Computational Physics

Numerical Solutions to Second-Order Differential Equations

With Geophysical Applications

Lars Willas Dreyer And *Gabriel Sigurd Cabrera*

FYS3150

December 14, 2018

Abstract

In this report, we analyzed the well-known 1-D *diffusion equation* analytically, and proceed to develop a Forward Euler, Backward Euler, and Crank-Nicolson numerical solution using the analytical solution as a benchmark for accuracy. We found that the Crank-Nicolson and Backward Euler algorithms match the analytical solution best, while also maintaining a better stability.

We then proceed to develop a 2-D model for the diffusion equation using the Forward-Euler algorithm; this was then applied to a real world scenario, where the diffused quantity was *heat*. Specifically, we modeled the evolution of the Fennoscandian lithosphere over the course of one billion years, with the assumption that it once was home to a subduction zone. With the model divided into the upper crust, lower crust, and mantle, each layer would produce heat at a different base rate, while the mantle would have an additional heat production from radioactive decay of Uranium, Thorium, and Potassium.

Overall, our results made sense with respect to our expectations – namely, that the lithosphere would get hotter and hotter over time, and spread its heat over to the crust. Whether or not this reflects reality is an unanswered question, as we would need more data to determine whether or not the current temperature of the mantle is as our model predicts.

Introduction

In this report, we will use the Forward Euler, Backward Euler, and Crank-Nicolson algorithms to numerically approximate the 1-D and 2-D *diffusion equation*. We will investigate the accuracy of these algorithms as compared to their respective analytical results, and investigate this accuracy as a function of time-step (and of position-step in the 2-D model.)

Once this is accomplished, we will augment the model with an additional heat production term and a set of physical coefficients so as to be left with a numerical model for the *heat equation*; the purpose of this change is to simulate the geophysical processes of the Fennoscandian crust and its underlying mantle.

Our goal in developing this model will be to simulate the changes in temperature in the lithosphere's layers over the course of one-billion years. We will take the lithospheric layers' separate heat productions into account, as well as the heat production through the *radioactive decay* of Uranium, Thorium, and Potassium. Using this model, we will investigate the claim that there once existed an active subduction zone on the western coast of Norway.

On a final note, all simulations will be created using `julia` – a young programming language developed for scientists and mathematicians. The files used to complete this report are available on the University of Oslo's Github.

`https://github.uio.no/larswd/FYS3150/tree/master/project5/`

Theory

Solving the Diffusion Equation

The *diffusion equation* is a partial differential equation that represents the flow of a quantity throughout a medium based on a given set of initial conditions. It can be used to represent

the flow of heat or the diffusion of particles in a medium among other things, and is of vital importance in the field of physics. It is of the form:

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

To find the N-Dimensional *general solution* for $u(\mathbf{x}, t)$, we can begin by assuming it is separable, such that:

$$u(\mathbf{x}, t) = A_1(x_1)A_2(x_2)\cdots A_N(x_N)B(t)$$

Let us consider the 1-D case, where we set $\mathbf{x} = x$:

$$u(x, t) = A(x)B(t)$$

This means that we can rewrite the 1-D version of (1) and extract the terms' constants with respect to their derivatives:

$$\nabla^2 \left(A(x)B(t)\right) = \frac{\partial}{\partial t}\left(A(x)B(t)\right) \iff B(t)\nabla^2 A(x) = A(x)\frac{\partial}{\partial t}B(t)$$

We can then reorganize the above such that all x-dependent functions are on the left, while all t-dependent functions are on the right, and set them equal to a constant a:

$$\frac{1}{A(x)}\nabla^2 A(x) = \frac{1}{B(t)}\frac{\partial}{\partial t}B(t) = -a^2$$

We can then solve each differential separately, starting with the left-hand side:

$$\nabla^2 A(x) = -a^2 A(x) \implies A(x) = c_1 e^{iax} + c_2 e^{-iax} = c_1 \sin(ax) + c_2 \cos(ax)$$

The right-hand side is even simpler:

$$\frac{\partial}{\partial t}B(t) = aB(t) \implies B(t) = c_3 e^{-a^2 t}$$

Combining these gives us a general solution:

$$A(x)B(t) = c_1 c_3 \sin(ax)e^{-a^2 t} + c_2 c_3 \cos(ax)e^{-a^2 t}$$

We can then redefine our integration constants, giving us:

$$u(x, t) = \kappa_1 \sin(ax)e^{-a^2 t} + \kappa_2 \cos(ax)e^{-a^2 t}$$

To implement a set of boundary conditions, we will redefine our equation and let it be a function $v(x, t)$ where:

$$v(x, t) = \kappa_1 \sin(ax)e^{-a^2 t} + \kappa_2 \cos(ax)e^{-a^2 t} = u(x, t) - x$$

We will be simulating the heating of a rectangular array from the bottom, and as such wish to implement the boundary conditions $v(0,t) = v(1,t) = 0$ and $v(x,0) = -x$. We begin by setting up our first condition:

$$v(0,t) = \kappa_2 e^{-a^2 t} = 0 \implies \kappa_2 = 0$$

So our cosine term disappears. We can then set up our second relation:

$$v(1,t) = \kappa_1 \sin(a) e^{-a^2 t} = 0$$

Since the above must hold for all t, the sine term must always be zero – as such, this implies that $a = n\pi$ where $n \in \mathbb{Z}$. This leaves us with our last relation:

$$v(x,0) = \kappa_1 \sin(n\pi x) e^{-(n\pi)^2 t} = -x$$

Since this must hold for all n, we can create a superposition of possible coefficients κ_1. This gives us:

$$-x = \sum_{n=1}^{\infty} \kappa_n \sin(n\pi x)$$

We find that the above is the representation of a *Fourier series*; referring to an integration table [1] gives us that:

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

We can then flip the sign of x by increasing the exponent on the right-hand side by one. Finally, we will be simulating over a distance L, and as such we can set $n = L$, which gives us our solution:

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

If this is to be implemented programmatically, an upper bound must be chosen in lieu of ∞.

The Heat Equation

Using the aforementioned principles of the diffusion equation, we may apply the same concepts to a special case: *the heat equation*:

$$k\nabla^2 T(x,t) + Q = \rho c_p \frac{\partial}{\partial t} T(x,t) \tag{3}$$

Where given a medium, k is its thermal conductivity, Q is its heat production, ρ is its density, and c_p is its specific heat capacity.

