bml

0.1.0

Generated by Doxygen 1.8.9.1

Wed Sep 16 2015 14:41:28

Contents

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 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)
```

1.1.1 Supported Matrix Types

Support types:

- bml_matrix_t
- Colinear
- Noncolinear
- Blocked Bloch Matrix

1.1.2 Supported Precisions

The bml supports the following precisions:

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

1.1.3 Supported 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
  - $\mathrm{Tr}[A]$: bml_trace::bml_trace
  - $\mathrm{Tr}[AB]$: 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

1.2 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_precision, 100);
bml_deallocate(&A);
```

Author

Jamaludin Mohd-Yusof jamal@lanl.gov
Nicolas Bock nbock@lanl.gov
Susan M. Mniszewski smm@lanl.gov

Copyright

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Namespace Index

3.1 Namespace List

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

Chapter 4

Class Index

4.1 Class List

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

Chapter 5

File Index

5.1 File List

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

Chapter 6

Module Documentation

6.1 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)

6.1.1 Detailed Description

6.1.2 Function Documentation

6.1.2.1 void* bml_allocate_memory (const size_t *size*)

Allocate and zero a chunk of memory.

Parameters

size	The size of the memory.

Returns

A pointer to the allocated chunk.

6.1.2.2 void bml_deallocate (bml_matrix_t ** *A*)

Deallocate a matrix.

Parameters

A	The matrix.

Here is the call graph for this function:

6.1.2.3 void bml_free_memory (void * *ptr*)

Deallocate a chunk of memory.

Parameters

ptr	A pointer to the previously allocated chunk.

6.1.2.4 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

matrix_type	The matrix type.
matrix_precision	The precision of the matrix. The default is double precision.
N	The matrix size.
M	The number of non-zeroes per row.

Returns

The matrix.

6.1.2.5 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

matrix_type	The matrix type.
matrix_precision	The precision of the matrix. The default is double precision.
N	The matrix size.

M	The number of non-zeroes per row.

Returns

The matrix.

6.1.2.6 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

matrix_type	The matrix type.
matrix_precision	The precision of the matrix. The default is double precision.
N	The matrix size.
M	The number of non-zeroes per row.

Returns

The matrix.

6.2 Converting between Matrix Formats (C interface)

Functions

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

6.2.1 Detailed Description

6.2.2 Function Documentation

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

Convert a dense matrix into a bml matrix.

Parameters

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

Returns

The bml matrix

6.2.2.2 void* bml_convert_to_dense (const bml_matrix_t * *A*)

Convert a bml matrix into a dense matrix.

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

```
float *A_dense = bml_convert_to_dense(A_bml);
```

The matrix size can be queried with

```
int N = bml_get_size(A_bml);
```

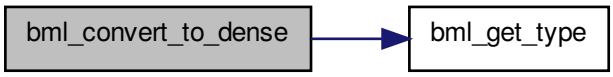
Parameters

A	The bml matrix

Returns

The dense matrix

Here is the call graph for this function:



6.3 Allocation and Deallocation Functions (Fortran interface)

Functions

- subroutine, public bml_allocate::bml_deallocate (a)

  Deallocate a matrix.
- subroutine, public bml_allocate::bml_zero_matrix (matrix_type, matrix_precision, n, a, m)

  Create the zero matrix.
- subroutine, public bml_allocate::bml_random_matrix (matrix_type, matrix_precision, n, a, m)

  Create a random matrix.
- subroutine, public bml_allocate::bml_identity_matrix (matrix_type, matrix_precision, n, a, m)

  Create the identity matrix.

6.3.1 Detailed Description

6.3.2 Function Documentation

6.3.2.1 subroutine, public bml_allocate::bml_deallocate (type(bml_matrix_t) *a*)

Deallocate a matrix.

Parameters

a	The matrix.

6.3.2.2 subroutine, public bml_allocate::bml_identity_matrix (character(len=∗), intent(in) *matrix_type,* character(len=∗), intent(in) *matrix_precision,* integer, intent(in) *n,* type(bml_matrix_t), intent(inout) *a,* integer, intent(in) *m*)

Create the identity matrix.

