Master Thesis

Department of physics/Stockholm University
and
NORDITA

Studies of Confined Brownian Motion using the Finite Difference Method

Liam Sebestyén

Supervisor:
Ralf Eichhorn

Co-Supervisor:
Supriya Krishnamurthy

February 2, 2018

Contents

Abstract

In this thesis the diffusive motion of a Brownian particle with a driving force in two dimensions and in the presence of hard walls is studied by numerically solving the Fokker-Planck equation using the finite difference method. The task is to develop a computer program and to use it to study a particle diffusing in a channel with a periodically varying cross section.

Chapter 1

Introduction

On scales of micrometer size and smaller, particles in an aqueous solution have a diffusive component in their motion coming from thermal fluctuation [1]. Faster moving water molecules hitting the particle leads to random displacements. This stochastic process is called Brownian motion.

Brownian motion was first studied by the Scottish botanist Robert Brown [2] in 1827. He was interested in the fertilization process of the then newly discovered plant *Clarkia pulchella*. Grains of pollen from the plant were suspended in water. He noticed that particles the size of 5-6 micrometer, that had been trapped inside the grains, performed an persistent oscillatory motion. He also studied inorganic particles and found the same type of motion. The process is named after him, although he was not able to explain it.

It was Albert Einstein who, in a paper 1905 [3], gave a theoretical description of the process. From statistical mechanics he showed that the random pressure differences coming from the collisions with the particles in the liquid would give rise to a random motion. He formulated the diffusion equation for a Brownian particle and also showed how the diffusion coefficient is related to the mean square displacement of a Brownian particle; the diffusion coefficient is one half of the mean square displacement. Marian Smoluchowski independently reached the same result, but published one year later [4].

In e.g. biological cells or artificial microfluidic devises, the Brownian motion is confined by hard walls in compartments or channel-like structures [6]. Consider a Brownian particle in a two dimensional channel with varying cross section with no applied force. This can be described by the Fick-Jacobs equation. This equation was derived by Merkel. H. Jacobs [7]. It approximates the system as a one dimensional periodic potential. The confinement of the particle gives rise to an entropic potential. The particle will have a higher probability of ending up where the cross section of the channel is the

largest. This probability effect is called an entropic force. This entropic force is also of interest in for example ion channels (proteins in cell membrane that allow ions to pass in or out of the cell) [8].

In this thesis the driven Brownian motion of a spherical shaped particle in two dimensions in the presence of hard walls is studied by numerically solving the Fokker-Planck equation using the finite difference method, with reflecting boundaries representing the hard walls. The task is to develop a computer program that is able to handle different structures of confining hard walls and to use it to study the diffusion of a particle in a channel with periodically varying cross section.

The Langevin equation, developed by Paul Langevin in 1908 [5], is a stochastic differential equation that describes the dynamics of a subset of degrees of freedom of the system. The Fokker-Plank equation describes the time evolution of the probability distribution of the particle. Solving the Fokker-Plank equation is a much more difficult approach from a computational perspective than solution of the Langevin equation using molecular dynamics or Monte Carlo simulations. There exist many algorithms for simulations of the Langevin equation for confined Brownian motion. Many of them contain hand waving arguments for dealing with the reflective boundaries [9]. With the program developed in this thesis, those methods can be checked.

An explanation of the theory will be given in section 1.1. In section 2.1 the finite difference method is explained. How the boundaries are going to be treated is discussed in section 2.2. The algorithm is presented and explained in section 2.3. The code should be easy to maintain and develop. Therefore an discussion of the design of the code is also included in section 2.3. Results for different test cases are shown in sections 3.1-3.4. The result for a particle in a periodic channel is shown in section 3.5.

1.1 Smoluchowski Equation

Consider a particle in a liquid. Suppose one would like to describe the motion of the particle. One way to do it is to write the Newtonian equation of motion for all N atoms in the system

$$m_i \frac{d^2 x_i}{dt^2} = -\frac{\partial}{\partial x_i} V(x_1, ..., x_N), \tag{1.1}$$

where m is the mass and x_i the position of atom i. Even in microscopic systems such as proteins the number of atoms are between 10^3 and 10^5, which makes the solution of (1.1) very computational expensive [10]. There

exists an alternative way. The problem can be described by a stochastic differential equation called the Langevin equation. It models a subset of degrees of freedom. Often it is the slow, macroscopic, degrees of freedom that are modelled, while the fast, microscopic, degrees of freedom are modelled by noise. For a Brownian particle with driven diffusive motion the Langevin equation is

$$m\ddot{\mathbf{x}} = -\gamma\dot{\mathbf{x}} + \mathbf{F}(\mathbf{x}) + \sigma\boldsymbol{\xi}(t), \tag{1.2}$$

where $\mathbf{x}$ is the position of the particle. The term to the left is the acceleration, where m is the mass of the particle. The first term to the right is the frictional forces coming from Stokes' law, where γ is the scalar friction constant. The second term is the external force. The last term is the time dependent fluctuating force, where σ is the amplitude of the fluctuations and $\boldsymbol{\xi}(t)$ is a Gaussian white noise,

$$\langle \xi_i(t) \rangle = 0, \tag{1.3}$$

$$\langle \xi_i(t_1)\xi_j(t_0) \rangle = \delta_{ij}\delta(t_1 - t_0). \tag{1.4}$$

The generic Langevin equation, for which (1.2) is a special case, can been derived from classical mechanics [11].

In this thesis the strong friction limit (the force of inertia is much smaller than the frictional force) will be considered,

$$|\gamma\dot{\mathbf{x}}| >> |m\ddot{\mathbf{x}}|. \tag{1.5}$$

With (1.5) the Langevin equation (1.2) becomes,

$$\gamma\dot{\mathbf{x}} = \mathbf{F}(\mathbf{x}) + \sigma\boldsymbol{\xi}(t), \tag{1.6}$$

The Fokker-Planck equation corresponding to (1.6) is

$$\frac{\partial p(\mathbf{x}, t|\mathbf{x}_0, t_0)}{\partial t} = \left(\nabla^2 \frac{\sigma^2}{2\gamma^2} - \nabla \cdot \frac{\mathbf{F}(\mathbf{x})}{\gamma} \right) p(\mathbf{x}, t|\mathbf{x}_0, t_0), \tag{1.7}$$

where $p(\mathbf{x}, t|\mathbf{x}_0, t_0)$ is the probability to find a particle at $\mathbf{x}$ at time t if it started at $\mathbf{x}_0$ at time t_0. The Fokker-Planck equation for the strong friction limit is called the Smoluchowski equation. With the diffusion coefficient $D = \frac{\sigma^2}{2\gamma^2}$ and the drift $\mathbf{v}(\mathbf{x}) = \frac{\mathbf{F}(\mathbf{x})}{\gamma}$ and assuming the parameters are spatially independent, equation (1.7) becomes

$$\frac{\partial p(\mathbf{x}, t|\mathbf{x}_0, t_0)}{\partial t} = \left(D\nabla^2 - \mathbf{v}\nabla \right) p(\mathbf{x}, t|\mathbf{x}_0, t_0). \tag{1.8}$$

Call the domain in which the diffusion take place Ω and the boundary to the domain Γ. The boundary Γ to the domain may allow passage of particles or not. The total probability in Ω is given by

$$N_\Omega(t|\mathbf{x}_0, t_0) = \int_\Omega p(\mathbf{x}, t|\mathbf{x}_0, t_0)\mathrm{d}\mathbf{x} \tag{1.9}$$

Take the time derivative of both sides in equation (1.9)

$$\frac{\partial N_\Omega(t|\mathbf{x}_0, t_0)}{\partial t} = \int_\Omega \frac{\partial p(\mathbf{x}, t|\mathbf{x}_0, t_0)}{\partial t}\mathrm{d}\mathbf{x} \tag{1.10}$$

Then one can equation (1.8) to get

$$\frac{\partial N_\Omega(t|\mathbf{x}_0, t_0)}{\partial t} = \int_\Omega \left(D\nabla^2 - \mathbf{v}\nabla\right) p(\mathbf{x}, t|\mathbf{x}_0, t_0)\mathrm{d}\mathbf{x} \tag{1.11}$$

Rewrite this as

$$\frac{\partial N_\Omega(t|\mathbf{x}_0, t_0)}{\partial t} = \int_\Omega \nabla \cdot (D\nabla - \mathbf{v})\, p(\mathbf{x}, t|\mathbf{x}_0, t_0)\mathrm{d}\mathbf{x} \tag{1.12}$$

and use Gauss theorem to get

$$\frac{\partial N_\Omega(t|\mathbf{x}_0, t_0)}{\partial t} = \int_\Gamma \mathrm{d}S \cdot (D\nabla - \mathbf{v})\, p(\mathbf{x}, t|\mathbf{x}_0, t_0). \tag{1.13}$$

Since (1.13) describes the rate of change of the total probability, the probability current $\mathbf{j}$ is minus the expression in the integrand,

$$\mathbf{j}(\mathbf{x}, t|\mathbf{x}_0, t_0) = (\mathbf{v} - D\nabla)\, p(\mathbf{x}, t|\mathbf{x}_0, t_0). \tag{1.14}$$

Since the Smoluchowski equation preserves the total probability the changes to N_Ω must comes from the boundary. The boundary conditions are written in terms of the flux at the boundary. There are three types of possible boundary conditions. The first type is the reflective boundary condition,

$$\left(n \cdot \mathbf{v}p(\mathbf{x}_\Gamma, t|\mathbf{x}_0, t_0) - D\frac{\partial p(\mathbf{x}_\Gamma, t|\mathbf{x}_0, t_0)}{\partial n}\right) = 0, \tag{1.15}$$

where $\mathbf{x}_\Gamma$ is a point on the boundary and n is the normal to the boundary at the position $\mathbf{x}_\Gamma$. No probability leaves the domain in this case, instead it is reflected. The second type is the absorbing boundary condition

$$p(\mathbf{x}_\Gamma, t|\mathbf{x}_0, t_0) = 0, \tag{1.16}$$

In this case all particles that hit the wall are taken out of the system. The third type of boundary condition is an intermediate case

$$\left(n \cdot \mathbf{v} p(\mathbf{x}_\Gamma, t|\mathbf{x}_0, t_0) - D \frac{p(\mathbf{x}_\Gamma, t|\mathbf{x}_0, t_0)}{\partial n} \right) = \mathbf{w} p(\mathbf{x}_\Gamma, t|\mathbf{x}_0, t_0), \tag{1.17}$$

where w is a parameter that describes the reactivity of the boundary. In this thesis reflective boundaries are studied.

The analytical solution of the Smoluchowski equation (1.8) in two dimensions can be computed only when the solution can be separated in the x- and y-direction. That is when there is no boundary, a boundary consisting of a straight line, two parallel lines or two orthogonal lines. In other cases numerical methods are needed to get a solution. The total two dimensional probability distribution with a reflecting boundary at $y = 0$ is

$$p(x, y) = p(x)p(y) = p_F(x)p_S(y), \tag{1.18}$$

where $p_F(x)$ is the equation for infinite space (no boundary) and $p_S(y)$ is the solution for a half space (boundary at $y = 0$). The one dimensional analytical solution for infinite space is [10]

$$p_F(x, t|x_0, 0) = \frac{1}{\sqrt{4\pi D t}} \exp\left[-\left(x - x_0 - v_x t\right)^2 / 4Dt \right]. \tag{1.19}$$

The analytical solution for a half space, with a boundary at $y = 0$, is [10]

$$p_S(y, t|y_0, 0) = \sum_{j=1}^{3} p_j(y, t|y_0, 0), \tag{1.20}$$

with

$$p_1(y, t|y_0, 0) = \frac{1}{\sqrt{4\pi D t}} \exp\left[-\left(y - y_0 - v_y t\right)^2 / 4Dt \right], \tag{1.21}$$

$$p_2(y, t|y_0, 0) = \frac{1}{\sqrt{4\pi D t}} \exp\left[-v_y y_0 / D - \left(y + y_0 - v_y t\right)^2 / 4Dt \right], \tag{1.22}$$

$$p_3(y, t|y0, 0) = \frac{-v_y}{2D} \exp\left(v_y y\right) \mathrm{erfc}\left[\left(y + y_0 + v_y t\right) / \sqrt{4Dt} \right]. \tag{1.23}$$

Consider now a "tilted" boundary, that is a boundary for which the angle between the boundary line and the x-axis is non-zero. The analytical solution for a tilted boundary can be computed from equation (1.18), by rotating the coordinates so that the boundary is at $\tilde{y} = 0$ in the rotated coordinate system $(\tilde{x}, \tilde{y})$. To get the solution $p(x, y)$, one needs the coordinates (x, y) in the rotated coordinate system $(\tilde{x}, \tilde{y})$. For this situation the solution can be

computed using $p(x,y) = p_S(\tilde{y}, t|\tilde{y}_0, 0)p_F(\tilde{x}, t|\tilde{x}_0, 0)$. The rotated coordinates is calculated from

$$\tilde{\mathbf{x}} = R\mathbf{x}, \tag{1.24}$$

where the rotation matrix R is

$$R = \begin{pmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{pmatrix}. \tag{1.25}$$

To get the drift normal to the boundary, which is needed in the boundary condition (1.15), the same rotation matrix is used,

$$\tilde{\mathbf{v}} = R\mathbf{v}. \tag{1.26}$$

The analytical expression for the current in the y-direction is

$$\begin{aligned} j_y = v_y p - D\frac{\partial p}{\partial y} = v_y p - D\left(\frac{\partial \tilde{x}}{\partial y}\frac{\partial p}{\partial \tilde{x}} + \frac{\partial \tilde{y}}{\partial y}\frac{\partial p}{\partial \tilde{y}}\right) = \\ v_y p - D\left(\sin\theta\frac{\partial p}{\partial \tilde{x}} + \cos\theta\frac{\partial p}{\partial \tilde{y}}\right) = \\ v_y p - D\left(\sin\theta p_S(\tilde{y})\frac{\partial p_F(\tilde{x})}{\partial \tilde{x}} + \cos\theta p_F(\tilde{x})\frac{\partial p_S(\tilde{y})}{\partial \tilde{y}}\right) \end{aligned} \tag{1.27}$$

Chapter 2

Method

2.1 Finite Difference Method

The finite difference method is used to get an approximate solution to a differential equation using discretization of the derivatives. Another way to solve differential equations is to use the finite element method which uses ansatzes and variational methods to find the solution. The finite difference method was chosen since it is easier to implement.

The one dimensional diffusion equation will be used as an example in this section,

$$\frac{\partial u}{\partial t} = D\nabla^2 u. \tag{2.1}$$

The models can easily be extended to more general cases. The equation 2.1 is (1.8) without the force term. Here u is used instead of p to follow the standard in numerical methods to use u for the solution.

Discretize the axis into n grid points x_i, $i = 1, ..., n$ with step size Δx. Let $u_i^{(n)}$ be a difference approximation of $u(x_i, t_n)$.

Discretize the spatial derivative in the equation using the central difference scheme

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2} + O((\Delta x)^2). \tag{2.2}$$

In one dimension the result is

$$\frac{\partial u}{\partial t} = D\left(\frac{u_{i+1}^{(n)} - 2u_i^{(n)} + u_{i-1}^{(n)}}{(\Delta x)^2}\right). \tag{2.3}$$

There are different ways to get the next time step $u_i^{(n+1)}$. The simplest

method is to use the explicit method,

$$\frac{u_i^{(n+1)} - u_i^{(n)}}{\Delta t} = D\left(\frac{u_{i+1}^{(n)} - 2u_i^{(n)} + u_{i-1}^{(n)}}{(\Delta x)^2}\right). \tag{2.4}$$

Very small time steps are needed to get a stable solution with this method, which makes the method inefficient. Another method is the implicit method, where the solution at the next time step is used in the approximation of the spatial derivatives,

$$\frac{u_i^{(n+1)} - u_i^{(n)}}{\Delta t} = D\left(\frac{u_{i+1}^{(n+1)} - 2u_i^{(n+1)} + u_{i-1}^{(n+1)}}{(\Delta x)^2}\right). \tag{2.5}$$

This method is, like the previous, only first order accurate in time. A method that is of second order in time is the Crank-Nicolson's method, which can be seen as the average of the explicit and implicit method. In one dimension

$$\frac{u_i^{(n+1)} - u_i^{(n)}}{\Delta t} = \frac{D}{2}\left[\left(\frac{u_{i+1}^{(n+1)} - 2u_i^{(n+1)} + u_{i-1}^{(n+1)}}{(\Delta x)^2}\right) + \left(\frac{u_{i+1}^{(n)} - 2u_i^{(n)} + u_{i-1}^{(n)}}{(\Delta x)^2}\right)\right]. \tag{2.6}$$

In two dimensions, discretize the x-axis into n_x grid points x_i, $i = 1, ..., n_x$ with step size Δx and discretize the y-axis into n_y points y_j, $j = 1, ..., n_y$ with step size Δy. Let $u_{i,j}^{(n)}$ be a difference approximation of $u(x_i, y_j, t_n)$. The Crank-Nicolson's method for (2.1) is then

$$\frac{u_{i,j}^{(n+1)} - u_{i,j}^{(n)}}{\Delta t} = \frac{D}{2}\left(\frac{u_{i+1,j}^{(n+1)} - 2u_{i,j}^{(n+1)} + u_{i-1,j}^{(n+1)}}{(\Delta x)^2} + \frac{u_{i+1,j}^{(n)} - 2u_{i,j}^{(n)} + u_{i-1,j}^{(n)}}{(\Delta x)^2} + \right.$$

$$\left. + \frac{u_{i,j+1}^{(n+1)} - 2u_{i,j}^{(n+1)} + u_{i,j-1}^{(n+1)}}{(\Delta y)^2} + \frac{u_{i,j+1}^{(n)} - 2u_{i,j}^{(n)} + u_{i,j-1}^{(n)}}{(\Delta y)^2}\right). \tag{2.7}$$

Crank-Nicolson's method is stable for all time step sizes and space step sizes as long as the time step size is small enough compared to the space step size [12]. How small depends on the problem.

