HeatConduction

Generated by Doxygen 1.8.13

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

File Index

3.1 File List

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

Chapter 4

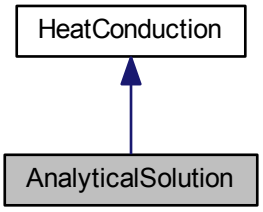
Class Documentation

4.1 AnalyticalSolution Class Reference

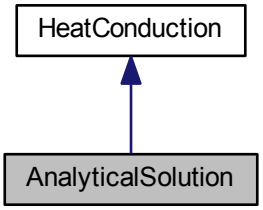
Sub Class used to calculate the analytical solution.

`#include <HeatConduction.h>`

Inheritance diagram for AnalyticalSolution:



Collaboration 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 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 is a sub class of HeatConduction. 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 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 92 of file HeatConduction.cpp.

4.1.3 Member Function Documentation

4.1.3.1 solve()

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

Solve with the analytical solution.

Returns

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

Reimplemented from HeatConduction.

Definition at line 100 of file HeatConduction.cpp.

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

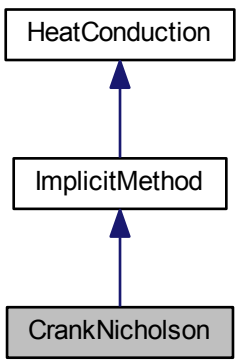
- HeatConduction.h
- HeatConduction.cpp

4.2 CrankNicholson Class Reference

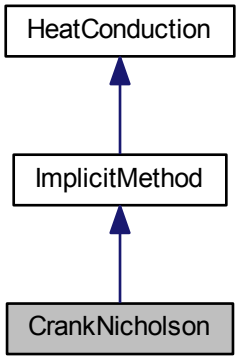
Sub sub Class used to calculate the Crank-Nicholson scheme.

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

Inheritance diagram for CrankNicholson:



Collaboration 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 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 is a sub class of ImplicitMethod. 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 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 340 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.

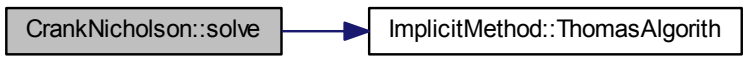
Returns

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

Reimplemented from ImplicitMethod.

Definition at line 348 of file HeatConduction.cpp.

Here is the call graph for this function:



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

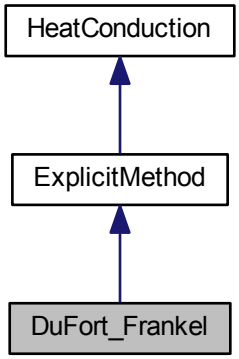
- HeatConduction.h
- HeatConduction.cpp

4.3 DuFort_Frankel Class Reference

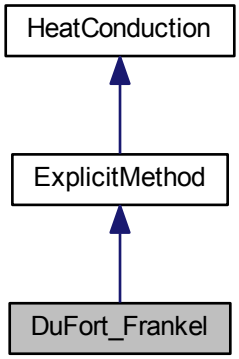
Sub sub Class used to calculate the DuFort_Frankel scheme.

`#include <HeatConduction.h>`

Inheritance diagram for DuFort_Frankel:



Collaboration 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 class.
- virtual void advance (int i)

  Calcul of un_plus1 according to DuFort_Frankel scheme.

Additional Inherited Members

4.3.1 Detailed Description

Sub sub Class used to calculate the DuFort_Frankel scheme.

DuFort_Frankel is a sub class of ExplicitMethod. 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 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 242 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 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.

Definition at line 251 of file HeatConduction.cpp.

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

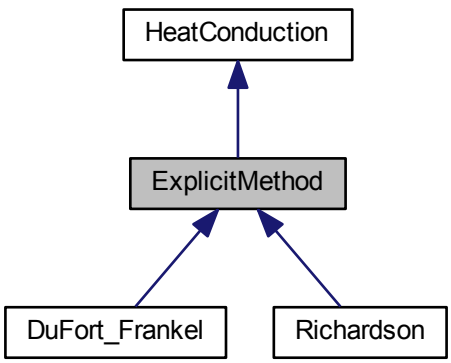
- HeatConduction.h
- HeatConduction.cpp

4.4 ExplicitMethod Class Reference

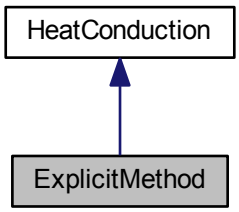
Sub Abstract Class used to calculate the Explicit scheme.

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

Inheritance diagram for ExplicitMethod:



Collaboration 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 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 is a sub class of HeatConduction. 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 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 126 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

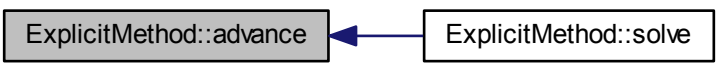
Returns

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

Reimplemented in Richardson, and DuFort_Frankel.

Definition at line 135 of file HeatConduction.cpp.

