UNIVERSITÄT ZU LÜBECK

University of Lübeck
Director: Prof. Dr. rer. nat. habil. Ralf Möller

On Markov Decision Processes with the Stochastic Differential Bellman Equation

Dissertation for Fulfilment of
Requirements for the Doctoral Degree
of the University of Lübeck

Submitted by

Merve Nur Cakir

from Hamburg
Lübeck 2024

First referee: Prof. Dr. rer. nat. habil. Ralf Möller

Second referee: Prof. Dr. rer. nat. Andreas Rössler

Date of oral examination: December 17, 2024

Approved for printing. Lübeck

Abstract

Stochastic differential equations play an important role in capturing the dynamics of complex systems, where uncertainty prevails in the form of noise. In complex systems noise is abundant, but its exact behaviour is unknown. However, noise can be simulated with stochastic processes. Stochastic calculi, such as the Itô formula, provide tools for navigating these systems. In this work, the adaptation of the Bellman equation, a cornerstone of dynamic programming, to the realm of stochastic differential equations is explored, facilitating the modeling of decision problems subject to noise. Value iteration and Q-learning, two well-known solution methods in machine learning, are extended to stochastic algorithms in order to approximate the solution for Markov decision processes with uncertainties modeled by the stochastic differential Bellman equation.

These stochastic algorithms enable a realistic approach to modeling and solving decision problems in stochastic environments efficiently. The stochastic value iteration is applied when the environment is fully known, while the stochastic Q-learning extends its utility even in cases where transition probabilities remain unknown. Through theoretical analyses and case studies, these algorithms demonstrate their efficacy and applicability, delivering meaningful results. Additionally, the stochastic Q-learning achieves superior rewards compared to the deterministic algorithm, indicating its ability to optimize decision processes in stochastic environments more effectively by exploring more states. Finally, the stochastic differential Bellman equation is formulated as a system of ordinary equations, providing an alternative solution. For this, the concept of the random dynamical system is explored, of which a stochastic differential equation is an example.

Kurzfassung

Stochastische Differentialgleichungen spielen eine wichtige Rolle bei der Erfassung der Dynamik komplexer Systeme, in denen Unsicherheit in Form von Rauschen vorherrscht. In komplexen Systemen ist Rauschen reichlich vorhanden, aber sein genaues Verhalten ist unbekannt. Rauschen kann jedoch mit stochastischen Prozessen simuliert werden. Stochastische Kalküle, wie die Itô-Formel, bieten Werkzeuge zur Navigation in diesen Systemen. In dieser Arbeit wird die Anpassung der Bellman-Gleichung, einem Eckpfeiler der dynamischen Programmierung, an den Bereich stochastischer Differentialgleichungen untersucht, um die Modellierung von Entscheidungsproblemen unter Rauschen zu erleichtern. Werte-Iteration und Q-Lernen, zwei bekannte Lösungsmethoden im maschinellen Lernen, werden zu stochastischen Algorithmen erweitert, um die Lösung von Markov-Entscheidungsprozessen mit Unsicherheiten zu approximieren, die durch die stochastische differenzielle Bellman-Gleichung modelliert werden.

Diese stochastischen Algorithmen ermöglichen einen realistischen Ansatz zur effizienten Modellierung und Lösung von Entscheidungsproblemen in stochastischen Umgebungen. Die stochastische Werte-Iteration wird angewendet, wenn die Umgebung vollständig bekannt ist, während das stochastische Q-Lernen seinen Nutzen auch in Fällen erweitert, in denen Übergangswahrscheinlichkeiten unbekannt bleiben. Durch theoretische Analysen und Fallstudien zeigen diese Algorithmen ihre Wirksamkeit und Anwendbarkeit und liefern aussagekräftige Ergebnisse. Darüber hinaus erreicht das stochastische Q-Lernen im Vergleich zum deterministischen Algorithmus höhere Belohnungen, was darauf hinweist, dass es Entscheidungsprozesse in stochastischen Umgebungen effektiver optimieren kann, indem es mehr Zustände erkundet. Schließlich wird die stochastische differenzielle Bellman-Gleichung als ein System gewöhnlicher Gleichungen formuliert, das eine alternative Lösung bietet. Dafür wird das Konzept des zufälligen dynamischen Systems untersucht, von dem eine stochastische Differentialgleichung ein Beispiel ist.

Acknowledgements

The journey of completing this doctoral thesis has been one of the most challenging yet enlightening experiences of my life. I would like to express my deepest gratitude to everyone who stood by my side with their endless support and encouragement, making this journey possible.

First and foremost, I would like to extend my heartfelt thanks to my supervisor, Prof. Dr. Möller, for stepping in during a critical time in my academic journey and becoming my doctoral advisor. His trust in me, valuable guidance, and direction greatly contributed to my personal and academic growth and helped me advance my research. Through deep discussions and his exceptional mentorship, he provided me with new ideas and constantly encouraged me. His support gave me the strength to persevere even in moments of doubt. I will always remain deeply grateful for his confidence and motivation.

I would also like to express my sincere gratitude to my late supervisor, Prof. Zimmermann, who entrusted me with this research project and guided me. He supported me not only during my PhD but also throughout my entire academic journey. His passion for research and unwavering belief in my abilities laid the foundation for this work and gave me the confidence to overcome obstacles. His sudden passing was a profound loss, but his legacy lives on in my work and will always hold a significant place in my heart.

To my husband, Hakan, I am deeply grateful for your tireless support, understanding, and companionship throughout my research journey. You were the first person I shared my dream of pursuing a doctorate with, and you encouraged me to take the first step toward this goal. Your love and support provided me with the foundation I needed to achieve my aspirations.

To my mother, your unconditional love, unwavering support, and inspiring strength have always lifted me through every challenge. Thanks to the opportunities you provided, I was able to pursue and grow in my academic life. Words cannot fully capture the emotional support you gave me. For standing by me through every hardship, I offer you my endless thanks. Of course, I extend my heartfelt gratitude to my entire family. My dear Uncle Erkan, I will never forget the conversations we had and the support you gave me. Your success in academia has always been an inspiration to me.

A heartfelt thank you also goes to my colleagues, especially Mahwish, I deeply appreciate your valuable advice and friendly support. Your contributions have enriched my work. To my dear friend Züleyha, who is like a sister to me, I sincerely thank you for your encouraging words and motivation. You were always there for me, giving me strength and inspiring me to keep moving forward.

Finally, I would like to thank everyone who contributed to my thesis and made this journey more meaningful.

Merve Nur Çakır

Contents

List of Figures

List of Tables

Notation

$\|x\|$	Absolute value of x
$\pi(s)$	Action taken in state s
$P(A \mid B)$	Conditional probability of A given B
B_t	Brownian motion
$\delta(\cdot)$	Dirac delta function
γ	Discount factor
$\mathbb{E}$	Expectation
$\mathrm{H}(\cdot)$	Hessian matrix
$\mathcal{N}(\mu, \sigma^2)$	Normal distribution with mean μ and variance σ^2
$\pi^*(s)$	Optimal action taken in state s
π^*	Optimal policy
Q^*	Optimal Q-function
V^*	Optimal value function
$V^*(s)$	Optimal value of state s
π	Policy
f_X	Probability density function
P_X	Probability distribution
$\mathbb{P}$	Probability measure
$Q(s, a)$	Q-function, value of taking action a in state s
Ω	Sample space
Δt	Stepsize
F_t	Stochastic integrating factor
$\mathrm{Tr}(A)$	Trace of A
V_t	Value function
$V(s)$	Value function of state s

Chapter 1

Introduction

1.1 Motivation

Artificial intelligence (AI) is able to simulate human behaviour and thinking, thereby simplifying certain tasks in everyday life. Machine learning (ML) methods provide the necessary tools for artificial intelligence. In the 1950s, machine learning research began with early investigations utilizing simple algorithms, enabling automated data analysis and computation tasks [44]. Nowadays, machine learning finds application in numerous real-world scenarios such as weather forecasting, traffic prediction, robotic systems, and search engines [35]. Even with very large amounts of data, so-called big data, machine learning can be used to analyze and evaluate data with reasonable effort. Addressing decision-making problems within this context involves finding the best sequence of decisions for a given environment with respect to a goal.

When faced with decision-making problems, humans evaluate options based on the potential outcomes, benefits, and costs associated with each action. Similarly, machine learning methods should dynamically construct decisions based on experience, feedback, and consideration of uncertainty. This concept led to the development of reinforcement learning in the 1980s [33], where an agent learns to make decisions by receiving feedback on its actions. An agent in artificial intelligence refers to any entity capable of perceiving its environment and taking actions to achieve specific objectives. These decision-making problems are modeled using the mathematical framework of Markov decision process. Feedback from the environment is provided in the form of rewards, indicating the quality of the decisions made by the agent.

In model-based learning, the agent determines the current state with certainty, but the consequence of taking an action is non-deterministic, as selecting an action may lead to various possible states with corresponding probabilities. Value iteration and policy iteration are fundamental algorithms for solving Markov decision processes. The value iteration algorithm calculates the optimal value function, which represents the expected cumulative reward from a given state by choosing the best action to maximize this reward. Then an optimal policy is determined by choosing actions in each state to maximize the optimal value function. Policy iteration starts with an initial

policy, computing its value function (policy evaluation), and updating the policy based on the current value function (policy improvement). Iterating between evaluation and improvement continues until convergence to the optimal policy is achieved and further improvement is no longer possible [43]. In contrast, Q-learning is a model-free learning algorithm that derives an optimal policy without requiring full knowledge of the environment, by exploring and exploiting it. This is accomplished by iteratively updating the Q-function, which represents the expected cumulative reward that an agent can attain from a certain state and action, based on observed rewards.

However, Markov decision processes have several limitations when applied to real-world decision problems. For instance, it is assumed that all uncertainty in the environment can be entirely captured by transition probabilities between states. In reality, these probabilities may be uncertain themselves or subject to estimation errors. In complex systems, noise is abundant, yet Markov decision processes cannot model noise, since the exact value of the noise variable is unknown. The environment may be influenced by external factors that are beyond the control of the agent. Also Q-learning is unable to handle such external factors.

Some proposed solutions in the literature include Hidden-mode Markov decision processes (HM-MDPs), a subclass of partially observable Markov decision processes (POMDPs) [26], worked out by Choi et al. in [7]. In HM-MDPs, the environment is divided into hidden modes reflecting uncertainty about the true state. A mode cannot be observed directly, agents infer these modes from observations and actions. Besides the complexity introduced by inferring hidden modes and potential errors due to observation noise, a key point is that uncertainties in state transitions or rewards are disregarded, which can significantly impact decision-making accuracy. Additionally, the number of modes is assumed to be known, and the exploration-exploitation issue is overlooked. This means, a broader modeling approach is necessary to fully capture the uncertainty of the environment in decision-making problems, due to the stochastic nature of real-world systems.

Consider the scenario where the agent can determine its own state, while the environment changes stochastically due to external influences. In financial mathematics and physics, systems influenced by random factors are represented by stochastic differential equations (SDEs). These equations expand differential equations by including stochastic processes like Brownian motion, which model the uncertainty by random fluctuations. The Fokker-Planck equation (FPE), a partial differential equation, is essential for examining how the probability density functions of stochastic differential equations evolve over time. It provides insights into the long-term behavior and stability of stochastic systems. For example, in financial mathematics, the Fokker-Planck equation models the probability distribution of prices, volatility or losses over time, which allows for a better understanding and assessment of fluctuations and the associated risks [50].

1.2 Objectives

The objective of this work is to develop more realistic decision-making models capable of better capturing uncertainties in real-world scenarios, while minimizing computational complexity. To achieve this goal, the integration of stochastic differential equations and Markov decision processes is explored with the aim of combining the advantages of both methods to address real-world decision-making challenges and advance the field of artificial intelligence. The focus lies in clarifying the modeling of decision problems involving external factors using stochastic differential equations, exploring efficient solution methods, and proposing algorithms for implementing these solutions in `R`, in the most manageable and compact way possible.

At the core of this work lies the Bellman equation, considered as a stochastic differential equation, referred to as the stochastic differential Bellman equation (SBE). The theoretical foundations of machine learning and stochastic dynamics are elaborated, existing approaches for solving stochastic differential equations are examined, and new algorithms and frameworks are proposed to bridge the gap between existing decision-making and stochastic modeling. This new approach enables the consideration of an environment that can change without the agent itself causing the changes, facilitating the modeling of real-world decision problems, offering versatility, and paving the way for more adaptable systems.

Nevertheless, the stochastic differential Bellman equation poses several challenges. Modeling decision problems with the stochastic differential Bellman equation leads to continuous modeling. Generally, solving stochastic differential equations is related to integrating stochastic processes with respect to Brownian motion, known as Itô integrals. The initial method employed in this work is to solve the new approach through stochastic integration. But stochastic integration may not always be feasible, since random variables are involved. Itô's formula is the most important and useful technique in stochastic calculus, especially when a closed-form solution for the stochastic differential equation exists, which refers to representing the solution using elementary functions [48]. However, most stochastic differential equations do not have a closed-form solution, such that even Itô's formula is insufficient. Therefore, the objective involves discretizing the continuous model and developing an algorithm capable of iteratively simulating this discrete representation. To be able to solve the stochastic differential Bellman equation discrete approximations, so-called numerical methods, are considered.

1.3 Contributions

The introduction of the stochastic differential Bellman equation has opened up new possibilities for modeling decision problems in the presence of noise, enabling a more realistic representation of decision-making within complex systems. The new approach has been extended to model-free learning, where the Q-function has been augmented with fluctuation terms to account for uncertainties. Different solution methods have

been explored to interpret and solve the obtained stochastic processes. While the Itô formula was applied, it only enabled finding a solution of the stochastic Bellman equation under a given policy, proving insufficient for the n-dimensional case aiming to find an optimal policy. The Euler-Maruyama method was utilized to achieve a discrete approximation. These simulations were iteratively applied, resulting in algorithms that are able to compute an optimal policy for a Markov decision process with uncertainties.

These algorithms, stochastic value iteration and stochastic Q-learning, have introduced new challenges, particularly regarding the behavior of simulations, which may not converge due to the incorporation of stochastic processes. To address this, modifications were made to the obtained equations, ensuring that results could be achieved within a certain number of iterations. The algorithms implemented in `R` effectively tackle practical decision problems, such as replacement problem and inventory management, considering noise presence. In terms of theoretical analysis the Fokker-Planck equation is applied and the second moment is estimated to describe the impact and long-term behavior of different fluctuation terms. Statistical tests were carried out to determine the influence of hyperparameters, leading to the conclusion that they indeed had an impact on the optimal policy.

The major results of this work are:

- A new equation has been derived, called the stochastic differential Bellman equation, which addresses decision problems in complex systems affected by random fluctuations.
- A closed-form solution for the scalar stochastic differential Bellman equation is calculated under a given policy using the Itô formula. For the general stochastic differential Bellman equation a closed-form solution cannot be given. Hence, the solution is simulated using two approaches: the Euler-Maruyama method and a modified version of it.
- Algorithms for computing an optimal policy are developed after successfully modeling decision problems in stochastic environments. The stochastic value iteration algorithm is based on the value iteration algorithm and is able to handle noise.
- A fluctuation term is added to the Q-function, for which a stochastic update rule is obtained. Iterating the stochastic update rule results in a new stochastic algorithm. In case-studies, it has been demonstrated that stochastic Q-learning achieves higher rewards compared to scenarios without noise.
- Stochastic value iteration and stochastic Q-learning algorithms are implemented in `R` and applied to real-world decision problems like replacement and inventory management, considering the presence of noise. Analyzing the impact of various hyperparameters allows for understanding the sensitivity of results and identifying optimal settings for efficient solutions. Despite the stochastic nature of these problems, both algorithms demonstrate the capability to find robust and stable solutions even under stochastic disturbances.

- Two different fluctuation terms are examined through statistical tests, moments analysis, and the application of the Fokker-Planck equation.
- The stochastic differential Bellman equation is transformed into a system of ordinary differential equations with random coefficients. This method opens a new door for an alternative solution of the stochastic differential Bellman equation and achieves good results even with large parameters.

All in all, the results of this work contribute to a better understanding of the complexity and uncertainty in dynamic systems and offer new perspectives for the development of future decision-making models in artificial intelligence.

1.4 Outline

This work is categorized into two main parts. In the first part, Chapter 2, fundamental concepts of machine learning are explored, including the core part of the work - the Bellman equation. Additionally, Markov decision processes are introduced along with their solutions utilizing techniques such as value iteration and the Q-learning algorithm. A review of probability theory is provided to comprehend stochastic dynamics, with an introduction to Brownian motion for modeling noise. Furthermore, basic principles of stochastic differential equations and their simulation are presented.

In Chapter 3, two new approaches, the stochastic differential Bellman equation and stochastic Q-learning, are introduced, and ways of simulating them are discussed. Statistical tests are used to explore the impact of parameters within the stochastic differential Bellman equation. In-depth mathematical methods, such as moments and Fokker-Planck equation, are employed to analyze the behavior of noise. These new approaches are applied to two different real-world decision problems; the replacement problem and inventory management problem, with subsequent discussions on the obtained results. Additionally, the concept of random dynamical systems is described and the conversion of stochastic differential equations to random differential equations is explained, opening new possibilities for alternative solutions. In particular, the conversion of the stochastic differential Bellman equation will be presented. Finally, the chapter concludes with a summary of findings and directions for future research.

Chapter 2

Machine Learning and Stochastic Dynamics

This chapter provides an introduction to fundamental principles in machine learning for decision processes and stochastic analysis. Markov decision processes are defined for modeling decision problems where outcomes are influenced by probabilistic transitions. The value iteration algorithm, based on the Bellman equation, as well as the Q-learning algorithm, are explored. Expanding the scope, the chapter delves into stochastic dynamics, commencing with the introduction of Brownian motion. Brownian motion is a stochastic process characterized by random fluctuations. The exploration proceeds to stochastic differential equations, which offer a formal framework for modeling dynamic systems subject to stochastic influences. Additionally, numerical methods for approximating solutions to stochastic differential equations are presented. Before delving into these topics, a brief overview of probability theory is provided to establish the necessary foundation for understanding decision-making in stochastic environments.

2.1 Probability Theory

An experiment with several possible outcomes, which cannot be predicted with certainty, is called a *random experiment* [32]. The set of all possible outcomes is referred to as the *sample space*, denoted by Ω. Each possible outcome is considered an *event* [17, 54]. Formally, a random experiment can be defined by a *probability space* $(\Omega, \mathcal{F}, P)$, where Ω is the sample space, $\mathcal{F}$ is a *σ-field* of events which are subsets of Ω with certain set-theoretic properties, and $P : \mathcal{F} \to [0, 1]$ is the probability function. An event E occurs *almost surely* (a.s.) if $P(E) = 1$. A σ-field in $\mathbb{R}$ is the *Borel σ-field* $\mathcal{B}(\mathbb{R})$, generated by all open sets of $\mathbb{R}$ constructed by countable unions. A set in the Borel σ-field is called *Borel set*. A *random variable* on a probability space is a function $X : \Omega \to \mathbb{R}$ that maps each outcome in the sample space to a real number. For each Borel set $A \in \mathcal{B}(\mathbb{R})$, the preimage of A is the set of all outcomes $\omega \in \Omega$ such that $X(\omega) \in A$, and it belongs to the σ-field $\mathcal{F}$:

$$X^{-1}(A) = \{\omega \in \Omega \mid X(\omega) \in A\}. \tag{2.1.1}$$

A random variable is called *discrete* if the sample space is finite or countable, and *continuous* if it can take on values from one or more intervals. This introduction focuses on the continuous case [22, 54]. In the definitions of functions for discrete random variables, the integral is replaced by a sum.

Definition 2.1.1 (Cumulative distribution function)**.** *The* cumulative distribution function *(cdf) of a random variable X is the probability that X takes on values less than or equal to x and is defined as*

$$F_X(x) = P(X \leq x), \quad x \in \mathbb{R}. \tag{2.1.2}$$

A cdf is monotonically increasing, that is, $F(y) \geq F(x)$ for all $x, y \in \mathbb{R}$ with $y \geq x$. The probabilities of occurrence of possible outcomes in a random experiment can be considered by the following functions.

Definition 2.1.2 (Probability density function)**.** *A continuous random variable X is defined by a* probability density function *(pdf) $f_X(x) = F'_X(x)$, where $F'_X(x)$ is the derivative of a cumulative distribution function*

$$F_X(x) = P(X \leq x) = \int_{-\infty}^{x} f_X(x)dx, \quad x \in \mathbb{R}. \tag{2.1.3}$$

The probability density function is the derivative of the monotonically increasing cdf, and so $f_X(x) \geq 0$. Furthermore, $F_X(\infty) = \int_{-\infty}^{\infty} f_X(x)dx = 1$. The expectation of a random variable depends on its distribution.

Definition 2.1.3 (Expectation)**.** *The* expectation *of a continuous random variable X with density function f_X is defined as*

$$\mathbb{E}_{f_X}[X] = \int_{-\infty}^{\infty} x f_X(x)dx, \tag{2.1.4}$$

whenever this integral converges absolutely.

The expectation defines the first moment and is also called *mean* [9]. In the following it is assumed that the expectation of a continuous random variable is absolutely convergent, and for simplicity, we resort to the conventional notation $\mathbb{E}[X]$. However, it is important to note that the expectation still depends on the probability density function. The expectation of a function $g(X)$ of a continuous random variable X is

$$\mathbb{E}[g(X)] = \int_{-\infty}^{\infty} g(x) f_X(x)dx. \tag{2.1.5}$$

A useful property of expectation is linearity.

Proposition 2.1.1. *Let X, Y be random variables.*

- $\mathbb{E}[cX] = c\mathbb{E}[X]$ *for each constant* c,
- $\mathbb{E}[X \pm Y] = \mathbb{E}[X] \pm \mathbb{E}[Y]$.

For each $k \geq 1$ the *k-th moment* of a random variable X is defined as $\mathbb{E}[X^k]$. The second central moment is called the *variance* of a random variable.

Definition 2.1.4 (Variance)**.** *The variance of a continuous random variable X is defined as*

$$\mathrm{Var}(X) = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right] = \int_{-\infty}^{\infty} (x - \mathbb{E}[X])^2 f_X(x) dx. \tag{2.1.6}$$

The variance of a random variable X can be expanded by Prop. 2.1.1 as follows:

$$\mathrm{Var}(X) = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right] = \mathbb{E}\left[X^2\right] - \mathbb{E}[X]^2. \tag{2.1.7}$$

The square root of the variance is called the *standard deviation*.

Example 2.1.1 (Normal distribution - Gaussian)**.** A *normally distributed* random variable X has the probability density function

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}}, \quad x \in \mathbb{R}, \tag{2.1.8}$$

where μ is the mean, σ^2 is the variance and $\sigma \neq 0$ is the standard deviation of the random variable X. The normal distribution is denoted by $\mathcal{N}(\mu, \sigma^2)$ and the notation $X \sim \mathcal{N}(\mu, \sigma^2)$ is used for the normally distributed random variable.

The distributions for common mean $\mu = 2$ and different standard deviation $\sigma \in \{1, 0.5\}$ are illustrated in Fig. 2.1. For smaller standard deviations, the bell curve is narrower, and the distribution is tighter around the mean, while for larger standard deviations, the distribution spreads out more, with data points farther from the mean.

```
> x <- seq(-2, 6, 0.1)
# Generate pdf of normally distributed random variable
# Mean 2, standard deviation 1
> f <- dnorm(x, mean = 2, sd = 1)
> plot(x, f, type = "l", ylim = c(0, 1), col = "red", lwd = 2)
# Mean 2, standard deviation 0.5
> lines(x, dnorm(x, mean = 2, sd = 0.5), col = "blue", lwd = 2)
```

The *standard normal distribution* is the Gaussian distribution with $\mu = 0$ and $\sigma^2 = 1$, which is denoted by $\mathcal{N}(0, 1)$. In this case, the cdf is given as

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \mathrm{e}^{\frac{-t^2}{2}} dt, \quad x \in \mathbb{R}. \tag{2.1.9}$$

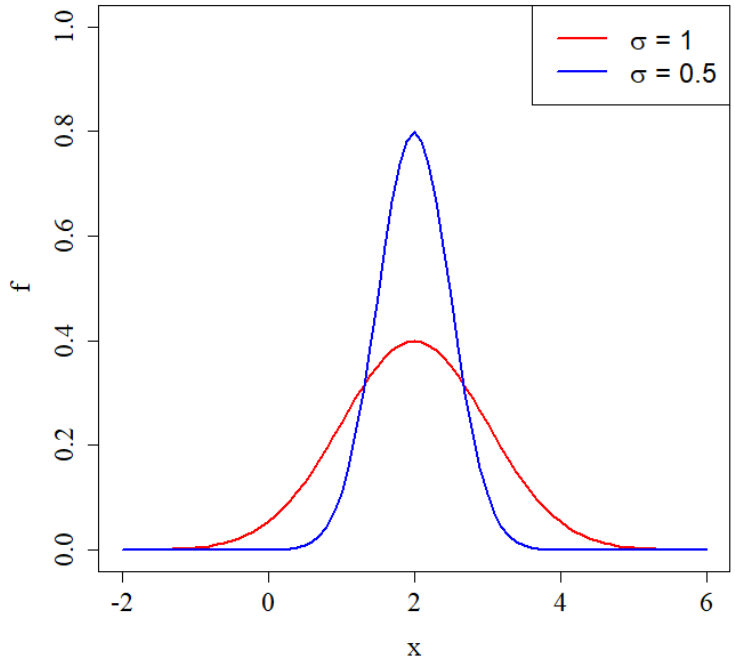


Figure 2.1: Normal distributions for different standard deviations.

For a normal distribution with mean μ, standard deviation σ, and probability density function f, the cdf is

$$F(x) = \Phi\Big(\frac{x-\mu}{\sigma}\Big), \quad x \in \mathbb{R}. \tag{2.1.10}$$

◇

Definition 2.1.5 (Joint probability function)**.** *The* joint probability function *of continuous random variables X and Y is denoted by $f_{X,Y}$. The cumulative distribution function is given by*

$$F_{X,Y}(x,y) = P\left(X \leq x, Y \leq y\right) = \int_{-\infty}^{y}\int_{-\infty}^{x} f_{X,Y}(s,t)dsdt, \quad x,y \in \mathbb{R}. \tag{2.1.11}$$

The probability distribution of a random variable in a pair of random variables occurring independent of other random variable is called the *marginal distribution* and is obtained by integrating the joint probability function for the other variable.

Definition 2.1.6 (Marginal probability distribution)**.** *The marginal distributions of two continuous random variables X and Y with joint probability function $f_{X,Y}(x,y)$ are*

$$f_X(x) = \int_{-\infty}^{\infty} f_{X,Y}(x,y)dy \quad \textit{and} \quad f_Y(y) = \int_{-\infty}^{\infty} f_{X,Y}(x,y)dx, \quad x,y \in \mathbb{R}.$$

Similar concepts apply to more than two random variables. Two continuous random variables X, Y are *independent* if

$$f_{X,Y}(x,y) = f_X(x) f_Y(y) \quad \text{for all } x, y \in \mathbb{R}, \tag{2.1.12}$$

where $f_X(x)$ and $f_Y(y)$ are their marginal probability distributions, or equivalently

$$F_{X,Y}(x,y) = F_X(x) F_Y(y) \quad \text{for all } x, y \in \mathbb{R}. \tag{2.1.13}$$

An n-dimensional random variable $X = (X_1, \ldots, X_n)^{\mathrm{T}}$ is called *random vector*. Two continuous random vectors $X = (X_1, \ldots, X_m)^{\mathrm{T}}$ and $Y = (Y_1, \ldots, Y_n)^{\mathrm{T}}$ are *independent* if

$$F_{X,Y}(x_1, \ldots, x_m, y_1, \ldots, y_n) = F_X(x_1, \ldots, x_m) \cdot F_Y(y_1, \ldots, y_n) \tag{2.1.14}$$

for all $x_1, \ldots, x_m, y_1, \ldots, y_n \in \mathbb{R}$. The *expectation* of an n-dimensional random vector $X = (X_1, \ldots, X_n)^{\mathrm{T}}$ is defined as

$$\mathbb{E}[X] = (\mathbb{E}[X_1], \ldots, \mathbb{E}[X_n])^{\mathrm{T}}. \tag{2.1.15}$$

Proposition 2.1.2. *Let X and Y be independent random variables. Then*

$$\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]. \tag{2.1.16}$$

The covariance is used to analyze how random variables are statistically related.

Definition 2.1.7 (Covariance)**.** *For a pair of random variables X and Y the* covariance *is defined as*

$$\mathrm{Cov}[X,Y] = \mathbb{E}\left[(X - \mathbb{E}[X])\,(Y - \mathbb{E}[Y])\right]. \tag{2.1.17}$$

The covariance can be simplified due to Prop. 2.1.1 as

$$\mathrm{Cov}[X,Y] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]. \tag{2.1.18}$$

If the random variables are *uncorrelated*, i.e. independent of each other, the covariance is zero.

The conditional functions of random variables are similar to the conditional probabilities of events. Let X and Y be random variables with joint probability function $f_{X,Y}$. Then the *conditional density* of X given $Y = y$, $y \in \mathbb{R}$, is defined as

$$f_{X|Y=y}(x) = \frac{f_{X,Y}(x,y)}{f_Y(y)}, \quad x \in \mathbb{R}, \tag{2.1.19}$$

and the *conditional expectation* is

$$\mathbb{E}[X \mid Y = y] = \int_{-\infty}^{\infty} x f_{X|Y=y}(x) dx. \tag{2.1.20}$$

The *conditional variance* of X given $Y = y$, $y \in \mathbb{R}$, is

$$\mathrm{Var}(X \mid Y = y) = \mathbb{E}\left[(X - \mathbb{E}[X \mid Y = y])^2 \mid Y\right] = \mathbb{E}[X^2 \mid Y] - (\mathbb{E}[X \mid Y])^2. \tag{2.1.21}$$

Proposition 2.1.3. *The law of total expectation states that*

$$\mathbb{E}[X] = \mathbb{E}\left(\mathbb{E}[X \mid Y]\right), \tag{2.1.22}$$

where X and Y are random variables and $\mathbb{E}[X \mid Y]$ is the conditional expectation.

Example 2.1.2 (Multivariate Gaussian distribution)**.** Let $X = (X_1, \ldots, X_n)^{\mathrm{T}}$ be a random vector with expectation $\mu = (\mu_1, \ldots, \mu_n)^{\mathrm{T}} \in \mathbb{R}^n$. The multivariate Gaussian distribution, often referred to as the multivariate normal distribution, for the random vector X is characterized by the probability density function

$$f_X(x) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \, \mathrm{e}^{-\frac{1}{2}(x-\mu)^{\mathrm{T}} \Sigma^{-1} (x-\mu)}, \tag{2.1.23}$$

where $|\Sigma|$ denotes the determinant of the covariance matrix $\Sigma \in \mathbb{R}^{n \times n}$, which is symmetric positive definite with diagonal elements equal to the variance σ_i^2 of X_i. If the random variables are uncorrelated, the covariance matrix yields

$$\Sigma = \begin{bmatrix} \sigma_1^2 & 0 & \ldots & 0 \\ 0 & \sigma^2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \sigma_n^2 \end{bmatrix} = \sigma^2 I_n,$$

where I_n is the $n \times n$ identity matrix.

In the two-dimensional case, let X and Y be normally distributed with expectations μ_X, μ_Y and variances σ_X^2, σ_Y^2, respectively. The correlation coefficient is defined as $\rho = \mathrm{Cov}[X, Y]/(\sigma_X \sigma_Y)$, with $|\rho| < 1$, indicating the degree of linear relationship between variables. Then the inverse of the covariance matrix is given as

$$\Sigma^{-1} = \frac{1}{1-\rho^2} \begin{bmatrix} \frac{1}{\sigma_X^2} & \frac{-\rho}{\sigma_X \sigma_Y} \\ \frac{-\rho}{\sigma_X \sigma_Y} & \frac{1}{\sigma_Y^2} \end{bmatrix}. \tag{2.1.24}$$

The joint probability density function yields

$$f_{X,Y}(x,y) = \frac{1}{2\pi \sigma_X \sigma_Y \sqrt{1-\rho^2}} \, \mathrm{e}^{-\frac{1}{2(1-\rho^2)} \left(\frac{(x-\mu_x)^2}{\sigma_X^2} - 2\rho \frac{(x-\mu_X)(y-\mu_Y)}{\sigma_X \sigma_Y} + \frac{(y-\mu_Y)^2}{\sigma_Y^2} \right)}. \tag{2.1.25}$$

The marginal probability distributions are the univariate Gaussian distributions. The conditional density of X given $Y = y$, $y \in \mathbb{R}$, is defined as

$$f_{X|Y=y}(x) = \frac{1}{\sqrt{2\pi(1-\rho^2)}\sigma_X} \, \mathrm{e}^{-\frac{\left(x - \mu_{X|Y=y}\right)^2}{2\sigma_{X|Y=y}^2}}, \tag{2.1.26}$$

where

$$\mu_{X|Y=y} = \mu_X + \frac{\sigma_X \rho (y - \mu_y)}{\sigma_Y} \quad \text{and} \quad \sigma_{X|Y=y}^2 = \sigma_X^2 (1-\rho^2).$$

Thus the conditional density is also an univariate Gaussian and will be denoted as $f_{X|Y=y}(x) \sim \mathcal{N}(\mu_{X|Y=y}, \sigma_{X|Y=y}^2)$.

$\diamond$

Finally, stochastic processes are introduced.

Definition 2.1.8 (Stochastic process)**.** *Let* $(\Omega, \mathcal{F}, \mathbb{P})$ *be a probability space. A* stochastic process *is a collection* $(X_t)_{t\in T}$ *of random variables* X_t *indexed by time* $t \in T$*, where* T *is an arbitrary index set.*

Example 2.1.3 (Gaussian process)**.** A stochastic process is a Gaussian process in which the random variables are normally distributed. Formally, a stochastic process $(X_t)_{t\in T}$ is Gaussian if

$$X_{t_1,\dots,t_k} = (X_{t_1}, \dots X_{t_k}) \quad \text{for all } t_1, \dots, t_k \in T, \tag{2.1.27}$$

follows a multivariate Gaussian distribution [4]. The variance of a Gaussian process is finite at any time t.

◇

A stochastic process is called *stationary* if its statistical properties and parameters, such as expectation and variance, do not change over time. In Fig. 2.2, a stationary process is represented by a standard Gaussian process and a non-stationary process by an exponentially growing trend, where the expectation increases exponentially over time, while the variance is not explicitly defined and remains constant in the given context. A stochastic process $(X_t)_{t\in T}$ has *stationary increments* if the probability density function of $X_t - X_s$ is the same for all $s, t \in T$ with $s < t$. The process has *independent increments* if, for all $0 \leq t_1 < \dots < t_n$, the increments $X_{t_2} - X_{t_1}, \dots, X_{t_n} - X_{t_{n-1}}$ are independent random variables.

Example 2.1.4.

1. A stationary process has stationary increments. Thus, a Gaussian process with constant mean and variance has stationary increments.

2. Consider the case of a Gaussian process $(X_t)_{t\in T}$ with zero mean, variance $\sigma^2 \in \mathbb{R}$, and uncorrelated random variables X_i, i.e. $\rho = 0$. Due to zero mean and independent random variables, the covariance of the increments yields $\mathrm{Cov}(X_t - X_{t-1}, X_{t-1}) = -\sigma^2$ and thus the process has dependent increments. This special case of Gaussian process is called *white noise*.

3. An example of a non-stationary continuous stochastic process with independent stationary increments is Brownian motion introduced in Sect. 2.3.1.

◇

2.2 Markov Decision Process and Reinforcement Learning

Decision-making across various environments is a fundamental challenge addressed by machine learning. This section explores the modeling of decision problems as Markov decision processes and investigates solution methods. Familiarity with discrete stochastics is assumed. Initially, fully observable environments and the value iteration algorithm to solve these problems are introduced [39,54]. The Bellman equation is based on

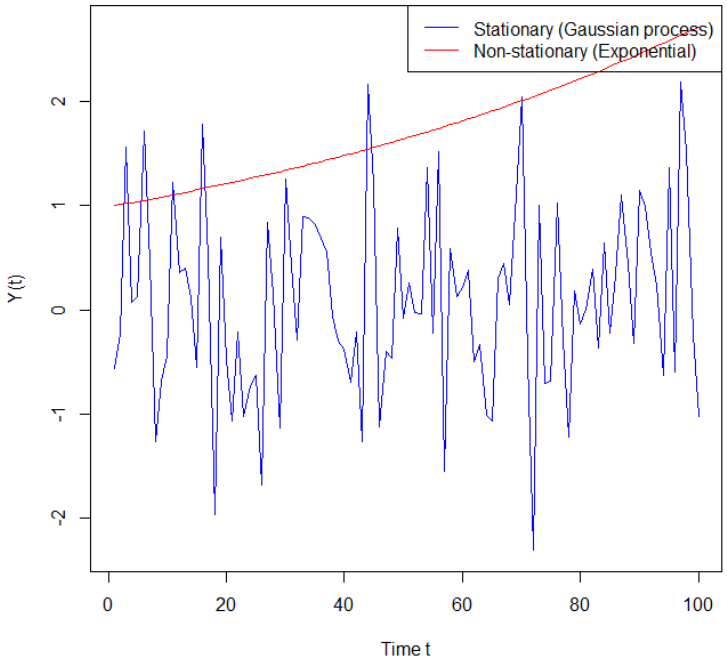


Figure 2.2: Examples of stationary and non-stationary processes.

dynamic programming, and describes the optimal structure of a problem by establishing the recursive relationship between the value of a state and the values of its successor states. The value iteration algorithm, in turn, is an iterative method for solving Markov decision processes that relies on the Bellman equation. Finally, the Q-learning algorithm is introduced. It is a model-free reinforcement learning method used to solve Markov decision processes without full knowledge of the environment.

2.2.1 Markov Decision Process

Consider a stochastic model in which a decision maker, called *agent*, observes the states of the system and chooses an action from a set of available alternatives based on the feedback from the system. The agent receives an immediate reward for performing that action in the current state. The goal is to find a sequence of actions that maximizes the overall reward. But the agent has to take future consequences into account, since the action taken at present affects the future steps of the model.

A *Markov decision process* assumes that the environment is fully observable. The environment is described by a non-empty set of states $S = \{s_1, \ldots, s_n\}$ and a non-empty set of available actions denoted by $A = \{a_1, a_2, \ldots, a_m\}$. A Markov decision process satisfies the *Markov property*, which states that the future is independent of the past given the present. Due to the Markov property it is possible to predict the next state

and reward given the current state and action. The simplest Markov decision process is a Markov chain.

Definition 2.2.1 (Markov chain)**.** *A* Markov chain *is a sequence of random variables* $(X_t)_{t\in\mathbb{N}}$ *with the Markov property, that is, for each* $t \geq 1$

$$P(X_t = s_t \mid X_1 = s_1, X_2 = s_2, \ldots, X_{t-1} = s_{t-1}) = P(X_t = s_t \mid X_{t-1} = s_{t-1}), \quad (2.2.1)$$

where $X_t = s_t$ *says that* X_t *is in state* s_t *at time* t*.*

Stochastic processes with the Markov property are termed *Markov processes*. The probability of transitioning from state s at time t to state s' at time $t+1$ over the common state set S is defined by the conditional probability $P_t(s' \mid s)$. A *transition probability matrix* $P_t : S \times S \to [0,1]$, $t \geq 1$, contains the probabilities of transitioning between states which has elements

$$(P_t)_{ij} = P(X_{t+1} = s_j \mid X_t = s_i), \quad 1 \leq i, j \leq n. \quad (2.2.2)$$

Each row sums up to 1. In the case where transition probabilities are independent of time t, the Markov chain is called *homogeneous* and the transition probability matrix is denoted by P.

Definition 2.2.2 (Probability distribution)**.** *A* probability distribution *of the state set* S *is a vector* $v = (v_1, \ldots, v_n)$ *where all entries are nonnegative and* $\sum_i v_i = 1$*. The value* v_i *is interpreted as the probability of being in state* s_i*,* $1 \leq i \leq n$*.*

Proposition 2.2.1. *Let* P *be the transition probability matrix of a homogeneous Markov chain and* v_0 *be the initial probability distribution. Then the probability distribution after* $n \geq 0$ *steps is given by*

$$v_n = v_0 P^n. \quad (2.2.3)$$

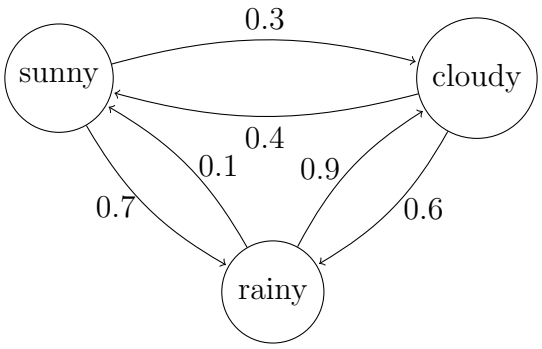


Figure 2.3: Markov chain for weather prediction.

Example 2.2.1 (Markov chain)**.** A Markov chain is employed for weather forecasting, where the weather can be categorized into three possible states: s_1 = "sunny", s_2 = "cloudy", and s_3 = "rainy". Based on historical data (experience), it is known that if the weather is sunny today, there is a probability of 0.7 that it will rain tomorrow and a 0.3 chance of it being cloudy. If it is currently cloudy, there is a 0.6 probability of rain tomorrow and a 0.4 chance of sunshine. Finally, if it rains today, there is a 0.9 probability it will be cloudy tomorrow and a 0.1 chance of sunshine. This example is illustrated in Fig. 2.3. This example is illustrated in Fig. 2.3. In `R`, the Markov chain for the weather example can be defined using the `markovchain` package as follows:

```
> library(markovchain)
> mc_weather <- new("markovchain",
  states = c("sunny", "cloudy", "rainy"),
  transitionMatrix = matrix(data = c(0,0.3,0.7,0.4,0,0.6,0.1,0.9,0),
  byrow = TRUE, nrow = 3), name = "Weather")
```

The function `new` generates a Markov chain in which the states and the probability matrix need to be specified. The weather in the next few days is to be forecast for the case that today is a sunny day, i.e. $v_0 = (1, 0, 0)$. Then

```
> v0 <- c(1,0,0)
> v2 <- v0 * (mc_weather^2)
> v2
     sunny cloudy rainy
[1,]  0.19   0.63  0.18
```

Thus if the sun shines today, there is a 0.63 probability that it will be cloudy the day after tomorrow, a 0.19 probability of sunshine, and a 0.18 probability of rain. ◇

A Markov decision process enables an agent to navigate its environment by selecting actions and learning from the outcomes. When faced with a decision, the agent chooses an action from a predefined set of possibilities, denoted as A. For each action $a \in A$, there exist probability matrices P_a that contains the transition probabilities of reaching the next state after performing this action. In an Markov decision process, there is a reward function R over the states in S, showing the immediate reward when the agent picks an action in a state. Rewards can be positive for gains or negative for losses. The reward function R can either depend solely on the current state s and the action a taken in that state, expressed as $R(s, a)$, or it can also take into account the next state s' resulting from the action, written as $R(s, s', a)$. Furthermore, a Markov chain is a Markov decision process with a single action at each state and no rewards.

