Matlab variable types

- integer
  - 123, -345, 0

- real or float
  - 12.2344
  - 5.445454

- engineering notation
  - $4.2323 \times 10^{-9}$

- complex
  - $i = \sqrt{-1} = 1i$
  - $34.23 + 21.21i$
  - $(1 + 1i) \times (1 - 1i) = 2$

- strings (put your words inside apostrophes)
  - handy for file names and messages
  - 'programming is fun'
  - s = 'Williamsburg'
Matlab variable types

- integer
  - 123, -345, 0
- real or float
  - 12.2344
  - 5.445454
  - engineering notation
    - \(1.2323 \times 10^{-9}\)
- complex
  - \(i = \sqrt{-1} = 1i\)
  - \(34.23+21.21i\)
  - \((1+1i) \times (1-1i) = 2\)
- strings (put your words inside apostrophes)
  - handy for file names and messages
  - 'programming is fun'
  - `s='Williamsburg'`

Some built in constants and functions

- \(\pi = 3.141592653589793238462643383279502 \cdots\)
  - use `pi`
- trigonometry functions
  - sin, cos, tan, cot
  - sind, cosd, tand, cotd
  - asin, acos, atan, acot
  - asind, acosd, atand, acotd
  - \(\sin(\pi/2)=1\)
  - \(\sin(90)=1\)
- hyperbolic functions
  - sinh, cosh, tanh, coth
  - asinh, acosh, atanh, acoth
- logarithms
  - natural log
  - base of 10 log10
- power
  - \(x^y\) use `x^y` or alternatively `power(x,y)`
  - \(e^y\) use `exp(y)`

Assignment operator

```matlab
x = 1.2 + 3.4
```
Assignment operator

\[ x = 1.2 + 3.4 \]

Despite the look \( = \) is not an equality operator. 
\( = \) is an assignment operator.

The expression above should be read as

- evaluate expression at the right hand side of equality symbol
- assign the result of the RHS to the variable on the left hand sign
- now variable \( x \) holds the value \( 4.6 \)

We are free to use the value of the variable \( x \) in any further expressions

\[ > x + 4.2 \]
\[ \text{ans} = 8.8 \]

Efficient editing - Tab-completion

Once you typed some expressions in “Command window”

- type couple of first symbols of variable or function name
- hit tab and you will get
  - either fully typed name (if it is uniq)
  - or little chart with choices
    - use <up> or <down> arrows to choose
    - alternatively <Ctrl-p>, <Ctrl-n>
  - then hit <enter> to make your choice

Help related commands

These are the most important commands

- docsearch word
  - will search for word in the help files and show up matched help files
  - example: docsearch trigonometry
- help name
  - output short help text into “Command window” about function/method named name
  - example: help sin
- doc name
  - show a reference page about function/method named name in the help browser
  - usually has more information compare to help name
  - example: doc sin
Operator Precedence

Look at the following Matlab expression

\[-2^4 + \tan(\pi/8+\pi/8)^2\]

Guess the answer.

\[- (2^4) + (\tan( (\pi/8+\pi/8) ))^2\]

\[- (16) + (\tan( (\pi/4) ))^2\]

\[-80 + (1)^2\]

= -80 + 1

= -79

Rule of thumb: if not sure use extra parentheses ().

Read more by executing `doc precedence` or searching for 'precedence' in the help browser.
Operator Precedence

Look at the following Matlab expression

\[-2^4*5 + \tan(\pi/8+\pi/8)^2\]

Guess the answer.

\[- (2^4)*5 + (\tan( (\pi/8+\pi/8) ))^2\]

\[- (16)*5 + (\tan( (\pi/4) ))^2\]

\[-80 + (1)^2 = -80 + 1\]

Rule of thumb: if not sure use extra parentheses ()

Read more by executing `doc precedence` or searching for 'precedence' in the help browser.
Matrices

Recall that Matlab stands for Matrix Laboratory
- So deep inside everything is a matrix (array)
- A number is the case of $1 \times 1$ matrix

Let's create a $3 \times 5$ matrix (3 rows and 5 columns)

```matlab
>> Mz=zeros(3,5)
Mz =
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```
This is not the only way, but it is one which make sure that matrix is filled with zeros
Note: It is possible to have more than 2 dimensional arrays.

Matrix elements assignment

