Intro to Matlab: Arithmetic and Figures

Aim: In this lab, we will review basic arithmetic on scalars, vectors and matrices in Matlab. We will also create and export a simple plot.

Learning outcomes: At the end of this lab, you will be able to:

1. Load `csv`-files containing numerical values only.
2. Create simple *scalars*, *vectors* and *matrices.*
3. Use Matlab functions including: `load`, `whos`, `mean`, `median`, `sin`
4. Use Matlab arithmetic operators including: +, `-`, `*`, `/`.
5. Explore the use of special characters: the semicolon (`;`), colon (`:`), and transpose (`'`).
6. Create a Figure with a simple line plot and export it in `*.png` format.

Data: Use the `data_vector.csv` and `data_matrix.csv` data files, available on Blackboard.

Directions: Using both Matlab, answer the questions below.

Check your answers: This is a formative assessment rather than summative (i.e. the results here do not contribute to your mark in the course). Answers will be available for you to check on Blackboard.

---

1. **Loading a *.csv file into Matlab**

   (a) Load the data

   ```
   >> data_vector = csvread('data_vector.csv');
   ```

   (b) **Where did the data go?**
   Method 1: the whos command. Do a `whos` to see what variables were created. In your command window, type

   ```
   >> whos
   ```

   Method 2: the Workspace window. Check the variables that you have loaded and active in your Workspace window. It also tells you the size of the variables, and their class (numeric, character).

   (c) **Size of the variable.** The size of the variable refers to how many entries it has. What is the `size` of the variable called "labA_vector"? (Use the Matlab command `help` to see how to use the command `size` by typing at the command prompt

   ```
   >> help size
   ```

   (c) ____________________

   (d) **Dimensionality of the variable.** In Matlab, we call a variable a 2-dimensional matrix if it has 2 non-singleton *dimensions.* A vector which is `100x1` is still 1-dimensional, while a matrix that is `2x2` is 2-dimensional. Is this new variable a matrix or a vector?

   ○ a matrix ○ a vector

   (e) **Extract data from the vector.** What is the value in the 4th element of the vector? You can answer this by looking in your Workspace, but you should learn to access the information using indices.

   ```
   >> data_vector(4)
   ```

   (e) ____________________

2. **Basic calculations.** Matlab has standard commands to calculate the mean or median of a dataset.

(a) **Mean.** Calculate the mean of the data using the `mean` command. This command takes one input: the name of the variable that you'd like to average.

```
>> mean(data_vector)
```

(a) ____________________

(b) Now instead of the command above, try the command

```
>> mean data_vector
```

Is the answer the same? ∴ **Brackets (parentheses) are important!**

(b) ____________________

(c) **Addition.** Add two elements together using the + operator:

```
>> data_vector(1) + data_vector(2)
```

Note how the cells or "elements" of the column are accessed in Matlab, using brackets and the row number.

(c) ____________________

(d) **Median.** Calculate the median using the `median` command.

```
>> median(data_vector)
```

(d) ____________________

3. **Working with a matrix in Matlab.**

(a) Load the file `data_matrix.csv`. What is the `size` of the variable loaded? (Recall: the `whos` command.)

(a) ____________________

(b) **Vector vs Matrix.** In Matlab, we call a variable a 2-dimensional matrix if it has 2 non-singleton dimensions. Is this new variable a matrix or a vector?

○ a matrix ○ a vector

(c) **Mean of a matrix.** Calculate the mean of this matrix,

```
>> mean(data_matrix)
```

(c) ____________________

(d) What was averaged here?

____

____

(e) The first dimension in Matlab (i.e. the `R` in the `RxC` of the size) is the number of rows and the second dimension is the number of columns. Is this the same as in Excel? (Load the *.csv file into Excel to check.)

○ the same ○ the opposite

4. **Arithmetic in Matlab.** Matlab can carry out all basic kinds of arithmetic. Most of the operators are the same as you'd expect.

(a) `>> 3+5*2`

(a) ____________________

(b) Recall *order of operations* for arithmetic. It applies in Matlab as well.

```
>> (3+5)*2
```

(b) ______________________

(c) `>> 3/5`

(c) ______________________

(d) Raise numbers to powers (i.e. squaring numbers)

```
>> 5^2
```

(d) ______________________

(e) `>> pi`

(e) ______________________

(f) `>> sin(pi)`
Note: that due to machine or numeric precision, this may not be exact.

(f) ______________________

(g) `>> log10(10)`

(g) ______________________

5. **Create new variables in Matlab.** One of the powerful things about programming languages is that you can "name" your data. Rather than carrying out a stream of calculations in the command window, you can work with different numbers at the same time, saving partial answers, and combining them later.

