

bml

1.2.3

Generated by Doxygen 1.8.13

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	Class Index	13
7.1	Class List	13
8	File Index	15
8.1	File List	15

9	Module Documentation	17
9.1	Allocation and Deallocation Functions (C interface)	17
9.1.1	Detailed Description	17
9.1.2	Function Documentation	17
9.1.2.1	bml_allocate_memory()	17
9.1.2.2	bml_allocated()	18
9.1.2.3	bml_banded_matrix()	18
9.1.2.4	bml_clear()	19
9.1.2.5	bml_deallocate()	19
9.1.2.6	bml_deallocate_domain()	19
9.1.2.7	bml_default_domain()	19
9.1.2.8	bml_free_memory()	20
9.1.2.9	bml_free_ptr()	20
9.1.2.10	bml_identity_matrix()	21
9.1.2.11	bml_noinit_allocate_memory()	21
9.1.2.12	bml_noinit_matrix()	21
9.1.2.13	bml_random_matrix()	22
9.1.2.14	bml_update_domain()	23
9.1.2.15	bml_zero_matrix()	23
9.2	Add Functions (C interface)	24
9.2.1	Detailed Description	24
9.2.2	Function Documentation	24
9.2.2.1	bml_add()	24
9.2.2.2	bml_add_identity()	24
9.2.2.3	bml_add_norm()	25
9.2.2.4	bml_scale_add_identity()	25
9.3	Converting between Matrix Formats (C interface)	26
9.3.1	Detailed Description	26
9.3.2	Function Documentation	26
9.3.2.1	bml_export_to_dense()	26
9.3.2.2	bml_import_from_dense()	26
9.4	Allocation and Deallocation Functions (Fortran interface)	28
9.5	Add Functions (Fortran interface)	29
9.6	Converting between Matrix Formats (Fortran interface)	30

10 Class Documentation	31
10.1 bml_domain_t Struct Reference	31
10.1.1 Detailed Description	31
10.1.2 Member Data Documentation	31
10.1.2.1 globalRowExtent	31
10.1.2.2 globalRowMax	32
10.1.2.3 globalRowMin	32
10.1.2.4 localDispl	32
10.1.2.5 localElements	32
10.1.2.6 localRowExtent	32
10.1.2.7 localRowMax	32
10.1.2.8 localRowMin	32
10.1.2.9 maxLocalExtent	32
10.1.2.10 minLocalExtent	33
10.1.2.11 totalCols	33
10.1.2.12 totalProcs	33
10.1.2.13 totalRows	33
11 File Documentation	35
11.1 /home/christian/bml/src/C-interface/bml.h File Reference	35
11.1.1 Detailed Description	35
11.2 /home/christian/bml/src/C-interface/bml_add.h File Reference	35
11.3 /home/christian/bml/src/C-interface/bml_adjungate_triangle.h File Reference	36
11.3.1 Function Documentation	36
11.3.1.1 bml_adjungate_triangle()	36
11.4 /home/christian/bml/src/C-interface/bml_allocate.h File Reference	36
11.5 /home/christian/bml/src/C-interface/bml_convert.h File Reference	37
11.5.1 Function Documentation	37
11.5.1.1 bml_convert()	37
11.6 /home/christian/bml/src/C-interface/bml_copy.h File Reference	38
11.6.1 Function Documentation	38

11.6.1.1	bml_copy()	38
11.6.1.2	bml_copy_domain()	38
11.6.1.3	bml_copy_new()	39
11.6.1.4	bml_reorder()	39
11.6.1.5	bml_restore_domain()	39
11.6.1.6	bml_save_domain()	40
11.7	/home/christian/bml/src/C-interface/bml_export.h File Reference	40
11.8	/home/christian/bml/src/C-interface/bml_getters.h File Reference	40
11.8.1	Function Documentation	40
11.8.1.1	bml_get()	40
11.8.1.2	bml_get_diagonal()	41
11.8.1.3	bml_get_row()	41
11.9	/home/christian/bml/src/C-interface/bml_import.h File Reference	42
11.10	/home/christian/bml/src/C-interface/bml_init.h File Reference	42
11.10.1	Function Documentation	42
11.10.1.1	bml_init()	42
11.10.1.2	bml_initF()	42
11.11	/home/christian/bml/src/C-interface/bml_introspection.h File Reference	43
11.11.1	Function Documentation	43
11.11.1.1	bml_get_bandwidth()	43
11.11.1.2	bml_get_distribution_mode()	43
11.11.1.3	bml_get_M()	44
11.11.1.4	bml_get_N()	44
11.11.1.5	bml_get_precision()	44
11.11.1.6	bml_get_row_bandwidth()	45
11.11.1.7	bml_get_sparsity()	45
11.11.1.8	bml_get_type()	46
11.12	/home/christian/bml/src/C-interface/bml_logger.h File Reference	46
11.12.1	Macro Definition Documentation	46
11.12.1.1	LOG_DEBUG	47

11.12.1.2 LOG_ERROR	47
11.12.1.3 LOG_INFO	47
11.12.1.4 LOG_WARN	47
11.12.2 Enumeration Type Documentation	47
11.12.2.1 bml_log_level_t	47
11.12.3 Function Documentation	48
11.12.3.1 bml_log()	48
11.12.3.2 bml_log_location()	48
11.13/home/christian/bml/src/C-interface/bml_multiply.h File Reference	48
11.13.1 Function Documentation	49
11.13.1.1 bml_multiply()	49
11.13.1.2 bml_multiply_AB()	49
11.13.1.3 bml_multiply_adjust_AB()	50
11.13.1.4 bml_multiply_x2()	50
11.14/home/christian/bml/src/C-interface/bml_norm.h File Reference	50
11.14.1 Function Documentation	51
11.14.1.1 bml_fnorm()	51
11.14.1.2 bml_fnorm2()	51
11.14.1.3 bml_sum_squares()	52
11.14.1.4 bml_sum_squares2()	52
11.14.1.5 bml_sum_squares_submatrix()	52
11.15/home/christian/bml/src/C-interface/bml_normalize.h File Reference	53
11.15.1 Function Documentation	53
11.15.1.1 bml_gershgorin()	53
11.15.1.2 bml_gershgorin_partial()	53
11.15.1.3 bml_normalize()	54
11.16/home/christian/bml/src/C-interface/bml_parallel.h File Reference	54
11.16.1 Function Documentation	54
11.16.1.1 bml_allGatherVParallel()	55
11.16.1.2 bml_getMyRank()	55

11.16.1.3 bml_getNRanks()	55
11.17/home/christian/bml/src/C-interface/bml_scale.h File Reference	55
11.17.1 Function Documentation	55
11.17.1.1 bml_scale()	56
11.17.1.2 bml_scale_inplace()	56
11.17.1.3 bml_scale_new()	56
11.18/home/christian/bml/src/C-interface/bml_setters.h File Reference	57
11.19/home/christian/bml/src/C-interface/bml_shutdown.h File Reference	57
11.19.1 Function Documentation	57
11.19.1.1 bml_shutdown()	57
11.19.1.2 bml_shutdownF()	57
11.20/home/christian/bml/src/C-interface/bml_submatrix.h File Reference	57
11.20.1 Function Documentation	58
11.20.1.1 bml_adjacency()	58
11.20.1.2 bml_adjacency_group()	58
11.20.1.3 bml_group_matrix()	59
11.20.1.4 bml_matrix2submatrix()	59
11.20.1.5 bml_matrix2submatrix_index()	60
11.20.1.6 bml_matrix2submatrix_index_graph()	60
11.20.1.7 bml_submatrix2matrix()	60
11.21/home/christian/bml/src/C-interface/bml_threshold.h File Reference	61
11.21.1 Function Documentation	61
11.21.1.1 bml_threshold()	61
11.21.1.2 bml_threshold_new()	62
11.22/home/christian/bml/src/C-interface/bml_trace.h File Reference	62
11.22.1 Function Documentation	62
11.22.1.1 bml_trace()	62
11.23/home/christian/bml/src/C-interface/bml_transpose.h File Reference	63
11.23.1 Function Documentation	63
11.23.1.1 bml_transpose()	63

