## BASIC INPUT

| | | | |
|---|---|---|---|
| Declare a variable: | `a=5;` | `...` | Continue input on next line |
| Declare a matrix: | `A = [1 2 3; 4 5 6];` | `;` | Suppress output |

## BASIC COMMANDS & TOOLS

| | | | |
|---|---|---|---|
| `help` *fn*<br>`doc` *fn* | Invoke help on function *fn* | `format short`<br>`format long` | Toggle output detail |
| `clc` | Clear command window | `tic` | Start timer |
| | | `toc` | Stop timer |

## COMPARISON OPERATORS

| | |
|---|---|
| `==` | equal to |
| `~=` | not equal to |
| `>` | greater than |
| `<` | less than |
| `>=` | greater than or equal to |
| `<=` | less than or equal to |

## LOGICAL OPERATORS

| | |
|---|---|
| `&&` | and |
| `\|\|` | or |
| `~` | unary not |
| `xor` | exclusive or (either) |

## MATHEMATICAL OPERATORS

| ELEMENTWISE | | LINEAR-ALGEBRAIC | |
|---|---|---|---|
| `+` | addition | | |
| `-` | subtraction | | |
| `.*` | multiplication | `*` | multiplication |
| `./` | right division | `/` | right division `B/A=B*inv(A)` |
| `.\` | left division | `\` | left division `A\B=inv(A)*B` |
| `+` | unary plus (positive) | | |
| `-` | unary minus (negative) | | |
| `:` | range | | |
| `.^` | exponentiation | `^` | exponentiation |
| `.'` | transpose | `'` | complex conjugate transpose |

## ARRAY FUNCTIONS

### CREATION

| | |
|---|---|
| `eye(`*n*`)` | *n*×*n* identity matrix |
| `eye(`*m*`,`*n*`)` | *m*×*n* identity matrix |
| `ones(…)` | Matrix of 1s |
| `rand(…)` | Random matrix [0,1) |
| `zeros(…)` | Matrix of 0s |
| `linspace(`*a*`,`*b*`,`*n*`)` | Return linearly-spaced vector of *n* points between values *a* and *b* |
| `meshgrid(`*xv*`,`*yv*`)` | Return 2 grids based on *xv*, *yv* |
| *a*`:`*b* | Span range from *a* to *b* |
| *a*`:`*d*`:`*b* | Span from *a* to *b* in steps of *d* |

### MANIPULATION

| | |
|---|---|
| `cat(`*dim*`,`*A*`,`*B*`)` | Concatenate arrays *A* and *B* along dimension *dim*. |
| `horzcat(`*A*`,`*B*`)` | Concatenate arrays horizontally |
| `vertcat(`*A*`,`*B*`)` | Concatenate arrays vertically |
| `reshape(`*A*`,`*x*`,`*y*`)` | Arrange *A* into new shape *x*×*y* |

### MATHEMATICAL

| | |
|---|---|
| `inv(`*A*`)` | Invert matrix *A* |
| `linsolve(`*A*`,`*b*`)` | Return solution of *A***x*=*b* |
| `eig(`*A*`)` | Return eigenvalues of *A* |

## PROGRAMMING CONSTRUCTS

```
for index = values
    statements
end
```

Execute code a specific number of times.

*values* may be a range or columns of an array.

```
while expr
    statements
end
```

Execute statements repeatedly while *expr* holds true.

```
if expr
    statements
elseif expr
    statements
else
    statements
end
```

Execute alternate blocks of code based on logical expressions *expr*.

| | | | |
|---|---|---|---|
| `break` | Terminate for or while loop | `function` | Declare new function (type “help function”) |
| `continue` | Jump to next iteration of for or while loop | `return` | Return control to invoking function |

## TRIGONOMETRIC & SPECIAL FUNCTIONS

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

## SYMBOLIC COMMANDS

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

Use the Symbolic Toolbox for anything more complicated than this.

## PLOTTING

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

## POLYNOMIALS

| | |
|---|---|
| `poly(v)` | Return coefficients of polynomial with roots `v` |
| `roots(v)` | Return roots of a polynomial with coefficients `v` |
| `polyder(v)` | Return derivative coefs of poly `v` |
| `polyval(v,x)` | Evaluate polynomial with coefficients v at value x |
| `polyint(v)` | Return integral coefs of poly `v` |
| `polyfit(x,y,n)` | Returns polynomial of `n`th degree which fits data in `x`,`y` best |

## NUMBER THEORY

| | |
|---|---|
| `factor(x)` | Return list of factors of `x` |
| `isprime(x)` | Return primeness of x |
| `primes(x)` | Return list of primes from zero up to and including `x` |

## FITTING

| | |
|---|---|
| `spline(x,y,xx)` | Return cubic-spline interpolation of `x`,`y` at `xx` |
| `interp2(X,Y,V,Xq,Yq)` | Return 2D linear interpolation at `Xq`,`Yq` based on values `V` on grid `X`,`Y` |

## ORDINARY DIFFERENTIAL EQUATIONS

`[T,Y] = ode45(odefun,tspan,y0,options)`

| | | | | | |
|---|---|---|---|---|---|
| `T` | vector of time pts | `Y` | solution array | | |
| `odefun` | handle to right side of DEs | `tspan` | interval `[t0, tf]` | `y0` | initial conditions |

*`options`*

```
options = odeset('name1',value1,'name2',value2,...)
options = odeset('RelTol',1e-8,'InitialStep',1e-5)
```