Explicit Forward Euler in 1-D

Diffusive problems can, assuming isotropic diffusion, be modelled by the following partial differential equation:

$$\nabla^2 u(\mathbf{x}, t) = \frac{\partial u(\mathbf{x}, t)}{\partial t} \tag{4}$$

We are initially going to be regarding the one-dimensional case, in which case equation 4 can be rewritten to

$$\frac{\partial^2 u(x, t)}{\partial x^2} = \frac{\partial u(x, t)}{\partial t} \tag{5}$$

To solve equation 5 numerically, one method we can use is the explicit Forward Euler method, where we approximate said equation to:

$$\frac{u(x, t + \Delta t) - u(x, t)}{\Delta t} = \frac{u(x + \Delta x, t) - 2u(x, t) + u(x - \Delta x, t)}{\Delta x^2} \tag{6}$$

Which can be easily discretised into

$$\frac{u_{i,j+1} - u_{i,j}}{\Delta t_j} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x_i^2} \tag{7}$$

By then assuming that Δt and Δx remain constant throughout the integration, we can then easily find every position at the next time value by the following equation

$$u_{i,j+1} = \alpha u_{i+1,j} + (1 - 2\alpha) u_{i,j} + \alpha u_{i-1,j} \tag{8}$$

where $\alpha = \frac{\Delta t}{\Delta x^2}$. If we now were to define a vector U_j to be the function value of $u(x_i, t_j)$ for all x_i at a given point in time j, then we see that the equation 8 above can be rewritten to the following matrix equation:

$$\mathbf{U}_j = \hat{A}\mathbf{U}_{j-1} = \begin{bmatrix} 1-2\alpha & \alpha & 0 & \dots & 0 & 0 \\ \alpha & 1-2\alpha & \alpha & \dots & 0 & 0 \\ 0 & \alpha & 1-2\alpha & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1-2\alpha & \alpha \\ 0 & 0 & 0 & \dots & \alpha & 1-2\alpha \end{bmatrix} \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ u_{3,j} \\ \vdots \\ u_{N-1,j} \\ u_{N,j} \end{bmatrix} \tag{9}$$

In our implementation we decided upon not implementing this algorithm as a matrix equation, though this relation does prove useful, as shown in the derivatin of the Crank-Nicolson scheme. For the forward Euler algorithm, we instead implement an algorithm based upon equation 8 which gives the following algorithm:

Algorithm 1 The one dimensional Forward Euler algorithm for solving the diffusion equation for an interval along the x-axis equal to $x \in [0, L]$ with a stepsize dx, and for a time interval $t \in [0, T]$ with a step size dt. We assume the initial conditions are given as fixed boundary conditions and one initial state at $t = t_0$. During one given time step the algorithm does $4N_x$ FLOPS, resulting in an $O(N) = 4N_xN_t$ FLOPS for the entire simulation.

$Nt = T/dt$
$Nx = /dx$
$\alpha = dt/dx^2$
$\beta = 1 - 2\alpha$
$U = zeros((Nt, Nx))$ ▷ Creating solution matrix
for $j = 2, 3, 4, ..., Nt$ **do**
    $U[j, :] = U[j - 1, :]$
    for $i = 1, 2, 3, 4, ...Nx - 1$ **do**
        $U[j + 1, i] = \alpha(U[j, i + 1] + U[j, i - 1]) + \beta U[j, i]$
    end for
end for

Implicit Backward Euler

Instead of differentiating the left hand side of equation 5 with a forward step, we can insteead take a backwards step, giving us the following equation:

$$\frac{u(x,t) - u(x,t-\Delta t)}{\Delta t} = \frac{u(x+\Delta x,t) - 2u(x,t) + u(x-\Delta x,t)}{\Delta x^2} \tag{10}$$

This can be restructured in a similar fashion to equation 8, except thiis time we are no longer working with an explicit method, but an implicit one. We have

$$u_{i,j-1} = -\alpha u_{i+1,j} + (1+2\alpha)u_{i,j} - \alpha u_{i-1,j} \tag{11}$$

This can in a similar fashion to the differential equation discussed in [2] be set up as a matrix equation, where we define U_j to be the vector of the values of $u(x_i, t_j)$ for all x_i at a time $t = t_j$

$$\mathbf{U}_{j-1} = \hat{A}\mathbf{U}_j = \begin{bmatrix} 1+2\alpha & -\alpha & 0 & \dots & 0 & 0 \\ -\alpha & 1+2\alpha & -\alpha & \dots & 0 & 0 \\ 0 & -\alpha & 1+2\alpha & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1+2\alpha & -\alpha \\ 0 & 0 & 0 & \dots & -\alpha & 1+2\alpha \end{bmatrix} \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ u_{3,j} \\ \vdots \\ u_{N-1,j} \\ u_{N,j} \end{bmatrix} \tag{12}$$

This is a Tridiagonal Toeplitz matrix, and can therefore be row reduced more efficiently than through Gaussian Elimination, as shown in [2]. The method discussed by us in the mentioned report is a good starting point, but the method is made for a second order ordinary differential equation. Yet this does not pose any further challenge than having do the process several times, as there are no limitations in the algorithm on it being performed multiple times, and both algorithms are developed with matrix equations on the form shown in equation 12. This gives us then the following algorithm to solve equation 5 using implicit backward Euler:

Algorithm 2 The one dimensional Backwards Euler algorithm for solving the diffusion equation for an interval along the x-axis equal to $x \in [0, L]$ with a stepsize dx, and for a time interval $t \in [0, T]$ with a step size dt. We assume the initial conditions are given as fixed boundary conditions and one initial state at $t = t_0$. During one given timestep the algorithm does $8N_x$ FLOPS resulting in $O(N) = 8N_xN_t$ FLOPS for the entire simulation

$Nt = T/dt$
$Nx = /dx$
$\alpha = dt/dx^2$
$\beta = 1 + 2\alpha$
$U = zeros((Nt, Nx))$ ▷ Creating solution matrix
for $j = 2, 3, 4, ..., Nt$ **do**
    $U[j, :] = U[j - 1, :]$
    $D = ones(Nx) * \beta$ ▷ Generating main diagonal of tridiagonal matrix
    for $i = 1, 2, 3, 4, ...Nx - 1$ **do** ▷ Forward substitution
        $q = -\alpha/D[i - 1]$
        $D[i] += -\alpha * q$
        $U[j, i] += q * U[j, i]$
    end for
    for $i = Nx - 1, Nx - 2, ..., 3, 2, 1$ **do** ▷ Backwards substitution
        $U[j, i - 1] += \alpha U[j - 1, i]$
        $U[j, i - 1] /= D[i - 1]$
    end for
end for

The Implicit Crank-Nicolson Scheme

The Crank-Nicolson scheme is derived if we instead of differentiating the position for only $t = t_j$, we can add in a second term for $t = t_{j+1}$. This gives that equation 5 can be approximated to

