pmat

0.0

Generated by Doxygen 1.9.7

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Namespace Documentation

5.1 pmat Namespace Reference

Namespaces

- namespace messages
- namespace utils

Classes

- class Array

  Abstract entity for arrays of generic dimension.
- class DecompositionCholesky
- class DecompositionPLU
- class DecompositionPQR
- class DecompositionSAS
- class Matrix
- class MatrixLowerTriangular
- class MatrixSkewSymmetric
- class MatrixSquare
- class MatrixSymmetric
- class MatrixSymmetry
- class MatrixTriangular
- class MatrixUpperTriangular
- class TMultiplicationManager
- class TMultiplicationPerformer
- class Vector

  Entity for one-dimensional array.

Enumerations

- enum class SubMatrixPos { lower , upper }
- enum class TriangType { UPPER , LOWER }

5.1.1 Enumeration Type Documentation

5.1.1.1 SubMatrixPos

```
enum class pmat::SubMatrixPos  [strong]
```

Enumerator

lower	
upper	

5.1.1.2 TriangType

```
enum class pmat::TriangType  [strong]
```

Enumerator

UPPER	
LOWER	

5.2 pmat::messages Namespace Reference

Variables

- constexpr const char * DATA_NOT_READ {"Error reading file data"}
- constexpr const char * FILE_NOT_OPEN {"Error opening file"}
- constexpr const char * INDEX_OUT {"Index out of bounds"}
- constexpr const char * NONCOMPT_SIZE_ARG {"Argument is not compatible in size"}
- constexpr const char * MATRIX_SINGULAR {"Matrix is singular"}
- constexpr const char * MATRIX_NOT_LU {"Matrix not LU decomposable"}
- constexpr const char * MATRIX_NOT_L {"Matrix not positive definite"}
- constexpr const char * DECOMP_NOT_LU {"Calculation mode was not set to Strict LU"}

5.2.1 Variable Documentation

5.2.1.1 DATA_NOT_READ

```
constexpr const char* pmat::messages::DATA_NOT_READ {"Error reading file data"}  [constexpr]
```

5.2.1.2 DECOMP_NOT_LU

```
constexpr const char* pmat::messages::DECOMP_NOT_LU {"Calculation mode was not set to Strict
LU"}  [constexpr]
```

5.2.1.3 FILE_NOT_OPEN

```
constexpr const char* pmat::messages::FILE_NOT_OPEN {"Error opening file"}  [constexpr]
```

5.2.1.4 INDEX_OUT

```
constexpr const char* pmat::messages::INDEX_OUT {¨Index out of bounds¨}  [constexpr]
```

5.2.1.5 MATRIX_NOT_L

```
constexpr const char* pmat::messages::MATRIX_NOT_L {¨Matrix not positive definite¨}  [constexpr]
```

5.2.1.6 MATRIX_NOT_LU

```
constexpr const char* pmat::messages::MATRIX_NOT_LU {¨Matrix not LU decomposable¨}  [constexpr]
```

5.2.1.7 MATRIX_SINGULAR

```
constexpr const char* pmat::messages::MATRIX_SINGULAR {¨Matrix is singular¨}  [constexpr]
```

5.2.1.8 NONCOMPT_SIZE_ARG

```
constexpr const char* pmat::messages::NONCOMPT_SIZE_ARG {¨Argument is not compatible in size¨}
[constexpr]
```

5.3 pmat::utils Namespace Reference

Chapter 6

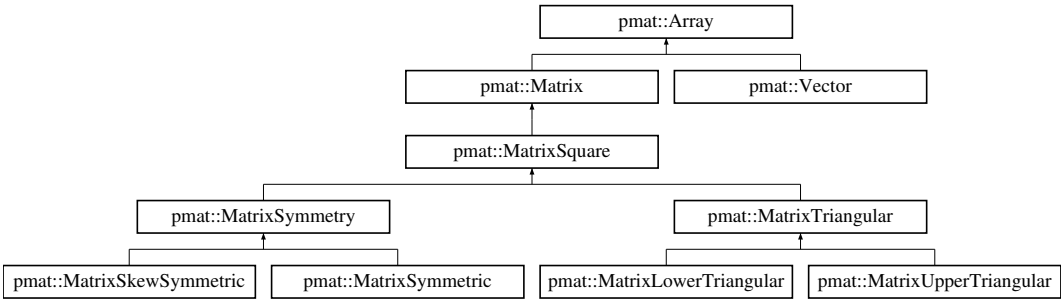
Class Documentation

6.1 pmat::Array Class Reference

Abstract entity for arrays of generic dimension.

`#include <Array.h>`

Inheritance diagram for pmat::Array:



Public Member Functions

- Array ()=default
- Array (const Array &array)=default
- Array (Array &&)=default
- Array & operator= (const Array &)=default
- Array & operator= (Array &&)=default
- virtual ~Array ()=default
- virtual unsigned length () const =0

  Informs the number of elements.
- virtual unsigned dimension () const =0

  Informs the dimension of the array.
- virtual void clear ()=0

  Removes all elements and sets size zero.
- virtual unsigned occurrences (const double &value) const =0

  Informes the number of occurrences of a value in the array.
- virtual void fillWithRandomValues (const double &min, const double &max)=0

  Fills the array with random values.

6.1.1 Detailed Description

Abstract entity for arrays of generic dimension.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 Array() [1/3]

```
pmat::Array::Array ( )  [default]
```

6.1.2.2 Array() [2/3]

```
pmat::Array::Array (
            const Array & array )  [default]
```

6.1.2.3 Array() [3/3]

```
pmat::Array::Array (
            Array &&  )  [default]
```

6.1.2.4 ~Array()

```
virtual pmat::Array::~Array ( )  [virtual], [default]
```

6.1.3 Member Function Documentation

6.1.3.1 clear()

```
virtual void pmat::Array::clear ( )  [pure virtual]
```

Removes all elements and sets size zero.

Implemented in pmat::Matrix, and pmat::Vector.

6.1.3.2 dimension()

```
virtual unsigned pmat::Array::dimension ( ) const  [pure virtual]
```

Informs the dimension of the array.

Returns

unsigned Dimension

Implemented in pmat::Matrix, and pmat::Vector.

6.1.3.3 fillWithRandomValues()

```
virtual void pmat::Array::fillWithRandomValues (
            const double & min,
            const double & max )  [pure virtual]
```

Fills the array with random values.

Parameters

min	Minimum acceptable value
max	Maximum acceptable value

Implemented in pmat::Matrix, pmat::MatrixLowerTriangular, pmat::MatrixSkewSymmetric, pmat::MatrixSymmetric, pmat::MatrixUpperTriangular, pmat::Vector, pmat::MatrixSymmetry, and pmat::MatrixTriangular.

6.1.3.4 length()

```
virtual unsigned pmat::Array::length ( ) const  [pure virtual]
```

Informs the number of elements.

Returns

unsigned Number of elements

Implemented in pmat::Matrix, pmat::MatrixSymmetry, pmat::MatrixTriangular, and pmat::Vector.

6.1.3.5 occurrences()

```
virtual unsigned pmat::Array::occurrences (
            const double & value ) const  [pure virtual]
```

Informes the number of occurrences of a value in the array.

Parameters

value	Value to be searched

Returns

unsigned Number of occurrences

Implemented in pmat::Matrix, and pmat::Vector.

6.1.3.6 operator=() [1/2]

```
Array & pmat::Array::operator= (
            Array &&  )  [default]
```

6.1.3.7 operator=() [2/2]

```
Array & pmat::Array::operator= (
            const Array &  )  [default]
```

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

- src/Array.h

6.2 pmat::DecompositionCholesky Class Reference

`#include <DecompositionCholesky.h>`

Public Member Functions

- DecompositionCholesky (const MatrixSymmetric &matrix)
- DecompositionCholesky (const DecompositionCholesky &chk)=default
- DecompositionCholesky (DecompositionCholesky &&chk)=default
- DecompositionCholesky & operator= (const DecompositionCholesky &chk)=default
- DecompositionCholesky & operator= (DecompositionCholesky &&chk)=default
- ~DecompositionCholesky ()=default
- const MatrixLowerTriangular & choleskyFactor ()

  Calculates the Cholesky factor of the associated matrix, if possible.
- double determinant ()

  Calculates the determinant.
- bool isInvertible ()

  Verifies whether the associated matrix is invertible or not.
- MatrixSquare inverse ()

  Calculates the inverse of the associated matrix, if possible.
- MatrixSymmetric inverseAsSymmetric ()

  Calculates the inverse of the associated matrix, if possible.
- Vector linearSolve (const Vector &rhs)

  Finds the solution of the linear system in which the associated matrix in on the left hand side.
- bool isPositiveDefinite ()

  Informs if the associated matrix is positive definite.

6.2.1 Constructor & Destructor Documentation

6.2.1.1 DecompositionCholesky() [1/3]

```
pmat::DecompositionCholesky::DecompositionCholesky (
            const MatrixSymmetric & matrix )
```

6.2.1.2 DecompositionCholesky() [2/3]

```
pmat::DecompositionCholesky::DecompositionCholesky (
            const DecompositionCholesky & chk )  [default]
```

6.2.1.3 DecompositionCholesky() [3/3]

```
pmat::DecompositionCholesky::DecompositionCholesky (
            DecompositionCholesky && chk )  [default]
```

6.2.1.4 ~DecompositionCholesky()

```
pmat::DecompositionCholesky::~DecompositionCholesky ( )  [default]
```

6.2.2 Member Function Documentation

6.2.2.1 choleskyFactor()

```
const pmat::MatrixLowerTriangular & pmat::DecompositionCholesky::choleskyFactor ( )
```

Calculates the Cholesky factor of the associated matrix, if possible.

The Cholesky Factor of positive-define matrix A is a lower-triangular matrix L where

$$A = LL^T$$

Returns

const MatrixLowerTriangular& Cholesky Factor

Exceptions

std::logic_error	Matrix not positive definite

6.2.2.2 determinant()

```
double pmat::DecompositionCholesky::determinant ( )
```

Calculates the determinant.

Returns

double Determinant

6.2.2.3 inverse()

```
pmat::MatrixSquare pmat::DecompositionCholesky::inverse ( )
```

Calculates the inverse of the associated matrix, if possible.

If the associated matrix S is invertible, its inverse is obtained from $S^{-1} = L^{-T}L^{-1}$

Returns

MatrixSquare The inverse of the associated matrix

Exceptions

std::logic_error	Matrix is singular

6.2.2.4 inverseAsSymmetric()

```
pmat::MatrixSymmetric pmat::DecompositionCholesky::inverseAsSymmetric ( )
```

Calculates the inverse of the associated matrix, if possible.

Returns

MatrixSymmetric The inverse of the associated matrix

Exceptions

std::logic_error	Matrix is singular

6.2.2.5 isInvertible()

```
bool pmat::DecompositionCholesky::isInvertible ( )
```

Verifies whether the associated matrix is invertible or not.

Returns

true The associated matrix is invertible

false The associated matrix is singular

6.2.2.6 isPositiveDefinite()

```
bool pmat::DecompositionCholesky::isPositiveDefinite ( )
```

Informs if the associated matrix is positive definite.

A symmetric matrix is considered to be positive definite if it is Cholesky decomposable

See also

¨Matrix Computations¨, Golub & Van Loan, ISBN 9789380250755, p. 164.

Returns

True if this matrix is positive definite

6.2.2.7 linearSolve()

```
pmat::Vector pmat::DecompositionCholesky::linearSolve (
            const Vector & rhs )
```

Finds the solution of the linear system in which the associated matrix in on the left hand side.

Parameters

rhs	The right hand side of the linear system

Returns

Vector Solution of the linear system

Exceptions

std::invalid_argument	Vector not compatible
std::logic_error	The associated matrix is singular

6.2.2.8 operator=() [1/2]

```
DecompositionCholesky & pmat::DecompositionCholesky::operator= (
            const DecompositionCholesky & chk )  [default]
```

6.2.2.9 operator=() [2/2]

```
DecompositionCholesky & pmat::DecompositionCholesky::operator= (
            DecompositionCholesky && chk )  [default]
```

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

- src/DecompositionCholesky.h
- src/DecompositionCholesky.cpp

6.3 pmat::DecompositionPLU Class Reference

```
#include <DecompositionPLU.h>
```

Public Member Functions

- DecompositionPLU (const MatrixSquare &matrix)
- DecompositionPLU (const MatrixSquare &matrix, bool strictLUMode)

  Construct a new Decomposition PLU calculator.
- DecompositionPLU (const DecompositionPLU &plu)=default
- DecompositionPLU (DecompositionPLU &&plu)=default
- DecompositionPLU & operator= (const DecompositionPLU &plu)=default
- DecompositionPLU & operator= (DecompositionPLU &&plu)=default
- ~DecompositionPLU ()=default
- const MatrixSquare & matP ()

  Calculates the permutation matrix P of the PLU Decomposition.
- const MatrixLowerTriangular & matL ()

Calculates the lower triangular matrix L of the PLU Decomposition.

- const MatrixUpperTriangular & matU ()

  Calculates the upper triangular matrix U of the PLU Decomposition.
- const std::vector< std::pair< unsigned, unsigned > > & swappedRows () const

  Generates a list of the rows swapped in order to calculate de PLU decomposition.
- double determinant ()

  Calculates the determinant.
- bool isStrictLUDecomposable ()

  Verifies whether the associated matrix is LU decomposable or not.
- bool isInvertible ()

  Verifies whether the associated matrix is invertible or not.
- MatrixSquare inverse ()

  Calculates the inverse of the associated matrix, if possible.
- bool isPositiveDefinite ()

  Verifies whether the associated matrix is positive definite or not.
- bool isOrthogonal ()

  Verifies whether the associated matrix is orthogonal or not.
- Vector linearSolve (const Vector &rhs)

  Finds the solution of the linear system in which the associated matrix in on the left hand side.
- bool isStrictLUMode () const

  Verifies whether the associated matrix is orthogonal or not.
- void setStrictLUMode ()

  Specifies if the decomposition is LU ou PLU.

6.3.1 Constructor & Destructor Documentation

6.3.1.1 DecompositionPLU() [1/4]

```
pmat::DecompositionPLU::DecompositionPLU (
            const MatrixSquare & matrix )
```

6.3.1.2 DecompositionPLU() [2/4]

```
pmat::DecompositionPLU::DecompositionPLU (
            const MatrixSquare & matrix,
            bool strictLUMode )
```

Construct a new Decomposition PLU calculator.

Parameters

matrix	Associated matrix
strictLUMode	specifies if the decomposition is LU or PLU

6.3.1.3 DecompositionPLU() [3/4]

```
pmat::DecompositionPLU::DecompositionPLU (
            const DecompositionPLU & plu )  [default]
```

6.3.1.4 DecompositionPLU() [4/4]

```
pmat::DecompositionPLU::DecompositionPLU (
            DecompositionPLU && plu )  [default]
```

6.3.1.5 ~DecompositionPLU()

```
pmat::DecompositionPLU::~DecompositionPLU ( )  [default]
```

6.3.2 Member Function Documentation

6.3.2.1 determinant()

```
double pmat::DecompositionPLU::determinant ( )
```

Calculates the determinant.

Returns

double Determinant

6.3.2.2 inverse()

```
pmat::MatrixSquare pmat::DecompositionPLU::inverse ( )
```

Calculates the inverse of the associated matrix, if possible.

If the associated matrix A is invertible, its inverse is obtained from $A^{-1} = U^{-1}L^{-1}P$

Returns

MatrixSquare The inverse of the associated matrix

Exceptions

std::logic_error	Matrix is singular

6.3.2.3 isInvertible()

```
bool pmat::DecompositionPLU::isInvertible ( )
```

Verifies whether the associated matrix is invertible or not.

Returns

true The associated matrix is invertible

false The associated matrix is singular

6.3.2.4 isOrthogonal()

```
bool pmat::DecompositionPLU::isOrthogonal ( )
```

Verifies whether the associated matrix is orthogonal or not.

A matrix is orthogonal if its inverse equals its transpose

Returns

true The associated matrix is orthogonal

false The associated matrix is not orthogonal

6.3.2.5 isPositiveDefinite()

```
bool pmat::DecompositionPLU::isPositiveDefinite ( )
```

Verifies whether the associated matrix is positive definite or not.

Considering the PLU decomposition, a matrix is considered to be positive definite if every diagonal element of U is positive

See also

¨Matrix Computations¨, Golub & Van Loan, ISBN 9789380250755, p. 161.

Returns

true The associated matrix is positive definite

false The associated matrix is not positive definite

6.3.2.6 isStrictLUDecomposable()

```
bool pmat::DecompositionPLU::isStrictLUDecomposable ( )
```

Verifies whether the associated matrix is LU decomposable or not.

Returns

true The associated matrix is LU decomposable

false The associated matrix is not LU decomposable

6.3.2.7 isStrictLUMode()

```
bool pmat::DecompositionPLU::isStrictLUMode ( ) const  [inline]
```

Verifies whether the associated matrix is orthogonal or not.

Returns

true The associated matrix is orthogonal

false The associated matrix is not orthogonal

6.3.2.8 linearSolve()

```
pmat::Vector pmat::DecompositionPLU::linearSolve (
            const Vector & rhs )
```

Finds the solution of the linear system in which the associated matrix in on the left hand side.

Parameters

rhs	The right hand side of the linear system

Returns

Vector Solution of the linear system

Exceptions

std::invalid_argument	Vector not compatible
std::logic_error	The associated matrix is singular

6.3.2.9 matL()

```
const pmat::MatrixLowerTriangular & pmat::DecompositionPLU::matL ( )
```

Calculates the lower triangular matrix L of the PLU Decomposition.

Returns

const MatrixLowerTriangular& Lower triangular matrix

6.3.2.10 matP()

```
const pmat::MatrixSquare & pmat::DecompositionPLU::matP ( )
```

Calculates the permutation matrix P of the PLU Decomposition.

Returns

const MatrixSquare& Permutation matrix

6.3.2.11 matU()

```
const pmat::MatrixUpperTriangular & pmat::DecompositionPLU::matU ( )
```

Calculates the upper triangular matrix U of the PLU Decomposition.

Returns

const MatrixUpperTriangular& Upper triangular matrix

6.3.2.12 operator=() [1/2]

```
DecompositionPLU & pmat::DecompositionPLU::operator= (
            const DecompositionPLU & plu )  [default]
```

6.3.2.13 operator=() [2/2]

```
DecompositionPLU & pmat::DecompositionPLU::operator= (
            DecompositionPLU && plu )  [default]
```

6.3.2.14 setStrictLUMode()

```
void pmat::DecompositionPLU::setStrictLUMode ( )
```

Specifies if the decomposition is LU ou PLU.

6.3.2.15 swappedRows()

```
const std::vector< std::pair< unsigned, unsigned > > & pmat::DecompositionPLU::swappedRows (
) const
```

Generates a list of the rows swapped in order to calculate de PLU decomposition.

Returns

const std::vector<std::pair<unsigned, unsigned>>& List of swapped rows

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

- src/DecompositionPLU.h
- src/DecompositionPLU.cpp

6.4 pmat::DecompositionPQR Class Reference

```
#include <DecompositionPQR.h>
```

Public Member Functions

- DecompositionPQR (const MatrixSquare &matrix)
- DecompositionPQR (const DecompositionPQR &pqr)=default
- DecompositionPQR (DecompositionPQR &&pqr)=default
- DecompositionPQR & operator= (const DecompositionPQR &pqr)=default
- DecompositionPQR & operator= (DecompositionPQR &&pqr)=default
- ~DecompositionPQR ()=default
- const MatrixSquare & matP ()

  Calculates the permutation matrix P of the PQR Decomposition.
- const MatrixSquare & matQ ()

  Calculates the orthonormal matrix Q of the PQR Decomposition.
- const MatrixUpperTriangular & matR ()

  Calculates the upper triangular matrix R of the PQR Decomposition.
- const unsigned & rank ()

  Calculates the rank ot the associated matrix.
- bool isInvertible ()

  Verifies whether the associated matrix is invertible or not.
- MatrixSquare inverse ()

  Calculates the inverse of the associated matrix, if possible.

6.4.1 Constructor & Destructor Documentation

6.4.1.1 DecompositionPQR() [1/3]

```
pmat::DecompositionPQR::DecompositionPQR (
            const MatrixSquare & matrix )
```

6.4.1.2 DecompositionPQR() [2/3]

```
pmat::DecompositionPQR::DecompositionPQR (
            const DecompositionPQR & pqr )  [default]
```

6.4.1.3 DecompositionPQR() [3/3]

```
pmat::DecompositionPQR::DecompositionPQR (
            DecompositionPQR && pqr )  [default]
```

6.4.1.4 ~DecompositionPQR()

```
pmat::DecompositionPQR::~DecompositionPQR ( )  [default]
```

6.4.2 Member Function Documentation

6.4.2.1 inverse()

```
pmat::MatrixSquare pmat::DecompositionPQR::inverse ( )
```

Calculates the inverse of the associated matrix, if possible.

If the associated matrix A is invertible, its inverse is obtained from $A^{-1} = PR^{-1}Q^{-1}$

Returns

MatrixSquare The inverse of the associated matrix

Exceptions

std::logic_error	Matrix is singular

Recovering adequate positions by swapping rows in reverse order of the swapped columns

6.4.2.2 isInvertible()

```
bool pmat::DecompositionPQR::isInvertible ( )
```

Verifies whether the associated matrix is invertible or not.

Returns

true The associated matrix is invertible

false The associated matrix is singular

6.4.2.3 matP()

```
const pmat::MatrixSquare & pmat::DecompositionPQR::matP ( )
```

Calculates the permutation matrix P of the PQR Decomposition.

Returns

const MatrixSquare& Permutation Matrix

6.4.2.4 matQ()

```
const pmat::MatrixSquare & pmat::DecompositionPQR::matQ ( )
```

Calculates the orthonormal matrix Q of the PQR Decomposition.

Returns

const MatrixSquare& Orthonormal matrix

6.4.2.5 matR()

```
const pmat::MatrixUpperTriangular & pmat::DecompositionPQR::matR ( )
```

Calculates the upper triangular matrix R of the PQR Decomposition.

Returns

const MatrixUpperTriangular& Matrix R

6.4.2.6 operator=() [1/2]

```
DecompositionPQR & pmat::DecompositionPQR::operator= (
            const DecompositionPQR & pqr )  [default]
```

6.4.2.7 operator=() [2/2]

```
DecompositionPQR & pmat::DecompositionPQR::operator= (
            DecompositionPQR && pqr )  [default]
```

6.4.2.8 rank()

```
const unsigned & pmat::DecompositionPQR::rank ( )
```

Calculates the rank ot the associated matrix.

The rank of a matrix is the maximum number of its linearly independent columns

Returns

const unsigned& Rank of the associated matrix

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

- src/DecompositionPQR.h
- src/DecompositionPQR.cpp

6.5 pmat::DecompositionSAS Class Reference

```
#include <DecompositionSAS.h>
```

Public Member Functions

- DecompositionSAS (const MatrixSquare &matrix)
- DecompositionSAS (const DecompositionSAS &sas)=default
- DecompositionSAS (DecompositionSAS &&sas)=default
- DecompositionSAS & operator= (const DecompositionSAS &sas)=default
- DecompositionSAS & operator= (DecompositionSAS &&sas)=default
- ~DecompositionSAS ()=default
- const MatrixSymmetric & matS ()

  Calculates the symmetric part of the associated matrix.
- const MatrixSkewSymmetric & matAS ()

  Calculates the skew-symmetric part of the associated matrix.

6.5.1 Constructor & Destructor Documentation

6.5.1.1 DecompositionSAS() [1/3]

```
pmat::DecompositionSAS::DecompositionSAS (
            const MatrixSquare & matrix )
```

6.5.1.2 DecompositionSAS() [2/3]

```
pmat::DecompositionSAS::DecompositionSAS (
            const DecompositionSAS & sas )  [default]
```

6.5.1.3 DecompositionSAS() [3/3]

```
pmat::DecompositionSAS::DecompositionSAS (
            DecompositionSAS && sas )  [default]
```

6.5.1.4 ~DecompositionSAS()

```
pmat::DecompositionSAS::~DecompositionSAS ( )  [default]
```

6.5.2 Member Function Documentation

6.5.2.1 matAS()

```
const pmat::MatrixSkewSymmetric & pmat::DecompositionSAS::matAS ( )
```

Calculates the skew-symmetric part of the associated matrix.

Returns

const MatrixSkewSymmetric& Skew-symmetric part

6.5.2.2 matS()

```
const pmat::MatrixSymmetric & pmat::DecompositionSAS::matS ( )
```

Calculates the symmetric part of the associated matrix.

Returns

const MatrixSymmetric& Symmetric part

6.5.2.3 operator=() [1/2]

```
DecompositionSAS & pmat::DecompositionSAS::operator= (
            const DecompositionSAS & sas )  [default]
```

6.5.2.4 operator=() [2/2]

```
DecompositionSAS & pmat::DecompositionSAS::operator= (
            DecompositionSAS && sas )  [default]
```

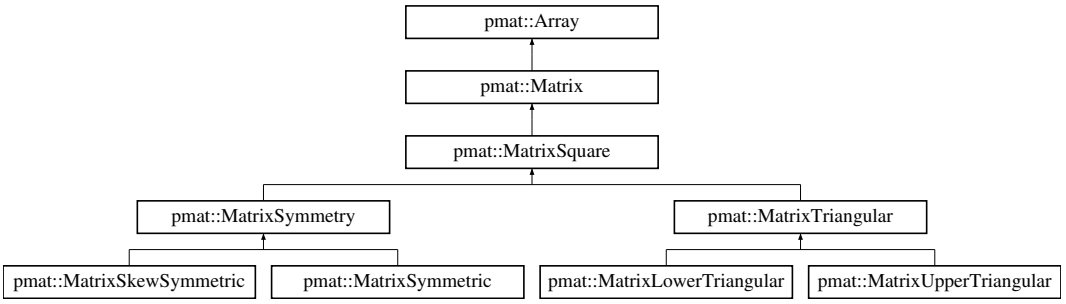
The documentation for this class was generated from the following files:

- src/DecompositionSAS.h
- src/DecompositionSAS.cpp

6.6 pmat::Matrix Class Reference

`#include <Matrix.h>`

Inheritance diagram for pmat::Matrix:



Public Member Functions

- Matrix ()=default
- Matrix (const unsigned &rowSize, const unsigned &columnSize)
- Matrix (const std::string &fileName)
- Matrix (const Matrix &matrix)
- Matrix (Matrix &&matrix) noexcept
- ~Matrix () override=default
- unsigned length () const override

  Informs the number of elements.
- unsigned dimension () const override

  Informs the dimension of the array.
- void resize (const unsigned &rowSize, const unsigned &columnSize)

  Clears this matrix and sets a new size.
- void clear () override

  Removes all elements and sets size zero.
- virtual void setValue (const double &value, const unsigned &row, const unsigned &column)

  Sets the informed value at the informed position.
- virtual double operator() (const unsigned &row, const unsigned &column) const

  Informs the value at the informed position.
- unsigned rowSize () const

  Informs the size of matrix row dimension.
- unsigned columnSize () const

  Informs the size of matrix column dimension.
- Matrix & operator= (const Matrix &matrix)
- Matrix & operator= (Matrix &&matrix) noexcept
- virtual bool operator== (const Matrix &matrix) const
- virtual double dotProduct (const Matrix &matrix) const

  Calculates the dot product of this matrix with the informed matrix.
- Matrix operator+ (const Matrix &matrix) const

  Sums this matrix with the informed matrix.
- void addBy (const Matrix &matrix)

  Sums this matrix with the informed matrix, setting the result in this matrix.
- Matrix operator- (const Matrix &matrix) const

  Subtracts this matrix with the informed matrix.

