MATLAB® C++ Math Library

The Language of Technical Computing

Computation

Visualization

Programming

The MATH WORKS

Reference

Version 2

How to Contact The MathWorks:

`www.mathworks.com`	Web
`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
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

508-647-7000 Phone

508-647-7001 Fax

The MathWorks, Inc. Mail
3 Apple Hill Drive
Natick, MA 01760-2098

For contact information about worldwide offices, see the MathWorks Web site.

MATLAB C++ Math Library Reference

Printing History:

January 1998	Version 1.2
January 1999	Revised for Version 2.0 (Release 11) Online only
September 1999	Revised for Version 2.1 (Release 12) Online only
June 2001	Revised for Version 2.2 (Release 12.1) Online only

Contents

Reference Page Format

This reference gives you quick access to the prototypes and call syntax for the MATLAB C++ Math Library functions. At the bottom of each page, you'll find a link to the documentation for the MATLAB version of the function. Use the MATLAB function 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++ Prototypes
- C++ Syntax
- MATLAB Syntax
- See Also links to the documentation of the MATLAB version of the function and to the calling conventions

Overloaded versions of the C++ functions or prototypes with defaulted input arguments represent the MATLAB syntax. You'll find a different prototype for each way of calling the MATLAB function.

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 prototypes for a function.
- The first C++ prototype listed should correspond to the first C++ call to a function listed under "C++ Syntax" and to the first call listed under "MATLAB Syntax." The second C++ prototype corresponds to the second C++ call and the second MATLAB call, and so forth.

Note 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 support objects. Because this version of the MATLAB C++ Math Library does not support objects, the corresponding MATLAB

function documentation regarding objects 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.

Notes on the Format

- Assignments to input arguments are not shown.
- Occasionally, a string, for example, `"nobalance"`, or an integer is passed as an argument if that string or integer is the only valid value for the argument.
- Occasionally, a call to `horzcat()` initializes a vector.
- The number of C++ prototypes may not match the number of documented calls to the MATLAB function. This mismatch occurs if one prototype supports two ways of calling the MATLAB function or if an additional prototype has been added to support the omission of input or output arguments supported by MATLAB.

Calling Conventions

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 "How to Call C++ Library Functions" section of Chapter 5 in the *MATLAB C++ Math Library User's Guide* for further discussion of the MATLAB C++ Math Library calling conventions and for a list of exceptions to the calling conventions. Also, see the `mwVarargin` and `mwVarargout` reference pages for information on how to call functions that can take any number of input or output arguments.

Constructing a C++ Prototype

A complete set of C++ prototypes appears on the reference page for each function. You can reconstruct the C++ prototypes for a library function by examining the MATLAB syntax for a function. In C++ an overloaded version of the function exists for each different way of calling the MATLAB function.

For example, the MATLAB function `svd()` has the following syntax.

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

To construct the C++ prototypes, follow this procedure.

1. Use the first return value, `U`, as the return value from the function. C++ routines can only return a single value. The data type for the return value is `mwArray`.
2. Add the remaining return values, `S` and `V`, as the first, second, etc., output arguments to the function. The data type for an output argument is `mwArray *`.
3. Add the number of input arguments to the prototype, `X` and `Zero`, one after another following the output arguments. The data type for an input argument is `mwArray`.

Here is the complete C++ prototype for the `svd` routine.

```
mwArray svd(mwArray *S, mwArray *V, const mwArray &X,
                const mwArray &Zero);
```

Note Contrast the data type for an output argument with the data type for an input argument. The type for an output argument is a pointer to an `mwArray`. The type for an input argument is an `mwArray`. The `const mwArray &` in the prototype improves the efficiency of the function, but you can ignore the `const` and `&` and just pass in an `mwArray`.)

Translating MATLAB Syntax into C++ Syntax

This procedure translates the MATLAB call `[U,S,V] = svd(X,0)` into a C++ call. 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.

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

   ```
   U =
   ```

3. Pass any other output arguments as the first arguments to the function.

   ```
   U = svd(&S,&V,
   ```

4. Pass the input arguments to the C++ function, following the output arguments.

   ```
   U = svd(&S,&V,X,0);
   ```

The translation is complete.

Note that if you see `[]` as a MATLAB input argument, you should pass `mwArray()` as the C++ argument. For example,

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

becomes

```
B = cplxpair(A,mwArray(),dim);
```

The mwArray Class

The mwArray class public interface consists of:

- Constructors and destructor
- Overloaded operators
- One user-defined conversion routine
- Memory management new and delete routines
- Array size query routines

Note The mwArray's public interface does not contain any mathematical operators or functions.

See "Extracting Data from an mwArray" in Chapter 10 of the *MATLAB C++ Math Library User's Guide* for documentation of the member functions GetData(), SetData(), ExtractScalar(), ExtractData(), and ToString().

Constructors

The mwArray interface provides many useful constructors. You can construct an mwArray object from the following types of data: a numerical scalar, an array of scalars, a string, an mxArray *, or another mwArray object.

`mwArray()`
Create an uninitialized array. An uninitialized array produces warnings when passed to MATLAB C++ Math Library functions. If an array is created using this default constructor, a value must be assigned to it before passing it to a MATLAB C++ Math Library function.

To create an empty double matrix that corresponds to [] in MATLAB, use the function empty().

`mwArray(const char *str)`
Create an array from a string. The constructor copies the string.

`mwArray(int32 rows, int32 cols, double *real, double *imag = 0)`:
Create an `mwArray` from either one or two arrays of double-precision floating-point numbers. If two arrays are specified, the constructor creates a complex array; both input arrays must be the same size. The data in the input arrays must be in column-major order, the reverse of C++'s usual row-major order. This constructor copies the input arrays.

Note that the last argument, `imag`, is assigned a value of zero in the constructor. `imag` is an optional argument. When you call this constructor, you do not need to specify the optional argument. Refer to a C++ reference guide for a more complete explanation of default arguments.

`mwArray(const mwArray &mtrx)`
Copy an `mwArray`. This constructor is the familiar C++ copy constructor, which copies the input array. For efficiency, this routine does not actually copy the data until the data is modified. The data is referenced through a pointer until a modification occurs.

`mwArray(const mxArray *mtrx)`
Make an `mwArray` from an `mxArray *`, such as might be returned by any of the routines in the MATLAB C Math Library or the Application Program Interface Library. This routine does *not* copy its input array, yet the destructor frees it; therefore the input array must be allocated on the heap. In most cases, for example, with matrices returned from the Application Program Interface Library, this is the desired behavior.

`mwArray(double start, double step, double stop)`
Create a ramp. This constructor operates just like the MATLAB colon operator. For example, the call `mwArray(1, 0.5, 3)` creates the vector `[ 1, 1.5, 2, 2.5, 3 ]`.

`mwArray(int32 start, int32 step, int32 stop)`
Create an integer ramp.

`mwArray(const mwSubArray & a)`
Create an `mwArray` from an `mwSubArray`. When an indexing operation is applied to an array, the result is not another array, but an `mwSubArray` object. An `mwSubArray` object remembers the indexing operation. Evaluation of the operation is deferred until the result is assigned or used in another expression.

This constructor evaluates the indexing operation encoded by the mwSubArray object and creates the appropriate array.

mwArray(double)
Create a 1-by-1 mwArray from a double-precision floating-point number.

mwArray(int)
Create an mwArray from an integer.

Operators

The mwArray class support three types of operators

- Indexing operators
- Stream I/O operators
- Assignment operators

Array Indexing Operators

Indexing is implemented through the complex interaction of three classes: mwArray, mwSubArray, and mwIndex. The indexing operator is (), and its usual argument is an mwIndex, which can be made from a scalar or another array. When applied to an mwArray, operator() returns an mwSubArray. The mwSubArray "remembers" the indexing operation and defers evaluation until the result is either assigned or referred to.

The MATLAB C++ Math Library supports one- and two-dimensional indexing.

mwSubArray operator()(const mwIndex &a) const
This routine implements one-dimensional indexing with an mwIndex object providing the subscript.

mwSubArray operator()(const mwIndex &a)
This routine modifies the contents of an array using one-dimensional indexing. Because this routine is non-const, calls to it are valid targets for the assignment operator.

mwSubArray operator()(const mwIndex &a, const mwIndex &b) const
This is the most general form of two-dimensional indexing. Because mwIndex objects can be made from integers, double-precision floating-point numbers

and even mwArrays, this routine can handle two-dimensional indexing of any type.

`mwSubArray operator()(const mwIndex &a, const mwIndex &b)`
Like its one-dimensional counterpart, this routine allows two-dimensional indexing expressions as the target of assignment statements.

Cell Content Indexing. These two versions of the `cell()` member function let you index into the contents of a cell. For example, `A.cell(1,2)` refers to the contents of the cell in the second column of the first row in an array `A`.

The `cell()` member functions follow the library convention for `varargin` functions. You can pass up to 32 arguments to the functions. To index into more than 32 dimensions, you must construct an `mwVarargin` object and pass it as the first argument. That object allows you to reference an additional 32 arguments, the first of which can again be an `mwVarargin` object.

The second non-`const` signature supports calls that are targets of the assignment operator and modify the contents of a cell.

```
mwArray cell(const mwVarargin &RI1,
             const mwArray &OI2=mwArray::DIN,
             const mwArray &OI3=mwArray::DIN,
             .
             .
             .
             const mwArray &OI32=mwArray::DIN ) const;

mwSubArray cell(const mwVarargin &RI1,
             const mwArray &OI2=mwArray::DIN,
             const mwArray &OI3=mwArray::DIN,
             .
             .
             .
             const mwArray &OI32=mwArray::DIN );
```

Structure Field Indexing. The two versions of the `field()` member function let you reference the field of a structure. For example, `A.field("name")` accesses the contents of the field called `name` within the structure `A`.

The second non-`const` signature supports calls that are targets of the assignment operator and modify the contents of a field.

```
mwArray field(const char *fieldname) const;
mwSubArray field(const char *fieldname);
```

Stream I/O Operators

The two operators, << and >>, are used for stream input and output. Technically, these stream operators are not member functions; they are friend functions.

`friend inline ostream& operator<<(ostream &os, const mwArray&)`
Calling this operator inserts an `mwArray` object into the given stream. If the stream is `cout`, the contents of the `mwArray` object appear on the terminal screen or elsewhere if standard output has been redirected on the command line. This function simply invokes `Write()` as described below.

`friend inline istream& operator>>(istream &is, mwArray&)`
This is the stream extraction operator, capable of extracting, or reading, an `mwArray` from a stream. The stream can be any C++ stream object, for example, standard input, a file, or a string. This function simply invokes `Read()` as described below.

The stream operators call `Read()` and `Write()`, `mwArray` public member functions.

`void Read(istream&)`
Reads an `mwArray` from an input stream. An array definition consists of an optional scale factor and asterisk, `*`, followed by a bracket `[`, one or more semicolon-separated rows of double-precision floating-point numbers, and a closing bracket `]`.

`void Write(ostream&, int32 precision =5, int32 line_width =75) const`
Formats `mwArray` objects using the given precision (number of digits) and line width, and then writes the objects into the given stream. `operator<<()` uses the default values shown above, which are appropriate for 80-character-wide terminals.

Note `Write()` writes arrays in exactly the format that `Read()` reads them. An array written by `Write()` can be read by `Read()`.

Assignment Operators

```
mwArray &operator=(const mwArray&);
```

The final operator, =, is the assignment operator. C++ requires that the assignment operator be a member function. Like the copy constructor, the assignment operator does not actually make a copy of the input array, but rather references (keeps a pointer to) the input array's data; this is an optimization made purely for efficiency, and has no effect on the semantics of assignment. If you write `A = B` and then modify `B`, the values in `A` will remain unchanged.

User-Defined Conversion

There is only one user-defined conversion: from an `mwArray` to a double-precision floating-point number. This conversion function only works if the `mwArray` is scalar (1-by-1) and noncomplex.

```
operator double() const;
```

Memory Management

Overloading the operators `new` and `delete` provides the necessary hooks for user-defined memory management. The MATLAB C++ Math Library has its own memory management scheme.

If this scheme is inappropriate for your application, you can modify it. However, you should not do so by overloading `new` and `delete`, because the `mwArray` class already contains overloaded versions of these operators.

```
void *operator new(size_t size)
```

```
void operator delete(void *ptr, size_t size)
```

Array Size

In MATLAB, the `size()` function returns the size of an array as an array. The MATLAB C++ Math Library provides a corresponding version of `size()` that also returns an array. Because this C++ version allocates an array to hold just two integers, it is not efficient. The `mwArray Size` member functions below return the size of an array more efficiently.

An array (a matrix is a special case) has two sizes: the number of its dimensions (for matrices, always two) and the actual size of each dimension. You can use these `Size()` functions to determine both the number of dimensions and the size of each dimension.

`int32 Size() const`
Return the number of dimensions. In this version of the library, this function always returns two.

`int32 Size(int32 dim) const`
Return the size (number of elements) of the indicated dimension. The integer argument to this function must be either 1 or 2.

`int32 Size(int32* dims, int maxdims=2) const`
Determine the sizes of all the dimensions of the array and return them via the given integer array, `dims`. `maxdims` is the maximum number of dimensions the function should return. The input integer array `dims` must contain enough space to store at least `maxdims` integers. If `maxdims` is less than the number of dimensions of the `mwArray`, the last dimension returned is the product of the remaining dimensions. This function's return value is the number of dimensions of the `mwArray`.

The mwVarargin and mwVarargout Classes

MATLAB supports functions that accept a variable number of input arguments and return a variable number of return values. The C++ Math Library defines two classes to handle these functions.

- The mwVarargin class
- The mwVarargout class

The mwVarargin Class

MATLAB C++ Math Library functions that take a variable number of input arguments have one `mwVarargin` argument followed by 31 additional `mwArray` arguments.

- If you pass 32 or fewer arguments, you can ignore the `mwVarargin` parameter and simply pass a series of `mwArrays` as with any other function.
- If you need to pass more than 32 inputs, you must construct an `mwVarargin` object and pass it as the `mwVarargin` parameter.

Constructors

The `mwVarargin` constructor has the standard `varargin` parameter list: one `mwVarargin` argument followed by 31 additional `mwArray` arguments. The `mwVarargin` constructors can be nested enabling you to pass an unlimited number of inputs.

The inputs used to construct the `mwVarargin` argument appear first on the argument list for the function, followed by the remaining 31 inputs. It is not necessary to fill out the `mwVarargin` constructor parameter list. The arguments can be distributed between the `mwVarargin` constructor and the remaining 31 arguments.

For example, the library function `horzcat()` is a `varargin` function that demonstrates the standard `varargin` parameter list. Its function prototype is

```
mwArray horzcat(const mwVarargin &in1=mwArray::DIN,
                const mwArray &in2=mwArray::DIN,
                .
                .
                .
```

```
                    const mwArray &in32=mwArray::DIN);
```

To pass 90 inputs to the `horzcat` function, make this call:

```
horzcat(mwVarargin(mwVarargin(p1,p2,...,p32), p33, ..., p63),
        p64, ..., p90);
```

The first 32 arguments are passed to an `mwVarargin` constructor that is nested as the first argument to another `mwVarargin` constructor. The next 31 arguments (`p33` through `p63`) are passed as `mwArray` arguments to the `mwVarargin` object that is the first argument to `horzcat()`. The remaining arguments (`p64` through `p90`) are passed as additional `mwArray` arguments to the function.

Note that the ... represent omitted aguments in the series and are not part of the actual function prototype or function call.

Note If a function takes any required output arguments, an `mwVarargout` argument, or any required or optional input arguments, these arguments precede the first `mwVarargin` argument in the list of arguments.

The mwVarargout Class

MATLAB C++ Math Library functions that produce a variable number of outputs have an `mwVarargout` parameter as their last output argument.

To retrieve the `varargout` outputs from the function, you need to construct an `mwVarargout` object. You pass the variables to which the outputs will be assigned to the `mwVarargout` constructor and then pass the `mwVarargout` object as the last output argument to the function.

The arguments to the `mwVarargout` constructor differ from normal output arguments in two ways. When constructing an `mwVarargout` object:

- You pass the array itself, not a pointer to the array, to the constructor.
- You can pass indexed expressions as inputs. Anything that can appear on the left hand side of an assignment can appear as an argument to the `mwVarargout` constructor.

For example, this code demonstrates a call to the M-function `size`, which takes a variable number of output arguments and a single input argument. The prototype for `size()` in C++ specifies an `mwVarargout` object, as its first parameter, and one or two input arguments. The call to `size()` in C++ corresponds to the call in M.

M code:

```
[x, y(2,3), z{:}] = size(m)
```

C++ prototype:

```
mwArray size(mwVarargout varargout,
   const mwArray &in1,
   const mwArray &in2=mwArray::DIN);
```

C++ call:

```
size(mwVarargout(x, y(2,3), z.cell(colon())), m);
```

Note that the function `size()` takes no other required output arguments besides a `varargout` argument. It is called a "pure" `varargout` function. In pure `varargout` functions, the return value of the function is the same as the value assigned to the first element of the `mwVarargout` object, in this case the variable `x`. When calling a pure `varargout` function, you do not need to assign the output of the function to the first output argument explicitly; simply pass it to the `mwVarargout` constructor. For all functions in the math library, if the first argument is `mwVarargout`, the function is pure `varargout`.

If other output arguments precede the `mwVarargout` parameter, then the return value is not part of the `mwVarargout` object and must be explicitly assigned to a return value.

Function Reference

This section contains an alphabetical listing of the routines in the MATLAB C++ Math Library.

Note For information about the MATLAB C++ Math Library utility routines, see “Utility Routine Reference” on page -407. These routines appear in a separate alphabetical listing.

Arithmetic Operators

Purpose

Matrix and array arithmetic

C++ Prototype

```
mwArray plus(const mwArray &A, const mwArray &B);
mwArray minus(const mwArray &A, const mwArray &B);
mwArray mtimes(const mwArray &A, const mwArray &B);
mwArray mrdivide(const mwArray &A, const mwArray &B);
mwArray mpower(const mwArray &A, const mwArray &B);
mwArray mldivide(const mwArray &A, const mwArray &B);
mwArray transpose(const mwArray &A);
mwArray times(const mwArray &A, const mwArray &B);
mwArray rdivide(const mwArray &A, const mwArray &B);
mwArray ldivide(const mwArray &A, const mwArray &B);
mwArray power(const mwArray &A, const mwArray &B);
mwArray ctranspose(const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, B;                // Input argument(s)
mwArray C;                   // Return value

C = A + B;
C = plus(A,B);

C = A - B;
C = minus(A,B);

C = A * B;
C = mtimes(A,B);

C = A / B;
C = mrdivide(A,B);

C = A ^ B;
C = mpower(A,B);

C = mldivide(A,B);
C = transpose(A);
C = times(A,B);
C = rdivide(A,B);
C = ldivide(A,B);
C = power(A,B);
C = ctranspose(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

Relational Operators < > <= >= == !=

Purpose

Relational operations

C++ Prototype

```
mwArray lt(const mwArray &A, const mwArray &B);
mwArray gt(const mwArray &A, const mwArray &B);
mwArray le(const mwArray &A, const mwArray &B);
mwArray ge(const mwArray &A, const mwArray &B);
mwArray eq(const mwArray &A, const mwArray &B);
mwArray neq(const mwArray &A, const mwArray &B);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A,B;                // Input argument(s)
mwArray C;                  // Return value

C = A < B;
C = lt(A,B);

C = A > B;
C = gt(A,B);

C = A <= B;
C = le(A,B);

C = A >= B;
C = ge(A,B);

C = A == B;
C = eq(A,B);

C = A != B;
C = neq(A,B);
```

Relational Operators < > <= >= == !=

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

```
mwArray and_func(const mwArray &A, const mwArray &B);
mwArray or_func(const mwArray &A, const mwArray &B);
mwArray not_func(const mwArray &A);
```

C Syntax

```
#include "matlab.hpp"

mwArray A, B;               // Input argument(s)
mwArray C;                  // Return value

C = and_func(A,B);
C = or_func(A,B);
C = not_func(A);
```

MATLAB Syntax

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

See Also

MATLAB Logical Operators Calling Conventions

Purpose

Absolute value and complex magnitude

C++ Prototype

```
mwArray abs(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray Y;              // Return value

Y = abs(X);
```

MATLAB Syntax

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

See Also

MATLAB abs

Calling Conventions

acos, acosh

Purpose

Inverse cosine and inverse hyperbolic cosine

C++ Prototype

```
mwArray acos(const mwArray &X);
mwArray acosh(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray Y;              // Return value

Y = acos(X);
Y = acosh(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

```
mwArray acot(const mwArray &X);
mwArray acoth(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray Y;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `acot`, `acoth` Calling Conventions

acsc, acsch

Purpose

Inverse cosecant and inverse hyperbolic cosecant

C++ Prototype

```
mwArray acsc(const mwArray &X);
mwArray acsch(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray Y;              // Return value

Y = acsc(X);
Y = acsch(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

```
mwArray all(const mwArray &A);
mwArray all(const mwArray &A, const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, dim;         // Input argument(s)
mwArray B;              // Return value

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

MATLAB Syntax

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

See Also MATLAB `all` Calling Conventions

angle

Purpose

Phase angle

C++ Prototype

```
mwArray angle(const mwArray &Z);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray Z;              // Input argument(s)
mwArray P;              // Return value

P = angle(Z);
```

MATLAB Syntax

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

See Also

MATLAB `angle` Calling Conventions

Purpose Test for any nonzeros

C++ Prototype

```
mwArray any(const mwArray &A);
mwArray any(const mwArray &A, const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, dim;         // Input argument(s)
mwArray B;              // Return value

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

MATLAB Syntax

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

See Also MATLAB any Calling Conventions

asec, asech

Purpose

Inverse secant and inverse hyperbolic secant

C++ Prototype

```
mwArray asec(const mwArray &X);
mwArray asech(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray Y;              // Return value

Y = asec(X);
Y = asech(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

```
mwArray asin(const mwArray &X);
mwArray asinh(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray Y;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `asin`, `asinh` Calling Conventions

atan, atanh

Purpose

Inverse tangent and inverse hyperbolic tangent

C++ Prototype

```
mwArray atan(const mwArray &X);
mwArray atanh(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray Y;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `atan`, `atanh` Calling Conventions

Purpose

Four-quadrant inverse tangent

C++ Prototype

```
mwArray atan2(const mwArray &Y, const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray Y, X;           // Input argument(s)
mwArray P;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `atan2` Calling Conventions

balance

Purpose

Improve accuracy of computed eigenvalues

C++ Prototype

```
mwArray balance(mwArray *B, const mwArray &A);
mwArray balance(const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Input argument(s)
mwArray B;              // Output argument(s)
mwArray D;              // Return value

D = balance(&B,A);
B = balance(A);
```

MATLAB Syntax

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

See Also

MATLAB `balance` Calling Conventions

Purpose

Base to decimal number conversion

C++ Prototype

```
mwArray base2dec(const mwArray &strn, const mwArray &base);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray strn;           // String array(s)
mwArray base;           // Input argument(s)
mwArray d;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `base2dec` Calling Conventions

beta, betainc, betaln

Purpose

Beta functions

C++ Prototype

```
mwArray beta(const mwArray &Z, const mwArray &W);
mwArray betainc(const mwArray &X, const mwArray &Z,
                const mwArray &W);
mwArray betaln(const mwArray &Z, const mwArray &W);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray Z, W, X;        // Input argument(s)
mwArray B, I, L;        // Return value

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

BiConjugate Gradients method

C++ Prototype

```
mwArray bicg(const mwArray &A,
             const mwArray &b=mwArray::DIN,
             const mwArray &tol=mwArray::DIN,
             const mwArray &maxit=mwArray::DIN,
             const mwArray &M1=mwArray::DIN,
             const mwArray &M2=mwArray::DIN,
             const mwArray &x=mwArray::DIN,
             const mwVarargin &in8=mwVarargin::DIN,
             const mwArray &in9=mwArray::DIN,
             .
             .
             .
             const mwArray &in39=mwArray::DIN);

mwArray bicg(mwArray *out1, mwArray *out2,
             mwArray *out3, mwArray *out4,
             const mwArray &in1,
             const mwArray &in2=mwArray::DIN,
             const mwArray &in3=mwArray::DIN,
             const mwArray &in4=mwArray::DIN,
             const mwArray &in5=mwArray::DIN,
             const mwArray &in6=mwArray::DIN,
             const mwArray &in7=mwArray::DIN,
             const mwVarargin &in8=mwVarargin::DIN,
             const mwArray &in9=mwArray::DIN,
             .
             .
             .
             const mwArray &in39=mwArray::DIN);
```

bicg

C++ Syntax

```
#include "matlab.hpp"

mwArray A, b, tol, maxit, M, M1, M2, x0;// Input argument(s)
mwArray flag, relres, iter, resvec;     // Output argument(s)
mwArray x;                              // Return value

x = bicg(A,b);
x = bicg(A,b,tol);
x = bicg(A,b,tol,maxit);
x = bicg(A,b,tol,maxit,M);
x = bicg(A,b,tol,maxit,M1,M2);
x = bicg(A,b,tol,maxit,M1,M2,x0);
x = bicg(A,b,tol,maxit,M1,M2,x0);
x = bicg(&flag,A,b,tol,maxit,M1,M2,x0);
x = bicg(&flag,&relres,A,b,tol,maxit,M1,M2,x0);
x = bicg(&flag,&relres,&iter,A,b,tol,maxit,M1,M2,x0);
x = bicg(&flag,&relres,&iter,&resvec,A,b,tol,maxit,M1,M2,x0);
```

MATLAB Syntax

```
x = bicg(A,b)
bicg(A,b,tol)
bicg(A,b,tol,maxit)
bicg(A,b,tol,maxit,M)
bicg(A,b,tol,maxit,M1,M2)
bicg(A,b,tol,maxit,M1,M2,x0)
x = bicg(A,b,tol,maxit,M1,M2,x0)
[x,flag] = bicg(A,b,tol,maxit,M1,M2,x0)
[x,flag,relres] = bicg(A,b,tol,maxit,M1,M2,x0)
[x,flag,relres,iter] = bicg(A,b,tol,maxit,M1,M2,x0)
[x,flag,relres,iter,resvec] = bicg(A,b,tol,maxit,M1,M2,x0)
```

