FYS4150 - Computational Physics
Project 5:
Diffusion of neurotransmitters in the synaptic cleft

Candidate numbers:

November 17, 2015

Abstract

In this project we study diffusion of neurotransmitters in the brain across the synaptic cleft. In particular, we look at the solution to the diffusion equation for a particular choice of boundary conditions. Our choice of boundary conditions allows for an analytical solution that is found and later compared to the numerical results. The numerical solutions are found by use of three different integration schemes for partial differential equations: the Forward Euler scheme, the Backward Euler scheme and the Crank-Nicolson scheme. Finally, we compare these deterministic schemes to an implementation based on Monte Carlo methods and random walks. We find that ...`blah, results, blah`...

- Github repository link with all source files and benchmark calculations: https://github.com/henrisro/Project5.
- A list of all the code files can be found in the code listing in the end of this document.

1 Introduction

Diffusion of signal molecules is the dominant way of transportation in the brain. In this project we are going to study the solutions to the diffusion equation with restriction to some special boundary conditions. The basic process is illustrated in figure 1. In (1) the vesicles inside the axon approach the presynaptic membrane and merge with it in (2). The vesicle contains neurotransmitters and release them across the synaptic cleft (of with d) in (3). If we denote the concentration of neurotransmitters at distance x and time t from the presynaptic membrane by $u(x,t)$, the dynamics can be described with the diffusion equation:

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

where $\nabla^2 = \frac{\partial^2}{\partial x^2}$ in the one dimensional case. This equation will be the subject of investigations with restrictions to the interval $x \in [0,d]$ for $d = 1$ and diffusion constant equal to unity $D = 1$. We will apply the boundary conditions

$$u(0,t) = 1 \ \ \forall t > 0 \quad \text{and} \quad u(d,t) = 0 \ \ \forall t > 0, \tag{2}$$

corresponding to a constant release of neurotransmitters at the presynaptic membrane ($u(0,t)$) and a constant absorption of transmitters at the postsynaptic membrane (at $u(d,t)$) at all times. The initial conditions will be taken to be:

$$u(x,t) = \begin{cases} 1 & \text{if } x = 0 \\ 0 & \text{if } x \in (0,d] \end{cases} \tag{3}$$

This basically means that all the neurotransmitters are located at $x = 0$ initially when the vesicles are opened.

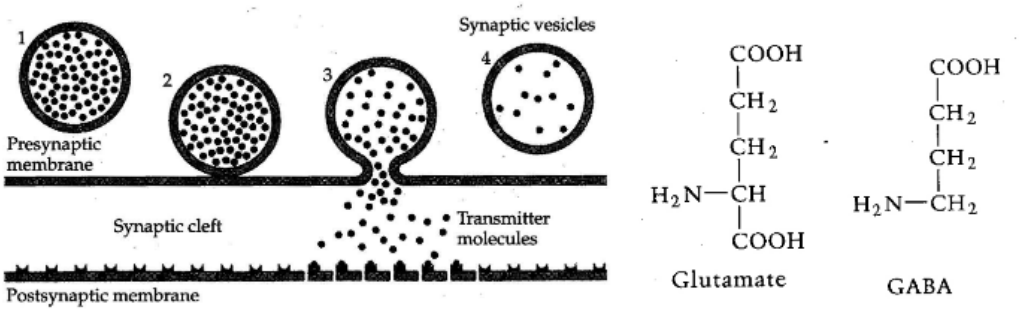

Figure 1: Left: Illustration of the release of vesicle from an axon. The neurotransmitters (two chemical structures are shown to the right) travel diffusively across the synaptic cleft and are absorbed in the postsynaptic membrane at a distance d away from the release point. The figure is taken from the project description [2], but originates from Thompson: "The Brain", Worth Publ., 2000.

The solutions will be studied by implementation of three different deterministic schemes in addition to a Monte Carlo based method. We will look in some detail at the Forward Euler scheme, the Backward Euler scheme and the Crank-Nicholson scheme for partial differential equations (PDEs). The methods are compared in terms of precision and stability.

2 Theory

In this section we derive the analytical solution to the diffusion problem outlined above. The closed form expression of the solution, although it will take the form of an infinite sum, will be valuable in terms of judging the numerical results later.

