Option pricing using Generative Adversarial Networks

MSc Financial Engineering & Management

Siem Peters

Supervisors University of Twente
L. Spierdijk
J. Osterrieder

Supervisor Simula Research Laboratory
X. Cai

University of Twente
Industrial Engineering & Management
Faculty of Behavioural, Management and Social Sciences

The Netherlands
26 January 2024

Abstract

This master thesis introduces a novel approach to price financial options, namely a GAN-QMC. It combines a Generative Adversarial Network (GAN) with a Quasi-Monte Carlo (QMC) simulation. The main objective is to improve the accuracy and efficiency of option pricing, specifically focusing on overcoming limitations associated with traditional QMC for option pricing. In this research, GANs are used to model the process of the underlying asset, aiming for a realistic representation of financial data as input to the QMC.

This thesis begins by building a theoretical framework, using relevant literature to place the topic within the broader landscape of option pricing, Monte Carlo (MC) simulation, and GANs. The methodology involves the development of the GAN-QMC framework including option pricing using MC/QMC, neural networks, GANs, and risk-neutral price path construction. Here we discovered that the GAN-QMC can only effectively be implemented as a semi-parametric method, relaxing certain distributional assumptions associated with traditional MC/QMC option pricing using the geometric Brownian motion.

The concepts of the GAN-QMC methodology were implemented. The GAN hyper-parameters were determined by using an automated tuning strategy based on perturbation theory. Three separate GANs were tuned using the historical stock prices of BHP Group Limited, Commonwealth Bank of Australia (CBA), and CSL Limited. The GAN's ability to generate financial stock returns was determined by comparing it to real-world data from exchange-listed companies. The Wasserstein GAN with Gradient Penalty was best in capturing the statistical properties exhibited by the real-world data but was still not perfect.

Afterward, we compared the performance of MC, QMC, GAN-MC, and GAN-QMC by pricing a dataset of options. In total, this dataset consisted of almost 500 European and American options on the stocks of BHP, CBA, and CSL, both in-the-money and out-of-the-money. Here we discovered that the GAN-QMC demonstrates comparable accuracy to QMC with an error of roughly 3.5%. In many conventional pricing methods including QMC, the returns of the underlying asset of an option are modeled using a standard normal distribution. Therefore, these results are interesting because they show that a GAN, which is non-parametric, can replace the theoretical way the underlying asset of an option is often modeled in a geometric Brownian motion. Moreover, GAN-QMC seems to show potential advantages in computational efficiency, being approximately 20% faster.

In summary, this thesis contributes to the field of option pricing by introducing GAN-QMC as a semi-parametric framework with potential advantages in accuracy and efficiency. In addition, a new strategy to tune GANs was successfully applied. While there are challenges and limitations, particularly in training a financial GAN, this study does suggest possibilities for future research to improve GANs and the GAN-QMC method further.

Preface

Dear Reader,

Hereby I present to you my Master's thesis on option pricing using Generative Adversarial Networks and Quasi-Monte Carlo simulation. The goal was to improve the accuracy and efficiency of pricing options through Quasi-Monte Carlo simulation by combining it with Artificial Intelligence.

My thesis was written at Simula Research Laboratory within the Department of High-Performance Computing where I very much enjoyed working. I had an educational time and got great support with many helpful improvements and suggestions from Professor Xing Cai. The informal working culture encouraged an environment where I could seek as much guidance and help as I desired. I am very grateful that Xing Cai gave me the opportunity to write my thesis at Simula Research Laboratory and I am very happy how my final period as a student turned out.

In addition, I would like to thank Laura Spierdijk and Joerg Osterrieder the supervisors from the University of Twente. I am particularly thankful to Laura Spierdijk for her valuable feedback and guidance, which steered this research in a positive direction. Lastly, I would like to thank my family, girlfriend, and friends for all their support during this period. I very much look forward to all the interesting challenges that lie ahead.

Enjoy reading this thesis!

Siem Peters

Oslo, Norway, January 2024

Contents

List of Figures

Chapter 1

Introduction

To kick off this thesis, we begin by introducing the topic and providing some background context for this research. From there, we identify the specific research problem and establish the relevant research questions. The subject of this thesis is the pricing of financial options. Options, a type of financial instrument, are contracts that give the holder the right to buy or sell an underlying asset at a specified price within a predetermined period. By developing and analysing a new pricing method using Artificial Intelligence this thesis aims to improve on existing option pricing frameworks. How the new pricing method works and which improvements it tries to make will all be explained in this chapter. More technical details regarding option pricing and the developed pricing methods are all explained in chapter 2 and 3.

1.1 The use of financial instruments

Many companies, organizations, and governments invest in a wide variety of assets. These investments come with risks, because unexpected events can happen, leading to massive losses. To protect the investor against such risks, financial instruments can be bought (Hull, 2000). These are special contracts allowing for potential gains of the investment while some of the associated risks can be covered through these products. Figuring out how much these financial instruments are worth is essential for effective risk management (McNeil et al., 2005).

To give a well-known comparison, this is similar to buying insurance for your car. You seek to protect yourself from unforeseen accidents, theft, or damage while being able to enjoy all the benefits of a car. You want to know how much it should cost, which risks it covers, and how much risk is left over for you. Similarly in the realm of finance, investors use financial instruments to protect themselves from negative events while at the same time being able to profit from positive market movements or business results. For example, a company doing business all over the world might want to protect itself from changes in currency exchange rates. Preferably, a change in currency rate should not completely offset the profit made in a foreign country. This can be achieved by certain types of financial instruments. Therefore, knowing how much you should pay for this risk reduction is crucial.

Risk management plays an important role in the financial sector. Financial instruments are not only used for potential gains but also to mitigate risks associated with investments and doing business (Bartram, 2019). Here you could for example think about a manufacturing company hedging against the risks associated with fluctuations in steel prices using future contracts to ensure that its operations are also profitable when the steel price is extremely high. Options can be used by for example an asset manager investing in some stocks to protect against a large sudden price drop. Determining the true value of these financial instruments is a complex challenge and is important because it should accommodate both protective and profit-seeking motives. Thus reliable valuation techniques are necessary for effective risk management.

In addition, the financial landscape is constantly evolving, influenced by developments in technology, regulatory changes, and shifting market dynamics. These changes, driven by technological innovation and regulatory responses (Williams, 2013), reshape the financial sector. Regulatory changes are installed to protect companies and citizens from risky investments that can go wrong at their expense. Here a reliable valuation, while also modeling the impact of these changes, of financial instruments is crucial to assess and be protected from the true risks of investments and other related events. Another quickly developing event that governments and citizens might want to protect themselves from is climate change resulting in larger and more frequently occurring natural disasters (Venturini, 2022). This can also be achieved using financial instruments, however, modeling the rare events of natural disasters is hard.

Regardless of the purpose of using a financial instrument, they all share one important aspect namely the need to have a reliable evaluation method to determine its true value. This ensures effective handling of risks and profit-seeking motives.

1.2 Derivatives and options

One type of financial instrument is a derivative. This is a contract that, as the name suggests, derives its value from some underlying asset, index, or reference rate (Quail et al., 2003). Derivatives can be used to create exposure to certain assets or to protect against risks. They are either a lock or an option. The most common lock products are futures, forwards, and swaps. These are contracts where both parties agree on some terms that they are obliged to fulfill on some specified date in the future. For example, a future is a contract where both parties agree to trade an asset for some pre-specified price in the future. This thesis is not focused on lock products but on the pricing of options.

The first option contract dates back to the period of ancient Greece. After predictions that the olive harvest in the next season would be exceptionally high, a mathematician bought the right to use some olive presses. When the olive harvest indeed turned out to be high, he rented the olive presses out for a value significantly above what he paid for it through his contract. The book Confusion de Confusiones was the first book talking about 'opsies', a tradeable instrument limiting risks on the Amsterdam stock exchange (Vega, 1688). Options have been traded officially since 1973. Nowadays the most traded financial instruments are options. In 2022, a total of 54 billion option contracts have been sold (Statista, 2023).

There are many different option types, all with their own specific functions and payoffs. In general, an option gives the buyer the right, but not the obligation, to buy or sell the underlying asset for a certain price at a specific date in time (Hull, 2000). The seller always has the obligation to fulfill the contract if the holder of the option wants to buy or sell the asset. The most common type of option is a European option. This option can only be exercised at the final date written on the option's contract. In contrast, the American option allows the buyer to exercise the option at any point during the lifetime of the option. This will be explained in more detail in chapter 3. Every option is either a put or a call option. A put option gives the buyer the right to sell the underlying asset and a call option the right to buy the underlying asset (Bolia et al., 2005).

There are also many types of exotic options all with their own rules and payoffs like a barrier and lookback option. These exotic options have special rights which result in complex payoff structures and are often tailor-made to meet specific market conditions or investor needs. For example, a lookback option allows the buyer, when the option contract ends, to exercise it during any part of its lifetime. Therefore, the buyer knows the history and can thus exercise it for the optimal payoff which improves the timing of market entry.

1.3 Pricing an option

Understanding the option pricing models is essential for pricing options and making well-informed investment or risk management decisions. There are various models, such as the Black-Scholes model (Hull, 2000), that help to determine the theoretical value of options based on many factors such as the underlying asset's price, the option's strike price, time to maturity, volatility, and risk-free rate. These models are important tools for traders and investors, helping to assess potential profits, risks, and strategies related to options trading. It is very hard to model all factors influencing an option and it therefore tends to be difficult to price an option. In the upcoming chapters of this thesis, we will go deeper into the current state of option pricing, its most well-known methods, and its challenges.

The method of pricing financial instruments where assumptions about the underlying asset or market conditions are made is called a parametric pricing method. If no assumptions need to be made, it is called a non-parametric pricing method. Relaxing some assumptions of an established parametric pricing method can be called a semi-parametric pricing method. The majority of pricing methods are parametric, however, the rapidly evolving field of Artificial Intelligence (AI) opens up enhanced possibilities for pricing options using non-parametric and semi-parametric approaches.

We will now briefly explain which elements influence the price of an option and which potential assumptions can be made because this impacts the difficulty of determining the true prices of options. Commonly it is assumed that the financial markets work in a very specific way. Many pricing methods lean heavily on the assumptions that assets adhere to specific models and that there are specific market conditions. Based on these assumptions the prices of financial instruments are calculated. However, the real world is often not that simple, and using a one-size-fits-all approach is not fair for every situation. Asset prices can move in all sorts of non-understood ways and sometimes do not follow predetermined rules at all. The behavior of asset prices is almost always unpredictable. These prices can fluctuate in ways that defy conventional models and expectations. This unpredictability makes the pricing of options more complex, as traditional methods, where some are discussed in chapter 2, may not adequately capture or adapt to these market dynamics. It is not the case that every parametric method makes all these assumptions that we now discuss, some can be relaxed, but this is to show which assumptions are often made in option pricing such that it becomes clear that they are not always valid. We will now discuss the main problems related to making different parametric assumptions for option pricing.

1.3.1 Asset dynamics

One primary assumption involves the dynamics or behavior of the underlying asset. Traditional models often assume that the asset follows a specific stochastic process, such as a geometric Brownian motion, which will be discussed in more detail in section 2.1. This assumption implies a continuous and normally distributed price process. However, in reality, asset prices can behave in more complex patterns and may not adhere strictly to these prescribed processes. Factors like market news, events, or sudden shifts in investor sentiment can significantly impact an asset's behavior, which can be hard to model accurately. Additionally, it has been shown that real-world asset returns do not always follow a normal distribution (Theodossiou, 2000), as often assumed.

1.3.2 Volatility

Volatility is a crucial parameter in option pricing models, representing the degree of price fluctuations of the underlying asset. Models typically assume that volatility is constant over the option's life, known as constant volatility or implied volatility. However, in reality, volatility

tends to vary over time due to changing market conditions, news events, or other external factors. Incorrect assumptions about volatility can lead to inaccurate option prices and thus mismanagement of risk.

Options with different strike prices and expiration dates will have different implied volatilities, which is known as the volatility smile (Gatheral, 2006). In contrast to historical volatility, which is based on past price movements, implied volatility represents the market's anticipation of future price volatility as deduced from current option prices. The volatility smile indicates that the implied volatility increases as the option moves further out-of-the-money or in-the-money. This characteristic can be effectively modeled using for example Stochastic Volatility (SV) models, which assume that the volatility is a random variable. These models capture the characteristics of the so-called volatility surface, modeling the relationship between implied volatility, strike prices, and expiration dates.

1.3.3 Risk-free rate

The risk-free rate is another essential parameter in option pricing, representing the return an investor can expect from a risk-free investment. We will discuss the concept of risk-neutrality and why this holds in section 2.1. Pricing models often assume a constant risk-free rate, derived from government bond yields. However, in practice, market interest rates can fluctuate due to economic conditions, central bank policies, and geopolitical events. Failing to accurately model the risk-free rate can distort option prices and risk management strategies.

1.3.4 Market liquidity and transaction costs

Pricing models often assume perfect market liquidity and neglect transaction costs. In reality, market liquidity can vary, impacting the bid-ask spread (Amihud et al., 1986) and ultimately affecting option prices. Transaction costs, including brokerage fees and taxes, are important considerations that can significantly influence the true price of an option, which might be especially relevant for high-frequency trading strategies.

1.3.5 Dividend yield

Options on stocks can often involve assumptions about the stock's dividends, although there have been many methods to relax this. Pricing models may assume constant or deterministic dividend yields, but in practice, dividend policies can change, leading to variations in expected future cash flows. Incorrect assumptions about dividend yields can result in inaccurate option prices.

1.3.6 Market efficiency

Traditional pricing models often assume market efficiency, implying that all relevant information is reflected in asset prices. However, markets can be inefficient, especially during volatile periods or with new, unexpected information. This is a more tricky assumption because it is really hard to quantify if a market is even inefficient in the first place. However, in the case of market inefficiency, it can lead to mispriced options and suboptimal investment decisions.

1.3.7 Core problem

Which main parametric methods exist and which assumptions they use is discussed in section 2.1. The core problem is the inaccurate valuation of financial options by making unrealistic assumptions about the underlying assets and market conditions which can lead to inaccurate pricing resulting in a multitude of problems including sub-optimal risk management and inefficient trading strategies. What we want to tackle in this thesis is to price financial options

without relying much on strict assumptions, possibly relaxing some of the aforementioned assumptions, especially for the asset dynamics. This hopefully improves the pricing accuracy for some types of options by using a newly developed pricing method involving fewer assumptions. We want a method that is flexible and adaptable to the market and can thus handle different situations better to price options more reliably.

1.4 Researching a new pricing method

As mentioned previously, efficient pricing and hedging of financial instruments play a vital role in modern finance, serving as essential tools for risk management and investment strategies. While conventional approaches like the Black-Scholes formula (Black et al., 1973) have shown success in certain scenarios, they heavily rely on strict assumptions especially regarding the underlying asset's price dynamics. As a consequence, these methods may fall short in accurately capturing the complexities of real-world financial markets. In addition, many of these methods are not well suited for options whose value depends on the price path of the underlying asset. Therefore research in more flexible and robust pricing techniques is a necessity to improve the accuracy and computational efficiency of pricing financial instruments. It is deemed important to research pricing techniques that can adapt to a wider range of underlying asset price processes.

We present an innovative research approach that combines two advanced techniques in computational finance: Generative Adversarial Networks (Goodfellow et al., 2014) (GANs) and Quasi-Monte Carlo (QMC) simulation (Sobol, 1990). The GAN will be used to create synthetic financial data. This data can be used to construct potential future price paths of the underlying asset. These generated price paths can be used to price the option using its payoff function in a simulation. A GAN is a machine learning model that consists of two neural networks, a Generator and a Discriminator, trained together in a process to create and evaluate realistic synthetic data. A QMC simulation will be used to sample many random price paths from a GAN from which the option price and relevant statistical measures can be calculated. QMC improves on regular Monte Carlo (MC) by using low-discrepancy sequences to generate more evenly spaced sample points such that its convergence improves. As the name suggests, it is therefore not truly random. Even though this is very technical and still sounds vague, both these concepts will be explained more elaborately in chapter 2 and 3. GAN has shown major success in many papers in the accurate data generation of images (Weng et al., 2022), natural language text (Rosa et al., 2022), and audio (Latifi et al., 2019). In addition, there have been some applications in other fields as well like physics (Kansal et al., 2020) and finance (Takahasi et al., 2019), although the amount of research is a bit scarcer here.

GANs allow us to model complex and high-dimensional distributions, capturing the underlying asset's price dynamics with solid accuracy and flexibility (Takahasi et al., 2019). This could limit the number of parametric assumptions necessary to capture the asset's process. Thus a GAN might help to provide a more realistic representation of the asset's behavior, hopefully improving the accuracy of the option's price by generating more realistic price paths. In addition, our hypothesis is that by using the GAN's synthetic data, generated during the QMC simulation, it might require fewer trials to achieve accurate estimates compared to using the quasi-random price paths of regular QMC, thus also improving the efficiency of pricing options.

Existing non-parametric pricing methods, such as Hutchinson's approach (Hutchinson et al., 1994), struggle with robustness and stability. This is because many of the non-parametric methods rely heavily on the specific characteristics of derivatives' regime data. Regime data refers to the data or observations collected during specific market regimes or market conditions like the very volatile period during the financial crisis in 2008 (Schwert, 2011). In contrast, GANs might be more adaptable and less dependent on any particular regime, making them more robust across different market scenarios. In addition, the combination of GANs and QMC could result in a good synergy, considering that the GAN's ability to mimic market data (Eckerli

et al., 2021) can supplement the robustness of the QMC simulation (Caflisch, 1998), allowing us to hopefully achieve more efficient and precise option valuations. By utilizing the strengths of both techniques, we hope to better approximate the underlying asset price distributions to improve efficiency and reduce pricing errors.

In summary, the primary goal of this research is to improve option pricing by introducing a new pricing framework using GANs and QMC simulation. We try to overcome the limitations of traditional parametric pricing models and improve the accuracy and efficiency of option valuations.

1.5 Research problem

We will now discuss the research questions that must be answered in order to find a suitable approach to pricing options through GAN and QMC.

1.5.1 Research Questions

The primary research question that guides this thesis is as follows:

> *Can a Generative Adversarial Network and Quasi-Monte Carlo simulation be combined to create a non-parametric pricing framework for pricing different types of options, to achieve more accurate and efficient valuations?*

To address the main research question comprehensively, the following sub-research questions need to be answered:

1. *How can a GAN be applied to model the underlying asset's price dynamics?*

   To answer the main research question, it is important to understand how a GAN works, how to build it effectively, which applications have already been established within finance, and some extensions that could lead to improvements. This will be achieved by studying the literature and doing an analysis using real-world data to determine how to effectively use the data and tune the GAN to model an asset's dynamics.

2. *How can the quasi-random sequence of QMC be combined with a GAN? And does a quasi-random sequence enhance the pricing accuracy of a GAN-based method?*

   The whole goal of QMC over MC is to use a quasi-random sequence instead of a pseudo-random sequence. One of our hypotheses is that a simulation incorporating a GAN can be made more computationally efficient using a quasi-random sequence. Whether this will actually be the case has to be researched considering that sampling from a GAN is not the same as sampling from a continuous probability distribution as done in regular QMC for option pricing. A quantitative analysis will be done using our developed method for European and American options to determine how to effectively combine a quasi-random sequence into a GAN. In addition, it is interesting to measure whether using a quasi-random sequence in the GAN-QMC leads to more accurate results compared to using a pseudo-random sequence (which would be the case comparing MC and QMC).

3. *How does the proposed GAN-QMC method compare to the standard QMC simulation for option pricing in terms of accuracy and computational efficiency?*

   The newly developed method will be compared to the already well-established MC and QMC simulation on a selection of accuracy and efficiency criteria to determine the effectiveness of this research.

As mentioned before, the reason for choosing this research topic is because traditional parametric pricing methods can lead to inaccurate option valuations and hedging errors in dynamic and complex financial markets, because these methods often lack stability which is a major issue for out-of-sample performance (Gradojevic et al., 2011). The proposed framework allows for a more realistic representation of the underlying asset dynamics, meaning that we will use a broader range of price processes, hopefully limiting the number of parametric assumptions. This might lead to a new and different suitable alternative to price options. Moreover, the integration of QMC simulation aims to reduce variance and improve the efficiency of option valuations (Joy et al., 1996). MC is very robust but also slow (Caflisch, 1998), thus the use of GAN with QMC might significantly increase convergence while keeping its robustness. The combination of GANs and QMC simulation for derivatives pricing and hedging is relatively unexplored. Some analysis has been done on regular MC and GANs for option pricing (W. Wang, 2022) (efficiency not measured). Therefore, our primary interest is whether using QMC in combination with GAN improves the accuracy and computational efficiency of option pricing compared to regular QMC (Snyder, 2000). The combination of specifically QMC simulation and GAN is still unexplored.

This research aims to fill the gaps in the existing literature by proposing a novel approach that integrates GANs and QMC simulation. With this research, we try to improve on or add to the field of study in the following manner:

- **Development of a pricing framework:** The integration of a GAN allows for the flexible modeling of complex price distributions, potentially eliminating the dependence on specific parametric assumptions. Here the choice of neural network could lead to new insights into the GAN's accuracy for generating financial time series data.
- **Improving accuracy, statistical stability, and computational efficiency:** By incorporating a GAN with QMC a new pricing method is developed for this thesis which might improve the stability of option pricing and reduce variance, leading to more efficient and accurate option valuations.
- **Enhancing risk management and portfolio optimization:** The proposed framework could lead to more reliable hedging strategies, improving risk management for financial practitioners.
- **Advancing computational finance:** By combining GANs and QMC, this research contributes to the advancement of computational techniques in finance, by analysing the effectiveness of this method, where the goal is to improve the accuracy and efficiency of option pricing.

1.6 Research structure

To answer sub-research question 1, it is important to know how to effectively use GANs to model the underlying asset's price dynamics. To build a suitable GAN model, we must first conduct a short literature review about how GANs work and their applications, see chapter 2. Once completed, we will decide on the architecture of the GAN model in chapter 3. In chapter 4 we will tune the chosen GANs. The GAN models will be tested on multiple aspects to determine whether they can somewhat capture the asset's dynamics and can thus effectively create asset price paths.

For sub-research question 2, we need to figure out how to combine the generated data of the GAN with QMC. To do this, a QMC base is necessary first. Therefore, a literature review is used to understand methods of pricing options through QMC, see chapter 2. The next step is to develop a simple but effective way to set up the QMC simulation for pricing options.

Beforehand, we must set up the option price formulas based on the payoff structure of each option type, which is done in chapter 3. Once this is completed and the GAN has been tuned, we can start to incorporate the data from the GAN into the QMC. Here, some trial and error will be necessary to determine different ways to incorporate the data into the simulation. In chapter 5 it will be concluded whether using a quasi-random sequence over a pseudo-random sequence improves the efficiency of the GAN-QMC simulation.

Sub-research question 3 is answered using the developed GAN-QMC. Before doing this, we discuss which data is chosen and the exact setup of the analysis, like the selection of performance measures in chapter 4. For European and American options on a selected market, we will calculate prices using the new method in chapter 5. Now we can compare the accuracy and efficiency of the developed method and QMC by itself to the real market value of the chosen option. The computational times of QMC will be compared to the GAN-QMC method because we hypothesize that QMC's efficiency for option pricing can be improved by implementing a GAN.

In addition, a sensitivity analysis is employed in chapter 5. During this, some of the input parameters of the pricing formula are varied one by one within a certain range to observe how the output of the simulation changes. This could for example be the strike price or time to maturity of the option. This allows us to determine under which parameter changes the difference between the pricing methods in terms of accuracy is most notable.

1.6.1 Market

To determine the effectiveness of the approach, we must conduct tests on out-of-sample real-life data. There are many different markets that the model can be trained and tested for, each with their own dynamics and asset characteristics that the GAN must capture. However, due to time constraints, we had to focus on a specific market type. The following markets are the most common option market types (Darskuviene, 2010):

1. **Equity options:** These options derive their value from single stocks. In this case, our research outcome will be determining the effectiveness of valuing and hedging options on publicly traded companies. This could be useful for improved risk management for investors and fund managers.

2. **Index options:** Somewhat similar to equity options, index options are based on market indices, like the S&P500. These are thus valued for a group of stocks, or bonds (such as a hypothetical portfolio). The results of the research could thus be applicable to investors with more diversified portfolios.

3. **Currency options:** These are based on the exchange rates between different currencies. Here, the effectiveness shows the GAN's ability to model the complexity of interrelated market dynamics. This could improve risk management for international businesses and financial institutions trying to hedge their exchange rate risks. However, due to the complexity of learning these market interactions and the existing focus of GAN research on other market types, currency options are deemed too difficult and time-consuming for this thesis.

4. **Commodity options:** These options are based on commodities like gold, oil, or wheat. Here the valuation is tied to physical assets. Even though this is a very relevant and important option type for many businesses, it is important to note that the most common financial instruments on commodity markets are not options, but futures. Thus I do not deem the potential improvements in commodity option pricing as relevant as in other markets.

5. **Interest rate options:** The value of these options is based on changes in interest rates. This means that the GAN should model the dynamics in fixed-income markets. Financial instruments in these markets are very important for risk management in financial institutions. However, the most commonly traded instruments are interest rate swaps, interest rate futures, and credit default swaps. Even though options on bonds or interest rates are somewhat common, we still decided to not focus the research on this particular market.

6. **Cryptocurrency options:** Here the value of the option is based on the price of a cryptocurrency. These markets are extremely volatile, and it thus might be difficult to accurately create synthetic data with the GAN. In addition, cryptocurrency options are traded in relatively low volume compared to other markets.

Considering all this, we decided to focus the research on equity options. The effectiveness will be applicable to a broader market context and will help both investors and business owners. In addition, we expect the results to also be applicable to index options considering that the underlying dynamics of a single stock and an index are somewhat similar (although volatility levels may differ).

1.7 Scope & Organisation

To recap, the scope of this research is to develop a new option pricing framework. This framework combines GANs and QMC simulation and tries to limit the number of parametric assumptions. The study aims to fill gaps in the existing literature and offers potential contributions to the field of study by trying to provide a more flexible, stable, and efficient framework for option pricing in dynamic and complex financial markets. The exploration of the sub-research questions makes sure that this thesis results in a comprehensive investigation of the proposed method and its implications on option pricing. A comparative analysis of the GAN-QMC to QMC/MC will be done to assess its effectiveness in both accuracy and efficiency.

This thesis is written at Simula Research Laboratory, an institution specializing in computational science and engineering working on cutting-edge research and collaborating with industry and academia. Simula provides a good environment for exploring novel solutions to complex computational problems. Specifically, this research is conducted at the Department of High-Performance Computing, solving problems in physics, engineering, and beyond. The department's expertise in computational science makes it well-suited for exploring innovative solutions to challenging computational problems in finance.

1.8 Thesis outline

Now follows a brief overview of the content of all chapters in this thesis.

Chapter 1: Introduction

In the introduction, we briefly provided some information about option pricing, the need for new pricing techniques, and the context of the thesis within the organization. We explained why a new option pricing framework is useful. Based on this our research problem with research questions and a brief explanation of how each question will be answered, followed.

Chapter 2: Theoretical framework

In the second chapter, we discover all the information necessary about option pricing (and its challenges), (Q)MC, and GAN through a literature review. This is used both to explain the

definitions and to find out which existing developments already exist. This theory will be used to create our own GAN-QMC pricing method.

Chapter 3: Methodology

In chapter 3, we explain how to price options through MC/QMC. We try to comprehensively explain the elements necessary to understand a GAN. Finally, the pricing through GAN-QMC will be shown. All the necessary notations will be explained, and a pseudo-code of the entire pricing framework will be made. The option pricing formulas, used to price the options in the simulation, are derived for our specific method and chosen notations.

Chapter 4: Quantitative analysis

This is the chapter where we will train and tune the GAN. The GAN will then be combined into a simulation. Here we explain the data choices; which stocks and option parameters are chosen, and how this data is implemented effectively. The GAN will be analysed based on some statistical properties. In addition, the statistical measures to quantify the performance of the different approaches are chosen.

Chapter 5: Empirical results

Based on the results of chapter 4, the accuracy and efficiency of the proposed method will be compared to regular QMC/MC. We will determine whether there was any statistically significant improvement in using a GAN to generate price paths. Here we also execute a sensitivity analysis of the option's parameters to determine how sensitive the model is to the changes in the parameters.

Chapter 6.2: Conclusion & Discussion

We conclude the thesis by providing a brief summary of the main findings and all the research questions will be answered concisely. We will discuss the outcome and the impact it can have on option pricing. In addition, we discuss the limitations and assumptions of our research and give recommendations for future research.

Chapter 2

Theoretical framework

In developing the GAN-QMC approach, it is crucial to define key concepts and establish existing knowledge within this field. This is necessary for chapter 3, where we delve into the technical details of the approach, including option pricing, GAN, and QMC simulation. This chapter offers a comprehensive overview to provide clarity on how to use this approach. Section 2.1 explains the fundamental concepts of option pricing, while section 2.2 discusses challenges associated with American and exotic option pricing. We then explain the usage of QMC for option pricing in section 2.3. Section 2.4 briefly explains the main statistical properties often exhibited by real-world financial time series data. Finally, in section 2.5, we explore the applications and extensions of GAN. Any additional technical details required will be thoroughly explained in later chapters. This chapter primarily offers a broad overview of these topics, drawing upon existing literature.

2.1 Option pricing

As previously mentioned, an option is a contract granting the holder the right to buy or sell the underlying asset at a predetermined price. The issuer (seller) of the option is obligated to fulfill the terms of the contract if the buyer decides to exercise the option. A long position involves buying the option, while a short position entails selling the option. The price at which the asset can be bought or sold is referred to as the strike price. The duration of the option is called the time to maturity. The act of deciding to buy/sell the underlying asset, utilizing the option's right, is known as exercising the option. The payoff of the option is the difference between the strike price and the underlying asset's price.

An important concept is risk-neutral pricing (Bingham et al., 2001). When valuing options, we assume an environment where investors are risk-neutral (Hull, 2000). The risk preferences of the investors do not influence the price. The price is not dependent on the expected return of the underlying asset but on the risk-free rate, often derived from the yields on government bonds with low default risk. Why this works will become clearer after understanding the main parametric approach in option pricing, thus we will return to this topic later on in the chapter. Even though the goal is not to make any parametric assumptions, the risk-neutral valuation concept still holds whenever a hedge position can be made (Cox and Ross, 1976). In addition, even relaxing some of the assumptions made in many parametric pricing methods can already lead to improved results.

As mentioned earlier, there are two main methods to price options: a parametric and a non-parametric approach (Altman et al., 2009). The choice of method depends on the option's characteristics and the underlying asset. We will now explain the idea behind both approaches and some of the most common developments in these fields. While there are various pricing methods, we will focus on the main methods, which are widely used and can provide a benchmark

for option pricing. These methods are also foundational, serving as the basis for more specialized models that address specific challenges or other types of financial instruments.

2.1.1 Parametric approach

In a parametric approach, the pricing of the option is based on a pre-determined mathematical formula, describing the relationship between various variables influencing the price. In 1973, F. Black and M. Scholes derived a closed-form solution for determining the price of a European option (Black et al., 1973), for which they received the Nobel Prize for Economics. This Black-Scholes model allowed for the calculation of the theoretical value of European-style options, taking into account factors such as the underlying stock price, the option's strike price, time to expiration, volatility, and risk-free interest rates. It is a simple and efficient method, which is widely used and provides a decent benchmark for option pricing. However, it made several, sometimes invalid, assumptions. For example, it assumes that the underlying asset's returns follow a normal distribution due to modeling the asset with a geometric Brownian motion, which is discussed later in this section. In addition, it assumes constant volatility, requires specific market conditions, and does not allow for the early exercise of American options. Although it made these assumptions, it did lay the foundation for modern financial derivatives markets, enabling a better understanding, pricing, and management of options.

In 1974, R.C. Merton developed an extension of the Black-Scholes model that incorporates additional factors to provide a more realistic valuation framework for options, especially for companies that face financial distress (Merton, 1974). This model is called the Merton model. It allows for the modeling of dividend payments, which results in more accurate pricing of options on dividend-paying stocks. It also overcomes the Black-Scholes' drawback of an asset's value never dropping below zero, which is a consequence of assuming a geometric Brownian motion. The Merton model incorporates the possibility of a firm's financial distress or bankruptcy by including a firm's total asset value, which covers both the value of its assets and the value of its outstanding debt. Since the firm's asset value fluctuates, there is a risk that it may fall below the value of its debt, which can lead to a default or bankruptcy. The Merton Model takes this into account when pricing options on the firm's equity. While the Black-Scholes model is mainly used for pricing options on individual stocks, the Merton model can also price options on the equity of companies. It does this by saying that the equity of a company is essentially a call option on the company's assets, with the strike price equal to the firm's debt. In reality, the Merton model is primarily used for the valuation of corporate debt and assessing credit risk rather than option pricing. However, the Merton model did demonstrate the importance of the Black-Scholes model in understanding options and their valuation in financial markets. In addition, it was able to successfully relax some of the Black-Scholes model's assumptions.

In 1976, R.C. Merton developed another extension of the Black-Scholes model. The assumption that the asset moves continuously through time was relaxed by Merton's mixed diffusion-jump model (Merton, 1976), allowing the asset to move both continuously through time and have jump processes. The mixed jump-diffusion model is not a direct replacement for the Black-Scholes Model in option pricing. Instead, it is more useful in situations where asset price movements are better described by a combination of continuous-time and occasional jumps, such as during periods of high market volatility. The mixed jump-diffusion model can be used to enhance the accuracy of option pricing models by capturing discontinuous price movements that the Black-Scholes model does not account for. However, it can be computationally more complex and may require additional data and parameters to estimate the jump component accurately.

Boyle (1977) developed an elaborate framework for pricing options using MC simulation. It outlined a method to approximate option values by simulating numerous random price paths for the underlying asset. This improved flexibility over previous pricing methods by allowing options with complex non-standard features and variable parameters. However, MC simulation

is computationally intensive, requiring a large number of simulations for accurate results. We come back to simulations for option pricing in section 2.3.

Cox, Ross, and Rubinstein (1979) developed a discrete-time pricing model. This model is often referred to as the Cox-Ross-Rubinstein (CRR) model used for pricing various options, including those with complex features. It does this by discretizing the time to expiration and modeling the price of the underlying asset through a binomial tree structure. The CCR model was the first Binomial tree model and was therefore also a significant advancement in quantitative finance. A Binomial option pricing model is flexible and can handle a wide range of options, including those with early exercise features, dividends, and variable volatility, making it suitable for complex financial instruments. However, it can be computationally intensive and may require a large number of time steps to achieve accurate results, which can make it less efficient for options with long time horizons or very fine time intervals.

In 1993, the Bi-Heston model was made as an improvement to the Black-Scholes model, allowing the volatility to be modeled with a 2-factor structure (Heston et al., 2009). This resulted in a better fit for implied volatility and improved the option's price accuracy (Rouah, 2013). Solutions for its stochastic differential equations were derived (Fallah et al., 2019). It can capture volatility clustering and mean reversion which makes it a good fit for market data with changing volatilities. However, due to the complex mathematics and computations, it is computationally intensive. This model is mainly useful for dealing with interest rate options and other financial instruments where interest rates play a significant role in determining its price, but it shows again the foundation that the Black-Scholes model laid for quantitative finance.

