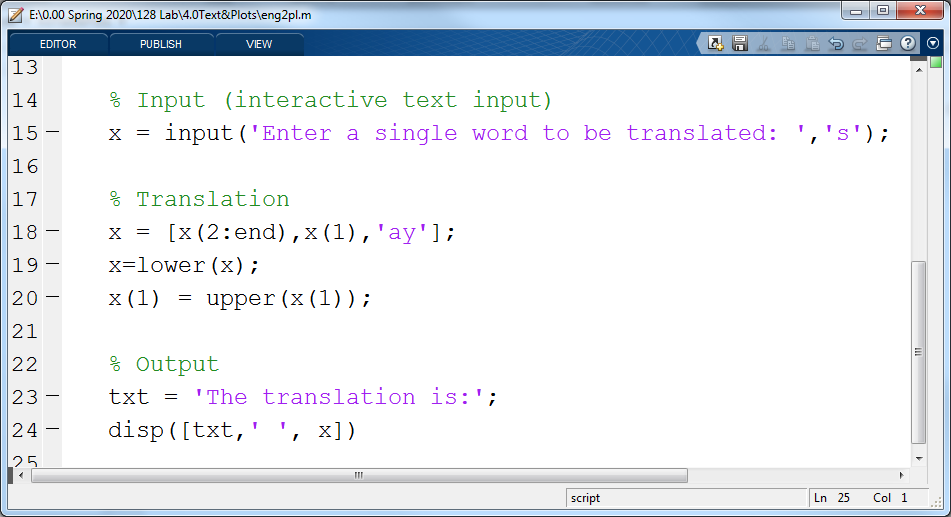
**Text Variables and Files in MATLAB**

1. **Text variables in m-file programs** 
   1. Try out the text program eng2pl.m   
      Download the program eng2pl.m. Run the program from the command window, putting in your first or last name when asked for a single word input. See what the program does. Then look at the code with a fellow student and see how it accomplishes its task. This code includes several examples of working with text variables. Figure 1 shows a copy of the editor window with the main body of this code.

**Figure 1:** The main code for eng2pl.m. notice that there are 12 rows of introductory comments that are not shown. This code shows several examples of using text variables.

* 1. How text is used in eng2pl.m (line numbers are referring to Figure 1).
     1. Creating a text variable using the ***input*** function (line 15): Here the input function asks the user for a word and then stores the response in the variable x. Notice a ‘s’ has been added as a second argument to the function. This tells the input function to expect a character string.
     2. Addressing text (line 20): Notice the first element of the string is addressed on both sides of the equal sign in order to take the first letter, capitalize it using the ***upper*** function and store the result back in the first position in x.
     3. Concatenating text with addressing (line 18): In this case, square brackets are used to combine three pieces of text one after the other. This is called concatenation. Addressing is used so the first piece is everything except the first letter, and the second piece is the first letter. Then, ay in single quotes is added to the end. The quotes are needed here because ay is the text to use, not a variable containing the text.
     4. Assigning text to a variable (line 23): Here the output label is simply put into a variable named txt using the assignment operator.
     5. Using text variables in input to ***disp*** (line 24): Here again text pieces are combined with square brackets. This includes a title, a space to separate parts and the “translated” word that is the output.

**Exercise:** Modify the function CylScript2 to (1) have an additional input command asking for the length units and (2) modify the output display titles to include the units that were input.

**Show your instructor or TA**

1. **Drying Experiment Data Setup**

To illustrate the various ways to read data into MATLAB we will be using some data on the drying of a small coffee sample. The data files are available on the course website.

1. **Preparation:** Copy drying experiment files to your computer, store in a directory you plan to use as your working directory for this unit. These files are available on the Lab web page ([www.etcs.pfw.edu/~moor/128/Lab.html](http://www.etcs.pfw.edu/~moor/128/Lab.html))
2. **Examine files:** Open the Windows Explorer (the file folder on the taskbar) and navigate to these files. Make sure the Explorer is showing the Preview Pane (ask if you do not know how to do this or understand what it is). Use the Preview Pane to take a quick look at each of these text and CSV files to get familiar with their setup. It is also helpful to set your windows explorer to show the file extensions
3. **Interactive Import: The Import Window**

