

asec	Inverse Secant.	
csc	Cosecant.	
acsc	Inverse Cosecant.	
cot	Cotangent.	
acot	Inverse Cotangent.	
exp	Exponential.	
log	Natural logarithm.	
log10	- Common (base 10) logarithm.	
sqrt	Square root.	
abs	Absolute value.	
angle	Phase angle.	
conj	complex conjugate.	
imag	complex imaginary part.	
real	complex real part.	
fix	Round towards zero.	
floor	Round towards Minus Infinity.	
ceil	Round towards Plus Infinity.	
round	Round towards nearest integer.	
mod	- Modulus (signed remainder after division).	
rem	remainder after division.	
sign	Signum.	
norm	Matrix or vector norm.	
rank	Matrix rank.	
det	Determinant.	
expm	Matrix Exponential.	
logm	Matrix logarithm.	
sqrtm	Matrix Square root.	
max	Largest component.	
min	Smallest component.	
mean	Average or mean value.	
median	Median value.	
std	Standard deviation.	
sort	Sort in ascending order.	
sortrows	Sort rows in ascending order.	
sum	Sum of elements.	
prod	product of elements.	
hist	Histogram.	
diff	Difference and approximate derivative.	
gradient	approximate gradient.	
fft	Discrete Fourier transform.	
plot -	Linear plot.	
loglog -	Log-log scale plot.	
semilogx -	Semi-log scale plot.	
semilogy -	Semi-log scale plot.	
polar -	Polar coordinate plot.	
plotyy -	Graphs with y tick labels on the left and right.	

axis	Control axis scaling and appearance.	
zoom	- Zoom in and out on a 2-D plot.	
grid	Grid lines.	
legend -	Graph legend.	
title -	Graph title.	
xlabel -	X-axis label.	
ylabel -	Y-axis label.	
mesh	3-D mesh surface.	
surf	3-D colored surface.	
fill3	- Filled 3-D polygons.	
colormap	Color look-up table.	
figure	Create figure window.	
axes	Create axes.	
char	- Create character array (string).	
strcat	Concatenate strings.	
num2str	Convert number to string.	
int2str	Convert integer to string.	
fopen	Open file.	
fclose	Close file.	
fread	Read binary data from file.	
fwrite	Write binary data to file.	
fscanf	Read formatted data from file.	
fprintf	Write formatted data to file.	
fgetl	Read line from file, discard newline character.	
fgets	Read line from file, keep newline character.	
input	Prompt for user input.	
clc	Clear command window.	
home	Send cursor home.	
pause	Wait for user response.	
now	current date and time as date number.	
date	current date as date string.	
clock	current date and time as date vector.	
abs	Magnitude.	
what	List MATLAB-specific files in directory.	
lookfor	Search all M-files for keyword.	
which	Locate functions and files.	
path	Get/set Search path.	
echo	Echo commands in M-files.	
more	Control paged output in command window.	
diary	Save text of MATLAB session.	
format	set output format.	
delete	Delete file.	
if	Conditionally Execute statements.	
else	if statement condition.	
elseif	if statement condition.	
end	Terminate scope of for, WHILE, SWITCH and if statements.	

for	Repeat statements a specific number of times.
while	Repeat statements an indefinite number of times.
break	Terminate execution of WHILE or for loop.
switch	SWITCH among several cases based On expression.
case	SWITCH statement case.
otherwise	Default SWITCH statement case.
return	return to invoking function.
reshape	Change size.
flipdim	Flip Matrix along specified dimension.
rot90	- Rotate matrix 90 degrees.
find	Find indices of nonzero elements.
end	Last index.
eps	floating point relative accuracy.
realmax	Largest positive floating point number.
realmin	Smallest positive floating point number.
why	Succinct answer.
sinh	Hyperbolic Sine.
asinh	Inverse Hyperbolic Sine.
cosh	Hyperbolic Cosine.
acosh	Inverse Hyperbolic Cosine.
tanh	Hyperbolic Tangent.
atanh	Inverse Hyperbolic Tangent.
sech	Hyperbolic Secant.
asech	Inverse Hyperbolic Secant.
csch	Hyperbolic Cosecant.
acsch	Inverse Hyperbolic Cosecant.
coth	Hyperbolic Cotangent.
acoth	Inverse Hyperbolic Cotangent.
log2	- Base 2 logarithm and dissect floating point number.
pow2	- Base 2 power and scale floating point number.
nextpow2	- Next higher power of 2.
cumsum	Cumulative Sum of elements.
cumprod	Cumulative product of elements.
corrcoef	Correlation coefficients.
cov	Covariance matrix.
fft2	two-dimensional Discrete Fourier transform.
fftn	N-dimensional Discrete Fourier transform.
roots	Find Polynomial roots.
residue	- Partial-fraction expansion (residues).
conv	multiply polynomials.
deconv	divide polynomials.
double	Convert string to numeric character codes.
sprintf	Write formatted data to string.
sscanf	Read string under format control.
demo	Run demonstrations.
ver	MATLAB, SIMULINK, and toolbox version information.

matlab-help-text

addpath	Add directory to Search path.	
rmpath	Remove directory from Search path.	
editpath	Modify Search path.	
!	Execute operating system command.	
dos	Execute DOS command and return result.	
unix	Execute UNIX command and return result.	
vms	Execute VMS DCL command and return result.	
fliplr	Flip Matrix in left/right direction.	
flipud	Flip Matrix in up/down direction.	
pack	Consolidate workspace memory.	

General purpose commands.

- General information
- Managing the workspace.
- Managing commands and functions.
- Managing the search path
- Controlling the command window.
- Operating system commands
- Debugging M-files.
- Profiling M-files.

Operators and special characters.

- Arithmetic operators.
- Relational operators.
- Logical operators.
- Special characters.
- Bitwise operators.
- Set operators.

Programming language constructs.

- Control flow.
- Evaluation and execution.
- Scripts, functions, and variables.
- Argument handling.
- Message display.
- Interactive input.

Elementary matrices and matrix manipulation.

- Elementary matrices.
- Basic array information.
- Matrix manipulation.
- Special variables and constants.
- Specialized matrices.

Elementary math functions.

- Trigonometric.
- Exponential.
- Complex.
- Rounding and remainder.

Specialized math functions.

- Specialized math functions.
- Number theoretic functions.
- Coordinate transforms.

Matrix functions - numerical linear algebra.

- Matrix analysis.
- Linear equations.
- Eigenvalues and singular values.
- Matrix functions.
- Factorization utilities

Data analysis and Fourier transforms.

- Basic operations.
- Finite differences.
- Correlation.
- Filtering and convolution.
- Fourier transforms.
- Sound and audio.
- Audio file inport/export.

Interpolation and polynomials.

- Data interpolation.
- Spline interpolation.
- Geometric analysis.
- Polynomials.

Function functions and ODE solvers.

- Optimization and root finding.
- Numerical integration (quadrature).
- Plotting.
- Inline function object.
- Ordinary differential equation solvers.
- ODE Option handling.
- ODE output functions.

Sparse matrices.

- Elementary sparse matrices.
- Full to sparse conversion.
- Working with sparse matrices.
- Reordering algorithms.
- Linear algebra.
- Linear Equations (iterative methods).
- Operations on graphs (trees).
- Miscellaneous.

Two dimensional graphs.

- Elementary X-Y graphs.
- Axis control.
- Graph annotation.
- Hardcopy and printing.

Three dimensional graphs.

- Elementary 3-D plots.
- Color control.
- Lighting.
- Color maps.
- Axis control.
- Viewpoint control.
- Graph annotation.
- Hardcopy and printing.

Specialized graphs.

- Specialized 2-D graphs.
- Contour and 2-1/2 D graphs.
- Specialized 3-D graphs.
- Images display and file I/O.
- Movies and animation.
- Color related functions.
- Solid modeling.

Handle Graphics.

- Figure window creation and control.
- Axis creation and control.
- Handle Graphics objects.
- Handle Graphics operations.
- Hardcopy and printing.
- Utilities.

Graphical user interface tools.

- GUI functions.
- GUI design tools.
- Dialog boxes.
- Menu utilities.
- Toolbar button group utilities.
- User-defined figure/axes property utilities.
- Miscellaneous utilities.

Character strings.

- General.
- String tests.
- String operations.
- String to number conversion.
- Base number conversion.

File input/output.

- File opening and closing.
- Binary file I/O.
- Formatted file I/O.
- String conversion.
- File positioning.
- File name handling
- File import/export functions.
- Image file import/export.
- Audio file import/export.
- Command window I/O

Time and dates.

- Current date and time.
- Basic functions.
- Date functions.
- Timing functions.

Data types and structures.

- Data types (classes)
- Multi-dimensional array functions.
- Cell array functions.
- Structure functions.
- Object oriented programming functions.
- Overloadable operators.

Signal Processing Toolbox.

- Waveform generation.
- Filter analysis and implementation.
- Linear system transformations.
- IIR digital filter design.
- IIR filter order selection.
- FIR filter design.
- Transforms.
- Statistical signal processing and spectral analysis.
- Windows.
- Parametric modeling.
- Specialized operations.
- Analog lowpass filter prototypes.
- Frequency translation.
- Filter discretization.
- Other.
- Signal GUI (Graphical User Interface).
- Demonstrations.

Symbolic Math Toolbox.

- Calculus.
- Linear Algebra.
- Simplification.
- Solution of Equations.
- Variable Precision Arithmetic.
- Integral Transforms.
- Conversions.
- Basic Operations.
- Special Functions.
- String handling utilities.
- Pedagogical and Graphical Applications.
- Demonstrations.
- Access to Maple. (Not available with Student Edition.)

General purpose commands.

MATLAB Toolbox Version 5.0 Student Edition 31-Dec-1996

General information

help - On-line help, display text at command line.
helpwin - On-line help, separate window for navigation.
helpdesk - Comprehensive hypertext documentation and troubleshooting.
demo - Run demonstrations.
ver - MATLAB, SIMULINK, and toolbox version information.
whatsnew - Display Readme files.
Readme - What's new in MATLAB 5.

Managing the workspace.

who - List current variables.
whos - List current variables, long form.
clear - Clear variables and functions from memory.
pack - Consolidate workspace memory.
load - Load workspace variables from disk.
save - Save workspace variables to disk.
quit - Quit MATLAB session.

Managing commands and functions.

what - List MATLAB-specific files in directory.
type - List M-file.
edit - Edit M-file.
lookfor - Search all M-files for keyword.
which - Locate functions and files.
pcode - Create pre-parsed pseudo-code file (P-file).
inmem - List functions in memory.
mex - Compile MEX-function.

Managing the search path

path - Get/set search path.
addpath - Add directory to search path.
rmpath - Remove directory from search path.
editpath - Modify search path.

Controlling the command window.

echo - Echo commands in M-files.
more - Control paged output in command window.
diary - Save text of MATLAB session.
format - Set output format.

Operating system commands

cd - Change current working directory.
pwd - Show (print) current working directory.
dir - List directory.
delete - Delete file.
getenv - Get environment variable.
! - Execute operating system command.
dos - Execute DOS command and return result.
unix - Execute UNIX command and return result.
vms - Execute VMS DCL command and return result.
web - Open Web browser on site or files.
computer - Computer type.

Debugging M-files.

debug - List debugging commands.
dbstop - Set breakpoint.
dbclear - Remove breakpoint.
dbcont - Continue execution.
dbdown - Change local workspace context.
dbstack - Display function call stack.
dbstatus - List all breakpoints.
dbstep - Execute one or more lines.
dbtype - List M-file with line numbers.
dbup - Change local workspace context.
dbquit - Quit debug mode.
dbmex - Debug MEX-files (UNIX only).

