

MatrixLibrary

1.0

Generated by Doxygen 1.9.8

1 MatrixLibrary	1
1.1 Introduction	1
1.2 Features	1
2 Hierarchical Index	3
2.1 Class Hierarchy	3
3 Class Index	5
3.1 Class List	5
4 File Index	7
4.1 File List	7
5 Class Documentation	9
5.1 EigsymResult Struct Reference	9
5.1.1 Detailed Description	9
5.1.2 Member Data Documentation	9
5.1.2.1 eigenvalues	9
5.1.2.2 eigenvectors	9
5.2 InvalidMatrixSize Class Reference	10
5.2.1 Detailed Description	10
5.2.2 Constructor & Destructor Documentation	10
5.2.2.1 InvalidMatrixSize() [1/2]	10
5.2.2.2 InvalidMatrixSize() [2/2]	10
5.3 Matrix Class Reference	10
5.3.1 Detailed Description	12
5.3.2 Constructor & Destructor Documentation	12
5.3.2.1 Matrix() [1/4]	12
5.3.2.2 Matrix() [2/4]	12
5.3.2.3 Matrix() [3/4]	12
5.3.2.4 Matrix() [4/4]	12
5.3.3 Member Function Documentation	12
5.3.3.1 diagmat() [1/2]	12
5.3.3.2 diagmat() [2/2]	13
5.3.3.3 eigsym()	13
5.3.3.4 get_data()	13
5.3.3.5 get_num_cols()	13
5.3.3.6 get_num_rows()	13
5.3.3.7 get_size()	13
5.3.3.8 householder_tridiagonalize()	13
5.3.3.9 Identity()	13
5.3.3.10 is_symmetric()	13
5.3.3.11 Ones()	14
5.3.3.12 operator()() [1/2]	14

5.3.3.13 operator() [2/2]	14
5.3.3.14 operator*() [1/2]	14
5.3.3.15 operator*() [2/2]	14
5.3.3.16 operator+()	15
5.3.3.17 operator-()	15
5.3.3.18 operator==()	15
5.3.3.19 QL()	15
5.3.3.20 Random()	15
5.3.3.21 save_hdf5() [1/2]	15
5.3.3.22 save_hdf5() [2/2]	15
5.3.3.23 transpose()	15
5.3.3.24 Zeros()	16
5.3.4 Member Data Documentation	16
5.3.4.1 matrix	16
5.3.4.2 num_cols	16
5.3.4.3 num_rows	16
5.3.4.4 size	16
5.4 QLEigenResult Struct Reference	16
5.4.1 Detailed Description	17
5.4.2 Member Data Documentation	17
5.4.2.1 eigenvalues	17
5.4.2.2 Q_ql	17
5.5 TridiagonalResult Struct Reference	17
5.5.1 Detailed Description	17
5.5.2 Member Data Documentation	17
5.5.2.1 d	17
5.5.2.2 e	17
5.5.2.3 Q_house	17
6 File Documentation	19
6.1 include/custom_exception.hpp File Reference	19
6.2 custom_exception.hpp	19
6.3 src/custom_exception.hpp File Reference	19
6.4 custom_exception.hpp	20
6.5 include/helper_func.hpp File Reference	20
6.5.1 Function Documentation	20
6.5.1.1 pythag()	20
6.5.1.2 SIGN()	20
6.6 helper_func.hpp	20
6.7 include/matrix.h File Reference	21
6.7.1 Typedef Documentation	21
6.7.1.1 vec	21

6.7.2 Function Documentation	21
6.7.2.1 operator<<()	21
6.8 matrix.h	22
6.9 include/print_vec.hpp File Reference	23
6.9.1 Function Documentation	23
6.9.1.1 operator<<()	23
6.10 print_vec.hpp	23
6.11 src/benchmark.hpp File Reference	23
6.12 benchmark.hpp	24
6.13 src/helper_func.cpp File Reference	24
6.13.1 Function Documentation	24
6.13.1.1 pythag()	24
6.13.1.2 SIGN()	25
6.14 helper_func.cpp	25
6.15 src/matrix_basic_func.cpp File Reference	25
6.16 src/matrix_eigendecompp.cpp File Reference	25
6.17 src/matrix_operators.cpp File Reference	26
6.17.1 Function Documentation	26
6.17.1.1 operator<<()	26
6.18 src/matrix_utilfuncs.cpp File Reference	26
Index	27

Chapter 1

MatrixLibrary

1.1 Introduction

MatrixLibrary is a matrix library with basic linear algebra functionality including eigenvalue decompositions for symmetric matrices.

