TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

DIPLOMARBEIT

Various Optimization Criteria for a Two-Dimensional Stochastic Control Problem

Ausgeführt am Institut für

Wirtschaftsmathematik

der Technischen Universität Wien

unter der Anleitung von

Ao. Univ. Prof. Dipl.-Ing. Dr. techn. Peter Grandits

durch

Magda-Denise Mirescu, B.Sc.

Wien, 2014

(Unterschrift)

Încheiere

Stinsă liniştirea noastră (şi aleasă),
Isarlâk încinsă, Isarlâk mireasă!
Dovediţii, mie, doisprezece turci
Între poleite pietre să mi-i culci:

Inima - raiaua, osul feţii spân,
Ţeasta, nervii torşi în barbă de stăpân,
Clatină-i la Ciprul Negru, în albeaţă
De sonoră vale într-o dimineaţă!

Vis al Dreptei Simple! Poate, geometria
Săbiilor trase la Alexandria,
Libere, sub ochiul de senin oţel,
În neclătinatul idol El Gahel.

Inegală creastă, suliţată cegă,
Lame limpezi duse-n ţara lui norvegă!
Răcoriţi ca scuţii zonele de aer,
Răsfiraţi cetatea norilor în caier,

Eu, sub piatra turcă, luat de Isarlâk,
La o albă apă intru - bâldâbâc.
Fie să-mi clipească vecinice, abstracte,
Din culoarea minţii, ca din prea vechi acte.
Eptagon cu vârfuri stelelor la fel.
Şapte semne, puse ciclic:

Dr. Ion Barbu, Romanian poet and mathematician

Statutory Declaration

I hereby declare on my honor to have written the present thesis independently, solely with the support of the listed literature references and to have submitted its content to no other examination authority.

(place, date)

(signature)

Abstract

Stochastic control theory deals with the behaviour of dynamical systems that are subject to randomness. This thesis presents and meticulously analyzes a two-dimensional model with two independent Brownian motions assumed to describe the conduct of two distinct companies in a market as depicted in Aldous (2000) and McKean and Shepp (2006). To influence their trajectories one can choose between three different policies: one can either push the bottom or the top company by adding the unit of drift at one's disposal to one of them or one can divide that unit of drift randomly amongst them.

It is shown that the optimal control strategy depends on the optimization criterion which can be to either maximize the probability that both companies survive or to maximize the expected number of companies that remain solvent. To that purpose both numerical approximations and Monte Carlo simulations are conducted.

Keywords: *stochastic control theory, dynamic optimization, Brownian motion, partial differential equations, stochastic differential equations, finite difference method, Monte Carlo methods*

Acknowledgements

Acknowledgements (Part 1)

> What is a teacher? I'll tell you: it isn't someone who teaches something, but someone who inspires the student to give of her best in order to discover what she already knows.
>
> **Paulo Coelho - The Witch of Portobello**

From a professional point of view, I wish to thank primarily Ao. Univ. Prof. Dipl.-Ing. Dr. techn. Peter Grandits for presenting me with an interesting topic for my master's thesis and for his guidance throughout the entire process having this paper as a result.

Secondly, but of no less importance, I want to give special thanks to all the professors whose lectures I attended and in whose seminars I took part. I learned a lot from each and every one of you.

Thank you all for helping me become the mathematician that I am today!

Acknowledgements (Part 2)

> The three great essentials to achieve anything worthwhile are, first, hard work; second, stick-to-itiveness and third, common sense.
>
> **Thomas A. Edison**

It has been a long journey, during which a lot has happened and changed in my life. However, some people always stood by my side and offered me their unconditional support whenever I needed it.

I want to thank these people, who - I am sure - know very well who I am referring to, for celebrating with me at my best and for getting me out of the darkest holes at my worst. I appreciate every second of their time that they chose to invest in me and bow to them with perpetual gratitude.

Thank you!

Contents

List of Figures

List of Tables

Chapter 1

Introduction

> "The sciences do not try to explain, they hardly even try to interpret, they mainly make models. By a model is meant a mathematical construct which, with the addition of certain verbal interpretations, describes observed phenomena. The justification of such a mathematical construct is solely and precisely that it is expected to work - that is, correctly describe phenomena from a reasonably wide area. Furthermore, it must satisfy certain aesthetic criteria - that is, in relation to how much it describes, it must be rather simple."
>
> **John von Neumann - Method in the Physical Sciences, 1955**

Applied mathematics is a special branch in the vast area of mathematical sciences that mainly lays emphasis on the formulation, study, solving and interpretation of *mathematical models*. As such, it can be viewed as a link between abstract mathematics and a multitude of fields like engineering, medicine, business, finance etc.

Control theory analyzes the behaviour of dynamical systems reproduced by mathematical models and submitted to control. When additionally subject to uncertainty, the usual control theory receives a so-called stochastic component, hence the derivation of the more specific discipline called *stochastic control theory*.

This thesis deals with a stochastic control model that represents the borderline between politics and economics as it evaluates two opposing political, economic and social systems: *socialism* and *capitalism*. The difference between them is modelled mathematically through two distinct optimization criteria, upon which the optimal control policy depends.

The current paper is structured as follows.

Chapter 1 provides a succinct introduction into stochastic control theory and modelling and offers insight into the mathematical model that lies at the core of this thesis.

Chapter 2 presents fundamental notions in stochastic analysis that are later used in other chapters. The reader is thus introduced to the meaning and properties of a stochastic process and to those of a special, widely-used example of such a construct which is the Brownian motion. Special attention is dedicated to probably the most important concept of stochastic analysis: the Itô formula. Martingales and stochastic differential equations are also briefly

mentioned.

Chapter 3 introduces the reader to stochastic optimization by offering a skeleton of the basic characteristics that define a stochastic optimization problem. These basic notions are then exemplified on diffusion processes for both finite and infinite time horizons. The dynamic programming principle (DPP) is stated and used to formally derive this chapter's highlight, the Hamilton-Jacobi-Bellman (HJB) equation. Last but not least, a tool for establishing whether a solution to the HJB-equation is a value function is offered: the verification theorem.

In Chapter 4 examples for the application of stochastic control theory in financial and actuarial mathematics are given. It debuts classically with the already world-renowned Merton model for portfolio allocation and continues with a lesser known model used to minimize ruin probabilities in insurance business.

Chapter 5 portrays methods for numerically solving partial differential equations (PDEs) as well as for simulating stochastic differential equations (SDEs). Worth mentioning in this context is the finite difference method for approximating PDEs, whose resulting system of linear equations is generally solved through iterative mechanisms such as Jacobi Iteration or Successive over Relaxation, but also Monte Carlo simulation for generating sample paths as given by a SDE through the Euler Scheme.

This thesis' underlying model is thoroughly described in Chapter 6. First, a theoretical overview of a more general model, as originally formulated by David Aldous, is delivered. Then, we restrict ourselves to a specific, two-dimensional model as formulated by Henry McKean and Lawrence Shepp that, depending on its optimization criterion, may determine which of the two systems, socialism or communism, is more advantageous.

The next chapter (Chapter 7) contains proposed solutions for the model presented in the previous chapter. It offers a detailed explanation of how the methods presented in Chapter 5 are applied on the model depicted in Chapter 6. It also analyzes each possible strategy and illustrates their immediate consequences.

After much labour, the time has come to reap the fruits in Chapter 8, that contains the results of the numerical approximations and the Monte Carlo simulations in some cases as tables but mostly as graphics. Interpretations of the tables and figures are additionally given.

Chapter 9 extends the two-dimensional model to a three-dimensional one and briefly presents its results.

The last chapter (Chapter 10) draws the obvious conclusions from the results produced in the previous chapter and formally ends this thesis.

Appendices A, B and C contain the source codes for everything programmed for this paper including the numerical approximations and Monte Carlo simulations.

Chapter 2

Basics of Stochastic Analysis

2.1 Stochastic Processes

A **stochastic process**, or more commonly referred to as a **random process**, is a term frequently used in probability theory, where it describes a collection of random variables representing the evolution of a system of aleatoric values over time.

As opposed to its counterpart, the **deterministic process**, which produces the same output from a given initial condition, a stochastic process is characterized by indefiniteness in its future evolution. In other words, even if the starting point is known, there is more than one possibility the process can have as output, some more probable than others, implying the use of probability distributions.

Such processes are encountered for example in finance (in stock price fluctuations), in medicine (in EKGs, EEGs), in engineering (in audio and video signals) etc.

Definition 2.1.1 (Stochastic Process) [2, page 51]

Let $(\Omega, \mathcal{F}, \mathbb{P})$ *be a probability space,* $\mathbb{T}$ *an index set and* $(E, \mathcal{B})$ *a measurable set. An* $(E, \mathcal{B})$*-valued* **stochastic process** *on* $(\Omega, \mathcal{F}, \mathbb{P})$ *is a family* $(X_t)_{t\in\mathbb{T}}$ *of random variables* $X_t : (\Omega, \mathcal{F}) \to (E, \mathcal{B})$*, for* $t \in \mathbb{T}$*.*

In the above definition $(E, \mathcal{B})$ represents the state space. For a fixed value t, the random variable X_t depicts the state of the process at time t.

Moreover, the index set $\mathbb{T}$ stands for a time interval, that may either be finite $\mathbb{T} = [0, T]$, $0 < T < \infty$, or infinite $\mathbb{T} = [0, \infty)$. The time parameter t that varies in $\mathbb{T}$ is allowed to be either discrete or continuous.

Furthermore, the mapping $X(., \omega) : t \in \mathbb{T} \to X_t(w) \in E$ is called the trajectory of the process that corresponds to the event ω, for each $\omega \in \Omega$.

Definition 2.1.2 (Càd-làg Process) [1, page 1]

A stochastic **process** $(X_t)_{t\in\mathbb{T}}$ *is said to be* **càd-làg** *(resp. continuous) if for each* $\omega \in \Omega$*, the path* $X(\omega)$ *is right-continuous and admits a left-limit (resp. is continuous).*

As time elapses, one has more and more information since the indeterminacy on the events of Ω becomes less and less uncertain. In stochastic analysis this aspect is referred to as filtration.

Definition 2.1.3 (Filtration) [1, page 2]

A **filtration** *on* $(\Omega, \mathcal{F}, \mathbb{P})$ *is an increasing family* $\mathbb{F} = (\mathcal{F}_t)_{t\in\mathbb{T}}$ *of* σ*-fields of* $\mathcal{F} : \mathcal{F}_s \subset \mathcal{F}_t \subset \mathcal{F}$ *for all* $0 \leq s \leq t$ *in* $\mathbb{T}$.

Thus, $\mathcal{F}_t$ contains the information known at time t and $\mathcal{F}_t$ increases with the passing of time.

Hereafter, we assume a filtration $\mathbb{F} = (\mathcal{F}_t)_{t\in\mathbb{T}}$ on $(\Omega, \mathcal{F}, \mathbb{P})$.

Definition 2.1.4 (Adapted Process) [1, page 2]

A **process** $(X_t)_{t\in\mathbb{T}}$ *is* **adapted** *(with respect to* $\mathbb{F}$*) if for all* $t \in \mathbb{T}$*,* X_t *is* $\mathcal{F}_t$*-measurable.*

Otherwise put, $(X_t)_{t\in\mathbb{T}}$ is $\mathcal{F}_t$-adapted if the value of X_t at time t only relies upon the information contained in the stochastic process up to time t.

Definition 2.1.5 (Progressively Measurable Process) [1, page 2]

A process $(X_t)_{t\in\mathbb{T}}$ *is* **progressively measurable** *(with respect to* $\mathbb{F}$*) if for any* $t \in \mathbb{T}$*, the mapping* $(s, \omega) \to X_s(\omega)$ *is measurable on* $[0, t] \times \Omega$ *equipped with the product* σ*-field* $\mathcal{B}([0, t]) \otimes \mathcal{F}_t$.

In the next step, we want to know if the first arrival time of an event, denoted as $\tau(\omega)$ occurred before time t, given the information provided by the filtration $\mathcal{F}_t$. Thus, one introduces the notion of stopping time.

Definition 2.1.6 (Stopping Time) [1, page 3]

1. *A random variable* $\tau : \Omega \to [0, \infty)$*, i.e. a random time, is a* **stopping time** *(with respect to the filtration* $\mathbb{F} = (\mathcal{F}_t)_{t\in\mathbb{T}}$*) if for all* $t \in \mathbb{T}$

$$\{\tau \leq t\} := \{\omega \in \Omega : \tau(\omega) \leq t\} \in \mathcal{F}_t. \tag{2.1}$$

2. *A stopping time* τ *is predictable if there exists a sequence of stopping times* $(\tau_n)_{n\geq 1}$ *such that we have almost surely:*

   (a) $\lim_{n\to\infty} \tau_n = \tau$*;*

   (b) $\tau_n < \tau \quad \forall n \quad$ *on* $\quad \{\tau > 0\}$.

We say that $(\tau_n)_{n\geq 1}$ *announces* τ.

Assuming a stochastic process $(X_t)_{t\in\mathbb{T}}$ and a stopping time τ, we define the random variable X_τ on $\{\tau \in \mathbb{T}\}$ by

$$X_\tau(\omega) = X_{\tau(\omega)}(\omega).$$

The next proposition provides important insight into stopping times.

Proposition 2.1.1 [1, page 4]
Let X be a càd-làg, adapted process, and Γ an open subset of $\mathbb{R}^d$.

1. *If the filtration $\mathbb{F} = (\mathcal{F}_t)_{t\in\mathbb{T}}$ is right-continuous, i.e. if $\mathcal{F}_t^+ := \cap_{s\geq t}\mathcal{F}_s = \mathcal{F}_t, \forall t \in \mathbb{T}$, then the hitting time of Γ defined by*

$$\sigma_\Gamma = \inf\{t \geq 0 : X_t \in \Gamma\}$$

(with the convention $\inf\emptyset = \infty$) is a stopping time.

2. *If X is continuous, then the exit time of Γ defined by*

$$\tau_\Gamma = \inf\{t \geq 0 : X_t \notin \Gamma\}$$

is a predictable stopping time.

Proof:

For a proof the interested reader is referred to **[33, page 43f]**. □

2.2 Brownian Motion

> "The grains of pollen were particles... of a figure between cylindrical and oblong, perhaps slightly flattened... While examining the form of these particles immersed in water, I observed many of them very evidently in motion; their motion consisting not only of a change in place in the fluid manifested by alterations in their relative positions... In a few instances the particle was seen to turn on its longer axis. These motions were such as to satisfy me, after frequently repeated observations, that they arose neither from currents in the fluid, nor from its gradual evaporation, but belonged to the particle itself."
>
> **Robert Brown - Miscellaneous Botanical Works Vol. I, 1866**

After introducing the notion of a stochastic process, the most natural thing that follows is to produce a basic example of such a construct and that is the Brownian motion, also referred to as a Wiener process, since Norbert Wiener (1894-1964) was the first (around 1923) to develop a rigorous mathematical foundation that describes it.

The Brownian motion is one of the most important continuous stochastic processes because of its interesting mathematical properties and its broad application range: from biology to medicine, physics, engineering, and finance. In financial mathematics for example, Brownian motion is very often used for modelling security prices.

Definition 2.2.1 (Standard one-dimensional Brownian Motion) [4, page 1]

A **standard one-dimensional Brownian motion** *is a continuous-time stochastic process* $(W_t)_{t\in\mathbb{T}}$ *valued in* $\mathbb{R}$ *with the following properties:*

1. $W_0 = 0$ *almost surely;*
2. *the sample trajectories* $t \to W_t$ *are (almost surely) continuous;*
3. *for any finite sequence of times* $t_0 < t_1 < \ldots < t_n$*, the increments*

$$W_{t_1} - W_{t_0}, W_{t_2} - W_{t_1}, \ldots, W_{t_n} - W_{t_{n-1}} \tag{2.2}$$

   are independent random variables;

4. *for any given times* $0 \leq s < t < \infty$*,* $W_t - W_s$ *has the Gaussian distribution* $\mathcal{N}(0, t-s)$ *with mean zero and variance* $t - s$*.*

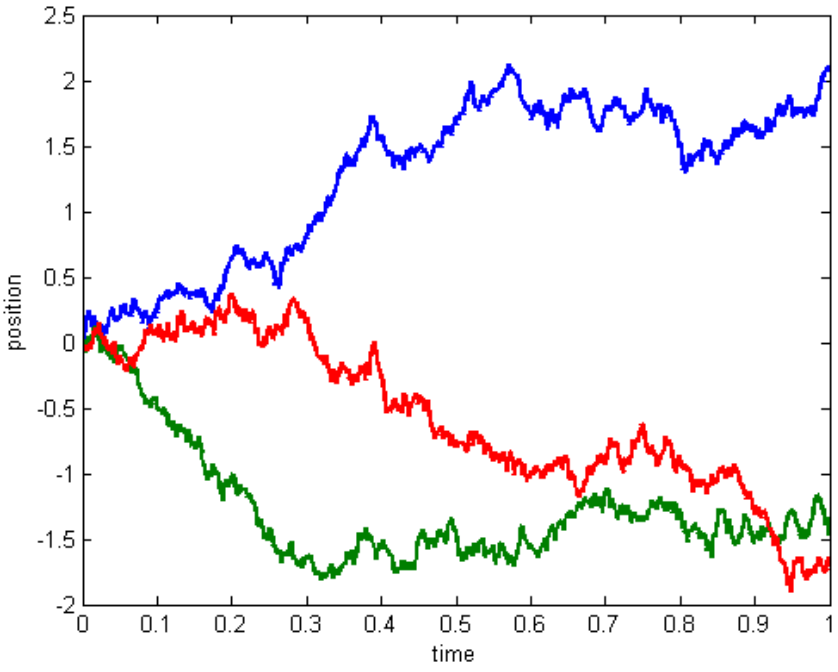

Figure 2.1: Sample paths of one-dimensional Brownian motion

For a proof for the existence of Brownian motion as a stochastic process $(W_t)_{t\in\mathbb{R}}$, satisfying properties 1.-4., the interested reader is referred to Chapter 1 of [5].

Figure 2.1 portrays three possible trajectories of standard one-dimensional Brownian motion.

Having defined the one-dimensional Brownian motion, the definition of a multi-dimensional Brownian motion follows.

Definition 2.2.2 (Standard d-dimensional Brownian Motion) [3, page 44]

Consider a continuous process $W = (W_t)_{t\in\mathbb{T}}$, where $W_t = (W_t^1, \ldots, W_t^d)_{t\in\mathbb{T}}$ is a d-dimensional random vector for each t. Let I_d be a d symmetric positive definite matrix. The process W is said to be a **d-dimensional standard Brownian motion** *valued in $\mathbb{R}^d$ with variance-covariance matrix I_d if the following conditions are satisfied:*

1. *$W_0 = 0$ almost surely;*
2. *the sample trajectories $t \to W_t$ are (almost surely) continuous;*
3. *for any finite sequence of times $t_0 < t_1 < \ldots < t_n$, the increments*

$$W_{t_1} - W_{t_0}, W_{t_2} - W_{t_1}, \ldots, W_{t_n} - W_{t_{n-1}} \tag{2.3}$$

   are independent random vectors;

4. *for any given times $0 \leq s < t < \infty$, $W_t - W_s$ follows a centred Gaussian distribution with variance-covariance matrix $(t-s)I_d$.*

Figure 2.2 displays three possible paths for two-dimensional Brownian motion.

Henceforth, all mathematical notions and properties introduced will refer to d-dimensional Brownian motion.

Definition 2.2.3 (Brownian motion with respect to a filtration) [1, page 5]

A vectorial (d-dimensional) Brownian motion on $\mathbb{T}$ with respect to a filtration $\mathbb{F} = (\mathcal{F}_t)_{t\in\mathbb{T}}$ is a continuous $\mathbb{F}$-adapted process, valued in $\mathbb{R}^d$, $(W_t)_{t\in\mathbb{R}} = (W_t^1, \ldots, W_t^d)_{t\in\mathbb{T}}$ such that:

1. *$W_0 = 0$;*
2. *for all $0 \leq s < t$ in $\mathbb{T}$, the increment $W_t - W_s$ is independent of $\mathcal{F}_s$ and follows a centred Gaussian distribution with variance-covariance matrix $(t-s)I_d$.*

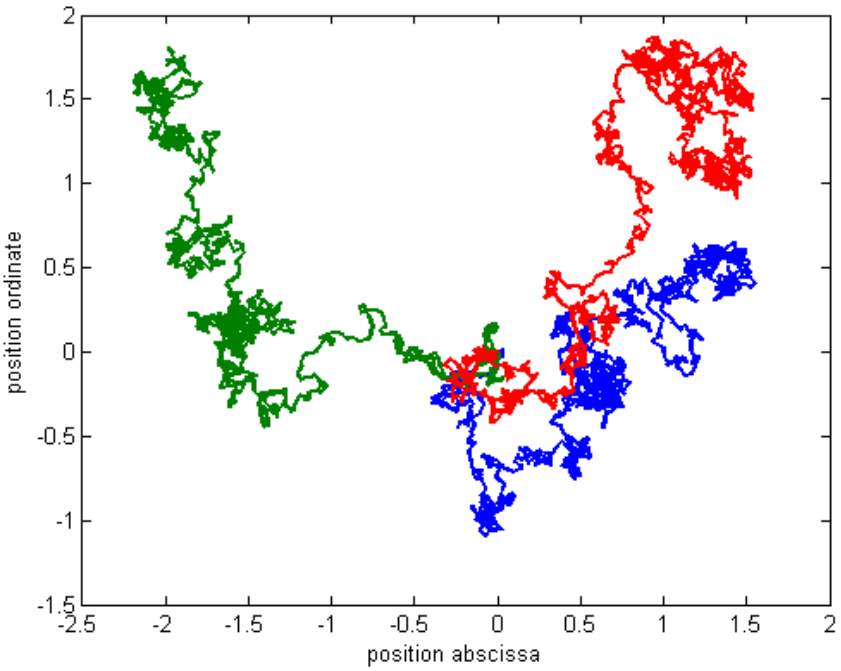

Figure 2.2: Sample paths of two-dimensional Brownian motion

As mentioned before, a Brownian motion has some very interesting mathematical properties, stated in the following

Proposition 2.2.1 [1, page 5]

Let $(W_t)_{t\in\mathbb{T}}$ be a Brownian motion with respect to $(\mathcal{F}_t)_{t\in\mathbb{T}}$.

1. **Symmetry**: *$(-W_t)_{t\in\mathbb{T}}$ is also a Brownian motion;*
2. **Scaling**: *for all $\lambda > 0$, the process $\left((1/\lambda)W_{\lambda^2 t}\right)_{t\in\mathbb{T}}$ is also a Brownian motion;*
3. **Invariance by translation**: *for all $s > 0$, the process $(W_{t+s} - W_s)_{t\in\mathbb{T}}$ is a standard Brownian motion independent of $\mathcal{F}_s$.*

Proof:

For a proof the interested reader is referred to **[33, page 20f]**. □

Definition 2.2.4 (Brownian Motion with drift) [6, page 153]

A process $(X_t), t \geq 0$ is called a (μ, σ^2) **Brownian motion with drift** *μ and volatility σ if it can be written in the form*

$$X_t = X_0 + \mu t + \sigma W_t,$$

where W_t is a standard Brownian motion.

A variation of the Brownian motion with drift that only allows positive values is the next in line to be defined geometric Brownian motion.

Definition 2.2.5 (Geometric Brownian Motion) [21, page 85]
A process $Y_t, t \geq 0$ with values in $(0, +\infty)$ is called a (μ, σ^2) **geometric Brownian motion** *if the process $X_t = \log(Y_t)$ is a $(\mu - \frac{\sigma^2}{2}, \sigma^2)$ Brownian motion with drift.*

We now want to provide a definition for a stopping time for Brownian motion.

Definition 2.2.6 (Stopping time for a Brownian motion) [8, page 7]
For a **Brownian motion** *in $\mathbb{R}^d$ and $a \in \mathbb{R}^d$, define $\tau_a = \inf\{t \geq 0 : W(t) = a\}$ as the first time W_t hits a.*

Bearing this in mind, in the next step we want to know what the hitting probability is in a Brownian motion process, i.e. the probability that such a process hits a given value for the first time. Such probabilities are utterly important in various problems in gambling and the theory of financial derivatives.

For a Brownian motion process with drift parameter μ and variance parameter σ^2, the following holds:

$$\mathbb{P}(\text{process hits } c \text{ before } -d) = \frac{1 - e^{-2d\mu/\sigma^2}}{1 - e^{-2(d+c)\mu/\sigma^2}}, \tag{2.4}$$

for any $c, d > 0$, where if $\mu = 0$, the probability should be read as $\frac{d}{d+c}$.

Finally, it can be easily verified that

$$\mathbb{P}(\text{process ever hits } -d) = e^{-2d\mu/\sigma^2}, \quad \text{for any } d > 0, \tag{2.5}$$

when the Brownian motion process has a **positive drift** parameter μ, and that

$$\mathbb{P}(\text{process ever hits c}) = e^{2c\mu/\sigma^2}, \quad \text{for any } c > 0, \tag{2.6}$$

when the process has a **negative drift** parameter μ. (see **[9, page 190]**)

2.3 Martingales

Martingales play a fundamental role in the theory of stochastic processes and in stochastic calculus because of their interesting and useful mathematical properties such as constant expectation, almost sure convergence etc.

A formal mathematical definition, whose main ingredient is the concept of conditional expectation, follows.

Definition 2.3.1 (Martingale) [10, page 183]

A stochastic process X_t (where time $t \in \mathbb{T}$), adapted to a filtration $\mathbb{F} = (\mathcal{F}_t)_{t \in T}$ is a **martingale** *if for any t, X_t is integrable, that is, $\mathbb{E}\big[X_t\big] < \infty$ and for any t and $s \in \mathbb{T}$ with $0 \leq s < t$,*

$$\mathbb{E}\big[X_t | \mathcal{F}_s\big] = X_s \quad \textit{almost surely.} \tag{2.7}$$

Now that the formal definition of a martingale is stated, it is time to name an example of such a mathematical construct. A typical one is the already known Brownian motion.

Definition 2.3.2 (Supermartingale / Submartingale) [10, page 183]

A stochastic process X_t, $t \in \mathbb{T}$, adapted to a filtration $\mathbb{F} = (\mathcal{F}_t)_{t \in \mathbb{T}}$ is a **supermartingale (submartingale)** *if it is integrable, that is, $\mathbb{E}\big[X_t\big] < \infty$ and for any t and $s \in \mathbb{T}$ with $0 \leq s < t$,*

$$\mathbb{E}\big[X_t | \mathcal{F}_s\big] \leq X_s \quad (\mathbb{E}\big[X_t | \mathcal{F}_s\big] \geq X_s) \quad \textit{almost surely.} \tag{2.8}$$

If the process $(X_t)_{t \in \mathbb{T}}$ is a supermartingale, then $-(X_t)_{t \in \mathbb{T}}$ is a submartingale.

As previously stated, the mean of a martingale is constant in t, while the mean of a supermartingale is non-increasing and the mean of a submartingale is non-decreasing in t.

Theorem 2.3.1 [10, page 184]

A supermartingale $(X_t)_{t \in \mathbb{T}}$, $\mathbb{T} = \big[0, T\big]$ is a martingale if and only if $\mathbb{E}\big[X_T\big] = \mathbb{E}\big[X_0\big]$.

Proof:

For a proof the interested reader is referred to **[10, page 184]**. □

Definition 2.3.3 (Local Martingale) [1, page 7]

Let X be a càd-làg adapted process. We say that X is a **local martingale** *if there exists a sequence of stopping times $(\tau_n)_{n \geq 1}$ such that $\lim\limits_{n \to \infty} \tau_n = \infty$ almost surely and the stopped process X^{τ_n} is a martingale for all n.*

2.4 Itô Calculus

The quintessence of stochastic analysis is represented by the so-called Itô calculus, which gives it a whole new meaning, entirely different from that of classical real-valued analysis. However, the world-renowned Itô formula is generally known as the stochastic calculus counterpart of the chain rule.

First, a class of functions upon which later definitions and terms depend is being introduced.

Definition 2.4.1 [11, page 25]
Let $\mathcal{V} = \mathcal{V}(S,T)$ be the class of functions

$$f(t,\omega) : [0,\infty) \times \Omega \to \mathbb{R} \tag{2.9}$$

such that

1. *$(t,\omega) \to f(t,\omega)$ is $\mathcal{B} \times \mathcal{F}$-measurable, where $\mathcal{B}$ denotes the Borel σ-algebra on $[0,\infty)$;*
2. *$f(t,\omega)$ is $\mathcal{F}_t$-adapted;*
3. *$\mathbb{E}\big[\int_S^T f(t,\omega)^2 dt\big] < \infty$.*

We are now able to define the Itô integral.

Definition 2.4.2 (One-dimensional Itô Integral) [11, page 29]
Let $f \in \mathcal{V} = \mathcal{V}(S,T)$. Then the **Itô integral** *of f (from S to T) is defined by*

$$\int_S^T f(t,\omega) dW_t(\omega) = \lim_{n\to\infty} \int_S^T \phi_n(t,\omega) dW_t(\omega) \quad (\text{limit in } L^2(\mathbb{P})), \tag{2.10}$$

where W_t is a standard Brownian motion and $(\phi_n)_{n\in\mathbb{N}}$ is a sequence of elementary functions such that

$$\mathbb{E}\left[\int_S^T (f(t,\omega) - \phi_n(t,\omega))^2 dt\right] \to 0 \quad \text{as } n \to \infty. \tag{2.11}$$

$\mathcal{V}$ is just one class of integrands f for which the Itô integral can be defined. In the following, a larger class of such functions is given.

Definition 2.4.3 [11, page 34f]
$\mathcal{W} = \mathcal{W}(S,T)$ denotes the class of processes $f(t,\omega) \in \mathbb{R}$ satisfying:

1. *$(t,\omega) \to f(t,\omega)$ is $\mathcal{B} \times \mathcal{F}$-measurable, where $\mathcal{B}$ denotes the Borel σ-algebra on $[0,\infty)$;*
2. *There exists an increasing family of σ-algebras $(\mathcal{F}_t)_{t\geq 0}$ such that*
   (a) *W_t is a martingale with respect to $\mathcal{F}_t$ and*
   (b) *f_t is $\mathcal{F}_t$-adapted;*
3. *$\mathbb{P}\big[\int_S^T f(s,\omega)^2 ds < \infty\big] = 1$.*

The time has come to define the notion of a one-dimensional Itô process.

Definition 2.4.4 (One-dimensional Itô Process) [11, page 43f]

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and W_t a one-dimensional Brownian motion on $(\Omega, \mathcal{F}, \mathbb{P})$. A **one-dimensional Itô process** *(or stochastic integral) is a stochastic process X_t on $(\Omega, \mathcal{F}, \mathbb{P})$ of the form*

$$X_t = X_0 + \int_0^t u(s,\omega)ds + \int_0^t v(s,\omega)dW_s, \tag{2.12}$$

where $v \in \mathcal{W}$, so that

$$\mathbb{P}\left[\int_0^t v(s,\omega)^2 ds < \infty, \forall t \geq 0\right] = 1. \tag{2.13}$$

We also assume that u is $\mathcal{F}_t$-adapted (see Definition 2.4.3) and

$$\mathbb{P}\left[\int_0^t |u(s,\omega)|ds < \infty, \forall t \geq 0\right] = 1. \tag{2.14}$$

An Itô process can also be written in the shorter, differential form

$$dX_t = udt + vdW_t. \tag{2.15}$$

We are now able to state the famous Itô formula.

Theorem 2.4.1 (The one-dimensional Itô Formula) [11, page 44]

Let X_t be an Itô process given by

$$dX_t = udt + vdW_t.$$

Let $f(t,x) \in C^2([0,\infty) \times \mathbb{R})$ (i.e. f is twice continuously differentiable on $[0,\infty) \times \mathbb{R}$). Then $Y_t = f(t, X_t)$ is again an Itô process, and

$$dY_t = \frac{\partial f}{\partial t}(t, X_t)dt + \frac{\partial f}{\partial x}(t, X_t)dX_t + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}(t, X_t) \cdot (dX_t)^2, \tag{2.16}$$

where $(dX_t)^2 = (dX_t) \cdot (dX_t)$ is computed according to the rules

$$dt \cdot dt = dt \cdot dW_t = dW_t \cdot dt = 0, \quad dW_t \cdot dW_t = dt.$$

Proof:

For a proof the interested reader is referred to **[11, page 46ff]**. □

Not only one-dimensional Itô processes and their corresponding Itô formula are essential in stochastic analysis, but also their multi-dimensional extension.