Definition 2.2.3 (Markov decision process)**.** *A Markov decision process (MDP) is a quadruple* $M = (S, A, P, R)$*, where*

- S *is a finite set of states,*
- A *is a finite set of actions,*
- P *is a collection of conditional probabilities* $P_a(s'|s) = P(s' \mid s, a)$ *such that the action* $a \in A$ *in state* s *will lead to state* s'*, and*
- $R : S \times S \times A \to \mathbb{R}$ *is the* reward function *such that the* expected immediate reward $R_a(s, s') = R(s, s', a)$ *is given by the transition from state* s *to state* s' *due to the action* a*.*

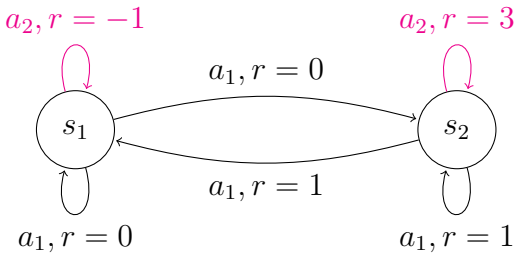


Figure 2.4: A Markov decision process with two states.

Example 2.2.2 (Markov Decision Process)**.** Consider a Markov decision process with two states, s_1 and s_2, and two available actions, a_1 and a_2. Taking action a_1 leads to the other state with a probability of 0.7 and remains in the same state with a probability of 0.3. On the other hand, action a_2 always keeps in the same state. The corresponding probability matrices are:

$$P_{a_1} = \begin{bmatrix} 0.3 & 0.7 \\ 0.7 & 0.3 \end{bmatrix} \text{ and } P_{a_2} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

This decision process, along with appropriate rewards, is illustrated in Fig. 2.4. Which action should be carried out in the states in order to obtain the best possible reward? ◇

Markov decision processes are discrete-time problems that divide time is into stages. If the set of stages is finite, $T = (t_1, \ldots, t_N)$, it is referred to as a *finite horizon problem*, otherwise as an *infinite horizon problem*. A *policy* is a sequence of decision rules indicating which action to take in each state. Formally, a policy is denoted as $\pi = (\pi(s_1), \ldots, \pi(s_n)) \in A^n$, where $S = \{s_1, \ldots, s_n\}$ is the given state set. Solving a Markov decision process means finding a policy that maximizes a sequence of rewards. Specifically, the aim is to maximize the sum of rewards, given by

$$\sum_{t \in T} R_{a_t}(s_t, s_{t+1}), \tag{2.2.4}$$

where $a_t = \pi(s_t)$ for each $t \in T$. In cases where the set T is infinite, this sum may be divergent. Therefore, a *discount factor* γ is added to ensure convergence. The objective then becomes to find a policy that maximizes the discounted sum over a potentially infinite horizon, given by

$$\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}), \tag{2.2.5}$$

where $a_t = \pi(s_t)$ for each $t \in T$, and γ is the discount factor, satisfying $0 \leq \gamma < 1$. This discounted sum encourages the agent to explore the environment instead of simply selecting the action with the highest immediate reward each time.

2.2.2 Bellman Optimality Equation

The *principle of optimality* was developed by Richard Bellman in 1957 and is the basis of dynamic programming. More precisely, it states that for a sequence of optimal decisions, the remaining decisions must also be optimal, regardless of the initial state and decision [3]. This principle enables the computation of optimal solutions to subproblems, which can then be utilized to find an optimal solution for the general problem. In the following, the value function $V_\pi(s)$ is introduced to formalize the equation derived from the principle of optimality. A *value function* is a function of the state set $V_\pi : S \to \mathbb{R}$ and describes the expected discounted reward starting at a given state s and then following a specific policy π. By considering the discounted sum (2.2.5) with $a_t = \pi(s_t)$ for a state s_t at time $t \in T$, the value function $V_\pi(s)$ is defined as

$$V_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) \;\middle|\; s_0 = s\right]. \tag{2.2.6}$$

This formulation captures the expected cumulative reward, taking into account the variability in state transitions and the discount factor γ which balances immediate and future rewards.

The value function can be calculated recursively using the linearity of expectation and the law of total expectation,

$$\begin{aligned}
V_\pi(s) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) \;\middle|\; s_0 = s\right] \\
&= \mathbb{E}\left[R_{\pi(s)}(s, s') + \gamma \sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) \;\middle|\; s_0 = s'\right] \\
&= \sum_{s' \in S} P(s' \mid s, \pi(s)) \left(R_{\pi(s)}(s, s') + \gamma \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) \;\middle|\; s_0 = s'\right]\right) \\
&= \sum_{s' \in S} P_{\pi(s)}(s' \mid s) \left[R_{\pi(s)}(s, s') + \gamma V_\pi(s')\right] \\
&= R(s, \pi(s)) + \gamma \sum_{s' \in S} P_{\pi(s)}(s' \mid s) V_\pi(s').
\end{aligned}$$

In the last equation, $R(s, \pi(s))$ is the expected reward in state s with action $\pi(s)$, i.e. $R(s, \pi(s)) = \sum_{s' \in S} P_{\pi(s)}(s' \mid s) R_{\pi(s)}(s, s')$.

Example 2.2.3 (Value function)**.** Reconsider Ex. 2.2.2 with policy $\pi = (a_2, a_1)$. Take the discount factor $\gamma = 0.6$ and the initial value function $V = (1, 1)^{\mathrm{T}}$. The values of states s_1 and s_2 are computed as

$$V_\pi(s_1) = R_{\pi(s_1)}(s_1) + 0.6 \cdot \sum_{s' \in S} P_{\pi(s_1)}(s' \mid s_1) V(s') = -1 + 0.6 \cdot (1 \cdot 1 + 0 \cdot 1) = -0.4,$$
$$V_\pi(s_2) = R_{\pi(s_2)}(s_2) + 0.6 \cdot \sum_{s' \in S} P_{\pi(s_2)}(s' \mid s_2) V(s') = 1 + 0.6 \cdot (0.7 \cdot 1 + 0.3 \cdot 1) = 1.6.$$

On the other hand, take the policy $\pi = (a_1, a_2)$. Then the value function is

$$V_\pi(s_1) = 0 + 0.6 \cdot (0.3 \cdot 1 + 0.7 \cdot 1) = 0.6,$$
$$V_\pi(s_2) = 3 + 0.6 \cdot (0 \cdot 1 + 1 \cdot 1) = 3.6.$$

So the second policy yields a greater value for each state. ◇

A policy π is *superior* to another policy π' if $V_\pi(s) \geq V_{\pi'}(s)$ for each state $s \in S$.

Definition 2.2.4 (Optimal policy)**.** *Given a Markov decision process, a policy π^* is called optimal if π^* is superior to all other policies.*

The objective of a Markov decision process is to identify an optimal policy.

Definition 2.2.5 (Bellman optimality equation)**.** *The* Bellman optimality equation *is defined by*

$$V^*(s) = \max_{a \in A} \Big\{ R(s, a) + \gamma \sum_{s' \in S} P(s' \mid s, a) \cdot V^*(s') \Big\} \tag{2.2.7}$$

and the optimal policy by

$$\pi^*(s) = \arg\max_{a \in A} \Big\{ R(s, a) + \gamma \sum_{s' \in S} P(s' \mid s, a) \cdot V^*(s') \Big\}. \tag{2.2.8}$$

The Bellman optimality equation provides a fundamental framework for identifying the optimal policy by maximizing the expected sum of rewards over time. In the following, the matrix notation for the Bellman optimality equation will be introduced. Let R_a be the $n \times 1$ column vector of rewards for action $a \in A$, P_a be the $n \times n$ probability matrix of transitions with fixed action a, and V be the $n \times 1$ column vector of state values. Then the equation can be expressed as

$$V_{t+1} = \max_{a \in A} \big\{ R_a + \gamma P_a V_t \big\}, \quad t \geq 0, \ V_0 \in \mathbb{R}^n. \tag{2.2.9}$$

Definition 2.2.6 (Bellman operator)**.** *The* L-operator *defined by*

$$LV_t = \max_{a \in A} \{ R_a + \gamma P_a V_t \} \tag{2.2.10}$$

is called the Bellman operator.

Example 2.2.4 (Bellman operator)**.** In view of Ex. 2.2.3, the Bellman equation can be calculated by applying the function `mdp_bellman_operator`. Let the discount factor be $\gamma = 0.6$, initial value function $V_0 = (1, 1)^{\mathrm{T}}$ and the actions be $a_1 = 1$ and $a_2 = 2$.

```
> library(MDPtoolbox)
> P <- array(0, c(2,2,2))
# Define probability matrices
> P[,,1] <- matrix(c(0.3, 0.7, 0.7, 0.3),byrow=TRUE,nrow=2)
> P[,,2] <- matrix(c(1, 0, 0, 1),byrow=TRUE,nrow=2)
# Define reward matrix
> R1 <- matrix(c(0, 1),byrow=TRUE,nrow=2)
> R2 <- matrix(c(-1, 3),byrow=TRUE,nrow=2)
> R <- cbind(R1, R2)
# Apply Bellman operator
> mdp_bellman_operator(P, R, 0.6, c(1,1))
$V
[1] 0.6 3.6
$policy
[1] 1 2
```

Thus the optimal policy is (a_1, a_2). ◇

The Bellman optimality equation (2.2.5) is a nonlinear equation due to the maximization operation. This equation needs to be solved for an optimal value function V^*. The uniqueness of the optimal value function V^* is considered and its approximation is studied. Recall that a *Banach space* is a complete vector space with a norm [8]. A vector space is considered *complete* if each Cauchy sequence of vectors in this space converges to a limit.

Definition 2.2.7 (Contraction mapping)**.** *Let* X *be a Banach space with norm* $\|\cdot\|$*. An operator* $T : X \to X$ *is called a* contraction mapping *if there exist a number* $\lambda \in [0, 1)$ *such that*

$$\|Ty - Tx\| \leq \lambda \|y - x\| \quad \text{for all } x, y \in X. \tag{2.2.11}$$

Theorem 2.2.1 (Banach Fixed-Point Theorem)**.** *Let* X *be a Banach space with a contraction mapping* $T : X \to X$*. There exists a unique fixed point* $x^* \in X$ *such that* $Tx^* = x^*$*. Moreover, the sequence* $(x_n)_{n \in \mathbb{N}_0}$ *defined by*

$$x_n = Tx_{n-1} = T^n x_0 \quad x_0 \in X, n \geq 1, \tag{2.2.12}$$

converges to x^**.*

This theorem provides a powerful tool for analyzing the convergence properties of iterative algorithms employed in solving optimization problems. In Denardo's observations (1967) [10], it was noted that the operator T defined in Eq. 2.2.7 acts as a contraction mapping on the value function V, which is a key element in the convergence analysis of the value iteration algorithm. Then the Banach fixed point theorem assures the existence of a unique fixed point, corresponding to the optimal value function V^*.

Theorem 2.2.2. *The L-operator is a contraction mapping on V with respect to the max norm $\|V\| = \max_{s\in S} |V(s)|$.*

The proof of Theorem 2.2.2 is based on the following assertion.

Proposition 2.2.2. *For each $t \geq 1$,*

$$\|LV_t - LV_{t-1}\| \leq \gamma \|V_t - V_{t-1}\|, \tag{2.2.13}$$

where $\gamma \in [0, 1)$.

Proof. For each $t \geq 1$,

$$\begin{aligned} \|LV_t - LV_{t-1}\| &= \left\| \max_a \{R_a + \gamma P_a V_t\} - \max_a \{R_a + \gamma P_a V_{t-1}\} \right\| \\ &\leq \max_a \{\|R_a + \gamma P_a V_t - (R_a + \gamma P_a V_{t-1})\|\} \\ &= \max_a \{\|\gamma P_a (V_t - V_{t-1})\|\} \\ &\leq \gamma \|V_t - V_{t-1}\|. \end{aligned} \tag{2.2.14}$$

The first inequality follows from the reverse triangle inequality [45], i.e. for all vectors $x, y \in \mathbb{R}^n$

$$|\max \|x\| - \max \|y\|| \leq \max \|x - y\|. \tag{2.2.15}$$

The last inequality holds since P_a is a probability matrix. □

Prop. 2.2.2 implies that for all $t \geq 1$,

$$\|LV_t - LV_{t-1}\| \leq \gamma \|V_t - V_{t-1}\| \leq \gamma^2 \|V_{t-1} - V_{t-2}\| \ldots \leq \gamma^t \|V_1 - V_0\|. \tag{2.2.16}$$

By the Banach fixed point theorem, the Bellman operator L has a unique fixed point V^* with $LV^* = V^*$ and the sequence $V_n = L^n V_0$ converges to V^* for any V_0. This completes the proof of Theorem 2.2.2. Based on this result, an iterative algorithm can be defined and an upper bound on its error, called Bellman residual, can be specified.

Proposition 2.2.3. *Let $\gamma \in [0, 1)$. If for some $t \geq 0$ and $\epsilon > 0$,*

$$\|V_{t+1} - V_t\| < \frac{\epsilon(1-\gamma)}{2\gamma}, \tag{2.2.17}$$

the Bellman residual *is given by*

$$\|V^* - V_{t+1}\| < \frac{\epsilon}{2}. \tag{2.2.18}$$

Proof. The term $\|V^* - V_{t+1}\|$ satisfies

$$\begin{aligned}\|V^* - V_{t+1}\| &= \|LV^* - LV_{t+1} + LV_{t+1} - LV_t\| \\ &\leq \|LV^* - LV_{t+1}\| + \|LV_{t+1} - LV_t\| \\ &\leq \gamma\|V^* - V_{t+1}\| + \gamma\|V_{t+1} - V_t\|,\end{aligned}$$

where the first inequality follows from the triangle inequality and the second one is due to the contraction property of the L-operator. Further the above inequality yields

$$(1-\gamma)\,\|V^* - V_{t+1}\| \leq \gamma\|V_{t+1} - V_t\| \tag{2.2.19}$$

and so

$$\|V^* - V_{t+1}\| \leq \frac{\gamma}{1-\gamma}\|V_{t+1} - V_t\|. \tag{2.2.20}$$

Hence whenever (2.2.17) holds, the term $\|V^* - V_{t+1}\|$ is bounded by $\frac{\epsilon}{2}$. □

2.2.3 Value Iteration

A Markov decision process can have several distinct optimal policies, which can be found using the value iteration algorithm. The value iteration algorithm requires full knowledge of the environment. It iteratively applies the Bellman optimality equation until the difference between successive value functions is small enough. In other words, the L-operator is applied recursively until the algorithm converges to the optimal value function V^*. The iteration step is denoted by $t \geq 0$ and the algorithm starts at $t = 0$. The initial state V_0 is a random value function, and the optimal policy is independent of the initial state. The steps of the value iteration algorithm are described below (Alg. 1).

1. Set $t = 0$ and $V_t(s) = 0$ for all $s \in S$. Choose $\epsilon > 0$.
2. Compute $V_{t+1}(s) = \max_{a\in A}\left\{R(s,s',a) + \sum_{s'\in S}\gamma P(s' \mid s,a)V_t(s')\right\}$.
3. If $\|V_{t+1} - V_t\| < \epsilon(1-\gamma)/2\gamma$ goto step 4, otherwise set $t = t+1$ goto step 2.
4. Extract policy $\pi(s) = \arg\max_{a\in A}\left\{R(s,a) + \sum_{s'\in S}\gamma P(s' \mid s,a)V_t(s')\right\}$.

Each iteration of the algorithm has time complexity $O(|A| \cdot |S|^2)$, since the for loop has runtime $|S|$ and the body of each for loop has runtime $O(|A| \cdot |S|)$. An implementation of the value iteration algorithm in `R` is given in the appendix.

Example 2.2.5 (Value Iteration)**.** The *forest management problem* is formulated as a Markov decision process, where the states refer to years and the available actions are "wait" and "cut". For three years, the state set is $S = \{s_1, s_2, s_3\}$ and the set of actions is $A = \{a_1, a_2\}$, where $a_1 = 1$ "wait" and $a_2 = 2$ is "cut". An action is carried out annually, where the first objective is to maintain an old forest for wildlife and the second is to earn money by selling cut wood. Moreover, there is a probability $p \in (0,1)$ that a fire burns down the forest. After a fire, the forest is in the youngest state s_1. Let $p = 0.4$, $r_1 = 1$, and $r_2 = 2$, where r_1 is the reward when the forest is in the oldest state and the action "wait" is performed and r_2 is the reward when action "cut" is carried out.

Algorithm 1: Value Iteration Algorithm

Input: MDP (S, A, P, R), $\epsilon > 0$ and $\gamma \in [0, 1)$
Output: Optimal policy π^*

$t = 0$
do
  for $s \in S$ **do**
    $V_{t+1}(s) = \max_{a \in A} \left\{ R(s,a) + \sum_{s' \in S} \gamma P(s' \mid s, a) V_t(s') \right\}$
  end
  $\theta = \|V_{t+1} - V_t\|$
  $t = t + 1$
while $\theta \geq \epsilon \frac{1-\gamma}{2\gamma}$
for $s \in S$ **do**
  $\pi^*(s) = \arg\max_{a \in A} \left\{ R(s,a) + \sum_{s' \in S} \gamma P(s' \mid s, a) V_t(s') \right\}$
end

```
> library(MDPtoolbox)
> MDP_forest <- mdp_example_forest(3,1,2,0.4)
> MDP_forest

$P
, , 1
     [,1] [,2] [,3]
[1,]  0.4  0.6  0.0
[2,]  0.4  0.0  0.6
[3,]  0.4  0.0  0.6
, , 2
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    1    0    0
[3,]    1    0    0

$R
     R1 R2
[1,]  0  0
[2,]  0  1
[3,]  1  2
```

The solution of this Markov decision process is given by the value iteration algorithm. Set $V_0 = (0, 0, 0)^{\mathrm{T}}$ and $\gamma = 0.9$.

```
> ValueIteration(3,P,R,0.9,c(0,0,0),0.01,0)
$Iteration
[1] 63
$V
```

```
[1] 3.501629 4.150980 5.150980
$‘Optimal Policy‘
[1] 1 2 2
```

The algorithm terminates for $\epsilon = 0.01$ after 63 steps with the value function $V^* = (3.50, 4.15, 5.15)^{\mathrm{T}}$ and the optimal policy $\pi^* = (1, 2, 2)$, which means that the sequence of best actions is ("wait", "cut", "cut"). For $\epsilon = 0.1$, the algorithm terminates already after 41 steps with the same optimal policy.

$\diamond$

Different values of the discount factor lead to different value functions and may also result in different optimal policies. Smaller values of γ put more weight on short-term rewards, while larger values prefer long-term gains and losses, but the algorithm requires more iterations to converge. Table 2.1 shows the outcomes of the value iteration algorithm with different discount factors applied to the previous example (Ex. 2.2.5) with $\epsilon = 0.01$.

γ	V^*	π^*	iterations
0.1	(0.062, 1.006, 2.006)	(1,2,2)	3
0.5	(0.45, 1.22, 2.22)	(1,2,2)	8
0.9	(3.50, 4.15, 5.15)	(1,2,2)	63

Table 2.1: Value iteration with different discount factors.

Theorem 2.2.3. *Let $\gamma \in \mathbb{R}$ with $0 \leq \gamma < 1$ and $V_t, R_a \in \mathbb{R}^n$ for all $t > 0$ and $a \in A$. The Bellman equation*

$$V_t = \max_{a \in A} \{R_a + \gamma P_a V_{t-1}\}, \tag{2.2.21}$$

converges for $t \to \infty$.

Proof. Denote the maximum reward by R. In view of convergence, it is sufficient to consider the equation

$$V_t = R + \gamma V_{t-1}, \quad t \geq 1. \tag{2.2.22}$$

By expansion, for each $t \geq 1$,

$$V_t = R + \gamma R + \ldots + \gamma^{t-1} R + \gamma^t V_0 = \sum_{k=0}^{t-1} \gamma^k R + \gamma^t V_0. \tag{2.2.23}$$

For $t \to \infty$, the first summand is a geometric series [47] with a limit point

$$\sum_{k=0}^{\infty} \gamma^k R = \frac{R}{1-\gamma}, \tag{2.2.24}$$

when $0 \leq \gamma < 1$. Hence, the series (V_t) converges for $t \to \infty$. $\square$

The value iteration algorithm converges to the optimal value function V^*. The number of steps required to calculate V^* can be estimated by using the above result.

Theorem 2.2.4. *For $\gamma \in (0,1)$, the value iteration algorithm approaches the optimal value function due to the convergence criterion in Prop. 2.2.3 in the following number of steps*

$$t > \frac{\ln\left(\frac{\epsilon(1-\gamma)}{2R}\right)}{\ln(\gamma)}, \tag{2.2.25}$$

where $\epsilon > 0$ and $R > 0$ is the maximum reward value.

Proof. Assuming $V_0 = 0$ for simplicity, by Theorem 2.2.3, the value functions are defined as

$$V_t = \sum_{k=0}^{t-1} \gamma^k R = \frac{R(\gamma^t - 1)}{\gamma - 1}, t \geq 1, \quad \text{and} \quad V^* = \sum_{k=0}^{\infty} \gamma^k R = \frac{R}{1-\gamma}.$$

The algorithm terminates when the difference of both value functions is small enough. Indeed, the difference is calculated as

$$\|V_t - V^*\| = \left|\frac{R(\gamma^t - 1)}{\gamma - 1} - \frac{R}{1-\gamma}\right| = \left|\frac{R\gamma^t}{\gamma - 1}\right| = \frac{R\gamma^t}{1-\gamma}. \tag{2.2.26}$$

If $\|V_t - V^*\| < \frac{\epsilon}{2}$ for some $\epsilon > 0$, then

$$\frac{R\gamma^t}{1-\gamma} < \frac{\epsilon}{2}, \tag{2.2.27}$$

which leads to

$$t \cdot \ln(\gamma) < \ln\left(\frac{\epsilon(1-\gamma)}{2R}\right). \tag{2.2.28}$$

Since $\ln(\gamma)$ is negative for $\gamma \in (0,1)$, the result follows.

□

2.2.4 Variant of Value Iteration

For the convergence of value iteration algorithm, the max norm was used. The span semi-norm can be chosen to improve the convergence result. The *span* of the value function V is defined as

$$\mathrm{sp}(V) = \max_{s \in S} V(s) - \min_{s \in S} V(s). \tag{2.2.29}$$

This variant of the value iteration algorithm converges if the span of the successive value functions is small enough, i.e. if

$$\mathrm{sp}(V_{t+1} - V_t) < \frac{\epsilon(1-\gamma)}{\gamma}, \tag{2.2.30}$$

where $0 \leq \gamma < 1$.

Example 2.2.6. The function `mdp_value_iteration` in the `MDPtoolbox` package is used to improve the runtime of the value iteration algorithm on the forest management problem.

```
> library(MDPtoolbox)
> mdp_value_iteration(P,R,0.9,0.1,1000,c(0,0,0))
$V
[1] 1.909201 2.555841 3.555841
$policy
[1] 1 2 2
$iter
[1] 8
$time
Time difference of 0.001013994 secs
```

The algorithm terminates already after 8 steps with optimal policy $\pi^* = (1, 2, 2)$.

◇

2.2.5 Q-Learning

Markov decision processes in fully observable environments can be solved with the value iteration algorithm, but a decision problem with unknown rewards or transition probabilities at the beginning cannot be tackled by the value iteration algorithm. A system in which the agent explores the environment is required. This is exactly where reinforcement learning comes into play. Reinforcement learning algorithms of type model-free learning are considered when the transition probabilities or rewards in a Markov decision process are unknown at the start. This section provides an introduction to reinforcement learning and describes the Q-learning algorithm introduced by Watkins [52], which implements model-free learning. Furthermore, this section will provide an illustrative example using the forest management problem, demonstrating how Q-learning can be applied in practical settings to derive optimal policies in dynamic environments.

Example 2.2.7 (k-armed bandit problem [49])**.** A classic one-state Markov decision problem is the k-armed bandit problem. The problem represents a doctor choosing between $k \geq 1$ experimental treatments for ill patients. The actions are the treatments and rewards are the health of the patients after the treatment. This learning problem gives rise to a system that is in feedback with the environment to select k actions with unknown rewards.

◇

Consider the discounted sum (2.2.5) when following a policy π. The action-value function $Q_\pi : S \times A \to \mathbb{R}$ describes the expected discounted reward starting in state s, processing action $a \in A$ and then following a given policy π:

$$Q_\pi(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_{a_t}(s_t, s_{t+1}) \;\middle|\; s_0 = s, a_0 = a\right], \tag{2.2.31}$$

where $\gamma \in [0, 1)$. This function is related to the value function in Sect. 2.2.2. The value function specifies the goodness of a state while the Q-function specifies the goodness of an action in a state [40].

Definition 2.2.8 (Q-function)**.** *Let the agent start in state $s \in S$, process action $a \in A$ and continue optimally. The value $R(s, a) \in \mathbb{R}$ is the reward that the agent can achieve in state s by performing action a. The action-value function for the policy π is defined as*

$$Q_\pi(s, a) = R(s, a) + \gamma \sum_{s' \in S} P(s' \mid s, a) V_\pi(s'), \tag{2.2.32}$$

where $V_\pi(s') = \max_{a' \in A}\{Q_\pi(s', a')\}$. The action-value function is also called the Q-function.

The optimal Q-function is defined by

$$Q^*(s, a) = \max_\pi\{Q_\pi(s, a)\} \tag{2.2.33}$$

taken over all possible policies π and the optimal policy is

$$\pi^*(s) = \arg\max_{a \in A}\{Q^*(s, a)\} \quad \text{for all } s \in S. \tag{2.2.34}$$

The optimal value function

$$V^*(s) = \max_{a \in A}\{Q^*(s, a)\} \tag{2.2.35}$$

is the optimal Q-function over all possible actions in a state.

The Q-function is defined in model-based reinforcement learning algorithms using transition probabilities to estimate the expected value of next-state action pairs. However, in model-free approaches such as *Q-learning*, transition probabilities are not explicitly used. Instead, the Q-function is updated based on observed rewards, allowing the agent to learn from its experiences. The estimation of the Q-function is accomplished using *Temporal-difference learning*, which updates the Q-function iteratively based on the old estimate and the error of the estimate, weighted by the learning rate:

$$\text{Estimate} = \text{OldEstimate} + \text{LearningRate}\ [\text{Target} - \text{OldEstimate}],$$

The Q-learning algorithm starts with random initial values, and the update rule results from the difference between the current and the new estimated Q-function weighted by the *learning rate* $\alpha > 0$. The learning rate can be constant or time-dependent, denoted by α_t.

Definition 2.2.9 (Q-learning)**.** *Let $R(s, a)$ be the reward for chosen action $a \in A$ in state $s \in S$ and let $\alpha_t \in (0, 1]$ be the learning rate at time $t \geq 0$. Then the Q-function is updated as follows:*

$$Q_{t+1}(s, a) = Q_t(s, a) + \alpha_{t+1}\left[R(s, a) + \gamma \max_{a' \in A}\{Q_t(s', a')\} - Q_t(s, a)\right], \tag{2.2.36}$$

where $\max_{a' \in A}\{Q_t(s', a')\}$ is the predicted maximum Q-value over all actions for the next state s' at time t.

The learning rate determines how much of the new information is used. The higher the learning rate, the more new information is taken into account and old information is disregarded. For instance, $\alpha_t = 1$ for all $t > 0$ means that the agent only considers the most recent information for learning:

$$Q(s,a) = R(s,a) + \gamma \max_{a' \in A}\{Q(s',a')\}. \tag{2.2.37}$$

The algorithm ends when the next state s' is a final or terminal state determined by the program. If this is the case, the sequence is called an *episode*. The last state is never updated, so it can be set to the reward value of the last state, i.e. $Q_0(s_F, a) = R(s_F, a)$ for all $a \in A$, where s_F denotes the final state. The algorithm computing an episode is described in Alg. 2. The number of episodes can be specified at the beginning, or a threshold for the difference of successive Q-functions can be used to determine convergence.

Algorithm 2: Q-Learning Algorithm

Input: Terminal state s_F, $\alpha_t > 0$ and $\gamma \in [0, 1)$
Output: Updated Q-function

Initialize $Q_0(s,a) = 0$ for all $s \in S$ and $a \in A$
$t = 0$
Observe current state s
do
  Select an action a
  Observe next state s' and reward $R(s,a)$
  $Q_{t+1}(s,a) = Q_t(s,a) + \alpha_{t+1}\left[R(s,a) + \gamma \max_{a'}\{Q_t(s',a')\} - Q_t(s,a)\right]$
  $s = s'$
  $t = t + 1$
while $s \neq s_F$;
return Q

The Q-table can be stored and updated at each time step. Thus the space complexity of Q-learning algorithm is $O(N \cdot |A| \cdot |S|)$ and the time complexity for K episodes is $O(N \cdot K)$, where N is the number of steps per episode.

Additionally, balancing *exploration*, where a random action is attempted, and *exploitation*, where the current best action is chosen, is crucial in Q-learning. An action can either be chosen completely at random, which is not recommended, or by a strategy like the ϵ-*Greedy strategy*. This strategy uses a small value $\epsilon > 0$, where ϵ is the probability of exploration and $1 - \epsilon$ is the probability of exploitation.

Example 2.2.8 (Q-learning on forest management)**.** Reconsider the forest management problem with three states, where s_3 is the final state. The function `mdp_Q_learning` in `R` can be utilized to calculate the optimal Q-function. For this, the learning rate $\alpha_t = \frac{1}{\sqrt{t+2}}$ and the discount factor $\gamma = 0.9$ are employed. The function performs 10000 episodes by default.

```
> library(MDPtoolbox)
> mdp_Q_learning(P, R, 0.9)$Q
         [,1]     [,2]
[1,] 3.313938 2.895431
[2,] 3.698780 3.993937
[3,] 4.112808 4.887533
> mdp_Q_learning(P, R, 0.9)$policy
[1] 1 2 2
```

The values approximate the optimal value function obtained in the value iteration algorithm, and the optimal policy remains the same.

◇

In the following, the convergence of the Q-learning algorithm is considered. Define $t_i(s,a)$ as the index of time t_i that action a is taken in state s [52]. The complete proof of the convergence can be found in [25], which is an alternative version to the proof provided by Watkins and the main idea is to use the difference $Q_t(s,a) - Q^*(s,a)$ in Eq. (2.2.36).

Theorem 2.2.5. *The Q-learning algorithm converges to the optimal function Q^* as $t \to \infty$ for $0 \leq \alpha_t < 1$ with probability 1 when the following holds:*

$$\sum_{i=1}^{\infty} \alpha_{t_i(s,a)} = \infty \quad \text{and} \quad \sum_{i=1}^{\infty} \alpha_{t_i(s,a)}^2 < \infty \text{ for all } s \in S \text{ and } a \in A. \tag{2.2.38}$$

The two conditions for convergence ensure that each state-action pair is visited infinitely often and that the learning rate decreases over time. The learning rate is often set to $\alpha_t = \frac{1}{t}$ due to the above conditions [41].

Example 2.2.9 (Influence of discount factor and learning rate)**.** This example analyzes the effect of the discount factor and learning rate on the convergence of the Q-learning algorithm. An implementation of Q-learning with constant learning rate α is considered.

The algorithm converges faster when a smaller discount factor is used. This occurs because a smaller discount factor gives less weight to future rewards. In Fig. 2.5, this behavior is illustrated by applying a learning rate $\alpha = 0.1$ to the forest management problem with different discount factors. With a discount factor of $\gamma = 0.9$, the algorithm converges to the optimal Q^*-function after 500 episodes, while for $\gamma = 0.5$, convergence is achieved after only 100 episodes. Moreover, the convergence rate is influenced by the learning rate. A smaller learning rate leads to slower convergence. This is exemplified in Fig. 2.6 for different learning rates.

◇

Understanding the influence of discount factor and learning rate is crucial in real-world applications where tuning these parameters is essential for achieving desired performance. For instance, in autonomous driving, rapid convergence to optimal policies is critical for ensuring safety and efficiency on the road.

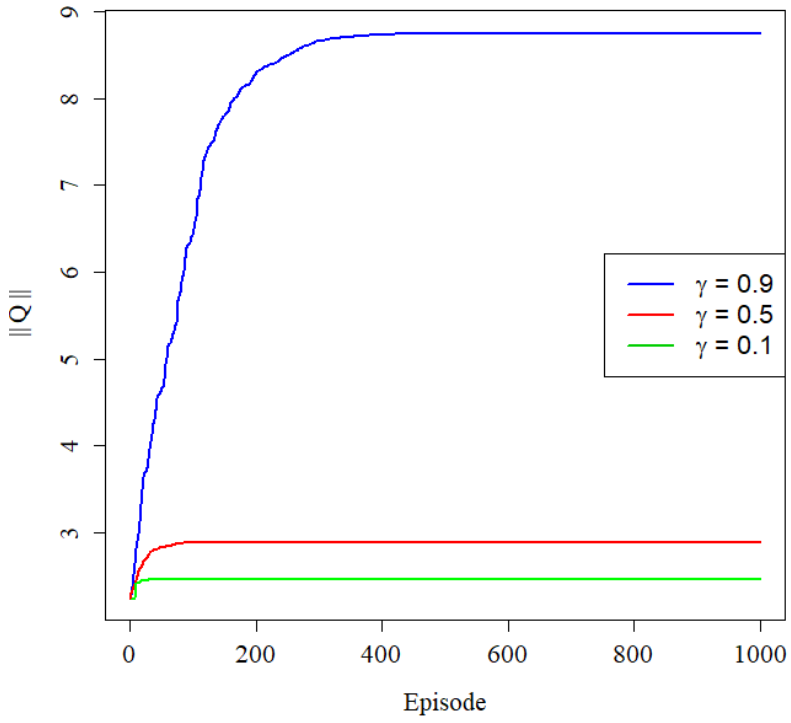


Figure 2.5: Q-learning with $\alpha = 0.1$.

Q-Learning in Practice

Several experiments were conducted to evaluate the practical performance of Q-learning utilizing various decision problems and known environments. First the value iteration algorithm was applied, followed by the application of the Q-learning algorithm without utilizing transition probabilities. Remarkably, Q-learning was able to identify the same optimal policy as the value iteration algorithm [2, 11, 51]. In some cases, multiple optimal policies were obtained by the Q-learning algorithm. This indicates that Q-learning may offer more flexibility, which can be beneficial in complex environments. However, it was observed that for larger problem instances, Q-learning required significantly more time to converge compared to the value iteration algorithm. For instance, on a 10×10 grid, while the value iteration algorithm typically completed within 1-2 seconds, Q-learning necessitated approximately 15 seconds to achieve an acceptable result after running through 20000 episodes [24].

These findings underscore the limitations of relying solely on the value iteration algorithm, especially in scenarios where transition probabilities are unknown. While Q-learning offers an alternative approach, it also suffers from the drawback of slower convergence, especially noticeable in larger problem instances.

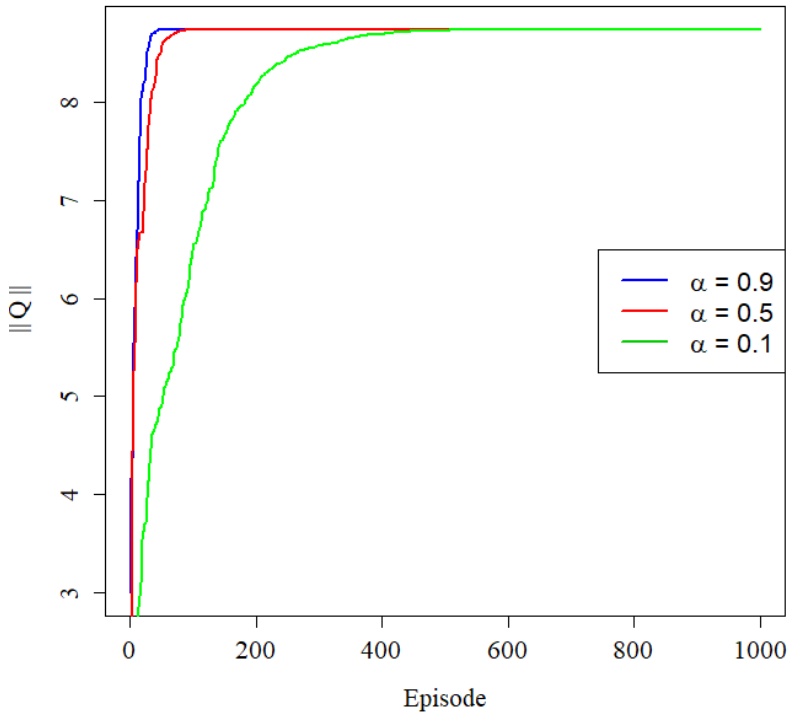


Figure 2.6: Q-learning with $\gamma = 0.9$.

2.3 Stochastic Dynamics

In dynamic systems, addressing the complexities of the real world is a significant challenge. This section provides a mathematical foundation of stochastic dynamics, where randomness and uncertainty are components of the system. Fundamental concepts such as Brownian motion, stochastic differential equations, Itô integrals, and Itô's formula form the cornerstone of this framework, facilitating the modeling and analysis of systems influenced by external factors [12]. Brownian motion involves random fluctuations over time, while stochastic differential equations describe the evolution of systems considering Brownian motion. The Itô integral, named after the Japanese mathematician Kiyoshi Itô, is a stochastic integral with respect to Brownian motion or other stochastic processes, such that application of standard integration methods may not be feasible [29]. Itô's formula is a fundamental result in stochastic calculus, providing a way to calculate differentials of functions that depend on stochastic processes [30].

In practical applications, solving stochastic differential equations analytically can be challenging or even impossible. Numerical methods, such as the Euler-Maruyama method, offer an alternative for approximating solutions to stochastic differential equations. The Euler-Maruyama method approximates the stochastic process at discrete time intervals, allowing for the simulation of stochastic systems and the analysis of

their behavior over time. For instance, in finance, Platen and Bruti-Liberati (2010) [38] examined the Euler-Maruyama method along with other numerical methods for solving stochastic differential equations with jumps, including derivative pricing, portfolio optimization, and risk management strategies. However, the discussion in existing literature often lacks insights into the latest developments and innovations in this field. Particularly notable are the advancements in applying machine learning techniques and addressing complex financial problems.

Integrating the principles of stochastic dynamics with machine learning methodologies enables the development of algorithms capable of effectively navigating environments characterized by noise and uncertainty. This concept serves as the key idea of the next chapter.

2.3.1 Brownian Motion

Brownian motion, also known as the Wiener process, is a real-valued stochastic process named after the mathematician Norbert Wiener, who mathematically formalized the process in the 1920s [53]. This process was first observed by the botanist Robert Brown in 1827, as he analyzed the irregular movement of pollen grains of Clarkia pulchella in water [5]. Nowadays, Brownian motion finds applications in various fields such as stochastics, financial mathematics, and physics. For instance, it can be used to describe the random fluctuations in the price of financial assets.

Brownian motion is an example of a continuous-time Markov process and is obtained by integrating white noise, a Gaussian process with zero mean, finite variance and uncorrelated random variables. The increments of Brownian motion are independent and normally distributed.

Definition 2.3.1 (Brownian motion)**.** *A stochastic process* $(B_t(\omega))_{t\geq 0}$ *defined on a probability space* $(\Omega, \mathcal{F}, \mathbb{P})$ *is called* Brownian motion *if the following holds:*

i) $B_0 = 0$ *(a.s.).*

ii) The paths $t \to B_t(\omega)$, $\omega \in \Omega$, *are continuous (a.s.).*

iii) The process (B_t) *has independent increments, i.e. for all* $0 \leq t_1 \leq \ldots \leq t_n$ *the increments* $B_{t_2} - B_{t_1}, \ldots, B_{t_n} - B_{t_{n-1}}$ *are independent random variables.*

iv) The process has stationary increments which are Gaussian distributed, i.e. for all s, t *with* $0 \leq s < t$, $B_t - B_s$ *follows a normal distribution with mean* 0 *and variance* $t - s$, *denoted as* $B_t - B_s \sim \mathcal{N}(0, t-s)$.

Definitions (i) and (iv) imply that $\mathbb{E}[B_t] = 0$ and $\mathrm{Var}(B_t) = \mathbb{E}\left[B_t^2\right] = t$, since $\mathbb{E}\left[B_t - B_s\right] = 0$ and $\mathbb{E}\left[(B_t - B_s)^2\right] = \mathrm{Var}(B_t - B_s) = t - s$ for all $t > s \geq 0$.

Brownian motion can be simulated by dividing the time interval $[0, T]$ with $T > 0$ into N subintervals of equal length $\Delta t = \frac{T}{N} > 0$, where $t_i = i \cdot \Delta t$ for all $0 \leq i \leq N$.

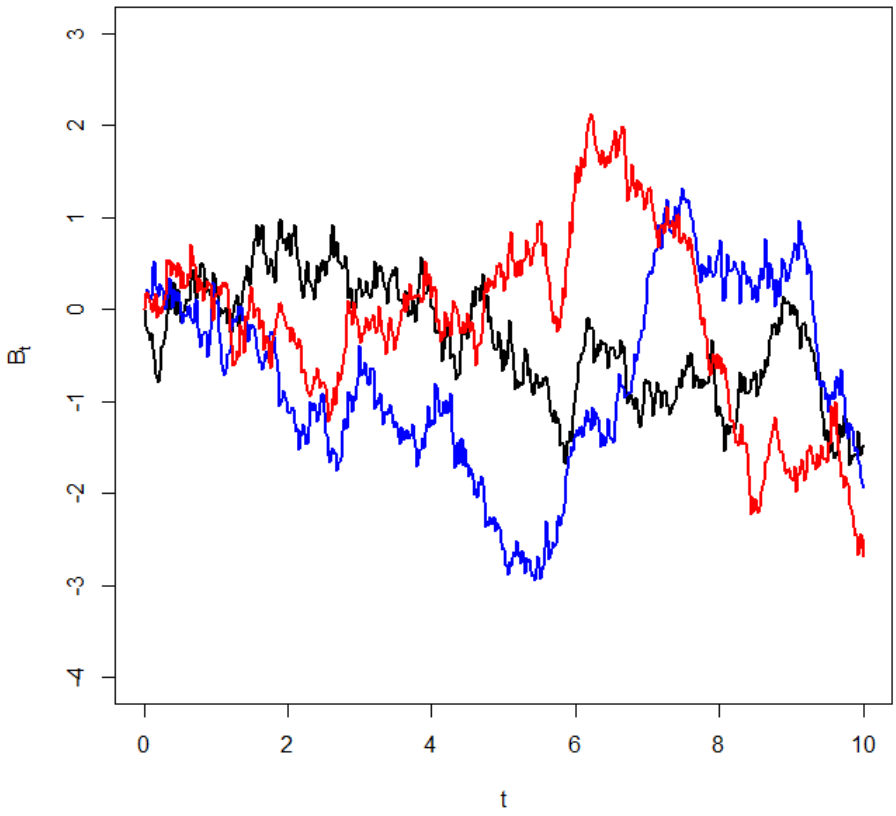


Figure 2.7: Paths of Brownian motion.

For each $i = 1, \ldots, N$, independent random variables with distribution $\mathcal{N}(0,1)$ are generated and denoted by Z_i. The independent increments can be computed as

$$B(t_i) - B(t_{i-1}) = \sqrt{\Delta t} Z_i, \quad 1 \leq i \leq N. \tag{2.3.1}$$

By definition $B(t_i) - B(t_{i-1}) \sim \mathcal{N}(0, t_i - t_{i-1})$, which is equal to $\sqrt{\Delta t} \cdot \mathcal{N}(0,1)$. The use of $\sqrt{\Delta t}$ in the equation ensures that the variance of each increment is indeed Δt. Then, Brownian motion at the selected points is the cumulative sum

$$B(t_i) = B(t_{i-1}) + \sqrt{\Delta t} Z_i \text{ with } B(t_0) = 0. \tag{2.3.2}$$

Note that the expected value of the exponential of a linear combination of the Brownian motion at time t, denoted by $\mathbb{E}[\mathrm{e}^{uB_t}]$ is given by $\mathrm{e}^{\frac{u^2 t}{2}}$ for all $u \in \mathbb{R}$. This can be derived by integrating the exponential function $\mathbb{E}[\mathrm{e}^{u\sqrt{\Delta t} Z}]$ with respect to the probability density function of a standard normal distribution, where $B_t = \sqrt{t} Z$ and Z follows a standard normal distribution.

