MATLAB Reference Card

Operators and special characters.	
+	Plus; addition operator.
-	Minus; subtraction operator.
*	Scalar and matrix multiplication operator.
.*	Array (element-by-element) multiplication operator.
^	Scalar and matrix exponentiation operator.
.^	Array (element-by-element) exponentiation operator.
\	Left-division operator.
/	Right-division operator.
.\	Array (element-by-element) left-division operator
./	Array (element-by-element) right-division operator.
:	Colon; generates regularly spaced elements and represents an entire row or column.
( )	Parentheses; encloses function arguments and array indices; overrides precedence.
[ ]	Brackets; encloses array elements.
{ }	Braces; encloses cell elements.
.	Decimal point.
. . .	Ellipsis; line-continuation operator.
,	Comma; separates statements, and elements in a row of an array.
;	Semicolon; separates columns in an array, and suppresses display.
%	Percent sign; designates a comment, and specifies formatting.
'	Quote sign and transpose operator.
.'	Non-conjugated transpose operator.
=	Assignment (replacement) operator.

Logical and relational operators.	
==	Relational operator; equal to.
~=	Relational operator, not equal to.
<	Relational operator, less than.
<=	Relational operator, less than or equal to.
>	Relational operator, greater than.
>=	Relational operator, greater than or equal to.
&	Logical operator, AND.
|	Logical operator, OR.
~	Logical operator, NOT.

Order of precedence.	
Highest	Parentheses, evaluated starting with the innermost pair.
	Transpose and exponentiation, evaluated left to right.
	Unary plus or minus and logical NOT (~), evaluated left to right.
	Multiplication and division, evaluated from left to right.

	Addition and subtraction, evaluated from left to right.
	Colon operator (:).
	Relational operators, evaluated from left to right.
	Logical AND (&), evaluated left to right.
Lowest	Logical OR (|), evaluated left to right.

Special variables and constants.	
`ans`	Most recent answer.
`eps`	Accuracy of floating point precision.
`i,j`	The imaginary unit (square root of -1).
`Inf`	Infinity.
`NaN`	Undefined numerical result (not a number).
`pi`	The number π.

Commands for managing a session.	
`clc`	Clears Command window.
`clear`	Removes variables from memory.
`doc`	Displays documentation.
`exist`	Checks for existence of file or variable.
`global`	Declares variables to be global.
`help`	Displays help text in the Command window.
`helpwin`	Displays help text in the Help Browser.
`lookfor`	Searches help entries for a keyword.
`quit`	Stops MATLAB.
`who`	Lists current variables.
`whos`	Lists current variables (long display).

System and file commands.	
`cd`	Changes current directory.
`date`	Displays current date.
`delete`	Deletes a file.
`diary`	Switches on/off diary file recording.
`dir`	Lists all files in current directory.
`get`	Returns diary status or filename.
`load`	Loads workspace variables from a file.
`path`	Displays search path.
`pwd`	Displays current directory.
`save`	Saves workspace to a file.
`type`	Displays contents of a file.
`what`	Lists all MATLAB files.
`wk1read`	Reads `.wk1` spreadsheet file.
`xlsread`	Reads `.xls` spreadsheet file.

Input/output commands.	
`disp`	Displays contents of an array or string.

`dlmwrite`	Writes formatted data to an ASCII file.
`format`	Controls screen-display format.
`fprintf`	Performs formatted writes to screen or file.
`input`	Displays prompts and waits for output.
`menu`	Displays a menu of choices
`;`	Suppresses screen printing.

Numeric display formats	
`format short`	Four decimal digits (default)
`format long`	16 decimal digits.
`format short e`	Five digits plus exponent.
`format long e`	16 digits plus exponents.
`format bank`	Two decimal digits.
`format +`	Positive, negative, or zero.
`format rat`	Rational approximation.
`format compact`	Suppresses some line feeds.
`format loose`	Resets to less compact display mode.