11.23.1.2 <code>bml_transpose_new()</code>	63
11.24/home/christian/bml/src/C-interface/bml_transpose_triangle.h File Reference	64
11.24.1 Function Documentation	64
11.24.1.1 <code>bml_transpose_triangle()</code>	64
11.25/home/christian/bml/src/C-interface/bml_types.h File Reference	64
11.25.1 Typedef Documentation	65
11.25.1.1 <code>bml_matrix_t</code>	65
11.25.1.2 <code>bml_vector_t</code>	65
11.25.2 Enumeration Type Documentation	65
11.25.2.1 <code>bml_dense_order_t</code>	65
11.25.2.2 <code>bml_distribution_mode_t</code>	66
11.25.2.3 <code>bml_matrix_precision_t</code>	66
11.25.2.4 <code>bml_matrix_type_t</code>	66
11.26/home/christian/bml/src/C-interface/bml_types_private.h File Reference	66
11.27/home/christian/bml/src/C-interface/bml_utilities.h File Reference	67
11.27.1 Function Documentation	67
11.27.1.1 <code>bml_print_bml_matrix()</code>	67
11.27.1.2 <code>bml_print_bml_vector()</code>	68
11.27.1.3 <code>bml_print_dense_matrix()</code>	68
11.27.1.4 <code>bml_print_dense_vector()</code>	68
11.27.1.5 <code>bml_read_bml_matrix()</code>	69
11.27.1.6 <code>bml_write_bml_matrix()</code>	69
Index	71

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

Bálint Aradi aradi@uni-bremen.de
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_import::bml_import_from_dense`
 - `bml_export::bml_export_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)	17
Add Functions (C interface)	24
Converting between Matrix Formats (C interface)	26
Allocation and Deallocation Functions (Fortran interface)	28
Add Functions (Fortran interface)	29
Converting between Matrix Formats (Fortran interface)	30

Chapter 7

Class Index

7.1 Class List

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

bml_domain_t	31
--	----

Chapter 8

File Index

8.1 File List

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

/home/christian/bml/src/C-interface/ blas.h	??
/home/christian/bml/src/C-interface/ bml.h	35
/home/christian/bml/src/C-interface/ bml_add.h	35
/home/christian/bml/src/C-interface/ bml_adjungate_triangle.h	36
/home/christian/bml/src/C-interface/ bml_allocate.h	36
/home/christian/bml/src/C-interface/ bml_convert.h	37
/home/christian/bml/src/C-interface/ bml_copy.h	38
/home/christian/bml/src/C-interface/ bml_diagonalize.h	??
/home/christian/bml/src/C-interface/ bml_elemental.h	??
/home/christian/bml/src/C-interface/ bml_export.h	40
/home/christian/bml/src/C-interface/ bml_getters.h	40
/home/christian/bml/src/C-interface/ bml_import.h	42
/home/christian/bml/src/C-interface/ bml_init.h	42
/home/christian/bml/src/C-interface/ bml_introspection.h	43
/home/christian/bml/src/C-interface/ bml_inverse.h	??
/home/christian/bml/src/C-interface/ bml_logger.h	46
/home/christian/bml/src/C-interface/ bml_multiply.h	48
/home/christian/bml/src/C-interface/ bml_norm.h	50
/home/christian/bml/src/C-interface/ bml_normalize.h	53
/home/christian/bml/src/C-interface/ bml_parallel.h	54
/home/christian/bml/src/C-interface/ bml_scale.h	55
/home/christian/bml/src/C-interface/ bml_setters.h	57
/home/christian/bml/src/C-interface/ bml_shutdown.h	57
/home/christian/bml/src/C-interface/ bml_submatrix.h	57
/home/christian/bml/src/C-interface/ bml_threshold.h	61
/home/christian/bml/src/C-interface/ bml_trace.h	62
/home/christian/bml/src/C-interface/ bml_transpose.h	63
/home/christian/bml/src/C-interface/ bml_transpose_triangle.h	64
/home/christian/bml/src/C-interface/ bml_types.h	64
/home/christian/bml/src/C-interface/ bml_types_private.h	66
/home/christian/bml/src/C-interface/ bml_utilities.h	67
/home/christian/bml/src/C-interface/ lapack.h	??

Chapter 9

Module Documentation

9.1 Allocation and Deallocation Functions (C interface)

Functions

- int `bml_allocated` (const `bml_matrix_t` *A)
- void * `bml_noinit_allocate_memory` (const `size_t` size)
- void * `bml_allocate_memory` (const `size_t` size)
- void `bml_free_memory` (void *ptr)
- void `bml_free_ptr` (void **ptr)
- void `bml_deallocate_domain` (`bml_domain_t` *D)
- void `bml_deallocate` (`bml_matrix_t` **A)
- void `bml_clear` (`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, const `bml_distribution_mode_t` distrib_mode)
- `bml_matrix_t` * `bml_noinit_matrix` (const `bml_matrix_type_t` matrix_type, const `bml_matrix_precision_t` matrix_precision, const int N, const int M, const `bml_distribution_mode_t` distrib_mode)
- `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, const `bml_distribution_mode_t` distrib_mode)
- `bml_matrix_t` * `bml_banded_matrix` (const `bml_matrix_type_t` matrix_type, const `bml_matrix_precision_t` matrix_precision, const int N, const int M, const `bml_distribution_mode_t` distrib_mode)
- `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, const `bml_distribution_mode_t` distrib_mode)
- `bml_domain_t` * `bml_default_domain` (const int N, const int M, const `bml_distribution_mode_t` distrib_mode)
- void `bml_update_domain` (`bml_matrix_t` *A, int *localPartMin, int *localPartMax, int *nnodesInPart)

9.1.1 Detailed Description

9.1.2 Function Documentation

9.1.2.1 `bml_allocate_memory()`

```
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.

9.1.2.2 bml_allocated()

```
int bml_allocated (
    const bml_matrix_t * A )
```

Check if matrix is allocated.

Parameters

<i>A</i>	Matrix
----------	--------

Returns

>0 if allocated, else -1

9.1.2.3 bml_banded_matrix()

```
bml_matrix_t* bml_banded_matrix (
    const bml_matrix_type_t matrix_type,
    const bml_matrix_precision_t matrix_precision,
    const int N,
    const int M,
    const bml_distribution_mode_t distrib_mode )
```

Allocate a banded 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 bandwidth of the matrix.
<i>distrib_mode</i>	The distribution mode.

Returns

The matrix.

9.1.2.4 bml_clear()

```
void bml_clear (
    bml_matrix_t * A )
```

Clear a matrix.

Parameters

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

9.1.2.5 bml_deallocate()

```
void bml_deallocate (
    bml_matrix_t ** A )
```

Deallocate a matrix.

Parameters

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

9.1.2.6 bml_deallocate_domain()

```
void bml_deallocate_domain (
    bml_domain_t * D )
```

Deallocate a domain.

Parameters

<i>D</i>	The domain.
----------	-------------

9.1.2.7 bml_default_domain()

```
bml_domain_t* bml_default_domain (
    const int N,
```

```
const int M,
const bml_distribution_mode_t distrib_mode )
```

Allocate a default domain for a bml matrix.

Parameters

<i>N</i>	The number of rows
<i>M</i>	The number of columns
<i>distrib_mode</i>	The distribution mode

Returns

The domain

For first rank

For middle ranks

For last rank

Number of elements and displacement per rank

9.1.2.8 bml_free_memory()

```
void bml_free_memory (
    void * ptr )
```

Deallocate a chunk of memory.

Parameters

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

9.1.2.9 bml_free_ptr()

```
void bml_free_ptr (
    void ** ptr )
```

De-allocate a chunk of memory that was allocated inside a C function.

Parameters

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

9.1.2.10 `bml_identity_matrix()`

```
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,
    const bml_distribution_mode_t distrib_mode )
```

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.
<i>distrib_mode</i>	The distribution mode.

Returns

The matrix.

9.1.2.11 `bml_noinit_allocate_memory()`

```
void* bml_noinit_allocate_memory (
    const size_t size )
```

Allocate a chunk of memory without initialization.

Parameters

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

Returns

A pointer to the allocated chunk.

9.1.2.12 `bml_noinit_matrix()`