The solutions $u_{i,j}^{(n)}$ can be written as a vector, with the grid points numbered in some way, for example $u_{in_x+j}^{(n)}$, see figure (2.1). In this case the

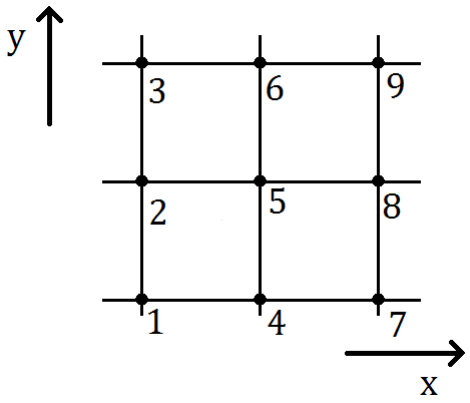

Figure 2.1: One way to number the grid points.

solution vector will have the form,

$$\mathbf{u}^{(n)} = \begin{pmatrix} u_1^{(n)} \\ u_2^{(n)} \\ \vdots \\ u_{in_y+j-1}^{(n)} \\ u_{in_y+j}^{(n)} \\ \vdots \\ u_{(i+1)n_y+j}^{(n)} \\ u_{(i+1)n_y+j+1}^{(n)} \\ \vdots \\ u_{n_x n_y} \end{pmatrix}. \tag{2.8}$$

Taking the solution at the next time step at the left hand side and the solution at the previous time at the right hand side, equation (2.7) becomes

$$u_{i,j}^{(n+1)} - \frac{D\Delta t}{2}\left(\frac{u_{i+1,j}^{(n+1)} - 2u_{i,j}^{(n+1)} + u_{i-1,j}^{(n+1)}}{(\Delta x)^2} + \frac{u_{i,j+1}^{(n+1)} - 2u_{i,j}^{(n+1)} + u_{i,j-1}^{(n+1)}}{(\Delta y)^2}\right) =$$

$$u_{i,j}^{(n)} + \frac{D\Delta t}{2}\left(\frac{u_{i+1,j}^{(n)} - 2u_{i,j}^{(n)} + u_{i-1,j}^{(n)}}{(\Delta x)^2} + \frac{u_{i,j+1}^{(n)} - 2u_{i,j}^{(n)} + u_{i,j-1}^{(n)}}{(\Delta y)^2}\right). \tag{2.9}$$

These equations (equation (2.9) for $i = 1, ..., n_x$ and $j = 1, ..., n_y$) can be written in matrix form as

$$A\mathbf{u}^{(n+1)} = B\mathbf{u}^{(n)}. \tag{2.10}$$

with A and B

$$A = \begin{pmatrix} 1+2\alpha_y+2\alpha_x & -\alpha_y & 0 & \cdots & 0 & -\alpha_x & 0 & \cdots & \cdots & 0 \\ -\alpha_y & 1+2\alpha_y+2\alpha_x & -\alpha_y & 0 & \cdots & 0 & -\alpha_x & \cdots & \cdots & 0 \\ 0 & -\alpha_y & 1+2\alpha_y+2\alpha_x & -\alpha_y & 0 & \cdots & 0 & -\alpha_x & \cdots & 0 \\ \cdots & \cdots & \ddots & \ddots & \ddots & \cdots & \cdots & \cdots & \ddots & \cdots \end{pmatrix}, \tag{2.11}$$

$$B = \begin{pmatrix} 1-2\alpha_y-2\alpha_x & \alpha_y & 0 & \cdots & 0 & \alpha_x & 0 & \cdots & \cdots & 0 \\ \alpha_y & 1-2\alpha_y-2\alpha_x & \alpha_y & 0 & \cdots & 0 & \alpha_x & \cdots & \cdots & 0 \\ 0 & \alpha_y & 1-2\alpha_y-2\alpha_x & \alpha_y & 0 & \cdots & 0 & \alpha_x & \cdots & 0 \\ \cdots & \cdots & \ddots & \ddots & \ddots & \cdots & \cdots & \cdots & \ddots & \cdots \end{pmatrix}, \tag{2.12}$$

with $\alpha_x = \frac{\Delta t D}{2(\Delta x)^2}$,

The Crank-Nicolson's method (and also the implicit method) gives in two or more dimensions a matrix A that is banded with a large band width. The structure of the matrix is shown in 2.2. The matrix A always has non-zero elements on the diagonal. Unless the previous point in the y-direction is outside ($j = 0$) there will be a non-zero element on the sub-diagonal. Unless the next point in the y-direction is outside ($j = N + 1$) there are non-zero elements on the super diagonal. These terms come from the discretized y-derivative. Row $in_x + y$ will have non-zero elements also on position $(i + 1)n_y + j$ and $(i - 1)n_y + j$ coming from the discretized x-derivative, unless the point $i + 1$ or $i - 1$ is outside the domain. The rest of the elements are zeroes. The band width is therefore $i + 2n_y$ elements.

The size of the matrix is $n^{n_d^{n_d}}$, with n being the number of points in each direction and n_d being the number of dimensions. Using 100 grid points in the x- and y-direction gives a matrix with 10^8 elements. An effective algorithm to solve the system of linear equations (2.10) will therefore be needed. This will be discussed in section 2.3.1.

An advantage of the implicit method compared to the Crank-Nicolson's method is that it treats the small-scale features in a more physical way. In one dimension, one can rewrite the difference equation using the implicit method (2.5) as

$$-\frac{D}{(\Delta x)^2}u_{i+1}^{(n+1)} + \left(\frac{2D}{(\Delta x)^2} - \frac{1}{2\Delta t}\right)u_i^{(n+1)} - \frac{D}{(\Delta x)^2}u_{i-1}^{(n+1)} = \frac{1}{2\Delta t}u_i^{(n)}. \tag{2.13}$$

For large time steps ($\Delta t \to \infty$) the equation becomes the difference equation for

$$\frac{\partial^2 u}{\partial x^2} = 0, \tag{2.14}$$

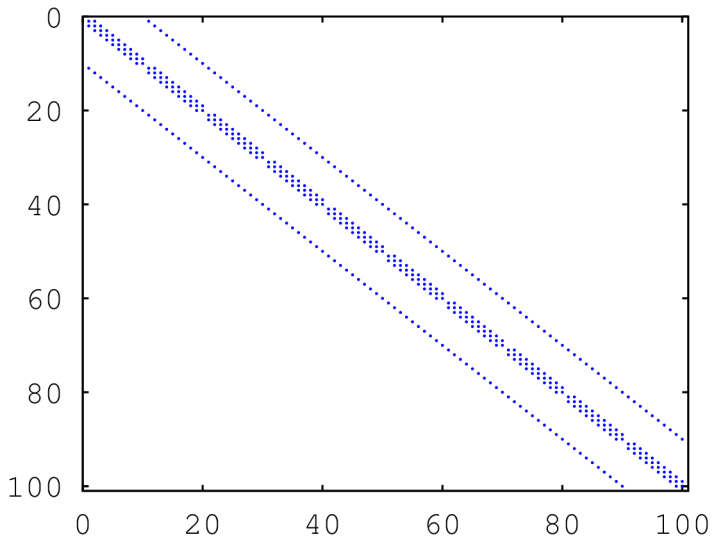

Figure 2.2: The structure of the matrix when using the Crank-Nicolson's method in two dimensions. Ten grid points have been used in the x- and y-direction. The matrix is therefore of size 100×100. The non-zero elements in the matrix are shown with blue dots. As can be seen, the matrix has a banded structure.

which is the stationary equation of the diffusion equation (2.1). The Crank-Nicolson's method on the other hand does not satisfy the stationary equation. In the Crank-Nicolson's method the small-scale features fluctuate, but do not change. It is too inefficient to have time steps of the order of the small scale fluctuations since the scales of interest are much larger. Therefore the Crank-Nicolson's method is preferred over the implicit method. To be sure that the small-scale features are in the stationary form one can do a few steps using the implicit method at the end of the simulation. [13]

2.2 Boundaries

2.2.1 Boundary Conditions

The boundary conditions can in the general case be written

$$a_0 \frac{\partial u(\mathbf{x}_\Gamma)}{\partial n} + a_1 u(\mathbf{x}_\Gamma) + a_2 = 0 \tag{2.15}$$

where $\mathbf{x}_\Gamma$ is a point at the boundary and n is the normal to the boundary. The values of the parameters a_0, a_1 and a_2 gives the type of boundary condition.

At the edges of the domain, the difference equations will need the solution of the point outside the domain. As an example, consider the one dimensional diffusion equation using the implicit method

$$\frac{u_i^{(n+1)} - u_i^{(n)}}{\Delta t} = D \left(\frac{u_{i+1}^{(n+1)} - 2u_i^{(n+1)} + u_{i-1}^{(n+1)}}{(\Delta x)^2} \right), \tag{2.16}$$

for $i = 1, ...N$. For $i = 1$

$$\frac{u_1^{(n+1)} - u_1^{(n)}}{\Delta t} = D \left(\frac{u_2^{(n+1)} - 2u_1^{(n+1)} + u_0^{(n+1)}}{(\Delta x)^2} \right). \tag{2.17}$$

The equation needs the solution at u_0, which is outside the domain and therefore unknown. The points outside the domain with a neighbouring point inside are called ghost points. The boundary condition is set by replacing the solution at the ghost points in the difference equations with the boundary condition.

To get an absorbing boundary (equation (1.16)), set the solution at the ghost points to zero.

Periodic boundary conditions in the x-direction are obtained by setting $u_{0,j}^{(n)} = u_{N,j}^{(n)}$ and $u_{N+1,j}^{(n)} = u_{1,j}^{(n)}$, if the domain is set so that $x_N - x_0$ is one period.

Reflective Boundary Condition

To get a reflective boundary condition (equation (1.15)), the derivative and solution at the boundary is needed. To approximate the derivative normal to the boundary at the boundary using points inside the domain and a ghost point, the method described in [14] is used. Let the point (x_i, y_j) be a ghost point, see figure 2.3. Calculate the normal n to the boundary Γ that goes through that ghost point. Use Lagrangian interpolation to get a second order approximation of the derivative

$$\frac{\partial u\left(\mathbf{x}_{i,j}^{\Gamma}\right)}{\partial n} = g_0 u_{i,j}^{(n)} + g_I u_I^{(n)} + g_{II} u_{II}^{(n)} + O((\Delta x)^2), \tag{2.18}$$

where $\mathbf{x}_{i,j}^{\Gamma}$ is the intersection point between the normal going through the ghost point and the boundary. The coefficients are

$$g_0 = \frac{3\xi_I - 2\xi_\Gamma}{2\xi_I^2}, \quad g_I = \frac{2\xi_\Gamma - 2\xi_I}{\xi_I^2}, \quad g_{II} = \frac{\xi_I - 2\xi_\Gamma}{2\xi_I^2}, \tag{2.19}$$

where ξ_Γ is the distance between the ghost point and the intersection point between the normal and the boundary. The parameter ξ_I is the distance between (x_i, y_j) and one of the four grid lines x_{i+1}, x_{i-1}, y_{j+1} or y_{j-1}, depending on the angle θ between the normal and the x-axis. Lets consider the case when $\pi/4 \leq \theta < 3\pi/4$. Then the grid line y_{j+1} is used. In this case the solution $u_I^{(n)}$ is the solution at the intersection point (x_I, y_I) between the normal and the y_{j+1} grid line and $u_{II}^{(n)}$ is the solution at the intersection point (x_{II}, y_{II}) between the normal and the y_{j+2} grid line. The solution $u_I^{(n)}$ is calculated using interpolation of three grid points along the y_{j+1} grid line. If $\pi/4 \leq \theta < \pi/2$ the grid points used are (x_i, y_{j+1}), (x_{i+1}, y_{j+1}) and (x_{i+2}, y_{j+1}). The solution $u_{II}^{(n)}$ is similarly calculated using interpolation of three grid points along the y_{j+2} grid line.

Depending on θ there are eight different expressions for $u_I^{(n)}$ and $u_{II}^{(n)}$.

$$\text{If} \quad \pi/4 \leq \theta < \pi/2 \quad \begin{cases} u_I^{(n)} = c_0 u_{i,j+1}^{(n)} + c_1 u_{i+1,j+1}^{(n)} + c_2 u_{i+2,j+1}^{(n)} \\ u_{II}^{(n)} = c_3 u_{i,j+2}^{(n)} + c_4 u_{i+1,j+2}^{(n)} + c_5 u_{i+2,j+2}^{(n)} \end{cases} \tag{2.20}$$

with the coefficients

$$c_0 = \frac{(\Delta x - \eta_I)(2\Delta x - \eta_I)}{2(\Delta x)^2}, \quad c_1 = \frac{\eta_I(2\Delta x - \eta_I)}{(\Delta x)^2}, \quad c_2 = \frac{-\eta_I(\Delta x - \eta_I)}{2(\Delta x)^2}. \tag{2.21}$$

with Δx being the step size and η_I being the distance between x_I and x_i. The coefficients in the expression for $u_{II}^{(n)}$ are the same as the coefficients in the

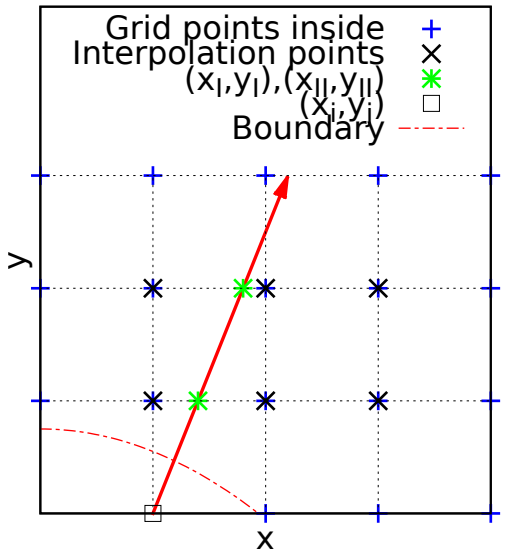

Figure 2.3: Picture explaining the interpolation of the derivative normal to the boundary at the boundary, for the case when the angle θ between the normal to the boundary and the x-axis is $\pi/4 \leq \theta \leq \pi/2$. The solution at the grid points called "Interpolation points" are used to get the solutions at (x_I, y_I) and (x_{II}, y_{II}). Thereafter the solutions at the ghost point $((x_i, y_j))$, (x_I, y_I) and (x_{II}, y_{II}) are used to get the derivative. The arrow is the normal to the boundary going through the ghost point (x_i, y_j).

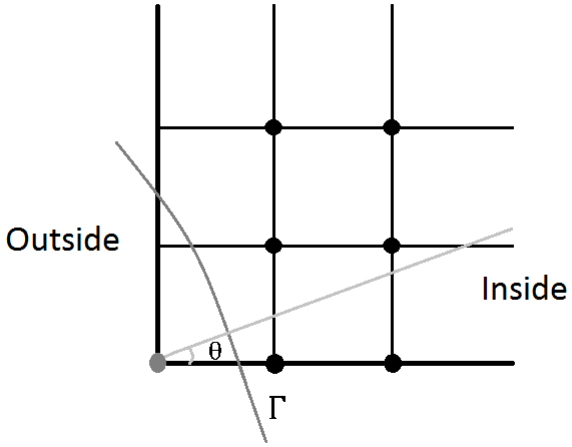

Figure 2.4: Which points to use when $0 \leq \theta \leq \pi/4$. The grey dot is the ghost point and the solutions at the black points are used to get $u_I^{(n)}$ and $u_{II}^{(n)}$.

expression for $u_I^{(n)}$ but with η_I being replaced by η_{II}, the distance between x_{II} and x_i. From trigonometry one get that $\eta_{II} = 2\eta_I$, so

$$c_3 = \frac{(\Delta x - 2\eta_I)(2\Delta x - 2\eta_I)}{2(\Delta x)^2}, \quad c_4 = \frac{2\eta_I(2\Delta x - 2\eta_I)}{(\Delta x)^2}, \quad c_5 = \frac{-2\eta_I(\Delta x - 2\eta_I)}{2(\Delta x)^2}. \tag{2.22}$$

See appendix A for the calculation of the coefficients (2.19) and (2.21).

In figure 2.4 the case when $0 \leq \theta < \pi/4$ is shown. In this case ξ_I is the distance between (x_i, y_j) and the x_{i+1} grid line. The coefficients in (2.21) are the same but with Δx being replaced by Δy and η_I being the distance between y_I and y_i. The other cases can be obtained by mirroring these results in the x_i and y_i grid lines.

If one or several of the interpolation grid points are outside of the boundary, one would still want the algorithm to work. Three other grid points can therefore be used in such a case. Preferably the points (x_I, y_I) and (x_{II}, y_{II}) should lie between two interpolation points. Consider the case shown in figure 2.3. If the point (x_{i+2}, y_{i+1}) would be outside, one can equally well use the solutions at the points (x_{i-1}, y_{j+1}), (x_i, y_{j+1}) and (x_{i+1}, y_{j+1}) to get $u_I^{(n)}$. If on the other hand the point (x_i, y_{i+1}) is outside, one can use the points (x_{i+1}, y_{j+1}), (x_{i+2}, y_{j+1}) and (x_{i+3}, y_{j+1}). This gives a less accurate result since all interpolation points are to the left of (x_I, y_I). If one of these points are outsider set (x_I, y_I) to the intersection between the normal and the x_{i+1} line and use the points (x_{i+1}, y_{j+1}), (x_{i+1}, y_{j+2}) and (x_{i+1}, y_{j+3}). This gives a

less accurate result since the distance between (x_I, y_I) and the boundary is larger.

