Study of multiple Heat Diffusion schemes

1.0.0

Generated by Doxygen 1.8.14

Contents

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

Chapter 2

Class Index

2.1 Class List

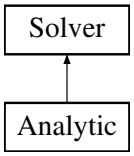
Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 Analytic Class Reference

Inheritance diagram for Analytic:



Public Member Functions

- Analytic (double dx, double dt, double L, double T, double D, double Tsur, double Tin)
- virtual Matrix computeSolution ()

Additional Inherited Members

3.1.1 Constructor & Destructor Documentation

3.1.1.1 Analytic()

```
Analytic::Analytic (
            double dx,
            double dt,
            double L,
            double T,
            double D,
            double Tsur,
            double Tin )
```

Construcs an analyic object

Exceptions

invalid_argument	("dx should be positive")
invalid_argument	("dt should be positive")
invalid_argument	("L should be positive")
invalid_argument	("T should be positive")
invalid_argument	("L should be equal or larger than dx")
invalid_argument	("T should be equal or larger than dt")

Parameters

dx	double. distance between two space steps
dt	double. time between two time steps
L	double. width of the 1D material to consider
T	double. Total time of the considerated problem
D	double. Diffusion coefficient of the material
Tsur	double. The temperature that will be applied on the boundaries of the material
Tin	double. The initial temperature of the material

3.1.2 Member Function Documentation

3.1.2.1 computeSolution()

```
Matrix Analytic::computeSolution ( )  [virtual]
```

Compute the solution and return it. This method is the analytical solution of the heat diffusion equation problem

Returns

Matrix. The computed matrix, can also be accesed through getComputedSolution()

See also

getComputedSolution()

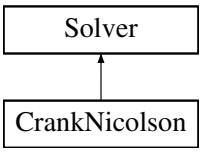
Implements Solver.

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

- Analytic.h
- Analytic.cpp

3.2 CrankNicolson Class Reference

Inheritance diagram for CrankNicolson:



Public Member Functions

- CrankNicolson (double dx, double dt, double L, double T, double D, double Tsur, double Tin)
- virtual Matrix computeSolution ()

Additional Inherited Members

3.2.1 Constructor & Destructor Documentation

3.2.1.1 CrankNicolson()

```
CrankNicolson::CrankNicolson (
            double dx,
            double dt,
            double L,
            double T,
            double D,
            double Tsur,
            double Tin )
```

Construcs a solver of the problem using Crank-Nicolson method

Exceptions

invalid_argument	("dx should be positive")
invalid_argument	("dt should be positive")
invalid_argument	("L should be positive")
invalid_argument	("T should be positive")
invalid_argument	("L should be equal or larger than dx")
invalid_argument	("T should be equal or larger than dt")

Parameters

dx	double. distance between two space steps
dt	double. time between two time steps
L	double. width of the 1D material to consider

Parameters

T	double. Total time of the considerated problem
D	double. Diffusion coefficient of the material
Tsur	double. The temperature that will be applied on the boundaries of the material
Tin	double. The initial temperature of the material

3.2.2 Member Function Documentation

3.2.2.1 computeSolution()

```
Matrix CrankNicolson::computeSolution ( )  [virtual]
```

Compute the solution and return it. This method is the Crank-Nicolson method applied to the heat diffusion equation problem

Returns

Matrix. The computed matrix, can also be accesed through getComputedSolution()

See also

getComputedSolution()

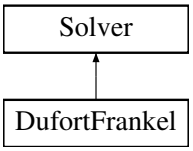
Implements Solver.

