The expected exit time in applications

A two-sided approach

A thesis
submitted in partial fulfilment
of the requirements for the degree
of
Master of Science
at the
University of Leiden
by

Martijn Onderwater

Department of Mathematics

Leiden, The Netherlands

June 13, 2003

University of Leiden

Department of Mathematics

The undersigned hereby certify that they have read and recommend for acceptance a thesis entitled "**The expected exit time in applications. A two-sided approach**" submitted by **Martijn Onderwater** in partial fulfillment of the requirements for the degree of **Master of Science**.

Supervisor:

Prof. Dr. S. M. Verduyn Lunel _____

First reader:

Prof. Dr. R. van der Hout _____

Second reader:

Dr. J. A. van de Griend _____

Third reader:

Prof. Dr. R. Tijdeman _____

June 13, 2003

Abstract

Consider the release into the air of a contaminated particle that could be harmful to nearby wildlife and agriculture. To understand the effect of this particle on the environment, it becomes important to find out whether it hits the ground and how long it takes to do so. Mathematically speaking, we are searching for the *the exit probability* and *the expected exit time*. Observations show that the particle doesn't just move along with the wind. It also performs some random motion which can cause it to move against the wind. This has to be taken into account when choosing a model to describe the movement of the particle.

In this thesis we look at the situation described above and also at other practical examples. Our main focus will be on finding the exit probability and the expected exit time. We start with an introduction to stochastic differential equations (SDE's), because the models we consider will be in that form. Connected with each SDE is a partial differential equation called the Fokker-Planck (FP) equation. This FP-equation will then lead us to our first means to obtain the expected exit time. Next, we introduce the numerical simulation of SDE's and this will provide us with a second way to obtain the expected exit time. Finally, four examples are chosen from non-mathematical research areas. Both approaches to finding the expected exit time will be applied and the results will be compared. The differential equations that arise in the second approach are approximated using singular perturbations. All results can be verified with the MATLAB code from the appendix. The examples we look at are a population model from biology, a membrane voltage model from neurology, a particle movement model from physics and a model for groundwater pollution from hydrology.

Acknowledgements

I would like to thank my parents for giving me the opportunity of a good education. Also I want to thank them and the rest of my family for their continuous support. Finally, I would like to thank my supervisor, Prof. Dr. S.M. Verduyn Lunel, for his advise and guidance.

Martijn Onderwater,

Leiden, the Netherlands
June 13, 2003.

Contents

List of Figures

List of Tables

Chapter 1

Introduction: What is Noise?

Nature is a wonderful thing. Not only for the family picnicking in the park or for Greenpeace members, but also for mathematicians. Nature shows us many interesting phenomena which we would like to explain and predict as good as possible. Take the weather for instance. For centuries, sailors have used their senses and experience to forecast the coming of bad weather. It was a necessary skill, since it helped to keep them alive and therefore it was passed on from father to son. With the development of mathematics, more and more people tried to grasp the concept of 'weather' in a set of mathematical equations. Recently, the discovery of the computer enabled us to make the models more complicated and also a lot more accurate. But in spite of all the progress we have made, the weather is still able to reduce our forecasts to toilet paper. This brings us to one of the key aspects of mathematical modelling : the translation from a physical problem to a set of mathematical equations is never perfect. This is due to a combination of uncertainties, complexities and ignorance on our part which inevitably cloud the modelling process.

One way to include these aspects in the model, is by adding a source of randomness (also called noise). Suppose that we want the quantity $X(t)$ to be the mathematical representation of some physical phenomenon and that, at some point in the modelling process, we found a model of the form

$$\frac{dX(t)}{dt} = f\left(t, X(t)\right). \tag{1.1}$$

The addition of a noise term to e.g. one of the parameters changes (1.1) to

$$\frac{dY(t)}{dt} = f\left(t, Y(t)\right) + g\left(t, Y(t)\right)\zeta(t) \tag{1.2}$$

where $\zeta(t)$ is the noise term and $Y(t)$ is the new (noise-dependent) quantity describing the

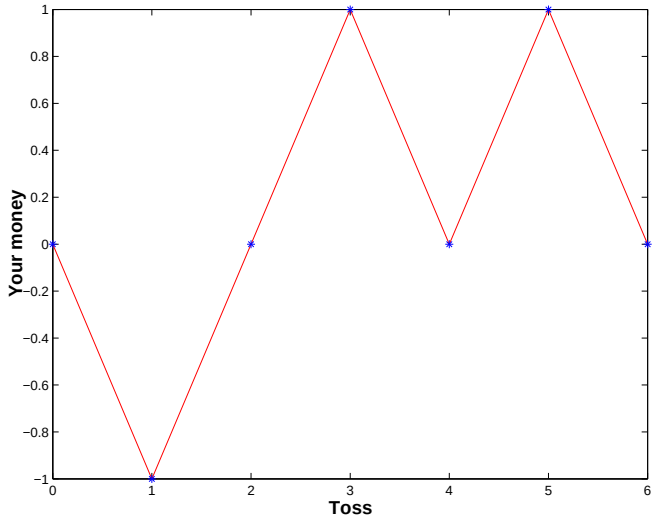

Figure 1.1: The result of the coin-tossing game

phenomenon. This of course leaves us with the question how to model $\zeta(t)$. One model which is often used, comes from a coin-tossing game.

Suppose that we play this coin-tossing game together. Every time you throw a head I give you €1, every time you throw a tail you give me €1. After six tosses (you're not allowed more than six), the sequence was THHTHT and we finished even (as illustrated in Figure 1.1). If I use R_i to mean the random amount, either €1 or $-$€1, you make on the ith toss then we have

$$\mathbb{E}(R_i) = 1 \cdot \mathbb{P}(R_i = 1) + -1 \cdot \mathbb{P}(R_i = -1) = \frac{1}{2}(1-1) = 0, \tag{1.3a}$$

$$\mathbb{E}(R_i^2) = (1)^2 \cdot \mathbb{P}(R_i = 1) + (-1)^2 \, \mathbb{P}(R_i = -1) = \frac{1}{2}(1+1) = 1, \tag{1.3b}$$

$$\mathbb{E}(R_i R_j) = \mathbb{E}(R_i)\mathbb{E}(R_j) = 0. \tag{1.3c}$$

Introduce S_i to mean the amount of money you have made up to and including the ith toss, so that

$$S_i = \sum_{j=1}^{i} R_j.$$

We assume that you start with no money, so $S_0 = 0$. If we now calculate expectations (before the game started BTW) we find (using (1.3))

$$\mathbb{E}(S_i) = 0 \quad \text{and} \quad \mathbb{E}\left(S_i^2\right) = \mathbb{E}\left(\left(\sum_{j=1}^{i} R_j\right)^2\right) = \sum_{j=1}^{i} \mathbb{E}\left(R_j^2\right) = i.$$

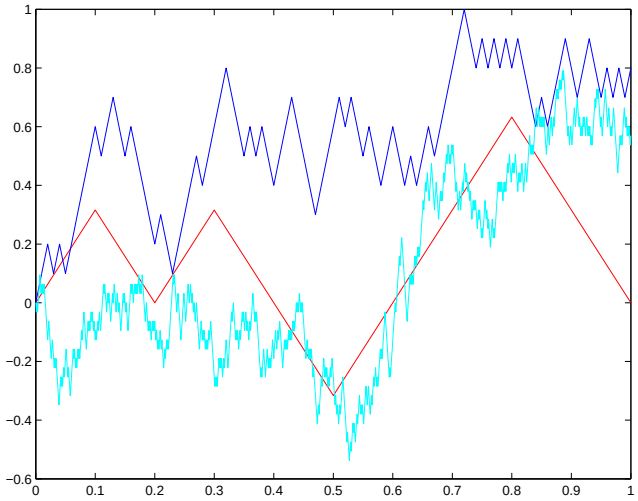

Figure 1.2: The results of the new coin-tossing game

Now I'm going to change the rules. First, I restrict the time to a period t and thus each throw of the coin takes a time $\frac{t}{6}$. Secondly, the size of the bet is changed from €1 to $\sqrt{\frac{t}{6}}$. Thus

$$\mathbb{E}(S_i) = 0 \quad \text{and} \quad \mathbb{E}\left(S_i^2\right) = i \cdot \left(\sqrt{\frac{t}{6}}\right)^2 = \frac{it}{6}.$$

Of course, this game will get boring swiftly and therefore I will speed it up a bit. You may now throw the coin as often as you like, say n times. The bet size is changed again, this time to $\sqrt{\frac{t}{n}}$. You still have to be finished after a period t. So we have

$$\mathbb{E}(S_i) = 0 \quad \text{and} \quad \mathbb{E}\left(S_i^2\right) = i \cdot \left(\sqrt{\frac{t}{n}}\right)^2 = \frac{it}{n}. \tag{1.4}$$

Figure 1.2 shows some results of this game, all lasting for a time 1 with different values of n. In Section 3.1 we will show that as $n \to \infty$, the S_i will behave more and more like a stochastic process known as *Brownian motion*. Brownian motion $-$ also called the *Wiener process* and denoted in this document by $W(t)$ $-$ will be treated in Section 2.1. The noise term ζ in (1.2) is then formally modelled as

$$\zeta(t) = \frac{\partial W(t)}{\partial t}.$$

Later, we will also see that Brownian motion is nowhere differentiable and therefore equation (1.2) is usually rewritten as

$$dX(t) = f(t, X(t))dt + g(t, X(t))dW(t)$$

and is called a *stochastic differential equation* (SDE). SDE's will play an important role in this document and will be treated further in Chapter 2.

The history of stochastic differential equations takes us back to the 19th century. In 1827, the Scottish botanist Robert Brown observed that pollen grains suspended in a liquid performed an irregular motion. Later, at the beginning of the 20th century, Albert Einstein and Smoluchowsky introduced random equations and showed how the results of random walk theory could be used to describe the phenomenon. It was about the same time that Langevin proposed a random equation describing the motion of Brownian particles. A year or two later, Campbell, dealing with the fluctuations in the emission of rays by a radioactive substance, proved a couple of theorems relating to the mean square behavior of the cumulative response due to such emissions. These instances mark the birth of stochastic integration and stochastic differential equations. Within a short period, Wiener, anticipating Kolmogorov's formalization of probability, undertook the mathematical analysis of Brownian motion. Since then, stochastic differential equations have found their way into systems in physics [34, 35, 36, 37], engineering [38, 39, 40, 41], biology [19, 20, 24, 25, 26, 27] and economics [5, 12, 13, 14].

This thesis

The aim of this thesis consists of three parts :

1. Show practical examples of stochastic differential equations;
2. Show how simulations of stochastic differential equations can be done on the computer;
3. Show the connection between stochastic differential equations and partial differential equations.

To achieve these objectives, we will consider a SDE and look for the expected exit time from an open domain (this is called 'the exit problem'). It will become clear that the expected exit time can be obtained from a differential equation and also by a simulation of the SDE.

I have tried to make this document in such a way that it is suitable for most students who are in the final stage of their mathematical Masters-education. You will not need any special probability knowledge ; just basic concepts such as expectation, variance and (normally distributed) random variables (see e.g. [1] or [2]). Everything you need to know about SDE will be explained. There will be a lot of differential equations in this thesis, so some experience in this area will be required.

This thesis is divided into 5 chapters, the first of which you have almost finished. The next chapter contains all the theory we will be needing in later chapters. It continues with Brownian motion and then introduces stochastic differential equations. The rest of the chapter is devoted to the Fokker-Planck equation and its use in exit from an open domain. The ideas for this chapter came mostly from a book by Grasman and Van Herwaarden([16]). The third chapter deals with the simulation of SDE. Using the software package 'MATLAB', it shows how to simulate Brownian motion, functions along Brownian paths, SDE's and finally the expected exit time. The simulation of SDE's is centered around the 'Euler-Maruyama' method, which is the stochastic analogue of the deterministic Euler method. The work in this chapter is very much based upon an article by D. Higham, see [44]. Chapter 4 contains four examples of SDE's. For each one, we will be looking for the expected exit time via the Fokker-Planck equation and via a simulation of the system. The solution to the Fokker-Planck equation is approximated using a technique called *singular perturbations* which is explained in that same chapter. The final chapter looks back at the thesis and offers some other interesting areas of research connected to the subjects in this document.

Chapter 2

Stochastic Calculus

In the introduction it was mentioned that there are two ways to find the expected exit time from an open domain : via a simulation of the model and via a differential equation. This chapter will look at the second. To do this, we will first need to find out what stochastic integrals are (Section 2.1). Section 2.2 will then continue with stochastic differential equations. The final section is about the 'differential equation' approach to the exit problem.

2.1 Stochastic integrals

The most important thing we need for stochastic integrals is Brownian motion. We already met it in the introduction and now it is time for a more mathematical formulation. To get started, we need the following definition :

Definition 2.1.1 *A stochastic process is a collection of random variables* $\{X_t\}_{t\in[0,\infty)}$ *assuming values in* $\mathbb{R}^n$. *In this document, it is denoted by* $X(t)$.

Now, one-dimensional Brownian motion $W(t)$ is an example of a random process satisfying:

1. (a) with probability 1, the mapping $t \mapsto W(t)$ is continuous ($t \geq 0$);

   (b) $W(0) = 0$;

2. For all $0 \leq t_0 < t_1 < t_2 < \ldots < t_k$, the increments

$$W(t_k) - W(t_{k-1}), \ldots, W(t_1) - W(t_0)$$

   are independent;

3. for all $t > s$ the increment $W(t) - W(s)$ has the normal distribution with expectation 0 and variance $t - s$, i.e.

$$P(W(t) - W(s) \in \Gamma) = \frac{1}{\sqrt{2\pi(t-s)}} \int_\Gamma e^{\frac{-y^2}{2(t-s)}} dy, \qquad \Gamma \subset \mathbb{R}.$$

For the proof that such a random process exists, we refer the reader to e.g. [6, Ex. 2.18] or [5, p. 12-14]. One property of Brownian motion that we will need is

1. (c) $W(t)$ is nowhere differentiable with probability 1,

see [3, Thm. 12.25] for the proof. We will only need these properties of Brownian motion and therefore we will not go into it any deeper. Interested readers are referred to [3, 5, 6].

A typical stochastic integral will look like

$$\int_0^T g(t, W(t)) dW(t)$$

where $g : [0, T] \times \mathbb{R} \to \mathbb{R}$. How do we evaluate this integral? As a first example of a stochastic integral, let us try to find a suitable discretization for $\int_0^T W(t) dW(t)$. In the deterministic setting, one often approximates integrals using either the *forward Euler*, *backward Euler* or the *trapezoidal* discretization. As the stepsize goes to 0, these approximations converge to the same value. But this is not the case in the stochastic world, as we will see now. Let $t_0, t_1, \ldots, t_N$ be a discretization of the interval $[0, T]$. Then the forward Euler approximation of our integral is given by

$$\sum_{n=0}^{N-1} W(t_n) \underbrace{(W(t_{n+1}) - W(t_n))}_{\Delta W_n}.$$

Taking expected values yields (using property 3 of the Brownian motion)

$$\mathbb{E}\left[\sum_{n=0}^{N-1} W(t_n) \Delta W(t)\right] = \sum_{n=0}^{N-1} \mathbb{E}[W(t_n)] \underbrace{\mathbb{E}[\Delta W_n]}_{=0} = 0.$$

Doing the same for the backward Euler discretization

$$\sum_{n=0}^{N-1} W(t_{n+1}) \Delta W_n$$

yields

$$\begin{aligned}\sum_{n=0}^{N-1}\mathbb{E}[W(t_{n+1})\Delta W_n] &= \sum_{n=0}^{N-1}\mathbb{E}[W(t_n)\Delta W_n]+\mathbb{E}[(\Delta W_n)^2]\\ &= \sum_{n=0}^{N-1}(t_{n+1}-t_n)=T\neq 0.\end{aligned}$$

Moreover, if we use the trapezoidal discretization

$$\sum_{n=0}^{N-1}\mathbb{E}\left[\frac{W(t_{n+1})+W(t_n)}{2}\Delta W_n\right]$$

we find

$$\begin{aligned}\sum_{n=0}^{N-1}\mathbb{E}\left[\frac{W(t_{n+1})+W(t_n)}{2}\Delta W_n\right] &= \sum_{n=0}^{N-1}\mathbb{E}[W(t_n)\Delta W_n]+\mathbb{E}[\Delta W_n/2]\\ &= \sum_{n=0}^{N-1}\left[\frac{t_{n+1}-t_n}{2}\right]=T/2\neq 0.\end{aligned}$$

This shows that we need more than just the integral to compute its expected value : we also need to specify which approximation method to use. The choice of the forward Euler approximation will lead to so called *Itô* integrals, while choosing the trapezoidal approximation will give us *Statonovich* integrals. We will only concern ourselves with the former. The precise definition comes from the following theorem:

Theorem 2.1.1 *Let $g:[0,T]\times\Omega\to\mathbb{R}$ be* adapted *(i.e. $g(t,\cdot)$ only depends on events which are generated by $W(s)$, $s<t$) and satisfy the relation*

$$\sqrt{\mathbb{E}\left[|g(t+\Delta t,\omega)-g(t,\omega)|^2\right]}\leq C\sqrt{\Delta t} \tag{2.1}$$

where C is a constant. Consider two different partitions of the time interval $[0,T]$

$$\{\overline{t}\}_{n=0}^{\overline{N}} \quad \textit{and} \quad \{\overline{\overline{t}}\}_{m=0}^{\overline{\overline{N}}} \tag{2.2}$$

with the corresponding forward Euler approximations

$$\overline{I}=\sum_{n=0}^{\overline{N}-1}g(\overline{t}_n,W(\overline{t}_n))(W(\overline{t}_{n+1})-W(\overline{t}_n)),$$

and

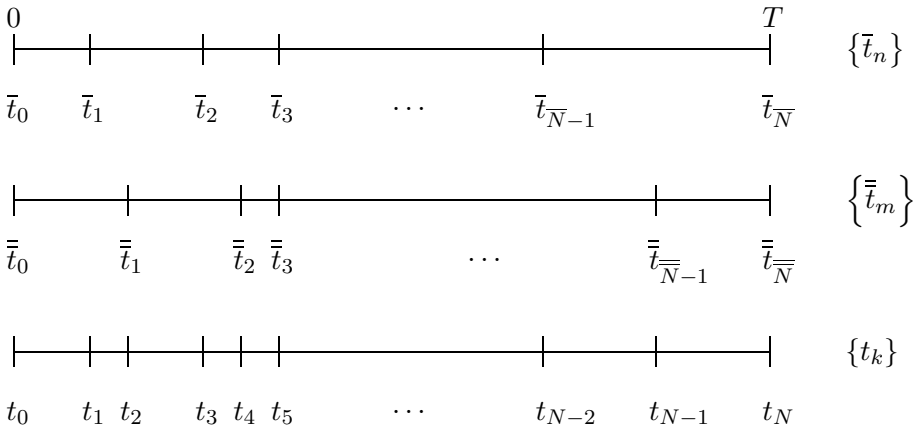

Figure 2.1: The mesh used in the proof

$$\overline{\overline{I}} = \sum_{m=0}^{\overline{\overline{N}}-1} g(\overline{\overline{t}}_m, W(\overline{\overline{t}}_m))(W(\overline{\overline{t}}_{m+1}) - W(\overline{\overline{t}}_m)),$$

and define the maximum stepsize to be

$$\Delta t_{\max} = \max\left[\max_{0\le n\le \overline{N}-1} \overline{t}_{n+1} - \overline{t}_n, \max_{0\le m\le \overline{\overline{N}}-1} \overline{\overline{t}}_{m+1} - \overline{\overline{t}}_m\right].$$

Then

$$\mathbb{E}\left[\left(\overline{I} - \overline{\overline{I}}\right)^2\right] \le O(\Delta t_{\max}).$$

Proof:

It is useful to introduce the finer grid made of the union of the nodes of the grids (2.2)

$$\{t_k\} \equiv \{\overline{t}_n\} \cup \left\{\overline{\overline{t}}_m\right\}.$$

Then in that grid we can write

$$\overline{I} - \overline{\overline{I}} = \sum_k \Delta g_k \Delta W_k,$$

where $\Delta g_k = g(\overline{t}_n, W(\overline{t}_n)) - g(\overline{\overline{t}}_m, W(\overline{\overline{t}}_m))$, $\Delta W_k = W(t_{k+1}) - W(t_k)$ and the indices m and n satisfy $t_k \in \left[\overline{\overline{t}}_m, \overline{\overline{t}}_{m+1}\right)$ and $t_k \in \left[\overline{t}_n, \overline{t}_{n+1}\right)$, as depicted in Figure 2.1.

Therefore,

$$\begin{aligned}
\mathbb{E}\left[(\overline{I}-\overline{\overline{I}})^2\right] &= \mathbb{E}\left[\sum_{k,l}\Delta g_k \Delta g_l \Delta W_k \Delta W_l\right] \\
&= 2\sum_{k>l}\mathbb{E}[\Delta g_k \Delta g_l \Delta W_k \Delta W_l] + \sum_k \mathbb{E}\left[(\Delta g_k)^2 (\Delta W_k)^2\right] \qquad (2.3)\\
&= 0 + \sum_k \mathbb{E}\left[(\Delta g_k)^2 (\Delta W_k)^2\right] = \sum_k \mathbb{E}\left[(\Delta g_k)^2\right]\Delta t_k \qquad (2.4)
\end{aligned}$$

where we have used (in (2.3)) that $\mathbb{E}[\Delta g_k \Delta g_l \Delta W_k \Delta W_l] = \mathbb{E}[\Delta g_k \Delta g_l \Delta W_k]\mathbb{E}[\Delta W_l] = 0$ because of the third property of Brownian motion. Taking squares in (2.1) gives us

$$|\Delta g_k|^2 \leq C^2 \Delta' t_k$$

where $\Delta' t_k = \overline{t}_n - \overline{\overline{t}}_m \leq \Delta t_{\max}$. Substitution of this into (2.4) gives

$$\mathbb{E}\left[(\overline{I}-\overline{\overline{I}})^2\right] \leq C^2 \Delta t_{\max} \sum_k \Delta t_k = C^2 T \Delta t_{\max}$$

which proves the theorem.

□

Thus if we take a sequence $\{I_{\Delta t}\}$ such that $\Delta t \to 0$ we see that it converges (in the sense of result of Theorem 2.2.1). The limit defines the Itô integral

$$\sum_i g_i \Delta W_i \to I \equiv \int_0^T g(s, W(s)) dW(s).$$

Some basic properties of the Itô integral are

1. $\displaystyle\int_0^T (c_1 f(s,\cdot) + c_2 g(s,\cdot)) dW(s) = c_1 \int_0^T f(s,\cdot) dW(s) + c_2 \int_0^T g(s,\cdot) dW(s),$
2. $\displaystyle\mathbb{E}\left[\int_0^T f(s,\cdot) dW(s)\right] = 0,$
3. $\displaystyle\mathbb{E}\left[\left(\int_0^T f(s,\cdot) dW(s)\right)\left(\int_0^T g(s,\cdot) dW(s)\right)\right] = \int_0^T \mathbb{E}[f(s,\cdot) g(s,\cdot)] ds.$

For the proof of these properties, we refer the reader to [11, Thm. 2.15].