See Also

MATLAB `bicg` Calling Conventions

Purpose

BiConjugate Gradients Stabilized method

C++ Prototype

```
mwArray bicgstab(const mwArray &in1,
                 const mwArray &in2=mwArray::DIN,
                 const mwArray &in3=mwArray::DIN,
                 const mwArray &in4=mwArray::DIN,
                 const mwArray &in5=mwArray::DIN,
                 const mwArray &in6=mwArray::DIN,
                 const mwArray &in7=mwArray::DIN,
                 const mwVarargin &in8=mwVarargin::DIN,
                 const mwArray &in9=mwArray::DIN,
                 .
                 .
                 .
                 const mwArray &in39=mwArray::DIN);

mwArray bicgstab(mwArray *out1, mwArray *out2,
                 mwArray *out3, mwArray *out4,
                 const mwArray &in1,
                 const mwArray &in2=mwArray::DIN,
                 const mwArray &in3=mwArray::DIN,
                 const mwArray &in4=mwArray::DIN,
                 const mwArray &in5=mwArray::DIN,
                 const mwArray &in6=mwArray::DIN,
                 const mwArray &in7=mwArray::DIN,
                 const mwVarargin &in8=mwVarargin::DIN,
                 const mwArray &in9=mwArray::DIN,
                 .
                 .
                 .
                 const mwArray &in39=mwArray::DIN);
```

bicgstab

C++ Syntax

```
#include "matlab.hpp"

mwArray A, b, tol, maxit, M, M1, M2, x0;// Input argument(s)
mwArray flag, relres, iter, resvec;     // Output argument(s)
mwArray x;                              // Return value

x = bicgstab(A,b);
x = bicgstab(A,b,tol);
x = bicgstab(A,b,tol,maxit);
x = bicgstab(A,b,tol,maxit,M);
x = bicgstab(A,b,tol,maxit,M1,M2);
x = bicgstab(A,b,tol,maxit,M1,M2,x0);
x = bicgstab(A,b,tol,maxit,M1,M2,x0);
x = bicgstab(&flag,A,b,tol,maxit,M1,M2,x0);
x = bicgstab(&flag,&relres,A,b,tol,maxit,M1,M2,x0);
x = bicgstab(&flag,&relres,&iter,A,b,tol,maxit,M1,M2,x0);
x = bicgstab(&flag,&relres,&iter,&resvec,A,b,tol,maxit,M1,M2,x0);
```

MATLAB Syntax

```
x = bicgstab(A,b)
bicgstab(A,b,tol)
bicgstab(A,b,tol,maxit)
bicgstab(A,b,tol,maxit,M)
bicgstab(A,b,tol,maxit,M1,M2)
bicgstab(A,b,tol,maxit,M1,M2,x0)
x = bicgstab(A,b,tol,maxit,M1,M2,x0)
[x,flag] = bicgstab(A,b,tol,maxit,M1,M2,x0)
[x,flag,relres] = bicgstab(A,b,tol,maxit,M1,M2,x0)
[x,flag,relres,iter] = bicgstab(A,b,tol,maxit,M1,M2,x0)
[x,flag,relres,iter,resvec] = bicgstab(A,b,tol,maxit,M1,M2,x0)
```

See Also

MATLAB `bicgstab` Calling Conventions

Purpose

Binary to decimal number conversion

C++ Prototype

```
mwArray bin2dec(const mwArray &binarystr);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray binarystr;          // Input argument(s)
mwArray decnumber;          // Return value

decnumber = bin2dec(binarystr);
```

MATLAB Syntax

`bin2dec(`*`binarystr`*`)`

See Also

MATLAB `bin2dec` Calling Conventions

bitand_func

Purpose

Bit-wise AND

C++ Prototype

```
mwArray bitand_func(const mwArray &A,
                    const mwArray &B=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, B;               // Input argument(s)
mwArray C;                  // Return value

C = bitand_func(A,B);
```

MATLAB Syntax

```
C = bitand(A,B)
```

See Also

MATLAB `bitand` Calling Conventions

Purpose

Complement bits

C++ Prototype

```
mwArray bitcmp(const mwArray &A, const mwArray &n=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, n;               // Input argument(s)
mwArray C;                  // Return value

C = bitcmp(A,n);
```

MATLAB Syntax

```
C = bitcmp(A,n)
```

See Also

MATLAB `bitcmp` Calling Conventions

bitget

Purpose

Get bit

C++ Prototype

```
mwArray bitget(const mwArray &A, const mwArray &bit=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, bit;             // Input argument(s)
mwArray C;                  // Return value

C = bitget(A,bit);
```

MATLAB Syntax

```
C = bitget(A,bit)
```

See Also

MATLAB `bitget` Calling Conventions

Purpose

Maximum floating-point integer

C++ Prototype

```
mwArray bitmax();
```

C++ Syntax

```
#include "matlab.hpp"

mwArray C;              // Return value

C = bitmax();
```

MATLAB Syntax

```
bitmax
```

See Also

MATLAB `bitmax` Calling Conventions

bitor_func

Purpose

Bit-wise OR

C++ Prototype

```
mwArray bitor_func(const mwArray &A,
                   const mwArray &B=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, B;               // Input argument(s)
mwArray C;                  // Return value

C = bitor_func(A,B);
```

MATLAB Syntax

```
C = bitor(A,B)
```

See Also

MATLAB `bitor` Calling Conventions

Purpose

Set bit

C++ Prototype

```
mwArray bitset(const mwArray &A,
               const mwArray &bit=mwArray::DIN,
               const mwArray &v=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, bit;             // Input argument(s)
mwArray C;                  // Return value

C = bitset(A,bit);
C = bitset(A,bit,v);
```

MATLAB Syntax

```
C = bitset(A,bit)
C = bitset(A,bit,v)
```

See Also

MATLAB `bitset` Calling Conventions

bitshift

Purpose

Bit-wise shift

C++ Prototype

```
mwArray bitshift(const mwArray &A, const mwArray &k=mwArray::DIN,
                 const mwArray &n=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, k, n;            // Input argument(s)
mwArray C;                  // Return value

C = bitshift(A,k,n);
C = bitshift(A,n);
```

MATLAB Syntax

```
C = bitshift(A,k,n)
C = bitshift(A,k)
```

See Also

MATLAB `bitshift` Calling Conventions

Purpose

Bit-wise XOR

C++ Prototype

```
mwArray bitxor(const mwArray &A, const mwArray &B=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, B;               // Input argument(s)
mwArray C;                  // Return value

C = bitxor(A,B);
```

MATLAB Syntax

```
C = bitxor(A,B)
```

See Also

MATLAB `bitxor` Calling Conventions

blanks

Purpose

A string of blanks

C++ Prototype

```
mwArray blanks(const mwArray &n);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray n;              // Input argument(s)
mwArray r;              // Return value

r = blanks(n);
```

MATLAB Syntax

```
blanks(n)
```

See Also

MATLAB `blanks` Calling Conventions

Purpose

Calendar

C++ Prototype

```
mwArray calendar(const mwArray &in1=mwArray::DIN,
                 const mwArray &in2=mwArray::DIN);

void Vcalendar(const mwArray &in1=mwArray::DIN,
               const mwArray &in2=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

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

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

Vcalendar();
Vcalendar(d);
Vcalendar(y,m);
```

MATLAB Syntax

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

See Also

MATLAB `calendar` Calling Conventions

cart2pol

Purpose

Transform Cartesian coordinates to polar or cylindrical

C++ Prototype

```
mwArray cart2pol(mwArray *RHO, mwArray *Z_out, const mwArray &X,
                 const mwArray &Y, const mwArray &Z_in);
mwArray cart2pol(mwArray *RHO, const mwArray &X, const mwArray &Y);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, Y, Z_in;     // Input argument(s)
mwArray RHO, Z_out;     // Output argument(s)
mwArray THETA;          // Return value

THETA = cart2pol(&RHO,&Z_out,X,Y,Z_in);
THETA = cart2pol(&RHO,X,Y);
```

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

```
mwArray cart2sph(mwArray *PHI, mwArray *R, const mwArray &X,
                 const mwArray &Y, const mwArray &Z);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, Y, Z;        // Input argument(s)
mwArray PHI, R;         // Output argument(s)
mwArray THETA;          // Return value

THETA = cart2sph(&PHI,&R,X,Y,Z);
```

MATLAB Syntax

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

See Also MATLAB `cart2sph` Calling Conventions

cat

Purpose

Concatenate arrays

C++ Prototype

```
mwArray cat(const mwArray &in1,
            const mwVarargin &in2=mwVarargin::DIN,
            const mwArray &in3=mwArray::DIN,
            .
            .
            .
            const mwArray &in33=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, B;               // Input argument(s)
mwArray A1, A2, A3, A4;     // Input argument(s)
mwArray dim;                // Input argument(s)
mwArray C;                  // Return value

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

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

```
mwArray cdf2rdf(mwArray *D_out, const mwArray &V_in,
                const mwArray &D_in);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray V_in, D_in;         // Input argument(s)
mwArray D_out;              // Output argument(s)
mwArray V;                  // Return value

V = cdf2rdf(&D_out,V_in,D_in);
```

MATLAB Syntax

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

See Also

MATLAB `cdf2rdf` Calling Conventions

ceil

Purpose

Round toward infinity

C++ Prototype

```
mwArray ceil(const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Input argument(s)
mwArray B;              // Return value

B = ceil(A);
```

MATLAB Syntax

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

See Also

MATLAB `ceil` Calling Conventions

Purpose

Create cell array

C++ Prototype

```
mwArray cell(const mwVarargin &in1,
             const mwArray &in2=mwArray::DIN,
             .
             .
             .
             const mwArray &in32=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray m, n, p, A;     // Input argument(s)
mwArray c;              // Return value

c = cell(n);
c = cell(m,n);
c = cell(horzcat(m,n));
c = cell(m,n,p,...);
c = cell(horzcat(m,n,p,...));
c = cell(size(A));
```

MATLAB Syntax

```
c = cell(n)
c = cell(m,n)
c = cell([m n])
c = cell(m,n,p,...)
c = cell([m n p ...])
c = cell(size(A))
```

See Also

MATLAB `cell` Calling Conventions

cell2struct

Purpose

Cell array to structure array conversion

C++ Prototype

```
mwArray cell2struct(const mwArray &c,
                    const mwArray &fields=mwArray::DIN,
                    const mwArray &dim=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray c, fields, dim;             // Input argument(s)
mwArray s;                          // Return value

s = cell2struct(c,fields,dim);
```

MATLAB Syntax

```
s = cell2struct(c,fields,dim)
```

See Also

MATLAB `cell2struct` Calling Conventions

Purpose Display cell array contents.

C++ Prototype

```
void celldisp(const mwArray &C, const mwArray &name=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray C, name;        // Input argument(s)

celldisp(C);
celldisp(C,name);
```

MATLAB Syntax

```
celldisp(C)
celldisp(C,name)
```

See Also MATLAB `celldisp` Calling Conventions

cellfun

Purpose

Apply a function to each element in a cell array

C++ Prototype

```
mwArray cellfun(const mwArray &fname, const mwArray &C=mwArray::DIN,
                const mwArray &k=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray C, k;           // Input argument(s)
mwArray D;              // Return value

D = cellfun("fname",C);
D = cellfun("size",C,k);
D = cellfun('isclass',C,classname)
```

MATLAB Syntax

```
D = cellfun('fname',C)
D = cellfun('size',C,k)
D = cellfun('isclass',C,classname)
```

See Also

MATLAB `cellfun` Calling Conventions

Purpose

Horizontally concatenate cell arrays; replacement for MATLAB cell concatenation operator ({ })

C++ Prototype

```
mwArray cellhcat(const mwVarargin &in1,
                 const mwArray &in2=mwArray::DIN,
                 .
                 .
                 .
                 const mwArray &in32=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, B, C;        // Input argument(s)
mwArray D;              // Return value

D = cellhcat(A,B);
D = cellhcat(A,B,C);
.
.
.
```

MATLAB Syntax

```
D = { A B };
D = { A B C };
```

See Also

MATLAB Special Characters Calling Conventions

cellstr

Purpose

Create cell array of strings from character array

C++ Prototype

```
mwArray cellstr(const mwArray &S);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray S;              // Input argument(s)
mwArray c;              // Return value

c = cellstr(S);
```

MATLAB Syntax

```
c = cellstr(S)
```

See Also

MATLAB `cellstr` Calling Conventions

Purpose

Conjugate Gradients Squared method

C++ Prototype

```
mwArray cgs(const mwArray &in1,
            const mwArray &in2=mwArray::DIN,
            const mwArray &in3=mwArray::DIN,
            const mwArray &in4=mwArray::DIN,
            const mwArray &in5=mwArray::DIN,
            const mwArray &in6=mwArray::DIN,
            const mwArray &in7=mwArray::DIN,
            const mwVarargin &in8=mwVarargin::DIN,
            const mwArray &in9=mwArray::DIN,
            .
            .
            .
            const mwArray &in39=mwArray::DIN);

mwArray cgs(mwArray *out1, mwArray *out2,
            mwArray *out3, mwArray *out4,
            const mwArray &in1,
            const mwArray &in2=mwArray::DIN,
            const mwArray &in3=mwArray::DIN,
            const mwArray &in4=mwArray::DIN,
            const mwArray &in5=mwArray::DIN,
            const mwArray &in6=mwArray::DIN,
            const mwArray &in7=mwArray::DIN,
            const mwVarargin &in8=mwVarargin::DIN,
            const mwArray &in9=mwArray::DIN,
            .
            .
            .
            const mwArray &in39=mwArray::DIN);
```

cgs

C++ Syntax

```
#include "matlab.hpp"

mwArray A, b, tol, maxit, M, M1, M2, x0;// Input argument(s)
mwArray flag, relres, iter, resvec;     // Output argument(s)
mwArray x;                              // Return value

x = cgs(A,b);
x = cgs(A,b,tol);
x = cgs(A,b,tol,maxit);
x = cgs(A,b,tol,maxit,M);
x = cgs(A,b,tol,maxit,M1,M2);
x = cgs(A,b,tol,maxit,M1,M2,x0);
x = cgs(A,b,tol,maxit,M1,M2,x0);
x = cgs(&flag,A,b,tol,maxit,M1,M2,x0);
x = cgs(&flag,&relres,A,b,tol,maxit,M1,M2,x0);
x = cgs(&flag,&relres,&iter,A,b,tol,maxit,M1,M2,x0);
x = cgs(&flag,&relres,&iter,&resvec,A,b,tol,maxit,M1,M2,x0);
```

MATLAB Syntax

```
x = cgs(A,b)
cgs(A,b,tol)
cgs(A,b,tol,maxit)
cgs(A,b,tol,maxit,M)
cgs(A,b,tol,maxit,M1,M2)
cgs(A,b,tol,maxit,M1,M2,x0)
x = cgs(A,b,tol,maxit,M1,M2,x0)
[x,flag] = cgs(A,b,tol,maxit,M1,M2,x0)
[x,flag,relres] = cgs(A,b,tol,maxit,M1,M2,x0)
[x,flag,relres,iter] = cgs(A,b,tol,maxit,M1,M2,x0)
[x,flag,relres,iter,resvec] = cgs(A,b,tol,maxit,M1,M2,x0)
```

See Also MATLAB cgs Calling Conventions

Purpose

Create character array (string)

C++ Prototype

```
mwArray char_func(const mwVarargin &in1,
                  const mwArray &in2=mwArray::DIN,
                  .
                  .
                  .
                  const mwArray &in32=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, C, t1, t2, t3;   // Input argument(s)
mwArray S;                  // Return value

S = char_func(X);
S = char_func(C);
S = char_func(t1,t2,t3,...);
```

MATLAB Syntax

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

See Also

MATLAB `char` Calling Conventions

chol

Purpose

Cholesky factorization

C++ Prototype

```
mwArray chol(const mwArray &X);
mwArray chol(mwArray *p, const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray p;              // Output argument(s)
mwArray R;              // Return value

R = chol(X);
R = chol(&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

```
mwArray cholupdate(const mwArray &R, const mwArray &x);
mwArray cholupdate(const mwArray &R, const mwArray &x,
                   const mwArray &flag);
mwArray cholupdate(mwArray *p, const mwArray &R, const mwArray &x,
                   const mwArray &flag);
```

C Syntax

```
#include "matlab.hpp"

mxArray *R, *x;         // Input argument(s)
mxArray *p;             // Output argument(s)
mxArray *R1;            // Return value

R1 = cholupdate(R,x);
R1 = cholupdate(R,x,"+");
R1 = cholupdate(R,x,"-");
R1 = cholupdate(&p,R,x,"-");
```

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

cholinc

Purpose

Incomplete Cholesky factorizations

C++ Prototype

```
mwArray cholinc(const mwArray &X,
                const mwArray &droptol=mwArray::DIN);
mwArray cholinc(mwArray *p,
                const mwArray &X,
                const mwArray &droptol=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, droptol, options;        // Input argument(s)
mwArray p;                          // Output argument(s)
mwArray R;                          // Return value

R = cholinc(X,droptol);
R = cholinc(X,options);
R = cholinc(X,"O");
R = cholinc(&p,X,"O");
R = cholinc(X,"inf");
```

MATLAB Syntax

```
R = cholinc(X,droptol)
R = cholinc(X,options)
R = cholinc(X,'O')
[R,p] = cholinc(X,'O')
R = cholinc(X,'inf')
```

See Also

MATLAB `cholinc` Calling Conventions

Purpose

Create object or return class of object

C++ Prototype

```
mwArray classname(const mwArray &object);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray object;             // Input argument(s)
mwArray str;                // Return value

str = classname(object);
```

MATLAB Syntax

```
str = class(object)
obj = class(s,'class_name')
obj = class(s,'class_name',parent1,parent2...)
```

See Also

MATLAB `class` Calling Conventions

clock_func

Purpose

Current time as a date vector

C++ Prototype

```
mwArray clock_func();
```

C++ Syntax

```
#include "matlab.hpp"

mwArray c;              // Return value

c = clock_func();
```

MATLAB Syntax

```
c = clock
```

See Also

MATLAB `clock` Calling Conventions

Purpose

Generate a sequence of indices

C++ Prototype

```
mwIndex colon();
mwIndex colon(mwArray start, mwArray end);
mwIndex colon(mwArray start, mwArray step, mwArray end);
```

Arguments

`start`
Initial value

`step`
Increment value

`end`
Final value

C++ Syntax

```
B = A(colon());
B = A(colon(1,10));
B = A(colon(1,2,10));
```

MATLAB Syntax

```
colon = start:stop
colon = start:step:stop
```

See Also

MATLAB `colon` Calling Conventions

colmmd

Purpose

Sparse column minimum degree permutation

C++ Prototype

```
mwArray colmmd(const mwArray &S);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray S;                  // Input argument(s)
mwArray p;                  // Return value

p = colmmd(S);
```

MATLAB Syntax

```
p = colmmd(S)
```

See Also

MATLAB `colmmd` Calling Conventions

Purpose

Sparse column permutation based on nonzero count

C++ Prototype

```
mwArray colperm(const mwArray &S);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray S;                      // Input argument(s)
mwArray j;                      // Return value

j = colperm(S);
```

MATLAB Syntax

```
j = colperm(S)
```

See Also

MATLAB `colperm` Calling Conventions

compan

Purpose

Companion matrix

C++ Prototype

```
mwArray compan(const mwArray &u);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray u;              // Input argument(s)
mwArray A;              // Return value

A = compan(u);
```

MATLAB Syntax

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

See Also

MATLAB compan Calling Conventions

Purpose

Identify the computer on which MATLAB is running

C++ Prototype

```
mwArray computer();
mwArray computer(mwArray *maxsize);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray maxsize;        // Output argument(s)
mwArray str;            // Return value

str = computer();
maxsize = computer(&maxsize);
```

MATLAB Syntax

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

See Also

MATLAB `computer` Calling Conventions

cond

Purpose

Condition number with respect to inversion

C++ Prototype

```
mwArray cond(const mwArray &X);
mwArray cond(const mwArray &X, const mwArray &p);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, p;               // Input argument(s)
mwArray c;                  // Return value

c = cond(X);
c = cond(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

```
mwArray condeig(const mwArray &A);
mwArray condeig(mwDoubleMatrix *D, mwArray *s, const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Input argument(s)
mwArray D, s;           // Output argument(s)
mwArray c, V;           // Return value

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

MATLAB Syntax

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

See Also

MATLAB `condeig` Calling Conventions

condest

Purpose

1-norm matrix condition number estimate

C++ Prototype

```
mwArray condest(const mwArray &A);
mwArray condest(mwArray *v, const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Input argument(s)
mwArray v;              // Output argument(s)
mwArray c;              // Return value

c = condest(A);
c = condest(&v,A);
```

MATLAB Syntax

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

See Also

MATLAB `condest` Calling Conventions

Purpose

Complex conjugate

C++ Prototype

```
mwArray conj(const mwArray &Z);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray Z;              // Input argument(s)
mwArray ZC;             // Return value

ZC = conj(Z);
```

MATLAB Syntax

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

See Also

MATLAB `conj` Calling Conventions

conv

Purpose

Convolution and polynomial multiplication

C++ Prototype

```
mwArray conv(const mwArray &u, const mwArray &v);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray u, v;           // Input argument(s)
mwArray w;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB conv

Calling Conventions

Purpose

Two-dimensional convolution

C++ Prototype

```
mwArray conv2(const mwArray &A, const mwArray &B);
mwArray conv2(const mwArray &hcol, const mwArray &hrow,
              const mwArray &A);
mwArray conv2(const mwArray &hcol, const mwArray &hrow,
              const mwArray &A, const mwArray &shape);
```

C++ Syntax

```
#include "matlab.hpp"

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

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

MATLAB Syntax

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

See Also

MATLAB `conv2` Calling Conventions

corrcoef

Purpose

Correlation coefficients

C++ Prototype

```
mwArray corrcoef(const mwArray &X);
mwArray corrcoef(const mwArray &x, const mwArray &y);
```

C++ Syntax

```
#include "matlab.hpp"

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

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

MATLAB Syntax

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

See Also

MATLAB `corrcoef` Calling Conventions

Purpose

Cosine and hyperbolic cosine

C++ Prototype

```
mwArray cos(const mwArray &X);
mwArray cosh(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray Y;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `cos`, `cosh` Calling Conventions

cot, coth

Purpose

Cotangent and hyperbolic cotangent

C++ Prototype

```
mwArray cot(const mwArray &X);
mwArray coth(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray Y;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `cot`, `coth` Calling Conventions

Purpose

Covariance matrix

C++ Prototype

```
mwArray cov(const mwArray &in1,
            const mwVarargin &in2=mwVarargin::DIN,
            const mwArray &in3=mwArray::DIN,
            .
            .
            .
            const mwArray &in33=mwArray::DIN)
```

C++ Syntax

```
#include "matlab.hpp"

mwArray x, y, Z;        // Input argument(s)
mwArray C;              // Return value

C = cov(x);
C = cov(x,y);
C = cov(x,y,Z);
```

MATLAB Syntax

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

See Also

MATLAB cov Calling Conventions

cplxpair

Purpose

Sort complex numbers into complex conjugate pairs

C++ Prototype

```
mwArray cplxpair(const mwArray &A);
mwArray cplxpair(const mwArray &A, const mwArray &tol);
mwArray cplxpair(const mwArray &A, const mwArray &tol,
                 const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, tol, dim;    // Input argument(s)
mwArray B;              // Return value

B = cplxpair(A);
B = cplxpair(A,tol);
B = cplxpair(A,empty(),dim);
B = cplxpair(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

```
mwArray cross(const mwArray &U, const mwArray &V);
mwArray cross(const mwArray &U, const mwArray &V,
              const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray U, V, dim;      // Input argument(s)
mwArray W;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `cross` Calling Conventions

csc, csch

Purpose

Cosecant and hyperbolic cosecant

C++ Prototype

```
mwArray csc(const mwArray &x);
mwArray csch(const mwArray &x);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray x;              // Input argument(s)
mwArray Y;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB csc, csch Calling Conventions

Purpose

Cumulative product

C++ Prototype

```
mwArray cumprod(const mwArray &A);
mwArray cumprod(const mwArray &A, const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, dim;             // Input argument(s)
mwArray B;                  // Return value

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

MATLAB Syntax

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

See Also

MATLAB `cumprod` Calling Conventions

cumsum

Purpose

Cumulative sum

C++ Prototype

```
mwArray cumsum(const mwArray &A);
mwArray cumsum(const mwArray &A, const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, dim;         // Input argument(s)
mwArray B;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `cumsum` Calling Conventions

Purpose

Cumulative trapezoidal numerical integration

C++ Prototype

```
mwArray cumtrapz(const mwArray &Y);
mwArray cumtrapz(const mwArray &X, const mwArray &Y);
mwArray cumtrapz(const mwArray &X, const mwArray &Y,
                 const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, Y, dim;      // Input argument(s)
mwArray Z;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `cumtrapz` Calling Conventions

date

Purpose

Current date string

C++ Prototype

```
mwArray date();
```

C++ Syntax

```
#include "matlab.hpp"

mwArray str;            // Return value

str = date();
```

MATLAB Syntax

```
str = date
```

See Also

MATLAB `date`

Calling Conventions

Purpose

Serial date number

C++ Prototype

```
mwArray datenum(const mwArray &Y,
                const mwArray &M=mwArray::DIN,
                const mwArray &D=mwArray::DIN,
                const mwArray &H=mwArray::DIN,
                const mwArray &MI=mwArray::DIN,
                const mwArray &S=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray str, P;             // String array(s)
mwArray Y, M, D, H, MI, S;  // Input argument(s)
mwArray N;                  // Return value

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

MATLAB Syntax

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

See Also

MATLAB `datenum` Calling Conventions

datestr

Purpose

Date string format

C++ Prototype

```
mwArray datestr(const mwArray &D,
                const mwArray &dateform=mwArray::DIN,
                const mwArray &P=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray dateform;       // Number or string
mwArray D, P;           // Input argument(s)
mwArray str;            // Return value

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

MATLAB Syntax

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

See Also

MATLAB `datestr` Calling Conventions

Purpose

Date components

C++ Prototype

```
mwArray datevec(const mwArray &A,
                const mwArray &P=mwArray::DIN);

mwArray datevec(mwArray *M, const mwArray &A,
                const mwArray &P=mwArray::DIN);

mwArray datevec(mwArray *M, mwArray *D,
                const mwArray &A,
                const mwArray &P=mwArray::DIN);

mwArray datevec(mwArray *M, mwArray *D, mwArray *H,
                const mwArray &A,
                const mwArray &P=mwArray::DIN);

mwArray datevec(mwArray *M, mwArray *D, mwArray *H,
                mwArray *MI, const mwArray &A,
                const mwArray &P=mwArray::DIN)

mwArray datevec(mwArray *M, mwArray *D, mwArray *H,
                mwArray *MI, mwArray *S, const mwArray &A,
                const mwArray &P=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Input argument(s)
mwArray M, D, H, MI, S; // Output argument(s)
mwArray C, Y;           // Return value

C = datevec(A);
C = datevec(A,P);
Y = datevec(&M,&D,&H,&MI,&S,A);
Y = datevec(&M,&D,&H,&MI,&S,A,P);
```

datevec

MATLAB Syntax

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

See Also

MATLAB `datevec` Calling Conventions

Purpose

Numerical double integration

C++ Prototype