1.2 Features

- [Matrix](#) arithmetic
- Householder tridiagonalization
- QL eigenvalue solver
- Symmetry Check
- Transpose
- HDF5 output

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

std::domain_error	10
InvalidMatrixSize	10
InvalidMatrixSize	10
EigsymResult	9
Matrix	10
QLEigenResult	16
TridiagonalResult	17

Chapter 3

Class Index

3.1 Class List

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

EigsymResult	
Final eigenvalues and eigenvectors of matrix	9
InvalidMatrixSize	
Exception thrown this error if the matrix size is invalid for specified operation	10
Matrix	
Matrix class with basic linear algebra operations	10
QLEigenResult	
Result of QL Algorithm	16
TridiagonalResult	
Result of Householder Tridiagonalization	17

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

include/custom_exception.hpp	19
include/helper_func.hpp	20
include/matrix.h	21
include/print_vec.hpp	23
src/benchmark.hpp	23
src/custom_exception.hpp	19
src/helper_func.cpp	24
src/matrix_basic_func.cpp	25
src/matrix_eigendecompp.cpp	25
src/matrix_operators.cpp	26
src/matrix_utilfuncs.cpp	26

Chapter 5

Class Documentation

5.1 EigsymResult Struct Reference

Final eigenvalues and eigenvectors of matrix.

```
#include <matrix.h>
```

Collaboration diagram for EigsymResult:

Public Attributes

- `vec eigenvalues`
- `Matrix eigenvectors`

5.1.1 Detailed Description

Final eigenvalues and eigenvectors of matrix.

This code is adapted from Numerical Recipes

5.1.2 Member Data Documentation

5.1.2.1 eigenvalues

```
vec EigsymResult::eigenvalues
```

5.1.2.2 eigenvectors

```
Matrix EigsymResult::eigenvectors
```

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

- `include/matrix.h`

5.2 InvalidMatrixSize Class Reference

Exception thrown this error if the matrix size is invalid for specified operation.

```
#include <custom_exception.hpp>
```

Inheritance diagram for InvalidMatrixSize:

Collaboration diagram for InvalidMatrixSize:

Public Member Functions

- [InvalidMatrixSize \(const std::string &what_arg\)](#)
- [InvalidMatrixSize \(const std::string &what_arg\)](#)

5.2.1 Detailed Description

Exception thrown this error if the matrix size is invalid for specified operation.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 InvalidMatrixSize() [1/2]

```
InvalidMatrixSize::InvalidMatrixSize (
    const std::string & what_arg ) [inline]
```

5.2.2.2 InvalidMatrixSize() [2/2]

```
InvalidMatrixSize::InvalidMatrixSize (
    const std::string & what_arg ) [inline]
```

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

- [include/custom_exception.hpp](#)
- [src/custom_exception.hpp](#)

5.3 Matrix Class Reference

[Matrix](#) class with basic linear algebra operations.

```
#include <matrix.h>
```

Public Member Functions

- **Matrix** (int rows, int cols)
*Construct a matrix of size (row * cols)*
- **Matrix** ()
- **Matrix** (const **vec** &values, int rows, int cols)
*Construct a matrix of size (row * cols) with data.*
- **Matrix** (**vec** &&values, int rows, int cols)
- int **get_num_rows** () const
- int **get_num_cols** () const
- int **get_size** () const
- const **vec** & **get_data** () const
- double & **operator()** (int x, int y)
Access a matrix element at (row, col)
- const double & **operator()** (int x, int y) const
Const reference to the matrix element at (row, col)
- bool **operator==** (const **Matrix** &other) const
- **Matrix operator+** (const **Matrix** &other) const
- **Matrix operator-** (const **Matrix** &other) const
- **Matrix operator*** (const **Matrix** &other) const
- **Matrix operator*** (double s) const
- bool **is_symmetric** (double tol) const
- **Matrix transpose** () const
- **TridiagonalResult** **householder_tridiagonalize** (bool yesvecs=true) const
- **QLEigenResult** **QL** (**vec** d, **vec** e) const
- **EigsymResult** **eigsym** () const

