Name: Andrew Brown	**Lab Time: T 12:00**

Names of people you worked with:
• Kevin • Cailin

Websites you used:
• Wolfram Alpha

Approximately how many hours did it take you to complete this assignment (to nearest whole number)?	5

By writing or typing your name below you affirm that all of the work contained herein is your own, and was not copied or copied and altered.

Andrew Brown

Note: Failure to sign this page will result in a 50-point penalty. Failure to list people you worked with may result in no grade for this homework. Failure to fill out hours approximation will result in a 10-point penalty.

Turn .zip files to Canvas or your assignment will not be graded

BEFORE YOU BEGIN

1) Please read "ENGR 112 Assignment Instructions.pdf" in the Canvas module "Start Here."

Learning Objectives:

You should be able to answer the following questions:

- What is a script file?
- What is the difference between a script file and the command window?
- How do you create variables?
- How can you examine the values of variables?
- How do you create equations, pass them variables, and print out the results?
- How do you display the values of variables?

Homework Guidelines:

1) Write one script per problem. Name each script something you'll remember (e.g. "prob1trigFunc.m").

2) Start problems with pseudo code (comments with %).
   a. Write an English description of the script/function at the top of the page.
   b. Outline your steps as comments before writing actual code.

3) Write your MATLAB statement(s) under each pseudocode step.
   a. Variables need units next to them as a comment.

4) Print results as fully formed sentences with fprintf().
   a. Use the correct number of decimal places for numbers and **include units** (if any).

5) Copy all required components into spaces below problem.

Specific Coding Notes:

Use variables for constants in equations (where it makes sense).

Example: You have the equation "$y = a + b - 2$". Given $a = 2$ and $b = 4$, find y.

☹ NO ☹	☺ YES ☺

`% Calculate y`
`y = 2 + 4 - 2;`
	`% Declare variables FIRST`
	`a = 2;`
	`b = 4;`
	
	`% Use those variables in the equation`
	`y = a + b - 2;`

Grading Checkpoints

Criteria	Component	No	Yes
[30%] Comments and Pseudocode	Declared units on all variables?		
	English description of problem at top?		
	Comments outlining your steps?		
[20%] Output formatting	Used fprintf() to make complete sentences (when required)?		
	Correct units on answers?		
	Correct number of decimal places?		
[50%] Functionality	Script computes correct value(s)?		
	Correctly converted units in script when needed?		

Problem 1

Write a script to solve and output the answers to the following equations. All quantities have no units. Assume x = 45. Format output to 5 decimal places.

$$y = \frac{5x^3\sqrt{2x^2}}{\cos(x+5)}$$, where x is in degrees

$$t = \frac{1}{\sin(\frac{1}{x})}$$, where x is radians

$$z = 2e^{x/380} + x\,ln(\frac{x^2}{3})$$

Self check: `y = xxxxxxxx.xx5xx , t= xx.xx37x and z = x9x.4xxx1`

Comments for grader/additional information (if any)

Script File

```
% Andrew Brown: Homework 1 Problem 1

clc
clear

%Practice declaring variables and preforming mathmatical computations.

%Define variables
x=45;

%Define variables as functions of x.
y=((5*x^3)*sqrt(2*x^(2)))/(cosd(x+5));
t=1/sin(1/x);
z=(2*exp(x/380))+(x*log(x^(2)/3));

%Print out x, y, t, and z in the command window.
fprintf('When x is %0.5f,\ny is %0.5f,\nt is %0.5f,\nand z is
%0.5f.\n', x, y, t, z)
```

Command Window Output

```
When x is 45.00000,
y is 45109452.96561,
t is 45.00370,
and z is 295.41351.
```

Problem 2

From elementary physics, we know that the distance a projectile travels (e.g. when fired from a cannon) depends on both the initial velocity v and the launch angle θ:

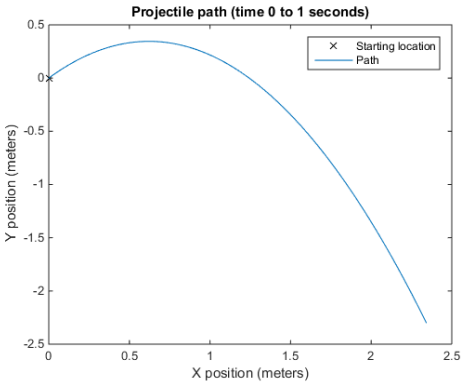
$$x(t) = v\,t\cos(\theta)$$
$$y(t) = v\,t\sin(\theta) - \frac{1}{2}gt^2$$

where *x(t)* and *y(t)* are the distances traveled in *x* and *y* respectively after time *t* has passed (*g* is gravity, *g* = 9.8 meters/second^2).

a) Given an initial velocity of 3.5 meters per second and an angle of 48 degrees above the ground, calculate the (x,y) location of the projectile at both time 0.45 seconds and 0.7 seconds.

Self-check: `x(0.45) = x.x5xx m, y(0.7) = -x.x8xx m`

b) Use your program to find the *t* value (approximately – within 0.1 meters) where the projectile hits the ground, assuming it was launched from ground height. Print out both the *t* value and the position. *Hint: The projectile goes up, then goes down… You do not need to write any new or special code to do this, just try different t values and let MATLAB do the calculation. Think of a glorified version of guess a number between 1 and 10…*



Self-check: $|y(t)| < 0.1$ for your *t* value

Format all output to 4 decimal places.

A picture to help you visualize the equations:

Comments for grader/additional information (if any)

I used wolfram alpha to solve y(t) for t.

Script File

```
% Andrew Brown: Homework 1 Problem 2

clc
clear

%Mathmatical calculations for projectile physics.

%a.)Given an initial velocity of 3.5 meters per second and an angle of
48
```

```
%degrees above the ground, alculate the (x,y) location of the
projectile
%at both time 0.45 seconds and 0.7 seconds.

%Define variables

v=3.5; %units in meters per second
theta=48; %units in degrees
g=9.8; %force of gravity in meters per second squared
t=input('Value for t in seconds:'); %user input for time in seconds.

%Calculations
xt=v*t*cosd(theta);
yt=(v*t*sind(theta))-((1/2)*g*t^2);

fprintf('At time %0.4f seconds, the (x,y) location of the projectile is
(%0.4f m, %0.4f m).\n', t, xt, yt)

%b.)Find the t value (within 0.1 m) where the projectile hits the
ground,
%assuming it was launched from ground height.

%Set yt=0 and solve for t to find the time when the projectile hits the
%ground. (Wolfram Alpha used for the simplification).
ytground=(2*v*sind(theta))/g;

%Print out the time when the projectile hits the gorund.
fprintf('The projectile hits the ground at %0.4f seconds.\n', ytground)
```

Command Window Output

```
Value for t in seconds:0.45
At time 0.4500 seconds, the (x,y) location of the projectile is (1.0539
m, 0.1782 m).
The projectile hits the ground at 0.5308 seconds.

Value for t in seconds:0.7
At time 0.7000 seconds, the (x,y) location of the projectile is (1.6394
m, -0.5803 m).
The projectile hits the ground at 0.5308 seconds.
```

Problem 3

The Richter scale is a measure of the intensity of an earthquake. The magnitude M (unitless) of an earthquake as measured on the Richter scale is

$$M = \frac{2}{3}\log_{10}\frac{E}{E_0}$$

where $E_0 = 10^{4.4}$ Joules is a constant (energy of a small reference earthquake). Write a program that takes as input (from the user) the magnitude of two earthquakes and outputs the energy difference between them. Check your answer using a magnitude 6.8 (Guatemala, 2017) and 9.0 (Japan, 2011) earthquake. Format your output in scientific notation.

