FYS3150 - Computational Physics

Project 5

08.12.2014

Alfred Alocias Mariadason

Candidate nr: 24

Table of Contents

1 Introduction

The Aim of this project is mainly to solve the diffusion equation in one- and two dimensions with a Markov process by employing Monte Carlo methods. The second part deals with solving the same equation with an explicit and implicit scheme.

The methods used is a random walk model (a Monte Carlo method) for the one-dimensional equation along with an explicit forward Euler scheme and an implicit backward scheme for the two-dimensional case with Jacobi's method as the iterative method for the latter. The one-dimensional random walk model is testet against an explicit forward euler (from previous project) along with an analytical solution, while the explicit and implicit schemes are compared against eachother and an analytical solution for the two dimensional diffusion equation. The stability conditions are also documented.

The programs used in this project can be found at this github link:

`https://github.com/Oo1Insane1oO/Project5v3`

2 Abstract

The reason for solving the diffusion equation is because we want to model the dominant way of transporting signals between neurons in the brain. This is archieved by diffusion of signal molecules called neurotransmitters across the synaptic cleft which seperates the cell membranes of two cells.
The basic process governs diffusion by that an action potential in the axon terminal, which is filled with neurotransmitters, gives rise to vesicles merging with the presynaptic (axon) membrane and then release the neurotransmitters into the presynaptic cleft. After which the neurotransmitters diffuse across the synaptic cleft to receptors on the postsynaptic which receives the signal. We can model this diffusion process mathematically with:

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

Here u is the concentration of neurotransmitters and D is a diffusion coefficient defined by the solvent in the postsynaptic side.

3 Theory

This section will document the theory behind the one-dimensional equation since the two-dimesional one ends up as a 2+1-dimensional problem, which means the theory behind it is essentially the same as with the one-dimensional equation..

As mentioned the process of transporting the signal is governed by diffusion. Mathematically we can model this by:

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

Here u is the concentration of the particular neurotransmitter(the mentioned molecule). D is the diffusion coefficient determined by the solvent in the synaptic cleft, so D is a coeffecient which changes by environment.

To make this problem easier we can make some assumptions.

- 1.) The neurotransmitters are released roughly equally.

- 2.) The synaptic cleft if roughly equally wide across its whole.

With these assumptions we can assume that the neurotransmitter concentration only varies in one direction, from presynaptic to postsynaptic. We can choose this to be our x-direction and the diffusion equation reduces to:

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

Now, in a real diffusion process the neurotransmitters will occacionally bump into the presynaptic cleft and be absorbed(temporally) by the receptor at the postsynaptic cleft. This imposes following boundary and initial conditions:

$$u(x=0, t>0) = u_0,\ u(x=d, \text{all } t) = 0,\ u(0<x<d, t<0) = 0$$

With this the solution to the process is given by:

$$\frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial u(x,t)}{\partial t},\ t>0,\ x\in[0,d]$$

The initial conditions are given by:

$$u(x,0) = 0, \quad 0<x<d$$

And the boundary conditions are:

$$u(0,t) = 1, \quad t>0$$
$$u(d,t) = 0, \quad t>0$$

3.1 Analytical solution to the one-dimensional equation

For the closed-form solution we can start by looking at the differential equation itself:

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

The steady-state solution is given, by the above boundary conditions, as a linear expression:

$$u_s(x) = 1 - x$$

With this we can make a new function $v(x,t)$ and solve for v rather than u:

$$v(x,t) = u(x,t) - u_s(x) = u(x,t) + x - 1$$

The boundary conditions for v is:

$$v(0) = v(d) = 0$$

Assuming a form of seperable variable($v(x,t) = F(x)G(t)$) we have that:

$$F(x) = A\cos kx + B\sin kx,\ G(t) = C\mathrm{e}^{-k^2 t}$$

The coefficient A, B and C can be found by the boundary conditions.

$$v(x,t) = 0 \Rightarrow (A\cos kx + B\sin kx)C = 0$$

We observe that $A = 0$, $C = 1$, and $k = \frac{n\pi}{d}$ fulfills the boundary conditions, however B now has n possibilities and we have:

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

The initial conditions for v gives:

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

To solve this we can use theory on Fourier series and obtain B_n. B_n is then given as the coefficent of the function $(x-1)$:

$$\begin{aligned}
B_n &= \frac{2}{d}\int_0^d (x-1)\sin\left(\frac{n\pi}{d}x\right)\mathrm{d}x \\
&= \frac{2}{d}\int_0^d \sin\left(\frac{n\pi}{d}x\right)\mathrm{d}x + \frac{2}{d}\int_0^d -\sin\left(\frac{n\pi}{d}x\right)\mathrm{d}x
\end{aligned}$$

Integration by parts gives:

$$\begin{aligned}
B_n &= \frac{2}{d}\Bigg[-x\frac{d}{n\pi}\cos\left(\frac{n\pi}{d}x\right) \\
&\quad + \int \frac{d}{n\pi}\cos\left(\frac{n\pi}{d}x\right)\mathrm{d}x\Bigg]_0^d + \frac{2}{d}\left[\frac{d}{n\pi}\cos\left(\frac{n\pi}{d}x\right)\right]_0^d \\
&= -\frac{2d}{n\pi}\cos(n\pi) + \frac{2}{d}\sin(n\pi) + \frac{2}{n\pi} \\
&= -\frac{2d}{n\pi}\cos(n\pi) + \frac{2}{d}\sin(n\pi) + \frac{2}{n\pi}\cos(n\pi) - \frac{2}{n\pi}
\end{aligned}$$

Since $n = 1, 2, \ldots$, we have that $\sin(n\pi) = 0$ and $\cos(n\pi) = (-1)^n$.

$$\begin{aligned}
B_n &= -\frac{2d}{n\pi}(-1)^n + \frac{2}{n\pi}(-1)^n - \frac{2}{n\pi} \\
B_n &= \frac{2}{n\pi}((-1)^n(1-d) - 1)
\end{aligned}$$

For our case $d = 1$, which gives:

$$B_n = -\frac{2}{n\pi}$$

So the closed-form solution is:

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

And for $d = 1$ we have:

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

3.2 Analytical solution in two dimensions

For the two-dimensional case we can use the so-called Dirichlet boundary conditions given as:

$$\begin{aligned} u(0,y,t) &= (1-y)\mathrm{e}^{t} & t \geq 0,\ \ 0 \leq y \leq 1 \\ u(1,y,t) &= (1-y)\mathrm{e}^{1+t} & t \geq 0,\ \ 0 \leq y \leq 1 \\ u(x,0,1) &= \mathrm{e}^{x+t} & t \geq 0,\ \ 0 \leq x \leq 1 \\ u(x,1,t) &= 0 & t \geq 0,\ \ 0 \leq x \leq 1 \end{aligned}$$

With initial condition:

$$u(x,y,0) = (1-y)\mathrm{e}^{x} \qquad 0 \leq x, y \leq 1$$

The closed-form solution(with these boundary conditions) is then:

$$u(x,y,t) = (1-y)\mathrm{e}^{x+t}$$

This analytical solution uses boundary conditions that are different from the ones that were used in the random walk model. In the random walk model the only boundary conditons given was as simple as just setting the values at the boundary to zero, that is kill all the walkers who jump to far. The analytical solution here requires stricter boundary conditions which in turn makes the results a bit cleaner, however this representation is not optimal for comparison with the random walk model. The solution is still viable to compare with the explicit and implicit schemes however since we use the same boundary conditions for those methods.

4 Random Walk

4.1 Random walk in one dimension

The random walk model is essentially a Markov process which incoroporates a probability to make a move. One chooses a uniform random distribution of numbers and use these to determine if our walker(a particle or neurotransmitter) will move left or right(we introduce up and down for the two-dimensional problem). Basically put we use the Markov process with constraints to which new random state(which move the walker makes) to accept and which to reject.

