In this practical we will revert to using Matlab, which has an extensive set of in-built functions for manipulating matrices. If your Matlab has become a little rusty, it may be helpful to have your practical notes from the Michaelmas term to hand (they are available via Camtools in case you have not brought them with you).

1. Entering matrices and vectors into Matlab

Given the matrix $A$

$$A = \begin{pmatrix} 7 & 3 & 5 \\ 6 & 12 & 1 \\ 9 & 3 & 0 \end{pmatrix}.$$ 

and the vector $x$

$$x = \begin{pmatrix} 1 \\ -2 \\ 5 \end{pmatrix},$$

calculate the vector $Ax$ by hand and write down your answer in the space below.

$$Ax = \begin{pmatrix} \text{answer} \end{pmatrix}.$$ 

Now suppose we wanted to check our answer using Matlab. The first thing we would need to know is how matrices can be entered into Matlab. There are two main options:

(i) Entering a matrix via the Command Window or a script file

Method 1: Simply type in the matrix in the same layout as you would see it written down:

```matlab
>> M1 = [7 3 5; 6 12 1; 9 3 0]
```

*Hint:* Don’t forget to use SQUARE BRACKETS around your matrix of values.

Method 2: Type in the matrix using semicolons (;) to separate the rows of the matrix

```matlab
>> M2 = [7 3 5; 6 12 1; 9 3 0]
```
(ii) Entering a matrix via the Workspace

The Workspace is by default in the top right hand side of your Matlab console.

To enter a new variable via the workspace click on the new variable icon or select the Workspace window and press CTRL-N on your keyboard. This will create an empty variable in the Workspace:

Double click on the grid icon – this should open the Variable Editor.

You can now enter your matrix into the variable editor just as you would enter the data into a spreadsheet. Notice as soon as you add a new row Matlab automatically fills in the rest of the row with zeroes to make the row the same length as the first row of data. Enter the values given in Matrix A at the start of this section and close the Variable Editor. To check the content of the variable M3 type M3 into the Command Window. Vectors can be entered into the Variable Editor in exactly the same way.

A note on entering vectors into Matlab

We can only multiply an $m \times n$ matrix and a $p \times 1$ vector if $n = p$. Matlab does not magically know which way round you intend a vector to be (ie. row or column). There are two options for telling Matlab that you want your vector to be $p$ rows by 1 column rather than 1 column by $p$ rows.

Method 1: Type the vector in the layout as you would see it written down - you can also use the Workspace to do this.

\[
\text{x} = \begin{bmatrix} 1 \\ -2 \\ 5 \end{bmatrix}
\]

Method 2: Type the vector as a row vector and then transpose it using the ‘’ symbol or using the built-in function transpose. Type the following into the Command Window and examine the output after each line of code:

\[
\text{xv2} = \begin{bmatrix} 1 & -2 & 5 \end{bmatrix} \quad \% \text{ a 1 by 3 vector}
\text{xv3} = \begin{bmatrix} 1 & -2 & 5 \end{bmatrix}' \quad \% \text{ a 3 by 1 vector}
\text{xv4} = \text{xv2}' \quad \% \text{ transposes xv2 to a 3 by 1 vector}
\text{xv5} = \text{transpose(xv2)} \quad \% \text{ ditto}
\text{size(xv2)} \quad \% \text{ outputs number of rows and columns in xv2}
\text{size(xv5)} \quad \% \text{ same for xv5}
\]
2. Matrix operations in Matlab

In the Michaelmas term Matlab practical sessions you were taught to use the \( \cdot \times \) operator to perform multiplication. This method of multiplication is called **element-by-element**. To perform standard matrix multiplication we drop the ‘dot’ and simply use \( \times \) without a dot. Adding and subtracting still use the + and – sign as they are always **element-by-element** operations.

### Task 1

(a) Enter the following matrices and vectors into a new script file in Matlab:

\[
A = \begin{pmatrix} 7 & 3 & 5 \\ 6 & 12 & 1 \\ 9 & 3 & 0 \end{pmatrix} \quad B = \begin{pmatrix} -2 & 3 & 2 \\ 1 & 12 & 4 \\ 8 & -1 & 3 \end{pmatrix} \quad C = \begin{pmatrix} 12 & 7 & 3 \\ 2 & 9 & 1 \end{pmatrix} \quad D = \begin{pmatrix} 1 & 3 & 2 \\ 0 & 2 & 1 \\ 1 & 2 & 3 \end{pmatrix}
\]

\[
x = \begin{pmatrix} 1 \\ -2 \\ 5 \end{pmatrix} \quad y = \begin{pmatrix} 2 \\ 0 \\ -2 \end{pmatrix}
\]

(b) Check you understand the output produced by the following commands

\[
A \times D \\
A \cdot \times D \\
C \cdot x \\
C \cdot \times x
\]

(c) Find the following matrix quantities by hand (or explain why they don’t exist)

\[
A + B, AB, AC, Bx, By, Cx, xy
\]

(d) Check your answers to part (iii) using Matlab.