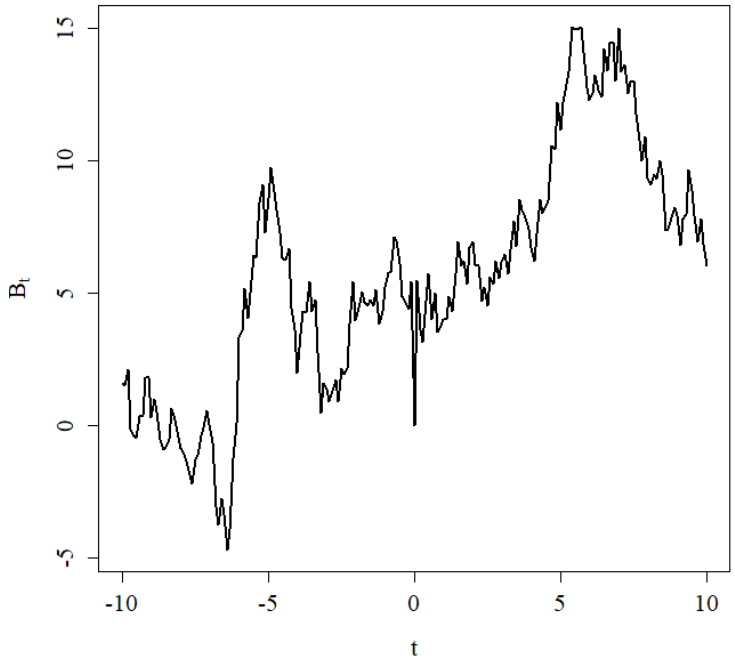


Figure 2.8: Two-sided Brownian motion.

Example 2.3.1. Brownian motion can be simulated in `R` using the function `BM`:

```
> library(Sim.DiffProc)
> B<-BM(N = 500, t0 = 0, T = 10)
```

Three paths of Brownian motion are shown in Fig. 2.7.

◇

Two-sided Brownian motion $(B_t)_{t\in\mathbb{R}}$ can be considered as two independent (scalar) Brownian motions, where (B_t) consists of (B_t^1) for $t \geq 0$ and $(B{-}t^2)$ for $t < 0$. This representation allows for the modeling of Brownian motion across both positive and negative time intervals. Formally, two-sided Brownian motion is defined as

$$(B_t) = \begin{cases} (B_t^1), & t \geq 0, \\ (B_{-t}^2), & t < 0. \end{cases} \tag{2.3.3}$$

Two-sided Brownian motion is simulated in the interval $[-T, T]$ with $T > 0$. The simulation for partition $[0, T]$ remains the same and for partition $[-T, 0]$, with $t_0 = -T$ and $t_N = 0$, the simulation proceeds as follows:

$$B_{t_N} = 0 \tag{2.3.4}$$

$$B_{t_i} = B_{t_{i+1}} + \sqrt{\Delta t} R_{i+1}, \quad 1 \leq i \leq N-1, \tag{2.3.5}$$

where $R_1, \ldots, R_N$ are independent random variables with distribution $\mathcal{N}(0, 1)$.

Example 2.3.2. Two-sided Brownian motion can be simulated in `R` as follows:

```
> t <- seq(-10,10,0.1)
# Generate normally distributed random variables
> y <- rnorm(t,mean=0,sd=1)
# Use cumulative sum to generate Brownian motion
> Bt <- c(cumsum(y))
> Bt[101] <- 0
> plot(t,Bt,type="l",ylab=expression(B[t]),lwd=2)
```

The function `rnorm` generates a vector of normally distributed random numbers and the function `cumsum` calculates the cumulative sum. The output of this code is depicted in Fig. 2.8.

◇

Proposition 2.3.1. *Brownian motion has the following properties:*

i) Brownian motion (B_t) *has the distribution* $\mathcal{N}(0,t)$*, i.e. the probability density function is*

$$\frac{1}{\sqrt{2\pi t}}\,\mathrm{e}^{\frac{-x^2}{2t}}, \quad t>0. \tag{2.3.6}$$

ii) The covariance is $\mathrm{Cov}[B_s,B_t]=\mathbb{E}[B_sB_t]=\min\{s,t\}$.

There are several versions of Brownian motion endowed with the same statistical properties, so statistically there is only one Brownian motion in terms of the distribution. Next, some advanced properties of Brownian motions are presented for which the following definition is required.

Definition 2.3.2 (Finite variation)**.** *Let* g *be a real-valued function. Consider all possible partitions of the time interval* $[0,T]$ *into subintervals of the form* $0=t_0<t_1<\ldots<t_i<\ldots<t_n=T$*. If there exists an integer* $M>0$ *such that*

$$\sum_{i=0}^{n-1}|g(t_{i+1})-g(t_i)|\leq M \tag{2.3.7}$$

for all partitions of $[0,T]$*, then* g *is of* finite variation*; otherwise,* g *is of* infinite variation*.*

For example, consider the function

$$g(x)=\begin{cases}0, & \text{if } x=0\\ x^2\sin(x^{-1}), & \text{otherwise}\end{cases} \tag{2.3.8}$$

on $[0,1]$. This function is of finite variation, since the x^2 term causes it to vanish rapidly [42]. A function of finite variation on the interval $[0,T]$ is differentiable almost everywhere. In particular, a Lipschitz continuous function is of finite variation. If the square is considered, it is called *quadratic variation*. A function with infinite variation can still have quadratic variation, as squaring will make small increments smaller, which is the case for Brownian motion.

Proposition 2.3.2. *Brownian motion has the following additional properties:*

i) Infinite variation: Almost every path of Brownian motion has infinite variation on each finite interval.

ii) Quadratic variation: Let $\{t_0^{(n)}, \dots, t_k^{(n)}\}$, $n \in \mathbb{N}$, *be a sequence of partitions of the time interval* $[a, b]$. *Then the quadratic variation converges as*

$$\lim_{\Delta t^{(n)} \to 0} \sum_i \left| B(t_{i+1}^{(n)}) - B(t_i^{(n)}) \right|^2 = b - a \tag{2.3.9}$$

where $\Delta t^{(n)} = \max_i |t_{i+1}^{(n)} - t_i^{(n)}|$.

iii) Nowhere differentiability: The paths of Brownian motion are almost surely nowhere differentiable.

The first property underscores the unpredictable nature of Brownian motion, which implies that it lacks uniform boundedness in its slope, rendering it non-differentiable at any point, consequently leading to the third property. The quadratic variation is crucial in the construction of stochastic integrals.

Brownian motion in higher dimensions

Consider n independent scalar Brownian motions $(B_t^1), \dots, (B_t^n)$. The stochastic process $(B_t)_{t \geq 0}$ defined as

$$(B_t) = ((B_t^1), \dots, (B_t^n))^{\mathrm{T}} \tag{2.3.10}$$

is called an *n-dimensional Brownian motion*. The conditions are similar to those of a scalar Brownian motion:

i) $B_0 = 0$ almost surely.

ii) (B_t) has continuous paths almost surely.

iii) (B_t) has independent increments.

iv) (B_t) has stationary increments and $B_t - B_s \sim \mathcal{N}(0, (t-s)I)$ for $t > s \geq 0$, where I is the $n \times n$ identity matrix.

The covariance matrix of an n-dimensional Brownian motion (B_t) is given by tI.

2.3.2 Stochastic Differential Equations

Dynamical systems are often influenced by uncertain parameters or fluctuating initial conditions. Some processes might be overlooked due to limited knowledge or analytical capabilities, leading to increased uncertainty of the solutions. This uncertainty is referred to as *noise* and can be represented by stochastic processes like Brownian motion.

Interestingly, noise can be beneficial and valuable in some cases. First, Langevin developed a differential equation to model the motion of Brownian particles and obtained a noisy equation known as stochastic differential equation [31]. Stochastic differential equations find application in modeling dynamical systems affected by noise. Examples of such applications include analyzing biological systems subject to random external disturbance, or modeling stocks with uncertain price movements [21]. Solutions to stochastic differential equations are continuous-time stochastic processes.

Stochastic differential equations provide valuable frameworks for understanding complex systems influenced by uncertainty. They enable to quantify the influence of noise on system behavior and predict the evolution of systems over time in uncertain environments.

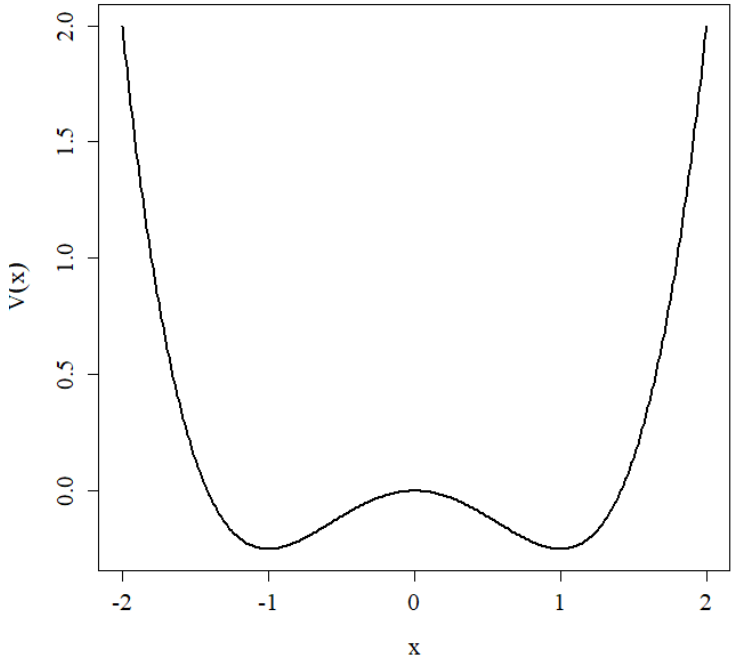


Figure 2.9: Plot of $V(x)$.

Deterministic dynamical systems are often described by differential equations. Consider a model where a single variable represents a quantity that can only change along one dimension, similar to a point moving along a straight line. This variable evolves over time according to the differential equation $\dot{x} = x - x^3$ [12], which exhibits three equilibrium states $-1, 0, 1$. The equation can be described by

$$\dot{x} = x - x^3 = -\frac{d}{dx}V(x), \tag{2.3.11}$$

where $V(x) = \frac{1}{4}x^4 - \frac{1}{2}x^2$ has two minimal values, as illustrated in Fig. 2.9. Duan referred to this system as a *double-well system* in his book, and we will use this term for

our discussion.

In complex systems, fluctuations are abundant. Assume that a double-well system is subjected to white noise, corresponding to the generalized derivative of Brownian motion. Then this system can be modeled as

$$\dot{x} = \left(x - x^3\right) + kx\frac{d}{dt}B_t, \tag{2.3.12}$$

where k is a constant and kx is the noise intensity. In differential form with the initial condition X_0, the equation can be written as

$$dX_t = \left(X_t - X_t^3\right)dt + kX_t dB_t, \quad X_0 = x_0, \tag{2.3.13}$$

which provides an example of a stochastic differential equation. Expressed in integral form, this equation becomes

$$X_t = x_0 + \int_0^t \left(X_u - X_u^3\right)du + k\int_0^t X_u dB_u. \tag{2.3.14}$$

The term $\int_0^t \left(X_u - X_u^3\right)du$ is a Riemann integral, but $\int_0^t X_u dB_u$ is a stochastic integral.

In complex systems subject to random fluctuations, understanding the dynamics requires stochastic integration. The Riemann-Stieltjes integral extends the Riemann integral by allowing integration with respect to non-constant functions. However, it reaches its limitations when considering integration with respect to stochastic processes. Recall the definition of Riemann-Stieltjes integral. A *Riemann-Stieltjes integral* on the interval $[0, T]$ has the form

$$\int_0^T h(t)dg(t), \tag{2.3.15}$$

with *integrand* h and *integrator* g. This integral is defined as the limit of the Riemann sum

$$\sum_{i=0}^{n-1} h(\tau_i)\left(g(t_{i+1}) - g(t_i)\right), \tag{2.3.16}$$

where $0 = t_0 < t_1 < \ldots < t_i < \ldots < t_n = T$ provides a partition of the interval $[0, T]$ and $\tau_i \in [t_i, t_{i+1}]$. Thus, if the limit of the sum exists, the Riemann-Stieltjes integral exists. A sufficient condition for existence is that the integrator is of finite variation.

Theorem 2.3.1 (Sufficient condition)**.** *If the integrand h is continuous on $[0, T]$ and the integrator g is of finite variation in $[0, T]$, then the Riemann-Stieltjes integral $\int_0^T h(t)dg(t)$ exists.*

As stated in Prop. 2.3.2, almost all paths of a Brownian motion are of infinite variation on $[0, T]$. Thus the stochastic integral $\int_0^t X_u dB_u$ is not a Riemann-Stieltjes integral.

Theorem 2.3.2 (Necessary condition)**.** *If the limit*

$$\lim_{n\to\infty} \sum_{i=0}^{n-1} h(\tau_i) \left(g(t_{i+1}) - g(t_i)\right), \tag{2.3.17}$$

with $\tau_i = t_i$ converges for each continuous integrand h, the integrator g is of finite variation.

Definition 2.3.3 (Itô integral)**.** *The stochastic integral*

$$\int_0^t h(t,\omega) dB_t(\omega) \tag{2.3.18}$$

is called Itô integral.

Consider the integral $\int_0^t X_s dB_s$, where X_t is a stochastic process and dB_t represents Brownian motion. In financial mathematics, a possible example for X_t could be the value of a stock, which evolves over time according to a stochastic differential equation. This differential equation could model the behavior of the stock price, taking into account returns, dividends, volatility, and other factors [6]. The change dB_t represents the random fluctuations in the market or the unpredictability of price movements. The solution of the integral $\int_0^t X_s dB_s$ measures the cumulative impact of market fluctuations on the value of the stock.

Next, the formal definition of stochastic differential equations will be explored.

Definition 2.3.4 (Scalar stochastic differential equation (SDE))**.** *A* scalar stochastic differential equation *is defined as*

$$dX_t = b(X_t)dt + \sigma(X_t)dB_t, \quad X_0 = x, \tag{2.3.19}$$

where B_t is a scalar Brownian motion. The first summand is called drift term *and the second one* diffusion term*. Moreover, $\sigma(X_t)$ is called* noise intensity*.*

SDE (2.3.19) can be interpreted in integral form as

$$X_t = x + \int_0^t b(X_u)du + \int_0^t \sigma(X_u)dB_u, \tag{2.3.20}$$

where the last term $\int_0^t \sigma(X_u)dB_u$ is an Itô integral, which is described in the following.

Definition 2.3.5 (n-dimensional stochastic differential equation)**.** *An n-*dimensional stochastic differential equation *is defined as*

$$dX_t = b(X_t)dt + \sigma(X_t)dB_t, \quad X_0 = x, \tag{2.3.21}$$

where $X_t = (X_t^1, X_t^2, ..., X_t^n)$ and $B_t = (B_t^1, B_t^2, ..., B_t^n)$ is an n-dimensional Brownian motion.

2.3.3 Itô's Integral

The Itô integral defines the integration of the stochastic process $h(t,\omega)$, called *integrand*, with respect to Brownian motion B_t, called *integrator*, in probability space $(\Omega, \mathcal{F}, P)$ [12]. When the Itô integral exists, it can be evaluated by a mean square limiting process.

Consider dividing the interval $[0,T]$ into subintervals of length $\Delta t = \frac{T}{n}$ with $t_0 = 0$, $t_i = i \cdot \Delta t$ for $1 \leq i \leq n-1$ and $t_n = T$. Let $B_i = B(t_i)$ and $\Delta B_i = B_{t_{i+1}} - B_{t_i}$. Consider the sum

$$\sum_{i=0}^{n-1} h(\tau_i, \omega) \left(B_{i+1}(\omega) - B_i(\omega)\right), \quad \tau_i \in [t_i, t_{i+1}],\ \omega \in \Omega. \tag{2.3.22}$$

As a consequence of unbounded variation of Brownian motion paths, the points τ_i within each subinterval $[t_i, t_{i+1}]$ cannot be chosen arbitrarily. For instance, if the integrand B_t is chosen and left end point $\tau_i = t_i$ of each interval $[t_i, t_{i+1}]$, the sum amounts to

$$s_n = \sum_{i=0}^{n-1} B_i(\omega) \left(B_{i+1}(\omega) - B_i(\omega)\right). \tag{2.3.23}$$

The mean of this sum is calculated as

$$\mathbb{E}[s_n] = \sum_{i=0}^{n-1} \mathbb{E}\left[\left(B_i(\omega) - B_0(\omega)\right)\left(B_{i+1}(\omega) - B_i(\omega)\right)\right] = 0 \tag{2.3.24}$$

since Brownian motion has independent increments. Thus, the Itô integral is evaluated at the left-hand side of an interval. Making the stepsize smaller and smaller allows to approach the true value of the integral.

Let the integrand $h(t,\omega)$ be continuous in t (a.s.), and divide the time interval $[T_0, T_f]$ into subintervals of equal length $\Delta t^n = \frac{T_f - T_0}{n}$ such that $T_0 = t_0^n < t_1^n < \ldots < t_n^n = T_f$, $n \in \mathbb{N}$, with $\Delta t^n \to 0$ as $n \to \infty$. The Itô integral is obtained by evaluating the limit in mean square

$$\int_{T_0}^{T_f} h(t,\omega) dB_t = \lim_{n\to\infty} \text{ in m.s. } \sum_{i=0}^{n-1} h(t_i^n, \omega) \left(B(t_{i+1}^n) - B(t_i^n)\right), \tag{2.3.25}$$

where m.s. denotes mean square and h is evaluated at the left end point of each subinterval $[t_i^n, t_{i+1}^n]$.

Example 2.3.3 (Itô integral)**.** Consider the Itô integral $\int_S^T B_t dB_t$. Divide the time interval $[S,T]$ with $S<T$ into subintervals of equal length Δt^n: $S=t_0^n<t_1^n<\ldots<t_n^n=T$, $n\in\mathbb{N}$. Taking the sum

$$\begin{aligned}\sum_i B(t_i^n)\left(B(t_{i+1}^n)-B(t_i^n)\right) &= \frac{1}{2}\sum_i\left(B^2(t_{i+1}^n)-B^2(t_i^n)\right)-\frac{1}{2}\sum_i\left(B(t_{i+1}^n)-B(t_i^n)\right)^2\\ &= \frac{1}{2}\left(B_T^2-B_S^2\right)-\frac{1}{2}\sum_i\left(B(t_{i+1}^n)-B(t_i^n)\right)^2.\end{aligned}$$

The second term converges in mean square to the length of the interval $T-S$ as the number of subintervals n tends to infinity due to the quadratic variation of Brownian motion:

$$\lim_{n\to\infty}\ \text{in m.s.}\ \sum_i\left(B(t_{i+1}^n)-B(t_i^n)\right)^2=T-S. \tag{2.3.26}$$

Hence $\int_S^T B_t dB_t=\frac{1}{2}\left(B_T^2-B_S^2\right)-\frac{1}{2}(T-S)$.

◇

In the following, some properties of the Itô integral that are crucial in the analysis and application of stochastic calculus are presented.

Proposition 2.3.3. *The Itô integral has the following properties.*

(i) Linearity:

$$\begin{aligned}&\int_S^T\left(f(t,\omega)+g(t,\omega)\right)dB_t=\int_S^T f(t,\omega)dB_t+\int_S^T g(t,\omega)dB_t\quad a.s.,\\ &\int_S^T cf(t,\omega)dB_t=c\int_S^T f(t,\omega)dB_t\quad a.s.,\\ &\int_S^T f(t,\omega)dB_t=\int_S^\tau f(t,\omega)dB_t+\int_\tau^T f(t,\omega)dB_t\quad a.s.,\end{aligned}$$

where c is a constant and $S<\tau<T$.

(ii) Zero mean property:

$$\mathbb{E}\left[\int_S^T f(t,\omega)dB_t\right]=0. \tag{2.3.27}$$

(iii) Itô isometry *in scalar case:*

$$\mathbb{E}\left[\left(\int_S^T f(t,\omega)dB_t\right)^2\right]=\mathbb{E}\left[\int_S^T f^2(t,\omega)dt\right]. \tag{2.3.28}$$

In the n-dimensional case, the Itô isometry for an $n\times n$ matrix $F(t,\omega)$ is expressed as

$$\mathbb{E}\left[\left\|\int_S^a F(t,\omega)dB_t\right\|^2\right]=\mathbb{E}\left[\int_S^a \mathrm{Tr}(FF^{\mathrm{T}})(t,\omega)dt\right], \tag{2.3.29}$$

where Tr denotes the trace operator.

2.3.4 Itô's Formula

One of the most important tools in the analysis of stochastic processes is Itô's formula, known as the chain rule for stochastic calculus. Itô's formula is used to bypass stochastic integrals and solve stochastic differential equations analytically. The key idea is applying the Taylor expansion to a given smooth function, discarding higher order terms due to the fact that Brownian motion has quadratic variation.

Integrating a stochastic differential equation provides a solution, but this process can be challenging, especially for complex or non-closed-form stochastic differential equations. Itô's formula provides an alternative method for solving stochastic differential equations. Rather than integrating directly, Itô's formula allows for the computation of derivatives of functions dependent on stochastic processes. Subsequently, one can solve the resulting differential equation for that function to obtain the solution to the stochastic differential equation.

Definition 2.3.6 (Itô's formula in scalar case)**.** *Consider the SDE*

$$dX_t = b(X_t)dt + \sigma(X_t)dB_t, \tag{2.3.30}$$

where b, σ are scalar functions and B_t is a scalar Brownian motion. Let g be a twice differentiable scalar function. Then Itô's formula *states*

$$\begin{aligned} dg(t, X_t) = &\left[\frac{\partial}{\partial t}g(t, X_t) + b(X_t)\frac{\partial}{\partial x}g(t, X_t) + \frac{1}{2}\sigma^2(X_t)\frac{\partial^2}{\partial x^2}g(t, X_t)\right]dt \\ &+ \frac{\partial g}{\partial x}(t, X_t)\sigma(X_t)dB_t. \end{aligned}$$

The term $\frac{1}{2}\sigma^2(X_t)\frac{\partial^2}{\partial x^2}g(t, X_t)$ is called the Itô correction term.

The key rules for applying Itô's formula are

$$dB_t dB_t = dt \quad \text{and} \quad dtdt = dtdB_t = 0.$$

Then Itô's formula is given equivalently as

$$dg(t, X_t) = \frac{\partial g}{\partial t}(t, X_t)dt + \frac{\partial g}{\partial x}(t, X_t)dX_t + \frac{1}{2}\frac{\partial^2 g}{\partial x^2}(t, X_t)(dX_t)^2, \tag{2.3.31}$$

where $(dX_t)^2$ is calculated using these rules.

Example 2.3.4 (Itô's formula - Geometric Brownian motion)**.** Consider the linear scalar SDE

$$dX_t = \mu X_t dt + \alpha X_t dB_t, \tag{2.3.32}$$

where μ and α are real constants and $X_t > 0$ (a.s.). Rewrite the SDE as

$$\frac{dX_t}{X_t} = \mu dt + \alpha dB_t. \tag{2.3.33}$$

For a deterministic differential equation, the left-hand side is $d\ln(X_t)$. Therefore the stochastic differential of $\ln(X_t)$ is considered. Applying Itô's formula gives

$$d\ln(X_t) = \frac{1}{X_t}dX_t + \frac{1}{2}\left(-\frac{1}{X_t^2}\right)(dX_t)^2 = \frac{dX_t}{X_t} - \frac{1}{2}\alpha^2 dt = \left(\mu - \frac{1}{2}\alpha^2\right)dt + \alpha dB_t. \tag{2.3.34}$$

Integration from 0 to t yields

$$\ln\left(\frac{X_t}{X_0}\right) = \left(\mu - \frac{1}{2}\alpha^2\right)t + \alpha B_t.$$

The solution of this SDE is obtained by taking the exponential,

$$X_t = X_0 e^{\left(\mu - \frac{1}{2}\alpha^2\right)t + \alpha B_t}. \tag{2.3.35}$$

This process is called *geometric Brownian motion* starting at X_0. For example, if $\mu = 0.8$ and $\alpha = 0.4$, geometric Brownian motion can be generated in `R` as follows:

```
> library(Sim.DiffProc)
# Generate geometric Brownian motion
# dX(t) = 0.8*X(t)*dt+0.4*X(t)*dB(t)
> GBM(N =100,x0=1,t0=0,T=1,theta=0.8,sigma=0.4)
```

This is depicted in Fig. 2.11. ◇

Definition 2.3.7 (Itô's formula in vector case)**.** *Consider the SDE system in* $\mathbb{R}^n$

$$dX_t = b(X_t)dt + \sigma(X_t)dB_t, \tag{2.3.36}$$

where $b \in \mathbb{R}^n, \sigma \in \mathbb{R}^{n\times n}$ *and* B_t *is an* n*-dimensional Brownian motion. For a scalar smooth function* $g(t,x)$*,* $(t,x) \in \mathbb{R}\times\mathbb{R}^n$*,* Itô's formula in vector case *is given as*

$$\begin{aligned} dg(t,X_t) = &\left[\frac{\partial}{\partial t}g(t,X_t) + b^T\nabla g(t,X_t) + \frac{1}{2}\operatorname{Tr}\left[\sigma\sigma^T \mathrm{H}(g)\right](t,X_t)\right]dt \\ &+ \left(\nabla g(t,X_t)\right)^T \sigma(X_t)dB_t, \end{aligned}$$

where Tr *denotes the trace of a matrix and* H *is the Hessian matrix.*

Example 2.3.5 (Two-dimensional Itô formula)**.** Consider the two-dimensional SDE system

$$\begin{aligned} dX_t &= Y_t dt + aX_t dB_t^1, \\ dY_t &= \left(-X_t + X_t^3\right)dt + bY_t dB_t^2. \end{aligned}$$

Applying Itô's formula to $g = \frac{1}{2}(x^2+y^2)$ yields

$$\frac{1}{2}d(X_t^2 + Y_t^2) = \left[X_t^3 Y_t + \frac{1}{2}\left(a^2X_t^2 + b^2Y_t^2\right)\right]dt + aX_t^2 dB_t^1 + bY_t^2 dB_t^2. \tag{2.3.37}$$

In `R` the function `snssde2d` can be used to generate a solution path of two-dimensional SDE.

```
> library(Sim.DiffProc)
# dX(t) = Y(t)*dt+0.5*X(t)*dB1(t)
# dY(t) = (-X(t)+X(t)^3)*dt+0.3*Y(t)dB2(t)
> fx <- expression(y , (-x+x^3))
> gx <- expression(0.5*x, 0.3*y)
> mod2d1 <- snssde2d(drift=fx,diffusion=gx,x0=c(x0=1,y0=0.5),M=1000,
                     method="taylor")
# Plot in plane
> plot2d(mod2d1, lwd=2)
```

The result is depicted in Fig. 2.10. ◇

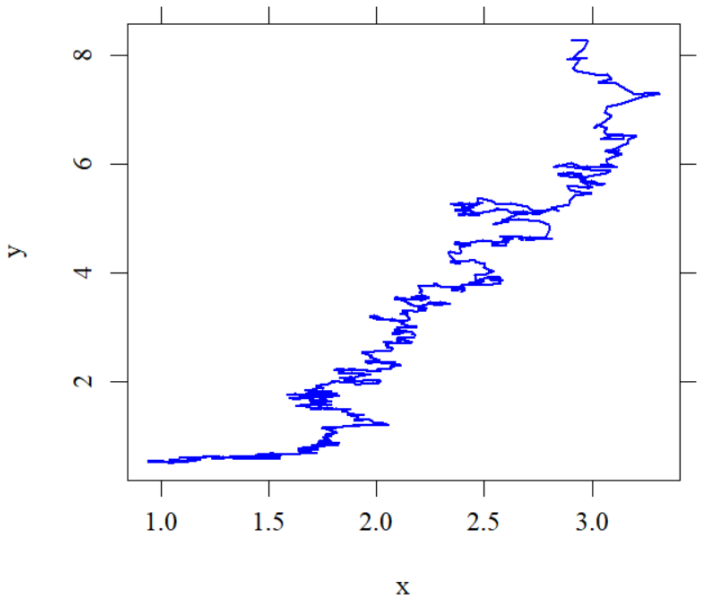


Figure 2.10: Solution of two-dimensional SDE (Ex. 2.3.5).

The *stochastic product rule* is obtained by applying the two-dimensional Itô formula to $g(x, y) = xy$:

$$d(X_t Y_t) = X_t dY_t + Y_t dX_t + dX_t dY_t. \tag{2.3.38}$$

Example 2.3.6 (Stochastic product rule)**.** Let $X_t = \sin(t)$ and $Y_t = B_t$ be a scalar Brownian motion. Then using the stochastic product rule and the fact that $dt dB_t = 0$,

$$\begin{aligned} d(\sin(t)B_t) &= d(\sin(t))B_t + \sin(t)dB_t + d(\sin(t))dB_t \\ &= \cos(t)B_t dt + \sin(t)dB_t + \cos(t)dt dB_t \\ &= \cos(t)B_t dt + \sin(t)dB_t. \end{aligned}$$

◇

Platen and Kloeden discuss the solution methodology for linear stochastic differential equations, drawing parallels to the process for linear ordinary differential equations (ODEs). They highlight that, similar to ODEs, the general solution of a linear SDE can be explicitly determined. The solution method involves finding a fundamental solution Φ_{t,t_0} of an associated homogeneous differential equation, which serves as a basis, as all other solutions can be expressed in terms of it. By utilizing Itô's formula and using the initial condition $\Phi_{t_0,t_0} = 1$, they derive an expression for the process $\ln \Phi_{t,t_0}$ [37]. Once the fundamental solution is determined, the general solution can be obtained by applying the stochastic product rule to $\Phi_{t,t_0}^{-1} X_t$ and then integrating it.

Theorem 2.3.3 (General linear scalar SDE)**.** *Consider a general linear stochastic differential equation*

$$dX_t = [a_1(t)X_t + a_2(t)]\,dt + [b_1(t)X_t + b_2(t)]\,dB_t, \quad X_{t_0} \text{ given}\,, \tag{2.3.39}$$

where a_1, a_2, b_1 and b_2 are deterministic functions and B_t is a scalar Brownian motion. The fundamental solution is given by

$$\Phi_{t,t_0} = \exp\left[\int_{t_0}^{t}\left(a_1(s) - \frac{1}{2}b_1^2(s)\right)ds + \int_{t_0}^{t} b_1(s)dB_s\right] \tag{2.3.40}$$

The general solution is

$$X_t = \Phi_{t,t_0}\left\{X_{t_0} + \int_{t_0}^{t}[a_2(s) - b_1(s)b_2(s)]\Phi_{s,t_0}^{-1}ds + \int_{t_0}^{t} b_2(s)\Phi_{s,t_0}^{-1}dB_s\right\}. \tag{2.3.41}$$

In particular, for

$$dX_t = [a_1(t)X_t + a_2(t)]dt + cdB_t, \quad X_{t_0} \text{ given}\,, \tag{2.3.42}$$

where $c > 0$ is a constant, the solution is

$$X_t = \mathrm{e}^{a_1(t-t_0)}\,X_{t_0} + \int_{t_0}^{t}\mathrm{e}^{a_1(t-s)}\,a_2(s)ds + c\int_{t_0}^{t}\mathrm{e}^{a_1(t-s)}\,dB_s. \tag{2.3.43}$$

Example 2.3.7 (Ornstein–Uhlenbeck)**.** Consider the *Langevin equation*

$$dX_t = -bX_t dt + adB_t, \quad X_0 = x_0, \tag{2.3.44}$$

where $a, b > 0$ are constants. The solution due to Eq. (2.3.43) is

$$X_t = x_0 e^{-bt} + ae^{-bt}\int_0^t e^{bs}dB_s \tag{2.3.45}$$

which is called the *Ornstein–Uhlenbeck process*. This is a Gaussian process and can be applied for instance to the modeling the evolution of phenotypes or interest rates in financial mathematics.

In the case that the initial value X_0 is a real number, the expectation due to the zero mean property of the Itô integral is

$$\mathbb{E}[X_t] = x_0 \, \mathrm{e}^{-bt} \tag{2.3.46}$$

and the variance, using the Itô isometry, is calculated as

$$\mathrm{Var}(X_t) = \mathbb{E}\left[\left(a \, \mathrm{e}^{-bt} \int_0^t \mathrm{e}^{bs} \, dB_s\right)^2\right] = a^2 \, \mathrm{e}^{-2bt} \, \mathbb{E}\left[\int_0^t \mathrm{e}^{2bs} \, ds\right] = \frac{a^2}{2b}\left(1 - \mathrm{e}^{-2bt}\right). \tag{2.3.47}$$

In the case of $X_0 \sim \mathcal{N}(0, \sigma^2)$, the expectation of X_t is zero and the variance is given by

$$\mathrm{Var}(X_t) = \sigma^2 \, \mathrm{e}^{-2bt} + \frac{a^2}{2b}\left(1 - \mathrm{e}^{-2bt}\right). \tag{2.3.48}$$

Moreover, the covariance is

$$\mathrm{Cov}[X_s, X_t] = \sigma^2 \, \mathrm{e}^{-b(s+t)} + \frac{a^2}{2b}\left(\mathrm{e}^{-b|s-t|} - \mathrm{e}^{-b(s+t)}\right). \tag{2.3.49}$$

◇

Example 2.3.8 (Random Oscillator)**.** A *random harmonic oscillator* is modeled by the second-order SDE

$$\ddot{x} + a\dot{x} + bx = \sigma dB_t, \tag{2.3.50}$$

where $a, b > 0$ are constants, σ is a constant and B_t is a scalar Brownian motion. The term $a\dot{x}$ provides a damping force. This equation can be rewritten as a first-order SDE system:

$$\begin{aligned} \dot{x} &= y, \\ \dot{y} &= -bx - ay + \sigma dB_t. \end{aligned}$$

Let $A = \begin{bmatrix} 0 & 1 \\ -b & -a \end{bmatrix}$ and $K = \begin{bmatrix} 0 \\ \sigma \end{bmatrix}$. This system of equations can be rewritten in matrix form as

$$\dot{X}_t = AX_t + KdB_t. \tag{2.3.51}$$

Then the solution is

$$X_t = X_0 \, \mathrm{e}^{At} + \int_0^t \mathrm{e}^{A(t-s)} \, KdB_s, \tag{2.3.52}$$

where e^A denotes the matrix exponential of the matrix A. ◇

2.3.5 Numerical Methods

Stochastic differential equations can be solved analytically using Itô's formula. However, this is not always possible and a closed-form solution does not always exist. In such cases, numerical methods become essential for approximating solutions [28, 46]. The well-known numerical methods for simulating stochastic differential equations are the Euler-Maruyama and Milstein methods. The Euler-Maruyama method was introduced by Gisiro Maruyama in 1955 and is an extension of the Euler method, an approximation for solving ordinary differential equations introduced by Leonhard Euler [34].

Definition 2.3.8 (Euler–Maruyama method)**.** *The* Euler–Maruyama method *has two phases:*

- *Forward in time:*
  For a partition of $[0,T] : 0 = t_0 < t_1 < \ldots < t_n = T$ *of equal stepsize* Δt*, the solution of the SDE (2.3.21) is approximated by the iterations*

$$X_{i+1} = X_i + b(X_i)\Delta t + \sigma(X_i)(B_{i+1} - B_i), \tag{2.3.53}$$

  where $X_i = X_{t_i}$ *for* $i = 0, 1, \ldots, n-1$.

- *Backward in time:*
  For a partition of $[-T,0] : -T = t_0 < t_1 < \ldots < t_n = 0$ *of equal stepsize* Δt*, the solution of the SDE (2.3.21) is approximated by*

$$X_i = X_{i+1} - b(X_{i+1})\Delta t - \sigma(X_{i+1})(B_{i+1} - B_i), \tag{2.3.54}$$

  where $i = n-1, n-2, \ldots, 1$.

The Milstein method is in the first part similar to the Euler-Maruyama method. However, a so-called "correction term" of second-order is added (Milstein's correction term) in which the derivative of $\sigma(X)$ is used. By including this correction term, the Milstein method improves the accuracy of the approximation.

Definition 2.3.9 (Milstein method)**.** *The approximation of the solution of (2.3.21) in the* Milstein method *is defined as*

$$X_{i+1} = X_i + b(X_i)\Delta t + \sigma(X_i)\Delta B_i + \frac{1}{2}\sigma(X_i)\frac{\partial \sigma}{\partial x}(X_i)\left((\Delta B_i)^2 - \Delta t\right), \tag{2.3.55}$$

where $\Delta B_i = B_{i+1} - B_i$.

In the case of $\frac{\partial \sigma}{\partial x}(X_i) = 0$, the Milstein method amounts to the Euler-Maruyama method.

Example 2.3.9 (Simulation of geometric Brownian motion)**.** Consider the approximation of geometric Brownian motion in Ex. 2.3.4. Using the Euler-Maruyama method the iteration is

$$X_{i+1} = X_i + \mu X_i \Delta t + \alpha X_i \Delta B_i \tag{2.3.56}$$

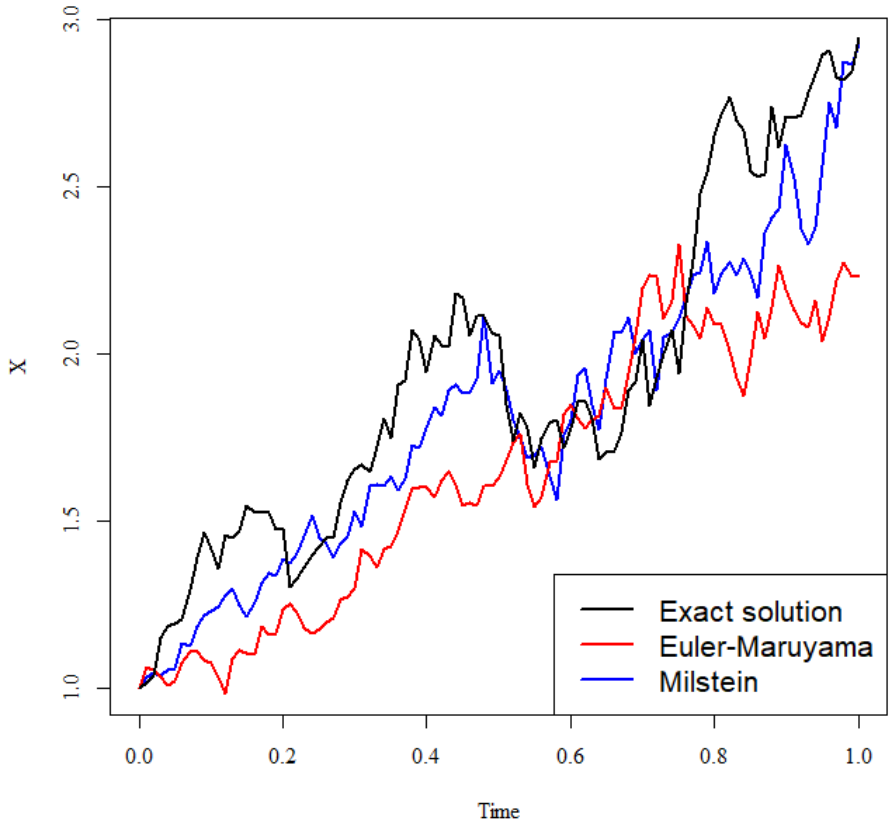


Figure 2.11: Exact solution of geometric Brownian motion and simulation with Euler-Maruyama and Milstein for $\mu = 0.8$, $\alpha = 0.4$, and $\Delta t = 0.01$.

and applying the Milstein method

$$X_{i+1} = X_i + \mu X_i \Delta t + \alpha X_i \Delta B_i + \frac{1}{2}\alpha^2 X_i \left((\Delta B_i)^2 - \Delta t\right). \tag{2.3.57}$$

Both simulations and the exact solution are plotted in Fig. 2.11 when $X_0 = 1$. Both simulations provide good approximations. However, the approximation of the Milstein method is slightly closer to the values of the exact solution. A numerical method for a scalar SDE can be applied using the function `snssde1d`.

```
> library(Sim.DiffProc)
> f <- expression(0.8*x)
> g <- expression(0.4*x)
# Generate numerical solutions
> euler <- snssde1d(N=100,t0=0,T=1,drift=f,diffusion=g,x0=1,
                    Dt=0.01,method="euler")
> milstein <- snssde1d(N=100,t0=0,T=1,drift=f,diffusion=g,x0=1,
```

```
                        Dt=0.01,method="milstein")
# Plot numerical solutions
> plot(euler, col="blue",lwd=2)
> lines(milstein, col="red",lwd=2)
# Plot exact solution
> lines(GBM(N=100,x0=1,t0=0,T=1,theta=0.8,sigma=0.4), col=1, lwd=2)
```

◇

Example 2.3.10. Reconsider the stochastic differential equation of the double well introduced in Sect. 2.3.2:

$$dX_t = \left(X_t - X_t^3\right) dt + kX_t dB_t. \tag{2.3.58}$$

A solution paths for this equation are depicted in Fig. 2.12 under two conditions: without noise ($k = 0$) and with noise ($k = 0.05$), simulated using the Euler-Maruyama method. The black line illustrates the solution paths for the case without noise, where only the deterministic part of the equation determines the behavior of the system. The blue line represents the solution paths for the case where noise is present. The noise leads to random fluctuations, causing the solution paths to deviate and follow a stochastic behavior.

◇

Convergence of simulation

Denote the simulation of a stochastic differential equation by $X_{\Delta t}(t)$ and the exact solution by $X(t)$. Weak convergence of the simulation is based on the error of the mean, while strong convergence is given by the mean of the error [23]. A simulation is *weakly convergent* if

$$\lim_{\Delta t \to 0} \left| \mathbb{E}\left[f(X_{\Delta t}(t))\right] - \mathbb{E}\left[f(X(t))\right] \right| = 0 \tag{2.3.59}$$

for every polynomial f [46]. A simulation is *strongly convergent* if

$$\lim_{\Delta t \to 0} \mathbb{E}\left[|X_{\Delta t}(t) - X(t)| \right] = 0. \tag{2.3.60}$$

Obviously, strong convergence implies weak convergence.