```matlab
>> Mz(2,4)=1 % 2nd row, 4th column
Mz =
     0     0     0     1     0
     0     0     0     0     0
     0     0     0     0     0
```

```matlab
>> Mz(3,5)=4 % 3rd row, 5th column
Mz =
     0     0     0     0     4
     0     0     0     1     0
     0     0     0     0     0
```
Alternative way to assign a matrix

- comma separates column elements
- semicolon separates row elements

```
Mz=[...
0, 0, 0, 0, 0;...
0, 0, 1, 0;...
0, 0, 0, 4]
```

Mz =
```
0 0 0 0 0
0 0 0 1 0
0 0 0 0 4
```

Notice ··· mark, which means that input continues on the next line

Strength of Matlab

Native matrix operations

```
Mz =
0 0 0 0
0 0 1 0
0 0 0 4
```

```
>> Mz+5
ans =
5 5 5 5 5
5 5 5 6 5
5 5 5 5 9
```

```
>> Mz*2
ans =
0 0 0 0 0
0 0 0 2 0
0 0 0 0 8
```

More example on matrices operations

```
Mz =
0 0 0 0
0 0 1 0
0 0 0 4
```

```
>> Mz+Mz
ans =
0 0 0 0 0
0 0 0 2 0
0 0 0 0 8
```

Matrix multiplication according to the linear algebra rules

```
>> Mz*Mz'
ans =
0 0 0
0 1 0
0 0 16
```

Here 

```
Mz'
```

corresponds to transposed matrix 

```
Mz
```

(i, j) = 

```
Mz(j, i)
```
More example on matrices operations

$$M_z = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 \end{bmatrix}$$

$$\text{>> } M_z + M_z$$

$$\text{ans} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 8 & 0 \end{bmatrix}$$

Matrix multiplication according to the linear algebra rules

$$\text{>> } M_z \times M_z'$$

$$\text{ans} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 16 \end{bmatrix}$$

Here $M_z'$ corresponds to transposed matrix $M_z$, i.e. $M_z'(i,j) = M_z(j,i)$

Matrix as a function argument

A function can take a matrix as the function argument, it will evaluate the value of the function for each matrix element

$$\text{>> } \sin(M_z)$$

$$\text{ans} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.8415 & 0 \\ 0 & 0 & 0 & -0.7568 & 0 \end{bmatrix}$$

Vectors and column vector

A special case of the matrix is it has only one dimension. Such matrices generally called vectors

- $m \times 1$ column vector
- $1 \times m$ just a vector

To create a vector

$$\text{>> } \text{ use comma to separate column elements}$$

$$v = [1, 2, 3, 4, 5, 6, 7, 8]$$

$$v = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}$$

$$\text{>> } \text{ alternatively you can use spaces}$$

$$v = [1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8]$$

$$v = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}$$

$$\text{>> } \text{ or mix of these two notations (NOT RECOMMENDED)}$$

$$v = [1 \ 2 \ 3, 4, 5, 6 7 8]$$

$$v = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}$$
Column vector

Construction of column vector

```
>> vc=[1; 2; 3]
% use semicolon to separate row elements
vc =
1
2
3
```

Y et one more way to create matrix

If you have prearranged vectors or column vectors you can use them

```
>> vc=[1; 2; 3];
% note that ; after a statement suppresses output
>> Mc=[vc, vc, vc]
Mc =
1 1 1
2 2 2
3 3 3
```

Colon (:) operator

The colon operator is extremely useful to create vectors or matrix indexes. It usually take form `start:increment:stop` and creates a vector with following values

```
>> v=5:2:11
v =
5 7 9 11
>> v2=12:-3:1
v2 =
12 9 6 3
```

Notes
Colon (:) operator

The colon (:) operator is extremely useful to create vectors or matrix indexes. It usually takes the form `start:increment:stop` and creates a vector with following values:

\[ \text{[ start, start+increment, \ldots , start+m*increment]} \]

where

\[ \text{min(start,stop)} \leq m \times \text{increment} \leq \text{max(start,stop)} \]

