MathematicalFunctions

5.0

Generated by Doxygen 1.8.14

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Module Documentation

5.1 Models

Modules

- Utils

5.1.1 Detailed Description

5.2 Utils

Modules

- Math

5.2.1 Detailed Description

5.3 Math

Files

- file gauss_quadrature.hh

  Gauss Quadrature implementation.
- file math_messages.hh

  Define the class MathMessages.
- file matrix3x3.hh

  Matrix math inline functions.
- file matrix3x3_inline.hh

  Matrix math inline functions.
- file numerical.hh

  Miscellaneous math inline functions.
- file numerical_inline.hh

  Vector math inline functions.
- file vector3.hh

  Vector math inline functions.
- file vector3_inline.hh

  Vector math inline functions.
- file dm_invert.cc

  Define Matrix3x3::invert.
- file dm_invert_symm.cc

  Define Matrix3x3::invert_symmetric.
- file gauss_quadrature.cc

  Define Gauss Quadrature functionality.
- file math_messages.cc

  Implement the class MathMessages.

Namespaces

- jeod

  Namespace jeod.

Macros

- #define GSL_SQRT_DBL_MIN 1.4916681462400413e-154
- #define MAKE_MATH_MESSAGE_CODE(id) JEOD_MAKE_MESSAGE_CODE(MathMessages, "utils/math/", id)

5.3.1 Detailed Description

5.3.2 Macro Definition Documentation

5.3.2.1 GSL_SQRT_DBL_MIN

```
#define GSL_SQRT_DBL_MIN 1.4916681462400413e-154
```

Definition at line 67 of file numerical_inline.hh.

Referenced by jeod::Numerical::square(), and jeod::Numerical::square_incr().

5.3.2.2 MAKE_MATH_MESSAGE_CODE

```
#define MAKE_MATH_MESSAGE_CODE(
              id ) JEOD_MAKE_MESSAGE_CODE(MathMessages, "utils/math/", id)
```

Definition at line 36 of file math_messages.cc.

Chapter 6

Namespace Documentation

6.1 jeod Namespace Reference

Namespace jeod.

Data Structures

- class GaussQuadrature
- class MathMessages

  Specifies the message IDs used in the math model.
- class Matrix3x3

  Provides static methods for operations that involve 3x3 matrices.
- class Numerical

  Provides miscellaneous numerical functions.
- class Vector3

  Provides static methods for operations that involve 3-vectors.

6.1.1 Detailed Description

Namespace jeod.

Chapter 7

Data Structure Documentation

7.1 jeod::GaussQuadrature Class Reference

`#include <gauss_quadrature.hh>`

Static Public Attributes

- static const int max_order = 8
- static const double gauss_weights [max_order+1][max_order]
- static const double gauss_xvalues [max_order+1][max_order]

7.1.1 Detailed Description

Definition at line 68 of file gauss_quadrature.hh.

7.1.2 Field Documentation

7.1.2.1 gauss_weights

```
const double jeod::GaussQuadrature::gauss_weights  [static]
```

Initial value:

```
=
{ { 0.0000000, 0.0000000, 0.0000000, 0.0000000, 0.0000000, 0.0000000,
    0.0000000, 0.0000000 },
  { 2.0000000, 0.0000000, 0.0000000, 0.0000000, 0.0000000, 0.0000000,
    0.0000000, 0.0000000 },
  { 1.0000000, 1.0000000, 0.0000000, 0.0000000, 0.0000000, 0.0000000,
    0.0000000, 0.0000000 },
  { 0.5555556, 0.8888889, 0.5555556, 0.0000000, 0.0000000, 0.0000000,
    0.0000000, 0.0000000 },
  { 0.3478548, 0.6521452, 0.6521452, 0.3478548, 0.0000000, 0.0000000,
    0.0000000, 0.0000000 },
  { 0.2369269, 0.4786287, 0.5688889, 0.4786287, 0.2369269, 0.0000000,
    0.0000000, 0.0000000 },
  { 0.1713245, 0.3607616, 0.4679139, 0.4679139, 0.3607616, 0.1713245,
    0.0000000, 0.0000000 },
  { 0.1294850, 0.2797054, 0.3818301, 0.4179592, 0.3818301, 0.2797054,
    0.1294850, 0.0000000 },
  { 0.1012285, 0.2223810, 0.3137067, 0.3626838, 0.3626838, 0.3137067,
    0.2223810, 0.1012285 }
}
```

Definition at line 72 of file gauss_quadrature.hh.

7.1.2.2 gauss_xvalues

```
const double jeod::GaussQuadrature::gauss_xvalues  [static]
```

Initial value:

```
=
{ { 0.0000000, 0.0000000, 0.0000000, 0.0000000, 0.0000000, 0.0000000,
    0.0000000, 0.0000000 },
  { 0.0000000, 0.0000000, 0.0000000, 0.0000000, 0.0000000, 0.0000000,
    0.0000000, 0.0000000 },
  {-0.5773503, 0.5773503, 0.0000000, 0.0000000, 0.0000000, 0.0000000,
    0.0000000, 0.0000000 },
  {-0.7745967, 0.0000000, 0.7745967, 0.0000000, 0.0000000, 0.0000000,
    0.0000000, 0.0000000 },
  {-0.8611363,-0.3399810, 0.3399810, 0.8611363, 0.0000000, 0.0000000,
    0.0000000, 0.0000000 },
  {-0.9061798,-0.5384693, 0.0000000, 0.5384693, 0.9061798, 0.0000000,
    0.0000000, 0.0000000 },
  {-0.9324695,-0.6612094,-0.2386192, 0.2386192, 0.6612094, 0.9324695,
    0.0000000, 0.0000000 },
  {-0.9491079,-0.7415312,-0.4058452, 0.0000000, 0.4058452, 0.7415312,
    0.9491079, 0.0000000 },
  {-0.9602899,-0.7966665,-0.5255324,-0.1834346, 0.1834346, 0.5255324,
    0.7966665, 0.9602899 }
}
```

Definition at line 73 of file gauss_quadrature.hh.

7.1.2.3 max_order

```
const int jeod::GaussQuadrature::max_order = 8  [static]
```

Definition at line 71 of file gauss_quadrature.hh.

