

bml  
0.1.0

Generated by Doxygen 1.8.9.1

Mon Nov 16 2015 17:03:43



# Contents

<b>1</b>	<b>Basic Matrix Library (bml)</b>	<b>1</b>
1.1	Usage Examples . . . . .	1
1.2	Modifying the library itself . . . . .	1
1.3	Planned Features . . . . .	1
<b>2</b>	<b>Future Plans</b>	<b>3</b>
2.1	Matrix Types . . . . .	3
2.2	Precisions . . . . .	3
2.3	Functions . . . . .	3
<b>3</b>	<b>C Usage</b>	<b>5</b>
<b>4</b>	<b>Fortran Usage</b>	<b>7</b>
<b>5</b>	<b>Developer Documentation</b>	<b>9</b>
5.1	Developer Suggested Workflow . . . . .	9
5.2	Coding Style . . . . .	9
<b>6</b>	<b>Deprecated List</b>	<b>11</b>
<b>7</b>	<b>Module Index</b>	<b>13</b>
7.1	Modules . . . . .	13
<b>8</b>	<b>Namespace Index</b>	<b>15</b>
8.1	Namespace List . . . . .	15
<b>9</b>	<b>Class Index</b>	<b>17</b>
9.1	Class List . . . . .	17
<b>10</b>	<b>File Index</b>	<b>19</b>
10.1	File List . . . . .	19
<b>11</b>	<b>Module Documentation</b>	<b>21</b>
11.1	Allocation and Deallocation Functions (C interface) . . . . .	21
11.1.1	Detailed Description . . . . .	21

11.1.2	Function Documentation	21
11.1.2.1	<a href="#">bml_allocate_memory</a>	21
11.1.2.2	<a href="#">bml_deallocate</a>	21
11.1.2.3	<a href="#">bml_free_memory</a>	22
11.1.2.4	<a href="#">bml_identity_matrix</a>	22
11.1.2.5	<a href="#">bml_random_matrix</a>	22
11.1.2.6	<a href="#">bml_zero_matrix</a>	23
11.2	Add Functions (C interface)	24
11.2.1	Detailed Description	24
11.2.2	Function Documentation	24
11.2.2.1	<a href="#">bml_add</a>	24
11.2.2.2	<a href="#">bml_add_identity</a>	24
11.3	Converting between Matrix Formats (C interface)	26
11.3.1	Detailed Description	26
11.3.2	Function Documentation	26
11.3.2.1	<a href="#">bml_export_to_dense</a>	26
11.3.2.2	<a href="#">bml_import_from_dense</a>	27
11.4	Allocation and Deallocation Functions (Fortran interface)	28
11.4.1	Detailed Description	28
11.4.2	Function Documentation	28
11.4.2.1	<a href="#">bml_deallocate</a>	28
11.4.2.2	<a href="#">bml_identity_matrix</a>	28
11.4.2.3	<a href="#">bml_random_matrix</a>	28
11.4.2.4	<a href="#">bml_zero_matrix</a>	29
11.5	Add Functions (Fortran interface)	31
11.5.1	Detailed Description	31
11.6	Converting between Matrix Formats (Fortran interface)	32
11.6.1	Detailed Description	32
11.6.2	Function Documentation	32
11.6.2.1	<a href="#">bml_convert_from_dense_double</a>	32
11.6.2.2	<a href="#">bml_convert_from_dense_double_complex</a>	32
11.6.2.3	<a href="#">bml_convert_from_dense_single_complex</a>	33
11.6.2.4	<a href="#">bml_convert_to_dense_double</a>	33
11.6.2.5	<a href="#">bml_convert_to_dense_double_complex</a>	33
11.6.2.6	<a href="#">bml_convert_to_dense_single</a>	33
11.6.2.7	<a href="#">bml_convert_to_dense_single_complex</a>	33
<b>12</b>	<b>Namespace Documentation</b>	<b>35</b>
12.1	<a href="#">bml Module Reference</a>	35
12.1.1	Detailed Description	35

12.2	<a href="#">bml_allocate_m Module Reference</a>	35
12.2.1	<a href="#">Detailed Description</a>	35
12.3	<a href="#">bml_copy_m Module Reference</a>	35
12.3.1	<a href="#">Detailed Description</a>	36
12.3.2	<a href="#">Function/Subroutine Documentation</a>	36
12.3.2.1	<a href="#">bml_copy</a>	36
12.4	<a href="#">bml_diagonalize_m Module Reference</a>	36
12.4.1	<a href="#">Detailed Description</a>	36
12.4.2	<a href="#">Function/Subroutine Documentation</a>	36
12.4.2.1	<a href="#">bml_diagonalize</a>	36
12.5	<a href="#">bml_error_m Module Reference</a>	36
12.5.1	<a href="#">Detailed Description</a>	37
12.5.2	<a href="#">Function/Subroutine Documentation</a>	37
12.5.2.1	<a href="#">bml_debug</a>	37
12.5.2.2	<a href="#">bml_error</a>	37
12.5.2.3	<a href="#">bml_warning</a>	37
12.6	<a href="#">bml_interface_m Module Reference</a>	38
12.6.1	<a href="#">Detailed Description</a>	38
12.6.2	<a href="#">Function/Subroutine Documentation</a>	38
12.6.2.1	<a href="#">get_enum_id</a>	38
12.6.3	<a href="#">Variable Documentation</a>	39
12.6.3.1	<a href="#">bml_matrix_precision_double_complex_enum_id</a>	39
12.6.3.2	<a href="#">bml_matrix_precision_double_real_enum_id</a>	39
12.6.3.3	<a href="#">bml_matrix_precision_single_complex_enum_id</a>	39
12.6.3.4	<a href="#">bml_matrix_precision_single_real_enum_id</a>	39
12.6.3.5	<a href="#">bml_matrix_precision_uninitialized_id</a>	39
12.6.3.6	<a href="#">bml_matrix_type_dense_enum_id</a>	39
12.6.3.7	<a href="#">bml_matrix_type_ellpack_enum_id</a>	39
12.6.3.8	<a href="#">bml_matrix_type_uninitialized_enum_id</a>	40
12.7	<a href="#">bml_introspection_m Module Reference</a>	40
12.7.1	<a href="#">Detailed Description</a>	40
12.7.2	<a href="#">Function/Subroutine Documentation</a>	40
12.7.2.1	<a href="#">bml_get_bandwidth</a>	40
12.7.2.2	<a href="#">bml_get_n</a>	40
12.7.2.3	<a href="#">bml_get_row_bandwidth</a>	40
12.8	<a href="#">bml_multiply_m Module Reference</a>	41
12.8.1	<a href="#">Detailed Description</a>	41
12.8.2	<a href="#">Function/Subroutine Documentation</a>	41
12.8.2.1	<a href="#">bml_multiply</a>	41
12.9	<a href="#">bml_scale_m Module Reference</a>	41

12.9.1 Detailed Description . . . . .	42
12.9.2 Function/Subroutine Documentation . . . . .	42
12.9.2.1 scale_two . . . . .	42
12.10 bml_trace_m Module Reference . . . . .	42
12.10.1 Detailed Description . . . . .	42
12.10.2 Function/Subroutine Documentation . . . . .	42
12.10.2.1 bml_trace . . . . .	42
12.11 bml_transpose_m Module Reference . . . . .	42
12.11.1 Detailed Description . . . . .	43
12.11.2 Function/Subroutine Documentation . . . . .	43
12.11.2.1 bml_transpose . . . . .	43
12.12 bml_types_m Module Reference . . . . .	43
12.12.1 Detailed Description . . . . .	43
12.13 bml_utilities_m Module Reference . . . . .	43
12.13.1 Detailed Description . . . . .	44
12.13.2 Function/Subroutine Documentation . . . . .	44
12.13.2.1 bml_print_bml_vector . . . . .	44
12.14 bml_utilities_matrix_type_m Module Reference . . . . .	44
12.14.1 Detailed Description . . . . .	44
<b>13 Class Documentation</b>	<b>45</b>
13.1 bml_types_m::bml_matrix_t Type Reference . . . . .	45
13.1.1 Detailed Description . . . . .	45
13.2 bml_types_m::bml_vector_t Type Reference . . . . .	45
13.2.1 Detailed Description . . . . .	45
<b>14 File Documentation</b>	<b>47</b>
14.1 /home/nbock/Work/bml/src/C-interface/bml.h File Reference . . . . .	47
14.1.1 Detailed Description . . . . .	47
14.2 /home/nbock/Work/bml/src/C-interface/bml_add.h File Reference . . . . .	48
14.3 /home/nbock/Work/bml/src/C-interface/bml_allocate.h File Reference . . . . .	48
14.4 /home/nbock/Work/bml/src/C-interface/bml_convert.h File Reference . . . . .	50
14.5 /home/nbock/Work/bml/src/C-interface/bml_copy.h File Reference . . . . .	50
14.5.1 Function Documentation . . . . .	51
14.5.1.1 bml_copy . . . . .	51
14.5.1.2 bml_copy_new . . . . .	51
14.6 /home/nbock/Work/bml/src/C-interface/bml_export.h File Reference . . . . .	52
14.6.1 Function Documentation . . . . .	53
14.6.1.1 bml_convert_to_dense . . . . .	53
14.7 /home/nbock/Work/bml/src/C-interface/bml_import.h File Reference . . . . .	53
14.7.1 Function Documentation . . . . .	54

14.7.1.1	<a href="#">bml_convert_from_dense</a>	54
14.8	<a href="#">/home/nbock/Work/bml/src/C-interface/bml_introspection.h File Reference</a>	55
14.8.1	Function Documentation	55
14.8.1.1	<a href="#">bml_get_bandwidth</a>	55
14.8.1.2	<a href="#">bml_get_M</a>	56
14.8.1.3	<a href="#">bml_get_N</a>	57
14.8.1.4	<a href="#">bml_get_precision</a>	58
14.8.1.5	<a href="#">bml_get_row_bandwidth</a>	59
14.8.1.6	<a href="#">bml_get_type</a>	60
14.9	<a href="#">/home/nbock/Work/bml/src/C-interface/bml_logger.h File Reference</a>	62
14.9.1	Macro Definition Documentation	63
14.9.1.1	<a href="#">LOG_DEBUG</a>	63
14.9.1.2	<a href="#">LOG_ERROR</a>	63
14.9.1.3	<a href="#">LOG_INFO</a>	63
14.9.1.4	<a href="#">LOG_WARN</a>	63
14.9.2	Enumeration Type Documentation	63
14.9.2.1	<a href="#">bml_log_level_t</a>	63
14.9.3	Function Documentation	63
14.9.3.1	<a href="#">bml_log</a>	63
14.9.3.2	<a href="#">bml_log_location</a>	64
14.10	<a href="#">/home/nbock/Work/bml/src/C-interface/bml_multiply.h File Reference</a>	64
14.10.1	Function Documentation	65
14.10.1.1	<a href="#">bml_multiply</a>	65
14.10.1.2	<a href="#">bml_multiply_AB</a>	65
14.10.1.3	<a href="#">bml_multiply_x2</a>	66
14.11	<a href="#">/home/nbock/Work/bml/src/C-interface/bml_scale.h File Reference</a>	66
14.11.1	Function Documentation	67
14.11.1.1	<a href="#">bml_scale</a>	67
14.11.1.2	<a href="#">bml_scale_inplace</a>	67
14.11.1.3	<a href="#">bml_scale_new</a>	68
14.12	<a href="#">/home/nbock/Work/bml/src/C-interface/bml_threshold.h File Reference</a>	69
14.12.1	Function Documentation	69
14.12.1.1	<a href="#">bml_threshold</a>	69
14.12.1.2	<a href="#">bml_threshold_new</a>	70
14.13	<a href="#">/home/nbock/Work/bml/src/C-interface/bml_trace.h File Reference</a>	71
14.13.1	Function Documentation	71
14.13.1.1	<a href="#">bml_trace</a>	71
14.14	<a href="#">/home/nbock/Work/bml/src/C-interface/bml_transpose.h File Reference</a>	72
14.14.1	Function Documentation	73
14.14.1.1	<a href="#">bml_transpose</a>	73

14.14.1.2 <code>bml_transpose_new</code> . . . . .	73
14.15/home/nbock/Work/bml/src/C-interface/bml_types.h File Reference . . . . .	74
14.15.1 Typedef Documentation . . . . .	74
14.15.1.1 <code>bml_matrix_t</code> . . . . .	74
14.15.1.2 <code>bml_vector_t</code> . . . . .	74
14.15.2 Enumeration Type Documentation . . . . .	75
14.15.2.1 <code>bml_dense_order_t</code> . . . . .	75
14.15.2.2 <code>bml_matrix_precision_t</code> . . . . .	75
14.15.2.3 <code>bml_matrix_type_t</code> . . . . .	75
14.16/home/nbock/Work/bml/src/C-interface/bml_types_private.h File Reference . . . . .	75
14.17/home/nbock/Work/bml/src/C-interface/bml_utilities.h File Reference . . . . .	76
14.17.1 Function Documentation . . . . .	76
14.17.1.1 <code>bml_print_bml_matrix</code> . . . . .	76
14.17.1.2 <code>bml_print_bml_vector</code> . . . . .	77
14.17.1.3 <code>bml_print_dense_matrix</code> . . . . .	77
14.17.1.4 <code>bml_print_dense_vector</code> . . . . .	78
14.18/home/nbock/Work/bml/src/C-interface/macros.h File Reference . . . . .	78
14.18.1 Macro Definition Documentation . . . . .	78
14.18.1.1 <code>COLMAJOR</code> . . . . .	78
14.18.1.2 <code>ROWMAJOR</code> . . . . .	78
<b>Index</b>	<b>79</b>



# Chapter 1

## Basic Matrix Library (bml)

This library implements a common API for linear algebra and matrix functions in C and Fortran. It offers several data structures for matrix storage and algorithms. Currently the following matrix data types are implemented:

- dense
- ellpack (sparse)
- csr (sparse)

### 1.1 Usage Examples

Usage examples can be found here:

- [Fortran Usage](#)
- [C Usage](#)

### 1.2 Modifying the library itself

If you are interested in modifying the library code itself, please have a look at the [Developer Documentation](#).