Self-check: `Difference is x.x39xxxe+xx`

Comments for grader/additional information (if any)

Script File

```
%Define variables.
E0=10^(4.4); %units in joules

M1=input('The magnitude of the first earthquake:');
M2=input('The magnitude of the second earthquake:');

%These are the equations for the energy of the earthquakes using thr
above inputs.
E1=(10^((3*M1)/2))*E0; %Units in Joules

E2=(10^((3*M2)/2))*E0; %Units in Joules

%Difference between the energy of the 2 earthquakes.
E=abs(E1-E2);

%Print out the results.
fprintf('The magnitude of the first earthquake was %0.6f and released
%0.6e Joules of energy.\n', M1, E1)
fprintf('The magnitude of the second earthquake was %0.6f and released
%0.6e Joules of energy.\n', M2, E2)
fprintf('The energy difference between the two earthquaks is %0.6e
Joules.\n', E)
```

Command Window Output

```
The magnitude of the first earthquake:6.8
The magnitude of the second earthquake:9.0
The magnitude of the first earthquake was 6.800000 and released
3.981072e+14 Joules of energy.
```

```
The magnitude of the second earthquake was 9.000000 and released
7.943282e+17 Joules of energy.
The energy difference between the two earthquakes is 7.939301e+17
Joules.
```

For the following see the Full Term assignments module in Canvas – the assignments and self-checks are both there. I'd suggest at least skimming through the entire document to get an idea of what you'll be doing in subsequent weeks.

Problem 4

Teacup problem 1 [Equations]

Comments for grader/additional information (if any)

Script File

```
% Andrew Brown Homework 1 Problem 4

clc
clear

%Teacup Part 1


%Part 1.1


%Position of the center of the teacup
x0Center=0.0;
y0Center=0.0;

%Position of the handle on the teacup.
x0Handle=0.2;
y0Handle=0.5;

fprintf('Part 1.1, Positions\n  Teacup is centered at: (%0.2f, %0.2f)
meters.\n', x0Center, y0Center)
fprintf('  Teacup handle is at: (%0.2f, %0.2f) meters.\n', x0Handle,
y0Handle)


%Part1.2a


%Define constants
theta=-25; %degrees
dx=-9.52; %translation distance in meters
dy=-3.05;%translation distance in meters

%Rotation then translation of the initial points.
%Rotation of the centers.
x0RotatedCenter=(x0Center*cosd(theta))-(y0Center*sind(theta));
y0RotatedCenter=(x0Center*sind(theta))+(y0Center*cosd(theta));

%Rotation of the handles.
x0RotatedHandle=(x0Handle*cosd(theta))-(y0Handle*sind(theta));
y0RotatedHandle=(x0Handle*sind(theta))+(y0Handle*cosd(theta));
```

```
%Translation of the centers.
xCenterRTrans=x0RotatedCenter+dx;
yCenterRTrans=y0RotatedCenter+dy;

%Translation of the handles.
xHandleRTrans=x0RotatedHandle+dx;
yHandleRTrans=y0RotatedHandle+dy;

%Print results of rotating then translating the teacup center and
handle.
fprintf('Part 1.2a, rotate teacup center by %0.2f degrees then
translate by (%0.2f, %0.2f) meters.\n  Ended up at: (%0.2f, %0.2f)
meters.\n', theta, dx, dy, xCenterRTrans, yCenterRTrans)
fprintf('Part 1.2a, rotate teacup handle by %0.2f degrees then
translate by (%0.2f, %0.2f) meters.\n  Ended up at: (%0.2f, %0.2f)
meters.\n', theta, dx, dy, xHandleRTrans, yHandleRTrans)


%Part 1.2b


%Translation then rotation of the initial points.

%Translation of the centers.
x0CenterTrans=x0Center+dx;
y0CenterTrans=y0Center+dy;

%Translation of the handles.
x0HandleTrans=x0Handle+dx;
y0HandleTrans=y0Handle+dy;

%Rotation of the centers.
xTransCenterRotation=(x0CenterTrans*cosd(theta))-
(y0CenterTrans*sind(theta));
yTransCenterRotation=(x0CenterTrans*sind(theta))+(y0CenterTrans*cosd(th
eta));

%Rotation of the handles.
xTransHandleRotation=(x0HandleTrans*cosd(theta))-
(y0HandleTrans*sind(theta));
yTransHandleRotation=(x0HandleTrans*sind(theta))+(y0HandleTrans*cosd(th
eta));

%Print results of translating then rotating the teacup center and
handle.
fprintf('Part 1.2b, translate by (%0.2f, %0.2f) meters then rotate
teacup center by %0.2f degrees.\n  Ended up at: (%0.2f, %0.2f)
meters.\n', dx, dy, theta, xTransCenterRotation, yTransCenterRotation)
fprintf('Part 1.2b, translate by (%0.2f, %0.2f) meters then rotate
teacup handle by %0.2f degrees.\n  Ended up at: (%0.2f, %0.2f)
meters.\n', dx, dy, theta, xTransHandleRotation, yTransHandleRotation)
```