Profiling M-files.

profile - Profile M-file execution time.

Operators and special characters.

Arithmetic operators.

plus	- Plus	+
uplus	- Unary plus	+
minus	- Minus	-
uminus	- Unary minus	-
mtimes	- Matrix multiply	*
times	- Array multiply	.*
mpower	- Matrix power	^
power	- Array power	.^
mldivide	- Backslash or left matrix divide	\
mrdivide	- Slash or right matrix divide	/
ldivide	- Left array divide	.\
rdivide	- Right array divide	./
kron	- Kronecker tensor product	kron

Relational operators.

eq	- Equal	==
ne	- Not equal	~=
lt	- Less than	<
gt	- Greater than	>
le	- Less than or equal	<=
ge	- Greater than or equal	>=

Logical operators.

and	- Logical AND	&
or	- Logical OR	
not	- Logical NOT	~
xor	- Logical EXCLUSIVE OR	
any	- True if any element of vector is nonzero	
all	- True if all elements of vector are nonzero	

Special characters.

colon	- Colon	:
paren	- Parentheses and subscripting	()
paren	- Parentheses and subscripting	()
paren	- Brackets	[]
paren	- Braces and subscripting	{ }
paren	- Braces and subscripting	{ }
punct	- Decimal point	.
punct	- Structure field access	.
punct	- Parent directory	..
punct	- Continuation	...
punct	- Separator	,
punct	- Semicolon	;
punct	- Comment	%
punct	- Invoke operating system command	!
punct	- Assignment	=
punct	- Quote	'
transpose	- Transpose	.'
ctranspose	- Complex conjugate transpose	'
horzcat	- Horizontal concatenation	[,]
vertcat	- Vertical concatenation	{;}
subsasgn	- Subscripted assignment	(),{ }..
subsref	- Subscripted reference	(),{ }..
subsindex	- Subscript index	

Bitwise operators.

bitand	- Bit-wise AND.
bitcmp	- Complement bits.
bitor	- Bit-wise OR.
bitmax	- Maximum floating point integer.
bitxor	- Bit-wise XOR.
bitset	- Set bit.
bitget	- Get bit.
bitshift	- Bit-wise shift.

Set operators.

union	- Set union.
unique	- Set unique.
intersect	- Set intersection.
setdiff	- Set difference.
setxor	- Set exclusive-or.
ismember	- True for set member.

See also ARITH, RELOP, SLASH.

Programming language constructs.

Control flow.

- if - Conditionally execute statements.
- else - IF statement condition.
- elseif - IF statement condition.
- end - Terminate scope of FOR, WHILE, SWITCH and IF statements.
- for - Repeat statements a specific number of times.
- while - Repeat statements an indefinite number of times.
- break - Terminate execution of WHILE or FOR loop.
- switch - Switch among several cases based on expression.
- case - SWITCH statement case.
- otherwise - Default SWITCH statement case.
- return - Return to invoking function.

Evaluation and execution.

- eval - Execute string with MATLAB expression.
- feval - Execute function specified by string.
- evalin - Evaluate expression in workspace.
- builtin - Execute built-in function from overloaded method.
- assignin - Assign variable in workspace.
- run - Run script.

Scripts, functions, and variables.

- script - About MATLAB scripts and M-files.
- function - Add new function.
- global - Define global variable.
- mfilename - Name of currently executing M-file.
- lists - Comma separated lists.
- exist - Check if variables or functions are defined.
- isglobal - True for global variables.

Argument handling.

- nargchk - Validate number of input arguments.
- nargin - Number of function input arguments.
- nargout - Number of function output arguments.
- varargin - Variable length input argument list.
- varargout - Variable length output argument list.
- inputname - Input argument name.

Message display.

- error - Display error message and abort function.
- warning - Display warning message.
- lasterr - Last error message.
- errortrap - Skip error during testing.
- disp - Display an array.
- fprintf - Display formatted message.
- sprintf - Write formatted data to a string.

Interactive input.

- input - Prompt for user input.
- keyboard - Invoke keyboard from M-file.
- pause - Wait for user response.
- uimenu - Create user interface menu.
- uicontrol - Create user interface control.

Elementary matrices and matrix manipulation.

Elementary matrices.

zeros - Zeros array.
ones - Ones array.
eye - Identity matrix.
repmat - Replicate and tile array.
rand - Uniformly distributed random numbers.
randn - Normally distributed random numbers.
linspace - Linearly spaced vector.
logspace - Logarithmically spaced vector.
meshgrid - X and Y arrays for 3-D plots.
: - Regularly spaced vector and index into matrix.

Basic array information.

size - Size of matrix.
length - Length of vector.
ndims - Number of dimensions.
disp - Display matrix or text.
isempty - True for empty matrix.
isequal - True if arrays are identical.
isnumeric - True for numeric arrays.
islogical - True for logical array.
logical - Convert numeric values to logical.

Matrix manipulation.

reshape - Change size.
diag - Diagonal matrices and diagonals of matrix.
tril - Extract lower triangular part.
triu - Extract upper triangular part.
fliplr - Flip matrix in left/right direction.
flipud - Flip matrix in up/down direction.
flipdim - Flip matrix along specified dimension.
rot90 - Rotate matrix 90 degrees.
: - Regularly spaced vector and index into matrix.
find - Find indices of nonzero elements.
end - Last index.
sub2ind - Linear index from multiple subscripts.
ind2sub - Multiple subscripts from linear index.

Special variables and constants.

ans - Most recent answer.
eps - Floating point relative accuracy.
realmax - Largest positive floating point number.
realmin - Smallest positive floating point number.
pi - 3.1415926535897....
i, j - Imaginary unit.
inf - Infinity.
NaN - Not-a-Number.
isnan - True for Not-a-Number.
isinf - True for infinite elements.
isfinite - True for finite elements.
flops - Floating point operation count.
why - Succinct answer.

Specialized matrices.

compan - Companion matrix.
gallery - Higham test matrices.
hadamard - Hadamard matrix.
hankel - Hankel matrix.
hilb - Hilbert matrix.
invhilb - Inverse Hilbert matrix.
magic - Magic square.
pascal - Pascal matrix.
rosser - Classic symmetric eigenvalue test problem.
toeplitz - Toeplitz matrix.
vander - Vandermonde matrix.
wilkinson - Wilkinson's eigenvalue test matrix.

Elementary math functions.

Trigonometric.

sin - Sine.
sinh - Hyperbolic sine.
asin - Inverse sine.
asinh - Inverse hyperbolic sine.
cos - Cosine.
cosh - Hyperbolic cosine.
acos - Inverse cosine.
acosh - Inverse hyperbolic cosine.
tan - Tangent.
tanh - Hyperbolic tangent.
atan - Inverse tangent.
atan2 - Four quadrant inverse tangent.
atanh - Inverse hyperbolic tangent.
sec - Secant.
sech - Hyperbolic secant.
asec - Inverse secant.
asech - Inverse hyperbolic secant.
csc - Cosecant.
csch - Hyperbolic cosecant.
acsc - Inverse cosecant.
acsch - Inverse hyperbolic cosecant.
cot - Cotangent.
coth - Hyperbolic cotangent.
acot - Inverse cotangent.
acoth - Inverse hyperbolic cotangent.

Exponential.

exp - Exponential.
log - Natural logarithm.
log10 - Common (base 10) logarithm.
log2 - Base 2 logarithm and dissect floating point number.
pow2 - Base 2 power and scale floating point number.
sqrt - Square root.
nextpow2 - Next higher power of 2.

Complex.

abs - Absolute value.
angle - Phase angle.
conj - Complex conjugate.
imag - Complex imaginary part.
real - Complex real part.
unwrap - Unwrap phase angle.
isreal - True for real array.
cplxpair - Sort numbers into complex conjugate pairs.

Rounding and remainder.

fix - Round towards zero.
floor - Round towards minus infinity.
ceil - Round towards plus infinity.
round - Round towards nearest integer.
mod - Modulus (signed remainder after division).
rem - Remainder after division.
sign - Signum.

Specialized math functions.

Specialized math functions.

airy - Airy functions.
besselj - Bessel function of the first kind.
bessely - Bessel function of the second kind.
besselh - Bessel functions of the third kind (Hankel function).
besseli - Modified Bessel function of the first kind.
besselk - Modified Bessel function of the second kind.
beta - Beta function.
betainc - Incomplete beta function.
betaln - Logarithm of beta function.
ellipj - Jacobi elliptic functions.
ellipke - Complete elliptic integral.
erf - Error function.
erfc - Complementary error function.
erfcx - Scaled complementary error function.
erfinv - Inverse error function.
expint - Exponential integral function.
gamma - Gamma function.
gammainc - Incomplete gamma function.
gammainv - Logarithm of gamma function.
legendre - Associated Legendre function.
cross - Vector cross product.

Number theoretic functions.

factor - Prime factors.
isprime - True for prime numbers.
primes - Generate list of prime numbers.
gcd - Greatest common divisor.
lcm - Least common multiple.
rat - Rational approximation.
rats - Rational output.
perms - All possible permutations.
nchoosek - All combinations of N elements taken K at a time.

Coordinate transforms.

cart2sph - Transform Cartesian to spherical coordinates.
cart2pol - Transform Cartesian to polar coordinates.
pol2cart - Transform polar to Cartesian coordinates.
sph2cart - Transform spherical to Cartesian coordinates.
hsv2rgb - Convert hue-saturation-value colors to red-green-blue.
rgb2hsv - Convert red-green-blue colors to hue-saturation-value.

Matrix functions - numerical linear algebra.

Matrix analysis.

- norm - Matrix or vector norm.
- normest - Estimate the matrix 2-norm.
- rank - Matrix rank.
- det - Determinant.
- trace - Sum of diagonal elements.
- null - Null space.
- orth - Orthogonalization.
- rref - Reduced row echelon form.
- subspace - Angle between two subspaces.

Linear equations.

- \ and / - Linear equation solution; use "help slash".
- inv - Matrix inverse.
- cond - Condition number with respect to inversion.
- condest - 1-norm condition number estimate.
- chol - Cholesky factorization.
- cholinc - Incomplete Cholesky factorization.
- lu - LU factorization.
- luinc - Incomplete LU factorization.
- qr - Orthogonal-triangular decomposition.
- nnls - Non-negative least-squares.
- pinv - Pseudoinverse.
- lsqov - Least squares with known covariance.

Eigenvalues and singular values.

- eig - Eigenvalues and eigenvectors.
- svd - Singular value decomposition.
- eigs - A few eigenvalues.
- svds - A few singular values.
- poly - Characteristic polynomial.
- polyeig - Polynomial eigenvalue problem.
- condeig - Condition number with respect to eigenvalues.
- hess - Hessenberg form.
- qz - QZ factorization for generalized eigenvalues.
- schur - Schur decomposition.

Matrix functions.

- expm - Matrix exponential.
- logm - Matrix logarithm.
- sqrtn - Matrix square root.
- funm - Evaluate general matrix function.

Factorization utilities

- qrdelete - Delete column from QR factorization.
- qrinsert - Insert column in QR factorization.
- rsf2csf - Real block diagonal form to complex diagonal form.
- cdf2rdf - Complex diagonal form to real block diagonal form.
- balance - Diagonal scaling to improve eigenvalue accuracy.
- planerot - Given's plane rotation.

Data analysis and Fourier transforms.

Basic operations.

