

Computational Methods & C++ Assignment

Augustin Reille

2 November 2017

Abstract

A one-space dimensional problem is considered, to examine the application of numerical schemes for the solution of partial differential equations.

Contents

1	Introduction	4
2	Methods	5
2.1	Explicit schemes	6
2.1.1	Dufort-Frankel	6
2.1.2	Richardson	6
2.2	Implicit schemes	6
2.2.1	Laasonen	6
2.2.2	Crank-Nicholson	7
2.3	Analytical solution	8
2.4	Object Oriented Design	8
3	Results	10
3.1	Schemes study	10
3.2	Laasonen method : effect of the step time	12
4	Discussion	13
5	Future work	14
6	Conclusion	15
7	Aknowledgments	16
8	References	17
9	Appendices	18

List of Figures

1	UML Diagram of the C++ object oriented code	8
2	Results with differents schemes at $t = 0,1h$	10
3	Results with differents schemes at different times	11

List of Tables

1	Errors for different schemes at $t = 0,1h$	10
2	Errors for different schemes at different times	11

1 Introduction

2 Methods

Several methods are used :

- Dufort-Frankel
- Richardson
- Laasonen simple implicit
- Crank-Nicholson

For all of the used schemes, the initialization of the result matrix is run the same way. First the study grid must be chosen. The initials conditions are:

- T_{int} of 100°F
- T_{sur} of 300°F
- $D = 0.1 ft^2/hr$

First it was assumed that $\Delta x = 0.05$ and $\Delta t = 0.01$, so that the result matrix could be initialized. We set:

$$n_{space} = \frac{L}{\Delta x} + 1 \quad (1)$$

$$n_{time} = \frac{T}{\Delta t} \quad (2)$$

n_{space} is the number of iterations over the grid, according to the coordinate x .
 n_{time} is the number of iterations over the grid, according to the time t .

$$\begin{pmatrix} i=0 & i=1 & \cdots & n_{space}-1 & n_{space} \\ T_{ext} & T_{int} & \cdots & T_{int} & T_{ext} \\ T_{ext} & 0 & \cdots & 0 & T_{ext} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ T_{ext} & 0 & \cdots & 0 & T_{ext} \end{pmatrix} \begin{matrix} j=0 \\ j=1 \\ \vdots \\ n_{time} \end{matrix}$$

This is how the matrix is initialized, according to initial conditions. For most of the used schemes, we need two previous time steps to calculate the current step. An order one scheme is used to find the second line of the matrix:

$$T_{i,1} = \frac{\alpha}{2}(T_{i+1,0} + T_{i-1,0}) + (1 - \alpha)T_{i,0} \quad (3)$$

with

$$\alpha = 2 \frac{D\delta t}{\delta x^2}$$

So we have:

$$\begin{matrix} i = 0 & i = 1 & \cdots & n_{space} - 1 & n_{space} \\ \begin{pmatrix} T_{ext} & T_{int} & \cdots & T_{int} & T_{ext} \\ T_{ext} & T_{i,1} & \cdots & T_{i,1} & T_{ext} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ T_{ext} & 0 & \cdots & 0 & T_{ext} \end{pmatrix} & \begin{matrix} j = 0 \\ j = 1 \\ \vdots \\ n_{time} \end{matrix} \end{matrix}$$

With this trick our matrix is ready to be used and filled up with every scheme.

2.1 Explicit schemes

For both of the explicit schemes used (Dufort-Frankel and Richardson schemes), the solving is run the same way : with a double for-loop, the matrix is filled up. The only thing that change is the way we advance in time and space.

