MATLAB Project #1: Getting Started with MATLAB

Note:
Record and hand in the printed the MATLAB commands and the results. Also, answer the questions.

Purpose:
To be familiar with the working environment of MATLAB and the various MATLAB commands. This can be used as a brief tutorial and as a reference for basic operations. Use MATLAB's help, helpwin, and demo commands or see a User’s Guide for more information.


1. A matrix is a rectangular array, and in linear algebra the entries will usually be numbers. The most direct way to create a matrix is to type the numbers between square brackets, using a space or comma to separate different entries and a semicolon or [Enter] to create row breaks.

Examples:

\[
\begin{align*}
\text{>> A} & = \begin{bmatrix} 1 & 2; 3 & 4; 5 & -6 \end{bmatrix} \text{[Enter]} \\
\text{>> B} & = \begin{bmatrix} 1 & -2 & 3 \\ 4 & 5 & -6 \end{bmatrix} \text{[Enter]} \\
\text{>> x} & = \begin{bmatrix} 4; 3.2; 2 \end{bmatrix} \text{[Enter]} \\
\text{>> X} & = \begin{bmatrix} 1, 2, 3 \end{bmatrix} \text{[Enter]}
\end{align*}
\]

To see a matrix you have created, type its name followed by [Enter]. Notice MATLAB is case sensitive -- for example, \( x \) and \( X \) are names for different objects:

\[
\begin{align*}
\text{>> A} & \text{ [Enter]} \\
\text{>> A, B} & \text{ [Enter]} \\
\text{>> X, x;} & \text{ [Enter]}
\end{align*}
\]

When you place a semicolon at the end of a command, the command will be executed but whatever it creates will not be displayed on the screen.

\[
\begin{align*}
\text{>> A; B; [Enter]} \\
\text{>> X, x; [Enter]}
\end{align*}
\]

2. The arrow keys. MATLAB keeps about 30 of your most recent command lines in memory and you can "arrow up" to retrieve a copy of any one of those. This can be useful when you want to correct a typing error, or execute a certain command repeatedly. Type the following line and record the error message:

\[
\begin{align*}
\text{>> Z} & = \begin{bmatrix} 1 & 2 & 3 & 4; 5 & 0 \end{bmatrix} \text{[Enter]} \\
\text{Error message:}
\end{align*}
\]
To correct such an error, you could retype the entire line. But this is easier: press the up arrow key on your keyboard one time to retrieve that last line typed, use the left arrow key to move the cursor so it is between 2 and 3, type a semicolon and then press [Enter] to cause the new line to execute. You can also use the right arrow key to move to the right through a line, and if you "arrow up" too far, use the down arrow key to back up. To erase characters, use the BackSpace or Delete keys. It does not matter where the cursor is when you press [Enter] to execute the line.

Exercise 1: Create the following matrices:

\[
A = \begin{bmatrix} 1 & 2 \\ 3 & -4 \\ 5 & 6 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & -2 & 3 \\ 4 & 5 & -6 \end{bmatrix}, \quad C = \begin{bmatrix} 2.5 & -1 \\ 0 & -2 \end{bmatrix},
\]

\[
D = \begin{bmatrix} 2 & -1 \\ \pi & 3 \\ -2 & 1 \end{bmatrix}, \quad E = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, \quad F = \begin{bmatrix} 3 \\ -5 \\ e \end{bmatrix}, \quad I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]

Notice that one entry in \( C \) is a decimal; this is why MATLAB displays every entry as a decimal. The constant \( \pi \) is denoted as \texttt{pi} in MATLAB. The constant \( e \) in the matrix \( F \) stands for the base of the natural logarithm, \( e = 2.71828…. \) Try using the exponential function \texttt{exp(.)}.

Question: Is there a simpler way to construct the matrix \( I \)?


1. **The size command.** When \( M \) is a matrix, the command \texttt{size(M)} returns a vector with two entries which are the number of rows and the number of columns in \( M \).