2.2 Stochastic Differential Equations

We now advance to the study of *stochastic integral equations*. The general form of a stochastic integral equation is

$$X(t) = X_0 + \int_0^T f(s, X(s))ds + \int_0^T g(s, X(s))dW(s), \tag{2.5}$$

where the first integral is deterministic and the second integral is a stochastic integral (as described in the previous section). It is usual to write (2.5) in its differential form

$$dX(t) = f(t, X(t))dt + g(t, X(t))dW(t), \qquad X(0) = X_0, \qquad 0 \leq t \leq T \tag{2.6}$$

and then it is called a *stochastic differential equation*. Note that if $g(t, X(t)) \equiv 0$, then (2.6) reduces to the deterministic differential equation $dX(t)/dt = f(t, X(t))$. We do *not* take the further step to divide (2.6) by dt, since Brownian motion is nowhere differentiable and thus the quantity $dW(t)/dt$ has no meaning.

Example : A very simple example is the SDE

$$dX(t) = f(t)dt + g(t)dW(t).$$

This can simply be integrated to yield the solution

$$X(t) = X_0 + \int_0^t f(s)ds + \int_0^t g(s)dW(s).$$

Of course life is not always this simple. In the deterministic setting, one often uses 'change of variables' to find a solution to a differential equation. The same tool is available in the stochastic world, but it is not completely the same. The following theorem gives *Itô's formula*:

Theorem 2.2.1 ***(Itô's formula in one dimension)*** *Suppose that the assumptions of Theorem (2.1.1) hold and that $X(t)$ satisfies the SDE (2.6) and let $h : (0, \infty) \times \mathbb{R} \to \mathbb{R}$ be a given bounded function in $C^2((0, \infty) \times \mathbb{R})$. Then $Y(t) = h(t, X(t))$ satisfies the SDE*

$$dY(t) = \frac{\partial h(t, X(t))}{\partial t}dt + \frac{\partial h(t, X(t))}{\partial x}dX(t) + \frac{1}{2}\frac{\partial^2 h(t, X(t))}{\partial x^2}(dX(t))^2. \tag{2.7}$$

For the proof, we refer the reader to e.g. [5, Thm. 4.1.2] or [11, Thm. 3.9]. The term

$(dX(t))^2$ in (2.7) can be evaluated using (2.6) and the rules

$$[dW(t)]^2 = dt, \tag{2.8a}$$

$$[dW(t)]^i = 0 \qquad \forall i > 2, \tag{2.8b}$$

$$dW(t)dt = 0, \tag{2.8c}$$

$$dt^i = 0, \qquad \forall i > 1. \tag{2.8d}$$

Note that change of variables in the stochastic setting (2.7) is the same as in the deterministic setting, except for the addition of the term

$$\frac{1}{2}\frac{\partial^2 h(t, X(t))}{\partial x^2}(dX(t))^2.$$

Example (Ornstein-Uhlenbeck Process) : Suppose we are looking for the solution of the SDE

$$dX(t) = -kX(t)dt + \sqrt{D}dW(t) \tag{2.9}$$

where k and D are constants. We solve this equation directly by making the change of variables $Y(t) = X(t)e^{kt}$. Then, by using (2.8), we get

$$(dX(t))^2 = k^2(X(t))^2(dt)^2 - 2k\sqrt{D}dtdW(t) + D(dW(t))^2 = 0 + 0 + Ddt = Ddt$$

and we find (using (2.7)) that $Y(t)$ satisfies the SDE

$$\begin{aligned} dY(t) &= kX(t)e^{kt}dt + e^{kt}dX(t) + \frac{1}{2}\cdot 0 \cdot D \cdot dt \\ &= kX(t)e^{kt}dt - kX(t)e^{kt}dt + \sqrt{D}e^{kt}dW(t) \\ &= \sqrt{D}e^{kt}dW(t). \end{aligned}$$

Integrating and returning to original variables yields

$$X(t) = X(0)e^{-kt} + \sqrt{D}\int_0^t e^{-k(t-s)}dW(s).$$

The process (2.9) is known as *the Ornstein-Uhlenbeck process*. It often turns up in applications of stochastic calculus ([19, 20, 29]).

Example (geometric Brownian motion) : Consider the SDE

$$dX(t) = \lambda X(t)dt + \mu X(t)dW(t) \tag{2.10}$$

where λ and μ are constants. We set $Y(t) = \ln(X(t))$ and change the variables

$$\begin{aligned} dY(t) &= \frac{1}{X(t)}dX(t) - \frac{1}{2}\frac{1}{X(t)^2}(dX(t))^2 \\ &= \lambda dt + \mu dW(t) - \frac{\mu^2}{2}dt \\ &= (\lambda - \frac{\mu^2}{2})dt + \mu dW(t) \end{aligned}$$

so that

$$Y(t) = Y(0) + (\lambda - \frac{\mu^2}{2})t + \mu\int_0^t dW(t) = Y(0) + (\lambda - \frac{\mu^2}{2})t + \mu W(t)$$

or in original variables

$$X(t) = X(0)e^{(\lambda - \frac{\mu^2}{2})t + \mu W(t))}. \tag{2.11}$$

Equation (2.10) is often called *Geometric Brownian motion* and has played a key role in the development of financial mathematics. The well known *Black Scholes partial differential equation* (see [12, 14]) can be derived from it.

Until now, we have only seen one-dimensional SDE's. Of course, there are higher dimensional SDE's and also a higher dimensional version of Itô's formula.

Theorem 2.2.2 *Let X(t) satisfy the system of SDE's*

$$\begin{cases} dX_1(t) = u_1 dt + v_{11}dW_1(t) + \ldots + v_{1m}dW_m(t) \\ \vdots \qquad\quad \vdots \qquad\quad \vdots \qquad\quad \vdots \\ dX_n(t) = u_n dt + v_{n1}dW_1(t) + \ldots + v_{nm}dW_m(t) \end{cases}$$

where $u_i = u_i(t, X(t))$ *and* $v_{ij} = v_{ij}(t, X(t))$. *Let* $h(t,x)$ *be a* C^2*-map from* $[0,\infty) \times \mathbb{R}^n$ *into* $\mathbb{R}^p$. *Then the process*

$$Y(t) = h(t, X(t))$$

is again an Itô process given by

$$dY_k = \frac{\partial h_k(t,X)}{\partial t}dt + \sum_i \frac{\partial h_k(t,X)}{\partial x_i}dX_i + \frac{1}{2}\sum_{i,j}\frac{\partial^2 h_k(t,X)}{\partial x_i \partial x_j}dX_i dX_j$$

where we can again use the rules (2.8) with (2.8a) replaced by its higher dimensional version

$$dW_i(t)dW_j(t) = \begin{cases} dt & i = j, \\ 0 & i \neq j. \end{cases}$$

The proof of this theorem can be found in [5, Thm. 4.2.1].

Remark : Until now, we have said nothing about uniqueness of solutions. With uniqueness we mean that for two solutions $X_1(t)$ and $X_2(t)$ of the SDE (2.6), using the same sample path of Brownian motion, we have

$$\mathbb{P}\left(\sup_{0\leq t\leq T}|X_1(t) - X_2(t)| > 0\right) = 0.$$

This is called *pathwise* (or *strong*) *uniqueness*. It is known (see [5, Thm. 5.2.1]) that (2.6) has a pathwise unique solution if its coefficients satisfy the following relations

1. *Lipschitz condition :* a constant K exists such that

   $$|f(t,x) - f(t,y)| + |g(t,x) - g(t,y)| \leq K|x - y|$$

   for all finite t and for all x and y.

2. *Growth condition :* a K exists such that for all finite t

   $$|f(t,x)| + |g(t,x)| \leq K(1 + |x|).$$

This section has only covered the basics of SDE's. For a more rigorous treatment, we refer the reader to one of the following textbooks [5, 6, 8, 9, 10].

2.3 The Fokker-Planck Equation and the exit problem

As we mentioned in the introduction and at the beginning of this chapter, we are interested in finding a way to obtain the expected exit time from an open domain when the system is given by a SDE. Now that we know what SDE are, we can investigate this further. We first have to restrict our attention to systems of the form

$$dX_i = b_i(X)dt + \sum_{j=1}^{k} \sigma_{ij}(X)dW_j(t), \quad X_i(0) = X_i^{(0)}, \quad i = 1, \ldots, n \tag{2.12}$$

where of course $X = X(t)$. Observe that (2.12) is time homogeneous, i.e. b_i and σ_{ij} are time-independent. The reason why we only look at time homogeneous systems will become clear later. We denote the open domain by Ω and its boundary bu $\partial\Omega$. In this section, we will show that the expected exit time of the system (2.12) from the domain Ω can be obtained from a deterministic differential equation.

The FP-equation

Consider the one dimensional version of (2.12)

$$dX = b(X)dt + \sigma(X)dW(t), \quad X(0) = X^{(0)}. \tag{2.13}$$

For an arbitrary function f we have (using Itô's formula (Theorem 2.2.1)) :

$$df(X) = f'(X)dX + \frac{1}{2}f''(X)(dX)^2,$$

or

$$df(X) = f'(X)\{b(X)dt + \sigma(X)dW(t)\} + \frac{1}{2}f''(X)\sigma^2(X)dt.$$

Taking expectations yields (use property 3 of Brownian motion)

$$\frac{\partial \mathbb{E}[f(X)]}{\partial t} = \mathbb{E}[f'(X)b(X)] + \frac{1}{2}\mathbb{E}[f''(X)\sigma^2(X)]. \tag{2.14}$$

Let

$$\mathbb{P}\left(X(t) = x\right) \tag{2.15}$$

be the probability that the system (2.13) is in state x at a given time t. Then the probability

density corresponding to (2.15), denoted by $p(t,x)$, can be used to rewrite (2.14) as

$$\begin{aligned}\int \left\{ f(x)\frac{\partial p(t,x)}{\partial t} \right\} dx &= \int \left\{ f'(x)b(x) + \frac{1}{2}f''(x)\sigma^2(x) \right\} p(t,x)dx \\ &= \int f(x) \left\{ -\frac{\partial}{\partial x}(b(x)p(t,x)) + \frac{1}{2}\frac{\partial^2}{\partial x^2}(\sigma^2(x)p(t,x)) \right\} dx\end{aligned}$$

where we have used partial integration. Since this result is valid for arbitrary function f, we find that $p(t,x)$ must satisfy

$$\frac{\partial p(t,x)}{\partial t} = -\frac{\partial}{\partial x}(b(x)p(t,x)) + \frac{1}{2}\frac{\partial^2}{\partial x^2}(\sigma^2(x)p(t,x)).$$

This is the one dimensional form of the *Fokker-Planck (FP) equation* (also known as the *Kolmogorov forward equation*). Observe that this is a deterministic partial differential equation for a probabilistic quantity! The boundary conditions to this differential equation depend on the domain where (2.13) is defined and on its practical interpretation. The general form of the FP-equation is

$$\frac{\partial p(t,x)}{\partial t} = -\sum_{i=1}^{n} \frac{\partial}{\partial x_i}(b_i(x)p(t,x)) + \frac{1}{2}\sum_{i,j=1}^{n} \frac{\partial^2}{\partial x_i \partial x_j}(a_{ij}(x)p(t,x))$$

where the a_{ij} are the elements of the matrix $A = \sigma\sigma^T$. The coefficient $b_i(x)$ is usually called the *drift* and $a_{ij}(x)$ or $\sigma_{ij}(x)$ the *diffusion* coefficient. As $t \to \infty$, the probability density $p(t,x)$ tends to the stationary distribution $p^{(s)}(x)$ which satisfies

$$Mp^{(s)}(x) = 0 \tag{2.16}$$

with

$$M = -\sum_{i=1}^{n} \frac{\partial}{\partial x_i}(b_i(x)\cdot) + \frac{1}{2}\sum_{i,j=1}^{n} \frac{\partial^2}{\partial x_i \partial x_j}(a_{ij}(x)\cdot). \tag{2.17}$$

(This is actually only true for time homogeneous systems, that is why we choose (2.12) time homogeneous.) The formal adjoint of the operator M given by (2.17) is the so-called backward operator $L = M^*$ or

$$L = \sum_{i=1}^{n} b_i(x)\frac{\partial}{\partial x_i} + \frac{1}{2}\sum_{i,j=1}^{n} a_{ij}(x)\frac{\partial^2}{\partial x_i \partial x_j} \quad .$$

As will be clear from the following, this operator is often used in the 'differential equation' approach to the exit problem.

The exit problem

Before we can find a differential equation for the expected exit time, we will need to take a closer look at the probability that the solution to the SDE (2.12) crosses the boundary (this is mathematically known as *the exit probability*). We suppose that initial condition for (2.12) is $X^{(0)} = x$ and that the operator L is elliptic, i.e. the matrix

$$A = (a_{ij})$$

is positive definite.

Let $u(x)$ be the solution to the problem

$$\begin{aligned} Lu &= 0 \quad \text{in} \quad \Omega \\ u &= f(x) \quad \text{at} \quad \partial\Omega \end{aligned} \tag{2.18}$$

and the auxiliary function $G(x, y)$ the solution of

$$\begin{aligned} M_y G &= \delta(x - y) \qquad x, y \in \Omega \\ G(x, y) &= 0 \qquad\qquad x \in \Omega, y \in \partial\Omega. \end{aligned}$$

where we have written M_y for the operator M (see 2.17) to emphasize that differentiation is w.r.t. y and not x. Evaluating the integral

$$I = \int_\Omega u(y) M_y G(x, y) - G(x, y) L u(y) dy$$

we see that on the one hand

$$I = u(x) \tag{2.19}$$

since $M_y G = \delta(x - y)$ and $Lu = 0$ in Ω. On the other hand, applying *Green's Formula* (see [61, p. 386]) yields

$$I = \int_{\partial\Omega} u(y) \frac{\partial G(x, y)}{\partial \nu} - G(x, y) \frac{\partial u(y)}{\partial \nu} dy. \tag{2.20}$$

Here ν is the outward normal to the boundary $\partial\Omega$. Putting (2.19) and (2.20) together and using the boundary conditions for G and u, we find

$$u(x) = \int_{\partial\Omega} f(y)\frac{\partial G(x,y)}{\partial \nu}dy.$$

Thus, using the auxiliary function $G(x,y)$, we have written the solution to (2.18) in terms of its boundary condition. The function G(x,y) is better known as a *Green's function* and is often used to solve differential equations, see e.g. [62] or [63].

In [7, §5.4], it is found that $\frac{\partial G(x,y)}{\partial \nu}$ is a function denoting the distribution at $\partial\Omega$ for the probability of arriving at $y \in \partial\Omega$ if starting in $x \in \Omega$. Now suppose that the boundary $\partial\Omega$ consists of two parts $\partial\Omega_0$ and $\partial\Omega_1$. If we choose f to be

$$f(x) = \begin{cases} 0 & \text{if} \quad x \in \partial\Omega_0 \\ 1 & \text{if} \quad x \in \partial\Omega_1 \end{cases}$$

then $u(x)$ can be viewed as the probability of leaving Ω through the part of the boundary $\partial\Omega_1$ if starting in $x \in \Omega$ (i.e. $u(x)$ is the exit probability).

We will need $u(x)$ to find the differential equation for the expected exit time. Define the function $q(t,x)$ as the probability of leaving Ω through $\partial\Omega_1$ after a time t. It satisfies (see [15, §5.4.2])

$$\frac{\partial q}{\partial t} = Lq \quad \text{in } \Omega \quad \text{for} \quad t > 0$$
$$q(0,x) = u(x) \quad \text{in } \Omega \quad \text{and} \quad q(t,x) = 0 \quad \text{at} \quad \partial\Omega.$$

Define $T(x)$ as

$$T(x) = \int_0^\infty q(t,x)dt.$$

Because the probability distribution of the stochastic exit time is given by

$$-\frac{1}{q(0,x)}\frac{\partial q}{\partial t}(t,x)$$

we have for the expected exit time (by partial integration)

$$T_1(x) = \int_0^\infty -\frac{s}{q(0,x)}\frac{\partial q}{\partial s}(s,x)ds = \int_0^\infty \frac{q(s,x)}{q(0,x)}ds = \frac{T(x)}{u(x)}. \tag{2.21}$$

From expression (2.21), stating that $T(x) = T_1(x)u(x)$, it follows that $T(x) = 0$ at $\partial\Omega$ because $u(x) = 0$ at $\partial\Omega_0$ and $T_1(x) = 0$ at $\partial\Omega_1$. Also

$$LT(x) = \int_0^\infty Lq(s,x)ds = \int_0^\infty \frac{\partial q}{\partial s}(s,x)ds = -q(0,x) = -u(x),$$

and thus the quantity $T(x)$ is completely determined by the boundary value problem

$$LT(x) = -u(x) \text{ in } \Omega, \qquad T(x) = 0 \text{ at } \partial\Omega. \tag{2.22}$$

Thus if we seek to find the expected exit time from the domain Ω through the part $\partial\Omega_1$ of its boundary, then we can proceed by solving $T(x)$ from (2.22) and then extract $T_1(x)$ using (2.21).

Remark: In the above, we have taken Ω_0 to be absorbing (i.e. $u(x) = 0$). We can also choose the boundary to be reflecting, i.e.

$$Lu(x) = 0 \quad \text{in } \Omega$$

$$\sum_{i,j=1}^{n} \nu_i a_{ij}(x) \frac{\partial u}{\partial x} = 0 \quad \text{at } \partial\Omega_0 \quad \text{and} \quad u = 1 \text{ at } \quad \partial\Omega_1,$$

admitting the simple solution $u(x) \equiv 1$. Then $T(x)$ satisfies

$$LT(x) = -1 \quad \text{in } \Omega$$

$$\sum_{i,j=1}^{n} \nu_i a_{ij}(x) \frac{\partial T}{\partial x} = 0 \quad \text{at } \partial\Omega_0 \quad \text{and} \quad T = 0 \text{ at } \quad \partial\Omega_1.$$

The vector ν is the outward normal to the boundary. For more details, see [15, p. 129].

Remark: If the boundary $\partial\Omega = \partial\Omega_1$, then the solution to the boundary value problem

$$Lu(x) = 0 \text{ in } \Omega, \qquad u(x) = 1 \text{ at } \partial\Omega$$

is given by $u(x) \equiv 1$ and thus exit takes place with probability 1. Then also $T_1(x) = T(x)$, denoting the expected exit time through any point of $\partial\Omega$.

An example

Consider the geometric Brownian motion from Section 2.2 for which the SDE is given by

$$dX(t) = \lambda X(t)dt + \mu X(t)dW(t), \qquad X(0) = x. \tag{2.23}$$

Let $X(t) \in \Omega = [1, 9]$ and set $\lambda = 5, \mu = 2$. The exit probability $u(x)$ satisfies $Lu = 0$ which takes the form

$$\lambda x \frac{\partial u}{\partial x} + \frac{1}{2}\mu^2 x^2 \frac{\partial^2 u}{\partial x^2} = 0. \tag{2.24a}$$

We want to know when $X(t)$ is expected to cross the right side of the interval (i.e. $x = 9$) and thus we impose boundary conditions

$$u(1) = 0, \qquad u(9) = 1. \tag{2.24b}$$

The solution to (2.24) is given by

$$u(x) = \frac{27}{26}(1 - x^{-3/2}). \tag{2.25}$$

The differential equation for the quantity $T(x)$ ($LT = -u$) now becomes

$$\lambda x \frac{\partial T}{\partial x} + \frac{1}{2}\mu^2 x^2 \frac{\partial^2 T}{\partial x^2} = -u$$

with boundary conditions

$$T(1) = 0, \qquad T(9) = 0.$$

Solving this yields

$$T(x) = -\frac{9}{26}\ln x - x^{-3/2}\left(\frac{9}{26}\ln x + \frac{126}{169}\ln 3\right) + \frac{126}{169}\ln 3.$$

The expected exit time is now obtained via the relation

$$T_1(x) = \frac{T(x)}{u(x)}. \tag{2.26}$$

The exit probability (2.25) and the expected exit time (2.26) are shown in Figure 2.2.

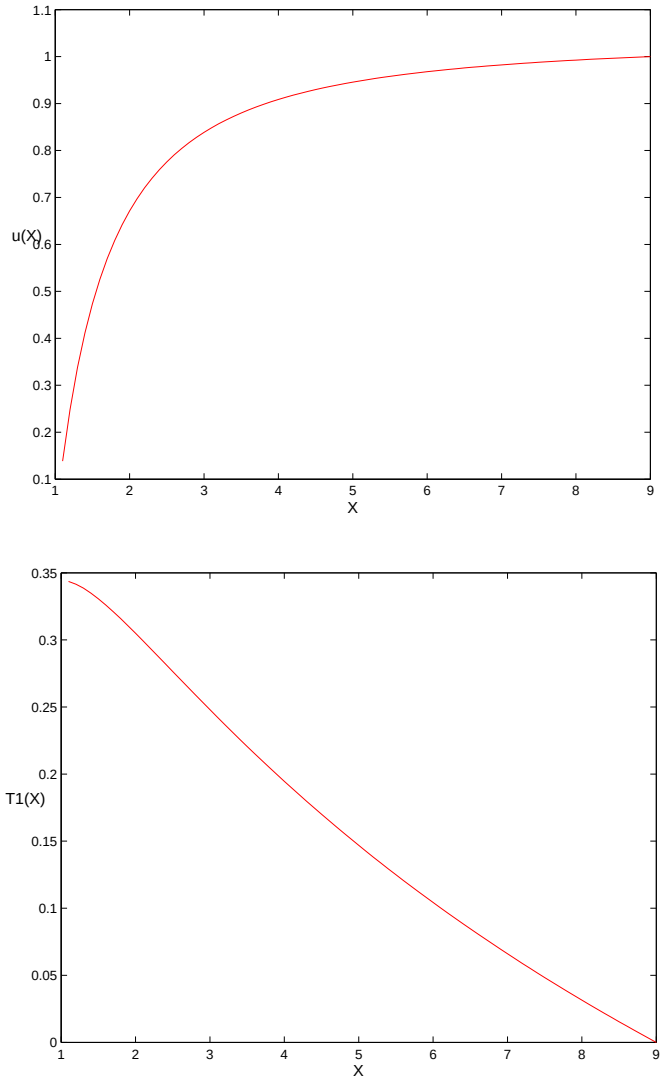

Figure 2.2: The exit probability (top) and the expected exit time (bottom).

Chapter 3

Simulation of Stochastic Differential Equations

In the previous chapter, we saw how we can obtain the exit probability and the expected exit time from a differential equation. In this chapter, we will show how the same information can be obtained via a simulation of the SDE. We will need to be able to simulate Brownian motion and thus we start with that. Then we look at stochastic differential equations, after which we will look at the exit probability and the expected exit time. All simulations are done with the software package 'MATLAB'.

3.1 Simulation of Brownian motion

To get started, we consider a discretized version of Brownian motion. We therefore choose a stepsize Δt and let W_j denote $W(t_j)$ with $t_j = j\Delta t$. According to the first property of Brownian motion (see Section 2.1) we have $W(0) = 0$ and from the second and third property we find

$$W_j = W_{j-1} + dW_j \qquad j = 1, 2, \ldots, N \; . \tag{3.1}$$

Here N denotes the number of steps that we take and dW_j is a normally distributed random variable with zero mean and variance Δt.

