

bml  
0.1.0

Generated by Doxygen 1.8.9.1

Fri Sep 18 2015 14:15:03



# 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>Module Index</b>	<b>11</b>
6.1	Modules . . . . .	11
<b>7</b>	<b>Namespace Index</b>	<b>13</b>
7.1	Namespace List . . . . .	13
<b>8</b>	<b>Class Index</b>	<b>15</b>
8.1	Class List . . . . .	15
<b>9</b>	<b>File Index</b>	<b>17</b>
9.1	File List . . . . .	17
<b>10</b>	<b>Module Documentation</b>	<b>19</b>
10.1	Allocation and Deallocation Functions (C interface) . . . . .	19
10.1.1	Detailed Description . . . . .	19
10.1.2	Function Documentation . . . . .	19
10.1.2.1	bml_allocate_memory . . . . .	19

10.1.2.2	<a href="#">bml_deallocate</a>	19
10.1.2.3	<a href="#">bml_free_memory</a>	20
10.1.2.4	<a href="#">bml_identity_matrix</a>	20
10.1.2.5	<a href="#">bml_random_matrix</a>	20
10.1.2.6	<a href="#">bml_zero_matrix</a>	21
10.2	<a href="#">Converting between Matrix Formats (C interface)</a>	22
10.2.1	<a href="#">Detailed Description</a>	22
10.2.2	<a href="#">Function Documentation</a>	22
10.2.2.1	<a href="#">bml_convert_from_dense</a>	22
10.2.2.2	<a href="#">bml_convert_to_dense</a>	22
10.3	<a href="#">Allocation and Deallocation Functions (Fortran interface)</a>	24
10.3.1	<a href="#">Detailed Description</a>	24
10.3.2	<a href="#">Function Documentation</a>	24
10.3.2.1	<a href="#">bml_deallocate</a>	24
10.3.2.2	<a href="#">bml_identity_matrix</a>	24
10.3.2.3	<a href="#">bml_random_matrix</a>	24
10.3.2.4	<a href="#">bml_zero_matrix</a>	25
10.4	<a href="#">Converting between Matrix Formats (Fortran interface)</a>	27
10.4.1	<a href="#">Detailed Description</a>	27
10.4.2	<a href="#">Function Documentation</a>	27
10.4.2.1	<a href="#">bml_convert_from_dense_double</a>	27
10.4.2.2	<a href="#">bml_convert_to_dense_double</a>	27
10.4.2.3	<a href="#">bml_convert_to_dense_single</a>	27
<b>11</b>	<b><a href="#">Namespace Documentation</a></b>	<b>29</b>
11.1	<a href="#">bml Module Reference</a>	29
11.1.1	<a href="#">Detailed Description</a>	29
11.2	<a href="#">bml_allocate Module Reference</a>	29
11.2.1	<a href="#">Detailed Description</a>	29
11.3	<a href="#">bml_interface Module Reference</a>	29
11.3.1	<a href="#">Detailed Description</a>	30
11.3.2	<a href="#">Function/Subroutine Documentation</a>	30
11.3.2.1	<a href="#">get_enum_id</a>	30
11.4	<a href="#">bml_introspection Module Reference</a>	30
11.4.1	<a href="#">Detailed Description</a>	30
11.4.2	<a href="#">Function/Subroutine Documentation</a>	31
11.4.2.1	<a href="#">bml_get_size</a>	31
11.5	<a href="#">bml_types Module Reference</a>	32
11.5.1	<a href="#">Detailed Description</a>	32
11.6	<a href="#">bml_utilities Module Reference</a>	32

