Name: Andrew Brown	**Lab Time: T 12:00**
People Worked With: Cailin Moore	**Websites Used:n/a**
Time spent on zyBooks (hrs):2	**Time spent on lab (hrs): 2**
Submission Instructions Turn all work in to Lab 4 on Gradescope (PDF) and Canvas (.zip file), even if it is not complete yet. If you are not finished, complete the assignment outside of lab and re-submit to Lab 4 on Gradescope and Canvas. All labs are typically due at the same time on Monday every week, but check Canvas if in doubt.	

Learning objectives:

- Create (and call) a simple function file.
- Create (and call) a simple anonymous function.
- Use an anonymous function to create a function with one parameter from a function with multiple parameters.

New MATLAB commands

These are highlighted in **bold** in the instructions below.

-
  ```
  function [output] = functionName(input1, input2)
        % What the function does; make sure to set output
  end
  ```
- `functionName = @(input1, input2) code; % anonymous function declaration`
- `fplot( functionName, start, stop ); % plot the function from start to stop`
- `axis equal % make sure the aspect ratio of the two axes are the same (ie, a circle will show up as a circle, not a squashed ellipse).`

Lab Problems

Files to download to your Lab 4 Folder

- *Lab4Prob3CountCommas.m*
- *DataFileLab4.txt*

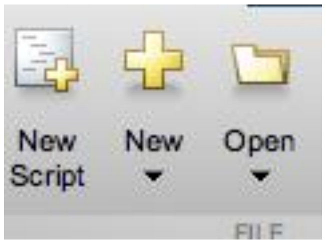
Getting Started

Check that Lab4Prob3CountCommas works – run the script, you should see:

```
Found 2 quoted strings and 4 commas
Comma locations:      14     25    123    125

>>
```

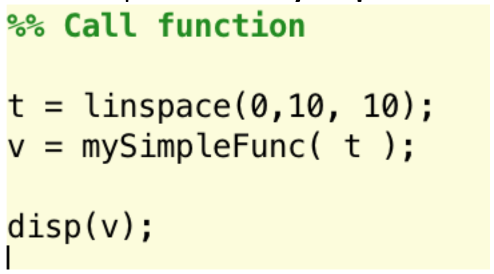
If it didn't work, make sure the DataFileLab4.txt file is in the same directory

Practice creating a function

- Create a function file that evaluates a simple equation and returns the answer
  - Create a new *function* script
  - Click on the arrow below New and pick Function
    - 

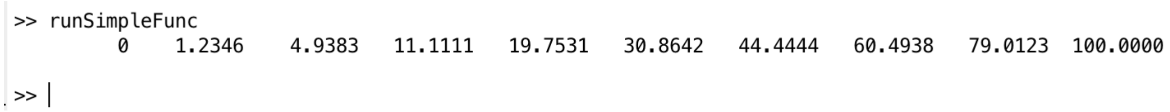
  - You should see a script appear with this in it:
    -
```
function [ output_args ] = untitled( input_args )
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here


end
```
  - Change output_args to be y, untitled to be mySimpleFunc, and input_args to be x
    -
```
function [ y ] = mySimpleFunc( x )
%mySimpleFunc Practice function to do y = x^2

end
```
  - Now write the y = x^2 function – make sure to use element-wise exponentiation so this function will work with an array
  - BTW, you MUST have a y = [blah] somewhere in this function

  - Use a different variable name for the output than the input
    - If you're editing a variable you can do
      - function [xOut] = mySimpleFunc(xIn)
      - … to make it clear which is the in and which is the out
- Save the function
  - You MUST save this function to mySimpleFunc.m – this is MATLAB's default, don't change it
  - Also save it in this week's lab directory or matlab won't be able to find it
- Create a new script and call **mySimpleFunc** in it