The documentation for this class was generated from the following files:

- gauss_quadrature.hh
- gauss_quadrature.cc

7.2 jeod::MathMessages Class Reference

Specifies the message IDs used in the math model.

```
#include <math_messages.hh>
```

Static Public Attributes

- static char const * ill_conditioned = "utils/math/" "ill_conditioned"

  Error issued when an ill-conditioned matrix is detected.

Private Member Functions

- MathMessages (void)

  Not implemented.
- MathMessages (const MathMessages &)

  Not implemented.
- MathMessages & operator= (const MathMessages &)

  Not implemented.

7.2.1 Detailed Description

Specifies the message IDs used in the math model.

Definition at line 73 of file math_messages.hh.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 MathMessages() [1/2]

```
jeod::MathMessages::MathMessages (
            void  )  [private]
```

Not implemented.

7.2.2.2 MathMessages() [2/2]

```
jeod::MathMessages::MathMessages (
            const MathMessages &  )  [private]
```

Not implemented.

7.2.3 Member Function Documentation

7.2.3.1 operator=()

```
MathMessages& jeod::MathMessages::operator= (
            const MathMessages &  )  [private]
```

Not implemented.

7.2.4 Field Documentation

7.2.4.1 ill_conditioned

```
char const * jeod::MathMessages::ill_conditioned = "utils/math/" "ill_conditioned"  [static]
```

Error issued when an ill-conditioned matrix is detected.

trick_units(–)

Definition at line 81 of file math_messages.hh.

Referenced by jeod::Matrix3x3::invert(), and jeod::Matrix3x3::invert_symmetric().

The documentation for this class was generated from the following files:

- math_messages.hh
- math_messages.cc

7.3 jeod::Matrix3x3 Class Reference

Provides static methods for operations that involve 3x3 matrices.

```
#include <matrix3x3.hh>
```

Static Public Member Functions

- static void initialize (double mat[3][3])
  Zero-fill matrix: mat[i][j] = 0.0.
- static void identity (double mat[3][3])
  Construct identity matrix: mat[i][j] = delta_ij.
- static void cross_matrix (double const vec[3], double cross_mat[3][3])
  Construct the skew symmetric cross product matrix: mat[i][k] = epsilon_ijk vec[j], epsilon_ijk is the Levi-Cevita symbol.
- static void outer_product (double const vec_left[3], double const vec_right[3], double prod[3][3])
  *Construct the outer product of two vectors: mat[i][j] = vec_left[i] * vec_right[j].*
- static void negate (double mat[3][3])
  Negated matrix in-place: mat[i][j] = -mat[i][j].
- static void transpose (double mat[3][3])
  Transpose matrix in-place: mat[i][j] = mat[j][i].
- static void scale (double scalar, double mat[3][3])
  *Scale matrix in-place, mat[i][j] = scalar * mat[i][j].*
- static void incr (double const addend[3][3], double mat[3][3])
  Increment matrix in-place: mat[i][j] = mat[i][j] + addend[i][j].
- static void decr (double const subtrahend[3][3], double mat[3][3])
  Decrement matrix in-place: mat[i][j] = mat[i][j] - subtrahend[i][j].
- static void copy (double const input_mat[3][3], double copy[3][3])
  Copy matrix: copy[i][j] = mat[i][j].

- static void negate (double const input_mat[3][3], double copy[3][3])

  Negate matrix: copy[i][j] = -mat[i][j].
- static void transpose (double const input_mat[3][3], double trans[3][3])

  Transpose matrix: copy[i][j] = mat[j][i].
- static void scale (double const mat[3][3], double scalar, double prod[3][3])

  Scale matrix: copy[i][j] = scalar ∗ mat[i][j].
- static void add (double const augend[3][3], double const addend[3][3], double sum[3][3])

  Add matrices: sum[i][j] = augend[i][j] + addend[i][j].
- static void subtract (double const minuend[3][3], double const subtrahend[3][3], double diff[3][3])

  Subtract matrices: diff[i][j] = minuend[i][j] - subtrahend[i][j].
- static void product (double const mat_left[3][3], double const mat_right[3][3], double prod[3][3])

  Compute the matrix product mat_left ∗ mat_right: prod[i][j] = mat_left[i][k] ∗ mat_right[k][j].
- static void product_left_transpose (double const mat_left[3][3], double const mat_right[3][3], double prod[3][3])

  Compute the matrix product mat_left^ T ∗ mat_right: prod[i][j] = mat_left[k][i] ∗ mat_right[k][j].
- static void product_right_transpose (double const mat_left[3][3], double const mat_right[3][3], double prod[3][3])

  Compute the matrix product mat_left ∗ mat_right^ T: prod[i][j] = sum_k mat_left[i][k] ∗ mat_right[j][k].
- static void product_transpose_transpose (double const mat_left[3][3], double const mat_right[3][3], double prod[3][3])

  Compute the matrix product mat_left^ T ∗ mat_right^ T: prod[i][j] = sum_k mat_left[k][i] ∗ mat_right[j][k].
- static void transform_matrix (double const trans[3][3], double const mat[3][3], double prod[3][3])

  Compute the matrix product trans ∗ mat ∗ trans^ T prod[i][j] = trans[i][k] ∗ mat[k][l] ∗ trans[j][l].
- static void transpose_transform_matrix (double const trans[3][3], double const mat[3][3], double prod[3][3])

  Compute the matrix product trans^ T ∗ mat ∗ trans prod[i][j] = trans[k][i] ∗ mat[k][l] ∗ trans[l][j].
- static int invert (double const matrix[3][3], double inverse[3][3])

  Compute the inverse of a 3x3 matrix.
- static int invert_symmetric (double const matrix[3][3], double inverse[3][3])

  Compute the inverse of a symmetric 3x3 matrix.
- static void print (double const mat[3][3])

  Print matrix to standard error.

7.3.1 Detailed Description

Provides static methods for operations that involve 3x3 matrices.

Definition at line 72 of file matrix3x3.hh.

7.3.2 Member Function Documentation

7.3.2.1 add()

```
void jeod::Matrix3x3::add (
            double const augend[3][3],
            double const addend[3][3],
            double sum[3][3] )  [inline], [static]
```

Add matrices: sum[i][j] = augend[i][j] + addend[i][j].

