MULTISCALE MODELING OF DIFFUSION PROCESSES IN THE BRAIN

by

Fredrik E Pettersen
f.e.pettersen@fys.uio.no

THESIS
for the degree of
MASTER OF SCIENCE

Faculty of Mathematics and Natural Sciences
University of Oslo

June 2013

Abstract

This is an abstract text.

To someone

This is a dedication to my cat.

Acknowledgements

I acknowledge my acknowledgements.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The very first approach was to simply try the problem on a bit. That is to try and substitute some small part of the mesh in a Finite Difference Diffusion solver (Froward Euler scheme) with a stochastic diffusion solver. A random walk method was implemented on part of the mesh to take over the equation-solving. This was done in 1 and 2 spatial dimensions with the aim of finding potential difficulties so that we can further investigate them.
Upon switching length-scales a fundamental question arises almost immediately; what is the contiuum limit? In our case this question takes a slightly different, and possibly more answearble form; what is the conversion rate beween the continuum model and the microscopic model, and by extension, what does a walker correspond to? The first instinct of this candidate was to just try some conversion rate (say some value corresponds to som enumber of walkers), and this was implemented in both 1 and 2 dimensions.

Chapter 2

Random walks

In this chapter we will take a closer look at random walks, both in general and the transition from the statistical view to partial differential equations. We will take a look at different algorithms to produce random walks, and discuss their pros and cons in light of this project.

2.1 Introduction to random walks

The most basic random walk is a walker on the x-axis which will take a step of a fixed length to the right with a probability p, or to the left with a probability $q = 1 - p$. Using (pseudo-) random numbers on a computer we can simulate the outturn of a random walk. For each step (of which there are N) we draw a random number, r, between 0 and 1 from some distribution (say a uniform one) which will be the probability. If $r \leq p$ the walker will take a step to the left, otherwise it will take a step to the left. After the N steps the walker will have taken R steps to the right, and $L = N - R$ steps to the left. The net displacement from the origin will be $S = R - L$.

2.1.1 Further discussion and analysis of the introduction

If we do sufficiently many walks, the net displacement will vary from $S = +N$ to $S = -N$ representing all steps to the right and all steps to the left respectively. The probability of all steps beeing to the right is $P_N(N) = p^N$. Should one of the steps be to the left, and the rest to the right we will get a net displacement of $S = N - 2$ with the probability $P_N(R = N - 1) = Np^{N-1}q$. We can generalize this to finding the probability of a walk with a R steps to the right as

$$P_N(R) = \binom{N}{R} p^R q^{N-R} \tag{2.1}$$

where $\binom{N}{R} = \frac{N!}{R!(N-R)!}$ is the number of walks which satisfy the net displacement in question, or the multiplicity of this walk in statistical mechanics terms. Equation 2.1 is the Bernoulli probability distribution, which is normalized.

$$\sum_{R=0}^{N} P_N(R) = (p+q)^N = 1^N = 1$$

We can use this distribution to calculate various average properties of a walk consisting of N steps. For example, the average number of steps to the right is

$$\langle R \rangle = \sum_{R=0}^{N} R P_N(R) = \sum_{R=0}^{N} \binom{N}{R} R p^R q^{N-R} =$$

$$p\frac{d}{dp}\sum_{R=0}^{N} \binom{N}{R} p^R q^{N-R} = p\frac{d}{dp}(p+q)^N = Np(p+q)^{N-1} = Np$$

From this we can also find the average value of the net displacement using $S = R - L = R - (N - R) = 2R - N$.

$$\langle S \rangle = \langle 2R \rangle - N = 2Np - N\underbrace{(p+q)}_{=1} = N(2p - p - q) = N(p - q)$$

We notice that the average net displacement is greatly dependent on the probability of the walk and that any symmetric walk will have an expected net displacement of zero. In many cases

we will be more interrested in the mean square displacement than the displacement itself. This can also be calculated rather straightforwardly.