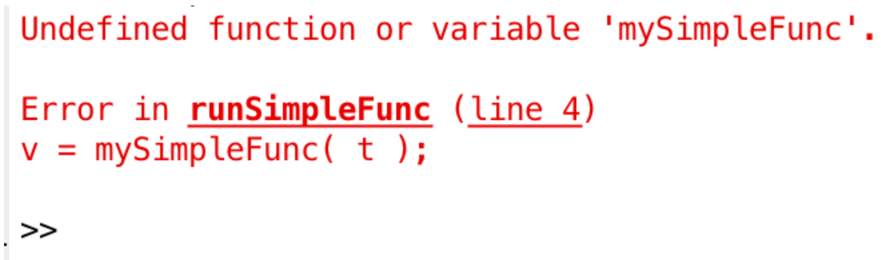
```
%% Call function

t = linspace(0,10, 10);
v = mySimpleFunc( t );

disp(v);
```

- You should get



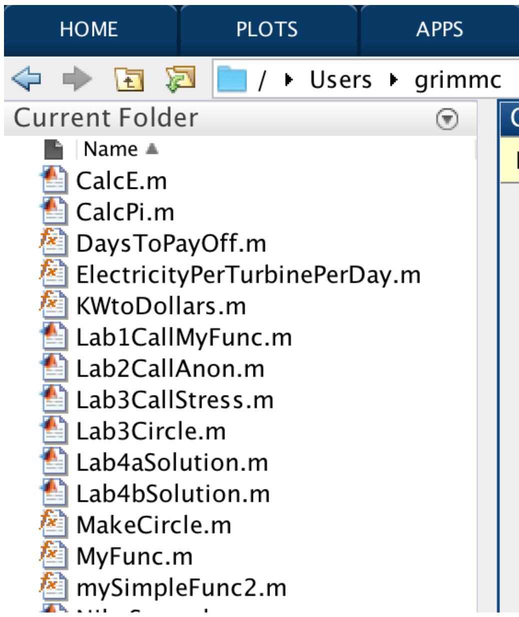
```
>> runSimpleFunc
         0    1.2346    4.9383   11.1111   19.7531   30.8642   44.4444   60.4938   79.0123  100.0000

>> |
```

- If it says



```
Undefined function or variable 'mySimpleFunc'.

Error in runSimpleFunc (line 4)
v = mySimpleFunc( t );

>>
```

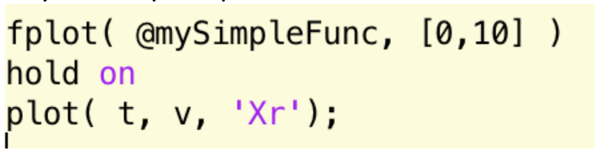
  - two things could be wrong; either you saved mySimpleFunc in the wrong place, or you mis-spelled the filename/function name (check capitalization in particular). To see if it's in the right place, look in the file window on the left of the command window to see if your function is there.



CalcE.m
CalcPi.m
DaysToPayOff.m
ElectricityPerTurbinePerDay.m
KWtoDollars.m
Lab1CallMyFunc.m
Lab2CallAnon.m
Lab3CallStress.m
Lab3Circle.m
Lab4aSolution.m
Lab4bSolution.m
MakeCircle.m
MyFunc.m
mySimpleFunc2.m

- **To Try:** Go to the finder/file explorer and move mySimpleFunc.m somewhere else and try running your script again – what happens? You should get the error above
- **To Try:** Try running the function itself (go to mySimpleFunc and hit run).
    - If your function has arguments then matlab needs to know what they are to run the function…
- **To Try:** Change the .^ to a . in mySimpleFunc and try running the script that calls mySimpleFunc again. What happens?

