

MULTISCALE MODELING OF DIFFUSION PROCESSES IN DENDRITES AND DENDRITIC SPINES

by

Fredrik Eksaa Pettersen

THESIS

for the degree of

MASTER OF SCIENCE



Faculty of Mathematics and Natural Sciences
University of Oslo

June 2014

Contents

1	Introduction	3
2	Theory	5
2.1	Random walks	6
2.1.1	Random number generator	7
2.2	Backward Euler schemes in 2 or more spatial dimensions	7
2.3	Combining micro and macro scale models	12
2.3.1	The algorithm	12
2.3.2	The conversion between length scales	12
2.3.3	Coupling the models through step length	13
2.3.4	Boundary conditions for the random walk	13
2.4	Potential problems or pitfalls	14
2.5	Other possible coupling methods	16
3	Analysis	19
3.1	The error estimate	20
3.2	Verification techniques	20
3.3	Testing the PDE solver	22
3.3.1	Verification by manufactured solutions	22
3.3.2	Verification by convergence tests	22
3.3.3	Verification of FE scheme by exact numerical solution .	23
3.3.4	Verification of BE scheme by exact numerical solution .	27
3.4	Verification of the RW solver	28
3.5	Verification of the hybrid solver	31
4	Physical application	33
4.1	Physical scope	34
4.2	Implementation	34
4.3	Parameters	36

List of Figures

2.2	Illustration of $C(x)$	7
2.3	Workaround for negative concentrations, illustration	14
2.4	16
3.6	Numerical exact error plots FE	27
3.7	Numerical exact errorplots for BE scheme	28
3.9	Test RW	30
3.10	Error test for BE combined with RW in 1D	31
3.11	Effects of increasing relative size of walk area	32
4.1	Difference between hybrid diffusion solver and dendrite - spine diffusion model	34

Chapter 1

Introduction

The diffusion equation governs a large number of physical processes on many different length scales. Both heat diffusion through a wall and the Brownian motion of dust particles can be described by the diffusion equation. In some diffusion processes a large number of particles will diffuse from a large volume into a small volume in such a way that only very few particles enter the smaller volume. An example of this in the diffusion of the enzyme PKC γ from the body of a brain cell via a dendrite of large volume into dendritic spines. Dendritic spines are the receiving end of a connection between two brain cells, and PKC γ is associated with long term reinforcement of such a connection resulting in learning or associative memory storage. The increased concentration of PKC γ in a dendritic spine is measured to around $5 \frac{\text{nMol}}{\text{L}}$ which translates to 1 – 2 PKC γ enzymes in the spine. Seeing as the diffusion equation is derived from a continuous concentration distribution, using it on the enzymes in the spine seems questionable.

This thesis will look at hybrid diffusion processes where part of the process has very few particles and other parts have enough particles to be described by a continuous diffusion equation. The part with few particles will have a higher resolution in both space and time, and will be modeled by a stochastic process. A large emphasis has been put on verification of all parts of the hybrid model. Both individually and for the combined, hybrid solver to verify the average behavior of the system.

Chapter 2

Theory

2.1 Random walks

A random walker is a point object which after a certain amount of time jumps a fixed step length either right or left with equal probability for jumping either way. In d spatial dimensions the axis on which the jump happens is chosen at random before the direction of the jump is decided.

Say a distribution of random walkers, C is chosen so that all the walkers are placed within an imagined volume, V of arbitrary shape. V is contained by a surface S and the relation is shown in Figure 2.1.

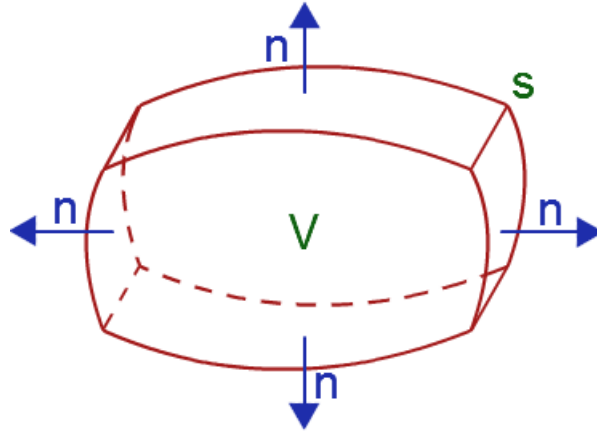


Figure 2.1

As the random walkers start to move some of them will leave the volume V which results in a change in C . The net movement of random walkers at a point is called the flux of walkers and is denoted by the flux vector \mathbf{J} . To find the number of walkers that leave V we integrate the normal component of \mathbf{J} over S . Assuming that no walkers disappear or are created the flux of walkers leaving V must be balanced by the decrease in C , expressed in equation (2.1).

$$\int_V \frac{\partial C}{\partial t} dV = \int_S \mathbf{J} \cdot \mathbf{n} dS \quad (2.1)$$

Through Greens theorem equation (2.1) can be reformulated

$$\int_V \frac{\partial C}{\partial t} dV = \int_V \nabla \cdot \mathbf{J} dV \quad (2.2)$$

The flux \mathbf{J} can be expressed by Fick's first law as the diffusive flux

$$\mathbf{J} = D \nabla C \quad (2.3)$$

where D is the diffusion constant. Since V was chosen as an arbitrary volume equation (2.2) is independent of the integration and the integrals can be dropped. The diffusive flux \mathbf{J} is also inserted to yield the diffusion equation

$$\frac{\partial C}{\partial t} = D \nabla^2 C \quad (2.4)$$

Throughout this thesis C will denote a spatial distribution of random walkers. C is best illustrated in Figure 2.2.

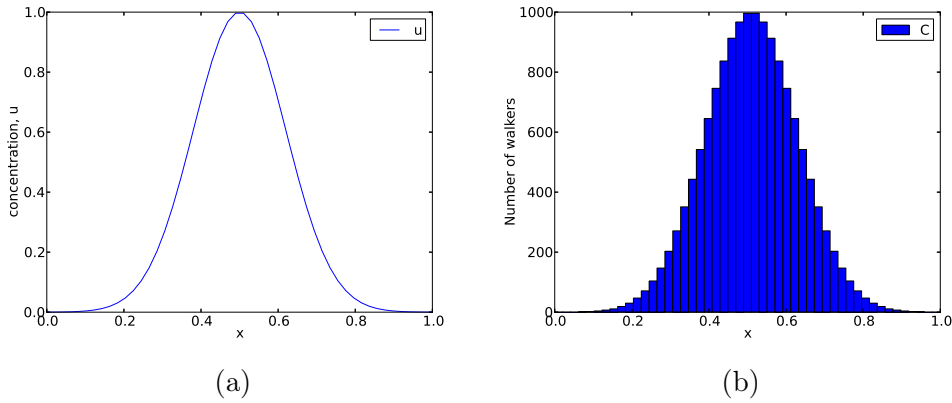


Figure 2.2: Illustration of the difference between $u(x, t)$ in (a) which is the solution to the diffusion equation and $C(x, t)$ in (b) which is the number of random walkers within an area of $\pm \frac{\Delta x}{2}$ around each mesh point.

2.1.1 Random number generator

To produce a random walk one must have random numbers. The random numbers in this thesis are produced by a five seeded xor-shift algorithm copied from George Marsaglia [4]. This generator is chosen because of its very large period ($\sim 10^{48}$) compared to the computational cost.

2.2 Backward Euler schemes in 2 or more spatial dimensions

The reader is assumed to know the basics of solving partial differential equations (PDEs) by finite difference methods, but for the sake of clarity the two schemes used in this thesis are written out.

$$\frac{u^{n+1} - u^n}{\Delta t} = \theta D \frac{\partial^2 u^{n+1}}{\partial x^2} + (1 - \theta) D \frac{\partial^2 u^n}{\partial x^2} \quad (2.5)$$

Having $\theta = 0$ gives the Forward Euler (FE) scheme, and $\theta = 1$ gives the Backward Euler (BE) scheme.