Parameters

matrix_type	The matrix type.
matrix_precision	The precision of the matrix.
n	The matrix size.
a	The matrix.
m	The extra arg.

6.3.2.3 subroutine, public bml_allocate::bml_random_matrix (character(len=∗), intent(in) *matrix_type,* character(len=∗), intent(in) *matrix_precision,* integer, intent(in) *n,* type(bml_matrix_t), intent(inout) *a,* integer, intent(in) *m*)

Create a random matrix.

Parameters

matrix_type	The matrix type.
matrix_precision	The precision of the matrix.
n	The matrix size.
a	The matrix.
m	The extra arg.

6.3.2.4 subroutine, public bml_allocate::bml_zero_matrix (character(len=*), intent(in) *matrix_type,* character(len=*), intent(in) *matrix_precision,* integer, intent(in) *n,* type(**bml_matrix_t**), intent(inout) *a,* integer, intent(in) *m*)

Create the zero matrix.

Parameters

matrix_type	The matrix type.
matrix_precision	The precision of the matrix.
n	The matrix size.
a	The matrix.
m	The extra arg.

6.4 Converting between Matrix Formats (Fortran interface)

Functions

- subroutine bml_convert::bml_convert_from_dense_double (matrix_type, matrix_precision, a_dense, a, threshold, m)

  Convert a dense matrix into a bml matrix.
- subroutine bml_convert::bml_convert_to_dense_single (a, a_dense)

  Convert a matrix into a dense matrix.
- subroutine bml_convert::bml_convert_to_dense_double (a, a_dense)

  Convert a matrix into a dense matrix.

6.4.1 Detailed Description

6.4.2 Function Documentation

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

Convert a dense matrix into a bml matrix.

Parameters

matrix_type	The matrix type
matrix_precision	The matrix precision
a_dense	The dense matrix
a	The bml matrix
threshold	The matrix element magnited threshold
m	the extra arg

6.4.2.2 subroutine bml_convert::bml_convert_to_dense_double (type(**bml_matrix_t**), intent(in) *a,* double precision, dimension(:, :), intent(out), pointer *a_dense*)

Convert a matrix into a dense matrix.

Parameters

a	The bml matrix
a_dense	The dense matrix

6.4.2.3 subroutine bml_convert::bml_convert_to_dense_single (type(**bml_matrix_t**), intent(in) *a,* real, dimension(:, :), intent(out), pointer *a_dense*)

Convert a matrix into a dense matrix.

Parameters

a	The bml matrix
a_dense	The dense matrix

Chapter 7

Namespace Documentation

7.1 bml Module Reference

Main matrix library module.

7.1.1 Detailed Description

Main matrix library module.

Use this modules in order to use the library.

7.2 bml_allocate Module Reference

Matrix allocation functions.

Functions/Subroutines

- subroutine, public bml_deallocate (a)

  Deallocate a matrix.
- subroutine, public bml_zero_matrix (matrix_type, matrix_precision, n, a, m)

  Create the zero matrix.
- subroutine, public bml_random_matrix (matrix_type, matrix_precision, n, a, m)

  Create a random matrix.
- subroutine, public bml_identity_matrix (matrix_type, matrix_precision, n, a, m)

  Create the identity matrix.

7.2.1 Detailed Description

Matrix allocation functions.

7.3 bml_interface Module Reference

Interface module.

Functions/Subroutines

- integer function get_enum_id (type_string)

  Convert the matrix type and precisions strings into enum values.

Variables

- integer, parameter bml_matrix_type_uninitialized_enum_id = 0

  The enum values of the C API. Keep this synchronized with the enum in bml_types.h.
- integer, parameter **bml_matrix_type_dense_enum_id** = 1
- integer, parameter **bml_matrix_precision_single_enum_id** = 0
- integer, parameter **bml_matrix_precision_double_enum_id** = 1

7.3.1 Detailed Description

Interface module.

7.3.2 Function/Subroutine Documentation

7.3.2.1 integer function bml_interface::get_enum_id (character(len=*), intent(in) *type_string*)

Convert the matrix type and precisions strings into enum values.

Parameters

type_string	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.

7.4 bml_introspection Module Reference

Introspection procedures.

Data Types

- interface bml_get_size_C

  Return the matrix size.