Array functions	
`cat`	Concatenates arrays.
`find`	Finds indices of nonzero elements.
`length`	Computes number of elements.
`linspace`	Creates regularly spaced vector.
`logspace`	Creates logarithmically spaced vector.
`max`	Returns largest element.
`min`	Returns smallest element.
`size`	Computes array size.
`sort`	Sorts each column.
`sum`	Sums each column.

Special matrices	
`eye`	Creates an identity matrix.
`ones`	Creates an array of ones.
`zeros`	Creates an array of zeros.

Matrix functions for solving linear equations	
`det`	Computes determinant of an array.
`inv`	Computes inverse of a matrix.
`pinv`	Computes pseudo-inverse of a matrix.
`rank`	Computes rank of a matrix.
`rref`	Computes reduced row echelon from.

Exponential and logarithmic functions	
`exp(x)`	Exponential; e^x.
`log(x)`	Natural logarithm; ln x.

`log10(x)`	Common (base ten) logarithm; $\log_{10}x$.
`sqrt(x)`	Square root of x.

Complex functions	
`abs(x)`	Absolute value of x.
`angle(x)`	Angle of a complex number x.
`conj(x)`	Complex conjugate of x.
`imag(x)`	Imaginary part of a complex number x.
`real(x)`	Real part of a complex number x.

Numeric functions	
`ceil`	Rounds to the nearest integer toward ∞.
`fix`	Rounds to the nearest integer toward zero.
`floor`	Rounds to the nearest integer toward -∞.
`round`	Rounds toward the nearest integer.
`sign(x)`	Returns +1, 0, or -1, depending on sign of x.

Trigonometric functions	
`acos(x)`	Inverse cosine; $\cos^{-1}x$.
`acot(x)`	Inverse cotangent; $\text{sec}^{-1}x$.
`acsc(x)`	Inverse cosecant; $\csc^{-1}x$.
`asec(x)`	Inverse secant; $\sec^{-1}x$.
`asin(x)`	Inverse sine; $\sin^{-1}x$.
`atan(x)`	Inverse tangent; $\tan^{-1}x$.
`atan2(y,x)`	Four-quadrant inverse tangent of y/x.
`cos(x)`	Cosine; cos x.
`cot(x)`	Cotangent; cot x.
`csc(x)`	Cosecant; csc x.
`sec(x)`	Secant; sec x.
`sin(x)`	Sine; sin x.
`tan(x)`	Tangent; tan x.

Hyperbolic functions	
`acosh(x)`	Inverse hyperbolic cosine; $\text{cosh}^{-1}x$.
`acoth(x)`	Inverse hyperbolic cotangent; $\text{coth}^{-1}x$.
`acsch(x)`	Inverse hyperbolic cosecant; $\text{csch}^{-1}x$.
`asech(x)`	Inverse hyperbolic secant; $\text{sech}^{-1}x$.
`asinh(x)`	Inverse hyperbolic sine; $\text{sinh}^{-1}x$.
`atanh(x)`	Inverse hyperbolic tangent; $\text{tanh}^{-1}x$.
`cosh(x)`	Hyperbolic cosine; cosh x.
`coth(x)`	Hyperbolic cotangent; cosh x / sinh x.
`csch(x)`	Hyperbolic cosecant; 1 / sinh x.
`sech(x)`	Hyperbolic secant; 1 / cosh x.
`sinh(x)`	Hyperbolic sine; sinh x.
`tanh(x)`	Hyperbolic tangent; sinh x / cosh x.

Polynomial functions	
`conv`	Computes product (convolution) of two polynomials.
`deconv`	Computes ratio (deconvolution) of polynomials.
`eig`	Computes the eigenvalues of a matrix.
`poly`	Computes polynomial from roots.
`polyfit`	Fits a polynomial to data.
`polyval`	Evaluates polynomial.
`roots`	Computes polynomial roots.

String functions	
`findstr`	Finds occurrences of a string.
`strcmp`	Compares strings.

Logical Functions	
`any`	True if any elements are nonzero.
`all`	True if all elements are nonzero.
`find`	Finds indices of nonzero elements.
`finite`	True if elements are finite.
`isnan`	True if elements are undefined.
`isempty`	True if elements are infinite.
`isreal`	True if elements are real.
`xor`	Exclusive OR.