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

- CrankNicolson.h
- CrankNicolson.cpp

3.3 DufortFrankel Class Reference

Inheritance diagram for DufortFrankel:



Public Member Functions

- DufortFrankel (double dx, double dt, double L, double T, double D, double Tsur, double Tin)
- virtual Matrix computeSolution ()

Additional Inherited Members

3.3.1 Constructor & Destructor Documentation

3.3.1.1 DufortFrankel()

```
DufortFrankel::DufortFrankel (
            double dx,
            double dt,
            double L,
            double T,
            double D,
            double Tsur,
            double Tin )
```

Construcs a solver of the problem using DuFort-Frankel method

Exceptions

invalid_argument	("dx should be positive")
invalid_argument	("dt should be positive")
invalid_argument	("L should be positive")
invalid_argument	("T should be positive")
invalid_argument	("L should be equal or larger than dx")
invalid_argument	("T should be equal or larger than dt")

Parameters

dx	double. distance between two space steps
dt	double. time between two time steps
L	double. width of the 1D material to consider
T	double. Total time of the considerated problem
D	double. Diffusion coefficient of the material
Tsur	double. The temperature that will be applied on the boundaries of the material
Tin	double. The initial temperature of the material

3.3.2 Member Function Documentation

3.3.2.1 computeSolution()

```
Matrix DufortFrankel::computeSolution ( )  [virtual]
```

Compute the solution and return it. This method is the Dufort-Frankel method applied to the heat diffusion equation problem

Returns

Matrix. The computed matrix, can also be accesed through getComputedSolution()

See also

getComputedSolution()

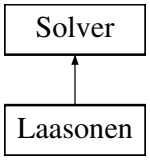
Implements Solver.

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

- DufortFrankel.h
- DufortFrankel.cpp

3.4 Laasonen Class Reference

Inheritance diagram for Laasonen:



Public Member Functions

- Laasonen (double dx, double dt, double L, double T, double D, double Tsur, double Tin)
- virtual Matrix computeSolution ()

Additional Inherited Members

3.4.1 Constructor & Destructor Documentation

3.4.1.1 Laasonen()

```
Laasonen::Laasonen (
            double dx,
            double dt,
            double L,
            double T,
            double D,
            double Tsur,
            double Tin )
```

Construcs a solver of the problem using Laasonen method

Exceptions

invalid_argument	("dx should be positive")
invalid_argument	("dt should be positive")
invalid_argument	("L should be positive")
invalid_argument	("T should be positive")
invalid_argument	("L should be equal or larger than dx")
invalid_argument	("T should be equal or larger than dt")

Parameters

dx	double. distance between two space steps
dt	double. time between two time steps
L	double. width of the 1D material to consider
T	double. Total time of the considerated problem
D	double. Diffusion coefficient of the material
Tsur	double. The temperature that will be applied on the boundaries of the material
Tin	double. The initial temperature of the material

3.4.2 Member Function Documentation

3.4.2.1 computeSolution()

```
Matrix Laasonen::computeSolution ( )  [virtual]
```

Compute the solution and return it. This method is the Laasonen method applied to the heat diffusion equation problem

Returns

Matrix. The computed matrix, can also be accesed through getComputedSolution()

See also

getComputedSolution()

Implements Solver.

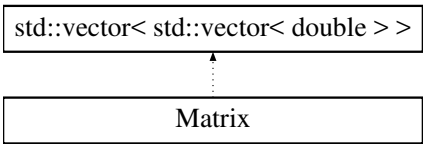
The documentation for this class was generated from the following files:

- Laasonen.h
- Laasonen.cpp

3.5 Matrix Class Reference

`#include <matrix.h>`

Inheritance diagram for Matrix:



Public Member Functions

- Matrix ()
- Matrix (int Nrows, int Ncols)
- Matrix (const Matrix &m)
- int getNrows () const
- int getNcols () const
- Matrix & operator= (const Matrix &m)
- bool operator== (const Matrix &m) const
- double one_norm () const
- double two_norm () const
- double uniform_norm () const
- Matrix operator* (const Matrix &a) const
- Matrix operator- (const Matrix &a) const
- Vector operator* (const Vector &v) const
- Matrix transpose () const