The process itself is fairly simple to visualize. First consider a walker in one dimension. It has probability R of moving to the right and probability L of moving to the left. at time $t = 0$ we place the walker in position $x = 0$. As we iterate over time the walker will then jump either left or right with step-length Δx for every time-step Δt. The algorithm for this process is straight forward and is as follows:

1. Decide the number of walkers, total time, time-step, and probability of movement.
2. Iterate over time and number of walkers.
3. Use a random uniformly distributed number to determine which way the walker moves.
4. Move the walker according to the random number and predefined probability.
5. Write out the data.

A pseudo code of the implementation is:

```
for $t = 0; t < (time/\Delta t); t++$ do
    for walker in numberWalkers do
        if position<minPosition || position>maxPosition then
            continue
        end if
        if left then
```

```
            position += 1
        else
            position -= 1
        end if
    end for
end for
```

The step-length used here is $l_0 = \sqrt{2D\Delta t}$. Later on we will use a gaussian normal distribution and determine a number ξ and set the step-length equal to $l_0 = \sqrt{2D\Delta t}\xi$. Note that for this project the implementations assumes that the probability for jumping is 50/50, that is 50% chance for jumping left or right.

4.2 Random Walk in two dimensions

In two dimensions we need to introduce another probability of moving up and down. Essentially we just extend the boundary at which the random numbers distribute, basically we extend it from a number between 0 and 1 to be between 0 and 2. Creating an array for both the x-position and y-position we can let the walkers jump in one dimension and create an index(similar for the one-dimensional case) and use this to update the value on the grid. The boundary is now set to a square lattice, which means that $u(0,0) = u(L,L) = 0$. A pseudo code of the implementations is as follows:

```
for $t = 0; t < (time/\Delta t); t++$ do
    for walker in numberWalkers do
        if positionX<minPosition || positionX>maxPosition then
            continue
        end if
        if positionY<minPosition || positionY>maxPosition then
            continue
        end if
        if left then
            positionX += 1
        else if right then
            positionX -= 1
        else if up then
            positionY += 1
        else
            positionY -= 1
        end if
        update values
    end for
end for
```

5 Explicit and Implicit Schemes

This section will deal with the explicit forward Euler scheme and the implicit backward Euler scheme. The last method is solved by using Jacobi's method as the iterative method, which is also covered in this section.

5.1 Explicit Forward Euler scheme in two dimensions

For the explicit scheme we assume that we have a square lattice of equal length with equal number of meshpoints in x- and y-directions. We start with discretizing the differential equation:

$$\frac{\partial^2 u}{\partial x^2} = \frac{u^l_{i+1,j} - 2u^l_{i,j} + u^l_{i-1,j}}{\Delta x^2}$$
$$\frac{\partial^2 u}{\partial y^2} = \frac{u^l_{i,j+1} - 2u^l_{i,j} + u^l_{i,j-1}}{\Delta x^2}$$

Using the forward Euler formula for the derivative of time we get:

$$\frac{\partial u}{\partial t} = \frac{u^{l+1}_{i,j} - u^l_{i,j}}{\Delta t}$$

Here i and j are respectively indexes for x and y while l is for time t. Defining $\alpha = \frac{\Delta t}{\Delta x^2}$ the differental equation can be rewritten, in its discretized form, as:

$$u^{l+1}_{i,j} = u^l_{i,j} + \alpha \left[u^l_{i+1,j} + u^l_{i-1,j} + u^l_{i,j+1} + u^l_{i,j-1} - 4u^l_{i,j} \right]$$

Now we can find the constraints for the stability, with other words find the stability condition of of the scheme. Starting off by assuming that the solution is on form $u(x, y, t) = Ae^{i(ax+by-\omega t)}$, we get that:

$$\begin{aligned}\frac{\partial^2 u}{\partial x^2} &= Ae^{i(ax+by-\omega t)} \frac{e^{i(a\Delta x)} - 2 + e^{-i(a\Delta x)}}{\Delta x^2} \\ &= Ae^{i(ax+by-\omega t)} \frac{2(\cos^2 (a\Delta x) - 1)}{\Delta x^2} \\ &= Ae^{i(ax+by-\omega t)} \frac{-4(\sin^2 (\frac{a\Delta x}{2}))}{\Delta x^2}\end{aligned}$$

similar caluclations for y yields:

$$\frac{\partial^2 u}{\partial y^2} = Ae^{i(ax+by-\omega t)} \frac{-4(\sin^2 (\frac{b\Delta y}{2}))}{\Delta y^2}$$

Inserting this into the diffusion equation and we get:

$$4\alpha \left(sin^2 \left(\frac{a\Delta x}{2} \right) + \sin^2 \left(\frac{b\Delta y}{2} \right) \right) = 1 - \mathrm{e}^{i\omega \Delta t}$$

One can see that $1 - \mathrm{e}^{i\omega\Delta t} \leq 2$, giving:

$$\alpha \leq \frac{1}{2 \left(sin^2 (\frac{a\Delta x}{2}) + \sin^2 \left(\frac{b\Delta y}{2} \right) \right)}$$

And the minimum of this expression gives:

$$\alpha \leq \frac{1}{2} \Rightarrow \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2} \Rightarrow \Delta x \geq \sqrt{2\Delta t}$$

5.2 Implicit Scheme with Jacobi's method

The implicit scheme uses the backward Euler formula to discretize the first time derivative. We start off the same as in the explicit scheme by discretizing the second derivatives of positions gaining:

$$\frac{\partial^2 u}{\partial x^2} = \frac{u^l_{i+1,j} - 2u^l_{i,j} + u^l_{i-1,j}}{\Delta x^2}$$
$$\frac{\partial^2 u}{\partial y^2} = \frac{u^l_{i,j+1} - 2u^l_{i,j} + u^l_{i,j-1}}{\Delta x^2}$$

And then the first derivative of time in its discretized form with the backward Euler formula is:

$$\frac{\partial u}{\partial t} = \frac{u^l_{i,j} - u^{l-1}_{i,j}}{\Delta t}$$

Inserting this back into the diffusion equation with the same α as previous we get:

$$u^l_{i,j} = 4\alpha u^l_{i,j} - \alpha \left[u^l_{i+1,j} +^l_{i-1,j} + u^l_{i,j+1} + u^l_{i,j-1} = u^{l-1}_{i,j} \right]$$

Resulting in:

$$u^l_{i,j} = \frac{1}{1+4\alpha} \left[\alpha(u^l_{i+1,j} +^l_{i-1,j} + u^l_{i,j+1} + u^l_{i,j-1}) + u^{l-1}_{i,j-1} \right]$$

With this equation we can implement Jacobi's method. The basic algorithm goes as follows:

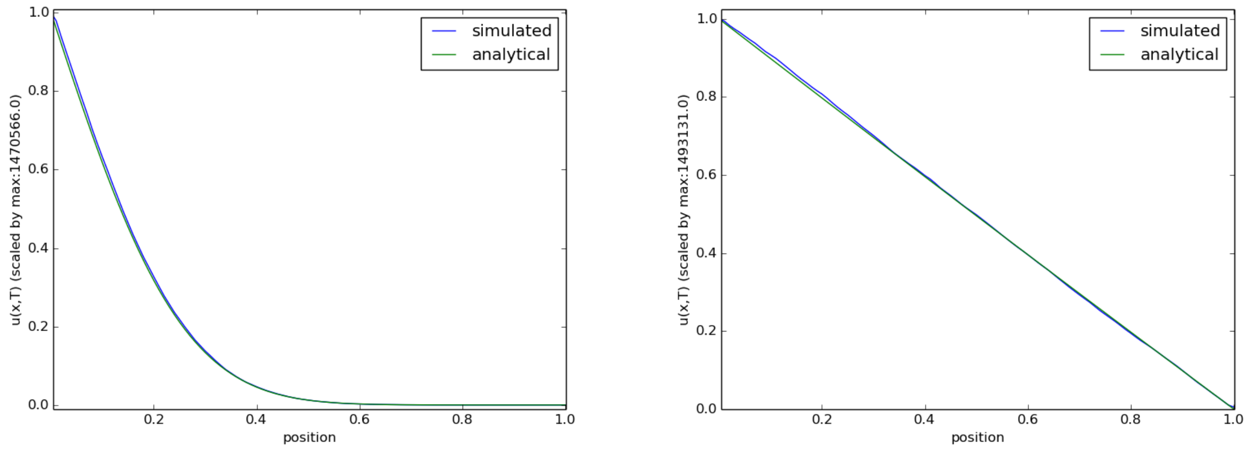
1. Determine convergence threshold.
2. Make a guess for the desired function($u(x, y)$ in our case).
3. Use the discretized function above to compute u at all interior points(x and y).
4. Check if prescribed convergence threshold is reached, continue if threshold is not reached.
5. Update values.
6. Repeat process from step 2.