- void subtractBy (const Matrix &matrix)

  Subtracts this matrix with the informed matrix, setting the result in this matrix.
- virtual Matrix operator∗ (const Matrix &matrix) const

  Multiplies this matrix and the informed matrix.
- virtual Vector operator∗ (const Vector &vector) const

  Multiplies this matrix and the informed vector, considered as a column matrix.
- Matrix operator∗ (const double &scalar) const

  Multiplies this matrix by the informed scalar.
- virtual void multiplyBy (const double &scalar)

  Multiplies this matrix and the informed scalar, setting the result in this matrix.
- Matrix multiply (const Matrix &matrix, unsigned nThreads)

  Multiplies this matrix by the informed scalar using multiple threads.
- Matrix multiplyHadamardBy (const Matrix &matrix) const

  Performs the Hadamard multiplication of this matrix and the informed matrix.
- virtual void multiplyRowBy (const unsigned &row, const double &scalar)

  Multiplies all the elements of the informed row by the informed scalar.
- virtual void multiplyColumnBy (const unsigned &column, const double &scalar)

  Multiplies all the elements of the informed column by the informed scalar.
- virtual void swapRows (const unsigned &rowA, const unsigned &rowB, const unsigned &startColumn, const unsigned &endColumn)

  Swaps the rows at the informed positions in a range of columns.
- virtual void swapRows (const unsigned &rowA, const unsigned &rowB)

  Swaps the rows at the informed positions.
- virtual void swapColumns (const unsigned &columnA, const unsigned &columnB, const unsigned &startRow, const unsigned &endRow)

  Swaps the columns at the informed positions in a range of rows.
- virtual void swapColumns (const unsigned &columnA, const unsigned &columnB)

  Swaps the columns at the informed positions.
- virtual void transpose ()

  Transposes this matrix.
- virtual double getFrobeniusNorm () const

  Calculates the Frobenius Norm of this matrix.
- void fillWithRandomValues (const double &min, const double &max) override

  Fills the array with random values.
- Vector rowToVector (const unsigned &row) const

  Gets the informed row as a vector.
- Vector columnToVector (const unsigned &column) const

  Gets the informed column as a vector.
- unsigned occurrences (const double &value) const override

  Informes the number of occurrences of a value in the array.
- virtual unsigned occurrencesInRow (const unsigned row, const double &value) const

  Informs the number of occurrences of the informed value at the informed row.
- virtual unsigned occurrencesInColumn (const unsigned column, const double &value) const

  Informs the number of occurrences of the informed value at the informed column.

Public Member Functions inherited from pmat::Array

- Array ()=default
- Array (const Array &array)=default
- Array (Array &&)=default
- Array & operator= (const Array &)=default
- Array & operator= (Array &&)=default
- virtual ~Array ()=default
- virtual unsigned length () const =0

  Informs the number of elements.
- virtual unsigned dimension () const =0

  Informs the dimension of the array.
- virtual void clear ()=0

  Removes all elements and sets size zero.
- virtual unsigned occurrences (const double &value) const =0

  Informes the number of occurrences of a value in the array.
- virtual void fillWithRandomValues (const double &min, const double &max)=0

  Fills the array with random values.

Protected Member Functions

- virtual unsigned vectorIndex (const unsigned &row, const unsigned &column) const
- double vectorElement (const unsigned &row, const unsigned &column) const
- void moveToThis (Matrix &&matrix)
- void initializeMembers (unsigned rowSize, unsigned columnSize, bool isTransposed)
- void copyMembers (const Matrix &matrix)
- bool isTransposed () const

6.6.1 Constructor & Destructor Documentation

6.6.1.1 Matrix() [1/5]

```
pmat::Matrix::Matrix ( )  [default]
```

6.6.1.2 Matrix() [2/5]

```
pmat::Matrix::Matrix (
            const unsigned & rowSize,
            const unsigned & columnSize )
```

6.6.1.3 Matrix() [3/5]

```
pmat::Matrix::Matrix (
            const std::string & fileName )
```

6.6.1.4 Matrix() [4/5]

```
pmat::Matrix::Matrix (
            const Matrix & matrix )  [inline]
```

6.6.1.5 Matrix() [5/5]

```
pmat::Matrix::Matrix (
            Matrix && matrix )  [inline], [noexcept]
```

6.6.1.6 ~Matrix()

```
pmat::Matrix::~Matrix ( )  [override], [default]
```

6.6.2 Member Function Documentation

6.6.2.1 addBy()

```
void pmat::Matrix::addBy (
            const Matrix & matrix )
```

Sums this matrix with the informed matrix, setting the result in this matrix.

Parameters

matrix	Second operand

Exceptions

std::invalid_argument	Incompatible sizes

6.6.2.2 clear()

```
void pmat::Matrix::clear ( )  [override], [virtual]
```

Removes all elements and sets size zero.

Implements pmat::Array.

6.6.2.3 columnSize()

```
unsigned pmat::Matrix::columnSize ( ) const  [inline]
```

Informs the size of matrix column dimension.

Returns

unsigned Row size

6.6.2.4 columnToVector()

```
pmat::Vector pmat::Matrix::columnToVector (
            const unsigned & column ) const
```

Gets the informed column as a vector.

Parameters

column	Column postion

Returns

Vector Informed column as a vector

Exceptions

std::invalid_argument	Index out of bounds

6.6.2.5 copyMembers()

```
void pmat::Matrix::copyMembers (
            const Matrix & matrix )  [protected]
```

6.6.2.6 dimension()

```
unsigned pmat::Matrix::dimension ( ) const  [inline], [override], [virtual]
```

Informs the dimension of the array.

Returns

unsigned Dimension

Implements pmat::Array.

6.6.2.7 dotProduct()

```
double pmat::Matrix::dotProduct (
            const Matrix & matrix ) const  [virtual]
```

Calculates the dot product of this matrix with the informed matrix.

The dot product of matrices A and B is

$$A : B = \sum_{i,j} A_{ij} B_{ij}$$

Parameters

matrix	Second operand

Returns

double Dot product result

Exceptions

std::invalid_argument	Operands are not compatible

Reimplemented in pmat::MatrixLowerTriangular, pmat::MatrixUpperTriangular, and pmat::MatrixTriangular.

6.6.2.8 fillWithRandomValues()

```
void pmat::Matrix::fillWithRandomValues (
            const double & min,
            const double & max )  [override], [virtual]
```

Fills the array with random values.

Parameters

min	Minimum acceptable value
max	Maximum acceptable value

Implements pmat::Array.

Reimplemented in pmat::MatrixLowerTriangular, pmat::MatrixSkewSymmetric, pmat::MatrixSymmetric, pmat::MatrixUpperTriangular, pmat::MatrixSymmetry, and pmat::MatrixTriangular.

6.6.2.9 getFrobeniusNorm()

```
double pmat::Matrix::getFrobeniusNorm ( ) const  [virtual]
```

Calculates the Frobenius Norm of this matrix.

Frobenius Norm of matrix A is calculated from the dot product the following way:

$$\sqrt{A : A}$$

Returns

Frobenius Norm result

6.6.2.10 initializeMembers()

```
void pmat::Matrix::initializeMembers (
            unsigned rowSize,
            unsigned columnSize,
            bool isTransposed )  [protected]
```

6.6.2.11 isTransposed()

```
bool pmat::Matrix::isTransposed ( ) const  [inline], [protected]
```

6.6.2.12 length()

```
unsigned pmat::Matrix::length ( ) const  [inline], [override], [virtual]
```

Informs the number of elements.

Returns

unsigned Number of elements

Implements pmat::Array.

Reimplemented in pmat::MatrixSymmetry, and pmat::MatrixTriangular.

6.6.2.13 moveToThis()

```
void pmat::Matrix::moveToThis (
            Matrix && matrix )  [protected]
```

6.6.2.14 multiply()

```
pmat::Matrix pmat::Matrix::multiply (
            const Matrix & matrix,
            unsigned nThreads )
```

Multiplies this matrix by the informed scalar using multiple threads.

Parameters

matrix	Right operand of the multiplication
nThreads	number of threads to be created

Returns

Matrix Multiplication Result

6.6.2.15 multiplyBy()

```
void pmat::Matrix::multiplyBy (
            const double & scalar )  [virtual]
```

Multiplies this matrix and the informed scalar, setting the result in this matrix.

Parameters

scalar	Second operand of the multiplication

Reimplemented in pmat::MatrixLowerTriangular, pmat::MatrixSkewSymmetric, pmat::MatrixSymmetric, and pmat::MatrixUpperTriangular.

6.6.2.16 multiplyColumnBy()

```
void pmat::Matrix::multiplyColumnBy (
            const unsigned & column,
            const double & scalar )  [virtual]
```

Multiplies all the elements of the informed column by the informed scalar.

Parameters

column	Column to be multiplied
scalar	Value to multiply the row

Exceptions

std::invalid_argument	Index out of bounds

6.6.2.17 multiplyHadamardBy()

```
pmat::Matrix pmat::Matrix::multiplyHadamardBy (
            const Matrix & matrix ) const
```

Performs the Hadamard multiplication of this matrix and the informed matrix.

The Hadamard multiplication C of matrices A and B is

$$C_{ij} = A_{ij} B_{ij}$$

Parameters

matrix	

Returns

Matrix Multiplication result

Exceptions

std::invalid_argument	Incompatible sizes

6.6.2.18 multiplyRowBy()

```
void pmat::Matrix::multiplyRowBy (
            const unsigned & row,
            const double & scalar )  [virtual]
```

Multiplies all the elements of the informed row by the informed scalar.

Parameters

row	Row to be multiplied
scalar	Value to multiply the row

Exceptions

std::invalid_argument	Index out of bounds

6.6.2.19 occurrences()

```
unsigned pmat::Matrix::occurrences (
            const double & value ) const  [override], [virtual]
```

Informes the number of occurrences of a value in the array.

Parameters

value	Value to be searched

Returns

unsigned Number of occurrences

Implements pmat::Array.

6.6.2.20 occurrencesInColumn()

```
unsigned pmat::Matrix::occurrencesInColumn (
            const unsigned column,
            const double & value ) const  [virtual]
```

Informs the number of occurrences of the informed value at the informed column.

Parameters

column	Column position
value	Value to be searched

Returns

unsigned Number of occurrences

Exceptions

std::invalid_argument	Index out of bounds

6.6.2.21 occurrencesInRow()

```
unsigned pmat::Matrix::occurrencesInRow (
            const unsigned row,
            const double & value ) const  [virtual]
```

Informs the number of occurrences of the informed value at the informed row.

Parameters

row	Row position
value	Value to be searched

Returns

unsigned Number of occurrences

Exceptions

std::invalid_argument	Index out of bounds

6.6.2.22 operator()()

```
double pmat::Matrix::operator() (
            const unsigned & row,
            const unsigned & column ) const  [virtual]
```

Informs the value at the informed position.

Parameters

row	Row position of the value to be informed
column	Column position of the value to be informed

Returns

double Value at the informed position

Reimplemented in pmat::MatrixLowerTriangular, pmat::MatrixSkewSymmetric, pmat::MatrixSymmetric, pmat::MatrixUpperTriangular, pmat::MatrixSymmetry, and pmat::MatrixTriangular.

6.6.2.23 operator∗() [1/3]

```
pmat::Matrix pmat::Matrix::operator* (
            const double & scalar ) const
```

Multiplies this matrix by the informed scalar.

Parameters

scalar	Second operand of the multiplication

Returns

Matrix Multiplication result

6.6.2.24 operator∗() [2/3]

```
pmat::Matrix pmat::Matrix::operator* (
            const Matrix & matrix ) const  [virtual]
```

Multiplies this matrix and the informed matrix.

Parameters

matrix	Right operand

Returns

Matrix Multiplication result

Exceptions

std::invalid_argument	Incompatible sizes

Reimplemented in pmat::MatrixSkewSymmetric, and pmat::MatrixSymmetric.

6.6.2.25 operator∗() [3/3]

```
pmat::Vector pmat::Matrix::operator* (
            const Vector & vector ) const  [virtual]
```

Multiplies this matrix and the informed vector, considered as a column matrix.

Parameters

vector	Right operand

Returns

Vector Multiplication result

Exceptions

std::invalid_argument	Incompatible sizes

Reimplemented in pmat::MatrixLowerTriangular, pmat::MatrixSkewSymmetric, pmat::MatrixSquare, pmat::MatrixSymmetric, and pmat::MatrixUpperTriangular.

6.6.2.26 operator+()

```
pmat::Matrix pmat::Matrix::operator+ (
            const Matrix & matrix ) const
```

Sums this matrix with the informed matrix.

Parameters

matrix	Second operand of the sum

Returns

Matrix Sum result

Exceptions

std::invalid_argument	Incompatible sizes

6.6.2.27 operator-()

```
pmat::Matrix pmat::Matrix::operator- (
            const Matrix & matrix ) const
```

Subtracts this matrix with the informed matrix.

Parameters

matrix	Second operand of the subtraction

Returns

Matrix Subtraction result

Exceptions

std::invalid_argument	Incompatible sizes

6.6.2.28 operator=() [1/2]

```
pmat::Matrix & pmat::Matrix::operator= (
            const Matrix & matrix )
```

6.6.2.29 operator=() [2/2]

```
pmat::Matrix & pmat::Matrix::operator= (
            Matrix && matrix )  [noexcept]
```

6.6.2.30 operator==()

```
bool pmat::Matrix::operator== (
            const Matrix & matrix ) const  [virtual]
```

6.6.2.31 resize()

```
void pmat::Matrix::resize (
            const unsigned & rowSize,
            const unsigned & columnSize )
```

Clears this matrix and sets a new size.

Parameters

rowSize	New row size
columnSize	New column size

6.6.2.32 rowSize()

```
unsigned pmat::Matrix::rowSize ( ) const  [inline]
```

Informs the size of matrix row dimension.

Returns

unsigned Row size

6.6.2.33 rowToVector()

```
pmat::Vector pmat::Matrix::rowToVector (
            const unsigned & row ) const
```

Gets the informed row as a vector.

Parameters

row	Row postion

Returns

Vector Informed row as a vector

Exceptions

std::invalid_argument	Index out of bounds

6.6.2.34 setValue()

```
void pmat::Matrix::setValue (
            const double & value,
            const unsigned & row,
            const unsigned & column )  [virtual]
```

Sets the informed value at the informed position.

Parameters

value	Value to be set
row	Row position
column	Column position

6.6.2.35 subtractBy()

```
void pmat::Matrix::subtractBy (
            const Matrix & matrix )
```

Subtracts this matrix with the informed matrix, setting the result in this matrix.

Parameters

matrix	Second operand

Exceptions

std::invalid_argument	Incompatible sizes

6.6.2.36 swapColumns() [1/2]

```
void pmat::Matrix::swapColumns (
```

```
            const unsigned & columnA,
            const unsigned & columnB )  [virtual]
```

Swaps the columns at the informed positions.

Parameters

columnA	Position A
columnB	Position B

Exceptions

std::invalid_argument	Index out of bounds

6.6.2.37 swapColumns() [2/2]

```
void pmat::Matrix::swapColumns (
            const unsigned & columnA,
            const unsigned & columnB,
            const unsigned & startRow,
            const unsigned & endRow )  [virtual]
```

Swaps the columns at the informed positions in a range of rows.

Parameters

columnA	Position A
columnB	Position B
startRow	Start row position
endRow	End row postion

Exceptions

std::invalid_argument	Index out of bounds

Reimplemented in pmat::MatrixLowerTriangular, pmat::MatrixUpperTriangular, and pmat::MatrixTriangular.

6.6.2.38 swapRows() [1/2]

```
void pmat::Matrix::swapRows (
            const unsigned & rowA,
            const unsigned & rowB )  [virtual]
```

Swaps the rows at the informed positions.

Parameters

rowA	Position A
rowB	Position B

Exceptions

std::invalid_argument	Index out of bounds

6.6.2.39 swapRows() [2/2]

```
void pmat::Matrix::swapRows (
            const unsigned & rowA,
            const unsigned & rowB,
            const unsigned & startColumn,
            const unsigned & endColumn ) [virtual]
```

Swaps the rows at the informed positions in a range of columns.

Parameters

rowA	Position A
rowB	Position B
startColumn	Start column position
endColumn	End column postion

Exceptions

std::invalid_argument	Index out of bounds

Reimplemented in pmat::MatrixLowerTriangular, pmat::MatrixUpperTriangular, and pmat::MatrixTriangular.

6.6.2.40 transpose()

```
void pmat::Matrix::transpose ( ) [virtual]
```

Transposes this matrix.

Reimplemented in pmat::MatrixSkewSymmetric, pmat::MatrixSymmetric, and pmat::MatrixSymmetry.

6.6.2.41 vectorElement()

```
double pmat::Matrix::vectorElement (
            const unsigned & row,
            const unsigned & column ) const [protected]
```

6.6.2.42 vectorIndex()

```
unsigned pmat::Matrix::vectorIndex (
            const unsigned & row,
            const unsigned & column ) const [protected], [virtual]
```

Reimplemented in pmat::MatrixLowerTriangular, pmat::MatrixSymmetry, pmat::MatrixUpperTriangular, and pmat::MatrixTriangular.

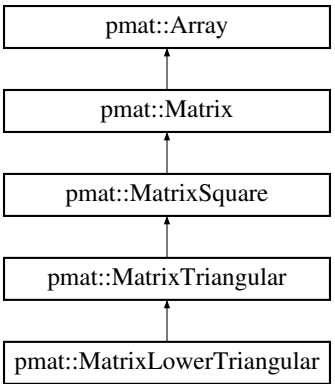
The documentation for this class was generated from the following files:

- src/Matrix.h
- src/Matrix.cpp

6.7 pmat::MatrixLowerTriangular Class Reference

`#include <MatrixLowerTriangular.h>`

Inheritance diagram for pmat::MatrixLowerTriangular:



Public Member Functions

- MatrixLowerTriangular ()=default
- MatrixLowerTriangular (const unsigned &size)
- MatrixLowerTriangular (const MatrixLowerTriangular &matrix)
- MatrixLowerTriangular (MatrixLowerTriangular &&matrix)
- MatrixLowerTriangular & operator= (const MatrixLowerTriangular &matrix)=default
- MatrixLowerTriangular & operator= (MatrixLowerTriangular &&matrix)=default
- ~MatrixLowerTriangular () override=default
- double operator() (const unsigned &row, const unsigned &column) const override

  Informs the value at the informed position.
- double dotProduct (const Matrix &matrix) const override

  Calculates the dot product of this matrix with the informed matrix.
- MatrixLowerTriangular operator+ (const MatrixLowerTriangular &matrix) const
- virtual void addBy (const MatrixLowerTriangular &matrix)
- MatrixLowerTriangular operator- (const MatrixLowerTriangular &matrix) const
- virtual void subtractBy (const MatrixLowerTriangular &matrix)
- MatrixLowerTriangular operator* (const double &scalar) const
- MatrixSquare operator* (const MatrixSquare &matrix) const
- void multiplyBy (const double &scalar) override

  Multiplies this matrix and the informed scalar, setting the result in this matrix.
- MatrixSquare operator+ (const MatrixSquare &matrix) const
- MatrixSquare operator- (const MatrixSquare &matrix) const
- MatrixSquare operator* (const MatrixTriangular &matrix) const
- MatrixLowerTriangular operator* (const MatrixLowerTriangular &matrix) const
- Vector operator* (const Vector &vector) const override

  Multiplies this matrix and the informed vector, considered as a column matrix.
- MatrixUpperTriangular getTranspose () const

  Gets the transposed matrix of this lower triangular matrix.
- void swapRows (const unsigned &rowA, const unsigned &rowB, const unsigned &startColumn, const unsigned &endColumn) override

  Swaps the rows at the informed positions in a range of columns.

- void swapColumns (const unsigned &colA, const unsigned &colB, const unsigned &startRow, const unsigned &endRow) override

  Swaps the columns at the informed positions in a range of rows.
- void fillWithRandomValues (const double &min, const double &max) override

  Fills the array with random values.
- TriangType type () const override

  Informs the triangular type of his matrix.
- MatrixLowerTriangular inverse ()

  Calculates the inverse of this matrix through back substitution.

Public Member Functions inherited from pmat::MatrixTriangular

- MatrixTriangular ()=default
- MatrixTriangular (const MatrixTriangular &matrix)
- MatrixTriangular (MatrixTriangular &&matrix)=default
- MatrixTriangular (const unsigned &size)
- ~MatrixTriangular () override=default
- MatrixTriangular & operator= (const MatrixTriangular &)=default
- MatrixTriangular & operator= (MatrixTriangular &&)=default
- unsigned length () const override

  Informs the number of elements.
- double operator() (const unsigned &row, const unsigned &column) const override=0

  Informs the value at the informed position.
- double dotProduct (const Matrix &matrix) const override=0

  Calculates the dot product of this matrix with the informed matrix.
- MatrixSquare operator* (const MatrixTriangular &matrix) const
- MatrixSquare getSwappedByRows (const unsigned &rowIndexA, const unsigned &rowIndexB) const
- MatrixSquare getSwappedByColumns (const unsigned &columnIndexA, const unsigned &columnIndexB) const
- void fillWithRandomValues (const double &min, const double &max) override=0

  Fills the array with random values.
- void swapRows (const unsigned &rowA, const unsigned &rowB, const unsigned &startColumn, const unsigned &endColumn) override=0

  Swaps the rows at the informed positions in a range of columns.
- void swapColumns (const unsigned &columnA, const unsigned &columnB, const unsigned &startRow, const unsigned &endRow) override=0

  Swaps the columns at the informed positions in a range of rows.
- virtual TriangType type () const =0
- double determinant ()

  Calculates the determinant of this matrix through its diagonal.
- virtual bool isInvertible ()

  Informs if this matrix is invertible by inspecting its diagonal.
- Vector linearSolve (const Vector &rhs)

  Calculates the solution of a linear system by back substitution.

Public Member Functions inherited from pmat::MatrixSquare

- MatrixSquare ()=default
- MatrixSquare (MatrixSquare &&matrix) noexcept=default
- MatrixSquare (Matrix &&matrix)
- MatrixSquare (const unsigned &size)
- MatrixSquare (const MatrixSquare &matrix)
- ~MatrixSquare () override=default
- MatrixSquare & operator= (const MatrixSquare &matrix)=default
- MatrixSquare & operator= (MatrixSquare &&matrix) noexcept=default
- unsigned size () const
- virtual void resize (const unsigned &size)
- MatrixSquare operator+ (const MatrixSquare &matrix) const
- MatrixSquare operator- (const MatrixSquare &matrix) const
- MatrixSquare operator* (const MatrixSquare &matrix) const
- MatrixSquare operator* (const double &scalar) const
- Vector operator* (const Vector &vector) const override

  Multiplies this matrix and the informed vector, considered as a column matrix.
- virtual void fillDiagonalWith (const double &value)
- virtual MatrixSquare multiplyByBiggerMatrix (const MatrixSquare &matrix, SubMatrixPos pos)

  Calculates the multiplication of this matrix and the first parameter.
- virtual double trace () const

  Calculates the trace of this matrix.
- DecompositionPLU decomposeToPLU () const

  Returns a calculator for the PLU Decomposition of this matrix.
- DecompositionSAS decomposeToSAS () const

  Returns a calculator for the SAS Decomposition of this matrix.
- DecompositionPQR decomposeToPQR () const

  Returns a calculator for the PQR Decomposition of this matrix.

Public Member Functions inherited from pmat::Matrix

- Matrix ()=default
- Matrix (const unsigned &rowSize, const unsigned &columnSize)
- Matrix (const std::string &fileName)
- Matrix (const Matrix &matrix)
- Matrix (Matrix &&matrix) noexcept
- ~Matrix () override=default
- unsigned length () const override

  Informs the number of elements.
- unsigned dimension () const override

  Informs the dimension of the array.
- void resize (const unsigned &rowSize, const unsigned &columnSize)

  Clears this matrix and sets a new size.
- void clear () override

  Removes all elements and sets size zero.
- virtual void setValue (const double &value, const unsigned &row, const unsigned &column)

  Sets the informed value at the informed position.
- virtual double operator() (const unsigned &row, const unsigned &column) const

  Informs the value at the informed position.
- unsigned rowSize () const

  Informs the size of matrix row dimension.