- max - Largest component.
- min - Smallest component.
- mean - Average or mean value.
- median - Median value.
- std - Standard deviation.
- sort - Sort in ascending order.
- sortrows - Sort rows in ascending order.
- sum - Sum of elements.
- prod - Product of elements.
- hist - Histogram.
- trapz - Trapezoidal numerical integration.
- cumsum - Cumulative sum of elements.
- cumprod - Cumulative product of elements.
- cumtrapz - Cumulative trapezoidal numerical integration.

Finite differences.

- diff - Difference and approximate derivative.
- gradient - Approximate gradient.
- del2 - Discrete Laplacian.

Correlation.

- corrcoef - Correlation coefficients.
- cov - Covariance matrix.
- subspace - Angle between subspaces.

Filtering and convolution.

- filter - One-dimensional digital filter.
- filter2 - Two-dimensional digital filter.
- conv - Convolution and polynomial multiplication.
- conv2 - Two-dimensional convolution.
- convn - N-dimensional convolution.
- deconv - Deconvolution and polynomial division.

Fourier transforms.

- fft - Discrete Fourier transform.
- fft2 - Two-dimensional discrete Fourier transform.
- fftn - N-dimensional discrete Fourier Transform.
- ifft - Inverse discrete Fourier transform.
- ifft2 - Two-dimensional inverse discrete Fourier transform.
- ifftn - N-dimensional inverse discrete Fourier Transform.
- fftshift - Move zeroth lag to center of spectrum.

Sound and audio.

- sound - Play vector as sound.
- soundsc - Autoscale and play vector as sound.
- speak - Convert input string to speech (Macintosh only).
- recordsound - Record sound (Macintosh only).
- soundcap - Sound capabilities (Macintosh only).
- mu2lin - Convert mu-law encoding to linear signal.
- lin2mu - Convert linear signal to mu-law encoding.

Audio file inport/export.

- auwrite - Write NeXT/SUN (".au") sound file.
- auread - Read NeXT/SUN (".au") sound file.
- wavwrite - Write Microsoft WAVE (".wav") sound file.
- wavread - Read Microsoft WAVE (".wav") sound file.
- readsnd - Read SND resources and files (Macintosh only).
- writesnd - Write SND resources and files (Macintosh only).

Interpolation and polynomials.

Data interpolation.

interp1 - 1-D interpolation (table lookup).
interp1q - Quick 1-D linear interpolation.
interpft - 1-D interpolation using FFT method.
interp2 - 2-D interpolation (table lookup).
interp3 - 3-D interpolation (table lookup).
interpn - N-D interpolation (table lookup).
griddata - Data gridding and surface fitting.

Spline interpolation.

spline - Cubic spline interpolation.
ppval - Evaluate piecewise polynomial.

Geometric analysis.

delaunay - Delaunay triangulation.
dsearch - Search Delaunay triangulation for nearest point.
tsearch - Closest triangle search.
convhull - Convex hull.
voronoi - Voronoi diagram.
inpolygon - True for points inside polygonal region.
rectint - Rectangle intersection area.
polyarea - Area of polygon.

Polynomials.

roots - Find polynomial roots.
poly - Convert roots to polynomial.
polyval - Evaluate polynomial.
polyvalm - Evaluate polynomial with matrix argument.
residue - Partial-fraction expansion (residues).
polyfit - Fit polynomial to data.
polyder - Differentiate polynomial.
conv - Multiply polynomials.
deconv - Divide polynomials.

Function functions and ODE solvers.

Optimization and root finding.

- fmin - Minimize function of one variable.
- fmins - Minimize function of several variables.
- fzero - Find zero of function of one variable.

Numerical integration (quadrature).

- quad - Numerically evaluate integral, low order method.
- quad8 - Numerically evaluate integral, higher order method.
- dblquad - Numerically evaluate double integral.

Plotting.

- ezplot - Easy to use function plotter.
- fplot - Plot function.

Inline function object.

- inline - Construct INLINE object.
- argnames - Argument names.
- formula - Function formula.
- char - Convert INLINE object to character array.

Ordinary differential equation solvers.

(If unsure about stiffness, try ODE45 first, then ODE15S.)

- ode45 - Solve non-stiff differential equations, medium order method.
- ode23 - Solve non-stiff differential equations, low order method.
- ode113 - Solve non-stiff differential equations, variable order method.
- ode15s - Solve stiff differential equations, variable order method.
- ode23s - Solve stiff differential equations, low order method.
- odefile - ODE file syntax.

ODE Option handling.

- odeset - Create/alter ODE OPTIONS structure.
- odeget - Get ODE OPTIONS parameters.

ODE output functions.

- odeplot - Time series ODE output function.
- odephas2 - 2-D phase plane ODE output function.
- odephas3 - 3-D phase plane ODE output function.
- odeprint - Command window printing ODE output function.

Sparse matrices.

Elementary sparse matrices.

speye - Sparse identity matrix.
sprand - Sparse uniformly distributed random matrix.
sprandn - Sparse normally distributed random matrix.
sprandsym - Sparse random symmetric matrix.
spdiags - Sparse matrix formed from diagonals.

Full to sparse conversion.

sparse - Create sparse matrix.
full - Convert sparse matrix to full matrix.
find - Find indices of nonzero elements.
spconvert - Import from sparse matrix external format.

Working with sparse matrices.

nnz - Number of nonzero matrix elements.
nonzeros - Nonzero matrix elements.
nzmax - Amount of storage allocated for nonzero matrix elements.
spones - Replace nonzero sparse matrix elements with ones.
spalloc - Allocate space for sparse matrix.
issparse - True for sparse matrix.
spfun - Apply function to nonzero matrix elements.
spy - Visualize sparsity pattern.

Reordering algorithms.

colmmd - Column minimum degree permutation.
symmmd - Symmetric minimum degree permutation.
symrcm - Symmetric reverse Cuthill-McKee permutation.
colperm - Column permutation.
randperm - Random permutation.
dmperm - Dulmage-Mendelsohn permutation.

Linear algebra.

eigs - A few eigenvalues.
svds - A few singular values.
luinc - Incomplete LU factorization.
cholinc - Incomplete Cholesky factorization.
normest - Estimate the matrix 2-norm.
condest - 1-norm condition number estimate.
sprank - Structural rank.

Linear Equations (iterative methods).

pcg - Preconditioned Conjugate Gradients Method.
bicg - BiConjugate Gradients Method.
bicgstab - BiConjugate Gradients Stabilized Method.
cgs - Conjugate Gradients Squared Method.
gmres - Generalized Minimum Residual Method.
qmr - Quasi-Minimal Residual Method.

Operations on graphs (trees).

treelayout - Lay out tree or forest.
treeplot - Plot picture of tree.
etree - Elimination tree.
etreeplot - Plot elimination tree.
gplot - Plot graph, as in "graph theory".

Miscellaneous.

sybfact - Symbolic factorization analysis.
spparms - Set parameters for sparse matrix routines.
spaugment - Form least squares augmented system.

Two dimensional graphs.

Elementary X-Y graphs.

plot - Linear plot.
loglog - Log-log scale plot.
semilogx - Semi-log scale plot.
semilogy - Semi-log scale plot.
polar - Polar coordinate plot.
plotyy - Graphs with y tick labels on the left and right.

Axis control.

axis - Control axis scaling and appearance.
zoom - Zoom in and out on a 2-D plot.
grid - Grid lines.
box - Axis box.
hold - Hold current graph.
axes - Create axes in arbitrary positions.
subplot - Create axes in tiled positions.

Graph annotation.

legend - Graph legend.
title - Graph title.
xlabel - X-axis label.
ylabel - Y-axis label.
text - Text annotation.
gtext - Place text with mouse.

Hardcopy and printing.

print - Print graph or SIMULINK system; or save graph to M-file.
printopt - Printer defaults.
orient - Set paper orientation.

See also GRAPH3D, SPECGRAPH.

Three dimensional graphs.

Elementary 3-D plots.

- plot3 - Plot lines and points in 3-D space.
- mesh - 3-D mesh surface.
- surf - 3-D colored surface.
- fill3 - Filled 3-D polygons.

Color control.

- colormap - Color look-up table.
- caxis - Pseudocolor axis scaling.
- shading - Color shading mode.
- hidden - Mesh hidden line removal mode.
- brighten - Brighten or darken color map.

Lighting.

- surfl - 3-D shaded surface with lighting.
- lighting - Lighting mode.
- material - Material reflectance mode.
- specular - Specular reflectance.
- diffuse - Diffuse reflectance.
- surfnorm - Surface normals.

Color maps.

- hsv - Hue-saturation-value color map.
- hot - Black-red-yellow-white color map.
- gray - Linear gray-scale color map.
- bone - Gray-scale with tinge of blue color map.
- copper - Linear copper-tone color map.
- pink - Pastel shades of pink color map.
- white - All white color map.
- flag - Alternating red, white, blue, and black color map.
- lines - Color map with the line colors.
- colorcube - Enhanced color-cube color map.
- jet - Variant of HSV.
- prism - Prism color map.
- cool - Shades of cyan and magenta color map.
- autumn - Shades of red and yellow color map.
- spring - Shades of magenta and yellow color map.
- winter - Shades of blue and green color map.
- summer - Shades of green and yellow color map.

Axis control.

- axis - Control axis scaling and appearance.
- zoom - Zoom in and out on a 2-D plot.
- grid - Grid lines.
- box - Axis box.
- hold - Hold current graph.
- axes - Create axes in arbitrary positions.
- subplot - Create axes in tiled positions.

Viewpoint control.

- view - 3-D graph viewpoint specification.
- viewmtx - View transformation matrix.
- rotate3d - Interactively rotate view of 3-D plot.

Graph annotation.

- title - Graph title.
- xlabel - X-axis label.
- ylabel - Y-axis label.
- zlabel - Z-axis label.
- colorbar - Display color bar (color scale).
- text - Text annotation.
- gtext - Mouse placement of text.

Hardcopy and printing.

- print - Print graph or SIMULINK system; or save graph to M-file.
- printopt - Printer defaults.
- orient - Set paper orientation.

See also GRAPH3D, SPECGRAPH.

Specialized graphs.

Specialized 2-D graphs.

- area - Filled area plot.
- bar - Bar graph.
- barh - Horizontal bar graph.
- bar3 - 3-D bar graph.
- bar3h - Horizontal 3-D bar graph.
- comet - Comet-like trajectory.
- errorbar - Error bar plot.
- ezplot - Easy to use function plotter.
- feather - Feather plot.
- fill - Filled 2-D polygons.
- fplot - Plot function.
- hist - Histogram.
- pareto - Pareto chart.
- pie - Pie chart.
- pie3 - 3-D pie chart.
- plotmatrix - Scatter plot matrix.
- ribbon - Draw 2-D lines as ribbons in 3-D.
- stem - Discrete sequence or "stem" plot.
- stairs - Stairstep plot.

Contour and 2-1/2 D graphs.

- contour - Contour plot.
- contourf - Filled contour plot.
- contour3 - 3-D Contour plot.
- clabel - Contour plot elevation labels.
- pcolor - Pseudocolor (checkerboard) plot.
- quiver - Quiver plot.
- voronoi - Voronoi diagram.

Specialized 3-D graphs.

- comet3 - 3-D comet-like trajectories.
- meshc - Combination mesh/contour plot.
- meshz - 3-D mesh with curtain.
- stem3 - 3-D stem plot.
- quiver3 - 3-D quiver plot.
- slice - Volumetric slice plot.
- surf - Combination surf/contour plot.
- trisurf - Triangular surface plot.
- trimesh - Triangular mesh plot.
- waterfall - Waterfall plot.

Images display and file I/O.