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

If we then separate the terms for $t = t_j$ and $t = t_{j+1}$ to their own sides of the equations, and introduce the variable $\alpha = \frac{\Delta t}{2\Delta x}$ we can then rewrite the equation above to be

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

The left and right hand side is both matrices, and can easily be recognised to be the matrices in equations 12 and 9 respectively, creating the following matrix equation:

$$\begin{bmatrix} 1-2\alpha & \alpha & \dots & 0 & 0 \\ \alpha & 1-2\alpha & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1-2\alpha & \alpha \\ 0 & 0 & \dots & \alpha & 1-2\alpha \end{bmatrix} \mathbf{U}_j = \begin{bmatrix} 1+2\alpha & -\alpha & \dots & 0 & 0 \\ -\alpha & 1+2\alpha & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1+2\alpha & -\alpha \\ 0 & 0 & \dots & -\alpha & 1+2\alpha \end{bmatrix} \mathbf{U}_{j+1} \tag{15}$$

From this equation it is clear that neither of the two previous algorithms alone are suited to the task of finding U_{j+1}. But since we assume U_j is known, we can in fact use the forward Euler

algorithm 1 in conjunction with the backward Euler algorithm 2 by first using the forward Euler algorithm on the left hand side of the equation to find a vector $\mathbf{R}_j$. When this vector is found, we get a problem similar to the backward Euler problem, and by then using algorithm 2 we will then have taken one step with the implicit Crank Nicolson scheme. In practice, it results in the following algorithm:

Algorithm 3 The one dimensional Crank Nicolson algorithm for solving the diffusion equation for an interval along the x-axis equal to $x \in [0, L]$ with a stepsize dx, and for a time interval $t \in [0, T]$ with a step size dt. We assume the initial conditions are given as fixed boundary conditions and one initial state at $t = t_0$. We here assume the Backward Euler algorithm shown in algorithm 2 is implemented correctly as a function *BWEuler(Uj,R, alpha)* while the forward Euler algorithm 1 is implemented correctly as a function *FWEuler(Uj, alpha)*. As the forward Euler algorithm does $4N_xN_t$ FLOPS and the Backward Euler algorithm does $8N_xN_t$ FLOPS for the entire simulation, it then follows that this algorithm performs at $O(N) = 12N_xN_t$

$N_t = T/dt$
$N_x = L/dx$
$\alpha = 0.5 * dt/dx^2$
$\beta = 1 + 2\alpha$
$U = zeros((Nt, Nx))$ ▷ Creating solution matrix
for $j = 2, 3, 4, ..., Nt$ **do**
    $U[j, :] = U[j - 1, :]$
    $R = FWEuler(U[j, :], alpha)$
    $U[j, :] = BWEuler(U[j, :], R, -alpha)$
end for

Using the Algorithms for 2-D Problems

Equation 4 does however also hold for multidimensional objects. In case of a two-dimensional diffusion problem, we would have to add a y-dependence to equation 5, giving us a new equation:

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

This equation can then for example be differentiated like we did for the one dimensional Forward

$$\frac{u_{j+1,k,i} - u_{j,k,i}}{\Delta t} = \frac{u_{j,k,i+1} - 2u_{i,j,k} + u_{j,k,i-1}}{\Delta x^2} + \frac{u_{j,k+1,i} - 2u_{j,k,i} + u_{j,k-1,i}}{\Delta y^2} \tag{17}$$

where the i index then is the coordinate along the x-direction and k is the coordinate in the y-direction, making $u_{j,k,i}$ the function value at a time t_j in the coordinate (k, i) in the ki-grid. Similar differentiations can be done for the Backward Euler and Crank Nicolson schemes, where we then simply get another term which now is dependent on y. Luckily, these differentiations can be done irrespectively of each other. The reason for this is that this system can best be understood as a grid of function values on the form

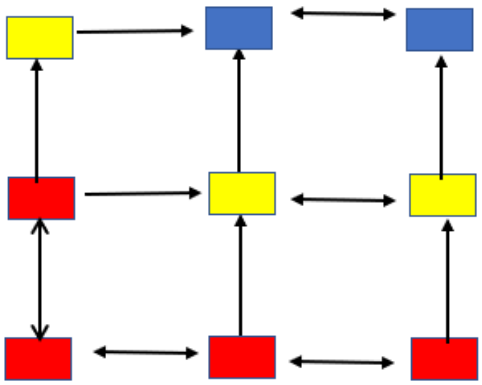


Figure 1: The grid is a simplified gird showing how the discretised diffusion equation works in reality. A net heat flow is going from the warm grid points to those with a lower temperature, here visulised with red (warm), yellow (medium) and blue cold.

As one can see on the figure, it is possible to hold one of the positional variables constant while integrating the other. Thus the algorithms 1 through 3 requires very little modification, as each the algorithms only has to be run twice, one in each direction.

The Lithosphere

Now that we have a system via which we may approximate the diffusion equation (and therefore the heat equation) numerically, we can apply these abstract concepts to a real-life scenario – the radioactive refertilization of the Fennoscandic[1] mantle.

More specifically, geologists theorize that there once existed a subduction zone in the Fennoscandic mantle. An interesting consequence of such a subduction zone is that it would have lead to contact between the mantle and a vast number of ocean sediments[2]. As a result, the heat generation in the mantle is theorized to have increased due to the radioactive decay of these sediments.

To investigate this, we will be creating a more detailed model in which heat generation occurs over the entire simulated array, at rates defined by its depth. We will be creating a model that represents a 150km deep, 200km wide 2-D slice of the lithosphere; we will divide this array into three depths:

Name	Upper Depth [km]	Lower Depth [km]	Heat Production [$\mu W/m^3$]
Upper Crust	0	20	1.4
Lower Crust	20	40	0.35
Mantle	40	120	0.05

Table 1: The heat production for different sections of the lithosphere.

[1] Fennoscandia refers to the Scandinavian peninsula, including Finland.

[2] These may have contained a variety of relatively unstable elements, such as Uranium, Thorium, and Potassium.

We will be using the heat equation presented in (3), and as such we will set our thermal conductivity to $k = 2.5\text{W/m}^\circ\text{C}$, our density to $\rho = 3.5 \times 10^3$ kg/m^3, and our specific heat capacity to $c_p = 1000\text{J/kg}^\circ\text{C}$.

To simulate the radioactive refertilization of the mantle, we will also be adding a variable heat production to the mantle. We will begin with an initial heat production of $Q_{\text{rad},0} = 0.5\mu\text{W/m}^3$, but this will decline over time in a manner determined by the half-lives of the decaying particles. To calculate this decline, we will be focusing on the decay of Uranium, Thorium, and Potassium, and assume the following:

Element	Half-Life [Gy]	Percent of Sediments
Uranium	4.47	40%
Thorium	14.0	40%
Potassium	1.25	20%

Table 2: Distribution and half-lives of the most prominent radioactive sediments in the mantle.