11.6.1 Detailed Description . . . . .	32
11.6.2 Function/Subroutine Documentation . . . . .	33
11.6.2.1 bml_print_matrix_double . . . . .	33
<b>12 Class Documentation</b>	<b>35</b>
12.1 bml_introspection::bml_get_size_C Interface Reference . . . . .	35
12.1.1 Detailed Description . . . . .	35
12.2 bml_types::bml_matrix_t Type Reference . . . . .	35
12.2.1 Detailed Description . . . . .	35
<b>13 File Documentation</b>	<b>37</b>
13.1 /home/nbock/Work/bml/src-new/C-interface/bml.h File Reference . . . . .	37
13.1.1 Detailed Description . . . . .	37
13.2 /home/nbock/Work/bml/src-new/C-interface/bml_allocate.h File Reference . . . . .	38
13.3 /home/nbock/Work/bml/src-new/C-interface/bml_convert.h File Reference . . . . .	39
13.4 /home/nbock/Work/bml/src-new/C-interface/bml_copy.h File Reference . . . . .	40
13.4.1 Function Documentation . . . . .	41
13.4.1.1 bml_copy . . . . .	41
13.5 /home/nbock/Work/bml/src-new/C-interface/bml_introspection.h File Reference . . . . .	41
13.5.1 Function Documentation . . . . .	42
13.5.1.1 bml_get_size . . . . .	42
13.5.1.2 bml_get_type . . . . .	43
13.6 /home/nbock/Work/bml/src-new/C-interface/bml_logger.h File Reference . . . . .	44
13.6.1 Macro Definition Documentation . . . . .	45
13.6.1.1 LOG_DEBUG . . . . .	45
13.6.1.2 LOG_ERROR . . . . .	45
13.6.1.3 LOG_INFO . . . . .	45
13.6.1.4 LOG_WARN . . . . .	45
13.6.2 Enumeration Type Documentation . . . . .	45
13.6.2.1 bml_log_level_t . . . . .	45
13.6.3 Function Documentation . . . . .	45
13.6.3.1 bml_log . . . . .	45
13.6.3.2 bml_log_location . . . . .	46
13.7 /home/nbock/Work/bml/src-new/C-interface/bml_types.h File Reference . . . . .	46
13.7.1 Typedef Documentation . . . . .	46
13.7.1.1 bml_matrix_t . . . . .	46
13.7.2 Enumeration Type Documentation . . . . .	46
13.7.2.1 bml_matrix_precision_t . . . . .	47
13.7.2.2 bml_matrix_type_t . . . . .	47
13.8 /home/nbock/Work/bml/src-new/C-interface/bml_types_private.h File Reference . . . . .	47
13.9 /home/nbock/Work/bml/src-new/C-interface/bml_utilities.h File Reference . . . . .	47

13.9.1	Function Documentation . . . . .	48
13.9.1.1	bml_print_matrix . . . . .	48
	<b>Index</b>	<b>49</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

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

# Module Index

### 6.1 Modules

Here is a list of all modules:

Allocation and Deallocation Functions (C interface) . . . . .	<a href="#">19</a>
Converting between Matrix Formats (C interface) . . . . .	<a href="#">22</a>
Allocation and Deallocation Functions (Fortran interface) . . . . .	<a href="#">24</a>
Converting between Matrix Formats (Fortran interface) . . . . .	<a href="#">27</a>



## Chapter 7

# Namespace Index

### 7.1 Namespace List

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

<a href="#">bml</a>	Main matrix library module . . . . .	29
<a href="#">bml_allocate</a>	Matrix allocation functions . . . . .	29
<a href="#">bml_interface</a>	Interface module . . . . .	29
<a href="#">bml_introspection</a>	Introspection procedures . . . . .	30
<a href="#">bml_types</a>	The basic bml types . . . . .	32
<a href="#">bml_utilities</a>	Utility matrix functions . . . . .	32



## Chapter 8

# Class Index

### 8.1 Class List

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

<a href="#">bml_introspection::bml_get_size_C</a>	
Return the matrix size . . . . .	<a href="#">35</a>
<a href="#">bml_types::bml_matrix_t</a>	
The bml matrix type . . . . .	<a href="#">35</a>