Definition 2.4.5 (Multi-dimensional Itô Process) [11, page 48]

Let $W(t,\omega) = (W_1(t,\omega), \ldots, W_m(t,\omega))$ denote m-dimensional Brownian motion. If each of the

processes $u_i(t,\omega)$ and $v_{ij}(t,\omega)$ satisfies the conditions given in Definition **2.4.4** *$(1 \leq i \leq n, \quad 1 \leq j \leq m)$, then we can form the following n Itô processes*

$$\begin{cases} dX_1 = u_1 dt + v_{11} dW_1 + \ldots + v_{1m} dW_m \\ \vdots \\ dX_n = u_n dt + v_{n1} dW_1 + \ldots + v_{nm} dW_m. \end{cases} \tag{2.17}$$

Or, in matrix notation simply

$$dX(t) = udt + vdW(t), \tag{2.18}$$

where

$$X(t) = \begin{pmatrix} X_1(t) \\ \vdots \\ X_n(t) \end{pmatrix}, u = \begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix}, v = \begin{pmatrix} v_{11} & \ldots & v_{1m} \\ \vdots & & \vdots \\ v_{n1} & \ldots & v_{nm} \end{pmatrix}, dW(t) = \begin{pmatrix} dW_1(t) \\ \vdots \\ dW_m(t) \end{pmatrix}. \tag{2.19}$$

Such a process $X(t)$ is called a **n-dimensional Itô process** *(or just an Itô process).*

Theorem 2.4.2 (The General Itô Formula) [11, page 48f]

Let

$$dX(t) = udt + vdW(t)$$

be a n-dimensional Itô process as above. Let $f(t,x) = (f_1(t,x), \ldots, f_p(t,x))$ be a C^2 map from $[0,\infty) \times \mathbb{R}^n$ into $\mathbb{R}^p$. Then the process $Y(t,\omega) = f(t, X_t)$ is again an Itô process, whose component number k, Y_k, is given by

$$dY_k = \frac{\partial f_k}{\partial t}(t,X)dt + \sum_i \frac{\partial f_k}{\partial x_i}(t,X)dX_i + \frac{1}{2}\sum_{i,j} \frac{\partial^2 f_k}{\partial x_i \partial x_j}(t,X)dX_i dX_j, \tag{2.20}$$

where $dW_i \cdot dW_j = \delta_{ij} \cdot dt, \quad dW_i \cdot dt = dt \cdot dW_i = 0.$

Here δ_{ij} represents Kronecker's delta, for which holds $\delta_{ij} = 1$, for $i = j$ and $\delta_{ij} = 0$, for $i \neq j$.

The proof for the multi-dimensional Itô formula is quite similar to the one-dimensional version. The interested reader is referred to **[11]** for a proof and a thorough depiction of Itô calculus.

2.5 Stochastic Differential Equations

First, we want to make a list of assumptions (see **[12]**), that we will need later on in this section. Let:

- $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space;
- $(\mathcal{F}_t)_{t \geq 0}$ be a filtration on that probability space;

- $W(t) = (W_1(t), \ldots, W_m(t))$, $t \geq 0$ be an m-dimensional Brownian motion defined on the probability space;
- $0 \leq t_0 < T < \infty$;
- $X_0 = X_{t_0}$ be a $\mathcal{F}_{t_0}$-measurable $\mathbb{R}^d$-valued random variable such that $\mathbb{E}\big[|X_0|^2\big] < \infty$;
- $f : \mathbb{R}^d \times \big[t_0, T\big] \to \mathbb{R}^d$ and $g : \mathbb{R}^d \times \big[t_0, T\big] \to \mathbb{R}^{d\times m}$ be Borel measurable.

Second, we consider the d-dimensional stochastic differential equation (SDE) of Itô type:

$$dX(t) = f(X(t), t)dt + g(X(t), t)dW(t) \quad \text{on} \quad \big[t_0, T\big], \tag{2.21}$$

having $X(t_0) = X_0$ as initial condition, which is equivalent to the following stochastic integral equation:

$$X(t) = X_0 + \int_{t_0}^{t} f(x(s), s)ds + \int_{t_0}^{t} g(x(s), s)dW(s) \quad \text{on} \quad \big[t_0, T\big], \tag{2.22}$$

Seeing these equations, any curious reader would ask himself what is the solution, how does it look like and what properties does it have.

Definition 2.5.1 (Solution to a Stochastic Differential Equation) [12, page 48]
An $\mathbb{R}^d$-valued process $(X_t)_{t_0\leq t\leq T}$ is called a **solution of the stochastic differential equation** (2.21) *if it has the following properties:*

1. $(X(t))_{t_0\leq t\leq T}$ *is continuous and $\mathcal{F}_t$-adapted;*
2. $(f(X(t), t)) \in \mathcal{L}^1(\big[t_0, T\big]; \mathbb{R}^d)$*;*
3. $(g(X(t), t)) \in \mathcal{L}^2(\big[t_0, T\big]; \mathbb{R}^{d\times m})$*;*
4. *equation* (2.22) *holds true for every $t \in \big[t_0, T\big]$ with probability* 1.

A solution $(X(t))$ is said to be unique if any other solution $(\overline{X}(t))$ is indistinguishable from $(X(t))$, that is

$$\mathbb{P}\Big(X(t) = \overline{X}(t), \quad \forall t \in \big[t_0, T\big]\Big) = 1.$$

Chapter 3

Introduction to Stochastic Control Theory

3.1 Introduction

This section is dedicated to sketching the general features that define a continuous-time stochastic optimization problem.

For a more detailed illustration, the interested reader is referred to [1] and [13].

First, we want to specify the basic mathematical frame for such a problem, i.e. we consider a dynamic system evolving in an uncertain environment, represented by a probability space $(\Omega, \mathcal{F}, \mathbb{P})$.

When referring to a stochastic optimization problem, the following characteristics are indispensable:

- **State process** X_t

  The state of the system consists of a set of random variables that serve their purpose of describing the problem. We denote by $X_t(\omega)$ the state process at time t, for a trajectory ω, symbolizing a possible world scenario. Furthermore, the continuous-time dynamics of the state system, i.e. $t \mapsto X_t(\omega)$ for all ω, is given by a stochastic differential equation (SDE) or process, as presented in the previous chapter.

- **Control process** u_t

  The dynamics of the system, i.e. the stochastic differential equation, is usually also influenced by a controllable variable, modelled as a process (u_t), whose value at any time t is given as a function of the available information. $\mathcal{U}$ represents the set of admissible controls.

- **Objective function** $J(X, u)$

  The aim of any optimization problem is to either maximize or minimize a certain functional, say $J(X, u)$, over all admissible controls $u \in \mathcal{U}$. Typically, objective functionals

are given by

$$\mathbb{E}\Big[\int_0^T f(X_t,\omega,u_t)dt + g(X_T,\omega)\Big], \quad \text{on a finite time horizon} \quad T<\infty \tag{3.1}$$

and

$$\mathbb{E}\Big[\int_0^\infty e^{-\beta t} f(X_t,\omega,u_t)dt\Big], \quad \text{on an infinite time horizon,} \tag{3.2}$$

where f is a running profit function, g a terminal reward function and $\beta > 0$ a discount factor.

- **Value function** $V(X)$

  The maximum value, also referred to as value function, is defined by

$$V(X) = \sup_{u\in\mathcal{U}} \quad J(X,u). \tag{3.3}$$

The main goal of stochastic control theory is to find an optimal control $u^* \in \mathcal{U}$, that maximizes the value function given a certain initial condition.

3.2 Controlled Diffusion Processes

Having already set the preliminaries, in this section we will go further in detail and offer a thorough mathematical description of stochastic control problems.

In order to proceed, we have to make a series of assumptions that will ease the way. Therefore, let:

- $(\Omega, \mathcal{F}, \mathbb{F} = (\mathcal{F}_t)_{t\geq 0}, \mathbb{P})$ be a filtered probability space, where the filtration satisfies the conditions of continuity and completeness;
- $W_t = (W_t^1, \ldots, W_t^d)$ be a d-dimensional Brownian motion;
- the control model be given by the following stochastic differential equation (SDE) valued in $\mathbb{R}^n$:

$$dX_t = b(t,X_t,u_t)dt + \sigma(t,X_t,u_t)dW_t; \tag{3.4}$$

- the control $u = (u_t)$ be a progressively measurable process with respect to $\mathbb{F}$, with values in $U \subset \mathbb{R}^m$;
- the functions $b : \mathbb{R}^+ \times \mathbb{R}^n \times U \to \mathbb{R}^n$ and $\sigma : \mathbb{R}^+ \times \mathbb{R}^n \times U \to \mathbb{R}^{n\times d}$ be measurable and satisfy a uniform Lipschitz condition in U: $\quad \exists K \geq 0, \quad \forall X, Y \in \mathbb{R}^n, \quad \forall u \in U,$

$$|b(t,X,u) - b(t,Y,u)| + |\sigma(t,X,u) - \sigma(t,Y,u)| \leq K|X-Y|. \tag{3.5}$$

3.2.1 The Finite Horizon Problem

In this subsection we will consider a finite horizon $0 < T < \infty$. Let $\mathcal{U}_0$ denote the set of control processes u such that

$$\mathbb{E}\Big[\int_0^T |b(t,0,u_t)|^2 + |\sigma(t,0,u_t)|^2 dt\Big] < \infty. \tag{3.6}$$

Conditions (3.5) and (3.6) ensure the existence and uniqueness of a solution to the stochastic differential equation with random coefficients for all $u \in \mathcal{U}_0$ and for any initial condition $(t,x) \in \big[0,T\big] \times \mathbb{R}^n$.

We denote by $\{X_s^{t,x}, t \le s \le T\}$ the solution with almost sure continuous paths starting from x at $s = t$.

Functional Objective

Let:

- $f : \big[0,T\big] \times \mathbb{R}^n \times A \to \mathbb{R}$ be a measurable function;
- $g : \mathbb{R}^n \to \mathbb{R}$ be a measurable function that either
  1. is lower-bounded;
     or
  2. satisfies a quadratic growth condition: $|g(x)| \le C(1+|x|^2), \quad \forall x \in \mathbb{R}^n, \quad$ for some constant C independent of x;
- $\mathcal{U}(t,x)$ denote the non-empty subset of controls $u \in \mathcal{U}$ such that

$$\mathbb{E}\Big[\int_t^T |f(s, X_s^{t,x}, u_s)| ds\Big] < \infty, \quad (t,x) \in \big[0,T\big] \times \mathbb{R}^n. \tag{3.7}$$

We are now able to define the **objective function** for the finite horizon problem, for all $(t,x) \in \big[0,T\big] \times \mathbb{R}^n$ and $u \in \mathcal{U}(t,x)$ as:

$$J(t,x,u) = \mathbb{E}\Big[\int_t^T f(s, X_s^{t,x}, u_s) ds + g(X^{t,x_T})\Big]; \tag{3.8}$$

Since the goal is to maximize the objective function over all control processes $u \in \mathcal{U}(t,x)$, the **value function** can be written in the following form:

$$V(t,x) = \sup_{u \in \mathcal{U}(t,x)} J(t,x,u). \tag{3.9}$$

A **control** $u^* \in \mathcal{U}(t,x)$ is said to be **optimal**, if

$$V(t,x) = J(t,x,u^*), \tag{3.10}$$

given an initial condition $(t,x) \in \big[0,T\big] \times \mathbb{R}^n$.

3.2.2 The Infinite Horizon Problem

In this subsection we will consider an infinite horizon $T = \infty$. Let $\mathcal{U}_0$ denote the set of control processes u such that

$$\mathbb{E}\Big[\int_0^T |b(t,0,u_t)|^2 + |\sigma(t,0,u_t)|^2 dt\Big] < \infty, \quad \forall T > 0. \tag{3.11}$$

We denote by $\{X_s^x, s \geq 0\}$ the unique solution to the SDE (3.4) starting from $x \in \mathbb{R}^n$ at $t = 0$, for a given control $u \in \mathcal{U}_0$.

Functional Objective

Let:

- $\beta > 0$;
- $f : \mathbb{R}^n \times A \to \mathbb{R}$ be a measurable function;
- $\mathcal{U}(x) \subset \mathcal{U}_0$ denote the non-empty subset of controls $u \in \mathcal{U}_0$ such that

$$\mathbb{E}\Big[\int_0^\infty e^{-\beta s}|f(X_s^x, u_s)|ds\Big] < \infty, \quad x \in \mathbb{R}^n. \tag{3.12}$$

Hence, we obtain the following forms:

$$J(x,u) = \mathbb{E}\Big[\int_0^\infty e^{-\beta s} f(X_s^x, u_s)ds\Big], \quad \forall x \in \mathbb{R}^n, \quad u \in \mathcal{U}(x), \tag{3.13}$$

for the **objective function** and

$$V(x) = \sup_{u \in \mathcal{U}(x)} J(x,u), \tag{3.14}$$

for the **value function**.

As previously mentioned, a **control** $u^* \in \mathcal{U}(x)$ is said to be **optimal** if

$$V(x) = J(x,u^*). \tag{3.15}$$

3.3 Dynamic Programming Principle

The dynamic programming principle (DPP) is one of the most important notions in dynamic optimization and stochastic control theory. It provides support for the solution technique developed by Richard Bellman in the 1950s.

The interested reader is referred to **[14]** for Richard Bellman's most famous work entitled "Dynamic Programming".

The formulation of the Dynamic Programming Principle for controlled diffusion processes, for both finite time and infinite time horizon is synthesized in the following

Theorem 3.3.1 (Dynamic Programming Principle) [1, page 40]

1. **Finite Horizon:** *let* $(t,x) \in [0,T] \times \mathbb{R}^n$. *Then we have:*

$$V(t,x) = \sup_{u \in \mathcal{U}(t,x)} \sup_{z \in \mathcal{Z}_{t,T}} \mathbb{E}\Big[\int_t^z f(s, X_s^{t,x}, u_s)ds + V(z, X_z^{t,x})\Big] \tag{3.16}$$

$$= \sup_{u \in \mathcal{U}(t,x)} \inf_{z \in \mathcal{Z}_{t,T}} \mathbb{E}\Big[\int_t^z f(s, X_s^{t,x}, u_s)ds + V(z, X_z^{t,x})\Big]. \tag{3.17}$$

2. **Infinite Horizon:** *let* $x \in \mathbb{R}^n$. *Then we have:*

$$V(x) = \sup_{u \in \mathcal{U}(x)} \sup_{z \in \mathcal{Z}} \mathbb{E}\Big[\int_0^z e^{-\beta s} f(X_s^x, u_s)ds + e^{-\beta z} V(X_z^x)\Big] \tag{3.18}$$

$$= \sup_{u \in \mathcal{U}(x)} \inf_{z \in \mathcal{Z}} \mathbb{E}\Big[\int_0^z e^{-\beta s} f(X_s^x, u_s)ds + e^{-\beta z} V(X_z^x)\Big], \tag{3.19}$$

with the convention that $e^{-\beta z(\omega)} = 0$ *when* $z(\omega) = \infty$.

Here, $\mathcal{Z}_{t,T}$ *respectively* $\mathcal{Z}$ *denote the sets of stopping times (i.e. the ending time of the controller's objective) valued in* $[t,T]$ *and* $[0,\infty]$ *respectively, for* $0 \leq t \leq T < \infty$, $t = 0$ *and* $T = \infty$ *respectively.*

Proof:

For a proof the interested reader is referred to **[1, page 41]**. □

3.4 Hamilton-Jacobi-Bellman Equation

The Hamilton-Jacobi-Bellman Equation (HJB), also called the dynamic programming equation, can be seen as a refinement of the dynamic programming principle, as it goes further into detail describing the local behaviour of the value function when the stopping time z is assumed to converge to t.

3.4.1 Derivation of the Hamilton-Jacobi-Bellman Equation

Finite Horizon Problem

As a first step, we want to establish the notation that we will use for the formal derivation of the HJB equation (see [1]). Therefore let:

- $\mathcal{S}_n$ denote the set of symmetric $n \times n$ matrices;
- $tr(A)$ be the trace of some $n \times n$ matrix A;

- D_x stand for the gradient vector of a function $x \to f(t,x) \in C^2$;
- D_x^2 symbolize the Hessian matrix of function f in $\mathcal{S}_n$;
- $\sigma(x,u)\sigma(x,u)^T$ denote the diffusion matrix;
- the stopping time be $z = t + h$;
- the control process be constant $u_t = u_c, \quad u_c \in U$ arbitrary;
- $\mathcal{L}^{u_c}$ stand for the operator associated with the diffusion (3.4) for the constant control u_c, that is defined as:

$$\mathcal{L}^{u_c} = b(x,u_c)D_x + \frac{1}{2}tr(\sigma(x,u_c)\sigma(x,u_c)^T D_x^2). \tag{3.20}$$

Taking into account the above assumptions, an equivalent representation of the dynamic programming principle is given by

$$V(t,x) \geq \mathbb{E}\Big[\int_t^{t+h} f(s, X_s^{t,x}, u_c)ds + V(t+h, X_{t+h}^{t,x})\Big]. \tag{3.21}$$

By applying Itô's formula between t and $t+h$ we get

$$V(t+h, X_{t+h}^{t,x}) = V(t,x) + \int_t^{t+h} \left(\frac{\partial V}{\partial t} + \mathcal{L}^{u_c}V\right)(s, X_s^{t,x})ds + \text{a (local) martingale with } \mathbb{E} = 0. \tag{3.22}$$

By inserting (3.22) in (3.21) we obtain

$$0 \geq \mathbb{E}\Big[\int_t^{t+h} \left(\frac{\partial V}{\partial t} + \mathcal{L}^{u_c}V\right)(s, X_s^{t,x}) + f(s, X_s^{t,x}, u_c)ds\Big].$$

Division by h, letting h converge to 0 together with the application of the mean-value theorem yield after additionally multiplying both sides with -1:

$$-\frac{\partial V}{\partial t}(t,x) - \sup_{u_c \in \mathcal{U}}\big[\mathcal{L}^{u_c}V(t,x) + f(t,x,u_c)\big] \geq 0. \tag{3.23}$$

Approaching the problem from another perspective, let us suppose $u^* \in \mathcal{U}$ is an optimal control. Then, the following holds true:

$$V(t,x) = \mathbb{E}\Big[\int_t^{t+h} f(s, X_s^*, u^*)ds + V(t+h, X_{t+h}^*)\Big],$$

with X^* denoting the solution to (3.4) starting from state x at time t, for the optimal control u^*. Through similar steps as above, we obtain an equation equivalent to (3.23) for the optimal control u_c

$$-\frac{\partial V}{\partial t}(t,x) - \mathcal{L}^{u_t^*}V(t,x) + f(t,x,u_t^*) = 0. \tag{3.24}$$

From (3.23) and (3.24) it becomes obvious that V should satisfy the following partial differential equation (PDE), also referred to as the **Hamilton-Jacobi-Bellman equation**

$$-\frac{\partial V}{\partial t}(t,x) - \sup_{u_c \in \mathcal{U}} \left[\mathcal{L}^{u_c} V(t,x) + f(t,x,u_c)\right] = 0, \quad \forall (t,x) \in \left[0,T\right) \times \mathbb{R}^n. \tag{3.25}$$

The literature often writes the above PDE in the following form:

$$-\frac{\partial V}{\partial t}(t,x) - H(t,x,D_x V(t,x), D_x^2 V(t,x))) = 0, \quad \forall (t,x) \in \left[0,T\right) \times \mathbb{R}^n, \tag{3.26}$$

with function H being defined as

$$H(t,x,p,M) = \sup_{u_c \in \mathcal{U}} \left[b(x,u_c)p + \frac{1}{2} tr(\sigma\sigma^T(x,u_c)M) + f(t,x,u_c)\right], \tag{3.27}$$

for $(t,x,p,M) \in \left[0,T\right] \times \mathbb{R}^n \times \mathbb{R}^n \times \mathcal{S}_n$ and being called the **Hamiltonian** of the control problem.

Finally, let us not forget the terminal condition associated with the above PDE:

$$V(T,x) = g(x), \quad \forall x \in \mathbb{R}^n. \tag{3.28}$$

Infinite Horizon Problem

The derivation of the HJB equation for the infinite horizon problem is carried out through similar arguments as in the finite horizon problem. Thus, we obtain an infinite horizon equivalent to (3.25):

$$-\beta V(x) - \sup_{u_c \in \mathcal{U}} \left[\mathcal{L}^{u_c} V(x) + f(x,u_c)\right] = 0, \quad \forall x \in \mathbb{R}^n, \tag{3.29}$$

which may also be written in **Hamiltonian** form as:

$$-\beta V(x) - H(x, D_x V(x), D_x^2 V(x)) = 0, \quad \forall x \in \mathbb{R}^n, \tag{3.30}$$

with

$$H(x,p,M) = \sup_{u_c \in \mathcal{U}} \left[b(x,u_c)p + \frac{1}{2} tr(\sigma\sigma^T(x,u_c)M) + f(x,u_c)\right], \quad \forall (x,p,M) \in \mathbb{R}^n \times \mathbb{R}^n \times \mathcal{S}_n. \tag{3.31}$$

3.5 Verification Theorem

From the previous section, we may draw the conclusion that the Hamilton-Jacobi-Bellman equation is a necessary condition for the value function $V(t,x)$. Now, the following question arises: is a solution to the HJB equation automatically a value function?

The answer to this question is the so-called **Verification Theorem**.

Theorem 3.5.1 (Verification Theorem - Finite Horizon Problem) [1, page 47]
Let W be a function in $C^{1,2}([0,T)\times\mathbb{R}^n)$, satisfying a quadratic growth condition, i.e. there exists a constant C such that:

$$|W(t,x)| \leq (1+|x|^2), \quad \forall [0,T]\times\mathbb{R}^n.$$

1. *Suppose that:*

$$-\frac{\partial W}{\partial t}(t,x) - \sup_{u_c\in U}\left[\mathcal{L}^{u_c}W(t,x) + f(t,x,u_c)\right] \geq 0, \quad (t,x)\in[0,T)\times\mathbb{R}^n, \tag{3.32}$$

$$W(T,x) \geq g(x), \quad x\in\mathbb{R}^n. \tag{3.33}$$

Then $W \geq V$ on $[0,T]\times\mathbb{R}^n$.

2. *Suppose further that $W(T,.) = g$ and there exists a measurable function $\widehat{u}(t,x)$, $(t,x)\in[0,T)\times\mathbb{R}^n$, valued in U such that:*

$$\begin{aligned}-\frac{\partial W}{\partial t}(t,x) - \sup_{u_c\in U}\left[\mathcal{L}^{u_c}W(t,x) + f(t,x,u_c)\right] &= -\frac{\partial W}{\partial t}(t,x) - \mathcal{L}^{\widehat{u}(t,x)}W(t,x) - f(t,x,\widehat{u}(t,x))\\ &= 0,\end{aligned}$$

the SDE

$$dX_s = b(X_s,\widehat{u}(s,X_s))ds + \sigma(X_s,\widehat{u}(s,X_s))dW_s$$

admits a unique solution, denoted by $\widehat{X}_s^{t,x}$, given an initial condition $X_t = x$, and the process $\{\widehat{u}(s,\widehat{X}_s^{t,x}), t\leq s\leq T\}$ lies in $\mathcal{U}(t,x)$.

Then $W = V$ on $[0,T]\times\mathbb{R}^n$ and $\widehat{u}$ is an optimal control.

Theorem 3.5.2 (Verification Theorem - Infinite Horizon Problem) [1, page 49]
Let W be a function in $C^2(\mathbb{R}^n)$, satisfying a quadratic growth condition.

1. *Suppose that:*

$$\beta W(x) - \sup_{u_c\in U}\left[\mathcal{L}^{u_c}W(x) + f(x,u_c)\right] \geq 0, \quad x\in\mathbb{R}^n, \tag{3.34}$$

$$\limsup_{T\to\infty} \quad e^{-\beta T}\mathbb{E}\left[W(X_T^x)\right] \geq 0, \quad \forall x\in\mathbb{R}^n, \quad \forall u\in\mathcal{U}(x). \tag{3.35}$$

Then $W \geq V$ on $\mathbb{R}^n$.

2. *Suppose further that for all $x\in\mathbb{R}^n$, there exists a measurable function $\widehat{u}(x)$, $x\in\mathbb{R}^n$, valued in U such that:*

$$\begin{aligned}\beta W(x) - \sup_{u_c\in U}\left[\mathcal{L}^{u_c}W(x) + f(x,u_c)\right] &= \beta W(x) - \mathcal{L}^{\widehat{u}(x)}W(x) - f(x,\widehat{u}(x))\\ &= 0,\end{aligned}$$

the SDE

$$dX_s = b(X_s,\widehat{u}(X_s))ds + \sigma(X_s,\widehat{u}(X_s))dW_s$$

admits a unique solution, denoted by $\widehat{X}_s^x$, given an initial condition $X_0 = x$, satisfying

$$\liminf_{T\to\infty} \quad e^{-\beta T}\mathbb{E}\left[W(X_T^x)\right] \leq 0$$

and the process $\{\widehat{u}(\widehat{X}_s^x), s \geq 0\}$ *lies in* $\mathcal{U}(x)$.

Then $W = V$, $\forall x \in \mathbb{R}^n$ *and* $\widehat{u}$ *is an optimal control.*

Proof:

For proofs for both the finite and infinite horizon cases, the interested reader is referred to **[1, page 47ff]** and **[1, page 50]**. □

Chapter 4

Examples in Financial and Actuarial Mathematics

4.1 Merton's Problem of Portfolio Allocation

General Description

In this model the agent has the opportunity to invest in two different types of instruments that are present in the financial market: a riskless bond and a risky asset.

At any time the agent can choose how much of his wealth is invested in the risky asset, automatically investing the remaining wealth into the riskfree bond.

The goal of this model, as originally formulated by Robert C. Merton himself in [25], is to maximize the expected utility of consumption.

Nomenclature

First we want to establish the notation that we will use later on in this section:

T	time endpoint
t	time varying in $[0, T]$
τ	time of ruin
x_0	initial capital at $t = 0$
$X(t)$	wealth at time t
$B_0(t)$	value of riskfree asset at time t
$B(t)$	price of risky asset at time t
$W(t)$	Brownian motion
$u_1(t)$	number of shares invested in the risky asset at time t
$u_2(t)$	consumption rate
$U(u_2, t)$	utility function
$\rho, \mu, \sigma, \beta, \alpha, c$	constants

Mathematical Assumptions

In order to proceed we want to define the general mathematical tools that we need for this model according to Merton **[25]**, Pham **[1, page 28f, page 51ff]** and Schmidli **[26, pages 114-120]**:

$(\Omega, \mathcal{F}, \mathbb{F}, \mathbb{P})$	a filtered probability space such that there exists a unique solution for the optimal strategy
$\mathbb{F} = (\mathcal{F}_t)_{t\geq 0}$	Brownian filtration, i.e. the smallest right-continuous filtration that makes $W(t)$ adapted
$\mathbb{T} = [0, T]$	time interval
$\mathbb{W} = [0, +\infty)$	wealth interval meaning that the agent is not allowed to have debts
$\boldsymbol{u}(t) = (u_1(t), u_2(t))$	two-dimensional vector representing the control process
$\mathcal{U} = (-\infty, +\infty) \times [0, +\infty)$	set of admissible control processes that are càd-làg adapted such that they admit a unique solution
$U(u_2, t)$	strictly increasing strictly concave continuous in both t and u_2 continuously differentiable with respect to u_2 $U(0, t) = 0$ for simplicity reasons $\lim_{u_2 \to \infty} U(u_2, t) = 0$

Dynamics of the Model

The value process $B_0(t)$ and the price process $B(t)$ evolve according to the following SDEs

$$dB_0(t) = \rho B_0(t) dt, \tag{4.1}$$

$$dB(t) = \big(\mu dt + \sigma dW(t)\big) B(t), \tag{4.2}$$

where $\rho, \mu, \sigma > 0$ and $\rho < \mu$.

The dynamics of the self-financed wealth process is described by

$$dX(t) = \big[\big(u_1(t)\mu + (1 - u_1(t))\rho\big) X(t) - u_2(t)\big] dt + \sigma u_1(t) X(t) dW(t). \tag{4.3}$$

Optimal Consumption

The value of the strategy $\boldsymbol{u}(t) = (u_1(t), u_2(t))$ is represented by the following objective functional:

$$J(x, \boldsymbol{u}(t), \tau) = \mathbb{E}\left[\int_0^{\tau \wedge T} e^{-\beta t} U(u_2(t), t) dt \quad | \quad X(0) = x_0\right], \tag{4.4}$$

where $\beta > 0$ stands for the discount rate and $\tau := \inf\{t : X(t) = 0\}$. For simplicity reasons, we will henceforth omit the stopping time and convene that $u_2(t) = 0$ for $t \geq \tau$.

The value function thus becomes

$$V(x) = \sup_{\boldsymbol{u}(t)} \quad J(x, \boldsymbol{u}(t)). \tag{4.5}$$

Lemma 4.1.1 [26, page 115]

The function $V(x)$ is strictly increasing and concave with boundary value V(0)=0.

Proof:

For a proof the interested reader is referred to **[26, page 115f]**. □

Hamilton-Jacobi-Bellman Equation

For $\boldsymbol{u} = (u_1, u_2) \in \mathcal{U}$, the operator $\mathcal{L}^{\boldsymbol{u}}$ specified in Subsection 3.4.1 has the following form when applied on function $V(x)$:

$$-\mathcal{L}^{\boldsymbol{u}}V(x) = \big[\big(u_1(t)\mu + (1-u_1(t))\rho\big)x - u_2(t)\big]V_x + \frac{1}{2}\sigma^2 u_1(t)^2 x^2 V_{xx}, \tag{4.6}$$

where V_x and V_{xx} are continuous, since V is additionally assumed to be twice continuously-differentiable.

Thus the HJB-equation has the following form

$$\sup_{(u_1,u_2)}\left(\big[\big(u_1(t)\mu + (1-u_1(t))\rho\big)x - u_2(t)\big]V_x + \frac{1}{2}\sigma^2 u_1(t)^2 x^2 V_{xx} + U(u_2) - \beta V\right) = 0. \tag{4.7}$$

Since $V(x)$ is concave, we look for a solution $x > 0$ with $V_x > 0$, $V_{xx} < 0$ and $\lim_{x\to 0+} V_x = \infty$.

Optimal Solutions

Through elementary calculations we obtain the following candidates for the optimal strategies:

$$u_1^* = -\frac{(\mu-\rho)V_x}{\sigma^2 x V_{xx}}, \tag{4.8}$$

$$u_2^* = (U')^{-1}(V_x). \tag{4.9}$$

Now, let us consider a specific form of the utility function

$$U(u_2(t)) = \frac{u_2(t)^\alpha}{\alpha}, \quad \alpha \in (0,1), \tag{4.10}$$

pertaining to the class of functions of type HARA (hyperbolic absolute risk aversion). Karatzas has considered more than just this one type of utility functions. The interested reader is referred to **[7]** for his work.

A solution to the HJB-equation is, in this case, given by

$$V(x) = cx^\alpha, \quad \alpha \in (0,1), \tag{4.11}$$

where c is a constant, implying that the first and second derivatives have the following forms:

$$V_x = c\alpha x^{\alpha-1}, \tag{4.12}$$

$$V_{xx} = c\alpha(\alpha-1)x^{\alpha-2}. \tag{4.13}$$

In view of recent information, the optimal strategies may be updated to

$$u_1^* = \frac{\mu - \rho}{\sigma^2(1-\alpha)}, \tag{4.14}$$

$$u_2^* = (\alpha c)^{\frac{1}{\alpha-1}} x. \tag{4.15}$$

By insertion of equations (4.10)-(4.15) into the HJB-equation, we can determine the exact value of the constant c which is positive if and only if

$$\beta > \frac{(\mu-\rho)^2}{2\sigma^2(1-\alpha)} + \rho\alpha. \tag{4.16}$$

By inserting equations (4.10)-(4.15) into equation (4.3) we get the following SDE:

$$dX^*(t) = \frac{1}{\alpha-1} X^*(t)\big((1-\alpha)\rho + \frac{(\mu-\rho)^2}{\sigma^2} - (1-\alpha)c^{\frac{1}{\alpha-1}}\big)dt + X^*(t)\frac{\mu-\rho}{\sigma}dW(t), \tag{4.17}$$

which through Itô calculus finally gives

$$X^*(t) = X_0 \exp\left(\big((1 - \frac{\mu-\rho}{1-\alpha}\sigma^2)r + \frac{(1-2\alpha)(\mu-\rho)^2}{2\sigma^2(1-\alpha)^2} - c^{\frac{1}{\alpha-1}}\big)dt + \frac{\mu-\rho}{(1-\alpha)\sigma}W(t)\right). \tag{4.18}$$

The only thing left to do now is to verify if u_1^* and u_2^* are indeed optimal strategies and if $V(x)$ actually is the maximal discounted utility. We do so by double-checking the validity of the equality

$$V(x) = J(x, u_1^*, u_2^*), \quad \forall x > 0. \tag{4.19}$$

Since u_1^* is constant and u_2^* is linear in x, $x^*(t)$ satisfies the SDE (4.3), which has thus an explicit solution.

4.2 Minimizing Ruin Probabilities in Insurance Business

General Description

In ruin theory business surplus is mostly defined as follows (see [27]):

$$\text{Business Surplus} \quad = \quad \text{Initial Capital} \quad + \quad \text{Income} \quad - \quad \text{Outflow.}$$

In insurance business the income process is determined by the total sum of premiums collected from the insurance takers and the originated investment income, while the outflow process mostly consists of the aroused claims of the insurance takers. Note that both these processes are subject to risk.

In this model we consider an insurance business that has a fixed amount to offer for investment purposes. Its investment portfolio consists of a risky and a non-risky asset.

The insurance business has the following investment policy, according to [27]:

- a fixed amount independent of the surplus will be invested at any time;

- a fraction of this amount will be invested in the risky asset, the remainder in the riskless one;

- the fraction may change in time depending on the portfolio allocation scheme that best corresponds to the goal of the model.

The goal of this model is to minimize the ruin probability of the insurance business.

