HeatConduction

Generated by Doxygen 1.8.13

Contents

1 Hierarchical Index **1**

1.1 Class Hierarchy . . . 1

2 Class Index **3**

2.1 Class List . . . 3

3 File Index **5**

3.1 File List . . . 5

4 Class Documentation **7**

4.1 AnalyticalSolution Class Reference . . . 7

4.1.1 Detailed Description . . . 7

4.1.2 Constructor & Destructor Documentation . . . 8

4.1.2.1 AnalyticalSolution() . . . 8

4.1.3 Member Function Documentation . . . 8

4.1.3.1 solve() . . . 8

4.2 CrankNicholson Class Reference . . . 9

4.2.1 Detailed Description . . . 9

4.2.2 Constructor & Destructor Documentation . . . 10

4.2.2.1 CrankNicholson() . . . 10

4.2.3 Member Function Documentation . . . 10

4.2.3.1 solve() . . . 10

4.3 DuFort_Frankel Class Reference . . . 11

4.3.1 Detailed Description . . . 11

4.3.2 Constructor & Destructor Documentation . . . 12

4.3.2.1 DuFort_Frankel() 12
4.3.3 Member Function Documentation 12
4.3.3.1 advance() 12
4.4 ExplicitMethod Class Reference 13
4.4.1 Detailed Description 13
4.4.2 Constructor & Destructor Documentation 14
4.4.2.1 ExplicitMethod() 14
4.4.3 Member Function Documentation 14
4.4.3.1 advance() 14
4.4.3.2 solve() 15
4.5 HeatConduction Class Reference 15
4.5.1 Detailed Description 16
4.5.2 Constructor & Destructor Documentation 17
4.5.2.1 HeatConduction() 17
4.5.3 Member Function Documentation 17
4.5.3.1 get_u_n() 17
4.5.3.2 solve() 18
4.6 ImplicitMethod Class Reference 18
4.6.1 Detailed Description 19
4.6.2 Constructor & Destructor Documentation 19
4.6.2.1 ImplicitMethod() 19
4.6.3 Member Function Documentation 20
4.6.3.1 solve() 20
4.6.3.2 ThomasAlgorith() 20
4.7 Laasonen Class Reference 21
4.7.1 Detailed Description 21
4.7.2 Constructor & Destructor Documentation 21
4.7.2.1 Laasonen() 21
4.7.3 Member Function Documentation 22
4.7.3.1 solve() 22
4.8 Richardson Class Reference 22
4.8.1 Detailed Description 23
4.8.2 Constructor & Destructor Documentation 23
4.8.2.1 Richardson() 23
4.8.3 Member Function Documentation 24
4.8.3.1 advance() 24

5 File Documentation 25

5.1 HeatConduction.cpp File Reference 25

5.1.1 Detailed Description 25

5.2 HeatConduction.h File Reference 26

5.2.1 Detailed Description 26

5.3 Norms.cpp File Reference 27

5.3.1 Detailed Description 27

5.3.2 Function Documentation 27

5.3.2.1 norm_one() 27

5.3.2.2 norm_two() 28

5.3.2.3 norm_uniform() 28

5.4 Norms.h File Reference 29

5.4.1 Detailed Description 29

5.4.2 Function Documentation 29

5.4.2.1 norm_one() 29

5.4.2.2 norm_two() 30

5.4.2.3 norm_uniform() 30

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

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

- HeatConduction 15
  - AnalyticalSolution 7
  - ExplicitMethod 13
    - DuFort_Frankel 11
    - Richardson 22
  - ImplicitMethod 18
    - CrankNicholson 9
    - Laasonen 21

Chapter 2

Class Index

2.1 Class List

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

AnalyticalSolution
Sub Class used to calculate the analytical solution 7
CrankNicholson
Sub sub Class used to calculate the Crank-Nicholson scheme 9
DuFort_Frankel
Sub sub Class used to calculate the **DuFort_Frankel** (p. 11) scheme 11
ExplicitMethod
Sub Abstract Class used to calculate the Explicit scheme 13
HeatConduction
Base abstract Class which include all the parameters to solve the problem 15
ImplicitMethod
Sub Abstract Class used to calculate the Implicit scheme 18
Laasonen
Sub sub Class used to calculate the **Laasonen** (p. 21) scheme 21
Richardson
Sub sub Class used to calculate the **Richardson** (p. 22) scheme 22

