MATLAB®

The Language of Technical Computing

Computation

Visualization

Programming

The MATH WORKS Inc.

C Math Library Reference

Version 1.2

How to Contact The MathWorks:

508-647-7000 Phone

508-647-7001 Fax

The MathWorks, Inc.
24 Prime Park Way
Natick, MA 01760-1500 Mail

`http://www.mathworks.com` Web
`ftp.mathworks.com` Anonymous FTP server
`comp.soft-sys.matlab` Newsgroup

`support@mathworks.com` Technical support
`suggest@mathworks.com` Product enhancement suggestions
`bugs@mathworks.com` Bug reports
`doc@mathworks.com` Documentation error reports
`subscribe@mathworks.com` Subscribing user registration
`service@mathworks.com` Order status, license renewals, passcodes
`info@mathworks.com` Sales, pricing, and general information

MATLAB C Math Library 1.2 Reference

Printing History: January 1998 New for MATLAB 5.2 (online version in HTML form)
January 1999 Online version in PDF form

Introduction

This reference gives you quick access to the prototypes and call syntax for the MATLAB C Math Library functions. The functions fall into two groups: the mathematical functions and the utility functions. This section discusses the organization of the reference pages.

Refer to the online *Application Program Interface Reference* for documentation of the `mx` routines that let you create, access, and delete arrays.

Reference Pages

Use the reference pages to look up the prototype and syntax for a MATLAB C Math Library function. At the bottom of each page, you'll find a link to the documentation for the MATLAB version of the function. Use that page to look up the description of the arguments and the behavior of the function.

Structure

A reference page for a MATLAB C Math Library function includes these sections:

- Purpose
- C Prototype
- C Syntax
- MATLAB Syntax
- See Also links to the MATLAB version of the function and to the calling conventions

One C prototype represents the MATLAB syntax.

To make the reference pages easier to read:

- The variable names that appear in the "MATLAB Syntax" section are used as parameter names in the prototype for a function.
- The first call to a function listed under "C Syntax" corresponds to the first call listed under "MATLAB Syntax." The second C call corresponds to the second MATLAB call, and so forth.

The “C Syntax” section shows only the calls supported by the library. When you link to the MATLAB version of the function, you may notice MATLAB syntax that supports multidimensional arrays, cell arrays, structures, and objects. Because this version of the MATLAB C Math Library does not support those data types, that documentation does not apply to the C version of the function.

Typographic Conventions

- String arrays, including those that represent a function name, are italicized to indicate that you must assign a value to the variable.
- In general, a lowercase variable name/argument indicates a vector.
- In general, an uppercase variable name/argument indicates a matrix.

What Isn’t Presented in the C Syntax Section

- Assignments to input arguments, including assignments to string arrays
- Deletion of allocated arrays (Calls to `mxDestroyArray()` are not shown.)

Exceptions

- Occasionally, you'll find a C prototype where the parameter names do not match those used in the “MATLAB Syntax” section. The correspondence between the arguments used to call the function and the C prototype is too varied to represent in one prototype. `O1`, `O2`, etc. and `I1`, `I2`, etc. substitute for the output argument and input argument names in the prototype.
- Occasionally, a call to `mxCreateString()` initializes a string array; a call to `mlfScalar()` initializes an array that stores an integer; a call to `mlfHorzcat()` initializes a vector.

Introduction

This section demonstrates the calling conventions that apply to the MATLAB C Math Library functions, including what data type to use for C input and output arguments, how to handle optional arguments, and how to handle MATLAB's multiple output values in C.

Refer to the "Calling Conventions" section of Chapter 3 in the *MATLAB C Math Library User's Guide* for further discussion of the calling conventions and for a list of exceptions to the calling conventions.

How the C Prototype Is Constructed

One C prototype supports all the possible ways to call a particular MATLAB C Math Library function. You can reconstruct the C prototype by examining the MATLAB syntax for a function.

In the following procedure, the MATLAB function `svd()` and the corresponding library function `mlfSvd()` are used to illustrate the process.

MATLAB Syntax

```
s = svd(X)
[U,S,V] = svd(X)
[U,S,V] = svd(X,0)
```

The C prototype for `mlfSvd()` is constructed step-by-step. Until the last step, the prototype is incomplete.

Adding the Output Arguments

1. Find the statement that includes the largest number of output arguments.

   Choose:

   ```
   [U,S,V] = svd(X,0)
   ```

2. Subtract out the first output argument, `U`, to be the return value from the function. The data type for the return value is `mxArray *`.

   ```
   mxArray *mlfSvd(
   ```

3. Add the remaining number of MATLAB output arguments, `S` and `V`, as the first, second, etc., arguments to the C function. The data type for a C output argument is `mxArray **`.

   ```
   mxArray *mlfSvd(mxArray **S, mxArray **V,
   ```

Adding the Input Arguments

1. Find the syntax that includes the largest number of input arguments.

   Choose:

   ```
   [U,S,V] = svd(X,0)
   ```

2. Add that number of input arguments, `X` and `Zero`, to the prototype, one after another following the output arguments. The data type for an input argument is `mxArray *`.

   ```
   mxArray *mlfSvd(mxArray **S, mxArray **V, mxArray *X,
                   mxArray *Zero);
   ```

The prototype is complete.

NOTE: Contrast the data type for an output argument with the data type for an input argument. The type for an output argument is the address of a pointer to an `mxArray`. The type for an input argument is a pointer to an `mxArray`.

How to Translate a MATLAB Call into a C Call

This procedure demonstrates how to translate the MATLAB `svd()` calls into MATLAB C Math Library calls to `mlfSvd()`. The procedure applies to library functions in general.

Note that within a call to a MATLAB C Math Library function, an output argument is preceded by `&`; an input argument is not.

MATLAB Syntax

```
s = svd(X)
[U,S,V] = svd(X)
[U,S,V] = svd(X,0)
```

The MATLAB arguments to `svd()` fall into these categories:

`U` (or `s`) is a required output argument.

`S` and `V` are optional output arguments.

`X` is a required input argument.

`Zero` is an optional input argument.

1. Declare input, output, and return variables as `mxArray *` variables, and assign values to the input variables.
2. Make the first output argument the return value from the function.

   ```
   s =
   U =
   U =
   ```

3. Pass any additional required or optional output arguments as the first arguments to the function. Pass a `NULL` argument wherever an optional output argument does not apply to the particular call.

   ```
   s = mlfSvd(NULL,NULL,
   U = mlfSvd(&S,&V,
   U = mlfSvd(&S,&V,
   ```

4 Pass any required or optional input arguments that apply to the C function, following the output arguments. Pass a NULL argument wherever an optional input argument does not apply to the particular call.

```
s = mlfSvd(NULL,NULL,X,NULL);
U = mlfSvd(&S,&V,X,NULL);
U = mlfSvd(&S,&V,X,Zero);
```

NOTE: NULL arguments always follow significant arguments; a NULL argument cannot appear between two output arguments or between two input arguments. Move the significant output or input argument before the NULL argument.

Purpose

Matrix and array arithmetic

C Prototype

```
/* Matrix Arithmetic */
mxArray *mlfPlus(mxArray *A, mxArray *B);
mxArray *mlfMinus(mxArray *A, mxArray *B);
mxArray *mlfUnaryminus(mxArray *A);
mxArray *mlfUminus(mxArray *A);
mxArray *mlfMtimes(mxArray *A, mxArray *B);
mxArray *mlfMrdivide(mxArray *A, mxArray *B);
mxArray *mlfMldivide(mxArray *A, mxArray *B);
mxArray *mlfMpower(mxArray *A, mxArray *B);
mxArray *mlfCtranspose(mxArray *A);

/* Array Arithmetic */
mxArray *mlfTimes(mxArray *A, mxArray *B);
mxArray *mlfRdivide(mxArray *A, mxArray *B);
mxArray *mlfMldivide(mxArray *A, mxArray *B);
mxArray *mlfMpower(mxArray *A, mxArray *B);
mxArray *mlfTranspose(mxArray *A);
```

Arithmetic Operators

C Syntax

```
#include "matlab.h"

mxArray *A, *B;             /* Input arguments */
mxArray *C;                 /* Return value */

/* Matrix Arithmetic */
C = mlfPlus(A,B);
C = mlfMinus(A,B);
C = mlfUnaryminus(A);
C = mlfUminus(A);
C = mlfMtimes(A,B);
C = mlfMrdivide(A,B);
C = mlfMldivide(A,B);
C = mlfMpower(A,B);
C = mlfCtranspose(A);

/* Array Arithmetic */
C = mlfTimes(A,B);
C = mlfRdivide(A,B);
C = mlfMldivide(A,B);
C = mlfMpower(A,B);
C = mlfTranspose(A);
```

MATLAB Syntax

```
A+B
A-B
A*B     A.*B
A/B     A./B
A\B     A.\B
A^B     A.^B
A'      A.'
```

See Also

MATLAB Arithmetic Operators Calling Conventions

Purpose

Relational operations

C Prototype

```
mxArray *mlfLt(mxArray *A, mxArray *B);
mxArray *mlfGt(mxArray *A, mxArray *B);
mxArray *mlfLe(mxArray *A, mxArray *B);
mxArray *mlfGe(mxArray *A, mxArray *B);
mxArray *mlfEq(mxArray *A, mxArray *B);
mxArray *mlfNe(mxArray *A, mxArray *B);
mxArray *mlfNeq(mxArray *A, mxArray *B);
```

C Syntax

```
#include "matlab.h"

mxArray *A, *B;             /* Input arguments */
mxArray *C;                 /* Return value */

C = mlfLt(A,B);
C = mlfGt(A,B);
C = mlfLe(A,B);
C = mlfGe(A,B);
C = mlfEq(A,B);
C = mlfNe(A,B);
C = mlfNeq(A,B);
```

MATLAB Syntax

```
A < B
A > B
A <= B
A >= B
A == B
A ~= B
```

See Also

MATLAB Relational Operators Calling Conventions

Logical Operators

Purpose

Logical operations

C Prototype

```
mxArray *mlfAnd(mxArray *A, mxArray *B);
mxArray *mlfOr(mxArray *A, mxArray *B);
mxArray *mlfNot(mxArray *A);
```

C Syntax

```
#include "matlab.h"

mxArray *A, *B;             /* Input arguments */
mxArray *C;                 /* Return value */

C = mlfAnd(A,B);
C = mlfOr(A,B);
C = mlfNot(A);
```

MATLAB Syntax

```
A & B
A | B
~A
```

See Also

MATLAB Logical Operators, Calling Conventions

Purpose

Absolute value and complex magnitude

C Prototype

```
mxArray *mlfAbs(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfAbs(X);
```

MATLAB Syntax

```
Y = abs(X)
```

See Also

MATLAB `abs` Calling Conventions

mlfAcos, mlfAcosh

Purpose

Inverse cosine and inverse hyperbolic cosine

C Prototype

```
mxArray *mlfAcos(mxArray *X);
mxArray *mlfAcosh(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfAcos(X);
Y = mlfAcosh(X);
```

MATLAB Syntax

```
Y = acos(X)
Y = acosh(X)
```

See Also

MATLAB acos, acosh Calling Conventions

Purpose

Inverse cotangent and inverse hyperbolic cotangent

C Prototype

```
mxArray *mlfAcot(mxArray *X);
mxArray *mlfAcoth(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfAcot(X);
Y = mlfAcoth(X);
```

MATLAB Syntax

```
Y = acot(X)
Y = acoth(X)
```

See Also

MATLAB `acot`, `acoth` Calling Conventions

mlfAcsc, mlfAcsch

Purpose

Inverse cosecant and inverse hyperbolic cosecant

C Prototype

```
mxArray *mlfAcsc(mxArray *X);
mxArray *mlfAcsch(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfAcsc(X);
Y = mlfAcsch(X);
```

MATLAB Syntax

```
Y = acsc(X)
Y = acsch(X)
```

See Also

MATLAB acsc, acsch Calling Conventions

Purpose

Test to determine if all elements are nonzero

C Prototype

```
mxArray *mlfAll(mxArray *A, mxArray *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *dim;           /* Optional input argument(s) */
mxArray *B;             /* Return value */

B = mlfAll(A,NULL);
B = mlfAll(A,dim);
```

MATLAB Syntax

```
B = all(A)
B = all(A,dim)
```

See Also

MATLAB `all` Calling Conventions

mlfAngle

Purpose

Phase angle

C Prototype

```
mxArray *mlfAngle(mxArray *Z);
```

C Syntax

```
#include "matlab.h"

mxArray *Z;             /* Required input argument(s) */
mxArray *P;             /* Return value */

P = mlfAngle(Z);
```

MATLAB Syntax

```
P = angle(Z)
```

See Also

MATLAB `angle` Calling Conventions

Purpose

Test for any nonzeros

C Prototype

```
mxArray *mlfAny(mxArray *A, mxArray *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *dim;           /* Optional input argument(s) */
mxArray *B;             /* Return value */

B = mlfAny(A,NULL);
B = mlfAny(A,dim);
```

MATLAB Syntax

```
B = any(A)
B = any(A,dim)
```

See Also

MATLAB any Calling Conventions

mlfAsec, mlfAsech

Purpose

Inverse secant and inverse hyperbolic secant

C Prototype

```
mxArray *mlfAsec(mxArray *X);
mxArray *mlfAsech(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfAsec(X);
Y = mlfAsech(X);
```

MATLAB Syntax

```
Y = asec(X)
Y = asech(X)
```

See Also

MATLAB asec, asech Calling Conventions

Purpose

Inverse sine and inverse hyperbolic sine

C Prototype

```
mxArray *mlfAsin(mxArray *X);
mxArray *mlfAsinh(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfAsin(X);
Y = mlfAsinh(X);
```

MATLAB Syntax

```
Y = asin(X)
Y = asinh(X)
```

See Also

MATLAB `asin`, `asinh` Calling Conventions

mlfAtan, mlfAtanh

Purpose

Inverse tangent and inverse hyperbolic tangent

C Prototype

```
mxArray *mlfAtan(mxArray *X);
mxArray *mlfAtanh(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfAtan(X);
Y = mlfAtanh(X);
```

MATLAB Syntax

```
Y = atan(X)
Y = atanh(X)
```

See Also

MATLAB `atan`, `atanh` Calling Conventions

Purpose

Four-quadrant inverse tangent

C Prototype

```
mxArray *mlfAtan2(mxArray *Y, mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *Y, *X;         /* Required input argument(s) */
mxArray *P;             /* Return value */

P = mlfAtan2(Y,X);
```

MATLAB Syntax

```
P = atan2(Y,X)
```

See Also

MATLAB `atan` Calling Conventions

mlfBalance

Purpose

Improve accuracy of computed eigenvalues

C Prototype

```
mxArray *mlfBalance(mxArray **B, mxArray *A);
```

C Syntax

```
#include "matlab.h"

mxArray *A, *B, *D;

D = mlfBalance(&B,A);
B = mlfBalance(NULL,A);
```

MATLAB Syntax

```
[D,B] = balance(A)
B = balance(A)
```

See Also

MATLAB `balance` Calling Conventions

Purpose

Base to decimal number conversion

C Prototype

```
mxArray *mlfBase2dec(mxArray *str, mxArray *base);
```

C Syntax

```
#include "matlab.h"

mxArray *str;               /* String array(s) */
mxArray *base;              /* Required input argument(s) */
mxArray *d;                 /* Return value */

d = mlfBase2dec(str,base);
```

MATLAB Syntax

```
d = base2dec('strn',base)
```

See Also

MATLAB `base2dec` Calling Conventions

mlfBeta, mlfBetainc, mlfBetaln

Purpose

Beta functions

C Prototype

```
mxArray *mlfBeta(mxArray *Z, mxArray *W);
mxArray *mlfBetainc(mxArray *X, mxArray *Z, mxArray *W);
mxArray *mlfBetaln(mxArray *Z, mxArray *W);
```

C Syntax

```
#include "matlab.h"

mxArray *Z, *W, *X;     /* Required input argument(s) */
mxArray *B, *I, *L;     /* Return value */

B = mlfBeta(Z,W);
I = mlfBetainc(X,Z,W);
L = mlfBetaln(Z,W);
```

MATLAB Syntax

```
B = beta(Z,W)
I = betainc(X,Z,W)
L = betaln(Z,W)
```

See Also

MATLAB `beta`, `betainc`, `betaln`Calling Conventions

Purpose

Binary to decimal number conversion

C Prototype

```
mxArray *mlfBin2dec(mxArray *binarystr);
```

C Syntax

```
#include "matlab.h"

mxArray *binarystr;         /* Required input argument(s) */
mxArray *decnumber;         /* Return value */

decnumber = mlfBin2dec(binarystr);
```

MATLAB Syntax

bin2dec(*binarystr*)

See Also

MATLAB `bin2dec` Calling Conventions

mlfBlanks

Purpose

A string of blanks

C Prototype

```
mxArray *mlfBlanks(mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *n;             /* Required input argument(s) */
mxArray *r;             /* Return value */

r = mlfBlanks(n);
```

MATLAB Syntax

```
blanks(n)
```

See Also

MATLAB `blanks` Calling Conventions

Purpose

Calendar

C Prototype

```
mxArray *mlfCalendar(mxArray *y, mxArray *m);
```

C Syntax

```
#include "matlab.h"

mxArray *d, *y, *m;         /* Input argument(s) */
mxArray *c;                 /* Return value */

c = mlfCalendar(NULL,NULL);
c = mlfCalendar(d,NULL);
c = mlfCalendar(y,m);
```

MATLAB Syntax

```
c = calendar
c = calendar(d)
c = calendar(y,m)
```

See Also

MATLAB `calendar` Calling Conventions

mlfCart2pol

Purpose

Transform Cartesian coordinates to polar or cylindrical

C Prototype

```
mxArray *mlfCart2pol(mxArray **RHO, mxArray **Z_out, mxArray *X,
                     mxArray *Y, mxArray *Z_in);
```

C Syntax

```
#include "matlab.h"

mxArray *X, *Y;         /* Required input argument(s) */
mxArray *Z_in;          /* Optional input argument(s) */
mxArray *RHO;           /* Required output argument(s) */
mxArray *Z_out;         /* Optional output argument(s) */
mxArray *THETA;         /* Return value */

THETA = mlfCart2pol(&RHO,&Z_out,X,Y,Z_in);
THETA = mlfCart2pol(&RHO,NULL,X,Y,NULL);
```

MATLAB Syntax

```
[THETA,RHO,Z] = cart2pol(X,Y,Z)
[THETA,RHO] = cart2pol(X,Y)
```

See Also

MATLAB `cart2pol` Calling Conventions

Purpose

Transform Cartesian coordinates to spherical

C Prototype

```
mxArray *mlfCart2sph(mxArray **PHI, mxArray **R, mxArray *X,
                     mxArray *Y, mxArray *Z);
```

C Syntax

```
#include "matlab.h"

mxArray *X, *Y, *Z;     /* Required input argument(s) */
mxArray *PHI,*R;        /* Required output argument(s) */
mxArray *THETA;         /* Return value */

THETA = mlfCart2sph(&PHI,&R,X,Y,Z);
```

MATLAB Syntax

```
[THETA,PHI,R] = cart2sph(X,Y,Z)
```

See Also

MATLAB `cart2sph` Calling Conventions

mlfCat

Purpose

Concatenate arrays. Minimum number of input arguments: three, maximum: user-defined. Terminate the list of arguments with a NULL.

C Prototype

```
mxArray *mlfCat(mxArray *dim, mxArray *A1,...);
```

C Syntax

```
#include "matlab.h"

mxArray *dim, *A1, *A2;     /* Required input argument(s) */
mxArray *A3, *A4;           /* Optional input argument(s) */
mxArray *C;                 /* Return value */

C = mlfCat(dim,A1,A2,NULL);
C = mlfCat(dim,A1,A2,A3,A4,NULL);
```

MATLAB Syntax

```
C = cat(dim,A,B)
C = cat(dim,A1,A2,A3,A4...)
```

See Also

MATLAB `cat` Calling Conventions

Purpose

Convert complex diagonal form to real block diagonal form

C Prototype

```
mxArray *mlfCdf2rdf(mxArray **D_out, mxArray *V_in, mxArray *D_in);
```

C Syntax

```
#include "matlab.h"

mxArray *V_in, *D_in;       /* Required input argument(s) */
mxArray *D_out;             /* Required output argument(s) */
mxArray *V_out;             /* Return value */

V_out = mlfCdf2rdf(&D_out,V_in,D_in);
```

MATLAB Syntax

```
[V,D] = cdf2rdf(V,D)
```

See Also

MATLAB `cdf2rdf` Calling Conventions

mlfCeil

Purpose

Round toward infinity

C Prototype

```
mxArray *mlfCeil(mxArray *A);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *B;             /* Return value */

B = mlfCeil(A);
```

MATLAB Syntax

```
B = ceil(A)
```

See Also

MATLAB `ceil` Calling Conventions

Purpose

Create character array (string). Minimum input arguments: one, maximum: user-defined. Terminate the list of arguments with a NULL.

