A numerical study of the diffusion/heat equation in one and two dimensions.

Written by:

Marius Enga, Patryk Krzyzaniak and Kristoffer Varslott

Department of Physics UiO

December 19, 2019

Contents

Abstract

This study aims to numerically solve the diffusion/heat equation for one and two dimensions. There is 3 different numerical methods that will be used, the explicit forward Euler, the implicit backward Euler and the Crank-Nicolson method. In one dimension the system reaches a steady-state quickly for all methods, where Crank-Nicolson was the numerical method with highest accuracy. For two dimensions, both the explicit and implicit Euler methods were used, were the absolute error was about 50% *lower for backwards Euler. We found that the methods which did not have a stability requirement in terms of the choice of dt and dx, far outperformed forward Euler inn all cases.*

I. Introduction

One of the most important partial differential equations (PDE) that occurs across many fields of science, is the diffusion/heat equation. Physiology, material science, geophysics, thermophysics and defect chemistry, are just a few examples where these PDEs are used and shows the broad applicability of these PDEs. The general form of these two equations are given by the following formula:

$$C\nabla^2\phi(\vec{r},t) = \frac{\partial\phi(\vec{r},t)}{\partial t}$$

This is a 3 spatial dimensional and time dependent PDE, where the constant C is the diffusion coefficient D with units m^2/s for the diffusion equation. For the heat equation, this constant becomes $C = \frac{\kappa}{c\rho}$, where κ, c and ρ are the thermal conductivity, specific heat capacity and the mass density respectively. We are however only interested in solving the PDE itself numerically, so we set the constant to 1. The diffusion equation and heat equation are now equal and will be solved in 1 and 2 dimensions.

In one dimension we will study the case where both ends are fixed at different temperatures (concentrations for diffusion), and see how the temperature distributes itself as time increases. Boundary conditions, initial condition and solution are given in appendix A. For two dimensions we will study a case where all the edges are zero with an initial distribution that will flatten out when time increases. See appendix B for solution and more details.

II. Method

In order to get an overview of which methods being used we take a closer look at the different types of methods in this section. Where we will look at three different types of methods for solving the diffusion equation [1]. There are three methods being implemented in this text - Forward Euler, Backward Euler and Crank-Nicolson. All in which is slightly different from one another.

i. **Explicit Scheme***: Forward Euler*

To start of we look at the Forward Euler method for solving partial differential equation, such as the diffusion equation [1].

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial u(x,t)}{\partial t} \tag{1}$$

This equation can be solved through a central difference approximation. If we look at the time-dependent derivative we get:

$$\frac{\partial u(x,t)}{\partial t} = u_t = \frac{u(x_i, t_j + \Delta t) - u(x_i, t_j)}{\Delta t} + O(\Delta t) \tag{2}$$

This expression is only exact if it contains the local approximation error of $O(\Delta t)$. Where Δt is the time-step, and t_j is the total time after j time steps. The same goes for step-length:

$$\begin{aligned} t_j &= j\Delta t \qquad j > 0 \\ x_i &= i\Delta x \qquad 0 \leq i \leq n+1 \end{aligned}$$

Where i & j is the iterations of steps and time respectively. The step-length Δx is defined as:

$$\Delta x = \frac{1}{n+1}$$

Further on, we need to approximate the second derivative which is dependent on the position.

$$\frac{\partial^2 u(x,t)}{\partial x^2} = u_{xx} = \frac{u(x_i + \Delta x, t_j) - 2u(x_i, t_j) + u(x_i - \Delta x, t_j)}{\Delta x^2} + O(\Delta x^2) \tag{3}$$

Where in this case we have an local approximation error of $O(\Delta x^2)$. Once we have the equations [2] [3], we can discretize these for simplifications. We can now write equation [1] in a discretized version:

$$\frac{u_{i,j+1} - u_{i,j}}{\Delta t} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2}$$

Rearranging this equation so that we can find the solution of the next time $j + 1$:

$$u_{i,j+1} = \alpha u_{i+1,j} + (1 - 2\alpha)u_{i,j} + \alpha u_{i-1,j} \tag{4}$$

Where we now have define $\alpha = \frac{\Delta t}{\Delta x^2}$. Now we would like to investigate the boundary conditions which is applied to our system. At the time $t = 0$, we have the solution:

$$u(x, 0) = u_{i,0} = 0 \qquad 0 < x < L$$

And for the boundary condition of the x-region, which stretches from $0 \rightarrow$ L, where L = 1, we get:

$$\begin{aligned} u(0,t) &= u_{0,j} = 0 \qquad & t \geq 0 \\ u(L,t) &= u_{n+1,j} = 1 \qquad & t \geq 0 \end{aligned}$$

Where n + 1 indicates the last iteration in space, yielding $L = x_{n+1} = (n + 1)\Delta x = 1$, since $\Delta x = 1/(n + 1)$. Once we have the boundary conditions of the diffusion equation we are able to rewrite it as a vector $\mathbf{V}_j$, which is dependent of time.

$$\mathbf{V}_j = \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ \dots \\ \dots \\ \dots \\ u_{n,j} \end{bmatrix}$$

If we take a closer look at equation 9 we can see that we can create a matrix consisting of α and -2α:

$$\mathbf{A}_{FE} = \begin{bmatrix} 1-2\alpha & \alpha & 0 & \dots & \dots & 0 \\ \alpha & 1-2\alpha & \alpha & 0 & \dots & \dots \\ 0 & \alpha & 1-2\alpha & \alpha & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \alpha & 1-2\alpha & \alpha \\ 0 & \dots & \dots & 0 & \alpha & 1-2\alpha \end{bmatrix}$$

By this we may write the discretized version of the diffusion equation in matrix form.

$$\begin{bmatrix} u_{1,j+1} \\ u_{2,j+1} \\ \dots \\ \dots \\ \dots \\ u_{n,j+1} \end{bmatrix} = \begin{bmatrix} 1-2\alpha & \alpha & 0 & \dots & \dots & 0 \\ \alpha & 1-2\alpha & \alpha & 0 & \dots & \dots \\ 0 & \alpha & 1-2\alpha & \alpha & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \alpha & 1-2\alpha & \alpha \\ 0 & \dots & \dots & 0 & \alpha & 1-2\alpha \end{bmatrix} \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ \dots \\ \dots \\ \dots \\ u_{n,j} \end{bmatrix}$$

In closed matrix form it reads:

$$\mathbf{V}_{j+1} = \mathbf{A}_{FE}\mathbf{V}_j \tag{5}$$

Further on we may rewrite $\mathbf{A}_{FE}$ as an identity matrix and a tridiagonal Toeplitz matrix $\mathbf{B}$, where $\mathbf{B}$ is:

$$\mathbf{B} = \begin{bmatrix} 2 & -1 & 0 & \dots & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & \dots \\ 0 & -1 & 2 & -1 & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & -1 & 2 & -1 \\ 0 & \dots & \dots & 0 & -1 & 2 \end{bmatrix}$$

$$\therefore \quad \mathbf{V}_{j+1} = (\hat{I} - \alpha\mathbf{B})\mathbf{V}_j \tag{6}$$

This is the Forward Euler method, the basic idea is that by knowing the previously calculated $\mathbf{V}_j$, it is possible to calculate the next $\mathbf{V}_{j+1}$. This is called the explicit scheme, since the the next function $u_{i,j+1}$ is explicitly given by equation [9].

ii. **Implicit Scheme**: *Backward Euler*

Backward Euler exhibits very much the same approach as Forward Euler. We start by approximating the the diffusion equation at hand.