This section will concern a simple, but elegant way of efficiently solving the diffusion equation by the implicit BE scheme in more than one spatial dimension. First, let us look at the BE discretization of the diffusion equation in 1D.

$$u_i^{n+1} = \frac{\Delta t}{2\Delta x^2} ((D_{i+1} + D_i)(u_{i+1}^{n+1} - u_i^{n+1}) - (D_i + D_{i-1})(u_i^{n+1} - u_{i-1}^{n+1})) + u_i^n \quad (2.6)$$

u is here the unknown function which solves the diffusion equation. Note that

$$u_i^n = u(t_0 + n \cdot \Delta t, x_0 + i \cdot \Delta x)$$

and not u to the n 'th power.

Neumann boundary conditions will be used, these are described in (2.7).

$$\frac{\partial u}{\partial n} = 0|_{\text{on boundary}} \quad (2.7)$$

Solving the BE scheme by hand for a very small mesh consisting of 4 mesh points will reveal the structure of the BE scheme.

$$\begin{aligned} u_0^{n+1} &= \frac{\Delta t}{2\Delta x^2} (2(D_0 + D_1)(u_1^{n+1} - u_0^{n+1})) + u_0^n \\ u_1^{n+1} &= \frac{\Delta t}{2\Delta x^2} ((D_2 + D_1)(u_2^{n+1} - u_1^{n+1}) - (D_1 + D_0)(u_1^{n+1} - u_0^{n+1})) + u_1^n \\ u_2^{n+1} &= \frac{\Delta t}{2\Delta x^2} ((D_3 + D_2)(u_3^{n+1} - u_2^{n+1}) - (D_2 + D_1)(u_2^{n+1} - u_1^{n+1})) + u_2^n \\ u_3^{n+1} &= \frac{\Delta t}{2\Delta x^2} (2(D_2 + D_3)(u_3^{n+1} - u_2^{n+1})) + u_3^n \end{aligned}$$

Rearranging this and setting $a = \frac{\Delta t}{2\Delta x^2}$ results in a normal system of linear equations

$$\begin{aligned} u_0^{n+1} (1 + 2a(D_0 + D_1)) - 2au_1^{n+1}(D_1 + D_0) &= u_0^n \\ u_1^{n+1} (1 + a(D_2 + 2D_1 + D_0)) - au_2^{n+1}(D_2 + D_1) - au_0^{n+1}(D_1 + D_0) &= u_1^n \\ u_2^{n+1} (1 + a(D_3 + 2D_2 + D_1)) - au_3^{n+1}(D_3 + D_2) - au_1^{n+1}(D_2 + D_1) &= u_2^n \\ u_3^{n+1} (1 + 2a(D_3 + D_2)) - 2au_2^{n+1}(D_3 + D_2) &= u_3^n \end{aligned}$$

which is arranged as

$$\begin{pmatrix} b_0 & c_0 & 0 & 0 \\ a_1 & b_1 & c_1 & 0 \\ 0 & a_2 & b_2 & c_2 \\ 0 & 0 & a_3 & b_3 \end{pmatrix} \mathbf{u}^{n+1} = \mathbf{u}^n \quad (2.8)$$

The coefficient matrix in (2.8) will be labeled \mathbf{M} throughout the thesis.

$$\mathbf{M}\mathbf{u}^n = \mathbf{u}^{n-1} \quad (2.9)$$

If a linear system like the one in (2.9) has a solution it can always be found by Gaussian elimination [check this](#). However, for a sparse linear system a lot of the calculations in a Gaussian elimination will be done with zeros. A dramatically more efficient way to solve the linear system in (2.9) is by exploiting the fact that it is tridiagonal. In order to be more consistent with the algorithm, (2.9) will be rewritten as

$$\mathbf{M}\mathbf{u} = \mathbf{u}\mathbf{p} \quad (2.10)$$

where $\mathbf{u}\mathbf{p}$ denotes the solution \mathbf{u} at the previous time step. Tridiagonal systems can be solved extremely efficiently by the “tridiag” algorithm listed below.

```
void tridiag(double *u, double *up, int N, double *a,
double *b, double *c){
    double *H = new double[N];
    double *g = new double[N];
    for(int i=0;i<N;i++){
        H[i] = 0;
        g[i] = 0;
    }
    g[0] = up[0]/b[0];
    H[0] = c[0]/b[0];
    for(int i=1; i<N; i++){
        //forward substitution
        H[i] = -c[i]/(b[i] + a[i]*H[i-1]);
        g[i] = (up[i] - a[i]*g[i-1])/(b[i] + a[i]*H[i-1]);
    }
    u[N-1] = g[N-1];
    for(int i=(N-2); i>=0; i--){
        //Backward substitution
        u[i] = g[i] - H[i]*u[i+1];
    }
}
```

In two spatial dimensions the BE discretization gives the scheme shown in equation (2.11)

$$u_{i,j}^n = \underbrace{\frac{-D\Delta t}{\Delta x^2}}_{\alpha} (u_{i+1,j}^{n+1} + u_{i-1,j}^{n+1}) + \underbrace{\left(1 + \frac{2D\Delta t}{\Delta x^2} + \frac{2D\Delta t}{\Delta y^2}\right)}_{\gamma} u_{i,j}^{n+1} - \underbrace{\frac{2D\Delta t}{\Delta y^2}}_{\beta} (u_{i,j+1}^{n+1} + u_{i,j-1}^{n+1}) \quad (2.11)$$

If (2.11) is solved by hand on a 3×3 mesh by the same steps as in 1D it gives the following linear system

$$\begin{pmatrix} \gamma & -2\beta & 0 & -2\alpha & 0 & 0 & 0 & 0 & 0 \\ -\beta & \gamma & -\beta & 0 & -2\alpha & 0 & 0 & 0 & 0 \\ 0 & -2\beta & \gamma & 0 & 0 & -2\alpha & 0 & 0 & 0 \\ -\alpha & 0 & 0 & \gamma & -2\beta & 0 & -\alpha & 0 & 0 \\ 0 & -\alpha & 0 & -\beta & \gamma & -\beta & 0 & -\alpha & 0 \\ 0 & 0 & -\alpha & 0 & -2\beta & \gamma & 0 & 0 & -\alpha \\ 0 & 0 & 0 & -2\alpha & 0 & 0 & \gamma & -2\beta & 0 \\ 0 & 0 & 0 & 0 & -2\alpha & 0 & -\beta & \gamma & -\beta \\ 0 & 0 & 0 & 0 & 0 & -2\alpha & 0 & -2\beta & \gamma \end{pmatrix} \mathbf{u}^{n+1} = \mathbf{u}^n \quad (2.12)$$

The dashed lines confine a total of nine 3×3 matrices which can be used to reformulate the $2n + 1 = 7$ band diagonal system as a block tridiagonal system as shown below.

$$\begin{pmatrix} B_0 & C_0 & 0 \\ A_1 & B_1 & C_1 \\ 0 & A_2 & B_2 \end{pmatrix} \mathbf{u}^{n+1} = \mathbf{u}^n \quad (2.13)$$

All entries in eq. (2.13) are 3×3 matrices. In the general case where the diffusion equation is discretized and solved on a mesh of size $n \times n$ the resulting block tridiagonal system will consist of n^2 matrix entries which all are $n \times n$ matrices.

One should note that the $n \times n$ matrix \mathbf{u} which solves the 2D diffusion equation has be rewritten as a vector \mathbf{u} of size n^2 .