### 1.3 Planned Features

We are planning to eventually support different matrix types and matrix operations on a variety of hardware platforms. For details, please have a look at our [future plans](#).

#### Author

Christian Negre [cnegre@lanl.gov](mailto:cnegre@lanl.gov)  
Jamaludin Mohd-Yusof [jamal@lanl.gov](mailto:jamal@lanl.gov)  
Nicolas Bock [nbock@lanl.gov](mailto:nbock@lanl.gov)  
Susan M. Mniszewski [mmm@lanl.gov](mailto:mmm@lanl.gov)

#### Copyright

Los Alamos National Laboratory 2015



## Chapter 2

# Future Plans

### 2.1 Matrix Types

Support types:

- `bml_matrix_t`
- Colinear
- Noncolinear
- Blocked Bloch Matrix

### 2.2 Precisions

The bml supports the following precisions:

- logical (for matrix masks)
- single real
- double real
- single complex
- double complex

### 2.3 Functions

The library supports the following matrix operations:

- Format Conversion
  - `bml_convert::bml_convert_from_dense`
  - `bml_convert::bml_convert_to_dense`
  - `bml_convert::bml_convert`
- Masking
  - Masked operations (restricted to a subgraph)
- Addition

- $\alpha A + \beta B$ : `bml_add::bml_add`
- $\alpha A + \beta$ : `bml_add::bml_add_identity`
- Copy
  - $B \leftarrow A$ : `bml_copy::bml_copy`
- Diagonalize
  - `bml_diagonalize::bml_diagonalize`
- Introspection
  - `bml_introspection::bml_get_type`
  - `bml_introspection::bml_get_size`
  - `bml_introspection::bml_get_bandwidth`
  - `bml_introspection::bml_get_spectral_range`
  - `bml_introspection::bml_get_HOMO_LUMO`
- Matrix manipulation:
  - `bml_get::bml_get`
  - `bml_get::bml_get_rows`
  - `bml_set::bml_set`
  - `bml_set::bml_set_rows`
- Multiplication
  - $\alpha A \times B + \beta C$ : `bml_multiply::bml_multiply`
- Printing
  - `bml_utilities::bml_print_matrix`
- Scaling
  - $A \leftarrow \alpha A$ : `bml_scale::bml_scale_one`
  - $B \leftarrow \alpha A$ : `bml_scale::bml_scale_two`
- Matrix trace
  - $\text{Tr}[A]$ : `bml_trace::bml_trace`
  - $\text{Tr}[AB]$ : `bml_trace::bml_product_trace`
- Matrix norm
  - 2-norm
  - Frobenius norm
- Matrix transpose
  - `bml_transpose::bml_transpose`
- Matrix commutator/anticommutator
  - `bml_commutator::bml_commutator`
  - `bml_commutator::bml_anticommutator`

Back to the [main page](#).

## Chapter 3

# C Usage

In C, the following example code does the same as the above Fortran code:

```
#include <bml.h>

bml_matrix_t *A = bml_zero_matrix(dense,
    single_real, 100);
bml_deallocate(&A);
```

Back to the [main page](#).



## Chapter 4

# Fortran Usage

The use of this library is pretty straightforward. In the application code, use the bml main module,

```
use bml
```

A matrix is of type

```
type(bml_matrix_t) :: a
```

There are two important things to note. First, although not explicitly state in the above example, the matrix is not yet allocated. Hence, the matrix needs to be allocated through an allocation procedure with the desired type and precision, e.g. dense:double, see the page on [allocation functions](#) for a complete list. For instance,

```
call bml_zero_matrix(BML_MATRIX_DENSE, BML_PRECISION_DOUBLE, 100, a)
```

will allocate a dense, double-precision,  $100 \times 100$  matrix which is initialized to zero. Additional functions allocate special matrices,

- `bml_allocate::bml_random_matrix` Allocate and initialize a random matrix.
- `bml_allocate::bml_identity_matrix` Allocate and initialize the identity matrix.

A matrix is deallocated by calling

```
call bml_deallocate(a)
```

Back to the [main page](#).





## Chapter 5

# Developer Documentation

### 5.1 Developer Suggested Workflow

We try to preserve a linear history in our main (master) branch. Instead of pulling (i.e. merging), we suggest you use:

```
$ git pull --rebase
```

And then

```
$ git push
```

To push your changes back to the server.

### 5.2 Coding Style

Please indent your C code using

```
$ indent -gnu -nut -i4 -bli0
```

Back to the [main page](#).



## Chapter 6

# Deprecated List

**globalScope> Member [bml\\_convert\\_from\\_dense](#)** (const bml\_matrix\_type\_t matrix\_type, const bml\_matrix\_precision\_t matrix\_precision, const bml\_dense\_order\_t order, const int N, const void \*A, const double threshold, const int M)

Deprecated API.

**globalScope> Member [bml\\_convert\\_to\\_dense](#)** (const bml\_matrix\_t \*A, const bml\_dense\_order\_t order)

Deprecated API.



## Chapter 7

# Module Index

### 7.1 Modules

Here is a list of all modules:

Allocation and Deallocation Functions (C interface) . . . . .	<a href="#">21</a>
Add Functions (C interface) . . . . .	<a href="#">24</a>
Converting between Matrix Formats (C interface) . . . . .	<a href="#">26</a>
Allocation and Deallocation Functions (Fortran interface) . . . . .	<a href="#">28</a>
Add Functions (Fortran interface) . . . . .	<a href="#">31</a>
Converting between Matrix Formats (Fortran interface) . . . . .	<a href="#">32</a>



## Chapter 8

# Namespace Index

### 8.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

<a href="#">bml</a>	Main matrix library module . . . . .	35
<a href="#">bml_allocate_m</a>	Matrix allocation functions . . . . .	35
<a href="#">bml_copy_m</a>	Copy operations on matrices . . . . .	35
<a href="#">bml_diagonalize_m</a>	Matrix diagonalization functions . . . . .	36
<a href="#">bml_error_m</a>	A module for error handling in bml . . . . .	36
<a href="#">bml_interface_m</a>	Interface module . . . . .	38
<a href="#">bml_introspection_m</a>	Introspection procedures . . . . .	40
<a href="#">bml_multiply_m</a>	Matrix multiplication . . . . .	41
<a href="#">bml_scale_m</a>	Matrix scaling for matrices . . . . .	41
<a href="#">bml_trace_m</a>	Matrix trace . . . . .	42
<a href="#">bml_transpose_m</a>	Transpose functions . . . . .	42
<a href="#">bml_types_m</a>	The basic bml types . . . . .	43
<a href="#">bml_utilities_m</a>	Utility matrix functions . . . . .	43
<a href="#">bml_utilities_matrix_type_m</a>	Utility matrix functions . . . . .	44





## Chapter 9

# Class Index

### 9.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">bml_types_m::bml_matrix_t</a>	
The bml matrix type . . . . .	45
<a href="#">bml_types_m::bml_vector_t</a>	
The bml vector type . . . . .	45



## Chapter 10

# File Index

### 10.1 File List

Here is a list of all documented files with brief descriptions:

/home/nbock/Work/bml/src/C-interface/ <b>blas.h</b>	??
/home/nbock/Work/bml/src/C-interface/ <b>bml.h</b>	47
/home/nbock/Work/bml/src/C-interface/ <b>bml_add.h</b>	48
/home/nbock/Work/bml/src/C-interface/ <b>bml_allocate.h</b>	48
/home/nbock/Work/bml/src/C-interface/ <b>bml_convert.h</b>	50
/home/nbock/Work/bml/src/C-interface/ <b>bml_copy.h</b>	50
/home/nbock/Work/bml/src/C-interface/ <b>bml_diagonalize.h</b>	??
/home/nbock/Work/bml/src/C-interface/ <b>bml_elemental.h</b>	??
/home/nbock/Work/bml/src/C-interface/ <b>bml_export.h</b>	52
/home/nbock/Work/bml/src/C-interface/ <b>bml_import.h</b>	53
/home/nbock/Work/bml/src/C-interface/ <b>bml_introspection.h</b>	55
/home/nbock/Work/bml/src/C-interface/ <b>bml_logger.h</b>	62
/home/nbock/Work/bml/src/C-interface/ <b>bml_multiply.h</b>	64
/home/nbock/Work/bml/src/C-interface/ <b>bml_scale.h</b>	66
/home/nbock/Work/bml/src/C-interface/ <b>bml_threshold.h</b>	69
/home/nbock/Work/bml/src/C-interface/ <b>bml_trace.h</b>	71
/home/nbock/Work/bml/src/C-interface/ <b>bml_transpose.h</b>	72
/home/nbock/Work/bml/src/C-interface/ <b>bml_types.h</b>	74
/home/nbock/Work/bml/src/C-interface/ <b>bml_types_private.h</b>	75
/home/nbock/Work/bml/src/C-interface/ <b>bml_utilities.h</b>	76
/home/nbock/Work/bml/src/C-interface/ <b>lapack.h</b>	??
/home/nbock/Work/bml/src/C-interface/ <b>macros.h</b>	78
/home/nbock/Work/bml/src/C-interface/ <b>typed.h</b>	??



# Chapter 11

## Module Documentation

### 11.1 Allocation and Deallocation Functions (C interface)

#### Functions

- void \* [bml\\_allocate\\_memory](#) (const size\_t size)
- void [bml\\_free\\_memory](#) (void \*ptr)
- void [bml\\_deallocate](#) ([bml\\_matrix\\_t](#) \*\*A)
- [bml\\_matrix\\_t](#) \* [bml\\_zero\\_matrix](#) (const [bml\\_matrix\\_type\\_t](#) matrix\_type, const [bml\\_matrix\\_precision\\_t](#) matrix\_precision, const int N, const int M)
- [bml\\_matrix\\_t](#) \* [bml\\_random\\_matrix](#) (const [bml\\_matrix\\_type\\_t](#) matrix\_type, const [bml\\_matrix\\_precision\\_t](#) matrix\_precision, const int N, const int M)
- [bml\\_matrix\\_t](#) \* [bml\\_identity\\_matrix](#) (const [bml\\_matrix\\_type\\_t](#) matrix\_type, const [bml\\_matrix\\_precision\\_t](#) matrix\_precision, const int N, const int M)

#### 11.1.1 Detailed Description

#### 11.1.2 Function Documentation

##### 11.1.2.1 void\* [bml\\_allocate\\_memory](#) ( const size\_t size )

Allocate and zero a chunk of memory.

#### Parameters

<i>size</i>	The size of the memory.
-------------	-------------------------

#### Returns

A pointer to the allocated chunk.

##### 11.1.2.2 void [bml\\_deallocate](#) ( [bml\\_matrix\\_t](#) \*\* A )

Deallocate a matrix.

#### Parameters

<i>A</i>	The matrix.
----------	-------------

Here is the call graph for this function:



#### 11.1.2.3 void bml\_free\_memory ( void \* *ptr* )

Deallocate a chunk of memory.

##### Parameters

<i>ptr</i>	A pointer to the previously allocated chunk.
------------	--

#### 11.1.2.4 **bml\_matrix\_t\*** bml\_identity\_matrix ( const bml\_matrix\_type\_t *matrix\_type*, const bml\_matrix\_precision\_t *matrix\_precision*, const int *N*, const int *M* )

Allocate the identity matrix.

Note that the matrix *A* will be newly allocated. The function does not check whether the matrix is already allocated.

##### Parameters

<i>matrix_type</i>	The matrix type.
<i>matrix_precision</i>	The precision of the matrix.
<i>N</i>	The matrix size.
<i>M</i>	The number of non-zeroes per row.

##### Returns

The matrix.

#### 11.1.2.5 **bml\_matrix\_t\*** bml\_random\_matrix ( const bml\_matrix\_type\_t *matrix\_type*, const bml\_matrix\_precision\_t *matrix\_precision*, const int *N*, const int *M* )

Allocate a random matrix.

Note that the matrix *A* will be newly allocated. The function does not check whether the matrix is already allocated.

##### Parameters

<i>matrix_type</i>	The matrix type.
<i>matrix_precision</i>	The precision of the matrix.
<i>N</i>	The matrix size.

$M$	The number of non-zeroes per row.
-----	-----------------------------------

**Returns**

The matrix.