Parameters

in	*augend*	Matrix
in	*addend*	Matrix
out	*sum*	Sum

Definition at line 410 of file matrix3x3_inline.hh.

7.3.2.2 copy()

```
void jeod::Matrix3x3::copy (
             double const input_mat[3][3],
             double copy[3][3] )  [inline], [static]
```

Copy matrix: copy[i][j] = mat[i][j].

Parameters

in	*input_mat*	Source matrix
out	*copy*	Matrix copy

Definition at line 296 of file matrix3x3_inline.hh.

Referenced by negate().

7.3.2.3 cross_matrix()

```
void jeod::Matrix3x3::cross_matrix (
             double const vec[3],
             double cross_mat[3][3] )  [inline], [static]
```

Construct the skew symmetric cross product matrix: mat[i][k] = epsilon_ijk vec[j], epsilon_ijk is the Levi-Cevita symbol.

Parameters

in	*vec*	Vector
out	*cross_mat*	Cross product matrix

Definition at line 115 of file matrix3x3_inline.hh.

7.3.2.4 decr()

```
void jeod::Matrix3x3::decr (
            double const subtrahend[3][3],
            double mat[3][3] )  [inline], [static]
```

Decrement matrix in-place: mat[i][j] = mat[i][j] - subtrahend[i][j].

Parameters

in	*subtrahend*	Decrement
in,out	*mat*	Decremented matrix

Definition at line 271 of file matrix3x3_inline.hh.

7.3.2.5 identity()

```
void jeod::Matrix3x3::identity (
            double mat[3][3] )  [inline], [static]
```

Construct identity matrix: mat[i][j] = delta_ij.

Parameters

out	*mat*	Identity matrix

Definition at line 94 of file matrix3x3_inline.hh.

7.3.2.6 incr()

```
void jeod::Matrix3x3::incr (
            double const addend[3][3],
            double mat[3][3] )  [inline], [static]
```

Increment matrix in-place: mat[i][j] = mat[i][j] + addend[i][j].

Parameters

in	*addend*	Increment
in,out	*mat*	Incremented matrix

Definition at line 246 of file matrix3x3_inline.hh.

7.3.2.7 initialize()

```
void jeod::Matrix3x3::initialize (
            double mat[3][3] )  [inline], [static]
```

Zero-fill matrix: mat[i][j] = 0.0.

Parameters

out	*mat*	Zero-filled matrix

Definition at line 76 of file matrix3x3_inline.hh.

7.3.2.8 invert()

```
int jeod::Matrix3x3::invert (
            double const matrix[3][3],
            double inverse[3][3] )  [static]
```

Compute the inverse of a 3x3 matrix.

Assumptions and Limitations

- Input and output matrices are distinct.
- Input matrix is well-conditioned.

Returns

0=success, non-zero=singular

Parameters

in	*matrix*	Matrix to invert
out	*inverse*	Inverse

Definition at line 75 of file dm_invert.cc.

References jeod::MathMessages::ill_conditioned.

7.3.2.9 invert_symmetric()

```
int jeod::Matrix3x3::invert_symmetric (
            double const matrix[3][3],
            double inverse[3][3] )  [static]
```

Compute the inverse of a symmetric 3x3 matrix.

Assumptions and Limitations

- Input and output matrices are distinct.
- Input matrix is symmetric.
- Determinate is non-zero.

Returns

0=success, non-zero=singular

Parameters

in	*matrix*	Symmetric matrix to invert
out	*inverse*	Inverse

Definition at line 76 of file dm_invert_symm.cc.

References jeod::MathMessages::ill_conditioned.

7.3.2.10 negate() [1/2]

```
void jeod::Matrix3x3::negate (
             double mat[3][3] )  [inline], [static]
```

Negated matrix in-place: mat[i][j] = -mat[i][j].

Parameters

in,out	*mat*	Negated matrix

Definition at line 169 of file matrix3x3_inline.hh.

7.3.2.11 negate() [2/2]

```
void jeod::Matrix3x3::negate (
             double const input_mat[3][3],
             double copy[3][3] )  [inline], [static]
```

Negate matrix: copy[i][j] = -mat[i][j].

Assumptions and Limitations

- Input and output matrices are distinct.

Parameters

in	*input_mat*	Source matrix
out	*copy*	Negated matrix

Definition at line 325 of file matrix3x3_inline.hh.

References copy().

7.3.2.12 outer_product()

```
void jeod::Matrix3x3::outer_product (
             double const vec_left[3],
             double const vec_right[3],
             double prod[3][3] )  [inline], [static]
```

Construct the outer product of two vectors: mat[i][j] = vec_left[i] ∗ vec_right[j].

Parameters

in	*vec_left*	Vector
in	*vec_right*	Vector
out	*prod*	Outer product matrix

Definition at line 145 of file matrix3x3_inline.hh.

7.3.2.13 print()

```
void jeod::Matrix3x3::print (
             double const mat[3][3] )  [inline], [static]
```

Print matrix to standard error.

Parameters

in	*mat*	Matrix to print

Definition at line 736 of file matrix3x3_inline.hh.

7.3.2.14 product()

```
void jeod::Matrix3x3::product (
             double const mat_left[3][3],
```

```
            double const mat_right[3][3],
            double prod[3][3] )  [inline], [static]
```

Compute the matrix product mat_left ∗ mat_right: prod[i][j] = mat_left[i][k] ∗ mat_right[k][j].

Assumptions and Limitations

- Input and output matrices are distinct.

Parameters

in	*mat_left*	Multiplier
in	*mat_right*	Multiplicand
out	*prod*	Product

Definition at line 468 of file matrix3x3_inline.hh.

Referenced by transform_matrix(), and transpose_transform_matrix().

7.3.2.15 product_left_transpose()

```
void jeod::Matrix3x3::product_left_transpose (
            double const mat_left[3][3],
            double const mat_right[3][3],
            double prod[3][3] )  [inline], [static]
```

Compute the matrix product mat_left^T ∗ mat_right: prod[i][j] = mat_left[k][i] ∗ mat_right[k][j].

Assumptions and Limitations

- Input and output matrices are distinct.

Parameters

in	*mat_left*	Multiplier
in	*mat_right*	Multiplicand
out	*prod*	Product

Definition at line 525 of file matrix3x3_inline.hh.