Note that with Jacobi's method rather than iterating over time directly, we make an initial guess and and iterate until the new value has converged accurately enough towards the guess. However we use the time-value to calculate the boundary conditions and we use the time-step in order to define the step-length which in turn is used to define the size of the position vectors.

6 Results

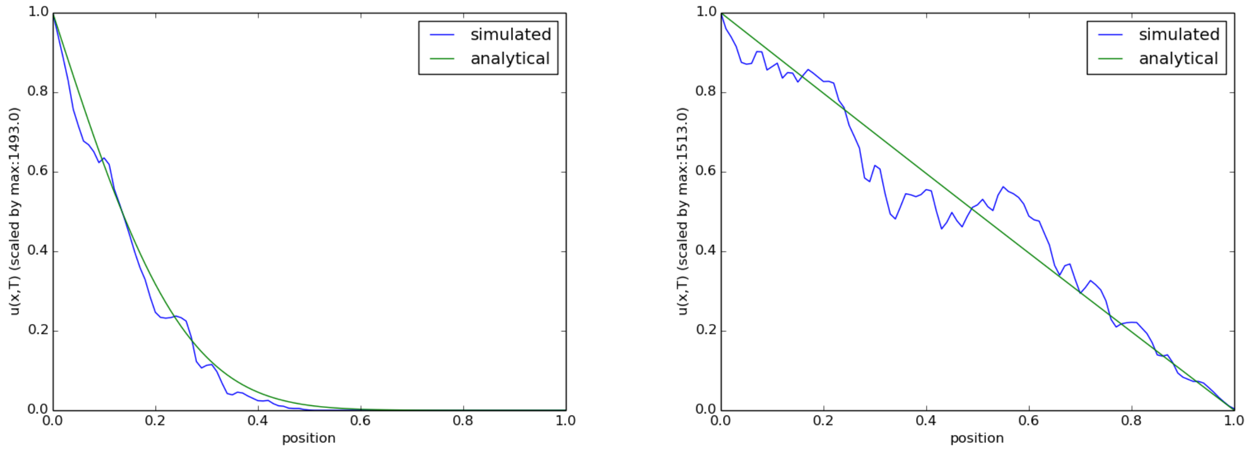
6.1 Random walk model in one dimension

Figure 1: Random walk model 10^6 walkers $\Delta t = 0.00005$



time=0.02s time=0.5s

Figure 2: Random walk model 10^3 walkers $\Delta t = 0.00005$

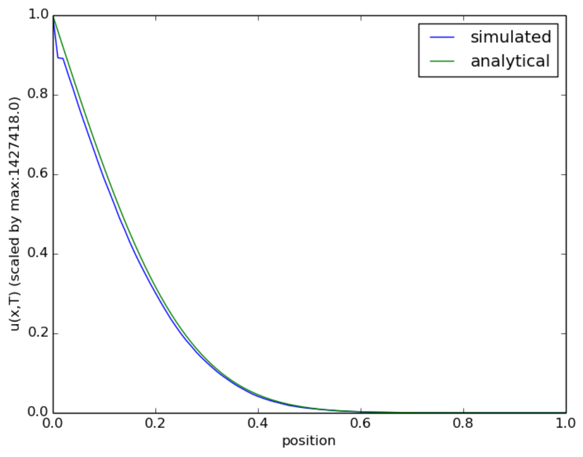


time=0.02s time=0.5s

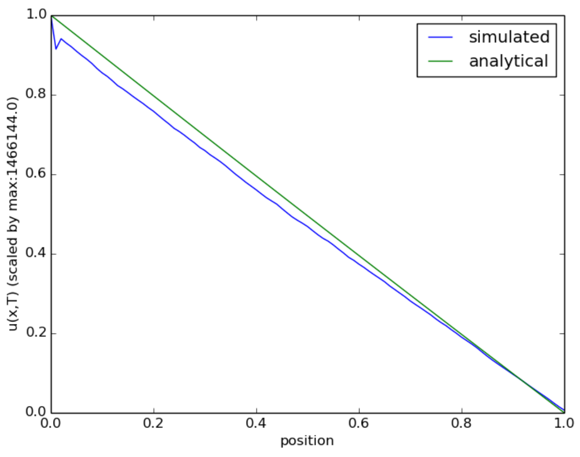
We can see here that the random walk model is fairly good as long as we keep the number of walkers high. In figure 2 one can see that reducing the number of walkers down to 1000 makes the model somewhat unstable. Also note that the distribution($u(x, T)$) is scaled by its original maximum value. This is done to get a cleaner graph that starts at 1. At $T = 0.5$ we observe that the solution is close to the steady state solution.

As mentioned there was also a model involving a gaussian distribution. The results of those simulations are as follow:

Figure 3: Gaussian random walk model 10^6 walkers $\Delta t = 0.00005$

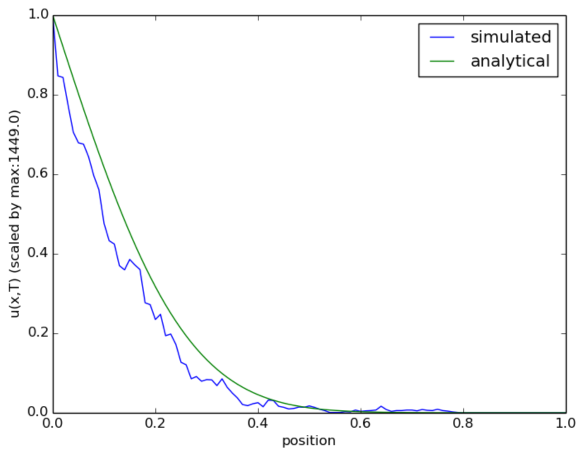


time=0.02s

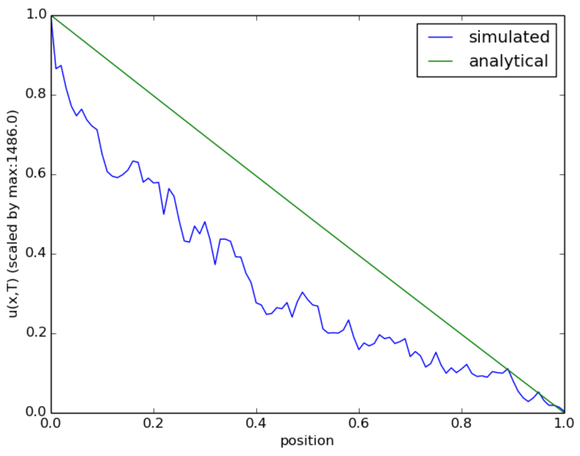


time=0.5s

Figure 4: Gaussian random walk model 10^3 walkers $\Delta t = 0.00005$



time=0.02s

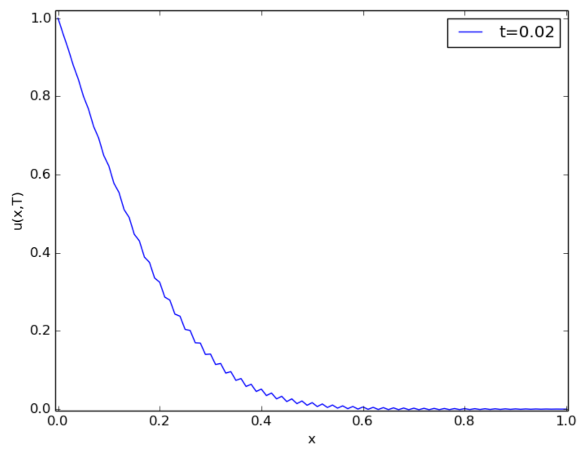


time=0.5s

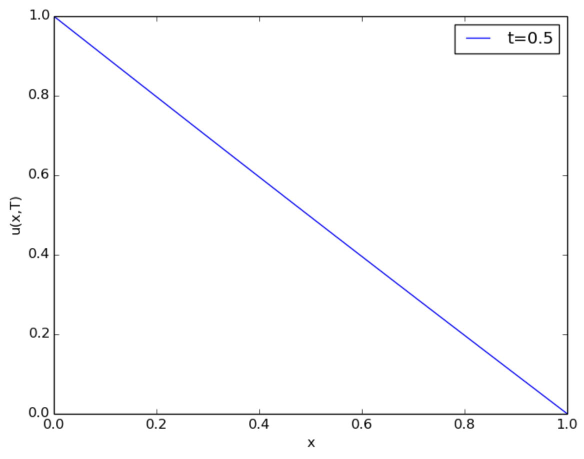
The gaussian model seems to actually be worse than the original random walk model, the graphs do not align them selves with the analytical solution as good as previously and reducing the number of walkers obviously made the model worse.

In the previous project(project 4) we were tasked with implementing an explicit scheme for solving the diffusion equation numerically. The resulting plots were:

Figure 5: Explicit forward Euler $\Delta x = 1./100, \Delta t = (\Delta x^2/2)$



time=0.02s



time=0.5s

The explicit method gets a bit unstable(the graph wobbles) for small time intervals, but is very stable as the time increases and matches very well the steady state solution for $T = 0.5$. The unstability comes from the poor stability condition given by:

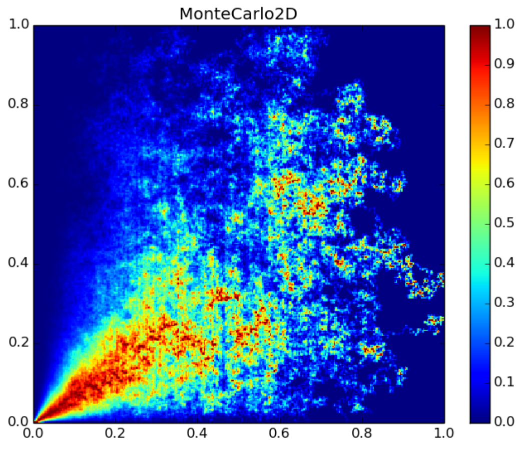
$$\alpha = \frac{\Delta t}{\Delta x^2} \leq \frac{1}{2}$$

Observe that even with the poor stability condition the solution still seems to be better than the random walk model with the gaussian distribution.

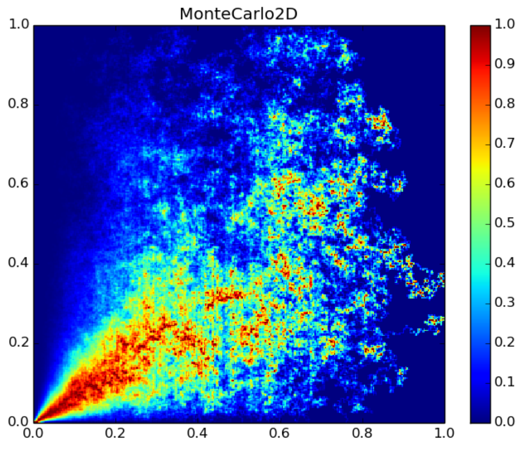
6.2 Random walk model in two dimensions

The plots below are all a gradient plot of x versus y. The colorbar on the side indicates the concentration value(a representation of the function u given by the diffusian equation) and is scaled by the original maximum value.