Static Public Member Functions

- static **Matrix Ones** (int rows, int cols)
Fill matrix of specified size with 1's.
- static **Matrix Zeros** (int rows, int cols)
Fill matrix of specified size with 0's.
- static **Matrix Random** (int rows, int cols)
Fill matrix of specific size with random doubles from [0.0, 1.0]
- static **Matrix Identity** (int n)
*Create square identity matrix with dimensions n*n.*
- static **Matrix diagmat** (const **vec** &vector)
- static **Matrix diagmat** (const **Matrix** &mat)
- static void **save_hdf5** (const **Matrix** &data, const std::string &filename, const std::string &dataset_name)
- static void **save_hdf5** (const **vec** &data, const std::string &filename, const std::string &dataset_name)

Private Attributes

- int **num_rows**
- int **num_cols**
- int **size**
- **vec matrix**

5.3.1 Detailed Description

`Matrix` class with basic linear algebra operations.

The matrix class stores elements in contiguous memory to improve cache locality This class supports multiplication, addition, subtraction, symmetry checks, transposing matrix, saving to HDF5 file and eigenvalue decompositions for symmetric matrices.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 `Matrix()` [1/4]

```
Matrix::Matrix (
    int rows,
    int cols )
```

Construct a matrix of size (row * cols)

5.3.2.2 `Matrix()` [2/4]

```
Matrix::Matrix ( )
```

5.3.2.3 `Matrix()` [3/4]

```
Matrix::Matrix (
    const vec & values,
    int rows,
    int cols )
```

Construct a matrix of size (row * cols) with data.

5.3.2.4 `Matrix()` [4/4]

```
Matrix::Matrix (
    vec && values,
    int rows,
    int cols )
```

5.3.3 Member Function Documentation

5.3.3.1 `diagmat()` [1/2]

```
Matrix Matrix::diagmat (
    const Matrix & mat ) [static]
```

5.3.3.2 diagmat() [2/2]

```
Matrix Matrix::diagmat (
    const vec & vector ) [static]
```

5.3.3.3 eigsym()

```
EigsymResult Matrix::eigsym ( ) const
```

5.3.3.4 get_data()

```
const vec & Matrix::get_data ( ) const
```

5.3.3.5 get_num_cols()

```
int Matrix::get_num_cols ( ) const
```

5.3.3.6 get_num_rows()

```
int Matrix::get_num_rows ( ) const
```

5.3.3.7 get_size()

```
int Matrix::get_size ( ) const
```

5.3.3.8 householder_tridiagonalize()

```
TridiagonalResult Matrix::householder_tridiagonalize (
    bool yesvecs = true ) const
```

5.3.3.9 Identity()

```
Matrix Matrix::Identity (
    int n ) [static]
```

Create square identity matrix with dimensions n*n.

5.3.3.10 is_symmetric()

```
bool Matrix::is_symmetric (
    double tol = 1e-12 ) const
```

5.3.3.11 `Ones()`

```
Matrix Matrix::Ones (
    int rows,
    int cols ) [static]
```

Fill matrix of specified size with 1's.

5.3.3.12 `operator()()` [1/2]

```
double & Matrix::operator() (
    int x,
    int y )
```

Access a matrix element at (row, col)

Parameters

<code>x</code>	row index (0-based)
<code>y</code>	column index (0-based)

Returns

reference to the matrix element

Exceptions

<i>OutOfBounds</i>	exception if the indices are invalid
--------------------	--------------------------------------

5.3.3.13 `operator()()` [2/2]

```
const double & Matrix::operator() (
    int x,
    int y ) const
```

Const reference to the matrix element at (row, col)

5.3.3.14 `operator*()` [1/2]

```
Matrix Matrix::operator* (
    const Matrix & other ) const
```

5.3.3.15 `operator*()` [2/2]

```
Matrix Matrix::operator* (
    double s ) const
```

5.3.3.16 operator+()

```
Matrix Matrix::operator+ (
    const Matrix & other ) const
```

5.3.3.17 operator-()

```
Matrix Matrix::operator- (
    const Matrix & other ) const
```

5.3.3.18 operator==()

```
bool Matrix::operator== (
    const Matrix & other ) const
```

