

bml
0.1.0

Generated by Doxygen 1.8.9.1

Mon Nov 16 2015 13:03:38

Contents

1	Basic Matrix Library (bml)	1
1.1	Usage Examples	1
1.2	Modifying the library itself	1
1.3	Planned Features	1
2	Future Plans	3
2.1	Matrix Types	3
2.2	Precisions	3
2.3	Functions	3
3	C Usage	5
4	Fortran Usage	7
5	Developer Documentation	9
5.1	Developer Suggested Workflow	9
5.2	Coding Style	9
6	Deprecated List	11
7	Module Index	13
7.1	Modules	13
8	Namespace Index	15
8.1	Namespace List	15
9	Class Index	17
9.1	Class List	17
10	File Index	19
10.1	File List	19
11	Module Documentation	21
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	bml_allocate_memory	21
11.1.2.2	bml_deallocate	21
11.1.2.3	bml_free_memory	22
11.1.2.4	bml_identity_matrix	22
11.1.2.5	bml_random_matrix	22
11.1.2.6	bml_zero_matrix	23
11.2	Add Functions (C interface)	24
11.2.1	Detailed Description	24
11.2.2	Function Documentation	24
11.2.2.1	bml_add	24
11.2.2.2	bml_add_identity	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	bml_export_to_dense	26
11.3.2.2	bml_import_from_dense	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	bml_deallocate	28
11.4.2.2	bml_identity_matrix	28
11.4.2.3	bml_random_matrix	28
11.4.2.4	bml_zero_matrix	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	bml_convert_from_dense_double	32
11.6.2.2	bml_convert_from_dense_double_complex	32
11.6.2.3	bml_convert_from_dense_single_complex	33
11.6.2.4	bml_convert_to_dense_double	33
11.6.2.5	bml_convert_to_dense_double_complex	33
11.6.2.6	bml_convert_to_dense_single	33
11.6.2.7	bml_convert_to_dense_single_complex	33
12	Namespace Documentation	35
12.1	bml Module Reference	35
12.1.1	Detailed Description	35

