# Fys3150–fall 2014: Project 5

Candidate 14

December 7, 2014

**Abstract**

In this project I continue an investigation of the diffusion process in neurotransmitters. Here I look at Monte-Carlo simulations using random walkers as particles. I look at simulations with both constant and gaussian distributed step lengths in one dimension and with gaussian distributed step lengths in two dimensions. I conclude that all these methods yield results that are consistent with the diffusion equation, but that they are computationally expensive. I look at the explicit forward Euler method and the implicit iterative Jacobi method of solving the two dimensional diffusion equation. I find that though the explicit method is fast and accurate it has a stability condition of $\Delta t \leq (1/4)\Delta x^2$ which may force you to choose a smaller timestep than needed. The implicit method is slower for a given timestep, but has no such requirement, meaning it may be used with much bigger timesteps than the explicit method may.

## 1 Introduction

In this project I continue the investigation of diffusion started in the previous project. There I looked at three ways of solving the one dimensional diffusion equation by the explicit forward Euler method, the implicit backward Euler method and the Crank-Nicholson method by applying them to the case of neurotransmitter diffusion. In this project I will model this system by the more intuitive Monte-Carlo (MC) style, letting each transmitter molecule be represented by a random walker. If the results of these simulations will correspond to the results found by solving the diffusion equation that will be a good indication that the diffusion equation gives a correct macro-description of the random micro-processes involved in diffusion.I will do this for both the one-dimensional case and a two dimensional case. For the two dimensional case, I will also look at two methods of solving the equation in this case: the two-dimensional explicit forward Euler method and the iterative Jacobi method. I will compare the solutions of the MC-cases and the numerical solutions of the partial differential equation (PDE) to the analytical solution of the PDE.

## 2 Physical problem

Neurons can communicate by diffusion of neurotransmitter molecules across a synapse. In this process the transmitters are released at the presynaptic membrane, flow across

the synaptic cleft and are absorbed by the postsynaptic membrane. The density of transmitter molecules at the presynaptic membrane is constant through the process, while the density at the postsynaptic membrane is constantly zero since the molecules are immediately absorbed. If one assumes that one is looking at the process far from the edges of the synaptic membranes the density should be constant in the plane paralell to the membranes, so the process may be well described by a one-dimensional simulation. I will also look at a two dimensional model where I include some direction along the membranes. I will only look at a segment of this second dimension that is of the same size as the distance between the membranes. Since this distance is much smaller than the width of the membranes, there should be nothing special about the boundaries of the second-dimensional segment. The obvious boundary conditions are then the periodic boundary conditions, representing the fact that particles are as likely to enter or exit the modeled segment of the second dimension. This should be a good model so long as we are looking at diffusion far from the edges of the synapse.

## 2.1 Monte-Carlo simulation

To simulate this process using a Monte-Carlo method I will treat each particle as a random walker. The root-mean-square of particles' displacement from its initial position after time $\Delta t$ in a diffusion process is given by

$$\sigma_x = \sqrt{(2D\Delta t)}, \tag{1}$$

where $D$ is the diffusion constant and $x$ is the displacement. I will simulate the one dimensional problem both by letting each particle move exactly $\sigma_x$ either forwards or backwards for each time step, and by choosing a steplength from a normal distribution so that the standard deviation of the distribution is $\sigma_x$.

## 2.2 Diffusion equation

The process may instead be described by the diffusion equation:

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

where $u$ is the density of neurotransmitters and D is the diffusion coefficient. The one dimensional version (letting x represent the first dimension) is

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}. \tag{3}$$

Immediately after the transmitters are released the concentration must be

$$u(x, t = 0) = N\delta(x),$$

where $N$ is the number of particles per membrane area and $\delta$ is the Dirac delta function. I will here set $N = 1$, which is simply a choice of units. Similarly i choose $D = 1$, and the width of the synaptic cleft is also 1. Now $u(0, t) = 1$ and $u(1, t) = 0$ so I have

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2},\ u(0, t) = 1,\ u(1, t) = 0,\ u(x, 0) = \delta(x). \tag{4}$$

For $t \to \infty$ the concentration will reach a steady state, so the time derivative will be 0. From equation 4 that state must be

$$u_s(x) = 1 - x. \tag{5}$$

Defining

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

we obtain the equation

$$\frac{\partial v(x,t)}{\partial t} = \frac{\partial^2 v(x,t)}{\partial x^2},\ v(0,t) = v(1,t) = 0,\ v(x,0) = \delta(x) + (x-1). \tag{6}$$

