Mastering MATLAB Toolbox Reference version 6.0

The *Mastering MATLAB Toolbox* is a collection of over 250 M-files written by the authors of *Mastering MATLAB 6* (ISBN 0-13-019468-9). The functions in the *Toolbox* range from commonly used utilities, to numerical analysis and optimization functions, to powerful high-level plotting routines, to polished graphical user interfaces. This is the third version of the *Mastering MATLAB Toolbox*, with the previous versions being associated with the two prior editions of the *Mastering MATLAB* text.

As with the prior versions of the *Toolbox*, this version is available free of charge from the *Mastering MATLAB* web site (see the URL shown in the header). This version of the *Toolbox* is distributed in P-code format, which is an encrypted binary format. As a result, no on-line help is available for the functions. That is, `>> help mmname` displays no help text for the function *mmname*. In addition, it is not possible to see the MATLAB code that implements the functions. If you wish to have access to the Toolbox M-files and have on-line help, you must register the *Toolbox* by following the directions given at the *Mastering MATLAB* web site. If you registered a prior version of the *Toolbox*, the web site provides options and directions for upgrading to the latest version of the *Toolbox*.

This reference manual is a complete rewrite of the manual that accompanied the prior edition of the *Toolbox*. The prior edition was originally written to be the last chapter in the text *Mastering MATLAB 5*. When the decision was made to remove it from the text, it became the downloadable reference manual.

> Not all features of all *Toolbox* functions are covered in the reference manual below. Many functions have alternative calling syntax and offer more sophisticated features than disclosed below. Complete help for all *Toolbox* functions can be obtained by registering the *Toolbox*.

Changes from the previous version:

This version of the *Toolbox* contains over 100 new M-files, including a rational polynomial object class named `mmrp`. In addition to the new M-files, the following M-files have been renamed, dramatically changed, deleted, or are obsolete but grandfathered.

Function	Status
`fsform`	Renamed as `fsformat`, now supports all forms.
`fsprod`	Now supports an arbitrary number of input arguments.
`fssum`	Now supports an arbitrary number of input arguments.
`fstable`	Bugs fixed and half-wave sine wave added.
`mm2dpfit`	Now allows weighted fitting.
`mmbrowse`	Obsolete, superseded by the *Workspace Browser*.
`mmcd`	Obsolete, superseded by the *Current Directory Browser*.
`mmcurve`	Renamed as `mmcurvelen`.
`mmismem`	Obsolete, `ismember` is now a MEX-file.
`mmisv5`	Obsolete, superseded by `mmisver`.
`mmload`	Obsolete, superseded by the *Import Wizard*.
`mmplot2`	Renamed as `mmplotyy`.
`mmpause`	Deleted, `pause` now supported fractional time.
`mmsave`	Obsolete.

Function	Status
mmscolor	Obsolete.
mmsetptr	Renamed as mmputptr.
mmspdata	Deleted, superseded by mmspget.
mmspint	Now supports an integration constant.
mmsview	Obsolete, superseded by *Figure* window tools.
mmzoom	Now performs a picture-in-a-picture zoom.

New functions:

Of the 100 or so new functions in the Toolbox, the following list documents the most important additions.

Function	Description
mmkeep	Clear variables except those listed.
mmakeidx	Make index vector from limits.
mmfindrc	Find first or last nonzero indices per row or column.
mmnumfun	Functions on numerical arrays.
mmrepeat	Repeat or count repeated elements in a vector.
mmsubdiv	Subdivide vector values.
mmwrap	Form matrix from circular shifted vector.
mmdiffsum	Differential sum of elements.
mmx	Expand singleton dimensions.
mmcellfun	Functions on cell array contents.
mmrmfield	Remove structure fields.
mmrnfield	Rename structure fields.
mmrofield	Reorder structure fields.
mmstructcat	Concatenate structures.
mmstructfun	Functions on structure fields.
mmv2struct	Pack/unpack variables to/from a structure.
mmstrtok	Find tokens in a string.
mmisver	Test for given version of MATLAB.
mmbuiltin	Built-in function names.
mmvarnames	M-file variable names.
mmatcmp	True if MAT-files are equal.
mmcount	Count occurrences of values in an array.
mmcurvex	Intersections of two curves.
mmlinefun	Functions on line segments.
mmsortcc	Sort vector into complex conjugate pairs.
mm2dp2p	2D polynomial to 1D polynomial.
mm2dpadd	2D polynomial addition.
mm2dpder	2D polynomial derivative.
mm2dpint	2D polynomial integral.
mmhermite	Cubic Hermite spline construction.
mmspbreak	Modify spline breakpoints.
mmspinfl	Spline inflection points.

Function	Description
mmspmath	Mathematics on splines.
mmsppaste	Paste spline piecewise polynomials.
fsarea	Area under a Fourier series.
fsdivide	Fourier series time division.
fsevenodd	Fourier series even and odd time parts.
fsinterp	Inverse interpolate Fourier series.
fsplot	Fourier series time plot.
fsstem	Fourier series stem plot.
mmgauss	Numerically evaluate cumulative integral.
mmquad	Numerically evaluate integral.
mmplotxx	Plot data with top and bottom X-axes.
mmfigpos	Position figure windows.
mmgetset	Get settable object properties.
mmgetundoc	Get undocumented object properties.
mmgrid	Custom axis grids.
mmxy	Show and get x-y coordinates using mouse.
mmclass	Object class existence.

Not included in the above list are the following new GUI functions.

Function	Description
mmpfit	Interactive polynomial curve fitting GUI.
mmspfit	Spline creation and manipulation GUI.
mmprobe3	Probe 3-D data using mouse.
mmstick	Set axis ticks using a GUI.

Function Reference:

The functions in the *Mastering MATLAB Toolbox* are described below based on where associated material appears in the text *Mastering MATLAB 6*. However, the functions created in Chapter 38 *Examples, Examples, Examples* appear in Chapters that discuss the underlying material.

Chapter 2 Basic Features

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
`mmdigit`	Round values to given significant digits.
`mmlog10`	Dissect decimal floating point numbers.
`mmmod`	Modulus integer count.
`mmpa`	Principal angle.
`mmquant`	Quantize values.
`mmsinc`	sin(*x*)/(*x*) function.

The function **`mmdigit`**`(X,N,B)` returns an output array the same size as `X` with its components rounded to `N` digits in base `B`. If `X` is complex, the real and imaginary components are rounded independently.

The function `[M,E]=`**`mmlog10`**`(X)` dissects the array `X` returning two arrays the same size as `X`. `M` contains the mantissa and `E` contains the integer exponent for each element in the input. Loosely speaking, this function separates numerical values into their scientific notation equivalents. The absolute value of the components of `M` are in the range $1 \le |M| < 10$.

The function **`mmmod`**`(X,Xmin,Xmax)` returns an array the same size as `X` where each component in `X` is wrapped so that the minimum is `Xmin` and the maximum is `Xmax`. For example, if

```
N = [-4 -3 -2 -1  0  1  2  3  4  5  6]
```

then

```
mmmod(N,3,7) = [ 6  7  3  4  5  6  7  3  4  5  6]
```

The function **`mmpa`** is the inverse of the MATLAB function `unwrap`. `mmpa(X)` returns the principle angles in the range $-\pi \le \theta < \pi$ associated with the elements of `X`.

The function **`mmquant`**`(X,N,Xmin,Xmax)` quantizes the values in `X` to `N` levels starting at `Xmin` and ending at `Xmax`.

Chapter 3 The MATLAB Desktop

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
`ed`	Shortcut for `edit`.
`hw`	Shortcut for `helpwin`.
`mmbytes`	Variable memory usage.
`mmkeep`	Clear variables or functions except for those listed.

Often the quickest way to open an existing M-file is to type `edit` *`filename`* at the MATLAB prompt. When this is done, the file *`filename`* can be anywhere on the MATLAB path, not just in the current directory. To simplify this further the function `ed` requires fewer keystrokes and allows more than one file to be opened simultaneously. Typing `ed filename1 filename2` … opens the MATLAB editor with all the listed files.

