Matlab Exercises
Part 1

version 7.1, EJP, 2019

1. Start matlab.

2. Enter the following

```
1 + 2
x = 1 + 2
x = 1 + 2;
y = x^2 + 2*x + 8
```

3. Enter the following

```
format longE
pi
```

You can use the arrow keys and the delete key to recall and edit previous commands. Press the up arrow key twice to recall the format command and delete the "**e**" and press enter. Then display **pi** again. Repeat with the following formats.

```
format shortE
format short
```

4. Enter the following

```
a = pi/6
sin(a)^2 + cos(a)^2
exp(2*log(3) + 3*log(2))
```

5. Enter the following

```
a = [1 2 3]
b = [4 5 6]
Z = a + i*b
```

i here is the square root of -1. You can use **j** instead if that is what you are use to.

6. Display the real and imaginary parts of **Z** and the conjugate of **Z**.

7. Display the angle and magnitude and of **Z**.

8. Try the following.

```
Z'
Z.'
```

9. Enter the following matrix

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

10. Try the following and ensure you can follow what is happening.

```
B = A + 4
A + B
A - B
A * B
A^2
A'
```

11. Solve the simultaneous equation on page 15 of the notes. **A** should already be present from exercise 10.

```
b = [ 5 ; 11 ]
x = A \ b
```

Now check that **x** is the correct solution.

```
A * x
```

12. It is just as easy to solve a hundred simultaneous equations in a hundred variables. First create a 100 by 100 matrix of random numbers.

```
A1 = rand(100);
```

If you forget to put in the semicolon, 10,000 numbers will be printed out.

Next create a column vector with 100 numbers

```
b = (1:100)'
```

Now solve

```
x = A1 \ b
```

Check that the solution is correct.

```
A1 * x
```

Part 2

1. You should have the two matrices

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \qquad B = A + 5$$

from exercise 10 in part 1.
If not, then enter them again.

2. Now try the following array operations.

```
A .* B
A ./ B
A .^ B
sin(A * pi/6)
D = A.^2
sqrt(D)
```

Clear the workspace of all variables.

```
clear
```

3. Plot the polynomial $2x^3 - x$

```
x = linspace(-1,1,100);
y = 2*x.^3 -x;
```

What happens if you don't include the dot ?

```
plot(x,y)
```

Don't close the figure containing the plot.

4. Plot the polynomial $2x^3 - x$ using the function polyval. First find out how to use polyval using the help.

```
doc  polyval

p = [4 0 -3 0]
y1 = polyval(p,x);

hold on
plot(x,y1,'g')
```

5. There are many functions that handle polynomials. Look them up in the help. Enter **doc polyval** again, then click on **Functions** on the blue banner at the top of the window.

What does the function **roots** do?

6. Plot the roots of the polynomial onto the graph.

```
r = roots(p)
ry = zeros(3,1)
```

The plot should still be held from exercise 4.

```
plot(r,ry,'rx')
```

Clear the figure

```
clf
```

7. Enter the following.

```
a = 2: 0.5 : 4

a(2)
a([2 4])
a(2:4)
a(2:end)
```

8. Enter the following

```
w = (1:5)' * (5:10)
```

This produce a 5 by 6 matrix that we can use in the next exercise.

Set the third element on the second row of **w** to **100**.

9. Enter the following

```
w(2:4,2:4)
w(2,:)
w(:,5)
w([1 5],[2,4])
w(:)
w(:) = 1:30
```

10. By now you should have a nice collection of variables. Try

```
who
whos
```

If you cannot see the workspace window, click on the **HOME** tab and then click on **Layout** in the **ENVIROMENT** section and select **Three Column**.

Enter the following in the command window.

```
save
clear
```

All variables should have been saved to **matlab.mat** . If you can't see this in the **Current Folder** window, right click in the window and select refresh.

The workspace window should be empty. Double click on **matlab.mat** to restore all your variables.

11. Produce a script called **mygraph**.

```
edit  mygraph
```

In **mygraph** enter

```
x = linspace(-2*pi,2*pi,100);
y = sin(x);
plot(x,y)
grid
```

Save by clicking on the icon and run by entering

```
mygraph
```

in the command window.

12. Add the following at the end of the script created above.

```
hold on
y1 = mysin(x);
plot(x,y1,'r:')
axis( [-2*pi,2*pi,-2,2] )
```

Click on the "Save and Run" icon

13. You should have a file called **mysqrt.m**. Edit this file

```
edit mysqrt


function x=mysqrt(A)

% My square root function.
%
%   x = mysqrt(A)
%
% x is the square root of A.


x=1;      %First guess
err=1;    %Set error to something to get started

while(err>0.0001)
   x = myfunction(A,x);    % call to local function below
   err = abs(A -x.^2);     % calculate the error
end

function  y = myfunction(A,x)
% Local function to calculate the
% Newton Raphson Iteration equation

  y = (x.^2+A)./(2*x);
```

Test the function by enter the following into the command window.

```
mysqrt(9)
mysqrt(2)
help  mysqrt
```

14. You are now going to use the debugger on the function above.

In addition to the editor, you need to be able to see the **Workspace** window.