```
mwArray dblquad(const mwArray &intfcn,
                const mwArray &inmin=mwArray::DIN,
                const mwArray &inmax=mwArray::DIN,
                const mwArray &outmin=mwArray::DIN,
                const mwArray &outmax=mwArray::DIN,
                const mwArray &tol=mwArray::DIN,
                const mwArray &method=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray func;                       // String array(s)
mwArray inmin, inmax, outmin;       // Input argument(s)
mwArray outmax, tol, method;        // Input argument(s)
mwArray result;                     // Return value

result = dblquad(func,inmin,inmax,outmin,outmax);
result = dblquad(func,inmin,inmax,outmin,outmax,tol);
result = dblquad(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

deal

Purpose

Deal inputs to outputs

C++ Prototype

```
mwArray deal(const mwVarargin &in1,
             const mwArray &in2=mwArray::DIN,
             .
             .
             .
             const mwArray &in32=mwArray::DIN);
mwArray deal(mwVarargout varargout,
             const mwVarargin &in1,
             const mwArray &in2=mwArray::DIN,
             .
             .
             .
             const mwArray &in32=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, X1, X2, X3;  // Input argument(s)
mwArray Y1, Y2, Y3, Y4; // Output argument(s)
mwArray Y;              // Return value

Y = deal(X);
deal(mwVarargout(Y1),X);
deal(mwVarargout(Y1,Y2),X);
deal(mwVarargout(Y1,Y2,Y3,...),X);
deal(mwVarargout(Y1),X1);
deal(mwVarargout(Y1,Y2),X1,X2);
deal(mwVarargout(Y1,Y2,Y3,...),X1,X2,X3,...);
```

MATLAB Syntax

```
[Y1,Y2,Y3,...] = deal(X)
[Y1,Y2,Y3,...] = deal(X1,X2,X3,...)
```

See Also

MATLAB `deal` Calling Conventions

Purpose

Strip trailing blanks from the end of a string

C++ Prototype

```
mwArray deblank(const mwArray &string);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray string, c_in;   // String array(s)
mwArray str, c;         // Return value

str = deblank(string);
c = deblank(c_in);
```

MATLAB Syntax

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

See Also

MATLAB `deblank` Calling Conventions

dec2base

Purpose

Decimal number to base conversion

C++ Prototype

```
mwArray dec2base(const mwArray &d, const mwArray &base);
mwArray dec2base(const mwArray &d, const mwArray &base,
                 const mwArray &n);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray d, base, n;         // Input argument(s)
mwArray str;                // Return value

str = dec2base(d,base);
str = dec2base(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

```
mwArray dec2bin(const mwArray &d);
mwArray dec2bin(const mwArray &d, const mwArray &n);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray d, n;               // Input argument(s)
mwArray str;                // Return value

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

MATLAB Syntax

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

See Also MATLAB `dec2bin` Calling Conventions

dec2hex

Purpose

Decimal to hexadecimal number conversion

C++ Prototype

```
mwArray dec2hex(const mwArray &d);
mwArray dec2hex(const mwArray &d, const mwArray &n);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray d, n;           // Input argument(s)
mwArray str;            // Return value

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

MATLAB Syntax

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

See Also

MATLAB `dec2hex` Calling Conventions

Purpose

Deconvolution and polynomial division

C++ Prototype

```
mwArray deconv(mwArray *r, const mwArray &v, const mwArray &u);
mwArray deconv(const mwArray &v, const mwArray &u);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray v, u;           // Input argument(s)
mwArray r;              // Output argument(s)
mwArray q;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `deconv` Calling Conventions

del2

Purpose

Discrete Laplacian

C++ Prototype

```
mwArray del2(const mwArray &in1,
             const mwVarargin &in2=mwVarargin::DIN,
             const mwArray &in3=mwArray::DIN,
             .
             .
             .
             const mwArray &in33=mwArray::DIN)
```

C++ Syntax

```
#include "matlab.hpp"

mwArray U, h, hx, hy;       // Input argument(s)
mwArray L;                  // Return value

L = del2(U);
L = del2(U,h);
L = del2(U,hx,hy);
L = del2(U,hx,hy,hz,...);
```

MATLAB Syntax

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

See Also

MATLAB `del2` Calling Conventions

Purpose

Matrix determinant

C++ Prototype

```
mwArray det(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray d;              // Return value

d = det(X);
```

MATLAB Syntax

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

See Also

MATLAB `det` Calling Conventions

deval

Purpose

Evaluate the solution of a differential equation problem

C++ Prototype

```
mwArray deval(const mwArray &sol,
              const mwArray &xint);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray sol;            // Input argument(s)
mwArray xint;           // Input argument(s)
mwArray sxint;          // Return value

sxint = deval(sol, xint);
```

MATLAB Syntax

```
sol = deval(sol, xint)
```

See Also

MATLAB `deval` Calling Conventions

Purpose

Diagonal matrices and diagonals of a matrix

C++ Prototype

```
mwArray diag(const mwArray &v, const mwArray &k);
mwArray diag(const mwArray &v);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray v, k, X;

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

MATLAB Syntax

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

See Also

MATLAB `diag` Calling Conventions

diff

Purpose

Differences and approximate derivatives

C++ Prototype

```
mwArray diff(const mwArray &X);
mwArray diff(const mwArray &X, const mwArray &n);
mwArray diff(const mwArray &X, const mwArray &n,
             const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, n, dim;      // Input argument(s)
mwArray Y;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `diff` Calling Conventions

Purpose

Display text or array

C++ Prototype

```
void disp(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)

disp(X);
```

MATLAB Syntax

```
disp(X)
```

See Also

MATLAB `disp` Calling Conventions

dmperm

Purpose

Dulmage-Mendelsohn decomposition

C ++ Prototype

```
mwArray dmperm(const mwArray &A);
mwArray dmperm(mwArray *q, mwArray *r, const mwArray &A);
mwArray dmperm(mwArray *q, mwArray *r, mwArray *s,
               const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Input argument(s)
mwArray q, r, s;        // Output argument(s)
mwArray p;              // Return value

p = dmperm(A);
p = dmperm(&q,&r,A);
p = dmperm(&q,&r,&s,A);
```

MATLAB Syntax

```
p = dmperm(A)
[p,q,r] = dmperm(A)
[p,q,r,s] = dmperm(A)
```

See Also

MATLAB `dmperm` Calling Conventions

Purpose Convert to double precision

C++ Prototype

```
mwArray double_func(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;                  // Input argument(s)
mwArray R;                  // Return value

R = double_func(X);
```

MATLAB Syntax

```
double(X)
```

See Also MATLAB `double` Calling Conventions

eig

Purpose

Eigenvalues and eigenvectors

C++ Prototype

```
mwArray eig(const mwArray &A);
mwArray eig(mwArray *D, const mwArray &A);
mwArray eig(mwArray &A, const mwArray &B);
mwArray eig(mwArray *D, const mwArray &A, const mwArray &B);
mwArray eig(mwArray *D, const mwArray &A, const mwArray &B, const
  mwArray &flag);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, B, flag;         // Input argument(s)
mwArray D;                  // Output argument(s)
mwArray d, V;               // Return value

d = eig(A);
V = eig(&D,A);
V = eig(&D,A,"nobalance");
d = eig(A,B);
V = eig(&D,A,B);
V = eig(&D,A,B,flag);
```

MATLAB Syntax

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

See Also

MATLAB `eig` Calling Conventions

Purpose

Find a few eigenvalues and eigenvectors

C++ Prototype

```
mwArray eigs(const mwVarargin &in1,
             const mwArray &in2=mwArray::DIN,
             .
             .
             .
             const mwArray &in32=mwArray::DIN)

mwArray eigs(mwArray *out1,
             const mwVarargin &in1,
             const mwArray &in2=mwArray::DIN,
             .
             .
             .
             const mwArray &in32=mwArray::DIN);

mwArray eigs(mwArray *out1, mwArray *out2,
             const mwVarargin &in1,
             const mwArray &in2=mwArray::DIN,
             .
             .
             .
             const mwArray &in32=mwArray::DIN);
```

eigs

C++ Syntax

```
#include "matlab.hpp"

mwArray A, n, B, k, sigma, options; // Input argument(s)
mwArray D, flag;                    // Output argument(s)
mwArray d, V;                       // Return value

d = eigs(A);
d = eigs("Afun",n);
d = eigs(A,B,k,sigma,options);
d = eigs("Afun",n,B,k,sigma,options);

V = eigs(&D,A);
V = eigs(&D,"Afun",n);
V = eigs(&D,A,B,k,sigma,options);
V = eigs(&D,"Afun",n,B,k,sigma,options);

V = eigs(&D,&flag,A);
V = eigs(&D,&flag,"Afun",n);
V = eigs(&D,&flag,A,B,k,sigma,options);
V = eigs(&D,&flag,"Afun",n,B,k,sigma,options);
```

MATLAB Syntax

```
d = eigs(A)
d = eigs('Afun',n)
d = eigs(A,B,k,sigma,options)
d = eigs('Afun',n,B,k,sigma,options)
[V,D] = eigs(A,...)
[V,D] = eigs('Afun',n,...)
[V,D,flag] = eigs(A,...)
[V,D,flag] = eigs('Afun',n,...)
```

See Also

MATLAB `eigs` Calling Conventions

Purpose

Jacobi elliptic functions

C++ Prototype

```
mwArray ellipj(mwArray *CN, mwArray *DN, const mwArray &U,
               const mwArray &M);
mwArray ellipj(mwArray *CN, mwArray *DN, const mwArray &U,
                const mwArray &M, const mwArray &tol);
mwArray ellipj(mwArray *CN, const mwArray &U, const mwArray &M);
mwArray ellipj(const mwArray &U, const mwArray &M);
mwArray ellipj(mwArray *CN, const mwArray &U, const mwArray &M,
               const mwArray &tol);
mwArray ellipj(const mwArray &U, const mwArray &M,
               const mwArray &tol);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray U, M, tol;      // Input argument(s)
mwArray CN, DN;         // Output argument(s)
mwArray SN;             // Return value

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

MATLAB Syntax

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

See Also

MATLAB `ellipj` Calling Conventions

ellipke

Purpose

Complete elliptic integrals of the first and second kind

C++ Prototype

```
mwArray ellipke(const mwArray &M);
mwArray ellipke(mwArray *E, const mwArray &M);
mwArray ellipke(mwArray *E, const mwArray &M, const mwArray &tol);
mwArray ellipke(const mwArray &M, const mwArray &tol);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray M, tol;         // Input argument(s)
mwArray E;              // Output argument(s)
mwArray K;              // Return value

K = ellipke(M);
K = ellipke(&E,M);
K = ellipke(&E,M,tol);
K = ellipke(M,tol);
```

MATLAB Syntax

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

See Also

MATLAB `ellipke` Calling Conventions

Purpose

Return an empty double matrix

C++ Prototype

```
mwArray empty();
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Return value

A = empty();
```

MATLAB Syntax

```
A = [];
```

end

Purpose

Generate the last index for an array dimension

C++ Prototype

```
mwArray end(mwArray &mat, mwArray &x, mwArray &y);
```

Arguments

`mat`
Array

`x`
The dimension where `end()` is used. (1 = row , 2 = column)

`y`
Number of indices in the subscript (for two-dimensional indexing, always 2; for one-dimensional indexing, always 1).

C++ Syntax

This example selects all but the first element in row three from array `A`:

```
A(3, colon(2, end(A, 2, 2)));
```

MATLAB Syntax

```
A(3, 2:end)
```

See Also

MATLAB `end` Calling Conventions

Purpose

End of month

C++ Prototype

```
mwArray eomday(const mwArray &Y, const mwArray &M);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray Y, M;           // Input argument(s)
mwArray E;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `eomday` Calling Conventions

eps

Purpose

Floating-point relative accuracy

C++ Prototype

```
mwArray eps();
```

C++ Syntax

```
#include "matlab.hpp"

mwArray R;              // Return value

R = eps();
```

MATLAB Syntax

```
eps
```

See Also

MATLAB `eps` Calling Conventions

Purpose

Error functions

C++ Prototype

```
mwArray erf(const mwArray &X);
mwArray erfc(const mwArray &X);
mwArray erfcx(const mwArray &X);
mwArray erfinv(const mwArray &Y);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, Y;

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
```

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`, `erfcx`, `erfinv` Calling Conventions

error

Purpose

Display error messages

C++ Prototype

```
void error(const mwArray &msg);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray msg;            // String array(s)

error(msg);
```

MATLAB Syntax

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

See Also

MATLAB `error` Calling Conventions

Purpose

Elapsed time

C++ Prototype

```
mwArray etime(const mwArray &t2, const mwArray &t1);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray t2, t1;         // Input argument(s)
mwArray e;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `etime` Calling Conventions

exp

Purpose

Exponential

C++ Prototype

```
mwArray exp(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray Y;              // Return value

Y = exp(X);
```

MATLAB Syntax

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

See Also

MATLAB `exp` Calling Conventions

Purpose

Exponential integral

C++ Prototype

```
mwArray expint(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray Y;              // Return value

Y = expint(X);
```

MATLAB Syntax

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

See Also

MATLAB `expint` Calling Conventions

expm

Purpose

Matrix exponential

C++ Prototype

```
mwArray expm(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray Y;              // Return value

Y = expm(X);
```

MATLAB Syntax

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

See Also

MATLAB `expm` Calling Conventions

Purpose Matrix exponential via Pade approximation

C++ Prototype

```
mwArray expm1(const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Input argument(s)
mwArray E;              // Return value

E = expm1(A);
```

MATLAB Syntax

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

See Also MATLAB `expm` Calling Conventions

expm2

Purpose

Matrix exponential via Taylor series

C++ Prototype

```
mwArray expm2(const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Input argument(s)
mwArray E;              // Return value

E = expm2(A);
```

MATLAB Syntax

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

See Also

MATLAB `expm` Calling Conventions

Purpose Matrix exponential via eigenvalues and eigenvectors

C++ Prototype

```
mwArray expm3(const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Input argument(s)
mwArray E;              // Return value

E = expm3(A);
```

MATLAB Syntax

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

See Also MATLAB `expm` Calling Conventions

eye

Purpose

Identity matrix

C++ Prototype

```
mwArray eye(const mwArray &n);
mwArray eye(const mwArray &m, const mwArray &n);
mwArray eye();
```

C++ Syntax

```
#include "matlab.hpp"

mwArray m, n, A;            // Input argument(s)
mwArray Y;                  // Return value

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

MATLAB Syntax

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

See Also

MATLAB `eye` Calling Conventions

Purpose

Prime factors

C++ Prototype

```
mwArray factor(const mwArray &n);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray n;              // Input argument(s)
mwArray f;              // Return value

f = factor(n);
```

MATLAB Syntax

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

See Also

MATLAB `factor` Calling Conventions

fclose

Purpose

Close one or more open files

C++ Prototype

```
mwArray fclose(const mwArray &fid);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray fid;                // Input argument(s)
mwArray status;             // Return value

status = fclose(fid);
status = fclose("all");
```

MATLAB Syntax

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

See Also

MATLAB `fclose` Calling Conventions

Purpose

Test for end-of-file

C++ Prototype

```
mwArray feof(const mwArray &fid);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray fid;            // Input argument(s)
mwArray eofstat;        // Return value

eofstat = feof(fid);
```

MATLAB Syntax

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

See Also

MATLAB `feof` Calling Conventions

ferror

Purpose

Query MATLAB about errors in file input or output

C++ Prototype

```
mwArray ferror(const mwArray &fid);
mwArray ferror(const mwArray &fid, const mwArray &clear);
mwArray ferror(mwArray *errnum, const mwArray &fid);
mwArray ferror(mwArray *errnum, const mwArray &fid,
               const mwArray &clear);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray fid;               // Input argument(s)
mwArray errnum;            // Output argument(s)
mwArray message;           // Return value

message = ferror(fid);
message = ferror(fid,"clear");
message = ferror(&errnum,fid);
message = ferror(&errnum,fid,"clear");
```

MATLAB Syntax

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

See Also

MATLAB `ferror` Calling Conventions

Purpose

Function evaluation

C++ Prototype

```
mwArray feval(mwVarargout vout,
              mlxFunctionPtr fcn, /* ptr to a function */
              const mwVarargin &in1,
              const mwArray &in2=mwArray::DIN,
              .
              .
              .
              const mwArray &in32=mwArray::DIN);

mwArray feval(mwVarargout vout,
              const mwArray &fcn, /* function name as string */
              const mwVarargin &in1,
              const mwArray &in2=mwArray::DIN,
              .
              .
              .
              const mwArray &in32=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray fcn,x1;             // Input argument(s)
mwArray y2;                 // Output argument(s)
mwArray y1;                 // Return value

y1 = feval(fcn);
y1 = feval(fcn,x1);
y1 = feval(&y2,fcn,x1,...);
y1 = feval("func");
y1 = feval("func",x1);
y1 = feval(&y2,"func",x1,...);
```

MATLAB Syntax

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

See Also

MATLAB `feval` Calling Conventions

fft

Purpose

One-dimensional fast Fourier transform

C++ Prototype

```
mwArray fft(const mwArray &X);
mwArray fft(const mwArray &X, const mwArray &n);
mwArray fft(const mwArray &X, const mwArray &n, const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, n, dim;          // Input argument(s)
mwArray Y;                  // Return value

Y = fft(X);
Y = fft(X,n);
Y = fft(X,empty(),dim);
Y = fft(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

```
mwArray fft2(const mwArray &X);
mwArray fft2(const mwArray &X, const mwArray &m, const mwArray &n);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, m, n;        // Input argument(s)
mwArray Y;              // Return value

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

MATLAB Syntax

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

See Also MATLAB `fft2` Calling Conventions

fftn

Purpose

Multidimensional fast Fourier transform

C++ Prototype

```
mwArray fftn(const mwArray &X, const mwArray &siz=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, siz;         // Input argument(s)
mwArray Y;              // Return value

Y = fftn(X)
Y = fftn(X,siz)
```

MATLAB Syntax

```
Y = fftn(X)
Y = fftn(X,siz)
```

See Also

MATLAB `fftn`

Calling Conventions

Purpose

Shift DC component of fast Fourier transform to center of spectrum

C++ Prototype

```
mwArray fftshift(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray Y;              // Return value

Y = fftshift(X);
```

MATLAB Syntax

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

See Also

MATLAB `fftshift` Calling Conventions

fgetl

Purpose

Return the next line of a file as a string without line terminator(s)

C++ Prototype

```
mwArray fgetl(const mwArray &fid);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray fid;            // Input argument(s)
mwArray line;           // Return value

line = fgetl(fid);
```

MATLAB Syntax

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

See Also

MATLAB `fgetl` Calling Conventions

Purpose

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

C++ Prototype

```
mwArray fgets(const mwArray &fid);
mwArray fgets(const mwArray &fid, const mwArray &nchar);
mwArray fgets(mwArray *EOL, const mwArray &fid);
mwArray fgets(mwArray *EOL, const mwArray &fid,
              const mwArray &nchar);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray fid, nchar;     // Input argument(s)
mwArray EOL;            // Output argument(s)
mwArray line;           // Return value

line = fgets(fid);
line = fgets(fid,nchar);
line = fgets(&EOL,fid);
line = fgets(&EOL,fid,nchar);
```

MATLAB Syntax

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

See Also

MATLAB `fgets` Calling Conventions

fieldnames

Purpose

Field names of a structure

C++ Prototype

```
mwArray fieldnames(const mwArray &s);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray s;                          // Input argument(s)
mwArray names;                      // Return value

names = fieldnames(s);
```

MATLAB Syntax

```
names = fieldnames(s)
```

See Also

MATLAB `fieldnames` Calling Conventions

Purpose

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

C++ Prototype

```
mwArray filter(const mwArray &b, const mwArray &a,
               const mwArray &X);
mwArray filter(mwArray *zf, const mwArray &b, const mwArray &a,
               const mwArray &X);
mwArray filter(const mwArray &b, const mwArray &a, const mwArray &X,
               const mwArray &zi);
mwArray filter(mwArray *zf, const mwArray &b, const mwArray &a,
               const mwArray &X, const mwArray &zi);
mwArray filter(const mwArray &b, const mwArray &a, const mwArray &X,
               const mwArray &zi, const mwArray &dim);
mwArray filter(mwArray *zf, const mwArray &b, const mwArray &a,
               const mwArray &X, const mwArray &zi,
               const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray b, a, X, zi, dim;   // Input argument(s)
mwArray zf;                 // Output argument(s)
mwArray y;                  // Return value

y = filter(b,a,X);
y = filter(&zf,b,a,X);
y = filter(b,a,X,zi);
y = filter(&zf,b,a,X,zi);
y = filter(b,a,X,zi,dim);
y = filter(&zf,b,a,X,zi,dim);
y = filter(b,a,X,empty(),dim);
y = filter(&zf,b,a,X,empty(),dim);
```

filter

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

Purpose

Two-dimensional digital filtering

C++ Prototype

```
mwArray filter2(const mwArray &h, const mwArray &X);
mwArray filter2(const mwArray &h, const mwArray &X,
                  const mwArray &shape);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray shape;          // String array(s)
mwArray h, X;           // Input argument(s)
mwArray Y;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `filter2` Calling Conventions

find

Purpose

Find indices and values of nonzero elements

C++ Prototype

```
mwArray find(const mwArray &X);
mwArray find(mwArray *j, const mwArray &X);
mwArray find(mwArray *j, mwArray *v, const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray j, v;           // Output argument(s)
mwArray k, i;           // Return value

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

MATLAB Syntax

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

See Also

MATLAB `find` Calling Conventions

Purpose

Find one string within another

C++ Prototype

```
mwArray findstr(const mwArray &str1, const mwArray &str2);
```

C++ Syntax

```
#include "matlab.hpp"

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

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

MATLAB Syntax

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

See Also

MATLAB `findstr` Calling Conventions

fix

Purpose

Round towards zero

C++ Prototype

```
mwArray fix(const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Input argument(s)
mwArray B;              // Return value

B = fix(A);
```

MATLAB Syntax

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

See Also

MATLAB `fix` Calling Conventions

Purpose

Flip matrices left-right

C++ Prototype

```
mwArray fliplr(const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Input argument(s)
mwArray B;              // Return value

B = fliplr(A);
```

MATLAB Syntax

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

See Also

MATLAB `fliplr` Calling Conventions

flipud

Purpose

Flip matrices up-down

C++ Prototype

```
mwArray flipud(const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Input argument(s)
mwArray B;              // Return value

B = flipud(A);
```

MATLAB Syntax

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

See Also

MATLAB `flipud` Calling Conventions

Purpose

Round towards minus infinity

C++ Prototype

```
mwArray floor(const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Input argument(s)
mwArray B;              // Return value

B = floor(A);
```

MATLAB Syntax

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

See Also

MATLAB `floor` Calling Conventions

Description

`B = floor(A)` rounds the elements of `A` to the nearest integers less than or equal to `A`. For complex `A`, the imaginary and real parts are rounded independently.

flops

Purpose

Count floating-point operations

Note This is an obsolete function. With the incorporation of LAPACK in MATLAB version 6, counting floating-point operations is no longer practical.

C++ Prototype

```
mwArray flops();
mwArray flops(const mwArray &m);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray f;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `flops` Calling Conventions

Purpose

Minimize a function of one variable

Note The `fmin` routine was replaced by `fminbnd` in Release 11 (MATLAB 5.3). In Release 12 (MATLAB 6.0), `fmin` displays a warning and calls `fminbnd`.

C++ Prototype

```
mwArray fmin(const mwArray &in1,
             const mwArray &in2=mwArray::DIN,
             const mwArray &in3=mwArray::DIN,
             const mwArray &in4=mwArray::DIN,
             const mwVarargin &in5=mwVarargin::DIN,
             const mwArray &in6=mwArray::DIN,
             .
             .
             .
             const mwArray &in36=mwArray::DIN);

mwArray fmin(mwArray *out1,
             const mwArray &in1,
             const mwArray &in2=mwArray::DIN,
             const mwArray &in3=mwArray::DIN,
             const mwArray &in4=mwArray::DIN,
             const mwVarargin &in5=mwVarargin::DIN,
             const mwArray &in6=mwArray::DIN,
             .
             .
             .
             const mwArray &in36=mwArray::DIN);
```

fmin

C++ Syntax

```
#include "matlab.hpp"

mwArray func;               // String array(s)
mwArray x1, x2;             // Input argument(s)
mwArray options_in, P1, P2; // Input argument(s)
mwArray options_out;        // Output argument(s)
mwArray x;                  // Return value

x = fmin(func,x1,x2);
x = fmin(func,x1,x2,options_in);
x = fmin(func,x1,x2,options_in,P1,P2,...);
x = fmin(&options_out,func,x1,x2);
x = fmin(&options_out,func,x1,x2,options_in);
x = fmin(&options_out,func,x1,x2,options_in,P1,P2,...);
```

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 one variable on a fixed interval

C++ Prototype

```
mwArray fminbnd(const mwArray &in1,
                const mwArray &in2=mwArray::DIN,
                const mwArray &in3=mwArray::DIN,
                const mwArray &in4=mwArray::DIN,
                const mwVarargin &in5=mwVarargin::DIN,
                const mwArray &in6=mwArray::DIN,
               .
               .
               .
                const mwArray &in36=mwArray::DIN);

mwArray fminbnd(mwArray *out1,
                const mwArray &in1,
                const mwArray &in2=mwArray::DIN,
                const mwArray &in3=mwArray::DIN,
                const mwArray &in4=mwArray::DIN,
                const mwVarargin &in5=mwVarargin::DIN,
                const mwArray &in6=mwArray::DIN,
               .
               .
               .
                const mwArray &in36=mwArray::DIN);

mwArray fminbnd(mwArray *out1,
                mwArray *out2,
                const mwArray &in1,
                const mwArray &in2=mwArray::DIN,
                const mwArray &in3=mwArray::DIN,
                const mwArray &in4=mwArray::DIN,
                const mwVarargin &in5=mwVarargin::DIN,
                const mwArray &in6=mwArray::DIN,
               .
               .
               .
                const mwArray &in36=mwArray::DIN);

mwArray fminbnd(mwArray *out1,
```

```
 mwArray *out2,
 mwArray *out3,
 const mwArray &in1,
 const mwArray &in2=mwArray::DIN,
 const mwArray &in3=mwArray::DIN,
 const mwArray &in4=mwArray::DIN,
 const mwVarargin &in5=mwVarargin::DIN,
 const mwArray &in6=mwArray::DIN,
.
.
.
 const mwArray &in36=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray func;               // String array(s)
mwArray x1, x2;             // Input argument(s)
mwArray options, P1, P2;    // Input argument(s)
mwArray fval;               // Output argument(s)
mwArray exitflag;           // Output argument(s)
mwArray output;             // Output argument(s)
mwArray x;                  // Return value

/* MATLAB syntax: x = fminbnd(func,x1,x2) */
x = fminbnd(func,x1,x2);

/* MATLAB syntax: x = fminbnd(func,x1,x2,options) */
x = fminbnd(func,x1,x2,options);

/* MATLAB syntax: x = fminbnd(func,x1,x2,options,P1,P2,...) */
x = fminbnd(func,x1,x2,options,P1,P2,...);

/* MATLAB syntax: [x,fval] = fminbnd(...) */
x = fminbnd(&fval,func,x1,x2);
x = fminbnd(&fval,func,x1,x2,options);
x = fminbnd(&fval,func,x1,x2,options,P1,P2,...);

/* MATLAB syntax: [x,fval,exitflag] = fminbnd(...) */
x = fminbnd(&fval,&exitflag,func,x1,x2);
x = fminbnd(&fval,&exitflag,func,x1,x2,options);
x = fminbnd(&fval,&exitflag,func,x1,x2,options,P1,P2,...);

/* MATLAB syntax: [x,fval,exitflag,output] = fminbnd(...) */
x = fminbnd(&fval,&exitflag,&output,func,x1,x2);
x = fminbnd(&fval,&exitflag,&output,func,x1,x2,options);
x = fminbnd(&fval,&exitflag,&output,func,x1,x2,options,P1,P2,...);
```

MATLAB Syntax

```
x = fminbnd(func,x1,x2)
x = fminbnd(func,x1,x2,options)
x = fminbnd(func,x1,x2,options,P1,P2,...)
[x,fval] = fminbnd(...)
[x,fval,exitflag] = fminbnd(...)
[x,fval,exitflag,output] = fminbnd(...)
```

See Also MATLAB `fminbnd` Calling Conventions

fmins

Purpose

Minimize a function of several variables

Note The `fmins` routine was replaced by `fminsearch` in Release 11 (MATLAB 5.3). In Release 12 (MATLAB 6.0), `fmins` displays a warning and calls `fminsearch`.

