

# MATLAB™ Command Summary

Copyright, 2002, Ravenna Park Publishing, Inc., Seattle, 206-524-3375

## Startup: Go!

open Matlab  
set path to your  
storage area  
**help**  
**help commandname**

## Stop!

**save m-files**  
**save workspace**

## Workspaces

Command workspace & function workspace are different. Transfer between them:

use GLOBAL in both  
**GLOBAL Re Nu**

Test the function and display all variables supposedly transferred.

Transfer name of fn:

**ode45(@rhs,...)**  
**ode45('rhs',...)**

## Display tools

remove semi-colons

**format long:** 1.234567890e+00

**format long e:** 1.234567890e+00

**format short:** 123.4568

**format short e:** 1.2346e+02

**disp(x):** x = 2.34

or just say **x:** = 2.34

**disp('now is the time')**

now is the time

To eliminate all variables

**clear all**

## Find Errors

???Undefined function or variable 'q'.

If q is a variable: it hasn't been set or wasn't transferred correctly. If q is a function or m-script: then MATLAB can't find it; set **set path**.

??? Index exceeds matrix dimensions.

Either the matrix A(i,j) was called with i or j too big, i.e. beyond the definition of A, or a function uses a matrix which has not been defined.

Use **pause** (to view displayed output).

## Matrices

Vectors are row (1x3)

**x = [2 3 4]** or

column (3x1)

**w = [ 2**

**3**

**4]**

Transpose is **w = x'**

Set matrix values

**A = [ 1 2 3 4 carriage return**

**5 6 7 8]**

A' is the transpose of A

**size(u)** gives dimensions of A

**size(A) = 2 4**

Operate on all elements at once.

If T(:) is a vector of T (K) we

might want it in C. **t = T - 273**

Operate on one element of

matrix

**t(i) = T(i) - 273**

Obtain rows from matrix

**a = A(1,:) or a = [1 2 3 4]**

Can use a or A(1,:) in calculations or plotting.

## Matrix multiplication

**A = [ 2 4 6**

**8 10 12**

**14 16 18]**

is 3x3

**A \* x' = [40**

**94**

**148]**

is [3x3][3x1] = [3x1]

Multiply terms element by element:

**x .\* x = [ 4 9 16]**

**x \* x' = 29**

is [1x3][3x1] = [1x1]

**x' \* x = [ 4 6 8**

**6 9 12**

**8 12 16]**

is [3x1][1x3] = [3x3]

**x \* x** is meaningless

**[1x3][1x3] = ?**

## Simple Plots

For a vector x with n entries, and a vector y with n entries, plot them in one of four ways:

**plot(x,y)** **loglog(x,y)**

**semilogx(x,y)** **semilogy(x,y)**

Plot more than one variable:

**plot(x, y1, x, y2)**

Add titles:

**title('This is the title')**

**xlabel('x')**

**ylabel('y')**

Add a legend:

**legend('first curve', 'second curve')**

## Add data to an existing plot

Issue the command:

**hold on**

and continue plotting. Further lines, symbols, etc. will be added to the existing figure. When done, say:

**hold off**

## Dress up your plot

Plot in different colors using:

**plot(x, y, 'r', x, y2, 'b')**

Use different markers

**plot(x, y, 'or', x, y2, '\*b')**

Use different line options.

**plot(x, y, '-r', x, y2, ':b')**

To get both the symbols and the lines, use both line styles and markers.

**plot(x, y, '-or', x, y2, '\*:b')**

Change the axis:

**axis([xmin xmax ymin ymax])** limits the plot to  $xmin \leq x \leq xmax$ ,  $ymin \leq y \leq ymax$ ,

regardless of the data.

**xlim([xmin xmax])** limits the plot to  $xmin \leq x \leq xmax$ , regardless of the data.

**ylim([ymin ymax])** limits the plot to  $ymin \leq y \leq ymax$ .

**help axis** for more information.

## Multiple plots on the same screen: subplot

2 x 3 array

**subplot(2,3,1)**

and the next plot is put into the first plot.

Subsequent plots are identified by the last number, and the plots are numbered from

left to right and top to bottom.

## 3D plots

To plot a function z(x,y), create an x-y grid (rectangular), evaluate the function at each grid point, and plot.

Create the x grid: **x = 0:0.05:1**

Create the y grid: **y = 1:0.2:3**

Create the combined mesh:

**[X,Y] = meshgrid(x,y)**

Evaluate the function: **Z = fn(x,y)**, where fn is an expression or function; **Z = X.\*X + Y.\*Y**

Plot the 3D plot: **mesh(X,Y,Z)**

or color in the surface: **surf(X,Y,Z)**

or do contours: **contour(X,Y,Z,20)** in 2D and

**contour3(X,Y,Z,20)** in 3D.

## Evaluate an integral

Define a set of points in the interval to be integrated, as the vector `x`. Then evaluate the function at each of these points. Calculate the integral using the trapezoid rule.

```
x = [0:0.1:2]
y = x.*x
area = trapz(x,y)
```

## Spline interpolation

Make a cubic B-spline pass through all the data points:

```
[ x(i), y(i), i = 1,...,m]
help spline gives this:
x = 0:10; y = sin(x);
xx = 0:0.25:10;
yy = spline(x,y,xx);
plot(x,y,'o',xx,yy)
```

## Loops

```
for i=1:10
    y(i) = ...
end
If want to stop before end:
for i=1:10
    y(i) = ...
    if y(i)>25 break,end
end
```

## Debugging

Insert the command `keyboard`. Then examine the variables

```
K> disp(x)
```

Keyboard session stops when you press return. If `keyboard` is in a function, use `dbup` to view variables in command workspace and `dbdown` to return to function workspace.