Nomenclature

Like in the previous section, we first want to establish the notation that we will use later on in this section:

t	time variable
s	current surplus
$S(t)$	surplus process
$C(t)$	claims process
λ	intensity of the claims process
X	claim amount
$F(x)$	distribution of the claim amount
$f(x)$	density of the claim amount
$I(t)$	investment return process
$R(t)$	risk process
A	fixed amount available for investment
$u(t)$	proportion of A invested in the risky asset
p	rate per time unit at which premiums are collected
$B_0(t)$	value of the non-risky asset
$B(t)$	price of the risky asset
$W(t)$	Brownian motion
r, ρ, μ, σ	constants

Mathematical Assumptions

We again make a series of assumptions that might come in handy later on:

$\mathcal{U} = \big[0, 1\big]$	set of admissible controls
$\phi(s) := \mathbb{P}\big(S(t) < 0 \quad \text{for some} \quad t \geq 0\big)$	infinite time ruin probability
$\psi(s) := 1 - \phi(s)$	probability of non-ruin or survival
$\psi(s)$	continuous on $\big[0, +\infty\big)$
	twice continuously-differentiable on $(0, +\infty)$
	$\psi(s) = 0$ for $s < 0$
	$\psi'(s) \geq 0, \quad \psi''(s) \leq 0$ for $s > 0$
	$\lim\limits_{s \to +\infty} \psi(s) = 1$

Dynamics of the Model

The value of the non-risky asset follows

$$dB_0(t) = \rho B_0(t)dt, \quad \rho > 0, \tag{4.20}$$

while the price of the risky asset evolves according to a geometric Brownian motion

$$dB(t) = \big(\mu dt + \sigma dW(t)\big)B(t), \quad \mu, \sigma \geq 0. \tag{4.21}$$

The risk process of the insurance business follows a Cramér-Lundberg process consisting of the collection rate of premiums and the claims process, thus being determined by

$$dR(t) = pdt - dC(t), \tag{4.22}$$
$$R(0) = r. \tag{4.23}$$

Worth mentioning at this point is that a claim of amount X may occur with probability $\lambda dt + \mathcal{O}(dt)$ leaving the probability of no claim occurrence to be $1 - \lambda dt + \mathcal{O}(dt)$. The amount $pdt + \mathcal{O}(dt)$ constitutes the premium income.

The evolution of the investment return process is governed by the following SDE

$$dI(t) = A(1 - u(t))\rho dt + Au(t)\mu dt + Au(t)\sigma dW(t), \tag{4.24}$$

where the amount $A(1 - u(t))\rho dt + \mathcal{O}(dt)$ stands for the investment income from the riskless asset whereas $Au(t)\mu dt + Au(t)\sigma dW(t)$ represents the amount received as an investment income from the risky asset. (see **[28, page 507]**)

Concluding, the dynamics of the surplus process of this insurance business is given by

$$dS(t) = dR(t) + dI(t), \quad t \geq 0, \tag{4.25}$$

where $S(0) = s$.

Optimal Control Problem

The control problem can now be formulated as follows:

$$\min_{0 \leq u(t) \leq 1} \quad \mathbb{P}\big(S(t) < 0 \quad \text{for some} \quad t \geq 0\big) \tag{4.26}$$
$$\text{s.t.} \quad dS(t) = pdt - dC(t) + A(1 - u(t))\rho dt + Au(t)\mu dt + Au(t)\sigma dW(t) \tag{4.27}$$
$$S(0) = s. \tag{4.28}$$

Hamilton-Jacobi-Bellman Equation

We now shift to considering the probability of non-ruin, which we, in turn, have to maximize.

For that purpose, we consider two distinct cases over a time interval $\big[0, dt\big]$, as suggested by Castillo and Parrocha in **[27]**, and assume additionally that no premium was received during this period:

1. **no claim** in which case the surplus grows to $s + pdt + dI(t)$;
2. **exactly one claim** in which case the surplus reduces to $s + dI(t) - X$.

By taking expectations we are now able to determine the probability of non-ruin as

$$\psi(s) = \lambda dt \mathbb{E}\big[\psi(s - X)\big] + (1 - \lambda dt)\psi(s + pdt + dI(t)). \tag{4.29}$$

By applying Itô's Formula we get

$$\psi(s) = \psi(s) + \Big[\frac{1}{2}\sigma^2 A^2 u^2 \psi''(s) + \big(p + A(1-u)\rho + Au\mu\big)\psi'(s) + \lambda \mathbb{E}\big[\psi(s - X) - \psi(s)\big]\Big]dt.$$

For the sake of simplicity we denoted the control variable simply as u, bearing in mind that it only depends on the current surplus level s.

The last equation finally leads to the HJB-equation of this control problem

$$\sup_{0 \le u \le 1} \Big[\frac{1}{2}\sigma^2 A^2 u^2 \psi''(s) + \big(p + A(1-u)\rho + Au\mu\big)\psi'(s) + \lambda \mathbb{E}\big[\psi(s - X) - \psi(s)\big]\Big] = 0. \tag{4.30}$$

Optimal Investment Strategy

Equation (4.30) attains a maximum at

$$\widehat{u} = -\frac{(\mu - \rho)\psi'(s)}{A\sigma^2 \psi''(s)}, \tag{4.31}$$

implying that the optimal control is only a function of the current surplus.

Inserting (4.31) into equation (4.30) yields the following differential equation, whose solution, in turn, determines the fraction of the optimal strategy

$$\lambda \mathbb{E}\big[\psi(s - X) - \psi(s)\big] = \frac{1}{2}\frac{(\rho - \mu)^2 \psi'(s)}{\sigma^2 \psi''(s)} - (p + A\mu)\psi'(s). \tag{4.32}$$

The theorem that follows is the verification theorem for this specific model, thus corroborating that $\widehat{u}$ is the optimal proportion corresponding to the current surplus s.

Theorem 4.2.1 [28, page 509]

Suppose there exists a solution $\psi_{\widehat{u}}(s)$ to the HJB equation (4.30) having maximizer as defined in equation (4.31) such that $\psi_{\widehat{u}}(0) > 0$, $\psi'_{\widehat{u}}(0) > 0$, $\psi_{\widehat{u}}(s) = 0$ for $s < 0$, $\lim_{s \to +\infty} \psi_{\widehat{u}}(s) = 1$ and $\psi_{\widehat{u}}(s)$ is twice continuously differentiable on $s > 0$.

If $u(t)$ is an arbitrary admissible strategy for which the corresponding surplus process $S(t)$ is defined on $0 \le t < \infty$, then the corresponding non-ruin probability $\psi_u(s)$ for this process with initial surplus s satisfies

$$\psi_u(s) \le \psi_{\widehat{u}}(s), \quad s \ge 0.$$

Proof:

For a proof the interested reader is referred to Castillo and Parrocha [27]. □

Remark 4.2.1 *The optimal control $\widehat{u}$, as given in equation (4.31), does not necessarily belong to the interval $[0,1]$, which is why we want to consider three distinct cases for the optimal control u^*:*

$$u^* = \begin{cases} 0, & \text{if } \widehat{u} < 0 \\ \widehat{u}, & \text{if } \widehat{u} \in [0,1] \\ 1, & \text{if } \widehat{u} > 1. \end{cases} \tag{4.33}$$

As a next step we want to specify the non-ruin probabilities, denoted as $\psi_0, \psi_{\widehat{u}}, \psi_1$, for each of these cases:

1. $u^* = 0$

$$\psi_0'(s) = \frac{\lambda}{p + A\rho}\mathbb{E}\big[\psi_0(s) - \psi(s - X)\big],$$

2. $u^* = \widehat{u}$

$$\psi_{\widehat{u}}'(s) = \left(\frac{(\rho-\mu)^2}{2\sigma^2}\int_0^s \frac{1}{\lambda\mathbb{E}\big[\psi_{\widehat{u}}(t) - \psi_{\widehat{u}}(t-X)\big] - (p+A\rho)\psi_{\widehat{u}}'(t)}dt + \frac{p+A\rho}{\lambda\psi_{\widehat{u}}(0)}\right)^{-1},$$

3. $u^* = 1$

$$\psi_1'(s) = \frac{2}{A^2\sigma^2}\int_{s_1}^s \Big(\lambda\mathbb{E}\big[\psi_1(t) - \psi_1(t-X)\big] - (p + A\mu)\psi_1'(t)\Big)dt + \psi_1(s_1).$$

For a formal derivation of the above functions as well as for properties of these non-ruin probabilities and for a proof of whether a solution to the HJB-equation exists, the interested reader is referred to [27], [28].

Chapter 5

A Theoretical Insight in Some Relevant Methodology

5.1 Numerical Approximation

In the following we want to devote our attention to partial differential equations (PDEs) in two independent variables with given initial (IC) or boundary conditions (BC).

Since this type of equations is difficult to solve **analytically** (i.e. exactly), a **numerical** procedure seems an agreeable alternative to find an **approximate** solution.

The interested reader is referred to Causon's and Mingham's work **[19]** on approximating partial differential equations.

5.1.1 Taylor's Formula

When considering numerical approximation, the first thing that comes to mind is Taylor series, for it is always a simple and effective tool to obtain valuable estimations. Let's recall the following

Theorem 5.1.1 (Taylor's Theorem) [19, page 17]

Let $V : [a,b] \to \mathbb{R}, \quad V, V', V'', \ldots, V^{(n-1)}$ *be continuous on* $[a,b]$ *and suppose* $V^{(n)}$ *exists on* (a,b). *Then for* $a < x_0 < x_0 + h < b$ *it holds:*

$$V(x_0+h) = V(x_0) + hV'(x_0) + h^2\frac{V''(x_0)}{2!} + \ldots + h^{n-1}\frac{V^{(n-1)}(x_0)}{(n-1)!} + \mathcal{O}(h^n). \tag{5.1}$$

Proof:

For a proof the interested reader is referred to **[20, page 152f]**. □

The core of the above theorem refers to the possibility of approximating a function in the

neighbourhood $x_0 + h$ of a known point x_0, for which the value of the function, but also the values of its derivatives are known.

In the above representation $\mathcal{O}(h^n)$ is an (unknown) error term of little interest in this thesis.

5.1.2 Finite Difference Method

The finite difference method (FDM) is an important tool in numeric analysis for approximating partial differential equations (PDE) with certain (given) initial or boundary conditions.

The approximation is achieved through discretization of the independent variables, i.e. by replacing the definition domain of the independent variables with a finite **grid** (also referred to as **mesh**) of points at which the function, whose dynamics is defined by the PDE, is known.

The FDM approximates all partial derivatives in the PDE at each grid point from values in its neighbourhood by applying Taylor's theorem, however, interpreted in a slightly different manner: in FDM both x_0 and $x_0 + h$ are grid points and both $V(x_0)$ and $V(x_0 + h)$ are known, thus permitting a rearrangement of equation (5.1) to obtain so-called **Finite Difference (FD)** approximations to derivatives with $\mathcal{O}(h^n)$ error terms.

Now, we want to derive some FD approximations to partial derivatives in a PDE with two independent variables x and y. In the following we denote by x_i, $i = 1, \ldots, N$ the grid of discrete x values and by y_j, $j = 1, \ldots, M$ the grid of discrete y values. For the sake of simplicity, we assume $N = M$ and constant grid spacing $\Delta x = \Delta y$, so that $x_{i+1} = x_i + \Delta x$, $\forall i = 1, \ldots, N$ and $y_{j+1} = y_j + \Delta x$, $\forall j = 1, \ldots, N$.

We will approximate the partial derivatives of V with respect to x while holding y constant. Thus, equation (5.1) becomes:

$$V(x_0 + \Delta x, y) = V(x_0, y) + \Delta x V_x(x_0, y) + \frac{(\Delta x)^2}{2!} V_{xx}(x_0, y) + \ldots + \frac{(\Delta x)^{n-1}}{(n-1)!} V_{(n-1)}(x_0, y) + \mathcal{O}\big((\Delta x)^n\big), \tag{5.2}$$

where

$V_x = \frac{\partial V}{\partial x}$,
$V_{xx} = \frac{\partial^2 V}{\partial x^2}$,
$V_{(n-1)} = \frac{\partial^{n-1} V}{\partial x^{n-1}}$.

Cutting the above equation short to $\mathcal{O}\big((\Delta x)^2\big)$ yields

$$V(x_0 + \Delta x, y) = V(x_0, y) + \Delta x V_x(x_0, y) + \mathcal{O}\big((\Delta x)^2\big). \tag{5.3}$$

Rearrangement of equation (5.3) succeeded by division by Δx gives

$$V_x(x_0, y) = \frac{V(x_0 + \Delta x, y) - V(x_0, y)}{\Delta x} - \mathcal{O}(\Delta x), \tag{5.4}$$

which after additionally dropping the error term $\mathcal{O}(\Delta x)$ finally becomes the approximation of the first partial derivative we were looking for:

$$V_x(x_0, y) \approx \frac{V(x_0 + \Delta x, y) - V(x_0, y)}{\Delta x}. \tag{5.5}$$

Equation (5.5) is called the **first order forward difference approximation** to $V_x(x_0, y)$ because of the **first** power of Δx in the error term $\mathcal{O}(\Delta x)$ and since we start at point x_0 and move **forward** to point $x_0 + \Delta x$.

We can also step **backwards** starting from the same point x_0, ultimately giving the **first order backward difference approximation**

$$V_x(x_0, y) \approx \frac{V(x_0, y) - V(x_0 - \Delta x, y)}{\Delta x} \tag{5.6}$$

and we can also get a so-called **second order central difference approximation** by taking more terms in the Taylor series and thus increasing the accuracy of the approximation

$$V_x(x_0, y) \approx \frac{V(x_0 + \Delta x, y) - V(x_0 - \Delta x, y)}{2\Delta x}. \tag{5.7}$$

So far we have shown how to approximate first order partial derivatives. Since most PDEs contain second order (or higher) partial derivatives, we want to be able to approximate them too. In the following we will focus on the approximation of second order unmixed partial derivatives.

Truncating equation (5.2) to $\mathcal{O}\big((\Delta x)^4\big)$ gives:

$$V(x_0 + \Delta x, y) = V(x_0, y) + \Delta x V_x(x_0, y) + \frac{(\Delta x)^2}{2!} V_{xx}(x_0, y) + \frac{(\Delta x)^3}{3!} V_{xxx}(x_0, y) + \mathcal{O}\big((\Delta x)^4\big) \tag{5.8}$$

Replacement of Δx by $-\Delta x$ in the above equation, followed by the addition of both equations gives

$$V(x_0 + \Delta x, y) + V(x_0 - \Delta x, y) = 2V(x_0, y) + (\Delta x)^2 V_{xx}(x_0, y) + \mathcal{O}\big((\Delta x)^4\big), \tag{5.9}$$

which through rearrangement and by dropping the $\mathcal{O}\big((\Delta x)^2\big)$ error term finally provides us with the **second order symmetric difference approximation** to $V_{xx}(x_0, y)$ we were looking for:

$$V_{xx}(x_0, y) \approx \frac{V(x_0 + \Delta x, y) - 2V(x_0, y) + V(x_0 - \Delta x, y)}{(\Delta x)^2}. \tag{5.10}$$

In the following, $V_{i,j}$ will denote the value of the function V at position x_i and position y_j, i.e. $V(x_i, y_j)$, for $i, j = 1, \ldots, N$. Thus, the possible approximations for all partial derivatives up to the order of 2 are given in Table **5.1**.

Partial Derivative	FD-Approximation	Type
V_x	$\frac{V_{i,j}-V_{i-1,j}}{\Delta x}$	backward in x
	$\frac{V_{i+1,j}-V_{i-1,j}}{2\Delta x}$	central in x
	$\frac{V_{i+1,j}-V_{i,j}}{\Delta x}$	forward in x
V_y	$\frac{V_{i,j}-V_{i,j-1}}{\Delta y}$	backward in y
	$\frac{V_{i,j+1}-V_{i,j-1}}{2\Delta y}$	central in y
	$\frac{V_{i,j+1}-V_{i,j}}{\Delta y}$	forward in y
V_{xx}	$\frac{V_{i-1,j}-2V_{i,j}+V_{i+1,j}}{(\Delta x)^2}$	symmetric in x
V_{yy}	$\frac{V_{i,j-1}-2V_{i,j}+V_{i,j+1}}{(\Delta y)^2}$	symmetric in y

Table 5.1: Finite Difference Approximations for Partial Derivatives

By replacing all derivatives in the PDE through their corresponding approximations, we obtain a **Finite Difference Scheme (FDS)** that produces a so-called five-point formula for each $i, j = 2, \ldots, N-1$:

$$V_{i,j} = a_1 V_{i-1,j} + a_2 V_{i,j+1} + a_3 V_{i+1,j} + a_4 V_{i,j-1}, \tag{5.11}$$

(where a_1, a_2, a_3 and a_4 are constants) that can ultimately be reduced to a system of linear equations of the form

$$A\boldsymbol{x} = \mathbf{b}, \tag{5.12}$$

where

A is a $(N-2)(N-2) \times (N-2)(N-2)$ matrix of coefficients,
$\boldsymbol{x}$ is the $(N-2)(N-2)$ vector of unknowns $V_{i,j}$ and
$\mathbf{b}$ is a $(N-2)(N-2)$ vector of constants,

from which the approximate solution can be attained.

The direct way to obtain such a solution is through the classical **Gaussian elimination**. How-

ever, this method is certainly not efficient especially when considering very large systems, which is why so-called **iterative methods** are preferred.

Before presenting some efficient iterative methods, we first need to define the distance between two vectors. Therefore let $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)^T$ and $\boldsymbol{y} = (y_1, y_2, \ldots, y_N)^T$ be two vectors in $\mathbb{R}^N$. The distance, denoted by $d(\boldsymbol{x}, \boldsymbol{y})$, between $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined by:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \max_i(|x_i - y_i|) = \|\boldsymbol{x} - \boldsymbol{y}\|_\infty. \tag{5.13}$$

and is also referred to as "infinity norm".

Jacobi Iteration

In the following the superscript, s, will stand for the iteration index. Thus, the **point-Jacobi formula** for each interior grid point (i, j) at the next iteration $s + 1$ is given by

$$V_{i,j}^{s+1} = a_1 V_{i-1,j}^s + a_2 V_{i,j+1}^s + a_3 V_{i+1,j}^s + a_4 V_{i,j-1}^s. \tag{5.14}$$

At $s = 0$ we only know the values of the function at its boundaries, but initiate all interior grid points to be equal to 0. Once an iteration s has been performed for all (i, j), $i, j = 2, \ldots, N-1$, we save all values in a vector $\boldsymbol{x}^s$ of dimension $(N-2)(N-2)$ and compute the distance between $\boldsymbol{x}^{s+1}$ and $\boldsymbol{x}^s$. If

$$\|\boldsymbol{x}^{s+1} - \boldsymbol{x}^s\|_\infty < \texttt{TOL}, \tag{5.15}$$

where `TOL` stands for a tolerance that is pre-defined, the iterative algorithm stops and the solution is the last computed iterative vector $\boldsymbol{x}^{s+1}$.

Gauss-Seidel Iteration

In the Jacobi Iteration we wait until the next iteration to use the values that we obtain in the current one. This may produce a loss of efficiency since some of the values can be already used during the same iteration. While traversing the grid from left to right starting in the upper corner, upon reaching position (i, j) we have already updated $V_{i-1,j}$ and $V_{i,j+1}$, therefore we can use them and formula (5.14) becomes

$$V_{i,j}^{s+1} = a_1 V_{i-1,j}^{s+1} + a_2 V_{i,j+1}^{s+1} + a_3 V_{i+1,j}^s + a_4 V_{i,j-1}^s, \tag{5.16}$$

which is referred to as the **Gauss-Seidel formula** or **point-Gauss-Seidel**.

The implementation of the Gauss-Seidel Method occurs analogous to that of the Jacobi Iteration.

Successive over Relaxation Method

This method essentially relies on the fact that in an iterative procedure the point value at the new iteration is defined by the point value at the previous iteration plus some residual (or error):

$$V_{i,j}^{s+1} = V_{i,j}^s + \mathcal{R}_{i,j}^s. \tag{5.17}$$

By weighing the residual with a so-called relaxation parameter $w \in (0, 2)$, giving

$$V_{i,j}^{s+1} = V_{i,j}^{s} + w\mathcal{R}_{i,j}^{s}, \tag{5.18}$$

we may speed up the convergence of the iterative scheme. For $w \in (0, 1)$ we speak of under-relaxation whereas for $w \in (1, 2)$ of over-relaxation.

The idea is the improvement of the Gauss-Seidel method as stated in **[19, 47f]**. The point-Gauss-Seidel iteration formula (5.16) can thus be rewritten as

$$V_{i,j}^{s+1} = V_{i,j}^{s} + a_1 V_{i-1,j}^{s+1} + a_2 V_{i,j+1}^{s+1} - V_{i,j}^{s} + a_3 V_{i+1,j}^{s} + a_4 V_{i,j-1}^{s}, \tag{5.19}$$

which yields formula (5.17) with

$$\mathcal{R}_{i,j}^{s} = a_1 V_{i-1,j}^{s+1} + a_2 V_{i,j+1}^{s+1} - V_{i,j}^{s} + a_3 V_{i+1,j}^{s} + a_4 V_{i,j-1}^{s}. \tag{5.20}$$

By additionally introducing the parameter w, we finally obtain

$$V_{i,j}^{s+1} = (1 - w)V_{i,j}^{s} + w(a_1 V_{i-1,j}^{s+1} + a_2 V_{i,j+1}^{s+1} + a_3 V_{i+1,j}^{s} + a_4 V_{i,j-1}^{s}), \tag{5.21}$$

which in the literature is referred to as the **point Successive over Relaxation (SoR) method**.

If $w = 1$, the SoR method obviously reduces to the Gauss-Seidel procedure.

This algorithm is to be implemented in an analogous way to the previously presented iterative methods.

5.2 Monte Carlo Methods

The Monte Carlo Methods were invented after World War II by the mathematician Stanislaw Ulam while working on the construction of the hydrogen bomb at the Los Alamos National Laboratory. He suggested the generation of random numbers through a computer program. Nevertheless, it was John von Neumann who in 1947 recognized their potential and carried out simulations on the ENIAC[1] for estimating neutron collisions. (see **[21, page 89]**)

Nicholas Metropolis named this technique **Monte Carlo** after the city from Monaco, where Ulam's uncle gambled in numerous casinos, the place where random numbers can be "naturally" generated for example by playing roulette. Monte Carlo methods are used in various fields from physics to industrial engineering, computer science, insurances and last but not least finance. (see **[21, page 89]**, **[21, Introduction]**)

The simulation is nothing but a technique to mathematically model a real, existing system. The mathematical model is defined by a series of equations and functional connections that are meant to describe the interactions between the components of the system. The behaviour of the **real** system is therefore being **virtually** imitated by creating and running a computer program based on numerically generated data. (see **[21, Introduction]**)

The modelling of uncertainty in complex systems, given by the generation of random numbers, is provided by probability theory. However, Monte Carlo methods are often considered

[1] Electronic Numerical Integrator And Computer was the first electronic computer ever built in the United States of America

to be the interface between the notion of probability and that of volume (see **[22, page 1]**) because of measure theory, which associates an event with a set of possible outcomes to a probability and defines the latter as the volume (or measure) of the respective event relative to that of an infinite number of possible outcomes. Therefore, Monte Carlo estimates the volume of a set by understanding it as a probability.

In the following we want to present two fundamental limit theorems in probability theory, that lie at the core of the Monte Carlo method. The interested reader is referred to **[23, page 85]** and **[24, page 178]** for rigorous proofs.

Theorem 5.2.1 (Strong Law of Large Numbers) [3, page 24]

If $X_1, X_2, \ldots$ are independent, identically distributed (iid) random variables with mean μ, then

$$\mathbb{P}\left\{\frac{X_1 + X_2 + \ldots + X_n}{n} \to \mu\right\} = 1. \tag{5.22}$$

Theorem 5.2.2 (Central Limit Theorem - Finite Variance) [3, page 24]

If $X_1, X_2, \ldots$ are iid random variables with mean μ and variance σ^2, then the distribution of

$$\frac{X_1 + X_2 + \ldots + X_n - n\mu}{n} \tag{5.23}$$

converges to the standard normal distribution. That is, for any $a \in \mathbb{R}$,

$$\mathbb{P}\left\{\frac{X_1 + X_2 + \ldots + X_n - n\mu}{\sigma\sqrt{n}} \leq a\right\} \to \int_{-\infty}^{a} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} dx \tag{5.24}$$

as $n \to \infty$.

5.2.1 Euler Scheme

Usually, stochastic differential equations are very difficult if not impossible to solve analytically, which is why one often resorts to Monte Carlo simulation.

Let us reconsider a stochastic differential equation as presented in Section 2.5 and Section 3.2, that is, according to **[22, page 339f]** and **[3, page 205f]**:

$$dX_t = b(t, X_t)dt + \sigma(t, X_t)dW_t, \tag{5.25}$$

with a fixed value for $X(0)$ and where $b(t, x)$ and $\sigma(t, x)$ are two continuous functions.

Furthermore, let us suppose we want to generate a sample path of X. This is where the **Euler Scheme** or **Euler Approximation** comes in for in order to do that, we first need to choose a finite interval $[0, T]$ and define a time discretization of the sort $0 = t_0 < t_1 < t_2 < \ldots < t_n = T$.

Next, we recursively obtain an approximation $\widehat{X}$ for process X as follows:

$$\widehat{X}(0) = X(0), \tag{5.26}$$

$$\widehat{X}(t_{i+1}) = \widehat{X}(t_i) + b(t_i, \widehat{X}(t_i))(t_{i+1} - t_i) + \sigma(t_i, \widehat{X}(t_i))(W(t_{i+1}) - W(t_i)) \tag{5.27}$$

$$= \widehat{X}(t_i) + b(t_i, \widehat{X}(t_i))(t_{i+1} - t_i) + \sigma(t_i, \widehat{X}(t_i))\sqrt{t_{i+1} - t_i}Z_{i+1}, \tag{5.28}$$

for $i = 0, \ldots, n-1$, where $Z_1, Z_2, \ldots, Z_n$ are independent, identically distributed standard normal random variables.

The Monte Carlo algorithm consists in generating N such sample paths in order to obtain N iid copies of X (denoted as $\widehat{X}_1, \widehat{X}_2, \ldots, \widehat{X}_N$), its Monte Carlo estimate therefore being:

$$\tilde{X} = \frac{\widehat{X}_1 + \widehat{X}_2 + \ldots + \widehat{X}_N}{N}. \tag{5.29}$$

See **[3, page 73f]**, **[3, page 205f]**.

Chapter 6

A Stochastic Control Model

This mathematical model was originally formulated by David Aldous (see [29] and [30]).

6.1 A Theoretical Overview

Suppose N particles perform independent Brownian motion on $[0, \infty)$ starting at initial state 1 at time 0. Furthermore, assume that these particles are killed upon reaching state 0, making state 0 absorbing. We call this the **uncontrolled process** and denote it as $(X_i(t), 0 \leq t < \infty, 1 \leq i \leq N)$.

Since we have promised a control model, we want to transform the above problem into a process that we can control. We achieve this by additionally assuming that we have at our disposal a unit of positive drift, which we can distribute amongst surviving particles at time t according to any control policy or strategy we choose.

Let this **controlled process** be specified by $((X_i^c(t), \mu_i(t)), 0 \leq t < \infty, 1 \leq i \leq N)$, with drift terms μ_i satisfying:

$$\mu_i(t) \geq 0, \tag{6.1}$$

$$\sum_i \mu_i(t) \leq 1, \tag{6.2}$$

$$\mu_i(t) = 0 \quad \text{and} \quad X_i^c(t) = 0, \quad \text{if} \quad \inf_{s \leq t} X_i^c(s) = 0. \tag{6.3}$$

Define S as the number of particles that survive forever:

$$S := \#\{i : X_i^c(t) > 0, \forall t\}. \tag{6.4}$$

As a next step, we want to maximize the expected number of particles that never get absorbed, i.e. $\max_{strategy} \mathbb{E}[S]$. Thus, two major questions arise:

1. **Which strategy maximizes $\mathbb{E}[S]$?**
2. **What is the resulting value, denoted by $V(N)$, of $\mathbb{E}[S]$?**

In this section, we will deal with both these questions, trying to find appropriate and perfectly justified answers.

Surprisingly enough, we are able to determine the range of $V(N)$ without having to identify and analyze the optimal strategy, as the following lemma shows.

Lemma 6.1.1 *Let $N \geq 1$ particles perform independent Brownian motion $(W_t)_{t\geq 0}$ as described above. Then*

$$\max_{strategy} \quad \mathbb{E}\big[S\big] \in \big[c_1 N^{1/2}, c_2 N^{1/2}\big] \quad as \quad N \to \infty, \tag{6.5}$$

where c_1 and c_2 are constants.

Proof:

This proof is an extended version of the proof given in [29].

1. **Lower Bound**

   Choose $1 \leq n \leq N$ and consider the following (permitted) strategy: assign drift $\frac{1}{n}$ to each particle only after reaching position $\frac{N}{n}$ and if it is one of the first n particles to do so.

   From the literature on Brownian motion we know that:

   $$\mathbb{P}\Big((W_t)_{t\geq 0}, \quad \mu = 0, \quad \text{hits} \quad \frac{N}{n} \quad \text{before hitting} \quad 0\Big) = \frac{n}{N}, \tag{6.6}$$

   $$\mathbb{P}\Big((W_t)_{t\geq 0}, \quad \mu > 0, \quad \text{hits} \quad 0 \quad | \quad W_0 = x\Big) = \exp(-2\mu x). \tag{6.7}$$

   Thus (with $\mu = \frac{1}{n}$ and $x = \frac{N}{n}$) it follows that:

   $$\mathbb{E}\big[S\big] = \Big(1 - \exp\Big(-\frac{2N}{n^2}\Big)\Big) \cdot \mathbb{E}\big[\min(n, R)\big], \tag{6.8}$$

   where R denotes a binomial random variable $\Big(R \sim \text{bin}\Big(N, \frac{n}{N}\Big)\Big)$.

   Letting $N \to \infty$ together with the assumption that $n \sim \theta N^{\frac{1}{2}}$ gives:

   $$\mathbb{E}\big[S\big] \approx \Big(1 - \exp\Big(-\frac{2}{\theta^2}\Big)\Big)\theta N^{\frac{1}{2}}. \tag{6.9}$$

   As Figure 6.1 shows, function $\theta \to (1 - e^{-\frac{2}{\theta^2}})\theta$ takes its maximum value (≈ 0.9) at $\theta \approx 1.25$ finally giving the lower bound of the form $c_1 N^{\frac{1}{2}}$.

2. **Upper Bound**

   Consider an arbitrary strategy and two types of Brownian motion (BM): with and without any drift μ.

   Define:

   $$f_t(x) := \mathbb{P}\Big(\inf_{0\leq s\leq t} \quad \text{BM}_s > 0 \quad | \quad \text{BM}_0 = x\Big), \quad x \geq 0 \quad \text{and} \tag{6.10}$$

   $$Y_t := \sum_i \quad f_{\tau - t}(P_i(t)), \quad 0 \leq t \leq \tau, \tag{6.11}$$

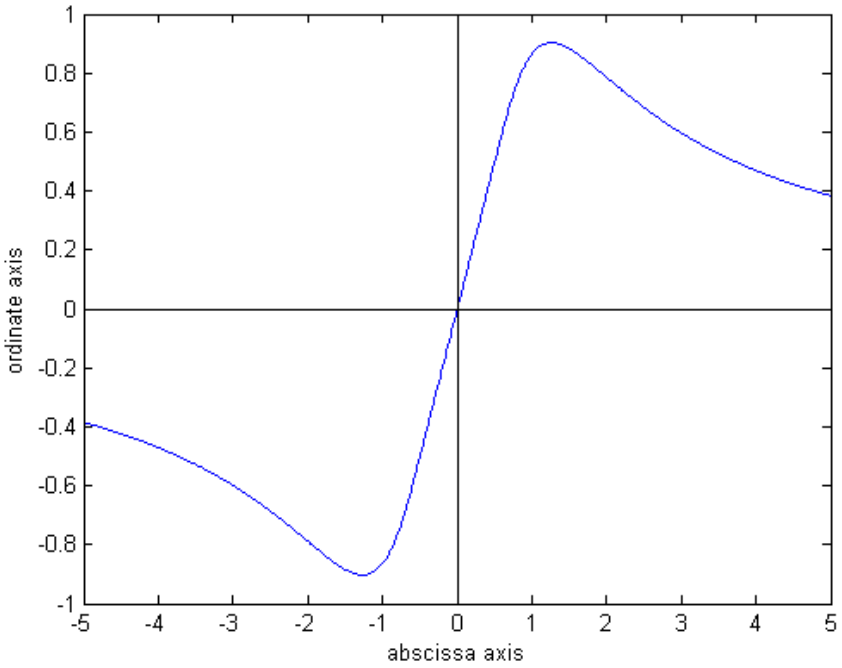

Figure 6.1: Graph of function $f(\theta) = (1 - e^{-\frac{2}{\theta^2}})\theta$

for some fix $\tau > 0$, where $P_i(t)$ denotes a (controlled or uncontrolled) process for particle i at time t.

First, let us consider the uncontrolled process $(X_i(t), 0 \leq t < \infty, 1 \leq i \leq N)$: in this case, Y_t is a martingale.

On the other hand, for the controlled process $((X_i^c(t), \mu_i(t)), 0 \leq t < \infty, 1 \leq i \leq N)$ we get:

$$dY_t = dM_t + \sum_i \mu_i(t) g_{\tau-t}(X_i^c(t)), \tag{6.12}$$

where M is some martingale and $g_t(x) := \frac{d}{dx} f_t(x)$, since $df_{\tau-t}(W_t) = dM_t + \mu g_{\tau-t}(W_t)$ for a Brownian motion W_t with drift μ.

