

bml

0.1.0

Generated by Doxygen 1.8.9.1

Mon Oct 5 2015 10:42:31

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	Module Index	11
6.1	Modules	11
7	Namespace Index	13
7.1	Namespace List	13
8	Class Index	15
8.1	Class List	15
9	File Index	17
9.1	File List	17
10	Module Documentation	19
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	bml_deallocate	19
10.1.2.3	bml_free_memory	20
10.1.2.4	bml_identity_matrix	20
10.1.2.5	bml_random_matrix	20
10.1.2.6	bml_zero_matrix	21
10.2	Add Functions (C interface)	22
10.2.1	Detailed Description	22
10.2.2	Function Documentation	22
10.2.2.1	bml_add	22
10.2.2.2	bml_add_identity	22
10.3	Converting between Matrix Formats (C interface)	24
10.3.1	Detailed Description	24
10.3.2	Function Documentation	24
10.3.2.1	bml_convert_from_dense	24
10.3.2.2	bml_convert_to_dense	24
10.4	Allocation and Deallocation Functions (Fortran interface)	26
10.4.1	Detailed Description	26
10.4.2	Function Documentation	26
10.4.2.1	bml_deallocate	26
10.4.2.2	bml_identity_matrix	26
10.4.2.3	bml_random_matrix	26
10.4.2.4	bml_zero_matrix	27
10.5	Add Functions (Fortran interface)	29
10.5.1	Detailed Description	29
10.6	Converting between Matrix Formats (Fortran interface)	30
10.6.1	Detailed Description	30
10.6.2	Function Documentation	30
10.6.2.1	bml_convert_from_dense_double	30
10.6.2.2	bml_convert_from_dense_double_complex	30
10.6.2.3	bml_convert_from_dense_single_complex	31
10.6.2.4	bml_convert_to_dense_double	31
10.6.2.5	bml_convert_to_dense_double_complex	31
10.6.2.6	bml_convert_to_dense_single	31
10.6.2.7	bml_convert_to_dense_single_complex	31
11	Namespace Documentation	33
11.1	bml Module Reference	33
11.1.1	Detailed Description	33
11.2	bml_allocate_m Module Reference	33
11.2.1	Detailed Description	33

11.3	bml_copy_m Module Reference	33
11.3.1	Detailed Description	34
11.3.2	Function/Subroutine Documentation	34
11.3.2.1	bml_copy	34
11.4	bml_diagonalize_m Module Reference	34
11.4.1	Detailed Description	34
11.4.2	Function/Subroutine Documentation	34
11.4.2.1	bml_diagonalize	34
11.5	bml_error_m Module Reference	34
11.5.1	Detailed Description	35
11.5.2	Function/Subroutine Documentation	35
11.5.2.1	bml_debug	35
11.5.2.2	bml_error	35
11.5.2.3	bml_warning	35
11.6	bml_interface_m Module Reference	36
11.6.1	Detailed Description	36
11.6.2	Function/Subroutine Documentation	36
11.6.2.1	get_enum_id	36
11.7	bml_introspection_m Module Reference	36
11.7.1	Detailed Description	37
11.7.2	Function/Subroutine Documentation	37
11.7.2.1	bml_get_bandwidth	37
11.7.2.2	bml_get_size	37
11.8	bml_multiply_m Module Reference	37
11.8.1	Detailed Description	37
11.8.2	Function/Subroutine Documentation	38
11.8.2.1	bml_multiply	38
11.9	bml_scale_m Module Reference	38
11.9.1	Detailed Description	38
11.9.2	Function/Subroutine Documentation	38
11.9.2.1	scale_two	38
11.10	bml_trace_m Module Reference	38
11.10.1	Detailed Description	39
11.10.2	Function/Subroutine Documentation	39
11.10.2.1	bml_trace	39
11.11	bml_transpose_m Module Reference	39
11.11.1	Detailed Description	39
11.11.2	Function/Subroutine Documentation	39
11.11.2.1	bml_transpose	39
11.12	bml_types_m Module Reference	39