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial u(x,t)}{\partial t}, t > 0, x \in [0, L]$$

Dividing up into a time-dependent derivative part and a space-dependent derivative part. The difference, however, is that this methods exploits the backwards formula. The time-dependent derivative is then formulated as such:

$$\frac{\partial u(x,t)}{\partial t} = u_t = \frac{u(x_i, t_j) - u(x_i, t_j - \Delta t)}{\Delta t} + O(\Delta t) \tag{7}$$

Where the truncation error $O(\Delta t)$, just as for the Forward Euler method. Step-length Δx and time-step Δt has the same definition as defined by Forward Euler method. The space-dependent derivative is written in the same way as the Forward Euler method.

$$\frac{\partial^2 u(x,t)}{\partial x^2} = u_{xx} = \frac{u(x_i + \Delta x, t_j) - 2u(x_i, t_j) + u(x_i - \Delta x, t_j)}{\Delta x^2} + O(\Delta x^2) \tag{8}$$

The same truncation error is added just as for the Forward Euler method. Once these equation is written out we discretize just as earlier.

$$\frac{u_{i,j} - u_{i,j-1}}{\Delta t} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2}$$

Rearranging this equation so that we can find the solution of previous time $j-1$ - hence Backward Euler.

$$u_{i,j-1} = -\alpha u_{i-1,j} + (1+2\alpha)u_{i,j} - \alpha u_{i+1,j} \tag{9}$$

Where we have defined α as previously, namely as $\frac{\Delta t}{\Delta x^2}$. Just as last section we can define a matrix $\mathbf{A}_{BE}$, subscript BE for Backward Euler.

$$\mathbf{A}_{BE} = \begin{bmatrix} 1+2\alpha & -\alpha & 0 & \dots & \dots & 0 \\ -\alpha & 1+2\alpha & -\alpha & 0 & \dots & \dots \\ 0 & -\alpha & 1+2\alpha & -\alpha & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & -\alpha & 1+2\alpha & -\alpha \\ 0 & \dots & \dots & 0 & -\alpha & 1+2\alpha \end{bmatrix}$$

We are using the same boundary conditions as last section. We can then also define the same vector $\mathbf{V}_j$, and compactly write the diffusion equation in matrix form.

$$\mathbf{V}_j = \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ \dots \\ \dots \\ \dots \\ u_{n,j} \end{bmatrix}$$

In compact matrix form we get a slightly different answer than the Forward Euler method, since we know are looking the the previous solution of time $u_{i,j-1}$:

$$\mathbf{V}_{j-1} = \mathbf{A}_{BE}\mathbf{V}_j \tag{10}$$

Where the $\mathbf{A}_{BE}$ may be rewritten as a tridagonal Toeplitz matrix $\mathbf{B}$. Little matrix algebra means we can rewrite the equation.

$$\mathbf{V}_{j-1} = (\hat{I} + \alpha\mathbf{B})\mathbf{V}_j \tag{11}$$

where $\mathbf{B}$ is defined as:

$$\mathbf{B} = \begin{bmatrix} 2 & -1 & 0 & \dots & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & \dots \\ 0 & -1 & 2 & -1 & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & -1 & 2 & -1 \\ 0 & \dots & \dots & 0 & -1 & 2 \end{bmatrix}$$

$$\therefore \quad \mathbf{V}_j = (\hat{I} + \alpha\mathbf{B})^{-1}\mathbf{V}_{j-1} \tag{12}$$

We can see that this method is different than the Forward Euler method, since this method relies on determining the vector $u_{i,j-1}$, instead of $u_{i,j+1}$, therefore it is called an implicit scheme.

iii. Crank-Nicolson scheme

The third and final method is the Crank-Nicolson scheme. The basic idea here is to combine the Forward Euler and Backward Euler in some sense. So for a more general approach than derived previously we can find a way to express the diffusion equation as a combination of explicit and implicit schemes. This gives us the so-called Crank-Nicolson Scheme, after Crank and Nicolson.

We begin with by introducing a parameter θ ($\theta - rule$), and and using θ to express both the explicit and implicit scheme, depending on the choice of θ:

Explicit scheme- Forward Euler

$$\frac{u_{i,j+1} - u_{i,j}}{\Delta t} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2}$$

Implicit scheme- Backward Euler

$$\frac{u_{i,j} - u_{i,j-1}}{\Delta t} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2}$$

These equations was derived in the previous subsection, now, by combining these and applying the $\theta - rule$ we get:

$$\frac{u_{i,j} - u_{i,j-1}}{\Delta t} = \frac{\theta}{\Delta x^2}(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) + \frac{1-\theta}{\Delta x^2}(u_{i+1,j-1} - 2u_{i,j-1} + u_{i-1,j-1}) \tag{13}$$

We can see by equation 13, that by setting $\theta = 0$, we simply end up with the Forward Euler - explicit scheme. The time-part indexed by j is moved one step earlier for the Forward Euler method in order to match it with the Backward Euler expression. Further on we see the we get the Backward Euler- Implicit scheme if θ is chosen to be 1. This is all fine, as we are able to reproduce the different schemes with a specific choice of θ. The last choice of θ that we will present is however the most interesting one, $\theta = 1/2$. For this choice of θ we end up with the Crank-Nicolson scheme. For the previous schemes we saw that the truncation error of the time derivative was $O(\Delta t)$, the Crank-Nicolson Scheme on the other hand has a truncation error which goes like $O(\Delta t^2)$.

The way we proceed from here is to rearrange equation 13 when $\theta = 1/2$ is implemented.

$$\frac{2(u_{i,j} - u_{i,j-1})}{\Delta t} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + \frac{u_{i+1,j-1} - 2u_{i,j-1} + u_{i-1,j-1}}{\Delta x^2}$$

Defining $\alpha = \Delta t/\Delta x^2$ as before:

$$2(u_{i,j} - u_{i,j-1}) = \alpha[u_{i+1,j} - 2u_{i,j} + u_{i-1,j}] + \alpha[u_{i+1,j-1} - 2u_{i,j-1} + u_{i-1,j-1}]$$

Further on by rearranging the equation so that we have the "Forward Euler" on one side and "Backward Euler" on the other side we get:

$$-\alpha u_{i+1,j} + (2 + 2\alpha)u_{i,j} - \alpha u_{i-1,j} = \alpha u_{i+1,j-1} + (2 - 2\alpha)u_{i,j-1} + \alpha u_{i-1,j-1} \tag{14}$$

Where we can identify the left-hand side to be:

$$\mathbf{A}_{CN_1} = \begin{bmatrix} 2+2\alpha & -\alpha & 0 & \dots & \dots & 0 \\ -\alpha & 2+2\alpha & -\alpha & 0 & \dots & \dots \\ 0 & -\alpha & 2+2\alpha & -\alpha & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & -\alpha & 2+2\alpha & -\alpha \\ 0 & \dots & \dots & 0 & -\alpha & 2+2\alpha \end{bmatrix}$$

And the right-hand side:

$$\mathbf{A}_{CN_2} = \begin{bmatrix} 2-2\alpha & \alpha & 0 & \dots & \dots & 0 \\ \alpha & 2-2\alpha & \alpha & 0 & \dots & \dots \\ 0 & \alpha & 2-2\alpha & \alpha & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \alpha & 2-2\alpha & \alpha \\ 0 & \dots & \dots & 0 & \alpha & 2-2\alpha \end{bmatrix}$$

These matrices - $\mathbf{A}_{CN_1}$ & $\mathbf{A}_{CN_2}$ are simply tridiagonal matrices, we can then construct a matrix $\mathbf{B}$, which will be the tridiagonal matrix:

$$\mathbf{B} = \begin{bmatrix} 2 & -1 & 0 & \dots & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & \dots \\ 0 & -1 & 2 & -1 & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & -1 & 2 & -1 \\ 0 & \dots & \dots & 0 & -1 & 2 \end{bmatrix}$$

We may rewrite equation 14 in a matrix- form:

$$(2\hat{I} + \alpha\mathbf{B})\mathbf{V}_j = (2\hat{I} - \alpha\mathbf{B})\mathbf{V}_{j-1}$$

$$\therefore \quad \mathbf{V}_j = (2\hat{I} + \alpha\mathbf{B})^{-1}(2\hat{I} - \alpha\mathbf{B})\mathbf{V}_{j-1} \tag{15}$$

Where $\mathbf{V}_j$ and $\mathbf{V}_{j-1}$ is as defined previously, simply vectors containing $(u_{i,j})$ for $i \in (1, n)$.

