Quantum algorithm for linear systems of equations for the multi-dimensional Black-Scholes equation

Eddie Kelly

B.Sc.

Thesis presented for the degree of

Masters by Research

to the

National University of Ireland Maynooth

Department of Physics

October 2024

Department Head

Dr. Neil Trappe

Research advisor

Prof. Jiri Vala

Acknowledgements

To my parents, Mary and Maurice and extended family, for your unwavering support and encouragement throughout the duration of this degree, without which the completion of this thesis would not have been possible. I would also like to thank Dr. Michael Dascal and Dr. Elton Zhu from Fidelity Investments, for providing me with the opportunity of an internship which ignited my interest in the subject matter of this thesis. I would like to thank Prof. Kae Nemoto for an exciting opportunity to collaborate on quantum information theory during an internship at the Okinawa Institute of Science and Technology in Japan. Finally, to my supervisor Prof. Jiri Vala, I am deeply grateful for your guidance and direction throughout the completion of this thesis and to the professional and personal opportunities you have afforded me during the course of this degree.

Abstract

The primary focus of this thesis is the investigation of the quantum algorithm for linear systems of equations (HHL) for the valuation of multi-asset options, a particular type of financial instrument. Quantum computing has the possibility to revolutionize many fields that are computationally intensive, such as quantitative finance. We extend the previous works on quantum solutions to the Black-Scholes equation for option pricing and provide its proof-of-principle implementation. We transform the problem of pricing a multi-asset option into a system of linear equations and employ the quantum algorithm due to Harrow, Hassidim and Lloyd to find its solution. Certain numerical characteristics of the matrix representing the system of linear equations determine a vital role in whether computational advantage can be achieved. The central question of this thesis is whether we can perturb the matrix that is to be inverted in the direction of more favourable numerical characteristics without compromising the accuracy of the final solution in representing the present value of the multi-asset option. Through specific examples, we show that this perturbation does not compromise the accuracy of the calculated value for the option.

After an introduction to options and their underlying mathematical description, we provide a derivation for the Black-Scholes equation using stochastic calculus and its corresponding solution for the vanilla European option through the Feynman-Kac formula. We continue with the numerical methods of finite difference approximations to convert the problem into a system of linear equations. Finally, after presentation of the quantum algorithm, we proceed with numerical simulations to determine (a) whether the aforementioned perturbation can be ameliorated with modified boundary conditions and (b) whether a working end-to-end quantum algorithm for option pricing for the case of a single-asset European option maybe achieved. Our simulation provides a proof-of-principle demonstration of the quantum algorithm.

Contents

Chapter 1

Option Pricing

The primary focus of this thesis is the application of the quantum algorithm for linear systems of equation (HHL) to the valuation of multi-asset options. In financial markets, assets are broadly defined as possessions which have value in an exchange or market [15]. Assets can either be classified as tangible, intangible or financial. Natural resources, property or any other entity whose value is derived from (a) its material presence and or (b) the commercial utility of its physical characteristics falls under the tangible assets class. Conversely, intangible assets are immaterial entities whose value is not derived from their physical form such as patents, copyrights and intellectual property. Meanwhile, financial assets or financial instruments may be defined as claims against the income or wealth of a business firm, household or unit of government represented usually by a certificate, receipt, computer record file, or other legal document, and usually created by or related to the lending of money [42]. Examples include money, equities, debt securities and derivatives. Options belong to this latter class of derivatives. As the name suggests, derivatives *derive* their value from the value of other separate assets [24].

1.1 Options

Derivatives can be broadly divided into two classes, forward and future contracts and option contracts [24]. Forward contracts are an agreement between two parties to exchange an asset for a specified price at a predetermined date, whereas futures refer to a set of standardised forward contracts that are traded on an exchange. In this scenario, the seller of the asset to be exchanged is said to adopt a *short* position and the buyer is referred to as adopting a *long* position. Options however can be thought of as an asymmetric contract wherein one party, called *the holder* has the right but not the obligation to execute a mutual transaction with the secondary party, called *the writer.* More precisely, a call option between two parties prescribes the right to the holder to buy an underlying asset at a mutually-agreed upon price (*strike*) on or before a specified date (*maturity*) from the writer [24]. A put option refers to an almost identical contractual agreement however the holder has a right to sell rather than buy. It is clear that the holder of an option has an economic advantage over the writer in the respect that they can choose to execute the transaction only if it is favourable. Option pricing, broadly speaking refers to the assigning of a fair valuation that a holder should be willing to pay to the writer for this luxury.

1.1.1 Single-asset options

The first division of option types are whether their value is contingent upon a single-asset or multiple asset. We first focus on the simpler and more traded case of single-asset options. The two most well known and traded option styles are European and American options. European options allow execution of a potential transaction between the two parties only at maturity. American options allows execution of a possible transaction anytime between commencement of the option and maturity. The payoff for either option style at execution are given as:

$$\text{Payoff}_{\substack{\text{Euro.}\\\text{put}}} = \max\left(K - S(T), 0\right) \qquad \text{Payoff}_{\substack{\text{Euro.}\\\text{call}}} = \max\left(S(T) - K, 0\right)$$

Here, the variable $S(T)$ refers to the value of the underlying asset at maturity, K refers to the strike price. Payoff refers to the overall profit the holder will receive if the transaction is executed if the underlying asset's price is $S(T)$. Strictly speaking, if we consider the European call option, the holder will receive a return of $S(T) - K$ at maturity as they can purchase the underlying asset at K and immediately resell the asset to the market at the market price $S(T)$. However, the holder will only follow through with the transaction if the asset's market price is greater than the strike price at which they can buy it. Otherwise, they would be voluntarily paying more than the market price for the asset. In the latter case, the presence of the $\max(\_\_, 0)$ in the payoff function reflects the fact that the holder will only execute the trade if financially favourable to do so. As American options can be executed at any stage during the lifetime of the option, the theoretical payoff functions for a put and call option are:

$$\text{Payoff}_{\substack{\text{Amer.}\\\text{put}}} = \max_{t\in[0,T]} (K - S(t), 0) \qquad \text{Payoff}_{\substack{\text{Amer.}\\\text{call}}} = \max_{t\in[0,T]} (S(t) - K, 0)$$

Where the variable $S(t)$ refers to the value of the underlying asset at a time t. European and American options are often referred to as *vanilla* option styles whereas *exotic* option styles such as Asian or Bermudan also exist however are traded with much less frequency. For Asian options, the average performance of the underlying asset determines the value of the option. More precisely, the payoff is not determined by the underlying price at maturity but rather by the average underlying price over some pre-set period of time. Furthermore within an option of this type, there exists various mathematical definitions of "average" such as arithmetic, geometric, continuous, etc. [52] For the sake of brevity, unless stated otherwise, all payoffs below will represent the "put" option of the corresponding option style.

$$\text{Payoff}_{Asian} = \max\big(K - \bar{S}_{avg}, 0\big)$$

Bermudan options are similar to American options, however the buyer has the right only to exercise a trade at a set (always discretely spaced) number of times:

$$\text{Payoff}_{Bermudan} = \max_{t_1,\ldots t_n} (K - S_{t_i}, 0)$$

For the sake of completeness, the last two main option types within the *exotic* class are binary options and barrier options. Options are used as risk mitigation strategies in financial markets. Options can be used to hedge an investment, whereby it effectively acts as a form of insurance. For example if one has a large position in a certain stock, taking out a put option would increase in value in the event of the stock price falling, therefore mitigating potential losses. Hedging and leveraging are the two primary motivations for options as a financial instrument.

1.1.2 Multi-asset options

For the body of this work, we will primarily focus on multi-asset options, however often the same principles that govern how single-assets trade also apply to the multi-asset case. Furthermore, within the class of multi-asset options, we will focus on European style multi-asset options, that is options which have a terminal payoff function and that can only be executed at maturity [43]. One class of multi-asset options are rainbow options, so called as the payoff function depends on multiple individual assets, each one being referred to as a 'colour' of the rainbow [38]. One common subclass of options within rainbow options are Worst-of and Best-of options. Worst-of (W.O) and Best-of (B.O) options are options wherein for each case you effectively receive, at maturity, a 'vanilla' European put option on the worst and best performing asset in the basket respectively:

$$\text{Payoff}_{W.O} = \max\left(K - \min\left(\text{S}_1(T), S_2(T), ..., S_n(T)\right), 0\right)$$

$$\text{Payoff}_{B.O} = \max\left(K - \max\left(\text{S}_1(T), S_2(T), ..., S_n(T)\right), 0\right)$$

Another type of option within the subclass of rainbow options are so called 'Best of assets or cash' [38] where the payoff function is just the maximum value of the strike K versus all the values of the underlying asset at maturity:

$$\text{Payoff}_{B.o.A.o.C} = \max\left(S_1, S_2, ..., S_n, K\right)$$

For the case of basket options, the payoff is determined by a weighted average of each of the underlying assets at maturity.

$$\text{Payoff}_{Basket} = \max\left(K - \sum_{i=1}^{n} w_i S_i(T), 0\right)$$

1.2 Pricing strategies for options

The variables of option type, strike price and spot price of the underlying asset as well as maturity are not sufficient to evaluate the price of an option. Statistical parameters characterising the tendency for the underlying asset price to fluctuate in price, *volatility* along with parameters describing current market conditions such as the *risk-free interest rate* are also necessary to price an option. Volatility σ

here represents the uncertainty the market has with respect to future movements in the price of the asset and risk-free interest rate, r, or just interest rate refers to the rate of return on a near risk-less investment. The volatility of an asset can be calculated from historical data on its performance and the risk-free interest rate is often taken to be the rate of returns for US Treasury bonds [7].

1.2.1 Single-asset options

There primarily exists two methods for the case of the evaluation of single-asset European and American style options. We will first examine the Black-Scholes-Merton model or shortened to the Black-Scholes model for the evaluation of European single-asset options. We will then review the ‘binomial’ and ’trinomial’ pricing models which are applicable to both European and American options.