Referenced by transpose_transform_matrix().

7.3.2.16 product_right_transpose()

```
void jeod::Matrix3x3::product_right_transpose (
            double const mat_left[3][3],
```

```
            double const mat_right[3][3],
            double prod[3][3] )  [inline], [static]
```

Compute the matrix product mat_left ∗ mat_right^T: prod[i][j] = sum_k mat_left[i][k] ∗ mat_right[j][k].

Assumptions and Limitations

- Input and output matrices are distinct.

Parameters

in	*mat_left*	Multiplier
in	*mat_right*	Multiplicand
out	*prod*	Product

Definition at line 582 of file matrix3x3_inline.hh.

Referenced by transform_matrix().

7.3.2.17 product_transpose_transpose()

```
void jeod::Matrix3x3::product_transpose_transpose (
            double const mat_left[3][3],
            double const mat_right[3][3],
            double prod[3][3] )  [inline], [static]
```

Compute the matrix product mat_left^T ∗ mat_right^T: prod[i][j] = sum_k mat_left[k][i] ∗ mat_right[j][k].

Assumptions and Limitations

- Input and output matrices are distinct.

Parameters

in	*mat_left*	Multiplier
in	*mat_right*	Multiplicand
out	*prod*	Product

Definition at line 639 of file matrix3x3_inline.hh.

7.3.2.18 scale() [1/2]

```
void jeod::Matrix3x3::scale (
            double scalar,
            double mat[3][3] )  [inline], [static]
```

Scale matrix in-place, mat[i][j] = scalar ∗ mat[i][j].

Parameters

in	*scalar*	Scalar
in,out	*mat*	Scaled matrix

Definition at line 221 of file matrix3x3_inline.hh.

7.3.2.19 scale() [2/2]

```
void jeod::Matrix3x3::scale (
            double const mat[3][3],
            double scalar,
            double prod[3][3] )  [inline], [static]
```

Scale matrix: copy[i][j] = scalar ∗ mat[i][j].

Parameters

in	*mat*	Matrix
in	*scalar*	Scalar
out	*prod*	Product

Definition at line 383 of file matrix3x3_inline.hh.

7.3.2.20 subtract()

```
void jeod::Matrix3x3::subtract (
            double const minuend[3][3],
            double const subtrahend[3][3],
            double diff[3][3] )  [inline], [static]
```

Subtract matrices: diff[i][j] = minuend[i][j] - subtrahend[i][j].

Parameters

in	*minuend*	Matrix
in	*subtrahend*	Matrix
out	*diff*	Difference

Definition at line 437 of file matrix3x3_inline.hh.

7.3.2.21 transform_matrix()

```
void jeod::Matrix3x3::transform_matrix (
            double const trans[3][3],
            double const mat[3][3],
            double prod[3][3] )  [inline], [static]
```

Compute the matrix product trans ∗ mat ∗ trans^T prod[i][j] = trans[i][k] ∗ mat[k][l] ∗ trans[j][l].

Assumptions and Limitations

- Input and output matrices are distinct.

Parameters

in	*trans*	Transformation matrix
in	*mat*	Matrix to transform
out	*prod*	Product

Definition at line 696 of file matrix3x3_inline.hh.

References product(), and product_right_transpose().

7.3.2.22 transpose() [1/2]

```
void jeod::Matrix3x3::transpose (
            double mat[3][3] )  [inline], [static]
```

Transpose matrix in-place: mat[i][j] = mat[j][i].

Parameters

in,out	*mat*	Transposed matrix

Definition at line 194 of file matrix3x3_inline.hh.

7.3.2.23 transpose() [2/2]

```
void jeod::Matrix3x3::transpose (
            double const input_mat[3][3],
            double trans[3][3] )  [inline], [static]
```

Transpose matrix: copy[i][j] = mat[j][i].

Assumptions and Limitations

- Input and output matrices are distinct.

Parameters

in	*input_mat*	Source matrix
out	*trans*	Matrix transpose

Definition at line 355 of file matrix3x3_inline.hh.

7.3.2.24 transpose_transform_matrix()

```
void jeod::Matrix3x3::transpose_transform_matrix (
             double const trans[3][3],
             double const mat[3][3],
             double prod[3][3] )  [inline], [static]
```

Compute the matrix product trans^T ∗ mat ∗ trans prod[i][j] = trans[k][i] ∗ mat[k][l] ∗ trans[l][j].

Assumptions and Limitations

- Input and output matrices are distinct.

Parameters

in	*trans*	Transformation matrix
in	*mat*	Matrix to transform
out	*prod*	Product

Definition at line 719 of file matrix3x3_inline.hh.

References product(), and product_left_transpose().

The documentation for this class was generated from the following files:

- matrix3x3.hh
- matrix3x3_inline.hh
- dm_invert.cc
- dm_invert_symm.cc

7.4 jeod::Numerical Class Reference

Provides miscellaneous numerical functions.

```
#include <numerical.hh>
```

Static Public Member Functions

- static double fabs (double x)

  Absolute value.
- static double square (double value)

  Compute the square of a number, protecting against undeflow.
- static double square_incr (double value, double &sum)

  Add number squared to accumulator, protecting against undeflow.
- static bool compare_exact (double x, double y)

  Compare two doubles for exact equality.

7.4.1 Detailed Description

Provides miscellaneous numerical functions.

Definition at line 67 of file numerical.hh.

7.4.2 Member Function Documentation

7.4.2.1 compare_exact()

```
bool jeod::Numerical::compare_exact (
            double x,
            double y )  [inline], [static]
```

Compare two doubles for exact equality.

Returns

whether inputs are exactly the same

Parameters

in	*x*	Value1
in	*y*	Value2

Definition at line 130 of file numerical_inline.hh.

7.4.2.2 fabs()

```
double jeod::Numerical::fabs (
            double x )  [inline], [static]
```

Absolute value.

Returns

Absolute value of x

Parameters

in	*x*	x

Definition at line 79 of file numerical_inline.hh.

Referenced by square(), square_incr(), and jeod::Vector3::zero_small().

7.4.2.3 square()

```
double jeod::Numerical::square (
            double value )  [inline], [static]
```

Compute the square of a number, protecting against undeflow.