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

HeatConduction.cpp
Different objects to resolve an Heat Conduction problem 25
HeatConduction.h
Different objects to resolve an Heat Conduction problem 26
main.cpp . . . . **??**
Norms.cpp
Functions to calculates norms 27
Norms.h
Functions to calculates norms 29

Chapter 4

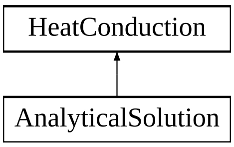
Class Documentation

4.1 AnalyticalSolution Class Reference

Sub Class used to calculate the analytical solution.

```
#include <HeatConduction.h>
```

Inheritance diagram for AnalyticalSolution:



Public Member Functions

- **AnalyticalSolution** (double **Tin_0**, double **Text_0**, double **Xmin**, double **Xmax**, double **Tend**, double **D**, double **dx**, double **dt**)

  *Constructor of the **AnalyticalSolution** (p. 7) class.*
- virtual void **solve** ()

  Solve with the analytical solution.

Additional Inherited Members

4.1.1 Detailed Description

Sub Class used to calculate the analytical solution.

AnalyticalSolution (p. 7) is a sub class of **HeatConduction** (p. 15). It use the attribut of the mother class to calculate the analytical solution.

Definition at line 56 of file HeatConduction.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 AnalyticalSolution()

```
AnalyticalSolution::AnalyticalSolution (
            double Tin_0,
            double Text_0,
            double Xmin,
            double Xmax,
            double Tend,
            double D,
            double dx,
            double dt )
```

Constructor of the **AnalyticalSolution** (p. 7) class.

Parameters

Tin←↩ _0	- initial condition Temperature inside
Text←↩ _0	- initial condition Temperature outside
Xmin	- the X position far left
Xmax	- the X position far right
Tend	- the end time of the simulation
D	- the difusivity of the wall
dx	- the space step
dt	- the time step

Definition at line 94 of file HeatConduction.cpp.

4.1.3 Member Function Documentation

4.1.3.1 solve()

```
void AnalyticalSolution::solve ( )  [virtual]
```

Solve with the analytical solution.

Parameters

none	

Returns

void - the result is stored in the vector u_n of the mother Class

Reimplemented from **HeatConduction** (p. 18).

Definition at line 103 of file HeatConduction.cpp.

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

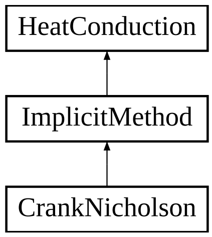
- **HeatConduction.h**
- **HeatConduction.cpp**

4.2 CrankNicholson Class Reference

Sub sub Class used to calculate the Crank-Nicholson scheme.

```
#include <HeatConduction.h>
```

Inheritance diagram for CrankNicholson:



Public Member Functions

- **CrankNicholson** (double **Tin_0**, double **Text_0**, double **Xmin**, double **Xmax**, double **Tend**, double **D**, double **dx**, double **dt**)

    Constructor of the ***Laasonen*** *(p. 21) class.*
- virtual void **solve** ()

    Solve method. The matrix abc and the vector d are define after the Crank-Nicholson scheme.

Additional Inherited Members

4.2.1 Detailed Description

Sub sub Class used to calculate the Crank-Nicholson scheme.

CrankNicholson (p. 9) is a sub class of **ImplicitMethod** (p. 18). It use the Crank-Nicholson scheme, an implicit scheme to calculate an Heat Conduction problem of a wall which have a temperature imposed at the extremities.

Definition at line 152 of file HeatConduction.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 CrankNicholson()

```
CrankNicholson::CrankNicholson (
            double Tin_0,
            double Text_0,
            double Xmin,
            double Xmax,
            double Tend,
            double D,
            double dx,
            double dt )
```

Constructor of the **Laasonen** (p. 21) class.

Parameters

Tin_0	- initial condition Temperature inside
Text_0	- initial condition Temperature outside
Xmin	- the X position far left
Xmax	- the X position far right
Tend	- the end time of the simulation
D	- the difusivity of the wall
dx	- the space step
dt	- the time step

Definition at line 347 of file HeatConduction.cpp.

4.2.3 Member Function Documentation

4.2.3.1 solve()

```
void CrankNicholson::solve ( )  [virtual]
```

Solve method. The matrix abc and the vector d are define after the Crank-Nicholson scheme.