iv. 2D implementation of diffusion equation

So far we have only talked about a 1+1 dimensional diffusion equation. We are also interested in the 2 + 1 dimensional case, this equation reads:

$$\frac{\partial u}{\partial t} = \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{16}$$

Where we have that u is dependent of x,y and t giving u(x,y,t). The methods for solving this in a discretized form is basically the same as for the previous case. The exception comes for the added dimension of-course. In our formalism we would like to index all three variables. As for the 1D case we could write u(x,t) as $u_{i,j}$, where i and j was position along x and time respectively. In 2D we rewrite this formality in order to take care of all three variables. This gives us an indexing $u^k_{i,j}$, where i and j is x and y respectively and k is the time index. We go about as previously, and defining the discretized second order derivative which is respect to x:

$$u_{xx} \approx \frac{u^k_{i+1,j} - 2u^k_{i,j} + u^k_{i-1,j}}{h^2}$$

Where h is defined as $h = L/(n+1)$, where L is the length of our square lattice. We create this square lattice defined by x and y, with equally many mesh points in each direction. Where $x = x_0 + ih$ and $y = y_0 + jh$. Now we discretized with respect to y:

$$u_{yy} \approx \frac{u^k_{i,j+1} - 2u^k_{i,j} + u^k_{i,j-1}}{h^2}$$

The time derivative may be written as such:

$$u_t \approx \frac{u^{k+1}_{i,j} - u^k_{i,j}}{\Delta t}$$

We recognise this equation by the Forward Euler method.Inserting these equation into equation 16 we have:

2D Explicit scheme:

$$u^{k+1}_{i,j} = u^k_{i,j} + \alpha[u^k_{i+1,j} + u^k_{i-1,j} + u^k_{i,j+1} + u^k_{i,j-1} - 4u^k_{i,j}]$$

We may write this equation in a more compact by defining $\Delta^k_{i,j} = [u^k_{i+1,j} + u^k_{i-1,j} + u^k_{i,j+1} + u^k_{i,j-1}]$, we then have:

$$u^{k+1}_{i,j} = u^k_{i,j} + \alpha[\Delta^k_{i,j} - 4u^k_{i,j}] \tag{17}$$

This is the final result, and this is the explicit scheme due to the choice of u_t. We have now defined $\alpha = \frac{\Delta t}{h^2}$. By knowing this it is easy to derive the implicit scheme in two dimensions. We use the Backward Euler method when solving for u_t giving us:

$$u_t \approx \frac{u^k_{i,j} - u^{k-1}_{i,j}}{\Delta t}$$

Using the same procedure as before we end up with the implicit scheme for 2D diffusion equation:

2D Implicit scheme:

$$u^k_{i,j} = \frac{1}{1+4\alpha}[\alpha(u^k_{i+1,j} + u^k_{i-1,j} + u^k_{i,j+1} + u^k_{i,j-1}) + u^{k-1}_{i,j}]$$

Where α is defined as for the 2D explicit case. As done in the explicit case, we may rewrite this into a more compact way with $\Delta^k_{i,j}$:

$$u^k_{i,j} = \frac{\alpha\Delta^k_{i,j} + u^{k-1}_{i,j}}{1+4\alpha} \tag{18}$$

v. Stability requirements

Stability requirements is a crucial part to be aware of when implementing the code. We have not talked about which stability requirements is necessary for the the three methods we have introduced. The stability condition for 1D Forward Euler is given as:

$$\frac{\Delta t}{\Delta x^2} \leq 1/2 \tag{19}$$

This stability requirement is weak as we can see. By choosing a step-size of 0.1 we need to have a time-step of at least 0.005s. We can therefore conclude that if our time interval is large, it would be difficult calculate the solution. The implicit scheme on the other hand satisfies all requirements giving it a much broader specter. The same goes for Crank-Nicolson- it also has no restriction of the ratio of time-step and step-size. In order to derive the relation of stability conditions for the various schemes we need to introduce the so-called spectral radius $\rho(\hat{A})$, where we have named $\hat{A} \rightarrow \mathbf{A}_{BE}$(Backward Euler), $\mathbf{A}_{FE}$(Forward Euler) and $\mathbf{A}_{CN_1}, \mathbf{A}_{CN_2}$ (Crank-Nicolson). The important note is that our matrices need to satisfy the condition:

$$\rho(\hat{A}) < 1 \tag{20}$$

whereas the spectral radius is defined as:

$$\rho(\hat{A}) = max\{|\lambda| : det(\hat{A} - \lambda\hat{I}) = 0\} \tag{21}$$

Where λ is the eigenvalue. By this the spectral radius is satisfied for the Backward Euler and Crank-Nicolson. For the Backward Euler the condition is satisfied because $\rho(\mathbf{A}_{BE}^{-1}) < 1$, the reverse matrix is due to the backwards iterations as mentioned previously. Table 1 gives an overview over the truncation error for the three methods introduced as well as the stability requirements. Truncation error is very much defined here, but further insight study of implementing multiple methods for treating the boundary condition to improve upon the truncation error is found here [1].

Table 1: Overview of Schemes truncation error and stability requirements

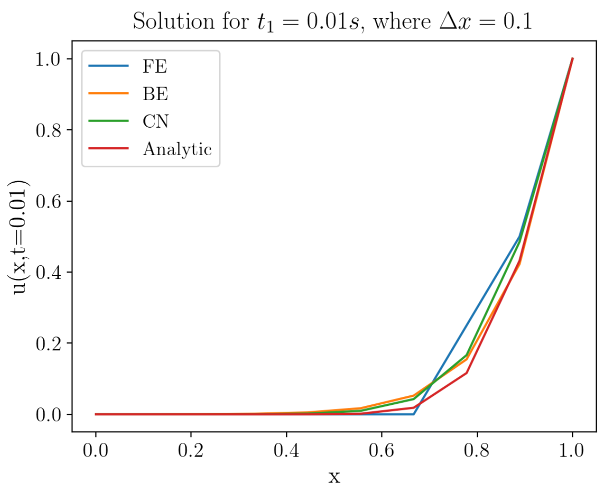
Scheme	**Truncation Error**	**Stability requirements**
Forward Euler	$O(\Delta x^2)$ & $O(\Delta t)$	$\Delta t \leq \frac{1}{2}\Delta x^2$
Backward Euler	$O(\Delta x^2)$ & $O(\Delta t)$	$\mathbb{N} \in \Delta x$ and Δt
Crank-Nicolson	$O(\Delta x^2)$ & $O(\Delta t^2)$	$\mathbb{N} \in \Delta x$ and Δt

III. Implementation

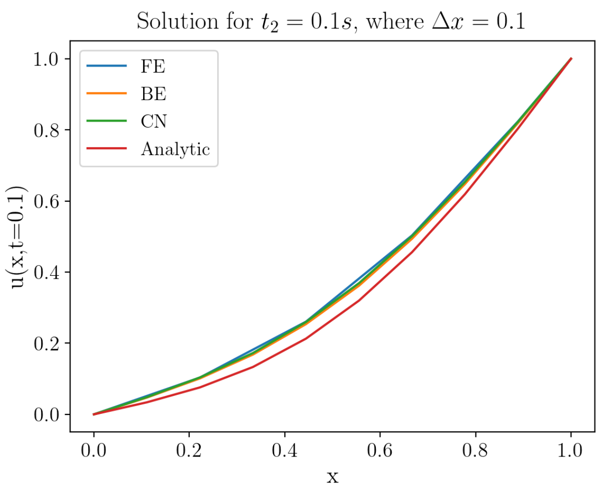
Programs used in this project can be found on https://github.com/patrykpk/FYS4150/tree/master/Project_5 and the "README.md" explains how to run the scripts. All calculations are done in C++, while the plotting is done in Python. Our C++ program is based on the codes found in the course's github repository https://github.com/CompPhysics/ComputationalPhysics/tree/master/doc/Projects/2019/Project5 and the content found in "Computational Physics Lecture Notes 2015" on Partial Differential Equations [2]. Arrays and matrices are generated using the Armadillo library [3] [4], to simplify some aspects that would otherwise require working with pointers. This section will include how we implemented the code and include some of the code:

IV. Numerical results

i. Diffusion Equation in 1D

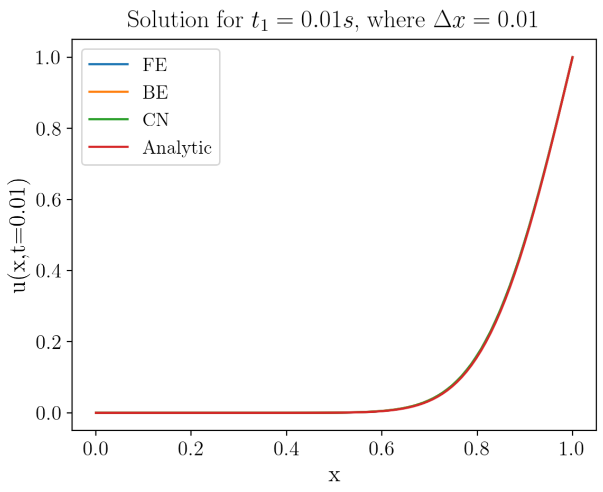


a. Solution with $\Delta x = 0.1$, and time = 0.015s

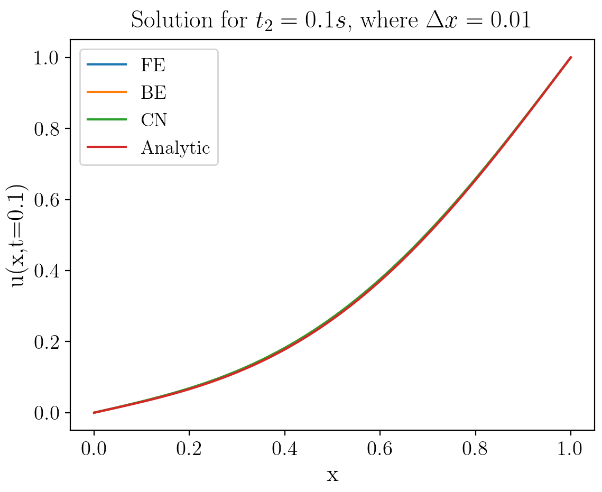


b. Solution with $\Delta x = 0.1$, and time = 0.15s

Figure 1: Solution of Diffusion equation for the three different methods as described under the section [Method]. Analytic solution is also depicted as red, where the analytic solution is derived in Appendix A. These two plots shows how the solution goes as time is increased. Figure a) is for time = 0.01s and figure b) is for time 0.1s. The step-length is set to $\Delta x = 0.1$

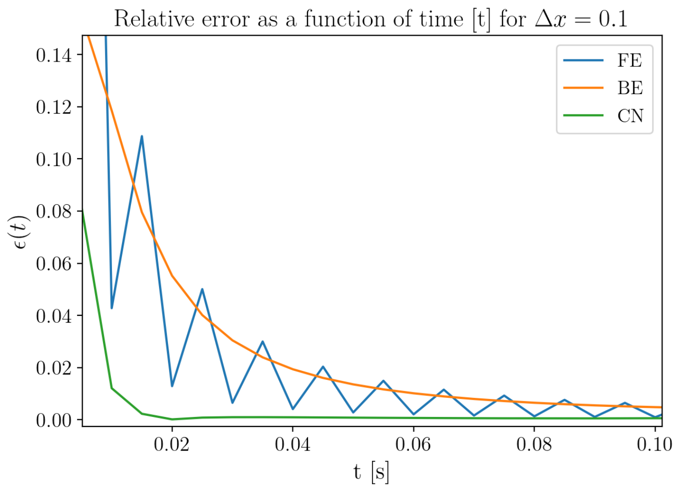


a. Solution with $\Delta x = 0.01$, and time = 0.01s

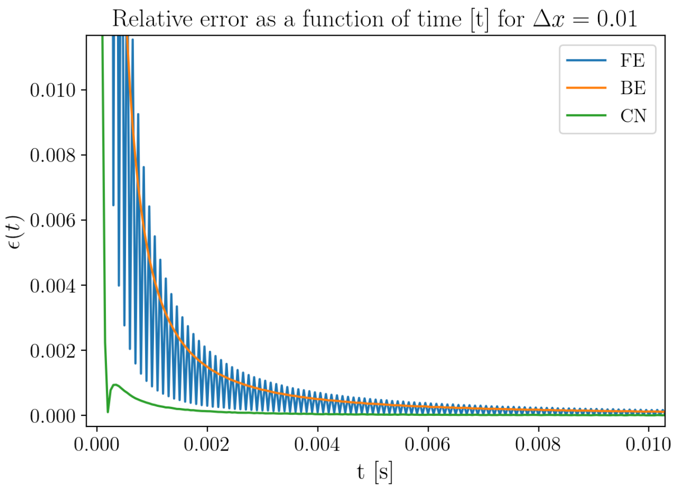


b. Solution with $\Delta x = 0.01$, and time = 0.1s

Figure 2: Solution of Diffusion equation for the three different methods as described under the section [Method]. Analytic solution is also depicted as red, where the analytic solution is derived in Appendix A. These two plots shows how the solution goes as time is increased. Figure a) is for time = 0.01s and figure b) is for time 0.1s. The step-length is set to $\Delta x = 0.01$



a. Relative error as a function of time for $\Delta x = 0.1$.



b. Relative error as a function of time for $\Delta x = 0.01$.

Figure 3: Relative error plotted as a function of time. Where figure a) is for $\Delta x = 0.1$ and figure b) is for $\Delta x = 0.01$. The plot is based on equation 22 for the various schemes introduced.

ii. Diffusion Equation in 2D

Results from solving the two-dimensional equation numerically will be presented in this section. Comparing explicit and implicit scheme towards the analytical expression with initial conditions as solved in Appendix B in equation 26.

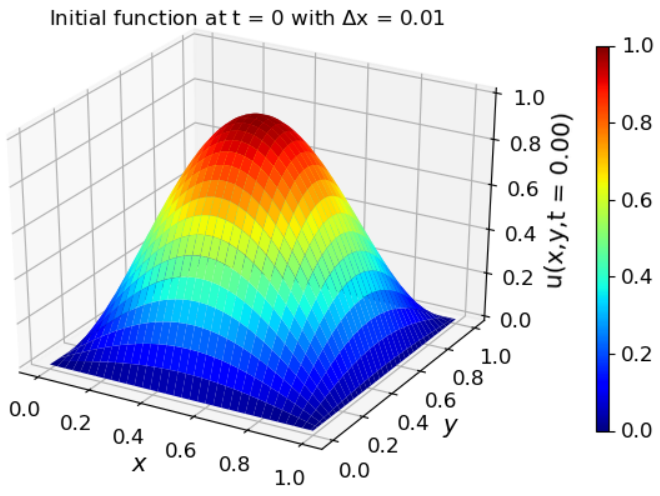
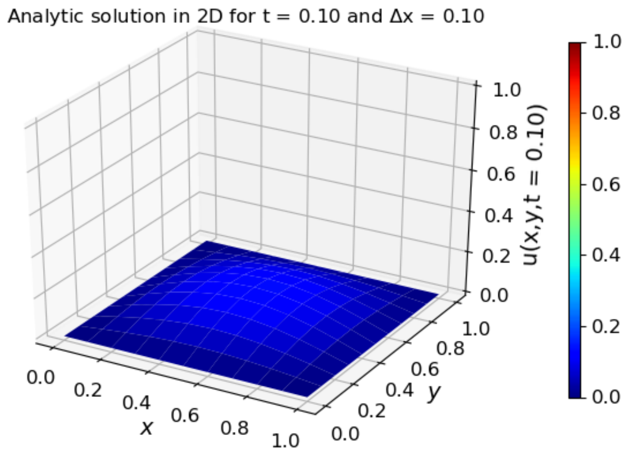
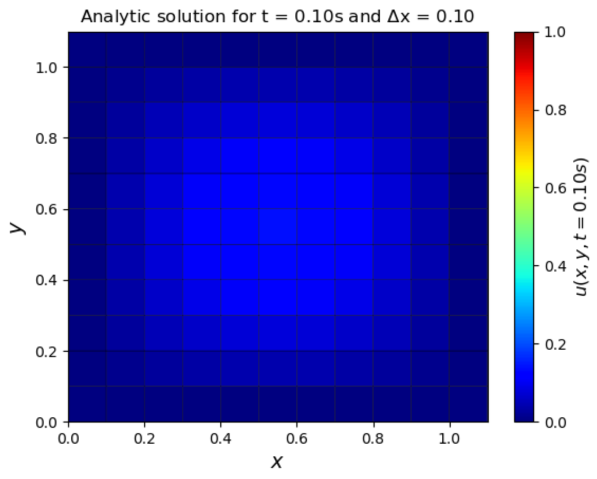


Figure 4: Visualization of the initial function at t $= 0$ with $\Delta x = 0.01$. Showing a peak at the middle point and decreasing towards the edges.