We now discuss the Black-Scholes-Merton (BSM) model because this will explain the most important concepts in option pricing including risk neutrality and asset dynamics. The notations used for this thesis is as follows:

- Strike price: K
- Time to maturity: T
- Time step (for discrete-time pricing): Δt
- Initial asset price: S_0
- Asset price at Maturity: S_T
- Risk-free rate: rf or r
- (Implied) Volatility: σ Or V (In programming)
- Price of Call option: C
- Price of Put option: P
- Payoff function of any option: $f(...)$

The stock price for BSM assumes that the percentage change of its value is normally distributed if calculated over a very short time period. If we define μ as the average real-world return of the asset, it can be denoted as:

$$\frac{\Delta S_t}{S_t} \sim \phi(\mu \Delta t, \sigma^2 \Delta t) \tag{2.1}$$

In this equation, σ is the volatility of the underlying asset and Δt the small time step from period t to $t+1$. The symbol $\phi(M, V)$ is the normal probability density function where the mean is denoted by the symbol M and the variance by V. From here it can be derived that the stock price at a future time T is lognormally distributed:

$$ln(S_T) \sim \phi \left[ln(S_0) + \left(\mu - \frac{\sigma^2}{2} \right) T, \sigma^2 T \right] \tag{2.2}$$

Here S_0 is the stock price at the initial starting date of the option. Therefore, the expectation of the future stock price is:

$$E(S_T) = S_0 e^{\mu T} \tag{2.3}$$

This shows that the future stock price is a function of the initial stock price, the expected return of the underlying asset, and the time period of the option. However, as mentioned before when we calculate the value of the option, it will not depend on μ, the real return of the asset, but on a risk-neutral return. A way to show that the return x, a risk-neutral return, earned in option pricing tends to be lower than the average real-world return μ of the underlying asset is by rewriting equation 2.3 as:

$$ln[E(S_T)] = ln(S_0) + \mu T \tag{2.4}$$

For a concave function, Jensen's inequality (Needham, 1993) shows that $ln[E(S_T)] > E[ln(S_T)]$. This results in $E[ln(\frac{S_T}{S_0})] < \mu T$ and thus $E(x) < \mu$, where x is the stochastic return earned from investing in an option. This shows that the average return of the option at maturity is smaller than the average real-world return of the underlying asset. This makes sense with the concept of risk-neutral valuation, considering that the risk-free rate tends to be lower than the average return of actual assets. Cox and Ross (1976) showed that the risk-neutrality assumption holds for any parametric or non-parametric option pricing method in general and not only for the Black-Scholes model.

We will explain the concept of the BSM differential equation in a simple manner. The idea is to set up a portfolio consisting of a position in the underlying asset and the option itself. Considering that there cannot be arbitrage (Dybvig et al., 1989) in the market (which is an assumption of BSM), the return of this portfolio will be the risk-free rate (Hull, 2000). This holds because the option's price is dependent on the asset's price, and is thus, for a very short period of time, perfectly correlated with the asset. If the right quantity (will not go into detail here) of both the option and the asset is bought, the gain/loss of one offsets the gain/loss of the other, such that at the end of the short period the value of the portfolio is known. However, considering that the price of the underlying asset will constantly change, the portfolio is only riskless for a very short period of time. In order to make it riskless again, a certain quantity of the underlying asset must be bought or sold. Having these offsetting positions in options and assets is called hedging. The constantly changing risk in the portfolio, and thus the need to hedge again, is not an issue for BSM. It is assumed that the short selling of securities is allowed and that there are no transaction costs or taxes, although these do not always hold in practice.

In total, there are 7 assumptions of the BSM model about market requirements and the stochastic process of the underlying asset. One of them is which process the underlying asset follows. From the Itô process (Leiva et al., 2016), a generalized Wiener process (Hinich et al., 2010), the stock price process can be derived into the following continuous model (a discrete-time model is also available):

$$dS_t = \mu S_t dt + \sigma S_t dz \tag{2.5}$$

This equation shows that the change in asset price over a very small period has a deterministic and stochastic part. The deterministic part is the drift of the asset μ and the stochastic part is the volatility of the asset σ driven by the geometric Brownian motion (Marathe et al., 2005). All these are assumptions about the stock price process. However, the geometric Brownian motion models the returns using a normal distribution which cannot capture fat-tailed distributions of financial returns properly (Taleb, 2007). The generalized Wiener process cannot properly

capture the dynamics of linearity and time reversibility which are often how financial assets behave (Hinich et al., 2010).

From Itô's lemma (Ito, 1951), it follows that the change in the price of an option f dependent on asset S can be written as:

$$df_t = \left(\frac{\delta f_t}{\delta S_t}\mu S_t + \frac{\delta f_t}{\delta t} + \frac{1}{2}\frac{\delta^2 f_t}{\delta S_t^2}\sigma^2 S_t^2\right)dt + \frac{\delta f_t}{\delta S_t}\sigma S_t dz \tag{2.6}$$

Here dz is the same Wiener process as in the Itô's process of equation 2.5. By constructing the aforementioned portfolio of -1 times the option (by selling/shorting the option) and buying $+\frac{\delta f_t}{\delta S_t}$ (Delta Δ) number of shares it can be shown that the Wiener process can be eliminated, leaving us with the change of the value of the portfolio in discrete time:

$$\Delta\Pi_t = \left(-\frac{\delta f_t}{\delta t} - \frac{1}{2}\frac{\delta^2 f_t}{\delta S_t^2}\sigma^2 S_t^2\right)\Delta t \tag{2.7}$$

Considering that the stochastic factor Δz has been eliminated and the portfolio has been perfectly hedged, it must be riskless for a very short period of time Δt. Following from the underlying assumptions of BSM, this portfolio must earn the risk-free rate, thus:

$$\Delta\Pi_t = r\Pi_t\Delta t \tag{2.8}$$

Which after substitution and some rewriting leads to the following equation (in continuous time):

$$\frac{\delta f_t}{\delta t} + rS_t\frac{\delta f_t}{\delta S_t} + \frac{1}{2}\sigma^2 S_t^2\frac{\delta^2 f_t}{\delta S_t^2} = rf \tag{2.9}$$

This equation is the BSM stochastic differential equation, which has a closed-form solution for the European call and put option, based on their boundary conditions (their future payoffs):

$$f_{call} = max(S_t - K, 0)$$

$$f_{put} = max(K - S_t, 0)$$

when $t = T$ (the option is at maturity / expiring date).

We will not go into more detail about the mathematics behind the parametric methods. The concepts of options, risk-neutral pricing, underlying asset dynamics, and mathematical models have briefly been explained by example. However, there can be some major downsides to these approaches where assumptions must be made about the market and the mathematical process of the underlying assets (Hinich et al., 2010) (Taleb, 2007) (Chaudhary et al., 2020). It might also be hard to accurately estimate the parameters used in these models, like the risk-free rate and volatility. As mentioned above, empirical evidence has shown, that some of the assumptions necessary for parametric option pricing don't always hold, and this gives rise to the use of non-parametric approaches.

2.1.2 Non-parametric approach

In non-parametric option pricing methods no statistical or economical assumptions about the underlying assets and market are necessary. The asset dynamics are captured using empirical data or computational techniques, thus potentially overcoming the limitations of BSM. These models tend to be more complex and behave like a black-box.

Numerous advancements have taken place in using machine learning to price options. We will briefly discuss the most common ones. During a study by Hutchinson et al. (1994), they developed a neural network model for option pricing that notably outperformed the Black-Scholes

model in terms of accuracy and efficiency. This was evident through statistical measures such as RMSE, MSE, and MAE. However, when it came to very short-term options, the Black-Scholes model performed better. The study highlighted that using data allows for a more adaptive approach, reducing dependency on specific characteristics of regime data, unlike parametric models. This non-parametric model also showed more robustness to specification errors due to its lack of restrictive parameter assumptions.

Artificial neural networks have demonstrated strong performance in valuing options (Anders et al., 1998), with superior out-of-sample performance compared to Black-Scholes resulting in an average 6% higher R^2. During another study, Support Vector Regression, Gaussian Processes, and neural networks were compared to the Black-Scholes, Merton, and Heston models (Park et al., 2014). The AI models all outperformed these well-acknowledged parametric models, particularly when the options weren't significantly out of the money.

Various machine learning techniques have been applied to forecast financial time series, which can be used for option pricing. Deepak et al. (2014) proposed 4 models to predict one-day-ahead stock indices prices. The models used were Linear Correlation, Rank Correlation, Regression Relief, and Random Forest models. All of these outperformed the already proficient Proximal Support Vector Machine (SVM). A relatively new and promising model is the XGBoost algorithm, showing very positive results in the performance for option pricing (Ivascu, 2021).

Kernel Density Estimation (KDE) (Yen, 2017) is another viable method for option pricing. KDE is a flexible and data-driven approach, estimating the probability density function of a random variable based on available data. Thus, no assumptions regarding the asset's dynamics are necessary. The estimated density function can be used for option pricing. However, it is important to note that this method heavily relies on the quality and amount of available data. Furthermore, the estimated density function is sensitive to the selected input parameters and is impacted disproportionately by outliers. KDE can be a useful tool if enough high-quality data is available.

Another approach that has been successfully employed multiple times is estimating the option's price using a parametric method and making a price adjustment using a non-parametric method. Mancini et al. (2009) developed an Automatic Correction of Errors (ACE) method, where the risk-neutral distribution of the underlying asset is estimated non-parametrically. The estimated price from a parametric model can be adjusted empirically with ACE, showcasing superior accuracy compared to a selection of other pricing models.

While parametric option pricing methods often demonstrate advantages by outperforming non-parametric methods, there are notable drawbacks. Firstly, substantial historical data is necessary to effectively train these models. The amount of required data depends on the model type and the desired level of accuracy. The results presented in the aforementioned papers assume the availability of sufficient high-quality data for model training. Without adequate data, these models might not outperform parametric models. Additionally, when the asset dynamics can be well-modeled, parametric models tend to yield greater accuracy for option pricing (Ivascu, 2021). Non-parametric models also tend to lack interpretability because they often represent black-boxes. Although Gradojevic et al. (2011) successfully introduced an approach using explainable AI techniques to enhance the interpretability of random forest and XGBoost, the inherent interpretability challenges remain for many other models. The recent focus and progress in explainable AI may potentially minimize this issue in the future.

2.2 American and exotic option pricing challenges

A challenge to pricing American and many exotic options is their path dependency. Closed-form solutions exist for calculating the exact price of certain barrier options (Rubinstein, 1991) and lookback options (Goldman et al., 1979). However, these formulas assume that the price follows a geometric Brownian motion, implying a normal distribution and constant volatility for the

underlying asset (Cao et al., 2022). It has been shown that asset returns are often not normally distributed (Theodossiou, 2000). In addition, empirical evidence shows that volatility is not constant (Chaudhary et al., 2020). Hence, volatility must be modeled as a stochastic factor and thus the exotic option prices cannot accurately be calculated using these closed-form solutions.

In essence, no analytically justifiable formulas are available for pricing exotic options with non-standard or complex features, particularly when their prices are presented discretely, even when the risk-neutral underlying can be modeled through an appropriate stochastic process (Kirkby, 2019). It means that if the options have a payoff structure that is for example path dependent, or you want to model the real-world asset's price process, closed-form solutions are not valid. Therefore there is a need for numerical methods to price these options. MC simulation offers a viable approach to valuing price path-dependent options while allowing for the incorporation of multiple assets, time-varying parameters, uncertainty, and stochastic processes (Grant et al., 1997).

MC simulations are extremely flexible for option pricing. A slight modification, allows you to price a completely new path-dependent option. Notably, it has been shown that MC is more robust compared to many other numerical methods, although with slower convergence (Caflisch, 1998). There exist many other numerical methods suited for pricing American and exotic options including the binomial tree model, finite difference, or the lattice method (Broadie et al., 1997). MC is often used, if the problem's complexity is too hard to implement with the binomial tree or lattice method (Joy et al., 1996), making it appropriate for highly complex financial instruments. When controlling for computational time, the accuracy of MC tends to be lower compared to the binomial tree and the finite difference method (Ding et al., 2017). To compensate the number of trials must be increased significantly, resulting in higher computational times, but making it equally accurate and more robust.

The idea is that if GANs can develop price paths that can more closely resemble the underlying asset compared to the geometric Brownian motion's price paths, the accuracy of a MC simulation can be improved by incorporating a GAN. This improvement aims to leverage QMC's and MC's positive properties while improving its convergence speed, which is important for accurate and efficient option pricing.

2.3 (Quasi-)Monte Carlo simulation

We have already discussed some of the advantages and disadvantages of MC simulation, but it is important to understand how it works in order to use it effectively for option pricing.

Stevens (2022) describes MC simulation as an 'experimental' calculation that uses, oftentimes billions, of random numbers to do experiments. The approach involves executing numerous trials using the mathematical model underlying the process of interest. The input parameters for each trial need to be carefully selected to adhere to the desired statistical distributions. Output values from each trial are saved, allowing relevant statistical measures like averages and confidence intervals to be calculated. For option pricing the assumption of risk neutrality is used to calculate the rate of return of the option's asset in equilibrium, from which the option's price can be calculated (Boyle, 1977). Fu et al. (2009) developed an approach for a sensitivity analysis of MC simulations.

Recent high-end applications of MC simulation showcase its practical utilities. For instance, MC simulation is used in distributed filtering, utilizing the lattice rule for the underlying mathematical model (S. Li et al., 2023). Kreuze et al. (2023) modeled data dynamics, with missing values, using multivariate random models based on copulas.

Standard MC simulations typically use pseudo-random numbers (Niederreiter, 1978) for each trial, often in combination with some underlying assumption of the mathematical model, such as assuming that the underlying asset price follows a geometric Brownian motion. To improve the asymptotic error rate, Birge (1995) developed the QMC method. In QMC simulation,

a sequence of deterministic numbers is used, called quasi-random numbers. This improves convergences and results in deterministic error bounds instead of probabilistic bounds (Joy et al., 1996), which has been shown through the pricing of basket and Asian options.

For QMC, using quasi-random numbers, a so-called low discrepancy sequence must be generated. In simple terms, this property allows the creation of evenly spaced numbers that can fill gaps created by previous numbers in the sequence (Joy et al., 1996). This is useful to simulate many diverse price paths that the underlying asset could follow without simulating many similar price paths. It improves convergence by requiring fewer trials to fill in 'missed' price paths compared to the number needed by using a pseudo-random sequence. Additionally, this approach allows for continuing the simulation until the desired level of accuracy is achieved. Achieving a reasonable level of accuracy involves balancing the number of trials and the width of the confidence interval of the accuracy. If a large percentage increase in trials is necessary to improve accuracy by a small percentage, the simulation has likely found a reasonable balance between accuracy and computational efficiency. It should be noted that for path-dependent options higher dimensional sequences are required.

To demonstrate the difference between the pseudo-random numbers of MC and the quasi-random numbers of QMC, figure 2.1 shows a comparison between sampled numbers from both methods. The MC numbers look completely random but will be biased through clumpiness with too few samples. On the other hand, the QMC numbers still appear random but not independent and most importantly, not clumpy.

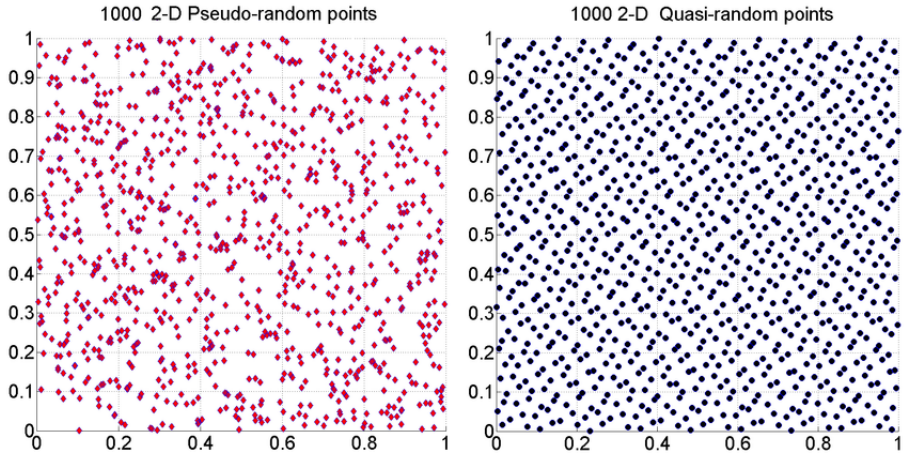

Figure 2.1: MC vs QMC

An important question here is, which quasi-random sequence is most effective for generating deterministic numbers for the trials in a simulation for option pricing? For the pricing of financial instruments, Faure sequences (Vandewoestyne et al., 2010) tend to offer some advantages compared to other methods for generating quasi-random sequences (Fox, 1986) (Niederreiter, 1992). However, for higher dimensions Sobol sequences outperformed sequences like Halton and Faure (Morokoff et al., 1995). The random sequence in MC can simply be replaced by the Sobol, Halton or Faure sequence, making their implementation relatively straightforward. Faure sequences can also be used to generate correlated random variables (Joy et al., 1996). This is useful for pricing options whose value is derived from multiple, often correlated assets, like basket options. However, this is not researched in this thesis. The most important aspect here is to remain consistent in the choice of quasi-random sequence for a fair comparative analysis. We come back to this in chapter 3.

Pricing American options with QMC is not that straightforward, due to the possibility of early exercise. Two viable methods are least-squares MC and an averaging method using lower and upper bounds. Dion et al. (2010) proposed a framework combining least-squares

MC (Longstaff et al., 2001) with randomized QMC. We will apply a similar method combining least-squares with QMC. Least squares MC is a two-step procedure, using backward iteration starting at maturity. It compares the future expected cash flow, calculated from least-squares, with the current payoff if the option is exercised, at each time step. By discounting this value back to the option's starting date for every time step, we can determine the exercise date that yields the highest expected future payoff. Using these payoffs, we can calculate the option's price. It is important to note that the price from this simulation represents a lower bound. A detailed explanation of least-squares MC will be given in section 3.1.

Alternatively, the price can be determined by setting up a lower and upper bound using the risk-neutral property. Taking the average of these bounds can result in a reasonable price estimate of an American option. The lower bound is simply the payoff of the option at maturity, thus not exercising it early and it therefore has the value of a European option. The upper bound is the future expected payoff at maturity, which should be calculated as a risk-neutral expectation, assuming that the payoff is higher than the value of the underlying asset (i.e. don't exercise options with a negative payoff) (Chen et al., 2002). This implies that any function that consistently remains higher than the maximum of continuation and early exercise must be an upper bound to the option's value, assuming continuous time. Although valid, it is difficult to determine which function would be effective, considering that you need to accurately estimate the future expected payoffs using a prediction model. Setting up a function that satisfies the requirement of being higher than the maximum at all time steps, without overshooting the targets significantly, is not trivial. Alternatively, it can be seen that under the risk-neutral assumption (and no-arbitrage), the maximum payoff of an American option is its price invested at the continuously compounded risk-free rate until maturity (which must hold for any option). The formulas of the lower and upper bound will be derived in section 3.1.

All the extra necessary technical details regarding QMC are discussed in chapter 3 and 4.

2.4 Statistical properties of financial time series

The GAN is used to generate financial data. This data should have similar statistical properties to the real-world financial time series data. Therefore it is important to test GANs on their ability to capture the most common statistical properties of financial return time series as defined by Cont (2001) and Takahasi et al. (2019).

Linear unpredictability

Linear unpredictability in a time series implies that there is no auto-correlation in the returns time series. The auto-correlation function is denoted as follows:

$$\frac{E[(r_t - \mu)(r_{t+k} - \mu)]}{\sigma^2} = Corr(r_t, r_{t+k}) \approx 0, for\ k \geq 1 \tag{2.10}$$

Whenever returns show a low degree of auto-correlation it indicates that the market is efficient (Chakraborti, 2011). Testing for significant auto-correlation can be done using the Ljung-Box test, proven to have good small sample properties (Brooks, 2014). The Ljung-Box test statistic is denoted as follows:

$$Q^* = T(T+2)\sum_{k=1}^{m} \frac{\hat{\tau}_k^2}{T-k} \sim \chi_m^2 \tag{2.11}$$

Here $\hat{\tau}_k$ represents the sample autocorrelation for lag k. Under the Null hypothesis, the time series does not show any significant sign of auto-correlation up to lag m and Q^* is chi-square distributed with m degrees of freedom.

Fat-tailed distribution

A fat-tailed distribution implies that there is more density in the tails of the distribution, and thus a larger probability of extreme events (returns), than a normal distribution. It has been shown that a probability distribution denoted as P(r) has tails with a power-law decay denoted as:

$$P(r) \propto r^{-\alpha} \tag{2.12}$$

The distribution of financial returns tends to have an exponent $\alpha \geq 3$ (Chakraborti, 2011). Preferably the α of the Generator's distribution should be somewhat similar to the real-world data, indicating a similar density in the tails.

To test for fat-tails, test the Null hypothesis that $\alpha \leq 3$ versus the alternative hypothesis that $\alpha > 3$. Fit the power-law distribution with $\alpha = 3$ and a floating alpha equal to the estimate of the data. Calculate the log-likelihood $log(L(A))$ for each fitted model and calculate the log-likelihood test statistic:

$$TS = 2\left[log(L(model_{\alpha>3})) - log(L(model_{\alpha=3}))\right] \sim \chi_1^2 \tag{2.13}$$

Considering this is a one-sided test, we can test as follows:

1. $H_0 : \beta \leq 0$ where $\beta = \alpha - 3$
2. $H_1 : \beta > 0$
3. Define $p - val_1$ as the p-value of the one-sided test.
4. Define $p - val_2$ as the p-value of the log-likelihood test statistic TS.
5. If $\beta > 0$ then $p - val_1 = \frac{p-val_2}{2}$
6. If $\beta <= 0$ then $p - val_1 = 1 - \frac{p-val_2}{2}$

The Null hypothesis is rejected if β is larger than 0 and the p-value is small. If $\beta > 0$ and $p - val_1 < 0.05$, then we reject the Null hypothesis and conclude that with a 95% level of confidence, the power law exponent α is larger than 3, and the data has a fat-tailed distribution. Considering that this test is an adjusted 2-sided test, we actually test $H_0 : \beta = 0$ vs $H_1 : \beta \neq 0$. Thus if $\beta < 0$ and $p-val_1 < 0.05$ the data has less density in the tails than a normal distribution.

Volatility clustering

Volatility clustering is another commonly occurring property of financial time series. Volatility clustering means that small/large price fluctuations temporarily cluster together with other small/large fluctuations (Brooks, 2014). This implies that the absolute returns show some form of auto-correlation. This can also be measured using the power-law decay as follows:

$$Corr(|r_t|, |r_{t+k}|) \propto k^{-\beta} \tag{2.14}$$

The exponent tends to be slightly positive such that $0 < \beta < 1$.

A simple test for volatility clustering is to apply the Ljung-Box test on the absolute returns $|r|$. The test statistic remains the same, but the sample auto-correlation $\hat{\tau}_k$ is calculated on the absolute returns. Under the Null hypothesis, the absolute returns do not show any significant auto-correlation, implying that there is no volatility clustering present in the data.

Leverage effect

The leverage effect observed in the market means that returns are negatively correlated with future volatility (Bouchaud et al., 2001). This occurs because a firm's profitability or value is amplified by the use of financial leverage (debt), leading to greater fluctuations in returns, both positive and negative. This can be quantified with a lead-lag correlation function:

$$L(k) = \frac{E[r(t)|r(t+k)|^2 - r(t)|r(t)|^2]}{E[|r(t)|^2]^2} \tag{2.15}$$

Almost all stocks exhibit some form of leverage effects, but whether the correlation is negative or positive depends on the market conditions (Qiu et al., 2006). This measure tends to follow an exponential decay as well (Bouchaud et al., 2001). Testing for the statistical significance of leverage effects is hard. A simple alternative is to analyse a graph to determine whether the data shows signs of leverage effects.

Coarse-fine volatility correlation

This property involves the correlation between 2 types of volatility, namely between coarse and fine volatility. These are defined respectively as:

$$v_c^\tau(t) = \left| \sum_{i=1}^{\tau} r_{t-i} \right| \tag{2.16}$$

$$v_f^\tau(t) = \sum_{i=1}^{\tau} |r_{t-i}| \tag{2.17}$$

It can be shown that fine volatility can, to some extent, predict coarse volatility in financial markets (Muller et al., 1997). This can be measured through the difference in the lead-lag correlation function, which often exhibits negative asymmetry. The lead-lag correlation function is defined as:

$$\rho_{cf}^\tau(k) = Corr(v_c^\tau(t+k), v_f^\tau(t)) \tag{2.18}$$

Then we have the following negative asymmetry:

$$\Delta\rho_{cf}^\tau(k) = \rho_{cf}^\tau(k) - \rho_{cf}^\tau(-k) \tag{2.19}$$

If $\Delta\rho_{cf}^\tau(k)$ is negative then fine volatility has the ability to predict coarse volatility. In financial markets, it is expected that the coarse-fine volatility correlation $\Delta\rho_{cf}^\tau(k)$ is slightly negative (Muller et al., 1997). This property is very noisy, making it oftentimes necessary to measure by averaging multiple time series. Again, testing for statistical significance of coarse-fine volatility correlation is difficult, instead, graphs could be analysed to determine its presence in the data.

Gain/loss asymmetry

The gain/loss asymmetry refers to the phenomenon where the decline in stock price occurs at a faster rate than the increase in price. This can be measured by counting the number of time-steps t' until a predefined stock price return θ is reached starting at time t (Jensen et al., 2003):

$$T_{wait}^t(\theta) = \begin{Bmatrix} inf\{t'|log(p_{t+t'}) - log(p_t) \geq \theta,\ t' > 0\}\ (\theta > 0) \\ inf\{t'|log(p_{t+t'}) - log(p_t) \leq \theta,\ t' > 0\}\ (\theta < 0) \end{Bmatrix} \tag{2.20}$$

The probability distributions of $T_{wait}^t(\theta)$ for $\theta > 0$ and $\theta < 0$ should show that the peak of negative returns occurs slightly before the peak of positive returns. This property is also very

noisy and needs to be analysed by averaging multiple-time series. In financial markets, negative returns decline at a faster rate than positive returns increase (Jensen et al., 2003). Determining the statistically significant presence of gain/loss asymmetry might be best tested using graphs as well.

Preferably, the GAN will generate data with statistical properties similar to the real-world data at least with regards to a fat-tailed distribution, linear unpredictability, leverage effect, and volatility clustering. Coarse-fine volatility correlation and the gain/loss asymmetry are less of a priority due to their measuring difficulties.

2.5 Generative Adversarial Networks

In this section, we discuss extensions of GANs and explore their applications in the realm of finance. Goodfellow et al. (2014) developed the first GAN. It is an unsupervised learning model consisting of a Generator to create synthetic data and a Discriminator determining whether some proposed data is real or fake. Both the Generator and Discriminator use neural networks to carry out their tasks. Trained simultaneously, the goal is for the Generator to create samples so realistic that the Discriminator cannot distinguish them from real data. Important technical details of GANs are discussed in section 3.2.

Starting with its applications, a GAN-based stock price prediction framework was developed by Zhou et al. (2018). This framework aimed to predict price movements in high-frequency stock market data by minimizing the Discriminator's loss, combining a Long Short-Term Memory (LSTM) and a convolutional neural network. Some GANs outperformed models like an ARIMA-GARCH, artificial neural network, and a SVM on Root Mean Square Relative Error (RMSRE) but not on Direction Prediction Accuracy (DPA). Only the GAN-Fusion Discriminator (GAN-FD) framework was able to beat the other methods in most scenarios. Hence, building a GAN that universally outperforms other models, consistently captures important statistical properties well, and generates the most accurate data proves to be challenging.

GANs have also proven to be effective in financial fraud detection outperforming other classification methods (Zheng et al., 2018). Due to its success, this GAN has been implemented in the fraud detection systems of multiple banks. For deploying efficient trading strategies, Koshiyama et al. (2020) developed a framework that can be fine-tuned using a conditional GAN. This approach outperformed alternatives like an ensemble scheme and other model validation methods, particularly when those methods could not generate a positive alpha, referring to the return on investment greater than the expected return given the level of risk involved. To quantify the effectiveness of trading strategies, Coletta et al. (2021) built a simulation using a GAN to create orders, trying to mimic market realism and responsiveness, to test the effectiveness of a trading strategy.

While GANs have successfully reproduced financial market data, their effectiveness depended on the selected underlying neural network (Eckerli et al., 2021). It was shown that a GAN struggles to capture all statistical properties of financial time series data, such as heavy-tails in the probability density function, volatility clustering, and average returns around zero. Some GANs perform better at capturing specific statistics but struggle with others. None of the tested GANs was able to capture all these properties simultaneously. Furthermore, generating data with similar statistical properties did not necessarily imply that the generated returns could be used to construct realistic price paths. Therefore, when assessing its performance, it was not sufficient to only analyse some performance metrics but to also qualitatively analyse graphs showing the generated returns, price paths, probability density functions, and auto-correlation functions.

X. Li et al. (2021) combined a Gaussian Graph Model with a GAN to predict future stock prices for some Chinese markets, outperforming other modeling options like ARIMA-GARCHA, SVM, LSTM, GAN, and GGM-LSTM. The Generator used a LSTM neural network and the

Discriminator a Convolutional Neural Network (CNN). Although it outperformed the other models significantly, their developed GAN still struggled to generate realistic time series data far into the future, especially when the training set size was smaller than the desired length of the generated series. It appears that this GAN was mainly effective for generating relatively short time series paths, where generating a couple of months' worth of trading data seems doable, provided the test set was sufficiently large. Kumar et al. (2021) build a similar GAN combining LSTM and CNN, resulting in improved prediction accuracy. It outperformed LSTM by 4.35% and reduced computational times by 78 seconds (roughly a 200-second time for LSTM). Soleymani et al. (2022) developed a temporal-GAN using both LSTM and temporal CNN, resulting in a solid performance for stock price predictions, forecasting 20 or 30 days ahead. K. Zhang et al. (2019) built a GAN using LSTM for the Generator and Multi-Layer perceptron for the Discriminator to predict future S&P500 values, showing promising results compared to some other machine learning models. Different researchers also used the same neural networks for their GANs and it outperformed other models on 11 out of 36 different stock datasets (Y. Zhang et al., 2021). This suggests that GANs can accurately capture asset dynamics, but that their effectiveness depends on the specific stock they are trained on. Vuletic et al. (2023) designed a new type of economics-driven loss function for the Generator, more suitable for classification tasks. It was shown to be effective for generating equity data but the degree of effectiveness was also dependent on the specific dataset used. LSTM, a type of recurrent neural network, seems promising for generating financial time series data. However, using a recurrent neural network for the Discriminator tends to destabilize the training process more (Metz et al., 2017). This is a significant reason why some researchers use a LSTM for the Generator and a convolutional neural network for the Discriminator.

W. Wang (2022) combined MC simulation and a GAN for pricing European puts/calls, American puts/calls, futures, and forwards. The model was trained on historical stock price data, with an adjustable training set size to manually find a balance between training loss and mode collapse. Once the GAN was trained and optimized, synthetic financial data was created. Considering that recent stock data should have a larger impact on the option price, the time series were ordered based on their similarities. The series closely resembling recent market behavior were used in the MC simulation to price the options using risk-neutral valuation. This technique appears promising and interesting for further exploration due to its impressive accuracy. It did not take into account the pricing efficiency through computational time and it did not use QMC. Thus leveraging the capabilities of GANs in combination with the aforementioned strengths of QMC, may result in improved accuracy and computational efficiency of option pricing. How the GAN is combined with the QMC is explained in section 3.3.

A critical aspect in all these GAN applications is the need for proper training to mitigate mode collapse, a significant challenge for GAN and probably its most significant drawback (Lala et al., 2018). Many extensions have been proposed, trying to improve the learning stability or the output quality in general. One of these relevant extensions is conditional GAN (Mirza et al., 2014). Here, the data generation process is improved by the ability to supplement it with extra information that can be conditioned on both the Generator and Discriminator. Another extension is the Wasserstein GAN (W-GAN) (Chintala et al., 2017), where researchers proposed a technique to improve the learning stability by changing the loss function, which can help in battling mode collapse. The W-GAN was improved by adding a gradient penalty (W-GAN-GP), which further improved training stability resulting in faster convergence and a better ability to generate high-quality samples (Gulrajani et al., 2017). Especially useful for image generation, Radford et al. (2015) combined Deep Convolutional neural networks and GAN (DCGAN), achieving the generation of higher quality images. Marti (2019) developed a GAN capable of generating financial correlation matrices based on the returns of financial assets, valuable for risk management and trading strategies. Zoufal et al. (2019) developed a Quantum GAN, where the training data must be loaded into quantum states. This approach appeared

promising for financial applications, yielding similar results to competing methods while using only a quarter of the data, assuming that the data can be loaded into quantum states. Self-attention GAN (SAGAN) uses convolutional neural networks and adds a self-attention layer, allowing the Generator to use data from all feature locations, which improves the generation of high-quality images (H. Zhang et al., 2018). For a complete overview, Z. Wang et al. (2020) made a convenient figure showing all existing GAN types sorted by architectures and loss functions, see figure B.1 in appendix B.

Chapter 3

Methodology

The goal of the methodology chapter is to give a clear and structured overview of all steps and decisions taken to answer the main research question. We start by explaining the concept of pricing European and American options through a MC simulation. Then we explain some technical concepts of the GAN and make several architectural decisions. Lastly, we discuss how a GAN can be combined with a MC simulation to construct risk-neutral price paths using stock data. The decisions made in this chapter are important because they can influence the quality of the results for option pricing.

Figure 3.1 shows a concise overview of all the important concepts related to pricing options through MC, QMC, and GAN-QMC. Throughout this chapter, all these concepts are explained to give more clarity on our thesis topic.

3.1 Pricing options with MC/QMC simulation

We start by explaining how to price options using a MC/QMC simulation. This gives us a deeper understanding necessary for combining MC/QMC with a GAN. The basic process of pricing an option through MC/QMC (Boyle, 1977) is presented in figure 3.2. It starts by setting up the payoff of each option type and collecting relevant data for the stock of a company. Subsequently, we need to make an assumption about the process of the underlying asset. Using this stochastic process, random price paths are then generated through the simulation, allowing the calculation of the payoff at each price path, from which the average payoff is obtained. Discounting this back to the initial time step T_0 yields the estimated price of an option. Pricing options through MC/QMC results in an expected present value of the payoffs of an option, the estimated option price. The more price paths are generated, the closer this estimate will be to the theoretical option price under the risk-neutral framework. The rate of convergence, the speed at which the theoretical value of the option price is reached, is the square root of the number of simulated paths, often denoted as $O(\sqrt{N})$ (Jabbour et al., 2011).

Due to the lack of available data for exotic options, the choice was made to focus this thesis on European and American options. In the following 2 sections, the formulas to price these options through MC/QMC simulation are discussed. In section 3.1.3, we discuss how the simulated asset prices are calculated.