11.12.1 Detailed Description	40
11.13 bml_utilities_m Module Reference	40
11.13.1 Detailed Description	40
11.13.2 Function/Subroutine Documentation	40
11.13.2.1 bml_print_bml_vector	40
11.14 bml_utilities_matrix_type_m Module Reference	41
11.14.1 Detailed Description	41
12 Class Documentation	43
12.1 bml_types_m::bml_matrix_t Type Reference	43
12.1.1 Detailed Description	43
12.2 bml_types_m::bml_vector_t Type Reference	43
12.2.1 Detailed Description	43
13 File Documentation	45
13.1 /home/nbock/Work/bml/src-new/C-interface/bml.h File Reference	45
13.1.1 Detailed Description	45
13.2 /home/nbock/Work/bml/src-new/C-interface/bml_add.h File Reference	46
13.3 /home/nbock/Work/bml/src-new/C-interface/bml_allocate.h File Reference	46
13.4 /home/nbock/Work/bml/src-new/C-interface/bml_convert.h File Reference	47
13.5 /home/nbock/Work/bml/src-new/C-interface/bml_copy.h File Reference	48
13.5.1 Function Documentation	49
13.5.1.1 bml_copy	49
13.5.1.2 bml_copy_new	49
13.6 /home/nbock/Work/bml/src-new/C-interface/bml_introspection.h File Reference	50
13.6.1 Function Documentation	50
13.6.1.1 bml_get_size	50
13.6.1.2 bml_get_type	51
13.7 /home/nbock/Work/bml/src-new/C-interface/bml_logger.h File Reference	52
13.7.1 Macro Definition Documentation	54
13.7.1.1 LOG_DEBUG	54
13.7.1.2 LOG_ERROR	54
13.7.1.3 LOG_INFO	54
13.7.1.4 LOG_WARN	54
13.7.2 Enumeration Type Documentation	54
13.7.2.1 bml_log_level_t	54
13.7.3 Function Documentation	54
13.7.3.1 bml_log	54
13.7.3.2 bml_log_location	55
13.8 /home/nbock/Work/bml/src-new/C-interface/bml_multiply.h File Reference	55
13.8.1 Function Documentation	55

13.8.1.1	bml_multiply	55
13.8.1.2	bml_multiply_x2	56
13.9	/home/nbock/Work/bml/src-new/C-interface/bml_scale.h File Reference	56
13.9.1	Function Documentation	57
13.9.1.1	bml_scale	57
13.9.1.2	bml_scale_new	58
13.10	/home/nbock/Work/bml/src-new/C-interface/bml_trace.h File Reference	59
13.10.1	Function Documentation	59
13.10.1.1	bml_trace	59
13.11	/home/nbock/Work/bml/src-new/C-interface/bml_types.h File Reference	60
13.11.1	Typedef Documentation	60
13.11.1.1	bml_matrix_t	60
13.11.1.2	bml_vector_t	60
13.11.2	Enumeration Type Documentation	60
13.11.2.1	bml_matrix_precision_t	60
13.11.2.2	bml_matrix_type_t	60
13.12	/home/nbock/Work/bml/src-new/C-interface/bml_types_private.h File Reference	61
13.13	/home/nbock/Work/bml/src-new/C-interface/bml_utilities.h File Reference	61
13.13.1	Function Documentation	62
13.13.1.1	bml_print_bml_matrix	62
13.13.1.2	bml_print_bml_vector	62
13.13.1.3	bml_print_dense_matrix	62
Index		63

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
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

Module Index

6.1 Modules

Here is a list of all modules:

Allocation and Deallocation Functions (C interface)	19
Add Functions (C interface)	22
Converting between Matrix Formats (C interface)	24
Allocation and Deallocation Functions (Fortran interface)	26
Add Functions (Fortran interface)	29
Converting between Matrix Formats (Fortran interface)	30

Chapter 7

Namespace Index

7.1 Namespace List

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

bml	Main matrix library module	33
bml_allocate_m	Matrix allocation functions	33
bml_copy_m	Copy operations on matrices	33
bml_diagonalize_m	Matrix diagonalization functions	34
bml_error_m	A module for error handling in bml	34
bml_interface_m	Interface module	36
bml_introspection_m	Introspection procedures	36
bml_multiply_m	Matrix multiplication	37
bml_scale_m	Matrix scaling for matrices	38
bml_trace_m	Matrix trace	38
bml_transpose_m	Transpose functions	39
bml_types_m	The basic bml types	39
bml_utilities_m	Utility matrix functions	40
bml_utilities_matrix_type_m	Utility matrix functions	41

Chapter 8

Class Index

8.1 Class List

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

bml_types_m::bml_matrix_t	
The bml matrix type	43
bml_types_m::bml_vector_t	
The bml vector type	43

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/ blas.h	??
/home/nbock/Work/bml/src-new/C-interface/ bml.h	45
/home/nbock/Work/bml/src-new/C-interface/ bml_add.h	46
/home/nbock/Work/bml/src-new/C-interface/ bml_allocate.h	46
/home/nbock/Work/bml/src-new/C-interface/ bml_convert.h	47
/home/nbock/Work/bml/src-new/C-interface/ bml_copy.h	48
/home/nbock/Work/bml/src-new/C-interface/ bml_elemental.h	??
/home/nbock/Work/bml/src-new/C-interface/ bml_introspection.h	50
/home/nbock/Work/bml/src-new/C-interface/ bml_logger.h	52
/home/nbock/Work/bml/src-new/C-interface/ bml_multiply.h	55
/home/nbock/Work/bml/src-new/C-interface/ bml_scale.h	56
/home/nbock/Work/bml/src-new/C-interface/ bml_trace.h	59
/home/nbock/Work/bml/src-new/C-interface/ bml_types.h	60
/home/nbock/Work/bml/src-new/C-interface/ bml_types_private.h	61
/home/nbock/Work/bml/src-new/C-interface/ bml_utilities.h	61
/home/nbock/Work/bml/src-new/C-interface/ lapack.h	??
/home/nbock/Work/bml/src-new/C-interface/ typed.h	??