Returns

value^2 or zero if too small

Parameters

in	*value*	Value

Definition at line 92 of file numerical_inline.hh.

References fabs(), and GSL_SQRT_DBL_MIN.

7.4.2.4 square_incr()

```
double jeod::Numerical::square_incr (
            double value,
            double & sum )  [inline], [static]
```

Add number squared to accumulator, protecting against undeflow.

Returns

Accumulated value

Parameters

in	*value*	Value
in,out	*sum*	Accumulator

Definition at line 112 of file numerical_inline.hh.

References fabs(), and GSL_SQRT_DBL_MIN.

The documentation for this class was generated from the following files:

- numerical.hh
- numerical_inline.hh

7.5 jeod::Vector3 Class Reference

Provides static methods for operations that involve 3-vectors.

`#include <vector3.hh>`

Static Public Member Functions

- static double * initialize (double vec[3])

  Zero-fill vector, vec[i] = 0.0.
- static double * unit (unsigned int index, double vec[3])

  Construct unit vector, vec[i] = delta_ij (delta_ij is the Kronecker delta)
- static double * fill (double scalar, double vec[3])

  Construct a vector from scalar, vec[i] = scalar.
- static double * zero_small (double limit, double vec[3])

  Zero-out small components of a vector, vec[i] = 0 if abs(vec[i]) < limit.
- static double * copy (double const vec[3], double copy[3])

  Copy vector contents, copy[i] = vec[i].
- static double dot (double const vec2[3], double const vec1[3])

  *Compute vector inner product, result = sum_i vec1[i] * vec2[i].*
- static double vmagsq (double const vec[3])

  Compute square of vector magnitude, result = dot(vec,vec), but protects against underflow.
- static double vmag (double const vec[3])

  Compute vector magnitude, result = sqrt(vmagsq(vec))
- static double * normalize (double vec[3])

  *Make vector a unit vector in-place, vec = vec * 1/vmag(vec)*
- static double * normalize (double const vec[3], double unit_vec[3])

  *Construct unit vector, unit_vec = vec * 1/vmag(vec)*
- static double * scale (double scalar, double vec[3])

  Scale a vector in-place, vec[i] = scalar.
- static double * scale (double const vec[3], double scalar, double prod[3])

  *Scale a vector, prod[i] = vec[i] * scalar.*
- static double * negate (double vec[3])

  Negate vector in-place, vec[i] = -vec[i].
- static double * negate (double const vec[3], double copy[3])

  Negate vector, copy[i] = -vec[i].
- static double * transform (double const tmat[3][3], double const vec[3], double prod[3])

  *Transform a column vector, prod[i] = tmat[i][j]*vec[j].*
- static double * transform (double const tmat[3][3], double vec[3])

  *Transform a column vector in-place, vec[i] <- tmat[i][j]*vec[j].*

- static double * transform_transpose (double const tmat[3][3], double const vec[3], double prod[3])

  *Transform a column vector with the transpose, prod[i] = tmat[j][i]*vec[j].*
- static double * transform_transpose (double const tmat[3][3], double vec[3])

  *Transform a column vector in-place with the transpose, vec[i] <- tmat[j][i]*vec[j].*
- static double * incr (double const addend[3], double vec[3])

  Increment a vector, vec[i] += addend[i].
- static double * incr (double const addend1[3], double const addend2[3], double vec[3])

  Increment a vector, vec[i] += addend1[i] + addend2[i].
- static double * decr (double const subtrahend[3], double vec[3])

  Decrement a vector, vec[i] -= subtrahend[i].
- static double * decr (double const subtrahend1[3], double const subtrahend2[3], double vec[3])

  Decrement a vector, vec[i] -= subtrahend1[i] + subtrahend2[i].
- static double * sum (double const addend1[3], double const addend2[3], double vec[3])

  Compute the sum of two vectors, vec[i] = addend1[i] + addend2[i].
- static double * sum (double const addend1[3], double const addend2[3], double const addend3[3], double vec[3])

  Compute the sum of three vectors, vec[i] = addend1[i] + addend2[i] + addend3[i].
- static double * diff (double const minuend[3], double const subtrahend[3], double vec[3])

  Compute the difference between two vectors, diff[i] = minuend[i] - subtrehend[i].
- static double * cross (double const vec_left[3], double const vec_right[3], double prod[3])

  *Compute the cross product between two vectors, prod[i] = epsilon_ijk * vec_left[j] * vec_right[k].*
- static double * scale_incr (double const vec[3], double scalar, double prod[3])

  *Increment a vector with a scaled vector, prod[i] += scalar*vec[i].*
- static double * scale_decr (double const vec[3], double scalar, double prod[3])

  *Decrement a vector with a scaled vector, prod[i] += scalar*vec[i].*
- static double * cross_incr (double const vec_left[3], double const vec_right[3], double prod[3])

  *Increment a vector with the the cross product between two vectors, prod[i] += epsilon_ijk * vec_left[j] * vec_right[k].*
- static double * cross_decr (double const vec_left[3], double const vec_right[3], double prod[3])

  *Decrement a vector with the the cross product between two vectors, prod[i] -= epsilon_ijk * vec_left[j] * vec_right[k].*
- static double * transform_incr (double const tmat[3][3], double const vec[3], double prod[3])

  *Increment a vector with a transformed column vector, prod[i] += tmat[i][j]*vec[j].*
- static double * transform_decr (double const tmat[3][3], double const vec[3], double prod[3])

  *Decrement a vector with a transformed column vector, prod[i] += tmat[i][j]*vec[j].*
- static double * transform_transpose_incr (double const tmat[3][3], double const vec[3], double prod[3])

  *Increment a vector with a transpose-transformed column vector, prod[i] += tmat[j][i]*vec[j].*
- static double * transform_transpose_decr (double const tmat[3][3], double const vec[3], double prod[3])

  *decrement a vector with a transpose-transformed column vector, prod[i] -= tmat[j][i]*vec[j]*

7.5.1 Detailed Description

Provides static methods for operations that involve 3-vectors.

Definition at line 67 of file vector3.hh.

7.5.2 Member Function Documentation

7.5.2.1 copy()

```
double * jeod::Vector3::copy (
            double const vec[3],
            double copy[3] )  [inline], [static]
```

Copy vector contents, copy[i] = vec[i].