        - Remember to make sure your function is robust to array inputs – basically, this means putting .* ./ etc everywhere
- **To Try:** Put a break point in mySimpleFunc at the y = x.^2 line
    - Run the calling script
    - Look in the variable window – do t or v or y exist? Make sure you understand why they don't
    - Continue out of the function
    - Do x and y exist? Make sure you understand why they don't
- Add this to your script to plot



```
fplot( @mySimpleFunc, [0,10] )
hold on
plot( t, v, 'Xr');
```

    - The @ symbol tells matlab to look for a function file
    - How many points does **fplot** plot with?
        - Unlike **plot**, it adjusts the number of points depending on how fast the function is changing…

Problem 1

Write a function file that calculates y(x) = sin(2πx)(4x^2 + 3). Plot it using fplot, x goes from -8 to 8. Use your function to print out the value of y for x = -1.25 and x = 1.25.

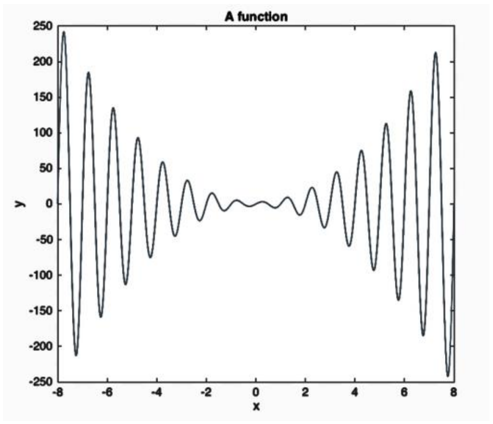
Deliverables:

1. Calling script
2. Function file for the equation
3. Plot
4. Command window output

Step by Step Instructions:

- Follow the guidelines above in Getting Started to make your function script
- Test it first by calling it from the command window with a 1.25 and 0
- Write the script to call the function
- Call fplot – remember to use @fcName and to give it the start and stop values []
    - o fplot will default to a blue line; that's fine
- Make sure you fill out all four boxes below

Self-check:



```
For x = -1.25, y = -X.250000
For x = 1.25, y = X.250000
>>
```

Grading Criteria:

[20 pts] [Comments in script and comments in function]

[20 pts] [Function file]

[10 pts] [Equation correct in function file]

[30 pts] [Function file called correctly in fplot/used fplot]

[20 pts] [Function file called to print out values of x and y]

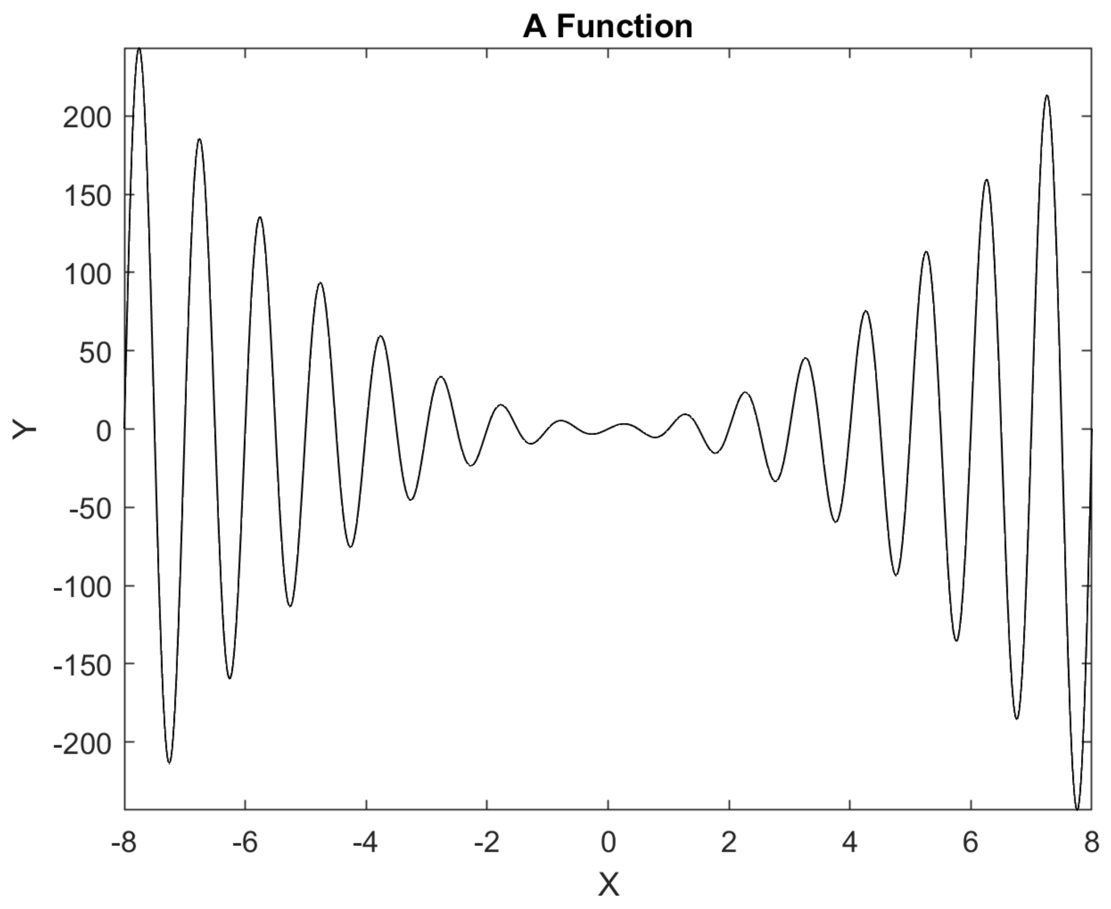
Answer script here:

```
%Andrew Brown Lab 4 Script 1

clc
clear
close all

%Practice calling a function

fplot(@lab4Script1Function,[-8,8],'k')
title('A Function')
xlabel('X')
ylabel('Y')
fprintf('For x = -1.25, y = %0.6f\n',lab4Script1Function(-1.25))
fprintf('For x = 1.25, y = %0.6f\n',lab4Script1Function(1.25))
```

