FYS3150 - Project 5 - Diffusion of neurotransmitters in the synaptic cleft

Vilde Flugsrud and Mari Dahl Eggen

November 22, 2015

Contents

1 Introduction **3**

2 Theory **3**

2.1 Analytical solution of the diffusion equation 3

2.2 Numerical solution of the diffusion equation 7

2.2.1 The explicit Forward Euler algorithm 7

2.2.2 The implicit Backward Euler algorithm 7

2.2.3 The implicit Crank-Nicolson scheme 8

2.2.4 Monte Carlo methods and random walks 9

2.3 Truncation errors 9

2.3.1 Forward Euler 10

2.3.2 Backward Euler 10

2.3.3 Crank-Nicolson scheme 11

2.4 Stability properties 11

2.4.1 Implicit Backward Euler 11

2.4.2 Explicit Forward Euler 13

2.4.3 Implicit Crank-Nicolson scheme 14

3 Method **16**

3.1 Simulation of diffusion of neurotransmitters 16

3.2 Implementation of algorithms 18

3.2.1 Forward Euler 18

3.2.2 Backward Euler 18

3.2.3 Crank-Nicolson 18

3.3 Unit tests and verification of results 18

3.3.1 Unit tests in Forward Euler, Backward Euler and the Crank-Nicolson scheme 18

3.3.2 Verification of results 18

4 Result and discussion **19**

4.1 Truncation error and stability properties 19

4.2 Numerical calculated results 19

4.3 Relative error in numerical calculated results 19

4.4 Crank-Nicolson scheme 19

5 Conclusion **19**

6 Comments **19**

7 References **19**

Abstract

hei

1 Introduction

2 Theory

2.1 Analytical solution of the diffusion equation

We will now find a closed-form solution of the diffusion equation in one dimension, where the diffusion constant is set to $D = 1$, so the partial differential equation to solve is

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t}. \tag{1}$$

The initial condition is

$$u(x, 0) = 0 \quad \text{for} \quad 0 < x < 1,$$

and the boundary conditions are

$$u(0, t) = 1 \quad \text{for} \quad t > 0, \quad \text{and} \quad u(1, t) = 0 \quad \text{for} \quad t > 0.$$

We are given the steady-state solution of the Eq. (1) for our given boundary conditions, that is, the solution of the system when u is independent of time. It is $u_s(x) = 1 - x$. To solve the time dependent equation we introduce a new function $v(x, t) = u(x, t) - u_s(x)$, so that we get the new initial condition

$$v(x, 0) = -u_s(x) \quad \text{for} \quad 0 < x < 1, \tag{2}$$

and the new boundary conditions

$$v(0, t) = 0 \quad \text{for} \quad t > 0, \quad \text{and} \quad v(1, t) = 0 \quad \text{for} \quad t > 0. \tag{3}$$

We can use the function $v(x,t)$ because

$$\frac{\partial^2(u(x,t)-u_s(x))}{\partial x^2} = \frac{\partial(u(x,t)-u_s(x))}{\partial t} \quad \Rightarrow \quad \frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t}.$$

We see that the general solution we find for $u(x,t)$ also will hold for $v(x,t)$. To find a solution we make the ansatz

$$u(x,t) = z(x)w(t),$$

so

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t} \quad \Rightarrow \quad u_{xx} = u_t \quad \Rightarrow \quad wz_{xx} = zw_t \quad \Rightarrow \quad \frac{z_{xx}}{z} = \frac{w_t}{w}.$$

Now the left hand side and the right hand side of the equal sign is independent of each other, so they have to be constant. We introduce the constant $-\lambda^2$, so that

$$\frac{z_{xx}}{z} = \frac{w_t}{w} = -\lambda^2,$$

and then the two differential equations

$$z_{xx} + \lambda^2 z = 0 \quad \text{and} \quad w_t + \lambda^2 w = 0$$

are left to be solved. These are two standard differential equations with general solutions

$$z(x) = A\cos(\lambda x) + B\sin(\lambda x) \quad \text{and} \quad w(t) = Ce^{-\lambda^2 t}.$$

If we put these solutions together we find the general solution of our problem in Eq. (1), for the new function $v(x,t)$.

$$v(x,t) = u(x,t) - u_s(x) = \left(A\cos(\lambda x) + B\sin(\lambda x)\right) Ce^{-\lambda^2 t} \tag{4}$$

Now we can use the boundary conditions in Eq. (3) to find the closed-form solution of $v(x,t)$.

$$v(0,t) = (A\cos(0) + B\sin(0))\, Ce^{-\lambda^2 t} = ACe^{-\lambda^2 t} = 0 \quad \Rightarrow \quad A = 0$$

$$v(1,t) = (A\cos(\lambda) + B\sin(\lambda))\, Ce^{-\lambda^2 t} = B\sin(\lambda)Ce^{-\lambda^2 t} = 0$$

$$\Rightarrow \quad \sin(\lambda) = 0 \quad \Rightarrow \quad \lambda_n = n\pi \quad \text{for} \quad n = 1, 2, 3, ...$$