The simulation $X_{\Delta t}(t)$ *converges weakly with order* p if

$$\left| \mathbb{E}\left[f(X_{\Delta t}(t))\right] - \mathbb{E}\left[f(X(t))\right] \right| \in O\left((\Delta t)^p\right). \tag{2.3.61}$$

The simulation $X_{\Delta t}(t)$ *converges strongly with order* p if

$$\mathbb{E}\left[|X_{\Delta t}(t) - X(t)| \right] \in O\left((\Delta t)^p\right). \tag{2.3.62}$$

The Euler-Maruyama is weakly convergent with order 1 and strongly convergent with order 0.5. On the other hand, the Milstein method has an order of 1. It is easily

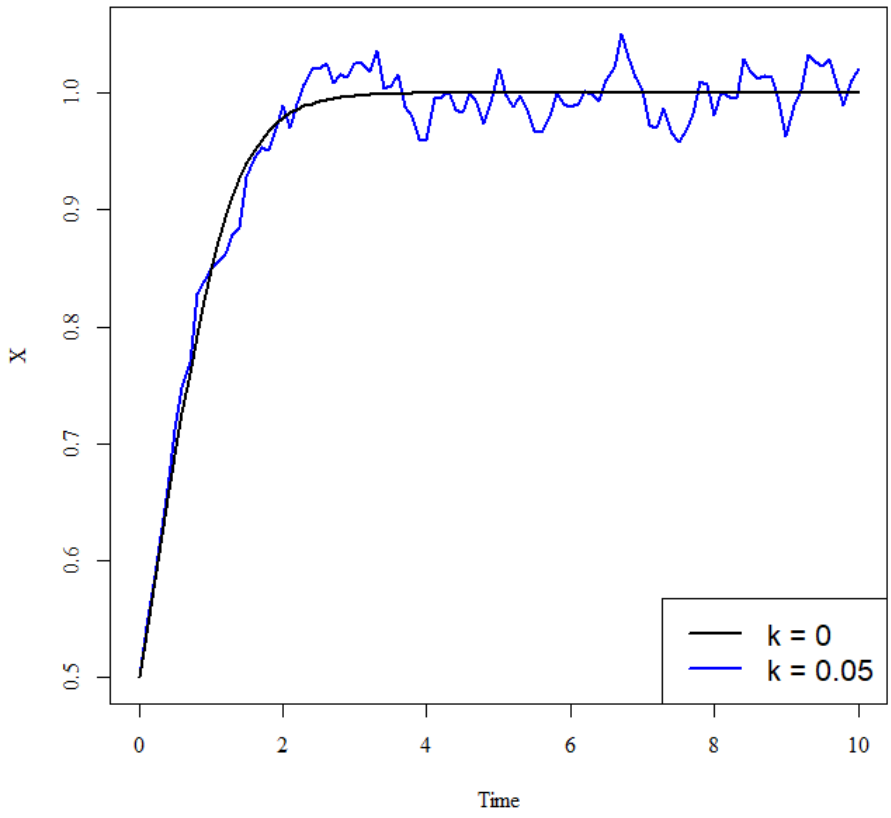


Figure 2.12: Solution path without noise and with noise.

applicable to scalar diffusion, but gets difficult in the multidimensional case.

A differential equation can be solved exactly or provided with a diffusion term to account for uncertain data (Fig. 2.12). In the latter case, the solution seems to yield a more realistic representation of the model behaviour. In the next chapter, similar to differential equations, the Bellman equation will be examined by adding a fluctuation term.

Chapter 3

Stochastic Differential Dynamic Programming

In decision problems within complex systems, noise is abundant. Nevertheless, its presence can be beneficial, as it enhances the robustness of systems against environmental uncertainties. Furthermore, noise represents diverse perspectives, reflecting the varied viewpoints of individuals involved in the decision-making process [27]. This chapter combines machine learning and stochastic dynamics by adding random fluctuations to the Bellman equation, called stochastic differential Bellman equation. It begins by attempting to solve this equation using the Itô formula, encountering challenges due to its nonlinearity. To find optimal policies, the Euler-Maruyama method is employed, providing an approximation of the solution. However, simulations grow exponentially, leading to the exploration of alternative approaches. An iterative algorithm similar to the value iteration is introduced, resulting in the stochastic value iteration algorithm. This algorithm incorporates Brownian motion and still converges in some cases, especially when the hyperparameters are sufficiently small.

Practical insights are drawn from the forest management problem, serving as a guiding example. Analysis of two noise models, additive and multiplicative noise, across various parameter combinations reveals distinct impacts. Statistical tests highlight significant differences between these noise models. The investigation into the moments of the stochastic differential Bellman equation is aimed at detecting how noise influences the system. Probability density functions are explored using the Fokker-Planck equation.

In reinforcement learning, adding noise to the system can help agents explore the environment more effectively, leading to discovering new strategies that might otherwise remain unexplored. Q-learning is expanded to integrate random fluctuations into the Q-function, resulting in the development of the stochastic Q-learning algorithm. Two case studies, the replacement problem and the inventory problem, are presented. Through stochastic value iteration and stochastic Q-learning algorithm, the behavior of noise models across diverse parameter settings is explored, exposing increased rewards. The chapter concludes with a brief exploration of random dynamical systems,

converting the stochastic differential Bellman equation to an ordinary differential equation with Brownian motion. This alternative perspective enriches the understanding of solving stochastic differential equations.

In this chapter, all theorems have been developed independently, and therefore, each is accompanied by a comprehensive proof.

3.1 Stochastic Differential Bellman Equation

Noisy fluctuations abound in dynamic systems. While the Bellman equation describes the behavior of a deterministic dynamic system, in which the system itself may be deterministic while the actions of agents are stochastic, this section introduces stochastic dynamics into the Bellman equation. This involves incorporating random or unpredictable changes into the model to better capture reality. A diffusion term is added to the Bellman equation to model these noisy fluctuations. This new approach is called *stochastic differential Bellman equation*. The development of stochastic differential dynamics from the Bellman equation to the stochastic differential Bellman equation is analogous to the development of stochastic differential equations from differential equations. Similar to differential equations, the introduction of noisy variation into the Bellman equation is achieved through a diffusion term. The most common forms of fluctuation terms correspond to additive and multiplicative noise.

The introduced stochastic differential Bellman equation integrates random fluctuations to enhance its ability to capture real-world dynamics and offers flexibility in handling noisy environments. In the following, the parameters remain the same as in the matrix notation for the Bellman equation in (2.2.9):

- A is the set of all actions,
- $P_a \in \mathbb{R}^{n\times n}$ is the probability matrix for action $a \in A$,
- $R_a \in \mathbb{R}^n$ is the reward function representing the immediate rewards for all states under action $a \in A$.

Additionally,

- $\sigma_a \in \mathbb{R}^+$ is the noise intensity of action $a \in A$,
- $B_t(\omega)$ is an n-dimensional Brownian motion.

Definition 3.1.1 (Stochastic differential Bellman equation with additive noise)**.** *Let* $0 \leq \gamma < 1$. *The* n-dimensional stochastic differential Bellman equation (SBE) with additive noise *is defined as*

$$dV_t = \max_{a\in A} \{R_a + \gamma P_a V_{t-1} dt + \gamma \sigma_a dB_t\}, \quad t > 0, V_0 \in \mathbb{R}^n. \tag{3.1.1}$$

Definition 3.1.2 (Stochastic differential Bellman equation with multiplicative noise)**.** *Let* $0 \leq \gamma < 1$. *The n-dimensional* stochastic differential Bellman equation with multiplicative noise *is defined as*

$$dV_t = \max_{a \in A} \{R_a + \gamma P_a V_{t-1} dt + \gamma \sigma_a V_{t-1} dB_t\}, \quad t > 0, V_0 \in \mathbb{R}^n. \tag{3.1.2}$$

The noise term differs between these models. Additive noise involves simply adding noise to the Bellman equation without changing the intensity depending on any parameters. On the other hand, multiplicative noise means that the noise term depends on the previous value function. If the previous value function is large, then the impact of noise will be large at that time.

The stochastic differential Bellman equation is a nonlinear SDE due to the presence of the maximum function, for which there is no general method for solving. It is not possible to apply Itô's formula to the stochastic differential Bellman equation. The stochastic differential Bellman equation has the form $\max_{a \in A} g(x, a)$. Recall the definition of the derivative of a function $f(x)$, that is

$$\frac{d}{dx} f(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}. \tag{3.1.3}$$

Let $f(x) = \max_{a \in A} g(x, a)$, where $f : \mathbb{R} \to \mathbb{R}$ and $g : \mathbb{R} \times A \to \mathbb{R}$, with a discrete set A. The function $g(x, a)$ is assumed to be differentiable with respect to x. Then

$$\begin{aligned}
\frac{d}{dx} f(x) &= \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \\
&= \lim_{h \to 0} \frac{\max_a \{g(x+h, a)\} - \max_a \{g(x, a)\}}{h} \\
&\neq \lim_{h \to 0} \frac{\max_a \{g(x+h, a) - g(x, a)\}}{h} \\
&= \max_a \lim_{h \to 0} \frac{g(x+h, a) - g(x, a)}{h} \\
&= \max_a \frac{\partial}{\partial x} g(x, a).
\end{aligned}$$

Thus one cannot swap the order of $\max_a$ and $\lim_{h \to 0}$. However, the maximum function can be omitted if the policy is given in advance. In the following, the stochastic differential Bellman equation with given policy is considered in the scalar case. In particular, the transition probability matrix is $P_\pi = (1)$. These discussions then lead to specific solutions of the stochastic differential Bellman equation.

Example 3.1.1 (Scalar MDP subject to noise)**.** Consider a cleaning robot that has the function of vacuuming and floor mopping. Assuming it has reached a cleaning location, where there is only one state representing the current cleaning location, the robot is tasked with deciding whether to vacuum or mop the floor. The available actions are denoted by a_1 and a_2 respectively. The transition probability is one for both actions, but the rewards are different. In this case, vacuuming the floor has a -1 penalty

and mopping the floor has a reward of 2. Moreover, environmental disturbances like sudden changes in lighting, unexpected obstacles on the floor, or uneven surfaces could introduce noise into the actions of the cleaning robot. Consider this noise by $\sigma_1 = 0.1$ for action a_1 and by $\sigma_2 = 0.05$ for action a_2. To address this problem, a concept is needed that accounts for the uncertainty introduced by the noise and guides the robot in making decisions that maximize its cumulative reward over time.

$\diamond$

Theorem 3.1.1. *The scalar stochastic differential Bellman equation driven by additive noise with given policy π and $P_\pi = (1)$*

$$dV_t = (R_\pi + \gamma V_{t-1})\, dt + \gamma\sigma_\pi dB_t, \quad V_0 \in \mathbb{R}, t > 0 \tag{3.1.4}$$

has the solution

$$V_t = \mathrm{e}^{\gamma t}\, V_0 + \frac{\mathrm{e}^{\gamma t} - 1}{\gamma} R_\pi + \gamma\sigma_\pi\, \mathrm{e}^{\gamma t} \int_0^t \mathrm{e}^{-\gamma s}\, dB_s. \tag{3.1.5}$$

Proof. For a general linear stochastic differential equation

$$dX_t = [a_1(t)X_t + a_2(t)]\, dt + [b_1(t)X_t + b_2(t)]\, dB_t, \quad X_{t_0} \text{ given }, \tag{3.1.6}$$

where a_1, a_2, b_1 and b_2 are deterministic functions and B_t is a scalar Brownian motion, the solution given in (2.3.41) is

$$X_t = \Phi_{t,t_0} \left\{ X_{t_0} + \int_{t_0}^t \left[a_2(s) - b_1(s)b_2(s)\right] \Phi_{s,t_0}^{-1} ds + \int_{t_0}^t b_2(s)\Phi_{s,t_0}^{-1} dB_s \right\}. \tag{3.1.7}$$

The SBE with additive noise has the setting $a_1 = \gamma$, $a_2(t) = R_\pi$, $b_1 = 0$, and $b_2 = \gamma\sigma_\pi$. Thus the solution of (3.1.4) is

$$\begin{aligned} V_t &= e^{\gamma t}V_0 + e^{\gamma t} \int_0^t e^{-\gamma s} R_\pi ds + e^{\gamma t} \int_0^t e^{-\gamma s} \gamma\sigma_\pi dB_s \\ &= \mathrm{e}^{\gamma t}\, V_0 + \frac{\mathrm{e}^{\gamma t} - 1}{\gamma} R_\pi + \gamma\sigma_\pi\, \mathrm{e}^{\gamma t} \int_0^t \mathrm{e}^{-\gamma s}\, dB_s. \end{aligned}$$

$\square$

Theorem 3.1.2. *The scalar stochastic differential Bellman equation driven by multiplicative noise with given policy π and $P_\pi = (1)$*

$$dV_t = (R_\pi + \gamma V_{t-1})\, dt + \gamma\sigma_\pi V_{t-1} dB_t, \quad V_0 \in \mathbb{R}, t > 0 \tag{3.1.8}$$

has the solution

$$V_t = \mathrm{e}^{\left(\gamma - \frac{1}{2}\gamma^2\sigma_\pi^2\right)t + \gamma\sigma_\pi B_t} \left[V_0 + R_\pi \int_0^t \mathrm{e}^{\left(\frac{1}{2}\gamma^2\sigma_\pi^2 - \gamma\right)s - \gamma\sigma_\pi B_s}\, ds \right]. \tag{3.1.9}$$

Proof. The proof is analogous to the solution of SBE with additive noise. Using the solution (2.3.41) with $a_1 = \gamma$, $a_2 = R_\pi, b_1 = \gamma\sigma_\pi$ and $b_2 = 0$ the fundamental solution yields

$$\Phi_{t,t_0} = \exp\left[\int_{t_0}^{t}\left(\gamma - \frac{1}{2}\gamma^2\sigma_\pi^2\right)ds + \int_{t_0}^{t}\gamma\sigma_\pi dB_s\right] \tag{3.1.10}$$

and the solution of (3.1.8) is

$$V_t = \Phi_{t,t_0}\left[V_0 + R_\pi\int_0^t \Phi_{s,t_0}^{-1}ds\right]. \tag{3.1.11}$$

□

The expectation of the stochastic differential Bellman equation with given policy π for $V_0 \in \mathbb{R}$ is

$$\mathbb{E}[V_t] = e^{\gamma t}V_0 + \frac{e^{\gamma t}-1}{\gamma}R_\pi, \tag{3.1.12}$$

due to the zero mean property (Prop. 2.3.3). Note that the expectation does not depend on additive and multiplicative noise.

The stochastic differential Bellman equation cannot be solved exactly, but the solution can be approximated by known numerical methods.

3.2 Simulations of Stochastic Differential Bellman Equation

The stochastic differential Bellman equation enables capturing uncertainty in complex systems. However, it is a continuous model, which can pose challenges for practical implementation. The contribution of this section lies in the application of the Euler-Maruyama method to handle this continuous stochastic differential Bellman equation. This discretization procedure facilitates the conversion of the continuous model into a discrete format, enhancing its numerical tractability. Through this discretization, decision problems under uncertainty can be analyzed efficiently. Moreover, it enables to derive solutions to these problems in a computationally feasible manner.

Two different numerical approximations are studied. The first approach employs the Euler-Maruyama method in its traditional form, while the second approach utilizes a modified variant of this simulation. Furthermore, the aim is to provide insights into their respective strengths, weaknesses, and applicability in solving decision problems under uncertainty described by the stochastic differential Bellman equation.

3.2.1 Simulation via Euler-Maruyama Method

In the following, a numerical solution to the stochastic differential Bellman equation is obtained using the Euler–Maruyama method. This discretized equation serves as the foundation for the subsequent algorithmic approach. It is substituted in the value iteration algorithm, which is called the stochastic value iteration algorithm. The stochastic value iteration algorithm represents a novel adaptation of the traditional value iteration algorithm to accommodate uncertainty in Markov decision processes. By incorporating the numerical solution obtained through the Euler–Maruyama method, the algorithm iteratively refines value function estimates to converge towards optimal policies in the presence of noise.

However, it has been observed that the value function in the stochastic value iteration algorithm utilizing the Euler–Maruyama simulation often exhibits undesirable behavior. The value function tends to grow rapidly and exhibits erratic fluctuations, making it challenging to establish a reliable termination criterion for the algorithm. Without a clear stopping condition, the algorithm may continue indefinitely. To address this issue, the span semi-norm plays a crucial role. It ensures that the value function maintains smoothness and prevents abrupt jumps, making fluctuations manageable. This enables faster convergence of the algorithm. Thus, it is considered to enhance the performance of the stochastic value iteration algorithm.

Theorem 3.2.1. *For each partition of the interval $[0,T]$ into n equal subintervals of length $\Delta t > 0$, the Euler–Maruyama simulation of the stochastic differential Bellman equation with additive noise is given as*

$$V_{i+1} = V_i + \max_{a\in A}\left\{(R_a + \gamma P_a V_i)\,\Delta t + \gamma\sigma_a\,(B_{i+1} - B_i)\right\}, \quad 0 \le i \le n-1, \tag{3.2.1}$$

and with multiplicative noise as

$$V_{i+1} = V_i + \max_{a\in A}\left\{(R_a + \gamma P_a V_i)\,\Delta t + \gamma\sigma_a V_i(B_{i+1} - B_i)\right\}, \quad 0 \le i \le n-1 \tag{3.2.2}$$

where $V_i = V_{t_i}$.

Proof. The stochastic differential Bellman equation has the form

$$dV_t = \max_a \left\{b(a,t,V_t)dt + \sigma(a,t,V_t)dB_t\right\}. \tag{3.2.3}$$

Applying the Euler-Maruyama method (2.3.8) yields

$$V_{i+1} = V_i + \max_a \left\{b(a,t_i,V_i)\Delta t + \sigma(a,t_i,V_i)\Delta B_i\right\}. \tag{3.2.4}$$

where $\Delta B_i = B_{i+1} - B_i$ and $0 \le i \le n-1$. □

Example 3.2.1 (Euler-Maruyama simulation of the SBE)**.** Reconsider the forest management problem in Ex. 2.2.5. Assume that the system is subject to noise with $\sigma_1 = 0.5$ and $\sigma_2 = 0.1$. Initialize $V_0 = (0,0,0)^{\mathrm{T}}$. The simulations for the time interval $[0,1]$ are shown with a stepsize of $\Delta t = 0.01$ in Fig. 3.1. In the case of multiplicative noise, the value function sharply increases around $t = 0.3$. In view of additive noise, the value function increases less sharply compared to the multiplicative noise, but more keenly when compared to the deterministic case. ◇

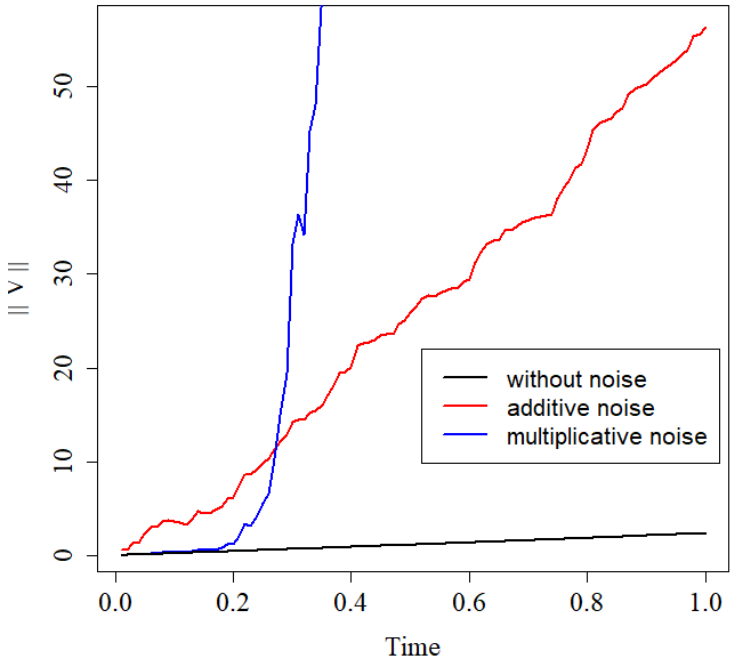


Figure 3.1: Euler-Maruyama simulations of SBE with additive and multiplicative noise, as well as without any noise.

So far, the stochastic differential Bellman equation is discretized and its solution is approximated using the Euler-Maruyama method. The next step is to leverage this simulation to determine an optimal policy for a Markov decision process subject to noise. However, in the context of stochastic environments, the direct application of the value iteration algorithm becomes challenging due to the inherent uncertainty. To address this, a new approach called the *stochastic value iteration algorithm* will be introduced. Similar to the value iteration algorithm, the simulation will be iterated to refine the value function estimates. The initial state V_0 is just a random value function, while the noise intensity σ has to be specified beforehand. Then the optimal policy in matrix notation is given by

$$\pi^* = \arg\max_{a \in A} \{V^* + (R_a + P_a V^*)\Delta t + \gamma\sigma(t_i, V*)\Delta B_i\} \tag{3.2.5}$$

where $\Delta B_i = (B_{i+1} - B_i)$. Note that the diffusion term $\sigma(t_i, V_i)(B_{i+1} - B_i)$ is not bounded. The increments of Brownian motion can be simulated using a random variable which is normally distributed, but the value of this random variable is not bounded. Only the probability density function of a normal distribution has a maximum value, not the random variable itself. To consider the impact of the unbounded nature of the diffusion term, the equation is iterated until the difference between successive value functions is small enough, or for a certain number of iterations. The steps of the stochastic value iteration algorithm with additive noise are described in Alg. 3 and an

implementation in R can be found in the appendix in simplified form without error handling.

Example 3.2.2 (Stochastic value iteration with Euler-Maruyama)**.** The stochastic value iteration algorithm using the Euler-Maruyama simulation is considered in the following. It is applied to the previous example with $\epsilon = 0.1$ and $\Delta t = 0.01$.

```
# (...) Define prob. and reward matrices, noise, Brownian motion
# Generate numerical solution for additive noise
> EulersimSBE(P,R,0.9,c(0,0,0),0.1,0,sigma,dis,0.01,1)



$`Iterations`
[1] 1000
$V
[1] 163795.6 163815.8 163825.8
$`Optimal Policy`
[1] 1 1 1
# Generate numerical solution for multiplicative noise
> EulersimSBE(P,R,0.9,c(0,0,0),0.1,0,sigma,dis,0.01,2)
$`Iterations`
[1] 7
$V
[1] 0.001330079 0.094497518 0.217939552
$`Optimal Policy`
[1] 1 2 2
```

The algorithm with additive noise stops after 1000 steps, yielding a large value function. In this case, the optimal policy is $\pi^* = (1, 1, 1)$, indicating that the agent only receives a reward of one in the last state. Conversely, with multiplicative noise, the algorithm terminates after seven steps, resulting in a small value function. The optimal policy is $\pi^* = (1, 2, 2)$, which means ("wait", "cut", "cut"), allowing the agent to acquire more rewards, akin to the deterministic case.

In situations involving additive noise, the value function tends to be large. To improve the convergence, the variant of value iteration algorithm utilizing the span semi-norm will be employed within the stochastic value iteration algorithm using the Euler-Maruyama simulation.

◇

Algorithm 3: Stochastic value iteration with Euler-Maruyama

Input: V_0, $\epsilon > 0$, $\gamma \in [0, 1)$, $\sigma_a > 0$ for all $a \in A$ and $\Delta t > 0$.

$i = 0$
do
  for $s \in S$ **do**
    $V_{i+1}(s) = V_i(s) + \max_{a \in A}\{[R(s,a) + \sum_{s' \in S} \gamma P(s' \mid s, a) V_i(s')]\Delta t$
      $+\gamma \sigma_a (B_{i+1} - B_i)\}$
  end
  $\theta = \|V_{i+1} - V_i\|$
  $i = i + 1$
while $\theta \geq \epsilon \frac{1-\gamma}{2\gamma}$ *or* $i \leq 1000$
for $s \in S$ **do**
  $\pi^*(s) = \arg\max_{a \in A}\{V_i(s) + [R(s,a) + \sum_{s' \in S} \gamma P(s' \mid s, a) V_i(s')]\Delta t$
    $+\gamma \sigma_a (B_{i+1} - B_i)\}$
end

3.2.2 Variant of Stochastic Value Iteration

The stochastic value iteration algorithm (Alg. 3) will be changed such that the convergence criterion is

$$\mathrm{sp}(V_{i+1} - V_i) < \theta \quad \text{or} \quad i \leq 1000,$$

with

$$\mathrm{sp}(V) = \max_{s \in S} V(s) - \min_{s \in S} V(s). \tag{3.2.6}$$

For a stepsize of $\Delta t = 0.1$, the algorithm with additive noise terminates after 423 steps with the optimal policy $\pi^* = (1, 1, 1)$. Higher noise values seem to be beneficial in terms of the number of iterations, as demonstrated in Table 3.1. For the stepsize $\Delta t = 0.01$, the algorithm with additive noise takes only one step, whereas the algorithm with multiplicative noise terminates after seven iterations with $V^* = (0.001, 0.094, 0.217)^{\mathrm{T}}$ and optimal policy $\pi^* = (1, 2, 2)$.

Noise value	Additive noise	Multiplicative noise
$\sigma_1 = \sigma_2 = 0$	441	441
$\sigma_1 = \sigma_2 = 0.2$	434	351
$\sigma_1 = \sigma_2 = 0.5$	427	282
$\sigma_1 = \sigma_2 = 0.7$	426	204
$\sigma_1 = \sigma_2 = 0.9$	421	153

Table 3.1: Number of iterations for the variant of stochastic value iteration with $\Delta t = 0.1$.

This variant makes it possible to improve the performance of the stochastic value iteration with Euler-Maruyama. Compared to the deterministic method, this algorithm

requires significantly fewer steps to terminate, regardless of which noise model is used. In the following, another simulation method will be studied to explore further benefits of noise.

3.2.3 Modified Simulation

In the preceding simulation employing the Euler-Maruyama method, excessive fluctuations and significant growth in the value function was observed. The equation presented in this section represents a discrete approximation derived from the stochastic differential Bellman equation, wherein the addition of the previous value is omitted. The diffusion term is approximated using the Euler-Maruyama method and subsequently substituted into the stochastic differential Bellman equation. Furthermore, the equation is discretized by dividing dt into small time steps Δt. In the case without noise, where $\sigma = 0$, and $\Delta t = 1$, this approximation simplifies to a form resembling the classical Bellman equation, and the stochastic value iteration reduces to the value iteration algorithm.

Theorem 3.2.2. *For each partition of the interval $[0, T]$ into n equal subintervals of length $\Delta t > 0$, the modified simulation of the stochastic differential Bellman equation with additive noise is given by*

$$V_{i+1} = \max_{a \in A} \left\{ (R_a + \gamma P_a V_i)\Delta t + \gamma \sigma_a \left(B_{i+1} - B_i \right) \right\}, \quad 0 \leq i \leq n-1 \tag{3.2.7}$$

and with multiplicative noise by

$$V_{i+1} = \max_{a \in A} \left\{ \left(R_a + \gamma P_a V_i\right) \Delta t + \gamma \sigma_a V_i (B_{i+1} - B_i) \right\}, \quad 0 \leq i \leq n-1. \tag{3.2.8}$$

Proof. The stochastic differential Bellman equation has the following form

$$dV_t = \max_{a} \left\{ b(a, t, V_t)dt + \sigma(a, t, V_t)dB_t \right\}, \quad V_0 \in \mathbb{R}^n, t > 0. \tag{3.2.9}$$

For simulating the equation, the following approximations are employed:

$$\int_{t_i}^{t_{i+1}} b(a, s, V_s)ds \approx b(a, t_i, V_i)\Delta t \quad \text{and} \quad \int_{t_i}^{t_{i+1}} \sigma(a, s, V_s)dB_s \approx \sigma(a, t_i, V_i)\Delta B_i.$$

Thus, the discrete approximation becomes

$$V_{i+1} = \max_{a \in A} \left\{ \left(R_a + \gamma P_a V_i\right) \Delta t + \gamma \sigma(a, t_i, V_i) \left(B_{i+1} - B_i \right) \right\}, \quad 0 \leq i \leq n-1, \tag{3.2.10}$$

where $\sigma(a, t_i, V_i)$ represents either additive noise σ_a or multiplicative noise $\sigma_a V_i$.

□

Example 3.2.3 (Modified simulation of SBE)**.** Reconsider Ex. 3.2.1 with $\sigma_1 = 0.5$, $\sigma_2 = 0.1$, and $V_0 = (0, 0, 0)^{\mathrm{T}}$. The modified simulation for the stepsize $\Delta t = 0.01$ in the time interval $[0, 1]$ is illustrated in Fig. 3.2. In this example, the oscillations in multiplicative noise are smaller compared to those in additive noise. However, both noise models result in a lower value function compared to that in the previous simulation.

◇

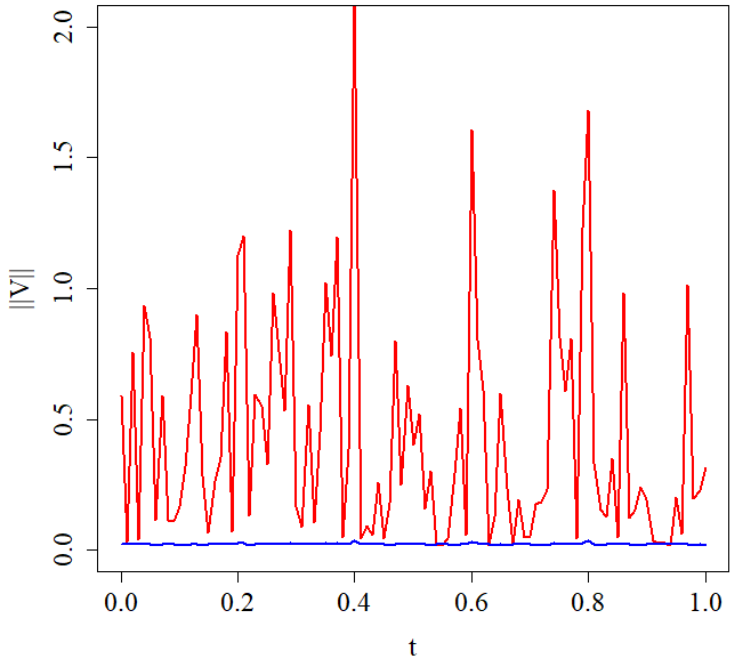


Figure 3.2: Modified simulation of SBE with additive noise (red line) and multiplicative noise (blue line).

The modified simulation is iterated to determine the optimal value function and the optimal policy of a Markov decision process with noise. In practice, the convergence of the stochastic value iteration is influenced by various factors, including the choice of stepsize Δt. The modified simulation is iterated until the difference between successive value functions becomes sufficiently small or until a predetermined number of iterations is reached.

Example 3.2.4 (Stochastic value iteration with modified simulation)**.** For a comparison, apply the modified simulation to Ex. 3.2.1 with $\epsilon = 0.1$ and $\Delta t = 0.01$ as in Ex. 3.2.2.

```
# (...) Define prob. and reward matrices, noise, Brownian motion
# Generate numerical solution for SBE with additive noise
> SBEalgo(P,R,0.9,c(0,0,0),0.1,0,sigma,dis,0.01,1)
$`Iterations`
[1] 71
$V
[1] -0.04042095 -0.03042095 -0.02042095
$`Optimal Policy`
[1] 2 2 2
# Generate numerical solution for SBE with mult. noise
```

```
> SBEalgo(P,R,0.9,c(0,0,0),0.1,0,sigma,dis,0.01,2)
$`Iterations`
[1] 2
$V
[1] 0.000054000 0.009699275 0.019398550
$`Optimal Policy`
[1] 1 2 2
```

The stochastic value iteration algorithm with additive noise terminates after 71 steps, yielding the optimal policy $\pi^* = (2, 2, 2)$. In contrast, the previous simulation did not terminate in the case of additive noise. The algorithm with multiplicative noise converges in just two iterations, returning the optimal policy $\pi^* = (1, 2, 2)$, identical to the deterministic case. The smaller oscillations in multiplicative noise compared to those in additive noise may explain the faster convergence.

◇

3.2.4 Influence of Parameters

The parameters affecting the results of stochastic value iteration using the modified Euler-Maruyama simulation are analyzed. The choice of the stepsize Δt is important when solving the stochastic differential Bellman equation numerically. This is just one aspect of many when evaluating and analyzing the stochastic value iteration.

- Discount factor γ: The discount factor should be chosen based on the decision problem. If immediate profit is important, γ should be as small as possible. If the gain in the future is of interest, then γ should be close to one. With a higher discount factor, the modified simulation of the stochastic differential Bellman equation needs more iterations to terminate (Sect. 3.2.5). This is also related to the fact that the influence of noise is greater for a larger discount factor.

- Stepsize Δt: With a small stepsize, the computation in the simulations is more efficient and less error-prone. This can be explained by considering the Euler method without noise and writing the first value function as

$$V_1 = V_0 + \Delta t f(t_0, V_0). \tag{3.2.11}$$

Using Taylor expansion of V around t_0, it follows that

$$V(t_0 + \Delta t) = V(t_0) + \Delta t V'(t_0) + \frac{1}{2}\Delta t^2 V''(t_0) + O(\Delta t^3), \tag{3.2.12}$$

where $V'(t_0) = f(t_0, V_0)$. The error in each computational step, called *local truncation error* (LTE), is calculated by subtracting the first equation from the second

$$\text{LTE} = V(t_0 + \Delta t) - V_1 = \frac{1}{2}\Delta t^2 V''(t_0) + O(\Delta t^3). \tag{3.2.13}$$

Thus for small stepsize Δt, the error in an iteration step is approximately proportional to Δt^2.

- Noise intensity σ: In some cases, the introduction of noise can yield faster convergence towards the optimal policy compared to simulations conducted without noise. The impact of σ on convergence varies. In some cases, higher noise may accelerate convergence, but in general the simulation requires more iterations to terminate with larger σ-values than with smaller values. In the context of the multiplicative noise model, the optimal policy remains largely unaffected by changes in σ. It tends to remain consistent across different σ values. But in the additive noise model, different σ-values can change the optimal policy. This is reflected in the practical results worked out in Sect. 3.2.5 as well as in the theoretical results in Sect. 3.3.

The last point not listed is Brownian motion. However, it is possible to get different results using the same parameters due to the diffusion term, which does not follow the same path each time. This is a stochastic process such that each realization in the simulation can be different, and particularly the number of iterations can increase or decrease.

3.2.5 Comparison of Simulations in Practice

In this section, the practical aspects of the stochastic value iteration algorithm are explored, with a focus on its application to the forest management problem. Both simulations, utilizing Euler-Maruyama and the modified equation, are examined, and a comparative analysis between the two noise models is conducted.

Comparison for the modified simulation

In the following, the behaviour of the stochastic value iteration algorithm using the modified simulation with both additive and multiplicative noise is studied under the influence of the parameters. For this, the stochastic value iteration algorithm was executed under the same conditions for both noise models. In the subsequent figures, the results of the additive noise algorithm are depicted by the red line, while the blue line represents the multiplicative noise algorithm and the green line corresponds to the deterministic case. To ensure the simulations are less error-prone, a stepsize of $\Delta t = 0.1$ is chosen. The initial value function is set to $V_0 = (1, 1, 1)^{\mathrm{T}}$, and the termination threshold is set to $\epsilon = 0.1$.

First, the value functions are analyzed by setting $\gamma = 0.8$, $\sigma_1 = 0.2$, and $\sigma_2 = 0.7$. The norm of the value function at each iteration i is shown in Fig. 3.3. The lines remain constant from the iteration onwards where the algorithm converges. The multiplicative noise algorithm approximates the deterministic values well. However, the additive noise introduces significant fluctuations in the values. Similar behavior is observed for other σ values as well.

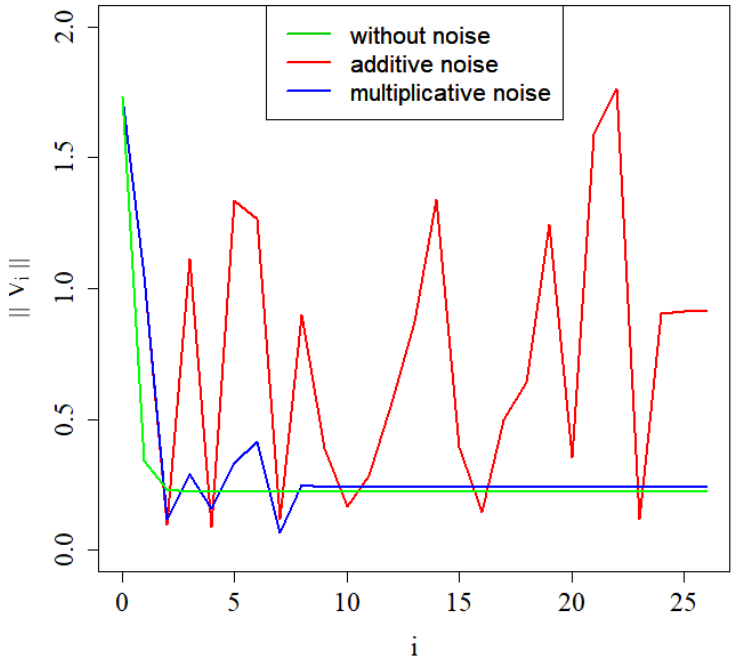


Figure 3.3: Plot of the value function for i-th iteration.

Secondly, the algorithms were performed for $\gamma \in \{0.1, 0.2, \ldots, 0.9\}$ and different σ-values. For each noise model, the algorithms were run 50 times. It was observed that the optimal policy remains almost always the same in the case of multiplicative noise, while for the additive noise algorithm, it varies depending on the parameters. For example, when $\gamma = 0.9$, the optimal policy in the additive noise model is predominantly $\pi^* = (2, 2, 2)$, whereas for $\gamma = 0.5$, the optimal policy changes due to σ to $\pi^* = (1, 1, 1)$ and for $\gamma = 0.1$ to $\pi^* = (1, 2, 2)$. This means that for a short-term gain, the optimal decision under additive noise in the forest management problem involves ("wait","cut","cut"). Long-term gain yields the decision to cut in each state, which can be interpreted such that waiting is too risky, although the reward is then zero. In the case of multiplicative noise, the optimal policy consistently remains $\pi^* = (1, 2, 2)$ independent of the discount factor. This shows that the algorithm is more robust and short-term or long-term gain does not impact the decision-making process.

The optimal value functions for different discount factors γ with $\sigma_1 = \sigma_2 = 0.1$ are depicted in Fig. 3.4. In the deterministic case (without noise), the value function remains almost constant. The case of multiplicative noise is slightly fluctuating with increasing norm up to $\gamma = 0.7$. The additive noise does not follow such a pattern. The discount factor has no significant impact on the results of the algorithm with multiplicative noise when compared to the deterministic case. This highlights the robustness of the algorithm subject to multiplicative noise, as it maintains stability across different

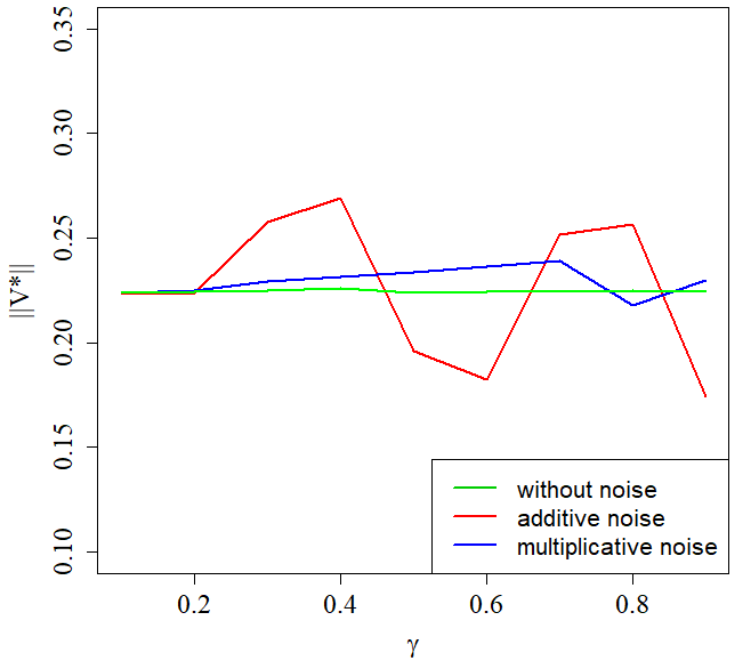


Figure 3.4: Plot of the value functions for different discount factors.

discount factors.

Finally, the efficiency of the algorithms is compared. The practical results indicate that, in some cases, the evaluation of the stochastic differential Bellman equation driven by multiplicative noise provides better results in terms of efficiency compared to the equation with additive noise. However, for some parameters the multiplicative noise algorithm fails to terminate while the additive noise algorithm does. This occurs when large stepsizes and large discount factors are simultaneously chosen. In the following, the number of iterations of the simulation with additive and multiplicative noise, considering $\gamma = 0.9$, are examined. In the additive noise model, the average number of iterations is about 130 with 28% reaching the same optimal policy as the deterministic case. In the multiplicative noise model, the average number of iterations lies at 47 and each time the same optimal policy is achieved (100%). The simulations with multiplicative noise tend to be faster than those with additive noise. Note that exceptions can occur due to the stochastic nature of Brownian motion. In the additive noise model, when $\sigma_1 = \sigma_2$, the policy remains the same as in the deterministic model since noise is simply added to the deterministic part. The algorithm terminates after about 94 steps. As observed, the algorithm with additive noise stops even when $\sigma_1, \sigma_2 > 1$, whereas the multiplicative noise algorithm does not.

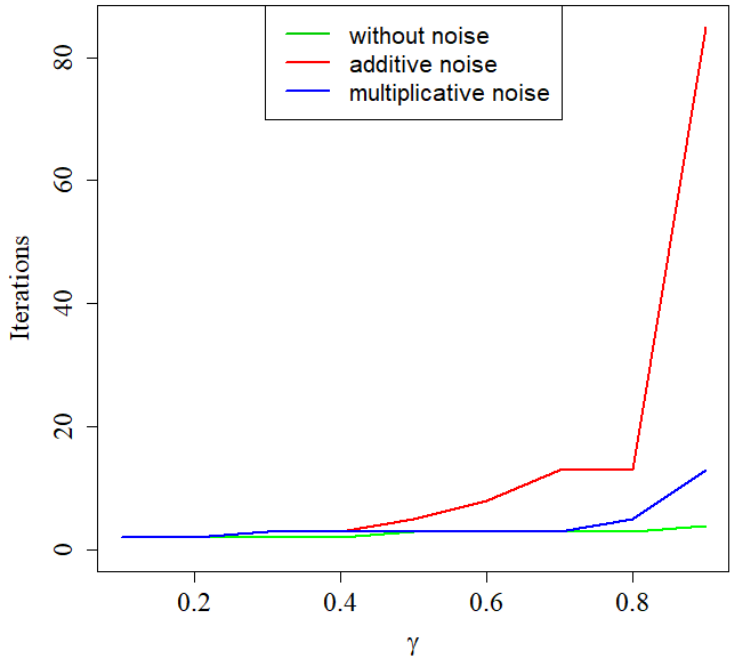


Figure 3.5: Number of iterations for different discount factors.

The results regarding the average number of iterations and the minimum number of iterations for different stepsizes are presented in Table 3.2. The noise intensities at which the algorithm terminates fastest are specified. This table shows that the multiplicative noise algorithm converges faster than in the additive case independent of the stepsize up to one. Moreover, the optimal value function V^* in the multiplicative noise model is almost the same as in the deterministic case as long as the stepsize is less than one. For stepsize $\Delta t = 1$, the algorithm with multiplicative noise does not terminate with $\gamma = 0.9$, regardless of the choice of σ. This is due to the local truncation error shown in (3.2.13). The local error increases with higher stepsize. Nonetheless, the convergence is also influenced by the discount factor. With $\gamma \leq 0.7$, the algorithm with multiplicative noise also terminates for the stepsize $\Delta t = 1$. If the discount factor is set to $\gamma = 0.1$, the algorithm stops after two iterations in both noise models. A larger discount factor causes the algorithms to take more time to terminate, which is shown for $\sigma_1 = \sigma_2 = 0.1$ in Fig. 3.5. This distinction becomes more evident for $\gamma \geq 0.5$.