Like the function `ed`, the function `hw` is also a shortcut. `hw` *`topic`* is equivalent to `helpwin` *`topic`*.

As an alternative to the tools in the *Workspace* browser and the `clear` command, the function `mmkeep` clears all variables or functions except those listed as command arguments.

Chapter 5 Arrays and Array Operations

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
mmakeidx	Make index vector from limits.
mmdeal	Deal data into individual arguments.
mmfind	Find indices of a vector in a matrix.
mmfindrc	Find first or last nonzero indices per row or column.
mmlimit	Limit values between extremes.
mmnumfun	Functions on numerical arrays.
mmono	Test for monotonic vector.
mmrand	Uniformly distributed arrays.
mmrandn	Normally distributed arrays.
mmrepeat	Repeat or count repeated values in a vector.
mmshiftd	Shift or circularly shift matrix rows.
mmshiftr	Shift or circularly shift matrix columns.
mmsubdiv	Subdivide vector values.
mmtrim	Trim negligible array elements.
mmwrap	Form matrix from circular shifted vector.

The functions `mmakeidx`, `mmfindrc`, `mmrepeat`, `mmsubdiv`, and `mmwrap` are developed in Chapter 38 of the text and therefore require no further explanation.

The function **`mmdeal`** extends the capabilities of the function MATLAB function `deal`. For example, `[a,b,c,…]=mmdeal(S)` where `S` is a string array, returns individually deblanked rows in consecutive output arguments.

The function **`mmfind`**`(x,A)` returns the row (column) indices where the row (column) vector `x` appears in the array `A`.

The function **`mmlimit`**`(X,Xmin,Xmax)` returns an array the same size as `X` where each element of `X` is limited to the range of $Xmin \le X \le Xmax$.

The function `[A,B,C,...]=`**`MMNUMFUN`**`(FUN,X,Y,Z,...)` applies the function described by `FUN` to each of the remaining input arrays, returning arrays to corresponding output variables.

The function **`mmono`**`(x)` returns -2 if the vector x is strictly decreasing, -1 if it is decreasing, 1 if it is increasing, 2 if it is strictly increasing, and 0 if it is neither increasing nor decreasing.

The function **`mmtrim`**`(X)` trims (sets to zero) negligible elements in the array `X`. If `X` is complex, negligible real and imaginary parts are trimmed.

Chapter 6 Mulitdimensional Arrays

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
`mmdiffsum`	Differential sum of elements.
`mmx`	Expand singleton dimensions.

The above functions are developed in Chapter 38 of the text and therefore require no further explanation.

Chapter 7 Cell Arrays and Structures

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
mmcellfun	Functions on cell array contents.
mmcellstr	Create cell array of strings.
mmrmfield	Remove structure fields.
mmrnfield	Rename structure fields.
mmrofield	Reorder structure fields.
mmstructcat	Concatenate structures.
mmstructfun	Functions on contents of structure fields.
mmv2struct	Pack/unpack variables to/from a scalar structure

The function **MMCELLFUN**(FUN,C) applies the function described by FUN to each cell in the cell array C, returning a cell array the same size as C containing the function results.

The function **mmcellstr**(S) creates a cell array of strings from the input argument S, which can be a single string with embedded new line characters, vertical bars |, or a string array. When S is a string array, mmcellstr is approximately three times faster than the standard MATLAB function cellstr.

The function **mmstructfun**(FUN,S) applies the function described by FUN to each field in the structure S, returning a structure with the same field names containing the function results.

The functions mmrmfield, mmrnfield, mmrofield, mmstructcat, and mmv2struct are developed in Chapter 38 of the text and therefore require no further explanation.

Chapter 8 Character Strings

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
`mmfindstr`	Find first string in second string.
`mmisdigit`	True for digits in strings.
`mmonoff`	String ON/OFF to from logical conversion.
`mmstrcmpl`	Lexical string comparison.
`mmstrrep`	String replacement without overlaps.
`mmstrtok`	Find tokens in a string.
`mmstrtrim`	Trim leading and/or trailing white space from string.

The function **`mmfindstr`**`(S1,S2)` returns starting indices of nonoverlapping occurances of `S2` in `S1`. This is different than the MATLAB function `findstr`, which finds all occurences of the shorter string in the longer.

The function **`mmonoff`** converts the strings `'off'` and `'on'` to logical `0` and `1` respectively and vise versa.

The function **`MMSTRCMPL`**`(S1,S2)` compares the string `S2` to `S1` in the dictionary order sense. If `S2` appears before `S1`, `-1` is returned. If `S2` appears after `S1`, `+1` is returned.

The function **`MMSTRTOK`**`(S,D)` returns a cell array containing all tokens in the string `S` delimited by any of the characters in `D`. If `D` is empty or not given, white space is assumed. This function vectorizes the MATLAB function `strtok` to return all tokens in a single call.

The function **`mmstrtrim`** generalizes the operation of the MATLAB function `deblank`, which deletes trailing blanks from an input string. `mmstrtrim(S)` deletes both leading and trailing blanks; `mmstrtrim(S,'lead')` deletes just leading blanks; `mmstrtrim(S,'trail')` deletes just trailing blanks.

Chapter 9 Relational and Logical Operations

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
mmempty	Substitute value if empty.
mmisequal	True for elements equal within tolerance.
mmisflint	True for floating point integers.
mmisvect	True for vectors.
mmisver	True for given MATLAB version.

The function **mmempty**(A,B) returns A if A is not empty, otherwise B is returned.

The function **MMISEQUAL**(A,B,TOL) returns an array the same size as A or B containing logical True where the numerical arrays A and B are equal within a tolerance of +/- TOL and logical False where they are not.

The function **MMISVER**(N) for integer N returns True if running any MATLAB version N.n.m. For example, MMISVER(4) returns True if running any version 4.n.m. MMISVER(X) for noninteger X returns True if running MATLAB version X. For example MMISVER(5.3) returns True if running version 5.3 or 5.3.1. MMISVER(RXX) for string RXX denoting the release number, e.g. 'R12' returns TRUE if running the designated release number. X=MMISVER returns the N.n version of MATLAB ignoring any third digit. For example version 6.0.1 returns X=6.0. [N,n,m]=MMISVER returns the N.n.m version of MATLAB as separate integers. For example [N,n,m]=MMISVER returns N=5, n=2, m=1 for version 5.2.1. MMISVER('R') returns the numerical release number of MATLAB.

Chapter 11 Function M-files

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
`mmbuiltin`	Built-in function names.
`mmswap`	Swap two variables.
`mmvarnames`	M-file variable names.

The function `[B,M,X,C]=MMBUILTIN` returns cell arrays of strings containing the names of built-in functions in `B`, M-file functions in `M`, MEX-file functions in `X`, and classes in `C`. Only basic MATLAB information is returned.

The function **`mmswap`** is discussed in Chapter 11 in the text and therefore requires no further explanation.

The function **`mmvarnames`** parses an M-file and returns a cell array of strings containing the variable names found in the file.

Chapter 13 File and Directory Management

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
`mmatcmp`	True if MAT-file contents are equal.
`mmfiledate`	Get file modification date.

The function `MMATCMP('File1','File2')` returns logical True (1) if the contents of the two MAT files are equal. They are equal if they both contain the same variables and those variables contain the same values.

The function `MMFILEDATE File1 File2 ...` displays the file modification dates for the listed files. Blank lines are displayed for files that do not exist.

Chapter 14 Set, Bit, and Base Functions

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
`mmintersect`	Set intersection within tolerance.
`mmunique`	Set unique within tolerance.

The functions `mmintersect` and `mmunique` extend the MATLAB functions `intersect` and `unique` respectively to allow for set inclusion if within a specified tolerance.

Chapter 16 Matrix Algebra

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
`mmrwls`	Recursive weighted least squares.

The function `[x,P]=MMRWLS(y,A,W)` computes the weighted least squares solution `x` that minimizes `(y-Ax)'*diag(W)*(y-Ax)`. The weighting vector `W` must have a length equal to `y`, or be a scalar in which case all equations are given the scalar weight. `W(i)` is the weight given to the `i`th row of `Ax=y`. If requested, the matrix `P` contains information for optional future recursive calls.
`[xn,Pn]=MMRWLS(yn,An,Wn,x,P)` computes the recursive weighted least squares solution `xn`, given new equations `yn=An*x`, where `Wn` is the weighting vector associated with the new data, and `x` and `P` are the output from the previous function call. `Pn` is the updated `P` matrix.