2.1 Analytical solution

To solve the one dimensional diffusion equation,

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}, \tag{4}$$

with the boundary conditions in (2) and initial conditions in (3), we apply the standard *separation of variables* ansatz:

$$u(x,t) = X(x)T(t) + u_s(x) \tag{5}$$

We here separate out the steady-state solution $u_s(x) = 1 - x$, which trivially satisfies equation (4). When inserted into equation (4) we arrive at:

$$\frac{1}{X}\frac{d^2X}{dx} = \frac{1}{T}\frac{dT}{dt} = -k^2 \tag{6}$$

The last equality (i.e. both sides equal to a constant) follows since the left hand side is independent of t and the right hand side independent of x. Looking first at the $X(x)$ equation, we realize that it is simply the harmonic oscillator equation with solution $X(x) = a\cos(kx) + b\sin(kx)$. The boundary condition in equation (2) translates to (since we must have $X(0) = X(1) = 0$ after adding the steady-state solution) $a = 0$ and $\sin(k) = 0 \Rightarrow k_n = n\pi$ for $n \in \mathbb{N}$. We thus have:

$$X_n(x) = b_n \sin(n\pi) \tag{7}$$

The equation for $T(t)$ is a separable first order differential equation and have solutions:

$$T_n(x) = \exp(-(n\pi)^2 t) \tag{8}$$

The general solution is then a sum of (possibly all) the modes $X_n(x)T_n(t)$:

$$u(x,t) = 1 - x + \sum_{n=1}^{\infty} b_n \sin(n\pi x) e^{-(n\pi)^2 t} \tag{9}$$

The final step is to determine the coefficients b_n such that $u(x,0)$ fits the initial conditions in equation (3). This can be rewritten as

$$\sum_{n=1}^{\infty} b_n \sin(n\pi x) = u_0(x) = \begin{cases} 0 & \text{if } x = 0 \\ x - 1 & \text{if } x \in (0,1] \end{cases} \tag{10}$$

This is just as the Fourier series of the odd extension of the function $u_0(x)$ on the interval $[0,1]$. The coefficients are determined by Fourier's trick (see for instance [1]):

$$b_n = 2\int_0^1 dx\ (x-1)\sin(n\pi x) = -\frac{2}{n\pi} \tag{11}$$

We hence have arrived at the final closed form solution:

$$\boxed{u(x,t) = 1 - x - \frac{2}{\pi}\sum_{n=1}^{\infty} \frac{\sin(n\pi x)}{n} e^{-(n\pi)^2 t}} \tag{12}$$

3 Method / Algorithm

This section is devoted to the discussion of the three deterministic schemes for PDEs mentioned earlier. We derive the algorithms and investigate their stability properties expressed in terms of Δx and Δt. First some general nomenclature. We discretize the interval $[0, 1]$ with a step length determined by some integer n:

$$\Delta x = \frac{1}{n+1} \tag{13}$$

such that $x \mapsto x_i = i\Delta x$ for $i \in \{0, 1, \ldots, n+1\}$. A corresponding discretization in time is given by Δt (we will later discuss how this should be chosen depending on Δx, i.e. n). This means that $t \mapsto t_j = j\Delta t$ for $j \in \mathbb{N}$. The following discussion will be made in the context of the diffusion equation, here written compactly as $u_{xx} = u_t$ (the subscripts refer to derivatives). After discretization we use the notation $u(x, t) \mapsto u(x_i, t_j) \equiv u_{i,j}$. The following definition will also appear frequently in what follows:

$$\alpha \equiv \frac{\Delta t}{\Delta x^2} \tag{14}$$

3.1 Forward Euler scheme

In this scheme we use the two simplest possible approximations for the derivatives u_{xx} and u_t:

$$u_{xx} \approx \frac{u(x+\Delta x, t) - 2u(x,t) + u(x-\Delta x, t)}{\Delta x^2} \mapsto \frac{1}{\Delta x^2}(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) \tag{15}$$

with truncation error of the order $\mathcal{O}(\Delta x^2)$ and

$$u_t \approx \frac{u(x, t+\Delta t) - u(x,t)}{\Delta t} \mapsto \frac{1}{\Delta t}(u_{i,j+1} - u_{i,j}) \tag{16}$$

with truncation error of the order $\mathcal{O}(\Delta t)$. This leads directly to an explicit scheme when equating the above expressions:

$$u_{xx} = u_t \Rightarrow \tag{17}$$

$$\frac{1}{\Delta x^2}(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) = \frac{1}{\Delta t}(u_{i,j+1} - u_{i,j}) \Rightarrow \tag{18}$$

$$u_{i,j+1} = \alpha u_{i+1,j} + (1 - 2\alpha)u_{i,j} + \alpha u_{i-1,j} \tag{19}$$