Then we have

$$v_n(x,t) = B_n C_n \sin(n\pi x) e^{-(n\pi)^2 t} = A_n \sin(n\pi x) e^{-(n\pi)^2 t},$$

and

$$v(x,t) = \sum_{n=1}^{\infty} A_n \sin(n\pi x) e^{-(n\pi)^2 t}$$

At last we can use the initial condition in Eq. (2) to find an expression for the n-dependent constant A_n.

$$v(x,0) = \sum_{n=1}^{\infty} A_n \sin(n\pi x) = -u_s(x) = x - 1$$

We recognize this as a Fourier series. In general we have

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos\left(\frac{n\pi x}{P}\right) + b_n \sin\left(\frac{n\pi x}{P}\right) \right) = \sum_{n=1}^{\infty} b_n \sin\left(n\pi x\right), \tag{5}$$

where we have defined $a_0 = a_n = 0$ and $P = 1$. In the Fourier series b_n is defied as

$$b_n = \frac{2}{P} \int_{x_0}^{x_0+P} f(x) \cdot \sin\left(\frac{n\pi x}{P}\right) dx. \tag{6}$$

If we use Eq. (5) and (6) in our problem to find A_n we get

$$A_n = 2\int_0^1 (x-1)\sin(n\pi x)\,dx = 2\left[\int_0^1 x\sin(n\pi x)dx - \int_0^1 \sin(n\pi x)dx\right],$$

and by use of integration by parts we get

$$= 2\left[\left[-\frac{x}{n\pi}\cos(n\pi x)\right]_0^1 + \frac{1}{n\pi}\int_0^1 \cos(n\pi x)dx - \frac{1}{n\pi}\left[-\cos(n\pi x)\right]_0^1\right]$$

$$= 2\left[\left(-\frac{1}{n\pi}\cos(n\pi)\right) + \left[\frac{1}{n\pi}\cdot\frac{1}{n\pi}\sin(n\pi x)\right]_0^1 + \frac{1}{n\pi}\left(\cos(n\pi)-\cos(0)\right)\right]$$

$$= 2\left[-\frac{(-1)^n}{n\pi} + \frac{1}{n^2\pi^2}\left(\sin(n\pi)-\sin(0)\right) + \frac{1}{n\pi}\left((-1)^n-1\right)\right]$$

$$= 2\left[-\frac{(-1)^n}{n\pi} + \frac{(-1)^n}{n\pi} - \frac{1}{n\pi}\right] = -\frac{2}{n\pi},$$

which gives us the closed-form solution

$$v(x,t) = -\frac{2}{\pi}\sum_{n=1}^{\infty}\frac{1}{n}\sin(n\pi x)\,e^{-(n\pi)^2 t}. \tag{7}$$

This gives us the concentration of the neurotransmitters in the synaptic cleft as

$$u(x,t) = 1 - x - \frac{2}{\pi}\sum_{n=1}^{\infty}\frac{1}{n}\sin(n\pi x)\,e^{-(n\pi)^2 t}.$$

2.2 Numerical solution of the diffusion equation

2.2.1 The explicit Forward Euler algorithm

In the explicit forward Euler algorithm we have that

$$\frac{\partial u}{\partial t} = u_t \approx \frac{u(x_i, t_j + \Delta t) - u(x_i, t_j)}{\Delta t} = \frac{u_i^{j+1} - u_i^j}{\Delta t} \tag{8}$$

$$\frac{\partial^2 u}{\partial x^2} = u_{xx} \approx \frac{u(x_i + \Delta x, t_j) - 2u(x_i, t_j) + u(x_i - \Delta x, t_j)}{\Delta x^2} \tag{9}$$

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

which gives us Eq. (1) on the form

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

$$\Rightarrow \quad u_i^{j+1} = u_i^j + \alpha \left(u_{i+1}^j - 2u_i^j + u_{i-1}^j \right) \tag{10}$$

where $\alpha = \frac{\Delta t}{\Delta x^2}$. This is an explicit formula for the unknown, which is the value of the concentration at the time $j + 1$, and it can thus be calculated directly.

2.2.2 The implicit Backward Euler algorithm

The partial derivatives are in the implicit backward Euler algorithm given by

$$u_t \approx \frac{u(x_i, t_j) - u(x_i, t_j - \Delta t)}{\Delta t} = \frac{u_i^j - u_i^{j-1}}{\Delta t} \tag{11}$$

$$u_{xx} \approx \frac{u(x_i + \Delta x, t_j) - 2u(x_i, t_j) + u(x_i - \Delta x, t_j)}{\Delta x^2} \tag{12}$$