In order to solve this block tridiagonal linear system the "tridiag" solver from before must be generalized to support matrices as entries in \mathbf{M} . Luckily this generalization is trivial and all that is needed is to replace divisions by matrix inverses. An updated *pseudo coded* scheme is listed in (2.14)

There is a forward substitution

$$\begin{aligned}
H_0 &= -B_0^{-1}C_0 \\
\mathbf{g}_0 &= B_0^{-1}\mathbf{u}\mathbf{p}_0 \\
H_i &= -(B_i + A_i H_{i-1})^{-1} C_i \\
\mathbf{g}_i &= (B_i + A_i H_{i-1})^{-1} (\mathbf{u}\mathbf{p}_i - A_i \mathbf{g}_{i-1})
\end{aligned} \tag{2.14}$$

Followed by a backward substitution

$$\begin{aligned}
\mathbf{x}_{n-1} &= \mathbf{g}_{n-1} \\
\mathbf{x}_i &= \mathbf{g}_i + H_i \mathbf{x}_{i+1}
\end{aligned}$$

It is possible to solve the BE scheme in 3D by the block tridiagonal solver as well. The linear system will be $2n^2 + 1$ band diagonal and must first be reduced to a block matrix which is $2n + 1$ band diagonal like the one in eq. (2.12) before it is further reduced to a block tridiagonal form. Matrix entries will be $n^2 \times n^2$ matrices.

Efficiency of the block tridiagonal algorithm

In general the Forward Euler (FE) scheme will solve a discrete PDE in d spatial dimensions with n mesh points in each dimension using

$$\mathcal{O}(n^d)$$

floating point operations (FLOPs). This is the maximum efficiency possible for a PDE in d dimensions (without using some form of symmetry argument) and will be the benchmark for the BE scheme as well.

Implicit schemes like BE result in linear systems which can be solved by multiplying both sides of the equation with the inverse of the matrix in question.

$$\mathbf{M}^{-1}\mathbf{M}\mathbf{u} = \mathbf{M}^{-1}\mathbf{u}\mathbf{p}$$

Inverting a dense matrix demands $\mathcal{O}(n^3)$ FLOPs for an $n \times n$ matrix. By rewriting the previously mentioned $n^2 \times n^2$ band diagonal matrix as a block tridiagonal matrix and using the modified block tridiagonal solver we have reduced the demanded number of FLOPs from $\mathcal{O}(n^6)$ to $n \cdot \mathcal{O}(n^3) = \mathcal{O}(n^4)$. Though an improvement of 2 orders is very good, compared to the FE scheme this is still one order larger. there is still room for improvement, however. As long as the mass matrix is unchanged, which it will be as long as the time step and diffusion constant are constant, the n matrix inversions need only

be done once. In other words, the H_i terms from eq. (2.14) can be stored. The remaining calculations include three matrix vector multiplications, all of which demand n^2 FLOPs, thus further reducing the computational cost to $\mathcal{O}(n^2)$ FLOPs which is the same as the FE scheme.

2.3 Combining micro and macro scale models

2.3.1 The algorithm

After setting an initial condition and diffusion constant the diffusion problem is solved on both the microscopic and macroscopic meshes and combined into a common solution by the following steps.

- The result from previous PDE time step, \mathbf{up} , is converted to a distribution of random walkers and sent to the RW solver
- The RW solver does a predefined number of micro scale time steps which correspond to one PDE time step
- The result from the RW solver is converted back to a concentration and this replaces the PDE solution
- \mathbf{up} is then used as input to calculate the next time step

2.3.2 The conversion between length scales

As the previous section states the result from the last PDE time step is converted to a distribution of random walkers. This is done by specifying a conversion rate denoted Hc (as was done by Plapp and Karma [5]) which is defined in equation (2.15)

$$C_{ij} = Hc \cdot u_{ij} \quad (2.15)$$

As before u_{ij} is the solution to the 2D diffusion equation evaluated at $x_0 + i\Delta x$ and $y_0 + j\Delta y$, and C_{ij} is the number of random walkers located in the rectangle defined by $x_i \pm \frac{\Delta x}{2}$ and $y_j \pm \frac{\Delta y}{2}$. Walkers are given a random position within this rectangle at the beginning of each PDE time step.

Equation (2.15) effectively states that one "unit" of the solution u will directly correspond to Hc random walkers.

2.3.3 Coupling the models through step length

To ensure that the τ time steps performed by the RW solver sums up to one time step for the PDE solver a limitation must be imposed on the RW solver. This limitation can only be imposed on the step length of the random walkers, so the task at hand is to relate the step length to the PDE time step and τ .

Through an Einstein relation the variance in position is coupled to the diffusion constant.

$$\langle \tilde{\Delta x}^2 \rangle = 2Dd\tilde{\Delta t} \quad (2.16)$$

Where $\tilde{\Delta t}$ is the time step for the random walkers which is exactly $\tilde{\Delta t} = \frac{\Delta t}{\tau}$. Similarly $\tilde{\Delta x}$ is the spatial resolution on the micro scale which is also the step length, l , for the random walkers. Rewriting equation (2.16) gives the final expression for the step length

$$l = \sqrt{2dD\frac{\Delta t}{\tau}} \quad (2.17)$$

2.3.4 Boundary conditions for the random walk

Because the RW area must correspond to an area on the PDE mesh which has a finite, constant size the RW model must have boundary conditions imposed on it. In order to make absolutely sure that no walkers disappear and by extension that the amount of concentration is conserved, perfectly reflecting boundaries are chosen for the RW model. Reflecting boundaries correspond to zero flux Neumann boundaries

$$\frac{\partial C}{\partial n} = 0 \quad (2.18)$$

Neumann boundaries might seem unphysical, but the redistribution of walkers at each PDE time step must also to some extent be considered as a boundary condition.

In hindsight, a possibly better choice of boundary conditions for the RW model would have been perfect exchange of flux between the two solvers.

$$\frac{\partial u}{\partial n} = \frac{\partial C}{\partial n} \quad (2.19)$$

Since time does not allow for this to be tested it will be left as a reference to possible future work.

2.4 Potential problems or pitfalls

This section will identify and discuss a few obvious difficulties which might arise in this project. As far as possible solutions or workarounds will be presented, but some problems might not be solvable.

Negative concentration of walkers

Physically a negative concentration does not make sense, but if an initial condition which takes negative values is imposed on the system the software will try to allocate a negative number of walkers. Trying to handle this is more of an oddity than anything else, but a workaround has been found. If the absolute value of the negative concentration is taken as input to the RW solver and the sign at each mesh point is stored while the RW solver advances the system, the resulting solution can be multiplied by the stored sign to give back a negative concentration. The workaround will at least keep the simulation from crashing, but has a fundamental problem which is illustrated in Figure 2.3.

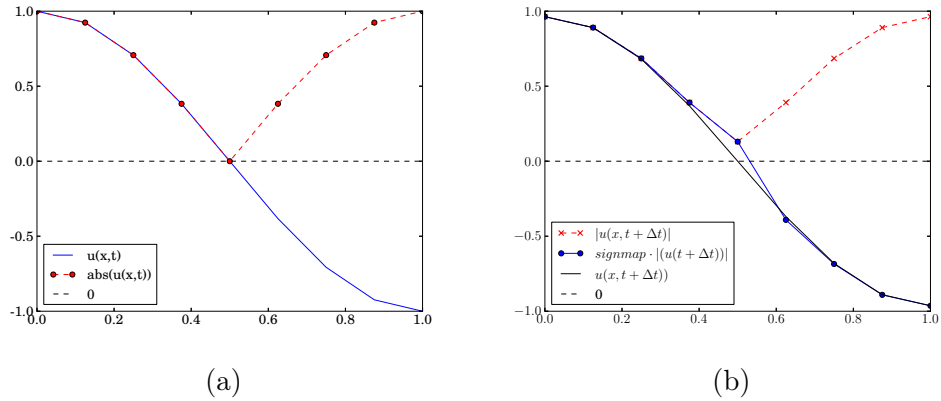


Figure 2.3: An illustration of the proposed workaround for negative concentrations and an illustration of how it performs (b). The fundamental problem is that a diffusion process will even out discontinuities such as the one in (a). The result of this evening out is shown in (b) as an increase in concentration in the middle mesh point which should remain equal to zero.