Friends

- std::istream & operator>> (std::istream &is, Matrix &m)
- std::ostream & operator<< (std::ostream &os, const Matrix &m)
- std::ifstream & operator>> (std::ifstream &ifs, Matrix &m)
- std::ofstream & operator<< (std::ofstream &ofs, const Matrix &m)

3.5.1 Detailed Description

A matrix class for data storage of a 2D array of doubles
The implementation is derived from the standard container vector std::vector
We use private inheritance to base our vector upon the library version whilst usto expose only those base class functions we wish to use - in this the array access operator []

The Matrix class provides:
-basic constructors for creating a matrix object from other matrix object, by creating empty matrix of a given size,
-input and oput operation via >> and << operators using keyboard or file
-basic operations like access via [] operator, assignment and comparision

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Matrix() [1/3]

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

Default constructor. Intialize an empty Matrix object

See also

Matrix(int Nrows, int Ncols)
Matrix(const Matrix& m)

3.5.2.2 Matrix() [2/3]

```
Matrix::Matrix (
            int Nrows,
            int Ncols )
```

Alternate constructor. build a matrix Nrows by Ncols

See also

Matrix()
Matrix(const Matrix& m)

Exceptions

invalid_argument	("matrix size negative or zero")

Parameters

Nrows	int. number of rows in matrix
Ncols	int. number of columns in matrix

3.5.2.3 Matrix() [3/3]

```
Matrix::Matrix (
            const Matrix & m )
```

Copy constructor. build a matrix from another matrix

See also

Matrix()
Matrix(int Nrows, int Ncols)

Parameters

m	Matrix&. matrix to copy from

3.5.3 Member Function Documentation

3.5.3.1 getNcols()

```
int Matrix::getNcols ( ) const
```

Normal public get method. get the number of columns

See also

int getNrows()const

Returns

int. number of columns in matrix

3.5.3.2 getNrows()

```
int Matrix::getNrows ( ) const
```

Normal public get method. get the number of rows

See also

int getNcols()const

Returns

int. number of rows in matrix

3.5.3.3 one_norm()

```
double Matrix::one_norm ( ) const
```

Normal public method that returns a double. It returns L1 norm of matrix

See also

two_norm()const
uniform_norm()const

Returns

double. matrix L1 norm

3.5.3.4 operator∗() [1/2]

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

Overloaded ∗operator that returns a Matrix. It Performs matrix by matrix multiplication.

See also

operator∗(const Matrix & a) const

Exceptions

out_of_range	("Matrix access error") One or more of the matrix have a zero size
std::out_of_range	("uncompatible matrix sizes") Number of columns in first matrix do not match number of columns in second matrix

Returns

Matrix. matrix-matrix product

Parameters

a	Matrix. matrix to multiply by

3.5.3.5 operator∗() [2/2]

```
Vector Matrix::operator* (
            const Vector & v ) const
```

Overloaded ∗operator that returns a Vector. It Performs matrix by vector multiplication.

See also

operator∗(const Matrix & a)const

Exceptions

std::out_of_range	("Matrix access error") matrix has a zero size
std::out_of_range	("Vector access error") vector has a zero size
std::out_of_range	("uncompatible matrix-vector sizes") Number of columns in matrix do not match the vector size

Returns

Vector. matrix-vector product

Parameters

v	Vector. Vector to multiply by

3.5.3.6 operator-()

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

Overloaded -operator that returns a Matrix. It Performs matrix by matrix substraction.

See also

operator-(const Matrix & a) const

Exceptions

out_of_range	("Matrix access error") One or more of the matrix have a zero size
std::out_of_range	("uncompatible matrix sizes") Number of columns/rows in first matrix do not match number of columns or/and rows in second matrix

Returns

Matrix. matrix-matrix product

Parameters

a	Matrix. matrix to multiply by

3.5.3.7 operator=()

```
Matrix & Matrix::operator= (
            const Matrix & m )
```

Overloaded assignment operator

See also

operator==(const Matrix& m)const

Returns

Matrix&. the matrix on the left of the assignment

Parameters

m	Matrix&. Matrix to assign from

3.5.3.8 operator==()