The two-dimensional case is essentially the same, the equation is:

$$\frac{\partial v(x,y,t)}{\partial t} = \frac{\partial^2 v(x,y,t)}{\partial x^2} + \frac{\partial^2 v(x,y,t)}{\partial y^2}, \tag{7}$$

with conditions:

$$v(0,y,t) = v(1,y,t) = 0,\ v(x,0,t) = v(x,1,t),\ v(x,y,0) = \delta(x) + (x-1).$$

# 3 Analytical solution

In appendix A I show that the analytical solution of equation 6 is

$$v(x,t) = \sum_{n=1}^{\infty} -\frac{2}{n\pi} \sin(n\pi x) e^{-(n\pi)^2 t}. \tag{8}$$

From this it is simple to find the solution in two dimensions. From equation 7 the initial state of $v$ is independant of $y$. The periodic boundary conditions offer no restraint on the evolution of $v$ in the $x$-direction, so it is clear that the solution will be independent of $y$. Then equation 7 is reduced to equation 6 and the solution is equation 8. The solution at t = 0.02 is plotted in figure 1.

# 4 Algorithms

## 4.1 Monte-Carlo

The algorithm for the MC simulations are simply random walks with some tests to satisfy the boundary conditions. Suitible tests in this case are:

- if a particle moves to $x \geq 1$, remove it from the simulation
- if a particle moves from $x = 0$, add a new particle at zero
- if a particle moves to $x \leq 0$, remove it

. For the two dimensional case the additional tests to account for the periodic boundary conditions of y are

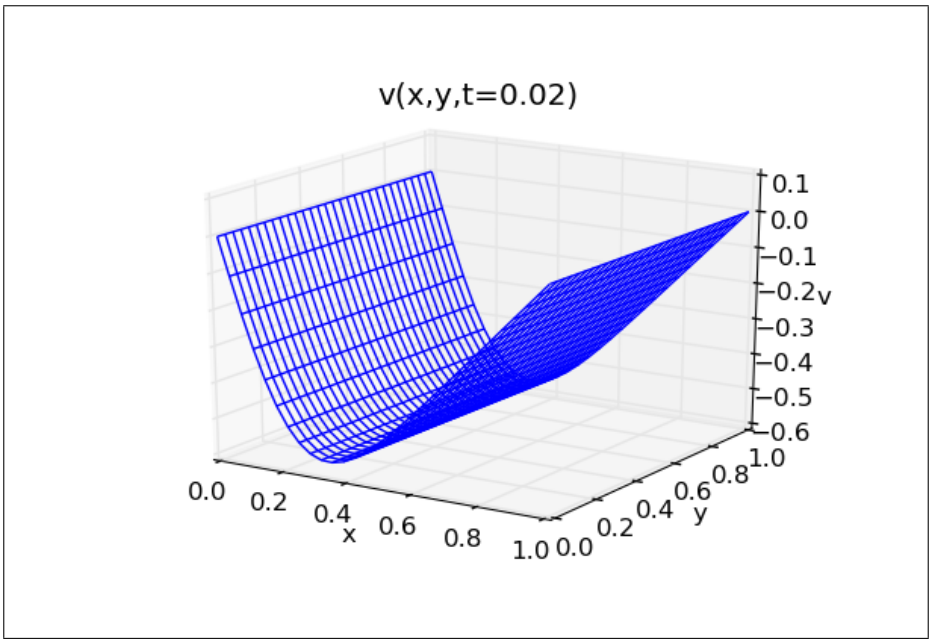


Figure 1: Analytical solution at time t = 0.02.

- if a particle moves to $y < 0$, place it in $y + 1$
- if a particle moves to $y > 1$, place it in $y - 1$

This algorithm is implemented in the programs `const.cpp`, `gauss.cpp` and `2dgauss.cpp` (found, along with the other program files and makefiles at `https://github.com/khhelland/comp_project_5`).

## 4.2 Explicit method

To discretize the two dimensional diffusion equation one may approximate the time derivative by the forward two point formula and both the spatial derivatives by the three point formula. Doing so results in the equation (in discretized notation $u(x_i, y_j, t_k) = u^k_{i,j}$):

$$\frac{u^{k+1}_{i,j} - u^k_{i,j}}{\Delta t} = \frac{u^k_{i+1,j} - 2u^k_{i,j} + u^k_{i-1,j}}{\Delta x^2} + \frac{u^k_{i,j+1} - 2u^k_{i,j} + u^k_{i,j-1}}{\Delta y^2}$$

Letting $\Delta x = \Delta y = h$:

$$u^{k+1}_{i,j} = u^k_{i,j} + \frac{\Delta t}{h^2}\left(u^k_{i+1,j} + u^k_{i-1,j}u^k_{i,j+1} + u^k_{i,j-1} - 4u^k_{i,j}\right)$$

Defining $\alpha = \Delta t/h^2$ and $\Delta^k_{i,j} = u^k_{i+1,j} + u^k_{i-1,j}u^k_{i,j+1} + u^k_{i,j-1}$ we obtain

$$u^{k+1}_{i,j} = (1 - 4\alpha)u^k_{i,j} + \alpha\Delta^k_{i,j}. \tag{9}$$

This equation is an explicit way of updating $u$ to simulate the diffusion process.

