What is MATLAB®?

MATLAB®

- Short for Matrix Laboratory → matrix data structures are at the heart of programming in MATLAB
- We will consider arrays in $n$-dimensions where

\[ 1D \implies \text{vectors and } 2D \implies \text{matrices, e.t.c.} \]
What is MATLAB®?

MATLAB®

- Short for Matrix Laboratory → *matrix data structures are at the heart of programming in MATLAB*
- We will consider **arrays** in *n*-dimensions where

  \[ 1D \implies \text{vectors and } 2D \implies \text{matrices, e.t.c.} \]

- In addition MATLAB is also a **programming language**
  - It is an interpreted language (i.e no need to compile)
  - Commands are executed line by line
Why MATLAB®?

1. MATLAB is used by engineers and scientists worldwide to analyze and design products and systems.
Why MATLAB®?

1. MATLAB is used by engineers and scientists worldwide to analyze and design products and systems.

2. MATLAB is a powerful programming tool for Mathematics and Statistics because it has built-in functions for performing a wide variety of mathematical operations, plotting and visualization of data.
Why MATLAB®?

1. MATLAB is used by engineers and scientists worldwide to analyze and design products and systems.

2. MATLAB is a powerful programming tool for Mathematics and Statistics because it has built-in functions for performing a wide variety of mathematical operations, plotting, and visualization of data.

3. MATLAB has additional packages that can be installed for specific subject areas, e.g., Symbolic Math Toolbox™ for performing calculus on functions, Deep Learning Toolbox™ for exploring AI algorithms.
Why MATLAB®?

1. MATLAB is used by engineers and scientists worldwide to analyze and design products and systems.
2. MATLAB is a powerful programming tool for Mathematics and Statistics because it has built-in functions for performing a wide variety of mathematical operations, plotting and visualization of data.
3. MATLAB has additional packages that can be installed for specific subject areas, e.g., Symbolic Math Toolbox™ for performing calculus on functions, Deep Learning Toolbox™ for exploring AI algorithms.
4. The MATLAB programming environment allows one to customize programs to fit specifications.
Objectives

- Familiarize yourself with the MATLAB programming environment
  
  At the most basic level we can think of MATLAB as a very powerful graphing calculator! (with many built-in functions)

- Learn how to use MATLAB to:
  1. execute simple commands.
  2. create variables and display their values
MATLAB WINDOWS

MATLAB has 8 main windows, the first 4 are on the main screen by default.

<table>
<thead>
<tr>
<th>Window</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command window</td>
<td>Enter commands and variables, run programs</td>
</tr>
<tr>
<td>Command history window</td>
<td>History of commands entered in window</td>
</tr>
<tr>
<td>Workspace window</td>
<td>Information about current variables</td>
</tr>
<tr>
<td>Current directory window</td>
<td>Shows files in the current folder/directory</td>
</tr>
<tr>
<td>Figure window</td>
<td>Output from graphic commands</td>
</tr>
<tr>
<td>Editor window</td>
<td>Create and debug scripts and functions</td>
</tr>
<tr>
<td>Help window</td>
<td>Get help on functions and scripts</td>
</tr>
<tr>
<td>Launch pad window</td>
<td>Access to tools, demos and documentation</td>
</tr>
</tbody>
</table>
MATLAB WINDOWS

MATLAB has 8 main windows, the first 4 are on the main screen by default

<table>
<thead>
<tr>
<th>Command window</th>
<th>Enter commands and variables, run programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command history window</td>
<td>History of commands entered in window</td>
</tr>
<tr>
<td>Workspace window</td>
<td>Information about current variables</td>
</tr>
<tr>
<td>Current directory window</td>
<td>Shows files in the current folder/directory</td>
</tr>
<tr>
<td>Figure window</td>
<td>Output from graphic commands</td>
</tr>
<tr>
<td>Editor window</td>
<td>Create and debug scripts and functions</td>
</tr>
<tr>
<td>Help window</td>
<td>Get help on functions and scripts</td>
</tr>
<tr>
<td>Launch pad window</td>
<td>Access to tools, demos and documentation</td>
</tr>
</tbody>
</table>

Some commands

- `clc` - Clear command window
- `clf` - Clear figure
- `exit` or `quit` - Quit MATLAB
- `disp('text')` - Displays text
Variables and Expressions

**Variable** - A value that can change depending on conditions or information passed to the program. In contrast, a program may have data values that are fixed constants.

\[ \text{Expressions} \]

These are created using values, variables previously created, mathematical operators and built-in functions e.g. 

\[ \text{>> 2*cos(1) + sqrt(2) } \]

\[ \text{ans} = 2.4948 \]

Adding a semi-colon (;) at the end of the expression suppresses screen output.
Variables and Expressions

**Variable** - A value that can change depending on conditions or information passed to the program. In contrast, a program may have data values that are fixed constants.

**Expressions** - These are created using values, variables previously created, mathematical operators and built in functions e.g.

\[
>> 2\times \cos(1) + \sqrt{2} \\
\text{ans} = \\
2.4948
\]
Variables and Expressions

**Variable** - A value that can change depending on conditions or information passed to the program. In contrast, a program may have data values that are fixed constants.