5.3.3.19 QL()

```
QLEigenResult Matrix::QL (
    vec d,
    vec e ) const
```

5.3.3.20 Random()

```
Matrix Matrix::Random (
    int rows,
    int cols ) [static]
```

Fill matrix of specific size with random doubles from [0.0, 1.0)

5.3.3.21 save_hdf5() [1/2]

```
void Matrix::save_hdf5 (
    const Matrix & data,
    const std::string & filename,
    const std::string & dataset_name ) [static]
```

5.3.3.22 save_hdf5() [2/2]

```
void Matrix::save_hdf5 (
    const vec & data,
    const std::string & filename,
    const std::string & dataset_name ) [static]
```

5.3.3.23 transpose()

```
Matrix Matrix::transpose ( ) const
```

5.3.3.24 Zeros()

```
Matrix Matrix::Zeros (
    int rows,
    int cols ) [static]
```

Fill matrix of specified size with 0's.

5.3.4 Member Data Documentation

5.3.4.1 matrix

```
vec Matrix::matrix [private]
```

5.3.4.2 num_cols

```
int Matrix::num_cols [private]
```

5.3.4.3 num_rows

```
int Matrix::num_rows [private]
```

5.3.4.4 size

```
int Matrix::size [private]
```

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

- include/matrix.h
- src/matrix_basic_func.cpp
- src/matrix_eigendecompp.cpp
- src/matrix_operators.cpp
- src/matrix_utilfuncs.cpp

5.4 QLEigenResult Struct Reference

Result of QL Algorithm.

```
#include <matrix.h>
```

Collaboration diagram for QLEigenResult:

Public Attributes

- vec eigenvalues
- Matrix Q_ql

5.4.1 Detailed Description

Result of QL Algorithm.

This code is adapted from Numerical Recipes

5.4.2 Member Data Documentation

5.4.2.1 eigenvalues

```
vec QLEigenResult::eigenvalues
```

5.4.2.2 Q_ql

```
Matrix QLEigenResult::Q_ql
```

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

- [include/matrix.h](#)

5.5 TridiagonalResult Struct Reference

Result of Householder Tridiagonalization.

```
#include <matrix.h>
```

Collaboration diagram for TridiagonalResult:

Public Attributes

- [vec d](#)
- [vec e](#)
- [Matrix Q_house](#)

5.5.1 Detailed Description

Result of Householder Tridiagonalization.

This code is adapted from Numerical Recipes

5.5.2 Member Data Documentation

5.5.2.1 d

```
vec TridiagonalResult::d
```

5.5.2.2 e

```
vec TridiagonalResult::e
```

5.5.2.3 Q_house

```
Matrix TridiagonalResult::Q_house
```

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

- [include/matrix.h](#)

Chapter 6

File Documentation

6.1 include/custom_exception.hpp File Reference

```
#include <stdexcept>
```

Include dependency graph for custom_exception.hpp: This graph shows which files directly or indirectly include this file:

Classes

- class [InvalidMatrixSize](#)

Exception thrown this error if the matrix size is invalid for specified operation.

6.2 custom_exception.hpp

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include <stdexcept>
00003
00008 class InvalidMatrixSize : public std::domain_error {
00009 public:
0010     InvalidMatrixSize(const std::string& what_arg) : std::domain_error(what_arg)
0011     {}
0012 };
```

6.3 src/custom_exception.hpp File Reference

```
#include <stdexcept>
```

Include dependency graph for custom_exception.hpp:

Classes

- class [InvalidMatrixSize](#)

Exception thrown this error if the matrix size is invalid for specified operation.

6.4 custom_exception.hpp

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include <stdexcept>
00003
00004 class InvalidMatrixSize : public std::domain_error {
00005 public:
00006     InvalidMatrixSize(const std::string& what_arg) : std::domain_error(what_arg)
00007     {}
00008 };
```

6.5 include/helper_func.hpp File Reference

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

Functions

- double **SIGN** (double a, double b)
*Return a value equivalent to |a| * sign(b)*
- double **pythag** (double a, double b)
Computes sqrt(a^2 + b^2) without destructive overflow or underflow.

6.5.1 Function Documentation

6.5.1.1 pythag()

```
double pythag (
    double a,
    double b )
```

Computes $\sqrt{a^2 + b^2}$ without destructive overflow or underflow.