The boundary condition equation (1.15) also needs the solution at the boundary. One can get the solution at the boundary using $u_{i,j}^{(n)}$, $u_I^{(n)}$ and $u_{II}^{(n)}$ as

$$u\left(\mathbf{x}_{i,j}^{\Gamma}\right)^{(n)} = l_0 u_{i,j}^{(n)} + l_I u_I^{(n)} + l_{II} u_{II}^{(n)} + O((\Delta x)^2), \tag{2.23}$$

where the coefficients are obtained in a similar way as the coefficients (2.21), but with the distance between the points Δx replaced with ξ_I and η_I being replaced with ξ_Γ,

$$l_0 = \frac{(\xi_I - \xi_\Gamma)(2\xi_I - \xi_\Gamma)}{2(\xi_I)^2}, \quad l_1 = \frac{\xi_\Gamma(2\xi_I - \xi_\Gamma)}{(\xi_I)^2}, \quad l_2 = \frac{-\xi_\Gamma(\xi_I - \xi_\Gamma)}{2(\xi_I)^2}. \tag{2.24}$$

Equation (2.15), (2.18) and (2.23) gives the expression for the discretized reflective boundary condition

$$-D\left(g_0 u_{i,j}^{(n)} + g_I u_I^{(n)} + g_{II} u_{II}^{(n)}\right) + v_{\tilde{y}}\left(l_0 u_{i,j}^{(n)} + l_I u_I^{(n)} + l_{II} u_{II}^{(n)}\right) = 0. \tag{2.25}$$

From this expression one can get the solution at the ghost point,

$$u_{i,j}^{(n)} = \frac{1}{-Dg_0 + v_{\tilde{y}} l_0}\left((Dg_I - v_{\tilde{y}} l_I) u_I^{(n)} + (Dg_{II} - v_{\tilde{y}} l_{II}) u_{II}^{(n)}\right). \tag{2.26}$$

Setting up the matrices

The left and right hand side matrices are first set up without taking the boundary condition into account. These matrices are the same as for free diffusion. The system of linear equations becomes

$$\tilde{A}\mathbf{u}^{(n+1)} = \tilde{B}\mathbf{u}^{(n)}. \tag{2.27}$$

Without a force term, $\tilde{A}$ and $\tilde{B}$ are the same as in 2.11 and 2.12.

From the boundary condition one get an expression for the solution at the ghost point $u_{i,j}^{(n)}$ that can be written as

$$u_{in_x+j}^{(n)} = \mathbf{C}_{in_x+j}\mathbf{u}. \tag{2.28}$$

For a reflective boundary the vector $\mathbf{C}_{in_x+j}$ is, using equation (2.25),

$$\mathbf{C}_{in_x+j} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -1 \\ 0 \\ \vdots \\ 0 \\ \frac{1}{-Dg_0+v_{\tilde{y}}l_0}(Dg_I - v_{\tilde{y}}l_I)c_0 \\ \frac{1}{-Dg_0+v_{\tilde{y}}l_0}(Dg_I - v_{\tilde{y}}l_I)c_1 \\ \frac{1}{-Dg_0+v_{\tilde{y}}l_0}(Dg_I - v_{\tilde{y}}l_I)c_2 \\ 0 \\ \vdots \\ 0 \\ \frac{1}{-Dg_0+v_{\tilde{y}}l_0}(Dg_{II} - v_{\tilde{y}}l_{II})c_3 \\ \frac{1}{-Dg_0+v_{\tilde{y}}l_0}(Dg_{II} - v_{\tilde{y}}l_{II})c_4 \\ \frac{1}{-Dg_0+v_{\tilde{y}}l_0}(Dg_{II} - v_{\tilde{y}}l_{II})c_5 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{2.29}$$

where the element -1 is in row $in_x + j$. Solutions $u_{i,j}^{(n)}$ that aren't solutions at a ghost point can also be written in the form (2.28) with $\mathbf{C}_{in_x+j}$ being a vector of zeros. Then one can construct a matrix A_{BC},

$$A_{BC} = \begin{pmatrix} \sum_{k=0}^{n_x n_y} \tilde{A}_{0,k} C_j^T \\ \sum_{k=0}^{n_x n_y} \tilde{A}_{1,k} C_j^T \\ \vdots \\ \sum_{k=0}^{n_x n_y} \tilde{A}_{n_x n_y,k} C_j^T \end{pmatrix}, \tag{2.30}$$

where $\tilde{A}_{m,k}$ is the element on row m and column k in the matrix $\tilde{A}$. Similarly a matrix B_{BC} can be constructed,

$$B_{BC} = \begin{pmatrix} \sum_{k=0}^{n_x n_y} \tilde{B}_{0,k} C_j^T \\ \sum_{k=0}^{n_x n_y} \tilde{B}_{1,k} C_j^T \\ \vdots \\ \sum_{k=0}^{n_x n_y} \tilde{B}_{n_x n_y,k} C_j^T \end{pmatrix}. \tag{2.31}$$

Adding these matrices to the ones in equation (2.27) gives

$$A = \tilde{A} + A_{BC} \tag{2.32}$$

and

$$B = \tilde{B} + B_{BC} \tag{2.33}$$

One then gets the system of linear equations including the boundary condition,

$$A\mathbf{u}^{(n+1)} = B\mathbf{u}^{(n)}. \tag{2.34}$$

Since the previous solution $\mathbf{u}^{(n)}$ is known, one can introduce the vector $\mathbf{b} = B\mathbf{u}^{(n)}$,

$$A\mathbf{u}^{(n+1)} = \mathbf{b}. \tag{2.35}$$

An example of the structure of the matrix A is shown in figure 2.5. Ten grid points are used in the x- and y-direction which gives a matrix of size 100×100. The boundary is the same as in figure 3.25. Without the boundary conditions, the structure would be the same as in figure 2.2. The boundary condition changes the structure of the matrix. It is still banded, since the discretized equation for the solution at a point is still only depending on points close by. The solution $u_{i,j}^{(n+1)}$ can depend on the solution at $u_{i+2,j+2}^{(n+1)}$ or $u_{i-2,j-2}^{(n+1)}$. The matrix elements at column $(i+2)n_y+j+2$ or $(i-2)n_y+j-2$ in line $in_y + j$ might therefore be non-zero. The solution $u_{i,j}^{(n+1)}$ cannot depend on solutions at points further away than this. The band width is therefore $4n_x + 4$.

2.2.2 Corners

All points need to have at least one neighbour in the x- and y-direction inside the domain to make it possible to write the discretized equations for those points. To be sure the boundary condition equations (2.18) can be written for all points inside the domain, all points need to have two neighbours in the x- and y-direction inside the domain. This can be seen as the smallest resolution possible. If there is an corner where the resolution is less than the needed resolution, an extra boundary segment is constructed so that those points are not included, see figure 2.6.

If one point has several neighbouring edges, there is one boundary condition equation for each neighbouring boundary for that point. If the points with not enough resolution are excluded, a grid point with two neighbouring points will have one boundary in the x-direction and the other in the y-direction. The solution at the grid point can then be replaced in the x-derivatives by the boundary condition for the boundary in the x-direction and similarly in the y-direction.

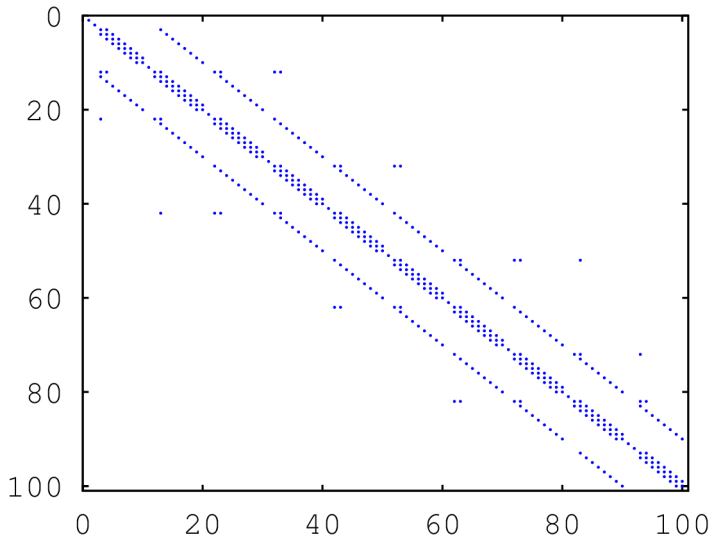

Figure 2.5: An example of the structure of the matrix A in equation 2.35 with $n_x = n_y = 10$, which gives a matrix of size 100×100. The boundary is the same as in figure 3.25. Without the boundary conditions, it would look like figure 2.2. The boundary condition changes the structure of the matrix. It is still banded, but with a larger band width.

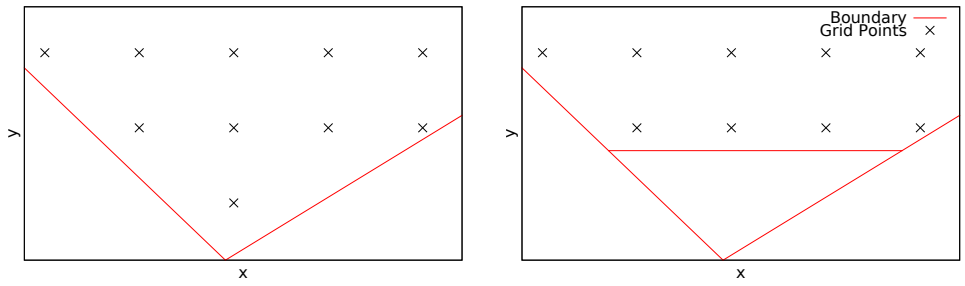

Figure 2.6: The point in the bottom corner have no neighbours in the x-direction and therefore the discretized equations can't be written for that point. A boundary segment is therefore added to remove the point.

2.3 The Algorithm

The algorithm works as follows: First construct the initial solution. The initial solution is taken to be a delta function that has had time to spread out a little bit to a Gaussian. Then construct the right and left hand side free diffusion matrices $\tilde{A}$ and $\tilde{B}$ in equation (2.27). These matrices are constant since the parameters are time-independent. After that, the boundary conditions (2.29) are calculated and stored. The matrices A_{BC} (equation (2.30)) and B_{BC} (equation (2.31)) are constructed. These matrices are added to the free diffusion matrices to get A (equation (2.32)) and B (equation (2.33)).

Iterate the time:

- Construct the right hand side vector $\mathbf{b}$ in 2.35.
- Solve the system of linear equations (2.35).

2.3.1 Design of the Program

The program was written in C++, which is an object oriented programming language, unlike e.g. C. Object oriented design is a way of designing the code using encapsulation. This makes it easier to improve the code since it limits the amount of changes that need to be done if one part is changed, which reduces the number of errors made and therefore the debugging time. It also makes it easier to reuse parts of the program.

Object oriented programming languages focus on objects instead of functions. The code is divided into modules called classes. Functions associated with an object are put together in a class. A class can be seen as a blueprint for an object.

Routines from the GNU Scientific Library (GSL) were used for the sparse matrices and the vectors. It is better to use already existing code as much as possible since it tends to be well tested, reliable and efficient.

Figure 2.7 shows a class diagram for the program for all classes that were constructed. The dark grey box is the algorithm. The arrows indicate which other classes a certain class uses. The arrow points to the class used.

The System Class

The System class contains everything that defines the system, like the parameters in the equation and the size of the domain.

The domain size is set by calculating the space needed using the simulation time t, the diffusion coefficient and the drift. The initial condition is

a delta function that has had time to spread out a little bit. The minimum point for the domain in the x-direction is

$$x_{min} = x_0 - d_x, \tag{2.36}$$

where x_0 is the starting position and d_x is half the total space needed in the x-direction,

$$d_x = x_{init} - x_{adv} - x_{diff}, \tag{2.37}$$

where x_{init} is the length needed for the initial solution, x_{adv} is the length needed for the advection and x_{diff} if the length needed for the diffusion. The space needed for the initial solution is computed by finding the points x_{init} where the absolute value of the flux (1.14) normal to the boundary at the boundary is smaller than a given tolerance ϵ,

$$\left| -D\frac{\partial p(x_{init}, y)}{\partial x} + v_x p(x_{init}, y) \right| < \epsilon. \tag{2.38}$$

The witdh of the Gaussian at time t is

$$\sigma = \sqrt{2Dt} \tag{2.39}$$

The length of the domain should be large enough to avoid a noticeable probability leak, but not larger than that because of efficiency reasons. The length needed for the diffusion is then

$$x_{diff} = a\sigma \tag{2.40}$$

where a is a number larger than 3. In the program a was chosen to be $a = 4$. The advection length is

$$x_{adv} = v_x t \tag{2.41}$$

The minimum and maximum points in the y-direction can be found in a corresponding way.

The Grid Class

The Grid class uses the System class since it needs the size of the domain to construct the grid. The grid was chosen to always be rectangular, for the cause of simplicity. Only including the points inside the domain would make the matrices smaller and the algorithm faster, but would be much more difficult to implement. The reason is that if for example one have a point (x_i, y_j) and wants its neighbour to the right in the x-direction $(x_i + 1, y_j)$, its index with a rectangular grid is $(i + 1)n_x + j$. With a grid that is not rectangular the number of x points vary from row to row, so one would need to keep track of the number of points there are both to the right of (x_i, y_j) in the row and to the left of $(x_i + 1, y_j)$ in the row above to be able to get the index of $(x_i + 1, y_j)$.

The InitialSolution Class

The InitialSolution class represents the discretized initial solution. It uses the System class, but that is not shown in figure 2.7 since it uses a class that uses System. The initial solution is either read from a file or computed from an analytical expression. Either the initial probability distribution is placed far away from the boundaries so that the analytical expression for free diffusion can be used, or a previously obtained solution stored in a file is used as an initial solution. The stored solution needs to have a grid with the same step size as before.

The Solution Class

The Solution class represents the solution at the current time. It uses the solution vector from the InitialSolution class as the starting value.

The LinEqMatrix Class

The LinEqMatrix class represents the left or right hand side matrices in the system of linear equations (2.27) or (2.34). The LinEqSparseMatrix is a derived class from LinEqMatrix, that is, it has the functions in the base class (LinEqMatrix) plus additional functions and data members. It stores the matrix as a sparse matrix using GSL compressed row storage. The CnMatrix class is a derived class from LinEqSparseMatrix that represents the matrix obtained from the Crank-Nicolson's method, that is the matrices $\tilde{A}$ and $\tilde{B}$ in equation (2.27). The matrices $\tilde{A}$ and $\tilde{B}$ are represented by different objects, but are constructed from the same class. The boundary condition matrices A_{BC} and B_{BC} (equations (2.30) and (2.31)) are added to $\tilde{A}$ and $\tilde{B}$ and the solutions A and B (equations (2.32) and (2.33)) are stored in the CnMatrix objects.

The BoundaryMatrix Class

The BoundaryMatrix class represents the matrices A_{BC} (equation (2.30)) and B_{BC} (equation (2.31)). There is one matrix A_{BC} and one matrix B_{BC} for each boundary segment, so that each matrix can be tested separately.

The BcCoeff Class

This class represents the vectors $\mathbf{C}$ (equation (2.29)). The vectors $\mathbf{C}$ for the ghost points are calculated and stored in a matrix.

The Interpolation Class

The interpolation class represents the Lagrange interpolation. It was put in a separate class since it was used both by the BcCoeff class and when calculating the probability current.

The Boundary Class

The boundary is one object. The boundary has one or more boundary segments. The boundary segment can be of different types, e.g. a line or parabola. This is done by having derived classes such as LineSegment and ParabolicSegment.

The program was designed so that one only needs to add a new class with functions returning the function of the boundary segment $y = f(x)$ and the derivative $y = f'(x)$ to be able to use a new type of boundary segment. Each boundary segment has a boundary condition that can be either open or reflective.

The Rotation Class

The rotation class represents the rotation (1.24). It was put in a separate class since it was used both to calculate the boundary condition and to calculate the analytical solution.

The SolveSparse Class

The SolveSparse class represents the solver that solves the system of linear equations. To solve equation (2.35), a sparse iterative solver was used, namely the Generalized Minimum Residual Method (GMRES) from the GNU Scientific Library.

To use an effective exact solver, one needs to make use of the structure of the matrix. The structure of the matrix A is shown in figure 2.5. It depends on the boundary used, but is always very sparse and has a banded structure with a large band width. If one would store only the elements in the band and use an exact solver for banded matrices one would need to store on the order of $n_x^2 n_y$ elements. The order of non-zero elements are $n_x n_y$. Most of the stored elements would thus be zero, which would be very inefficient. An iterative solver is therefore much faster to use.

The RhsVector Class

The RhsVector class represents the vector $\mathbf{b}$ in equation (2.35).

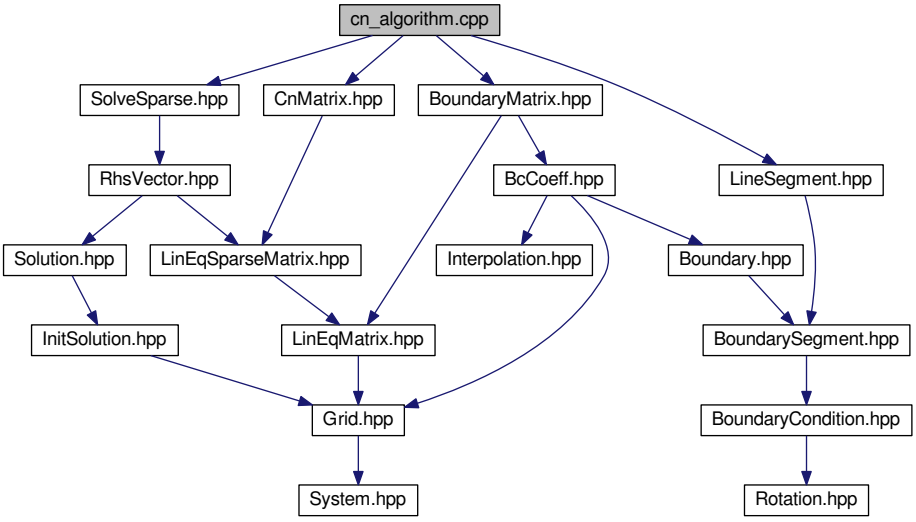

Figure 2.7: Class diagram for the algorithm. The diagram contains all the classes used by the algorithm (except the C++ standard library classes and the GSL classes). The arrows points towards the classes a certain class uses.

Chapter 3

Results

3.1 Boundary at $y = 0$

The program were first checked for the simplest case, a straight line at $y = 0$. The set up is shown in figure 3.1, with the starting point slightly above the boundary, $x_0 = 2.0$, $y_0 = 1.8$ and with $v_x = 0$ and $v_y = -1$. The domain is from $x_{min} = -10.6$ to $x_{max} = 14.6$ in the x-direction and $y_{min} = 0$ to $y_{max} = 14.4$ in the y-direction.

The initial solution was set to the free diffusion equation (1.19) with $t = 0.004$. The time step size were taken to be $\min(\Delta x, \Delta y)/20$. A smaller time step size was tried, $\min(\Delta x, \Delta y)/200$, but that did not change the results. This time step size and initial solution were used in all simulations.