The final decision regarding the noise model is determined by the specific application. However, investigating the effects of additive and multiplicative noise can still provide valuable insights. For example, exploring the practical implications of additive and multiplicative noise in a real-world application can help validate theoretical considerations and quantify the actual performance differences between the approaches. Additionally, such analysis can aid in identifying potential weaknesses or uncertainties

Δt	Without noise	SBE additive	SBE multiplicative
1	$V^* = \begin{bmatrix} 3.48 \\ 4.12 \\ 5.12 \end{bmatrix}$ # iterations 44	$V^* = \begin{bmatrix} 3.15 \\ 3.80 \\ 4.80 \end{bmatrix}$ $\pi^* = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$ # iterations 17 $\sigma_1 = \sigma_2 = 0.1$ average #292	-
0.1	$V^* = \begin{bmatrix} 0.005 \\ 0.100 \\ 0.200 \end{bmatrix}$ # iterations 4	$V^* = \begin{bmatrix} -0.002 \\ 0.097 \\ 0.197 \end{bmatrix}$ $\pi^* = \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix}$ # iterations 5 $\sigma_1 = 0.5$, $\sigma_2 = 0.1$ average #129	$V^* = \begin{bmatrix} 0.005 \\ 0.099 \\ 0.198 \end{bmatrix}$ $\pi^* = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$ # iterations 3 $\sigma_1 = 0.5$, $\sigma_2 = 0.1$ average #46
0.01	$V^* = \begin{bmatrix} 0.00005 \\ 0.01000 \\ 0.02000 \end{bmatrix}$ # iterations 3	$V^* = \begin{bmatrix} 0.627 \\ 0.627 \\ 0.637 \end{bmatrix}$ $\pi^* = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ # iterations 2 $\sigma_1 = 0.9$, $\sigma_2 = 0.1$ average #127	$V^* = \begin{bmatrix} 0.0001 \\ 0.0107 \\ 0.0214 \end{bmatrix}$ $\pi^* = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$ # iterations 3 $\sigma_1 = \sigma_2 = 0.1$ average #7
0.0009	$V^* = \begin{bmatrix} 0.000001 \\ 0.000900 \\ 0.001800 \end{bmatrix}$ # iterations 2	$V^* = \begin{bmatrix} 0.6219 \\ 0.6228 \\ 0.6237 \end{bmatrix}$ $\pi^* = \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix}$ # iterations 2 $\sigma_1 = 0.1$, $\sigma_2 = 0.9$ average #100	$V^* = \begin{bmatrix} -0.00003 \\ 0.00080 \\ 0.00163 \end{bmatrix}$ $\pi^* = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$ # iterations 2 $\sigma_1 = \sigma_2 = 0.1$ average #4

Table 3.2: Comparison for different stepsizes, $\gamma = 0.9$, and $\epsilon = 0.1$.

in the model, offering insights into the robustness of the results. Furthermore, it allows for an expansion of understanding on how these different noise modeling approaches operate and impact the outcomes.

Comparison of the Euler-Maruyama simulations

The stochastic value iteration algorithm employing Euler-Maruyama simulation is executed with parameters $\Delta t = 0.1$ and $\gamma = 0.9$, utilizing different noise intensities. For the threshold $\epsilon = 0.1$, both noise models do not terminate, since the fluctuations caused by noise are larger than the allowed tolerance. Therefore, it may be necessary to choose a larger threshold value. When $\epsilon = 1.8$, the algorithm incorporating multiplicative noise terminates in average in 10 steps. At first glance, this seems more efficient than the previous simulation, but this difference is based on the large threshold ϵ. The optimal policy varies between $(1, 1, 1)$ and $(2, 2, 2)$, which means lower rewards are gained than in the previous simulations. For $\sigma_1 = \sigma_2$, the optimal policy $\pi^* = (1, 2, 2)$ is obtained predominantly. Conversely, the algorithm utilizing additive noise fails to converge altogether. Only for smaller stepsizes less than or equal to 0.01, both algorithms terminate in two or three steps.

In contrast to the modified simulation, a larger discount factor leads to faster convergence but not in general. For example, in view of the multiplicative noise algorithm, the following results are obtained when using $\sigma_1 = 0.9$ and $\sigma_2 = 0.5$:

- $\gamma = 0.9$: number of iterations is 8
- $\gamma = 0.5$: number of iterations is 6
- $\gamma = 0.1$: number of iterations is 22,

with optimal policy $\pi^* = (2, 2, 2)$ for all three cases.

The value function in the case of multiplicative noise follows a path like Brownian motion. It can increase and decrease suddenly because the diffusion term can be larger due to the previous value function and if the increment of Brownian motion results in a negative number, then it is subtracted from the drift term. After the same number of iterations, the values in the simulation with additive noise are larger than the values in the simulation without noise. A general order cannot be given, but the values in the simulation with multiplicative noise are often smaller than the values with additive noise. Thus the observation can be stated as $V_{\text{det}} \leq V_{\text{mult}} \leq V_{\text{add}}$. When this order occurs and when it does not is discussed in Sect. 3.3.

Interim Conclusion

The practical comparison of the stochastic value iteration algorithm with different noise models and simulation methods provides valuable insights into their behavior and performance in real-world applications. The results clearly demonstrate that decision problems can be effectively addressed in noisy environments, with noise actually enhancing the convergence and efficiency of algorithms. The modified simulation provides more robust and accurate results. Moreover, the multiplicative noise model generally yields superior results in terms of efficiency and higher rewards. Using the modified simulation, a more accurate approximation of the value function is achieved with multiplicative noise than with additive noise. Utilizing the multiplicative noise model tends to result in collecting more rewards in a shorter time and is less affected by parameter variations. Conversely, in the Euler-Maruyama simulation, the additive noise model often fails to terminate, even when the termination condition is set at a high threshold. This simulation may require smaller stepsizes to ensure numerical stability, particularly for higher discount factors and noise intensities.

3.2.6 Parameter Tuning

Parameter tuning is applied to find the optimal parameter adjustment for a given algorithm. This method can be used to provide the best parameter setting for the stochastic value iteration algorithm, which returns an optimal value function V^*. There are two main approaches to this process. One idea is to compare the outputs to find the parameter setting that leads to the smallest optimal value function, such that the difference to the deterministic value function is minimized. The other option is to minimize the

difference between the successive value functions $V_{t+1} - V_t$, with $V_{t+1} = V^*$. Both approaches can be used to minimize the distance to the value function obtained in value iteration algorithm.

In the following, the stochastic value iteration algorithm using the modified simulation is denoted by SBE_SIM(), which can be applied to both noise models.

Definition 3.2.1 (Parameter estimation for SBE)**.** *Let $\theta = (\gamma, \Delta t, \sigma_a)$ be the parameter model for the stochastic differential Bellman equation, where γ is the discount factor, Δt the stepsize and σ_a the noise of the action a. The* parameter estimation *for the stochastic value iteration algorithm is to find the best parameter θ^* such that*

$$\theta^* = \arg\min_{\theta} \text{SBE\_SIM}(\theta), \tag{3.2.14}$$

where SBE_SIM() *returns the norm of the optimal value function V^*.*

Iterated local search (ILS) is a stochastic search method for tuning the parameters of an algorithm [1]. It starts the local search each time with a different initial configuration and then compares the solutions to choose the best one. Several local optima can usually be found by random restarts based on the idea that the best solution is often close to a local optimum. This is done until no further improvements can be found. For instance, the hill climbing algorithm can be used for the local search. Perturbation can be defined using the concept of neighborhood to find a nearby local optimum. The pseudo-code for the iterated local search is given in Alg. 4.

Algorithm 4: Iterated Local Search

Input: θ: the parameter configuration space.
Output: θ^*: the best parameter configuration.

```
θ_0 =GenerateInitialSolution
θ* =LocalSearch(θ_0)
while termination condition not met do
    θ' =Perturbation(θ*)
    θ = LocalSearch(θ')
    if better(θ, θ*) then
        θ* = θ
    end
end
```

This search method is applied to the stochastic value iteration algorithm using hill climbing for local search to determine the best selection of the parameters $(\gamma, \Delta t, \sigma_a)$. The total number of iterations for the local search is set to $n = 1000$ and the number of restarts to 30. The forest management problem is considered with the initial value function $V_0 = (1, 1, 1)^T$. Searching for the best parameters $\theta^* = (\gamma, \Delta t, \sigma_1, \sigma_2)$ using the norm of V^* yields in the case of additive noise

$$\theta^* = (0.8, 0.000001, 0.44, 0.007)$$

and in the case of multiplicative noise

$$\theta^* = (0.8, 0.000006, 0.46, 0.89).$$

In both noise models the discount factor is the same and the stepsize is very small. The noise intensities for different actions are in similar distance to each other. The additive noise intensity of action a_2 is smaller than the intensity of multiplicative noise. Searching for the best parameter setting with respect to the norm of the difference between the successive value functions yields in the case of additive noise

$$\theta^* = (0.7, 0.01, 0.44, 0.31)$$

and in the case of multiplicative noise

$$\theta^* = (0.8, 0.0003, 0.57, 0.02).$$

Similar discount factor values result in both cases. The stepsize is increased in both noise models. In the case of additive noise, it is increased from 0.000001 to 0.01 and the stochastic algorithm achieves good results even for this stepsize.

The suggested parameters lead to the optimal policy $\pi^* = (2, 2, 2)$ in both noise models. That means that for the forest management problem subject to noise, the best decision using the parameter setting given by iterated local search is to cut wood each year. If the problem is solved for a given discount factor and stepsize, the same optimal policy as in the deterministic case is reached in both noise models. For example, with $\gamma = 0.9$, $\Delta t = 0.1$ and applying the iterated local search for $\theta = (\sigma_1, \sigma_2)$, the stochastic value iteration algorithm using additive noise yields $\theta^* = (0.83, 0.99)$ and multiplicative noise $\theta^* = (0.98, 0.53)$. This indicates that even high noise values can lead to the best result, suggesting that noise can be beneficial. Then the optimal policy in both noise models is $\pi^* = (1, 2, 2)$.

3.2.7 Statistical Tests

Statistical tests are used to determine whether there exists a *statistically significant* relationship between variables. If the variables are not statistically significant, they could be explained by chance. The null hypothesis states that there is no significant difference between groups. A p-value higher than the significance level is a strong evidence for the null hypothesis. Otherwise, the null hypothesis is rejected. A typical value for the significance level is $\alpha = 0.05$, which indicates 5% risk of concluding an actual non-significant difference as significant. In the following, the Wilcoxon-Mann-Whitney test is applied to the modified simulation of the stochastic differential Bellman equation, and confidence intervals are calculated to ascertain the relationship between the two noise models.

Wilcoxon-Mann-Whitney Test

The *Wilcoxon-Mann-Whitney test*, or *Wilcoxon rank-sum test*, is a non-parametric method applied to independent samples. It is used for non-normally distributed variables when the variables are at least ordinal. For dependent samples, the Wilcoxon signed-rank test is used. In contrast to the parametric tests, which assume a normal distribution and are based on the mean, the Wilcoxon-Mann-Whitney test shows the difference of the median values. The null hypothesis states that there is no statistically significant difference between the underlying distributions.

The Wilcoxon-Mann-Whitney test first sorts the observations of both groups in ascending order, calculates the ranks for both groups, and then sums the ranks of each group. The significance is calculated for small sample sizes and the asymptotic value for large samples ($n > 30$). In the following test case, the sample size is $n = 100$. The additive noise and multiplicative noise algorithms are executed 50 times each, and the norms of the value functions are analyzed. The Wilcoxon-Mann-Whitney test is applied to the samples and yields different results depending on the stepsize.

For the stepsize $\Delta t = 0.5$ and $\epsilon = 0.1$, the algorithm with multiplicative noise does not terminate, therefore the variable is set to $\epsilon = 0.5$.

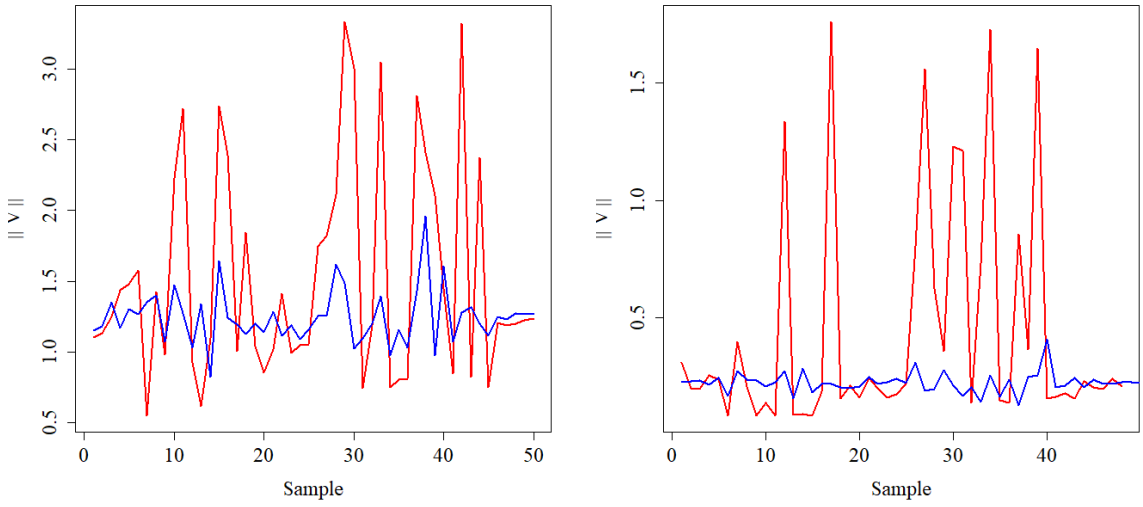
```
> wilcox.test(data_add, data_mult, paired=FALSE)
Wilcoxon rank sum test with continuity correction
W = 1289, p-value = 0.7907
```

The calculated p-value is $0.79 > 0.05$ and so the null hypothesis cannot be rejected. For the stepsize $\Delta t = 0.1$ and $\epsilon = 0.1$, the p-value is $0.43 > 0.05$ and thus the null hypothesis cannot be rejected as well. Other stepsizes $\Delta t \geq 0.1$ end up with the same result. Hence, there is no statistically significant difference between the two noise models concerning their value functions for stepsizes $\Delta t \geq 0.1$.

However, for smaller stepsizes different results are obtained. For the stepsize $\Delta t = 0.01$ and $\epsilon = 0.1$, the obtained p-value is very small, leading to the rejection of the null hypothesis. Similarly, for the stepsize $\Delta t = 0.0009$, the null hypothesis is also rejected. The two noise models differ statistically significantly from each other with regard to their value functions for stepsizes $\Delta t < 0.1$.

The value functions $\|V^*\|$ for different stepsizes are shown in Fig. 3.6. The fluctuations in the multiplicative noise decrease with smaller stepsizes and are almost constant. This shows that small stepsizes should be used to avoid high fluctuations and related errors.

This test indicates that the choice of stepsize influences the value functions. This decision can have significant impacts on the accuracy and stability of the model. Smaller stepsizes can help reduce the effects of noise and improve the stability of the model, particularly in the case of multiplicative noise, where smaller stepsizes lead to fewer



(a) (b)

Figure 3.6: Value functions in both noise models (a) $\Delta t = 0.5$ (b) $\Delta t = 0.1$.

fluctuations. For larger stepsizes, the choice of noise model may be less important, as no significant differences were found.

Confidence Interval and Absolute Error

A *confidence interval* (CI) proposes a range of values that is likely to include an unknown parameter such as the mean. It serves as a useful measure of how well the sample represents the population, particularly when the sample is normally distributed. For non-normally distributed samples, the confidence interval can be calculated by bootstrap resampling [13]. Familiarity with bootstrap resampling is assumed.

In order to analyze the stochastic differential Bellman equation with additive and multiplicative noise, the confidence intervals are calculated by bootstrap resampling. The stochastic value iteration algorithm is executed multiple times with different noise intensities and stepsizes and the optimal value functions are analyzed. The average of the norms is denoted by $\overline{V^*}$. The confidence level 95% is used, corresponding to a z-value of 1.96 from the standard normal distribution (z-Table [16]). Then the confidence interval is calculated as

$$\left[\overline{V^*} - z \cdot \frac{\sigma}{\sqrt{n}},\ \overline{V^*} + z\frac{\sigma}{\sqrt{n}}\right], \tag{3.2.15}$$

where σ denotes the standard deviation and n represents the sample size. Both noise models are analyzed with the initial condition $V_0 = (1, 1, 1)^{\mathrm{T}}$ and executed $n_1 = n_2 = 50$ times. The results are presented in the following and compared with each other.

For the stepsize $\Delta t = 0.1$ the confidence interval of the additive noise is $[0.31, 0.58]$.

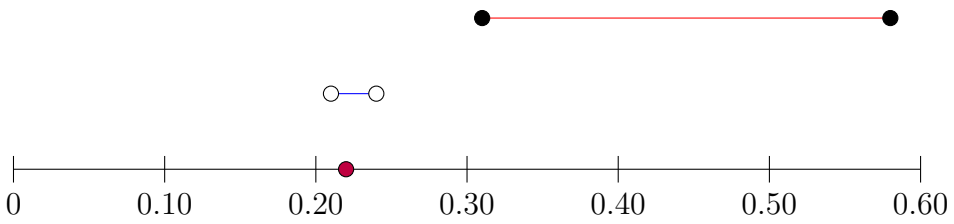


Figure 3.7: Confidence intervals. Additive noise (red line), multiplicative noise (blue line), and $\|V^*_{\text{det}}\|$ (purple circle).

Thus with probability 0.95 a confidence interval is found in which the true mean will lie between 0.31 and 0.58. This interval is wider than the confidence interval of the multiplicative noise which is $[0.21, 0.24]$. The value function is more widely distributed with additive noise. The estimated mean 0.42 for the additive noise model is not included in the interval for the multiplicative noise model, and the estimated mean 0.22 for the multiplicative noise model is not contained in the interval for the additive noise model. There is no overlap between these confidence intervals, indicating that the difference is statistically significant. This implies that the observed differences in the value functions between the two noise models are unlikely to have occurred by chance. In the absence of noise, the norm of $V^* = (0.0056, 0.1, 0.2)^{\mathrm{T}}$ is

$$\|V^*_{\text{det}}\| = 0.22$$

which lies in the confidence interval of the multiplicative noise model but not in that of the additive noise model. The confidence intervals for both noise models and the norm of V^* in the deterministic case are depicted in Fig. 3.7.

The results of the confidence intervals for different stepsizes and $\epsilon = 0.1$ are illustrated in Table 3.3. In all three cases, the observed differences are statistically significant. For stepsize $\Delta t = 0.0009$ a very small confidence interval is obtained for the multiplicative noise. The explanation for this is that the algorithm terminates after three or at most four steps independent of noise such that the value function is not widely distributed. For the stepsize $\Delta t = 0.5$ and $\epsilon = 0.5$ the confidence interval for the stochastic Bellman equation with additive noise is $[1.327, 1.763]$ and with multiplicative noise $[1.199, 1.305]$. There is still no overlap even when the lower bound of the additive noise is very close to the upper bound of the multiplicative noise and thus the difference is statistically significant. The calculation was also performed for the stepsize $\Delta t = 1$, but in most cases, both algorithms did not terminate, which leads to unreliable results and is therefore not included in the table.

The absolute errors between the calculated mean of the noise models and the norm of the value function in the case without noise are shown in Table 3.4. The *absolute error* is defined as

$$\Delta V^* = \left| \overline{V^*} - \|V^*_{\text{det}}\| \right|. \tag{3.2.16}$$

Δt	Additive noise	Multiplicative noise
0.1	$[0.313, 0.585]$	$[0.214, 0.239]$
0.01	$[0.214, 0.443]$	$[0.021, 0.026]$
0.0009	$[0.233, 0.486]$	$[0.002, 0.004]$

Table 3.3: Confidence intervals for $\epsilon = 0.1$.

Δt	$\|V^*_{\text{det}}\|$	ΔV^* additive noise	ΔV^* multiplicative noise
0.5	1.269	0.256	0.022
0.1	0.2244	0.2042	0.001
0.01	0.0223	0.2943	0.0009
0.0009	0.0020	0.3324	0.001

Table 3.4: Absolute error for different stepsizes.

The absolute error in the multiplicative noise model is smaller for each stepsize than in the additive noise model. In the multiplicative noise model, fluctuations are influenced by the previous value of the state, meaning that if the previous value is large, the noise at the current time tends to be large as well. Here, the stepsizes are small ($\Delta t \leq 0.1$) such that the value functions and the effect of the multiplicative noise also remain small. Consequently, the simulation incorporating multiplicative noise approximates the deterministic case better compared to those incorporating additive noise.

3.3 Moments of Stochastic Differential Bellman Equation

In this section, the behaviour of the stochastic differential Bellman equation and the influence of noise are analyzed theoretically. As previously discussed in Sect. 3.1, Itô's formula is inapplicable to the stochastic differential Bellman equation due to the non-linearity introduced by the maximum function. Thus, the exact solution cannot be obtained using Itô's formula. Hence, the stochastic differential Bellman equation with given policy π will be used to eliminate the necessity for the maximum function.

3.3.1 Mean and Mean Energy

In the diffusion term, the noise intensity is represented by $\sqrt{\epsilon}$, where $\epsilon > 0$. In the case of additive noise, that is

$$dV_t = (R_\pi + \gamma P_\pi V_{t-1})\, dt + \gamma\sqrt{\epsilon} dB_t, \quad t \geq 1 \tag{3.3.1}$$

and with multiplicative noise

$$dV_t = (R_\pi + \gamma P_\pi V_{t-1})\, dt + \gamma\sqrt{\epsilon} V_{t-1} dB_t, \quad t \geq 1. \tag{3.3.2}$$

The first moment or mean of the stochastic differential Bellman equation is not affected by noise as the Itô integral with respect to Brownian motion has zero mean (Prop. 2.3.3). Thus the mean of the stochastic differential Bellman equation is given as

$$\mathbb{E}V_t = \mathbb{E}\left[V_0 + \int R_\pi + \gamma P_\pi V_{t-1} dt\right] + \underbrace{\mathbb{E}\left[\gamma\sqrt{\epsilon}\int dB_t\right]}_{=0}. \tag{3.3.3}$$

To detect the influence of noise, Itô's formula is applied to the function $g = \frac{1}{2}\|V_t\|^2$. This quantity represents the energy of the system. The gradient of g is $\nabla g = V_t$ and the Hessian matrix is given by $\mathrm{H}(g) = I$, where I represents the identity matrix. By applying Itô's formula (2.3.7), it follows

$$\frac{1}{2}d\|V_t\|^2 = \left[(R_\pi + \gamma P_\pi V_{t-1})\cdot V_t + \frac{1}{2}\operatorname{Tr}\left(\sigma\sigma^{\mathrm{T}}\right)\right]dt + V_t \cdot \sigma(V_t) dB_t \tag{3.3.4}$$

where $\cdot$ denotes the scalar product and σ the diffusion term of the stochastic differential Bellman equation. The Itô integral has zero mean, so taking the mean on both sides yields

$$\frac{1}{2}\frac{d}{dt}\mathbb{E}\|V_t\|^2 = \mathbb{E}\left[(R_\pi + \gamma P_\pi V_{t-1})\cdot V_t + \frac{1}{2}\operatorname{Tr}\left(\sigma\sigma^{\mathrm{T}}\right)\right], \tag{3.3.5}$$

which is called the *mean energy change*.

The mean energy change of the stochastic differential Bellman equation can be estimated using the Gronwall inequality to obtain an upper bound [12]. If $y(t) \geq 0$, $g(t)$ and $h(t)$ are integrable and $\frac{dy}{dt} \leq g(t)y + h(t)$ for $t \geq t_0$, the *Gronwall inequality* states

$$y(t) \leq y(t_0)e^{\int_{t_0}^t g(\tau)d\tau} + \int_{t_0}^t h(s)e^{\int_s^t g(\tau)d\tau}ds, \quad t \geq t_0. \tag{3.3.6}$$

In the case where $\frac{dy}{dt} \leq gy + h$ holds for $t \geq t_0$ with g and h as constants and $t_0 = 0$, the inequality simplifies to

$$y(t) \leq y(0)e^{gt} - \frac{h}{g}\left(1 - e^{gt}\right), \quad t \geq 0. \tag{3.3.7}$$

In the deterministic case, the value function is monotonically increasing but when stochastic processes such as Brownian motion are introduced, the value function may fluctuate due to the uncertainty in the system. However, if the rewards across different states are similar, positive, and exhibit minimal fluctuations, meaning they lie within similar ranges, the value function could monotonically increase. This property can be used to provide estimates independently of the value V_{t-1} in Eq. (3.3.5). Subsequently, more general estimates are provided where this property is not utilized. Denote the elements of the probability matrix as $(P_\pi)_{i,j} = p_{i,j}$ and the rewards as $(R_\pi)_i = r_i$.

Theorem 3.3.1. *For the n-dimensional stochastic differential Bellman equation with additive noise, the estimate of the mean energy under the assumption $V_{t-1} \leq V_t$ is*

$$\mathbb{E}\|V_t\|^2 \leq \mathbb{E}\|V_0\|^2 e^{gt} - \frac{\sum_{i=1}^n r_i^2 + n\gamma^2\epsilon}{g}(1 - e^{gt}), \quad t \geq 1 \tag{3.3.8}$$

where $g = 1 + \gamma(2 + n)$ and $\epsilon > 0$.

Proof. The mean energy change (3.3.5) with additive noise is

$$\frac{1}{2}\frac{d}{dt}\mathbb{E}\|V_t\|^2 = \mathbb{E}\left[(R_\pi + \gamma P_\pi V_{t-1}) \cdot V_t\right] + \frac{1}{2}n\gamma^2\epsilon. \tag{3.3.9}$$

Evaluating the scalar product in the first term gives

$$\begin{aligned}(R_\pi + \gamma P_\pi V_{t-1}) \cdot V_t = r_1 V_t(1) + \ldots + r_n V_t(n) \\ + \gamma \sum_{i=j}^n p_{i,j} V_{t-1}(j) V_t(j) + \gamma \sum_{i \neq j}^n p_{i,j} V_{t-1}(j) V_t(i).\end{aligned}$$

By the assumption $V_{t-1} \leq V_t$ and so $V_{t-1}V_t \leq V_t^2$. Using V_t^2 instead of $V_{t-1}V_t$ yields

$$\begin{aligned}(R_\pi + \gamma P_\pi V_{t-1}) \cdot V_t \leq r_1 V_t(1) + \ldots + r_n V_t(n) \\ + \gamma \sum_{i=j}^n p_{i,j} V_t(j)^2 + \gamma \sum_{i \neq j}^n p_{i,j} V_{t-1}(j) V_t(i).\end{aligned}$$

For all $x, y \in \mathbb{R}$, $0 \leq x^2 + y^2 - 2xy = (x - y)^2$ and so $xy \leq \frac{1}{2}(x^2 + y^2)$. Using this inequality,

$$r_1 V_t(1) + \ldots + r_n V_t(n) \leq \frac{1}{2}\left(r_1^2 + \ldots + r_n^2 + V_t(1)^2 + \ldots + V_t(n)^2\right). \tag{3.3.10}$$

Similarly, $V_{t-1}(1)V_t(2) \leq \frac{1}{2}\big(V_t(1)^2 + V_t(2)^2\big) \leq \frac{1}{2}\big(V_t(1)^2 + V_t(2)^2 + \ldots + V_t(n)^2\big)$. Thus

$$\begin{aligned}(R_\pi + \gamma P_\pi V_{t-1}) \cdot V_t \leq \frac{1}{2}\sum_{i=1}^n r_i^2 + \frac{1}{2}\big(V_t(1)^2 + \ldots + V_t(n)^2\big) \\ + \gamma \sum_{i=j}^n p_{i,j} V_t(j)^2 + \frac{1}{2}\gamma \sum_{i \neq j}^n p_{i,j}\big(V_t(1)^2 + \ldots + V_t(n)^2\big).\end{aligned}$$

Furthermore, $\max\{p_{1,1}, \ldots, p_{n,n}\} \leq 1$ and $\sum_{i \neq j} p_{i,j} \leq n$. Putting this into Eq. (3.3.9) and formulating w.r.t. the norm of V_t yields

$$\frac{d}{dt}\mathbb{E}\|V_t\|^2 \leq \ (1 + 2\gamma + \gamma n)\,\mathbb{E}\|V_t\|^2 + \sum_{i=1}^n r_i^2 + n\gamma^2\epsilon. \tag{3.3.11}$$

This inequality has the general form

$$\frac{dy}{dt} \leq gy + h. \tag{3.3.12}$$

By setting $g = 1+\gamma(2+n)$ and $h = \sum_{i=1}^{n} r_i^2 + n\gamma^2\epsilon$ and using the Gronwall inequality, it yields

$$\mathbb{E}\|V_t\|^2 \leq \mathbb{E}\|V_0\|^2 e^{gt} - \frac{h}{g}\left(1 - e^{gt}\right), \quad t \geq 1. \tag{3.3.13}$$

More precisely, $g = 1+\gamma\left(2\max\{\operatorname{diag} P_\pi\} + \sum_{i\neq j} p_{i,j}\right)$, where diag denotes the diagonal elements of a matrix. □

Theorem 3.3.2. *For the n-dimensional stochastic differential Bellman equation driven by multiplicative noise under the assumption $V_{t-1} \leq V_t$, the estimate of the mean energy is*

$$\mathbb{E}\|V_t\|^2 \leq \mathbb{E}\|V_0\|^2 e^{gt} - \frac{\sum_{i=1}^{n} r_i^2}{g}\left(1 - e^{gt}\right), \quad t \geq 1, \tag{3.3.14}$$

where $g = 1+\gamma(2+n)+\gamma^2\epsilon$ and $\epsilon > 0$.

Proof. The proof is analogous to that of Thm. 3.3.1. The mean energy change (3.3.5) with multiplicative noise is

$$\frac{1}{2}\frac{d}{dt}\mathbb{E}\|V_t\|^2 = \mathbb{E}\left[(R_\pi + \gamma P_\pi V_{t-1})\cdot V_t\right] + \frac{1}{2}\gamma^2\epsilon\mathbb{E}\left[\left(V_{t-1}(1)^2 + \ldots + V_{t-1}(n)^2\right)\right]. \tag{3.3.15}$$

By the assumption $V_{t-1} \leq V_t$ which implies $V_{t-1}^2 \leq V_t V_{t-1} \leq V_t^2$. Thus

$$\frac{d}{dt}\mathbb{E}\|V_t\|^2 \leq \left(1 + 2\gamma + \gamma n + \gamma^2\epsilon\right)\mathbb{E}\|V_t\|^2 + \sum_{i=1}^{n} r_i^2. \tag{3.3.16}$$

In this case, by the Gronwall inequality, $g = 1+\gamma(2+n)+\gamma^2\epsilon$ and $h = \sum_{i=1}^{n} r_i^2$.

□

Theorem 3.3.3. *The estimate of the mean energy of the n-dimensional stochastic differential Bellman equation without noise for $V_{t-1} \leq V_t$ is*

$$\mathbb{E}\|V_t\|^2 \leq \mathbb{E}\|V_0\|^2 e^{gt} - \frac{\sum_{i=1}^{n} r_i^2}{g}\left(1 - e^{gt}\right), \quad t \geq 1, \tag{3.3.17}$$

where $g = 1+\gamma(2+n)$.

Proof. The mean energy change (3.3.5) without the diffusion term becomes

$$\begin{aligned}\frac{1}{2}\frac{d}{dt}\mathbb{E}\|V_t\|^2 &= \mathbb{E}\left[(R_\pi + \gamma P_\pi V_{t-1})\cdot V_t\right] \\ &\leq \left[\frac{1}{2} + \gamma\operatorname{diag} P_\pi + \frac{1}{2}\gamma\sum_{i\neq j} p_{i,j}\right]\mathbb{E}\|V_t\|^2 + \frac{1}{2}\sum_{i=1}^{n} r_i^2,\end{aligned}$$

where diag denotes the diagonal elements of a matrix. This is the same as in the case of additive noise but without the noise term $\frac{1}{2}\gamma^2\epsilon$. Thus $h = \sum_{i=1}^{n} r_i^2$ and g remain the same.

□

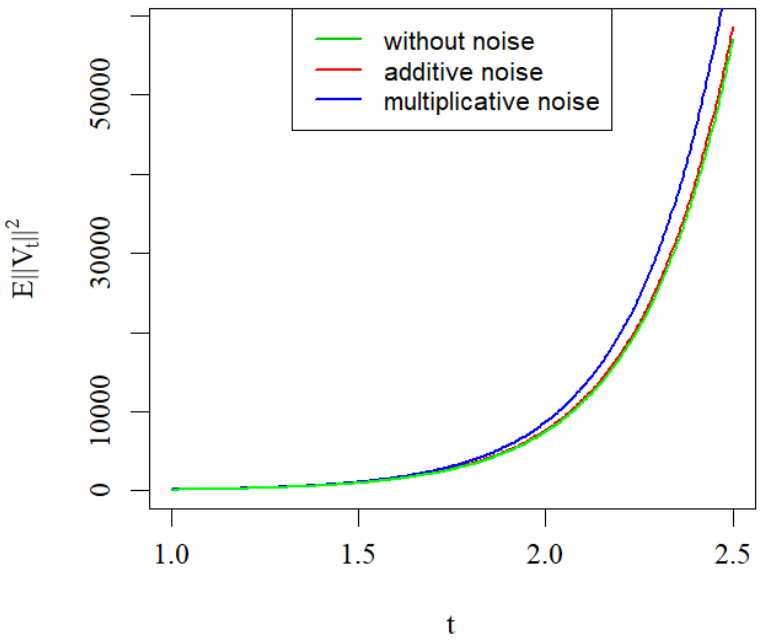


Figure 3.8: Upper bounds of the mean energy change for the additive noise, multiplicative noise, and the deterministic case.

The simplification of the above estimates is based on the bounds $\max\{\operatorname{diag} P_\pi\} \leq 1$ and $\sum_{i \neq j} p_{i,j} \leq n$. The following example makes use of the exact values.

Example 3.3.1 (Estimation of the mean energy of SBE)**.** Consider the forest management problem with given policy $\pi = (1, 2, 2)$. Then the reward and probability matrices depending on the policy are

$$R_\pi = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} \quad \text{and} \quad P_\pi = \begin{bmatrix} 0.4 & 0.6 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

Let $V_0 = (1, 0, 0)^{\mathrm{T}}$, $t = 1$, $\gamma = 0.9$, and $\epsilon = 0.1$. In the case of additive noise,

$$\mathbb{E}\|V_1\|^2 \leq \mathbb{E}\|V_0\|^2 e^{4.06} - \frac{5 + 3 \cdot 0.9^2 \cdot 0.1}{4.06} \left(1 - e^{4.06}\right) = 131.54,$$

since $g = 1 + 0.9 \cdot (2 \cdot 0.4 + 2.6) = 4.06$. In the case of multiplicative noise, $g = 1 + 0.9 \cdot (2 \cdot 0.4 + 2.6) + 0.9^2 \cdot 0.1 = 4.141$ and so

$$\mathbb{E}\|V_1\|^2 \leq \mathbb{E}\|V_0\|^2 e^{4.141} - \frac{5}{4.141} \left(1 - e^{4.141}\right) = 137.56.$$

Finally, the estimate without noise is

$$\mathbb{E}\|V_1\|^2 \leq \mathbb{E}\|V_0\|^2 e^{4.06} - \frac{5}{4.06}\left(1 - e^{4.06}\right) = 128.13.$$

The upper bounds (for $t = 1$) satisfy the following inequalities

$$\mathbb{E}_{\text{det}} < \mathbb{E}_{\text{add}} < \mathbb{E}_{\text{mult}}.$$

The upper bounds of the moments are studied in the case of $t \geq 1$. The values increase exponentially over time; Fig. 3.8 only shows the values up to $t = 2.5$. There seems to be an overlap between the upper bounds for the additive noise and the deterministic case, but upon closer inspection the upper bounds of the additive noise model are slightly larger.

◇

Theorem 3.3.4. *The upper bounds for the stochastic differential Bellman equation are in the following order*

$$\mathbb{E}_{\text{det}}\|V_t\|^2 < \mathbb{E}_{\text{add}}\|V_t\|^2 < \mathbb{E}_{\text{mult}}\|V_t\|^2, \quad t \geq 1,$$

where $V_{t-1} \leq V_t$.

Proof. Set $g = 1 + \gamma(2\max\{\operatorname{diag} P_\pi\} + \sum_{i\neq j} p_{i,j})$. Since $g > 1$ and $e^{gt} > 1$ for $t \geq 1$ the following inequalities hold. First, to show $\mathbb{E}_{\text{det}} < \mathbb{E}_{\text{add}}$, calculate the difference of the upper bounds

$$\begin{aligned}\mathbb{E}_{\text{add}} - \mathbb{E}_{\text{det}} &= \left[\mathbb{E}\|V_0\|^2 e^{gt} - \frac{\sum_{i=1}^n r_i^2 + n\gamma^2\epsilon}{g}\left(1 - e^{gt}\right)\right] - \left[\mathbb{E}\|V_0\|^2 e^{gt} - \frac{\sum_{i=1}^n r_i^2}{g}\left(1 - e^{gt}\right)\right] \\ &= -\frac{\sum_{i=1}^n r_i^2 + n\gamma^2\epsilon}{g}\left(1 - e^{gt}\right) + \frac{\sum_{i=1}^n r_i^2}{g}\left(1 - e^{gt}\right) = \frac{n\gamma^2\epsilon}{g}\left(e^{gt} - 1\right) > 0.\end{aligned}$$

Second, to prove $\mathbb{E}_{\text{add}} < \mathbb{E}_{\text{mult}}$, consider the difference

$$\begin{aligned}&\mathbb{E}_{\text{mult}} - \mathbb{E}_{\text{add}} \\ &\quad= \left[\mathbb{E}\|V_0\|^2 e^{(g+\gamma^2\epsilon)t} - \tfrac{\sum_i r_i^2}{g+\gamma^2\epsilon}\left(1 - e^{(g+\gamma^2\epsilon)t}\right)\right] - \left[\mathbb{E}\|V_0\|^2 e^{gt} - \tfrac{\sum_i r_i^2 + \gamma^2\epsilon}{g}\left(1 - e^{gt}\right)\right] \\ &\quad= \mathbb{E}\|V_0\|^2 e^{gt} e^{\gamma^2\epsilon t} - \mathbb{E}\|V_0\|^2 e^{gt} - \tfrac{\sum_i r_i^2}{g+\gamma^2\epsilon}\left(1 - e^{gt} e^{\gamma^2\epsilon t}\right) + \tfrac{\sum_i r_i^2 + \gamma^2\epsilon}{g}\left(1 - e^{gt}\right) \\ &\quad= \left(\mathbb{E}\|V_0\|^2 e^{\gamma^2\epsilon t} - \mathbb{E}\|V_0\|^2 + \tfrac{\sum_i r_i^2}{g+\gamma^2\epsilon} e^{\gamma^2\epsilon t} - \tfrac{\sum_i r_i^2 + \gamma^2\epsilon}{g}\right) e^{gt} + \tfrac{\left(\sum_i r_i^2 + g + \gamma^2\epsilon\right)\gamma^2\epsilon}{g(g+\gamma^2\epsilon)}.\end{aligned}$$

Since $\mathbb{E}\|V_0\|^2 \geq 0$ and $e^{\gamma^2\epsilon t} > 1$, it holds $\mathbb{E}\|V_0\|^2 e^{\gamma^2\epsilon t} - \mathbb{E}\|V_0\|^2 > 0$ and $\frac{\sum_i r_i^2}{g+\gamma^2\epsilon} e^{\gamma^2\epsilon t} - \frac{\sum_i r_i^2 + \gamma^2\epsilon}{g} > 0$. Thus, $\mathbb{E}_{\text{mult}} - \mathbb{E}_{\text{add}} > 0$. □

The previous estimates are based on the assumption of an increasing value function, i.e. $V_{t-1} \leq V_t$. However, as discussed in Sect. 3.2.5, the value function obtained through the stochastic value iteration algorithm is not always increasing and follows a path like Brownian motion. To define upper bounds applicable to simulations in such scenarios, this assumption is omitted in the following analysis.

