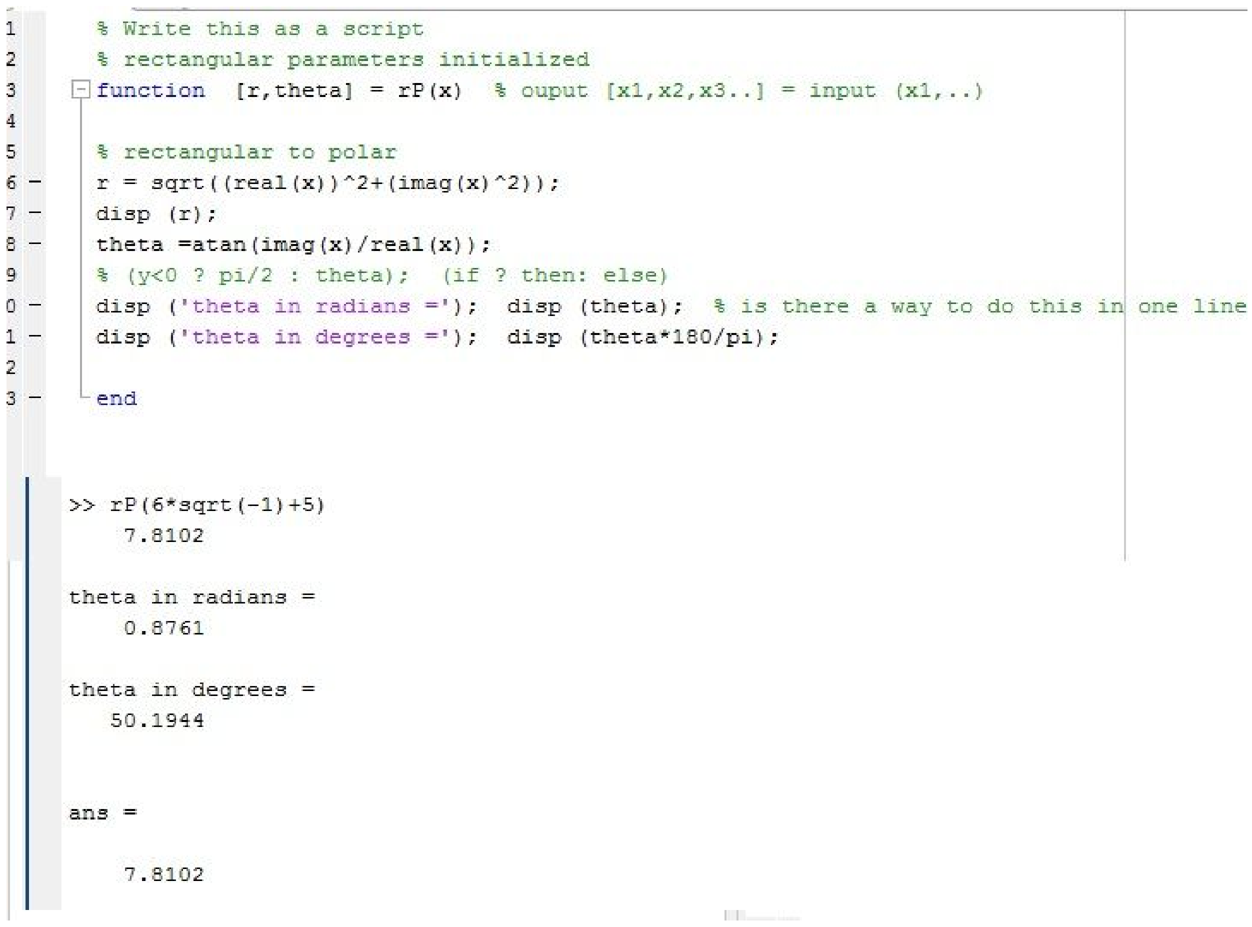
```
xSol(t) =

20 - 15*exp(-t/2)
```

Graph of the solution to the ODE given initial conditions:



This script uses a built in dsolve function from the MATLAB library you can just place the assigned differential equation along with the initial conditions into the argument of the function and it returns the solution xSol(t). I created a parametric plot by assigning a set of values that t can have using the linspace function. Then I placed the function I found set as xSol(t) a dependent variable with t as my independent variable into the arguments of the plot function. As you can see the curve should pass through the initial condition we were given in point form.