11.1.2.6 `bml_matrix_t* bml_zero_matrix ( const bml_matrix_type_t matrix_type, const bml_matrix_precision_t matrix_precision, const int N, const int M )`

Allocate the zero matrix.

Note that the matrix  $A$  will be newly allocated. The function does not check whether the matrix is already allocated.

**Parameters**

<i>matrix_type</i>	The matrix type.
<i>matrix_precision</i>	The precision of the matrix.
$N$	The matrix size.
$M$	The number of non-zeroes per row.

**Returns**

The matrix.

## 11.2 Add Functions (C interface)

### Functions

- void `bml_add` (`bml_matrix_t` \*A, const `bml_matrix_t` \*B, const double alpha, const double beta, const double threshold)
- void `bml_add_identity` (`bml_matrix_t` \*A, const double beta, const double threshold)

### 11.2.1 Detailed Description

### 11.2.2 Function Documentation

11.2.2.1 void `bml_add` ( `bml_matrix_t` \* A, const `bml_matrix_t` \* B, const double *alpha*, const double *beta*, const double *threshold* )

Matrix addition.

$$A \leftarrow \alpha A + \beta B$$

Parameters

<i>A</i>	Matrix A
<i>B</i>	Matrix B
<i>alpha</i>	Scalar factor multiplied by A
<i>beta</i>	Scalar factor multiplied by B
<i>threshold</i>	Threshold for matrix addition

Here is the call graph for this function:



11.2.2.2 void `bml_add_identity` ( `bml_matrix_t` \* A, const double *beta*, const double *threshold* )

Matrix addition.

$$A \leftarrow A + \beta \text{Id}$$

Parameters

<i>A</i>	Matrix A
<i>beta</i>	Scalar factor multiplied by A
<i>threshold</i>	Threshold for matrix addition



Here is the call graph for this function:



## 11.3 Converting between Matrix Formats (C interface)

### Functions

- void \* [bml\\_export\\_to\\_dense](#) (const [bml\\_matrix\\_t](#) \*A, const [bml\\_dense\\_order\\_t](#) order)
- [bml\\_matrix\\_t](#) \* [bml\\_import\\_from\\_dense](#) (const [bml\\_matrix\\_type\\_t](#) matrix\_type, const [bml\\_matrix\\_precision\\_t](#) matrix\_precision, const [bml\\_dense\\_order\\_t](#) order, const int N, const void \*A, const double threshold, const int M)

### 11.3.1 Detailed Description

### 11.3.2 Function Documentation

#### 11.3.2.1 void\* [bml\\_export\\_to\\_dense](#) ( const [bml\\_matrix\\_t](#) \* A, const [bml\\_dense\\_order\\_t](#) order )

Export a bml matrix.

The returned pointer has to be typecase into the proper real type. If the bml matrix is a single precision matrix, then the following should be used:

```
float *A_dense = bml\_convert\_to\_dense(A_bml);
```

The matrix size can be queried with

```
int N = bml\_get\_size(A_bml);
```

#### Parameters

<i>A</i>	The bml matrix
<i>order</i>	The matrix element order

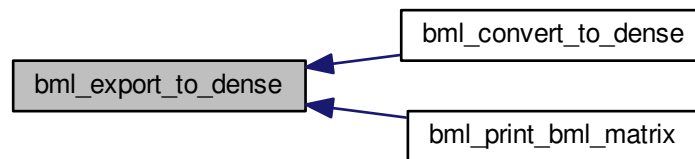
#### Returns

The dense matrix

Here is the call graph for this function:



Here is the caller graph for this function:



**11.3.2.2** `bml_matrix_t* bml_import_from_dense ( const bml_matrix_type_t matrix_type, const bml_matrix_precision_t matrix_precision, const bml_dense_order_t order, const int N, const void * A, const double threshold, const int M )`

Import a dense matrix.

Parameters

<i>matrix_type</i>	The matrix type
<i>matrix_precision</i>	The real precision
<i>order</i>	The dense matrix element order
<i>N</i>	The number of rows/columns
<i>A</i>	The dense matrix
<i>threshold</i>	The matrix element magnited threshold
<i>M</i>	The number of non-zeroes per row

Returns

The bml matrix

Here is the caller graph for this function:



## 11.4 Allocation and Deallocation Functions (Fortran interface)

### Functions

- subroutine, public `bml_allocate_m::bml_deallocate` (*a*)  
*Deallocate a matrix.*
- subroutine, public `bml_allocate_m::bml_zero_matrix` (*matrix\_type*, *matrix\_precision*, *n*, *m*, *a*)  
*Create the zero matrix.*
- subroutine, public `bml_allocate_m::bml_random_matrix` (*matrix\_type*, *matrix\_precision*, *n*, *m*, *a*)  
*Create a random matrix.*
- subroutine, public `bml_allocate_m::bml_identity_matrix` (*matrix\_type*, *matrix\_precision*, *n*, *m*, *a*)  
*Create the identity matrix.*

### 11.4.1 Detailed Description

### 11.4.2 Function Documentation

11.4.2.1 subroutine, public `bml_allocate_m::bml_deallocate` ( *type(bml\_matrix\_t)* *a* )

Deallocate a matrix.

Parameters

<i>a</i>	The matrix.
----------	-------------

11.4.2.2 subroutine, public `bml_allocate_m::bml_identity_matrix` ( *character(len=\*)*, *intent(in)* *matrix\_type*, *character(len=\*)*, *intent(in)* *matrix\_precision*, *integer*, *intent(in)* *n*, *integer*, *intent(in)* *m*, *type(bml\_matrix\_t)*, *intent(inout)* *a* )

Create the identity matrix.

Parameters

<i>matrix_type</i>	The matrix type.
<i>matrix_precision</i>	The precision of the matrix.
<i>n</i>	The matrix size.
<i>a</i>	The matrix.
<i>m</i>	The extra arg.

11.4.2.3 subroutine, public `bml_allocate_m::bml_random_matrix` ( *character(len=\*)*, *intent(in)* *matrix\_type*, *character(len=\*)*, *intent(in)* *matrix\_precision*, *integer*, *intent(in)* *n*, *integer*, *intent(in)* *m*, *type(bml\_matrix\_t)*, *intent(inout)* *a* )

Create a random matrix.

Parameters

<i>matrix_type</i>	The matrix type.
<i>matrix_precision</i>	The precision of the matrix.
<i>n</i>	The matrix size.
<i>a</i>	The matrix.
<i>m</i>	The extra arg.

11.4.2.4 subroutine, public bml\_allocate\_m::bml\_zero\_matrix ( character(len=\*), intent(in) *matrix\_type*, character(len=\*),  
intent(in) *matrix\_precision*, integer, intent(in) *n*, integer, intent(in) *m*, type(bml\_matrix\_t), intent(inout) *a* )

Create the zero matrix.

## Parameters

<i>matrix_type</i>	The matrix type.
<i>matrix_precision</i>	The precision of the matrix.
<i>n</i>	The matrix size.
<i>a</i>	The matrix.
<i>m</i>	The extra arg.

## 11.5 Add Functions (Fortran interface)

### 11.5.1 Detailed Description

## 11.6 Converting between Matrix Formats (Fortran interface)

### Functions

- subroutine `bml_convert_m::bml_convert_from_dense_double` (`matrix_type`, `a_dense`, `a`, `threshold`, `m`)  
*Convert a dense matrix into a bml matrix.*
- subroutine `bml_convert_m::bml_convert_from_dense_single_complex` (`matrix_type`, `a_dense`, `a`, `threshold`, `m`)  
*Convert a dense matrix into a bml matrix.*
- subroutine `bml_convert_m::bml_convert_from_dense_double_complex` (`matrix_type`, `a_dense`, `a`, `threshold`, `m`)  
*Convert a dense matrix into a bml matrix.*
- subroutine `bml_convert_m::bml_convert_to_dense_single` (`a`, `a_dense`)  
*Convert a matrix into a dense matrix.*
- subroutine `bml_convert_m::bml_convert_to_dense_double` (`a`, `a_dense`)  
*Convert a matrix into a dense matrix.*
- subroutine `bml_convert_m::bml_convert_to_dense_single_complex` (`a`, `a_dense`)  
*Convert a matrix into a dense matrix.*
- subroutine `bml_convert_m::bml_convert_to_dense_double_complex` (`a`, `a_dense`)  
*Convert a matrix into a dense matrix.*

### 11.6.1 Detailed Description

### 11.6.2 Function Documentation

- 11.6.2.1 subroutine `bml_convert_m::bml_convert_from_dense_double` ( `character(len=*)`, `intent(in)` `matrix_type`, `double precision`, `dimension(:, :)`, `intent(in)`, `target` `a_dense`, `type(bml_matrix_t)`, `intent(inout)` `a`, `double precision`, `intent(in)`, optional `threshold`, `integer`, `intent(in)`, optional `m` )

Convert a dense matrix into a bml matrix.

#### Parameters

<code>matrix_type</code>	The matrix type
<code>a_dense</code>	The dense matrix
<code>a</code>	The bml matrix
<code>threshold</code>	The matrix element magnited threshold
<code>m</code>	the extra arg

- 11.6.2.2 subroutine `bml_convert_m::bml_convert_from_dense_double_complex` ( `character(len=*)`, `intent(in)` `matrix_type`, `complex(kind(0.0d0))`, `dimension(:, :)`, `intent(in)`, `target` `a_dense`, `type(bml_matrix_t)`, `intent(inout)` `a`, `double precision`, `intent(in)`, optional `threshold`, `integer`, `intent(in)`, optional `m` )

Convert a dense matrix into a bml matrix.

#### Parameters

<code>matrix_type</code>	The matrix type
<code>a_dense</code>	The dense matrix
<code>a</code>	The bml matrix



<i>threshold</i>	The matrix element magnited threshold
<i>m</i>	the extra arg

11.6.2.3 subroutine `bml_convert_m::bml_convert_from_dense_single_complex` ( `character(len=*)`, `intent(in)` *matrix\_type*, `complex`, `dimension(:, :)`, `intent(in)`, `target` *a\_dense*, `type(bml_matrix_t)`, `intent(inout)` *a*, `double precision`, `intent(in)`, optional *threshold*, `integer`, `intent(in)`, optional *m* )

Convert a dense matrix into a bml matrix.

Parameters

<i>matrix_type</i>	The matrix type
<i>a_dense</i>	The dense matrix
<i>a</i>	The bml matrix
<i>threshold</i>	The matrix element magnited threshold
<i>m</i>	The extra arg

11.6.2.4 subroutine `bml_convert_m::bml_convert_to_dense_double` ( `type(bml_matrix_t)`, `intent(in)` *a*, `double precision`, `dimension(:, :)`, `intent(inout)`, allocatable *a\_dense* )

Convert a matrix into a dense matrix.

Parameters

<i>a</i>	The bml matrix
<i>a_dense</i>	The dense matrix

11.6.2.5 subroutine `bml_convert_m::bml_convert_to_dense_double_complex` ( `type(bml_matrix_t)`, `intent(in)` *a*, `complex(kind(0d0))`, `dimension(:, :)`, `intent(out)`, allocatable *a\_dense* )

Convert a matrix into a dense matrix.

Parameters

<i>a</i>	The bml matrix
<i>a_dense</i>	The dense matrix

11.6.2.6 subroutine `bml_convert_m::bml_convert_to_dense_single` ( `type(bml_matrix_t)`, `intent(in)` *a*, `real`, `dimension(:, :)`, `intent(inout)`, allocatable *a\_dense* )

Convert a matrix into a dense matrix.

Parameters

<i>a</i>	The bml matrix
<i>a_dense</i>	The dense matrix

11.6.2.7 subroutine `bml_convert_m::bml_convert_to_dense_single_complex` ( `type(bml_matrix_t)`, `intent(in)` *a*, `complex`, `dimension(:, :)`, `intent(out)`, allocatable *a\_dense* )

Convert a matrix into a dense matrix.

## Parameters

<i>a</i>	The bml matrix
<i>a_dense</i>	The dense matrix

## Chapter 12

# Namespace Documentation

### 12.1 bml Module Reference

Main matrix library module.

#### 12.1.1 Detailed Description

Main matrix library module.

Use this modules in order to use the library.

### 12.2 bml\_allocate\_m Module Reference

Matrix allocation functions.

#### Functions/Subroutines

- subroutine, public [bml\\_deallocate](#) (a)  
*Deallocate a matrix.*
- subroutine, public [bml\\_zero\\_matrix](#) (matrix\_type, matrix\_precision, n, m, a)  
*Create the zero matrix.*
- subroutine, public [bml\\_random\\_matrix](#) (matrix\_type, matrix\_precision, n, m, a)  
*Create a random matrix.*
- subroutine, public [bml\\_identity\\_matrix](#) (matrix\_type, matrix\_precision, n, m, a)  
*Create the identity matrix.*

#### 12.2.1 Detailed Description

Matrix allocation functions.

### 12.3 bml\_copy\_m Module Reference

Copy operations on matrices.

## Functions/Subroutines

- subroutine, public [bml\\_copy](#) (a, b)  
*Copy a matrix - result is a new matrix.*

### 12.3.1 Detailed Description

Copy operations on matrices.

### 12.3.2 Function/Subroutine Documentation

12.3.2.1 subroutine, public `bml_copy_m::bml_copy ( type(bml_matrix_t), intent(in) a, type(bml_matrix_t), intent(inout) b )`

Copy a matrix - result is a new matrix.