Chapter 17 Data Analysis

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
mmax	Array maximum value.
mmcount	Count occurrences of values in an array.
mmcummax	Indices of cumulative maxima.
mmcurvelen	Length along a plane curve.
mmcurvex	Intersections of two curves.
mmin	Array minimum value.
mmlinefun	Functions on line segments and points on a plane.
mmpeaks	Find indices of relative extrema.
mmsort	Generalized 2D sorting.
mmsortcc	Sort vector into complex conjugate pairs.

The function `[Y,N]=MMCOUNT(A,S)` returns vectors `Y` and `N` where `Y` contains the sorted unique values in `A` and `N(i)` contains the number of occurances of `A(i)` in `S`.

The function `MMCUMMAX(X,'max')` returns the indices of cumulative maxima in `X(:)`. That is, an index value `i` is returned if `X(i)>X(1:i-1)`.

The function `MMCURVELEN(X,Y)` computes the length of the plane curve described by the data in `X` and `Y`.

The function `[X,Y]=MMCURVEX(H1,H2)` finds the intersection points of the two curves on the *x-y* plane identified by the line object handles `H1` and `H2`.

The function `mmlinefun` performs various geometrical tasks between line segments or between points on a plane and line segments.

The function `MMPEAKS(Y,'max')` return the indices where `Y(:)` has local maxima. `MMPEAKS(Y,'min')` return the indices where `Y(:)` has local minima.

The function `MMSORT(X,N)` sorts the 2D array `X` down its `N`-th column in ascending order returning an array where each column is ordered by the `N`-th column sort. `MMSORT(X,N,'col')`, `MMSORT(X,N,'col','ascend')` and `MMSORT(X,N,'ascend')` do the same thing. `MMSORT(X,N,'col','descend')` and `MMSORT(X,N,'descend')` sorts `X` down its `N`-th column in descending order. `MMSORT(X,N,'row')` sorts `X` across its `N`-th row in ascending order returning an array where each row is ordered by the `N`-th row sort. `MMSORT(X,N,'row','descend')` sorts `X` across its `N`-th row in descending order.

The function `MMSORTCC(X)` sorts the vector `X` by increasing real part. Complex numbers are sorted into complex conjugate pairs. Values sharing the same real part are sorted by increasing magnitude of their imaginary parts, with a+jb appearing before a-jb. This function is generally at least an order of magnitude faster than the MATLAB function `CPLXPAIR`.

Chapter 18 Data Interpolation

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
`mmsearch`	1D NONmonotonic linear interpolation.

The function `XI=MMSEARCH(Y,X,Yval)` linearly interpolates the vector `Y` to find `Yval` and returns the associated interpolated values from the vector `X` which must have the same length as `Y`. All crossings are found and `Y` is commonly not monotonic. Each crossing is returned as a separate row in `XI`. If `Yval` is not found, `XI=[]`.

Chapter 19 Polynomials

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
mm2dp2p	2D polynomial to 1D polynomial.
mm2dpadd	2D polynomial addition.
mm2dpchk	2D polynomial check and parse.
mm2dpder	2D polynomial derivative.
mm2dpfit	2D polynomial curve fitting.
mm2dpint	2D polynomial integration.
mm2dpstr	2D polynomial to string conversion.
mm2dpval	2D polynomial evaluation.
mm2dpxy	2D polynomial variable swap.
mmp2pm	Polynomial to polynomial matrix conversion.
mmp2str	Polynomial to string conversion.
mmpadd	Polynomial addition.
mmpfit	Polynomial curve fitting GUI.
mmpintrp	Inverse interpolate polynomial.
mmpm2p	Polynomial matrix to polynomial conversion.
mmpmder	Polynomial matrix derivative.
mmpmeval	Polynomial matrix evaluation.
mmpmfit	Polynomial matrix curve fitting.
mmpmint	Polynomial matrix integration.
mmpmsel	Select subset of polynomial matrix.
mmpoly	Make real polynomial from roots and polynomials.
mmpscale	Scale polynomial, $A(x) \rightarrow A(x/b)$.
mmpshift	Shift polynomial, $A(x) \rightarrow A(x+b)$.
mmpsim	Polynomial simplification, strip leading zeros.
mmrwpfit	Recursive weighted polynomial curve fitting.

The above functions define 2D polynomials as polynomials in X and Y that contain terms up to X^{Nx} and Y^{Ny} and all possible crossproduct terms X^iY^j where $i \le Nx$, $j \le Ny$ and $i + j \le \max(Nx,Ny)$. This corresponds to terms in the Taylor series expansion of a function of two variables. For example, $Nx = 2$, $Ny = 2$, gives the quadratic form $p_1 + p_2X + p_3X^2 + p_4Y + p_5XY + p_6Y^2$. Note that there are no terms containing XY^2, X^2Y, or X^2Y^2.

The function **mm2dp2p**`(P,'x',Val)` evaluates the 2D polynomial `P` at `x=Val` and returns the resulting 1D polynomial in Y in standard form. The function `MM2DP2P(P,'y',Val)` evaluates the 2D polynomial `P` at `y=Val` and returns the resulting 1D polynomial in X in standard form.

The function **MM2DPFIT**`(X,Y,Z,Nx,Ny)` fits the data in `X`,`Y`,`Z` to 2D polynomials in `X` and `Y` having orders `Nx` and `Ny` respectively. `X`,`Y`, and `Z` must all be the same size where `Z(i)=f(X(i),Y(i))`. `MM2DPFIT(X,Y,Z,W,Nx,Ny)` weights the data using `W`. `W(i)` is the weight associated with `X(i)`, `Y(i)`, and `Z(i)`. All elements of `W` must be positive. `X` and `Y` can be the plaid output of `MESHGRID`.

The function **MM2DPVAL**`(P,X,Y)` evaluates the 2D polynomial `P` at the values in `X` and `Y`. `X` and `Y` must be the same size and can be the plaid output of `MESHGRID`.

As used in the *Toolbox*, a polynomial matrix is simply a numerical matrix where each row contains a standard polynomial vector. Functions that work on polynomial matrices work on standard

polynomial vectors since they are simply polynomial matrices having one row.

The function `P=MMP2PM(P1,P2,P3,...)` builds a polynomial matrix `P` from the individual polynomials `P1`,`P2`, etc.

The function `MMP2STR(P)` converts the polynomial vector `P` into string representation. For example, `P = [2 3 4]` becomes the string `'2x^2 + 3x + 4'`. `MMP2STR(P,V)` generates the string using the string variable `V` as the parameter instead of `x`. `MMP2STR([2 3 4],'z')` becomes `'2z^2 + 3z + 4'`. `MMP2STR(P,1)` or `MMP2STR(P,V,1)` factors the polynomial into the product of a constant and a monic polynomial. `MMP2STR([2 3 4],1)` becomes `'2*(x^2 + 1.5x + 2)'`.

The function `MMPFIT(X,Y)` fits a polynomial `P(X)` to the data in `X` and `Y` in the least-squares sense and presents a GUI for manipulating the fit.

The function `MMPINTRP(P,y)` finds all real values of `x` where the polynomial `y=P(x)` has the scalar value `y`. If no values are found, an empty matrix is returned.

The function `MMPM2P(P,i)` extracts the polynomial in the `i`th row of the polynomial matrix `P`. `[P1,P2,P3,...] = MMPM2P(P,I)` extracts the polynomials in the rows indexed by the array `I`, i.e., `P1=P(I(1),:)`, etc.

The function `Y=MMPMEVAL(P,x)` evaluates polynomial matrix `P` at the values in `x`. The `j`th column of `Y` the evaluation of `P(j,:)` at `x(:)`. `Y(i,j)` is the evaluation of the `j`th polynomial at `x(i)`.