```matlab
>> v=5:2:11
v =
     5     7     9    11
```

It is also possible to have negative `increment`:

```matlab
>> v2=12:-3:1
v2 =
  12    9     6     3
```

Colon (:) operator continued

Another form `start:stop` in this case `increment = 1`

```matlab
>> v1=1:5
v1 =
     1     2     3     4     5
```

Notice that:

```matlab
>> v3=5:1
v3 =

Empty matrix: 1-by-0
```

Produce somewhat unexpected result, since default increment is positive.
Slicing matrices

It is handy to choose a subset (block) from the matrix. We have a matrix \( M_v \) with size 3 \( \times \) 8 and we want to choose all elements from columns 2,5,6.

\[
\begin{bmatrix}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
2 & 4 & 6 & 8 & 10 & 12 & 14 & 16 \\
3 & 6 & 9 & 12 & 15 & 18 & 21 & 24
\end{bmatrix}
\]

\( M_v(:,[2,5,6]) \)

\[
\begin{bmatrix}
2 & 5 & 6 \\
4 & 10 & 12 \\
6 & 15 & 18
\end{bmatrix}
\]

The meaning of the colon is choose all. Notice also that we use vector to specify desired columns.

Plotting

Suppose you have a vector with values of \( x \) coordinates and we want to plot \( \sin(x) \).

\[
\begin{align*}
\text{x} &= \text{linspace}(0,2\pi,10) \\
\text{y} &= \sin(\text{x})
\end{align*}
\]

\[
\begin{bmatrix}
0 & 0.6428 & 0.9848 & 0.8660 & 0.3420 & -0.3420 & -0.8660 & -0.9848 & -0.6428 & -0.0000
\end{bmatrix}
\]

\( \text{plot(x,y,'o')} \) % alternatively plot(x,sin(x),'o')

\( \text{xlabel('x (radians)')} \)

\( \text{ylabel('\sin(x)'}) \)

\( \text{title('Plot of \sin(x)'}) \)

Saving plots

Now we want to save the figure, use \textit{print}.

\( \text{print('-dpdf', 'sin_of_x')} \)

This will generate file \textit{sin_of_x.pdf} notice automatic fileextension addition.

For 3D plots, please see help files for \textit{plot3, mesh, surf}.
Now we want to save the figure, use `print`

```matlab
>> print('-dpdf', 'sin_of_x')
```

This will generate file `sin_of_x.pdf` notice automatic file extension addition.
The `-d` switch stands for output format (`pdf`, `ps`, `eps`, `png`...)
To generate `png` file

```matlab
>> print('-dpng', '-r100', 'sin_of_x')
```

By default figure size is 8 x 6 inches, the `-r` switch tells the figure resolution in dpi (dots per inch). In this case it is 100 dpi so resulting image will be 800 x 600 pixels.

For 3D plots, please see help files for `plot3`, `mesh`, `surf`

---

Special array arithmetic operators

There are special arithmetic operators which applied to the elements of matrices (disregard linear algebra rules), they start with .

```matlab
>> x=1:3
x = 1 2 3
>> x.*x % will generate an error
>> x.*x % equivalent to x.^2 (see below)
ans = 1 4 9
```

```matlab
>> x.^2
ans = 1 4 9
```

```matlab
>> x./x
ans = 1 1 1
```
Special array arithmetic operators continued

>> m = [1,2,3;4,5,6;7,8,9]
m =
1 2 3
4 5 6
7 8 9

Linear algebra rules

>> m * m
ans =
30 36 42
66 81 96
102 126 150

Element wise operation

>> m .* m
ans =
1 4  9
16 25 36
49 64 81

>> m^m % undefined
ans =
1 4  27
256 3125 46656
823543 16777216 387420489

Element wise operation

>> m.^m
ans =
1  4  27
256 3125 46656
823543 16777216 387420489

Special array arithmetic operator ./

>> m = [1,2,3;4,5,6;7,8,9]
m =
1 2 3
4 5 6
7 8 9

Linear algebra rules

>> m/m % unity matrix
ans =
1 0 0
0 1 0
0 0 1

Element wise operation

>> m./m % matrix of ones
ans =
1 1 1
1 1 1
1 1 1