$$\langle R^2\rangle = \sum_{R=0}^{N} R^2 P_N(R) = \sum_{R=0}^{N} \binom{N}{R} R^2 p^R q^{N-R} =$$
$$\left(p\frac{d}{dp}\right)^2 \sum_{R=0}^{N} \binom{N}{R} p^R q^{N-R} = \left(p\frac{d}{dp}\right)^2 (p+q)^N$$
$$= Np(p+q)^{N-1} + p^2 N(N-1)(p+q)^{N-2} = (Np)^2 + Np(1-p) = (Np)^2 + Npq$$

Like before, the average nett displacement is given as $S^2 = (2R-N)^2$ and we obtain

$$\langle S^2\rangle = 4\langle R^2\langle -4N\langle R\rangle + N^2 = 4((Np)^2 + Npq) - 4N^2p + N^2$$
$$= N^2(4p^2 - 4p + 1) + 4Npq = N^2(2p-1)^2 + 4Npq = N^2(p-q)^2 + 4Npq$$

which for the 1D symmetric walk gives $\langle S^2\rangle = N$ and the variance, denoted $\langle \Delta S^2\rangle = \langle\langle S^2\rangle - \langle S\rangle^2\rangle$, is found by instertion as

$$\langle \Delta S^2\rangle = \langle N^2(p-q)^2 + 4Npq - (N(p-q))^2\rangle = 4Npql^2 \tag{2.2}$$

where l is the step length ($\Delta x_{i+1} = \Delta x_i \pm l$).
When the number of steps gets very large we can approximate the Bernoulli distribution 2.1 by the Gaussian distribution. This is most easily done in the symmertric case where $p = q = \frac{1}{2}$, but it is sufficient for the steplengths to have a finite variance (*find something to refer to*). The Bernoulli distribution then simplifies to

$$P(S,N) = \left(\frac{1}{2}\right)^N \frac{N!}{R!L!} \tag{2.3}$$

on which we apply Stirlings famous formula for large factorials $n! \simeq \sqrt{2\pi n} n^n e^{-n}$.

$$P(S,N) = \left(\frac{1}{2}\right)^N \frac{N!}{R!L!}$$
$$= \exp\left(-N\ln 2 + \ln\sqrt{2\pi N} + N\ln N - \ln\sqrt{2\pi R} - R\ln R - \ln\sqrt{2\pi L} - L\ln L\right)$$
$$= \sqrt{\frac{N}{2\pi RL}}\exp\left(-R\ln\frac{2R}{N} - L\ln\frac{2L}{N}\right)$$

Where we have used $R + L = N$. We now insert for $\frac{2R}{N} = 1 + \frac{S}{N}$ and $\frac{2L}{N} = 1 - \frac{S}{N}$ and expand the logarithms to first order, $RL = \frac{N^2 - S^2}{4}$ in the prefactor, and approximate $1 - \frac{S^2}{N^2} \simeq 1$. This gives

$$P(S,N) = \sqrt{\frac{2}{\pi N}}c \tag{2.4}$$

which is an ordinary, descrete Gaussian distribution with $\langle S\rangle = 0$ and $\langle S^2\rangle = N$. If we keep assuming that the walker is on the x-axis, and let the step length, a, get small the final position will be $x = Sa$ which we can assume is a continous variable. Similarly, we let the time interval between each step, τ, be small and let the walk run for a contious time $t = N\tau$. This changes the distribution 2.4 to

$$P(x,t) = \frac{1}{2a}\sqrt{\frac{2\tau}{\pi t}} e^{-\frac{x^2\tau}{2a^2 t}}. \tag{2.5}$$

The prefactor $\frac{1}{2a}$ is needed to normalize the continous probability distribution since the sepatation between each possible final position in walks with the same number of steps is $\Delta x = 2a$. We also introduce the diffusion constant

$$D = \frac{a^2}{2\tau} \tag{2.6}$$

making the distribution

$$P(x,t) = \sqrt{\frac{1}{4\pi Dt}} e^{-\frac{x^2}{4Dt}} \tag{2.7}$$

2.1.2 More general Random Walks

In the more general case, the position of a random walker,$\mathbf{r}$ at a time t_i is given by the sum