```
bml_matrix_t* bml_noinit_matrix (
    const bml_matrix_type_t matrix_type,
    const bml_matrix_precision_t matrix_precision,
```

```

const int N,
const int M,
const bml_distribution_mode_t distrib_mode )

```

Allocate a matrix without initializing.

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.
<i>distrib_mode</i>	The distribution mode.

Returns

The matrix.

9.1.2.13 bml_random_matrix()

```

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,
    const bml_distribution_mode_t distrib_mode )

```

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.
<i>M</i>	The number of non-zeroes per row.
<i>distrib_mode</i>	The distribution mode.

Returns

The matrix.

9.1.2.14 bml_update_domain()

```
void bml_update_domain (
    bml_matrix_t * A,
    int * localPartMin,
    int * localPartMax,
    int * nnodesInPart )
```

Update a domain for a bml matrix.

Parameters

<i>A</i>	Matrix with domain
<i>localPartMin</i>	First part on each rank
<i>localPartMax</i>	Last part on each rank
<i>nnodesInPart</i>	Number of nodes in each part

9.1.2.15 bml_zero_matrix()

```
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,
    const bml_distribution_mode_t distrib_mode )
```

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.
<i>N</i>	The matrix size.
<i>M</i>	The number of non-zeroes per row.
<i>distrib_mode</i>	The distribution mode.

Returns

The matrix.

9.2 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)
- double `bml_add_norm` (`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)
- void `bml_scale_add_identity` (`bml_matrix_t` *A, const double alpha, const double beta, const double threshold)

9.2.1 Detailed Description

9.2.2 Function Documentation

9.2.2.1 `bml_add()`

```
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

9.2.2.2 `bml_add_identity()`

```
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 I
<i>threshold</i>	Threshold for matrix addition

9.2.2.3 bml_add_norm()

```
double bml_add_norm (
    bml_matrix_t * A,
    const bml_matrix_t * B,
    const double alpha,
    const double beta,
    const double threshold )
```

Matrix addition with calculation of TrNorm.

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

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

9.2.2.4 bml_scale_add_identity()

```
void bml_scale_add_identity (
    bml_matrix_t * A,
    const double alpha,
    const double beta,
    const double threshold )
```

Matrix addition.

$$A \leftarrow \alpha A + \beta Id$$

Parameters

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

9.3 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 int M, const void *A, const double threshold, const `bml_distribution_mode_t` distrib_mode)

9.3.1 Detailed Description

9.3.2 Function Documentation

9.3.2.1 `bml_export_to_dense()`

```
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_export_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

9.3.2.2 `bml_import_from_dense()`

```
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 int M,  
const void * A,  
const double threshold,  
const bml_distribution_mode_t distrib_mode )
```

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>M</i>	The number of non-zeroes per row
<i>A</i>	The dense matrix
<i>threshold</i>	The matrix element magnited threshold

Returns

The bml matrix

9.4 Allocation and Deallocation Functions (Fortran interface)

9.5 Add Functions (Fortran interface)

9.6 Converting between Matrix Formats (Fortran interface)

Chapter 10

Class Documentation

10.1 bml_domain_t Struct Reference

```
#include <bml_types.h>
```

Public Attributes

- int [totalProcs](#)
- int [totalRows](#)
- int [totalCols](#)
- int [globalRowMin](#)
- int [globalRowMax](#)
- int [globalRowExtent](#)
- int [maxLocalExtent](#)
- int [minLocalExtent](#)
- int * [localRowMin](#)
- int * [localRowMax](#)
- int * [localRowExtent](#)
- int * [localElements](#)
- int * [localDispl](#)

10.1.1 Detailed Description

Decomposition for working in parallel.

10.1.2 Member Data Documentation

10.1.2.1 globalRowExtent

```
int bml_domain_t::globalRowExtent
```

global total rows

10.1.2.2 globalRowMax

```
int bml_domain_t::globalRowMax
```

global maximum row number

10.1.2.3 globalRowMin

```
int bml_domain_t::globalRowMin
```

global minimum row number

10.1.2.4 localDispl

```
int* bml_domain_t::localDispl
```

local displacements per rank for 2D

10.1.2.5 localElements

```
int* bml_domain_t::localElements
```

local number of elements per rank

10.1.2.6 localRowExtent

```
int* bml_domain_t::localRowExtent
```

extent of rows per rank, localRowMax - localRowMin

10.1.2.7 localRowMax

```
int* bml_domain_t::localRowMax
```

maximum row per rank

10.1.2.8 localRowMin

```
int* bml_domain_t::localRowMin
```

minimum row per rank

10.1.2.9 maxLocalExtent

```
int bml_domain_t::maxLocalExtent
```

maximum extent for most processors

10.1.2.10 minLocalExtent

```
int bml_domain_t::minLocalExtent
```

minimum extent for last processors

10.1.2.11 totalCols

```
int bml_domain_t::totalCols
```

total number of columns

10.1.2.12 totalProcs

```
int bml_domain_t::totalProcs
```

number of processors

10.1.2.13 totalRows

```
int bml_domain_t::totalRows
```

total number of rows

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

- [/home/christian/bml/src/C-interface/bml_types.h](#)

Chapter 11

File Documentation

11.1 /home/christian/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_getters.h"
#include "bml_import.h"
#include "bml_init.h"
#include "bml_introspection.h"
#include "bml_inverse.h"
#include "bml_logger.h"
#include "bml_multiply.h"
#include "bml_normalize.h"
#include "bml_norm.h"
#include "bml_parallel.h"
#include "bml_scale.h"
#include "bml_setters.h"
#include "bml_shutdown.h"
#include "bml_submatrix.h"
#include "bml_threshold.h"
#include "bml_trace.h"
#include "bml_transpose.h"
#include "bml_utilities.h"
```

11.1.1 Detailed Description

Copyright

Los Alamos National Laboratory 2015

11.2 /home/christian/bml/src/C-interface/bml_add.h File Reference

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

Functions

- void [bml_add](#) ([bml_matrix_t](#) *A, const [bml_matrix_t](#) *B, const double alpha, const double beta, const double threshold)
- double [bml_add_norm](#) ([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)
- void [bml_scale_add_identity](#) ([bml_matrix_t](#) *A, const double alpha, const double beta, const double threshold)

11.3 /home/christian/bml/src/C-interface/bml_adjungate_triangle.h File Reference

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

Functions

- void [bml_adjungate_triangle](#) ([bml_matrix_t](#) *A, char *triangle)

11.3.1 Function Documentation

11.3.1.1 [bml_adjungate_triangle\(\)](#)

```
void bml_adjungate_triangle (
    bml\_matrix\_t * A,
    char * triangle )
```

Adjungates (conjugate transpose) a triangle of a matrix in place.

Parameters

<i>A</i>	The matrix for which the triangle should be adjungated
<i>triangle</i>	Which triangle to adjungate ('u': upper, 'l': lower)

11.4 /home/christian/bml/src/C-interface/bml_allocate.h File Reference

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

Functions

- void * [bml_allocate_memory](#) (const size_t s)

- void * [bml_noinit_allocate_memory](#) (const size_t s)
- void [bml_free_memory](#) (void *ptr)
- void [bml_free_ptr](#) (void **ptr)
- void [bml_deallocate](#) ([bml_matrix_t](#) **A)
- void [bml_clear](#) ([bml_matrix_t](#) *A)
- int [bml_allocated](#) (const [bml_matrix_t](#) *A)
- [bml_matrix_t](#) * [bml_noinit_matrix](#) (const [bml_matrix_type_t](#) matrix_type, const [bml_matrix_precision_t](#) matrix_precision, const int N, const int M, const [bml_distribution_mode_t](#) distrib_mode)
- [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, const [bml_distribution_mode_t](#) distrib_mode)
- [bml_matrix_t](#) * [bml_banded_matrix](#) (const [bml_matrix_type_t](#) matrix_type, const [bml_matrix_precision_t](#) matrix_precision, const int N, const int M, const [bml_distribution_mode_t](#) distrib_mode)
- [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, const [bml_distribution_mode_t](#) distrib_mode)
- [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, const [bml_distribution_mode_t](#) distrib_mode)
- void [bml_deallocate_domain](#) ([bml_domain_t](#) *D)
- [bml_domain_t](#) * [bml_default_domain](#) (const int N, const int M, const [bml_distribution_mode_t](#) distrib_mode)
- void [bml_update_domain](#) ([bml_matrix_t](#) *A, int *localPartMin, int *localPartMax, int *nnodesInPart)

11.5 /home/christian/bml/src/C-interface/bml_convert.h File Reference

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

Functions

- [bml_matrix_t](#) * [bml_convert](#) (const [bml_matrix_t](#) *A, const [bml_matrix_type_t](#) matrix_type, const [bml_matrix_precision_t](#) matrix_precision, const int M, const [bml_distribution_mode_t](#) distrib_mode)

11.5.1 Function Documentation

11.5.1.1 [bml_convert\(\)](#)