Parameters

<i>a</i>	Matrix to copy
<i>b</i>	The copy

## 12.4 bml\_diagonalize\_m Module Reference

Matrix diagonalization functions.

## Functions/Subroutines

- subroutine, public [bml\\_diagonalize](#) (a, eigenvalues, eigenvectors)  
*Diagonalize a matrix.*

### 12.4.1 Detailed Description

Matrix diagonalization functions.

### 12.4.2 Function/Subroutine Documentation

12.4.2.1 subroutine, public `bml_diagonalize_m::bml_diagonalize ( type(bml_matrix_t), intent(in) a, double precision, dimension(:), intent(inout), target eigenvalues, type(bml_matrix_t), intent(inout) eigenvectors )`

Diagonalize a matrix.

Parameters

<i>a</i>	The matrix.
<i>eigenvalues</i>	The corresponding eigenvalues.
<i>eigenvectors</i>	The set of eigenvectors.

## 12.5 bml\_error\_m Module Reference

A module for error handling in bml.

## Functions/Subroutines

- subroutine, public [bml\\_error](#) (file, line, message)  
*Common error handling of bml. This function writes out an error message and exits.*
- subroutine, public [bml\\_warning](#) (file, line, message)  
*Common error handling of bml. This function writes out a non-fatal warning message.*
- subroutine, public [bml\\_debug](#) (file, line, message)  
*Common error handling of bml. This function writes out a non-fatal warning message.*

### 12.5.1 Detailed Description

A module for error handling in bml.

#### Copyright

Los Alamos National Laboratory 2015

### 12.5.2 Function/Subroutine Documentation

**12.5.2.1** subroutine, public `bml_error_m::bml_debug ( character(len=*), intent(in) file, integer, intent(in) line, character(len=*), intent(in) message )`

Common error handling of bml. This function writes out a non-fatal warning message.

In the future one could imagine something more like exceptions, in which the error gets passed up the call stack.

#### Parameters

<i>file</i>	The filename in which the error occurred.
<i>line</i>	The line number in that file.
<i>message</i>	The error message.

**12.5.2.2** subroutine, public `bml_error_m::bml_error ( character(len=*), intent(in) file, integer, intent(in) line, character(len=*), intent(in) message )`

Common error handling of bml. This function writes out an error message and exits.

In the future one could imagine something more like exceptions, in which the error gets passed up the call stack.

#### Parameters

<i>file</i>	The filename in which the error occurred.
<i>line</i>	The line number in that file.
<i>message</i>	The error message.

**12.5.2.3** subroutine, public `bml_error_m::bml_warning ( character(len=*), intent(in) file, integer, intent(in) line, character(len=*), intent(in) message )`

Common error handling of bml. This function writes out a non-fatal warning message.

In the future one could imagine something more like exceptions, in which the error gets passed up the call stack.

## Parameters

<i>file</i>	The filename in which the error occurred.
<i>line</i>	The line number in that file.
<i>message</i>	The error message.

## 12.6 bml\_interface\_m Module Reference

Interface module.

### Functions/Subroutines

- integer function, public [get\\_enum\\_id](#) (type\_string)  
*Convert the matrix type and precisions strings into enum values.*

### Variables

- integer, parameter [bml\\_matrix\\_type\\_uninitialized\\_enum\\_id](#) = 0  
*The enum values of the C API. Keep this synchronized with the enum in [bml\\_types.h](#).*
- integer, parameter [bml\\_matrix\\_type\\_dense\\_enum\\_id](#) = 1  
*The enum values of the C API. Keep this synchronized with the enum in [bml\\_types.h](#).*
- integer, parameter [bml\\_matrix\\_type\\_ellpack\\_enum\\_id](#) = 2  
*The enum values of the C API. Keep this synchronized with the enum in [bml\\_types.h](#).*
- integer, parameter [bml\\_matrix\\_precision\\_uninitialized\\_id](#) = 0  
*The enum values of the C API. Keep this synchronized with the enum in [bml\\_types.h](#).*
- integer, parameter [bml\\_matrix\\_precision\\_single\\_real\\_enum\\_id](#) = 1  
*The enum values of the C API. Keep this synchronized with the enum in [bml\\_types.h](#).*
- integer, parameter [bml\\_matrix\\_precision\\_double\\_real\\_enum\\_id](#) = 2  
*The enum values of the C API. Keep this synchronized with the enum in [bml\\_types.h](#).*
- integer, parameter [bml\\_matrix\\_precision\\_single\\_complex\\_enum\\_id](#) = 3  
*The enum values of the C API. Keep this synchronized with the enum in [bml\\_types.h](#).*
- integer, parameter [bml\\_matrix\\_precision\\_double\\_complex\\_enum\\_id](#) = 4  
*The enum values of the C API. Keep this synchronized with the enum in [bml\\_types.h](#).*
- integer, parameter, public [bml\\_dense\\_column\\_major](#) = 1  
*The dense matrix element order.*

### 12.6.1 Detailed Description

Interface module.

### 12.6.2 Function/Subroutine Documentation

#### 12.6.2.1 integer function, public [bml\\_interface\\_m::get\\_enum\\_id](#) ( character(len=\*), intent(in) type\_string )

Convert the matrix type and precisions strings into enum values.

## Parameters

<i>type_string</i>	The string used in the Fortran API to identify the matrix type and precision.
--------------------	---

## Returns

The corresponding integer value matching the enum values in `bml_matrix_types_t` and `bml_matrix_precision_t`.

## 12.6.3 Variable Documentation

12.6.3.1 integer, parameter `bml_interface_m::bml_matrix_precision_double_complex_enum_id = 4`

The enum values of the C API. Keep this synchronized with the enum in [bml\\_types.h](#).  
Matrix precision is double complex.

12.6.3.2 integer, parameter `bml_interface_m::bml_matrix_precision_double_real_enum_id = 2`

The enum values of the C API. Keep this synchronized with the enum in [bml\\_types.h](#).  
Matrix precision is double real.

12.6.3.3 integer, parameter `bml_interface_m::bml_matrix_precision_single_complex_enum_id = 3`

The enum values of the C API. Keep this synchronized with the enum in [bml\\_types.h](#).  
Matrix precision is single complex.

12.6.3.4 integer, parameter `bml_interface_m::bml_matrix_precision_single_real_enum_id = 1`

The enum values of the C API. Keep this synchronized with the enum in [bml\\_types.h](#).  
Matrix precision is single real.

12.6.3.5 integer, parameter `bml_interface_m::bml_matrix_precision_uninitialized_id = 0`

The enum values of the C API. Keep this synchronized with the enum in [bml\\_types.h](#).  
Matrix precision is uninitialized.

12.6.3.6 integer, parameter `bml_interface_m::bml_matrix_type_dense_enum_id = 1`

The enum values of the C API. Keep this synchronized with the enum in [bml\\_types.h](#).  
Matrix type is dense.

12.6.3.7 integer, parameter `bml_interface_m::bml_matrix_type_ellpack_enum_id = 2`

The enum values of the C API. Keep this synchronized with the enum in [bml\\_types.h](#).  
Matrix type is ellpack.

12.6.3.8 integer, parameter `bml_interface_m::bml_matrix_type_uninitialized_enum_id = 0`

The enum values of the C API. Keep this synchronized with the enum in [bml\\_types.h](#).

Matrix type is uninitialized.

## 12.7 bml\_introspection\_m Module Reference

Introspection procedures.

### Functions/Subroutines

- integer function, public [bml\\_get\\_n](#) (a)  
*Return the matrix size.*
- integer function, public [bml\\_get\\_row\\_bandwidth](#) (a, i)  
*Get the bandwidth of non-zero elements in a given row.*
- integer function, public [bml\\_get\\_bandwidth](#) (a)  
*Get the bandwidth of non-zero elements of a matrix.*

### 12.7.1 Detailed Description

Introspection procedures.

### 12.7.2 Function/Subroutine Documentation

12.7.2.1 integer function, public `bml_introspection_m::bml_get_bandwidth ( type(bml_matrix_t), intent(in) a )`

Get the bandwidth of non-zero elements of a matrix.

Parameters

<i>a</i>	The matrix.
----------	-------------

Returns

The bandwidth of non-zero elements (bandwidth) of the matrix.

12.7.2.2 integer function, public `bml_introspection_m::bml_get_n ( type(bml_matrix_t), intent(in) a )`

Return the matrix size.

Parameters

<i>a</i>	The matrix.
----------	-------------

Returns

The matrix size.

12.7.2.3 integer function, public `bml_introspection_m::bml_get_row_bandwidth ( type(bml_matrix_t), intent(in) a, integer, intent(in) i )`

Get the bandwidth of non-zero elements in a given row.



## Parameters

<i>a</i>	The matrix.
<i>i</i>	The row.

## Returns

The bandwidth of non-zero elements (bandwidth) on that row.

## 12.8 bml\_multiply\_m Module Reference

Matrix multiplication.

### Functions/Subroutines

- subroutine, public [bml\\_multiply](#) (a, b, c, alpha, beta)

*Multiply two matrices.*

#### 12.8.1 Detailed Description

Matrix multiplication.

#### 12.8.2 Function/Subroutine Documentation

12.8.2.1 subroutine, public bml\_multiply\_m::bml\_multiply ( type(bml\_matrix\_t), intent(in) *a*, type(bml\_matrix\_t), intent(in) *b*, type(bml\_matrix\_t), intent(inout) *c*, double precision, intent(in), optional *alpha*, double precision, intent(in), optional *beta* )

Multiply two matrices.

$$C \leftarrow \alpha A \times B + \beta C$$

The optional scaling factors  $\alpha$  and  $\beta$  default to  $\alpha = 1$  and  $\beta = 0$ .

## Parameters

<i>a</i>	Matrix <i>A</i> .
<i>b</i>	Matrix <i>B</i> .
<i>c</i>	Matrix <i>C</i> .
<i>alpha</i>	The factor $\alpha$ .
<i>beta</i>	The factor $\beta$ .

## 12.9 bml\_scale\_m Module Reference

Matrix scaling for matrices.

### Functions/Subroutines

- subroutine [scale\\_two](#) (alpha, a, c)

*Scale a bml matrix.*

### 12.9.1 Detailed Description

Matrix scaling for matrices.

### 12.9.2 Function/Subroutine Documentation

12.9.2.1 subroutine `bml_scale_m::scale_two` ( double precision, intent(in) *alpha*, type(`bml_matrix_t`), intent(in) *a*, type(`bml_matrix_t`), intent(inout) *c* )

Scale a bml matrix.

$$C \leftarrow \alpha A$$

Parameters

<i>alpha</i>	The factor
<i>a</i>	The matrix
<i>c</i>	The matrix

## 12.10 `bml_trace_m` Module Reference

Matrix trace.

### Functions/Subroutines

- double precision function, public [bml\\_trace](#) (*a*)  
*Calculate the trace of a matrix.*

### 12.10.1 Detailed Description

Matrix trace.

### 12.10.2 Function/Subroutine Documentation

12.10.2.1 double precision function, public `bml_trace_m::bml_trace` ( class(`bml_matrix_t`), intent(in) *a* )

Calculate the trace of a matrix.

$$\leftarrow \text{Tr}[A]$$

Parameters

<i>a</i>	The matrix.
----------	-------------

## 12.11 `bml_transpose_m` Module Reference

Transpose functions.

### Functions/Subroutines

- subroutine, public [bml\\_transpose](#) (*a*, *a\_t*)  
*Return the transpose of a matrix.*

### 12.11.1 Detailed Description

Transpose functions.

### 12.11.2 Function/Subroutine Documentation

12.11.2.1 subroutine, public `bml_transpose_m::bml_transpose` ( `type(bml_matrix_t), intent(in) a`, `type(bml_matrix_t), intent(inout) a_t` )

Return the transpose of a matrix.

Parameters

<code>a</code>	The matrix.
<code>a_t</code>	The transpose.

## 12.12 bml\_types\_m Module Reference

The basic bml types.

### Data Types

- type `bml_matrix_t`  
*The bml matrix type.*
- type `bml_vector_t`  
*The bml vector type.*

### Variables

- character(len=\*), parameter `bml_matrix_dense` = "dense"  
*The bml-dense matrix type identifier.*
- character(len=\*), parameter `bml_matrix_ellpack` = "ellpack"  
*The bml-ellpack matrix type identifier.*
- character(len=\*), parameter `bml_precision_single_real` = "single\_real"  
*The single precision identifier.*
- character(len=\*), parameter `bml_precision_double_real` = "double\_real"  
*The double-precision identifier.*
- character(len=\*), parameter `bml_precision_single_complex` = "single\_complex"  
*The single precision identifier.*
- character(len=\*), parameter `bml_precision_double_complex` = "double\_complex"  
*The double-precision identifier.*

### 12.12.1 Detailed Description

The basic bml types.

## 12.13 bml\_utilities\_m Module Reference

Utility matrix functions.

## Functions/Subroutines

- subroutine [bml\\_print\\_bml\\_vector](#) (tag, v, i\_l, i\_u)  
*Print a bml vector.*

### 12.13.1 Detailed Description

Utility matrix functions.