In the above representation, g_t is the density function of the absolute value of a normally distributed random variable with mean 0 and variance t:

$$f(x) = \frac{1}{\sqrt{2\pi t}} e^{-\frac{x^2}{2t}}, \quad X \sim \mathcal{N}(0, t) \tag{6.13}$$

$$f(x) = \sqrt{\frac{2}{\pi t}} e^{-\frac{x^2}{2t}}, \quad |X| \quad \text{half normally distributed with} \quad \mu = 0 \quad \text{and} \quad \sigma^2 = t. \tag{6.14}$$

It is a known fact that the maximum of the latter density function is attained at $x = 0$, giving maximum value $\sqrt{\frac{2}{\pi t}}$ and thus implying

$$dY_t \leq dM_t + \sqrt{\frac{2}{\pi(\tau - t)}}. \tag{6.15}$$

Furthermore, integrating over $0 \leq t \leq \tau$ finally gives

$$Y_\tau - Y_0 \leq M_\tau - M_0 + \int_0^\tau \sqrt{\frac{2}{\pi(\tau - t)}} dt. \tag{6.16}$$

Expectation of the above expression yields

$$\mathbb{E}\big[Y_\tau\big] \leq \mathbb{E}\big[Y_0\big] + \frac{2\sqrt{2}}{\sqrt{\pi}} \tau^{\frac{1}{2}}. \tag{6.17}$$

Now we only need an approximation for Y_0:

$$Y_0 = N f_\tau(1) \leq N \sqrt{\frac{2}{\pi\tau}}. \tag{6.18}$$

Finally, consider Y_τ to be the number of particles that survive until τ:

$$Y_\tau = S_\tau := \#\{i : X_i^c(\tau) > 0\}. \tag{6.19}$$

Then we have

$$\mathbb{E}\big[S_\tau\big] \leq N \sqrt{\frac{2}{\pi}} \tau^{-\frac{1}{2}} + \frac{2\sqrt{2}}{\sqrt{\pi}} \tau^{\frac{1}{2}}. \tag{6.20}$$

Since

$$\mathbb{E}\big[S\big] \leq \mathbb{E}\big[S_\tau\big], \tag{6.21}$$

we actually have

$$\mathbb{E}\big[S\big] \leq N \sqrt{\frac{2}{\pi}} \tau^{-\frac{1}{2}} + \frac{2\sqrt{2}}{\sqrt{\pi}} \tau^{\frac{1}{2}}. \tag{6.22}$$

As Figure 6.2 shows, the right side of (6.22) attains a minimum at $\tau = \frac{N}{2}$ with value $\frac{4}{\sqrt{\pi}} N^{\frac{1}{2}}$, thus giving an upper bound of the form $c_2 N^{\frac{1}{2}}$ and terminating the proof. □

Having provided an answer for question **2.** in the form of an interval, it is now time to dedicate ourselves to finding a solution to question **1.**.

As we have seen in Section 2.2, one of the properties of Brownian motion is the so-called **scaling**. Thus, for a Brownian motion W_t on $(-\infty, +\infty)$, we know we can for example rescale time by N and space by $N^{\frac{1}{2}}$ and still get back Brownian motion.

Therefore, in the following replace $W(t)$ by $N^{-\frac{1}{2}} W(tN)$.

As a result of Lemma 6.1.1, we know that the limit:

$$\lim_{N \to \infty} N^{-\frac{1}{2}} \mathbb{E}\big[S\big], \tag{6.23}$$

should exist.

First, we want to consider the uncontrolled process and afterwards the what we consider to be optimally controlled process.

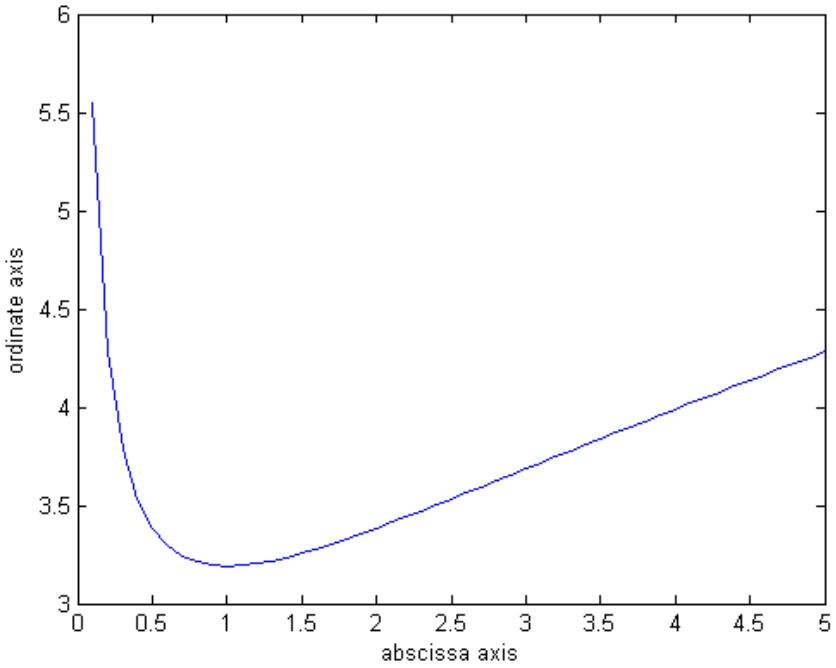

Figure 6.2: Graph of function $f(\tau) = N\sqrt{\frac{2}{\pi}}\tau^{-\frac{1}{2}} + \frac{2\sqrt{2}}{\sqrt{\pi}}\tau^{\frac{1}{2}}$ for $N = 2$

The uncontrolled process

Initially, consider the setting without any drift, that is let N independent Brownian particles start at 1 and kill them upon reaching 0. Moreover, assign "mass" $N^{-\frac{1}{2}}$ to each particle.

Under the Brownian scaling mentioned earlier, we are able to deduce the following so-called "fluid limit" result:

$$N^{-\frac{1}{2}}\#\{i : X_i(tN) \leq yN^{\frac{1}{2}}\} \to_p \int_0^y f(t,x)dx, \quad \forall y \geq 0, \forall t > 0 \quad \text{as} \quad N \to \infty, \tag{6.24}$$

where $f(t,x)$ is a "mass density" at time t, that satisfies the heat equation

$$\frac{d}{dt}f = \frac{1}{2}\frac{d^2}{dx^2}f, \tag{6.25}$$

with (absorbing) boundary condition

$$f(t,0) = 0. \tag{6.26}$$

The optimally controlled process

Now consider the following control policy: **assign drift** 1 **to the lowest particle**.

Intuitively, we assess that the controlled process has a similar behaviour to the uncontrolled

one, i.e. there also exists a fluid limit, however for a different mass density function $f^c(t,x)$:

$$N^{-\frac{1}{2}}\#\{i : X_i^c(tN) \leq yN^{\frac{1}{2}}\} \to_p \int_0^y f^c(t,x)dx, \quad \forall y \geq 0, \forall t > 0 \quad \text{as} \quad N \to \infty, \tag{6.27}$$

In addition, we also conjecture that this density function $f^c(t,x)$ should also solve the heat equation, however with a different boundary condition and it should also exhibit similar t-behaviour as the density function of the uncontrolled process:

$$f(t,x) \sim f^c(t,x) \quad \text{as} \quad t \to 0 \quad \text{with} \quad x \sim \theta t^{\frac{1}{2}}, \quad \theta > 0. \tag{6.28}$$

As mentioned before, $f^c(t,x)$ satisfies the heat equation but for another boundary condition, which is obviously the effect of the control policy. Therefore the boundary, denoted by $b(t)$, changes in the following sense:

$$b(t) = 0, \quad 0 \leq t \leq t_0, \tag{6.29}$$
$$b(t) > 0, \quad t > t_0. \tag{6.30}$$

leading to the boundary condition

$$f(t, b(t)) = c, \quad 0 \leq t < \infty. \tag{6.31}$$

For some t_0, $b(t)$ has a unique form given by (6.29) and (6.30), yielding a unique density function $f^c(t,x)$, defined on the domain $D := \{(t,x) : t \geq 0, b(t) \leq x\}$, such that it satisfies:

- the heat equation on D°;
- the boundary condition (6.31);
- the extra boundary condition $\frac{d}{dx}f^c(t,x)|_{x=b(t)} = 0$ for $t > t_0$ and
- the fluid limit of the controlled process.

The above statement implies that for $t \leq t_0 N$, the position of the left-most particle, denoted by $L(t)$, will be close to 0, (relative to the previously described Brownian scaling), but not greater than 0, as it previously has been, when choosing an arbitrary control policy (see proof of Lemma **6.1.1**, upper bound).

Thus, instead of the inequality (6.20), we obtain an approximate equality:

$$\mathbb{E}\big[S_\tau\big] \approx N\sqrt{\frac{2}{\pi}}\tau^{-\frac{1}{2}} + \frac{2\sqrt{2}}{\sqrt{\pi}}\tau^{\frac{1}{2}}, \quad \tau \leq t_0 N. \tag{6.32}$$

From the previously asserted behaviour of the boundary $b(t)$, it results moreover that the number of particles killed after $t_0 N$ is negligible, thus implying yet another approximation:

$$\mathbb{E}\big[S\big] \approx \mathbb{E}\big[S_{t_0 N}\big]. \tag{6.33}$$

From the proof of Lemma **6.1.1**, we know that the right side of (6.32) is minimized at $\tau = \frac{N}{2}$, yielding minimum value $\frac{4}{\sqrt{\pi}}N^{\frac{1}{2}}$. We speculate that $t_0 = \frac{1}{2}$ for the following 2 reasons:

1. t_0 **cannot be greater than** $\frac{1}{2}$, since $\mathbb{E}\big[S_\tau\big]$ is a priori decreasing, attains minimum value at $\tau = \frac{1}{2}N$ and the approximation (6.32) holds;

2. t_0 **cannot be less than** $\frac{1}{2}$, because $\mathbb{E}\big[S\big] \leq \mathbb{E}\big[S_{\frac{1}{2}N}\big]$ and (6.33) holds.

To conclude, when choosing policy **"apply drift 1 to the lowest particle"**, we get the maximum value of the previously defined interval for $N \to \infty$:

$$\lim_{N\to\infty} N^{-\frac{1}{2}}\mathbb{E}\big[S\big] = \frac{4}{\sqrt{\pi}}. \tag{6.34}$$

6.2 A Specific Two-Dimensional Model

6.2.1 The Real World Setting

As we all know, politics plays a major role in the economy of any country in this world. From a position of power, such as for example the government's, one can influence the development and evolution of the economy for the better or the worse, through the policies that one chooses.

Suppose the government of any given country is intent on giving tax breaks to companies. Furthermore assume that it has two possible policies at its disposal:

1. **Democratic Policy:** tax breaks to **weak** companies;
2. **Republican Policy:** tax breaks to **rich** companies.

Which strategy is better for the entire economy of that country: to support weak companies in the hopes of keeping them alive and thus reduce unemployment, or aiding rich companies in expanding and becoming even richer? What does "better for the entire economy" actually mean?

Before producing an answer to the above questions, we translate the above problem into a (corresponding) mathematical model.

6.2.2 The Mathematics Behind the Setting

Suppose the above setting only applies to two companies. The following model was first formulated by David Aldous (see [29] and [30]).

Let $W_i(t)$, $i = 1, 2$ be *independent* Brownian motions, that become controllable by adding drifts $\mu_i(t)$ with unit sum. Thus the **controlled processes** satisfy for $i = 1, 2$ and $t \geq 0$:

$$dX_i(t) = \mu_i(t)dt + dW_i(t), \quad X_i(0) = x_i, \tag{6.35}$$

$$\mu_i(t) \geq -\delta^{1}, \quad \mu_1 + \mu_2 = 1, \quad -\frac{1}{2} \leq \delta < \infty, \tag{6.36}$$

where x_i and δ are given.

[1] David Aldous formulated the model with $\mu_i \geq 0$. This version, suggested by Oded Palmon, generalizes the model by assuming that money can be given to the poor after being "stolen" from the rich.

For every $t \geq 0$ we are allowed to choose the drifts $\mu_i(t)$ *instantaneously*, however without any prior knowledge about the future increments of the independent standard Brownian motions $W_i(t)$. If $X_i(t)$ hits 0 for some finite value of t, it is assumed that company i goes bankrupt and disappears from the market.

Let $S = \{0, 1, 2\}$ denote the set of possible numbers of companies that never go bankrupt, i.e. that survive forever and s be some element of this set. Having the above processes in mind, there are two possible **optimization criteria** that one can take into consideration:

1. maximize the **probability** that both companies survive forever:

$$\max \quad \mathbb{P}(s = 2), \tag{6.37}$$

2. maximize the **expected number** of companies that survive forever:

$$\max \quad 1 \cdot \mathbb{P}(s = 1) + 2 \cdot \mathbb{P}(s = 2). \tag{6.38}$$

Through the insertion of a parameter, denoted by α, one can easily melt these two optimization criteria into a single one:

$$\max \quad \alpha \cdot \mathbb{P}(s = 1) + (1 - \alpha) \cdot \mathbb{P}(s = 2), \quad \alpha \in \left[0, \frac{1}{2}\right]. \tag{6.39}$$

Thus, when $\alpha = 0$, the optimization problem reduces to the first optimization criterion, while for any $\alpha \neq 0$ in the previously mentioned interval, the problem becomes an equivalent of the second optimization criterion.

In order to maximize the above problem, the decision-maker has to choose between two distinct strategies:

1. **Push-Bottom Strategy or Democratic Policy**: assign drift $\mu = 1$ to the poorer company;
2. **Push-Top Strategy or Republican Policy**: assign drift $\mu = 1$ to the richer company.

The problem is more difficult for dimensions greater than two and for unequal diffusion constants of the two Brownian motions.

6.2.3 The Difficulties of the Model

John von Neumann's and Oskar Morgenstern's "Theory of Games and Economic Behavior" (see **[15]**) is thought to be the foundation of stochastic optimization theory with applications to finance and economics. It later sparked the debut of the **theory of linear programming** and initiated problems of optimal control and optimal stopping times as they are known today.

An early example of such a (one-dimensional) problem, formulated by Paul Samuelson and solved by Henry McKean is pricing the perpetual American option (see **[16]**). Through the **principle of smooth fit** (see **[17, pages 1-6]**) solutions to such problems may be guessed, thus avoiding having to solve various nonlinear differential equations one is inevitably faced with

while applying the dynamic programming approach.

After a brief introduction to the evolution of stochastic optimization theory and some alleged solving techniques, the time has come to present the predicaments of the above model, as stated by the authors (Henry McKean and Larry Shepp) of [18] themselves:

1. **Too much discreteness**
   One of the problems with the presented model is that it is too discrete and that it does not involve Itô calculus, which makes it difficult to find an optimal solution.

2. **Dimensionality higher than 1**
   The above model involves two independent Brownian motions, unlike previously solved problems in finance and economics, that only considered a single Brownian motion. Whereas the principle of smooth fit allows the solution to be guessed for the latter, the solution to the former problem breaks new ground.

3. **Lack of explicit solution for proof**
   If the optimal control strategy for the above model can be somehow guessed and the corresponding value of optimal payoff be found, then proving that the guessed solution is equivalent to the optimum is trivial.

   On the other hand, an existence proof alone, without having found an explicit solution, is considered to be insufficient and unsatisfactory, since the supermartingale inequalities can never be verified.

   Unfortunately, an explicit solution to the above model could only be guessed for the exclusive parameters $\alpha = 0$ and $\delta = 0$.

6.2.4 A Guessed Optimal Solution

"With great luck and also great difficulty", the authors of [18] have guessed the optimal payoff function V, but only for the case $\delta = 0$ and $\alpha = 0$ with the democratic policy (also referred to as "Robin Hood" policy) as optimal control strategy. In the following, we will show that the push-bottom strategy is optimal.

Let

$$V(x_1, x_2) = 1 - e^{-2\min(x_1,x_2)} - 2\min(x_1, x_2)e^{-(x_1+x_2)} \tag{6.40}$$

denote the probability that both companies survive forever, π stand for the policy of choice (either pushing the bottom or the top company) and $V^\pi(x_1, x_2)$ be the probability that both companies survive forever under policy π.

Show that:

$$V^\pi(x_1, x_2) \leq V(x_1, x_2), \tag{6.41}$$

where V denotes the probability that both companies survive when pushing the bottom company.

We know that the process $P(t) = V(X_1^\pi, X_2^\pi)$ is a supermartingale if and only if its Itô differential satisfies the following inequation:

$$\mathbb{E}\big[dP(t)\big] = \max(V_{x_1}, V_{x_2}) + \frac{1}{2}(V_{x_1x_1} + V_{x_2x_2}) \leq 0, \tag{6.42}$$

for all (x_1, x_2) in the first quadrant, where $V \in C^1$ and $V \in C^2$ for $x_1 = x_2$, i.e on the diagonal.

Process $S(t) = V(x_1, x_2)$ has drift:

$$-2e^{-2x_1} + 2e^{-(x_1+x_2)} - 2x_1 e^{-(x_1+x_2)} \leq 0, \quad x_1 \leq x_2, \tag{6.43}$$

thus giving

$$\mathbb{E}\big[S(\infty)\big] \leq S(0) = V(x_1, x_2). \tag{6.44}$$

However, at $t = \infty$ the following cases can occur:

- both companies have survived, yielding $S(\infty) = 1$;
- only one company has survived and the other one has gone broke, giving $S(\infty) = 0$.

Hence, for any policy π, the supermartingale inequality assures that the probability that both companies survive is no more than $V(x_1, x_2)$.

Since $V(x_1, x_2)$ is per construction bounded, we achieve equality throughout the proof, for the sole policy "pushing the bottom company", finally showing that V gives the value of the optimal payoff when de facto choosing the optimal control strategy.

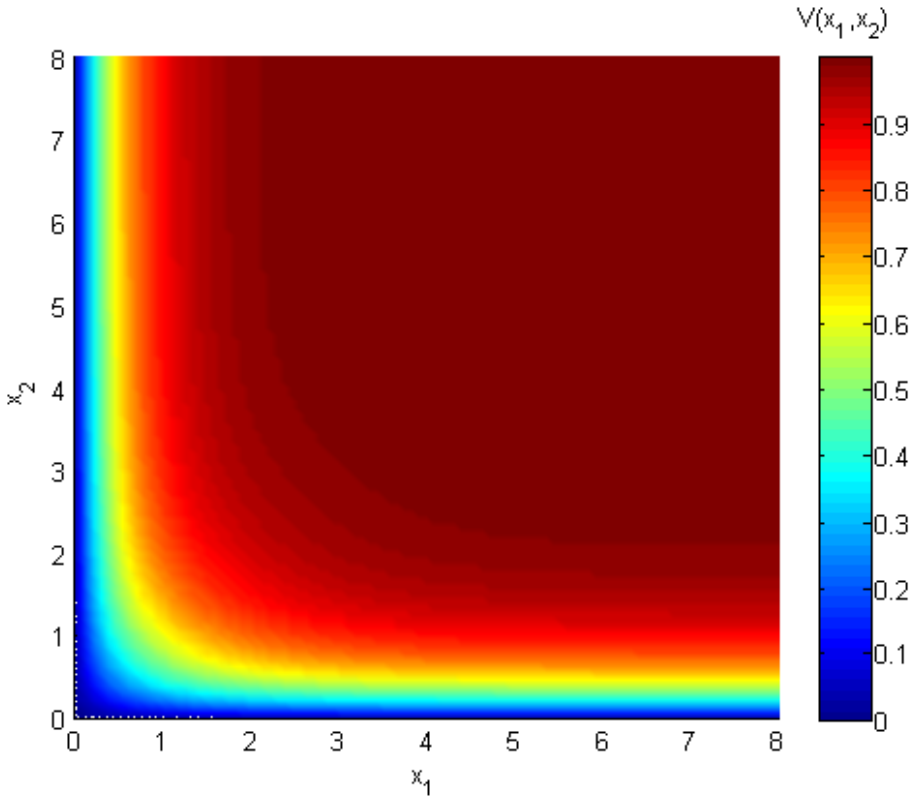

Figure 6.3: Surface plot of function V, for $\alpha = 0$

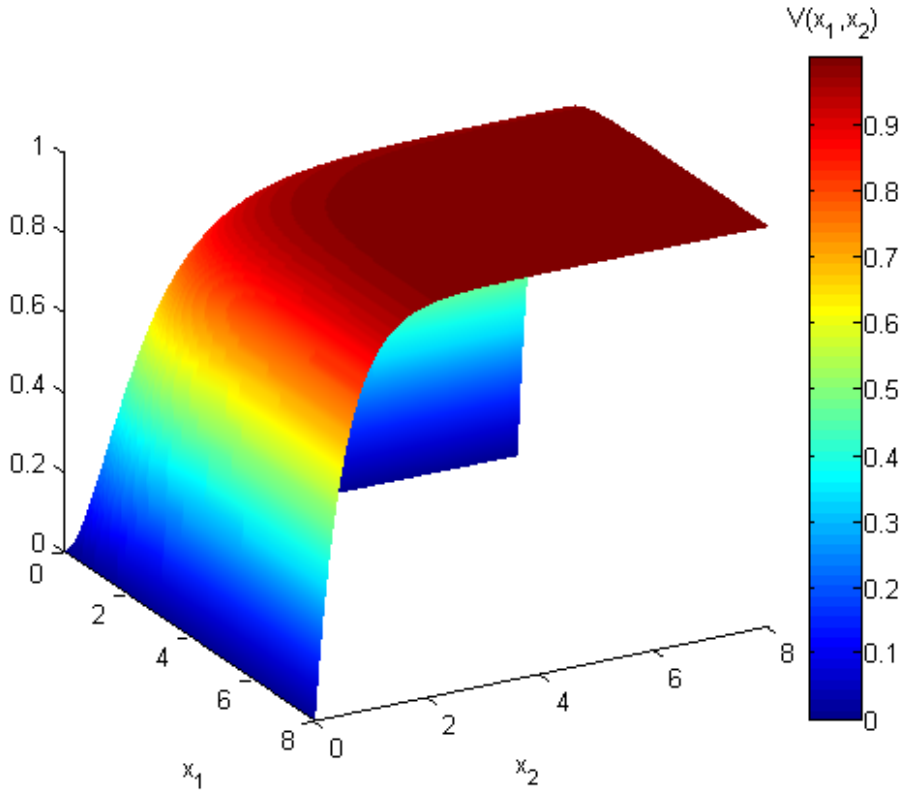

Figure 6.4: 3D plot of function V, for $\alpha = 0$

Chapter 7

Proposed Solutions

The authors of [18] suggest a numeric approximation for the solution of the model presented in the previous chapter.

Suppose we want to maximize the generalized optimization criterion over all nonanticipating policies π:

$$\max_{\pi} \quad \alpha \mathbb{P}(s=1) + (1-\alpha)\mathbb{P}(s=2). \tag{7.1}$$

7.1 Numerical Approximation

In the next step we want to obtain an approximate optimal payoff function by numerically solving the following partial differential equation (PDE) with boundary conditions (BC):

$$\text{PDE:} \quad \frac{1}{2}(V_{x_1x_1} + V_{x_2x_2}) + \max(V_{x_1}, V_{x_2}) = 0, \tag{7.2}$$

BC: along the axes / at infinity

$$V(x_1, 0) = \alpha(1 - e^{-2x_1}), \qquad V(x_1, \infty) = \alpha e^{-2x_1} + (1-\alpha)(1 - e^{-2x_1}), \tag{7.3}$$

$$V(0, x_2) = \alpha(1 - e^{-2x_2}), \qquad V(\infty, x_2) = \alpha e^{-2x_2} + (1-\alpha)(1 - e^{-2x_2}), \tag{7.4}$$

resulting from the fact that Brownian motion with $\mu = 1$ and $\sigma = 1$ starting at x_1 has, as already known, probability $1 - e^{-2x_1}$ of never hitting 0.

However, since the presence of ∞ is troublsome to say the least, satisfactory results can be obtained through discretization, i.e. by replacing ∞ with finite values $x_{1_{\max}} = x_{2_{\max}} = \text{const}$ and using discrete instead of continuous data.

Thus we obtain the following:

$$\text{PDE:} \quad \frac{1}{2}(V_{x_1x_1} + V_{x_2x_2}) + \max(V_{x_1}, V_{x_2}) = 0,$$

BC: along the axes at $x_{1_{\max}} = x_{2_{\max}} = \text{const}$

$$V(x_1, 0) = \alpha\frac{1 - e^{-2x_1}}{1 - e^{-2x_{1\max}}}, \qquad V(x_1, x_{2_{\max}}) = \alpha\Big(1 - \frac{e^{-2x_1}}{1 - e^{-2x_{1\max}}}\Big) + (1 - \alpha)\Big(\frac{e^{-2x_1}}{1 - e^{-2x_{1\max}}}\Big),$$

$$V(0, x_2) = \alpha\frac{1 - e^{-2x_2}}{1 - e^{-2x_{2\max}}}, \qquad V(x_{1_{\max}}, x_2) = \alpha\Big(1 - \frac{e^{-2x_2}}{1 - e^{-2x_{2\max}}}\Big) + (1 - \alpha)\Big(\frac{e^{-2x_2}}{1 - e^{-2x_{2\max}}}\Big),$$

where

$$s(x_1) := \mathbb{P}(X_1(\tau) = x_{1_{\max}}) = \frac{1 - e^{-2x_1}}{1 - e^{-2x_{1\max}}} = V(x_1, 0), \tag{7.5}$$

if

$$X_1(t) = t + W(t) \tag{7.6}$$

and

$$\tau := \inf\{t | X_1(t) \notin [0, x_{1_{\max}}]\}. \tag{7.7}$$

Figure 7.1 portrays the boundary conditions of the above PDE. Since the function V is sym-

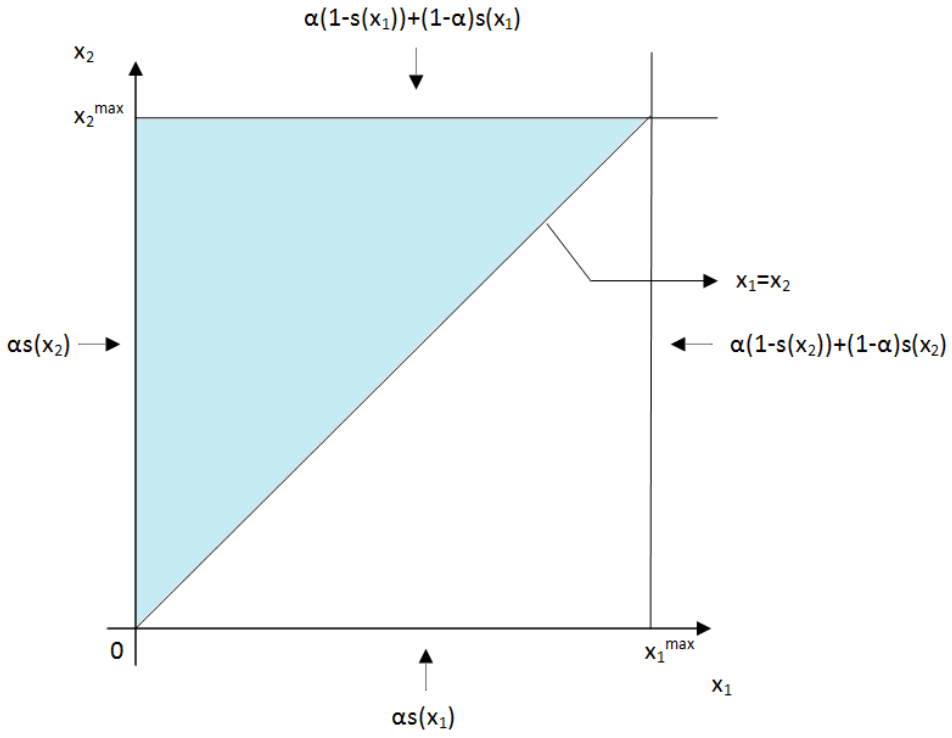

Figure 7.1: Boundary conditions of the discretized model

metric about the second diagonal, it suffices to only consider for example the shaded area, i.e $x_1 \leq x_2$.

Next, we consider two cases corresponding to the democratic and the republican policy respectively.

1. **Democratic Policy (Push-Bottom Strategy)**

   In this case, the PDE has the following form:

$$\frac{1}{2}(V_{x_1x_1} + V_{x_2x_2}) + V_{x_1} = 0, \quad \text{for} \quad x_1 \leq x_2 \tag{7.8}$$

   and the BC remain unchanged.

   By replacing all partial derivatives with their corresponding Finite Difference Approximations (recall Section **5.1**, particularly Table **5.1**) and specifically choosing the central order difference approximation to compute V_{x_1} and V_{x_2}[1], we obtain the following equation:

$$\frac{V_{i+1,j} - V_{i-1,j}}{2\Delta x_1} + \frac{1}{2}\frac{V_{i+1,j} - 2V_{i,j} + V_{i-1,j}}{(\Delta x_1)^2} + \frac{V_{i,j+1} - 2V_{i,j} + V_{i,j-1}}{(\Delta x_2)^2} = 0, \tag{7.9}$$

   which, after reducing to a common denominator and sorting out the $V_{i,j}$ terms, yields:

$$V_{i,j} = \frac{(1+\Delta x_1)(\Delta x_2)^2 V_{i+1,j} + (1-\Delta x_1)(\Delta x_2)^2 V_{i-1,j} + (\Delta x_1)^2(V_{i,j+1} + V_{i,j-1})}{2\big((\Delta x_1)^2 + (\Delta x_2)^2\big)}. \tag{7.10}$$

   Division of both the nominator and the denominator by $(\Delta x_2)^2$ and insertion of $c := \frac{\Delta x_1}{\Delta x_2}$ gives

$$V_{i,j} = \frac{(1+\Delta x_1)V_{i+1,j} + (1-\Delta x_1)V_{i-1,j} + c^2(V_{i,j+1} + V_{i,j-1})}{2(c^2+1)}. \tag{7.11}$$

   Finally, we consider discrete, equidistant variables, translating into $\Delta x_1 = \Delta x_2$, which is equivalent to $c = 1$ and which produces the following Finite Difference Scheme:

$$V_{i,j} = \frac{(1+\Delta x_1)V_{i+1,j} + (1-\Delta x_1)V_{i-1,j} + V_{i,j+1} + V_{i,j-1}}{4}. \tag{7.12}$$

   - **Jacobi Iteration**

     Solving equation (7.12) through Jacobi Iteration implies initializing all values $V_{i,j}$ at 0 for all interior grid points for the first iteration at $s = 0$. The formula then transforms into

$$V_{i,j}^{s+1} = \frac{(1+\Delta x_1)V_{i+1,j}^{s} + (1-\Delta x_1)V_{i-1,j}^{s} + V_{i,j+1}^{s} + V_{i,j-1}^{s}}{4}.$$

   - **Successive over Relaxation**

     As mentioned before, this method allows us to use in the current iteration already approximated values in the same iteration, thus modifying equation (7.12) in the following way:

$$V_{i,j}^{s+1} = (1-w)V_{i,j}^{s} + w\frac{(1+\Delta x_1)V_{i+1,j}^{s} + (1-\Delta x_1)V_{i-1,j}^{s+1} + V_{i,j+1}^{s+1} + V_{i,j-1}^{s}}{4}.$$

     Setting $w = 1$ gives the **Gauss-Seidel formula** for equation (7.12).

[1] x_1 obviously stands for x and x_2 for y

2. **Republican Policy (Push-Top Strategy)**

   In this case, the PDE has the following form:

   $$\frac{1}{2}(V_{x_1x_1} + V_{x_2x_2}) + V_{x_2} = 0, \quad \text{for} \quad x_1 \leq x_2 \tag{7.13}$$

   and the BC remain unchanged.

   Through similar calculations, an analogous FDS is obtained for this second case:

   $$V_{i,j} = \frac{(1+\Delta x_2)V_{i,j+1} + (1-\Delta x_2)V_{i,j-1} + V_{i+1,j} + V_{i-1,j}}{4}. \tag{7.14}$$

   - **Jacobi Iteration**

     In this case the formula is:

     $$V_{i,j}^{s+1} = \frac{(1+\Delta x_2)V_{i,j+1}^{s} + (1-\Delta x_2)V_{i,j-1}^{s} + V_{i+1,j}^{s} + V_{i-1,j}^{s}}{4},$$

     where for $s = 0$ all inner points of the grid are set to be 0.

   - **Successive over Relaxation**

     In this case the formula is:

     $$V_{i,j}^{s+1} = (1-w)V_{i,j}^{s} + w\frac{(1+\Delta x_2)V_{i,j+1}^{s+1} + (1-\Delta x_2)V_{i,j-1}^{s} + V_{i+1,j}^{s} + V_{i-1,j}^{s+1}}{4},$$

     again implying that values computed in the current iteration may be used to approximate other values within the same iteration.

     $w = 1$ again yields the **Gauss-Seidel formula** for equation (7.13).

3. **Mixed Policy (Strategy)**

   This is the case where we implement the PDE as initially given, i.e. as

   $$\frac{1}{2}(V_{x_1x_1} + V_{x_2x_2}) + \max(V_{x_1}, V_{x_2}), \quad \text{for} \quad x_1 \leq x_2, \tag{7.15}$$

   with unchanged boundary conditions.

