

BASIC INPUT		
Declare a variable:	<code>a=5;</code>	<code>...</code> Continue input on next line
Declare a matrix:	<code>A = [1 2 3; 4 5 6];</code>	<code>;</code> Suppress output

BASIC COMMANDS & TOOLS			
<code>help fn</code>	Invoke help on function <i>fn</i>	<code>format short</code>	Toggle output detail
<code>doc fn</code>		<code>format long</code>	
<code>clc</code>	Clear command window	<code>tic</code>	Start timer
		<code>toc</code>	Stop timer

COMPARISON OPERATORS		MATHEMATICAL OPERATORS	
<code>==</code> equal to		<b>ELEMENTWISE OPERATORS</b>	<b>LINEAR-ALGEBRAIC OPERATORS</b>
<code>~=</code> not equal to		<code>+</code> addition	
<code>&gt;</code> greater than		<code>-</code> subtraction	
<code>&lt;</code> less than		<code>.*</code> multiplication	<code>*</code> multiplication
<code>&gt;=</code> greater than or equal to		<code>./</code> right division	<code>/</code> right division $B/A=B*\text{inv}(A)$
<code>&lt;=</code> less than or equal to		<code>.\</code> left division	<code>\</code> left division $A\backslash B=\text{inv}(A)*B$
LOGICAL OPERATORS		<code>+</code> unary plus (positive)	
<code>&amp;&amp;</code> and		<code>-</code> unary minus (negative)	
<code>  </code> or		<code>:</code> range	
<code>~</code> unary not		<code>.^</code> exponentiation	<code>^</code> exponentiation
<code>xor</code> exclusive or (either)		<code>.'</code> transpose	<code>'</code> complex conjugate transpose

ARRAY FUNCTIONS			
CREATION		MANIPULATION	
<code>eye(n)</code>	$n \times n$ identity matrix	<code>cat(dim,A,B)</code>	Concatenate arrays <i>A</i> and <i>B</i> along dimension <i>dim</i> .
<code>eye(m,n)</code>	$m \times n$ identity matrix	<code>horzcat(A,B)</code>	Concatenate arrays horizontally
<code>ones(...)</code>	Matrix of 1s	<code>vertcat(A,B)</code>	Concatenate arrays vertically
<code>rand(...)</code>	Random matrix [0,1)	<code>reshape(A,x,y)</code>	Arrange <i>A</i> into new shape <i>xy</i>
<code>zeros(...)</code>	Matrix of 0s	<b>MATHEMATICAL</b>	
<code>linspace(a,b,n)</code>	Return linearly-spaced vector of <i>n</i> points between values <i>a</i> and <i>b</i>	<code>inv(A)</code>	Invert matrix <i>A</i>
<code>meshgrid(xv,yv)</code>	Return 2 grids based on <i>xv</i> , <i>yv</i>	<code>linsolve(A,b)</code>	Return solution of $A*x=b$
<code>a:b</code>	Span range from <i>a</i> to <i>b</i>	<code>eig(A)</code>	Return eigenvalues of <i>A</i>
<code>a:d:b</code>	Span from <i>a</i> to <i>b</i> in steps of <i>d</i>		

PROGRAMMING CONSTRUCTS			
<code>for index = values</code> <code>statements</code> <code>end</code>	Execute code a specific number of times.  <i>values</i> may be a range or columns of an array.	<code>if expr</code> <code>statements</code> <code>elseif expr</code> <code>statements</code> <code>else</code> <code>statements</code> <code>end</code>	Execute alternate blocks of code based on logical expressions <i>expr</i> .
<code>while expr</code> <code>statements</code> <code>end</code>	Execute statements repeatedly while <i>expr</i> holds true.		
<code>break</code>	Terminate for or while loop	<code>function</code>	Declare new function (type "help function")
<code>continue</code>	Jump to next iteration of for or while loop	<code>return</code>	Return control to invoking function