C++ Prototype

```
mwArray fmins(const mwArray &in1,
              const mwArray &in2=mwArray::DIN,
              const mwArray &in3=mwArray::DIN,
              const mwArray &in4=mwArray::DIN,
              const mwVarargin &in5=mwVarargin::DIN,
              const mwArray &in6=mwArray::DIN,
              .
              .
              .
              const mwArray &in36=mwArray::DIN);

mwArray fmins(mwArray *out1,
              const mwArray &in1,
              const mwArray &in2=mwArray::DIN,
              const mwArray &in3=mwArray::DIN,
              const mwArray &in4=mwArray::DIN,
              const mwVarargin &in5=mwVarargin::DIN,
              const mwArray &in6=mwArray::DIN,
              .
              .
              .
              const mwArray &in36=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray func;                   // String array(s)
mwArray x0, options_in;         // Input argument(s)
mwArray P1, P2;                 // Input argument(s)
mwArray options_out;            // Output argument(s)
mwArray x;                      // Return value

x = fmins(func,x0);
x = fmins(func,x0,options_in);
x = fmins(func,x0,options_in,empty(),P1,P2,...);
x = fmins(&options_out,func,x0);
x = fmins(&options_out,func,x0,options_in);
x = fmins(&options_out,func,x0,options_in,empty(),P1,P2,...);
```

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

fminsearch

Purpose Minimize a function of several variables

C++ Prototype

```
mwArray fminsearch(const mwArray &in1,
                   const mwArray &in2=mwArray::DIN,
                   const mwArray &in3=mwArray::DIN,
                   const mwVarargin &in4=mwVarargin::DIN,
                   const mwArray &in5=mwArray::DIN,
                   .
                   .
                   .
                   const mwArray &in35=mwArray::DIN);

mwArray fminsearch(mwArray *out1,
                   const mwArray &in1,
                   const mwArray &in2=mwArray::DIN,
                   const mwArray &in3=mwArray::DIN,
                   const mwVarargin &in4=mwVarargin::DIN,
                   const mwArray &in5=mwArray::DIN,
                   .
                   .
                   .
                   const mwArray &in35=mwArray::DIN);

mwArray fminsearch(mwArray *out1,
                   mwArray *out2,
                   const mwArray &in1,
                   const mwArray &in2=mwArray::DIN,
                   const mwArray &in3=mwArray::DIN,
                   const mwVarargin &in4=mwVarargin::DIN,
                   const mwArray &in5=mwArray::DIN,
                   .
                   .
                   .
                   const mwArray &in35=mwArray::DIN);

mwArray fminsearch(mwArray *out1,
                   mwArray *out2,
                   mwArray *out3,
                   const mwArray &in1,
```

```
const mwArray &in2=mwArray::DIN,
const mwArray &in3=mwArray::DIN,
const mwVarargin &in4=mwVarargin::DIN,
const mwArray &in5=mwArray::DIN,
.
.
.
const mwArray &in35=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray func;                   // String array(s)
mwArray x0, options;            // Input argument(s)
mwArray P1, P2;                 // Input argument(s)
mwArray fval;                   // Output argument(s)
mwArray exitflag;               // Output argument(s)
mwArray output;                 // Output argument(s)
mwArray x;                      // Return value

/* MATLAB syntax: x = fminsearch(func,x0) */
x = fminsearch(func,x0);

/* MATLAB syntax: x = fminsearch(func,x0,options) */
x = fminsearch(func,x0,options);

/* MATLAB syntax: x = fminsearch(func,x0,options,P1,P2,...) */
x = fminsearch(func,x0,options,P1,P2,...);

/* MATLAB syntax: [x,fval] = fminsearch(...) */
x = fminsearch(&fval,func,x0);
x = fminsearch(&fval,func,x0,options);
x = fminsearch(&fval,func,x0,options,P1,P2,...);

/* MATLAB syntax: [x,fval,exitflag] = fminsearch(...) */
x = fminsearch(&fval,&exitflag,func,x0);
x = fminsearch(&fval,&exitflag,func,x0,options);
x = fminsearch(&fval,&exitflag,func,x0,options,P1,P2,...);

/* MATLAB syntax: [x,fval,exitflag,output] = fminsearch(...) */
x = fminsearch(&fval,&exitflag,&output,func,x0);
x = fminsearch(&fval,&exitflag,&output,func,x0,options);
x = fminsearch(&fval,&exitflag,&output,func,x0,options,P1,P2,...);
```

fminsearch

MATLAB Syntax

```
x = fminsearch(fun,x0)
x = fminsearch(fun,x0,options)
x = fminsearch(fun,x0,options,P1,P2,...)
[x,fval] = fminsearch(...)
[x,fval,exitflag] = fminsearch(...)
[x,fval,exitflag,output] = fminsearch(...)
```

See Also

MATLAB `fminsearch` Calling Conventions

Purpose

Open a file or obtain information about open files

C++ Prototype

```
mwArray fopen(const mwArray &filename, const mwArray &permission);
mwArray fopen(mwArray *message, const mwArray &filename,
              const mwArray &permission, const mwArray &format);
mwArray fopen(const mwArray &all);
mwArray fopen(mwArray *permission, mwArray *format,
              const mwArray &fid);
mwArray fopen(const mwArray &filename, const mwArray &permission,
              const mwArray &format);
```

C++ Syntax

```
#include "matlab.hpp"

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

fid = fopen(filename,permission);
fid = fopen(&message,filename,permission,format);
fids = fopen("all");
filename = fopen(&permission,&format,fid);
fid = fopen(filename,permission,format);
```

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

format

Purpose

Control the output display format

C++ Prototype

```
void format();
void format(const mwArray &a);
void format(const mwArray &a, const mwArray &b);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray a, b;               // Input argument(s)

format();
format(a);
format(a,b);
```

MATLAB Syntax

MATLAB performs all computations in double precision.

See Also

MATLAB `format` Calling Conventions

Purpose

Write formatted data to file

C++ Prototype

```
mwArray fprintf(const mwArray &fid,
                const mwVarargin &format=mwVarargin::DIN,
                const mwArray &A3=mwArray::DIN,
                .
                .
                .
                const mwArray &A33=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray format;             // String array(s)
mwArray fid;                // Input argument(s)
mwArray A1, A2;             // Input argument(s)
mwArray count;              // Return value

count = fprintf(fid,format,A1);
count = fprintf(fid,format,A1,A2,...);
count = fprintf(format,A1);
count = fprintf(format,A1,A2,...);
count = fprintf(format);
```

MATLAB Syntax

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

See Also

MATLAB `fprintf` Calling Conventions

fread

Purpose

Read binary data from file

C++ Prototype

```
mwArray fread(mwArray *count, const mwArray &fid,
              const mwArray &size, const mwArray &precision);
mwArray fread(mwArray *count, const mwArray &fid,
              const mwArray &size, const mwArray &precision,
              const mwArray &skip);
mwArray fread(const mwArray &fid);
mwArray fread(mwArray *count, const mwArray &fid);
mwArray fread(const mwArray &fid, const mwArray &size);
mwArray fread(mwArray *count, const mwArray &fid,
              const mwArray &size);
mwArray fread(const mwArray &fid, const mwArray &size,
               const mwArray &precision);
mwArray fread(const mwArray &fid, const mwArray &size,
              const mwArray &precision, const mwArray &skip);
```

C++ Syntax

```
#include "matlab.hpp"

mwArrat precision;              // Input argument(s)
mwArray fid, size, skip;        // Input argument(s)
mwArray count;                  // Output argument(s)
mwArray A;                      // Return value

A = fread(&count,fid,size,precision);
A = fread(&count,fid,size,precision,skip);
A = fread(fid);
A = fread(&count,fid);
A = fread(fid,size);
A = fread(&count,fid,size);
A = fread(fid,size,precision);
A = fread(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

Purpose

Determine frequency spacing for frequency response

C++ Prototype

```
mwArray freqspace(mwArray *f2, const mwArray &n);
mwArray freqspace(const mwArray &n);
mwArray freqspace(mwArray *f2, const mwArray &n,
                  const mwArray &flag);
mwArray freqspace(const mwArray &N, const mwArray &flag);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray n, N, flag;         // Input argument(s)
mwArray f2, y1;             // Output argument(s)
mwArray f1, x1, f;          // Return value

f1 = freqspace(&f2,n);
f1 = freqspace(&f2,horzcat(m,n));
x1 = freqspace(&y1,n,"meshgrid");
x1 = freqspace(&y1,horzcat(m,n),"meshgrid")
f = freqspace(N);
f = freqspace(N,"whole");
```

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

frewind

Purpose

Rewind an open file

C++ Prototype

```
mwArray frewind(const mwArray &fid);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray fid;            // Input argument(s)
mwArray R;              // Return value

R = frewind(fid);
```

MATLAB Syntax

```
frewind(fid)
```

See Also

MATLAB `frewind` Calling Conventions

Purpose

Read formatted data from file

C++ Prototype

```
mwArray fscanf(const mwArray &fid, const mwArray &format);
mwArray fscanf(mwArray *count, const mwArray &fid,
               const mwArray &format, const mwArray &size);
mwArray fscanf(const mwArray &fid, const mwArray &format,
               const mwArray &size);
mwArray fscanf(mwArray *count, const mwArray &fid,
               const mwArray &format);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray format;             // String array(s)
mwArray fid, size;          // Input argument(s)
mwArray count;              // Output argument(s)
mwArray A;                  // Return value

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

MATLAB Syntax

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

See Also

MATLAB `fscanf` Calling Conventions

fseek

Purpose

Set file position indicator

C++ Prototype

```
mwArray fseek(const mwArray &fid, const mwArray &offset,
              const mwArray &origin);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray origin;             // String array(s)
mwArray fid, offset;        // Input argument(s)
mwArray status;             // Return value

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

MATLAB Syntax

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

See Also

MATLAB `fseek` Calling Conventions

Purpose

Get file position indicator

C++ Prototype

```
mwArray ftell(const mwArray &fid);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray fid;                // Input argument(s)
mwArray position;           // Return value

position = ftell(fid);
```

MATLAB Syntax

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

See Also

MATLAB `ftell` Calling Conventions

full

Purpose

Convert sparse matrix to full matrix

C++ Prototype

```
mwArray full(const mwArray &S);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray S;                  // Input argument(s)
mwArray A;                  // Return value

A = full(S);
```

MATLAB Syntax

```
A = full(S)
```

See Also

MATLAB `full` Calling Conventions

Purpose

Evaluate functions of a matrix

C++ Prototype

```
mwArray funm(const mwArray &X, const mwArray &func);
mwArray funm(mwArray *estrr, const mwArray &X,
             const mwArray &func);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray func;           // String array(s)
mwArray X;              // Input argument(s)
mwArray estrr;          // Output argument(s)
mwArray Y;              // Return value

Y = funm(X,func);
Y = funm(&estrr,X,func);
```

MATLAB Syntax

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

See Also

MATLAB `funm` Calling Conventions

fwrite

Purpose

Write binary data to a file

C++ Prototype

```
mwArray fwrite(const mwArray &fid, const mwArray &A,
                  const mwArray &precision);
mwArray fwrite(const mwArray &fid, const mwArray &A,
                  const mwArray &precision, const mwArray &skip);
mwArray fwrite(const mwArray &fid, const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray precision;              // Input argument(s)
mwArray fid, A, skip;           // Input argument(s)
mwArray count;                  // Return value

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

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

```
mwArray fzero(const mwArray &in1,
              const mwArray &in2=mwArray::DIN,
              const mwVarargin &in3=mwVarargin::DIN,
              const mwArray &in4=mwArray::DIN,
              .
              .
              .
              const mwArray &in34=mwArray::DIN);

mwArray fzero(mwArray *out1,
              const mwArray &in1,
              const mwArray &in2=mwArray::DIN,
              const mwVarargin &in3=mwVarargin::DIN,
              const mwArray &in4=mwArray::DIN,
              .
              .
              .
              const mwArray &in34=mwArray::DIN);

mwArray fzero(mwArray *out1,
              mwArray *out2,
              const mwArray &in1,
              const mwArray &in2=mwArray::DIN,
              const mwVarargin &in3=mwVarargin::DIN,
              const mwArray &in4=mwArray::DIN,
              .
              .
              .
              const mwArray &in34=mwArray::DIN);
```

```
mwArray fzero(mwArray *out1,
              mwArray *out2,
              mwArray *out3,
              const mwArray &in1,
              const mwArray &in2=mwArray::DIN,
              const mwVarargin &in3=mwVarargin::DIN,
              const mwArray &in4=mwArray::DIN,
              .
              .
              .
              const mwArray &in34=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray fun;                    // String array(s)
mwArray x0, options;            // Input argument(s)
mwArray P1, P2;                 // Input argument(s)
mwArray fval, exitflag, output; // Output argument(s)
mwArray x;                      // Return value

x = fzero(fun,x0);
x = fzero(fun,x0,options);
x = fzero(fun,x0,options,P1,P2,...);

x = fzero(&fval,fun,x0);
x = fzero(&fval,fun,x0,options);
x = fzero(&fval,fun,x0,options,P1,P2,...);

x = fzero(&fval,&exitflag,fun,x0);
x = fzero(&fval,&exitflag,fun,x0,options);
x = fzero(&fval,&exitflag,fun,x0,options,P1,P2,...);

x = fzero(&fval,&exitflag,&output,fun,x0);
x = fzero(&fval,&exitflag,&output,fun,x0,options);
x = fzero(&fval,&exitflag,&output,fun,x0,options,P1,P2,...);
```

MATLAB Syntax

```
x = fzero(fun,x0)
x = fzero(fun,x0,options)
x = fzero(fun,x0,options,P1,P2,...)
[x,fval] = fzero(...)
[x,fval,exitflag] = fzero(...)
[x,fval,exitflag,output] = fzero(...)
```

See Also MATLAB `fzero` Calling Conventions

gamma, gammainc, gammaln

Purpose

Gamma functions

C++ Prototype

```
mwArray gamma(const mwArray &A);
mwArray gamma(const mwArray &X, const mwArray &A);
mwArray gammainc(const mwArray &X, const mwArray &A);
mwArray gammaln(const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

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

Y = gamma(A);                   // Gamma function
Y = gamma(X,A);
Y = gammainc(X,A);              // Incomplete gamma function
Y = gammaln(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, gammalnCalling Conventions

Purpose

Greatest common divisor

C++ Prototype

```
mwArray gcd(const mwArray &A, const mwArray &B);
mwArray gcd(mwArray *C, mwArray *D, const mwArray &A,
            const mwArray &B);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, B;           // Input argument(s)
mwArray C, D;           // Output argument(s)
mwArray G;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `gcd` Calling Conventions

getfield

Purpose

Get field of structure array

C++ Prototype

```
mwArray getfield(const mwArray &in1,
                 const mwVarargin &in2=mwVarargin::DIN,
                 const mwArray &in3=mwArray::DIN,
                 .
                 .
                 .
                 const mwArray &in33=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray s;                      // Input argument(s)
mwArray i,j,k;                  // Input argument(s)
mwArray f;                      // Return value

f = getfield(s,"field");
f = getfield(s,cellhcat(i,j),"field",cellhcat(k));
```

MATLAB Syntax

```
f = getfield(s,'field')
f = getfield(s,{i,j},'field',{k})
```

See Also

MATLAB `getfield` Calling Conventions

Purpose

Generalized Minimum Residual method (with restarts)

C++ Prototype

```
mwArray gmres(const mwArray &in1,
              const mwArray &in2=mwArray::DIN,
              const mwArray &in3=mwArray::DIN,
              const mwArray &in4=mwArray::DIN,
              const mwArray &in5=mwArray::DIN,
              const mwArray &in6=mwArray::DIN,
              const mwArray &in7=mwArray::DIN,
              const mwArray &in8=mwArray::DIN,
              const mwVarargin &in9=mwVarargin::DIN,
              const mwArray &in10=mwArray::DIN,
              .
              .
              .
              const mwArray &in40=mwArray::DIN);

mwArray gmres(mwArray *out1, mwArray *out2,
              mwArray *out3, mwArray *out4,
              const mwArray &in1,
              const mwArray &in2=mwArray::DIN,
              const mwArray &in3=mwArray::DIN,
              const mwArray &in4=mwArray::DIN,
              const mwArray &in5=mwArray::DIN,
              const mwArray &in6=mwArray::DIN,
              const mwArray &in7=mwArray::DIN,
              const mwArray &in8=mwArray::DIN,
              const mwVarargin &in9=mwVarargin::DIN,
              const mwArray &in10=mwArray::DIN,
              .
              .
              .
              const mwArray &in40=mwArray::DIN);
```

gmres

C++ Syntax

```
#include "matlab.hpp"

mwArray A, b, restart, tol;         // Input argument(s)
mwArray maxit, M, M1, M2, x0;       // Input argument(s)
mwArray flag, relres, iter, resvec;     // Output argument(s)
mwArray x;                              // Return value

x = gmres(A,b,restart);
x = gmres(A,b,restart,tol);
x = gmres(A,b,restart,tol,maxit);
x = gmres(A,b,restart,tol,maxit,M);
x = gmres(A,b,restart,tol,maxit,M1,M2);
x = gmres(A,b,restart,tol,maxit,M1,M2,x0);
x = gmres(A,b,restart,tol,maxit,M1,M2,x0);
x = gmres(&flag,A,b,restart,tol,maxit,M1,M2,x0);
x = gmres(&flag,&relres,A,b,restart,tol,maxit,M1,M2,x0);
x = gmres(&flag,&relres,&iter,A,b,restart,tol,maxit,M1,M2,x0);
x = gmres(&flag,&relres,&iter,&resvec,
           A,b,restart,tol,maxit,M1,M2,x0);
```

MATLAB Syntax

```
x = gmres(A,b,restart)
gmres(A,b,restart,tol)
gmres(A,b,restart,tol,maxit)
gmres(A,b,restart,tol,maxit,M)
gmres(A,b,restart,tol,maxit,M1,M2)
gmres(A,b,restart,tol,maxit,M1,M2,x0)
x = gmres(A,b,restart,tol,maxit,M1,M2,x0)
[x,flag] = gmres(A,b,restart,tol,maxit,M1,M2,x0)
[x,flag,relres] = gmres(A,b,restart,tol,maxit,M1,M2,x0)
[x,flag,relres,iter] = gmres(A,b,restart,tol,maxit,M1,M2,x0)
[x,flag,relres,iter,resvec] = gmres(A,b,restart,tol,maxit,M1,M2,x0)
```

See Also

MATLAB `gmres` Calling Conventions

Purpose

Numerical gradient

C++ Prototype

```
mwArray gradient(const mwArray &in1,
                 const mwVarargin &in2=mwVarargin::DIN,
                 const mwArray &in3=mwArray::DIN,
                 .
                 .
                 .
                 const mwArray &in33=mwArray::DIN);

mwArray gradient(mwVarargout varargout,
                 const mwArray &in1,
                 const mwVarargin &in2=mwVarargin::DIN,
                 const mwArray &in3=mwArray::DIN,
                 .
                 .
                 .
                 const mwArray &in33=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray F, h, h1, h2;       // Input argument(s)
mwArray FY, FZ;             // Output argument(s)
mwArray FX;                 // Return value

FX = gradient(F);
gradient(mwVarargout(FX,FY),F);
gradient(mwVarargout(FX,FY,FZ,...),F);

FX = gradient(F,h);
gradient(mwVarargout(FX,FY),F,h);
gradient(mwVarargout(FX,FY,FZ,...),F,h);

FX = gradient(F,h1,h2,...);
gradient(mwVarargout(FX,FY),F,h1,h2,...);
gradient(mwVarargout(FX,FY,FZ,...),F,h1,h2,...);
```

gradient

MATLAB Syntax

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

See Also MATLAB `gradient` Calling Conventions

Purpose

Data gridding

C++ Prototype

```
mwArray griddata(const mwArray &x, const mwArray &y,
                 const mwArray &z, const mwArray &XI,
                 const mwArray &YI);
mwArray griddata(mwArray *YI, mwArray *ZI, const mwArray &x,
                 const mwArray &y, const mwArray &z,
                 const mwArray &xi, const mwArray &YI);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray x, y, z, xi, yi;    // Input argument(s)
mwArray YI;                 // Output argument(s)
mwArray ZI, XI;

ZI = griddata(x,y,z,XI,YI);
XI = griddata(&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)
[...] = griddata(...,method)
```

See Also

MATLAB `griddata` Calling Conventions

hadamard

Purpose

Hadamard matrix

C++ Prototype

```
mwArray hadamard(const mwArray &n);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray n;                  // Input argument(s)
mwArray H;                  // Return value

H = hadamard(n);
```

MATLAB Syntax

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

See Also

MATLAB `hadamard` Calling Conventions

Purpose Hankel matrix

C++ Prototype

```
mwArray hankel(const mwArray &c);
mwArray hankel(const mwArray &c, const mwArray &r);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray c, r;           // Input argument(s)
mwArray H;              // Return value

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

MATLAB Syntax

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

See Also MATLAB `hankel` Calling Conventions

hess

Purpose

Hessenberg form of a matrix

C++ Prototype

```
mwArray hess(mwArray *H, const mwArray &A);
mwArray hess(const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, H, P;            // Input argument(s)

P = hess(&H,A);
H = hess(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

```
mwArray hex2dec(const mwArray &hex_value);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray hex_value;      // Hexadecimal integer or string array
mwArray d;              // Return value

d = hex2dec(hex_value);
```

MATLAB Syntax

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

See Also

MATLAB `hex2dec` Calling Conventions

hex2num

Purpose

Hexadecimal to double number conversion

C++ Prototype

```
mwArray hex2num(const mwArray &hex_value);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray hex_value;      // String array(s)
mwArray f;              // Return value

f = hex2num(hex_value);
```

MATLAB Syntax

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

See Also

MATLAB `hex2num` Calling Conventions

Purpose

Hilbert matrix

C++ Prototype

```
mwArray hilb(const mwArray &n);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray n;              // Input argument(s)
mwArray H;              // Return value

H = hilb(n);
```

MATLAB Syntax

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

See Also

MATLAB `hilb` Calling Conventions

horzcat

Purpose

Horizontal concatenation

C++ Prototype

```
mwArray horzcat(const mwVarargin &in1,
                const mwArray &in2=mwArray::DIN,
                .
                .
                .
                const mwArray &in32=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, B, C;            // Input argument(s)
mwArray R;                  // Return value

R = horzcat(A);
R = horzcat(A,B);
R = horzcat(A,B,C,...);
```

MATLAB Syntax

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

See Also

Calling Conventions

Purpose

Imaginary unit

C++ Prototype

```
mwArray i();
```

C++ Syntax

```
#include "matlab.hpp"

mwArray R;                  // Return value

R = i();
```

MATLAB Syntax

```
i
```

See Also

MATLAB `i` Calling Conventions

icubic

Purpose

One-dimensional cubic interpolation

This MATLAB 4 function has been subsumed into `interp1` in MATLAB 5.