By trial and error it seems that the stability condition on this method is $\alpha \leq 1/4$. This makes sense, as it is in some sense a combination of two one-dimensional forward

Euler methods. In project 4, I showed that that method has a stability condition $\alpha \leq 1/2$, meaning that the error amplification factor is $2\alpha$. In the two dimensional case then, the factor is $2 \times 2\alpha$. Requiring that the factor is less than one leads to the observed stability condition $\alpha \leq 1/4$.

### 4.3 Implicit method

If we use the backward two point method for approximating the time derivative in the diffusion equation rather than the forward as above we obtain the following instead of equation 9

$$(1 + 4\alpha)u_{i,j}^{k+1} - \alpha\Delta_{i,j}^{k+1} = u_{i,j}^{k}, \tag{10}$$

where the symbols mean the same as in section 4.2. Equation 10 implicitly defines $u$ after one timestep as the matrix that obeys the equation. To find this matrix one may use Jacobi's iterative method. This method is based on guessing the solution, then updating the guess. Letting $l$ represent the guess-number, one finds the next guess by rearanging equation 10 to

$${}^{l+1}u_{i,j}^{k+1} = \frac{1}{1 + 4\alpha}\left(\alpha({}^{l}\Delta_{i,j}^{k+1}) + u_{i,j}^{k}\right) \tag{11}$$

This operation is performed until ${}^{l+1}u_{i,j}^{k+1} \approx {}^{l}u_{i,j}^{k+1}$. In that case, $u$ aproximately obeys equation 10. The convergence condition for this method is that if one rewrote equation 10 as a matrix equation $Au_{k+1} = u_k$ the matrix A is positive definite. In this case the matrix would be positive definite, so the method will converge on the correct solution.

Since this method is based on an implicit definition of the matrix at the next timestep there is no clear error amplification factor, and one should expect the method to be stable for all $\alpha$. By testing the algorithm for different values of $\alpha$ I could not find one that made the method unstable, so I conclude that it is stable for all values of $\alpha$.

This method, along with the explicit method, is implemented in the appended program `2dpde.cpp`.

## 5 Results

### 5.1 Monte-Carlo simulations

#### 5.1.1 1D constant step

Figures 2 and 3 show normalized histograms of the positions of the particles in the one dimensional constant step MC simulations at times T = 1 and T = 0.02 together with plots of the analytical solution for $u(x)$. The simulations were run with $10^4$ particles at $x = 0$. The histograms are normalized by setting the value of the first bin to 1.

#### 5.1.2 1D gaussian step

Figures 4 and 5 show normalized histograms of the positions of the particles in the one dimensional gaussian step MC simulations at times T = 1 and T = 0.02 together with plots of the analytical solution $u(x)$. The simulations were run with $10^4$ particles at $x = 0$. The value of the first bin is much greater than in the others for both times. This