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.
<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.
<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.
N	The matrix size.
M	The number of non-zeroes per row.

Returns

The matrix.

10.2 Add Functions (C interface)

Functions

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

10.2.1 Detailed Description

10.2.2 Function Documentation

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

Matrix addition.

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

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:



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

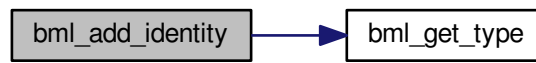
Matrix addition.

$A = A + \beta * I$

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:



10.3 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*)

10.3.1 Detailed Description

10.3.2 Function Documentation

10.3.2.1 `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

10.3.2.2 `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);
```

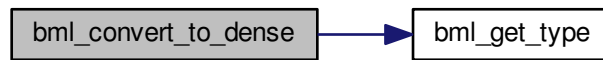
Parameters

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

Returns

The dense matrix

Here is the call graph for this function:



10.4 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.

10.4.1 Detailed Description

10.4.2 Function Documentation

10.4.2.1 subroutine, public `bml_allocate_m::bml_deallocate` (*type(bml_matrix_t)* *a*)

Deallocate a matrix.

Parameters

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

10.4.2.2 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.

10.4.2.3 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.

10.4.2.4 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.

10.5 Add Functions (Fortran interface)

10.5.1 Detailed Description

10.6 Converting between Matrix Formats (Fortran interface)

Functions

- subroutine `bml_convert_m::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_m::bml_convert_from_dense_single_complex` (`matrix_type`, `matrix_precision`, `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`, `matrix_precision`, `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.

10.6.1 Detailed Description

10.6.2 Function Documentation

- 10.6.2.1 subroutine `bml_convert_m::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

- 10.6.2.2 subroutine `bml_convert_m::bml_convert_from_dense_double_complex` (`character(len=*)`, `intent(in) matrix_type`, `character(len=*)`, `intent(in) matrix_precision`, `complex(kind(0.0d0))`, `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
--------------------------	-----------------

<i>matrix_precision</i>	The matrix precision
<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

10.6.2.3 subroutine `bml_convert_m::bml_convert_from_dense_single_complex` (`character(len=*)`, `intent(in)` *matrix_type*, `character(len=*)`, `intent(in)` *matrix_precision*, `complex`, `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

<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

10.6.2.4 subroutine `bml_convert_m::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

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

10.6.2.5 subroutine `bml_convert_m::bml_convert_to_dense_double_complex` (`type(bml_matrix_t)`, `intent(in)` *a*, `complex(kind(0d0))`, `dimension(:, :)`, `intent(out)`, `pointer` *a_dense*)

Convert a matrix into a dense matrix.

Parameters

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

10.6.2.6 subroutine `bml_convert_m::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

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

10.6.2.7 subroutine `bml_convert_m::bml_convert_to_dense_single_complex` (`type(bml_matrix_t)`, `intent(in)` *a*, `complex`, `dimension(:, :)`, `intent(out)`, `pointer` *a_dense*)

Convert a matrix into a dense matrix.

Parameters

<i>a</i>	The bml matrix
<i>a_dense</i>	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_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.

11.2.1 Detailed Description

Matrix allocation functions.

11.3 bml_copy_m Module Reference

Copy operations on matrices.

Functions/Subroutines

- subroutine [bml_copy](#) (a, b)
Copy (assign) a matrix to another one.

11.3.1 Detailed Description

Copy operations on matrices.

11.3.2 Function/Subroutine Documentation

11.3.2.1 subroutine [bml_copy_m::bml_copy](#) (type([bml_matrix_t](#)), intent(in) *a*, type([bml_matrix_t](#)), intent(inout) *b*)

Copy (assign) a matrix to another one.

This operation performs $B \leftarrow A$.

Parameters

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

11.4 [bml_diagonalize_m](#) Module Reference

Matrix diagonalization functions.

Functions/Subroutines

- subroutine [bml_diagonalize](#) (a, eigenvectors, eigenvalues)
Diagonalize a matrix.

11.4.1 Detailed Description

Matrix diagonalization functions.

11.4.2 Function/Subroutine Documentation

11.4.2.1 subroutine [bml_diagonalize_m::bml_diagonalize](#) (type([bml_matrix_t](#)), intent(in) *a*, type([bml_matrix_t](#)), intent(inout) *eigenvectors*, type([bml_vector_t](#)), intent(inout) *eigenvalues*)

Diagonalize a matrix.

Parameters

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

11.5 [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.

11.5.1 Detailed Description

A module for error handling in bml.

Copyright

Los Alamos National Laboratory 2015

11.5.2 Function/Subroutine Documentation

11.5.2.1 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.

11.5.2.2 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.

11.5.2.3 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.

11.6 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
- integer, parameter [bml_matrix_precision_single_enum_id](#) = 0
- integer, parameter [bml_matrix_precision_double_enum_id](#) = 1

11.6.1 Detailed Description

Interface module.

11.6.2 Function/Subroutine Documentation

11.6.2.1 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](#).

11.7 bml_introspection_m Module Reference

Introspection procedures.

Functions/Subroutines

- integer function, public [bml_get_size](#) (a)
Return the matrix size.

- integer function, public `bml_get_bandwidth` (a, i)
Get the number of non-zero elements in a given row.

11.7.1 Detailed Description

Introspection procedures.

11.7.2 Function/Subroutine Documentation

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

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

Parameters

a	The matrix.
i	The row.

Returns

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

11.7.2.2 integer function, public `bml_introspection_m::bml_get_size` (type(`bml_matrix_t`), intent(in) a)

Return the matrix size.

Parameters

a	The matrix.
---	-------------

Returns

The matrix size.

11.8 bml_multiply_m Module Reference

Matrix multiplication.

Functions/Subroutines

- subroutine `bml_multiply` (a, b, c, alpha, beta)
Multiply two matrices.

11.8.1 Detailed Description

Matrix multiplication.

11.8.2 Function/Subroutine Documentation

11.8.2.1 subroutine `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 .

11.9 bml_scale_m Module Reference

Matrix scaling for matrices.

Functions/Subroutines

- subroutine `scale_two` (*alpha*, *a*, *c*)
Scale a bml matrix.

11.9.1 Detailed Description

Matrix scaling for matrices.

11.9.2 Function/Subroutine Documentation

11.9.2.1 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

11.10 bml_trace_m Module Reference

Matrix trace.

Functions/Subroutines

- double precision function [bml_trace](#) (a)
Calculate the trace of a matrix.

11.10.1 Detailed Description

Matrix trace.

11.10.2 Function/Subroutine Documentation

11.10.2.1 double precision function `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.
----------	-------------

11.11 bml_transpose_m Module Reference

Transpose functions.

Functions/Subroutines

- subroutine [bml_transpose](#) (a, a_t)
Return the transpose of a matrix.

11.11.1 Detailed Description

Transpose functions.

11.11.2 Function/Subroutine Documentation

11.11.2.1 subroutine `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.

11.12 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` = "single-precision"
The single precision identifier.
- character(len=*), parameter `bml_precision_double` = "double-precision"
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.

11.12.1 Detailed Description

The basic bml types.

11.13 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.

11.13.1 Detailed Description

Utility matrix functions.

11.13.2 Function/Subroutine Documentation

- 11.13.2.1 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.

11.14 bml_utilities_matrix_type_m Module Reference

Utility matrix functions.

11.14.1 Detailed Description

Utility matrix functions.

Chapter 12

Class Documentation

12.1 `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.

12.1.1 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_m.F90`

12.2 `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.

12.2.1 Detailed Description

The bml vector type.

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

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

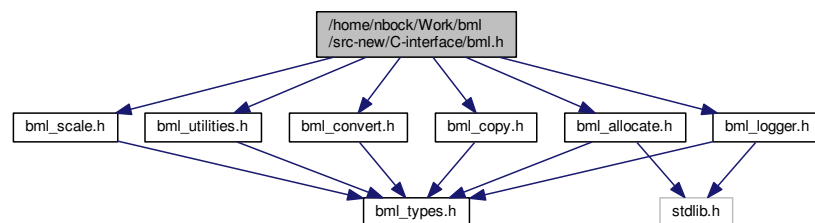
Chapter 13

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_scale.h"  
#include "bml_utilities.h"
```