Observe here the relation between the quantity S_j from the introduction and W_j. In the final version of our 'game', we found that S_j is a random variable with expectation 0 and variance $\frac{jt}{n}$ (see (1.4)). Thus we have that

$$\mathbb{E}\left(S_j - S_{j-1}\right) = 0 \quad , \quad \mathbb{E}\left(\left(S_j - S_{j-1}\right)^2\right) = \frac{t}{n}$$

i.e. $S_j - S_{j-1}$ is a random variable with expectation 0 and variance $\frac{t}{n}$. If we take $\Delta t = \frac{t}{n}$, then the same goes for $W_j - W_{j-1}$. The difference that remains is that the Brownian motion is not just any random variable, but a normally distributed random variable. The proof that $S_j - S_{j-1}$ approaches $W_j - W_{j-1}$ as $n \to \infty$ is actually an application of the well known Central Limit Theorem and can be found in [6, p. 13].

We return now to the discretized version of Brownian motion W_j. Expression (3.1) can be seen as a numerical recipe to simulate Brownian motion. An implementation of this recipe can be found in Appendix A.1. We start the simulation by setting the state of MATLAB's random generator to 10000. This makes sure that the simulations can be repeated by interested readers. Next, we define our stepsize `dt` as $1/500$ and set `N=500`. We will be using arrays `W` and `dW`, which we preallocate with zeros for efficiency. `W` will hold the simulated path of Brownian motion and `dW` will hold the normally distributed increments. The random numbers generated by `randn` are drawn from a normal distribution with zero mean and unit variance (i.e. from $N(0,1)$) and thus have to be multiplied by $\sqrt{\Delta t}$. The `for`-loop performs the actual simulation. Note that MATLAB starts arrays at index 1 and thus we have to do the first approximation outside the loop. The array `W` which is thus created is called a *discretized Brownian path* (or a *sample path*).

Remember that Brownian motion is a random process and thus each sample path should look different. To illustrate this, we simulate 2 more sample paths. The results of these simulations are plotted in Figure 3.1. It clearly shows the randomness of Brownian motion : the blue path doesn't seem to increase much, contrary to the yellow path. The red path started by increasing, then dramatically decreases and finally starts to increase again. Of course, this is just what we see in the figure. If we would have continued the simulation to e.g. $t = 10$ (instead of $t = 1$), the situation might be completely different.

Efficiency considerations

From the viewpoint of efficiency, `for`-loops should be avoided as much as possible. This can be achieved by using so-called 'vectorized' commands. Our `for`-loop can be vectorized in the following manner. In stead of calling `randn` with arguments `(1,1)`, we now call it

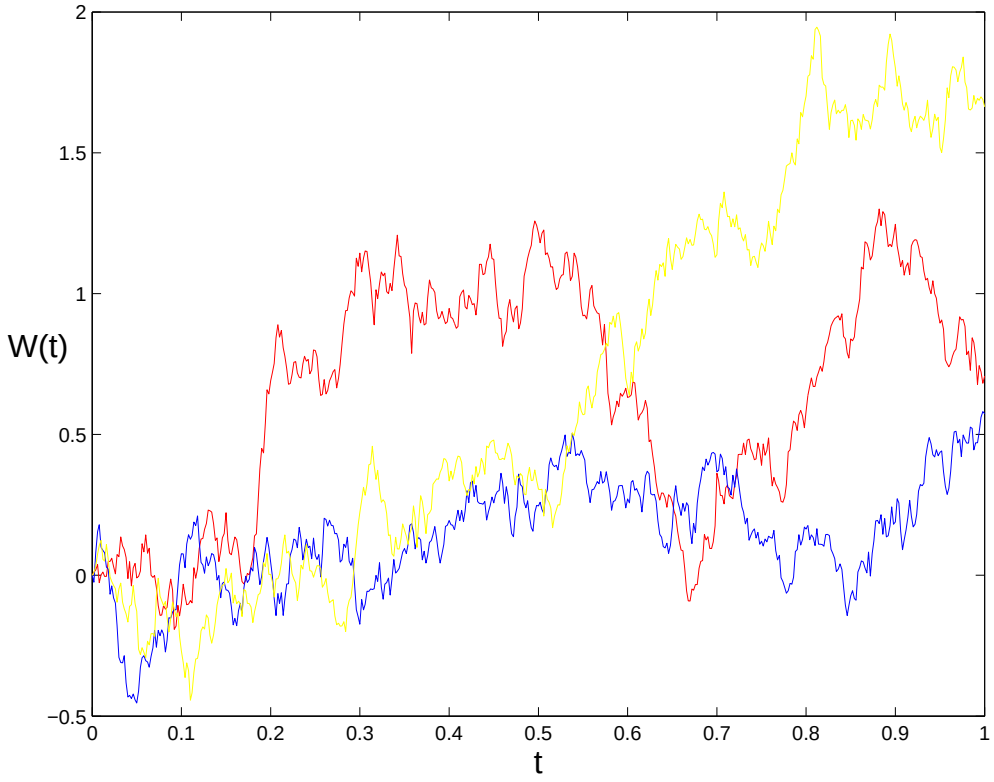

Figure 3.1: Three sample paths of Brownian motion

with arguments `(1,N)`. This causes `randn` to generate an array of random numbers drawn from $N(0,1)$. We again store these values in `dW`. We now use the command `cumsum` to calculate the cumulative sum of `dW`. Thus `W(j)` becomes $dW(1) + dW(2) + \ldots + dW(j)$ (which is the same as (3.1)). The `for`-loop can now be replaced by

```
dW=randn(1,N);
W=cumsum(dW);
```

This still leaves us with the same two lines written down 3 times (because we had 3 `for`-loops). Therefore, we vectorize the commands one level further. We call `randn` with arguments `(3,N)`, which makes `dW` a $3 \times N$ matrix of normally distributed random numbers. Each row of `dW` now contains the increments for one of the three sample paths. We now again apply the command `cumsum` to generate the Brownian paths. Since `dW` is a matrix, we have to tell `cumsum` to calculate cumulative sums over the column (i.e. second) dimension. Listing BrownV.m (A.2) shows the resulting code. Output of BrownV.m is analogous to the output of BrownU.m .

The difference in computing time is only slightly noticeable with this example, but as pro-

N	BrownU.m	BrownV.m	gain
500	0.8023s	0.2267s	354%
5000	1.8762s	0.3952s	475%
50000	14.7858s	0.6975s	2120%
500000	147.9821s	4.7308s	3128%

Table 3.1: Computation times (in seconds) for BrownU.m and BrownV.m for different values of N

grams become more difficult the vectorized version can become several orders of magnitude faster than the unvectorized version. This is illustrated in Table 3.1, where we have listed computation times for both versions against different values for N. The data in the table clearly shows the improvement. Another advantage of vectorized commands is that besides faster, they are also shorter and thus save space in code-listings.

Many MATLAB commands have a vectorized version. Besides the commands `randn` and `cumsum`, we will also use the vectorized version of the multiplication (`*`) and power (`^`). In vectorized notation, they become `.*` and `.^` respectively. There left and right arguments are arrays or matrices on which they perform their normal operation element-wise. For instance

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} .* \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 5 & 12 \\ 21 & 32 \end{pmatrix}$$

and

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} .\hat{} \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix} = \begin{pmatrix} 1 & 64 \\ 2187 & 65536 \end{pmatrix}.$$

In the rest of this document, we will use vectorized commands wherever it is possible.

Functions along Brownian paths

Now that we know how to simulate Brownian motion, it is fairly easy to simulate a function along a Brownian path. Consider the function

$$u(t) = e^{3t+2W(t)}, \tag{3.2}$$

which is a special case of the Geometric Brownian motion (see expression (2.11)) with $\lambda = 5$ and $\mu = 2$). The code in the file BrownF.m (A.3) evaluates the function $u(t)$ along a path of Brownian motion. The Brownian path is created in the same way as in the previous

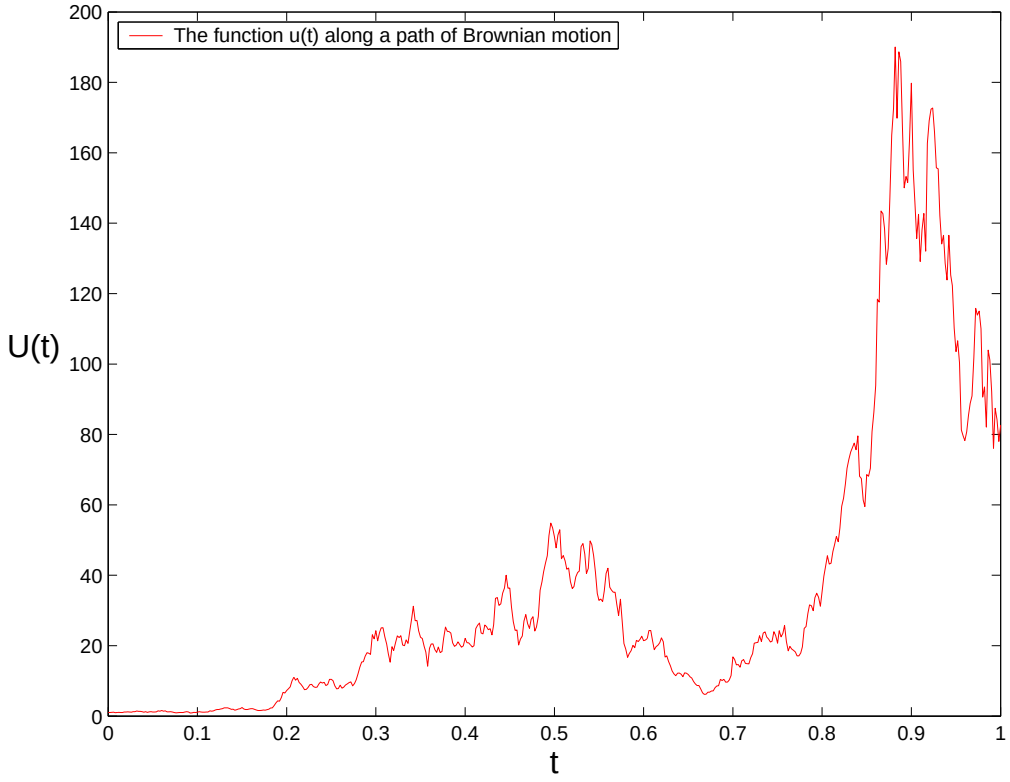

Figure 3.2: The function $u(t)$ as in (3.2) along a sample path of Brownian motion.

section, with the use of vectorized commands. The expression

```
U=exp(3*t + 2*W)
```

then fills the array `U` with the simulated values of $u(t)$ from (3.2). The result is plotted in Figure 3.2.

3.2 Simulation of SDE's

Now that we have some experience with simulating Brownian motion, we can get started with stochastic differential equations. SDE's are usually approximated using the *Euler-Maruyama method*. Consider the SDE

$$dX = b(X)dt + \sigma(X)dW(t), \quad X(0) = X_0. \tag{3.3}$$

Then the discretization of (3.3)

$$X_j = X_{j-1} + b(X_{j-1})\Delta t + \sigma(X_{j-1})dW_j, \qquad j = 1, \ldots, N, \tag{3.4}$$

is called the Euler-Maruyama (EM) approximation to (3.3). Here X_j is the approximation to $X(j\Delta t)$ and, as in the previous section, Δt is the stepsize, $dW_j = W_j - W_{j-1}$ and N is the number of steps we will take. Observe that if $\sigma(X) \equiv 0$, then (3.4) reduces to the deterministic Euler method. Together with MATLAB, (3.4) enables us to simulate sample paths of the solution to (3.3).

We have actually already seen a simulation of a SDE. The expression (3.1) from Section 3.1 (with W_j and W_{j-1} replaced by X_j and X_{j-1})

$$X_j = X_{j-1} + dW_j$$

is the EM approximation to the SDE

$$dX(t) = dW(t).$$

This SDE admits the simple solution $X(t) = W(t)$ and thus Section 3.1 shows how to simulate Brownian motion with the EM method.

As a second example, lets consider the geometric Brownian motion process (2.10) from the previous chapter. Its EM approximation is given by

$$X_j = X_{j-1} + \lambda X_{j-1}\Delta t + \mu X_{j-1} dW_j \qquad j = 1, \ldots, N.$$

Using parameter values $\lambda = 5$, $\mu = 2$, this reduces to

$$X_j = X_{j-1} + 5X_{j-1}\Delta t + 2X_{j-1} dW_j \qquad j = 1, \ldots, N \tag{3.5}$$

with true solution (see (2.11))

$$X(t) = e^{3t+2W(t)}. \tag{3.6}$$

In the file GBMEM.m (A.4) we perform one simulation using (3.5) and compare the result with the true solution (3.6). We take $N = 500$ steps with a stepsize of $\Delta t = 1/500$. The Brownian path and the true solution are computed first, in the same way as in the previous section. The true solution is then stored in `Xtrue`. The `for`-loop then fills the array `Xem` with the EM-approximation. Unfortunately, we can not replace the for loop with vectorized commands. The result is shown in Figure 3.3. `Xtrue` is plotted as a blue line and `Xem` as

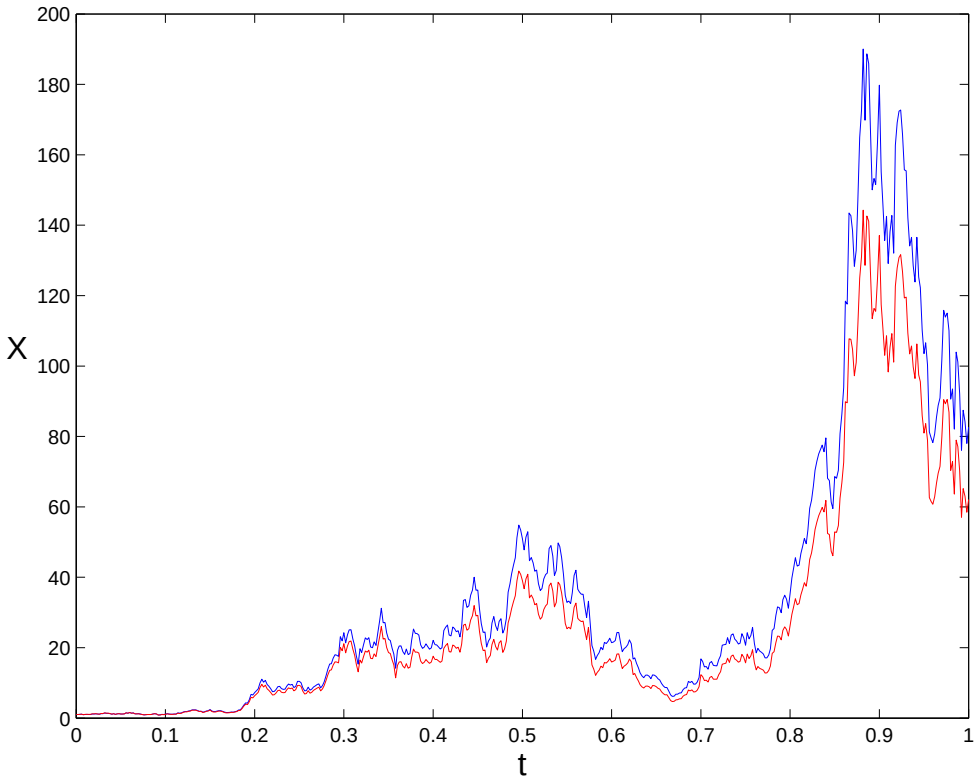

Figure 3.3: The true solution (3.6) (blue/upper line) and the EM-approximation (3.5) (red/lower line).

a red line. We see that for small values of t, the approximation matches the true solution nicely. As t gets larger, the difference between the two grows larger. This makes sense from a numerical point of view. From (3.4), it is clear that X_j depends on the values of $\{X_l | 0 \leq l < j\}$. A numerical error in one of the X_l's is thus passed on to X_j. For small j, X_j will not suffer much from this, since it is based on only a few X_l's. But as j gets larger, X_j depends on more X_l's and thus the numerical error in X_j will become larger. The next section will take a closer look at these errors.

3.3 Error in the Euler-Maruyama method

3.3.1 Strong convergence

We would like to say something about the error in the EM-approximation. As before, let $X(t)$ be the solution to (3.3) and X_j the EM-approximation to $X(t_j)$. The error is thus

$$error_{EM} = |X(t_j) - X_j| .$$

Remembering that $X(t_j)$ and X_j are random variables, we see that the error is also a random variable. We use the expectation of this random variable to arrive at the notion of strong convergence. A method is said to have *strong order of convergence* equal to γ, if there exists a constant C (independent of Δt) such that

$$\mathbb{E}\left(|X(t_j) - X_j|\right) \leq C\Delta t^{\gamma} \tag{3.7}$$

for any fixed t_j ($0 \leq j \leq N$).

In the deterministic setting (i.e. when $\sigma \equiv 0$), the expectation on the left-hand side of (3.7) falls out and we are left with the deterministic definition of convergence. It is known that the deterministic Euler method is convergent of order $\gamma = 1$. In the stochastic world, things are different. From e.g. [42, Thm. 10.2.2], we know that the EM method has strong order of convergence $\gamma = \frac{1}{2}$. Thus if we want to decrease the expected value of the error, we can simply make the stepsize smaller.

Off course we would like to inspect this numerically, but this presents us with a new problem. Since we have no expression for the distribution of $error_{EM}$, we have no way of determining $\mathbb{E}(error_{EM})$. We handle this problem by applying a method called the *Monte Carlo* (MC) method.

3.3.2 The Monte Carlo method

Consider the quantity

$$\frac{1}{M}\sum_{r=1}^{M} Y(\omega_r) \tag{3.8}$$

where $Y(\omega_r)$ is a sample path of the random variable Y. The Central Limit Theorem states that (3.8) is approximately normally distributed with parameters μ and σ^2/M, where $\mu = \mathbb{E}(Y)$ and $\sigma^2 = VAR(Y)$. Thus we see that

$$\frac{1}{M}\sum_{r=1}^{M} Y(\omega_r) \to \mu = \mathbb{E}(Y) \qquad \text{as } M \to \infty.$$

The quantity (3.8) can thus be used to approximate $\mathbb{E}(Y)$. This method for approximating expected values is known as the Monte Carlo method. In simpler terms,(3.8) computes M sample paths of Y and returns the sample average as an approximation to the expected value of Y. It is known (see [66, Thm. 12.4]) that for the approximation (3.8) we have

$$\left| \frac{1}{M} \sum_{r=1}^{M} Y(\omega_r) - \mathbb{E}(Y) \right| = o(\frac{1}{\sqrt{M}})$$

and thus the MC-method converges with order $\frac{1}{\sqrt{M}}$. More information about the Monte Carlo method can be found in [64, 65].

3.3.3 Evaluation of the error

We will use the MC-method to evaluate the error (3.7). In computer simulations, it is not possible to do infinitely many simulations. The best we can do is to take M 'large'. The approximation to the expectation we will then obtain is thus influence by a so-called *statistical error* (besides the numerical error which was already present in (3.7)). For now, we will suppose that $M = 10000$ is large enough to make the statistical error invisible. We say a little bit more about the statistical error in the next section.

We are now finally in a position to numerically investigate the claim that the expected value of the numerical error decreases as the stepsize is decreased. We will perform a Monte Carlo simulation of the EM approximation (3.5) to the geometric Brownian motion (2.10), using $M = 10000$ sample paths and different values for Δt. We will look at the error at time $t = 0.5$. The code can be found in Appendix A.5. We start by computing `M` sample paths of Brownian Motion using a small stepsize ($\Delta t = 1/1600$). These sample paths are used to fill the MATLAB variable `Xtrue` with the true solution at $t = 0.5$. We then use a `for`-loop to iterate over the stepsizes $R \cdot \Delta t$, for $R = 16, 8, 4, 2$ and 1. These stepsizes are then used in the next `for`-loop to generate `M` EM-appxoximations. Observe that the Brownian increments $W(jR\Delta t) - W((j-1)R\Delta t)$ are obtained from the relation

$$W(jR\Delta t) - W((j-1)R\Delta t) = \sum_{k=jR-R+1}^{jR} dW_k$$

where dW is the array that we previously filled with increments for the smaller stepsize $\Delta t = 1/1600$.

Then, the command

```
ErrorEM(i+1)=mean(abs(Xtrue-Xtemp))
```

is used to compute the mean error, which is our approximation to the expected value of the error. Having done this, we can continue with the next stepsize. Table 3.2 shows the results.

Δt	$error_{EM}$
1/100	2.0857
1/200	1.4839
1/400	1.0139
1/800	0.7144
1/1600	0.4933

Table 3.2: Error in the EM approximation to (3.5) to the geometric Brownian motion at time $t = 0.5$, for different values of Δt.

The mean of the error clearly decreases as the stepsize is made smaller, supporting the claim made in expression (3.7).

Remark: Using the data from Table 3.2, we can numerically confirm the strong order of convergence of the EM-method. If (3.7) holds with equality, then, taking logs,

$$\log\left(\mathbb{E}\left(|X(t_j) - X_j|\right)\right) = \log C + \gamma \Delta t.$$

We can now make a least squares fit for C and γ using the MATLAB commands

```
dt=[1/100,1/200,1/400,1/800,1/1600];
err=[2.0857,1.4839,1.0139,0.7144,0.4933];
A=[ones(5,1),log(dt)'];
rhs=log(err)';
sol=A\rhs;
gamma=sol(2)
res=norm(A*sol-rhs)
```

which gives gamma≈ 0.5215 with least squares residual of 0.0177. These results are consistent with the strong order of convergence of order $\frac{1}{2}$ for the EM-method that we claimed before.

3.4 The expected exit time

Having done all the preliminary work, we can now focus on numerically finding the expected exit time. As in Section 2.3, we suppose that we are dealing with a system in the form of a SDE. The SDE is defined on a domain Ω with boundary $\partial\Omega = \partial\Omega_0 \cup \partial\Omega_1$. We want to know the time at which a sample path crosses the boundary $\partial\Omega_1$.

The expected exit time can be approximated in the following manner. We start by simu-

lating a sample path of the SDE by the means which we developed in Section 3.2. While simulating this sample path, we observe whether it crosses the boundary. This gives us an approximation to the exit time of the current sample path. We use a Monte Carlo approach to finding the expected exit time of the system. Thus we simulate the exit time of M sample paths and approximate the expected exit time by the sample average.

Of course, not all sample paths necessarily cross the boundary. A sample path can also cross the boundary $\partial\Omega_0$. If this boundary is absorbing then the sample path can no longer exit through $\partial\Omega_1$. We can use this to approximate the exit probability. If we remember the number of sample paths that don't exit (say Q), then we can approximate the exit probability by $\frac{M}{Q+M}$. The example at the end of this section illustrates how we can find the expected exit time and the exit probability.

Accuracy

Before we start the example, we need to say something more about the error introduced by this way of simulating the expected exit time. This error consists of two parts :

- Discretization error : this is a consequence of using the EM-method to simulate the system;
- Statistical error : this is due to the use of the Monte Carlo method to approximate the expected value of the exit time.