(a)



(b)

Figure 5: Analytic solution for the diffusion after $t = 0.10s$ shown in 3D and as a heat map. Elements of the function $u(x, y, t)$ falling rapidly towards equilibrium. Figure a) visualizes the values in 3D and b) visualizes the same values in 2D as a heat map.

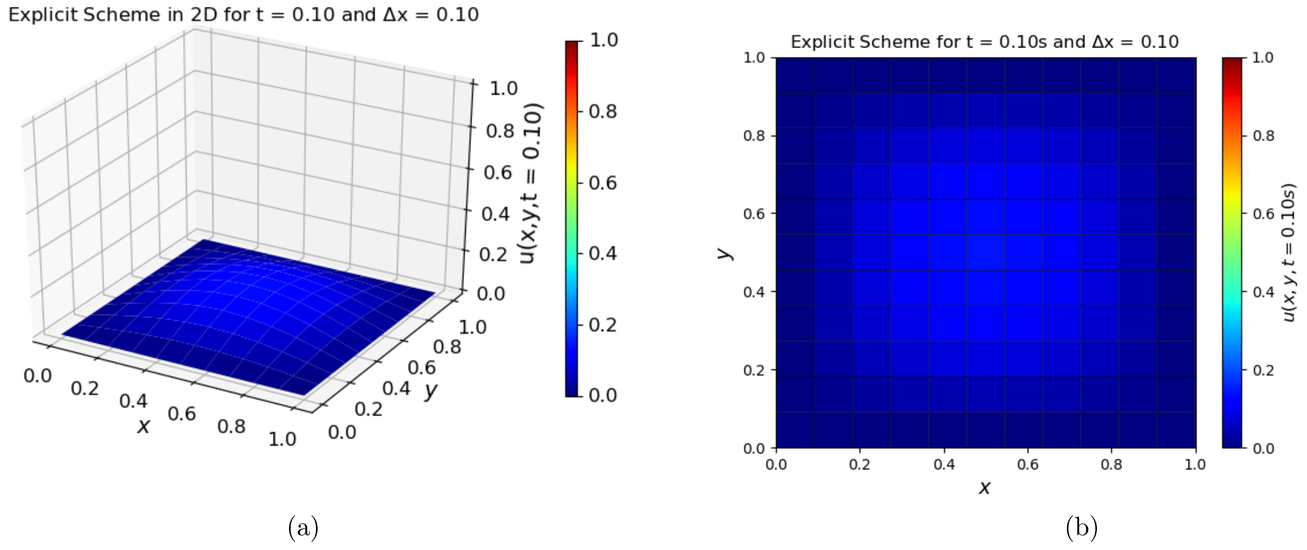


Figure 6: Explicit scheme for the diffusion after $t = 0.10s$ with $\Delta x = 0.10$ and $\Delta t = 2.5 \times 10^{-3}s$ given from the stability condition of $\Delta t \leq \frac{1}{4}\Delta x^2$. Figure a) visualizes the values in 3D and b) visualizes the same values in 2D as a heat map.

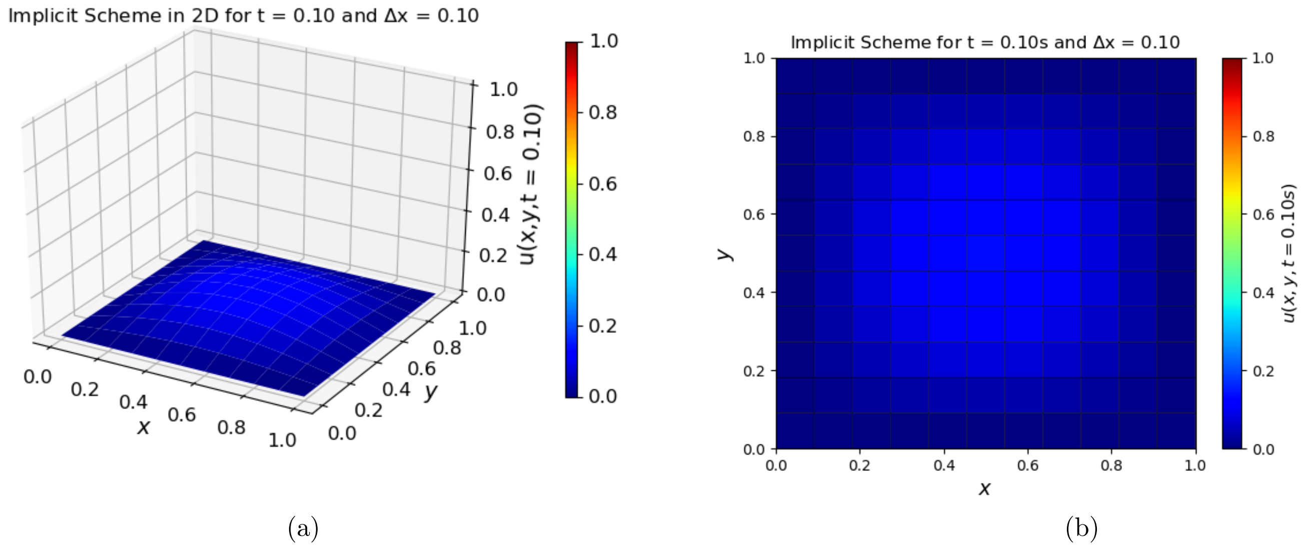
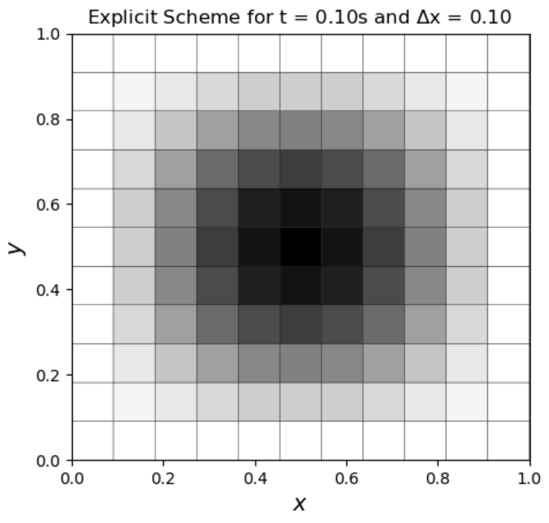
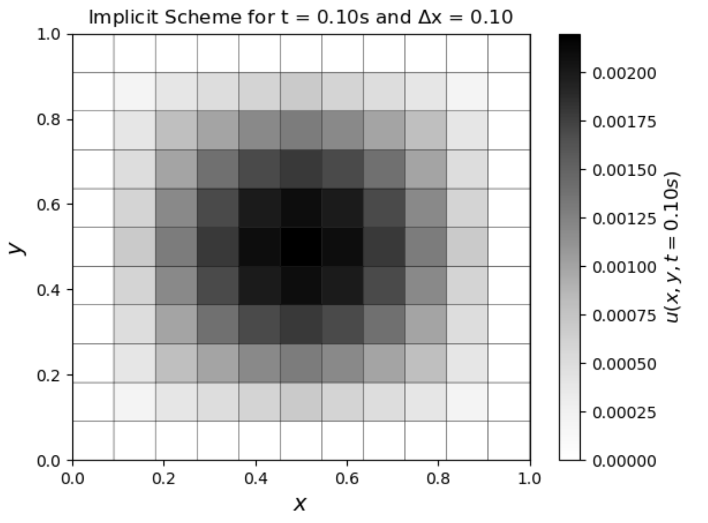