Returns

Copied vector

Parameters

in	*vec*	Source vector
out	*copy*	Copied vector

Definition at line 158 of file vector3_inline.hh.

Referenced by negate(), normalize(), transform(), and transform_transpose().

7.5.2.2 cross()

```
double * jeod::Vector3::cross (
            double const vec_left[3],
            double const vec_right[3],
            double prod[3] )  [inline], [static]
```

Compute the cross product between two vectors, prod[i] = epsilon_ijk $*$ vec_left[j] $*$ vec_right[k].

Returns

Cross product vector

Parameters

in	*vec_left*	Left vector
in	*vec_right*	Right vector
out	*prod*	Cross product vector

Definition at line 602 of file vector3_inline.hh.

7.5.2.3 cross_decr()

```
double * jeod::Vector3::cross_decr (
            double const vec_left[3],
```

```
            double const vec_right[3],
            double prod[3] )  [inline], [static]
```

Decrement a vector with the the cross product between two vectors, prod[i] -= epsilon_ijk ∗ vec_left[j] ∗ vec_right[k].

Returns

Decremented vector

Parameters

in	*vec_left*	Left vector
in	*vec_right*	Right vector
in,out	*prod*	Decremented vector

Definition at line 690 of file vector3_inline.hh.

7.5.2.4 cross_incr()

```
double * jeod::Vector3::cross_incr (
            double const vec_left[3],
            double const vec_right[3],
            double prod[3] )  [inline], [static]
```

Increment a vector with the the cross product between two vectors, prod[i] += epsilon_ijk ∗ vec_left[j] ∗ vec_right[k].

Returns

Cross product vector

Parameters

in	*vec_left*	Left vector
in	*vec_right*	Right vector
in,out	*prod*	Cross product vector

Definition at line 668 of file vector3_inline.hh.

7.5.2.5 decr() [1/2]

```
double * jeod::Vector3::decr (
            double const subtrahend[3],
            double vec[3] )  [inline], [static]
```

Decrement a vector, vec[i] -= subtrahend[i].

Returns

Decremented vector

Parameters

in	*subtrahend*	Decrement
in,out	*vec*	Vector

Definition at line 491 of file vector3_inline.hh.

7.5.2.6 decr() [2/2]

```
double * jeod::Vector3::decr (
            double const subtrahend1[3],
            double const subtrahend2[3],
            double vec[3] )  [inline], [static]
```

Decrement a vector, vec[i] -= subtrahend1[i] + subtrahend2[i].

Returns

Decremented vector

Parameters

in	*subtrahend1*	Decrement
in	*subtrahend2*	Decrement
in,out	*vec*	Vector

Definition at line 512 of file vector3_inline.hh.

7.5.2.7 diff()

```
double * jeod::Vector3::diff (
            double const minuend[3],
            double const subtrahend[3],
            double vec[3] )  [inline], [static]
```

Compute the difference between two vectors, diff[i] = minuend[i] - subtrehend[i].

Returns

Difference vector

Parameters

in	*minuend*	Minuend
in	*subtrahend*	Subtrahend
out	*vec*	Difference vector

Definition at line 580 of file vector3_inline.hh.

7.5.2.8 dot()

```
double jeod::Vector3::dot (
            double const vec2[3],
            double const vec1[3] )  [inline], [static]
```

Compute vector inner product, result = sum_i vec1[i] ∗ vec2[i].

Returns

Inner product

Parameters

in	*vec2*	Vector 2
in	*vec1*	Vector 1

Definition at line 178 of file vector3_inline.hh.

7.5.2.9 fill()

```
double * jeod::Vector3::fill (
            double scalar,
            double vec[3] )  [inline], [static]
```

Construct a vector from scalar, vec[i] = scalar.

Returns

Filled vector

Parameters

in	*scalar*	Scalar
out	*vec*	Filled vector

Definition at line 112 of file vector3_inline.hh.

7.5.2.10 incr() [1/2]

```
double * jeod::Vector3::incr (
            double const addend[3],
            double vec[3] )  [inline], [static]
```

Increment a vector, vec[i] += addend[i].

Returns

Incremented vector

Parameters

in	*addend*	Increment
in,out	*vec*	Vector

Definition at line 449 of file vector3_inline.hh.

7.5.2.11 incr() [2/2]

```
double * jeod::Vector3::incr (
            double const addend1[3],
            double const addend2[3],
            double vec[3] )  [inline], [static]
```

Increment a vector, vec[i] += addend1[i] + addend2[i].

Returns

Incremented vector

Parameters

in	*addend1*	Increment
in	*addend2*	Increment
in,out	*vec*	Vector

Definition at line 470 of file vector3_inline.hh.

7.5.2.12 initialize()

```
double * jeod::Vector3::initialize (
            double vec[3] )  [inline], [static]
```

Zero-fill vector, vec[i] = 0.0.

Returns

Zero-filled vector

Parameters

out	*vec*	Zero-filled vector

Definition at line 78 of file vector3_inline.hh.

Referenced by normalize().

7.5.2.13 negate() [1/2]

```
double * jeod::Vector3::negate (
            double vec[3] )  [inline], [static]
```

Negate vector in-place, vec[i] = -vec[i].

Returns

Negated vector

Parameters

in,out	*vec*	Vector

Definition at line 306 of file vector3_inline.hh.

7.5.2.14 negate() [2/2]

```
double * jeod::Vector3::negate (
            double const vec[3],
            double copy[3] )  [inline], [static]
```

Negate vector, copy[i] = -vec[i].

Returns

Negated vector

Parameters

in	*vec*	Source vector
out	*copy*	Negated vector

Definition at line 325 of file vector3_inline.hh.

References copy().

7.5.2.15 normalize() [1/2]

```
double * jeod::Vector3::normalize (
            double vec[3] )  [inline], [static]
```

Make vector a unit vector in-place, vec = vec * 1/vmag(vec)

Returns

Normalized vector

Parameters

in,out	*vec*	Vector

Definition at line 223 of file vector3_inline.hh.

References initialize(), scale(), and vmag().

Referenced by normalize().

