FYS4150 - Computational Physics

PROSJEKT 5

Ei litta titt på pre- og postsynaptisk membran

Kandidat 72

Dato: 11. desember 2015

Sammendrag

I dette prosjektet løser vi diffusjonslikninga med forover og bakover Euler, Crank-Nicolson og ved Monte Carlo-simulering. Metodene gir alle til en viss grad korrekt svar, den enkleste metoden å implementere er MC-metoden. Vi studerer også trunkeringsfeil for de forskjellige metodene og ser om de stemmer overens med resultatene, noe de ikke gjør her. En diskusjon av feilene blir også gjort.

Kode for prosjektet er tilgjengelig på GitHub-domenet: https://github.com/jonvegards/FYS4150/tree/master/project5, mens kode for plotting *etc.* er lagt i https://github.com/jonvegards/FYS4150/tree/master/build-project5

Introduksjon

Overgangen mellom to nerveceller kalles en synapse. For at det skal kunne overføres signaler fra den ene cella til den andre så må "noe" diffundere gjennom synapsen, dette "noe" er signalmolekyler som kalles nevrotransmittere. Disse molekylene kommer i en blobb som vil koble seg på den presynaptiske membranen, og så slippe molekylene fri slik at de kan diffundere over den synaptiske kløfta og komme fram til den postsynaptiske membranen der de vil bli tatt i mot av reseptorer som bringer signalet videre i cella.

For å modellere dette kan vi bruke diffusjonslikninga i én dimensjon,

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

Hvor vi da har antatt visse ting om systemet vårt: den synaptiske kløfta er jevn og like bred overalt, og at konsentrasjonen av nevrotransmittere varierer kun på langs i bevegelsesretninga.

Vi skal løse denne likninga både analytisk og numerisk slik at vi kan teste programmet på høvelig vis og sammenlikne de ulike metodene. De tre numeriske metodene som testes ut er ekplisitt forover-Euler, implisitt bakover-Euler, og Crank-Nicholson-metoden. Resultatene fra disse metodene vil testes for forskjellige parametre og vi vil finne ut hvilken som gir best resultat.

Systemet beskrevet ovenfor kan også simuleres med Monte Carlo-metoder og tilfeldig gange. Vi vil bruke en algoritme som baserer seg på at nevrotransmitterne kan hoppe fram og tilbake som virrevandrere og så sammenlikne resultatene fra denne metoden med de fra diffusjonslikningssimuleringa.

Teori

Å løse systemet analytisk er essensielt for å kunne sjekke at våre numeriske resultater stemmer. Vi setter diffusjonskonstanten D lik 1 slik at vi får en dimensjonsløs likning. Diffusjonslikninga er en separabel likning, slik at vi kan skrive den

$$\mathcal{T}(t)\frac{\partial^2 \mathcal{X}(x)}{\partial x^2} = \mathcal{X}(x)\frac{\partial \mathcal{T}(t)}{\partial t}$$
$$\Rightarrow \frac{1}{\mathcal{X}(x)}\frac{\partial^2 \mathcal{X}(x)}{\partial x^2} = -\lambda^2 \quad \frac{1}{\mathcal{T}(t)}\frac{\partial \mathcal{T}(t)}{\partial t} = -\lambda^2,$$

hvor λ er en konstant vi snart skal finne. Det viser seg imdlertid at det lønner seg å skrive løsninga som

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

hvor $u_s(x)$ er likevektstilstanden løsninga vår skal konvergere mot. Første leddet i løsninga vil gå mot null ettersom tida øker, så vi står igjen med likevektsleddet som er $u_s(x) = 1 - x$. Vi finner så $\mathcal{X}(x)$,

$$\begin{aligned} \frac{\partial^2 \mathcal{X}(x)}{\partial x^2} &= -\lambda^2 \mathcal{X}(x) \\ \Rightarrow \mathcal{X}(x) &= A\sin(\lambda x) + B\cos(\lambda x). \end{aligned}$$

Tidsdelen av løsninga finner vi ved å løse,

$$\begin{aligned} \frac{\partial \mathcal{T}(t)}{\partial t} &= -\lambda^2 \mathcal{T}(t) \\ \Rightarrow \mathcal{T}(t) &= Ce^{-\lambda^2 t}. \end{aligned}$$

Dette gir oss

$$u(x,t) = A\left(\sin(\lambda x) + B\cos(\lambda x)\right)e^{-\lambda^2 t} + 1 - x,$$

Initialbetingelsene er $u(x = d = 1, \text{alle } t) = 0$, $u(x = 0, t > 0) = u_0 = 1$ som gir oss

$$u(x,t) = A_n \sin\left(\frac{\pi x}{d}\right) e^{-\lambda^2 t} + 1 - x.$$

Denne løsninga har mange moduser, slik at vi må skrive den som en sum av alle modene,