European Options

The Black-Scholes model for the case of single-asset European option type with vanilla style options was described in the seminal work [6]. This celebrated model was conceived by Black and Scholes under certain reasonable assumptions about market dynamics and efficiency which leads to the reformulation of the problem as the solution to a parabolic differential equation with the ‘initial’ condition just being the value of the option at maturity, i.e. just the payoff function. The Black-Scholes equation for one underlying asset is given by:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

where the variables V refers to the price of the option, S the value of the underlying asset, σ the volatility of the underlying asset and r the risk-free interest rate. An analytic solution was derived for the partial differential equation and terminal conditions associated with the European put and call options thereby enabling the rapid and efficient valuation of options. The exact solution is given by:

$$\begin{aligned} V_{\text{call}} &= \Phi(d_+)S_0 - \Phi(d_-)Ke^{-rT} \\ V_{\text{put}} &= \Phi(-d_-)Ke^{-rT} - \Phi(-d_+)S_0 \end{aligned}$$

where S_0 is the current value of the option and Φ is the error function. The variables d_+ and d_- are defined as:

$$
\begin{aligned}
d_+ &= \frac{1}{\sigma\sqrt{T}}\left(\ln\left(\frac{S_0}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)T\right) \\
d_- &= d_+ - \sigma\sqrt{T}
\end{aligned}
$$

American Options

The essential difference that affects the valuation techniques between American and European options is that American options allow early-exercise whereas European options do not. This means that there does not exist a future point in time (i.e. maturity) in which the value of the option is known explicitly. Mathematically speaking, the boundary condition has been transformed from a fixed boundary condition to a free boundary condition. This flexibilty significantly complicates evaluation methods for American options [54]. Hence, an analytic expression for the value of American call or put option cannot be derived under the assumptions of the Black-Scholes model (except in the special case of an American call with no dividends). Analytic solutions involving infinite series expansions have been found [18], however numerical techniques must be employed to approximate the expressions. The binomial model, first outlined by Cox, Ross and Rubenstein in [12] is one of the most well known pricing strategies. The idea is to discretize time into small periods and construct a binomial tree that bifurcates the tree at each time-step. With each bifurcation, the authors assume the asset price either increases or decreases by a multiplicative factor each with an associated probability. The value of the option at present is then calculated by evaluating the price of the option at the final layer of nodes and then recursively calculate the value of the options at previous time-steps given the current values. The binomial model can also be used to calculate both the value of European and American options, however it is considerably more computationally expensive than the analytic expression of the Black-Scholes model for the case of European options.

1.2.2 Multi-asset options

Options which are contingent upon two or more risky underlying assets are called multi-asset options. The dynamics of multi-asset options are governed by a multi-dimensional variant of the Black-Scholes equation. Similar to the case of single-asset options, different types of multi-asset options are distinguished by their payoff functions [29]. The multi-dimensional variant of the Black Scholes is given by:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sum_{i,j=1}^{d} \rho_{ij}\sigma_i\sigma_j S_i S_j \frac{\partial^2 V}{\partial S_i \partial S_j} + \sum_{i=1}^{d} r S_i \frac{\partial V}{\partial S_i} - rV = 0 \tag{1.1}$$

where the variables d refers to the number of underlying assets comprising the option, S_i is the value of the i^{th} underlying asset comprising the option, ρ_{ij} is the correlation between the underlying assets S_i and S_j, σ_i is the volatility of the asset S_i, r is the risk-free interest rate and V is the value of the option. The quantities $(\hat{\sigma})_{ij} = \rho_{ij}\sigma_i\sigma_j$ form the entries of the covariance matrix that jointly governs the evolution of the prices of all of the underlying assets that govern the prices. It can be shown [29] that for non-dividend paying options that the solution to (1.1) subject to some terminal pay-off condition:

$$V(T, S_1, S_2, ..., S_d) = P(S_1, S_2, ..., S_d)$$

has the following analytic solution for the price of the option V_0:

$$V_0 = \frac{e^{-rT}}{(2\pi T)^{\frac{d}{2}}\,|\det(\hat{\sigma})|^{\frac{1}{2}}} \int_0^\infty ... \int_0^\infty \frac{P(\eta_1, \eta_2, ..., \eta_d)}{\prod_{k=1}^{d} \eta_k} \exp\left(-\frac{\vec{\alpha}^T(\hat{\sigma})^{-1}\vec{\alpha}}{2T}\right) d\eta_1 ... d\eta_d \tag{1.2}$$

where the components of $\vec{\alpha}$ are defined as:

$$(\vec{\alpha})_i = \ln\left(\frac{S_i}{\eta_i}\right) + \left(r - \frac{\sigma_i^2}{2}\right) T$$

As highlighted in [29], this integral in equation (1.2) is notoriously difficult to evaluate numerically with singularities in the integrand at the boundary of the integration region $I = [0, \infty)^d$. However, for special types of multi-asset options, closed form solutions exist, for example Worst-of and Best-of options [30]. Another such example is an European exchange option involving two assets where the payoff function is defined as:

$$V(T, S_1, S_2) = \max\left(S_1(T) - S_2(T), 0\right)$$

The price of the option in this instance is a modified Black-Scholes model however with a new volatility parameter $\tilde{\sigma} = \sqrt{\sigma_1^2 + \sigma_2^2 - 2\rho\sigma_1\sigma_2}$ and risk-free interest parameter set to zero. Under the assumption of non-dividend paying options, the present value of the option can be expressed as:

$$V_{\text{Exchange}} = S_1(0)\Phi(d_1) - S_2(0)\Phi(d_2)$$

where d_1 and d_2 are defined as:

$$d_1 = \frac{\ln\left(\frac{S_1(0)}{S_2(0)}\right) + \frac{\tilde{\sigma}^2 T}{2}}{\tilde{\sigma}\sqrt{T}} \qquad d_2 = d_1 - \sigma\sqrt{T} \tag{1.3}$$

Apart from these specific multi-asset option types, we have to resort to numerical methods for the evaluation of the remaining options within this family. These numerical techniques include but are not limited to the finite difference method, Monte Carlo methods [3], higher dimensional equivalents of the binomial method (multinomial tree methods)[46] to name but a few. Our focus is the application of the finite difference method to multi-asset options with fixed terminal payoff conditions.

Chapter 2

Introduction to Stochastic Calculus

In this chapter, we outline a mathematical framework to describe how the prices of assets evolve as well as interpreting concepts such as 'market efficiency' as a useful mathematical statement that we can utilize to price options. To characterize the dynamics of underlying asset prices, the field of stochastic calculus is required. We first review measure theory, the basis of stochastic calculus, before illustrating how the dynamics of underlying asset prices and their associated probability distributions can be captured cogently through the language of stochastic calculus. To conclude we show how the closed-form expression for the price of a European call or put option (originally derived by Black and Scholes as previously stated) can be derived from stochastic integration using the Feynman-Kac formula.

2.1 Motivation

The Black-Scholes partial differential equation is derived from a number of reasonable assumptions about market efficiency and the dynamics of the price of the underlying asset [6]. Once these assumptions have been made, the Black-Scholes equation can be derived in multiple ways. One approach assumes a so-called *delta* hedging strategy that a rational actor would follow [25]. In this strategy, the actor would hold simultaneously, in a portfolio, an option based on a given underlying asset and a certain amount of the underlying asset. By configuring this portfolio appropriately, the actor can effectively eliminate any risk associated with the portfolio, implying the value of the portfolio should accrue interest in the same manner as a near risk-less financial instrument. The second method to derive the solution to the Black-Scholes equation is the Feynman-Kac formula, which provides a link between stochastic processes and partial differential equations. The assumption about the evolution of the price of the underlying assets, namely an Ito drift process, will naturally lead to the solution of the Black-Scholes equation through the Feynman-Kac formula.

We proceed with some preliminaries with measure-theoretic approach to probability so that we can motivate what a stochastic process is and its relevant mathematical properties. We will provide a heuristic derivation of Ito's lemma for stochastic differential equations which will lead to the derivation of the Black-Scholes equation. A final substitution of the stochastic process governing asset price evolution into Feynman-Kac formula will yield the deterministic Black-Scholes equation for European single-asset options.

2.2 Measure-theoretic probability

Probability has its basis in measure theory. Randomness in probability theory is represented by a tuple of three objects $(\Omega, \mathcal{F}, \mathbb{P})$ which is called a *probability space.* Ω refers to the 'sample space' which represents all possible outcomes of some given process of interest. $\mathcal{F}$ refers to the 'event space' which broadly encapsulates all possible combinations of outcomes that could be seen. More rigorously, $\mathcal{F}$ is the Borel σ-algebra of Ω. The Borel σ-algebra of Ω is defined as the class of subsets of Ω that contains Ω and is closed under complementation and countable union of sets [2]:

$$\begin{aligned} A \in \mathcal{F} \quad &\implies A^c \in \mathcal{F} \\ A_i \in \mathcal{F} \quad \forall i \in \mathbb{N} \quad &\implies \bigcup_{i=1}^{\infty} A_i \in \mathcal{F} \end{aligned}$$

In the case that the sample space is discrete and finite, the event space reduces to being the power set of the sample space ($\mathcal{F} = \mathcal{P}(\Omega)$). Finally, $\mathbb{P} : \mathcal{F} \to [0, 1]$ is a function or *measure* that assigns a probability to each of the possible events in $\mathcal{F}$. The only requirements on $\mathbb{P}$ are:

$$\mathbb{P}(\Omega) = 1$$
$$\mathbb{P}(A \cup B) = \mathbb{P}(A) + \mathbb{P}(B) \quad \forall A, B, \in \mathcal{F}, \quad A \cap B = \emptyset$$

These requirements reflect the usual intuition that at least one outcome will be observed from the process and that the probability of mutually exclusive events is the sum of the probabilities of each event. Consider a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ and a separate *measurable space* $(E, \mathcal{E})$ which consists of some space E and a σ-algebra $\mathcal{E}$ constructed on E (The only difference from a probability space being that we have not required the existence of a measure for $\mathcal{E}$). A random variable $X : \Omega \to E$ on this probability space is a measurable function from the sample space Ω to the space E. A measurable function is defined by:

$$\text{X is measurable} \iff \text{if } A \in \mathcal{E} \implies X^{-1}(A) \in \mathcal{F}$$