The numerical probability distribution along $x = 2.0$ at $t = 2.0$ is shown in figure 3.2 together with the analytical solution calculated from (1.18). The difference between the numerical and analytical solution is not visible. The probability current together with the analytical solution for the current (1.14) is shown in figure 3.3. The current is zero at $y = 0$ as expected because of the boundary condition. The difference between the numerical and analytical solution is very small also in this case.

Several tests were made to check that the solution is correct. The total probability (equation (1.9)) as a function of time is shown in figure 3.4. Since there should be no flux at the boundary, the total probability should be conserved. The discretization of the boundary condition introduces a small current, which leads to the norm changing over time. This probability leak should decrease with a finer grid, which it does.

The integrated boundary current

$$j_\Gamma(t) = \int_{x=x_{min}}^{x=x_{max}} j_y(x, y = y_\Gamma, t)\mathrm{d}x \tag{3.1}$$

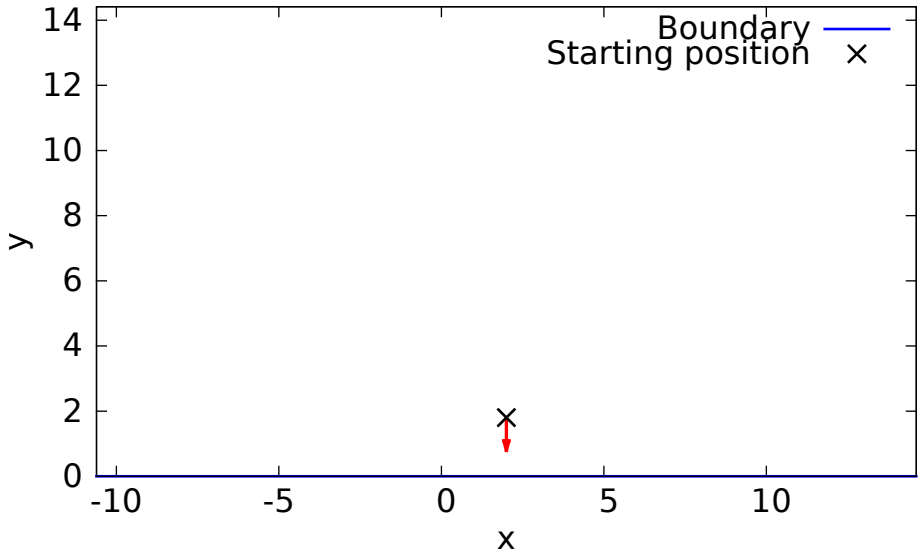

Figure 3.1: The system with a boundary at $y = 0$. The starting position is $x_0 = 2.0$, $y_0 = 1.8$. The red arrow shows the drift, $v_x = 0$, $v_y = -1$. The area in the plot is the domain included in the calculations.

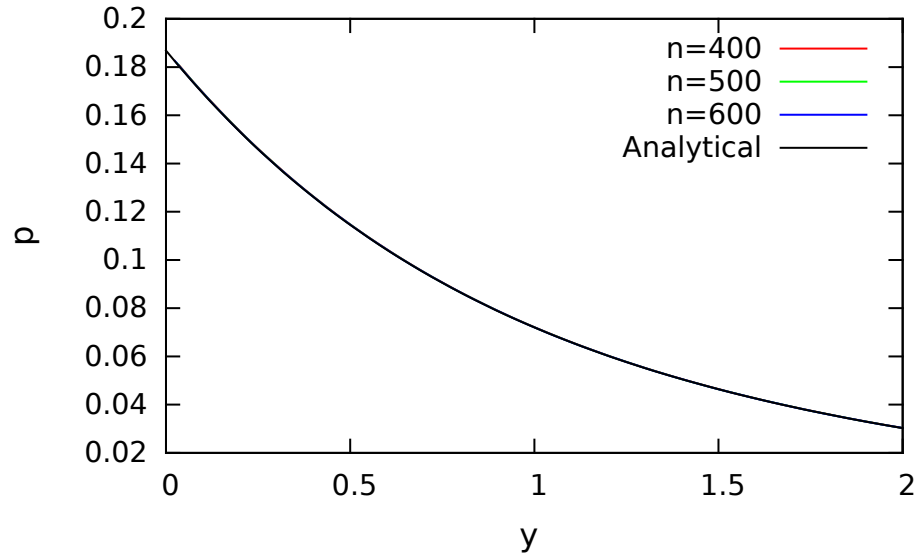

Figure 3.2: The numerical probability distribution along $x = 2.0$ for the set up in figure 3.1 at time $t = 2.0$ together with the analytical. The difference between the numerical and analytical solutions is not visible.

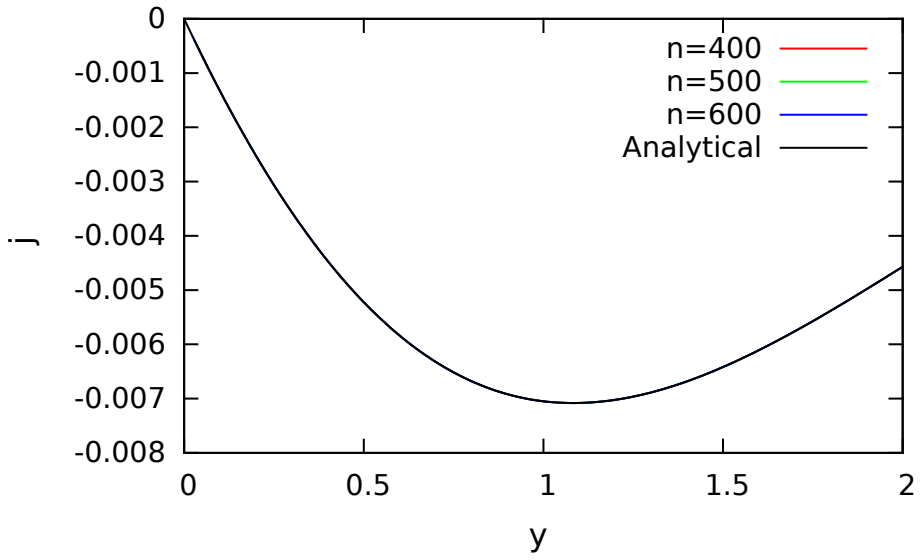

Figure 3.3: The numerical result for the probability current along $x = 2.0$ for the set up in figure 3.1 at $t = 2.0$ together with the analytical result. The numerical and analytical results are indistinguishable.

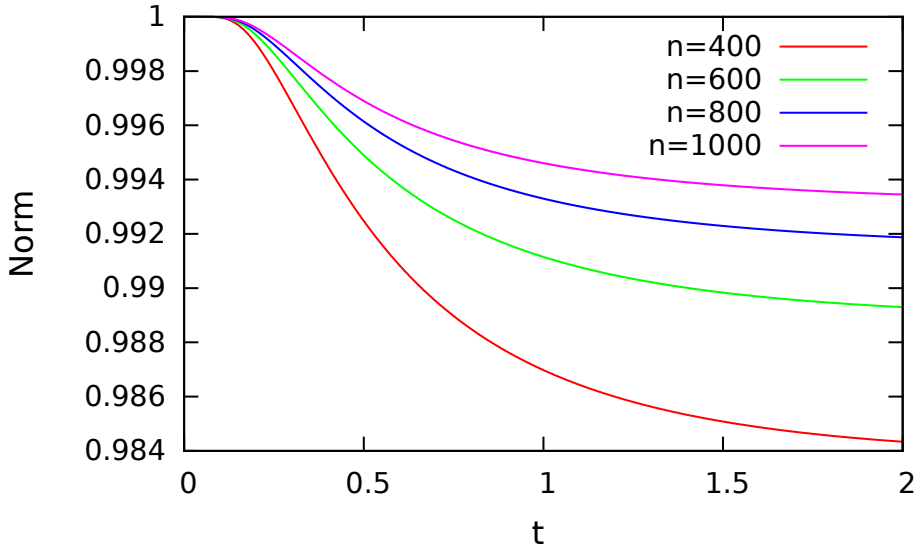

Figure 3.4: The total probability (equation (1.9)) for the set up in figure 3.1. The norm is not conserved, but the error gets smaller with a smaller step size

as a function of time is shown in figure 3.5. Because of the boundary condition the current at the boundary should be zero. It will not be exactly zero (to machine precision) since another (one-direction) discretization is used to calculate it compared to the discretization used to impose the condition in the equations. To impose the boundary condition in the system of linear equations (2.35), equation (2.25) is used. For a boundary at $y = 0$, $\xi_I = \Delta y$, and $\eta_I = 0$. Equation (2.25) then becomes

$$j_{\tilde{y}}(\mathbf{x}^{\Gamma}_{i,j}) = -D\left(g_0 u^{(n)}_{i,j} + g_I u^{(n)}_{i,j+1} + g_{II} u^{(n)}_{i,j+2}\right) + v_{\tilde{y}}\left(l_0 u^{(n)}_{i,j} + l_I u^{(n)}_{i,j+1} + l_{II} u^{(n)}_{i,j+2}\right) = 0, \tag{3.2}$$

with $u^{(n)}_{i,j}$ being the ghost point. Since the ghost point is not included in the domain, the solution at this point is not known. To calculated the current at the boundary from the obtained solution a point inside with a neighbouring point outside (x_{i_E}, y_{j_E}) is used instead of the ghost point. A similar interpolation as the one in section 2.2.1 is used, only replacing the ghost point with (x_{i_E}, y_{j_E}). The result is the same as in equation (2.18) but with ξ_Γ replaced with $-\xi_\Gamma$. For a boundary at $y = 0$ the expression becomes

$$j_{\tilde{y}} = -D\left(d_0 u^{(n)}_{i,j+1} + d_I u^{(n)}_{i,j+2} + d_{II} u^{(n)}_{i,j+3}\right) + v_{\tilde{y}}\left(b_0 u^{(n)}_{i,j+1} + b_I u^{(n)}_{i,j+2} + b_{II} u^{(n)}_{i,j+3}\right) \tag{3.3}$$

The coefficients d_0, d_I and d_{II} are

$$d_0 = \frac{3\xi_I + 2\xi_E}{2\xi_I^2}, \quad d_I = \frac{-2\xi_E + 2\xi_I}{\xi_I^2}, \quad d_{II} = \frac{\xi_I + 2\xi_E}{2\xi_I^2}, \tag{3.4}$$

where ξ_E is the distance between (x_{i_E}, y_{j_E}) and the boundary along the normal. The coefficients b_0, b_I and b_{II} are

$$b_0 = \frac{(\xi_I + \xi_E)(2\xi_I + \xi_E)}{2(\xi_I)^2}, \quad b_1 = \frac{-\xi_E(2\xi_I + \xi_E)}{(\xi_I)^2}, \quad b_2 = \frac{\xi_E(\xi_I + \xi_E)}{2(\xi_I)^2}. \tag{3.5}$$

Since the expressions (3.2) and (3.3) are different they will have different discretization errors and therefore the results obtained from one of them will be slightly different from the result obtained with the other.

A finer grid makes the boundary current smaller. To check the convergence the boundary current 3.1 is integrated over time for different step size. One expects that the boundary current would decrease quadratically with

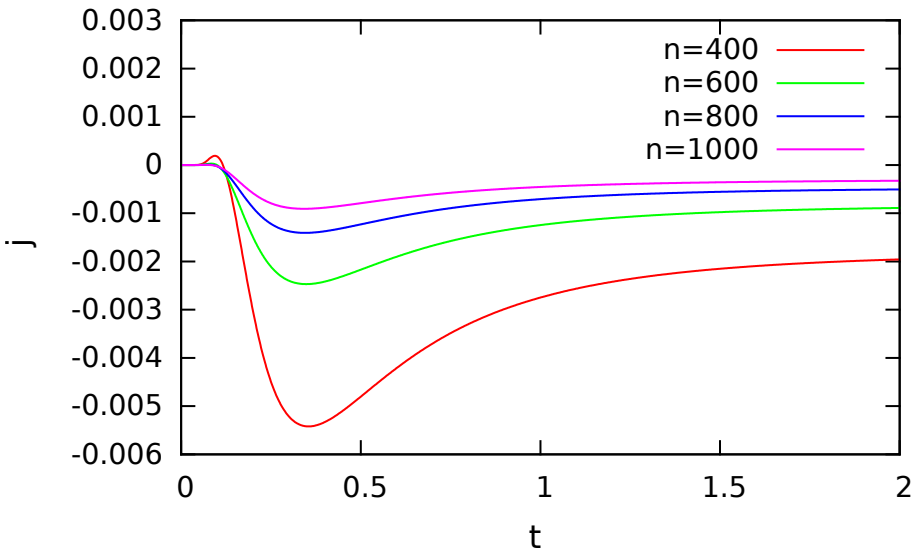

Figure 3.5: The integrated boundary current calculated from (3.1) at $t = 2.0$. The initial set up is shown in figure 3.1. The boundary current is not zero, but decreases with a smaller step size.

the step size since the Crank-Nicolson's method has quadratic convergence. If the convergence is quadratic one should have that

$$\log\left(\int_0^{t_{max}} j_\Gamma(t)\mathrm{d}t\right) = 2\log(\Delta y). \tag{3.6}$$

Figure 3.6 shows that this is indeed the case.

Another test that was made was to start with a stationary solution $\exp(vy)$ in the y-direction, see figure 3.7. The boundary current is shown in figure 3.8. The solution should be stable, which is not the case. The current at the boundary is a non-zero constant, so the solution will blow up after a long time. The size of the boundary current decreases with the space step size.

3.2 Tilted Boundary

The second case that was studied is a "tilted" line boundary, that is there is a non-zero angle between the boundary and the x-axis. The system with a boundary at $y = 0.8x$, starting position at $x_0 = 2.0$, $y_0 = 3.4$ and drift $v_x = 0$, $v_y = -1$ is shown in figure 3.9. The angle between the x-axis and the boundary is about 0.7. The domain is from -10.6 to 14.6 in the x-direction

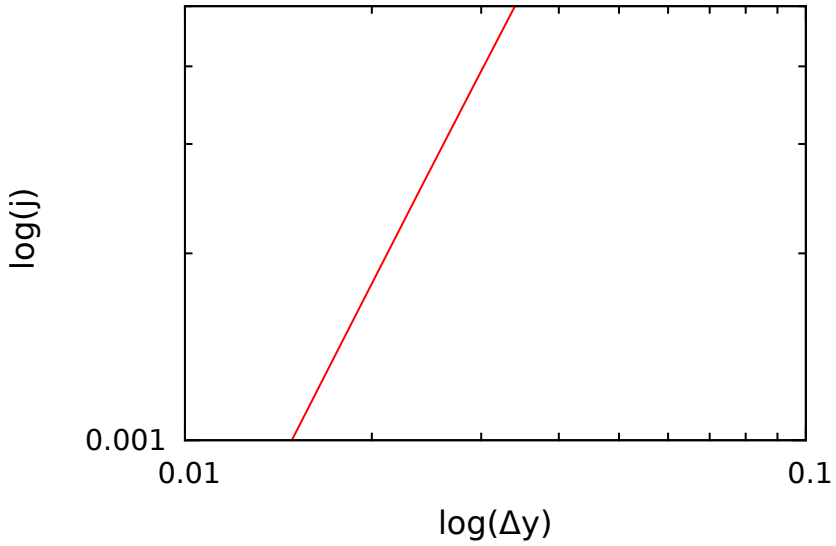

Figure 3.6: The time integrated boundary current with $t_{max} = 2.0$ computed using (3.6) as a function of the step size in the y-direction. The initial set up is shown in figure 3.1. The line has a slope of approximately 2. The boundary current therefore decreases quadratically with the step size. The axis scales are equal.

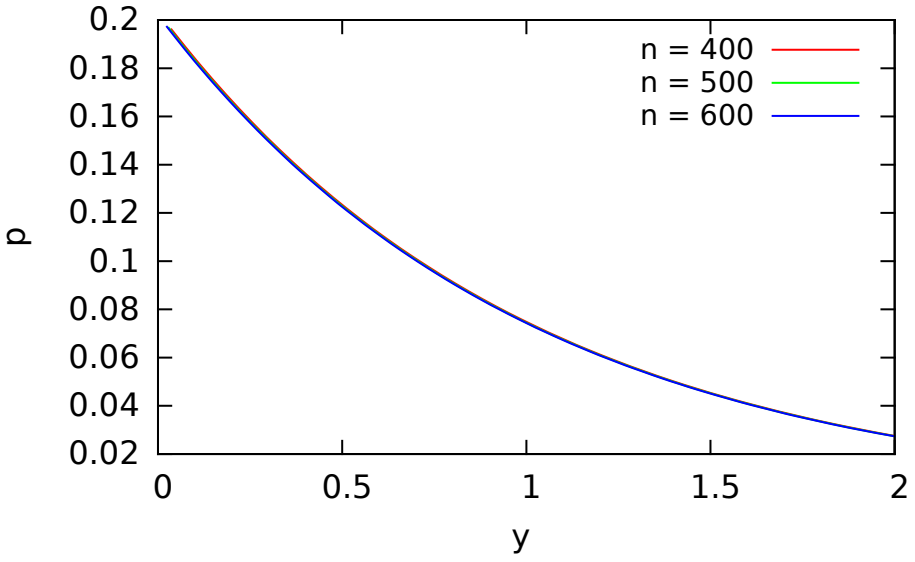

Figure 3.7: The solution along $x = 2.0$ with the same system as in figure 3.1 but starting with a stationary distribution.