There is actually a third source of errors : a *systematical error* due to the representation of decimal numbers in a computer. In this document we will suppose that systematical errors are negligible compared to the other two error sources.

We already saw in the previous section that reducing the stepsize makes the EM-method more accurate and thus the discretization error in our approximation to the expected exit time can be treated by decreasing the stepsize. Also, the statistical error can be made smaller by increasing the number of Monte Carlo simulations M. Although these ways our usefull to reduce the error in our results, they say nothing about the accuracy of the simulation results. Since we can't always compare the simulation results to the exact outcome, we would like to say something about accuracy.

We will deal here with a way to estimate the statistical error. The idea is to do a number of (L) small Monte Carlo simulations (of size M) of the expected exit time and use this data to construct a 100(1-α)% confidence interval ($\alpha \in [0, 1]$). This interval is given by

$$(\hat{\epsilon} - \Delta\hat{\epsilon}, \hat{\epsilon} + \Delta\hat{\epsilon})$$

where $\hat{\epsilon}$ is the average of the L MC-simulations and

$$\Delta\hat{\epsilon} = t_{1-\alpha,L-1}\sqrt{\frac{\sigma^2}{L}}.$$

Here, σ^2 is the variance of the L MC-simulations and $t_{1-\alpha,L-1}$ is obtained from the Student t-distribution with $L-1$ degrees of freedom (see e.g. [2, Sec. 6.9] or [4, Sect. 7.4].
Our approximation to the expected exit time, with the statistical error removed from it, is with 100(1-α)% certainty within this interval. Thus the size of the confidence interval gives us some information about the magnitude of the statistical error. In the example below, and in the rest of this document, we will use 100(1-α)% confidence intervals with $\alpha = 0.10$.

An example

We want to verify the results of the example from the end of Section 2.3. Therefore, we again use the geometric Brownian motion (see expression (2.23)) with parameters $\lambda = 5, \mu = 2$ and take the domain Ω to be the interval $[1, 9]$. We want to know when the system is expected to cross the right side of the interval. If it crosses the left side of the interval, we are not interested in it anymore. The starting value X_0 is varied from 2 to 8.
The code can be found in Appendix A.6. We did `L=20` Monte Carlo simulations with `M=50` sample paths to compute the average exit time and a stepsize of `dt=0.00025`. The listing contains four `for`-loops. The first loop is for the different startingpoints, the second is for the `L` MC simulations and the third for the `M` sample paths of one of the MC simulations. The final loop simulates a single sample path. This loop contains two `if`-statements, which monitor whether the sample path has crossed either of the two boundaries. At the end of each MC simulation, we use the commands

```
Tsim(l+1)=mean(total);
Usim(l+1)=sims/(sims+Q);
```

to compute the mean exit time and the exit probability. Having done this for each MC-simulation, we use

```
avg=mean(Usim);
```

```
delta=1.73*sqrt(var(Usim)/L);
CIU(:,Xzero-1)=[avg-delta,avg+delta]';
```

to create a 90% confidence interval for the exit probability for the current starting point (for `L=20` and $\alpha = 0.10$ we have $t_{1-\alpha,L-1} \approx 1.73$). A 90% confidence interval for the expected exit time is computed in the same way. When the simulation is ready, the matrices `CIU` and `CIT` contain the confidence intervals for all the different starting points. The first row contains the lower boundary of the confidence interval, the second row the upper boundary.

Table 3.3 shows the results (I is the confidence interval), together with the true solutions obtained from (2.25) and (2.26). The same information is shown in Figures 3.4 and 3.5. We see that the simulation results and the true solution match eachother nicely. Our way of simulating the expected exit time (and the exit probability) seems to work.

X_0	`Utrue`	I	`Ttrue`	I
2	0.6713	(0.6531 , 0.6994)	0.3050	(0.3065 , 0.3345)
3	0.8386	(0.8444 , 0.8759)	0.2480	(0.2466 , 0.2650)
4	0.9087	(0.9043 , 0.9357)	0.1946	(0.1928 , 0.2129)
5	0.9456	(0.9212 , 0.9467)	0.1469	(0.1391 , 0.1572)
6	0.9678	(0.9576 , 0.9737)	0.1043	(0.1016 , 0.1213)
7	0.9824	(0.9687 , 0.9854)	0.0661	(0.0658 , 0.0813)
8	0.9926	(0.9910 , 0.9993)	0.0315	(0.0380 , 0.0486)

Table 3.3: Confidence interval I and exact values for the exit probability and expected exit time of the geometric Brownian motion.

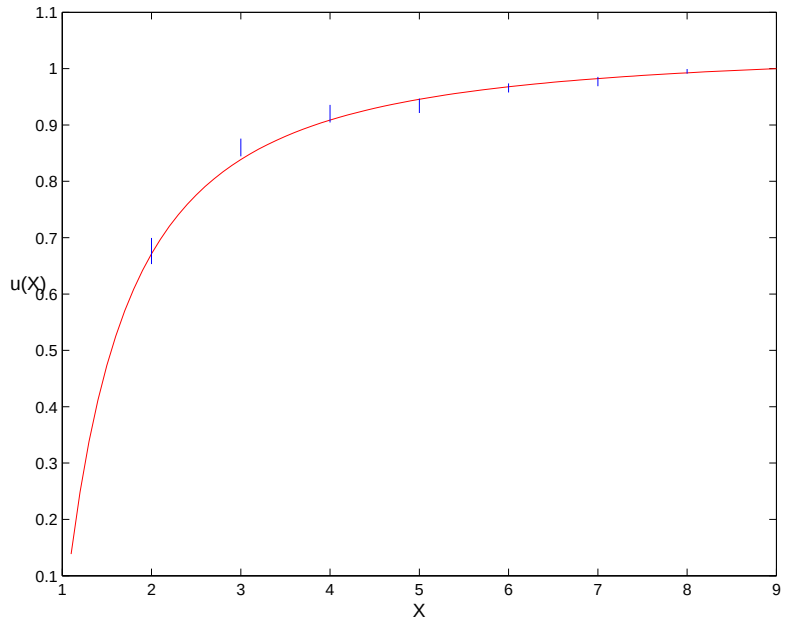

Figure 3.4: The exit probability for geometric Brownian motion obtained from (2.25) and the simulation results from Table 3.3.

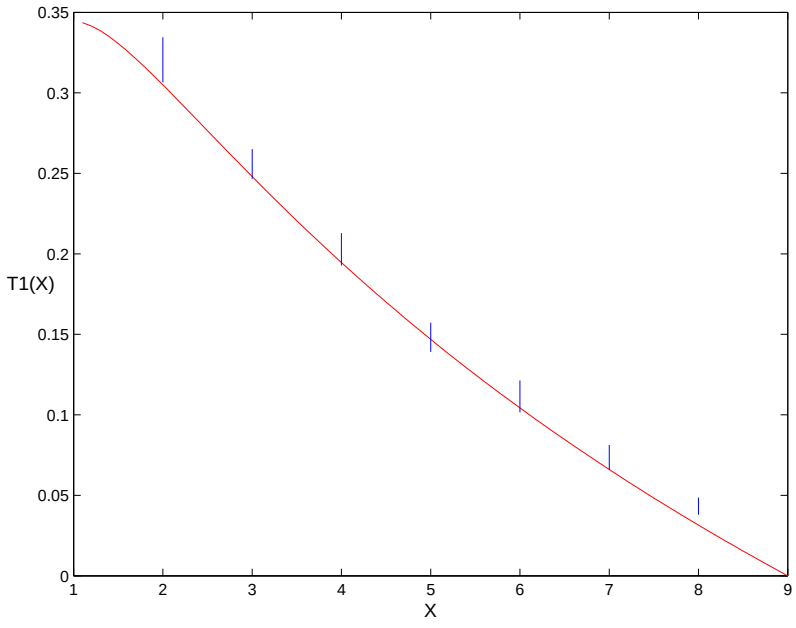

Figure 3.5: The expected exit time for geometric Brownian motion obtained from (2.26) and the simulation results from Table 3.3.

Chapter 4

Applications of Stochastic Differential Equations

In this chapter, we will be looking at applications of SDE in a number of different areas. We will consider SDE of the form

$$dX_i(t) = b_i(X(t))dt + \epsilon \sum_{j=1}^{k} \sigma_{ij}(X(t))dW_j(t), \quad X_i(0) = x_i, \quad i = 1, \ldots, n \qquad (4.1)$$

where $W_j(t)$ is again Brownian motion and $\epsilon > 0$ indicates the size of the noise. We will consider four models, all of which are given by a SDE. The models will come from different areas of research : biology, neurology, physics and hydrology.

We will be looking for the exit probability and the expected exit time of the system. We will find these quantities in two different ways : (1) from a differential equation, which we discussed in Section 2.3 and (2) from a simulation of the SDE as discussed in Section 3.4. We have already used both approaches on the geometric Brownian motion (2.10), see the examples at the end of Sections 2.3 and 3.4.

In stead of solving the differential equations exact, we will use a technique called *singular perturbations* to approximate their solutions. The singular perturbations method looks at what happens to the differential equations as the noise-parameter ϵ in (4.1) is decreased towards 0. The technique will be explained in the examples.

4.1 Stochastic Population Dynamics

An interesting application of SDE is in the field of stochastic population dynamics. Usually models are made for the evolution of the size of a population. The model we introduce here

was obtained from Grasman and Van Herwaarden [16, p. 55].

Suppose that the size of the population is influenced by birth, death and a carrier capacity of the environment. The carrier capacity is a parameter that models e.g. food problems : when a population is small in size there is food for everybody, but as it gets larger (towards the carrier capacity) there emerges a food shortage. The carrier capacity then has a negative effect on the growth of the population.

Suppose that the size of the population we consider is given by $N(t)$ and suppose that the birth rate b is such that

$$N \to N+1 \quad \text{with probability} \quad bN\Delta t$$

and that the death rate d is such that

$$N \to N-1 \quad \text{with probability} \quad (dN + \alpha K^{-1}N^2)\Delta t.$$

In this $\alpha = b - d > 0$ is the intrinsic growth rate and K is the carrier capacity. It is noted that $K \gg 1$. Furthermore, we suppose that the probability that two of the above events occur in the same time interval is of $O((\Delta t)^2)$ (and thus we ignore it). We now introduce the scaled variable

$$x(t) = \frac{N(t)}{K}.$$

Setting $\Delta x = x(t+\Delta t) - x(t)$ (the change in x over the time interval $(t, t+\Delta t)$), we find

$$\mathbb{E}[\Delta x] = \alpha(1-x)x\Delta t$$

and

$$\mathbb{E}\left[(\Delta x)^2\right] = K^{-1}(\beta x + \alpha x^2)\Delta t \quad \text{with} \quad \beta = b + d.$$

Ignoring terms of $O((\Delta t)^2)$, we obtain for the variance

$$Var[\Delta x] = K^{-1}(\beta x + \alpha x^2)\Delta t.$$

Approximating this process with a SDE gives

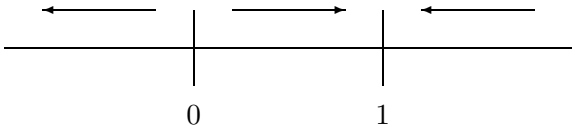

Figure 4.1: Phase plane of (4.3)

$$dX(t) = b(X(t))dt + \epsilon\sigma(X(t))dW(t), \quad X(0) = x, \quad X(t) \in [0, \infty), \tag{4.2}$$

with

$$\begin{aligned} b(x) &= \alpha(1-x)x, \\ \sigma(x) &= \sqrt{\beta x + \alpha x^2}, \\ \epsilon &= \sqrt{K^{-1}}. \end{aligned}$$

Note that $X(t)$ is now a continuous variable and that $\mathbb{E}[dX] = \mathbb{E}[\Delta x]$ and $\mathbb{E}\left[(dX)^2\right] = \mathbb{E}\left[(\Delta x))^2\right]$. The deterministic (noiseless) version of this SDE is

$$X'(t) = \alpha(1 - X(t))X(t), \quad X(0) = x, \tag{4.3}$$

which is known as the *logistic equation*. Rewriting this as

$$\frac{X'(t)}{(1 - X(t))X(t)} = \alpha, \quad X(0) = x$$

and using

$$\frac{1}{(1 - X(t))X(t)} = \frac{1}{X(t)} + \frac{1}{1 - X(t)}$$

we find that it has solution

$$X(t) = \frac{Ce^{\alpha t}}{1 + Ce^{\alpha t}}, \quad \text{with} \quad C = \frac{x}{1 - x},$$

for $x \neq 1$. The phaseplane of (4.3) is given by Figure 4.1. It illustrates that, in the absence of noise, $X(t) \to 1$ as $t \to \infty$ and thus the size of the population approaches the carrier capacity. Also, the drift is away from 0 and thus the population will not become extinct. For the model with noise, this is not true anymore. The noise term can influence the size of the population in both a positive and negative way. Mathematically, this means that $X(t)$ can now move against the drift. Thus the size of the population could become larger

than the carrier capacity and the population can even become extinct. This raises some interesting questions : what is the probability that the population becomes extinct and when is that expected to happen? We address these questions using the approach from Section 2.3.

We saw that the probability of extinction $u(x)$ (the 'exit probability') satisfies

$$Lu = 0 \tag{4.4a}$$

where L is the operator

$$L = b(x)\frac{\partial}{\partial x} + \frac{\epsilon^2}{2}a(x)\frac{\partial^2}{\partial x^2}$$

with $b(x)$ and $a(x) = \sigma(x)^2$ as in the SDE (4.2). Boundary conditions depend on the problem at hand. In our population model, we want the population to become extinct if $X(t) = 0$ and thus we set

$$u(0) = 1. \tag{4.4b}$$

We will allow the population to become infinitely large and still be able to become extinct and thus we set

$$u'(\infty) = 0 \tag{4.4c}$$

which is a reflecting boundary (see the remark on page 20). This completes the differential equation for $u(x)$. From the same section, we know that the expected extinction time ('the expected exit time') is given by

$$T_1(x) = \frac{T(x)}{u(x)},$$

where $T(x)$ satisfies the differential equation

$$LT = -u \tag{4.5a}$$

with boundary conditions

$$T(0) = 0, \quad T'(\infty) = 0. \tag{4.5b}$$

The solutions to these differential equations will give us the answer to the questions we asked. We could of course try to find exact solutions, but we choose to approximate these solutions using a method called *singular perturbations*.

Singular Perturbation Analysis

The singular perturbation (SP) method looks at what happens to the differential equations for u and T when the noise parameter ϵ vanishes. To do this, we will need to consider two different cases : (1) the starting value x is far away from the boundary $\{x = 0\}$ and (2) x is close to the boundary $\{x = 0\}$. This makes sense from a practical point of view. If the initial size of the population is large, we will not expect the small random changes due to the noise term to cause the population to become extinct quickly. It will probably take a long time. But if initially the population is already close to extinction, the noise term can have much influence even if it is very small. Mathematically this means that if x is large, we simply let $\epsilon \to 0$. If x is small, we introduce a local coordinate with which we zoom into the area around the boundary (this is usually called a *boundary layer*). Both will give a differential equation that we can solve. Of course these two solutions have to be combined into one global solution. We do this by imposing a condition that matches the two solutions together. Taking this approach, both for $u(x)$ and $T(x)$, we find the answer to our questions. The finer details can be found in [56, 57, 58].

In our population model, the situation is a bit easier. The differential equation for $u(x)$ (4.4) has the simple solution $u(x) \equiv 1$ (see the remark on page 20) and thus no SP-approximation is necessary. In terms of the model, this means that the population becomes extinct with probability 1 and thus we already have the answer to our first question. We now apply the SP-analysis to the differential equation for the expected exit time. The expected exit time $T_1(x)$ is now the same as the quantity $T(x)$ and thus satisfies

$$LT = -1, \quad T(0) = 0, \quad T'(\infty) = 0.$$

Because of the previous considerations, we suppose that $T(x)$ is large if x is far away from 0. We set $T(x) = C(\epsilon)\tau(x)$ with $C(\epsilon)$ getting larger as $\epsilon \to 0$. Then $\tau(x)$ satisfies

$$L\tau = -C(\epsilon)^{-1}, \quad \tau(0) = 0, \quad \tau'(\infty) = 0. \tag{4.6}$$

Following the SP analysis, we let $\epsilon \to 0$ and find

$$b(x)\frac{\partial \tau}{\partial x} = 0$$

or $\tau(x)$ =constant. Because $C(\epsilon)$ is still undetermined, we set $\tau(x) = 1$. This is often called the *outer solution*.

When we are closer to the boundary $x = 0$ (i.e. inside the boundary layer), the diffusion coefficient will play a larger role. T rapidly changes from $C(\epsilon)$ to 0 and thus $\tau(x)$ rapidly changes from 1 to 0. We zoom into this region by introducing the local coordinate

$$\xi = \frac{x}{\epsilon^\rho}.$$

Substitution into (4.6) yields

$$\alpha\xi\epsilon^\rho \cdot \epsilon^{-\rho}\frac{\partial \tau}{\partial \xi} + \frac{\epsilon^2}{2}\beta\xi\epsilon^\rho \cdot \epsilon^{-2\rho}\frac{\partial^2 \tau}{\partial \xi^2} = -C(\epsilon)^{-1}$$

where we've used that $b(x) \sim \alpha x$ and $a(x) \sim \beta x$ as $x \to 0$. We choose ρ such that both terms on the left are of equal order of magnitude in ϵ (i.e. $\rho = 2$). Letting $\epsilon \to 0$ and dividing by ξ gives us the differential equation

$$\alpha\frac{\partial \tau}{\partial \xi} + \frac{1}{2}\beta\frac{\partial^2 \tau}{\partial \xi^2} = 0. \tag{4.7}$$

The boundary condition at $\xi = x = 0$ is simply inherited from (4.5). The other boundary condition is chosen such that the solution inside the boundary layer (the *inner solution*) matches the outer solution as $\xi \to \infty$, i.e. $\tau(\xi) = 1$ as $\xi \to \infty$.

Dividing by $\beta/2$ and integrating (4.7) directly gives

$$\frac{2\alpha}{\beta}\tau + \frac{\partial \tau}{\partial \xi} = C_1. \tag{4.8}$$

Multiplying (4.8) with the integrating factor $e^{2\alpha\xi/\beta}$ transforms it into

$$\left(e^{\frac{2\alpha\xi}{\beta}}\tau\right)' = C_1 e^{\frac{2\alpha\xi}{\beta}}$$

which yields for $\tau(\xi)$

$$\tau(\xi) = C_1 + C_2 e^{-\frac{2\alpha\xi}{\beta}}.$$

Using the boundary conditions gives $C_1 = 1$, $C_2 = -1$. Transforming back to x gives us

$$\tau(x;\epsilon) = 1 - e^{\frac{-2\alpha x}{\beta\epsilon^2}}. \tag{4.9}$$

This leaves us with the task to find the constant $C(\epsilon)$. In the calculations we will be needing the stationary distribution $p^{(s)}(x)$ (the probability of being in state $x \in [0,\infty)$ at $t = \infty$). It satisfies (see (2.16) and (2.17))

$$-\frac{\partial}{\partial x}(b(x)p^{(s)}(x)) + \frac{\epsilon^2}{2}\frac{\partial^2}{\partial x^2}(a(x)p^{(s)}(x)) = 0.$$

Integrating from x to ∞ yields

$$b(x)p^{(s)}(x) - \frac{\epsilon^2}{2}\frac{\partial}{\partial x}(a(x)p^{(s)}(x)) = C_1$$

and using apriori information that $p^{(s)}(x)$ and $p^{(s)'}(x)$ are exponentially small at $x = \infty$, we find that $C_1 = 0$. Integrating from x to x^*, with x^* an arbitrarily chosen point in $[0,\infty)$, we obtain

$$p^{(s)}(x) = a(x^*)p(x^*)\frac{e^{-2\phi(x)/\epsilon^2}}{a(x)} \tag{4.10}$$

where

$$\phi(x) = \int_x^{x^*} \frac{b(s)}{a(s)} ds$$

and the value of $a(x^*)p(x^*)$ is determined from the constraint

$$\int_0^\infty p^{(s)}(x)dx = 1$$

We choose $x^* = \overline{x}$, where $\overline{x}$ is the equilibrium solution to $x'(t) = b(x)$ (i.e. $\overline{x} = 1$). So we have $\phi(\overline{x}) = 0$ and $\phi'(\overline{x}) = 0$. Also $\phi'(x) = -b(x)/a(x)$ is negative for $x < \overline{x}$ and positive for $x > \overline{x}$, see Figure 4.2.