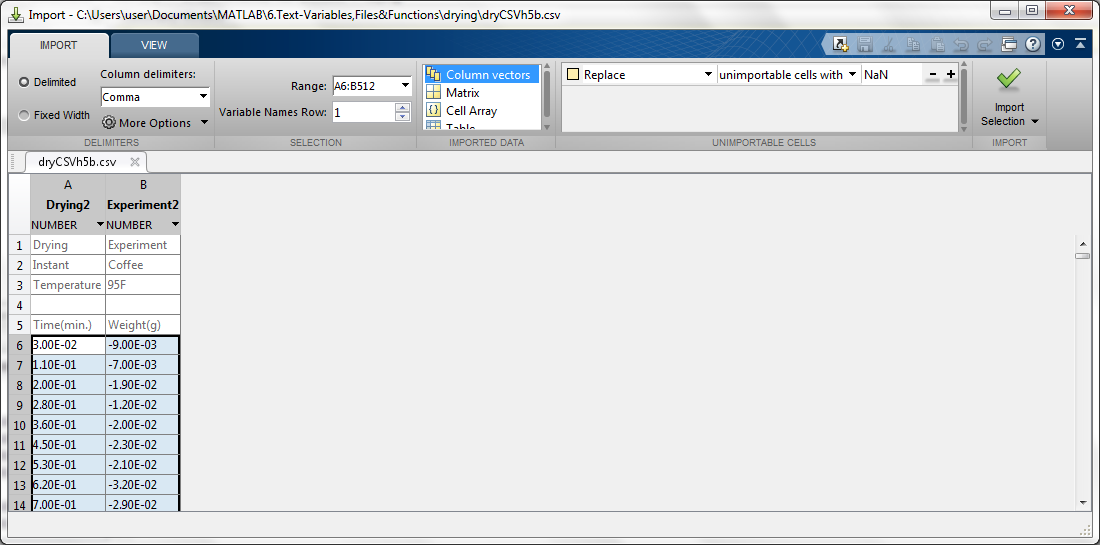
CSV files: Files that end in a .csv are “Comma Separated Values” files. They are usually coded in ASCII (American Standard Code for Information Interchange) with commas separating the values. This is a common form of text file used by Excel. You may have noticed that Windows categories them as Excel files. They are a much more general file type than simply an Excel file.

**Exercise:** These files can be read into the workspace using MATLAB’s ***Import*** window:

In the MATLAB’s ***Current folder*** window double-click on one of the dryCSVh5a.csv file. This opens the Import Window shown in Figure 2. Follow the steps shown in this figure.

**③ Output Type:**  select the type of variable to create in MATLAB.

For this lab, use the “***Column Vectors***” option only!

**Figure 2:** The file Import window. This window appears when the CSV file is double clicked on. It shows a portion of the data and gives setup options for the inport.

**④ Import Selection:**

Once other steps are complete, Click here.

**② Variable Names Row:** Set this to the row of the column names (it might be there automatically)

**① Column delimiters:**

Click on drop down arrow to see other options. Set to Comma

Note: Text files (ending in .txt, .dta, .dat): Ordinary text files do not open the Import window automatically (double click and they are opened in an editor window but not imported in to the workspace). They can be interactively imported by:

1. highlight a .txt file (e.g., dryh4.txt) and then click on the ***Import Data*** icon on the *HOME* tab or
2. dragging and dropping the file from the *Current Folder* window to the ***Workspace*** window

Either of these approaches will open the Import window. Set up the import as before to read in the file.

**Important Limitation:** These are very helpful (and easy) import options but unfortunately cannot be used in an m-file. For that a function is needed that be used inside a program. One of the easiest ways to do this is with MATLAB’s ***importdata*** function. For the rest of the lab we will be using this function.

1. **Importing files in the command window or in an m-file (using *importdata*) & Plotting:**

The *importdata* function will easily load most text/csv files. Note, for this exercise it is expected you are working from the command window while learning to use this function.