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

which gives us Eq. (1) on the form

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

$$\Rightarrow \quad u_i^{j-1} = u_i^j - \alpha\left(u_{i+1}^j - 2u_i^j + u_{i-1}^j\right), \tag{13}$$

where $\alpha = \frac{\Delta t}{\Delta x^2}$. This equation can be represented as a matrix equation on the form

$$U_{j-1} = \hat{A}U_j = (\mathbb{1} + \alpha\hat{B})U_j,$$

where

$$U_j = \begin{bmatrix} u_1^j \\ u_2^j \\ \vdots \\ u_n^j \end{bmatrix}, \quad U_{j-1} = \begin{bmatrix} u_1^{j-1} \\ u_2^{j-1} \\ \vdots \\ u_n^{j-1} \end{bmatrix}, \quad \hat{B} = \begin{bmatrix} 2 & -1 & 0 & 0 & \dots & 0 \\ -1 & 2 & -1 & 0 & & \vdots \\ 0 & -1 & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & \ddots & \ddots & -1 \\ 0 & \dots & \dots & \dots & -1 & 2 \end{bmatrix}, \tag{14}$$

and $\mathbb{1}$ is the identity matrix. Our unknown points are stored in U_j, and we can find them my solving the matrix equation

$$U_j = (\mathbb{1} + \alpha\hat{B})^{-1}U_{j-1}. \tag{15}$$

2.2.3 The implicit Crank-Nicolson scheme

In the Crank-Nicolson scheme, the derivatives we are looking at in this project, are given by

$$u_t \approx \frac{u(x_i, t_j + \Delta t) - u(x_i, t_j)}{\Delta t} = \frac{u_i^{j+1} - u_i^j}{\Delta t} \tag{16}$$

$$u_{xx} \approx \frac{1}{2}\left(\frac{u(x_i+\Delta x,t_j)-2u(x_i,t_j)+u(x_i-\Delta x,t_j)}{\Delta x^2}+\right. \tag{17}$$

$$\left.\frac{u(x_i+\Delta x,t_j+\Delta t)-2u(x_i,t_j+\Delta t)+u(x_i-\Delta x,t_j+\Delta t)}{\Delta x^2}\right).$$

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

which gives us Eq. (1) on the form

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

$$\Rightarrow \quad 2u_i^{j+1}-2u_i^j = \alpha\left(u_{i+1}^j-2u_i^j+u_{i-1}^j+u_{i+1}^{j+1}-2u_i^{j+1}+u_{i-1}^{j+1}\right)$$

$$\Rightarrow \quad -\alpha u_{i-1}^j+(-2+2\alpha)u_i^j-\alpha u_{i+1}^j = \alpha u_{i-1}^{j+1}+(-2-2\alpha)u_i^{j+1}+\alpha u_{i+1}^{j+1}$$

$$\Rightarrow \quad (-2\mathbb{1}+\alpha\hat{B})U_j = (-2\mathbb{1}-\alpha\hat{B})V_{j+1} \quad \Rightarrow \quad (2\mathbb{1}-\alpha\hat{B})U_{j-1} = (2\mathbb{1}+\alpha\hat{B})U_j$$

$$U_j = (2\mathbb{1}+\alpha\hat{B})^{-1}(2\mathbb{1}-\alpha\hat{B})U_{j-1}, \tag{18}$$

where U_j, U_{j-1} and $\hat{B}$ is given in Eq. (14) and $\mathbb{1}$ is the identity matrix. We need to solve this matrix equation to find our unknown points.

2.2.4 Monte Carlo methods and random walks

2.3 Truncation errors

The basic idea behind Forward Euler, Backward Euler and Crank-Nicolson scheme is to advance a solution at point $u(x_i,t_j)$, to a solution at point $u(x_{i+1},t_{j+1})$. This three methods stems from the linear Taylor polynomial

$$f(x) = \sum_{n=0}^{\infty}\frac{f^{(n)}(a)}{n!}(x-a)^n, \tag{19}$$

and therefore we can use it to estimate the truncation error in the three methods. Since we are looking at a function of one dimension in space and one dimension in time, we have to find the truncation error for both separately. The resulting truncation errors are listed in Table 1.

2.3.1 Forward Euler

First we will find the local truncation error in time. Then we need to find a expression for Eq. (8), by use of Eq. (19). We are looking for a solution at point u_i^{j+1}, so that is the point we will express as a Taylor polynomial.

$$u_i^{j+1} = u_i^j + (u_i^j)_t \Delta t + \frac{(u_i^j)_{tt}}{2}\Delta t^2 + \dots$$

$$\Rightarrow \quad (u_i^j)_t \simeq \frac{u_i^{j+1} - u_i^j}{\Delta t} - \frac{(u_i^j)_{tt}}{2}\Delta t = \frac{u_i^{j+1} - u_i^j}{\Delta t} - \mathcal{O}(\Delta t)$$

We see that the biggest error term in Eq. (8) is of the order $\mathcal{O}(\Delta t)$. Now we will do the same procedure, a bit extended, to find the local truncation error in space.