3.1.1 European options

The payoff of an option is the difference between the strike price and the underlying asset's price (Hull, 2000). The price of an option, under the risk-neutrality framework, is the option's payoff discounted using the risk-free rate. The obtained price is only risk-neutral if the asset is simulated using a stochastic process with a risk-neutral drift. The call and put options have

Figure 3.1: Overview of Methodology

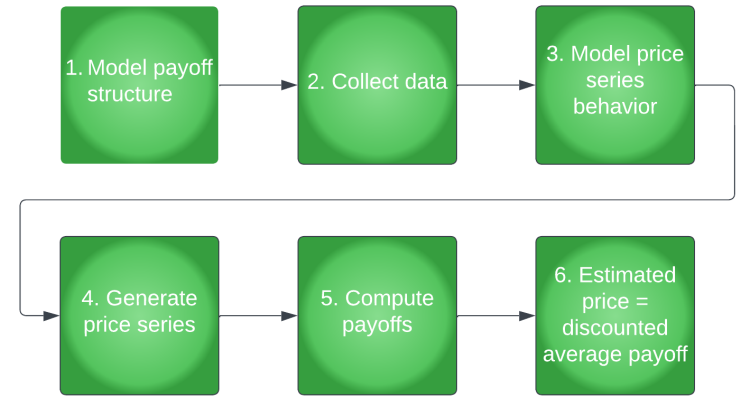

Figure 3.2: Outline QMC valuation

payoffs of $max(S_T - K, 0)$ and $max(K - S_T, 0)$ respectively. So the prices of a European call and put option are:

$$Call_{eur} = e^{-rT}\hat{E}[max(S_T - K, 0)] \tag{3.1}$$

$$Put_{eur} = e^{-rT}\hat{E}[max(K - S_T, 0)] \tag{3.2}$$

It can be seen that e^{-rT} is the discount factor with the risk-neutral rate r and $\hat{E}[max(K-S_T, 0)]$ is the expected risk-neutral payoff at time T. By assuming a geometric Brownian motion with standard normally distributed returns, it can be derived that the stock price itself follows a lognormal distribution:

$$S_T = S_0\, e^{(r-\frac{\sigma^2}{2})T+\sigma W_T} \tag{3.3}$$

Here $(r - \frac{\sigma^2}{2})T + \sigma W_T$ is $N(0, \sigma^2 T)$ distributed. This can lead to the Black-Scholes integral to price European options, see appendix A. However, most path-dependent options lack an exact solution to the integral. To price these options we simulate many price paths to calculate the risk-neutral expectations in equations 3.1 and 3.2. How these price paths are simulated, to calculate the estimated option prices, will be discussed in section 3.1.3.

3.1.2 American options

The second type of option we price in this thesis is the American option. This option gives you the right to buy or sell the underlying asset at any time during the option's lifetime (Brennan et al., 1977). On the publicly traded market, American option prices are almost always higher than their European counterpart. We now show why this seems counter-intuitive at first.

If we construct a portfolio with value V consisting of an American call option and cash needed to buy the underlying asset at maturity:

- Call option payoff when exercised: $S_T - K$
- Cash grows at the risk-free rate to K at maturity: Ke^{-rT}
- Portfolio value if exercised at time $t < T$: $V = (S_T - K) + Ke^{-r(T-t)} < S_T$
- Portfolio value if exercised (or not) at T: $V = max(S_T - K, 0) + K = max(S_T, K) \geq S_T$

This shows that the theoretical value of this portfolio is always greater (or equal) than the value of S_T when the option is held until maturity. Consequently, the option holder should not exercise the American option early. In a risk-neutral and no-arbitrage scenario, an option should, at most, yield the risk-free rate. This maximum payoff is only achieved by holding the option until maturity. There are several reasons why this discrepancy in price might occur like the added flexibility, interest rates (change in cost of carrying/rollover rate), or dividends on the underlying asset.

As outlined in chapter 2, there are 2 main methods to price American options. We will apply and compare both least-squares MC and the averaging method.

Least-squares

American options can be priced using a least-squares regression in a MC simulation (Longstaff et al., 2001). Considering that the future asset price is unknown, we predict its future expected value using least-squares. The idea is to compare the continuation value, which is the expected future value when holding the option (not exercising) until the next time step $t+1$, with the immediate exercise value at t, the payoff when exercising it now. Let t_e denote the period in time at which we choose to exercise the option, i.e., the time we anticipate the payoff to be maximal. Then the price of the American call and put options can be determined as:

$$C_{am} = e^{-rt_e}\hat{E}[max(S_{t_e} - K, 0)] \tag{3.4}$$

$$P_{am} = e^{-rt_e}\hat{E}[max(K - S_{t_e}, 0)] \tag{3.5}$$

However, how to determine the continuation value, considering that the future stock price is unknown? For this, we predict the future expected cash flow using a least-squares model (Watson, 1967). By regressing the future cash flow on the current stock price, a least-squares model is obtained. Using this model, the conditional expectation of the cash flow in time step $t+1$ given the stock price in time step t can be calculated as follows:

$$E(CF_{t+1}|S_t) = w_0 + w_1 S_t + w_2 S_t^2 \tag{3.6}$$

This equation is the expected cash flow in the next time step, based on the current stock price. The parameters w_0, w_1, and w_2 are estimated using OLS by using the actual obtained cash flow CF_{t+1} from the next period and the current stock price S_t. Once the random price paths are generated, we must calculate the cash flows recursively working our way backward. The cash flow in the last time period is known, namely $max(S_T - K, 0)$ for a call. Using the discounted cash flow of $t+1$, and the known stock price in period t, estimate the linear regression of equation 3.6. Use this regression to calculate the expected continuation value. Compare this continuation value with the immediate exercise payoff. Use the higher of the immediate exercise payoff and the expected continuation value for the new cash flow. Repeat this process iteratively all the way back to $T = 1$. This recursion allows us to determine, for every price path, the optimal exercise period denoted as t_e to maximize the payoff. Hence, during the backward progression, we can adjust the option exercise time period, should a more profitable opportunity arise. Now we just discount the cash flow from the exercise date, back to the purchase date of the option. This computation results in the estimated price of an American option, similar to equation 3.4 and 3.5.

Averaging method

As an alternative to the least-squares MC, we will also price the American options with a simpler method, namely the averaging method. In chapter 5, both methods will be compared based on their pricing accuracy. In this averaging method, the price is calculated by averaging the upper and lower bounds of the American option price. Using the principle of risk neutrality,

an obvious statement is that the lower bound of the call must be less than or equal to the average payoff at maturity (derived from e.g. a simulation). This lower bound must be lower or equal to investing cash equal to the price of the call at the risk-free rate (risk-neutrality assumption). This can be denoted as:

$$C \leq \frac{1}{N}\sum_{i=1}^{N} max(S_{i,T} - K, 0) \leq (1 + r\Delta t)^T C \tag{3.7}$$

Based on this the lower and upper bound of an American call can be calculated:

$$1 \leq \frac{\frac{1}{N}\sum_{i=1}^{N} max(S_{i,T} - K, 0)}{C} \leq e^{rT} \tag{3.8}$$

$$\left(\frac{1}{N}\sum_{i=1}^{N} max(S_{i,T} - K, 0)\right)^{-1} \leq C^{-1} \leq \frac{e^{rT}}{\frac{1}{N}\sum_{i=1}^{N} max(S_{i,T} - K, 0)} \tag{3.9}$$

$$e^{-rT}\frac{1}{N}\sum_{i=1}^{N} max(S_{i,T} - K, 0) \leq C_{am} \leq \frac{1}{N}\sum_{i=1}^{N} max(S_{i,T} - K, 0) \tag{3.10}$$

In the limit of N, this becomes:

$$e^{-rT}\hat{E}[max(S_T - K, 0)] \leq C_{am} \leq \hat{E}[max(S_T - K, 0)] \tag{3.11}$$

This shows that the lower bound of an American call option is the price of the European call option. The upper bound is the expected payoff at maturity as if the option was executed at the initial time $t = 0$ and thus the payoff would have been received at the purchasing date of the option. The least-squares American option prices are compared to the following average prices:

$$C_{am} = \frac{1 + e^{-rT}}{2N}\sum_{i=1}^{N} max(S_{i,T} - K, 0) \tag{3.12}$$

$$P_{am} = \frac{1 + e^{-rT}}{2N}\sum_{i=1}^{N} max(K - S_{i,T}, 0) \tag{3.13}$$

These equations will be solved by simulating many price paths in the MC/QMC simulation (identical to solving the European option equations), see section 3.1.3. The American put and call prices from this averaging method will be compared to the least-squares pricing method to determine which is more accurate and computationally efficient.

3.1.3 Simulate the option price

To price options using MC or QMC simulations, a step-by-step process can be followed by combining the results of the previous sections:

1. **Model the Stock Price:** Begin by modeling the stock price using a geometric Brownian motion. The underlying assumption is that the returns of the asset follow a standard normal distribution. For a small time step (Δt), this can be expressed using the BSM model (Hull, 2000):

$$ln(S_{t+\Delta t}) = ln(S_t) + \left(r - \frac{\sigma^2}{2}\right)\Delta t + \sigma\epsilon_t\sqrt{\Delta t} \tag{3.14}$$

   Here, ϵ_t represents a standard normal random variable. This equation is essential for modeling the dynamics of the stock price over time.

2. **Discretize the Process:** In order to generate price paths, we discretize the model by restructuring it as follows:

$$S_{t+1} = S_t e^{\left(r - \frac{\sigma^2}{2}\right)\Delta t + \sigma\epsilon_t\sqrt{\Delta t}} \tag{3.15}$$

   This equation provides a framework for sampling future price paths at discrete intervals.

3. **Generate Price Paths:** For MC, the creation of price paths involves the sampling of many random numbers from a standard normal distribution, and these values are then inserted into the equation described in the previous step. This process is repeated numerous times, denoted as the number of trials, to generate a distribution of potential future stock prices.

4. **Utilize Quasi-Monte Carlo:** Alternatively, for QMC, sampling from the standard normal distribution is done using quasi-random numbers. Here we use the Halton sequence because it's easy to implement and shows equal performance to other sequences for low dimensionality problems (Fox, 1986). These quasi-random sequences aim to reduce the discrepancy between points within an interval, resulting in more efficient sampling as discussed in section 2.3, effectively minimizing the maximum distance between 2 points within the interval. Rather than sampling a purely random number (MC), a smarter choice is made based on the previous number to minimize the distance between all points within a certain interval (QMC).

5. **Price the Options:** Once numerous price paths have been generated using either MC or QMC, option pricing formulas 3.1, 3.2, 3.12, and 3.13 are calculated. Here the risk-free rate and implied volatility need to be estimated (Bianconi et al., 2015).

6. **Estimate Option Prices:** The outcome of these calculations yields estimated option prices based on the MC or QMC simulations. These estimates will then be compared to the real-world prices and estimated prices obtained using other pricing methods, such as GAN-QMC or alternative pricing models. This comparison helps assess the accuracy and computational efficiency of the regular MC/QMC simulation approach, which are used as a benchmark pricing method for the GAN-QMC.

In summary, the pricing of options through MC or QMC simulations involves modeling stock price dynamics, discretizing the process, generating price paths, applying the option pricing formulas, and ultimately estimating option prices based on the simulations. These estimated prices can be used for comparative analysis against our GAN-QMC method.

3.2 GANs

In chapter 2, some relevant literature about option pricing, MC simulation, and GANs has been discussed. In this section, we explain how a GAN works, which is necessary to effectively develop a GAN in chapter 4. This section explains how a GAN operates by explaining its different components and their functions. We use all these concepts to build several GANs to determine if we can effectively implement a GAN into a MC simulation to price options.

3.2.1 Neural networks

As explained in chapter 2, a GAN consists of 2 neural networks. To understand the GAN-QMC method, it is important to also understand the fundamentals of a neural network. Therefore, we will explain a simple neural network without using overly complicated notations. A simple neural network has been visualized in figure 3.3.

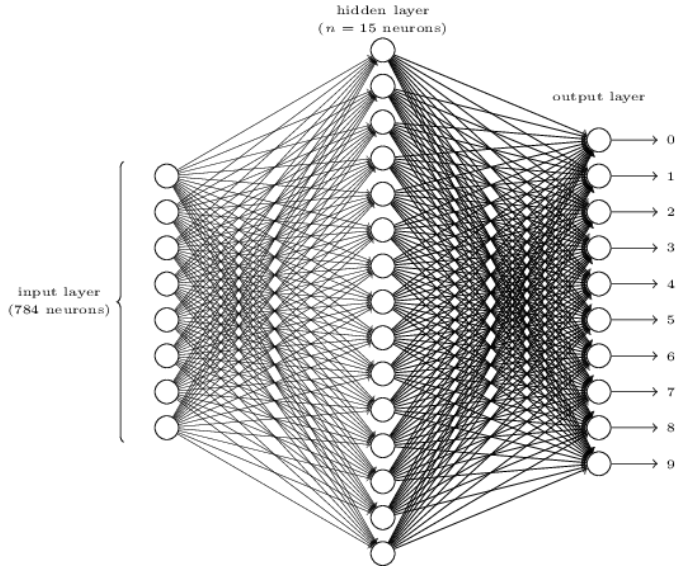

Figure 3.3: Simple neural network

A neural network consists of many neurons (Dongare et al., 2012). A neuron can be thought as of something that holds a number between 0 and 1 (although it can vary as explained below). The value in a neuron is called the activation. The neural network starts with a vector of neurons all corresponding to one of the input values of a series. For instance, one neuron could represent the stock price for a single trading day, and the entire vector, perhaps with a length of 252, could correspond to a trading year considering that there are 252 trading days in 1 year. This input vector is the first layer of a neural network. How the input vector is mapped to have values between 0 and 1 is explained at the end of this section. The output vector also consists of neurons, equal to the desired length of the output vector. So for generating a 1 trading month ahead forecast, the output layer is a vector of size 21, consisting of 21 neurons. This is commonly called the output layer. The value between 0 and 1 of a neuron in the output layer can be interpreted as the probability of an event or class occurring.

Between the input and output layers, there are so-called hidden layers (Dongare et al., 2012). The number of hidden layers is typically determined through experimentation, depending on the specific problem at-hand. The activations in one layer determine the activations in the next layer, transmitting information from the input layer through all hidden layers to the output layer. Therefore, the pattern of activations in the input vector causes some pattern in the first hidden layer, which then causes some pattern in the next, all the way until the output layer. The function of the hidden layers also depends on the specific problem. The hidden layers have specific architectures to capture temporal dependencies and patterns in the data. For stock price prediction using an input vector of 252 days, the hidden layer could for example group the 252 days into 21 batches calculating its average, from which the output layer can be calculated. This is just an example for illustration purposes. How a stock price prediction network actually behaves is totally dependent on factors like the number of hidden layers, the input and output vector size, and the statistical properties of the training data. A neural network is basically a black-box and it is usually the case that the exact functions of the hidden layers are not fully understood.

The hidden layer consists of several neurons such that it can supply meaningful information to the next layer. For simpler networks with e.g. fully connected layers, each individual neuron is connected to all neurons in the next layer (Sainath et al., 2015). Although this is not the case for e.g. recurrent and convolutional neural networks, the idea remains similar. Each connection is assigned a weight parameter such that every neuron in the first layer has a weight for every

connection to each neuron in the second layer. You can imagine that the number of weights in a neural network with an input size of 252 and an output size of 21 with 2 hidden layers is extremely large. The activation value of the node in the second layer is calculated by multiplying the weight parameters of every connected node with its corresponding activation value. This is illustrated in figure 3.4.

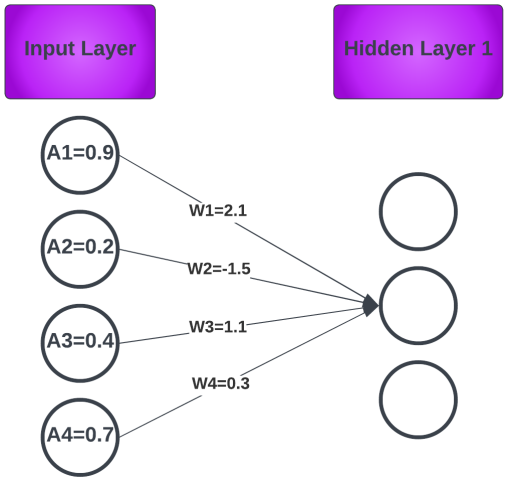

Figure 3.4: Calculating hidden layer's activation value

By calculating the weighted sum we get the value of the middle node of hidden layer 1 for our example:

$$w_1a_1 + w_2a_2 + w_3a_3 + w_4a_4 = 2.1 * 0.9 - 1.5 * 0.2 + 1.1 * 0.4 + 0.3 * 0.7 = 2.24 \tag{3.16}$$

The weight parameters, given to each connection, are based on relevant dependencies and patterns in the data. How these are calculated for the GAN will be discussed in section 3.2.7. What is most important here, is that each weight parameter needs to be calculated first in order to determine the activation value in the next layer. The activation values should always be within the range from 0 to 1. Therefore, a function is needed that takes as input the value of the weighted sum and outputs it as a value between 0 and 1. A commonly used function is the Sigmoid function (Narayan, 1997), a type of logistic curve. The sigmoid function is:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.17}$$

This transforms very negative inputs close to 0 and very positive inputs close to 1, and it steadily increases from inputs around 0. Another possibility that is more used today in modern neural networks is the ReLU function (Bai, 2022). This function is more computationally efficient and can help to overcome the vanishing gradient problem. The ReLU function is denoted as:

$$ReLU(a) = max(0, a) \tag{3.18}$$

However, using the ReLU function implies that the values of the activation in all neurons are within $[0, \infty]$ instead of $[0, 1]$, thus changing the interpretability of the network as explained here. Possibly, both ReLU and Sigmoid can be combined such that all activation values in the hidden layers are calculated using the ReLU function and the Sigmoid function is used (in the output layer) to output values between 0 and 1. For this example, to maintain consistency we continue by using the Sigmoid function. Using the weighted sum and the Sigmoid function we get:

$$\sigma(w_1 a_1 + w_2 a_2 + w_3 a_3 + w_4 a_4) \tag{3.19}$$

Therefore, the activation of the neuron in the next layer is just a measure of how positive the weighted sum going into this neuron is. If this neuron should only get values significantly above 0 if its weighted sum is above a certain threshold, a bias can be added to this function. Calculating this activation value must be done for every neuron in every layer. To make it more clear, we denote the entire architecture with matrix notations:

$$\sigma\left(\begin{bmatrix} w_{0,0} & w_{0,1} & \dots & w_{0,n} \\ w_{1,0} & w_{1,1} & \dots & w_{1,n} \\ \dots & \dots & \dots & \dots \\ w_{k,0} & w_{k,1} & \dots & w_{k,n} \end{bmatrix} \begin{bmatrix} a_0^0 \\ a_1^0 \\ \dots \\ a_n^0 \end{bmatrix} + \begin{bmatrix} b_0 \\ b_1 \\ \dots \\ b_n \end{bmatrix}\right) = \begin{bmatrix} a_0^1 \\ a_1^1 \\ \dots \\ a_n^1 \end{bmatrix} \tag{3.20}$$

This can compactly be written as:

$$\boldsymbol{a}^{(1)} = \sigma(\boldsymbol{w}\boldsymbol{a}^{(0)} + \boldsymbol{b}) \tag{3.21}$$

The entire network can be seen as a big function, taking in some data and returning an output vector, by using each individual neuron as a small intermediate function. The term learning quite simply refers to finding the right weights and biases to solve the problem at-hand that a researcher wants to solve using a chosen neural network. In the next sections, we explain how GANs work and how they incorporate neural networks.

3.2.2 How GANs work

Neural networks can be trained using a supervised learning approach (Winston, 1992). This approach can be generalized as follows. It starts with some input data, often called training data. This data is used to train a neural network, which generates an output, resulting in a prediction. The predicted output is then compared with the expected output from the training set (or a test set) during the analysis. Based on these results, the model can be updated, as shown in figure 3.5.

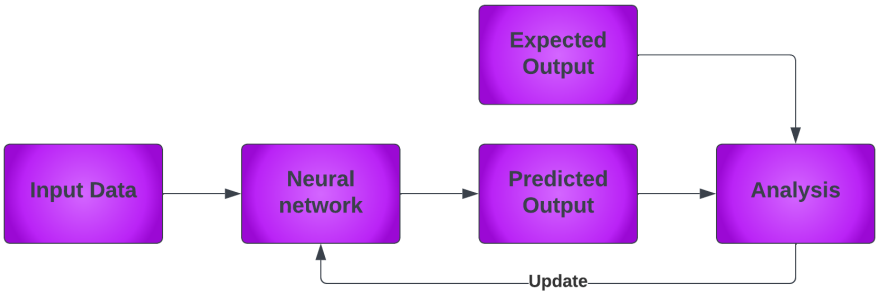

Figure 3.5: Supervised learning training process

In contrast to the example above, the GAN framework, developed by Goodfellow et al. (2014), is an unsupervised learning model. It consists of 2 sub-models: a generative model (Generator), used to capture the data's distribution and generate synthetic data, and a discriminative model (Discriminator), used to calculate the probability that some proposed data comes from training data rather than the Generator. The Generator's goal is to maximize the probability of the Discriminator making a mistake by creating data that closely resembles real data. This suggests that the GAN can accurately mimic the statistical properties of the test data.

The Discriminator can be pre-trained on real financial data trying to capture all of its important statistical properties, patterns, features, and structures, present in this dataset. The

Discriminator is given a batch of both real and fake data, to test whether it can discriminate the fake data from real. The fake data is generated by the Generator. The Generator is provided with random data, a form of noise, from which it constructs fake financial data for the Discriminator to determine whether it appears real or fake. Both the Generator and Discriminator are then "informed" whether the Discriminator's determination was true or not. Based on this, the loser will have to change its behavior by updating its weight parameters (and potentially biases). This learning process is often called a zero-sum game. This process must continue until the Generator can create such good synthetic financial data, that the Discriminator fails to recognize what is real and what is fake. See figure 3.6 for an overview of this process.

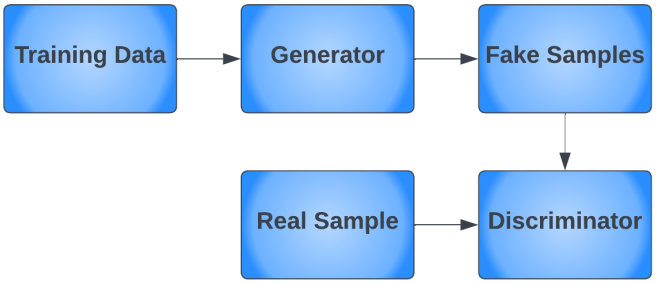

Figure 3.6: GAN training process

3.2.3 Minimax loss function

To measure the distance between the distributions of real and synthetic data, a loss function is used (Christoffersen et al., 2004). The original GAN from Goodfellow et al. (2014) used a minimax loss function. The Generator aims to minimize the loss given by the following equation:

$$min\ log(1 - D(G(z))) \tag{3.22}$$

Here, $G(z)$ represents the output of the Generator when given noise z. The Generator can only influence its own output $G(z)$ and not the Discriminator's probability estimate $D(x)$ that the Generator's data is classified as real data x. $D(G(Z))$ denotes the estimated probability of the Discriminator that fake data is real. The minimax loss function of the GAN becomes:

$$\min_G \max_D \ V(D,G) \tag{3.23}$$

$$where\ \ V(D,G) = E_x[log(D(x))] + E_z[log(1 - D(G(z))]$$

Here, E_z represents the expected value based on all random inputs given to the Generator. If we denote D and G as some probability measure taking random noise, then $V(D,G)$ is a metric between the two measurable functions. In simple terms, this just implies that the Discriminator wants to maximize the function and the Generator wants to minimize this function. The minimax loss function quantifies the discrepancy between the Discriminator's predictions and the true values by utilizing binary cross-entropy. This loss, commonly used in binary classification problems, tries to improve the Discriminator's ability to distinguish between two classes, namely real and fake data. In training, the Discriminator is iteratively improved to maximize the minimax loss, effectively maximizing the binary cross-entropy between itself and the Generator. The Generator will be improved to minimize the same function. If this can be achieved at a similar rate, it will lead to a Generator that can create synthetic data indistinguishable from real data.

3.2.4 Training difficulties

Training a GAN properly is often quite difficult (Kodali et al., 2017). During GAN training, two models are trained simultaneously, and an adjustment of one model's parameters affects the loss function. The GAN might never converge and consistently 'jump around' because an equilibrium between the Generator's and Discriminator's parameters can not be found. This indicates that there are issues with the learning stability of the GAN. This can be understood by an example. If the Generator's ability to create synthetic data improves much faster than the Discriminator's ability to distinguish it from fake data, the Generator can only be trained on poor feedback meaning that the Generator's parameters might never converge to the optimal values. This happens because the Discriminator always loses the zero-sum game, thus not getting and giving helpful feedback, resulting in parameters 'jumping around' within a small interval. Additionally, if the Generator always wins the zero-sum game, it is never forced to change its parameters further. This results in undiversified low output quality from the GAN.

A well-known problem when training GANs is mode collapse (Kodali et al., 2017). This happens when the Generator is stuck in a local minimum, while the global minimum must be found. This leads to the issue that the Generator can only create a limited range of similar samples, capturing only a portion of the data's distribution. The Discriminator can easily distinguish between real and fake data, causing the GAN to stop developing, resulting in an undiversified output. In reality, the Generator should provide both realistic and a wide variety of diversified data.

Another issue is the vanishing gradient problem (Hochreiter, 1998). During GAN training, this can happen when the Discriminator improves so much faster than the Generator, such that the Generator cannot properly be trained further. In that case, the Generator is not given enough useful data by the Discriminator to improve itself. The gradient becomes too small, such that the parameters of the Generator do not or barely change.

3.2.5 Wasserstein loss function and gradient penalty

To reduce the aforementioned training difficulties, the Wasserstein loss function can be used (Chintala et al., 2017) as an alternative to the minimax loss function. This measurement uses the Wasserstein distance, also often called the Earth mover's distance, to measure how much work is required to transform a given distribution into the true distribution. The Generator tries to minimize and the Discriminator to maximize the following Wasserstein loss function:

$$\min_{\mu_G} \max_{D} \; L_{WGAN}(\mu_G, D) \tag{3.24}$$

$$where \;\; V(D,G) = E_{\mu_G}[D(x)] - E_{\mu_{ref}}[D(x)]$$

Here $E_{\mu_G}[D(x)]$ is the expected value of the Discriminator's output when it evaluates data samples generated by the Generator. $E_{\mu_{ref}}[D(x)]$ is the expected value of the Discriminator's output when it evaluates real data samples from the real-world (reference) distribution. The Wasserstein loss function has been demonstrated to improve the learning stability and reduce the issues regarding mode collapse and vanishing gradient (Chintala et al., 2017).

In a regular GAN, the Discriminator often improves faster than the Generator, outputting more extreme values and thus providing less informative feedback to the Generator, hindering the learning process. The W-GAN, incorporating the Wasserstein distance, addresses this issue by substituting the Discriminator with a Critic. The Critic ensures that the sigmoid function is replaced by a linear activation function, eliminating the output constraint between 0 and 1. This means that the cost function continues to grow, irrespective of the divergence between distributions. This approach prevents the gradient from approaching zero, mitigating the vanishing gradient problem and minimizing mode collapse.

However, a prerequisite for using the Wasserstein loss is that the Critic must be 1-Lipschitz continuous. This means that the norm of the gradient should not exceed the value 1 at any point. This ensures the continuity (and differentiability) of the Wasserstein loss function. If this is handled correctly, then during training its growth will be stable, thereby validating the use of the underlying Wasserstein or Earth mover's distance. For W-GAN this is achieved using weight clipping, which forces the weights of the Critic to be within some interval. If any of the calculated weights of the neural network are outside this interval, the weights are simply clipped, setting values too high or too low to a predefined maximum/minimum. However, the downside is that weight clipping may limit the Critic's learning ability and, consequently, impact the overall performance of the GAN. As a result, extensive hyperparameter tuning becomes necessary to adapt the model for effective training, which is already a very complex task for GANs in the first place.

A solution to this problem was provided with the development of a W-GAN-GP (Gulrajani et al., 2017), where GP stands for gradient penalty. The gradient penalty is an alternative method, to weight clipping, to ensure 1-Lipschitz continuity. The W-GAN-GP provides a smoother way of enforcing continuity by incorporating a gradient penalty into the Wasserstein loss function. The gradient penalty involves penalizing the model based on the gradient norm of the Discriminator's output with respect to interpolated samples.

3.2.6 Optimizer

The primary goal of a GAN is to optimize a specific loss function, like the minimax or Wasserstein loss function. To find the local minimum of these functions, an optimizer is used. The gradient descent-based optimizer (Dogo et al., 2018) is a commonly used optimizer. It is a numerical method trying to find the function's minimum using small iterations. An example of a popular gradient descent-based optimizer is the Adaptive moment estimation (Adam) (Singarimbun et al., 2019).

During every training epoch, the parameters in the neural networks must be changed in such a way that the loss function is minimized. The parameters of the neural network that lost the game are updated, based on the impact they have on the loss function which is explained in section 3.2.7. The gradient descent-based optimizer is used to find the local minimum of the loss function with these updated parameters.

The parameters of the neural networks are adjusted by calculating the impact they have on the loss function. The Generator cannot directly influence the loss function, it's the output of the Discriminator that influences the Generator's loss impact (Eckerli et al., 2021). Since the Generator cannot directly influence the loss function, the impact of the Generator's parameters must be measured based on the impact they have through the Discriminator's parameters. Thus, backpropagation is necessary, starting from the output and going back through the Discriminator (taking into account its parameters) to the Generator. If both the Discriminator and Generator can be improved at the same rate, the Generator would eventually be able to create synthetic data that is indistinguishable from real data.

3.2.7 Update parameters

We now explain how the parameters of the Generator and Discriminator are adjusted. As mentioned in section 3.2.2, the Generator tries to minimize and the Discriminator tries to maximize the following minimax loss function:

$$\min_G \max_D \; E_x[log(D(x))] + E_z[log(1 - D(G(z))] \tag{3.25}$$

The parameters of the Generator and Discriminator are adjusted using many training epochs (Arjovsky et al., 2017). A single training epoch goes as follows:

1. Randomly sample a batch of m true values: $x_1, x_2, ..., x_m$ and calculate $D(x_i)$
2. Randomly sample a batch of m random noise samples: $z_1, z_2, ..., z_m$ and calculate $G(z_i)$
3. Use the output of the Generator $G(z_i)$ as input for the Discriminator to calculate $D(G(z_i))$
4. Calculate the Discriminator's loss: $L_D = \frac{1}{m} \sum_{i=1}^{m} [log(D(x_i)) + log(1 - D(G(z_i)))]$
5. Update the Discriminator parameters using the function L_D and: $\theta_d^{new} = \theta_d^{old} + \lambda \nabla_{\theta_d^{old}} L_D$
6. Calculate the Generator's loss: $L_G = \frac{1}{m} \sum_{i=1}^{m} [log(D(G(z_i)))]$
7. Update the Generator's parameters using the function L_G and: $\theta_g^{new} = \theta_g^{old} + \lambda \nabla_{\theta_g^{old}} L_G$

Here θ_d and θ_g are the parameters to optimize in the Discriminator and the Generator respectively. $\nabla_{\theta_d^{old}}$ is the gradient of the function with respect to the Discriminator's parameters. The gradients are computed using backpropagation, going from the error of the loss function through the network's layers using the chain rule. By ascending the gradient, we can determine how to adjust each parameter efficiently to maximize the loss function, which is the objective of the Discriminator. The Generator's parameters are updated by descending the gradient $\nabla_{\theta_g^{old}}$ to minimize the loss function.

Goodfellow et al. (2014) proved an important property of this approach. The minimum of the loss function can only be achieved if the probability function of the Generator matches the probability function of the real data, represented by the Discriminator. This proof, provided in appendix A, is fundamental to GAN. This property is important for our option pricing framework, as it allows us to capture essential statistical properties of financial time series. If this can be achieved during the training of the GAN, it implies that synthetic financial data cannot be distinguished from its real data counterpart any better than a random guess.

3.2.8 GAN Architectures

In Chapter 2, various GANs and their applications have been discussed. In this subsection, we discuss which architectures and loss functions will be implemented. In chapter 4 these different GANs are compared based on some statistical measures often exhibited in financial time series data, as discussed in section 2.4. The best performing GAN will be used in the GAN-QMC to price options.

The first GAN we build is a LSTM-GAN using a LSTM neural network for both the Generator and Discriminator. This choice is made because LSTM by itself has shown to be effective in predicting financial time series data, outperforming other models like random forests and deep neural networks in terms of predictive accuracy (Fischer et al., 2018). Furthermore, as shown before, integrating LSTM within a GAN has given promising results for generating financial data (Z. Zhang et al., 2020). The strength of LSTM lies in its ability to maintain information in memory over extended sequences. This is especially valuable for an analysis using financial data, where capturing long-term dependencies and understanding characteristics like volatility clustering and heavy tails is crucial. Traditional GANs typically use Convolutional Neural Networks (CNNs). These are better suited for grid-like data such as images and spatial data. Given that financial time series data is sequential, the use of a recurrent network like LSTM is a rational choice. The LSTM-GAN will use the standard minimax loss function using binary-cross entropy.

While LSTM offers several advantages over other neural networks for financial data modeling, it comes with potential challenges. Research has shown that recurrent Discriminators, including those utilizing LSTM, are particularly difficult to train (Metz et al., 2017). To address this issue, we will also develop a LSTM-TCN-GAN. This uses a Temporal Convolutional neural network (TCN) for the Discriminator and a LSTM for the Generator. A LSTM network seems

more suited to model financial time series. However, if the GAN cannot be trained properly using the LSTM-GAN's architecture, using a network suffering less from training instability is a reasonable choice. The LSTM-TCN-GAN uses the standard minimax loss function.

To improve the issues regarding mode collapse and the vanishing gradient problem we will also develop a Wasserstein GAN using the gradient penalty. Our W-GAN-GP will use the same architecture as the LSTM-TCN-GAN, but uses the Wasserstein loss function instead of the minimax loss function. In chapter 4 all these GANs are compared on their ability to create realistic financial time series data.

3.3 Combining GAN with MC/QMC

In this section, we discuss how the GAN can be combined in the MC simulation to price options. It is necessary to make some training data choices first. Then we show how the GAN can be used in the geometric Brownian motion to construct risk-neutral price paths. Finally, we give a brief overview of the GAN-QMC method using pseudo code and recap the methodology section.

3.3.1 Prices vs Returns

An important decision is to train the GAN on returns instead of prices because stock prices are non-stationary. Strict non-stationarity implies that the entire probability distribution of the time series is not constant over time (Brooks, 2014).

If y is the price of a financial asset, the time series would be strict stationary if its probability distribution is the same over time:

$$Fy_{t_1}, y_{t_2}, ..., y_{t_T}(y_1, ..., y_T) = Fy_{t_{1+k}}, y_{t_{2+k}}, ..., y_{t_{T+k}}(y_1, ..., y_T) \; \forall k \in \mathbb{Z} \tag{3.26}$$

This implies that the expectation and variance are constant over time, and the covariance only depends on the number of lags. Financial price series tend not to be stationary (Kendall et al., 1953). The non-stationarity can lead to difficulties when modeling and making predictions. Therefore, most studies about financial GANs use the (log) returns of the financial assets. Log returns are $X_t = ln\left(\frac{S_t}{S_{t-1}}\right)$. The returns time series of financial assets is stationary and this can significantly help statistical modeling. Therefore we use the GAN to generate log returns.

3.3.2 Training data

We now show how the training data set is constructed. A stock price vector is denoted as:

$$\boldsymbol{S}_{t,T} = (S_t, S_{t+1}, ..., S_{t+T}) \tag{3.27}$$

where $\boldsymbol{S}_{t,T}$ is a multivariate random variable with length T. A realization of a stock price vector is denoted as:

$$\boldsymbol{s}_{t,T} = (s_t, s_{t+1}, ..., s_{t+T}) \tag{3.28}$$

Here, the capital letter $\boldsymbol{S}_{t,T}$ represents the stochastic (multivariate random variable) price path vector and the small letter $\boldsymbol{s}_{t,T}$ represents the deterministic realized price path vector (also for the real historical price paths).