- unsigned columnSize () const

  Informs the size of matrix column dimension.
- Matrix & operator= (const Matrix &matrix)
- Matrix & operator= (Matrix &&matrix) noexcept
- virtual bool operator== (const Matrix &matrix) const
- virtual double dotProduct (const Matrix &matrix) const

  Calculates the dot product of this matrix with the informed matrix.
- Matrix operator+ (const Matrix &matrix) const

  Sums this matrix with the informed matrix.
- void addBy (const Matrix &matrix)

  Sums this matrix with the informed matrix, setting the result in this matrix.
- Matrix operator- (const Matrix &matrix) const

  Subtracts this matrix with the informed matrix.
- void subtractBy (const Matrix &matrix)

  Subtracts this matrix with the informed matrix, setting the result in this matrix.
- virtual Matrix operator∗ (const Matrix &matrix) const

  Multiplies this matrix and the informed matrix.
- virtual Vector operator∗ (const Vector &vector) const

  Multiplies this matrix and the informed vector, considered as a column matrix.
- Matrix operator∗ (const double &scalar) const

  Multiplies this matrix by the informed scalar.
- virtual void multiplyBy (const double &scalar)

  Multiplies this matrix and the informed scalar, setting the result in this matrix.
- Matrix multiply (const Matrix &matrix, unsigned nThreads)

  Multiplies this matrix by the informed scalar using multiple threads.
- Matrix multiplyHadamardBy (const Matrix &matrix) const

  Performs the Hadamard multiplication of this matrix and the informed matrix.
- virtual void multiplyRowBy (const unsigned &row, const double &scalar)

  Multiplies all the elements of the informed row by the informed scalar.
- virtual void multiplyColumnBy (const unsigned &column, const double &scalar)

  Multiplies all the elements of the informed column by the informed scalar.
- virtual void swapRows (const unsigned &rowA, const unsigned &rowB, const unsigned &startColumn, const unsigned &endColumn)

  Swaps the rows at the informed positions in a range of columns.
- virtual void swapRows (const unsigned &rowA, const unsigned &rowB)

  Swaps the rows at the informed positions.
- virtual void swapColumns (const unsigned &columnA, const unsigned &columnB, const unsigned &startRow, const unsigned &endRow)

  Swaps the columns at the informed positions in a range of rows.
- virtual void swapColumns (const unsigned &columnA, const unsigned &columnB)

  Swaps the columns at the informed positions.
- virtual void transpose ()

  Transposes this matrix.
- virtual double getFrobeniusNorm () const

  Calculates the Frobenius Norm of this matrix.
- void fillWithRandomValues (const double &min, const double &max) override

  Fills the array with random values.
- Vector rowToVector (const unsigned &row) const

  Gets the informed row as a vector.
- Vector columnToVector (const unsigned &column) const

  Gets the informed column as a vector.

- unsigned occurrences (const double &value) const override

  Informes the number of occurrences of a value in the array.
- virtual unsigned occurrencesInRow (const unsigned row, const double &value) const

  Informs the number of occurrences of the informed value at the informed row.
- virtual unsigned occurrencesInColumn (const unsigned column, const double &value) const

  Informs the number of occurrences of the informed value at the informed column.

Public Member Functions inherited from pmat::Array

- Array ()=default
- Array (const Array &array)=default
- Array (Array &&)=default
- Array & operator= (const Array &)=default
- Array & operator= (Array &&)=default
- virtual ~Array ()=default
- virtual unsigned length () const =0

  Informs the number of elements.
- virtual unsigned dimension () const =0

  Informs the dimension of the array.
- virtual void clear ()=0

  Removes all elements and sets size zero.
- virtual unsigned occurrences (const double &value) const =0

  Informes the number of occurrences of a value in the array.
- virtual void fillWithRandomValues (const double &min, const double &max)=0

  Fills the array with random values.

Protected Member Functions

- unsigned vectorIndex (const unsigned &i, const unsigned &j) const override

- unsigned vectorIndex (const unsigned &i, const unsigned &j) const override=0

Protected Member Functions inherited from pmat::Matrix

- virtual unsigned vectorIndex (const unsigned &row, const unsigned &column) const
- double vectorElement (const unsigned &row, const unsigned &column) const
- void moveToThis (Matrix &&matrix)
- void initializeMembers (unsigned rowSize, unsigned columnSize, bool isTransposed)
- void copyMembers (const Matrix &matrix)
- bool isTransposed () const

Additional Inherited Members

Static Public Member Functions inherited from pmat::MatrixTriangular

- static void findInverseByBackSubstitution (const MatrixTriangular &matrix, MatrixTriangular &resp)

  Independent function for finding inverse by back substitution.
- static pmat::Vector findSolutionByBackSubstitution (const MatrixTriangular &matrix, const Vector &rhs)

  Independent function for find the solution of a linear system by back substitution.

6.7.1 Constructor & Destructor Documentation

6.7.1.1 MatrixLowerTriangular() [1/4]

```
pmat::MatrixLowerTriangular::MatrixLowerTriangular ( )  [default]
```

6.7.1.2 MatrixLowerTriangular() [2/4]

```
pmat::MatrixLowerTriangular::MatrixLowerTriangular (
            const unsigned & size )  [inline], [explicit]
```

6.7.1.3 MatrixLowerTriangular() [3/4]

```
pmat::MatrixLowerTriangular::MatrixLowerTriangular (
            const MatrixLowerTriangular & matrix )  [inline]
```

6.7.1.4 MatrixLowerTriangular() [4/4]

```
pmat::MatrixLowerTriangular::MatrixLowerTriangular (
            MatrixLowerTriangular && matrix )  [inline]
```

6.7.1.5 ~MatrixLowerTriangular()

```
pmat::MatrixLowerTriangular::~MatrixLowerTriangular ( )  [override], [default]
```

6.7.2 Member Function Documentation

6.7.2.1 addBy()

```
void pmat::MatrixLowerTriangular::addBy (
            const MatrixLowerTriangular & matrix )  [virtual]
```

6.7.2.2 dotProduct()

```
double pmat::MatrixLowerTriangular::dotProduct (
            const Matrix & matrix ) const  [override], [virtual]
```

Calculates the dot product of this matrix with the informed matrix.

The dot product of matrices A and B is

$$A : B = \sum_{i,j} A_{ij} B_{ij}$$

Parameters

matrix	Second operand

Returns

double Dot product result

Exceptions

std::invalid_argument	Operands are not compatible

Implements pmat::MatrixTriangular.

6.7.2.3 fillWithRandomValues()

```
void pmat::MatrixLowerTriangular::fillWithRandomValues (
            const double & min,
            const double & max )  [override], [virtual]
```

Fills the array with random values.

Parameters

min	Minimum acceptable value
max	Maximum acceptable value

Implements pmat::MatrixTriangular.

6.7.2.4 getTranspose()

```
pmat::MatrixUpperTriangular pmat::MatrixLowerTriangular::getTranspose ( ) const
```

Gets the transposed matrix of this lower triangular matrix.

Returns

MatrixUpperTriangular Transposed matrix

6.7.2.5 inverse()

```
pmat::MatrixLowerTriangular pmat::MatrixLowerTriangular::inverse ( )
```

Calculates the inverse of this matrix through back substitution.

Returns

MatrixLowerTriangular The inverse of this matrix

Exceptions

throw	std::logic_error Singular matrix

6.7.2.6 multiplyBy()

```
void pmat::MatrixLowerTriangular::multiplyBy (
            const double & scalar )  [override], [virtual]
```

Multiplies this matrix and the informed scalar, setting the result in this matrix.

Parameters

scalar	Second operand of the multiplication

Reimplemented from pmat::Matrix.

6.7.2.7 operator()()

```
double pmat::MatrixLowerTriangular::operator() (
            const unsigned & row,
            const unsigned & column ) const  [override], [virtual]
```

Informs the value at the informed position.

Parameters

row	Row position of the value to be informed
column	Column position of the value to be informed

Returns

double Value at the informed position

Implements pmat::MatrixTriangular.

6.7.2.8 operator*() [1/5]

```
pmat::MatrixLowerTriangular pmat::MatrixLowerTriangular::operator* (
            const double & scalar ) const
```

6.7.2.9 operator*() [2/5]

```
pmat::MatrixLowerTriangular pmat::MatrixLowerTriangular::operator* (
            const MatrixLowerTriangular & matrix ) const
```

6.7.2.10 operator*() [3/5]

```
pmat::MatrixSquare pmat::MatrixLowerTriangular::operator* (
            const MatrixSquare & matrix ) const
```

6.7.2.11 operator*() [4/5]

```
pmat::MatrixSquare pmat::MatrixLowerTriangular::operator* (
            const MatrixTriangular & matrix ) const
```

6.7.2.12 operator*() [5/5]

```
pmat::Vector pmat::MatrixLowerTriangular::operator* (
            const Vector & vector ) const  [override], [virtual]
```

Multiplies this matrix and the informed vector, considered as a column matrix.

Parameters

vector	Right operand

Returns

Vector Multiplication result

Exceptions

std::invalid_argument	Incompatible sizes

Reimplemented from pmat::Matrix.

6.7.2.13 operator+() [1/2]

```
pmat::MatrixLowerTriangular pmat::MatrixLowerTriangular::operator+ (
            const MatrixLowerTriangular & matrix ) const
```

6.7.2.14 operator+() [2/2]

```
pmat::MatrixSquare pmat::MatrixLowerTriangular::operator+ (
            const MatrixSquare & matrix ) const
```

6.7.2.15 operator-() [1/2]

```
pmat::MatrixLowerTriangular pmat::MatrixLowerTriangular::operator- (
            const MatrixLowerTriangular & matrix ) const
```

6.7.2.16 operator-() [2/2]

```
pmat::MatrixSquare pmat::MatrixLowerTriangular::operator- (
            const MatrixSquare & matrix ) const
```

6.7.2.17 operator=() [1/2]

```
MatrixLowerTriangular & pmat::MatrixLowerTriangular::operator= (
            const MatrixLowerTriangular & matrix )  [default]
```

6.7.2.18 operator=() [2/2]

```
MatrixLowerTriangular & pmat::MatrixLowerTriangular::operator= (
            MatrixLowerTriangular && matrix )  [default]
```

6.7.2.19 subtractBy()

```
void pmat::MatrixLowerTriangular::subtractBy (
            const MatrixLowerTriangular & matrix )  [virtual]
```

6.7.2.20 swapColumns()

```
void pmat::MatrixLowerTriangular::swapColumns (
            const unsigned & columnA,
            const unsigned & columnB,
            const unsigned & startRow,
            const unsigned & endRow )  [override], [virtual]
```

Swaps the columns at the informed positions in a range of rows.

Parameters

columnA	Position A
columnB	Position B
startRow	Start row position
endRow	End row postion

Exceptions

std::invalid_argument	Index out of bounds

Implements pmat::MatrixTriangular.

6.7.2.21 swapRows()

```
void pmat::MatrixLowerTriangular::swapRows (
            const unsigned & rowA,
```

```
            const unsigned & rowB,
            const unsigned & startColumn,
            const unsigned & endColumn )  [override], [virtual]
```

Swaps the rows at the informed positions in a range of columns.

Parameters

rowA	Position A
rowB	Position B
startColumn	Start column position
endColumn	End column postion

Exceptions

std::invalid_argument	Index out of bounds

Implements pmat::MatrixTriangular.

6.7.2.22 type()

```
TriangType pmat::MatrixLowerTriangular::type ( ) const  [inline], [override], [virtual]
```

Informs the triangular type of his matrix.

Returns

TriangType Triangular type

Implements pmat::MatrixTriangular.

6.7.2.23 vectorIndex()

```
unsigned pmat::MatrixLowerTriangular::vectorIndex (
            const unsigned & i,
            const unsigned & j ) const  [override], [protected], [virtual]
```

Implements pmat::MatrixTriangular.

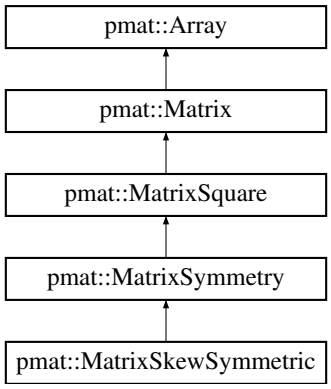
The documentation for this class was generated from the following files:

- src/MatrixLowerTriangular.h
- src/MatrixLowerTriangular.cpp

6.8 pmat::MatrixSkewSymmetric Class Reference

`#include <MatrixSkewSymmetric.h>`

Inheritance diagram for pmat::MatrixSkewSymmetric:



Public Member Functions

- MatrixSkewSymmetric ()=default
- MatrixSkewSymmetric (const unsigned &size)
- MatrixSkewSymmetric (const MatrixSkewSymmetric &matrix)
- MatrixSkewSymmetric (MatrixSkewSymmetric &&matrix)
- MatrixSkewSymmetric & operator= (const MatrixSkewSymmetric &matrix)=default
- MatrixSkewSymmetric & operator= (MatrixSkewSymmetric &&matrix)=default
- ~MatrixSkewSymmetric () override=default
- double operator() (const unsigned &row, const unsigned &column) const override

  Informs the value at the informed position.
- MatrixSkewSymmetric operator+ (const MatrixSkewSymmetric &matrix) const
- MatrixSquare operator+ (const MatrixSymmetry &matrix) const
- virtual void addBy (const MatrixSkewSymmetric &matrix)
- MatrixSkewSymmetric operator- (const MatrixSkewSymmetric &matrix) const
- MatrixSquare operator- (const MatrixSymmetry &matrix) const
- virtual void subtractBy (const MatrixSkewSymmetric &matrix)
- MatrixSkewSymmetric operator* (const double &scalar) const
- MatrixSquare operator* (const MatrixSkewSymmetric &matrix) const
- Vector operator* (const Vector &vector) const override

  Multiplies this matrix and the informed vector, considered as a column matrix.
- Matrix operator* (const Matrix &matrix) const override

  Multiplies this matrix and the informed matrix.
- void multiplyBy (const double &scalar) override

  Multiplies this matrix and the informed scalar, setting the result in this matrix.
- void transpose () override

  Transposes this matrix.
- void fillWithRandomValues (const double &min, const double &max) override

  Fills the array with random values.

Public Member Functions inherited from pmat::MatrixSymmetry

- MatrixSymmetry ()=default
- MatrixSymmetry (const MatrixSymmetry &matrix)
- MatrixSymmetry (MatrixSymmetry &&matrix)=default
- MatrixSymmetry (const unsigned &size)
- ~MatrixSymmetry () override=default
- MatrixSymmetry & operator= (const MatrixSymmetry &)=default
- MatrixSymmetry & operator= (MatrixSymmetry &&)=default
- unsigned length () const override

  Informs the number of elements.
- double operator() (const unsigned &row, const unsigned &column) const override=0

  Informs the value at the informed position.
- void transpose () override=0

  Transposes this matrix.
- void fillWithRandomValues (const double &min, const double &max) override=0

  Fills the array with random values.

Public Member Functions inherited from pmat::MatrixSquare

- MatrixSquare ()=default
- MatrixSquare (MatrixSquare &&matrix) noexcept=default
- MatrixSquare (Matrix &&matrix)
- MatrixSquare (const unsigned &size)
- MatrixSquare (const MatrixSquare &matrix)
- ~MatrixSquare () override=default
- MatrixSquare & operator= (const MatrixSquare &matrix)=default
- MatrixSquare & operator= (MatrixSquare &&matrix) noexcept=default
- unsigned size () const
- virtual void resize (const unsigned &size)
- MatrixSquare operator+ (const MatrixSquare &matrix) const
- MatrixSquare operator- (const MatrixSquare &matrix) const
- MatrixSquare operator∗ (const MatrixSquare &matrix) const
- MatrixSquare operator∗ (const double &scalar) const
- Vector operator∗ (const Vector &vector) const override

  Multiplies this matrix and the informed vector, considered as a column matrix.
- virtual void fillDiagonalWith (const double &value)
- virtual MatrixSquare multiplyByBiggerMatrix (const MatrixSquare &matrix, SubMatrixPos pos)

  Calculates the multiplication of this matrix and the first parameter.
- virtual double trace () const

  Calculates the trace of this matrix.
- DecompositionPLU decomposeToPLU () const

  Returns a calculator for the PLU Decomposition of this matrix.
- DecompositionSAS decomposeToSAS () const

  Returns a calculator for the SAS Decomposition of this matrix.
- DecompositionPQR decomposeToPQR () const

  Returns a calculator for the PQR Decomposition of this matrix.

Public Member Functions inherited from pmat::Matrix

- Matrix ()=default
- Matrix (const unsigned &rowSize, const unsigned &columnSize)
- Matrix (const std::string &fileName)
- Matrix (const Matrix &matrix)
- Matrix (Matrix &&matrix) noexcept
- ~Matrix () override=default
- unsigned length () const override

  Informs the number of elements.
- unsigned dimension () const override

  Informs the dimension of the array.
- void resize (const unsigned &rowSize, const unsigned &columnSize)

  Clears this matrix and sets a new size.
- void clear () override

  Removes all elements and sets size zero.
- virtual void setValue (const double &value, const unsigned &row, const unsigned &column)

  Sets the informed value at the informed position.
- virtual double operator() (const unsigned &row, const unsigned &column) const

  Informs the value at the informed position.
- unsigned rowSize () const

  Informs the size of matrix row dimension.
- unsigned columnSize () const

  Informs the size of matrix column dimension.
- Matrix & operator= (const Matrix &matrix)
- Matrix & operator= (Matrix &&matrix) noexcept
- virtual bool operator== (const Matrix &matrix) const
- virtual double dotProduct (const Matrix &matrix) const

  Calculates the dot product of this matrix with the informed matrix.
- Matrix operator+ (const Matrix &matrix) const

  Sums this matrix with the informed matrix.
- void addBy (const Matrix &matrix)

  Sums this matrix with the informed matrix, setting the result in this matrix.
- Matrix operator- (const Matrix &matrix) const

  Subtracts this matrix with the informed matrix.
- void subtractBy (const Matrix &matrix)

  Subtracts this matrix with the informed matrix, setting the result in this matrix.
- virtual Matrix operator* (const Matrix &matrix) const

  Multiplies this matrix and the informed matrix.
- virtual Vector operator* (const Vector &vector) const

  Multiplies this matrix and the informed vector, considered as a column matrix.
- Matrix operator* (const double &scalar) const

  Multiplies this matrix by the informed scalar.
- virtual void multiplyBy (const double &scalar)

  Multiplies this matrix and the informed scalar, setting the result in this matrix.
- Matrix multiply (const Matrix &matrix, unsigned nThreads)

  Multiplies this matrix by the informed scalar using multiple threads.
- Matrix multiplyHadamardBy (const Matrix &matrix) const

  Performs the Hadamard multiplication of this matrix and the informed matrix.
- virtual void multiplyRowBy (const unsigned &row, const double &scalar)

  Multiplies all the elements of the informed row by the informed scalar.
- virtual void multiplyColumnBy (const unsigned &column, const double &scalar)

Multiplies all the elements of the informed column by the informed scalar.

- virtual void swapRows (const unsigned &rowA, const unsigned &rowB, const unsigned &startColumn, const unsigned &endColumn)

  Swaps the rows at the informed positions in a range of columns.
- virtual void swapRows (const unsigned &rowA, const unsigned &rowB)

  Swaps the rows at the informed positions.
- virtual void swapColumns (const unsigned &columnA, const unsigned &columnB, const unsigned &startRow, const unsigned &endRow)

  Swaps the columns at the informed positions in a range of rows.
- virtual void swapColumns (const unsigned &columnA, const unsigned &columnB)

  Swaps the columns at the informed positions.
- virtual void transpose ()

  Transposes this matrix.
- virtual double getFrobeniusNorm () const

  Calculates the Frobenius Norm of this matrix.
- void fillWithRandomValues (const double &min, const double &max) override

  Fills the array with random values.
- Vector rowToVector (const unsigned &row) const

  Gets the informed row as a vector.
- Vector columnToVector (const unsigned &column) const

  Gets the informed column as a vector.
- unsigned occurrences (const double &value) const override

  Informes the number of occurrences of a value in the array.
- virtual unsigned occurrencesInRow (const unsigned row, const double &value) const

  Informs the number of occurrences of the informed value at the informed row.
- virtual unsigned occurrencesInColumn (const unsigned column, const double &value) const

  Informs the number of occurrences of the informed value at the informed column.

Public Member Functions inherited from pmat::Array

- Array ()=default
- Array (const Array &array)=default
- Array (Array &&)=default
- Array & operator= (const Array &)=default
- Array & operator= (Array &&)=default
- virtual ~Array ()=default
- virtual unsigned length () const =0

  Informs the number of elements.
- virtual unsigned dimension () const =0

  Informs the dimension of the array.
- virtual void clear ()=0

  Removes all elements and sets size zero.
- virtual unsigned occurrences (const double &value) const =0

  Informes the number of occurrences of a value in the array.
- virtual void fillWithRandomValues (const double &min, const double &max)=0

  Fills the array with random values.

Additional Inherited Members

Protected Member Functions inherited from pmat::MatrixSymmetry

- unsigned vectorIndex (const unsigned &i, const unsigned &j) const override

Protected Member Functions inherited from pmat::Matrix

- virtual unsigned vectorIndex (const unsigned &row, const unsigned &column) const
- double vectorElement (const unsigned &row, const unsigned &column) const
- void moveToThis (Matrix &&matrix)
- void initializeMembers (unsigned rowSize, unsigned columnSize, bool isTransposed)
- void copyMembers (const Matrix &matrix)
- bool isTransposed () const

6.8.1 Constructor & Destructor Documentation

6.8.1.1 MatrixSkewSymmetric() [1/4]

```
pmat::MatrixSkewSymmetric::MatrixSkewSymmetric ( )  [default]
```

6.8.1.2 MatrixSkewSymmetric() [2/4]

```
pmat::MatrixSkewSymmetric::MatrixSkewSymmetric (
            const unsigned & size )  [inline], [explicit]
```

6.8.1.3 MatrixSkewSymmetric() [3/4]

```
pmat::MatrixSkewSymmetric::MatrixSkewSymmetric (
            const MatrixSkewSymmetric & matrix )  [inline]
```

6.8.1.4 MatrixSkewSymmetric() [4/4]

```
pmat::MatrixSkewSymmetric::MatrixSkewSymmetric (
            MatrixSkewSymmetric && matrix )  [inline]
```

6.8.1.5 ~MatrixSkewSymmetric()

```
pmat::MatrixSkewSymmetric::~MatrixSkewSymmetric ( )  [override], [default]
```

6.8.2 Member Function Documentation

6.8.2.1 addBy()

```
void pmat::MatrixSkewSymmetric::addBy (
            const MatrixSkewSymmetric & matrix )  [virtual]
```

6.8.2.2 fillWithRandomValues()

```
void pmat::MatrixSkewSymmetric::fillWithRandomValues (
            const double & min,
            const double & max )  [override], [virtual]
```

Fills the array with random values.

Parameters

min	Minimum acceptable value
max	Maximum acceptable value

Implements pmat::MatrixSymmetry.

6.8.2.3 multiplyBy()

```
void pmat::MatrixSkewSymmetric::multiplyBy (
            const double & scalar )  [override], [virtual]
```

Multiplies this matrix and the informed scalar, setting the result in this matrix.

Parameters

scalar	Second operand of the multiplication

Reimplemented from pmat::Matrix.

6.8.2.4 operator()()

```
double pmat::MatrixSkewSymmetric::operator() (
            const unsigned & row,
            const unsigned & column ) const  [override], [virtual]
```

Informs the value at the informed position.

Parameters

row	Row position of the value to be informed
column	Column position of the value to be informed

Returns

double Value at the informed position

Implements pmat::MatrixSymmetry.

6.8.2.5 operator*() [1/4]

```
pmat::MatrixSkewSymmetric pmat::MatrixSkewSymmetric::operator* (
            const double & scalar ) const
```

6.8.2.6 operator*() [2/4]

```
pmat::Matrix pmat::MatrixSkewSymmetric::operator* (
            const Matrix & matrix ) const  [override], [virtual]
```

Multiplies this matrix and the informed matrix.

Parameters

matrix	Right operand

Returns

Matrix Multiplication result

Exceptions

std::invalid_argument	Incompatible sizes

Reimplemented from pmat::Matrix.

6.8.2.7 operator∗() [3/4]

```
pmat::MatrixSquare pmat::MatrixSkewSymmetric::operator* (
            const MatrixSkewSymmetric & matrix ) const
```

6.8.2.8 operator∗() [4/4]

```
pmat::Vector pmat::MatrixSkewSymmetric::operator* (
            const Vector & vector ) const  [override], [virtual]
```

Multiplies this matrix and the informed vector, considered as a column matrix.

Parameters

vector	Right operand

Returns

Vector Multiplication result

Exceptions

std::invalid_argument	Incompatible sizes

Reimplemented from pmat::Matrix.

6.8.2.9 operator+() [1/2]

```
pmat::MatrixSkewSymmetric pmat::MatrixSkewSymmetric::operator+ (
            const MatrixSkewSymmetric & matrix ) const
```

6.8.2.10 operator+() [2/2]

```
pmat::MatrixSquare pmat::MatrixSkewSymmetric::operator+ (
            const MatrixSymmetry & matrix ) const
```

6.8.2.11 operator-() [1/2]

```
pmat::MatrixSkewSymmetric pmat::MatrixSkewSymmetric::operator- (
            const MatrixSkewSymmetric & matrix ) const
```

6.8.2.12 operator-() [2/2]

```
pmat::MatrixSquare pmat::MatrixSkewSymmetric::operator- (
            const MatrixSymmetry & matrix ) const
```

6.8.2.13 operator=() [1/2]

```
MatrixSkewSymmetric & pmat::MatrixSkewSymmetric::operator= (
            const MatrixSkewSymmetric & matrix )  [default]
```

6.8.2.14 operator=() [2/2]

```
MatrixSkewSymmetric & pmat::MatrixSkewSymmetric::operator= (
            MatrixSkewSymmetric && matrix )  [default]
```

6.8.2.15 subtractBy()

```
void pmat::MatrixSkewSymmetric::subtractBy (
            const MatrixSkewSymmetric & matrix )  [virtual]
```

6.8.2.16 transpose()

```
void pmat::MatrixSkewSymmetric::transpose ( )  [override], [virtual]
```

Transposes this matrix.

Implements pmat::MatrixSymmetry.