Parameters

none	

Returns

void - the result is stored in the vector u_n of the mother Class

Reimplemented from **ImplicitMethod** (p. 20).

Definition at line 356 of file HeatConduction.cpp.

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

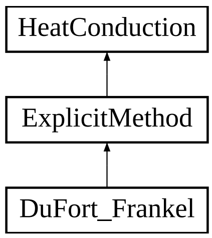
- **HeatConduction.h**
- **HeatConduction.cpp**

4.3 DuFort_Frankel Class Reference

Sub sub Class used to calculate the **DuFort_Frankel** (p. 11) scheme.

```
#include <HeatConduction.h>
```

Inheritance diagram for DuFort_Frankel:



Public Member Functions

- **DuFort_Frankel** (double **Tin_0**, double **Text_0**, double **Xmin**, double **Xmax**, double **Tend**, double **D**, double **dx**, double **dt**)

  Constructor of the ***DuFort_Frankel*** *(p. 11) class.*
- virtual void **advance** (int i)

  Calcul of un_plus1 according to ***DuFort_Frankel*** *(p. 11) scheme.*

Additional Inherited Members

4.3.1 Detailed Description

Sub sub Class used to calculate the **DuFort_Frankel** (p. 11) scheme.

DuFort_Frankel (p. 11) is a sub class of **ExplicitMethod** (p. 13). It use the DuFort-Frankel scheme, a second order explicit scheme to calculate an Heat Conduction problem of a wall which have a temperature imposed at the extremities.

Definition at line 107 of file HeatConduction.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 DuFort_Frankel()

```
DuFort_Frankel::DuFort_Frankel (
            double Tin_0,
            double Text_0,
            double Xmin,
            double Xmax,
            double Tend,
            double D,
            double dx,
            double dt )
```

Constructor of the **DuFort_Frankel** (p. 11) class.

Parameters

Tin_0	- initial condition Temperature inside
Text_0	- initial condition Temperature outside
Xmin	- the X position far left
Xmax	- the X position far right
Tend	- the end time of the simulation
D	- the difusivity of the wall
dx	- the space step
dt	- the time step

Definition at line 248 of file HeatConduction.cpp.

4.3.3 Member Function Documentation

4.3.3.1 advance()

```
void DuFort_Frankel::advance (
            int i ) [virtual]
```

Calcul of un_plus1 according to **DuFort_Frankel** (p. 11) scheme.

Parameters

i	- the space iteration at which is the solve method

Returns

void - the result is stored in the vector u_nplus1 of the mother Class

Reimplemented from **ExplicitMethod** (p. 14).

Definition at line 257 of file HeatConduction.cpp.

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

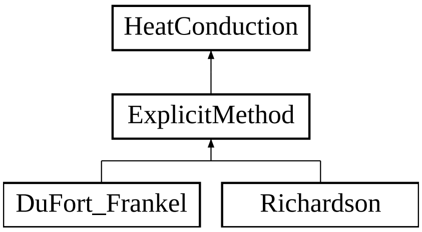
- **HeatConduction.h**
- **HeatConduction.cpp**

4.4 ExplicitMethod Class Reference

Sub Abstract Class used to calculate the Explicit scheme.

```
#include <HeatConduction.h>
```

Inheritance diagram for ExplicitMethod:



Public Member Functions

- **ExplicitMethod** (double **Tin_0**, double **Text_0**, double **Xmin**, double **Xmax**, double **Tend**, double **D**, double **dx**, double **dt**)

    *Constructor of the **ExplicitMethod** (p. 13) class.*
- virtual void **solve** ()

    Solve regroup the common part of the Explicit Method.
- virtual void **advance** (int i)

    Abstract method implemented in the sub sub classes.

Additional Inherited Members

4.4.1 Detailed Description

Sub Abstract Class used to calculate the Explicit scheme.

ExplicitMethod (p. 13) is a sub class of **HeatConduction** (p. 15). Both explicit method share the same solve method, which is implemented in this class. The advance method is an abstract method implemented in the sub sub classes.

Definition at line 70 of file HeatConduction.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 ExplicitMethod()

```
ExplicitMethod::ExplicitMethod (
            double Tin_0,
            double Text_0,
            double Xmin,
            double Xmax,
            double Tend,
            double D,
            double dx,
            double dt )
```

Constructor of the **ExplicitMethod** (p. 13) class.