We gather some historical data of a stock to train the GAN, denoted as a realization $\boldsymbol{s}_{1,n}$ where n is the number of time steps in that set. n must be large enough to make sure that the distribution of the time series $\boldsymbol{s}_{t,n}$ does not depend on the initial value t. Preferably, the distributions of $\boldsymbol{s}_t$ and $\boldsymbol{s}_{t,n}$ are identical. This indicates that the generated data will have enough variation to include a large degree of volatility present in the market. To achieve stable training, it is important to split the training data into the right number of partitions. Denoting

the training set $\boldsymbol{TS}$ as a matrix of many partitions $\boldsymbol{s}_{t,T}$ and the price of the asset s at time t as s_t, we get:

$$\boldsymbol{TS}_{h,T} = \begin{Bmatrix} s_1 & s_{1+h} & ... & s_{1+\lfloor \frac{n-T}{h} \rfloor h} \\ s_2 & s_{2+h} & ... & ... \\ ... & ... & ... & ... \\ ... & ... & ... & ... \\ s_{1+T} & s_{1+T+h} & ... & ... \end{Bmatrix} = \left\{ \boldsymbol{s}_{1,T}, \boldsymbol{s}_{1+h,T}, ..., \boldsymbol{s}_{1+\lfloor \frac{n-T}{h} \rfloor h,T} \right\}$$

The symbol $\lfloor a \rfloor$ is a floor function and calculates the largest integer smaller or equal to a. The variable h is the step size. The step size indicates how many time steps t each batch is shifted. Considering that T is fixed, the step size also determines how many batches the training data will have. A value of h near zero results in many almost identical partitions, increasing the chance of mode collapse. For very large h there will be no more overlap between partitions. This improves the training of the Discriminator due to only having a few diversified batches, meaning that it will be hard for the Generator to deceive it. This could result in the Generator never getting valuable feedback and this could lead to the vanishing gradient problem. The step size is determined when tuning the GAN, which will be explained in section 4.3.2. It should be noted that the stock prices s in $\boldsymbol{TS}_{h,T}$ will be transformed into log returns using $X_t = ln\left(\frac{S_t}{S_{t-1}}\right)$ before training the GAN.

3.3.3 Real-world to Risk-neutral price paths

We now explain how the GAN is used to generate risk-neutral price paths, necessary to price the options in a simulation. The GAN is trained on real-world log returns of an asset. Using the output of the GAN directly to construct price paths will not result in risk-neutral price paths. However, to price options under risk neutrality, we need to use the GAN to construct risk-neutral price paths. This is achieved by using the geometric Brownian motion and replacing the standard normal distribution with standardized output sampled from the GAN.

The standard geometric Brownian motion used in option pricing is defined by the following stochastic differential equation (Hull, 2000):

$$dS_t = \mu S_t dt + \sigma S_t dW_t \tag{3.29}$$

In this equation, W_t is a Wiener process. The most important properties of a Wiener process are as follows (Shreve, 2004):

1. W_t has independent increments such that $W_{t+u} - W_t$, with $u \geq 0$, are independent of past values of W_s with $s < t$.
2. The increments $W_{t+u} - W_t$ are Gaussian, so normally distributed with a mean of 0 and a variance of u.
3. W_t must be continuous, therefore t must be continuous.

This implies that $W_{t_1} - W_{s_1}$ and $W_{t_2} - W_{s_2}$ are independent Gaussian random variables where $0 \leq s_1 < t_1 \leq s_2 < t_2$.

It can be seen that the geometric Brownian motion is defined as the exponential of a Wiener process with a drift and volatility component. In other words, a geometric Brownian motion is essentially a Wiener process modified by the exponential function. It is used to model the randomness or uncertainty in the financial market. For option pricing, it is used to model the movement of asset prices. However, returns of an asset tend not to follow a Gaussian distribution (Theodossiou, 2000), thus our interest in changing this property.

To solve the geometric Brownian motion, Ito's lemma is used (Shreve, 2004). Ito's lemma states that for a process X described by the stochastic differential equation (SDE) $dX_t = \mu dt + \sigma dW_t$, the change in the process $Y(t) = f(t, X(t))$ can be denoted as:

$$dY_t = \left(\frac{\partial f}{\partial t} + \mu\frac{\partial f}{\partial x_t} + \frac{1}{2}\sigma^2\frac{\partial^2 f}{\partial x_t^2}\right)dt + \sigma\frac{\partial f}{\partial x_t}dW_t \tag{3.30}$$

To solve the geometric Brownian motion for the stock price as shown in equation 3.29, set $Y(t) = ln(S(t))$, $X(t) = S(t)$, and $f(t,x) = ln(x)$. This implies that $\frac{\partial f}{\partial t} = 0$ and $\frac{\partial f}{\partial x} = \frac{1}{x}$. Ito's lemma states that the solution to the geometric Brownian motion is equation 3.30. The symbol W_t is the standard Wiener process where $W_T - W_t \sim \sqrt{T-t}Z$ where $Z \sim N(0,1)$. The random variable Z has an independent and identical normal distribution. Using equation 3.30, the solution to the geometric Brownian motion of our stock price process becomes (Shreve, 2004):

$$\frac{dY_t}{dt} = \frac{1}{S_t}\frac{dS_t}{dt} - \frac{1}{2S_t^2}(dS_t)^2 \tag{3.31}$$

$$\frac{dY_t}{dt} = \frac{1}{S_t}(rS_t dt + \sigma S_t dW_t) - \frac{1}{2S_t^2}(\sigma S_t dW_t)^2 \tag{3.32}$$

$$dY_t = \left(r - \frac{\sigma^2}{2}\right)dt + \sigma dW_t \tag{3.33}$$

Integrating both sides from 0 up to t:

$$\int_0^t dY_t = \int_0^t \left(r - \frac{\sigma^2}{2}\right)dt + \int_0^t \sigma dW_t \tag{3.34}$$

$$ln\left(\frac{S_t}{S_{t-1}}\right) = \left(r - \frac{\sigma^2}{2}\right)t + \sigma W_t \tag{3.35}$$

$$\ln\left(\frac{S_t}{S_{t-1}}\right) = \left(r - \frac{\sigma^2}{2}\right)t + \sigma W_t \tag{3.36}$$

$$\frac{S_t}{S_{t-1}} = e^{\left(r-\frac{\sigma^2}{2}\right)t+\sigma W_t} \tag{3.37}$$

$$S_t = S_{t-1}e^{\left(r-\frac{\sigma^2}{2}\right)t+\sigma W_t} \tag{3.38}$$

This equation can be used to model the continuous-time evolution of stock prices assuming the stock price follows a geometric Brownian motion. When pricing options using a MC simulation, the stock price must be simulated, implying that the stock price must be modeled in discrete time. We discretize time into intervals of size Δt and approximate the increment $W_{t+\Delta t} - W_t$. The Wiener process has independent and stationary normally distributed increments. This allows for the approximation of the increment $W_{t+\Delta t} - W_t$ using a standard normal variable $Z \sim N(0,1)$ resulting in $W_t \approx \sqrt{\Delta t}Z_t$. The symbol $\sqrt{\Delta t}$ can be seen as a scaling factor to scale the variance of Z_t to match the variance of the Wiener process increments ensuring that the variance of $\sqrt{\Delta t}Z_t$ is Δt because $Var(\sqrt{\Delta t}Z_t) = (\sqrt{\Delta t})^2 Var(Z_t) = \Delta t * 1 = \Delta t$. These convenient properties allow us to write equation 3.38 as the following discrete variant:

$$S_t = S_{t-1}e^{\left(r-\frac{\sigma^2}{2}\right)\Delta t+\sigma\sqrt{\Delta t}Z_t} \tag{3.39}$$

However, research has shown that real-world returns are not always normally distributed (Theodossiou, 2000). The idea to generate risk-neutral returns by using a GAN is to change the

normality (Gaussian) assumption. For the GAN-QMC method, we swap the standard normal random variable Z_t out by treating the output of the Generator of a GAN as an empirical distribution of the stock returns. This distribution must be independent and have a mean of zero with constant variance equal to the time step. We standardize the output of the GAN, before using it in the geometric Brownian motion to construct risk-neutral price paths. Our idea is that if the GAN can be trained perfectly, and exactly mimics all statistical properties of real-world stock returns, it is a better approximation of the returns than the normality assumption. This should lead to more realistic risk-neutral price paths and thus potentially result in more accurate option prices. However, since the normality assumption is not far off, the GAN might need to be near perfect in order to measure any statistically significant accuracy improvement. Equation 3.39 can be rewritten, replacing the standard normal distribution Z_t by treating the standardized output of the Generator G_t as the empirical distribution of the stock returns, as follows:

$$S_t = S_{t-1} e^{\left(r - \frac{\sigma^2}{2}\right)\Delta t + \sigma\sqrt{\Delta t} G_t} \tag{3.40}$$

By using the risk-free rate for r and the implied volatility of the option of interest for σ, we can generate risk-neutral price paths for the GAN-QMC by sampling from the Generator of the GAN instead of sampling from the standard normal distribution as done in a regular MC/QMC simulation for option pricing.

The output of the Generator is being treated as an empirical distribution. This distribution is not absolutely continuous because it doesn't have a probability density function that is integrable. Since the Generator is not absolutely continuous it cannot be a stable random variable (Lukacs et al., 1970). This means that the following equation does not necessarily hold:

$$G_1 + G_2 + ... + G_n \overset{d}{\neq} c_n G + d_n \tag{3.41}$$

The consequence of this is that the length of the discretization interval Δt, the time step between 2 consecutive samples from G, might impact the validity of the risk-neutral paths generated by equation 3.40. In section 4.4 we analyse, for different time steps Δt, the difference in mean between price paths generated using equation 3.40 compared to the regular method using equation 3.39. This sanity check allows us to analyse whether the choice of Δt impacts the ability of the GAN in combination with the geometric Brownian motion to create risk-neutral price paths.

To recap, to use a GAN to construct risk-neutral price paths, the standard normal distribution in the geometric Brownian motion will be replaced by outputs sampled from the Generator of the GAN. The process of generating risk-neutral price paths goes as follows:

1. Gather a dataset of some listed stock: $\boldsymbol{s}_{t,T} = s_t, s_{t+1}, ..., s_{t+T}$
2. Calculate log returns: $x_t = ln\left(\frac{s_{t+1}}{s_t}\right)$
3. Train GAN on $\boldsymbol{x}$
4. Generate (a vector of) returns using the Generator called $\boldsymbol{g}$
5. Standardize the returns: $\boldsymbol{g}_{sd} = \frac{\boldsymbol{g}_{sd} - \mu}{\sigma}$, where $\mu = E(\boldsymbol{g})$ and $\sigma_{GAN} = \sqrt{Var(\boldsymbol{g})}$
6. Construct price paths using formula 3.40 where we sample returns $\boldsymbol{g}$ from G_t

For the QMC simulation, the synthetic returns $\boldsymbol{g}$ are not randomly sampled from the Generator. Instead, millions of random numbers are sampled from the Generator first. Then the Halton sequence is used to sample from this large dataset. This is done because it is impossible to sample from the inverse cumulative distribution of a GAN without using a technique

like kernel density estimation to estimate its probability density function. The quasi-random numbers are used in equation 3.40 to generate price paths, similar to the procedure above.

3.3.4 Pseudo code GAN-QMC

Algorithm 1 shows a brief pseudo-code of the GAN-QMC method to price a European or American option. In appendix A a pseudo-code of option pricing using MC, QMC, and GAN-QMC is shown.

Algorithm 1 Pseudo code GAN-QMC

Input: $N, \boldsymbol{s}_{1,n}, T_0, T, K, r, C, P, M, h$
Split historical data $\boldsymbol{s}_{1,n}$ into $\boldsymbol{TS}_{h,T}$
for $i = 1, 2, \ldots, M$ **do**
    Train Generator on random noise $\boldsymbol{Z}$
    Train Discriminator on $\boldsymbol{TS}_{h,T}$ and $G(\boldsymbol{Z})$
    Calculate losses L_G & L_D and update parameters θ_g^{new} & θ_d^{new}
for $i = 1, 2, \ldots, N$ **do**
    Sample from the Generator: $\epsilon = \{G(\boldsymbol{Z})$ or $G(\boldsymbol{HQ})\}$
    Use the GBM to construct a price path: $\hat{\boldsymbol{s}}_{i,T}^{i} = (\hat{s}_{n+1}, \hat{s}_{n+2}, \ldots, \hat{s}_{n+T})$
Calculate average payoff of European/American option based on $\hat{\boldsymbol{s}}_T$
Calculate the option estimate $\hat{C}$ or $\hat{P}$ by discounting the payoff with r
Output: $\hat{C}$ or $\hat{P}$

3.3.5 Recap of Methodology

The goal of this thesis is to price options using GAN-QMC and analyse whether it is more accurate and computationally efficient than pricing through MC/QMC. The MC simulation generates many price paths by randomly sampling from the standard normal distribution and using these samples in the geometric Brownian motion. QMC differentiates by quasi-random sampling from the standard normal distribution using the Halton sequence. The generated price paths are used to estimate the European and American option prices based on their respective payoffs.

The GAN is an AI consisting of 2 neural networks, trained together in a zero-sum game. Important choices that influence the GAN's ability to generate financial data are the neural network's architecture, the loss function, and the type of training data. The GAN is optimized by minimizing the loss function. In section 4.4, the GAN's ability to generate financial data is determined by analysing whether its output has statistical properties similar to the real-world data.

To price options using the GAN-QMC, we sample either randomly or quasi-randomly (using the Halton sequence) from the Generator. These samples are used in the geometric Brownian motion to construct risk-neutral price paths and estimate the option prices, similar to the regular MC/QMC. The expected improvement in accuracy is due to the GAN's ability to better mimic the true statistical properties of the option's underlying asset compared to the standard normal distribution used in the geometric Brownian motion. We used the functional form of the geometric Brownian motion and modeled the price returns using the GAN in order to generate risk-neutral price paths. In the discussion, we delve into the assumptions of GAN-QMC and explore suggestions to relax some of these assumptions.

Chapter 4

Model training and validation

In the previous chapter, we explained the methods and concepts necessary to price options through MC, QMC, and GAN-QMC. This chapter is dedicated to the training and validation of these models. Section 4.1 discusses data decisions and lays the theoretical foundation of the performance metrics necessary to quantify the GAN-QMC's ability to estimate option prices. Section 4.2 briefly elaborates on the implementation and validation of the MC/QMC simulations. Section 4.3 discusses the initial GAN hyper-parameter selection and theory behind the GAN tuning process, all based on existing literature. Finally, theory is put into practice in section 4, fully dedicated to the execution of the tuning process and analysis of all tuned GANs. In chapter 5 the best-performing GAN is used in the GAN-QMC and analysed using the chosen performance measures.

4.1 Performance metrics

To determine the performance of the different pricing methods, data of a stock exchange-listed company with options must be available to train the GAN and price options. Subsequently, statistical measures and graphs will be used to compare the accuracy and computational efficiency of these pricing methods.

4.1.1 Data

As explained in chapter 1, our focus is on options with a stock as the underlying asset. In this chapter, the specific focus is on the company BHP Group Limited (BHP, 2022), an Australian mining and metals company. This was chosen due to its large market cap and liquid European and American options with a decent amount of data easily available.

2 datasets are utilized, namely a dataset containing stock prices for GAN training and a dataset containing option prices (and their parameters) used for pricing purposes, hereafter referred to as stock-dataset and option-dataset, respectively. These datasets are used to demonstrate the development of the different methods, while Chapter 5 expands the analysis to incorporate data from multiple companies, using the same GAN-QMC setup and tuning process. The option-datasets of all companies can be found on the following GitHub page: https://github.com/sP22222/GANs-Q-MC. All datasets and their descriptive statistics will be explained in more detail in chapter 5.

To tune the GAN, the stock-dataset is split up into a training and test set. The training data consists of the daily closing prices of the BHP stock from 1980 up to 2010 and the test set of the closing prices from 2010 up to mid-2023. To price options in the simulation, the option-dataset of BHP is used, extracted from the Australian stock exchange (Exchange, 2023).

It consists of 250 options, both American and European with a variety of different parameter ranges (e.g. strike price, time to maturity, etc.).

4.1.2 Statistical measures

The statistical measure used to determine the accuracy of the different pricing methods is the mean absolute percentage error (MAPE) (Myttenaere et al., 2016):

$$MAPE = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{Prediction_i - Observed_i}{Observed_i}\right| \tag{4.1}$$

Here, N denotes the number of entries in the option-dataset. The MAPE indicates, on average, how much the simulated value deviates from the true value. Due to the inherent randomness in MC/QMC simulation, the error for each estimated option price carries some level of uncertainty because $Prediction_i$ is not deterministic.

To assess the statistical significance between different pricing methods, the MAPE is expressed as an asymptotic 95% confidence interval (CI):

$$95\% - CI = \left[MAPE_{avg} - 1.96\frac{SD(MAPE)}{\sqrt{N}}, MAPE_{avg} + 1.96\frac{SD(MAPE)}{\sqrt{N}}\right] \tag{4.2}$$

Here, $MAPE_{avg}$ is the average MAPE over all options, and $SD(MAPE)$ is the standard deviation of the MAPEs. The CI is an approximation, becoming more accurate with larger sample sizes.

It is important to note that the MAPE itself is uncertain due to the randomness in the MC simulation making the CI provide an approximate range. If the GAN-QMC's CI is higher than QMC's CI with no overlap, there is approximately a 95% confidence that the accuracy of GAN-QMC is higher than QMC. CIs will be calculated separately for puts, calls, European, and American options.

To compare computational times, we simply take the average time it takes to generate the price paths and price the option itself. The final measure is the average time it took to price 1 option.

4.1.3 Graphs

We will extend beyond the conventional accuracy metrics considering that the overall average MAPE across all option prices may not always reveal all statistically significant differences. Instead, we must accept that a pricing method could show superior performance within specific intervals of option parameters and show poor results for others. Our objective is to investigate the performance of each method concerning different sets of option-parameter values based on the book of Hull (2000), analysing where each method excels or falls short.

Moneyness

The moneyness of an option is the profit gained when exercising the option. For a European call option, this is $S_T - K$, and for a European put $K - S_T$. A negative value implies that the option is out-of-the-money (OTM), a value around 0 is called at-the-money (ATM), and a positive value is called in-the-money (ITM). By plotting moneyness against pricing errors, insights into a method's proficiency in pricing OTM, ATM, and ITM options are gained.

Maturity

The maturity, the time it takes until the option expires, has an impact on the price of an option. The option price decreases as it reaches maturity, which is called the time decay of an option often called theta and mathematically denoted as $\Theta = -\frac{\partial V}{\partial t}$. By plotting the maturity against the pricing error, an estimate of the method's ability to capture theta can be made. The method that is most accurate for options with long maturities can be considered to be the best in capturing and predicting the impact of time decay.

Implied Volatility

Another graph that we will analyse is the implied volatility of the option against the error of the pricing method. This is often called vega and denotes the sensitivity of the option with regard to the implied volatility, written as $\nu = \frac{\partial V}{\partial \sigma}$. It is interesting for us to see whether a specific pricing method is better at pricing options with the implied volatility in a specific range.

Convergence

To have a secondary metric for computational efficiency, we will also analyse the convergence of the simulation. This is done by plotting the MAPE versus the number of trials in the simulation. Based on this graph it can be determined how many trials are necessary before an option price converges. It is expected that QMC converges faster than MC, but there is most likely not much difference between QMC and GAN-QMC.

4.2 MC & QMC simulation

Building on chapter 3, the MC and QMC simulations are developed. These simulations can later be extended by sampling from the GAN's Generator instead of the standard normal distribution as explained in chapter 3.

It is important to ensure that the simulations behave as expected. This was achieved with a simple sensitivity analysis of the simulations to price ATM European put and call options. Tables C.1 and C.2 in appendix C show the parameters that were varied with regards to their respective expected impact and observed impact on the option price. The option prices responded as expected to the changes in parameters indicating that our simulations can price options reliably. The results of the sensitivity analysis also extend to the GAN-QMC approach considering it uses the exact same simulation, only the distribution of the stock returns is different. A more elaborate sensitivity analysis for the GAN-QMC can be found in chapter 5. In addition, it is worth mentioning that based on some initial testing the QMC is significantly more accurate than MC when the number of trials is low (< 10.000). This discrepancy becomes less apparent once the number of trials increases significantly. These results correspond with the expected differences between MC and QMC.

4.3 Training of GAN

We build the LSTM-GAN, LSTM-TCN-GAN, and W-GAN-GP as explained in subsection 3.2.8. However, based on some initial training, we noticed that it is difficult to select hyper-parameters that can effectively train the GANs. Therefore, a tuning method is necessary to determine a good set of hyper-parameters to train the GANs. To determine whether the GANs can create realistic synthetic financial data, their output will be compared with the real-world data, based on a selection of statistical properties often exhibited by financial time series as discussed in section 2.4.

4.3.1 Hyper-parameters

We now discuss which initial hyper-parameters are chosen as a starting point for tuning the GAN. Starting with the batch length, the length of a single individual training set, certain researchers used extremely large batch lengths of around 8000 (Takahasi et al., 2019). Some initial GAN training showed that very large batches, consisting of thousands of individual observations required a lot of memory. Thus training using many batches consisting of 10 years worth of data did not work well. So either the number of batches per training cycle or the length of each batch must be small. Vuletic et al. (2023) used batch lengths of 100, which is just under 5 trading months, whereas some researchers used even smaller batches (Soleymani et al., 2022). A batch length of 250 seems like a reasonable starting value, striking a balance between computational time and the degree of variation within each batch. The batch size, denoting the number of batches used per training epoch, showed considerable variation among researchers. However, a reoccurring pattern suggests that as the batch length increases, the batch size tends to decrease (Takahasi et al., 2019). Considering that we start with a moderate batch length, we decide to also start with a moderate number of batches, namely 24.

A high learning rate of 0.001 has been shown to result in quicker convergence (Ghosh et al., 2020). However, higher learning rates can also lead to training difficulties. Other papers showed success when lowering the learning rate to 0.0001 (Lala et al., 2018) (Vuletic et al., 2023). Therefore, we start with a learning rate of 0.0001 for both the Generator and Discriminator, but these can independently vary in the tuning process.

In the literature, the number of layers for the Generator and Discriminator for generating financial data seems to always be somewhat low, often somewhere between 1 and 5 (Soleymani et al., 2022) (Zhou et al., 2018). Therefore, we start with 1 layer each and allow it to increase in the tuning process. The number of hidden dimensions in the Generator and Discriminator refers to the number of neurons in a hidden layer. This is hard to determine because the choice is very much dependent on the training data. Some initial training showed that an arbitrary but decent starting point seems to be 100. The length of the Generator's input is the dimensionality of the random noise vector used for training, which is the number of nodes of the input layer of the Generator's LSTM. In practice, most GANs seem to use a value between 10 and 100 (Vuletic et al., 2023) (Mirza et al., 2014). A larger value can allow the GAN to capture more complex dependencies in the data but will also increase the computational time, so somewhat smaller values are favored (Dumont et al., 2022). Therefore we will start with a value of 100.

For the LSTM-TCN-GAN and W-GAN-GP, the dropout rate can also be varied, which is the probability of an iteration being dropped out. Large values result in slower convergence. This value might be seen as less important considering that few researchers mention their dropout rate, probably because this value is dependent on which activation function is used. The dropout rate reduces overfitting at the cost of computational times (Srivastava et al., 2014), therefore we deem a starting value of 0.2 as a good balance. Lastly, we could pre-train the Discriminator, to potentially improve the training stability. We start by pre-training the Discriminator with 5 epochs.

The choice for the number of epochs in the GAN tuning process is dependent on the computational power available. Based on some initial testing, we decided that a value of 50 seems reasonable for tuning the GAN. The GANs can later be trained on more epochs using only the optimal set of hyper-parameters (the output of the tuning process).

The LSTM-GAN uses the Sigmoid activation function (Narayan, 1997). As discussed in chapter 3, the ReLU activation function can overcome the vanishing gradient problem (Bai, 2022). Therefore, we decided to implement the ReLU function in the LSTM-TCN-GAN and W-GAN-GP to improve on the simpler LSTM-GAN.

In addition, we have given reasonable lower and upper bounds to all hyper-parameters and an adjustment factor Δ. All results are shown in table 4.1 and are necessary to tune the GANs, which will be explained in section 4.3.2.

Hyper-parameter	Lower Limit	Upper Limit	Δ	Initial Value
Batch length	50	1000	0.1	250
Batch size	8	64	0.1	24
Step size	10	batch length	0.1	50
Learning rate G	0.00001	0.0002	0.1	0.0001
Learning rate D	0.00001	0.0002	0.1	0.0001
Number of layers G	1	5	1.5	1
Number of layers D	1	5	1.5	1
Number of hidden dimensions G	50	300	0.1	100
Number of hidden dimensions D (LSTM)	50	300	0.1	100
Dropout rate D (TCN)	0	0.4	0.1	0.2
Length of input G	50	200	0.1	100
Number of pre-training epochs D	0	10	0.1	5

Table 4.1: Hyper-parameters

4.3.2 Tuning the hyper-parameters

Tuning the hyper-parameters of a GAN is a complex task, given that adjusting one parameter can impact the quality of another. Therefore, a sequential optimization approach, starting with the optimization of a single parameter, might lead to sub-optimal results.

Tuning the hyper-parameters of a GAN can be denoted as a bilevel optimization problem (Dumont et al., 2022):

$$\min_{\boldsymbol{p},\boldsymbol{w}} U(\boldsymbol{p}, \boldsymbol{w}^*; \mathcal{D}_{val}) \tag{4.3}$$

$$s.t.\ \boldsymbol{p} \in \Omega$$

$$\boldsymbol{w}^* \in \min_{\boldsymbol{w} \in W} l(\boldsymbol{w}; \boldsymbol{p}, \mathcal{D}_{train})$$

Here, the hyper-parameters are represented by the vector $\boldsymbol{p}$. The symbol Ω can be interpreted as the entire domain of possible values as defined in table 4.1. The symbol U represents some inverse performance measure, which must be minimized with respect to the hyper-parameters $\boldsymbol{p}$. Minimizing U should result in optimized weight parameters and biases $\boldsymbol{w}$ (explained in section 3.2.1) for the real-world data $\mathcal{D}_{val}$. It should be noted that the optimal weight parameters $\boldsymbol{w}^*$ of the neural network of the Generator are found only by changing the values of the hyper-parameters $\boldsymbol{p}$, and thus not by changing $\boldsymbol{w}$ directly.

No matter the selected hyper-parameter values, we must always obtain the optimal $\boldsymbol{w}^*$ conditional on any set $\boldsymbol{p} \in \Omega$, which is achieved by minimizing $l(\boldsymbol{w}; \boldsymbol{p}, \mathcal{D}_{train})$. For LSTM-GAN and LSTM-TCN-GAN the minimax function is used for l where the goal is to minimize the cross-entropy, to obtain $\boldsymbol{w}^*$. For W-GAN-GP, the Wasserstein loss function is used for l instead. Thus $\boldsymbol{w}^*$ must be interpreted as the optimal weights and biases given some hyper-parameters $\boldsymbol{p}$ achieved by minimizing l, and might thus not be the global optimal values given the optimal hyper-parameters $\boldsymbol{p}^*$.

To tune the GAN, an additional measurement U is necessary to determine the quality of the GAN given a set of hyper-parameters $\boldsymbol{p}$. This is the reason why tuning a GAN is considered a bilevel optimization problem, l is minimized to obtain $\boldsymbol{w}^*$ while trying to minimize U to obtain $\boldsymbol{p}^*$. It needs a second objective function that quantifies the GAN's ability to generate realistic data. Here, we also choose the Wasserstein distance for U. Although confusing, this means that the GAN is tuned, i.e. the optimal hyper-parameters p^* are found, by using the Wasserstein distance as an objective function. A second loss function, either minimax or Wasserstein, is

used to determine the optimal weights $\boldsymbol{w}^*$ given a set of hyper-parameters $\boldsymbol{p}$ as explained in section 3.2.3. To clearly differentiate, to tune the GAN we minimize the objective function U and then we find the optimal weights $\boldsymbol{w}^*$ given a set of hyper-parameters $\boldsymbol{p}$ by minimizing a loss function l. The loss function l is either the minimax or Wasserstein loss function.

The Wasserstein distance U can discretely be defined for this optimization problem as:

$$W(\mathcal{D}_{GAN}, \mathcal{D}_{val}) = \sum_i |F_{GAN}(x_i) - F_{val}(x_i)| \tag{4.4}$$

This is a metric to compare the probability distribution of the real-world data $\mathcal{D}_{val}$ and the GAN generated data $\mathcal{D}_{GAN}$ by calculating their distance. It is an alternative to JS divergence, normally used in GANs, see appendix A and section 3.2.7. The Wasserstein measure offers a smoother representation of the distance between 2 probability distributions, especially for lower dimensions and with little overlap between them.

It is important to note that any inverse performance metric that measures the GAN's ability to generate realistic data can be used for U. We could for example have made a combination of the statistical properties mentioned in section 2.4. However, creating a performance measure combining these properties would make the tuning process significantly more difficult and time-consuming, which is the reason for choosing the Wasserstein distance for U.

Preferably, the GAN is trained on the optimal hyper-parameters $\boldsymbol{p}^*$, a vector of parameters where $U(\boldsymbol{p}^*, \boldsymbol{w}^*; \mathcal{D}_{val}) \leq U(\boldsymbol{p}^{(j)}, \boldsymbol{w}^*; \mathcal{D}_{val})\ \forall \boldsymbol{p}^{(j)} \in \Omega$. Considering that it might be too time-consuming or even impossible to find $\boldsymbol{p}^*$, we want to deploy a simple and fast GAN hyper-parameter tuning strategy, which is currently lacking in the literature. We decided to employ an automated parameter tuning strategy (Sæternes et al., 2023) because it is simple to implement and can easily be scaled to the computational power available. The goal here is to find a set of hyper-parameters that improves on the initial choice $\boldsymbol{p}^{(0)}$ shown in table 4.1. This is achieved using perturbation theory, where an approximate solution is found by combining solutions of simpler intermediate problems. GAN hyper-parameter tuning is a novel topic and perturbation theory has never been applied to tune the hyper-parameters of a GAN. For each iteration j we randomly change the hyper-parameters within their boundaries using Δ to end up with a new improved set $\boldsymbol{p}^{(j)}$ satisfying $U(\boldsymbol{p}^{(j)}, \boldsymbol{w}^*; \mathcal{D}_{val}) \leq U(\boldsymbol{p}^{(j-1)}, \boldsymbol{w}^*; \mathcal{D}_{val})$, where j represents one iteration.

During each iteration j, the m new vectors of the hyper-parameters are created using:

$$\boldsymbol{p}^{(j)_k} = \boldsymbol{p}^{(j-1)} + \boldsymbol{b}_k \cdot \boldsymbol{u}_i \cdot \boldsymbol{p}^{(j-1)} \tag{4.5}$$

$$for\ k = 1, ..., m$$

For $\boldsymbol{u}_i$ we choose a normal probability distribution with a mean of 0 and standard deviation equal to Δ_i, the pre-defined change of each hyper-parameter $i = 1, ..., n$, as shown in table 4.1. The symbol $\boldsymbol{b}_k$ represents a probability distribution where $\mathbb{P}(\boldsymbol{b}_{ki} = 1) = 0.5$ and $\mathbb{P}(\boldsymbol{b}_{ki} = 0) = 0.5$. This (almost) ensures that not every variable is perturbed during each training cycle k. It is important to ensure that every parameter $\boldsymbol{p}^{(j)_k}$ remains within the pre-specified lower and upper bounds. Therefore, if the boundary is exceeded, we simply set the value to the pre-defined minimum/maximum. In addition, for integer-valued hyper-parameters, like the number of LSTM layers, we simply round off to the nearest integer.

Now we train the GAN for every new parameter set $\boldsymbol{p}^{(j)_k}$ to obtain m inverse performance measures $U(\boldsymbol{p}^{(j)_k}, \boldsymbol{w}^*; \mathcal{D}_{val})$. Using these results, we can construct an additional parameter $\boldsymbol{p}^{(j)_{m+1}}$ using the information from all parameter sets $\boldsymbol{p}^{(j)_k}$. This is achieved by calculating the following vector:

$$\boldsymbol{s} = \frac{\sum_{k=1}^{m} [U_k^j < \infty](U^{j-1} - U_k^j)(\boldsymbol{p}^{(j)_k} - \boldsymbol{p}^{(j-1)})}{\sum_{k=1}^{m} [U_k^j < \infty]} \tag{4.6}$$

Here [.] is 1 if its condition is true and 0 if false. The vector $\boldsymbol{s}$ has an individual entry for each parameter i denoted as $\boldsymbol{s} = (s_1, ..., s_n)$. Then we can calculate the additional set of hyper-parameters:

$$p_i^{(j)_{m+1}} = p_i^{(j-1)} + \frac{s_i}{p_i^{(j-1)} U^{j-1}} \tag{4.7}$$

Here $p_i^{(j)_{m+1}}$ for $i = 1, ..., n$ are all individual hyper-parameters in the set $\boldsymbol{p}^{(j)_{m+1}}$. If a hyper-parameter of the previous set is 0, e.g. $p_i^{(j-1)} = 0$, we use:

$$p_i^{(j)_{m+1}} = \frac{s_i}{U^{j-1}} \tag{4.8}$$

Now we run the code again to obtain $U(\boldsymbol{p}^{(j)_{m+1}}, \boldsymbol{w}^*; \mathcal{D}_{val})$. For the next iteration $j+1$, we choose the set $\boldsymbol{p}^{(j)}$ from iteration j such that it satisfies $U(\boldsymbol{p}^{(j)}, \boldsymbol{w}^*; \mathcal{D}_{val}) = min(U^{j-1}, U_1^j, ..., U_m^j, U_{m+1}^j)$. This implies that we continue the tuning process with the set that resulted in the GAN with the lowest Wasserstein distance. Due to choosing only a limited number of iterations j, there is no guarantee to find the optimal set of hyper-parameters $\boldsymbol{p}^*$. However, using this approach, we will find hyper-parameters that are at least as good as our initial values that were chosen based on the GAN literature in subsection 4.3.1.

While employing this method, we noticed that we had multiple sets of hyper-parameters all with very similar Wasserstein distances, especially at the end of the tuning process. Therefore we compared all GANs where its Wasserstein distance was under a pre-specified threshold. Particularly helpful considering that not all GANs with a similar Wasserstein distance could capture the statistical properties of the real-world data equally well. Based on the output of this tuning process, the best LSTM-GAN, LSTM-TCN-GAN, and W-GAN-GP are shown in section 4.4. The best sets of hyper-parameters can be found in table C.3 in appendix C in case a reader is interested, although the results are dependent on the dataset used to train the GAN. In the next section, these best-performing GANs will be analysed.

4.4 Quantitative analysis of GAN output

To validate the GAN, its ability to create realistic financial returns is determined based on graphs and tests of the most common statistical properties often exhibited by financial time series as discussed in section 2.4. These properties are compared to the real-world data to determine whether we were able to train a decent GAN. To somewhat reduce the effects of the noisiness, especially an issue when measuring the coarse-fine volatility correlation and gain/loss asymmetry, we decided to sample 5 batches of length 1000 and averaged the statistics out. This makes the time-series graphs in the following subsections look random but it improves the quality of the statistical measures.