### 12.13.2 Function/Subroutine Documentation

12.13.2.1 subroutine `bml_utilities_m::bml_print_bml_vector` ( character(len=\*) intent(in) *tag*, type(`bml_vector_t`) intent(in), target *v*, integer intent(in) *i\_l*, integer intent(in) *i\_u* )

Print a bml vector.

Parameters

<i>tag</i>	A string to print before the matrix.
<i>v</i>	The vector.
<i>i_l</i>	The lower row bound.
<i>i_u</i>	The upper row bound.

## 12.14 bml\_utilities\_matrix\_type\_m Module Reference

Utility matrix functions.

### 12.14.1 Detailed Description

Utility matrix functions.

## Chapter 13

# Class Documentation

### 13.1 `bml_types_m::bml_matrix_t` Type Reference

The bml matrix type.

#### Public Attributes

- `type(c_ptr) ptr = C_NULL_PTR`  
*The C pointer to the matrix.*

#### 13.1.1 Detailed Description

The bml matrix type.

The documentation for this type was generated from the following file:

- `/home/nbock/Work/bml/src/Fortran-interface/bml_types_m.F90`

### 13.2 `bml_types_m::bml_vector_t` Type Reference

The bml vector type.

#### Public Attributes

- `type(c_ptr) ptr = C_NULL_PTR`  
*The C pointer to the vector.*

#### 13.2.1 Detailed Description

The bml vector type.

The documentation for this type was generated from the following file:

- `/home/nbock/Work/bml/src/Fortran-interface/bml_types_m.F90`



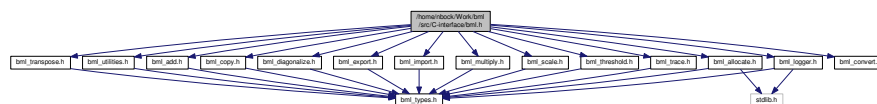
## Chapter 14

# File Documentation

### 14.1 /home/nbock/Work/bml/src/C-interface/bml.h File Reference

```
#include "bml_add.h"  
#include "bml_allocate.h"  
#include "bml_convert.h"  
#include "bml_copy.h"  
#include "bml_diagonalize.h"  
#include "bml_export.h"  
#include "bml_import.h"  
#include "bml_logger.h"  
#include "bml_multiply.h"  
#include "bml_scale.h"  
#include "bml_threshold.h"  
#include "bml_trace.h"  
#include "bml_transpose.h"  
#include "bml_utilities.h"
```

Include dependency graph for bml.h:



#### 14.1.1 Detailed Description

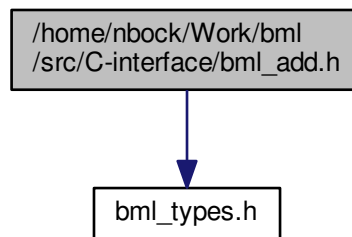
## Copyright

Los Alamos National Laboratory 2015

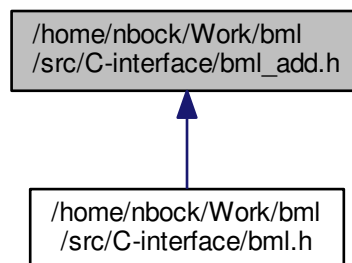
## 14.2 /home/nbock/Work/bml/src/C-interface/bml\_add.h File Reference

```
#include "bml_types.h"
```

Include dependency graph for bml\_add.h:



This graph shows which files directly or indirectly include this file:



### Functions

- void `bml_add` (`bml_matrix_t` \*A, const `bml_matrix_t` \*B, const double alpha, const double beta, const double threshold)
- void `bml_add_identity` (`bml_matrix_t` \*A, const double beta, const double threshold)

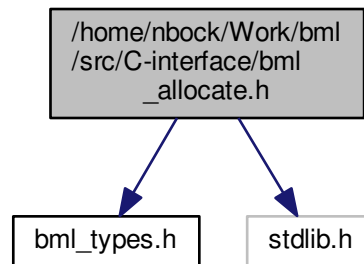
## 14.3 /home/nbock/Work/bml/src/C-interface/bml\_allocate.h File Reference

```
#include "bml_types.h"
```

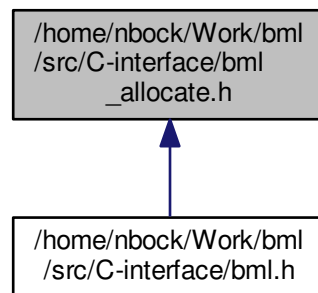
```
#include <stdlib.h>
```



Include dependency graph for bml\_allocate.h:



This graph shows which files directly or indirectly include this file:

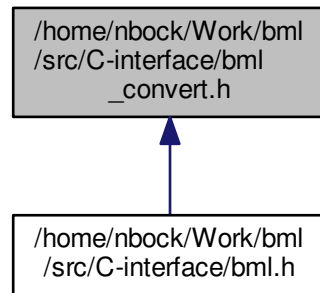


## Functions

- void \* [bml\\_allocate\\_memory](#) (const size\_t s)
- void [bml\\_free\\_memory](#) (void \*ptr)
- void [bml\\_deallocate](#) (bml\_matrix\_t \*\*A)
- bml\_matrix\_t \* [bml\\_zero\\_matrix](#) (const bml\_matrix\_type\_t matrix\_type, const bml\_matrix\_precision\_t matrix\_precision, const int N, const int M)
- bml\_matrix\_t \* [bml\\_random\\_matrix](#) (const bml\_matrix\_type\_t matrix\_type, const bml\_matrix\_precision\_t matrix\_precision, const int N, const int M)
- bml\_matrix\_t \* [bml\\_identity\\_matrix](#) (const bml\_matrix\_type\_t matrix\_type, const bml\_matrix\_precision\_t matrix\_precision, const int N, const int M)

## 14.4 /home/nbock/Work/bml/src/C-interface/bml\_convert.h File Reference

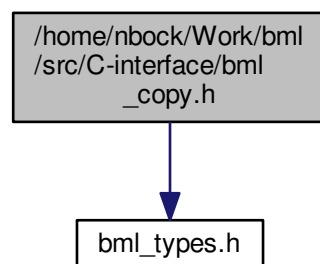
This graph shows which files directly or indirectly include this file:



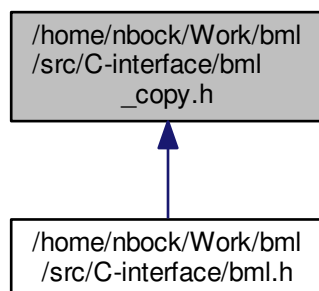
## 14.5 /home/nbock/Work/bml/src/C-interface/bml\_copy.h File Reference

```
#include "bml_types.h"
```

Include dependency graph for `bml_copy.h`:



This graph shows which files directly or indirectly include this file:



## Functions

- [bml\\_matrix\\_t \\* bml\\_copy\\_new](#) (const [bml\\_matrix\\_t](#) \*A)
- void [bml\\_copy](#) (const [bml\\_matrix\\_t](#) \*A, [bml\\_matrix\\_t](#) \*B)

### 14.5.1 Function Documentation

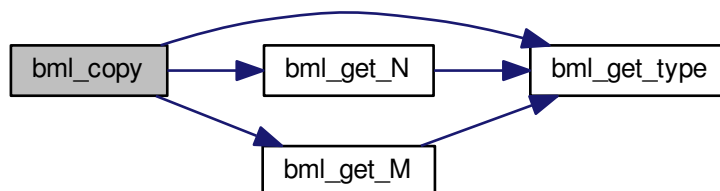
#### 14.5.1.1 void bml\_copy ( const bml\_matrix\_t \* A, bml\_matrix\_t \* B )

Copy a matrix.

##### Parameters

<i>A</i>	Matrix to copy
<i>B</i>	Copy of Matrix A

Here is the call graph for this function:



#### 14.5.1.2 bml\_matrix\_t\* bml\_copy\_new ( const bml\_matrix\_t \* A )

Copy a matrix - result is a new matrix.

## Parameters

<i>A</i>	Matrix to copy
----------	----------------

## Returns

A Copy of A

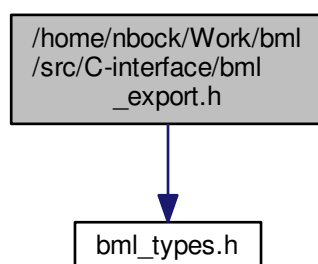
Here is the call graph for this function:



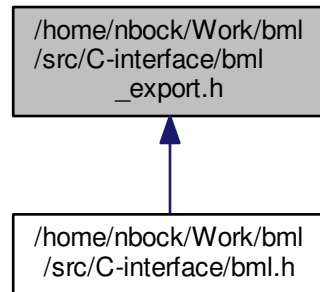
## 14.6 /home/nbock/Work/bml/src/C-interface/bml\_export.h File Reference

```
#include "bml_types.h"
```

Include dependency graph for `bml_export.h`:



This graph shows which files directly or indirectly include this file:



## Functions

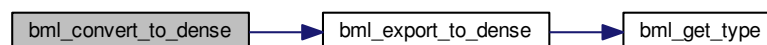
- void \* [bml\\_convert\\_to\\_dense](#) (const [bml\\_matrix\\_t](#) \*A, const [bml\\_dense\\_order\\_t](#) order)
- void \* [bml\\_export\\_to\\_dense](#) (const [bml\\_matrix\\_t](#) \*A, const [bml\\_dense\\_order\\_t](#) order)

### 14.6.1 Function Documentation

14.6.1.1 void\* [bml\\_convert\\_to\\_dense](#) ( const [bml\\_matrix\\_t](#) \* A, const [bml\\_dense\\_order\\_t](#) order )

**Deprecated** Deprecated API.

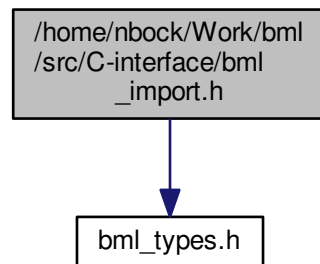
Here is the call graph for this function:



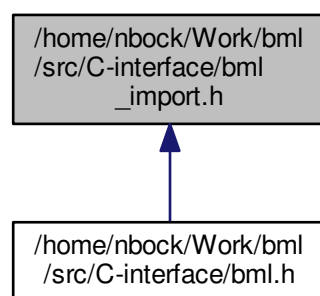
## 14.7 /home/nbock/Work/bml/src/C-interface/bml\_import.h File Reference

```
#include "bml_types.h"
```

Include dependency graph for `bml_import.h`:



This graph shows which files directly or indirectly include this file:



## Functions

- `bml_matrix_t * bml_import_from_dense` (const `bml_matrix_type_t` *matrix\_type*, const `bml_matrix_precision_t` *matrix\_precision*, const `bml_dense_order_t` *order*, const int *N*, const void \**A*, const double *threshold*, const int *M*)
- `bml_matrix_t * bml_convert_from_dense` (const `bml_matrix_type_t` *matrix\_type*, const `bml_matrix_precision_t` *matrix\_precision*, const `bml_dense_order_t` *order*, const int *N*, const void \**A*, const double *threshold*, const int *M*)

### 14.7.1 Function Documentation

14.7.1.1 `bml_matrix_t* bml_convert_from_dense` ( const `bml_matrix_type_t` *matrix\_type*, const `bml_matrix_precision_t` *matrix\_precision*, const `bml_dense_order_t` *order*, const int *N*, const void \* *A*, const double *threshold*, const int *M* )

**Deprecated** Deprecated API.

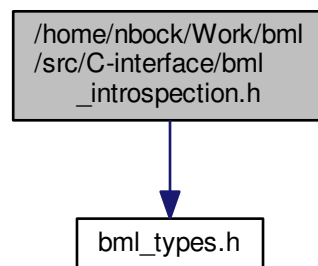
Here is the call graph for this function:



## 14.8 /home/nbock/Work/bml/src/C-interface/bml\_introspection.h File Reference

```
#include "bml_types.h"
```

Include dependency graph for bml\_introspection.h:



### Functions

- [bml\\_matrix\\_type\\_t bml\\_get\\_type](#) (const [bml\\_matrix\\_t](#) \*A)
- [bml\\_matrix\\_precision\\_t bml\\_get\\_precision](#) (const [bml\\_matrix\\_t](#) \*A)
- [int bml\\_get\\_N](#) (const [bml\\_matrix\\_t](#) \*A)
- [int bml\\_get\\_M](#) (const [bml\\_matrix\\_t](#) \*A)
- [int bml\\_get\\_row\\_bandwidth](#) (const [bml\\_matrix\\_t](#) \*A, const int i)
- [int bml\\_get\\_bandwidth](#) (const [bml\\_matrix\\_t](#) \*A)

### 14.8.1 Function Documentation

#### 14.8.1.1 [int bml\\_get\\_bandwidth](#) ( const [bml\\_matrix\\_t](#) \* A )

Return the bandwidth of a matrix.

**Parameters**

---

<i>A</i>	The bml matrix.
----------	-----------------

**Returns**

The bandwidth of row *i*.

Here is the call graph for this function:

**14.8.1.2 int bml\_get\_M ( const bml\_matrix\_t \* *A* )**

Return the matrix parameter *M*.