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

Det gjenstår å finne koeffisientene A_n, den finner vi fra initialbetingelsen $u(x,0) = 0$,

$$0 = 1 - x + \sum_{n=1}^{\infty} A_n \sin\left(\frac{\pi n x}{d}\right),$$

vi ser at dette minner oss om Fourieranalyse og at vi derfor kan finne A_n som en Fourierkoeffisientene til en funksjon $f(x)$,

$$A_n = \frac{2}{d}\int_0^d f(x)\sin\left(\frac{n\pi x}{d}\right)\,\mathrm{d}x\,.$$

Dette gir oss at $f(x) = x - 1$ og vi får integralet

$$\begin{aligned}
A_n &= 2\int_0^1 (x-1)\sin(n\pi x)\ \mathrm{d}x \\
&= 2\int_0^1 x\sin(n\pi x)\ \mathrm{d}x - 2\int_0^1 \sin(n\pi x)\ \mathrm{d}x \\
&= 2\left(\frac{-\cos(n\pi x)x}{n\pi}\Big|_{x=0}^1 + \int_0^1 \frac{\cos(n\pi x)}{n\pi}\,\mathrm{d}x\right) - \frac{4}{\pi n} \\
&= \frac{2}{n\pi} - \frac{4}{\pi n} \\
&= -\frac{2}{\pi n}.
\end{aligned}$$

Vår endelige analytiske løsning blir da,

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

De tre metodene Som allerede nevnt skal vi løse diffusjonslikninga numerisk ved hjelp av tre forskjellige metoder, vi ser først på den eksplisitte forover-Euler-metoden.

$$u_t \approx \frac{u(x_i, t_j + \Delta t) - u(x_i, t_j)}{\Delta t} = \frac{u_{i,j+1} - u_{i,j}}{\Delta t}$$

og

$$u_{xx} \approx \frac{u(x_i + \Delta x, t_j) - 2u(x_i, t_j) + u(x_i - \Delta x, t_j)}{\Delta x^2} = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2},$$

hvor vi har diskretisert tid og rom, $i, j = 0, 1, \ldots$ Vi husker hvordan diffusjonslikninga ser ut og innser at vi kan skrive

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

hvor $\alpha = \Delta t/h^2$. Vi kan så definere en vektor som inneholder alle funksjonsverdiene til u,

$$V_j^T = [u_{0,j}, \ldots, u_{n,j}].$$

Denne vektoren trenger noen venner, nemlig matriser som sier hvordan vi skal behandle funksjonsverdiene. Vi setter $\hat{A} \equiv \mathbb{1} + \alpha\hat{B}$, hvor vi har,

$$\begin{pmatrix}
2 & -1 & 0 & \ldots & 0 \\
-1 & 2 & -1 & 0 & \vdots \\
0 & \ddots & \ddots & \ddots & -1 \\
\vdots & \ldots & \ldots & -1 & 2
\end{pmatrix}. \tag{2}$$

Vi kan da skrive diffusjonslikninga på matriseform: $\hat{V}_{j+1} = \hat{A}\hat{V}_j$, akkurat som i prosjekt 1. Dette kan vi enkelt implementere i programmet vårt som en dobbel for-løkke som går over tid og rom. Vi vil da ikke definere hele matriser, siden

$\hat{B}$ er tridiagonal så kan vi operere med vektorer og gange riktige elementer sammen slik at vi slipper å sløse FLOPS ved å gange med null mange ganger.

Vi kan så se på implisitt bakover-Euler-metoden. Til forskjell fra den eksplisitte metoden over, så bruker vi her det forrige tidssteget til å finne det nye. Vi har da,

$$u_t \approx \frac{u(x_i,t_j)-u(x_i,t_j-\Delta t)}{\Delta t} = \frac{u_{i,j}-u_{i,j-1}}{\Delta t},$$

mens u_{xx} er den samme som i stad. Dette skriver vi så sammen til å være

$$u_{i,j-1} = u_{i,j} - \alpha\left(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}\right). \tag{3}$$

På matriseform får vi da $\hat{C} = \mathbb{1} - \alpha\hat{B}$ og dermed $\hat{C}\hat{V}_j = V_{j-1} \Rightarrow \hat{V}_{j+1} = \hat{C}^{-1}\hat{V}_j$. Vi kan altså her bruke vår egenproduserte tridiagonale regnemaskin fra prosjekt 1.