$$\mathbf{r}(t_i) = \sum_{j=0}^{i} \Delta \mathbf{x}(t_j) \tag{2.8}$$

where $\Delta \mathbf{x}(t_j) = (\Delta x(t_j), \Delta y(t_j), \Delta z(t_j))$ in 3D. Each $\Delta x, y, z$ is a random number drawn from a distribution with a finite variance $\sigma^2 = \langle \Delta x^2 \rangle$. By the central limit theorem, any stochastic process with a well defined mean and variance can, given enough samples, be approximated by a Gaussian distribution. This means that the probability of finding the walker at some position x after M steps is

$$P(x, M) \propto e^{-\frac{x^2}{2M\sigma^2}} \tag{2.9}$$

Remember that the actual gaussian distribution is

$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{(n-\mu)^2}{2\sigma^2}\right)$$

We can introduce an Einstein relation $\sigma^2 = 2D\Delta t$ and the obvios relation $t = M\Delta t$ to get a more desirable exponent. We see that $\langle \Delta x^2 \rangle = 2Dt$. *The introduction of the Einstein relation might put some restrictions on our model.* Normalizing the expression gives us

$$P(x,t) = \sqrt{\frac{1}{4Dt}} e^{-\frac{x^2}{4Dt}} \tag{2.10}$$

If we have a large number, N, of walkers, their concentration will be $C(x,t) = NP(x,t)$. The concentration is conserved, so any amount that flows out of an area must reflect as a decrease in concentration. We can express this by the flow of concentration

$$J_x = \frac{d}{dt} \int_x^\infty C(x,t)dx = -D\frac{\partial C(x,t)}{\partial x} \tag{2.11}$$

or $\mathbf{J} = -D\nabla C$. Using some vector calculus we obtain

$$\frac{d}{dt}\int_V C(x,t)dV = -\oint \nabla C \cdot d\mathbf{S} = -\int_V D\nabla^2 C dV \tag{2.12}$$

which is the diffusion equation.

$$\frac{\partial C}{\partial t} = D\nabla^2 C \tag{2.13}$$

By insertion we can check that this version (2.10) of the gaussian distribution fulfills the diffusion eqution.

$$\frac{\partial P}{\partial t} = -\frac{4\pi D \exp\left(-\frac{x^2}{4Dt}\right)}{2\sqrt{(4\pi Dt)^3}} + \frac{x^2 \exp\left(-\frac{x^2}{4Dt}\right)}{4Dt^2\sqrt{4\pi Dt}}$$

$$= \exp\left(-\frac{x^2}{4Dt}\right)\left(\frac{8Dx^2}{2\sqrt{\pi}(4Dt)^{5/2}} - \frac{(4D)^2 t}{2\sqrt{\pi}(4Dt)^{5/2}}\right) = \frac{4D\exp\left(-\frac{x^2}{4Dt}\right)(x^2 - 2Dt)}{\sqrt{\pi}(4Dt)^{5/2}}$$

$$D\frac{\partial^2 P}{\partial x^2} = \frac{D}{\sqrt{4\pi Dt}}\frac{\partial}{\partial x}\left[-\exp\left(\frac{-x^2}{4Dt}\right)\left(\frac{-2x}{4Dt}\right)\right]$$

$$= \frac{2D}{4Dt\sqrt{4\pi Dt}}\exp\left(\frac{-x^2}{4Dt}\right)\left[1 - x\left(\frac{2x}{4Dt}\right)\right] = \frac{4D\exp\left(-\frac{x^2}{4Dt}\right)(x^2 - 2Dt)}{\sqrt{\pi}(4Dt)^{5/2}}$$

2.1.3 Choosing random walk algorithm

As this article points out [] the simplest random walk model, which places walker on discrete mesh points and uses a fixed step length, has been used with great succsess to model diffusion processes. However, this model will struggle with reproducing anisotropic diffusion, that is $D = D(x)$. *Author* also suggests a method for improving the results by adjusting the step length according to to position, thus effectively adjusting the diffusion constant of the walk as well. We see then that the simplest model is rather robust, and well tested. However, the aim of this project is to combine two realistic models for diffusion on different length scales, and the simplest random walk model has one fundamental flaw in that view; it is not a realistic model for diffusing particles. Brownian motion is a more realistic physical model for diffusing particles, and can (I think) quite easily be modified to model anisotropic diffusion as well. We can model brownian motion simply by using equation 2.8, and we can with a bit of work expand it to model collisions between walkers as well.