Command Window Output

```
Part 1.1, Positions
  Teacup is centered at: (0.00, 0.00) meters.
  Teacup handle is at: (0.20, 0.50) meters.
Part 1.2a, rotate teacup center by -25.00 degrees then translate by (-
9.52, -3.05) meters.
  Ended up at: (-9.52, -3.05) meters.
Part 1.2a, rotate teacup handle by -25.00 degrees then translate by (-
9.52, -3.05) meters.
  Ended up at: (-9.13, -2.68) meters.
Part 1.2b, translate by (-9.52, -3.05) meters then rotate teacup center
by -25.00 degrees.
  Ended up at: (-9.92, 1.26) meters.
Part 1.2b, translate by (-9.52, -3.05) meters then rotate teacup handle
by -25.00 degrees.
  Ended up at: (-9.52, 1.63) meters.
```

Problem 5

Epidemic problem 1 [Equations]

Comments for grader/additional information (if any)

Script File

```
% Andrew Brown Homework 1 Problem 5

clc
clear

% Epidemic Part 1

%Part 1.1

%Initial Values
a=10; %the contact rate: the average # of people a person comes in
contact with.
b=1.25; %the amount of time in days that a person is infectious
S0=500; %Susceptibles, those who have never had the illness and can
catch it.
I0=100; %Infectives, those who are infected and are contagious.
R0=0; % Recovered, those who already had the illness and are immune.
h=0.05; %timestep in days

%Print out parameters.
fprintf('Parameters: avg contacted, %0.0f, number of days infectious,
%0.2f\n', a, b)

%Equations used for calculations of the first timestep.
N=S0+I0+R0; %Total population
dS=(-a*S0*I0)/N; %How S changes
dI=((a*S0*I0)/N)-(I0/b); %How I changes
dR=I0/b; %How R changes

%Print out starting S, I, R, and N values.
fprintf('Starting values: S=%0.0f, I=%0.0f, R=%0.0f, total = %0.0f\n',
S0, I0, R0, N)

%Calculations for values over time using the first timestep.
S1=S0+(h*dS); %# of S after time h.
I1=I0+(h*dI); %# of I after time h.
R1=R0+(h*dR); %# of R after time h.

%Equations used for calculations of the second timestep.
dS=(-a*S1*I1)/N; %How S changes
dI=((a*S1*I1)/N)-(I1/b); %How I changes
dR=I1/b; %How R changes

%Calculations for values over time using the second timestep.
S2=S1+(h*dS); %# of S after time h.
I2=I1+(h*dI); %# of I after time h.
R2=R1+(h*dR); %# of R after time h.
```

```
%Print out ending values after 2 timesteps
fprintf('Ending values: S=%0.2f, I=%0.2f, R=%0.2f, total = %0.2f\n\n',
S2, I2, R2, N)

%Part 1.2

%Initial Values
a=10; %the contact rate: the average # of people a person comes in
contact with.
b=1.25; %the amount of time in days that a person is infectious
S0=2000; %Susceptibles, those who have never had the illness and can
catch it.
I0=100; %Infectives, those who are infected and are contagious.
R0=0; % Recovered, those who already had the illness and are immune.
h=0.05; %timestep in days

%Equations used for calculations of the first timestep.
N=S0+I0+R0; %Total population
dS=(-a*S0*I0)/N; %How S changes
dI=((a*S0*I0)/N)-(I0/b); %How I changes
dR=I0/b; %How R changes

%Print out starting S, I, R, and N values.
fprintf('Starting values: S=%0.0f, I=%0.0f, R=%0.0f, total = %0.0f\n',
S0, I0, R0, N)

%Calculations for values over time using the first timestep.
S1=S0+(h*dS); %# of S after time h.
I1=I0+(h*dI); %# of I after time h.
R1=R0+(h*dR); %# of R after time h.

%Equations used for calculations of the second timestep.
dS=(-a*S1*I1)/N; %How S changes
dI=((a*S1*I1)/N)-(I1/b); %How I changes
dR=I1/b; %How R changes

%Calculations for values over time using the second timestep.
S2=S1+(h*dS); %# of S after time h.
I2=I1+(h*dI); %# of I after time h.
R2=R1+(h*dR); %# of R after time h.

%Print out ending values after 2 timesteps
fprintf('Ending values: S=%0.2f, I=%0.2f, R=%0.2f, total = %0.2f\n\n',
S2, I2, R2, N)
```