Function scripts here:

```
function [y] = lab4Script1Function(x)
%lab4Script1function
%   practice creating a function with a calculation in it
y=sin(2*pi.*x).*(4.*x.^(2)+3);
end
```

Plot here:



Command window output

```
For x = -1.25, y = -9.250000
For x = 1.25, y = 9.250000
```

Problem 2

Repeat problem 1, but this time use an anonymous function instead of a function file.

Deliverables:

5. Script with anonymous function
6. Plot using fplot
7. Command window output

Step by Step Instructions:

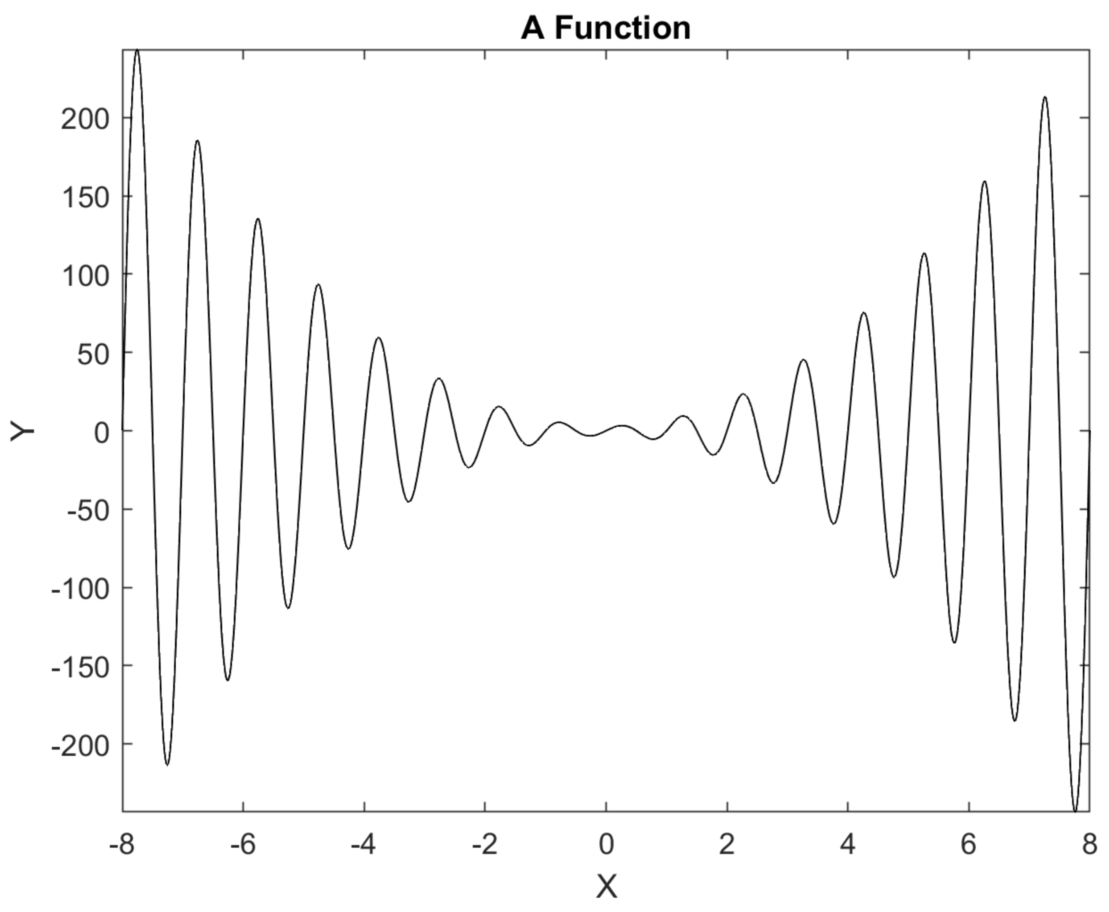
- Copy your calling script from Problem 1
- Change the name of the function to be LabFcAnonymous (or something similar)
- Write the LabFcAnonymous function BEFORE you call it
    - LabFcAnonymous = @(x) …
- Remember to take the @ out of the fplot – you don't need it because LabFcAnonymous is already a reference to a function

Self-check: *Same as above*

Grading Criteria:
[20 pts] [Comments in script and comments in function]
[20 pts] [anonymous function]
[10 pts] [Equation correct in anonymous function]
[30 pts] [Anonymous function called correctly in fplot/used fplot]
[20 pts] [Anonymous function called to print out values of x and y]

Answer script here:

```
%Andrew Brown Lab 4 Script 2

clc
clear
close all

%Practice making an anonymous function

lab4Script2Anonymous=@(x) sin(2*pi.*x).*(4.*x.^(2)+3);

fplot(lab4Script2Anonymous,[-8,8],'k')
title('A Function')
xlabel('X')
ylabel('Y')
fprintf('For x = -1.25, y = %0.6f\n',lab4Script2Anonymous(-1.25))
fprintf('For x = 1.25, y = %0.6f\n',lab4Script2Anonymous(1.25))
```

Plot here:

Command window output

`For x = -1.25, y = -9.250000`
`For x = 1.25, y = 9.250000`

Problem 3

Take the code that counts commas from the script Lab4Prob3CountCommas.m and move it into a function. Alter the script so that it calls your newly created function.
Extra credit: Modify the function to return the number of commas inside ***the function as a separate variable.***

Note: Moving existing code that's working into its own function is something that you'll end up doing alllll the time. This is a step-by-step guided version of doing that. Follow the instructions, even if you think you can just "do" it.