Since the radioactive heat production decreases over time, we must create a function $Q_{\text{rad}}(t)$ using the above data. To begin, we will need a general equation for *exponential decay*:

$$I(t) = I_0 \left(\frac{1}{2}\right)^{\frac{t}{t_{1/2}}}$$

Where $I(t)$ is the intensity of an exponentially decreasing process over time, I_0 is the initial intensity, t is time, and $t_{1/2}$ is the half-life of the governing quantity.

We can apply these principles to a system of three decaying quantities, via the following:

$$Q_{\text{rad}}(t) = Q_{\text{rad},0}\left(0.4\left(\frac{1}{2}\right)^{\frac{t}{4.47\times10^9}} + 0.4\left(\frac{1}{2}\right)^{\frac{t}{1.4\times10^{10}}} + 0.2\left(\frac{1}{2}\right)^{\frac{t}{1.25\times10^9}}\right)$$

Finally, we can use all the information presented in this subsection, in conjunction with (3), to give us the differential equation we will be solving:

$$2.5\nabla^2 T + Q_{\text{tot}}(t, y) = 3.5 \times 10^6 \frac{\partial T}{\partial t} \tag{18}$$

With the total heat production Q_{tot} given by two terms:

$$Q_{\text{tot}}(t, y) = Q(y) + Q_{rad}(t) \tag{19}$$

Where the first term depends on the depth y:

$$Q(y) = \begin{cases} 1.4, & \text{if } -20 < y < 0 \\ 0.35, & \text{if } -40 \leq y < 20 \\ 0.05, & \text{if } -120 \leq y < -40 \end{cases} \tag{20}$$

And the second term is given by:

$$Q_{\text{rad}}(t) = 0.2 \left(\frac{1}{2}\right)^{\frac{t}{4.47\times10^9}} + 0.2 \left(\frac{1}{2}\right)^{\frac{t}{1.4\times10^{10}}} + 0.1 \left(\frac{1}{2}\right)^{\frac{t}{1.25\times10^9}} \tag{21}$$

This is unlikely to have a nice analytical solution, but if we omit the radioactive heat producing term, we can apply the methods presented in **Solving the Diffusion Equation** and **The Heat Equation** to find an analytical result.

Results

We attempted to compare the analytical solution to the 1-D diffusion equation to the corresponding numerical results, and ended up with the following result for a 1-D row of zeros, appended by a single one. The simulation ran for 0.1 dimensionless units of time, with $dt = 10^{-3}$; the row has a dimensionless length of 0.2, with $dx = 10^{-2}$.

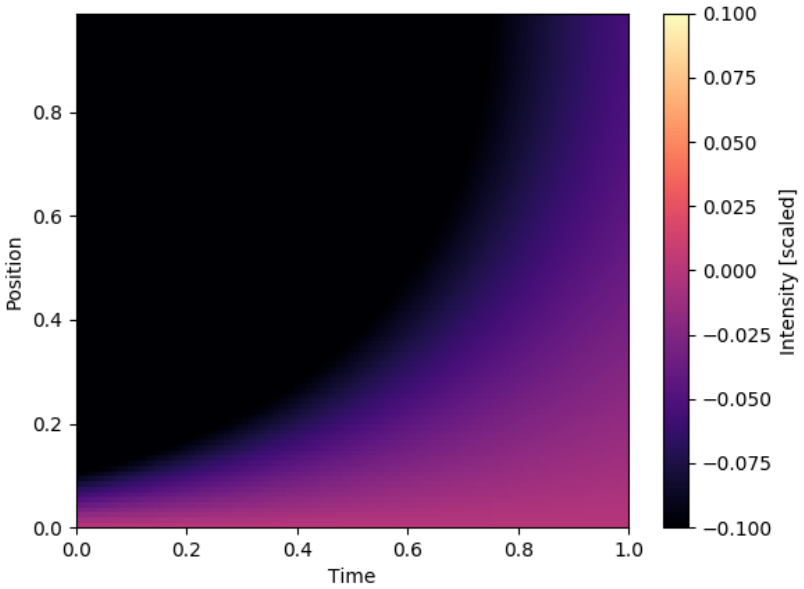


Figure 2: The analytical solution for the diffusion equation.

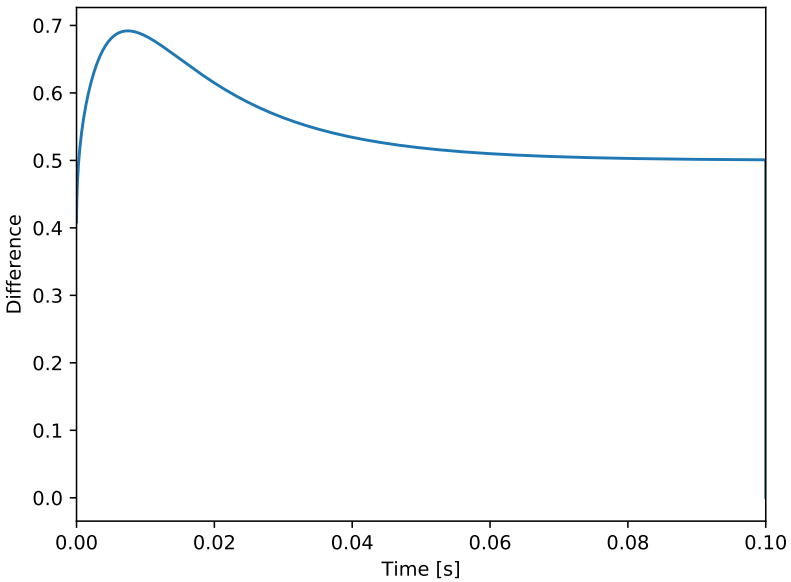


Figure 3: The mean difference in the 1-D numerical and analytical solution of the diffusion equation over time, using the *Forward Euler* algorithm.