C Prototype

```
mxArray *mlfChar(mxArray *RI1, ... );
```

C Syntax

```
#include "matlab.h"

mxArray *t1;            /* Required input argument(s) */
mxArray *t2, *t3;       /* Optional input argument(s) */
mxArray *S;             /* Return value */

S = mlfChar(t1,NULL);
S = mlfChar(t1,t2,t3,NULL);
```

MATLAB Syntax

```
S = char(X)
S = char(t1,t2,t3...)
```

See Also

MATLAB `char` Calling Conventions

mlfChol

Purpose

Cholesky factorization

C Prototype

```
mxArray *mlfChol(mxArray **p, mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *p;             /* Optional output argument(s) */
mxArray *R;             /* Return value */

R = mlfChol(NULL,X);
R = mlfChol(&p,X);
```

MATLAB Syntax

```
R = chol(X)
[R,p] = chol(X)
```

See Also

MATLAB `chol` Calling Conventions

Purpose

Rank 1 update to Cholesky factorization

C Prototype

```
mxArray *mlfCholupdate(mxArray **p, mxArray *R, mxArray *x,
                       mxArray *flag);
```

C Syntax

```
#include "matlab.h"

mxArray *R, *x;         /* Required input argument(s) */
mxArray *flag;          /* Optional string input argument */
mxArray *p;             /* Optional output argument(s) */
mxArray *R1;            /* Return value */

R1 = mlfCholupdate(NULL,R,x,NULL);
R1 = mlfCholupdate(NULL,R,x,flag);
R1 = mlfCholupdate(&p,R,x,flag);
```

MATLAB Syntax

```
R1 = cholupdate(R,x)
R1 = cholupdate(R,x,'+')
R1 = cholupdate(R,x,'-')
[R1,p] = cholupdate(R,x,'-')
```

See Also

MATLAB `cholupdate` Calling Conventions

mlfClassName

Purpose

Create object or return class of object

C Prototype

```
mxArray *mlfClassName(mxArray *obj);
```

C Syntax

```
#include "matlab.h"

mxArray *obj;               /* Required input argument(s) */
mxArray *str;               /* Return value */

str = mlfClassName(obj);
```

MATLAB Syntax

```
str = class(object)
```

See Also

MATLAB `class` Calling Conventions

Purpose

Current time as a date vector

C Prototype

```
mxArray *mlfClock();
```

C Syntax

```
#include "matlab.h"

mxArray *c;             /* Return value */

c = mlfClock();
```

MATLAB Syntax

```
c = clock
```

See Also

MATLAB `clock` Calling Conventions

mlfCompan

Purpose

Companion matrix

C Prototype

```
mxArray *mlfCompan(mxArray *u);
```

C Syntax

```
#include "matlab.h"

mxArray *u;             /* Required input argument(s) */
mxArray *A;             /* Return value */

A = mlfCompan(u);
```

MATLAB Syntax

```
A = compan(u)
```

See Also

MATLAB `compan` Calling Conventions

Purpose

Identify the computer on which MATLAB is running

C Prototype

```
mxArray *mlfComputer(mxArray **maxsize);
```

C Syntax

```
#include "matlab.h"

mxArray *maxsize;           /* Optional input argument(s) */
mxArray *str;               /* Return value */

str = mlfComputer(NULL);
str = mlfComputer(&maxsize);
```

MATLAB Syntax

```
str = computer
[str,maxsize] = computer
```

See Also

MATLAB `computer` Calling Conventions

mlfCond

Purpose

Condition number with respect to inversion

C Prototype

```
mxArray *mlfCond(mxArray *X, *p);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *p;             /* Optional input argument(s) */
mxArray *c;             /* Return value */

c = mlfCond(X,NULL);
c = mlfCond(X,p);
```

MATLAB Syntax

```
c = cond(X)
c = cond(X,p)
```

See Also

MATLAB cond Calling Conventions

Purpose

Condition number with respect to eigenvalues

C Prototype

```
mxArray *mlfCondeig(mxArray **D, mxArray **s, mxArray *A);
```

C Syntax

```
#include "matlab.h"

mxArray *A;                 /* Required input argument(s) */
mxArray *D, *s;             /* Optional output argument(s) */
mxArray *c, *V;             /* Return value */

c = mlfCondeig(NULL,NULL,A);
V = mlfCondeig(&D,&s,A);
```

MATLAB Syntax

```
c = condeig(A)
[V,D,s] = condeig(A)
```

See Also

MATLAB `condeig` Calling Conventions

mlfCondest

Purpose

1-norm matrix condition number estimate

C Prototype

```
mxArray *mlfCondest(mxArray **v, mxArray *A);
```

C Syntax

```
#include "matlab.h"

mxArray *A;                 /* Required input argument(s) */
mxArray *v;                 /* Optional output argument(s) */
mxArray *c;                 /* Return value */

c = mlfCondest(NULL,A);
c = mlfCondest(&v,A);
```

MATLAB Syntax

```
c = condest(A)
[c,v] = condest(A)
```

See Also

MATLAB `condest` Calling Conventions

Purpose

Complex conjugate

C Prototype

```
mxArray *mlfConj(mxArray *Z);
```

C Syntax

```
#include "matlab.h"

mxArray *Z;             /* Required input argument(s) */
mxArray *ZC;            /* Return value */

ZC = mlfConj(Z);
```

MATLAB Syntax

```
ZC = conj(Z)
```

See Also

MATLAB `conj` Calling Conventions

mlfConv

Purpose

Convolution and polynomial multiplication

C Prototype

```
mxArray *mlfConv(mxArray *u, mxArray *v);
```

C Syntax

```
#include "matlab.h"

mxArray *u, *v;         /* Required input argument(s) */
mxArray *w;             /* Return value */

w = mlfConv(u,v);
```

MATLAB Syntax

```
w = conv(u,v)
```

See Also

MATLAB conv Calling Conventions

Purpose

Two-dimensional convolution

C Prototype

```
mxArray *mlfConv2(mxArray *I1, mxArray *I2, mxArray *I3,
                  mxArray *I4);
```

C Syntax

```
#include "matlab.h"

mxArray *shape;                 /* String array(s) */
mxArray *A, *B, *hcol, *hrow;   /* Input argument(s) */
mxArray *C;                     /* Return value */

C = mlfConv2(A,B,NULL,NULL);
C = mlfConv2(A,B,shape,NULL);
C = mlfConv2(hcol,hrow,NULL,NULL);
C = mlfConv2(hcol,hrow,A,shape);
```

MATLAB Syntax

```
C = conv2(A,B)
C = conv2(hcol,hrow,A)
C = conv2(...,'shape')
```

See Also

MATLAB `conv2` Calling Conventions

mlfCorrcoef

Purpose

Correlation coefficients

C Prototype

```
mxArray *mlfCorrcoef(mxArray *X, mxArray *y);
```

C Syntax

```
#include "matlab.h"

mxArray *X, *x, *y;     /* Input argument(s) */
mxArray *S;             /* Return value */

S = mlfCorrcoef(X,NULL);
S = mlfCorrcoef(x,y);
```

MATLAB Syntax

```
S = corrcoef(X)
S = corrcoef(x,y)
```

See Also

MATLAB `corrcoef` Calling Conventions

Purpose

Cosine and hyperbolic cosine

C Prototype

```
mxArray *mlfCos(mxArray *X);
mxArray *mlfCosh(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfCos(X);
Y = mlfCosh(X);
```

MATLAB Syntax

```
Y = cos(X)
Y = cosh(X)
```

See Also

MATLAB cos, cosh Calling Conventions

mlfCot, mlfCoth

Purpose

Cotangent and hyperbolic cotangent

C Prototype

```
mxArray *mlfCot(mxArray *X);
mxArray *mlfCoth(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfCot(X);
Y = mlfCoth(X);
```

MATLAB Syntax

```
Y = cot(X)
Y = coth(X)
```

See Also

MATLAB `cot`, `coth` Calling Conventions

Purpose

Covariance matrix

C Prototype

```
mxArray *mlfCov(mxArray *x, mxArray *Y, mxArray *Z);
```

C Syntax

```
#include "matlab.h"

mxArray *x;             /* Required input argument(s) */
mxArray *Y, *Z;         /* Optional input argument(s) */
mxArray *C;             /* Return value */

Z = mlfScalar(0);
C = mlfCov(x,Z,NULL);
C = mlfCov(x,Y,Z);
```

MATLAB Syntax

```
C = cov(X)
C = cov(x,y)
```

See Also

MATLAB cov Calling Conventions

mlfCplxpair

Purpose

Sort complex numbers into complex conjugate pairs

C Prototype

```
mxArray *mlfCplxpair(mxArray *A, mxArray *tol, mxArray *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *A;                     /* Required input argument(s) */
mxArray *tol, *dim;             /* Optional input argument(s) */
mxArray *null_matrix, *Zero;    /* Optional input argument(s) */
mxArray *B;                     /* Return value */

B = mlfCplxpair(A,NULL,NULL);
B = mlfCplxpair(A,tol,NULL);

Zero = mlfScalar(0);
null_matrix = mlfZeros(Zero,Zero);
B = mlfCplxpair(A,null_matrix,dim);

B = mlfCplxpair(A,tol,dim);
```

MATLAB Syntax

```
B = cplxpair(A)
B = cplxpair(A,tol)
B = cplxpair(A,[],dim)
B = cplxpair(A,tol,dim)
```

See Also

MATLAB `cplxpair` Calling Conventions

Purpose

Vector cross product

C Prototype

```
mxArray *mlfCross(mxArray *U, mxArray *V, mxArray *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *U, *V;         /* Required input argument(s) */
mxArray *dim;           /* Optional input argument(s) */
mxArray *W;             /* Return value */

W = mlfCross(U,V,NULL);
W = mlfCross(U,V,dim);
```

MATLAB Syntax

```
W = cross(U,V)
W = cross(U,V,dim)
```

See Also

MATLAB cross Calling Conventions

mlfCsc, mlfCsch

Purpose

Cosecant and hyperbolic cosecant

C Prototype

```
mxArray *mlfCsc(mxArray *x);
mxArray *mlfCsch(mxArray *x);
```

C Syntax

```
#include "matlab.h"

mxArray *x;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfCsc(x);
Y = mlfCsch(x);
```

MATLAB Syntax

```
Y = csc(x)
Y = csch(x)
```

See Also

MATLAB csc, csch Calling Conventions

Purpose

Cumulative product

C Prototype

```
mxArray *mlfCumprod(mxArray *A, *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *dim;           /* Optional input argument(s) */
mxArray *B;             /* Return value */

B = mlfCumprod(A,NULL);
B = mlfCumprod(A,dim);
```

MATLAB Syntax

```
B = cumprod(A)
B = cumprod(A,dim)
```

See Also

MATLAB `cumprod` Calling Conventions

mlfCumsum

Purpose

Cumulative sum

C Prototype

```
mxArray *mlfCumsum(mxArray *A, *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *dim;           /* Optional input argument(s) */
mxArray *B;             /* Return value */

B = mlfCumsum(A,NULL);
B = mlfCumsum(A,dim);
```

MATLAB Syntax

```
B = cumsum(A)
B = cumsum(A,dim)
```

See Also

MATLAB `cumsum` Calling Conventions

Purpose

Cumulative trapezoidal numerical integration

C Prototype

```
mxArray *mlfCumtrapz(mxArray *Y, mxArray *X, mxArray *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *Y;                 /* Required input argument(s) */
mxArray *X, *dim;           /* Optional input argument(s) */
mxArray *Z;                 /* Return value */

Z = mlfCumtrapz(Y,NULL,NULL);
Z = mlfCumtrapz(X,Y,NULL);
Z = mlfCumtrapz(Y,dim,NULL);
Z = mlfCumtrapz(X,Y,dim);
```

MATLAB Syntax

```
Z = cumtrapz(Y)
Z = cumtrapz(X,Y)
Z = cumtrapz(... dim)
```

See Also

MATLAB `cumtrapz` Calling Conventions

mlfDate

Purpose

Current date string

C Prototype

```
mxArray *mlfDate();
```

C Syntax

```
#include "matlab.h"

mxArray *str;           /* Return value */

str = mlfDate();
```

MATLAB Syntax

```
str = date
```

See Also

MATLAB `date` Calling Conventions

Purpose

Serial date number

C Prototype

```
mxArray *mlfDatenum(mxArray *Y, mxArray *M, mxArray *D, mxArray *H,
                    mxArray *MI, mxArray *S);
```

C Syntax

```
#include "matlab.h"

mxArray *str;               /* String array(s) */
mxArray *Y, *M, *D;         /* Input argument(s) */
mxArray *H, *MI, *S;        /* Input argument(s) */

N = mlfDatenum(str,NULL,NULL,NULL,NULL,NULL);
N = mlfDatenum(Y,M,D,NULL,NULL,NULL);
N = mlfDatenum(Y,M,D,H,MI,S);
```

MATLAB Syntax

```
N = datenum(str)
N = datenum(Y,M,D)
N = datenum(Y,M,D,H,MI,S)
```

See Also

MATLAB `datenum` Calling Conventions

mlfDatestr

Purpose

Date string format

C Prototype

```
mxArray *mlfDatestr(mxArray *D, mxArray *dateform);
```

C Syntax

```
#include "matlab.h"

mxArray *dateform;          /* Numeric or string array */
mxArray *D;                 /* Required input argument(s) */
mxArray *str;               /* Return value */

str = mlfDatestr(D,dateform);
```

MATLAB Syntax

```
str = datestr(D,dateform)
```

See Also

MATLAB `datestr` Calling Conventions

Purpose

Date components

C Prototype

```
mxArray *mlfDatevec(mxArray **M, mxArray **D, mxArray **H,
                    mxArray **MI, mxArray **S, mxArray *A);
```

C Syntax

```
#include "matlab.h"

mxArray *A;                     /* Required input argument(s) */
mxArray *M, *D, *H, *MI, *S;    /* Optional output argument(s) */
mxArray *C, *Y;                 /* Return value */

C = mlfDatevec(NULL,NULL,NULL,NULL,NULL,A);
Y = mlfDatevec(&M,&D,&H,&MI,&S,A);
```

MATLAB Syntax

```
C = datevec(A)
[Y,M,D,H,MI,S] = datevec(A)
```

See Also

MATLAB `datevec` Calling Conventions

mlfDblquad

Purpose

Numerical double integration

C Prototype

```
mxArray *mlfDblquad(mxArray *func, mxArray *inmin, mxArray *inmax,
                    mxArray *outmin, mxArray *outmax,
                    mxArray *tol, mxArray *method);
```

C Syntax

```
#include "matlab.h"

mxArray *func;                  /* String array(s) */
mxArray *inmin, *inmax;         /* Required input argument(s) */
mxArray *outmin, *outmax;       /* Required input argument(s) */
mxArray *tol, *method;          /* Optional input argument(s) */
mxArray *result;                /* Return value */

result = mlfDblquad(func,inmin,inmax,outmin,outmax,
                    NULL,NULL);
result = mlfDblquad(func,inmin,inmax,outmin,outmax,
                    tol,NULL);
result = mlfDblquad(func,inmin,inmax,outmin,outmax,
                    tol,method);
```

MATLAB Syntax

```
result = dblquad('fun',inmin,inmax,outmin,outmax)
result = dblquad('fun',inmin,inmax,outmin,outmax,tol)
result = dblquad('fun',inmin,inmax,outmin,outmax,tol,method)
```

See Also

MATLAB `dblquad` Calling Conventions

Purpose

Strip trailing blanks from the end of a string

C Prototype

```
mxArray *mlfDeblank(mxArray *str_in);
```

C Syntax

```
#include "matlab.h"

mxArray *str_in;        /* String array(s) */
mxArray *str;           /* Return value */

str = mlfDeblank(str_in);
```

MATLAB Syntax

```
str = deblank(str)
```

See Also

MATLAB `deblank` Calling Conventions

mlfDec2base

Purpose

Decimal number to base conversion

C Prototype

```
mxArray *mlfDec2base(mxArray *d, mxArray *base, mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *d, *base;          /* Required input argument(s) */
mxArray *n;                 /* Optional input argument(s) */
mxArray *str;               /* Return value */

str = mlfDec2base(d,base,NULL);
str = mlfDec2base(d,base,n);
```

MATLAB Syntax

```
str = dec2base(d,base)
str = dec2base(d,base,n)
```

See Also

MATLAB `dec2base` Calling Conventions

Purpose

Decimal to binary number conversion

C Prototype

```
mxArray *mlfDec2bin(mxArray *d, mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *d;                 /* Required input argument(s) */
mxArray *n;                 /* Optional input argument(s) */
mxArray *str;               /* Return value */

str = mlfDec2bin(d,NULL);
str = mlfDec2bin(d,n);
```

MATLAB Syntax

```
str = dec2bin(d)
str = dec2bin(d,n)
```

See Also

MATLAB `dec2bin` Calling Conventions

mlfDec2hex

Purpose

Decimal to hexadecimal number conversion

C Prototype

```
mxArray *mlfDec2hex(mxArray *d, mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *d;             /* Required input argument(s) */
mxArray *n;             /* Optional input argument(s) */
mxArray *str;           /* Return value */

str = mlfDec2hex(d,NULL);
str = mlfDec2hex(d,n);
```

MATLAB Syntax

```
str = dec2hex(d)
str = dec2hex(d,n)
```

See Also

MATLAB `dec2hex` Calling Conventions

Purpose

Deconvolution and polynomial division

C Prototype

```
mxArray *mlfDeconv(mxArray **r, mxArray *v, mxArray *u);
```

C Syntax

```
#include "matlab.h"

mxArray *v, *u;         /* Required input argument(s) */
mxArray *r;             /* Required output argument(s) */
mxArray *q;             /* Return value */

q = mlfDeconv(&r,v,u);
```

MATLAB Syntax

```
[q,r] = deconv(v,u)
```

See Also

MATLAB deconv Calling Conventions

mlfDel2

Purpose

Discrete Laplacian

C Prototype

```
mxArray *mlfDel2(mxArray *U, mxArray *hx, mxArray *hy);
```

C Syntax

```
#include "matlab.h"

mxArray *U;             /* Required input argument(s) */
mxArray *hx, *hy;       /* Optional input argument(s) */
mxArray *L;             /* Return value */

L = mlfDel2(U,NULL,NULL);
L = mlfDel2(U,hx,NULL);
L = mlfDel2(U,hx,hy);
```

MATLAB Syntax

```
L = del2(U)
L = del2(U,h)
L = del2(U,hx,hy)
```

See Also

MATLAB `del2` Calling Conventions

Purpose

Matrix determinant

C Prototype

```
mxArray *mlfDet(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *d;             /* Return value */

d = mlfDet(X);
```

MATLAB Syntax

```
d = det(X)
```

See Also

MATLAB `det` Calling Conventions

mlfDiag

Purpose

Diagonal matrices and diagonals of a matrix. The variable v can be a vector or a matrix.

C Prototype

```
mxArray *mlfDiag(mxArray *v, mxArray *k);
```

C Syntax

```
#include "matlab.h"

mxArray *v, *k, *X;

X = mlfDiag(v,k);
X = mlfDiag(v,NULL);
v = mlfDiag(X,k);
v = mlfDiag(X,NULL);
```

MATLAB Syntax

```
X = diag(v,k)
X = diag(v)
v = diag(X,k)
v = diag(X)
```

See Also

MATLAB `diag` Calling Conventions

Purpose

Differences and approximate derivatives

C Prototype

```
mxArray *mlfDiff(mxArray *X, mxArray *n, mxArray *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *n, *dim;       /* Optional input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfDiff(X,NULL,NULL);
Y = mlfDiff(X,n,NULL);
Y = mlfDiff(X,n,dim);
```

MATLAB Syntax

```
Y = diff(X)
Y = diff(X,n)
Y = diff(X,n,dim)
```

See Also

MATLAB `diff` Calling Conventions

mlfDisp

Purpose

Display text or array

C Prototype

```
void mlfDisp(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */

mlfDisp(X);
```

MATLAB Syntax

```
disp(X)
```

See Also

MATLAB `disp` Calling Conventions

Purpose

Convert to double precision

C Prototype

```
mxArray *mlfDouble(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;                 /* Required input argument(s) */
mxArray *R;                 /* Return value */

R = mlfDouble(X);
```

MATLAB Syntax

```
double(X)
```

See Also

MATLAB `double` Calling Conventions

mlfEig

Purpose

Eigenvalues and eigenvectors

C Prototype

```
mxArray *mlfEig(mxArray **D, mxArray *A, mxArray *B);
```

C Syntax

```
#include "matlab.h"

mxArray *nobalance_str;     /* String array(s) */
mxArray *A;                 /* Required input argument(s) */
mxArray *B;                 /* Optional input argument(s) */
mxArray *D;                 /* Optional output argument(s) */
mxArray *d, V;              /* Return value */

d = mlfEig(NULL,A,NULL);
V = mlfEig(&D,A,NULL);

nobalance_str = mxCreateString("no_balance");
V = mlfEig(&D,A,nobalance_str);

d = mlfEig(NULL,A,B);
V = mlfEig(&D,A,B);
```

MATLAB Syntax

```
d = eig(A)
[V,D] = eig(A)
[V,D] = eig(A,'nobalance')
d = eig(A,B)
[V,D] = eig(A,B)
```

See Also

MATLAB `eig` Calling Conventions

Purpose

Jacobi elliptic functions

C Prototype

```
mxArray *mlfEllipj(mxArray **CN, mxArray **DN, mxArray *U,
                   mxArray *M, mxArray *tol);
```

C Syntax

```
#include "matlab.h"

mxArray *U, *M;         /* Required input argument(s) */
mxArray *tol;           /* Optional input argument(s) */
mxArray *CN, *DN;       /* Required output argument(s) */
mxArray *SN;            /* Return value */

SN = mlfEllipj(&CN,&DN,U,M,NULL);
SN = mlfEllipj(&CN,&DN,U,M,tol);
```

MATLAB Syntax

```
[SN,CN,DN]  =  ellipj(U,M)
[SN,CN,DN]  =  ellipj(U,M,tol)
```

See Also

MATLAB `ellipj` Calling Conventions

mlfEllipke

Purpose

Complete elliptic integrals of the first and second kind

C Prototype

```
mxArray *mlfEllipke(mxArray **E, mxArray *M, mxArray *tol);
```

C Syntax

```
#include "matlab.h"

mxArray *M;             /* Required input argument(s) */
mxArray *tol;           /* Optional input argument(s) */
mxArray *E;             /* Optional output argument(s) */
mxArray *K;             /* Return value */

K = mlfEllipke(NULL,M,NULL);
K = mlfEllipke(&E,M,NULL);
K = mlfEllipke(&E,M,tol);
```

MATLAB Syntax

```
K = ellipke(M)
[K,E] = ellipke(M)
[K,E] = ellipke(M,tol)
```

See Also

MATLAB `ellipke` Calling Conventions

Purpose

End of month

C Prototype

```
mxArray *mlfEomday(mxArray *Y, mxArray *M);
```

C Syntax

```
#include "matlab.h"

mxArray *Y, *M;             /* Required input argument(s) */
mxArray *E;                 /* Return value */

E = mlfEomday(Y,M);
```

MATLAB Syntax

```
E = eomday(Y,M)
```

See Also

MATLAB eomday Calling Conventions

mlfEps

Purpose

Floating-point relative accuracy

C Prototype

```
mxArray *mlfEps();
```

C Syntax

```
#include "matlab.h"

mxArray *R;             /* Return value */

R = mlfEps();
```

MATLAB Syntax

```
eps
```

See Also

MATLAB `eps` Calling Conventions

Purpose

Error functions

C Prototype

```
mxArray *mlfErf(mxArray *X);
mxArray *mlfErfc(mxArray *X);
mxArray *mlfErfcx(mxArray *X);
mxArray *mlfErfinv(mxArray *Y);
```

C Syntax

```
#include "matlab.h"

mxArray *X, *Y;

Y = mlfErf(X);          /* Error function */
Y = mlfErfc(X);         /* Complementary error function */
Y = mlfErfcx(X);        /* Scaled complementary error function */
X = mlfErfinv(Y);       /* Inverse of the error function */
```

MATLAB Syntax

```
Y = erf(X)              /* Error function */
Y = erfc(X)             /* Complementary error function */
Y = erfcx(X)            /* Scaled complementary error function */
X = erfinv(Y)           /* Inverse of the error function */
```

See Also

MATLAB `erf`, `erfc`, `erfx`, `erfinv`Calling Conventions

mlfError

Purpose

Display error messages

C Prototype

```
void mlfError(mxArray *mssg);
```

C Syntax

```
#include "matlab.h"

mxArray *mssg;              /* String array(s) */

mlfError(mssg);
```

MATLAB Syntax

```
error('error_message')
```

MATLAB `error` Calling Conventions

Purpose

Elapsed time

C Prototype

```
mxArray *mlfEtime(mxArray *t2, mxArray *t1);
```

C Syntax

```
#include "matlab.h"

mxArray *t2, *t1;       /* Required input argument(s) */
mxArray *e;             /* Return value */

e = mlfEtime(t2,t1);
```

MATLAB Syntax

```
e = etime(t2,t1)
```

See Also

MATLAB `etime` Calling Conventions

mlfExp

Purpose

Exponential

C Prototype

```
mxArray *mlfExp(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfExp(X);
```

MATLAB Syntax

```
Y = exp(X)
```

See Also

MATLAB `exp` Calling Conventions

Purpose

Exponential integral

C Prototype

```
mxArray *mlfExpint(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfExpint(X);
```

MATLAB Syntax

```
Y = expint(X)
```

See Also

MATLAB `expint` Calling Conventions

mlfExpm

Purpose

Matrix exponential

C Prototype

```
mxArray *mlfExpm(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfExpm(X);
```

MATLAB Syntax

```
Y = expm(X)
```

See Also

MATLAB `expm` Calling Conventions

Purpose

Matrix exponential via Pade approximation.