Smooth solutions

A diffusion process is very effective when it comes to dampening fast fluctuations, and so any solution of the diffusion equation will be smooth. When a stochastic process is introduced, rapid fluctuations from one time-step to the

next might follow. In this case a dilemma arises; on the one hand there is the smoothness of the solution to consider, on the other hand the stochastic term was introduced believing that it adds detail to the model. The approach we tried to use to this was to do some curve-fitting using both of the solutions. A polynomial regression model was implemented in 1D, but regardless of degree and what points were used, the result was a lot worse than just the average of the two solutions. The idea behind this polynomial regression model is described in more detail in section 2.5.

In the end the order of integration was switched so that the PDE solver can reduce the worst of the fluctuations from the RW model. This ensures some smoothness and some fluctuations and is considered a working compromise.

Number of time steps on the random walk level

The reason for introducing a random walk model in the diffusion solver was to represent parts of the mesh at a higher resolution than the rest of the process. This means that the RW solver must take τ time steps for each PDE time step. τ could in principle be any integer, but should be chosen large enough so the central limit theorem is satisfied. This usually means $\tau \geq 50$.

Alternatively τ can be calculated from the assumption that the error term wrt. time arising from the RW model is proportional to the square root of the time step. This assumption comes from the fact that the step length is proportional to the square root of the time step.

$$\epsilon \propto \sqrt{\tilde{\Delta}t} \quad (2.20)$$

In order to make the error of the same size as the error from the PDE model we use

$$\begin{aligned} \tilde{\Delta}t &= \frac{\Delta t}{\tau} \\ \Rightarrow \epsilon &\propto \sqrt{\frac{\Delta t}{\tau}} \\ \mathcal{O}(\Delta t) &\geq \sqrt{\frac{\Delta t}{\tau}} \\ \Rightarrow \tau &\geq \frac{1}{\Delta t} \end{aligned}$$

Note that the error associated with the number of walkers introduced is of much higher significance than the error from the time step.

2.5 Other possible coupling methods

Running the solution from the PDE model through a RW model gives two solutions for a part of the mesh. As it turns out it is sufficient to simply replace the PDE solution with the RW solution in this area, which is nice seeing as it is the simplest possible method. There are however some other ways to do this which have been tested, and found inadequate or simply been disregarded. Three of these are mentioned below.

- Averaging the two solutions
This method is simple and reduces fluctuations by approximately one half (approximately because fluctuations will be present after the first time step). The reason an average was scraped is simply that results just as good were achieved by only replacing the solution.
- Polynomial regression
Explain what polynomial regression is?? In order to smoothen out the solution from the RW model a polynomial regression function was tested. The principle was to look at the solution from the RW model as a set of data points which have some dependency on their position on the PDE mesh. Figure 2.4 shows how this method worked for a slightly exaggerated case.

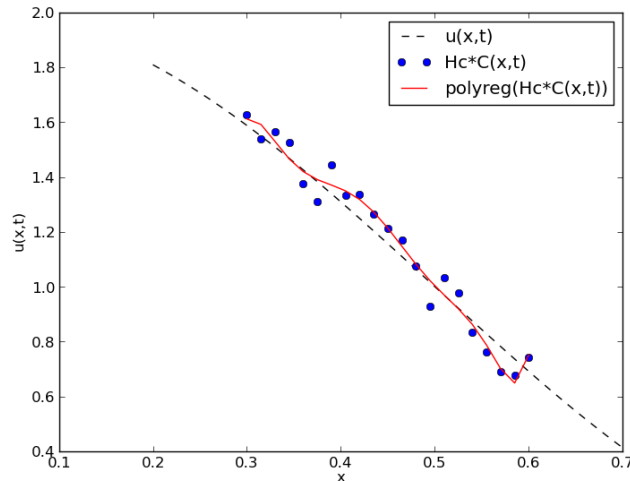


Figure 2.4

This approach turns out to have two problems; first and foremost the

amount of concentration is no longer conserved, secondly the endpoints are usually way off.

- Cubic spline interpolation
Explain what cubic spline is?? Cubic spline interpolation has the same problem as polynomial regression when it comes to conserving energy.

Chapter 3

Analysis

3.1 The error estimate

Solving PDEs numerically will result in errors because derivatives are approximated by finite differences and the PDE is required to be fulfilled only on the mesh points. For the schemes used in this thesis the error, ϵ will follow equation (3.1)

$$\epsilon = C_x \Delta x^2 + C_t \Delta t \quad (3.1)$$

where the coefficients C_x and C_t are unknown. Notice that there is one term arising from the time derivative and one from the spatial derivative and that they are of different order.

The error is measured by comparing the result from a numerical simulation to an exact solution, u_e and taking the norm of the difference. Specifically ϵ is measured by the L2 norm which is defined in equation (3.2)

$$\begin{aligned} \epsilon(t^n) &= \|u(t^n) - u_e(t^n)\|_2 \\ &= \iint \sqrt{(u(t^n, x, y) - u_e(t^n, x, y))^2} dx dy \\ &\approx \sqrt{\Delta x \Delta y \sum_{i=0}^n \sum_{j=0}^n (u(t^n, x_i, y_j) - u_e(t^n, x_i, y_j))^2} \end{aligned} \quad (3.2)$$

ϵ is time dependent because it allows for investigation of the evolution of the error over the course of a simulation. Some of the error tests will require a single number as an error measure. In these cases the norm of $\epsilon(t)$, defined in eq. (3.3) is used.

$$\epsilon = \sqrt{\Delta t \sum_{n=0}^T \epsilon(t^n)^2} \quad (3.3)$$

3.2 Verification techniques

This thesis will focus on three verification techniques which are described below. The aim for all of these techniques is to make sure that the error term follows equation (3.1). Since an incorrect implementation of the spatial derivative will cause the solution to be unstable which is easily noticed through visual inspection, the tests will focus on verifying the time derivative. Isolating the contribution to $\epsilon(t)$ from the time derivative will be necessary and is ensured by setting $\Delta t \gg \Delta x^2$. A time step of this size violates the stability criterion for the FE scheme, and so some of the tests are omitted

for this scheme.

The verification techniques are

- Manufactured solutions

By choosing an adequate initial condition the exact solution to the diffusion equation can be found with relative ease. The chosen solution is

$$u(x, y, t) = e^{-\pi^2 t} \cos(\pi x) \cos(\pi y) + 1 \quad (3.4)$$

The point of these tests is to verify that $\epsilon \sim \Delta t$ and the tests can be done for a time step which fulfills the stability criterion.

- Convergence tests

In the general case the error term is proportional to the time step to some power r when the error from the time derivative is dominant.

$$\epsilon \simeq C_t \Delta t^r \quad (3.5)$$

By comparing the error from two simulations with different time steps the exponent r , called the convergence rate, can be found

$$\begin{aligned} \epsilon_1 &\simeq C_t \Delta t_1^r \\ \epsilon_2 &\simeq C_t \Delta t_2^r \end{aligned}$$

the two expressions are divided

$$\begin{aligned} \frac{\epsilon_1}{\epsilon_2} &\simeq \frac{C_t \Delta t_1^r}{C_t \Delta t_2^r} \\ \log \left(\frac{\epsilon_1}{\epsilon_2} \right) &\simeq r \log \left(\frac{\Delta t_1}{\Delta t_2} \right) \\ r &\simeq \frac{\log(\epsilon_1/\epsilon_2)}{\log(\Delta t_1/\Delta t_2)} \end{aligned}$$

The FE and BE schemes have errors proportional to Δt so the tests are expected to measure $r = 1$.

- Exact numerical solutions

The numerical schemes are actually reformulations of the PDE we are trying to solve as difference equations which have their own exact solutions. These will be called the numerical exact solutions, and they are slightly different from the exact solutions to the PDE. The reason for finding the numerical exact solutions is that the scheme theoretically will represent this solution with no error. In practice there will always be round off errors and other factors, but an error term close to machine precision is expected.