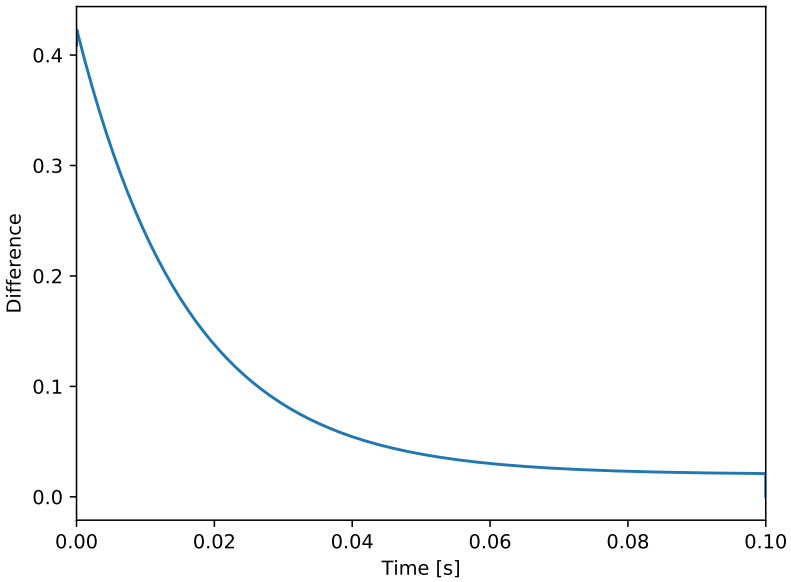


Figure 4: The mean difference in the 1-D numerical and analytical solution of the diffusion equation over time, using the *Backward Euler* algorithm.

Using the same simulation settings, we then changed dt such that the ratio $\frac{dt}{dx^2}$ would vary between 0.1 and 0.75, and ran each simulation for 50 different ratios and kept the mean difference in each final time step:

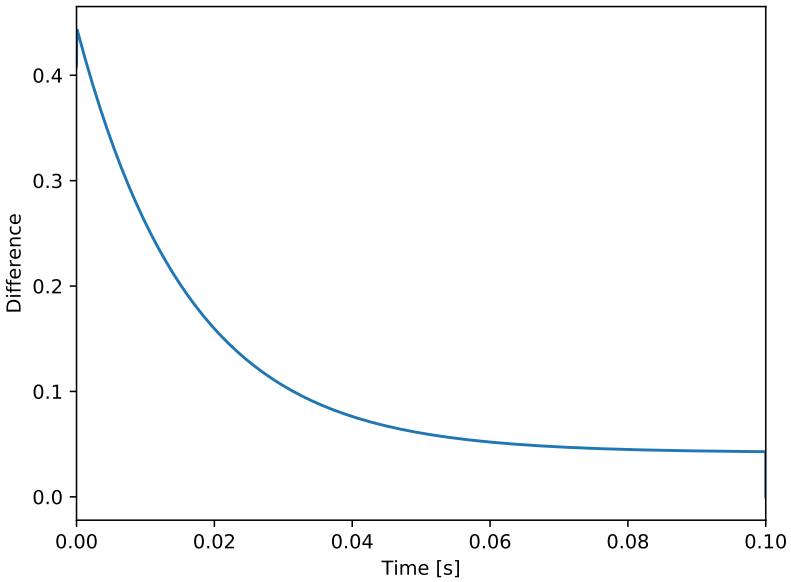


Figure 5: The mean difference in the 1-D numerical and analytical solution of the diffusion equation over time, using the *Crank-Nicolson* algorithm.

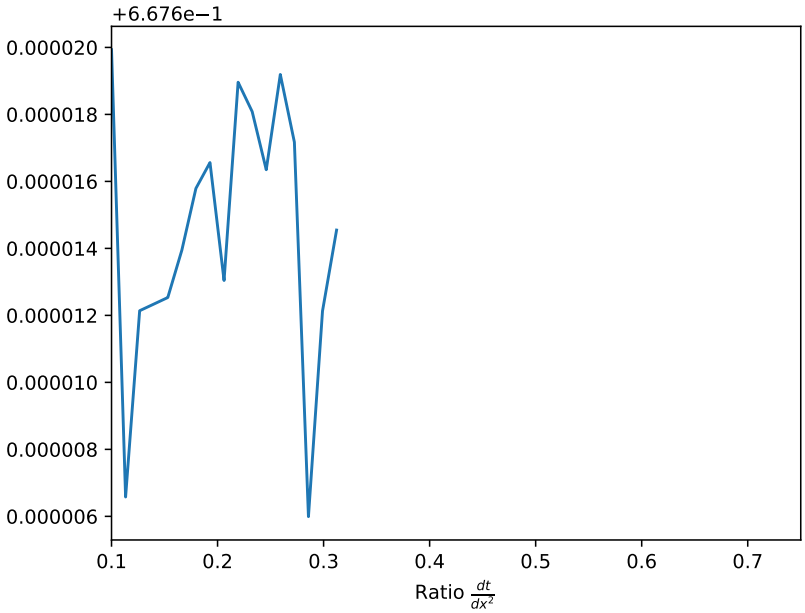


Figure 6: The mean difference in the 1-D numerical and analytical solution of the diffusion equation as a function of the ratio $\frac{dt}{dx^2}$, using the *Forward Euler* algorithm.

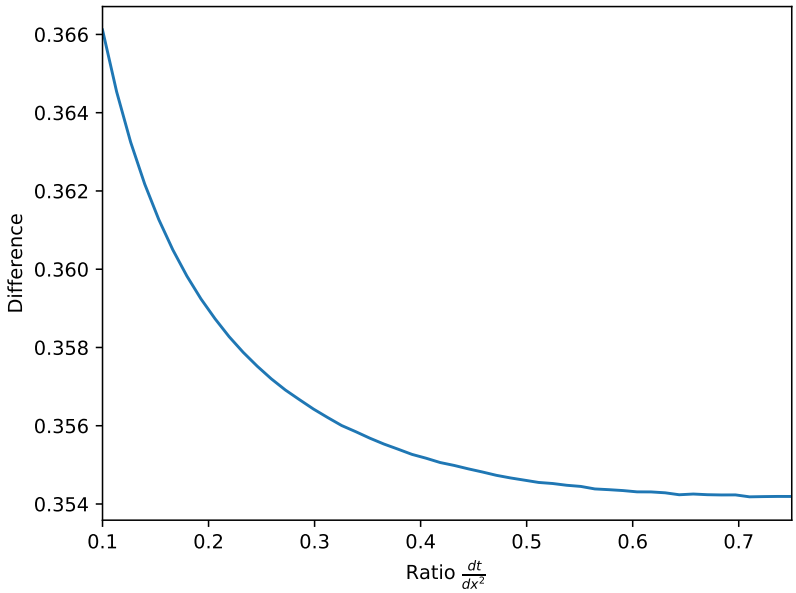


Figure 7: The mean difference in the 1-D numerical and analytical solution of the diffusion equation as a function of the ratio $\frac{dt}{dx^2}$, using the *Backward Euler* algorithm.