Siste metode ut er den implisitte Crank-Nicolson-metoden. Vi har et tidssentrert skjema, $(x, t+\Delta t/2)$

$$u_t \approx \frac{u(x_i,t_j)-u(x_i,t_j-\Delta t)}{\Delta t} = \frac{u_{i,j}-u_{i,j-1}}{\Delta t}. \tag{4}$$

Den andreordens deriverte ser nå slik ut,

$$\begin{aligned} u_{xx} &\approx \frac{1}{2}\left(\frac{u(x_i+\Delta x,t_j)-2u(x_i,t_j)+u(x_i-\Delta x,t_j)}{\Delta x^2}+\right.\\ &\left.\frac{u(x_i+\Delta x,t_j-\Delta t)-2u(x_i,t_j-\Delta t)+u(x_i-\Delta x,t_j-\Delta t)}{\Delta x^2}\right)\\ &= \frac{1}{2}\left(\frac{u_{i+1,j}-2u_{i,j}+u_{i-1,j}}{\Delta x^2}+\frac{u_{i+1,j-1}-2u_{i,j-1}+u_{i-1,j-1}}{\Delta x^2}\right). \end{aligned} \tag{5}$$

Vi setter dette så sammen og får,

$$\begin{aligned} 2u_{i,j} - 2u_{i,j-1} &= \alpha\left(u_{i+1,j} - 2u_{i,j} + u_{i-1,j}\right) + \alpha\left(u_{i+1,j-1} - 2u_{i,j-1} + u_{i-1,j-1}\right)\\ (2-2\alpha)u_{i,j-1} + \alpha u_{i+1,j-1} + \alpha u_{i-1,j-1} &= (2+2\alpha)u_{i,j} - \alpha u_{i+1,j} - \alpha u_{i-1,j}, \end{aligned} \tag{6}$$

Vi kjenner igjen høyre- og venstresidene som det vi fikk fra de henholdvis eksplisitte og implisitte Euler-metodene, slik at vi kan skrive

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

hvor faktoren 2 kommer fra den halve vi ganget u_{xx} med. For å løse dette skriver vi

$$\hat{V}_j = (2\mathbb{1} + \alpha\hat{B})^{-1}(2\mathbb{1} - \alpha\hat{B})\hat{V}_{j-1}.$$

Vi har nå det meste klart for å implementere disse metodene i programmet vårt. Men før vi gjør det så er det lurt å se litt nærmere på hvilke verdier av Δt og Δx som gir oss stabile løsninger. Spektralradien til en matrise er definert som

$$\rho(\hat{A}) = \max\left\{|\lambda| : \det(\hat{A} - \lambda\mathbb{1}) = 0\right\},$$

altså absoluttverdien til den største egenverdien. Dette kan tolkes som radien til den minste sirkelen i $\mathbb{C}^2$, med senter i origo, som omslutter alle egenverdiene til matrisa $\hat{A}$. Hvis vi har $\rho(\hat{A}) < 1$, så vil løsningen vår *alltid* konvergere mot

en stabil løsning. Hvis matrisa $\hat{A}$ er positiv definitt så er dette kravet alltid tilfredsstilt. Så for våre tre metoder og de tilhørende tre matriselikninger, så kan vi sjekke om dette kravet er tilfredsstilt.

Først eksplisitt forover-Euler, da har vi $\hat{A} = \mathbb{1} + \alpha\hat{B}$. Egenverdiene til identitetsmatrisa er 1, mens vi for $\hat{B}$ bør faktisk regne det ut så vi er helt sikre (selv om vi vet at en reell tridiagonal matrise er positiv definitt). Vi kan skrive

$$b_{i,j} = 2\delta_{i,j} - \delta_{i+1,j} - \delta_{i-1,j}.$$

Egenverdilikninga er gitt som

$$\hat{B}\hat{v} = \lambda\hat{v}$$

$$\begin{aligned}
\Rightarrow (\hat{B}\hat{v})_i &= \lambda_i \hat{v}_i \\
&= \sum_{j=1}^{n} (2\delta_{i,j} - \delta_{i+1,j} - \delta_{i-1,j}) v_j \\
&= 2v_i - v_{i+1} - v_{i-1} = \lambda_i v_i.
\end{aligned}$$

Vi kan så velge en basis, $\sin(\beta\theta)$, å uttrykke egenvektorene i slik at vi får

$$2\sin(i\theta) - \sin(i+1\theta) - \sin(i-1\theta) = \lambda_i \sin(i\theta).$$

Bruker vi så identiteten $\sin(x+y) + \sin(x-y) = 2\cos(x)\sin(y)$ så kan vi forenkle uttrykket over til,

$$\begin{aligned}
2(1-\cos(i\theta))\sin(i\theta) &= \lambda_i \sin(i\theta) \\
\lambda_i &= 2(1-\cos(i\theta)).
\end{aligned}$$

Egenverdiene for $\hat{A}$ blir da $\Gamma = 1 - 2\alpha(1-\cos(i\theta))$

$$\Rightarrow -1 < 1 - 2\alpha(1-\cos(i\theta)) < 1 \Rightarrow \alpha < \frac{1}{2},$$

hvilket impliserer at vi har kravet,

$$\frac{\Delta t}{\Delta x^2} < \frac{1}{2},$$

som betyr at vi ikke kan velge tids- og lengdesteg som vi vil og fremdeles få et konvergerende resultat.