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

Overloaded comparison operator returns true or false depending on whether the matrices are the same or not

See also

operator=(const Matrix& m)

Returns

bool. true or false

Parameters

m	Matrix&. Matrix to compare to

3.5.3.9 transpose()

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

public method that returns the transpose of the matrix. It returns the transpose of matrix

Returns

Matrix. matrix transpose

3.5.3.10 two_norm()

```
double Matrix::two_norm ( ) const
```

Normal public method that returns a double. It returns L2 norm of matrix

See also

one_norm()const
uniform_norm()const

Returns

double. matrix L2 norm

3.5.3.11 uniform_norm()

```
double Matrix::uniform_norm ( ) const
```

Normal public method that returns a double. It returns L_max norm of matrix

See also

one_norm()const
two_norm()const

Returns

double. matrix L_max norm

3.5.4 Friends And Related Function Documentation

3.5.4.1 operator<< [1/2]

```
std::ostream& operator<< (
            std::ostream & os,
            const Matrix & m )  [friend]
```

Overloaded ostream << operator. Display output if matrix has size user will be asked to input only matrix values if matrix was not initialized user can choose matrix size and input it values

See also

operator>>(std::ifstream& ifs, Matrix& m)
operator>>(std::istream& is, Matrix& m)
operator<<(std::ostream& os, const Matrix& m)

Returns

std::ostream&. The ostream object

Parameters

os	Display output stream
m	Matrix to read from

3.5.4.2 operator<< [2/2]

```
std::ofstream& operator<< (
            std::ofstream & ofs,
            const Matrix & m )  [friend]
```

Overloaded ofstream << operator. File output the file output operator is compatible with file input operator, ie. everything written can be read later.

See also

operator>>(std::ifstream& ifs, Matrix& m)
operator<<(std::ofstream& ofs, const Matrix& m)
operator>>(std::istream& is, Matrix& m)

Exceptions

std::invalid_argument	("file read error - negative matrix size");

Returns

std::ofstream&. The ofstream object

Parameters

m	Matrix to read from

3.5.4.3 operator>> [1/2]

```
std::istream& operator>> (
            std::istream & is,
            Matrix & m )  [friend]
```

Overloaded istream >> operator. Keyboard input if matrix has size user will be asked to input only matrix values if matrix was not initialized user can choose matrix size and input it values

See also

operator<<(std::ofstream& ofs, const Matrix& m)
operator>>(std::istream& is, Matrix& m)
operator<<(std::ostream& os, const Matrix& m)

Exceptions

std::invalid_argument	("read error - negative matrix size");

Returns

std::istream&. The istream object

Parameters

is	Keyboard input stream
m	Matrix to write into

3.5.4.4 operator>> [2/2]

```
std::ifstream& operator>> (
            std::ifstream & ifs,
            Matrix & m )  [friend]
```

Overloaded ifstream >> operator. File input the file output operator is compatible with file input operator, ie. everything written can be read later.

See also

operator>>(std::ifstream& ifs, Matrix& m)
operator<<(std::ofstream& ofs, const Matrix& m)
operator<<(std::ostream& os, const Matrix& m)

Returns

std::ifstream&. The ifstream object

Parameters

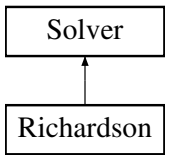
ifs	Input file stream with opened matrix file
m	Matrix to write into

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

- matrix.h
- matrix.cpp

3.6 Richardson Class Reference

Inheritance diagram for Richardson:



Public Member Functions

- Richardson (double dx, double dt, double L, double T, double D, double Tsur, double Tin)
- virtual Matrix computeSolution ()

Additional Inherited Members

3.6.1 Constructor & Destructor Documentation

3.6.1.1 Richardson()

```
Richardson::Richardson (
            double dx,
            double dt,
            double L,
            double T,
            double D,
            double Tsur,
            double Tin )
```