- image - Display image.
- imagesc - Scale data and display as image.
- colormap - Color look-up table.
- gray - Linear gray-scale color map.
- contrast - Gray scale color map to enhance image contrast.
- brighten - Brighten or darken color map.
- colorbar - Display color bar (color scale).
- imread - Read image from graphics file.
- imwrite - Write image to graphics file.
- imfinfo - Information about graphics file.

Movies and animation.

- capture - Screen capture of current figure.
- moviein - Initialize movie frame memory.
- getframe - Get movie frame.
- movie - Play recorded movie frames.
- qtwrite - Translate movie into QuickTime format (Macintosh only).
- rotate - Rotate object about specified origin and direction.
- frame2im - Convert movie frame to indexed image.
- im2frame - Convert index image into movie format.

Color related functions.

- spinmap - Spin color map.
- rgbplot - Plot color map.
- colstyle - Parse color and style from string.

Solid modeling.

- cylinder - Generate cylinder.
- sphere - Generate sphere.
- patch - Create patch.

Handle Graphics.

Figure window creation and control.

figure - Create figure window.
gcf - Get handle to current figure.
clf - Clear current figure.
shg - Show graph window.
close - Close figure.
refresh - Refresh figure.

Axis creation and control.

subplot - Create axes in tiled positions.
axes - Create axes in arbitrary positions.
gca - Get handle to current axes.
cla - Clear current axes.
axis - Control axis scaling and appearance.
box - Axis box.
caxis - Control pseudocolor axis scaling.
hold - Hold current graph.
ishold - Return hold state.

Handle Graphics objects.

figure - Create figure window.
axes - Create axes.
line - Create line.
text - Create text.
patch - Create patch.
surface - Create surface.
image - Create image.
light - Create light.
uicontrol - Create user interface control.
uimenu - Create user interface menu.

Handle Graphics operations.

set - Set object properties.
get - Get object properties.
reset - Reset object properties.
delete - Delete object.
gco - Get handle to current object.
gcbo - Get handle to current callback object.
gcbf - Get handle to current callback figure.
drawnow - Flush pending graphics events.
findobj - Find objects with specified property values.
copyobj - Make copy of graphics object and its children.

Hardcopy and printing.

print - Print graph or SIMULINK system; or save graph to M-file.
printopt - Printer defaults.
orient - Set paper orientation.

Utilities.

closereq - Figure close request function.
newplot - M-file preamble for NextPlot property.
ishandle - True for graphics handles.

See also GRAPH2D, GRAPH3D, SPECGRAPH.

Graphical user interface tools.

GUI functions.

- uicontrol - Create user interface control.
- uimenu - Create user interface menu.
- ginput - Graphical input from mouse.
- dragrect - Drag XOR rectangles with mouse.
- rbbox - Rubberband box.
- selectmoveresize - Interactively select, move, resize, or copy objects.
- waitforbuttonpress - Wait for key/buttonpress over figure.
- waitfor - Block execution and wait for event.
- uiwait - Block execution and wait for resume.
- uiresume - Resume execution of blocked M-file.

GUI design tools.

- guide - Design GUI.
- align - Align uicontrols and axes.
- cbedit - Edit callback.
- menuedit - Edit menu.
- propedit - Edit property.

Dialog boxes.

- dialog - Create dialog figure.
- axlimdlg - Axes limits dialog box.
- errordlg - Error dialog box.
- helpdlg - Help dialog box.
- inputdlg - Input dialog box.
- listdlg - List selection dialog box.
- menu - Generate menu of choices for user input.
- msgbox - Message box.
- questdlg - Question dialog box.
- warnrdlg - Warning dialog box.
- uigetfile - Standard open file dialog box.
- uiputfile - Standard save file dialog box.
- uisetcolor - Color selection dialog box.
- uisetfont - Font selection dialog box.
- pagedlg - Page position dialog box.
- printdlg - Print dialog box.
- waitbar - Display wait bar.

Menu utilities.

- makemenu - Create menu structure.
- menubar - Computer dependent default setting for MenuBar property.
- umtogggle - Toggle "checked" status of uimenu object.
- windows - Create submenu for "Window" menu item.

Toolbar button group utilities.

- btngroup - Create toolbar button group.
- btnstate - Query state of toolbar button group.
- btnpress - Button press manager for toolbar button group.
- btndown - Depress button in toolbar button group.
- btnup - Raise button in toolbar button group.

User-defined figure/axes property utilities.

- clrprop - Clear user-defined property.
- getprop - Get value of user-defined property.
- setupprop - Set user-defined property.

Miscellaneous utilities.

- allchild - Get all object children.
- hidegui - Hide/unhide GUI.
- edtext - Interactive editing of axes text objects.
- getstatus - Get status text string in figure.
- setstatus - Set status text string in figure.
- popupstr - Get popup menu selection string.
- remapfig - Transform figure objects' positions.
- setptr - Set figure pointer.
- getptr - Get figure pointer.
- overobj - Get handle of object the pointer is over.

Character strings.

General.

- char - Create character array (string).
- double - Convert string to numeric character codes.
- cellstr - Create cell array of strings from character array.
- blanks - String of blanks.
- deblank - Remove trailing blanks.
- eval - Execute string with MATLAB expression.

String tests.

- ischar - True for character array (string).
- iscellstr - True for cell array of strings.
- isletter - True for letters of the alphabet.
- isspace - True for white space characters.

String operations.

- strcat - Concatenate strings.
- strvcat - Vertically concatenate strings.
- strcmp - Compare strings.
- strncmp - Compare first N characters of strings.
- findstr - Find one string within another.
- strjust - Justify character array.
- strmatch - Find possible matches for string.
- strrep - Replace string with another.
- strtok - Find token in string.
- upper - Convert string to uppercase.
- lower - Convert string to lowercase.

String to number conversion.

- num2str - Convert number to string.
- int2str - Convert integer to string.
- mat2str - Convert matrix to eval'able string.
- str2num - Convert string to number.
- sprintf - Write formatted data to string.
- sscanf - Read string under format control.

Base number conversion.

- hex2num - Convert IEEE hexadecimal to double precision number.
- hex2dec - Convert hexadecimal string to decimal integer.
- dec2hex - Convert decimal integer to hexadecimal string.
- bin2dec - Convert binary string to decimal integer.
- dec2bin - Convert decimal integer to binary string.
- base2dec - Convert base B string to decimal integer.
- dec2base - Convert decimal integer to base B string.

See also STRINGS.

File input/output.

File opening and closing.

fopen - Open file.
fclose - Close file.

Binary file I/O.

fread - Read binary data from file.
fwrite - Write binary data to file.

Formatted file I/O.

fscanf - Read formatted data from file.
fprintf - Write formatted data to file.
fgetl - Read line from file, discard newline character.
fgets - Read line from file, keep newline character.
input - Prompt for user input.

String conversion.

sprintf - Write formatted data to string.
sscanf - Read string under format control.

File positioning.

ferror - Inquire file error status.
feof - Test for end-of-file.
fseek - Set file position indicator.
ftell - Get file position indicator.
frewind - Rewind file.

File name handling

matlabroot - Root directory of MATLAB installation.
filesep - Directory separator for this platform.
pathsep - Path separator for this platform.
mexext - MEX filename extension for this platform.
fullfile - Build full filename from parts.
partialpath - Partial pathnames.
tempdir - Get temporary directory.
tempname - Get temporary file.

File import/export functions.

load - Load workspace from MAT-file.
save - Save workspace to MAT-file.
dlmread - Read ASCII delimited file.
dlmwrite - Write ASCII delimited file.
wklread - Read spreadsheet (WK1) file.
wklwrite - Write spreadsheet (WK1) file.

Image file import/export.

imread - Read image from graphics file.
imwrite - Write image to graphics file.
iminfo - Return information about graphics file.

Audio file import/export.

auwrite - Write NeXT/SUN (.au) sound file.
auread - Read NeXT/SUN (.au) sound file.
wavwrite - Write Microsoft WAVE (.wav) sound file.
wavread - Read Microsoft WAVE (.wav) sound file.

Command window I/O

clc - Clear command window.
home - Send cursor home.
disp - Display array.
input - Prompt for user input.
pause - Wait for user response.

Time and dates.

Current date and time.

now - Current date and time as date number.
date - Current date as date string.
clock - Current date and time as date vector.

Basic functions.

datenum - Serial date number.
datestr - String representation of date.
datevec - Date components.

Date functions.

calendar - Calendar.
weekday - Day of week.
eomday - End of month.
datetick - Date formatted tick labels.

Timing functions.

cputime - CPU time in seconds.
tic, toc - Stopwatch timer.
etime - Elapsed time.
pause - Wait in seconds.

Data types and structures.

Data types (classes)

double - Convert to double precision.
sparse - Create sparse matrix.
char - Create character array (string).
cell - Create cell array.
struct - Create or convert to structure array.
uint8 - Convert to unsigned 8-bit integer.
inline - Construct INLINE object.

Multi-dimensional array functions.

cat - Concatenate arrays.
ndims - Number of dimensions.
ndgrid - Generate arrays for N-D functions and interpolation.
permute - Permute array dimensions.
ipermute - Inverse permute array dimensions.
shiftdim - Shift dimensions.
squeeze - Remove singleton dimensions.

Cell array functions.

cell - Create cell array.
celldisp - Display cell array contents.
cellplot - Display graphical depiction of cell array.
num2cell - Convert numeric array into cell array.
deal - Deal inputs to outputs.
cell2struct - Convert cell array into structure array.
struct2cell - Convert structure array into cell array.
iscell - True for cell array.

Structure functions.

struct - Create or convert to structure array.
fieldnames - Get structure field names.
getfield - Get structure field contents.
setfield - Set structure field contents.
rmfield - Remove structure field.
isfield - True if field is in structure array.
isstruct - True for structures.

Object oriented programming functions.

class - Create object or return object class.
struct - Convert object to structure array.
methods - Display class method names.
isa - True if object is a given class.
isobject - True for objects.
inferiorto - Inferior class relationship.
superiorto - Superior class relationship.

Overloadable operators.

minus - Overloadable method for a-b.
plus - Overloadable method for a+b.
times - Overloadable method for a.*b.
mtimes - Overloadable method for a*b.
mldivide - Overloadable method for a\b.
mrdivide - Overloadable method for a/b.
rdivide - Overloadable method for a./b.
ldivide - Overloadable method for a.\b.
power - Overloadable method for a.^b.
mpower - Overloadable method for a^b.
uminus - Overloadable method for -a.
uplus - Overloadable method for +a.
horzcat - Overloadable method for [a b].
vertcat - Overloadable method for [a;b].
le - Overloadable method for a<=b.
lt - Overloadable method for a<b.
gt - Overloadable method for a>b.
ge - Overloadable method for a>=b.
eq - Overloadable method for a==b.
ne - Overloadable method for a~=b.
not - Overloadable method for ~a.
and - Overloadable method for a&b.
or - Overloadable method for a|b.
subsasgn - Overloadable method for a(i)=b, a(i)=b, and a.field=b.
subsref - Overloadable method for a(i), a{i}, and a.field.
colon - Overloadable method for a:b.
transpose - Overloadable method for a.'
ctranspose - Overloadable method for a'
subsindex - Overloadable method for x(a).

Signal Processing Toolbox.

Version 4.0 Student Edition 31-Dec-1996

What's new.

Readme - New features, bug fixes, and changes in this version.

Waveform generation.

chirp - Swept-frequency cosine generator.
diric - Dirichlet (periodic sinc) function.
gauspuls - Gaussian pulse generator.
pulstran - Pulse train generator.
rectpuls - Sampled aperiodic rectangle generator.
sawtooth - Sawtooth function.
sinc - Sinc or $\sin(\pi x)/(\pi x)$ function
square - Square wave function.
tripuls - Sampled aperiodic triangle generator.

