Name: Andrew Brown	**Lab Time** *(ECampus write "ECampus")*: T 12:00

Names of people you worked with:

- Kevin Daellenbach
- Caillin Moore

Websites you used:

-

Approximately how many hours did it take you to complete this assignment (to nearest whole number)?	6

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

Learning Objectives:

- Fitting with polynomial
  - Fitting a pure polynomial
  - Using polynomials to fit non-polynomial functions
- Script correction

Homework Guidelines:

1) Make sure you plot the data and check that your fitting result is reasonable (the line approximately matches the points).
2) More practice re-using code and making functions.
3) More practice with making code "general".
   a. We use dlmread to read in data rather than defining it in the script so we may later change the data without changing the script.

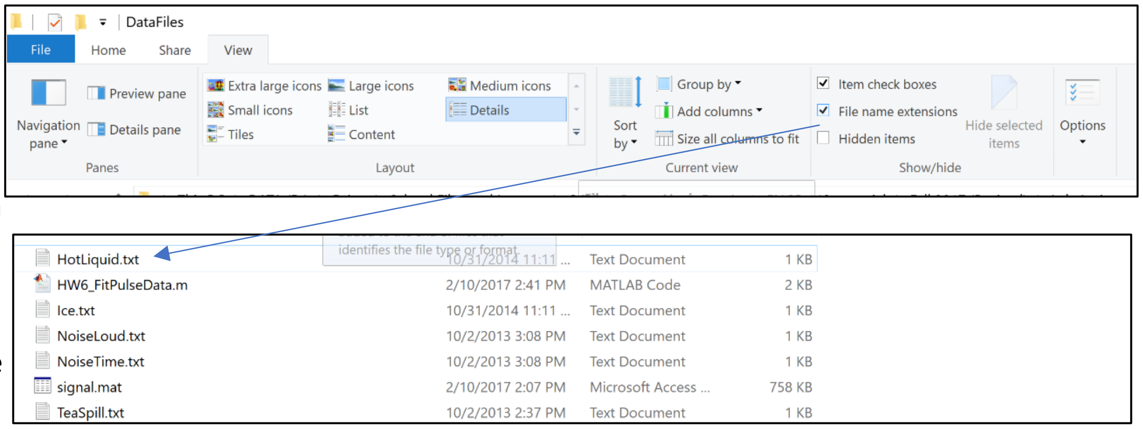
Specific Coding Notes:

Files to download to your Homework6 Folder

- Download the zipped folder DataFiles.zip from Canvas. Unzip it and put all the files in the same folder as Homework6.
  - Test you can read them using the command line and this line of code: `hotWater = dlmread('HotWater.txt');`

Here's common problems you'll want to look out for:

- Syntax: `variable = dlmread('text_file.txt')`
  - `dlmread()` takes a string – make sure you use the correct quotation marks ('')
- The file HotWater.txt must be in the **same directory (file folder)** as your matlab scripts. Otherwise, you will get an error saying "file not found".
  - You may put the text files in a subdirectory and use the file address like `dlmread('subdirname/HotWater.txt');`
- Sometimes, depending on how you download and save the files, Windows will append a different ending to your filename. Do view->file extensions in the file explorer to see if this is the case.
- I f a l l e l



HotLiquid.txt	10/31/2014 11:11 ...	Text Document	1 KB
HW6_FitPulseData.m	2/10/2017 2:41 PM	MATLAB Code	2 KB
Ice.txt	10/31/2014 11:11 ...	Text Document	1 KB
NoiseLoud.txt	10/2/2013 3:08 PM	Text Document	1 KB
NoiseTime.txt	10/2/2013 3:08 PM	Text Document	1 KB
signal.mat	2/10/2017 2:07 PM	Microsoft Access ...	758 KB
TeaSpill.txt	10/2/2013 2:37 PM	Text Document	1 KB

se fails, open up the file with the Matlab editor (pick show all in the file dialog). Make sure you save the file as HotWater.txt, with a .txt ending, in the matlab directory.

- DO NOT copy and paste the values in your script as a variable. Use `dlmread()`.

Grading Checkpoints

Criteria	Component	No	Yes
[20%] Comments and Pseudocode	Declared units on all variables?		
	English description of problem at top?		
	Comments outlining your steps?		
[10%] Output formatting	Used fprintf() to make complete sentences (when required)?		
	Correct units on answers?		
	Correct number of decimal places?		
[70%] Functionality	Script computes correct value(s)?		
	Correctly converted units in script when needed?		
	Used dlmread instead of copying and pasting values		
	Make sure textfiles are in the right folder		

We have bolded and italicized primary deliverables for each problem as per request of students to have clearer instructions.

Problem 1

The following data represents the volume of ice needed to chill a given volume of hot liquid. (Use HotLiquid.txt and Ice.txt from the DataFiles.zip folder. See instructions above for help.)