2.1.1 Dufort-Frankel

Explicitly, for Dufort-Frankel method, we have:

$$T_j^{n+1} = \left(\frac{1-\alpha}{1+\alpha}\right)T_j^{n-1} + \left(\frac{\alpha}{1+\alpha}\right)(T_{j+1}^n + T_{j-1}^n) \quad (4)$$

with

$$\alpha = 2\frac{D\delta t}{\delta x^2}$$

2.1.2 Richardson

Explicitly, for Richardson method, we have:

$$T_j^{n+1} = T_j^{n-1} + \alpha(T_{j-1}^n - 2T_j^n + T_{j+1}^n) \quad (5)$$

with

$$\alpha = 2\frac{D\delta t}{\delta x^2}$$

2.2 Implicit schemes

For both implicit schemes, the vector solution is given by a matrix equation. A *LU* factorization is used once, before a for-loop, to resolve the system for each space step.

2.2.1 Laasonen

We have :

$$\frac{T_j^{n+1} - T_j^n}{\Delta t} = D \frac{T_{j+1}^{n+1} - 2T_j^{n+1} + T_{j-1}^{n+1}}{\Delta x^2} \quad (6)$$

Written in a matrical way, the solution at space step $n + 1$ is given by :

$$\begin{bmatrix} 1+2C & -C & 0 & \cdots & 0 \\ -C & 1+2C & -C & \ddots & \vdots \\ 0 & -C & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 1+2C & -C \\ 0 & \cdots & 0 & -C & 1+2C \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_i \\ \vdots \\ T_{ntime} \end{bmatrix}_{n+1} = \begin{bmatrix} T_1 + CT_0 \\ T_2 \\ \vdots \\ T_i \\ \vdots \\ T_{ntime} + CT_f \end{bmatrix}_n \quad (7)$$

with

$$C = \frac{D\Delta t}{\Delta x^2}$$

The first vector at space n is our boundaries conditions vector, so it is known. In this scheme we must be aware that the right-term vector is actually bigger than the other, of 2 values, which corresponds to the exterior temperatures, which stays the same all the time.

We apply the LU solve algorithm for each space step, at the reduced vector, and add the $n + 1$ vector to our result matrix. The new right term vector is the one found with the LU solve algorithm.

2.2.2 Crank-Nicholson

We have :

$$\frac{T_j^{n+1} - T_j^n}{\Delta t} = D \frac{1}{2} \left(\frac{T_{j+1}^{n+1} - 2T_j^{n+1} + T_{j-1}^{n+1}}{\Delta x^2} + \frac{T_{j+1}^n - 2T_j^n + T_{j-1}^n}{\Delta x^2} \right) \quad (8)$$

Written in a matrical way, the solution at space step $n + 1$ is given by :

$$\begin{bmatrix} 1+C & -C/2 & 0 & \cdots & 0 \\ -C/2 & 1+C & -C/2 & \ddots & \vdots \\ 0 & -C/2 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 1+C & -C/2 \\ 0 & \cdots & 0 & -C/2 & 1+C \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \\ \vdots \\ T_i \\ \vdots \\ T_{ntime} \end{bmatrix}_{n+1} = \begin{bmatrix} 1-C & C/2 & 0 & \cdots & 0 \\ C/2 & 1-C & C/2 & \ddots & \vdots \\ 0 & C/2 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 1-C & C/2 \\ 0 & \cdots & 0 & C/2 & 1-C \end{bmatrix} \begin{bmatrix} T_1 + CT_0 \\ T_2 \\ \vdots \\ T_i \\ \vdots \\ T_{ntime} + CT_f \end{bmatrix}_n \quad (9)$$

with

$$C = \frac{D\Delta t}{\Delta x^2}$$

The method is the same as Laasonen scheme, except that the left term vector found with the *LU* solve algorithm is multiplied by the right term matrix before the next space step.

2.3 Analytical solution

The analytical solution of the problem considered will be used to compare our results and to calculate the errors. The result of the analytical solution for a time t and a position x is given by :

$$T = T_{ext} + 2(T_{int} - T_{ext}) \sum_{m=1}^{m=\infty} e^{-D(m\pi/L)^2 t} \frac{1 - (-1)^m}{m\pi} \sin\left(\frac{m\pi x}{L}\right) \quad (10)$$

2.4 Object Oriented Design

The object oriented classes for the C++ code are designed the following way :