Include dependency graph for bml.h:



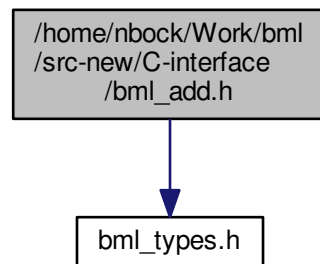
13.1.1 Detailed Description

Copyright

Los Alamos National Laboratory 2015

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

```
#include "bml_types.h"
Include dependency graph for bml_add.h:
```

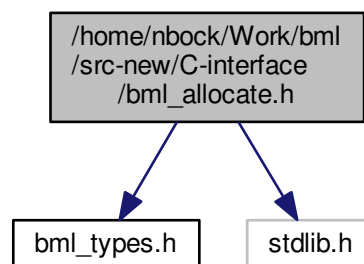


Functions

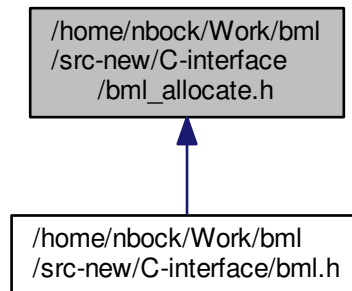
- void `bml_add` (const `bml_matrix_t` *A, const `bml_matrix_t` *B, const double alpha, const double beta, const double threshold)
- void `bml_add_identity` (const `bml_matrix_t` *A, const double beta, const double threshold)