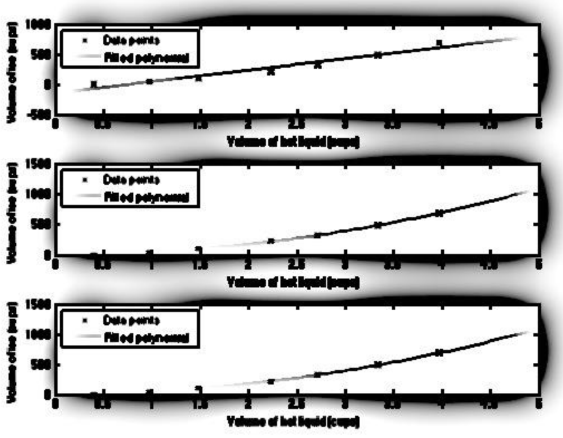
Volume of Hot Liquid (cups)	0.40	0.97	1.48	…	3.34	3.97
Volume of Ice (cups)	8.28	43.04	98.03	…	487.7	687.84

a) ***Fit a 1st, 2nd, 3rd, and 4th degree polynomial to the dataset described by volume of liquid vs. volume of ice.*** Use subplot to draw all four. Plot your dataset as points, and your fitted equation as a solid line in the domain 1 ≤ Vwater ≤ 5. Use a legend to identify the data points and the fitted equation.

b) Which degree do you think best represents the data? Why? (Hint: how much ice is needed to chill 0 cups of hot liquid?) ***Answer in the appropriate box.***

c) Use your chosen degree polynomial to find and ***print*** the volume of ice needed to chill 1.5 cups of hot liquid.

Self-Check:
Part a).



Part c): Answer will vary depending on the degree chosen, but you can check the graph to make sure the answer is in the right ballpark.

Comments for grader/additional information (if any)

Script File

```
%Andrew Brown Homework 6 Problem 1

clc
clear
```

```
close all

%Practice reading files and using polynomial functions

hot=dlmread('HotLiquid.txt'); %read hotliquid.txt
cold=dlmread('Ice.txt'); %read ice.txt

legendLabel={'Deg 1 Polynomial','Deg 2 Polynomial','Deg 3 Polynomial','Deg 4
Polynomial'}; %legend labels
for i=1:4
    P=polyfit(hot,cold,i); %create polynomials of varying degrees
    polyFunc=@(hot) polyval(P,hot); %evaluate the polynomials at the given
values of hot
    subplot(4,1,i) %make subplots
    plot(hot,cold,'rx'); %plot original data points
    hold on
    fplot(polyFunc,'k','Linewidth',1) %plot your x values (hot) vs the
evaluated polynomial functions
    axis([1,5,0,1000]) %define axes
    xlabel('Volume of hot liquid (cups)') %define x label
    ylabel('Volume of Ice (cups)') %define y label
    legend('Data Points',legendLabel{i}) %define legend
    if i==2 %print out the volume of ice needed to chill 1.5 cups of hot
liquid for the 2nd degree polynomial
        fprintf('The volume of ice needed to chill 1.5 cups of hot liquid
is %0.0f cups\n', polyFunc(1.5))
    end
end
```

Function Files

```
% Copy and paste your functions here. Must be size 10, same as MATLAB font
and color.
```

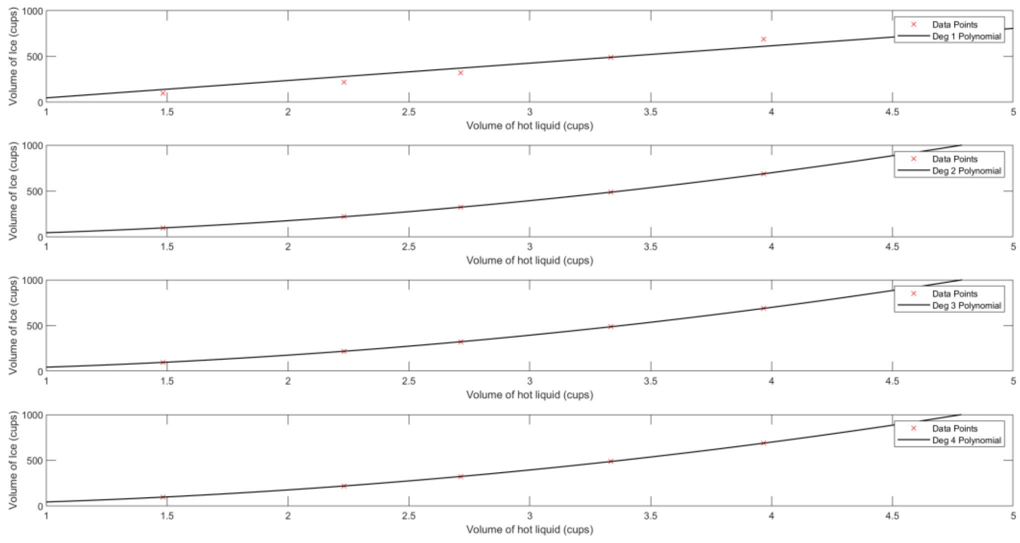
Command Window Output

```
The volume of ice needed to chill 1.5 cups of hot liquid is 100 cups
```

Answers to question(s) asked in the homework (if any)

b) Second degree

Image Output


Volume of Ice (cups)
Volume of hot liquid (cups)
Data Points
Deg 1 Polynomial
Deg 2 Polynomial
Deg 3 Polynomial
Deg 4 Polynomial

Problem 2

Perform a data linearization similar to Lecture 6-2. This question has 5 parts:

a) Create a simulated data set
b) Create a set of plots to determine the best transformation to linearize the data
c) Fit a first order polynomial (y = mx + b) to the linearized data to find the constants m and b
d) Reconstruct the original equation from m and b in the linear fitted polynomial
e) Plot the fitted equation.