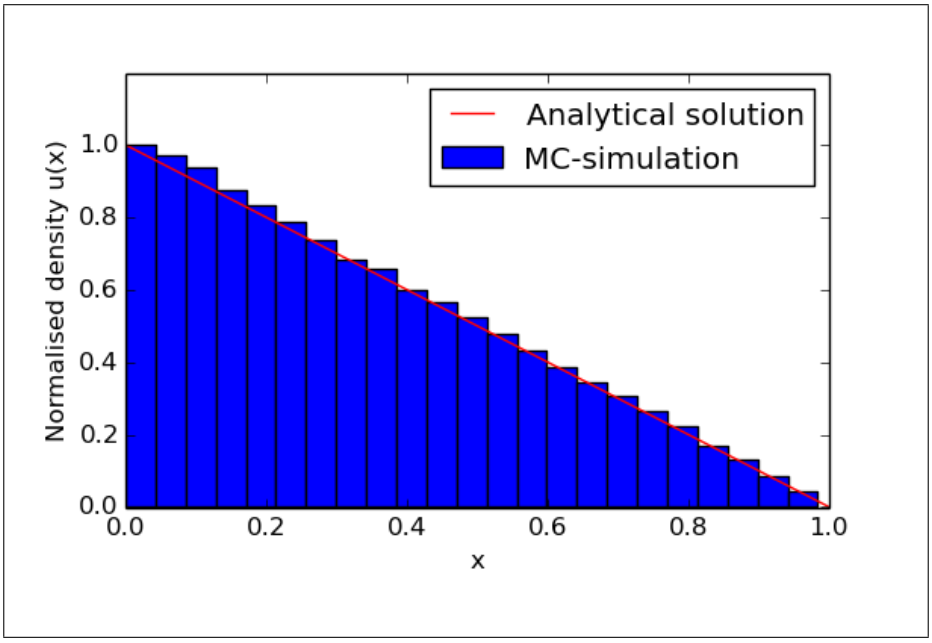


Figure 2: Normalized histogram of results of 1D constant step MC simulation after T = 1, plotted with the analytical solution u(x) of equation 6 at this time.

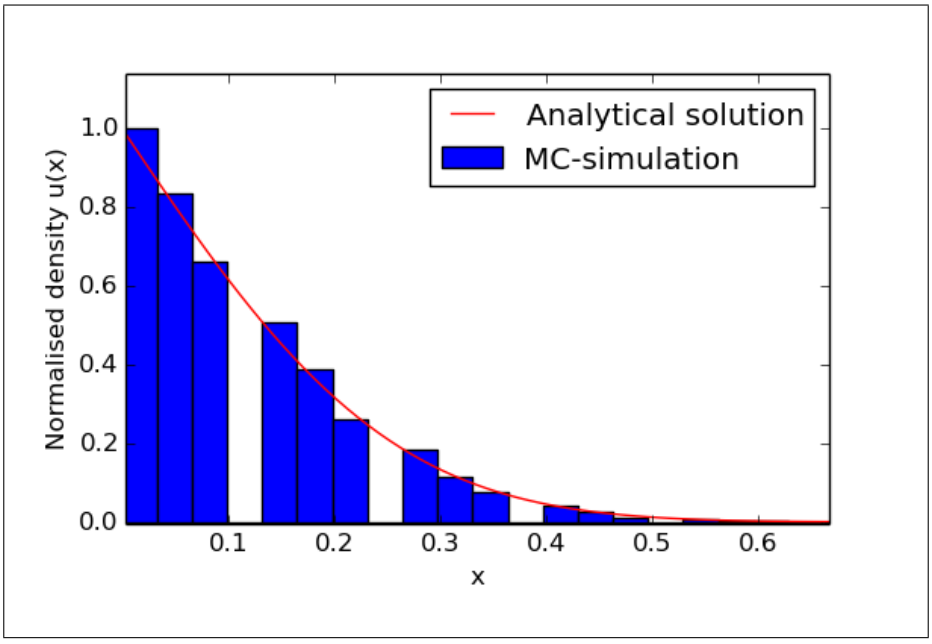


Figure 3: Normalized histogram of results of 1D constant step MC simulation after T = 0.02, plotted with the analytical solution u(x) of equation 6 at this time.

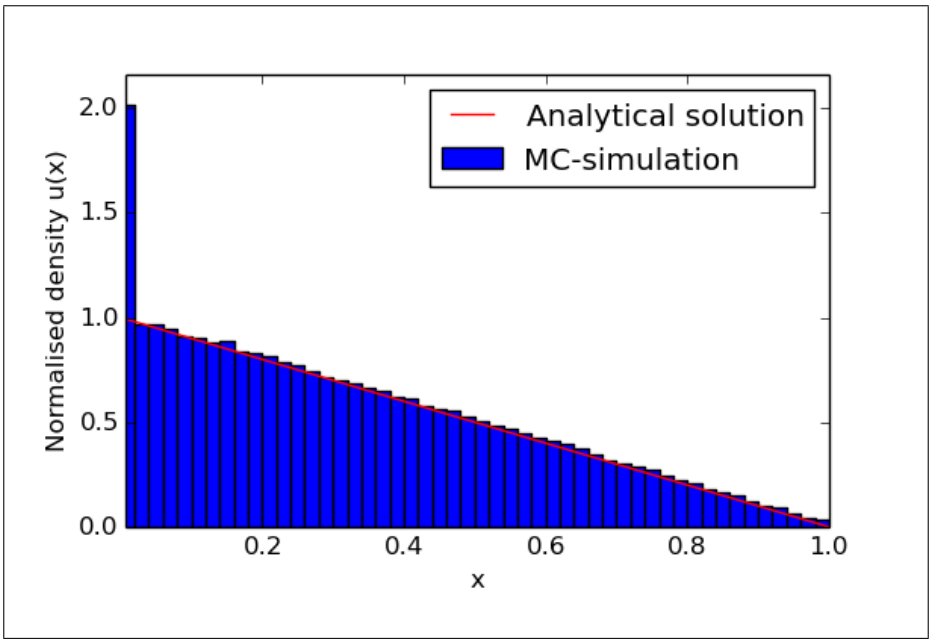


Figure 4: Normalized histogram of results of 1D gaussian-step MC simulation after T = 1, plotted with the analytical solution u(x) of equation 6 at this time.