The function `P=MMPMFIT(x,Y,N)` where `x` is a vector and `Y` is a matrix having as many rows as `x(:)`, returns a matrix `P` of polynomials of order `N`, with the `i`th row of `P` being the polynomial associated with the `i`th column of `Y`.

The function `MMPMSEL(P,I)` returns a polynomial matrix by selecting those indexed by the vector `I`. Leading zero columns are deleted from the selection.

The function `MMPOLY` makes real polynomials from root locations and polynomials. `MMPOLY` features: Root locations can be entered as separate input arguments. e.g., `MMPOLY(-1,-2,-3) = POLY([-1;-2;-3])=(x+1)(x+2)(x+3)`, Input arguments can be *COLUMN vectors of ROOTS*, e.g., `MMPOLY(-1,[-2;-3],-4)=POLY([-1;-2;-3;-4]) = (x+1)(x+2)(x+3)(x+4)`, Input arguments can be *ROW vectors of POLYNOMIALS*, e.g., `MMPOLY(0,-1,[1 2 2])=POLY([0;-1;ROOTS([1 2 2])]) = x(x+1)(x^2+2x+2)`, The output is always a real polynomial. Complex conjugate roots are added if they do not appear in the input. e.g., `MMPOLY(0,-1+j*2)=POLY([0;-1+j*2;-1-j*2]) = x^3+2x^2+5x`.

Chapter 20 Cubic Splines

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
mmhermite	Cubic Hermite spline construction.
mmsparea	Cubic spline area.
mmspbreak	Modify spline breakpoints.
mmspchk	Check spline piecewise polynomial.
mmspcut	Extract or cut out part of a spline.
mmspder	Spline derivative.
mmspfit	Spline creation and manipulation GUI.
mmspget	Get spline data.
mmspii	Inverse interpolate spline.
mmspinfl	Spline inflection points.
mmspint	Spline integration.
mmspjump	Find spline discontinuities.
mmspline	Cubic spline construction with method choice.
mmspmath	Mathematics on spline piecewise polynomials.
mmsppaste	Paste spline piecewise polynomials.
mmspplot	Plot spline piecewise polynomials.
mmspxtrm	Cubic spline extremes.

The function **MMHERMITE**(X,Y,DY) returns the piecewise polynomial in pp-form that fits the data in X, Y, and DY. Y is a vector of data values at the breakpoints in X. DY is a vector of continuous slopes DY/DX at each breakpoint.

The function **MMSPAREA**(PP,Xmin,Xmax) computes the area under the curve described by the cubic spline piecewise polynomial PP from Xmin to Xmax.

The function **MMSPCUT**(PP,Xmin,Xmax) extracts the spline piecewise polynomial from PP, that contains the range [Xmin, Xmax].

The function PPD=**MMSPDER**(PP) returns the piecewise polynomial vector PPD describing the cubic spline derivative of the curve described by the piecewise polynomial in PP.

The function **MMSPFIT**(PP) returns a spline piecewise polynomial based on the spline piecewise polynomial PP after manipulation by a GUI.

The function Xi=**MMSPII**(PP,yi) inverse interpolates the piecewise polynomial PP, to find the points Xi where the spline has the scalar value yi.

The function [X,Y,S]=**MMSPINFL**(PP) returns the points X where the piecewise polynomial PP has zero curvature. Y=PPVAL(PP,X) and S is a vector containing the spline slopes at the points in X.

The function PPI=**MMSPINT**(PP,C) returns the piecewise polynomial vector PPI describing the integral of the cubic spline described by the piecewise polynomial in PP and having integration constant C. This function is much faster than the function mmppint described in the text.

The function [ix,idx,iddx]=**MMSPJUMP**(PP,TOL) returns vectors identifying breakpoint indices where the spline described by the piecewise polynomial PP has discontinuities in its values, slopes, and curvatures in the variables ix, idx, and iddx respectively. TOL=[RelTol AbsTol] gives desired relative and absolute tolerances for discontinuity determination. If not given TOL=[1e-3 1e-6].

The function **MMSPLINE**(X,Y,METHOD,P) computes the cubic spline interpolant from the data in X and Y, using the method in METHOD and optional parameter vector P required by some methods.

`MMSPLINE` returns the piecewise polynomial pp-form to be evaluated with `PPVAL`. `METHOD` can be any of the following: `'clamped'`, `'natural'`, `'extrap'`, `'parabolic'`, `'curvature'`, `'periodic'`, `'aperiodic'`.

The function **`MMSPMATH`**`(PP1,'op',PP2)` performs the mathematical operation `'op'` on the spline piecewise polynomials `PP1` and `PP2` and returns the resulting piecewise polynomial. `'op'` is one of the following: `'+'`, `'-'`, or `'*'`.

The function **`MMSPPASTE`**`(PP1,PP2)` pastes spline piecewise polynomial `PP2` into `PP1` returning the combined piecewise polynomial.

The function **`MMSPPLOT`**`(PP)` plots the spline having piecewise polynomial `PP` over its range of definition. `MMSPPLOT(XR,PP)` plots `PP` over the range specified by the two element vector `XR=[Xmin, Xmax]`.

The function `[Xi,Yi,C]=`**`MMSPXTRM`**`(PP)` interpolates the piecewise polynomial `PP`, to find the points `Xi` and `Yi` where the spline has zero slope. `C` is a vector containing the curvature of the spline at `Xi`.

Chapter 21 Fourier Analysis

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
fsangle	Angle between Fourier series.
fsarea	Area under Fourier series.
fsdelay	Add time delay to Fourier series.
fsderiv	Fourier series derivative.
fsdivide	Fourier series time division.
fseval	Fourier series evaluation.
fsevenodd	Fourier series even and odd time parts.
fsfind	Find Fourier series approximation.
fsformat	Fourier series format conversion.
fsharm	Fourier series harmonic component selection.
fsindex	Fourier series harmonic index vector.
fsinterp	Fourier series inverse interpolation.
fsintgrl	Fourier series integral.
fsmsv	Fourier series mean square value.
fspeak	Fourier series peak value.
fsperiod	Change Fourier series period.
fspf	Fourier series power factor.
fsplot	Fourier series function plot.
fsprod	Fourier series time product.
fsresize	Resize a Fourier series.
fsresp	Fourier series linear system response.
fsround	Round Fourier series coefficients.
fsselect	Fourier series harmonic selection.
fssize	Fourier series size.
fsstem	Fourier series stem plot.
fssum	Fourier series addition.
fssym	Enforce Fourier series symmetry.
fstable	Fourier series table.
fsthd	Fourier series total harmonic distortion.
mmfftbin	FFT bin frequencies.
mmfftpfc	FFT positive frequency components.
mmftfind	Find Fourier transform approximation.
mmwindow	Generate window functions.

In the *Toolbox*, Fourier series are stored as a row vector of the coefficients of the complex exponential form of the Fourier series as illustrated in the text *Mastering MATLAB 6*.

The function `A=FSANGLE(Kn,Fn,N)` returns the angle `A` in radians between the `N`th harmonic component of `Kn` and `Fn`. If `N` is not given, `N=+1` is assumed.

The function `FSAREA(Kn,Tmin,Tmax,T)` returns the area under the function described by the Fourier series `Kn` having period `T` from `Tmin` to `Tmax`.

The function `FSDELAY(Kn,D)` produces a Fourier Series from the Fourier Series `Kn`, delayed by the normalized time delay in `D`. `D` = delay/period, `D` > 0 delays, `D` < 0 advances the signal.

The function **FSDERIV**(Kn,T) returns the FS coeficients of the derivative of *f*(*t*) whose FS coeficients are given by Kn and whose fundamental period is T.

The function **FSEVAL**(Kn,t,T) computes values of a real-valued function given its complex exponential Fourier series coefficients Kn, at the points given in t where the period is T.

The function [En,On]=**FSEVENODD**(Kn) returns in En the FS of the even time function portion of the time function in Kn and returns in On the FS of the odd time function.