Here is the caller graph for this function:



4.4.3.2 solve()

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

Solve regroup the common part of the Explicit Method.

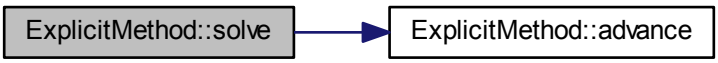
Returns

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

Reimplemented from HeatConduction.

Definition at line 143 of file HeatConduction.cpp.

Here is the call graph for this function:



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

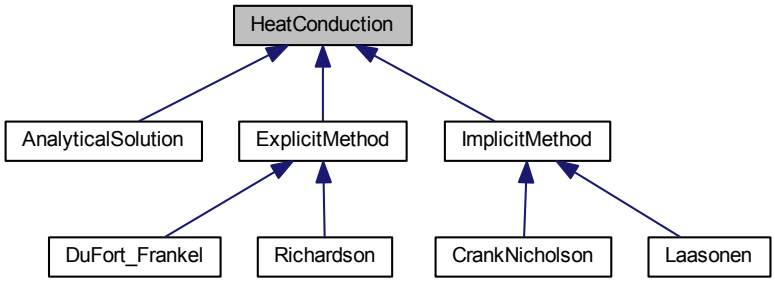
- HeatConduction.h
- HeatConduction.cpp

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 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 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.

Returns

u_n - a vector attribute of the mother Class

Definition at line 71 of file HeatConduction.cpp.

4.5.3.2 solve()

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

Abstract solve.

Returns

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

Reimplemented in CrankNicholson, Laasonen, ImplicitMethod, ExplicitMethod, and AnalyticalSolution.

Definition at line 63 of file HeatConduction.cpp.

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

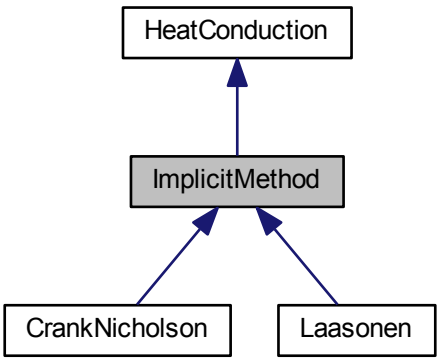
- HeatConduction.h
- HeatConduction.cpp

4.6 ImplicitMethod Class Reference

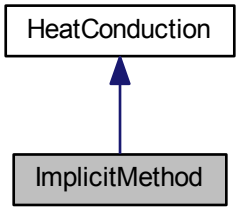
Sub Abstract Class used to calculate the Implicit scheme.

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

Inheritance diagram for ImplicitMethod:



Collaboration 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 class.
- virtual void solve ()

  Abstract solve.
- void ThomasAlgorith ()

  The Thomas Algorith, to solve Tridiagonal matrix problem.

Protected Attributes

- double m

  var needed in the Thomas Algorithm
- std::vector< double > a

  lower tridiagonal vector of the matrix
- std::vector< double > b

  middle tridiagonal vector of the matrix
- std::vector< double > c

  upper tridiagonal vector of the matrix
- std::vector< double > d

  vector on the right of the equation

4.6.1 Detailed Description

Sub Abstract Class used to calculate the Implicit scheme.

ImplicitMethod is a sub class of HeatConduction. 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 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 177 of file HeatConduction.cpp.

4.6.3 Member Function Documentation

4.6.3.1 solve()

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

Abstract solve.

Returns

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

Reimplemented from HeatConduction.

Reimplemented in CrankNicholson, and Laasonen.

Definition at line 198 of file HeatConduction.cpp.

4.6.3.2 ThomasAlgorith()

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

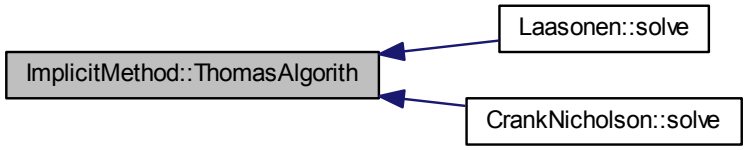
The Thomas Algorith, to solve Tridiagonal matrix problem.

Returns

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

Definition at line 206 of file HeatConduction.cpp.

Here is the caller graph for this function:



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

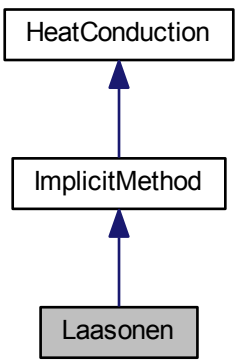
- HeatConduction.h
- HeatConduction.cpp

4.7 Laasonen Class Reference

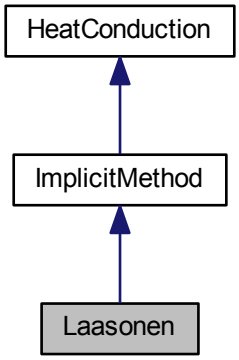
Sub sub Class used to calculate the Laasonen scheme.