12.2	bml_allocate_m Module Reference	35
12.2.1	Detailed Description	35
12.3	bml_copy_m Module Reference	35
12.3.1	Detailed Description	36
12.3.2	Function/Subroutine Documentation	36
12.3.2.1	bml_copy	36
12.4	bml_diagonalize_m Module Reference	36
12.4.1	Detailed Description	36
12.4.2	Function/Subroutine Documentation	36
12.4.2.1	bml_diagonalize	36
12.5	bml_error_m Module Reference	36
12.5.1	Detailed Description	37
12.5.2	Function/Subroutine Documentation	37
12.5.2.1	bml_debug	37
12.5.2.2	bml_error	37
12.5.2.3	bml_warning	37
12.6	bml_interface_m Module Reference	38
12.6.1	Detailed Description	38
12.6.2	Function/Subroutine Documentation	38
12.6.2.1	get_enum_id	38
12.6.3	Variable Documentation	39
12.6.3.1	bml_matrix_precision_double_complex_enum_id	39
12.6.3.2	bml_matrix_precision_double_real_enum_id	39
12.6.3.3	bml_matrix_precision_single_complex_enum_id	39
12.6.3.4	bml_matrix_precision_single_real_enum_id	39
12.6.3.5	bml_matrix_precision_uninitialized_id	39
12.6.3.6	bml_matrix_type_dense_enum_id	39
12.6.3.7	bml_matrix_type_ellpack_enum_id	39
12.6.3.8	bml_matrix_type_uninitialized_enum_id	40
12.7	bml_introspection_m Module Reference	40
12.7.1	Detailed Description	40
12.7.2	Function/Subroutine Documentation	40
12.7.2.1	bml_get_bandwidth	40
12.7.2.2	bml_get_n	40
12.7.2.3	bml_get_row_bandwidth	40
12.8	bml_multiply_m Module Reference	41
12.8.1	Detailed Description	41
12.8.2	Function/Subroutine Documentation	41
12.8.2.1	bml_multiply	41
12.9	bml_scale_m Module Reference	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
13 Class Documentation	45
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
14 File Documentation	47
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	bml_convert_from_dense	54
14.8	/home/nbock/Work/bml/src/C-interface/bml_introspection.h File Reference	55
14.8.1	Function Documentation	55
14.8.1.1	bml_get_bandwidth	55
14.8.1.2	bml_get_M	56
14.8.1.3	bml_get_N	57
14.8.1.4	bml_get_precision	58
14.8.1.5	bml_get_row_bandwidth	59
14.8.1.6	bml_get_type	59
14.9	/home/nbock/Work/bml/src/C-interface/bml_logger.h File Reference	60
14.9.1	Macro Definition Documentation	62
14.9.1.1	LOG_DEBUG	62
14.9.1.2	LOG_ERROR	62
14.9.1.3	LOG_INFO	62
14.9.1.4	LOG_WARN	62
14.9.2	Enumeration Type Documentation	62
14.9.2.1	bml_log_level_t	62
14.9.3	Function Documentation	62
14.9.3.1	bml_log	62
14.9.3.2	bml_log_location	63
14.10	/home/nbock/Work/bml/src/C-interface/bml_multiply.h File Reference	63
14.10.1	Function Documentation	64
14.10.1.1	bml_multiply	64
14.10.1.2	bml_multiply_AB	64
14.10.1.3	bml_multiply_x2	65
14.11	/home/nbock/Work/bml/src/C-interface/bml_scale.h File Reference	65
14.11.1	Function Documentation	66
14.11.1.1	bml_scale	66
14.11.1.2	bml_scale_inplace	66
14.11.1.3	bml_scale_new	67
14.12	/home/nbock/Work/bml/src/C-interface/bml_threshold.h File Reference	68
14.12.1	Function Documentation	68
14.12.1.1	bml_threshold	68
14.12.1.2	bml_threshold_new	69
14.13	/home/nbock/Work/bml/src/C-interface/bml_trace.h File Reference	70
14.13.1	Function Documentation	70
14.13.1.1	bml_trace	70
14.14	/home/nbock/Work/bml/src/C-interface/bml_transpose.h File Reference	71
14.14.1	Function Documentation	72
14.14.1.1	bml_transpose	72

14.14.1.2 <code>bml_transpose_new</code>	72
14.15/home/nbock/Work/bml/src/C-interface/bml_types.h File Reference	73
14.15.1 Typedef Documentation	73
14.15.1.1 <code>bml_matrix_t</code>	73
14.15.1.2 <code>bml_vector_t</code>	73
14.15.2 Enumeration Type Documentation	74
14.15.2.1 <code>bml_dense_order_t</code>	74
14.15.2.2 <code>bml_matrix_precision_t</code>	74
14.15.2.3 <code>bml_matrix_type_t</code>	74
14.16/home/nbock/Work/bml/src/C-interface/bml_types_private.h File Reference	74
14.17/home/nbock/Work/bml/src/C-interface/bml_utilities.h File Reference	75
14.17.1 Function Documentation	75
14.17.1.1 <code>bml_print_bml_matrix</code>	75
14.17.1.2 <code>bml_print_bml_vector</code>	76
14.17.1.3 <code>bml_print_dense_matrix</code>	76
14.17.1.4 <code>bml_print_dense_vector</code>	76
14.18/home/nbock/Work/bml/src/C-interface/macros.h File Reference	77
14.18.1 Macro Definition Documentation	77
14.18.1.1 <code>COLMAJOR</code>	77
14.18.1.2 <code>ROWMAJOR</code>	77
Index	79

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
Jamaludin Mohd-Yusof jamal@lanl.gov
Nicolas Bock nbock@lanl.gov
Susan M. Mniszewski 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, 100×100 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)	21
Add Functions (C interface)	24
Converting between Matrix Formats (C interface)	26
Allocation and Deallocation Functions (Fortran interface)	28
Add Functions (Fortran interface)	31
Converting between Matrix Formats (Fortran interface)	32