C Prototype

```
mxArray *mlfExpm1(mxArray *A);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *E;             /* Return value */

E = mlfExpm1(A);
```

MATLAB Syntax

```
E = expm1(A)
```

See Also

MATLAB `expm1` Calling Conventions

mlfExpm2

Purpose

Matrix exponential via Taylor series.

C Prototype

```
mxArray *mlfExpm2(mxArray *A);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *E;             /* Return value */

E = mlfExpm2(A);
```

MATLAB Syntax

```
E = expm2(A)
```

See Also

MATLAB `expm2` Calling Conventions

Purpose

Matrix exponential via eigenvalues and eigenvectors.

C Prototype

```
mxArray *mlfExpm3(mxArray *A);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *E;             /* Return value */

E = mlfExpm3(A);
```

MATLAB Syntax

```
E = expm3(A)
```

See Also

MATLAB `expm3` Calling Conventions

mlfEye

Purpose

Identity matrix

C Prototype

```
mxArray *mlfEye(mxArray *m, mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *n, *m, *A, *S;     /* Input argument(s) */
mxArray *Y;                 /* Return value */

Y = mlfEye(n,NULL);
Y = mlfEye(m,n);

S = mlfSize(NULL,A,NULL);
Y = mlfEye(S,NULL);
```

MATLAB Syntax

```
Y  =  eye(n)
Y  =  eye(m,n)
Y  =  eye(size(A))
```

See Also

MATLAB eye Calling Conventions

Purpose

Prime factors

C Prototype

```
mxArray *mlfFactor(mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *n;                 /* Required input argument(s) */
mxArray *f;                 /* Return value */

f = mlfFactor(n);
```

MATLAB Syntax

```
f = factor(n)
```

See Also

MATLAB `factor` Calling Conventions

mlfFclose

Purpose

Close one or more open files

C Prototype

```
mxArray *mlfFclose(mxArray *fid);
```

C Syntax

```
#include "matlab.h"

mxArray *all_str;       /* String array(s) */
mxArray *fid;           /* Required input argument(s) */
mxArray *status;        /* Return value */

status = mlfFclose(fid);

all_str = mxCreateString("all");
status = mlfFclose(all_str);
```

MATLAB Syntax

```
status = fclose(fid)
status = fclose('all')
```

See Also

MATLAB `fclose` Calling Conventions

Purpose

Test for end-of-file

C Prototype

```
mxArray *mlfFeof(mxArray *fid);
```

C Syntax

```
#include "matlab.h"

mxArray *fid;           /* Required input argument(s) */
mxArray *eofstat;       /* Return value */

eofstat = mlfFeof(fid);
```

MATLAB Syntax

```
eofstat = feof(fid)
```

See Also

MATLAB `feof` Calling Conventions

mlfFerror

Purpose

Query MATLAB about errors in file input or output

C Prototype

```
mxArray *mlfFerror(mxArray **errnum, mxArray *fid,
                   mxArray *clear);
```

C Syntax

```
#include "matlab.h"

mxArray *fid;           /* Required input argument(s) */
mxArray *clear;         /* Optional input argument(s) */
mxArray *errnum;        /* Optional output argument(s) */
mxArray *message;       /* Return value */

message = mlfFerror(NULL,fid,NULL);
message = mlfFerror(NULL,fid,clear);
message = mlfFerror(&errnum,fid,NULL);
message = mlfFerror(&errnum,fid,clear);
```

MATLAB Syntax

```
message = ferror(fid)
message = ferror(fid,'clear')
[message,errnum] = ferror(...)
```

See Also

MATLAB `ferror` Calling Conventions

Purpose

Function evaluation. Set nrhs and nlhs to the number of arguments in the input (rhs) and output (lhs) argument arrays.

C Prototype

```
void mlfFeval(int nlhs, mxArray **lhs, int nrhs, mxArray **rhs,
              char *name);
```

C Syntax

```
#include "matlab.h"

char *name;                 /* Name of function to be executed */
int nlhs, nrhs;
mxArray *lhs[1], *rhs[2];   /* 1 and 2 specific to this example */
mxArray *A, *B;             /* Input arguments */
mxArray *x;                 /* Return from the executed function */

char *name = "foo";
nlhs = 1;
nrhs = 2;
rhs[0] = A;
rhs[1] = B;
mlfFeval(nlhs,lhs,nrhs,rhs,name);
x = lhs[0];
```

MATLAB Syntax

```
[y1,y2, ...] = feval(function,x1,...,xn)
```

See Also

MATLAB `feval` Calling Conventions

mlfFft

Purpose

One-dimensional fast Fourier transform

C Prototype

```
mxArray *mlfFft(mxArray *X, mxArray *n, mxArray *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *X;                     /* Required input argument(s) */
mxArray *n, *dim;               /* Optional input argument(s) */
mxArray *null_matrix, *Zero;    /* Optional input argument(s) */
mxArray *Y;                     /* Return value */

Y = mlfFft(X,NULL,NULL);
Y = mlfFft(X,n,NULL);

Zero = mlfScalar(0);
null_matrix = mlfZeros(Zero,Zero);
Y = mlfFft(X,null_matrix,dim);

Y = mlfFft(X,n,dim);
```

MATLAB Syntax

```
Y = fft(X)
Y = fft(X,n)
Y = fft(X,[],dim)
Y = fft(X,n,dim)
```

See Also

MATLAB `fft` Calling Conventions

Purpose Two-dimensional fast Fourier transform

C Prototype

```
mxArray *mlfFft2(mxArray *X, mxArray *m, mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *m, *n;         /* Optional input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfFft2(X,NULL,NULL);
Y = mlfFft2(X,m,n);
```

MATLAB Syntax

```
Y = fft2(X)
Y = fft2(X,m,n)
```

See Also MATLAB `fft2` Calling Conventions

mlfFftshift

Purpose

Shift DC component of fast Fourier transform to center of spectrum

C Prototype

```
mxArray *mlfFftshift(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfFftshift(X);
```

MATLAB Syntax

```
Y = fftshift(X)
```

See Also

MATLAB `fftshift` Calling Conventions

Purpose

Read line from file, discard newline character

C Prototype

```
mxArray *mlfFgetl(mxArray *fid);
```

C Syntax

```
#include "matlab.h"

mxArray *fid;           /* Required input argument(s) */
mxArray *line;          /* Return value */

line = mlfFgetl(fid);
```

MATLAB Syntax

```
line = fgetl(fid)
```

See Also

MATLAB `fgetl` Calling Conventions

mlfFgets

Purpose

Return the next line of a file as a string with line terminator(s)

C Prototype

```
mxArray *mlfFgets(mxArray **EOL, mxArray *fid, mxArray *nchar);
```

C Syntax

```
#include "matlab.h"

mxArray *fid;           /* Required input argument(s) */
mxArray *nchar;         /* Optional input argument(s) */
mxArray *EOL;           /* Optional output argument(s) */
mxArray *line;          /* Return value */

line = mlfFgets(NULL,fid,NULL);
line = mlfFgets(NULL,fid,nchar);
line = mlfFgets(&EOL,fid,NULL);
line = mlfFgets(&EOL,fid,nchar);
```

MATLAB Syntax

```
line = fgets(fid)
line = fgets(fid,nchar)
```

See Also

MATLAB `fgets` Calling Conventions

Purpose

Filter data with an infinite impulse response (IIR) or finite impulse response (FIR) filter

C Prototype

```
mxArray *mlfFilter(mxArray **zf, mxArray *b, mxArray *a,
                   mxArray *X, mxArray *zi, mxArray *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *b, *a, *X;             /* Required input argument(s) */
mxArray *null_array, *Zero;     /* Optional input argument(s) */
mxArray *zi, *dim;              /* Optional input argument(s) */
mxArray *zf;                    /* Optional output argument(s) */
mxArray *y;                     /* Return value */

y = mlfFilter(NULL,b,a,X,NULL,NULL);
y = mlfFilter(&zf,b,a,X,NULL,NULL);
y = mlfFilter(&zf,b,a,X,zi,NULL);

Zero = mlfScalar(0);
null_array = mlfZeros(Zero,Zero);
y = mlfFilter(NULL,b,a,X,null_array,zi);
y = mlfFilter(&zf,b,a,X,null_array,dim);
```

MATLAB Syntax

```
y = filter(b,a,X)
[y,zf] = filter(b,a,X)
[y,zf] = filter(b,a,X,zi)
y = filter(b,a,X,zi,dim)
[...] = filter(b,a,X,[],dim)
```

See Also

MATLAB `filter` Calling Conventions

mlfFilter2

Purpose

Two-dimensional digital filtering

C Prototype

```
mxArray *mlfFilter2(mxArray *h, mxArray *X, mxArray *shape);
```

C Syntax

```
#include "matlab.h"

mxArray *h, *X;         /* Required input argument(s) */
mxArray *shape;         /* Optional input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfFilter2(h,X,NULL);
Y = mlfFilter2(h,X,shape);
```

MATLAB Syntax

```
Y = filter2(h,X)
Y = filter2(h,X,shape)
```

See Also

MATLAB `filter2` Calling Conventions

Purpose

Find indices and values of nonzero elements

C Prototype

```
mxArray *mlfFind(mxArray **j, mxArray **v, mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *j, *v;         /* Optional output argument(s) */
mxArray *k, *i;         /* Return value */

k = mlfFind(NULL,NULL,X);
i = mlfFind(&j,NULL,X);
i = mlfFind(&j,&v,X);
```

MATLAB Syntax

```
k = find(X)
[i,j] = find(X)
[i,j,v] = find(X)
```

See Also

MATLAB `find` Calling Conventions

mlfFindstr

Purpose

Find one string within another

C Prototype

```
mxArray *mlfFindstr(mxArray *str1, mxArray *str2);
```

C Syntax

```
#include "matlab.h"

mxArray *str1, *str2;   /* String array(s) */
mxArray *k;             /* Return value */

k = mlfFindstr(str1,str2);
```

MATLAB Syntax

```
k = findstr(str1,str2)
```

See Also

MATLAB `findstr` Calling Conventions

Purpose

Round towards zero

C Prototype

```
mxArray *mlfFix(mxArray *A);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *B;             /* Return value */

B = mlfFix(A);
```

MATLAB Syntax

```
B = fix(A)
```

See Also

MATLAB `fix` Calling Conventions

mlfFliplr

Purpose

Flip matrices left-right

C Prototype

```
mxArray *mlfFliplr(mxArray *A);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *B;             /* Return value */

B = mlfFliplr(A);
```

MATLAB Syntax

```
B = fliplr(A)
```

See Also

MATLAB `fliplpr` Calling Conventions

Purpose

Flip matrices up-down

C Prototype

```
mxArray *mlfFlipud(mxArray *A);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *B;             /* Return value */

B = mlfFlipud(A);
```

MATLAB Syntax

```
B = flipud(A)
```

See Also

MATLAB `flipud` Calling Conventions

mlfFloor

Purpose

Round towards minus infinity

C Prototype

```
mxArray *mlfFloor (mxArray *A);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *B;             /* Return value */

B = mlfFloor(A);
```

MATLAB Syntax

```
B = floor(A)
```

See Also

MATLAB `floor` Calling Conventions

Purpose

Count floating-point operations

C Prototype

```
mxArray *mlfFlops(mxArray *m);
```

C Syntax

```
#include "matlab.h"

mxArray *m=mlfScalar(0);    /* Optional input argument(s) */
mxArray *f;                 /* Return value */

f = mlfFlops(NULL);
f = mlfFlops(m);
```

MATLAB Syntax

```
f = flops
flops(0)
```

See Also

MATLAB `flops` Calling Conventions

mlfFmin

Purpose

Minimize a function of one variable

C Prototype

```
mxArray *mlfFmin(mxArray **options_out, mxArray *func, mxArray *x1,
                 mxArray *x2, mxArray *options_in, mxArray *P);
```

C Syntax

```
#include "matlab.h"

mxArray *func;          /* String array(s) */
mxArray *x1, *x2;       /* Required input argument(s) */
mxArray *options_in, *P;/* Optional input argument(s) */
mxArray *options_out;   /* Optional output argument(s) */
mxArray *x;             /* Return value */

x = mlfFmin(NULL,func,x1,x2,NULL,NULL);
x = mlfFmin(NULL,func,x1,x2,options_in,NULL);
x = mlfFmin(NULL,func,x1,x2,options_in,P);
x = mlfFmin(&options_out,func,x1,x2,NULL,NULL);
x = mlfFmin(&options_out,func,x1,x2,options_in,NULL);
x = mlfFmin(&options_out,func,x1,x2,options_in,P);
```

MATLAB Syntax

```
x = fmin('fun',x1,x2)
x = fmin('fun',x1,x2,options)
x = fmin('fun',x1,x2,options,P1,P2, ...)
[x,options] = fmin(...)
```

See Also

MATLAB `fmin` Calling Conventions

Purpose

Minimize a function of several variables

C Prototype

```
mxArray *mlfFmins(mxArray **options_out, mxArray *func, mxArray *x0,
                  mxArray *options_in, mxArray *null_matrix,
                  mxArray *P1);
```

C Syntax

```
#include "matlab.h"

mxArray *func;                  /* String array(s) */
mxArray *x0;                    /* Required input argument(s) */
mxArray *options_in, *P1;       /* Optional input argument(s) */
mxArray *null_matrix, *Zero;    /* Optional input argument(s) */
mxArray *options_out;           /* Optional output argument(s) */
mxArray *x;                     /* Return value */

x = mlfFmins(NULL,func,x0,NULL,NULL,NULL);
x = mlfFmins(NULL,func,x0,options_in,NULL,NULL);

Zero = mlfScalar(0);
null_matrix = mlfZeros(Zero,Zero);
x = mlfFmins(NULL,func,x0,options_in,null_matrix,P1);

x = mlfFmins(&options_out,func,x0,NULL,NULL,NULL);
x = mlfFmins(&options_out,func,x0,options_in,NULL,NULL);

Zero = mlfScalar(0);
null_matrix = mlfZeros(Zero,Zero);
x = mlfFmins(&options_out,func,x0,options_in,null_matrix,P1);
```

MATLAB Syntax

```
x = fmins('fun',x0)
x = fmins('fun',x0,options)
x = fmins('fun',x0,options,[],P1,P2, ...)
[x,options] = fmins(...)
```

See Also

MATLAB `fmins` Calling Conventions

mlfFopen

Purpose

Open a file or obtain information about open files

C Prototype

```
mxArray *mlfFopen(mxArray **O1, mxArray **O2, mxArray *I1,
                  mxArray *I2, mxArray *I3);
```

C Syntax

```
#include "matlab.h"

mxArray *format;               /* String array(s) */
mxArray *all_str, *permission; /* String array(s) */
mxArray *message, *filename;   /* String array(s) */
mxArray *fid, *fids;           /* Return value */

fid = mlfFopen(NULL,NULL,filename,permission,NULL);
fid = mlfFopen(&message,NULL,filename,permission,format);

all_str = mxCreateString("all");
fids = mlfFopen(NULL,NULL,all_str,NULL,NULL);

filename = mlfFopen(&permission,&format,fid,NULL,NULL);
```

MATLAB Syntax

```
fid = fopen(filename,permission)
[fid,message] = fopen(filename,permission,format)
fids = fopen('all')
[filename,permission, format] = fopen(fid)
```

See Also

MATLAB `fopen` Calling Conventions

Purpose

Control the output display format. This function's variables contain strings.

C Prototype

```
void mlfFormat(mxArray *a, mxArray *b);
```

C Syntax

```
#include "matlab.h"

mxArray *a, *b;         /* Optional input argument(s) */

mlfFormat(NULL,NULL);
mlfFormat(a,NULL);
mlfFormat(a,b);
```

MATLAB Syntax

MATLAB performs all computations in double precision.

See Also

MATLAB `format` Calling Conventions

mlfFprintf

Purpose

Write formatted data to file. Minimum number of input arguments: one; maximum number of input arguments: user-defined. Terminate the list of arguments with NULL.