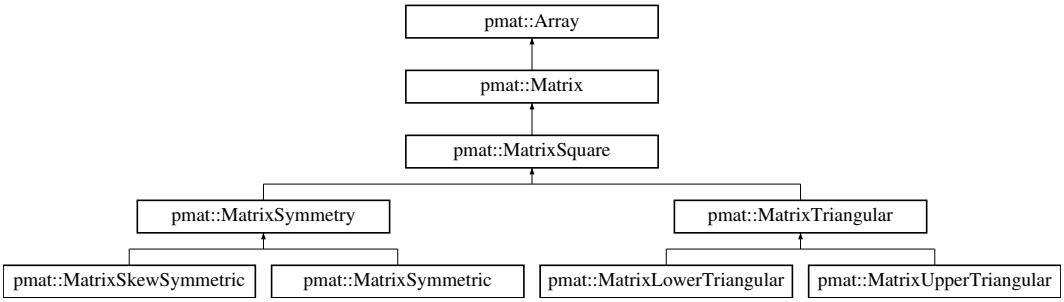
The documentation for this class was generated from the following files:

- src/MatrixSkewSymmetric.h
- src/MatrixSkewSymmetric.cpp

6.9 pmat::MatrixSquare Class Reference

`#include <MatrixSquare.h>`

Inheritance diagram for pmat::MatrixSquare:



Public Member Functions

- MatrixSquare ()=default
- MatrixSquare (MatrixSquare &&matrix) noexcept=default
- MatrixSquare (Matrix &&matrix)
- MatrixSquare (const unsigned &size)
- MatrixSquare (const MatrixSquare &matrix)
- ~MatrixSquare () override=default
- MatrixSquare & operator= (const MatrixSquare &matrix)=default
- MatrixSquare & operator= (MatrixSquare &&matrix) noexcept=default
- unsigned size () const
- virtual void resize (const unsigned &size)
- MatrixSquare operator+ (const MatrixSquare &matrix) const
- MatrixSquare operator- (const MatrixSquare &matrix) const
- MatrixSquare operator∗ (const MatrixSquare &matrix) const
- MatrixSquare operator∗ (const double &scalar) const
- Vector operator∗ (const Vector &vector) const override

  Multiplies this matrix and the informed vector, considered as a column matrix.
- virtual void fillDiagonalWith (const double &value)
- virtual MatrixSquare multiplyByBiggerMatrix (const MatrixSquare &matrix, SubMatrixPos pos)

  Calculates the multiplication of this matrix and the first parameter.
- virtual double trace () const

  Calculates the trace of this matrix.
- DecompositionPLU decomposeToPLU () const

  Returns a calculator for the PLU Decomposition of this matrix.
- DecompositionSAS decomposeToSAS () const

  Returns a calculator for the SAS Decomposition of this matrix.
- DecompositionPQR decomposeToPQR () const

  Returns a calculator for the PQR Decomposition of this matrix.

Public Member Functions inherited from pmat::Matrix

- Matrix ()=default
- Matrix (const unsigned &rowSize, const unsigned &columnSize)
- Matrix (const std::string &fileName)
- Matrix (const Matrix &matrix)
- Matrix (Matrix &&matrix) noexcept
- ~Matrix () override=default
- unsigned length () const override

  Informs the number of elements.
- unsigned dimension () const override

  Informs the dimension of the array.
- void resize (const unsigned &rowSize, const unsigned &columnSize)

  Clears this matrix and sets a new size.
- void clear () override

  Removes all elements and sets size zero.
- virtual void setValue (const double &value, const unsigned &row, const unsigned &column)

  Sets the informed value at the informed position.
- virtual double operator() (const unsigned &row, const unsigned &column) const

  Informs the value at the informed position.
- unsigned rowSize () const

  Informs the size of matrix row dimension.
- unsigned columnSize () const

  Informs the size of matrix column dimension.
- Matrix & operator= (const Matrix &matrix)
- Matrix & operator= (Matrix &&matrix) noexcept
- virtual bool operator== (const Matrix &matrix) const
- virtual double dotProduct (const Matrix &matrix) const

  Calculates the dot product of this matrix with the informed matrix.
- Matrix operator+ (const Matrix &matrix) const

  Sums this matrix with the informed matrix.
- void addBy (const Matrix &matrix)

  Sums this matrix with the informed matrix, setting the result in this matrix.
- Matrix operator- (const Matrix &matrix) const

  Subtracts this matrix with the informed matrix.
- void subtractBy (const Matrix &matrix)

  Subtracts this matrix with the informed matrix, setting the result in this matrix.
- virtual Matrix operator* (const Matrix &matrix) const

  Multiplies this matrix and the informed matrix.
- virtual Vector operator* (const Vector &vector) const

  Multiplies this matrix and the informed vector, considered as a column matrix.
- Matrix operator* (const double &scalar) const

  Multiplies this matrix by the informed scalar.
- virtual void multiplyBy (const double &scalar)

  Multiplies this matrix and the informed scalar, setting the result in this matrix.
- Matrix multiply (const Matrix &matrix, unsigned nThreads)

  Multiplies this matrix by the informed scalar using multiple threads.
- Matrix multiplyHadamardBy (const Matrix &matrix) const

  Performs the Hadamard multiplication of this matrix and the informed matrix.
- virtual void multiplyRowBy (const unsigned &row, const double &scalar)

  Multiplies all the elements of the informed row by the informed scalar.
- virtual void multiplyColumnBy (const unsigned &column, const double &scalar)

Multiplies all the elements of the informed column by the informed scalar.

- virtual void swapRows (const unsigned &rowA, const unsigned &rowB, const unsigned &startColumn, const unsigned &endColumn)

  Swaps the rows at the informed positions in a range of columns.
- virtual void swapRows (const unsigned &rowA, const unsigned &rowB)

  Swaps the rows at the informed positions.
- virtual void swapColumns (const unsigned &columnA, const unsigned &columnB, const unsigned &startRow, const unsigned &endRow)

  Swaps the columns at the informed positions in a range of rows.
- virtual void swapColumns (const unsigned &columnA, const unsigned &columnB)

  Swaps the columns at the informed positions.
- virtual void transpose ()

  Transposes this matrix.
- virtual double getFrobeniusNorm () const

  Calculates the Frobenius Norm of this matrix.
- void fillWithRandomValues (const double &min, const double &max) override

  Fills the array with random values.
- Vector rowToVector (const unsigned &row) const

  Gets the informed row as a vector.
- Vector columnToVector (const unsigned &column) const

  Gets the informed column as a vector.
- unsigned occurrences (const double &value) const override

  Informes the number of occurrences of a value in the array.
- virtual unsigned occurrencesInRow (const unsigned row, const double &value) const

  Informs the number of occurrences of the informed value at the informed row.
- virtual unsigned occurrencesInColumn (const unsigned column, const double &value) const

  Informs the number of occurrences of the informed value at the informed column.

Public Member Functions inherited from pmat::Array

- Array ()=default
- Array (const Array &array)=default
- Array (Array &&)=default
- Array & operator= (const Array &)=default
- Array & operator= (Array &&)=default
- virtual ~Array ()=default
- virtual unsigned length () const =0

  Informs the number of elements.
- virtual unsigned dimension () const =0

  Informs the dimension of the array.
- virtual void clear ()=0

  Removes all elements and sets size zero.
- virtual unsigned occurrences (const double &value) const =0

  Informes the number of occurrences of a value in the array.
- virtual void fillWithRandomValues (const double &min, const double &max)=0

  Fills the array with random values.

Additional Inherited Members

Protected Member Functions inherited from pmat::Matrix

- virtual unsigned vectorIndex (const unsigned &row, const unsigned &column) const
- double vectorElement (const unsigned &row, const unsigned &column) const
- void moveToThis (Matrix &&matrix)
- void initializeMembers (unsigned rowSize, unsigned columnSize, bool isTransposed)
- void copyMembers (const Matrix &matrix)
- bool isTransposed () const

6.9.1 Constructor & Destructor Documentation

6.9.1.1 MatrixSquare() [1/5]

```
pmat::MatrixSquare::MatrixSquare ( )  [default]
```

6.9.1.2 MatrixSquare() [2/5]

```
pmat::MatrixSquare::MatrixSquare (
            MatrixSquare && matrix )  [default], [noexcept]
```

6.9.1.3 MatrixSquare() [3/5]

```
pmat::MatrixSquare::MatrixSquare (
            Matrix && matrix )
```

6.9.1.4 MatrixSquare() [4/5]

```
pmat::MatrixSquare::MatrixSquare (
            const unsigned & size )  [inline], [explicit]
```

6.9.1.5 MatrixSquare() [5/5]

```
pmat::MatrixSquare::MatrixSquare (
            const MatrixSquare & matrix )
```

6.9.1.6 ~MatrixSquare()

```
pmat::MatrixSquare::~MatrixSquare ( )  [override], [default]
```

6.9.2 Member Function Documentation

6.9.2.1 decomposeToPLU()

```
pmat::DecompositionPLU pmat::MatrixSquare::decomposeToPLU ( ) const
```

Returns a calculator for the PLU Decomposition of this matrix.

Returns

DecompositionPLU PLU calculator

6.9.2.2 decomposeToPQR()

```
pmat::DecompositionPQR pmat::MatrixSquare::decomposeToPQR ( ) const
```

Returns a calculator for the PQR Decomposition of this matrix.

Returns

DecompositionPQR PQR calculator

6.9.2.3 decomposeToSAS()

```
pmat::DecompositionSAS pmat::MatrixSquare::decomposeToSAS ( ) const
```

Returns a calculator for the SAS Decomposition of this matrix.

Returns

DecompositionSAS SAS calculator

6.9.2.4 fillDiagonalWith()

```
void pmat::MatrixSquare::fillDiagonalWith (
            const double & value )  [virtual]
```

6.9.2.5 multiplyByBiggerMatrix()

```
pmat::MatrixSquare pmat::MatrixSquare::multiplyByBiggerMatrix (
            const MatrixSquare & matrix,
            SubMatrixPos pos )  [virtual]
```

Calculates the multiplication of this matrix and the first parameter.

Given matrices A with size n and B with size $m > n$, this function performs $A'.B$ of size m where $A' = [A \quad 0; \quad 0 \quad 1]$ or $A' = [1 \quad 0; \quad 0 \quad A]$.

Parameters

matrix	The right operand, which must not be smaller
pos	Position of this matrix on the matrix A'

Returns

The product of A' and the parameter

Exceptions

invalid_argument	Parameters are not compatible

6.9.2.6 operator∗() [1/3]

```
pmat::MatrixSquare pmat::MatrixSquare::operator* (
            const double & scalar ) const
```

6.9.2.7 operator∗() [2/3]

```
pmat::MatrixSquare pmat::MatrixSquare::operator* (
            const MatrixSquare & matrix ) const
```

6.9.2.8 operator∗() [3/3]

```
pmat::Vector pmat::MatrixSquare::operator* (
            const Vector & vector ) const  [override], [virtual]
```

Multiplies this matrix and the informed vector, considered as a column matrix.

Parameters

vector	Right operand

Returns

Vector Multiplication result

Exceptions

std::invalid_argument	Incompatible sizes

Reimplemented from pmat::Matrix.

Reimplemented in pmat::MatrixSymmetric, and pmat::MatrixUpperTriangular.

6.9.2.9 operator+()

```
pmat::MatrixSquare pmat::MatrixSquare::operator+ (
            const MatrixSquare & matrix ) const
```

6.9.2.10 operator-()

```
pmat::MatrixSquare pmat::MatrixSquare::operator- (
            const MatrixSquare & matrix ) const
```

6.9.2.11 operator=() [1/2]

```
MatrixSquare & pmat::MatrixSquare::operator= (
            const MatrixSquare & matrix )  [default]
```

6.9.2.12 operator=() [2/2]

```
MatrixSquare & pmat::MatrixSquare::operator= (
            MatrixSquare && matrix )  [default], [noexcept]
```

6.9.2.13 resize()

```
void pmat::MatrixSquare::resize (
            const unsigned & size )  [virtual]
```

6.9.2.14 size()

```
unsigned pmat::MatrixSquare::size ( ) const
```

6.9.2.15 trace()

```
double pmat::MatrixSquare::trace ( ) const  [virtual]
```

Calculates the trace of this matrix.

Returns

double Trace of this matrix

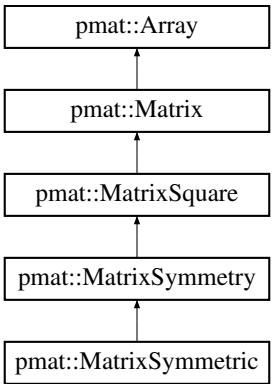
The documentation for this class was generated from the following files:

- src/MatrixSquare.h
- src/MatrixSquare.cpp

6.10 pmat::MatrixSymmetric Class Reference

`#include <MatrixSymmetric.h>`

Inheritance diagram for pmat::MatrixSymmetric:



Public Member Functions

- MatrixSymmetric ()=default
- MatrixSymmetric (const unsigned &size)
- MatrixSymmetric (const MatrixSymmetric &matrix)=default
- MatrixSymmetric (MatrixSymmetric &&matrix)
- MatrixSymmetric & operator= (const MatrixSymmetric &matrix)=default
- MatrixSymmetric & operator= (MatrixSymmetric &&matrix)=default
- ~MatrixSymmetric () override=default
- double operator() (const unsigned &row, const unsigned &column) const override

  Informs the value at the informed position.
- MatrixSymmetric operator+ (const MatrixSymmetric &matrix) const
- MatrixSquare operator+ (const MatrixSymmetry &matrix) const
- virtual void addBy (const MatrixSymmetric &matrix)
- MatrixSymmetric operator- (const MatrixSymmetric &matrix) const
- MatrixSquare operator- (const MatrixSymmetry &matrix) const
- virtual void subtractBy (const MatrixSymmetric &matrix)
- MatrixSymmetric operator* (const double &scalar) const
- MatrixSquare operator* (const MatrixSymmetric &matrix) const
- Matrix operator* (const Matrix &matrix) const override

  Multiplies this matrix and the informed matrix.
- Vector operator* (const Vector &vector) const override

  Multiplies this matrix and the informed vector, considered as a column matrix.
- void multiplyBy (const double &scalar) override

  Multiplies this matrix and the informed scalar, setting the result in this matrix.
- void transpose () override

  Transposes this matrix.
- void fillWithRandomValues (const double &min, const double &max) override

  Fills the array with random values.
- DecompositionCholesky decomposeToCholesky ()

  Returns a calculator for the Cholesky Decomposition of this matrix.

Public Member Functions inherited from pmat::MatrixSymmetry

- MatrixSymmetry ()=default
- MatrixSymmetry (const MatrixSymmetry &matrix)
- MatrixSymmetry (MatrixSymmetry &&matrix)=default
- MatrixSymmetry (const unsigned &size)
- ~MatrixSymmetry () override=default
- MatrixSymmetry & operator= (const MatrixSymmetry &)=default
- MatrixSymmetry & operator= (MatrixSymmetry &&)=default
- unsigned length () const override

  Informs the number of elements.
- double operator() (const unsigned &row, const unsigned &column) const override=0

  Informs the value at the informed position.
- void transpose () override=0

  Transposes this matrix.
- void fillWithRandomValues (const double &min, const double &max) override=0

  Fills the array with random values.

Public Member Functions inherited from pmat::MatrixSquare

- MatrixSquare ()=default
- MatrixSquare (MatrixSquare &&matrix) noexcept=default
- MatrixSquare (Matrix &&matrix)
- MatrixSquare (const unsigned &size)
- MatrixSquare (const MatrixSquare &matrix)
- ~MatrixSquare () override=default
- MatrixSquare & operator= (const MatrixSquare &matrix)=default
- MatrixSquare & operator= (MatrixSquare &&matrix) noexcept=default
- unsigned size () const
- virtual void resize (const unsigned &size)
- MatrixSquare operator+ (const MatrixSquare &matrix) const
- MatrixSquare operator- (const MatrixSquare &matrix) const
- MatrixSquare operator∗ (const MatrixSquare &matrix) const
- MatrixSquare operator∗ (const double &scalar) const
- Vector operator∗ (const Vector &vector) const override

  Multiplies this matrix and the informed vector, considered as a column matrix.
- virtual void fillDiagonalWith (const double &value)
- virtual MatrixSquare multiplyByBiggerMatrix (const MatrixSquare &matrix, SubMatrixPos pos)

  Calculates the multiplication of this matrix and the first parameter.
- virtual double trace () const

  Calculates the trace of this matrix.
- DecompositionPLU decomposeToPLU () const

  Returns a calculator for the PLU Decomposition of this matrix.
- DecompositionSAS decomposeToSAS () const

  Returns a calculator for the SAS Decomposition of this matrix.
- DecompositionPQR decomposeToPQR () const

  Returns a calculator for the PQR Decomposition of this matrix.

Public Member Functions inherited from pmat::Matrix

- Matrix ()=default
- Matrix (const unsigned &rowSize, const unsigned &columnSize)
- Matrix (const std::string &fileName)
- Matrix (const Matrix &matrix)
- Matrix (Matrix &&matrix) noexcept
- ~Matrix () override=default
- unsigned length () const override

  Informs the number of elements.
- unsigned dimension () const override

  Informs the dimension of the array.
- void resize (const unsigned &rowSize, const unsigned &columnSize)

  Clears this matrix and sets a new size.
- void clear () override

  Removes all elements and sets size zero.
- virtual void setValue (const double &value, const unsigned &row, const unsigned &column)

  Sets the informed value at the informed position.
- virtual double operator() (const unsigned &row, const unsigned &column) const

  Informs the value at the informed position.
- unsigned rowSize () const

  Informs the size of matrix row dimension.
- unsigned columnSize () const

  Informs the size of matrix column dimension.
- Matrix & operator= (const Matrix &matrix)
- Matrix & operator= (Matrix &&matrix) noexcept
- virtual bool operator== (const Matrix &matrix) const
- virtual double dotProduct (const Matrix &matrix) const

  Calculates the dot product of this matrix with the informed matrix.
- Matrix operator+ (const Matrix &matrix) const

  Sums this matrix with the informed matrix.
- void addBy (const Matrix &matrix)

  Sums this matrix with the informed matrix, setting the result in this matrix.
- Matrix operator- (const Matrix &matrix) const

  Subtracts this matrix with the informed matrix.
- void subtractBy (const Matrix &matrix)

  Subtracts this matrix with the informed matrix, setting the result in this matrix.
- virtual Matrix operator* (const Matrix &matrix) const

  Multiplies this matrix and the informed matrix.
- virtual Vector operator* (const Vector &vector) const

  Multiplies this matrix and the informed vector, considered as a column matrix.
- Matrix operator* (const double &scalar) const

  Multiplies this matrix by the informed scalar.
- virtual void multiplyBy (const double &scalar)

  Multiplies this matrix and the informed scalar, setting the result in this matrix.
- Matrix multiply (const Matrix &matrix, unsigned nThreads)

  Multiplies this matrix by the informed scalar using multiple threads.
- Matrix multiplyHadamardBy (const Matrix &matrix) const

  Performs the Hadamard multiplication of this matrix and the informed matrix.
- virtual void multiplyRowBy (const unsigned &row, const double &scalar)

  Multiplies all the elements of the informed row by the informed scalar.
- virtual void multiplyColumnBy (const unsigned &column, const double &scalar)

Multiplies all the elements of the informed column by the informed scalar.

- virtual void swapRows (const unsigned &rowA, const unsigned &rowB, const unsigned &startColumn, const unsigned &endColumn)

  Swaps the rows at the informed positions in a range of columns.
- virtual void swapRows (const unsigned &rowA, const unsigned &rowB)

  Swaps the rows at the informed positions.
- virtual void swapColumns (const unsigned &columnA, const unsigned &columnB, const unsigned &startRow, const unsigned &endRow)

  Swaps the columns at the informed positions in a range of rows.
- virtual void swapColumns (const unsigned &columnA, const unsigned &columnB)

  Swaps the columns at the informed positions.
- virtual void transpose ()

  Transposes this matrix.
- virtual double getFrobeniusNorm () const

  Calculates the Frobenius Norm of this matrix.
- void fillWithRandomValues (const double &min, const double &max) override

  Fills the array with random values.
- Vector rowToVector (const unsigned &row) const

  Gets the informed row as a vector.
- Vector columnToVector (const unsigned &column) const

  Gets the informed column as a vector.
- unsigned occurrences (const double &value) const override

  Informes the number of occurrences of a value in the array.
- virtual unsigned occurrencesInRow (const unsigned row, const double &value) const

  Informs the number of occurrences of the informed value at the informed row.
- virtual unsigned occurrencesInColumn (const unsigned column, const double &value) const

  Informs the number of occurrences of the informed value at the informed column.

Public Member Functions inherited from pmat::Array

- Array ()=default
- Array (const Array &array)=default
- Array (Array &&)=default
- Array & operator= (const Array &)=default
- Array & operator= (Array &&)=default
- virtual ~Array ()=default
- virtual unsigned length () const =0

  Informs the number of elements.
- virtual unsigned dimension () const =0

  Informs the dimension of the array.
- virtual void clear ()=0

  Removes all elements and sets size zero.
- virtual unsigned occurrences (const double &value) const =0

  Informes the number of occurrences of a value in the array.
- virtual void fillWithRandomValues (const double &min, const double &max)=0

  Fills the array with random values.

Additional Inherited Members

Protected Member Functions inherited from pmat::MatrixSymmetry

- unsigned vectorIndex (const unsigned &i, const unsigned &j) const override

Protected Member Functions inherited from pmat::Matrix

- virtual unsigned vectorIndex (const unsigned &row, const unsigned &column) const
- double vectorElement (const unsigned &row, const unsigned &column) const
- void moveToThis (Matrix &&matrix)
- void initializeMembers (unsigned rowSize, unsigned columnSize, bool isTransposed)
- void copyMembers (const Matrix &matrix)
- bool isTransposed () const

6.10.1 Constructor & Destructor Documentation

6.10.1.1 MatrixSymmetric() [1/4]

```
pmat::MatrixSymmetric::MatrixSymmetric ( )  [default]
```

6.10.1.2 MatrixSymmetric() [2/4]

```
pmat::MatrixSymmetric::MatrixSymmetric (
            const unsigned & size )  [inline], [explicit]
```

6.10.1.3 MatrixSymmetric() [3/4]

```
pmat::MatrixSymmetric::MatrixSymmetric (
            const MatrixSymmetric & matrix )  [default]
```

6.10.1.4 MatrixSymmetric() [4/4]

```
pmat::MatrixSymmetric::MatrixSymmetric (
            MatrixSymmetric && matrix )  [inline]
```

6.10.1.5 ~MatrixSymmetric()

```
pmat::MatrixSymmetric::~MatrixSymmetric ( )  [override], [default]
```

6.10.2 Member Function Documentation

6.10.2.1 addBy()

```
void pmat::MatrixSymmetric::addBy (
            const MatrixSymmetric & matrix )  [virtual]
```

6.10.2.2 decomposeToCholesky()

```
pmat::DecompositionCholesky pmat::MatrixSymmetric::decomposeToCholesky ( )
```

Returns a calculator for the Cholesky Decomposition of this matrix.

Returns

DecompositionCholesky Cholesky calculator

6.10.2.3 fillWithRandomValues()

```
void pmat::MatrixSymmetric::fillWithRandomValues (
            const double & min,
            const double & max )  [override], [virtual]
```

Fills the array with random values.

Parameters

min	Minimum acceptable value
max	Maximum acceptable value

Implements pmat::MatrixSymmetry.

6.10.2.4 multiplyBy()

```
void pmat::MatrixSymmetric::multiplyBy (
            const double & scalar )  [override], [virtual]
```

Multiplies this matrix and the informed scalar, setting the result in this matrix.

Parameters

scalar	Second operand of the multiplication

Reimplemented from pmat::Matrix.

6.10.2.5 operator()()

```
double pmat::MatrixSymmetric::operator() (
            const unsigned & row,
            const unsigned & column ) const  [override], [virtual]
```

Informs the value at the informed position.

Parameters

row	Row position of the value to be informed
column	Column position of the value to be informed

Returns

double Value at the informed position

Implements pmat::MatrixSymmetry.

6.10.2.6 operator*() [1/4]

```
pmat::MatrixSymmetric pmat::MatrixSymmetric::operator* (
            const double & scalar ) const
```

6.10.2.7 operator*() [2/4]

```
pmat::Matrix pmat::MatrixSymmetric::operator* (
            const Matrix & matrix ) const  [override], [virtual]
```

Multiplies this matrix and the informed matrix.

Parameters

matrix	Right operand

Returns

Matrix Multiplication result

Exceptions

std::invalid_argument	Incompatible sizes

Reimplemented from pmat::Matrix.

6.10.2.8 operator*() [3/4]

```
pmat::MatrixSquare pmat::MatrixSymmetric::operator* (
            const MatrixSymmetric & matrix ) const
```

6.10.2.9 operator*() [4/4]

```
pmat::Vector pmat::MatrixSymmetric::operator* (
            const Vector & vector ) const  [override], [virtual]
```

Multiplies this matrix and the informed vector, considered as a column matrix.

Parameters

vector	Right operand

Returns

Vector Multiplication result

Exceptions

std::invalid_argument	Incompatible sizes

Reimplemented from pmat::MatrixSquare.

6.10.2.10 operator+() [1/2]

```
pmat::MatrixSymmetric pmat::MatrixSymmetric::operator+ (
            const MatrixSymmetric & matrix ) const
```

6.10.2.11 operator+() [2/2]

```
pmat::MatrixSquare pmat::MatrixSymmetric::operator+ (
            const MatrixSymmetry & matrix ) const
```

6.10.2.12 operator-() [1/2]

```
pmat::MatrixSymmetric pmat::MatrixSymmetric::operator- (
            const MatrixSymmetric & matrix ) const
```

6.10.2.13 operator-() [2/2]

```
pmat::MatrixSquare pmat::MatrixSymmetric::operator- (
            const MatrixSymmetry & matrix ) const
```

6.10.2.14 operator=() [1/2]

```
MatrixSymmetric & pmat::MatrixSymmetric::operator= (
            const MatrixSymmetric & matrix )  [default]
```

6.10.2.15 operator=() [2/2]

```
MatrixSymmetric & pmat::MatrixSymmetric::operator= (
            MatrixSymmetric && matrix )  [default]
```

6.10.2.16 subtractBy()

```
void pmat::MatrixSymmetric::subtractBy (
            const MatrixSymmetric & matrix )  [virtual]
```

6.10.2.17 transpose()

```
void pmat::MatrixSymmetric::transpose ( )  [inline], [override], [virtual]
```

Transposes this matrix.

Implements pmat::MatrixSymmetry.