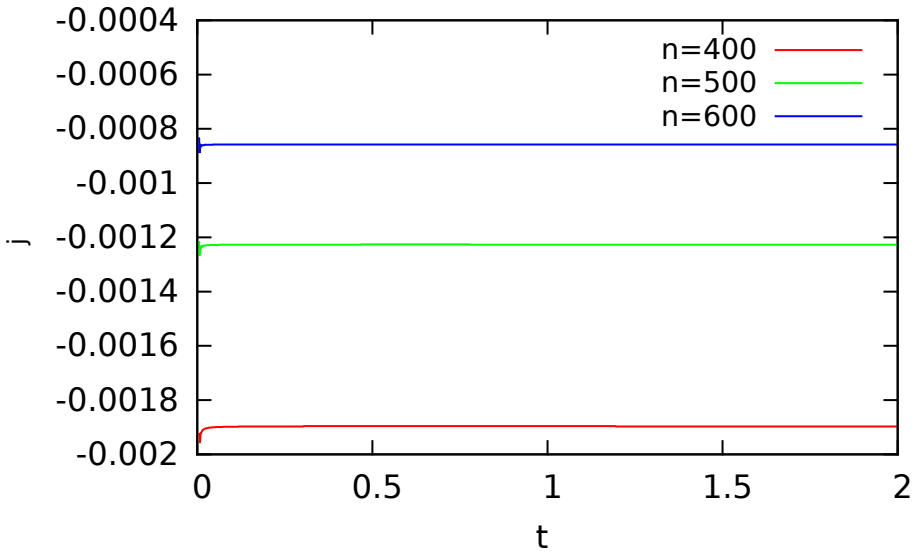

Figure 3.8: The probability current integrated along the boundary for the same system as in figure 3.1 but starting with a stationary distribution. The current is not zero as wanted so the solution will be inaccurate after a long time. A finer grid makes the error smaller.

and -8.5 to 16.0 in the y-direction. The numerical probability distribution along $x = 2.0$ at $t = 2.0$ is compared to the analytical probability distribution along in figure 3.10. The difference between the numerical and the analytical solution is not visible.

The probability current along $x = 2.0$ is shown in figure 3.11 together with the analytical result from (1.27). The difference is very small also in this case. The current normal to the boundary at $\tilde{x} = 2.6$ is shown in figure 3.12. At the boundary ($y = 1.6$, $\tilde{y} = 0$), $\tilde{x} = 2.6$ corresponds to $x = 2.0$. The error is much larger at the boundary than at points further away. The result improves with a finer grid.

The integrated boundary current (figure 3.13) decreases when the step size decreases, as wanted.

Since the agreement between the results improve with a decreasing step size one can conclude that the algorithm is working also for a tilted boundary.

3.3 Two Lines

A system consisting of two boundaries with angle $\pi/2$ between them can be compared with an analytical solution, since the solution in the x- and

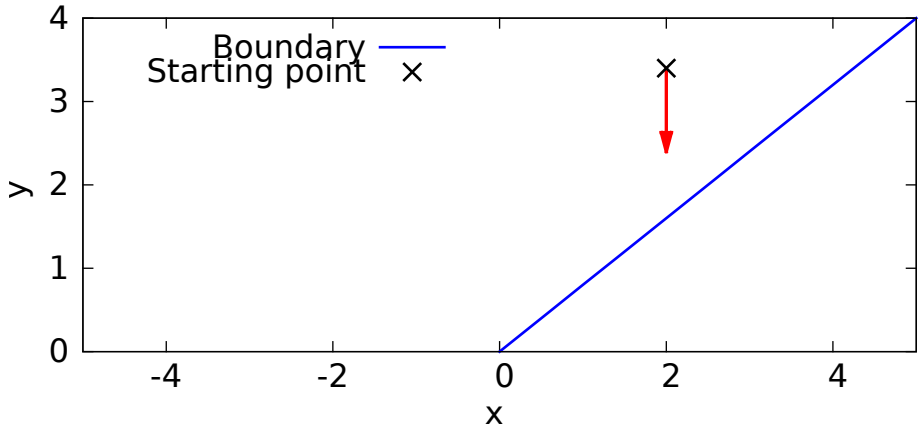

Figure 3.9: The system with boundaries at $y = 0.8x$. The starting position is $x_0 = 2.0$, $y_0 = 3.4$. The angle between the x-axis and the boundary is about 0.7. The red arrow shows the drift, $v_x = 0$, $v_y = -1$. The area in the plot is the domain included in the calculations.

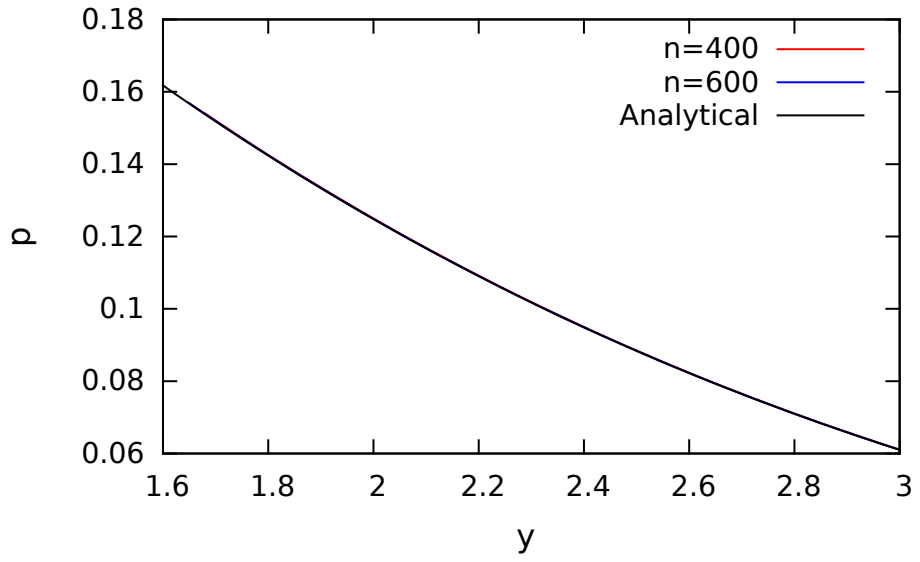

Figure 3.10: The probability distribution along $x = 2.0$ for the system in figure 3.9 at time $t = 2.0$. The number of points are $n_x = n_y = n$. The difference between the analytical solution and the numerical is not visible.

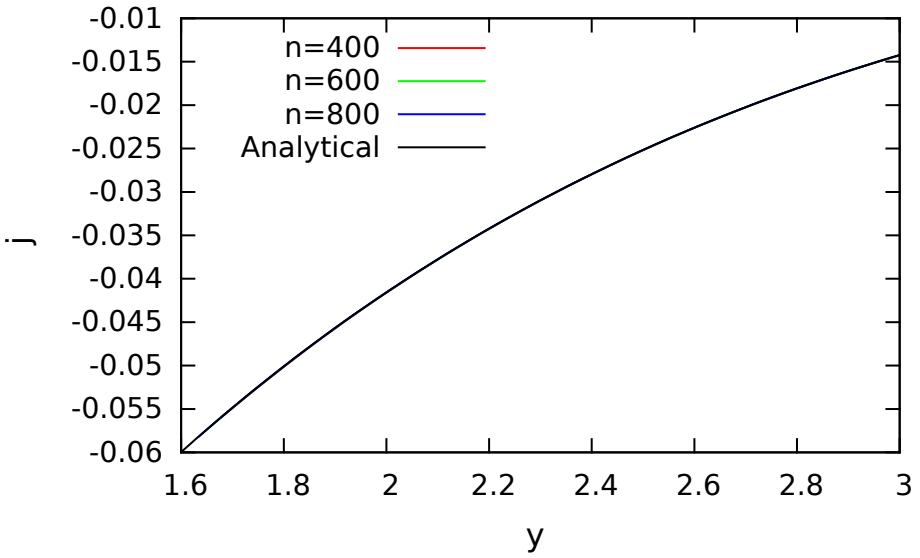

Figure 3.11: The probability current along $x = 2.0$ for the system in figure 3.9 at time $t = 2.0$. The number of points are $n_x = n_y = n$. The difference between the analytical solution and the numerical is not visible.

y-direction separate. One can use the one dimensional solution for a half plane (1.20) in both the x- and y-direction. Figure 3.14 shows the case with boundaries at $y = 0$ and $x = 0$, $v_x = v_y = -1$ and $x_0 = y_0 = 1.0$. The domain size is 0 to 14.6 in the x- and y-direction. The probability distribution after $t = 2.0$ is shown in figure 3.15. The number of grid points used were $n_x = n_y = 1000$ which gives the step size $\Delta x = \Delta y = 0.015$. Figure 3.16 shows a comparison with the analytical solution. The numerical solution is almost indistinguishable from the analytical solution, except close to the boundaries.

The same set up as in figure 3.14 but a slightly different drift, $v_x = -0.5$, $v_y = -1$, is shown in figure 3.17. The comparison between the analytical and numerical probability distributions is shown in figure 3.18. The domain size and step sizes are the same as in the previous case. Also in this case the difference between the numerical and analytical solution is very small.

When the angle between the lines is not $\pi/2$ there is no analytical solution. The case with boundaries at $y = 0$ and $y = 0.8x$, $v_x = -1.0$, $v_y = 0$, $x_0 = 2.5$, $y_0 = 1.0$ is shown in figure 3.19. The probability distribution at $t = 2$ is shown in figure 3.20. The domain is from 0 to 15.1 in the x-direction and 0 to 13.6 in the y-direction. The number of points used were $n_x = n_y = 1000$ which gives the step size $\Delta x = 0.015$ and $\Delta y = 0.014$. The corner has been

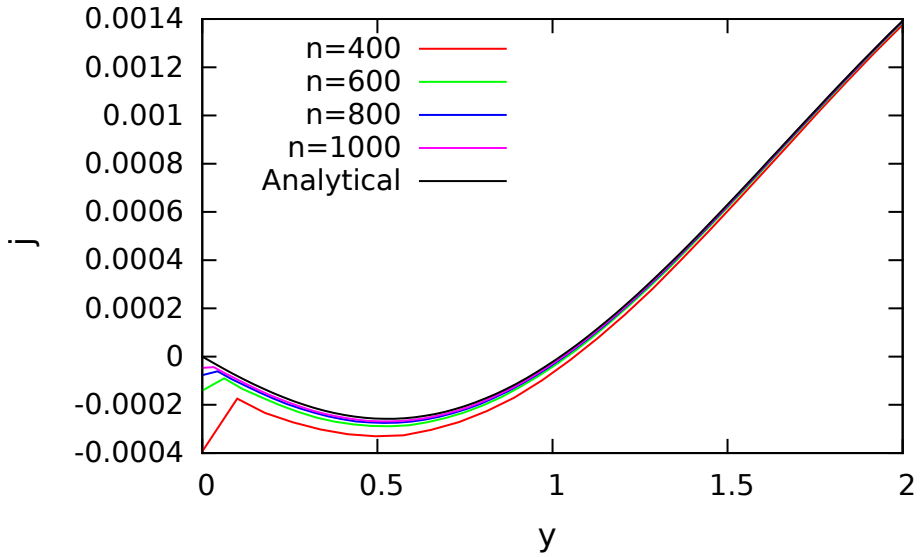

Figure 3.12: The current normal to the boundary along $\tilde{x} = 2.6$ for the system in figure 3.9 at time $t = 2.0$ compared to the analytical solution. At the boundary ($y = 1.6$), $\tilde{x} = 2.6$ corresponds to $x = 2.0$. The number of points are $n_x = n_y = n$. The difference between the numerical solutions and analytical solution is clearly visible. The error decreases with a smaller step size.

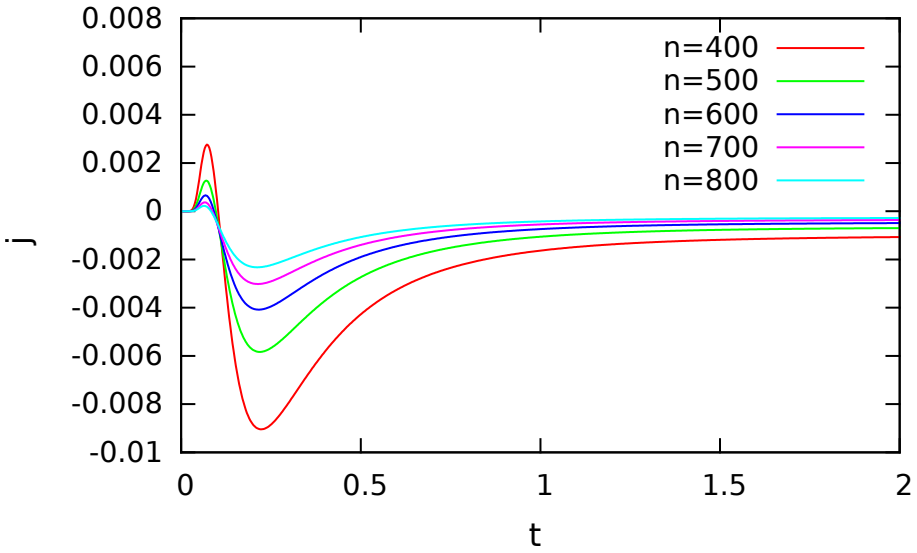

Figure 3.13: The integrated probability current along the reflective for the system in figure 3.9. The number of points are $n_x = n_y = n$. The current becomes smaller with a smaller steps size, as wanted.

removed because of lack of resolution.

3.4 Other Segment Shapes

Two other forms of boundary segments were tried, a parabola and a sine function. Figure 3.21 shows the set up for a boundary at $y = 0.1x^2$ and with $x_0 = 0$, $y_0 = 0.7$, $t = 2$ and $v_x = 0$, $v_y = -1$. The domain is from -12.6 to 12.6 in the x-direction and 0 to 14.1 in the y-direction. The number of points used were $n_x = n_y = 1000$ which gives $\Delta x = 0.025$ and $\Delta y = 0.013$. The probability distribution is shown in figure 3.22. The result is symmetric as expected.

Figure 3.23 shows the set up with $x_0 = -3$ and $y_0 = 2$. The domain is from -15.6 to 9.6 in the x-direction and 0 to 14.6 in the y-direction. The number of points used were $n_x = n_y = 1000$ which gives $\Delta x = 0.025$ and $\Delta y = 0.015$. The probability distribution is shown in figure 3.24. The distribution slides down the boundary towards the bottom.

The set up with a boundary at $y = \sin(x)$ with $x_0 = 2.2$, $y_0 = 1.5$, $v_x = 0$, $v_y = -1$, is shown in figure 3.25. The domain is from -10.4 to 14.8 in the x-direction and -1 to 14.2 in the y-direction. The probability distribution at $t = 2$ and $n_x = n_y = 1000$ which gives $\Delta x = 0.025$ and $\Delta y = 0.015$ is

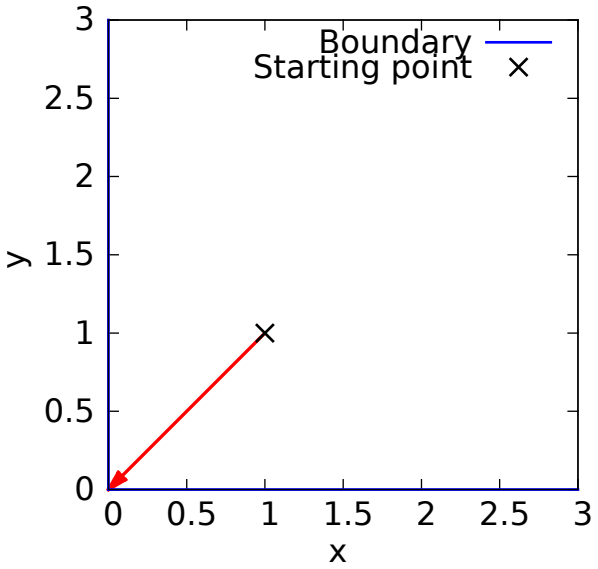

Figure 3.14: The system with boundaries at $y = 0$ and $x = 0$. The starting position is $x_0 = y_0 = 1.0$. The red arrow shows the drift, $v_x = v_y = -1$.

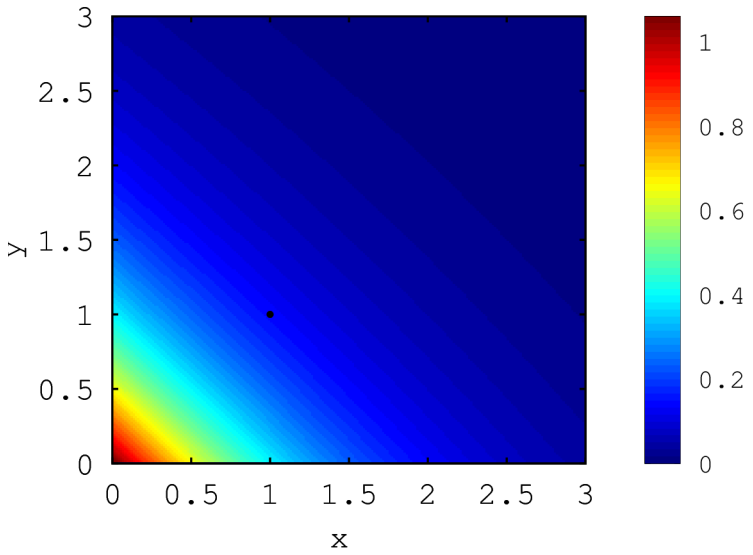

Figure 3.15: The probability distribution at $t = 2.0$ for the system in figure 3.14. The step size is $\Delta x = \Delta y = 0.015$. The black dot is the starting point.

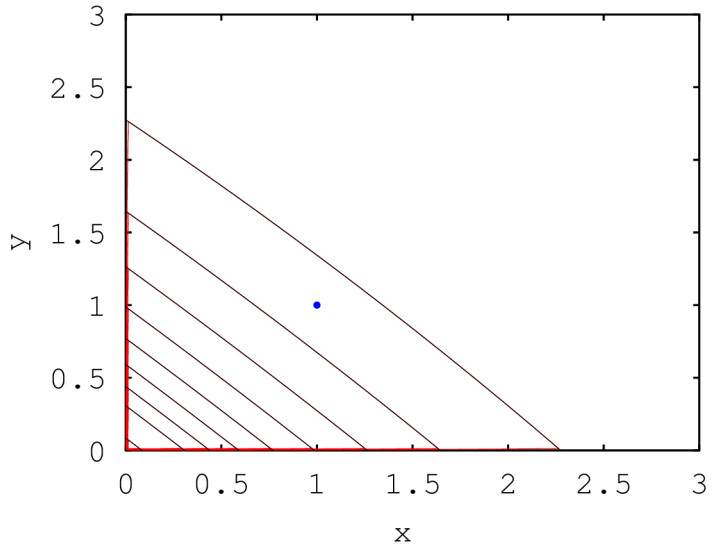

Figure 3.16: Contour plot of the same probability distribution as in figure 3.15 (red) together with the analytical solution (black). The solutions are almost indistinguishable. The contour lines shows the probability at $p = 0.1, 0.2, ...$ The blue dot is the starting point.