**Parameters**

<i>A</i>	The matrix.
----------	-------------

**Returns**

The matrix parameter *M*.

Here is the call graph for this function:



Here is the caller graph for this function:





14.8.1.3 `int bml_get_N ( const bml_matrix_t * A )`

Return the matrix size.

## Parameters

<i>A</i>	The matrix.
----------	-------------

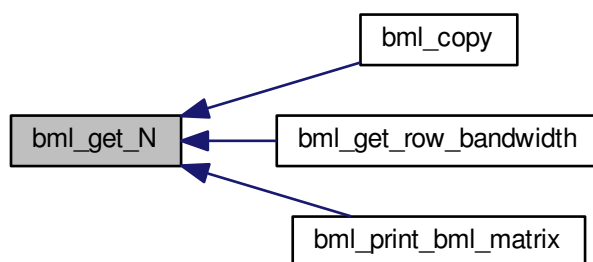
## Returns

The matrix size.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.8.1.4 `bml_matrix_precision_t bml_get_precision ( const bml_matrix_t * A )`

Return the matrix precision.

## Parameters

<i>A</i>	The matrix.
----------	-------------

**Returns**

The matrix precision.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 14.8.1.5 int bml\_get\_row\_bandwidth ( const bml\_matrix\_t \* A, const int i )

Return the bandwidth of a row in the matrix.

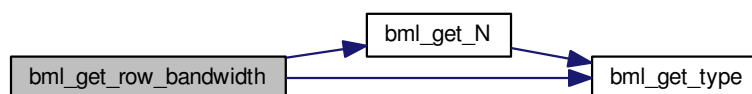
**Parameters**

<i>A</i>	The bml matrix.
<i>i</i>	The row index.

**Returns**

The bandwidth of row *i*.

Here is the call graph for this function:



14.8.1.6 `bml_matrix_type_t bml_get_type ( const bml_matrix_t * A )`

Returns the matrix type.

If the matrix is not initialized yet, a type of "uninitialized" is returned.

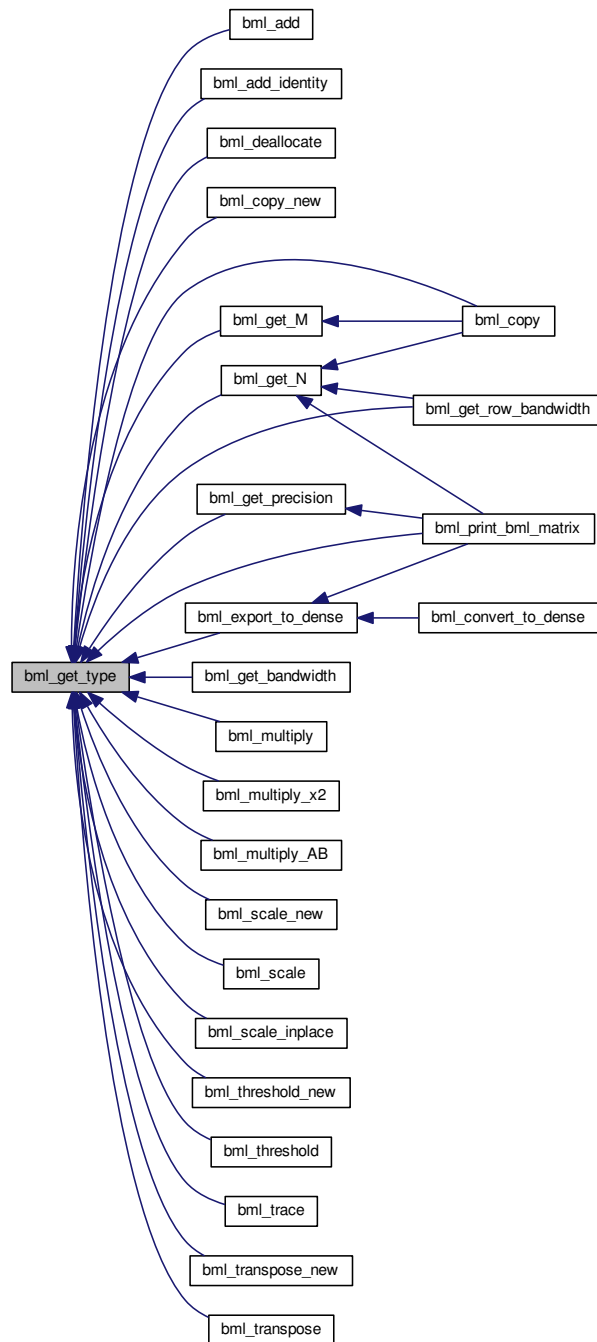
## Parameters

<i>A</i>	The matrix.
----------	-------------

## Returns

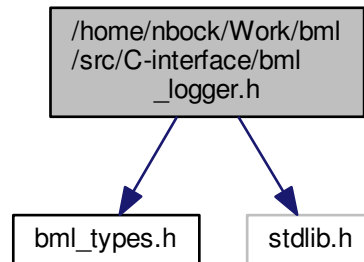
The matrix type.

Here is the caller graph for this function:

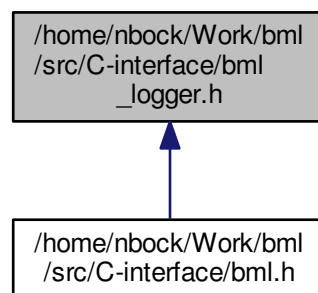


## 14.9 /home/nbock/Work/bml/src/C-interface/bml\_logger.h File Reference

```
#include "bml_types.h"
#include <stdlib.h>
Include dependency graph for bml_logger.h:
```



This graph shows which files directly or indirectly include this file:



### Macros

- `#define LOG_DEBUG(format, ...) bml_log_location(BML_LOG_DEBUG, __FILE__, __LINE__, format, ##__VA_ARGS__)`
- `#define LOG_INFO(format, ...) bml_log(BML_LOG_INFO, format, ##__VA_ARGS__)`
- `#define LOG_WARN(format, ...) bml_log_location(BML_LOG_WARNING, __FILE__, __LINE__, format, ##__VA_ARGS__)`
- `#define LOG_ERROR(format, ...) bml_log_location(BML_LOG_ERROR, __FILE__, __LINE__, format, ##__VA_ARGS__)`

### Enumerations

- enum `bml_log_level_t` { `BML_LOG_DEBUG`, `BML_LOG_INFO`, `BML_LOG_WARNING`, `BML_LOG_ERROR` }

## Functions

- void [bml\\_log](#) (const [bml\\_log\\_level\\_t](#) log\_level, const char \*format,...)
- void [bml\\_log\\_location](#) (const [bml\\_log\\_level\\_t](#) log\_level, const char \*filename, const int linenumber, const char \*format,...)

### 14.9.1 Macro Definition Documentation

14.9.1.1 `#define LOG_DEBUG( format, ... ) bml_log_location(BML_LOG_DEBUG, __FILE__, __LINE__, format, ##__VA_ARGS__)`

Convenience macro to write a BML\_LOG\_DEBUG level message.

14.9.1.2 `#define LOG_ERROR( format, ... ) bml_log_location(BML_LOG_ERROR, __FILE__, __LINE__, format, ##__VA_ARGS__)`

Convenience macro to write a BML\_LOG\_ERROR level message.

14.9.1.3 `#define LOG_INFO( format, ... ) bml_log(BML_LOG_INFO, format, ##__VA_ARGS__)`

Convenience macro to write a BML\_LOG\_INFO level message.

14.9.1.4 `#define LOG_WARN( format, ... ) bml_log_location(BML_LOG_WARNING, __FILE__, __LINE__, format, ##__VA_ARGS__)`

Convenience macro to write a BML\_LOG\_WARNING level message.

### 14.9.2 Enumeration Type Documentation

14.9.2.1 `enum bml_log_level_t`

The log-levels.

Enumerator

**BML\_LOG\_DEBUG** Debugging messages.

**BML\_LOG\_INFO** Info messages.

**BML\_LOG\_WARNING** Warning messages.

**BML\_LOG\_ERROR** Error messages.

### 14.9.3 Function Documentation

14.9.3.1 `void bml_log ( const bml_log_level_t log_level, const char * format, ... )`

Log a message.

Parameters

<i>log_level</i>	The log level.
------------------	----------------

<i>format</i>	The format (as in printf()).
---------------	------------------------------

14.9.3.2 void bml\_log\_location ( const bml\_log\_level\_t *log\_level*, const char \* *filename*, const int *linenumber*, const char \* *format*, ... )

Log a message with location, i.e. filename and linenumber..

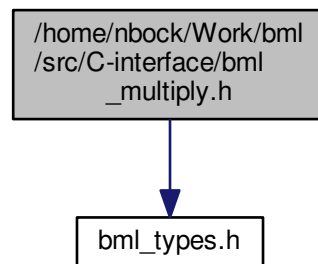
Parameters

<i>log_level</i>	The log level.
<i>filename</i>	The filename to log.
<i>linenumber</i>	The linenumber.
<i>format</i>	The format (as in printf()).

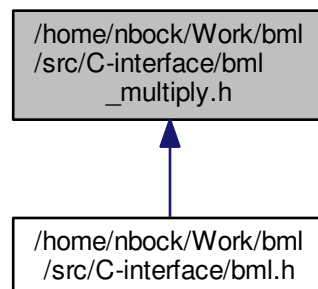
## 14.10 /home/nbock/Work/bml/src/C-interface/bml\_multiply.h File Reference

```
#include "bml_types.h"
```

Include dependency graph for bml\_multiply.h:



This graph shows which files directly or indirectly include this file:





## Functions

- void `bml_multiply` (const `bml_matrix_t` \*A, const `bml_matrix_t` \*B, `bml_matrix_t` \*C, const double alpha, const double beta, const double threshold)
- void `bml_multiply_x2` (const `bml_matrix_t` \*X, `bml_matrix_t` \*X2, const double threshold)
- void `bml_multiply_AB` (const `bml_matrix_t` \*A, const `bml_matrix_t` \*B, `bml_matrix_t` \*C, const double threshold)

### 14.10.1 Function Documentation

14.10.1.1 void `bml_multiply` ( const `bml_matrix_t` \* A, const `bml_matrix_t` \* B, `bml_matrix_t` \* C, const double *alpha*, const double *beta*, const double *threshold* )

Matrix multiply.

$C = \alpha * A * B + \beta * C$

Parameters

<i>A</i>	Matrix A
<i>B</i>	Matrix B
<i>C</i>	Matrix C
<i>alpha</i>	Scalar factor that multiplies A * B
<i>beta</i>	Scalar factor that multiplies C
<i>threshold</i>	Threshold for multiplication

Here is the call graph for this function:



14.10.1.2 void `bml_multiply_AB` ( const `bml_matrix_t` \* A, const `bml_matrix_t` \* B, `bml_matrix_t` \* C, const double *threshold* )

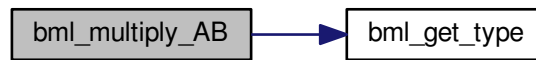
Matrix multiply.

$C = A * B$

Parameters

<i>A</i>	Matrix A
<i>B</i>	Matrix B
<i>C</i>	Matrix C
<i>threshold</i>	Threshold for multiplication

Here is the call graph for this function:



14.10.1.3 void bml\_multiply\_x2 ( const bml\_matrix\_t \* X, bml\_matrix\_t \* X2, const double *threshold* )

Matrix multiply.

$$X^2 \leftarrow X X$$

Parameters

<i>X</i>	Matrix X
<i>X2</i>	MatrixX2
<i>threshold</i>	Threshold for multiplication

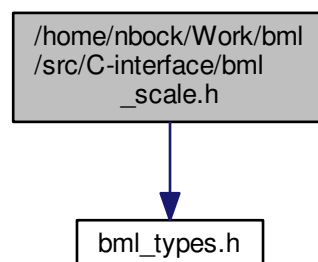
Here is the call graph for this function:



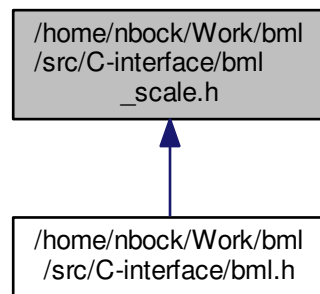
## 14.11 /home/nbock/Work/bml/src/C-interface/bml\_scale.h File Reference

```
#include "bml_types.h"
```

Include dependency graph for bml\_scale.h:



This graph shows which files directly or indirectly include this file:



## Functions

- `bml_matrix_t * bml_scale_new` (const double *scale\_factor*, const `bml_matrix_t *A`)
- void `bml_scale` (const double *scale\_factor*, const `bml_matrix_t *A`, `bml_matrix_t *B`)
- void `bml_scale_inplace` (const double *scale\_factor*, `bml_matrix_t *A`)

### 14.11.1 Function Documentation

14.11.1.1 void `bml_scale` ( const double *scale\_factor*, const `bml_matrix_t * A`, `bml_matrix_t * B` )

Scale a matrix - resulting matrix exists.