General Estimates

Assume that the value function is not increasing, which is the case in the simulation. If $V_{t-1}V_t$ is not replaced by V_t^2, then applying the scalar product in the mean energy change yields

$$(R_\pi + \gamma P_\pi V_{t-1}) \cdot V_t = r_1 V_t(1) + \ldots + r_n V_t(n) + \gamma \left(\sum_{j=1}^{n} p_{1,j} V_{t-1}(j) V_t(1) + \ldots + \sum_{j=1}^{n} p_{n,j} V_{t-1}(j) V_t(n) \right).$$

Using $V_{t-1}(i)V_t(j) \leq \frac{1}{2}(V_{t-1}(i)^2 + V_t(j)^2)$ gives

$$\begin{aligned}(R_\pi + \gamma P_\pi V_{t-1}) \cdot V_t \leq{}& \frac{1}{2} \sum_{i=1} r_i^2 + \frac{1}{2} \left(V_t(1)^2 + \ldots + V_t(n)^2 \right) \\ &+ \frac{1}{2}\gamma \left(\sum_{i=1}^{n} p_{i,1} V_{t-1}(1)^2 + \ldots + \sum_{i=1}^{n} p_{i,n} V_{t-1}(n)^2 \right) \\ &+ \frac{1}{2}\gamma \left(\sum_{j=1}^{n} p_{1,j} V_t(1)^2 + \ldots + \sum_{j=1}^{n} p_{n,j} V_t(n)^2 \right).\end{aligned}$$

Since $\max\{\sum_{i=1}^{n} p_{i,1}, \ldots, \sum_{i=1}^{n} p_{i,n}\} \leq n$ and each row of the probability matrix sums up to 1,

$$(R_\pi + \gamma P_\pi V_{t-1}) \cdot V_t \leq \frac{1}{2} \left(\sum_{i=1}^{n} r_i^2 + \gamma n \|V_{t-1}\|^2 \right) + \frac{1}{2} (1+\gamma) \|V_t\|^2. \tag{3.3.18}$$

Therefore, the general mean energy change is

$$\frac{1}{2}\frac{d}{dt}\mathbb{E}\|V_t\|^2 \leq \frac{1}{2} \left(\sum_{i=1}^{n} r_i^2 + \gamma n \mathbb{E}\|V_{t-1}\|^2 \right) + \frac{1}{2} (1+\gamma) \mathbb{E}\|V_t\|^2 + \frac{1}{2}\mathbb{E}\left[\mathrm{Tr}\left(\sigma\sigma^{\mathrm{T}} \right) \right]. \tag{3.3.19}$$

Theorem 3.3.5 (General estimate for additive noise)**.** *The estimate of the general mean energy for the n-dimensional stochastic differential Bellman equation with additive noise is*

$$\mathbb{E}\|V_t\|^2 \leq \mathbb{E}\|V_0\|^2 e^{(1+\gamma)t} - \frac{\sum_{i=1}^{n} r_i^2 + \gamma n \left(\mathbb{E}\|V_{t-1}\|^2 + \gamma\epsilon \right)}{1+\gamma} \left(1 - e^{(1+\gamma)t} \right), \quad t \geq 1. \tag{3.3.20}$$

Proof. The general mean energy change with additive noise is

$$\frac{1}{2}\frac{d}{dt}\mathbb{E}\|V_t\|^2 \leq \frac{1}{2} \left(\sum_{i=1}^{n} r_i^2 + \gamma n \mathbb{E}\|V_{t-1}\|^2 \right) + \frac{1}{2} (1+\gamma) \mathbb{E}\|V_t\|^2 + \frac{1}{2} n \gamma^2 \epsilon. \tag{3.3.21}$$

By the Gronwall inequality, $g = 1 + \gamma$ and $h = \sum_{i=1}^{n} r_i^2 + \gamma n \left(\mathbb{E}\|V_{t-1}\|^2 + \gamma\epsilon \right)$. □

Theorem 3.3.6 (General estimate for multiplicative noise)**.** *For the n-dimensional stochastic differential Bellman equation with multiplicative noise, the estimate of the general mean energy is*

$$\mathbb{E}\|V_t\|^2 \leq \mathbb{E}\|V_0\|^2 e^{(1+\gamma)t} - \frac{\sum_{i=1}^n r_i^2 + (\gamma n + \gamma^2\epsilon)\mathbb{E}\|V_{t-1}\|^2}{1+\gamma}\left(1 - e^{(1+\gamma)t}\right), \quad t \geq 1. \tag{3.3.22}$$

Proof. In this case,

$$\frac{d}{dt}\mathbb{E}\|V_t\|^2 \leq \left(\sum_{i=1}^n r_i^2 + \gamma n \mathbb{E}\|V_{t-1}\|^2\right) + (1+\gamma)\,\mathbb{E}\|V_t\|^2 + \gamma^2\epsilon\mathbb{E}\|V_{t-1}\|^2. \tag{3.3.23}$$

By the Gronwall inequality, $g = 1+\gamma$ and $h = \sum_{i=1}^n r_i^2 + (\gamma n + \gamma^2\epsilon)\,\mathbb{E}\|V_{t-1}\|^2$. □

Theorem 3.3.7 (General estimate without noise)**.** *The estimate of the general mean energy for the n-dimensional stochastic differential Bellman equation without noise is*

$$\mathbb{E}\|V_t\|^2 \leq \mathbb{E}\|V_0\|^2 e^{(1+\gamma)t} - \frac{\sum_{i=1}^n r_i^2 + \gamma n\mathbb{E}\|V_{t-1}\|^2}{1+\gamma}\left(1 - e^{(1+\gamma)t}\right), \quad t \geq 1. \tag{3.3.24}$$

Proof. The proof is analog to that of additive noise. □

Theorem 3.3.8. *The mean energies for the stochastic differential Bellman equations for $t \geq 1$ satisfy the following inequalities*

(i) For $\mathbb{E}\|V_{t-1}\|^2 > n$,

$$\mathbb{E}_{\text{det}}\|V_t\|^2 < \mathbb{E}_{\text{add}}\|V_t\|^2 < \mathbb{E}_{\text{mult}}\|V_t\|^2.$$

(ii) For $0 \leq \mathbb{E}\|V_{t-1}\|^2 < n$,

$$\mathbb{E}_{\text{det}}\|V_t\|^2 < \mathbb{E}_{\text{mult}}\|V_t\|^2 < \mathbb{E}_{\text{add}}\|V_t\|^2.$$

Proof. Subtracting the additive noise estimate from the multiplicative noise estimate gives

$$\begin{aligned}
\mathbb{E}_{\text{mult}} - \mathbb{E}_{\text{add}} &= \mathbb{E}\|V_0\|^2 e^{(1+\gamma)t} - \frac{\sum_{i=1}^n r_i^2 + (\gamma n + \gamma^2\epsilon)\mathbb{E}\|V_{t-1}\|^2}{1+\gamma}\left(1 - e^{(1+\gamma)t}\right) \\
&\quad - \left[\mathbb{E}\|V_0\|^2 e^{(1+\gamma)t} - \frac{\sum_{i=1}^n r_i^2 + \gamma n\mathbb{E}\|V_{t-1}\|^2 + n\gamma^2\epsilon}{1+\gamma}\left(1 - e^{(1+\gamma)t}\right)\right] \\
&= \frac{\gamma^2\epsilon\,(n - \mathbb{E}\|V_{t-1}\|^2)}{1+\gamma}\left(1 - e^{(1+\gamma)t}\right).
\end{aligned}$$

Since $\gamma \geq 0$, $e^{1+\gamma} > 2$ and so $1 - e^{(1+\gamma)t}$ yields a negative value. The fraction is negative if and only if $n < \mathbb{E}\|V_{t-1}\|^2$. If so, $\mathbb{E}_{\text{add}} < \mathbb{E}_{\text{mult}}$. Moreover, if $n > \mathbb{E}\|V_{t-1}\|^2$, $\mathbb{E}_{\text{mult}} < \mathbb{E}_{\text{add}}$. The other inequalities are similarly proved. □

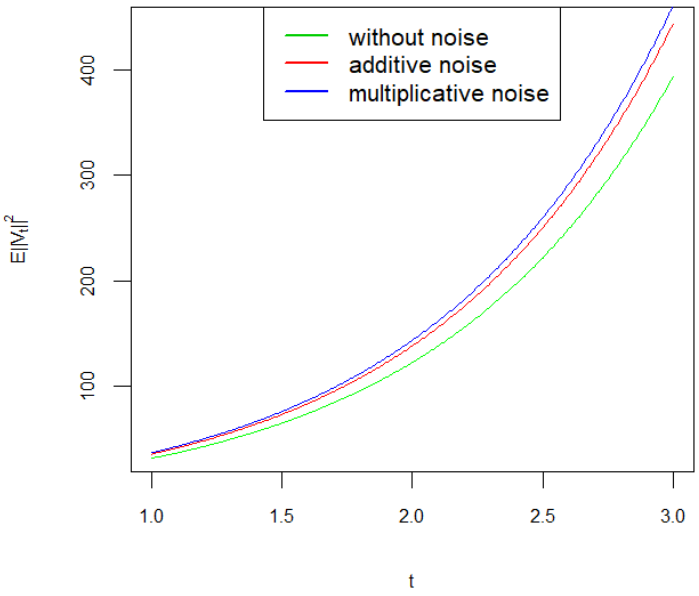


Figure 3.9: General upper bounds for $\mathbb{E}\|V_{t-1}\|^2 = 4$.

Example 3.3.2 (Mean energy of SBE)**.** Based on the values in Ex. 3.3.1, the general order of the estimates is analyzed for different values of $\mathbb{E}\|V_{t-1}\|^2$. Note that $n = 3$. Let $\gamma = \epsilon = 0.9$, $\mathbb{E}\|V_0\|^2 = 1$, and $t = 2$.

- Case $\mathbb{E}\|V_{t-1}\|^2 \geq 3$: Let $\mathbb{E}\|V_1\|^2 = 4$.
  For the additive noise,

$$\begin{aligned}\mathbb{E}\|V_2\|^2 \leq & \mathbb{E}\|V_0\|^2 e^{1.09\cdot 2} - \frac{5 + 0.9 \cdot 3 \cdot \mathbb{E}\|V_1\|^2 + 3 \cdot 0.9^2 \cdot 0.9}{1.09}\left(1 - e^{1.09\cdot 2}\right) \\ = & e^{2.18} - \frac{17.98}{1.09}\left(1 - e^{2.18}\right) = 138.32.\end{aligned}$$

  The multiplicative noise model yields

$$\begin{aligned}\mathbb{E}\|V_2\|^2 \leq & \mathbb{E}\|V_0\|^2 e^{1.09\cdot 2} - \frac{5 + (0.9 \cdot 3 + 0.9^2 \cdot 0.9)\mathbb{E}\|V_1\|^2}{1.09}\left(1 - e^{1.09\cdot 2}\right) \\ = & e^{2.18} - \frac{18.71}{1.09}\left(1 - e^{2.18}\right) = 143.57.\end{aligned}$$

  In the deterministic case, the upper bound is

$$\begin{aligned}\mathbb{E}\|V_2\|^2 \leq & \mathbb{E}\|V_0\|^2 e^{1.09\cdot 2} - \frac{5 + 0.9 \cdot 3 \cdot \mathbb{E}\|V_1\|^2}{1.09}\left(1 - e^{1.09\cdot 2}\right) \\ = & e^{2.18} - \frac{15.8}{1.09}\left(1 - e^{2.18}\right) = 122.58.\end{aligned}$$

The upper bound of the additive noise model is closer to the deterministic case.

- Case $\mathbb{E}\|V_{t-1}\|^2 < 3$: Let $\mathbb{E}\|V_1\|^2 = 0.5$.
  The estimate with additive noise is

$$\mathbb{E}\|V_2\|^2 \leq e^{2.18} - \frac{8.53}{1.09}\left(1 - e^{2.18}\right) = 70.29.$$

  and with multiplicative noise is

$$\mathbb{E}\|V_2\|^2 \leq e^{2.18} - \frac{6.71}{1.09}\left(1 - e^{2.18}\right) = 57.18.$$

  The estimate without noise is

$$\mathbb{E}\|V_2\|^2 \leq e^{2.18} - \frac{6.35}{1.09}\left(1 - e^{2.18}\right) = 54.55.$$

  In this case, the upper bound with multiplicative noise is closer to the upper bound without noise.

Both cases are illustrated for $t \geq 1$ in Figs. 3.9 and 3.10. ◇

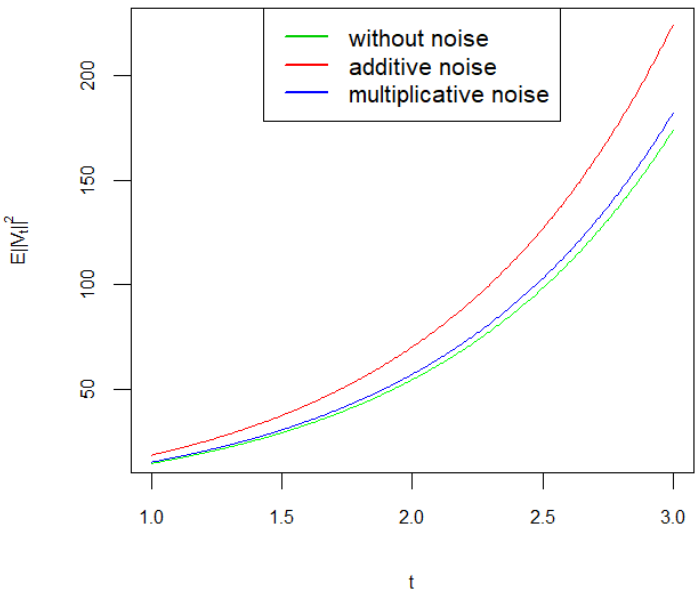


Figure 3.10: General upper bounds for $\mathbb{E}\|V_{t-1}\|^2 = 0.5$.

	$\epsilon = 0.1$	$\epsilon = 0.9$	without noise
$\gamma = 0.1$	$\mathbb{E}_{\text{add}}\|V_2\|^2 \leq 47.69$ $\mathbb{E}_{\text{mult}}\|V_2\|^2 \leq 47.70$	$\mathbb{E}_{\text{add}}\|V_2\|^2 \leq 47.84$ $\mathbb{E}_{\text{mult}}\|V_2\|^2 \leq 47.90$	$\mathbb{E}_{\text{det}}\|V_2\|^2 \leq 47.67$
$\gamma = 0.9$	$\mathbb{E}_{\text{add}}\|V_2\|^2 \leq 124.33$ $\mathbb{E}_{\text{mult}}\|V_2\|^2 \leq 124.91$	$\mathbb{E}_{\text{add}}\|V_2\|^2 \leq 138.32$ $\mathbb{E}_{\text{mult}}\|V_2\|^2 \leq 143.57$	$\mathbb{E}_{\text{det}}\|V_2\|^2 \leq 122.58$

Table 3.5: Influence of noise and discount factor.

The value function in the multiplicative noise model grows rapidly and becomes large. In this case, the effect of multiplicative noise on the mean energy is greater compared to additive noise. However, if the value function remains relatively small, such as when using a small stepsize in the simulation, the effect of multiplicative noise decreases, and the upper bound approaches the case without noise. Moreover, as the noise intensity decreases, the upper bounds of both noise models converge towards the upper bound of the deterministic case. Conversely, with increasing noise intensity, the difference between the upper bounds increases. This is illustrated in Table 3.5 using the values presented in Ex. 3.3.2 for $\mathbb{E}\|V_1\|^2 = 4$. The table demonstrates that for a small discount factor, the upper bounds of both noise models closely approximate the deterministic case and the difference is almost in decimal places. How close or far they are depends on the noise intensity, but even with a large noise the difference to the deterministic case remains small. These observations emphasize the importance of the discount factor, particularly when it approaches one, as it highlights the influence of noise. The impact of noise in the stochastic differential Bellman equation increases with higher discount factors. That is, long-term gains are more affected by noise in the mean energy change.

3.3.2 Variance

The variance represents the spread or dispersion of the values of the value function around its mean value over time and offers another possibility to illustrate the impact of noise. Analyzing the variance over time offers insights into the stability and robustness of the system. It can be determined whether the system tends to stabilize or become more unpredictable as time advances. In the following, the variance of the stochastic differential Bellman equation for both noise models is worked out step by step. The drift coefficient is the same for both noise models, so for the sake of simplicity the drift term is set to 0. The stochastic differential Bellman equation with additive noise and zero drift term

$$dV_t = \gamma\sqrt{\epsilon}dB_t, \quad V_0 \in \mathbb{R}, t > 0, \tag{3.3.25}$$

has the solution

$$V_t = V_0 + \int_0^t \gamma\sqrt{\epsilon}dB_s. \tag{3.3.26}$$

The expectation in view of the zero mean property (2.3.27) is V_0, since V_0 is a real number. The variance is calculated as

$$\mathrm{Var}(V_t) = \mathbb{E}\left[(V_t - \mathbb{E}[V_t])^2\right] = \mathbb{E}\left[\left(\int_0^t \gamma\sqrt{\epsilon} dB_s\right)^2\right]. \tag{3.3.27}$$

Then due to the Itô isometry $\mathrm{Var}(V_t) = \gamma^2\epsilon t$. In a similar manner, for the stochastic differential Bellman equation with multiplicative noise and zero drift term given by

$$dV_t = \gamma\sqrt{\epsilon} V_{t-1} dB_t, \quad V_0 \in \mathbb{R}, t > 0, \tag{3.3.28}$$

the solution due to Thm. 2.3.3 is

$$V_t = \exp\left(\gamma\sqrt{\epsilon} B_t - \frac{1}{2}\gamma^2\epsilon t\right) V_0 \tag{3.3.29}$$

with expectation V_0. The variance is then computed as

$$\begin{aligned}\mathrm{Var}(V_t) &= \mathbb{E}\left[V_t^2\right] - \mathbb{E}[V_t]^2 = \mathbb{E}\left[\exp\left(2\gamma\sqrt{\epsilon} B_t - \gamma^2\epsilon t\right) V_0^2\right] - V_0^2 \\ &= \exp\left(2\gamma^2\epsilon t - \gamma^2\epsilon t\right) V_0^2 - V_0^2 = V_0^2\left(\exp\left(\gamma^2\epsilon t\right) - 1\right).\end{aligned}$$

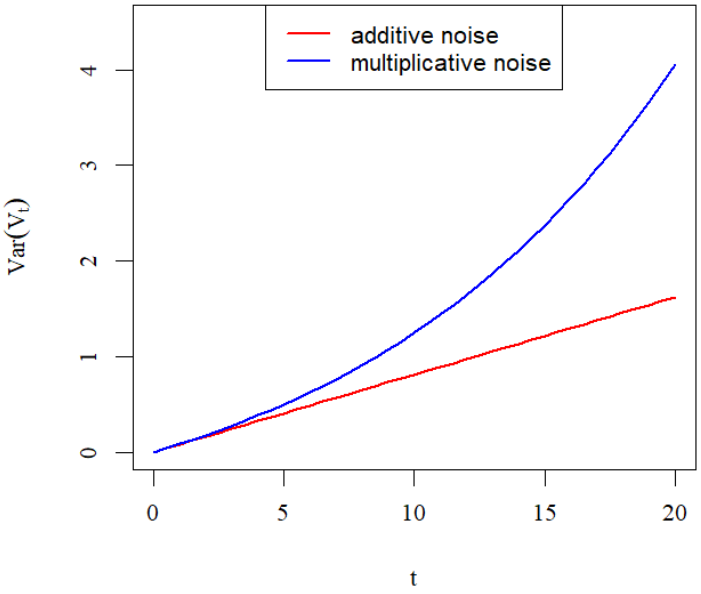


Figure 3.11: Variance of stochastic differential Bellman equation with $V_0 = 1$ and $\gamma = 0.9$.

The variance of the stochastic differential Bellman equation with multiplicative noise increases as the value function grows. A higher variance indicates greater variability

in the behavior of the value function due to stochastic influences. The case $V_0 = 1$, $\gamma = 0.9$ and $\epsilon = 0.1$ is depicted in Fig. 3.11. In this case, the variance of the SBE with multiplicative noise is greater than that of additive noise for $t > 0$, i.e. the effect of multiplicative noise is stronger. For smaller previous value function, $V_0^2 = 0.5$, and the same configurations with $\gamma = 0.9$ and $\epsilon = 0.1$, the variance of the SBE with additive noise is up to $t = 15$ larger than that of the multiplicative noise, which is illustrated in Fig. 3.12. Moreover, it is observed that the larger the discount factor, the greater the difference between the two variances, regardless of the previous value function.

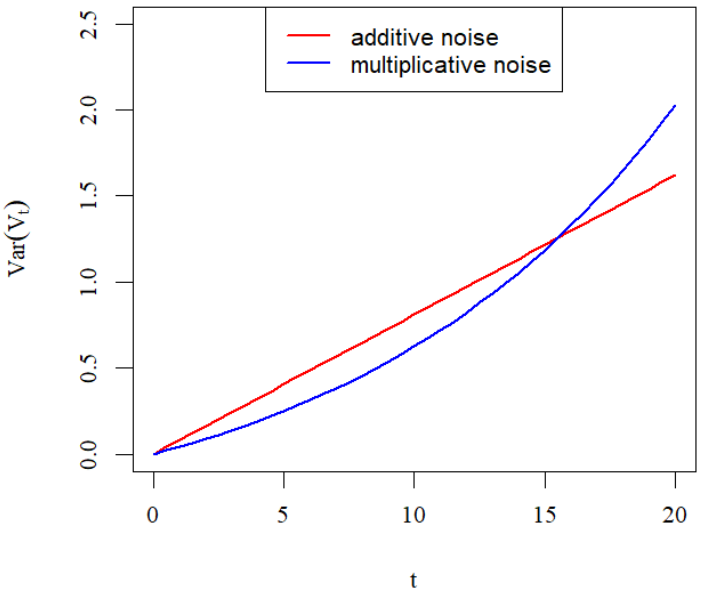


Figure 3.12: Variance of stochastic differential Bellman equation with $V_0^2 = 0.5$ and $\gamma = 0.9$.

3.3.3 Fokker-Planck Equation

The Fokker-Planck equation (FPE) is the basis of an analytical method used to describe the time evolution of probability distributions. It was first discovered in 1914 in the collaborative work of Fokker and Einstein [14] and finds applications across various fields such as physics and engineering. In [36, 55], a wide range of topics in the field of stochastic processes, including the Fokker-Planck equation and its various applications are explored. Fokker-Planck equation is obtained on the one hand by taking the expectation of a function and then its derivative. On the other hand, Itô's formula is applied to the function and subsequently the expectation is taken. The comparison of both results leads to the derivation of a linear partial equation known as the Fokker-Planck

equation. In formulating the Fokker-Planck equation, the *adjoint operator* A^* is used, which is often referred to as the Fokker-Planck operator.

The *divergence* of a vector $V(x,y) = (v_1(x,y), v_2(x,y))^{\mathrm{T}}$ is denoted by $\nabla \cdot (V(x,y))$, and is calculated as

$$\nabla \cdot (V(x,y)) = \frac{\partial v_1(x,y)}{\partial x} + \frac{\partial v_2(x,y)}{\partial y}. \tag{3.3.30}$$

Definition 3.3.1 (One-dimensional Fokker-Planck equation)**.** One-dimensional Fokker-Planck equation *for a scalar stochastic differential equation*

$$dX_t = b(X_t)dt + \sigma(X_t)dB_t, \quad X_0 = x_0 \in \mathbb{R}, \tag{3.3.31}$$

is defined as

$$\frac{\partial}{\partial t}p(x,t) = \frac{1}{2}\frac{\partial^2}{\partial x^2}\sigma^2 p(x,t) - \frac{\partial}{\partial x}bp(x,t). \tag{3.3.32}$$

In the following, the Fokker-Planck equation for the Ornstein-Uhlenbeck process is provided [15].

Example 3.3.3 (Fokker-Planck equation - Ornstein-Uhlenbeck process)**.** Reconsider the Langevin equation

$$dX_t = -bX_t dt + adB_t, \quad X_0 = x_0, \tag{3.3.33}$$

with $a, b > 0$. The Fokker-Planck equation is

$$\frac{\partial}{\partial t}p(x,t) = \frac{a^2}{2}\frac{\partial^2}{\partial x^2}p(x,t) + b\frac{\partial}{\partial x}xp(x,t). \tag{3.3.34}$$

◇

Definition 3.3.2 (Fokker-Planck equation in higher dimension)**.** *For an n-dimensional stochastic differential equation*

$$dX_t = b(X_t)dt + \sigma(X_t)dB_t, \quad X_0 = x_0 \in \mathbb{R}^n, \tag{3.3.35}$$

the Fokker-Planck equation in higher dimension *is defined as*

$$\frac{\partial}{\partial t}p(x,t) = A^* p(x,t), \tag{3.3.36}$$

where

$$A^* p = \frac{1}{2}\operatorname{Tr}\left(\mathrm{H}(\sigma\sigma^T p)\right) - \nabla \cdot (bp) \tag{3.3.37}$$

and p is the probability density function. The initial condition is $p(x,0) = \delta(x - x_0)$, where $\delta(\cdot)$ denotes the Dirac delta function.

So far, the general form of the Fokker-Planck equation for multidimensional stochastic differential equations has been elucidated. This is a well-established concept in the literature. The following theorem demonstrates the specific application of the Fokker-Planck equation to the stochastic differential Bellman equation. This represents a novel contribution to the field of study. It illustrates how the Fokker-Planck equation can be utilized to model and analyze the probability density in a particular stochastic control problem.

Theorem 3.3.9. *The Fokker-Planck equation for an n-dimensional stochastic differential Bellman equation with given policy π is*

$$A^*p = \frac{1}{2}\operatorname{Tr}\left(\mathrm{H}(\sigma_\pi \sigma_\pi^T p)\right) - \nabla \cdot \left((R_\pi + \gamma P_\pi V_{t-1})p\right), \tag{3.3.38}$$

with the initial condition $p(x,0) = \delta(x - x_0)$.

Proof. For the stochastic differential Bellman equation

$$dV_t = (R_\pi + \gamma P_\pi V_{t-1})\, dt + \sigma_\pi(V_t) dB_t$$

the FPE is given by Eq. (3.3.37) with $\sigma = \sigma_\pi$ and $b = R_\pi + \gamma P_\pi V_{t-1}$. Note that $\sigma_\pi(V_t)$ can be additive $\gamma\sigma_\pi$ or multiplicative $\gamma\sigma_\pi V_{t-1}$. □

The Fokker-Planck equation for both additive and multiplicative noise are provided in the following.

Theorem 3.3.10. *Consider the n-dimensional stochastic differential Bellman equation with given policy π and additive noise*

$$dV_t = (R_\pi + \gamma P_\pi V_{t-1})\, dt + \gamma\sqrt{\epsilon} dB_t, \tag{3.3.39}$$

where $\gamma \in (0,1)$ and $\epsilon > 0$. The probability density function satisfies

$$\frac{\partial}{\partial t}p(x,t) = \frac{\gamma^2\epsilon}{2}\sum_{i=1}^{n}\frac{\partial^2}{\partial x_i^2}p(x,t) - \sum_{j=1}^{n}\frac{\partial}{\partial x_j}v_j p(x,t), \tag{3.3.40}$$

where v_j is the j-th row entry of the vector $R_\pi + \gamma P_\pi V_{t-1}$.

Proof. Utilizing the Fokker-Planck equation for the stochastic differential Bellman equation with additive noise (Eq. (3.3.38)) gives

$$A^*p(x,t) = \frac{1}{2}\operatorname{Tr}\left(\mathrm{H}(\gamma^2\epsilon p(x,t))\right) - \nabla \cdot \left((R_\pi + \gamma P_\pi V_{t-1})p(x,t)\right). \tag{3.3.41}$$

The Hessian matrix is computed as

$$\mathrm{H}(\gamma^2\epsilon p) = \gamma^2\epsilon \begin{bmatrix} \frac{\partial^2}{\partial x_1^2}p & \cdots & \frac{\partial^2}{\partial x_1 \partial x_n}p \\ \vdots & \ddots & \vdots \\ \frac{\partial^2}{\partial x_n \partial x_1}p & \cdots & \frac{\partial^2}{\partial x_n^2}p \end{bmatrix}.$$

Applying the trace and the divergence provides

$$A^*p = \frac{\gamma^2\epsilon}{2}\sum_{i=1}^{n}\frac{\partial^2}{\partial x_i^2}p(x,t) - \sum_{j=1}^{n}\frac{\partial}{\partial x_j}v_j p(x,t),$$

where v_j is the j-th row entry of vector $R_\pi + \gamma P_\pi V_{t-1}$. □

Theorem 3.3.11. *The probability density function of the n-dimensional stochastic differential Bellman equation with multiplicative noise*

$$dV_t = (R_\pi + \gamma P_\pi V_{t-1})\, dt + \gamma\sqrt{\epsilon} V_{t-1} dB_t \tag{3.3.42}$$

satisfies

$$\frac{\partial}{\partial t} p(x,t) = \frac{\gamma^2 \epsilon}{2} \sum_{i,j}^{n} \frac{\partial^2}{\partial x_i x_j} \left(V_{t-1}^2\right)_{i,j} p(x,t) - \sum_{j=1}^{n} \frac{\partial}{\partial x_j} v_j p(x,t), \tag{3.3.43}$$

where v_j represents the j-th row entry of the vector $R_\pi + \gamma P_\pi V_{t-1}$.

Proof. Eq. (3.3.38) with multiplicative noise gives

$$A^* p(x,t) = \frac{1}{2} \operatorname{Tr}\left(\mathrm{H}(\gamma^2 \epsilon V_{t-1}^2 p(x,t))\right) - \nabla \cdot ((R_\pi + \gamma P_\pi V_{t-1}) p(x,t)). \tag{3.3.44}$$

Taking the Hessian matrix and applying the trace and the divergence yields

$$A^* p = \frac{\gamma^2 \epsilon}{2} \sum_{i,j}^{n} \frac{\partial^2}{\partial x_i x_j} \left(V_{t-1}^2\right)_{i,j} p(x,t) - \sum_{j=1}^{n} \frac{\partial}{\partial x_j} v_j p(x,t),$$

where v_j is the j-th row entry of vector $R_\pi + \gamma P_\pi V_{t-1}$. □

Example 3.3.4 (Fokker-Planck equation for SBE)**.** For the two-dimensional case, write $V_{t-1} = (X, Y)^{\mathrm{T}}$, $V_\pi = R_\pi + \gamma P_\pi V_{t-1}$ and $V_0 = (x_0, y_0)^{\mathrm{T}}$. Then the Fokker-Planck equation for the stochastic differential Bellman equation driven by a multiplicative noise is

$$p_t = \frac{\gamma^2 \epsilon}{2} \left(\left(x^2 p\right)_{xx} + \left(y^2 p\right)_{yy}\right) - \left[\left(V_{\pi;1} p\right)_x + \left(V_{\pi;2} p\right)_y\right], \tag{3.3.45}$$

where $V_{\pi;i}$ is the i-th row entry of the vector V_π. The initial condition is given by

$$p(x,y,0) = \delta(x - x_0, y - y_0). \tag{3.3.46}$$

Moreover, the Fokker-Planck equation for the stochastic differential Bellman equation with additive noise is

$$p_t = \frac{\gamma^2 \epsilon}{2} (p_{xx} + p_{yy}) - \left[\left(V_{\pi;1} p\right)_x + \left(V_{\pi;2} p\right)_y\right]. \tag{3.3.47}$$

◇

Stationary solution of the Fokker-Planck equation

In this section, the stationary solution of the one-dimensional Fokker-Planck equation is presented [18]. The Fokker-Planck equation for a stochastic differential Bellman equation is defined as

$$p_t = \frac{1}{2} \operatorname{Tr}\left(\mathrm{H}(\sigma_\pi \sigma_\pi^{\mathrm{T}} p)\right) - \nabla \cdot ((R_\pi + \gamma P_\pi V_{t-1}) p). \tag{3.3.48}$$

Suppose the drift term is zero, that is, $R_\pi + \gamma P_\pi V_{t-1} = 0$. Then the one-dimensional Fokker-Planck equation simplifies to

$$p_t = \frac{1}{2}\frac{\partial^2}{\partial x^2}\sigma_\pi \sigma_\pi^{\mathrm{T}} p(x,t). \tag{3.3.49}$$

This equation provides insight into the behavior of the probability density function under the absence of the drift term.

Theorem 3.3.12. *For the stochastic differential Bellman equation with additive noise and zero drift term, the one-dimensional Fokker-Planck equation due to Eq. (3.3.49) is*

$$p_t = \frac{\gamma^2 \epsilon}{2}\frac{\partial^2}{\partial x^2} p(x,t). \tag{3.3.50}$$

The probability density function (PDF) satisfying this FPE is

$$p(x,t) = \frac{1}{\sqrt{2\pi\gamma^2\epsilon t}}\, \mathrm{e}^{-\frac{(x-x_0)^2}{2\gamma^2\epsilon t}}. \tag{3.3.51}$$

Proof. The expectation and the variance of a stochastic differential Bellman equation are calculated in Sect. 3.3.2. The solution of the SBE with additive noise is

$$V_t = V_0 + \int_0^t \gamma\sqrt{\epsilon} dB_s, \quad V_0 = x_0 \in \mathbb{R} \tag{3.3.52}$$

which is a Gaussian process with expectation x_0 and variance $\gamma^2\epsilon t$. Consequently, the associated probability density function is given by

$$p(x,t) = \frac{1}{\sqrt{2\pi\gamma^2\epsilon t}}\, \mathrm{e}^{-\frac{(x-x_0)^2}{2\gamma^2\epsilon t}}.$$

□

Theorem 3.3.13. *For the stochastic differential Bellman equation with multiplicative noise and zero drift term, the one-dimensional Fokker-Planck equation w.r.t. Eq. (3.3.49) is*

$$p_t = \frac{\gamma^2 \epsilon}{2}\frac{\partial^2}{\partial x^2} V_{t-1}^2 p(x,t). \tag{3.3.53}$$

The probability density function (PDF) satisfying this FPE is

$$p(x,t) = \frac{1}{\sqrt{2\pi\gamma^2\epsilon t}x} \exp\left(-\frac{[\ln(x) - \mu_x]^2}{2\gamma^2\epsilon t}\right). \tag{3.3.54}$$

where $\mu_x = \ln(x_0) - \frac{\gamma^2\epsilon t}{2}$.

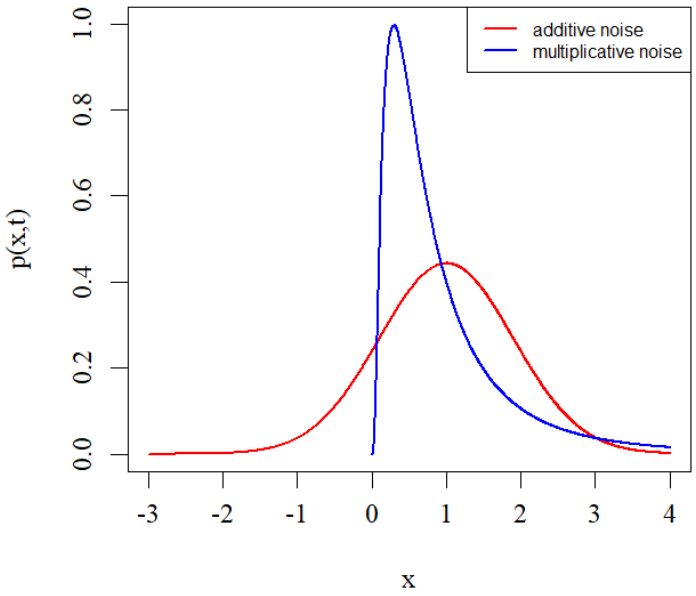


Figure 3.13: PDFs of SBE with additive noise and multiplicative noise for $\gamma = 0.9$.

Proof. Analogue to the case of additive noise, the solution of the SBE with multiplicative noise given in (3.3.29) is

$$V_t = \exp\left(\gamma\sqrt{\epsilon}B_t - \frac{\gamma^2\epsilon t}{2}\right)V_0, \quad V_0 = x_0 \in \mathbb{R}$$

which is log-normally distributed with location parameter $\ln(x_0) - \frac{\gamma^2\epsilon t}{2}$ and squared scale parameter $\gamma^2\epsilon t$. □

Example 3.3.5 (Probability density function of SBE)**.** The probability density function of a stochastic differential Bellman equation is examined under both additive and multiplicative noise. Let $\epsilon = 0.1$ and $V_0 = 1$. The probability density functions at $t = 10$ for different discount factors are illustrated in Fig. 3.13 and Fig. 3.14. The larger the discount factor, the larger the difference between the two PDFs. The smaller the discount factor, the more similarly the functions are distributed. When focusing solely on the probability density function of the SDE with additive noise, then it can be seen that the larger the discount factor, the more widespread the PDF is, which is shown in Fig. 3.15. Thus, it can be concluded that the effect of noise depends on the discount factor. ◇

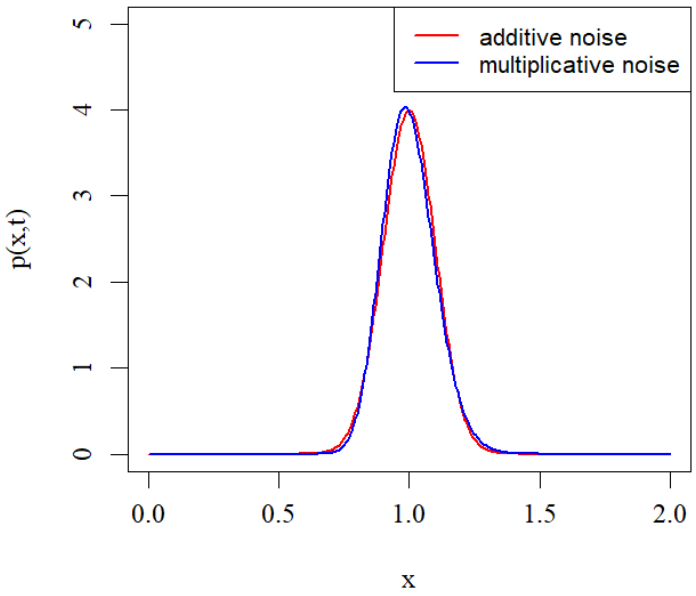


Figure 3.14: PDFs of SBE with additive noise and multiplicative noise for $\gamma = 0.1$.

3.4 Reinforcement Learning in Noisy Environments

Previous research has extensively explored Q-learning and its variants in various contexts as discussed in Sect. 2.2.5. However, real-world environments often exhibit stochastic dynamics, which cannot be handled in traditional Q-learning algorithm. To address these challenges, the update rule is adapted to incorporate the noise, enabling agents to learn optimal policies in stochastic environments with unknown transition probabilities. This contribution represents an advancement in reinforcement learning, offering a method for tackling real-world decision-making problems under uncertainty. First, the stochastic differential Q-function is introduced, which is the cumulative reward that the agent can expect while acting optimally under noise.

Definition 3.4.1 (Stochastic differential Q-function with additive noise)**.** *Let* $\gamma \in [0, 1)$ *and* $t \geq 1$. *The* stochastic differential Q-function with additive noise *is defined as*

$$dQ_t(a) = (R_a + \gamma P_a V_{t-1})\, dt + \gamma \sigma_a dB_t, \quad Q_0(a) \in \mathbb{R}, \tag{3.4.1}$$

where $R_a \in \mathbb{R}$ *for each* $a \in A$, $V_{t-1} = \max_{a \in A}\{Q_{t-1}(a)\}$ *and* B_t *is the Brownian motion.*

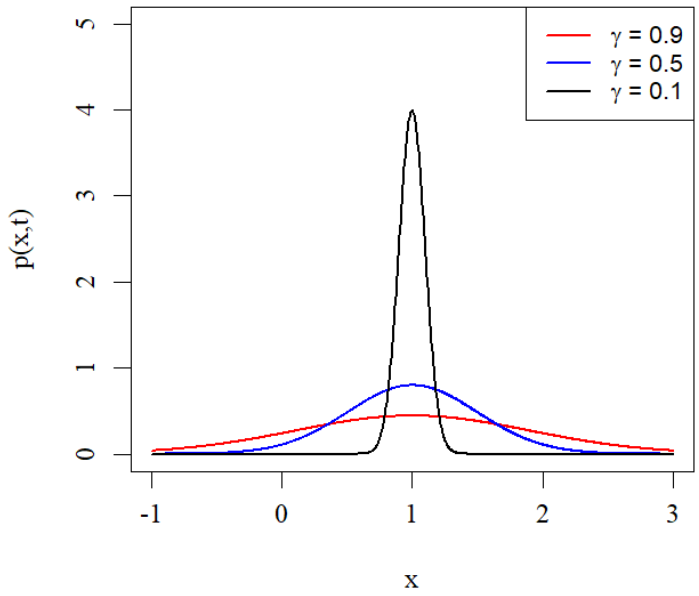


Figure 3.15: PDFs of SBE with additive noise for different discount factors.

Definition 3.4.2 (Stochastic differential Q-function with multiplicative noise)**.** *For* $\gamma \in [0,1)$ *and* $t \geq 1$*, the* stochastic differential Q-function driven by multiplicative noise *is*

$$dQ_t(a) = (R_a + \gamma P_a V_{t-1})\, dt + \gamma \sigma_a Q_{t-1}(a) dB_t, \quad Q_0(a) \in \mathbb{R}, \tag{3.4.2}$$

where $R_a \in \mathbb{R}$ *for each* $a \in A$*,* $V_{t-1} = \max_{a \in A}\{Q_{t-1}(a)\}$ *and* B_t *is the Brownian motion.*

The solution of the stochastic differential Q-function can be obtained by using the known approximations. This provides a computationally tractable method for approximating the stochastic differential Q-function.

Theorem 3.4.1. *The numerical solution of the stochastic* Q*-function is given as*

$$Q_{i+1}(a) = (R_a + \gamma P_a V_i)\, \Delta t + \gamma \sigma(Q_i)\, (B_{i+1} - B_i)\,, \tag{3.4.3}$$

where $V_i = \max_{a \in A}\{Q_i(a)\}$ *and* $\sigma(Q_i)$ *is either additive noise* σ_a *or multiplicative noise* $\sigma_a Q_i(a)$*.*

Proof. Recall that the given stochastic differential equation is interpreted as

$$Q_t(a) = Q_0(a) + \int_0^t (R_a + \gamma P_a V_{u-1})\, du + \int_0^t \sigma(Q_{u-1}) dB_u. \tag{3.4.4}$$

The integral $\int_0^t (R_a + \gamma P_a V_{u-1})\, du$ is approximated by $(R_a + \gamma P_a V_{t-1})\Delta t$ and the Itô integral by $\sigma(Q_{t-1})\Delta B_i$, where $\Delta B_i = (B_{i+1} - B_i)$. This discretization leads to the numerical solution. □

3.4.1 Stochastic Q-Learning Algorithm

To adapt the Q-learning algorithm to the stochastic differential Q-function, a stochastic update rule will be introduced that takes into account the uncertainty of the environment. From here on, a constant learning rate $\alpha_t = \alpha$ for all $t \geq 0$ is assumed.

Definition 3.4.3 (Stochastic Q-learning algorithm)**.** *The stochastic Q-learning algorithm uses the following update rule*

$$\begin{aligned} &Q_{i+1}(s,a) = \\ &\quad Q_i(s,a) + \alpha\left[R(s,a) + \gamma V_i(s') - Q_i(s,a)\right]\Delta t + \alpha\gamma\sigma(Q_i)\left(B_{i+1} - B_i\right), \end{aligned}$$

where $\sigma(Q_i)$ is either additive noise σ_a or multiplicative noise $\sigma_a Q_i(s,a)$ and $V_i(s') = \max_a\{Q_i(s',a)\}$ is the predicted maximum Q-value over all actions for the next state s'.

Example 3.4.1 (Stochastic Q-learning)**.** Consider Ex. 2.2.8 in the stochastic case with additive noise. Let noise be dependent on the chosen action, $\sigma_1 = 0.5$, $\sigma_2 = 0.3$ and put the stepsize to $\Delta t = 1$. The Q-function at the beginning is set to

$$Q_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 2 \end{bmatrix}.$$

Assume that the agent chooses the action a_1, such that state s_2 is reached and the calculated value is $Q_1(s_1,a_1) = 0.033$. In state s_2, assume the agent takes action a_2 with the reward $R(s_2,a_2) = 1$, so

$$\begin{aligned} &Q_2(s_2,a_2) \\ &\quad = Q_1(s_2,a_2) + 0.1 \cdot \Big[1 + 0.9 \cdot \max_a\{Q_1(s_1,a)\} - Q_1(s_2,a_2)\Big] + 0.027 \cdot (B_2 - B_1) \\ &\quad = 0.1 \cdot [1 + 0.9 \cdot 0.033] + 0.027 \cdot (B_2 - B_1) = 0.094. \end{aligned}$$

The Q-function is updated in the following order $Q_3(s_1,a_1)$, $Q_4(s_2,a_2)$, $Q_5(s_1,a_1)$, $Q_6(s_2,a_1)$. After updating $Q_6(s_2,a_1)$, the episode ends and the updated Q-function is

$$Q_6 = \begin{bmatrix} 0.14 & 0 \\ 0.22 & 0.18 \\ 1 & 2 \end{bmatrix}.$$

The optimal value function for the stochastic Q-learning subject to additive noise is $V^* = (0.14, 0.22, 2)$ with the optimal policy $\pi^* = (1,1,2)$. In the case of multiplicative noise, the same order of actions leads to the updated Q-function

$$Q_6 = \begin{bmatrix} 0.025 & 0 \\ 0.180 & 0.189 \\ 1 & 2 \end{bmatrix}$$

with the optimal policy $\pi^* = (1, 2, 2)$. So different optimal policies are obtained depending on which noise model is used. In this example, the optimal policy obtained in the case of multiplicative noise leads to higher rewards.

$\diamond$

The algorithm is given in the following.

Algorithm 5: Stochastic Q-Learning Algorithm

Input: Terminal state s_F, $\alpha_t > 0$, $\gamma \in [0, 1)$, $\sigma_a > 0$ for all $a \in A$ and $\Delta t > 0$.
Output: Updated Q-function

Initialize $Q_0(s, a) = 0$ for all $s \in S$ and $a \in A$
$i = 0$
Observe current state s
do
  Select an action a
  Observe next state s' and reward $R(s, a)$
  $Q_{i+1}(s, a) =$
   $Q_i(s, a) + \alpha \left[R(s, a) + \gamma \max_{a'}\{Q_i(s', a')\} - Q_i(s, a)\right] \Delta t + \alpha\gamma\sigma_a(B_{i+1} - B_i)$
  $s = s'$
  $i = i + 1$
while $s \neq s_F$;
return Q

3.4.2 Practical Results

In the following, Ex. 3.4.1 is tackled using stochastic Q-learning with additive noise and multiplicative noise in order to compare the results, where the order of actions is not predefined. The algorithms are configured to run for 1000 episodes, with different discount factors and σ-values for learning rate $\alpha \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$. The stepsize is set to 0.1. In both noise models, the optimal policy remains almost always the same, that is, $\pi^* = (1, 2, 2)$, independent of the learning rate, discount factor and noise intensity. However, the optimal policy $\pi^* = (2, 2, 2)$ also occurs.

```
# Stochastic Q-learning with additive noise
> StocQ_add(states, 0.1, 0.9, sigma, dis, 3, 0.1, 1000)
$Episodes
[1] 1000
$`Running time`
Time difference of 0.09376097 secs
$Q
        wait      cut
s1 10.135217 9.710252
s2  1.965597 9.816785
s3  1.000000 2.000000
$`Optimal Policy`
```

```
s1 s2 s3
 1  2  2

# Stochastic Q-learning with multiplicative noise
> StocQ_mult(states, 0.1, 0.9, sigma, dis, 3, 0.1, 1000)
$Episodes
[1] 1000
$`Running time`
Time difference of 0.125047 secs
$Q
           wait      cut
s1 1.009406e+06 996883.8
s2 2.058122e+00 412858.6
s3 1.000000e+00      2.0
$`Optimal Policy`
s1 s2 s3
 1  2  2
```

The Q-values in the case of multiplicative noise are very large after 1000 episodes. Thus the next step is to optimize the stochastic Q-learning algorithm using the ϵ-Greedy strategy, introduced in the previous chapter. This optimization is implemented with $\epsilon = 0.1$.