Parameters

Tin←↩ _0	- initial condition Temperature inside
Text←↩ _0	- initial condition Temperature outside
Xmin	- the X position far left
Xmax	- the X position far right
Tend	- the end time of the simulation
D	- the difusivity of the wall
dx	- the space step
dt	- the time step

Definition at line 129 of file HeatConduction.cpp.

4.4.3 Member Function Documentation

4.4.3.1 advance()

```
void ExplicitMethod::advance (
            int i ) [virtual]
```

Abstract method implemented in the sub sub classes.

Parameters

i	- the space iteration at which is the solve method

Returns

void - the result is stored in the vector u_nplus1 of the mother Class

Reimplemented in **Richardson** (p. 24), and **DuFort_Frankel** (p. 12).

Definition at line 138 of file HeatConduction.cpp.

4.4.3.2 solve()

```
void ExplicitMethod::solve ( ) [virtual]
```

Solve regroup the common part of the Explicit Method.

Parameters

none	

Returns

void - the result is stored in the vector u_n of the mother Class

Reimplemented from **HeatConduction** (p. 18).

Definition at line 147 of file HeatConduction.cpp.

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

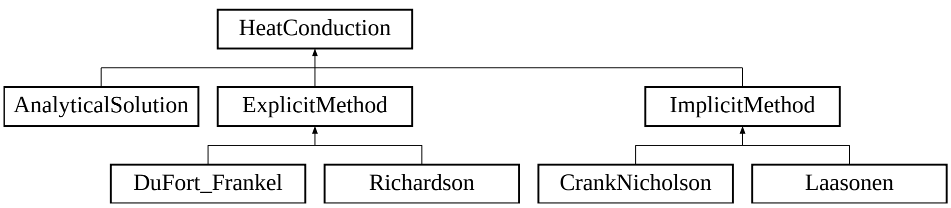
- **HeatConduction.h**
- **HeatConduction.cpp**

4.5 HeatConduction Class Reference

Base abstract Class which include all the parameters to solve the problem.

`#include <HeatConduction.h>`

Inheritance diagram for HeatConduction:



Public Member Functions

- **HeatConduction** (double **Tin_0**, double **Text_0**, double **Xmin**, double **Xmax**, double **Tend**, double **D**, double **dx**, double **dt**)

  *Constructor of the **HeatConduction** (*p. *15) class.*
- virtual void **solve** ()

  Abstract solve.
- std::vector< double > **get_u_n** () const

  Get method of the attribute u_n.

Protected Attributes

- double **Tin_0**

  initial condition Temperature
- double **Text_0**

  initial condition Temperature
- double **Xmin**

  initial condition Position
- double **Xmax**

  initial condition Position
- double **Tend**

  initial condition Time
- double **D**

  initial condition D
- double **dx**

  space step
- double **dt**

  time step
- int **n**

  number of time steps
- int **s**

  number of space steps
- double **r**

  calculation made once instead of multiple time
- std::vector< double > **u_nplus1**

  solution values vector n+1
- std::vector< double > **u_n**

  solution values vector n
- std::vector< double > **u_nminus1**

  solution values vector n-1

4.5.1 Detailed Description

Base abstract Class which include all the parameters to solve the problem.

Heat Conduction is an object, in which attributes is a paramaters of the problem, and also have vectors which will be used to store the solution. It includes an abstract method solve, which will call the solve methods corresponding to the type of scheme the user need.

Definition at line 27 of file HeatConduction.h.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 HeatConduction()

```
HeatConduction::HeatConduction (
            double Tin_0,
            double Text_0,
            double Xmin,
            double Xmax,
            double Tend,
            double D,
            double dx,
            double dt )
```

Constructor of the **HeatConduction** (p. 15) class.

Parameters

Tin_0	- initial condition Temperature inside
Text_0	- initial condition Temperature outside
Xmin	- the X position far left
Xmax	- the X position far right
Tend	- the end time of the simulation
D	- the difusivity of the wall
dx	- the space step
dt	- the time step

Definition at line 38 of file HeatConduction.cpp.

4.5.3 Member Function Documentation

4.5.3.1 get_u_n()

```
std::vector< double > HeatConduction::get_u_n ( ) const
```

Get method of the attribute u_n.

Parameters

none	

Returns

u_n - a vector attribute of the mother Class

Definition at line 73 of file HeatConduction.cpp.