Filter analysis and implementation.

abs - Magnitude.
angle - Phase angle.
casfilt - Cascade filter implementation.
conv - Convolution.
fftfilt - Overlap-add filter implementation.
filter - Filter implementation.
filtfilt - Zero-phase version of filter.
filtic - Determine filter initial conditions.
freqs - Laplace transform frequency response.
freqspace - Frequency spacing for frequency response.
freqz - Z-transform frequency response.
grpdelay - Group delay.
impz - Impulse response (discrete).
latcfilt - Lattice filter implementation.
unwrap - Unwrap phase.
upfirdn - Up sample, FIR filter, down sample.
zplane - Discrete pole-zero plot.

Linear system transformations.

convmtx - Convolution matrix.
latc2tf - Lattice or lattice ladder to transfer function conversion.
poly2rc - Polynomial to reflection coefficients transformation.
rc2poly - Reflection coefficients to polynomial transformation.
residuez - Z-transform partial fraction expansion.
sos2ss - Second-order sections to state-space conversion.
sos2tf - Second-order sections to transfer function conversion.
sos2zp - Second-order sections to zero-pole conversion.
ss2sos - State-space to second-order sections conversion.
ss2tf - State-space to transfer function conversion.
ss2zp - State-space to zero-pole conversion.
tf2latc - Transfer function to lattice or lattice ladder conversion.
tf2ss - Transfer function to state-space conversion.
tf2zp - Transfer function to zero-pole conversion.
zp2sos - Zero-pole to second-order sections conversion.
zp2ss - Zero-pole to state-space conversion.
zp2tf - Zero-pole to transfer function conversion.

IIR digital filter design.

butter - Butterworth filter design.
cheby1 - Chebyshev type I filter design.
cheby2 - Chebyshev type II filter design.
ellip - Elliptic filter design.
maxflat - Generalized Butterworth lowpass filter design.
yulewalk - Yule-Walker filter design.

IIR filter order selection.

buttord - Butterworth filter order selection.
cheblord - Chebyshev type I filter order selection.
cheb2ord - Chebyshev type II filter order selection.
ellipord - Elliptic filter order selection.

FIR filter design.

cremez - Complex and nonlinear phase equiripple FIR filter design.
firl - Window based FIR filter design - low, high, band, stop, multi.
fir2 - Window based FIR filter design - arbitrary response.
fircls - Constrained Least Squares filter design - arbitrary response.
fircls1 - Constrained Least Squares FIR filter design - low and highpass.
firls - FIR filter design - arbitrary response with transition bands.
firrcos - Raised cosine FIR filter design.
intfilt - Interpolation FIR filter design.
kaiserord - Window based filter order selection using Kaiser window.
remez - Parks-McClellan optimal FIR filter design.

remezord - Parks-McClellan filter order selection.

Transforms.

- cztf - Chirp-z transform.
- dct - Discrete cosine transform.
- dftmtx - Discrete Fourier transform matrix.
- fft - Fast Fourier transform.
- fftshift - Swap vector halves.
- hilbert - Hilbert transform.
- idct - Inverse discrete cosine transform.
- ifft - Inverse fast Fourier transform.

Statistical signal processing and spectral analysis.

- cohere - Coherence function estimate.
- corrcoef - Correlation coefficients.
- cov - Covariance matrix.
- csd - Cross Spectral Density.
- pmem - Power Spectrum estimate via MEM (Maximum Entropy Method).
- pmtm - Power Spectrum estimate via the Thomson multitaper method.
- pmusic - Power Spectrum estimate via MUSIC eigenvector method.
- psd - Power Spectral Density.
- spectrum - psd, csd, cohere and tfe combined.
- tfe - Transfer function estimate.
- xcorr - Cross-correlation function.
- xcov - Covariance function.

Windows.

- bartlett - Bartlett window.
- blackman - Blackman window.
- boxcar - Rectangular window.
- chebwin - Chebyshev window.
- hamming - Hamming window.
- hanning - Hanning window.
- kaiser - Kaiser window.
- triang - Triangular window.

Parametric modeling.

- invfreqs - Analog filter fit to frequency response.
- invfreqz - Discrete filter fit to frequency response.
- levinson - Levinson-Durbin recursion.
- lpc - Linear Predictive Coefficients using autocorrelation method.
- prony - Prony's discrete filter fit to time response.
- stmcb - Steiglitz-McBride iteration for ARMA modeling.
- ident - See the System Identification Toolbox.

Specialized operations.

- cceps - Complex cepstrum.
- decimate - Resample data at a lower sample rate.
- deconv - Deconvolution.
- demod - Demodulation for communications simulation.
- dpss - Discrete prolate spheroidal sequences (Slepian sequences).
- dpsscload - Remove discrete prolate spheroidal sequences from database.
- dpssdir - Discrete prolate spheroidal sequence database directory.
- dpssload - Load discrete prolate spheroidal sequences from database.
- dpsssave - Save discrete prolate spheroidal sequences in database.
- interp - Resample data at a higher sample rate.
- interp1 - General 1-D interpolation. (MATLAB Toolbox)
- medfilt1 - 1-Dimensional median filtering.
- modulate - Modulation for communications simulation.
- rceps - Real cepstrum and minimum phase reconstruction.
- resample - Resample sequence with new sampling rate.
- specgram - Spectrogram, for speech signals.
- spline - Cubic spline interpolation.
- vco - Voltage controlled oscillator.

Analog lowpass filter prototypes.

- besselap - Bessel filter prototype.
- buttap - Butterworth filter prototype.
- cheblap - Chebyshev type I filter prototype (passband ripple).
- cheb2ap - Chebyshev type II filter prototype (stopband ripple).
- ellipap - Elliptic filter prototype.

Frequency translation.

- lp2bp - Lowpass to bandpass analog filter transformation.
- lp2bs - Lowpass to bandstop analog filter transformation.
- lp2hp - Lowpass to highpass analog filter transformation.
- lp2lp - Lowpass to lowpass analog filter transformation.

Filter discretization.

- bilinear - Bilinear transformation with optional prewarping.
- impinvar - Impulse invariance analog to digital conversion.

Other.

besself - Bessel analog filter design.
conv2 - 2-D convolution.
cplxpair - Order vector into complex conjugate pairs.
detrrend - Linear trend removal.
fft2 - 2-D fast Fourier transform.
fftshift - Swap quadrants of array.
ifft2 - Inverse 2-D fast Fourier transform.
polystab - Polynomial stabilization.
stem - Plot discrete data sequence.
strips - Strip plot.
xcorr2 - 2-D cross-correlation.

Signal GUI (Graphical User Interface).

sptool - Signal Processing Tool interface.

Demonstrations.

cztdemo - Chirp-z transform and FFT demonstration.
filtdemo - Filter design demonstration.
moddemo - Modulation/demodulation demonstration.
sosdemo - Second-order sections demonstration.

Symbolic Math Toolbox.

Version 2.0 Student Edition 18-Feb-1997

Calculus.

diff - Differentiate.
int - Integrate.
limit - Limit.
taylor - Taylor series.
jacobian - Jacobian matrix.
symsum - Summation of series.

Linear Algebra.

diag - Create or extract diagonals.
triu - Upper triangle.
tril - Lower triangle.
inv - Matrix inverse.
det - Determinant.
rank - Rank.
rref - Reduced row echelon form.
null - Basis for null space.
colspace - Basis for column space.
eig - Eigenvalues and eigenvectors.
svd - Singular values and singular vectors.
jordan - Jordan canonical (normal) form.
poly - Characteristic polynomial.
expm - Matrix exponential.

Simplification.

simplify - Simplify.
expand - Expand.
factor - Factor.
collect - Collect.
simple - Search for shortest form.
numden - Numerator and denominator.
horner - Nested polynomial representation.
subexpr - Rewrite in terms of subexpressions.
subs - Symbolic substitution.

Solution of Equations.

solve - Symbolic solution of algebraic equations.
dsolve - Symbolic solution of differential equations.
finverse - Functional inverse.
compose - Functional composition.

Variable Precision Arithmetic.

vpa - Variable precision arithmetic.
digits - Set variable precision accuracy.

Integral Transforms.

fourier - Fourier transform.
laplace - Laplace transform.
ztrans - Z transform.
ifourier - Inverse Fourier transform.
ilaplace - Inverse Laplace transform.
iztrans - Inverse Z transform.

Conversions.

double - Convert symbolic matrix to double.
poly2sym - Coefficient vector to symbolic polynomial.
sym2poly - Symbolic polynomial to coefficient vector.
char - Convert sym object to string.

Basic Operations.

sym - Create symbolic object.
syms - Short-cut for constructing symbolic objects.
findsym - Determine symbolic variables.
pretty - Pretty print a symbolic expression.
latex - LaTeX representation of a symbolic expression.
ccode - C code representation of a symbolic expression.
fortran - Fortran representation of a symbolic expression.

Special Functions.

sinint - Sine integral.
cosint - Cosine integral.
zeta - Riemann zeta function.
lambertw - Lambert W function.

String handling utilities.

isvarname - Check for a valid variable name.
vectorize - Vectorize a symbolic expression.

Pedagogical and Graphical Applications.

- rsums - Riemann sums.
- ezplot - Easy to use function plotter.
- funtool - Function calculator.

Demonstrations.

- symintro - Introduction to the Symbolic Toolbox.
- symcalcdemo - Calculus demonstration.
- symlindemo - Demonstrate symbolic linear algebra.
- symvpdemo - Demonstrate variable precision arithmetic.
- symrotdemo - Study plane rotations.
- symeqndemo - Demonstrate symbolic equation solving.

Access to Maple. (Not available with Student Edition.)