Chapter 8

Namespace Index

8.1 Namespace List

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

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

Chapter 9

Class Index

9.1 Class List

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

bml_types_m::bml_matrix_t	
The bml matrix type	45
bml_types_m::bml_vector_t	
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/ blas.h	??
/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_diagonalize.h	??
/home/nbock/Work/bml/src/C-interface/ bml_elemental.h	??
/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	60
/home/nbock/Work/bml/src/C-interface/ bml_multiply.h	63
/home/nbock/Work/bml/src/C-interface/ bml_scale.h	65
/home/nbock/Work/bml/src/C-interface/ bml_threshold.h	68
/home/nbock/Work/bml/src/C-interface/ bml_trace.h	70
/home/nbock/Work/bml/src/C-interface/ bml_transpose.h	71
/home/nbock/Work/bml/src/C-interface/ bml_types.h	73
/home/nbock/Work/bml/src/C-interface/ bml_types_private.h	74
/home/nbock/Work/bml/src/C-interface/ bml_utilities.h	75
/home/nbock/Work/bml/src/C-interface/ lapack.h	??
/home/nbock/Work/bml/src/C-interface/ macros.h	77
/home/nbock/Work/bml/src/C-interface/ typed.h	??

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);
```

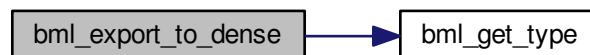
Parameters

<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 α and β 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 α .
<i>beta</i>	The factor β .

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

<i>a</i>	The matrix.
<i>a_t</i>	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`

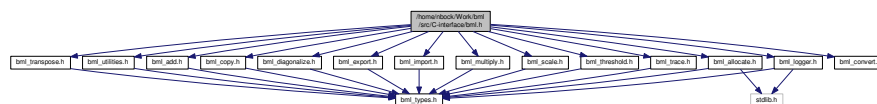
Chapter 14

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

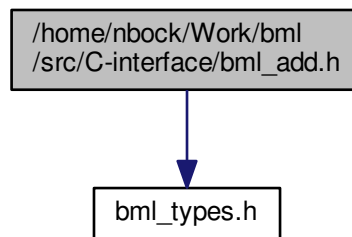
Copyright

Los Alamos National Laboratory 2015

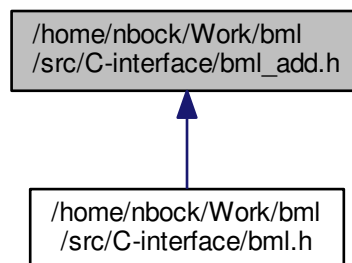
14.2 /home/nbock/Work/bml/src/C-interface/bml_add.h File Reference

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

Include dependency graph for bml_add.h:



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



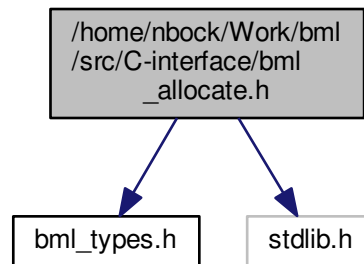
Functions

- 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)

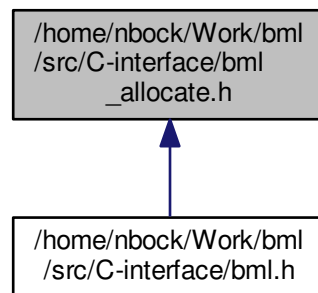
14.3 /home/nbock/Work/bml/src/C-interface/bml_allocate.h File Reference

```
#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:

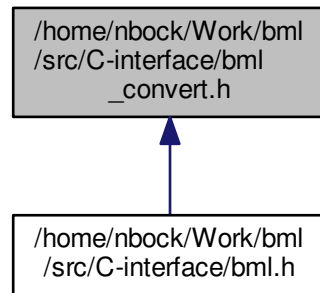


Functions

- 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

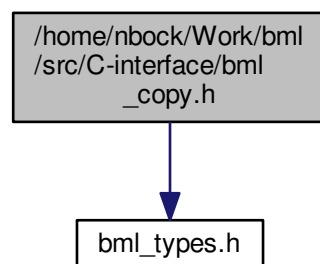
This graph shows which files directly or indirectly include this file:



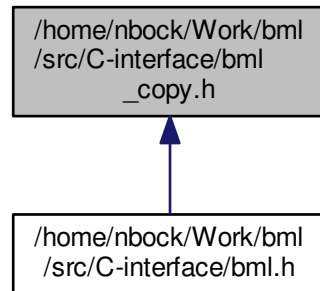
14.5 /home/nbock/Work/bml/src/C-interface/bml_copy.h File Reference

```
#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

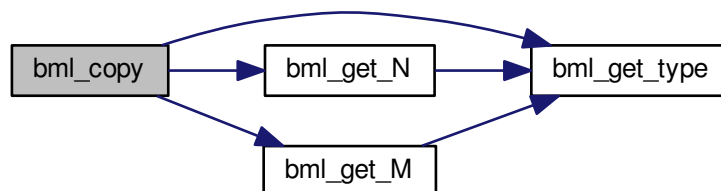
14.5.1.1 `void bml_copy (const bml_matrix_t * A, bml_matrix_t * B)`

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

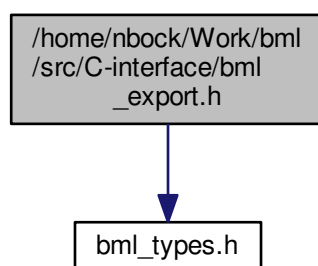
Here is the call graph for this function:



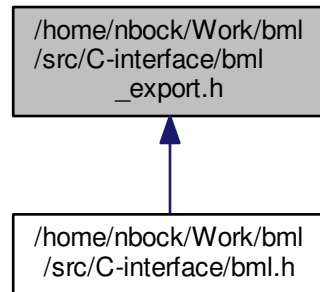
14.6 /home/nbock/Work/bml/src/C-interface/bml_export.h File Reference

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

Include dependency graph for `bml_export.h`:



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



Functions

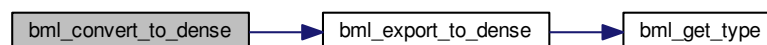
- void * [bml_convert_to_dense](#) (const [bml_matrix_t](#) *A, const [bml_dense_order_t](#) order)
- void * [bml_export_to_dense](#) (const [bml_matrix_t](#) *A, const [bml_dense_order_t](#) order)

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.

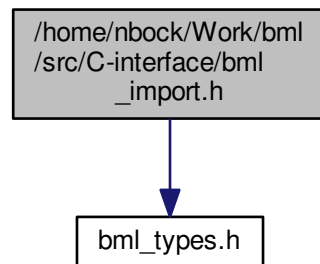
Here is the call graph for this function:



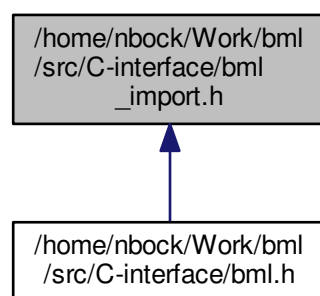
14.7 /home/nbock/Work/bml/src/C-interface/bml_import.h File Reference

```
#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.

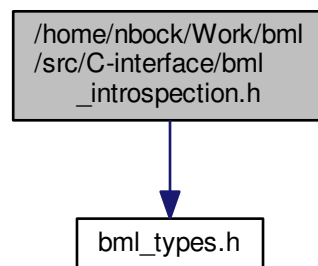
Here is the call graph for this function:



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.
----------	-------------

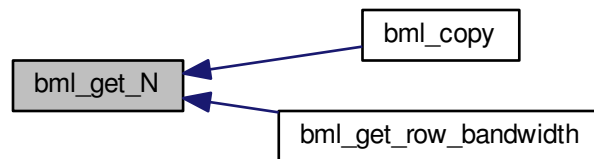
Returns

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:



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.