Radioactive decay is modeled by the equation:

$$A = A_0 e^{rt}$$

where

A = the amount of mass as a function of time (mg)
A_0 = the original mass of decaying material (mg)
r = the decay rate
t = the amount of time elapsed (years)

a) Generate an set of mass measurements over time for Radium-226 (decay rate: 0.0436% per year). Use $0 \le t \le 4000$ years with r = -0.000436 and A_0 = 20mg to generate 20 *evenly spaced* measurements of A. Next, add noise to your A vector to represent measurement error. Do this with the following code:

```
addSampleNoise = (rand([1 numberOfSamples])-0.5) .* (original_data_A.*0.1);
A_noisy = original_data_A + addSampleNoise;
```

- This code generates a vector of length *nSamples* containing random values between -0.5 and +0.5, then scales each measurement by 10%.
- *original_data_A* is the A that you created above. The end result (*A_noisy*) has an artificial measurement error of up to 5% per measurement.
- Feel free to change the variable names.

Print the first and last values of A_noisy to four decimal places.

b) Determine which type of function provides the best fit by plotting the noisy data using the following transformations. Use the subplot command to include them in the same window.

x-axis (your time data)	**y-axis** (A_noisy)	**Function**
x	y	Linear - y = mx + b
ln(x)	ln(y)	Power - $y = bx^m$
x	ln(y)	Exponential - $y = be^{mx}$
ln(x)	y	Logarithmic - y = m*ln(x)+b

The best function is the one whose data points are closest to a straight line. Run the script, decide upon the best function, and ***output your choice of function to the command window (linear, power, exponential, or logarithmic***). (*Hint: Does your choice match the form of the equation in part a?)*

c) Use **polyfit** to find a first order polynomial (degree n = 1) that fits the linearized data. For example, if you choose the Power function, then you should do a polyfit using ln(x) and ln(y) as shown in the table above. ***Print the coefficient values of m and b (in y=mx+b) to 6 decimal places.***

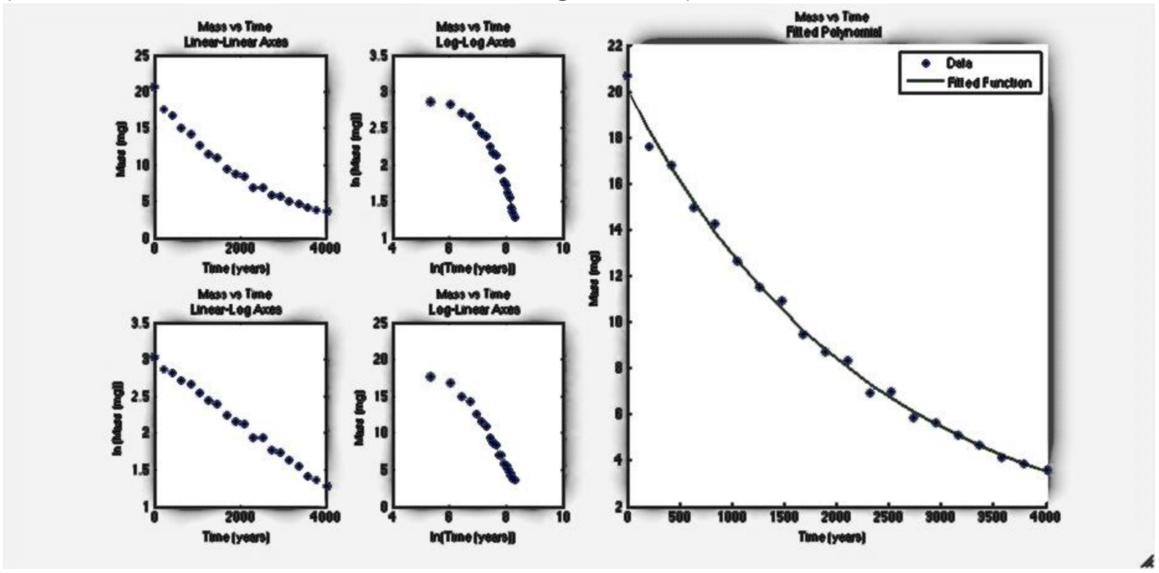
d) Convert *m* and *b* back into A_0 and *r*. ***Print*** A_0 ***and r to 6 decimal places***. See the slides "Why does this work?" and "Now reconstruct function" in Lecture 6-2.

e) Use your fitted A_0 and *r* values to ***plot the fitted function and the data on the same plot***.

Self-check:

a)
```
First noisy sample: XX.XXXX mg at t = 0 years
Last noisy sample: 3.XXXX mg at t = 4000 years
```

b & e)
(b is shown on the left below, e is on the right below)



c & d)
No self check. The solutions that you get back should be very close to your original A_0 and r values, but they probably won't be exactly the same. The fitted function plot should closely match your noisy data points.