### Task 2

(a) By hand, find the determinant of the matrices A and B.

(b) Check your answers in Matlab using the built-in function \texttt{det}.

(c) What happens if you use the \texttt{det} function on the matrix C? Why?

(d) Use Matlab to verify \( \det(AB) = \det(BA) = \det(A)\det(B) \) for the particular matrices A and B given above in Task 1.
4. Eigenvalues and Eigenvectors

The **Eigenvalues** of a (square) matrix can be found using the built-in function `eig`. To test this function, type the following into the Command Window:

```matlab
>> M = [2 7; -1 -6]
>> lambda = eig(M)
```

The **Eigenvectors** can be obtained using the same built-in function. However, we now need to tell Matlab to output two pieces of information. To do this we assign the answer to a vector of two variables. Type the following into the Command Window:

```matlab
>> [v, lambda] = eig(M)
>> v
>> lambda
```

You should get

\[
\lambda = \begin{pmatrix} 1 & 0 \\ 0 & -5 \end{pmatrix} \quad \text{and} \quad v = \begin{pmatrix} 0.9899 & -0.7071 \\ -0.1414 & 0.7071 \end{pmatrix}
\]

The Eigenvalues are the values in the diagonal of matrix \( \lambda \). And the corresponding Eigenvectors are the columns of the matrix \( v \). Note that Matlab has automatically normalised the Eigenvectors such that given a vector \( \begin{pmatrix} x \\ y \end{pmatrix} \) say, the values of \( x \) and \( y \) are scaled such that \( \sqrt{x^2 + y^2} = 1 \) (recall that only the direction of an Eigenvector is important, not the actual values).

**Task 3**

(a) What output do you get if you type \( v(1,1) \) into the command window?

(b) What output do you get if you type \( v(2,1) \) into the command window?

(c) What output do you get if you type \( v(:,1) \) into the command window?

These commands extract specific values from the matrix \( v \). The first integer gives the row number and the second integer the column number.

(d) Separate the matrix \( v \) into the two separate Eigenvectors \( v_1 \) and \( v_2 \).

(e) Use element by element division (\( ./ \)) to find an integer form of the Eigenvectors (ie. a way of writing the Eigenvectors such that both elements are integers).

5. Powers of matrices

Note that, just like multiplication, there are two ways of taking a power of a matrix, depending on whether or not you want the operation to be performed element-by-element. Try typing the code on the following page into Matlab, and check you understand the results.
Note that the “usual” interpretation of matrix powers, and the one used in the lectures, is as repeated matrix multiplication (i.e., the version that is not taken element-by-element and so does not involve a dot).

6. For you to do

**Task 4**

Use Matlab to find the Eigenvalues and Eigenvectors for the matrices G and K defined as

\[
G = \begin{pmatrix} 4 & 1 \\ 6 & 3 \end{pmatrix} \quad \text{and} \quad K = \begin{pmatrix} 5 & 2 & 0 \\ 2 & 6 & 2 \\ 0 & 2 & 7 \end{pmatrix}.
\]

Verify the calculation by hand.

**Task 5**

If the matrix F and the vector z are defined as

\[
F = \begin{pmatrix} 6 & 8 \\ 4 & 2 \end{pmatrix} \quad \text{and} \quad z = \begin{pmatrix} 1 \\ 2 \end{pmatrix}.
\]

Do the following calculations by hand, then check your answers using Matlab

(a) Find the Eigenvectors and Eigenvalues of F.

(b) Express z as \( z = \alpha v_1 + \beta v_2 \) where \( v_1 \) and \( v_2 \) are the Eigenvectors of F and where \( \alpha \) and \( \beta \) are constants that you should determine.

The lecture tomorrow will explore why this might be a good idea.

You should aim to get at least to this point by the end of the practical.

**Optional programming task for keen programmers only**

Write a function which
- Takes as input two matrices of any size
- Checks the matrices can be multiplied
- Uses for loops to calculate the product of the two matrices
- Outputs the resultant matrix

Check your function works by comparing your output with the built-in matrix multiplication.