2.1.4 Potential problems or pitfalls with combining solutions

There are a few obvious difficulties we can expect to run into in our planned project. Future ones will be added here as well.

- Different timescales
  The PDE-solver will be operating with some timestep Δt which will, depending on the discretization of the PDE, have some constraints and will definately have an impact on the error. The walkers will, as we have just seen, solve the diffusion equation as well, but with some different $\Delta\tilde{t}$ which is smaller than the timestep on the PDE level. Depending on the couplig chosen between the two models this difference will have some effect or a catastrophic effect on the error. Running some number of steps, N, on the random-walk level should eventually sum up to the timestep on the PDE level, $\sum_{i=0}^{N} \Delta\tilde{t} = \Delta t$, but it might turn out to be difficult to makes sure that this is fulfilled.

- Boundary conditions
  To combine the two models we will need to put restricting boundary conditions on the random walks. This is not usually done (as far as I have seen), but not very difficult. Finding a boundary condition that accurately models the actual system turns out to be

quite straightforward, so long as the walk-domain is not on the actual boundary of the whole system. We can assume that the number of walkers in the walk-domain is conserved for each PDE-timestep, and thus no walkers can escape the domain. Implementing perfectly reflecting boundaries solves this quite well. This means that the flux of walkers out of a boundary is zero, which is the same as Neumann boundary conditions on the PDE level.
Dirichlet boundaries can (probably) be implemented by adding or removing walkers on the boundaries (or in a buffer-zone around them) untill we have the desired concentration of walkers.

- Negative concentration of walkers
  Should not be to difficult...

2.2 Actually combining

This section will deal with the actual combination of the two models.

2.2.1 *Some subtitle*

The basic structure of the program is to have one solver-object which contains one PDE-solver for the normal diffusion equation, and a linked list of random walk-solvers and their relevant areas. Before we start we must add an initial condition along with some parameters such as the diffusion constant (/tensor) and Δt, and we have the opportunity to mark areas on the mesh where we want random walk solvers. The method for adding walk-areas will map them to an index and set the initial condition for the walk. In the future we plan to add the possibility of setting boundary conditions and having anisotropy follow into the random walk solvers as well. At each timestep we call the solve-method of the combined solver, which in turn calls the solve method for the PDE-solver. We then loop over the random walk solvers and call their solve-methods. The results of these are inserted in the solution from the PDE using some rutine (e.g. the average of the two) and the timestep is done.

2.2.2 Probability distribution and timesteps

As we saw in section **??** the probability of finding a walker at a position x_i after some N timesteps (on the walk-scale) is (in the limit of large N) given as the gaussian distribution. In our application, however, we are not interrested in finding the walker at an exact position, but in an interval around the meshpoints sent to the walk-solver. This interval is (for obvious reasons) $x_i \pm \frac{\Delta x}{2}$ where Δx is the mesh resolution on the PDE level. We will also run the walk solver for some n timesteps on the random-walk scale (where n steps on the random walk scale is the same as one step on the PDE scale). This slightly modifies our distribution into

$$P(x_i \pm \Delta x, t_{n+1}) = \frac{1}{\sqrt{4\pi DN\Delta\tilde{t}}} \exp\left(\frac{(x \pm \Delta x)^2}{4DN\Delta\tilde{t}}\right) \tag{2.14}$$

This makes the concentration of walkers $C(x,t) = MP(x,t)$

$$C(x_i \pm \Delta x, t_{n+1}) = \frac{M}{\sqrt{4\pi DN\Delta\tilde{t}}} \exp\left(\frac{(x \pm \Delta x)^2}{4DN\Delta\tilde{t}}\right) \tag{2.15}$$

For each PDE-timestep we reset the walkers to have some new initial condition. This is done to make sure that statistical fluctuations will not put the diffusive process “off course”. The point is that $C(x_i \pm \Delta x, t_{n+1})$ will be dependent on the initial condition $C(x_i \pm \Delta x, t_n)$.

