# Study of multiple Heat Diffusion schemes 1.0.0

Generated by Doxygen 1.8.14

# **Contents**

1	Hier	archica	Index	1
	1.1	Class I	Hierarchy	1
2	Clas	s Index		3
	2.1	Class I	_ist	3
3	Clas	s Docu	mentation	5
	3.1	Analyti	c Class Reference	5
		3.1.1	Constructor & Destructor Documentation	5
			3.1.1.1 Analytic()	5
		3.1.2	Member Function Documentation	6
			3.1.2.1 computeSolution()	6
	3.2	Crank	Nicolson Class Reference	7
		3.2.1	Constructor & Destructor Documentation	7
			3.2.1.1 CrankNicolson()	7
		3.2.2	Member Function Documentation	8
			3.2.2.1 computeSolution()	8
	3.3	Dufortl	Frankel Class Reference	8
		3.3.1	Constructor & Destructor Documentation	9
			3.3.1.1 DufortFrankel()	9
		3.3.2	Member Function Documentation	9
			3.3.2.1 computeSolution()	9
	3.4	Laasor	nen Class Reference	0
		3.4.1	Constructor & Destructor Documentation	0

ii CONTENTS

		3.4.1.1	Laasonen()	 	10
	3.4.2	Member F	Function Documentation	 	11
		3.4.2.1	computeSolution()	 	11
3.5	Matrix	Class Refer	rence	 	12
	3.5.1	Detailed D	Description	 	12
	3.5.2	Constructo	or & Destructor Documentation	 	13
		3.5.2.1	Matrix() [1/3]	 	13
		3.5.2.2	Matrix() [2/3]	 	13
		3.5.2.3	Matrix() [3/3]	 	13
	3.5.3	Member F	Function Documentation	 	14
		3.5.3.1	getNcols()	 	14
		3.5.3.2	getNrows()	 	14
		3.5.3.3	one_norm()	 	15
		3.5.3.4	operator*() [1/2]	 	15
		3.5.3.5	operator*() [2/2]	 	15
		3.5.3.6	operator-()	 	16
		3.5.3.7	operator=()	 	17
		3.5.3.8	operator==()	 	17
		3.5.3.9	transpose()	 	17
		3.5.3.10	two_norm()	 	18
		3.5.3.11	uniform_norm()	 	18
	3.5.4	Friends Ar	nd Related Function Documentation	 	18
		3.5.4.1	operator<< [1/2]	 	18
		3.5.4.2	operator<< [2/2]	 	19
		3.5.4.3	operator>> [1/2]	 	19
		3.5.4.4	operator>> [2/2]	 	20
3.6	Richard	dson Class	Reference	 	20
	3.6.1	Constructo	or & Destructor Documentation	 	21
		3.6.1.1	Richardson()	 	21
	3.6.2	Member F	Function Documentation	 	22

CONTENTS

		3.6.2.1	computeSolution()	22
3.7	Solver	Class Ref	erence	22
	3.7.1	Construc	tor & Destructor Documentation	23
		3.7.1.1	Solver()	23
		3.7.1.2	~Solver()	24
	3.7.2	Member	Function Documentation	24
		3.7.2.1	computeSolution()	24
		3.7.2.2	getComputedSolution()	24
		3.7.2.3	getD()	25
		3.7.2.4	getDT()	25
		3.7.2.5	getDX()	25
		3.7.2.6	getL()	25
		3.7.2.7	getT()	26
		3.7.2.8	getTin()	26
		3.7.2.9	getTsur()	26
	3.7.3	Friends A	And Related Function Documentation	26
		3.7.3.1	operator<< [1/2]	26
		3.7.3.2	operator<< [2/2]	27
3.8	Vector	Class Ref	erence	27
	3.8.1	Detailed	Description	28
	3.8.2	Construc	etor & Destructor Documentation	28
		3.8.2.1	Vector() [1/3]	28
		3.8.2.2	Vector() [2/3]	28
		3.8.2.3	Vector() [3/3]	29
	3.8.3	Member	Function Documentation	29
		3.8.3.1	getSize()	29
		3.8.3.2	one_norm()	29
		3.8.3.3	operator=()	29
		3.8.3.4	operator==()	30
		3.8.3.5	two_norm()	30
		3.8.3.6	uniform_norm()	31
	3.8.4	Friends A	And Related Function Documentation	31
		3.8.4.1	operator<< [1/2]	31
		3.8.4.2	operator<< [2/2]	32
		3.8.4.3	operator>> [1/2]	32
		3.8.4.4	operator>> [2/2]	33
Index				35

# **Chapter 1**

# **Hierarchical Index**

# 1.1 Class Hierarchy

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

er	. 22
Analytic	5
CrankNicolson	7
OufortFrankel	8
aasonen	
Richardson	20
or	
Matrix	12
/ector	27

2 Hierarchical Index

# Chapter 2

# **Class Index**

# 2.1 Class List

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

Analytic										 							 				Ę
CrankNicolson										 							 				7
DufortFrankel										 							 				8
Laasonen										 							 				10
Matrix										 							 				12
Richardson .										 							 				20
Solver										 							 				22
Vector										 							 				27

4 Class Index

# **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()

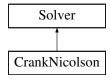
Implements Solver.