4.5.3.2 solve()

```
void HeatConduction::solve ( )  [virtual]
```

Abstract solve.

Parameters

none	

Returns

void - the result is stored in the vector u_n of the mother Class

Reimplemented in **CrankNicholson** (p. 10), **Laasonen** (p. 22), **ImplicitMethod** (p. 20), **ExplicitMethod** (p. 15), and **AnalyticalSolution** (p. 8).

Definition at line 64 of file HeatConduction.cpp.

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

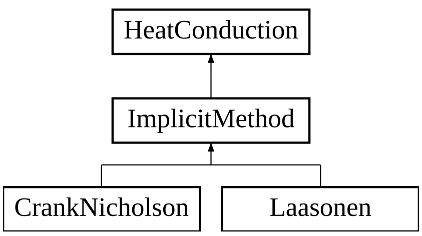
- **HeatConduction.h**
- **HeatConduction.cpp**

4.6 ImplicitMethod Class Reference

Sub Abstract Class used to calculate the Implicit scheme.

```
#include <HeatConduction.h>
```

Inheritance diagram for ImplicitMethod:



Public Member Functions

- **ImplicitMethod** (double **Tin_0**, double **Text_0**, double **Xmin**, double **Xmax**, double **Tend**, double **D**, double **dx**, double **dt**)

    Constructor of the ***ImplicitMethod*** *(p. 18) class.*
- virtual void **solve** ()

    Abstract solve.
- void **ThomasAlgorith** ()

    The Thomas Algorith, to solve Tridiagonal matrix problem.

Protected Attributes

- double **m**
- std::vector< double > **a**
- std::vector< double > **b**
- std::vector< double > **c**
- std::vector< double > **d**

4.6.1 Detailed Description

Sub Abstract Class used to calculate the Implicit scheme.

ImplicitMethod (p. 18) is a sub class of **HeatConduction** (p. 15). Both implicit method share the Thomas Algorith, which is implemented in this class. The solve method is an abstract method implemented in the sub sub classes.

Definition at line 85 of file HeatConduction.h.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 ImplicitMethod()

```
ImplicitMethod::ImplicitMethod (
            double Tin_0,
            double Text_0,
            double Xmin,
            double Xmax,
            double Tend,
            double D,
            double dx,
            double dt )
```

Constructor of the **ImplicitMethod** (p. 18) class.

Parameters

Tin↩_0	- initial condition Temperature inside
Text↩_0	- initial condition Temperature outside
Xmin	- the X position far left
Xmax	- the X position far right
Tend	- the end time of the simulation
D	- the difusivity of the wall

Definition at line 181 of file HeatConduction.cpp.

4.6.3 Member Function Documentation

4.6.3.1 solve()

```
void ImplicitMethod::solve ( ) [virtual]
```

Abstract solve.

Parameters

none	

Returns

void - the result is stored in the vector u_n of the mother Class

Reimplemented from **HeatConduction** (p. 18).

Reimplemented in **CrankNicholson** (p. 10), and **Laasonen** (p. 22).

Definition at line 203 of file HeatConduction.cpp.

4.6.3.2 ThomasAlgorith()

```
void ImplicitMethod::ThomasAlgorith ( )
```

The Thomas Algorith, to solve Tridiagonal matrix problem.

Parameters

none	

Returns

void - the result is stored in the vector u_n of the mother Class

Definition at line 212 of file HeatConduction.cpp.

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

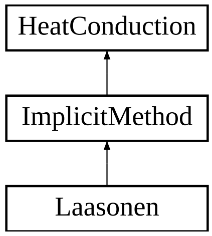
- **HeatConduction.h**
- **HeatConduction.cpp**

4.7 Laasonen Class Reference

Sub sub Class used to calculate the **Laasonen** (p. 21) scheme.

```
#include <HeatConduction.h>
```

Inheritance diagram for Laasonen:



Public Member Functions

- **Laasonen** (double **Tin_0**, double **Text_0**, double **Xmin**, double **Xmax**, double **Tend**, double **D**, double **dx**, double **dt**)

  *Constructor of the **Laasonen** (p. 21) class.*
- virtual void **solve** ()

  *Solve method. The matrix abc and the vector d are define after the **Laasonen** (p. 21) scheme.*

Additional Inherited Members

4.7.1 Detailed Description

Sub sub Class used to calculate the **Laasonen** (p. 21) scheme.