Construcs a solver of the problem using Richardson method

Exceptions

invalid_argument	("dx should be positive")
invalid_argument	("dt should be positive")
invalid_argument	("L should be positive")
invalid_argument	("T should be positive")
invalid_argument	("L should be equal or larger than dx")
invalid_argument	("T should be equal or larger than dt")

Parameters

dx	double. distance between two space steps
dt	double. time between two time steps
L	double. width of the 1D material to consider
T	double. Total time of the considerated problem
D	double. Diffusion coefficient of the material
Tsur	double. The temperature that will be applied on the boundaries of the material
Tin	double. The initial temperature of the material

3.6.2 Member Function Documentation

3.6.2.1 computeSolution()

```
Matrix Richardson::computeSolution ( )  [virtual]
```

Compute the solution and return it. This method is the Richardson method applied to the heat diffusion equation problem

Returns

Matrix. The computed matrix, can also be accesed through getComputedSolution()

See also

getComputedSolution()

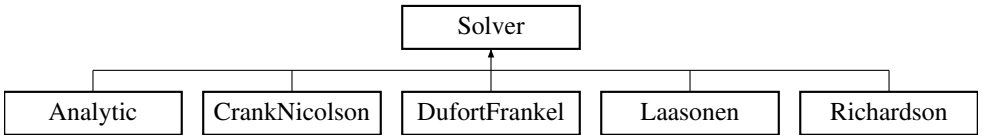
Implements Solver.

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

- Richardson.h
- Richardson.cpp

3.7 Solver Class Reference

Inheritance diagram for Solver:



Public Member Functions

- Solver (double dx, double dt, double L, double T, double D, double Tsur, double Tin)
- Matrix getComputedSolution ()
- double getDT ()
- double getDX ()
- double getL ()
- double getT ()
- double getD ()
- double getTsur ()
- double getTin ()
- virtual Matrix computeSolution ()=0
- virtual ~Solver ()

Protected Attributes

- Matrix **computedSolution**
- double **dx**
- double **dt**
- double **L**
- double **T**
- double **D**
- double **Tsur**
- double **Tin**

Friends

- std::ostream & operator<< (std::ostream &os, Solver &m)
- std::ofstream & operator<< (std::ofstream &ifs, Solver &m)

3.7.1 Constructor & Destructor Documentation

3.7.1.1 Solver()

```
Solver::Solver (
            double dx,
            double dt,
            double L,
            double T,
            double D,
            double Tsur,
            double Tin )
```

Construcs a solver of the problem, can not be instanciated as an object since it is a virtual base class

Exceptions

invalid_argument	("dx should be positive")
invalid_argument	("dt should be positive")
invalid_argument	("L should be positive")
invalid_argument	("T should be positive")
invalid_argument	("L should be equal or larger than dx")
invalid_argument	("T should be equal or larger than dt")

Parameters

dx	double. distance between two space steps
dt	double. time between two time steps
L	double. width of the 1D material to consider
T	double. Total time of the considerated problem
D	double. Diffusion coefficient of the material

Parameters

Tsur	double. The temperature that will be applied on the boundaries of the material
Tin	double. The initial temperature of the material

3.7.1.2 ~Solver()

```
virtual Solver::~Solver ( )  [inline], [virtual]
```

Destroys the object

3.7.2 Member Function Documentation

3.7.2.1 computeSolution()

```
virtual Matrix Solver::computeSolution ( )  [pure virtual]
```

Compute the solution and return it. This method must be implemented in the child class if you want it not to be virtual

Returns

Matrix. The computed matrix, can also be accesed through getComputedSolution()

See also

getComputedSolution()

Implemented in CrankNicolson, Analytic, DufortFrankel, Laasonen, and Richardson.

3.7.2.2 getComputedSolution()

```
Matrix Solver::getComputedSolution ( )
```

Return the computed matrix, computeSolution() has to be called first to get the solution matrix.