The function Fn=**FSFIND**('FUN',T,N,M) computes the complex exponential Fourier series of a real valued signal described by the function FUN which is the character string name of a user created M-file function. The function is called as f=FUN(t) where t is a vector over 0≤t≤T. T is the period of the function. N is the number of harmonics. M is an optional padding number. N+M Fourier series harmonics are found and only N are retained. Since the FFT is used, in most cases, M>0 increases accuracy.

The function **FSFORMAT** converts one Fourier series format into another based on the number of input and output arguments provided: [An,Bn,Ao]=FSFORMAT(Kn) converts exponential to trigonometric, [An,Bn,Ao]=FSFORMAT(Cn,Tn) converts alternate to exponential, [Cn,Tn]=FSFORMAT(Kn)converts exponential to alternate, [Cn,Tn]=FSFORMAT(An,Bn,Ao) converts trigonometric to alternate, Kn=FSFORMAT(Cn,Tn) converts alternate to exponential, and Kn=FSFORMAT(An,Bn,Ao) converts trigonometric to exponential. In these calls An,Bn,Ao are vectors containing the trigonometric format coefficients, Cn,Tn are vectors containing the alternate trigonometric coefficients where Cn(1) is the DC component, Cn(i) is (i-1)th harmonic amplitude, Tn(1)=0, and Tn(i) is (i-1)th harmonic angle in *degrees*, Kn = vector containing exponential coefficients as used in *Toolbox* functions.

The function **FSHARM**(Kn,N) returns the Fourier series components of Kn at the harmonic indices given in N.

The function **FSINDEX**(Kn) returns a row vector of harmonic indices based on the Fourier series vector Kn. For example, if Kn has N positive harmonics, FSINDEX returns -N:N.

The function **FSINTERP**(Kn,Val,T) returns the time points where the function *f*(*t*) described by the Fourier Series Kn has the scalar value Val. The returned time points are in the range [0,T) where T is the period.

The function **FSINTGRL**(Kn,T) returns the FS coeficients of the integral of *f*(*t*) whose FS coeficients are given by Kn and whose fundamental period is T. The integral returned is zero at *t*=0. The average or DC value of *f*(*t*) is ignored.

The function [F,Tp]=**FSPEAK**(Kn,Tspan,T) returns the peak value F of the function described by the Fourier Series Kn and the point Tp where it occurs. Tspan=[Tmin Tmax] is the range to search for the peak. Beware of Gibb's phenomenon.

The function **FSPLOT**(Kn) plots the FS Kn over the range [0 T] where T=1 is the period. FSPLOT(TK,Kn) plots the FS Kn over the range specified in TK. TK must be either [Tmin/T Tmax/T] or [Tmin Tmax T], where the range plotted is [Tmin Tmax].

The function **FSPROD**(Fn,Kn,...) returns the Fourier Series of the product of time functions having Fourier Series Fn, Kn, etc.

The function **FSRESIZE**(Kn,N) resizes the complex exponential FS Kn to have ±N harmonics. If N is greater than the number of harmonics in Kn, zeros are added to the result. If N is less than the number of harmonics in Kn, the result is a truncated version of the input.

The function **FSSELECT**(Kn,N) returns a Fourier series vector the same size as Kn, containing only the harmonics selected by the index vector N. All other harmonics are set to zero. Symmetry is enforced in N. That is if a harmonic index n appears in N, -n is added if it is not present. Indices outside the range contained in Kn are ignored.

The function **FSSTEM**(Kn,Nmin,Nmax) creates a stem plot of the amplitude of the Fourier series coefficients Kn, starting at harmonic Nmin and ending at harmonic Nmax.

The function **FSSYM**(Kn,TYPE) enforces the symmetry condition given by TYPE on the complex exponential Fourier series Kn. TYPE must be one of the following: 'even', 'odd', 'half', 'triple', or 'nodc'.

The function **FSTABLE**(FUN,N,P1) returns a Fourier Series vector of the function FUN having N positive harmonics and optional parameter P1. The zero-to-peak value of the function is 1. FUN is one of the following: 'square', 'trap' with P1 being the fractional duty cycle, 'sawtooth', 'triangle', 'pulse' with P1 being the fractional duty cycle, 'bipolar' with P1 being the fractional duty cycle, 'full', 'half', 'sine', 'cosine', 'dc'.

The function **FSTHD**(Kn) computes the total harmonic distortion of the Fouries series Kn, using its fundamental component as the reference signal.

The function **MMFFTBIN**(X,Ts) returns the continuous time bin frequencies in rad/sec associated with the components of the FFT data in X. Ts is the sampling period of the underlying time domain signal.

The function Y=**MMFFTPFC**(X) returns the DC and positive frequency components of the FFT data in X.

The function [F,w]=**MMFTFIND**('FUN',Tmin,Tmax,N) computes the Fourier transform of the real valued signal described by the function FUN which is the character string name of a user created M-file function. The function is called as f=FUN(t) where t=linspace(Tmin,Tmax,N). Make N a power of 2 for speed.

The function **MMWINDOW**(TYPE,N,alpha) creates a window vector of type TYPE having a length equal to the scalar N. TYPE is a string designating the window type desired: 'rec' = Rectangular or Boxcar, 'bar' = Bartlett (triangle with zero endpoints), 'tri' = Triangular (nonzero endpoints), 'han' = Hann or Hanning, 'ham' = Hamming, 'bla' = Blackman common coefs., 'blx' = Blackman exact coefs., 'rie' = Riemann {sin(x)/x}, 'tuk' = Tukey, 0< alpha < 1, 'poi' = Poisson, 0< alpha < inf, 'cau' = Cauchy, 1< alpha < inf, 'gau' = Gaussian, 1< alpha < inf. Reference: F.J. Harris, "On the Use of Windows for Harmonic Analysis with the Discrete Fourier Transform," *IEEE Proc.*, vol. 66, no. 1, Jan. 1978, pp 51-83.

Chapter 22 Optimization

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
mmfminc	Minimization with inequality constraints.
mmfmminc_	Helper function for mmfminc.
mmfminu	Minimize function of several variables.
mmfsolve	Solve a set of nonlinear equations.
mmlceval	Evaluate a linear combination of functions.
mmlcfit	Curve fit to a linear combination of functions.
mmnlfit	Nonlinear curve fitting.
mmnlfit2	2D nonlinear curve fitting.
mmnlfit_	Helper function for mmnlfit and mmnlfit2.
mmsneval	Simple nonlinear curve fit evaluation.
mmsnfit	Simple nonlinear curve fit by transformation.

The function **mmfminc** is described by its help text:

```
>> help mmfminc
 MMFMINC Function Minimization with Inequality Constraints. (MM)
  X=MMFMINC('FUN',Xo,K,TOL,OPTIONS) find X that minimizes a scalar
  function of a vector argument f(X), subject to a vector of
  inequality constraints G(X)<=0 as desribed in FUN.M.
  MMFMINC solves:    min f(X)   subject to  G(X)<=0
                      X
  A simple penalty function approach is used. The cost function
          P(X,K(i)) = f(X) + K(i)*sum((G(X)>0).*G(X).^2)
  is minimized using FMINSEARCH for each successive value K(i) using
  TOL(i) as the desired tolerance on X and P(X,K(i)) for the (i)th pass.
  Normally K is strictly increasing and TOL is strictly decreasing.
  OPTIONS is optional parameter structure as returned by OPTIMSET
  to set specific options for FMINSEARCH.
  The M-file FUN.M must evaluate both f(X) and G(X) and be written as
  [f,G]=FUN(X), where f=f(X) is a scalar and G=G(X).
```