$$I: \quad u_{i+1}^j = u_i^j + (u_i^j)_x \Delta x + \frac{(u_i^j)_{xx}}{2}\Delta x^2 + \frac{(u_i^j)_{xxx}}{6}\Delta x^3 + \frac{(u_i^j)_{xxxx}}{24}\Delta x^4 + \dots$$

$$II: \quad u_{i-1}^j = u_i^j - (u_i^j)_x \Delta x + \frac{(u_i^j)_{xx}}{2}\Delta x^2 - \frac{(u_i^j)_{xxx}}{6}\Delta x^3 + \frac{(u_i^j)_{xxxx}}{24}\Delta x^4 + \dots$$

$$I + II: \quad u_{i+1}^j + u_{i-1}^j \simeq 2u_i^j + (u_i^j)_{xx}\Delta x^2 + \frac{(u_i^j)_{xxxx}}{12}\Delta x^4$$

$$(u_i^j)_{xx} \simeq \frac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{\Delta x^2} - \frac{(u_i^j)_{xxxx}}{12}\Delta x^2 = \frac{u_{i+1}^j - 2u_i^j + u_{i-1}^j}{\Delta x^2} - \mathcal{O}(\Delta x^2)$$

The biggest error term in Eq. (9) is then of order $\mathcal{O}(\Delta x^2)$.

2.3.2 Backward Euler

We use the same procedure as we did in the previous section to find the local truncation error in time for Backward Euler. We then want to find an expression for Eq. (11).

$$u_i^{j-1} = u_i^j - (u_i^j)_t \Delta t + \frac{(u_i^j)_{tt}}{2} \Delta t^2 + \dots$$

$$\Rightarrow \quad (u_i^j)_t = \frac{u_i^j - u_i^{j-1}}{\Delta t} + \frac{(u_i^j)_{tt}}{2} \Delta t = \frac{u_i^j - u_i^{j-1}}{\Delta t} + \mathcal{O}(\Delta t)$$

When it comes to the local truncation error in space we see that Eq. (12) is the same as Eq. (9), and so the local truncation error is also the same. See the derivation in the previous section, where we find that the biggest error term in space is of order $\mathcal{O}(\Delta x^2)$.

2.3.3 Crank-Nicolson scheme

The derivation of the local truncation error in space and time for the Crank-Nicolson scheme is equivalent to the derivations we did in the two previous sections, but with some more calculations, that we pass in this project. The derivation can be seen in REF... It appears that the biggest error term in Eq. (16) is $\mathcal{O}(\Delta t^2)$, and $\mathcal{O}(\Delta x^2)$ in Eq. (17).

2.4 Stability properties

We can investigate the stability properties of Forward Euler, Backward Euler and Crank-Nicolson scheme by use of the spectral radius of the coefficient matrix for each of the algorithms. The spectral radius is given by

$$\rho(\hat{A}) = \{|\lambda| : \quad \det(\hat{A} - \lambda \mathbb{1})\},$$

which means that the spectral radius is equal to the maximum eigenvalue of the given matrix $\hat{A}$. If the restriction

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

is fulfilled, we know that the system we are looking at can reach a steady state, which must be the case in this project. The restrictions found for the algorithms in this section are listed in Table 1.

2.4.1 Implicit Backward Euler

In 2.2.2 we showed that Eq. (1) can be solved by use of the Backward Euler method, and that the algorithm has the form as we can see in Eq. (15). We then have the coefficient matrix $\hat{A}^{-1} = (\mathbb{1} + \alpha \hat{B})^{-1}$, and want to find its eigenvalues $\lambda_{\hat{A}^{-1}}$, to find the maximum eigenvalue, so we can investigate the algorithms stability properties. Since we have the relation

$$\hat{A}v_k = \lambda_k v_k \quad \Rightarrow \quad \hat{A}^{-1}\hat{A}v_k = \lambda_k \hat{A}^{-1}v_k$$

$$\Rightarrow \quad \mathbb{1}v_k = \lambda_k \hat{A}^{-1}v_k \quad \Rightarrow \quad \frac{1}{\lambda_k} = \hat{A}^{-1}v_k,$$

we have that $\lambda_{\hat{A}^{-1}} = \frac{1}{\lambda_{\hat{A}}}$, so the easiest is to find $\lambda_{\hat{A}}$ first. We have that

$$\lambda_{\hat{A}} = \lambda_{\mathbb{1}} + \alpha\lambda_{\hat{B}}, \tag{21}$$

where $\lambda_{\mathbb{1}} = 1$ are the eigenvalues of the identity matrix and $\lambda_{\hat{B}}$ are the eigenvalues of $\hat{B}$, a matrix which can be seen in Eq. (14). $\hat{B}$ is tridiagonal and Toeplitz, which means that the eigenvalues has a closed form solution (REF...)