Returns

Matrix, the computed matrix

3.7.2.3 getD()

```
double Solver::getD ( )
```

get the diffusion coefficient

Returns

- double. the diffusion coefficient considered

3.7.2.4 getDT()

```
double Solver::getDT ( )
```

get the time step

Returns

- double. the time step considered

3.7.2.5 getDX()

```
double Solver::getDX ( )
```

get the space step

Returns

- double. the space step considered

3.7.2.6 getL()

```
double Solver::getL ( )
```

get the width of the material

Returns

- double. the width considered

3.7.2.7 getT()

```
double Solver::getT ( )
```

get the overall time

Returns

- double. the overall time considered

3.7.2.8 getTin()

```
double Solver::getTin ( )
```

get the initial temperature

Returns

- double. the initial temperature considered

3.7.2.9 getTsur()

```
double Solver::getTsur ( )
```

get the temperature at the surface

Returns

- double. the temperature applied on the surface

3.7.3 Friends And Related Function Documentation

3.7.3.1 operator<< [1/2]

```
std::ostream& operator<< (
            std::ostream & os,
            Solver & m )  [friend]
```

redifinition of the << operator to the screen, displays the time every 0.1seconde from 0 to 0.5 seconde.

Returns

the generated stream

3.7.3.2 operator<< [2/2]

```
std::ofstream& operator<< (
            std::ofstream & ifs,
            Solver & m )  [friend]
```

redifinition of the << operator to a file, displays the time every 0.1seconde from 0 to 0.5 seconde. Especially mafe for GNUPlot usage.

Returns

the generated stream

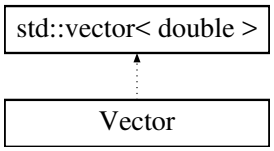
The documentation for this class was generated from the following files:

- Solver.h
- Solver.cpp

3.8 Vector Class Reference

```
#include <vector.h>
```

Inheritance diagram for Vector:



Public Member Functions

- Vector ()
- Vector (int Num)
- Vector (const Vector &v)
- Vector & operator= (const Vector &v)
- bool operator== (const Vector &v) const
- int getSize () const
- double one_norm () const
- double two_norm () const
- double uniform_norm () const

Friends

- std::istream & operator>> (std::istream &is, Vector &v)
- std::ostream & operator<< (std::ostream &os, const Vector &v)
- std::ifstream & operator>> (std::ifstream &ifs, Vector &v)
- std::ofstream & operator<< (std::ofstream &ofs, const Vector &v)

3.8.1 Detailed Description

A vector class for data storage of a 1D array of doubles
The implementation is derived from the standard container vector std::vector
We use private inheritance to base our vector upon the library version whilst usto expose only those base class functions we wish to use - in this the array access operator []

The Vector class provides:
-basic constructors for creating vector obcjet from other vector object, or by creating empty vector of a given size,
-input and oput operation via >> and << operators using keyboard or file
-basic operations like access via [] operator, assignment and comparision

3.8.2 Constructor & Destructor Documentation

3.8.2.1 Vector() [1/3]

```
Vector::Vector ( )
```

Default constructor. Intialize an empty Vector object

See also

Vector(int Num)
Vector(const Vector& v)

3.8.2.2 Vector() [2/3]

```
Vector::Vector (
            int Num )  [explicit]
```

Explicit alterative constructor takes an intiger. it is explicit since implicit type conversion int -> vector doesn't make sense Intialize Vector object of size Num

See also

Vector()
Vector(const Vector& v)

Exceptions

invalid_argument	("vector size negative")

Parameters

Num	int. Size of a vector

3.8.2.3 Vector() [3/3]

```
Vector::Vector (
            const Vector & v )
```

Copy constructor takes an Vector object reference. Intialize Vector object with another Vector object

See also

Vector()
Vector(int Num)

