Matlab variable types

- integer
  - 123, -345, 0
- real or float
  - 12.2344
  - 5.445454
- engineering notation
  - $4.2323 \times 10^{-9}$
- imaginary
  - $1i = \sqrt{-1}$
  - $34.23 + 21.21i$
  - $(1 + 1i) \times (1 - 1i) = 2$
- 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**
  - \(4.232\times 10^{-9}\)
- **imaginary**
  - \(\sqrt{-1}\)
  - \(34.23 + 21.21i\)
  - \((1+1i) \times (1-1i) = 2\)
- **strings (put your words inside apostrophes)**
  - handy for file names and messages
  - 'programming is fun'
  - s='Williamsburg'

Some built in constants and functions

- \(\pi = 3.141592653589793238462643383279502 \ldots\)
- use `pi`
- **trigonometry functions**
  - By default angle is in radians
  - But can be done in degrees
  - `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`
  - `log10`
- **power**
  - `x^y` use `x^y` or alternatively `power(x,y)`
  - `e^y` use `exp(y)`

Assignment operator

\[x = 1.2 + 3.4\]
Assignment operator

\[ x = 1.2 + 3.4 \]

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

\[ \gg x = 1.2 + 3.4 \]
\[ x = 4.6000 \]

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.

\[ \gg x + 4.2 \]
\[ \text{ans} = 8.8000 \]

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

Look at the following Matlab expression

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

Guess the answer.

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

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

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

\[-80 + 1\]

\[-79\]

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

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

Look at the following Matlab expression

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

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
  - also referred as array or table
  - a number is the case of $1 \times 1$ matrix

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

```
>> 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

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

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

- comma separates column elements
- semicolon separates row elements

```matlab
>> Mz=[ ... 
0, 0, 0, 0, 0; ... 
0, 0, 0, 1, 0; ... 
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

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

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

More example on matrices operations

```matlab
>> 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

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

Here $Mz'$ corresponds to transposed matrix $Mz$, i.e. $Mz'(i,j) = Mz(j,i)$
More example on matrices operations

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

\[ \begin{array}{c}
\text{>> } M_z + M_z \\
\text{ans} = \\
\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 8 \end{bmatrix}
\end{array} \]

Matrix multiplication according to the linear algebra rules

\[ \begin{array}{c}
\text{>> } M_z \times M_z' \\
\text{ans} = \\
\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 16 \end{bmatrix}
\end{array} \]

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

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

\[ \begin{array}{c}
\text{>> } \sin(M_z) \\
\text{ans} = \\
\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.8415 & 0 \\ 0 & 0 & 0 & -0.7568 & 0 \end{bmatrix}
\end{array} \]

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

\[ \begin{array}{c}
\text{>> } \text{use comma to separate column elements} \\
\text{v} = [1, 2, 3, 4, 5, 6, 7, 8] \\
\text{v} = \\
\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}
\end{array} \]

\[ \begin{array}{c}
\text{>> } \text{alternatively you can use spaces} \\
\text{v} = [1 2 3 4 5 6 7 8] \\
\text{v} = \\
\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}
\end{array} \]

\[ \begin{array}{c}
\text{>> } \text{or mix of these two notations (NOT RECOMMENDED)} \\
\text{v} = [1 2 3, 4, 5, 6 7 8] \\
\text{v} = \\
\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}
\end{array} \]
Column vector

Construction of column vector

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

Yet one more way to create matrix

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

```matlab
>> 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
```

```matlab
v =
1 2 3 4 5 6 7 8
>> Mv=[v; 2*v; 3*v]
Mv =
1 2 3 4 5 6 7 8
2 4 6 8 10 12 14 16
3 6 9 12 15 18 21 24
```

Colon (:) operator

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

```matlab
[ start, start+increment, ... , start+m*increment]
```

where

```matlab
m=1, 2, 3, 4, ... and
min(start,stop)≤start + m*increment≤max(start,stop)
```
Colon (:) operator

The colon operator is extremely useful to create vectors or matrix indexes. It usually takes the form \[ \text{start} : \text{increment} : \text{stop} \] and creates a vector with following values:

\[ [ \text{start}, \text{start} + 1 \times \text{increment}, \ldots, \text{start} + m \times \text{increment} ] \]

where

\[ m = 1, 2, 3, 4, \ldots \]

and

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

\[ \gg \ v = 5:2:11 \]

\[ v = [5, 7, 9, 11] \]

It is also possible to have negative increment.

\[ \gg \ v2 = 12:-3:1 \]

\[ v2 = [12, 9, 6, 3] \]

Colon (:) operator continued

One can use form \[ \text{start} : \text{stop} \] with the default \[ \text{increment} = 1 \]

\[ \gg \ v1 = 1:5 \]

\[ v1 = [1, 2, 3, 4, 5] \]

But there are some peculiarities:

\[ \gg \ v3 = 5:1 \]

\[ v3 = \text{Empty matrix: 1-by-0} \]

provides somewhat unexpected result, naively you would expect \[ v3 = 5 \]. But there are some built extra conditions, see them by executing

\[ \gg \ \text{help} : \]
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.

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

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

The meaning of the \( : \) now 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 \( \sin(x) \).

\[
\begin{bmatrix}
0 & 0.6981 & 1.3963 & 2.0944 & 2.7925 & 3.4907 \\
4.1888 & 4.8869 & 5.5851 & 6.2832 \\
\end{bmatrix}
\]

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

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

Increasing font size for plots
Default font size

\[
\begin{bmatrix}
\end{bmatrix}
\]

Increased font size

\[
\begin{bmatrix}
\end{bmatrix}
\]

To save the figure use `print`

\[
\begin{bmatrix}
\end{bmatrix}
\]

This will generate file `sin_of_x.pdf` notice automatic file extension addition.
Saving plots

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 designates the output format:

```
pdf, ps, eps, png...
```

Matlab still generates `pdf` with a lot of empty space unsuitable for use as figures. It is better to save into `eps` format and then convert it to a desired one.

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

Or generate a `png` file which can be directly used with `pdflatex`

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

By default figure size is 8 × 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 × 600 pixels.

Array element-wise arithmetic operators

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

- `.*` element-wise multiplication
  ```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
  ```

- `.^` element-wise power operator
  ```matlab
  >> x.^2
  ans = 1 4 9
  ```
Array element-wise arithmetic operators

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

- **element-wise multiplication**

  ```
  >> 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
  ```

- **element-wise power operator**

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

- **element-wise division**

  ```
  >> x./x
  ans = 1 1 1
  ```

Array element-wise arithmetic operators continued

```
>> m=[1,2,3;4,5,6;7,8,9]
m =
 1  2  3
 4  5  6
 7  8  9
>> m*m
ans =
 30  36  42
 66  81  96
102 126 150
>> m.*m
ans =
  1   4   9
 16  25  36
49  64  81
```