Ser vi derimot på det implisitte skjemaet så skal vi finne egenverdiene til $\hat{C} = \mathbb{1} + \alpha\hat{B}$, som ved samme utregning som over gir,

$$\lambda_i = 1 + 2\alpha(1-\cos(i\theta)).$$

Å nei, vi har en egenverdi som alltid er større enn 1! Frykt ei, det implisitte skjemaet finner neste tidssteg ved $V_j = \hat{C}^{-1}V_{j-1}$, vi må altså ha at $(1/\lambda_i) < 1$, som er tilfredsstilt her for alle verdier av Δx og Δt. Hurra!

For Crank-Nicolson blir egenverdiene $\gamma = 1 - 2\alpha(1-\cos(i\theta))/(1+2\alpha(1-\cos(i\theta)))$, som gir oss,

$$\begin{aligned}
1 &> \frac{1-2\alpha(1-\cos(i\theta))}{1+2\alpha(1-\cos(i\theta))} \\
1-2\alpha(1-\cos(i\theta)) &> 1+2\alpha(1-\cos(i\theta)) \\
0 &< 2\alpha[(1-\cos(i\theta)) + (1+\cos(i\theta))] \\
&< 4\alpha,
\end{aligned}$$

som åpenbart alltid er tilfredsstilt.

Trunkeringsfeil Vi finner trunkeringsfeilen ved å sette inn Taylorutvikle faktorene i uttrykkene som tilnærmer de tids- og posisjonsderiverte for de forskjellige metodene. Vi gjentar først hvordan de tilnærmede deriverte er, deretter Taylorutvikler vi alle de forskjellige leddene og setter så inn i uttrykkene våre.

Eksplisitt metode

$$u_t \approx \frac{u(x,t+\Delta t)-u(x,t)}{\Delta t} \tag{7}$$

$$u_{xx} \approx \frac{u(x+\Delta x,t)-2u(x,t)+u(x-\Delta x,t)}{\Delta x^2}, \tag{8}$$

Implisitt metode Som nevnt tidligere er det kun forskjell på den tidsderiverte, vi bruker her det foregående punktet for å finne den tidsderiverte,

$$u_t \approx \frac{u(x,t)-u(x,t-\Delta t)}{\Delta t}, \tag{9}$$

Crank-Nicolson bruker (4) og (5).

Vi har nå en del ledd å Taylorutvikle, heldigvis kan vi se på [2] og finne en del av utviklingene der. Vi ser på utviklingene for de eksplisitte og implisitte metodene først, da utvikler vi om $t+\Delta t$ og $x+\Delta x$.

$$u(x+\Delta x,t) = u(x,t) + \frac{\partial u(x,t)}{\partial x}\Delta x + \frac{\partial^2 u(x,t)}{2\partial x^2}\Delta x^2 + \mathcal{O}(\Delta x^3), \tag{10}$$

$$u(x-\Delta x,t) = u(x,t) - \frac{\partial u(x,t)}{\partial x}\Delta x + \frac{\partial^2 u(x,t)}{2\partial x^2}\Delta x^2 + \mathcal{O}(\Delta x^3), \tag{11}$$

$$u(x,t+\Delta t) = u(x,t) + \frac{\partial u(x,t)}{\partial t}\Delta t + \mathcal{O}(\Delta t^2), \tag{12}$$

$$u(x,t-\Delta t) = u(x,t) - \frac{\partial u(x,t)}{\partial t}\Delta t + \mathcal{O}(\Delta t^2). \tag{13}$$

Når vi skal utvikle for Crank Nicolson så må vi huske på å ta hensyn til det halve tidssteget, altså å utviklet om $t' = t + \Delta t/2$.

$$\begin{aligned} u(x+\Delta x,t+\Delta t) = u(x,t') &+ \frac{\partial u(x,t')}{\partial x}\Delta x + \frac{\partial u(x,t')}{\partial t}\frac{\Delta t}{2} + \frac{\partial^2 u(x,t')}{2\partial x^2}\Delta x^2 \\ &+ \frac{\partial^2 u(x,t')}{2\partial t^2}\frac{\Delta t^2}{4} + \frac{\partial^2 u(x,t')}{\partial x \partial t}\frac{\Delta t}{2}\Delta x + \mathcal{O}(\Delta t^3) \end{aligned} \tag{14}$$

$$\begin{aligned} u(x-\Delta x,t+\Delta t) = u(x,t') &- \frac{\partial u(x,t')}{\partial x}\Delta x + \frac{\partial u(x,t')}{\partial t}\frac{\Delta t}{2} + \frac{\partial^2 u(x,t')}{2\partial x^2}\Delta x^2 \\ &+ \frac{\partial^2 u(x,t')}{2\partial t^2}\frac{\Delta t^2}{4} - \frac{\partial^2 u(x,t')}{\partial x \partial t}\frac{\Delta t}{2}\Delta x + \mathcal{O}(\Delta t^3) \end{aligned} \tag{15}$$