Figure 6: Randomwalk two dimensions 10^6 walkers $\Delta t = 0.000005$

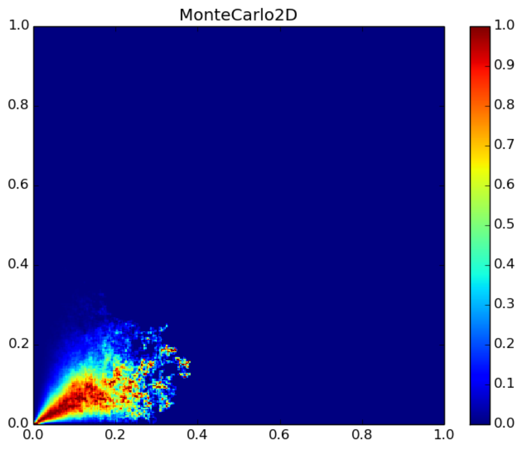


time=0.2s

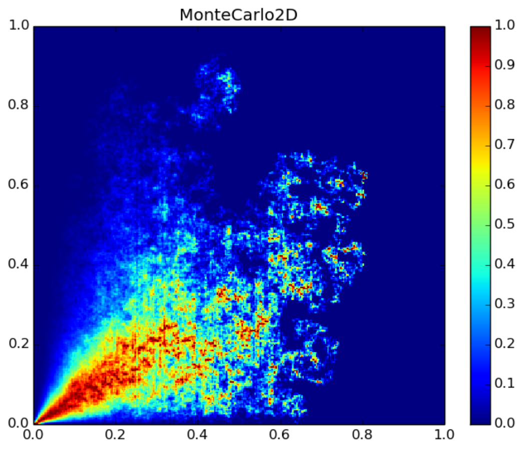


time=0.5s

Figure 7: Randomwalk two dimensions 10^6 walkers $\Delta t = 0.000005$

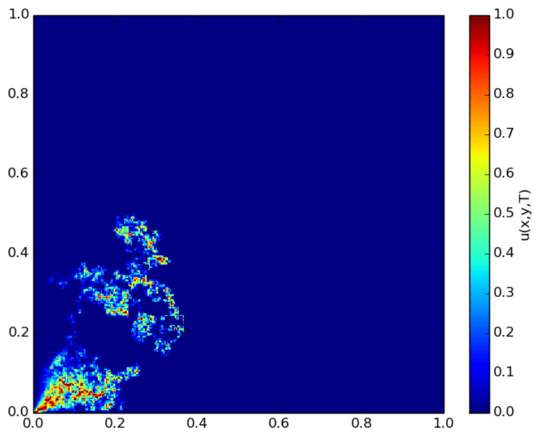


time=0.01s

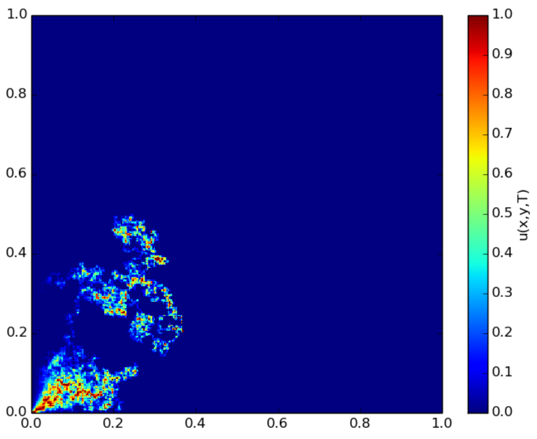


time=0.08s

Figure 8: Randomwalk two dimensions 10^4 walkers $\Delta t = 0.000005$

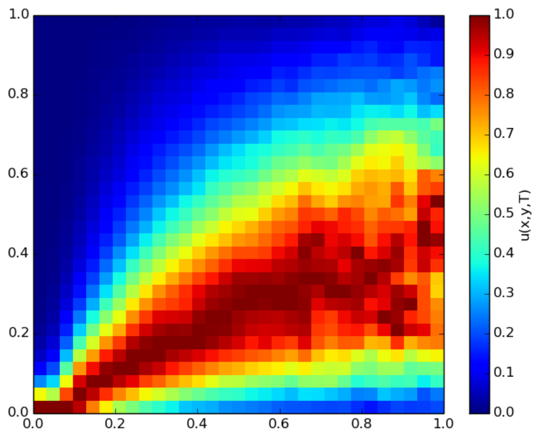


time=0.2s

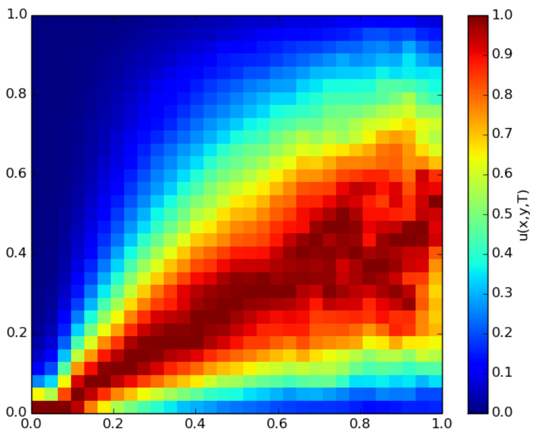


time=0.5s

Figure 9: Randomwalk two dimensions 10^6 walkers $\Delta t = 0.0005$



time=0.2s



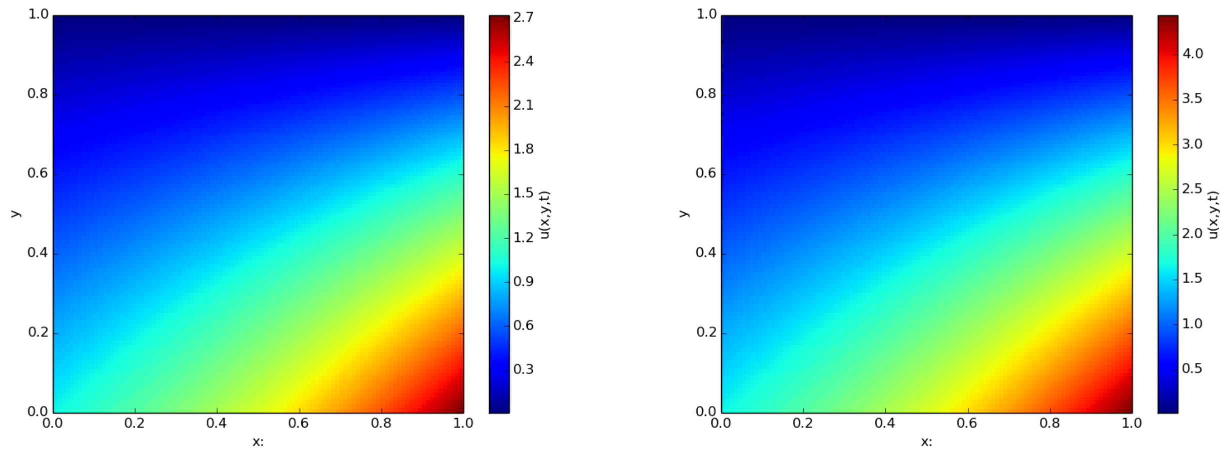
time=0.5s

The resulting plots are looking good and seem to represent a physically natural process of diffusion. It is a natural result that releasing particles in a solvent will make them spread like we see above.
Reducing the number of walker(while maintaining a low time-step) made the plots «less complete». Essentially this just means that there were not enough walkers initially to make a good distribution for the simulation resulting in a poor representation of the diffusion process.
In addition, reducing the time-step offered a pixelated and unclear plot.

6.3 Analytic solution

For the analytic solution we have an actual function that describes the process of diffusion at every meshpoint. The resulting plots incorporate a better flow to them which is good for comparing with the explicit and implicit schemes described above, but for the random walk model the analytical solution is not an accurate base for comparison. The reason for this is described above in section 3.2. The resulting plots:

Figure 10: Analytical two dimensions



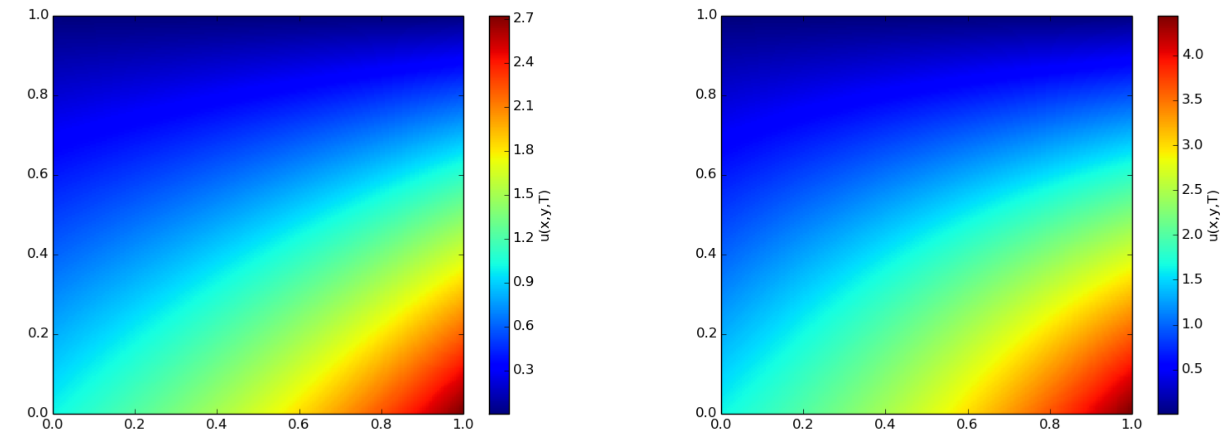
time=0.01s time=0.5s

As presumed the plots have a better flow to them, the solutions at different time-values look pretty much exactly the same, but the concentration is much larger as the time increses(see the colorbars). The concentration is here initiated at 1 since $u(0,0,0) = 1$.

6.4 Explicit scheme

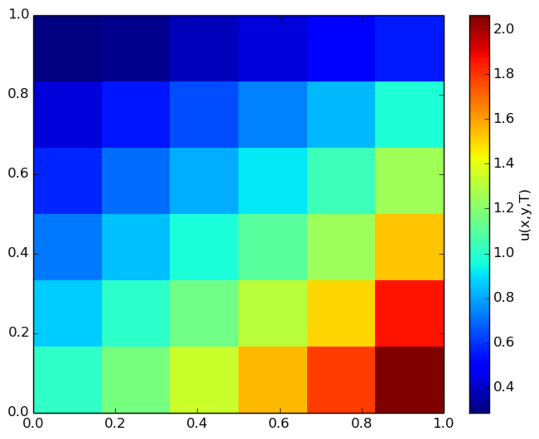
Here we will plot the explicit scheme with the stability criteria sustained and also when it is not.