4.4.1 Statistical properties real-world data

Figure 4.1 shows the statistical properties of the log returns of the BHP stock. By analysing the time series graph we can see that its mean is around 0 and volatility clustering is present. Based on the plot, there appears to be little auto-correlation in the returns. The Ljung-Box test, up to lag 10, resulted in $p-value \approx 0.4\%$, rejecting the Null hypothesis and indicating statistically significant auto-correlation. The power law exponent α is larger than 3 (tested $\alpha - 3 = \beta > 0$) and $p-val_1 \approx 0$ (as explained in section 2.4) indicating statistically significant heavy-tails. The absolute returns are correlated and also follow a power law distribution, indicating volatility clustering. The Ljung-Box test of the absolute returns up to lag 10 resulted in $p-value \approx 0\%$, indicating statistically significant volatility clustering. The leverage effect is also present in the data, showcased by both positive and negative correlations between returns and future

volatility. The coarse-fine volatility is a noisy measure, but the orange line showcases that fine volatility can somewhat predict coarse volatility due to the correlation difference being slightly negative. The gain/loss asymmetry is actually a bit too noisy to compare. However, the peak value indicates that a price decline (Red) is reached slightly faster than a price increase (Blue), although the results here are not convincing.

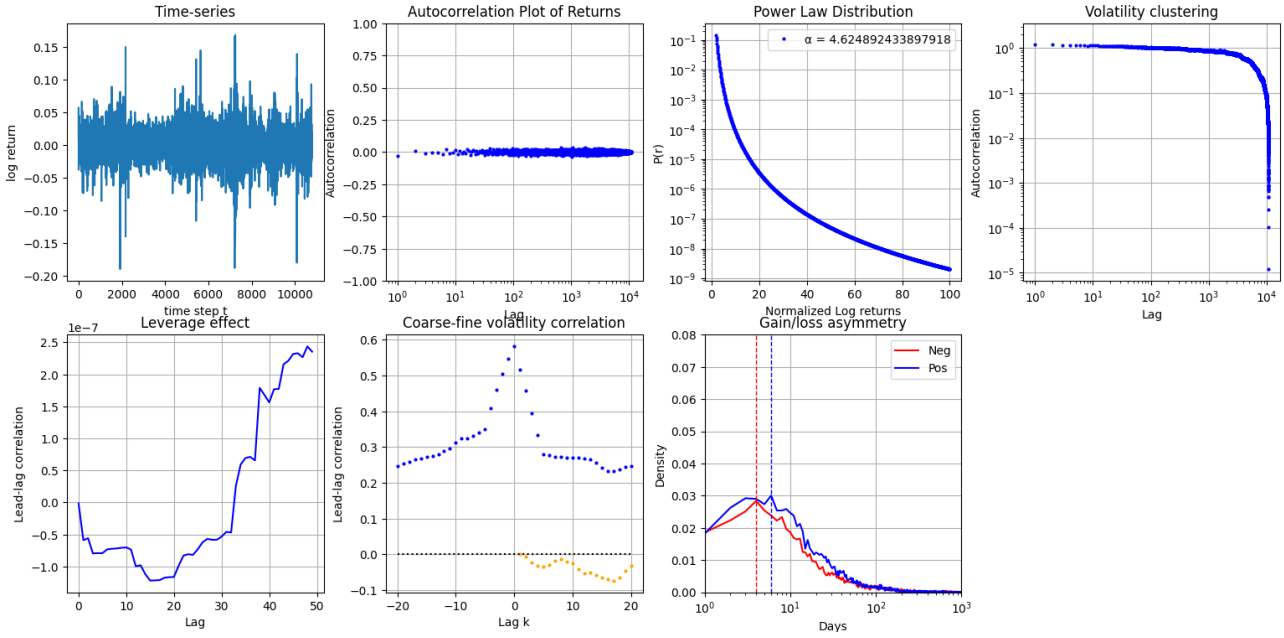

Figure 4.1: Statistical properties of BHP returns

4.4.2 Statistical properties of LSTM-GAN

Figure 4.2 shows the output of the LSTM-GAN. It is immediately clear that this GAN did not perform well considering that the time series graph shows negative bias in the returns, which also influences the other statistics. The returns appear to be positively auto-correlated, although converging to 0 for higher lags. The Ljung-Box test for auto-correlation up to lag 10 resulted in $p - value \approx 0\%$, indeed indicating statistically significant auto-correlation. Both the real-world and GAN's data show significant auto-correlation, however, the graph indicates stronger auto-correlation in the GAN's data.

The distribution in the tails is very similar to the real-world data shown by the power law exponent α having a value near the real-world stock-dataset's α. Based on the likelihood ratio test, the power law exponent α is larger than 3 and $p-val_1 \approx 0$ indicating statistically significant heavy-tails, similar to those observed in the real-world data. It is important to note that the likelihood ratio test assumes that the Null hypothesis (H_0) specifies a fixed parameter value for the unrestricted data. However, in the case of comparing the exponent α between GAN data and real-world data, both α parameters are random variables influenced by the specific sample chosen. This randomness and dependence on the sample make it challenging to satisfy the assumption of fixed parameters required by the likelihood ratio test. This is the reason not to use the likelihood ratio test directly to test whether the GAN data's power law exponent α is equal to the real-world data's α. Instead, we only tested whether both have significant signs of heavy-tails and not whether they have an equal distribution in the tails.

Volatility clustering is present in the GAN, although slightly higher for lower lags, and it converges towards zero for higher lags similar to the real-world stock-dataset. The Ljung-box test of the absolute returns up to lag 10 has $p-value \approx 0\%$, implying significant volatility clustering. The GAN seems to over-capture the leverage effect, shown by lead-lag auto-correlations

surpassing the real-world values. It is not too relevant whether the lead-lag correlation is negative or positive considering that it is dependent on the current market conditions (Qiu et al., 2006). The real-world data also shows both positive and negative leverage effects. As mentioned before, the coarse-fine volatility and gain/loss asymmetry are noisy measures thus the results should be taken with a grain of salt. The fine volatility cannot really predict coarse volatility in the GAN data, $\Delta\rho^{\tau}_{cf}(k)$ (the orange dotted line) remains around 0. Lastly, the gain/loss asymmetry seems present considering that the peak of negative returns occurs before the peak of positive returns. However, it should be noted that the GAN created so few positive returns, as can be seen from the time-series graph, that only a couple of positive measurements are included in the calculation making this property incomparable. In conclusion, the LSTM-GAN does not produce data that is very similar to the real-world data. It should be noted that the GAN output will be standardized before using it in the geometric Brownian motion of the GAN-QMC simulation, removing the negative bias. Standardizing a distribution will only affect the gain/loss asymmetry due to the way this statistic is defined and not the other statistics chosen here.

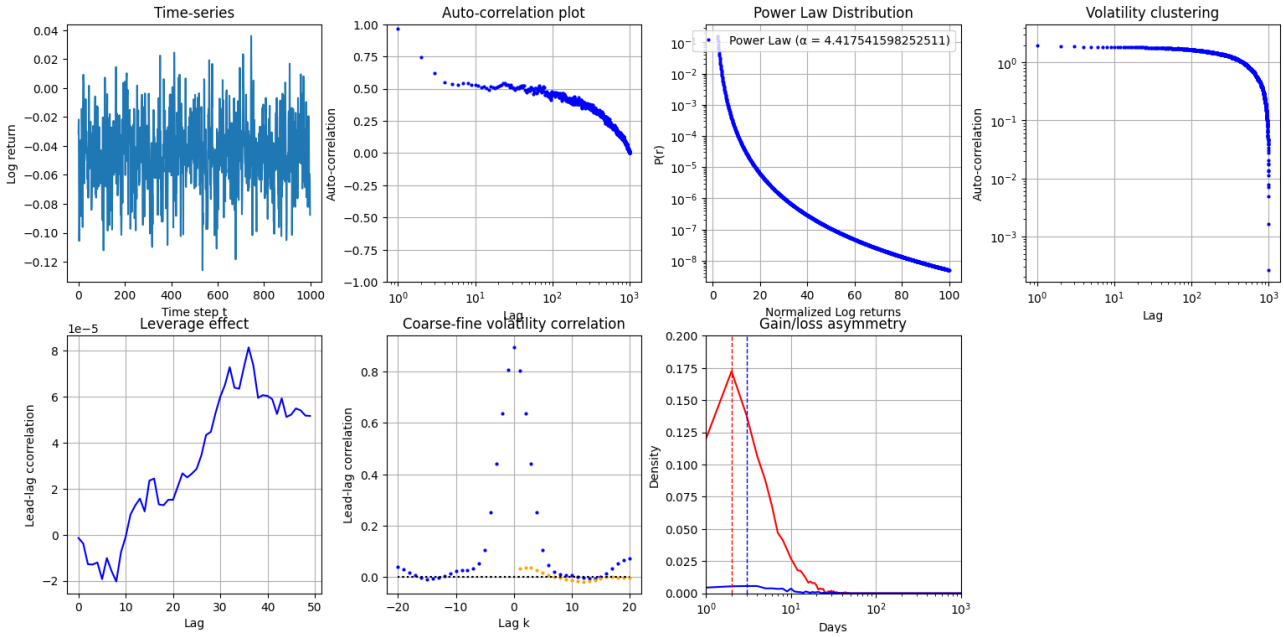

Figure 4.2: Statistical properties of LSTM-GAN

4.4.3 Statistical properties of LSTM-TCN-GAN

Figure 4.3 depicts the statistical properties of the LSTM-TCN-GAN. The time series shows a small bit of negative bias in the returns. The auto-correlation is large for a low number of lags but fades away afterward. The Ljung-Box test up to lag 10 shows that the auto-correlation is significant with $p - value \approx 0\%$. The likelihood ratio test shows that the power law exponent α is larger than 3 and $p - val_1 \approx 0\%$ indicating statistically significant heavy-tails. Volatility clustering is being captured because the Ljung-box test for absolute returns up to lag 10 has $p - value \approx 0\%$. However, there appears to be too much auto-correlation for lower lags similar to the LSTM-GAN. The leverage effect seems to be captured well considering that the absolute maximum lead-lag correlation is similar to that of the real-world stock-dataset. The coarse-fine volatility correlation and gain/loss asymmetry show promising results but granted, these are noisy measures either way. The negative asymmetry between coarse and fine volatility seems to be present but is very small. The peak of negative returns occurs slightly before positive returns similar to the real-world stock-dataset. Overall the LSTM-TCN-GAN performs

noticeably better than the LSTM-GAN.

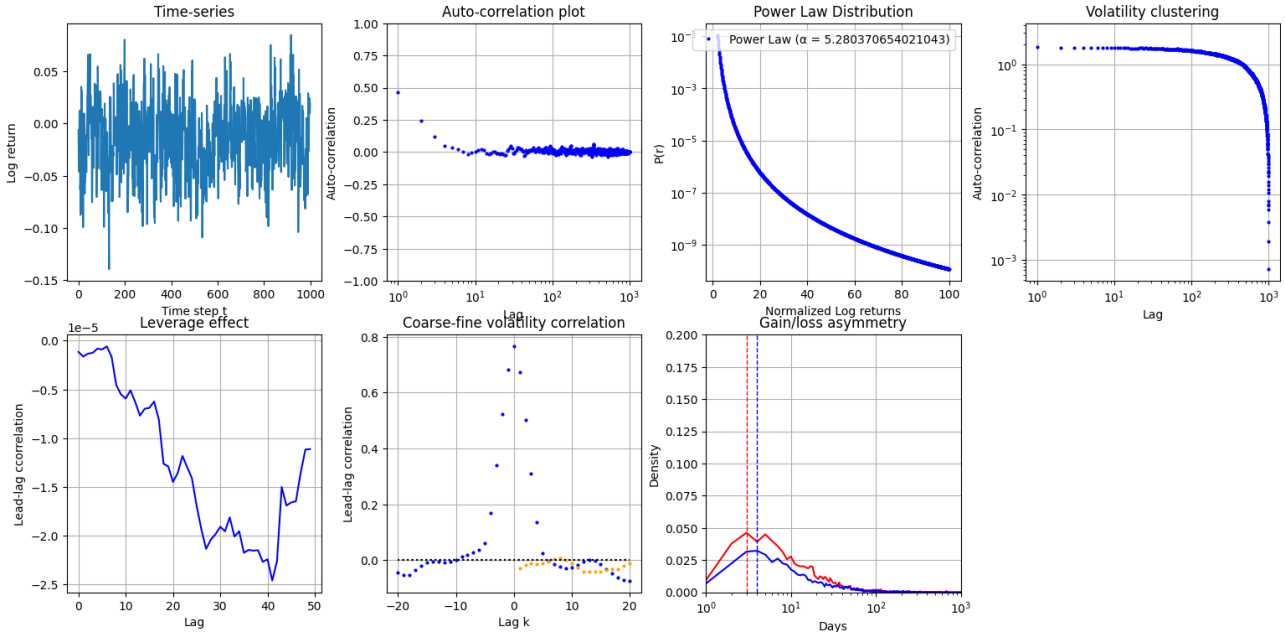

Figure 4.3: Statistical properties of LSTM-TCN-GAN

4.4.4 Statistical properties of W-GAN-GP

In figure 4.4 the results of the W-GAN-GP are shown. The average returns are roughly zero. The W-GAN-GP shows signs of auto-correlation up to lag 10, statistically significant as the Ljung-Box $p - value \approx 0\%$, although it quickly converges to zero. The power law exponent α is larger than 3 and $p - val_1 \approx 0\%$, indicating statistically significant heavy-tails. Volatility clustering is statistically significantly present in the data (Ljung-Box $p - value \approx 0\%$ up to lag 10) but like the other GANs, the auto-correlation in the absolute returns is too high for low lags. The leverage effects are definitely captured showing both positive and negative lead-lag correlations, although it is a bit stronger than the real-world data. Fine volatility has almost no predictive ability for coarse volatility shown by the orange dotted line hovering around 0. The negative returns decline at a faster rate than the positive returns increase considering that the peak of the negative cumulative returns distribution occurs before the positive one, similar to the real-world data.

4.4.5 Conclusion of GAN's abilities

The figures showed that none of the GANs were able to capture all the statistical properties identically to the real-world dataset. Figure 4.5 shows the W-GAN-GP's output compared to a large random sample of standard normally distributed data. Interestingly enough, the W-GAN-GP creates data with a similar distribution to the standard normal distribution. The other GANs deviated slightly more from the standard normal distribution.

Figures 4.2, 4.3, and 4.4 show some interesting observations and differences between the GANs. LSTM-GAN was good at capturing heavy-tails and solid at capturing volatility clustering, and leverage effects. The LSTM-TCN-GAN was good at capturing the auto-correlation and leverage effects and was solid at capturing volatility clustering and the gain/loss asymmetry. The W-GAN-GP has a mean around zero and captured the heavy-tails quite well. In addition, it was solid at capturing auto-correlation, volatility clustering, leverage effects, and

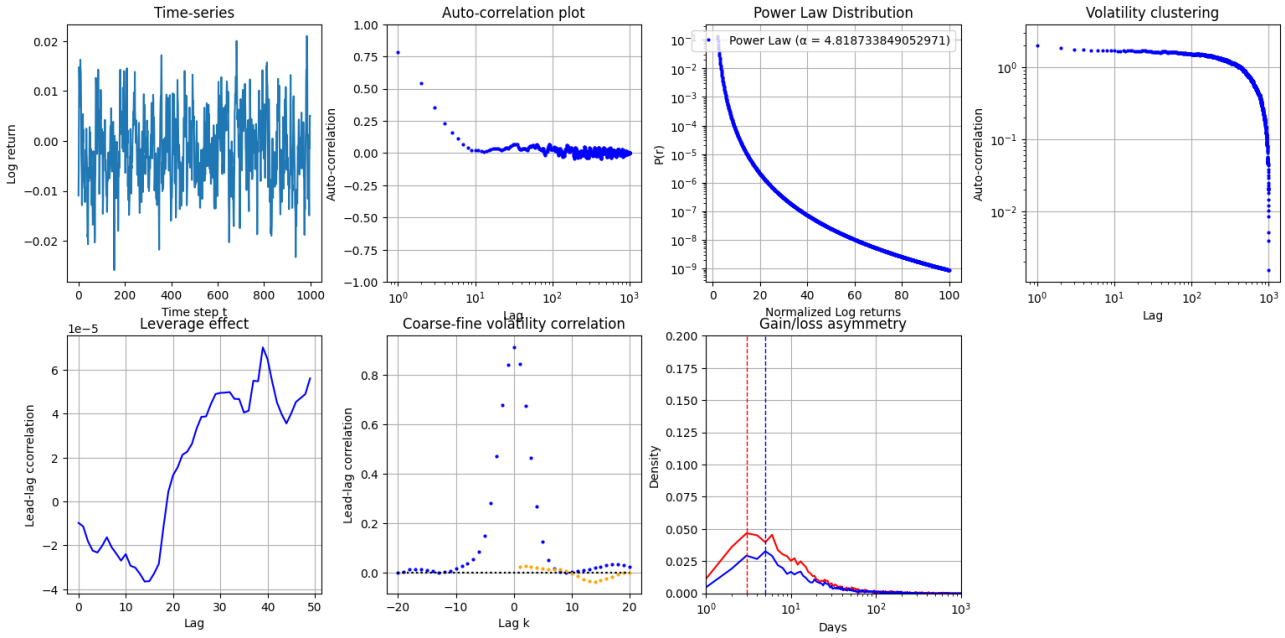

Figure 4.4: Statistical properties of W-GAN-GP

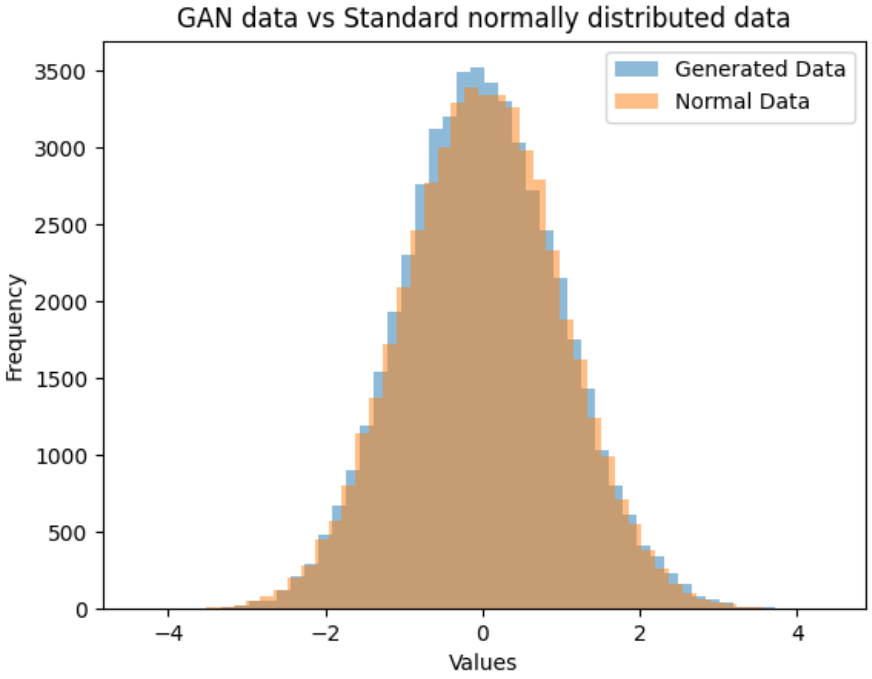

Figure 4.5: Histogram of GAN data vs standard normally distributed data

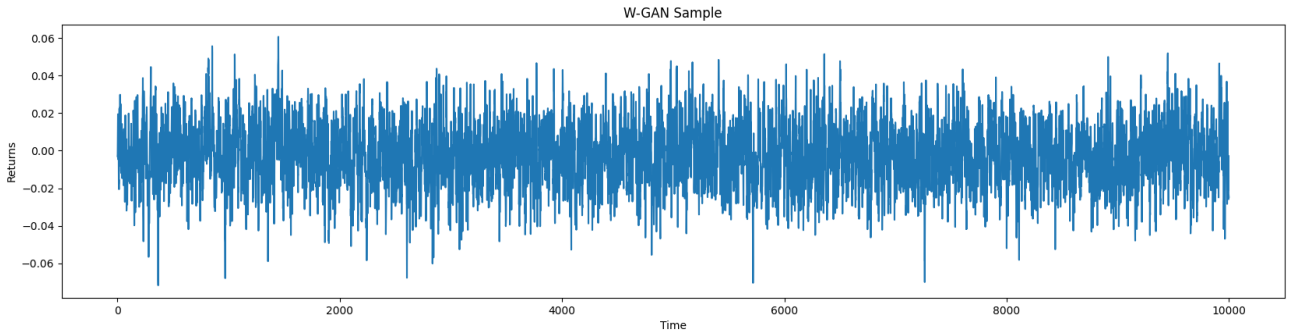

Figure 4.6: W-GAN-GP Sample

gain/loss asymmetry. None of the GANs were able to capture any significant coarse-fine volatility correlation, although LSTM-TCN-GAN might be slightly less poor than the other GANs. To conclude, the W-GAN-GP seems to perform best of all GANs tested in this thesis. However, the GAN's ability to capture all the statistical properties of a financial time series is not great which seems to be in line with previous research (Eckerli et al., 2021). This will most likely harm the effectiveness of the approach when pricing options using a GAN in the MC/QMC simulation.

It should be noted that the time-series graph is also averaged over multiple GAN samples, making it appear random. This graph was thus mainly useful to check the average return. Figure 4.6 shows a single sample from the W-GAN-GP clearly showing some degree of autocorrelation and volatility clustering, looking less random and somewhat more comparative to the real-world log returns of the BHP stock.

Instead of deploying GANs, alternative models could have been used to capture the statistical properties of the underlying asset. A solid possibility is to use a volatility model. However, research shows that many of the volatility models are not able to capture all statistical properties of real-world financial time series data either (He, 2020) (Malmsten et al., 2010). Similar to GANs, when selecting a volatility model, there is always a trade-off involved between which statistical property is modeled most realistically depending on the choice of model.

4.4.6 Sanity check W-GAN-GP

We use the W-GAN-GP to perform the sanity check mentioned in subsection 3.3.3 because the W-GAN-GP outperforms the other GANs. The goal of the sanity check is to ensure that the GAN generates risk-neutral price paths for different values of Δt. The GAN is trained on daily historical stock prices, thus $\Delta t = 1$ is a daily price change. Table 4.2 shows that the geometric Brownian motion has an error, the percentage difference between the risk-free rate and the GAN's returns, independent of the time step Δt and returns distribution (not controlling for the little amount of uncertainty of the random samples). These results are surprising considering that we expected that a large value of Δt would make the results of the GAN invalid because it is not an actual continuous probability distribution. This is good news for the GAN-QMC because this indicates that the choice Δt does not impact the validity of the results. This allows us to select Δt based on the amount of computational power available (a smaller Δt results in higher computational times but probably higher accuracy).

Distribution in GBM	$\Delta t = 100$		$\Delta t = 1$		$\Delta t = 0.01$	
	$\hat{\mu}$	error	$\hat{\mu}$	error	$\hat{\mu}$	error
GAN	1.93e−2	3.64%	2.05e−4	2.30%	2.07e−6	3.39%
Normal	1.95e−2	2.67%	2.07e−4	3.48%	1.92e−6	3.97%

Table 4.2: Sanity check

Chapter 5

Empirical Results

In this chapter, the performance of all pricing methods is analysed. In addition, we conduct a sensitivity analysis to determine the distribution of the option price estimates for different input parameters.

5.1 Analysis

In this section, the MC, QMC, and GAN-QMC are compared for 3 different companies. The GAN-QMC is tested both with a quasi-random and pseudo-random sequence, the latter thus actually being a GAN-MC method. In addition to the company BHP, the GAN was tuned separately for 2 additional companies, namely the Commonwealth Bank of Australia (CBA) and CSL Limited. The statistical properties of the GANs trained on these 2 companies are shown in appendix B. The GAN-QMC method is implemented using a W-GAN-GP because it was deemed most capable of capturing the statistical properties of financial time series as shown in section 4.4.

5.1.1 Data

As explained in subsection 4.1.1, we used stock-datasets to tune the GANs and an option-dataset for subsequent analysis. Each stock-dataset consisted of the daily closing prices of one of the selected companies and was split into a training and test set. The training data of CBA ran from 1992 up to 2015 and the test set from 2015 up to mid-2023. The training data of CSL ran from 1994 up to 2015 and the test set from 2015 up to mid-2023.

The option-dataset contains historically traded European and American options, including their corresponding parameters, combined for all companies. For each option priced using GAN-QMC, the GAN trained on the option's underlying asset was used. For example, when pricing a European call option on the company BHP using GAN-QMC, the W-GAN-GP trained on the BHP stock-dataset is used.

In total, the option-dataset consists of 454 options, where 185 are European and 269 are American. The moneyness ranges from -18.39 to 44.55 with most options being around 0 as displayed in figure B.8 in appendix B. The time to maturity ranges from 3 to 93 days. The entire option-dataset can be found on the following GitHub page: https://github.com/sP22222/GANs-Q-MC.

5.1.2 Least-squares MC vs Averaging method

The results of the least-squares MC against the averaging method are discussed first because the best-performing method will be used for the remainder of all analyses in this chapter when

pricing American options. Table 5.1 shows that there is no statistically significant difference between both pricing methods although the mean MAPE indicates that the averaging method slightly outperforms the least-squares MC. The computational time of the averaging method with a value of 0.021 is much faster than least-squares MC with a value of 1.66. This is mainly due to the necessity to fit a least-squares regression for every time step, slowing down the computations for the least-squares MC. Considering that doing all these analyses is very computationally intensive, a low computational time is important, thus we continue this chapter by using the averaging pricing method for American options.

Pricing Method	**Mean MAPE**	**95% CI MAPE**
Least-squares	3.71%	[3.45%, 3.97%]
Averaging	3.41%	[3.17%, 3.66%]

Table 5.1: MAPE Least-squares vs Averaging method

5.1.3 Statistical Measures

The statistical measures are calculated for BHP, CBA, and CSL for MC, QMC, GAN-MC, and GAN-QMC all using the W-GAN-GP. The measures are averaged over the 3 companies. The main goal is to assess whether using a GAN within QMC simulation can improve the accuracy and computational efficiency of option pricing. This analysis is done for 500, 5000, and 50.000 simulation trials for time step $\Delta t = 1$.

Accuracy

The combined results for BHP/CSL/CBA simulations are presented in tables 5.2, 5.3, and 5.4. The tabulated values represent the 95% CI of the MAPE: [Lower bound, mean MAPE, Upper bound].

Number of	European Call			European Put		
Trials (N)	500	5000	50000	500	5000	50000
MC	[4.89%, 6.01%, 7.13%]	[3.46%, 4.30%, 5.15%]	[2.81%, 3.55%, 4.29%]	[6.12%, 7.52%, 8.92%]	[3.15%, 4.12%, 5.10%]	[2.71%, 3.65%, 6.00%]
QMC	[3.45%, 4.41%, 5.38%]	[3.06%, 3.84%, 4.62%]	[2.89%, 3.64%, 4.40%]	[3.74%, 5.01%, 6.27%]	[2.65%, 3.56%, 4.46%]	[2.64%, 3.54%, 4.44%]
GAN-MC	[5.91%, 7.16%, 8.42%]	[3.00%, 3.87%, 4.74%]	[2.92%, 3.70%, 4.45%]	[6.71%, 8.28%, 9.85%]	[2.89%, 3.83%, 4.77%]	[2.41%, 3.36%, 4.31%]
GAN-QMC	[3.75%, 4.73%, 5.71%]	[2.93%, 3.70%, 4.47%]	[2.91%, 36.9%, 4.45%]	[3.98%, 5.10%, 6.21%]	[2.78%, 3.70%, 4.55%]	[2.52%, 3.42%, 4.32%]

Table 5.2: MAPE European Options

Number of	American Call			American Put		
Trials (N)	500	5000	50000	500	5000	50000
MC	[6.22%, 7.39%, 8.55%]	[3.10%, 3.71%, 4.34%]	[2.91%, 3.52%, 4.14%]	[6.79%, 8.38%, 9.97%]	[3.47%, 4.23%, 4.99%]	[2.80%, 3.56%, 4.33%]
QMC	[4.24%, 5.02%, 5.79%]	[2.87%, 3.49%, 4.10%]	[2.87%, 3.48%, 4.09%]	[5.53%, 6.61%, 7.68%]	[2.93%, 3.65%, 4.37%]	[2.77%, 3.53%, 4.28%]
GAN-MC	[5.71%, 6.74%, 7.77%]	[3.28%, 3.97%, 4.66%]	[2.82%, 3.43%, 4.03%]	[6.63%, 7.93%, 9.22%]	[3.79%, 4.64%, 5.50%]	[2.78%, 3.53%, 4.28%]
GAN-QMC	[3.92%, 4.70%, 5.48%]	[2.93%, 3.56%, 4.20%]	[2.87%, 3.48%, 4.09%]	[4.90%, 5.94%, 6.98%]	[3.07%, 3.83%, 4.59%]	[2.77%, 3.52%, 4.26%]

Table 5.3: MAPE American Options

For 500 trials the QMC and GAN-QMC significantly outperform the MC and GAN-MC, aligning with our expectations. All pricing methods converge once the number of trials is increased. While GAN-QMC did not statistically outperform the QMC method, it is interesting to note that the GAN-MC and GAN-QMC outperform the MC and QMC respectively for 500 simulation trials for American options and vice versa for European options. These differences, however, were not statistically significant.

Combining all option types revealed no statistically significant difference between GAN-QMC and QMC. Again, it can be said that maybe for a low number of simulation trials, the GAN-QMC is slightly superior to all other methods, but these results are not statistically significant. For a low number of trials, GAN-QMC does statistically outperform the GAN-MC

Number of Trials (N)	Calls & Puts		
	500	5000	50000
MC	[6.85%, 7.48%, 8.11%]	[3.65%, 4.04%, 4.43%]	[3.13%, 3.51%, 3.88%]
QMC	[4.95%, 5.48%, 6.01%]	[3.24%, 3.61%, 3.98%]	[3.12%, 3.49%, 3.86%]
GAN-MC	[6.78%, 7.42%, 8.07%]	[3.63%, 4.02%, 4.41%]	[3.16%, 3.53%, 3.90%]
GAN-QMC	[4.62%, 5.10%, 5.58%]	[3.23%, 3.60%, 3.97%]	[3.12%, 3.49%, 3.86%]

Table 5.4: Combined MAPE Options

method. This indicates, as expected, that using a quasi-random sequence in the GAN-based simulation pricing method improves its convergence. The GAN-QMC seems to behave similarly to the QMC indicating that we can indeed model the returns of the underlying asset using a GAN instead of assuming normality. By replacing the standard normal distribution in the geometric Brownian motion with the GAN (sampling from the Generator instead of the standard normal distribution) we can achieve similar results for modeling the stock price process without making distributional assumptions.

Efficiency

Measuring computational efficiency is tricky because it is partially dependent on which coding libraries are used. The quasi-random sequences were generated with different libraries than the pseudo-random sequences. The pseudo-random sequences are more common and those libraries have therefore been developed in C to have speed advantages like more efficient memory allocation. So for a fair computational efficiency comparison, all methods should be programmed in a fast language like C with the same libraries. In that case, we expect quasi-random sampling only to be slightly slower than pseudo-random sampling. To combat these implementation mismatches, we only compare QMC with GAN-QMC (same implementation) because these have shown to be the most accurate for the least number of trials.

Table 5.5 shows the 95% CI's of the computational times of QMC and GAN-QMC for 50.000 trials, implemented with the exact same programming libraries. The GAN-QMC is statistically significantly the fastest pricing method, showing that combining QMC with a GAN is relatively efficient. This suggests that quasi-random sampling from an empirical distribution is faster than from a continuous distribution, probably due to no need for extra computations once the empirical distribution has been sampled.

If we adjust the errors in table 5.4 with their computational times, the error would have been 4.51% for QMC compared to 3.60% for GAN-QMC. Although this comparison assumes a linear increase in performance, which figure 5.1 showed not to be the case. In addition, we did not take into account GAN training time and the construction of an empirical distribution. It is nevertheless interesting to see that a 20% difference in computational time has a big impact on accuracy and makes the GAN-QMC significantly more accurate than regular QMC. To conclude, the exact pricing model used in a QMC simulation does impact the computational times and the GAN-QMC seems to be most computationally efficient.

Method	**Computational time (s)**
QMC	[4.92e−2, 5.34e−2, 5.77e−2]
GAN-QMC	[3.93e−2, 4.26e−2, 4.60e−2]

Table 5.5: Computational times QMC vs GAN-QMC

We also conducted a short analysis for various time steps Δt. However, some initial testing showed similar differences between QMC and GAN-QMC, so we decided not to conduct a more in-depth analysis. Lowering Δt lowers the MAPE of all methods and reduces the width of the

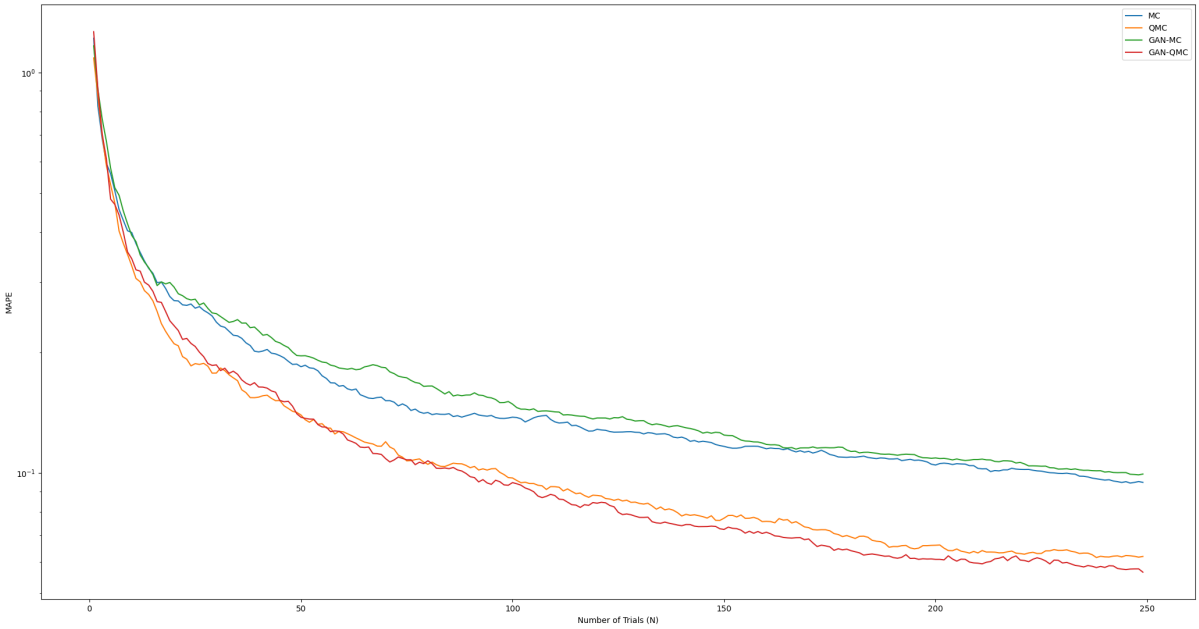

Figure 5.1: Simulation convergence

confidence intervals significantly. However, the MAPEs converge to the same small confidence intervals instead, still not measuring any significant difference between the methods. It should also be noted that lowering Δt leads to issues when generating a quasi-random sequence because the dimensionality can become too large. If computational time is not an issue, all methods converge to the same low MAPE of 2% with roughly 10.000 trials for Δt as small as possible. If computational times are important, the best results seem to occur using GAN-QMC with a couple of thousand trials by decreasing Δt only slightly (e.g. $\Delta t = 0.5$).