C Prototype

```
mxArray *mlfFprintf(mxArray *A,...);
```

C Syntax

```
#include "matlab.h"

mxArray *format;            /* String array(s) */
mxArray *A;                 /* Required input argument(s) */
mxArray *fid, *B, *C;       /* Optional input argument(s) */
mxArray *count, *R;         /* Return value */

count = mlfFprintf(fid,format,A,NULL);
count = mlfFprintf(fid,format,A,B,C,NULL);
R = mlfFprintf(format,A,NULL);
R = mlfFprintf(format,A,B,C,NULL);
```

MATLAB Syntax

```
count = fprintf(fid,format,A,...)
fprintf(format,A,...)
```

See Also

MATLAB `fprintf` Calling Conventions

Purpose

Read binary data from file

C Prototype

```
mxArray *mlfFread(mxArray **count, mxArray *fid, mxArray *size,
                  mxArray *precision, mxArray *skip);
```

C Syntax

```
#include "matlab.h"

mxArray *precision;         /* String array(s) */
mxArray *fid, *size;        /* Required input argument(s) */
mxArray *skip;              /* Optional input argument(s) */
mxArray *count;             /* Required output argument(s) */
mxArray *A;                 /* Return value */

A = mlfFread(&count,fid,size,precision,NULL);
A = mlfFread(&count,fid,size,precision,skip);
```

MATLAB Syntax

```
[A,count] = fread(fid,size,precision)
[A,count] = fread(fid,size,precision,skip)
```

See Also

MATLAB `fread` Calling Conventions

mlfFreqspace

Purpose

Determine frequency spacing for frequency response

C Prototype

```
mxArray *mlfFreqspace(mxArray **f2, mxArray *n, mxArray *whole_str);
```

C Syntax

```
#include "matlab.h"

mxArray *x;                          /* Dimension vector */
mxArray *meshgrid_str, *whole_str;   /* String array(s) */
mxArray *n, *N;                      /* Input argument(s) */
mxArray *f2, *y1;                    /* Optional output argument(s)
  */
mxArray *f1, *x1, *f;                /* Return value */

f1 = mlfFreqspace(&f2,n,NULL);

x = mlfHorzcat(m,n,NULL);
meshgrid_str = mxCreateString("meshgrid");
f1 = mlfFreqspace(&f2,x,NULL);
x1 = mlfFreqspace(&y1,n,meshgrid_str);
x1 = mlfFreqspace(&y1,x,meshgrid_str);

f = mlfFreqspace(NULL,N,NULL);

whole_str = mxCreateString("whole");
f = mlfFreqspace(NULL,N,whole_str);
```

MATLAB Syntax

```
[f1,f2] = freqspace(n)
[f1,f2] = freqspace([m n])
[x1,y1] = freqspace(...,'meshgrid')
f = freqspace(N)
f = freqspace(N,'whole')
```

See Also

MATLAB `freqspace` Calling Conventions

Purpose

Rewind an open file

C Prototype

```
mxArray *mlfFrewind(mxArray *fid);
```

C Syntax

```
#include "matlab.h"

mxArray *fid;           /* Required input argument(s) */
mxArray *R;             /* Return value */

R = mlfFrewind(fid);
```

MATLAB Syntax

```
frewind(fid)
```

See Also

MATLAB `fid` Calling Conventions

mlfFscanf

Purpose

Read formatted data from file

C Prototype

```
mxArray *mlfFscanf(mxArray **count, mxArray *fid, mxArray *format,
                   mxArray *size);
```

C Syntax

```
#include "matlab.h"

mxArray *format;        /* String array(s) */
mxArray *fid;           /* Required input argument(s) */
mxArray *size;          /* Optional input argument(s) */
mxArray *count;         /* Optional output argument(s) */
mxArray *A;             /* Return value */

A = mlfFscanf(NULL,fid,format,NULL);
A = mlfFscanf(&count,fid,format,size);
```

MATLAB Syntax

```
A = fscanf(fid,format)
[A,count] = fscanf(fid,format,size)
```

See Also

MATLAB `fscanf` Calling Conventions

Purpose

Set file position indicator

C Prototype

```
mxArray *mlfFseek(mxArray *fid, mxArray *offset, mxArray *origin);
```

C Syntax

```
#include "matlab.h"

mxArray *origin;            /* String array(s) */
mxArray *fid, *offset;      /* Required input argument(s) */
mxArray *status;            /* Return value */

status = mlfFseek(fid,offset,origin);
```

MATLAB Syntax

```
status = fseek(fid,offset,origin)
```

See Also

MATLAB `fseek` Calling Conventions

mlfFtell

Purpose

Get file position indicator

C Prototype

```
mxArray *mlfFtell (mxArray *fid);
```

C Syntax

```
#include "matlab.h"

mxArray *fid;           /* Required input argument(s) */
mxArray *position;      /* Return value */

position = mlfFtell(fid);
```

MATLAB Syntax

```
position = ftell(fid)
```

See Also

MATLAB `ftell` Calling Conventions

Purpose

Evaluate functions of a matrix

C Prototype

```
mxArray *mlfFunm(mxArray **esterr, mxArray *X, mxArray *func);
```

C Syntax

```
#include "matlab.h"

mxArray *func;          /* String array(s) */
mxArray *X;             /* Required input argument(s) */
mxArray *esterr;        /* Optional output argument(s) */
mxArray *Y;             /* Return value */

Y = mlfFunm(NULL,X,func);
Y = mlfFunm(&esterr,X,func);
```

MATLAB Syntax

```
Y = funm(X,'function')
[Y,esterr] = funm(X,'function')
```

See Also

MATLAB `funm` Calling Conventions

mlfFwrite

Purpose

Write binary data to a file

C Prototype

```
mxArray *mlfFwrite(mxArray *fid, mxArray *A, mxArray *precision,
                   mxArray *skip);
```

C Syntax

```
#include "matlab.h"

mxArray *precision;         /* String array(s) */
mxArray *fid, *A;           /* Required input argument(s) */
mxArray *skip;              /* Optional input argument(s) */
mxArray *count;             /* Return value */

count = mlfFwrite(fid,A,precision,NULL);
count = mlfFwrite(fid,A,precision,skip);
```

MATLAB Syntax

```
count = fwrite(fid,A,precision)
count = fwrite(fid,A,precision,skip)
```

See Also

MATLAB `fwrite` Calling Conventions

Purpose

Zero of a function of one variable

C Prototype

```
mxArray *mlfFzero(mxArray *func, mxArray *x, mxArray *tol,
                  mxArray *trace, mxArray *P1);
```

C Syntax

```
#include "matlab.h"

mxArray *func, *x;          /* Required input argument(s) */
mxArray *tol, *trace, *P1;  /* Optional input argument(s) */
mxArray *z;                 /* Return value */

z = mlfFzero(func,x,NULL,NULL,NULL);
z = mlfFzero(func,x,tol,NULL,NULL);
z = mlfFzero(func,x,tol,trace,NULL);
z = mlfFzero(func,x,tol,trace,P1);
```

MATLAB Syntax

```
z = fzero('fun',x)
z = fzero('fun',x,tol)
z = fzero('fun',x,tol,trace)
z = fzero('fun',x,tol,trace,P1,P2,...)
```

See Also

MATLAB `fzero` Calling Conventions

.

mlfGamma, mlfGammainc, mlfGammaln

Purpose

Gamma functions

C Prototype

```
mxArray *mlfGamma(mxArray *A, mxArray *DoNotUse);
mxArray *mlfGammainc(mxArray *X, mxArray *A);
mxArray *mlfGammaln(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *A, X;              /* Input argument(s) */
mxArray *Y;                 /* Return value */

Y = mlfGamma(A,NULL);       /* Gamma function accepts only */
                            /* one argument */
Y = mlfGammainc(X,A);       /* Incomplete gamma function */
Y = mlfGammaln(A);          /* Logarithm of gamma function */
```

MATLAB Syntax

```
Y = gamma(A)                /* Gamma function */
Y = gammainc(X,A)           /* Incomplete gamma function */
Y = gammaln(A)              /* Logarithm of gamma function */
```

See Also

MATLAB `gamma`, `gammainc`, `gammaln`Calling Conventions

Purpose

Greatest common divisor

C Prototype

```
mxArray *mlfGcd(mxArray **C, mxArray **D, mxArray *A, mxArray *B);
```

C Syntax

```
#include "matlab.h"

mxArray *A, *B;         /* Required input argument(s) */
mxArray *C, *D;         /* Optional output argument(s) */
mxArray *G;             /* Return value */

G = mlfGcd(NULL,NULL,A,B);
G = mlfGcd(&C,&D,A,B);
```

MATLAB Syntax

```
G = gcd(A,B)
[G,C,D] = gcd(A,B)
```

See Also

MATLAB `gcd` Calling Conventions

mlfGradient

Purpose

Numerical gradient

C Prototype

```
mxArray *mlfGradient(mxArray **FY, mxArray *F, mxArray *h,
                     mxArray *h1);
```

C Syntax

```
#include "matlab.h"

mxArray *F;             /* Required input argument(s) */
mxArray *h, *h1;        /* Optional input argument(s) */
mxArray *FY;            /* Optional output argument(s) */
mxArray *FX;            /* Return value */

FX = mlfGradient(NULL,F,NULL,NULL);
FX = mlfGradient(NULL,F,h,NULL);
FX = mlfGradient(&FY,F,NULL,NULL);
FX = mlfGradient(&FY,F,h,NULL);
FX = mlfGradient(&FY,F,h1,h2);
```

MATLAB Syntax

```
FX = gradient(F)
[FX,FY] = gradient(F)
[...] = gradient(F,h)
[...] = gradient(F,h1,h2,...)
```

See Also

MATLAB `gradient` Calling Conventions

Purpose

Data gridding

C Prototype

```
mxArray *mlfGriddata(mxArray **YI, mxArray **ZI, mxArray *x,
                     mxArray *y, mxArray *z, mxArray *xi,
                     mxArray *yi);
```

C Syntax

```
#include "matlab.h"

mxArray *method;            /* String array(s) */
mxArray *x, *y, *z;         /* Required input argument(s) */
mxArray *xi, *yi;           /* Optional input argument(s) */
mxArray *XI, *YI, *ZI;

ZI = mlfGriddata(NULL,NULL,x,y,z,XI,YI);
XI = mlfGriddata(&YI,&ZI,x,y,z,xi,yi);
```

MATLAB Syntax

```
ZI = griddata(x,y,z,XI,YI)
[XI,YI,ZI] = griddata(x,y,z,xi,yi)
```

See Also

MATLAB `griddata` Calling Conventions

mlfHadamard

Purpose

Hadamard matrix

C Prototype

```
mxArray *mlfHadamard(mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *n;             /* Required input argument(s) */
mxArray *H;             /* Return value */

H = mlfHadamard(n);
```

MATLAB Syntax

```
H = hadamard(n)
```

See Also

MATLAB `hadamard` Calling Conventions

Purpose

Hankel matrix

C Prototype

```
mxArray *mlfHankel(mxArray *c, mxArray *r);
```

C Syntax

```
#include "matlab.h"

mxArray *c;             /* Required input argument(s) */
mxArray *r;             /* Optional input argument(s) */
mxArray *H;             /* Return value */

H = mlfHankel(c,NULL);
H = mlfHankel(c,r);
```

MATLAB Syntax

```
H = hankel(c)
H = hankel(c,r)
```

See Also

MATLAB `hankel` Calling Conventions

mlfHess

Purpose

Hessenberg form of a matrix

C Prototype

```
mxArray *mlfHess(mxArray **H, mxArray *A);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *H, *P;

P = mlfHess(&H,A);
H = mlfHess(NULL,A);
```

MATLAB Syntax

```
[P,H] = hess(A)
H = hess(A)
```

See Also

MATLAB `hess` Calling Conventions

Purpose IEEE hexadecimal to decimal number conversion

C Prototype

```
mxArray *mlfHex2dec(mxArray *hex_value);
```

C Syntax

```
#include "matlab.h"

mxArray *hex_value;     /* Hexadecimal integer or string array */
mxArray *d;             /* Return value */

d = mlfHex2dec(hex_value);
```

MATLAB Syntax

```
d = hex2dec('hex_value')
```

See Also MATLAB `hex2dec` Calling Conventions

mlfHex2num

Purpose

Hexadecimal to double number conversion

C Prototype

```
mxArray *mlfHex2num(mxArray *hex_value);
```

C Syntax

```
#include "matlab.h"

mxArray *hex_value;     /* String array(s) */
mxArray *f;             /* Return value */

f = mlfHex2num(hex_value);
```

MATLAB Syntax

```
f = hex2num('hex_value')
```

See Also

MATLAB `hex2num` Calling Conventions

Purpose

Hilbert matrix

C Prototype

```
mxArray *mlfHilb(mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *n;             /* Required input argument(s) */
mxArray *H;             /* Return value */

H = mlfHilb(n);
```

MATLAB Syntax

```
H = hilb(n)
```

See Also

MATLAB `hilb` Calling Conventions

mlfHorzcat

Purpose

Horizontal concatenation. Minimum number of input arguments: two, maximum: user-defined. Terminate all argument lists with a NULL.

C Prototype

```
mxArray *mlfHorzcat(mxArray *A, ...);
```

C Syntax

```
#include "matlab.h"

mxArray *A, *B;             /* Required input argument(s) */
mxArray *C, *D;             /* Optional output argument(s) */
mxArray *R;                 /* Return value */

R = mlfHorzcat(A,B,NULL);
R = mlfHorzcat(A,B,C,D,NULL);
```

MATLAB Syntax

```
[A,B,C...]
horzcat(A,B,C...)
```

See Also

MATLAB `horzcat` Calling Conventions

Purpose

Imaginary unit

C Prototype

```
mxArray *mlfI(void);
```

C Syntax

```
#include "matlab.h"

mxArray *R;                 /* Return value */

R = mlfI();
```

MATLAB Syntax

```
i
```

See Also

MATLAB `i` Calling Conventions

mlfIcubic

Purpose

One-dimensional cubic Interpolation

This MATLAB 4 function has been subsumed into `mlfInterp1`.

See Also

MATLAB `interp1` Calling Conventions

Purpose

Inverse one-dimensional fast Fourier transform

C Prototype

```
mxArray *mlfIfft(mxArray *X, mxArray *n, mxArray *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *X;                     /* Required input argument(s) */
mxArray *n, *dim;               /* Optional input argument(s) */
mxArray *null_matrix, *Zero;    /* Optional input argument(s) */
mxArray *y;                     /* Return value */

y = mlfIfft(X,NULL,NULL);
y = mlfIfft(X,n,NULL);

Zero = mlfScalar(0);
null_matrix = mlfZeros(Zero,Zero);
y = mlfIfft(X,null_matrix,dim);

y = mlfIfft(X,n,dim);
```

MATLAB Syntax

```
y = ifft(X)
y = ifft(X,n)
y = ifft(X,[],dim)
y = ifft(X,n,dim)
```

See Also

MATLAB `ifft` Calling Conventions

mlfIfft2

Purpose

Inverse two-dimensional fast Fourier transform

C Prototype

```
mxArray *mlfIfft2(mxArray *X, mxArray *m, mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *m, *n;         /* Optional input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfIfft2(X,NULL,NULL);
Y = mlfIfft2(X,m,n);
```

MATLAB Syntax

```
Y = ifft2(X)
Y = ifft2(X,m,n)
```

See Also

MATLAB `ifft2` Calling Conventions

Purpose

Imaginary part of a complex number

C Prototype

```
mxArray *mlfImag(mxArray *Z);
```

C Syntax

```
#include "matlab.h"

mxArray *Z;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfImag(Z);
```

MATLAB Syntax

```
Y = imag(Z)
```

See Also

MATLAB `imag` Calling Conventions

mlfInf

Purpose

Infinity

C Prototype

```
mxArray *mlfInf();
```

C Syntax

```
#include "matlab.h"

mxArray *R;             /* Return value */

R = mlfInf();
```

MATLAB Syntax

```
Inf
```

See Also

MATLAB `inf` Calling Conventions

Purpose

Detect points inside a polygonal region

C Prototype

```
mxArray *mlfInpolygon(mxArray *X, mxArray *Y, mxArray *xv,
                      mxArray *yv);
```

C Syntax

```
#include "matlab.h"

mxArray *X, *Y, *xv, *yv;   /* Required input argument(s) */
mxArray *IN;                /* Return value */

IN = mlfInpolygon(X,Y,xv,yv);
```

MATLAB Syntax

```
IN = inpolygon(X,Y,xv,yv)
```

See Also

MATLAB `inpolygon` Calling Conventions

mlfInt2str

Purpose

Integer to string conversion

C Prototype

```
mxArray *mlfInt2str(mxArray *N);
```

C Syntax

```
#include "matlab.h"

mxArray *N;             /* Required input argument(s) */
mxArray *str;           /* Return value */

str = mlfInt2str(N);
```

MATLAB Syntax

```
str = int2str(N)
```

See Also

MATLAB `int2str` Calling Conventions

Purpose

One-dimensional data interpolation (table lookup)

C Prototype

```
mxArray *mlfInterp1(mxArray *x, mxArray *Y, mxArray *xi,
                    mxArray *method);
```

C Syntax

```
#include "matlab.h"

mxArray *x, *Y, *xi;    /* Required input argument(s) */
mxArray *method;        /* String array(s) */
mxArray *yi;            /* Return value */

yi = mlfInterp1(x,Y,xi,NULL);
yi = mlfInterp1(x,Y,xi,method);
```

MATLAB Syntax

```
yi = interp1(x,Y,xi)
yi = interp1(x,Y,xi,method)
```

See Also

MATLAB `interp1` Calling Conventions

mlfInterp1q

Purpose

Quick one-dimensional linear interpolation

C Prototype

```
mxArray *mlfInterp1q(mxArray *x, mxArray *Y, mxArray *xi);
```

C Syntax

```
#include "matlab.h"

mxArray *x, *Y, *xi;    /* Required input argument(s) */
mxArray *F;             /* Return value */

F = mlfInterp1q(x,Y,xi);
```

MATLAB Syntax

```
F = interp1q(x,Y,xi)
```

See Also

MATLAB `interp1q` Calling Conventions

Purpose

Two-dimensional data interpolation (table lookup)

C Prototype

```
mxArray *mlfInterp2(mxArray *X, mxArray *Y, mxArray *Z,
                    mxArray *XI, mxArray *YI, mxArray *method);
```

C Syntax

```
#include "matlab.h"

mxArray *method;            /* String array(s) */
mxArray *Z, *XI, *YI;       /* Input argument(s) */
mxArray *X, *Y, *ntimes;    /* Input argument(s) */
mxArray *ZI;                /* Return value */

ZI = mlfInterp2(X,Y,Z,XI,YI,NULL);
ZI = mlfInterp2(Z,XI,YI,NULL,NULL,NULL);
ZI = mlfInterp2(Z,ntimes,NULL,NULL,NULL,NULL);
ZI = mlfInterp2(X,Y,Z,XI,YI,method);
```

MATLAB Syntax

```
ZI = interp2(X,Y,Z,XI,YI)
ZI = interp2(Z,XI,YI)
ZI = interp2(Z,ntimes)
ZI = interp2(X,Y,Z,XI,YI,method)
```

See Also

MATLAB `interp2` Calling Conventions

mlfInterp4

Purpose

2-D bilinear data interpolation

This MATLAB 4 function has been subsumed by `mlfInterp2` in MATLAB 5.

See Also

MATLAB `interp2` Calling Conventions

Purpose 2-D bicubic data interpolation

This MATLAB 4 function has been subsumed by `mlfInterp2` in MATLAB 5.

See Also MATLAB `interp2` Calling Conventions

mlfInterp6

Purpose

2-D Nearest neighbor interpolation

This MATLAB 4 function has been subsumed by `mlfInterp2` in MATLAB 5.

See Also

MATLAB `interp2` Calling Conventions

Purpose

One-dimensional interpolation using the fast Fourier transform method

C Prototype

```
mxArray *mlfInterpft(mxArray *x, mxArray *n, mxArray *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *x, *n;         /* Required input argument(s) */
mxArray *dim;           /* Optional input argument(s) */
mxArray *y;             /* Return value */

y = mlfInterpft(x,n,NULL);
y = mlfInterpft(x,n,dim);
```

MATLAB Syntax

```
y = interpft(x,n)
y = interpft(x,n,dim)
```

See Also

MATLAB `interpft` Calling Conventions

mlfInv

Purpose

Matrix inverse

C Prototype

```
mxArray *mlfInv(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfInv(X);
```

MATLAB Syntax

```
Y = inv(X)
```

See Also

MATLAB `inv` Calling Conventions

Purpose

Inverse of the Hilbert matrix

C Prototype

```
mxArray *mlfInvhilb(mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *n;             /* Required input argument(s) */
mxArray *H;             /* Return value */

H = mlfInvhilb(n);
```

MATLAB Syntax

```
H = invhilb(n)
```

See Also

MATLAB `invhilb` Calling Conventions

mlfIpermute

Purpose

Inverse permute the dimensions of a multidimensional array

C Prototype

```
mxArray *mlfIpermute(mxArray *B, mxArray *order);
```

C Syntax

```
#include "matlab.h"

mxArray *B, *order;         /* Required input argument(s) */
mxArray *A;                 /* Return value */

A = mlfIpermute(B,order);
```

MATLAB Syntax

```
A = ipermute(B,order)
```

See Also

MATLAB `ipermute` Calling Conventions

Purpose

Detect state

C Prototype

```
mxArray *mlfIschar(mxArray *A);
mxArray *mlfIsempty(mxArray *S);
mxArray *mlfIsequal(mxArray *A, mxArray *B...);
mxArray *mlfIsfinite(mxArray *A);
mxArray *mlfIsieee();
mxArray *mlfIsinf(mxArray *A);
mxArrat *mlfIsletter(mxArray *str);
mxArray *mlfIslogical(mxArray *A);
mxArray *mlfIsnan(mxArray *A);
mxArray *mlfIsnumeric(mxArray *A);
mxArray *mlfIsprime(mxArray *A);
mxArray *mlfIsreal(mxArray *A);
mxArray *mlfIsspace(mxArray *str);
mxArray *mlfIsstudent();
mxArray *mlfIsunix();
mxArray *mlfIsvms();
```

C Syntax

```
#include "matlab.h"

mxArray *str;               /* String array(s) */
mxArray *A, *B, *S;         /* Required input argument(s) */
mxArray *k, *TF;            /* Return value */

k = mlfIschar(S);               k = mlfIsempty(A);
k = mlfIsequal(A,B,NULL);       TF = mlfIsfinite(A);
k = mlfIsieee();                TF = mlfIsinf(A);
TF = mlfIsletter(str);          k = mlfIslogical(A);
TF = mlfIsnan(A);               k = mlfIsnumeric(A);
TF = mlfIsprime(A);             k = mlfIsreal(A);
TF = mlfIsspace(str);           k = mlfIsstudent();
k = mlfIsunix();                k = mlfIsvms();
```

mlfIs*

MATLAB Syntax

```
k = ischar(S)
k = isempty(A)
k = isequal(A,B,...)
k = isieee
TF = isfinite(A)
TF = isinf(A)
TF = isletter('str')
k = islogical(A)
k = isnumeric(A)
TF = isnan(A)
TF = isprime(A)
k = isreal(A)
TF = isspace('str')
k = isstudent
k = isunix
k = isvms
```

See Also MATLAB `is*` Calling Conventions

Purpose

Detect an object of a given class

C Prototype

```
mxArray *mlfIsa(mxArray *obj, mxArray *class_name);
```

C Syntax

```
#include "matlab.h"

mxArray class_name;         /* String array(s) */
mxArray obj;                /* Required input argument(s) */

K = mlfIsa(obj,class_name);
```

MATLAB Syntax

```
K = isa(obj,'class_name')
```

See Also

MATLAB `isa` Calling Conventions

mlfIsmember

Purpose

Detect members of a set

C Prototype

```
mxArray *mlfIsmember(mxArray *a, mxArray *S, mxArray *rows);
```

C Syntax

```
#include "matlab.h"

mxArray *a, *A, *S;         /* Input argument(s) */
mxArray *rows_str;          /* String array(s) */
mxArray *k;                 /* Return value */

k = mlfIsmember(a,S,NULL);

rows_str = mxCreateString("rows");
k = mlfIsmember(A,S,rows_str);
```

MATLAB Syntax

```
k = ismember(a,S)
k = ismember(A,S,'rows')
```

See Also

MATLAB `ismember` Calling Conventions

Purpose

Detect strings

This MATLAB 4 function has been renamed `mlfIsChar` (`mlfIs*`) in MATLAB 5.