Because $\phi(x)$ is smallest in $x = \overline{x}$, the function $p^{(s)}(x)$ will have a peak at $x = \overline{x}$. In fact,

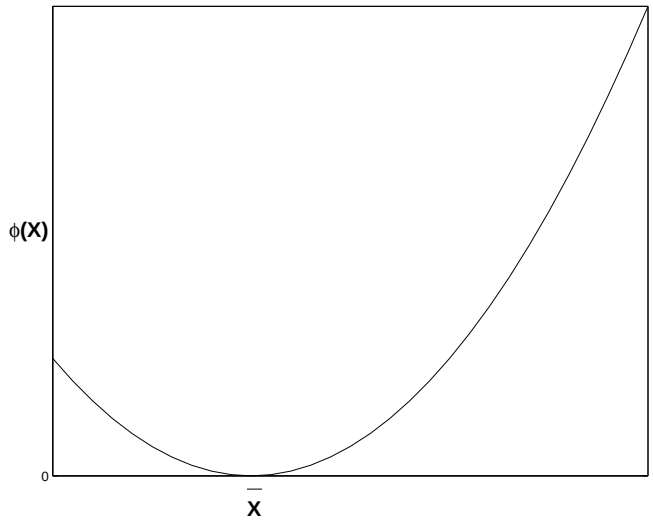

Figure 4.2: Graph of $\phi(x)$

$p^{(s)}(x)$ will decay exponentially fast (w.r.t. ϵ) away from $x = \overline{x}$. We inspect the region around $x = \overline{x}$ by looking at the Taylor expansion of $p^{(s)}(x)$

$$p^{(s)}(x) = \frac{C_2}{a(x)} e^{-2\left[0+0+\frac{1}{2}\phi''(\overline{x})(x-\overline{x})^2+\frac{1}{6}\phi'''(\overline{x})(x-\overline{x})^3+\ldots\right]/\epsilon^2}$$

and then applying the transformation

$$x = \overline{x} + \xi\epsilon$$

giving

$$p^{(s)}(\xi) = \frac{C_2}{a(\overline{x}+\xi\epsilon)} e^{-\phi''(\overline{x})\xi^2+\frac{1}{3}\phi'''(\overline{x})\xi^3\epsilon-\ldots}.$$

Letting $\epsilon \to 0$ we find

$$p^{(s)}(\xi) = \frac{\widetilde{C}_2}{a(\overline{x})} e^{-\phi''(\overline{x})\xi^2}$$

or

$$p^{(s)}(x;\epsilon) = \frac{\widetilde{C}_2}{a(\overline{x})} e^{-\phi''(\overline{x})(\overline{x}-x)^2/\epsilon^2}.$$

Furthermore, $p^{(s)}(x;\epsilon)$ can be written as

$$p^{(s)}(x;\epsilon) = \frac{\widetilde{C}_2\sqrt{2\pi\omega^2}}{a(\overline{x})} \cdot \frac{1}{\sqrt{2\pi\omega^2}} e^{-\frac{(x-\overline{x})^2}{2\omega^2}} \tag{4.11}$$

where

$$\omega^2 = \frac{\epsilon^2}{2\phi''(\overline{x})} = -\frac{a(\overline{x})\epsilon^2}{2b'(\overline{x})}$$

and

$$\frac{1}{\sqrt{2\pi\omega^2}} e^{-\frac{(x-\overline{x})^2}{2\omega^2}} \sim N(\overline{x}, \omega^2)$$

(i.e. the quantity on the left is normally distributed with mean $\overline{x}$ and variance ω^2). Now we are ready to return to our quest to find the constant $C(\epsilon)$. We do this by evaluating the integral

$$\int_{\delta\epsilon^2}^{\infty} p^{(s)} LT - TMp^{(s)} dx. \tag{4.12}$$

Filling in the operators L and M,

$$L = b(x)\frac{\partial}{\partial x} + \frac{\epsilon^2}{2} a(x) \frac{\partial^2}{\partial x^2}$$

$$M = -\frac{\partial}{\partial x}(b(x)\cdot) + \frac{\epsilon^2}{2}\frac{\partial^2}{\partial x^2}(a(x)\cdot)$$

we find that (4.12) becomes

$$\begin{aligned}\frac{\epsilon^2}{2}\int_{\delta\epsilon^2}^{\infty} \left\{a(x)p^{(s)}(x)\right\} T''(x) - \left\{a(x)p^{(s)}(x)\right\}'' T(x) dx \\ + \int_{\delta\epsilon^2}^{\infty} b(x)p^{(s)}(x)T'(x) + (b(x)p^{(s)}(x))'T(x) dx.\end{aligned}$$

Using *Green's Formula* (see [61, p. 386]), the first term reduces to

$$\frac{\epsilon^2}{2}\left\{(a(x)p^{(s)}(x))T'(x) - (a(x)p^{(s)}(x))'T(x)\right\}\Big|_{x=\delta\epsilon^2}^{\infty}$$

Partial integration transforms the second term to

$$b(x)p^{(s)}(x)T(x)\Big|_{x=\delta\epsilon^2}^{\infty}.$$

Thus our integral (4.12) becomes

$$\begin{aligned}\int_{\delta\epsilon^2}^{\infty} p^{(s)}LT - TMp^{(s)}dx &= \Big[\tfrac{\epsilon^2}{2}a(x)\left\{p^{(s)}(x)T'(x) - T(x)p'^{(s)}(x)\right\} \\ &\quad + \left(b(x) - \tfrac{\epsilon^2}{2}a'(x)\right)p^{(s)}(x)T(x))\Big]\Big|_{x=\delta\epsilon^2}^{\infty}.\end{aligned} \tag{4.13}$$

Note that we couldn't start integration at $x = 0$, because $p^{(s)}(x)$ is singular at that point. We will let $\delta \to 0$ afterwards. As before, we have $LT = -1$, $Mp^{(s)} = 0$, at $x = \infty$ $p^{(s)}(x)$ and $p^{(s)'}(x)$ are exponentially small and finally $T(\delta\epsilon^2) \to 0$ as $\delta \to 0$. Thus formula (4.13) reduces to

$$-\int_{\delta\epsilon^2}^{\infty} p^{(s)}(x)dx = -\frac{\epsilon^2}{2}a(\delta\epsilon^2)p^{(s)}(\delta\epsilon^2)T'(\delta\epsilon^2).$$

At the left-hand side, we use that the integral gets its largest contribution around $x = \overline{x}$ and thus

$$\begin{aligned}-\int_{\delta\epsilon^2}^{\infty} p^{(s)}(x)dx \approx -\int_{\delta\epsilon^2}^{\infty} p^{(s)}(x;\epsilon)dx &= -\tfrac{\widetilde{C}_2\sqrt{2\pi\omega^2}}{a(\overline{x})}\int_{\delta\epsilon^2}^{\infty}\frac{1}{\sqrt{2\pi\omega^2}}e^{-\frac{(x-\overline{x})^2}{2\omega^2}}dx \\ &= -\tfrac{\widetilde{C}_2\sqrt{2\pi\omega^2}}{a(\overline{x})} = -\tfrac{\widetilde{C}_2\sqrt{2\pi\omega^2}}{\alpha+\beta}.\end{aligned} \tag{4.14}$$

At the right-hand side we find

$$\begin{aligned}-\frac{\epsilon^2}{2}a(\delta\epsilon^2)p^{(s)}(\delta\epsilon^2)T'(\delta\epsilon^2) &\overset{(a)}{=} -\frac{\epsilon^2}{2}a(\delta\epsilon^2)p^{(s)}(\delta\epsilon^2)C(\epsilon)\tau'(\delta\epsilon^2) \\ &\overset{(b)}{=} -\frac{\epsilon^2}{2}a(\delta\epsilon^2)p^{(s)}(\delta\epsilon^2)C(\epsilon)\frac{2\alpha}{\beta\epsilon^2}e^{\frac{-2\alpha\delta\epsilon^2}{\beta\epsilon^2}} \\ &\overset{(c)}{=} -\frac{\epsilon^2}{2}a(\delta\epsilon^2)C_2\frac{e^{-2\phi(\delta\epsilon^2)/\epsilon^2}}{a(\delta\epsilon^2)}C(\epsilon)\frac{2\alpha}{\beta\epsilon^2}e^{\frac{-2\alpha\delta\epsilon^2}{\beta\epsilon^2}} \\ &\overset{(d)}{=} -\frac{C_2\alpha}{\beta}e^{-2\phi(0)/\epsilon^2}C(\epsilon)\end{aligned} \tag{4.15}$$

where we've used

(a) $T(x) \approx C(\epsilon)\tau(x;\epsilon)$, with $\tau(x;\epsilon)$ as in (4.9);

(b) the expression (4.9) for $\tau(x;\epsilon)$;

(c) the expression (4.10) for the stationary distribution $p^{(s)}(x)$;

(d) $\delta \to 0$ and some cleaning up.

Putting the left-hand side (4.14) and the right-hand side (4.15) together gives us

$$-\frac{\widetilde{C}_2\sqrt{2\pi\omega^2}}{\alpha+\beta} \approx -\frac{C_2\alpha}{\beta}e^{-2\phi(0)/\epsilon^2}C(\epsilon)$$

and thus (observe that $C_2 \to \widetilde{C}_2$ as $\epsilon \to 0$)

$$C(\epsilon) \approx \frac{\beta\sqrt{2\pi\omega^2}}{(\alpha+\beta)\alpha}e^{2\phi(0)/\epsilon^2}. \tag{4.16}$$

Note that indeed $C(\epsilon)$ gets larger as ϵ gets larger, as we required before. So finally, we come to point where we can write down the approximated expression of the expected extinction time $T(x)$ (using (4.16) and (4.9)):

$$T(x;\epsilon) = (1 - e^{\frac{-2\alpha x}{\beta\epsilon^2}})\frac{\beta\sqrt{2\pi\omega^2}}{(\alpha+\beta)\alpha}e^{2\phi(0)/\epsilon^2} \tag{4.17}$$

where

$$\omega^2 = \frac{\epsilon^2}{2\phi''(x)} = -\frac{a(\overline{x})\epsilon^2}{2b'(\overline{x})} = -\frac{(\alpha+\beta)\epsilon^2}{2\cdot-\alpha} = \frac{(\alpha+\beta)\epsilon^2}{2\alpha}.$$

The result can be seen as the solid red line in Figure 4.3, where we have plotted T against K. We used parameter values $b = 30$, $d = 29$ and $x = 1$. We will now attempt to verify this result with a simulation of the system.

Simulation

The listing A.7 shows the code for the Monte Carlo simulation of the SDE (4.2). Its EM-approximation is given by

$$X_{j+1} = X_j + \alpha(1-X_j)X_j\Delta t + \frac{1}{\sqrt{K}}\sqrt{\beta X_j + \alpha X_j^2}dW_j, \qquad j = 0, 1, \ldots.$$

where $K = \epsilon^{-2}$ as before. The parameters are again $b = 30$, $d = 29$ and $X_0 = 1$ and we chose to perform simulations with $K = 10, 15, \ldots, 60$. The stepsize is set to `dt=0.01` and we performed `L=20` MC-simulations with `M=50` sample paths of the SDE. The code is analogous to the code from the example of Section 3.4. We recognize the four `while`-loops. The `if`-statement in the inner loop checks whether the SDE has crossed $X = 0$ (i.e. whether the population has become extinct). The array `CIT` finally contains the required

K	I
10	(0.6844 , 0.7738)
15	(1.0069 , 1.1171)
20	(1.3231 , 1.4692)
25	(1.5599 , 1.7138)
30	(1.9386 , 2.1644)
35	(2.2786 , 2.5221)
40	(2.6497 , 2.9195)
45	(3.1882 , 3.4497)
50	(3.5489 , 3.8257)
55	(3.9817 , 4.3578)

Table 4.1: Simulation results of the population dynamics model

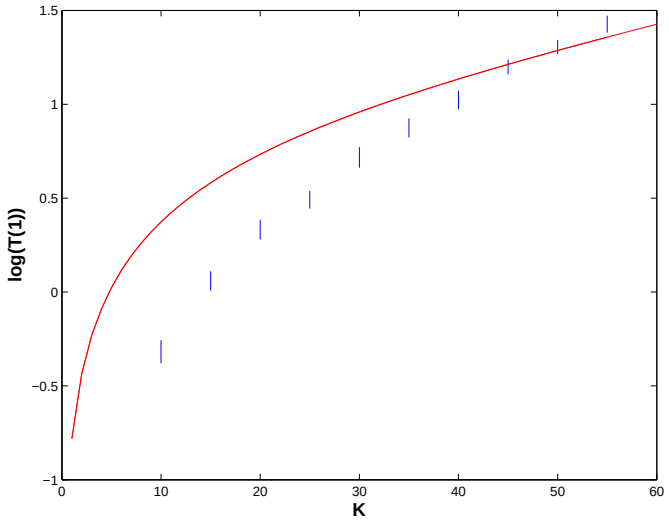

Figure 4.3: The SP-approximation (4.17) and the results of a Monte Carlo simulation

confidence intervals. These intervals are shown in Table 4.1 as I and plotted in Figure 4.3.

Conclusion

We see from the data in Figure 4.3 that the expected extinction time of the population becomes larger as ϵ gets smaller. This is in agreement with what we said before since, in the absence of noise, the population won't become extinct and thus the expected extinction time should be infinitely large. The simulation results have the same behavior : as K gets larger, the confidence intervals move upward.

Remember that both ways of obtaining the expected exit time are not exact : they are approximations. How do we expect the graph of the true expected exit time to look? We know that the SP-approximation becomes more accurate as K gets larger. The simulation results

are influenced by discretization errors and thus will become more inaccurate. Thus for small values of K, we trust the simulation results and we expect the graph of the true expected exit time to run through the confidence intervals. For larger values of K, we trust the SP-approximation.

The model we treated in this section, was chosen because it could be derived from scratch and because it could be used to illustrate the theory in this document. Population dynamics is a popular subject with mathematical modelers and thus many other models can be found, see e.g. [17, 18, 21, 22, 23].

4.2 Firing of a Neuron

Setting

Over the past hundred years, a lot of research has been done on the structure and function of the brain. The elementary processing units of the central nervous system are called 'neurons'. Every neuron is connected to many other neurons and together they form a complex network. A tiny piece of this network can be seen in Figure 4.4. The drawing shows some triangular shapes A, B, C, D, E (the cell-bodies of the neurons) and a lot of wires interconnecting the neurons. In reality, a region of one cubic millimeter contains over 10^4 cell bodies and kilometers of wire. Besides neurons, there are also other elements in the cortex. For instance, there are supportercells (called glia cells) which are used for energy supply and structural stabilization of braintissue. The cortex also contains different types of neurons, but one type is by far the most common. It is the *spiking neuron* or *firing neuron* and it is the one we will consider in this section.

A neuron consists of three parts : the dendrites, the soma (also called cell-body) and the axon (see Figure 4.5). Roughly speaking, the dendrites are the input devices that deliver information from other neurons to the soma. The soma is the 'central processing unit' which performs some processing step and then produces output. This output is then taken over by the axon and passed on to dendrites of other neurons. Neurons send their information using an electrochemical process. The most important electrically charged chemicals (called 'ions') are Sodium (Na^+) and Potassium (K^+). There are also some negatively charged molecules and ions present. The soma is surrounded by a membrane which lets through some ions and blocks others. This passing and blocking of ions is done by opening and closing so-called 'ion channels'. Besides these channels, there is also a pump in the mem-

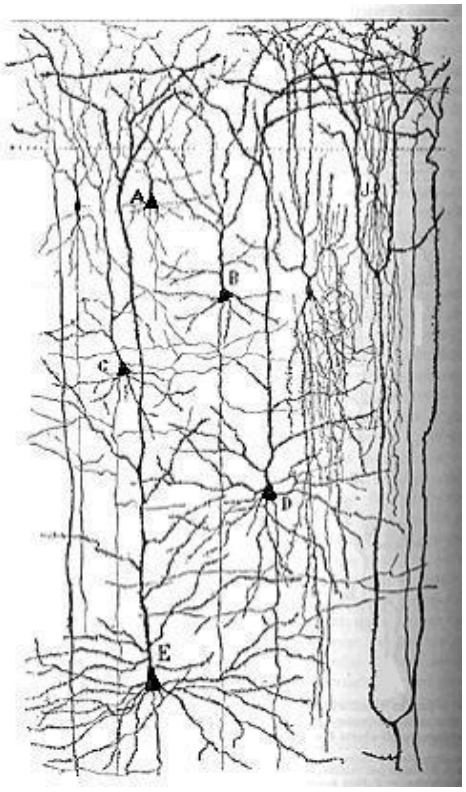

Figure 4.4: Drawing of a part of the human cortex by Santiago Ramón y Cajal [59]

brane which pushes out three Na^+ ions for every two K^+ ions it pushes in.
When all these forces balance and when the neuron receives no input signals, then it is said to be at rest. Its situation is then characterized by the following statements:

- The inside of the membrane is negatively charged relative to the outside. This negative potential across the membrane is called *resting potential* and is about -70 mV. Although the different ions try to settle this balance, they cannot because the membrane only allows some ions to pass;
- There are relatively many K^+ ions on the inside of the membrane and relatively many Na^+ ions on the outside.
- K^+ ions can easily pass through the membrane whilst the Na^+ ions cannot.

Inputsignals to the neuron can be both positive (excitory) and negative (inhibitory). When the input is positive, the Na^+-channels open and the Na^+ ions rush in (see Figure 4.6). This causes the membrane potential to become less negative ('depolarized'). When the membrane is depolarized to about -50mV (this is called the *firing threshold*), all Na^+ channels

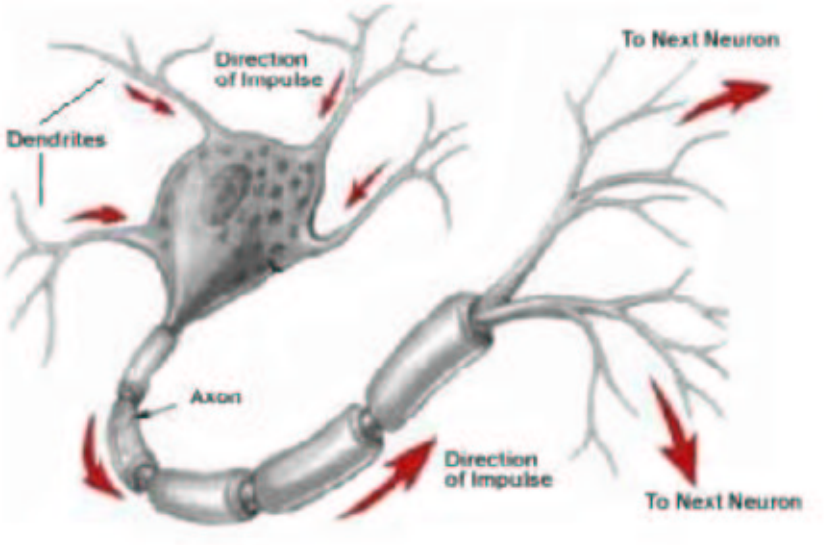

Figure 4.5: Schema of a neuron

open and the inside suddenly becomes positive relative to the outside. The neuron is said to be firing (therefore it is called a 'firing neuron'). By this time the K^+-channels open (and the Na^+-channels start to close) and the K^+ ions rush out to balance the difference in voltage. The K^+-channels actually stay open a bit to long causing the membrane potential to fall below the resting potential (the membrane is then 'hyperpolarized'). The channels and the pump now gradually reset the concentrations of the ions back to their original values and the membrane returns to rest. During the period that the membrane is returning to rest, there can be no firing of the neuron. This time is called the *refractory period*. Meanwhile, the axon has taken over the high potential and delivers it to the place where it makes contact with the dendrite of another neuron. The size of the signal and the state of the other neuron then decide whether to pass the signal as a positive or a negative input signal.

Similarly, negative signals cause the membrane to be hyperpolarized. It is not so that the membrane can become endlessly hyperpolarized. At a certain value, called the inhibitory reversal potential, the negative signals no longer hyperpolarize the membrane but depolarize it. There is also an excitory reversal potential for positive pulses, but it plays a less important role because the firing threshold is usually far below the excitory reversal potential.

The aspect of a firing neuron that is usually modelled is the difference of the membrane potential from the resting potential, see e.g. [19, 29, 30]. We took a model that is due to Kallianpur [28]. The SDE found there takes the form

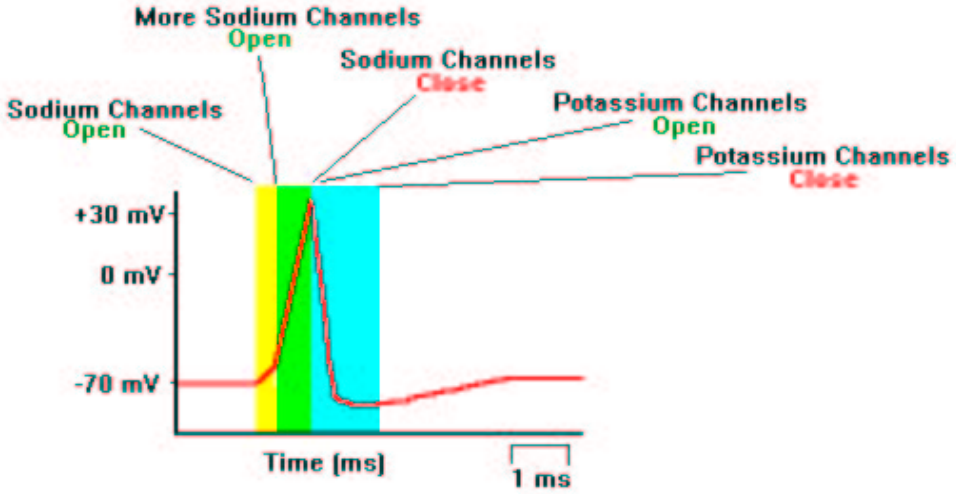

Figure 4.6: Membrane potential when a neuron fires.

$$\begin{aligned} dX(t) &= \left[\tfrac{-X(t)}{\nu} + \lambda\alpha(VE - X(t)) + \omega\beta(X(t) - VI)\right] dt + \\ &\quad \sqrt{\lambda\alpha^2(VE - X(t))^2 + \omega\beta^2(X(t) - VI)^2}\, dW(t), \quad X(0) = x, \end{aligned}$$

where ν is the refractory period, λ is the size of excitory signals, ω is the size if inhibitory signals, VE is the size of the excitory reversal potential and VI the size of the inhibitory reversal potential. Finally $\alpha \in (0, 1)$ and $\beta \in (-1, 0)$ are parameters. We consider the excitory and inhibitory signals to be of the same magnitude and thus we set $\lambda = \omega := \epsilon^2$. This gives us

$$\begin{aligned} dX(t) &= \left[\tfrac{-X(t)}{\nu} + \epsilon^2\alpha(VE - X(t)) + \epsilon^2\beta(X(t) - VI)\right] dt + \\ &\quad \epsilon\sqrt{\alpha^2(VE - X(t))^2 + \beta^2(X(t) - VI)^2}\, dW(t), \qquad X(0) = x. \end{aligned} \tag{4.18}$$

We are interested in finding the probability that the membrane potential is depolarized to the firing threshold and when this is expected to happen. We will call this the firing probability and the expected firing time respectively. Of course this depends on the initial value of the membrane potential (x). We will again use the differential equations from Section 2.3 and the singular perturbations technique from the previous section to approximate their solutions.

Singular perturbation analysis

Before we can get started with the SP-analysis, we rewrite the SDE (4.18) in the form

$$dX(t) = b(X(t);\epsilon)dt + \epsilon\sqrt{a(X(t))}dW(t),$$

with

$$b(x;\epsilon) = \frac{-x}{\nu} + \epsilon^2\alpha(VE - x) + \epsilon^2\beta(x - VI) \tag{4.19}$$

and

$$a(x) = \alpha^2(VE - x)^2 + \beta^2(x - VI)^2. \tag{4.20}$$

This will make the differential equations a lot easier to read.
The starting value x can be chosen anywhere from $-\infty$ to the firing threshold. We denote the firing threshold by S. The firing probability $u(x)$ now satisfies

$$Lu = 0, \tag{4.21a}$$

where the operator L takes the form

$$L = b(x)\frac{\partial}{\partial x} + \frac{1}{2}a(x)\frac{\partial^2}{\partial x^2}$$

with $b(x)$ and $a(x) = \sigma(x)^2$ as in (4.19) and (4.20). Of course, we want the firing probability to be 1 if we start in $x = S$ and thus we set

$$u(S) = 1. \tag{4.21b}$$

The boundary condition at $-\infty$ is chosen to be reflecting, i.e.

$$u'(-\infty) = 0. \tag{4.21c}$$

The solution to (4.21) is simply given by $u \equiv 1$ (see again the remark on page 20). This means that the neuron will always fire. How long it is expected that it takes the neuron to fire, can be found from the differential equation for the expected firing time $T(x)$. It satisfies

$$LT = -1$$

with

$$T'(-\infty) = 0, \qquad T(S) = 0.$$

As in the previous example, we set $T(x) = \tau(x)C(\epsilon)$. We again find an outer solution $\tau(x) \equiv 1$. Now we zoom into the area around $x = S$ by using the coordinate stretching

$$x = S + \xi\epsilon^2.$$

We find that $\tau(x)$ satisfies

$$b(S;\epsilon) \cdot \epsilon^{-2}\frac{\partial \tau}{\partial \xi} + \frac{\epsilon^2}{2}\epsilon^{-4}\frac{\partial^2 \tau}{\partial \xi^2}a(S) = -C(\epsilon)^{-1},$$

with boundary condition

$$\tau(0) = 0 \tag{4.22}$$

and matching condition

$$\tau(-\infty) = 1. \tag{4.23}$$

After multiplying by ϵ^2 and letting $\epsilon \to 0$, we obtain for τ

$$b(S;0)\frac{\partial \tau}{\partial \xi} + \frac{a(S)}{2}\frac{\partial^2 \tau}{\partial \xi^2} = 0$$

which together with (4.22) and (4.23) yields

$$\tau(\xi) = 1 - e^{\frac{-2b(S;0)}{a(S)}\xi},$$

or in original coördinates

$$\tau(x;\epsilon) = 1 - e^{\frac{-2b(S;0)}{a(S)}\frac{x-S}{\epsilon^2}}.$$

The constant $C(\epsilon)$ is again obtained from the integral

$$\int_{-\infty}^{S} p^{(s)}LT - TMp^{(s)}dx$$

which takes the form

$$\begin{aligned}\int_{-\infty}^{S} p^{(s)}LT - TMp^{(s)}dx &= \left[\tfrac{\epsilon^2}{2}a(x)\left\{p^{(s)}(x)T'(x) - T(x)p'^{(s)}(x)\right\}\right. \\ &\quad \left. + \ (b(x) - \tfrac{\epsilon^2}{2}a'(x))p^{(s)}(x)T(x)\right]\Big|_{x=-\infty}^{S}.\end{aligned}$$

This reduces to

$$-\int_{-\infty}^{S} p^{(s)}(x)dx = \frac{\epsilon^2}{2}a(S)p^{(s)}(S)T'(S). \tag{4.24}$$

Here, $p^{(s)}(x)$ is again given by (see (4.10))

$$p^{(s)}(x) = C_2\frac{e^{-2\phi(x)/\epsilon^2}}{a(x)}, \qquad C_2 = \text{constant} \tag{4.25}$$

and its approximation around $x = \overline{x}$ (see 4.11)

$$p^{(s)}(x;\epsilon) = \frac{\widetilde{C}_2\sqrt{2\pi\omega^2}}{a(\overline{x})}\cdot\frac{1}{\sqrt{2\pi\omega^2}}e^{-\frac{(x-\overline{x})^2}{2\omega^2}} \tag{4.26}$$

where

$$\omega^2 = \frac{\epsilon^2}{2\phi''(\overline{x})} = -\frac{a(\overline{x})\epsilon^2}{2b'(\overline{x};0)}$$

and $\overline{x}$ is again the equilibrium solution to $x'(t) = b(x;0)$ (i.e. $\overline{x} = 0$).