3.3 Testing the PDE solver

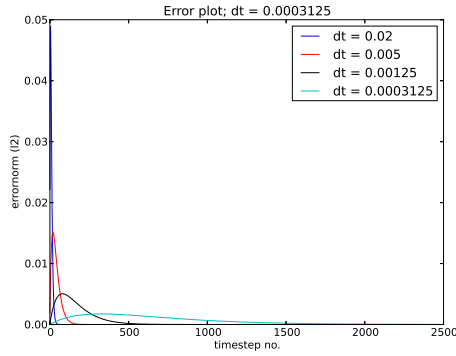
3.3.1 Verification by manufactured solutions

Equation (3.4) solves the diffusion equation, so using $u(x, y, t = 0)$ as the initial condition for a simulation will give us both the numerical solution and the exact solution. An important property of the discretization scheme is that the difference between these solutions is of the same order as the time step.

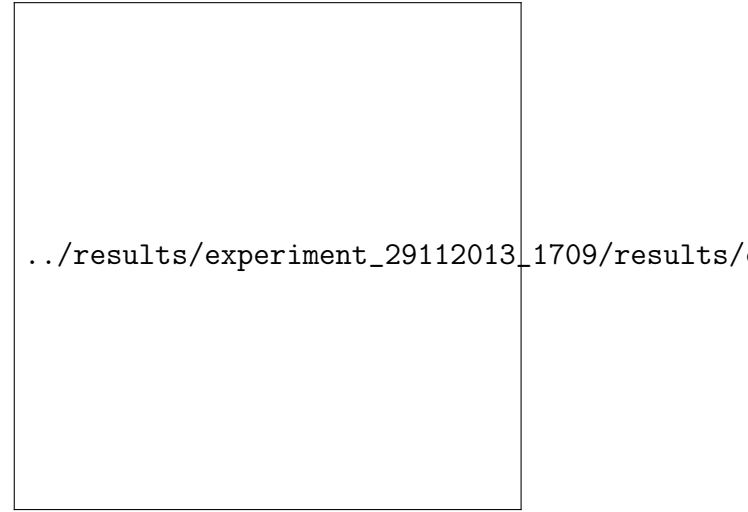
$$\epsilon(t) \sim \mathcal{O}(\Delta t)$$

Figures 3.1 and 3.2 show that for both the FE and BE scheme in both 1D and 2D the error is of the expected magnitude. Another interesting property of the error plots is that the error tends to zero after a large number of time steps. By inserting the limit $t \rightarrow \infty$ in equation (3.4) we observe that the error is expected to tend to zero because the limit value of one can be exactly recreated by both schemes.

$$\lim_{t \rightarrow \infty} e^{-\pi^2 t} \cos(\pi x) \cos(\pi y) + 1 = 1$$



(a)



(b)

Figure 3.1

3.3.2 Verification by convergence tests

The convergence tests are done by isolating the error term from either the time derivative or the spatial derivative, and refining the relevant discretiza-

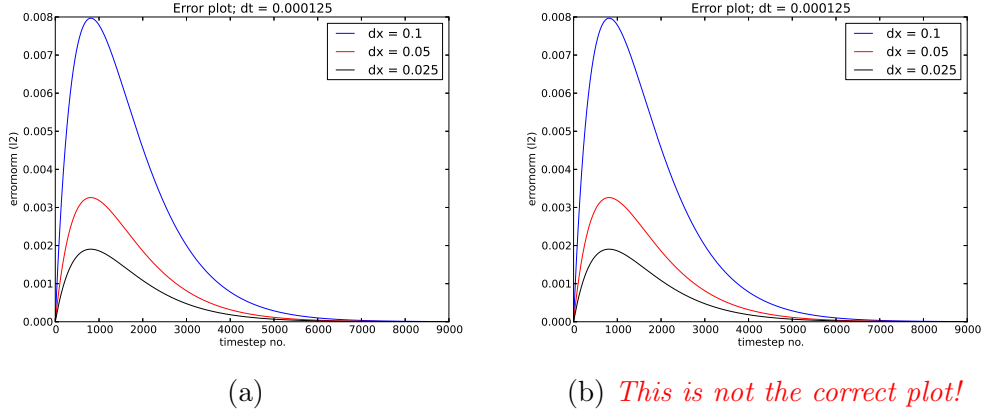


Figure 3.2

tion parameter over several simulations. Figures 3.3 and 3.4 verify that the error associated with the time derivative is of the expected order while figure ?? verifies the error from the spatial derivative for the BE scheme.

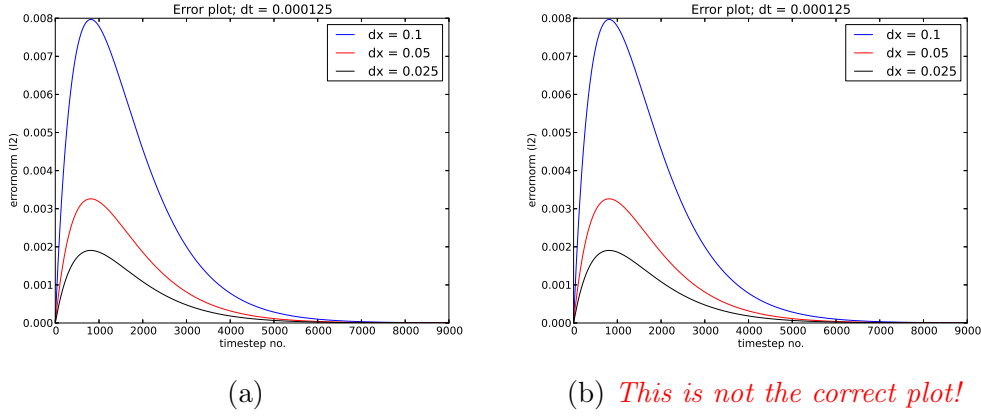
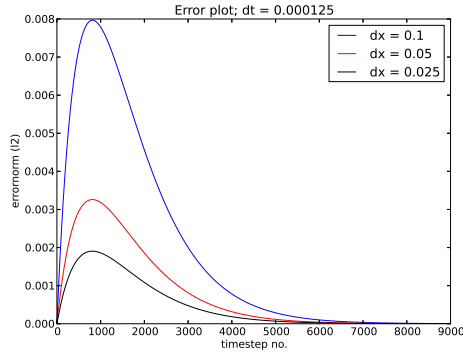


Figure 3.3

3.3.3 Verification of FE scheme by exact numerical solution

Discretization of the diffusion equation by the FE scheme yields the following numerical scheme in 1D

$$u^{n+1} = D\Delta t u_{xx}^n + u^n \quad (3.6)$$



(a)

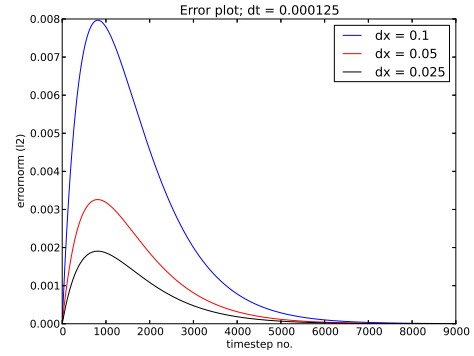
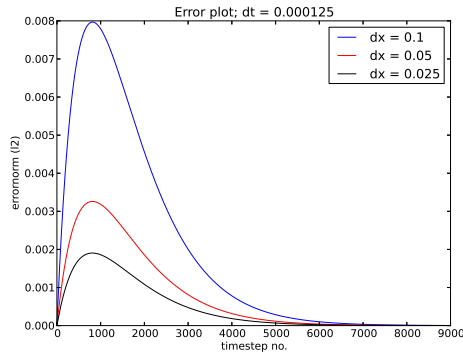
(b) *This is not the correct plot!*

Figure 3.4



(a)