Or phrased differently, the preimage of any set within the σ-algebra of E ($\mathcal{E}$) under a measurable function is an element of the σ-algebra of Ω ($\mathcal{F}$). A random variable assigns to each sample ω in the sample space Ω, an element of E and assigns to each event from the event space $\mathcal{F}$, an element of $\mathcal{E}$ ($\mathcal{E}$ being the corresponding σ algebra of E). The probability that X takes on a particular value for a measurable set B from the σ-algebra $\mathcal{E}$ is given as:

$$\mathbb{P}(X \in B) = \mathbb{P}(\{\omega \in \Omega | X(\omega) \in B\})$$

X is called a measurable function as we have managed to construct a measure on E by determining the preimage of the measurable sets in $\mathcal{E}$ in $\mathcal{F}$ and using the respective measure $\mathbb{P}$ on $\mathcal{F}$ for these sets. The expected value of a random variable $\mathbb{E}(X)$ is given by the formula:

$$\mathbb{E}(X) = \int_{\omega \in \Omega} X(\omega) d\mathbb{P}(\omega)$$

To give an example of a random variable, consider the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ corresponding to the roll of a fair die. Let Z be a random variable such that it returns 1 for an even result of the die and 0 otherwise:

$$Z : (\underbrace{\{1,2,3,4,5,6\}}_{\Omega}, \underbrace{\mathcal{P}(\{1,2,3,4,5,6\})}_{\mathcal{F}}) \to (\underbrace{\{0,1\}}_{E}, \underbrace{\{\{0\},\{1\},\{0,1\},\emptyset\}}_{\mathcal{E}})$$

$$\omega \longmapsto \begin{cases} 1 & \text{if} \quad \omega \in \{2,4,6\} \\ 0 & \text{otherwise} \end{cases}$$

where $\emptyset$ denotes the empty set. The sample $\{1\}$ is mapped to $\{0\}$ and event $\{2,3\}$ is mapped to $\{0,1\}$. The probability of the event $B = \{1\}$ is given by:

$$\mathbb{P}(Z \in \{1\}) \iff \mathbb{P}(\omega \in \{2,4,6\}) = \frac{1}{2}$$

which matches the intuition that there is 50 % chance of rolling an even number on a fair die. One question that may be asked is why the sophisticated machinery of Borel-σ algebras and existence of measures are needed. In the discrete case, these tools are somewhat redundant. However in the continuous case, for example a real-valued random variable X, it is meaningless to consider every set within the power set of $\mathbb{R}$ and to assign a probability to each 'event'. The Borel σ-algebra of $\mathbb{R}$ restricts us to consider only meaningful sets in this instance (i.e. intervals).

2.2.1 Stochastic Processes and Martingales

A stochastic process S is a collection of random variables defined on some probability space which are indexed by some parameter t from a set T ($\{X_t(\omega) : t \in T\}$). Often as stochastic processes refer to quantities that vary with time, T is chosen such that $T = [0, \infty)$. Explicitly, a stochastic process is map given by [32]:

$$S : ([0,\infty) \times \Omega, \mathcal{B}([0,\infty)) \otimes \mathcal{F}) \to (E, \mathcal{E})$$

$$(t, \omega) \mapsto X_t(\omega) \in E$$

Similarly, events from $\mathcal{B}([0,\infty)) \otimes \mathcal{F}$ are mapped to events in $\mathcal{E}$

$$\underbrace{A}_{\in \mathcal{B}([0,\infty)) \otimes \mathcal{F}} \mapsto \underbrace{B}_{\in \mathcal{E}}$$

Fixing some $\omega \in \Omega$, the map from $t \mapsto X_t(\omega)$ is defined as *realisation* or an instantiation of the stochastic process. A filtration $\{\mathcal{F}_t\}_{t\geq 0}$ is family of Borel-σ algebras such that:

$$\mathcal{F}_{t_1} \subseteq \mathcal{F}_{t_2} \subseteq \mathcal{F} \qquad \forall t_1, t_2 \in [0, \infty); t_1 < t_2$$

The increasing nature of each of the Borel-σ algebras within the family represents the increasing amount of information we have with regards to what has happened thus far in the course of the 'realisation'. A process is adapted to a filtration $\{\mathcal{F}_t\}_{t\in T}$ if each random variable $\{X_t\}$ in the process is $\mathcal{F}_t$-measurable.

$$\text{if } A \in \mathcal{E} \implies X_t^{-1}(A) \in \mathcal{F}_t$$

The final object to be defined is a martingale, which is an adapted process X for some filtration $\{\mathcal{F}_t\}_{t\in T}$ for which the following two conditions hold.

$$\mathbb{E}(X_{t_2}|\mathcal{F}_{t_1}) = X_{t_1}, \quad \mathbb{E}(|X_{t_2}|) < \infty \qquad \forall t_1, t_2 \in [0, \infty); t_1 < t_2$$

Broadly speaking, the first condition states that the best estimate for the expected value for the process at t_2 given all the information up t_1 is just what the process was valued at t_1, namely X_{t_1}. The second condition states that the absolute value of any of the constituent random processes does not blow up to infinity.

Wiener process

One of the most ubiquitous stochastic processes that occur in nature is the Wiener process (also known as Brownian motion). A Wiener process W_t is a continuous, adapted process that has the following three distinct properties:

1 $W_0 = 0$ almost surely
(almost surely (a.s.) implies $\mathbb{P}(\{X_t(\omega) : t \in T; X_0(\omega) \neq 0\}) = 0$)
(Phrased differently, the probability measure of the event of all trajectories that don't start at zero is itself zero)

2 $W_{t+a} - W_t$ for $a \geq 0$
is independent of the previous evolution of the process $\{W_{t'}\}; t' < t$.

3 $W_{t+a} - W_t \sim \mathcal{N}(0, a)$

where $\mathcal{N}(\mu, \sigma)$ refers to the normal distribution of mean μ and variance σ. A Wiener process can also be thought of as a limiting process of a collection of independent and identically distributed random variables (i.i.d) via Donsker's theorem [14]:

$$W_n(t) = \frac{1}{\sqrt{n}} \sum_{1 \leq k \leq \lfloor nt \rfloor} \zeta_i$$

$$\zeta_i \sim \mathcal{N}(0,1) \quad \forall i$$

As a Wiener process is a martingale (namely the property that the expected value of the process is just the current value of the process), and given the fact that $W_0 = 0$, we can deduce:

$$\mathbb{E}(W_t) = 0$$

The variance of a Wiener process can be calculated through:

$$W_{0+t} - \underbrace{W_0}_{0 \quad a.s.} \sim \mathcal{N}(0,t)$$

$$W_t \sim \mathcal{N}(0,t) \rightarrow \text{Var}(W_t) = t$$

The final step before we can derive Ito's lemma are the multiplication rules or *Box calculus* associated with the Wiener process:

$$dWdt = 0 \quad dW^2 = dt \quad dt^2 = 0$$

The above rules can be derived from the expressions for the expectation and variance of a Wiener process. A derivation of these rules can be found within appendix (A). We proceed how the application of the above product rules can lead to a heuristic derivation of Ito's lemma for a single process and hence for a single asset.

2.2.2 Ito's lemma - Single Process

We now proceed with Ito's lemma [27]. Ito's lemma relates the differential of some function *df* to the underlying stochastic process X_t that it is a function of (i.e. $f(X_t)$). To proceed, we first introduce the concept of an Ito drift process. An Ito drift process X_t is a stochastic process whose expected value experiences a deterministic constant drift in time. It satisfies the following stochastic differential equation (SDE):

$$dX_t = \mu dt + \sigma dW_t$$

This process is also known as arithmetic Brownian motion. However, Ito's lemma holds for more general processes such as:

$$dX_t = a(t, X)dt + b(t, X)dW_t$$

where the coefficients $a(t, X), b(t, X)$ are deterministic functions of the current time and value of the process. We shall derive Ito's lemma for this more general form of SDE. For the case of a single process X_t, consider a function $f(t, X)$ that is differentiable once with respect to t and twice differentiable with respect to X. Ito's lemma addresses how a SDE can be derived for f given the SDE for X_t. If we consider a Taylor expansion of $f(t, X)$:

$$df = \frac{\partial f}{\partial t}(dt) + \frac{1}{2}\frac{\partial^2 f}{\partial t^2}(dt)^2 + \frac{\partial f}{\partial X_t}(dX_t) + \frac{1}{2}\frac{\partial^2 f}{\partial X_t^2}(dX_t)^2 + \frac{1}{2}\frac{\partial^2 f}{\partial X_t \partial t}(dX_t dt) \quad (2.1)$$

Returning to the SDE for the process, applying the *box calculus* rules yields:

$$\begin{aligned} dX_t^2 &= (a(t, X)dt + b(t, X)dW_t)^2 \qquad (2.2) \\ &\to a(t, X)^2(dt)^2 + 2a(t, X)b(t, X)\underbrace{(dW dt)}_{0} + b(t, X)^2\underbrace{(dW_t)^2}_{dt} \\ &\to a(t, X)^2 dt^2 + b(t, X)dt \end{aligned}$$

Similarly for the term $dX_t dt$, we can see that:

$$\begin{aligned} dX_t dt &= (a(t, X)dt + b(t, X)dW_t)dt \qquad (2.3) \\ &\to a(t, X)(dt)^2 + b(t, X)\underbrace{(dW_t dt)}_{0} \\ &\to a(t, X)dt^2 \end{aligned}$$

Substituting (2.2) and (2.3) into (2.1) yields:

$$\begin{aligned} df = \frac{\partial f}{\partial t}(dt) + \frac{1}{2}\frac{\partial^2 f}{\partial t^2}(dt)^2 + \frac{\partial f}{\partial X_t}(a(t, X)dt + b(t, X)dW_t) \\ + \frac{1}{2}\frac{\partial^2 f}{\partial X_t^2}(a(t, X)^2 dt^2 + b(t, X)^2 dt) + \frac{1}{2}\frac{\partial^2 f}{\partial X_t \partial t}(a(t, X)dt^2) \end{aligned}$$

Recombining like terms yields:

$$df = \left(\frac{\partial f}{\partial t} + a(t,X)\frac{\partial f}{\partial X_t} + \frac{b(t,X)^2}{2}\frac{\partial^2 f}{\partial X_t^2}\right)dt + \left(b(t,X)\frac{\partial f}{\partial X_t}\right)dW_t$$
$$+ \left(\frac{1}{2}\frac{\partial^2 f}{\partial t^2} + \frac{a(t,X)^2}{2}\frac{\partial^2 f}{\partial X_t^2} + \frac{a(t,X)^2}{2}\frac{\partial^2 f}{\partial X_t \partial t}\right)dt^2$$

Considering only first order terms in dW_t and dt yields Ito's lemma:

$$df = \left(\frac{\partial f}{\partial t} + a(t,X)\frac{\partial f}{\partial X_t} + \frac{b(t,X)^2}{2}\frac{\partial^2 f}{\partial X_t^2}\right)dt + \left(b(t,X)\frac{\partial f}{\partial X_t}\right)dW_t$$

In order to discuss stochastic differential equations, we will need the ability to *integrate* a certain stochastic process X_t over another stochastic process, (specifically a *square-integrable* martingale W_t). The Ito integral will be represented as:

$$\int X dW$$

Following the notation of [17], if X can be represented as a *simple* process (which implies the process only changes at discrete points, not continuously):

$$X_t(\omega) = \xi_0 \mathbb{1}_0(\omega) + \sum_{i=0}^{\infty} \xi_i \mathbb{1}_{(t_i,t_{i+1}]}(t)$$

and W is some *square-integrable* martingale, then the Ito integral can be defined as:

$$\int_0^t X dW = \lim_{\max_i |t_{i+1}-t_i| \to 0} \sum_{i=0}^{\infty} \xi_i \left(W_{\min(t,t_{i+1})} - W_{\min(t,t_i)}\right) \tag{2.4}$$

The exact mathematical details of this integral and technical subtleties regarding X and W are outside the scope of this thesis. However, the one result we will use is that Ito integral of a simple process is a martingale [32].

$$\mathbb{E}\left[\int_t^{t'} X dW \middle| \mathcal{F}_s\right] = \int_t^s X dW \tag{2.5}$$

As the space of simple processes is dense in the space of all measurable, $\mathcal{F}_s$-adapted processes [32], equation (2.5) holds also for this class of processes.