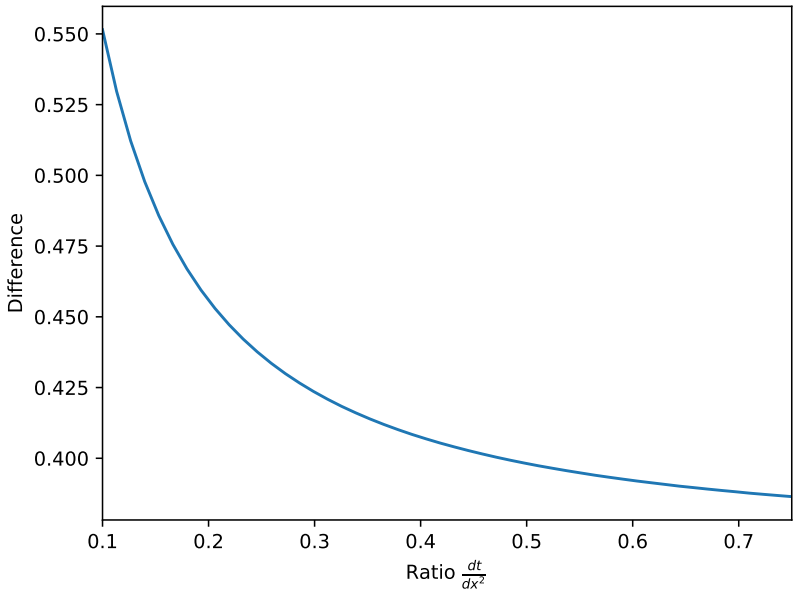


Figure 8: The mean difference in the 1-D numerical and analytical solution of the diffusion equation as a function of the ratio $\frac{dt}{dx^2}$, using the *Crank-Nicolson* algorithm.

Next, we implemented the algorithms presented in 1, 2, and 3 on a simple set of initial conditions, whereby we have a grid of zeros whose right side is composed of ones. Each simulation ran for 100 dimensionless units of time, with a time step of $dt = 10^{-3}$; each grid has a height and width of 40×10, with $dx = 1$.

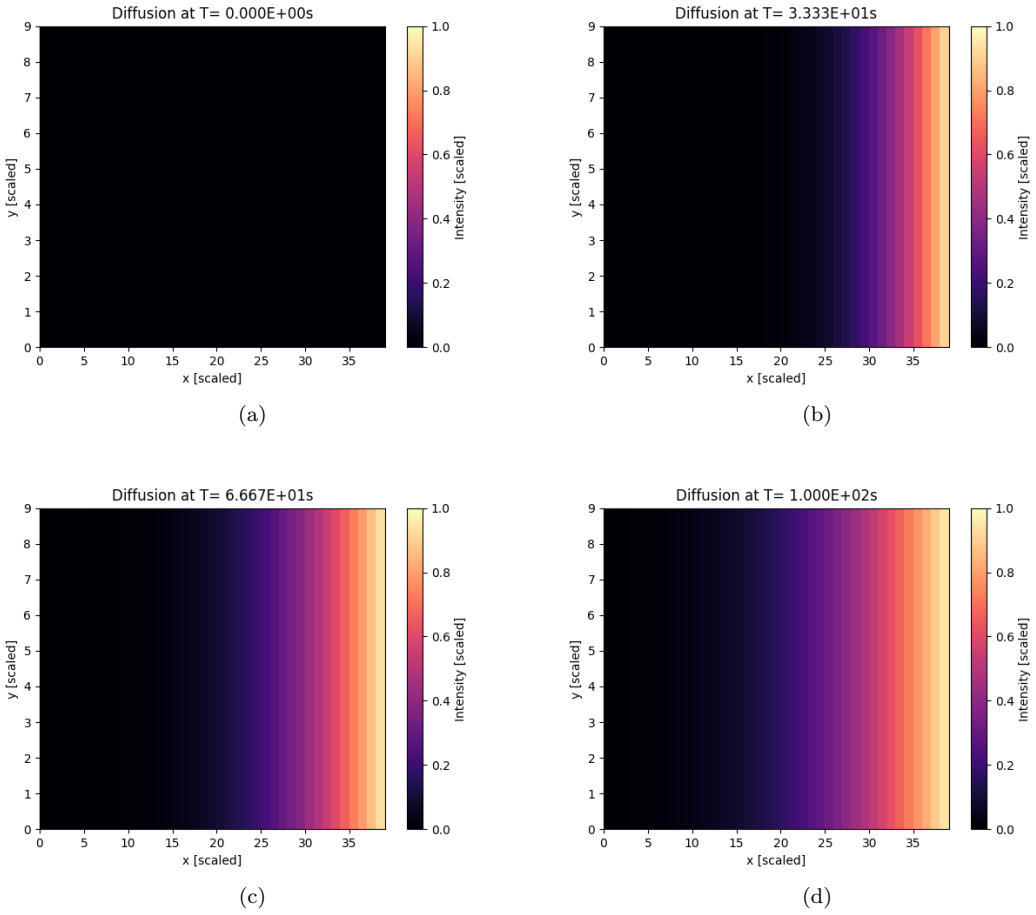
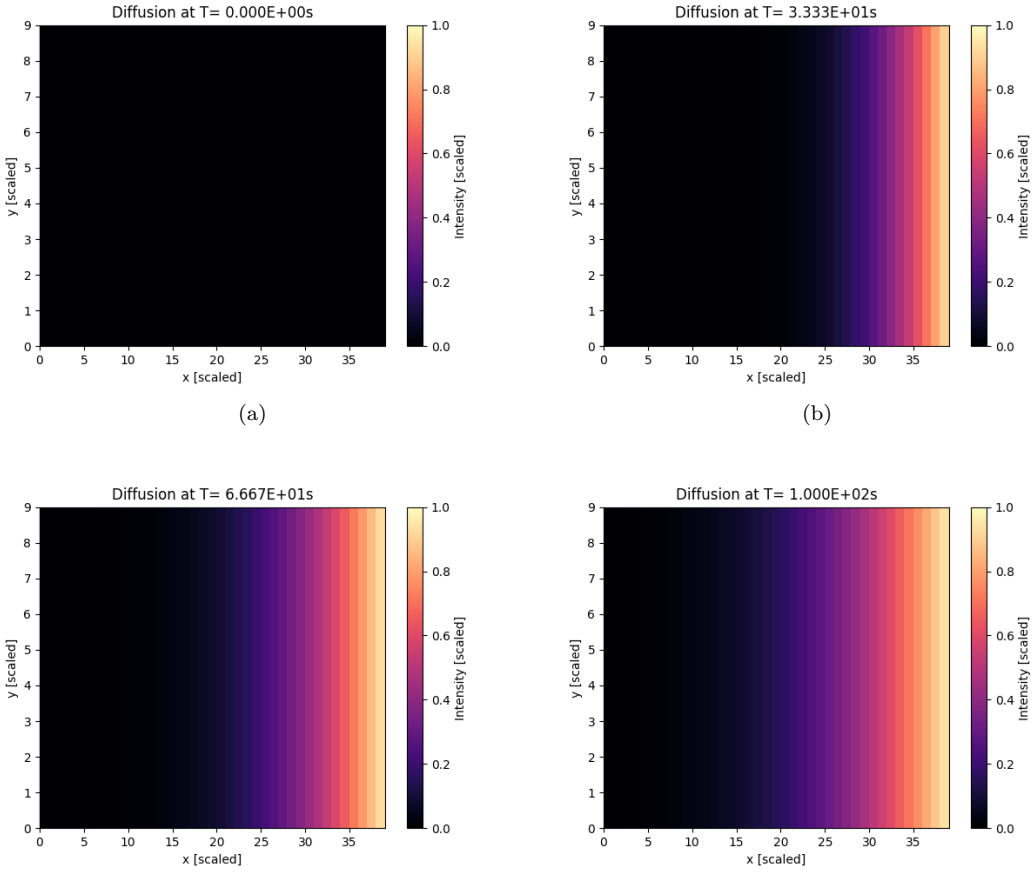