See Also

MATLAB `ischar` Calling Conventions

mlfJ

Purpose

Imaginary unit

C Prototype

```
mxArray * mlfJ(void);
```

C Syntax

```
#include "matlab.h"

mxArray *R;             /* Return value */

R = mlfJ();
```

MATLAB Syntax

```
j
```

See Also

MATLAB `j` Calling Conventions

Purpose

Kronecker tensor product

C Prototype

```
mxArray *mlfKron(mxArray *X, mxArray *Y);
```

C Syntax

```
#include "matlab.h"

mxArray *X, *Y;         /* Required input argument(s) */
mxArray *K;             /* Return value */

K = mlfKron(X,Y);
```

MATLAB Syntax

```
K = kron(X,Y)
```

See Also

MATLAB `kron` Calling Conventions

mlfLcm

Purpose

Least common multiple

C Prototype

```
mxArray *mlfLcm(mxArray *A, mxArray *B);
```

C Syntax

```
#include "matlab.h"

mxArray *A, *B;         /* Required input argument(s) */
mxArray *L;             /* Return value */

L = mlfLcm(A,B);
```

MATLAB Syntax

```
L = lcm(A,B)
```

See Also

MATLAB `lcm` Calling Conventions

Purpose

Associated Legendre functions

C Prototype

```
mxArray *mlfLegendre(mxArray *n, mxArray *X, mxArray *sch_str);
```

C Syntax

```
#include "matlab.h"

mxArray *n, *X;             /* Required input argument(s) */
mxArray *sch_str;           /* String array(s) */
mxArray *P, *S;             /* Return value */

P = mlfLegendre(n,X,NULL);

sch_str = mxCreateString("sch");
S = mlfLegendre(n,X,sch_str);
```

MATLAB Syntax

```
P = legendre(n,X)
S = legendre(n,X,'sch')
```

See Also

MATLAB `legendre` Calling Conventions

mlfLength

Purpose

Length of vector

C Prototype

```
mxArray *mlfLength(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *n;             /* Return value */

n = mlfLength(X);
```

MATLAB Syntax

```
n = length(X)
```

See Also

MATLAB `length` Calling Conventions

Purpose

Linear to mu-law conversion

C Prototype

```
mxArray *mlfLin2mu(mxArray *y);
```

C Syntax

```
#include "matlab.h"

mxArray *y;                 /* Required input argument(s) */
mxArray *mu;                /* Return value */

mu = mlfLin2mu(y);
```

MATLAB Syntax

```
mu = lin2mu(y)
```

See Also

MATLAB `lin2mu` Calling Conventions

mlfLinspace

Purpose

Generate linearly spaced vectors

C Prototype

```
mxArray *mlfLinspace(mxArray *a, mxArray *b, mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *a, *b;         /* Required input argument(s) */
mxArray *n;             /* Optional input argument(s) */
mxArray *y;             /* Return value */

y = mlfLinspace(a,b,NULL);
y = mlfLinspace(a,b,n);
```

MATLAB Syntax

```
y = linspace(a,b)
y = linspace(a,b,n)
```

See Also

MATLAB `linspace` Calling Conventions

Purpose

Load variables from disk. Terminate the argument list to `mlfLoad()` with `NULL`.

C Prototype

```
void mlfLoad(const char *file, ... );
```

C Syntax

```
#include "matlab.h"

char *file
mxArray *x, *y, *z;     /* Output arguments */

mlfLoad(file,"X",&x,NULL);
mlfLoad(file,"X",&x,"Y",&y,"Z",&z,NULL);
.
.
.
```

MATLAB Syntax

```
load fname X
load fname X,Y,Z
load fname X,Y,Z...
```

See Also

MATLAB `load` Calling Conventions

mlfLog

Purpose

Natural logarithm

C Prototype

```
mxArray *mlfLog(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfLog(X);
```

MATLAB Syntax

```
Y = log(X)
```

See Also

MATLAB `log` Calling Conventions

Purpose

Base 2 logarithm and dissect floating-point numbers into exponent and mantissa

C Prototype

```
mxArray *mlfLog2(mxArray **E, mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *E;             /* Optional output argument(s) */
mxArray *Y, *F;         /* Return value */

Y = mlfLog2(NULL,X);
F = mlfLog2(&E,X);
```

MATLAB Syntax

```
Y = log2(X)
[F,E] = log2(X)
```

See Also

MATLAB `log2` Calling Conventions

mlfLog10

Purpose

Common (base 10) logarithm

C Prototype

```
mxArray *mlfLog10(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfLog10(X);
```

MATLAB Syntax

```
Y = log10(X)
```

See Also

MATLAB `log10` Calling Conventions

Purpose

Convert numeric values to logical

C Prototype

```
mxArray *mlfLogical(mxArray *A);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *K;             /* Return value */

K = mlfLogical(A);
```

MATLAB Syntax

```
K = logical(A)
```

See Also

MATLAB `logical` Calling Conventions

mlfLogm

Purpose

Matrix logarithm

C Prototype

```
mxArray *mlfLogm(mxArray **esterr, mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *esterr;        /* Optional output argument(s) */
mxArray *Y;             /* Return value */

Y = mlfLogm(NULL,X);
Y = mlfLogm(&esterr,X);
```

MATLAB Syntax

```
Y = logm(X)
[Y,esterr] = logm(X)
```

See Also

MATLAB `logm` Calling Conventions

Purpose

Generate logarithmically spaced vectors

C Prototype

```
mxArray *mlfLogspace(mxArray *a, mxArray *b, mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *a, *b, *n, *pi;/* Input argument(s) */
mxArray *y;             /* Return value */

y = mlfLogspace(a,b,NULL);
y = mlfLogspace(a,b,n);
y = mlfLogspace(a,pi,NULL);
```

MATLAB Syntax

```
y = logspace(a,b)
y = logspace(a,b,n)
y = logspace(a,pi)
```

See Also

MATLAB `logspace` Calling Conventions

mlfLower

Purpose

Convert string to lower case

C Prototype

```
mxArray *mlfLower(mxArray *str);
```

C Syntax

```
#include "matlab.h"

mxArray *str;              /* String array(s) */
mxArray *t;                /* Return value */

t = mlfLower(str);
```

MATLAB Syntax

```
t = lower('str')
```

See Also

MATLAB `lower` Calling Conventions

Purpose

Least squares solution in the presence of known covariance

C Prototype

```
mxArray *mlfLscov(mxArray **dx, mxArray *A, mxArray *b, mxArray *V);
```

C Syntax

```
#include "matlab.h"

mxArray *A, *b, *V;         /* Required input argument(s) */
mxArray *dx;                /* Optional output argument(s) */
mxArray *x;                 /* Return value */

x = mlfLscov(NULL,A,b,V);
x = mlfLscov(&dx,A,b,V);
```

MATLAB Syntax

```
x = lscov(A,b,V)
[x,dx] = lscov(A,b,V)
```

See Also

MATLAB `lscov` Calling Conventions

mlfLu

Purpose

LU matrix factorization

C Prototype

```
mxArray *mlfLu(mxArray **U, mxArray **P, mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *U, *P;         /* Optional output argument(s) */
mxArray *L;             /* Return value */

L = mlfLu(&U,NULL,X);
L = mlfLu(&U,&P,X);
L = mlfLu(NULL,NULL,X);
```

MATLAB Syntax

```
[L,U] = lu(X)
[L,U,P] = lu(X)
lu(X)
```

See Also

MATLAB `lu` Calling Conventions

Purpose

Magic square

C Prototype

```
mxArray *mlfMagic(mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *n;             /* Required input argument(s) */
mxArray *M;             /* Return value */

M = mlfMagic(n);
```

MATLAB Syntax

```
M = magic(n)
```

See Also

MATLAB `magic` Calling Conventions

mlfMat2str

Purpose

Convert a matrix into a string

C Prototype

```
mxArray *mlfMat2str(mxArray *A, mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *n;             /* Optional input argument(s) */
mxArray *str;           /* Return value */

str = mlfMat2str(A,NULL);
str = mlfMat2str(A,n);
```

MATLAB Syntax

```
str = mat2str(A)
str = mat2str(A,n)
```

See Also

MATLAB `mat2str` Calling Conventions

Purpose

Maximum elements of an array

C Prototype

```
mxArray *mlfMax(mxArray **I, mxArray *A, mxArray *B, mxArray *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *A;                     /* Required input argument(s) */
mxArray *B, *dim;               /* Optional input argument(s) */
mxArray *null_matrix, *Zero;    /* Optional input argument(s) */
mxArray *I;                     /* Optional output argument(s) */
mxArray *C;                     /* Return value */

C = mlfMax(NULL,A,NULL,NULL);
C = mlfMax(NULL,A,B,NULL);

Zero = mlfScalar(0);
null_matrix = mlfZeros(Zero,Zero);
C = mlfMax(NULL,A,null_matrix,dim);

C = mlfMax(&I,A,NULL,NULL);
C = mlfMax(&I,A,null_matrix,dim);
```

MATLAB Syntax

```
C = max(A)
C = max(A,B)
C = max(A,[],dim)
[C,I] = max(...)
```

See Also

MATLAB `max` Calling Conventions

mlfMean

Purpose

Average or mean value of arrays

C Prototype

```
mxArray *mlfMean(mxArray *A, mxArray *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *dim;           /* Optional input argument(s) */
mxArray *M;             /* Return value */

M = mlfMean(A,NULL);
M = mlfMean(A,dim);
```

MATLAB Syntax

```
M = mean(A)
M = mean(A,dim)
```

See Also

MATLAB `mean` Calling Conventions

Purpose

Median value of arrays

C Prototype

```
mxArray *mlfMedian(mxArray *A, mxArray *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *dim;           /* Optional input argument(s) */
mxArray *M;             /* Return value */

M = mlfMedian(A,NULL);
M = mlfMedian(A,dim);
```

MATLAB Syntax

```
M = median(A)
M = median(A,dim)
```

See Also

MATLAB `median` Calling Conventions

mlfMeshgrid

Purpose

Generate X and Y matrices for three-dimensional plots

C Prototype

```
mxArray *mlfMeshgrid(mxArray **Y, mxArray **Z, mxArray *x,
                     mxArray *y, mxArray *z);
```

C Syntax

```
#include "matlab.h"

mxArray *x;             /* Required input argument(s) */
mxArray *y, *z;         /* Optional input argument(s) */
mxArray *Y;             /* Required output argument(s) */
mxArray *Z;             /* Optional output argument(s) */
mxArray *X;             /* Return value */

X = mlfMeshgrid(&Y,NULL,x,y,NULL);
X = mlfMeshgrid(&Y,NULL,x,NULL,NULL);
X = mlfMeshgrid(&Y,&Z,x,y,z);
```

MATLAB Syntax

```
[X,Y] = meshgrid(x,y)
[X,Y] = meshgrid(x)
[X,Y,Z] = meshgrid(x,y,z)
```

See Also

MATLAB `meshgrid` Calling Conventions

Purpose

The name of the currently running M-file

C Prototype

```
mxArray *mlfMfilename();
```

C Syntax

```
#include "matlab.h"

mxArray *R;             /* Return value */

R = mlfMfilename();
```

MATLAB Syntax

```
mfilename
```

Description

`mlfMfilename` returns a string containing a NULL matrix. There is no current M-file when a C application executes. For those standalone programs built by compiling M-files with the MATLAB compiler, the MATLAB compiler will translate the call into the appropriate string. Otherwise, for standalone programs built by hand, `mfilename` returns an empty string.

See Also

Calling Conventions

mlfMin

Purpose

Minimum elements of an array

C Prototype

```
mxArray *mlfMin(mxArray **I, mxArray *A, mxArray *B, mxArray *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *A;                     /* Required input argument(s) */
mxArray *B, *dim;               /* Optional input argument(s) */
mxArray *null_matrix, *Zero;    /* Optional input argument(s) */
mxArray *I;                     /* Optional output argument(s) */
mxArray *C;                     /* Return value */

C = mlfMin(NULL,A,NULL,NULL);
C = mlfMin(NULL,A,B,NULL);

Zero = mlfScalar(0);
null_matrix = mlfZeros(Zero,Zero);
C = mlfMin(NULL,A,null_matrix,dim);

C = mlfMin(&I,A,NULL,NULL);
C = mlfMin(&I,A,null_matrix,dim);
```

MATLAB Syntax

```
C = min(A)
C = min(A,B)
C = min(A,[],dim)
[C,I] = min(...)
```

See Also

MATLAB `min` Calling Conventions

Purpose

Modulus (signed remainder after division)

C Prototype

```
mxArray *mlfMod(mxArray *X, mxArray *Y);
```

C Syntax

```
#include "matlab.h"

mxArray *X, *Y;             /* Required input argument(s) */
mxArray *M;                 /* Return value */

M = mlfMod(X,Y);
```

MATLAB Syntax

```
M = mod(X,Y)
```

See Also

MATLAB `mod` Calling Conventions

mlfMu2lin

Purpose

Mu-law to linear conversion

C Prototype

```
mxArray *mlfMu2lin(mxArray *mu);
```

C Syntax

```
#include "matlab.h"

mxArray *mu;                /* Required input argument(s) */
mxArray *y;                 /* Return value */

y = mlfmu2lin(mu);
```

MATLAB Syntax

```
y = mu2lin(mu)
```

See Also

MATLAB `mu2lin` Calling Conventions

Purpose

Not-a-Number

C Prototype

```
mxArray *mlfNan();
```

C Syntax

```
#include "matlab.h"

mxArray *R;             /* Return value */

R = mlfNan();
```

MATLAB Syntax

```
NaN
```

See Also

MATLAB `NaN`, Calling Conventions

mlfNargchk

Purpose

Check number of input arguments

C Prototype

```
mxArray *mlfNargchk(mxArray *low, mxArray *high, mxArray *number);
```

C Syntax

```
#include "matlab.h"

mxArray *low, *high, *number;   /* Required input argument(s) */
mxArray *msg;                   /* Return value */

msg = mlfNargchk(low,high,number);
```

MATLAB Syntax

```
msg = nargchk(low,high,number)
```

See Also

MATLAB `nargchk` Calling Conventions

Purpose

All combinations of the *n* elements in v taken *k* at a time

C Prototype

```
mxArray *mlfNchoosek(mxArray *v, mxArray *k);
```

C Syntax

```
#include "matlab.h"

mxArray *v;                 /* Required input vector of length n */
mxArray *k;                 /* Required input scalar, group size */
mxArray *C;                 /* Output array of combinations */

C = mlfNchoosek(v,k);
```

MATLAB Syntax

```
C = nchoosek(v,k)
```

See Also

MATLAB nchoosek Calling Conventions

mlfNdims

Purpose

Number of array dimensions

C Prototype

```
mxArray *mlfNdims(mxArray *A);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *n;             /* Return value */

n = mlfNdims(A);
```

MATLAB Syntax

```
n = ndims(A)
```

Description

This function always returns 2 for version 1.2 of the Math Library.

See Also

MATLAB `ndims` Calling Conventions

Purpose

Next power of two

C Prototype

```
mxArray *mlfNextpow2(mxArray *A);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *p;             /* Return value */

p = mlfNextpow2(A);
```

MATLAB Syntax

```
p = nextpow2(A)
```

See Also

MATLAB `nextpow2` Calling Conventions

mlfNnls

Purpose

Nonnegative least squares

C Prototype

```
mxArray *mlfNnls(mxArray **w, mxArray *A, mxArray *b, mxArray *tol);
```

C Syntax

```
#include "matlab.h"

mxArray *A, *b;         /* Required input argument(s) */
mxArray *tol;           /* Optional input argument(s) */
mxArray *w;             /* Optional output argument(s) */
mxArray *x;             /* Return value */

x = mlfNnls(NULL,A,b,NULL);
x = mlfNnls(NULL,A,b,tol);
x = mlfNnls(&w,A,b,NULL);
x = mlfNnls(&w,A,b,tol);
```

MATLAB Syntax

```
x = nnls(A,b)
x = nnls(A,b,tol)
[x,w] = nnls(A,b)
[x,w] = nnls(A,b,tol)
```

See Also

MATLAB `nnls` Calling Conventions

Purpose

Vector and matrix norms

C Prototype

```
mxArray *mlfNorm(mxArray *A, mxArray *p);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *p;             /* Optional input argument(s) */
mxArray *n;             /* Return value */

n = mlfNorm(A,NULL);
n = mlfNorm(A,p);
```

MATLAB Syntax

```
n = norm(A)
n = norm(A,p)
```

See Also

MATLAB `norm` Calling Conventions

mlfNormest

Purpose

Two-norm estimate

C Prototype

```
mxArray *mlfNormest(mxArray **count, mxArray *S, mxArray *tol);
```

C Syntax

```
#include "matlab.h"

mxArray *S;             /* Required input argument(s) */
mxArray *tol;           /* Optional input argument(s) */
mxArray *count;         /* Optional output argument(s) */
mxArray *nrm;           /* Return value */

nrm = mlfNormest(NULL,S,NULL);
nrm = mlfNormest(NULL,S,tol);
nrm = mlfNormest(&count,S,NULL);
nrm = mlfNormest(&count,S,tol);
```

MATLAB Syntax

```
nrm = normest(S)
nrm = normest(S,tol)
[nrm,count] = normest(...)
```

See Also

MATLAB `normest` Calling Conventions

Purpose

Current date and time

C Prototype

```
mxArray *mlfNow();
```

C Syntax

```
#include "matlab.h"

mxArray *t;                 /* Return value */

t = now();
```

MATLAB Syntax

```
t = now
```

See Also

MATLAB now Calling Conventions

mlfNull

Purpose

Null space of a matrix

C Prototype

```
mxArray *mlfNull(mxArray *A, mxArray *how);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *how;           /* Optional input argument(s) */
mxArray *B;             /* Return value */

B = mlfNull(A);
B = mlfNull(A,how);
```

MATLAB Syntax

```
B = null(A)
```

See Also

MATLAB `null` Calling Conventions

Purpose

Number to string conversion

C Prototype

```
mxArray *mlfNum2str(mxArray *A, mxArray *format);
```

C Syntax

```
#include "matlab.h"

mxArray *format;                /* String array(s) */
mxArray *A;                     /* Required input argument(s) */
mxArray *precision;             /* Optional input argument(s) */
mxArray *str;                   /* Return value */

str = mlfNum2str(A,NULL);
str = mlfNum2str(A,precision);
str = num2str(A,format);
```

MATLAB Syntax

```
str = num2str(A)
str = num2str(A,precision)
str = num2str(A,format)
```

See Also

MATLAB `num2str` Calling Conventions

mlfOde45, mlfOde23, mlfOde113, mlfOde15s, mlfOde23s

Purpose

Solve differential equations

C Prototype

Substitute `mlfOde45`, `mlfOde23`, etc. for *solver*.

```
mxArray *solver(mxArray **Y, mxArray **TE, mxArray **YE,
                mxArray **IE, mxArray **O6, mxArray *func,
                mxArray *tspan, mxArray *y0, mxArray *options,
                mxArray *p);
```

C Syntax

```
#include "matlab.h"

mxArray *func, *model;              /* String array(s) */
mxArray *tspan, *y0, *options, *p;  /* Input argument(s) */
mxArray *X, *Y, *TE, *YE, *IE;      /* Output arguments */
mxArray *T;                         /* Return value */

T = solver(&Y,NULL,NULL,NULL,NULL,func,tspan,y0,NULL,NULL);
T = solver(&Y,NULL,NULL,NULL,NULL,func,tspan,y0,options,NULL);
T = solver(&Y,NULL,NULL,NULL,NULL,func,tspan,y0,options,p);
T = solver(&Y,&TE,&YE,&IE,NULL,func,tspan,y0,options,NULL);
T = solver(&X,&Y,NULL,NULL,NULL,model,tspan,y0,options,p);
```

MATLAB Syntax

```
[T,Y] = solver('F',tspan,y0)
[T,Y] = solver('F',tspan,y0,options)
[T,Y] = solver('F',tspan,y0,options,p1,p2...)
[T,Y,TE,YE,IE] = solver('F',tspan,y0,options)
[T,X,Y] = solver('model',tspan,y0,options,ut,p1,p2,...)
```

See Also

MATLAB `ode45`, `ode23`, `ode113`, `ode15s`, `ode23s`Calling Conventions

Purpose

Extract properties from options structure created with odeset

C Prototype

```
mxArray *mlfOdeget(mxArray *options, mxArray *name_str,
                   mxArray *default);
```

C Syntax

```
#include "matlab.h"

mxArray *name_str;          /* String array(s) */
mxArray *options;           /* Required input argument(s) */
mxArray *default;           /* Optional input argument(s) */
mxArray *o;                 /* Return value */

o = mlfOdeget(options,name_str,NULL);
o = mlfOdeget(options,name_str,default);
```

MATLAB Syntax

```
o = odeget(options,'name')
o = odeget(options,'name',default)
```

See Also

MATLAB odeget Calling Conventions

mlfOdeset

Purpose

Create or alter options structure for input to ODE solvers

C Prototype

```
mxArray *mlfOdeset(mxArray *I1, mxArray *I2, mxArray *I3,
                   mxArray *I4, mxArray *I5, mxArray *I6,
                   mxArray *I7, mxArray *I8, mxArray *I9,
                   mxArray *I10, mxArray *I11, mxArray *I12,
                   mxArray *I13, mxArray *I14, mxArray *I15,
                   mxArray *I16, mxArray *I17, mxArray *I18,
                   mxArray *I19, mxArray *I20, mxArray *I21,
                   mxArray *I22, mxArray *I25, mxArray *I26,
                   mxArray *I27, mxArray *I28, mxArray *I29,
                   mxArray *I30, mxArray *I31);
```

C Syntax

```
#include "matlab.h"

mxArray *name1,...*name15;          /* String array(s) */
mxArray *value1...*value15;         /* Input values */
mxArray *oldopts, *newopts;         /* Input argument(s) */
mxArray *options;                   /* Return value */

options = mlfOdeset(name1,value1,name2,value2,NULL,NULL,...NULL);
options = mlfOdeset(oldopts,name1,value1,NULL,NULL,...NULL);
options = mlfOdeset(oldopts,newopts,NULL,NULL,...NULL);
options = mlfOdeset();
```