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()

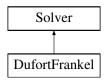
Implements Solver.

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()

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.

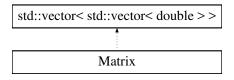
The documentation for this class was generated from the following files:

- · 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)</li>
- 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 Matrix Class Reference

# 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]
```

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]
```

Copy constructor. build a matrix from another matrix

90	0	2	len	

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 Matrix Class Reference

```
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]
```

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]
```

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-()

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 Matrix Class Reference

#### 3.5.3.7 operator=()

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==()

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

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

3.5 Matrix Class Reference 19

#### **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]</pre>
```

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()

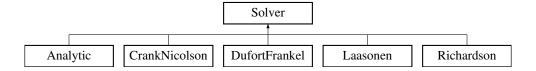
Implements Solver.

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 ()

3.7 Solver Class Reference 23

# **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 $\sim$ 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 Solver Class Reference 25

```
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
```

Generated by Doxygen

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]</pre>
```

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

#### Returns

the generated stream

3.8 Vector Class Reference 27

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

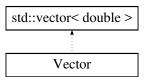
The documentation for this class was generated from the following files:

- · 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)</li>

# 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]
```

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
INUIII	IIII. OIZE UI A VECIUI

```
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 )
```

Generated by Doxygen

See also

Overloaded assignment operator

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==()

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
```

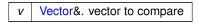
#### Returns

bool. true or false

# **Exceptions**

invalid\_argument ("incompatible vector sizes\n")

#### **Parameters**



## 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]
```

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	

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

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

3.8 Vector Class Reference 33

# **Exceptions**

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

#### **Parameters**

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

# 

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)
```

Vector & v ) [friend]

#### 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

0.1	
~Solver	getNrows, 14
Solver, 24	Matrix, 13
Analytic 5	one_norm, 14
Analytic, 5	operator<<, 18, 19
Analytic, 5	operator>>, 19, 20
computeSolution, 6	operator*, 15
computeSolution	operator-, 16
Analytic, 6	operator=, 17
-	operator==, 17
CrankNicolson, 8	transpose, 17
DufortFrankel, 9	two_norm, 18
Laasonen, 11	uniform_norm, 18
Richardson, 22	
Solver, 24	one_norm
CrankNicolson, 7	Matrix, 14
computeSolution, 8	Vector, 29
CrankNicolson, 7	operator<<
DufautFraulial 0	Matrix, 18, 19
DufortFrankel, 8	Solver, 26
computeSolution, 9	Vector, 31, 32
DufortFrankel, 9	operator>>
actComputedColution	Matrix, 19, 20
getComputedSolution	Vector, 32, 33
Solver, 24	operator*
getDT	Matrix, 15
Solver, 25	operator-
getDX	Matrix, 16
Solver, 25	operator=
getNcols	Matrix, 17
Matrix, 14	Vector, 29
getNrows	operator==
Matrix, 14	Matrix, 17
getSize	
Vector, 29	Vector, 30
getTin	Richardson, 20
Solver, 26	computeSolution, 22
getTsur	Richardson, 21
Solver, 26	Hichardson, 21
getD	Solver, 22
Solver, 24	$\sim$ Solver, 24
getL	computeSolution, 24
Solver, 25	getComputedSolution, 24
getT	getDT, 25
Solver, 25	_
	getDX, 25
Laasonen, 10	getTin, 26
computeSolution, 11	getTsur, 26
Laasonen, 10	getD, 24
W. 1. 10	getL, 25
Matrix, 12	getT, 25
getNcols, 14	operator<<, 26

36 INDEX

```
Solver, 23
transpose
    Matrix, 17
two_norm
    Matrix, 18
    Vector, 30
uniform_norm
    Matrix, 18
    Vector, 31
Vector, 27
    getSize, 29
    one_norm, 29
    operator << , 31, 32
    operator>>, 32, 33
    operator=, 29
    operator==, 30
    two_norm, 30
    uniform_norm, 31
    Vector, 28, 29
```