5.1.4 Analysing graphs

We now analyse the graphs for different parameters as discussed in section 4.1.3. The statistics of all graphs are calculated on price estimates of roughly 500 options.

Convergence

The convergence graph, depicted in figure 5.1, shows patterns that are as expected for a low number of simulation trials. First of all, the quasi-random sequence performs better than its pseudo-random counterpart. The MC simulation seems to slightly outperform the MC-GAN, but the difference is minor. The GAN-QMC slightly outperforms the QMC, but here the differences are small too. Overall using a GAN does not result in faster convergence, but as expected a quasi-random sequence does. Considering the MAPEs converged quickly, we will do the rest of the analysis with 500 trials, significantly reducing the computational times. This choice might show differences in performance somewhat more noticeably, but the results will have a bit more uncertainty. To battle this, we average each performance metric and only analyse its trends. Plotting every single individual data point gives hard-to-read results either way. The plots were constructed by fitting a polynomial function to the data points by minimizing the least-squares error.

Moneyness

Figure 5.2 shows the moneyness of the option against the MAPE for each pricing method. Considering that most options were ATM, the difference in trend between QMC and GAN-

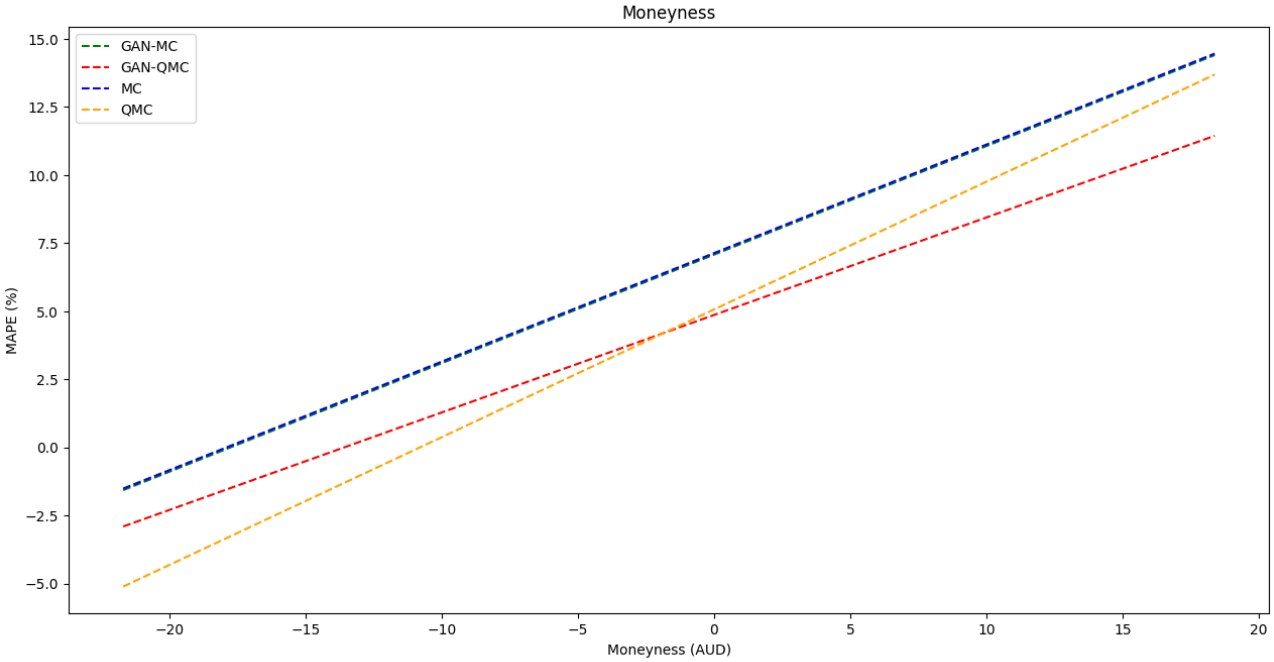

Figure 5.2: Moneyness

QMC is too small to draw any conclusions. It seems that the chosen method (assuming using the same random sequence) does not influence the ability to price options ITM, ATM, or OTM at least within the moneyness range $[-10, 10]$.

Maturity

Figure 5.3 shows really interesting results. The longer the time to maturity of the option becomes, the better both GAN methods perform relative to the regular MC/QMC whose performance decreases the more maturity increases. For options where the time to maturity is more than 3 months, the differences can easily be more than 1.5 percentage points. Potentially, the performance difference becomes more noticeable because the GAN models the returns slightly more realistic than a standard normal distribution, which becomes more noticeable the larger the maturity of the option. Preferably we would do a more in-depth analysis using thousands of options with a maturity above 3 months. However, this was not possible since not much data is available here because the liquidity of free-of-charge data for long-maturity options was an issue.

Implied Volatility

The option-dataset mainly contains options with yearly implied volatilities in the range $[10\%, 35\%]$. There are 12 options in the dataset where the implied volatility is far above 35%. We took these outliers out of the data because they do influence the trend but there aren't enough observations to draw any conclusions on the ability to price options with such large implied volatilities. Figure 5.4 shows the trends of the MAPE with regard to the implied volatility for all methods. The performance differences between MC/QMC and GAN-MC/GAN-QMC never become very significant.

Combining all results, the performance between MC/QMC and GAN-MC/GAN-QMC seems

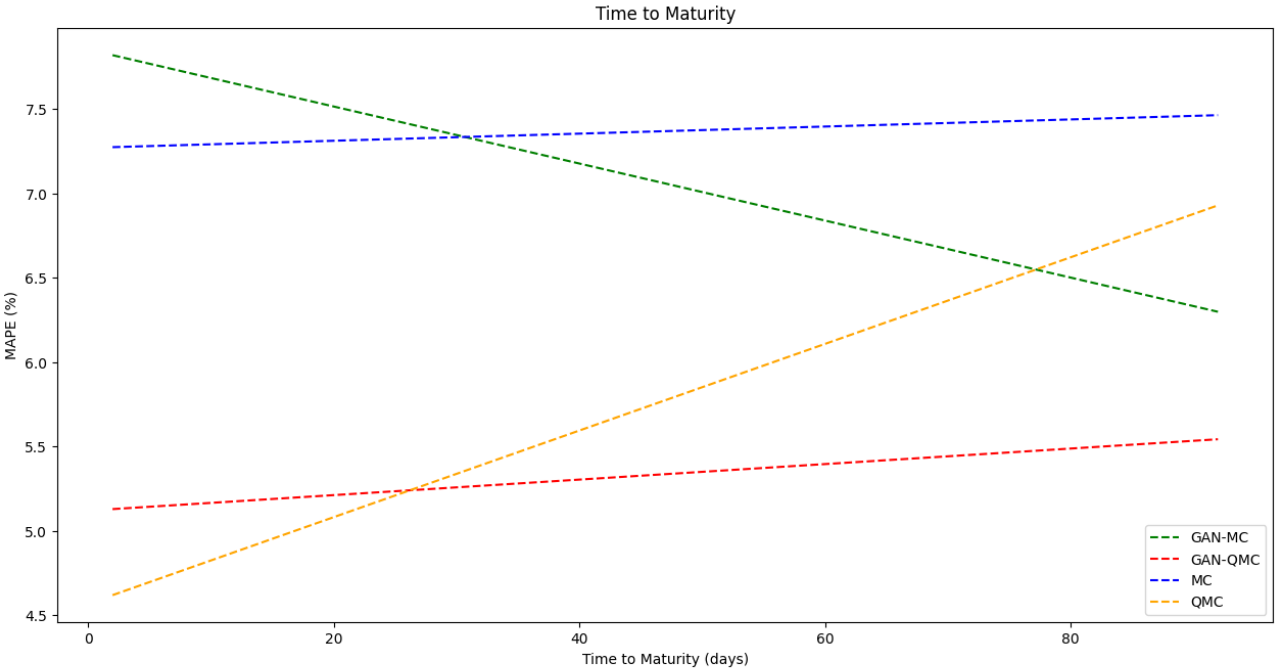

Figure 5.3: Time to Maturity

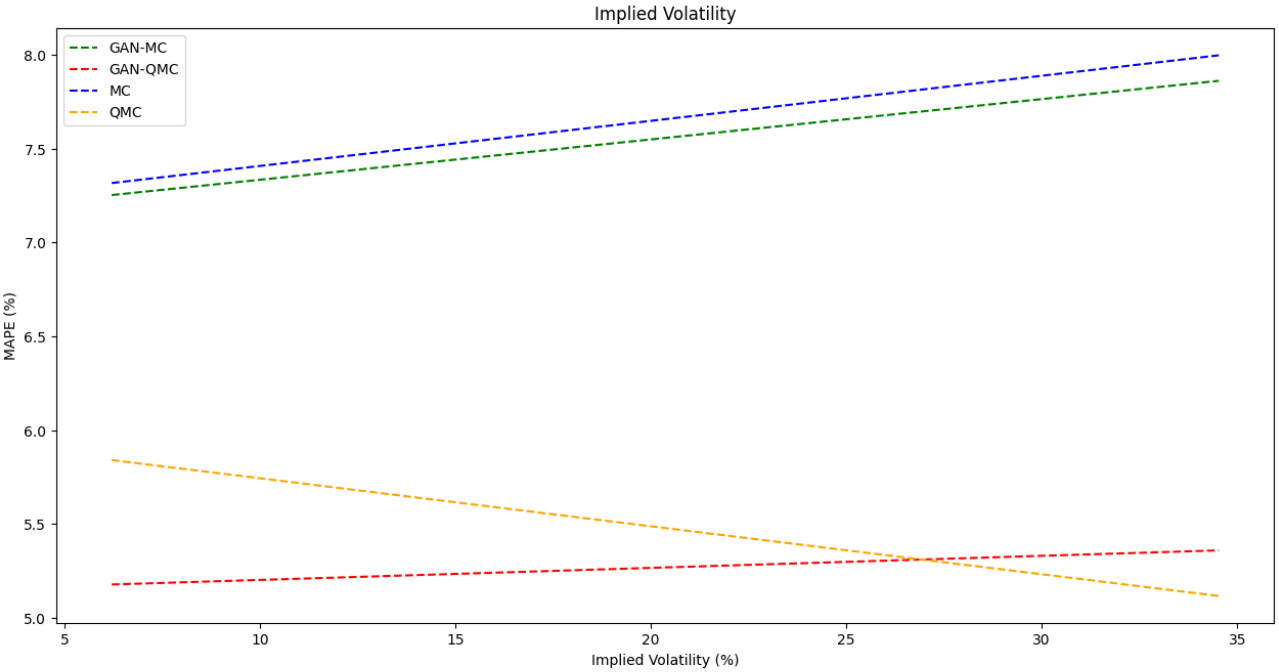

Figure 5.4: Implied volatility

pretty similar overall. However, for options with a long maturity both GAN-MC and GAN-QMC seem to perform better than their counterparts.

5.2 Sensitivity analysis

To analyse some details and differences between the pricing methods more in-depth, we will deploy a simple sensitivity analysis. We follow a simplified approach based on the paper of Fu et al. (2009) to determine the sensitivity of the different methods with regard to some of the input parameters.

5.2.1 Sensitivity analysis explained

The sensitivity analysis consists of 4 steps. The goal is to determine how sensitive the option price is to changes in the input parameters.

1. **Parameter selection:** The key parameters to change must be selected first. For option pricing using a MC simulation, we deemed the stock price, strike price, time to maturity, implied volatility, and risk-free rate as relevant parameters.

2. **Vary the parameters:** The chosen parameters must be varied within a pre-specified range, wide enough to capture a significant amount of parameter impact on the option price. The stock and strike price will both be varied within the interval $[80, 120]$. The time to maturity will be varied within the interval $[10, 250]$, the implied volatility within $[5\%, 65\%]$, and the risk free rate within $[0.1\%, 6\%]$.

3. **Run the simulation:** We will create datasets where 1 parameter at a time is adjusted within the pre-determined interval. The simulations must be run on all these different datasets.

4. **Analyse the results:** This step can be achieved in 2 ways. The first possibility is to calculate the gradient of the estimated option prices with respect to each parameter (Fu et al., 2009). The second possibility, and our choice for this thesis, is to plot the distributions of the estimated option prices for each parameter. This allows us to quickly see the similarities and differences between all pricing methods with regard to their parameter sensitivity. This could partially help to understand why and when a pricing method works well or not.

5.2.2 Sensitivity analysis results

We now analyse the results of the sensitivity analysis comparing MC, QMC, GAN-MC, and GAN-QMC. The distributions for the various selected parameters are shown in appendix B. The distributions were estimated using a Gaussian kernel-density estimation (Yen, 2017), with the bandwidth determined by Scott's method (Scott, 1994). Overall the results show almost complete overlap in distributions between all methods. The sensitivity of the stock price, strike price, implied volatility, and risk-free rate do not differ that much. The sensitivity towards maturity seems to be the largest, however is influenced by OTM options. Therefore we decided to do a more in-depth sensitivity analysis for the time to maturity of ATM and ITM options, see figure 5.5. This shows that the higher the estimated price the more the methods differ, especially within the interval $[10, 20]$. Since a longer maturity increases the option price, and no other parameter was varied in figure 5.5, it can be concluded that the difference in sensitivities of the methods is influenced by maturity. This explains why the methods differ most in their pricing accuracy for different maturities as shown in section 5.1.4

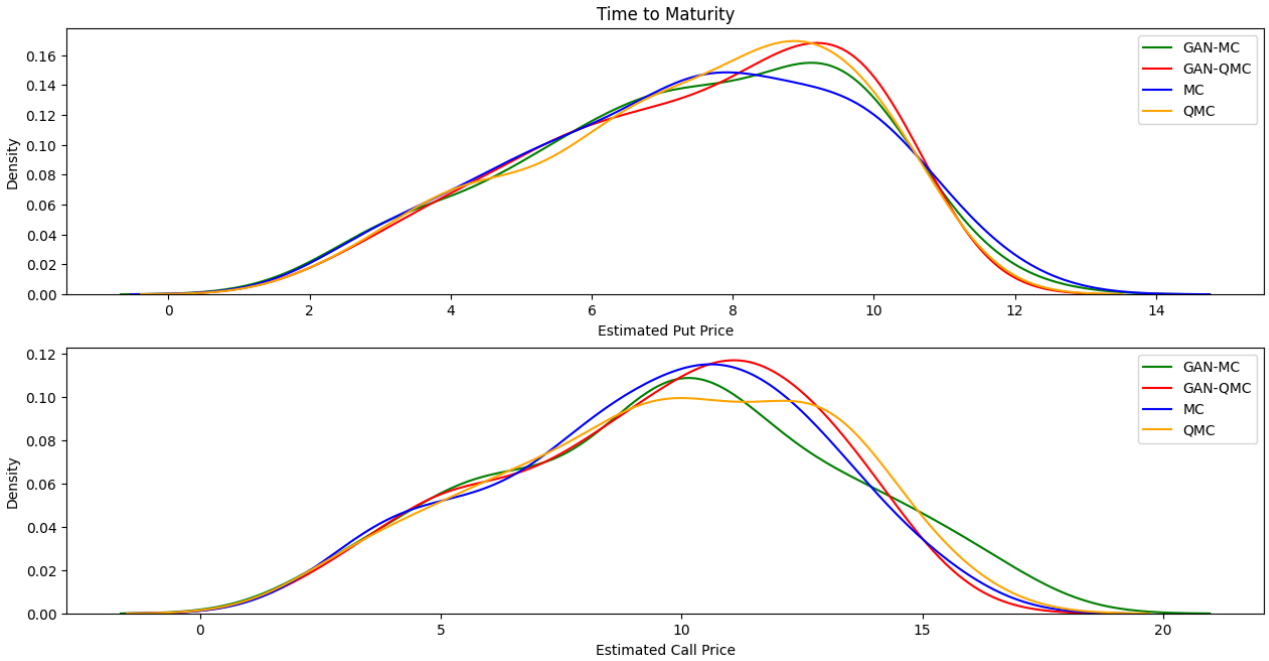

Figure 5.5: Sensitivity analysis of Time to Maturity ATM & ITM

5.3 GAN Tuning results

We also want to reflect on the effectiveness of the automated hyper-parameter tuning process (Sæternes et al., 2023) considering that it has never been employed to tune a GAN. The tuning process has been deployed to 9 GANs. Usually the objective function $U(p, w^*; \mathcal{D}_{val})$ of a tuned GAN decreased by at least a factor of 10 compared to the initial starting hyper-parameters (based on the literature), thus showing promising results. The GAN trained on the initial hyper-parameters lacked any ability to create financial data. Towards the end of the tuning process, its ability significantly improved into a moderately performing GAN as has been shown in section 4.4.

The tuning process was a technique to end up with a set of hyper-parameters at least as good as the initial values. It was surprising that this tuning process was able to decrease the objective function so much, with a relatively low number of iterations. A sequential approach did not work because changing 1 variable influences the impact of another. Alternatively, trying every combination, within a certain interval, would require way too much computational power resulting in a huge training time.

Figure 5.6 shows a plot of the Wasserstein distance U throughout the tuning process. Again, note that this is not the same objective function that the Generator and Discriminator use to play the zero-sum game, necessary to train a single GAN. This method worked well because even if a GAN was trained that resulted in a large U, due to the calculation of equation 4.6 of subsection 4.3.2, the result would be flipped by assuming that changing the parameters in the opposite direction must be better. The graph shows that when a good GAN was found that resulted in a low Wasserstein distance, then the GANs trained in the next iterations would have relatively low Wasserstein distances as well. This suggests that employing an iterative approach, where the previous best GAN serves as a new starting point, from which new GANs are trained using random sets of hyper-parameters around the best GAN, proves to be an effective strategy.

During the tuning process equation 4.5 of subsection 4.3.2 was first used 5 times and its results would then be combined into the special equation 4.6. This process was repeated for 12 iterations. Based on all 9 tuned GANs, equation 4.6 resulted in the best set of hyper-

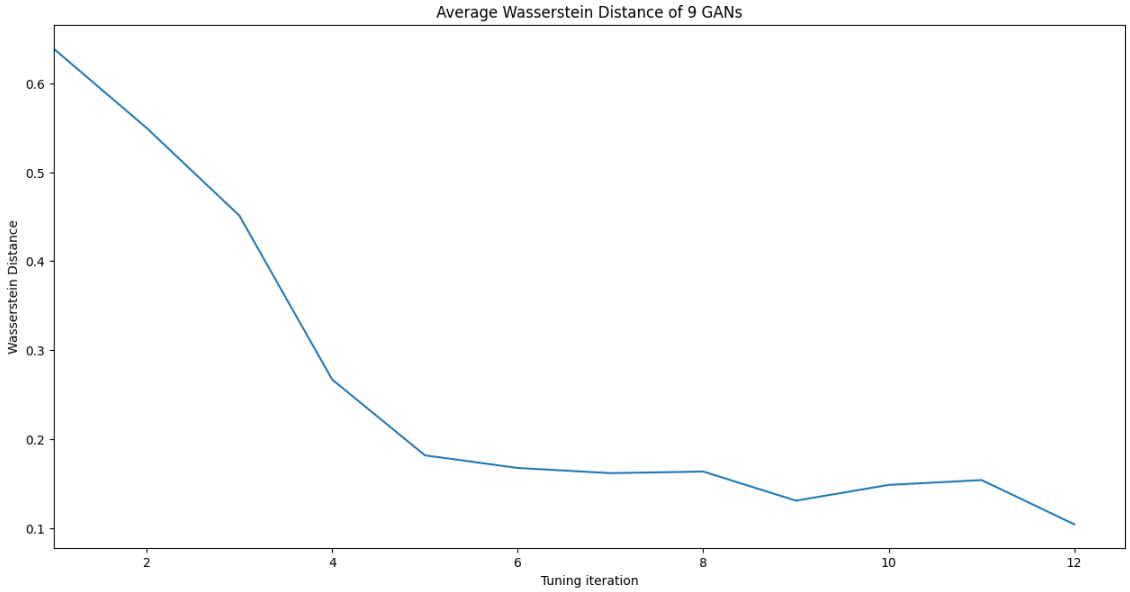

Figure 5.6: Wasserstein distances of Tuning process

parameters 2 times, suggesting that this special formula performed no better than a completely random search. However, if the same tuning process had been employed using only 6 instead of 12 iterations, equation 4.6 would have resulted in the lowest Wasserstein distance 4 times. This implies that the special equation works especially well when a researcher really wants to minimize the computational time of the tuning process by minimizing the number of tuning iterations.

The automated hyper-parameter tuning process might work well because it creates a good set of hyper-parameters combining the results of random searches into the special equation 4.6 of subsection 4.3.2. Once a set of hyper-parameters that resulted in a low objective function had been found (usually within a couple of iterations), the random searches of equation 4.5 ensured that an even slightly better set could be found. So the special equation is helpful to quickly find a good set of hyper-parameters and once this is achieved, randomly searching around this set works well. Even though this tuning process seems promising, it is important to note that a sample of 9 tuned GANs is rather small to draw very convincing conclusions about the effectiveness of this tuning process.

Using formula 4.6, while only continuing with the best GAN after every tuning iteration, resulted in a constantly better set of hyper-parameters where the final set was the GAN that minimized U such that $U(p^{(f)}, w^*; \mathcal{D}_{val}) = min(U^{f-1}, U_1^f, ..., U_m^f, U_{m+1}^f)$ where f is the final iteration. To conclude, this method worked well to determine which hyper-parameters to choose for the GAN training and always improved noticeably on the initial chosen set. The biggest improvements were made in the first couple of tuning iterations. The objective function would always converge to some small value. This tuning process did not necessarily result in the optimal set p^*, considering that the GAN's output is not the same as the real-world data, but it did result in a solid set resulting in a low objective function, capturing some of the statistical properties, while not needing much computational power.

Chapter 6

Conclusion

In this thesis, we developed a new pricing method combining a GAN with a QMC simulation. All relevant code used in this thesis can be found on the following GitHub page: https://github.com/sP22222/GANs-Q-MC. We tested whether the pricing of options using a QMC simulation can be improved by modeling the returns of the underlying asset with a GAN, hopefully, more closely capturing the statistical properties of the underlying asset. We will now provide the key findings of our research, interpret the results, discuss the practical implications, and cover the limitations. Finally, recommendations for further research based on the gaps in this thesis will be given.

6.1 Key Findings

In chapter 5, we carefully examined and compared MC, QMC, GAN-MC, and GAN-QMC to analyse how accurate and efficient they are in their ability to price options. The goal of this research was to improve MC/QMC option pricing by building a new pricing framework implementing a GAN into a QMC simulation, trying to overcome some of the limitations of traditional parametric models. The primary research question was formulated as follows:

> *Can a Generative Adversarial Network and Quasi-Monte Carlo simulation be combined to create a non-parametric pricing framework for pricing different types of options, to achieve more accurate and efficient valuations?*

The foundational groundwork was laid by constructing a theoretical framework, using relevant literature to place the topic within a broader context and to explain the main aspects of option pricing, MC simulation, and GANs. Building on this, the methodology for the GAN-QMC was developed including among other things option pricing using MC/QMC, neural networks, GANs, and the mathematics behind constructing risk-neutral price paths. This step revealed the importance of making certain assumptions to transform the GAN's output into risk-neutral price paths, leading to the realization that a fully non-parametric pricing framework was difficult to attain. Thus the GAN-QMC method is characterized as a semi-parametric pricing method, having fewer assumptions than conventional MC/QMC pricing methods. The assumption was made that the stock follows a process with the same functional form as the geometric Brownian motion, but without assuming that the returns follow a standard normal distribution, as is typically assumed in MC/QMC option pricing. Instead, the distribution of stock returns was captured with a GAN, thereby avoiding distributional assumptions. In section 6.3, we discuss an alternative GAN to create risk-neutral price paths without making any assumptions.

The methodology was implemented by developing the MC/QMC simulation and the GANs. The hyper-parameters of the GANs were tuned using an automated parameter tuning strategy

which improved on the initial GAN, where the hyper-parameters were chosen based on the literature. There have been attempts in the literature to stably tune the parameters of a GAN, but the results appeared mixed. Therefore our novel exploration of applying an existing tuning strategy, which leveraged perturbation theory, to GAN training. Based on the analysis of the tuned GANs we concluded that the W-GAN-GP was better at generating data that captures the most important statistical properties of the real-world financial data than LSTM-GAN and LSTM-TCN-GAN. This indicates that it is important and useful to implement techniques that can improve training stability like Wasserstein distance, Gradient Penalty, and a convolutional Discriminator. Therefore the tuned W-GAN-GP was implemented into the GAN-QMC.

The comparative analysis involved pricing options using MC, QMC, GAN-MC, and GAN-QMC using data from 3 different companies listed on the Australian Stock Exchange. The empirical results, found in chapter 5, showed equal results in terms of accuracy between QMC and GAN-QMC. Despite the GAN not capturing the statistical properties of financial data perfectly, the GAN-QMC did show comparable performance to QMC, suggesting equal performance when modeling the distribution of stock returns through a GAN compared to a standard normal distribution. This implies that we can use a GAN as a non-parametric method to model the returns of a stock. Alternatively, this could also be interpreted that the GAN almost ended up being an empirical variant of the normal distribution, although it's difficult to prove which continuous distribution a GAN would be closest to. Achieving equal performance with GAN-QMC, while modeling the returns of the asset with a GAN instead of assuming normality, is an interesting theoretical result.

Notably, the GAN-QMC potentially outperforms the QMC for a low number of simulation trials, at approximately 500 trials. However, achieving small CIs, to actually draw this conclusion, requires a substantial volume of option data, often requiring money. Our analysis also suggested that the GAN-QMC outperforms QMC for long-maturity options, although data limitations make this claim questionable. For pricing American options, the least-squares and averaging methods showed no significant difference in accuracy, but the averaging method was much faster.

From an efficiency standpoint, sampling from an empirical distribution requires no calculations compared to sampling from a standard normal distribution, which resulted in the GAN-QMC method being about 20% faster. It is important to note that this did not include the GAN training and construction of an empirical distribution (dataset). So if the main goal is low computational times while having a solid accuracy, then the GAN-QMC offers an advantage over regular QMC when the number of trials is low and Δt is not very small.

A not-so-trivial detail has been deployed when quasi-randomly sampling from the Generator. In essence, feeding quasi-random noise to a neural network, the Generator, does not guarantee a quasi-random output. We deployed a pragmatic approach by sampling billions of random numbers from the Generator of the tuned GAN. By ordering this dataset, it was treated as an empirical distribution from which we were able to quasi-randomly sample. This also seems to be the main reason why the GAN-QMC's computational times are faster than QMC because there is no need for any extra calculations once the empirical distribution has been made. In contrast, sampling from a normal distribution requires costly calculations. Deploying the same technique for regular QMC by approximating the normal distribution might result in similar accuracy and computational efficiency as GAN-QMC.

6.2 Discussion

During this research, we developed a semi-parametric pricing framework, not making distributional assumptions, using GANs and QMC. However, several assumptions were made to construct risk-neutral price paths. In this section, we suggest methods to relax several assumptions of QMC/GAN-QMC. We also discuss the limitations, interpretations, and implications of

this research.

6.2.1 Assumptions of GAN-QMC

Option pricing is done in a risk-neutral framework but the GAN was trained on historical stock return data and thus outputs synthetic real-world returns. We needed a way to transform the GAN's output into risk-neutral price paths. To achieve this the geometric Brownian motion was used which implies that the GAN-QMC method had to make the following assumptions:

1. The stock has the functional form of a geometric Brownian motion thus having constant drift and volatility
2. The stock pays no dividends
3. Continuous time
4. No transaction costs or taxes
5. Market is efficient
6. No stock price jumps
7. Constant interest rates
8. Liquid market

Starting with the positives, the GAN-QMC method does not make any distributional assumptions about the returns of the asset, instead, the GAN is used to model the returns of the asset. Assumption 1 can be relaxed by using a FIN-GAN allowing the immediate generation of risk-neutral price paths, which we discuss in more detail in section 6.3. The downsides of FIN-GANs are very unstable training and difficult implementation using encoders and decoders (Tan et al., 2022). The reason not to do this in our research was simply because some initial testing showed the extreme difficulty in building and training a solid performing FIN-GAN. It should be noted that the FIN-GAN is the only method recommended in this section that could lead to potential improvements of GAN-QMC over QMC. All other recommended methods to relax assumptions can be applied to both QMC and GAN-QMC which might thus increase the performance of both equally.

An alternative way to relax assumption 1 is by explicitly modeling the drift and volatility. This could be done with for example a mean reverting stochastic process for the drift and a GARCH model for the volatility (He, 2020).

One straightforward improvement to fix assumption 2 is to incorporate the dividend yield of the underlying stock into the model. The dividend yield represents the annual dividend payment as a percentage of the stock's current price. By adjusting the stock price downward to account for expected dividends, the simulation can better reflect the impact of dividends on the option price. The impact of assumption 3 can be minimized by using small time steps Δt in the simulation improving the discrete-time approximation, although significantly increasing computational times. For assumption 4, both transaction costs and taxes can be incorporated into a MC simulation, but this is heavily dependent on which option is priced in which market. Assumption 5 is not really an issue because stock markets, more often than not, tend to be pretty efficient (Davidson et al., 1982) particularly over an extended time horizon beyond a single day (Worthington et al., 2006). If the GAN is trained perfectly it should include stock price jumps and thus solve assumption 6. Otherwise, the MC can be extended by incorporating a jump-diffusion model (Kirkby, 2019). Assumption 7 can be relaxed by modeling the interest rate using a stochastic interest rate model such as the Cox-Ingersoll-Ross model (Ross et al., 1985). Assumption 8 is not an issue when trading low volumes. Otherwise, a market impact model,

simulating how large traded quantities affect the stock price, can be incorporated (Almgren et al., 2005).

It should be noted that these improvements can be made to both QMC and GAN-QMC (apart from FIN-GAN), such that both methods probably become more accurate to a similar degree by relaxing these assumptions. In addition, these improvements would make the simulation more complex and would increase computational times. Therefore it is important to test which assumptions have a significant negative impact on the pricing accuracy and are thus worth relaxing using complex models.

6.2.2 Limitations

The main limitation of the GAN-QMC is simply the GAN's ability to generate financial time series data. This was partially due to training stability issues. Even when using the automated tuning process the GAN did not perform great. Considering the importance of accurately modeling the option's underlying asset when using GAN-QMC, the GAN's lacking abilities most likely hindered the performance of our new pricing method. Particularly noteworthy was the difficulty in capturing the same volatility clustering as present in the real-world data without capturing too much auto-correlation.

Another limitation is that GANs are a black-box method, meaning that the interpretability of the method is hard. Therefore, explaining the model, and its intricacies, of the returns of the underlying asset to stakeholders or regulators might be challenging or near impossible.

Then there is the limitation of generating risk-neutral price paths with a GAN. We decided to use a geometric Brownian motion leading to several assumptions. This is most likely the reason that GAN-QMC performed so similarly to regular QMC because the stochastic process of the GAN-QMC uses the same functional form, only changing the distribution of the returns.

Apart from limitations arising from parametric assumptions, there are also some methodological limitations, data being the major one. The data availability impacts both the GAN training and analysis. Longer time series seemed to be better for GAN training, noticed because training the GAN on BHP data worked better than on CSL/CBA data, where BHP has been exchange listed the longest. The GAN can only be trained if a company has been exchange-listed for many years, limiting the effectiveness and applicability of the GAN-QMC.

The initial goal of this research was to price exotic options because we expected our method to be particularly useful for price path-dependent options. Due to a lack of freely available historical option pricing data, we decided to focus on European and American options. The code provided on GitHub still contains functions to price exotic options allowing the pricing of Asian, barrier, and lookback options. The amount of freely available historical European and American option pricing data is also limited, meaning that the analysis was done on a relatively small option-dataset.

Due to the time constraint, the choice was made to limit the comparative analysis of the pricing methods to stocks. Therefore, the effectiveness in other markets, like the currency or commodity market, is still unexplored. In addition, this research only compares the GAN-QMC to regular QMC. A comparative analysis with many other common option pricing methods is missing, mainly due to limited time. However, considering the comparable performance of both methods, previous research comparing QMC to other pricing methods can be used to assess the performance of GAN-QMC compared to other existing pricing methods.

6.2.3 Interpretation and implications of the results

The key findings, as outlined in chapter 5, indicate that the GAN-QMC method performs equally well as the regular QMC method in terms of pricing accuracy. Despite the GAN not exactly capturing the statistical properties of financial data, the GAN-QMC did demonstrate

comparable performance to QMC. However, certain nuances and potential applications arise when analysing the method more in-depth.

The GAN-QMC method is a semi-parametric pricing method and showcased comparable performance to traditional QMC for option pricing. Notably, the GAN-QMC exhibited potential advantages over QMC in terms of computational efficiency, being approximately 20% faster. Another intriguing observation was the potential outperformance of GAN-QMC for long-maturity options, although the claim's reliability was questionable due to data limitations. The computational advantage of GAN-QMC in combination with solid accuracy indicates hopeful results for practical applications where computational time is a critical factor. The method's comparative speed advantage, although small, could be leveraged by financial institutions in scenarios where computational time is a crucial factor when pricing long-maturity path-dependent options. However, it should be noted that if the goal of a financial institution is speed over accuracy, it might not consider using simulations to price options at all.

It is interesting to note that an LSTM-GAN often performs similarly to just training a LSTM neural network for creating financial data (K. Zhang et al., 2019). This could imply that it would inherently not matter whether we used the Generator of the GAN or trained an LSTM by itself.

The idea of the GAN-QMC was to improve the pricing accuracy of path-dependent options by more closely modeling the underlying asset which ended up being more challenging than expected. Developing a GAN requires high set-up costs, making this a costly method to develop compared to simpler pricing methods including regular QMC. However, if distributional assumptions are not desired, our developed method can serve as an interesting and effective alternative. In that case, other non-parametric methods to model the asset within the simulation should be considered as well. Such a simulation can be extended by some of the recommendations mentioned in section 6.2.2.

Financial practitioners need to recognize these limitations of the GAN-QMC, particularly the need for certain assumptions and the challenges associated with interpreting GANs as black-box models. A clear understanding of these limitations is crucial when applying the GAN-QMC method in real-world scenarios, although this holds for any option pricing method. If done correctly, the GAN-QMC can be a solid and theoretically interesting alternative to price a wide variety of options without making any distributional assumptions.

6.3 Further Research

In this section, we discuss interesting future research to improve on or add to our research. Most importantly, we suggest using a risk-neutral GAN to directly create risk-neutral financial data. This suggestion might be the most notable way to make the GAN-QMC significantly better than regular QMC for option pricing. We have the following suggestions for further research:

1. **Risk-neutral GANs:** The most interesting and potentially impactful future research is discussed first. We used the GAN in the geometric Brownian motion to generate risk-neutral price paths. Instead, using the GAN to generate risk-neutral data directly makes the GAN-QMC a completely different pricing method than regular QMC. Instead of modeling the stock price process by combining the GAN with the functional form of the geometric Brownian motion, just a risk-neutral GAN can be used. This will set GAN-QMC apart from QMC and could potentially make GAN-QMC more accurate than QMC. Tan et al. (2022) developed a FIN-GAN allowing for the creation of risk-neutral price paths, but implementation and training stability were significant issues here. Future research in the creation of risk-neutral price paths using neural networks could improve the GAN-QMC's accuracy. In addition, it would get rid of some of the assumptions mentioned in section 6.2. The downside is that creating risk-neutral price paths using

a more complex neural network by additionally feeding it the option's implied volatility might lead to increased computational times.