- maple - Access Maple kernel.
- mfun - Numeric evaluation of Maple functions.
- mfunlist - List of functions for MFUN.
- mhhelp - Maple help.
- procread - Install a Maple procedure. (Requires Extended Toolbox.)


```
function f=fixdivbyzero(x)
% FIXDIBYZERO replaces zero values by eps to avoid div by zero errors
% f=fixdivbyzero(x) returns eps if x=0; otherwise it simply returns x

f=x+(x==0)*eps;
```

```
function blank(bnum)
%%%BLANK displays blank lines
%%% BLANK displays one blankline
%%% BLANK(bnum) displays bnum number of blank lines

if nargin == 0, bnum=1; end;

for n=1:bnum;
    disp(' ');
end;
```

```
function r=rad(degrees)
% RAD converts degrees to radians
% r=RAD(degrees) returns the number in degrees
%
r=pi*degrees/180;
```

```
function d=rad2deg(radians)
% RAD2DEG converts radians to degrees
% d=RAD2DEG(radians) returns the number in degrees
%
d=180*radians/(2*pi);
```

```

function new = changemenu(names,old)
% CHANGEMENU generates a menu to change various parameters
% new = ROUNDTRIPABCD(name_cell_array,old) takes a 1xn array of n variables and a 1xn cell array
% containing information for the labels of the menu, asks for user input through the menu, and
outputs
% the 1xn array of new variables.
% typical input : names = {'Distance to Mirror 1', 'Focal Length of Mirror 1', 'Distance to Mirror 2'}
% old = {500, 300,400,0,0,0}
% (the last three numbers represent the last changed field, the change, and the amount of the change)
% exit code is given by 999 in the (last changed field) value
% written by Phil Tsai 11-03-99
% last revision 11-04-99
%
```

```

lastchoice1 = old(1,4);
lastchoice2 = old(1,5);
lastchange=old(1,6);
new = old;
changeby = 0;
changeto = 0;
choice1 = 0;
choice2 = lastchoice2;
```

```

choicelist{1} = 'Pick a parameter to change';
choicelist{2} = 'Repeat Last';
```

```

[r,c] = size(old);
num_elements = c-3;
```

```

for j=1:num_elements
    choicelist{j+2} = char(names(1,j));
end
```

```

choicelist{num_elements+3} = 'Exit Program';
choice1 = strmenu(choicelist);
```

```

% if choice1 = 1 then repeat
% if choice1 = (num_elements+2) then return lastchoice1=999 exit code
```

```

if choice1==num_elements+2, new(1,num_elements+1)=999; return; end;
```

```

if choice1~=1
    choice2 = menu('Pick one:','Change to..','Increase by..','Decrease by..');
end
```

```

if choice2 ==1
    changeto = input('Change to : ');
else
    if choice1~=1 & choice2==2, changeby = input('Increase by: '); end;
    if choice1~=1 & choice2==3, changeby = input('Decrease by: '); end;
end
```

```
if isempty(changeto); changeto=0; end;  
if isempty(changeby);changeby=0;end;
```

```
if choice1==1  
    if lastchoice1 == 0 ; new = old; return; end;  
    if lastchoice2 == 0; new = old; return; end;  
    choice1 = lastchoice1;  
    choice2 = lastchoice2;  
    changeby = lastchange;  
end
```

```
if choice2 == 1  
    if changeto == 0, changeto = inf ; end;  
    new(1,choice1-1) = changeto;  
end
```

```
if choice2 == 2,    new(1,choice1-1) = old(1,choice1-1) + changeby;    end  
if choice2 == 3,    new(1,choice1-1) = old(1,choice1-1) - changeby;    end
```

```
new(1,num_elements+1)=choice1;  
new(1,num_elements+2)=choice2;  
new(1,num_elements+3)=changeby;
```

```
function value = request(question,default_value,minv,maxv)
% REQUEST(question,default_value,min,max) : input prompt with default value
% value = request(question,default_value) prompts the user with question, and shows
% the default value. If the user hits return without entering an input,
% the function will return default_value as the output value. Otherwise, the user
% input is returned.
% If the minv and maxv are provided, then the function checks to see if the
% input is within the specified range; if not, either maxv or minv is returned

reqstring = [question,' [default = ',num2str(default_value),'] -> '];
temp_value = input(reqstring);

if isempty(temp_value)
    value = default_value;
    return;
end

if nargin < 3
    value = temp_value;
    return;
end

if nargin == 3; maxv = inf; end;

if temp_value > maxv
    value = maxv;
    disp([' WARNING, the requested number must be less than ',num2str(maxv)]);
    disp([' the value of this input has been set to ',num2str(maxv)]);
    return;
end

if temp_value < minv
    value = minv;
    disp([' WARNING, the requested number must be greater than ',num2str(minv)]);
    disp([' the value of this input has been set to ',num2str(minv)]);
    return;
end

value = temp_value;
```

```
function k = strmenu(myinput);
%STRMENU Generate a menu of choices for user input (in form of cell array).
% K = MENU(cell_array) displays a menu with cell_array(1) as header
% and cell_array(2), etc as choices
%
% cell array should be a 1xn cell array
% created as follows:
% cell_array{1} = 'Choose a color'
% cell_array{2} = '1) Red'
% cell_array{3} = '1) Blue'
% cell_array{4} = '1) Green'
%
% will produce the following menu on the screen:
%
% ----- Choose a color -----
%
% 1) Red
% 2) Blue
% 3) Green
%
% Select a menu number:
%
% The number entered by the user in response to the prompt is
% returned. On machines that support it, the local menu system
% is used. The maximum number of menu items is 32.
%
% See also MENU.

% J.N. Little 4-21-87, revised 4-13-92 by LS.
% Copyright (c) 1984-96 by The MathWorks, Inc.
% $Revision: 5.7 $ $Date: 1996/05/09 18:48:57 $
% Modified by Phil Tsai 99/11/03 to accept cell array
```

```
[myrows,mycols]=size(myinput);
myinput{33}='end';
s0=char(myinput(1));
s1=char(myinput(2));
if mycols>2; s2=char(myinput(3)); end;
if mycols>3; s3=char(myinput(4)); end;
if mycols>4; s4=char(myinput(5)); end;
if mycols>5; s5=char(myinput(6)); end;
if mycols>6; s6=char(myinput(7)); end;
if mycols>7; s7=char(myinput(8)); end;
if mycols>8; s8=char(myinput(9)); end;
if mycols>9; s9=char(myinput(10)); end;
if mycols>10; s10=char(myinput(11)); end;
if mycols>11; s11=char(myinput(12)); end;
if mycols>12; s12=char(myinput(13)); end;
if mycols>13; s13=char(myinput(14)); end;
if mycols>14; s14=char(myinput(15)); end;
if mycols>15; s15=char(myinput(16)); end;
if mycols>16; s16=char(myinput(17)); end;
```



```

if mycols>17; s17=char(myinput(18)); end;
if mycols>18; s18=char(myinput(19)); end;
if mycols>19; s19=char(myinput(20)); end;
if mycols>20; s20=char(myinput(21)); end;
if mycols>21; s21=char(myinput(22)); end;
if mycols>22; s22=char(myinput(23)); end;
if mycols>23; s23=char(myinput(24)); end;
if mycols>24; s24=char(myinput(25)); end;
if mycols>25; s25=char(myinput(26)); end;
if mycols>26; s26=char(myinput(27)); end;
if mycols>27; s27=char(myinput(28)); end;
if mycols>28; s28=char(myinput(29)); end;
if mycols>29; s29=char(myinput(30)); end;
if mycols>30; s30=char(myinput(31)); end;
if mycols>31; s31=char(myinput(32)); end;
if mycols>32; s32=char(myinput(33)); end;

c = computer;
display = 1;
PC = strcmp(c(1:2),'PC');
if ~strcmp(c(1:2),'PC') & ~strcmp(c(1:2),'MA')
% might be unix or VMS
    if isunix
        display = length(getenv('DISPLAY')) > 0;
    else
        display = length(getenv('DECW$DISPLAY')) > 0;
    end
end
if ~display
    while 1,
        disp(' ')
        disp(['----- ',s0,' -----'])
        disp(' ')
        for n=1:(mycols-1)
            disp([' ',int2str(n),' ',eval(['s',int2str(n)])])
        end
        disp(' ')
        k = input('Select a menu number: ');
        if isempty(k), k = -1; end;
        if (k < 1) | (k > mycols - 1) | (~isreal(k)) | (isnan(k)) | isinf(k),
            disp(' ')
            disp('Selection out of range. Try again.')
        else
            return
        end
    end
end
end

kids = get(0,'Children');
.f ~isempty(kids)
    otherfig =(gcf);
    M=get(otherfig,'Colormap');
else

```

```

M=get(0,'DefaultFigureColormap');
end

xedge = 30;
yedge = 35;
ybord = 30;
width = 30;
avwidth = 7; % actually 6.8886 +/- 0.4887
height = 30;
imax = 1;
maxlen = length(s0);
for i = 1:mycols-1
    mx = length(eval(['s',int2str(i)]));
    if mx > maxlen
        maxlen = mx;
        imax = i;
    end
end
end
twidth = 1.2*maxlen*avwidth;
% now figure out total dimensions needed so things can get placed in pixels
mwwidth = twidth + width + 2*xedge;
mwheight = (mycols+1)*yedge;
ss = get(0,'ScreenSize');
swidth = ss(3); sheight = ss(4);
%left = (swidth-mwwidth)/2;
eft = 20;
bottom = sheight-mwheight-ybord;
rect = [left bottom mwwidth mwheight];
fig = figure('units','pixels','Position',rect,'number','off','name',' ', ...
    'resize','off','Colormap',M);
set(gca,'units','normalized','Position',[0 0 1 1]); axis off;
% Place title
t = text(mwwidth/2,mwheight-yedge/2,s0,'Horizontal','center',...
    'Vertical','top','units','pixels');
for ii=(mycols-1):-1:1
    i = mycols - ii;
    h1 = uicontrol('units','pixels','position', ...
        [xedge (i-.5)*yedge width+twidth height]);
    set(h1,'callback',['set(gcf,"userdata",int2str(ii))']);
    set(h1,'string',[' ', eval(['s',int2str(ii)])])
    set(h1,'HorizontalAlignment','left');
% left justify string inside button
end
waitfor(gcf,'userdata')
k = get(gcf,'userdata');
delete(fig)
if ~isempty(kids)
    ch = get(0,'children');
    if ~isempty(ch),
        if ~isempty(find(ch == otherfig)), % Make sure figure is there
            if strcmp(get(otherfig,'Visible'),'on')
                set(0,'CurrentFigure',otherfig);
            end
        end
    end
end