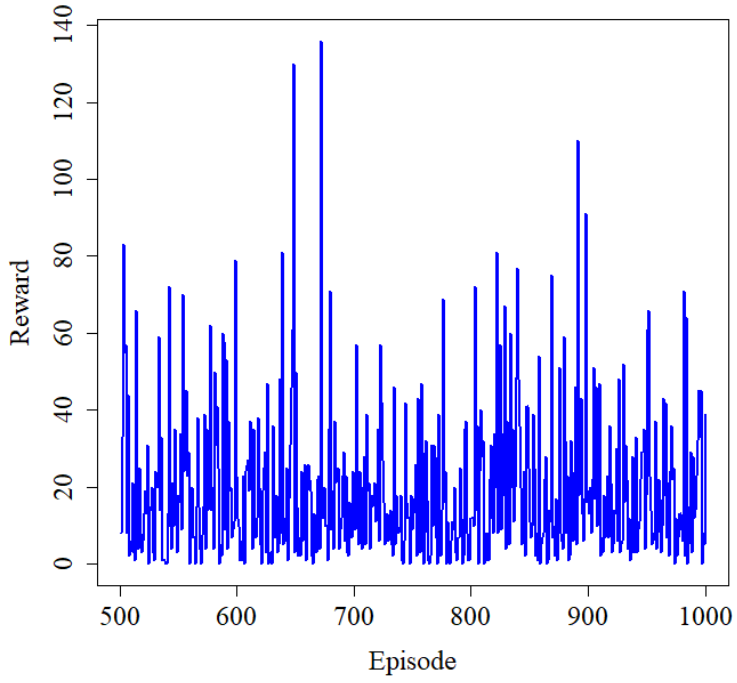


Figure 3.16: Maximum reward per episode (multiplicative noise).

```
# Stochastic Q-learning using Epsilon-Greedy with additive noise
> StocQ_add_eps(states, 0.1, 0.9, sigma, dis, 3, 0.1, 1000)
$Episodes
[1] 1000
$`Running time`
Time difference of 1.553728 secs
$Q
        wait      cut
s1 11.816696 11.23425
s2  1.889765 11.94642
s3  1.000000  2.00000

$`Optimal Policy`
s1 s2 s3
 1  2  2

# Stochastic Q-learning using Epsilon-Greedy with multiplicative noise
> StocQ_mult_eps(states, 0.1, 0.9, sigma, dis, 3, 1, 1000)
$Episodes
[1] 1000
$`Running time`
Time difference of 3.409875 secs
$Q
        wait      cut
s1 11.871830 11.47081
s2  1.944877 12.80417
s3  1.000000  2.00000
$`Optimal Policy`
s1 s2 s3
 1  2  2
```

The optimal policies remain the same between the two noise models, but there are notable differences in runtime performance. Incorporating the ϵ-Greedy strategy results in increased runtimes. The larger the discount factor and the learning rate, the longer the algorithms take. The stepsize also affects the runtime, the smaller the stepsize, the longer the algorithms need to terminate. For example, in case of $\Delta t = 0.01$, the runtime of the additive noise algorithm increases to 7.56 seconds while for $\Delta t = 1$ it only takes 0.62 seconds. Despite differences in runtime, the values obtained in the case of multiplicative noise remain similar to those in the additive case. Furthermore, considering the mean and the maximum rewards, the multiplicative noise algorithm reaches greater rewards compared to the additive noise algorithm. In both noise models, incorporating noise yields higher rewards compared to scenarios without noise. The maximum rewards per episode for the multiplicative case is illustrated in Fig. 3.16.

```
# Maximum and Mean of rewards in case of additive noise
> max(data_add)
```

```
[1] 100
> mean(data_add)
[1] 17.976
# Maximum and Mean of rewards in case of multiplicative noise
> max(data_mult)
[1] 136
> mean(data_mult)
[1] 19.456
# Maximum and Mean of rewards without noise
> max(data_det)
[1] 9
> mean(data_det)
[1] 1.054
```

The comparison between the average reward per episode reveals an unexpected trend: the average reward per episode is smaller in the case of multiplicative noise compared to the additive noise. The reason is that the number of steps per episode is larger in the case of multiplicative noise. The average rewards per episode for both noise models are illustrated in Fig. 3.17, focusing on the last 300 episodes. As each episode in the case of multiplicative noise comprises more steps, the agent has more opportunities for exploration. However, during these additional steps, there may also be more errors or undesirable actions, potentially reducing the average reward per episode.

```
> mean(avrg_add) # Mean of the average rewards with additive noise
[1] 0.3273806
> mean(avrg_mult) # Mean of the average rewards with multip. noise
[1] 0.1176477
```

3.5 Case Studies

In this section, real-life decision problems are considered and solved using the methods presented in this work. In the first case study, a machine replacement problem is solved through stochastic value iteration using the modified Euler-Maruyama method and the results are analyzed in relation to different parameters. In the second case study, an inventory problem is tackled utilizing four algorithms: value iteration, stochastic value iteration, Q-learning and stochastic Q-learning algorithm.

3.5.1 Replacement Problem

So far, the forest management problem was discussed in detail, where the value function was small, so the stochastic value iteration algorithm with multiplicative noise performs better than with additive noise. In this section, a replacement problem is considered where the value function is large. The replacement problem subject to noise is addressed using the modified Euler-Maruyama method by examining the influence of the parameters. Different stepsizes, discount factors, and noise values are taken into

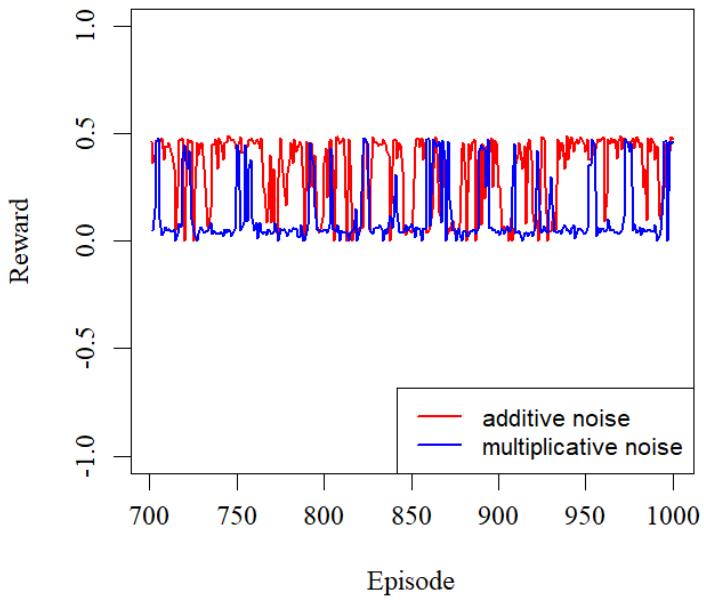


Figure 3.17: Average reward per episode with additive noise and multiplicative noise.

account. This stochastic replacement problem is based on the work of Hastings [20] in which the repair limit replacement method is presented. The repair limit replacement problem involves replacing or repairing an item, which is decided based on its age and a predefined repair limit. If the estimated repair cost is less than the repair limit, the item will be repaired. The failure rate depends on the age of the item. By interpretation, in state s_i the item is of age $i \geq 0$.

Assume that a machine can last up to two years. State s_0 indicates a machine in the first year of life (age 0), state s_1 stands for the age of 1 and in state s_2 the machine reaches the maximum age limit 2. The set of actions is $A = \{a_1, a_2\}$ where $a_1 = 1$ is repair and $a_2 = 2$ is replace. During the first year of life in s_0, the failure rate of a machine is two failures per year and during the second year in s_1, three failures per year. The cost of a subsequent repair has a negative exponential distribution with €100 in the first year and €150 in the second year. The repair limit is €300 in the first year and €100 in the second year. The cost of a new machine is €400. Then the probability of replacement at a single repair can be calculated using the repair limit and the cost of subsequent repair. In state s_0, the probability of replacement at a single repair is calculated as

$$r_0 = \exp\left(-\frac{300}{100}\right) = 0.0498.$$

The probability of survival p_0, i.e. that a machine will be in the next state without

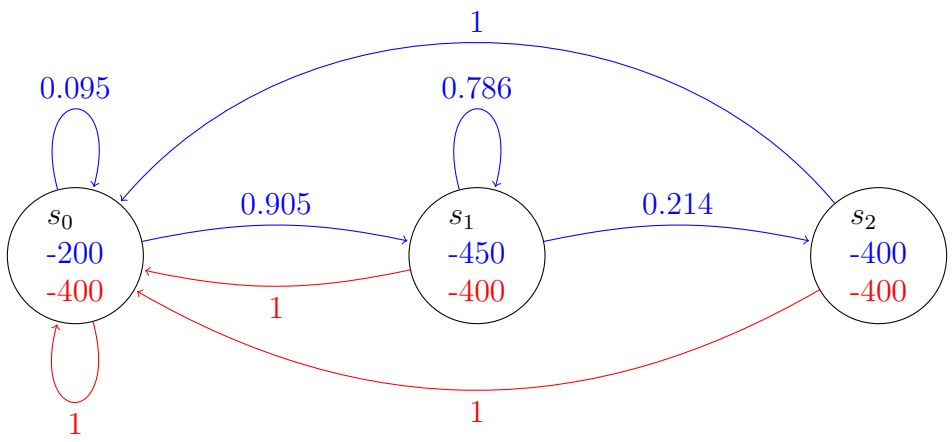


Figure 3.18: Replacement problem.

replacement is calculated using r_0 and the failure rate in the first year as

$$p_0 = \exp(-2 \cdot 0.0498) = 0.905.$$

Similarly, in state s_1 the probability of replacement at a single repair is

$$r_1 = \exp\left(-\frac{100}{150}\right) = 0.513.$$

And the probability of reaching the next state without replacement is

$$p_1 = \exp(-3 \cdot 0.513) = 0.214,$$

since the failure rate in the second year is three. Thus the probability matrices are

$$P_{(,,1)} = \begin{bmatrix} 0.095 & 0.905 & 0 \\ 0 & 0.786 & 0.214 \\ 1 & 0 & 0 \end{bmatrix} \text{ and } P_{(,,2)} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

For simplicity, the costs are rewards. During the first year of life, two failures can occur with repair cost of €100 each, so the reward at s_0 is -200. During the second year, three failures can occur with a repair cost of €150 each, so the reward in s_1 is -450. In state s_2, the maximum age is reached such that the machine must be replaced. The acquisition cost is -400 in each state. The reward matrices are

$$R_{(,1)} = \begin{bmatrix} -200 \\ -450 \\ -400 \end{bmatrix} \text{ and } R_{(,2)} = \begin{bmatrix} -400 \\ -400 \\ -400 \end{bmatrix}.$$

The replacement problem is sketched in Fig. 3.18, where the blue edges represent the action repair, and the red edges the action replace.

Suppose this replacement problem is subject to noise. The goal is to determine an optimal policy that minimizes the cost. Replacing the machine too early leads to a large loss in value, while replacing it too late leads to high maintenance costs. The stochastic replacement problem will be solved using the stochastic differential Bellman equation and stochastic value iteration algorithm. Moreover, the influence of the parameters $\Delta t, \gamma$, and σ will be analyzed with regard to the number of iterations, optimal policy, and the value functions.

First, consider the stepsizes $\Delta t \in [2^{-10}, 2^0]$ and put

$$\epsilon = 0.1,\ \gamma = 0.9,\ \sigma_1 = 0.5,\ \sigma_2 = 0.2.$$

The stochastic value iteration algorithm with additive noise terminates for the stepsizes $\Delta t < 2^{-2}$ in 94 steps. However, for $\Delta t \geq 2^{-2}$ it fails to converge within 1000 steps. On the other hand, the algorithm driven by multiplicative noise stops for $\Delta t < 2^{-8}$ in 93 steps and increases only slightly to 94 for $2^{-8} \leq \Delta t < 2^{-3}$. For $\Delta t \geq 2^{-3}$ it does not terminate within 1000 steps. For example, considering $\Delta t = 2^{-4}$ the optimal value function in the case of additive noise is

$$V^* = \begin{bmatrix} -13.87 \\ -25.77 \\ -25.76 \end{bmatrix} \quad \text{with } \pi^* = (1, 2, 1)$$

and in the case of multiplicative noise

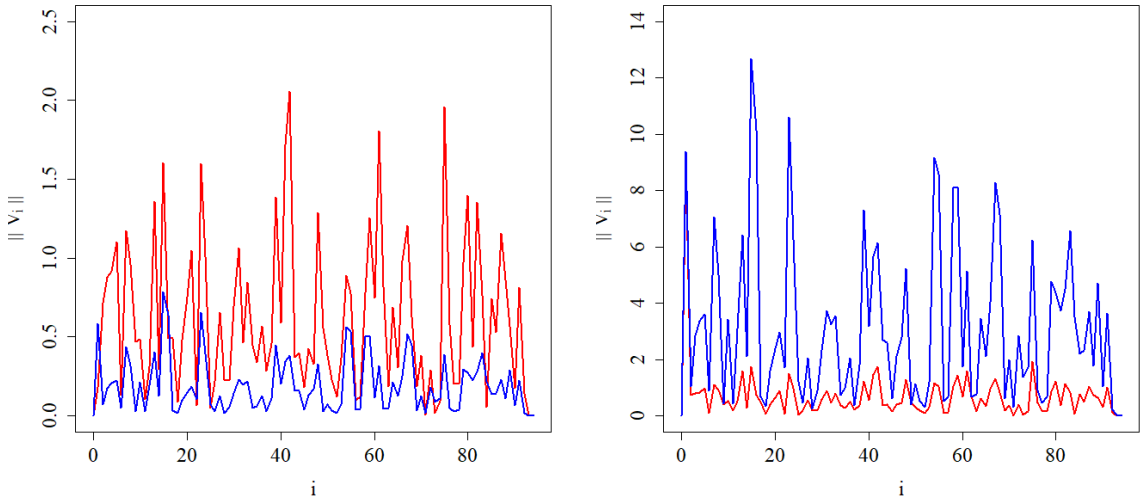
$$V^* = \begin{bmatrix} -14.07 \\ -25.92 \\ -25.92 \end{bmatrix} \quad \text{with } \pi^* = (1, 2, 2).$$

Both value functions are rather close together. The action performed in the last state makes no difference, since the maximum age is reached, and the machine must be replaced. The deterministic optimal value function is

$$V^* = \begin{bmatrix} -13.88 \\ -25.78 \\ -25.78 \end{bmatrix} \quad \text{with } \pi^* = (1, 2, 1).$$

Thus, the stochastic value iteration algorithm with additive noise performs better than with multiplicative noise in terms of approximating the value function.

The value functions per iteration for both noise models are depicted in Fig. 3.19. In sub-figure (a), the results for the smallest stepsize $\Delta t = 2^{-10}$ are shown and in sub-figure (b) for the stepsize $\Delta t = 2^{-6}$. In the case of additive noise, the value function remains similar for both stepsizes. However, due to Thm. 3.3.8, the value function in the multiplicative noise model increases with increasing stepsizes and thus deviates further from the deterministic case. The norm $\|V\|$ with $\Delta t = 2^{-10}$ is smaller than $n = 3$ and with $\Delta t = 2^{-6}$ it is larger. Consequently, in sub-figure (a), the value function



(a) (b)

Figure 3.19: Value functions for different stepsizes $\Delta t = 2^{-10}$ (a) and $\Delta t = 2^{-6}$ (b) with additive noise (red line) and multiplicative noise (blue line).

with multiplicative noise is smaller than that with additive noise. In sub-figure (b), the reverse situation occurs. Nonetheless, both noise models terminate with similar value functions. Overall, these findings provide valuable insights into the dynamics of the algorithm under different noise models and parameters, guiding the selection of appropriate parameters for effectively solving similar decision problems.

The value functions depicted in Fig. 3.20 correspond to the case of multiplicative noise, with stepsizes $\Delta t \in \{2^{-10}, 2^{-8}, 2^{-6}\}$. Smaller stepsizes correspond to smaller diffusion terms. As already mentioned, the algorithm with multiplicative noise does not stop for $\Delta t \geq 2^{-3}$ within 1000 steps. Therefore, in the following, stepsizes below 0.1 are considered, where both algorithms terminate. The initial value function is set to $V_0 = (1, 1, 1)^T$.

Second, consider the discount factors $\gamma \in \{0.1, 0, 2, \ldots, 0.9\}$ to examine the effect, with the following parameters:

$$\epsilon = 0.1, \ \Delta t = 0.01, \ \sigma_1 = 0.5, \ \sigma_2 = 0.2.$$

The algorithm takes longer to converge with larger discount factors. The number of iterations for different discount factors are illustrated in Fig. 3.21. For $\gamma \in [0.5, 0.8]$, the number of iterations for the stochastic differential Bellman equation with multiplicative noise is slightly higher than that with additive noise. Additionally, the value function tends to increase with higher discount factors. In the case of multiplicative noise the value function appears in the diffusion term such that the influence of noise is larger.

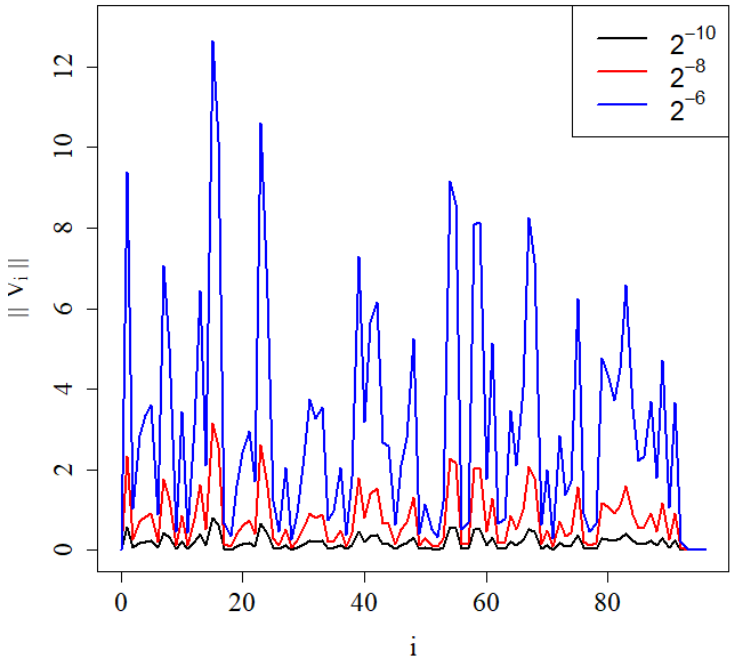


Figure 3.20: Approximations of the SBE with multiplicative noise for different stepsizes.

Therefore, it takes longer for the algorithm to converge until the difference between two successive value functions becomes sufficiently small. Two different optimal policies are reached depending on the discount factor, either $\pi^* = (1, 2, 1)$ or $\pi^* = (1, 2, 2)$. In the case of multiplicative noise, the optimal policy is $\pi^* = (1, 2, 2)$ for $\gamma \geq 0.4$ and $\pi^* = (1, 2, 1)$ for $\gamma < 0.4$. In this example, both policies are the same, as the machine is always replaced in the last state.

In order to analyze the influence of the noise intensity, the optimal value functions are compared for $\sigma_1 \in \{0, 0.1, 0.2, \ldots, 1\}$. Put

$$\gamma = 0.9,\ \Delta t = 0.01,\ \epsilon = 0.1,\ \sigma_2 = 0.5.$$

The number of iterations remains the same in both noise models. The results are shown in Fig. 3.22. Interestingly, the optimal value functions change only slightly depending on the σ_1-value. With increasing intensity of the multiplicative noise for action a_1, the values $\|V^*\|$ increase. The value function with multiplicative noise is greater than that with additive noise. In the deterministic case, the optimal value function is $V^*_{\text{det}} = 6.03$. The additive noise model achieves a *better* approximation in this regard. For $\sigma_1 < 0.5$, the optimal policy in the stochastic differential Bellman equation with additive noise is $\pi^* = (1, 2, 2)$, while for $\sigma_1 \geq 0.5$, it becomes $\pi^* = (1, 2, 1)$. Conversely, in the case of multiplicative noise, the optimal policy $\pi^* = (1, 2, 1)$ is obtained for $\sigma_1 \leq 0.5$, and $\pi^* = (1, 2, 2)$ for $\sigma_1 > 0.5$. Since both policies are the same, there is no dependence on

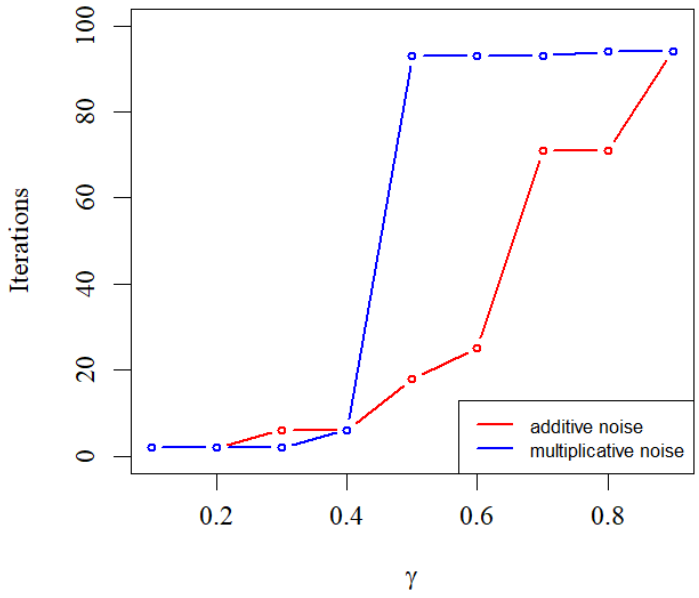


Figure 3.21: Number of iterations for different discount factors with additive noise and multiplicative noise.

the noise intensity.

Similarly, the intensity σ_2 is analyzed in the same way. The results exhibit a pattern akin to the analysis of σ_1. For the stochastic value iteration algorithm with multiplicative noise, the values $\|V^*\|$ increase with increasing noise intensity for the action a_2.

In summary, the stochastic differential Bellman equation driven by multiplicative noise is more affected by noise and the value function is greater than that with additive noise, as expected in Thm. 3.3.8 ($n = 3$). For stepsize $\Delta t > 0.1$ the algorithm with multiplicative noise does not terminate within 1000 steps. Regardless of the chosen parameters, the optimal policy proposes that the machine should be repaired in the first year of life and replaced in the second year. At the age 2 (state s_2), the machine is always replaced.

Applying the ILS, the best parameter setting $\theta^* = (\gamma, \Delta t, \sigma_1, \sigma_2)$ for the stochastic value iteration with additive noise is

$$\theta = (0.8, 0.0009, 0.50, 0.90).$$

The algorithm terminates with these parameters after 71 iterations, resulting in the optimal value function $V^* = (-0.35, -0.58, -0.53)^T$ and the optimal policy $\pi^* = (1, 1, 1)$.

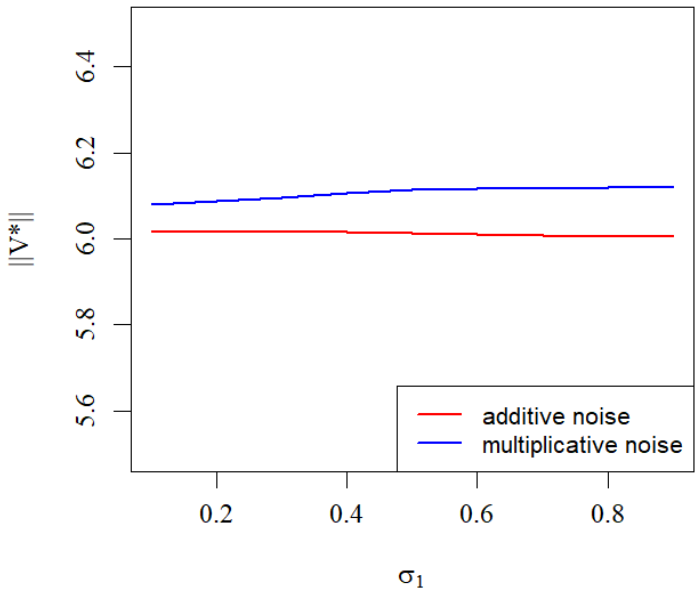


Figure 3.22: Optimal value functions due to noise of action a_1 with additive noise and multiplicative noise.

For the multiplicative noise, the best parameter setting using ILS is

$$\theta = (0.8, 0.0009, 0.60, 0.70).$$

Remarkably, the algorithm terminates with these parameters already after 17 steps with $V^* = (-0.21, -0.44, -0.42)^T$ and more sensible optimal policy $\pi^* = (1, 2, 1)$. The discount factor and the stepsize are the same in both noise models, but both models yield different optimal policies. This highlights the significant impact of noise characteristics on decision-making processes. In the deterministic case, the best parameter setting is $\theta = (0.3, 0.0009)$ and the algorithm terminates after only two steps with $V^* = (-0.18, -0.36, -0.36)^T$ and $\pi^* = (1, 2, 1)$. This demonstrates once again the influence of the discount factor on algorithm efficiency.

Another optimal parameter setting for the stochastic value iteration with additive noise found by the ILS is

$$\theta = (0.4, 0.0009, 0.36, 0.84)$$

and for the case of multiplicative noise is

$$\theta = (0.3, 0.0009, 0.92, 0.89).$$

So the intensity of the multiplicative noise can be larger than that for additive noise if the discount factor is small. Despite this difference, the optimal policy obtained using

these parameters is $\pi^* = (1, 2, 2)$ for both noise models. In this decision problem, the performance in terms of the number of iterations is not better than in the deterministic case. Nevertheless, this method remains valuable because it offers the possibility of minimizing the cost in a stochastic model by accounting unpredictable disturbances.

3.5.2 Inventory Problem

Consider the decision problem of a store that has to order the right number of units each month in align with customer demand. If the shop orders too much, money will be wasted. On the other hand, if the order quantity is not sufficient, the shop may lose customers. Thus, determining the optimal number of order quantity is crucial for the success of the store. This section describes an inventory management problem inspired by Puterman [39], but in a discounted version. Four algorithms are applied to address this problem: value iteration, stochastic value iteration, Q-learning and stochastic Q-learning.

Let $t = 1, 2, \ldots, N$ denote the months. The inventory has capacity M and the states represent the stock number s in the state set $S = \{0, 1, \ldots, M\}$. The actions correspond to the additional stock to be ordered each month. For a given state s, the action set is $A = \{0, 1, \ldots, M - s\}$. When the stock quantity is s, the cost of ordering new stock a is calculated as:

$$O(a) = \begin{cases} K + c(a), & a > 0, \\ 0, & a = 0, \end{cases} \tag{3.5.1}$$

where K denotes the fixed cost and $c(a)$ the variable cost. The reward is the expected income $F(s + a)$ minus the ordering cost $O(a)$ and holding cost $h(s + a)$,

$$\begin{aligned} r_t(s, a) &= F(s + a) - O(a) - h(s + a), \quad t > 1, \\ r_1(s, a) &= g(s), \end{aligned}$$

where $g(s)$ is the salvage value. Let D_t denote the random demand in month t with a known probability distribution $p_j = P(D_t = j)$, $j \in \mathbb{N}_0$. If j units are demanded and the inventory exceeds the demand, the revenue is $f(j)$, if the demand exceeds the inventory, the revenue is $f(a)$. Then the expected present value of the inventory is calculated as

$$F(a) = \sum_{j=0}^{a-1} f(j)p_j + f(a)P(D_t \geq a), \tag{3.5.2}$$

where

$$P(D_t \geq a) = \sum_{d=a}^{\infty} p_d. \tag{3.5.3}$$

The transition probabilities are defined as

$$P_t(j|s, a) = \begin{cases} 0, & j > s + a, \\ p_{s+a-j}, & s + a \geq j > 0, \\ P(D_t \geq s + a), & j = 0. \end{cases} \tag{3.5.4}$$

The probability is zero for $j > s + a$, because the number of sales cannot exceed the stock. The probability that the demand exceeds $s+a$ units is denoted by $P(D_t \geq s + a)$.

In order to keep the structure of the value iteration algorithm as in Eq. (2.2.9) with increasing time t, the results in this example are interpreted backwards in time. For example, the policy $\pi_N(0) = 3$ means that if at the beginning of the first month the inventory stock is zero, then three items will be ordered.

Solution of the Inventory Problem via Value Iteration For $N = 3$ months, let the inventory capacity be $M = 3$ and the fixed cost $K = 4$. The variable cost for a given order quantity a is calculated as $c(a) = 2a$, and the holding cost is $h(a) = a$. The salvage cost is $g(a) = 0$, and the income from a sales is $f(a) = 8a$. The probabilities are given by

$$p_d = \begin{cases} 0.25, & d = 0, \\ 0.5, & d = 1, \\ 0.25, & d = 2. \end{cases}$$

The expected present values are calculated as

$$\begin{aligned} F(0) &= 0, \\ F(1) &= 0 \cdot 0.25 + 8 \cdot 0.75 = 6, \\ F(2) &= 0 \cdot 0.25 + 8 \cdot 0.5 + 16 \cdot 0.25 = 8, \\ F(3) &= 0 \cdot 0.25 + 8 \cdot 0.5 + 16 \cdot 0.25 = 8. \end{aligned}$$

The reward functions are $R_0 = (0, 5, 6, 5)^{\mathrm{T}}$, $R_1 = (-1, 0, -1, *)^{\mathrm{T}}$, $R_2 = (-2, -3, *, *)^{\mathrm{T}}$, and $R_3 = (-5, *, *, *)^{\mathrm{T}}$, where $*$ denotes an infeasible action. The transition probabilities are given as

	$p_t(j\|s,a)$			
$s+a$	$j=0$	$j=1$	$j=2$	$j=3$
0	1	0	0	0
1	0.75	0.25	0	0
2	0.25	0.5	0.25	0
3	0	0.25	0.5	0.25

These probabilities can be defined as probability matrices P_j depending on the order quantity as

$$P_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0.75 & 0.25 & 0 & 0 \\ 0.25 & 0.5 & 0.25 & 0 \\ 0 & 0.25 & 0.5 & 0.25 \end{bmatrix}, \quad P_1 = \begin{bmatrix} 0.75 & 0.25 & 0 & 0 \\ 0.25 & 0.5 & 0.25 & 0 \\ 0 & 0.25 & 0.5 & 0.25 \\ * & * & * & * \end{bmatrix},$$

$$P_2 = \begin{bmatrix} 0.25 & 0.5 & 0.25 & 0 \\ 0 & 0.25 & 0.5 & 0.25 \\ * & * & * & * \\ * & * & * & * \end{bmatrix}, \quad P_3 = \begin{bmatrix} 0 & 0.25 & 0.5 & 0.25 \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{bmatrix}.$$

Let $\gamma = 0.9$ be the discount factor. The algorithm starts at $t = 1$ with $V_0 = (0, 0, 0, 0)^{\mathrm{T}}$. The first value function yields the maximum of rewards, since the initial value function is zero, that is, $V_1 = (0, 5, 6, 5)^{\mathrm{T}}$ with policy $\pi_1 = (0, 0, 0, 0)$. Then $V_2 = (1.60, 6.12, 9.60, 9.95)^{\mathrm{T}}$ and $V_3 = (3.27, 7.45, 11.27, 12.93)^{\mathrm{T}}$, both with the same policy $\pi = (2, 0, 0, 0)$. These policies are interpreted as follows. If the inventory is zero at the beginning of the first and second month, order two units, otherwise do not order. Order in the last month for no stock number.

If the algorithm runs until the difference between the value functions is small enough, e.g. for $\epsilon = 0.1$, the algorithm terminates after 63 steps with the value $V^* = (17.50, 21.69, 25.41, 27.50)^{\mathrm{T}}$ with optimal policy $\pi^* = (3, 0, 0, 0)$. The optimal policy is to order three units if the stock number is zero at the first month, i.e. full capacity is used.

Solution via Q-learning In addition to the value iteration algorithm, the Q-learning algorithm can also be used to approximate a solution to this problem. For this, set the learning rate to $\alpha = 0.1$. Initialize the Q-function to zero with the last row equal to the corresponding rewards. Then the solution of the inventory problem via Q-learning algorithm after 1000 episodes is given by the Q-function

$$Q = \begin{bmatrix} 35.60 & 35.81 & 40.27 & -0.50 \\ 40.81 & 41.53 & 1.49 & 4.49 \\ 47.66 & 3.49 & 4.49 & 4.49 \\ 5.00 & 0.00 & 0.00 & 0.00 \end{bmatrix}$$

and the optimal policy is $\pi^* = (2, 1, 0, 0)$. Therefore, in the first month two units should be ordered if the stock number is zero, and one unit if the stock number is one.

The value function tends to be higher in Q-learning compared to the value iteration algorithm. In the following, the inventory problem subject to noise is solved.

Solution via Stochastic Value Iteration Consider the inventory problem with noise and approximate the solution via the stochastic value iteration using the modified Euler-Maruyama method (see Sect. 3.2.3). Assume that noise is uniform across all actions, $\sigma_0 = \sigma_1 = \sigma_2 = \sigma_3 = 0.1$. With the same conditions and constraints as in the above calculation for the value iteration algorithm and $\Delta t = 1$, the value function for additive noise is computed as

$$V_1 = \max_{a \in A} \{(R_a + 0.9 \cdot P_a V_0) \cdot 1 + 0.9 \cdot 0.1 \cdot (B_{t_1} - B_{t_0})\} = (0.19, 5.19, 6.19, 5.19)^T$$

In the case of multiplicative noise, the value function V_1 is not affected, since V_0 is zero. Then the value function is the maximum of rewards, i.e. $V_1 = (0, 5, 6, 5)^T$.

If the additive noise algorithm runs for twelve months, the following policy is obtained:

- for the first nine months order three units if the stock number is zero,
- if in the 10th and 11th months the stock number is zero, then order two units,
- do not order anything in the 12th month.

In the deterministic case, if the value iteration algorithm runs for 12 months, the same optimal policy is obtained.

In the multiplicative noise model, the optimal policy is slightly different:

- in the 3rd, 6th, and 9th month order three units if the stock number is zero,
- in remaining months except the last month, order two units if the stock number is zero
- do not order anything in the 12th month.

Thus, the approach is similar to that with additive noise, but tends to order fewer items to avoid wasting money.

Let the discount factor be $\gamma = 0.9$ and the stepsize $\Delta t = 1$. If the stochastic value iteration runs until the differences between the successive value functions is less than $\epsilon = 0.1$, the algorithm driven by additive noise terminates after 43 steps with optimal policy $\pi^* = (3, 0, 0, 0)$. The value iteration algorithm (without noise) terminates after 63 steps, so with additive noise it requires *fewer* steps to terminate. The algorithm with multiplicative noise terminates with the optimal policy $\pi^* = (2, 0, 0, 0)$, but it converges for $\gamma \leq 0.7$ within at most 95 steps. For $\sigma_0 = 0.05$, $\sigma_1 = \sigma_2 = 0.1$, and $\sigma_3 = 0.003$, it also terminates for $\gamma = 0.8$ but with a significantly higher number of iterations, at least 300. In this example, the algorithm with multiplicative noise takes longer than that with additive noise.

One explanation for this could be that multiplicative noise tends to amplify the effects of uncertainty in the system. Since multiplicative noise is proportional to the previous value, small fluctuations can have a magnified impact on the overall outcome. This increased value can make it more challenging for the algorithm to converge to a stable solution, requiring more iterations to effectively navigate in uncertain environments.

Solution via Stochastic Q-learning The stochastic Q-learning algorithm is applied to the inventory problem with noise. Setting $\sigma_0 = \sigma_1 = \sigma_2 = \sigma_3 = 0.1$, the resulting Q-function after 1000 episodes in the case of additive noise is

$$Q = \begin{bmatrix} 37.49 & 36.68 & 41.83 & -4.95 \\ 41.68 & 42.32 & -3.01 & 0.01 \\ 49.04 & -0.97 & 0.01 & 0.03 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

with the optimal policy $\pi^* = (2, 1, 0, 0)$. Executing the algorithm with different noise intensities such as $\sigma_0 = 0.05$, $\sigma_1 = \sigma_2 = 0.1$ and $\sigma_3 = 0.003$ leads to the policy $\pi^* = (2, 0, 0, 0)$. In the multiplicative noise model, the resulting Q-function after 1000 episodes is

$$Q = \begin{bmatrix} 33.38 & 33.28 & 36.13 & -5.18 \\ 37.18 & 35.37 & -3.07 & 0 \\ 40.64 & -1.02 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

with the optimal policy $\pi^* = (2, 0, 0, 0)$. Different runs result in the optimal policy $\pi^* = (2, 1, 0, 0)$ here as well.

In terms of rewards, the cumulative rewards per episode are *larger* when using stochastic Q-learning with noise compared to that without noise, which is an important point when evaluating the algorithm in terms of performance. The case of additive noise and without noise is illustrated in Fig. 3.23.

```
# Maximum and Mean of rewards in case of additive noise
> max(inventory_add)
[1] 23
> mean(inventory_add)
[1] -1.02
# Maximum and Mean of rewards in case of multip. noise
> max(inventory_mult)
[1] 21
> mean(inventory_mult)
[1] -1.775
# Maximum and Mean of rewards without noise
> max(inventory_det)
[1] 16
> mean(inventory_det)
[1] -1.82
```

Overall, adding noise can assist the stochastic Q-learning algorithm in finding better solutions by enhancing exploration and increasing robustness to uncertainty. This can lead to higher cumulative rewards per episode.

Conclusion Stochastic algorithms such as stochastic value iteration and stochastic Q-learning demonstrate robustness and adaptability in uncertain environments, making them well-suited for real-world decision-making problems characterized by uncertainty and noise. In the deterministic case, the solution via value iteration algorithm tends to utilize inventory capacity more efficiently than Q-learning. In the stochastic case, both algorithms, stochastic value iteration and stochastic Q-learning, showcase similar performance. The only difference is that the optimal policy in stochastic value iteration utilizes the full capacity.

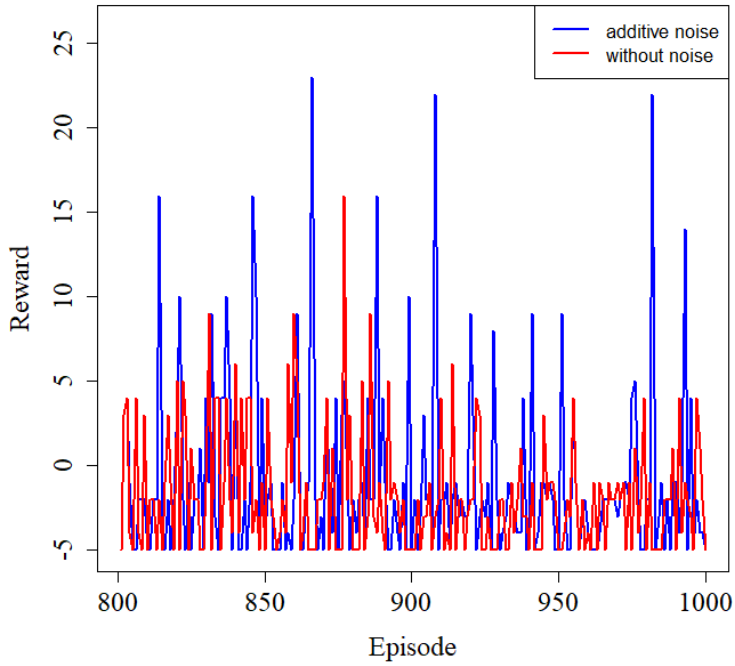


Figure 3.23: Rewards per episode in stochastic Q-learning with additive noise and without any noise.

Comparing deterministic methods with stochastic approaches, the latter tend to yield superior results. Stochastic value iteration proves to be more efficient than value iteration. Furthermore, in this decision problem, it is observed that the algorithm with additive noise converges to a solution in significantly fewer steps compared to that with multiplicative noise. This can be attributed to the larger stepsize utilized, leading to faster growth of the diffusion term in the multiplicative noise model. Furthermore, introducing noise, whether additive or multiplicative, enhances the performance of stochastic Q-learning by promoting exploration, resulting in higher cumulative rewards per episode compared to deterministic methods. Despite variations in noise intensities, the optimal policies derived from stochastic algorithms remain consistent across different runs. This indicates the reliability and effectiveness of these algorithms in finding optimal solutions.

In conclusion, both algorithms, stochastic value iteration and stochastic Q-learning, can be used to successfully tackle the inventory problem in noisy environments.

3.6 Random Dynamical Systems

In this section, the concept of random dynamical systems (RDS) is briefly studied in the context of stochastic differential equations. *Random differential equations* (RDEs) are ordinary differential equations with random coefficients and they generate random dynamical systems. A stochastic differential equation can be transformed into a system of random differential equations. From the solution of the random differential equation, the solution of the corresponding stochastic differential equation can be obtained. This procedure is applied to the stochastic differential Bellman equation such that the equation can be analyzed as an ordinary differential equation. Fundamental concepts such as Wiener shift and cocycle property are introduced. Additionally, solution mappings for stochastic differential Bellman equations with additive and multiplicative noise are presented, along with their cocycle properties.

Random dynamical systems are based on a specific probability space, known as the canonical probability space. Let $C(\mathbb{R}, \mathbb{R}^n)$ denote the space of all continuous real-valued functions on $\mathbb{R}$. Consider two-sided Brownian motion in $\mathbb{R}^n$ as a random vector taking values in $C(\mathbb{R}, \mathbb{R}^n)$,

$$B : \Omega \to C(\mathbb{R}, \mathbb{R}^n) \ : \ \omega \mapsto \omega(t), \quad t \in \mathbb{R}. \tag{3.6.1}$$

Let $\mathcal{B}(C(\mathbb{R}, \mathbb{R}^n))$ denote the Borel σ-field on $C(\mathbb{R}, \mathbb{R}^n)$. This Borel σ-field is generated by all open sets of $C(\mathbb{R}, \mathbb{R}^n)$ constructed by taking countable unions. A probability measure can be defined on the Borel σ-field which is called the *Wiener measure*.

Definition 3.6.1 (Canonical probability space)**.** *Let $\mathcal{B}(C(\mathbb{R}, \mathbb{R}^n))$ denote the Borel σ-field and $\mathbb{P}_B$ the Wiener measure. Then the* canonical probability space *for B_t or for an SDE system driven by B_t is given as $(C(\mathbb{R}, \mathbb{R}^n), \mathcal{B}(\mathbb{R}, \mathbb{R}^n), \mathbb{P}_B)$.*

Denote the canonical probability space by $(\Omega, \mathcal{F}, \mathbb{P})$. The samples of the canonical sample space Ω are paths of Brownian motion such that the increments $\omega(t+s) - \omega(t)$ can be analyzed, resulting in a measurable dynamical system in Ω, called the Wiener shift.

Definition 3.6.2 (Wiener shift)**.** *The* Wiener shift *θ_t is a mapping in the canonical sample space Ω for each $t \in \mathbb{R}$,*

$$\theta_t : \Omega \to \Omega \ : \ \omega \mapsto \hat{\omega},$$

where $\theta_t\omega(s) = \hat{\omega}(s)$ and $\hat{\omega}(s) = \omega(t+s) - \omega(t)$ for all $s, t \in \mathbb{R}$.