#### Parameters

<i>scale_factor</i>	Scale factor for A
<i>A</i>	Matrix to scale
<i>B</i>	Scaled Matrix

Here is the call graph for this function:



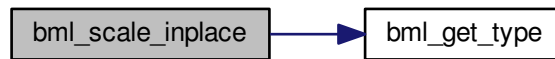
14.11.1.2 void `bml_scale_inplace` ( const double *scale\_factor*, `bml_matrix_t * A` )

Scale a matrix in place, i.e. the matrix is overwritten.

## Parameters

<i>scale_factor</i>	Scale factor for A
<i>A</i>	[inout] Matrix to scale

Here is the call graph for this function:



#### 14.11.1.3 `bml_matrix_t* bml_scale_new ( const double scale_factor, const bml_matrix_t * A )`

Scale a matrix - resulting matrix is new.

## Parameters

<i>scale_factor</i>	Scale factor for A
<i>A</i>	Matrix to scale

## Returns

A Scaled Copy of A

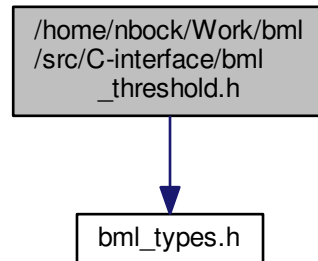
Here is the call graph for this function:



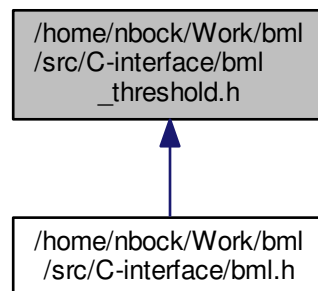
## 14.12 /home/nbock/Work/bml/src/C-interface/bml\_threshold.h File Reference

```
#include "bml_types.h"
```

Include dependency graph for bml\_threshold.h:



This graph shows which files directly or indirectly include this file:



### Functions

- [bml\\_matrix\\_t \\* bml\\_threshold\\_new](#) (const [bml\\_matrix\\_t](#) \*A, const double threshold)
- void [bml\\_threshold](#) ([bml\\_matrix\\_t](#) \*A, const double threshold)

#### 14.12.1 Function Documentation

14.12.1.1 void [bml\\_threshold](#) ( [bml\\_matrix\\_t](#) \* A, const double *threshold* )

Threshold matrix.

## Parameters

<i>A</i>	Matrix to be thresholded
<i>threshold</i>	Threshold value

## Returns

Thresholded A

Here is the call graph for this function:



#### 14.12.1.2 `bml_matrix_t* bml_threshold_new ( const bml_matrix_t * A, const double threshold )`

Threshold matrix.

## Parameters

<i>A</i>	Matrix to be thresholded
<i>threshold</i>	Threshold value

## Returns

Thresholded A

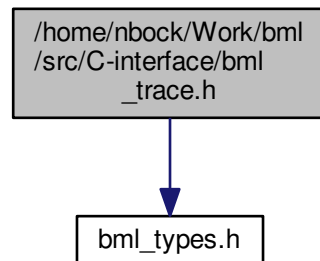
Here is the call graph for this function:



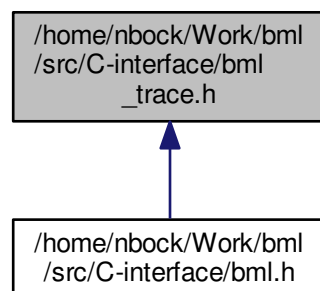
## 14.13 /home/nbock/Work/bml/src/C-interface/bml\_trace.h File Reference

```
#include "bml_types.h"
```

Include dependency graph for bml\_trace.h:



This graph shows which files directly or indirectly include this file:



### Functions

- double `bml_trace` (const `bml_matrix_t` \*A)

#### 14.13.1 Function Documentation

14.13.1.1 double `bml_trace` ( const `bml_matrix_t` \* A )

Calculate trace of a matrix.

## Parameters

<i>A</i>	Matrix to calculate trace for
----------	-------------------------------

## Returns

Trace of *A*

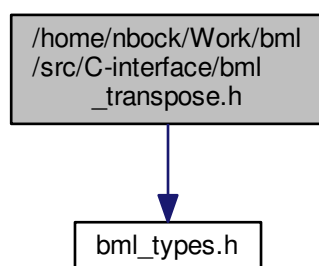
Here is the call graph for this function:



## 14.14 /home/nbock/Work/bml/src/C-interface/bml\_transpose.h File Reference

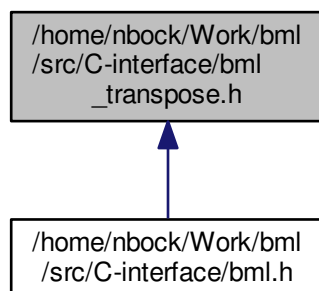
```
#include "bml_types.h"
```

Include dependency graph for `bml_transpose.h`:





This graph shows which files directly or indirectly include this file:



## Functions

- `bml_matrix_t * bml_transpose_new` (const `bml_matrix_t` \*A)
- void `bml_transpose` (const `bml_matrix_t` \*A)

### 14.14.1 Function Documentation

#### 14.14.1.1 void bml\_transpose ( const bml\_matrix\_t \* A )

Transpose matrix.

##### Parameters

A	Matrix to be transposed
---	-------------------------

##### Returns

Transposed A

Here is the call graph for this function:



#### 14.14.1.2 bml\_matrix\_t\* bml\_transpose\_new ( const bml\_matrix\_t \* A )

Transpose matrix.

## Parameters

<b>A</b>	Matrix to be transposed
----------	-------------------------

## Returns

Transposed A

Here is the call graph for this function:



## 14.15 /home/nbock/Work/bml/src/C-interface/bml\_types.h File Reference

This graph shows which files directly or indirectly include this file:



### Typedefs

- typedef void [bml\\_vector\\_t](#)
- typedef void [bml\\_matrix\\_t](#)

### Enumerations

- enum [bml\\_matrix\\_type\\_t](#) { [type\\_uninitialized](#), [dense](#), [ellpack](#), [csr](#) }
- enum [bml\\_matrix\\_precision\\_t](#) { [precision\\_uninitialized](#), [single\\_real](#), [double\\_real](#), [single\\_complex](#), [double\\_complex](#) }
- enum [bml\\_dense\\_order\\_t](#) { [dense\\_row\\_major](#), [dense\\_column\\_major](#) }

#### 14.15.1 Typedef Documentation

##### 14.15.1.1 typedef void [bml\\_matrix\\_t](#)

The matrix type.

##### 14.15.1.2 typedef void [bml\\_vector\\_t](#)

The vector type.

## 14.15.2 Enumeration Type Documentation

### 14.15.2.1 enum bml\_dense\_order\_t

The supported dense matrix elements orderings.

Enumerator

***dense\_row\_major*** row-major order.

***dense\_column\_major*** column-major order.

### 14.15.2.2 enum bml\_matrix\_precision\_t

The supported real precisions.

Enumerator

***precision\_uninitialized*** The matrix is not initialized.

***single\_real*** Matrix data is stored in single precision (float).

***double\_real*** Matrix data is stored in double precision (double).

***single\_complex*** Matrix data is stored in single-complex precision (float).

***double\_complex*** Matrix data is stored in double-complex precision (double).

### 14.15.2.3 enum bml\_matrix\_type\_t

The supported matrix types.

Enumerator

***type\_uninitialized*** The matrix is not initialized.

***dense*** Dense matrix.

***ellpack*** ELLPACK matrix.

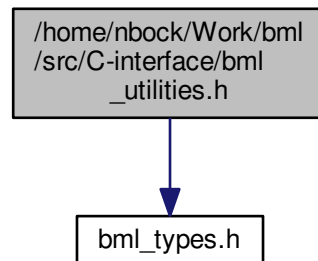
***csr*** CSR matrix.

## 14.16 /home/nbock/Work/bml/src/C-interface/bml\_types\_private.h File Reference

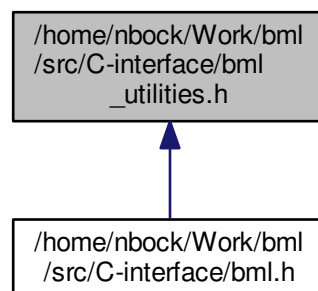
## 14.17 /home/nbock/Work/bml/src/C-interface/bml\_utilities.h File Reference

```
#include "bml_types.h"
```

Include dependency graph for bml\_utilities.h:



This graph shows which files directly or indirectly include this file:



### Functions

- void [bml\\_print\\_dense\\_matrix](#) (const int N, const [bml\\_matrix\\_precision\\_t](#) matrix\_precision, const [bml\\_dense\\_order\\_t](#) order, const void \*A, const int i\_l, const int i\_u, const int j\_l, const int j\_u)
- void [bml\\_print\\_dense\\_vector](#) (const int N, [bml\\_matrix\\_precision\\_t](#) matrix\_precision, const void \*v, const int i\_l, const int i\_u)
- void [bml\\_print\\_bml\\_vector](#) (const [bml\\_vector\\_t](#) \*v, const int i\_l, const int i\_u)
- void [bml\\_print\\_bml\\_matrix](#) (const [bml\\_matrix\\_t](#) \*A, const int i\_l, const int i\_u, const int j\_l, const int j\_u)

### 14.17.1 Function Documentation

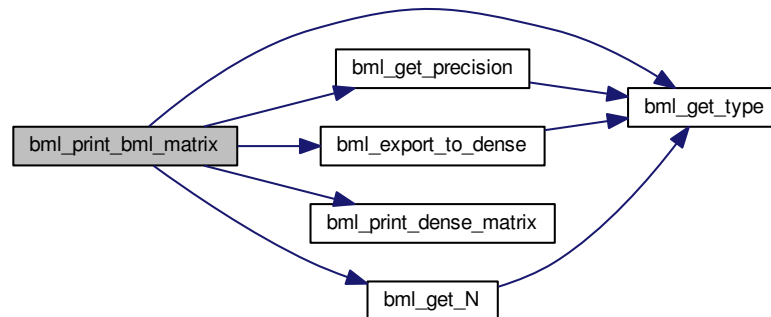
14.17.1.1 void [bml\\_print\\_bml\\_matrix](#) ( const [bml\\_matrix\\_t](#) \* A, const int *i\_l*, const int *i\_u*, const int *j\_l*, const int *j\_u* )

Print a dense matrix.

## Parameters

<i>A</i>	The matrix.
<i>i_l</i>	The lower row index.
<i>i_u</i>	The upper row index.
<i>j_l</i>	The lower column index.
<i>j_u</i>	The upper column index.

Here is the call graph for this function:



#### 14.17.1.2 void bml\_print\_bml\_vector ( const bml\_vector\_t \* *v*, const int *i\_l*, const int *i\_u* )

Print a bml vector.

## Parameters

<i>v</i>	The vector.
<i>i_l</i>	The lower row index.
<i>i_u</i>	The upper row index.

#### 14.17.1.3 void bml\_print\_dense\_matrix ( const int *N*, const bml\_matrix\_precision\_t *matrix\_precision*, const bml\_dense\_order\_t *order*, const void \* *A*, const int *i\_l*, const int *i\_u*, const int *j\_l*, const int *j\_u* )

Print a dense matrix.

## Parameters

<i>N</i>	The number of rows/columns.
<i>matrix_precision</i>	The real precision.
<i>order</i>	The matrix element order.
<i>A</i>	The matrix.
<i>i_l</i>	The lower row index.
<i>i_u</i>	The upper row index.
<i>j_l</i>	The lower column index.
<i>j_u</i>	The upper column index.

Here is the caller graph for this function:



14.17.1.4 void `bml_print_dense_vector` ( const int *N*, `bml_matrix_precision_t` *matrix\_precision*, const void \* *v*, const int *i\_l*, const int *i\_u* )

Print a dense vector.

Parameters

<i>N</i>	The number of rows/columns.
<i>matrix_precision</i>	The real precision.
<i>v</i>	The vector.
<i>i_l</i>	The lower row index.
<i>i_u</i>	The upper row index.

## 14.18 /home/nbock/Work/bml/src/C-interface/macros.h File Reference

### Macros

- #define `ROWMAJOR`(*i*, *j*, *M*, *N*) (*i*) \* (*M*) + (*j*)
- #define `COLMAJOR`(*i*, *j*, *M*, *N*) (*i*) + (*N*) \* (*j*)

### 14.18.1 Macro Definition Documentation

14.18.1.1 #define `COLMAJOR`( *i*, *j*, *M*, *N* ) (*i*) + (*N*) \* (*j*)

Column major access.

14.18.1.2 #define `ROWMAJOR`( *i*, *j*, *M*, *N* ) (*i*) \* (*M*) + (*j*)

Row major access.