Comments for grader/additional information (if any)

Script File

```
%Andrew Brown Homework 6 Probelm 2

clc
clear

%Data linearization with a simulated data set

%Generate measurments over time of a radium sample
A0=20; %mg of radium
r=-0.000436; %decay rate
t=linspace(0,4000,20); %time in years
A=@(t) A0.*exp(r*t); %amount of mass of radium in mg as a function of time
data_A=A(t); %find 20 evenly spaced measurments of A from 0 to 4000 years

%adds random noise with error of up to +- 5% to the data
addSampleNoise=(rand(size(t))-0.5.*(data_A.*0.1));
A_noisy=data_A+addSampleNoise;

%print the first and last values of A_noisy
fprintf('First noisy sample: %0.4f mg at t = 0 years\nLast noisy
sample: %0.4f mg at t = 4000 years\n',A_noisy(1),A_noisy(end))

%Plot 4 differt transformations of the data

%Plot linear plot
subplot(2,4,1) %plot in first position
plot(t,A_noisy,'.k','Markersize',12) %Linear
xlabel('Time (years)') % x label
ylabel('Mass (mg)') % y label
title('Mass vs. Time Linear-Linear Axes') %title the plot
axis([0,4000,0,25]) %proper axes

%Plot power plot
subplot(2,4,2) %plot in second position
plot(log(t),log(A_noisy),'.k','Markersize',12) %Power
xlabel('ln(Time) (years)') % x label
ylabel('ln(Mass) (mg)') % y label
title('Mass vs. Time Log-Log Axes') %title the plot
axis([4,10,1,3.5]) %proper axes

%Plot Exponential plot
subplot(2,4,5) %plot in 5th position leaving 3 and 4 blank
plot(t,log(A_noisy),'.k','Markersize',12) %Exponential
xlabel('Time (years)') % x label
ylabel('ln(Mass) (mg)') %y label
title('Mass vs. Time Linear-Log Axes') %title the plot
axis([0,4000,1,3.5]) %proper axes
```

```
%Plot Logarithmic plot
subplot(2,4,6) %plot in sixth position
plot(log(t),A_noisy,'.k','Markersize',12) %Logarithmic
xlabel('ln(Time) (years)') % x label
ylabel('Mass (mg)') % y label
title('Mass vs. Time Log-Linear Axes') %title the plot
axis([4,10,0,25]) %proper axes

%Plot the linear-log data with its line of best fit as an exponential
subplot('Position',[0.55,0.1,0.4,0.85]) %properly position the new plot
P=polyfit(t,log(A_noisy),1); %get m and b coefffficients
fprintf('Polynomial is y=mt+b with m = %0.6f and b
= %0.6f\n',P(1),P(2)) %print out m and b coefficients
polyFunc=@(t) exp(P(2)).*exp(P(1).*t); %find an exponential fnction with the
same coeffs
fplot(polyFunc,'k') %plot the fit function
hold on
plot(t,A_noisy,'.k','Markersize',12) %plot the original data on the same
plot
xlabel('Time (years)') % x label
ylabel('Mass (mg)') % y label
title('Mass vs. Time Fitted Polynomial') %title of graph
legend('Fitted Function','Data') %make a legend
```

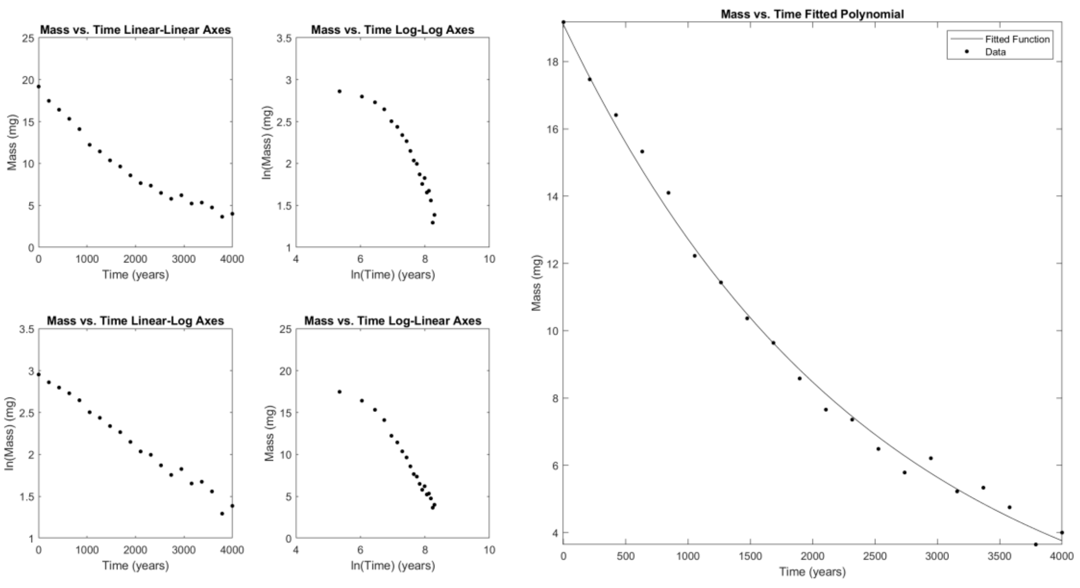
Function Files

```
% Copy and paste your functions here. Must be size 10, same as MATLAB font
and color.
```

Command Window Output

```
First noisy sample: 19.7060 mg at t = 0 years
Last noisy sample: 3.3548 mg at t = 4000 years
Polynomial is y=mt+b with m = -0.000408 and b = 2.966572
```

Image Output


Mass vs. Time Linear-Linear Axes
Mass vs. Time Log-Log Axes
Mass vs. Time Linear-Log Axes
Mass vs. Time Log-Linear Axes
Mass vs. Time Fitted Polynomial
Fitted Function
Data
Mass (mg)
ln(Mass) (mg)
Time (years)
ln(Time) (years)