13.3 /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:



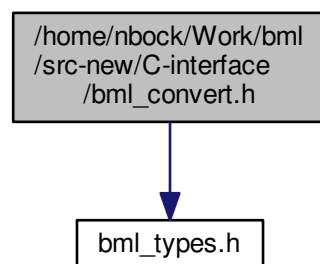
Functions

- 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)

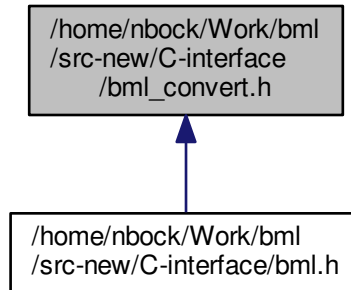
13.4 /home/nbock/Work/bml/src-new/C-interface/bml_convert.h File Reference

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

Include dependency graph for `bml_convert.h`:



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



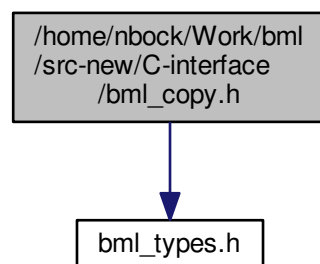
Functions

- `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)

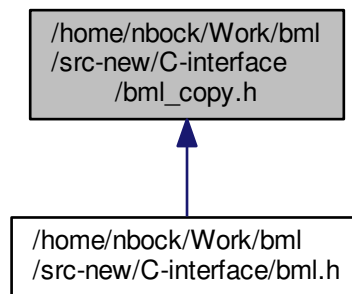
13.5 /home/nbock/Work/bml/src-new/C-interface/bml_copy.h File Reference

```
#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_new (const bml_matrix_t *A)`
- `void bml_copy (const bml_matrix_t *A, const bml_matrix_t *B)`

13.5.1 Function Documentation

13.5.1.1 `void bml_copy (const bml_matrix_t * A, const 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:



13.5.1.2 `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

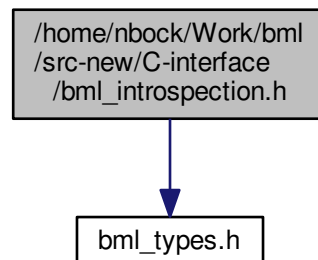
Here is the call graph for this function:



13.6 /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)

13.6.1 Function Documentation

13.6.1.1 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:

**13.6.1.2 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.

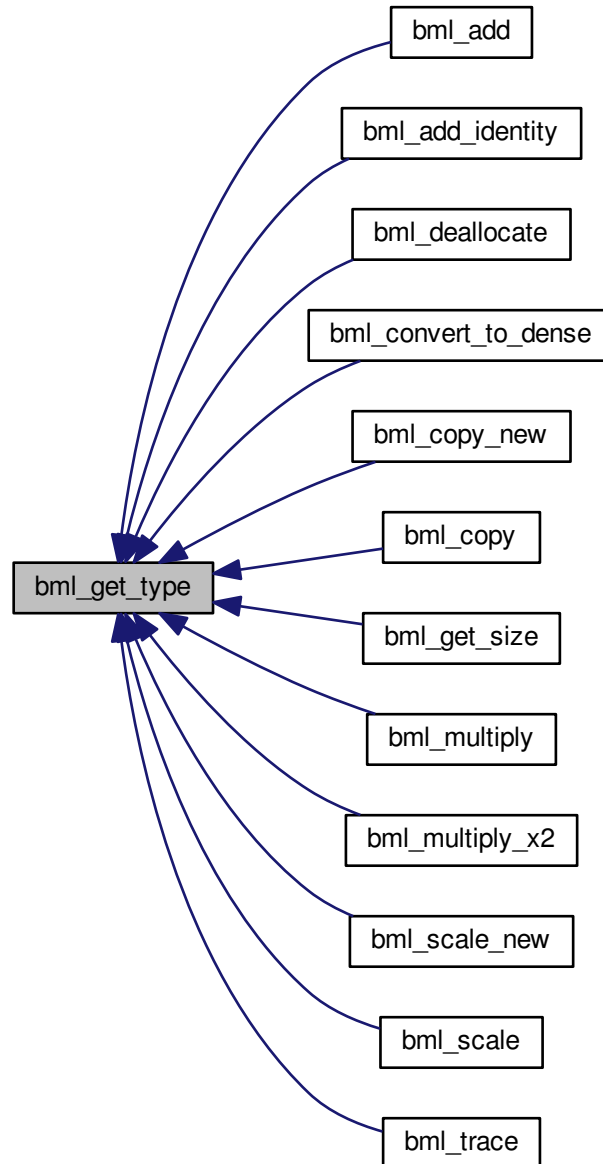
Parameters

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

Returns

The matrix type.

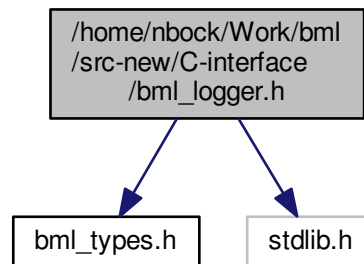
Here is the caller graph for this function:



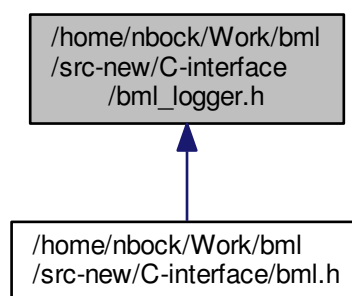
13.7 /home/nbock/Work/bml/src-new/C-interface/bml_logger.h File Reference

```
#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`,...)