Helper function for eigenvalue and eigenvector calculation. Adapted from Numerical Recipes.

6.5.1.2 SIGN()

```
double SIGN (
    double a,
    double b )
```

Return a value equivalent to $|a| * \text{sign}(b)$

Helper function for eigenvalue and eigenvector calculation. Adapted from Numerical Recipes.

6.6 helper_func.hpp

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00009 double SIGN(double a, double b);
00010
00017 double pythag(double a, double b);
```

6.7 include/matrix.h File Reference

```
#include <iostream>
#include <vector>
#include <limits>
#include "custom_exception.hpp"
#include "helper_func.hpp"
#include <cmath>
#include <iomanip>
#include <stdexcept>
#include <random>
```

Include dependency graph for matrix.h: This graph shows which files directly or indirectly include this file:

Classes

- class [Matrix](#)
Matrix class with basic linear algebra operations.
- struct [TridiagonalResult](#)
Result of Householder Tridiagonalization.
- struct [QLEigenResult](#)
Result of QL Algorithm.
- struct [EigsymResult](#)
Final eigenvalues and eigenvectors of matrix.

Typedefs

- typedef std::vector< double > [vec](#)
Alias for vector of doubles, used for matrix storage.

Functions

- std::ostream & [operator<<](#) (std::ostream &out, const [Matrix](#) &M)

6.7.1 Typedef Documentation

6.7.1.1 [vec](#)

```
typedef std::vector<double> vec
```

Alias for vector of doubles, used for matrix storage.

6.7.2 Function Documentation

6.7.2.1 [operator<<\(\)](#)

```
std::ostream & operator<< (
    std::ostream & out,
    const Matrix & M )
```

6.8 matrix.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include <iostream>
00004 #include <vector>
00005 #include <limits>
00006 #include "custom_exception.hpp"
00007 #include "helper_func.hpp" // numerical recipes helper functions
00008 #include <cmath>
00009 #include <iomanip>
00010 #include <stdexcept>
00011 #include <random>
00012
00013 typedef std::vector<double> vec;
00014
00015 // Forward declarations of global result types
00016 struct TridiagonalResult;
00017 struct EigsymResult;
00018 struct QLEigenResult;
00019
00020 class Matrix {
00021     private:
00022     int num_rows;
00023     int num_cols;
00024     int size;
00025     vec matrix;
00026
00027     public:
00028     // === Constructors ===
00029     Matrix(int rows, int cols);
00030     Matrix();
00031     Matrix(const vec& values, int rows, int cols);
00032     Matrix(vec&& values, int rows, int cols);
00033
00034     // === Factory Methods ===
00035     static Matrix Ones(int rows, int cols);
00036
00037     static Matrix Zeros(int rows, int cols);
00038
00039     static Matrix Random(int rows, int cols);
00040
00041     static Matrix Identity(int n);
00042
00043     // === Accessors ===
00044     int get_num_rows() const;
00045     int get_num_cols() const;
00046     int get_size() const;
00047     const vec& get_data() const;
00048
00049     // === Operators ===
00050     double& operator()(int x, int y);
00051
00052     const double& operator()(int x, int y) const;
00053
00054     bool operator==(const Matrix& other) const;
00055     Matrix operator+(const Matrix& other) const;
00056     Matrix operator-(const Matrix& other) const;
00057     Matrix operator*(const Matrix& other) const;
00058     Matrix operator*(double s) const;
00059
00060     // === Linear Algebra Functionality ===
00061     static Matrix diagmat(const vec& vector);
00062     static Matrix diagmat(const Matrix& mat);
00063     bool is_symmetric(double tol) const;
00064     Matrix transpose() const;
00065     TridiagonalResult householder_tridiagonalize(bool yesvecs = true) const;
00066     QLEigenResult QL(vec d, vec e) const;
00067     EigsymResult eigsym() const;
00068
00069     // === Saving to HDF5 File ===
00070     static void save_hdf5(const Matrix& data, const std::string& filename, const std::string&
00071     dataset_name);
00072     static void save_hdf5(const vec& data, const std::string& filename, const std::string&
00073     dataset_name);
00074
00075 };
00076
00077 struct TridiagonalResult {
00078     vec d; // diagonal elements
00079     vec e; // off-diagonal elements
00080     Matrix Q_house; // orthogonal transformation matrix
00081 };
00082