Probability distribution of the underlying asset prices

One of the first applications of Ito's lemma is determining the probability distribution of possible prices for each of the underlying assets S_i at a time t. One can express the current value of an option as a conditional expectation of the probability distribution for future movements in the asset price weighted by the value of the option under said movement as indicated by [35]. As we will show later, determining the value of the option via this method is far more amenable from the perspective of a quantum algorithm and therefore we will have to be able to produce a quantum state that is proportional to said probability distribution as subroutine during the quantum algorithm. Focusing on the case of a single-asset S_t, its evolution is given by the following SDE:

$$dS_t = \underbrace{\mu S_t}_{a(t,S_t)} dt + \underbrace{\sigma S_t}_{b(t,S_t)} dW \tag{2.6}$$

This type of stochastic process is known as geometric Brownian motion. In the above SDE (2.6), W is the standard Wiener process as previously encountered. There are a number of justifications [24] for why geometric Brownian motion models the price of commodities, mainly that historical prices for stocks and other assets tend to follow the type of trajectories one would expect with geometric Brownian Motion. Additionally, geometric Brownian motion is necessarily positive-valued which obviously reflects the prices of underlying assets in an option, whereas for an arbitrary process this is not necessarily true. Our aim now is to determine S_t given its differential dS_t in (2.6). The presence of S_t on the right hand side of (2.6) prevents us from directly integrating both sides of the equation to find S_t; it is not an Ito drift process (which we could directly integrate). However, by considering the process $f(S_t)$ for a carefully chosen function f, the corresponding differential $df(S_t)$ will be equal to an Ito drift process which we can directly integrate. Once integrated, we can transform to the previous process through application of f^{-1}. For geometric Brownian motion, the function that achieves this transformation is the natural logarithm.

We can determine what the differential of $d\ln(S_t)$ with Ito's lemma(i.e. $f = \ln(S)$) to get:

$$d\ln(S_t) = \left(\frac{\partial \ln(S_t)}{\partial t} + \mu S_t \frac{\partial \ln(S_t)}{\partial S_t} + \frac{(\sigma S_t)^2}{2}\frac{\partial^2 \ln(S_t)}{\partial S_t^2}\right) dt + \left(\sigma S_t \frac{\partial \ln(S_t)}{\partial S_t}\right) dW_t \quad (2.7)$$

$$d\ln(S_t) = \left(\mu S_t\left(\frac{1}{S_t}\right) + \frac{\sigma^2 S_t^2}{2}\left(-\frac{1}{S_t^2}\right)\right) dt + \left(\sigma S_t\left(\frac{1}{S_t}\right)\right) dW_t$$

$$d\ln(S_t) = \left(\mu - \frac{\sigma^2}{2}\right) dt + \sigma dW_t$$

Integrating yields:

$$\int d\ln(S_t) = \left(\mu - \frac{\sigma^2}{2}\right)\int dt + \sigma \int dW_t \quad (2.8)$$

$$\ln(S_t) = \left(\mu - \frac{\sigma^2}{2}\right) t + \sigma W_t \quad (2.9)$$

Transforming back to the initial process through $f^{-1}(x) = \exp(x)$

$$\Rightarrow S_t = S_0 \exp\left(\left(\mu - \frac{\sigma^2}{2}\right) t + \sigma W_t\right)$$

Where S_0 is the value of the process at $t = 0$ and is a constant of integration. Returning to the Wiener process, the third property that defines said process states that the increments are normally distributed. This coupled with the facts that $\mathbb{E}(W_t) = 0$ and $\text{Var}(W_t) = t$ implies the probability density function for a Wiener process at a time t is given by:

$$W_t \sim \mathcal{N}(0, t)$$

$$\mathbb{P}(W_t \in (x, x + dx)) = \frac{1}{\sqrt{2\pi t}} \exp\left(-\frac{x^2}{2t}\right) dx$$

We now derive the probability distribution for geometric Brownian motion. If we return to the equation (2.9), the moments of the distribution of $\ln(S_t)$ can be determined as:

$$\begin{aligned}
\mathbb{E}\left(\ln(S_t)\right) &= \mathbb{E}\left(\ln(S_0) + \left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t\right) \\
&= \mathbb{E}\left(\ln(S_0)\right) + \mathbb{E}\left(\left(\mu - \frac{\sigma^2}{2}\right)t\right) + \mathbb{E}\left(\sigma W_t\right) \\
&= \ln(S_0) + \left(\mu - \frac{\sigma^2}{2}\right)t + \sigma \underbrace{\mathbb{E}(W_t)}_{=0} \rightarrow \ln(S_0) + \left(\mu - \frac{\sigma^2}{2}\right)t
\end{aligned}$$

Where the linearity of the expectation is used in (2.10). For the case of the variance, it can be calculated as:

$$\begin{aligned}
\mathrm{Var}(\ln(S_t)) &= \mathrm{Var}\left(\ln(S_0) + \left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W_t\right) \\
&= \mathrm{Var}(\sigma W_t) = \sigma^2 \underbrace{\mathrm{Var}(W_t)}_{=t} \rightarrow \sigma^2 t
\end{aligned}$$

Here we have used the translation invariance and scaling property of the variance. It can be deduced now that:

$$\ln(S_t) \sim \mathcal{N}\left(\ln(S_0) + \left(\mu - \frac{\sigma^2}{2}\right)t, \sigma^2 t\right) \tag{2.10}$$

This is the defining property of a lognormal distribution for S_t. It can be shown that the probability density function of this distribution is given by:

$$S_t \sim \text{Lognormal}\left(\ln(S_0) + \left(\mu - \frac{\sigma^2}{2}\right)t, \sigma^2 t\right)$$

$$\mathbb{P}(S_t \in (\tilde{S}, \tilde{S} + \delta\tilde{S})) \approx \frac{1}{\tilde{S}\sigma\sqrt{2\pi t}} \exp\left(-\frac{\left(\ln(\tilde{S}) - (\mu - \frac{\sigma^2}{2})t - \ln(S_0)\right)^2}{2\sigma^2 t}\right)\delta\tilde{S}$$

$$\mathbb{P}(S_t \in (\tilde{S}, \tilde{S} + \delta\tilde{S})) \approx \frac{1}{\tilde{S}\sigma\sqrt{2\pi t}} \exp\left(-\frac{\left(\ln(\frac{\tilde{S}}{S_0}) - (\mu - \frac{\sigma^2}{2})t\right)^2}{2\sigma^2 t}\right)\delta\tilde{S}$$

2.2.3 Ito's lemma - Multiple Processes

As we have now motivated the analytic solution and probability density function for the case of single quantity undergoing geometric Brownian motion, we now turn our efforts to the multi-asset case. Consider a collection of quantities $\{S_1(t), S_2(t), ..., S_d(t)\}$ each of them undergoing corresponding Brownian motions

$\{W_1(t), W_2(t), ..., W_d(t)\}$.

$$dS_i(t) = rS_i(t)dt + \sigma_i S_i(t)dW_i(t) \qquad \forall i \in I := \{1, 2, ..., d\} \tag{2.11}$$

We can see from formula (2.10) that the probability distribution for each asset is given by:

$$\ln(S_i) \sim \mathcal{N}\left(\left(r - \frac{\sigma_i^2}{2}\right), \sigma_i^2 t\right) \Leftrightarrow \ln(S_i) = \left(r - \frac{\sigma_i^2}{2}\right) t + \sigma_i^2 t W_t$$

We are now interested the joint distribution for all of these assets together. In the case of each the assets being pairwise independent, the joint probability distribution is simply the product of each of the respective probability density functions for each asset. However, the quantities $\{S_i\}_{i\in I}$ may exhibit correlations between how they jointly evolve. These possible pairwise correlations can be captured by a correlation matrix:

$$dW_i dW_j = \rho_{ij} dt$$
$$\text{Corr}(W_i, W_j) = \rho_{ij}$$

However, due to non-independence of the Brownian motions $\{W_1, W_2, ..., W_d\}$, we must effectively change to a new set of independent Brownian motions $\{Z_1, Z_2, ..., Z_3\}$. This will permit us to describe the joint probability distribution of the assets. We provide in the appendix (A.1), a derivation of this joint probability distribution. Regardless, if we define the following vector of random variables $\ln(\vec{\mathbf{S}}) := (\ln(S_1), \ln(S_2), ..., \ln(S_d))$ it can be shown that:

$$\ln(\vec{\mathbf{S}}) \sim \mathcal{N}(\vec{\boldsymbol{\mu}}, \boldsymbol{\Sigma})$$

where the corresponding joint probability $f(\vec{\mathbf{x}})$ distribution is given by:

$$f(\vec{\mathbf{x}}) = \frac{1}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}})^T \boldsymbol{\Sigma}^{-1}(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}})\right)$$

where the quantities $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ are given by:

$$\vec{\boldsymbol{\mu}} = \begin{pmatrix} \left(r - \frac{\sigma_1^2}{2}\right) t \\ \left(r - \frac{\sigma_2^2}{2}\right) t \\ \vdots \\ \left(r - \frac{\sigma_d^2}{2}\right) t \end{pmatrix} \qquad \boldsymbol{\Sigma} = \begin{pmatrix} t\sigma_1^2 & t\rho_{12}\sigma_1\sigma_2 & \dots & t\rho_{1d}\sigma_1\sigma_d \\ t\rho_{12}\sigma_1\sigma_2 & t\sigma_2^2 & & \vdots \\ \vdots & & \ddots & \\ t\rho_{1d}\sigma_1\sigma_d & \dots & & t\sigma_d^2 \end{pmatrix} \tag{2.12}$$