Problem 3

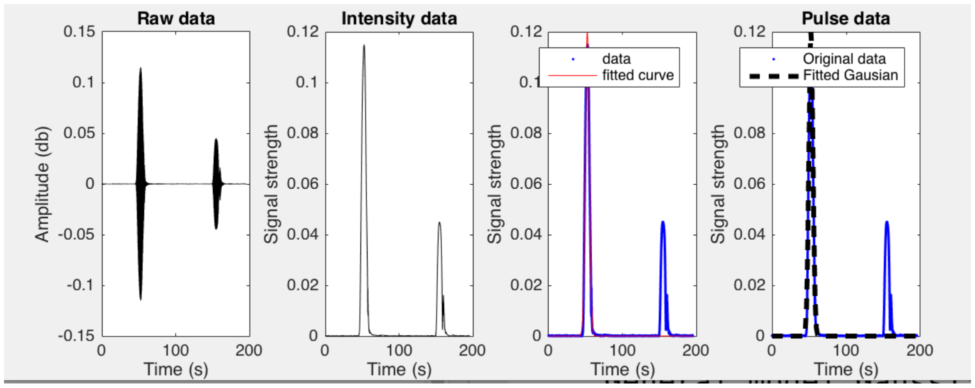
Fix an existing script so it performs correctly. The script HW6_FitPulseData currently reads in a signal from a file (signal.mat) and fits a single Gaussian (a bell shaped curve) to it. However, as you see below, the data has **two** peaks, not one, so we must fix the script to fit **two** Gaussians: one to the first peak, one to the second.

To-do:

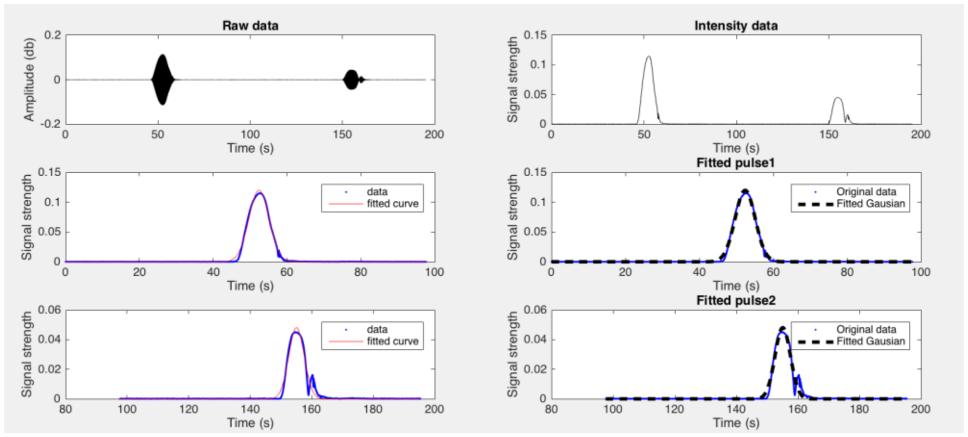
1) Break the data into two time periods and fit one Gaussian to each time period. ***Plot the results.***
2) ***For each bell curve, print out when the peak occurs, the value of the peak, and the standard deviation.***
   a. Hint: Print out gaussFitted by typing `gaussFitted` in the command line. You will see what coefficients are in the function. You will need to Google meaning of the coefficients. You can retrieve each of these coefficients by typing `gaussFitted.a1`, `gaussFitted.b1`, etc.

Self-check:

Before:



After:



(Numbers may vary slightly)

```
Gaussian 1 centered at 5X.3X, peak 0.1XX, sd 3.XX4
Gaussian 2 centered at 15X.0X, peak 0.04XX, sd 3.XX1
```

For the curious, this is actual sound data captured from a bat – each of the peaks is a "chirp" generated by the bat. This is part of a project to try to understand how the bats use sonar and how the sounds they make are related to ear and nose movement.

Comments for grader/additional information (if any)

Script File

```
%% Script for fitting a gaussian to a signal
% Edit this script to produce a decent fit

clear;
clc;
clf;

%% Load the data in - you can save variables using the 'save' command,
%   then read them in using the 'load' command. Very similar to dlmread
%   and dlmwrite, but for any matlab variable
dataLoaded = load('signal.mat');
dataRaw = dataLoaded.data;  % Get data out
dataRaw1=dataRaw([1:floor(end/2)]);
dataRaw2=dataRaw([ceil(end/2):end]);
% Convert from signal to intensity at frequency
dataIntensity = abs( hilbert(dataRaw) );
dataIntensity1 = abs( hilbert(dataRaw1) );
dataIntensity2 = abs( hilbert(dataRaw2) );
% Data rate is 512 frames per second
lengthFrame = 1.0/512.0; % in seconds
times = 0:lengthFrame:lengthFrame * (length(dataIntensity)-0.5);
times1 = 0:lengthFrame:lengthFrame * (length(dataIntensity1)-0.5);
times2 = (5e4)/512+(0:lengthFrame:lengthFrame * (length(dataIntensity2)-
0.5));

% Plot raw signal data
nRows = 3;
nCols = 2;
subplot(nRows, nCols, 1);
plot( times, dataRaw, '-k' );
xlabel('Time (s)');
ylabel('Amplitude (db)')
title('Raw data');
axis([0,200,-0.2,0.2])

% Plot envelope
subplot(nRows, nCols, 2);
```