Example:

\[
>> \text{size(A)} \quad [\text{Enter}] \quad (\text{The [Enter] key is omitted afterwards.})
\]

\[
\text{ans} = \\
3 \ 2
\]

Note: \texttt{ans} is a temporary name for the last thing you calculated, if you did not assign a name for that result.

2. **Accessing particular matrix entries.** If you want to see a matrix which you have stored, type its name. To correct entries in a stored matrix, you must reassign them with a command. Try the following commands to view and change various things in the matrix \( D \) you created above. In each part type the first command line to see what the matrix and certain entries look like before you change them; then type the second command line to cause a change.
Examples: Type and see what happens

```
>> D, D(3,1)
>> D(3,1) = 2
>> D
>> D([1 3], [1 2])
>> D, D([1 3], :)
```

**Exercise 2:**
1. Calculate the size of each of the matrices you have created, B, C, D, E, F, I, X, x.
2. Display D(:, 2). Change D(3, 2) to -1.

**Question:** Try the **ans** and **whos** commands. What do these commands do?

**Note:**
1. The effect of the colon in D(:, 2) is to say "take all rows", and its effect in D([1 3], :) is to say "take all columns."
2. The command **help** can provide immediate assistance for any command whose name you know. For example, type **help size**. Also try **help help**.

**Part 3. Special MATLAB functions for creating matrices: eye, zeros, ones, diag.**

Examples: Type and see what happens

```
>> eye(3)
>> zeros(3)
>> zeros(3,5)
>> ones(2,3)
>> ones(size(D))
>> diag([4 5 6 7])
>> diag([4 5 6 7], -1)
>> C, diag(C)
```

**Exercise 3:**
Type simple commands to create the following matrices. For each, record the command you used:

\[
\begin{bmatrix}
1 & 0 \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
0 & 0 \\
0 & 0
\end{bmatrix}
\begin{bmatrix}
1 & 1 \\
1 & 1
\end{bmatrix}
\begin{bmatrix}
7 & 0 & 0 & 0 \\
0 & 0.1 & 0 & 0 \\
0 & 0 & 6 & 0 \\
0 & 0 & 0 & -2
\end{bmatrix}
\]

**Part 4. Using the colon to create vectors with evenly spaced entries.**

In linear algebra, a **vector** is an ordered n-tuple; thus one-row and one-column matrices like those we called x and X above would be called vectors. Frequently it is useful to be able to create a vector with evenly spaced entries (for example, in loops, or to create data
for plotting). This is easy to do with the colon (:).

Examples:
\[
\begin{align*}
\text{>> v1} & \text{ = } 1:5 \\
\text{>> v2} & \text{ = } 1:0.5:3 \\
\text{>> v3} & \text{ = } 5:-1:-2 \\
\end{align*}
\]

**Exercise 4.**
Use the colon notation to create each of the following vectors. Record the command you used for each:

(a) \([-1 \ 0 \ 1 \ 2 \ 3 \ 4]\)
(b) \([4 \ 3.5 \ 3 \ 2.5 \ 2 \ 1.5 \ 1]\)
(c) The numbers from 0 to 2, spaced 0.1 apart.

**Part 5. The format command.**

This controls how things look on the screen. Type each of the following commands and record the result carefully. Notice that "e" means exponent -- in fact, it means multiply by some power of 10 – for example, 1.2345e002 is the number 1.2345 \times 10^2.

Examples:
Try these commands and describe the results:
\[
\begin{align*}
\text{>> R} & \text{ = } 123.125 \\
\text{>> format long, R} \\
\text{>> format short e, R} \\
\text{>> format short, R} \\
\end{align*}
\]

The default mode for display of numbers is **format short**. To restore the default mode at any time, type **format**. The command **format compact** is very useful. It reduces the number of blank lines on the screen, allowing you to see more of what you have done recently.

**Exercise 5.**
Try the following and describe the effect of each:
\[
\begin{align*}
\text{>> C, D} \\
\text{>> format compact, C, D} \\
\text{>> format, C, D} \\
\end{align*}
\]

**Question:** Does the display format affect the internal precision of a number in MATLAB? Verify it.