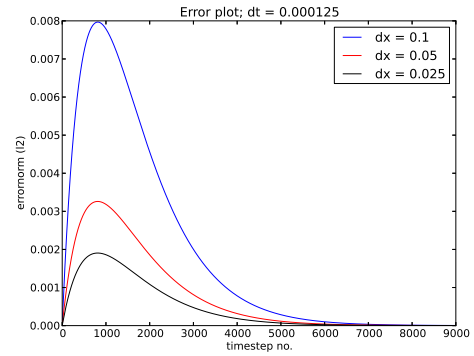
(b) *This is not the correct plot!*

Figure 3.5

where u_{xx} denotes the double derivative of u with respect to x . To illustrate how the equation is solved by the computer, the first four iterations are

written out

$$\begin{aligned}
u^1 &= D\Delta t u_{xx}^0 + u^0 \\
u^2 &= D\Delta t u_{xx}^1 + u^1 \\
&= D\Delta t [D\Delta t u_{4x}^0 + u_{2x}^0] + u^0 \\
&= (D\Delta t)^2 u_{4x}^0 + 2D\Delta t u_{2x}^0 + u^0 \\
u^3 &= D\Delta t u_{xx}^2 + u^2 \\
&= D\Delta t [(D\Delta t)^2 u_{6x}^0 + 2D\Delta t u_{4x}^0 + u_{2x}^0] + (D\Delta t)^2 u_{4x}^0 + 2D\Delta t u_{2x}^0 + u^0 \\
&= (D\Delta t)^3 u_{6x}^0 + 3(D\Delta t)^2 u_{4x}^0 + 3D\Delta t u_{2x}^0 + u^0 \\
u^4 &= D\Delta t u_{xx}^3 + u^3 = \dots \\
&= (D\Delta t)^4 u_{8x}^0 + 4(D\Delta t)^3 u_{6x}^0 + 6(D\Delta t)^2 u_{4x}^0 + 4D\Delta t u_{2x}^0 + u^0
\end{aligned}$$

From the iterations above a pattern emerges for iteration $n + 1$

$$u^{n+1} = \sum_{i=0}^n \binom{n}{i} (D\Delta t)^i u_{2ix}^0 \quad (3.7)$$

Where u^0 is the initial condition

$$u^0 = \cos(\pi x) \quad (3.8)$$

The spatial derivatives are found as

$$\begin{aligned}
u_{xx}^0 &= \frac{1}{\Delta x^2} (\cos(\pi(x + \Delta x)) - 2\cos(\pi x) + \cos(\pi(x - \Delta x))) \\
&= \frac{2}{\Delta x^2} (\cos(\pi\Delta x) - 1) \cos(\pi x) \\
u_{4x}^0 &= [u_{xx}^0]_{xx} \frac{1}{\Delta x^2} \left[\frac{u_{xx}^0}{\cos(\pi x)} (\cos(\pi(x + \Delta x)) - 2\cos(\pi x) + \cos(\pi(x - \Delta x))) \right] \\
&= \frac{4}{\Delta x^2} (\cos(\pi\Delta x) - 1)^2 \cos(\pi x) \\
&\dots
\end{aligned}$$

The pattern continues allowing the final numerical exact solution to be expressed in equation (3.9).

$$u^{n+1} = \sum_{i=0}^n \binom{n}{i} (D\Delta t)^i \frac{2^i}{\Delta x^{2i}} (\cos(\pi\Delta x) - 1)^i \cos(\pi x) \quad (3.9)$$

Although the FE scheme is expected to represent equation (3.9) to machine precision ($\epsilon \approx 10^{-16}$) there are two problems with the solution which will have an effect on the error:

- Δx^{2i} will quickly tend to zero, and the computer will interpret it as zero. This will cause division by zero, which again results in “Not a number” (nan) and ruins the simulation. Testing if $\Delta x^{2i} > 0$ and returning zero if the test fails will fix the problem. The argumentation for ignoring the troublesome terms is given below.
- $\binom{n}{i}$ goes to infinity for large n and i . The computer can only represent numbers up to $\sim 10^{308}$, which limits the number of steps to 170 since $n! > 10^{308}$ for $n > 170$.

As a side note, equation (3.9) illustrates how the stability criterion for the FE scheme comes into place. In the numerical exact solution the exponential which is found in the exact solution to the PDE (eq. 3.4) is replaced by an amplification factor A^n . This amplification factor can be found in equation (3.9) as

$$A^n = \left(\frac{2D\Delta t}{\Delta x^2} \right)^i \quad (3.10)$$

Inserting a time step larger than the stability criterion ($\Delta t \leq \frac{\Delta x^2}{2D}$) will make the amplification factor A larger than one which in turn will make the solution blow up.

The stability criterion also illustrates why the terms where

$$\frac{1}{\Delta x^{2i}} \rightarrow \infty$$

can be dropped. By the stability criterion, the time step will cancel out Δx^2 , and the result will be a number smaller than 1 raised to a rather large power, i , resulting in a number comparable to zero.

The results from comparing a 1D simulation to the numerical exact is shown in Figure 3.6a. As expected the error is larger than machine precision by at most two orders of magnitude because of accumulating error terms from the dropped terms in eq. (3.9).

Using the same method as in the 1D case, a numerical exact solution can be found to the 2D FE scheme.

$$u^{n+1} = \sum_{i=0}^n \binom{n}{i} (D\Delta t)^i \left[2^{i-1} \cos(\pi x) \cos(\pi y) \left(\frac{(\cos(\pi \Delta x))^i}{\Delta x^{2i}} + \frac{(\cos(\pi \Delta y))^i}{\Delta y^{2i}} \right) \right] \quad (3.11)$$

The same problems as in the 1D case will apply to equation (3.11) with the same solutions. Figure 3.6b shows how the 2D simulation compares to the numerical exact solution. As was the case in 1D the error is larger than

machine precision, but much smaller than Δt suggesting that the scheme is implemented correctly.

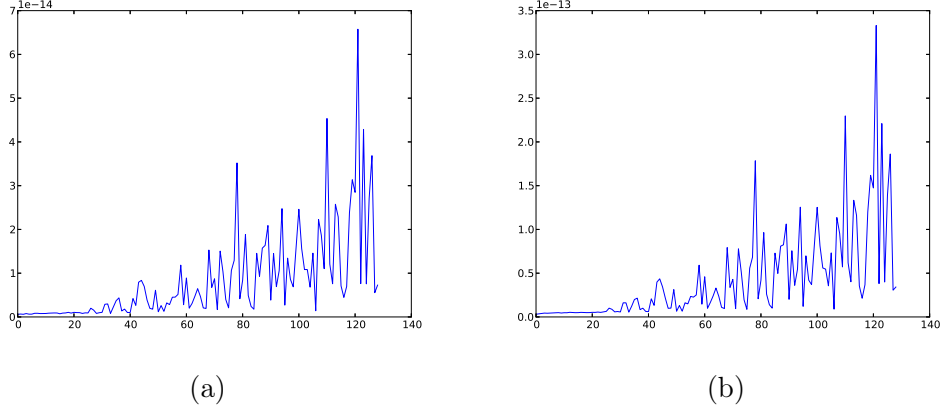


Figure 3.6: Numerical solution from the FE scheme in 1D (a) and 2D (b) versus the exact numerical solution of the FE scheme. Some of the terms in the numerical exact solutions are ignored to prevent overflow and this is responsible for the increasing error which is slightly larger than machine precision.

3.3.4 Verification of BE scheme by exact numerical solution

The exact numerical solution to the BE scheme is found by solving the linear system which arises from the discretization.

$$\begin{aligned}
 \mathbf{M}\mathbf{u}^{n+1} &= \mathbf{u}^n \\
 \mathbf{u}^{n+1} &= \mathbf{M}^{-1}\mathbf{u}^n \\
 &= \mathbf{M}^{-1}(\mathbf{M}^{-1}\mathbf{u}^{n-1})
 \end{aligned}$$

Doing the separation to the end relates the n 'th time step to the initial condition

$$\mathbf{u}^{n+1} = (\mathbf{M}^{-1})^{n+1} \mathbf{u}^0 \quad (3.12)$$

Note that $(\mathbf{M}^{-1})^{n+1}$ is the inverse of \mathbf{M} to the $n + 1$ 'th power.