## Chapter 9

# File Index

### 9.1 File List

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

/home/nbock/Work/bml/src-new/C-interface/ <a href="#">bml.h</a> . . . . .	37
/home/nbock/Work/bml/src-new/C-interface/ <a href="#">bml_allocate.h</a> . . . . .	38
/home/nbock/Work/bml/src-new/C-interface/ <a href="#">bml_convert.h</a> . . . . .	39
/home/nbock/Work/bml/src-new/C-interface/ <a href="#">bml_copy.h</a> . . . . .	40
/home/nbock/Work/bml/src-new/C-interface/ <a href="#">bml_introspection.h</a> . . . . .	41
/home/nbock/Work/bml/src-new/C-interface/ <a href="#">bml_logger.h</a> . . . . .	44
/home/nbock/Work/bml/src-new/C-interface/ <a href="#">bml_types.h</a> . . . . .	46
/home/nbock/Work/bml/src-new/C-interface/ <a href="#">bml_types_private.h</a> . . . . .	47
/home/nbock/Work/bml/src-new/C-interface/ <a href="#">bml_utilities.h</a> . . . . .	47





# Chapter 10

## Module Documentation

### 10.1 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)

#### 10.1.1 Detailed Description

#### 10.1.2 Function Documentation

##### 10.1.2.1 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.

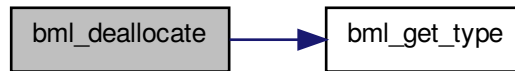
##### 10.1.2.2 void [bml\\_deallocate](#) ( [bml\\_matrix\\_t](#) \*\* A )

Deallocate a matrix.

#### Parameters

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

Here is the call graph for this function:



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

Deallocate a chunk of memory.

##### Parameters

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

#### 10.1.2.4 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. The default is double precision.
<i>N</i>	The matrix size.
<i>M</i>	The number of non-zeroes per row.

##### Returns

The matrix.

#### 10.1.2.5 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. The default is double precision.
<i>N</i>	The matrix size.

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

**Returns**

The matrix.

10.1.2.6 **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. The default is double precision.
$N$	The matrix size.
$M$	The number of non-zeroes per row.

**Returns**

The matrix.

## 10.2 Converting between Matrix Formats (C interface)

### Functions

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

### 10.2.1 Detailed Description

### 10.2.2 Function Documentation

10.2.2.1 `bml_matrix_t* bml_convert_from_dense ( const bml_matrix_type_t matrix_type, const bml_matrix_precision_t matrix_precision, const int N, const void * A, const double threshold, const int M )`

Convert a dense matrix into a bml matrix.

#### Parameters

<i>matrix_type</i>	The matrix type
<i>matrix_precision</i>	The real precision
<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

10.2.2.2 `void* bml_convert_to_dense ( const bml_matrix_t * A )`

Convert a bml matrix into a dense 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
----------	----------------

**Returns**

The dense matrix

Here is the call graph for this function:



## 10.3 Allocation and Deallocation Functions (Fortran interface)

### Functions

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

### 10.3.1 Detailed Description

### 10.3.2 Function Documentation

#### 10.3.2.1 subroutine, public `bml_allocate::bml_deallocate` ( *type(bml\_matrix\_t)* *a* )

Deallocate a matrix.

Parameters

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

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

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.

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

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.

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

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.



## 10.4 Converting between Matrix Formats (Fortran interface)

### Functions

- subroutine `bml_convert::bml_convert_from_dense_double` (`matrix_type`, `matrix_precision`, `a_dense`, `a`, `threshold`, `m`)  
*Convert a dense matrix into a bml matrix.*
- subroutine `bml_convert::bml_convert_to_dense_single` (`a`, `a_dense`)  
*Convert a matrix into a dense matrix.*
- subroutine `bml_convert::bml_convert_to_dense_double` (`a`, `a_dense`)  
*Convert a matrix into a dense matrix.*

### 10.4.1 Detailed Description

### 10.4.2 Function Documentation

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

Convert a dense matrix into a bml matrix.

#### Parameters

<code>matrix_type</code>	The matrix type
<code>matrix_precision</code>	The matrix precision
<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

- 10.4.2.2 subroutine `bml_convert::bml_convert_to_dense_double` ( `type(bml_matrix_t)`, `intent(in)` `a`, `double precision`, `dimension(:, :)`, `intent(out)`, `pointer` `a_dense` )

Convert a matrix into a dense matrix.