is probably due to the choice of bins and the fact that there will be $10^4$ particles within $10^{-14}$ of 0, while there will also be particles in the rest of bin one. Because of this, I chose to normalize the histograms by forcing the value of the second bin to take the value of the analytical solution for $u(x)$ at the middle of this bin.

### 5.1.3 2D gaussian step

Figures 6 and 7 show normalized wireframe "histograms" of the positions of the particles in the two dimensional gaussian step MC simulations at times T = 1 and T = 0.02 together with plots of the analytical solution $u(x, y)$. The simulations were run with $10^4$ particles at $x = 0$ spread out over the y-segment. There is the same issue with the first bin in the x-directions here as with the one dimensional gaussian step, so the histograms are normalized by requireing that the second bin in the x-direction and the second bin in the y-direction match the value of the analytical solution in the middle of this bin. (The bin-number in the y-direction was chosen arbitrarily, and may have affected the plots).

### 5.1.4 Cost

The time it took to run each of the three MC simulations with time step $dt = 10^{-3}$ and total time $T = 1$ and $10^4$ initial particles is found in table 1.

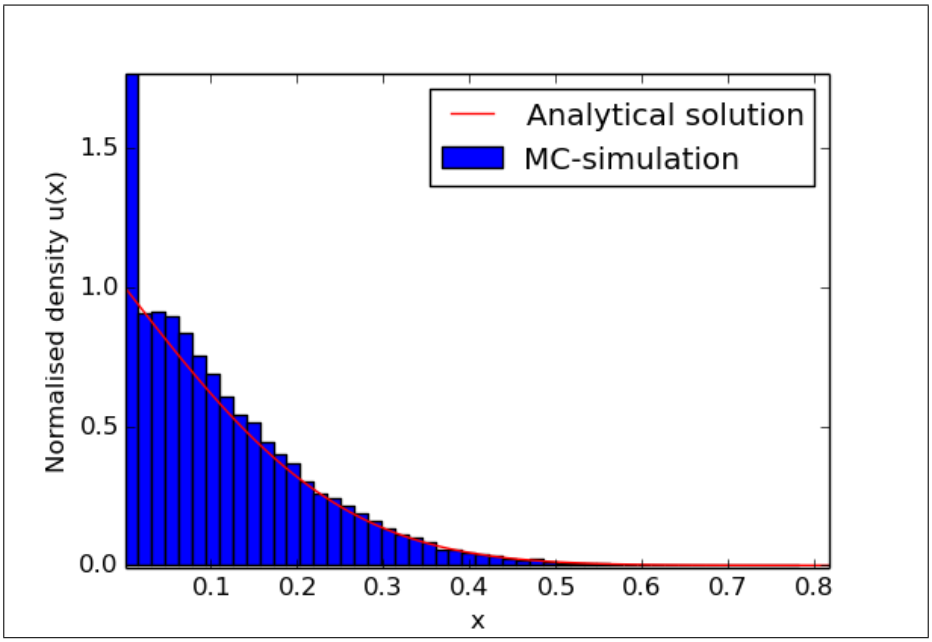


Figure 5: Normalized histogram of results of 1D gaussian-step MC simulation after T = 0.02, plotted with the analytical solution u(x) of equation 6 at this time.