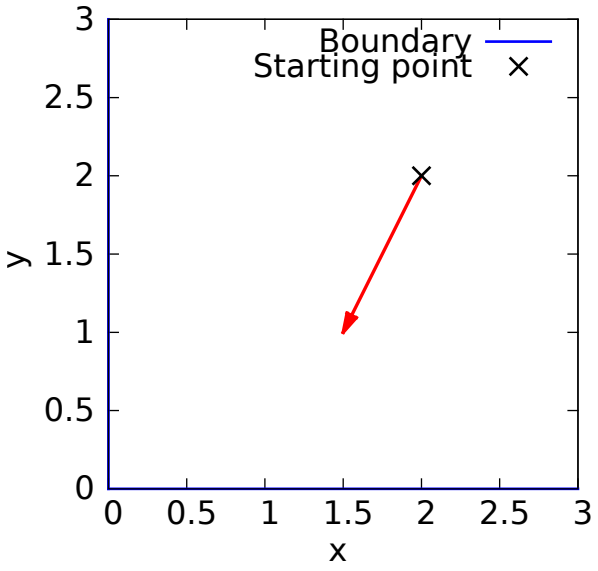

Figure 3.17: The system with boundaries at $y = 0$ and $x = 0$. The starting position is $x_0 = y_0 = 2.0$. The red arrow shows the drift, $v_x = -0.5$, $v_y = -1$.

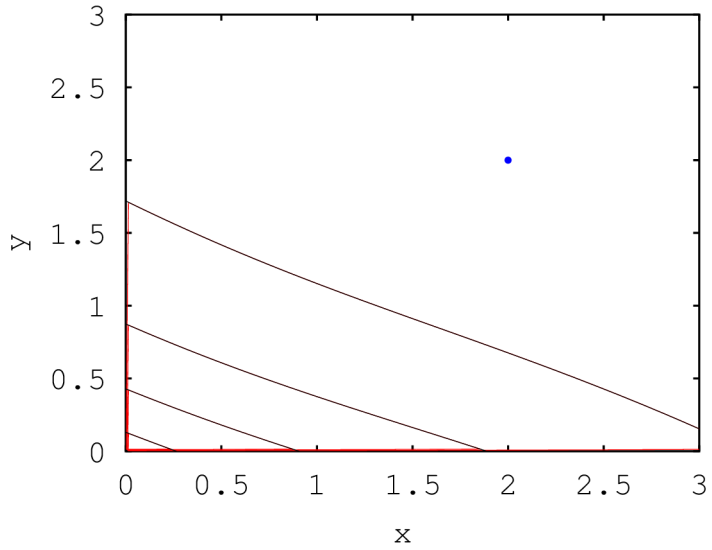

Figure 3.18: Contour plot of the probability distribution at time $t = 2.0$ for the same system as in figure 3.17. The step size is $\Delta x = \Delta y = 0.015$. The numerical solution (red) together with the analytical solution (black). The solutions are almost indistinguishable. The contour lines shows the probability at $p = 0.1, 0.2, ...$ The blue dot is the starting point.

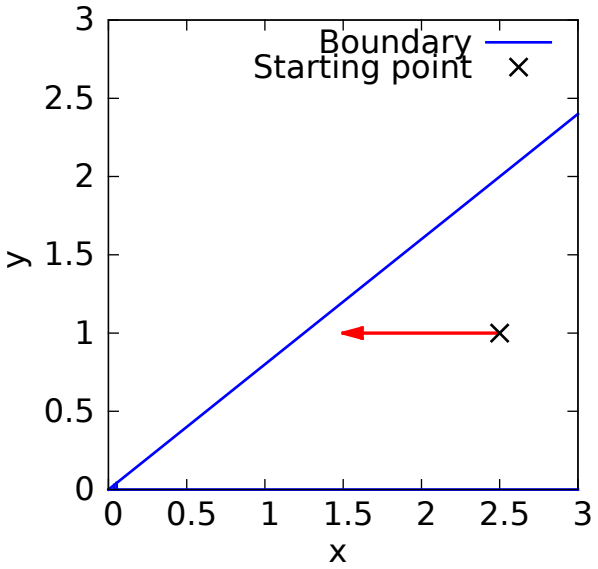

Figure 3.19: The system with boundaries at $y = 0.8x$ and $x = 0$. The starting position is $x_0 = 2.5$, $y_0 = 1.0$. The red arrow shows the drift, $v_x = -1$, $v_y = 0$.

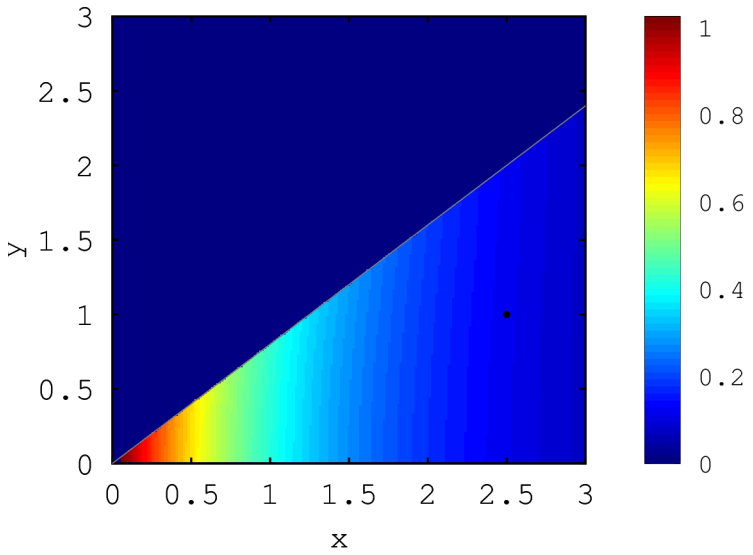

Figure 3.20: Probability distribution at $t = 2.0$ for the system in figure 3.19. The domain is from 0 to 15.1 in the x-direction and 0 to 13.6 in the y-direction. The number of grid points used were $n_x = n_y = 1000$ which gives and $\Delta x = \Delta y = 0.014$. The black dot is the starting point.

shown in figure 3.26. Also this picture looks as expected. The starting point is to the right of the peak in the middle. Most of the distribution ends up to the right. Because of diffusion some parts ends up to the left.

3.5 Periodic Channel

The system of main interest in this thesis is a channel with a periodically varying cross section. The channel was constructed by having a lower boundary $y = 0$ and a upper boundary $y = A\sin(cx) + b$. Periodic boundary conditions were implemented in the x-direction. The opening $b - A$ is much smaller than the width of the period. The force is along the x-axis. In figure 3.27 the system with $y = \sin(0.2x) + 1.1$ and $x_0 = 7.9$, $y_0 = 0.7$ is shown. The solution after a long time when the system has reach stationary, $t = 100$, is shown in figure 3.28.

The probability distribution as a function of x was calculated from

$$p(x) = \int_{y_{min}}^{y_{max}} p(x, y)\mathrm{d}y. \tag{3.7}$$

The distributions calculated from 3.7 at different times is shown in figure

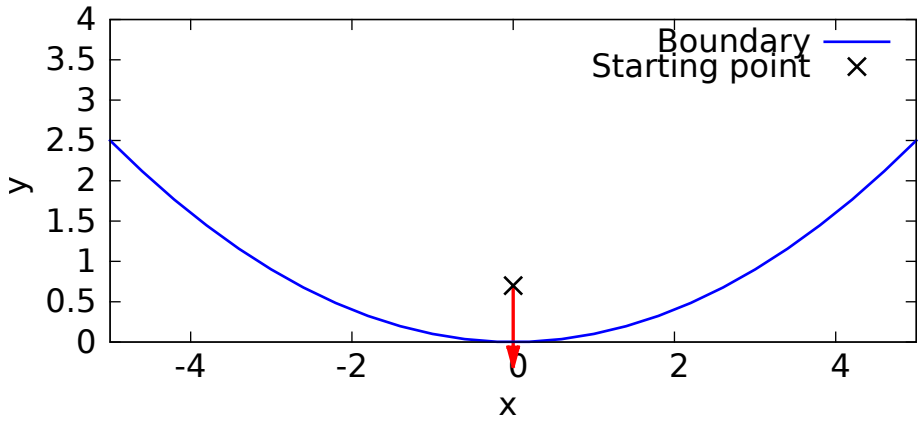

Figure 3.21: The system with a boundary at $y = 0.1x^2$. The starting position is $x_0 = 0$, $y_0 = 0.7$. The red arrow shows the drift, $v_x = 0$, $v_y = -1$.

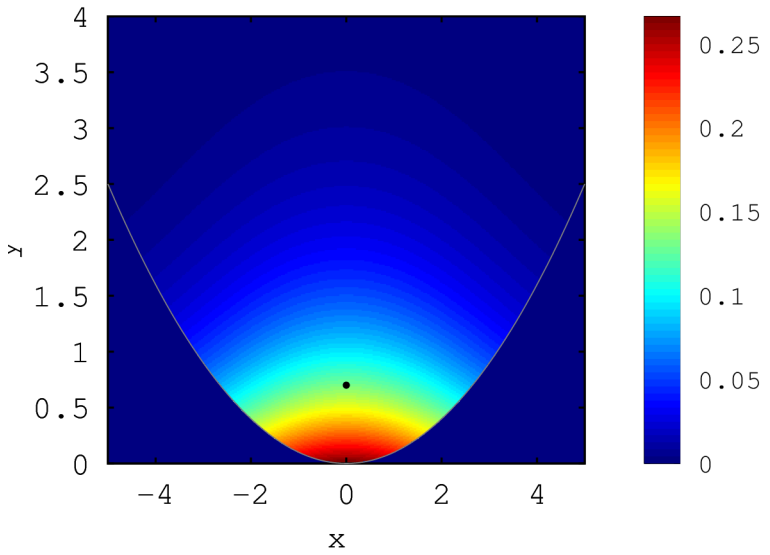

Figure 3.22: $y = 0.1x^2$, $x_0 = 0$, $y_0 = 0.7$, $t = 2$, $v_x = 0$, $v_y = -1$, $\Delta x = 0.025$ and $\Delta y = 0.013$. The black dot is the starting point.

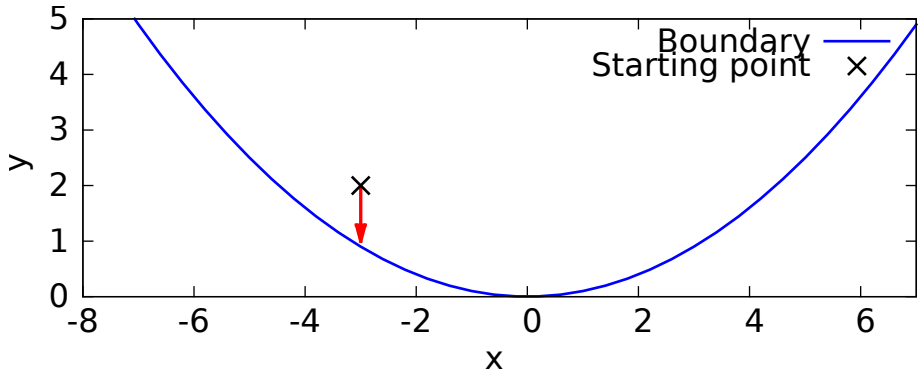

Figure 3.23: The system with a boundary at $y = 0.1x^2$. The starting position is $x_0 = -3$, $y_0 = 2$. The red arrow shows the drift, $v_x = 0$, $v_y = -1$.

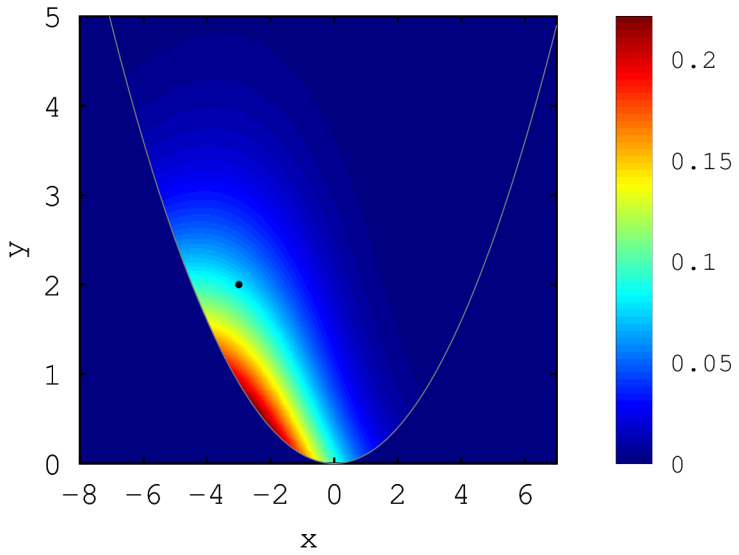

Figure 3.24: $y = 0.1x^2$, $x_0 = -3$, $y_0 = 2$, $t = 2$, $v_x = 0$, $v_y = -1$, $\Delta x = 0.025$ and $\Delta y = 0.015$. The black dot is the starting point.

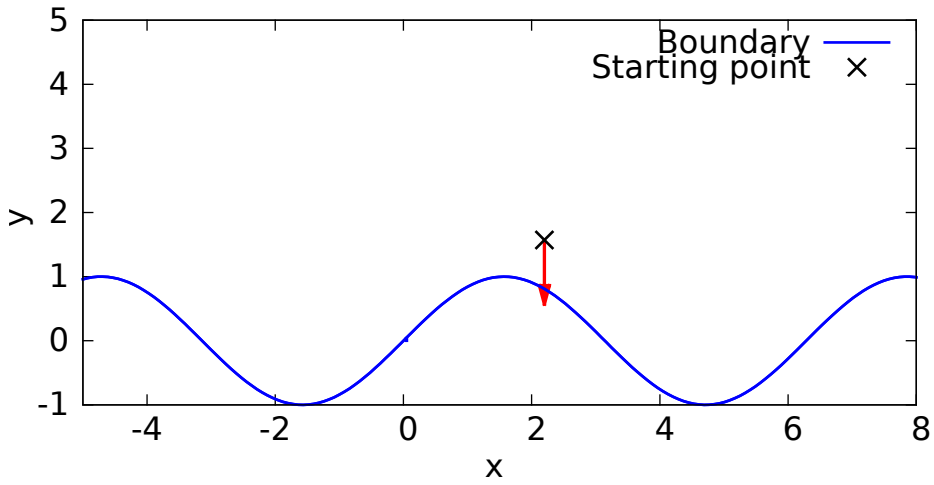

Figure 3.25: The system with a boundary at $y = \sin(x)$. The starting position is $x_0 = 2.2$, $y_0 = 1.5$. The red arrow shows the drift $v_x = 0$, $v_y = -1$.

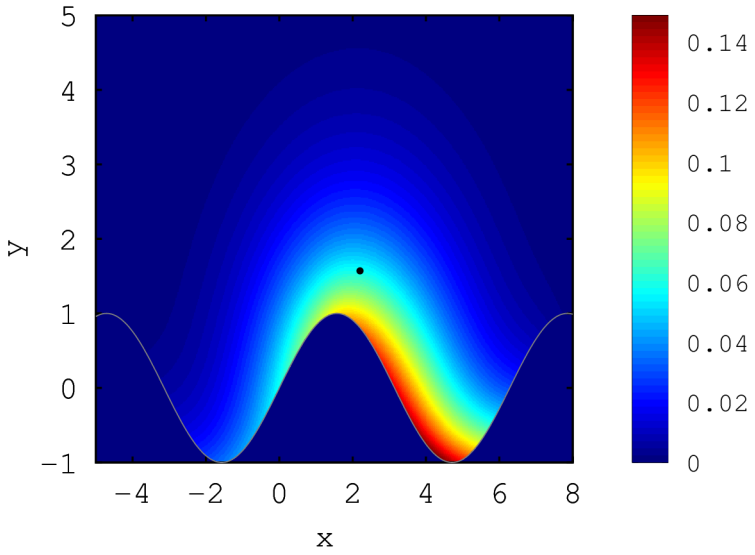

Figure 3.26: The probability distribution at $t = 2$ for the system in figure 3.25. The number of grid points used where $n_x = n_y = 1000$ which gives $\Delta x = 0.025$ and $\Delta y = 0.015$. The black dot is the starting point.

3.29. The stationary distribution is almost flat. It would be completely flat without the channel, so the channel doesn't affect the solution so much in this case. The distribution starts at $x_0 = 7.9$ and $y_0 = 0.7$. The domain is from -7.9 to 23.5 in the x-direction (one period) and from 0 to 2.1 in the y-direction. Using $n_x = 500$ and $n_y = 250$ gives the step size $\Delta x = 0.063$ and $\Delta y = 0.0084$. At $t = 20$ parts of the distribution have passed the first opening. At $t = 40$ parts of the distribution have reached the second opening. The solution is almost in stationary at $t = 80$.

The numerical inaccuracy is visible near the openings. It comes from the positioning of the grid point near the boundary. The distribution at $t = 20$ with different step sizes Δx and Δy is shown in figure 3.30. The step size in the y-direction affects the solution a lot, while decreasing the step size in the x-direction makes a very small difference. Reducing the step size in y by half makes a the boundary much smoother, see figure 3.32, while reducing the step size in x makes almost no difference, see figure 3.31. The step size in the x-direction was therefore taken to be larger than the step size in the y-direction.

With a smaller drift $v_x = 0.3$ the stationary distribution is less flat see figure 3.33. The distributions for $t = 80$, $t = 100$ and $t = 120$ are almost identical. Figure 3.34 shows the stationary distribution for different velocities. The smaller the drift the more the distribution is affected by the boundary above. The place where it is the most likely to find the particle is just before an opening. The narrowing gap slows the particle down. The smaller the force the longer the time it will on average take for the particle to get through the opening.

The system with a larger channel opening is shown in figure 3.35. The stationary distributions for the two different sizes of the channel opening $A-b$ is shown in figure 3.36. The larger the opening, the flatter the distribution, as expected since a larger opening makes the distribution less affected by the upper boundary.