```
bml\_matrix\_t* bml\_convert (
    const bml\_matrix\_t * A,
    const bml\_matrix\_type\_t matrix_type,
    const bml\_matrix\_precision\_t matrix_precision,
    const int M,
    const bml\_distribution\_mode\_t distrib_mode )
```

Convert a bml matrix to another type.

$A \rightarrow B$

Parameters

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

Returns

The converted matrix *B*.

11.6 /home/christian/bml/src/C-interface/bml_copy.h File Reference

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

Functions

- [bml_matrix_t * bml_copy_new](#) (const [bml_matrix_t](#) *A)
- void [bml_copy](#) (const [bml_matrix_t](#) *A, [bml_matrix_t](#) *B)
- void [bml_reorder](#) ([bml_matrix_t](#) *A, int *perm)
- void [bml_copy_domain](#) (const [bml_domain_t](#) *A, [bml_domain_t](#) *B)
- void [bml_save_domain](#) ([bml_matrix_t](#) *A)
- void [bml_restore_domain](#) ([bml_matrix_t](#) *A)

11.6.1 Function Documentation

11.6.1.1 [bml_copy\(\)](#)

```
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

11.6.1.2 [bml_copy_domain\(\)](#)

```
void bml_copy_domain (
    const bml\_domain\_t * A,
    bml\_domain\_t * B )
```


Copy a domain.

Parameters

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

11.6.1.3 bml_copy_new()

```
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

11.6.1.4 bml_reorder()

```
void bml_reorder (
    bml_matrix_t * A,
    int * perm )
```

Reorder a matrix in place.

Parameters

<i>A</i>	Matrix to reorder
<i>perm</i>	permutation vector for reordering

11.6.1.5 bml_restore_domain()

```
void bml_restore_domain (
    bml_matrix_t * A )
```

Restore to saved domain for bml matrix.

Parameters

<i>A</i>	Matrix with domain
----------	--------------------

11.6.1.6 `bml_save_domain()`

```
void bml_save_domain (
    bml_matrix_t * A )
```

Save current domain for bml matrix.

Parameters

<i>A</i>	Matrix with domain
----------	--------------------

11.7 `/home/christian/bml/src/C-interface/bml_export.h` File Reference

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

Functions

- void * `bml_export_to_dense` (const `bml_matrix_t` *A, const `bml_dense_order_t` order)

11.8 `/home/christian/bml/src/C-interface/bml_getters.h` File Reference

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

Functions

- void * `bml_get` (const `bml_matrix_t` *A, const int i, const int j)
- void * `bml_get_row` (`bml_matrix_t` *A, const int i)
- void * `bml_get_diagonal` (`bml_matrix_t` *A)

11.8.1 Function Documentation

11.8.1.1 `bml_get()`

```
void* bml_get (
    const bml_matrix_t * A,
    const int i,
    const int j )
```

Return a single matrix element.

Parameters

<i>i</i>	The row index
<i>j</i>	The column index
<i>A</i>	The bml matrix

Returns

The matrix element

11.8.1.2 bml_get_diagonal()

```
void* bml_get_diagonal (
    bml_matrix_t * A )
```

Get the diagonal.

Parameters

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

Returns

The diagonal (an array)

11.8.1.3 bml_get_row()

```
void* bml_get_row (
    bml_matrix_t * A,
    const int i )
```

Get a whole row.

Parameters

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

Returns

An array (needs to be cast into the appropriate type).

11.9 /home/christian/bml/src/C-interface/bml_import.h File Reference

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

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 int M, const void *A, const double threshold, const [bml_distribution_mode_t](#) distrib_mode)

11.10 /home/christian/bml/src/C-interface/bml_init.h File Reference

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

Functions

- void [bml_init](#) (int *argc, char ***argv)
- void [bml_initF](#) (int fcomm)

11.10.1 Function Documentation

11.10.1.1 [bml_init\(\)](#)

```
void bml_init (
    int * argc,
    char *** argv )
```

Initialize.

Parameters

<i>argc</i>	Number of args
<i>argv</i>	Args

11.10.1.2 [bml_initF\(\)](#)

```
void bml_initF (
    int fcomm )
```

Initialize from Fortran.

Parameters

<i>Comm</i>	from Fortran
-------------	--------------

11.11 /home/christian/bml/src/C-interface/bml_introspection.h File Reference

```
#include "bml_types.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)
- [double bml_get_sparsity](#) (const [bml_matrix_t](#) *A, const double threshold)
- [bml_distribution_mode_t bml_get_distribution_mode](#) (const [bml_matrix_t](#) *A)

11.11.1 Function Documentation

11.11.1.1 [bml_get_bandwidth\(\)](#)

```
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.

11.11.1.2 [bml_get_distribution_mode\(\)](#)

```
bml\_distribution\_mode\_t bml_get_distribution_mode (
    const bml\_matrix\_t * A )
```

Return the distribution mode of a matrix.

Parameters

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

Returns

The distibution mode of matrix A.

11.11.1.3 bml_get_M()

```
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.

11.11.1.4 bml_get_N()

```
int bml_get_N (  
    const bml_matrix_t * A )
```

Return the matrix size.

Parameters

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

Returns

The matrix size.

11.11.1.5 bml_get_precision()

```
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.

11.11.1.6 bml_get_row_bandwidth()

```
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*.

11.11.1.7 bml_get_sparsity()

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

Return the sparsity of a matrix.

Parameters

<i>A</i>	The bml matrix.
<i>threshold</i>	The threshold used to compute the sparsity.

Returns

The sparsity of matrix *A*.

11.11.1.8 bml_get_type()

```
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

A	The matrix.
----------	-------------

Returns

The matrix type.

11.12 /home/christian/bml/src/C-interface/bml_logger.h File Reference

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

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,...)

11.12.1 Macro Definition Documentation

11.12.1.1 LOG_DEBUG

```
#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.

11.12.1.2 LOG_ERROR

```
#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.

11.12.1.3 LOG_INFO

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

Convenience macro to write a BML_LOG_INFO level message.

11.12.1.4 LOG_WARN

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

Convenience macro to write a BML_LOG_WARNING level message.

11.12.2 Enumeration Type Documentation

11.12.2.1 bml_log_level_t

```
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.

11.12.3 Function Documentation

11.12.3.1 bml_log()

```
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()).

11.12.3.2 bml_log_location()

```
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()).