$$\lambda_k = a + 2\sqrt{bc}\cos\left(\frac{k\pi}{n+1}\right), \quad \text{for} \quad k = 1, \ldots, n,$$

where a are the elements on the diagonal, b and c are the elements in the diagonal above and below the mid diagonal respectively, and n is the size of the square matrix $\hat{B}$. The eigenvalues of $\hat{B}$ are then given by

$$\lambda_k = 2 + 2\sqrt{(-1)(-1)}\cos\left(\frac{k\pi}{n+1}\right) = 2 + 2\cos(\theta_k) = 2(1 + \cos(\theta_k)), \tag{22}$$

where we have defined $\theta_k = \frac{k\pi}{n+1}$. Now we can find the eigenvalues of $\hat{A}$ by use of Eq. (24), and thus find the eigenvalues of $\hat{A}^{-1}$.

$$\lambda_{\hat{A},k} = 1 + 2\alpha(1 + \cos(\theta_k))$$

We know that $-1 < \cos(\theta_k) < 1$ and $\alpha > 0$, which leads to that we always have $\lambda_{\hat{A},k} > 1$. This leads to that all the eigenvalues of $\hat{A}^{-1}$ are

$$\lambda_{\hat{A}^{-1},k} = \frac{1}{\lambda_{\hat{A},k}} = \frac{1}{1 + 2\alpha(1 + \cos(\theta_k))} < 1,$$

which means that the biggest eigenvalue of $\hat{A}^{-1}$ is less that one. Because of this Eq. (20) is always fulfilled, and we have no restrictions on α for this algorithm.

2.4.2 Explicit Forward Euler

By looking at Eq. (10) it is easy to realize that it can be rewritten to a matrix equation on the same form as Eq. (13), by just a few changes. The matrix equation is

$$U_j = \hat{A}U_{j-1} = (\mathbb{1} - \alpha\hat{B})U_{j-1},$$

where U_j, U_{j-1} and $\hat{B}$ are given in Eq. (14), and $\mathbb{1}$ is the identity matrix. Now we want to find the eigenvalues of $\hat{A}$, so that we can find its biggest eigenvalue, and then see if there is any restrictions to follow when the algorithm is executed. The eigenvalues of $\hat{A}$ is given by

$$\lambda_{\hat{A}} = \lambda_{\mathbb{1}} - \alpha\lambda_{\hat{B}},$$

where $\lambda_{\mathbb{1}} = 1$ are the eigenvalues of the identity matrix and $\lambda_{\hat{B}}$ are the eigenvalues of $\hat{B}$. The eigenvalues of $\hat{B}$ are given in Eq. (22), and by use of this we find the eigenvalues of $\hat{A}$ to be

$$\lambda_{\hat{A},k} = 1 - 2\alpha(1 + \cos(\theta_k))$$

We can see that the smallest possible value of θ_k gives us the biggest eigenvalue of $\hat{A}$. Since $\theta_k = \frac{k\pi}{n+1}$ for $k = 1, \ldots, n$, θ_k can not be zero, but we suppose that n is big so that the minimum value is $\theta_k \simeq 0$. If we insert this into the previous equation and uses the restriction in Eq. (20), we can find the restriction on α, so that we are sure to reach a steady state by use of the Backward Euler method.

$$-1 < (1 - 2\alpha(1 + \cos(0))) < 1 \quad \Rightarrow \quad -1 < (1 - 4\alpha) < 1$$

$$(I): \quad -1 < 1 - 4\alpha \quad \Rightarrow \quad \alpha < \frac{1}{2}$$

$$(II): \quad 1 - 4\alpha < 1 \quad \Rightarrow \quad \alpha > 0$$

Restriction (II) is already fulfilled, because $\alpha = \frac{\Delta t}{\Delta x^2}$, and the step lengths will always be defined as positive. Restriction (I) will however put a restriction on the magnitude of the step lengths of time and space. We get that

$$\frac{\Delta t}{\Delta x^2} < \frac{1}{2} \quad \Rightarrow \quad \Delta t < \frac{\Delta x^2}{2}. \tag{23}$$

2.4.3 Implicit Crank-Nicolson scheme

In 2.2.3 we showed that Eq. (1) can be solved by use of the Crank-Nicolson scheme, and that the algorithm has the form as we can see in Eq. (18). We then have the coefficient matrix $\hat{C} = (\mathbb{1} + \alpha\hat{B})^{-1}(\mathbb{1} - \alpha\hat{B})^{-1}$, and want to find its eigenvalues $\lambda_{\hat{C}}$, to find the maximum eigenvalue, so we can investigate the algorithms stability properties. The eigenvalues are given by

$$\lambda_{\hat{C}} = \lambda_{\hat{A}^{-1}}\lambda_{\hat{A}}, \tag{24}$$

where

$$\lambda_{\hat{A}^{-1}} = \frac{1}{1 + 2\alpha(1 + \cos(\theta_k))} \quad \text{and} \quad \lambda_{\hat{A}} = 1 - 2\alpha(1 + \cos(\theta_k))$$

are found in 2.4.1 and 2.4.2 respectively. We then have the eigenvalues

$$\lambda_{\hat{C}} = \frac{1 - 2\alpha(1 + \cos(\theta_k))}{1 + 2\alpha(1 + \cos(\theta_k))} = \frac{1 - \mu_k}{1 + \mu_k}.$$