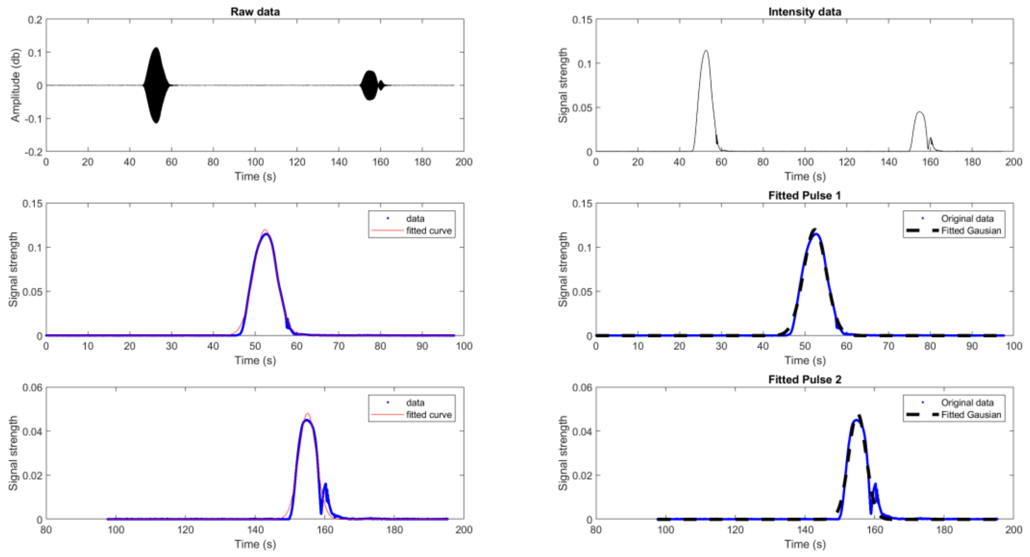
```
plot( times, dataIntensity, '-k' );
xlabel('Time (s)');
ylabel('Signal strength')
title('Intensity data');
axis([0,200,0,0.15])

% Fit a Gaussian to the entire thing
gaussFitted = fit( times', dataIntensity', 'gauss1' );
gaussFitted1 = fit( times1', dataIntensity1', 'gauss1' );
gaussFitted2 = fit( times2', dataIntensity2', 'gauss1' );

% plot the result
subplot(nRows, nCols, 3);
% plot 'knows' what kind of data gaussFitted is, so it plots the right
% thing
plot(gaussFitted1, times1, dataIntensity1);
xlabel('Time (s)');
ylabel('Signal strength')
axis([0,100,0,0.15])

% plot the result
subplot(nRows, nCols, 5);
% plot 'knows' what kind of data gaussFitted is, so it plots the right
% thing
plot(gaussFitted2, times2, dataIntensity2);
xlabel('Time (s)');
ylabel('Signal strength')
axis([80,200,0,0.06])

% Evaluate the fitted Gaussian and plot
%   You can treat gaussFitted as a function and 'call' it with t value
fittedValues1 = gaussFitted1(times1);
subplot(nRows, nCols, 4);
plot( times1, dataIntensity1, '.b' );
hold on;
plot( times1, fittedValues1, '--k', 'LineWidth', 3 );
legend('Original data', 'Fitted Gausian');
xlabel('Time (s)');
ylabel('Signal strength')
title('Fitted Pulse 1');
axis([0,100,0,0.15])
fprintf('Gaussian 1 centered at %0.2f, peak %0.4f,
sd %0.4f\n',gaussFitted1.b1,gaussFitted1.a1,gaussFitted1.c1)


% Evaluate the fitted Gaussian and plot
%   You can treat gaussFitted as a function and 'call' it with t value
fittedValues2 = gaussFitted2(times2);
subplot(nRows, nCols, 6);
plot( times2, dataIntensity2, '.b' );
hold on;
plot( times2, fittedValues2, '--k', 'LineWidth', 3 );
legend('Original data', 'Fitted Gausian');
xlabel('Time (s)');
ylabel('Signal strength')
title('Fitted Pulse 2');
```

```
axis([80,200,0,0.06])
fprintf('Gaussian 2 centered at %0.2f, peak %0.4f,
sd %0.4f\n',gaussFitted2.b1,gaussFitted2.a1,gaussFitted2.c1)
```

Command Window Output

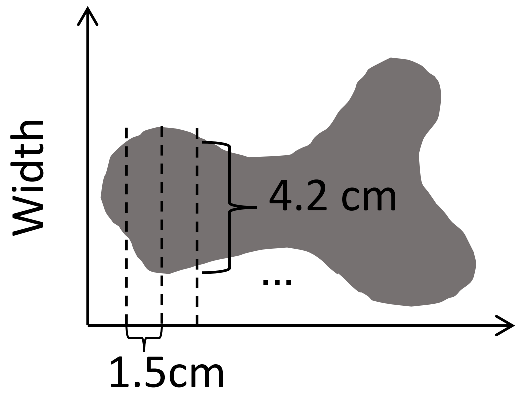
```
Gaussian 1 centered at 52.39, peak 0.1198, sd 3.8480
Gaussian 2 centered at 155.07, peak 0.0481, sd 3.5402
>>
```

Image Output



Problem 4

The surface area of a tea spill is estimated by measuring the width (in cm) of the spill at equally spaced intervals of 1.5 cm (width data is found in TeaSpill.txt). Use numerical integration to estimate the area of the tea spill. Plot the width.