## Timing

```
turn on clock - tic
turn off clock - toc
elapsed_time = 2.34
```

## Solve Algebraic Equations

Solve  $f_i(\{y_j\}) = 0, i = 1, \dots, n$ , for the vector  $\{y_j\}$ . Set the initial guess of the solution, call `fsolve` and name the function which evaluates  $f_i$ .

```
p0 = [0 0]
z = fsolve('prob2',p0)
z =
    -1.4456 -2.4122
```

If this program does not work well, try making up an initial value problem and solving it using implicit methods, integrating to a long time.

$$\frac{dy_i}{dt} = f_i(\{y_j\}) \quad y_i(0) = \text{initial guess}$$

## Interpolate data, evaluate the polynomial, and plot the result.

For data points  $x(i), z(i), i = 1, \dots, m$ , fit a polynomial of degree  $n$  in the least squares sense. `p = polyfit(x,z,n)`

$$y = p_1 x^n + p_2 x^{n-1} + \dots + p_n x + p_{n+1}$$

The function `v = polyval(p,w)` evaluates the polynomial `p` at the `w` points (`w` can be a vector). Plot the data points (`y`) and the polynomial (`v`)

```
w = [0:1:10] v = polyval(p,w)
plot(x,y,'ro',w,v,'b-')
```

## Conditional Statements

```
if (condition1)
    ...
else
    ...
end
if (condition1)
    ...
elseif (condition2)
    ...
else
    ...
end
switch num
    case 1 % execute when num = 1
        y = ...
    case 2 % execute when num = 2
        y = ...
    otherwise % execute when num is
        neither 1 or 2
        disp('there is an error')
end
```

## Integrating ordinary differential equations that are initial value problems

Set the initial conditions, the time span for the integration, and the name of the function being integrated.

```
y0 = [0 1 3]
tspan = [0 10]
[t , y] = ode45('rhs',tspan, y0)
If you want data output at specific times, use
tspan = [0:1:10] for data at t = 0, 1, 2, ..., 10.
```

## Plotting results from integration of partial differential equations using method of lines.

Result is the value of the solution at several fixed positions and all times.

```
tspan = [0 10], y0 = [...]
```

```
[ t , y] = ode45('rhs',tspan,y0)
plot(t,y)
```

This gives a plot of each variable  $y(t,i)$  versus time.  $y(t,i)$  is the solution at point  $x(i)$  and time  $t$ . To plot the solution versus  $x(i)$ , at various times: specify that you want the solution at a select number of times.

```
tspan = [0:1:10] % 11 different times
y0 = [...], [ t , y] = ode45('rhs',tspan,y0)
x = [...] % there are the same number of entries for x
as for y0
hold on
for i=1:11
    plot(x(:),y(i,:))
end
hold off
```

## In/Out

- Ask user for input

```
viscosity = input('What is the viscosity (Pa s)? ')
• Display in specified format
fprintf( '%5.3f %10.5fn',x,y)
2.345 234.56789
f = floating point, \n is carriage return
e = exponential format
5.3 means five characters, three after decimal pt.
• Write text file; open file, write it, close it.
fid = fopen( 'ChE Data','w') % w for write
fprintf(fid, '%5.3f %10.5fn',[x,y])
fclose(fid)
• Read text file; open file, read it, close it.
fid = fopen( 'ChE Data','r') % r for read
x = fscanf(fid, '%5.3f %10.5f')
fclose(fid)
```

## More Complicated Plots

Possible symbols:

.	point		
o	circle		
x	x-mark	The colors are in the default order for multiple plots, but white is not used).	<u>Line types</u>
+	plus		- solid
*	star	b blue	: dotted
s	square	g green	- . dashdot
d	diamond	r red	-- dashed
v	triangle (down)	c cyan	
^	triangle (up)	m magenta	
<	triangle (left)	y yellow	
>	triangle (right)	k black	
p	pentagram	w white	
h	hexagram		

## Use Greek letters and symbols in the text.

(These are T<sub>E</sub>X commands.)

α \alpha	∞ \infty	\rm normal
β \beta	≥ \geq	\bf bold
γ \gamma	≤ \leq	\it italic
.	.	∂ \partial
ω \omega	± \pm	
Γ \Gamma	ℜ \Re	
Δ \Delta	ℑ \Im	
Θ \Theta		
.	.	
Ω \Omega		