Figure 7: Implicit scheme for the diffusion after $t = 0.10s$ with $\Delta x = 0.10$ and $\Delta t = 2.5 \times 10^{-3}s$ given from the stability condition of $\Delta t \leq \frac{1}{4}\Delta x^2$ for the Explicit scheme. This was used in order to compare the different schemes. Figure a) visualizes the values in 3D and b) visualizes the same values in 2D as a heat map

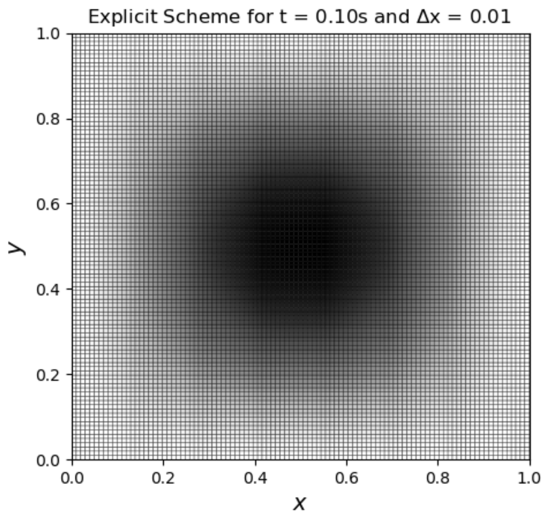


(a)

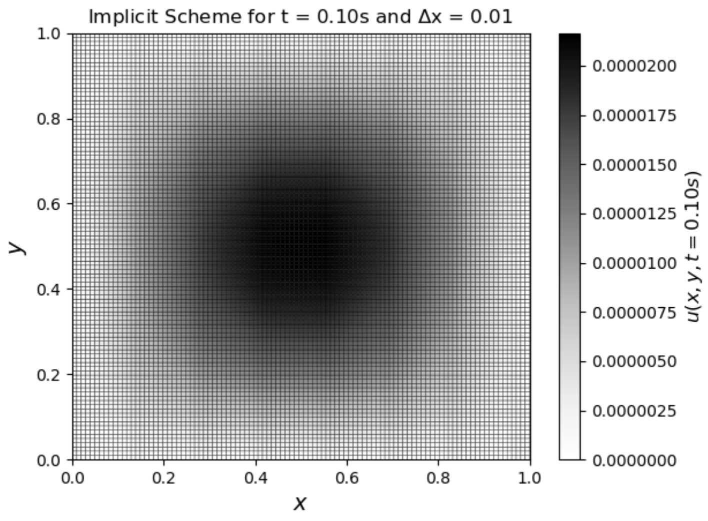


(b)

Figure 8: Absolute error between different schemes and analytic solution at $t = 0.10s$ with $\Delta x = 0.10$ when stability condition for Explicit scheme is fulfilled giving $\Delta t = 2.5 \times 10^{-3}s$. Figure a) shows the Explicit scheme and figure b) the Implicit scheme.



(a)



(b)

Figure 9: Absolute error between different schemes and analytic solution at $t = 0.10s$ with $\Delta x = 0.01$ when stability condition for Explicit scheme is fulfilled giving $\Delta t = 2.5 \times 10^{-5}s$. Figure a) shows the Explicit scheme and figure b) the Implicit scheme.

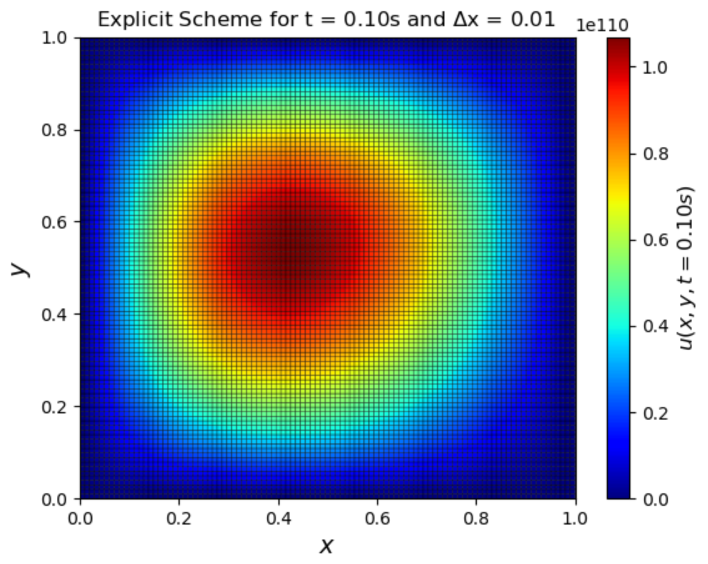


Figure 10: Explicit scheme for the diffusion after $t = 0.10s$ with $\Delta x = 0.01$ and $\Delta t = 2.6 \times 10^{-5}s$. Numerical solution when the stability condition of $\Delta t \leq \frac{1}{4}\Delta x^2$ is not fulfilled with this value of Δt. Function values of $u(x, y, t = 0.10s)$ diverge resulting in values in the magnitude of 10^{110}.

V. Discussion

i. Diffusion in 1D

When solving the diffusion equation for 1D, we implemented the code according the numerical solution derived under section *method*. All three methods where implemented and under the section *Results/Diffusion Equation in 1D* the solutions is plotted under two different step-lengths, $\Delta x = [0.01, 0.1]$. Figure 1, shows solution of u(x,t=0.1) and u(x,t=0.01), when the step-length is chosen to be $\Delta x = 0.1$. As we know the criterion for Forward Euler is according to table 1: $\Delta t \leq \frac{1}{2}\Delta x^2$, whereas, if $\Delta x = 0.1$, the time-step needs to be 0.005s. This relation is implemented for all methods, although one do not necessarily need to have this relation for Backward Euler and Crank-Nicolson, as discussed under section *Method/Stability requirements*. In figure 1a) one sees that the solution has not reached steady state $\rightarrow$ u(x,t $= \infty$) = ax + b. This linearity shows that the solution is in steady state. By figure 1b) on the other-hand the time is increased and one can see that the solution approaches the linearity in which we have steady state. All these methods is compared with the analytic solution, which is derived under Appendix A. The sum in the analytical solution were chosen to be truncated at 100 summations, as the solution quickly converged and ∞ must be approximated on a computer anyways.

Figure 2 shows the solution to the differential equation for the various methods introduced for $\Delta x = 0.01$. This is also under the stability condition required by Forward Euler, giving a time-step of $\Delta t = 5 \times 10^{-5}s$, in these two plots we have a hard time distinguishing the different methods. Whereas for figure 1, the distinction was clear. At the same time we see that the methods overlaps neatly with the analytic solution depicted in red.

Moreover, we may analyse our results by relative error $\epsilon(t)$, The relative error for the to figures [1][2] is shown in figure 3. The relative error for our system is written as:

$$\epsilon(t_j) = \frac{\sqrt{\sum\limits_{i=0}^{N-1} \left[FDS(x_i,t_j) - f(x_i,t_j)\right]^2}}{\sqrt{\sum\limits_{i=0}^{N-1} f(x_i,t_j)^2}} \tag{22}$$

Where FDS is the finite discrete schemes in which we analyse. The summation goes over all possible positions x_i. $f(x,t)$ is the analytic solution of 1D diffusion equation. By creating a double for-loop we run over all times in the outer for-loop and run over all the positions at that given time. Once we are in the inner for-loop we make the summation of of the nominator and denominator separately in order to avoid division by zero. Collecting the sum and defining the relative error. Figure 3a) shows the relative error as a function of time for $\Delta x = 0.1$. This figure shows that the relative error is smallest for Crank-Nicolson (green line). This is however a very satisfying result, since the truncation error goes as Δx^2 & Δt^2 for Crank-Nicolson, and for the Forward and Backward Euler methods it goes as Δx^2 & Δt as shown in table 1. By this conviction the Crank-Nicolson scheme should show the smallest relative error, which it does. If we take a look at figure 3b) we see that the relative error for all methods reaches a smaller relative error than for the $\Delta x = 0.1$ situation. With smaller step-length we are able to better approximate the analytical value, as we could see by figure 2 .