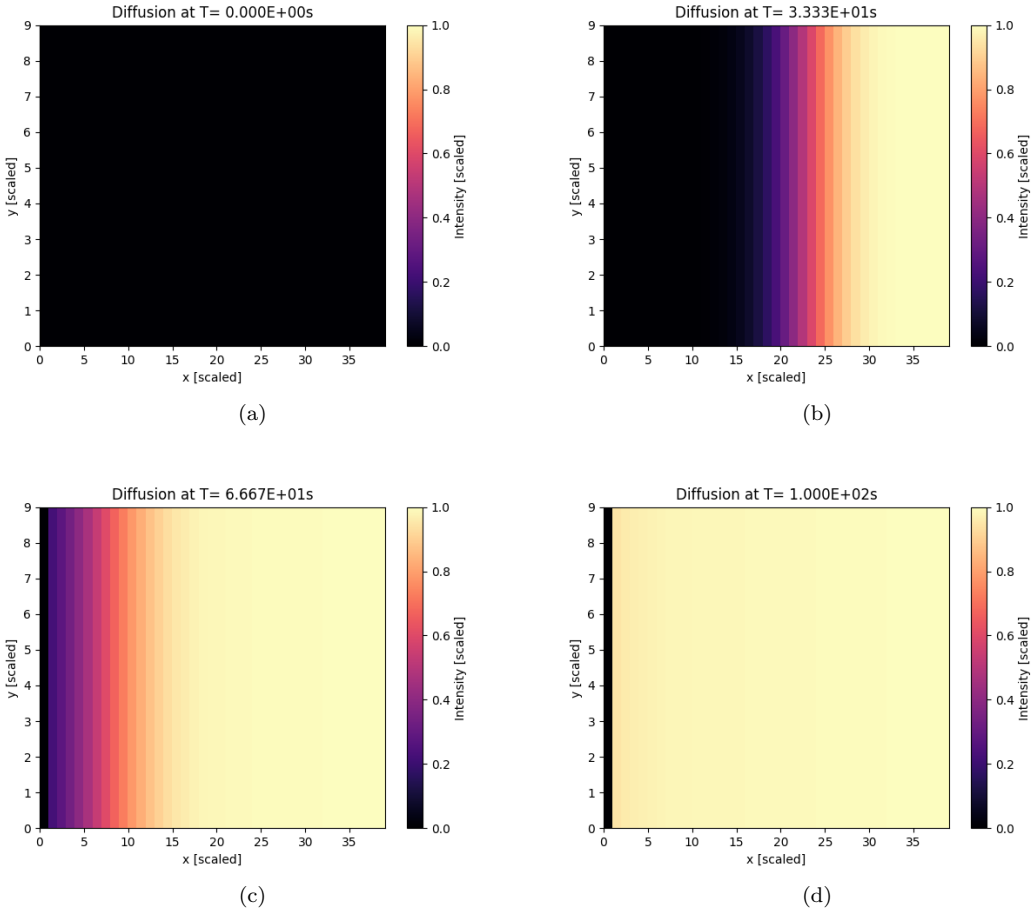
Figure 9: The 2-D diffusion equation using the *Forward Euler* algorithm.



(a) (b)

(c) (d)

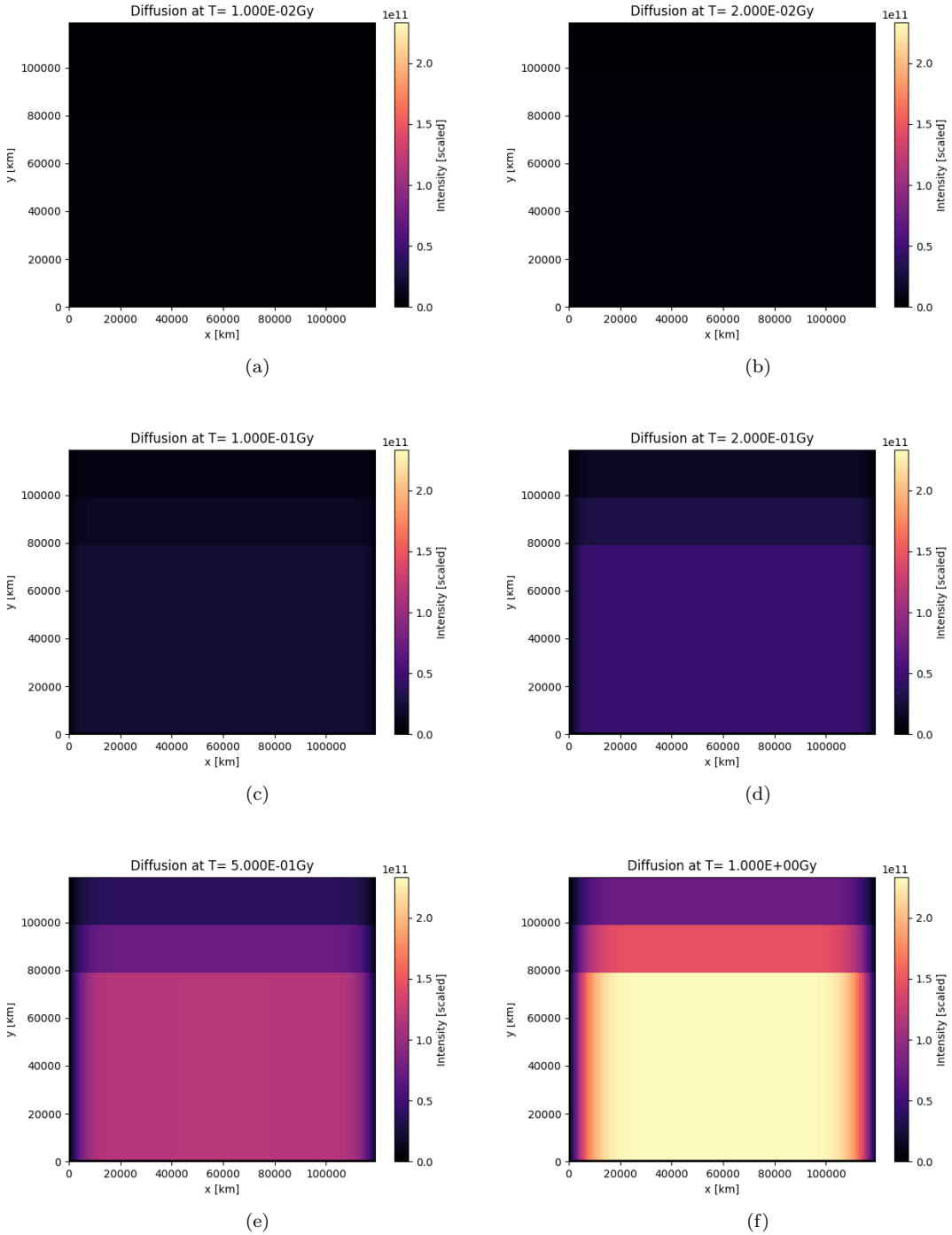
Figure 10: The 2-D diffusion equation using the *Backward Euler* algorithm.