The system with a smaller period is shown in figure 3.37. The upper boundary is at $y = \sin(0.3x) + 1.1$. The probability distributions calculated from (3.7) at different times is shown in figure 3.38. The step size used is $\Delta x = 0.042$ and $\Delta y = 0.023$. The system is almost at stationary at $t = 60$. The distribution thus approaches stationary faster than when $c = 0.2$, as expected since there is less space to diffuse over.

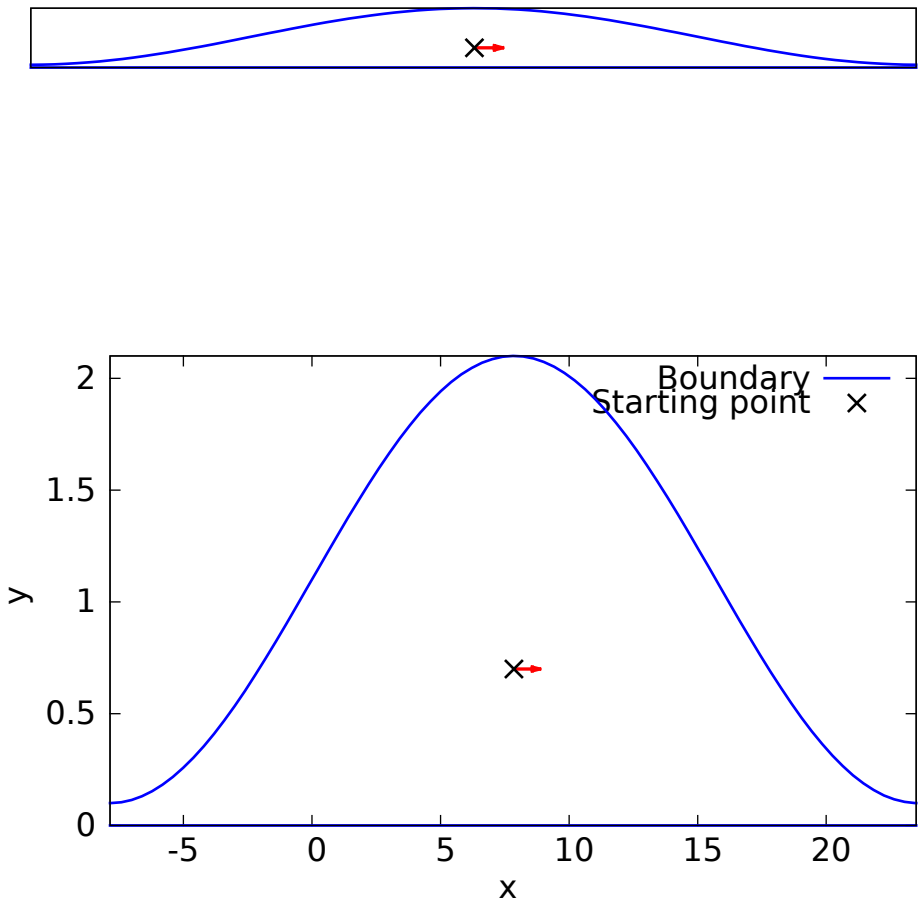

Figure 3.27: The system with a periodic channel with a boundary at $y = \sin(0.2x) + 1.1$. The starting position is $x_0 = 7.9$, $y_0 = 0.7$. The red arrow shows the drift, $v_x = 1$, $v_y = 0$. The picture on top shows the system with the scale in the x-axis equal to the scale in the y-axis.

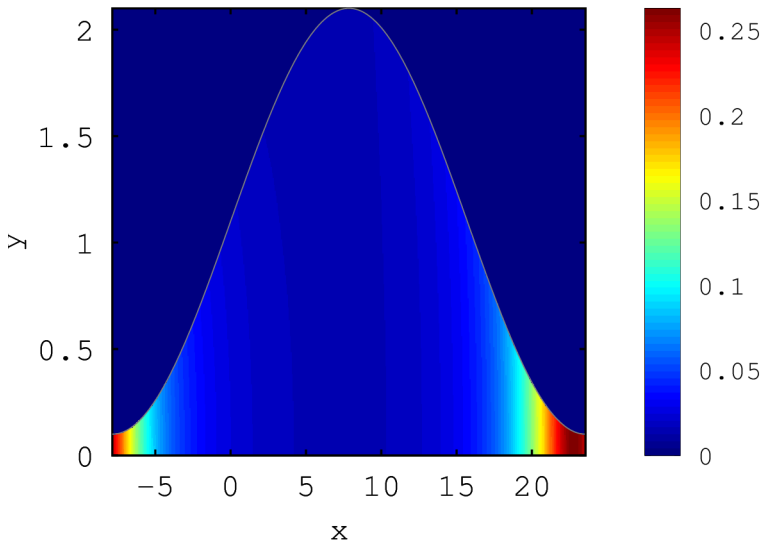

Figure 3.28: The probability distribution at $t = 100$ for the system shown in figure 3.27. The step size is $\Delta x = 0.063$ and $\Delta y = 0.0084$.

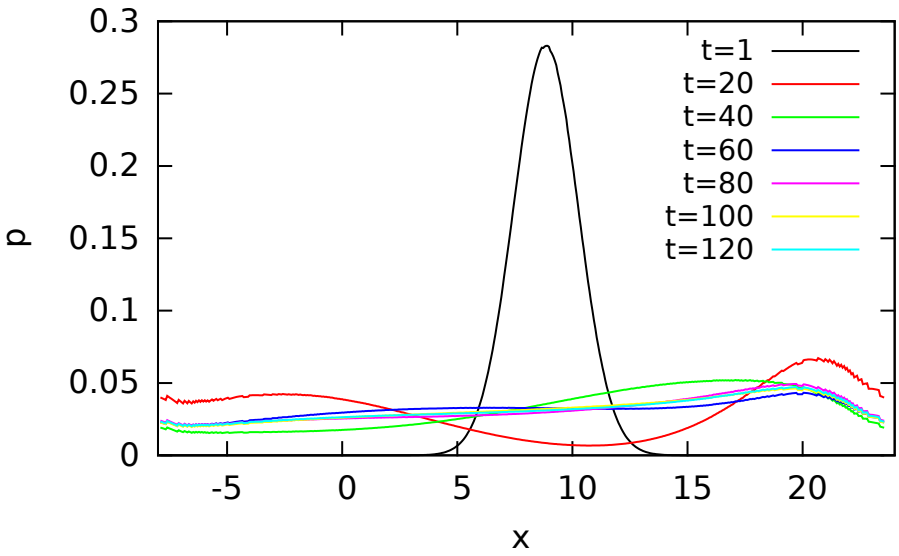

Figure 3.29: The probability as a function of x calculated from (3.7) for the system shown in 3.27 for different times. The step size is $\Delta x = 0.063$ and $\Delta y = 0.0084$. The system is close to stationary at $t = 80$. After that it changes very little.

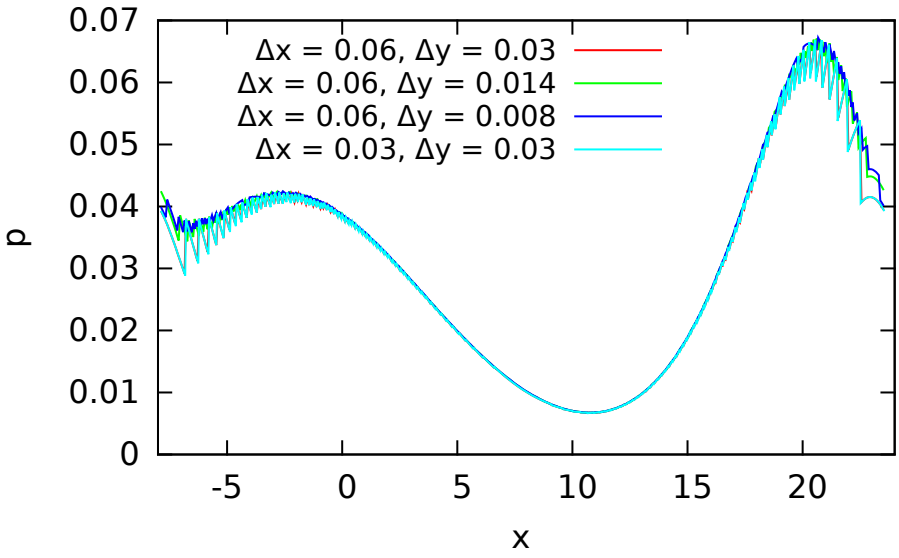

Figure 3.30: The distribution calculated from (3.7) at $t = 20$ with different step sizes Δx and Δy. Decreasing the step size in the y-direction changes the solution more than if one decreases the step size in the x-direction.

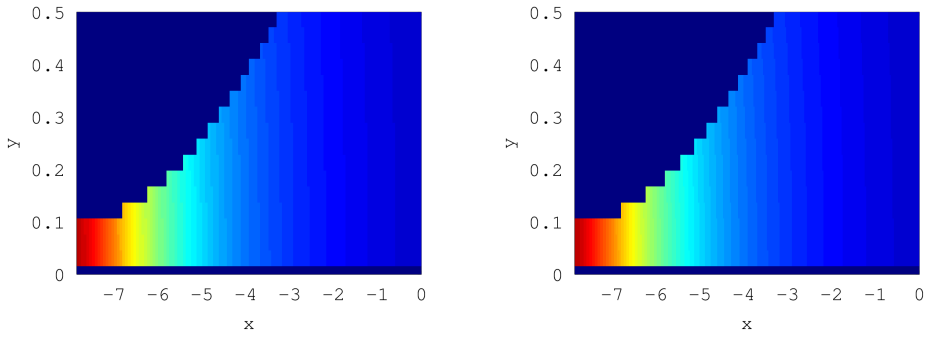

Figure 3.31: Probability distributions at $t = 20$ with $\Delta y = 0.03$. The plot to the left has $\Delta x = 0.06$ and the plot to the right has $\Delta x = 0.03$. There is no visible difference between the pictures.

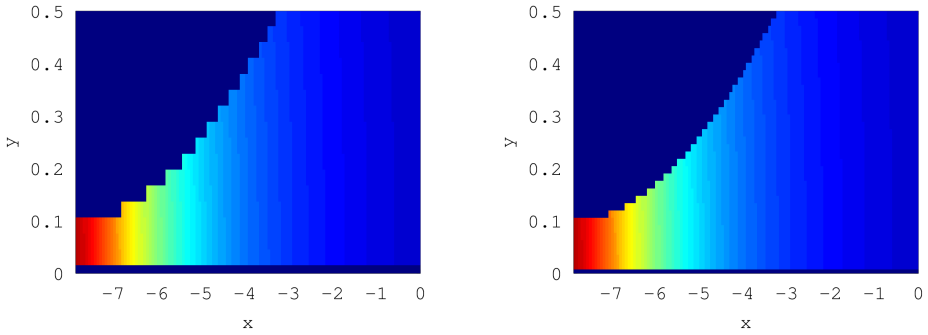

Figure 3.32: Probability distributions at $t = 20$ with $\Delta x = 0.06$. The plot to the left has $\Delta y = 0.03$ and the plot to the right has $\Delta y = 0.014$. The upper boundary becomes much smother with a smaller Δy.

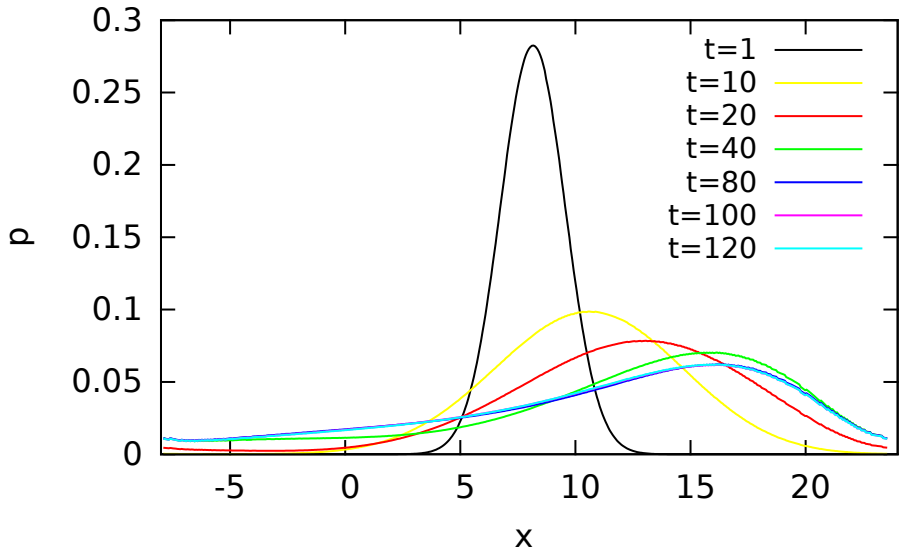

Figure 3.33: The same system as in 3.27 but with $v_x = 0.3$ instead of 1. The smaller force makes the stationary distribution much larger at the right opening. The step size is $\Delta x = 0.063$ and $\Delta y = 0.0084$.

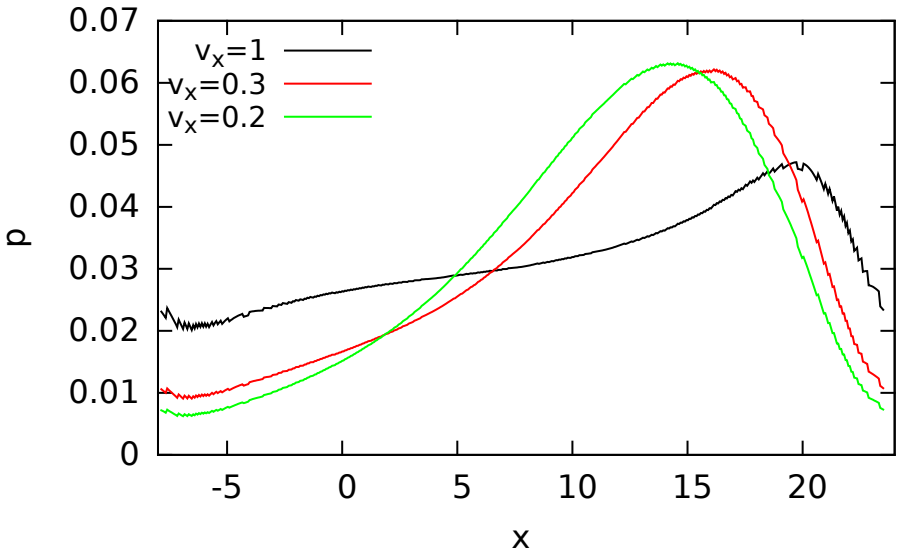

Figure 3.34: The stationary distributions ($t = 120$) calculated from (3.7) for three different velocities for the same system as in 3.27. The larger the force, the flatter the stationary distribution is.

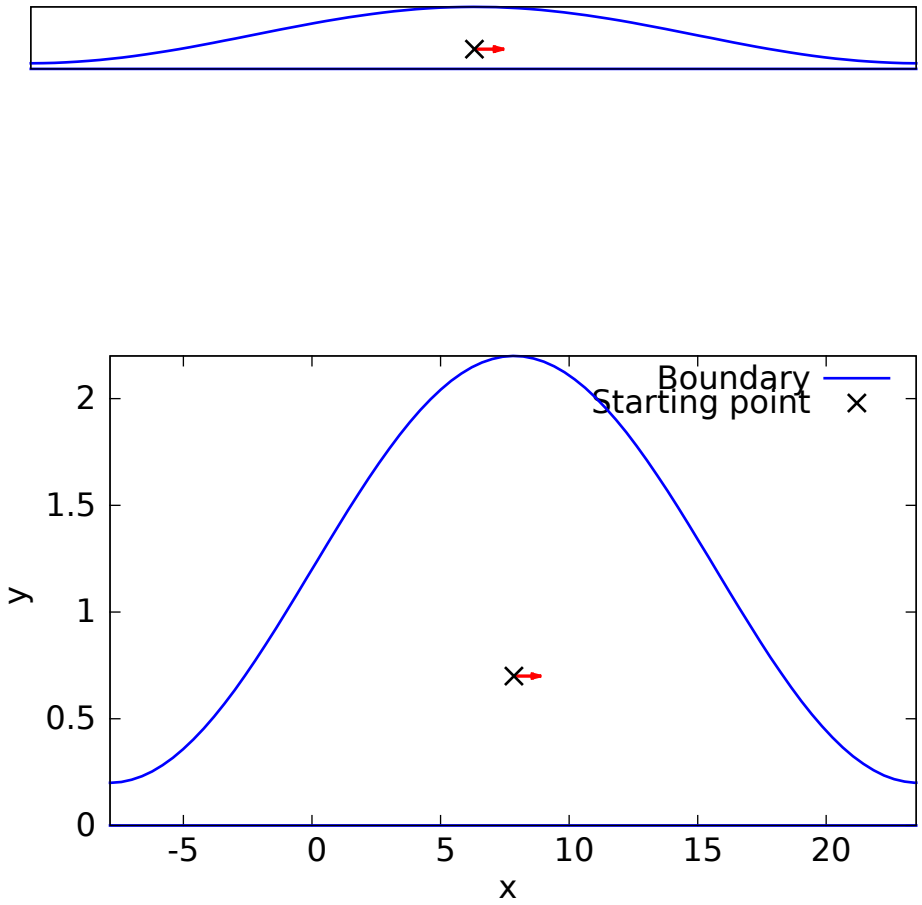

Figure 3.35: The system with a periodic channel with an upper boundary at $y = \sin(0.3x) + 1.2$. The starting position is $x_0 = 7.9$, $y_0 = 0.7$. The red arrow shows the drift, $v_x = 1$, $v_y = 0$. The picture on top shows the system to scale.

