MATLAB TUTORIAL

1) Matlab Panels

2) Operations and Variables

3) Matrices

4) Scripts

5) Plotting

6) Simulink
1) **Matlab panels:**

- **Command Window:** You can enter the commands after the `>>` sign.
- **Command History:** Return the commands that you enter at the command line. To clear the Command History click right on it then click “Clear Command History”.
- **Workspace:** Data that you create or import are explored here.

2) **Operations and Variables:**

<table>
<thead>
<tr>
<th>Command</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&gt;&gt; (-1+5)*4/3</code></td>
<td><code>ans = 5.3333</code></td>
</tr>
<tr>
<td><code>&gt;&gt; log(10)</code></td>
<td><code>ans = 2.3026</code></td>
</tr>
<tr>
<td><code>&gt;&gt; exp(0)</code></td>
<td><code>ans = 1</code></td>
</tr>
<tr>
<td><code>&gt;&gt; sqrt(81)</code></td>
<td><code>ans = 9</code></td>
</tr>
<tr>
<td><code>&gt;&gt; cos(pi)</code></td>
<td><code>ans = -1</code></td>
</tr>
<tr>
<td><code>&gt;&gt; sin(30*pi/180)</code></td>
<td><code>ans = 0.5000</code></td>
</tr>
</tbody>
</table>

Matlab has an online documentation facility. You can access it by `help` or `doc` command (e.g. `>> help sin`)
You can also use $j$ to indicate the complex numbers. $\pi$ has the value 3.14159.

Inside the cos, sin, tan, atan, etc. commands, you should write the angle in terms of radian. So, convert the degree to radian!

\[
\text{Y} = \text{ceil}(\text{X}) \text{ rounds each element of X to the nearest integer greater than or equal to that element.}
\]
\[
\text{Y} = \text{floor}(\text{X}) \text{ rounds each element of X to the nearest integer less than or equal to that element.}
\]
\[
\text{Y} = \text{round}(\text{X}) \text{ rounds each element of X to the nearest integer. In the case of a tie, where an element has a fractional part of exactly 0.5, the round function rounds away from zero to the integer with larger magnitude.}
\]


In Matlab, you should define the variables. Otherwise it gives error!

When you put the ; at the end of the command it still define the variable and write to the workspace. In order to get a clear command window use ; at the end of the commands.

If the variables that you want to use do not have a specific value, then you should define them as a symbolic function by `syms` command.
Cleaning up symbolic statements:

```
>> simplify (ans) simplifies ans
```

```
syms a
X=(cos(a))^2+(sin(a))^2
simplify(X)
ans = 1
```

3) Matrices:

Comma or space seperated values construct rows. You can create columns by using ; sign.

<table>
<thead>
<tr>
<th>Constructing a matrix</th>
<th>Size</th>
<th>Eigenvalues</th>
<th>Inverse of a matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;&gt; A=[0 1 3; 2 4 -6; 3 1 7] A =</td>
<td>&gt;&gt; size(A)</td>
<td>&gt;&gt; eig(A)</td>
<td>&gt;&gt; inv(A)</td>
</tr>
<tr>
<td></td>
<td>ans = 3 3</td>
<td>ans =</td>
<td>ans =</td>
</tr>
<tr>
<td></td>
<td>0 1 3</td>
<td>-1.4900 + 0.0000i</td>
<td>-0.5484 0.0645 0.2903</td>
</tr>
<tr>
<td></td>
<td>2 4 -6</td>
<td>6.2450 + 1.6156i</td>
<td>0.5161 0.1452 -0.0968</td>
</tr>
<tr>
<td></td>
<td>3 1 7</td>
<td>6.2450 - 1.6156i</td>
<td>0.1613 -0.0484 0.0323</td>
</tr>
</tbody>
</table>

Transpose

```
>> A' gives the same result
```

```
[V,D] = eig(A) : Eigenvalue decomposition
[U,S,V]=svd(A) : Singularvalue decomposition
[Q,R] = qr(A) : QR decomposition
```

```
>> det(A) gives the determinant of A matrix
```

```
>> rank(A) gives the rank of A matrix
```

```
eye(n) creates an identity matrix with a size n x n
zeros(n) creates a zero matrix with a size n x n
ones(n) creates a matrix consisting of 1 with a size n x n
```
4) Scripts

To create a new editor document, write `>> edit` at the command line or click to the “New Script” as shown in the following figure.

Why do we use editor (m file)?

- To make changes on it
- To save and use it later
* means that it is not saved. Save the file and then press the Run button.

Anything following percent sign (%) is a comment! Comments helps you to understand the code when you use it later. Hence, you can save your time.