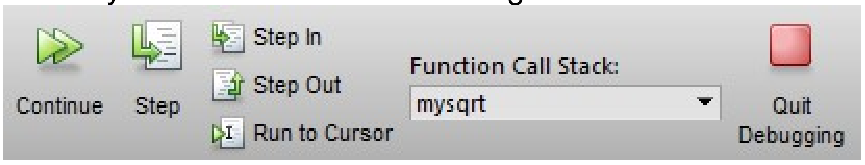
In the editor, find the line containing **x = 1;** Between the line number and the line, you will find a "-" sign on its own vertical bar. Clicking on this sign turns it into a red dot. This is a break point. A break point is where the program will stop so that you can debug the program. You can toggle the break point on and off by clicking on it.

With a break point set on the line noted above, run the program.

mysqrt(2)

The program will stop on the line with the breakpoint. A green arrow indicates the next line to run. Note that the "Workspace" window is showing variables available in the program. You can use the variable editor or the command line to change the value of the variables.

On the icon bar you will now find the following icons.



If you put the cursor over an icon and leave it for a while, a description of what the icon does is displayed.

The **Step** icon steps through the program one line at a time. Step through at least one loop of the while loop. Notice the variables changing in the "Workspace" window.

The **Step In** icon is very like step, except when the line contains a call to a function. **Step** will treat the function as one line of code, while **Step In** will step into the function and step through the function. Try this out.

Step Out reverse a **Step In**. If you are inside a function, a **Step Out** will execute the rest of the function, exit the function and then stop again. Step into the local function and try a **Step Out**. If you hit **Step Out** in the top function, the program will run to completion.

The **Continue** icon restarts the program from the current point in the program. It will run to the next breakpoint or completion. Try this now.

Close the editor.

15. Enter the following

```
degrees = 0:6:360;
rad = degrees * pi /180;
plot(sin(rad))
```

Close the graph and repeat the first two commands by double clicking on them in the **Command History** window. If you cannot see this window, from the **HOME** tab select **Layout► Command History ►Docked**

16. Drag the plot command from the **Command History** into the command window and change it to :-

```
plot(degrees,sin(rad))
```

See how the plot changes.

17. Hold down control and select each of the following commands in the **Command History** Window,

```
degrees = 0:6:360;
rad = degrees * pi /180;
plot(degrees,sin(rad))
```

then press the right mouse button and select **Create Script**.
Save the file as **graph1.m**

18. Annotate the graph by adding the following lines to the script.

```
axis([0 360 -1 1])
title('Sine Wave')
xlabel('degrees')
ylabel('value')
grid
```

Observe the resulting graph.

19. Change the title and the plot command so that a cosine wave is plotted in green. Save as **graph2**.

20. Change the title and the plot command so that tan is plotted in red. Change the y limits to -10 to 10. Save as **graph3**.

21. Now try a subplot command. In the command window enter.

```
subplot(3,1,1)
graph1
subplot(3,1,2)
graph2
subplot(3,1,3)
graph3
```

Don't forget that you can use the cursor keys to recall previous commands.

22. Clear the figure and enter the following.

```
clf
graph1
hold on
graph2
hold off
figure
graph3
```

Now delete the extra figures.

```
delete(2)
```

And clear the workspace

```
clear
```

23. You should have a spreadsheet call **XLfile.xls** in the current directory.

Double click on the file **XLfile.xls**

For **Output Type**, select **Column vectors**.

Click on the green tick to import the data.

Close the import window.

If you look at the Workspace window, you will see that the data from the spread sheet has been imported into MATLAB.

24. Hold down the control key and select X and Y in the workspace window.

On the MATLAB icon bar, select the **PLOTS** tab.

Select **plot**.

25. Under **View** on the figure icon bar, select **Property Editor**.

Click on the line of the graph.

Try different line styles, thickness and colour.

Try different markers. Change the size of the markers and try different fill and line colours.

Click on the white background of the graph.

Add a title "My Graph" and a grid in the x and y directions.

Set the XLabel to "Time" and set the XLimits to 0 to 2*pi.

Set the YLabel to "Amplitude".

From the file Menu, on the banner of the figure, select
Generate Code.

A function containing the code is opened in the MATLAB editor.
Save the file to **createfigure.m**

Close the editor and the figure.

26. Run the M file you have just created.

createfigure(X,Y)