2. **GANs for finance:** At least partially the performance of the GAN-QMC was limited due to the GAN's abilities. The GAN did not mimic the real-world financial stock returns perfectly, which most likely impacted the GAN-QMC's accuracy negatively. GANs are initially designed for image and text generation, where performance seems to be outstanding. However, the application for generating financial time series is less explored. A noticeable issue is training stability which is particularly sensitive to the choice of hyper-parameters. To find a good set of hyper-parameters, without using too much computational power, we employed an automated parameter tuning strategy. This tuning strategy requires an inverse performance measure that quantifies the performance of a GAN given a set of hyper-parameters. We used the Wasserstein distance which is the same measure used to calculate the weights and biases of the neural network itself. The Wasserstein distance measures the distance between the distribution of the GAN-generated data and some real-world validation dataset. We are not only interested in the distance but also in finding a set of hyper-parameters that can train a GAN outputting data with similar statistical properties to the real-world financial time series (validation set). To achieve this, research should be done to develop an inverse financial objective function that specifically quantifies statistical properties useful to measure the quality of financial data. Tuning a GAN would still be a bilevel optimization problem but using a financial objective function to tune the GAN hyper-parameters and the Wasserstein distance to calculate the weights and biases of the neural networks. This strategy might enable more stable GAN training resulting in more realistic synthetic data.

3. **Quasi-random GANs:** Neural networks have been used to create quasi-random output directly (Hofert et al., 2021), omitting the necessity to generate a quasi-random sequence using for example Halton or Sobol. Future research into a financial GAN that directly develops quasi-random outputs could solve some of the GAN-QMC's issues. In particular, there will be no need to construct a dataset resembling the empirical distribution of the GAN, prior to running the simulation, from which to quasi-randomly sample. This might lead to a slight accuracy improvement because in that case there is no need to approximate the GAN's output by sampling billions of random numbers. More noticeably, it might lead to significant efficiency improvements considering the possibility of quasi-randomly sampling without calculating a quasi-random Sobol or Halton sequence. The downside will be a more complex architecture potentially leading to even more unstable training.

4. **GANs for American options:** Least-squares can be deployed to price American options, but requires a new fitted least-squares regression for every time-step of an option, making it computationally inefficient whenever the time-step Δt becomes small. Considering that neural networks have been used in the past to predict future stock prices (Fischer et al., 2018), it would be interesting to analyse whether the Generator of the GAN can be used to price American options directly. In this case, the least-squares regression will be replaced by the Generator of the GAN, to make future cash flow predictions for each time-step. This might increase the computational efficiency of the least-squares MC while maintaining similar levels of accuracy. It should be noted that this does assume that the GAN can indeed be improved to generate realistic financial data and thus also make realistic predictions.

5. **AI & geometric Brownian motion:** Instead of using a GAN in the geometric Brownian motion, other types of AI that can generate financial time series data can be explored. Especially AI that has more stable training and has been successful many times in the past is of interest. Analysing the performance of the same method by substituting the

normal distribution in the geometric Brownian motion with a different AI might prove the effectiveness of other AI in modeling financial assets.

6. **GAN-QMC analysis:** The GAN-QMC method has only been tested on stocks. Future research, analysing the method for different markets might lead to interesting results. The difference in stability between markets might impact the GAN's ability to generate realistic data, influencing the accuracy of the GAN-QMC. In addition, the exploration of the GAN-QMC method for exotic options could also reveal interesting insights.

   The GAN-QMC has only been compared to regular MC/QMC. This has not been a problem in this study due to the close performance of GAN-QMC to regular QMC. Since the accuracy of GAN-QMC and QMC are so similar, we could use a comparative analysis of QMC to other pricing methods to determine the performance of GAN-QMC within the literature. However, would the GAN-QMC have been significantly different, future research comparing our method to other existing methods might lead to useful insights into the GAN-QMC's abilities.

   Extending the comparative analysis to other financial instruments might show differences between GAN-QMC and regular QMC for certain instruments. Once the GAN-QMC method has been refined, by developing a better GAN, it would be useful to do a more in-depth comparative analysis between various pricing methods for different financial instruments from several markets.

7. **Interpratibility of GAN:** Neural networks are a black-box, making it difficult to understand how it comes to its output. There have already been many studies on explainable AI, including those for neural networks. Using some of these techniques to explain the behavior of the Generator of the GAN helps to understand how the synthetic financial time series are created. This could also help when making the financial objective function, because it might become clearer why the GAN struggles to recreate certain statistical properties, allowing us to steer the training process better with a tailor-made objective function.

6.4 Conclusion Recap

To conclude, in this thesis, we developed a semi-parametric pricing framework by combining a GAN into QMC. We were able to use a GAN to model the underlying asset of an option. The GAN was used instead of the normal distribution in the geometric Brownian motion to create risk-neutral price paths. The developed GAN-QMC showed similar performance to regular QMC indicating the success of using the GAN, a non-parametric method, to model the returns of stocks. Due to sampling from an empirical distribution (dataset) being quicker than from a continuous probability distribution, the GAN-QMC was more computationally efficient than regular QMC. Considering that the GAN-QMC is already as accurate as QMC while the GAN's ability to create synthetic financial data is not perfect, accuracy improvements might be made if the GAN itself can be improved further. The similar performance of GAN-QMC and QMC, while the GAN did not generate perfect synthetic financial data, indicates that we could improve the accuracy by further developing the GAN tuning strategy by for example adding a financial objective function. The most notable impact might be made by further developing a GAN that generates risk-neutral financial data, especially focusing on training stability. This shows that there is still a lot to learn and improve in this research area, as discussed in section 6.3. Overall, the GAN-QMC is an interesting and accurate option pricing method when distributional assumptions are undesirable. In addition, this research unveiled valuable theoretical insights, potential applications, and directions for future research. Financial practitioners should consider its advantages in specific scenarios while acknowledging the method's

limitations and the need for ongoing advancements of financial GANs.

Bibliography

Almgren, R. et al. (2005). "Direct Estimation of Equity Market Impact". In: *University of Pennsylvania*. DOI: https://www.cis.upenn.edu/~mkearns/finread/costestim.pdf.

Altman, D.G. and J.M. Bland (2009). "Parametric v non-parametric methods for data analysis". In: *BMJ* 338. DOI: https://doi.org/10.1136/bmj.a3167.

Amihud, Y. and H. Mendelson (1986). "Asset pricing and the bid-ask spread". In: *Journal of Financial Economics* 17.2, pp. 223–249. DOI: https://doi.org/10.1016/0304-405X(86)90065-6.

Anders, U. and O. Korn (1998). "Improving the pricing of options: a neural network approach". In: *Neural Networks and Financial Economics* 17.5-6, pp. 369–388. DOI: https://doi.org/10.1002/(SICI)1099-131X(1998090)17:5/6<369::AID-FOR702>3.0.CO;2-S.

Arjovsky, M. and L. Bottou (2017). "Towards principled methods for training generative adversarial networks". In: *Cornell university*. DOI: https://arxiv.org/pdf/1701.04862.

Bai, Y. (2022). "ReLU-Function and derived function review". In: *SHS Web of Conferences* 144.02006. DOI: https://doi.org/10.1051/shsconf/202214402006.

Bartram, S.M. (2019). "Corporate hedging and speculation with derivatives". In: *Journal of Corporate Finance* 57, pp. 9–34. DOI: https://doi.org/10.1016/j.jcorpfin.2017.09.023.

BHP (2022). In: *BHP Group Limited*. DOI: https://www.bhp.com/.

Bianconi, M., S. MacLachlan, and M. Sammon (2015). "Implied volatility and the risk-free rate of return in options markets". In: *The North American Journal of Economics and Finance* 31, pp. 1–26. DOI: https://doi.org/10.1016/j.najef.2014.10.003.

Bingham, N.H. and R. Kiesel (2001). "Risk-Neutral valuation: Pricing and hedging of financial derivatives". In: DOI: https://ci.nii.ac.jp/ncid/BA3750618X.

Birge, J.R. (1995). "Quasi-Monte Carlo approaches to option pricing". In: *The University of Michigan*.

Black, F. and M. Scholes (1973). "The Pricing of Options and Corporate Liabilities". In: *Journal of Political Economy* 81.3, pp. 637–654. DOI: https://www.jstor.org/stable/1831029.

Bolia, N. and S. Juneja (2005). "Monte Carlo methods for pricing financial options". In: *Sadhana-academy Proceedings in Engineering Sciences* 30.2-1, pp. 347–385. DOI: https://doi.org/10.1007/bf02706251.

Bouchaud, J.P., A. Matacz, and M. Potters (2001). "Leverage effect in financial markets: the retarded volatility model". In: *Physical review letters* 87.22. DOI: https://doi.org/10.1103/physrevlett.87.228701.

Boyle, P.P. (1977). "Options: A Monte Carlo approach". In: *Journal of Financial Economics* 4.3, pp. 323–338. DOI: https://doi.org/10.1016/0304-405X(77)90005-8.

Brennan, M.J. and E.S. Schwartz (1977). "The valuation of American put options". In: *The Journal of Finance* 32.2, pp. 449–462. DOI: https://doi.org/10.2307/2326779.

Broadie, M. and J. Detemple (1997). "Recent advances in numerical methods for pricing derivative securities". In: pp. 43–66.

Brooks, C. (2014). "Introductory Econometrics for Finance". In: DOI: https://doi.org/10.1017/cbo9781139540872.

Caflisch, R.E. (1998). "Monte Carlo and quasi-Monte Carlo methods". In: *Acta Numerica* 7, pp. 1–49. DOI: https://doi.org/10.1017/S0962492900002804.

Cao, J., J.H. Kim, and X. Li (2022). "Valuation of barrier and lookback options under hybrid CEV and stochastic volatility". In: *Auckland University of Technology*. DOI: https://doi.org/10.1016/j.matcom.2023.01.035.

Chakraborti, A. (2011). "Econophysics review: I. Empirical facts". In: *Quantitative Finance* 11.7, pp. 991–1012. DOI: https://dx.doi.org/10.1080/14697688.2010.539248.

Chaudhary, R., P. Bakhshi, and H. Gupta (2020). "Volatility in International Stock Markets: An Empirical Study during COVID-19". In: *Journal of Risk Financial Management* 13.9, p. 208. DOI: https://doi.org/10.3390/jrfm13090208.

Chen, R.R. and S.K. Yeh (2002). "Analytical upper bounds for American option prices". In: *The journal of financial and quantitative analysis* 37.1, pp. 117–135. DOI: https://doi.org/10.2307/3594997.

Chintala, S. and M. Arjovsky (2017). "Wasserstein GAN". In: *Conference on Machine learning* 70, pp. 214–223. DOI: https://doi.org/10.1145/3422622.

Christoffersen, P. and K. Jacobs (2004). "The importance of the loss function in option valuation". In: *Journal of Financial Economics* 72.2, pp. 291–318. DOI: https://doi.org/10.1016/j.jfineco.2003.02.001.

Coletta, A., M. Prata, and M. Conti (2021). "Towards Realistic Market Simulations: a Generative Adversarial Networks Approach". In: *2nd ACM International Conference on AI in Finance*. DOI: https://doi.org/10.48550/arXiv.2110.13287.

Cont, R. (2001). "Empirical properties of asset returns: stylized facts and statistical issues". In: *Quantitative finance* 1.2, pp. 223–236. DOI: https://doi.org/10.1080/713665670.

Cox, J.C. and S.A. Ross (1976). "The valuation of options for alternative stochastic processes". In: *Journal of Financial Economics* 3.1-2, pp. 145–166. DOI: https://doi.org/10.1016/0304-405x(76)90023-4.

Cox, J.C., S.A. Ross, and M. Rubinstein (1979). "Option pricing: A simplified approach". In: *Journal of Financial Economics* 7.3, pp. 229–263. DOI: https://doi.org/10.1016/0304-405X(79)90015-1.

Darskuviene, V. (2010). "Financial Markets". In: *Vytautas Magnus University: Leonardo Da Vinci programme*.

Davidson, L. and R.T. Froyen (1982). "Monetary policy and stock returns: Are stock markets efficient?" In: *Review* 64. DOI: https://doi.org/10.20955/r.64.3-12.cfc.

Deepak, K., M.S. Suraj, and T. Manoj (2014). "Proximal support vector machine based hybrid prediction models for trend forecasting in financial markets". In: *Expert systems with applications* 41.11, pp. 5227–5237. DOI: https://doi.org/10.1016/j.eswa.2014.01.032.

Ding, Y., Q. Wu, and X. Wang (2017). "An forward difference method and a comparative study of the numerical methods for American put option pricing". In.

Dion, M. and P. L'Ecuyer (2010). "American option pricing with randomized quasi-Monte Carlo simulations". In: *Proceedings of the 2010 Winter Simulation Conference*. DOI: https://doi.org/10.1109/wsc.2010.5678966.

Dogo, E.M. and O.J. Afolabi (2018). "A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks". In: *International Conference on Computational techniques, Electronics and Mechanical Systems*, pp. 92–99. DOI: https://doi.org/10.1109/CTEMS.2018.8769211.

Dongare, A.D., R.R. Kharde, and A.D. Kachare (2012). "Introduction to Artificial Neural Network". In: *International Journal of Engineering and Innovative Technology* 2.1. DOI: https://doi.org/10.48175/IJARSCT-8159.

Dumont, V., X. Ju, and J.L. Mueller (2022). "Hyperparameter optimization of generative adversarial network models for High-Energy physics simulations". In: *Research Square*. DOI: https://doi.org/10.21203/rs.3.rs-2181360/v1.

Dybvig, P.H. and S.A. Ross (1989). "Arbitrage". In: *Finance*. DOI: ISBN978-0-333-49535-3.

Eckerli, F. and J. Osterrieder (2021). "Generative Adversarial Networks in Finance: an overview". In: DOI: https://doi.org/10.48550/arXiv.2106.06364.

Exchange, Australian Securities (2023). "Single-stock derivatives - prices". In: DOI: https://www.asx.com.au/markets/trade-our-derivatives-market/derivatives-market-prices/single-stock-derivatives.

Fallah, S. and F. Mehrdoust (2019). "On the existence and uniqueness of the solution to the double Heston model equation and valuing lookback option". In: *Journal of Computational and Applied Mathematics* 350, pp. 412–422. DOI: https://doi.org/10.1016/j.cam.2018.10.045.

Fischer, T. and C. Krauss (2018). "Deep learning with long short-term memory networks for financial market predictions". In: *European Journal of Operational Research* 270.2, pp. 654–669. DOI: https://doi.org/10.1016/j.ejor.2017.11.054.

Fox, B.L. (1986). "Algorithm 647: Implementation and relative efficiency of quasirandom sequence generators". In: *ACM Transactions on Mathematical Software* 12.4, p. 362. DOI: https://doi.org/10.1145/22721.356187.

Fu, M.C. and J.Q. Hu (2009). "Sensitivity Analysis for Monte Carlo Simulation of Option Pricing". In: *Probability in the Engineering and Informational Sciences* 9.3, pp. 417–446. DOI: https://doi.org/10.1017/S0269964800003958.

Gatheral, J. (2006). "The volatility surface: A practitioner's guide". In: 1. DOI: https://ci.nii.ac.jp/ncid/BA83975704.

Ghosh, B., I.K. Dutta, and A. Carlson (2020). "An empirical analysis of generative adversarial network training times with varying batch sizes". In: *IEEE Annual Ubiquitous computing, electronics Mobile communication conference*. DOI: https://doi.org/10.1109/UEMCON51285.2020.9298092.

Goldman, M.B., H.B. Sosin, and M.A. Gatto (1979). "Path Dependent Options: Buy at the low, Sell at the high". In: *the Journal of Finance* 34.5, pp. 1111–1127. DOI: https://doi.org/10.2307/2327238.

Goodfellow, I.J., J Pouget-Abadie, and M. Mirza (2014). "Generative adversarial nets". In: *Conference on Neural Information Processing Systems* 2, pp. 2672–2680. DOI: https://doi.org/10.1145/3422622.

Gradojevic, N. and D. Kukolj (2011). "Parametric option pricing: A divide-and-conquer approach". In: *Physica D Nonlinear Phenomena* 240.19, pp. 1528–1535. DOI: https://doi.org/10.1016/j.physd.2011.07.001.

Grant, D., G. Vora, and D. Weeks (1997). "Path-dependent options: Extending the Monte-Carlo simulation approach". In: *Management Science* 43.11, pp. 1589–1602. DOI: https://doi.org/10.1287/mnsc.43.11.1589.

Gulrajani, I., F. Ahmed, and M. Arjovsky (2017). "Improved Training of Wasserstein GANs". In: DOI: https://doi.org/10.48550/arXiv.1704.00028.

He, J. (2020). "Capturing Four Stylized Facts of Financial Time Series in GARCH and Stochastic Volatility Models". In: *International Conference on Economic Management and Green Development* 3. DOI: https://doi.org/10.23977/icemgd2020.051.

Heston, S., P. Christoffersen, and K. Jacobs (2009). "The shape and term structure of the index option smirk: Why multifactor stochastic volatility models work so well". In: *Management Science* 55.12, pp. 1914–1932. DOI: https://doi.org/10.1287/mnsc.1090.1065.

Hinich, M.J., P. Wild, and J. Foster (2010). "Testing for the existence of a generalized Wiener process: the case of stock prices". In: *University of Queensland, School of Economics* 408.

Hochreiter, S. (1998). "The vanishing gradient problem during learning recurrent neural nets and problems solutions". In: *International journal of uncertainty, fuzziness, and knowledge-based systems* 6.2, pp. 107–116. DOI: https://doi.org/10.1142/S0218488598000094.

Hofert, M., A. Prasad, and M. Zhu (2021). "Quasi-Random sampling for multivariate distribution via generative neural networks". In: *Journal of Computational and Graphical Statistics* 30.3, pp. 647–670. DOI: https://doi.org/10.1080/10618600.2020.1868302.

Hull, J.C. (2000). "Options, Futures and Other Derivatives". In: DOI: http://ci.nii.ac.jp/ncid/BA42935800.

Hutchinson, J.M., A.W. Lo, and T. Poggio (1994). "A Nonparametric Approach to Pricing and Hedging Derivative Securities Via Learning Networks". In: *the Journal of Finance* 49.3, pp. 851–889. DOI: https://doi.org/10.1111/j.1540-6261.1994.tb00081.x.

Ito, K. (1951). "On stochastic differential equations". In: *Memoirs of the American mathematical society* 4.

Ivascu, C.F. (2021). "Option pricing using Machine learning". In: *Expert systems with applications* 163.113799. DOI: https://doi.org/10.1016/j.eswa.2020.113799.

Jabbour, G.M. and Y.K. Liu (2011). "Option pricing and Monte Carlo simulations". In: *Journal of Business Economics Research* 3.9. DOI: https://doi.org/10.19030/jber.v3i9.2802.

Jensen, M.H., A. Johansen, and I. Simonsen (2003). "Inverse statistics in economics: The gain-loss asymmetry". In: *Physica A: Statistical mechanics and its applications* 324.1-2, pp. 338–343. DOI: https://doi.org/10.1016/s0378-4371(02)01884-8.

Joy, C., P.P. Boyle, and K.S. Tan (1996). "Quasi-Monte Carlo Methods in Numerical Finance". In: *Management Science* 42.6, pp. 926–938. DOI: https://www.jstor.org/stable/2634604.

Kansal, R., J. Duarte, and B. Orzari (2020). "Graph generative adversarial networks for sparse data generation in high energy physics". In: *Cornell University*. DOI: http://export.arxiv.org/pdf/2012.00173.

Kendall, M.G. and A. Bradford Hill (1953). "The analysis of economic time series part I: Prices". In: *Journal of the Royal statistical society, Series A* 116.1, pp. 11–34. DOI: https://doi.org/10.2307/2980947.

Kirkby, J.L. (2019). "American and exotic option pricing with jump diffusions and other lévy processes". In: *Journal of Computational Finance* 22.3, pp. 89–148. DOI: 10.21314/JCF.2018.355.

Kodali, N., J. Abernethy, and J. Hays (2017). "On convergence and stability of GANs". In: *Computer vision and pattern recognition*. DOI: https://doi.org/10.48550/arXiv.1705.07215.

Koshiyama, A., N. Firoozye, and P. Treleaven (2020). "Generative adversarial networks for financial trading strategies fine-tuning and combination". In: *Quantitative Finance* 21.2, pp. 1–17. DOI: https://doi.org/10.1080/14697688.2020.1790635.

Kreuze, A., L. Dalla Valle, and C. Czado (2023). "Bayesian multivariate nonlinear state space copula models". In: *Journal of Computational Statistics and Data Analysis* 188.107820. DOI: https://doi.org/10.1016/j.csda.2023.107820.

Kumar, A., A. Alsadoon, and P.W.C. Prasad (2021). "Generative Adversarial Network (GAN) and Enhanced Root Mean Square Error (ERMSE): Deep Learning for Stock Price Movement Prediction". In: *Multimedia Tools and Applications*. DOI: https://doi.org/10.48550/arXiv.2112.03946.

Lala, S., M. Shady, and A. Belyaeva (2018). "Evaluation of mode collapse in Generative Adversarial Networks". In: *Massachusetts Institute of Technology*. DOI: https://api.semanticscholar.org/CorpusID:214622026.

Latifi, S. and N. Torres-Reyes (2019). "Audio enhancement and synthesis using generative adversarial networks: A survey". In: *International Journal of Computer Applications* 182.35, pp. 27–31. DOI: https://doi.org/10.5120/ijca2019918334.

Leiva, R. and D. Ivan (2016). "The impact of Kiyoshi Itô's stochastic calculus of financial economics". In: *Odeon* 10, pp. 157–184. DOI: https://doi.org/10.18601/17941113.n10.07.

Li, S., Z. Zhang, and P. Liu (2023). "The design of distributed filtering based on lattice rule". In: *Journal of Signal Processing* 213.109185. DOI: https://doi.org/10.1016/10.1016/j.sigpro.2023.109185.

Li, X., Y. Liu, and C. Huang (2021). "Financial time series prediction based on GGM-GAN". In: *16th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*. DOI: https://doi.org/10.1109/iske54062.2021.9755397.

Longstaff, F.A. and E.S. Schwartz (2001). "Valuing American options by simulation: A simple least-squares approach". In: *Review of Financial Studies* 14.1, pp. 113–147. DOI: https://doi.org/10.1093/rfs/14.1.113.

Lukacs, E. and H. Langer (1970). "Characteristic functions". In: *Journal of Applied Mathematics and Mechanics* 52.5, p. 317. DOI: https://doi.org/10.1002/zamm.19720520507.

Malmsten, H. and T. Teräsvirta (2010). "Stylized Facts of Financial Time Series and Three Popular Models of Volatility". In: *Special Issue on Granger Econometrics and Statistical Modeling* 3.3. DOI: https://EconPapers.repec.org/RePEc:hhs:hastef:0563.

Mancini, L. and J. Fan (2009). "Option pricing with Model-Guided nonparametric methods". In: *Journal of the American Statistical Association* 104.488, pp. 1351–1372. DOI: https://doi.org/10.1198/jasa.2009.ap08171.

Marathe, R.R. and S.M. Ryan (2005). "On the validity of the geometric Brownian motion assumption". In: *The engineering economist* 50.2, pp. 159–192. DOI: https://doi.org/10.1080/00137910590949904.

Marti, G. (2019). "CorrGAN: Sampling Realistic Financial Correlation Matrices Using Generative Adversarial Networks". In: *International Conference on Acoustics, Speech and Signal Processing*. DOI: https://doi.org/10.1109/icassp40776.2020.9053276.

McNeil, A.J., R. Frey, and P. Embrechts (2005). "Quantitative Risk Management: Concepts, techniques, and tools". In: DOI: https://people.math.ethz.ch/~embrecht/ftp/quebec.pdf.

Merton, R.C. (1974). "On the pricing of corporate debt: The risk structure of interest rates". In: *Journal of Finance* 29.2, pp. 449–470. DOI: https://doi.org/10.1111/j.1540-6261.1974.tb03058.x.

— (1976). "Option pricing when underlying stock returns are discontinuous". In: *Journal of Financial Economics* 3.1-2, p. 125. DOI: https://doi.org/10.1016/0304-405X(76)90022-2.

Metz, L., B. Poole, and D. Pfau (2017). "Unrolled generative adversarial networks". In: *International Conference on Learning Representation*. DOI: https://openreview.net/forum?id=BydrOIcle.

Mirza, M. and S. Osindero (2014). "Conditional Generative Adversarial Nets". In: *Cornell University*. DOI: https://arxiv.org/pdf/1411.1784v1.

Morokoff, W.J. and R.E. Caflisch (1995). "Quasi-Monte Carlo Integration". In: *Journal of Computational Physics* 122.2, pp. 218–230. DOI: https://doi.org/10.1006/jcph.1995.1209.

Muller, U.A., M. Dacorogna, and R.D. Dave (1997). "Volatilities of different time resolutions - Analyzing the dynamics of market components". In: *Journal of Empirical Finance* 4.2-3, pp. 213–239. DOI: https://doi.org/10.1016/s0927-5398(97)00007-8.

Myttenaere, A. de et al. (2016). "Mean absolute percentage error for regression models". In: *Neurocomputing* 192, pp. 38–48. DOI: https://doi.org/10.1016/j.neucom.2015.12.114.

Narayan, S. (1997). "The generalized sigmoid activation function: Competitive supervised learning". In: *Information Sciences* 99.1-2, pp. 69–82. DOI: https://doi.org/10.1016/S0020-0255(96)00200-9.

Needham, T. (1993). "A visual explanation of Jensen's inequality". In: *The American mathematical monthly* 100.8, pp. 768–771. DOI: https://doi.org/10.2307/2324783sc.

Niederreiter, H. (1978). "Quasi-Monte Carlo methods and pseudo-random numbers". In: *Bulletin of the American Mathematical Society* 84.6, p. 957. DOI: https://doi.org/10.1090/S0002-9904-1978-14532-7.

— (1992). "Random number generation and quasi-monte carlo methods". In: DOI: https://doi.org/10.1137/1.9781611970081.

Park, H., N. Kim, and J. Lee (2014). "Parametric models and non-parametric machine learning models for predicting option prices: Empirical comparison study over KOSPI 200 Index options". In: *Expert systems with applications* 41.11, pp. 5227–5237. DOI: https://doi.org/10.1016/j.eswa.2014.01.032.

Qiu, T., B. Zheng, and F. Ren (2006). "Return-volatility correlation in financial dynamics". In: *Physical review letters* 73.6. DOI: https://doi.org/10.1103/physreve.73.065103.

Quail, R. and J.A. Overdahl (2003). "Financial Derivatives". In: *3rd Edition*, p. 336. DOI: ISBN: 978-0-471-46766-3.

Radford, A., L. Metz, and S. Chintala (2015). "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *Cornell University*. DOI: http://export.arxiv.org/pdf/1511.06434.

Rosa, G.H. de and J.P. Papa (2022). "A survey on text generation using generative adversarial networks". In: *Pattern recognition* 119. DOI: https://doi.org/10.48550/arXiv.2212.11119.

Ross, S.A., J.E. Ingersoll, and J.C. Cox (1985). "A theory of the term structure of interest rates". In: *Econometrica* 53.2, p. 385. DOI: https://doi.org/10.1142/9789812701022_0005.

Rouah, F.D. (2013). "The Heston model and its Extensions in Matlab and C". In: *In John Wiley Sons*. DOI: https://doi.org/10.1002/9781118656471.

Rubinstein, M. (1991). "Breaking down the barriers". In: *Risk Magazine* 4, pp. 28–35.

Sæternes, E.H., A. Thune, and X. Cai (2023). "Automated parameter tuning with accuracy control for efficient reservoir simulations". In: *Simula Research Laboratory*. DOI: https://dx.doi.org/10.2139/ssrn.4520812.

Sainath, T.N., O. Vinyals, and A. Senior (2015). "Convolutional, long short-term memory, fully connected deep neural networks". In: *IEEE Transactions on Speech and Audio Processing*, pp. 4580–4584. DOI: https://doi.org/10.1109/ICASSP.2015.7178838.

Schwert, G.W. (2011). "Stock volatility during the recent financial crisis". In: *Journal of European Financial Management* 17.5, pp. 789–805. DOI: https://doi.org/10.1111/j.1468-036X.2011.00620.x.

Scott, D.W. (1994). "Multivariate density estimation: theory, practice, and visualization". In: *Journal of the American Statistical Association* 89.425. DOI: https://doi.org/10.2307/2291242.

Shreve, S.E. (2004). "Stochastic calculus for Finance II". In: *Springer Finance*. DOI: https://doi.org/10.1007/978-1-4757-4296-1.

Singarimbun, R.N., E.B. Nababan, and O.S. Sitompul (2019). "Adaptive moment estimation to minimize square error in backpropagation algorithm". In: *International Conference of Computer Science and Information Technology*. DOI: https://doi.org/10.1109/ICoSNIKOM48755.2019.9111563.

Snyder, W.C. (2000). "Accuracy estimation for quasi-Monte Carlo simulations". In: *Mathematics and Computers in Simulation* 54.1-3, pp. 131–143. DOI: https://doi.org/10.1016/S0378-4754(00)00204-4.

Sobol, I.M. (1990). "Quasi-Monte Carlo methods". In: *Progress in Nuclear Energy* 24.1-3, pp. 55–61. DOI: https://doi.org/10.1016/0149-1970(90)90022-W.

Soleymani, F. and E. Paquet (2022). "Long-term financial predictions based on Feynman-Dirac path integrals, deep Bayesian networks and temporal generative adversarial networks". In: *Machine learning with Applications* 7.100255. DOI: https://doi.org/10.1016/j.mlwa.2022.100255.

Srivastava, N., G.E. Hinton, and A. Krizhevsky (2014). "Dropout: a simple way to prevent neural networks from overfitting". In: *Journal of Machine Learning Research* 15.1, pp. 1929–1958. DOI: https://jmlr.csail.mit.edu/papers/volume15/srivastava14a/srivastava14a.pdf.

Statista (2023). "Global futures and options volume 2022". In: DOI: https://www.statista.com/statistics/377025/global-futures-and-options-volume/.

Stevens, A. (2022). "Monte-Carlo Simulation: An Introduction for Engineers and Scientists". In: *CRC Press*.

Takahasi, S., Y. Chen, and K. Tanaka-Ishii (2019). "Modeling financial time-series with generative adversarial networks". In: *Physica A* 527.121261, pp. 1–14. DOI: https://doi.org/10.1016/j.physa.2019.121261.

Taleb, N.N. (2007). "The black swan: The impact of the highly improbable". In: *Random house* 2. DOI: https://doi.org/10.5860/choice.45-1430.

Tan, X., Z. Zhang, and X. Zhao (2022). "DeepPricing: pricing convertible bonds based on financial time-series generative adversarial networks". In: *Financial Innovation* 8.1. DOI: https://doi.org/10.1186/s40854-022-00369-y.

Theodossiou, P. (2000). "Skewed generalized error distribution of financial assets and option pricing". In: *Multinational finance journal* 19.4, pp. 223–266. DOI: https://dx.doi.org/10.2139/ssrn.219679.

Vandewoestyne, B., H. Chi, and R. Cools (2010). "Computational investigations of scrambled faure sequences". In: *Mathematics and computers in Simulation* 81.3, pp. 522–535. DOI: https://doi.org/10.1016/j.matcom.2009.09.007.

Vega, J. de la (1688). "Confusion de Confusiones: Portions descriptive of the Amsterdam Stock Exchange". In: DOI: https://doi.org/10.2139/ssrn.2114849.

Venturini, A. (2022). "Climate change, risk factors and stock returns: A review of the literature". In: *International Review of Financial Analysis* 79.101934. DOI: https://doi.org/10.1016/j.irfa.2021.101934.

Vuletic, M., F. Prenzel, and M. Cucuringu (2023). "FIN-GAN: Forecasting and classifying financial time series via generative adversarial networks". In: *Social Science Research Network*. DOI: https://doi.org/10.2139/ssrn.4328302.

Wang, W. (2022). "GAN-MC: a Variance Reduction Tool for Derivatives Pricing". In: DOI: https://doi.org/10.48550/arXiv.2212.00197.

Wang, Z., Q. She, and T.E. Ward (2020). "Generative adversarial networks in computer vision: A survey and taxonomy". In: *Computer vision and pattern recognition*. DOI: https://doi.org/10.48550/arXiv.1906.01529.

Watson, G.S. (1967). "Linear least squares regression". In: *Ann. Math. Statist.* 38.6, pp. 1679–1699. DOI: https://doi.org/10.1214/aoms/1177698603.

Weng, Z.C. and F.C. Tsai (2022). "A systematic literature review of law enforcement image recognition methods based on generative adversarial networks framework". In: *Procedia Computer Science* 207, pp. 3635–3644. DOI: https://doi.org/10.1016/j.procs.2022.09.423.

Williams, J.W. (2013). "Regulatory technologies, risky subjects, and financial boundaries: Governing 'fraud' in the financial markets". In: *Accounting Organizations and Society* 38.6-7, pp. 554–558. DOI: https://doi.org/10.1016/j.aos.2012.08.001.

Winston, P.H. (1992). "Artificial Intelligence". In: *Massachusetts Institue of Technology* 3. DOI: ISBN0-201-53377-4.

Worthington, A.C. and H. Higgs (2006). "Efficiency in the Australian stock market, 1875-2006: A note on extreme long-run random walk behaviour". In: *University of Wollongong - Faculty of Business*. DOI: https://ro.uow.edu.au/cgi/viewcontent.cgi?article=1020&context=accfinwp.

Yen, Y.C. Chen (2017). "A tutorial on kernel density estimation and recent advances". In: *Biostatistics and Epidemiology* 1, pp. 161–187. DOI: https://doi.org/10.1080/24709360.2017.1396742.

Zhang, H., I. Goodfellow, and D. Metaxas (2018). "Self-Attention Generative Adversarial Networks". In: DOI: https://doi.org/10.48550/arXiv.1805.08318.

Zhang, K., G. Zhong, and J. Dong (2019). "Stock Market Prediction Based on Generative Adversarial Network". In: *Procedia Computer Science* 147, pp. 400–406. DOI: https://doi.org/10.1016/j.procs.2019.01.256.

Zhang, Y., J. Li, and H. Wang (2021). "Sentiment-guided adversarial learning for stock price prediction". In: *Mathematics of Computation and Data Science* 7. DOI: https://doi.org/10.1111/j.1540-6261.1974.tb03058.x.

Zhang, Z. and L Yang (2020). "A generative adversarial network-based method for generating negative financial samples". In: *International Journal of Distributed Sensor Networks* 16.2. DOI: https://doi.org/10.1177/1550147720907053.

Zheng, Y.J., X.H. Zhou, and W.G. Sheng (2018). "Generative adversarial network based telecom fraud detection at the receiving bank". In: *Neural networks* 102, pp. 78–86. DOI: https://doi.org/10.1016/j.neunet.2018.02.015.

Zhou, X., Z. Pan, and G. Hu (2018). "Stock market prediction on high-frequency data using generative adversarial nets". In: *Mathematical problems in engineering* 1, pp. 1–11. DOI: https://doi.org/10.1155/2018/4907423.

Zoufal, C., A. Lucchi, and S. Woerner (2019). "Quantum Generative Adversarial Networks for learning and loading random distributions". In: *npj Quantum Information* 5.103. DOI: https://doi.org/10.1038/s41534-019-0223-2.