Or equivalently again:

$$\mathbb{P}\left(\ln(\vec{\mathbf{S}}) \in \Pi_{i=1}^d (x_i, x_i + \delta x_i)\right) \approx \frac{\exp\left(-\frac{1}{2}(\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}})^T \boldsymbol{\Sigma}^{-1} (\vec{\mathbf{x}} - \vec{\boldsymbol{\mu}})\right)}{\sqrt{(2\pi)^d \det(\boldsymbol{\Sigma})}} \left(\Pi_{i=1}^d \delta x_i\right)$$

In a similar fashion to determining the probability of density function for the asset S_t from equation (2.10), the joint distribution of the underlying assets, namely $\vec{\mathbf{S}}$ will follow a multivariate lognormal distribution:

$$\vec{\mathbf{S}} \sim \text{Lognormal}(\vec{\boldsymbol{\mu}}, \boldsymbol{\Sigma})$$
$$f(\vec{\mathbf{x}}) = \frac{1}{\left(\Pi_{i=1}^d x_i\right) \sqrt{(2\pi)^d \det(\boldsymbol{\Sigma})}} \exp\left(-\frac{1}{2}(\ln(\vec{\mathbf{x}}) - \vec{\boldsymbol{\mu}})^T \boldsymbol{\Sigma}^{-1} (\ln(\vec{\mathbf{x}}) - \vec{\boldsymbol{\mu}})\right) \tag{2.13}$$

Or equivalently again:

$$\mathbb{P}\left(\vec{\mathbf{S}} \in \Pi_{i=1}^d (x_i, x_i + \delta x_i)\right) \approx \frac{\exp\left(-\frac{1}{2}(\ln(\vec{\mathbf{x}}) - \vec{\boldsymbol{\mu}})^T \boldsymbol{\Sigma}^{-1} (\ln(\vec{\mathbf{x}}) - \vec{\boldsymbol{\mu}})\right)}{\left(\Pi_{i=1}^d x_i\right) \sqrt{(2\pi)^d \det(\boldsymbol{\Sigma})}} \left(\Pi_{i=1}^d \delta x_i\right) \tag{2.14}$$

The above probability density function will be a crucial component in estimating the present value of a multi-asset option. Broadly speaking, the quantum algorithm for linear systems for equations (HHL) will output a quantum state whose amplitudes are proportional to the value of option at every possible configuration of prices for the underlying assets at present. From a pricing perspective, we only care about the amplitude which corresponds to the actual prices of the underlying assets at present. Trying to estimate the value of this single component from the output of the quantum linear systems algorithm will take exponentially many runs of the circuit. The insight of the authors from [35] was that the price of the option at present is globally distributed if we consider possible prices for the option at some future time and then re-express the current value of the option as a conditional expectation over future prices. Here ‘globally distributed’ means that every amplitude of the quantum state collectively encodes some information about the current value of the option. Hence, as we shall discuss later on in Chapter 5, if a quantum state whose amplitudes are proportional to the distribution above can be produced, the present value of the option can be estimated far more expediently.

2.2.4 Black-Scholes Equation

Consider the following portfolio consisting of long position on one multi-asset option $V(t, S_1, S_2, ..., S_d)$ for which we want to price and a short position of varying amounts $(\Delta_i)$of the underlying assets, upon which the value of the option is dependent on:

$$\Pi = -V + \sum_{l=1}^{d} \Delta_l S_l \tag{2.15}$$

Consider the change in the value of the portfolio over a small time period:

$$d\Pi = -dV + \sum_{l=1}^{d} \Delta_l dS_l \tag{2.16}$$

To evaluate the change $d\Pi$, we first need to evaluate dV. We proceed in a similar manner to how Ito's lemma was derived for the univariate case. Consider a Taylor expansion of $V(t, S_1, S_2, ..., S_d)$ yields:

$$dV = \frac{\partial V}{\partial t}(dt) + \sum_{i=1} \left(\frac{\partial V}{\partial S_i}\right)(dS_i) + \frac{1}{2}\sum_{j,k=1}^{d} \left(\frac{\partial^2 V}{\partial S_j \partial S_k}\right)(dS_j dS_k) + ... \tag{2.17}$$

In order to proceed, the expressions dS_i and $dS_j dS_k$ need to be decomposed in terms of the differentials dt and dW_i in the same manner as the univariate case. Hence, we now utilise the *box* calculus rules for the multi-asset case:

$$dW_j dW_k = \rho_{jk} dt \quad \forall j, k; \quad dW_i dt = 0 \quad \forall i \tag{2.18}$$

Where ρ_{jk} are the components of the covariance matrix ρ as alluded to in the previous section. As well as the stochastic differential equation governing each quantity S_i:

$$dS_i = rS_i dt + \sigma_i S_i dW_i$$

Expanding $dS_j dS_k$ yields:

$$\begin{aligned}
dS_j dS_k &= (rS_j dt + \sigma_j S_j dW_j)(rS_j dt + \sigma_j S_j dW_j) \\
&= \Big(r^2 S_j S_k \underbrace{(dt)^2}_{=0} + r\sigma_k S_j S_k \underbrace{(dW_k dt)}_{=0} \\
&+ r\sigma_j S_j S_k \underbrace{(dW_j dt)}_{=0} + \sigma_j \sigma_k S_j S_k \underbrace{(dW_j dW_k)}_{=\rho_{jk} dt} \Big) \\
&= \rho_{jk} \sigma_j \sigma_k S_j S_k dt
\end{aligned}$$

Returning to equation (2.17) and substituting the above expressions yields:

$$\begin{aligned}
dV &= \frac{\partial V}{\partial t}(dt) + \sum_{i=1} \left(\frac{\partial V}{\partial S_i} \right) (rS_i dt + \sigma_i S_i dW_i) \\
&+ \frac{1}{2} \sum_{j,k=1}^{d} \left(\frac{\partial^2 V}{\partial S_j \partial S_k} \right) (\rho_{jk} \sigma_j \sigma_k S_j S_k dt) \\
&= \left(\frac{\partial V}{\partial t} + \sum_{i=1}^{d} rS_i \left(\frac{\partial V}{\partial S_i} \right) + \frac{1}{2} \sum_{j,k=1}^{d} \rho_{jk} \sigma_j \sigma_k S_j S_k \left(\frac{\partial^2 V}{\partial S_j \partial S_k} \right) \right) dt \\
&+ \sum_{i=1}^{d} \sigma_i S_i \left(\frac{\partial V}{\partial S_i} \right) dW_i
\end{aligned}$$

Returning to equation (2.16), and substituting the above expression for dV yields:

$$\begin{aligned}
d\Pi &= -\left(\frac{\partial V}{\partial t} + \sum_{i=1}^{d} rS_i \left(\frac{\partial V}{\partial S_i} \right) + \frac{1}{2} \sum_{j,k=1}^{d} \rho_{jk} \sigma_j \sigma_k S_j S_k \left(\frac{\partial^2 V}{\partial S_j \partial S_k} \right) \right) dt \\
&- \sum_{i=1}^{d} \sigma_i S_i \left(\frac{\partial V}{\partial S_i} \right) dW_i + \sum_{l=1}^{d} \Delta_l (rS_l dt + \sigma_l S_l dW_l) \\
\Rightarrow &- \left(\frac{\partial V}{\partial t} + \sum_{i=1}^{d} rS_i \left(\left(\frac{\partial V}{\partial S_i} \right) - \Delta_i \right) + \frac{1}{2} \sum_{j,k=1}^{d} \rho_{jk} \sigma_j \sigma_k S_j S_k \left(\frac{\partial^2 V}{\partial S_j \partial S_k} \right) \right) dt \\
&- \sum_{i=1}^{d} \sigma_i S_i \left(\Delta_i - \left(\frac{\partial V}{\partial S_i} \right) \right) dW_i
\end{aligned} \tag{2.19}$$

The insight of Black and Scholes in their landmark work [6] was that by holding a suitably sized short position in the underlying asset upon which the option derives its value, risk can be effectively eliminated. Mathematically speaking, with regards to the last two lines of equation (2.19), by setting $\Delta_i = \frac{\partial V}{\partial S_i} \quad \forall i$ (i.e holding $\frac{\partial V}{\partial S_i}$ sized short position in each asset S_i), the only stochastic term dW_i

will vanish. The implication is that the holder of this portfolio assumes no risk. Therefore, the portfolio should accrue interest like any other risk-less financial instrument such as U.S. treasury bonds:

$$d\Pi = r\Pi dt$$

Hence if we set $\Delta_i = \frac{\partial V}{\partial S_i}$ and the left hand side of equation (2.19) equal to $r\Pi dt$ yields:

$$\begin{aligned}
r\Pi dt &= -\left(\frac{\partial V}{\partial t} + \sum_{i=1}^{d} rS_i \left(\left(\frac{\partial V}{\partial S_i}\right) - \left(\frac{\partial V}{\partial S_i}\right)\right)\right. \\
&+ \left.\frac{1}{2}\sum_{j,k=1}^{d} \rho_{jk}\sigma_j\sigma_k S_j S_k \left(\frac{\partial^2 V}{\partial S_j \partial S_k}\right)\right) dt \\
&- \sum_{i=1}^{d} \sigma_i S_i \left(\left(\frac{\partial V}{\partial S_i}\right) - \left(\frac{\partial V}{\partial S_i}\right)\right) dW_i \\
&\Rightarrow \; r\Pi dt = -\left(\frac{\partial V}{\partial t} + \frac{1}{2}\sum_{j,k=1}^{d} \rho_{jk}\sigma_j\sigma_k S_j S_k \left(\frac{\partial^2 V}{\partial S_j \partial S_k}\right)\right) dt
\end{aligned}$$

Finally, recalling the defintion of $\Pi = -V + \sum_{i=1}^{d} \left(\frac{\partial V}{\partial S_i}\right) S_i$, we can finally arrive at the multi-asset Black-Scholes equation:

$$\begin{aligned}
-rV + \sum_{i=1}^{d} rS_i \left(\frac{\partial V}{\partial S_i}\right) &= -\left(\frac{\partial V}{\partial t} + \frac{1}{2}\sum_{j,k=1}^{d} \rho_{jk}\sigma_j\sigma_k S_j S_k \left(\frac{\partial^2 V}{\partial S_j \partial S_k}\right)\right) \\
\Longrightarrow \frac{\partial V}{\partial t} + \frac{1}{2}\sum_{j,k=1}^{d} \rho_{jk}\sigma_j\sigma_k S_j S_k \left(\frac{\partial^2 V}{\partial S_j \partial S_k}\right) &+ \sum_{i=1}^{d} rS_i \left(\frac{\partial V}{\partial S_i}\right) - rV = 0 \qquad (2.20)
\end{aligned}$$