13.7.1 Macro Definition Documentation

13.7.1.1 `#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.

13.7.1.2 `#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.

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

Convenience macro to write a BML_LOG_INFO level message.

13.7.1.4 `#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.

13.7.2 Enumeration Type Documentation

13.7.2.1 `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.

13.7.3 Function Documentation

13.7.3.1 `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()).
---------------	------------------------------

13.7.3.2 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..

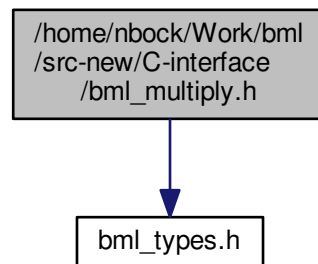
Parameters

<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()).

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

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

Include dependency graph for bml_multiply.h:



Functions

- void [bml_multiply](#) (const [bml_matrix_t](#) *A, const [bml_matrix_t](#) *B, const [bml_matrix_t](#) *C, const double alpha, const double beta, const double threshold)
- void [bml_multiply_x2](#) (const [bml_matrix_t](#) *X, const [bml_matrix_t](#) *X2, const double threshold)

13.8.1 Function Documentation

13.8.1.1 void bml_multiply (const [bml_matrix_t](#) * *A*, const [bml_matrix_t](#) * *B*, const [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:



13.8.1.2 void bml_multiply_x2 (const bml_matrix_t * X, const bml_matrix_t * X2, const double *threshold*)

Matrix multiply.

$X2 = X * X$

Parameters

<i>X</i>	Matrix X
<i>X2</i>	MatrixX2
<i>trX</i>	Trace of X
<i>trX2</i>	Trace of X2
<i>threshold</i>	Threshold for multiplication

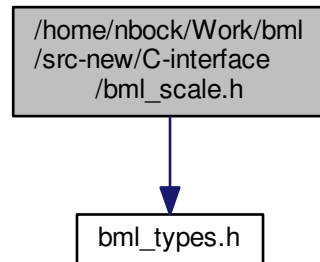
Here is the call graph for this function:



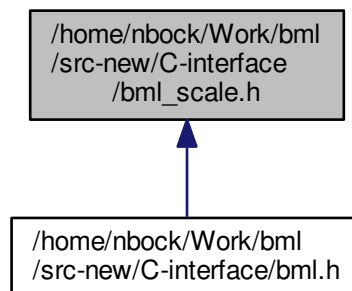
13.9 /home/nbock/Work/bml/src-new/C-interface/bml_scale.h File Reference

```
#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`, const `bml_matrix_t *B`)

13.9.1 Function Documentation

13.9.1.1 void `bml_scale` (const double *scale_factor*, const `bml_matrix_t * A`, const `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:



13.9.1.2 `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

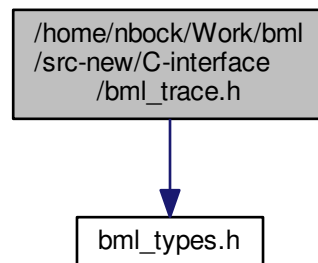
Here is the call graph for this function:



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

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

Include dependency graph for bml_trace.h:



Functions

- double [bml_trace](#) (const [bml_matrix_t](#) *A)

13.10.1 Function Documentation

13.10.1.1 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:



dense Dense matrix.

ellpack ELLPACK matrix.

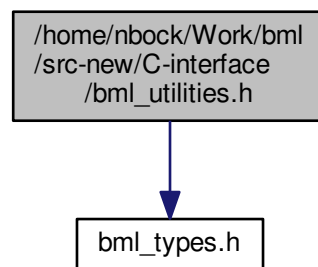
csr CSR matrix.

13.12 /home/nbock/Work/bml/src-new/C-interface/bml_types_private.h File Reference

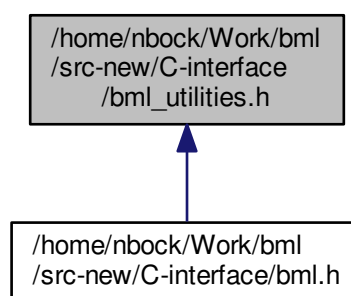
13.13 /home/nbock/Work/bml/src-new/C-interface/bml_utilities.h File Reference

```
#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_dense_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)
- 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)

13.13.1 Function Documentation

13.13.1.1 void bml_print_bml_matrix (const bml_matrix_t * *A*, const int *i_l*, const int *i_u*, const int *j_l*, const int *j_u*)

Print a dense matrix.