$$\begin{aligned} u(x+\Delta x,t) = u(x,t') &+ \frac{\partial u(x,t')}{\partial x}\Delta x - \frac{\partial u(x,t')}{\partial t}\frac{\Delta t}{2} + \frac{\partial^2 u(x,t')}{2\partial x^2}\Delta x^2 \\ &+ \frac{\partial^2 u(x,t')}{2\partial t^2}\frac{\Delta t^2}{4} - \frac{\partial^2 u(x,t')}{\partial x \partial t}\frac{\Delta t}{2}\Delta x + \mathcal{O}(\Delta t^3) \end{aligned} \tag{16}$$

$$\begin{aligned} u(x-\Delta x,t) = u(x,t') &- \frac{\partial u(x,t')}{\partial x}\Delta x - \frac{\partial u(x,t')}{\partial t}\frac{\Delta t}{2} + \frac{\partial^2 u(x,t')}{2\partial x^2}\Delta x^2 \\ &+ \frac{\partial^2 u(x,t')}{2\partial t^2}\frac{\Delta t^2}{4} + \frac{\partial^2 u(x,t')}{\partial x \partial t}\frac{\Delta t}{2}\Delta x + \mathcal{O}(\Delta t^3) \end{aligned} \tag{17}$$

$$u(x,t+\Delta t) = u(x,t') + \frac{\partial u(x,t')}{\partial t}\frac{\Delta t}{2} + \frac{\partial^2 u(x,t')}{2\partial t^2}\frac{\Delta t^2}{4} + \mathcal{O}(\Delta t^3) \tag{18}$$

$$u(x,t) = u(x,t') - \frac{\partial u(x,t')}{\partial t}\frac{\Delta t}{2} + \frac{\partial^2 u(x,t')}{2\partial t^2}\Delta t^2 + \mathcal{O}(\Delta t^3) \tag{19}$$

Vi er klare til å sette inn! Vi tar den posisjonsderiverte først siden den er lik for eksplisitt og implisitt.

$$\begin{aligned} u_{xx} \approx & \left(u(x,t) + \frac{\partial u(x,t)}{\partial x}\Delta x + \frac{\partial^2 u(x,t)}{2\partial x^2}\Delta x^2 + \mathcal{O}(\Delta x^3) - 2u(x,t) \right. \\ & \left. + u(x,t) - \frac{\partial u(x,t)}{\partial x}\Delta x + \frac{\partial^2 u(x,t)}{2\partial x^2}\Delta x^2 + \mathcal{O}(\Delta x^3) \right) \frac{1}{\Delta x^2} \\ = & \frac{\partial^2 u(x,t)}{\partial x^2} + \mathcal{O}(\Delta x^2). \end{aligned} \tag{20}$$

Vi fortsetter med den tidsderiverte for eksplisitt metode,

$$\begin{aligned} u_t \approx & \left(u(x,t) + \frac{\partial u(x,t)}{\partial t}\Delta t - u(x,t) + \mathcal{O}(\Delta t^2) \right) \frac{1}{\Delta t} \\ = & \frac{\partial u(x,t)}{\partial t} + \mathcal{O}(\Delta t). \end{aligned} \tag{21}$$

For en implisitte metoden får vi

$$\begin{aligned} u_t \approx & \left(u(x,t) - u(x,t) + \frac{\partial u(x,t)}{\partial t}\Delta t + \mathcal{O}(\Delta t^2) \right) \frac{1}{\Delta t} \\ = & \frac{\partial u(x,t)}{\partial t} + \mathcal{O}(\Delta t). \end{aligned} \tag{22}$$

Til slutt skal vi sette inn for Crank-Nicolson. Vi setter inn i (4),

$$\begin{aligned} \frac{u_{i,j+1} - u_{i,j}}{\Delta t} = & \left(u(x,t') + \frac{\partial u(x,t')}{\partial t}\frac{\Delta t}{2} + \frac{\partial^2 u(x,t')}{2\partial t^2}\Delta t^2 - u(x,t') + \frac{\partial u(x,t')}{\partial t}\frac{\Delta t}{2} \right. \\ & \left. - \frac{\partial^2 u(x,t')}{2\partial t^2}\Delta t^2 \right) \frac{1}{\Delta t} + \mathcal{O}(\Delta t^2) \\ = & \frac{\partial u(x,t')}{\partial t} + \mathcal{O}(\Delta t^2). \end{aligned}$$