At the left-hand side of (4.24) we use that the integral gets its largest contribution around $x = \overline{x}$ and thus we use the approximation (4.26) to arrive at

$$-\int_{-\infty}^{S} p^{(s)}(x)dx \approx -\frac{\widetilde{C}_2\sqrt{2\pi\omega^2}}{a(\overline{x})}.$$

At the right-hand side we obtain as before

$$\begin{aligned}\tfrac{\epsilon^2}{2}a(S)p^{(s)}(S)T'(S) &= \tfrac{\epsilon^2}{2}a(S)\cdot C_2\tfrac{e^{-2\phi(S)/\epsilon^2}}{a(S)}\cdot\tfrac{2b(S;0)}{a(S)\epsilon^2}C(\epsilon) \\ &= \tfrac{C_2 b(S;0)}{a(S)}e^{-2\phi(S)/\epsilon^2}C(\epsilon).\end{aligned}$$

Putting the left-hand side and right-hand side together gives us the constant $C(\epsilon)$:

$$C(\epsilon) \approx \frac{a(S)\sqrt{2\pi\omega^2}}{-b(S;0)a(\overline{x})}e^{2\phi(S)/\epsilon^2}.$$

The SP-approximation of $T(x)$ then becomes

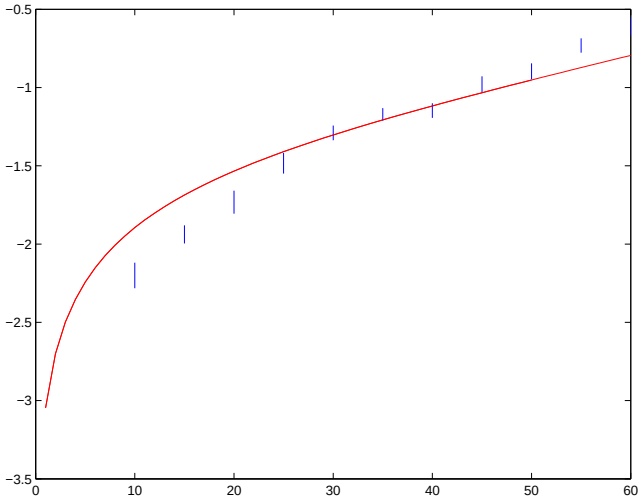

Figure 4.7: Simulation results of the neuron model from Table 4.2 and the SP-approximation from expression (4.27).

$$T(x;\epsilon) = (1 - e^{\frac{-2b(S;0)}{a(S)}\frac{x-S}{\epsilon^2}})\frac{a(S)\sqrt{2\pi\omega^2}}{-b(S;0)a(\overline{x})}e^{2\phi(S)/\epsilon^2} \tag{4.27}$$

where

$$\begin{aligned} b(S;0) &= -S/\nu \\ a(S) &= \alpha^2(VE - S)^2 + \beta^2(S - VI)^2 \\ a(\overline{x}) &= \alpha^2 VE^2 + \beta^2 VI^2 \\ \omega^2 &= -\frac{a(\overline{x})\epsilon^2}{2b'(\overline{x};0)} = \frac{(\alpha^2 VE^2 + \beta^2 VI^2)\epsilon^2}{2/\nu}. \end{aligned}$$

The solid line in Figure 4.7 shows $\log(T(S))$ as in expression (4.27) plotted against $K = \frac{1}{\epsilon^2}$. We used parameters $\nu = 0.1$, $\alpha = 0.9$, $\beta = -0.9$ and $x = 0$. We set the excitory and inhibitory reversal potential at 50 and -7 respectively. The firing threshold S is set at 2. The blue vertical lines are confidence intervals for the expected exit time, which we obtained from a simulation of the system. The code corresponding to this simulation can be found in Appendix A.8.

Conclusion

If the neuron receives no input, then it should not fire (i.e. the expected firing time is infinitely large). In expression (4.27), we can see that $T(x;\epsilon)$ is exponential in K and thus the expected firing time gets exponentially larger as the input ϵ decreases to 0. Thus the SP-approximation behaves the way we expect it to.

$K = \frac{1}{\epsilon^2}$	Tsim
10	(0.1021 , 0.1202)
15	(0.1359 , 0.1526)
20	(0.1643 , 0.1904)
25	(0.2123 , 0.2422)
30	(0.2628 , 0.2885)
35	(0.2980 , 0.3226)
40	(0.3030 , 0.3328)
45	(0.3559 , 0.3952)
55	(0.3876 , 0.4295)

Table 4.2: Simulation results of a Monte Carlo simulation of the neuron model

As we observed in the previous section, we expect the true graph of the expected firing time to go through the confidence intervals coresponding to the small values of K. As K gets larger, we should trust the SP-approximation more than the simulation results. It is not clear where exactly we switch from the one to the other. It seems that the confidence interval corresponding to $K = 40$ is to low compared to the others and the interval corresponding to $K = 55$ to high. We probably have to trust the SP-approximation after $K = 50$.
A downside to the approach we took is that we had to put the firing threshold at the low value of 2. Since the neuron fires if the membrane potential is depolarized from -70mV to -50mV, the firing threshold should have been set to 20. We had to lower it, because otherwise simulations would have taken to long. However, it does not stop this from being a nice illustration of the theory in this thesis.

4.3 Particle movement

In this section we will consider a model for the movement of a particle in the air. Contrary to the models in the previous two sections, the model in this section is two-dimensional. We suppose that the particle can only move left/right and up/down. We also suppose that movement of the particle is influenced by a constant wind to the right. Due to the collision of the particle with molecules in the air, the particle doesn't just move along with the wind. It also performs a random motion. A model from the book by Grasman and Van Herwaarden [16, p. 76] is

$$\begin{aligned} dX(t) &= \nu dt + \sqrt{2D_L} dW_L(t) \qquad , X(0) = x \\ dY(t) &= \sqrt{2D_T} dW_T(t) \qquad , Y(0) = y \end{aligned} \tag{4.28}$$

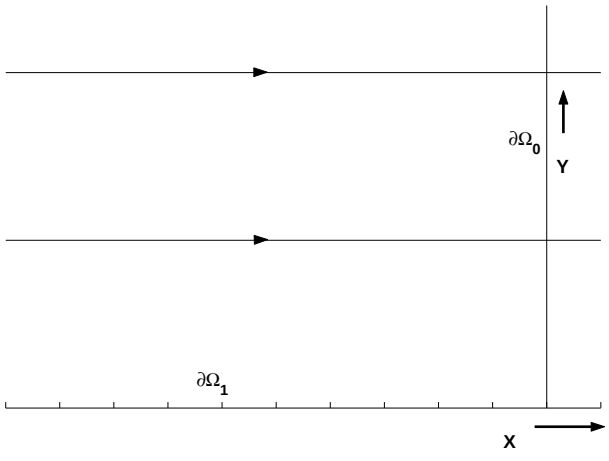

Figure 4.8: Flow parallel to the x-axis

where $X(t)$ is the movement in the horizontal direction, $Y(t)$ is the movement in the vertical direction and ν indicates the speed of the wind. The random movement is modelled via a Brownian motion in the longitudinal (horizontal) direction ($W_L(t)$) and a Brownian motion in the transversal (vertical) direction ($W_T(t)$). The magnitude of these random motions is indicated by D_L and D_T respectively.

We want to know the probability that the particle reaches the ground and how long that would take. This could be interesting if e.g. the particle is polluted. We suppose that the ground $\partial\Omega_1$ is given by $\{(X,Y)|X<0, Y=0\}$. The particle can freely move in the area $\Omega = \{(X,Y)|X<0,\ Y>0\}$. If the particle crosses the line $\partial\Omega_0 = \{(X,Y)|X=0, Y>0\}$, then we are not interested in its movement anymore. All this is illustrated in Figure 4.8. We will use the same approach that was used in the previous two sections.

For the model (4.28), the operator L takes the form

$$L = \nu\frac{\partial}{\partial x} + \left(D_L\frac{\partial^2}{\partial x^2} + D_T\frac{\partial^2}{\partial y^2}\right).$$

The probability that the particle reaches the ground ($u(x,y)$) is thus given by the differential equation

$$Lu = 0. \tag{4.29a}$$

The boundary conditions are obtained from the practical considerations above and are

$$u(x,y) = 0 \quad \text{for } (x,y) \text{ at } \partial\Omega_0 \quad \text{and} \quad u(x,y) = 1 \quad \text{for } (x,y) \text{ at } \partial\Omega_1. \tag{4.29b}$$

The time that it is expected to take the particle to reach the ground ($T_1(x,y)$) is then given by $\frac{T(x,y)}{u(x,y)}$ with $T(x,y)$ the solution to

$$LT = -u \tag{4.30a}$$

with boundary conditions

$$T(x,y) = 0 \quad \text{for } (x,y) \text{ at } \partial\Omega_0 \quad \text{and} \quad T(x,y) = 0 \quad \text{for } (x,y) \text{ at } \partial\Omega_1. \tag{4.30b}$$

Having stated the differential equations, we can get started to find the SP-approximations to their solutions.

Singular perturbations

First we set $D_L = D_T = \frac{\epsilon^2}{2}$ in order to have the same notation as in the previous two sections. Inserting this into (4.29) yields

$$\nu\frac{\partial u}{\partial x} + \frac{\epsilon^2}{2}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) = 0 \tag{4.31a}$$

with boundary conditions

$$u(x,y) = 0 \quad \text{for } (x,y) \text{ at } \partial\Omega_0 \quad \text{and} \quad u(x,y) = 1 \quad \text{for } (x,y) \text{ at } \partial\Omega_1. \tag{4.31b}$$

Looking at this differential equation, we see that it no longer has a simple solution and thus we will make an SP-approximation. Letting $\epsilon \to 0$ and using the first boundary condition, we find that $u \equiv 0$ for y away from $\partial\Omega_1$. To inspect the boundary layer around $\partial\Omega_1$, we introduce the local coordinate

$$\xi = \frac{y}{\epsilon}. \tag{4.32}$$

Substitution into (4.31) and letting $\epsilon \to 0$ gives

$$\nu\frac{\partial u}{\partial x} + \frac{1}{2}\frac{\partial^2 u}{\partial \xi^2} = 0$$

with boundary conditions

$$u(x,0) = 1, \quad u(0,\xi) = 0$$

and matching condition

$$\lim_{\xi\to\infty} u(x,\xi) = 0.$$

With the aid of the transformation $\frac{\xi}{\sqrt{-x/\nu}}$, we find the solution

$$u(x,\xi) = \sqrt{\frac{2}{\pi}} \int_{\frac{\xi}{\sqrt{-x/\nu}}}^{\infty} e^{-\frac{1}{2}s^2} ds.$$

Returning to original variables, we find

$$u(x,y;\epsilon) = \sqrt{\frac{2}{\pi}} \int_{\frac{y}{\epsilon\sqrt{-x/\nu}}}^{\infty} e^{-\frac{1}{2}s^2} ds. \tag{4.33}$$

We now turn our attention to the quantity $T(x,y)$. Inserting ϵ into (4.30) yields

$$\nu\frac{\partial T}{\partial x} + \frac{\epsilon^2}{2}\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right) = -u \tag{4.34a}$$

with boundary conditions

$$T(x,y) = 0 \quad \text{for } (x,y) \text{ at } \partial\Omega_0 \quad \text{and} \quad T(x,y) = 0 \quad \text{for } (x,y) \text{ at } \partial\Omega_1. \tag{4.34b}$$

In the same way as we did for u, we find the outer solution $T \equiv 0$. We again zoom into the boundary layer by introducing the local coordinate ξ (as in (4.32)). Substitution into (4.34) gives

$$\nu\frac{\partial T}{\partial x} + \frac{1}{2}\frac{\partial^2 T}{\partial \xi^2} = -u(x,\xi)$$

with boundary conditions and matching condition

$$T(x,0) = 0, \quad T(0,\xi) = 0, \quad \lim_{\xi\to\infty} T(x,\xi) = 0.$$

The solution is given by

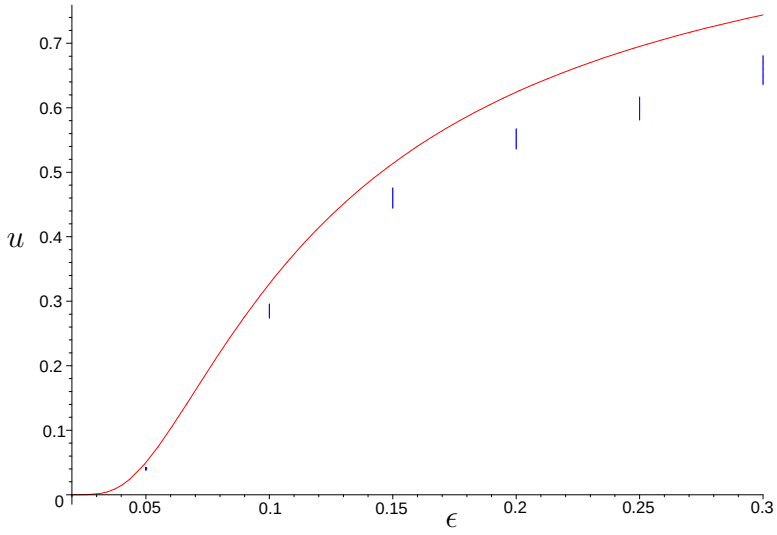

Figure 4.9: SP-approximation to $u(x, y)$ as in (4.33) and the simulation results from Table 4.3

$$T(x,\xi) = \sqrt{\frac{2}{\pi}} \int_{\frac{\xi}{\sqrt{-x/\nu}}}^{\infty} \frac{\xi^2}{s^2} e^{-\frac{1}{2}s^2} ds.$$

or in original variables

$$T(x,y;\epsilon) = \sqrt{\frac{2}{\pi}} \int_{\frac{y}{\epsilon\sqrt{-x/\nu}}}^{\infty} \frac{y^2}{\epsilon^2 s^2} e^{-\frac{1}{2}s^2} ds.$$

Thus $T_1(x, y; \epsilon)$ is given by

$$T_1(x,y;\epsilon) = u(x,y;\epsilon)^{-1} \sqrt{\frac{2}{\pi}} \int_{\frac{y}{\epsilon\sqrt{-x/\nu}}}^{\infty} \frac{y^2}{\epsilon^2 s^2} e^{-\frac{1}{2}s^2} ds. \tag{4.35}$$

The quantities $u(x, y; \epsilon)$ and $T(x, y; \epsilon)$ have been plotted in Figures 4.9 and 4.10 respectively. The blue vertical lines are 90% confidence intervals obtained from a simulation of the system (which we will discuss next).

Simulation

Listing A.9 gives the code for the Monte Carlo simulation. The parameter values we used are $y = 0.12$, $x = -1.5$, $\nu = 1$, and as before $M = 50$, $L = 20$. The value of ϵ is varied from 0.05 to 0.30, which is done by the outer `while`-loop. The inner while loop uses the EM-approximation to (4.28), given by

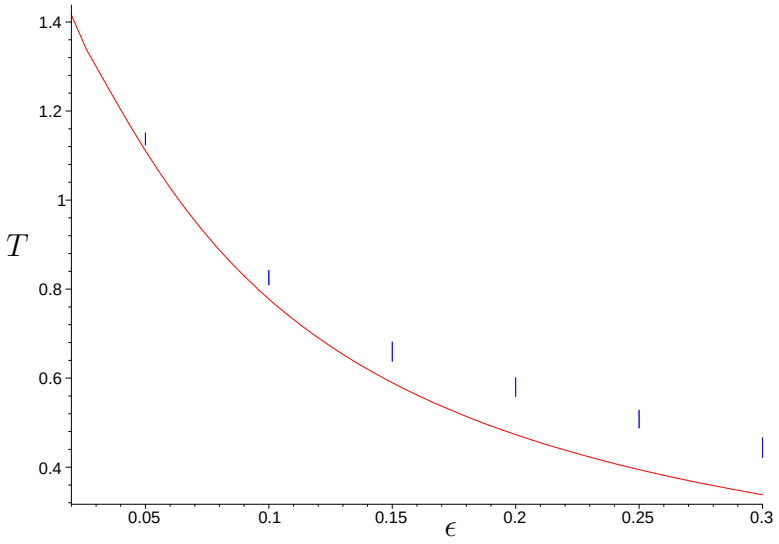

Figure 4.10: SP-approximation to $T_1(x, y)$ as in (4.35) and the simulation results from Table 4.3

$$\begin{cases} X_j = X_{j-1} + \nu dt + \sqrt{2D_L} dW L_j \\ Y_j = Y_{j-1} + \sqrt{2D_T} dW T_j \end{cases}$$

to approximate a sample path of the system (4.28). The two `if`-statements check whether it exits through one of the two boundaries. If it exits through $\partial\Omega_1$ (first `if`), then we record its exit time as before. If it exits through $\partial\Omega_0$ (second `if`), then we are not interested in it anymore and we mark it as not-exiting (i.e. we increase `Q`). At the end of each Monte Carlo simulation, we use `Q` in the command

```
Usim(l+1)=sims/(sims+Q);
```

to approximate the probability that the particle hits the ground $\partial\Omega_1$ (as we explained in Section 3.4).

At the end, the arrays `CIU` and `CIT` are filled with the 90% confidence intervals for u and T_1. The resulting intervals, denoted by I_u and I_T respectively, can be found in Table 4.3. They are also plotted in Figures 4.9 and 4.10.

Conclusion

We seen from Figure 4.9 that as the noise gets smaller, the probability that the particle hits the ground becomes smaller. This makes sense, since in the absence of noise the particle will follow the wind and will thus not hit the ground. If the magnitude of the noise is

ϵ	I_u	I_T
0.30	(0.6355 , 0.6811)	(0.4219 , 0.4665)
0.25	(0.5808 , 0.6160)	(0.4876 , 0.5285)
0.20	(0.5385 , 0.5676)	(0.5585 , 0.6013)
0.15	(0.4441 , 0.4760)	(0.6375 , 0.6815)
0.10	(0.2736 , 0.2960)	(0.8092 , 0.8424)
0.05	(0.0380 , 0.0426)	(1.1236 , 1.1508)

Table 4.3: Simulation results for the particle movement model

small and the particle does hit the ground, than it has probably taken a long time to do so. This is confirmed by Figure 4.10, where we see that the SP-approximation to the expected exit time becomes larger as ϵ gets smaller. The same observations can be made about the simulation results and thus both approximations behave as we would expect from their practical interpretation.

4.4 Groundwater pollution

In this section, we consider the flow of groundwater that is confined in a layer called 'aquifer'. We suppose that the thickness of the aquifer is small and thus that we can view the groundwaterflow as a two-dimensional problem. We are interested in the path followed by a single particle, which we consider to be contaminated. We assume that the flow is given by the differential equation

$$\begin{pmatrix} X'(t) \\ Y'(t) \end{pmatrix} = v(X(t), Y(t)), \tag{4.36}$$

with

$$v(x, y) = \begin{pmatrix} -x \\ y \end{pmatrix}. \tag{4.37}$$

We assume that the contaminated particle starts in the point $P = (x, y)$, i.e. the initial condition is

$$\begin{pmatrix} X(0) \\ Y(0) \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix}. \tag{4.38}$$

Observe that the solution of (4.36)-(4.38) is given by

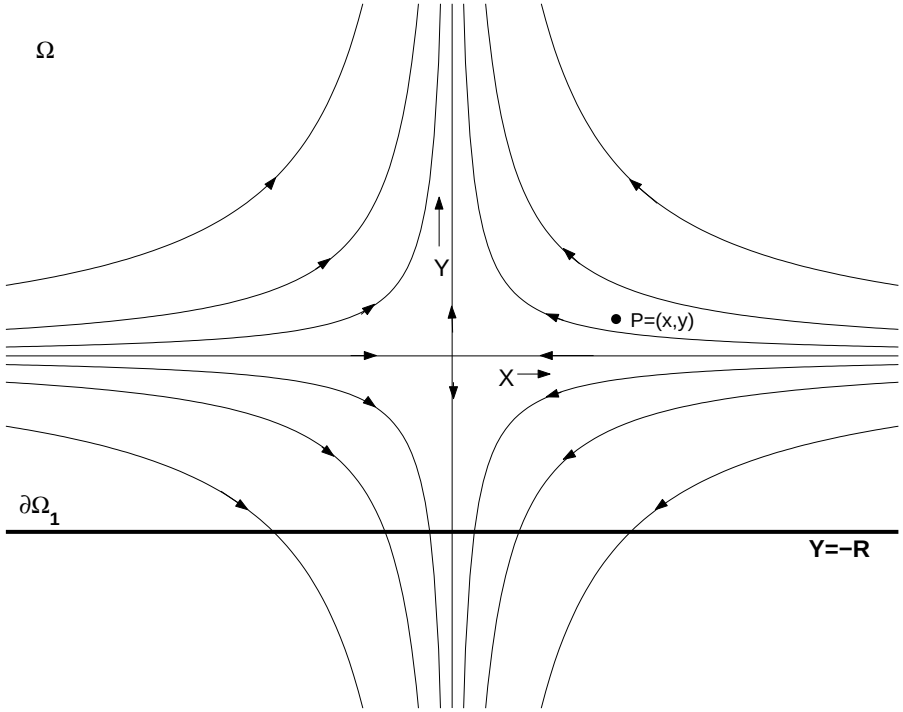

Figure 4.11: Symmetric 2D flow of groundwater in the aquifer

$$\begin{pmatrix} X(t) \\ Y(t) \end{pmatrix} = \begin{pmatrix} xe^{-t} \\ ye^{t} \end{pmatrix}$$

and thus, for a particle following the flow, $X(t)Y(t) = xy$ =constant. The lines in Figure 4.11 symbolize the flow for different values of x and y. Observe that if a particle that follows the flow starts in the upper halfplane, then it will always remain there. A similar statement goes for particles that start in de lower halfplane. The line $\{Y = 0\}$, which separates the two halfplanes, is often called the *separating streamline*.