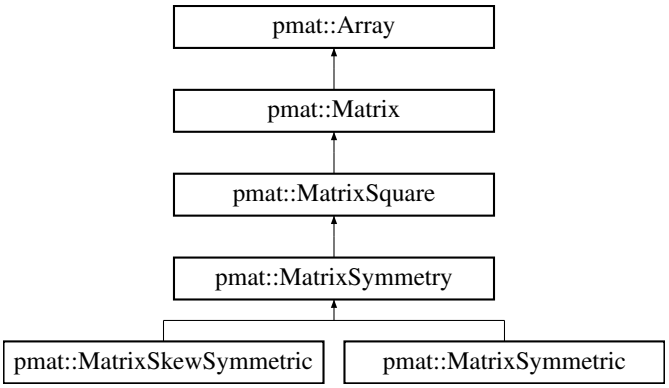
The documentation for this class was generated from the following files:

- src/MatrixSymmetric.h
- src/MatrixSymmetric.cpp

6.11 pmat::MatrixSymmetry Class Reference

```
#include <MatrixSymmetry.h>
```

Inheritance diagram for pmat::MatrixSymmetry:



Public Member Functions

- MatrixSymmetry ()=default
- MatrixSymmetry (const MatrixSymmetry &matrix)
- MatrixSymmetry (MatrixSymmetry &&matrix)=default
- MatrixSymmetry (const unsigned &size)
- ~MatrixSymmetry () override=default
- MatrixSymmetry & operator= (const MatrixSymmetry &)=default
- MatrixSymmetry & operator= (MatrixSymmetry &&)=default
- unsigned length () const override

  Informs the number of elements.
- double operator() (const unsigned &row, const unsigned &column) const override=0

  Informs the value at the informed position.
- void transpose () override=0

  Transposes this matrix.
- void fillWithRandomValues (const double &min, const double &max) override=0

  Fills the array with random values.

Public Member Functions inherited from pmat::MatrixSquare

- MatrixSquare ()=default
- MatrixSquare (MatrixSquare &&matrix) noexcept=default
- MatrixSquare (Matrix &&matrix)
- MatrixSquare (const unsigned &size)
- MatrixSquare (const MatrixSquare &matrix)
- ~MatrixSquare () override=default
- MatrixSquare & operator= (const MatrixSquare &matrix)=default
- MatrixSquare & operator= (MatrixSquare &&matrix) noexcept=default
- unsigned size () const
- virtual void resize (const unsigned &size)
- MatrixSquare operator+ (const MatrixSquare &matrix) const
- MatrixSquare operator- (const MatrixSquare &matrix) const
- MatrixSquare operator* (const MatrixSquare &matrix) const
- MatrixSquare operator* (const double &scalar) const
- Vector operator* (const Vector &vector) const override

  Multiplies this matrix and the informed vector, considered as a column matrix.
- virtual void fillDiagonalWith (const double &value)
- virtual MatrixSquare multiplyByBiggerMatrix (const MatrixSquare &matrix, SubMatrixPos pos)

  Calculates the multiplication of this matrix and the first parameter.
- virtual double trace () const

  Calculates the trace of this matrix.
- DecompositionPLU decomposeToPLU () const

  Returns a calculator for the PLU Decomposition of this matrix.
- DecompositionSAS decomposeToSAS () const

  Returns a calculator for the SAS Decomposition of this matrix.
- DecompositionPQR decomposeToPQR () const

  Returns a calculator for the PQR Decomposition of this matrix.

Public Member Functions inherited from pmat::Matrix

- Matrix ()=default
- Matrix (const unsigned &rowSize, const unsigned &columnSize)
- Matrix (const std::string &fileName)
- Matrix (const Matrix &matrix)
- Matrix (Matrix &&matrix) noexcept
- ~Matrix () override=default
- unsigned length () const override

  Informs the number of elements.
- unsigned dimension () const override

  Informs the dimension of the array.
- void resize (const unsigned &rowSize, const unsigned &columnSize)

  Clears this matrix and sets a new size.
- void clear () override

  Removes all elements and sets size zero.
- virtual void setValue (const double &value, const unsigned &row, const unsigned &column)

  Sets the informed value at the informed position.
- virtual double operator() (const unsigned &row, const unsigned &column) const

  Informs the value at the informed position.
- unsigned rowSize () const

  Informs the size of matrix row dimension.

- unsigned columnSize () const

  Informs the size of matrix column dimension.
- Matrix & operator= (const Matrix &matrix)
- Matrix & operator= (Matrix &&matrix) noexcept
- virtual bool operator== (const Matrix &matrix) const
- virtual double dotProduct (const Matrix &matrix) const

  Calculates the dot product of this matrix with the informed matrix.
- Matrix operator+ (const Matrix &matrix) const

  Sums this matrix with the informed matrix.
- void addBy (const Matrix &matrix)

  Sums this matrix with the informed matrix, setting the result in this matrix.
- Matrix operator- (const Matrix &matrix) const

  Subtracts this matrix with the informed matrix.
- void subtractBy (const Matrix &matrix)

  Subtracts this matrix with the informed matrix, setting the result in this matrix.
- virtual Matrix operator∗ (const Matrix &matrix) const

  Multiplies this matrix and the informed matrix.
- virtual Vector operator∗ (const Vector &vector) const

  Multiplies this matrix and the informed vector, considered as a column matrix.
- Matrix operator∗ (const double &scalar) const

  Multiplies this matrix by the informed scalar.
- virtual void multiplyBy (const double &scalar)

  Multiplies this matrix and the informed scalar, setting the result in this matrix.
- Matrix multiply (const Matrix &matrix, unsigned nThreads)

  Multiplies this matrix by the informed scalar using multiple threads.
- Matrix multiplyHadamardBy (const Matrix &matrix) const

  Performs the Hadamard multiplication of this matrix and the informed matrix.
- virtual void multiplyRowBy (const unsigned &row, const double &scalar)

  Multiplies all the elements of the informed row by the informed scalar.
- virtual void multiplyColumnBy (const unsigned &column, const double &scalar)

  Multiplies all the elements of the informed column by the informed scalar.
- virtual void swapRows (const unsigned &rowA, const unsigned &rowB, const unsigned &startColumn, const unsigned &endColumn)

  Swaps the rows at the informed positions in a range of columns.
- virtual void swapRows (const unsigned &rowA, const unsigned &rowB)

  Swaps the rows at the informed positions.
- virtual void swapColumns (const unsigned &columnA, const unsigned &columnB, const unsigned &startRow, const unsigned &endRow)

  Swaps the columns at the informed positions in a range of rows.
- virtual void swapColumns (const unsigned &columnA, const unsigned &columnB)

  Swaps the columns at the informed positions.
- virtual void transpose ()

  Transposes this matrix.
- virtual double getFrobeniusNorm () const

  Calculates the Frobenius Norm of this matrix.
- void fillWithRandomValues (const double &min, const double &max) override

  Fills the array with random values.
- Vector rowToVector (const unsigned &row) const

  Gets the informed row as a vector.
- Vector columnToVector (const unsigned &column) const

  Gets the informed column as a vector.

- unsigned occurrences (const double &value) const override

  Informes the number of occurrences of a value in the array.
- virtual unsigned occurrencesInRow (const unsigned row, const double &value) const

  Informs the number of occurrences of the informed value at the informed row.
- virtual unsigned occurrencesInColumn (const unsigned column, const double &value) const

  Informs the number of occurrences of the informed value at the informed column.

Public Member Functions inherited from pmat::Array

- Array ()=default
- Array (const Array &array)=default
- Array (Array &&)=default
- Array & operator= (const Array &)=default
- Array & operator= (Array &&)=default
- virtual ~Array ()=default
- virtual unsigned length () const =0

  Informs the number of elements.
- virtual unsigned dimension () const =0

  Informs the dimension of the array.
- virtual void clear ()=0

  Removes all elements and sets size zero.
- virtual unsigned occurrences (const double &value) const =0

  Informes the number of occurrences of a value in the array.
- virtual void fillWithRandomValues (const double &min, const double &max)=0

  Fills the array with random values.

Protected Member Functions

- unsigned vectorIndex (const unsigned &i, const unsigned &j) const override

Protected Member Functions inherited from pmat::Matrix

- virtual unsigned vectorIndex (const unsigned &row, const unsigned &column) const
- double vectorElement (const unsigned &row, const unsigned &column) const
- void moveToThis (Matrix &&matrix)
- void initializeMembers (unsigned rowSize, unsigned columnSize, bool isTransposed)
- void copyMembers (const Matrix &matrix)
- bool isTransposed () const

6.11.1 Constructor & Destructor Documentation

6.11.1.1 MatrixSymmetry() [1/4]

```
pmat::MatrixSymmetry::MatrixSymmetry ( )  [default]
```

6.11.1.2 MatrixSymmetry() [2/4]

```
pmat::MatrixSymmetry::MatrixSymmetry (
            const MatrixSymmetry & matrix )
```

6.11.1.3 MatrixSymmetry() [3/4]

```
pmat::MatrixSymmetry::MatrixSymmetry (
            MatrixSymmetry && matrix )  [default]
```

6.11.1.4 MatrixSymmetry() [4/4]

```
pmat::MatrixSymmetry::MatrixSymmetry (
            const unsigned & size )  [inline], [explicit]
```

6.11.1.5 ~MatrixSymmetry()

```
pmat::MatrixSymmetry::~MatrixSymmetry ( )  [override], [default]
```

6.11.2 Member Function Documentation

6.11.2.1 fillWithRandomValues()

```
void pmat::MatrixSymmetry::fillWithRandomValues (
            const double & min,
            const double & max )  [override], [pure virtual]
```

Fills the array with random values.

Parameters

min	Minimum acceptable value
max	Maximum acceptable value

Reimplemented from pmat::Matrix.

Implemented in pmat::MatrixSkewSymmetric, and pmat::MatrixSymmetric.

6.11.2.2 length()

```
unsigned pmat::MatrixSymmetry::length ( ) const  [override], [virtual]
```

Informs the number of elements.

Returns

unsigned Number of elements

Reimplemented from pmat::Matrix.

6.11.2.3 operator()()

```
double pmat::MatrixSymmetry::operator() (
            const unsigned & row,
            const unsigned & column ) const  [override], [pure virtual]
```

Informs the value at the informed position.

Parameters

row	Row position of the value to be informed
column	Column position of the value to be informed

Returns

double Value at the informed position

Reimplemented from pmat::Matrix.

Implemented in pmat::MatrixSkewSymmetric, and pmat::MatrixSymmetric.

6.11.2.4 operator=() [1/2]

```
MatrixSymmetry & pmat::MatrixSymmetry::operator= (
            const MatrixSymmetry &  )  [default]
```

6.11.2.5 operator=() [2/2]

```
MatrixSymmetry & pmat::MatrixSymmetry::operator= (
            MatrixSymmetry &&  )  [default]
```

6.11.2.6 transpose()

```
void pmat::MatrixSymmetry::transpose ( )  [override], [pure virtual]
```

Transposes this matrix.

Reimplemented from pmat::Matrix.

Implemented in pmat::MatrixSkewSymmetric, and pmat::MatrixSymmetric.

6.11.2.7 vectorIndex()

```
unsigned pmat::MatrixSymmetry::vectorIndex (
            const unsigned & i,
            const unsigned & j ) const  [override], [protected], [virtual]
```

Reimplemented from pmat::Matrix.

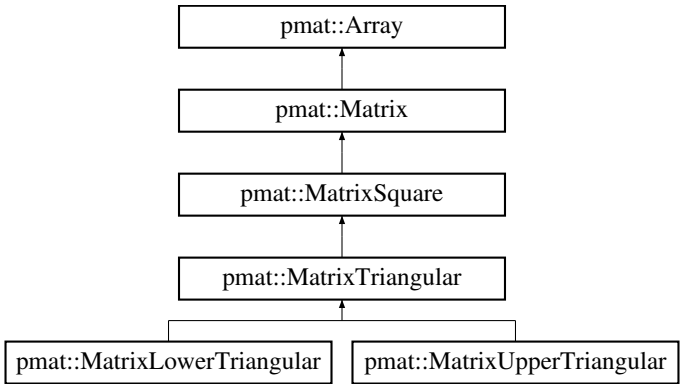
The documentation for this class was generated from the following files:

- src/MatrixSymmetry.h
- src/MatrixSymmetry.cpp

6.12 pmat::MatrixTriangular Class Reference

`#include <MatrixTriangular.h>`

Inheritance diagram for pmat::MatrixTriangular:



Public Member Functions

- MatrixTriangular ()=default
- MatrixTriangular (const MatrixTriangular &matrix)
- MatrixTriangular (MatrixTriangular &&matrix)=default
- MatrixTriangular (const unsigned &size)
- ~MatrixTriangular () override=default
- MatrixTriangular & operator= (const MatrixTriangular &)=default
- MatrixTriangular & operator= (MatrixTriangular &&)=default
- unsigned length () const override

  Informs the number of elements.
- double operator() (const unsigned &row, const unsigned &column) const override=0

  Informs the value at the informed position.
- double dotProduct (const Matrix &matrix) const override=0

  Calculates the dot product of this matrix with the informed matrix.
- MatrixSquare operator* (const MatrixTriangular &matrix) const
- MatrixSquare getSwappedByRows (const unsigned &rowIndexA, const unsigned &rowIndexB) const
- MatrixSquare getSwappedByColumns (const unsigned &columnIndexA, const unsigned &columnIndexB) const
- void fillWithRandomValues (const double &min, const double &max) override=0

  Fills the array with random values.
- void swapRows (const unsigned &rowA, const unsigned &rowB, const unsigned &startColumn, const unsigned &endColumn) override=0

  Swaps the rows at the informed positions in a range of columns.
- void swapColumns (const unsigned &columnA, const unsigned &columnB, const unsigned &startRow, const unsigned &endRow) override=0

  Swaps the columns at the informed positions in a range of rows.
- virtual TriangType type () const =0
- double determinant ()

  Calculates the determinant of this matrix through its diagonal.
- virtual bool isInvertible ()

  Informs if this matrix is invertible by inspecting its diagonal.
- Vector linearSolve (const Vector &rhs)

  Calculates the solution of a linear system by back substitution.

Public Member Functions inherited from pmat::MatrixSquare

- MatrixSquare ()=default
- MatrixSquare (MatrixSquare &&matrix) noexcept=default
- MatrixSquare (Matrix &&matrix)
- MatrixSquare (const unsigned &size)
- MatrixSquare (const MatrixSquare &matrix)
- ~MatrixSquare () override=default
- MatrixSquare & operator= (const MatrixSquare &matrix)=default
- MatrixSquare & operator= (MatrixSquare &&matrix) noexcept=default
- unsigned size () const
- virtual void resize (const unsigned &size)
- MatrixSquare operator+ (const MatrixSquare &matrix) const
- MatrixSquare operator- (const MatrixSquare &matrix) const
- MatrixSquare operator* (const MatrixSquare &matrix) const
- MatrixSquare operator* (const double &scalar) const
- Vector operator* (const Vector &vector) const override

  Multiplies this matrix and the informed vector, considered as a column matrix.
- virtual void fillDiagonalWith (const double &value)
- virtual MatrixSquare multiplyByBiggerMatrix (const MatrixSquare &matrix, SubMatrixPos pos)

  Calculates the multiplication of this matrix and the first parameter.
- virtual double trace () const

  Calculates the trace of this matrix.
- DecompositionPLU decomposeToPLU () const

  Returns a calculator for the PLU Decomposition of this matrix.
- DecompositionSAS decomposeToSAS () const

  Returns a calculator for the SAS Decomposition of this matrix.
- DecompositionPQR decomposeToPQR () const

  Returns a calculator for the PQR Decomposition of this matrix.

Public Member Functions inherited from pmat::Matrix

- Matrix ()=default
- Matrix (const unsigned &rowSize, const unsigned &columnSize)
- Matrix (const std::string &fileName)
- Matrix (const Matrix &matrix)
- Matrix (Matrix &&matrix) noexcept
- ~Matrix () override=default
- unsigned length () const override

  Informs the number of elements.
- unsigned dimension () const override

  Informs the dimension of the array.
- void resize (const unsigned &rowSize, const unsigned &columnSize)

  Clears this matrix and sets a new size.
- void clear () override

  Removes all elements and sets size zero.
- virtual void setValue (const double &value, const unsigned &row, const unsigned &column)

  Sets the informed value at the informed position.
- virtual double operator() (const unsigned &row, const unsigned &column) const

  Informs the value at the informed position.
- unsigned rowSize () const

  Informs the size of matrix row dimension.

- unsigned columnSize () const

  Informs the size of matrix column dimension.
- Matrix & operator= (const Matrix &matrix)
- Matrix & operator= (Matrix &&matrix) noexcept
- virtual bool operator== (const Matrix &matrix) const
- virtual double dotProduct (const Matrix &matrix) const

  Calculates the dot product of this matrix with the informed matrix.
- Matrix operator+ (const Matrix &matrix) const

  Sums this matrix with the informed matrix.
- void addBy (const Matrix &matrix)

  Sums this matrix with the informed matrix, setting the result in this matrix.
- Matrix operator- (const Matrix &matrix) const

  Subtracts this matrix with the informed matrix.
- void subtractBy (const Matrix &matrix)

  Subtracts this matrix with the informed matrix, setting the result in this matrix.
- virtual Matrix operator∗ (const Matrix &matrix) const

  Multiplies this matrix and the informed matrix.
- virtual Vector operator∗ (const Vector &vector) const

  Multiplies this matrix and the informed vector, considered as a column matrix.
- Matrix operator∗ (const double &scalar) const

  Multiplies this matrix by the informed scalar.
- virtual void multiplyBy (const double &scalar)

  Multiplies this matrix and the informed scalar, setting the result in this matrix.
- Matrix multiply (const Matrix &matrix, unsigned nThreads)

  Multiplies this matrix by the informed scalar using multiple threads.
- Matrix multiplyHadamardBy (const Matrix &matrix) const

  Performs the Hadamard multiplication of this matrix and the informed matrix.
- virtual void multiplyRowBy (const unsigned &row, const double &scalar)

  Multiplies all the elements of the informed row by the informed scalar.
- virtual void multiplyColumnBy (const unsigned &column, const double &scalar)

  Multiplies all the elements of the informed column by the informed scalar.
- virtual void swapRows (const unsigned &rowA, const unsigned &rowB, const unsigned &startColumn, const unsigned &endColumn)

  Swaps the rows at the informed positions in a range of columns.
- virtual void swapRows (const unsigned &rowA, const unsigned &rowB)

  Swaps the rows at the informed positions.
- virtual void swapColumns (const unsigned &columnA, const unsigned &columnB, const unsigned &startRow, const unsigned &endRow)

  Swaps the columns at the informed positions in a range of rows.
- virtual void swapColumns (const unsigned &columnA, const unsigned &columnB)

  Swaps the columns at the informed positions.
- virtual void transpose ()

  Transposes this matrix.
- virtual double getFrobeniusNorm () const

  Calculates the Frobenius Norm of this matrix.
- void fillWithRandomValues (const double &min, const double &max) override

  Fills the array with random values.
- Vector rowToVector (const unsigned &row) const

  Gets the informed row as a vector.
- Vector columnToVector (const unsigned &column) const

  Gets the informed column as a vector.

- unsigned occurrences (const double &value) const override

  Informes the number of occurrences of a value in the array.
- virtual unsigned occurrencesInRow (const unsigned row, const double &value) const

  Informs the number of occurrences of the informed value at the informed row.
- virtual unsigned occurrencesInColumn (const unsigned column, const double &value) const

  Informs the number of occurrences of the informed value at the informed column.

Public Member Functions inherited from pmat::Array

- Array ()=default
- Array (const Array &array)=default
- Array (Array &&)=default
- Array & operator= (const Array &)=default
- Array & operator= (Array &&)=default
- virtual ~Array ()=default
- virtual unsigned length () const =0

  Informs the number of elements.
- virtual unsigned dimension () const =0

  Informs the dimension of the array.
- virtual void clear ()=0

  Removes all elements and sets size zero.
- virtual unsigned occurrences (const double &value) const =0

  Informes the number of occurrences of a value in the array.
- virtual void fillWithRandomValues (const double &min, const double &max)=0

  Fills the array with random values.

Static Public Member Functions

- static void findInverseByBackSubstitution (const MatrixTriangular &matrix, MatrixTriangular &resp)

  Independent function for finding inverse by back substitution.
- static pmat::Vector findSolutionByBackSubstitution (const MatrixTriangular &matrix, const Vector &rhs)

  Independent function for find the solution of a linear system by back substitution.

Protected Member Functions

- unsigned vectorIndex (const unsigned &i, const unsigned &j) const override=0

Protected Member Functions inherited from pmat::Matrix

- virtual unsigned vectorIndex (const unsigned &row, const unsigned &column) const
- double vectorElement (const unsigned &row, const unsigned &column) const
- void moveToThis (Matrix &&matrix)
- void initializeMembers (unsigned rowSize, unsigned columnSize, bool isTransposed)
- void copyMembers (const Matrix &matrix)
- bool isTransposed () const

6.12.1 Constructor & Destructor Documentation

6.12.1.1 MatrixTriangular() [1/4]

```
pmat::MatrixTriangular::MatrixTriangular ( )  [default]
```

6.12.1.2 MatrixTriangular() [2/4]

```
pmat::MatrixTriangular::MatrixTriangular (
            const MatrixTriangular & matrix )
```

6.12.1.3 MatrixTriangular() [3/4]

```
pmat::MatrixTriangular::MatrixTriangular (
            MatrixTriangular && matrix )  [default]
```

6.12.1.4 MatrixTriangular() [4/4]

```
pmat::MatrixTriangular::MatrixTriangular (
            const unsigned & size )  [inline], [explicit]
```

6.12.1.5 ~MatrixTriangular()

```
pmat::MatrixTriangular::~MatrixTriangular ( )  [override], [default]
```

6.12.2 Member Function Documentation

6.12.2.1 determinant()

```
double pmat::MatrixTriangular::determinant ( )
```

Calculates the determinant of this matrix through its diagonal.

Returns

double Determinant of this matrix

6.12.2.2 dotProduct()

```
double pmat::MatrixTriangular::dotProduct (
            const Matrix & matrix ) const  [override], [pure virtual]
```

Calculates the dot product of this matrix with the informed matrix.

The dot product of matrices A and B is

$$A : B = \sum_{i,j} A_{ij} B_{ij}$$

Parameters

matrix	Second operand

Returns

double Dot product result

Exceptions

std::invalid_argument	Operands are not compatible

Reimplemented from pmat::Matrix.

Implemented in pmat::MatrixLowerTriangular, and pmat::MatrixUpperTriangular.

6.12.2.3 fillWithRandomValues()

```
void pmat::MatrixTriangular::fillWithRandomValues (
            const double & min,
            const double & max )  [override], [pure virtual]
```

Fills the array with random values.

Parameters

min	Minimum acceptable value
max	Maximum acceptable value

Reimplemented from pmat::Matrix.

Implemented in pmat::MatrixLowerTriangular, and pmat::MatrixUpperTriangular.

6.12.2.4 findInverseByBackSubstitution()

```
void pmat::MatrixTriangular::findInverseByBackSubstitution (
            const MatrixTriangular & matrix,
            MatrixTriangular & resp )  [static]
```

Independent function for finding inverse by back substitution.

Parameters

matrix	Matrix to be inverted
resp	Inverse of the first argument

6.12.2.5 findSolutionByBackSubstitution()

```
pmat::Vector pmat::MatrixTriangular::findSolutionByBackSubstitution (
            const MatrixTriangular & matrix,
            const Vector & rhs )  [static]
```

Independent function for find the solution of a linear system by back substitution.

Parameters

matrix	Known matrix of the linear system
rhs	

Returns

pmat::Vector Solution of the linear system

6.12.2.6 getSwappedByColumns()

```
pmat::MatrixSquare pmat::MatrixTriangular::getSwappedByColumns (
            const unsigned & columnIndexA,
            const unsigned & columnIndexB ) const
```

6.12.2.7 getSwappedByRows()

```
pmat::MatrixSquare pmat::MatrixTriangular::getSwappedByRows (
            const unsigned & rowIndexA,
            const unsigned & rowIndexB ) const
```

6.12.2.8 isInvertible()

```
bool pmat::MatrixTriangular::isInvertible ( )  [virtual]
```

Informs if this matrix is invertible by inspecting its diagonal.

Returns

true Matrix is invertible

false Matrix is not invertible

6.12.2.9 length()

```
unsigned pmat::MatrixTriangular::length ( ) const  [override], [virtual]
```

Informs the number of elements.

Returns

unsigned Number of elements

Reimplemented from pmat::Matrix.

6.12.2.10 linearSolve()

```
pmat::Vector pmat::MatrixTriangular::linearSolve (
            const Vector & rhs )
```

Calculates the solution of a linear system by back substitution.

Parameters

rhs	Right hand side of the linear system

Returns

Vector Solution of the linear system

6.12.2.11 operator()()

```
double pmat::MatrixTriangular::operator() (
            const unsigned & row,
            const unsigned & column ) const  [override], [pure virtual]
```

Informs the value at the informed position.

Parameters

row	Row position of the value to be informed
column	Column position of the value to be informed

Returns

double Value at the informed position

Reimplemented from pmat::Matrix.

Implemented in pmat::MatrixLowerTriangular, and pmat::MatrixUpperTriangular.

6.12.2.12 operator∗()

```
pmat::MatrixSquare pmat::MatrixTriangular::operator* (
            const MatrixTriangular & matrix ) const
```

6.12.2.13 operator=() [1/2]

```
MatrixTriangular & pmat::MatrixTriangular::operator= (
            const MatrixTriangular &  )  [default]
```

6.12.2.14 operator=() [2/2]

```
MatrixTriangular & pmat::MatrixTriangular::operator= (
            MatrixTriangular &&  )  [default]
```

6.12.2.15 swapColumns()

```
void pmat::MatrixTriangular::swapColumns (
            const unsigned & columnA,
            const unsigned & columnB,
            const unsigned & startRow,
            const unsigned & endRow )  [override], [pure virtual]
```

Swaps the columns at the informed positions in a range of rows.

Parameters

columnA	Position A
columnB	Position B
startRow	Start row position
endRow	End row postion

Exceptions

std::invalid_argument	Index out of bounds

Reimplemented from pmat::Matrix.

Implemented in pmat::MatrixLowerTriangular, and pmat::MatrixUpperTriangular.

6.12.2.16 swapRows()

```
void pmat::MatrixTriangular::swapRows (
            const unsigned & rowA,
            const unsigned & rowB,
            const unsigned & startColumn,
            const unsigned & endColumn )  [override], [pure virtual]
```

Swaps the rows at the informed positions in a range of columns.

Parameters

rowA	Position A
rowB	Position B
startColumn	Start column position
endColumn	End column postion

Exceptions

std::invalid_argument	Index out of bounds

Reimplemented from pmat::Matrix.

Implemented in pmat::MatrixLowerTriangular, and pmat::MatrixUpperTriangular.