   The FDS for this case has the following form:

   $$V_{i,j} = \frac{1}{4}(V_{i+1,j} + V_{i-1,j} + V_{i,j+1} + V_{i,j-1}) + \\ + \frac{\Delta x_1}{4}\max\Big((V_{i+1,j} - V_{i-1,j}), (V_{i,j+1} - V_{i,j-1})\Big). \tag{7.16}$$

Remark 7.1.1 *Since we assumed equality between Δx_1 and Δx_2, it is irrelevant which one of them we use to obtain the needed finite difference scheme.*

- **Jacobi Iteration**

  In this case the formula is:

$$V_{i,j}^{s+1} = \frac{1}{4}(V_{i+1,j}^{s} + V_{i-1,j}^{s} + V_{i,j+1}^{s} + V_{i,j-1}^{s}) + \\ + \frac{\Delta x_1}{4} \max\Big((V_{i+1,j}^{s} - V_{i-1,j}^{s}), (V_{i,j+1}^{s} - V_{i,j-1}^{s})\Big),$$

  where for $s = 0$ all inner grid points are initialized to be 0.

- **Successive over Relaxation**

  The formula for this this method is:

$$V_{i,j}^{s+1} = (1-w)V_{i,j}^{s} + w\frac{1}{4}(V_{i+1,j}^{s} + V_{i-1,j}^{s+1} + V_{i,j+1}^{s+1} + V_{i,j-1}^{s}) + \\ + \frac{\Delta x_1}{4} \max\Big((V_{i+1,j}^{s} - V_{i-1,j}^{s+1}), (V_{i,j+1}^{s+1} - V_{i,j-1}^{s})\Big),$$

  thus permitting the use of some of the values computed in the current iteration within the same iteration.

  The **Gauss-Seidel formula** for equation (7.16) is again given for $w = 1$.

Remark 7.1.2 *The notation used thus far does not correspond to the notation used later on in the source codes. In the usual notation $V_{i,j}$, i stands for the x_1-variable and j for the x_2-variable whereas when coding, the matrix notation is used. Therefore, in $V(i,j)$ i refers to the x_2-mesh points and j to the x_1-mesh points.*

7.2 Monte Carlo Simulation

1. **No Strategy / Policy**

   First we take a look at the initial situation, i.e. when no policy has been yet applied.

   In the initial situation we have two companies modelled as Brownian motions $(W_i(t))_{t\geq 0}$, starting at $x_i > 0$, for $i = 1, 2$:

$$X_i(t_0) = x_i, \quad i = 1, 2, \tag{7.17}$$

$$X_i(t_{j+1}) = X_i(t_j) + \sigma_i \sqrt{t_{j+1} - t_j} Z_i(t_{j+1}), \tag{7.18}$$

   where $\sigma = 1$ in the original model and $Z_i(t_j)$, $i = 1, 2$, $j = 1, \ldots, n$ are independent, identically distributed standard normal random variables.

   If $X_i(t_j) \leq 0$, then company i is assumed to have gone bankrupt at time point t_j, implying that its value and the values of its successors in time is set to be 0:

$$X_i(t_j) \leq 0 \quad \Rightarrow \quad X_i(t_k) = 0, \quad \text{for} \quad k = j, j+1, \ldots, n. \tag{7.19}$$

2. **Arbitrary Strategy / Policy**

   In this case we distribute the unit of positive drift amongst the two companies regardless

of who needs it the most, i.e. **randomly**.

Recalling Subsection 6.2.2, we know that the drifts μ_i, $i = 1, 2$ should satisfy:

$$\mu_i(t_j) \geq -\delta, \quad -\frac{1}{2} \leq \delta < \infty, \tag{7.20}$$

$$\mu_1(t_j) + \mu_2(t_j) = 1, \quad \forall j = 1, \ldots, n. \tag{7.21}$$

Therefore, we assume one of the drifts $\mu_i(t_j)$ to be uniformly distributed on the interval $\big(-\delta(t_j), 1-(-\delta(t_j))\big)$ and compute the other one as follows:

$$\mu_1(t_j) \sim U\big(-\delta(t_j), 1+\delta(t_j)\big) \quad \Rightarrow \quad \mu_2(t_j) = 1 - \mu_1(t_j) \quad \forall j = 1, \ldots, n. \tag{7.22}$$

The process described above thus becomes:

$$X_i(t_0) = x_i, \quad i = 1, 2, \tag{7.23}$$

$$X_i(t_{j+1}) = X_i(t_j) + \mu_i(t_{j+1})(t_{j+1} - t_j) + \sigma_i \sqrt{t_{j+1} - t_j} Z_i(t_{j+1}), \tag{7.24}$$

for $j = 0, \ldots, n-1$ and equation (7.19) applies here too.

3. **Push-Bottom Strategy / Democratic Policy**

   This is the case where the unit drift is being entirely given to the poorer company namely at each time point t_j, $j = 1, \ldots, n$.

   After generating data for the 2 companies according to no strategy, their values are compared at each time point t_j, $\forall j = 1, \ldots, n$:

   - $\min\limits_{i=1,2} \quad X_i(t_j) = X_1(t_j)$

     Assigning drift 1 to the bottom company translates to:

     $$X_i(t_0) = x_i, \quad i = 1, 2, \tag{7.25}$$

     $$X_1(t_{j+1}) = X_1(t_j) + 1 \cdot (t_{j+1} - t_j) + \sigma_1 \sqrt{t_{j+1} - t_j} Z_1(t_{j+1}), \tag{7.26}$$

     $$X_2(t_{j+i}) = X_2(t_j) + \sigma_2 \sqrt{t_{j+1} - t_j} Z_2(t_{j+1}). \tag{7.27}$$

   - $\min\limits_{i=1,2} \quad X_i(t_j) = X_2(t_j)$

     Applying drift 1 to the poorer company gives:

     $$X_i(t_0) = x_i, \quad i = 1, 2, \tag{7.28}$$

     $$X_1(t_{j+i}) = X_1(t_j) + \sigma_1 \sqrt{t_{j+1} - t_j} Z_1(t_{j+1}), \tag{7.29}$$

     $$X_2(t_{j+1}) = X_2(t_j) + 1 \cdot (t_{j+1} - t_j) + \sigma_2 \sqrt{t_{j+1} - t_j} Z_2(t_{j+1}). \tag{7.30}$$

   Ultimately, if a company goes bankrupt, condition (7.19) holds.

4. **Push-Top Strategy / Republican Policy**

   In this case, the unit drift is distributed to the richer company at each time point t_j, $j = 1, \ldots, n$ that is after comparing data generated according to no policy.

- $\max\limits_{i=1,2} \quad X_i(t_j) = X_1(t_j)$

  Applying drift 1 to the richer company translates to:

$$\begin{aligned} &X_i(t_0) = x_i, \quad i = 1, 2, \\ &X_1(t_{j+1}) = X_1(t_j) + 1 \cdot (t_{j+1} - t_j) + \sigma_1 \sqrt{t_{j+1} - t_j} Z_1(t_{j+1}), \\ &X_2(t_{j+i}) = X_2(t_j) + \sigma_2 \sqrt{t_{j+1} - t_j} Z_2(t_{j+1}). \end{aligned}$$

- $\max\limits_{i=1,2} \quad X_i(t_j) = X_2(t_j)$

  Giving the unit drift to the richer company means:

$$\begin{aligned} &X_i(t_0) = x_i, \quad i = 1, 2, \\ &X_1(t_{j+i}) = X_1(t_j) + \sigma_1 \sqrt{t_{j+1} - t_j} Z_1(t_{j+1}), \\ &X_2(t_{j+1}) = X_2(t_j) + 1 \cdot (t_{j+1} - t_j) + \sigma_2 \sqrt{t_{j+1} - t_j} Z_2(t_{j+1}). \end{aligned}$$

Ultimately, if a company goes bankrupt, condition (7.19) holds.

Remark 7.2.1 *It is assumed that once a company goes bankrupt, the surviving company automatically receives the full unit of drift until its own potential bankruptcy.*

Chapter 8

Results

8.1 Numerical Approximation

Three separate numerical approximations have been conducted, one for each policy, their resulting functions being denoted by V_1 for the democratic, V_2 for the republican and V for the mixed policy respectively.

To that purpose, we assumed $x_{1\max} = x_{2\max} = 8$, $N = 200$ grid points (excluding 0) and $\Delta x = \Delta y = 0.04$ and conducted approximations of the objective function for three values of the parameter α: $\alpha = \frac{1}{3}$, $\alpha = \frac{2}{5}$ and $\alpha = \frac{1}{2}$.

With these parameters, the finite difference scheme depicted in Section 7.1 was successfully implemented and the approximations of functions V_1, V_2 and V were found using Jacobian Iteration and Successive over Relaxation (with parameters $w = \frac{1}{2}$, $w = \frac{3}{2}$ and $w = 1$, the latter transforming the Successive over Relaxation method into Gauss-Seidel Iteration), all of which leading to very similar results, the difference between them consisting in diverse iteration numbers.

Worth mentioning at this point is that the capacity of the computer on which the approximations were conducted did not allow the use of greater values for N and $x_{1\max}$ and $x_{2\max}$ since it ran out of memory.

In the following, plots for the obtained results can be found. In the pictures below, black means that the bottom company is pushed while white stands for pushing the top company. On the anti-diagonal, it is irrelevant which company is pushed.

First, let us recall how the partial differential equation describing the dynamics of the objective function looked like:

$$\frac{1}{2}(V_{x_1x_1} + V_{x_2x_2}) + \max(V_{x_1}, V_{x_2}) = 0. \tag{8.1}$$

8.1.1 Democratic Policy

As previously mentioned, the democratic policy corresponds to choosing V_{x_1} as the maximum in the above equation, for $x_1 \leq x_2$, therefore enabling the support for the bottom company. Thus we obtain an approximation of function V which we will denote by V_1.

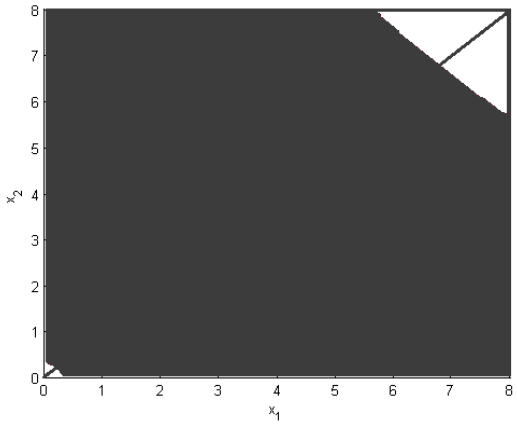

Figure 8.1: $V_{x_1} - V_{x_2}$ for function V_1 with $\alpha = \frac{1}{3}$

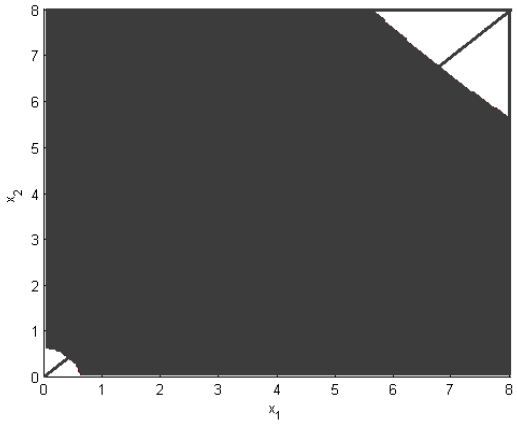

Figure 8.2: $V_{x_1} - V_{x_2}$ for function V_1 with $\alpha = \frac{2}{5}$

As can be seen in Figures 8.1 and 8.2, in the area near the origin, i.e. where both companies are in danger of going bankrupt, the top company is being pushed. For $\alpha = \frac{2}{5}$ this area gets larger. When both companies are well off, pushing the top company also appears to be giving greater values for the objective function. However this area may be the effect of the chosen values for the numerical approximation. Since the upper and the right boundaries exhibit greater values than the axes boundaries (particularly the right upper corner takes the greatest values), the approximation of V_y is more prone to producing greater values than that of V_x.

$\alpha = \frac{1}{2}$ is a special case, where it appears to be optimal to only push the top company, as

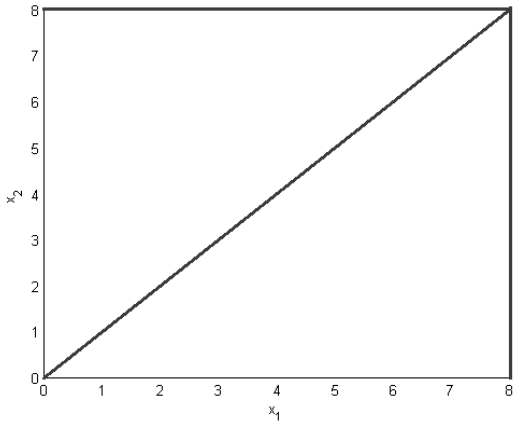

Figure 8.3: $V_{x_1} - V_{x_2}$ for function V_1 with $\alpha = \frac{1}{2}$

results from Figure 8.3.

8.1.2 Republican Policy

The republican policy corresponds to choosing V_{x_2} as the maximum in equation (8.1), for $x_1 \leq x_2$ (see Section 7.1) i.e. it is thus made sure that the top company is being "pushed". Consequently we get an approximation of function V which we denote by V_2.

Figures 8.4 and 8.5 show yet again that near the origin, it is better to push the top company, whereby the white area is bigger the greater α is. However, the white area opposed to the origin is even bigger in this case than in the previous one and is subject to the same effect as mentioned previously.

Figure 8.6 exhibits the same behaviour for $\alpha = \frac{1}{2}$ as Figure 8.3 does (see previous subsection).

Figure 8.4: $V_{x_2} - V_{x_1}$ for function V_2 with $\alpha = \frac{1}{3}$

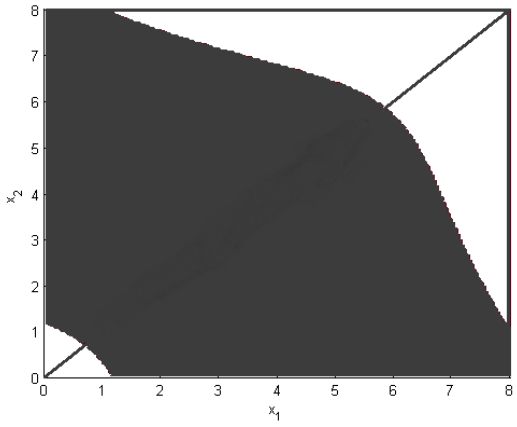

Figure 8.5: $V_{x_2} - V_{x_1}$ for function V_2 with $\alpha = \frac{2}{5}$

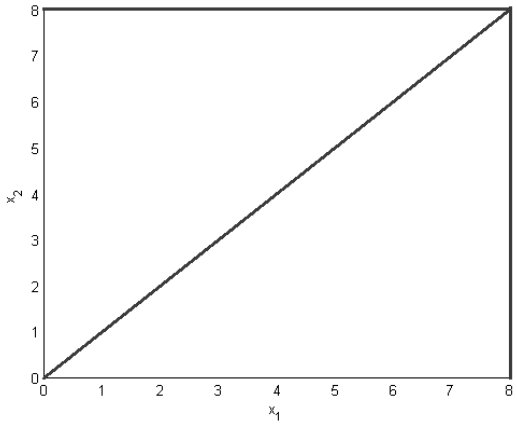

Figure 8.6: $V_{x_2} - V_{x_1}$ for function V_2 with $\alpha = \frac{1}{2}$

8.1.3 Mixed Policy

It is now obvious, that no pure democratic or republican policy is the optimal objective function. Truth be told, for $0 < \alpha < \frac{1}{2}$ the optimal policy is a mixture of both the push-bottom and the push-top strategies. Nonetheless, for the sole, special case of $\alpha = \frac{1}{2}$, the optimal policy is the pure republican policy.

The optimal objective functions for $\alpha = \frac{1}{3}$, $\alpha = \frac{2}{5}$ and $\alpha = \frac{1}{2}$ are given below in Figures 8.7, 8.8 and 8.9.

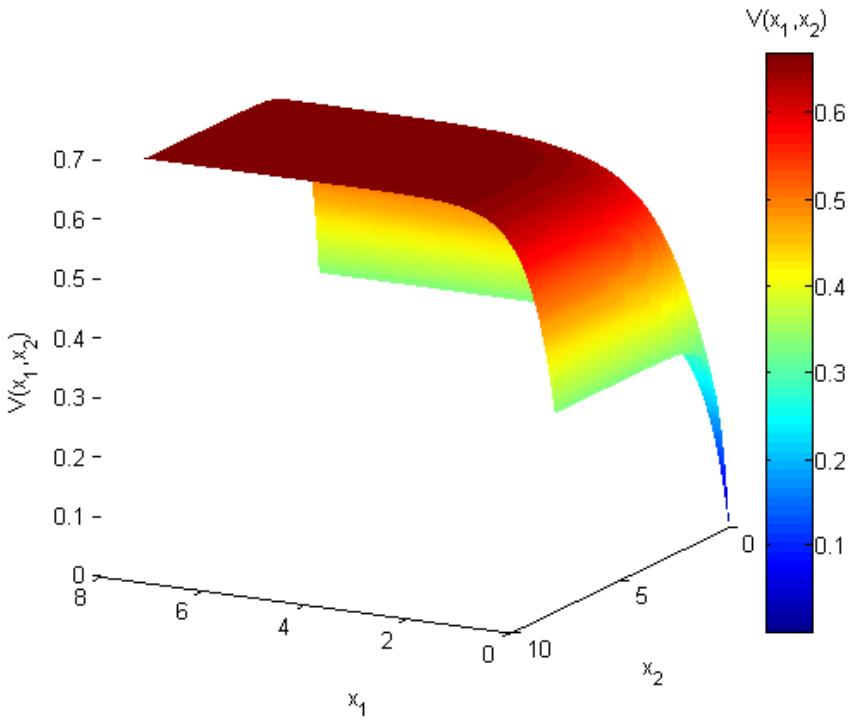

Figure 8.7: Optimal function V for $\alpha = \frac{1}{3}$

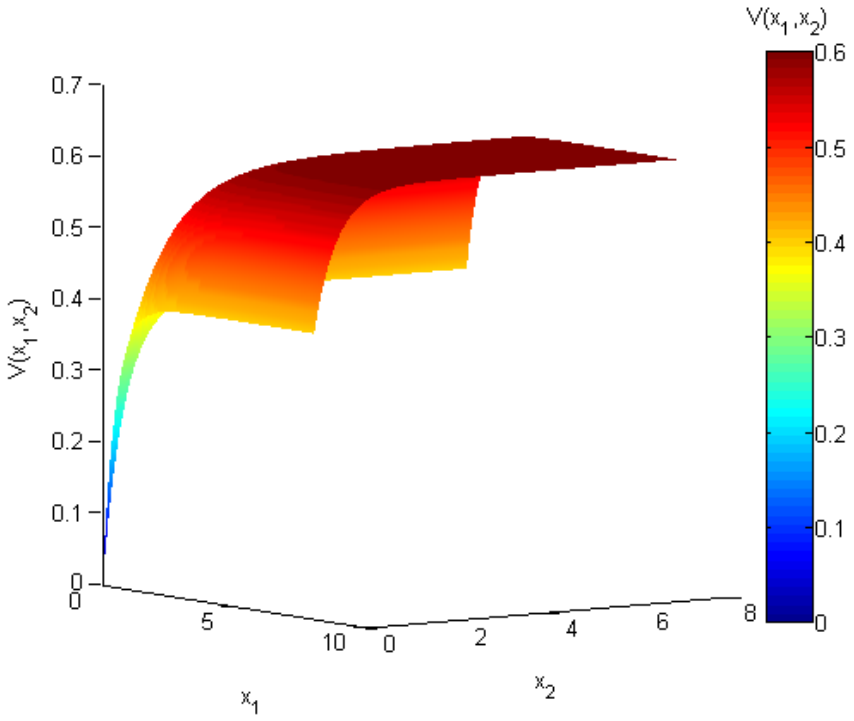

Figure 8.8: Optimal function V for $\alpha = \frac{2}{5}$

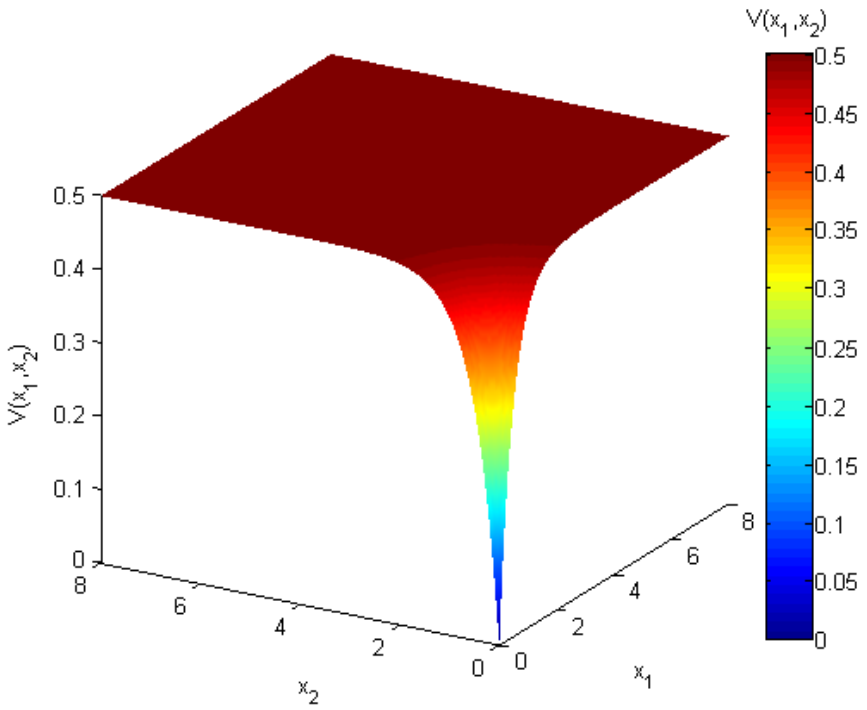

Figure 8.9: Optimal function V for $\alpha = \frac{1}{2}$

8.1.4 Convergence Optimization

In this subsection, we want to make a comparison between the efficiency of the implemented iterative methods. To this end, we have conducted several experiments, varying the number of grid points. Thus, Jacobi Iteration, Gauss-Seidel Iteration and Successive over Relaxation (both under- and over-relaxation) were submitted to an efficiency test for $N = 200$, $N = 100$ and $N = 50$ grid points.

The resulting numbers of iterations for V_1, V_2 and V, for $\alpha = \frac{1}{3}$, $\alpha = \frac{2}{5}$, $\alpha = \frac{1}{2}$, are given in Table 8.1.

It is evident, that the Successive over Relaxation method with parameter $w = \frac{3}{2}$ signifying over-relaxation, is by far the most efficient iterative method since it needs the smallest number of iterations to reach the same result as all the other ones.

Noteworthy is that for $\alpha = \frac{1}{2}$, attaining the optimal result requires the same number of iterations for V_2 and V, again suggesting that in this case it is optimal to always choose the republican policy and push the top company.

			Jacobi	Gauss-Seidel	SoR ($w = \frac{1}{2}$)	SoR ($w = \frac{3}{2}$)
$N = 200$	$\alpha = \frac{1}{3}$	V_1	67.891	35.305	99.411	12.475
		V_2	29.831	15.426	43.709	5.393
		V	58.950	30.782	85.979	10.934
	$\alpha = \frac{2}{5}$	V_1	67.474	35.097	98.786	12.406
		V_2	29.689	15.354	43.495	5.369
		V	57.630	30.121	84.000	10.713
	$\alpha = \frac{1}{2}$	V_1	66.754	34.736	97.705	12.286
		V_2	29.450	15.234	43.135	5.328
		V	29.450	15.234	43.135	5.328
$N = 100$	$\alpha = \frac{1}{3}$	V_1	18.346	9.509	26.915	3.336
		V_2	8.016	4.122	11.761	1.419
		V	16.100	8.363	23.533	2.938
	$\alpha = \frac{2}{5}$	V_1	18.242	9.457	26.758	3.318
		V_2	7.981	4.105	11.707	1.413
		V	15.769	8.198	23.036	2.883
	$\alpha = \frac{1}{2}$	V_1	18.062	9.367	26.488	3.289
		V_2	7.921	4.047	11.617	1.402
		V	7.921	4.047	11.617	1.402
$N = 50$	$\alpha = \frac{1}{3}$	V_1	4.927	2.545	7.244	880
		V_2	2.138	1.091	3.144	360
		V	4.363	2.253	6.392	775
	$\alpha = \frac{2}{5}$	V_1	4.901	2.532	7.205	876
		V_2	2.128	1.087	5.130	359
		V	4.280	2.212	6.268	761
	$\alpha = \frac{1}{2}$	V_1	4.856	2.510	7.137	868
		V_2	2.114	1.079	3.108	356
		V	2.114	1.079	3.108	356

Table 8.1: Number of iterations used to approximate V_1, V_2 and V

8.2 Monte Carlo Simulation

The Monte Carlo simulation was implemented for $n = 100$ time points (excluding 0) over the time interval $[0, T]$, with $T = 10$, thus determining the time step-size to be $\Delta t = 0.1$.

$N = 10.000$ simulations were conducted and the objective function was calculated depending on three values for the parameter α: $\alpha = 0$ (corresponding to the probability that both companies survive forever), $\alpha = \frac{1}{3}$ (corresponding to the expected number of companies that remain solvent) and $\alpha = \frac{1}{2}$ (meaning that the probability that both companies survive and the probability that only one company survives are equally weighed).

First, an initial situation is being considered. For the initial situation, we have chosen the following values: $x_1 = 10^{-5}$ and $x_2 = 2$ translating into almost bankruptcy for one company while the other one is well off, $\sigma_1 = \sigma_2 = 1$, $\delta = 0$.

Figures 8.10 and 8.11 portray the initial situation, showing for $\alpha = 0$ and $\alpha = \frac{1}{3}$ that pushing the bottom company is optimal. Figures 8.12 and 8.13 show that pushing the richer company is slightly better than pushing the weaker one, for $\alpha = \frac{1}{2}$.

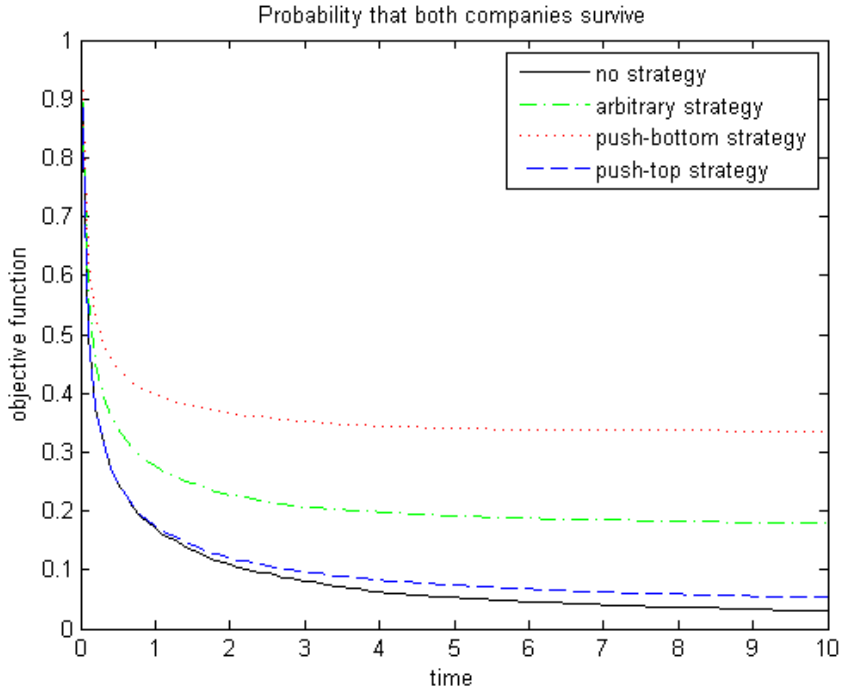

Figure 8.10: Objective function with $x_1 = 10^{-5}, x_2 = 2, \sigma_1 = \sigma_2 = 1, \delta = 0$, for $\alpha = 0$

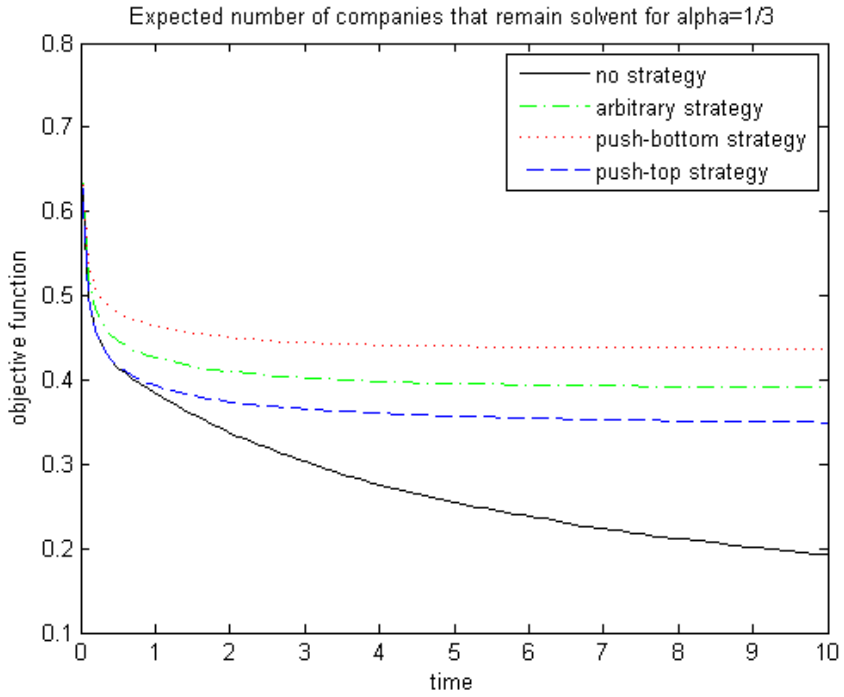

Figure 8.11: Objective function with $x_1 = 10^{-5}, x_2 = 2, \sigma_1 = \sigma_2 = 1, \delta = 0$, for $\alpha = \frac{1}{3}$

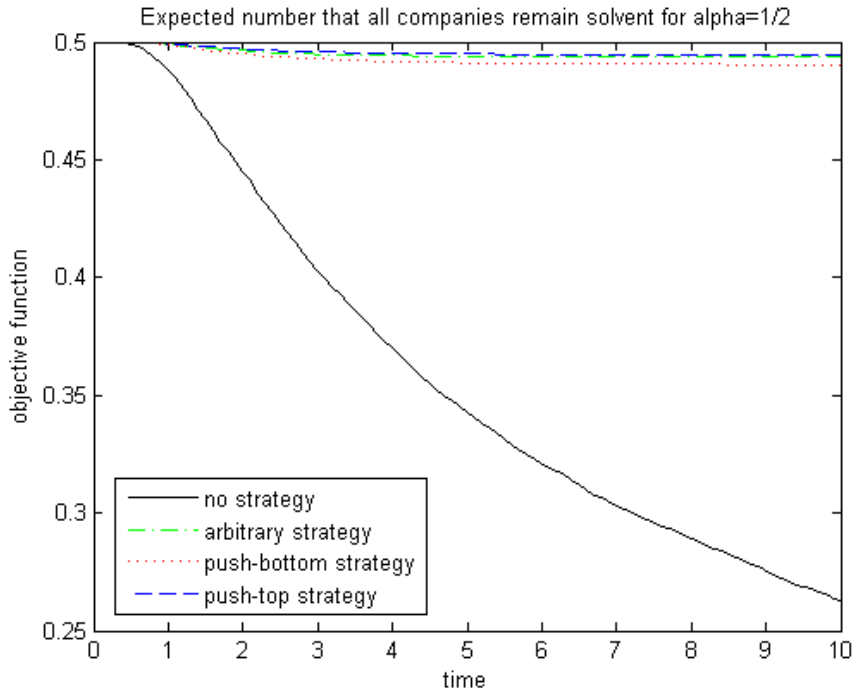

Figure 8.12: Objective function with $x_1 = 10^{-5}, x_2 = 2, \sigma_1 = \sigma_2 = 1, \delta = 0$, for $\alpha = \frac{1}{2}$

Table 8.2 portrays the evolution of the number of companies if no policy, the arbitrary, demo-

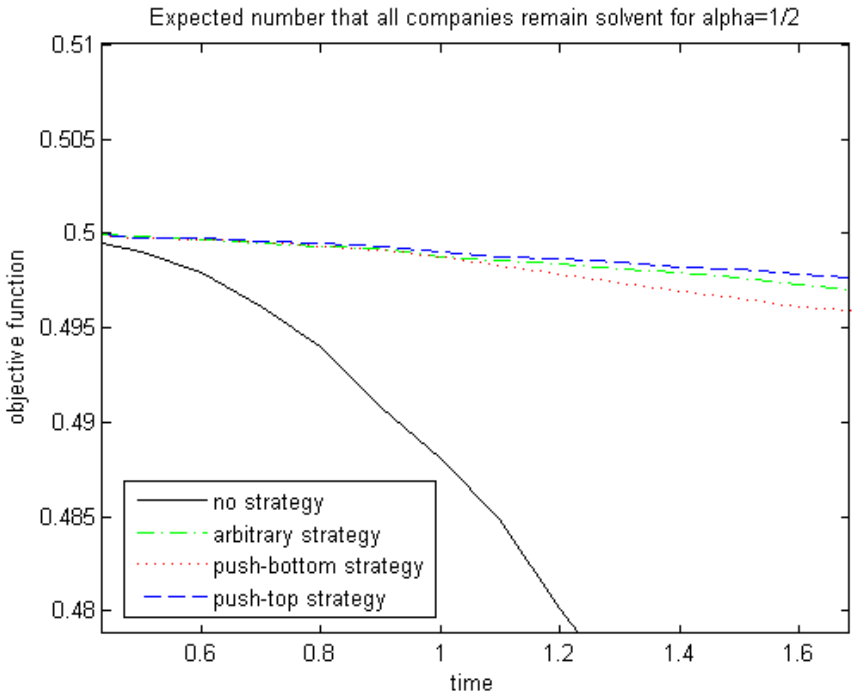

Figure 8.13: Excerpt from Figure 8.12

cratic and republican policies are implemented, for the first 10 time points. At t_0 it is assumed that the starting points x_i, $i = 1, 2$ are all greater than 0.