Miscellaneous mathematical functions	
`cross`	Computes cross products.
`dot`	Computes dot products.
`function`	Creates a user-defined function.

Cell array functions	
`cell`	Creates cell array.
`celldisp`	Displays cell array.
`cellplot`	Displays graphical representation of cell array.
`num2cell`	Converts numeric array to cell array.
`deal`	Matches input and output lists.
`iscell`	Identifies cell array.

Structure functions	
`fieldnames`	Returns field names in a structured array.
`getfield`	Returns field contents of a structure array.
`isfield`	Identifies a structure field array.
`isstruct`	Identifies a structure array.
`rmfield`	Removes a field from a structure array.
`setfield`	Sets contents of a field.
`struct`	Creates structure array.

Basic xy plotting commands	
`axis`	Sets axis limits.
`fplot`	Intelligent plotting of functions.
`ginput`	Reads coordinates of the cursor position.
`grid`	Displays gridlines.
`plot`	Generates x-y plot.
`print`	Prints plot or saves to a file.
`title`	Puts text at top of plot.
`xlabel`	Adds text label to x-axis.
`ylabel`	Adds text label to y-axis.

Plot-enhancement commands	
`axes`	Creates axes objects.
`gtext`	Enables label placement by mouse.
`hold`	Freezes current plot.
`legend`	Legend placement by mouse.
`refresh`	Redraws current figure window.
`set`	Specifies properties of objects such as axes.
`subplot`	Creates plots in sub-windows.
`text`	Places string in figure.

Specialized plot functions	
`bar`	Creates bar chart.
`loglog`	Creates log-log plot.
`plotyy`	Enables plotting on left and right axes.
`polar`	Creates polar plot.
`quiver`	Plot velocity vectors as arrows.
`semilogx`	Creates semilog plot (logarithmic abscissa).
`semilogy`	Creates semilog plot (logarithmic ordinate).
`stairs`	Creates stair plot.
`stem`	Creates stem plot.

Three-dimensional plotting functions	
`contour`	Creates contour plot.
`mesh`	Creates 3D mesh surface plot.
`meshc`	Same as `mesh` with contour plot underneath.
`meshz`	Same as `mesh` with vertical lines underneath.
`plot3`	Creates 3D plots from lines and points.
`surf`	Creates shaded 3D mesh surface plot.
`surfc`	Same as `surf` with contour plot underneath.
`meshgrid`	Creates rectangular grid.
`waterfall`	Same as `mesh` with mesh lines in one direction.
`zlabel`	Adds text label to z-axis.

Program flow control	
`break`	Terminates execution of a loop.
`case`	Provides alternate execution path within `switch` structure.
`continue`	Passes control to the next iteration of a `for` or `while` loop.
`else`	Delineates alternate block of statements.
`elseif`	Conditionally executes statements.
`end`	Terminates `for`, `while`, and `if` statements.
`for`	Repeats statements a specific number of times.
`if`	Executes statements conditionally.
`otherwise`	Provides optional control within a `switch` structure.
`switch`	Directs program execution by comparing input with `case` expressions.
`while`	Repeats statements an indefinite number of times.

Optimization and root-finding functions	
`fminbnd`	Finds the minimum of a function of one variable.
`fminsearch`	Finds the minimum of a multivariable function.
`fminunc`	Finds the unconstrained minimum of a multivariable function
`fmincon`	Finds the minimum of a multivariable function with constraints.
`fzero`	Finds the zero of a function.
`optimset`	Creates optimization options structure.

Histogram functions	
`bar`	Creates a bar chart.
`hist`	Aggregates the data into bins.

Statistical functions	
`cumsum`	Computes the cumulative sum across a row.
`erf(x)`	Computes the error function, erf(x).
`mean`	Computes the mean.
`median`	Computes the median.
`std`	Computes the standard deviation.