The function **mmfminu** is described by its help text:

```
>> help mmfminu
 MMFMINU Minimize a Function of Several Variables. (MM)
  [Xf,TermCode]=MMFMINU('FUN',Xo,Options,P1,P2,...)
  tries to find a vector X that is a local minimizer of FUN(X) starting from
  the initial guess Xo, passing optional parameters P1,P2,... to FUN.
  'FUN' is the name of the function M-file that evaluates FUN(X,...).
  Options is an optional structure that defines algorithm behavior.
  If Options is empty, default Options are used.

  Options=MMFMINU('Name',Value,...) set values in Options structure based
  on the Name/Value pairs:
  Name        Values  {default} Description
  'Display'   ['on' {'off'}]    Display Iteration Information
  'XRelTol'   {1e-4}            Relative Error Tolerance in X
  'FRelTol'   {1e-4}            Relative Error Tolerance in FUN(X)
```

```
  'Gradient'  {'finite'}        Finite Difference Gradient
  'Gradient'   'GNAME'          Analytic Gradient in GNAME.M
  'Hessian'   {'fun'}           |FUN(Xo)|*Identity Initial Hessian
  'Hessian'    'eye'            Identity Initial Hessian
  'Hessian'     H0              Matrix H0 is Initial Hessian
  'MaxIter'   {100}             Maximum number of iterations

  Options=MMFMINU(Options,'Name','Value',...) updates the Options structure
  with new parameter values.

  Xf = final approximation
  TermCode = termination code:
             1 = normal return, 2 = change in X too small
             3 = line search failure, 4 = too many iterations
```

The function **mmfsolve** is described by its help text:

```
>> help mmfsolve
 MMFSOLVE Solve a Set of Nonlinear Equations. (MM)
  [Xf,TermCode]=MMFSOLVE('FUN',Xo,Options,P1,P2,...)
  finds a zero of the vector function FUN(X,P1,P2,...) starting from
  the initial guess Xo, passing optional parameters P1,P2,... to FUN.
  'FUN' is the name of the function M-file that evaluates FUN(X,...).
  Options is an optional structure that defines algorithm behavior.
  If Options is empty, i.e., [], default Options are used.

  Options=MMFSOLVE('Name',Value,...) sets values in Options structure
  based on the Name/Value pairs:
  Name        Values {default}   Description
  'Display'   ['on' {'off'}]     Display iteration information
  'Jacobian'  {'broyden'}        Broyden's Jacobian approximation
  'Jacobian'   'finite'          Finite Difference Jacobian
  'Jacobian'  'JNAME'            Analytic Jacobian in JNAME.M
  'FunTol'     {1e-7}            NORM(FUN(X),1) stopping tolerance
  'MaxIter'    {100}             Maximum number of iterations
  'MaxStep'    value             Maximum step size in X allowed
  'Scale'     ['on' {'off'}]     Scale algorithm using Xo

  Options=MMFSOLVE(Options,'Name','Value',...) updates the Options structure
  with new parameter values.

  Xf = final approximation
  TermCode = termination code:
             1 = normal return, 2 = two steps too small
             3 = line search failure, 4 = too many iterations
             5 = five steps too big, 6 = stuck at minimizer
```

The function **MMLCEVAL**`(x,C,FUN)` evaluates the linear combination of functions defined by the string `FUN` at the values in `x`. That is, `MMLCEVAL` returns the `y` where `y = C(1)*FUN1(x) + C(2)*FUN2(x) ... + C(n)*FUNn(x)`. `FUN` is an M-file returning a matrix having `length(x)` rows and `n` columns, where `n` is the number of functions, with the `i`th column of the output being `FUNi(x)`.

The function **MMLCFIT**`(x,y,FUN)` fits the data vector pairs x and y to the functions defined by the string `FUN`. That is, `MMLCFIT` returns the row vector `c` where `y = C(1)*FUN1(x) + C(2)*FUN2(x) ... + C(n)*FUNn(x)` is minimized in the least squares sense. `FUN` is an M-file returning a matrix having `length(x)` rows and `n` columns, where `n` is the number of functions, with the `i`th column of the output

being `FUNi(x)`. `x` is passed to each function `FUNi(x)` as a column vector. A column vector result is expected.

The function `P=`**`MMNLFIT`**`(X,Y,FUN,P0,OPTIONS)` fits the data in `X` and `Y` in a least squares sense to the function described in `FUN`. `MMNLFIT` solves: `min ||Y - FUN(X,P)||^2`, where `X` is a vector of the independent variable. `FUN(X,P)` must return an array the same size as `Y`. `P` is a vector of parameters to be determined and `P0` is the initial guess. `MMNLFIT` calls the standard MATLAB function `FMINSEARCH` to perform the minimization. `OPTIONS` is an optional structure as used by `FMINSEARCH`.

The function **`MMSNEVAL`**`(x,A,B,N)` evaluates a fitted nonlinear function from `MMSNFIT` at the points in `x`, given the coefficients `A` and `B`. `N` identifies the function to be evaluated.

The function **`mmsnfit`** is described by its help text:

```
>> help mmsnfit
 MMSNFIT Simple Nonlinear Curve Fit by Transformation. (MM)
  [A,B]=MMSNFIT(x,y,N) performs least squares curve fitting by
  linearizing the data, fitting it to a straight line,
  then inverting the linearization. A and B are the desired
  curve fit coefficients.
  N identifies the function to be fit:

   N | function     N | function        N | function
   --+-----------   ------------------  ---------------
   0 | y= A*x+B     3 | y= 1/(A*x+B)    6 | y= A*x^B
   1 | y= (A/x)+B   4 | y= 1/(A*x+B)^2  7 | y= A*log(x)+B
   2 | y= A/(x+B)   5 | y= x/(A*x+B)    8 | y= A*exp(B*x)
                                       9 | y= A*x*exp(B*x)
```

Chapter 23 Integration and Differentiation

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
`mmderiv`	Derivative using weighted central differences.
`mmgauss`	Numerically evaluate cumulative integral.
`mmintgrl`	Cumulative integral using Simpson's rule.
`mmquad`	Numerically evaluate integral.
`mmvolume`	Cumulative volume integral using trapezoidal rule.

The function **`MMDERIV`**`(X,Y)` computes the derivative of the function $y=f(x)$ given the data in `X` and `Y`. The length of `X` must equal the length of `Y`, but `X` need not be equally spaced. Weighted central differences is used, which is based on an incremental quadratic polynomial fit.

The function `Q=`**`MMGAUSS`**`(FUN,Xlims,N)` approximates the integral of `FUN(X)` using an `N` point Gauss quadrature rule, which is exact for a polynomial up to degree 2*`N`-1. The integration limits are given by `Xlims=[LowerLimit, UpperLimit]`.

The function **`MMINTGRL`**`(X,Y)` computes the integral of the function $y=f(x)$ given the data in `X` and `Y`. The length of `X` must equal the length of `Y`. `X` need not be equally spaced. Simpson's rule is used, which is based on incremental quadratic polynomial fit. This method is generally much more accurate than the trapezoidal rule used in `CUMTRAPZ`.

The function `Q=`**`MMQUAD`**`(FUN,Xlims,TOL)` approximates the integral of `FUN(X)` given integration limits `Xlims=[LowerLimit, UpperLimit]` and relative and absolute error tolerances `TOL=[RelTol AbsTol]`. `FUN` can be a character string expression, an inline function, or an M-file function. `FUN` is not evaluated at either limit.

The function `V=`**`MMVOLUME`**`(X,Y,Z)` computes the cummulative integral of the function $z=f(x,y)$ as tabulated in `X`,`Y`, and `Z`. `X` and `Y` are plaid matrices created by `MESHGRID`. `V(i,j)` is the volume under `Z` from `x(1)` to `x(j)` and `y(1)` to `y(i)`. `V(end)` is the total volume under $f(x,y)$.

Chapter 24 Differential Equations

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
mmlsim	Linear system simulation using mmodess.
mmlsim_	Helper function for mmlsim.
mmode45	ODE solution using 4-5 order mmodess.
mmode45p	Plotted ODE solution using 4-5 order mmodess.
mmodechi	ODE cubic Hermite interpolation.
mmodeini	Initialize ODE parameters for mmodess.
mmodess	Single step ODE solution, 4-5 order.