#### Parameters

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

- 10.4.2.3 subroutine `bml_convert::bml_convert_to_dense_single` ( `type(bml_matrix_t)`, `intent(in)` `a`, `real`, `dimension(:, :)`, `intent(out)`, `pointer` `a_dense` )

Convert a matrix into a dense matrix.

#### Parameters

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



# Chapter 11

## Namespace Documentation

### 11.1 bml Module Reference

Main matrix library module.

#### 11.1.1 Detailed Description

Main matrix library module.

Use this modules in order to use the library.

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

#### 11.2.1 Detailed Description

Matrix allocation functions.

### 11.3 bml\_interface Module Reference

Interface module.

## Functions/Subroutines

- integer function [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
- integer, parameter [bml\\_matrix\\_precision\\_single\\_enum\\_id](#) = 0
- integer, parameter [bml\\_matrix\\_precision\\_double\\_enum\\_id](#) = 1

### 11.3.1 Detailed Description

Interface module.

### 11.3.2 Function/Subroutine Documentation

#### 11.3.2.1 integer function [bml\\_interface::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](#).

## 11.4 [bml\\_introspection](#) Module Reference

Introspection procedures.

### Data Types

- interface [bml\\_get\\_size\\_C](#)  
*Return the matrix size.*

### Functions/Subroutines

- integer function [bml\\_get\\_size](#) (a)  
*Return the matrix size.*

#### 11.4.1 Detailed Description

Introspection procedures.

## 11.4.2 Function/Subroutine Documentation

11.4.2.1 integer function `bml_introspection::bml_get_size ( type(bml_matrix_t), intent(in) a )`

Return the matrix size.

## Parameters

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

## Returns

The matrix size.

## 11.5 bml\_types Module Reference

The basic bml types.

### Data Types

- type [bml\\_matrix\\_t](#)  
*The bml matrix 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](#) = "single-precision"  
*The single precision identifier.*
- character(len=\*), parameter [bml\\_precision\\_double](#) = "double-precision"  
*The double-precision identifier.*

#### 11.5.1 Detailed Description

The basic bml types.

## 11.6 bml\_utilities Module Reference

Utility matrix functions.

### Functions/Subroutines

- subroutine [bml\\_print\\_matrix\\_double](#) (tag, a, i\_l, i\_u, j\_l, j\_u)  
*Print a dense matrix.*

#### 11.6.1 Detailed Description

Utility matrix functions.

## 11.6.2 Function/Subroutine Documentation

11.6.2.1 subroutine `bml_utilities::bml_print_matrix_double` ( character(len=\*) *tag*, double precision, dimension(:, :),  
intent(in), target *a*, integer, intent(in) *i\_l*, integer, intent(in) *i\_u*, integer, intent(in) *j\_l*, integer, intent(in) *j\_u* )

Print a dense matrix.

## Parameters

<i>tag</i>	A string to print before the matrix.
<i>a</i>	The matrix.
<i>i_l</i>	The lower row bound.
<i>i_u</i>	The upper row bound.
<i>j_l</i>	The lower column bound.
<i>j_u</i>	The upper column bound.



# Chapter 12

## Class Documentation

### 12.1 `bml_introspection::bml_get_size_C` Interface Reference

Return the matrix size.

#### Public Member Functions

- `integer(c_int)` function **`bml_get_size_c`** (a)

#### 12.1.1 Detailed Description

Return the matrix size.

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

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

### 12.2 `bml_types::bml_matrix_t` Type Reference

The bml matrix type.

#### Public Attributes

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

#### 12.2.1 Detailed Description

The bml matrix type.