Random number functions	
`rand`	Generates uniformly distributed random numbers between 0 and 1; sets and retrieves the state.
`randn`	Generates normally distributed random numbers; sets and retrieves the state.
`randperm`	Generates random permutations of integers.

Polynomial functions	
`poly`	Computes the coefficients of a polynomial and its roots.
`polyfit`	Fits a polynomial to data.
`polyval`	Evaluates a polynomial and generates error estimates.

`roots`	Computes the roots of a polynomial from its coefficients.

Interpolation functions	
`interp1`	Linear and cubic-spline interpolation of a uni-variable function.
`interp2`	Linear interpolation of a function of two variables.
`spline`	Cubic-spline interpolation.
`unmkpp`	Computes the coefficients of cubic-spline polynomials.

Numerical differentiation and integration functions	
`diff(x)`	Computes the differences between adjacent elements in a vector x
`polyder`	Differentiates a polynomial, a polynomial product, or a polynomial quotient.
`quad`	Numerical integration with adaptive Simpson's rule.
`quad1`	Numerical integration with Lobatto quadrature.
`trapz`	Numerical integration with the trapezoidal rule.

ODE solvers	
`ode23`	Nonstiff, low-order solver.
`ode45`	Nonstiff, medium-order solver.
`ode113`	Nonstiff, variable-order solver.
`ode23s`	Stiff, low-order solver.
`ode23t`	Moderately-stiff, trapezoidal rule solver.
`ode23tb`	Stiff, low-order solver.
`ode15s`	Stiff, variable-order solver.
`odeset`	Creates integrator options structure for ODE solvers.

Predefined input functions	
`gensig`	Generates a periodic sine, square, or pulse input.
`sawtooth`	Generates a periodic sawtooth input.
`square`	Generates a square wave input.
`stepfun`	Generates a step function input.

Functions for creating and evaluating symbolic expressions	
`class`	Returns the class of an expression.
`digits`	Sets the number of decimal digits used to do variable precision arithmetic.
`double`	Converts an expression to numeric form.
`ezplot`	Generates a plot of a symbolic expression.
`findsym`	Finds the symbolic variables in a symbolic expression.
`numden`	Returns the numerator and denominator of an expression.
`sym`	Creates a symbolic variable.
`syms`	Creates one or more symbolic variables.
`vpa`	Sets the number of digits used to evaluate expressions.

Functions for manipulating symbolic expressions	
collect	Collects coefficients of like powers in an expression
expand	Expands an expression by carrying out powers.
factor	Factors an expression.
poly2sym	Converts a polynomial coefficient vector to a symbolic polynomial.
pretty	Displays an expression in a form that resembles typeset mathematics.
simple	Searches for the shortest form of an expression.
simplify	Simplifies an expression using Maple's simplification rules.
subs	Substitutes variables or expressions.
sym2poly	Converts an expression to a polynomial coefficient vector.

Symbolic solution of equations	
solve	Solves symbolic equations.
dsolve	Returns the symbolic solution of a differential equation or set of equations.

Symbolic calculus functions	
diff	Returns the derivative of an expression.
dirac	Dirac delta function (unit impulse).
heaviside	Heaviside function (unit step).
int	Returns the integral of an expression.
limit	Returns the limit of an expression.
symsum	Returns the symbolic summation of an expression.
taylor	Returns the Taylor series of a function.

Symbolic linear algebra functions	
det	Returns the determinant of a matrix.
eig	Returns the eigenvalues (characteristic values) of a matrix.
inv	Returns the inverse of a matrix.
poly	Returns the characteristic polynomial of a matrix.

Laplace transform functions	
ilaplace	Returns the inverse Laplace transform.
laplace	Returns the Laplace transform.