Appendices

Appendix A

Formula's and proofs

Black-Scholes

The call price is:

$$Call = e^{-rT} E[max(S_T - K, 0)] \tag{A.1}$$

Using the previously derived results we get:

$$S_T = S_0 \, e^{(r - \frac{\sigma^2}{2})T + \sigma W_T} \tag{A.2}$$

And $(r - \frac{\sigma^2}{2})T + \sigma W_T$ is $N(0, \sigma^2 T)$ distributed. We can calculate the expectation of the payoff with these results:

$$C_0 = e^{-rT} \int_{-\infty}^{\infty} max(S_0 e^x - K, 0) e^{-\frac{(x-(r-\frac{\sigma^2}{2})T)^2}{2\sigma^2 T}} \frac{1}{\sqrt{2\pi\sigma^2 T}} dx \tag{A.3}$$

Rewriting to:

$$C_0 = e^{-rT} \int_{log(\frac{K}{S_0})}^{\infty} (S_0 e^x - K) e^{-\frac{(x-(r-\frac{\sigma^2}{2})T)^2}{2\sigma^2 T}} \frac{1}{\sqrt{2\pi\sigma^2 T}} dx \tag{A.4}$$

This integral does have a solution, which is the Black-Scholes pricing Formula. The price of a European call is:

$$C = S_0 e^{-qt} N(d_1) - K e^{-rt} N(d_2) \tag{A.5}$$

Where:

C = Call option price
S_0 = Current stock price
K = Strike price of the option
t = Time to expiration (in years)
r = Risk-free interest rate (annual)
q = Dividend yield (annual)
N = Cumulative standard normal distribution Function

$$d_1 = \frac{\ln(S_0/K) + (r - q + \frac{\sigma^2}{2})t}{\sigma\sqrt{t}}$$

$$d_2 = d_1 - \sigma\sqrt{t}$$

GAN proof

The cost function is:

$$V(D,G) = E_x[log(D(x))] + E_z[log(1 - D(G(z))] \tag{A.6}$$

$$= \int_x p_{data}(x) log(D(x)) dx + \int_z p_z(z) log(1 - D(g(z))) dz$$

$$= \int_x p_{data}(x) log(D(x)) + p_g(x) log(1 - D(x)) dx$$

If we solve this function to get the optimal Discriminator, we can see that if the probability distribution of the Generator and Discriminator are identical it always returns a value of 0.5:

$$D(x) = \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} = \frac{1}{2} \tag{A.7}$$

This makes sense considering that if the real and fake data seem identical, the Discriminator can only make a random guess, resulting in a probability of being correct of 0.5. Using this:

$$V(D,G) = E_x[log(0.5)] + E_z[log(0.5)] = -log(4) \tag{A.8}$$

So when the distributions of the Generator and Discriminator are equal the maximum of the loss function is $-log(4)$. Using equation A.7 we get:

$$V(D,G) = E_x[log(\frac{p_{data}(x)}{p_{data}(x) + p_g(x)})] + E_z[log(\frac{p_g(x)}{p_{data}(x) + p_g(x)})] \tag{A.9}$$

KL Divergence states:

$$D_{KL}(P||Q) = E_{x \sim p}\left[log\left(\frac{P(x)}{Q(x)}\right)\right] \tag{A.10}$$

Using this property and after some rewriting of the loss function:

$$V(D,G) = -log(4) + KL\left(p_{data}||\frac{p_{data} + p_g}{2}\right) + KL\left(p_g||\frac{p_{data} + p_g}{2}\right)] \tag{A.11}$$

The Jensen-Shannon Divergence (JSD) states:

$$JSD(P||Q) = \frac{1}{2} D_{KL}\left(P||\frac{P+Q}{2}\right) + \frac{1}{2} D_{KL}\left(Q||\frac{P+Q}{2}\right) \tag{A.12}$$

Which can be interpreted as a distance measure between 2 probability distributions. Using these results we get:

$$V(D,G) = -log(4) + 2JSD(p_{data}||p_g) \tag{A.13}$$

To get the optimal Generator we need the:

$$\min_G V(D,G) = -log(4) + 2JSD(p_{data}||p_g) \tag{A.14}$$

The minimum of JSD is 0 but only achieved when the probability functions are equal, e.g. $p_{data}(x) = p_g(x)$. Thus the cost function has one minimum which is only achieved if the Generator's distribution exactly matches the distribution of the Discriminator. Such that we get:

$$\min_G V(D,G) = -log(4) \tag{A.15}$$

Which we showed before to be the value if the probability distribution of the Generator and Discriminator are identical, thus proving that the minimum of the loss function can only be achieved if the probability function of the Generator matches the probability function of the real data.

Pseudo-Code GAN-QMC

Algorithm 2 Pseudo code all simulations

Input: $N, \theta, \boldsymbol{s_{1,n}}, T_0, T, K, r, C, P,$ Option type, B
function LEAST-SQUARES AMC($\boldsymbol{S}, T_0, T, K, r$, option type)
    for $i = T-1, T-1, ..., T_0$ **do**
        Calculate CF_{i+1}
        Regress CF on s_i
        Predict $\hat{CF_{i+1}}$
        $CF_i = Max(\hat{CF_{i+1}}, payoff_i)$
    t_e = Time step with highest expected value
    return AM Price (discount payoff at t_e)
function AVG AMC($\boldsymbol{S}, T_0, T, K, r$, option type)
    low = discounted payoff
    $high$ = payoff at maturity
    Return $\frac{low+high}{2}$
function EUROPEAN($\boldsymbol{S}, T_0, T, K, r$, option type)
    P = discounted payoff at T
    return European Price
Generate random sequence $\boldsymbol{Z}$
for Number of samples **do**
    $\boldsymbol{S}$ = Generate Price path with $\boldsymbol{Z}$
    val = Run Function of interest using $\boldsymbol{S}$
MC $\hat{C}$ or $\hat{P} = Avg(val)$
Generate Sobol sequence $\boldsymbol{SQ}$
for Number of samples **do**
    $\boldsymbol{S}$ = Generate Price path with $\boldsymbol{SQ}$
    val = Run Function of interest using $\boldsymbol{S}$
QMC $\hat{C}$ or $\hat{P} = Avg(val)$
Calculate accuracy and efficiency metrics
Output: metrics

Appendix B

Figures

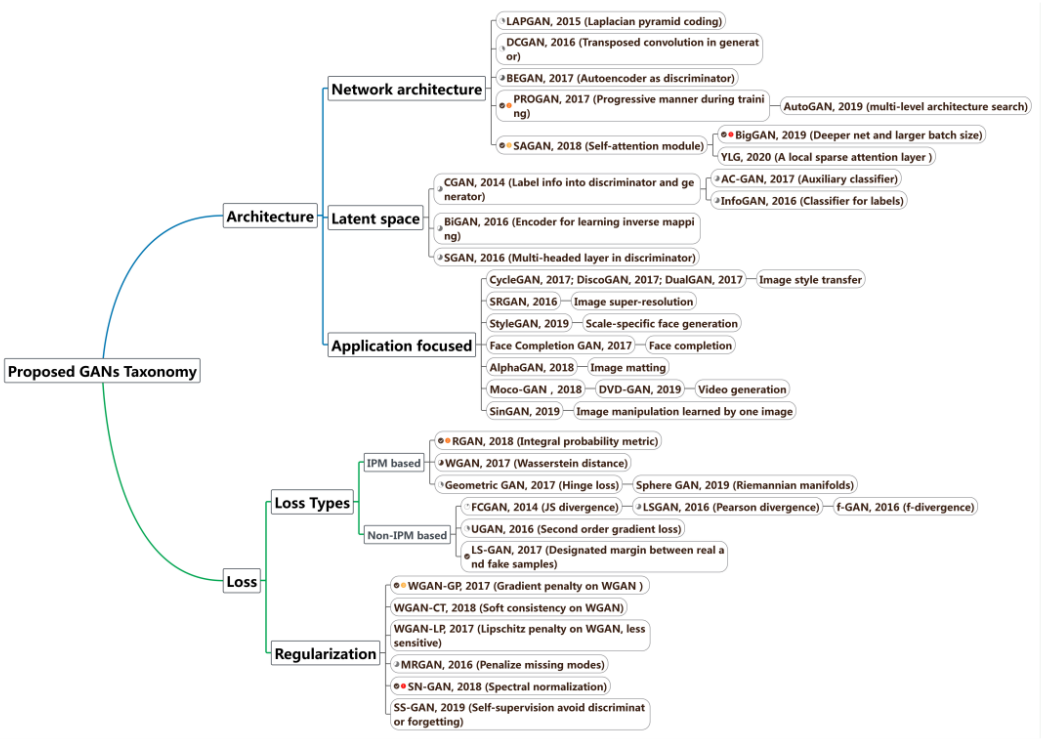

Figure B.1: GAN architectures and loss functions

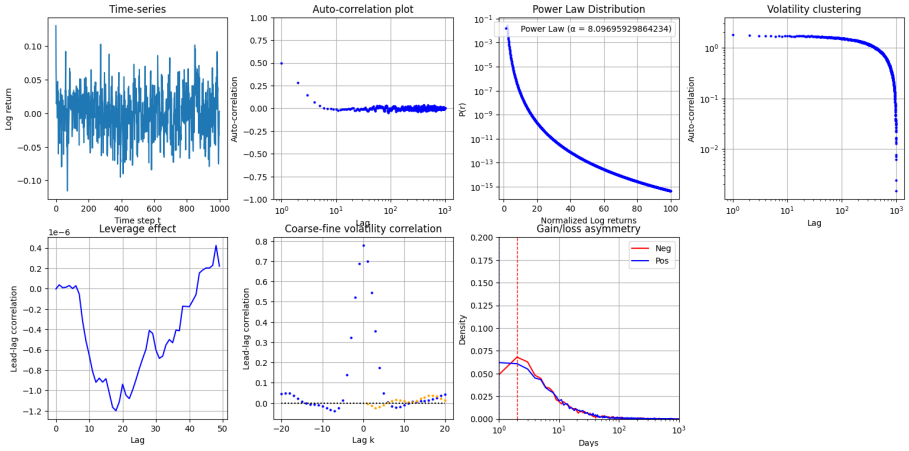

Figure B.2: Statistical properties of LSTM-GAN for CBA

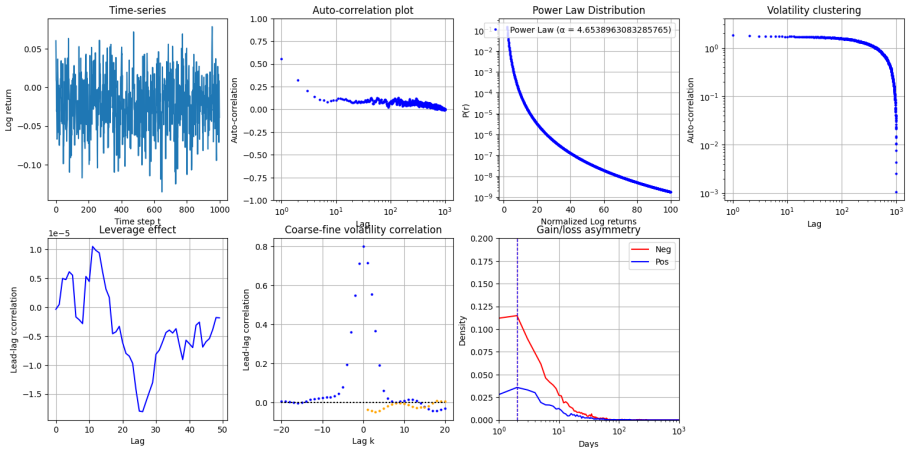

Figure B.3: Statistical properties of LSTM-TCN-GAN for CBA

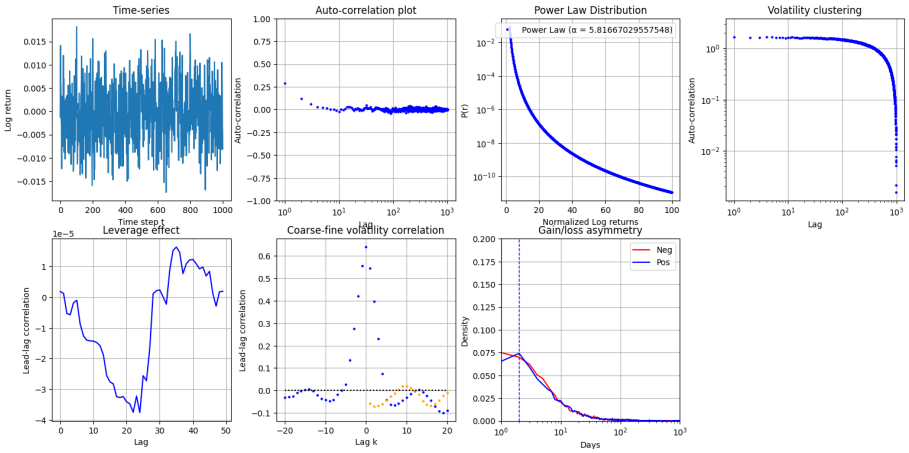

Figure B.4: Statistical properties of W-GAN for CBA

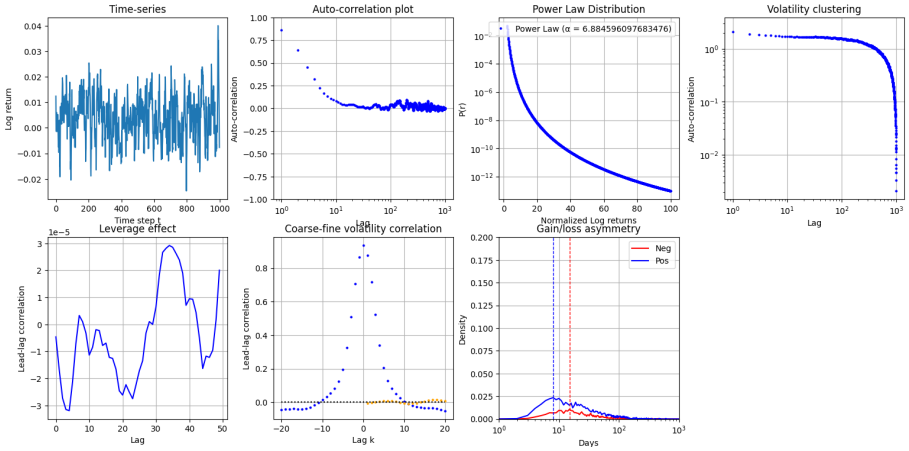

Figure B.5: Statistical properties of LSTM-GAN for CSL

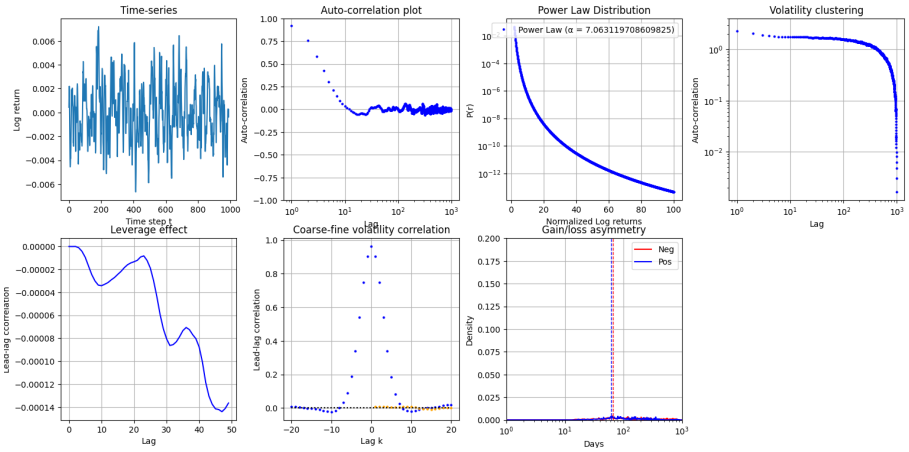

Figure B.6: Statistical properties of LSTM-TCN-GAN for CSL

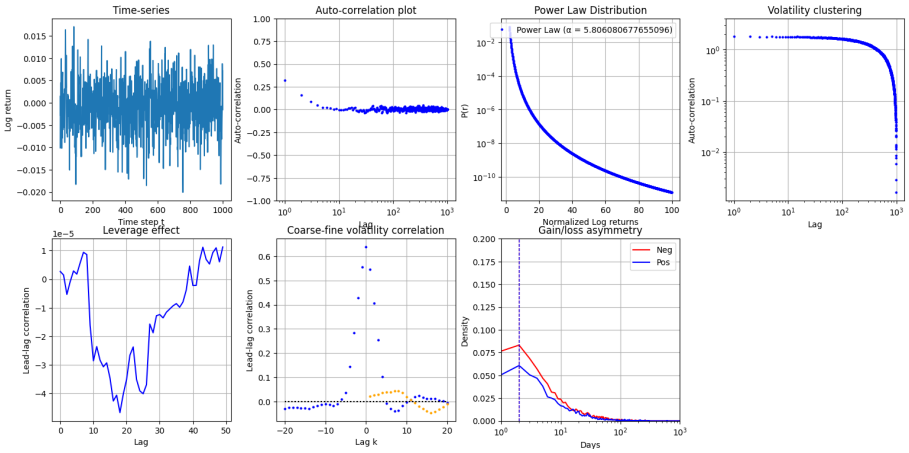

Figure B.7: Statistical properties of W-GAN for CSL

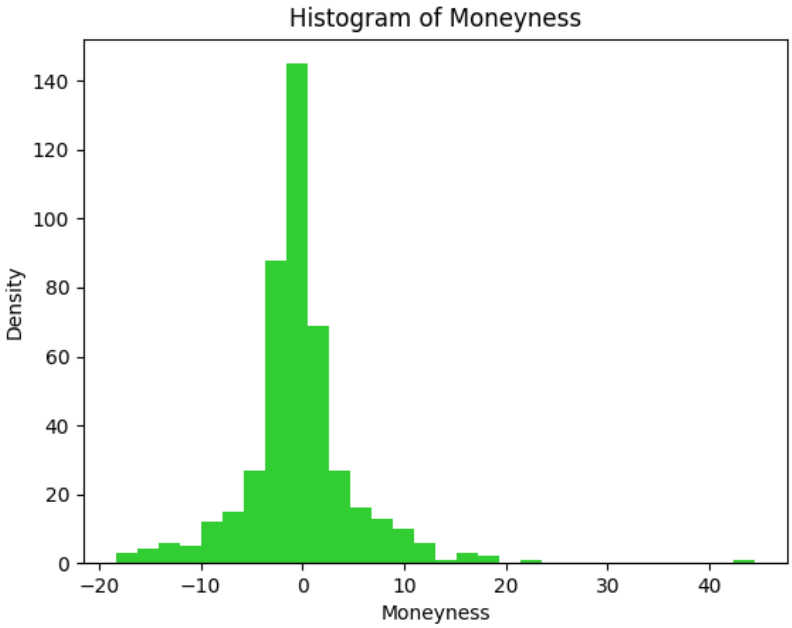

Figure B.8: Histogram of Moneyness of option-dataset

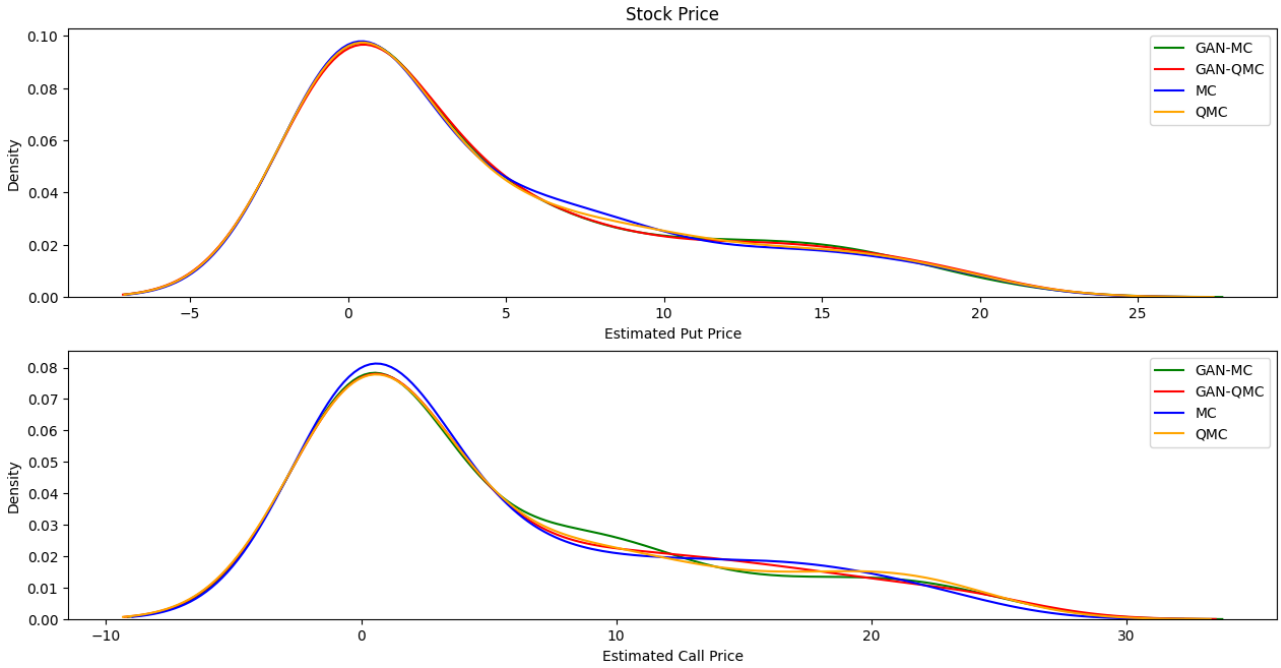

Figure B.9: Sensitivity of Stock price

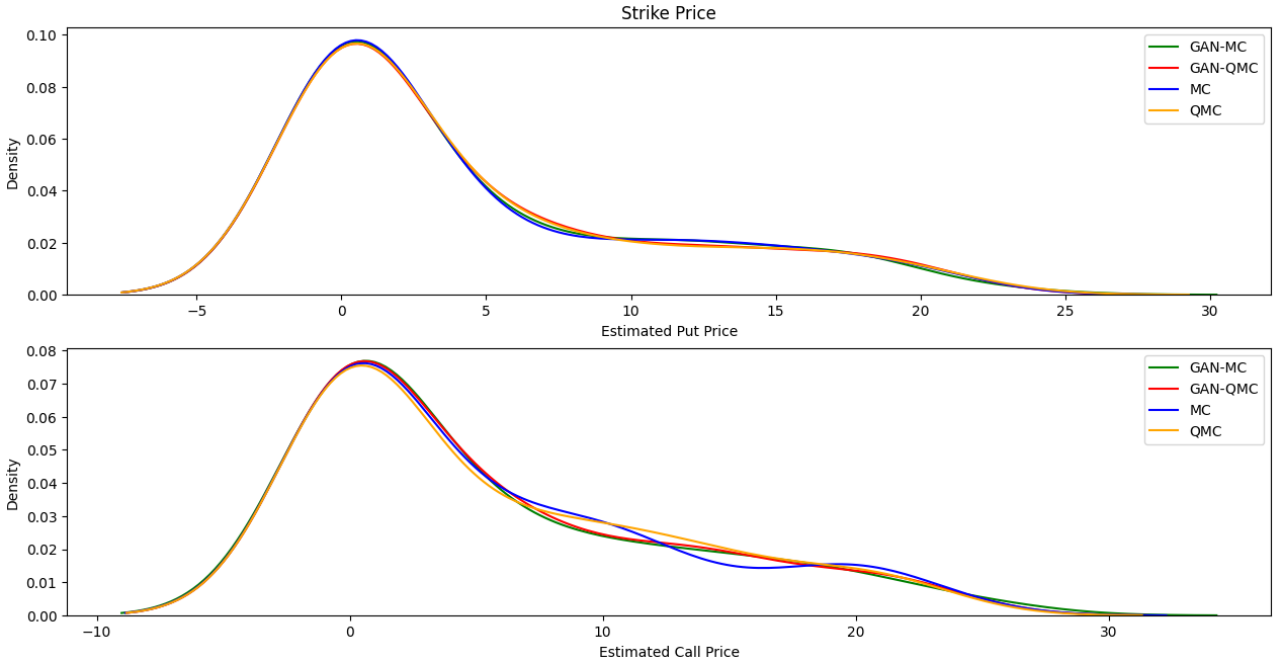

Figure B.10: Sensitivity of Strike price

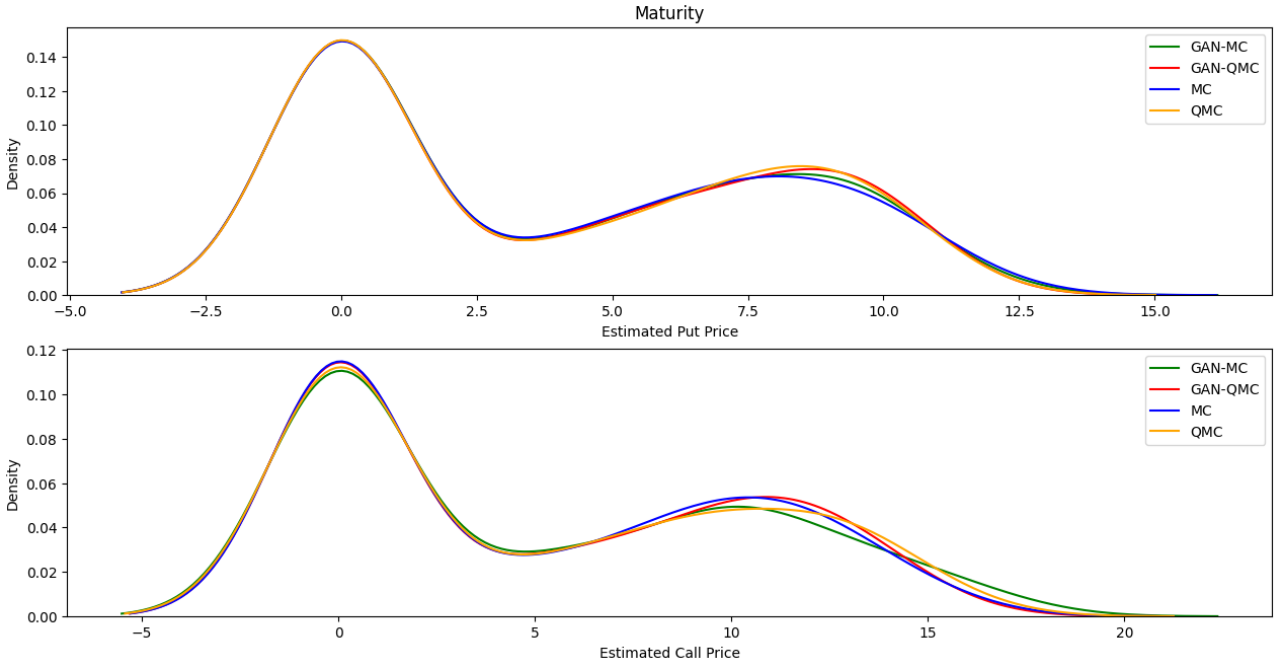

Figure B.11: Sensitivity of Time to Maturity

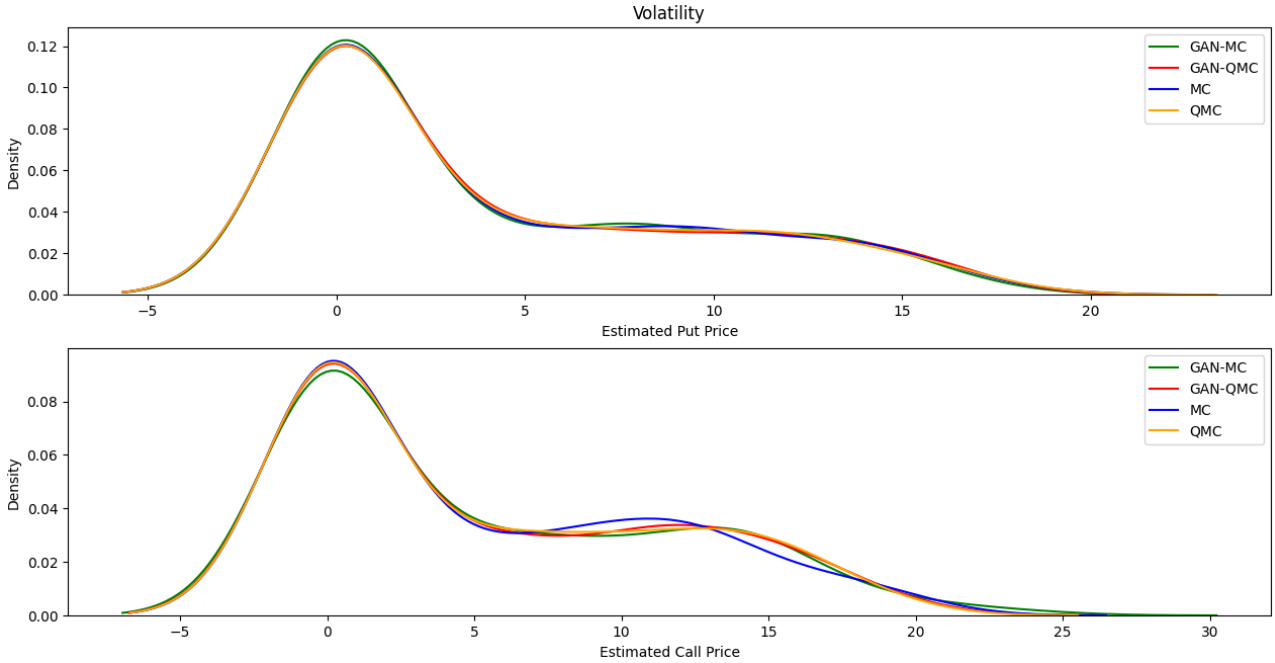

Figure B.12: Sensitivity analysis of Implied Volatility

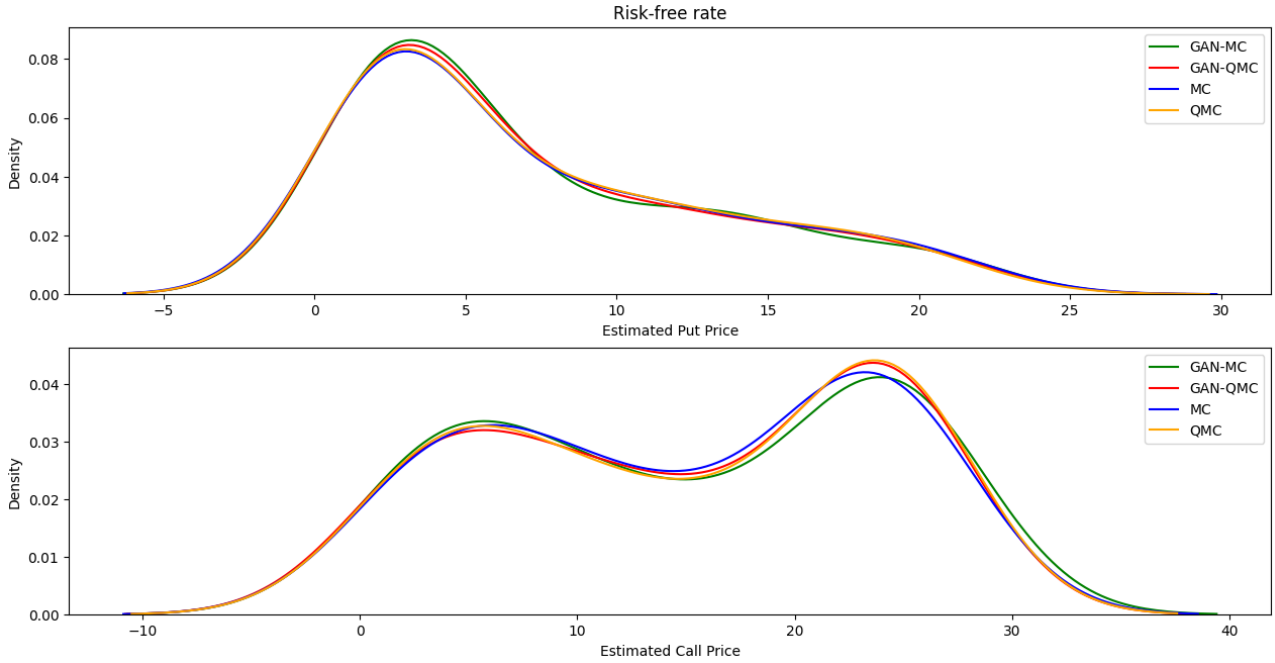

Figure B.13: Sensitivity analysis of Risk-free rate

Appendix C

Tables

Parameter	Direction	Expectation	Observed MC	Observed QMC
Strike	Decrease	Increase	Increase	Increase
Strike	Increase	Decrease	Decrease	Decrease
Time to Maturity	Decrease	Decrease	Decrease	Decrease
Time to Maturity	Increase	Decrease	Decrease	Decrease
Volatility	Decrease	Decrease	Decrease	Decrease
Volatility	Increase	Increase	Increase	Increase
Risk-free rate	Decrease	Decreases	Decreases	Decreases
Risk-free rate	Increase	Increases	Increases	Increases
Stock price	Decrease	Decrease	Decrease	Decrease
Stock price	Increase	Increase	Increase	Increase

Table C.1: Sensitivity analysis call option MC & QMC

Parameter	Direction	Expectation	Observed MC	Observed QMC
Strike	Decrease	Decrease	Decrease	Decrease
Strike	Increase	Increase	Increase	Increase
Time to Maturity	Decrease	Decrease	Decrease	Decrease
Time to Maturity	Increase	Increase	Increase	Increase
Volatility	Decrease	Decrease	Decrease	Decrease
Volatility	Increase	Increase	Increase	Increase
Risk-free rate	Decrease	Increase	Increase	Increase
Risk-free rate	Increase	Decrease	Decrease	Decrease
Stock price	Decrease	Increase	Increase	Increase
Stock price	Increase	Decrease	Decrease	Decrease

Table C.2: Sensitivity analysis put option MC & QMC

Hyper-parameter	**LSTM-GAN**	**LSTM-TCN-GAN**	**W-GAN**
Batch length	392	259	200
Batch size	25	22	10
Step size	26	47	51
Learning rate G	0.00001	0.00001	0.0000221
Learning rate D	0.0002	0.0002	0.0000696
Number of layers G	1	1	2
Number of layers D	5	1	5
Number of hidden dimensions G	62	79	50
Number of hidden dimensions D (LSTM)	151	-	-
Dropout rate D (TCN)	-	0	0
Length of input G	50	99	200
Number of pre-training epochs D	0	5	0

Table C.3: Optimal hyper-parameters BHP

Hyper-parameter	**LSTM-GAN**	**LSTM-TCN-GAN**	**W-GAN**
Batch length	79	321	256
Batch size	64	24	10
Step size	77	45	100
Learning rate G	0.00001018	0.00001008	0.00001
Learning rate D	0.00001	0.00001147	0.0002
Number of layers G	2	3	1
Number of layers D	1	1	5
Number of hidden dimensions G	76	81	192
Number of hidden dimensions D (LSTM)	68	-	-
Dropout rate D (TCN)	-	0	0
Length of input G	66	91	150
Number of pre-training epochs D	6	3	7

Table C.4: Optimal hyper-parameters CSL

Hyper-parameter	**LSTM-GAN**	**LSTM-TCN-GAN**	**W-GAN**
Batch length	303	140	70
Batch size	37	64	64
Step size	100	64	55
Learning rate G	0.00001	0.00001	0.0000996
Learning rate D	0.00001	0.00001	0.0002
Number of layers G	1	1	1
Number of layers D	3	5	5
Number of hidden dimensions G	67	97	70
Number of hidden dimensions D (LSTM)	81	-	-
Dropout rate D (TCN)	-	0	0
Length of input G	78	89	98
Number of pre-training epochs D	0	6	3

Table C.5: Optimal hyper-parameters CBA