Figure 11: Explicit scheme $\Delta t = 0.000005$

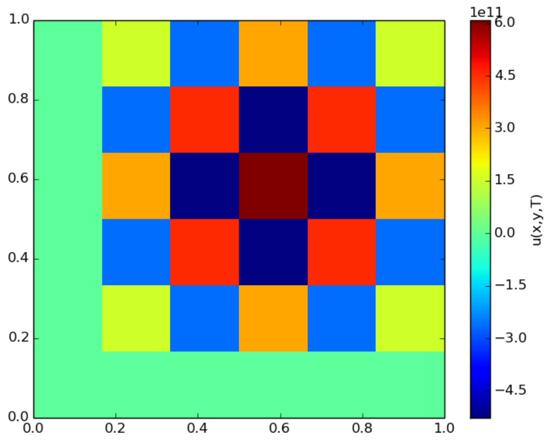


time=0.01s time=0.5s

Figure 12: Explicit scheme $\Delta t = 0.005$

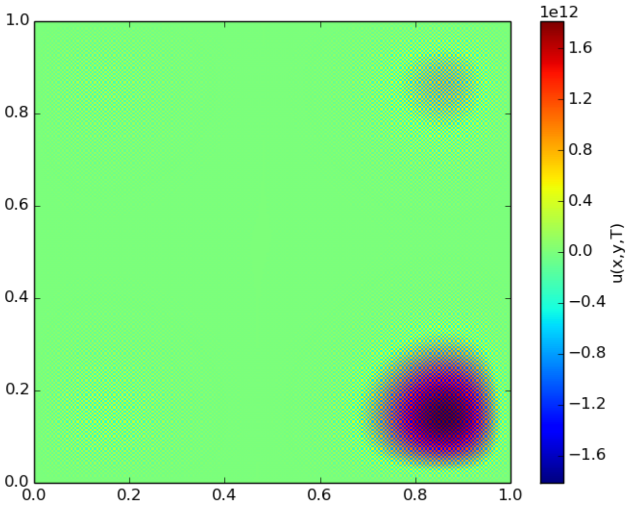


time=0.01s



time=0.5s

Figure 13: Explicit scheme unstable

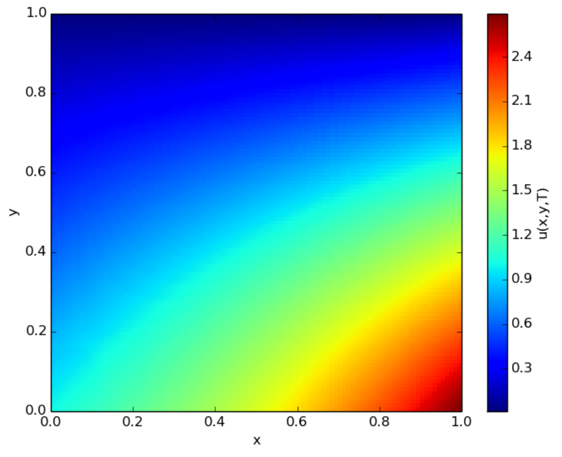


As long as the stability criteria is sustained the explicit scheme works well for low values of Δt, but the moment the stability criteria is not upheld the solution gets ridiculously unstable.

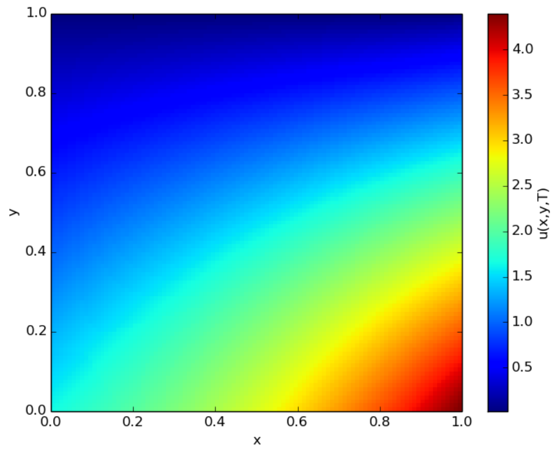
6.5 Implicit scheme

The implicit scheme will always be stable for any combination of Δt and Δx. The results look like these.

Figure 14: Implicit scheme $\Delta t = 0.00005$

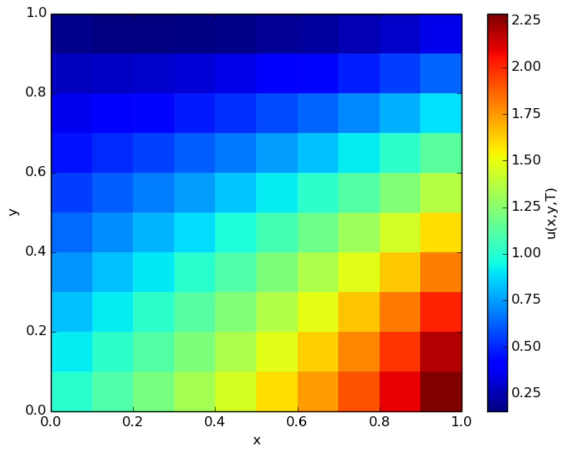


time=0.01s

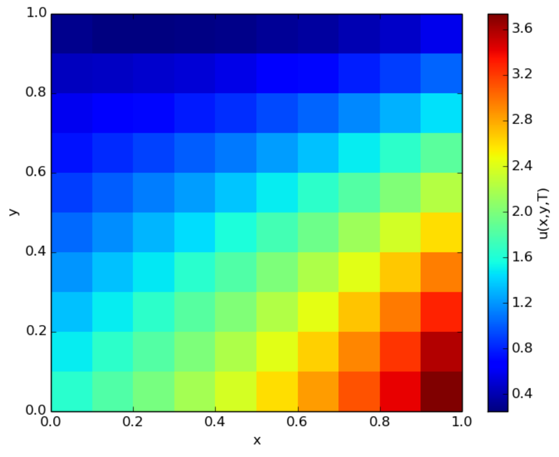


time=0.5s

Figure 15: Implicit scheme $\Delta t = 0.005$



time=0.01s



time=0.5s

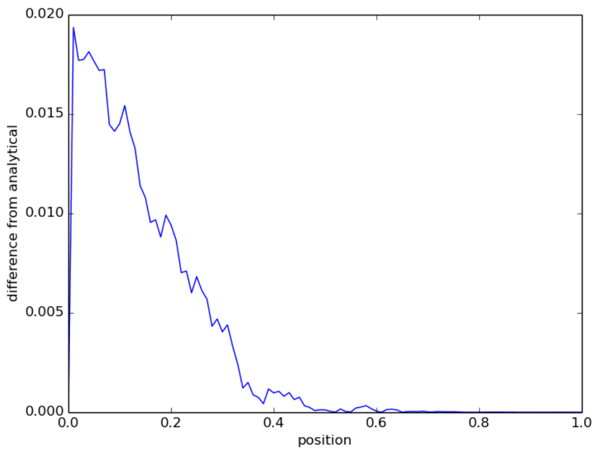
As we can see the implicit scheme is still very much stable.

7 Error estimates

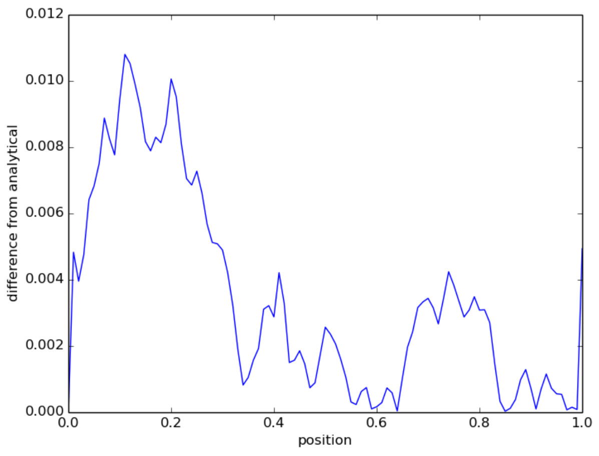
This section will cover error estimates for the different methods and schemes used in the sections above.