	No Policy		Arbitrary Policy		Democratic Policy		Republican Policy	
	$\#_1$	$\#_2$	$\#_1$	$\#_2$	$\#_1$	$\#_2$	$\#_1$	$\#_2$
t_0	0	10.000	0	10.000	0	10.000	0	10.000
t_1	5.029	4.971	4.413	5.587	3.769	6.231	5.029	4.971
t_2	6.297	3.703	5.536	4.464	4.764	5.236	6.297	3.703
t_3	6.876	3.122	6.074	3.926	5.218	4.781	6.877	3.123
t_4	7.237	2.759	6.364	3.634	5.456	4.542	7.239	2.760
t_5	7.480	2.501	6.596	3.401	5.623	4.373	7.494	2.503
t_6	7.688	2.277	6.760	3.234	5.747	4.247	7.706	2.289
t_7	7.842	2.086	6.928	3.062	5.839	4.150	7.885	2.107
t_8	7.952	1.941	7.024	2.959	5.935	4.050	8.024	1.965
t_9	8.038	1.797	7.113	2.865	5.992	3.986	8.147	1.837
t_{10}	8.070	1.704	7.180	2.792	6.050	3.922	8.233	1.746

Table 8.2: Number of times that a company and both companies survive in the initial situation

The evolution of the number of times that only one company survives and of the number of times that both companies survive over t_j, $j = 1, \ldots, 10$ indicates that the democratic policy majorly enhances the number of times that both companies survive and that there also exists a strong correlation between the republican policy and the growing number of times that a single company survives. As intuited, the arbitrary policy is a more moderate policy than both the democratic and the republican ones, which do seem rather extreme.

Now, we will continue on by varying the parameters.

First in line are the initial values x_1 and x_2.

What happens if, for example, both companies have very low starting points, i.e. if they are both in danger of going bankrupt?

In Figures 8.14, 8.15, 8.16 and 8.17, we keep all parameters intact except the initial values which we change to $x_1 = 10^{-7}$ and $x_2 = 10^{-10}$.

Obviously, if the criterion is to maximize the probability that both companies survive, $\alpha = 0$, the democratic policy is optimal, whereas if the criterion is to maximize the expected number of companies that remain solvent $(\alpha = \frac{1}{3})$ and $(\alpha = \frac{1}{2})$, the best policy appears to be the republican one.

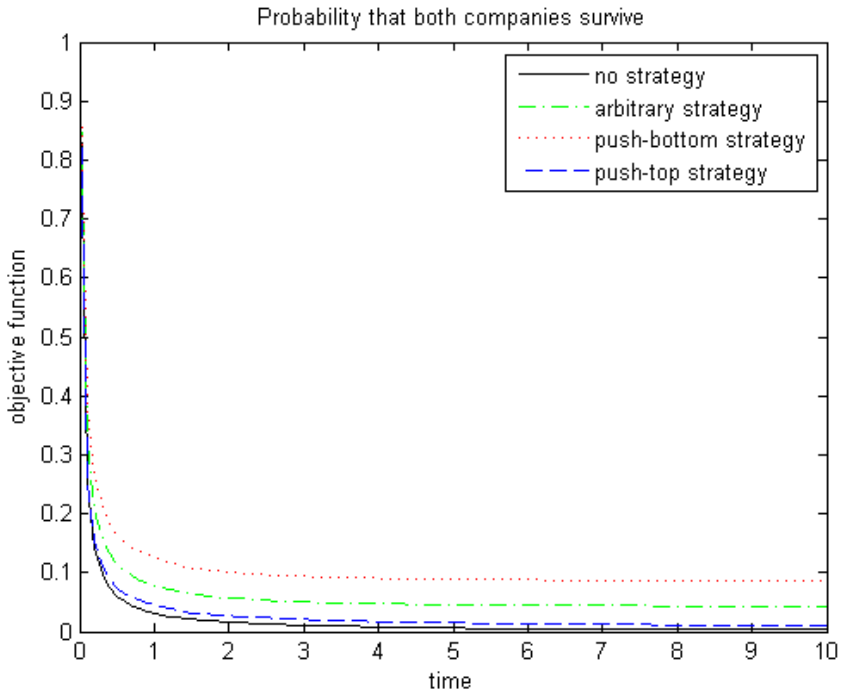

Figure 8.14: Objective function with $x_1 = 10^{-7}, x_2 = 10^{-10}, \sigma_1 = \sigma_2 = 1, \delta = 0$, for $\alpha = 0$

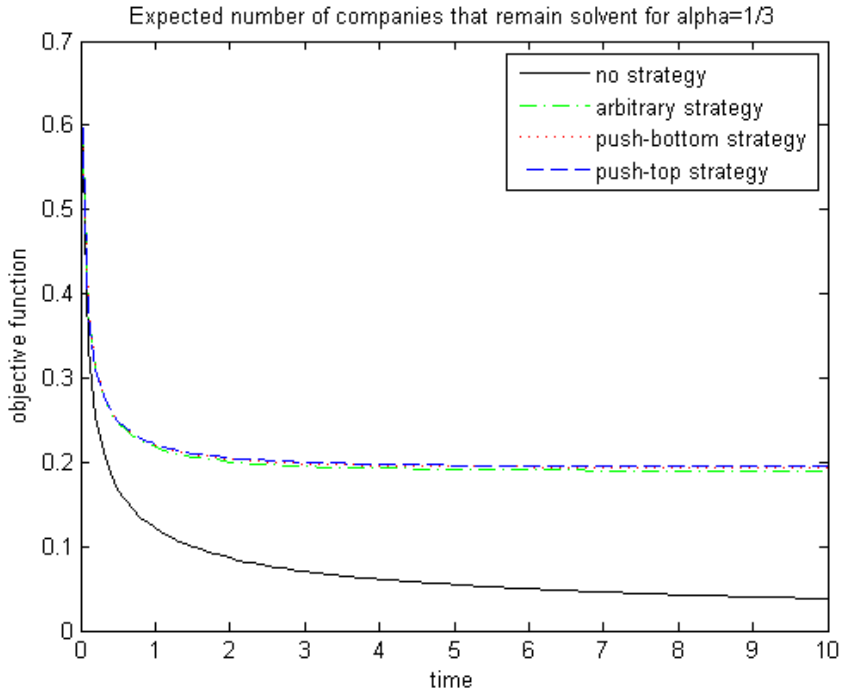

Figure 8.15: Objective function with $x_1 = 10^{-7}, x_2 = 10^{-10}, \sigma_1 = \sigma_2 = 1, \delta = 0$, for $\alpha = \frac{1}{3}$

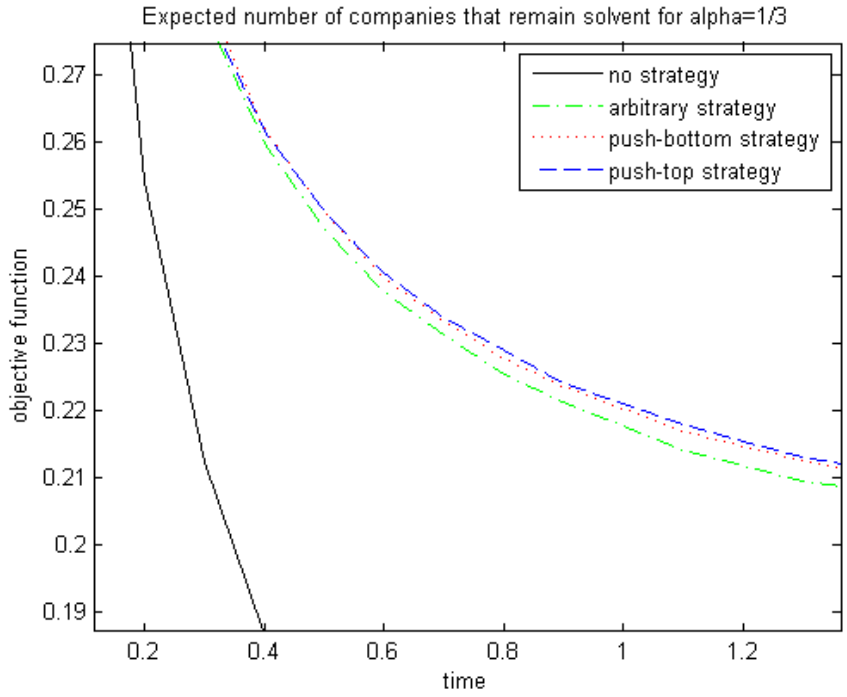

Figure 8.16: Excerpt from Figure **8.15**

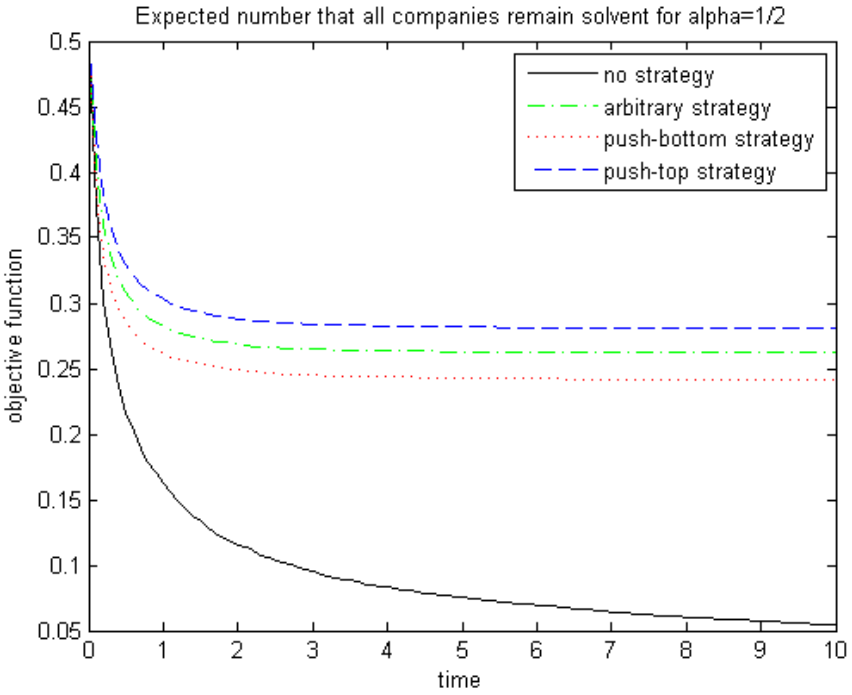

Figure 8.17: Objective function with $x_1 = 10^{-7}, x_2 = 10^{-10}, \sigma_1 = \sigma_2 = 1, \delta = 0$, for $\alpha = \frac{1}{2}$

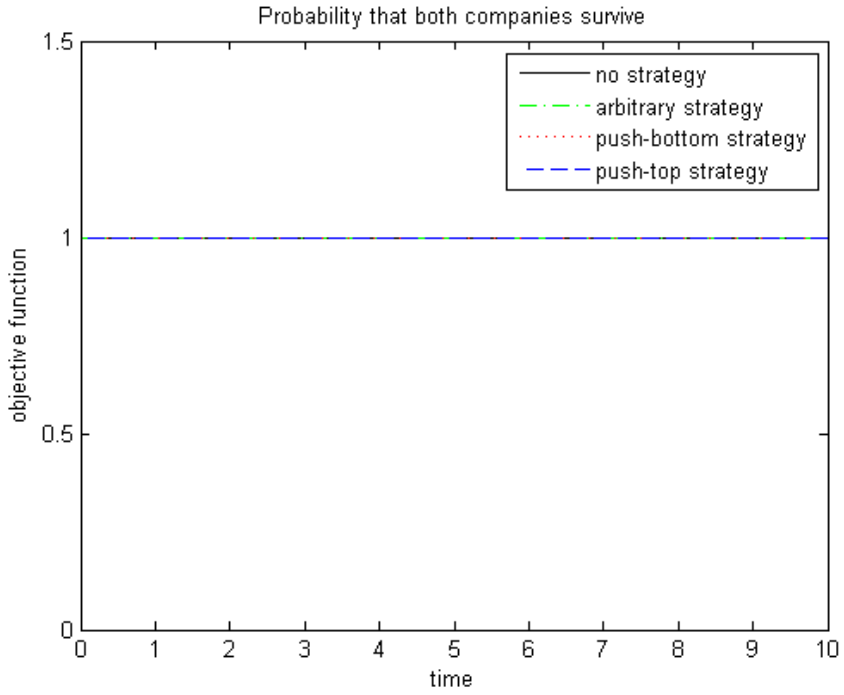

Figure 8.18: Objective function with $x_1 = 14, x_2 = 14, \sigma_1 = \sigma_2 = 1, \delta = 0$, for $\alpha = 0$

But what happens if both companies are very well off at the beginning?

By taking for example $x_1 = 14$ and $x_2 = 14$ it is revealed that no company goes bankrupt and that therefore all policies are optimal as Figure 8.18 shows.

Remark 8.2.1 *Figure 8.18 only shows the result of the simulation for $\alpha = 0$. The behaviour of the objective function is preserved for $\alpha = \frac{1}{3}$ and $\alpha = \frac{1}{2}$, respectively, however, the straight line is located at $\frac{2}{3}$ and $\frac{1}{2}$, respectively.*

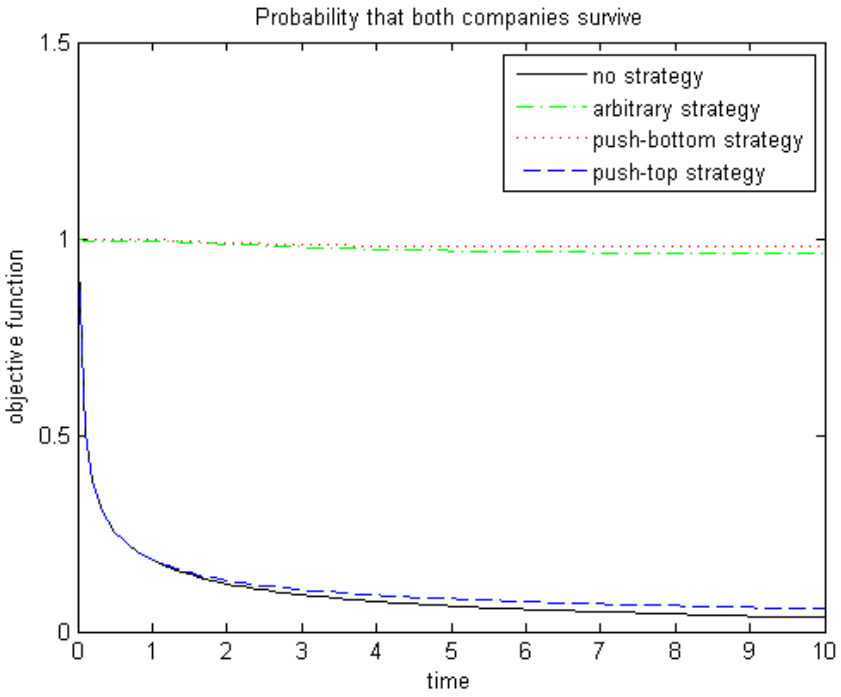

Figure 8.19: Objective function with $x_1 = 10^{-5}, x_2 = 2, \sigma_1 = 0.004, \sigma_2 = 0.8, \delta = 0$, for $\alpha = 0$

We now want to vary the volatilities, which is why we take for the initially near bankrupt company $\sigma_1 = 0.004$ and for the better endowed company $\sigma_2 = 0.8$.

Figures **8.19**, **8.20**, **8.21** and **8.22** reveal that pushing the bottom company is the optimal strategy for all three values of α.

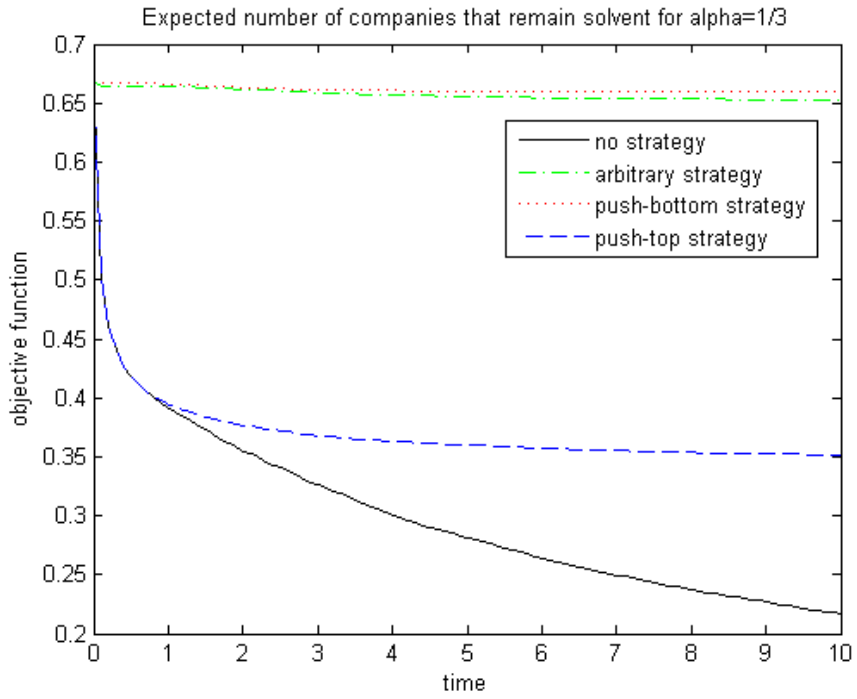

Figure 8.20: Objective function with $x_1 = 10^{-5}, x_2 = 2, \sigma_1 = 0.004, \sigma_2 = 0.8, \delta = 0$, for $\alpha = \frac{1}{3}$

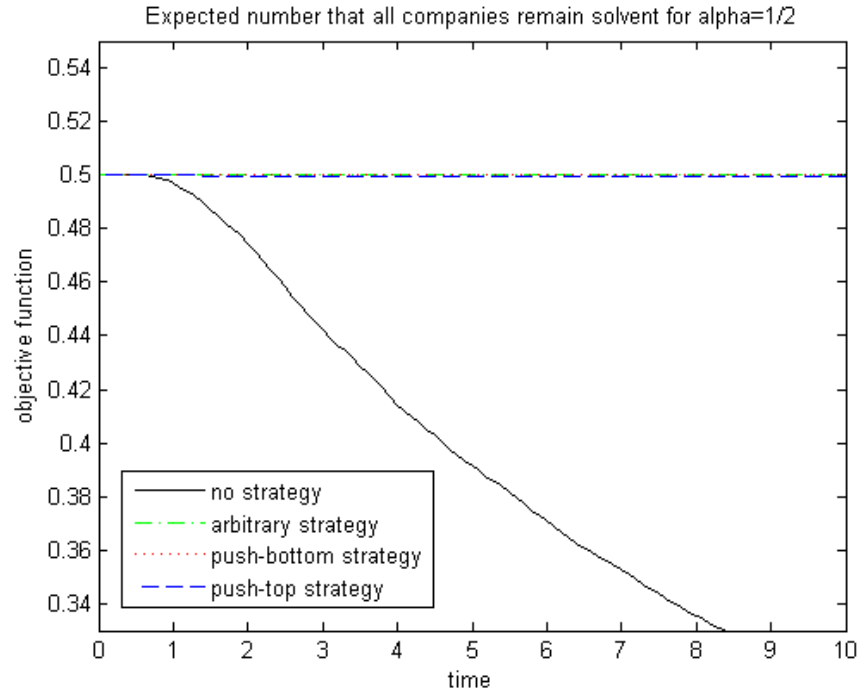

Figure 8.21: Objective function with $x_1 = 10^{-5}, x_2 = 2, \sigma_1 = 0.004, \sigma_2 = 0.8, \delta = 0$, for $\alpha = \frac{1}{2}$

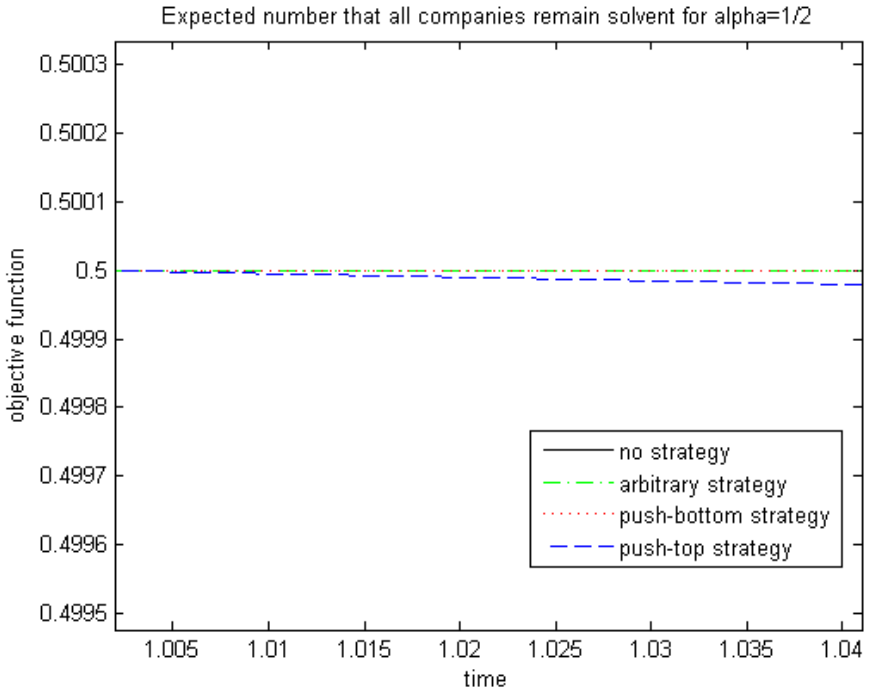

Figure 8.22: Excerpt from Figure 8.21

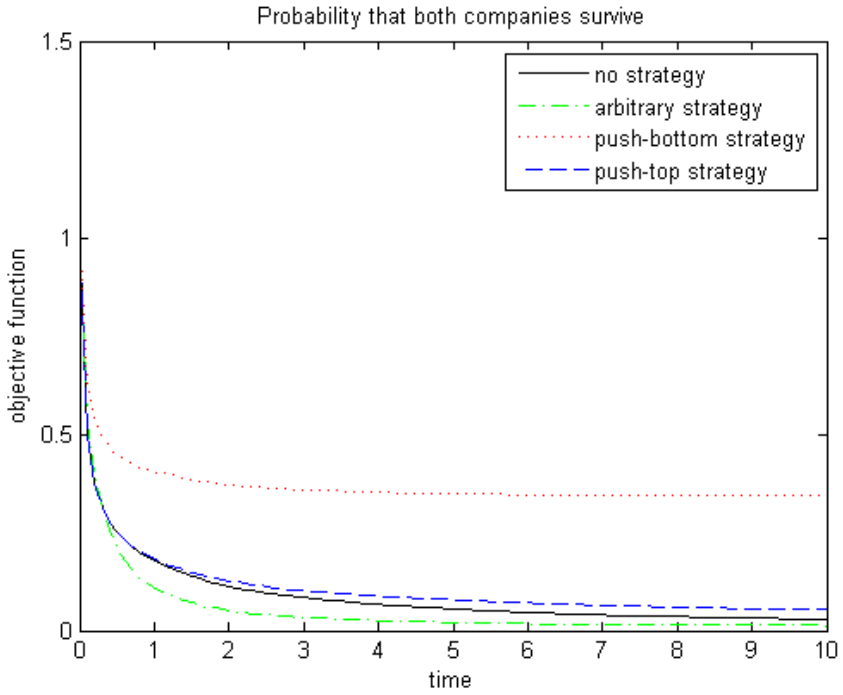

Figure 8.23: Objective function with $x_1 = 10^{-5}, x_2 = 2, \sigma_1 = \sigma_2 = 1, \delta = 10$, for $\alpha = 0$

Finally, what is left to do is to vary the parameter δ.

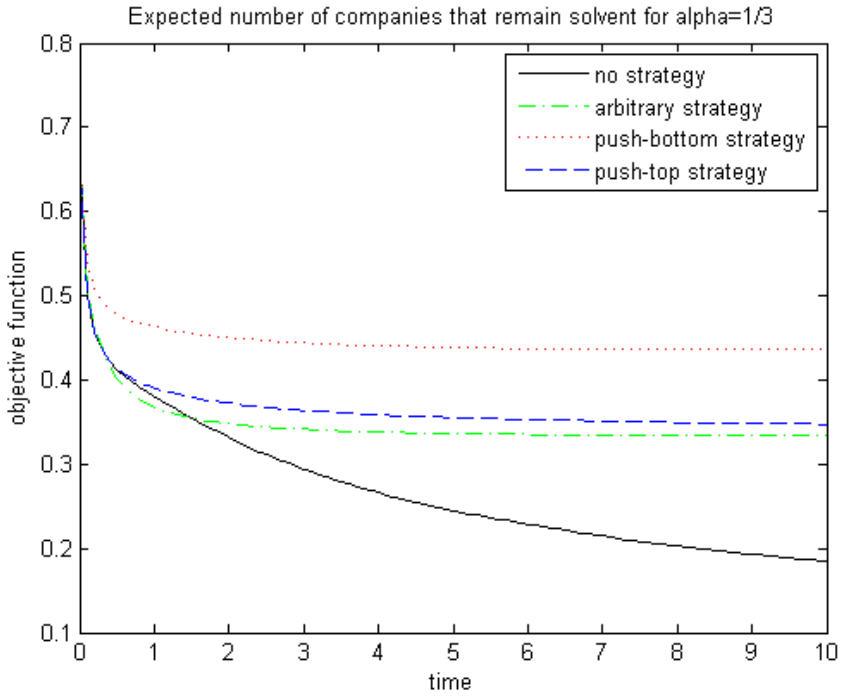

Figure 8.24: Objective function with $x_1 = 10^{-5}, x_2 = 2, \sigma_1 = \sigma_2 = 1, \delta = 10$, for $\alpha = \frac{1}{3}$

In the case of $\delta = 10$, the democratic policy is the best strategy for $\alpha = 0$ and $\alpha = \frac{1}{3}$ as

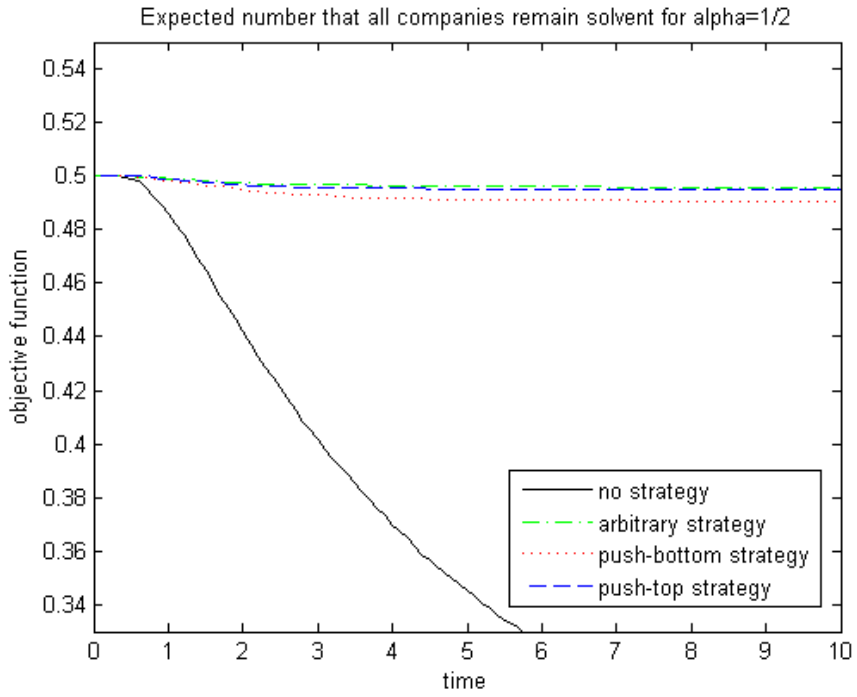

Figure 8.25: Objective function with $x_1 = 10^{-5}, x_2 = 2, \sigma_1 = \sigma_2 = 1, \delta = 10$, for $\alpha = \frac{1}{2}$

indicated by Figures 8.23 and 8.24, however, for $\alpha = \frac{1}{2}$, Figure 8.25 suggests that the optimal strategy is the arbitrary one.

Chapter 9

Extensions to Higher Dimensionality

9.1 The Three-Dimensional Case

After analyzing the outcomes of the 2-dimensional model, the time has come to ask the following question:

What is the optimal policy for higher dimensions?

This thesis also deals with the three-dimensional extension of the model presented in Section 6.2.

We therefore assume three companies on the market governed by Brownian motion, as given before. Thus, the **controlled processes** satisfy for $i = 1, 2, 3$ and $t \geq 0$:

$$dX_i(t) = \mu_i(t)dt + dW_i(t), \quad X_i(0) = x_i, \tag{9.1}$$

$$\mu_i(t) \geq 0, \quad \mu_1 + \mu_2 + \mu_3 = 1, \tag{9.2}$$

where x_i are given.

Either the probability that all three companies survive or the expected number of companies that never go bankrupt are yet again the two possible optimization criteria, however, this time without the scaling effect given by α.

Again, one can either push the bottom or the top company. Which policy is better in the 3-dimensional case?

Figures 9.1 and 9.2 indicate that for $x_1 = 10^{-1}, x_2 = 1, x_3 = 2$, with $\sigma_1 = \sigma_2 = \sigma_3 = 1$ the democratic policy is the optimal one.

For an in-depth analysis, resulting from the variation of this model's parameters, the interested reader is referred to the source codes inserted in this thesis' Appendix.

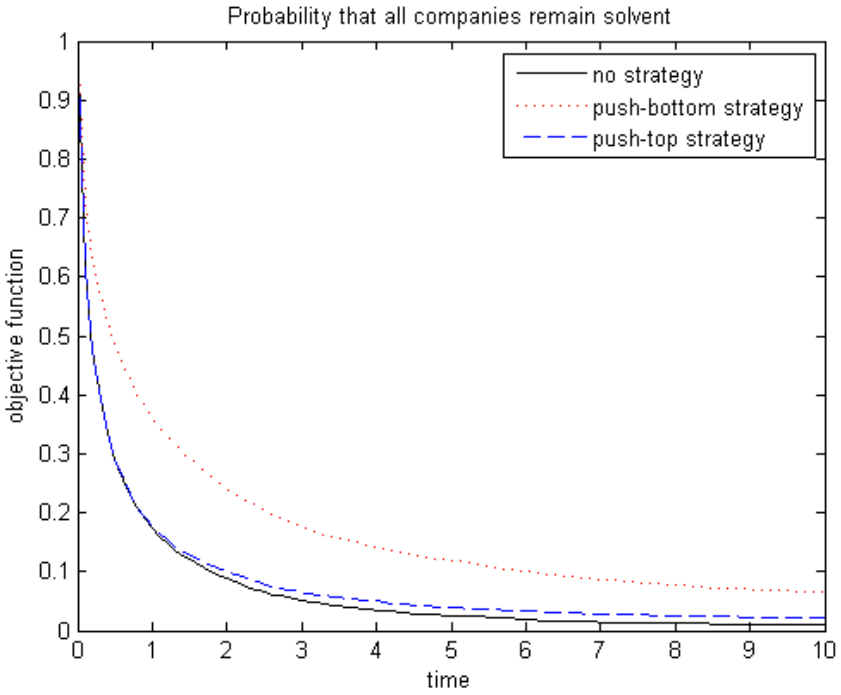

Figure 9.1: Objective function with $x_1 = 10^{-1}, x_2 = 1, x_3 = 2, \sigma_1 = \sigma_2 = \sigma_3 = 1$

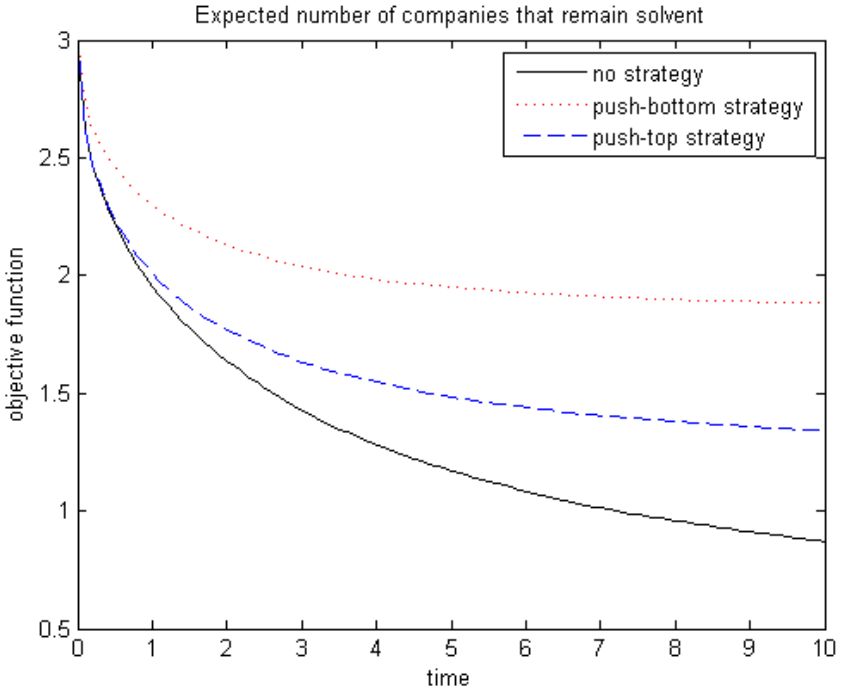

Figure 9.2: Objective function with $x_1 = 10^{-1}, x_2 = 1, x_3 = 2, \sigma_1 = \sigma_2 = \sigma_3 = 1$

Chapter 10

Conclusions

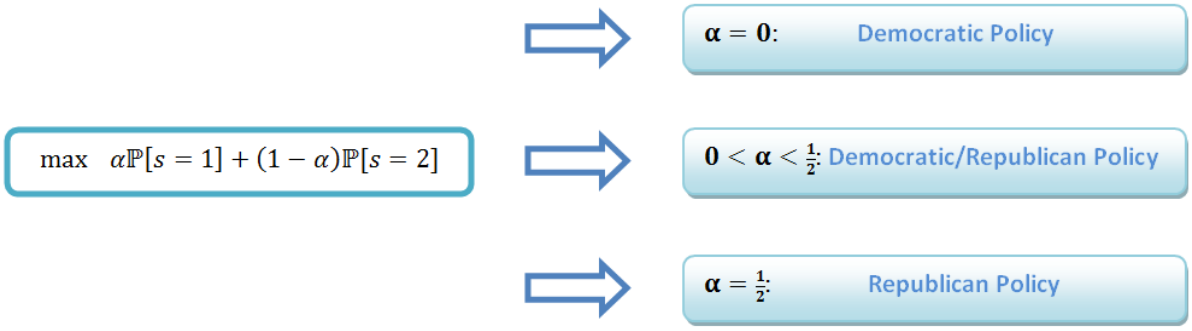

Figure 10.1: Conclusion summary

Case $\alpha = 0$

The case $\alpha = 0$ corresponds to maximizing the probability that both companies remain solvent.