```

```

00150 struct QLEigenResult {
00151     vec eigenvalues;
00152     Matrix Q_ql;
00153 };
00154
00160 struct EigsymResult {
00161     vec eigenvalues;
00162     Matrix eigenvectors;
00163 };
00164
00165 // === Printing Functionality ===
00166 std::ostream& operator<<(std::ostream& out, const Matrix & M);

```

6.9 include/print_vec.hpp File Reference

```

#include <iostream>
#include <vector>
#include <iomanip>
Include dependency graph for print_vec.hpp:

```

Functions

- std::ostream & **operator<<** (std::ostream &out, const std::vector< double > &v)
Overload print operator for vector.

6.9.1 Function Documentation

6.9.1.1 operator<<()

```

std::ostream & operator<< (
    std::ostream & out,
    const std::vector< double > & v ) [inline]

```

Overload print operator for vector.

6.10 print_vec.hpp

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002 #include <iostream>
00003 #include <vector>
00004 #include <iomanip>
00005
00009 inline std::ostream& operator<<(std::ostream& out, const std::vector<double>& v) {
00010     out << std::fixed << std::setprecision(4);
00011     for (double x : v) {
00012         out << x << "\n";
00013     }
00014     return out;
00015 }

```

6.11 src/benchmark.hpp File Reference

```

#include <vector>
#include <chrono>
Include dependency graph for benchmark.hpp:

```

6.12 benchmark.hpp

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include <vector>
00004 #include <chrono>
00005
00006 template <typename T>
00007 long benchmark_time(T func, int iterations=25) {
00008     std::vector<double> vec_diff;
00009     vec_diff.reserve(iterations);
00010     for (int i = 0; i < iterations; ++i) {
00011         auto start = std::chrono::steady_clock::now();
00012         func();
00013         auto end = std::chrono::steady_clock::now();
00014
00015         // Convert to nanoseconds
00016         double diff = std::chrono::duration<double, std::nano>(end - start).count();
00017         vec_diff.push_back(diff);
00018     }
00019
00020     // get average
00021     double avg = 0.0;
00022     for (double val : vec_diff) {
00023         avg += val;
00024     }
00025     // return time diff in microseconds
00026     return avg / vec_diff.size();
00027 }
00028
00029 // need to implement L2 norm for this benchmarking
00030 template <typename T>
00031 long benchmark_acc(T func1, T func2) {
00032     auto diff = func1() - func2();
00033     return diff;
00034 }
```

6.13 src/helper_func.cpp File Reference

```
#include "helper_func.hpp"
#include <cmath>
```

Include dependency graph for helper_func.cpp: This graph shows which files directly or indirectly include this file:

Functions

- double **SIGN** (double a, double b)
*Return a value equivalent to $|a| * sign(b)$*
- double **pythag** (const double a, const double b)
Computes $\sqrt{a^2 + b^2}$ without destructive overflow or underflow.

6.13.1 Function Documentation

6.13.1.1 pythag()

```
double pythag (
    double a,
    double b )
```

Computes $\sqrt{a^2 + b^2}$ without destructive overflow or underflow.

Helper function for eigenvalue and eigenvector calculation. Adapted from Numerical Recipes.

6.13.1.2 SIGN()

```
double SIGN (
    double a,
    double b )
```

Return a value equivalent to $|a| * \text{sign}(b)$

Helper function for eigenvalue and eigenvector calculation. Adapted from Numerical Recipes.

6.14 helper_func.cpp

[Go to the documentation of this file.](#)

```
00001 #include "helper_func.hpp"
00002 #include <cmath>
00003
00004 double SIGN(double a, double b) {
00005     if ( b >= 0.0 ) {
00006         return std::fabs(a);
00007     } else {
00008         return -std::fabs(a);
00009     }
00010 }
00011
00012 double pythag(const double a, const double b) {
00013     // computes sqrt(a^2 + b^2) without destructive underflow or overflow
00014     double absa = std::abs(a);
00015     double absb = std::abs(b);
00016     if (absa > absb) {
00017         return absa * sqrt(1.0 + (absb/absa)*(absb/absa));
00018     } else {
00019         return absb * sqrt(1.0 + (absa/absb)*(absa/absb));
00020     }
00021 }
```