Self check: Area of tea spill 8X.X78 (cm^2)

Comments for grader/additional information (if any)

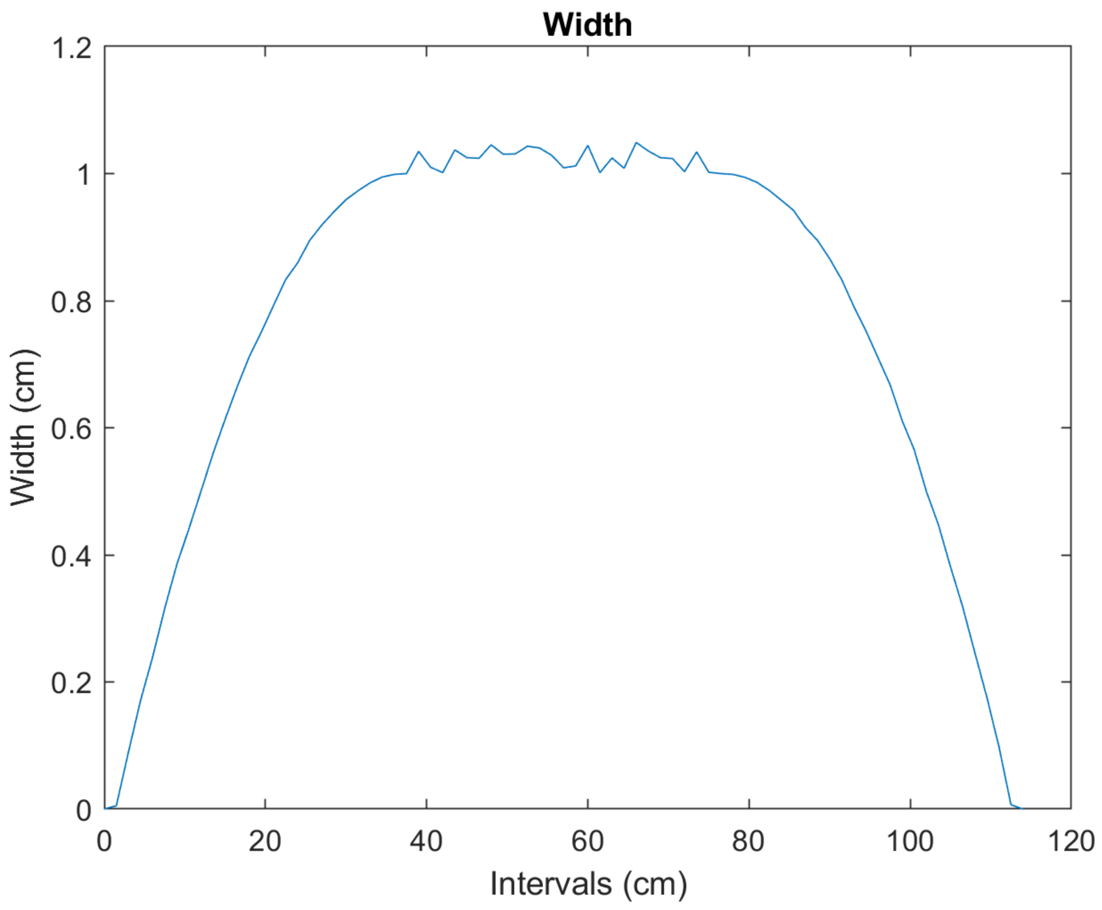
Script File

```
%Andrew Brown Homework 6 Problem 4

clc
clear
close all

%Find area of tea spill
widthData=dlmread('TeaSpill.txt'); %read the teaspill width (cm) data
int=0:1.5:1.5*length(widthData)-1; %make 1.5 cm intervals
area=trapz(int,widthData); %calculate area using the correct intervals
fprintf('Area of Tea Spill: %0.3f (cm^2)\n', area) %print out the calcuated
area
plot(int,widthData) %plot the width
xlabel('Intervals (cm)') %x label
ylabel('Width (cm)') %y label
title('Width') %title
```

Command Window Output

```
Area of Tea Spill: 88.278 (cm^2)
```

Image Output



Problem 5 [Extra credit]

Choose any previous HW problem of your choice where it is appropriate to save the data in a text file.

Copy your code from that homework problem and add a dlmwrite function that saves the data output in a file. ***You need to only submit the .m file***; no need to upload the output text file. You will get credit only if the dlmwrite function works successfully.