Parameters

<i>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.

13.13.1.2 void bml_print_bml_vector (const bml_vector_t * *v*, const int *i_l*, const int *i_u*)

Print a bml vector.

Parameters

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

13.13.1.3 void bml_print_dense_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](#), [45](#)
- [/home/nbock/Work/bml/src-new/C-interface/bml_add.h](#),
[46](#)
- [/home/nbock/Work/bml/src-new/C-interface/bml_↔](#)
[allocate.h](#), [46](#)
- [/home/nbock/Work/bml/src-new/C-interface/bml_↔](#)
[convert.h](#), [47](#)
- [/home/nbock/Work/bml/src-new/C-interface/bml_↔](#)
[copy.h](#), [48](#)
- [/home/nbock/Work/bml/src-new/C-interface/bml_↔](#)
[introspection.h](#), [50](#)
- [/home/nbock/Work/bml/src-new/C-interface/bml_↔](#)
[logger.h](#), [52](#)
- [/home/nbock/Work/bml/src-new/C-interface/bml_↔](#)
[multiply.h](#), [55](#)
- [/home/nbock/Work/bml/src-new/C-interface/bml_↔](#)
[scale.h](#), [56](#)
- [/home/nbock/Work/bml/src-new/C-interface/bml_↔](#)
[trace.h](#), [59](#)
- [/home/nbock/Work/bml/src-new/C-interface/bml_↔](#)
[types.h](#), [60](#)
- [/home/nbock/Work/bml/src-new/C-interface/bml_↔](#)
[types_private.h](#), [61](#)
- [/home/nbock/Work/bml/src-new/C-interface/bml_↔](#)
[utilities.h](#), [61](#)
- Add Functions (C interface), [22](#)
 - [bml_add](#), [22](#)
 - [bml_add_identity](#), [22](#)
- Add Functions (Fortran interface), [29](#)
- 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), [26](#)
 - [bml_deallocate](#), [26](#)
 - [bml_identity_matrix](#), [26](#)
 - [bml_random_matrix](#), [26](#)
 - [bml_zero_matrix](#), [26](#)
- BML_LOG_DEBUG
 - [bml_logger.h](#), [54](#)
- BML_LOG_ERROR
 - [bml_logger.h](#), [54](#)
- BML_LOG_INFO
 - [bml_logger.h](#), [54](#)
- BML_LOG_WARNING
 - [bml_logger.h](#), [54](#)
- [bml](#), [33](#)
- [bml_add](#)
 - Add Functions (C interface), [22](#)
- [bml_add_identity](#)
 - Add Functions (C interface), [22](#)
- [bml_allocate_m](#), [33](#)
- [bml_allocate_memory](#)
 - Allocation and Deallocation Functions (C interface),
[19](#)
- [bml_convert_from_dense](#)
 - Converting between Matrix Formats (C interface),
[24](#)
- [bml_convert_from_dense_double](#)
 - Converting between Matrix Formats (Fortran interface), [30](#)
- [bml_convert_from_dense_double_complex](#)
 - Converting between Matrix Formats (Fortran interface), [30](#)
- [bml_convert_from_dense_single_complex](#)
 - Converting between Matrix Formats (Fortran interface), [31](#)
- [bml_convert_to_dense](#)
 - Converting between Matrix Formats (C interface),
[24](#)
- [bml_convert_to_dense_double](#)
 - Converting between Matrix Formats (Fortran interface), [31](#)
- [bml_convert_to_dense_double_complex](#)
 - Converting between Matrix Formats (Fortran interface), [31](#)
- [bml_convert_to_dense_single](#)
 - Converting between Matrix Formats (Fortran interface), [31](#)
- [bml_convert_to_dense_single_complex](#)
 - Converting between Matrix Formats (Fortran interface), [31](#)
- [bml_copy](#)
 - [bml_copy.h](#), [49](#)
 - [bml_copy_m](#), [34](#)
- [bml_copy.h](#)
 - [bml_copy](#), [49](#)
 - [bml_copy_new](#), [49](#)
- [bml_copy_m](#), [33](#)
 - [bml_copy](#), [34](#)
- [bml_copy_new](#)
 - [bml_copy.h](#), [49](#)
- [bml_deallocate](#)