Vi setter så inn i (5)

$$\begin{aligned} & \frac{1}{2}\left(\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + \frac{u_{i+1,j-1} - 2u_{i,j-1} + u_{i-1,j-1}}{\Delta x^2} \right) \\ & = \frac{1}{2\Delta x^2}\left[u(x,t') + \frac{\partial u(x,t')}{\partial x}\Delta x + \frac{\partial u(x,t')}{\partial t}\frac{\Delta t}{2} + \frac{\partial^2 u(x,t')}{2\partial x^2}\Delta x^2 \right. \\ & + \frac{\partial^2 u(x,t')}{2\partial t^2}\frac{\Delta t^2}{4} + \frac{\partial^2 u(x,t')}{\partial x \partial t}\frac{\Delta t}{2}\Delta x - 2\left(u(x,t') + \frac{\partial u(x,t')}{\partial t}\frac{\Delta t}{2} + \frac{\partial^2 u(x,t')}{2\partial t^2}\frac{\Delta t^2}{4} \right) \\ & + u(x,t') - \frac{\partial u(x,t')}{\partial x}\Delta x + \frac{\partial u(x,t')}{\partial t}\frac{\Delta t}{2} + \frac{\partial^2 u(x,t')}{2\partial x^2}\Delta x^2 \\ & + \frac{\partial^2 u(x,t')}{2\partial t^2}\frac{\Delta t^2}{4} - \frac{\partial^2 u(x,t')}{\partial x \partial t}\frac{\Delta t}{2}\Delta x + u(x,t') + \frac{\partial u(x,t')}{\partial x}\Delta x - \frac{\partial u(x,t')}{\partial t}\frac{\Delta t}{2} \\ & + \frac{\partial^2 u(x,t')}{2\partial x^2}\Delta x^2 + \frac{\partial^2 u(x,t')}{2\partial t^2}\frac{\Delta t^2}{4} - \frac{\partial^2 u(x,t')}{\partial x \partial t}\frac{\Delta t}{2}\Delta x - 2\left(u(x,t') - \frac{\partial u(x,t')}{\partial t}\frac{\Delta t}{2} \right. \\ & \left. + \frac{\partial^2 u(x,t')}{2\partial t^2}\frac{\Delta t^2}{4} \right) + u(x,t') - \frac{\partial u(x,t')}{\partial x}\Delta x - \frac{\partial u(x,t')}{\partial t}\frac{\Delta t}{2} + \frac{\partial^2 u(x,t')}{2\partial x^2}\Delta x^2 \\ & \left. + \frac{\partial^2 u(x,t')}{2\partial t^2}\frac{\Delta t^2}{4} + \frac{\partial^2 u(x,t')}{\partial x \partial t}\frac{\Delta t}{2}\Delta x \right] + \mathcal{O}(\Delta x^2) \\ & = \frac{\partial^2 u(x,t')}{2\partial x^2} + \mathcal{O}(\Delta x^2). \end{aligned}$$

Oppsummert kan vi sette resultatene våre i tabell (1)

Metode	Trunkering	Stabilitetskrav
Eksplisitt	$\mathcal{O}(\Delta x^2)$ og $\mathcal{O}(\Delta t)$	$\Delta t \leq \Delta x^2/2$
Implisitt	$\mathcal{O}(\Delta x^2)$ og $\mathcal{O}(\Delta t)$	$\forall \quad \Delta t$ og Δx^2
Crank-Nicolson	$\mathcal{O}(\Delta x^2)$ og $\mathcal{O}(\Delta t^2)$	$\forall \quad \Delta t$ og Δx^2

Tabell 1: *Vi ser her trunkeringsfeil og stabilitetskrav for de tre forskjellige metodene. Vi noterer oss at Crank-Nicolson ser ut til å være den beste metoden.*

Metode

Eksplisitt, implisitt og Crank-Nicolson I (1) og (9) så har vi i grunnen alt vi trenger for å kunne implementere alle tre metodene. Den eksplisitte metoden er en regn matrisemultiplikasjon som trenger én for-løkke, mens for den implisitte metoden kan vi bruke den tridiagonale løseren vi brukte i prosjekt 1, vi har dog reimplementert den for å luke ut insekter som satt i fra det prosjektet. Crank-Nicolson-metoden bruker de to foregående metodene i kombinasjon med halve tidssteg. Rent algoritmisk kan vi skrive det som følger.

- Generer starttilstand etc.
- Regn ut matrisemultiplikasjonen med den eksplisitte metoden.
- Bruk resultatet fra forrige punkt til å sende inn i den tridiagonale løseren.
- Gjenta for så mange tidssteg du vil.

Monte Carlo-metoden Vi kan også simulere diffusjon ved hjelp av Monte Carlo-simuleringer. Prinsippet er ganske enkelt, vi lar hver partikkel som skal over den synaptiske kløfta være representert av en virrevandrer og sjekker hvor langt den kommer på et visst antall tidssteg. I dette prosjektet skal vi følge [1] ved å bruke en steglengde på $\sqrt{2D\Delta t}$, hvor D er diffusjonskonstanten som vi har satt til å være 1. Vi skal også sjekke om det har noe å si om vi endrer steglengden til å være gaussisk fordelt om 1 med standardavvik 0.