Laasonen (p. 21) is a sub class of **ImplicitMethod** (p. 18). It use the **Laasonen** (p. 21) scheme, an implicit scheme to calculate an Heat Conduction problem of a wall which have a temperature imposed at the extremities.

Definition at line 137 of file HeatConduction.h.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 Laasonen()

```
Laasonen::Laasonen (
            double Tin_0,
            double Text_0,
            double Xmin,
            double Xmax,
            double Tend,
            double D,
            double dx,
            double dt )
```

Constructor of the **Laasonen** (p. 21) class.

Parameters

Tin↵_0	- initial condition Temperature inside
Text↵_0	- initial condition Temperature outside
Xmin	- the X position far left
Xmax	- the X position far right
Tend	- the end time of the simulation
D	- the difusivity of the wall
dx	- the space step
dt	- the time step

Definition at line 300 of file HeatConduction.cpp.

4.7.3 Member Function Documentation

4.7.3.1 solve()

```
void Laasonen::solve ( )  [virtual]
```

Solve method. The matrix abc and the vector d are define after the **Laasonen** (p. 21) scheme.

Parameters

none	

Returns

void - the result is stored in the vector u_n of the mother Class

Reimplemented from **ImplicitMethod** (p. 20).

Definition at line 309 of file HeatConduction.cpp.

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

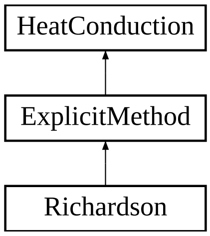
- **HeatConduction.h**
- **HeatConduction.cpp**

4.8 Richardson Class Reference

Sub sub Class used to calculate the **Richardson** (p. 22) scheme.

```
#include <HeatConduction.h>
```

Inheritance diagram for Richardson:



Public Member Functions

- **Richardson** (double **Tin_0**, double **Text_0**, double **Xmin**, double **Xmax**, double **Tend**, double **D**, double **dx**, double **dt**)

  Constructor of the ***Richardson*** *(p. 22) class.*
- virtual void **advance** (int i)

  Calcul of un_plus1 according to ***Richardson*** *(p. 22) scheme.*

Additional Inherited Members

4.8.1 Detailed Description

Sub sub Class used to calculate the **Richardson** (p. 22) scheme.

Richardson (p. 22) is a sub class of **ExplicitMethod** (p. 13). It use the **Richardson** (p. 22) scheme, a second order explicit scheme to calculate an Heat Conduction problem of a wall which have a temperature imposed at the extremities.

Definition at line 122 of file HeatConduction.h.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 Richardson()

```
Richardson::Richardson (
             double Tin_0,
             double Text_0,
             double Xmin,
             double Xmax,
             double Tend,
             double D,
             double dx,
             double dt )
```

Constructor of the **Richardson** (p. 22) class.

Parameters

Tin←↩_0	- initial condition Temperature inside
Text←↩_0	- initial condition Temperature outside
Xmin	- the X position far left
Xmax	- the X position far right
Tend	- the end time of the simulation
D	- the difusivity of the wall
dx	- the space step
dt	- the time step

Definition at line 274 of file HeatConduction.cpp.

4.8.3 Member Function Documentation

4.8.3.1 advance()

```
void Richardson::advance (
            int i ) [virtual]
```

Calcul of un_plus1 according to **Richardson** (p. 22) scheme.

Parameters

i	- the space iteration at which is the solve method

Returns

void - the result is stored in the vector u_nplus1 of the mother Class

Reimplemented from **ExplicitMethod** (p. 14).

Definition at line 283 of file HeatConduction.cpp.

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

- **HeatConduction.h**
- **HeatConduction.cpp**

Chapter 5

File Documentation

5.1 HeatConduction.cpp File Reference

Different objects to resolve an Heat Conduction problem.

```
#include "HeatConduction.h"
#include <cmath>
```

Variables

- const double **pi** = atan(1) $*$ 4

5.1.1 Detailed Description

Different objects to resolve an Heat Conduction problem.

Author

M Le Clec'h

Version

1.0

Date

05 December 2017

There are 4 schemes which can be use :

- The DuFort-Frankel scheme
- The **Richardson** (p. 22) scheme
- The **Laasonen** (p. 21) scheme
- The Crank-Nicholson scheme It can also provide the analytical solution.