According to our knowledge about the truncation errors and stability for those different schemes we would assume that the behaviour for Forward-Euler looks rather strange as the error oscillates in Figure 3. After some investigation as what could be the cause for this strange behaviour we came across one possible reason after examining our results. This came up during development of the unit test as the results did not really match. During indexation and constructing of the matrix there was one index missing meaning that perhaps the values did not span the space as expected and thus not satisfying the stability condition resulting in the behaviour as seen in Figure 3. Further improvement for this would be to investigate this case and fixing it and determining if this was the case.

ii. Diffusion in 2D

Initial condition with the corresponding analytical expression as seen in Appendix B in equation 26 is shown in Figure 4. Initially the function has highest peak in the middle with gradually decreasing function values towards the edges. Figure 5 shows the function at a later stage when $t = 0.10s$ with a smaller $\Delta x = 0.10$. Behaviour of the system is rapid as after only $t = 0.10s$ the function values have been reduced significantly.

Numerically solving the two-dimensional equation using explicit and implicit scheme for the same time of $t = 0.10s$ with $\Delta x = 0.10$ and $\Delta t = 2.5 \times 10^{-3}s$ was performed and can be seen in Figure 6 for the explicit scheme and Figure 7 for the implicit scheme. In comparison to the analytical solution from equation 26 in appendix B, the shape and color bar indicating the function value both for the implicit and explicit resemble the analytical one. Meaning that the behaviour is as expected and that both the schemes reproduce the same features for different time values. Due to to the similarity in the results for both schemes and being unable to distinguish which method that gives better approximation, the absolute error was determined for the same time.

Absolute error for the explicit and implicit scheme from the analytical values are seen in Figure 8 for $\Delta x = 0.1$. According to the values in this figure, implicit scheme seems to give much better description of the system. For both cases the absolute error lies in the in the order of 10^{-3}, where the error is at its highest at the midpoint. This seems reasonable as the analytical function approaches zero when enough time has passed, where the the values towards the edges will change less resulting in the highest deviation being at the midpoint.

Acquiring even better numerical approximation can be seen in Figure 9 for both the explicit and implicit scheme. This figure shows the same time value of $t = 0.10s$, but the Δx is decreased to $\Delta x = 0.01$. This change in Δx requires different Δt that fulfills the stability condition of $\Delta t \leq \frac{1}{4}\Delta x^2$, which in this case was $\Delta t = 2.5 \times 10^{-5}s$. Due to both lower Δx and Δt the absolute error is even lower in the order of 10^{-5}. From both figure 8 and 9, a reasonable approximation can be made, that the absolute error for the implicit method is about half the value as for the explicit method.

In order to establish that the explicit scheme is unstable when the condition is unsatisfied the value of Δt was chosen slightly above the criteria in Figure 10. Also this simulation was performed with $\Delta x = 0.01$ and around the same time $t \approx 0.10s$. Right away the figure might look correct, but further investigation show that the values are no longer symmetric as they are slightly off-centered and the color bar also indicates that the values are in the order of 10^{110} which are not representable. From this observation we can confirm that the stability criteria for the explicit scheme holds and that values of Δx and Δt that do not satisfy $\Delta t \leq \frac{1}{4}\Delta x^2$ will diverge.

VI. Conclusion

By numerically analysing three different methods for solving the 1D diffusion equation, we could determine which method was optimal or superior for that matter. We compared these methods for two different step-lengths which gave interesting results. These results gave us a clear indication that large step-size yielded a higher relative error, and this would be approved upon when step-size was decreased to $\Delta x = 0.01$ - We could then see that it was difficult to distinguish between the different schemes and the analytic result, resulting in small relative error. It was discovered that the Crank-Nicolson scheme was the method in which gave the least relative error, however some minor flaws may have given us a condition which is not satisfied for Forward Euler. Regardless, the truncation error of Crank-Nicolson scheme compared with the other gave us a clear indication of which method that would be best - the results confirmed this.

Results from simulations when solving the 2D diffusion equation indicated that the implicit scheme obtained better approximation to the analytical values regardless of Δx used. In addition when the explicit scheme was used the choice for Δt was restricted due to the stability condition. No such condition was found for the implicit scheme. As the implicit scheme seems to outperform the explicit scheme in both 1D and 2D and is not restricted by a stability condition it is the suggested alternative when solving such problems.

References

[1] Varun Prakash Puneria. *Truncation error analysis for diffusion schemes in boundary regions of unstructured meshes.* PhD thesis, University of British Columbia, 2015.

[2] M. Hjorth-Jensen. Computational Physics. *University of Oslo*, `https://github.com/CompPhysics/ComputationalPhysics`, 2013.

[3] Conrad Sanderson and Ryan Curtin. Armadillo: a template-based C++ library for linear algebra. *Journal of Open Source Software*, 1(3):26, 2016.

[4] Conrad Sanderson and Ryan Curtin. Practical Sparse Matrices in C++ with Hybrid Storage and Template-Based Expression Optimisation. *Mathematical and Computational Applications*, 24(3):70, 2019.

Appendices

A. Analytical solution to the one dimensional heat equation.

In one dimension, the heat equation have the following form:

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial u(x,t)}{\partial t}, \quad t > 0, \quad x \in [0, L]$$

We have a Dirichlet boundary problem, where one end is fixed at zero and the other at one at all times, $u(0,t) = 0 \quad \& \quad u(L,t) = 1$ for $t \geq 1$. A physical interpretation of this problem can be a rod fixed at both ends to two different heat sources that keeps the temperature at each side fixed. Heat will then flow from the hot end to the cold end until steady-state temperature distribution is achieved. The initial condition $\Big(u(x,0) = 0 \quad 0 < x < L\Big)$ states that the rod is at zero for every position except for the end points which are defined from the boundary condition. For all $t \geq t_{ss}$, where t_{ss} is the time where steady-state is achieved in the system, the solution is expected to be a linear function $u_0(x) = ax + b$, which satisfies the 1D-Laplace equation $\big(\frac{d^2 u_0(x)}{dx^2} = 0\big)$. From the boundary condition at $x = 0$, the constant $b = 0$ and $a = 1/L$. An ansatz on the solution to be a product of a space dependent part and a time dependent part in addition to the linear term will be used to try to solve the partial differential problem at hand $\big(u(x,t) = \frac{x}{L} + F(x)G(t)\big)$. The notation used in these calculations will be $(')$ for space derivatives and $(\cdot)$ over the function for time derivatives.