* 1. CSV/Text files with header rows: The file dryCSVh5a.csv has five header rows (three that describe the experiment, a blank line and then a row of column labels or headers). Import this file using the following command:   
      >> stuff = importdata(‘dryCSVh5a.csv’)  
     Leave the semicolon off so you can see the overall result. Notice it creates a new file arrangement with three parts:   
      1) data = a 507 x 2 array of the numbers  
      2) textdata = a 5 x 2 cell array if the header data   
      3) colheaders = a 1 x 2 cell array of the last line in header

This type of arrangement is called a “structure” in MATLAB

* 1. MATLAB structures are a way to combine different types of data in a single workspace item. The sub-elements of a structure are called fields. Each field contains only a single data class (i.e., one data type). The structure that was created by the previous import command has three fields. A field can be accessed using the format *StructureName.FieldName*. Try the following example:

**Complete the following commands, answer the questions and show your instructor or assistant:**

at the command line type the following commands (using the variable name you assigned in b. above).   
 >> a = stuff.data;  
 >>b = stuff.data(1:5, 1);

What did each of these executables return (look in your workspace to see information about the variables created)?

Below write a MATLAB line that would create a vector called “time” that contains the first column of the data in this structure.

>>

* 1. Plotting the imported data: To see what has been imported creating a plot is helpful.   
     To prepare a plot of this data complete the following steps:
     1. Copy the two columns from the structure to new (and simpler) variables   
         >> t = stuff.data(:, 1);

>> m = stuff.data(:, 2);

* + 1. Plot these two variables using the plot function

>> plot(t, m,**'.'**)

* + 1. Add axis labels:  
        >> xlabel 'Time(s)'  
        >> ylabel 'Mass Loss (g)'

**Complete and Show your instructor:**

Use the ***importdata*** function to import one of the drying data files.

Plot the resulting weight loss versus time data.

Show the commands and plot to your instructor or assistant.

Notes on the plot function:

* **Data:** In all plot commands the x & y vectors must be the same length
* **Symbols:** The **‘.’** in the above plot command causes each point to be plotted as a period (usually only used for large data sets like this one). The character in the quotes can be changed to produce different symbols. For example:   
   ‘o’ produces a circle as the symbol and   
   ‘p’ produces a five-pointed star.   
  More symbols and options can be found by typing >> help plot  
  If no symbol is specified the plot function will connect the points with lines.
* **Multiple curves:** >> plot(t1,y1,’p’, t2,y2)

Plots y1 vs t1 with stars and y2 vs t2 with connected points on one graph,   
More vector pairs can be added.

* **Adding a Legend:** >> legend( ‘title1’, ‘title2’)

Adds legend to identify multiple series, include as many titles as there are separate plotting pairs in the plot command.

1. **Variations on the theme: Other cases and options when using importdata**
2. Simple Data Files: When the data file is only an array of numbers (i.e., no header rows), the data will be imported into a simple matrix variable instead of a structure.
3. Using a stored file name: The file name for an import can also be stored in a variable. For example:

>> fname = 'dryCSVh5a.csv';

>> NewStuff.data =importdata(fname)

Notice that the file name needs quotes when it is stored in the variable fname. However, fname does not need quotes when used in the importdata function because it is a variable containing the text data, not the literal text data itself. This approach is particularly helpful in various programs.

1. Loading a poorly formatted file: Try importing dryCSV5b.csv with the importdata function. This import has some problems (the actual data is not imported!). To get this one to load MATLAB needs some specific help on its structure. The importdata functions has several helpful options. Typing “help importdata” in the command window will give some of these options. In the help file are the following lines:

importdata(FILENAME, DELIM, NHEADERLINES) loads data from ASCII file

FILENAME is the same as before,   
DELIM is to specify the delimiter and   
NHEADERLINES is the number of rows of header information

For this problematic data set try the following:   
 >> stuff5 = importdata('dryCSVh5b.csv', ',' , 5)  
This should take care of the problem because we have specified the delimiter (a coma is specified in this case by the ‘,’) and the number of text header lines before the actual data begins.

1. Cell Arrays (Optional Aside): Notice that the two fields in the usual structure that contain text data (i.e., textdata & colheaders) both are listed as type cell and have brackets {} around their dimensions or content. These two are a special type of array called a cell array. To address cell data, the indexes should be in curly braces, { }, to result in a normal text variable.   
    e.g. try >> col1name = stuff2.colheaders{1}
2. **Recording and Graphing Accelerometer Data** (problem to turn in next week)

*Overview****:*** In this part the goal is to take accelerometer data with a smart device (you may work with another student to take the data), download the data to each team members computer. Individually import the data into MATLAB and plot the result. In the end we would like to develop a function that can import the accelerometer data into MATLAB and plot it automatically.