Command Window Output

```
Parameters: avg contacted, 10, number of days infectious, 1.25
Starting values: S=500, I=100, R=0, total = 600
Ending values: S=405.75, I=184.74, R=9.51, total = 600.00

Starting values: S=2000, I=100, R=0, total = 2100
Ending values: S=1885.62, I=204.64, R=9.74, total = 2100.00
```

Problem 6

Euler leaf problem 1 [Equations]

Comments for grader/additional information (if any)

Part D) No
Part E) Taking one larger timestep is more accurate because there is no midpoint where the numbers get rounded to 4 decimals.

Script File

```
% Andrew Brown Homework 1 Problem 6

clc
clear

% Euler Leaf Part 1

%Initial Conditions
x0=5.3; %position in meters
v0=-0.5; %velocity in meters/second
m=5.2; %mass in kilograms
h=input('Timestep:'); %time in seconds


%Part A: Find the new position and velocity after taking one timestep.


%Force Function (in Newtons)
Fxv=3*cos(6*x0)*x0-(0.9*v0);

%Update Position equations
a=Fxv/m;
x1=x0+h*v0;
v1=v0+h*a;

%Print out old position, new position and timestep.
fprintf('Part A)\nOld position: %0.4f meters, new position %0.4f
meters, time step %0.4f\n', x0, x1, h)
fprintf('Old velocity: %0.4f m/s, new velocity %0.4f m/s, time step
%0.4f\n', v0, v1, h)


%Part B: The position and velocity after taking another step starting
with
%the values calculated in Part A.

%New input for time.
h=input('Timestep:');

%Force Function (in Newtons)version 2.0
Fxv2=3*cos(6*x1)*x1-(0.9*v1);
```

```
%Update Position equations version 2.0
a2=Fxv2/m;
x2=x1+h*v1;
v2=v1+h*a2;

%Print out old position, new position and timestep version 2.0
fprintf('Part B)\nOld position: %0.4f meters, new position %0.4f
meters, time step %0.4f\n', x1, x2, h)
fprintf('Old velocity: %0.4f m/s, new velocity %0.4f m/s, time step
%0.4f\n', v1, v2, h)

clear

%Part C: Calculate the new position and velocity after taking one
timestep with h=0.02.

%Initial Conditions Version 2.0
x0=5.3; %position in meters
v0=-0.5; %velocity in meters/second
m=5.2; %mass in kilograms
h=input('Timestep:'); %time in seconds

%Force Function (in Newtons)again.
Fxv=3*cos(6*x0)*x0-(0.9*v0);

%Update Position equations again.
a=Fxv/m;
x1=x0+h*v0;
v1=v0+h*a;

%Print out old position, new position and timestep again.
fprintf('Part C)\nOld position: %0.4f meters, new position %0.4f
meters, time step %0.4f\n', x0, x1, h)
fprintf('Old velocity: %0.4f m/s, new velocity %0.4f m/s, time step
%0.4f\n', v0, v1, h)
```

Command Window Output

```
Timestep:0.01
Part A)
Old position: 5.3000 meters, new position 5.2950 meters, time step
0.0100
Old velocity: -0.5000 m/s, new velocity -0.4708 m/s, time step 0.0100
Timestep:0.01
Part B)
Old position: 5.2950 meters, new position 5.2903 meters, time step
0.0100
Old velocity: -0.4708 m/s, new velocity -0.4413 m/s, time step 0.0100
Timestep:0.02
Part C)
Old position: 5.3000 meters, new position 5.2900 meters, time step
0.0200
Old velocity: -0.5000 m/s, new velocity -0.4416 m/s, time step 0.0200
```