The property obtained from the above definition is expressed as follows:

$$B_s(\theta_t\omega) = B_{t+s}(\omega) - B_t(\omega). \tag{3.6.2}$$

When s is small approaching zero, the right-hand side can be interpreted as $dB_t(\omega)$ and so θ_t can be considered as noise in a stochastic differential equation.

Proposition 3.6.1. *The Wiener shift θ_t has following properties*

i) $\theta_0 = \mathrm{Id}$,

ii) $\theta_s \theta_t = \theta_{s+t}$ *for all* $s, t \in \mathbb{R}$,

iii) the mapping $(t, \omega) \mapsto \theta_t \omega$ *is measurable and* $\theta_t \mathbb{P} = \mathbb{P}$ *for all* $t \in \mathbb{R}$, *where* $\mathbb{P}$ *is the Wiener measure.*

Definition 3.6.3 (Random Dynamical System (RDS))**.** *A* random dynamical system *in the state space* $X = \mathbb{R}^n$ *with the time set* $\mathbb{T}$ *and probability space* $(\Omega, \mathcal{F}, \mathbb{P})$ *consists of*

- *a Wiener shift* $(\theta_t)_{t \in \mathbb{T}}$ *on* Ω *such that* $(t, \omega) \mapsto \theta_t \omega$ *is a measurable flow that leaves* $\mathbb{P}$ *invariant, i.e.* $\theta_t \mathbb{P} = \mathbb{P}$ *for all* $t \in \mathbb{T}$,

- *a cocycle* φ *over* θ_t, *i.e. a measurable mapping*

$$\varphi : \mathbb{T} \times \Omega \times X \to X : (t, \omega, x) \mapsto \varphi(t, \omega, x)$$

such that $(t, x) \mapsto \varphi(t, \omega, x)$ *is continuous for all* $\omega \in \Omega$ *and the family* $\varphi(t, \omega, \cdot) = \varphi(t, \omega) : X \to X$ *of random mappings has the* cocycle property

$$\varphi(0, \omega) = \mathrm{Id}_X, \tag{3.6.3}$$
$$\varphi(t + s, \omega, x) = \varphi(t, \theta_s \omega, \varphi(s, \omega, x)) \tag{3.6.4}$$

for all $t, s \in \mathbb{T}, x \in X$ *and* $\omega \in \Omega$. Id_X *is the identity mapping on the state space* X *and* φ *is called a RDS over* θ_t *or simply a cocycle.*

In the following, these theoretical concepts will be applied to solve Markov decision processes in the presence of noise. Specifically, which is modeled by the stochastic differential Bellman equation. This approach expands the applicability of Random dynamical systems to real-world decision problems. The cocycle property can be shown for the stochastic differential Bellman equation with given policy π. By replacing $\gamma \sigma_\pi$ with a constant $c > 0$, the solution mapping for the SBE with additive noise due to Thm. 3.1.1 is given by

$$\varphi(t, \omega, V) = \mathrm{e}^{\gamma t} V + \frac{\mathrm{e}^{\gamma t} - 1}{\gamma} R_\pi + c \int_0^t \mathrm{e}^{\gamma (t-u)} \, dB_u(\omega). \tag{3.6.5}$$

For the SBE with multiplicative noise, the solution mapping is given as

$$\varphi(t, \omega, V) = \mathrm{e}^{\left(\gamma - \frac{c^2}{2}\right) t + c B_t} \left[V + R_\pi \int_0^t \mathrm{e}^{\left(\frac{c^2}{2} - \gamma\right) u - c B_u} \, du \right]. \tag{3.6.6}$$

Theorem 3.6.1. *The solution mapping for the SBE with additive noise has the cocycle property.*

Proof. Clearly $\varphi(0,\omega,V) = V$, since $\mathrm{e}^0 = 1$ and $B_0 = 0$. Furthermore,

$$\varphi(t+s,\omega,V) = \mathrm{e}^{\gamma(t+s)}\,V + \frac{\mathrm{e}^{\gamma(t+s)}-1}{\gamma}R_\pi + c\int_0^{t+s}\mathrm{e}^{\gamma(t+s-u)}\,dB_u(\omega).$$

On the other hand,

$$\begin{aligned}
&\varphi(t,\theta_s\omega,\varphi(s,\omega,V))\\
&= \mathrm{e}^{\gamma t}\,\varphi(s,\omega,V) + \frac{\mathrm{e}^{\gamma t}-1}{\gamma}R_\pi + c\int_0^t \mathrm{e}^{\gamma(t-u)}\,dB_u(\theta_s\omega)\\
&= \mathrm{e}^{\gamma t}\left[\mathrm{e}^{\gamma s}\,V + \frac{\mathrm{e}^{\gamma s}-1}{\gamma}R_\pi + c\int_0^s \mathrm{e}^{\gamma(s-u)}\,dB_u(\omega)\right] + \frac{\mathrm{e}^{\gamma t}-1}{\gamma}R_\pi + c\int_0^t \mathrm{e}^{\gamma(t-u)}\,dB_u(\theta_s\omega)\\
&= \left[\mathrm{e}^{\gamma(t+s)}\,V + \frac{\mathrm{e}^{\gamma(t+s)}-1}{\gamma}R_\pi + c\int_0^s \mathrm{e}^{\gamma(t+s-u)}\,dB_u(\omega)\right] + c\int_0^t \mathrm{e}^{\gamma(t-u)}\,dB_u(\theta_s\omega).
\end{aligned}$$

It holds

$$c\int_0^t \mathrm{e}^{\gamma(t-u)}\,dB_u(\theta_s\omega) = c\int_s^{t+s}\mathrm{e}^{\gamma(t+s-u)}\,dB_u(\omega),$$

since $dB_u(\theta_s\omega) = d(B_{s+u} - B_s)$. Thus

$$c\int_0^s \mathrm{e}^{\gamma(t+s-u)}\,dB_u(\omega) + c\int_s^{t+s}\mathrm{e}^{\gamma(t+s-u)}\,dB_u(\omega) = c\int_0^{t+s}\mathrm{e}^{\gamma(t+s-u)}\,dB_u(\omega),$$

and hence $\varphi(t+s,\omega,V) = \varphi(t,\theta_s\omega,\varphi(s,\omega,V))$. So $\varphi(t,\omega,V)$ is a cocycle.

□

Theorem 3.6.2. *The solution mapping for the SBE with multiplicative noise has the cocycle property.*

Proof. The proof is analogous to that of the additive noise model. It holds $\varphi(0,\omega,V) = V$ and

$$\varphi(t+s,\omega,V) = \mathrm{e}^{\left(\gamma-\frac{c^2}{2}\right)(t+s)+cB_{t+s}}\left[V + R_\pi\int_0^{t+s}\mathrm{e}^{\left(\frac{c^2}{2}-\gamma\right)u-cB_u}\,du\right].$$

Furthermore,

$$\begin{aligned}
\varphi(t,\theta_s\omega,\varphi(s,\omega,V)) &= \mathrm{e}^{\left(\gamma-\frac{c^2}{2}\right)t+cB_t(\theta_s\omega)}\left[\varphi(s,\omega,V) + R_\pi\int_0^t \mathrm{e}^{\left(\frac{c^2}{2}-\gamma\right)u-cB_u(\theta_s\omega)}\,du\right]\\
&= \mathrm{e}^{\left(\gamma-\frac{c^2}{2}\right)t+cB_t(\theta_s\omega)}\,\mathrm{e}^{\left(\gamma-\frac{c^2}{2}\right)s+cB_s}\left(V + R_\pi\int_0^s \mathrm{e}^{\left(\frac{c^2}{2}-\gamma\right)u-cB_u}\,du\right)\\
&\quad + \mathrm{e}^{\left(\gamma-\frac{c^2}{2}\right)t+cB_t(\theta_s\omega)}\,R_\pi\int_0^t \mathrm{e}^{\left(\frac{c^2}{2}-\gamma\right)u-cB_u(\theta_s\omega)}\,du\\
&= \mathrm{e}^{\left(\gamma-\frac{c^2}{2}\right)(t+s)+cB_{t+s}(\omega)}\left(V + R_\pi\int_0^s \mathrm{e}^{\left(\frac{c^2}{2}-\gamma\right)u-cB_u}\,du\right)\\
&\quad + \mathrm{e}^{\left(\gamma-\frac{c^2}{2}\right)(t+s)+cB_{t+s}(\omega)}\,R_\pi\int_0^t \mathrm{e}^{\left(\frac{c^2}{2}-\gamma\right)(u+s)-cB_{u+s}(\omega)}\,du,
\end{aligned}$$

since due to (3.6.2) $B_t(\theta_s\omega) = B_{t+s}(\omega) - B_s(\omega)$. The last integral in the above equation yields

$$\int_0^t \mathrm{e}^{\left(\frac{c^2}{2}-\gamma\right)(u+s)-cB_{u+s}(\omega)}\,du = \int_s^{t+s} \mathrm{e}^{\left(\frac{c^2}{2}-\gamma\right)u-cB_u}\,du,$$

and hence $\varphi(t,\omega,V)$ is a cocyle. □

For converting a stochastic differential equation to a random differential equation, a stochastic integrating factor is introduced, which is derived by taking the reciprocal of the solution for the corresponding stochastic differential equation without the drift term [12].

As previously mentioned, random differential equations include stochastic processes like Brownian motion. An example of a scalar random differential equation is given by $dX_t = -X + \sin(B_t)dt$, where B_t is scalar Brownian motion [19]. A random differential equation therefore does not require any stochastic calculus and can be computed pathwise deterministically using standard calculus. In the following, the conversion of an stochastic differential equation to a random differential equation will be explored. Recall the stochastic product rule (2.3.38), derived by applying Itô's formula to the function $g(x,y) = xy$,

$$d(X_tY_t) = X_tdY_t + Y_tdX_t + dX_tdY_t. \tag{3.6.7}$$

Definition 3.6.4 (RDE, additive noise)**.** *For a scalar SDE with additive noise*

$$dX_t = b(t,X_t)dt + \sigma(t)dB_t, \quad X_0 = x, \tag{3.6.8}$$

the stochastic integrating factor is

$$F_t = \int_0^t \sigma(s)dB_s. \tag{3.6.9}$$

If the noise is constant $\sigma(t) = \sigma$, the integrating factor simplifies to $F_t = \sigma B_t$. Define $Y_t = X_t - F_t$ and so $X_t = Y_t + F_t$. Then the stochastic differential equation can be rewritten as a random differential equation

$$\frac{dY_t}{dt} = b(t, Y_t + F_t). \tag{3.6.10}$$

Definition 3.6.5 (RDE, multiplicative noise)**.** *For a scalar SDE with multiplicative noise*

$$dX_t = b(t,X_t)dt + \sigma(t)X_tdB_t, \quad X_0 = x, \tag{3.6.11}$$

the stochastic integrating factor becomes

$$F_t(\omega) = \exp\left(\frac{1}{2}\int_0^t \sigma^2(s)ds - \int_0^t \sigma(s)dB_s\right). \tag{3.6.12}$$

Multiplying the SDE by the stochastic integrating factor F_t and applying the stochastic product rule, it yields

$$d(F_t X_t) = F_t b(t, X_t)dt, \tag{3.6.13}$$

since $dF_t = \sigma^2(t)F_t dt - \sigma(t)F_t dB_t$ by using Itô's formula. Define $Y_t(\omega) = F_t(\omega)X_t(\omega)$, so $X_t = F_t^{-1}Y_t$. Then the stochastic differential equation can be converted to the following random differential equation

$$\frac{dY_t}{dt} = F_t b(t, F_t^{-1}Y_t), \quad Y_0 = x. \tag{3.6.14}$$

The solution Y_t of this RDE provides the solution of the original SDE using $X_t = F_t^{-1}Y_t$.

In the following theorems, the conversion of the stochastic differential Bellman equation into a random differential equation is demonstrated.

Theorem 3.6.3. *Consider the stochastic differential Bellman equation driven by additive noise*

$$dV_t = (R_\pi + \gamma P_\pi V_{t-1})\, dt + \gamma\sqrt{\epsilon}dB_t. \tag{3.6.15}$$

The stochastic integrating factor is

$$F_t = \gamma\sqrt{\epsilon}B_t. \tag{3.6.16}$$

For $V_t = Y_{t-1} + F_t$, the stochastic differential equation is converted to the random differential equation

$$\dot{Y}_t = R_\pi + \gamma P_\pi(Y_{t-1} + F_t) = R_\pi + \gamma P_\pi\big(Y_{t-1} + \gamma\sqrt{\epsilon}B_t\big).$$

Proof. In view of $V_t = Y_{t-1} + F_t$,

$$\begin{aligned} dV_t &= dY_{t-1} + dF_t \\ &= R_\pi + \gamma P_\pi(Y_{t-1} + F_t) + \gamma\sqrt{\epsilon}dB_t \\ &= b(t, Y_{t-1} + F_t)dt + \sigma(t)dB_t \\ &= b(t, V_t)dt + \sigma(t)dB_t \\ &= R_\pi + \gamma P_\pi V_{t-1}dt + \gamma\sqrt{\epsilon}dB_t. \end{aligned}$$

□

Theorem 3.6.4. *For the stochastic differential Bellman equation driven by multiplicative noise*

$$dV_t = (R_\pi + \gamma P_\pi V_{t-1})\, dt + \gamma\sqrt{\epsilon}V_{t-1}dB_t \tag{3.6.17}$$

the stochastic integrating factor is

$$F_t = \exp\left(\frac{1}{2}\int_0^t \gamma^2\epsilon ds - \int_0^t \gamma\sqrt{\epsilon}dB_s\right) = \exp\left(\frac{1}{2}\gamma^2\epsilon t - \gamma\sqrt{\epsilon}B_t\right). \tag{3.6.18}$$

The stochastic differential Bellman equation is converted to the RDE

$$\dot{Y}_t = e^{\frac{1}{2}\gamma^2\epsilon t - \gamma\sqrt{\epsilon}B_t}R_\pi + \gamma P_\pi Y_{t-1}, \tag{3.6.19}$$

where $V_t = F_t^{-1}Y_{t-1}$.

Proof. The SBE driven by multiplicative noise can also be written as a system of scalar SDEs

$$\begin{aligned} dV_t(1) &= R_\pi(1) + \gamma p_\pi(1) V_{t-1} dt + \gamma\sqrt{\epsilon} V_{t-1}(1) dB_t \\ &\cdots \\ dV_t(n) &= R_\pi(n) + \gamma p_\pi(n) V_{t-1} dt + \gamma\sqrt{\epsilon} V_{t-1}(n) dB_t, \end{aligned}$$

where $p_\pi(i)$ denotes the i-th row of the probability matrix P_π. Then the stochastic integrating factor is

$$F_t = \exp\left(\frac{1}{2}\int_0^t \gamma^2 \epsilon ds - \int_0^t \gamma\sqrt{\epsilon} dB_s\right) = \exp\left(\frac{1}{2}\gamma^2\epsilon t - \gamma\sqrt{\epsilon} B_t\right).$$

These SDEs are then converted to the following system of RDEs

$$\begin{aligned} \dot{Y}_t(1) &= \mathrm{e}^{\frac{1}{2}\gamma^2\epsilon t - \gamma\sqrt{\epsilon}B_t} R_\pi(1) + \gamma p_\pi(1) Y_{t-1} \\ &\cdots \\ \dot{Y}_t(n) &= \mathrm{e}^{\frac{1}{2}\gamma^2\epsilon t - \gamma\sqrt{\epsilon}B_t} R_\pi(n) + \gamma p_\pi(n) Y_{t-1}, \end{aligned}$$

where $Y_t = F_t V_t$. □

To obtain the solution of the random differential equation and consequently that of the stochastic differential Bellman equation, the solution provided by Theorem 2.3.3 will be applied.

Theorem 3.6.5. *The solution of the scalar RDE with additive noise and $P_\pi = (1)$ is*

$$Y_t = \mathrm{e}^{\gamma t} Y_0 + \frac{\mathrm{e}^{\gamma t} - 1}{\gamma} R_\pi + \mathrm{e}^{\gamma t} \int_0^t \gamma^2 \sqrt{\epsilon} B_s \, \mathrm{e}^{-\gamma s} \, ds. \tag{3.6.20}$$

Then the solution of the stochastic differential Bellman equation with additive noise is given as

$$V_t = Y_{t-1} + F_t = \mathrm{e}^{\gamma t} Y_0 + \frac{\mathrm{e}^{\gamma t} - 1}{\gamma} R_\pi + \mathrm{e}^{\gamma t} \int_0^t \gamma^2 \sqrt{\epsilon} B_s \, \mathrm{e}^{-\gamma s} \, ds + \gamma\sqrt{\epsilon} B_t. \tag{3.6.21}$$

Proof. Set $a_1 = \gamma$ and $a_2 = R_\pi + \gamma^2\sqrt{\epsilon} B_t$. Then

$$\begin{aligned} Y_t &= \mathrm{e}^{\gamma t}\left[Y_0 + \int_0^t \left(R_\pi + \gamma^2\sqrt{\epsilon} B_s\right) \mathrm{e}^{-\gamma s} \, ds\right] \\ &= \mathrm{e}^{\gamma t} Y_0 + \frac{\mathrm{e}^{\gamma t} - 1}{\gamma} R_\pi + \mathrm{e}^{\gamma t} \int_0^t \gamma^2 \sqrt{\epsilon} B_s \, \mathrm{e}^{-\gamma s} \, ds. \end{aligned}$$

□

Theorem 3.6.6. *The solution of the scalar RDE with multiplicative noise and $P_\pi = (1)$ is given as*

$$Y_t = \mathrm{e}^{\gamma t} Y_0 + R_\pi \, \mathrm{e}^{\gamma t} \int_0^t \mathrm{e}^{\left(\frac{1}{2}\gamma^2\epsilon - \gamma\right)s - \gamma\sqrt{\epsilon}B_s} \, ds. \tag{3.6.22}$$

The solution of the stochastic differential Bellman equation with multiplicative noise is given as

$$V_t = F_t^{-1} Y_{t-1} = \mathrm{e}^{\gamma\sqrt{\epsilon}B_t - \frac{1}{2}\gamma^2\epsilon t}\left[\mathrm{e}^{\gamma t}\, Y_0 + R_\pi\, \mathrm{e}^{\gamma t}\int_0^t \mathrm{e}^{\left(\frac{1}{2}\gamma^2\epsilon-\gamma\right)s-\gamma\sqrt{\epsilon}B_s}\, ds\right]. \tag{3.6.23}$$

Proof. In this case, $a_1 = \gamma$ and $a_2 = R_\pi\, \mathrm{e}^{\frac{1}{2}\gamma^2\epsilon t - \gamma\sqrt{\epsilon}B_t}$. Thus,

$$\begin{aligned} Y_t &= \mathrm{e}^{\gamma t}\left[Y_0 + R_\pi \int_0^t \mathrm{e}^{\frac{1}{2}\gamma^2\epsilon s - \gamma\sqrt{\epsilon}B_s}\, \mathrm{e}^{-\gamma s}\, ds\right] \\ &= \mathrm{e}^{\gamma t}\, Y_0 + R_\pi\, \mathrm{e}^{\gamma t}\int_0^t \mathrm{e}^{\left(\frac{1}{2}\gamma^2\epsilon-\gamma\right)s-\gamma\sqrt{\epsilon}B_s}\, ds. \end{aligned}$$

□

This approach provides an alternative method to solve the stochastic differential Bellman equation by approximating it with a system of ordinary differential equations. Note that this approximation is applicable only when a policy is given, and it cannot be used to find an optimal policy of a Markov decision process with noise. In the following, the solutions to these random differential equations will be obtained using the Euler method in order to determine an optimal policy. Thereby the stochastic algorithm will be used with the same scheme as in Sect. 3.2.3.

Theorem 3.6.7. *Consider the partition of the interval $[0, T]$ into n equal subintervals of length $\Delta t > 0$. The approximation of the random differential equation driven by additive noise is*

$$Y_{i+1} = \max_{a\in A}\left\{\left[R_a + \gamma P_a\left(Y_i + \gamma\sigma_a B_i\right)\right]\Delta t\right\}, \quad 0 \le i \le n-1$$

and by multiplicative noise is

$$Y_{i+1} = \max_{a\in A}\left\{\left(\mathrm{e}^{\frac{1}{2}\gamma^2\sigma_a^2 i - \gamma\sigma_a B_i}\, R_a + \gamma P_a Y_i\right)\Delta t\right\}, \quad 0 \le i \le n-1,$$

where $B_i = B_{t_i}$ and $Y_i = Y_{t_i}$.

For the sake of comparison, some examples that have been previously discussed in this work are briefly solved using the approach outlined above.

Example 3.6.1 (RDE for solving forest management problem)**.** Reconsider the forest management problem with $\sigma_1 = 0.5$ and $\sigma_2 = 0.1$. The stochastic algorithms are executed for the stepsize $\Delta t = 0.1$.

```
# Generate numerical solution for RDE with additive noise
> RDEalgo(P,R,0.9,c(0,0,0),0.1,0,sigma,dis,0.1,1)
$‘Iterations‘
[1] 25
$Y
[1] 0.03497527 0.10881869 0.20881869
```

```
$‘Optimal Policy‘
[1] 1 2 2
# Generate numerical solution for RDE with mult. noise
> RDEalgo(P,R,0.9,c(0,0,0),0.1,0,sigma,dis,0.1,2)
$‘Iterations‘
[1] 2
$Y
[1] 0.0054 0.1000 0.2000
$‘Optimal Policy‘
[1] 1 2 2
```

Also in this case the algorithm with multiplicative noise terminates faster compared to the one with additive noise. However, both algorithms yield the same optimal policy. The stepsize $\Delta t = 0.01$ leads both algorithms to terminate within two steps. An advantage of these RDE algorithms over those for SBE is that they also terminate for the stepsize 1. The algorithm with additive noise then needs 51 steps and the one with multiplicative noise only 47 steps. This shows that this approach can achieve an optimal policy even with large parameters. Moreover, when the Euclidean norm is used, the value iteration takes 47 steps, so the RDE algorithm with multiplicative noise performs as efficiently as the value iteration.

◇

Example 3.6.2 (RDE for solving replacement problem)**.** The replacement problem with $\sigma_1 = 0.5$ and $\sigma_2 = 0.1$ is solved for the stepsize $\Delta t = 0.1$.

```
# Generate numerical solution for RDE with additive noise
> RDEalgo(P,R,0.9,c(0,0,0),0.1,0,sigma,dis,0.1,1)
$‘Iterations‘
[1] 71
$Y
[1] -23.65199 -42.13575 -42.13575
$‘Optimal Policy‘
[1] 1 2 2
# Generate numerical solution for RDE with mult. noise
> RDEalgo(P,R,0.9,c(0,0,0),0.1,0,sigma,dis,0.1,2)
$‘Iterations‘
[1] 5
$Y
[1] -23.20666 -41.85646 -41.85646
$‘Optimal Policy‘
[1] 1 2 1
```

The modified Euler-Maruyama simulation was able to calculate the optimal policy in 94 steps. With the RDE simulation, the algorithm with additive noise needs 71 steps and with multiplicative noise only 5 steps. The same optimal policy is achieved, but much *faster*.

◇

Example 3.6.3 (RDE for solving inventory problem)**.** Reconsider the inventory problem. Let $\sigma_1 = 0.5$ and $\sigma_2 = 0.1$. The inventory problem is solved for the stepsize $\Delta t = 1$.

```
# Generate numerical solution for RDE with additive noise
> RDEalgo(Pinv,Rinv,0.9,c(0,0,0,0),0.1,0,sinv,dis,1,1)
$`Iterations`
[1] 57
$Y
[1] 17.55224 21.74168 25.46458 27.55224
$`Optimal Policy`
[1] 4 1 1 1
# Generate numerical solution for RDE with mult. noise
> RDEalgo(Pinv,Rinv,0.9,c(0,0,0,0),0.1,0,sinv,dis,1,2)
$`Iterations`
[1] 64
$Y
[1] 17.50873 21.69818 25.42108 27.50873
$`Optimal Policy`
[1] 4 1 1 1
```

In this case the efficiency is not shown for the additive noise model but for the multiplicative noise model. The modified Euler-Maruyama simulation with additive noise takes 43 steps to terminate but with multiplicative noise it does not terminate within 1000 steps. In contrast, the RDE simulation with multiplicative noise terminates in 64 steps with the optimal policy $\pi^* = (3, 0, 0, 0)$, so the full inventory capacity is used. $\diamond$

Based on these examples, the RDE algorithms offer a balance between computational efficiency and solution accuracy. They have computational advantages over traditional methods like value iteration and Euler-Maruyama simulations. Specifically, RDE algorithms require fewer iterations to converge to an optimal policy.

Chapter 4

Conclusion and Future Work

In conclusion, the development of the stochastic differential Bellman equation represents a significant advancement in modeling complex systems and plays a crucial role in addressing uncertainty in real-world scenarios. By incorporating random fluctuations, such as additive and multiplicative noise, into the Bellman equation, real-world decision problems can be effectively tackled in uncertain environments. The applications of the stochastic differential Bellman equation hold promise across various fields, ranging from finance to engineering, where precise modeling of stochastic processes is essential for decision-making and predictions.

Moreover, by formulating the Bellman equation as a stochastic differential equation, it enables the application of mathematical methods like the Itô formula and interpretation as a Fokker-Planck equation. This facilitates the analysis of system dynamics over time, such as examining the probability density function and understanding how different noise models influence distribution dynamics. The additive noise model tends to have a stronger impact on the distribution when the discount factor is higher, resulting in wider probability density functions.

However, the stochastic differential Bellman equation is a nonlinear equation for which there is no general solution method. Therefore, numerical methods are required to obtain solutions. This work focus on developing efficient algorithms for solving decision problems modeled by the stochastic differential Bellman equation and exploring its applicability to various real-world problems. The Euler-Maruyama method is employed, and the stochastic value iteration algorithm is derived to determine optimal policies in stochastic environments. The direct application of the Euler-Maruyama method often results in numerical instabilities and inaccurate results. To counter this challenge, a modified simulation offering a more effective solution is developed. The stochastic value iteration algorithm has demonstrated efficacy in both theoretical analyses and practical applications. It enables solving decision problems under uncertainty and noise, which is a significant improvement over the known deterministic value iteration algorithm.

Applying iterated local search for parameter tuning in the stochastic value iteration algorithm across various scenarios subject to both additive and multiplicative noise underscores the robustness of the algorithm. Even under higher noise intensities, the stochastic value iteration algorithm can still converge to the same optimal policy as in the deterministic case, facilitated by the identified parameters. Statistical tests, including the Wilcoxon-Mann-Whitney test, revealed statistically significant differences between the value functions generated by additive and multiplicative noise models. The choice of the stepsize influences the accuracy and stability of the model. Smaller stepsizes yield more precise results and reduce fluctuations, particularly in the presence of multiplicative noise. If the value functions are significantly different, it may indicate that one noise model variant more accurately describes the system than the other. For larger stepsizes, the choice of noise model may be less important, as no significant differences were found. Moreover, the value function in the case of multiplicative noise consistently resulted in smaller absolute errors compared to the additive noise model, indicating a closer approximation to the deterministic case. Analyzing the upper bounds of the mean energy changes and variances provides insight into the impact of noise on the value function. Higher discount factors increase the difference in variances between the two noise models. This implies that the discount factor has a significant influence on the long-term stability of the system and can magnify its response to noise.

Another contribution of this work is the adaptation of Q-learning for real-world dynamics, termed as the stochastic Q-learning. By incorporating random fluctuations into the Q-function and adapting the update rule to account for noise, it becomes possible to solve decision problems in unknown environments characterized by uncertainty and noise. While both noise models yield similar optimal policies, their impact on runtime, average rewards per episode, and exploration dynamics differ significantly. The incorporation of the ϵ-Greedy strategy enhances exploration but leads to increased runtimes, especially with larger discount factors and learning rates. The maximum of the rewards per episode is higher in the presence of multiplicative noise compared to the case of additive noise, indicating its potential to discover more rewarding actions. The algorithm with multiplicative noise focuses more on exploration.

The algorithms were applied to two real-life decision problems. By exploring various parameters such as discount factors, stepsizes, and noise intensities, several key findings emerged. In one case, a replacement problem was considered and solved using the stochastic value iteration algorithm. This example was chosen to illustrate scenarios where the value functions are large, resulting in greater fluctuations caused by multiplicative noise. In this case, the multiplicative noise model demonstrates higher sensitivity to noise and generates higher value functions compared to the additive noise model. By accounting for unpredictable disturbances and uncertainty, the stochastic value iteration algorithm facilitates the development of optimal policies that balance cost-effectiveness with equipment reliability. In the context of the inventory problem, where decision-making is influenced by uncertain demand and noise, stochastic value iteration and stochastic Q-learning demonstrate adaptability. Stochastic value iteration proves to be more efficient in utilizing inventory capacity compared to the

stochastic Q-learning. However, stochastic Q-learning showcases exploration capabilities leading to higher rewards per episode. Overall, introducing noise enhances the performance of stochastic algorithms by promoting exploration and robustness to uncertainties. Despite variations in noise intensities, the optimal policies derived from stochastic algorithms remain consistent across different runs, indicating their reliability and effectiveness in finding optimal solutions in noisy environments. In conclusion, both stochastic value iteration and stochastic Q-learning offer effective approaches for addressing the real-life decision problems in noisy environments, providing valuable insights.

Finally, the application of random differential equations as a novel approach to solving the stochastic differential Bellman equation is explored. By introducing stochastic integrating factors, random differential equations are derived that approximate the solutions of stochastic differential Bellman equation with different noise models. The computational efficiency of the RDE algorithms compared to the stochastic methods such as stochastic value iteration is demonstrated. RDE algorithms consistently terminated faster, requiring fewer iterations to converge to the optimal policy and can also handle large parameters. All in all, the use of random differential equations represents a valuable method for solving Markov decision processes with uncertainties and related decision-making problems.

4.1 Future Work

The contribution of this work consists of being able to apply methods of machine learning even under noise and to achieve fast and reliable results in stochastic environments. Future work could involve validating the presented stochastic algorithms in real-world environments and exploring their applicability in areas such as robotics and healthcare to further demonstrate their performance and practicality. Through continuous refinement and adaptation, the aim is to address the challenges of decision-making in uncertain environments and develop innovative solutions for real-world problems. While the current research focuses on solving the linear stochastic differential Bellman equation with a given policy, future work could concentrate on finding methods to solve the nonlinear equation exactly.

For stochastic Q-learning, further analysis could explore alternative strategies like the softmax-method instead of the ϵ-Greedy strategy. Additionally, compared to the stochastic value iteration algorithm, the stochastic Q-learning may face limitations if the sets of states and actions are too large due to the storage and updating requirements of the Q-table, which also increases the runtime. Future research could analyze the behavior of noise in deep Q-learning (DQN), where noise occurs with weights and biases.

Regarding algorithms using random differential equations, future analyses could focus on investigating error susceptibility in simulations and exploring how adjusting different parameters may improve computational efficiency and solution accuracy.

Appendices

R-codes
Value Iteration Algorithm

```
ValueIteration <- function(N,P,R,discount,Vprev,eps,iter)
{
 iter <- iter+1 # Increment the iteration counter
 # Execute the Bellman operator
 result <- mdp_bellman_operator(P, R, discount, Vprev)
 Vnew <- result$V # Extract the value for the new iteration
 p <- result$policy # Extract the new optimal policy
 V <- Vnew - Vprev  # Calculate the difference of successive values
 V <- matrix(V, ncol =1) # Ensure V remains a column vector
 Norm <- norm(V,"M") # Compute the norm of the difference
 theta <- eps*(1-discount)/(2*discount) # Compute threshold for convergence
 if(Norm<theta){ # Check if convergence is achieved
 # If yes, return the results
  return(list("Iteration" = iter, "V" = Vnew, "Optimal Policy" = p))
 }
 # If not, proceed to the next iteration
 return(ValueIteration(N,P,R,discount,Vnew,eps,iter))
}
```

Algorithms for the simulation of the SBE

Stochastic differential Bellman equation algorithm: Inputs: probability matrix, reward matrix, discount factor, initial value function, termination coefficient, iteration, noise intensity matrix, Brownian motion, stepsize, noise model (1 for additive noise, 2 for multiplicative noise).

```
SBEalgo <- function(P,R,disc,Vprev,eps,count,sigma,dis,Dt,noise){
 count <- count+1 # Increment the iteration counter
 # Calculate increment of Brownian motion
 brownian <- dis[count+1] - dis[count]
 if(noise=1){
  result <- sbeadd(P,R,disc,Vprev,brownian,sigma,Dt)
  Vnew <- result$V
  p <- result$policy
 }else{
  result <- sbemult(P,R,disc,Vprev,brownian,sigma,Dt)
  Vnew <- result$V
  p <- result$policy
 }
 V <- Vnew - Vprev
 V <- matrix(V, ncol =1)
 Norm <- norm(V,"M")
 theta <- eps*(1-disc)/(2*disc)
 # Check if convergence or the maximum number of iterations is reached
 if(Norm<theta || count == 1000){
```

```
  return(list("Iterations" = count, "V" = Vnew, "Optimal Policy" = p))
 }
 return(SBEalgo(P,R,disc,Vnew,eps,count,sigma,dis,Dt,noise))
}
```

Sub-algorithm for the case of additive noise:

```
sbeadd <- function(P,R,disc,Vprev,Brownian,sigma,Dt){
 V <- matrix(0, dim(R)[1], dim(R)[2]) # Initialize the value function
 if(is.list(P)){ # Check if transition matrix is a list
  A <- length(P) # Number of actions
  for (a in 1:A){
   noise <- disc * sigma[,a] * Brownian # Calculate diffusion term
   # Update value function for each action
   V[,a] <- as.matrix(((R[,a] + disc * P[a][1] %*% Vprev) * Dt) + noise)
  }
 }else{
  A <- dim(P)[3]
  for(a in 1:A){
   noise <- disc * sigma[,a] * Brownian # Calculate diffusion term
   # Update value function for each action
   V[,a] <- ((R[,a] + (disc * P[,,a] %*% Vprev)) * Dt) + noise
  }
 }
 # Return the updated value function and corresponding policy
 return(list("V" = apply(V, 1, max), "policy" = apply(V, 1, which.max)))
}
```

Sub-algorithm for the case of multiplicative noise:

```
sbemult <- function(P,R,disc,Vprev,Brownian,sigma,Dt){
 V <- matrix(0, dim(R)[1], dim(R)[2]) # Initialize the value function
 if(is.list(P)){
  A <- length(P)
  for (a in 1:A){
   noise <- disc * sigma[,a] * Vprev * Brownian # Calculate diffusion term
   # Update value function for each action
   V[,a] <- as.matrix(((R[,a] + disc * P[a][1] %*% Vprev) * Dt) + noise)
  }
 }else{
  A <- dim(P)[3]
  for(a in 1:A){
   noise <- disc * sigma[,a] * Vprev * Brownian # Calculate diffusion term
   # Update value function for each action
   V[,a] <- ((R[,a] + (disc * P[,,a] %*% Vprev)) * Dt) + noise
  }
 }
```

```
 # Return the updated value function and corresponding policy
 return(list("V" = apply(V, 1, max), "policy" = apply(V, 1, which.max)))
}
```

The function `EulersimSBE` proceeds similar to the above algorithms, only the previous value function is added up in the calculation.

RDE Algorithms for the SBE

Sub-algorithm for the case of additive noise:

```
rdeadd <- function(P,R,disc,Yprev,Brownian,sigma,Dt){
 Y <- matrix(0, dim(R)[1], dim(R)[2])
 if(is.list(P)){
  A <- length(P)
  for (a in 1:A){
   noise <- disc * sigma[,a] * Brownian # Calculate diffusion term
   # Update Y for each action
   Y[,a] <- as.matrix((R[,a] + (disc * P[a][1] %*% (Yprev+noise))) * Dt)
  }
 }else{
  A <- dim(P)[3]
  for(a in 1:A){
   noise <- disc * sigma[,a] * Brownian # Calculate diffusion term
   # Update Y for each action
   Y[,a] <- ((R[,a] + (disc * P[,,a] %*% (Yprev+noise))) * Dt)
  }
 }
 # Return the updated Y and corresponding policy
 return(list("Y" = apply(Y, 1, max), "policy" = apply(Y, 1, which.max)))
}
```

Sub-algorithm with multiplicative noise:

```
rdemult <- function(P,R,disc,Yprev,Brownian,sigma,Dt,count){
 Y <- matrix(0, dim(R)[1], dim(R)[2])
 if(is.list(P)){
  A <- length(P)
  for (a in 1:A){
  # Calculate diffusion term for each action
   noise <- 0.5*disc^2 * sigma[,a]^2*count - disc * sigma[,a] * Brownian
   # Update Y for each action
   Y[,a] <- as.matrix((exp(noise)*R[,a] + disc * P[a][1] %*% Yprev) * Dt)
  }
 }else{
  A <- dim(P)[3]
  for(a in 1:A){
```

```
  # Calculate diffusion term for each action
   noise <- 0.5*disc^2 * sigma[,a]^2*count - disc * sigma[,a] * Brownian
   # Update Y for each action
   Y[,a] <- ((R[,a] + (disc * P[,,a] %*% Yprev)) * Dt)
  }
 }
 # Return the updated Y and corresponding policy
 return(list("Y" = apply(Y, 1, max), "policy" = apply(Y, 1, which.max)))
}
```

Bibliography

[1] Abualhaija, S. *D-Bees: a novel global algorithm for solving word sense disambiguation.* Technical University of Hamburg (Doctoral dissertation), 2016.

[2] Baiazid, O. *Methods of Machine Learning and their application.* TUHH, Bachelor-thesis, 2021.

[3] Bellman, R. E. *Dynamic Programming.* Princeton University Press, 1957.

[4] Brereton, T. *Reading Course.* Lecture Notes, Ulm University, Institute of Stochastics, 2015.

[5] Brown, R. A brief account of microscopical observations made in the months of june, july and august, 1827, on the particles contained in the pollen of plants; and on the general existence of active molecules in organic and inorganic bodies. *The Philosophical Magazine, 161–173* (1905).

[6] Carlstein, A. Stochastic calculus: Understanding brownian motion and quadratic variation. *The University of Chicago Mathematics, REU paper* (2019).

[7] Choi, Yeung, and Zhang. An environment model for nonstationary reinforcement learning. *In Advances in Neural Information Processing Systems 12, The MIT Press, 987–993* (2000).

[8] Conway, J. B. *A course in functional analysis.* Springer, 2007.

[9] Dehling, and Haupt. *Einführung in die Wahrscheinlichkeitstheorie und Statistik.* Springer, 2004.

[10] Denardo, E. V. Contraction mappings in the theory underlying dynamic programming. *SIAM Review, 169-177* (1967).

[11] Dieckow, N. *Methods of Machine Learning and their Application to the Repair Limit Replacement Problem.* TUHH, Bachelor-thesis, 2021.

[12] Duan, J. *An Introduction to Stochastic Dynamics.* Cambridge University Press, 2015.

[13] Efron, B. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics, 1-26* (1979).

[14] Fokker, A. D. Die mittlere energie rotierender elektrischer dipole im strahlungsfeld. *Annals Phys., 810-820* (1914).

[15] Frank, T. D. *Nonlinear Fokker-Planck Equations Fundamentals and Applications*. Springer Series in Synergetics (SSSYN), 2005.

[16] Glen, S. *StatisticsHowTo.com*, Accessed December, 2020.

[17] Grimmett, and Welsh. *Probability - An Introduction*. Oxford Univ. Press, 2014.

[18] Haardt, M. *Fokker-Planck Gleichung*, Accessed March, 2021.

[19] Han, and Kloeden. *Random Ordinary Differential Equations and Their Numerical Solution*. Springer, 2010.

[20] Hastings, N. The repair limit replacement method. *Journal of Opertional Research Society, 337-349* (1969).

[21] Hemminger, J. *Directing Matter and Energy: Five Challenges for Science and the Imagination*. A Report from the Basic Energy Sciences Advisory Committee, U.S. Department of Energy, 2007.

[22] Hesse, C. *Wahrscheinlichkeitstheorie*. Vieweg+Teubner, 2009.

[23] Higham, D. An algorithmic introduction to numerical simulation of stochastic differential equations. *SIAM Review, 525-546* (2001).

[24] Hüber, P. M. *Iterative methods of machine learning and their application*. TUHH, Bachelor-thesis, 2021.

[25] Jaakkola, Jordan, and Singh. Convergence of stochastic iterative dynamic programming algorithms.

[26] Kaelbling, Littman, and Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence 101, 99–134* (1998).

[27] Kahneman, Sibony, and Sunstein. *Noise: A Flaw in Human Judgment*. New York Times, 2021.

[28] Keller-Ressel, M. *Stochastische Analysis*. Lecture Notes, TU Dresden, 2015.

[29] Kiyoshi, I. Stochastic integrals. *Proceedings of the Imperial Academy, 519-524* (1944).

[30] Kiyoshi, I. Stochastic differential equations in a differentiable manifold. *Nagoya Mathematical Journal, 35-47* (1950).

[31] Langevin, P. Sur la théorie du mouvement brownien [on the theory of brownian motion]. *Comptes Rendus de l'Academie des Sciences, 530-533* (1908).

[32] Lefebvre, M. *Applied Probability and Statistics*. Springer, 2010.

[33] Marr, B. *A Short History of Machine Learning – Every Manager Should Read*, Accessed February, 2021.

[34] Maruyama, G. Continuous markov processes and stochastic equations. *Rend. Circ. Mat. Palermo, 48-90* (1955).

[35] Oberoi, A. *15 Machine Learning Examples from Day-to-Day Life*, Accessed August, 2021.

[36] Pavliotis, G. A. *Stochastic Processes and Applications: Diffusion Processes, the Fokker-Planck and Langevin Equations*. Springer, 2014.

[37] Platen, and Kloeden. *Numerical Solution of Stochastic Differential Equations*. Springer, 1999.

[38] Platen, E., and Bruti-Liberati, N. *Numerical Solution of Stochastic Differential Equations with Jumps in Finance*. Springer, 2010.

[39] Puterman, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 1994.

[40] Ravichandiran, S. *Hands-On Reinforcement Learning with Python*. Packt Publishing, 2018.

[41] Robbins, and Monro. A stochastic approximation method. *The Annals of Mathematical Statistics, 400–407* (1951).

[42] Rowland, Todd, and Weisstein. *Bounded Variation*, Accessed February, 2022.

[43] Russell, and Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2010.

[44] Samuel, A. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development, 210-229* (1959).

[45] Sasane, A. *A Friendly Approach to Functional Analysis*. World Scientific, 2017.

[46] Sauer, T. Computational solution of stochastic differential equations. *WIREs Computational Statistics, 362-371* (2013).

[47] Stewart, J. *Calculus*. Brooks Cole, 2002.

[48] Stirzaker, D. *Stochastic Processes and Models*. Oxford University Press, 2005.

[49] Sutton, and Barto. *Reinforcement Learning: An introduction*. The MIT Press, 2018.

[50] Tanaka, H. Summary of the fokker-planck equation approach to asset price fluctuations. *Research Institute for Mathematical Sciences, Kyoto University, 176-190* (2003).

[51] Wantou, B. D. *Verfahren des Maschinellen Lernens zur Entscheidungsfindung.* TUHH, Bachelorarbeit, 2021.

[52] Watkins, and Dayan. Q-learning. *Machine Learning 8, 279–292* (1989).

[53] Wiener, N. Differential space. *Journal of Mathematics and Physics, 131-174* (1923).

[54] Zimmermann, K.-H. *Soft Computing.* Lecture Notes, Hamburg University of Technology, 2017.

[55] Øksendal, B. *Stochastic Differential Equations: An Introduction with Applications.* Springer, 2003.

Index