(a) (b) (c) (d)

Figure 11: The 2-D diffusion equation using the *Crank Nicolson* algorithm.

Using the 2-D Forward Euler algorithm, we modeled the Fennoscandian lithosphere over the course of one-billion years, and over an area 150km deep and 150km wide. We used a time step of $dt = 5 \times 10^6$ years, with $dx = 10^3$ km.



(a) (b) (c) (d) (e) (f)

Figure 12: The 2-D diffusion equation modeling the lithosphere model presented in **The Lithosphere**, using the *Forward Euler* algorithm.

Discussion

The 1-D Diffusion Equation

We begin with the analytical solution shown in Figure 2, whose time-position distribution matches our expectation that the heat will spread at a rate governed by equation (2).

Using this, we then have Figures 3, 4, and 5 which show the mean differences between the Forward Euler, Backward Euler, and Crank-Nicolson algorithms and the analytical solution as a function of time, which show that the model is more precise as time passes, rather then initially precise and less precise over time.

To determine how to best set up our ratio $\frac{dt}{dx^2}$, we see in Figure 6 that the Forward Euler algorithm breaks down when $\frac{dt}{dx^2} > 0.4$, and that the difference between the analytical and numerical solutions are minimized near this point.

The Backward Euler algorithm shown in Figure 7 is not restricted in such a way, nor is its analytical-numerical difference minimized at the aforementioned ratio, but the model doesn't break down once the ratio increases past that point and simply becomes more precise as the ratio increases.

As for the Crank-Nicolson algorithm shown in Figure 8, we see that the ratio gets continuously better as it increases, mirroring much of the behavior of the Backward Euler algorithm.

It is important to note that the backward Euler algorithm, as we implemented it, seems to be slightly better than the Crank-Nicolson scheme. Considering the fact that Crank-Nicolson is a higher degree numerical scheme, with an error of $O(\Delta t^2)$ while Backward Euler is a first order method, as explained in [3]. This becomes especially clear if we compare the figures 7 and 8, where while their performance is similar, the Backwards Euler scheme does slightly better. If one then looks at figures 11 one can see the temperature in the system rapidly does increase, which doesn't happen at all in the backward Euler case.

The 2-D Diffusion Equation

Our results in Figures 9 and 10 appear to match quite well, these being the Forward and Backward Euler alogorithm results. As expected, the heat spreads from the right to left and dissipates as a function of distance, in a nearly identical way. This implies that our results are likely correct, as these algorithms differ rather significantly in their implementations.

On the other hand, the Crank-Nicolson algorithm results shown in Figure 11 don't appear to give the same results as our two other algorithms. Here, the heat spreads much more intensely as we move away from the section adjacent to the right boundary, when compared to Figures 9 and 10.

The Lithosphere

In our lithosphere model, we investigated the heat distribution in the mantle and crust over a billion years; given the results shown in Figure 12, we found that these generally matched our expectations – the heat was most concentrated in the mantle and slowly spread upwards toward the lower and upper crusts. In addition to this, we noticed that the temperature in the mantle tended to increase over time without stopping, likely due to the fact that the radioactive decay

had not yet decreased by a significant enough amount at that point in time.

Conclusion

Overall, it appears as though our numerical models are imperfect when compared to their analytical counterpart, yet we can still see that the general trend and rates of diffusion remain proportional to those found in the analytical solutions – this suggests that there may be a scaling issue with our algorithms that managed to get by unnoticed. Nevertheless, we can conclude that our numerical approximations are close enough to be used with confidence in our lithospheric model, as the differences in our 1-D analytical and numerical solutions are not so large as to merit the discounting of our models.

We also have reason to believe that the Crank-Nicolson scheme was implemented incorrectly, as the diffusion is extremely rapid and ends up dominating the entire system. Our most likely guess for this is that the lower boundary condition $u(0,t) = 0$ wasn't correctly implemented in the algorithm, but due to time constraints we did not manage to fix this problem.

Having concluded that our model is close enough, we then carried on our lithosphere simulation, and observed that the temperature in the mantle increased significantly over the course of a billion years; this makes sense, as the radioactive heat production in the mantle did not decrease by a noticeable amount over the course of the simulation, as the shortest half-life was longer than the simulation runtime.

Unfortunately, we cannot conclude with certainty as to whether or not our lithosphere model follows the true course of the Fennoscandian geological evolution, however it is somewhat unlikely that the temperature would increase so much over the last billion years without consequence to the surrounding environment, so there is most likely a number of factors being left our of this simulation that would more accurately reflects reality.

References

[1] K. Rottmann, *Matematisk formelsamling*. Bracan forl., 1995.

[2] L. W. Dreyer and G. S. Cabrera, "Gaussian elimination and algorithmic optimisation," Sep 2018. [Online]. Available: https://github.uio.no/larswd/FYS3150/blob/master/project1/LaTeX/Final/project_1_larswd_gabric.pdf

[3] M. Hjorth-Jensen, *Computational Physics*. [Online]. Available: https://github.com/CompPhysics/ComputationalPhysics/blob/master/doc/Lectures/lectures2015.pdf