To fulfill the restriction in Eq. (20), the maximum eigenvalue have to fulfill

$$-1 < \frac{1 - \mu_k}{1 + \mu_k} < 1$$

$$(I): \quad -1 < \frac{1 - \mu_k}{1 + \mu_k} \quad \Rightarrow \quad -2 - \mu_k < 2 - \mu_k \quad \Rightarrow \quad -2 < 2$$

$$(II): \quad \frac{1 - \mu_k}{1 + \mu_k} < 1 \quad \Rightarrow \quad 2 - \mu_k < 2 + \mu_k \quad \Rightarrow \quad \mu_k > 0$$

We can see that (I) is always fulfilled, but we have to take a closer look at (II). $\mu_k = 2\alpha(1 + \cos(\theta_k))$ and $\theta_k = \frac{k\pi}{n+1}$ for $k = 1, \ldots, n$. If we suppose that n is big, we see

that the smallest possible value of θ_k is $\theta_k \simeq 0$ and that its biggest possible value is $\theta_k \simeq \pi$. We insert these boundaries into (II), to find the restrictions of the Clark-Nicolson scheme.

$$(i) : 2\alpha(1 + \cos(0)) > 0 \quad \Rightarrow \quad 4\alpha > 0$$

$$(ii) : 2\alpha(1 + \cos(\pi)) > 0 \quad \Rightarrow \quad 4\alpha > 0 \quad \Rightarrow \quad 2\alpha(1 - 1) > 0$$

We always have that $\alpha = \frac{\Delta t}{\Delta x^2} > 0$, because the step lengths always are defined to be positive. Then it is easy to see that (i) always is fulfilled. The expression in (ii) indicates that $0 > 0$, but this is never going to be the case since θ_k never is going to reach the value π exactly, it will just approach π, so we will always have that $\cos(\theta_k) < 1$. This means that (ii) always is fulfilled as well, and we can conclude that the Crank-Nicolson scheme do not have any restrictions, which means that the system always will reach a steady state.

3 Method

3.1 Simulation of diffusion of neurotransmitters

Neurotransmitters are information carrying molecules that are used in transport of signals between neurons in the brain. They are transported from a presynaptic cell to a postsynaptic cell, across a synaptic cleft, which is the small space that separates the cell membranes of the two cells. In Figure 1 the cells and synaptic cleft are illustrated. The transport process of the neurotransmitters in the synaptic cleft is governed by diffusion, and can therefore be described mathematically by the diffusion equation

$$D\nabla^2 u = \frac{\partial u}{\partial t}$$

where u is the concentration of particular neurotransmitters and D is the diffusion constant in this particular synaptic cleft.

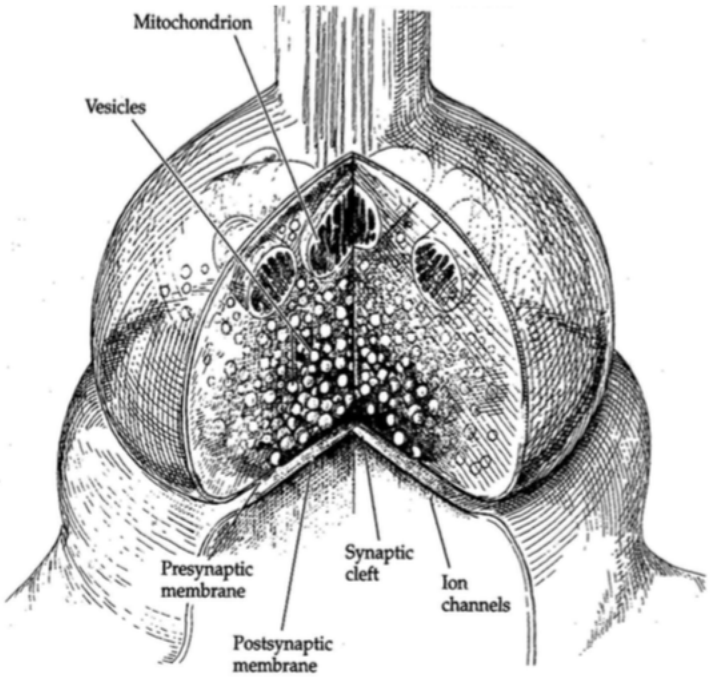

Figure 1: Illustration of how the presynaptic and postsynaptic cell are lying against each other, when the presynaptic cell are sending signals to the postsynaptic cell through the synaptic cleft. The synaptic cleft is the small space between the cell membranes. (From assignment text for project 5.)

To simplify the problem we assume that the presynaptic cell releases the neurotransmitters roughly at the same time along its membrane, and that the whole synaptic cleft has approximately the same width everywhere. Then, because the area of the synaptic cleft is so large compared to its width, we can assume that the neurotransmitter concentration only varies in the direction from the presynaptic cell to the postsynaptic cell. This gives us the opportunity to look at the problem in only one spatial direction. We choose the direction from the presynaptic cell to the postsynaptic cell to be the x-direction, and the diffusion equation reduces to

$$D\frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t}.$$

The simplified system is shown in Figure 2, and we see that the neurotransmitters are released from the presynaptic cell at $x = 0$, and absorbed by the postsynaptic cell at $x = d$.