**MATLAB work must be done individually.**

***Specific Steps:***

1. Team up with another student and at least one smart device (tablet or phone).
2. If you have not already done so, download an accelerometer app to the device. Direct links to accelerometer apps can be found by clicking on the “Accelerometer Apps” link on the web site or go directly to (<http://www.etcs.pfw.edu/~moor/128/Lab/ResourceFiles/accel/index.html>).   
   The apps we have experience with are:   
   **For Android devices:** Physics Toolbox Accelerometer or Physics Toolbox Suite   
    ( For the latter, use the g-force meter only, not the linear acceleration option)

**For Apple devices:** Acceleration (by IU!) or Physics Toolbox Accelerometer

1. Take a few minutes (< 10 min.) to experiment with the app to get familiar with how it responds. Complete a couple of short experiments (10 – 15 seconds). First hold the phone so it is flat and just walk in a straight line. Then try something else simple (E.g., shake the device, rotate the device, run with it, go down stairs, …).   
   Notice: the at-rest readings, the units, and the impact of various activities.
2. Select a specific activity to record the data. Redo the activity carefully with the apps recording function running. Minimize the recording of before and after data (although you likely want an at-rest” baseline.)
3. Transfer the recorded data to the lab computer make sure everyone on the team has a copy.

You should be able to read this data in using the importdata function in the command window.

**To Turn in Next Week:** Write a script to read in and plot data from an accelerometer experiment. Figure 3 shows the first page of a Program Development Sheet (see rubric online). This sheet indicates that the name of the file should be inputted interactively using the ***input*** function; This name should then be used to read in the file using the ***importdata*** function. Notice the difference in spelling these two functions (input with an ‘n’ vs. importdata with an ‘m’).   
  
The resulting output should be a single figure that shows the x, y and z values in m/s2 vs. time (see notes on plot function). Pay close attention to the validation requirement noted at the end of this sheet. Test this function with two experiments: 1) the walking file (from step 3 in Section II), and 2) with your selected activity (from step 4 in Section II).

**Figure 3:** A program Development Worksheet for the accelerometer problem.

**Program Development Worksheet:**

Problem ID \_Acclerometer Script Programmer \_\_\_ \_\_ \_\_\_\_

Set Up/ Planning Type of Program: ✓Script  Function

1. Problem Statement:

Develop a script that can import accelerometer data from the CSV file and create a figure with a properly labeled and formatted graph of the acceleration in the x, y and z directions. Accelerations must be in m/s2.

1. Inputs: (full name, variable to be used, units)

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Description | Units or Values | Input Source\* |
| filename | The filename of the .csv file containing the accelerometer data. | A text string | Interactive input |
| AccData | The acceleration data (from a .csv file read into a MATLAB structure). | Time in seconds  Acceleration in g | File |

\* Possible sources: command line, file, interactive input

1. Output: (full name, variable to be used, units)

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Description | Units or Values | Output type\* |
| x, y, z vs. t | Plot all three on one figure | Time in seconds  Acceleration in m/s2 | Figure Window |

\* Possible types: command line, file, display, figure window

1. Solution Steps (order of these two parts may be varied):
2. Perform calculation on test case(s) (2) Identify the steps/equations to be used in code
3. Request the name of an input file from the user and store in a variable.
4. Read in file using filename variable from step 1.
5. Create t, x, y, & z vectors from the structure read in in step 2
6. Convert x, y an z to m/s2
7. Create a plot of x, y, and z vs. t as a single plot with three y-variables or   
   Create a plot for each y-variable vs t (i.e., x vs. t, y vs. t and z vs. t) and organize these ad three subplots in one figure window

Either approach should include complete labeling matching required figure standards

**Validation Requirement:** 1. Create two different cell phone sensor data files

2. Run script on each file (without editing the script in between).

3. For one of the output plots, pick 3 or 4 test points in the plot and compare to equivalent rows from the original file to make sure your plot is correct. Include data curser labels on the plot to show the chosen points and their coordinates.