RtoP script in matlab :

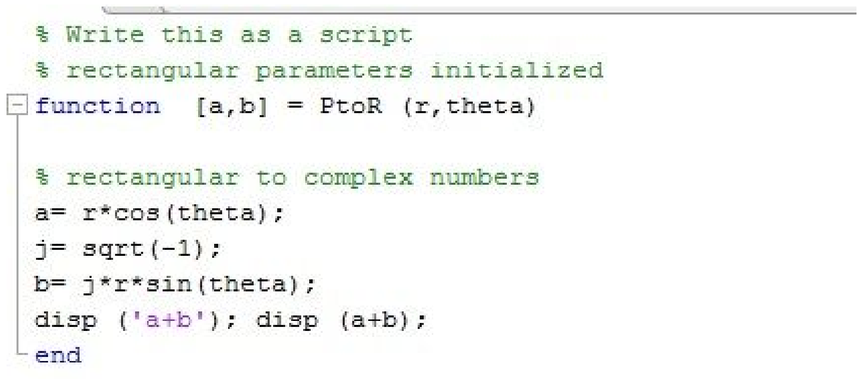


```
% Write this as a script
% rectangular parameters initialized
function  [r,theta] = rP(x)  % ouput [x1,x2,x3..] = input (x1,..)

% rectangular to polar
r = sqrt((real(x))^2+(imag(x)^2));
disp (r);
theta =atan(imag(x)/real(x));
% (y<0 ? pi/2 : theta);  (if ? then: else)
disp ('theta in radians =');  disp (theta);  % is there a way to do this in one line
disp ('theta in degrees =');  disp (theta*180/pi);

end
```

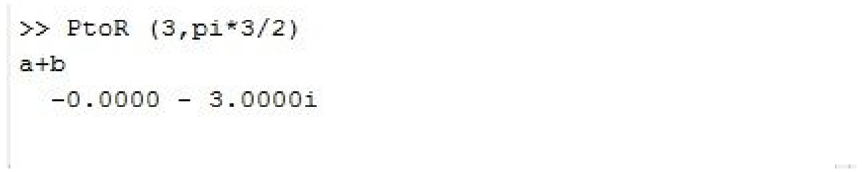
```
>> rP(6*sqrt(-1)+5)
    7.8102

theta in radians =
    0.8761

theta in degrees =
   50.1944


ans =

    7.8102
```

This script converts a complex number (rectangular) as one input into two outputs in terms of R and theta (polar form). With the comments within the program its self explanatory for the most part. These types of calculations will be invaluable once we start calculating phasors of time varying voltages etc. In the photo above below the script it was executed by calling the function in the console. One important observation I made during his lab was that the name of the .M file has to be the same as the name of the function in order for it to work.

PtoR script in matlab:



```
% Write this as a script
% rectangular parameters initialized
function  [a,b] = PtoR (r,theta)

% rectangular to complex numbers
a= r*cos(theta);
j= sqrt(-1);
b= j*r*sin(theta);
disp ('a+b'); disp (a+b);
end
```



```
>> PtoR (3,pi*3/2)
a+b
  -0.0000 - 3.0000i
```

This script converts r and theta (polar form) to a complex number. PtoR in the console is a function call the comma works as a delimiter between the two values that go within the argument of the function.

Conclusion:
Even though these programs were relatively simple more troubleshooting was required than what I would've expected. Common to many of these errors were just misunderstandings with how the syntax worked or whimsical typos that went unnoticed. Supplemental to learning what types of errors that could be made I learned how to construct a .M scripts in MATLAB as well as where to look for code that has already been created to implement it in any way need be. The main takeaway from being in lab is it's much more efficient to program once well rested earlier in the day.