Functions/Subroutines

- integer function bml_get_size (a)

  Return the matrix size.

7.4.1 Detailed Description

Introspection procedures.

7.4.2 Function/Subroutine Documentation

7.4.2.1 integer function bml_introspection::bml_get_size (type(bml_matrix_t), intent(in) *a*)

Return the matrix size.

Parameters

a	The matrix.

Returns

The matrix size.

7.5 bml_types Module Reference

The basic bml types.

Data Types

- type bml_matrix_t

  The bml matrix type.

Variables

- character(len=∗), parameter bml_matrix_dense = "dense"

  The bml-dense matrix type identifier.
- character(len=∗), parameter bml_matrix_ellpack = "ellpack"

  The bml-ellpack matrix type identifier.
- character(len=∗), parameter bml_precision_single = "single-precision"

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

  The double-precision identifier.

7.5.1 Detailed Description

The basic bml types.

7.6 bml_utilities Module Reference

Utility matrix functions.

Functions/Subroutines

- subroutine bml_print_matrix_double (tag, a, i_l, i_u, j_l, j_u)

  Print a dense matrix.

7.6.1 Detailed Description

Utility matrix functions.

7.6.2 Function/Subroutine Documentation

7.6.2.1 subroutine bml_utilities::bml_print_matrix_double (character(len=*), intent(in) *tag,* double precision, dimension(:, :), intent(in), target *a,* integer, intent(in) *i_l,* integer, intent(in) *i_u,* integer, intent(in) *j_l,* integer, intent(in) *j_u*)

Print a dense matrix.

Parameters

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

Chapter 8

Class Documentation

8.1 bml_introspection::bml_get_size_C Interface Reference

Return the matrix size.

Public Member Functions

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

8.1.1 Detailed Description

Return the matrix size.

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

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

8.2 bml_types::bml_matrix_t Type Reference

The bml matrix type.

Public Attributes

- type(c_ptr) ptr = C_NULL_PTR

  The C pointer to the matrix.

8.2.1 Detailed Description

The bml matrix type.

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

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

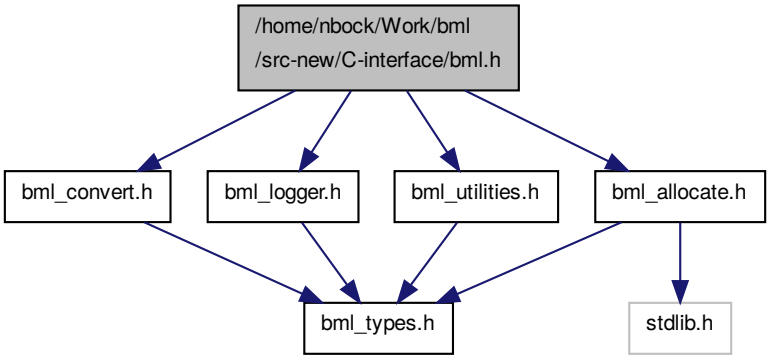
Chapter 9

File Documentation

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

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

Include dependency graph for bml.h:



9.1.1 Detailed Description

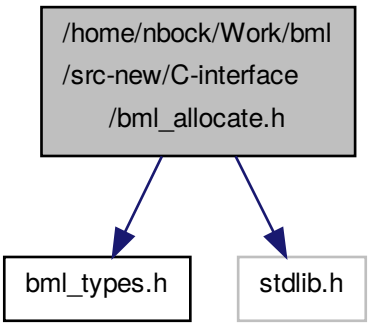
Copyright

Los Alamos National Laboratory 2015

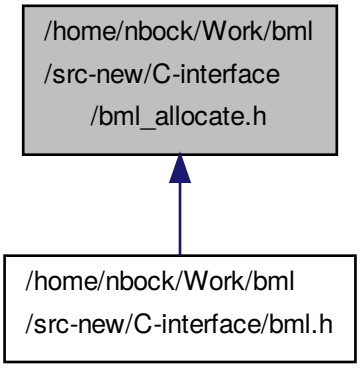
9.2 /home/nbock/Work/bml/src-new/C-interface/bml_allocate.h File Reference

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

Include dependency graph for bml_allocate.h:



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



Functions

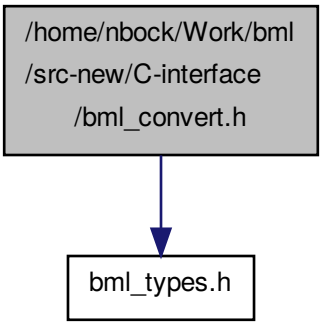
- void * bml_allocate_memory (const size_t s)
- void bml_free_memory (void *ptr)
- void bml_deallocate (bml_matrix_t **A)

- bml_matrix_t * bml_zero_matrix (const bml_matrix_type_t matrix_type, const bml_matrix_precision_t matrix_precision, const int N, const int M)
- bml_matrix_t * bml_random_matrix (const bml_matrix_type_t matrix_type, const bml_matrix_precision_t matrix_precision, const int N, const int M)
- bml_matrix_t * bml_identity_matrix (const bml_matrix_type_t matrix_type, const bml_matrix_precision_t matrix_precision, const int N, const int M)

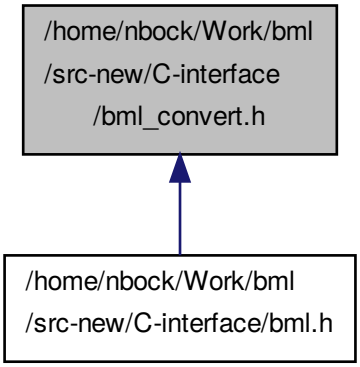
9.3 /home/nbock/Work/bml/src-new/C-interface/bml_convert.h File Reference

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

Include dependency graph for bml_convert.h:



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



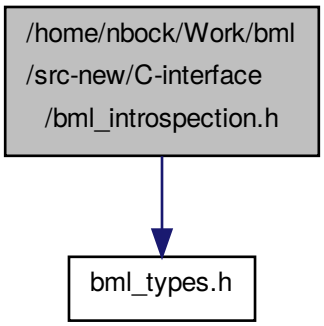
Functions

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

9.4 /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)

9.4.1 Function Documentation

9.4.1.1 int bml_get_size (const bml_matrix_t * A)

Return the matrix size.

Parameters

A	The matrix.

Returns

The matrix size.

Here is the call graph for this function:

9.4.1.2 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 "unitialized" is returned.

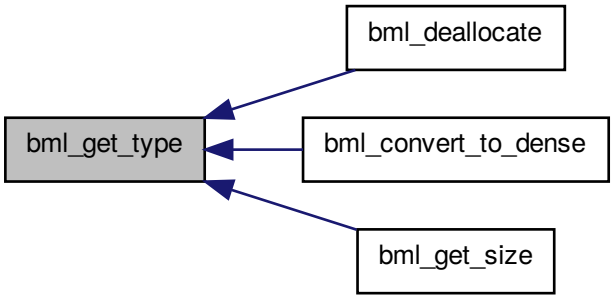
Parameters

A	The matrix.

Returns

The matrix type.

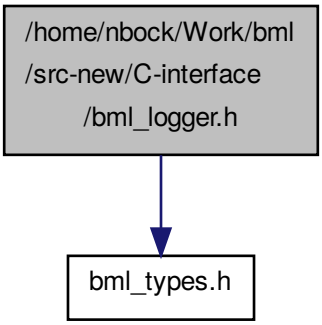
Here is the caller graph for this function:



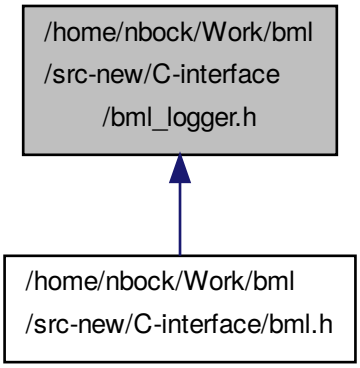
9.5 /home/nbock/Work/bml/src-new/C-interface/bml_logger.h File Reference

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

Include dependency graph for bml_logger.h:



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



Macros

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

Enumerations

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

Functions

- void bml_log (const bml_log_level_t log_level, const char *format,...)
- void bml_log_location (const bml_log_level_t log_level, const char *filename, const int linenumber, const char *format,...)