MATLAB Syntax

```
options = odeset('name1',value1,'name2',value2,...)
options = odeset(oldopts,'name1',value1,...)
options = odeset(oldopts,newopts)
odeset
```

See Also

MATLAB odeset Calling Conventions

Purpose

Create an array of all ones

C Prototype

```
mxArray *mlfOnes(mxArray *m, mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *x;                 /* Dimension vector */
mxArray *m, *n;             /* Input argument(s) */
mxArray *S, *A;             /* Input argument(s) */
mxArray *Y;                 /* Return value */

Y = mlfOnes(n,NULL);
Y = mlfOnes(m,n);

x = mlfHorzcat(m,n,NULL);
Y = mlfOnes(x,NULL);

S = mlfSize(NULL,A,NULL);
Y = mlfOnes(S,NULL);
```

MATLAB Syntax

```
Y = ones(n)
Y = ones(m,n)
Y = ones([m n])
Y = ones(size(A))
```

See Also

MATLAB ones Calling Conventions

mlfOrth

Purpose

Range space of a matrix

C Prototype

```
mxArray *mlfOrth(mxArray *A);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *B;             /* Return value */

B = mlfOrth(A);
```

MATLAB Syntax

```
B = orth(A)
```

See Also

MATLAB `orth` Calling Conventions

Purpose

Pascal matrix

C Prototype

```
mxArray *mlfPascal(mxArray *n, mxArray *k);
```

C Syntax

```
#include "matlab.h"

mxArray *n;             /* Required input argument(s) */
mxArray *one, *two;     /* Optional input argument(s) */
mxArray *A;             /* Return value */

one = mlfScalar(1);
two = mlfScalar(2);

A = mlfPascal(n,NULL);
A = mlfPascal(n,one);
A = mlfPascal(n,two);
```

MATLAB Syntax

```
A = pascal(n)
A = pascal(n,1)
A = pascal(n,2)
```

See Also

MATLAB `pascal` Calling Conventions

mlfPerms

Purpose

All possible permutations

C Prototype

```
mxArray *mlfPerms(mxArray * );
```

C Syntax

```
#include "matlab.h"

mxArray *v;                 /* Required input argument(s) */
mxArray *P;                 /* Return value */

P = mlfPerms(v);
```

MATLAB Syntax

```
P = perms(v)
```

See Also

MATLAB `perms` Calling Conventions

Purpose

Rearrange the dimensions of a multidimensional array

C Prototype

```
mxArray *mlfPermute(mxArray *A, mxArray *order);
```

C Syntax

```
#include "matlab.h"

mxArray *A, *order;     /* Required input argument(s) */
mxArray *A;             /* Return value */

B = mlfPermute(A,order);
```

MATLAB Syntax

```
B = permute(A,order)
```

See Also

MATLAB `permute` Calling Conventions

mlfPi

Purpose

Ratio of a circle's circumference to its diameter, π

C Prototype

```
mxArray *mlfPi();
```

C Syntax

```
#include "matlab.h"

mxArray *R;             /* Return value */

R = mlfPi();
```

MATLAB Syntax

```
pi
```

See Also

MATLAB `pi` Calling Conventions

Purpose

Moore-Penrose pseudoinverse of a matrix

C Prototype

```
mxArray *mlfPinv(mxArray *A, mxArray *tol);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *tol;           /* Optional input argument(s) */
mxArray *B;             /* Return value */

B = mlfPinv(A,NULL);
B = mlfPinv(A,tol);
```

MATLAB Syntax

```
B = pinv(A)
B = pinv(A,tol)
```

See Also

MATLAB `pinv` Calling Conventions

mlfPlanerot

Purpose

Given's plane rotation.

C Prototype

```
mxArray *mlfPlanerot(mxArray **y, mxArray *x);
```

C Syntax

```
#include "matlab.h"

mxArray *x;                 /* Required input argument(s) */
mxArray *y;                 /* Required output argument(s) */
mxArray *g;                 /* Return value */

g = mlfPlanerot(&y,x);
```

MATLAB Syntax

```
[g,y] = planerot(x)
```

See Also

MATLAB `planerot` Calling Conventions

Purpose

Transform polar or cylindrical coordinates to Cartesian

C Prototype

```
mxArray *mlfPol2cart(mxArray **Y, mxArray **Z_out, mxArray *THETA,
                     mxArray *RHO, mxArray *Z_in);
```

C Syntax

```
#include "matlab.h"

mxArray *THETA, *RHO;   /* Required input argument(s) */
mxArray *Z_in;          /* Optional input argument(s) */
mxArray *Y;             /* Required output argument(s) */
mxArray *Z_out;         /* Optional output argument(s) */
mxArray *X;             /* Return value */

X = mlfPol2cart(&Y,NULL,THETA,RHO,NULL);
X = mlfPol2cart(&Y,&Z_out,THETA,RHO,Z_in);
```

MATLAB Syntax

```
[X,Y] = pol2cart(THETA,RHO)
[X,Y,Z] = pol2cart(THETA,RHO,Z)
```

See Also

MATLAB `pol2cart` Calling Conventions

mlfPoly

Purpose

Polynomial with specified roots

C Prototype

```
mxArray *mlfPoly(mxArray *A);
```

C Syntax

```
#include "matlab.h"

mxArray *A, *r;         /* Input argument(s) */
mxArray *p;             /* Return value */

p = mlfPoly(A);
p = mlfPoly(r);
```

MATLAB Syntax

```
p = poly(A)
p = poly(r)
```

See Also

MATLAB `poly` Calling Conventions

Purpose

Area of polygon

C Prototype

```
mxArray *mlfPolyarea(mxArray *X, mxArray *Y, mxArray *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *X, *Y;             /* Required input argument(s) */
mxArray *dim;               /* Optional input argument(s) */
mxArray *A;                 /* Return value */

A = mlfPolyarea(X,Y,NULL);
A = mlfPolyarea(X,Y,dim);
```

MATLAB Syntax

```
A = polyarea(X,Y)
A = polyarea(X,Y,dim)
```

See Also

MATLAB `polyarea` Calling Conventions

mlfPolyder

Purpose

Polynomial derivative

C Prototype

```
mxArray *mlfPolyder(mxArray **d, mxArray *a, mxArray *b);
```

C Syntax

```
#include "matlab.h"

mxArray *a, *b, *p;         /* Input argument(s) */
mxArray *k, *q;             /* Return value */

k = mlfPolyder(NULL,p,NULL);
k = mlfPolyder(NULL,a,b);
q = mlfPolyder(&d,b,a);
```

MATLAB Syntax

```
k = polyder(p)
k = polyder(a,b)
[q,d] = polyder(b,a)
```

See Also

MATLAB `polyder` Calling Conventions

Purpose

Polynomial eigenvalue problem

C Prototype

```
mxArray *mlfPolyeig(mxArray **e, mxArray *A0, mxArray *A1,
                    mxArray *A2, mxArray *A3, mxArray *A4,
                    mxArray *A5, mxArray *A6, mxArray *A7,
                    mxArray *A8, mxArray *A9, mxArray *A10,
                    mxArray *A11, mxArray *A12, mxArray *A13,
                    mxArray *A14, mxArray *A15, mxArray *A16,
                    mxArray *A17, mxArray *A18, mxArray *A19,
                    mxArray *A20);
```

C Syntax

```
#include "matlab.h"

mxArray *A0, *A1, *A2, *A3, *A4;    /* Required input argument(s)
   */
mwArray *A5, *A6, *A7, *A8, *A9;    /* Required input argument(s)
   */
mxArray *A10, *A11, *A12, *A13;     /* Required input argument(s)
   */
mxArray *A14, *A15, *A16, *A17;     /* Required input argument(s)
   */
mxArray *A18, *A19, *A20;           /* Required input argument(s)
   */
mxArray *e;                         /* Required output argument(s)
   */
mxArray *X;                         /* Return value */

X = mlfPolyeig(&e,A0,A1,...A20);
```

MATLAB Syntax

```
[X,e] = polyeig(A0,A1,...Ap)
```

See Also

MATLAB `polyeig` Calling Conventions

mlfPolyfit

Purpose

Polynomial curve fitting

C Prototype

```
mxArray *mlfPolyfit(mxArray **s, mxArray *x, mxArray *y,
                    mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *x, *y, *n;         /* Required input argument(s) */
mxArray *s;                 /* Optional output argument(s) */
mxArray *p;                 /* Return value */

p = mlfPolyfit(NULL,x,y,n);
p = mlfPolyfit(&s,x,y,n);
```

MATLAB Syntax

```
p = polyfit(x,y,n)
[p,s] = polyfit(x,y,n)
```

See Also

MATLAB `polyfit` Calling Conventions

Purpose

Polynomial evaluation

C Prototype

```
mxArray *mlfPolyval(mxArray **delta, mxArray *p, mxArray *x,
                    mxArray *S);
```

C Syntax

```
#include "matlab.h"

mxArray *p, *x;         /* Required input argument(s) */
mxArray *S;             /* Optional input argument(s) */
mxArray *delta;         /* Optional output argument(s) */
mxArray *y;             /* Return value */

y = mlfPolyval(NULL,p,x,NULL);
y = mlfPolyval(&delta,p,x,S);
```

MATLAB Syntax

```
y = polyval(p,x)
[y,delta] = polyval(p,x,S)
```

See Also

MATLAB `polyval` Calling Conventions

mlfPolyvalm

Purpose

Matrix polynomial evaluation

C Prototype

```
mxArray *mlfPolyvalm(mxArray *p, mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *p, *X;         /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfPolyvalm(p,X);
```

MATLAB Syntax

```
Y = polyvalm(p,X)
```

See Also

MATLAB `polyvalm` Calling Conventions

Purpose

Base 2 power and scale floating-point numbers

C Prototype

```
mxArray *mlfPow2(mxArray *F, mxArray *E);
```

C Syntax

```
#include "matlab.h"

mxArray *Y, *F, *E;         /* Input argument(s) */
mxArray *X;                 /* Return value */

X = mlfPow2(Y,NULL);
X = mlfPow2(F,E);
```

MATLAB Syntax

```
X = pow2(Y)
X = pow2(F,E)
```

See Also

MATLAB `pow2` Calling Conventions

mlfPrimes

Purpose

Generate list of prime numbers

C Prototype

```
mxArray *mlfPrimes(mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *n;                 /* Required input argument(s) */
mxArray *p;                 /* Return value */

p = mlfPrimes(n);
```

MATLAB Syntax

```
p = primes(n)
```

See Also

MATLAB `primes` Calling Conventions

Purpose

Product of array elements

C Prototype

```
mxArray *mlfProd(mxArray *A, *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *A;                 /* Required input argument(s) */
mxArray *dim;               /* Optional input argument(s) */
mxArray *B;                 /* Return value */

B = mlfProd(A,NULL);
B = mlfProd(A,dim);
```

MATLAB Syntax

```
B = prod(A)
B = prod(A,dim)
```

See Also

MATLAB `prod` Calling Conventions

mlfQr

Purpose

Orthogonal-triangular decomposition

C Prototype

```
mxArray *mlfQr(mxArray **R, mxArray **E, mxArray *X, mxArray *Zero);
```

C Syntax

```
#include "matlab.h"

mxArray *X;                 /* Required input argument(s) */
mxArray *Zero;              /* Optional input argument(s) */
mxArray *R, *E;             /* Optional output argument(s) */
mxArray *Q, *A;             /* Return value */

Q = mlfQr(&R,NULL,X,NULL);
Q = mlfQr(&R,&E,X,NULL);
Q = mlfQr(&R,NULL,X,Zero);
Q = mlfQr(&R,&E,X,Zero);
A = mlfQr(NULL,NULL,X,NULL);
```

MATLAB Syntax

```
[Q,R] = qr(X)
[Q,R,E] = qr(X)
[Q,R] = qr(X,0)
[Q,R,E] = qr(X,0)
A = qr(X)
```

See Also

MATLAB `qr` Calling Conventions

Purpose

Delete column from QR factorization

C Prototype

```
mxArray *mlfQrdelete(mxArray **R_out, mxArray *Q_in, mxArray *R_in,
                     mxArray *j);
```

C Syntax

```
#include "matlab.h"

mxArray *Q_in, *R_in, *j;   /* Required input argument(s) */
mxArray *R_out;             /* Required output argument(s) */
mxArray *Q;                 /* Return value */

Q = mlfQrdelete(&R_out,Q_in,R_in,j);
```

MATLAB Syntax

```
[Q,R] = qrdelete(Q,R,j);
```

See Also

MATLAB `qrdelete` Calling Conventions

mlfQrinsert

Purpose

Insert column in QR factorization

C Prototype

```
mxArray *mlfQrinsert(mxArray **R_out, mxArray *Q_in, mxArray *R_in,
                     mxArray *j, mxArray *x);
```

C Syntax

```
#include "matlab.h"

mxArray *Q_in, *R_in, *j, *x;   /* Required input argument(s) */
mxArray *R_out;                 /* Required output argument(s) */
mxArray *Q;                     /* Return value */

Q = mlfQrinsert(&R_out,Q_in,R_in,j,x);
```

MATLAB Syntax

```
[Q,R] = qrinsert(Q,R,j,x)
```

See Also

MATLAB `qrinsert` Calling Conventions

Purpose

Numerical evaluation of integrals

C Prototype

```
mxArray *mlfQuad(mxArray **cnt_lhs_, mxArray *func, mxArray *a,
                 mxArray *b, mxArray *tol, mxArray *trace,
                 mxArray *P);
mxArray *mlfQuad8(mxArray **cnt_lhs_, mxArray *func, mxArray *a,
                 mxArray *b, mxArray *tol, mxArray *trace,
                 mxArray *P);
```

C Syntax

```
#include "matlab.h"

mxArray *func;              /* String array(s) */
mxArray *a, *b, *tol;       /* Required input argument(s) */
mxArray *trace, *P;         /* Optional input argument(s) */
mxArray *q;                 /* Return value */

q = mlfQuad(func,a,b,NULL,NULL,NULL);
q = mlfQuad(func,a,b,tol,NULL,NULL);
q = mlfQuad(func,a,b,tol,trace,NULL);
q = mlfQuad(func,a,b,tol,trace,P);

q = mlfQuad8(func,a,b,NULL,NULL,NULL);
q = mlfQuad8(func,a,b,tol,NULL,NULL);
q = mlfQuad8(func,a,b,tol,trace,NULL);
q = mlfQuad8(func,a,b,tol,trace,P);
```

MATLAB Syntax

```
q = quad('fun',a,b)
q = quad('fun',a,b,tol)
q = quad('fun',a,b,tol,trace)
q = quad('fun',a,b,tol,trace,P1,P2,...)
q = quad8(...)
```

See Also

MATLAB quad, quad8 Calling Conventions

mlfQz

Purpose

QZ factorization for generalized eigenvalues

C Prototype

```
mxArray *mlfQz(mxArray **BB, mxArray **Q, mxArray **Z, mxArray **V,
               mxArray *A, mxArray *B);
```

C Syntax

```
#include "matlab.h"

mxArray *A, *B;             /* Required input argument(s) */
mxArray *BB, *Q, *Z, *V;    /* Required output argument(s) */
mxArray *AA;                /* Return value */

AA = mlfQz(&BB,&Q,&Z,&V,A,B);
```

MATLAB Syntax

```
[AA,BB,Q,Z,V] = qz(A,B)
```

See Also

MATLAB qz Calling Conventions

Purpose

Uniformly distributed random numbers and arrays

C Prototype

```
mxArray *mlfRand(mxArray *m, mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *x;                 /* Dimension vector */
mxArray *state_str;         /* String array(s) */
mxArray *size, *state;      /* Input argument(s) */
mxArray *m, *n, *A;         /* Input argument(s) */
mxArray *Y;                 /* Return value */

Y = mlfRand(n,NULL);
Y = mlfRand(m,n);

x = mlfHorzcat(m,n,NULL);
Y = mlfRand(x,NULL);

size = mlfSize(NULL,A,NULL);
Y = mlfRand(size,NULL);

Y = mlfRand(NULL,NULL);

state_str = mxCreateString("state");
Y = mlfRand(state_str,state);
```

MATLAB Syntax

```
Y = rand(n)
Y = rand(m,n)
Y = rand([m n])
Y = rand(size(A))
rand
s = rand('state')
```

See Also

MATLAB `rand` Calling Conventions

mlfRandn

Purpose

Normally distributed random numbers and arrays

C Prototype

```
mxArray *mlfRandn(mxArray *m, mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *m, *n;         /* Optional input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfRandn(n,NULL);
Y = mlfRandn(m,n);
Y = mlfRandn(NULL,NULL);
```

MATLAB Syntax

```
Y = randn(n)
Y = randn(m,n)
randn
```

See Also

MATLAB `randn` Calling Conventions

Purpose

Rank of a matrix

C Prototype

```
mxArray *mlfRank(mxArray *A, mxArray *tol);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *tol;           /* Optional input argument(s) */
mxArray *k;             /* Return value */

k = mlfRank(A,NULL);
k = mlfRank(A,tol);
```

MATLAB Syntax

```
k = rank(A)
k = rank(A,tol)
```

See Also

MATLAB `rank` Calling Conventions

mlfRat, mlfRats

Purpose

Rational fraction approximation

C Prototype

```
mxArray *mlfRat(mxArray **D, mxArray *X, mxArray *tol);
mxArray *mlfRats(mxArray *X, mxArray *strlength);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *tol;           /* Optional input for mlfRat */
mxArray *strlength;     /* Optional input for mlfRats */
mxArray *D;             /* Required output argument for mlfRat */
mxArray *N, *str;       /* Return values for mlfRat */
mxArray *S;             /* Return value for mlfRats */

N = mlfRat(&D,X,NULL);
N = mlfRat(&D,X,tol);
str = mlfRat(NULL,X,NULL);
str = mlfRat(NULL,X,tol);

S = mlfRats(X,strlength);
S = mlRats(X,NULL);
```

MATLAB Syntax

```
[N,D] = rat(X)
[N,D] = rat(X,tol)
rat(...)
S = rats(X,strlength)
S = rats(X)
```

See Also

MATLAB `rat`, `rats` Calling Conventions

Purpose

Matrix reciprocal condition number estimate

C Prototype

```
mxArray *mlfRcond(mxArray *A);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *c;             /* Return value */

c = mlfRcond(A);
```

MATLAB Syntax

```
c = rcond(A)
```

See Also

MATLAB `rcond` Calling Conventions

mlfReal

Purpose

Real part of complex number

C Prototype

```
mxArray *mlfReal(mxArray *Z);
```

C Syntax

```
#include "matlab.h"

mxArray *Z;             /* Required input argument(s) */
mxArray *X;             /* Return value */

X = mlfReal(Z);
```

MATLAB Syntax

```
X = real(Z)
```

See Also

MATLAB `real` Calling Conventions

Purpose

Largest positive floating-point number

C Prototype

```
mxArray *mlfRealmax();
```

C Syntax

```
#include "matlab.h"

mxArray *n;             /* Return value */

n = mlfRealmax();
```

MATLAB Syntax

```
n = realmax
```

See Also

MATLAB `realmax` Calling Conventions

mlfRealmin

Purpose

Smallest positive floating-point number

C Prototype

```
mxArray *mlfRealmin();
```

C Syntax

```
#include "matlab.h"

mxArray *n;             /* Return value */

n = mlfRealmin();
```

MATLAB Syntax

```
n = realmin
```

See Also

MATLAB `realmin` Calling Conventions

Purpose

Rectangle intersection area

C Prototype

```
mxArray *mlfRectint(mxArray *a, mxArray *b);
```

C Syntax

```
#include "matlab.h"

mxArray *a, *b;         /* Required input argument(s) */
mxArray *R;             /* Return value */

R = mlfRectint(a,b);
```

MATLAB Syntax

```
rectint(a,b)
```

See Also

MATLAB `rectint` Calling Conventions

mlfRem

Purpose

Remainder after division

C Prototype

```
mxArray *mlfRem(mxArray *X, mxArray *Y);
```

C Syntax

```
#include "matlab.h"

mxArray *X, *Y;         /* Required input argument(s) */
mxArray *R;             /* Return value */

R = mlfRem(X,Y);
```

MATLAB Syntax

```
R = rem(X,Y)
```

See Also

MATLAB `rem` Calling Conventions

Purpose

Replicate and tile an array

C Prototype

```
mxArray *mlfRepmat(mxArray *A, mxArray *m, mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *x;                 /* Dimension vector */
mxArray *A, *m, *n;         /* Input argument(s) */
mxArray *B;                 /* Return value */

B = mlfRepmat(A,m,n);

x = mlfHorzcat(m,n,NULL);
B = mlfRepmat(A,x,NULL);
```

MATLAB Syntax

```
B = repmat(A,m,n)
B = repmat(A,[m n])
```

See Also

MATLAB `repmat` Calling Conventions

mlfReshape

Purpose

Reshape array. Terminate the list of arguments with a NULL.