7.1 Error random walk 1D

Figure 16: Error random walk model 10^6 walkers $\Delta t = 0.00005$

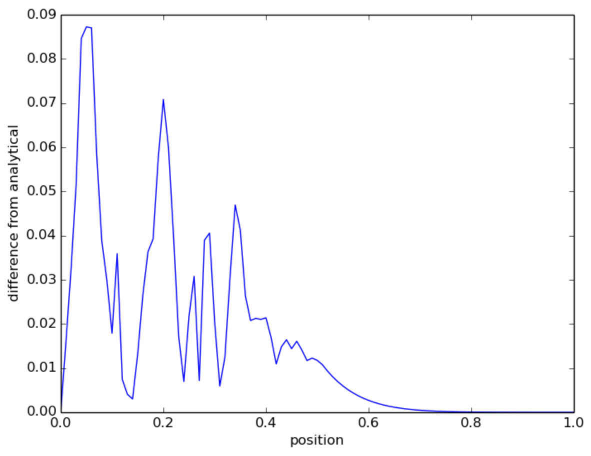


time=0.02s

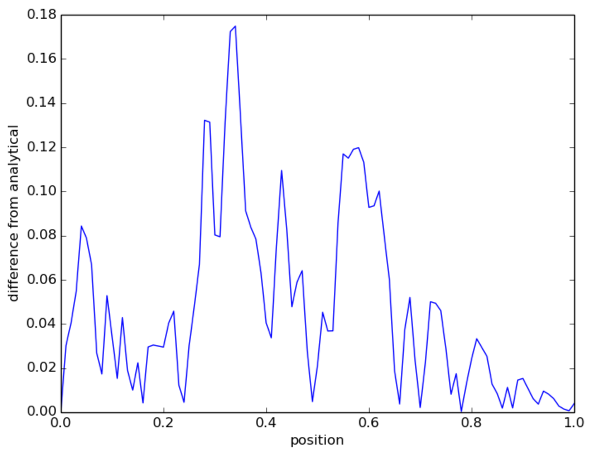


time=0.5s

Figure 17: Error random walk model 10^4 walkers $\Delta t = 0.0005$



time=0.02s

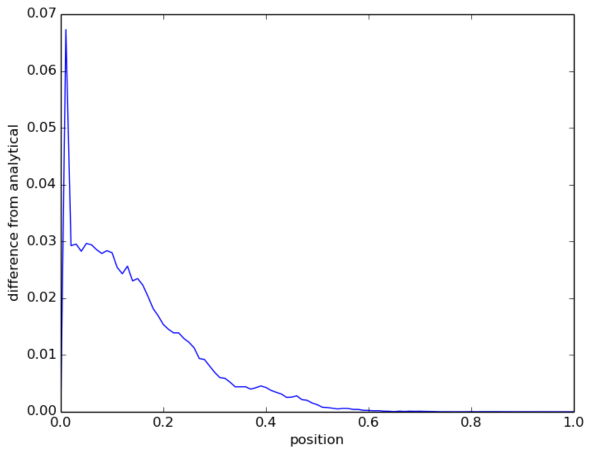


time=0.5s

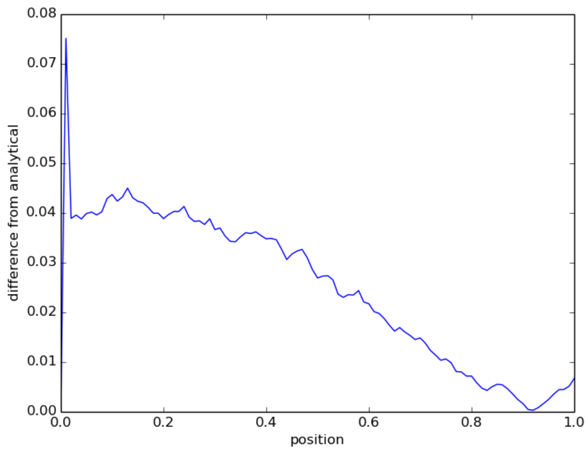
Here we observe that the error is considerably low if we have a large number of walkers, however the error increses as the total time increases. When the number of walkers is reduced the error gets very much present and we can see that for $T = 0.5$ the maximum spike is up at 0.17 which is quite large.

7.2 Error random walk 1D with gauss

Figure 18: Error random walk gauss 10^6 walkers $\Delta t = 0.00005$

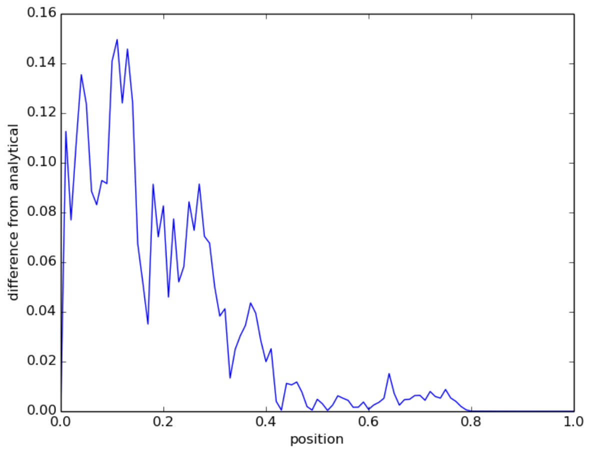


time=0.02s

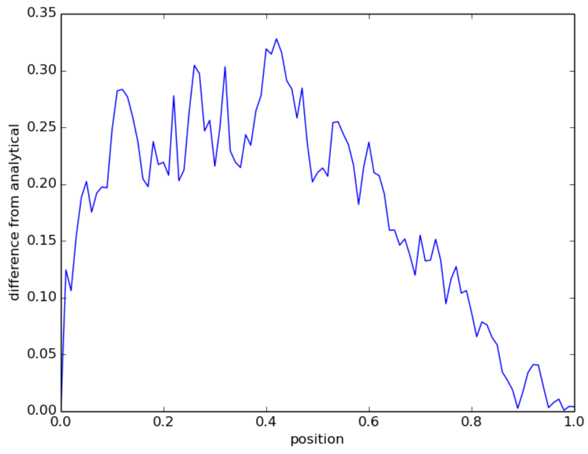


time=0.5s

Figure 19: Error random walk gauss 10^4 walkers $\Delta t = 0.0005$



time=0.02s

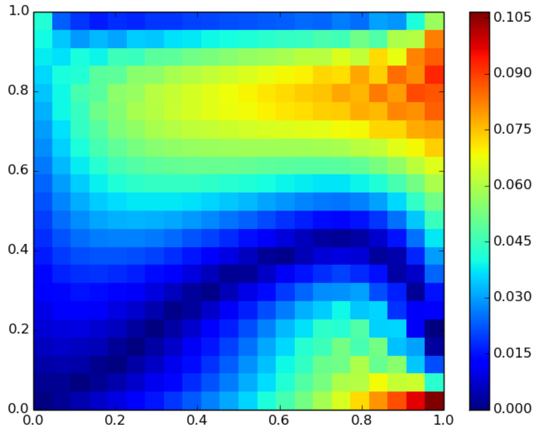


time=0.5s

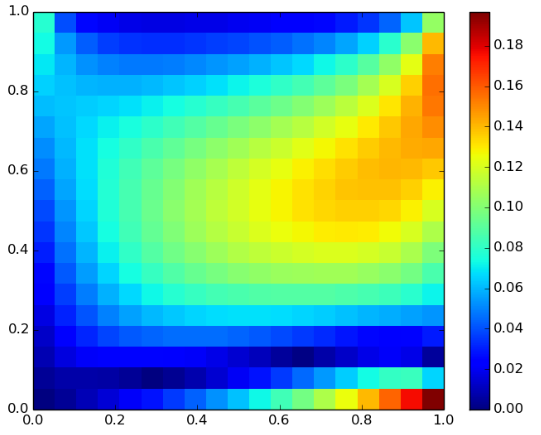
With the extra gaussian distribution the model actually gets worse. The error has a big spike in the beginning before decreasing, however the decrease is much slower than before and the overall error is still larger. An increased total time also increased the error here aswell. The natural presumption for the results with the gaussian model is that it should be better, we are introducing a distribution which should give more values around the desired criteria which in turn should give a better result. Also reducing the number of walkers obviously did not yield any better results.