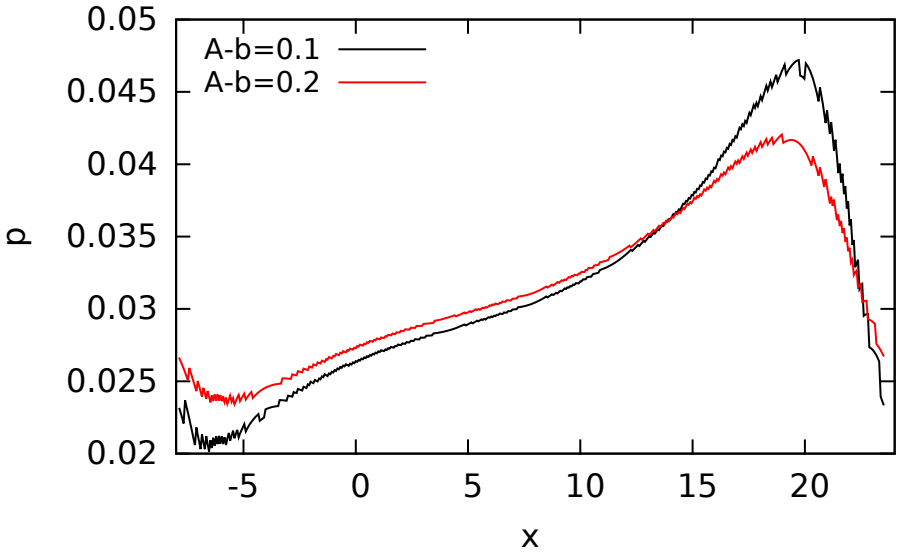

Figure 3.36: The stationary distributions ($t = 120$) for different sizes of the channel openings. The system for $A - b = 0.1$ is shown in 3.27 and the system for $A - b = 0.2$ is shown in 3.35. The stationary distributions become flatter with a larger channel opening.

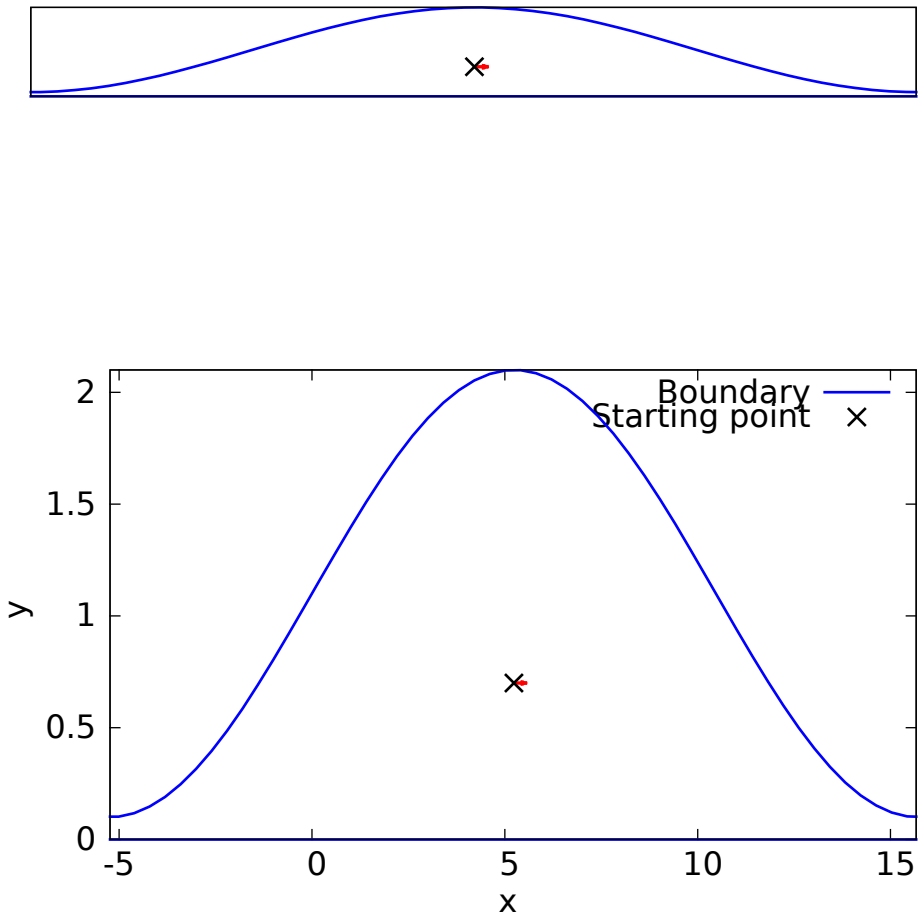

Figure 3.37: The system with a periodic channel with an upper boundary at $y = (\sin(0.3x) + 1.1)$. The starting position is $x_0 = 5.2$, $y_0 = 0.7$. The red arrow shows the drift, $v_x = 0.3$, $v_y = 0$. The period is shorter and the drift smaller than in figure 3.27. The picture on top shows the system to scale.

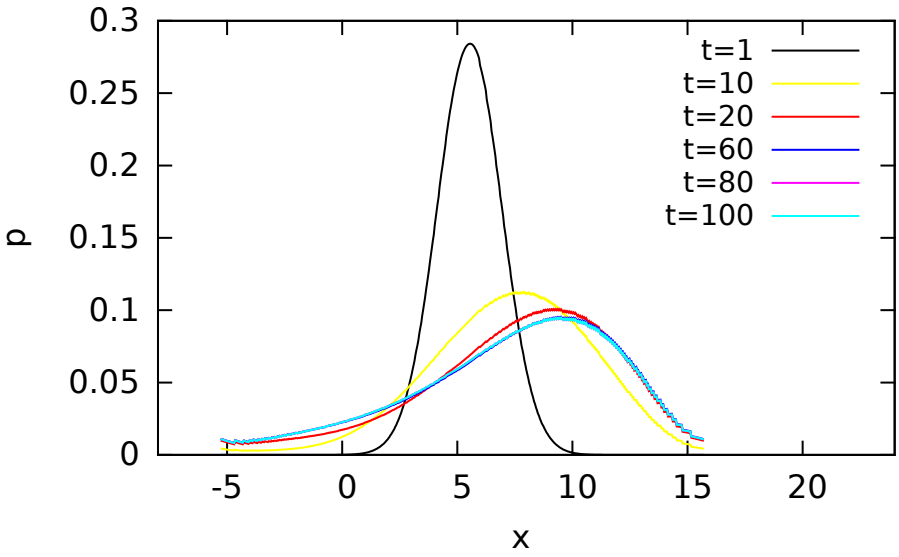

Figure 3.38: The probability distributions integrated over y calculated from (3.7) at different times for the system shown in 3.37. A shorter period makes the system reach stationary faster.

Chapter 4

Conclusions and Outlook

In this thesis a program has been developed to study the diffusive motion of a Brownian particle with a driving force in the presence of reflecting walls. For one and two line boundaries there were a perfect agreement between the numerical and analytical result. The stationary solution was unstable, although very weakly, which probably is a result of the Crank-Nicolson's method. This instability will only affect the solution if the spatial step sizes are too small, so the numerical results are still useful.

In section 3.4 it was shown that the program works for boundaries made up of parabolic functions and sine functions. The solution of a few different channels with periodically varying cross section were shown in section 3.5. The result looked reasonable. The code is suited to compare to the Fick-Jacobs equation and similar approximations to test their validity.

The program can be extended in many ways. It would be useful to be able to change the resolution of the initial solution using interpolation. Then one could run first a simulation with a fine grid and then, when the solution has spread out, do another simulation with a larger step size.

More complicated boundary shapes can be made possible. The program can easily be extended to work for boundary shapes that can't be written as $y = f(x)$, for example a sphere.

One way to extend the program is to allow for space dependent coefficients, that is allowing for the diffusion coefficient and drift to be space dependent. This would make the difference equations slightly more complicated, but would otherwise not change the algorithm.

One other way to extend the program is to make it work for time dependent coefficients. Then the matrices $\tilde{A}$ and $\tilde{B}$ in equation (2.27) would need be to changed at every time step, which would increase the computation time.

Allowing for time dependent boundaries would be even more computa-

tionally expensive. One would then need to calculate the boundary condition at each time step, which would increase the computation time a lot.

One important extension would be to allow for ellipsoidal particles instead of only spherical ones. Many particles of interest in biological systems have non-spherical shapes like proteins [15], DNA fragments and cells. With ellipsoidal particles the rotation of the particle has to be taken into account. The rotation makes the problem (2+1) dimensional. One needs to solve the time dependent joint probability density of the combined particle rotation and translation. The grid and the matrices will be then three dimensional. The Grid class already works for three dimensions and changing the matrix classes to work with one more dimension would be easy. The boundary conditions would be much more difficult to implement.

Appendices

Appendix A

Calculation of Boundary Condition Coefficients

With a set of points $(x_0, y_0), ..., (x_k, y_k)$ the Lagrangian interpolation polynomial can be written

$$L(x) := \sum_{j=0}^{k} y_j l_j(x), \tag{A.1}$$

with the basis polynomials

$$l_j(x) := \prod_{0 \leq m \leq k, m \neq j} \frac{x - x_m}{x_j - x_m} \tag{A.2}$$

With the points $(x_{i,j+1}, u_{i,j+1})^{(n)}$, $(x_{i+1,j+1}, u_{i+1,j+1})^{(n)}$ and $(x_{i+2,j+1}, u_{i+2,j+1})^{(n)}$ the interpolation polynomial becomes

$$\begin{aligned} L_I(x) = u^{(n)}_{i,j+1} \frac{(x - x_{i+1,j+1})}{(x_{i,j+1} - x_{i+1,j+1})} \frac{(x - x_{i+2,j+1})}{(x_{i,j+1} - x_{i+2,j+1})} + \\ + u^{(n)}_{i+1,j+1} \frac{(x - x_{i,j+1})}{(x_{i+1,j+1} - x_{i,j+1})} \frac{(x - x_{i+2,j+1})}{(x_{i+1,j+1} - x_{i+2,j+1})} + \\ + u^{(n)}_{i+2,j+1} \frac{(x - x_{i,j+1})}{(x_{i+2,j+1} - x_{i,j+1})} \frac{(x - x_{i+1,j+1})}{(x_{i+2,j+1} - x_{i+1,j+1})} \end{aligned} \tag{A.3}$$

With $x = x_I$

$$\begin{aligned} L_I(x_I) = u^{(n)}_{i,j+1} \frac{(x_I - x_{i+1,j+1})}{(x_{i,j+1} - x_{i+1,j+1})} \frac{(x_I - x_{i+2,j+1})}{(x_{i,j+1} - x_{i+2,j+1})} + \\ + u^{(n)}_{i+1,j+1} \frac{(x_I - x_{i,j+1})}{(x_{i+1,j+1} - x_{i,j+1})} \frac{(x_I - x_{i+2,j+1})}{(x_{i+1,j+1} - x_{i+2,j+1})} + \end{aligned}$$

$$+u^{(n)}_{i+2,j+1}\frac{(x_I - x_{i,j+1})}{(x_{i+2,j+1} - x_{i,j+1})}\frac{(x_I - x_{i+1,j+1})}{(x_{i+2,j+1} - x_{i+1,j+1})} \tag{A.4}$$

Using $L_I(x_I) = u^{(n)}_I$ and that the distance between the grid points is Δx

$$\begin{aligned} u^{(n)}_I = u^{(n)}_{i,j+1}\frac{(x_I - x_{i+1,j+1})}{(-\Delta x)}\frac{(x_I - x_{i+2,j+1})}{(-2\Delta x)}+ \\ +u^{(n)}_{i+1,j+1}\frac{(x_I - x_{i,j+1})}{\Delta x}\frac{(x_I - x_{i+2,j+1})}{(-\Delta x)}+ \\ +u^{(n)}_{i+2,j+1}\frac{(x_I - x_{i,j+1})}{2\Delta x}\frac{(x_I - x_{i+1,j+1})}{\Delta x} \end{aligned} \tag{A.5}$$

With $x_I - x_{i,j+1} = \eta_I$

$$\begin{aligned} u^{(n)}_I = u^{(n)}_{i,j+1}\frac{(\Delta x - \eta_I)(2\Delta x - \eta_I)}{2(\Delta x)^2}+ \\ +u^{(n)}_{i+1,j+1}\frac{\eta_I(2\Delta x - \eta_I)}{(\Delta x)^2} + u^{(n)}_{i+2,j+1}\frac{-\eta_I(\Delta x - \eta_I)}{2(\Delta x)^2} \end{aligned} \tag{A.6}$$

This gives the first three coefficients in equation (2.20):

$$c_0 = \frac{(\Delta x - \eta_I)(2\Delta x - \eta_I)}{2(\Delta x)^2}, \quad c_1 = \frac{\eta_I(2\Delta x - \eta_I)}{(\Delta x)^2}, \quad c_2 = \frac{-\eta_I(\Delta x - \eta_I)}{2(\Delta x)^2} \tag{A.7}$$

To approximate the normal derivative, use the points $(0, u^{(n)}_{i,j})$, $(\xi_I, u^{(n)}_I)$ and $(\xi_{II}, u^{(n)}_{II})$

$$L(\xi) = u^{(n)}_{i,j}\frac{(\xi - \xi_I)}{(0 - \xi_I)}\frac{(\xi - \xi_{II})}{(0 - \xi_{II})}+u^{(n)}_I\frac{(\xi - 0)}{(\xi_I - 0)}\frac{(\xi - \xi_{II})}{(\xi_I - \xi_{II})}+u^{(n)}_{II}\frac{(\xi - 0)}{(\xi_{II} - 0)}\frac{(\xi - \xi_I)}{(\xi_{II} - \xi_I)} \tag{A.8}$$

Calculate the derivative at the boundary,

$$\begin{aligned} \left.\frac{\partial L}{\partial \xi}\right|_{\xi=\xi_\Gamma} = u^{(n)}_{i,j}\frac{1}{(-\xi_I)}\frac{(\xi_\Gamma - \xi_{II})}{(-\xi_{II})} + u^{(n)}_{i,j}\frac{(\xi_\Gamma - \xi_I)}{(-\xi_I)}\frac{1}{(-\xi_{II})}+ \\ +u^{(n)}_I\frac{1}{\xi_I}\frac{(\xi_\Gamma - \xi_{II})}{(\xi_I - \xi_{II})} + u^{(n)}_I\frac{\xi_\Gamma}{\xi_I}\frac{1}{(\xi_I - \xi_{II})}+ \\ +u^{(n)}_{II}\frac{1}{\xi_{II}}\frac{(\xi_\Gamma - \xi_I)}{(\xi_{II} - \xi_I)} + u^{(n)}_{II}\frac{\xi_\Gamma}{\xi_{II}}\frac{1}{(\xi_{II} - \xi_I)} \end{aligned} \tag{A.9}$$

With $\xi_{II} = 2\xi_I$

$$\left.\frac{\partial L}{\partial \xi}\right|_{\xi=\xi_\Gamma} = u^{(n)}_{i,j}\frac{1}{\xi_I}\frac{(\xi_\Gamma - 2\xi_I)}{(2\xi_I)} + u^{(n)}_{i,j}\frac{(\xi_\Gamma - \xi_I)}{\xi_I}\frac{1}{(2\xi_I)}+$$

$$+u_I^{(n)}\frac{1}{\xi_I}\frac{(\xi_\Gamma-2\xi_I)}{(\xi_I-2\xi_I)}+u_I^{(n)}\frac{\xi_\Gamma}{\xi_I}\frac{1}{(\xi_I-2\xi_I)}+$$
$$+u_{II}^{(n)}\frac{1}{2\xi_I}\frac{(\xi_\Gamma-\xi_I)}{(2\xi_I-\xi_I)}+u_{II}^{(n)}\frac{\xi_\Gamma}{2\xi_I}\frac{1}{(2\xi_I-\xi_I)} \tag{A.10}$$

Collecting the terms

$$\left.\frac{\partial L}{\partial \xi}\right|_{\xi=\xi_\Gamma}=u_{i,j}^{(n)}\frac{(2\xi_\Gamma-3\xi_I)}{2\xi_I^2}+u_I^{(n)}\frac{(2\xi_I-2\xi_\Gamma)}{\xi_I^2}+u_{II}^{(n)}\frac{(2\xi_\Gamma-\xi_I)}{2\xi_I^2} \tag{A.11}$$

This gives the coefficients in equation (2.18)

$$g_0=\frac{(2\xi_\Gamma-3\xi_I)}{2\xi_I^2},\quad g_I=\frac{(2\xi_I-2\xi_\Gamma)}{\xi_I^2},\quad g_{II}=\frac{(2\xi_\Gamma-\xi_I)}{2\xi_I^2} \tag{A.12}$$

References

1. Mazo, Robert M., Brownian motion: fluctuations, dynamics, and applications, Oxford University Press, Oxford, 2009[2002]

2. R. Brown. *A brief account of microscopical observations made in the months of June, July and August 1827, on the particles contained in the pollen of plants; and on the general existence of active molecules in organic and inorganic bodies*, Philosophical Magazine Series 2, **4(21)**, 161-173, 1828.

3. A. Einstein. *Über die von der molekularkinetischen theorie der wärme geforderte bewegung von in ruhenden üssigkeiten suspendierten teilchen*, Annalen der physik, **322(8)**, 549-560, 1905.

4. M. Smoluchowski. *Zur kinetischen theorie der brownschen molekularbewegung und der suspensionen*, Annalen der physik, **326(14)**, 756-780, 1906.

5. P. Langevin. *Sur la théorie du mouvement brownien*, C. R. Acad. Sci. Paris, **146**, 530-533, 1908.

6. Haug, Rolf., Advances in Solid State, Physics Springer Berlin Heidelberg, Berlin, Heidelberg, 2009

7. Jacobs, Merkel H., Diffusion Processes, Springer, 1967

8. Hille, Bertil, Ion channels of excitable membranes, 3. ed., Sinauer Associates, Sunderland, Mass., 2001

9. H. Behringer and R. Eichhorn, *Brownian dynamics simulations with hard-body interactions: Spherical particles*, J. Chem. Phys, **137**, 1641108 (2012).

10. Schulten, K. and Kosztin, I.. Lectures in Theoretical Biophysics. 2000

11. K. Kawasaki, *Simple derivations of generalized Langevin equations*, J. Phys. A: Math Nucl. Gen. 6:1289, (1973).

12. Edsberg, Lennart, Introduction to computation and modeling for differential equations, Wiley, Hoboken, N.J., 2008

13. Press, William H., Teukolsky, Saul A., Vetterling, William T. and Flannery, William B. . Numerical Recipes in C, Cambridge University Press 2002

14. H. Kreiss, N. A. Petersson and J. Yström, *Difference Approximations of the Neumann Problem for Second Order Wave Equation*, SIAM Journal on Numerical Analysis. **42(3)**, 1292-1323 (2004)

15. Alberts, Bruce, Molecular biology of the cell, 4. ed., Garland Science, New York, 2002