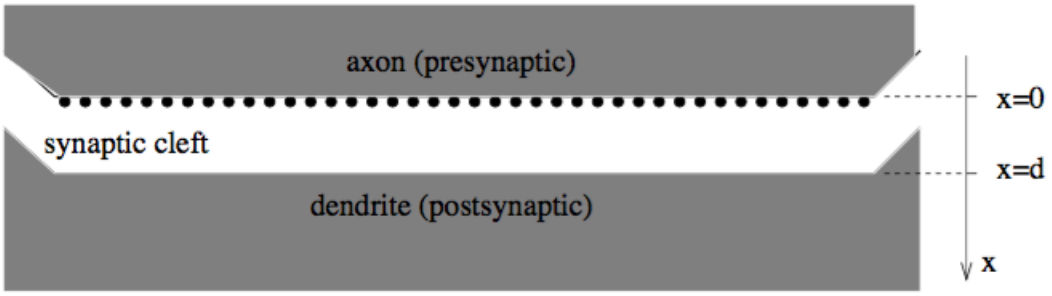

Figure 2: A model of the simplified system of two cells that are interchanging neurotransmitters. The neurotransmitters are sent from the presynaptic cell at $x = 0$, through the synaptic cleft, to the postsynaptic cell at $x = d$. (From assignment text for project 5.)

3.2 Implementation of algorithms

3.2.1 Forward Euler

3.2.2 Backward Euler

3.2.3 Crank-Nicolson

3.2.4 Monte Carlo methods and random walks

3.3 Unit tests and verification of results

To be sure that the results we get from our numerical calculations are correct, it is necessary to include some tests in the code. If there is some of the pre calculations in the algorithm that we know the answer of, or if there is some restrictions on the pre calculations, these are the parts we want to check with tests.

3.3.1 Unit tests in Forward Euler, Backward Euler and the Crank-Nicolson scheme

In 2.4.1, 2.4.2 and 2.4.3 we were looking for restrictions on the step lengths in time and space, from the restriction in Eq. (20). The restriction says that the absolute value of the maximum eigenvalue of the coefficient matrix in the algorithm have to be smaller that one, for the system to reach a steady state. We found that it was only the Forward Euler method that needed a restriction on the step length, namely $\Delta t < \frac{\Delta x^2}{2}$. This restriction was used as a unit test in the algorithm of Forward Euler.

We also had to find a way to test the two other methods. We showed that there was no restrictions on the step lengths for these, and this result came from the fact that the coefficient matrix of their algorithm do not have any eigenvalues with absolute value bigger that or equal to one. We used this fact to check if the restriction in Eq. (20) always was fulfilled. That is, we find the eigenvalues of the coefficient matrices numerically, and make sure that the absolute value of the matrices are less than one. This unit test was also used on the Forward Euler method, to make sure that the restriction we found on the step length was right.

3.3.2 Verification of results

In 2.1 we found the analytical solution of the diffusion equation that we are looking at in this project. All the numerical methods that we are using in this project are supposed to give us the same answers, namely the answers that the analytically calculated solution gives. If we compare the results from the numerical methods with the results from the analytical solution, this is a verification of that the numerical calculated results are correct. We can also use the results from the analytical solution to find the relative error in the results from the numerical methods.

4 Result and discussion

4.1 Truncation error and stability properties

Numerical method	Truncation error	Stability requirements
Forward Euler	$\mathcal{O}(\Delta t)$ and $\mathcal{O}(\Delta x^2)$	$\Delta t \leq \frac{1}{2}\Delta x^2$
Backward Euler	$\mathcal{O}(\Delta t)$ and $\mathcal{O}(\Delta x^2)$	Stable for all Δt and Δx.
Crank-Nicolson	$\mathcal{O}(\Delta t^2)$ and $\mathcal{O}(\Delta x^2)$	Stable for all Δt and Δx.

Table 1: This table lists the order of the local truncation error in space and time for three numerical methods. It also lists the requirements on the step lengths in space and time, to ensure that the numerical methods produce stable results.