2.2.5 Feynman-Kac - European Option

Consider the Black-Scholes equation in one dimension:

$$\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} - rV = 0 \qquad (2.21)$$

subject to the payoff condition at maturity $t = T$:

$$V(T, S) = \max(K - S, 0)$$

As the underlying value S is subject to the process $dS = rSdt + \sigma S dW_t$, we can apply the Feynmac-Kac formula [31]. The theorem can be stated [9] as:
Let $F(t,x), \mu(t,x), \sigma(t,x) : [0,T] \times \mathbb{R} \to \mathbb{R}$ and $h(x) : \mathbb{R} \to \mathbb{R}$. Consider the following PDE with terminal condition:

$$\begin{cases} \frac{\partial F}{\partial t}(t,x) + \mu(t,x)\frac{\partial F}{\partial x}(t,x) + \frac{\sigma^2(x,t)}{2}\frac{\partial^2 F}{\partial x^2}(t,x) = 0 \quad 0 \leq t \leq T \\ F(T,x) = h(x) \end{cases} \tag{2.22}$$

If x is governed by stochastic differential equation $dX_t = \mu(t, X_t)X_t dt + \sigma(t, X_t)W_t$ then the solution for any $0 \leq t \leq T$ can be expressed as:

$$F(t,x) = \mathbb{E}\left[h(X_t|\mathcal{F}_t)\right] = \mathbb{E}\left[h(X_T)|X_t = x\right])$$

Taking $F := e^{-rt}V$ maps (2.21) to (2.22). We now show a heuristic derivation of the Feynman-Kac formula as applied specifically to the Black-Scholes equation. Consider the following process wherein we introduce an auxiliary time parameter λ. By convention, we assume that present time t is zero:

$$\tilde{V}(\lambda) = e^{-r\lambda}V(\lambda, S_\lambda)$$

We now calculate the variation of the process $d\tilde{V}(\lambda)$

$$d\tilde{V}(\lambda) = d(e^{-r\lambda})V(\lambda, S_\lambda) + e^{-r\lambda}dV(\lambda, S_\lambda)$$

$$d\tilde{V}(\lambda) = (-re^{-r\lambda}d\lambda)V(\lambda, S_\lambda) + e^{-r\lambda}dV(\lambda, S_\lambda) \tag{2.23}$$

Applying Ito's lemma to the function V yields:

$$dV(\lambda, S_\lambda) = \left(\frac{\partial V}{\partial t} + rS\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2}\right)d\lambda + \left(\sigma S_\lambda \frac{\partial V}{\partial S}\right)dW_\lambda$$

As by assumption V satisfies (2.21), the first term and hence the overall variation can be written as:

$$dV(\lambda, S_\lambda) = rVd\lambda + \sigma S_\lambda \frac{\partial V}{\partial S}dW_\lambda$$

Substituting into (2.23) yields:

$$d\tilde{V}(\lambda) = -re^{-r\lambda}V d\lambda + e^{-r\lambda}\left(rV d\lambda + \sigma S_\lambda \frac{\partial V}{\partial S} dW_\lambda\right) = \sigma e^{-r\lambda} S_\lambda \frac{\partial V}{\partial S} dW_\lambda \quad (2.24)$$

Integrating (2.24):

$$\tilde{V}(t_2, S_{t_2}) - \tilde{V}(t_1, S_{t_1}) = \int_{t_1}^{t_2} \sigma e^{-r\lambda} S_\lambda \frac{\partial V}{\partial S} dW_\lambda$$

Taking expectations conditioned on the assumption the initial price of the underlying asset is $\bar{S}$:

$$\begin{aligned} \mathbb{E}\left[\tilde{V}(t_2, S_{t_2})\middle| S_{t_1} = \bar{S}\right] - \mathbb{E}\left[\tilde{V}(t_1, S_{t_1})\middle| S_{t_1} = \bar{S}\right] \\ = \mathbb{E}\left[\int_{t_1}^{t_2} \sigma e^{r\lambda} S_\lambda \frac{\partial V}{\partial S} dW_\lambda \middle| S_{t_1} = \bar{S}\right] \quad (2.25) \end{aligned}$$

If we define a secondary process $\tilde{H}(\lambda)$:

$$\tilde{H}(\lambda) := \int_{t_1}^{\lambda} \sigma e^{-r\lambda} S_\lambda \frac{\partial V}{\partial S} dW_\lambda$$

Equation (2.25) can be expressed as:

$$\mathbb{E}\left[\tilde{V}(t_2, S_{t_2})\middle| S_{t_1} = \bar{S}\right] - \mathbb{E}\left[\tilde{V}(t_1, S_{t_1})\middle| S_{t_1} = \bar{S}\right] = \mathbb{E}\left[\tilde{H}(t_2)\middle| S_{t_1} = \bar{S}\right]$$

As $\tilde{H}(\lambda)$ is a martingale (it is the Ito integral of the process $\sigma e^{-r\lambda} S_\lambda \frac{\partial V}{\partial S} dW_\lambda$) and recalling (2.5), we get:

$$\mathbb{E}\left[\tilde{H}(t_2)\middle| S_{t_1} = \bar{S}\right] = \tilde{H}(t_1) = 0$$

Finally, we see that:

$$\mathbb{E}\left[\tilde{V}(t_1, S_{t_1})\middle| S_{t_1} = \bar{S}\right] = \mathbb{E}\left[\tilde{V}(t_2, S_{t_2})\middle| S_{t_1} = \bar{S}\right]$$

$$e^{-rt_2}\mathbb{E}\left[V(t_2, S_{t_2})\middle| S_{t_1} = \bar{S}\right] = e^{-rt_1}\mathbb{E}\left[V(t_1, S_{t_1})\middle| S_{t_1} = \bar{S}\right]$$

$$\Rightarrow e^{-r(t_2 - t_1)}\mathbb{E}\left(\max(K - S_{t_2})\middle| S_{t_1} = \bar{S}\right) = V(t_1, \bar{S})$$

Hence we have shown that the value of the option at the current time t_1 and price $\bar{S}$ is simply the discounted expectation value of the payoff function given the initial price $\bar{S}$. To simplify the calculation of this expectation value, we will assume $t_1 = 0$ and $T = t_2$:

$$\text{Price of option} = V(0, \bar{S}) = e^{-rT}\mathbb{E}(\max(K - S_T)|S_0 = \bar{S})$$

To calculate the price of this option, we can re-express the conditional expectation as:

$$\begin{aligned}\mathbb{E}(\max(K - S_T)|S_0 = \bar{S}) &= \int_0^\infty \max(K - S)\mathbb{P}(S_T = S|S_0 = \bar{S})dS \\ = \int_0^K (K - S)\frac{1}{S\sigma\sqrt{2\pi T}} &\exp\left(-\frac{\left(\ln(S) - (r - \frac{\sigma^2}{2})T - \ln(S_0)\right)^2}{2\sigma^2 T}\right) dS \quad (2.26)\end{aligned}$$

$$\begin{aligned}&= K\int_0^K \frac{1}{\sigma\sqrt{2\pi T}} \exp\left(-\frac{\left(\ln(S) - (r - \frac{\sigma^2}{2})T - \ln(S_0)\right)^2}{2\sigma^2 T}\right)\left(\frac{dS}{S}\right) \\ &\quad - \int_0^K \frac{1}{\sigma\sqrt{2\pi T}} \exp\left(-\frac{\left(\ln(S) - (r - \frac{\sigma^2}{2})T - \ln(S_0)\right)^2}{2\sigma^2 T}\right) dS \quad (2.27)\end{aligned}$$

Using the variable change $Y = \ln(S)$, $(dY = dS/S)$ transforms the integrals:

$$\begin{aligned}&= K\int_{-\infty}^{\ln(K)} \frac{1}{\sigma\sqrt{2\pi T}} \exp\left(-\frac{\left(Y - (r - \frac{\sigma^2}{2})T - \ln(S_0)\right)^2}{2\sigma^2 T}\right) dY \\ &\quad - \int_{-\infty}^{\ln(K)} \frac{1}{\sigma\sqrt{2\pi T}} \exp\left(-\frac{\left(Y - (r - \frac{\sigma^2}{2})T - \ln(S_0)\right)^2}{2\sigma^2 T}\right) \exp(Y)dY \quad (2.28)\end{aligned}$$

$$= K \int_{-\infty}^{\ln(K)} \frac{1}{\sigma\sqrt{2\pi T}} \exp\left(-\frac{\left(Y - (r - \frac{\sigma^2}{2})T - \ln(S_0)\right)^2}{2\sigma^2 T} \right) dY$$

$$- \int_{-\infty}^{\ln(K)} \frac{1}{\sigma\sqrt{2\pi T}} \exp\left(Y - \frac{\left(Y - \left(r - \frac{\sigma^2}{2}\right) T - \ln(S_0)\right)^2}{2\sigma^2 T} \right) dY \quad (2.29)$$

The second variable change is $\tilde{Y} = Y - (r - \frac{\sigma^2}{2})T - \ln(S_0)$, yielding:

$$= K \int_{-\infty}^{\ln(K)-(r-\frac{\sigma^2}{2})T-\ln(S_0)} \frac{1}{\sigma\sqrt{2\pi T}} \exp\left(-\frac{\tilde{Y}^2}{2\sigma^2 T} \right) d\tilde{Y}$$

$$- \int_{-\infty}^{\ln(K)-(r-\frac{\sigma^2}{2})T-\ln(S_0)} \frac{1}{\sigma\sqrt{2\pi T}} \exp\left(\tilde{Y} - \frac{\tilde{Y}^2}{2\sigma^2 T} + (r - \frac{\sigma^2}{2})T + \ln(S_0) \right) d\tilde{Y} \quad (2.30)$$

$$= K \int_{-\infty}^{\ln(K/S_0)-(r-\frac{\sigma^2}{2})T} \frac{1}{\sigma\sqrt{2\pi T}} \exp\left(-\frac{\tilde{Y}^2}{2\sigma^2 T} \right) d\tilde{Y}$$

$$- S_0 \int_{-\infty}^{\ln(K/S_0)-(r-\frac{\sigma^2}{2})T} \frac{1}{\sigma\sqrt{2\pi T}} \exp\left(\tilde{Y} - \frac{\tilde{Y}^2}{2\sigma^2 T} + \left(r - \frac{\sigma^2}{2}\right) T \right) d\tilde{Y} \quad (2.31)$$