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

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



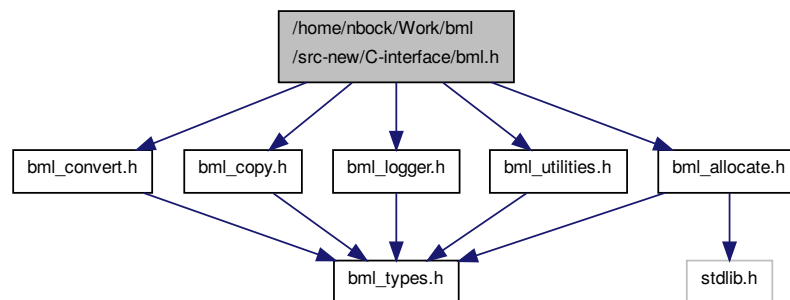
## Chapter 13

# File Documentation

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

```
#include "bml_allocate.h"  
#include "bml_convert.h"  
#include "bml_copy.h"  
#include "bml_logger.h"  
#include "bml_utilities.h"
```

Include dependency graph for bml.h:



#### 13.1.1 Detailed Description

## Copyright

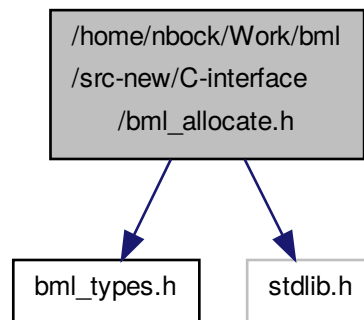
Los Alamos National Laboratory 2015

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

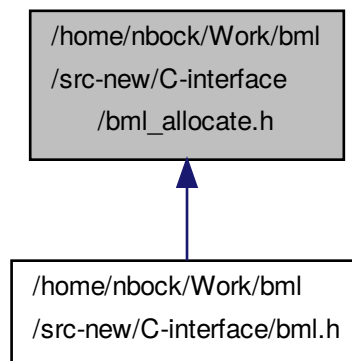
```
#include "bml_types.h"
```

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

Include dependency graph for bml\_allocate.h:



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



### Functions

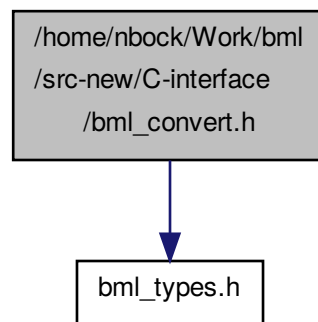
- void \* [bml\\_allocate\\_memory](#) (const size\_t s)
- void [bml\\_free\\_memory](#) (void \*ptr)
- void [bml\\_deallocate](#) ([bml\\_matrix\\_t](#) \*\*A)

- `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)

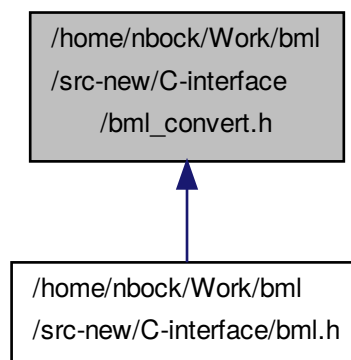
### 13.3 /home/nbock/Work/bml/src-new/C-interface/bml\_convert.h File Reference

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

Include dependency graph for bml\_convert.h:



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



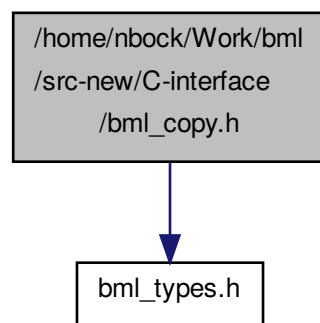
## Functions

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

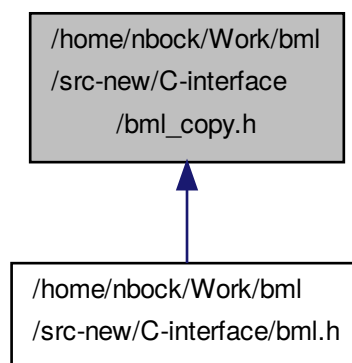
## 13.4 /home/nbock/Work/bml/src-new/C-interface/bml\_copy.h File Reference

```
#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` (const `bml_matrix_t` \*A)

### 13.4.1 Function Documentation

#### 13.4.1.1 `bml_matrix_t* bml_copy ( const bml_matrix_t * A )`

Copy a matrix.

##### Parameters

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

##### Returns

A Copy of A

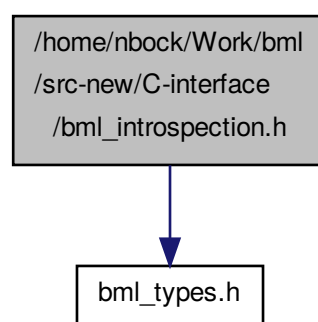
Here is the call graph for this function:



## 13.5 /home/nbock/Work/bml/src-new/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)`
- `int bml_get_size (const bml_matrix_t *A)`

### 13.5.1 Function Documentation

#### 13.5.1.1 `int bml_get_size ( 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:



### 13.5.1.2 `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.

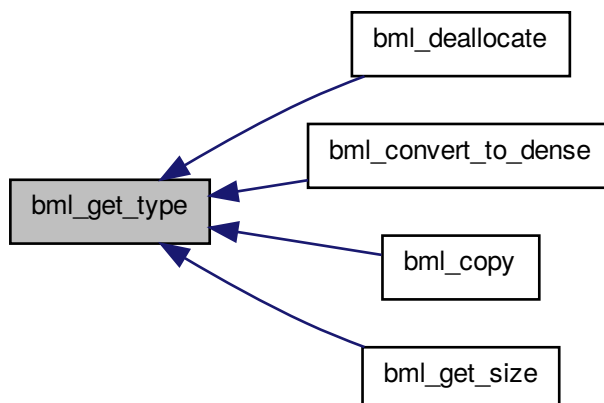
## Parameters

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

## Returns

The matrix type.

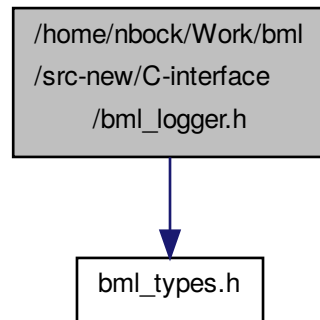
Here is the caller graph for this function:



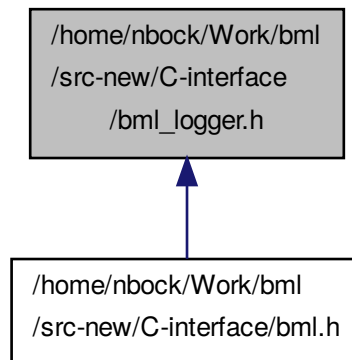
## 13.6 /home/nbock/Work/bml/src-new/C-interface/bml\_logger.h File Reference

```
#include "bml_types.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`,...)

### 13.6.1 Macro Definition Documentation

13.6.1.1 `#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.

13.6.1.2 `#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.

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

Convenience macro to write a `BML_LOG_INFO` level message.

13.6.1.4 `#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.

### 13.6.2 Enumeration Type Documentation

13.6.2.1 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.

### 13.6.3 Function Documentation

13.6.3.1 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()).

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

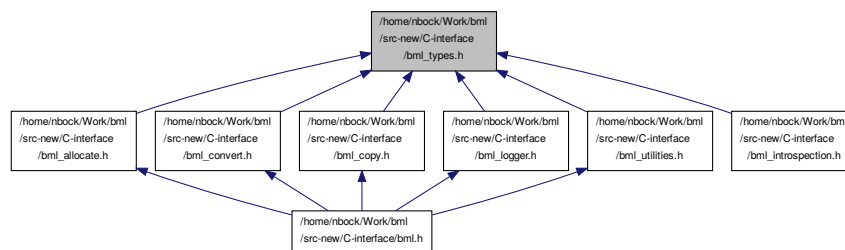
Log a message with location, i.e. filename and linenumber..

## 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()).

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

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



### Typedefs

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

### Enumerations

- enum [bml\\_matrix\\_type\\_t](#) { uninitialized, dense, ellpack, csr }
- enum [bml\\_matrix\\_precision\\_t](#) { single\_real, double\_real }

#### 13.7.1 Typedef Documentation

##### 13.7.1.1 typedef void bml\_matrix\_t

The matrix type.

#### 13.7.2 Enumeration Type Documentation

## 13.7.2.1 enum bml\_matrix\_precision\_t

The supported real precisions.

## Enumerator

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

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

## 13.7.2.2 enum bml\_matrix\_type\_t

The supported matrix types.