7.3 Error explicit scheme

Figure 20: Error explicit scheme $\Delta t = 0.00005$



time=0.01s

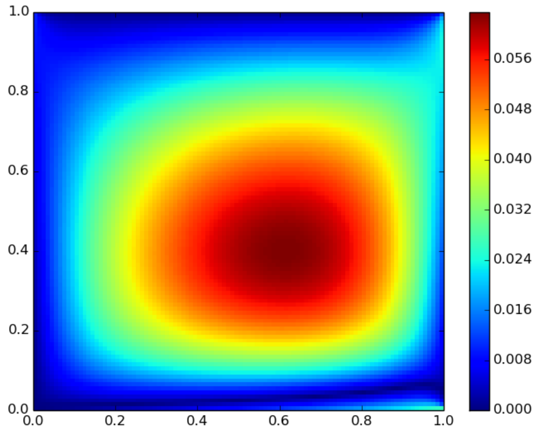


time=0.5s

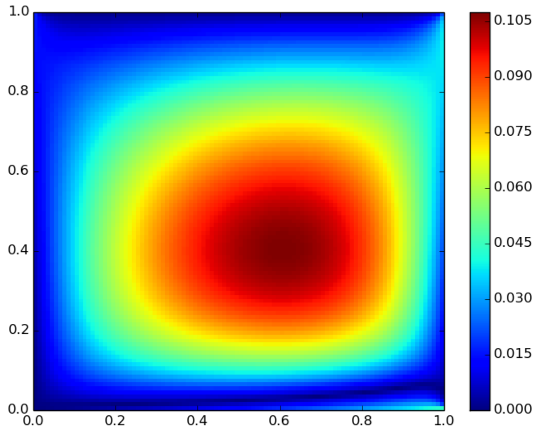
For the explicit scheme the error estimates are also done by checking the difference from the analytical solution. We can see that the error is large at the initial position but overall actually fairly good. For $T = 0.5$ the error is actually very large.

7.4 Error implicit scheme

Figure 21: Error implicit scheme $\Delta t = 0.00005$

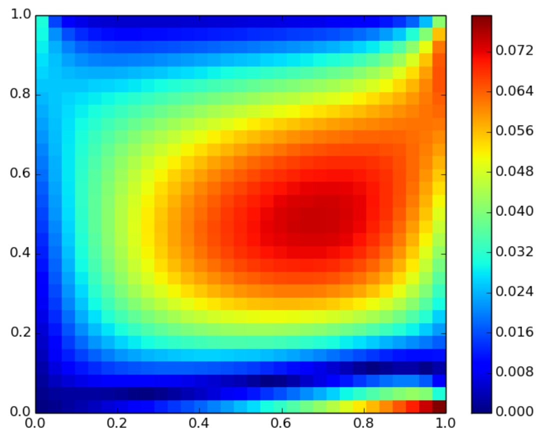


time=0.01s

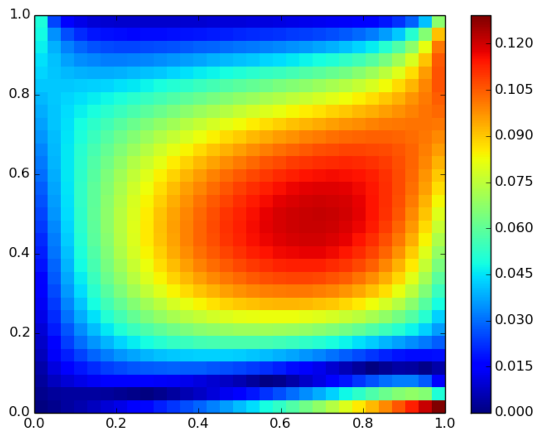


time=0.5s

Figure 22: Error implicit scheme $\Delta t = 0.0005$



time=0.01s



time=0.5s

The implicit scheme seems to be the favorable one, atleast when we look at the error estimates. The overall error is actually fairly low and the largest peak for $T = 0.5$ is 0.11, which is considerably lower than that of the explicit scheme. Note that the peak error stations itself in the middle, while the explicit method has errors everywhere. This behaviour might not be the most favorable for every situation, but atleast one can borderline the error and somewhat confine it. Increasing the time-step naturally gave a larger error.

8 Conclusion

This section is mainly for conluding the various results that were found in previous sections and to offer some critial views on the project itself.

8.1 Random walk model

The random walk model in 1D actually turned out to be somewhat applicable, however not good enough. Even with a large number of walkers and a low time-step the model couldn't reach the analytical solution very well. It worked considerably good for small time-values, but as the time increased towards the steady-state solution the graphs did not align very well. Besides, reducing the number of walkers also rendered the model mainly useless. On the positive side the model seemed to be fairly stable, altough it jumped around the analytical solution alot, it actually jumped around rather than «fly off into the sunset», that is get unstable and blow up.

We also observed that the gaussian distribution surprisingly resulted in worse results. The errors very larger, and it was slightly less stable aswell. This result may have many reasons, but the main concern is probably the fact that different seeds were not experimented with. The random number generator was also not the most optimal one giving raise to failure.

The random walk model was also compared to the explicit scheme(in one dimension). The problem with the explicit scheme from previous project was that it had a poor stability criteria, giving raise to instabilites. The stable plots were also a bit unstable as it «wobbled» since the time-step was to high. Still the explicit method was actually better than the random walk model with the gaussian distribution.

8.2 Random walk in two dimensions

In two dimensions the random walk turned out to be very good. The results gave a very intuitive look at the diffusion process. One could see the fluctuations and the concentration levels decrease as the particles moved away from the initial dispersion point.
The model also seemed very stable for all the tested time-values, including close to the steady-state. In addition reducing the number of walkers did not create any instabilites, the only concern was that the process itself was not replicated correctly. There simply weren't enough walkers to represent all the particles so the resulting plot was an «incomplete» representation of the diffusion process.

Reducing the time-step naturally resulted in poor resolution of the plot.

The error in the implementation was not taken into account in this project because the boundary conditions used for the random walk model did not have any analytical solution. For a more realistic problem this method needs more testing and alot more experimenting in order to keep the stability under check. With that said, the method itself proved to be very stable giving raise to higher probability of promising results. Plus the plots look really cool!

8.3 Explicit and implicit schemes

The explicit scheme with the forward Euler formula actually gave good results, aslong as the stability criteria was under control. The problem with the method was that choosing a larger time-value with a low time-step gave very unstable results and the moment the stability criteria wasn't archieved, the values blew up.
The explicit scheme is very simple to implement, but the poor stability condition deems it useless in most cases. In the situation of the diffusion equation the scheme seems viable, but the implicit method showed itself to be better. Both the error estimates were better for the implicit method compared to the explicit one. The most alarming fact about the explicit scheme was that the error was both large and distributed. The errors for the implicit scheme centered itself while the explicit had some error at the initial point while it decreased as one moved out and then it increased again. Giving raise to some concerns about instabilites.
The implicit scheme on the other hand gave some very good results as mentioned. The error was lower and it was very much stable. The concern about the method is however that it has a slow convergence criteria making it fairly slow scheme, but it too was easy to implement.
The essential part about choosing the implicit scheme over the explicit one is obviously the stability criteria, where the explicit scheme has a poor stability condition the implicit one is actually always(sort of) stable. The only problem is that it is actually fairly slow and has a convergence criteria that can be a hassle to handle for some situations. In case of the diffusion equation the convergence threshold proved to be very straight forward to implement.

9 Sources

Project from 2012
http://www.uio.no/studier/emner/matnat/fys/FYS3150/h12/undervisningsmateriale/projects/project-5%3A-diffusion-equation/project5_diffusion2012.pdf
The analytical solution for the two-dimensional equation was taken from here.

The project text
http://www.uio.no/studier/emner/matnat/fys/FYS3150/h14/undervisningsmateriale/projects/project-5-deadline-december-1/project5_diffusion.pdf
Heavily used for the theory part.

The lectur notes in computational physics by Morten Hjorth-Jensen
http://www.uio.no/studier/emner/matnat/fys/FYS3150/h14/undervisningsmateriale/Lecture%20Notes/lecture2014.pdf Chapter 10.2 and 10.3 for explicit and implicit scheme along with analytical solution to one-dimensional equation. Chapter 11.3 for randomnumber generator. Chapter 12.2 for random walk model.