7.5.2.16 normalize() [2/2]

```
double * jeod::Vector3::normalize (
            double const vec[3],
            double unit_vec[3] )  [inline], [static]
```

Construct unit vector, unit_vec = vec * 1/vmag(vec)

Returns

Unit vector

Parameters

in	*vec*	Vector
out	*unit_vec*	Unit vector

Definition at line 247 of file vector3_inline.hh.

References copy(), and normalize().

7.5.2.17 scale() [1/2]

```
double * jeod::Vector3::scale (
            double scalar,
            double vec[3] )  [inline], [static]
```

Scale a vector in-place, vec[i] = scalar.

Returns

Scaled vector

Parameters

in	*scalar*	Scalar
in,out	*vec*	Scaled vector

Definition at line 265 of file vector3_inline.hh.

Referenced by normalize().

7.5.2.18 scale() [2/2]

```
double * jeod::Vector3::scale (
            double const vec[3],
            double scalar,
            double prod[3] )  [inline], [static]
```

Scale a vector, prod[i] = vec[i] * scalar.

Returns

Scaled vector

Parameters

in	*vec*	Source vector
in	*scalar*	Scalar
out	*prod*	Scaled vector

Definition at line 286 of file vector3_inline.hh.

7.5.2.19 scale_decr()

```
double * jeod::Vector3::scale_decr (
            double const vec[3],
            double scalar,
            double prod[3] )  [inline], [static]
```

Decrement a vector with a scaled vector, prod[i] += scalar∗vec[i].

Returns

Decremented vector

Parameters

in	*vec*	Source vector
in	*scalar*	Scalar
in,out	*prod*	Decremented vector

Definition at line 646 of file vector3_inline.hh.

7.5.2.20 scale_incr()

```
double * jeod::Vector3::scale_incr (
            double const vec[3],
            double scalar,
            double prod[3] )  [inline], [static]
```

Increment a vector with a scaled vector, prod[i] += scalar∗vec[i].

Returns

Incremented vector

Parameters

in	*vec*	Source vector
in	*scalar*	Scalar
in,out	*prod*	Incremented vector

Definition at line 624 of file vector3_inline.hh.

7.5.2.21 sum() [1/2]

```
double * jeod::Vector3::sum (
             double const addend1[3],
             double const addend2[3],
             double vec[3] )  [inline], [static]
```

Compute the sum of two vectors, vec[i] = addend1[i] + addend2[i].

Returns

Sum vector

Parameters

in	*addend1*	Addend
in	*addend2*	Addend
out	*vec*	Sum vector

Definition at line 534 of file vector3_inline.hh.

7.5.2.22 sum() [2/2]

```
double * jeod::Vector3::sum (
             double const addend1[3],
             double const addend2[3],
             double const addend3[3],
             double vec[3] )  [inline], [static]
```

Compute the sum of three vectors, vec[i] = addend1[i] + addend2[i] + addend3[i].

Returns

Sum vector

Parameters

in	*addend1*	Addend
in	*addend2*	Addend
in	*addend3*	Addend
out	*vec*	Sum vector

Definition at line 557 of file vector3_inline.hh.

7.5.2.23 transform() [1/2]

```
double * jeod::Vector3::transform (
             double const tmat[3][3],
```

```
            double const vec[3],
            double prod[3] )  [inline], [static]
```

Transform a column vector, prod[i] = tmat[i][j]∗vec[j].

Returns

Transformed vector

Parameters

in	*tmat*	Transformation matrix
in	*vec*	Source vector
out	*prod*	Transformed vector

Definition at line 346 of file vector3_inline.hh.

Referenced by transform().

7.5.2.24 transform() [2/2]

```
double * jeod::Vector3::transform (
            double const tmat[3][3],
            double vec[3] )  [inline], [static]
```

Transform a column vector in-place, vec[i] <- tmat[i][j]∗vec[j].

Returns

Transformed vector

Parameters

in	*tmat*	Transformation matrix
in,out	*vec*	Transformed vector

Definition at line 376 of file vector3_inline.hh.

References copy(), and transform().

7.5.2.25 transform_decr()

```
double * jeod::Vector3::transform_decr (
            double const tmat[3][3],
            double const vec[3],
            double prod[3] )  [inline], [static]
```

Decrement a vector with a transformed column vector, prod[i] += tmat[i][j]∗vec[j].

Returns

Decremented vector

Parameters

in	*tmat*	Transformation matrix
in	*vec*	Source vector
in,out	*prod*	Decremented vector

Definition at line 743 of file vector3_inline.hh.

7.5.2.26 transform_incr()

```
double * jeod::Vector3::transform_incr (
            double const tmat[3][3],
            double const vec[3],
            double prod[3] )  [inline], [static]
```

Increment a vector with a transformed column vector, prod[i] += tmat[i][j]∗vec[j].

Returns

Incremented vector

Parameters

in	*tmat*	Transformation matrix
in	*vec*	Source vector
in,out	*prod*	Incremented vector

Definition at line 712 of file vector3_inline.hh.

7.5.2.27 transform_transpose() [1/2]

```
double * jeod::Vector3::transform_transpose (
            double const tmat[3][3],
            double const vec[3],
            double prod[3] )  [inline], [static]
```

Transform a column vector with the transpose, prod[i] = tmat[j][i]∗vec[j].

Returns

Transformed vector

Parameters

in	*tmat*	Transformation matrix
in	*vec*	Source vector
out	*prod*	Transformed vector

Definition at line 398 of file vector3_inline.hh.

Referenced by transform_transpose().

7.5.2.28 transform_transpose() [2/2]

```
double * jeod::Vector3::transform_transpose (
            double const tmat[3][3],
            double vec[3] )  [inline], [static]
```

Transform a column vector in-place with the transpose, vec[i] <- tmat[j][i]*vec[j].

Returns

Transformed vector

Parameters

in	*tmat*	Transformation matrix
in,out	*vec*	Transformed vector

Definition at line 428 of file vector3_inline.hh.

References copy(), and transform_transpose().

7.5.2.29 transform_transpose_decr()

```
double * jeod::Vector3::transform_transpose_decr (
            double const tmat[3][3],
            double const vec[3],
            double prod[3] )  [inline], [static]
```

decrement a vector with a transpose-transformed column vector, prod[i] -= tmat[j][i]*vec[j]

Returns

Decremented vector

Parameters

in	*tmat*	Transformation matrix
in	*vec*	Source vector
in,out	*prod*	Decremented vector

Definition at line 805 of file vector3_inline.hh.