## Enumerator

**uninitialized** The matrix is not initialized.

**dense** Dense matrix.

**ellpack** ELLPACK matrix.

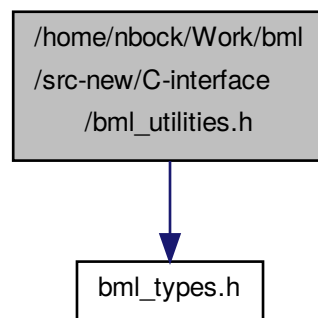
**csr** CSR matrix.

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

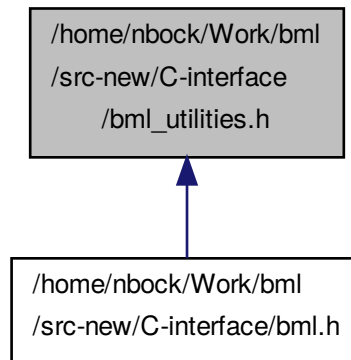
## 13.9 /home/nbock/Work/bml/src-new/C-interface/bml\_utilities.h File Reference

```
#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\\_matrix](#) (const int *N*, [bml\\_matrix\\_precision\\_t](#) *matrix\_precision*, const void \**A*, const int *i\_l*, const int *i\_u*, const int *j\_l*, const int *j\_u*)

### 13.9.1 Function Documentation

13.9.1.1 void [bml\\_print\\_matrix](#) ( const int *N*, [bml\\_matrix\\_precision\\_t](#) *matrix\_precision*, 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>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.

# Index

/home/nbock/Work/bml/src-new/C-interface/bml.h, [37](#)  
/home/nbock/Work/bml/src-new/C-interface/bml\_↔  
allocate.h, [38](#)  
/home/nbock/Work/bml/src-new/C-interface/bml\_↔  
convert.h, [39](#)  
/home/nbock/Work/bml/src-new/C-interface/bml\_↔  
copy.h, [40](#)  
/home/nbock/Work/bml/src-new/C-interface/bml\_↔  
introspection.h, [41](#)  
/home/nbock/Work/bml/src-new/C-interface/bml\_↔  
logger.h, [44](#)  
/home/nbock/Work/bml/src-new/C-interface/bml\_↔  
types.h, [46](#)  
/home/nbock/Work/bml/src-new/C-interface/bml\_↔  
types\_private.h, [47](#)  
/home/nbock/Work/bml/src-new/C-interface/bml\_↔  
utilities.h, [47](#)

Allocation and Deallocation Functions (C interface), [19](#)

bml\_allocate\_memory, [19](#)  
bml\_deallocate, [19](#)  
bml\_free\_memory, [20](#)  
bml\_identity\_matrix, [20](#)  
bml\_random\_matrix, [20](#)  
bml\_zero\_matrix, [21](#)

Allocation and Deallocation Functions (Fortran interface), [24](#)

bml\_deallocate, [24](#)  
bml\_identity\_matrix, [24](#)  
bml\_random\_matrix, [24](#)  
bml\_zero\_matrix, [24](#)

BML\_LOG\_DEBUG

bml\_logger.h, [45](#)

BML\_LOG\_ERROR

bml\_logger.h, [45](#)

BML\_LOG\_INFO

bml\_logger.h, [45](#)

BML\_LOG\_WARNING

bml\_logger.h, [45](#)

bml, [29](#)

bml\_allocate, [29](#)

bml\_allocate\_memory

Allocation and Deallocation Functions (C interface),  
[19](#)

bml\_convert\_from\_dense

Converting between Matrix Formats (C interface),  
[22](#)

bml\_convert\_from\_dense\_double

Converting between Matrix Formats (Fortran interface), [27](#)

bml\_convert\_to\_dense

Converting between Matrix Formats (C interface),  
[22](#)

bml\_convert\_to\_dense\_double

Converting between Matrix Formats (Fortran interface), [27](#)

bml\_convert\_to\_dense\_single

Converting between Matrix Formats (Fortran interface), [27](#)

bml\_copy

bml\_copy.h, [41](#)

bml\_copy.h

bml\_copy, [41](#)

bml\_deallocate

Allocation and Deallocation Functions (C interface),  
[19](#)