3.8.3 Member Function Documentation

3.8.3.1 getSize()

```
int Vector::getSize ( ) const
```

Normal get method that returns integer, the size of the vector

Returns

int. the size of the vector

3.8.3.2 one_norm()

```
double Vector::one_norm ( ) const
```

Normal public method that returns a double. It returns L1 norm of vector

See also

two_norm()const
uniform_norm()const

Returns

double. vectors L1 norm

3.8.3.3 operator=()

```
Vector & Vector::operator= (
            const Vector & v )
```

Overloaded assignment operator

See also

operator==(const Vector& v)const

Parameters

v	Vector to assign from

Returns

the object on the left of the assignment

Parameters

v	Vecto&. Vector to assign from

3.8.3.4 operator==()

```
bool Vector::operator== (
            const Vector & v ) const
```

Overloaded comparison operator returns true if vectors are the same within a tolerance (1.e-07)

See also

operator=(const Vector& v)
operator[ ](int i)
operator[ ](int i)const

Returns

bool. true or false

Exceptions

invalid_argument	("incompatible vector sizes\n")

Parameters

v	Vector&. vector to compare

3.8.3.5 two_norm()

```
double Vector::two_norm ( ) const
```

Normal public method that returns a double. It returns L2 norm of vector

See also

one_norm()const
uniform_norm()const

Returns

double. vectors L2 norm

3.8.3.6 uniform_norm()

```
double Vector::uniform_norm ( ) const
```

Normal public method that returns a double. It returns L_max norm of vector

See also

one_norm()const
two_norm()const

Exceptions

out_of_range	("vector access error") vector has zero size

Returns

double. vectors Lmax norm

3.8.4 Friends And Related Function Documentation

3.8.4.1 operator<< [1/2]

```
std::ostream& operator<< (
            std::ostream & os,
            const Vector & v )  [friend]
```

Overloaded ifstream << operator. Display output.

See also

operator>>(std::istream& is, Vector& v)
operator>>(std::ifstream& ifs, Vector& v)
operator<<(std::ofstream& ofs, const Vector& v)

Returns

std::ostream&. the output stream object os

Parameters

os	output file stream
v	vector to read from

3.8.4.2 operator<< [2/2]

```
std::ofstream& operator<< (
            std::ofstream & ofs,
            const Vector & v )  [friend]
```

Overloaded ofstream << operator. File output. the file output operator is compatible with file input operator, ie. everything written can be read later.

See also

operator>>(std::istream& is, Vector& v)
operator>>(std::ifstream& ifs, Vector& v)
operator<<(std::ostream& os, const Vector& v)

Returns

std::ofstream&. the output ofstream object ofs

Parameters

ofs	outputfile stream. With opened file
v	Vector&. vector to read from

3.8.4.3 operator>> [1/2]

```
std::istream& operator>> (
            std::istream & is,
            Vector & v )  [friend]
```

Overloaded istream >> operator. Keyboard input if vector has size user will be asked to input only vector values if vector was not initialized user can choose vector size and input it values

See also

operator>>(std::ifstream& ifs, Vector& v)
operator<<(std::ostream& os, const Vector& v)
operator<<(std::ofstream& ofs, const Vector& v)

Returns

std::istream&. the input stream object is

Exceptions

std::invalid_argument	("read error - negative vector size");

Parameters

is	keyboard input straem. For user input
v	Vector&. vector to write to

3.8.4.4 operator>> [2/2]

```
std::ifstream& operator>> (
            std::ifstream & ifs,
            Vector & v )  [friend]
```

Overloaded ifstream >> operator. File input the file output operator is compatible with file input operator, ie. everything written can be read later.

See also

operator>>(std::istream& is, Vector& v)
operator<<(std::ostream& os, const Vector& v)
operator<<(std::ofstream& ofs, const Vector& v)

Returns

ifstream&. the input ifstream object ifs

Exceptions

std::invalid_argument	("file read error - negative vector size");

Parameters

ifs	input file straem. With opened matrix file
v	Vector&. vector to write to

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

- vector.h
- vector.cpp

Index