The functions associated with this chapter are associated with the ODE solution function mmodess. **mmodess** is an alternative to the ODE functions in the MATLAB ODE suite. Usage of this function is described by its help text:

```
>> help mmodess
 MMODESS Single Step ODE Solution, 4-5th Order. (MM)
  [T,Y,YP,STATS]=MMODESS('yprime',t,y,yp) integrates the
  system of first order differential equations computed in the
  M-file yprime(t,y) from the time point t where y = y(t)
  and yp = yprime(t,y). yprime(t,y) must return a column vector.

  [T,Y,YP,STATS] are the results at the integrator chosen time T>t,
  where Y=Y(T), YP=yprime(T,Y), and STATS is a vector of statistics:
  STATS=[IERR FAIL ORDER] where IERR identifies the variable Y(IERR)
  which dominated the error in the step, FAIL is the number of failed steps
  encountered in this integration step and ORDER is the order
  of the accepted solution. Order is 2, 3, or 5.
  T is scalar, Y and YP are ROW vectors.

  Integration parameters must be initialized by MMODEINI.
  Typical usage:
                mmodeini default
                t=0;           % initial time
                y=[y1;y2;...]; % initial condition column vector
                yp=feval('yprime',t,y); % initial derivatives
                while test
                    [t,y,yp]=mmodess('yprime',t,y,yp);
                    % process data
                end
```

The function **mmodeini** is used to initialize mmodess as well as set and get algorithm parameters as described by its help text:

```
>> help mmodeini
 MMODEINI Initialize ODE Parameters for MMODESS.
  MMODEINI(Name,Value, . . .) sets ODE parameters described by
  Name/Value pairs. Available pairs and default values are

   Name          Value    Limits       Description
  'RelTol'      1e-3      >0          Relative Error Tolerance
```

```
'AbsTol'       1e-6      >0          Absolute Error Tolerance
'MinStep'      1e-10     >1e-12      Minimum Stepsize
'MaxStep'        1       >=Hmin      Maximum Stepsize
'NextStep'      []       >=Hmin      Next (or Initial) Stepsize
'SafetyFactor' 0.9     (0.75,.95)   Stepsize Safety Factor
'GrowthLimit'    5     (2,20)       Stepsize Growth Limit Ratio
'ShrinkLimit'  0.1     (.05,.5)     Stepsize Shrink Limit Ratio
'FallBack'     'on'    ('on','off') Enable Fall Back on Step Failure

Changes are cummulative from one call to the next.
Only the first two characters of each Name is required.
MMODEINI default  sets above parameters to their default settings.
MMODEINI(Name) returns the value of the parameter given by Name.
MMODEINI with no input arguments displays all parameter values.
```

Chapter 25 Two-Dimensional Graphics

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
mmarrow	Plot movable arrows.
mmfill	Plot of area between two curves.
mmplotc	2D plot with ASCII markers.
mmploti	Incremental 2D line plotting.
mmplotxx	Plot data with top and bottom X axes.
mmplotyy	Plot two data arrays on a common X axis.
mmplotz	Plot with axes drawn through zero.
mmpolar	Linear or logarithmic pole coordinate plot.
mmprobe	Probe data on 2D axis using mouse.
mmzoom	Picture in a picture zoom.

The function **MMARROW**`(X,Y,'Name','Value',...)` plots arrows on the current 2D plot where `X,Y` are vectors of *x*- and *y*-axis arrow tip coordinates respectively, and other arrow properties are set by the property name-value pairs described in the function's help text. `MMARROW` with no arguments places one default arrow in the center of the current plot and enables dragging and reshaping. `MMARROW on (off)` enables (disables) dragging and reshaping of an arrow with the mouse. Normal mouse selection drags the arrow; Right mouse button selection moves the arrow tail to the mouse position but leaves the arrow tip fixed. Clicking on an open area of the *figure* window outside the plot also disables dragging and reshaping.

The function **MMPLOTI**`(V)` initializes a 2D plot for future plotting by using the axis limits given in `V=[Xmin Xmax Ymin Ymax]`. `MMPLOTI(x,Y)` plots `Y` versus `x`, appending data to any prior calls. `x` is a vector and `Y` is a vector or column oriented data matrix having as many rows as `length(x)`. `MMPLOTI done` refreshes the final plot, turns `hold` off and auto scales the axis.

The function **MMPLOTXX**`(X1,X2,Y,S)` plots `Y` versus `X1` and `X2`. `S` is an optional color and line/marker styles for the plotted lines. `X1` appears across the bottom of the plot and `X2` appears across the top. `X1` and `X2` must be vectors of the same length, but `Y` may be a matrix. Two axes objects are created. `TITLE('String')` should be avoided as it will overwrite the top *x*-axis tick marks and label. `MMPLOTXX('TopLabelString')` places/changes the *x*-axis label for `X2`.

The function **MMPLOTYY**`(X,Y,S1,Ylim,R,S2,Rlim)` plots `Y` vs. `X` with `Y` labeled on the left hand side, and plots `R` vs. `X` with `R` labeled on the right hand side. `X` must be a vector, whereas `Y` and `R` can be matrices. `Ylim = [Ymin Ymax]` and `Rlim = [Rmin Rmax]` are optional row vectors specifying `Y` and `R` axis limits. `S1` and `S2` are optional color and line/marker style for `Y` and `R` data. `MMPLOTYY('RLabelstring')` places/changes a right side axis label. Only a single *axes* object is created.

The function **MMPLOTZ**`(X,Y)` plots vector `X` versus vector `Y` with axes drawn through the origin. If `X` or `Y` is a matrix, then the vector is plotted versus the rows or columns of the matrix, whichever line up.

The function **MMPOLAR**`(THETA,RHO,S1,S2,...)` generates a polar coordinate plot using the angle vector `THETA` in radians and radius array `RHO` and optional linestyle specifications `S1`, `S2`, etc. `THETA` must be a vector, but `RHO` can be either a vector the same size as `THETA` or a matrix having `length(THETA)` rows. `MMPOLAR(THETA,RHO,'log',S1,S2,...)` generates a polar plot with a log10 `RHO` axis. `MMPOLAR on {off}` enables {disables} feedback of mouse position over the current plot with the mouse button down. `MMPOLAR('Name','Value',...)` sets polar property values according to the

property name-property value pairs available in the help text of the function.

The function **mmprobe** is a general GUI for analyzing 2D line plots. This function places two vertical and two horizontal markers on the current axes to be dragged as desired using the mouse. The mantissa of the axis coordinate is displayed next to each marker line. The text area below the *x*-axis displays information about the box formed by the marker lines. Holding the **Control** key down as you drag moves both horizontal or vertical lines in unison. Pressing various keyboard keys invoke actions as described in the help text for `mmprobe`. To interpolate plotted data, click and drag on a plotted line. Holding the **Control** key down as you drag shows a tangent line as well. The text area below the *x*-axis displays information about the plotted data. Pressing various keyboard keys invoke actions as described in the help text.

The function **MMZOOM** creates a new *axes* containing the data inside a box formed by a click and drag with the mouse in the current *axes*. The new zoomed *axes* is placed in the upper left of the current *axes*, but can be moved with the mouse. Clicking in the figure border or issuing `MMZOOM RESET` disables dragging. `MMZOOM DRAG` reenables dragging. `MMZOOM SUBPLOT` creates the zoomed *axes* as a subplot below the original *axes*. The original *axes* must not itself be a subplot.

Chapter 26 Three-Dimensional Graphics

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
`mmhole`	Create hole in 3D graphics data.
`mminxy`	Minima of 3D data along X and Y axes.
`mmprobe3`	Probe data on 3D axes using mouse.
`mmxtract`	Extract subset of 3D graphics data.
`mmzoom3`	3D zoom using a rubberband box.

The function `Z=MMHOLE(X,Y,Z,Xlim,Ylim)` sets the data in `Z` to `NaN` corresponding to the limits in `Xlim=[Xmin Xmax]` and `Ylim=[Ymin Ymax]`. If `Xlim` or `Ylim` are empty they are assumed to be `[-inf inf]`. `X` and `Y` can be plaid matrices, e.g., created by `MESHGRID` or they can be vectors defining the *x*- and *y*-axes.

The function `[Xx,Yx,Zx,Xy,Yy,Zy]=MMINXY(X,Y,Z)` finds minima in the `Z` matrix along the axis directions in `X` and `Y`. `X` and `Y` can be plaid matrices, e.g., created by `MESHGRID` or they can be vectors defining the *x*- and *y*-axes.