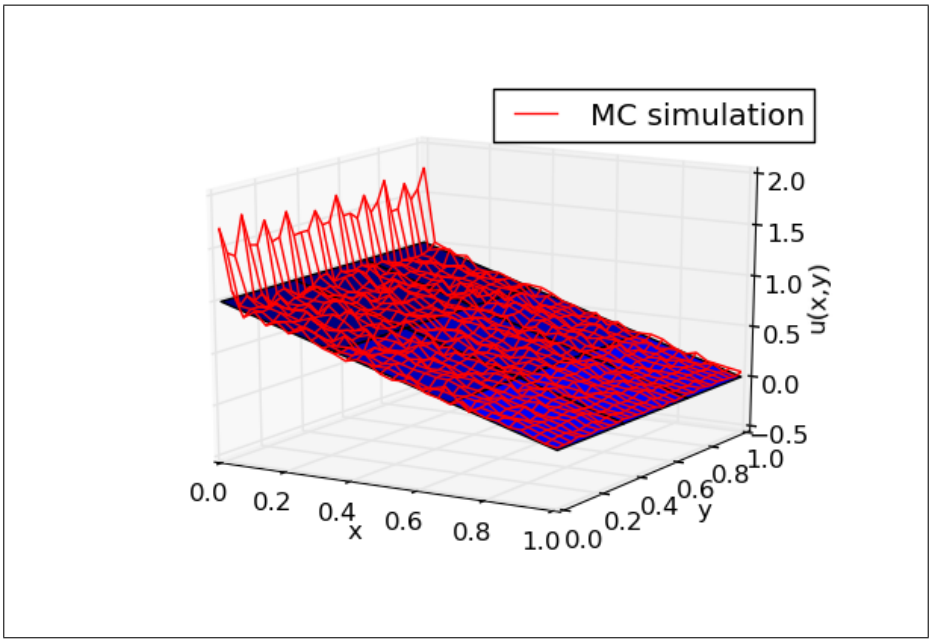


Figure 6: Normalized histogram (red wireframe) of results of 2D gaussian-step MC simulation after T = 0.02, plotted with the analytical solution (blue surface) u(x) of equation 7 at this time.

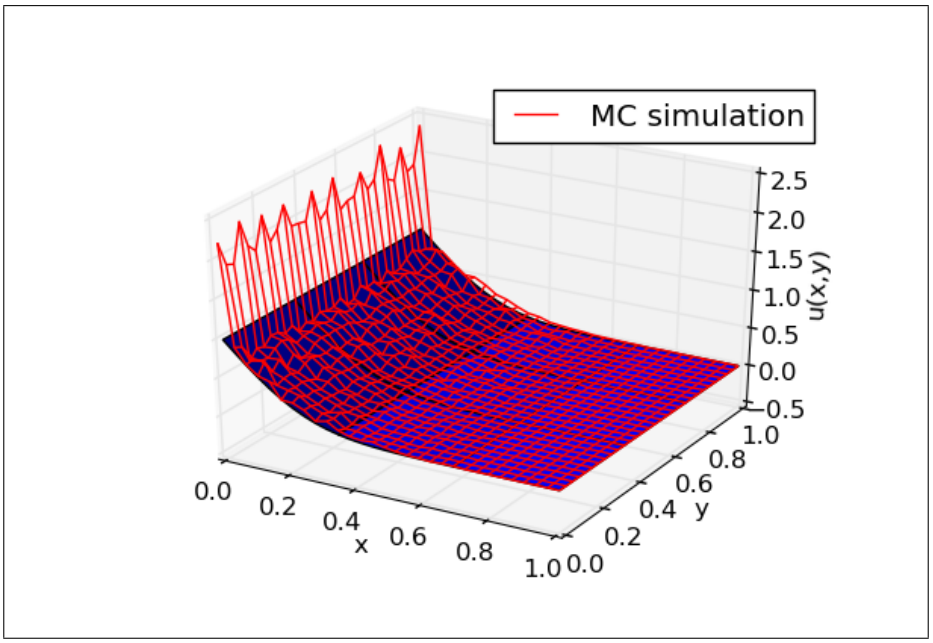


Figure 7: Normalized histogram (red wireframe) of results of 2D gaussian-step MC simulation after T = 0.02, plotted with the analytical solution (blue surface) u(x,y) of equation 7 at this time.

Table 1: Table of time used by Monte-Carlo simulations with time step $dt = 10^{-3}$ and total time $T = 1$ and $10^4$ initial particles.

| | Time [s] |
|---|---|
| 1D const-step | 1.77 |
| 1D gauss-step | 6.53 |
| 2D gauss-step | 24.4 |

Table 2: Table of maximal absolute and relative errors in solutions by the `2dpde.cpp` program with the explicit forward Euler method with $\alpha = 1/4$ and the implicit iterative Jacobi method with $\alpha = 1/4$ and $\alpha = 1$ all using $(30 \times 30)$-matrices compared to the analytical solution with the first 3000 terms of the sum included.

| | Maximal errors | | | | | |
|---|---|---|---|---|---|---|
| | Explicit method | | Implicit method | | Implicit method ($\alpha = 1$) | |
| Time | absolute | relative | absolute | relative | absolute | relative |
| 0.02 | 4.34e-3 | 1.27e-2 | 2.40e-3 | 4.26e-3 | 6.04e-3 | 2.44e-2 |
| 0.5 | 5.33e-05 | 2.60e-2 | 1.28e-4 | 4.03e-2 | 1.88e-4 | 4.82e-2 |