Comments for grader/additional information (if any)

Homework 3 Problem 1

Script File

```
%Andrew Brown Homework 6 Problem 5

clc
clear
close all

%Physics of a harmful fall on different planets using loops

%Define given constants
massPounds=120; %mass in pounds
massKg=massPounds/2.246; %mass in kilograms
PlanetNames={'Mercury','Venus','Earth','Mars','Jupiter','Saturn','Uranus','Ne
ptune','Pluto'}; %array of planet names
newPlanet=input('New Planet Gravity:'); %user input for a new planet's
gravity in m/s^2
gValues=[3.61, 8.83, 9.81, 3.75, 26.0, 11.2, 10.5, 13.3,
0.61]; %accelerations due to gravity in m/s^2
dlmwrite('gValues.txt',gValues)

%If the new planet is identical in size to an existing planet, say so, if
%not, add the new planet to gValues and run it through the rest of the
%code
if newPlanet~=gValues
    gValues(10)=newPlanet; %adds the new planet gravity to a new index in
gValues
else
    EqualPlanet=find(newPlanet==gValues); %stores the new planet as the array
index of its equal planet in gValues
    fprintf('The size of your planet is identical
to %s\n',PlanetNames{EqualPlanet(1)}) %Prints the planet that is equal to the
new planet input
end
h=linspace(1,200,200); %height off the ground in meters

%Calculate the energy given off by falling on each planet from variable
%heights
EPlanetStore=zeros(1,length(h)); %Pre-allocate the energy for the fall on
each plannet
```

```
for i=1:length(gValues)
    EPlanet=massKg*gValues(i)*h; %energy generated from the fall in joules on
earth
    EPlanetStore(i, :)=EPlanet;
end

%Calculate the height of an equivalent fall on differnet planets
PlanetHeightStore=zeros(1,length(h)); %Pre-allocate the individual planet
heights
for k=1:length(gValues)
    PlanetHeight=EPlanetStore(3,10)/(gValues(k)*massKg);
    PlanetHeightStore(k)=PlanetHeight;
end

%Plotting the graph for energy generated from a fall on Earth and the
%equivalent falls on different planets
for j=1:length(gValues)
    hold on
    plot(h,EPlanetStore(j, :))
    plot(PlanetHeightStore(j),EPlanetStore(3,10),'r*')
end

%Misc. graph labels, legend, and defining axes.
title('Energy Generated from a fall on Different Planets')
xlabel('Height (m)')
ylabel('Energy (J)')
axis([0,200,0,10^4])

%Make array for the legend labels so we can exclude the new planet labels
%if they do not exist
legendLabels={'Mercury','Equivalent fall for Mercury','Venus',...
    'Equivalent fall for Venus','Earth','Equivalent fall for Earth',...
    'Mars','Equivalent fall for Mars','Jupiter',...
    'Equivalent fall for Jupiter','Saturn','Equivalent fall for Saturn',...
    'Uranus','Equivalent fall for Uranus','Neptune',...
    'Equivalent fall for Neptune','Pluto','Equivalent fall for Pluto',...
    'New Planet','Equivalent fall for New Planet'};
legend(legendLabels(1:length(gValues)*2)) %make the legend relying off the
length of gValues
```

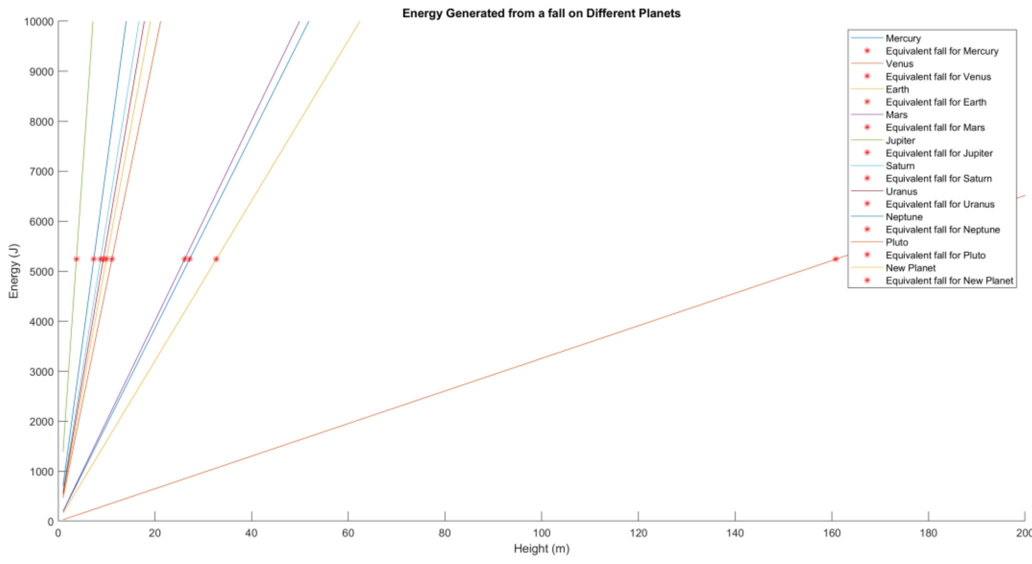
Function Files

```
% Copy and paste your functions here. Must be size 10, same as MATLAB font
and color.
```

Command Window Output

```
New Planet Gravity:3
>>
```

Image Output


Energy Generated from a fall on Different Planets
Energy (J)
Height (m)
Mercury
Equivalent fall for Mercury
Venus
Equivalent fall for Venus
Earth
Equivalent fall for Earth
Mars
Equivalent fall for Mars
Jupiter
Equivalent fall for Jupiter
Saturn
Equivalent fall for Saturn
Uranus
Equivalent fall for Uranus
Neptune
Equivalent fall for Neptune
Pluto
Equivalent fall for Pluto
New Planet
Equivalent fall for New Planet