Allocation and Deallocation Functions (Fortran interface), [24](#)

bml\_free\_memory

Allocation and Deallocation Functions (C interface),  
[20](#)

bml\_get\_size

bml\_introspection, [31](#)

bml\_introspection.h, [42](#)

bml\_get\_type

bml\_introspection.h, [43](#)

bml\_identity\_matrix

Allocation and Deallocation Functions (C interface),  
[20](#)

Allocation and Deallocation Functions (Fortran interface), [24](#)

bml\_interface, [29](#)

get\_enum\_id, [30](#)

bml\_introspection, [30](#)

bml\_get\_size, [31](#)

bml\_introspection.h

bml\_get\_size, [42](#)

bml\_get\_type, [43](#)

bml\_introspection::bml\_get\_size\_C, [35](#)

bml\_log

bml\_logger.h, [45](#)

bml\_log\_level\_t

bml\_logger.h, [45](#)

bml\_log\_location

bml\_logger.h, [46](#)

bml\_logger.h

BML\_LOG\_DEBUG, [45](#)

- BML\_LOG\_ERROR, [45](#)
- BML\_LOG\_INFO, [45](#)
- BML\_LOG\_WARNING, [45](#)
- bml\_log, [45](#)
- bml\_log\_level\_t, [45](#)
- bml\_log\_location, [46](#)
- LOG\_DEBUG, [45](#)
- LOG\_ERROR, [45](#)
- LOG\_INFO, [45](#)
- LOG\_WARN, [45](#)
- bml\_matrix\_precision\_t
  - bml\_types.h, [46](#)
- bml\_matrix\_t
  - bml\_types.h, [46](#)
- bml\_matrix\_type\_t
  - bml\_types.h, [47](#)
- bml\_print\_matrix
  - bml\_utilities.h, [48](#)
- bml\_print\_matrix\_double
  - bml\_utilities, [33](#)
- bml\_random\_matrix
  - Allocation and Deallocation Functions (C interface), [20](#)
  - Allocation and Deallocation Functions (Fortran interface), [24](#)
- bml\_types, [32](#)
- bml\_types.h
  - bml\_matrix\_precision\_t, [46](#)
  - bml\_matrix\_t, [46](#)
  - bml\_matrix\_type\_t, [47](#)
  - csr, [47](#)
  - dense, [47](#)
  - double\_real, [47](#)
  - ellpack, [47](#)
  - single\_real, [47](#)
  - uninitialized, [47](#)
- bml\_types::bml\_matrix\_t, [35](#)
- bml\_utilities, [32](#)
  - bml\_print\_matrix\_double, [33](#)
- bml\_utilities.h
  - bml\_print\_matrix, [48](#)
- bml\_zero\_matrix
  - Allocation and Deallocation Functions (C interface), [21](#)
  - Allocation and Deallocation Functions (Fortran interface), [24](#)
- Converting between Matrix Formats (C interface), [22](#)
  - bml\_convert\_from\_dense, [22](#)
  - bml\_convert\_to\_dense, [22](#)
- Converting between Matrix Formats (Fortran interface), [27](#)
  - bml\_convert\_from\_dense\_double, [27](#)
  - bml\_convert\_to\_dense\_double, [27](#)
  - bml\_convert\_to\_dense\_single, [27](#)
- csr
  - bml\_types.h, [47](#)
- dense
  - bml\_types.h, [47](#)
- double\_real
  - bml\_types.h, [47](#)
- ellpack
  - bml\_types.h, [47](#)
- get\_enum\_id
  - bml\_interface, [30](#)
- LOG\_DEBUG
  - bml\_logger.h, [45](#)
- LOG\_ERROR
  - bml\_logger.h, [45](#)
- LOG\_INFO
  - bml\_logger.h, [45](#)
- LOG\_WARN
  - bml\_logger.h, [45](#)
- single\_real
  - bml\_types.h, [47](#)
- uninitialized
  - bml\_types.h, [47](#)