Looking at the difference in timestep size between the two lengthscales we see from equation 2.4 that the stepsize on the random walk scale is dependent on the variance in the actual steps (This is in principle the Einstein relation).

$$\sigma^2 = \langle \Delta x^2 \rangle = 2DN\Delta\tilde{t} \implies \Delta\tilde{t} = \frac{\langle \Delta x^2 \rangle}{2DN} \tag{2.16}$$

Equating this with 2.2 gives us a first order approximation to the steplength, l

$$\begin{aligned} \langle \Delta x^2 \rangle &= 4pqNl^2 = 2DN\Delta\tilde{t} \\ l &= \sqrt{2D\Delta\tilde{t}}. \end{aligned} \tag{2.17}$$

Of course this is assuming that we use a random walk algorithm of fixed steplength.

Chapter 3

Analysis

3.1 Discretizing the PDE

To maintain a bit of generality we will look at the (potentially) anisotropic diffusion equation

$$\frac{\partial u}{\partial t} = \nabla D \nabla u + f \tag{3.1}$$

where f is some source term. The final expression and scheme will depend on how we chose to approximate the time derivative, but the spatial derivative will mostly have the same approximation.
We start off by doing the innermost derivative in one dimension. The generalization to more dimesions is trivial, and will consist of adding the same terms for the y and z derivatives.

$$\left[\frac{d}{dx}u\right]^n \approx \frac{u^n_{i+1/2} - u^n_{i-1/2}}{\Delta x}$$

Where we have made the approximate derivative around the point x_i. We then set $\phi(x) = D\frac{du}{dx}$ and do the second derivative

$$\left[\frac{d}{dx}\phi\right]^n \approx \frac{\phi^n_{i+1/2} - \phi^n_{i-1/2}}{\Delta x}$$

and insert for ϕ

$$\frac{\phi^n_{i+1/2} - \phi^n_{i-1/2}}{\Delta x} = \frac{1}{\Delta x^2}\left(D_{i+1/2}(u^n_{i+1} - u^n_{i+1}) - D_{i-1/2}(u^n_i - u^n_{i-1})\right)$$

Since we can only evaluate the diffusion constant at the mesh points (or strictly speaking since it is a lot simpler to do so) we must approximate $D_{i\pm1/2} \approx 0.5(D_{i\pm1} + D_i)$. Inerting this gives us

$$\nabla D \nabla u \approx \frac{1}{2(\Delta x + \Delta y + \Delta z)}\left((D_{i+1,j,k} + D_{i,j,k})(u_{i+1,j,k} - u_{i,j,k}) - (D_{i,j,k} + D_{i-1,j,k})(u_{i,j,k} - u_{i-1,j,k})\right) \tag{3.2}$$

3.2 Some discussion

In this numerical setup we will potentially introduce several new error sources in addition to the nomal errors introduced by numerical solution of any equation. When a part of the solution acquired is replaced by the solution from another model, which in this case is stochastic, we will change the initial condition to the next iteration in time. This might have a number of effects on our final solution. When we solve a differential equation numerically we only get an approximation to the actual solution because we are using approximate derivatives (see figure **??**). We can investigate this type of error by doing two simulations of the same problem, but replacing the solution in one of the simuations in some area by the random walk model for one timestep.

```
 path = '/home/fredriep/Dropbox/uio/thesis/doc/results/
    experiment_03102013_1218/results/'
import glob

i = []; e = []; s = []

for excl in sorted(glob.glob(path+'Excl*')):
  e.append(np.load(excl))
```

```
for incl in sorted(glob.glob(path+'Incl*')):
  i.append(np.load(incl))

for j in xrange(len(i)):
  s.append(np.max(np.abs(i[j]-e[j])))
  print s[-1]
```

As we can see this will print the maximum absolute difference between the two solutions. Since we are dealing with a diffusion process we expect that the first timestep should have the largest error and that the error will in time be killed by the diffusion process.

```
[...]
0.0
0.025832
0.0054384
0.00360192
0.00212352
0.0016940416
0.0012704
0.001173512704
0.0009942699008
0.00093037021184
0.00083786862592
0.000782433954202
0.000723955260948
0.000680186522141
0.000638864266527
0.000604769629612
```