## TRIGONOMETRIC & SPECIAL FUNCTIONS

<code>sin(x)</code>	Sine	<code>sinh(x)</code>	Hyp. sine	<code>sqrt(x)</code>	Square root
<code>cos(x)</code>	Cosine	<code>cosh(x)</code>	Hyp. cosine	<code>nthroot(x)</code>	Real $n$ th root of $x$
<code>tan(x)</code>	Tangent	<code>tanh(x)</code>	Hyp. tangent	<code>besselj(nu,x)</code>	Bessel function of the first kind of order $nu$
<code>exp(x)</code>	Exponential	<code>log(x)</code>	Natural logarithm (ln)	<code>erf(x)</code>	Error function
<code>expm1(x)</code>	$\exp(x)-1$ (accurate for small $x$ )	<code>log10(x)</code>	Logarithm base 10	<code>erfc(x)</code>	Complementary error function
<code>abs(x)</code>	Absolute value	<code>log1p(x)</code>	$\log(1+x)$ (accurate for small $x$ )	<code>gamma(x)</code>	$\Gamma$ function
(append "d" after any trigonometric name to yield result in degrees)				<code>i</code>	Imaginary unit

## SYMBOLIC COMMANDS

<code>sym('x')</code>	Declare vars $x \dots$ as sym objs	<code>int(f,x)</code>	Indefinite integral of $f$ w.r.t. $x$
<code>syms x ...</code>		<code>int(f,x,a,b)</code>	Definite integral of $f$ w.r.t. $x$ over range $a$ to $b$
<code>sym('#')</code>	Convert number $\#$ to sym objs		
<code>sym('expr')</code>	Convert expression $expr$ to sym obj		
<code>symvar(g)</code>	List sym vars in $g$	<code>diff(f,x)</code>	Derivative of $f$ w.r.t. $x$
<code>formula(f)</code>	Return formula for $f$	<code>diff(f,x,n)</code>	$n$ th derivative of $f$ w.r.t. $x$
<code>double(x)</code>	Calculate numerical value of $x$	<code>solve(f,x)</code>	Analytically solve equation $f$

Use the Symbolic Toolbox for anything more complicated than this.

## PLOTTING

<code>plot(x,y)</code>	Plot 2D data, $y$ versus $x$ .
<code>plot(x,y,...)</code>	"help plot" for full details.
<code>fplot(@fn,rng)</code>	Plot 2D function $fn$ over range $rng = [lo hi]$
<code>plot3(x,y,z)</code>	Plot 3D data as line
<code>surf(Z)</code>	Plot 3D data as surface
<code>surf(x,y,Z)</code>	
<code>ezsurf('fn')</code>	Plot 3D function as surface
<code>hold on</code>	Toggle persistence of previous plots
<code>hold off</code>	

## POLYNOMIALS

<code>poly(v)</code>	Return coefficients of polynomial with roots $v$
<code>roots(v)</code>	Return roots of a polynomial with coefficients $v$
<code>polyder(v)</code>	Return derivative coeffs of poly $v$
<code>polyval(v,x)</code>	Evaluate polynomial with coefficients $v$ at value $x$
<code>polyint(v)</code>	Return integral coeffs of poly $v$
<code>polyfit(x,y,n)</code>	Returns polynomial of $n$ th degree which fits data in $x,y$ best

## NUMBER THEORY

<code>factor(x)</code>	Return list of factors of $x$
<code>isprime(x)</code>	Return primeness of $x$
<code>primes(x)</code>	Return list of primes from zero up to and including $x$

## FITTING

<code>spline(x,y,xx)</code>	Return cubic-spline interpolation of $x,y$ at $xx$
<code>interp2(X,Y,V,Xq,Yq)</code>	Return 2D linear interpolation at $Xq,Yq$ based on values $V$ on grid $X,Y$

## ORDINARY DIFFERENTIAL EQUATIONS

<code>[T,Y] = ode45(odefun,tspan,y0,options)</code>	<b>T</b>	vector of time pts	<b>Y</b>	solution array
<b>odefun</b>	handle to right side of DEs	<b>tspan</b>	interval $[t_0, t_f]$	<b>y0</b> initial conditions
<b>options</b>	<code>options = odeset('name1',value1,'name2',value2,...)</code> <code>options = odeset('RelTol',1e-8,'InitialStep',1e-5)</code>			