You cannot run the commands that you write to the editor unless you don’t save it!
5) **Plotting**

\( y = \text{linspace}(n,m) \) returns a row vector of 100 evenly spaced points between \( n \) and \( m \).
\( y = \text{linspace}(n,m,k) \) generates \( n \) points. The spacing between the points is \((m-n)/(k-1)\).

```
>> t=(1:5)
t =
   1  2  3  4  5
>> t=(1:0.5:5)
t =
   1.0000  1.5000  2.0000  2.5000  3.0000  3.5000  4.0000  4.5000  5.0000
```

\( t(n:m) \) creates a vector from \( n \) to \( m \) with 1 increment

\( t(n:k:m) \) creates a vector from \( n \) to \( m \) with \( k \) increment

2D Plotting:

```
>> x=linspace(1,5);
>> y=sin(x);
>> plot(x,y)
x and y vectors must be same size. Otherwise you get an error!
```

You need to write axes names and put a title and/or legend.

```
>> x=linspace(1,5);
>> y=sin(x);
>> plot(x,y)
>> xlabel('RPM')
>> ylabel('Thrust')
>> title('Angular velocity versus thrust')
>> legend('A')
```

```
Angular velocity versus thrust

Thrust

1 1.5 2 2.5 3 3.5 4 4.5 5
-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 1

RPM
```
You can also insert x label, y label, title or legend by clicking on the insert part on the graph.

```matlab
x=linspace(1,5);
y=sin(x);
plot(x,y,'g*-')
hold on
z=cos(x)
plot(x,z,'r*-')
xlabel('RPM')
ylabel('Thrust')
title('Omega versus thrust')
legend('y=sin(x)','z=cos(x)','Location','southwest')
```
<table>
<thead>
<tr>
<th>Specifier</th>
<th>Line Style</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>Solid line (default)</td>
</tr>
<tr>
<td>--</td>
<td>Dashed line</td>
</tr>
<tr>
<td>:</td>
<td>Dotted line</td>
</tr>
<tr>
<td>-.,</td>
<td>Dash-dot line</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Specifier</th>
<th>Marker</th>
</tr>
</thead>
<tbody>
<tr>
<td>o</td>
<td>Circle</td>
</tr>
<tr>
<td>+</td>
<td>Plus sign</td>
</tr>
<tr>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>.</td>
<td>Point</td>
</tr>
<tr>
<td>x</td>
<td>Cross</td>
</tr>
<tr>
<td>s</td>
<td>Square</td>
</tr>
<tr>
<td>d</td>
<td>Diamond</td>
</tr>
<tr>
<td>^</td>
<td>Upward-pointing triangle</td>
</tr>
<tr>
<td>v</td>
<td>Downward-pointing triangle</td>
</tr>
<tr>
<td>&gt;</td>
<td>Right-pointing triangle</td>
</tr>
<tr>
<td>&lt;</td>
<td>Left-pointing triangle</td>
</tr>
<tr>
<td>p</td>
<td>Pentagram</td>
</tr>
<tr>
<td>h</td>
<td>Hexagram</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Specifier</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>yellow</td>
</tr>
<tr>
<td>m</td>
<td>magenta</td>
</tr>
<tr>
<td>c</td>
<td>cyan</td>
</tr>
<tr>
<td>r</td>
<td>red</td>
</tr>
<tr>
<td>g</td>
<td>green</td>
</tr>
<tr>
<td>b</td>
<td>blue</td>
</tr>
<tr>
<td>w</td>
<td>white</td>
</tr>
<tr>
<td>k</td>
<td>black</td>
</tr>
</tbody>
</table>
Figure Reference: MIT Introduction to Matlab, Lecture 2

Drawing subplots

```
x = linspace(0,10);
y1 = sin(x);
y2 = sin(5*x);
figure
subplot(2,1,1);
plot(x,y1)
subplot(2,1,2);
plot(x,y2)
x = linspace(0,10);
y1 = sin(x);
y2 = sin(5*x);
y3 = sin(7*x);
figure
subplot(3,1,1);
plot(x,y1)
subplot(3,1,2);
plot(x,y2)
subplot(3,1,3);
plot(x,y3)
```

```
subplot(x,y,z)
```

Makes x row
Makes y column
Activates zth one

e.g. subplot(2,1,2) means the figure consists of 2 rows, one column, and activates second one.

>> close all command closes all figures.
>>close[n m] command closes nth and mth figures.

3D Plotting:

```matlab
>> time=(0:0.4:10);
>> z=time;
>> x=sin(time);
>> y=cos(time);
>> plot3(x,y,z,'b','Linewidth',2)
```

Surface and mesh plots

```matlab
x=pi:0.1:pi;
y=pi:0.1:pi;
[a,b]=meshgrid(x,y); % to make matrices
Z=sin(a).*cos(b);
surf(x,y,Z)
```

```matlab
x=pi:0.1:pi;
y=pi:0.1:pi;
[a,b]=meshgrid(x,y); % to make matrices
c=sin(a).*cos(b);
contour(a,b,c)
```
6) **Simulink**

Simulink library browser includes various blocks that you need.

**Example: Mass-spring-damper**

**References:**

*Introduction to Programming in Matlab, MIT opencourseware*