See Also

MATLAB `interp1` Calling Conventions

Purpose

Inverse one-dimensional fast Fourier transform

C++ Prototype

```
mwArray ifft(const mwArray &X);
mwArray ifft(const mwArray &X, const mwArray &n);
mwArray ifft(const mwArray &X, const mwArray &n,
             const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, n, dim;      // Input argument(s)
mwArray y;              // Return value

y = ifft(X);
y = ifft(X,n);
y = ifft(X,empty(),dim);
y = ifft(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

ifft2

Purpose

Inverse two-dimensional fast Fourier transform

C++ Prototype

```
mwArray ifft2(const mwArray &X);
mwArray ifft2(const mwArray &X, const mwArray &m, const mwArray &n);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, m, n;            // Input argument(s)
mwArray Y;                  // Return value

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

MATLAB Syntax

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

See Also

MATLAB `ifft2` Calling Conventions

Purpose Inverse multidimensional fast Fourier transform

C++ Prototype

```
mwArray ifftn(const mwArray &X, const mwArray &siz=mwArray::DIN);
```

C++ Syntax

```
#include “matlab.h”

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

Y = ifftn(X);
Y = ifftn(X,siz);
```

MATLAB Syntax

```
Y = ifftn(X)
Y = ifftn(X,siz)
```

See Also MATLAB `ifftn` Calling Conventions

imag

Purpose

Imaginary part of a complex number

C++ Prototype

```
mwArray imag(const mwArray &Z);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray Z;              // Input argument(s)
mwArray Y;              // Return value

Y = imag(Z);
```

MATLAB Syntax

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

See Also

MATLAB `imag` Calling Conventions

Purpose

Subscripts from linear index

C++ Prototype

```
mwArray ind2sub(const mwArray &in1,
                const mwArray &in2=mwArray::DIN);


mwArray ind2sub(mwVarargout varargout,
                const mwArray &in1,
                const mwArray &in2=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray siz, IND;       // Input argument(s)
mwArray J, I1, I2, I3;  // Output argument(s)
mwArray I;              // Return value

I = ind2sub(&J,siz,IND);
I = ind2sub(mwVarargout(I1,I2,I3,...),siz,IND);
```

MATLAB Syntax

```
[I,J] = ind2sub(siz,IND)
[I1,I2,I3,...,In] = ind2sub(siz,IND)
```

inf

Purpose

Infinity

C++ Prototype

```
mwArray inf();
```

C++ Syntax

```
#include "matlab.hpp"

mwArray R;              // Return value

R = inf();
```

MATLAB Syntax

```
Inf
```

See Also

MATLAB `inf` Calling Conventions

Purpose

Detect points inside a polygonal region

C++ Prototype

```
mwArray inpolygon(const mwArray &x, const mwArray &y,
                  const mwArray &xv, const mwArray &yv);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, Y, xv, yv;       // Input argument(s)
mwArray IN;                 // Return value

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

MATLAB Syntax

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

See Also

MATLAB `inpolygon` Calling Conventions

int2str

Purpose

Integer to string conversion

C++ Prototype

```
mwArray int2str(const mwArray &N);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray N;              // Input argument(s)
mwArray str;            // Return value

str = int2str(N);
```

MATLAB Syntax

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

See Also

MATLAB `int2str` Calling Conventions

Purpose

One-dimensional data interpolation (table lookup)

C++ Prototype

```
mwArray interp1(const mwVarargin &in1,
                const mwArray &in2=mwArray::DIN,
                .
                .
                .
                const mwArray &in32=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray method;             // String array(s)
mwArray x, Y, xi;           // Input argument(s)
mwArray yi;                 // Return value

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

MATLAB Syntax

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

See Also

MATLAB `interp1` Calling Conventions

interp1q

Purpose

Quick one-dimensional linear interpolation

C++ Prototype

```
mwArray interp1q(const mwArray &x,
                 const mwArray &Y=mwArray::DIN,
                 const mwArray &xi=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray x, Y, xi;           // Input argument(s)
mwArray F;                  // Return value

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

MATLAB Syntax

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

See Also

Calling Conventions

Purpose

Two-dimensional data interpolation (table lookup)

C++ Prototype

```
mwArray interp2(const mwVarargin &in1,
                const mwArray &in2=mwArray::DIN,
                .
                .
                .
                const mwArray &in32=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

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

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

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

interp4

Purpose

Two-dimensional bilinear data interpolation

This MATLAB 4 function has been subsumed by `interp2` in MATLAB 5.

See Also

MATLAB `interp2` Calling Conventions

Purpose

Two-dimensional bicubic data interpolation

This MATLAB 4 function has been subsumed by `interp2` in MATLAB 5.

See Also

MATLAB `interp2` Calling Conventions

interp6

Purpose

Two-dimensional nearest neighbor interpolation

This MATLAB 4 function has been subsumed by `interp2` in MATLAB 5.

See Also

MATLAB `interp2` Calling Conventions

Purpose

One-dimensional interpolation using the fast Fourier transform method

C++ Prototype

```
mwArray interpft(const mwArray &x, const mwArray &n);
mwArray interpft(const mwArray &x, const mwArray &n,
                 const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray x, n, dim;          // Input argument(s)
mwArray y;                  // Return value

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

MATLAB Syntax

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

See Also

MATLAB `interpft` Calling Conventions

intersect

Purpose

Set intersection of two vectors

C++ Prototype

```
mwArray intersect(const mwArray &in1,
                  const mwArray &in2=mwArray::DIN,
                  const mwArray &in3=mwArray::DIN);

mwArray intersect(mwArray *out1,
                  mwArray *out2,
                  const mwArray &in1,
                  const mwArray &in2=mwArray::DIN,
                  const mwArray &in3=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray a, b;                       // Input argument(s)
mwArray ia, ib;                     // Output argument(s)
mwArray c;                          // Return value

c = intersect(a,b);
c = intersect(A,B,"rows");
c = intersect(&ia,&ib,a,b);
c = intersect(&ia,&ib,A,B,"rows");
```

MATLAB Syntax

```
c = intersect(a,b)
c = intersect(A,B,'rows')
[c,ia,ib] = intersect(...)
```

See Also

MATLAB `intersect` Calling Conventions

Purpose

Matrix inverse

C++ Prototype

```
mwArray inv(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray Y;              // Return value

Y = inv(X);
```

MATLAB Syntax

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

See Also

MATLAB `inv` Calling Conventions

invhilb

Purpose

Inverse of the Hilbert matrix

C++ Prototype

```
mwArray invhilb(const mwArray &n);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray n;              // Input argument(s)
mwArray H;              // Return value

H = invhilb(n);
```

MATLAB Syntax

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

See Also

MATLAB `invhilb` Calling Conventions

Purpose Inverse permute the dimensions of a multidimensional array

C++ Prototype

```
mwArray ipermute(const mwArray &B, const mwArray &order);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray B, order;           // Input argument(s)
mwArray A;                  // Return value

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

MATLAB Syntax

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

See Also MATLAB `ipermute` Calling Conventions

is*

Purpose

Detect state

C++ Prototype

```
mwArray iscell(const mwArray &C);
mwArray iscellstr(const mwArray &S);
mwArray ischar(const mwArray &A);
mwArray isempty(const mwArray &A);
mwArray isequal(const mwArray &in1,
                const mwVarargin &in2=mwVarargin::DIN,
                const mwArray &in3=mwArray::DIN,
                .
                .
                .
                const mwArray &in33=mwArray::DIN);
mwArray isfield(const mwArray &S,
                const mwArray &field=mwArray::DIN);
mwArray isfinite(const mwArray &A);
mwArray isinf(const mwArray &A);
mwArray isletter(const mwArray &A);
mwArray islogical(const mwArray &A);
mwArray isnan(const mwArray &A);
mwArray isnumeric(const mwArray &A);
mwArray isprime(const mwArray &A);
mwArray isreal(const mwArray &A);
mwArray isspace(const mwArray &str);
mwArray issparse(const mwArray &S);
mwArray isstruct(const mwArray &S);
mwArray isstudent();
mwArray isunix();
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, B, C, D, S;      // Input argument(s)
mwArray k, TF;              // Return value

k = iscell(C);                  k = islogical(A);
k = iscellstr(S);               TF = isnan(A);
k = ischar(S);                  k = isnumeric(A);
k = isempty(A);                 k = isobject(A);
k = isequal(A,B,C,D);           TF = isprime(A);
k = isfield(S,"field");         k = isreal(A);
TF = isfinite(A);               TF = isspace("str");
k = isglobal(NAME);             k = issparse(S);
TF = ishandle(H);               k = isstruct(S);
k = ishold();                   k = isstudent();
TF = isinf(A);                  k = isunix();
TF = isletter("str");
```

MATLAB Syntax

```
k = iscell(C)                   k = islogical(A)
k = iscellstr(S)                TF = isnan(A)
k = ischar(S)                   k = isnumeric(A)
k = isempty(A)                  k = isobject(A)
k = isequal(A,B,...)            TF = isprime(A)
k = isfield(S,'field')          k = isreal(A)
TF = isfinite(A)                TF = isspace('str')
k = isglobal(NAME)              k = issparse(S)
TF = ishandle(H)                k = isstruct(S)
k = ishold                      k = isstudent
TF = isinf(A)                   k = isunix
TF = isletter('str')
```

See Also

MATLAB is | Calling Conventions

isa

Purpose

Detect an object of a given class

C++ Prototype

```
mwArray isa(const mwArray &obj, const mwArray &classname);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray classname;          // String array(s)
mwArray obj;                // Input argument(s)

K = isa(obj,classname);
```

MATLAB Syntax

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

See Also

MATLAB `isa` Calling Conventions

Purpose

Matrix complexity

C++ Prototype

```
mwArray iscomplex(const mwArray &m);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray m;              // Input argument(s)
mwArray R;              // Return value

R = iscomplex(m);
```

MATLAB Syntax

```
iscomplex(m)
```

See Also

Calling Conventions

ismember

Purpose

Detect members of a set

C++ Prototype

```
mwArray ismember(const mwArray &a, const mwArray &S);
mwArray ismember(const mwArray &A, const mwArray &S,
                 const mwArray &flag);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray a, A, S;            // Input argument(s)
mwArray k;                  // Return value

k = ismember(a,S);
k = ismember(A,S,"rows");
```

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 `ischar` (`is*`) in MATLAB 5.

See Also

MATLAB `ischar` Calling Conventions

j

Purpose

Imaginary unit

C++ Prototype

```
mwArray j(void);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray R;              // Return value

R = j();
```

MATLAB Syntax

```
j
```

See Also

MATLAB `j` Calling Conventions

Purpose

Kronecker tensor product

C++ Prototype

```
mwArray kron(const mwArray &X, const mwArray &Y);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, Y;           // Input argument(s)
mwArray K;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `kron`

Calling Conventions

lcm

Purpose

Least common multiple

C++ Prototype

```
mwArray lcm(const mwArray &A, const mwArray &B);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, B;           // Input argument(s)
mwArray L;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `lcm` Calling Conventions

Purpose

Associated Legendre functions

C++ Prototype

```
mwArray legendre(const mwArray &n, const mwArray &X);
mwArray legendre(const mwArray &n, const mwArray &X,
                 const mwArray &sch);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray n, X;           // Input argument(s)
mwArray P, S;           // Return value

P = legendre(n,X);
S = legendre(n,X,"sch");
```

MATLAB Syntax

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

See Also

MATLAB `legendre` Calling Conventions

length

Purpose

Length of vector

C++ Prototype

```
mwArray length(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray n;              // Return value

n = length(X);
```

MATLAB Syntax

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

See Also

MATLAB `length` Calling Conventions

Purpose

Linear to mu-law conversion

C++ Prototype

```
mwArray lin2mu(const mwArray &y);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray y;                  // Input argument(s)
mwArray mu;                 // Return value

mu = lin2mu(y);
```

MATLAB Syntax

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

See Also

MATLAB `lin2mu` Calling Conventions

linspace

Purpose

Generate linearly spaced vectors

C++ Prototype

```
mwArray linspace(const mwArray &a, const mwArray &b);
mwArray linspace(const mwArray &a, const mwArray &b,
                 const mwArray &n);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray a, b, n;        // Input argument(s)
mwArray y;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `linspace` Calling Conventions

Purpose

Load up to 16 mwArray variables from disk

C++ Prototype

```
void load(const mwArray &file,
          const char* name1, mwArray *var1,
          const char* name2=NULL, mwArray *var2=NULL,
          .
          .
          .
          const char* name16=NULL, mwArray *var16=NULL);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray file;           // String array(s);
mwArray x, y, z;        // Input argument(s)

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

MATLAB Syntax

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

See Also

MATLAB load Calling Conventions

log

Purpose

Natural logarithm

C++ Prototype

```
mwArray log(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray Y;              // Return value

Y = log(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

```
mwArray log2(const mwArray &X);
mwArray log2(mwArray *E, const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray E;              // Output argument(s)
mwArray Y, F;           // Return value

Y = log2(X);
F = log2(&E,X);
```

MATLAB Syntax

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

See Also

MATLAB `log2` Calling Conventions

log10

Purpose

Common (base 10) logarithm

C++ Prototype

```
mwArray log10(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray Y;              // Return value

Y = log10(X);
```

MATLAB Syntax

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

See Also

MATLAB `log10` Calling Conventions

Purpose

Convert numeric values to logical

C++ Prototype

```
mwArray logical(const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Input argument(s)
mwArray K;              // Return value

K = logical(A);
```

MATLAB Syntax

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

See Also

MATLAB `logical` Calling Conventions

logm

Purpose

Matrix logarithm

C++ Prototype

```
mwArray logm(const mwArray &X);
mwArray logm(mwArray *esterr, const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;                  // Input argument(s)
mwArray esterr;             // Output argument(s)
mwArray Y;                  // Return value

Y = logm(X);
Y = logm(&esterr,X);
```

MATLAB Syntax

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

See Also

MATLAB `logm` Calling Conventions

Purpose

Generate logarithmically spaced vectors

C++ Prototype

```
mwArray logspace(const mwArray &a, const mwArray &b);
mwArray logspace(const mwArray &a, const mwArray &b,
                 const mwArray &n);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray a, b, n;            // Input argument(s)
mwArray y;                  // Return value

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

MATLAB Syntax

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

See Also

MATLAB `logspace` Calling Conventions

lower

Purpose

Convert string to lower case

C++ Prototype

```
mwArray lower(const mwArray &str);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray str;                // String array(s)
mwArray t;                  // Return value

t = lower(str);
```

MATLAB Syntax

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

See Also

MATLAB `lower` Calling Conventions

Purpose

Least squares solution in the presence of known covariance

C++ Prototype

```
mwArray lscov(const mwArray &A, const mwArray &b,
              const mwArray &V);
mwArray lscov(mwArray *dx, const mwArray &A, const mwArray &b,
              const mwArray &V);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, b, V;        // Input argument(s)
mwArray dx;             // Output argument(s)
mwArray x;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `lscov` Calling Conventions

lsqnonneg

Purpose Linear least squares with nonnegativity constraints

C++ Prototype

```
mwArray lsqnonneg(const mwArray &in1,
                  const mwArray &in2=mwArray::DIN,
                  const mwArray &in3=mwArray::DIN,
                  const mwArray &in4=mwArray::DIN)

mwArray lsqnonneg(mwArray *out1,
                  const mwArray &in1,
                  const mwArray &in2=mwArray::DIN,
                  const mwArray &in3=mwArray::DIN,
                  const mwArray &in4=mwArray::DIN)

mwArray lsqnonneg(mwArray *out1,
                  mwArray *out2,
                  const mwArray &in1,
                  const mwArray &in2=mwArray::DIN,
                  const mwArray &in3=mwArray::DIN,
                  const mwArray &in4=mwArray::DIN)

mwArray lsqnonneg(mwArray *out1,
                  mwArray *out2,
                  mwArray *out3,
                  const mwArray &in1,
                  const mwArray &in2=mwArray::DIN,
                  const mwArray &in3=mwArray::DIN,
                  const mwArray &in4=mwArray::DIN)

mwArray lsqnonneg(mwArray *out1,
                  mwArray *out2,
                  mwArray *out3,
                  mwArray *out4,
                  const mwArray &in1,
                  const mwArray &in2=mwArray::DIN,
                  const mwArray &in3=mwArray::DIN,
                  const mwArray &in4=mwArray::DIN)

mwArray lsqnonneg(mwArray *out1,
                  mwArray *out2,
```

```
mwArray *out3,
mwArray *out4,
mwArray *out5,
const mwArray &in1,
const mwArray &in2=mwArray::DIN,
const mwArray &in3=mwArray::DIN,
const mwArray &in4=mwArray::DIN)
```

C++ Syntax

```
#include "matlab.hpp"

mwArray C, *d;          // Input argument(s)
mwArray x0, *options;   // Input argument(s)
mwArray resnorm;        // Output argument(s)
mwArray residual;       // Output argument(s)
mwArray exitflag;       // Output argument(s)
mwArray output;         // Output argument(s)
mwArray lambda;         // Output argument(s)
mwArray x;              // Return value

/* MATLAB syntax: x = lsqnonneg(C,d) */
x = lsqnonneg(C,d);

/* MATLAB syntax: x = lsqnonneg(C,d,x0) */
x = lsqnonneg(C,d,x0);

/* MATLAB syntax: x = lsqnonneg(C,d,x0,options) */
x = lsqnonneg(C,d,x0,options);

/* MATLAB syntax: [x, resnorm ] = lsqnonneg(...) */
x = lsqnonneg(&resnorm,C,d);
x = lsqnonneg(&resnorm,C,d,x0);
x = lsqnonneg(&resnorm,C,d,x0,options);

/* MATLAB syntax: [x, resnorm, residual ] = lsqnonneg(...) */
x = lsqnonneg(&resnorm,&residual,C,d);
x = lsqnonneg(&resnorm,&residual,C,d,x0);
x = lsqnonneg(&resnorm,&residual,C,d,x0,options);

/* MATLAB syntax: [x,resnorm,residual,exitflag] = lsqnonneg(...) */
x = lsqnonneg(&resnorm,&residual,&exitflag,C,d);
x = lsqnonneg(&resnorm,&residual,&exitflag,C,d,x0);
x = lsqnonneg(&resnorm,&residual,&exitflag,C,d,x0,options);

/* MATLAB: [x,resnorm,residual,exitflag,output] = lsqnonneg(...) *
x = lsqnonneg(&resnorm,&residual,&exitflag,&output,C,d);
```

```
x = lsqnonneg(&resnorm,&residual,&exitflag,&output,C,d,x0);
x = lsqnonneg(&resnorm,&residual,&exitflag,&output,C,d,x0,options);

/* [x,resnorm,residual,exitflag,output,lambda] = lsqnonneg(...) */
x = lsqnonneg(&resnorm,&residual,&exitflag,&output,&lambda,C,d);
x = lsqnonneg(&resnorm,&residual,&exitflag,&output,&lambda,C,d,x0);
x = lsqnonneg(&resnorm,&residual,&exitflag,&output,&lambda,C,d,x0,
              options);
```

MATLAB Syntax

```
x = lsqnonneg(C,d)
x = lsqnonneg(C,d,x0)
x = lsqnonneg(C,d,x0,options)
[x,resnorm] = lsqnonneg(...)
[x,resnorm,residual] = lsqnonneg(...)
[x,resnorm,residual,exitflag] = lsqnonneg(...)
[x,resnorm,residual,exitflag,output] = lsqnonneg(...)
[x,resnorm,residual,exitflag,output,lambda] = lsqnonneg(...)
```

See Also MATLAB `lsqnonneg` Calling Conventions

Purpose

LU matrix factorization

C++ Prototype

```
mwArray lu(const mwArray &X,
           const mwArray &thresh=mwArray::DIN);

mwArray lu(mwArray *U,
           const mwArray &X,
           const mwArray &thresh=mwArray::DIN);

mwArray lu(mwArray *U,
           mwArray *P,
           const mwArray &X,
           const mwArray &thresh=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, thresh;          // Input argument(s)
mwArray U, P;               // Output argument(s)
mwArray L;                  // Return value

L = lu(&U,X);
L = lu(&U,&P,X);
L = lu(X);
L = lu(X,thresh);
```

MATLAB Syntax

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

See Also

MATLAB `lu` Calling Conventions

luinc

Purpose

Incomplete LU matrix factorizations

C++ Prototype

```
mwArray luinc(const mwArray &X, const mwArray &droptol=mwArray::DIN)
mwArray luinc(mwArray *U,
              const mwArray &X,
              const mwArray &droptol=mwArray::DIN)
mwArray luinc(mwArray *U,
              mwArray *P,
              const mwArray &X,
              const mwArray &droptol=mwArray::DIN)
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, droptol, options;    // Input argument(s)
mwArray U, P;                   // Output argument(s)
mwArray L;                      // Return value

L = luinc(X,"0");
L = luinc(&U,X,"0");
L = luinc(&U,&P,X,"0");
L = luinc(X,droptol);
L = luinc(X,options);
L = luinc(&U,X,options);
L = luinc(&U,X,droptol);
L = luinc(&U,&P,X,options);
L = luinc(&U,&P,X,droptol);
```

MATLAB Syntax

```
luinc(X,'0')
[L,U] = luinc(X,'0')
[L,U,P] = luinc(X,'0')
luinc(X,droptol)
luinc(X,options)
[L,U] = luinc(X,options)
[L,U] = luinc(X,droptol)
[L,U,P] = luinc(X,options)
[L,U,P] = luinc(X,droptol)
```

See Also MATLAB `luinc` Calling Conventions

magic

Purpose

Magic square

C++ Prototype

```
mwArray magic(const mwArray &n);
mwArray magic();
```

C++ Syntax

```
#include "matlab.hpp"

mwArray n;              // Input argument(s)
mwArray M;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `magic` Calling Conventions

Purpose

Convert a matrix into a string

C++ Prototype

```
mwArray mat2str(const mwArray &A);
mwArray mat2str(const mwArray &A, const mwArray &n);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, n;           // Input argument(s)
mwArray str;            // Return value

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

MATLAB Syntax

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

See Also

MATLAB `mat2str` Calling Conventions

max

Purpose

Maximum elements of an array

C++ Prototype

```
mwArray max(const mwArray &A);
mwArray max(const mwArray &A, const mwArray &B);
mwArray max(const mwArray &A, const mwArray &B,
            const mwArray &dim);
mwArray max(mwArray *I, const mwArray &A);
mwArray max(mwArray *I, const mwArray &A, const mwArray &B);
mwArray max(mwArray *I, const mwArray &A, const mwArray &mtrx,
            const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, B, dim;      // Input argument(s)
mwArray I;              // Output argument(s)
mwArray C;              // Return value

C = max(A);
C = max(A,B);
C = max(A,empty(),dim);
C = max(&I,A);
C = max(&I,A,empty(),dim);
```

MATLAB Syntax

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

See Also

MATLAB `max` Calling Conventions

Purpose

Average or mean value of arrays

C++ Prototype

```
mwArray mean(const mwArray &A);
mwArray mean(const mwArray &A, const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, dim;         // Input argument(s)
mwArray M;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `mean` Calling Conventions

median

Purpose

Median value of arrays

C++ Prototype

```
mwArray median(const mwArray &A);
mwArray median(const mwArray &A, const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, dim;         // Input argument(s)
mwArray M;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `median` Calling Conventions

Purpose

Generate X and Y matrices for three-dimensional plots

C++ Prototype

```
mwArray meshgrid(mwArray *Y, const mwArray &x, const mwArray &y);
mwArray meshgrid(mwArray *Y, const mwArray &x);
mwArray meshgrid(mwArray *Y, mwArray *Z, const mwArray &x,
                 const mwArray &y, const mwArray &z);
mwArray meshgrid(const mwArray &x);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray x, y, z;            // Input argument(s)
mwArray Y, Z;               // Output argument(s)
mwArray X;                  // Return value

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

MATLAB Syntax

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

See Also

MATLAB `meshgrid` Calling Conventions

mfilename

Purpose

The name of the currently running M-file

C++ Prototype

```
mwArray mfilename();
```

C++ Syntax

```
#include "matlab.hpp"

mwArray R;              // Return value

R = mfilename();
```

MATLAB Syntax

```
mfilename
```

See Also

MATLAB `mfilename` Calling Conventions

Purpose

Minimum elements of an array

C++ Prototype

```
mwArray min(const mwArray &A);
mwArray min(const mwArray &A, const mwArray &B);
mwArray min(const mwArray &A, const mwArray &B,
            const mwArray &dim);
mwArray min(mwArray *I, const mwArray &A);
mwArray min(mwArray *I, const mwArray &A, const mwArray &B);
mwArray min(mwArray *I, const mwArray &A, const mwArray &B,
            const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, B, dim;          // Input argument(s)
mwArray I;                  // Output argument(s)
mwArray C;                  // Return value

C = min(A);
C = min(A,B);
C = min(A,empty(),dim);
C = min(&I,A);
C = min(&I,A,empty(),dim);
```

MATLAB Syntax

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

See Also

MATLAB `min` Calling Conventions

mod

Purpose

Modulus (signed remainder after division)

C++ Prototype

```
mwArray mod(const mwArray &X, const mwArray &Y);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, Y;               // Input argument(s)
mwArray M;                  // Return value

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

MATLAB Syntax

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

See Also

MATLAB `mod` Calling Conventions

Purpose

Mu-law to linear conversion

C++ Prototype

```
mwArray mu2lin(const mwArray &y);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray mu;                 // Input argument(s)
mwArray y;                  // Return value

y = mu2lin(mu);
```

MATLAB Syntax

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

See Also

MATLAB `mu2lin` Calling Conventions

nan

Purpose

Not-a-Number

C++ Prototype

```
mwArray nan();
```

C++ Syntax

```
#include "matlab.hpp"

mwArray R;              // Return value

R = nan();
```

MATLAB Syntax

```
NaN
```

See Also

MATLAB `NaN` Calling Conventions

Description

`NaN` returns the IEEE arithmetic representation for Not-a-Number (`NaN`). These result from operations which have undefined numerical results.

Purpose Check number of input arguments

C++ Prototype

```
mwArray nargchk(const mwArray &low, const mwArray &high,
                const mwArray &number);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray low, high, number;  // Input argument(s)
mwArray msg;                // Return value

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

MATLAB Syntax

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

See Also MATLAB `nargchk` Calling Conventions

nchoosek

Purpose

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

C++ Prototype

```
mwArray nchoosek(const mwArray &v, const mwArray &k);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray v, k;               // Input argument(s)
mwArray C;                  // Return value

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

MATLAB Syntax

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

See Also

MATLAB nchoosek Calling Conventions

Purpose Number of array dimensions

C++ Prototype

```
mwArray ndims(const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Input argument(s)
mwArray n;              // Return value

n = ndims(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

nextpow2

Purpose

Next power of two

C++ Prototype

```
mwArray nextpow2(const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Input argument(s)
mwArray p;              // Return value

p = nextpow2(A);
```

MATLAB Syntax

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

See Also

MATLAB `nextpow2` Calling Conventions

Purpose

Nonnegative least squares

Note The `nnls` routine was replaced by `lsqnonneg` in Release 11 (MATLAB 5.3). In Release 12 (MATLAB 6.0), `nnls` displays a warning and calls `lsqnonneg`.