9.5.1 Macro Definition Documentation

9.5.1.1 #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.

9.5.1.2 #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.

9.5.1.3 #define LOG_INFO(*format, ...*) bml_log(BML_LOG_INFO, format, ##__VA_ARGS__)

Convenience macro to write a BML_LOG_INFO level message.

9.5.1.4 #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.

9.5.2 Enumeration Type Documentation

9.5.2.1 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.

9.5.3 Function Documentation

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

Log a message.

Parameters

log_level	The log level.

format	The format (as in printf()).

9.5.3.2 void bml_log_location (const bml_log_level_t *log_level*, const char * *filename*, const int *linenumber*, const char * *format*, ...)

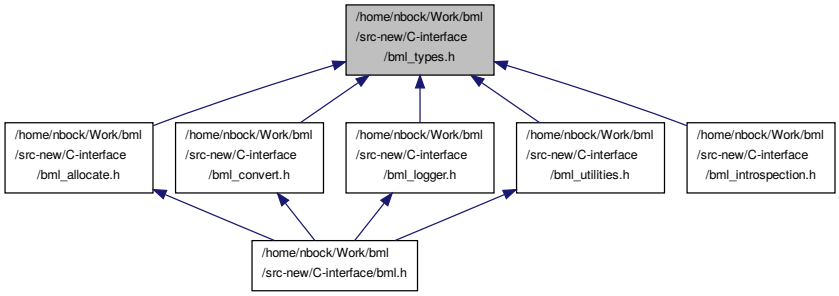
Log a message with location, i.e. filename and linenumber..

Parameters

log_level	The log level.
filename	The filename to log.
linenumber	The linenumber.
format	The format (as in printf()).

9.6 /home/nbock/Work/bml/src-new/C-interface/bml_types.h File Reference

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



Typedefs

- typedef void bml_matrix_t

Enumerations

- enum bml_matrix_type_t { uninitialized, dense, ellpack, csr }
- enum bml_matrix_precision_t { single_precision, double_precision }

9.6.1 Typedef Documentation

9.6.1.1 typedef void bml_matrix_t

The matrix type.

9.6.2 Enumeration Type Documentation

9.6.2.1 enum bml_matrix_precision_t

The supported real precisions.

Enumerator

single_precision Matrix data is stored in single precision (float).

double_precision Matrix data is stored in double precision (double).

9.6.2.2 enum bml_matrix_type_t

The supported matrix types.

Enumerator

uninitialized The matrix is not initialized.

dense Dense matrix.

ellpack ELLPACK matrix.

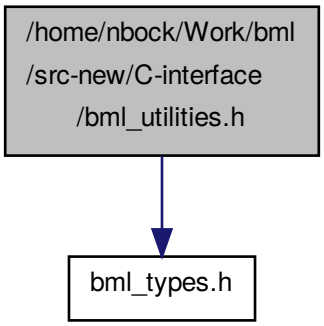
csr CSR matrix.

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

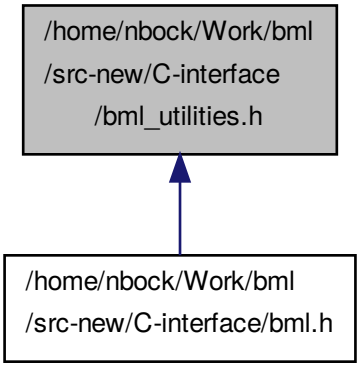
9.8 /home/nbock/Work/bml/src-new/C-interface/bml_utilities.h File Reference

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

Include dependency graph for bml_utilities.h:



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



Functions

- void bml_print_matrix (const int N, bml_matrix_precision_t matrix_precision, const void ∗A, const int i_l, const int i_u, const int j_l, const int j_u)

9.8.1 Function Documentation

9.8.1.1 void bml_print_matrix (const int *N*, bml_matrix_precision_t *matrix_precision*, const void ∗ *A*, const int *i_l*, const int *i_u*, const int *j_l*, const int *j_u*)

Print a dense matrix.

Parameters

N	The number of rows/columns.
matrix_precision	The real precision.
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.

Index