Sample Program (Lecture02a.m)

```
function Lecture02a()
% Lecture02a - this program simulates UNSsteady 1d heat
%    CONDduction with (unsteady) temperatures prescribed at the boundaries
%
% written by John Dannenhoffer

clear

% get all inputs
xmax = input('Enter xmax: ');              % maximum depth                   (m)
I    = input('Enter I:    ');              % number of points in i direction (-)
tmax = input('Enter tmax: ');              % maximum time                    (d)
N    = input('Enter N:    ');              % maximum number of time steps    (-)

alfa = 0.52 * 3600 * 24 / 2050 / 1840;     % diffusivity                     (m2/d)
fprintf(1, 'Enter alfa: %f\n', alfa);

dx   = xmax / I;                           % distance between nodes          (m)
dt   = tmax / N;                           % time step                       (d)

II   = I + 1;
NN   = N + 1;

Fo   = alfa * dt / (dx^2);                 % Fourier number

% set up space array and initial temperatures
for ii = 1 : II
    nn = 1;
    x(ii   ) = (ii-1) * dx;                % spatial position                (m)
    t(   nn) = 0;                          % initial time                    (d)
    T(ii,nn) = 20;                         % initial temperature             (C)
end % for ii

% step through remaining times
for nn = 2 : NN
    t(nn) = (nn-1) * dt;

    % set up the coefficients...

    % ...left boundary condition
    atri(1) = 1;
    btri(1) = 0;
    ctri(1) = 0;
    if (t(nn) < 60)                        % left boundary temperature       (C)
        dtri(1) = -15;
    else
        dtri(1) = +20;
    end % if

    % ...interior (governing equation)
    for ii = 2 : II-1
        atri(ii) = 1 + 2 * Fo;
        btri(ii) =       - Fo;
        ctri(ii) =       - Fo;
        dtri(ii) = T(ii,nn-1);
    end % for ii

    % ...right boundary condition
    atri(II) = 1;
    btri(II) = 0;
    ctri(II) = 0;
    dtri(II) = 20;                          % right boundary temperature     (C)

    % use the Thomas algorithm to solve the tridiagonal system
    Tnew = thomas(II, atri, btri, ctri, dtri);

    % put the "results" into the temperature array
    for ii = 1 : II
```

```
        T(ii,nn) = Tnew(ii);
    end % for ii
end % for nn

% find the minimum depth for which T > 0
mindepth = x(II);

for ii = II-1 : -1 : 1
    okay = 1;

    for nn = 1 : NN
        if (T(ii,nn) < 0)
            okay = 0;
        end % if
    end % for nn

    if (okay == 1)
        mindepth = x(ii);
    end % if
end % for ii

fprintf(1, '\nMinimum depth that pipe can be buried is approximately %f(m)\n', mindepth);

% plot a contour of the results
for nn = 1 : NN
    for ii = 1 : II
        tt(ii,nn) = t(nn);
        xx(ii,nn) = x(ii);
    end % for ii
end % for nn

[cs,h] = contourf(xx, tt, T); ...
    colorbar
    title('Contours of temperature (C)')
    xlabel('x (m)')
    ylabel('t (d)')

%---------------------------------------------------------------------

function x = thomas(n, a, b, c, d);
% THOMAS Solve a tridiagonal matrix using the Thomas algorithm
%
%  [a1  c1                 ] [x1]   [d1]
%  [b2  a2  c2             ] [x2]   [d2]
%  [    b3  a3  c3         ] [x3]   [d3]
%  [                       ] [. ] = [. ]
%  [             ...       ] [. ]   [. ]
%  [                       ] [. ]   [. ]
%  [                 bn  an ] [xn]   [dn]
%
% inputs:
%    n              size of matrix
%    a              array of      diagonal elements
%    b              array of   subdiagonal elements
%    c              array of superdiagonal elements
%    d              array of right-hand sides
% output:
%    x              array of solutions

% Forward elimination

p(1) = - c(1) / a(1);
q(1) =   d(1) / a(1);

for i = 2 : n
    p(i) =  - b(i)                  / (a(i) + c(i) * p(i-1));
    q(i) = (  d(i) - c(i) * q(i-1)) / (a(i) + c(i) * p(i-1));
end % for i

% Back substitution
```

```
x(n) = q(n);

for i = n-1 : -1 : 1
    x(i) = p(i) * x(i+1) + q(i);
end % for i%
```