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

Inheritance diagram for Laasonen:



Collaboration 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 class.
- virtual void solve ()

  Solve method. The matrix abc and the vector d are define after the Laasonen scheme.

Additional Inherited Members

4.7.1 Detailed Description

Sub sub Class used to calculate the Laasonen scheme.

Laasonen is a sub class of ImplicitMethod. It use the Laasonen 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 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 294 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 scheme.

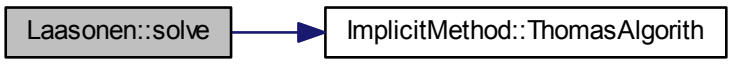
Returns

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

Reimplemented from ImplicitMethod.

Definition at line 302 of file HeatConduction.cpp.

Here is the call graph for this function:



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

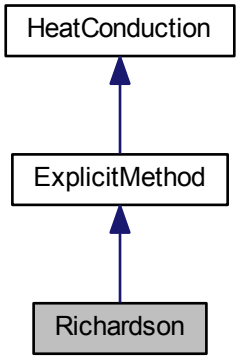
- HeatConduction.h
- HeatConduction.cpp

4.8 Richardson Class Reference

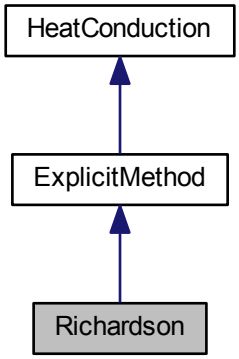
Sub sub Class used to calculate the Richardson scheme.

`#include <HeatConduction.h>`

Inheritance diagram for Richardson:



Collaboration 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 class.
- virtual void advance (int i)

  Calcul of un_plus1 according to Richardson scheme.

Additional Inherited Members

4.8.1 Detailed Description

Sub sub Class used to calculate the Richardson scheme.

Richardson is a sub class of ExplicitMethod. It use the Richardson 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 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 268 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 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.

Definition at line 277 of file HeatConduction.cpp.

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

- HeatConduction.h
- HeatConduction.cpp

Chapter 5

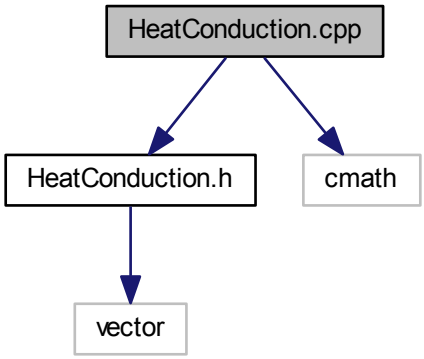
File Documentation

5.1 HeatConduction.cpp File Reference

Different objects to resolve an Heat Conduction problem.

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

Include dependency graph for HeatConduction.cpp:



Variables

- const double pi = atan(1) ∗ 4

  define pi

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 :

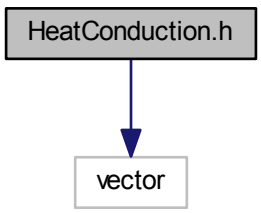
- The DuFort-Frankel scheme
- The Richardson scheme
- The Laasonen scheme
- The Crank-Nicholson scheme It can also provide the analytical solution.

5.2 HeatConduction.h File Reference

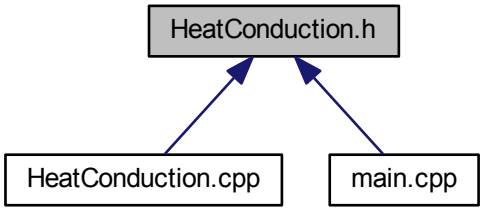
Different objects to resolve an Heat Conduction problem.

```
#include <vector>
```

Include dependency graph for HeatConduction.h:



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



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 scheme.
- class Richardson

  Sub sub Class used to calculate the Richardson scheme.
- class Laasonen

  Sub sub Class used to calculate the Laasonen 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 :

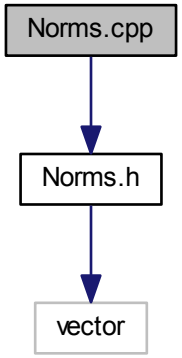
- The DuFort-Frankel scheme
- The Richardson scheme
- The Laasonen scheme
- The Crank-Nicholson scheme It can also provide the analytical solution.

5.3 Norms.cpp File Reference

Functions to calculates norms.

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

Include dependency graph for Norms.cpp:



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.

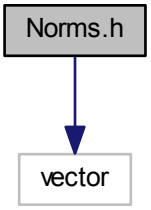
Definition at line 53 of file Norms.cpp.

5.4 Norms.h File Reference

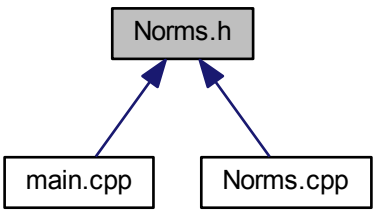
Functions to calculates norms.

```
#include <vector>
```

Include dependency graph for Norms.h:



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



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.