# Index

- /home/nbock/Work/bml/src/C-interface/bml.h, [47](#)
- /home/nbock/Work/bml/src/C-interface/bml\_add.h, [48](#)
- /home/nbock/Work/bml/src/C-interface/bml\_allocate.h, [48](#)
- /home/nbock/Work/bml/src/C-interface/bml\_convert.h, [50](#)
- /home/nbock/Work/bml/src/C-interface/bml\_copy.h, [50](#)
- /home/nbock/Work/bml/src/C-interface/bml\_export.h, [52](#)
- /home/nbock/Work/bml/src/C-interface/bml\_import.h, [53](#)
- /home/nbock/Work/bml/src/C-interface/bml\_introspection.h, [55](#)
- /home/nbock/Work/bml/src/C-interface/bml\_logger.h, [62](#)
- /home/nbock/Work/bml/src/C-interface/bml\_multiply.h, [64](#)
- /home/nbock/Work/bml/src/C-interface/bml\_scale.h, [66](#)
- /home/nbock/Work/bml/src/C-interface/bml\_threshold.h, [69](#)
- /home/nbock/Work/bml/src/C-interface/bml\_trace.h, [71](#)
- /home/nbock/Work/bml/src/C-interface/bml\_transpose.h, [72](#)
- /home/nbock/Work/bml/src/C-interface/bml\_types.h, [74](#)
- /home/nbock/Work/bml/src/C-interface/bml\_types\_private.h, [75](#)
- /home/nbock/Work/bml/src/C-interface/bml\_utilities.h, [76](#)
- /home/nbock/Work/bml/src/C-interface/macros.h, [78](#)
- Add Functions (C interface), [24](#)
  - bml\_add, [24](#)
  - bml\_add\_identity, [24](#)
- Add Functions (Fortran interface), [31](#)
- Allocation and Deallocation Functions (C interface), [21](#)
  - bml\_allocate\_memory, [21](#)
  - bml\_deallocate, [21](#)
  - bml\_free\_memory, [22](#)
  - bml\_identity\_matrix, [22](#)
  - bml\_random\_matrix, [22](#)
  - bml\_zero\_matrix, [23](#)
- Allocation and Deallocation Functions (Fortran interface), [28](#)
  - bml\_deallocate, [28](#)
  - bml\_identity\_matrix, [28](#)
  - bml\_random\_matrix, [28](#)
  - bml\_zero\_matrix, [28](#)
- BML\_LOG\_DEBUG
  - bml\_logger.h, [63](#)
- BML\_LOG\_ERROR
  - bml\_logger.h, [63](#)
- BML\_LOG\_INFO
  - bml\_logger.h, [63](#)
- BML\_LOG\_WARNING
  - bml\_logger.h, [63](#)
- bml, [35](#)
- bml\_add
  - Add Functions (C interface), [24](#)
- bml\_add\_identity
  - Add Functions (C interface), [24](#)
- bml\_allocate\_m, [35](#)
- bml\_allocate\_memory
  - Allocation and Deallocation Functions (C interface), [21](#)
- bml\_convert\_from\_dense
  - bml\_import.h, [54](#)
- bml\_convert\_from\_dense\_double
  - Converting between Matrix Formats (Fortran interface), [32](#)
- bml\_convert\_from\_dense\_double\_complex
  - Converting between Matrix Formats (Fortran interface), [32](#)
- bml\_convert\_from\_dense\_single\_complex
  - Converting between Matrix Formats (Fortran interface), [33](#)
- bml\_convert\_to\_dense
  - bml\_export.h, [53](#)
- bml\_convert\_to\_dense\_double
  - Converting between Matrix Formats (Fortran interface), [33](#)
- bml\_convert\_to\_dense\_double\_complex
  - Converting between Matrix Formats (Fortran interface), [33](#)
- bml\_convert\_to\_dense\_single
  - Converting between Matrix Formats (Fortran interface), [33](#)
- bml\_convert\_to\_dense\_single\_complex
  - Converting between Matrix Formats (Fortran interface), [33](#)
- bml\_copy
  - bml\_copy.h, [51](#)
  - bml\_copy\_m, [36](#)
- bml\_copy.h
  - bml\_copy, [51](#)
  - bml\_copy\_new, [51](#)
- bml\_copy\_m, [35](#)
- bml\_copy\_new
  - bml\_copy, [36](#)
- bml\_copy\_new
  - bml\_copy.h, [51](#)
- bml\_deallocate
  - Allocation and Deallocation Functions (C interface),

- 21
- Allocation and Deallocation Functions (Fortran interface), 28
- bml\_debug
  - bml\_error\_m, 37
- bml\_dense\_order\_t
  - bml\_types.h, 75
- bml\_diagonalize
  - bml\_diagonalize\_m, 36
- bml\_diagonalize\_m, 36
  - bml\_diagonalize, 36
- bml\_error
  - bml\_error\_m, 37
- bml\_error\_m, 36
  - bml\_debug, 37
  - bml\_error, 37
  - bml\_warning, 37
- bml\_export.h
  - bml\_convert\_to\_dense, 53
- bml\_export\_to\_dense
  - Converting between Matrix Formats (C interface), 26
- bml\_free\_memory
  - Allocation and Deallocation Functions (C interface), 22
- bml\_get\_M
  - bml\_introspection.h, 56
- bml\_get\_N
  - bml\_introspection.h, 56
- bml\_get\_bandwidth
  - bml\_introspection.h, 55
  - bml\_introspection\_m, 40
- bml\_get\_n
  - bml\_introspection\_m, 40
- bml\_get\_precision
  - bml\_introspection.h, 58
- bml\_get\_row\_bandwidth
  - bml\_introspection.h, 59
  - bml\_introspection\_m, 40
- bml\_get\_type
  - bml\_introspection.h, 59
- bml\_identity\_matrix
  - Allocation and Deallocation Functions (C interface), 22
  - Allocation and Deallocation Functions (Fortran interface), 28
- bml\_import.h
  - bml\_convert\_from\_dense, 54
- bml\_import\_from\_dense
  - Converting between Matrix Formats (C interface), 27
- bml\_interface\_m, 38
  - bml\_matrix\_precision\_double\_complex\_enum\_id, 39
  - bml\_matrix\_precision\_double\_real\_enum\_id, 39
  - bml\_matrix\_precision\_single\_complex\_enum\_id, 39
  - bml\_matrix\_precision\_single\_real\_enum\_id, 39
- bml\_matrix\_precision\_uninitialized\_id, 39
- bml\_matrix\_type\_dense\_enum\_id, 39
- bml\_matrix\_type\_ellpack\_enum\_id, 39
- bml\_matrix\_type\_uninitialized\_enum\_id, 39
- get\_enum\_id, 38
- bml\_introspection.h
  - bml\_get\_M, 56
  - bml\_get\_N, 56
  - bml\_get\_bandwidth, 55
  - bml\_get\_precision, 58
  - bml\_get\_row\_bandwidth, 59
  - bml\_get\_type, 59
- bml\_introspection\_m, 40
  - bml\_get\_bandwidth, 40
  - bml\_get\_n, 40
  - bml\_get\_row\_bandwidth, 40
- bml\_log
  - bml\_logger.h, 63
- bml\_log\_level\_t
  - bml\_logger.h, 63
- bml\_log\_location
  - bml\_logger.h, 64
- bml\_logger.h
  - BML\_LOG\_DEBUG, 63
  - BML\_LOG\_ERROR, 63
  - BML\_LOG\_INFO, 63
  - BML\_LOG\_WARNING, 63
  - bml\_log, 63
  - bml\_log\_level\_t, 63
  - bml\_log\_location, 64
  - LOG\_DEBUG, 63
  - LOG\_ERROR, 63
  - LOG\_INFO, 63
  - LOG\_WARN, 63
- bml\_matrix\_precision\_double\_complex\_enum\_id
  - bml\_interface\_m, 39
- bml\_matrix\_precision\_double\_real\_enum\_id
  - bml\_interface\_m, 39
- bml\_matrix\_precision\_single\_complex\_enum\_id
  - bml\_interface\_m, 39
- bml\_matrix\_precision\_single\_real\_enum\_id
  - bml\_interface\_m, 39
- bml\_matrix\_precision\_t
  - bml\_types.h, 75
- bml\_matrix\_precision\_uninitialized\_id
  - bml\_interface\_m, 39
- bml\_matrix\_t
  - bml\_types.h, 74
- bml\_matrix\_type\_dense\_enum\_id
  - bml\_interface\_m, 39
- bml\_matrix\_type\_ellpack\_enum\_id
  - bml\_interface\_m, 39
- bml\_matrix\_type\_t
  - bml\_types.h, 75
- bml\_matrix\_type\_uninitialized\_enum\_id
  - bml\_interface\_m, 39
- bml\_multiply
  - bml\_multiply.h, 65



- bml\_multiply\_m, 41
- bml\_multiply.h
  - bml\_multiply, 65
  - bml\_multiply\_AB, 65
  - bml\_multiply\_x2, 66
- bml\_multiply\_AB
  - bml\_multiply.h, 65
- bml\_multiply\_m, 41
  - bml\_multiply, 41
- bml\_multiply\_x2
  - bml\_multiply.h, 66
- bml\_print\_bml\_matrix
  - bml\_utilities.h, 76
- bml\_print\_bml\_vector
  - bml\_utilities.h, 77
  - bml\_utilities\_m, 44
- bml\_print\_dense\_matrix
  - bml\_utilities.h, 77
- bml\_print\_dense\_vector
  - bml\_utilities.h, 78
- bml\_random\_matrix
  - Allocation and Deallocation Functions (C interface), 22
  - Allocation and Deallocation Functions (Fortran interface), 28
- bml\_scale
  - bml\_scale.h, 67
- bml\_scale.h
  - bml\_scale, 67
  - bml\_scale\_inplace, 67
  - bml\_scale\_new, 68
- bml\_scale\_inplace
  - bml\_scale.h, 67
- bml\_scale\_m, 41
  - scale\_two, 42
- bml\_scale\_new
  - bml\_scale.h, 68
- bml\_threshold
  - bml\_threshold.h, 69
- bml\_threshold.h
  - bml\_threshold, 69
  - bml\_threshold\_new, 70
- bml\_threshold\_new
  - bml\_threshold.h, 70
- bml\_trace
  - bml\_trace.h, 71
  - bml\_trace\_m, 42
- bml\_trace.h
  - bml\_trace, 71
- bml\_trace\_m, 42
  - bml\_trace, 42
- bml\_transpose
  - bml\_transpose.h, 73
  - bml\_transpose\_m, 43
- bml\_transpose.h
  - bml\_transpose, 73
  - bml\_transpose\_new, 73
- bml\_transpose\_m, 42
  - bml\_transpose, 43
- bml\_transpose\_new
  - bml\_transpose.h, 73
- bml\_types.h
  - bml\_dense\_order\_t, 75
  - bml\_matrix\_precision\_t, 75
  - bml\_matrix\_t, 74
  - bml\_matrix\_type\_t, 75
  - bml\_vector\_t, 74
  - csr, 75
  - dense, 75
  - dense\_column\_major, 75
  - dense\_row\_major, 75
  - double\_complex, 75
  - double\_real, 75
  - ellpack, 75
  - precision\_uninitialized, 75
  - single\_complex, 75
  - single\_real, 75
  - type\_uninitialized, 75
- bml\_types\_m, 43
- bml\_types\_m::bml\_matrix\_t, 45
- bml\_types\_m::bml\_vector\_t, 45
- bml\_utilities.h
  - bml\_print\_bml\_matrix, 76
  - bml\_print\_bml\_vector, 77
  - bml\_print\_dense\_matrix, 77
  - bml\_print\_dense\_vector, 78
- bml\_utilities\_m, 43
  - bml\_print\_bml\_vector, 44
- bml\_utilities\_matrix\_type\_m, 44
- bml\_vector\_t
  - bml\_types.h, 74
- bml\_warning
  - bml\_error\_m, 37
- bml\_zero\_matrix
  - Allocation and Deallocation Functions (C interface), 23
  - Allocation and Deallocation Functions (Fortran interface), 28
- COLMAJOR
  - macros.h, 78
- Converting between Matrix Formats (C interface), 26
  - bml\_export\_to\_dense, 26
  - bml\_import\_from\_dense, 27
- Converting between Matrix Formats (Fortran interface), 32
  - bml\_convert\_from\_dense\_double, 32
  - bml\_convert\_from\_dense\_double\_complex, 32
  - bml\_convert\_from\_dense\_single\_complex, 33
  - bml\_convert\_to\_dense\_double, 33
  - bml\_convert\_to\_dense\_double\_complex, 33
  - bml\_convert\_to\_dense\_single, 33
  - bml\_convert\_to\_dense\_single\_complex, 33
- csr
  - bml\_types.h, 75
- dense

- bml\_types.h, [75](#)
- dense\_column\_major
  - bml\_types.h, [75](#)
- dense\_row\_major
  - bml\_types.h, [75](#)
- double\_complex
  - bml\_types.h, [75](#)
- double\_real
  - bml\_types.h, [75](#)
- ellpack
  - bml\_types.h, [75](#)
- get\_enum\_id
  - bml\_interface\_m, [38](#)
- LOG\_DEBUG
  - bml\_logger.h, [63](#)
- LOG\_ERROR
  - bml\_logger.h, [63](#)
- LOG\_INFO
  - bml\_logger.h, [63](#)
- LOG\_WARN
  - bml\_logger.h, [63](#)
- macros.h
  - COLMAJOR, [78](#)
  - ROWMAJOR, [78](#)
- precision\_uninitialized
  - bml\_types.h, [75](#)
- ROWMAJOR
  - macros.h, [78](#)
- scale\_two
  - bml\_scale\_m, [42](#)
- single\_complex
  - bml\_types.h, [75](#)
- single\_real
  - bml\_types.h, [75](#)
- type\_uninitialized
  - bml\_types.h, [75](#)