The error is larger than Δt by a factor of 10 ($\Delta t = 0.002$)... This might be improved by increasing the number of walkers, or finding a better steplength for the walkers as well as improving the walk-algorithm etc.. The above results were obtained by using the previous timestep as input to the random walk solver. In most numerical algorithms (Euler-Chromer etc) one will benefit from using the newest timestep as early as possible. Doing the same experiment with the modification of using the latest timestep as input to the random walk solver we obtain the following:

```
[...]
0.0
0.06
0.014
0.0072
0.0036
0.00232
0.0014912
0.0011744
0.000803712
0.000703744
0.00059240448
0.000536887296
0.0004960012288
0.00047304564736
0.000437414641664
0.00041956655104
```

That is, the initial error is larger, but it is reduced faster. However, this is a stochastic process, and one should average over many experiments. An average over (only) ten experiments using twice the number of walkers (M=100) shows that the errors seem to be converging towards the same value regardless of which timestep is used as input.

3.3 Manifactured Solutions

A normal way of checking that our sceme of choise is implemented correctly is by making an exact solution to the equation and checking that the error is of the expected order. As a first, simple implenetation we have worked with the explicit Forward Euler discretization of the simplest form of the diffusion equation **??**. This discretization is expected to have an error-term of the order of Δt, which again is limited by a stability criterion. We can now decide that the solution to equation **??** should be

$$u(x,t) = e^{-t\pi^2}\cos(\pi x) + 1 \tag{3.3}$$

which satisfies our equation if we set the diffusion constant to 1.

$$\frac{\partial}{\partial t}e^{-t\pi^2}\cos(\pi x) + 1 = D\frac{\partial^2}{\partial x^2}e^{-t\pi^2}\cos(\pi x) + 1 \tag{3.4}$$

$$-\pi^2 e^{-t\pi^2}\cos(\pi x) = -\pi^2 e^{-t\pi^2}\cos(\pi x) + 1 \implies 1 = 1 \tag{3.5}$$

As we saw in section **??** the Forward Euler scheme is expected to have an error of the order of Δt. Testing only the scheme first, in 1D we get the following plot 3.1 of the maximum of the absolute value of the difference between the exact solution and the numerical solution to the equation.

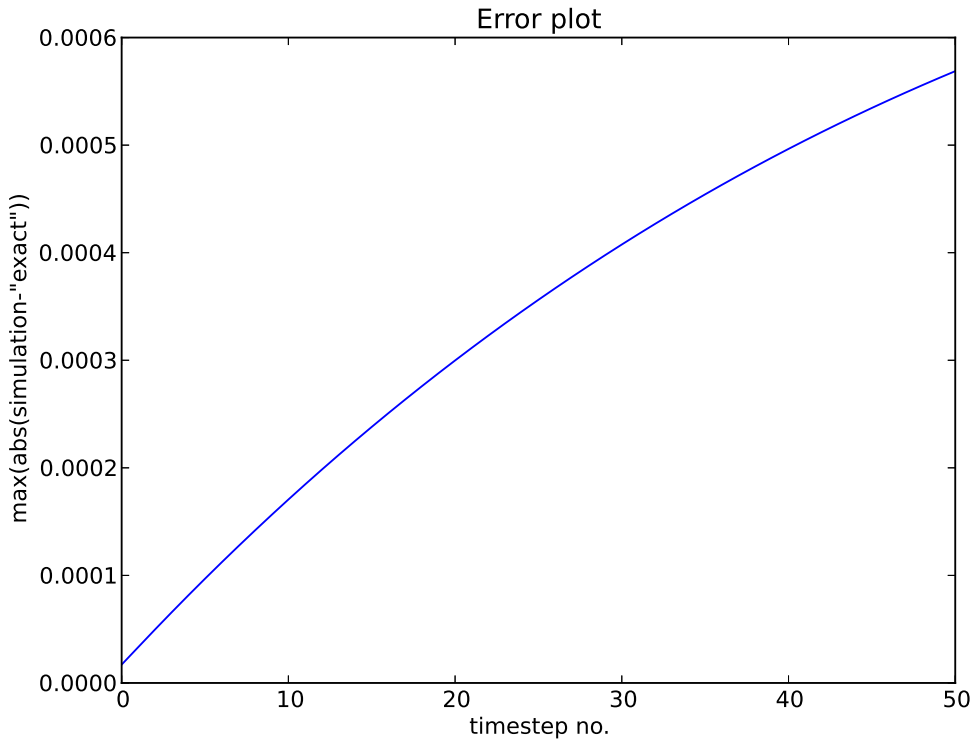

Figure 3.1: Numerical error for 1D Forwar Euler discretization.