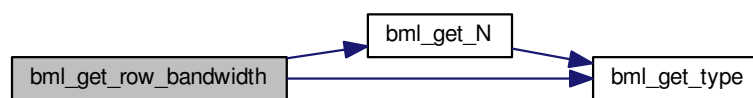
Parameters

<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.

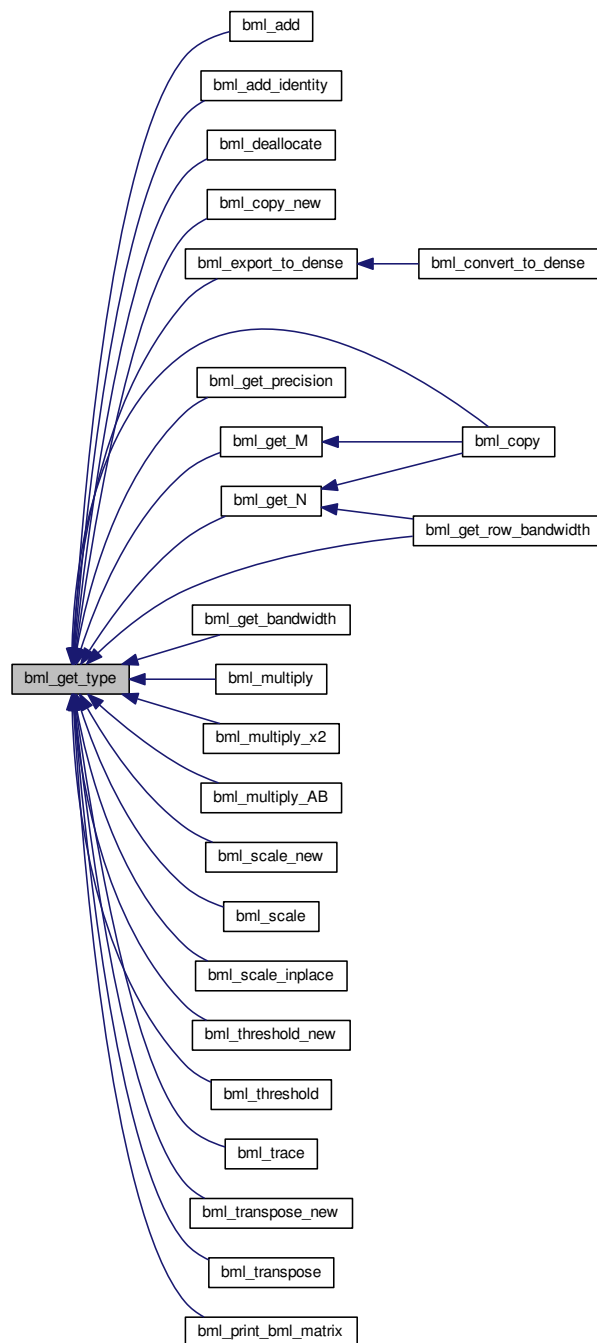
Parameters

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

Returns

The matrix type.

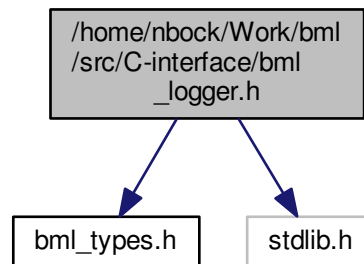
Here is the caller graph for this function:



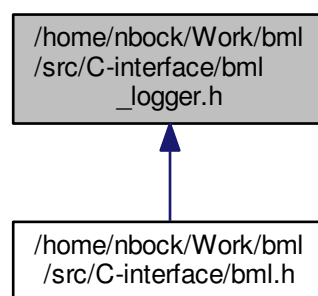
14.9 /home/nbock/Work/bml/src/C-interface/bml_logger.h File Reference

```
#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..

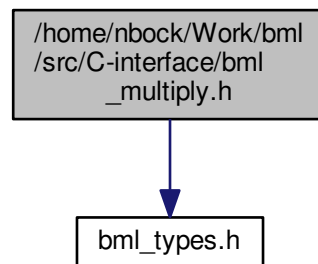
Parameters

<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()).

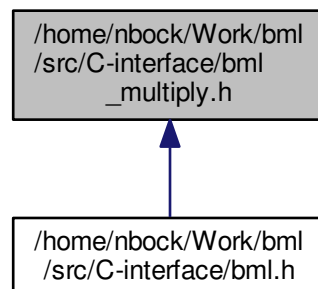
14.10 /home/nbock/Work/bml/src/C-interface/bml_multiply.h File Reference

```
#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*)

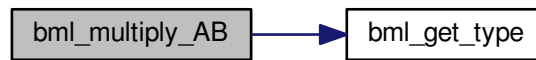
Matrix multiply.

$C = A * B$

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

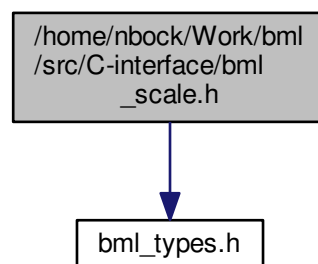
Here is the call graph for this function:



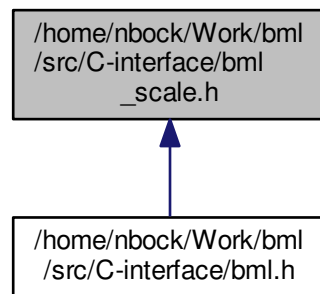
14.11 /home/nbock/Work/bml/src/C-interface/bml_scale.h File Reference

```
#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

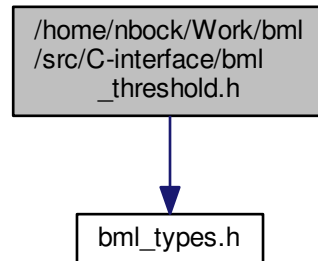
Here is the call graph for this function:



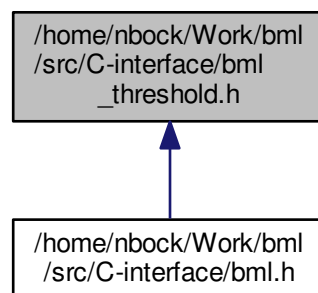
14.12 /home/nbock/Work/bml/src/C-interface/bml_threshold.h File Reference

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

14.12.1 Function Documentation

14.12.1.1 void `bml_threshold` (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:



14.12.1.2 `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

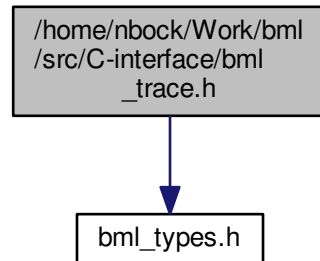
Here is the call graph for this function:



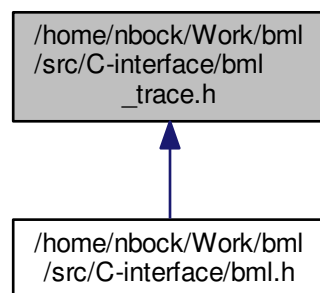
14.13 /home/nbock/Work/bml/src/C-interface/bml_trace.h File Reference

```
#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*

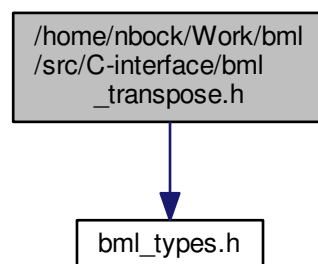
Here is the call graph for this function:



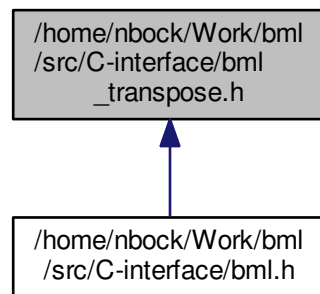
14.14 /home/nbock/Work/bml/src/C-interface/bml_transpose.h File Reference

```
#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.

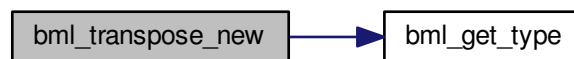
Parameters

A	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.

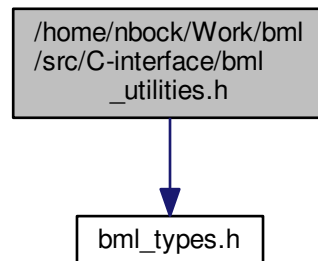
csr CSR matrix.

14.16 `/home/nbock/Work/bml/src/C-interface/bml_types_private.h` File Reference

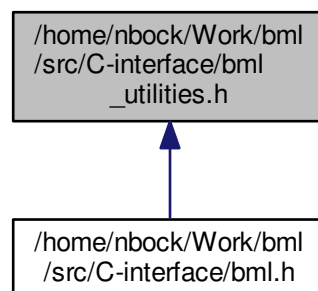
14.17 /home/nbock/Work/bml/src/C-interface/bml_utilities.h File Reference

```
#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

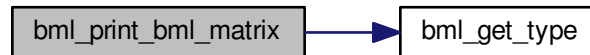
14.17.1.1 void [bml_print_bml_matrix](#) (const [bml_matrix_t](#) * A, const int *i_l*, const int *i_u*, const int *j_l*, const int *j_u*)

Print a dense matrix.

Parameters

A	The matrix.
i_l	The lower row index.
i_u	The upper row index.
j_l	The lower column index.
j_u	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

v	The vector.
i_l	The lower row index.
i_u	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

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

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

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

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

Macros

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

14.18.1 Macro Definition Documentation

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

Column major access.

14.18.1.2 #define ROWMAJOR(i, j, N) (i) * (N) + (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, [60](#)
- /home/nbock/Work/bml/src/C-interface/bml_multiply.h, [63](#)
- /home/nbock/Work/bml/src/C-interface/bml_scale.h, [65](#)
- /home/nbock/Work/bml/src/C-interface/bml_threshold.h, [68](#)
- /home/nbock/Work/bml/src/C-interface/bml_trace.h, [70](#)
- /home/nbock/Work/bml/src/C-interface/bml_transpose.h, [71](#)
- /home/nbock/Work/bml/src/C-interface/bml_types.h, [73](#)
- /home/nbock/Work/bml/src/C-interface/bml_types_private.h, [74](#)
- /home/nbock/Work/bml/src/C-interface/bml_utilities.h, [75](#)
- /home/nbock/Work/bml/src/C-interface/macros.h, [77](#)
- 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, [62](#)
- BML_LOG_ERROR
 - bml_logger.h, [62](#)
- BML_LOG_INFO
 - bml_logger.h, [62](#)
- BML_LOG_WARNING
 - bml_logger.h, [62](#)
- 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.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, 74
- 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), 26
- 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, 62
- bml_log_level_t
 - bml_logger.h, 62
- bml_log_location
 - bml_logger.h, 63
- bml_logger.h
 - BML_LOG_DEBUG, 62
 - BML_LOG_ERROR, 62
 - BML_LOG_INFO, 62
 - BML_LOG_WARNING, 62
 - bml_log, 62
 - bml_log_level_t, 62
 - bml_log_location, 63
 - LOG_DEBUG, 62
 - LOG_ERROR, 62
 - LOG_INFO, 62
 - LOG_WARN, 62
- 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, 74
- bml_matrix_precision_uninitialized_id
 - bml_interface_m, 39
- bml_matrix_t
 - bml_types.h, 73
- 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, 74
- bml_matrix_type_uninitialized_enum_id
 - bml_interface_m, 39
- bml_multiply
 - bml_multiply.h, 64