6.12.2.17 type()

```
virtual TriangType pmat::MatrixTriangular::type ( ) const  [pure virtual]
```

Implemented in pmat::MatrixLowerTriangular, and pmat::MatrixUpperTriangular.

6.12.2.18 vectorIndex()

```
unsigned pmat::MatrixTriangular::vectorIndex (
            const unsigned & i,
            const unsigned & j ) const  [override], [protected], [pure virtual]
```

Reimplemented from pmat::Matrix.

Implemented in pmat::MatrixLowerTriangular, and pmat::MatrixUpperTriangular.

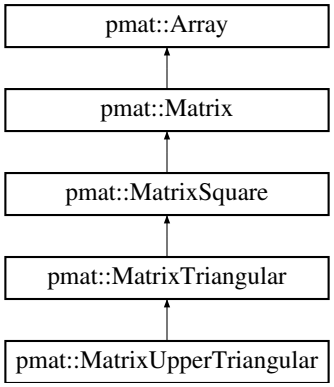
The documentation for this class was generated from the following files:

- src/MatrixTriangular.h
- src/MatrixTriangular.cpp

6.13 pmat::MatrixUpperTriangular Class Reference

```
#include <MatrixUpperTriangular.h>
```

Inheritance diagram for pmat::MatrixUpperTriangular:



Public Member Functions

- MatrixUpperTriangular ()=default
- MatrixUpperTriangular (const MatrixUpperTriangular &matrix)=default
- MatrixUpperTriangular (MatrixUpperTriangular &&matrix)=default
- MatrixUpperTriangular (const unsigned &size)
- ~MatrixUpperTriangular () override=default
- MatrixUpperTriangular & operator= (const MatrixUpperTriangular &matrix)=default
- MatrixUpperTriangular & operator= (MatrixUpperTriangular &&matrix)=default
- double operator() (const unsigned &row, const unsigned &column) const override

  Informs the value at the informed position.
- double dotProduct (const Matrix &matrix) const override

  Calculates the dot product of this matrix with the informed matrix.
- MatrixUpperTriangular operator+ (const MatrixUpperTriangular &matrix) const
- virtual void addBy (const MatrixUpperTriangular &matrix)
- MatrixUpperTriangular operator- (const MatrixUpperTriangular &matrix) const
- virtual void subtractBy (const MatrixUpperTriangular &matrix)

- MatrixUpperTriangular operator∗ (const double &scalar) const
- MatrixSquare operator∗ (const MatrixSquare &matrix) const
- void multiplyBy (const double &scalar) override

  Multiplies this matrix and the informed scalar, setting the result in this matrix.
- MatrixSquare operator+ (const MatrixSquare &matrix) const
- MatrixSquare operator- (const MatrixSquare &matrix) const
- Vector operator∗ (const Vector &vector) const override

  Multiplies this matrix and the informed vector, considered as a column matrix.
- MatrixUpperTriangular operator∗ (const MatrixUpperTriangular &matrix) const
- MatrixSquare operator∗ (const MatrixTriangular &matrix) const
- MatrixLowerTriangular getTranspose () const

  Gets the transposed matrix of this lower triangular matrix.
- void swapRows (const unsigned &rowA, const unsigned &rowB, const unsigned &startColumn, const unsigned &endColumn) override

  Swaps the rows at the informed positions in a range of columns.
- void swapColumns (const unsigned &colA, const unsigned &colB, const unsigned &startRow, const unsigned &endRow) override

  Swaps the columns at the informed positions in a range of rows.
- void fillWithRandomValues (const double &min, const double &max) override

  Fills the array with random values.
- TriangType type () const override

  Informs the triangular type of his matrix.
- MatrixUpperTriangular inverse ()

  Calculates the inverse of this matrix through back substitution.

Public Member Functions inherited from pmat::MatrixTriangular

- MatrixTriangular ()=default
- MatrixTriangular (const MatrixTriangular &matrix)
- MatrixTriangular (MatrixTriangular &&matrix)=default
- MatrixTriangular (const unsigned &size)
- ∼MatrixTriangular () override=default
- MatrixTriangular & operator= (const MatrixTriangular &)=default
- MatrixTriangular & operator= (MatrixTriangular &&)=default
- unsigned length () const override

  Informs the number of elements.
- double operator() (const unsigned &row, const unsigned &column) const override=0

  Informs the value at the informed position.
- double dotProduct (const Matrix &matrix) const override=0

  Calculates the dot product of this matrix with the informed matrix.
- MatrixSquare operator∗ (const MatrixTriangular &matrix) const
- MatrixSquare getSwappedByRows (const unsigned &rowIndexA, const unsigned &rowIndexB) const
- MatrixSquare getSwappedByColumns (const unsigned &columnIndexA, const unsigned &columnIndexB) const
- void fillWithRandomValues (const double &min, const double &max) override=0

  Fills the array with random values.
- void swapRows (const unsigned &rowA, const unsigned &rowB, const unsigned &startColumn, const unsigned &endColumn) override=0

  Swaps the rows at the informed positions in a range of columns.
- void swapColumns (const unsigned &columnA, const unsigned &columnB, const unsigned &startRow, const unsigned &endRow) override=0

  Swaps the columns at the informed positions in a range of rows.

- virtual TriangType type () const =0
- double determinant ()

  Calculates the determinant of this matrix through its diagonal.
- virtual bool isInvertible ()

  Informs if this matrix is invertible by inspecting its diagonal.
- Vector linearSolve (const Vector &rhs)

  Calculates the solution of a linear system by back substitution.

Public Member Functions inherited from pmat::MatrixSquare

- MatrixSquare ()=default
- MatrixSquare (MatrixSquare &&matrix) noexcept=default
- MatrixSquare (Matrix &&matrix)
- MatrixSquare (const unsigned &size)
- MatrixSquare (const MatrixSquare &matrix)
- ~MatrixSquare () override=default
- MatrixSquare & operator= (const MatrixSquare &matrix)=default
- MatrixSquare & operator= (MatrixSquare &&matrix) noexcept=default
- unsigned size () const
- virtual void resize (const unsigned &size)
- MatrixSquare operator+ (const MatrixSquare &matrix) const
- MatrixSquare operator- (const MatrixSquare &matrix) const
- MatrixSquare operator* (const MatrixSquare &matrix) const
- MatrixSquare operator* (const double &scalar) const
- Vector operator* (const Vector &vector) const override

  Multiplies this matrix and the informed vector, considered as a column matrix.
- virtual void fillDiagonalWith (const double &value)
- virtual MatrixSquare multiplyByBiggerMatrix (const MatrixSquare &matrix, SubMatrixPos pos)

  Calculates the multiplication of this matrix and the first parameter.
- virtual double trace () const

  Calculates the trace of this matrix.
- DecompositionPLU decomposeToPLU () const

  Returns a calculator for the PLU Decomposition of this matrix.
- DecompositionSAS decomposeToSAS () const

  Returns a calculator for the SAS Decomposition of this matrix.
- DecompositionPQR decomposeToPQR () const

  Returns a calculator for the PQR Decomposition of this matrix.

Public Member Functions inherited from pmat::Matrix

- Matrix ()=default
- Matrix (const unsigned &rowSize, const unsigned &columnSize)
- Matrix (const std::string &fileName)
- Matrix (const Matrix &matrix)
- Matrix (Matrix &&matrix) noexcept
- ~Matrix () override=default
- unsigned length () const override

  Informs the number of elements.
- unsigned dimension () const override

  Informs the dimension of the array.
- void resize (const unsigned &rowSize, const unsigned &columnSize)

Clears this matrix and sets a new size.

- void clear () override

  Removes all elements and sets size zero.

- virtual void setValue (const double &value, const unsigned &row, const unsigned &column)

  Sets the informed value at the informed position.

- virtual double operator() (const unsigned &row, const unsigned &column) const

  Informs the value at the informed position.

- unsigned rowSize () const

  Informs the size of matrix row dimension.

- unsigned columnSize () const

  Informs the size of matrix column dimension.

- Matrix & operator= (const Matrix &matrix)
- Matrix & operator= (Matrix &&matrix) noexcept
- virtual bool operator== (const Matrix &matrix) const
- virtual double dotProduct (const Matrix &matrix) const

  Calculates the dot product of this matrix with the informed matrix.

- Matrix operator+ (const Matrix &matrix) const

  Sums this matrix with the informed matrix.

- void addBy (const Matrix &matrix)

  Sums this matrix with the informed matrix, setting the result in this matrix.

- Matrix operator- (const Matrix &matrix) const

  Subtracts this matrix with the informed matrix.

- void subtractBy (const Matrix &matrix)

  Subtracts this matrix with the informed matrix, setting the result in this matrix.

- virtual Matrix operator∗ (const Matrix &matrix) const

  Multiplies this matrix and the informed matrix.

- virtual Vector operator∗ (const Vector &vector) const

  Multiplies this matrix and the informed vector, considered as a column matrix.

- Matrix operator∗ (const double &scalar) const

  Multiplies this matrix by the informed scalar.

- virtual void multiplyBy (const double &scalar)

  Multiplies this matrix and the informed scalar, setting the result in this matrix.

- Matrix multiply (const Matrix &matrix, unsigned nThreads)

  Multiplies this matrix by the informed scalar using multiple threads.

- Matrix multiplyHadamardBy (const Matrix &matrix) const

  Performs the Hadamard multiplication of this matrix and the informed matrix.

- virtual void multiplyRowBy (const unsigned &row, const double &scalar)

  Multiplies all the elements of the informed row by the informed scalar.

- virtual void multiplyColumnBy (const unsigned &column, const double &scalar)

  Multiplies all the elements of the informed column by the informed scalar.

- virtual void swapRows (const unsigned &rowA, const unsigned &rowB, const unsigned &startColumn, const unsigned &endColumn)

  Swaps the rows at the informed positions in a range of columns.

- virtual void swapRows (const unsigned &rowA, const unsigned &rowB)

  Swaps the rows at the informed positions.

- virtual void swapColumns (const unsigned &columnA, const unsigned &columnB, const unsigned &startRow, const unsigned &endRow)

  Swaps the columns at the informed positions in a range of rows.

- virtual void swapColumns (const unsigned &columnA, const unsigned &columnB)

  Swaps the columns at the informed positions.

- virtual void transpose ()

Transposes this matrix.

- virtual double getFrobeniusNorm () const

  Calculates the Frobenius Norm of this matrix.
- void fillWithRandomValues (const double &min, const double &max) override

  Fills the array with random values.
- Vector rowToVector (const unsigned &row) const

  Gets the informed row as a vector.
- Vector columnToVector (const unsigned &column) const

  Gets the informed column as a vector.
- unsigned occurrences (const double &value) const override

  Informes the number of occurrences of a value in the array.
- virtual unsigned occurrencesInRow (const unsigned row, const double &value) const

  Informs the number of occurrences of the informed value at the informed row.
- virtual unsigned occurrencesInColumn (const unsigned column, const double &value) const

  Informs the number of occurrences of the informed value at the informed column.

Public Member Functions inherited from pmat::Array

- Array ()=default
- Array (const Array &array)=default
- Array (Array &&)=default
- Array & operator= (const Array &)=default
- Array & operator= (Array &&)=default
- virtual ~Array ()=default
- virtual unsigned length () const =0

  Informs the number of elements.
- virtual unsigned dimension () const =0

  Informs the dimension of the array.
- virtual void clear ()=0

  Removes all elements and sets size zero.
- virtual unsigned occurrences (const double &value) const =0

  Informes the number of occurrences of a value in the array.
- virtual void fillWithRandomValues (const double &min, const double &max)=0

  Fills the array with random values.

Protected Member Functions

- unsigned vectorIndex (const unsigned &i, const unsigned &j) const override

- unsigned vectorIndex (const unsigned &i, const unsigned &j) const override=0

Protected Member Functions inherited from pmat::Matrix

- virtual unsigned vectorIndex (const unsigned &row, const unsigned &column) const
- double vectorElement (const unsigned &row, const unsigned &column) const
- void moveToThis (Matrix &&matrix)
- void initializeMembers (unsigned rowSize, unsigned columnSize, bool isTransposed)
- void copyMembers (const Matrix &matrix)
- bool isTransposed () const

Additional Inherited Members

Static Public Member Functions inherited from pmat::MatrixTriangular

- static void findInverseByBackSubstitution (const MatrixTriangular &matrix, MatrixTriangular &resp)

  Independent function for finding inverse by back substitution.
- static pmat::Vector findSolutionByBackSubstitution (const MatrixTriangular &matrix, const Vector &rhs)

  Independent function for find the solution of a linear system by back substitution.

6.13.1 Constructor & Destructor Documentation

6.13.1.1 MatrixUpperTriangular() [1/4]

```
pmat::MatrixUpperTriangular::MatrixUpperTriangular ( )  [default]
```

6.13.1.2 MatrixUpperTriangular() [2/4]

```
pmat::MatrixUpperTriangular::MatrixUpperTriangular (
            const MatrixUpperTriangular & matrix )  [default]
```

6.13.1.3 MatrixUpperTriangular() [3/4]

```
pmat::MatrixUpperTriangular::MatrixUpperTriangular (
            MatrixUpperTriangular && matrix )  [default]
```

6.13.1.4 MatrixUpperTriangular() [4/4]

```
pmat::MatrixUpperTriangular::MatrixUpperTriangular (
            const unsigned & size )  [inline], [explicit]
```

6.13.1.5 ~MatrixUpperTriangular()

```
pmat::MatrixUpperTriangular::~MatrixUpperTriangular ( )  [override], [default]
```

6.13.2 Member Function Documentation

6.13.2.1 addBy()

```
void pmat::MatrixUpperTriangular::addBy (
            const MatrixUpperTriangular & matrix )  [virtual]
```

6.13.2.2 dotProduct()

```
double pmat::MatrixUpperTriangular::dotProduct (
            const Matrix & matrix ) const  [override], [virtual]
```

Calculates the dot product of this matrix with the informed matrix.

The dot product of matrices A and B is

$$A : B = \sum_{i,j} A_{ij} B_{ij}$$

Parameters

matrix	Second operand

Returns

double Dot product result

Exceptions

std::invalid_argument	Operands are not compatible

Implements pmat::MatrixTriangular.

6.13.2.3 fillWithRandomValues()

```
void pmat::MatrixUpperTriangular::fillWithRandomValues (
            const double & min,
            const double & max )  [override], [virtual]
```

Fills the array with random values.

Parameters

min	Minimum acceptable value
max	Maximum acceptable value

Implements pmat::MatrixTriangular.

6.13.2.4 getTranspose()

```
pmat::MatrixLowerTriangular pmat::MatrixUpperTriangular::getTranspose ( ) const
```

Gets the transposed matrix of this lower triangular matrix.

Returns

MatrixUpperTriangular Transposed matrix

6.13.2.5 inverse()

```
pmat::MatrixUpperTriangular pmat::MatrixUpperTriangular::inverse ( )
```

Calculates the inverse of this matrix through back substitution.

Returns

MatrixUpperTriangular The inverse of this matrix

Exceptions

throw	std::logic_error Singular matrix

6.13.2.6 multiplyBy()

```
void pmat::MatrixUpperTriangular::multiplyBy (
            const double & scalar )  [override], [virtual]
```

Multiplies this matrix and the informed scalar, setting the result in this matrix.

Parameters

scalar	Second operand of the multiplication

Reimplemented from pmat::Matrix.

6.13.2.7 operator()()

```
double pmat::MatrixUpperTriangular::operator() (
            const unsigned & row,
            const unsigned & column ) const  [override], [virtual]
```

Informs the value at the informed position.

Parameters

row	Row position of the value to be informed
column	Column position of the value to be informed

Returns

double Value at the informed position

Implements pmat::MatrixTriangular.

6.13.2.8 operator*() [1/5]

```
pmat::MatrixUpperTriangular pmat::MatrixUpperTriangular::operator* (
            const double & scalar ) const
```

6.13.2.9 operator*() [2/5]

```
pmat::MatrixSquare pmat::MatrixUpperTriangular::operator* (
            const MatrixSquare & matrix ) const
```

6.13.2.10 operator∗() [3/5]

```
pmat::MatrixSquare pmat::MatrixUpperTriangular::operator* (
            const MatrixTriangular & matrix ) const
```

6.13.2.11 operator∗() [4/5]

```
pmat::MatrixUpperTriangular pmat::MatrixUpperTriangular::operator* (
            const MatrixUpperTriangular & matrix ) const
```

6.13.2.12 operator∗() [5/5]

```
pmat::Vector pmat::MatrixUpperTriangular::operator* (
            const Vector & vector ) const  [override], [virtual]
```

Multiplies this matrix and the informed vector, considered as a column matrix.

Parameters

vector	Right operand

Returns

Vector Multiplication result

Exceptions

std::invalid_argument	Incompatible sizes

Reimplemented from pmat::MatrixSquare.

6.13.2.13 operator+() [1/2]

```
pmat::MatrixSquare pmat::MatrixUpperTriangular::operator+ (
            const MatrixSquare & matrix ) const
```

6.13.2.14 operator+() [2/2]

```
pmat::MatrixUpperTriangular pmat::MatrixUpperTriangular::operator+ (
            const MatrixUpperTriangular & matrix ) const
```

6.13.2.15 operator-() [1/2]

```
pmat::MatrixSquare pmat::MatrixUpperTriangular::operator- (
            const MatrixSquare & matrix ) const
```

6.13.2.16 operator-() [2/2]

```
pmat::MatrixUpperTriangular pmat::MatrixUpperTriangular::operator- (
            const MatrixUpperTriangular & matrix ) const
```

6.13.2.17 operator=() [1/2]

```
MatrixUpperTriangular & pmat::MatrixUpperTriangular::operator= (
            const MatrixUpperTriangular & matrix )  [default]
```

6.13.2.18 operator=() [2/2]

```
MatrixUpperTriangular & pmat::MatrixUpperTriangular::operator= (
            MatrixUpperTriangular && matrix )  [default]
```

6.13.2.19 subtractBy()

```
void pmat::MatrixUpperTriangular::subtractBy (
            const MatrixUpperTriangular & matrix )  [virtual]
```

6.13.2.20 swapColumns()

```
void pmat::MatrixUpperTriangular::swapColumns (
            const unsigned & columnA,
            const unsigned & columnB,
            const unsigned & startRow,
            const unsigned & endRow )  [override], [virtual]
```

Swaps the columns at the informed positions in a range of rows.

Parameters

columnA	Position A
columnB	Position B
startRow	Start row position
endRow	End row postion

Exceptions

std::invalid_argument	Index out of bounds

Implements pmat::MatrixTriangular.

6.13.2.21 swapRows()

```
void pmat::MatrixUpperTriangular::swapRows (
            const unsigned & rowA,
```

```
            const unsigned & rowB,
            const unsigned & startColumn,
            const unsigned & endColumn )  [override], [virtual]
```

Swaps the rows at the informed positions in a range of columns.

Parameters

rowA	Position A
rowB	Position B
startColumn	Start column position
endColumn	End column postion

Exceptions

std::invalid_argument	Index out of bounds

Implements pmat::MatrixTriangular.

6.13.2.22 type()

```
TriangType pmat::MatrixUpperTriangular::type ( ) const  [inline], [override], [virtual]
```

Informs the triangular type of his matrix.

Returns

TriangType Triangular type

Implements pmat::MatrixTriangular.

6.13.2.23 vectorIndex()

```
unsigned pmat::MatrixUpperTriangular::vectorIndex (
            const unsigned & i,
            const unsigned & j ) const  [override], [protected], [virtual]
```

Implements pmat::MatrixTriangular.

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

- src/MatrixUpperTriangular.h
- src/MatrixUpperTriangular.cpp

6.14 pmat::TMultiplicationManager Class Reference

```
#include <TMultiplicationManager.h>
```

Public Member Functions

- TMultiplicationManager (const Matrix &operandFirst, const Matrix &operandSecond, Matrix &result)
- TMultiplicationManager (const TMultiplicationManager &)=delete
- TMultiplicationManager (TMultiplicationManager &&)=delete
- TMultiplicationManager & operator= (const TMultiplicationManager &)=delete
- TMultiplicationManager & operator= (TMultiplicationManager &&)=delete
- ~TMultiplicationManager ()=default
- const Matrix & operandFirst () const
- const Matrix & operandSecond () const
- void setResultValue (const double &value, const unsigned &row, const unsigned &column)
- bool getNextRowColumn (unsigned id)
- void multiply (int nThreads)

6.14.1 Constructor & Destructor Documentation

6.14.1.1 TMultiplicationManager() [1/3]

```
pmat::TMultiplicationManager::TMultiplicationManager (
            const Matrix & operandFirst,
            const Matrix & operandSecond,
            Matrix & result )
```

6.14.1.2 TMultiplicationManager() [2/3]

```
pmat::TMultiplicationManager::TMultiplicationManager (
            const TMultiplicationManager &  )  [delete]
```

6.14.1.3 TMultiplicationManager() [3/3]

```
pmat::TMultiplicationManager::TMultiplicationManager (
            TMultiplicationManager &&  )  [delete]
```

6.14.1.4 ~TMultiplicationManager()

```
pmat::TMultiplicationManager::~TMultiplicationManager ( )  [default]
```

6.14.2 Member Function Documentation

6.14.2.1 getNextRowColumn()

```
bool pmat::TMultiplicationManager::getNextRowColumn (
            unsigned id )
```

6.14.2.2 multiply()

```
void pmat::TMultiplicationManager::multiply (
            int nThreads )
```

6.14.2.3 operandFirst()

```
const Matrix & pmat::TMultiplicationManager::operandFirst ( ) const  [inline]
```

6.14.2.4 operandSecond()

```
const Matrix & pmat::TMultiplicationManager::operandSecond ( ) const  [inline]
```

6.14.2.5 operator=() [1/2]

```
TMultiplicationManager & pmat::TMultiplicationManager::operator= (
            const TMultiplicationManager &  )  [delete]
```

6.14.2.6 operator=() [2/2]

```
TMultiplicationManager & pmat::TMultiplicationManager::operator= (
            TMultiplicationManager &&  )  [delete]
```

6.14.2.7 setResultValue()

```
void pmat::TMultiplicationManager::setResultValue (
            const double & value,
            const unsigned & row,
            const unsigned & column )
```

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

- src/TMultiplicationManager.h
- src/TMultiplicationManager.cpp

6.15 pmat::TMultiplicationPerformer Class Reference

```
#include <TMultiplicationPerformer.h>
```

Public Member Functions

- TMultiplicationPerformer (unsigned id, TMultiplicationManager &manager)
- TMultiplicationPerformer (const TMultiplicationPerformer &)=default
- TMultiplicationPerformer (TMultiplicationPerformer &&)=default
- TMultiplicationPerformer & operator= (const TMultiplicationPerformer &)=default
- TMultiplicationPerformer & operator= (TMultiplicationPerformer &&)=default
- ~TMultiplicationPerformer ()=default
- void start ()
- void setRowColumn (const unsigned &row, const unsigned &column)

6.15.1 Constructor & Destructor Documentation

6.15.1.1 TMultiplicationPerformer() [1/3]

```
pmat::TMultiplicationPerformer::TMultiplicationPerformer (
            unsigned id,
            TMultiplicationManager & manager )  [inline]
```

6.15.1.2 TMultiplicationPerformer() [2/3]

```
pmat::TMultiplicationPerformer::TMultiplicationPerformer (
            const TMultiplicationPerformer &  )  [default]
```

6.15.1.3 TMultiplicationPerformer() [3/3]

```
pmat::TMultiplicationPerformer::TMultiplicationPerformer (
            TMultiplicationPerformer &&  )  [default]
```

6.15.1.4 ~TMultiplicationPerformer()

```
pmat::TMultiplicationPerformer::~TMultiplicationPerformer ( )  [default]
```

6.15.2 Member Function Documentation

6.15.2.1 operator=() [1/2]

```
TMultiplicationPerformer & pmat::TMultiplicationPerformer::operator= (
            const TMultiplicationPerformer &  )  [default]
```

6.15.2.2 operator=() [2/2]

```
TMultiplicationPerformer & pmat::TMultiplicationPerformer::operator= (
            TMultiplicationPerformer &&  )  [default]
```

6.15.2.3 setRowColumn()

```
void pmat::TMultiplicationPerformer::setRowColumn (
            const unsigned & row,
            const unsigned & column )
```

6.15.2.4 start()

```
void pmat::TMultiplicationPerformer::start ( )
```

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

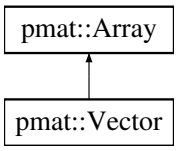
- src/TMultiplicationPerformer.h
- src/TMultiplicationPerformer.cpp

6.16 pmat::Vector Class Reference

Entity for one-dimensional array.

`#include <Vector.h>`

Inheritance diagram for pmat::Vector:



Public Member Functions

- Vector ()=default
- Vector (unsigned size)
- Vector (const Vector &vector)
- Vector (Vector &&vector) noexcept
- ~Vector () override=default
- unsigned length () const override

  Informs the number of elements.
- unsigned dimension () const override

  Informs the dimension of the array.
- unsigned size () const

  Informs the size of vector dimension.
- void resize (const unsigned &size)

  Clears this vector and sets a new size.
- void clear () override

  Removes all elements and sets size zero.
- void emplaceBack (const double &value)
- void setValue (const double &value, const unsigned &index)

  Sets the informed value at the informed position.

- const double & operator() (const unsigned &index) const

  Informs the value at the informed position.
- Vector & operator= (const Vector &vector)
- Vector & operator= (Vector &&vector) noexcept
- bool operator== (const Vector &vector) const
- Vector operator+ (const Vector &vector) const

  Sums this vector and the informed vector.
- void addBy (const Vector &vector)

  Sums this vector and the informed vector, setting the result in this vector.
- Vector operator- (const Vector &vector) const

  Subtracts this vector and the informed vector.
- void subtractBy (const Vector &vector)

  Subtracts this vector and the informed vector, setting the result in this vector.
- Vector operator∗ (const double &scalar) const

  Multiplies this vector by the informed scalar.
- void multiplyBy (const double &scalar)

  Multiplies this vector by the informed scalar, setting the result in this vector.
- double dotProduct (const Vector &vector) const

  Calculates the dot product of this vector with the informed vector.
- double frobeniusNorm () const

  Calculates the Frobenius Norm of this vector.
- Vector getUnitaryVector () const

  Gets the unitary vector related to this vector.
- unsigned occurrences (const double &value) const override

  Informes the number of occurrences of a value in the array.
- void fillWithRandomValues (const double &min, const double &max) override

  Fills the array with random values.
- void swapElements (const unsigned &elmIndexA, const unsigned &elmIndexB)

  Swaps the elements of the vector according to the informed positions.
- void ascendingSort ()

  Sorts the values of this vector in ascending order.
- void descendingSort ()

  Sorts the values of this vector in descending order.
- Matrix toColumnMatrix () const

  Converts this vector to a column matrix.
- Matrix toRowMatrix () const

  Converts this vector to a row matrix.