In [18] McKean and Shepp guessed the optimal function (see Subsection 6.2.4), indicating that it results from always pushing the bottom company. The results obtained from the many Monte Carlo simulations conducted fully support this hypothesis.

Therefore, the **democratic policy** is the optimal policy if the criterion is to **maximize the probability that both companies survive**.

Case $0 < \alpha < \frac{1}{2}$

The case $0 < \alpha < \frac{1}{2}$ corresponds to maximizing the expected number of companies that never go bankrupt weighed with the help of α (see Subsection 6.2.2), whereby the probability that both companies survive is given more importance through a greater weighing factor.

The results of both the numerical approximations and the Monte Carlo simulations suggest the optimal policy is the democratic one except for the case where both companies are in danger of going bankrupt. In this particular situation, it is optimal to choose the republican policy and push the top company.

The Monte Carlo simulation additionally shows that if both companies are well-off, either policy is optimal.

Hence, if the criterion is to **maximize the expected number of companies that survive forever, for** $0 < \alpha < \frac{1}{2}$, the optimal policy is a **mixture of the democratic and the republican policies**.

Case $\alpha = \frac{1}{2}$

The case $\alpha = \frac{1}{2}$ corresponds to maximizing the expected number of companies that never go bankrupt , whereby the probability that one company survives and the probability that both companies survive are equally weighed.

The numerical approximations indicate that always pushing the top company is the optimal policy in this case. This hypothesis is also backed up by the Monte Carlo simulations.

Thus, if the criterion is to **maximize the expected number of companies that survive forever, for** $\alpha = \frac{1}{2}$, the optimal policy is the **republican policy**.

Appendix A

Source Codes General

The appendices are dedicated to posting the source codes of every algorithm implemented for this thesis. The programming language used to do so is MATLAB[1] (Version R2013b).

The source codes were implemented on a Sony Vaio laptop having the following features:

- CPU: Intel Core 2 Duo Processor T7500;
- Memory: 3 GB/Go DDR2 SDRAM;
- HDD: 320 GB/Go;
- Operating system: Windows 7 32-bit

Source code for producing the graph in Figure 2.1

```
% ==== Generating One-Dimensional Brownian Motion ===

% Method parameters

N=3;
n=1e3;
T=1;

% Setup Mesh

dt=T/n;
sq_dt=sqrt(dt);
t=0:dt:T;

% Creating array for the 1-dimensional Brownian motion

W=zeros(n+1,N);
W(1,:)=0;

% Generation of standard normally distributed random numbers

Z=[zeros(1,N);randn(n,N)];

% Generation of Brownian motions via Euler Scheme

W=cumsum(W+sq_dt*Z);
```

[1] www.mathworks.com

```
% Plot

plot(t,W,'LineWidth',1.5);
xlabel('time');
ylabel('position');
% axis([0 1 -10 10])
```

Source code for producing the graph in Figure 2.2

```
% ==== Generating Two-Dimensional Brownian Motion ====

% Model parameters

N=3;
n=1e3;
T=1;

% Setup Mesh

dt=T/n;
sq_dt=sqrt(dt);
t=0:dt:T;

% Creating array for the 2-dimensional Brownian motion

W_x=zeros(n+1,N);
W_x(1,:)=0;
W_y=zeros(n+1,N);
W_y(1,:)=0;

% Generation of standard normally distributed random numbers

Z_x=[zeros(1,N);randn(n,N)];
Z_y=[zeros(1,N);randn(n,N)];

% Generation of Brownian motions via Euler Scheme

W_x=cumsum(W_x+sq_dt*Z_x);
W_y=cumsum(W_y+sq_dt*Z_y);

% Plot

plot(W_x,W_y,'LineWidth',1.5);
xlabel('position abscissa');
ylabel('position ordinate');
```

Source code for producing the graph in Figure 6.1

```
x=-5:.01:5;
y=(1-exp(-2./x.^2)).*x;
plot(x,y), hold on
xL = xlim;
yL = ylim;
line([0 0], yL);  %x-axis
line(xL, [0 0]);  %y-axis
hold off;
xlabel('abscissa axis');
ylabel('ordinate axis');
```

Source code for producing the graph in Figure 6.2

```
x=0:.1:5;
y=2*sqrt(2./(pi.*x))+2.*sqrt(2/pi.*x);
plot(x,y);
xlabel('abscissa axis');
ylabel('ordinate axis');
```

Source code for producing the graphs in Figures 6.3 and 6.4

```
% ==== Guessed Solution ====

% Setup Parameters

N=2*1e2;
x_max=8;
y_max=8;

% Setup Mesh

delta_x=x_max/N;
delta_y=y_max/N;
x_mesh=0:delta_x:x_max;
y_mesh=y_max:-delta_y:0;
[X_mesh,Y_mesh]=meshgrid(x_mesh,y_mesh);

% Guessed Solution for the sole case alpha=0, delta=0

V=1-exp(-2*min(X_mesh,Y_mesh))-2*min(X_mesh,Y_mesh).*exp(-(X_mesh+Y_mesh));

% Plot

surface(X_mesh,Y_mesh,V), shading interp;
rotate3d on;
grid off;
xlabel('x_1');
ylabel('x_2');
title(colorbar,'V(x_1,x_2)');
```

Appendix B

Source Code Numerical Approximation

B.1 Jacobi Iteration

```
% ========= Finite Difference Method through Jacobi Iteration ==========

% ======= Parameters =======

% Model parameters

x_max=8;
y_max=8;
alpha=1/3;   % other values: 1/2, 2/5

% Finite Difference Method parameters

N=2*1e2;     % other coefficients: 1, 1/2, 1/5, 1/10

% ======= Setup Mesh =======

dx=x_max/N;
dy=y_max/N;
x_mesh=0:dx:x_max;
y_mesh=y_max:-dy:0;

% ======= Boundary Conditions =======

% Creating array for the Push-Bottom Strategy (Democratic Policy)

V1=zeros(N+1,N+1);
V1(1,:)=(1-alpha)*(1-exp(-2*x_mesh))/(1-exp(-2*x_max))+alpha*(1-(1-exp(-2*x_mesh))/(1-exp(-2*x_max)));
V1(:,1)=alpha*(1-exp(-2*y_mesh))/(1-exp(-2*y_max));

% Creating array for the Push-Top Strategy (Republican Policy)

V2=V1;

% Creating array for the Mixed Strategy / Policy

V=V1;

% ======= Jacobi Iteration =======

TOL=1e-10;           % initialization of the tolerance

% ===== Push-Bottom Strategy (Democratic Policy) =====

% === Approximation of function V1 ===

iterations1=0;
```

```
distance1=TOL+1;
V1_new=V1;

while(distance1>TOL)
    iterations1=iterations1+1;
    c1=0;
    for i=2:N-1
        c1=c1+1;
        for j=2:N-c1
            V1_new(i,j)=((1-dx)*V1(i,j-1)+V1(i-1,j)+(1+dx)*V1(i,j+1)+V1(i+1,j))/4;
        end
    end
    for i=2:N
        j=N-i+2;
        V1_new(i,j)=((2-dx)*V1(i,j-1)+(2+dx)*V1(i-1,j))/4;
    end
    difference1=V1_new(2:N,2:N)-V1(2:N,2:N);
    distance1=max(max(difference1));
    V1(2:N,2:N)=V1_new(2:N,2:N);
end

display(iterations1);
V1=V1+fliplr(tril(flipud(V1),-1))';

% === Approximation of functions V1_x and V1_y ===

% = V1_x =

V1_x=zeros(N+1,N+1);
V1_x(1,:)=2*(1-2*alpha)*exp(-2*x_mesh)/(1-exp(-2*x_max));
V1_x(1,1)=0;

% Values of V1_x up to anti-diagonal-1

c1_x=0;
for j=2:N-1
    c1_x=c1_x+1;
    for i=2:N-c1_x
        V1_x(i,j)=(V1(i,j+1)-V1(i,j-1))/(2*dx);
    end
end

% Values of V1_x on anti-diagonal

for i=2:N
    j=N-i+2;
    V1_x(i,j)=(V1(i-1,j)-V1(i,j-1))/(2*dx);
end

V1_x=V1_x+fliplr(tril(flipud(V1_x),-1))';

% = V_y =

V1_y=zeros(N+1,N+1);
V1_y(:,1)=2*alpha*exp(-2*y_mesh)/(1-exp(-2*y_max));
V1_y(N+1,1)=0;
V1_y(1,N+1)=V1_x(1,N+1);

% Values of V1_y up to anti-diagonal-1

c1_y=0;
for j=2:N-1
    c1_y=c1_y+1;
    for i=2:N-c1_y
        V1_y(i,j)=(V1(i-1,j)-V1(i+1,j))/(2*dy);
    end
end

% Values of V1_y on anti-diagonal

for i=2:N
    j=N-i+2;
    V1_y(i,j)=(V1(i-1,j)-V1(i,j-1))/(2*dy);
end
```

```
V1_y=V1_y+fliplr(tril(flipud(V1_y),-1))';

% Difference between V1_x and V1_y

diff1=V1_x-V1_y;

for i=1:N+1
    for j=1:N+1
        if (diff1(i,j)<0)
            diff1(i,j)=0;
        else
            diff1(i,j)=1;
        end
    end
end

% ===== Push-Top Strategy (Republican Policy) =====

% === Approximation of function V2 ===

iterations2=0;
distance2=TOL+1;
V2_new=V2;

while(distance2>TOL)
    iterations2=iterations2+1;
    c2=0;
    for i=2:N-1
        c2=c2+1;
        for j=2:N-c2
            V2_new(i,j)=(V2(i,j-1)+(1+dy)*V2(i-1,j)+V2(i,j+1)+(1-dy)*V2(i+1,j))/4;
        end
    end
    for i=2:N
        j=N-i+2;
        V2_new(i,j)=((2-dy)*V2(i,j-1)+(2+dy)*V2(i-1,j))/4;
    end
    difference2=V2_new(2:N,2:N)-V2(2:N,2:N);
    distance2=max(max(difference2));
    V2(2:N,2:N)=V2_new(2:N,2:N);
end

display(iterations2);
V2=V2+fliplr(tril(flipud(V2),-1))';

% === Approximation of functions V2_x and V2_y ===

% = V2_x =

V2_x=zeros(N+1,N+1);
V2_x(1,:)=2*(1-2*alpha)*exp(-2*x_mesh)/(1-exp(-2*x_max));
V2_x(1,1)=0;

% Values of V2_x up to anti-diagonal-1

c2_x=0;
for j=2:N-1
    c2_x=c2_x+1;
    for i=2:N-c2_x
        V2_x(i,j)=(V2(i,j+1)-V2(i,j-1))/(2*dx);
    end
end

% Values of V2_x on anti-diagonal

for i=2:N
    j=N-i+2;
    V2_x(i,j)=(V2(i-1,j)-V2(i,j-1))/(2*dx);
end

V2_x=V2_x+fliplr(tril(flipud(V2_x),-1))';

% = V2_y =
```

```
V2_y=zeros(N+1,N+1);
V2_y(:,1)=2*alpha*exp(-2*y_mesh)/(1-exp(-2*y_max));
V2_y(N+1,1)=0;
V2_y(1,N+1)=V2_x(1,N+1);

% Values of V2_y up to anti-diagonal-1

c2_y=0;
for j=2:N-1
    c2_y=c2_y+1;
    for i=2:N-c2_y
        V2_y(i,j)=(V2(i-1,j)-V2(i+1,j))/(2*dy);
    end
end

% Values of V2_y on anti-diagonal

for i=2:N
    j=N-i+2;
    V2_y(i,j)=(V2(i-1,j)-V2(i,j-1))/(2*dy);
end

V2_y=V2_y+fliplr(tril(flipud(V2_y),-1))';

diff2=V2_y-V2_x;

for i=1:N+1
    for j=1:N+1
        if (diff2(i,j)<=0)
            diff2(i,j)=1;
        else
            diff2(i,j)=0;
        end
    end
end

% ===== Mixed Strategy / Policy =====

% === Approximation of function V ===

iterations=0;
distance=TOL+1;
V_new=V;

while(distance>TOL)
    iterations=iterations+1;
    c=0;
    for i=2:N-1
        c=c+1;
        for j=2:N-c
            V_new(i,j)=(V(i,j+1)+V(i,j-1)+V(i-1,j)+V(i+1,j)+dx*max(V(i,j+1)-V(i,j-1),V(i-1,j)-V(i+1,j)))/4;
        end
    end
    for i=2:N
        j=N-i+2;
        V_new(i,j)=((2-dx)*V(i,j-1)+(2+dx)*V(i-1,j))/4;
    end
    difference=V_new(2:N,2:N)-V(2:N,2:N);
    distance=max(max(difference));
    V(2:N,2:N)=V_new(2:N,2:N);
end

display(iterations);
V=V+fliplr(tril(flipud(V),-1))';

% === Approximation of functions V_x and V_y ===

% = V_x =

V_x=zeros(N+1,N+1);
V_x(1,:)=2*(1-2*alpha)*exp(-2*x_mesh)/(1-exp(-2*x_max));
V_x(1,1)=0;
```

```
% Values of V_x up to anti-diagonal-1

c_x=0;
for j=2:N-1
    c_x=c_x+1;
    for i=2:N-c_x
        V_x(i,j)=(V(i,j+1)-V(i,j-1))/(2*dx);
    end
end

% Values of V_x on anti-diagonal

for i=2:N
    j=N-i+2;
    V_x(i,j)=(V(i-1,j)-V(i,j-1))/(2*dx);
end

V_x=V_x+fliplr(tril(flipud(V_x),-1))';

% = V_y =

V_y=zeros(N+1,N+1);
V_y(:,1)=2*alpha*exp(-2*y_mesh)/(1-exp(-2*y_max));
V_y(N+1,1)=0;
V_y(1,N+1)=V_x(1,N+1);

% Values of V_y up to anti-diagonal-1

c_y=0;
for j=2:N-1
    c_y=c_y+1;
    for i=2:N-c_y
        V_y(i,j)=(V(i-1,j)-V(i+1,j))/(2*dy);
    end
end

% Values of V_y on anti-diagonal

for i=2:N
    j=N-i+2;
    V_y(i,j)=(V(i-1,j)-V(i,j-1))/(2*dy);
end

V_y=V_y+fliplr(tril(flipud(V_y),-1))';

% Difference between V_x and V_y

diff=V_x-V_y;

for i=1:N+1
    for j=1:N+1
        if (diff(i,j)<0)
            diff(i,j)=0;
        else
            diff(i,j)=1;
        end
    end
end

% ======= Plots =======

figure (1)

surface(x_mesh,y_mesh,diff1), shading interp;
rotate3d on;
colormap(jet);
grid off;
xlabel('x_1');
ylabel('x_2');

hold on;

figure(2)
```

```
surface(x_mesh,y_mesh,diff2), shading interp;
rotate3d on;
colormap(jet);
grid off;
xlabel('x_1');
ylabel('x_2');

hold on;

figure(3)

surface(x_mesh,y_mesh,diff), shading interp;
rotate3d on;
colormap(jet);
grid off;
xlabel('x_1');
ylabel('x_2');

hold on;

figure(4)

surface(x_mesh,y_mesh,V1), shading interp;
rotate3d on;
colormap(jet);
grid off;
xlabel('x_1');
ylabel('x_2');
zlabel('V_1(x_1,x_2)');
title(colorbar,'V_1(x_1,x_2)');

hold on;

figure(5)

surface(x_mesh,y_mesh,V2), shading interp;
rotate3d on;
colormap(jet);
grid off;
xlabel('x_1');
ylabel('x_2');
zlabel('V_2(x_1,x_2)');
title(colorbar,'V_2(x_1,x_2)');

hold on;

figure(6)

surface(x_mesh,y_mesh,V), shading interp;
rotate3d on;
colormap(jet);
grid off;
xlabel('x_1');
ylabel('x_2');
zlabel('V(x_1,x_2)');
title(colorbar,'V(x_1,x_2)');

hold off;
```

B.2 Successive over Relaxation

```
% ========= Finite Difference Method through Successive over Relaxation ==========

% ======= Parameters =======

% Model parameters

x_max=8;
y_max=8;
alpha=1/3;   % other values: 1/2, 2/5
```

```
% Finite Difference Method parameters

N=2*1e2;     % other coefficients: 1, 1/2, 1/5, 1/10
w=3/2;       % other values: 1/2, 1 (Gauss Seidel)

% ===== Setup Mesh =====

dx=x_max/N;
dy=y_max/N;
x_mesh=0:dx:x_max;
y_mesh=y_max:-dy:0;

% ======= Boundary Conditions =======

% Creating array for the Push-Bottom Strategy (Democratic Policy)

V1=zeros(N+1,N+1);
V1(1,:)=(1-alpha)*(1-exp(-2*x_mesh))/(1-exp(-2*x_max))+alpha*(1-(1-exp(-2*x_mesh))/(1-exp(-2*x_max)));
V1(:,1)=alpha*(1-exp(-2*y_mesh))/(1-exp(-2*y_max));

% Creating array for the Push-Top Strategy (Republican Policy)

V2=V1;

% Creating array for the Mixed Strategy / Policy

V=V1;

% ======= Successive over Relaxation =======

TOL=1e-10;           % initialization of the tolerance

% ===== Push-Bottom Strategy (Democratic Policy) =====

% === Approximation of function V ===

iterations1=0;
distance1=TOL+1;
V1_new=V1;

while(distance1>TOL)
    iterations1=iterations1+1;
    c1=0;
    for i=2:N-1
        c1=c1+1;
        for j=2:N-c1
            V1_new(i,j)=(1-w)*V1(i,j)+w*((1-dx)*V1_new(i,j-1)+V1_new(i-1,j)+(1+dx)*V1(i,j+1)+V1(i+1,j))/4;
        end
    end
    for i=2:N
        j=N-i+2;
        V1_new(i,j)=(1-w)*V1(i,j)+w*((2-dx)*V1_new(i,j-1)+(2+dx)*V1_new(i-1,j))/4;
    end
    difference1=V1_new(2:N,2:N)-V1(2:N,2:N);
    distance1=max(max(difference1));
    V1(2:N,2:N)=V1_new(2:N,2:N);
end

display(iterations1);
V1=V1+fliplr(tril(flipud(V1),-1))';

% === Approximation of functions V_x and V_y ===

% = V_x =

V1_x=zeros(N+1,N+1);
V1_x(1,:)=2*(1-2*alpha)*exp(-2*x_mesh)/(1-exp(-2*x_max));
V1_x(1,1)=0;

% Values of V_x up to anti-diagonal-1

c1_x=0;
for j=2:N-1
    c1_x=c1_x+1;
```

```
    for i=2:N-c1_x
        V1_x(i,j)=(V1(i,j+1)-V1(i,j-1))/(2*dx);
    end
end

% Values of V_x on anti-diagonal

for i=2:N
    j=N-i+2;
    V1_x(i,j)=(V1(i-1,j)-V1(i,j-1))/(2*dx);
end

V1_x=V1_x+fliplr(tril(flipud(V1_x),-1))';

% = V_y =

V1_y=zeros(N+1,N+1);
V1_y(:,1)=2*alpha*exp(-2*y_mesh)/(1-exp(-2*y_max));
V1_y(N+1,1)=0;
V1_y(1,N+1)=V1_x(1,N+1);

% Values of V_y up to anti-diagonal-1

c1_y=0;
for j=2:N-1
    c1_y=c1_y+1;
    for i=2:N-c1_y
        V1_y(i,j)=(V1(i-1,j)-V1(i+1,j))/(2*dy);
    end
end

% Values of V_y on anti-diagonal

for i=2:N
    j=N-i+2;
    V1_y(i,j)=(V1(i-1,j)-V1(i,j-1))/(2*dy);
end

V1_y=V1_y+fliplr(tril(flipud(V1_y),-1))';

% Difference between V_x and V_y

diff1=V1_x-V1_y;

for i=1:N+1
    for j=1:N+1
        if (diff1(i,j)<0)
            diff1(i,j)=0;
        else
            diff1(i,j)=1;
        end
    end
end

% ===== Push-Top Strategy (Republican Policy) =====

% === Approximation of function V ===

iterations2=0;
distance2=TOL+1;
V2_new=V2;

while(distance2>TOL)
    iterations2=iterations2+1;
    c2=0;
    for i=2:N-1
        c2=c2+1;
        for j=2:N-c2
            V2_new(i,j)=(1-w)*V2(i,j)+w*(V2_new(i,j-1)+(1+dy)*V2_new(i-1,j)+V2(i,j+1)+(1-dy)*V2(i+1,j))/4;
        end
    end
    for i=2:N
        j=N-i+2;
        V2_new(i,j)=(1-w)*V2(i,j)+w*((2-dy)*V2_new(i,j-1)+(2+dy)*V2_new(i-1,j))/4;
```

```
    end
    difference2=V2_new(2:N,2:N)-V2(2:N,2:N);
    distance2=max(max(difference2));
    V2(2:N,2:N)=V2_new(2:N,2:N);
end

display(iterations2);
V2=V2+fliplr(tril(flipud(V2),-1))';

% === Approximation of functions V_x and V_y ===

% = V_x =

V2_x=zeros(N+1,N+1);
V2_x(1,:)=2*(1-2*alpha)*exp(-2*x_mesh)/(1-exp(-2*x_max));
V2_x(1,1)=0;

% Values of V_x up to anti-diagonal-1

c2_x=0;
for j=2:N-1
    c2_x=c2_x+1;
    for i=2:N-c2_x
        V2_x(i,j)=(V2(i,j+1)-V2(i,j-1))/(2*dx);
    end
end

% Values of V_x on anti-diagonal

for i=2:N
    j=N-i+2;
    V2_x(i,j)=(V2(i-1,j)-V2(i,j-1))/(2*dx);
end

V2_x=V2_x+fliplr(tril(flipud(V2_x),-1))';

% = V_y =

V2_y=zeros(N+1,N+1);
V2_y(:,1)=2*exp(-2*y_mesh)/(1-exp(-2*y_max));
V2_y(N+1,1)=0;
V2_y(1,N+1)=V2_x(1,N+1);

% Values of V_y up to anti-diagonal-1

c2_y=0;
for j=2:N-1
    c2_y=c2_y+1;
    for i=2:N-c2_y
        V2_y(i,j)=(V2(i-1,j)-V2(i+1,j))/(2*dy);
    end
end

% Values of V_y on anti-diagonal

for i=2:N
    j=N-i+2;
    V2_y(i,j)=(V2(i-1,j)-V2(i,j-1))/(2*dy);
end

V2_y=V2_y+fliplr(tril(flipud(V2_y),-1))';

diff2=V2_y-V2_x;

for i=1:N+1
    for j=1:N+1
        if (diff2(i,j)<=0)
            diff2(i,j)=1;
        else
            diff2(i,j)=0;
        end
    end
end
```

```
% ===== Mixed Strategy / Policy =====

% === Approximation of function V ===

iterations=0;
distance=TOL+1;
V_new=V;

while(distance>TOL)
    iterations=iterations+1;
    c=0;
    for i=2:N-1
        c=c+1;
        for j=2:N-c
            V_new(i,j)=(1-w)*V(i,j)+w*(V(i,j+1)+V_new(i,j-1)+V_new(i-1,j)+V(i+1,j)+dx*max(V(i,j+1)-V_new(i,j-1),V_new(i-1,j)-V(i+1,j)))/4;
        end
    end
    for i=2:N
        j=N-i+2;
        V_new(i,j)=(1-w)*V(i,j)+w*((2-dx)*V_new(i,j-1)+(2+dx)*V_new(i-1,j))/4;
    end
    difference=V_new(2:N,2:N)-V(2:N,2:N);
    distance=max(max(difference));
    V(2:N,2:N)=V_new(2:N,2:N);
end

display(iterations);
V=V+fliplr(tril(flipud(V),-1))';

% === Approximation of functions V_x and V_y ===

% = V_x =

V_x=zeros(N+1,N+1);
V_x(1,:)=2*(1-2*alpha)*exp(-2*x_mesh)/(1-exp(-2*x_max));
V_x(1,1)=0;

% Values of V_x up to anti-diagonal-1

c_x=0;
for j=2:N-1
    c_x=c_x+1;
    for i=2:N-c_x
        V_x(i,j)=(V(i,j+1)-V(i,j-1))/(2*dx);
    end
end

% Values of V_x on anti-diagonal

for i=2:N
    j=N-i+2;
    V_x(i,j)=(V(i-1,j)-V(i,j-1))/(2*dx);
end

V_x=V_x+fliplr(tril(flipud(V_x),-1))';

% = V_y =

V_y=zeros(N+1,N+1);
V_y(:,1)=2*exp(-2*y_mesh)/(1-exp(-2*y_max));
V_y(N+1,1)=0;
V_y(1,N+1)=V_x(1,N+1);

% Values of V_y up to anti-diagonal-1

c_y=0;
for j=2:N-1
    c_y=c_y+1;
    for i=2:N-c_y
        V_y(i,j)=(V(i-1,j)-V(i+1,j))/(2*dy);
    end
end

% Values of V_y on anti-diagonal
```

```
for i=2:N
    j=N-i+2;
    V_y(i,j)=(V(i-1,j)-V(i,j-1))/(2*dy);
end

V_y=V_y+fliplr(tril(flipud(V_y),-1))';

% Difference between V_x and V_y

diff=V_x-V_y;

for i=1:N+1
    for j=1:N+1
        if (diff(i,j)<0)
            diff(i,j)=0;
        else
            diff(i,j)=1;
        end
    end
end

% ======= Plots =======

figure (1)

surface(x_mesh,y_mesh,diff1), shading interp;
rotate3d on;
colormap(jet);
grid off;
xlabel('x_1');
ylabel('x_2');

hold on;

figure(2)

surface(x_mesh,y_mesh,diff2), shading interp;
rotate3d on;
colormap(jet);
grid off;
xlabel('x_1');
ylabel('x_2');

hold on;

figure(3)

surface(x_mesh,y_mesh,diff), shading interp;
rotate3d on;
colormap(jet);
grid off;
xlabel('x_1');
ylabel('x_2');

hold on;

figure(4)

surface(x_mesh,y_mesh,V1), shading interp;
rotate3d on;
colormap(jet);
grid off;
xlabel('x_1');
ylabel('x_2');
zlabel('V_1(x_1,x_2)');
title(colorbar,'V_1(x_1,x_2)');

hold on;

figure(5)

surface(x_mesh,y_mesh,V2), shading interp;
rotate3d on;
```

```
colormap(jet);
grid off;
xlabel('x_1');
ylabel('x_2');
zlabel('V_2(x_1,x_2)');
title(colorbar,'V_2(x_1,x_2)');

hold on;

figure(6)

surface(x_mesh,y_mesh,V), shading interp;
rotate3d on;
colormap(jet);
grid off;
xlabel('x_1');
ylabel('x_2');
zlabel('V(x_1,x_2)');
title(colorbar,'V(x_1,x_2)');

hold off;
```