- bml_multiply_m, 41
- bml_multiply.h
 - bml_multiply, 64
 - bml_multiply_AB, 64
 - bml_multiply_x2, 65
- bml_multiply_AB
 - bml_multiply.h, 64
- bml_multiply_m, 41
 - bml_multiply, 41
- bml_multiply_x2
 - bml_multiply.h, 65
- bml_print_bml_matrix
 - bml_utilities.h, 75
- bml_print_bml_vector
 - bml_utilities.h, 76
 - bml_utilities_m, 44
- bml_print_dense_matrix
 - bml_utilities.h, 76
- bml_print_dense_vector
 - bml_utilities.h, 76
- bml_random_matrix
 - Allocation and Deallocation Functions (C interface), 22
 - Allocation and Deallocation Functions (Fortran interface), 28
- bml_scale
 - bml_scale.h, 66
- bml_scale.h
 - bml_scale, 66
 - bml_scale_inplace, 66
 - bml_scale_new, 67
- bml_scale_inplace
 - bml_scale.h, 66
- bml_scale_m, 41
 - scale_two, 42
- bml_scale_new
 - bml_scale.h, 67
- bml_threshold
 - bml_threshold.h, 68
- bml_threshold.h
 - bml_threshold, 68
 - bml_threshold_new, 69
- bml_threshold_new
 - bml_threshold.h, 69
- bml_trace
 - bml_trace.h, 70
 - bml_trace_m, 42
- bml_trace.h
 - bml_trace, 70
- bml_trace_m, 42
 - bml_trace, 42
- bml_transpose
 - bml_transpose.h, 72
 - bml_transpose_m, 43
- bml_transpose.h
 - bml_transpose, 72
 - bml_transpose_new, 72
- bml_transpose_m, 42
 - bml_transpose, 43
- bml_transpose_new
 - bml_transpose.h, 72
- bml_types.h
 - bml_dense_order_t, 74
 - bml_matrix_precision_t, 74
 - bml_matrix_t, 73
 - bml_matrix_type_t, 74
 - bml_vector_t, 73
 - csr, 74
 - dense, 74
 - dense_column_major, 74
 - dense_row_major, 74
 - double_complex, 74
 - double_real, 74
 - ellpack, 74
 - precision_uninitialized, 74
 - single_complex, 74
 - single_real, 74
 - type_uninitialized, 74
- 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, 75
 - bml_print_bml_vector, 76
 - bml_print_dense_matrix, 76
 - bml_print_dense_vector, 76
- bml_utilities_m, 43
 - bml_print_bml_vector, 44
- bml_utilities_matrix_type_m, 44
- bml_vector_t
 - bml_types.h, 73
- 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, 77
- Converting between Matrix Formats (C interface), 26
 - bml_export_to_dense, 26
 - bml_import_from_dense, 26
- 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, 74
- dense

- bml_types.h, [74](#)
- dense_column_major
 - bml_types.h, [74](#)
- dense_row_major
 - bml_types.h, [74](#)
- double_complex
 - bml_types.h, [74](#)
- double_real
 - bml_types.h, [74](#)
- ellpack
 - bml_types.h, [74](#)
- get_enum_id
 - bml_interface_m, [38](#)
- LOG_DEBUG
 - bml_logger.h, [62](#)
- LOG_ERROR
 - bml_logger.h, [62](#)
- LOG_INFO
 - bml_logger.h, [62](#)
- LOG_WARN
 - bml_logger.h, [62](#)
- macros.h
 - COLMAJOR, [77](#)
 - ROWMAJOR, [77](#)
- precision_uninitialized
 - bml_types.h, [74](#)
- ROWMAJOR
 - macros.h, [77](#)
- scale_two
 - bml_scale_m, [42](#)
- single_complex
 - bml_types.h, [74](#)
- single_real
 - bml_types.h, [74](#)
- type_uninitialized
 - bml_types.h, [74](#)