Public Member Functions inherited from pmat::Array

- Array ()=default
- Array (const Array &array)=default
- Array (Array &&)=default
- Array & operator= (const Array &)=default
- Array & operator= (Array &&)=default
- virtual ∼Array ()=default
- virtual unsigned length () const =0

  Informs the number of elements.
- virtual unsigned dimension () const =0

  Informs the dimension of the array.
- virtual void clear ()=0

Removes all elements and sets size zero.

- virtual unsigned occurrences (const double &value) const =0

  Informes the number of occurrences of a value in the array.

- virtual void fillWithRandomValues (const double &min, const double &max)=0

  Fills the array with random values.

6.16.1 Detailed Description

Entity for one-dimensional array.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 Vector() [1/4]

```
pmat::Vector::Vector ( )  [default]
```

6.16.2.2 Vector() [2/4]

```
pmat::Vector::Vector (
            unsigned size )  [inline]
```

6.16.2.3 Vector() [3/4]

```
pmat::Vector::Vector (
            const Vector & vector )
```

6.16.2.4 Vector() [4/4]

```
pmat::Vector::Vector (
            Vector && vector )  [inline], [noexcept]
```

6.16.2.5 ~Vector()

```
pmat::Vector::~Vector ( )  [override], [default]
```

6.16.3 Member Function Documentation

6.16.3.1 addBy()

```
void pmat::Vector::addBy (
            const Vector & vector )
```

Sums this vector and the informed vector, setting the result in this vector.

Parameters

vector	Second operand

6.16.3.2 ascendingSort()

```
void pmat::Vector::ascendingSort ( )
```

Sorts the values of this vector in ascending order.

6.16.3.3 clear()

```
void pmat::Vector::clear ( )  [inline], [override], [virtual]
```

Removes all elements and sets size zero.

Implements pmat::Array.

6.16.3.4 descendingSort()

```
void pmat::Vector::descendingSort ( )
```

Sorts the values of this vector in descending order.

6.16.3.5 dimension()

```
unsigned pmat::Vector::dimension ( ) const  [inline], [override], [virtual]
```

Informs the dimension of the array.

Returns

unsigned Dimension

Implements pmat::Array.

6.16.3.6 dotProduct()

```
double pmat::Vector::dotProduct (
            const Vector & vector ) const
```

Calculates the dot product of this vector with the informed vector.

Parameters

vector	Second operand

The dot product of vectors v and u is

$$v.u = \sum_i v_i u_i$$

Returns

double Dot product result

Exceptions

std::invalid_argument	Operands are not compatible

6.16.3.7 emplaceBack()

```
void pmat::Vector::emplaceBack (
            const double & value )
```

6.16.3.8 fillWithRandomValues()

```
void pmat::Vector::fillWithRandomValues (
            const double & min,
            const double & max )  [override], [virtual]
```

Fills the array with random values.

Parameters

min	Minimum acceptable value
max	Maximum acceptable value

Implements pmat::Array.

6.16.3.9 frobeniusNorm()

```
double pmat::Vector::frobeniusNorm ( ) const
```

Calculates the Frobenius Norm of this vector.

Frobenius Norm of vector v is calculated from the dot product the following way:

$$\sqrt{v : v}$$

Returns

Frobenius Norm result

6.16.3.10 getUnitaryVector()

```
pmat::Vector pmat::Vector::getUnitaryVector ( ) const
```

Gets the unitary vector related to this vector.

Returns

Vector Unitary vector

6.16.3.11 length()

```
unsigned pmat::Vector::length ( ) const  [inline], [override], [virtual]
```

Informs the number of elements.

Returns

unsigned Number of elements

Implements pmat::Array.

6.16.3.12 multiplyBy()

```
void pmat::Vector::multiplyBy (
            const double & scalar )
```

Multiplies this vector by the informed scalar, setting the result in this vector.

Parameters

scalar	Second operand of the multiplication

6.16.3.13 occurrences()

```
unsigned pmat::Vector::occurrences (
            const double & value ) const  [override], [virtual]
```

Informes the number of occurrences of a value in the array.

Parameters

value	Value to be searched

Returns

unsigned Number of occurrences

Implements pmat::Array.

6.16.3.14 operator()()

```
const double & pmat::Vector::operator() (
            const unsigned & index ) const
```

Informs the value at the informed position.

Parameters

index	Position of the value to be informed

Returns

const double& Value at the informed position

6.16.3.15 operator∗()

```
pmat::Vector pmat::Vector::operator* (
            const double & scalar ) const
```

Multiplies this vector by the informed scalar.

Parameters

scalar	Second operand of the multiplication

Returns

Vector Multiplication result

6.16.3.16 operator+()

```
pmat::Vector pmat::Vector::operator+ (
            const Vector & vector ) const
```

Sums this vector and the informed vector.

Parameters

vector	Second operand of the sum

Returns

Vector Sum result

6.16.3.17 operator-()

```
pmat::Vector pmat::Vector::operator- (
            const Vector & vector ) const
```

Subtracts this vector and the informed vector.

Parameters

vector	Right operand of the sum

Returns

Vector Sum result

6.16.3.18 operator=() [1/2]

```
pmat::Vector & pmat::Vector::operator= (
            const Vector & vector )
```

6.16.3.19 operator=() [2/2]

```
pmat::Vector & pmat::Vector::operator= (
            Vector && vector )  [noexcept]
```

6.16.3.20 operator==()

```
bool pmat::Vector::operator== (
            const Vector & vector ) const
```

6.16.3.21 resize()

```
void pmat::Vector::resize (
            const unsigned & size )
```

Clears this vector and sets a new size.

Parameters

size	New size

6.16.3.22 setValue()

```
void pmat::Vector::setValue (
            const double & value,
            const unsigned & index )
```

Sets the informed value at the informed position.

Parameters

value	Value to be set
index	Position in vector

6.16.3.23 size()

```
unsigned pmat::Vector::size ( ) const  [inline]
```

Informs the size of vector dimension.

Returns

unsigned Size of vector dimension

6.16.3.24 subtractBy()

```
void pmat::Vector::subtractBy (
            const Vector & vector )
```

Subtracts this vector and the informed vector, setting the result in this vector.

Parameters

vector	Right operand

6.16.3.25 swapElements()

```
void pmat::Vector::swapElements (
            const unsigned & elmIndexA,
            const unsigned & elmIndexB )
```

Swaps the elements of the vector according to the informed positions.

Parameters

elmIndexA	Position A
elmIndexB	Position B

Exceptions

std::invalid_argument	Index out of bounds

6.16.3.26 toColumnMatrix()

```
pmat::Matrix pmat::Vector::toColumnMatrix ( ) const
```

Converts this vector to a column matrix.

Returns

Matrix Column matrix

6.16.3.27 toRowMatrix()

```
pmat::Matrix pmat::Vector::toRowMatrix ( ) const
```

Converts this vector to a row matrix.

Returns

Matrix Row matrix

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

- src/Vector.h
- src/Vector.cpp

Chapter 7

File Documentation

7.1 src/Array.cpp File Reference

```
#include ¨Array.h¨
```

7.2 src/Array.h File Reference

```
#include <vector>
```

Classes

- class pmat::Array

  Abstract entity for arrays of generic dimension.

Namespaces

- namespace pmat

7.3 Array.h

Go to the documentation of this file.

```
00001 #ifndef ABSTRACTARRAY_H
00002 #define ABSTRACTARRAY_H
00003 #pragma once
00004
00005 #include <vector>
00006
00007 namespace pmat {
00008
00013 class Array {
00014
00015    public:
00016       Array() = default;
00017       Array(const Array &array) = default;
00018       Array(Array &&) = default;
```

```
00019       Array &operator=(const Array &) = default;
00020       Array &operator=(Array &&) = default;
00021       virtual ~Array() = default;
00022
00028       [[nodiscard]] virtual unsigned length() const = 0;
00029
00035       [[nodiscard]] virtual unsigned dimension() const = 0;
00036
00041       virtual void clear() = 0;
00042
00049       [[nodiscard]] virtual unsigned occurrences(const double &value) const = 0;
00050
00057       virtual void fillWithRandomValues(const double &min, const double &max) = 0;
00058 };
00059
00060 } // namespace pmat
00061
00062 #endif // ABSTRACTARRAY_H
```

7.4 src/DecompositionCholesky.cpp File Reference

```
#include ¨DecompositionCholesky.h¨
#include ¨MatrixSymmetric.h¨
#include ¨utils.h¨
#include <cmath>
#include <stdexcept>
```

7.5 src/DecompositionCholesky.h File Reference

```
#include ¨DecompositionPLU.h¨
#include ¨MatrixLowerTriangular.h¨
#include ¨MatrixSymmetric.h¨
```

Classes

- class pmat::DecompositionCholesky

Namespaces

- namespace pmat

7.6 DecompositionCholesky.h

Go to the documentation of this file.

```
00001 #ifndef DecompositionCholesky_H
00002 #define DecompositionCholesky_H
00003 #pragma once
00004
00005 #include "DecompositionPLU.h"
00006 #include "MatrixLowerTriangular.h"
00007 #include "MatrixSymmetric.h"
00008
00009 namespace pmat {
00010
00011 // class MatrixSymmetric;
00012
```

```
00013 class DecompositionCholesky {
00014    private:
00015       const MatrixSymmetric *_matrix;
00016       MatrixLowerTriangular _factor{};
00017       bool _calculated{false};
00018       DecompositionPLU _plu;
00019       void calculate();
00020
00021    public:
00022       DecompositionCholesky(const MatrixSymmetric &matrix);
00023       DecompositionCholesky(const DecompositionCholesky &chk) = default;
00024       DecompositionCholesky(DecompositionCholesky &&chk) = default;
00025       DecompositionCholesky &operator=(const DecompositionCholesky &chk) = default;
00026       DecompositionCholesky &operator=(DecompositionCholesky &&chk) = default;
00027       ~DecompositionCholesky() = default;
00028
00037       const MatrixLowerTriangular &choleskyFactor();
00038
00044       double determinant();
00045
00052       bool isInvertible();
00053
00062       MatrixSquare inverse();
00063
00070       MatrixSymmetric inverseAsSymmetric();
00071
00081       Vector linearSolve(const Vector &rhs);
00082
00092       bool isPositiveDefinite();
00093 };
00094
00095 } // namespace pmat
00096
00097 #endif
```

7.7 src/DecompositionPLU.cpp File Reference

```
#include ¨DecompositionPLU.h¨
#include ¨utils.h¨
#include <stdexcept>
```

7.8 src/DecompositionPLU.h File Reference

```
#include ¨MatrixLowerTriangular.h¨
#include ¨MatrixSquare.h¨
#include ¨MatrixUpperTriangular.h¨
```

Classes

- class pmat::DecompositionPLU

Namespaces

- namespace pmat

7.9 DecompositionPLU.h

Go to the documentation of this file.

```
00001 #ifndef DecompositionPLU_H
00002 #define DecompositionPLU_H
00003 #pragma once
00004
00005 #include "MatrixLowerTriangular.h"
00006 #include "MatrixSquare.h"
00007 #include "MatrixUpperTriangular.h"
00008
00009 namespace pmat {
00010
00011 class DecompositionPLU {
00012    private:
00013       const MatrixSquare *_matrix;
00014       bool _strictLUMode{false};
00015       MatrixSquare _matP{};
00016       MatrixLowerTriangular _matL{};
00017       MatrixUpperTriangular _matU{};
00018       std::vector<std::pair<unsigned, unsigned» _swappedRows;
00019       bool _changeSignForDet{false};
00020       bool _calculated{false};
00021
00022       void swapRowsBellow(MatrixSquare &matU, const unsigned &idxPivot);
00023       void nullifyElementBellow(MatrixSquare &matU, const unsigned &idxPivot);
00024       void calculate();
00025
00026    public:
00027       DecompositionPLU(const MatrixSquare &matrix);
00028
00035       DecompositionPLU(const MatrixSquare &matrix, bool strictLUMode);
00036
00037       DecompositionPLU(const DecompositionPLU &plu) = default;
00038       DecompositionPLU(DecompositionPLU &&plu) = default;
00039       DecompositionPLU &operator=(const DecompositionPLU &plu) = default;
00040       DecompositionPLU &operator=(DecompositionPLU &&plu) = default;
00041       ~DecompositionPLU() = default;
00042
00048       [[nodiscard]] const MatrixSquare &matP();
00049
00055       [[nodiscard]] const MatrixLowerTriangular &matL();
00056
00062       [[nodiscard]] const MatrixUpperTriangular &matU();
00063
00069       [[nodiscard]] const std::vector<std::pair<unsigned, unsigned» &swappedRows() const;
00070
00076       [[nodiscard]] double determinant();
00077
00084       bool isStrictLUDecomposable();
00085
00092       bool isInvertible();
00093
00102       MatrixSquare inverse();
00103
00113       bool isPositiveDefinite();
00114
00122       bool isOrthogonal();
00123
00133       Vector linearSolve(const Vector &rhs);
00134
00141       [[nodiscard]] bool isStrictLUMode() const { return _strictLUMode; }
00142
00147       void setStrictLUMode();
00148 };
00149
00150 } // namespace pmat
00151 #endif
```

7.10 src/DecompositionPQR.cpp File Reference

```
#include ¨DecompositionPQR.h¨
#include ¨utils.h¨
#include <cmath>
#include <stdexcept>
```

7.11 src/DecompositionPQR.h File Reference

```
#include ¨MatrixSquare.h¨
#include ¨MatrixUpperTriangular.h¨
```

Classes

- class pmat::DecompositionPQR

Namespaces

- namespace pmat

7.12 DecompositionPQR.h

Go to the documentation of this file.

```
00001 #ifndef DecompositionPQR_H
00002 #define DecompositionPQR_H
00003 #pragma once
00004
00005 #include "MatrixSquare.h"
00006 #include "MatrixUpperTriangular.h"
00007
00008 namespace pmat {
00009
00010 class DecompositionPQR {
00011    private:
00012       const MatrixSquare *_matrix;
00013       MatrixSquare _matP;
00014       MatrixSquare _matQ;
00015       MatrixUpperTriangular _matR;
00016       std::vector<std::pair<unsigned, unsigned» _swappedColumns;
00017       unsigned _rank{0};
00018       bool _calculated{false};
00019
00020       [[nodiscard]] MatrixSquare calculateHouseholderSubMatrix(const MatrixSquare &partialR,
00021                                                                const unsigned idxPivot) const;
00022       void swapPivotColumn(MatrixSquare &partialR, const unsigned &idxPivot);
00023
00024       void calculate();
00025
00026    public:
00027       DecompositionPQR(const MatrixSquare &matrix);
00028       DecompositionPQR(const DecompositionPQR &pqr) = default;
00029       DecompositionPQR(DecompositionPQR &&pqr) = default;
00030       DecompositionPQR &operator=(const DecompositionPQR &pqr) = default;
00031       DecompositionPQR &operator=(DecompositionPQR &&pqr) = default;
00032       ~DecompositionPQR() = default;
00033
00039       const MatrixSquare &matP();
00040
00046       const MatrixSquare &matQ();
00047
00053       const MatrixUpperTriangular &matR();
00054
00061       const unsigned &rank();
00062
00069       bool isInvertible();
00070
00079       MatrixSquare inverse();
00080 };
00081 } // namespace pmat
00082 #endif
```

7.13 src/DecompositionSAS.cpp File Reference

```
#include ¨DecompositionSAS.h¨
#include ¨utils.h¨
```

7.14 src/DecompositionSAS.h File Reference

```
#include ¨MatrixSkewSymmetric.h¨
#include ¨MatrixSquare.h¨
#include ¨MatrixSymmetric.h¨
```

Classes

- class pmat::DecompositionSAS

Namespaces

- namespace pmat

7.15 DecompositionSAS.h

Go to the documentation of this file.

```
00001 #ifndef DecompositionSAS_H
00002 #define DecompositionSAS_H
00003 #pragma once
00004
00005 #include "MatrixSkewSymmetric.h"
00006 #include "MatrixSquare.h"
00007 #include "MatrixSymmetric.h"
00008
00009 namespace pmat {
00010
00011 class DecompositionSAS {
00012    private:
00013       const MatrixSquare *_matrix;
00014       MatrixSymmetric _matS{};
00015       MatrixSkewSymmetric _matAS{};
00016       bool _calculated{false};
00017
00018       void calculate();
00019
00020    public:
00021       DecompositionSAS(const MatrixSquare &matrix);
00022       DecompositionSAS(const DecompositionSAS &sas) = default;
00023       DecompositionSAS(DecompositionSAS &&sas) = default;
00024       DecompositionSAS &operator=(const DecompositionSAS &sas) = default;
00025       DecompositionSAS &operator=(DecompositionSAS &&sas) = default;
00026       ~DecompositionSAS() = default;
00027
00033       const MatrixSymmetric &matS();
00034
00040       const MatrixSkewSymmetric &matAS();
00041 };
00042
00043 } // namespace pmat
00044 #endif
```

7.16 src/Matrix.cpp File Reference

```
#include ¨Matrix.h¨
#include ¨Messages.h¨
#include ¨TMultiplicationManager.h¨
#include ¨utils.h¨
#include <fstream>
#include <locale>
#include <random>
#include <sstream>
#include <stdexcept>
#include <system_error>
#include <utility>
#include <vector>
```

7.17 src/Matrix.h File Reference

```
#include ¨Array.h¨
#include ¨Vector.h¨
#include <algorithm>
#include <string>
```

Classes

- class pmat::Matrix

Namespaces

- namespace pmat

7.18 Matrix.h

Go to the documentation of this file.

```
00001 #ifndef MATRIX_H
00002 #define MATRIX_H
00003 #pragma once
00004
00005 #include "Array.h"
00006 #include "Vector.h"
00007 #include <algorithm>
00008 #include <string>
00009
00010 namespace pmat {
00011
00012 class Matrix : public pmat::Array {
00013    private:
00014       std::vector<double> _matrix{};
00015       bool _isTransposed{false};
00016       unsigned _rowSize{0}, _columnSize{0};
00017
00018    protected:
00019       [[nodiscard]] virtual unsigned vectorIndex(const unsigned &row, const unsigned &column) const;
00020       [[nodiscard]] double vectorElement(const unsigned &row, const unsigned &column) const;
00021       void moveToThis(Matrix &&matrix);
00022
```

```
00023       void initializeMembers(unsigned rowSize, unsigned columnSize, bool isTransposed);
00024       void copyMembers(const Matrix &matrix);
00025       [[nodiscard]] bool isTransposed() const { return _isTransposed; }
00026
00027   public:
00028       Matrix() = default;
00029       Matrix(const unsigned &rowSize, const unsigned &columnSize);
00030       Matrix(const std::string &fileName);
00031       Matrix(const Matrix &matrix)
00032           : _matrix{matrix._matrix}, _rowSize{matrix.rowSize()}, _columnSize{matrix.columnSize()},
00033             _isTransposed{matrix._isTransposed} {};
00034       Matrix(Matrix &&matrix) noexcept
00035           : _matrix{std::move(matrix._matrix)}, _rowSize{matrix.rowSize()},
00036             _columnSize{matrix.columnSize()}, _isTransposed{matrix._isTransposed} {}
00037       ~Matrix() override = default;
00038       [[nodiscard]] unsigned length() const override { return _rowSize * _columnSize; }
00039       [[nodiscard]] inline unsigned dimension() const override { return 2; }
00040
00047       void resize(const unsigned &rowSize, const unsigned &columnSize);
00048
00049       void clear() override;
00050
00058       virtual void setValue(const double &value, const unsigned &row, const unsigned &column);
00059
00067       virtual double operator()(const unsigned &row, const unsigned &column) const;
00068
00074       [[nodiscard]] inline unsigned rowSize() const { return _rowSize; }
00075
00081       [[nodiscard]] inline unsigned columnSize() const { return _columnSize; }
00082
00083       Matrix &operator=(const Matrix &matrix);
00084       Matrix &operator=(Matrix &&matrix) noexcept;
00085       virtual bool operator==(const Matrix &matrix) const;
00086
00096       [[nodiscard]] virtual double dotProduct(const Matrix &matrix) const;
00097
00105       Matrix operator+(const Matrix &matrix) const;
00106
00113       void addBy(const Matrix &matrix);
00114
00122       Matrix operator-(const Matrix &matrix) const;
00123
00130       void subtractBy(const Matrix &matrix);
00131
00139       virtual Matrix operator*(const Matrix &matrix) const;
00140
00148       virtual Vector operator*(const Vector &vector) const;
00149
00156       Matrix operator*(const double &scalar) const;
00157
00163       virtual void multiplyBy(const double &scalar);
00164
00172       Matrix multiply(const Matrix &matrix, unsigned nThreads);
00173
00183       [[nodiscard]] Matrix multiplyHadamardBy(const Matrix &matrix) const;
00184
00192       virtual void multiplyRowBy(const unsigned &row, const double &scalar);
00193
00201       virtual void multiplyColumnBy(const unsigned &column, const double &scalar);
00202
00212       virtual void swapRows(const unsigned &rowA, const unsigned &rowB, const unsigned &startColumn,
00213                             const unsigned &endColumn);
00214
00222       virtual void swapRows(const unsigned &rowA, const unsigned &rowB);
00223
00233       virtual void swapColumns(const unsigned &columnA, const unsigned &columnB,
00234                                const unsigned &startRow, const unsigned &endRow);
00235
00243       virtual void swapColumns(const unsigned &columnA, const unsigned &columnB);
00244
00249       virtual void transpose();
00250
00257       [[nodiscard]] virtual double getFrobeniusNorm() const;
00258
00259       void fillWithRandomValues(const double &min, const double &max) override;
00260
00268       [[nodiscard]] Vector rowToVector(const unsigned &row) const;
00269
00277       [[nodiscard]] Vector columnToVector(const unsigned &column) const;
00278
00279       [[nodiscard]] unsigned occurrences(const double &value) const override;
00280
00289       [[nodiscard]] virtual unsigned occurrencesInRow(const unsigned row,
00290                                                       const double &value) const;
00291
00300       [[nodiscard]] virtual unsigned occurrencesInColumn(const unsigned column,
00301                                                          const double &value) const;
```

```
00302 };
00303
00304 } // namespace pmat
00305
00306 #endif
```

7.19 src/MatrixLowerTriangular.cpp File Reference

```
#include ¨MatrixLowerTriangular.h¨
#include ¨MatrixUpperTriangular.h¨
#include ¨utils.h¨
#include <random>
#include <stdexcept>
```

7.20 src/MatrixLowerTriangular.h File Reference

```
#include ¨MatrixTriangular.h¨
```

Classes

- class pmat::MatrixLowerTriangular

Namespaces

- namespace pmat

7.21 MatrixLowerTriangular.h

Go to the documentation of this file.

```
00001 #ifndef MATRIXLOWERTRIANGULAR_H
00002 #define MATRIXLOWERTRIANGULAR_H
00003 #pragma once
00004
00005 #include "MatrixTriangular.h"
00006
00007 namespace pmat {
00008
00009 class MatrixUpperTriangular;
00010
00011 class MatrixLowerTriangular : public pmat::MatrixTriangular {
00012    protected:
00013       [[nodiscard]] unsigned vectorIndex(const unsigned &i, const unsigned &j) const override;
00014
00015    public:
00016       MatrixLowerTriangular() = default;
00017       explicit MatrixLowerTriangular(const unsigned &size)
00018           : MatrixTriangular::MatrixTriangular(size){};
00019       MatrixLowerTriangular(const MatrixLowerTriangular &matrix)
00020           : MatrixTriangular::MatrixTriangular{std::move(matrix)} {}
00021       MatrixLowerTriangular(MatrixLowerTriangular &&matrix)
00022           : MatrixTriangular::MatrixTriangular{std::move(matrix)} {};
00023       MatrixLowerTriangular &operator=(const MatrixLowerTriangular &matrix) = default;
00024       MatrixLowerTriangular &operator=(MatrixLowerTriangular &&matrix) = default;
00025       ~MatrixLowerTriangular() override = default;
00026       double operator()(const unsigned &row, const unsigned &column) const override;
00027       [[nodiscard]] double dotProduct(const Matrix &matrix) const override;
00028       MatrixLowerTriangular operator+(const MatrixLowerTriangular &matrix) const;
```

```
00029       virtual void addBy(const MatrixLowerTriangular &matrix);
00030       MatrixLowerTriangular operator-(const MatrixLowerTriangular &matrix) const;
00031       virtual void subtractBy(const MatrixLowerTriangular &matrix);
00032       MatrixLowerTriangular operator*(const double &scalar) const;
00033       MatrixSquare operator*(const MatrixSquare &matrix) const;
00034       void multiplyBy(const double &scalar) override;
00035       MatrixSquare operator+(const MatrixSquare &matrix) const;
00036       MatrixSquare operator-(const MatrixSquare &matrix) const;
00037       MatrixSquare operator*(const MatrixTriangular &matrix) const;
00038       MatrixLowerTriangular operator*(const MatrixLowerTriangular &matrix) const;
00039       Vector operator*(const Vector &vector) const override;
00040
00046       [[nodiscard]] MatrixUpperTriangular getTranspose() const;
00047
00048       void swapRows(const unsigned &rowA, const unsigned &rowB, const unsigned &startColumn,
00049                     const unsigned &endColumn) override;
00050       void swapColumns(const unsigned &colA, const unsigned &colB, const unsigned &startRow,
00051                        const unsigned &endRow) override;
00052       void fillWithRandomValues(const double &min, const double &max) override;
00053
00059       [[nodiscard]] TriangType type() const override { return TriangType::LOWER; };
00060
00067       MatrixLowerTriangular inverse();
00068 };
00069
00070 } // namespace pmat
00071
00072 #endif
```

7.22 src/MatrixSkewSymmetric.cpp File Reference

```
#include ¨MatrixSkewSymmetric.h¨
#include ¨Messages.h¨
#include ¨utils.h¨
#include <random>
#include <stdexcept>
```

7.23 src/MatrixSkewSymmetric.h File Reference

```
#include ¨MatrixSymmetry.h¨
```