11.13 /home/christian/bml/src/C-interface/bml_multiply.h File Reference

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

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)
- void [bml_multiply_adjust_AB](#) (const [bml_matrix_t](#) *A, const [bml_matrix_t](#) *B, [bml_matrix_t](#) *C, const double threshold)

11.13.1 Function Documentation

11.13.1.1 [bml_multiply\(\)](#)

```
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.

$$C \leftarrow \alpha AB + \beta C$$

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

11.13.1.2 [bml_multiply_AB\(\)](#)

```
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

11.13.1.3 bml_multiply_adjust_AB()

```
void bml_multiply_adjust_AB (
    const bml_matrix_t * A,
    const bml_matrix_t * B,
    bml_matrix_t * C,
    const double threshold )
```

Matrix multiply with threshold adjustment.

$$C = A * B$$

Parameters

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

11.13.1.4 bml_multiply_x2()

```
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

11.14 /home/christian/bml/src/C-interface/bml_norm.h File Reference

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

Functions

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

- double [bml_sum_squares2](#) (const [bml_matrix_t](#) *A, const [bml_matrix_t](#) *B, const double alpha, const double beta, const double threshold)
- double [bml_sum_squares_submatrix](#) (const [bml_matrix_t](#) *A, const int core_size)
- double [bml_fnorm](#) (const [bml_matrix_t](#) *A)
- double [bml_fnorm2](#) (const [bml_matrix_t](#) *A, const [bml_matrix_t](#) *B)

11.14.1 Function Documentation

11.14.1.1 [bml_fnorm\(\)](#)

```
double bml_fnorm (
    const bml\_matrix\_t * A )
```

Calculate the Frobenius norm of a matrix.

Parameters

<i>A</i>	Matrix A
----------	----------

Returns

Frobenius norm of Matrix A

11.14.1.2 [bml_fnorm2\(\)](#)

```
double bml_fnorm2 (
    const bml\_matrix\_t * A,
    const bml\_matrix\_t * B )
```

Calculate the Frobenius norm of 2 matrices.

Parameters

<i>A</i>	Matrix A
<i>B</i>	Matrix B

Returns

Frobenius norm of Matrix A

11.14.1.3 bml_sum_squares()

```
double bml_sum_squares (
    const bml_matrix_t * A )
```

Calculate the sum of squares of all the elements of a matrix.

Parameters

<i>A</i>	Matrix A
----------	----------

Returns

sum of squares of all elements in A

11.14.1.4 bml_sum_squares2()

```
double bml_sum_squares2 (
    const bml_matrix_t * A,
    const bml_matrix_t * B,
    const double alpha,
    const double beta,
    const double threshold )
```

Calculate sum of squares of all the elements of A + B

Parameters

<i>A</i>	Matrix
<i>B</i>	Matrix
<i>alpha</i>	Multiplier for matrix A
<i>beta</i>	Multiplier for matrix B
<i>threshold</i>	Threshold

Returns

sum of squares of $\alpha * A + \beta * B$

11.14.1.5 bml_sum_squares_submatrix()

```
double bml_sum_squares_submatrix (
    const bml_matrix_t * A,
    const int core_size )
```

Calculate the sum of squares of all the elements of a matrix.

Parameters

<i>A</i>	Matrix A
<i>core_pos</i>	Core rows in A
<i>core_size</i>	Number of core rows

Returns

sum of squares of all elements in A

11.15 /home/christian/bml/src/C-interface/bml_normalize.h File Reference

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

Functions

- void [bml_normalize](#) ([bml_matrix_t](#) *A, const double mineval, const double maxeval)
- void * [bml_gershgorin](#) (const [bml_matrix_t](#) *A)
- void * [bml_gershgorin_partial](#) (const [bml_matrix_t](#) *A, const int nrows)

11.15.1 Function Documentation**11.15.1.1 bml_gershgorin()**

```
void* bml_gershgorin (
    const bml\_matrix\_t * A )
```

Calculate Gershgorin bounds.

Parameters

<i>A</i>	Matrix to scale returns mineval Calculated min value returns maxeval Calculated max value
----------	---

11.15.1.2 bml_gershgorin_partial()

```
void* bml_gershgorin_partial (
    const bml\_matrix\_t * A,
    const int nrows )
```

Calculate Gershgorin bounds for partial matrix.

Parameters

<i>A</i>	Matrix to scale
<i>nrows</i>	Number of rows used returns mineval Calculated min value returns maxeval Calculated max value

11.15.1.3 `bml_normalize()`

```
void bml_normalize (
    bml_matrix_t * A,
    const double mineval,
    const double maxeval )
```

Normalize matrix given Gershgorin bounds.

Parameters

<i>A</i>	Matrix to scale
<i>mineval</i>	Calculated min value
<i>maxeval</i>	Calculated max value

11.16 `/home/christian/bml/src/C-interface/bml_parallel.h` File Reference

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

Functions

- int `bml_getNRanks` (void)
- int `bml_getMyRank` (void)
- void `bml_initParallelF` (int fcomm)
- void `bml_shutdownParallelF` ()
- int `bml_printRank` (void)
- void `bml_initParallel` (int *argc, char ***argv)
- void `bml_shutdownParallel` (void)
- void `bml_barrierParallel` (void)
- void `bml_sumRealReduce` (double *value)
- void `bml_minRealReduce` (double *value)
- void `bml_maxRealReduce` (double *value)
- void `bml_allGatherVParallel` (`bml_matrix_t` *A)

11.16.1 Function Documentation

11.16.1.1 bml_allGatherVParallel()

```
void bml_allGatherVParallel (
    bml_matrix_t * A )
```

Exchange pieces of matrix across MPI ranks.

Parameters

<i>A</i>	Matrix A
----------	----------

11.16.1.2 bml_getMyRank()

```
int bml_getMyRank (
    void )
```

Get local MPI rank.

11.16.1.3 bml_getNRanks()

```
int bml_getNRanks (
    void )
```

Initialize.

Parameters

<i>argc</i>	Number of args
<i>argv</i>	ArgsGet number of MPI ranks.

11.17 /home/christian/bml/src/C-interface/bml_scale.h File Reference

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

Functions

- [bml_matrix_t * bml_scale_new](#) (const void *scale_factor, const [bml_matrix_t](#) *A)
- void [bml_scale](#) (const void *scale_factor, const [bml_matrix_t](#) *A, [bml_matrix_t](#) *B)
- void [bml_scale_inplace](#) (const void *scale_factor, [bml_matrix_t](#) *A)

11.17.1 Function Documentation

11.17.1.1 bml_scale()

```
void bml_scale (
    const void * 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

11.17.1.2 bml_scale_inplace()

```
void bml_scale_inplace (
    const void * 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

11.17.1.3 bml_scale_new()

```
bml_matrix_t* bml_scale_new (
    const void * 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

11.18 /home/christian/bml/src/C-interface/bml_setters.h File Reference

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

Functions

- void **bml_set_element_new** ([bml_matrix_t](#) *A, const int i, const int j, const void *value)
- void **bml_set_element** ([bml_matrix_t](#) *A, const int i, const int j, const void *value)
- void **bml_set_row** ([bml_matrix_t](#) *A, const int i, const void *row, const double threshold)
- void **bml_set_diagonal** ([bml_matrix_t](#) *A, const void *diagonal, const double threshold)

11.19 /home/christian/bml/src/C-interface/bml_shutdown.h File Reference

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

Functions

- void [bml_shutdown](#) ()
- void [bml_shutdownF](#) ()

11.19.1 Function Documentation

11.19.1.1 [bml_shutdown\(\)](#)

```
void bml_shutdown ( )
```

Shutdown.

11.19.1.2 [bml_shutdownF\(\)](#)

```
void bml_shutdownF ( )
```

Shutdown from Fortran.

11.20 /home/christian/bml/src/C-interface/bml_submatrix.h File Reference

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

Functions

- void [bml_matrix2submatrix_index](#) (const [bml_matrix_t](#) *A, const [bml_matrix_t](#) *B, const int *nodelist, const int nsize, int *core_halo_index, int *vsize, const int double_jump_flag)
- void [bml_matrix2submatrix_index_graph](#) (const [bml_matrix_t](#) *B, const int *nodelist, const int nsize, int *core_halo_index, int *vsize, const int double_jump_flag)
- void [bml_matrix2submatrix](#) (const [bml_matrix_t](#) *A, [bml_matrix_t](#) *B, const int *core_halo_index, const int lsize)
- void [bml_submatrix2matrix](#) (const [bml_matrix_t](#) *A, [bml_matrix_t](#) *B, const int *core_halo_index, const int lsize, const int lsize, const double threshold)
- void [bml_adjacency](#) (const [bml_matrix_t](#) *A, int *xadj, int *adjncy, const int base_flag)
- void [bml_adjacency_group](#) (const [bml_matrix_t](#) *A, const int *hindex, const int nnodes, int *xadj, int *adjncy, const int base_flag)
- [bml_matrix_t](#) * [bml_group_matrix](#) (const [bml_matrix_t](#) *A, const int *hindex, const int ngroups, const double threshold)

11.20.1 Function Documentation

11.20.1.1 [bml_adjacency\(\)](#)

```
void bml_adjacency (
    const bml\_matrix\_t * A,
    int * xadj,
    int * adjncy,
    const int base_flag )
```

Assemble adjacency structures from matrix based on rows.