Table 3: Time used by each algorithm with N space points in both directions and total time 1. The explicit forward Euler method was run with $\alpha = 1/4$ and the implicit Jacobi method was run with $\alpha = 1/4$ and $\alpha = 1$.

| | Time [s] | | |
|---|---|---|---|
| $N$ | Explicit | Implicit | Implicit($\alpha = 1$) |
| 10 | 2.71e-4 | 1.21e-3 | 8.58e-4 |
| 30 | 1.87e-2 | 9.03e-2 | 6.60e-2 |
| 50 | 8.80e-2 | 6.14e-1 | 5.48e-1 |
| 100 | 1.32e0 | 8.14e0 | 5.53e-0 |
| 200 | 2.51e1 | 1.29e2 | 7.78e1 |

## 5.2 Diffusion equation

### 5.2.1 Accuracy

The maximal relative and absolute errors in the solutions by the `2dpde` program compared to the analytical solution with the first 3000 terms included at times 0.02 and 0.5 are found in table 2. These errors are based on using 30 space points in each direction.

### 5.2.2 Cost

To test the speed of the algorithms i ran the `2dpde.cpp` program with up to $T = 1$ for $N_x = 10, 30, 50, 100, 200$ space points. I used $\alpha = 1/4$ for the explicit algorithm as this is the stability requirement. For the implicit Jacobi algorithm I used $\alpha = 1/4$ to test the algorithm against the explicit one for the same number of time steps, however one of the strengths of this algorithm compared to the explicit one is precisely that one is free to choose $\alpha$, and may use a larger time step. To see how choice of $\alpha$ affected the speed I also ran the implicit method with $\alpha = 1$. The time values are found in table 3.This table shows that for the same time step the explicit method is significantly faster than implicit, and that the implicit method with larger timesteps falls somewhere in between the others.

# 6 Discussion

## 6.1 MC-methods

### 6.1.1 Accuracy

Figure 2 shows that 1D constant step MC simulation yielded results that fit very well with the analytical steady state solution. The histogram in figure 3 at a time with significant change in the state also fits the analytical solution quite well, although there are some empty spaces in the histogram that are probably mainly due to the choice of bins in the histogram.

Apart from the first bin, where the density appears to be twice the expected value, figure 4 shows that the gaussian step simulation also fits the steady state analytical solution very well. For the more curved solution at t = 0.02, figure 5 shows that the histogram fits the curvature quite well, but seems to be consistently too high-valued. This may be explained by noting that the bin-value I chose to normalize the histogram is lower than the next value, which is unexpected. If this is an error it would make the other values of the histogram too high and explain the discrepancy. I have no explanation for this value beeing lower than the next.

The two dimensional case is harder to visualize in a meaningful way, and the included figures 6 and 7 are not too informative. Looking at them it is clear that there is greater variation between the bins in theese cases than in the 1D cases. This is not surprising as there are many more bins per particle in the 2D histograms. Despite the variations it is clear that the histograms follow the analytical solutions quite closely for both the included times.

In all these cases it seems that the MC-simulations have yielded results that are very similar to those obtained from solving the diffusion equation, showing that these methods describe the same type of processes.

### 6.1.2 Cost

Because it is difficult to obtain quantitative measures of the accuracy of these methods without running them many many times with different seeds it is not obvious how to compare these times to the ones for the PDE solvers. Nevertheless the qualitative results of the plots in section 5.1 suggest that the accuracy for the time steps and number of particles used is comparable to the accuracy of the PDE solvers with quite few spacial steps. In that case it is clear from tables 1 and 3 that the MC methods are very computationally expensive.

## 6.2 Diffusion Equation

### 6.2.1 Accuracy

Table 2 shows that at $t = 0.02$ the implicit method with $\alpha = 1/4$ is the most accurate, though the difference between the methods is not very large. The absolute errors are here all of order $10^{-3} \sim h^2$. For $t = 0.5$ the explicit method is the most accurate. We also see that the absolute error is much smaller than the relative error here, meaning that the value of $v$ is probably very small at this time. Looking at the analytical solution 8 one sees that

the smallest of the exponential suppression factors at this time is $\exp(-\pi^2/2) \sim 10^{-3}$. The implicit method is based on considering two matrices equal if the mean difference between the two is smaller than some value. In my implementation this value is $10^{-7}$. Allthough this value is much smaller than $10^{-3}$ it might be close to the value of $v$ near the edges. In this case the solution will probably not converge further at these points, which may explain the large errors at $t = 0.5$. The implicit method with $\alpha = 1$ is the least accurate for both time points, but the errors are still not that much greater than for the other methods.