Classes

- class pmat::MatrixSkewSymmetric

Namespaces

- namespace pmat

7.24 MatrixSkewSymmetric.h

Go to the documentation of this file.

```
00001 #ifndef MATRIXSKEWSYMMETRIC_H
00002 #define MATRIXSKEWSYMMETRIC_H
00003
00004 #pragma once
00005
00006 #include "MatrixSymmetry.h"
00007
00008 namespace pmat {
00009
00010 class MatrixSkewSymmetric : public pmat::MatrixSymmetry {
00011
00012    public:
00013       MatrixSkewSymmetric() = default;
00014       explicit MatrixSkewSymmetric(const unsigned &size) : MatrixSymmetry::MatrixSymmetry(size){};
00015       MatrixSkewSymmetric(const MatrixSkewSymmetric &matrix)
00016           : MatrixSymmetry::MatrixSymmetry{std::move(matrix)} {}
00017       MatrixSkewSymmetric(MatrixSkewSymmetric &&matrix)
00018           : MatrixSymmetry::MatrixSymmetry{std::move(matrix)} {};
00019       MatrixSkewSymmetric &operator=(const MatrixSkewSymmetric &matrix) = default;
00020       MatrixSkewSymmetric &operator=(MatrixSkewSymmetric &&matrix) = default;
00021       ~MatrixSkewSymmetric() override = default;
00022       double operator()(const unsigned &row, const unsigned &column) const override;
00023       MatrixSkewSymmetric operator+(const MatrixSkewSymmetric &matrix) const;
00024       MatrixSquare operator+(const MatrixSymmetry &matrix) const;
00025       virtual void addBy(const MatrixSkewSymmetric &matrix);
00026       MatrixSkewSymmetric operator-(const MatrixSkewSymmetric &matrix) const;
00027       MatrixSquare operator-(const MatrixSymmetry &matrix) const;
00028       virtual void subtractBy(const MatrixSkewSymmetric &matrix);
00029       MatrixSkewSymmetric operator*(const double &scalar) const;
00030       MatrixSquare operator*(const MatrixSkewSymmetric &matrix) const;
00031       Vector operator*(const Vector &vector) const override;
00032       Matrix operator*(const Matrix &matrix) const override;
00033       void multiplyBy(const double &scalar) override;
00034       void transpose() override;
00035       void fillWithRandomValues(const double &min, const double &max) override;
00036 };
00037
00038 } // namespace pmat
00039 #endif
```

7.25 src/MatrixSquare.cpp File Reference

```
#include ¨MatrixSquare.h¨
#include ¨DecompositionPLU.h¨
#include ¨DecompositionPQR.h¨
#include ¨DecompositionSAS.h¨
#include ¨Messages.h¨
#include ¨utils.h¨
#include <stdexcept>
```

7.26 src/MatrixSquare.h File Reference

```
#include ¨Matrix.h¨
#include <memory>
```

Classes

- class pmat::MatrixSquare

Namespaces

- namespace pmat

Enumerations

- enum class pmat::SubMatrixPos { pmat::lower , pmat::upper }

7.27 MatrixSquare.h

Go to the documentation of this file.

```
00001 #ifndef MATRIXSQUARE_H
00002 #define MATRIXSQUARE_H
00003 #pragma once
00004
00005 #include "Matrix.h"
00006 #include <memory>
00007
00008 namespace pmat {
00009
00010 class DecompositionPLU;
00011 class DecompositionSAS;
00012 class DecompositionPQR;
00013
00014 enum class SubMatrixPos { lower, upper };
00015
00016 class MatrixSquare : public Matrix {
00017    public:
00018       MatrixSquare() = default;
00019       MatrixSquare(MatrixSquare &&matrix) noexcept = default;
00020       MatrixSquare(Matrix &&matrix);
00021       explicit MatrixSquare(const unsigned &size) : Matrix{size, size} {}
00022       MatrixSquare(const MatrixSquare &matrix);
00023       ~MatrixSquare() override = default;
00024       MatrixSquare &operator=(const MatrixSquare &matrix) = default;
00025       MatrixSquare &operator=(MatrixSquare &&matrix) noexcept = default;
00026       [[nodiscard]] unsigned size() const;
00027       virtual void resize(const unsigned &size);
00028       MatrixSquare operator+(const MatrixSquare &matrix) const;
00029       MatrixSquare operator-(const MatrixSquare &matrix) const;
00030       MatrixSquare operator*(const MatrixSquare &matrix) const;
00031       MatrixSquare operator*(const double &scalar) const;
00032       Vector operator*(const Vector &vector) const override;
00033       virtual void fillDiagonalWith(const double &value);
00034
00046       virtual MatrixSquare multiplyByBiggerMatrix(const MatrixSquare &matrix, SubMatrixPos pos);
00047
00053       [[nodiscard]] virtual double trace() const;
00054
00060       [[nodiscard]] DecompositionPLU decomposeToPLU() const;
00061
00067       [[nodiscard]] DecompositionSAS decomposeToSAS() const;
00068
00074       [[nodiscard]] DecompositionPQR decomposeToPQR() const;
00075 };
00076
00077 } // namespace pmat
00078
00079 #endif
```

7.28 src/MatrixSymmetric.cpp File Reference

```
#include ¨MatrixSymmetric.h¨
#include ¨DecompositionCholesky.h¨
#include <random>
#include <stdexcept>
```

7.29 src/MatrixSymmetric.h File Reference

```
#include ¨MatrixSymmetry.h¨
```

Classes

- class pmat::MatrixSymmetric

Namespaces

- namespace pmat

7.30 MatrixSymmetric.h

Go to the documentation of this file.

```
00001 #ifndef MATRIXSYMMETRIC_H
00002 #define MATRIXSYMMETRIC_H
00003 #pragma once
00004
00005 #include "MatrixSymmetry.h"
00006
00007 namespace pmat {
00008
00009 class DecompositionCholesky;
00010
00011 class MatrixSymmetric : public pmat::MatrixSymmetry {
00012
00013    public:
00014       MatrixSymmetric() = default;
00015       explicit MatrixSymmetric(const unsigned &size) : MatrixSymmetry::MatrixSymmetry(size){};
00016       MatrixSymmetric(const MatrixSymmetric &matrix) = default;
00017       MatrixSymmetric(MatrixSymmetric &&matrix)
00018           : MatrixSymmetry::MatrixSymmetry{std::move(matrix)} {};
00019       MatrixSymmetric &operator=(const MatrixSymmetric &matrix) = default;
00020       MatrixSymmetric &operator=(MatrixSymmetric &&matrix) = default;
00021       ~MatrixSymmetric() override = default;
00022       double operator()(const unsigned &row, const unsigned &column) const override;
00023       MatrixSymmetric operator+(const MatrixSymmetric &matrix) const;
00024       MatrixSquare operator+(const MatrixSymmetry &matrix) const;
00025       virtual void addBy(const MatrixSymmetric &matrix);
00026       MatrixSymmetric operator-(const MatrixSymmetric &matrix) const;
00027       MatrixSquare operator-(const MatrixSymmetry &matrix) const;
00028       virtual void subtractBy(const MatrixSymmetric &matrix);
00029       MatrixSymmetric operator*(const double &scalar) const;
00030       MatrixSquare operator*(const MatrixSymmetric &matrix) const;
00031       Matrix operator*(const Matrix &matrix) const override;
00032       Vector operator*(const Vector &vector) const override;
00033       void multiplyBy(const double &scalar) override;
00034       void transpose() override{};
00035       void fillWithRandomValues(const double &min, const double &max) override;
00036
00042       DecompositionCholesky decomposeToCholesky();
00043 };
00044
00045 } // namespace pmat
00046
00047 #endif
```

7.31 src/MatrixSymmetry.cpp File Reference

```
#include ¨MatrixSymmetry.h¨
```

7.32 src/MatrixSymmetry.h File Reference

```
#include ¨MatrixSquare.h¨
```

Classes

- class pmat::MatrixSymmetry

Namespaces

- namespace pmat

7.33 MatrixSymmetry.h

Go to the documentation of this file.

```
00001 #ifndef MATRIXSIMMETRY_H
00002 #define MATRIXSIMMETRY_H
00003 #pragma once
00004
00005 #include "MatrixSquare.h"
00006
00007 namespace pmat {
00008
00009 class MatrixSymmetry : public MatrixSquare {
00010    private:
00011       // The following functions are not valid for matrices with symmetry
00012       void swapRows(const unsigned &rowA, const unsigned &rowB, const unsigned &startColumn,
00013                     const unsigned &endColumn) override{};
00014       void swapColumns(const unsigned &columnA, const unsigned &columnB, const unsigned &startRow,
00015                        const unsigned &endRow) override{};
00016
00017    protected:
00018       [[nodiscard]] unsigned vectorIndex(const unsigned &i, const unsigned &j) const override;
00019
00020    public:
00021       MatrixSymmetry() = default;
00022       MatrixSymmetry(const MatrixSymmetry &matrix);
00023       MatrixSymmetry(MatrixSymmetry &&matrix) = default;
00024       explicit MatrixSymmetry(const unsigned &size) { this->initializeMembers(size, size, false); };
00025       ~MatrixSymmetry() override = default;
00026       MatrixSymmetry &operator=(const MatrixSymmetry &) = default;
00027       MatrixSymmetry &operator=(MatrixSymmetry &&) = default;
00028       [[nodiscard]] unsigned length() const override;
00029       double operator()(const unsigned &row, const unsigned &column) const override = 0;
00030       void transpose() override = 0;
00031       void fillWithRandomValues(const double &min, const double &max) override = 0;
00032 };
00033
00034 } // namespace pmat
00035
00036 #endif
```

7.34 src/MatrixTriangular.cpp File Reference

```
#include ¨MatrixTriangular.h¨
#include ¨utils.h¨
#include <stdexcept>
```

7.35 src/MatrixTriangular.h File Reference

```
#include ¨MatrixSquare.h¨
#include ¨Messages.h¨
```

Classes

- class pmat::MatrixTriangular

Namespaces

- namespace pmat

Enumerations

- enum class pmat::TriangType { pmat::UPPER , pmat::LOWER }

7.36 MatrixTriangular.h

Go to the documentation of this file.

```
00001 #include "MatrixSquare.h"
00002 #include "Messages.h"
00003
00004 #ifndef MATRIXTRIANGULAR_H
00005 #define MATRIXTRIANGULAR_H
00006 #pragma once
00007
00008 namespace pmat {
00009
00010 enum class TriangType { UPPER, LOWER };
00011
00012 class MatrixTriangular : public pmat::MatrixSquare {
00013    private:
00014       void transpose() override {}
00015
00016    protected:
00017       [[nodiscard]] unsigned vectorIndex(const unsigned &i, const unsigned &j) const override = 0;
00018
00019    public:
00020       MatrixTriangular() = default;
00021       MatrixTriangular(const MatrixTriangular &matrix);
00022       MatrixTriangular(MatrixTriangular &&matrix) = default;
00023       explicit MatrixTriangular(const unsigned &size) {
00024          this->initializeMembers(size, size, false);
00025       };
00026       ~MatrixTriangular() override = default;
00027       MatrixTriangular &operator=(const MatrixTriangular &) = default;
00028       MatrixTriangular &operator=(MatrixTriangular &&) = default;
00029       [[nodiscard]] unsigned length() const override;
00030       double operator()(const unsigned &row, const unsigned &column) const override = 0;
00031       [[nodiscard]] double dotProduct(const Matrix &matrix) const override = 0;
00032       MatrixSquare operator*(const MatrixTriangular &matrix) const;
00033       [[nodiscard]] MatrixSquare getSwappedByRows(const unsigned &rowIndexA,
00034                                                   const unsigned &rowIndexB) const;
00035       [[nodiscard]] MatrixSquare getSwappedByColumns(const unsigned &columnIndexA,
00036                                                      const unsigned &columnIndexB) const;
00037       void fillWithRandomValues(const double &min, const double &max) override = 0;
00038       void swapRows(const unsigned &rowA, const unsigned &rowB, const unsigned &startColumn,
00039                     const unsigned &endColumn) override = 0;
00040       void swapColumns(const unsigned &columnA, const unsigned &columnB, const unsigned &startRow,
00041                        const unsigned &endRow) override = 0;
00042       [[nodiscard]] virtual TriangType type() const = 0;
00043
00049       double determinant();
00050
00057       virtual bool isInvertible();
```

```
00058
00065       Vector linearSolve(const Vector &rhs);
00066
00073       static void findInverseByBackSubstitution(const MatrixTriangular &matrix,
00074                                                 MatrixTriangular &resp);
00075
00083       static pmat::Vector findSolutionByBackSubstitution(const MatrixTriangular &matrix,
00084                                                          const Vector &rhs);
00085 };
00086
00087 } // namespace pmat
00088 #endif
```

7.37 src/MatrixUpperTriangular.cpp File Reference

```
#include ¨MatrixUpperTriangular.h¨
#include ¨utils.h¨
#include <random>
#include <stdexcept>
```

7.38 src/MatrixUpperTriangular.h File Reference

```
#include ¨MatrixLowerTriangular.h¨
#include ¨MatrixTriangular.h¨
```

Classes

- class pmat::MatrixUpperTriangular

Namespaces

- namespace pmat

7.39 MatrixUpperTriangular.h

Go to the documentation of this file.

```
00001 #ifndef MATRIXUPPERTRIANGULAR_H
00002 #define MATRIXUPPERTRIANGULAR_H
00003 #pragma once
00004
00005 #include "MatrixLowerTriangular.h"
00006 #include "MatrixTriangular.h"
00007
00008 namespace pmat {
00009
00010 class MatrixUpperTriangular : public pmat::MatrixTriangular {
00011
00012    protected:
00013       [[nodiscard]] unsigned vectorIndex(const unsigned &i, const unsigned &j) const override;
00014
00015    public:
00016       MatrixUpperTriangular() = default;
00017       MatrixUpperTriangular(const MatrixUpperTriangular &matrix) = default;
00018       MatrixUpperTriangular(MatrixUpperTriangular &&matrix) = default;
00019       explicit MatrixUpperTriangular(const unsigned &size)
00020           : MatrixTriangular::MatrixTriangular(size){};
00021       ~MatrixUpperTriangular() override = default;
```

```
00022       MatrixUpperTriangular &operator=(const MatrixUpperTriangular &matrix) = default;
00023       MatrixUpperTriangular &operator=(MatrixUpperTriangular &&matrix) = default;
00024       double operator()(const unsigned &row, const unsigned &column) const override;
00025       [[nodiscard]] double dotProduct(const Matrix &matrix) const override;
00026       MatrixUpperTriangular operator+(const MatrixUpperTriangular &matrix) const;
00027       virtual void addBy(const MatrixUpperTriangular &matrix);
00028       MatrixUpperTriangular operator-(const MatrixUpperTriangular &matrix) const;
00029       virtual void subtractBy(const MatrixUpperTriangular &matrix);
00030       MatrixUpperTriangular operator*(const double &scalar) const;
00031       MatrixSquare operator*(const MatrixSquare &matrix) const;
00032       void multiplyBy(const double &scalar) override;
00033       MatrixSquare operator+(const MatrixSquare &matrix) const;
00034       MatrixSquare operator-(const MatrixSquare &matrix) const;
00035       Vector operator*(const Vector &vector) const override;
00036       MatrixUpperTriangular operator*(const MatrixUpperTriangular &matrix) const;
00037       MatrixSquare operator*(const MatrixTriangular &matrix) const;
00038
00044       [[nodiscard]] MatrixLowerTriangular getTranspose() const;
00045
00046       void swapRows(const unsigned &rowA, const unsigned &rowB, const unsigned &startColumn,
00047                     const unsigned &endColumn) override;
00048       void swapColumns(const unsigned &colA, const unsigned &colB, const unsigned &startRow,
00049                        const unsigned &endRow) override;
00050       void fillWithRandomValues(const double &min, const double &max) override;
00051
00057       [[nodiscard]] TriangType type() const override { return TriangType::UPPER; };
00058
00065       MatrixUpperTriangular inverse();
00066 };
00067
00068 } // namespace pmat
00069
00070 #endif
```

7.40 src/Messages.h File Reference

Namespaces

- namespace pmat
- namespace pmat::messages

Variables

- constexpr const char * pmat::messages::DATA_NOT_READ {"Error reading file data"}
- constexpr const char * pmat::messages::FILE_NOT_OPEN {"Error opening file"}
- constexpr const char * pmat::messages::INDEX_OUT {"Index out of bounds"}
- constexpr const char * pmat::messages::NONCOMPT_SIZE_ARG {"Argument is not compatible in size"}
- constexpr const char * pmat::messages::MATRIX_SINGULAR {"Matrix is singular"}
- constexpr const char * pmat::messages::MATRIX_NOT_LU {"Matrix not LU decomposable"}
- constexpr const char * pmat::messages::MATRIX_NOT_L {"Matrix not positive definite"}
- constexpr const char * pmat::messages::DECOMP_NOT_LU {"Calculation mode was not set to Strict LU"}

7.41 Messages.h

Go to the documentation of this file.

```
00001 #ifndef MESSAGES_H
00002 #define MESSAGES_H
00003 #pragma once
00004
00005 namespace pmat {
00006
00007 namespace messages {
00008
00009 constexpr const char *DATA_NOT_READ{"Error reading file data"};
00010 constexpr const char *FILE_NOT_OPEN{"Error opening file"};
00011
```

```
00012 constexpr const char *INDEX_OUT{"Index out of bounds"};
00013 constexpr const char *NONCOMPT_SIZE_ARG{"Argument is not compatible in size"};
00014
00015 constexpr const char *MATRIX_SINGULAR{"Matrix is singular"};
00016 constexpr const char *MATRIX_NOT_LU{"Matrix not LU decomposable"};
00017 constexpr const char *MATRIX_NOT_L{"Matrix not positive definite"};
00018 constexpr const char *DECOMP_NOT_LU{"Calculation mode was not set to Strict LU"};
00019
00020 } // namespace messages
00021
00022 } // namespace pmat
00023
00024 #endif
```

7.42 src/pmat_main.cpp File Reference

```
#include ¨MatrixSquare.h¨
#include ¨MatrixUpperTriangular.h¨
#include <iostream>
```

Functions

- int main ()

7.42.1 Function Documentation

7.42.1.1 main()

```
int main ( )
```

7.43 src/TMultiplicationManager.cpp File Reference

```
#include ¨TMultiplicationManager.h¨
#include ¨Messages.h¨
#include <future>
#include <memory>
#include <mutex>
```

7.44 src/TMultiplicationManager.h File Reference

```
#include <memory>
#include <mutex>
#include ¨Matrix.h¨
#include ¨TMultiplicationPerformer.h¨
#include <utility>
```

Classes

- class pmat::TMultiplicationManager

Namespaces

- namespace pmat

7.45 TMultiplicationManager.h

Go to the documentation of this file.

```
00001 #ifndef TOPERATIONMANAGER_H
00002 #define TOPERATIONMANAGER_H
00003 #include <memory>
00004 #include <mutex>
00005 #pragma once
00006
00007 #include "Matrix.h"
00008 #include "TMultiplicationPerformer.h"
00009 #include <utility>
00010
00011 namespace pmat {
00012
00013 class TMultiplicationManager {
00014    private:
00015       const Matrix *_operandFirst{nullptr};
00016       const Matrix *_operandSecond{nullptr};
00017       Matrix *_result{nullptr};
00018       std::mutex mtx1, mtx2;
00019       unsigned _lastRow{0};
00020       unsigned _lastColumn{0};
00021       std::vector<std::shared_ptr<TMultiplicationPerformer» _performers{};
00022
00023    public:
00024       TMultiplicationManager(const Matrix &operandFirst, const Matrix &operandSecond,
00025                              Matrix &result);
00026       TMultiplicationManager(const TMultiplicationManager &) = delete;
00027       TMultiplicationManager(TMultiplicationManager &&) = delete;
00028       TMultiplicationManager &operator=(const TMultiplicationManager &) = delete;
00029       TMultiplicationManager &operator=(TMultiplicationManager &&) = delete;
00030       ~TMultiplicationManager() = default;
00031
00032       [[nodiscard]] const Matrix &operandFirst() const { return *_operandFirst; }
00033       [[nodiscard]] const Matrix &operandSecond() const { return *_operandSecond; }
00034       void setResultValue(const double &value, const unsigned &row, const unsigned &column);
00035       bool getNextRowColumn(unsigned id);
00036       void multiply(int nThreads);
00037 };
00038
00039 } // namespace pmat
00040
00041 #endif
```

7.46 src/TMultiplicationPerformer.cpp File Reference

```
#include ¨TMultiplicationPerformer.h¨
#include ¨TMultiplicationManager.h¨
#include ¨utils.h¨
```

7.47 src/TMultiplicationPerformer.h File Reference

Classes

- class pmat::TMultiplicationPerformer

Namespaces

- namespace pmat

7.48 TMultiplicationPerformer.h

Go to the documentation of this file.

```
00001 #ifndef TOPERATIONPERFORMER_H
00002 #define TOPERATIONPERFORMER_H
00003
00004 #pragma once
00005
00006 namespace pmat {
00007
00008 class TMultiplicationManager;
00009
00010 class TMultiplicationPerformer {
00011    private:
00012       unsigned _id;
00013       TMultiplicationManager *_manager{nullptr};
00014       unsigned _row{0};
00015       unsigned _column{0};
00016
00017    public:
00018       TMultiplicationPerformer(unsigned id, TMultiplicationManager &manager)
00019           : _id{id}, _manager{&manager} {}
00020       TMultiplicationPerformer(const TMultiplicationPerformer &) = default;
00021       TMultiplicationPerformer(TMultiplicationPerformer &&) = default;
00022       TMultiplicationPerformer &operator=(const TMultiplicationPerformer &) = default;
00023       TMultiplicationPerformer &operator=(TMultiplicationPerformer &&) = default;
00024       ~TMultiplicationPerformer() = default;
00025
00026       void start();
00027       void setRowColumn(const unsigned &row, const unsigned &column);
00028 };
00029
00030 } // namespace pmat
00031
00032 #endif
```

7.49 src/utils.h File Reference

Namespaces

- namespace pmat
- namespace pmat::utils

7.50 utils.h

Go to the documentation of this file.

```
00001 #pragma once
00002
00003 namespace pmat {
00004 namespace utils {
00005
00006 // Tolerance for relational operators with doubles
00007 static const double DIF_TOLERANCE = 0.0000000001;
00008
00009 static const double ZERO = 0.0000000000;
00010
00011 static const double ONE = 1.0000000000;
00012
00013 static const double TWO = 2.0000000000;
00014
00015 static const double MINUS_ONE = -1.0000000000;
00016
00017 static const double ONE_HALF = 0.5000000000;
```

```
00018
00019 static const unsigned NUM_THREADS = 5;
00020
00021 static const double &max(const double &a, const double &b) {
00022    return a > b ? a : b;
00023 };
00024
00025 static inline double abs(const double &a) {
00026    return a > ZERO ? a : -a;
00027 };
00028
00029 static inline bool areEqual(const double &a, const double &b) {
00030    const double m = max(abs(a), abs(b));
00031    return m < DIF_TOLERANCE ? true : (abs(a - b) / m) < DIF_TOLERANCE;
00032 }
00033
00034 static inline bool isZero(const double &a) {
00035    return areEqual(a, ZERO);
00036 }
00037
00038 static inline bool isOne(const double &a) {
00039    return areEqual(a, ONE);
00040 }
00041
00042 static inline double signOf(const double &a) {
00043    return a < 0 ? -ONE : ONE;
00044 }
00045
00046 } // namespace utils
00047 } // namespace pmat
```

7.51 src/Vector.cpp File Reference

```
#include ¨Vector.h¨
#include ¨Matrix.h¨
#include ¨Messages.h¨
#include ¨utils.h¨
#include <random>
#include <stdexcept>
#include <utility>
```

7.52 src/Vector.h File Reference

```
#include ¨Array.h¨
#include <vector>
```

Classes

- class pmat::Vector

  Entity for one-dimensional array.

Namespaces

- namespace pmat

7.53 Vector.h

Go to the documentation of this file.

```
00001 #ifndef VECTOR_H
00002 #define VECTOR_H
00003 #pragma once
00004
00005 #include "Array.h"
00006 #include <vector>
00007
00008 // class Matrix;
00009
00010 namespace pmat {
00011
00012 class Matrix;
00013
00018 class Vector : public Array {
00019    private:
00020       std::vector<double> _vector{};
00021
00022    public:
00023       Vector() = default;
00024       Vector(unsigned size) : _vector(size, 0.){};
00025       Vector(const Vector &vector);
00026       Vector(Vector &&vector) noexcept : _vector{std::move(vector._vector)} {};
00027       ~Vector() override = default;
00028       [[nodiscard]] unsigned length() const override { return _vector.size(); };
00029       [[nodiscard]] unsigned dimension() const override { return 1; };
00030
00036       [[nodiscard]] unsigned size() const { return _vector.size(); }
00037
00043       void resize(const unsigned &size);
00044
00045       void clear() override { _vector.clear(); };
00046       void emplaceBack(const double &value);
00047
00054       void setValue(const double &value, const unsigned &index);
00055
00062       const double &operator()(const unsigned &index) const;
00063
00064       Vector &operator=(const Vector &vector);
00065       Vector &operator=(Vector &&vector) noexcept;
00066       bool operator==(const Vector &vector) const;
00067
00074       Vector operator+(const Vector &vector) const;
00075
00081       void addBy(const Vector &vector);
00082
00089       Vector operator-(const Vector &vector) const;
00090
00096       void subtractBy(const Vector &vector);
00097
00104       Vector operator*(const double &scalar) const;
00105
00111       void multiplyBy(const double &scalar);
00112
00123       [[nodiscard]] double dotProduct(const Vector &vector) const;
00124
00131       [[nodiscard]] double frobeniusNorm() const;
00132
00138       [[nodiscard]] Vector getUnitaryVector() const;
00139       [[nodiscard]] unsigned occurrences(const double &value) const override;
00140       void fillWithRandomValues(const double &min, const double &max) override;
00141
00149       void swapElements(const unsigned &elmIndexA, const unsigned &elmIndexB);
00150
00155       void ascendingSort();
00156
00161       void descendingSort();
00162
00168       [[nodiscard]] Matrix toColumnMatrix() const;
00169
00175       [[nodiscard]] Matrix toRowMatrix() const;
00176 };
00177
00178 } // namespace pmat
00179 #endif
```

Index