Parameters

<i>A</i>	Submatrix A
<i>xadj</i>	index to start of each row
<i>adjncy</i>	adjacency vector
<i>base_flag</i>	to return 0- or 1-based

11.20.1.2 [bml_adjacency_group\(\)](#)

```
void bml_adjacency_group (
    const bml\_matrix\_t * A,
    const int * hindex,
    const int nnodes,
    int * xadj,
    int * adjncy,
    const int base_flag )
```

Assemble adjacency structures from matrix based on groups of rows.

Parameters

<i>A</i>	Submatrix A
<i>hindex</i>	Index for each node element
<i>nnodes</i>	Number of groups
<i>xadj</i>	index to start of each row
<i>adjncy</i>	adjacency vector
<i>base_flag</i>	return 0- or 1-based

11.20.1.3 bml_group_matrix()

```
bml_matrix_t* bml_group_matrix (
    const bml_matrix_t * A,
    const int * hindex,
    const int ngroups,
    const double threshold )
```

Assemble matrix based on groups of rows from a matrix.

Parameters

<i>A</i>	Matrix A
<i>hindex</i>	Indeces of nodes
<i>ngroups</i>	Number of groups
<i>threshold</i>	Threshold for graph

11.20.1.4 bml_matrix2submatrix()

```
void bml_matrix2submatrix (
    const bml_matrix_t * A,
    bml_matrix_t * B,
    const int * core_halo_index,
    const int lsize )
```

Extract a submatrix from a matrix given a set of core+halo rows.

Parameters

<i>A</i>	Matrix A
<i>B</i>	Submatrix B
<i>core_halo_index</i>	Set of row indeces for submatrix
<i>lsize</i>	Number of indeces

11.20.1.5 bml_matrix2submatrix_index()

```
void bml_matrix2submatrix_index (
    const bml_matrix_t * A,
    const bml_matrix_t * B,
    const int * nodelist,
    const int nsize,
    int * core_halo_index,
    int * vsize,
    const int double_jump_flag )
```

Determine element indices for submatrix, given a set of nodes/orbitals.

Parameters

<i>A</i>	Hamiltonian matrix A
<i>B</i>	Graph matrix B
<i>nodelist</i>	List of node/orbital indeces
<i>nsize</i>	Size of nodelist
<i>core_halo_index</i>	List of core+halo indeces
<i>vsize</i>	Size of core_halo_index and core_pos
<i>double_jump_flag</i>	Flag to use double jump (0=no, 1=yes)

11.20.1.6 bml_matrix2submatrix_index_graph()

```
void bml_matrix2submatrix_index_graph (
    const bml_matrix_t * B,
    const int * nodelist,
    const int nsize,
    int * core_halo_index,
    int * vsize,
    const int double_jump_flag )
```

Determine element indices for submatrix, given a set of nodes/orbitals.

Parameters

<i>B</i>	Graph matrix B
<i>nodelist</i>	List of node/orbital indeces
<i>nsize</i>	Size of nodelist
<i>core_halo_index</i>	List of core+halo indeces
<i>vsize</i>	Size of core_halo_index and core_pos
<i>double_jump_flag</i>	Flag to use double jump (0=no, 1=yes)

11.20.1.7 bml_submatrix2matrix()

```
void bml_submatrix2matrix (
```

```

const bml_matrix_t * A,
bml_matrix_t * B,
const int * core_halo_index,
const int lsize,
const int llsize,
const double threshold )

```

Assemble submatrix into a full matrix based on core+halo indeces.

Parameters

<i>A</i>	Submatrix A
<i>B</i>	Matrix B
<i>core_halo_index</i>	Set of submatrix row indeces
<i>lsize</i>	Number of indeces
<i>llsize</i>	Number of core positions

11.21 /home/christian/bml/src/C-interface/bml_threshold.h File Reference

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

Functions

- [bml_matrix_t * bml_threshold_new](#) (const [bml_matrix_t](#) *A, const double threshold)
- void [bml_threshold](#) ([bml_matrix_t](#) *A, const double threshold)

11.21.1 Function Documentation

11.21.1.1 bml_threshold()

```

void bml_threshold (
    bml_matrix_t * A,
    const double threshold )

```

Threshold matrix.

Parameters

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

Returns

Thresholded A

11.21.1.2 bml_threshold_new()

```
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

11.22 /home/christian/bml/src/C-interface/bml_trace.h File Reference

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

Functions

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

11.22.1 Function Documentation

11.22.1.1 bml_trace()

```
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

11.23 /home/christian/bml/src/C-interface/bml_transpose.h File Reference

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

Functions

- [bml_matrix_t * bml_transpose_new](#) (const [bml_matrix_t](#) *A)
- void [bml_transpose](#) ([bml_matrix_t](#) *A)

11.23.1 Function Documentation

11.23.1.1 [bml_transpose\(\)](#)

```
void bml_transpose (  
    bml\_matrix\_t * A )
```

Transpose matrix.

Parameters

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

Returns

Transposed A

11.23.1.2 [bml_transpose_new\(\)](#)

```
bml\_matrix\_t* bml_transpose_new (  
    const bml\_matrix\_t * A )
```

Transpose matrix.

Parameters

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

Returns

Transposed A

11.24 /home/christian/bml/src/C-interface/bml_transpose_triangle.h File Reference

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

Functions

- void [bml_transpose_triangle](#) ([bml_matrix_t](#) *A, char triangle)

11.24.1 Function Documentation**11.24.1.1 bml_transpose_triangle()**

```
void bml_transpose_triangle (
    bml\_matrix\_t * A,
    char triangle )
```

Transposes a triangle of a matrix in place.

Parameters

<i>A</i>	The matrix for which the triangle should be transposed
<i>triangle</i>	Which triangle to transpose ('u': upper, 'l': lower)

11.25 /home/christian/bml/src/C-interface/bml_types.h File Reference**Classes**

- struct [bml_domain_t](#)

Typedefs

- typedef void [bml_vector_t](#)
- typedef void [bml_matrix_t](#)
- typedef struct [bml_domain_t](#) **bml_domain_t**

Enumerations

- enum `bml_matrix_type_t` {
 `type_uninitialized`, `dense`, `ellpack`, `ellsort`,
 `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` }
- enum `bml_distribution_mode_t` { `sequential`, `distributed`, `graph_distributed` }

11.25.1 Typedef Documentation

11.25.1.1 `bml_matrix_t`

```
typedef void bml_matrix_t
```

The matrix type.

11.25.1.2 `bml_vector_t`

```
typedef void bml_vector_t
```

The vector type.

11.25.2 Enumeration Type Documentation

11.25.2.1 `bml_dense_order_t`

```
enum bml_dense_order_t
```

The supported dense matrix elements orderings.

Enumerator

<code>dense_row_major</code>	row-major order.
<code>dense_column_major</code>	column-major order.

11.25.2.2 bml_distribution_mode_t

enum `bml_distribution_mode_t`

The supported distribution modes.

Enumerator

<code>sequential</code>	Each rank works on the full matrix.
<code>distributed</code>	Each rank works on its part of the matrix.
<code>graph_distributed</code>	Each rank works on its set of graph partitions.

11.25.2.3 bml_matrix_precision_t

enum `bml_matrix_precision_t`

The supported real precisions.

Enumerator

<code>precision_uninitialized</code>	The matrix is not initialized.
<code>single_real</code>	Matrix data is stored in single precision (float).
<code>double_real</code>	Matrix data is stored in double precision (double).
<code>single_complex</code>	Matrix data is stored in single-complex precision (float).
<code>double_complex</code>	Matrix data is stored in double-complex precision (double).

11.25.2.4 bml_matrix_type_t

enum `bml_matrix_type_t`

The supported matrix types.

Enumerator

<code>type_uninitialized</code>	The matrix is not initialized.
<code>dense</code>	Dense matrix.
<code>ellpack</code>	ELLPACK matrix.
<code>ellsort</code>	ELLSORT matrix.
<code>csr</code>	CSR matrix.