Appendix C

Source Code Monte Carlo Simulation

C.1 Two-Dimensional Case

```
% ======= Monte Carlo Simulation for the Two-Dimensional Model =======

% ===== Parameters =====

% Model parameters

sigma1=1;            % volatility of the first Brownian Motion process
sigma2=1;            % volatility of the second Brownian Motion process
delta=0;             % parameter determining range of drifts
alpha_1=0;           % parameter within the objective function
alpha_2=1/3;         %               - '' -
alpha_3=1/2;         %               - '' -

% Initial values for both companies at t=0

X1_t0=1e-5;           % initial value of the first company at t=0
X2_t0=2;              % initial value of the second company at t=0

% Monte Carlo Method parameters

n=1e2;               % number of discrete time points excluding t=0
N=1e4;               % number of Monte Carlo trials
T=1e1;               % time endpoint

% ===== Setup Mesh =====

dt=T/n;              % distance between equidistant time points
sq_dt=sqrt(dt);      % root of the previous distance
t_mesh=0:dt:T;       % discrete time points

% ===== Monte Carlo Simulation =====

% Generation of standard normally distributed random numbers

Z1=[zeros(1,N);randn(n,N)];
Z2=[zeros(1,N);randn(n,N)];

% === No Strategy (or Policy) ===

X1=zeros(n+1,N);
X1(1,:)=X1_t0;         % preallocation of arrays
X2=zeros(n+1,N);
X2(1,:)=X2_t0;

% Generation of the two Brownian Motion processes via Euler Scheme
```

```
X1=cumsum(X1+sigma1*sq_dt*Z1);
X2=cumsum(X2+sigma2*sq_dt*Z2);

% Setting values less than 0 and following values to be equal to 0

for j=1:N
    for i=2:n+1
        if (X1(i,j)<=0)
            X1(i:n+1,j)=0;
        else
        end
        if(X2(i,j)<=0)
            X2(i:n+1,j)=0;
        else
        end
    end
end

% Number of times (<=N) that company X1 remains solvent while simultaneously
% company X2 goes bankrupt

counter1x=sum(X1>0 & X2==0,2);

% Number of times (<=N) that company X1 goes bankrupt while simultaneously
% company X2 remains solvent

counter1y=sum(X1==0 & X2>0,2);

% Number of times (<=N) that both companies remain solvent

counter2=sum(X1>0 & X2>0,2);

% Objective functions

obj_fun_1=(counter1x+counter1y+2*counter2)/N;
obj_fun_2=(alpha_1*(counter1x+counter1y)+(1-alpha_1)*counter2)/N;
obj_fun_3=(alpha_2*(counter1x+counter1y)+(1-alpha_2)*counter2)/N;
obj_fun_4=(alpha_3*(counter1x+counter1y)+(1-alpha_3)*counter2)/N;

% === Arbitrary Strategy ===

% Generation of random numbers that are drawn from a uniform distribution in
% the interval (-delta , 1-(-delta))

mu1=[zeros(1,N);-delta+(1-2*(-delta))*rand(n,N)];
mu2=ones(n+1,N)-mu1-[ones(1,N);zeros(n,N)];

X1_a=zeros(n+1,N);
X1_a(1,:)=X1_t0;         % preallocation of arrays
X2_a=zeros(n+1,N);
X2_a(1,:)=X2_t0;

for j=1:N
    for i=2:n+1
        if (X1_a(i-1,j)>0) && (X2_a(i-1,j)>0)
            X1_a(i,j)=X1_a(i-1,j)+mu1(i,j)*dt+sigma1*sq_dt*Z1(i,j);
            X2_a(i,j)=X2_a(i-1,j)+mu2(i,j)*dt+sigma2*sq_dt*Z2(i,j);
                    if (X1_a(i,j)<0)
                        X1_a(i,j)=0;
                    end
                    if (X2_a(i,j)<0)
                        X2_a(i,j)=0;
                    end
        elseif (X1_a(i-1,j)==0) && (X2_a(i-1,j)>0)
            X1_a(i,j)=0;
            X2_a(i,j)=X2_a(i-1,j)+dt+sigma2*sq_dt*Z2(i,j);
            if (X2_a(i,j)<0)
                X2_a(i,j)=0;
            end
        elseif (X1_a(i-1,j)>0) && (X2_a(i-1,j)==0)
            X1_a(i,j)=X1_a(i-1,j)+dt+sigma1*sq_dt*Z1(i,j);
            X2_a(i,j)=0;
            if (X1_a(i,j)<0)
                X1_a(i,j)=0;
```

```
            end
        else
            X1_a(i,j)=0;
            X2_a(i,j)=0;
        end
    end
end

% Number of times (<=N) that company X1 remains solvent while
% simultaneously company X2 goes bankrupt

counter1x_a=sum(X1_a>0 & X2_a==0,2);

% Number of times (<=N) that company X1 goes bankrupt while simultaneously
% company X2 remains solvent

counter1y_a=sum(X1_a==0 & X2_a>0,2);

% Number of times (<=N) that both companies remain solvent

counter2_a=sum(X1_a>0 & X2_a>0,2);

% Objective functions

obj_fun_a_1=(counter1x_a+counter1y_a+2*counter2_a)/N;
obj_fun_a_2=(alpha_1*(counter1x_a+counter1y_a)+(1-alpha_1)*counter2_a)/N;
obj_fun_a_3=(alpha_2*(counter1x_a+counter1y_a)+(1-alpha_2)*counter2_a)/N;
obj_fun_a_4=(alpha_3*(counter1x_a+counter1y_a)+(1-alpha_3)*counter2_a)/N;

% === Push-Bottom Strategy (Democratic Policy) ===

X1_d=zeros(n+1,N);
X1_d(1,:)=X1_t0;          % preallocation of arrays
X2_d=zeros(n+1,N);
X2_d(1,:)=X2_t0;

for j=1:N
    for i=2:n+1
        if (X1_d(i-1,j)>0) && (X2_d(i-1,j)>0)
            X1_d(i,j)=X1_d(i-1,j)+sigma1*sq_dt*Z1(i,j);
            X2_d(i,j)=X2_d(i-1,j)+sigma2*sq_dt*Z2(i,j);
                if (X1_d(i,j)<=X2_d(i,j))
                    X1_d(i,j)=X1_d(i-1,j)+dt+sigma1*sq_dt*Z1(i,j);
                    if (X1_d(i,j)<0)
                        X1_d(i,j)=0;
                    end
                    if (X2_d(i,j)<0)
                        X2_d(i,j)=0;
                    end
                else
                    X2_d(i,j)=X2_d(i-1,j)+dt+sigma2*sq_dt*Z2(i,j);
                    if (X1_d(i,j)<0)
                        X1_d(i,j)=0;
                    end
                    if (X2_d(i,j)<0)
                        X2_d(i,j)=0;
                    end
                end
        elseif (X1_d(i-1,j)==0) && (X2_d(i-1,j)>0)
            X1_d(i,j)=0;
            X2_d(i,j)=X2_d(i-1,j)+dt+sigma2*sq_dt*Z2(i,j);
            if (X2_d(i,j)<0)
                X2_d(i,j)=0;
            end
        elseif (X1_d(i-1,j)>0) && (X2_d(i-1,j)==0)
            X1_d(i,j)=X1_d(i-1,j)+dt+sigma1*sq_dt*Z1(i,j);
            X2_d(i,j)=0;
            if (X1_d(i,j)<0)
                X1_d(i,j)=0;
            end
        else
            X1_d(i,j)=0;
            X2_d(i,j)=0;
        end
```

```
    end
end

% Number of times (<=N) that company X1 remains solvent while simultaneously
% company X2 goes bankrupt

counter1x_d=sum(X1_d>0 & X2_d==0,2);

% Number of times (<=N) that company X1 goes bankrupt while simultaneously
% company X2 remains solvent

counter1y_d=sum(X1_d==0 & X2_d>0,2);

% Number of times (<=N) that both companies remain solvent

counter2_d=sum(X1_d>0 & X2_d>0,2);

% Objective functions

obj_fun_d_1=(counter1x_d+counter1y_d+2*counter2_d)/N;
obj_fun_d_2=(alpha_1*(counter1x_d+counter1y_d)+(1-alpha_1)*counter2_d)/N;
obj_fun_d_3=(alpha_2*(counter1x_d+counter1y_d)+(1-alpha_2)*counter2_d)/N;
obj_fun_d_4=(alpha_3*(counter1x_d+counter1y_d)+(1-alpha_3)*counter2_d)/N;

% === Push-Top Strategy (Republican Policy) ===

X1_r=zeros(n+1,N);
X1_r(1,:)=X1_t0;          % preallocation of arrays
X2_r=zeros(n+1,N);
X2_r(1,:)=X2_t0;

for j=1:N
    for i=2:n+1
        if (X1_r(i-1,j)>0) && (X2_r(i-1,j)>0)
            X1_r(i,j)=X1_r(i-1,j)+sigma1*sq_dt*Z1(i,j);
            X2_r(i,j)=X2_r(i-1,j)+sigma2*sq_dt*Z2(i,j);
                if (X1_r(i,j)>X2_r(i,j))
                    X1_r(i,j)=X1_r(i-1,j)+dt+sigma1*sq_dt*Z1(i,j);
                    if (X1_r(i,j)<0)
                        X1_r(i,j)=0;
                    end
                    if (X2_r(i,j)<0)
                        X2_r(i,j)=0;
                    end
                else
                    X2_r(i,j)=X2_r(i-1,j)+dt+sigma2*sq_dt*Z2(i,j);
                    if (X1_r(i,j)<0)
                        X1_r(i,j)=0;
                    end
                    if (X2_r(i,j)<0)
                        X2_r(i,j)=0;
                    end
                end
        elseif (X1_r(i-1,j)==0) && (X2_r(i-1,j)>0)
            X1_r(i,j)=0;
            X2_r(i,j)=X2_r(i-1,j)+dt+sigma2*sq_dt*Z2(i,j);
            if (X2_r(i,j)<0)
                X2_r(i,j)=0;
            end
        elseif (X1_r(i-1,j)>0) && (X2_r(i-1,j)==0)
            X1_r(i,j)=X1_r(i-1,j)+dt+sigma1*sq_dt*Z1(i,j);
            X2_r(i,j)=0;
            if (X1_r(i,j)<0)
                X1_r(i,j)=0;
            end
        else
            X1_r(i,j)=0;
            X2_r(i,j)=0;
        end
    end
end

% Number of times (<=N) that company X1 remains solvent while simultaneously
% company X2 goes bankrupt
```

```
counter1x_r=sum(X1_r>0 & X2_r==0,2);

% Number of times (<=N) that company X1 goes bankrupt while simultaneously
% company X2 remains solvent

counter1y_r=sum(X1_r==0 & X2_r>0,2);

% Number of times (<=N) that both companies remain solvent

counter2_r=sum(X1_r>0 & X2_r>0,2);

% Objective functions

obj_fun_r_1=(counter1x_r+counter1y_r+2*counter2_r)/N;
obj_fun_r_2=(alpha_1*(counter1x_r+counter1y_r)+(1-alpha_1)*counter2_r)/N;
obj_fun_r_3=(alpha_2*(counter1x_r+counter1y_r)+(1-alpha_2)*counter2_r)/N;
obj_fun_r_4=(alpha_3*(counter1x_r+counter1y_r)+(1-alpha_3)*counter2_r)/N;

% ===== Plots =====

figure(1);

subplot(2,2,1);
plot(t_mesh,obj_fun_1,'-k',t_mesh,obj_fun_a_1,'-.g',t_mesh,obj_fun_d_1,':r',t_mesh,obj_fun_r_1,'--b');
legend('no strategy','arbitrary strategy','push-bottom strategy','push-top strategy');
xlabel('time');
ylabel('objective function');
title('Expected number of companies that remain solvent');

subplot(2,2,2);
plot(t_mesh,obj_fun_2,'-k',t_mesh,obj_fun_a_2,'-.g',t_mesh,obj_fun_d_2,':r',t_mesh,obj_fun_r_2,'--b');
legend('no strategy','arbitrary strategy','push-bottom strategy','push-top strategy');
axis([0 T 0 1.5])
xlabel('time');
ylabel('objective function');
title('Probability that both companies survive');

subplot(2,2,3);
plot(t_mesh,obj_fun_3,'-k',t_mesh,obj_fun_a_3,'-.g',t_mesh,obj_fun_d_3,':r',t_mesh,obj_fun_r_3,'--b');
legend('no strategy','arbitrary strategy','push-bottom strategy','push-top strategy');
xlabel('time');
ylabel('objective function');
title('Expected number of companies that remain solvent for alpha=1/3');

subplot(2,2,4);
plot(t_mesh,obj_fun_4,'-k',t_mesh,obj_fun_a_4,'-.g',t_mesh,obj_fun_d_4,':r',t_mesh,obj_fun_r_4,'--b');
legend('no strategy','arbitrary strategy','push-bottom strategy','push-top strategy');
axis([0 T 0.33 0.55]);
xlabel('time');
ylabel('objective function');
title('Expected number that all companies remain solvent for alpha=1/2');
```

C.2 Three-Dimensional Case

```
% ======= Monte Carlo Simulation for the Three-Dimensional Model =======

% ===== Parameters =====

% Model parameters

sigma1=1;             % volatility of the first Brownian Motion process
sigma2=1;             % volatility of the second Brownian Motion process
sigma3=1;             % volatility of the third Brownian Motion process

% Initial values for both companies at t=0

X1_t0=1e-1;           % initial value of the first company at t=0
X2_t0=1;              % initial value of the second company at t=0
X3_t0=2;              % initial value of the third company at t=0
```

```
% Monte Carlo Method parameters

n=1e2;              % number of discrete time points excluding t=0
N=1e4;              % number of Monte Carlo trials
T=1e1;              % time endpoint

% ===== Setup Mesh =====

dt=T/n;             % distance between equidistant time points
sq_dt=sqrt(dt);     % root of the previous distance
t_mesh=0:dt:T;      % discrete time points

% ===== Monte Carlo Simulation =====

% Generation of standard normally distributed random numbers

Z1=[zeros(1,N);randn(n,N)];
Z2=[zeros(1,N);randn(n,N)];
Z3=[zeros(1,N);randn(n,N)];

% === No Strategy (or Policy) ===

X1=zeros(n+1,N);
X1(1,:)=X1_t0;
X2=zeros(n+1,N);        % preallocation of arrays
X2(1,:)=X2_t0;
X3=zeros(n+1,N);
X3(1,:)=X3_t0;

% Generation of the three Brownian Motion processes via Euler Scheme

X1=cumsum(X1+sigma1*sq_dt*Z1);
X2=cumsum(X2+sigma2*sq_dt*Z2);
X3=cumsum(X3+sigma3*sq_dt*Z3);

% Setting values less than 0 and following values to be equal to 0

for j=1:N
    for i=2:n+1
        if (X1(i,j)<=0)
            X1(i:n+1,j)=0;
        else
        end
        if (X2(i,j)<=0)
            X2(i:n+1,j)=0;
        else
        end
        if (X3(i,j)<=0)
            X3(i:n+1,j)=0;
        else
        end
    end
end

% Number of times (<=N) that company X1 remains solvent while simultaneously
% companies X2 and X3 go bankrupt

counter1x=sum(X1>0 & X2==0 & X3==0,2);

% Number of times (<=N) that company X2 remains solvent while simultaneously
% companies X1 and X3 go bankrupt

counter1y=sum(X1==0 & X2>0 & X3==0,2);

% Number of times (<=N) that company X3 remains solvent while simultaneously
% companies X1 and X2 go bankrupt

counter1z=sum(X1==0 & X2==0 & X3>0,2);

% Number of times (<=N) that companies X1 and X2 remain solvent
% while simultaneously company X3 goes bankrupt

counter2xy=sum(X1>0 & X2>0 & X3==0,2);
```

```
% Number of times (<=N) that companies X1 and X3 remain solvent
% while simultaneously company X2 goes bankrupt

counter2xz=sum(X1>0 & X2==0 & X3>0,2);

% Number of times (<=N) that companies X2 and X3 remain solvent
% while simultaneously company X1 goes bankrupt

counter2yz=sum(X1==0 & X2>0 & X3>0,2);

% Number of times (<=N) that all companies remain solvent

counter3=sum(X1>0 & X2>0 & X3>0,2);

% Objective functions

obj_fun_1=(counter1x+counter1y+counter1z+2*(counter2xy+counter2xz+counter2yz)+3*counter3)/N;
obj_fun_2=counter3/N;

% === Push-Bottom Strategy (Democratic Policy) ===

X1_d=zeros(n+1,N);
X1_d(1,:)=X1_t0;
X2_d=zeros(n+1,N);       % preallocation of arrays
X2_d(1,:)=X2_t0;
X3_d=zeros(n+1,N);
X3_d(1,:)=X3_t0;

for j=1:N
    for i=2:n+1
       if (X3_d(i-1,j)>0)
           if (X1_d(i-1,j)>0) && (X2_d(i-1,j)>0)
               X1_d(i,j)=X1_d(i-1,j)+sigma1*sq_dt*Z1(i,j);
               X2_d(i,j)=X2_d(i-1,j)+sigma2*sq_dt*Z2(i,j);
               X3_d(i,j)=X3_d(i-1,j)+sigma3*sq_dt*Z3(i,j);
               if (X1_d(i,j)<=X2_d(i,j)<=X3_d(i,j)) || (X1_d(i,j)<=X3_d(i,j)<=X2_d(i,j))
                   X1_d(i,j)=X1_d(i-1,j)+dt+sigma1*sq_dt*Z1(i,j);
                   if (X1_d(i,j)<0)
                       X1_d(i,j)=0;
                   end
                   if (X2_d(i,j)<0)
                       X2_d(i,j)=0;
                   end
                   if (X3_d(i,j)<0)
                       X3_d(i,j)=0;
                   end
               elseif (X2_d(i,j)<=X1_d(i,j)<=X3_d(i,j)) || (X2_d(i,j)<=X3_d(i,j)<=X1_d(i,j))
                   X2_d(i,j)=X2_d(i-1,j)+dt+sigma2*sq_dt*Z2(i,j);
                   if (X1_d(i,j)<0)
                       X1_d(i,j)=0;
                   end
                   if (X2_d(i,j)<0)
                       X2_d(i,j)=0;
                   end
                   if (X3_d(i,j)<0)
                       X3_d(i,j)=0;
                   end
               else
                   X3_d(i,j)=X3_d(i-1,j)+dt+sigma3*sq_dt*Z3(i,j);
                   if (X1_d(i,j)<0)
                       X1_d(i,j)=0;
                   end
                   if (X2_d(i,j)<0)
                       X2_d(i,j)=0;
                   end
                   if (X3_d(i,j)<0)
                       X3_d(i,j)=0;
                   end
               end
           elseif (X1_d(i-1,j)==0) && (X2_d(i-1,j)>0)
               X1_d(i,j)=0;
               X2_d(i,j)=X2_d(i-1,j)+sq_dt*sigma2*Z2(i,j);
               X3_d(i,j)=X3_d(i-1,j)+sq_dt*sigma3*Z3(i,j);
               if (X2_d(i,j)<=X3_d(i,j))
```

```
            X2_d(i,j)=X2_d(i-1,j)+dt+sq_dt*sigma2*Z2(i,j);
            if (X2_d(i,j)<0)
                X2_d(i,j)=0;
            end
            if (X3_d(i,j)<0)
                X3_d(i,j)=0;
            end
        else
            X3_d(i,j)=X3_d(i-1,j)+dt+sq_dt*sigma3*Z3(i,j);
            if (X2_d(i,j)<0)
                X2_d(i,j)=0;
            end
            if (X3_d(i,j)<0)
                X3_d(i,j)=0;
            end
        end
    elseif (X1_d(i-1,j)>0) && (X2_d(i-1,j)==0)
        X1_d(i,j)=X1_d(i-1,j)+sigma1*sq_dt*Z1(i,j);
        X2_d(i,j)=0;
        X3_d(i,j)=X3_d(i-1,j)+sigma3*sq_dt*Z3(i,j);
        if (X1_d(i,j)<=X3_d(i,j))
            X1_d(i,j)=X1_d(i-1,j)+dt+sigma1*sq_dt*Z1(i,j);
            if (X1_d(i,j)<0)
                X1_d(i,j)=0;
            end
            if (X3_d(i,j)<0)
                X3_d(i,j)=0;
            end
        else
            X3_d(i,j)=X3_d(i-1,j)+dt+sigma3*sq_dt*Z3(i,j);
            if (X1_d(i,j)<0)
                X1_d(i,j)=0;
            end
            if (X3_d(i,j)<0)
                X3_d(i,j)=0;
            end
        end
    else
        X1_d(i,j)=0;
        X2_d(i,j)=0;
        X3_d(i,j)=X3_d(i-1,j)+dt+sigma3*sq_dt*Z3(i,j);
        if (X3_d(i,j)<0)
            X3_d(i,j)=0;
        end
    end
else
    X3_d(i,j)=0;
    if (X1_d(i-1,j)>0) && (X2_d(i-1,j)>0)
        X1_d(i,j)=X1_d(i-1,j)+sigma1*sq_dt*Z1(i,j);
        X2_d(i,j)=X2_d(i-1,j)+sigma2*sq_dt*Z2(i,j);
        if (X1_d(i,j)<=X2_d(i,j))
            X1_d(i,j)=X1_d(i-1,j)+dt+sigma1*sq_dt*Z1(i,j);
            if (X1_d(i,j)<0)
                X1_d(i,j)=0;
            end
            if (X2_d(i,j)<0)
                X2_d(i,j)=0;
            end
        else
            X2_d(i,j)=X2_d(i-1,j)+dt+sigma2*sq_dt*Z2(i,j);
            if (X1_d(i,j)<0)
                X1_d(i,j)=0;
            end
            if (X2_d(i,j)<0)
                X2_d(i,j)=0;
            end
        end
    elseif (X1_d(i-1,j)==0) && (X2_d(i-1,j)>0)
        X1_d(i,j)=0;
        X2_d(i,j)=X2_d(i-1,j)+dt+sigma2*sq_dt*Z2(i,j);
        if (X2_d(i,j)<0)
            X2_d(i,j)=0;
        end
    elseif (X1_d(i-1,j)>0) && (X2_d(i-1,j)==0)
```

```
                X1_d(i,j)=X1_d(i-1,j)+dt+sigma1*sq_dt*Z1(i,j);
                X2_d(i,j)=0;
                if (X1_d(i,j)<0)
                    X1_d(i,j)=0;
                end
            else
                X1_d(i,j)=0;
                X2_d(i,j)=0;
            end
        end
    end
end

% Number of times (<=N) that company X1 remains solvent while simultaneously
% companies X2 and X3 go bankrupt

counter1x_d=sum(X1_d>0 & X2_d==0 & X3_d==0,2);

% Number of times (<=N) that company X2 remains solvent while simultaneously
% companies X1 and X3 go bankrupt

counter1y_d=sum(X1_d==0 & X2_d>0 & X3_d==0,2);

% Number of times (<=N) that company X3 remains solvent while simultaneously
% companies X1 and X2 go bankrupt

counter1z_d=sum(X1_d==0 & X2_d==0 & X3_d>0,2);

% Number of times (<=N) that companies X1 and X2 remain solvent while
% simultaneously company X3 goes bankrupt

counter2xy_d=sum(X1_d>0 & X2_d>0 & X3_d==0,2);

% Number of times (<=N) that companies X1 and X3 remain solvent
% while simultaneously company X2 goes bankrupt

counter2xz_d=sum(X1_d>0 & X2_d==0 & X3_d>0,2);

% Number of times (<=N) that companies X2 and X3 remain solvent
% while simultaneously company X1 goes bankrupt

counter2yz_d=sum(X1_d==0 & X2_d>0 & X3_d>0,2);

% Number of times (<=N) that all companies remain solvent

counter3_d=sum(X1_d>0 & X2_d>0 & X3_d>0,2);

% Objective functions

obj_fun_d_1=(counter1x_d+counter1y_d+counter1z_d+2*(counter2xy_d+counter2xz_d+counter2yz_d)+3*counter3_d)/N;
obj_fun_d_2=counter3_d/N;

% === Push-Top Strategy (Republican Policy) ===

X1_r=zeros(n+1,N);
X1_r(1,:)=X1_t0;
X2_r=zeros(n+1,N);       % preallocation of arrays
X2_r(1,:)=X2_t0;
X3_r=zeros(n+1,N);
X3_r(1,:)=X3_t0;

for j=1:N
    for i=2:n+1
       if (X3_r(i-1,j)>0)
            if (X1_r(i-1,j)>0) && (X2_r(i-1,j)>0)
                X1_r(i,j)=X1_r(i-1,j)+sigma1*sq_dt*Z1(i,j);
                X2_r(i,j)=X2_r(i-1,j)+sigma2*sq_dt*Z2(i,j);
                X3_r(i,j)=X3_r(i-1,j)+sigma3*sq_dt*Z3(i,j);
                if (X1_r(i,j)>=X2_r(i,j)>=X3_r(i,j)) || (X1_r(i,j)>=X3_r(i,j)>=X2_r(i,j))
                    X1_r(i,j)=X1_r(i-1,j)+dt+sigma1*sq_dt*Z1(i,j);
                    if (X1_r(i,j)<0)
                        X1_r(i,j)=0;
                    end
                    if (X2_r(i,j)<0)
```

```
            X2_r(i,j)=0;
        end
        if (X3_r(i,j)<0)
            X3_r(i,j)=0;
        end
    elseif (X2_r(i,j)>=X1_r(i,j)>=X3_r(i,j)) || (X2_r(i,j)>=X3_r(i,j)>=X1_r(i,j))
        X2_r(i,j)=X2_r(i-1,j)+dt+sigma2*sq_dt*Z2(i,j);
        if (X1_r(i,j)<0)
            X1_r(i,j)=0;
        end
        if (X2_r(i,j)<0)
            X2_r(i,j)=0;
        end
        if (X3_r(i,j)<0)
            X3_r(i,j)=0;
        end
    else
        X3_r(i,j)=X3_r(i-1,j)+dt+sigma3*sq_dt*Z3(i,j);
        if (X1_r(i,j)<0)
            X1_r(i,j)=0;
        end
        if (X2_r(i,j)<0)
            X2_r(i,j)=0;
        end
        if (X3_r(i,j)<0)
            X3_r(i,j)=0;
        end
    end
elseif (X1_r(i-1,j)==0) && (X2_r(i-1,j)>0)
    X1_r(i,j)=0;
    X2_r(i,j)=X2_r(i-1,j)+sq_dt*sigma2*Z2(i,j);
    X3_r(i,j)=X3_r(i-1,j)+sq_dt*sigma3*Z3(i,j);
    if (X2_r(i,j)>=X3_r(i,j))
        X2_r(i,j)=X2_r(i-1,j)+dt+sq_dt*sigma2*Z2(i,j);
        if (X2_r(i,j)<0)
            X2_r(i,j)=0;
        end
        if (X3_r(i,j)<0)
            X3_r(i,j)=0;
        end
    else
        X3_r(i,j)=X3_r(i-1,j)+dt+sq_dt*sigma3*Z3(i,j);
        if (X2_r(i,j)<0)
            X2_r(i,j)=0;
        end
        if (X3_r(i,j)<0)
            X3_r(i,j)=0;
        end
    end
elseif (X1_r(i-1,j)>0) && (X2_r(i-1,j)==0)
    X1_r(i,j)=X1_r(i-1,j)+sigma1*sq_dt*Z1(i,j);
    X2_r(i,j)=0;
    X3_r(i,j)=X3_r(i-1,j)+sigma3*sq_dt*Z3(i,j);
    if (X1_r(i,j)>=X3_r(i,j))
        X1_r(i,j)=X1_r(i-1,j)+dt+sigma1*sq_dt*Z1(i,j);
        if (X1_r(i,j)<0)
            X1_r(i,j)=0;
        end
        if (X3_r(i,j)<0)
            X3_r(i,j)=0;
        end
    else
        X3_r(i,j)=X3_r(i-1,j)+dt+sigma3*sq_dt*Z3(i,j);
        if (X1_r(i,j)<0)
            X1_r(i,j)=0;
        end
        if (X3_r(i,j)<0)
            X3_r(i,j)=0;
        end
    end
else
    X1_r(i,j)=0;
    X2_r(i,j)=0;
    X3_r(i,j)=X3_r(i-1,j)+dt+sigma3*sq_dt*Z3(i,j);
```

```
            if (X3_r(i,j)<0)
                X3_r(i,j)=0;
            end
        end
    else
        X3_r(i,j)=0;
        if (X1_r(i-1,j)>0) && (X2_r(i-1,j)>0)
            X1_r(i,j)=X1_r(i-1,j)+sigma1*sq_dt*Z1(i,j);
            X2_r(i,j)=X2_r(i-1,j)+sigma2*sq_dt*Z2(i,j);
            if (X1_r(i,j)>=X2_r(i,j))
                X1_r(i,j)=X1_r(i-1,j)+dt+sigma1*sq_dt*Z1(i,j);
                if (X1_r(i,j)<0)
                    X1_r(i,j)=0;
                end
                if (X2_r(i,j)<0)
                    X2_r(i,j)=0;
                end
            else
                X2_r(i,j)=X2_r(i-1,j)+dt+sigma2*sq_dt*Z2(i,j);
                if (X1_r(i,j)<0)
                    X1_r(i,j)=0;
                end
                if (X2_r(i,j)<0)
                    X2_r(i,j)=0;
                end
            end
        elseif (X1_r(i-1,j)==0) && (X2_r(i-1,j)>0)
            X1_r(i,j)=0;
            X2_r(i,j)=X2_r(i-1,j)+dt+sigma2*sq_dt*Z2(i,j);
            if (X2_r(i,j)<0)
                X2_r(i,j)=0;
            end
        elseif (X1_r(i-1,j)>0) && (X2_r(i-1,j)==0)
            X1_r(i,j)=X1_r(i-1,j)+dt+sigma1*sq_dt*Z1(i,j);
            X2_r(i,j)=0;
            if (X1_r(i,j)<0)
                X1_r(i,j)=0;
            end
        else
            X1_r(i,j)=0;
            X2_r(i,j)=0;
        end
    end
  end
end

% Number of times (<=N) that company X1 remains solvent while simultaneously
% companies X2 and X3 go bankrupt

counter1x_r=sum(X1_r>0 & X2_r==0 & X3_r==0,2);

% Number of times (<=N) that company X2 remains solvent while simultaneously
% companies X1 and X3 go bankrupt

counter1y_r=sum(X1_r==0 & X2_r>0 & X3_r==0,2);

% Number of times (<=N) that company X3 remains solvent while simultaneously
% companies X1 and X2 go bankrupt

counter1z_r=sum(X1_r==0 & X2_r==0 & X3_r>0,2);

% Number of times (<=N) that companies X1 and X2 remain solvent while
% simultaneously company X3 goes bankrupt

counter2xy_r=sum(X1_r>0 & X2_r>0 & X3_r==0,2);

% Number of times (<=N) that companies X1 and X3 remain solvent while
% simultaneously company X2 goes bankrupt

counter2xz_r=sum(X1_r>0 & X2_r==0 & X3_r>0,2);

% Number of times (<=N) that companies X2 and X3 remain solvent while
% simultaneously company X1 goes bankrupt
```

```
counter2yz_r=sum(X1_r==0 & X2_r>0 & X3_r>0,2);

% Number of times (<=N) that all companies remain solvent

counter3_r=sum(X1_r>0 & X2_r>0 & X3_r>0,2);

% Objective functions

obj_fun_r_1=(counter1x_r+counter1y_r+counter1z_r+2*(counter2xy_r+counter2xz_r+counter2yz_r)+3*counter3_r)/N;
obj_fun_r_2=counter3_r/N;

% ===== Plots =====

figure(1)

plot(t_mesh,obj_fun_1,'-k',t_mesh,obj_fun_d_1,':r',t_mesh,obj_fun_r_1,'--b');
legend('no strategy','push-bottom strategy','push-top strategy');
xlabel('time');
ylabel('objective function');
title('Expected number of companies that remain solvent');

hold on;

figure(2)

plot(t_mesh,obj_fun_2,'-k',t_mesh,obj_fun_d_2,':r',t_mesh,obj_fun_r_2,'--b');
legend('no strategy','push-bottom strategy','push-top strategy');
xlabel('time');
ylabel('objective function');
title('Probability that all companies remain solvent');

hold off;
```

Bibliography

[1] Huyên Pham. *Continuous-time Stochastic Control and Optimization with Financial Applications.* Springer-Verlag, Paris, 2009. 4, 5, 7, 8, 10, 15, 19, 22, 23, 26

[2] Vincenzo Capasso, David Bakstein. *An Introduction to Continuous-Time Stochastic Processes.* Birkhäuser, Boston Basel Berlin, 2012. 3

[3] Hui Wang. *Monte Carlo Simulation with Applications to Finance.* Chapman Hall/CRC Financial Mathematics, 2012. 7, 39, 40

[4] Nicolas Pivault. *An Elementary Introduction to Stochastic Interest Rate Modeling (2nd Edition).* World Scientific, 2012. 6

[5] Daniel Revuz, Marc Yor. *Continuous Martingales and Brownian Motion.* Springer-Verlag, 1994. 7

[6] Joseph Chang. *Stochastic Processes.* Yale University, 2007. 8

[7] Ioannis Karatzas. *Optimization Problems in the Theory of Continuous Trading.* Journal of the Society for Industrial and Applied Mathematics, 6:1221-1259, 1989. 27

[8] Andy Dahl. *A Rigorous Introduction to Brownian Motion.* University of Chicago, 2010. 9

[9] Henk Tijms. *Understanding Probability.* Cambridge University Press, 2012. 9

[10] Fima C. Klebaner. *Introduction to Stochastic Calculus with Applications.* Imperial College Press, 2005. 10

[11] Bernt Karsten Øksendal. *Stochastic Differential Equations. An Introduction with Applications.* Springer-Verlag, Heidelberg New York, 2000. 11, 12, 13

[12] Xuerong Mao. *Stochastic Differential Equations and Applications.* Woodhead Publishing, Oxford Cambridge Philadelphia New Delhi, 2007. 13, 14

[13] Wendell Helms Fleming, Halil Mete Soner. *Controlled Markov Processes and Viscosity Solutions.* Springer-Verlag USA, 2006. 15

[14] Richard Bellman. *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957. 18

[15] John Von Neumann, Oskar Morgenstern. *Theory of Games and Economic Behavior.* Princeton University Press, Princeton, NJ, 1953. 48

[16] Henry McKean. *Appendix: A Free Boundary Problem for the Heat Equation Arising from a Problem in Mathematical Economics.* Industrial Management Rev., 6, 32-39, 1965. 48

[17] Goran Peskir. *Principle of Smooth Fit and Diffusions with Angles.* Stochastics, Vol. 79, No. 3-4, 2007, (293-302). 48

[18] Henry McKean, Lawrence Shepp. *The Advantage of Capitalism vs. Socialism depends on the Criterion.* Journal of Mathematical Sciences, Vol. 139, No.3, pages 6589-6594, 2006. 49, 53, 81

[19] Derek Causon, Clive Mingham. *Introductory Finite Difference Methods for PDEs.* Ventus Publishing ApS, 2010. 33, 38

[20] Elimhan Mahmudov. *Single Variable Differential and Integral Calculus: Mathematical Analysis.* Atlantis Press, 2013. 33

[21] Emilia Petrişor. *Simulare Monte Carlo.* Editura Politehnica, Timişoara, 2006. 9, 38

[22] Paul Glasserman. *Monte Carlo Methods in Financial Engineering.* Springer, New York, 2003. 39

[23] Patrick Billingsley. *Probability and Measure.* John Wiley Sons, New York, 1995. 39

[24] Kai Lai Chung. *A Course in Probability Theory.* Academic Press, New York, 2000. 39

[25] Robert C. Merton. *Optimum Consumption and Portfolio Rules in a Continuous-Time Model.* Journal of Economic Theory, 3:373-413, 1971. 25, 26

[26] Hanspeter Schmidli. *Stochastic Control in Insurance.* Springer Verlag, Köln, 2007. 26, 27

[27] Martina T. Castillo, Gilbert Parrocha. *Stochastic Control Theory for Optimal Investment.* Society of Actuaries ARCH2004.1, 2004.
Download available from: http://library.soa.org/library-pdf/arch04v38n1_17.pdf 28, 30, 31, 32

[28] Martina T. Castillo. *Stochastic Control: Alternative Tool in Insurance Risk Management.* International Actuarial Association, 2005.
Download available from: https://www.business.unsw.edu.au/about/schools/risk-actuarial/research/publications 30, 31, 32

[29] David Aldous. *"Up the River" game story.* 2000.
Download available from: http://www.stat.berkeley.edu/~aldous/Research/OP/river.pdf 41, 42, 47

[30] David Aldous. *Up the river to self-organized criticality.* 2002.
Download available from: http://oz.berkeley.edu/DBconf/packet_3.pdf 41, 47

[31] Peter Mörters, Yuval Peres. *Brownian Motion.* Cambridge University Press, 2010.

[32] George S. Fishman. *Monte Carlo: Concepts, Algorithms and Applications.* Springer Series in Operations Research, 1997.

[33] Daniel Revuz, Marc Yor. *Continuous Martingales and Brownian Motion.* Springer Verlag, 1999. 5, 8