We then introduce an area on the domain where we swich models from the normal PDE to an average of the PDE solution and the result of a random walk simulation where the initial condition is the last timestep from the PDE converted to walkers by the conversion rate given in equation 3.6. In this case we have used the parameters $a = 3$, $\Delta t = \frac{\Delta x^2}{3.0}$, $\Delta x = \frac{1}{20}$. These parameters makes one unit of $u(x,t)$ equal to some 1000 walkers.

$$C_{ij} = \frac{a}{\Delta t} U_{ij} \tag{3.6}$$

The area where the model has been replaced is between $x = 0.6$ and $x = 0.7$, which is three mesh points. In the same way as for only the simple 1D PDE case we compare the combined numerical solution from the two models to the exact solution. Figure **??** shows that the error is still of the order of Δt, and the difference between the two models are negligable.

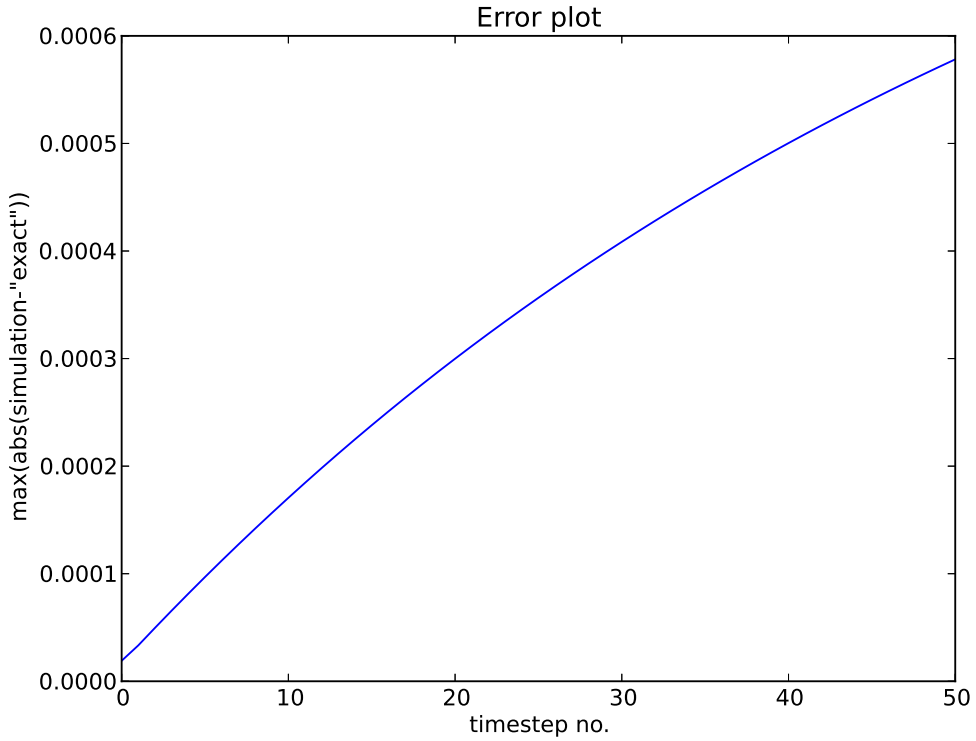

Figure 3.2: Numerical error for 1D Forward Euler discretization combined with random walk model between $x = 0.6$ and $x = 0.7$.

3.3.1 2D

Doing the same tests is 2D gives slightly different results; adding a 2D walk-domain has an influence on the error, but a rather small one. This can, however be tweaked by increasing the conversion parameters.