11.26 /home/christian/bml/src/C-interface/bml_types_private.h File Reference

11.27 /home/christian/bml/src/C-interface/bml_utilities.h File Reference

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

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)
- void [bml_read_bml_matrix](#) (const [bml_matrix_t](#) *A, const char *filename)
- void [bml_write_bml_matrix](#) (const [bml_matrix_t](#) *A, const char *filename)

11.27.1 Function Documentation

11.27.1.1 bml_print_bml_matrix()

```
void bml_print_bml_matrix (
    const bml\_matrix\_t * A,
    const int i_l,
    const int i_u,
    const int j_l,
    const int j_u )
```

Print a dense matrix.

Parameters

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

11.27.1.2 bml_print_bml_vector()

```
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	
i_l	The upper row index.
i_u	

11.27.1.3 bml_print_dense_matrix()

```
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.

11.27.1.4 bml_print_dense_vector()

```
void bml_print_dense_vector (
    const int N,
```

```

    bml_matrix_precision_t matrix_precision,
    const void * v,
    const int i_l,
    const int i_u )

```

Print a dense vector.

Parameters

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

11.27.1.5 bml_read_bml_matrix()

```

void bml_read_bml_matrix (
    const bml_matrix_t * A,
    const char * filename )

```

Read a bml matrix from a Matrix Market file.

Parameters

<i>A</i>	The matrix
<i>filename</i>	The file containing matrix

11.27.1.6 bml_write_bml_matrix()

```

void bml_write_bml_matrix (
    const bml_matrix_t * A,
    const char * filename )

```

Write a bml matrix to a Matrix Market file.

Parameters

<i>A</i>	The matrix
<i>filename</i>	The file containing matrix

Index

- /home/christian/bml/src/C-interface/bml.h, [35](#)
- /home/christian/bml/src/C-interface/bml_add.h, [35](#)
- /home/christian/bml/src/C-interface/bml_adjungate_↔
triangle.h, [36](#)
- /home/christian/bml/src/C-interface/bml_allocate.h, [36](#)
- /home/christian/bml/src/C-interface/bml_convert.h, [37](#)
- /home/christian/bml/src/C-interface/bml_copy.h, [38](#)
- /home/christian/bml/src/C-interface/bml_export.h, [40](#)
- /home/christian/bml/src/C-interface/bml_getters.h, [40](#)
- /home/christian/bml/src/C-interface/bml_import.h, [42](#)
- /home/christian/bml/src/C-interface/bml_init.h, [42](#)
- /home/christian/bml/src/C-interface/bml_introspection_↔
h, [43](#)
- /home/christian/bml/src/C-interface/bml_logger.h, [46](#)
- /home/christian/bml/src/C-interface/bml_multiply.h, [48](#)
- /home/christian/bml/src/C-interface/bml_norm.h, [50](#)
- /home/christian/bml/src/C-interface/bml_normalize.h, [53](#)
- /home/christian/bml/src/C-interface/bml_parallel.h, [54](#)
- /home/christian/bml/src/C-interface/bml_scale.h, [55](#)
- /home/christian/bml/src/C-interface/bml_setters.h, [57](#)
- /home/christian/bml/src/C-interface/bml_shutdown.h, [57](#)
- /home/christian/bml/src/C-interface/bml_submatrix.h, [57](#)
- /home/christian/bml/src/C-interface/bml_threshold.h, [61](#)
- /home/christian/bml/src/C-interface/bml_trace.h, [62](#)
- /home/christian/bml/src/C-interface/bml_transpose.h, [63](#)
- /home/christian/bml/src/C-interface/bml_transpose_↔
triangle.h, [64](#)
- /home/christian/bml/src/C-interface/bml_types.h, [64](#)
- /home/christian/bml/src/C-interface/bml_types_↔
private.h, [66](#)
- /home/christian/bml/src/C-interface/bml_utilities.h, [67](#)
- Add Functions (C interface), [24](#)
 - bml_add, [24](#)
 - bml_add_identity, [24](#)
 - bml_add_norm, [25](#)
 - bml_scale_add_identity, [25](#)
- Add Functions (Fortran interface), [29](#)
- Allocation and Deallocation Functions (C interface), [17](#)
 - bml_allocate_memory, [17](#)
 - bml_allocated, [18](#)
 - bml_banded_matrix, [18](#)
 - bml_clear, [19](#)
 - bml_deallocate, [19](#)
 - bml_deallocate_domain, [19](#)
 - bml_default_domain, [19](#)
 - bml_free_memory, [20](#)
 - bml_free_ptr, [20](#)
 - bml_identity_matrix, [20](#)
 - bml_noinit_allocate_memory, [21](#)
 - bml_noinit_matrix, [21](#)
 - bml_random_matrix, [22](#)
 - bml_update_domain, [22](#)
 - bml_zero_matrix, [23](#)
- Allocation and Deallocation Functions (Fortran inter-
face), [28](#)
- bml_add
 - Add Functions (C interface), [24](#)
- bml_add_identity
 - Add Functions (C interface), [24](#)
- bml_add_norm
 - Add Functions (C interface), [25](#)
- bml_adjacency
 - bml_submatrix.h, [58](#)
- bml_adjacency_group
 - bml_submatrix.h, [58](#)
- bml_adjungate_triangle
 - bml_adjungate_triangle.h, [36](#)
- bml_adjungate_triangle.h
 - bml_adjungate_triangle, [36](#)
- bml_allGatherVParallel
 - bml_parallel.h, [54](#)
- bml_allocate_memory
 - Allocation and Deallocation Functions (C interface),
[17](#)
- bml_allocated
 - Allocation and Deallocation Functions (C interface),
[18](#)
- bml_banded_matrix
 - Allocation and Deallocation Functions (C interface),
[18](#)
- bml_clear
 - Allocation and Deallocation Functions (C interface),
[19](#)
- bml_convert
 - bml_convert.h, [37](#)
- bml_convert.h
 - bml_convert, [37](#)
- bml_copy
 - bml_copy.h, [38](#)
- bml_copy.h
 - bml_copy, [38](#)
 - bml_copy_domain, [38](#)
 - bml_copy_new, [39](#)
 - bml_reorder, [39](#)
 - bml_restore_domain, [39](#)
 - bml_save_domain, [40](#)
- bml_copy_domain
 - bml_copy.h, [38](#)