Algoritmen vår vil basere seg på en virrevandring, slik som i [2],

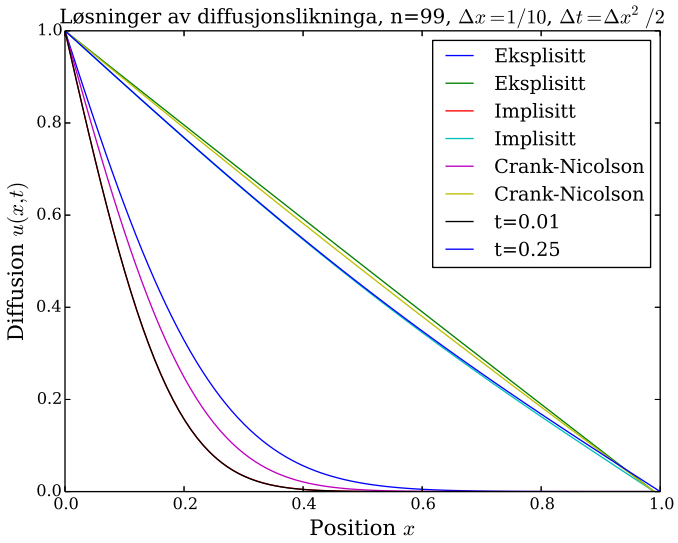
- Generer initialbetingelser
- Løkke over tid og vandrere
- Gjør en MC-test for å bestemme om vandreren skal gå fram- eller bakover
- Oppdater posisjonsverdier
- Gjenta for alle partikler for et antall tidssteg
- Regn ut sannsynlighetsdistribusjonen for hvor ofte hver posisjon mellom synapsekantene ble besøkt av en vandrer

Siden vi har diskretisert intervallet mellom 0 og 1 i 100 båser, så må vi sjekke om partikkelen klarer å hoppe langt nok når vi har en steglengde som varierer. Dette løser vi ved å sette inn noen flere if-tester i den indre for-løkka, ellers så er algoritmen den samme.

Enhetstesting I dette prosjektet blir hele programmet enhetstestet på det viset at vi sammenlikner den numeriske løsninga med den analytiske.

Resultat og diskusjon

Vi kan endelig se på resultatene! Overraskende nok ser vi i fig. (1) at Crank-Nicolson ikke er den beste metoden i dette tilfellet. Ut i fra tab. (1) så burde CN være den beste metoden. Grunner for at vi ikke fikk det samme resultatet her kan være at vårt valg av tidssteg for Crank-Nicolson ikke stemmer overens med måten vi har implementert algoritmen på, siden kurven legger seg over den analytiske løsningen kan det tenkes at tidsstegene er for lange. Men det er også pussig at den implisitte metoden legger seg så nøyaktig oppå den analytiske løsninga. En annen feilkilde kan være at tidsstegene mellom den analytiske og de numeriske løsningene ikke stemmer helt overens siden vi har brukt Python til å plotte den analytiske løsninga, da kan det ha blitt en liten forskjell underveis. Men igjen, siden CN-metoden legger seg så godt inntil den eksplisitte metoden er det grunn til å tro at det er tidsstegene som er litt for store.

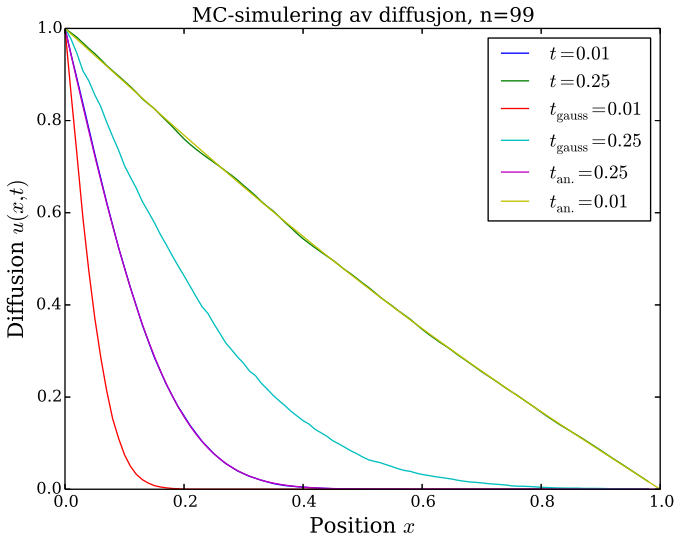


Figur 1: *Det ser ut til at den implisitte metoden fungerer best. Det er litt overraskende, siden Crank-Nicolson burde "tatt det beste" fra to verdener, i tillegg til at den har et tidssentrert skjema. En mulig feil her kan være at tidsstegene brukt i den analytiske løsninga og den numeriske ikke stemmer overens.*