```

end
end
end

```
function w = waveform(frequencies,amplitudes,centers,startp,endp,pstep,index)
% WAVEFORM produces the shape of a dispersed wavepacket
% w =
waveform(frequencies,amplitudes,centers,start_pos,end_pos,pos_step_size)
% returns a 2 x [(end_pos-start_pos)/po_step_size]
% array of positions and interfered amplitudes based on an input of
% frequencies, their amplitudes, and their centers (phase information)
% the desired start,end and
% (each input is a 1 x n array)
%
% written by Phil Tsai on 11-06-99
% last revision : 11-06-99
%

temp(1,1:(fix((endp - startp)/pstep))+1) = startp:pstep:endp;
temp(2,1:(fix((endp - startp)/pstep))+1)=0;
k = frequencies(1,:) * index / 3e8;

[ row,num_freqs] = size(frequencies);
for j = 1:num_freqs
    temp(2,:) = temp(2,:) + cos(k(1,j).*(temp(1,:)-centers(1,j)));
end;

w = temp;
```

```
%testmatrix.m
```

```
% this program simply generates some usable test matrices
```

```
cell_array =      { 'm' , 3 , 0 , 0 ;...  
                   'p' , 5 , 0 , 0 ;...  
                   'l' , 8 , 0 , 0 ;...  
                   'i' , 1.5 , 1.4 , 0 ;...  
                   'c' , 1.5 , 1.4 , 3 }
```

```
freq_amps = gaussian(1,20,0.2);
```

```
freq_amps(1,:) = freq_amps(1,:) *1e12 + 7.5e14
```

```
[ r,c] = size(freq_amps);
```

```
for j = 1:c;
```

```
    centers(1,j) = 0;
```

```
end;
```

```
function qo=stablecavity(roundtrip_ABCD_matrix)
% STABLECAVITY returns the q_parameter for a standing wave in a cavity given the ABCD matrix
% qo = STABLECAVITY(roundtrip_ABCD_matrix) yields an imaginary number output for a 2x2 matrix
input
% for a round trip in the cavity.
% calculation is based on assumption that the q-parameter will be unchanged after a roundtrip
% THIS PROGRAM ASSUMES AN EMPTY CAVITY WITH NO LASER ROD
% written by Phil Tsai 11-03-99
% last revision 11-04-99

RTA = roundtrip_ABCD_matrix(1,1);
RTB = roundtrip_ABCD_matrix(1,2);
RTC = roundtrip_ABCD_matrix(2,1);
RTD = roundtrip_ABCD_matrix(2,2);

q_parameter = ((RTA-RTD) + sqrt(((RTD-RTA) ^2) + 4*RTB*RTC)) / (2*RTC);

                                %the correct sign in the quadratic formula
if imag(1/q_parameter) > 0;    % must be chosen to yield a real spot size
    q_parameter = ((RTA-RTD) - sqrt(((RTD-RTA) ^2) + 4*RTB*RTC)) / (2*RTC);
end

if imag(1/q_parameter) > 0;    %q(1) should now yield a real spot size,
    blank(3);                  % but just in case....
    disp ('WAIT! SOMETHING IS WRONG! q(1) yields imaginary spot size!');
    blank(3);
end

qo = q_parameter;
```

```
function [spot_size,radius_of_curvature] = spotsize(q_parameter,wavelength,ambient_index)
% SPOTSIZE calculates the spotsize(half-diameter) and radius of curvature of a gaussian beam
% [spot,radius]=SPOTSIZE(q_parameter,wavelength,index) gives the spot size and radius of
% curvature for a gaussian beam with the input of the q-parameter, the wavelength of light,
% and the index of the propagation medium
% written by Phil Tsai 11-04-99
% last revision 11-10-99

temp = imag(1./q_parameter);
temp=fixdivbyzero(temp);
temp=(-1*wavelength)./(temp*pi*ambient_index);
spot_size=sqrt(temp);

if nargout<2; return; end

temp = real(1./q_parameter);
temp = fixdivbyzero(temp);
radius_of_curvature = 1./temp;
```

```
function q = qparam(spotsize, radius, wavelength, index)
% QPARAM calculates the q-parameter of a gaussian beam of given spotsize(half-diameter) and radius
% q = QPARAM(spotsize,radius_curvature,wavelength,index) returns the q_parameter for a gaussian
% beam at the point where it has the inputted spotsize & radius_curvature for the given wavelength
% and ambient index of propagation
% written by Phil Tsai 11-10-99
% last revision 11-10-99
%

term1 = 1/radius;
term2 = wavelength / (pi*index*spotsize^2);
q = 1/ (term1 - i * term2);
```



```
function M=optics(type,param1,param2,param3)
% OPTICS is an alias for function OPTIC
% written by Phil Tsai 11-01-99
% last revision 11-04-99
%

if nargin == 4, M=optic(type,param1,param2,param3); end
if nargin == 3, M=optic(type,param1,param2); end
if nargin == 2, M=optic(type,param1); end
if nargin <2, optic; end
```

```
% beamdivergence.m outputs the beam divergences for a given spot size and a range of radii of curvature
%
% written by Phil Tsai 11-10-99
% last revision 11-10-99
%

spot = request('Enter half-diameter spotsize(in mm) at output of laser',0.35);
start_radius = request('Enter starting radius of curvature (in mm)',spot*1000);
stop_radius = request('Enter stopping radius of curvature (in mm)',spot*2000);
step_size = request('Enter step size of radius (in mm)',spot*10);
distance = request('Enter distance(in mm) over which to evaluate beam',10);
wavelength = request('Enter wavelength of laser (IN NANOMETERS)',800);
wavelength = wavelength/1e6;
index = request('Enter index of propagation medium',1);

clear divergence;
count = 1;
for r = start_radius:step_size:stop_radius;
    term1 = 1/r;
    term2 = -i*wavelength/(pi*index*spot^2);
    qo = 1/(term1+term2);
    q1 = qo + distance;
    temp = -1*imag(1/q1);
    spot2 = 1/(sqrt(temp*pi*index/wavelength));
    y = spot2-spot;
    angle = atan(y/distance);
    divergence(1,count) = r;
    divergence(2,count) = angle*1000;
    count=count+1;
end

plot(divergence(1,:),divergence(2,:))
xlabel('radius of curvature (in mm) at laser output');
ylabel('half-angle beam divergence at laser output in milliradians');
grid;
```

```

function [rt,indiv] = roundtripabcd(cell_array)
% ROUNDTRIPABCD generated the ABCD ray matrix for a round trip in a laser cavity
%   rt = ROUNDTRIPABCD(cell_array) takes a nx4 cell_array containing information for the
%       sequential ABCD matrices for the optics and the propagation through the cavity
%   [rt,indiv] =ROUNDTRIPABCD(cell array) returns both the roundtrip ABCD matrix
%       and the individual ABCD matrices in a 3-dim'l array
%
%   cell_array = {'m', 3 , 0 , 0      <- mirror of radius of curvature = +3
%                 'p', 5 , 0 , 0      <- propagation through distance = 5
%                 'l', 8 , 0 , 0      <- lens of focal length +8
%                 'i', 1.5 , 1.4 , 0   <- planar interface between n=1.5 & n=1.4
%                 'c', 1.5 , 1.4 , 3 } <- spherical interface between n=1.5 & n=1.4
%                                     and radius of curvature = 3
% written by Phil Tsai 11-03-99
% last revision 11-04-99
%

[num_matrices,dummy] = size(cell_array);

%generate 3-dim'l array containing the 2x2 individual matrices
% the round trip includes each element twice, EXCEPT for the first
% and last matrices, which are assumed to be the end mirrors

for j = 1 : num_matrices
    switch cell_array{j,1}
        case 'm', ABCD(:, :,j) = optic('mirror',cell_array{j,2});
        case 'p', ABCD(:, :,j) = optic('prop',cell_array{j,2});
        case 'l', ABCD(:, :,j) = optic('lens',cell_array{j,2});
        case 'i', ABCD(:, :,j) = optic('interface',cell_array{j,2},cell_array{j,3});
        case 'c', ABCD(:, :,j) = optic('cinterface',cell_array{j,2},cell_array{j,3},...
            cell_array{j,4});
    end
end

for j = num_matrices-1 : -1 : 2
    k = 2*num_matrices - j;
    switch cell_array{j,1}
        case 'm', ABCD(:, :,k) = optic('mirror',cell_array{j,2});
        case 'p', ABCD(:, :,k) = optic('prop',cell_array{j,2});
        case 'l', ABCD(:, :,k) = optic('lens',cell_array{j,2});
        case 'i', ABCD(:, :,k) = optic('interface',cell_array{j,2},cell_array{j,3});
        case 'c', ABCD(:, :,k) = optic('cinterface',cell_array{j,2},cell_array{j,3},...
            cell_array{j,4});
    end
end

% multiply the matrices together to form the ABCD matrix
% for a round trip in the cavity, starting at the output coupler

ROUNDTRIP=eye(2);

for j = 1 : 2*num_matrices - 2;

```

9/6/00 5:21 PM

roundtripabcd.m

Page 2/2

```
ROUNDTRIP=ROUNDTRIP*ABCD(:,i);  
end;
```

```
rt = ROUNDTRIP;  
if nargout ==2 , indiv = ABCD; end
```

```

function M=optic(type,param1,param2,param3)
% OPTIC returns the ABCD ray matrix for an optical element
% OPTIC displays instructions for the function
% OPTIC(lens,focal_length) returns matrix for a thin lens of focal length = focal_length
% OPTIC(mirror,radius) returns matrix for a mirror of radius of curvature = radius
%     (positive radius = center of curvature on incident side)
% OPTIC(go,length) returns matrix for propagation of distance = length
%     OPTIC(prop,length) returns matrix for propagation of distance = length
%     OPTIC(travel,length) returns matrix for propagation of distance = length
% OPTIC(interface,incident_index,transmitted_index) returns matrix for planar interface
% OPTIC(cinterface,incident_index,transmitted_index,radius) returns matrix for a spherical interface
% OPTIC(sinterface,incident_index,transmitted_index,radius) returns matrix for a spherical interface
%     (positive radius = center of curvature on incident side)
%
%See Also : GO, PROP, TRAVEL, MIRROR, LENS, INTERFACE
%
% written by Phil Tsai 11-02-99
% last revision 11-04-99
%

if nargin == 0,
    blank(2);
    disp('OPTIC function returns the ABCD ray matrix for an optical element')
    disp('OPTIC displays instructions for the function')
    disp('OPTIC("lens",focal_length) returns matrix for a lens of stated focal length')
    disp('OPTIC("mirror",radius) returns matrix for a mirror of stated radius of curvature')
    disp('     (positive radius = center of curvature on incident side)')
    disp('OPTIC("go",distance) returns matrix for propagation of stated distance')
    disp(' OPTIC("prop",distance) returns matrix for propagation stated distance')
    disp(' OPTIC("travel",distance) returns matrix for propagation stated distance')
    disp('OPTIC("interface",n1,n2) returns matrix for planar interface indices n1 to n2')
    disp('OPTIC("cinterface",incident_index,transmitted_index,radius) returns matrix for a spherical
interface')
    disp('     OPTIC("sinterface",incident_index,transmitted_index,radius) returns matrix for a
spherical interface')
    disp('     (positive radius = center of curvature on incident side)')
    blank(2);
else
    switch type
        case {'lens','len','l'}
            M=[1 , 0 ; -1/param1 , 1];
        case {'mirror','miror','m'}
            M=[1 , 0 ; -2/param1 , 1];
        case {'go','prop','travel','p','g','t'}
            M=[1 , param1 ; 0 , 1];
        case {'interface','inter','i'}
            M=[1 , 0 ; 0 , param1/param2];
        case {'cinterface','sinterface','cinter','sinter'}
            M=[1 , 0 ; (param2-param1)/(param2*param3) , param1/param2];
        otherwise
            blank(2)
            disp('Error in input: type "optic" for instruction');
    end
end