C Prototype

```
mxArray *mlfReshape(mxArray *A, mxArray *m, ...);
```

C Syntax

```
#include "matlab.h"

mxArray *A;                 /* Required input argument(s) */
mxArray *m, *n, *siz;       /* Optional input argument(s) */
mxArray *B;                 /* Return value */

B = mlfReshape(A,m,n,NULL);
B = mlfReshape(A,siz,NULL);
```

MATLAB Syntax

```
B = reshape(A,m,n)
B = reshape(A,siz)
```

See Also

MATLAB `reshape` Calling Conventions

Purpose

Residue of a repeated pole

C Prototype

```
mxArray *mlfResi2(mxArray *u, mxArray *v, mxArray *pole, mxArray *n,
                  mxArray *k);
```

C Syntax

```
#include "matlab.h"

mxArray *u, *v, *pole;      /* Required input argument(s) */
mxArray *n, *k;             /* Required input argument(s) */
mxArray *R;                 /* Return value */

R = mlfResi2(u,v,pole,n,k);
```

MATLAB Syntax

```
resi2(u,v,pole,n,k)
```

See Also

MATLAB `resi2` Calling Conventions

mlfResidue

Purpose

Convert between partial fraction expansion and polynomial coefficients

C Prototype

```
mxArray *mlfResidue(mxArray **O1, mxArray **O2, mxArray *I1,
                    mxArray *I2, mxArray *I3);
```

C Syntax

```
#include "matlab.h"

mxArray *r, *p, *k;
mxArray *b, *a;

r = mlfResidue(&p,&k,b,a,NULL);
b = mlfResidue(&a,NULL,r,p,k);
```

MATLAB Syntax

```
[r,p,k] = residue(b,a)
[b,a] = residue(r,p,k)
```

See Also

MATLAB `residue` Calling Conventions

Purpose

Polynomial roots

C Prototype

```
mxArray *mlfRoots(mxArray *c);
```

C Syntax

```
#include "matlab.h"

mxArray *c;             /* Required input argument(s) */
mxArray *r;             /* Return value */

r = mlfRoots(c);
```

MATLAB Syntax

```
r = roots(c)
```

See Also

MATLAB `roots` Calling Conventions

mlfRosser

Purpose

Classic symmetric eigenvalue test matrix (Rosser matrix)

C Prototype

```
mxArray *mlfRosser();
```

C Syntax

```
#include "matlab.h"

mxArray c;                  /* Required input argument(s) */
mxArray A;                  /* Return value */

A = mlfRosser();
```

MATLAB Syntax

```
[A,B,C,...] = gallery('tmfun',P1,P2,...)
gallery(3)              a badly conditioned 3-by-3 matrix
gallery(5)              an interesting eigenvalue problem
```

Description

`A = mlfRosser();` returns the Rosser matrix. This matrix was a challenge for many matrix eigenvalue algorithms. But the Francis QR algorithm, as perfected by Wilkinson and implemented in EISPACK and MATLAB, has no trouble with it. The matrix is 8-by-8 with integer elements. It has:

- A double eigenvalue
- Three nearly equal eigenvalues
- Dominant eigenvalues of opposite sign
- A zero eigenvalue
- A small, nonzero eigenvalue

See Also

MATLAB `gallery` Calling Conventions

Purpose

Rotate matrix 90 degrees

C Prototype

```
mxArray *mlfRot90(mxArray *A, mxArray *k);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *k;             /* Optional input argument(s) */
mxArray *B;             /* Return value */

B = mlfRot90(A,NULL);
B = mlfRot90(A,k);
```

MATLAB Syntax

```
B = rot90(A)
B = rot90(A,k)
```

See Also

MATLAB `rot90` Calling Conventions

mlfRound

Purpose

Round to nearest integer

C Prototype

```
mxArray *mlfRound(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfRound(X);
```

MATLAB Syntax

```
Y = round(X)
```

See Also

MATLAB `round` Calling Conventions

Purpose

Reduced row echelon form

C Prototype

```
mxArray *mlfRref(mxArray **jb, mxArray *A, mxArray *tol);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *tol;           /* Optional input argument(s) */
mxArray *jb;            /* Optional output argument(s) */
mxArray *R;             /* Return value */

R = mlfRref(NULL,A,NULL);
R = mlfRref(&jb,A,NULL);
R = mlfRref(&jb,A,tol);
```

MATLAB Syntax

```
R = rref(A)
[R,jb] = rref(A)
[R,jb] = rref(A,tol)
```

See Also

MATLAB `rref` Calling Conventions

mlfRsf2csf

Purpose

Convert real Schur form to complex Schur form

C Prototype

```
mxArray *mlfRsf2csf(mxArray **T_out, mxArray *U_in, mxArray *T_in);
```

C Syntax

```
#include "matlab.h"

mxArray *U_in, *T_in;       /* Required input argument(s) */
mxArray *T_out;             /* Required output argument(s) */
mxArray *U_out;             /* Return value */

U_out = mlfRsf2csf(&T_out,U_in,T_in);
```

MATLAB Syntax

```
[U,T] = rsf2csf(U,T)
```

See Also

MATLAB `rsf2csf` Calling Conventions

Purpose

Save variables to disk. Terminate the argument list to `mlfSave()` with a `NULL`.

C Prototype

```
void mlfSave(const char *file, const char *mode, ... );
```

C Syntax

```
#include "matlab.h"

char *file * mode;
mxArray *x, *y, *z;

mlfSave(file,"w","X",x,NULL);           /* overwrite data */
mlfSave(file,"u","X",x,"Y",y,"Z",z,NULL);/* append to data */
```

MATLAB Syntax

```
save fname X
save fname X,Y,Z
```

See Also

MATLAB save Calling Conventions

mlfSchur

Purpose

Schur decomposition

C Prototype

```
mxArray *mlfSchur(mxArray **T, mxArray *A);
```

C Syntax

```
#include "matlab.h"

mxArray *A;                 /* Required input argument(s) */
mxArray *T, *U;             /* Return value */

U = mlfSchur(&T,A);
T = mlfSchur(NULL,A);
```

MATLAB Syntax

```
[U,T] = schur(A)
T = schur(A)
```

See Also

MATLAB `schur` Calling Conventions

Purpose

Secant and hyperbolic secant

C Prototype

```
mxArray *mlfSec(mxArray *X);
mxArray *mlfSech(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfSec(X);
Y = mlfSech(X);
```

MATLAB Syntax

```
Y = sec(X)
Y = sech(X)
```

See Also

MATLAB sec, sech Calling Conventions

mlfSetdiff

Purpose

Return the set difference of two vectors

C Prototype

```
mxArray *mlfSetdiff(mxArray **i, mxArray *A, mxArray *B,
                    mxArray *rows_str);
```

C Syntax

```
#include "matlab.h"

mxArray *rows_str;          /* String array(s) */
mxArray *a, *b, *A, *B;     /* Input argument(s) */
mxArray *i;                 /* Optional output argument(s) */
mxArray *c;                 /* Return value */

c = mlfSetdiff(NULL,a,b,NULL);

rows_str = mxCreateString("rows");
c = mlfSetdiff(NULL,A,B,rows_str);

c = mlfSetdiff(&i,a,b,NULL);
c = mlfSetdiff(&i,A,B,rows_str);
```

MATLAB Syntax

```
c = setdiff(a,b)
c = setdiff(A,B,'rows')
[c,i] = setdiff(...)
```

See Also

MATLAB `setdiff` Calling Conventions

Purpose

Set string flag

This MATLAB 4 function has been renamed `mlfChar` in MATLAB 5.

See Also

MATLAB `mlfChar` Calling Conventions

mlfSetxor

Purpose

Set exclusive-or of two vectors

C Prototype

```
mxArray *mlfSetxor(mxArray **ia, mxArray **ib, mxArray *A,
                   mxArray *B, mxArray *rows_str);
```

C Syntax

```
#include "matlab.h"

mxArray *rows_str;          /* String array(s) */
mxArray *a, *b, *A, *B;     /* Input argument(s) */
mxArray *ia, *ib;           /* Optional output argument(s) */
mxArray *c;                 /* Return value */

c = mlfSetxor(NULL,NULL,a,b,NULL);

rows_str = mlfChar("rows");
c = mlfSetxor(NULL,NULL,A,B,rows_str);

c = mlfSetxor(&ia,&ib,a,b,NULL);
c = mlfSetxor(&ia,&ib,A,B,rows_str);
```

MATLAB Syntax

```
c = setxor(a,b)
c = setxor(A,B,'rows')
[c,ia,ib] = setxor(...)
```

See Also

MATLAB `setxor` Calling Conventions

Purpose

Shift dimensions

C Prototype

```
mxArray *mlfShiftdim(mxArray **nshifts, mxArray *X, mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *X;                 /* Required input argument(s) */
mxArray *n;                 /* Optional input argument(s) */
mxArray *nshifts;           /* Optional output argument(s) */
mxArray *B;                 /* Return value */

B = mlfShiftdim(NULL,X,n);
B = mlfShiftdim(&nshifts,X,NULL);
```

MATLAB Syntax

```
B = shiftdim(X,n)
[B,nshifts] = shiftdim(X)
```

See Also

MATLAB `shiftdim` Calling Conventions

mlfSign

Purpose

Signum function

C Prototype

```
mxArray *mlfSign(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfSign(X);
```

MATLAB Syntax

```
Y = sign(X)
```

See Also

MATLAB `sign` Calling Conventions

Purpose

Sine and hyperbolic sine

C Prototype

```
mxArray *mlfSin(mxArray *X);
mxArray *mlfSinh(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfSin(X);
Y = mlfSinh(X);
```

MATLAB Syntax

```
Y = sin(X)
Y = sinh(X)
```

See Also

MATLAB `sin`, `sinh` Calling Conventions

mlfSize

Purpose

Array dimensions

C Prototype

```
mxArray *mlfSize(mxArray **n, mxArray *X, mxArray *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *dim;           /* Optional input argument(s) */
mxArray *n;             /* Optional output argument(s) */
mxArray *d, *m;         /* Return value */

d = mlfSize(NULL,X,NULL);
m = mlfSize(&n,X,NULL);
m = mlfSize(NULL,X,dim);
```

MATLAB Syntax

```
d = size(X)
[m,n] = size(X)
m = size(X,dim)
```

See Also

MATLAB `size` Calling Conventions

Purpose

Sort elements in ascending order

C Prototype

```
mxArray *mlfSort(mxArray **INDEX, mxArray *A, mxArray *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *dim;           /* Optional input argument(s) */
mxArray *INDEX;         /* Optional output argument(s) */
mxArray *B;             /* Return value */

B = mlfSort(NULL,A,NULL);
B = mlfSort(&INDEX,A,NULL);
B = mlfSort(NULL,A,dim);
```

MATLAB Syntax

```
B = sort(A)
[B,INDEX] = sort(A)
B = sort(A,dim)
```

See Also

MATLAB `sort` Calling Conventions

mlfSortrows

Purpose

Sort rows in ascending order

C Prototype

```
mxArray *mlfSortrows(mxArray **index, mxArray *A, mxArray *column);
```

C Syntax

```
#include "matlab.h"

mxArray *A;                 /* Required input argument(s) */
mxArray *column;            /* Optional input argument(s) */
mxArray *index;             /* Optional output argument(s) */
mxArray *B;                 /* Return value */

B = mlfSortrows(NULL,A,NULL);
B = mlfSortrows(NULL,A,column);
B = mlfSortrows(&index,A,column);
```

MATLAB Syntax

```
B = sortrows(A)
B = sortrows(A,column)
[B,index] = sortrows(A)
```

See Also

MATLAB `sortrows` Calling Conventions

Purpose

Transform spherical coordinates to Cartesian

C Prototype

```
mxArray *mlfSph2cart(mxArray **y, mxArray **z, mxArray *THETA,
                     mxArray *PHI, mxArray *R);
```

C Syntax

```
#include "matlab.h"

mxArray *THETA, *PHI, *R;   /* Required input argument(s) */
mxArray *y, *z;             /* Required output argument(s) */
mxArray *x;                 /* Return value */

x = mlfSph2cart(&y,&z,THETA,PHI,R);
```

MATLAB Syntax

```
[x,y,z] = sph2cart(THETA,PHI,R)
```

See Also

MATLAB `sph2cart` Calling Conventions

mlfSpline

Purpose

Cubic spline interpolation

C Prototype

```
mxArray *mlfSpline(mxArray *x, mxArray *y, mxArray *xi);
```

C Syntax

```
#include "matlab.h"

mxArray *x, *y;             /* Required input argument(s) */
mxArray *xi;                /* Optional input argument(s) */
mxArray *yi, *pp;           /* Return value */

yi = mlfSpline(x,y,xi);
pp = mlfSpline(x,y,NULL);
```

MATLAB Syntax

```
yi = spline(x,y,xi)
pp = spline(x,y)
```

See Also

MATLAB `spline` Calling Conventions

Purpose

Write formatted data to a string. Minimum number of input arguments: two, maximum: user-defined. Terminate the list of arguments with a `NULL`.

C Prototype

```
mxArray *mlfSprintf(mxArray **errmsg,...);
```

C Syntax

```
#include "matlab.h"

mxArray *format;            /* String array(s) */
mxArray *A, *B;             /* Input argument(s) */
mxArray *errmsg;            /* Optional output argument(s) */
mxArray *s;                 /* Return value */

s = mlfSprintf(NULL,format,A,NULL);
s = mlfSprintf(&errmsg,format,A,NULL);
s = mlfSprintf(&errmsg,format,A,B,NULL);
```

MATLAB Syntax

```
s = sprintf(format,A,...)
[s,errmsg] = sprintf(format,A,...)
```

See Also

MATLAB `sprintf` Calling Conventions

mlfSqrt

Purpose

Square root

C Prototype

```
mxArray *mlfSqrt(mxArray *A);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *B;             /* Return value */

B = mlfSqrt(A);
```

MATLAB Syntax

```
B = sqrt(A)
```

See Also

MATLAB `sqrt` Calling Conventions

Purpose

Matrix square root

C Prototype

```
mxArray *mlfSqrtm(mxArray **esterr, mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *esterr;        /* Optional output argument(s) */
mxArray *Y;             /* Return value */

Y = mlfSqrtm(NULL,X);
Y = mlfSqrtm(&esterr,X);
```

MATLAB Syntax

```
Y = sqrtm(X)
[Y,esterr] = sqrtm(X)
```

See Also

MATLAB `sqrtm` Calling Conventions

mlfSscanf

Purpose

Read string under format control

C Prototype

```
mxArray *mlfSscanf(mxArray **count, mxArray **errmsg,
                   mxArray **nextindex, mxArray *s,
                   mxArray *format, mxArray *size);
```

C Syntax

```
#include "matlab.h"

mxArray *format;                    /* String array(s) */
mxArray *s;                         /* Required input argument(s)
  */
mxArray *size;                      /* Optional input argument(s)
  */
mxArray *count, *errmsg, *nextindex;/* Optional output argument(s)
  */
mxArray *A;                         /* Return value */

A = mlfSscanf(NULL,NULL,NULL,s,format,NULL);
A = mlfSscanf(NULL,NULL,NULL,s,format,size);
A = mlfSscanf(&count,&errmsg,&nextindex,s,format,NULL);
A = mlfSscanf(&count,&errmsg,&nextindex,s,format,size);
```

MATLAB Syntax

```
A = sscanf(s,format)
A = sscanf(s,format,size)
[A,count,errmsg,nextindex] = sscanf(...)
```

See Also

MATLAB sscanf Calling Conventions

Purpose

Standard deviation

C Prototype

```
mxArray *mlfStd(mxArray *x, mxArray *flag, mxArray *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *x;                 /* Required input argument(s) */
mxArray *flag, *dim;        /* Optional input argument(s) */
mxArray *s;                 /* Return value */

s = mlfStd(x,NULL,NULL);
s = mlfStd(x,flag,NULL);
s = mlfStd(x,flag,dim);
```

MATLAB Syntax

```
s = std(X)
s = std(X,flag)
s = std(X,flag,dim)
```

See Also

MATLAB `std` Calling Conventions

mlfStr2mat

Purpose

Form blank padded character matrix from strings.

C Prototype

```
mxArray * mlfStr2mat(mxArray *str1, ... );
```

C Syntax

```
#include "matlab.h"

mxArray *str1, mxArray *str2;   /* String array(s) */
mxArray *S;                     /* Return value */

S = mlfStr2mat(str1,NULL);
S = mlfStr2mat(str1,str2,NULL);
.
.
.
```

MATLAB Syntax

```
S = str2mat(t1,t2,t3,...)
```

See Also

MATLAB `str2mat` Calling Conventions

Purpose

String to number conversion

C Prototype

```
mxArray *mlfStr2num(mxArray *str);
```

C Syntax

```
#include "matlab.h"

mxArray *str;           /* String array(s) */
mxArray *x;             /* Return value */

x = mlfStr2num(str);
```

MATLAB Syntax

```
x = str2num('str')
```

See Also

MATLAB `str2num` Calling Conventions

mlfStrcat

Purpose

String concatenation. Minimum number of input variables: two. Maximum: user-defined. Terminate all arguments lists with a NULL.

C Prototype

```
mxArray *mlfStrcat(mxArray *s1, ... );
```

C Syntax

```
include "matlab.h"

mxArray *s1;                /* Required input argument(s) */
mxArray *s2, *s3, *s4;      /* Optional input argument(s) */
mxArray *t;                 /* Return value */

t = mlfStrcat(s1,s2,NULL);
t = mlfStrcat(s1,s2,s3,s4,NULL);
```

MATLAB Syntax

```
t = strcat(s1,s2,s3,...)
```

See Also

MATLAB `strcat` Calling Conventions

Purpose

Compare strings

C Prototype

```
mxArray *mlfStrcmp(mxArray *str1, mxArray *str2);
```

C Syntax

```
#include "matlab.h"

mxArray *str1, *str2;       /* String array(s) */
mxArray *S, *T;             /* Input argument(s) */
mxArray *k, *TF;            /* Return value */

k = mlfStrcmp(str1,str2);
TF = mlfStrcmp(S,T);
```

MATLAB Syntax

```
k = strcmp('str1','str2')
TF = strcmp(S,T)
```

See Also

MATLAB `strcmp` Calling Conventions

mlfStrjust

Purpose

Justify a character array

C Prototype

```
mxArray *mlfStrjust(mxArray *S);
```

C Syntax

```
#include "matlab.h"

mxArray *S;                 /* Required input argument(s) */
mxArray *R;                 /* Return value */

R = mlfStrjust(S);
```

MATLAB Syntax

```
strjust(S)
```

See Also

MATLAB `strjust` Calling Conventions

Purpose

Compare the first n characters of two strings

C Prototype

```
mxArray *mlfStrncmp(mxArray *str1, mxArray *str2, mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *str1, *str2;       /* String array(s) */
mxArray *S, *T, *n;         /* Input argument(s) */
mxArray *k, *TF;            /* Return value */

k = mlfStrncmp(str1,str2,n);
TF = mlfStrncmp(S,T,n);
```

MATLAB Syntax

```
k = strncmp('str1','str2',n)
TF = strncmp(S,T,n)
```

See Also

MATlAB `strncmp` Calling Conventions

mlfStrrep

Purpose

String search and replace

C Prototype

```
mxArray *mlfStrrep(mxArray *str1, mxArray *str2, mxArray *str3);
```

C Syntax

```
#include "matlab.h"

mxArray *str1, *str2, *str3;    /* String array(s) */
mxArray *str;                   /* Return value */

str = mlfStrrep(str1,str2,str3);
```

MATLAB Syntax

```
str = strrep(str1,str2,str3)
```

See Also

MATLAB `strrep` Calling Conventions

Purpose

First token in string

C Prototype

```
mxArray *mlfStrtok(mxArray **rem, mxArray *str, mxArray *delimiter);
```

C Syntax

```
#include "matlab.h"

mxArray *str;           /* String array(s) */
mxArray *delimiter;     /* Optional input argument(s) */
mxArray *rem;           /* Optional output argument(s) */
mxArray *token;         /* Return value */

token = mlfStrtok(NULL,str,delimiter);
token = mlfStrtok(NULL,str,NULL);
token = mlfStrtok(&rem,str,NULL);
token = mlfStrtok(&rem,str,delimiter);
```

MATLAB Syntax

```
token = strtok('str',delimiter)
token = strtok('str')
[token,rem] = strtok(...)
```

See Also

MATLAB `strtok` Calling Conventions

mlfStrvcat

Purpose

Vertical concatenation of strings. Minimum number of input arguments: two, maximum: user-defined. Terminate the list of arguments with a NULL.