Taking the inverse of \mathbf{M} will result in a dense matrix where a lot of the entries are close to zero (e.g. 10^{-20}). Doing calculations with such a matrix gives a lot of round off errors which will reduce the accuracy of the numerical exact. The error should theoretically be machine precision, but is expected to

at least be much smaller than Δt . Errors from both 1D and 2D simulations are shown in Figure 3.7

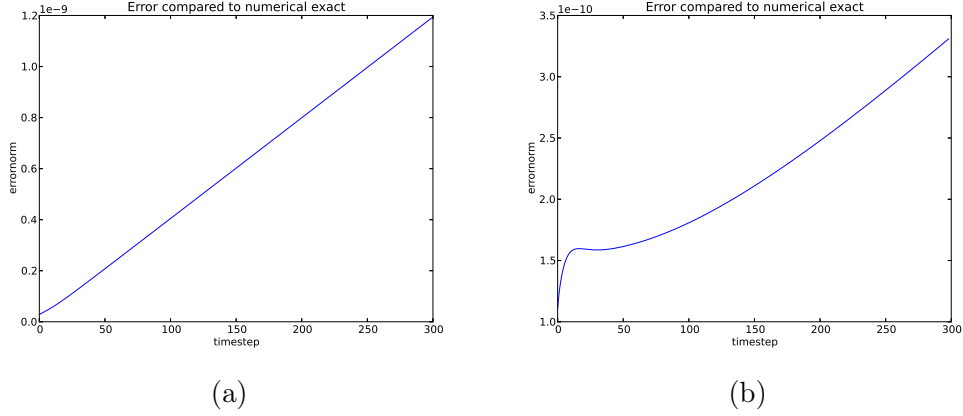


Figure 3.7: Plots showing the error for the BE scheme in 1D (a) and 2D (b) compared to the numerical exact solution. The error is not machine precision, but significantly smaller than Δt which for these simulations is $\Delta t = 0.01$. This increased error originates in the many roundoff errors in the inverted matrix where a lot of terms 10^{-16} and smaller.

3.4 Verification of the RW solver

The aim of this section is to verify that the implemented RW model will solve the diffusion equation with on the correct time scale, and to verify that the error term from the RW solver is dependent on the conversion factor, Hc . In order to carry out the tests a new initial condition that can easily be recreated by random walkers will be introduced.

$$u(t = 0, x) = H\left(x - \frac{1}{2}\right) \quad (3.13)$$

$H(x)$ denotes the Heaviside step function defined by its properties

$$H(x - a) = \begin{cases} 1 & x \geq a \\ 0 & x < a \end{cases} \quad (3.14)$$

The new initial condition gives a new exact solution which is found by separation of variables as

$$u(x, t) = \frac{1}{2} + \sum_{n=1}^{\infty} \frac{2 \sin(\frac{n\pi}{2})}{n\pi} e^{-(n\pi)^2 t} \cos(n\pi x) \quad (3.15)$$

Figure 3.9 verifies that the error term improves by adding more walkers. And figure 3.8 shows that the RW model will fulfill the chosen time step on the PDE level. The convergence is not very good, but this is expected because of the fluctuations which causes the error to stabilize around a certain value rather than converge to zero.

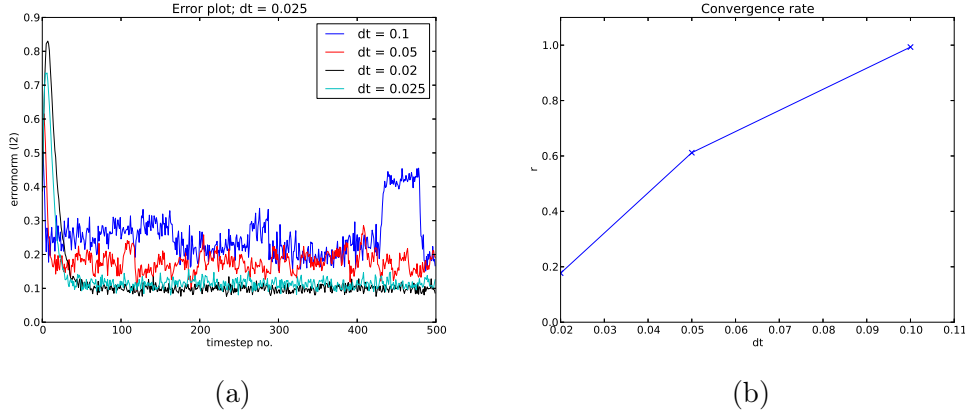


Figure 3.8: Error plot (a) and convergence test (b) for 1D RW solver using a Heaviside step function as initial condition. In these tests both the time step and the conversion factor are changed for each simulation, and the conversion factor follows the previously proposed limit $Hc \geq \frac{1}{\Delta t^2}$. For each Δt the RW simulation does 250 steps with a step length calculated from (2.17). The expected convergence rate is 0.5, and to some extent this is achieved here, note however that due to fluctuations in the solution getting a good error measure is difficult and beyond our control.

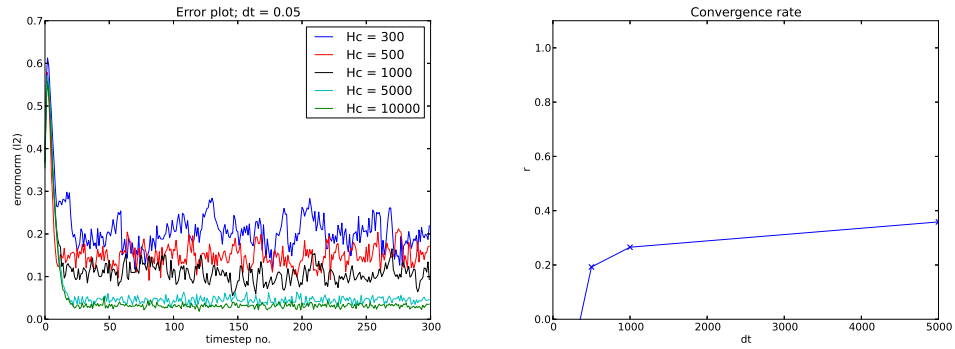


Figure 3.9: As a comparison to Figures 3.8 this test has been done for the same 1D Heaviside step function as initial condition, but keeps the time step fixed at $\Delta t = 0.05$ and increases the conversion factor. The convergence rate (fig. b) is worse than for the time convergence test, and seems to reach a limit where increasing the number of walkers has little effect on the error.

3.5 Verification of the hybrid solver

This section aims to achieve first order convergence in time for the hybrid solver by introducing a sufficient number of walkers. Effects of varying the number of mesh points affected by the RW solver will also be illustrated.

The number of walkers needed to achieve first order convergence in time for the hybrid solver is governed by

$$Hc \geq \frac{1}{\Delta t^2} \quad (3.16)$$

This demand makes use of the FE scheme difficult because of the stability criterion, and so all tests are done with the BE scheme. Figure 3.10 shows first order convergence for the hybrid model.