```

9/6/00 5:22 PM

optic.m

Page 2/2

```
blank(2)  
M=zeros(2);
```

```
end
```

```
end
```

```
% gaussianbeam.m plots the gaussian spotsize as a function of distance for a given optical train
%
% written by Phil Tsai 11-10-99
% last revision 11-10-99
%
```

```
clear element distance
```

```
num_optics = request('Enter number of optical elements',1);
index = request('Enter index of propagation medium',1);
lamda = request('Enter vacuum wavelength (IN NANOMETERS) of light',800);
lamda = lamda/1e6;
start_spot = request('Enter starting spot size (half-diameter) in mm',.35);
start_radius = request('Enter starting radius of curvature in mm',415);
num_steps = request('Enter num steps between elements',100);
```

```
for j = 1:num_optics
    promptstring = ['Enter distance (in mm) to element #',int2str(j)];
    distance(j) = request(promptstring,1400);
    promptstring = ['Enter focal length (in mm) for element #',int2str(j)];
    element(j) = request(promptstring,1000);
end;
promptstring = ['Enter distance to final point',int2str(j)];
distance(num_optics+1) = request(promptstring,1000);
valuearray(1,2*num_optics+2:2*num_optics+4) = 0;
```

```
while 1 == 1;
```

```
clear position spot radius q ABCD
q(1) = qparam(start_spot,start_radius,lamda,index);
spot(1) = start_spot;
radius(1) = start_radius;
position(1) = 0;
```

```
for j = 1:num_optics
    ABCD(:,j) = lens(element(j));
end
```

```
count = 1;
```

```
for j = 1:num_optics
    stepsize=distance(j)/num_steps;
    for k = 0 : stepsize : distance(j)
        count=count+1;
        position(count)= position(count-1)+stepsize;
        q(count)=q(count-1)+stepsize;
        [spot(count),radius(count)] = spotsize(q(count),lamda,index);
    end
```

```
    q(count)=(ABCD(1,1,j)*q(count)+ABCD(1,2,j))/(ABCD(2,1,j)*q(count)+ABCD(2,2,j));
end
```

```
stepsize=distance(num_optics+1)/num_steps;
for k = 0 : stepsize : distance(num_optics+1)
```

```
count=count+1;
position(count)= position(count-1)+stepsize;
q(count)=q(count-1)+stepsize;
[spot(count),radius(count)] = spotsize(q(count),lamda,index);
end

plot(position(:),spot(:));
xlabel('position in mm');
ylabel('spotsize (half-diameter)');
grid;

for j = 1:num_optics
    valuearray(1,2*j-1) = distance(j);
    valuearray(1,2*j) = element(j);
    namearray{1,2*j-1} = ['Distance to Mirror #',int2str(j)];
    namearray{1,2*j} = ['Focal Length of Mirror #',int2str(j)];
end;
valuearray(1,2*num_optics+1) = distance(num_optics+1);
namearray{1,2*num_optics+1} = 'Distance to end';

valuearray = changemenu(namearray,valuearray);
if valuearray(1,2*num_optics+2) == 999; break; end;

for j = 1:num_optics
    distance(j) = valuearray(1,2*j-1);
    element(j) = valuearray(1,2*j);
end;
distance(num_optics+1) = valuearray(1,2*num_optics+1);

disp(valuearray(1,1:2*num_optics+1));
end;
```



```

% emptycavity.m program calculates the spot size in an empty cavity with n mirrors
% this program assumes no laser rod material
% written by Phil Tsai 11-02-99
% last revision 11-05-99

blank(3);
disp('M1 = output coupler; other mirrors sequentially follow beam path from M1')
blank
disp('Enter 0 for Plane Mirrors')
blank(2)

% first ask user out how many mirrors are in the cavity
num_mirrors_default_value = 5;
num_mirrors = request('How many mirrors in the cavity',num_mirrors_default_value);

%assign default values to mirror curvatures and spacings
switch num_mirrors
    case 2
        radius_default_value = [ 150 150 0 0 0 0 0 0 0 0 0 0 0 0 0 ];
        distance_default_value = [ 166 0 0 0 0 0 0 0 0 0 0 0 0 0 ];
    case 3
        radius_default_value = [ 0 150 150 0 0 0 0 0 0 0 0 0 0 0 0 ];
        distance_default_value = [ 750 166 0 0 0 0 0 0 0 0 0 0 0 0 ];
    case 4
        radius_default_value = [ 0 150 150 0 0 0 0 0 0 0 0 0 0 0 0 ];
        distance_default_value = [ 750 166 750 0 0 0 0 0 0 0 0 0 0 0 ];
    case 5
        radius_default_value = [ 0 100 100 100 0 0 0 0 0 0 0 0 0 0 0 ];
        distance_default_value = [ 750 106 700 50 0 0 0 0 0 0 0 0 0 0 ];
    case 6
        radius_default_value = [ 0 0 150 150 0 0 0 0 0 0 0 0 0 0 0 ];
        distance_default_value = [ 50 750 166 750 50 0 0 0 0 0 0 0 0 0 ];
    case 7
        radius_default_value = [ 0 0 150 150 0 0 0 0 0 0 0 0 0 0 0 ];
        distance_default_value = [ 50 750 166 750 50 50 0 0 0 0 0 0 0 0 ];
end
wavelength_default_value = 1050;    %this value is in nanometers
step_resolution = 10;
ambient_index = 1;
clear q spotsize R position choicelist distance radius_curv cell_array;

% ask user for custom mirror curvatures and spacings
% and defaults to the defaults values if no input is given
for j = 1:num_mirrors
    prompt_str = ['Enter radius of curvature of Mirror ',int2str(j),' in mm'];
    radius_curv(j) = request(prompt_str,radius_default_value(j));
end

for j = 1:num_mirrors-1
    prompt_str = ['Enter distance between Mirrors ',int2str(j),' & ',int2str(j+1),' in mm'];

```

```

    distance(j) = request(prompt_str,distance_default_value(j));
end

wavelength = request('Enter wavelength IN NANOMETERS',wavelength_default_value);
wavelength = wavelength/1e6;          %convert nanometers to mm
blank
disp('Please wait...');
blank

%here is the part of the program that loops upon changes to parameters
while 0==0;

% of course, planar mirrors have a radius of curvature of infinity, not zero
% so, we fix that here
for j = 1:num_mirrors
    if radius_curv(j) == 0 ; radius_curv(j) = inf ; end;
end

% create a cell array that contains the sequence of mirrors and propagations
% through the laser cavity; use this cell_array as input for the
% home-made function ROUNDTRIPABCD to generate ABCD matrices and
% total matrix for a round trip through the cavity, starting at M1
or j = 1 : 2 : 2*(num_mirrors - 1)
    cell_array(j,:) = {'m',radius_curv((j+1)/2),0,0};
    cell_array(j+1,:) = {'p',distance((j+1)/2),0,0};
end
cell_array(2*num_mirrors-1,:) = {'m',radius_curv(num_mirrors),0,0};

% call home-made functions ROUNDTRIPABCD and STABLECAVITY
% STABLECAVITY gives the standing wave q-parameter at mirror1
[ROUNDTRIP,ABCD] = roundtripabcd(cell_array);
q(1) = stablecavity(ROUNDTRIP);
position(1) = 0;

% Now, step through the cavity and record the q-parameter as a function
% of position, by dividing each segment into discrete steps
counter=1;
for j=1:num_mirrors-1
    stepsize=distance(j)/step_resolution;

    for k=stepsize:stepsize:distance(j)
        counter=counter+1;
        position(counter)=position(counter-1)+stepsize;
        q(counter)=q(counter-1)+stepsize;
    end

    MA=ABCD(1,1,2*j+1);
    MB=ABCD(1,2,2*j+1);
    MC=ABCD(2,1,2*j+1);

```

```

    MD=ABCD(2,2,2*j+1);
    q(counter)=(MA*q(counter)+MB)/(MC*q(counter)+MD);
end

for j=num_mirrors-1:-1:1
    stepsize=distance(j)/step_resolution;

    for k=stepsize:stepsize:distance(j)
        counter=counter+1;
        position(counter)=position(counter-1)+stepsize;
        q(counter)=q(counter-1)+stepsize;
    end

    MA=ABCD(1,1,2*j-1);
    MB=ABCD(1,2,2*j-1);
    MC=ABCD(2,1,2*j-1);
    MD=ABCD(2,2,2*j-1);
    q(counter)=(MA*q(counter)+MB)/(MC*q(counter)+MD);
end
num_steps = counter;

% Calculate the spotsize and radius of curvature of the beam as a function
% of position, from the q-parameter using homemade function spotsize
[spotsize,R] = spotsize(q,wavelength,ambient_index);

figure(1);
plot(position,spotsize,'b-');
xlabel('Position in Cavity (in mm)');
ylabel('Spotsize in mm');

% optional figure 2 plots radius of curvature vs position
%figure(2);
%plot(position,R,'r-');
%xlabel('Position in Cavity (in mm)');
%ylabel('Radius of curvature in mm');

% This section allows the user to adjust the parameters and replot
% the spotsize with the new parameters

choicelist{1} = 'Pick a parameter to change';
choicelist{2} = 'Repeat Last';

for j=1:num_mirrors-1
    choicelist{j+2} = strcat('Radius of M',int2str(j));
    choicelist{j+num_mirrors+2} = strcat('Distance M',int2str(j),...
        ' -> M',int2str(j+1));
end

choicelist{num_mirrors+2} = strcat('Radius of M',int2str(num_mirrors));
choicelist{2*num_mirrors+2} = 'Exit Program';
choice1 = strmenu(choicelist);

```

```

% if choice1 = 1 then repeat
% if choice1 = (2 : num_mirrors+1) then change radius_curv(choice1-1)
% if choice1 = (num_mirrors+2 : 2*num_mirrors) then change
%     distance(choice1-num_mirrors-1)
% if choice1 = 2*num_mirrors+1 then exit

if choice1==2*num_mirrors+1, break; end;
if choice1~=1
    choice2 = menu('Pick one:', 'Change to..', 'Increase by..', 'Decrease by..');
end

if choice2 ==1
    newvalue = input('Change to : ');
else
    if choice1~=1 & choice2==2, increaseby = input('Increase by: '); end;
    if choice1~=1 & choice2==3, decreaseby = input('Decrease by: '); end;
end

if choice1==1, choice1 = lastchoice1; end;

if choice2==1
    if choice1 > 1 & choice1 <= num_mirrors+1;
        if newvalue == 0, newvalue = inf; end;
        radius_curv(choice1-1) = newvalue
    end
    if choice1 >= num_mirrors+2 & choice1 <= 2*num_mirrors
        distance(choice1-num_mirrors-1) = newvalue
    end
end

if choice2==2
    if choice1 > 1 & choice1 <= num_mirrors+1;
        radius_curv(choice1-1)=radius_curv(choice1-1)+increaseby
        if radius_curv(choice1-1)==0, radius_curv(choice1-1)=inf, end;
    end
    if choice1 >= num_mirrors+2 & choice1 <= 2*num_mirrors
        distance(choice1-num_mirrors-1)=distance(choice1-num_mirrors-1)+increaseby
    end
end

if choice2==3
    if choice1 > 1 & choice1 <= num_mirrors+1;
        radius_curv(choice1-1)=radius_curv(choice1-1)-decreaseby
    end
    if choice1 >= num_mirrors+2 & choice1 <= 2*num_mirrors
        distance(choice1-num_mirrors-1)=distance(choice1-num_mirrors-1)-decreaseby
        if distance(choice1-num_mirrors-1)<1, distance(choice1-num_mirrors-1)=1, end;
    end
end

lastchoice1 = choice1;

end

```

```

function n=opindex(material_name,wavelength)
% OPINDEX gives the index of refraction at a particular wavelength for a variety of materials
%   n=OPINDEX(material_name,wavelength) inputs the material_name as a string
%       and the desired wavelength (in NANOMETERS), and outputs the index of refraction
%   OPINDEX with no parameters shows a list of available material names
%
%   written by Phil Tsai on 11-05-99
%       using dispersion data provided by David Fittinghoff
%   last updated: 11-05-99
%

%data is stored in a cell array of the following form
% data(n) = {'name', disperion_form, parameter1, parameter2, .etc...}

data(1,:) = {'Silica', 13, 0.696116, 0.00467914826, 0.4079426, 0.0135120631,
0.8974794,97.340025,0,0,0,0};
data(2,:) = {'SF10',3,2.8784725,-0.010565453,0.03327942,0.0020551378,-
0.00011396226,0.000016340021,0,0,0,0};
data(3,:) = {'LaK31',3,2.828385,-0.014963716,0.0182108258,0.00036816151,-
0.0000086283328,0.00000075649349,0,0,0,0};
data(4,:) = {'e-Sapphire',4,1.5039759,7.40298e-
2,0.55069141,0.1216529,6.5927379,20.072248,0,0,0,0};

[num_materials,dummy] = size(data);
material_choice = 0;

if nargin == 0
    blank(2);
    disp('The currently available materials are:');
    blank;
    for j=1:num_materials;
        disp(data{j,1});
    end;
    blank(2);
    return
end

for j = 1:num_materials;
    if strcmp(upper(data{j,1}),upper(material_name)), material_choice = j; end;
end
if material_choice == 0;
    blank
    disp('No match to material name. Type "opindex" for list of materials');
    blank
    return;
end

for j = 3:12
    p(j-2) = data{material_choice,j};           % p & x are just redefinitions to make
end                                           % the formulas easier to type out
x=wavelength/1000;                          % (and converts wavelength to microns)

switch data{material_choice,2}

```

case 3

n=sqrt(p(1)+p(2)*x^2+p(3)/x^2+p(4)/x^4+p(5)/x^6+p(6)/x^8);

case 4

n=sqrt(1+p(1)*x^2/(x^2-p(2)^2)+p(3)*x^2/(x^2-p(4)^2)+p(5)*x^2/(x^2-p(6)^2));

case 13

n=sqrt(1 + (x^2) * (p(1) / (x^2-p(2)) + p(3)/(x^2-p(4)) + p(5) / (x-p(6))));

end