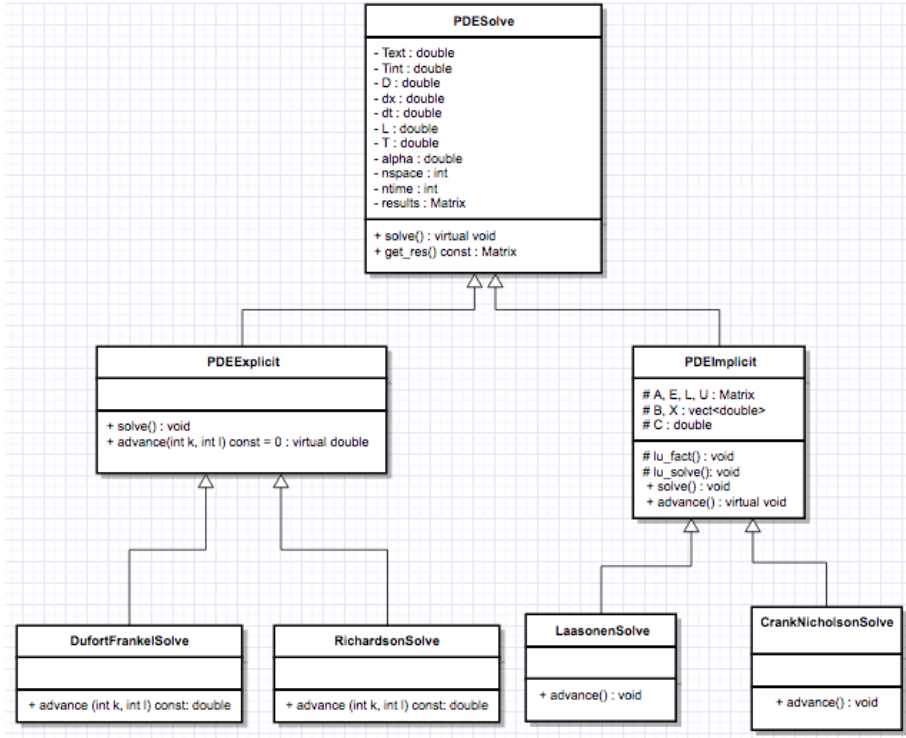


Figure 1: UML Diagram of the C++ object oriented code

There is a main base class called PDESolve, which initializes all the variables, and two functions :

- `get_res()` : returns the results Matrix
- `solve()` : virtual function which will be overwritten in subclasses

There is two subclasses of `PDESolve` : `PDEExplicit` and `PDEImplicit`. Each one has a `solve()` function, and a virtual advance function, which corresponds at the way the space and time steps are incremented. The `PDEImplicit` class also initializes all the matrixes and functions needed for the LU factorization and solve. Each subclass, which corresponds to each method, has its own particular advance method.

3 Results

3.1 Schemes study

Note that results from Richardson scheme are not displayed on the following plots, due to its instability : the incorrect values polluted our charts.