Deliverables:

1. Function file that takes in a string and returns the number of quoted strings and number of commas found (NOT in quoted strings)
2. Modified Lab4Prob3CountCommas.m script that calls your function
3. Command window output

Step by Step Instructions:

- Create a function file using the "New"->script menu
- Edit the function file header as follows:
  - Change the name from `untitled` to something useful like `MyFindCommas`
  - Replace `input_args` with `myStr` – the variable name for the string that is currently used in `Lab4Prob3CountCommas`
    - Best practice is to go into `lab4Prob3CountCommas` and copy `myStr` and then select and replace `input_args` – this makes sure that you spelled it correctly
  - Replace `output_args` with the desired outputs, number of quotes and comma list
    - Again, cut and paste the variable names (`commaLocations` and `nCountQuotes`)
    - Don't forget a comma between the output arguments… you can put the output arguments in any order.
- Edit the description to be something useful
  - Copy the comment from `Lab4Prob3CountCommas`
  - Also a good idea to add one line describing each input and each output variable,
  - eg
    - INPUT: myStr is a string that contains some number of commas and quoted strings
- Cut and paste the actual code from `Lab4Prob3CountCommas` into `MyFindCommas`

- Save both files – do "save as" for the `Lab4Prob3CountCommas` and rename that script to something like `MyLab4Prob3`.m
- Check that MyFindCommas does not have any red bars on the right hand side, indicating an error – if it does, that probably means you got the input argument name wrong OR you didn't cut the entire bit of code.

- Copy the FUNCTION HEADER from MyFindCommas back to MyLab4Prob3 – put it where you just finished cutting out the code, i.e., between the two ------ lines
  - i.e., copy and paste function [stuff] = MyFindCommas(myStr)
- Remove the function keyword and add a semi-colon to the end.
- Run! It should just work. Things that can go wrong:
  - You didn't use the variable names that were there before (you changed the variable names)
  - You swapped the order of the output variables
  - Note: None of this should happen if you followed the cut and paste directions above…
- Extra credit:
  - Add another output variable to the function for counting the number of commas inside the function
    - Set it to zero initially
  - Go to the calling script and add another output variable (or cut and paste the modified output variables from the function file)
    - Alter the print statement to also print the number of commas in the string
  - Check: If you run your script it should print out zero
  - Now go back to your function file and have it count the number of commas inside the string
    - Which if statement do you need to modify?

Self-check:

```
Found 2 quoted strings and 4 commas
Comma locations:     14    25   123   125

>>

Extra credit, add a line:
Found 2 commas in quoted strings
```

Grading Criteria:
[20 pts] [Created function file]
[10 pts] [Put comments in the function file]
[20 pts] [Filled in input/output arguments correctly]
[30 pts] [Called new function file correctly]
[20 pts] [Output still works]
[10 pts EC] Adding another output variable
[10 pts EC] Correctly getting that output variable back and printing it out

[10 pts EC] Correctly altering the function code to count the number of commas in the string

Answer script here:

```
% Starting script for Lab 4: Create a function from code
% Name:Andrew Brown

% Usual clear/clc to clear command window and variables
clear;
clc;

% Open up a text file with some text in it
fid = fopen('DataFileLab4.txt', 'r');
% Read in the text file as one string
myStr = fscanf(fid, '%s'); % %s says read in as string
% Good practice - close file
fclose(fid);


% Replace code with a call to your created function
nCountQuotes = MyFindCommas(myStr);
commaLocations = (MyFindCommas(myStr));

% Print out results
fprintf('Found %0.0f quoted strings and %0.0f commas\n', nCountQuotes,
length(commaLocations) );
fprintf('Comma locations: ');
disp( commaLocations)
```

Function scripts here:

```
% Starting script for Lab 4: Create a function from code
% Name:Andrew Brown

% Usual clear/clc to clear command window and variables
clear;
clc;

% Open up a text file with some text in it
fid = fopen('DataFileLab4.txt', 'r');
% Read in the text file as one string
myStr = fscanf(fid, '%s'); % %s says read in as string
% Good practice - close file
fclose(fid);


% Replace code with a call to your created function
[commaLocations,nCountQuotes,nCommasInQuotes] = MyFindCommas(myStr);

% Print out results
fprintf('Found %0.0f quoted strings and %0.0f commas\n', nCountQuotes,
length(commaLocations) );
fprintf('Comma locations: ');
disp(commaLocations)
fprintf('Found %0.0f commas in quoted strings\n',nCommasInQuotes)
```

Command window output

```
Found 2 quoted strings and 4 commas
Comma locations:     14    25   123   125

Found 2 commas in quoted strings
```

Extra Credit: Problem 1

Print with three values of a – [1 4 -2] – all in the same window. Must use fplot and an anonymous function and a for loop.