- Allocation and Deallocation Functions (C interface), 19
- Allocation and Deallocation Functions (Fortran interface), 26
- bml_debug
 - bml_error_m, 35
- bml_diagonalize
 - bml_diagonalize_m, 34
- bml_diagonalize_m, 34
 - bml_diagonalize, 34
- bml_error
 - bml_error_m, 35
- bml_error_m, 34
 - bml_debug, 35
 - bml_error, 35
 - bml_warning, 35
- bml_free_memory
 - Allocation and Deallocation Functions (C interface), 20
- bml_get_bandwidth
 - bml_introspection_m, 37
- bml_get_size
 - bml_introspection.h, 50
 - bml_introspection_m, 37
- bml_get_type
 - bml_introspection.h, 51
- bml_identity_matrix
 - Allocation and Deallocation Functions (C interface), 20
 - Allocation and Deallocation Functions (Fortran interface), 26
- bml_interface_m, 36
 - get_enum_id, 36
- bml_introspection.h
 - bml_get_size, 50
 - bml_get_type, 51
- bml_introspection_m, 36
 - bml_get_bandwidth, 37
 - bml_get_size, 37
- bml_log
 - bml_logger.h, 54
- bml_log_level_t
 - bml_logger.h, 54
- bml_log_location
 - bml_logger.h, 55
- bml_logger.h
 - BML_LOG_DEBUG, 54
 - BML_LOG_ERROR, 54
 - BML_LOG_INFO, 54
 - BML_LOG_WARNING, 54
 - bml_log, 54
 - bml_log_level_t, 54
 - bml_log_location, 55
 - LOG_DEBUG, 54
 - LOG_ERROR, 54
 - LOG_INFO, 54
 - LOG_WARN, 54
- bml_matrix_precision_t
 - bml_types.h, 60
- bml_matrix_t
 - bml_types.h, 60
- bml_matrix_type_t
 - bml_types.h, 60
- bml_multiply
 - bml_multiply.h, 55
 - bml_multiply_m, 38
- bml_multiply.h
 - bml_multiply, 55
 - bml_multiply_x2, 56
- bml_multiply_m, 37
 - bml_multiply, 38
- bml_multiply_x2
 - bml_multiply.h, 56
- bml_print_bml_matrix
 - bml_utilities.h, 62
- bml_print_bml_vector
 - bml_utilities.h, 62
 - bml_utilities_m, 40
- bml_print_dense_matrix
 - bml_utilities.h, 62
- bml_random_matrix
 - Allocation and Deallocation Functions (C interface), 20
 - Allocation and Deallocation Functions (Fortran interface), 26
- bml_scale
 - bml_scale.h, 57
- bml_scale.h
 - bml_scale, 57
 - bml_scale_new, 58
- bml_scale_m, 38
 - scale_two, 38
- bml_scale_new
 - bml_scale.h, 58
- bml_trace
 - bml_trace.h, 59
 - bml_trace_m, 39
- bml_trace.h
 - bml_trace, 59
- bml_trace_m, 38
 - bml_trace, 39
- bml_transpose
 - bml_transpose_m, 39
- bml_transpose_m, 39
 - bml_transpose, 39
- bml_types.h
 - bml_matrix_precision_t, 60
 - bml_matrix_t, 60
 - bml_matrix_type_t, 60
 - bml_vector_t, 60
 - csr, 61
 - dense, 60
 - double_complex, 60
 - double_real, 60
 - ellpack, 61
 - single_complex, 60

- single_real, [60](#)
 - uninitialized, [60](#)
- bml_types_m, [39](#)
- bml_types_m::bml_matrix_t, [43](#)
- bml_types_m::bml_vector_t, [43](#)
- bml_utilities.h
 - bml_print_bml_matrix, [62](#)
 - bml_print_bml_vector, [62](#)
 - bml_print_dense_matrix, [62](#)
- bml_utilities_m, [40](#)
 - bml_print_bml_vector, [40](#)
- bml_utilities_matrix_type_m, [41](#)
- bml_vector_t
 - bml_types.h, [60](#)
- bml_warning
 - bml_error_m, [35](#)
- bml_zero_matrix
 - Allocation and Deallocation Functions (C interface), [21](#)
 - Allocation and Deallocation Functions (Fortran interface), [26](#)
- Converting between Matrix Formats (C interface), [24](#)
 - bml_convert_from_dense, [24](#)
 - bml_convert_to_dense, [24](#)
- Converting between Matrix Formats (Fortran interface), [30](#)
 - bml_convert_from_dense_double, [30](#)
 - bml_convert_from_dense_double_complex, [30](#)
 - bml_convert_from_dense_single_complex, [31](#)
 - bml_convert_to_dense_double, [31](#)
 - bml_convert_to_dense_double_complex, [31](#)
 - bml_convert_to_dense_single, [31](#)
 - bml_convert_to_dense_single_complex, [31](#)
- csr
 - bml_types.h, [61](#)
- dense
 - bml_types.h, [60](#)
- double_complex
 - bml_types.h, [60](#)
- double_real
 - bml_types.h, [60](#)
- ellpack
 - bml_types.h, [61](#)
- get_enum_id
 - bml_interface_m, [36](#)
- LOG_DEBUG
 - bml_logger.h, [54](#)
- LOG_ERROR
 - bml_logger.h, [54](#)
- LOG_INFO
 - bml_logger.h, [54](#)
- LOG_WARN
 - bml_logger.h, [54](#)
- scale_two