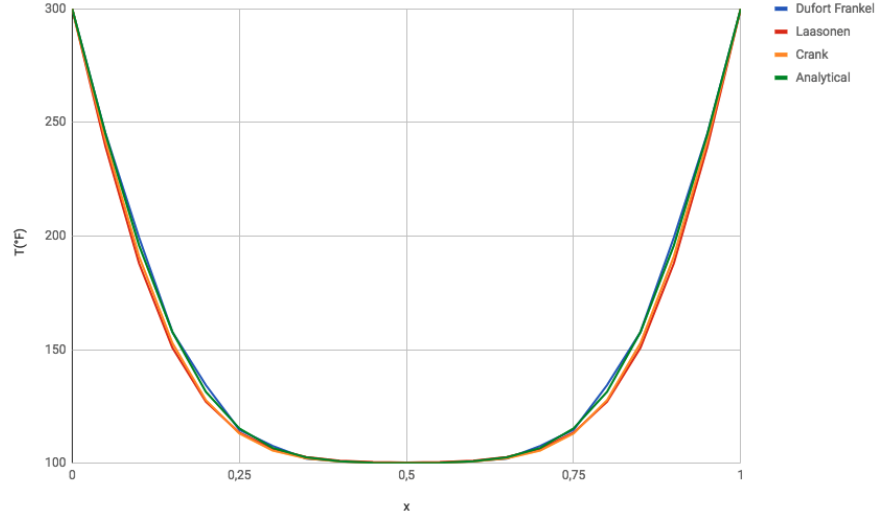


Figure 2: Results with differents schemes at $t = 0,1h$

We can see that except for the Richardson scheme, the results are quite the same. The errors had been calculated to evaluate the accuracy of each method.

Table 1: Errors for different schemes at $t = 0,1h$

Errors	Dufort-Frankel	Laasonen	Crank-Nicholson
1-norm	0,56%	1,72%	1,26%
2-norm	24,43%	2,35%	1,61%
∞ -norm	1,18%	2,62%	1,61%

We observe that for the 1-norm and the ∞ -norm, Dufort-Frankel method is the most accurate, followed by the Crank-Nicholson method then by the Laasonen method. Let see the evolution of the error while time is increased.

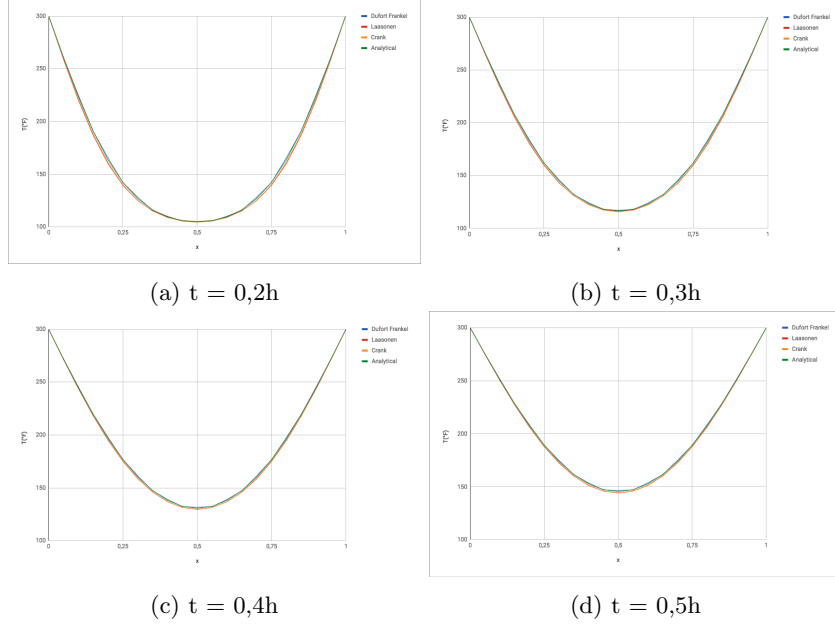


Figure 3: Results with different schemes at different times

Table 2: Errors for different schemes at different times

Errors	Dufort-Frankel	Laasonen	Crank-Nicholson
$t = 0,2h$			
1-norm	0,37%	1,05%	0,80%
2-norm	14,11%	1,25%	0,87%
∞ -norm	0,63%	1,28%	0,81%
$t = 0,3h$			
1-norm	0,27%	0,77%	0,60%
2-norm	10,50%	0,84%	0,62%
∞ -norm	0,42%	0,83%	0,54%
$t = 0,4h$			
1-norm	0,22%	0,64%	0,49%
2-norm	8,84%	0,67%	0,52%
∞ -norm	0,34%	0,60%	0,43%
$t = 0,5h$			
1-norm	0,18%	0,55%	0,42%
2-norm	7,86%	0,58%	0,45%
∞ -norm	0,32%	0,51%	0,41%

We observe that the error is globally decreasing as time increase. The most accurate methods appears to be the Crank-Nicholson method, then the Laasonen method.

3.2 Laasonen method : effect of the step time

4 Discussion

5 Future work

6 Conclusion

7 Acknowledgments

8 References

9 Appendices