**Expressions** - These are created using values, variables previously created, mathematical operators and built in functions e.g.

\[
\text{>> } 2\cos(1) + \sqrt{2}
\]

\[
\text{ans = }
\]

\[
2.4948
\]

Adding a semi-colon (;) at the end of the expression suppresses screen output
Expressions and formatting

- The default in MATLAB is to display numbers that have decimal points with 4 decimal places.
- There are many other options
  1. `format long` will result in 15 decimal places
     ```matlab
     >> format long
     >> 2*cos(1) + sqrt(2)
     ans =
     2.494818174109374
     ```
  2. `format shortEng` will result in 4 decimal places engineering format.
     ```matlab
     >> format shortEng
     >> 2*cos(1) + sqrt(2)
     ans =
     2.4948e+000
     ```
Basic formats displaying $\pi$

<table>
<thead>
<tr>
<th>Format Style</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td>format short (default)</td>
<td>3.1416</td>
</tr>
<tr>
<td>format long</td>
<td>3.141592653589793</td>
</tr>
<tr>
<td>format rat</td>
<td>355/113</td>
</tr>
<tr>
<td>format bank</td>
<td>3.14</td>
</tr>
<tr>
<td>format short e</td>
<td>3.1416e+00</td>
</tr>
<tr>
<td>format long e</td>
<td>3.141592653589793e+00</td>
</tr>
<tr>
<td>format compact</td>
<td>(no blank lines)</td>
</tr>
</tbody>
</table>
Creating Variables

We create variables in MATLAB using an assignment statement

```matlab
>> variablename = expression
```
Creating Variables

We create variables in MATLAB using an assignment statement

```matlab
>> variablename = expression
```

**REMARKS**

1. The ‘=' is an assignment operator (unlike the mathematics equal sign, does not mean equality)
2. The `expression` is evaluated at the result is stored in `variablename`
Creating Variables

We create variables in MATLAB using an assignment statement

```
>> variablename = expression
```

**REMARKS**

1. The ‘=’ is an assignment operator (unlike the mathematics equal sign, does not mean equality)
2. The `expression` is evaluated at the result is stored in `variablename`

*Use variable names that make sense, e.g. if a variable to store the radius, the name `radius` makes sense, `y` does not!*
Rules for variable names

1. Must begin with a letter of the alphabet, after that it may contain letters, digits, and the underscore but NO SPACES! e.g. `interest_rate, my_num_1`, e.t.c.

2. The limit to the number of characters in a variable name is 63 stored in `namelengthmax`.

3. MATLAB is case sensitive so `INTEREST_RATE` is different from `Interest_rate`, e.t.c.

4. Certain words are reserved so they cannot be used as variable names.

5. Names of built-in functions cannot be used as variable names.
Commands related to Variables

1. **who** shows variables that have been defined
2. **whos** shows more information about the type of variables
3. **clear** clears out all variables. You can also specify a specific variable to clear.
Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>addition</td>
</tr>
<tr>
<td>−</td>
<td>negation, subtraction</td>
</tr>
<tr>
<td>/</td>
<td>division (division by, e.g. 10/5 = 2)</td>
</tr>
<tr>
<td>\</td>
<td>division (division into, e.g. 5 \ 10 = 2)</td>
</tr>
<tr>
<td>^</td>
<td>exponentiation (e.g. 5^2 = 25)</td>
</tr>
</tbody>
</table>
### MATLAB builtin functions and constants

- **sqrt(x)**: square root
- **exp(x)**: Exponential ($e^x$)
- **abs(x)**: Absolute value
- **log(x)**: Natural logarithm
- **sin(x)**: Sine of $x$
- **cos(x)**: Cosine of $x$
- **tan(x)**: Tangent of $x$
- **pi**: $\pi$
- **i**: $\sqrt{-1}$
- **inf**: $\infty$
- **NaN**: “not a number”, such as the result of $0/0$. 

Some basic functions and constants:
Help and `lookfor` commands

There are many built-in functions in MATLAB:

- Use `lookfor` can be used to find a specific command to perform a task.
- Use `help` to find out information about a known command.
Vectors and Matrices

- **Vectors** and **matrices** are used to store values of the same types.
- A **vector** can be either *column vector* or a *row vector*.
- **Matrices** can be visualized as a table of values with dimensions $r \times c$ ($r$ is the number of rows and $c$ is the number of columns).
Creating row vectors

Place the values that you want in the vector in square brackets separated by either spaces or commas. e.g.

\[
\text{vec} = [1 \ 2 \ 3 \ 4 \ 5] \\
\text{vec} = \\
\begin{array}{ccccc}
1 & 2 & 3 & 4 & 5 \\
\end{array}
\]

\[
\text{vec} = [1,2,3,4,5] \\
\text{vec} = \\
\begin{array}{ccccc}
1 & 2 & 3 & 4 & 5 \\
\end{array}
\]
Creating row vectors - *colon operator* and *linspace* functions

If the values of the vectors are regularly spaced, the *colon operator* can be used to iterate through these values. e.g.

```
>> vec = 1:5
vec =
  1  2  3  4  5
```

A *step value* can also be specified with another colon in the form *(first:step:last)*

```
>>vec = 1:2:9
vec =
  1  3  5  7  9
```