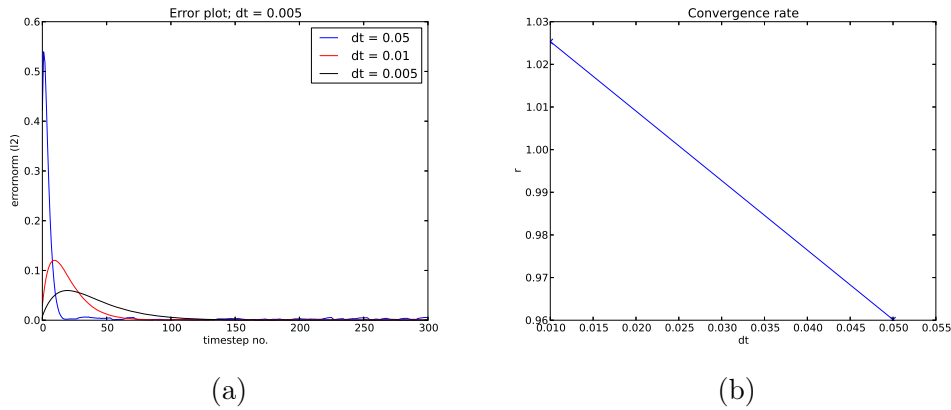
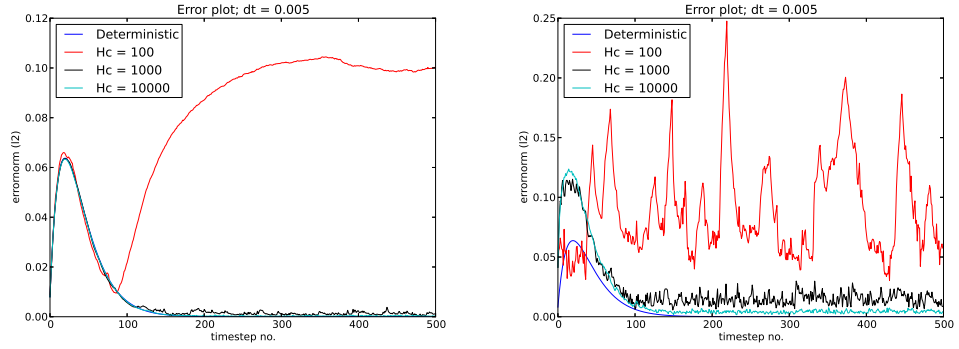


Figure 3.10: 3.10a shows the error plot for a test where Δx was fixed at $\Delta x = \frac{1}{100}$ and Δt was reduced from 0.05 to 0.01 and finally to 0.005. The conversion rate, Hc was updated for each simulation to have the value $Hc = \frac{1}{\Delta t^2}$, meaning the error from the walkers should be smaller than the error from the time derivative. Walkers are placed on 10% of the mesh from $x = 0.4$ to $x = 0.5$. 3.10b shows the convergence rate in time for the same test.

Decreasing or increasing the relative size of the microscopic model as serious effects on the error and is illustrated in figure 3.11



(a) Having walkers on 5% of the mesh points. (b) Having walkers on 35% of the mesh points.

Figure 3.11: The effect of increasing the size of the walk area for a fixed $\Delta t = 0.05$ and $\Delta x = 0.01$ using the BE discretization.

Chapter 4

Physical application

The following chapter is included to show that the developed hybrid diffusion solver can be applied to real life problems with only minor modifications.

4.1 Physical scope

Her bør det stå noe om hva vi holder på med og hvor det kommer fra, samt hvorfor det er interessant og hva vi faktisk kommer til å måle (diffusjonstid for walkers gjennom spine neck).

PKC γ is associated with memory storage and will diffuse out of the cell body (soma) of a neuron through a dendrite into a dendritic spine to reinforce the absorption of neurotransmitters. This application is aimed at modeling PKC γ diffusion in a dendrite and into spines. Because of symmetry the dendrite will be modeled as 1D continuous diffusion while the diffusion in spines will be modeled as a random walk. Effectively we will investigate the diffusion time for random walkers through spine necks which are very narrow ($\leq 0.5\mu\text{m}$). The results will be compared to results by Craske et.al [2].

4.2 Implementation

There is a difference between the approach of the developed hybrid solver and the dendrite - spine system with respect to geometry which is best summarized in figure 4.1.

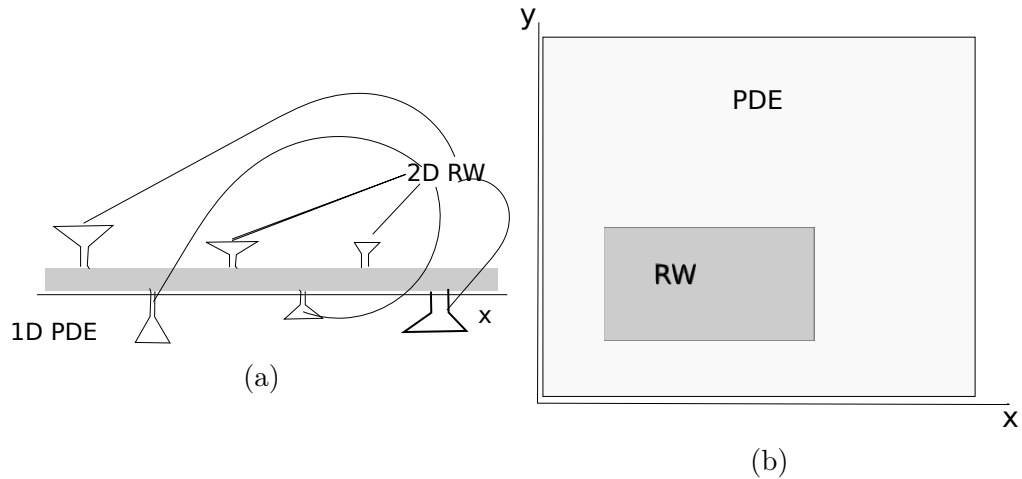


Figure 4.1: The geometric difference between the original hybrid diffusion solver (b) and the problem of PKC γ diffusing into dendritic spines (a).

The hybrid diffusion solver has been slightly modified in order to recreate the new geometry. Mathematically the largest difference is that the random walkers are left alone, meaning that the diffusion equation is not solved in the spines. There are also some differences in the coupling; at each PDE time step there is a probability for an enzyme to diffuse into a spine if the concentration at the beginning of the spine neck is large enough. This probability decreases with the number of enzymes which have already diffused into the spine. The width of a spine neck is determined by the number of PDE mesh points it connects to, all other length parameters are determined randomly but required to fit with measurements from Arrelano et.al [1].

4.3 Parameters

The new problem requires the introduction of some additional parameters as well as a reformulation of limiting factors for some known parameters, all of which are described in the following table.

Parameter	Explanation	Expression/ typical value	Origin
Δt	time-step	Δx^2	stability criterion FE scheme
Δx	spatial resolution	$\frac{1}{2}\min(\text{spine neck diameter})$	estimated
Hc	conversion factor	5-24	estimated by calculations of concentration levels taken from [3] and spine/dendrite volume ratios. See later for discussion.
$u(t = 0)$	initial condition value	$5 \frac{\text{nMol}}{\text{L}}$	estimated from values found in [3]
p_{ds}	probability to diffuse into a spine	$0.1 \cdot \Delta x \cdot \Delta t$	estimated. An important ability of this parameter should be that wide necked spines have larger probability and that a certain flux is maintained (on average), meaning that the flux should be independent of Δt
p_{ab}	probability for PKC γ to be absorbed, and removed from simulation, per time-step taken in spine head	100%	estimated

Table 4.1: Parameters which play an important role in the simulation of PKC γ diffusion into dendritic spines with explanations, expressions/typical values and an indication as to where the value/expression has its origin.

Bibliography

- [1] Jon I Arellano et al. “Ultrastructure of dendritic spines: correlation between synaptic and spine morphologies”. In: *Frontiers in neuroscience* 1.1 (2007), p. 131.
- [2] Madeleine L Craske et al. “Spines and neurite branches function as geometric attractors that enhance protein kinase C action”. In: ().
- [3] Peter E Light et al. “Protein Kinase C–Induced Changes in the Stoichiometry of ATP Binding Activate Cardiac ATP-Sensitive K⁺ Channels A Possible Mechanistic Link to Ischemic Preconditioning”. In: ().
- [4] George Marsaglia. “Xorshift rngs”. In: *Journal of Statistical Software* 8.14 (2003), pp. 1–6.
- [5] Mathis Plapp and Alain Karma. “Multiscale finite-difference-diffusion-Monte-Carlo method for simulating dendritic solidification”. In: *Journal of Computational Physics* 165.2 (2000), pp. 592–619.