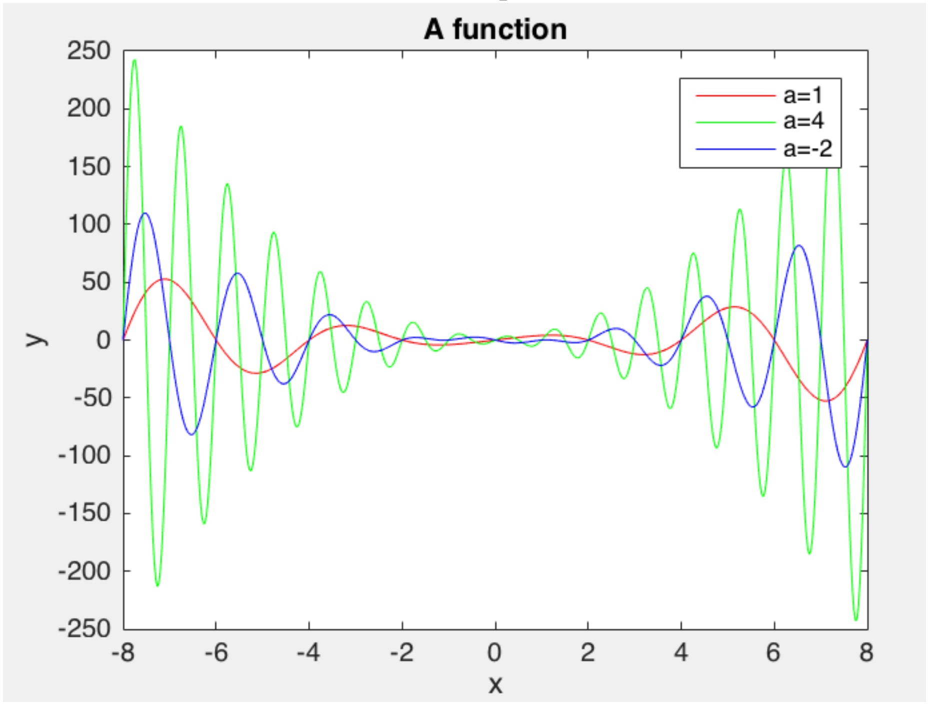
Step by Step Instructions:

- Start with your extra credit script from problem 3
- Declare **a** and use it to wrap a for loop around the plot
  - Make sure the loop iterations depend on the number of elements of a
  - Don't forget hold on
  - Don't forget to change **a** to **a(k)**
- If you want different colors, you'll need:

```
strColors = {'-r', '-g', '-b'};
```

  - fplot(…, strColors{k}) – notice curly brackets

Self-check: *Note: You do not need to plot in color and put a legend in; they are included to make it easier for you to see the answer*



Grading Criteria:
[10 pts] Declare a as an array
[20 pts] For loop, correct number of iterations
[10 pts] For loop depends on number of elements in a
[30 pts] Correctly calling fplot with an anonymous function that uses a
[20 pts] Plot is correct

[+10 pts] Colors set correctly
Answer script here:
`% Copy and paste your script here. Make sure your formatting looks the same as MATLAB, with` **`size 10 font`**`.`
Function scripts here:
`% Copy and paste all functions here. Make sure your formatting looks the same as MATLAB, with` **`size 10 font`**`.`
Plot here:
`Save your plot as a png and paste it here.`

zyBooks Challenge Exercises

Do the challenge activities for the following in Week 4

1. Custom functions
    a. Plus x: A first function
    b. Function definition: Double down.
    c. Function definition: Volume of a pyramid
    d. Function call in expression: Reduced pricing
2. Anonymous functions
    a. Travel speed