### 6.3 Comparison of MC and PDE methods

Looking at the cost and accuracy of all the different simulations it seems that the 2D PDE solvers came much closer to the analytical solution of the equation than the MC simulations did, even for only a few space steps. Assuming that the analytical solution represents the *true* process, we can conclude that using PDE solvers are the more accurate method. Comparing the time cost of the 2D MC simulation ($\sim 10^1$ seconds) with the time cost of the implicit Jacobi PDE solver for $30 \times 30$ space points ($\sim 10^{-1}$ seconds) it is clear that using PDE solvers are also much faster. One virtue of the MC-methods over the PDE-solvers are that they consist of many completely independantly proceses, and thus may be paralelized very simply and efficiently, perhaps so much that a very large MC-simulation may end up being more efficient than a PDE solver.

## 7 Conclusion

The point of including MC simulations in this project was that such simulations bear a closer resemblance to the physical case studied than PDEs, and are more intuitive. The problem with these types of simulation seems to be first and foremost cost of computing power, which was much greater for the MC-simulations than the simulations of the PDEs. Comparing the results of the MC-simulations with the solutions to the diffusion equation has shown that the diffusion equation does in fact describe the diffusion process quite well, and that one may use this equation to model diffusion in a good way.

In choosing the method of solving the equation there is no obvious "winner". If the accuracy demands are such that it is reasonable to use a time step that is smaller than $h^2/4$ anyway, the explicit method will a good choice as this is fast and accurate. If speed is important and one may use a time step that is much larger than this it would probably be better to use the implicit method with a large time step, as the results will still be reasonable, even if the accuracy will not be great.

## A Analytical solution in one dimension

I found the analytical solution of equation 6 in project 4, the derivation is included here. First I assume that the solution is seperable so that

$$v(x,t) = F(x)G(t).$$

Now equation 6 says

$$\frac{1}{F}\frac{\mathrm{d}^2F}{\mathrm{d}x^2} = \frac{1}{G}\frac{\mathrm{d}G}{\mathrm{d}t}$$

Since the sides of the equation are independent of each other, each must be a constant. By naming this constant $-\lambda^2$ we obtain the equations

$$\frac{\mathrm{d}^2F}{\mathrm{d}x^2} = -\lambda^2 F,$$

with solution

$$F = A\sin(\lambda x) + B\cos(\lambda x),$$

and

$$\frac{\mathrm{d}G}{\mathrm{d}t} = -\lambda^2 G,$$

with solution

$$G = Ce^{-\lambda^2 t}.$$

From the boundary conditions on $v$, we see that $B = 0$ and $\lambda = n\pi$ for some integer $n$. Letting $A_n = A(n)C(n)$ the general solution of 6 is

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

To find $A_n$

$$v(x,0) = \sum_{n=1}^{\infty} A_n \sin(n\pi x)$$

$$\sin(m\pi x)v(x,0) = \sum_{n=1}^{\infty} A_n \sin(n\pi x)\sin(m\pi x),$$

where $m$ is an integer.

$$\int_0^1 \sin(m\pi x)v(x,0)\,\mathrm{d}x = \int_0^1 \sum_{n=1}^{\infty} A_n \sin(n\pi x)\sin(m\pi x)\,\mathrm{d}x.$$

I assume that it will be allright to take the sum after evaluting the integral so that

$$\int_0^1 \sin(m\pi x)v(x,0)\,\mathrm{d}x = \sum_{n=1}^{\infty} A_n \int_0^1 \sin(n\pi x)\sin(m\pi x)\,\mathrm{d}x$$

$$\int_0^1 \sin(m\pi x)v(x,0)\,\mathrm{d}x = \sum_{n=1}^{\infty} A_n \frac{\delta_{m,n}}{2}$$

$$\int_0^1 \sin(m\pi x)v(x,0)\,\mathrm{d}x = \frac{A_m}{2}.$$

So

$$\frac{A_n}{2} = \int_0^1 \sin(n\pi x)(\delta(x) + (x-1))\,\mathrm{d}x$$

$$\frac{A_n}{2} = sin(n\pi 0) + \int_0^1 \sin(n\pi x)(x-1)\,\mathrm{d}x$$

$$A_n = 2\left(\frac{\sin(n\pi) - n\pi}{(n\pi)^2}\right) = -\frac{2}{n\pi}.$$

And the solution becomes equation 8

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