- bml_copy_new
 - bml_copy.h, [39](#)
- bml_deallocate
 - Allocation and Deallocation Functions (C interface), [19](#)
- bml_deallocate_domain
 - Allocation and Deallocation Functions (C interface), [19](#)
- bml_default_domain
 - Allocation and Deallocation Functions (C interface), [19](#)
- bml_dense_order_t
 - bml_types.h, [65](#)
- bml_distribution_mode_t
 - bml_types.h, [65](#)
- bml_domain_t, [31](#)
 - globalRowExtent, [31](#)
 - globalRowMax, [31](#)
 - globalRowMin, [32](#)
 - localDispl, [32](#)
 - localElements, [32](#)
 - localRowExtent, [32](#)
 - localRowMax, [32](#)
 - localRowMin, [32](#)
 - maxLocalExtent, [32](#)
 - minLocalExtent, [32](#)
 - totalCols, [33](#)
 - totalProcs, [33](#)
 - totalRows, [33](#)
- bml_export_to_dense
 - Converting between Matrix Formats (C interface), [26](#)
- bml_fnorm
 - bml_norm.h, [51](#)
- bml_fnorm2
 - bml_norm.h, [51](#)
- bml_free_memory
 - Allocation and Deallocation Functions (C interface), [20](#)
- bml_free_ptr
 - Allocation and Deallocation Functions (C interface), [20](#)
- bml_gershgorin
 - bml_normalize.h, [53](#)
- bml_gershgorin_partial
 - bml_normalize.h, [53](#)
- bml_get
 - bml_getters.h, [40](#)
- bml_get_bandwidth
 - bml_introspection.h, [43](#)
- bml_get_diagonal
 - bml_getters.h, [41](#)
- bml_get_distribution_mode
 - bml_introspection.h, [43](#)
- bml_get_M
 - bml_introspection.h, [44](#)
- bml_get_N
 - bml_introspection.h, [44](#)
- bml_get_precision
 - bml_introspection.h, [44](#)
- bml_get_row
 - bml_getters.h, [41](#)
- bml_get_row_bandwidth
 - bml_introspection.h, [45](#)
- bml_get_sparsity
 - bml_introspection.h, [45](#)
- bml_get_type
 - bml_introspection.h, [45](#)
- bml_getMyRank
 - bml_parallel.h, [55](#)
- bml_getNRanks
 - bml_parallel.h, [55](#)
- bml_getters.h
 - bml_get, [40](#)
 - bml_get_diagonal, [41](#)
 - bml_get_row, [41](#)
- bml_group_matrix
 - bml_submatrix.h, [59](#)
- bml_identity_matrix
 - Allocation and Deallocation Functions (C interface), [20](#)
- bml_import_from_dense
 - Converting between Matrix Formats (C interface), [26](#)
- bml_init
 - bml_init.h, [42](#)
- bml_init.h
 - bml_init, [42](#)
 - bml_initF, [42](#)
- bml_initF
 - bml_init.h, [42](#)
- bml_introspection.h
 - bml_get_bandwidth, [43](#)
 - bml_get_distribution_mode, [43](#)
 - bml_get_M, [44](#)
 - bml_get_N, [44](#)
 - bml_get_precision, [44](#)
 - bml_get_row_bandwidth, [45](#)
 - bml_get_sparsity, [45](#)
 - bml_get_type, [45](#)
- bml_log
 - bml_logger.h, [48](#)
- bml_log_level_t
 - bml_logger.h, [47](#)
- bml_log_location
 - bml_logger.h, [48](#)
- bml_logger.h
 - bml_log, [48](#)
 - bml_log_level_t, [47](#)
 - bml_log_location, [48](#)
 - LOG_DEBUG, [46](#)
 - LOG_ERROR, [47](#)
 - LOG_INFO, [47](#)
 - LOG_WARN, [47](#)
- bml_matrix2submatrix
 - bml_submatrix.h, [59](#)

- bml_matrix2submatrix_index
 - bml_submatrix.h, 59
- bml_matrix2submatrix_index_graph
 - bml_submatrix.h, 60
- bml_matrix_precision_t
 - bml_types.h, 66
- bml_matrix_t
 - bml_types.h, 65
- bml_matrix_type_t
 - bml_types.h, 66
- bml_multiply
 - bml_multiply.h, 49
- bml_multiply.h
 - bml_multiply, 49
 - bml_multiply_AB, 49
 - bml_multiply_adjust_AB, 50
 - bml_multiply_x2, 50
- bml_multiply_AB
 - bml_multiply.h, 49
- bml_multiply_adjust_AB
 - bml_multiply.h, 50
- bml_multiply_x2
 - bml_multiply.h, 50
- bml_noinit_allocate_memory
 - Allocation and Deallocation Functions (C interface), 21
- bml_noinit_matrix
 - Allocation and Deallocation Functions (C interface), 21
- bml_norm.h
 - bml_fnorm, 51
 - bml_fnorm2, 51
 - bml_sum_squares, 51
 - bml_sum_squares2, 52
 - bml_sum_squares_submatrix, 52
- bml_normalize
 - bml_normalize.h, 54
- bml_normalize.h
 - bml_gershgorin, 53
 - bml_gershgorin_partial, 53
 - bml_normalize, 54
- bml_parallel.h
 - bml_allGatherVParallel, 54
 - bml_getMyRank, 55
 - bml_getNRanks, 55
- bml_print_bml_matrix
 - bml_utilities.h, 67
- bml_print_bml_vector
 - bml_utilities.h, 67
- bml_print_dense_matrix
 - bml_utilities.h, 68
- bml_print_dense_vector
 - bml_utilities.h, 68
- bml_random_matrix
 - Allocation and Deallocation Functions (C interface), 22
- bml_read_bml_matrix
 - bml_utilities.h, 69
- bml_reorder
 - bml_copy.h, 39
- bml_restore_domain
 - bml_copy.h, 39
- bml_save_domain
 - bml_copy.h, 40
- bml_scale
 - bml_scale.h, 55
- bml_scale.h
 - bml_scale, 55
 - bml_scale_inplace, 56
 - bml_scale_new, 56
- bml_scale_add_identity
 - Add Functions (C interface), 25
- bml_scale_inplace
 - bml_scale.h, 56
- bml_scale_new
 - bml_scale.h, 56
- bml_shutdown
 - bml_shutdown.h, 57
- bml_shutdown.h
 - bml_shutdown, 57
 - bml_shutdownF, 57
- bml_shutdownF
 - bml_shutdown.h, 57
- bml_submatrix.h
 - bml_adjacency, 58
 - bml_adjacency_group, 58
 - bml_group_matrix, 59
 - bml_matrix2submatrix, 59
 - bml_matrix2submatrix_index, 59
 - bml_matrix2submatrix_index_graph, 60
 - bml_submatrix2matrix, 60
- bml_submatrix2matrix
 - bml_submatrix.h, 60
- bml_sum_squares
 - bml_norm.h, 51
- bml_sum_squares2
 - bml_norm.h, 52
- bml_sum_squares_submatrix
 - bml_norm.h, 52
- bml_threshold
 - bml_threshold.h, 61
- bml_threshold.h
 - bml_threshold, 61
 - bml_threshold_new, 62
- bml_threshold_new
 - bml_threshold.h, 62
- bml_trace
 - bml_trace.h, 62
- bml_trace.h
 - bml_trace, 62
- bml_transpose
 - bml_transpose.h, 63
- bml_transpose.h
 - bml_transpose, 63
 - bml_transpose_new, 63
- bml_transpose_new

- bml_transpose.h, [63](#)
- bml_transpose_triangle
 - bml_transpose_triangle.h, [64](#)
- bml_transpose_triangle.h
 - bml_transpose_triangle, [64](#)
- bml_types.h
 - bml_dense_order_t, [65](#)
 - bml_distribution_mode_t, [65](#)
 - bml_matrix_precision_t, [66](#)
 - bml_matrix_t, [65](#)
 - bml_matrix_type_t, [66](#)
 - bml_vector_t, [65](#)
- bml_update_domain
 - Allocation and Deallocation Functions (C interface), [22](#)
- bml_utilities.h
 - bml_print_bml_matrix, [67](#)
 - bml_print_bml_vector, [67](#)
 - bml_print_dense_matrix, [68](#)
 - bml_print_dense_vector, [68](#)
 - bml_read_bml_matrix, [69](#)
 - bml_write_bml_matrix, [69](#)
- bml_vector_t
 - bml_types.h, [65](#)
- bml_write_bml_matrix
 - bml_utilities.h, [69](#)
- bml_zero_matrix
 - Allocation and Deallocation Functions (C interface), [23](#)
- Converting between Matrix Formats (C interface), [26](#)
 - bml_export_to_dense, [26](#)
 - bml_import_from_dense, [26](#)
- Converting between Matrix Formats (Fortran interface), [30](#)
- globalRowExtent
 - bml_domain_t, [31](#)
- globalRowMax
 - bml_domain_t, [31](#)
- globalRowMin
 - bml_domain_t, [32](#)
- LOG_DEBUG
 - bml_logger.h, [46](#)
- LOG_ERROR
 - bml_logger.h, [47](#)
- LOG_INFO
 - bml_logger.h, [47](#)
- LOG_WARN
 - bml_logger.h, [47](#)
- localDispl
 - bml_domain_t, [32](#)
- localElements
 - bml_domain_t, [32](#)
- localRowExtent
 - bml_domain_t, [32](#)
- localRowMax
 - bml_domain_t, [32](#)
- localRowMin
 - bml_domain_t, [32](#)
- maxLocalExtent
 - bml_domain_t, [32](#)
- minLocalExtent
 - bml_domain_t, [32](#)
- totalCols
 - bml_domain_t, [33](#)
- totalProcs
 - bml_domain_t, [33](#)
- totalRows
 - bml_domain_t, [33](#)