The function `MMPROBE3` duplicates the current figure containing a 3D surface and creates three other subplots containing: a contour plot of the surface, a 2D line plot of `Z` vs `X` at a specific `Y` value, and a 2D line plot of `Z` vs `Y` at a specific `X` value. The two line plots can be changed by dragging the corresponding marker line in the contour plot, or by entering the desired `X` or `Y` values into the edit boxes provided. Pressing various keyboard keys with the mouse over the plot invoke actions described in the help text to the function.

The function `[Xx,Yx,Zx]=MMXTRACT(X,Y,Z,Xlim,Ylim)` extracts a data subset from the set `[X,Y,Z]` that reside within the limits in `Xlim=[Xmin Xmax]` and `Ylim=[Ymin Ymax]`. If `Xlim` or `Ylim` are empty they are assumed to be `[-inf inf]`. `X` and `Y` can be plaid matrices, e.g., created by `MESHGRID` or they can be vectors defining the *x*- and *y*-axes.

The function `MMZOOM3` changes the view of the current axes to 2-D then the user drags a rubberband box to select part of the *X*-*Y* plane for zooming. Finally the 3-D plot is redrawn with the new *X*- and *y*-axis limits and the *z*-axis remains autoscaled. Works for a single surface plots only.

Chapter 27 Using Color and Light

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
`mmap`	Single color colormap.
`mmrgb`	Color specification conversion and substitution.
`mmrgb2gray`	Convert colormap to grayscale.
`rainbow`	Colormap variant to `hsv`.

The function **`MMAP`**`(C,M)` makes a colormap of length `M` starting with the basic colorspec `C`. The map changes from dark to light, but does not include black or white.

The function **`MMRGB`**`(S)` where `S` is a single character colorspec or the complete name of a standard color, returns its numerical RGB equivalent.

The function **`MMRGB2GRAY`**`(M)` returns a colormap the same size as `M` with colors replaced by their NTSC brightness components.

The function `RAINBOW(M)` returns a rainbow colormap with `M` entries.

Chapter 30 Handle Graphics

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
`mmbox`	Get 2D axis vector of a rubberband box.
`mmedit`	Edit axes text using mouse.
`mmfigpos`	Position figure windows.
`mmfitpos`	Fit position with another object.
`mmgca`	Get current axes if it exists.
`mmgcf`	Get current figure if it exists.
`mmget`	Get multiple object properties.
`mmgetpos`	Get object position in specified units.
`mmgetpt`	Get graphical point with interpolation.
`mmgetset`	Get settable object property structure.
`mmgetsiz`	Get font size in specified units.
`mmgetundoc`	Get undocumented object properties.
`mmginput`	Graphical input using mouse.
`mmgrid`	Custom axis grid.
`mmgui`	Double-click activation of GUIs.
`mminrect`	True when point is inside position rectangle.
`mmis2d`	True for axes that are 2D.
`mmpaper`	Set default paper properties.
`mmplotpos`	Subplot position vectors.
`mmprintf`	Data array to string matrix conversion.
`mmputptr`	Place mouse pointer.
`mmsetpos`	Set object position relative to another object.
`mmtext`	Place and drag text with mouse.
`mmxy`	Show and get x-y coordinates with mouse.
`mmzap`	Delete graphics object using mouse.

The function **`MMBOX`** returns an axis vector `[xmin xmax ymin ymax]` of a rubberband box drawn in the current axes by the user with the mouse. The returned vector cannot be larger than the current axes limits, nor smaller than 1/20 of the limits.

The function **`MMFIGpos`** `cascade` cascades currently visible figures. `mmfigpos tile` tiles currently visible figures. `mmfigpos off` resets all visible figures to the default position. `mmfigpos full` changes the current figure to full screen.

The function **`MMFITPOS`**`(P,H,Units)` Modifies the position vector `P` as required to make it fit within the position rectangle of the object having handle `H`. `Units` defines the units used and is one of: `'pixels'`, `'normalized'`, `'points'`, `'inches'`, `'cent'`.

The function **`MMGETPOS`**`(H,Units)` returns the position vector associated with the graphics object having handle `H` in the units specified by `Units`. `Units` is one of: `'pixels'`, `'normalized'`, `'points'`, `'inches'`, `'cent'`. `'Units'` equal to `'data'` is valid for *text* objects.

The function `[X,Y,Ydot]=`**`MMGETPT`**`(XY,Xdata,Ydata,W)` uses the point `XY=[x y]` to find the nearest point on the piecewise linear interpolation of the graphical data vectors `Xdata` and `Ydata`. `W` is an optional scalar weight that gives weight `W` to the *x*-axis data and `(1-W)` weight to the *y*-axis data. By default `W=1` and only *x*-axis data is used to find the closest point. `X` and `Y` are the coordinates of the

interpolated point. `Ydot` is the slope at the interpolated point based on quadratic interpolation of the three `Xdata` and `Ydata` points nearest to `XY`.

The function **`MMGETSET`**`(H)` gets all the properties of the object having handle `H` in the form of a structure and removes those fields that are read only.

The function **`MMGETSIZ`**`(H,Units)` returns the font size associated with the *axes*, *text*, or *uicontrol* object having handle `H` in the units specified by `Units`. `Units` is one of: `'points'`, `'normalized'`, `'pixels'`, `'inches'`, `'centimeters'`.

The function **`MMGETUNDOC`** `OBJECT` or `MMGETUNDOC(H)` displays a list of undocumented property names for the handle graphics object having handle `H` or indentified by the string `OBJECT`.

The function `[X,Y]=`**`MMGINPUT`**`(N)` gets `N` points from the current *axes* at points selected with a mouse button press. The `N` points are restricted to the piecewise linear interpolation of the plotted line data points. Striking ANY key on the keyboard aborts the process.

The function **`MMGRID`**`(xyz,V)` places grid lines along the axis specified by the string `xyz` at the locations given in the numerical vector `V`. For example, `MMGRID('y',[-1 2])` places grid lines at `y=-1` and `y=2`. Standard grid lines and tick locations are not changed. More than one axis or multiple commands for one axis can be specified by repeating input arguments. For example, `MMGRID('xyz1',V1,'xyz2',V2,...)`. If `V='on'` or `'off'`, standard grid lines are added or removed. For example, `MMGRID('x','on')` turns the *x*-axis grid on. `MMGRID('x','on','y',[-1 2])` turns the standard *x*-axis grid on and places custom grid lines at `y=-1` and `y=2`.

The function **`MMPRINTF`**`(FORMAT,A)` formats the numerical array `A` using the specified `FORMAT` string, and returns a string matrix whose `i`th row is the formatted `i`th element of `A`.

The function **`MMPUTPTR`**`(H)` sets the mouse pointer location to the center of the object having handle `H`.

The function **`MMSETPOS`**`(P,H,Units,Horizontal,Vertical)` modifies the position vector `P` as required relative to position rectangle of the object having handle `H`. `Units`, `Horizontal`, and `Vertical` are strings specifying the units used and the desired relative location as described by the help text for `mmsetpos`.

Chapter 31 Graphical User Interfaces

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
`mmsaxes`	Set axes specifications using a GUI.
`mmsfont`	Set font characteristics using a GUI.
`mmsline`	Set line specifications using a GUI.
`mmsmap`	Set figure colormap using a GUI.
`mmssurf`	Set surface specifications using a GUI.
`mmstext`	Set text specifications using a GUI.
`mmstick`	Set axes tick specifications using a GUI.

The functions listed in the table above are self-explanitory GUIs.

Chapter 33 MATLAB Classes and OOP

The *Mastering MATLAB Toolbox* functions associated with this chapter are shown in the table below.

Function	Description
`mmclass`	Object class existence.
`mmrp`	Rational polynomial object class.

The function **`MMCLASS`** returns a cell array of strings containing the names of MATLAB object classes available with this license. `MMCLASS('ClassName')` returns logical True (1) if the class having the name `'ClassName'` exists with this license. Otherwise logical False (0) is returned.

The function `mmrp` is the constructor for a rational polynomial class. The `@mmrp` subdirectory contains many overloaded operators and functions (See the `Contents.m` file in the *Toolbox* for a listing.)