6.15 src/matrix_basic_func.cpp File Reference

```
#include "matrix.h"
#include <algorithm>
Include dependency graph for matrix_basic_func.cpp:
```

6.16 src/matrix_eigendecompp.cpp File Reference

```
#include "matrix.h"
#include "helper_func.cpp"
#include <cmath>
#include <stdexcept>
#include <limits>
#include <algorithm>
#include <numeric>
Include dependency graph for matrix_eigendecompp.cpp:
```

6.17 src/matrix_operators.cpp File Reference

```
#include "matrix.h"
#include <algorithm>
#include <iomanip>
#include <cmath>
#include <limits>
Include dependency graph for matrix_operators.cpp:
```

Functions

- std::ostream & **operator<<** (std::ostream &out, const **Matrix** &M)

6.17.1 Function Documentation

6.17.1.1 **operator<<()**

```
std::ostream & operator<< (
    std::ostream & out,
    const Matrix & M )
```

6.18 src/matrix_utilfuncs.cpp File Reference

```
#include "matrix.h"
#include <cmath>
#include <highfive/H5File.hpp>
Include dependency graph for matrix_utilfuncs.cpp:
```

Index

d
 TridiagonalResult, 17
diagmat
 Matrix, 12

e
 TridiagonalResult, 17
eigenvalues
 EigsymResult, 9
 QLEigenResult, 17
eigenvectors
 EigsymResult, 9
eigsym
 Matrix, 13
EigsymResult, 9
 eigenvalues, 9
 eigenvectors, 9

get_data
 Matrix, 13
get_num_cols
 Matrix, 13
get_num_rows
 Matrix, 13
get_size
 Matrix, 13

helper_func.cpp
 pythag, 24
 SIGN, 24
helper_func.hpp
 pythag, 20
 SIGN, 20
householder_tridiagonalize
 Matrix, 13

Identity
 Matrix, 13
include/custom_exception.hpp, 19
include/helper_func.hpp, 20
include/matrix.h, 21, 22
include/print_vec.hpp, 23
InvalidMatrixSize, 10
 InvalidMatrixSize, 10
is_symmetric
 Matrix, 13

Matrix, 10
 diagmat, 12
 eigsym, 13
 get_data, 13

 get_num_cols, 13
 get_num_rows, 13
 get_size, 13
 householder_tridiagonalize, 13
 Identity, 13
 is_symmetric, 13
 Matrix, 12
 matrix, 16
 num_cols, 16
 num_rows, 16
 Ones, 13
 operator(), 14
 operator+, 14
 operator-, 15
 operator==, 15
 operator*, 14
 QL, 15
 Random, 15
 save_hdf5, 15
 size, 16
 transpose, 15
 Zeros, 15

matrix

matrix.h
 operator<<, 21
 vec, 21
matrix_operators.cpp
 operator<<, 26

MatrixLibrary, 1

 num_cols
 Matrix, 16
 num_rows
 Matrix, 16

Ones
 Matrix, 13

operator<<
 matrix.h, 21
 matrix_operators.cpp, 26
 print_vec.hpp, 23

operator()
 Matrix, 14

operator+
 Matrix, 14

operator-
 Matrix, 15

operator==
 Matrix, 15

operator*
 Matrix, 14

print_vec.hpp
 operator<<, 23

pythag
 helper_func.cpp, 24
 helper_func.hpp, 20

Q_house
 TridiagonalResult, 17

Q_ql
 QLEigenResult, 17

QL
 Matrix, 15
 QLEigenResult, 16
 eigenvalues, 17
 Q_ql, 17

Random
 Matrix, 15

save_hdf5
 Matrix, 15

SIGN
 helper_func.cpp, 24
 helper_func.hpp, 20

size
 Matrix, 16

src/benchmark.hpp, 23, 24

src/custom_exception.hpp, 19, 20

src/helper_func.cpp, 24, 25

src/matrix_basic_func.cpp, 25

src/matrix_eigendecompp.cpp, 25

src/matrix_operators.cpp, 26

src/matrix_utilfuncs.cpp, 26

transpose
 Matrix, 15
 TridiagonalResult, 17
 d, 17
 e, 17
 Q_house, 17

vec
 matrix.h, 21

Zeros
 Matrix, 15