C++ Prototype

```
mwArray nnls(const mwArray &A, const mwArray &b);
mwArray nnls(const mwArray &A, const mwArray &b,
             const mwArray &tol);
mwArray nnls(mwArray *w, const mwArray &A, const mwArray &b);
mwArray nnls(mwArray *w, const mwArray &A, const mwArray &b,
             const mwArray &tol);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, b, tol;      // Input argument(s)
mwArray w;              // Output argument(s)
mwArray x;              // Return value

x = nnls(A,b);
x = nnls(A,b,tol);
x = nnls(&w,A,b);
x = nnls(&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

nnz

Purpose

Number of nonzero matrix elements

C++ Prototype

```
mwArray nnz(const mwArray &X)
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray n;              // Return value

n = nnz(X);
```

MATLAB Syntax

```
n = nnz(X)
```

See Also

MATLAB `nnz`

Calling Conventions

Purpose

Nonzero matrix elements

C++ Prototype

```
mwArray nonzeros(const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Input argument(s)
mwArray s;              // Return value

s = nonzeros(A);
```

MATLAB Syntax

```
s = nonzeros(A)
```

See Also

MATLAB `nonzeros` Calling Conventions

norm

Purpose

Vector and matrix norms

C++ Prototype

```
mwArray norm(const mwArray &A);
mwArray norm(const mwArray &A, const mwArray &p);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, p;           // Input argument(s)
mwArray n;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `norm`

Calling Conventions

Purpose

2-norm estimate

C++ Prototype

```
mwArray normest(const mwArray &S);
mwArray normest(const mwArray &S, const mwArray &tol);
mwArray normest(mwArray *count, const mwArray &S);
mwArray normest(mwArray *count, const mwArray &S,
                const mwArray &tol);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray S, tol;             // Input argument(s)
mwArray count;              // Output argument(s)
mwArray nrm;                // Return value

nrm = normest(S);
nrm = normest(S,tol);
nrm = normest(&count,S);
nrm = normest(&count,S,tol);
```

MATLAB Syntax

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

See Also

MATLAB `normest` Calling Conventions

now

Purpose

Current date and time

C++ Prototype

```
mwArray now();
```

C++ Syntax

```
#include "matlab.hpp"

mwArray t;              // Return value

t = now();
```

MATLAB Syntax

```
t = now
```

See Also

MATLAB `now`

Calling Conventions

Purpose

Null space of a matrix

C++ Prototype

```
mwArray null(const mwArray &A);
mwArray null(const mwArray &A, const mwArray &basis);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, basis;       // Input argument(s)
mwArray B;              // Return value

B = null(A);
B = null(A,"ortho");
B = null(A,"rational");
```

MATLAB Syntax

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

See Also

MATLAB `null`

Calling Conventions

num2cell

Purpose

Convert a numeric array into a cell array

C++ Prototype

```
mwArray num2cell(const mwArray &A,
                 const mwArray &dims=mwArray::DIN)
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, dims;        // Input argument(s)
mwArray c;              // Return value

c = num2cell(A);
c = num2cell(A,dims);
```

MATLAB Syntax

```
c = num2cell(A)
c = num2cell(A,dims)
```

See Also

MATLAB `num2cell` Calling Conventions

Purpose

Number to string conversion

C++ Prototype

```
mwArray num2str(const mwArray &A);
mwArray num2str(const mwArray &A, const mwArray &precision);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray format;             // String array(s)
mwArray A, precision;       // Input argument(s)
mwArray str;                // Return value

str = num2str(A);
str = num2str(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

nzmax

Purpose

Amount of storage allocated for nonzero matrix elements

C++ Prototype

```
mwArray nzmax(const mwArray &S)
```

C++ Syntax

```
#include "matlab.hpp"

mwArray S;              // Input argument(s)
mwArray n;              // Return value

n = nzmax(S);
```

MATLAB Syntax

```
n = nzmax(S)
```

See Also

MATLAB `nzmax` Calling Conventions

Purpose

Solve differential equations

C++ Prototype

```
mwArray solver(mwArray *Y, const mwArray &F,
                const mwArray &tspan,
                const mwArray &y0);
mwArray solver(mwArray *Y, const mwArray &F,
                const mwArray &tspan,
                const mwArray &y0,
                const mwArray &options);
mwArray solver(mwArray *Y, const mwArray &F,
                const mwArray &tspan,
                const mwArray &y0,
                const mwArray &options,
                const mwArray &p);
mwArray solver(mwArray *Y,
                mwArray *TE,
                mwArray *YE,
                mwArray *IE,
                mwArray *O6,
                const mwArray &F,
                const mwArray &tspan,
                const mwArray &y0,
                const mwArray &options);
mwArray solver(mwArray *Y,
                mwArray *TE,
                mwArray *YE,
                mwArray *IE,
                mwArray *O6,
                const mwArray &F,
                const mwArray &tspan,
                const mwArray &y0,
                const mwArray &options,
                const mwArray &p);
mwArray solver(const mwArray &F,
                const mwArray &interval,
                const mwArray &y0);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray F, model;               // String array(s)
mwArray tspan, y0, interval;    // Input argument(s)
mwArray options, p1, p2;        // Input argument(s)
mwArray Y, TE, YE, IE, O6;      // Output argument(s)
mwArray T, sol;                 // Return value

T = solver(&Y,F,tspan,y0);
T = solver(&Y,F,tspan,y0,options);
T = solver(&Y,F,tspan,y0,options,p);
T = solver(&Y,&TE,&YE,&IE,&O6,F,tspan,y0,options);
T = solver(&Y,&TE,&YE,&IE,,&O6,F,tspan,y0,options,p);
sol = solver(F,interval,y0);
```

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,...)
sol = solver('F',interval,y0)
```

See Also

MATLAB ode45, ode23, ode113, ode15s, ode23sCalling Conventions

Purpose

Extract properties from options structure created with odeset

C++ Prototype

```
mwArray odeget(const mwArray &options,
               const mwArray &name=mwArray::DIN,
               const mwArray &default=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray name;               // String array(s)
mwArray options, default;   // Input argument(s)
mwArray o;                  // Return value

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

MATLAB Syntax

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

See Also

MATLAB odeget Calling Conventions

odeset

Purpose

Create or alter options structure for input to ODE solvers

C++ Prototype

```
mwArray odeset(const mwVarargin &in1,
               const mwArray &in2=mwArray::DIN,
               .
               .
               .
               const mwArray &in32=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray name, name1, name2;     // Input argument(s)
mwArray value, value1, value2;  // Input argument(s)
mwArray oldopts, newopts;       // Input argument(s)
mwArray options;                // Return value

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

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

```
mwArray ones(const mwVarargin &in1=mwVarargin::DIN,
             const mwArray &in2=mwArray::DIN,
             .
             .
             .
             const mwArray &in32=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray m, n, A;            // Input argument(s)
mwArray d1, d2, d3;         // Input argument(s)
mwArray Y;                  // Return value

Y = ones(n);
Y = ones(m,n);
Y = ones(horzcat(m,n));
Y = ones(d1,d2,d3,...);
Y = ones(horzcat(d1,d2,d3,...));
Y = ones(size(A));
```

MATLAB Syntax

```
Y = ones(n)
Y = ones(m,n)
Y = ones([m n])
Y = ones(d1,d2,d3,...)
Y = ones([d1 d2 d3...])
Y = ones(size(A))
```

See Also

MATLAB ones Calling Conventions

optimget

Purpose

Get optimization options structure parameter values

C++ Prototype

```
mwArray optimget(const mwArray &in1,
                 const mwArray &in2=mwArray::DIN,
                 const mwArray &in3=mwArray::DIN,
                 const mwArray &in4=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray options, default;   // Input argument(s)
mwArray param;              // String array(s)
mwArray val;                // Return value

val = optimget(options,param);
val = optimget(options,param,default);
```

MATLAB Syntax

```
val = optimget(options,'param')
val = optimget(options,'param',default)
```

See Also

MATLAB `optimget` Calling Conventions

Purpose

Create or edit optimization options parameter structure

C++ Prototype

```
mwArray optimset(const mwVarargin &in1,
                 const mwArray &in2=mwArray::DIN,
                 .
                 .
                 .
                 const mwArray &in32=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray value1, value2;             // Input argument(s)
mwArray param1, param2;             // String array(s)
mwArray optimfun, oldopts, newopts; // Input argument(s)
mwArray options;                    // Return value

options = optimset(param1,value1,param2,value2,...);
optimset();
options = optimset();
options = optimset(optimfun);
options = optimset(oldopts,param1,value1,...);
options = optimset(oldopts,newopts);
```

MATLAB Syntax

```
options = optimset('param1',value1,'param2',value2,...)
optimset
options = optimset
options = optimset(optimfun)
options = optimset(oldopts,'param1',value1,...)
options = optimset(oldopts,newopts)
```

See Also

MATLAB `optimset` Calling Conventions

orth

Purpose

Range space of a matrix

C++ Prototype

```
mwArray orth(const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Input argument(s)
mwArray B;              // Return value

B = orth(A);
```

MATLAB Syntax

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

See Also

MATLAB `orth` Calling Conventions

Purpose

Pascal matrix

C++ Prototype

```
mwArray pascal(const mwArray &n);
mwArray pascal(const mwArray &n, const mwArray &k);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray n;                  // Input argument(s)
mwArray A;                  // Return value

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

MATLAB Syntax

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

See Also

MATLAB `pascal` Calling Conventions

pcg

Purpose

Preconditioned Conjugate Gradients method

C++ Prototype

```
mwArray pcg(const mwArray &in1,
            const mwArray &in2=mwArray::DIN,
            const mwArray &in3=mwArray::DIN,
            const mwArray &in4=mwArray::DIN,
            const mwArray &in5=mwArray::DIN,
            const mwArray &in6=mwArray::DIN,
            const mwArray &in7=mwArray::DIN,
            const mwVarargin &in8=mwVarargin::DIN,
            const mwArray &in9=mwArray::DIN,
            .
            .
            .
            const mwArray &in39=mwArray::DIN);

mwArray pcg(mwArray *out1, mwArray *out2,
            mwArray *out3, mwArray *out4,
            const mwArray &in1,
            const mwArray &in2=mwArray::DIN,
            const mwArray &in3=mwArray::DIN,
            const mwArray &in4=mwArray::DIN,
            const mwArray &in5=mwArray::DIN,
            const mwArray &in6=mwArray::DIN,
            const mwArray &in7=mwArray::DIN,
            const mwVarargin &in8=mwVarargin::DIN,
            const mwArray &in9=mwArray::DIN,
            .
            .
            .
            const mwArray &in39=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, b, tol, maxit, M, M1, M2, x0;// Input argument(s)
mwArray flag, relres, iter, resvec;     // Output argument(s)
mwArray x;                              // Return value

x = pcg(A,b);
x = pcg(A,b,tol);
x = pcg(A,b,tol,maxit);
x = pcg(A,b,tol,maxit,M);
x = pcg(A,b,tol,maxit,M1,M2);
x = pcg(A,b,tol,maxit,M1,M2,x0);
x = pcg(A,b,tol,maxit,M1,M2,x0);
x = pcg(&flag,A,b,tol,maxit,M1,M2,x0);
x = pcg(&flag,&relres,A,b,tol,maxit,M1,M2,x0);
x = pcg(&flag,&relres,&iter,A,b,tol,maxit,M1,M2,x0);
x = pcg(&flag,&relres,&iter,&resvec,A,b,tol,maxit,M1,M2,x0);
```

MATLAB Syntax

```
x = pcg(A,b)
pcg(A,b,tol)
pcg(A,b,tol,maxit)
pcg(A,b,tol,maxit,M)
pcg(A,b,tol,maxit,M1,M2)
pcg(A,b,tol,maxit,M1,M2,x0)
x = pcg(A,b,tol,maxit,M1,M2,x0)
[x,flag] = pcg(A,b,tol,maxit,M1,M2,x0)
[x,flag,relres] = pcg(A,b,tol,maxit,M1,M2,x0)
[x,flag,relres,iter] = pcg(A,b,tol,maxit,M1,M2,x0)
[x,flag,relres,iter,resvec] = pcg(A,b,tol,maxit,M1,M2,x0)
```

See Also

MATLAB pcg Calling Conventions

pchip

Purpose

Piecewise Cubic Hermite Interpolating Polynomial (PCHIP)

C++ Prototype

```
mwArray pchip(const mwArray &x,const mwArray &y);
mwArray pchip(const mwArray &x,const mwArray &y,const mwArray &xi );
```

C++ Syntax

```
#include "matlab.hpp"

mwArray x, y, x1;       // Input argument(s)
mwArray y1, pp;         // Output argument(s)

pp = pchip(x,y);
yi = pchip(x,y,xi);
```

MATLAB Syntax

```
pp = pchip(x,y);
yi = pchip(x,y,xi);
```

See Also

MATLAB `pchip` Calling Conventions

Purpose

All possible permutations

C++ Prototype

```
mwArray perms(const mwArray &v);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray v;              // Input argument(s)
mwArray P;              // Output argument(s)

P = perms(v);
```

MATLAB Syntax

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

See Also

MATLAB `perms` Calling Conventions

permute

Purpose

Rearrange the dimensions of a multidimensional array

C++ Prototype

```
mwArray permute(const mwArray &A, const mwArray &order);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, order;           // Input argument(s)
mwArray B;                  // Return value

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

MATLAB Syntax

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

See Also

MATLAB `permute` Calling Conventions

Purpose

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

C++ Prototype

```
mwArray pi();
```

C++ Syntax

```
#include "matlab.hpp"

mwArray R;              // Return value

R = pi();
```

MATLAB Syntax

```
pi
```

See Also

MATLAB `pi` Calling Conventions

pinv

Purpose

Moore-Penrose pseudoinverse of a matrix

C++ Prototype

```
mwArray pinv(const mwArray &A);
mwArray pinv(const mwArray &A, const mwArray &tol);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, tol;         // Input argument(s)
mwArray B;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `pinv` Calling Conventions

Purpose

Given's plane rotation

C++ Prototype

```
mwArray planerot(mwArray *y, const mwArray &x);
mwArray planerot(const mwArray &x);
```

C++ Syntax

```
#include "matlab.h"

mxArray x;                  // Input argument(s)
mxArray y;                  // Output argument(s)
mxArray g;                  // Return value

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

MATLAB Syntax

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

See Also

Calling Conventions

pol2cart

Purpose

Transform polar or cylindrical coordinates to Cartesian

C++ Prototype

```
mwArray pol2cart(mwArray *Y, const mwArray &THETA,
                 const mwArray &RHO);
mwArray pol2cart(mwArray *Y, mwArray *Z_out,
                 const mwArray &THETA, const mwArray &RHO,
                 const mwArray &Z_in);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray THETA, RHO, Z_in;   // Input argument(s)
mwArray Y, Z_out;           // Output argument(s)
mwArray X;                  // Return value

X = pol2cart(&Y,THETA,RHO);
X = pol2cart(&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

Purpose Polynomial with specified roots

C++ Prototype

```
mwArray poly(const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

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

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

MATLAB Syntax

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

See Also MATLAB `poly` Calling Conventions

polyarea

Purpose

Area of a polygon

C++ Prototype

```
mwArray polyarea(const mwArray &X, const mwArray &Y);
mwArray polyarea(const mwArray &X, const mwArray &Y,
                 const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, Y, dim;          // Input argument(s)
mwArray A;                  // Return value

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

MATLAB Syntax

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

See Also

MATLAB `polyarea` Calling Conventions

Purpose

Polynomial derivative

C++ Prototype

```
mwArray polyder(const mwArray &p);
mwArray polyder(const mwArray &a, const mwArray &b);
mwArray polyder(mwArray *d, const mwArray &b, const mwArray &a);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray p, a, b;        // Input argument(s)
mwArray d;              // Output argument(s)
mwArray k, q;           // Return value

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

MATLAB Syntax

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

See Also

MATLAB `polyder` Calling Conventions

polyeig

Purpose

Polynomial eigenvalue problem

C++ Prototype

```
mwArray polyeig(mwArray *out1,
                const mwVarargin &in1,
                const mwArray &in2=mwArray::DIN,
                .
                .
                .
                const mwArray &in32=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A0, A1;             // Input argument(s)
mwArray e;                  // Output argument(s)
mwArray X;                  // Return value

X = polyeig(&e,A0,A1,...Ap);
e = polyeig(A0,A1,...Ap);
```

MATLAB Syntax

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

See Also

MATLAB `polyeig` Calling Conventions

Purpose

Polynomial curve fitting

C++ Prototype

```
mwArray polyfit(const mwArray &x, const mwArray &y,
                const mwArray &n);

mwArray polyfit(mwArray *s, const mwArray &x, const mwArray &y,
                const mwArray &n);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray x, y, n;            // Input argument(s)
mwArray s;                  // Output argument(s)
mwArray p;                  // Return value

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

MATLAB Syntax

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

See Also

MATLAB `polyfit` Calling Conventions

polyval

Purpose

Polynomial evaluation

C++ Prototype

```
mwArray polyval(const mwArray &p, const mwArray &x);

mwArray polyval(const mwArray &p, const mwArray &x,
                const mwArray &S);

mwArray polyval(mwArray *delta, const mwArray &p, const mwArray &x,
                const mwArray &S);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray p, x, S;            // Input argument(s)
mwArray delta;              // Output argument(s)
mwArray y;                  // Return value

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

MATLAB Syntax

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

See Also

MATLAB `polyval` Calling Conventions

Purpose

Matrix polynomial evaluation

C++ Prototype

```
mwArray polyvalm(const mwArray &p, const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray p, X;           // Input argument(s)
mwArray Y;              // Return value

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

MATLAB Syntax

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

See Also

MATLAB `polyvalm` Calling Conventions

pow2

Purpose

Base 2 power and scale floating-point numbers

C++ Prototype

```
mwArray pow2(const mwArray &Y);
mwArray pow2(const mwArray &F, const mwArray &E);
```

C++ Syntax

```
#include "matlab.hpp"

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

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

MATLAB Syntax

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

See Also

MATLAB pow2 Calling Conventions

Purpose

Generate list of prime numbers

C++ Prototype

```
mwArray primes(const mwArray &n);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray n;              // Input argument(s)
mwArray p;              // Return value

p = primes(n);
```

MATLAB Syntax

```
p = primes(n)
```

See Also

MATLAB `primes` Calling Conventions

prod

Purpose

Product of array elements

C++ Prototype

```
mwArray prod(const mwArray &A);
mwArray prod(const mwArray &A, const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, dim;             // Input argument(s)
mwArray B;                  // Return value

B = prod(A);
B = prod(A,dim);
```

MATLAB Syntax

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

See Also

MATLAB `prod` Calling Conventions

Purpose

Quasi-Minimal Residual method

C++ Prototype

```
mwArray qmr(const mwArray &in1,
            const mwArray &in2=mwArray::DIN,
            const mwArray &in3=mwArray::DIN,
            const mwArray &in4=mwArray::DIN,
            const mwArray &in5=mwArray::DIN,
            const mwArray &in6=mwArray::DIN,
            const mwArray &in7=mwArray::DIN,
            const mwVarargin &in8=mwVarargin::DIN,
            const mwArray &in9=mwArray::DIN,
            .
            .
            .
            const mwArray &in39=mwArray::DIN);

mwArray qmr(mwArray *out1, mwArray *out2,
            mwArray *out3, mwArray *out4,
            const mwArray &in1,
            const mwArray &in2=mwArray::DIN,
            const mwArray &in3=mwArray::DIN,
            const mwArray &in4=mwArray::DIN,
            const mwArray &in5=mwArray::DIN,
            const mwArray &in6=mwArray::DIN,
            const mwArray &in7=mwArray::DIN,
            const mwVarargin &in8=mwVarargin::DIN,
            const mwArray &in9=mwArray::DIN,
            .
            .
            .
            const mwArray &in39=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, b, tol, maxit, M, M1, M2, x0;// Input argument(s)
mwArray flag, relres, iter, resvec;     // Output argument(s)
mwArray x;                              // Return value

x = qmr(A,b);
qmr(A,b,tol);
qmr(A,b,tol,maxit);
qmr(A,b,tol,maxit,M);
qmr(A,b,tol,maxit,M1,M2);
qmr(A,b,tol,maxit,M1,M2,x0);
x = qmr(A,b,tol,maxit,M1,M2,x0);
x = qmr(&flag,A,b,tol,maxit,M1,M2,x0);
x = qmr(&flag,&relres,A,b,tol,maxit,M1,M2,x0);
x = qmr(&flag,&relres,&iter,A,b,tol,maxit,M1,M2,x0);
x = qmr(&flag,&relres,&iter,&resvec,A,b,tol,maxit,M1,M2,x0);
```

MATLAB Syntax

```
x = qmr(A,b)
qmr(A,b,tol)
qmr(A,b,tol,maxit)
qmr(A,b,tol,maxit,M1)
qmr(A,b,tol,maxit,M1,M2)
qmr(A,b,tol,maxit,M1,M2,x0)
x = qmr(A,b,tol,maxit,M1,M2,x0)
[x,flag] = qmr(A,b,tol,maxit,M1,M2,x0)
[x,flag,relres] = qmr(A,b,tol,maxit,M1,M2,x0)
[x,flag,relres,iter] = qmr(A,b,tol,maxit,M1,M2,x0)
[x,flag,relres,iter,resvec] = qmr(A,b,tol,maxit,M1,M2,x0)
```

Purpose

Orthogonal-triangular decomposition

C++ Prototype

```
mwArray qr(const mwArray &in1,
           const mwArray &in2=mwArray::DIN,
           const mwArray &in3=mwArray::DIN);

mwArray qr(mwArray *out1,
           const mwArray &in1,
           const mwArray &in2=mwArray::DIN,
           const mwArray &in3=mwArray::DIN);

mwArray qr(mwArray *out1,
           mwArray *out2,
           const mwArray &in1,
           const mwArray &in2=mwArray::DIN,
           const mwArray &in3=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray R, E;           // Output argument(s)
mwArray Q, A;           // Return value

Q = qr(&R,X);
Q = qr(&R,&E,X);
Q = qr(&R,X,0);
Q = qr(&R,&E,X,0);
A = qr(X);
```

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

qrdelete

Purpose

Delete column from QR factorization

C++ Prototype

```
mwArray qrdelete(mwArray *R_out, const mwArray &Q_in,
                 const mwArray &R_in, const mwArray &j);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray Q_in, R_in, j;      // Input argument(s)
mwArray R_out;              // Output argument(s)
mwArray Q;                  // Return value

Q = qrdelete(&R_out,Q_in,R_in,j);
```

MATLAB Syntax

```
[Q,R] = qrdelete(Q,R,j)
```

See Also

MATLAB `qrdelete` Calling Conventions

Purpose

Insert column in QR factorization

C++ Prototype

```
mwArray qrinsert(mwArray *R_out, const mwArray &Q_in,
                 const mwArray &R_in, const mwArray &j,
                 const mwArray &x);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray Q_in, R_in, j, x;   // Input argument(s)
mwArray R_out;              // Output argument(s)
mwArray Q;                  // Return value

Q = qrinsert(&R_out,Q_in,R_in,j,x);
```

MATLAB Syntax

```
[Q,R] = qrinsert(Q,R,j,x)
```

See Also

MATLAB `qrinsert` Calling Conventions

quad_func, quad8

Purpose

Numerical evaluation of integrals

Note The quad8 function, which implemented a higher order method, is obsolete. The quadl function is its recommended replacement.

C++ Prototype

```
mwArray quad_func(const mwArray &in1,
                  const mwArray &in2=mwArray::DIN,
                  const mwArray &in3=mwArray::DIN,
                  const mwArray &in4=mwArray::DIN,
                  const mwArray &in5=mwArray::DIN,
                  const mwVarargin &in6=mwVarargin::DIN,
                  const mwArray &in7=mwArray::DIN,
                  .
                  .
                  .
                  const mwArray &in37=mwArray::DIN);

mwArray quad_func(mwArray *out1,
                  const mwArray &in1,
                  const mwArray &in2=mwArray::DIN,
                  const mwArray &in3=mwArray::DIN,
                  const mwArray &in4=mwArray::DIN,
                  const mwArray &in5=mwArray::DIN,
                  const mwVarargin &in6=mwVarargin::DIN,
                  const mwArray &in7=mwArray::DIN,
                  .
                  .
                  .
                  const mwArray &in37=mwArray::DIN);
```

```
mwArray quad8(const mwArray &in1,
              const mwArray &in2=mwArray::DIN,
              const mwArray &in3=mwArray::DIN,
              const mwArray &in4=mwArray::DIN,
              const mwArray &in5=mwArray::DIN,
              const mwVarargin &in6=mwVarargin::DIN,
              const mwArray &in7=mwArray::DIN,
              .
              .
              .
              const mwArray &in37=mwArray::DIN);

mwArray quad8(mwArray *out1,
              const mwArray &in1,
              const mwArray &in2=mwArray::DIN,
              const mwArray &in3=mwArray::DIN,
              const mwArray &in4=mwArray::DIN,
              const mwArray &in5=mwArray::DIN,
              const mwVarargin &in6=mwVarargin::DIN,
              const mwArray &in7=mwArray::DIN,
              .
              .
              .
              const mwArray &in37=mwArray::DIN);
```

quad_func, quad8

C++ Syntax

```
#include "matlab.hpp"

mwArray func;               // String array(s)
mwArray a, b, tol;          // Input argument(s)
mwArray trace, P1;          // Input argument(s)
mwArray count;              // Output argument(s)
mwArray q;                  // Return value

q = quad_func(func,a,b);
q = quad_func(func,a,b,tol);
q = quad_func(func,a,b,tol,trace);
q = quad_func(func,a,b,tol,trace,P1,P2,...);

q = quad8(func,a,b);
q = quad8(func,a,b,tol);
q = quad8(func,a,b,tol,trace);
q = quad8(func,a,b,tol,trace,P1,P2,...);

q = quad8(&count,func,a,b);
q = quad8(&count,func,a,b,tol);
q = quad8(&count,func,a,b,tol,trace,P1,P2,...);
```

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, quadl Calling Conventions

Purpose

QZ factorization for generalized eigenvalues

C++ Prototype

```
mwArray qz(mwArray *BB, mwArray *Q, mwArray *Z, mwArray *V,
           mwArray *W, const mwArray &A, const mwArray &B);

mwArray qz(mwArray *BB, mwArray *Q, mwArray *Z, mwArray *V,
           mwArray *W, const mwArray &A, const mwArray &B,
           const mwArray &flag);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, B, flag;         // Input argument(s)
mwArray BB, Q, Z, V, W;     // Output argument(s)
mwArray AA;                 // Return value

AA = qz(&BB,&Q,&Z,&V,&W,A,B);
AA = qz(&BB,&Q,&Z,&V,&W,A,B,flag);
```

MATLAB Syntax

```
[AA,BB,Q,Z,V,W] = qz(A,B)
[AA,BB,Q,Z,V,W] = qz(A,B,flag)
```

See Also

MATLAB `qz` Calling Conventions

ramp

Purpose Generate a vector of elements

C++ Prototype

```
mwArray ramp(mwArray start, mwArray end);
mwArray ramp(mwArray start, mwArray step, mwArray end);
```

Purpose

Uniformly distributed random numbers and arrays

C++ Prototype

```
mwArray rand(const mwVarargin &in1=mwVarargin::DIN,
             const mwArray &in2=mwArray::DIN,
             .
             .
             .
             const mwArray &in32=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray state;          // String array(s)
mwArray m, n, p, A;     // Input argument(s)
mwArray Y, s;           // Return value

Y = rand(n);
Y = rand(m,n);
Y = rand(horzcat(m,n));
Y = rand(m,n,p,...);
Y = rand(horzcat(m,n,p,...));
Y = rand(size(A));
Y = rand();

s = rand("state");
s = rand("state",state);
```

MATLAB Syntax

```
Y = rand(n)
Y = rand(m,n)
Y = rand([m n])
Y = rand(m,n,p,...)
Y = rand([m n p...])
Y = rand(size(A))
rand
s = rand('state')
s = rand('state',state);
```

See Also

MATLAB `rand` Calling Conventions

randn

Purpose

Normally distributed random numbers and arrays

C++ Prototype

```
mwArray randn(const mwVarargin &in1=mwVarargin::DIN,
              const mwArray &in2=mwArray::DIN,
              .
              .
              .
              const mwArray &in32=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray m, n, p, A, S;  // Input argument(s)
mwArray Y;              // Return value

Y = randn(n);
Y = randn(m,n);
Y = randn(horzcat(m,n));
Y = randn(m,n,p,...);
Y = randn(horzcat(m,n,p,...));
Y = randn(size(A));
Y = randn();
Y = randn("state");
Y = randn("state",S);
```