Videre har vi den MC-simulerte løsninga. Her har vi foretatt en virrevandring med 10^7 partikler over like lang tid som for de andre metodene. Vi ser i fig. (2) de to MC-metodene plottet sammen med den analytiske løsninga. Det første vi noterer oss er at når steglengdene er gaussiske så må vi la simuleringa gå for en lenger tid siden partiklene ikke alltid klarer å hoppe langt nok for å komme i

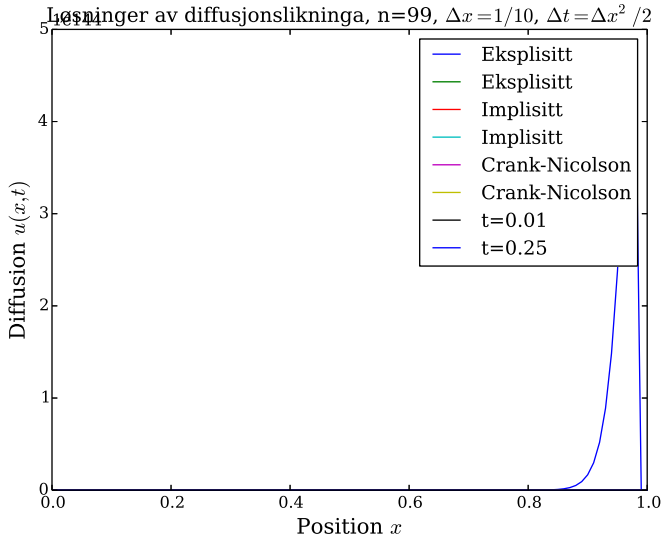
neste bås. Det andre vi noterer oss er at den numeriske løsninga legger seg pent oppå den analytiske løsninga, igjen kan vi tenke på fig. (1) og vurdere hvorvidt steglengden til CN-metoden er for lang eller ei, siden MC-løsninga legger seg så fint oppå den analytiske kan vi nok konkludere med at CN-metoden går for litt langt.

Men siden vi så at MC-simuleringa må kjøre lenger ved en gaussisk varierende steglengde kan vi også prøve å ta hensyn til at den varierer, *i.e.* akkumelere opp hoppene, slik at to hopp kan være nok til at vi kan oppdatere posisjonsvektorene i programmet.



Figur 2: *Monte Carlo-simulering der steglengden er fast og gaussisk varierende. Vi ser at siden partiklene ikke alltid klarer å hoppe langt nok at simuleringa må kjøre lenger for å oppnå samme resultat som ved fast steglengde.*

Som en siste figur viser vi hva som skjer vi setter $\Delta t = 3h^2/2$, som ikke tilfredsstiller stabilitetskravet til den eksplisitte metoden.



Figur 3: *Vi ser at stabilitetskravet ikke er noe humbug! Når* $\Delta t = 3h^2/2$ *så divergerer løsninga til ad undas.*

Konklusjon

I dette prosjektet så har vi sett på fire måter å løse samme problem på. Vi har sett at Crank-Nicolson var en litt skuffende metode her, men hadde vi hatt mer tid til rådighet kunne vi dykket dypere i koden og prøvd å rette opp feilen. Vår konklusjon på at den ikke fungerte tilfredsstillende er at steglengden ikke stemmer helt. Videre simulerte vi systemet med Monte Carlo-metoder, det fungerte utmerket med en fast steglengde. En feil som kan ha blitt gjort i dette programmet er at steglengden vi brukte kanskje ikke stemte helt overens med det [1] brukte siden vi skar gjennom og delte opp den synaptiske kløfta i 100 båser, men antakeligvis ville ikke det hatt mye å si.

Til slutt kan vi nevne at når steglengden varierer gaussisk så burde vi tatt hensyn til at to små hopp kan være nok til at partikkelen kommer seg til neste bås, da ville denne metoden antakeligvis vært ganske nærme den med fast steglengde. Når vi ikke tar hensyn til det slik vi har det i denne rapporten, så vil det naturligvis ta lenger tid for systemet å nå likevekt når ikke hvert forsøk på å hoppe gjør at partikkelen hopper fram eller tilbake.

Kommentarer til prosjektet

Dette prosjektet kom på et ubeleilig tidspunkt og det var derfor ikke tid til å rette opp feilene som har blitt kommentert underveis i rapporten. Utenom det så har det vært et bra prosjekt.

Referanser

[1] L. Farnell, W.G. Gibson, "Monte Carlo simulation of diffusion in a spatially nonhomogeneous medium: A biased random walk on an asymmetrical lattice," Journ. of Comp. Phys., **208**, (2005), 253-265, ISSN 0021-9991, http://www.sciencedirect.com/science/article/pii/S0021999105001087

[2] M. H. Jensen, "Random walks, Brownian motion and the Metropolis algorithm," http://compphysics.github.io/ComputationalPhysics1/doc/pub/rw/html/rw.html,

[3] M. H. Jensen, "Partial differential equations," http://compphysics.github.io/ComputationalPhysics1/doc/pub/pde/html/pde.html,