The third variable change is $\hat{Y} = \frac{\tilde{Y}}{\sigma\sqrt{T}}$ yielding:

$$= K \int_{-\infty}^{\frac{\ln(K/S_0)-(r-\frac{\sigma^2}{2})T}{\sigma\sqrt{T}}} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\hat{Y}^2}{2} \right) d\hat{Y}$$

$$- S_0 \int_{-\infty}^{\frac{\ln(K/S_0)-\left(r-\frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}} \frac{1}{\sqrt{2\pi}} \exp\left(\sigma\sqrt{T}\hat{Y} - \frac{\hat{Y}^2}{2} + \left(r - \frac{\sigma^2}{2}\right) T \right) d\hat{Y} \quad (2.32)$$

$$= K \int_{-\infty}^{-\frac{\ln(S_0/K)+(r-\frac{\sigma^2}{2})T}{\sigma\sqrt{T}}} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\hat{Y}^2}{2} \right) d\hat{Y}$$

$$- S_0 \int_{-\infty}^{-\frac{\ln(S_0/K)+\left(r-\frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}} \frac{1}{\sqrt{2\pi}} \exp\left(\sigma\sqrt{T}\hat{Y} - \frac{\hat{Y}^2}{2} + \left(r - \frac{\sigma^2}{2}\right) T \right) d\hat{Y} \quad (2.33)$$

Additionally, to avoid clutter, we will define the upper limit in each integral in the conventional manner as the variable:

$$d_- = \frac{\ln\left(\frac{S_0}{K}\right) + (r - \frac{\sigma^2}{2})T}{\sigma\sqrt{T}}$$

To proceed in evaluating the above integrals, we need to be able to transform the variables one last time such that both integrals are equal to the error functions of two parameters. As the first integral is already centered at zero, we can focus on completing the square in the exponential of the second integral as:

$$\begin{aligned}\sigma\sqrt{T}\hat{Y} - \frac{\hat{Y}^2}{2} \quad + \quad \left(r - \frac{\sigma^2}{2}\right)T &= -\frac{1}{2}\left(\hat{Y}^2 - 2\sigma\sqrt{T}\hat{Y} + (\sigma^2 - 2r)T\right) \\ &= -\frac{1}{2}\left(\left(\hat{Y} - \sigma\sqrt{T}\right)^2 + (\sigma^2 - 2r)T - \sigma^2 T\right) \\ &= -\frac{1}{2}\left(\hat{Y} - \sigma\sqrt{T}\right)^2 + rT\end{aligned}$$

Reinserting to equation (2.33) yields:

$$= K\int_{-\infty}^{-d_-} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\hat{Y}^2}{2}\right) d\hat{Y} - S_0 e^{rT} \int_{-\infty}^{-d_-} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\left(\hat{Y} - \sigma\sqrt{T}\right)^2}{2}\right) d\hat{Y}$$

$$= K\int_{-\infty}^{-d_-} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\hat{Y}^2}{2}\right) d\hat{Y} - S_0 e^{rT} \int_{-\infty}^{-d_- - \sigma\sqrt{T}} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{\hat{Y}^2}{2}\right) d\hat{Y} \tag{2.34}$$

Recalling the definition of the error function:

$$\Phi(x) := \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} dt$$

We can rewrite equation (2.34) now as:

$$\mathbb{E}\left[\max(K - S_T)|S_0 = \bar{S}\right] = K\Phi(-d_-) - S_0 e^{rT}\Phi(-(d_- + \sigma\sqrt{T}))$$

$$\Rightarrow \mathbb{E}\left[\max(K - S_T)|S_0 = \bar{S}\right] = K\Phi(-d_-) - S_0 e^{rT}\Phi(-d_+)$$

Recalling that the value of the option is the time discounted price of this expectation, we arrive at the final analytic formula for the price of European-put style option:

$$V_{Put} = e^{-rT} K \Phi(-d_-) - S_0 \Phi(-d_+)$$

$$d_- = \frac{\ln\left(\frac{S_0}{K}\right) + (r - \frac{\sigma^2}{2})T}{\sigma\sqrt{T}} \qquad d_+ = d_- + \sigma\sqrt{T}$$

Changing the sign in the payoff function and redoing the above calculation or alternatively using put call parity ($V_{Call} - V_{Put} = S_0 - e^{-rT}K$), one can also determine the price of a European call option as:

$$V_{Call} = S_0 \Phi(d_+) - e^{-rT} K \Phi(d_-)$$

2.3 Discussion

In this chapter, we provided a brief introduction to measure-theoretic probability and its application to option pricing. Although somewhat mathematically intensive, we were able to heuristically derive the probability distribution for a collection of assets and the associated partial differential equation that governs their evolution. We also provided our own alternative formulation of the value of an European call and put option with the Feynman-Kac formula. These two results will prove to be vital when implementing the final quantum algorithm that will evaluate the price of an option. We did not excessively focus on some technical subtleties required for a mathematically rigorous treatment of stochastic integration. We focused on deriving results that will be necessary to the next chapter which relates to the numerical techniques used to solve partial differential equations.

Chapter 3

Numerical Methods for Partial Differential Equations

In this chapter, we will examine numerical methods for solutions of partial differential equations (PDEs). As the price of an European style multi asset option is governed by the multi-dimensional Black-Scholes PDE along with some appropriate initial and boundary conditions, a robust understanding of numerical approaches to solving PDEs will be critical to evaluating options prices as well as assisting when it comes to deployment of a quantum algorithm for finite difference methods. We first illustrate a first principles approach with a review of finite differences for the case of univariate and multivariate functions. We then make our first initial connection to quantum computing by providing an explicit decomposition of the difference matrices that arise in the finite difference method in terms of a basis of Pauli operators and their superpositions, called here a pseudo-Pauli basis. This is a critical mathematical framework for engineering useful quantum circuits or unitaries within the field of quantum computation. We then proceed with a derivation of how the Black-Scholes equation can be first transformed into a linear differential equation and then further transformed into a matrix differential equation ($\dot{\vec{x}} = \mathbf{A}\vec{x} + \vec{b}$) upon suitable spatial discretization through the method of lines technique. Finally, we provide an alternative derivation from [5] of the analytic solution to the previously derived matrix differential equation with an additional relaxation about the assumption of time independence of the vector $\vec{b}$, building on the initial work by [5]. To conclude the chapter, we show how this analytic solution can be encoded as a matrix inversion problem with the newly relaxed assumption about the time independence of $\vec{b}$.

3.1 Overview

Numerical techniques surrounding the finite difference method for solutions of partial differential equations form a vast field of active research [48]. Techniques vary from the finite difference method, finite element method and volume methods, spectral methods, multi-grid methods to name but a few. For comprehensive discussion of these techniques can be found in [40] Specific terminology that relates to partial differential equations such as explicit schemes, implicit schemes, Euler's method, Runge- Kutta will be further expanded upon if encountered in the following discussion. In the Black-Scholes equation, partial derivatives up to second order are taken with respect to the 'spatial' variables, namely the price of each of the underlying assets. Therefore for solving the discretized version of the equation we need to be able to approximate these derivatives on the grid. Lets restrict ourselves initially to one dimension for easier elucidation. Consider some interval $I \subset \mathbb{R}$. This interval can be discretized into N equidistant steps with the leftmost and rightmost interval labelled by x_0 and x_N respectively and $h = \frac{x_N - x_0}{N}$. Henceforth, we will refer to the subset of gridpoints from I as Ω_I. The interior of Ω_I, (i.e $\Omega_I / \{x_0, x_N\}$) will be denoted as $\widetilde{\Omega}_I$. One immediate issue from the discretization is that derivatives cannot be evaluated on the boundaries. Hence, approximations of derivatives can only be evaluated with respect to the interior points of the grid, namely $\widetilde{\Omega}_I$ For some function $V : I \to \mathbb{R}$ we can approximate the derivative of V as the *central first order difference* on $\widetilde{\Omega}_I$ as:

$$\frac{\partial V}{\partial x}(x_i) \approx \frac{V(x_{i+1}) - V(x_{i-1})}{2h}$$

Similarly, we can compute the *central second order difference* by evaluating the derivative at two neighbouring points and quantifying the rate of change:

$$\frac{\partial^2 V}{\partial x^2}(x_i) \approx \frac{\frac{V(x_{i+1}) - V(x_i)}{h} - \frac{V(x_i) - V(x_{i-1})}{h}}{h} = \frac{V(x_{i+1}) - 2V(x_i) + V(x_{i-1})}{h^2}$$

These formulae above can be re-expressed as such in matrix representation. For the first order derivative this has representation as:

$$\begin{pmatrix} V'(x_1) \\ V'(x_2) \\ \vdots \\ \\ V'(x_{N-1}) \end{pmatrix} = \frac{1}{2h} \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ -1 & 0 & 1 & & \vdots \\ 0 & -1 & 0 & \ddots & \\ \vdots & & \ddots & \ddots & 1 \\ 0 & \dots & & -1 & 0 \end{pmatrix} \begin{pmatrix} V(x_1) \\ V(x_2) \\ \vdots \\ \\ V(x_{N-1}) \end{pmatrix}$$

Similarly, for the second order derivative, the representation is given as:

$$\begin{pmatrix} V''(x_1) \\ V''(x_2) \\ \vdots \\ \\ V''(x_{N-1}) \end{pmatrix} = \frac{1}{h^2} \begin{pmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & & \vdots \\ 0 & 1 & -2 & \ddots & \\ \vdots & & \ddots & \ddots & 1 \\ 0 & \dots & & 1 & -2 \end{pmatrix} \begin{pmatrix} V(x_1) \\ V(x_2) \\ \vdots \\ \\ V(x_{N-1}) \end{pmatrix}$$

The indexing from x_1 to x_{N-1} reflects the fact that these difference matrices can only be used to approximate the derivatives within the interior of the boundary.