7.5.2.30 transform_transpose_incr()

```
double * jeod::Vector3::transform_transpose_incr (
            double const tmat[3][3],
            double const vec[3],
            double prod[3] )  [inline], [static]
```

Increment a vector with a transpose-transformed column vector, prod[i] += tmat[j][i]∗vec[j].

Returns

Incremented vector

Parameters

in	*tmat*	Transformation matrix
in	*vec*	Source vector
in,out	*prod*	Incremented vector

Definition at line 774 of file vector3_inline.hh.

7.5.2.31 unit()

```
double * jeod::Vector3::unit (
            unsigned int index,
            double vec[3] )  [inline], [static]
```

Construct unit vector, vec[i] = delta_ij (delta_ij is the Kronecker delta)

Returns

Unit vector

Parameters

in	*index*	Unit index: 0,1,2=x,y,z hat
out	*vec*	Unit vector

Definition at line 94 of file vector3_inline.hh.

7.5.2.32 vmag()

```
double jeod::Vector3::vmag (
            double const vec[3] )  [inline], [static]
```

Compute vector magnitude, result = sqrt(vmagsq(vec))

Returns

Vector magnitude

Parameters

in	*vec*	Vector

Definition at line 209 of file vector3_inline.hh.

References vmagsq().

Referenced by normalize().

7.5.2.33 vmagsq()

```
double jeod::Vector3::vmagsq (
            double const vec[3] )  [inline], [static]
```

Compute square of vector magnitude, result = dot(vec,vec), but protects against underflow.

Returns

Inner product

Parameters

in	*vec*	Vector

Definition at line 195 of file vector3_inline.hh.

Referenced by vmag().

7.5.2.34 zero_small()

```
double * jeod::Vector3::zero_small (
            double limit,
            double vec[3] )  [inline], [static]
```

Zero-out small components of a vector, vec[i] = 0 if abs(vec[i]) $<$ limit.

Returns

Truncated vector

Parameters

in	*limit*	Limit
in,out	*vec*	Truncated vector

Definition at line 129 of file vector3_inline.hh.

References jeod::Numerical::fabs().

The documentation for this class was generated from the following files:

- vector3.hh
- vector3_inline.hh

Chapter 8

File Documentation

8.1 dm_invert.cc File Reference

Define Matrix3x3::invert.

```
#include <cmath>
#include "utils/message/include/message_handler.hh"
#include "../include/matrix3x3.hh"
#include "../include/math_messages.hh"
```

Namespaces

- jeod

  Namespace jeod.

8.1.1 Detailed Description

Define Matrix3x3::invert.

8.2 dm_invert_symm.cc File Reference

Define Matrix3x3::invert_symmetric.

```
#include <cmath>
#include "utils/message/include/message_handler.hh"
#include "../include/matrix3x3.hh"
#include "../include/math_messages.hh"
```

Namespaces

- jeod

  Namespace jeod.

8.2.1 Detailed Description

Define Matrix3x3::invert_symmetric.

8.3 gauss_quadrature.cc File Reference

Define Gauss Quadrature functionality.

```
#include "../include/gauss_quadrature.hh"
```

Namespaces

- jeod
  Namespace jeod.

8.3.1 Detailed Description

Define Gauss Quadrature functionality.

8.4 gauss_quadrature.hh File Reference

Gauss Quadrature implementation.

Data Structures

- class jeod::GaussQuadrature

Namespaces

- jeod
  Namespace jeod.

8.4.1 Detailed Description

Gauss Quadrature implementation.

8.5 math_messages.cc File Reference

Implement the class MathMessages.

```
#include "utils/message/include/make_message_code.hh"
#include "../include/math_messages.hh"
```

Namespaces

- jeod

  Namespace jeod.

Macros

- #define MAKE_MATH_MESSAGE_CODE(id) JEOD_MAKE_MESSAGE_CODE(MathMessages, "utils/math/", id)

8.5.1 Detailed Description

Implement the class MathMessages.

8.6 math_messages.hh File Reference

Define the class MathMessages.

Data Structures

- class jeod::MathMessages

  Specifies the message IDs used in the math model.

Namespaces

- jeod

  Namespace jeod.

8.6.1 Detailed Description

Define the class MathMessages.

8.7 matrix3x3.hh File Reference

Matrix math inline functions.

```
#include "matrix3x3_inline.hh"
```

Data Structures

- class jeod::Matrix3x3

  Provides static methods for operations that involve 3x3 matrices.

Namespaces

- jeod

  Namespace jeod.

8.7.1 Detailed Description

Matrix math inline functions.

8.8 matrix3x3_inline.hh File Reference

Matrix math inline functions.

```
#include <cstdio>
#include "matrix3x3.hh"
```

Namespaces

- jeod

  Namespace jeod.

8.8.1 Detailed Description

Matrix math inline functions.

8.9 numerical.hh File Reference

Miscellaneous math inline functions.

```
#include "numerical_inline.hh"
```

Data Structures

- class jeod::Numerical

  Provides miscellaneous numerical functions.

Namespaces

- jeod

  Namespace jeod.

8.9.1 Detailed Description

Miscellaneous math inline functions.

8.10 numerical_inline.hh File Reference

Vector math inline functions.

```
#include "numerical.hh"
```

Namespaces

- jeod

  Namespace jeod.

Macros

- #define GSL_SQRT_DBL_MIN 1.4916681462400413e-154

8.10.1 Detailed Description

Vector math inline functions.

8.11 vector3.hh File Reference

Vector math inline functions.

```
#include "vector3_inline.hh"
```

Data Structures

- class jeod::Vector3

  Provides static methods for operations that involve 3-vectors.

Namespaces

- jeod

  Namespace jeod.

8.11.1 Detailed Description

Vector math inline functions.

8.12 vector3_inline.hh File Reference

Vector math inline functions.

```
#include <cmath>
#include "vector3.hh"
#include "numerical.hh"
```

Namespaces

- jeod

  Namespace jeod.

8.12.1 Detailed Description

Vector math inline functions.

Index