(a) **Scalar variables.** Create a *scalar* variable. Then execute a `>> whos a` to see what you've done. Examine `a` in your Workspace.

```
>> a=3
```

(b) **Vector variables.** Create a vector

```
>> x=[1 2 3]
```

(c) Repeat this using `y` instead, but "suppress the output" by using a semi-colon (`;`)

```
>> y=[1 2 3];
```

Note: The only difference between (b) and (c) is that in (c), the output is not printed to the screen; the variable is still created. It can be useful to suppress large output.

(d) **Manipulating vector orientation.** Change the orientation of `y` by *transposing* the vector using an apostrophe (`'`)

```
>> z = y'
```

Use the `whos` command to see how `z` and `y` differ. *Recall:* that the transpose of a matrix means to switch the rows and columns.

(e) **Using functions on variables.** Calculate the mean of your variable `y`. Recall the command `mean` from question 1(a).

(e) ______________________

(f) **Repeating calculations for each element of a vector.** Some calculations can be carried out on individual elements of a vector rather than on the whole thing. Calculate the square of the individual elements in the vector `x`. In Matlab, this is accomplished by using the 'dot' operator, as

```
>> x.^2
```

(f) ____________________

(g) **Create a matrix.** Create a 2-dimensional matrix `w`. How many rows and columns does `w` have?

```
>> w = [[1 2 3]; [4 5 6]]
```

(g) ____________________

(h) **Create a vector using the colon.** For regular vectors, like the vector containing the numbers 1 through 10, you can use a special operator to write it quickly. Create a vector containing the numbers from 1 to 10 using a colon (`:`). How long is `c`?

```
>> c = 1:10
```

(h) ____________________

(i) In the previous example, the numbers were spaced by 1, which is the default. You can also space them by an arbitrary value. Create a vector containing the numbers from 1 to 10 but spaced by 0.5. How long is the 1-dimensional vector `d`?

```
>> d = 1:.5:10
```

(i) ____________________

(j) Create a time vector called `time` that is 3 years long, on a daily time resolution.

```
>> time = 0:1:365*3;
```

How long is the vector?

(j) ____________________

Using a function

In the above sections, we loaded data, and tried out some basic arithmetic. You also used a *function* in 4(f) and 4(g). In maths, a function describes a relationship between two or more variables. For example, if we have that 'y is a function of x',

$$y = f(x) = 3x + 7 \tag{1}$$

where the function f operates on the input x, as described by the $ax + b$ expression. In Matlab, a function is a command that can operate on inputs. Here, we will explore the `sin` function further.

6. **Use trigonometric functions and create a figure.**
From maths, recall that: $sin(\pi) = 0$ while $\cos(\pi) = 1$.

(a) You can check this in Matlab as

```
>> sin(pi)
>> cos(pi)
```

Here we evaluated the function `sin` given a *scalar* (single number) input. In this case, the input was `pi`.

(b) You can also calculate the sign of a *vector* input, e.g. a range of times. Create a sinusoid with an annual harmonic (i.e. its period $T = 365$ days) using the `sin` function and the vector `time` created above in 5(j).

```
>> y = sin(2*pi/365 * time);
```

(c) **Create Figure 1** by using the command `figure`. This command takes one input, a numeric, telling Matlab which figure window to create (if it doesn't exist) or activate (if it's already open).

```
>> figure(1)
```

Plot the sinusoid using the `plot` command. This command takes two inputs, the time vector and the data vector.

```
>> plot(time, y)
```

Type a `>> help plot` at the command line to see what options you can use for the inputs. *Note: Your line plot should have 3 peaks and 3 troughs if the time vector was correctly created.* How many peaks does your line plot have?

(c) ____________________

(d) Create an x-axis which is marked in months using

```
>> datetick('x','mmm');
```

For `datetick` we gave it two inputs or arguments. The first one says to apply the function on the x-axis. The second is a formatting string. Try a `>>help datetick` to see other options for formatting. If you want an annual tick on the x-axis, what command should you use?

(d) ____________________

(e) In what month does y peak?

(e) ____________________

(f) Create a new sinusoid using the `cos` function. Add this to the figure in a new colour. (See `>> help plot` to see how to change the colour of the line.)
When does this sinusoid peak?

(f) ____________________

(g) Changing the amplitude. Now plot $y = 3\cos(2\pi t/T)$. Now what is the range of y?

(g) ____________________

(h) **Export the figure.** In the Figure window, use the `File>Save as` choice to save the figure into a file on your computer. Change the default "File format" to `*.png`. Save the figure as `Fig1.png`.