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial u(x,t)}{\partial t}$$

$$G(t)F''(x) = F(x)\dot{G}(t)$$

$$\frac{1}{F(x)}F''(x) = \frac{1}{G(t)}\dot{G}(t) = -k^2$$

The last equation states that the spacial part on the left hand side (L.H.S) and the time part on the right hand side (R.H.S) must be equal to some constant. This constant is chosen to be squared for reasons shown later in the calculations, while the minus sign is there to make it so that the solutions will not "blow up" when time goes on. First we will solve the time-dependent part:

$$\frac{1}{G(t)}\dot{G}(t) = -k^2$$

$$\frac{1}{G(t)} \cdot \frac{dG(t)}{dt} = -k^2$$

$$\int \frac{dG(t)}{G(t)} = \int -k^2 dt$$

$$\ln G(t) = -k^2 t$$

$$G(t) = e^{-k^2 t}$$

No constant is included in the above expression, since all constants will be baked into the spacial part constant. From the solution, the choice of negative constant now makes sense, as the time dependent part will fall exponentially to zero for large values of t. Next we solve for the spacial part, this is a well known equation called the Helmholtz equation $\big(d^2F(x)/dx^2 + k^2F(x) = 0\big)$. This equation generalizes to more dimensions as $\nabla^2 F(x_1, x_2, ..., x_n) + k^2 F(x_1, x_2, ..., x_n) = 0$, which must be further separated, which will be done in the other section of the appendix, the solution to the 2D problem. Helmholtz equation will be solved using the complex conjugate roots of the auxiliary equation.

$$\frac{d^2F(x)}{dx^2} + k^2F(x) = 0$$

$$\lambda^2 + k^2 = 0 \implies \lambda_\pm = \pm ik$$

$$\implies F(x) = A\sin{(kx)} + B\cos{(kx)}$$

The $\cos{(kx)}$ term from the solution of Helmholtz, will be disregarded because of the boundary condition stating that $u(0,t) = 0$. The other boundary condition for $x = L$ can be achieved when $\sin{(kL)} = 0$ which gives us the quantization $k = \frac{n\pi}{L}$.

$$u_n(x,t) = A_n \sin{(\frac{n\pi}{L}x)} e^{-\frac{n^2\pi^2}{L^2}t} \tag{23}$$

Equation 23 gives us a basis of eigenfunctions that can be spanned to give the whole solution.

$$u(x,t) = u_0(x) + \sum_{n=1}^{\infty} u_n(x,t) = \frac{x}{L} + \sum_{n=1}^{\infty} A_n \sin{(\frac{n\pi}{L}x)} e^{-\frac{n^2\pi^2}{L^2}t}$$

The next step is to find the coefficients A_n, by finding the Fourier sine series that can describe the initial temperature distribution in the rod. This function can be modeled with a Heaviside step function, $H(x-L)$, i.e. 0 for $x < L$ and 1 for $x \geq L$, but for this problem we have $x \in [0, L]$, so we are not interested in points beyond L.

$$u(x,0) = H(x-L) = \frac{x}{L} + \sum_{n=1}^{\infty} A_n \sin{(\frac{n\pi}{L}x)}$$

$$H(x-L) - \frac{x}{L} = \sum_{n=1}^{\infty} A_n \sin{(\frac{n\pi}{L}x)}$$

$$\implies A_n = \frac{2}{L}\int_0^L \big(H(x-L) - \frac{x}{L}\big)\sin{(\frac{n\pi}{L}x)}dx$$

$$= H(x-L)\bigg(\frac{2\cos{(n\pi)}}{n\pi} - \frac{2\cos{(\frac{n\pi}{L}x)}}{n\pi}\bigg) - \frac{2\sin{(\frac{n\pi}{L}x)}}{n^2\pi^2} + \frac{2x\cos{(\frac{n\pi}{L}x)}}{n\pi L}\Bigg|_0^L$$

$$A_n = \frac{2\cdot(-1)^n}{n\pi}$$

The solution for this integral was found through the use of WolframAlpha.[1] The integral could have been split up into two integrals, one with the Heaviside, and one with the linear term. Solving this would have given 0 for the first integral, meaning the only contribution comes from the linear term integral. This means that an initial approximation of $u(x,0) = 0$ could have been made and would end with the same result. Inserting the limits 0 and L in the equation terminates all terms except the last cosine term for $x = L$. This cosine term will change between (-1) and 1 for odd and even n respectively, so that $A_n = \frac{2\cdot(-1)^n}{n\pi}$. From this we now have an analytical solution to the 1D heat equation for the given boundary conditions.

$$u(x,t) = \frac{x}{L} + \sum_{n=1}^{\infty} 2\frac{(-1)^n}{n\pi}\sin\left(\frac{n\pi}{L}x\right)e^{-\frac{n^2\pi^2}{L^2}t} \tag{24}$$

B. Analytical solution to the two dimensional heat equation

In two dimensions the heat equation looks fairly equal to the one dimensional. Using the nabla operator for Cartesian coordinates in two dimensions give the following partial derivative equation.

$$\Big(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\Big)u(x,y,t) = \nabla^2 u(x,y,t) = \frac{\partial u(x,y,t)}{\partial t} \tag{25}$$

The boundaries will be fixed to zero, giving the boundary conditions $u(0,y,t) = u(L,y,t) = u(x,0,t) = u(x,L,t) = 0$. Initial condition for the surface at $t = 0$ is given by the function $f(x,y) = \sin\left(\frac{\pi}{L}x\right)\sin\left(\frac{\pi}{L}y\right)$. By variable separation, such as in the 1 dimensional case, an analytical solution can be made for the given conditions.
Ansatz:

$$u(x,y,t) = X(x)Y(y)T(t)$$

$$X(x)Y(y)\frac{\partial T(t)}{\partial t} = Y(y)T(t)\frac{\partial^2 X(x)}{\partial x^2} + X(x)T(t)\frac{\partial^2 Y(y)}{\partial y^2}$$

$$\frac{1}{T(t)}\frac{\partial T(t)}{\partial t} = \frac{1}{X(x)}\frac{\partial^2 X(x)}{\partial x^2} + \frac{1}{Y(y)}\frac{\partial^2 Y(y)}{\partial y^2} = -k^2$$

So far this looks like the calculation for the one dimensional case, and the temporal part actually has the same solution: $T(t) = e^{-k^2 t}$. Next is to calculate the space parts.

$$-k^2 = \frac{1}{X(x)}\frac{\partial^2 X(x)}{\partial x^2} + \frac{1}{Y(y)}\frac{\partial^2 Y(y)}{\partial y^2}$$

$$-\left(\frac{1}{Y(y)}\frac{\partial^2 Y(y)}{\partial y^2} + k^2\right) = \frac{1}{X(x)}\frac{\partial^2 X(x)}{\partial x^2} = -p^2$$

$X(x)$:

$$\nabla^2 X(x) + p^2 X(x) = 0$$

$$\implies X(x) = \sin(px) + \cos(px)$$

[1]`https://bit.ly/2PjawmT` – Wolfram Alpha link for solution.

$Y(y)$:

$$\nabla^2 Y(y) + Y(y)(k^2 - p^2) = \nabla^2 Y(y) + Y(y)q^2 = 0$$

$$\implies Y(y) = \sin{(qy)} + \cos{(qy)}$$

Both the x-term and y-term have the form of Helmholtz equation, solved as explained for one dimension. Due to the boundary conditions, the cosine terms from both must be disregarded and the constants p and q will be quantified as $\frac{n\pi}{L}$ and $\frac{m\pi}{L}$ respectively. This implies the constant $k^2 = (n^2+m^2)\pi^2/L^2$, and we can write a general solution as a superposition as in the one dimensional case.

$$u(x,y,t) = \sum_{n=1}^{\infty}\sum_{m=1}^{\infty} A_{nm} \sin{(\frac{n\pi}{L}x)} \sin{(\frac{m\pi}{L}y)} e^{-(n+m)^2\frac{\pi^2}{L^2}t}$$

$$A_{nm} = \left(\frac{2}{L}\right)^2 \int_0^L \int_0^L \sin{(\frac{\pi}{L}x)} \sin{(\frac{\pi}{L}y)} \sin{(\frac{n\pi}{L}x)} \sin{(\frac{m\pi}{L}y)} dxdy = 1$$

When calculating the coefficient A_{nm}, the orthogonality properties of sine functions are used: $\int_0^L \sin{(\frac{n\pi}{L}x)} \sin{(\frac{m\pi}{L}x)} dx = \frac{L}{2}\delta_{nm}$. This property made sure that the only contribution to the sum comes when $n = m = 1$ giving the analytical solution as follows:

$$u(x,y,t) = \sin{(\frac{\pi x}{L})} \sin{(\frac{\pi y}{L})} e^{-\frac{2\pi^2}{L^2}t} \tag{26}$$