4.2 Numerical calculated results

4.3 Relative error in numerical calculated results

Plot where restriction is not met
- Global error in plot. Bigger for each step. - Flattens out in top

4.4 Crank-Nicolson scheme

5 Conclusion

6 Comments

7 References

<https://en.wikipedia.org/wiki/Taylor_series>
<https://en.wikipedia.org/wiki/Fourier_series>
<https://en.wikipedia.org/wiki/Tridiagonal_matrix>
<https://en.wikipedia.org/wiki/Toeplitz_matrix>
Mørken

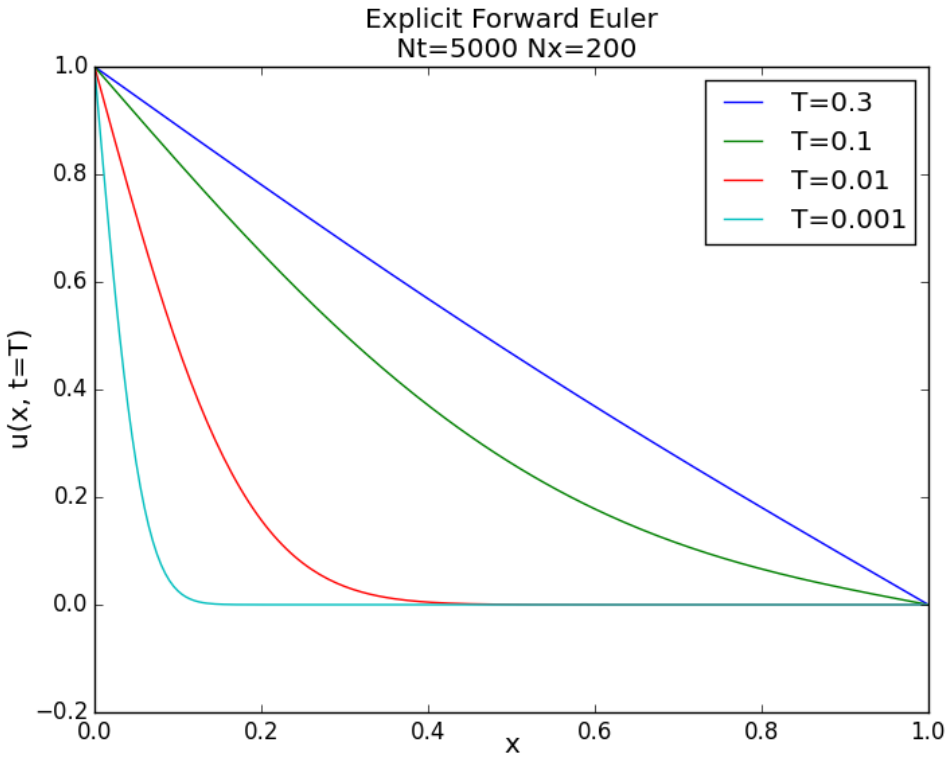

Figure 3: The concentration of neurotransmitters as a function of position in the synapitc cleft for four different points in time. This results is computed by use of the Forward Euler method.

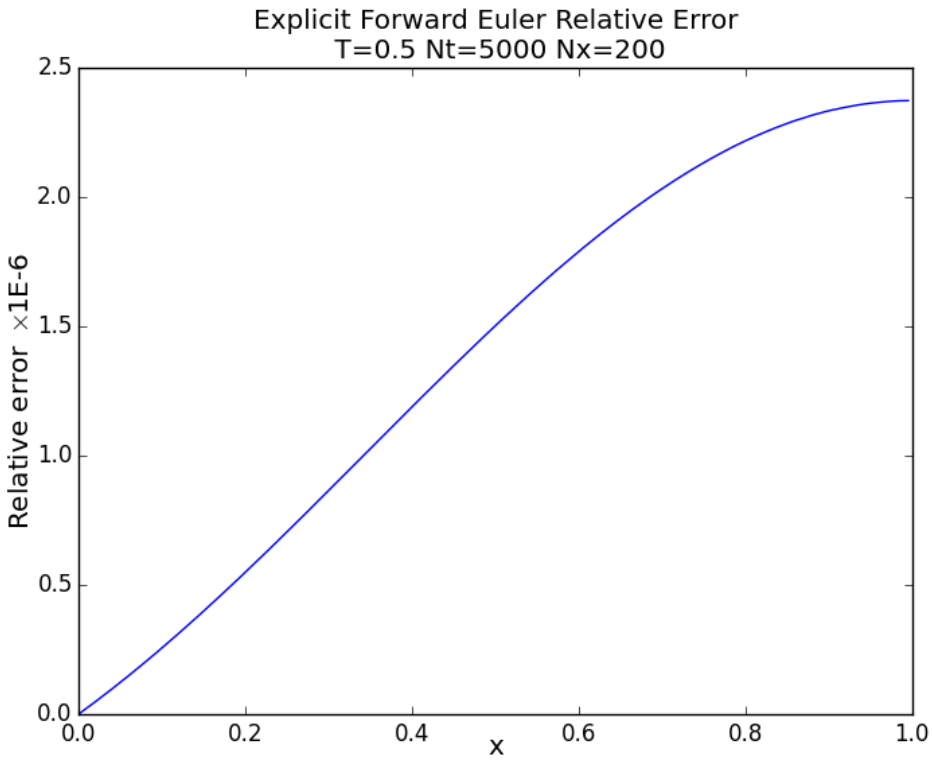

Figure 4: This graph shows how the relative error of data computed by use of the Forward Euler method, changes with position.