This scheme is explicit (recursive) in the sense that it gives a straight forward recipe of how to find the solution at the next time step given the solution at the previous one. It should be initialized with the initial condition, $u_{i,0} = u_0(x_i)$ as given in equation (10). In terms of a matrix equation we can write:

$$\boldsymbol{U}_{j+1} = A\boldsymbol{U}_j \tag{20}$$

$$\begin{bmatrix} u_{1,j+1} \\ u_{2,j+1} \\ \vdots \\ u_{n,j+1} \end{bmatrix} = \begin{bmatrix} 1-2\alpha & \alpha & 0 & \cdots & 0 & 0 \\ \alpha & 1-2\alpha & \alpha & \cdots & 0 & 0 \\ & \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & \alpha & 1-2\alpha \end{bmatrix} \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{n,j} \end{bmatrix} \tag{21}$$

with the initial condition (in our case) $\boldsymbol{U}_0 = (u_0(0), u_0(\Delta x), \ldots, u_0(1))^T = (1, 0, \ldots, 0)^T$.

3.2 Backward Euler scheme

The Backward Euler scheme (just like for ordinary differential equations) exploits a time derivative approximation with a function evaluation backwards in time and the same approximation for u_{xx} as in equation (15). We (re)state both approximations to have a clear argument:

$$u_{xx} \approx \frac{u(x+\Delta x, t) - 2u(x,t) + u(x-\Delta x, t)}{\Delta x^2} \mapsto \frac{1}{\Delta x^2}(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) \tag{22}$$

and

$$u_t \approx \frac{u(x,t) - u(x, t-\Delta t)}{\Delta t} \mapsto \frac{1}{\Delta t}(u_{i,j} - u_{i,j-1}) \tag{23}$$

with truncation errors of the orders $\mathcal{O}(\Delta x^2)$ and $\mathcal{O}(\Delta t)$ respectively. Equating these leaves us with:

$$u_{xx} = u_t \Rightarrow \tag{24}$$

$$\frac{1}{\Delta x^2}(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) = \frac{1}{\Delta t}(u_{i,j} - u_{i,j-1}) \Rightarrow \tag{25}$$

$$u_{i,j-1} = -\alpha u_{i+1,j} + (1+2\alpha)u_{i,j} - \alpha u_{i-1,j} \tag{26}$$

This recursion relation backwards in time does, as compared to the Forward Euler scheme, not give us the solution for the next time step in terms of the previous one. It instead takes the following form when written in matrix notation:

$$B\boldsymbol{U}_j = \boldsymbol{U}_{j-1} \tag{27}$$

$$\begin{bmatrix} 1+2\alpha & -\alpha & 0 & \cdots & 0 & 0 \\ -\alpha & 1+2\alpha & -\alpha & \cdots & 0 & 0 \\ & \vdots & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & -\alpha & 1+2\alpha \end{bmatrix} \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ \vdots \\ u_{n,j} \end{bmatrix} = \begin{bmatrix} u_{1,j-1} \\ u_{2,j-1} \\ \vdots \\ u_{n,j-1} \end{bmatrix} \tag{28}$$

This asks for, at each time step j, nothing else than a set of linear equations to be solved. Phrased differently, a matrix inversion (meaning we must find B^{-1} with B tridiagonal as

above) would give us the solution at each time step straight away: $\boldsymbol{U}_j = B^{-1}\boldsymbol{U}_{j-1}$. However, a matrix-vector multiplication at each time step is computationally consuming as it costs $\mathcal{O}(n^2)$ FLOPS and the matrix inversion even worse with its $\mathcal{O}(n^3)$ FLOPS. From Project 1, however, we know that solving such equations for a tridiagonal matrix can be done by an algorithm that takes only $\mathcal{O}(n)$ FLOPS; far superior when the dimension becomes large.

3.3 Crank-Nicolson scheme

3.4 Monte Carlo methods and random walks

4 Results

5 Discussion

6 Conclusion

7 Appendix

References

[1] M. L. Boas. *Mathematical Methods in the Physical Sciences.* Kaye Pace, 2006.

[2] M. Hjort-Jensen. Computational Physics, Lecture Notes Fall 2015. Department of Physics, University of Oslo, 2015.

Code listing