C Prototype

```
mxArray *mlfStrvcat(mxArray *t1, ... );
```

C Syntax

```
#include "matlab.h"

mxArray *t1;                    /* Required input argument(s) */
mxArray *t2, *t3, *t4;          /* Optional input argument(s) */
mxArray *S;                     /* Return value */

S = mlfStrvcat(t1,t2,NULL);         /* Concatenates two strings */
S = mlfStrvcat(t1,t2,t3,t4,NULL);   /* Concatenates four strings */
```

MATLAB Syntax

```
S = strvcat(t1,t2,t3,...)
```

See Also

MATLAB `strvcat` Calling Conventions

Purpose

Angle between two subspaces

C Prototype

```
mxArray *mlfSubspace(mxArray *A, mxArray *B);
```

C Syntax

```
#include "matlab.h"

mxArray *A, *B;         /* Required input argument(s) */
mxArray *theta;         /* Return value */

theta = mlfSubspace(A,B);
```

MATLAB Syntax

```
theta = subspace(A,B)
```

See Also

MATLAB subspace Calling Conventions

mlfSum

Purpose

Sum of array elements

C Prototype

```
mxArray *mlfSum(mxArray *A, mxArray *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *sum;           /* Optional output argument(s) */
mxArray *B;             /* Return value */

B = sum(A,NULL);
B = sum(A,dim);
```

MATLAB Syntax

```
B = sum(A)
B = sum(A,dim)
```

See Also

MATLAB `sum` Calling Conventions

Purpose

Singular value decomposition

C Prototype

```
mxArray *mlfSvd(mxArray **S, mxArray **V, mxArray *X,
                mxArray *Zero);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *S, *V;         /* Optional output argument(s) */
mxArray *Zero;          /* Optional input argument(s) */
mxArray *U, *s;         /* Return value */

Zero = mlfScalar(0);
s = mlfSvd(NULL,NULL,X,NULL);
U = mlfSvd(&S,&V,X,NULL);
U = mlfSvd(&S,&V,X,Zero);
```

MATLAB Syntax

```
s = svd(X)
[U,S,V] = svd(X)
[U,S,V] = svd(X,0)
```

See Also

MATLAB svd Calling Conventions

mlfTan, mlfTanh

Purpose

Tangent and hyperbolic tangent

C Prototype

```
mxArray *mlfTan(mxArray *X);
mxArray *mlfTanh(mxArray *X);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *Y;             /* Return value */

Y = mlfTan(X);
Y = mlfTanh(X);
```

MATLAB Syntax

```
Y = tan(X)
Y = tanh(X)
```

See Also

MATLAB `tan`, `tanh` Calling Conventions

Purpose

Stopwatch timer

C Prototype

```
void mxArray *mlfTic();
mxArray *mlfToc();
```

C Syntax

```
#include "matlab.h"

mxArray *t;                 /* Return value */

mlfTic();
    any statements
t = mlfToc();
```

MATLAB Syntax

```
tic
    any statements
toc
t = toc
```

See Also

MATLAB `tic`, `toc` Calling Conventions

mlfToeplitz

Purpose

Toeplitz matrix

C Prototype

```
mxArray *mlfToeplitz(mxArray *c, mxArray *r);
```

C Syntax

```
#include "matlab.h"

mxArray *r;             /* Required input argument(s) */
mxArray *c;             /* Optional input argument(s) */
mxArray *T;             /* Return value */

T = mlfToeplitz(c,r);
T = mlfToeplitz(r,NULL);
```

MATLAB Syntax

```
T = toeplitz(c,r)
T = toeplitz(r)
```

See Also

MATLAB `toeplitz` Calling Conventions

Purpose

Sum of diagonal elements

C Prototype

```
mxArray *mlfTrace(mxArray *a);
```

C Syntax

```
#include "matlab.h"

mxArray *A;             /* Required input argument(s) */
mxArray *b;             /* Return value */

b = mlfTrace(A);
```

MATLAB Syntax

```
b = trace(A)
```

See Also

MATLAB `trace` Calling Conventions

mlfTrapz

Purpose

Trapezoidal numerical integration

C Prototype

```
mxArray *mlfTrapz(mxArray *Y, mxArray *X, mxArray *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *Y;             /* Required input argument(s) */
mxArray *X;             /* Optional input argument(s) */
mxArray *Z;             /* Return value */

Z = mlfTrapz(Y,NULL,NULL);
Z = mlfTrapz(X,Y,NULL);
Z = mlfTrapz(Y,dim,NULL);
Z = mlfTrapz(X,Y,dim);
```

MATLAB Syntax

```
Z = trapz(Y)
Z = trapz(X,Y)
Z = trapz(...,dim)
```

See Also

MATLAB `trapz` Calling Conventions

Purpose

Lower triangular part of a matrix

C Prototype

```
mxArray *mlfTril(mxArray *X, mxArray *k);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *k;             /* Optional input argument(s) */
mxArray *L;             /* Return value */

L = mlfTril(X,NULL);
L = mlfTril(X,k);
```

MATLAB Syntax

```
L = tril(X)
L = tril(X,k)
```

See Also

MATLAB `tril` Calling Conventions

mlfTriu

Purpose

Upper triangular part of a matrix

C Prototype

```
mxArray *mlfTriu(mxArray *X, mxArray *k);
```

C Syntax

```
#include "matlab.h"

mxArray *X;             /* Required input argument(s) */
mxArray *k;             /* Optional input argument(s) */
mxArray *U;             /* Return value */

U = mlfTriu(X,NULL);
U = mlfTriu(X,k);
```

MATLAB Syntax

```
U = triu(X)
U = triu(X,k)
```

See Also

MATLAB `triu` Calling Conventions

Purpose

Set union of two vectors

C Prototype

```
mxArray *mlfUnion(mxArray **ia, mxArray **ib, mxArray *a,
                  mxArray *b, mxArray *rows_str);
```

C Syntax

```
#include "matlab.h"

mxArray *a, *b, *A, *B;     /* Input argument(s) */
mxArray *ia, *ib;           /* Optional output argument(s) */
mxArray *rows_str;          /* String array(s) */
mxArray *c;                 /* Return value */

c = mlfUnion(NULL,NULL,a,b,NULL);

rows_str = mxCreateString("rows");
c = mlfUnion(NULL,NULL,A,B,rows_str);

c = mlfUnion(&ia,&ib,a,b,NULL);
c = mlfUnion(&ia,&ib,A,B,rows_str);
```

MATLAB Syntax

```
c = union(a,b)
c = union(A,B,'rows')
[c,ia,ib] = union(...)
```

See Also

MATLAB union Calling Conventions

mlfUnique

Purpose

Unique elements of a vector

C Prototype

```
mxArray *mlfUnique(mxArray **i, mxArray **j, mxArray *a,
                   mxArray *rows_str);
```

C Syntax

```
#include "matlab.h"

mxArray *a, *A;             /* Input argument(s) */
mxArray *i, *j;             /* Optional output argument(s) */
mxArray *rows_str;          /* String array(s) */
mxArray *b;                 /* Return value */

b = mlfUnique(NULL,NULL,a,NULL);

rows_str = mxCreateString("rows");
b = mlfUnique(NULL,NULL,A,rows_str);

b = mlfUnique(&i,&j,a,NULL);
b = mlfUnique(&i,&j,A,rows_str);
```

MATLAB Syntax

```
b = unique(a)
b = unique(A,'rows')
[b,index] = unique(...)
```

See Also

MATLAB `unique` Calling Conventions

Purpose

Correct phase angles

C Prototype

```
mxArray *mlfUnwrap(mxArray *P, mxArray *tol, mxArray *dim);
```

C Syntax

```
#include "matlab.h"

mxArray *P;                     /* Required input argument(s) */
mxArray *null_matrix, *Zero;    /* Optional input argument(s) */
mxArray *tol, *dim;             /* Optional input argument(s) */
mxArray *Q;                     /* Return value */

Q = mlfUnwrap(P,NULL,NULL);
Q = mlfUnwrap(P,tol,NULL);

Zero = mlfScalar(0);
null_matrix = mlfZeros(Zero,Zero);
Q = mlfUnwrap(P,null_matrix,dim);

Q = mlfUnwrap(P,tol,dim);
```

MATLAB Syntax

```
Q = unwrap(P)
Q = unwrap(P,tol)
Q = unwrap(P,[],dim)
Q = unwrap(P,tol,dim)
```

See Also

MATLAB unwrap Calling Conventions

mlfUpper

Purpose

Convert string to upper case

C Prototype

```
mxArray *mlfUpper(mxArray *str);
```

C Syntax

```
#include "matlab.h"

mxArray *str;           /* String array(s) */
mxArray *t;             /* Return value */

t  = mlfUpper(str);
```

MATLAB Syntax

```
t  = upper('str')
```

See Also

MATLAB `upper` Calling Conventions

Purpose

Test matrix (Vandermonde matrix)

C Prototype

```
mxArray *mlfVander(mxArray *c);
```

C Syntax

```
#include "matlab.h"

mxArray c;                  /* Required input argument(s) */
mxArray A;                  /* Return value */

A = mlfVander(c);
```

MATLAB Syntax

```
[A,B,C,...] = gallery('tmfun',P1,P2,...)
gallery(3)              a badly conditioned 3-by-3 matrix
gallery(5)              an interesting eigenvalue problem
```

Description

`A = mlfVander(c);` returns the Vandermonde matrix whose second to last column is `c`. In MATLAB, the `j`th column of a Vandermonde matrix is given by `A(:,j) = C^(n-j)`.

See Also

MATLAB `gallery` Calling Conventions

mlfVertcat

Purpose

Vertical concatenation. Minimum number of input arguments: two, maximum: user-defined. Terminate the list of arguments with a NULL.

C Prototype

```
mxArray *mlfVertcat(mxArray *A, ...);
```

C Syntax

```
#include "matlab.h"

mxArray *A, *B;             /* Required input argument(s) */
mxArray *C;                 /* Optional output argument(s) */
mxArray *R;                 /* Return value */

R = mlfVertcat(*A,*B,NULL);
R = mlfVertcat(*A,*B,*C,NULL);
```

MATLAB Syntax

```
[A;B;C...]
vertcat(A,B,C...)
```

See Also

MATLAB `vertcat` Calling Conventions

Purpose

Display warning message

C Prototype

```
mxArray *mlfWarning(mxArray **f, mxArray *message);
```

C Syntax

```
#include "matlab.h"

mxArray *message, *on_str, *off_str;/* String array(s) */
mxArray *backtrace_str, debug_str;  /* String array(s) */
mxArray *once_str, *always_str;     /* String array(s) */
mxArray *f;                         /* Optional output argument(s)
   */
mxArray *s;                         /* Return value */

message = mxCreateString("I'm sorry Dave");
on_str = mxCreateString("on");
off_str = mxCreateString("off");
backtrace_str = mxCreateString("backtrace");
debug_str = mxCreateString("debug");
once_str = mxCreateString("once");
always_str = mxCreateString("always");

s = mlfWarning(NULL,message);
s = mlfWarning(NULL,on_str);
s = mlfWarning(NULL,off_str);
s = mlfWarning(NULL,backtrace_str);
s = mlfWarning(NULL,debug_str);
s = mlfWarning(NULL,once_str);
s = mlfWarning(NULL,always_str);
s = mlfWarning(&f,NULL);
```

mlfWarning

MATLAB Syntax

```
warning('message')
warning on
warning off
warning backtrace
warning debug
warning once
warning always
[s,f] = warning
```

See Also MATLAB `warning` Calling Conventions

Purpose

Day of the week

C Prototype

```
mxArray *mlfWeekday(mxArray **S, mxArray *D);
```

C Syntax

```
#include "matlab.h"

mxArray *D;                 /* Required input argument(s) */
mxArray *S;                 /* Required output argument(s) */
mxArray *N;                 /* Return value */

N = mlfWeekday(&S,D);
```

MATLAB Syntax

```
[N,S] = weekday(D)
```

See Also

MATLAB `weekday` Calling Conventions

mlfWilkinson

Purpose

Wilkinson's eigenvalue test matrix

C Prototype

```
mxArray *mlfWilkinson(mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *n;             /* Required input argument(s) */
mxArray *W;             /* Return value */

W = mlfWilkinson(n);
```

MATLAB Syntax

```
W = wilkinson(n)
```

See Also

MATLAB `wilkinson` Calling Conventions

Purpose

Exclusive or

C Prototype

```
mxArray *mlfXor(mxArray *A, mxArray *B);
```

C Syntax

```
#include "matlab.h"

mxArray *A, *B;         /* Required input argument(s) */
mxArray *C;             /* Return value */

C = mlfXor(A,B);
```

MATLAB Syntax

```
C = xor(A,B)
```

See Also

MATLAB xor Calling Conventions

mlfZeros

Purpose

Create an array of all zeros

C Prototype

```
mxArray *mlfZeros(mxArray *m, mxArray *n);
```

C Syntax

```
#include "matlab.h"

mxArray *x;                     /* Dimension vector */
mxArray *m, *n;                 /* Input argument(s) */
mxArray *S, *A;                 /* Input argument(s) */
mxArray *B;                     /* Return value */

B = mlfZeros(n,NULL);
B = mlfZeros(m,n);

x = mlfHorzcat(m,n,NULL);
B = mlfZeros(x,NULL);

S = mlfSize(NULL,A,NULL);
B = mlfZeros(S,NULL);
```

MATLAB Syntax

```
B = zeros(n)
B = zeros(m,n)
B = zeros([m n])
B = zeros(size(A))
```

See Also

MATLAB `zeros` Calling Conventions

Purpose

This function handles assignments that include indexing

C Prototype

```
void mlfArrayAssign(mxArray *destination, mxArray *source, ... );
```

Arguments

`mxArray *destination`
Specifies the destination array that will be modified.

`mxArray *source`
Specifies the source array that contains the new values for the destination array.

`optional mxArray* arguments`
Specify one or two indices that form the subscript for the *destination* array.

Terminate the argument list by passing `NULL` as the last argument.

Return

This function returns `void`. The result of the assignment is stored in the argument `destination`.

Description

Use the function `mlfArrayAssign()` to make assignments that involve indexing. The arguments to `mlfArrayAssign()` consist of a destination array, a source array, and one or two index arrays that represent the subscript. The subscript specifies the elements that are to be modified in the destination array; the source array specifies the new values for those elements. The subscript is only applied to the destination array.

The functions are defined to accept a variable number of indices. Supply one index `mxArray` argument to perform one-dimensional indexing. Supply two index `mxArray` arguments to perform two-dimensional indexing.

Example

```
mxArray *fortyfive = mlfScalar(45);
mxArray *three = mlfScalar(3);
mxArray *one = mlfScalar(1);
mlfArrayAssign(A, fortyfive, three, one, NULL);
```

writes the value 45 into the element at row three, column one of array `A`. If you assign a value to a location that does not exist in the array, the array grows to include that element.

See Also

`mlfArrayDelete`, `mlfArrayRef`, `mlfColon`, `mlfCreateColonIndex`, `mlfEnd`

mlfArrayDelete

Purpose

Delete elements from an array

C Prototype

```
void mlfArrayDelete(mxArray *destination, mxArray *index1, ... );
```

Arguments

`mxArray *destination`
Specifies the array that you want to delete elements from.

`mxArray *index1`
Specifies an index that is used to form the subscript.

`optional mxArray* arguments`
Additional index arguments that are used to form the subscript.

Terminate the argument list by passing `NULL` as the last argument.

Return

This function returns `void`. The result of the deletion is stored in the argument `destination`.

Description

Use the function `mlfArrayDelete()` to delete elements from an array. This function is equivalent to the MATLAB statement, `A(B) = []`. Instead of specifying a subscript for the elements you want to replace with other values, specify a subscript for the elements you want removed from the array. The MATLAB C Math Library removes those elements and shrinks the array.

When you delete a single element from a matrix, the matrix is converted into a row vector that contains one fewer element than the original matrix. You can also delete more than one element from a matrix, shrinking the matrix by that number of elements. To retain the rectangularity of the matrix, however, you must delete one or more entire rows or columns.

Example

```
mlfArrayDelete(A, three, one, NULL);
```

This function removes the element at row three, column one from array `A`. Note that removing an element from a matrix reshapes the matrix into a vector.

See Also

`mlfArrayAssign`, `mlfArrayRef`, `mlfColon`, `mlfCreateColonIndex`, `mlfEnd`

Purpose

This function handles array references

C Prototype

```
mxArray *mlfArrayRef(mxArray *array, ... );
```

Arguments

`mxArray *array`
Specifies the target array.

`optional mxArray* arguments`
Specify the indices that form the subscript. Pass one index for one-dimensional indexing. Pass two indices for two-dimensional indexing.

Terminate the argument list by passing `NULL` as the last argument.

Return

This function returns a pointer to a newly allocated `mxArray` that contains the result of the indexing operation.

Description

`mlfArrayRef()` extracts the elements specified by the subscript from the target array and returns the result in a new `mxArray`. `mlfArrayRef()` is the only indexing function to return a value.

Example

```
mxArray *two = mlfScalar(2), B;
B = mlfArrayRef(A, two, two, NULL);
```

This statement selects the element at row 2, column 2 in array `A` and returns it in `B`.

See Also

`mlfArrayAssign`, `mlfArrayDelete`, `mlfColon`, `mlfCreateColonIndex`, `mlfEnd`

mlfColon

Purpose

Create vectors and use in array subscripting

C Prototype

```
mxArray *mlfColon(mxArray *start, mxArray *step, mxArray *end);
```

Arguments

`mxArray *start`
Initial value

`mxArray *step`
Increment value

`mxArray *end`
Final value

Description

This function lets you specify a vector index.

Example

```
mxArray *vector_index, *one, *ten;
one = mlfScalar(1);
ten = mlfScalar(10);
vector_index = mlfColon(one, ten, NULL);
```

This function specifies the vector `[ 1 2 3 4 5 6 7 8 9 10 ]`.

```
mxArray *colon;
colon = mlfColon(NULL,NULL,NULL);
```

This call is equivalent to a call to `mlfCreateColonIndex()`.

See Also

`mlfArrayAssign`, `mlfArrayDelete`, `mlfArrayRef`, `mlfCreateColonIndex`, `mlfEnd`

Purpose

Create and initialize a complex 1-by-1 array

C Prototype

```
mxArray *mlfComplexScalar(double v, double i);
```

Arguments

`double v`
Initial content of the real part of the array

`double i`
Initial content of the imaginary part of the array

Description

This function creates a complex 1-by-1 array whose contents are initialized to the real part, `v`, and the imaginary part, `i`.

See Also

`mlfScalar`

mlfCreateColonIndex

Purpose

Create an array that acts like the colon operator when passed as an index to an indexing function

C Prototype

```
mxArray *mlfCreateColonIndex(void);
```

Description

The `mlfCreateColonIndex()` index, which loosely interpreted means "all," selects, for example, all the columns in a row or all the rows in a column.

Example

```
mxArray *colon;
colon = mlfCreateColonIndex();
B = mlfArrayRef(A, one, colon, NULL);
```

The call to `mlfArrayRef()` selects all the elements in the first row of array `A` and assigns them to array `B`.

See Also

`mlfArrayAssign`, `mlfArrayDelete`, `mlfArrayRef`, `mlfColon`, `mlfEnd`

Purpose

Generate the last index for an array dimension

C Prototype

```
mxArray *mlfEnd(mxArray *array, mxArray *dim, mxArray *numindices);
```

Arguments

`mxArray *array`
Specifies the target array.

`mxArray *dim`
Dimension where `mlfEnd()` is used

`mxArray *numindices`
Number of indices in the subscript

Description

The `mlfEnd()` function, which corresponds to the MATLAB `end()` function, provides another way of specifying a vector index. Given an array, a dimension (1 = row , 2 = column), and the number of indices in the subscript, `mlfEnd()` returns the index of the last element in the specified dimension. You then use that scalar array to generate a vector index to be used in one or two-dimensional indexing..

Given the row dimension, `mlfEnd()` returns the number of columns. Given the column dimension, it returns the number of rows. For a matrix and a one-dimensional index, `mlfEnd()` treats the matrix like a vector and returns the number of elements in the matrix.The number of indices in the subscript corresponds to the number of index arguments you pass to `mlfArrayRef()`.

Example

```
mxArray *end, *index, *two, *B;
two = mlfScalar(2);
end = mlfEnd(A, two, two);
index = mlfColon(two, end, NULL);
B = mlfArrayRef(A, three, index, NULL);
```

This code selects all but the first element in row three of array `A`. The first `two` argument to `mlfEnd()` identifies the dimension where `mlfEnd()` is used, here the column dimension. The second `two` argument indicates the number of indices in the subscript; for two-dimensional indexing, it is always two.

See Also

`mlfArrayAssign`, `mlfArrayDelete`, `mlfArrayRef`, `mlfColon`, `mlfCreateColonIndex`

mlfFevalTableSetup

Purpose

Registers a thunk function table with the MATLAB C Math Library.

C Prototype

```
void mlfFevalTableSetup ( mlfFuncTab *mlfUfuncTable );
```

Arguments

`mlfFuncTab *mlfUfuncTable`
Pointer to a local `feval` table. Each entry is composed of a string representing a function name, a pointer to that function, and a pointer to a thunk function that knows how to execute the function.

Description

A call to `mlfFevalTableSetup()` adds the entries in a local table to the MATLAB C Math Library built-in `feval` function table. `mlfFeval()` accesses the library's built-in function table to locate the function pointers that are associated with a given function name.

Purpose Format output similar to `printf`

C Prototype

```
int mlfPrintf(const char *fmt, ...);
```

Arguments

`const char *fmt`
String to print. String may include `printf`-style format characters that specify the format for subsequent strings.

Description Uses the installed print handler to display the output.

See Also `mlfPrintMatrix`, `mlfSetPrintHandler`

mlfPrintMatrix

Purpose Print the contents of an array

C Prototype
```
void mlfPrintMatrix(mxArray *m);
```

Arguments
`mxArray *m`
Array to print

Description `mlfPrintMatrix()` calls the installed print handler.

See Also `mlfPrintf`, `mlfSetPrintHandler`

Purpose Create and initialize a 1-by-1 array

C Prototype

```
mxArray *mlfScalar(double v);
```

Arguments

`double v`
Initial contents of the array

Description This function creates a 1-by-1 array whose contents are initialized to the value of v.

Example

```
mxArray *one = mlfScalar(1);
```

See Also `mlfComplexScalar`

mlfSetErrorHandler

Purpose

Register an error handler function with the MATLAB C Math Library

C Prototype

```
void mlfSetErrorHandler(void(* EH)(const char*, bool));
```

Arguments

`void(* EH)(const char*, bool)`
A pointer to a function that takes a `char *` argument and a boolean argument that indicates whether the first argument is an error message or a warning. The MATLAB C Math Library calls this function rather than its default error handler when an error or warning must be displayed .

Description

This function lets you to control how errors are displayed and handled.

Purpose

Set the MATLAB C Math Library's memory management functions

C Prototype

```
void mlfSetLibraryAllocFcns(calloc_proc calloc_fcn,
  free_proc free_fcn, realloc_proc realloc_fcn, malloc_proc
  malloc_fcn);
```

Arguments

`calloc_proc calloc_fcn`
The function that `mxCalloc` uses to perform memory allocation operations

`free_proc free_fcn`
The function that `mxFree` uses to perform memory deallocation (freeing) operations

`realloc_proc realloc_fcn`
The function that `mxRealloc` uses to perform memory reallocation operations

`malloc_proc malloc_fcn`
The function to be called in place of `malloc` to perform memory allocation operations

Definition

This function lets you register your own allocation and deallocation routines with the MATLAB C Math Library. It gives you complete control over memory management.

mlfSetPrintHandler

Purpose

Register a print handler with the MATLAB C Math Library

C Prototype

```
void mlfSetPrintHandler(void(* PH)(const char *));
```

Arguments

`void(* PH)(const char *)`
Pointer to a function that takes a single argument, a `const char *` (the message to be displayed), and returns `void`. This function displays the character string.

Description

Instead of calling `printf` directly, the MATLAB C Math Library calls a print handler when it needs to display an error message or warning. The default print handler used by the library takes a single argument, a `const char *` (the message to be displayed), and returns `void`.

To register your function and change which print handler the library uses, you must call the routine `mlfSetPrintHandler`. If you use an alternate print handler, you must call `mlfSetPrintHandler` before calling other library routines.

See Also

`mlfSetErrorHandler`

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Z