MATLAB Syntax

```
Y = randn(n)
Y = randn(m,n)
Y = randn([m n])
Y = randn(m,n,p,...)
Y = randn([m n p...])
Y = randn(size(A))
randn
s = randn('state')
s = randn('state',S)
```

See Also

MATLAB `randn` Calling Conventions

Purpose

Random permutation

C++ Prototype

```
mwArray randperm(const mwArray &n);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray n;              // Input argument(s)
mwArray p;              // Return value

p = randperm(n);
```

MATLAB Syntax

```
p = randperm(n)
```

See Also

MATLAB `randperm` Calling Conventions

rank

Purpose

Rank of a matrix

C++ Prototype

```
mwArray rank(const mwArray &A);
mwArray rank(const mwArray &A, const mwArray &tol);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, tol;         // Input argument(s)
mwArray k;              // Return value

k = rank(A);
k = rank(A,tol);
```

MATLAB Syntax

```
k = rank(A)
k = rank(A,tol)
```

See Also

MATLAB `rank`

Calling Conventions

Purpose

Rational fraction approximation

C++ Prototype

```
mwArray rat(mwArray *D, const mwArray &X);
mwArray rat(mwArray *D, const mwArray &X, const mwArray &tol);
mwArray rat(const mwArray &X);
mwArray rat(const mwArray &X, const mwArray &tol);
mwArray rats(const mwArray &X, const mwArray &stln);
mwArray rats(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, tol, stln;   // Input argument(s)
mwArray D;              // Output argument(s)
mwArray N, S;           // Return value

N = rat(&D,X);
N = rat(&D,X,tol);
N = rat(X);
N = rat(X,tol);

S = rats(X,strlen);
S = rats(X);
```

MATLAB Syntax

```
[N,D] = rat(X)
[N,D] = rat(X,tol)
rat(...)
S = rats(X,strlen)
S = rats(X)
```

See Also

MATLAB `rat`, `rats` Calling Conventions

rcond

Purpose

Matrix reciprocal condition number estimate

C++ Prototype

```
mwArray rcond(const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Input argument(s)
mwArray c;              // Return value

c = rcond(A);
```

MATLAB Syntax

```
c = rcond(A)
```

See Also

MATLAB `rcond` Calling Conventions

Purpose

Real part of complex number

C++ Prototype

```
mwArray real(const mwArray &Z);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray Z;              // Input argument(s)
mwArray X;              // Return value

X = real(Z);
```

MATLAB Syntax

```
X = real(Z)
```

See Also

MATLAB `real` Calling Conventions

realmax

Purpose

Largest positive floating-point number

C++ Prototype

```
mwArray realmax();
```

C++ Syntax

```
#include "matlab.hpp"

mwArray n;              // Return value

n = realmax();
```

MATLAB Syntax

```
n = realmax
```

See Also

MATLAB `realmax` Calling Conventions

Purpose

Smallest positive floating-point number

C++ Prototype

```
mwArray realmin();
```

C++ Syntax

```
#include "matlab.hpp"

mwArray n;              // Return value

n = realmin();
```

MATLAB Syntax

```
n = realmin
```

See Also

MATLAB `realmin` Calling Conventions

rectint

Purpose

Rectangle intersection area

C++ Prototype

```
mwArray rectint(const mwArray &a, const mwArray &b);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray a, b;               // Input argument(s)
mwArray R;                  // Return value

R = rectint(a,b);
```

MATLAB Syntax

```
rectint(a,b)
```

See Also

MATLAB `rectint` Calling Conventions

Purpose

Remainder after division

C++ Prototype

```
mwArray rem(const mwArray &X, const mwArray &Y);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, Y;           // Input argument(s)
mwArray R;              // Return value

R = rem(X,Y);
```

MATLAB Syntax

```
R = rem(X,Y)
```

See Also

MATLAB `rem`

Calling Conventions

repmat

Purpose

Replicate and tile an array

C++ Prototype

```
mwArray repmat(const mwArray &A, const mwArray &m,
               const mwArray &n);
mwArray repmat(const mwArray &A, const mwArray &dims);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, m, n, p;     // Input argument(s)
mwArray B;              // Return value

B = repmat(A,m,n);
B = repmat(A,horzcat(m,n));
B = repmat(A,horzcat(m,n,p,...));
```

MATLAB Syntax

```
B = repmat(A,m,n)
B = repmat(A,[m n])
B = repmat(A,[m n p...])
```

See Also

MATLAB `repmat` Calling Conventions

Purpose

Reshape array

C++ Prototype

```
mwArray reshape(const mwArray &in1,
                const mwVarargin &in2=mwVarargin::DIN,
                const mwArray &in3=mwArray::DIN,
                .
                .
                .
                const mwArray &in33=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, m, siz, n, p;// Input argument(s)
mwArray B;              // Return value

B = reshape(A,m,n);
B = reshape(A,m,n,p,...);
B = reshape(A,horzcat(m,n,p,...));
B = reshape(A,...,empty(),...);
B = reshape(A,siz);
```

MATLAB Syntax

```
B = reshape(A,m,n)
B = reshape(A,m,n,p,...)
B = reshape(A,[m n p...])
B = reshape(A,...,[],...)
B = reshape(A,siz)
```

See Also

MATLAB reshape Calling Conventions

resi2

Purpose

Residue of a repeated pole

C++ Prototype

```
mwArray resi2(const mwArray &u, const mwArray &v,
              const mwArray &pole, const mwArray &n,
              const mwArray &k);

mwArray resi2(const mwArray &u, const mwArray &v,
              const mwArray &pole, const mwArray &n);

mwArray resi2(const mwArray &u, const mwArray &v,
              const mwArray &pole);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray u, v, pole, n, k;       // Input argument(s)
mwArray R;                      // Return value

R = resi2(u,v,pole,n,k);
R = resi2(u,v,pole,n);
R = resi2(u,v,pole);
```

MATLAB Syntax

```
resi2(u,v,pole,n,k)
```

See Also

Calling Conventions

Purpose

Convert between partial fraction expansion and polynomial coefficients

C++ Prototype

```
mwArray residue(mwArray *p, mwArray *k, const mwArray &b,
                const mwArray &a);

mwArray residue(mwArray *a, const mwArray &r, const mwArray &p,
                const mwArray &k);

mwArray residue(const mwArray &b, const mwArray &a);

mwArray residue(const mwArray &r, const mwArray &p,
                const mwArray &k);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray r, p, k, b, a;

r = residue(&p,&k,b,a);
b = residue(&a,r,p,k);
r = residue(b,a);
b = residue(r,p,k);
```

MATLAB Syntax

```
[r,p,k] = residue(b,a)
[b,a] = residue(r,p,k)
```

See Also

MATLAB `residue` Calling Conventions

rmfield

Purpose

Remove structure fields

C++ Prototype

```
mwArray rmfield(const mwArray &s,
                const mwArray &FIELDS=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray s, FIELDS;              // Input argument(s)

s = rmfield(s,"field");
s = rmfield(s,FIELDS);
```

MATLAB Syntax

```
s = rmfield(s,'field')
s = rmfield(s,FIELDS)
```

See Also

MATLAB `rmfield` Calling Conventions

Purpose

Polynomial roots

C++ Prototype

```
mwArray roots(const mwArray &c);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray c;              // Input argument(s)
mwArray r;              // Return value

r = roots(c);
```

MATLAB Syntax

```
r = roots(c)
```

See Also

MATLAB `roots` Calling Conventions

rosser

Purpose

Classic symmetric eigenvalue test matrix (Rosser)

C++ Prototype

```
mwArray rosser();
```

C++ Syntax

```
#include "matlab.hpp"

mwArray c;                  // Input argument(s)
mwArray A;                  // Return value

A = rosser();
```

MATLAB Syntax

```
[A,B,C,...] = gallery('tmfun',P1,P2,...)
gallery(3)              a badly conditioned 3-by-3 matrix
gallery(5)              an interesting eigenvalue problem
```

See Also

MATLAB `gallery` Calling Conventions

Purpose Rotate matrix 90 degrees

C++ Prototype

```
mwArray rot90(const mwArray &A);
mwArray rot90(const mwArray &A, const mwArray &k);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, k;           // Input argument(s)
mwArray B;              // Return value

B = rot90(A);
B = rot90(A,k);
```

MATLAB Syntax

```
B = rot90(A)
B = rot90(A,k)
```

See Also MATLAB `rot90` Calling Conventions

round

Purpose

Round to nearest integer

C++ Prototype

```
mwArray round(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray Y;              // Return value

Y = round(X);
```

MATLAB Syntax

```
Y = round(X)
```

See Also

MATLAB `round`

Calling Conventions

Purpose

Reduced row echelon form

C++ Prototype

```
mwArray rref(const mwArray &A);
mwArray rref(mwArray *jb, const mwArray &A);
mwArray rref(mwArray *jb, const mwArray &A, const mwArray &tol);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, tol;         // Input argument(s)
mwArray jb;             // Output argument(s)
mwArray R;              // Return value

R = rref(A);
R = rref(&jb,A);
R = rref(&jb,A,tol);
```

MATLAB Syntax

```
R = rref(A)
[R,jb] = rref(A)
[R,jb] = rref(A,tol)
```

See Also

MATLAB `rref` Calling Conventions

rsf2csf

Purpose

Convert real Schur form to complex Schur form

C++ Prototype

```
mwArray rsf2csf(mwArray *T_out, const mwArray &U_in,
                const mwArray &T_in);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray U_in, T_in;     // Input argument(s)
mwArray T_out;          // Output argument(s)
mwArray U_out;          // Return value

U_out = rsf2csf(&T_out,U_in,T_in);
```

MATLAB Syntax

```
[U,T] = rsf2csf(U,T)
```

See Also

MATLAB `rsf2csf` Calling Conventions

Purpose

Save up to 16 mwArray variables to disk.

C++ Prototype

```
void save(const mwArray &file,
       const char* name1, const mwArray &var1,
       const char* name2=NULL, const mwArray &var2=mwArray::DIN,
       .
       .
       .
       const char* name16=NULL, const mwArray &var16=mwArray::DIN);

void save( const mwArray &file, const char* mode,
       const char* name1, const mwArray &var1,
       const char* name2=NULL, const mwArray &var2=mwArray::DIN,
       .
       .
       .
       const char* name16=NULL, const mwArray &var16=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray fname;
mwArray x, y, z;

save(fname,"X",x);
save(fname,"w","X",x);                  // overwrites data
save(fname,"X",x,"Y",y,"Z",z,...);
save(fname,"u","X",x,"Y",y,"Z",z,...);  // appends data
```

MATLAB Syntax

```
save fname X
save fname X,Y,Z
```

See Also

MATLAB save Calling Conventions

schur

Purpose

Schur decomposition

C++ Prototype

```
mwArray schur(mwArray *T, const mwArray &A);
mwArray schur(const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Input argument(s)
mwArray T;              // Output argument and Return value
mwArray U;              // Return value

U = schur(&T,A);
T = schur(A);
```

MATLAB Syntax

```
[U,T] = schur(A)
T = schur(A)
```

See Also

MATLAB `schur` Calling Conventions

Purpose

Secant and hyperbolic secant

C++ Prototype

```
mwArray sec(const mwArray &X);
mwArray sech(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray Y;              // Return value

Y = sec(X);
Y = sech(X);
```

MATLAB Syntax

```
Y = sec(X)
Y = sech(X)
```

See Also

MATLAB `sec`, `sech` Calling Conventions

setdiff

Purpose

Return the set difference of two vectors

C++ Prototype

```
mwArray setdiff(const mwArray &a, const mwArray &b);
mwArray setdiff(const mwArray &A, const mwArray &B,
                const mwArray &flag);
mwArray setdiff(mwArray *i, const mwArray &a, const mwArray &b);
mwArray setdiff(mwArray *i, const mwArray &A, const mwArray &B,
                const mwArray &flag);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray a, b, A, B;         // Input argument(s)
mwArray i;                  // Output argument(s)
mwArray c;                  // Return value

c = setdiff(a,b);
c = setdiff(A,B,"rows");
c = setdiff(&i,a,b);
c = setdiff(&i,A,B,"rows");
```

MATLAB Syntax

```
c = setdiff(a,b)
c = setdiff(A,B,'rows')
[c,i] = setdiff(...)
```

See Also

MATLAB `setdiff` Calling Conventions

Purpose Set field of structure array

C++ Prototype

```
mwArray setfield(const mwArray &in1,
                 const mwVarargin &in2=mwVarargin::DIN,
                 const mwArray &in3=mwArray::DIN,
                 .
                 .
                 .
                 const mwArray &in33=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray s,i,j,k,v;                  // Input argument(s)
mwArray s;                          // Return value

s = setfield(s,"field",v);
s = setfield(s,cellhcat(i,j),"field",cellhcat(k),v);
```

MATLAB Syntax

```
s = setfield(s,'field',v)
s = setfield(s,{i,j},'field',{k},v)
```

See Also MATLAB `setfield` Calling Conventions

setstr

Purpose

Set string flag

This MATLAB 4 function has been renamed `char_func` in MATLAB 5.

See Also

MATLAB `char` Calling Conventions

Purpose

Set exclusive-or of two vectors

C++ Prototype

```
mwArray setxor(const mwArray &a, const mwArray &b);
mwArray setxor(const mwArray &A, const mwArray &B,
               const mwArray &flag);
mwArray setxor(mwArray *ia, mwArray *ib, const mwArray &a,
               const mwArray &b);
mwArray setxor(mwArray *ia, mwArray *ib, const mwArray &A,
               const mwArray &B, const mwArray &flag);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray a, b, A, B;             // Input argument(s)
mwArray ia, ib;                 // Output argument(s)
mwArray c;                      // Return value

c = setxor(a,b);
c = setxor(A,B,"rows");
c = setxor(&ia,&ib,a,b);
c = setxor(&ia,&ib,A,B,"rows");
```

MATLAB Syntax

```
c = setxor(a,b)
c = setxor(A,B,'rows')
[c,ia,ib] = setxor(...)
```

See Also

MATLAB setxor Calling Conventions

shiftdim

Purpose

Shift dimensions

C++ Prototype

```
mwArray shiftdim(const mwArray &X, const mwArray &n); // NA for
   customers
mwArray shiftdim(const mwArray &X);
mwArray shiftdim(mwArray *nshifts, const mwArray &X);
mwArray shiftdim(mwArray *nshifts, const mwArray &X,
                 const mwArray &n); // NA for customers
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;                  // Input argument(s)
mwArray nshifts;            // Output argument(s)
mwArray B;                  // Return value

B = shiftdim(X);
B = shiftdim(&nshifts,X);
```

MATLAB Syntax

```
B = shiftdim(X,n)
[B,nshifts] = shiftdim(X)
```

See Also

MATLAB `shiftdim` Calling Conventions

Purpose

Signum function

C++ Prototype

```
mwArray sign(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray Y;              // Return value

Y = sign(X);
```

MATLAB Syntax

```
Y = sign(X)
```

See Also

MATLAB `sign` Calling Conventions

sin, sinh

Purpose

Sine and hyperbolic sine

C++ Prototype

```
mwArray sin(const mwArray &X);
mwArray sinh(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray Y;              // Return value

Y = sin(X);
Y = sinh(X);
```

MATLAB Syntax

```
Y = sin(X)
Y = sinh(X)
```

See Also

MATLAB `sin`, `sinh` Calling Conventions

Purpose

Array dimensions

C++ Prototype

```
mwArray size(const mwArray &X, const mwArray &dim=mwArray::DIN);

mwArray size(mwVarargout varargout,
             const mwArray &X,
             const mwArray &dim=mwArray::DIN);

int size(int *cols, const mwArray &array);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, dim;             // Input argument(s)
mwArray n;                  // Output argument(s)
mwArray d1, d2, ..., dn;    // Output argument(s)
mwArray d, m;               // Return value

int i;                      // Return value
int j;                      // Output argument(s)

d = size(X);
size(mwVarargout(m,n),X);
m = size(X,dim);
size(mwVarargout(d1, d2, ..., dn), X);// X has n dimensions
i = size(&j,X);         // An efficient version of size(X,dim);
```

MATLAB Syntax

```
d = size(X)
[m,n] = size(X)
m = size(X,dim)
[d1,d2,...,dn] = size(X)
```

See Also

MATLAB `size` Calling Conventions

sort

Purpose

Sort elements in ascending order

C++ Prototype

```
mwArray sort(const mwArray &A);
mwArray sort(mwArray *INDEX, const mwArray &A);
mwArray sort(const mwArray &A, const mwArray &dim);
mwArray sort(mwArray *INDEX, const mwArray &A, const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, dim;         // Input argument(s)
mwArray INDEX;          // Output argument(s)
mwArray B;              // Return value

B = sort(A);
B = sort(&INDEX,A);
B = sort(A,dim);
B = sort(&INDEX,A,dim);
```

MATLAB Syntax

```
B = sort(A)
[B,INDEX] = sort(A)
B = sort(A,dim)
```

See Also

MATLAB `sort` Calling Conventions

Purpose

Sort rows in ascending order

C++ Prototype

```
mwArray sortrows(const mwArray &A);
mwArray sortrows(const mwArray &A, const mwArray &column);
mwArray sortrows(mwArray *index, const mwArray &A);
mwArray sortrows(mwArray *index, const mwArray &A,
                 const mwArray &column);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, column;              // Input argument(s)
mwArray index;                  // Output argument(s)
mwArray B;                      // Return value

B = sortrows(A);
B = sortrows(A,column);
B = sortrows(&index,A);
B = sortrows(&index,A,column);
```

MATLAB Syntax

```
B = sortrows(A)
B = sortrows(A,column)
[B,index] = sortrows(A)
```

See Also

MATLAB `sortrows` Calling Conventions

spalloc

Purpose

Allocate space for sparse matrix

C++ Prototype

```
mwArray spalloc(const mwArray &m,
                const mwArray &n=mwArray::DIN,
                const mwArray &nzmax=mwArray::DIN)
```

C++ Syntax

```
#include "matlab.hpp"

mwArray m, n, nzmax;        // Input argument(s)
mwArray S;                  // Return value

S = spalloc(m,n,nzmax);
```

MATLAB Syntax

```
S = spalloc(m,n,nzmax)
```

See Also

MATLAB `spalloc` Calling Conventions

Purpose

Create sparse matrix

C++ Prototype

```
mwArray sparse(const mwArray &i,
               const mwArray &j=mwArray::DIN,
               const mwArray &s=mwArray::DIN,
               const mwArray &m=mwArray::DIN,
               const mwArray &n=mwArray::DIN,
               const mwArray &nzmax=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, i, j, s, m, n, nzmax;// Input argument(s)
mwArray S;                      // Return value

S = sparse(A);
S = sparse(i,j,s,m,n,nzmax);
S = sparse(i,j,s,m,n);
S = sparse(i,j,s);
S = sparse(m,n);
```

MATLAB Syntax

```
S = sparse(A)
S = sparse(i,j,s,m,n,nzmax)
S = sparse(i,j,s,m,n)
S = sparse(i,j,s)
S = sparse(m,n)
```

See Also

MATLAB sparse Calling Conventions

spconvert

Purpose

Import matrix from sparse matrix external format

C++ Prototype

```
mwArray spconvert(const mwArray &D)
```

C++ Syntax

```
#include "matlab.hpp"

mwArray D;                  // Input argument(s)
mwArray S;                  // Return value

S = spconvert(D);
```

MATLAB Syntax

```
S = spconvert(D)
```

See Also

MATLAB `spconvert` Calling Conventions

Purpose

Extract and create sparse band and diagonal matrices

C++ Prototype

```
mwArray spdiags(const mwArray &in1,
                const mwArray &in2=mwArray::DIN,
                const mwArray &in3=mwArray::DIN,
                const mwArray &in4=mwArray::DIN);

mwArray spdiags(mwArray *out1,
                const mwArray &in1);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray m, n;
mwArray A, B, d;

B = spdiags(&d,A);
B = spdiags(A,d);
A = spdiags(B,d,A);
A = spdiags(B,d,m,n);
```

MATLAB Syntax

```
[B,d] = spdiags(A)
B = spdiags(A,d)
A = spdiags(B,d,A)
A = spdiags(B,d,m,n)
```

See Also

MATLAB spdiags Calling Conventions

speye

Purpose

Sparse identity matrix

C++ Prototype

```
mwArray speye(const mwArray &m,
              const mwArray &n=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray m, n;           // Input argument(s)
mwArray S;              // Return value

S = speye(m,n);
S = speye(n);
```

MATLAB Syntax

```
S = speye(m,n)
S = speye(n)
```

See Also

MATLAB `speye` Calling Conventions

Purpose

Apply function to nonzero sparse matrix elements

C++ Prototype

```
mwArray spfun(const mwArray &fcn,
              const mwArray &S=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray S;              // Input argument(s)
mwArray f;              // Return value

f = spfun("function",S);
```

MATLAB Syntax

```
f = spfun('function',S)
```

See Also

MATLAB `spfun` Calling Conventions

sph2cart

Purpose

Transform spherical coordinates to Cartesian

C++ Prototype

```
mwArray sph2cart(mwArray *y, mwArray *z, const mwArray &THETA,
                 const mwArray &PHI, const mwArray &R);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray THETA, PHI, R;  // Input argument(s)
mwArray y, z;           // Output argument(s)
mwArray x;              // Return value

x = sph2cart(&y,&z,THETA,PHI,R);
```

MATLAB Syntax

```
[x,y,z] = sph2cart(THETA,PHI,R)
```

See Also

MATLAB `sph2cart` Calling Conventions

Purpose

Cubic spline interpolation

C++ Prototype

```
mwArray spline(const mwArray &x,
               const mwArray &y=mwArray::DIN,
               const mwArray &xi=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray x, y, xi;       // Input argument(s)
mwArray yi, pp;         // Return value

yi = spline(x,y,xi);
pp = spline(x,y);
```

MATLAB Syntax

```
yi = spline(x,y,xi)
pp = spline(x,y)
```

See Also

MATLAB `spline` Calling Conventions

spones

Purpose

Replace nonzero sparse matrix elements with ones

C++ Prototype

```
mwArray spones(const mwArray &S);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray S;              // Input argument(s)
mwArray R;              // Return value

R = spones(S);
```

MATLAB Syntax

```
R = spones(S)
```

See Also

MATLAB spones Calling Conventions

Purpose

Set parameters for sparse matrix routines

C++ Prototype

```
mwArray spparms(const mwArray &in1=mwArray::DIN,
                const mwArray &in2=mwArray::DIN);

mwArray spparms(mwArray *values,
                const mwArray &key=mwArray::DIN,
                const mwArray &value=mwArray::DIN);

void Vspparms(const mwArray &in1=mwArray::DIN,
                const mwArray &in2=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray values, keys, value;

Vspparms("key",value);
Vspparms();
values = spparms();
keys = spparms(&values);
Vspparms(values);
value = spparms("key");
Vspparms("default");
Vspparms("tight");
```

Note Use `Vspparms` if you are not assigning the result to an `mwArray`.