As in the previous example, it is not sufficient to model the movement simply by the flow. The particle exhibits a random motion which sometimes causes the particle to cross the separating streamline (which is impossible if it would follow the flow). One model describing this behavior is due to Van Herwaarden [31, p. 29] and is given by

$$\begin{aligned} dX(t) &= \left(v_x + \tfrac{\partial D_{xx}}{\partial x} + \tfrac{\partial D_{xy}}{\partial y}\right) dt + \sqrt{2a_L|v|}\tfrac{v_x}{|v|}dW_L(t) + \sqrt{2a_T|v|}\tfrac{v_y}{|v|}dW_T(t) \\ dY(t) &= \left(v_y + \tfrac{\partial D_{xy}}{\partial x} + \tfrac{\partial D_{yy}}{\partial y}\right) dt + \sqrt{2a_L|v|}\tfrac{v_y}{|v|}dW_L(t) - \sqrt{2a_T|v|}\tfrac{v_x}{|v|}dW_T(t) \end{aligned} \tag{4.39}$$

where $v(x,y) = (v_x, v_y)$ is given by (4.37) and

$$\begin{aligned} D_{xx} &= a_T|v| + \tfrac{(a_L - a_T)x^2}{|v|} \\ D_{xy} &= \tfrac{-(a_L - a_T)xy}{|v|} \\ D_{yy} &= a_T|v| + \tfrac{(a_L - a_T)y^2}{|v|}. \end{aligned}$$

The parameters a_T and a_L indicate the magnitude of the random motion in the transversal (vertical) and longitudinal (horizontal) direction respectively. We are interested in the probability that a contaminated particle reaches the boundary $\partial\Omega_1$ and the expected arrival time at that boundary. There could for instance be a drinkwater pump or a farm at this boundary, which makes the arrival of a contaminated particle worth looking at. We continue in the same way as in the previous example.

The operator L now takes the form

$$\begin{aligned} L &= \left(v_x + \frac{\partial D_{xx}}{\partial x} + \frac{\partial D_{xy}}{\partial y}\right)\frac{\partial}{\partial x} + \left(v_y + \frac{\partial D_{xy}}{\partial x} + \frac{\partial D_{yy}}{\partial y}\right)\frac{\partial}{\partial y} + \\ &\quad \frac{1}{2}\left(2Dxx\frac{\partial^2}{\partial x^2} + 4Dxy\frac{\partial^2}{\partial x \partial y} + 2Dyy\frac{\partial^2}{\partial y^2}\right). \end{aligned}$$

which can be simplified to

$$L = -x\frac{\partial}{\partial x} + y\frac{\partial}{\partial y} + \frac{\partial}{\partial x}\left(D_{xx}\frac{\partial}{\partial x} + D_{xy}\frac{\partial}{\partial y}\right) + \frac{\partial}{\partial y}\left(D_{xy}\frac{\partial}{\partial x} + D_{yy}\frac{\partial}{\partial y}\right).$$

As before, the probability that the particle reaches the boundary $\partial\Omega_1$ ($u(x,y)$) satisfies

$$Lu = 0, \quad u(x,-R) = 1 \quad \text{and} \quad \lim_{y\to\infty} u(x,y) = 0. \tag{4.40}$$

The boundary conditions are obvious from the practical interpretation. The time that we expect the particle to reach the boundary $T_1(x,y)$ uses the quantity $T(x,y)$ which is the solution to

$$LT = -u, \quad T(x,-R) = 0 \quad \text{and} \quad \lim_{y\to\infty} T(x,y) = 0.$$

We are now ready to find the SP-approximations to $u(x,y)$ and $T(x,y)$.

We consider the magnitudes of the noise-terms (a_L and a_T) to be small and therefore we set $a_L = a_T = \epsilon^2$. Away from the separating streamline, the particle will follow the flow and thus we let $\epsilon \to 0$. Since $D_{xx}, D_{xy}, D_{yy} \to 0$, the differential equation (4.40) for $u(x, y)$ becomes

$$-x\frac{\partial u}{\partial x} + y\frac{\partial u}{\partial y} = 0.$$

From the boundary conditions, we now see that $u(x, y) = 1$ if starting in the lower halfplane and $u(x, y) = 0$ if starting in the upper half plane. So if the particle starts in the upper halfplane, then it will not hit the boundary ($u = 0$) and if it starts in the lower halfplane then it will hit the boundary ($u = 1$). Looking at Figure 4.11, we see that this is exactly what we would expect from a particle that follows the flow.

We now concentrate on what happens inside the boundary layer. We only consider the case $x > 0$, because of symmetry considerations. In our analysis, we have to exclude a small region around the origin where our approximation is not valid. Since this approximation has only local importance, we will not treat it further. Substitution of the local coordinate $\xi = \frac{y}{\epsilon}$ into (4.40) gives

$$-x\frac{\partial u}{\partial x} + \xi\frac{\partial u}{\partial \xi} + \frac{\partial}{\partial x}\left(D_{xx}\frac{\partial u}{\partial x} + D_{xy}\frac{1}{\epsilon}\frac{\partial u}{\partial \xi}\right) + \frac{1}{\epsilon}\frac{\partial}{\partial \xi}\left(D_{xy}\frac{\partial u}{\partial x} + D_{yy}\frac{1}{\epsilon}\frac{\partial u}{\partial \xi}\right) = 0$$

Letting $\epsilon \to 0$, we find

$$-x\frac{\partial u}{\partial x} + \xi\frac{\partial u}{\partial \xi} + x\frac{\partial^2 u}{\partial \xi^2} = 0$$

with matching conditions

$$u(x, \xi) \to 0 \quad \text{for} \quad \xi \to \infty$$
$$u(x, \xi) \to 1 \quad \text{for} \quad \xi \to -\infty.$$

Setting $\eta = \xi/\sqrt{\frac{2}{3}x}$ yields

$$\eta\frac{\partial u}{\partial \eta} + \frac{\partial^2 u}{\partial \eta^2} = 0$$

which has as a solution

$$u(\eta) = C_1 + C_2 \int_{\eta}^{\infty} e^{-\frac{1}{2}s^2} ds$$

Returning to local coordinates and using the matching conditions gives

$$u(x, \xi) = \frac{1}{\sqrt{2\pi}} \int_{\xi/\sqrt{\frac{2}{3}x}}^{\infty} e^{-\frac{1}{2}s^2} ds.$$

or, in original variables

$$u(x, y; \epsilon) = \frac{1}{\sqrt{2\pi}} \int_{y/(\epsilon\sqrt{\frac{2}{3}x})}^{\infty} e^{-\frac{1}{2}s^2} ds. \tag{4.41}$$

The quantity $T(x, y)$ satisfies $LT = -u$. For starting points in the upper halfplane and outside the boundary layer, this becomes

$$-x\frac{\partial T}{\partial x} + y\frac{\partial T}{\partial y} = 0$$

since $u \approx 0$ there. Together with the boundary condition

$$\lim_{y\to\infty} T(x, y) = 0$$

this gives $T \equiv 0$. For starting points in the lower halfplane and outside the boundary layer, we find for $T(x, y)$

$$-x\frac{\partial T}{\partial x} + y\frac{\partial T}{\partial y} = -1$$

with boundary condition

$$\lim_{y\to -R} T(x, y) = 0.$$

This gives (using the transformation $z = \frac{-y}{2x}$)

$$T(x, y) = -\ln(-y) + \ln(R).$$

For starting points inside the boundary layer, we have ($\xi = \frac{y}{\epsilon}$)

$$-x\frac{\partial T}{\partial x} + \xi\frac{\partial T}{\partial \xi} + x\frac{\partial^2 T}{\partial \xi^2} = -u(x,\xi)$$

with matching conditions

$$T(x,\xi) = 0 \qquad \text{for} \quad \xi \to \infty$$
$$T(x,\xi) = -\ln(-\xi) + \ln(R) - \ln(\epsilon) \qquad \text{for } \xi < 0 \text{ outside the boundary layer.}$$

Set $T(x,\xi) = T_p(x,\xi) + T_h(x,\xi)$ where $T_p(x,\xi)$ is a particular solution and $T_h(x,\xi)$ a homogeneous solution. We easily find a particular solution

$$T_p(x,\xi) = u(x,\xi)\ln(x).$$

Thus $T_h(x,\xi)$ satisfies

$$-x\frac{\partial T_h}{\partial x} + \xi\frac{\partial T_h}{\partial \xi} + x\frac{\partial^2 T_h}{\partial \xi^2} = 0. \tag{4.42}$$

Using new coordinates

$$\tau = \frac{1}{3}x^3 \qquad \text{and} \qquad \eta = x\xi$$

transforms (4.42) into

$$\frac{\partial T_h}{\partial \tau} = \frac{\partial^2 T_h}{\partial^2 \eta}. \tag{4.43}$$

Observe that $T_p(x,\xi) \to 0$ as $\xi \to \infty$ and $T_p(x,\xi) \to \ln(x)$ as $\xi \to \infty$. Thus the matching conditions for (4.43) are

$$T_h(0,\eta) = 0 \quad \text{for} \quad \eta > 0$$
$$T_h(0,\eta) = -\ln(-\eta) + \ln(R) - \ln(\epsilon) \qquad \text{for} \quad \eta < 0.$$

This is satisfied by

$$T_h(\tau,\eta) = \frac{1}{\sqrt{2\pi}}\int_{\eta/\sqrt{2\tau}}^{\infty}\left(-\ln(\epsilon) + \ln(R) - \ln(-\eta + s\sqrt{2\tau})\right)e^{-\frac{1}{2}s^2}ds$$

or in local coordinates

$$T_h(x,\xi) = \frac{1}{\sqrt{2\pi}} \int_{\xi/\sqrt{\frac{2}{3}x}}^{\infty} \left(-\ln(\epsilon) + \ln(R) - \ln(-x\xi + sx\sqrt{\frac{2}{3}x}) \right) e^{-\frac{1}{2}s^2} ds.$$

Thus we have for $T(x,\xi)$

$$\begin{aligned} T(x,\xi) &= \ln(x)u(x,\xi) + (-\ln(\epsilon) + \ln(R) - \ln(x))\, u(x,\xi) + \\ &\quad \tfrac{1}{\sqrt{2\pi}} \int_{\xi/\sqrt{\frac{2}{3}x}}^{\infty} \left(-\ln(-\xi + s\sqrt{\frac{2}{3}x}) \right) e^{-\frac{1}{2}s^2} ds \end{aligned}$$

or

$$\begin{aligned} T(x,y;\epsilon) &= (-\ln(\epsilon) + \ln(R))\, u(x,y) + \\ &\quad \tfrac{1}{\sqrt{2\pi}} \int_{y/(\epsilon\sqrt{\frac{2}{3}x})}^{\infty} \left(-\ln(-\frac{y}{\epsilon} + s\sqrt{\frac{2}{3}x}) \right) e^{-\frac{1}{2}s^2} ds. \end{aligned}$$

The time that we expect the particle to take to reach the boundary ($T_1(x,y)$) is thus approximated by

$$\begin{aligned} T_1(x,y;\epsilon) &= -\ln(\epsilon) + \ln(R) + \\ &\quad \left\{ \int_{y/(\epsilon\sqrt{\frac{2}{3}x})}^{\infty} e^{-\frac{1}{2}s^2} ds \right\}^{-1} \int_{y/(\epsilon\sqrt{\frac{2}{3}x})}^{\infty} \left(-\ln(-\frac{y}{\epsilon} + s\sqrt{\frac{2}{3}x}) \right) e^{-\frac{1}{2}s^2} ds. \end{aligned} \tag{4.44}$$

Simulation

Listing A.10 shows the code that performs `L=20` MC-simulations of size `M=50`. We used stepsize `dt=0.01` and startingpoint $(x,y) = (4, 0.2)$. Simulation of the time that we expect the particle to take to reach the boundary is done in the same way as in the previous sections. For the simulation of the probability that the particle reaches the boundary, we have to be a bit carefull.

A sample path doesn't exit through the boundary $\partial\Omega_1$, if it passes the line $\{Y = \infty\}$. Of course, when simulating the SDE with MATLAB, we can not check if the sample path passes that line. Therefore we 'lower' the boundary at $Y = \infty$ to $Y = 10$. A sample path is now marked as not-exiting if it passes this boundary (i.e. `Q` is incremented). The quantity `Q`

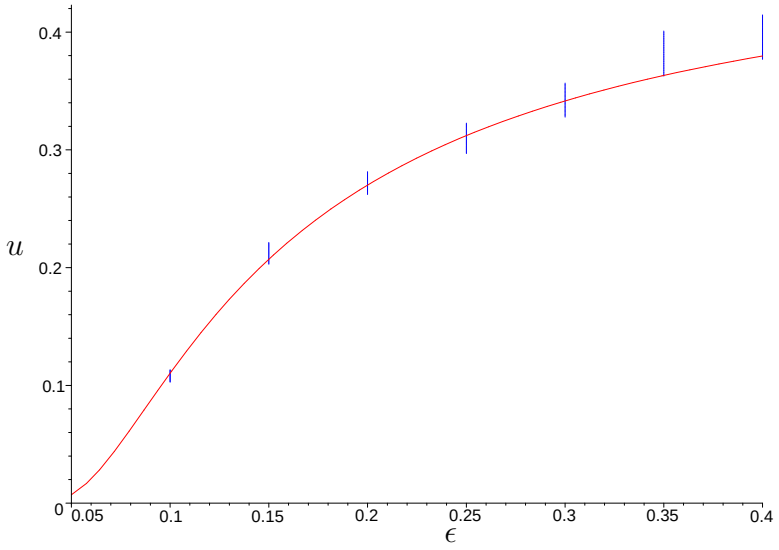

Figure 4.12: SP-approximation to $u(x, y)$ as in (4.41) and the simulation results from Table 4.4

is then used in the same way as before to simulate the probability that the particle reaches the boundary.

Table 4.4 shows the resulting confidence intervals for different values of ϵ.

ϵ	I_u	I_{T_1}
0.40	(0.1020 , 0.1116)	(3.6859 , 3.8102)
0.35	(0.2003 , 0.2129)	(3.0611 , 3.1798)
0.30	(0.2635 , 0.2825)	(2.7129 , 2.8075)
0.25	(0.2921 , 0.3240)	(2.4172 , 2.5649)
0.20	(0.3240 , 0.3531)	(2.1706 , 2.2946)
0.15	(0.3559 , 0.3928)	(1.9492 , 2.0879)
0.10	(0.3865 , 0.4201)	(1.7967 , 1.9145)

Table 4.4: Simulation results of the groundwater pollution model

Conclusion

The SP-approximations (4.41) and (4.44) are plotted in Figures 4.12 and 4.13, together with the simulation results. From the figures it is clear that as $\epsilon \to 0$, the probability that the particle reaches the pumps decreases. This indicates that in the absence of noise, the particle will simply follow the flow. The expected arrival time at the boundary increases as $\epsilon \to 0$, which is what we wanted it to do. The simulation results verify the SP-approximations, although they seem to deviate from the SP-approximation more as ϵ gets larger. This is not very strange, since the SP-approximation becomes less accurate as ϵ gets larger.

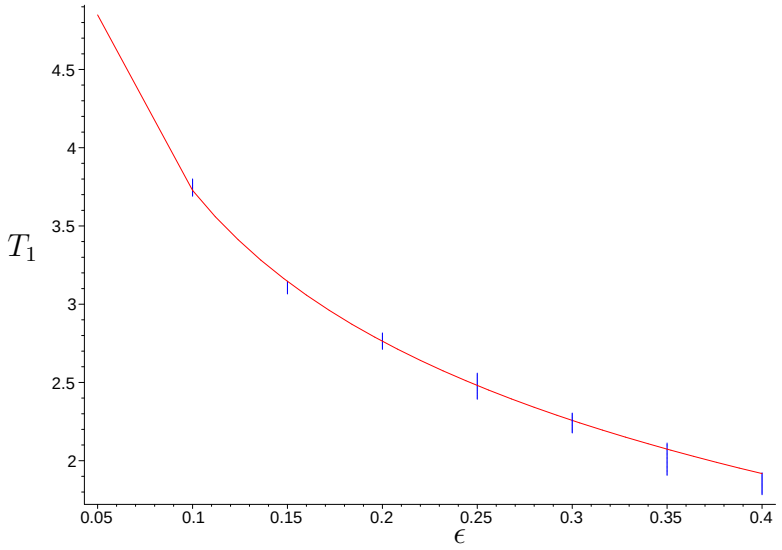

Figure 4.13: SP-approximation to $T_1(x, y)$ as in (4.44) and the simulation results from Table 4.4

Readers who want to know more about the applications of SDE in hydrology can read e.g. [32, 33] and the references therein.

Chapter 5

Final Remarks

The examples from Chapter 4 are just a few applications of SDE. There are many other examples to be found, see the references in the Introduction and Chapter 4. Before we look at what lies beyond this thesis, we have to make two comments. The first is about our way of simulating the expected exit time. We used the EM-method to simulate SDE's and then used this simulation to find an approximation to the expected exit time. As we said in Section 3.4, this approximation is influenced by a discretization error and a statistical error. We used confidence intervals to get an idea of the magnitude of the statistical error, but said nothing about the size of the discretization error. There is not much literature dealing with the discretization error and thus there is still some work to be done. Interested readers can perhaps use [54, 55] as a starting point.

The second comment is about the singular perturbations, which we used in Chapter 4 to approximate the differential equations. Singular perturbations are a popular way of approximating differential equations, although they can become rather complex. In Chapter 7 of [16] there are two examples which approximate the stationary distribution $p^{(s)}(x)$ by a WKB-expansion

$$p^{(s)}(x) \approx \omega(x)e^{-\Psi(x)}.$$

The coefficients $\omega(x)$ and $\Psi(x)$ have to be found using numerical integration, seriously complicating the calculations. In these situations, it is easier to approximate the differential equations using numerical techniques. This approach is used in [51, 52].

Further reading

An interesting topic, of which we have only touched the surface, is the numerical analysis of SDE. Our work was focused on introducing the Euler-Maruyama method (3.4) [43], but there are other numerical methods for approximating SDE's. There is for instance the stochastic theta method (also called semi-implicit Euler method), the Millstein scheme and Taylor schemes. For a complete treatment of these (and many other) schemes, we refer the reader to [42]. The main property of these methods that is usually studied is their convergence behavior, but some also look at stability [45, 46]. There are also stochastic versions of Runge Kuta schemes. More about these can be found in [48, 49, 53]. An overview of recent literature is given in [50].

In Chapter 2, when dealing with the geometric Brownian motion, we already mentioned the world of financial mathematics. Besides SDE's, this area of research also uses partial differential equations and numerical analysis and could therefore be worth looking at for people who read this document. A nice introduction can be found in e.g. [12, 13, 14].

Finally, I should mention the existence of the Maple package 'Stochastic' [60]. The package includes routines useful for finding explicit solutions of SDE, routines for finding or constructing numerical schemes and many others. The symbolic power of Maple combined with the computing and graphical possibilities of MATLAB are ideal for dealing with SDE (see e.g. [47]).