5.2 HeatConduction.h File Reference

Different objects to resolve an Heat Conduction problem.

```
#include <vector>
```

Classes

- class **HeatConduction**
  Base abstract Class which include all the parameters to solve the problem.
- class **AnalyticalSolution**
  Sub Class used to calculate the analytical solution.
- class **ExplicitMethod**
  Sub Abstract Class used to calculate the Explicit scheme.
- class **ImplicitMethod**
  Sub Abstract Class used to calculate the Implicit scheme.
- class **DuFort_Frankel**
  Sub sub Class used to calculate the ***DuFort_Frankel*** *(p. 11) scheme.*
- class **Richardson**
  Sub sub Class used to calculate the ***Richardson*** *(p. 22) scheme.*
- class **Laasonen**
  Sub sub Class used to calculate the ***Laasonen*** *(p. 21) scheme.*
- class **CrankNicholson**
  Sub sub Class used to calculate the Crank-Nicholson scheme.

5.2.1 Detailed Description

Different objects to resolve an Heat Conduction problem.

Author

M Le Clec'h

Version

1.0

Date

05 December 2017

There are 4 schemes which can be use :

- The DuFort-Frankel scheme
- The **Richardson** (p. 22) scheme
- The **Laasonen** (p. 21) scheme
- The Crank-Nicholson scheme It can also provide the analytical solution.

5.3 Norms.cpp File Reference

Functions to calculates norms.

```
#include "Norms.h"
```

Functions

- double **norm_one** (std::vector< double > solution)
  Function to calculate the first norm.
- double **norm_two** (std::vector< double > solution)
  Function to calculate the Euclidean norm.
- double **norm_uniform** (std::vector< double > solution)
  Function to calculate the Infinite norm.

5.3.1 Detailed Description

Functions to calculates norms.

Author

M Le Clec'h

Version

1.0

Date

05 December 2017

There are 3 norms which can be calculated :

- The norm one
- The norm two
- The uniform norm

5.3.2 Function Documentation

5.3.2.1 norm_one()

```
norm_one (
            std::vector< double > solution )
```

Function to calculate the first norm.

Parameters

solution	Vector object on which we need to calculate the norm one.

Returns

sum The result of the calculation.

Definition at line 23 of file Norms.cpp.

5.3.2.2 norm_two()

```
norm_two (
            std::vector< double > solution )
```

Function to calculate the Euclidean norm.

Parameters

solution	Vector object on which we need to calculate the second one.

Returns

sum The result of the calculation.

Definition at line 38 of file Norms.cpp.

5.3.2.3 norm_uniform()

```
norm_uniform (
            std::vector< double > solution )
```

Function to calculate the Infinite norm.

Parameters

solution	Vector object on which we need to calculate the uniform one.

Returns

sum The result of the calculation.

Definition at line 53 of file Norms.cpp.

5.4 Norms.h File Reference

Functions to calculates norms.

```
#include <vector>
```

Functions

- double **norm_one** (std::vector< double > solution)
  Function to calculate the first norm.
- double **norm_two** (std::vector< double > solution)
  Function to calculate the Euclidean norm.
- double **norm_uniform** (std::vector< double > solution)
  Function to calculate the Infinite norm.

5.4.1 Detailed Description

Functions to calculates norms.

Author

M Le Clec'h

Version

1.0

Date

05 December 2017

There are 3 norms which can be calculated :

- The norm one
- The norm two
- The uniform norm

5.4.2 Function Documentation

5.4.2.1 norm_one()

```
double norm_one (
            std::vector< double > solution )
```

Function to calculate the first norm.

Parameters

solution	Vector object on which we need to calculate the norm one.

Returns

sum The result of the calculation.

Definition at line 23 of file Norms.cpp.

5.4.2.2 norm_two()

```
double norm_two (
            std::vector< double > solution )
```

Function to calculate the Euclidean norm.

Parameters

solution	Vector object on which we need to calculate the second one.

Returns

sum The result of the calculation.

Definition at line 38 of file Norms.cpp.

5.4.2.3 norm_uniform()

```
double norm_uniform (
            std::vector< double > solution )
```

Function to calculate the Infinite norm.

Parameters

solution	Vector object on which we need to calculate the uniform one.

Returns

sum The result of the calculation.

Definition at line 53 of file Norms.cpp.