MATLAB Syntax

```
spparms('key',value)
spparms
values = spparms
[keys,values] = spparms
spparms(values)
value = spparms('key')
spparms('default')
spparms('tight')
```

See Also MATLAB `spparms` Calling Conventions

Purpose

Sparse uniformly distributed random matrix

C++ Prototype

```
mwArray sprand(const mwArray &m,
               const mwArray &n=mwArray::DIN,
               const mwArray &density=mwArray::DIN,
               const mwArray &rc=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray S, m, n, density, rc;   // Input argument(s)
mwArray R;                      // Return value

R = sprand(S);
R = sprand(m,n,density);
R = sprand(m,n,density,rc);
```

MATLAB Syntax

```
R = sprand(S)
R = sprand(m,n,density)
R = sprand(m,n,density,rc)
```

See Also

MATLAB sprand Calling Conventions

sprandn

Purpose

Sparse normally distributed random matrix

C++ Prototype

```
mwArray sprandn(const mwArray &m,
                const mwArray &n=mwArray::DIN,
                const mwArray &density=mwArray::DIN,
                const mwArray &rc=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray S, m, n, density, rc;   // Input argument(s)
mwArray R;                      // Return value

R = sprandn(S);
R = sprandn(m,n,density);
R = sprandn(m,n,density,rc);
```

MATLAB Syntax

```
R = sprandn(S)
R = sprandn(m,n,density)
R = sprandn(m,n,density,rc)
```

See Also

MATLAB `sprandn` Calling Conventions

Purpose

Sparse symmetric random matrix

C++ Prototype

```
mwArray sprandsym(const mwArray &n,
                  const mwArray &density=mwArray::DIN,
                  const mwArray &rc=mwArray::DIN,
                  const mwArray &kind=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray S, n, density, rc, kind;    // Input argument(s)
mwArray R;                          // Return value

R = sprandsym(S);
R = sprandsym(n,density);
R = sprandsym(n,density,rc);
R = sprandsym(n,density,rc,kind);
```

MATLAB Syntax

```
R = sprandsym(S)
R = sprandsym(n,density)
R = sprandsym(n,density,rc)
R = sprandsym(n,density,rc,kind)
```

See Also

MATLAB `sprandsym` Calling Conventions

sprintf

Purpose

Write formatted data to a string

C++ Prototype

```
mwArray sprintf(const mwArray &RI1);

mwArray sprintf(const mwArray &in1,
                mwVarargin in2,
                const mwArray &in3=mwArray::DIN,
                .
                .
                .
                const mwArray &in33=mwArray::DIN);

mwArray sprintf(mwArray *out1,
                const mwArray &in1,
                const mwVarargin &in2=mwVarargin::DIN,
                const mwArray &in3=mwArray::DIN,
                .
                .
                .
                const mwArray &in33=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray format;             // String array(s)
mwArray A1, A2;             // Input argument(s)
mwArray errmsg;             // Output argument(s)
mwArray count;              // Return value
mwArray s;                  // Return value

count = sprintf(format);

count = sprintf(format,A1);
count = sprintf(format,A1,A2,...);
s = sprintf(&errmsg,format,A1);
s = sprintf(&errmsg,format,A1,A2,...);
```

MATLAB Syntax

```
s = sprintf(format,A,...)
[s,errmsg] = sprintf(format,A,...)
```

See Also

MATLAB `sprintf` Calling Conventions

sqrt

Purpose

Square root

C++ Prototype

```
mwArray sqrt(const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Input argument(s)
mwArray B;              // Return value

B = sqrt(A);
```

MATLAB Syntax

```
B = sqrt(A)
```

See Also

MATLAB `sqrt` Calling Conventions

Purpose

Matrix square root

C++ Prototype

```
mwArray sqrtm(const mwArray &X);
mwArray sqrtm(mwArray *esterr, const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray esterr;         // Output argument(s)
mwArray Y;              // Return value

Y = sqrtm(X);
Y = sqrtm(&esterr,X);
```

MATLAB Syntax

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

See Also

MATLAB `sqrtm` Calling Conventions

sscanf

Purpose

Read string under format control

C++ Prototype

```
mwArray sscanf(const mwArray &s, const mwArray &format);
mwArray sscanf(const mwArray &s, const mwArray &format,
               const mwArray &size);
mwArray sscanf(mwArray *count, mwArray *errmsg, mwArray *nextindex,
               const mwArray &s, const mwArray &format,
               const mwArray &size);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray s, format;                  // String array(s)
mwArray size;                       // Input argument(s)
mwArray count, errmsg, nextindex;   // Output argument(s)
mwArray A;                          // Return value

A = sscanf(s,format);
A = sscanf(s,format,size);
A = sscanf(&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

```
mwArray std_func(const mwArray &X);
mwArray std_func(const mwArray &X, const mwArray &flag);
mwArray std_func(const mwArray &X, const mwArray &flag,
            const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, flag, dim;   // Input argument(s)
mwArray s;              // Return value

s = std_func(X);
s = std_func(X,flag);
s = std_func(X,flag,dim);
```

MATLAB Syntax

```
s = std(X)
s = std(X,flag)
s = std(X,flag,dim)
```

See Also

MATLAB `std` Calling Conventions

str2double

Purpose

Convert string to double-precision value

C++ Prototype

```
mwArray str2double(const mwArray &C);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray C;              // Input argument(s)
mwArray X;              // Return value

x = str2double("str");
X = str2double(C);
```

MATLAB Syntax

```
x = str2double('str')
X = str2double(C)
```

See Also

MATLAB `str2double` Calling Conventions

Purpose

Form blank padded character matrix from strings

Note This routine will become obsolete in a future version. Use char_func instead.

C++ Prototype

```
mwArray str2mat(const mwVarargin &in1,
                const mwArray &in2=mwArray::DIN,
                .
                .
                .
                const mwArray &in32=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray s1, s2, s3;     // Input argument(s)
mwArray R;              // Return value

R = str2mat(s1);
R = str2mat(s1,s2);
R = str2mat(s1,s2,s3,...);
```

MATLAB Syntax

```
t = str2mat(s1,s2,s3,...)
```

See Also

MATLAB char Calling Conventions

str2num

Purpose

String to number conversion

C++ Prototype

```
mwArray str2num(const mwArray &str);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray str;            // String array(s)
mwArray x;              // Return value

x = str2num(str);
```

MATLAB Syntax

```
x = str2num('str')
```

See Also

MATLAB `str2num` Calling Conventions

Purpose

String concatenation

C++ Prototype

```
strcat(const mwVarargin &in1,
       const mwArray &in2=mwArray::DIN,
       .
       .
       .
       const mwArray &in32=mwArray::DIN)
```

C++ Syntax

```
#include "matlab.hpp"

mwArray s1, s2, s3;     // Input argument(s)
mwArray t;              // Return value

t = strcat(s1,s2);
t = strcat(s1,s2,s3,...);
```

MATLAB Syntax

```
t = strcat(s1,s2,s3,...)
```

See Also

MATLAB strcat Calling Conventions

strcmp

Purpose

Compare strings

C++ Prototype

```
mwArray strcmp(const mwArray &str1,
               const mwArray &str2=mwArray::DIN)
```

C++ Syntax

```
#include "matlab.hpp"

mwArray S, T;           // Input argument(s)
mwArray k, TF;          // Return value

k = strcmp("str1","str2");
TF = strcmp(S,T);
```

MATLAB Syntax

```
k = strcmp('str1','str2')
TF = strcmp(S,T)
```

See Also

MATLAB `strcmp` Calling Conventions

Purpose

Compare strings ignoring case

C++ Prototype

```
mwArray strcmpi(const mwArray &str1,
                const mwArray &str2);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray S, T;           // Input argument(s)
mwArray k, TF;          // Return value

k = strcmpi("str1","str2");
TF = strcmpi(S,T);
```

MATLAB Syntax

```
k = strcmpi(str1,str2)
TF = strcmpi(S,T)
```

See Also

MATLAB `strcmpi` Calling Conventions

strjust

Purpose

Justify a character array

C++ Prototype

```
mwArray strjust(const mwArray &S,
                const mwArray &justify=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray S;                  // Input argument(s)
mwArray T;                  // Return value

T = strjust(S);
T = strjust(S,"right");
T = strjust(S,"left");
T = strjust(S,"center");
```

MATLAB Syntax

```
T = strjust(S)
T = strjust(S,'right')
T = strjust(S,'left')
T = strjust(S,'center')
```

See Also

MATLAB `strjust` Calling Conventions

Purpose Find possible matches for a string

C++ Prototype

```
mwArray strmatch(const mwArray &str,
                 const mwArray &STRS,
                 const mwArray &flag=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray STRS;                        // Input argument(s)

i = strmatch("str",STRS);
i = strmatch("str",STRS,"exact");
```

MATLAB Syntax

```
i = strmatch('str',STRS)
i = strmatch('str',STRS,'exact')
```

See Also MATLAB `strmatch` Calling Conventions

strncmp

Purpose

Compare the first n characters of two strings

C++ Prototype

```
mwArray strncmp(const mwArray &str1,
                const mwArray &str2=mwArray::DIN,
                const mwArray &n=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray S, T, n;        // Input argument(s)
mwArray k, TF;          // Return value

k = strncmp("str1","str2",n);
TF = strncmp(S,T,n);
```

MATLAB Syntax

```
k = strncmp('str1','str2',n)
TF = strncmp(S,T,n)
```

See Also

MATLAB `strncmp` Calling Conventions

Purpose

Compare first n characters of strings ignoring case

C++ Prototype

```
mwArray strncmpi(const mwArray &str1,
                 const mwArray &str2,
                 const mwArray &n);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray S, T, n;            // Input argument(s)
mwArray k, TF;              // Return value

k = strncmpi("str1","str2",n);
TF = strncmpi(S,T,n);
```

MATLAB Syntax

```
k = strncmpi('str1','str2',n)
TF = strncmpi(S,T,n)
```

See Also

MATLAB `strncmpi` Calling Conventions

strrep

Purpose

String search and replace

C++ Prototype

```
mwArray strrep(const mwArray &str1, const mwArray &str2,
               const mwArray &str3);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray str1, str2, str3;   // String array(s)
mwArray str;                // Return value

str = strrep(str1,str2,str3);
```

MATLAB Syntax

```
str = strrep(str1,str2,str3)
```

See Also

MATLAB `strrep` Calling Conventions

Purpose

First token in string

C++ Prototype

```
mwArray strtok(const mwArray &str, const mwArray &delimiter);
mwArray strtok(const mwArray &str);
mwArray strtok(mwArray *rem, const mwArray &str);
mwArray strtok(mwArray *rem, const mwArray &str,
               const mwArray &delimiter);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray str;                // String array
mwArray delimiter;          // Input argument(s)
mwArray rem;                // Output argument(s)
mwArray token;              // Return value

token = strtok(str,delimiter);
token = strtok(str);
token = strtok(&rem,str);
token = strtok(&rem,str,delimiter);
```

MATLAB Syntax

```
token = strtok('str',delimiter)
token = strtok('str')
[token,rem] = strtok(...)
```

See Also

MATLAB `strtok` Calling Conventions

struct_func

Purpose

Create structure array

C++ Prototype

```
mwArray struct_func(const mwVarargin &in1,
                    const mwArray &in2=mwArray::DIN,
                    .
                    .
                    .
                    const mwArray &in32=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray values1, values2;   // Input argument(s)
mwArray s;                  // Return value

s = struct_func("field1",values1,"field2",values2,...);
```

MATLAB Syntax

```
s = struct('field1',values1,'field2',values2,...)
```

See Also

MATLAB `struct` Calling Conventions

Purpose

Structure to cell array conversion

C++ Prototype

```
mwArray struct2cell(const mwArray &s);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray s;              // Input argument(s)
mwArray c;              // Return value

c = struct2cell(s);
```

MATLAB Syntax

```
c = struct2cell(s)
```

See Also

MATLAB `struct2cell` Calling Conventions

strvcat

Purpose

Vertical concatenation of strings

C++ Prototype

```
mwArray strvcat(const mwVarargin &in1,
                const mwArray &in2=mwArray::DIN,
                .
                .
                .
                const mwArray &in32=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray t1, t2, t3;     // Input argument(s)
mwArray S;              // Return value

S = strvcat(t1,t2);
S = strvcat(t1,t2,t3,...);
```

MATLAB Syntax

```
S = strvcat(t1,t2,t3,...)
```

See Also

MATLAB `strvcat` Calling Conventions

Purpose Single index from subscripts

C++ Prototype

```
mwArray sub2ind(const mwArray &in1,
                const mwVarargin &in2,
                const mwArray &in3,
                const mwArray &in4=mwArray::DIN,
                .
                .
                .
                const mwArray &in33=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

IND = sub2ind(siz,I,J);
IND = sub2ind(siz,I1,I2,I3,...);
```

MATLAB Syntax

```
IND = sub2ind(siz,I,J)
IND = sub2ind(siz,I1,I2,...,In)
```

See Also MATLAB `sub2ind` Calling Conventions

subspace

Purpose

Angle between two subspaces

C++ Prototype

```
mwArray subspace(const mwArray &A, const mwArray &B);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, B;           // Input argument(s)
mwArray theta;          // Return value

theta = subspace(A,B);
```

MATLAB Syntax

```
theta = subspace(A,B)
```

See Also

MATLAB subspace

Calling Conventions

Purpose

Sum of array elements

C++ Prototype

```
mwArray sum(const mwArray &A);
mwArray sum(const mwArray &A, const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, dim;         // Input argument(s)
mwArray B;              // Return value

B = sum(A);
B = sum(A,dim);
```

MATLAB Syntax

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

See Also

MATLAB `sum` Calling Conventions

svd

Purpose

Singular value decomposition

C++ Prototype

```
mwArray svd(const mwArray &X);
mwArray svd(mwArray *S, mwArray *V, const mwArray &X);
mwArray svd(mwArray *S, mwArray *V, const mwArray &X,
            const mwArray &Zero);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray S, V;           // Output argument(s)
mwArray s, U;           // Return value

s = svd(X);
U = svd(&S,&V,X);
U = svd(&S,&V,X,0);
```

MATLAB Syntax

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

See Also

MATLAB `svd` Calling Conventions

Purpose

A few singular values

C++ Prototype

```
mwArray svds(const mwVarargin &in1,
             const mwArray &in2=mwArray::DIN,
             .
             .
             .
             const mwArray &in32=mwArray::DIN);

mwArray svds(mwArray *out1, mwArray *out2,
             const mwVarargin &in1,
             const mwArray &in2=mwArray::DIN,
             .
             .
             .
             const mwArray &in32=mwArray::DIN);

mwArray svds(mwArray *out1, mwArray *out2, mwArray *out3,
             const mwVarargin &in1,
             const mwArray &in2=mwArray::DIN,
             .
             .
             .
             const mwArray &in32=mwArray::DIN);
```

svds

C++ Syntax

```
#include "matlab.hpp"

mwArray A, k;                    // Input argument(s)
mwArray S, V;                    // Output argument(s)
mwArray s, U;                    // Return value

s = svds(A);
s = svds(A,k);
s = svds(A,k,0);

U = svds(&S,&V,A);
U = svds(&S,&V,A,k);
U = svds(&S,&V,A,k,0);
U = svds(&S,&V,A,...);
```

MATLAB Syntax

```
s = svds(A)
s = svds(A,k)
s = svds(A,k,0)
[U,S,V] = svds(A,...)
```

See Also

MATLAB `svds` | Calling Conventions

Purpose

Sparse symmetric minimum degree ordering

C++ Prototype

```
inline mwArray symmmd(const mwArray &S);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray S;                      // Input argument(s)
mwArray p;                      // Return value

p = symmmd(S);
```

MATLAB Syntax

```
p = symmmd(S)
```

See Also

MATLAB `symmmd` Calling Conventions

symrcm

Purpose

Sparse reverse Cuthill-McKee ordering

C++ Prototype

```
mwArray symrcm(const mwArray &S);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray S;              // Input argument(s)
mwArray r;              // Return value

r = symrcm(S);
```

MATLAB Syntax

```
r = symrcm(S)
```

See Also

MATLAB `symrcm` Calling Conventions

Purpose

Tangent and hyperbolic tangent

C++ Prototype

```
mwArray tan(const mwArray &X);
mwArray tanh(const mwArray &X);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X;              // Input argument(s)
mwArray Y;              // Return value

Y = tan(X);
Y = tanh(X);
```

MATLAB Syntax

```
Y = tan(X)
Y = tanh(X)
```

See Also

MATLAB `tan`, `tanh` Calling Conventions

tic, toc, Vtoc

Purpose

Stopwatch timer

C++ Prototype

```
mwArray tic();
mwArray toc();
void Vtoc()
```

C++ Syntax

```
#include "matlab.hpp"

mwArray t;              // Return value

tic();
    any statements
Vtoc();
t = toc();
```

MATLAB Syntax

```
tic
    any statements
toc
t = toc
```

See Also

MATLAB `tic`, `toc` Calling Conventions

Purpose

Convert an array to a Boolean value by reducing the rank of the array to a scalar

C++ Prototype

```
bool tobool(const mwArray &t);
```

C++ Syntax

```
#include "matlab.hpp"

if (tobool(A != 0))
    {
        // test succeeded, do something
    }
```

toeplitz

Purpose

Toeplitz matrix

C++ Prototype

```
mwArray toeplitz(const mwArray &c, const mwArray &r);
mwArray toeplitz(const mwArray &r);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray c, r;           // Input argument(s)
mwArray T;              // Return value

T = toeplitz(c,r);
T = toeplitz(r);
```

MATLAB Syntax

```
T = toeplitz(c,r)
T = toeplitz(r)
```

See Also

MATLAB `toeplitz` Calling Conventions

Purpose

Sum of diagonal elements

C++ Prototype

```
mwArray trace(const mwArray &A);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A;              // Input argument(s)
mwArray b;              // Return value

b = trace(A);
```

MATLAB Syntax

```
b = trace(A)
```

See Also

MATLAB `trace` Calling Conventions

trapz

Purpose

Trapezoidal numerical integration

C++ Prototype

```
mwArray trapz(const mwArray &Y);
mwArray trapz(const mwArray &X, const mwArray &Y);
mwArray trapz(const mwArray &X, const mwArray &Y,
              const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, Y, dim;          // Input argument(s)
mwArray Z;                  // Return value

Z = trapz(Y);
Z = trapz(X,Y);
Z = trapz(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

```
mwArray tril(const mwArray &X);
mwArray tril(const mwArray &X, const mwArray &k);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, k;           // Input argument(s)
mwArray L;              // Return value

L = tril(X);
L = tril(X,k);
```

MATLAB Syntax

```
L = tril(X)
L = tril(X,k)
```

See Also

MATLAB `tril`

Calling Conventions

triu

Purpose

Upper triangular part of a matrix

C++ Prototype

```
mwArray triu(const mwArray &X);
mwArray triu(const mwArray &X, const mwArray &k);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray X, k;           // Input argument(s)
mwArray U;              // Return value

U = triu(X);
U = triu(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

```
mwArray union_func(const mwArray &a, const mwArray &b);
mwArray union_func(mwArray &A, const mwArray &B,
                   const mwArray &flag);
mwArray union_func(mwArray *ia, mwArray *ib, const mwArray &a,
                   const mwArray &b);
mwArray union_func(mwArray *ia, mwArray *ib, const mwArray &A,
                   const mwArray &B, const mwArray &flag);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray a, b, A, B;             // Input argument(s)
mwArray ia, ib;                 // Output argument(s)
mwArray c;                      // Return value

c = union_func(a,b);
c = union_func(A,B,"rows");
c = union_func(&ia,&ib,a,b);
c = union_func(&ia,&ib,A,B,"rows");
```

MATLAB Syntax

```
c = union(a,b)
c = union(A,B,'rows')
[c,ia,ib] = union(...)
```

See Also

MATLAB union Calling Conventions

unique

Purpose

Unique elements of a vector

C++ Prototype

```
mwArray unique(const mwArray &a);
mwArray unique(const mwArray &A, const mwArray &flag);
mwArray unique(mwArray *index, const mwArray &a);
mwArray unique(mwArray *index, const mwArray &A,
               const mwArray &flag);
mwArray unique(mwArray *index, mwArray *j, const mwArray &a);
mwArray unique(mwArray *index, mwArray *j, const mwArray &A,
               const mwArray &flag);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray a, A;               // Input argument(s)
mwArray index, j;           // Output argument(s)
mwArray b;                  // Return value

b = unique(a);
b = unique(A,"rows");
b = unique(&index,a);
b = unique(&index,A,"rows");
b = unique(&index,&j,a);
b = unique(&index,&j,A,"rows");
```

MATLAB Syntax

```
b = unique(a)
b = unique(A,'rows')
[b,index] = unique(...)
[b,index,j] = unique(...)
```

See Also

MATLAB `unique` Calling Conventions

Purpose

Correct phase angles

C++ Prototype

```
mwArray unwrap(const mwArray &P);
mwArray unwrap(const mwArray &P, const mwArray &tol);
mwArray unwrap(const mwArray &P, const mwArray &tol,
               const mwArray &dim);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray P, tol, dim;        // Input argument(s)
mwArray Q;                  // Return value

Q = unwrap(P);
Q = unwrap(P,tol);
Q = unwrap(P,empty(),dim);
Q = unwrap(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

upper

Purpose

Convert string to upper case

C++ Prototype

```
mwArray upper(const mwArray &str);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray str;                // String array(s)
mwArray t;                  // Return value

t  = upper(str);
```

MATLAB Syntax

```
t  = upper('str')
```

See Also

MATLAB `upper` Calling Conventions

Purpose

Test matrix (Vandermonde matrix)

C++ Prototype

```
mwArray vander(const mwArray &c);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray c;                  // Input argument(s)
mwArray A;                  // Return value

A = vander(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 = vander(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

vertcat

Purpose

Vertical concatenation

C++ Prototype

```
mwArray vertcat(const mwVarargin &in1,
                const mwArray &in2=mwArray::DIN,
                .
                .
                .
                const mwArray &in32=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, B, C;            // Input argument(s)
mwArray R;                  // Return value

R = vertcat(A);
R = vertcat(A,B);
R = vertcat(A,B,C,...);
```

MATLAB Syntax

```
[A;B;C...]
vertcat(A,B,C...)
```

See Also

Calling Conventions

Purpose

Display warning message

C++ Prototype

```
mwArray warning(const mwArray &message);
mwArray warning(mwArray *f, const mwArray &message);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray f;                  // Output argument(s)
mwArray s;                  // Return value

s = warning("message");
s = warning("on");
s = warning("off");
s = warning("backtrace");
s = warning("debug");
s = warning("once");
s = warning("always");
s = warning(&f);
```

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

weekday

Purpose

Day of the week

C++ Prototype

```
mwArray weekday(mwArray *S, const mwArray &D);
mwArray weekday(const mwArray &D);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray D;              // Input argument(s)
mwArray S;              // Output argument(s)
mwArray N;              // Return value

N = weekday(&S,D);
N = weekday(D);
```

MATLAB Syntax

```
[N,S] = weekday(D)
```

See Also

MATLAB `weekday` Calling Conventions

Purpose

Wilkinson's eigenvalue test matrix

C++ Prototype

```
mwArray wilkinson(const mwArray &n);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray n;              // Input argument(s)
mwArray W;              // Return value

W = wilkinson(n);
```

MATLAB Syntax

```
W = wilkinson(n)
```

See Also

MATLAB `wilkinson` Calling Conventions

xor

Purpose

Exclusive or

C++ Prototype

```
mwArray xor(const mwArray &A, const mwArray &B);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray A, B;           // Input argument(s)
mwArray C;              // Return value

C = xor(A,B);
```

MATLAB Syntax

```
C = xor(A,B)
```

See Also

MATLAB `xor`

Calling Conventions

Purpose

Create an array of all zeros

C++ Prototype

```
mwArray zeros(const mwVarargin &in1=mwVarargin::DIN,
              const mwArray &in2=mwArray::DIN,
              .
              .
              .
              const mwArray &in32=mwArray::DIN);
```

C++ Syntax

```
#include "matlab.hpp"

mwArray m, n, p, A;         // Input argument(s)
mwArray d1, d2, d3;         // Input argument(s)
mwArray B;                  // Return value

B = zeros(n);
B = zeros(m,n);
B = zeros(horzcat(m,n));
B = zeros(d1,d2,d3,...);
B = zeros(horzcat(d1,d2,d3,...));
B = zeros(size(A));
```

MATLAB Syntax

```
B = zeros(n)
B = zeros(m,n)
B = zeros([m n])
B = zeros(d1,d2,d3,...)
B = zeros([d1 d2 d3...])
B = zeros(size(A))
```

See Also

MATLAB zeros Calling Conventions

Utility Routine Reference

This section contains all the MATLAB C++ Math Library utility routines. These routines provide array creation, array indexing, and other capabilities.

Purpose

Display the given exception

C++ Prototype

```
void mwDisplayException(const mwException &ex);
```

Arguments

`ex`
Exception

Description

`mwDisplayException()` sends an exception to the output function set by the most recent call to `mwSetExceptionMsgHandler()`. If `mwSetExceptionMsgHandler()` has never been called, `mwDisplayException()` uses the default error message handling function or the output function specified by a call to `mwSetErrorMsgHandler()`.

Example

```
// try-block
try
{
     eig(A);
}
// catch-block
catch(mwException &ex)
{
    mwDisplayException(ex);
}
```

See Also

`mwGetErrorMsgHandler`, `mwGetExceptionMsgHandler`, `mwSetErrorMsgHandler`, `mwSetExceptionMsgHandler`

mwGetErrorMsgHandler

Purpose

Return a pointer to the current error handler

C++ Prototype

```
mwErrorFunc mwGetErrorMsgHandler(void);
```

Description

`mwGetErrorMsgHandler` returns a pointer to the function specified in the most recent call to `mwSetErrorMsgHandler()` or to the default error handler, if you haven't specified an error handler. The definition of `mwErrorFunc`:

```
typedef void (*mwErrorFunc)(const char *, mwBool);
```

See Also

`mwDisplayException`, `mwGetExceptionMsgHandler`, `mwSetErrorMsgHandler`, `mwSetExceptionMsgHandler`

Purpose

Return a pointer to the current exception message handler

C++ Prototype

```
mxExceptionMsgFunc mwGetExceptionMsgHandler(void);
```

Description

`mwGetExceptionMsgHandler` returns a pointer to the function specified in the most recent call to `mwSetExceptionMsgHandler()` or to the default exception message handler, if you haven't specified an exception message handler.

See Also

`mwDisplayException`, `mwGetErrorMsgHandler`, `mwSetErrorMsgHandler`, `mwSetExceptionMsgHandler`

mwGetPrintHandler

Purpose Return a pointer to current print handler

C++ Prototype

```
mwOutputFunc mxGetPrintHandler(void);
```

Description `mwGetPrintHandler` returns a pointer to the function specified in the most recent call to `mwSetPrintHandler()` or to the default print handler, if you haven't specified a print handler.

See Also `mwSetPrintHandler`

Purpose

Register an error handling routine with the MATLAB C++ Math Library

C++ Prototype

```
void mwSetErrorMsgHandler(mwErrorFunc f);
```

Arguments

`mwErrorFunc f`

A pointer to an error handling routine that takes a `char *` and an `mwBool` as its arguments and returns `void`.

```
typedef void (*mwErrorFunc)(const char *, mwBool);
```

Description

If you want to separate error messages from "ordinary" output, call the function `mwSetErrorMsgHandler()` to replace the default handler. `mwSetErrorMsgHandler` sets the error handling routine. The error handler is responsible for handling all error message output.

See Also

`mwDisplayException`, `mwGetErrorMsgHandler`, `mwGetExceptionMsgHandler`, `mwSetExceptionMsgHandler`

mwSetExceptionMsgHandler

Purpose

Set an alternate exception handling function

C++ Prototype

```
void mwSetExceptionMsgHandler(mwExceptionMsgFunc f);
```

Arguments

`mwExceptionMsgFunc f`

Pointer to an exception handling function that takes an `mwException` as an argument and returns `void`.

```
typedef void (*mwExceptionMsgFunc)(const mwException &);
```

Description

The default exception handling function simply prints the exception using the error handling routine. If this behavior is inappropriate for your application, the `mwSetExceptionMsgHandler` function allows you to set an alternate exception handling function.

See Also

`mwDisplayException`, `mwGetErrorMsgHandler`, `mwGetExceptionMsgHandler`, `mwSetErrorMsgHandler`

Purpose

Set memory management functions for MATLAB C++ Math Library

C++ Prototype

```
void mwSetLibraryAllocFcns(
         mwMemCallocFunc callocProc,
         mwMemFreeFunc freeProc,
         mwMemReallocFunc reallocProc,
         mwMemAllocFunc mallocproc,
         mwMemCompactFunc=0);
```

Arguments

`callocProc`
A pointer to a function that allocates memory. `mwMemCallocFunc` is defined as:

```
typedef void *(*mwMemCallocFunc)(size_t, size_t);
```

`freeProc`
A pointer to a function that frees memory. `mwMemFreeFunc` is defined as:

```
typedef void (*mwMemFreeFunc)(void *);
```

`reallocProc`
A pointer to a function that reallocates memory. `mwMemReallocFunc` is defined as:

```
typedef void *(*mwMemReallocFunc)(void *, size_t);
```

`mallocproc`
A pointer to a function that allocates memory. `mwMemAllocFunc` is defined as:

```
typedef void *(*mwMemAllocFunc)(size_t);
```

`compactproc`
Not currently used.

Description

Sets the MATLAB C++ Math Library's memory management functions. Gives you complete control over memory management.

To set up your own memory management routines, you need to write four routines: two memory allocation routines, one memory reallocation routine, and one deallocation routine. You then call `mwSetLibraryAllocFcns()` to register those routines with the library.

You cannot omit any of the four routines. However, the last argument to `mwSetLibraryAllocFcns()`, `mwMemCompactFunc`, is not currently used and is

therefore initialized to zero. When you call `mwSetLibraryAllocFcns()`, you do not need to specify a value for it.

Purpose

Set the current print handling routine

C++ Prototype

```
void mwSetPrintHandler(mwOutputFunc f);
```

Arguments

`f`
Pointer to a function that takes a `char *` argument and returns `void`. The function displays the character string. `mwOutputFunc` is defined as:

```
typedef void (*mwOutputFunc)(const char *);
```

Description

`mwSetPrintHandler` sets the print handling routine. The print handler is responsible for handling all "normal" (nonerror) output. You must call `mwSetPrintHandler()` before calling other library routines. Otherwise the library uses the default print handler to display messages.

See Also

`mwGetPrintHandler`