Appendix A

Program Listings

A.1 Brownian motion (unvectorized)

```
%BrownU.m
%Plots 3 sample paths of Brownian motion
%See section 3.1 .

clear;

randn('state',10000);                    %set the state of the randn

%%%%first sim%%%%

dt=1/500;
N=500;

dW=zeros(1,N);                           %precallocate arrays...
W=zeros(1,N);                            %for efficiency

dW(1)=sqrt(dt)*randn(1,1);               %first approximation...
W(1)=0+dW(1);                            %because W(0) is not allowed
for j=2:N
  dW(j)=sqrt(dt)*randn(1,1);
  W(j)=W(j-1)+dW(j);
end

plot([0:dt:dt*N],[0,W],'r-'), hold on    %plot first sim

%%%%second sim%%%%

dW=zeros(1,N);                           %precallocate arrays...
W=zeros(1,N);                            %for efficiency

dW(1)=sqrt(dt)*randn(1,1);               %first approximation...
```

```
W(1)=0+dW(1);                          %because W(0) is not allowed
for j=2:N
  dW(j)=sqrt(dt)*randn(1,1);
  W(j)=W(j-1)+dW(j);
end

plot([0:dt:dt*N],[0,W],'b-'), hold on   %plot second sim

%%%%third sim%%%%

dW=zeros(1,N);                         %precallocate arrays...
W=zeros(1,N);                          %for efficiency

dW(1)=sqrt(dt)*randn(1,1);             %first approximation...
W(1)=0+dW(1);                          %because W(0) is not allowed
for j=2:N
  dW(j)=sqrt(dt)*randn(1,1);
  W(j)=W(j-1)+dW(j);
end

plot([0:dt:dt*N],[0,W],'y-'), hold off  %plot third sim

xlabel('t','Fontsize',16)
ylabel('W(t)','Fontsize',16,'Rotation',0)
```

A.2 Brownian motion (vectorized)

```
%BrownV.m
%Plots 3 sample paths of Brownian motion using vectorized commands.
%See section 3.1 .

clear

randn('state',10000);                    %set the state of the randn
dt=1/500;
N=500;

dW=sqrt(dt)*randn(3,N);                  %The Brownian increments
W=cumsum(dW,2);                          %The Brownian paths

plot([0:dt:dt*N],[0,W(1,:)],'r-'), hold on      %plot the result
plot([0:dt:dt*N],[0,W(2,:)],'b-'), hold on      %plot the result
plot([0:dt:dt*N],[0,W(3,:)],'y-'), hold off     %plot the result
xlabel('t','Fontsize',16)
ylabel('W(t)','Fontsize',16,'Rotation',0)
```

A.3 Function along a Brownian path

```
%BrownF.m
%plots the function u(t)=exp(3*t+2*W(t)) along a sample path
%  of Brownian motion
%See section 3.1 .

clear;
randn('state',10000);
dt=1/500;                                %stepsize
N=500;                                   %number of steps
t=[dt:dt:1];

dW=sqrt(dt)*randn(1,N);                  %The Brownian increments
W=cumsum(dW);                            %The Brownian path
U=exp(3*t + 2*W);                        %Function along the Brownian path

plot([0,t],[1,U],'r-');
xlabel('t','Fontsize',16)
ylabel('U(t)','Fontsize',16,'Rotation',0,'HorizontalAlignment','right')
legend('The function u(t) along a path of Brownian motion',2)
```

A.4 Euler-Maruyama method

```
%GBMEM.m
%Simulates one sample path of the SDE dX(t)=lambda*X(t)*dt+mu*X(t)*dW(t)
%See section 3.2 .

clear
randn('state',10000);
lambda=5;mu=2;Xzero=1;
dt=1/500;N=500;

dW=sqrt(dt)*randn(1,N);                          %Brownian increments
W=cumsum(dW);                                    %Sample paths
t=[dt:dt:dt*N];
Xtrue=Xzero*exp((lambda-1/2*mu^2)*t+mu*W);       %true solution

Xem=zeros(1,N);
Xtemp=Xzero;

for j=1:N                                        %start simulation
  Xtemp=Xtemp+lambda*Xtemp*dt+mu*Xtemp*dW(j);
  Xem(j)=Xtemp;
end

plot([0:dt:dt*N],[Xzero,Xtrue],'b-'), hold on    %plot results
plot([0:dt:dt*N],[Xzero,Xem],'r-'), hold off
xlabel('t','FontSize',16);
ylabel('X','FontSize',16,...
       'Rotation',0,'HorizontalAlignment','right');
```

A.5 Monte Carlo approximation using EM

```
%GBMMC.m
%Performs a Monte Carlo simulation of the SDE
%     dX(t)=lambda*X(t)*dt + mu*X(t)*dW(t)
%using different values for dt. The error at time t=0.5 is computed for
%each stepsize. See section 3.3 .

clear

randn('state',10000);
lambda=5;mu=2;Xzero=1;                          %some parameters
M=10000;                                        %10000 sample paths

ErrorEM=zeros(1,5);                             %will hold errors for
dt=1/(100*2^4);                                 %stepsize for true solution
N=50*2^4;                                       %number of steps
dW=sqrt(dt)*randn(M,N);                         %Brownian increments
W=cumsum(dW,2);                                 %Brownian paths
Xtrue=Xzero*exp((lambda-mu*mu/2)*dt*N+mu*W(:,end));   %True solution at t=0.5

for i=0:4
   R=2^(4-i);
   Dt=R*dt;                                     %increase stepsize
   L=N/R;                                       %decrease steps

   Xtemp=Xzero;                                 %EM-approximation
   for j=1:L
      Winc=sum(dW(:,R*(j-1)+1:R*j),2);
      Xtemp=Xtemp+lambda*Xtemp*Dt+mu*Xtemp.*Winc;
   end
   ErrorEM(i+1)=mean(abs(Xtrue-Xtemp));         %find error at t=0.5
end
```

A.6 Expected exit time

```
%GBM.m
%Find confidence intervals for the expected exit time of the SDE
%  dX(t)=lambda*X(t)*dt + mu*X(t)*dW(t)
%See section 3.4 .

clear;
randn('state',10000);

lambda=5;mu=2;                          %some parameters
a=1;b=9;

M=50;                                   %size of a MC-sim
L=20;                                   %number of MC-sims
Xzero=2;
dt=0.00025;                             %stepsize
CIU=zeros(2,7);                         %confidence interval for U
CIT=zeros(2,7);                         %confidence interval for T


while Xzero<9                           %the different starting points
  l=0;
  total=zeros(1,M);                     %exit time for each sample path
  Tsim=zeros(1,L);                      %mean exit time for each MC-sim
  Usim=zeros(1,L);                      %exit prob. for each MC-sim

  while l<L                             %loop for MC-sims
    sims=0;
    Q=0;
    while sims<M                        %loop for sample paths of a MC-sim
      flag=0;
      Xtemp=Xzero;
      count=0;
      while flag==0                     %loop for a single sample path
        Xtemp=Xtemp+lambda*Xtemp*dt+...  %EM-approximation
              mu*Xtemp*sqrt(dt)*randn(1,1);
        count=count+1;
        if Xtemp<=a                     %exit through left side of the domain
          Q=Q+1;
          flag=1;
        end
        if Xtemp>=b                     %exit through right side of the domain
          sims=sims+1;
          total(sims)=dt*count;
          flag=1;
        end
```

```
        end
      end

      Tsim(l+1)=mean(total);                %mean exit time for this MC-sim
      Usim(l+1)=sims/(sims+Q);              %exit prob. for this MC_sim
      l=l+1;                                %next MC-sim
    end

    avg=mean(Usim);                         %construct confidence interval for Usim
    delta=1.73*sqrt(var(Usim)/L);
    CIU(:,Xzero-1)=[avg-delta,avg+delta]';

    avg=mean(Tsim);                         %construct confidence interval for Tsim
    delta=1.73*sqrt(var(Tsim)/L);
    CIT(:,Xzero-1)=[avg-delta,avg+delta]';
    Xzero=Xzero+1;                          %next starting point
end
```

A.7 Stochastic Population Dynamics

```
%popdynMC.m
%performs L Monte Carlo simulations of size M to find a confidence
% interval for the expected exit time of the SDE from section 4.1 .

clear;
b=30;d=29;                                  %some parameters
alpha=b-d;beta=b+d;
Xzero=1;

randn('state',10000);
L=20;                                       %L MC-sims
M=50;                                       %M sample paths
K=10;                                       %value of parameter K
dt=0.01;                                    %stepsize
CIT=zeros(2,8);                             %will hold the confidence
                                            % intervals
while K<60                                  %loop for K
  Tsim=zeros(1,L);
  l=0;

  while l<L;                                %L MC-sims
    sims=0;
    total=zeros(1,M);

    while sims<M,                           %each MC sim uses M paths
      flag=0;
      Xtemp=Xzero;
      count=1;

      while flag==0,                        %simulate a sample path

        Xtemp=Xtemp+alpha*(1-Xtemp)*Xtemp*dt ...
        +sqrt((beta*Xtemp+alpha*Xtemp*Xtemp)/K)*sqrt(dt)*randn(1,1);

        if Xtemp<=0,                        %exit?
          sims=sims+1;
          total(sims)=dt*count;
          flag=1;
        end
        count=count+1;
      end
    end
    Tsim(l+1)=mean(total);
    l=l+1                                   %next MC sim
  end
```

```
    avg=mean(Tsim);
    delta=1.73*sqrt(var(Tsim)/L);
    CIT(:,K/5-1)=[avg-delta,avg+delta]';          %store confidence interval
    K=K+5                                        %next K
end
```

A.8 Firing of a neuron

```
%neuronMC.m
%
%Performs L Monte Carlo simulations of size M for the system in section 4.2,
% to find a confidence interval of the expected exit time.

clear
nu=.1; alpha=.9; beta=-.9;VE=50; VI=-7;S=2;          %some parameters
Xzero=0;                                             %starting value

randn('state',10000);

K=10;                                                %first value of K
M=50;                                                %number of MC-sims
L=20;                                                %number of sample paths

CIT=zeros(2,7);                                      %will contain the confidence
                                                     % intervals
total=zeros(1,M);                                    %will contain exit time of
                                                     % each sample path
Tsim=zeros(1,L);                                     %will contain mean exit times
                                                     % of each MC-sim
dt=0.0015;                                           %the stepsize

while K<65                                           %Loop for K
  epsilon=1/sqrt(K);
  l=0;
  while l<L                                          %L MC-sims
    sims=0;

    while sims<M                                     %each MC-sim uses M sample paths
      Xtemp=Xzero;
      flag=0;
      count=0;

      while flag==0                                  %simulate 1 sample path
        bx=-Xtemp/nu + epsilon^2*alpha*(VE-Xtemp)+epsilon^2*beta*(Xtemp-VI);
        sigmax=epsilon*sqrt(alpha^2*(VE-Xtemp)^2+beta^2*(Xtemp-VI)^2);
        Xtemp=Xtemp+bx*dt + sigmax*sqrt(dt)*randn(1,1);

        count=count+1;

        if Xtemp>=S                                  %Fire?
          flag=1;
          sims=sims+1;
          total(sims)=dt*count;
```

```
            end
          end
        end
        Tsim(l+1)=mean(total);                    %mean exit time for this MC-sim
        l=l+1                                     %next MC-sim
    end
    avg=mean(Tsim);
    delta=1.73*sqrt(var(Tsim)/L);
    CIT(:,K/5-1)=[avg-delta,avg+delta]';          %confidence interval for current K
    K=K+5                                         %next K
end
```

A.9 Particle movement

```
%PM.m
%Performs L Monte Carlo simulations of size M for the system in section 4.3,
% to find a confidence interval of the expected exit time.

clear
epsilon=0.05;                                   %some parameters
v=1;Xzero=-1.5;Yzero=0.12;

randn('state',10000);

dt=0.05;                                        %the stepsize
M=50;                                           %M sample paths for

                                                % a MC-simulation
L=20;                                           %L MC-simulations

CIU=zeros(2,6);                                 %confidence intervals
                                                % for u
CIT=zeros(2,6);                                 %confidence intervals
                                                % for T1

while epsilon<0.35;                             %loop for epsilon
  DT=epsilon^2/2;DL=epsilon^2/2;
  l=0;
  Usim=zeros(1,L);
  Tsim=zeros(1,L);
  while l<L                                     %L MC-sims
    sims=0;
    Q=0;
    total=zeros(1,M);
    while sims<M,                               %M paths per MC-sim
      flag=0;
      Xtemp=Xzero;
      Ytemp=Yzero;
      count=1;
      while flag==0,                            %simulate one sample path
        Xtemp=Xtemp+v*dt+sqrt(2*DL*dt)*randn(1,1);
        Ytemp=Ytemp+sqrt(2*DT*dt)*randn(1,1);

        if Ytemp<=0 & Xtemp<=0,                 %exit through horizontal axis
          sims=sims+1;
          total(sims)=dt*count;
         flag=1;
        end
        if Xtemp>0,                             %exit through vertical axis
```

```
            flag=1;
            Q=Q+1;
          end
          count=count+1;
        end
      end
    Usim(l+1)=sims/(sims+Q);                    %exit probability
    Tsim(l+1)=mean(total);                      %exit time for current MC-sim
    l=l+1                                       %next MC-sim
  end
  avg=mean(Usim);
  delta=1.73*sqrt(var(Usim)/L);
  CIU(:,round(epsilon/0.05))=[avg-delta,avg+delta]';%conf. int. for U
  avg=mean(Tsim);
  delta=1.73*sqrt(var(Tsim)/L);
  CIT(:,round(epsilon/0.05))=[avg-delta,avg+delta]';%conf. int. for T
  epsilon=epsilon+0.05
end
```

A.10 Groundwater Pollution

```
%GP.m
%
%Performs L Monte Carlo simulations of size M for the system in section 4.4,
% to find a confidence interval of the expected arrival time at the
% boundary {Y=-R}.

clear;
randn('state',10000);

Xzero=4;Yzero=0.2;                                  %starting point
R=2;                                                %lower boundary

dt=0.01;                                            %stepsize
M=50;                                               %M sample paths
L=20;                                               %L MC-sims
totaal=zeros(1,M);                                  %will hold exit times
                                                    % of sample paths
Usim=zeros(1,L);                                    %will hold exit prob.
                                                    % for each MC-sim
Tsim=zeros(1,L);                                    %will hold exit times
                                                    % for each MC-sim
CIU=zeros(2,6);                                     %conf. int. for U
CIT=zeros(2,6);                                     %conf. int. for T_1

while epsilon<0.4                                   %loop for epsilon
  al=epsilon^2;at=epsilon^2;
  l=0;
  while l<L                                         %L MC-sims
    Q=0;
    sims=0;
    while sims<M,                                   %M paths for each MC-sim

      flag=0;
      Xtemp=Xzero;
      Ytemp=Yzero;
      count=1;
      while flag==0,                                %loop for one sample path
        hulp=sqrt(Xtemp^2+Ytemp^2);

        Xtemp2=Xtemp;
                                                    %simulate x
        Xtemp=Xtemp+...
          (-Xtemp+at*Xtemp/hulp+(al-at)*Xtemp/hulp-(al-at)*Xtemp^3/(hulp^3)+...
          (al-at)*Xtemp*Ytemp^2/(hulp^3))*dt...
          -sqrt(2*al/hulp)*Xtemp*sqrt(dt)*randn(1,1)+...
```

```
        sqrt(2*at/hulp)*Ytemp*sqrt(dt)*randn(1,1);

                                                    %simulate y
      Ytemp=Ytemp+...
        (Ytemp+at*Ytemp/hulp+(al-at)*Ytemp/hulp-(al-at)*Ytemp^3/(hulp^3)+...
        (al-at)*Ytemp*Xtemp2^2/(hulp^3))*dt+...
        sqrt(2*al/hulp)*Ytemp*sqrt(dt)*randn(1,1)+...
        sqrt(2*at/hulp)*Xtemp2*sqrt(dt)*randn(1,1);

      if Ytemp<=-R,                                 %exit through Y=-R
        sims=sims+1;
        totaal(sims)=dt*count;
        flag=1;
      end
      if Ytemp>=10,                                 %exit through 'infinity'
        flag=1;
        Q=Q+1;
      end
      count=count+1;
    end
  end
  Tsim(l+1)=sum(totaal)/sims;                       %compute mean exit time
  Usim(l+1)=sims/(sims+Q);                          %compute exit prob.
  l=l+1
 end
 avg=mean(Usim);
 delta=1.73*sqrt(var(Usim)/L);
 CIU(:,round(epsilon/0.05)-1)=[avg-delta,avg+delta]';  %conf. int. for u
 avg=mean(Tsim);
 delta=1.73*sqrt(var(Tsim)/L);
 CIT(:,round(epsilon/0.05)-1)=[avg-delta,avg+delta]';  %conf. int. for T_1
 epsilon=epsilon+0.05
end
```

Bibliography

Probability theory

[1] P. Brémaud
An introduction to probabilistic modeling
Springer-Verlag, 1987.

[2] G. Grimmet & D. Welsh
Probability, an introduction
Oxford University Press, 1986.

[3] L. Breiman
Probability
Addison-Wesley, Reading, 1968.

[4] V. K. Rohatgi
An introduction to probability theory and mathematical statistics
Wiley & sons, New York, 1976.

Theory of Stochastic Differential equations

[5] B. Øksendahl
Stochastic differential equations. (An introduction with applications)
Springer 1998, fifth edition.

[6] Stochastic differential equations
O. van Gaans & J. van Neerven
University of Leiden, Math. dep. technical report no. MI-2001-18, 2001.

[7] Z. Schuss
Theory and applications of stochastic differential equations
Wiley series in probability and mathematical statistics
Wiley & sons 1980.

[8] S. K. Srinivasan & R. Vasudevan
Introduction to random differential equations and their applications
Elsevier, New York, 1971.

[9] T. T. Soong
Random differential equations in science and engineering
Academic Press, New York, 1973.

[10] L. Arnold
Stochastic differential equations : theory and applications
Wiley & sons, 1974.

Mathematical finance

[11] J. Goodman, K. Moon, A. Szepessy, R. Tempone, G. Zouraris
Stochastic and partial differential equations with adapted numerics
Lecture notes.

[12] P. Wilmott
Paul Wilmott introduces quantitative finance
Wiley & sons, 2001.

[13] A. G. Malliaris & W. A. Brooks
Stochastic methods in economics and finance
North-Holland, Amsterdam, 1982.

[14] P. Wilmott, S. Howison & Susan Howson
The mathematics of financial derivatives
Cambridge University Press, 1995

SDE and their connection to PDE

[15] C.W. Gardiner
Handbook of stochastic methods (for physics, chemistry and the natural sciences)
Springer 1985, 2^{nd} edition.

[16] J. Grasman, O.A. van Herwaarden
Asymptotic methods for the Fokker-Planck equation and the exit problem in applications
Springer-Verlag 1999.

[17] C. Knessl, B. J. Matkowski, Z. Schuss, C. Tier
An asymptotic theory of large deviations for Markov jump processes
SIAM J. Appl. Math, vol. 45, p. 1006-1028, 1985.

[18] R. M. Nisbet, W. S. Gurney
Modelling fluctuating populations
Wiley & sons, 1982.

[19] N. S. Goel, N. Richter-Dyn
Stochastic models in biology
Academic Press, 1974.

[20] L. M. Ricciardi
Diffusion processes and related topics in biology
Lecture Notes in Biomathematics 14
Springer-Verlag, Berlin, 1977.

[21] T. C. Gard
Introduction to stochastic differential equations
Marcel Dekker Inc., New York, 1988.

[22] H. Roozen
Equilibrium and extinction in stochastic populations dynamics
Bull. Math. Biol. 49, p. 671-696, 1987.

[23] C. Wissel & S. Stöcker
Extinction of populations by random influences
Theor. pop. biol. 39, p. 315-328, 1991.

[24] M. Turelli
Random environments and stochastic calculus
Theor. Pop. Biol. 12, p. 140-178, 1977

[25] L. M. Ricciardi
Stochastic population theory : diffusion processes
In ”Mathematical Ecology ; an introduction”, p. 191-238
Springer-Verlag, 1986.

[26] J. Grasman & R. HilleRisLambers
On local extinction in a metapopulation
Ecol. Mod. 103, p. 71-80, 1997.

[27] M. Mangel
Barrier transition driven by fluctuations, with applications to ecology and evolution
Theor. Pop. Biol. 45, p. 16-40, 1994.

Applications of SDE in neurology

[28] G. Kallianpur, R. Wolpert
Weak convergence of stochastic neural models
Lecture Notes in Biomathematics 70 (called 'Stochastic Methods in Biology')
Springer Verlag 1987.

[29] A. V. Holden
Models of the stochastic activity of neurons
Lecture Notes in Biomathematics 12
Springer-Verlag, Berlin, 1976.

[30] P. Lánský & V. Lánská
Diffusion approximation of the neuronal model with synaptic reversal potentials
Biol. Cyb. 56, p. 19-26, 1987.

Applications of SDE in hydrology

[31] O.A. van Herwaarden
Spread of pollution by dispersive groundwater flow
SIAM L. Appl. Math. 54, p. 26-41.

[32] H. Beckmann, H. Madsen, N. K. Poulsen, M. K. Nielsen
Grey box modelling of first flush and incoming wastewater at a wastewater treatment plant
Environmetrics 11, p. 1-12, 2000.

[33] J. Bear & A. Verruijt
Modeling groundwater flow and pollution
Reidel, Dordrecht, the Netherlands, p. 159-170, 1987.

Applications of SDE in physics

[34] S. Chandrasekhar
Stochastic problems in physics and astronomy
Rev. Mod. Phys. 15, p. 1-89, 1943.

[35] G. Papanicolau & J. B. Keller
SDE with applications to harmonic oscillation and wave propagation in random media
SIAM J. Appl. Math. 21, p. 287-305, 1971.

[36] G. Papanicolau
Wave propagation in a one-dimensional random medium
SIAM J. Appl. Math. 21, p. 13-18, 1971

[37] M. C. Torrent & M. San Miguel
Stochastic dynamics characterization of delayed laser threshold instability with swept control parameter
Phys. Rev. A 38, p. 245-251, 1988.

Applications of SDE in engineering

[38] G. Broggi, A. Colombo, L. A. Lugiate & P. Mandel
Influence of white noise on delayed bifurcations
Phys. Rev. A 30, p. 3635-3637, 1986.

[39] I. Kamarianakis & N. Frangos
Deterministic and stochastic differential equation modelling for electrical networks
HERCMA conference, Athens University of Economics and and Business, 2001.

[40] G. Unal, H. Krim & A. Yezzi
Stochastic differential equations and geometric flows
Accepted for publication in IEEE Transactions om Image Processing, 2002.

[41] A. Mehrotra & A. L. Sangiovanni-Vincintelli
Noise analysis of non-autonomous radio frequency circuits
International conference on Computer-Aided design, 1999.

Numerical Analysis of SDE

[42] P. Kloeden & E. Platen
Numerical solution of stochastic differential equations
Springer-Verlag, Berlin 1992

[43] G. Maruyama
Continuous Markov processes and stochastic equations
Rend. Circolo Math. Palermo 4, p. 48-90 (1955)

[44] D. J. Higham
An algorithmic introduction to numerical simulation of SDE
SIAM Review, Education Section, Vol 43, 2001.

[45] D. J. Higham
Mean-square and asymptotic stability of numerical methods for SDE
University of Strathclyde Mathematics report 39, 1998.

[46] S. Cyganowski & P. Kloeden
Stochastic stability examined through Maple
in 15th IMACS World Congress on Scientific Computation,
Modelling and Applied Mathematics, Vol. 1, Wissenschaft and Technik Verlag,
Berlin, 1997, p. 437-442.

[47] D. J. Higham & P. Kloeden
MAPLE and MATLAB for stochastic differential equations in finance
University of Strathclyde Mathematics report 3, 2001.

[48] K. Burrage & P. M. Burrage
General order conditions for stochastic Runge-Kutta methods for both commuting and non-commuting SDE systems
1997.

[49] K. Burrage & P. M. Burrage
High strong order Runge-Kutta method for ordinary SDE's
1996.

[50] E. Platen
An introduction to numerical methods for stochastic differential equation
Acta Numerica 8, p. 197-246, 1999.

[51] B. F. Spencer Jr. & L. A. Bergman
Numerical solution of the Fokker-Planck equation for first passage probabilities
Comp. Stoch. Meth., p. 359-370, 1991.

[52] M. P. Zorzano & L. Vazquez
Numerical solution for Fokker-Planck equation in accelerators
Particle Accelerators Conference, Vancouver, 1997.

[53] R. L. Honeycut
Stochastic Runge-Kutta algorithms. I. White noise
Phys. Rev. A 45, p. 600-603, 1992.

[54] E. Gobet
Weak approximation of killed diffusion using Euler schemes
Stochastic Processes and their Applications, Vol.87, p. 167-197, 2000.

[55] E. Gobet & S. Menozzi.
Discrete approximations of killed It processes.
Internal Report # 503 of CMAP, Ecole Polytechnique, 2003.

Singular Perturbations

[56] A. H. Nayfeh
Introduction to perturbation techniques
Wiley & Sons, New York, 1981.

[57] W. Wasow
Asymptotic expansion for ordinary differential equations
Dover Publications Inc., New York, 1965

[58] W. Eckhaus
Asymptotic analysis of singular perturbations
North-Holland Publishing Companym, Amsterdam, 1971.

Other

[59] Santiago Ramón y Cajal
Textura del Sistema Nervioso del Hombre y los Vertebrados
N. Moya, 1904, Madrid.

[60] S. Cyganowski
Maple package 'Stochastic'
On-line at http://www.math.uni-frankfurt.de/∼ numerik/maplestoch/

[61] T. M. Apostel
Calculus
Volume 2, Wiley & Sons, 1969.

[62] C. Lanczos
Linear differential operators
D. van Nostrand Company Ltd., London, 1961.

[63] G. F. Roach
Green's functions : introductory theory with applications
Van Nostrand Reinhold Company, London, 1970.

[64] J. M. Hammersley & D. C. Handscombe
Monte Carlo Methods
Wiley & sons, New York, 1964

[65] N. Bouleau & D. Lêpingle
Numerical methods for stochastic processes
Wiley & sons, New York, 1994

[66] R. N. Bhattacharga & R. Ranga Rao
Normal approximation and asymptotic expansion
Wiley & sons, New York, 1976.