3.1.1 Multi-dimensional derivative operators

We now consider a subset $I_1 \times I_2 \subset \mathbb{R}^2$. Examining the Black-Scholes equation, the spatial derivatives that need to be taken are of first and second order. Let's say for purpose of explanation, we want to examine the discretization matrix associated with taking the following derivative:

$$\frac{\partial^2 V}{\partial x_1 \partial x_2} = \frac{\partial}{\partial x_1}\left(\frac{\partial V}{\partial x_2}\right)$$

We examine how the composition of differential operators be realised in matrix notation. To proceed, we need to enumerate the gridpoints that arise from the discretization of this $I_1 \times I_2$ where:

$$\Omega_{I_1} = [x_0, x_1, ..., x_N]$$

$$\Omega_{I_2} = [y_0, y_1, ..., y_N]$$

More generally, for the d-dimensional case, we label axes by variables $x_1, x_2, ...x_d$ so that the gridpoints along each axis are enumerated as:

$$\begin{aligned}
\Omega_{I_1} &= [(x_1)_0, (x_1)_1, ..., (x_1)_N] \\
\Omega_{I_2} &= [(x_2)_0, (x_2)_1, ..., (x_2)_N] \\
&\vdots \\
\Omega_{I_d} &= [(x_d)_0, (x_d)_1, ..., (x_d)_N]
\end{aligned}$$

We assume in this example that the number of gridpoints in each direction is equal to N to simplify the analysis however this is not required. We exclude the boundary points from each interval in this enumeration (i.e. the set $\widetilde{\Omega}_{I_1} \times \widetilde{\Omega}_{I_1}$). The enumeration l from each point can be expressed from the following formula (for the 2D case)

$$((x_2)_j, (x_1)_k) \rightarrow l = (N-1)(j-1) + k$$

(i.e. $((x_2)_4, (x_1)_3)$ will correspond to the third grid point along the x_1 axis and fourth grid point along the x_2 axis which in this notation would be enumerated as : $3(N-1)+3$) For the d-dimensional case, the formula will generalise to :

$$((x_d)_{j_d}, (x_{d-1})_{j_{d-1}}, ..., (x_1)_{j_1}) \rightarrow l = \mathcal{N}(j_d, j_{d-1}, ..., j_1)$$

$$\mathcal{N}(j_d, j_{d-1}, ..., j_1) := \sum_{m=1}^{d} (N-1)^{m-1}(j_m - 1) + 1$$

The above indexing naturally leads taking the tensor product of both grids in some sense, yielding a vector of length $(N-1)^d$. For the two dimensional case, with $N+1=4$ gridpoints in each direction we see that the indexing will be:

$$\begin{aligned}
((x_2)_1, (x_1)_1) &\rightarrow 1 \\
((x_2)_1, (x_1)_2) &\rightarrow 2 \\
((x_2)_2, (x_2)_1) &\rightarrow 3 \\
((x_2)_2, (x_2)_2) &\rightarrow 4
\end{aligned}$$

We can think of this new vector as corresponding to the tensor product:

$$\begin{pmatrix}(x_2)_1\\(x_2)_2\end{pmatrix} \bigotimes \begin{pmatrix}(x_1)_1\\(x_1)_2\end{pmatrix} = \begin{pmatrix}((x_2)_1,(x_1)_1)\\((x_2)_1,(x_1)_2)\\((x_2)_2,(x_1)_1)\\((x_2)_2,(x_1)_2)\end{pmatrix}$$

The significance of this is that we can think of taking partial derivatives as tensor products of difference matrices composed together. Lets say again we have a two dimensional subset with N gridpoints in each direction and some bivariate function V defined on it. We want to approximate the partial derivative with respect to x_1 at the (p_2, p_1) location in the grid.

$$\left.\frac{\partial V}{\partial x_1}\right|_{((x_2)_{p_2},(x_1)_{p_1})} \approx \frac{V((x_2)_{p_2},(x_1)_{p_1+1}) - V((x_2)_{p_2},(x_1)_{p_1-1})}{h}$$

Now if we apply the operator $I \otimes D_1$ we get the following:

$$(I \otimes D_1) \begin{pmatrix} V((x_2)_1,(x_1)_1) \\ ... \\ V((x_2)_1,(x_1)_{N-1}) \\ V((x_2)_2,(x_1)_1) \\ ... \\ V((x_2)_2,(x_1)_{N-1}) \\ ... \end{pmatrix} = \begin{pmatrix} D_1 \begin{pmatrix} V((x_2)_1,(x_1)_1) \\ ... \\ V((x_2)_1,(x_1)_{N-1}) \end{pmatrix} \\ D_1 \begin{pmatrix} V((x_2)_2,(x_1)_1) \\ ... \\ V((x_2)_2,(x_1)_{N-1}) \end{pmatrix} \\ ... \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} \partial_{x_1} V((x_2)_1,(x_1)_1) \\ ... \\ \partial_{x_1} V((x_2)_1,(x_1)_{N-1}) \end{pmatrix} \\ \begin{pmatrix} \partial_{x_1} V((x_2)_2,(x_1)_1) \\ ... \\ \partial_{x_1} V((x_2)_2,(x_1)_{N-1}) \end{pmatrix} \\ ... \end{pmatrix}$$

The second equality arises as:

$$D_1 \begin{pmatrix} ... \\ V((x_2)_1,(x_1)_{p-1}) \\ V((x_2)_1,(x_1)_p) \\ V((x_2)_1,(x_1)_{p+1}) \\ ... \end{pmatrix} = \begin{pmatrix} ... \\ \frac{V((x_2)_1,(x_1)_p) - V((x_2)_1,(x_1)_{p-2})}{2h} \\ \frac{V((x_2)_1,(x_1)_{p+1}) - V((x_2)_1,(x_1)_{p-1})}{2h} \\ \frac{V((x_2)_1,(x_1)_{p+2}) - V((x_2)_1,(x_1)_p)}{2h} \\ ... \end{pmatrix} = \begin{pmatrix} ... \\ \partial_{x_1} V((x_2)_1,(x_1)_{p-1}) \\ \partial_{x_1} V((x_2)_1,(x_1)_p) \\ \partial_{x_1} V((x_2)_1,(x_1)_{p+1}) \\ ... \end{pmatrix}$$

Similarly, applying the operator $D_1 \otimes I$ will evaluate the partial derivative along the x_2 axis. For the N-dimensional case, applying the operator:

$$I^{\otimes d-l} \otimes D_1 \otimes I^{\otimes l-1}$$

amounts to taking the partial derivative along the x_l axis. Replacing D_1 with D_2 corresponds to evaluating the second derivative along that same axis. Taking mixed partial derivatives just amounts to composition of these matrices. Taking the mixed partial derivative $\frac{\partial^2}{\partial x_j \partial x_k}$, assuming without loss of generality that $k > j$ yields:

$$\underbrace{\left(I^{\otimes d-j} \otimes D_1 \otimes I^{\otimes j-1} I\right)}_{\partial_{x_j}} \underbrace{\left(I^{\otimes d-k} \otimes D_1 \otimes I^{\otimes k-1} I\right)}_{\partial_{x_k}}$$

$$\Rightarrow I^{\otimes d-j} D_1 \otimes I^{\otimes k-j-1} \otimes D_1 \otimes I^{\otimes k-1}$$

3.1.2 Connection to Pauli matrices

As a quantum algorithm will ultimately be used to invert and solve the yet to be derived matrix formulation of the above problem, it is prudent to see if the matrices above can be expressed in terms of the commonly used matrices in quantum computing [36], namely the Pauli matrices:

$$\sigma_0 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

We will now show that it is possible to express the matrices D_1 and D_2 as linear combinations of tensor products of Pauli operators. The general form for the above difference matrices are tridiagonal matrices, which can be expressed as:

$$A = \begin{pmatrix} b & a & 0 & \dots & 0 \\ c & b & a & & \vdots \\ 0 & c & b & \ddots & \\ \vdots & & \ddots & \ddots & a \\ 0 & \dots & & c & b \end{pmatrix}$$

In order to show how inductively we can construct these tridiagonal matrices, we first define two new matrices σ_+, σ_-:

$$\sigma_+ := \frac{1}{2}(\sigma_1 + i\sigma_2) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

$$\sigma_- := \frac{1}{2}(\sigma_1 - i\sigma_2) = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$$

To proceed with the inductive argument, we define the $N = 2^1$ dimensional tridiagonal matrix:

$$T_1 = bI + a\sigma_+ + c\sigma_- = \begin{pmatrix} b & a \\ c & b \end{pmatrix}$$

Using this as our base case, we can use this to create our 'next' matrix:

$$T_2 = I_2 \otimes T_1 + c(\sigma_- \otimes \sigma_+) + a(\sigma_+ \otimes \sigma_-)$$

$$\rightarrow \begin{pmatrix} b & a & 0 & 0 \\ c & b & 0 & 0 \\ 0 & 0 & b & a \\ 0 & 0 & c & b \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & a & 0 \\ 0 & c & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} b & a & 0 & 0 \\ c & b & a & 0 \\ 0 & c & b & a \\ 0 & 0 & c & b \end{pmatrix}$$

The recursive relationship can be defined between the matrices T_n where a *correction* term can be added each time:

$$T_{n+1} = I \otimes T_n + \underbrace{c(\sigma_- \otimes \sigma_+^{\otimes n}) + a(\sigma_+ \otimes \sigma_-^{\otimes n})}_{\text{Correction}}$$

This recursive relationship gives arise to the analytic form for the 2^n x 2^n tridiagonal matrix T_n:

$$T_n = \underbrace{\begin{pmatrix} b & a & 0 & \dots & 0 \\ c & b & a & & \vdots \\ 0 & c & b & \ddots & \\ \vdots & & \ddots & \ddots & a \\ 0 & \dots & & c & b \end{pmatrix}}_{2^n}$$

$$T_n = bI^{\otimes n} + \sum_{l=0}^{n-1} I^{\otimes n-1-l} \otimes \left(c(\sigma_- \otimes \sigma_+^{\otimes l}) + a(\sigma_+ \otimes \sigma_-^{\otimes l}\right)$$

To prove this formula, assume inductively that it is true for $n = k$:

$$T_k = bI^{\otimes k} + \sum_{l=0}^{k-1} I^{\otimes k-1-l} \otimes \left(c(\sigma_- \otimes \sigma_+^{\otimes l}) + a(\sigma_+ \otimes \sigma_-^{\otimes l}\right)$$

We will now show that the sum satisfies the recursive relationship:

$$T_{k+1} = I \otimes T_k + c(\sigma_- \otimes \sigma_+^{\otimes k}) + a(\sigma_+ \otimes \sigma_-^{\otimes k})$$

Applying the recursive definition yields:

$$\Rightarrow I \otimes \left(bI^{\otimes k} + \sum_{l=0}^{k-1} I^{\otimes k-1-l} \otimes \left(c(\sigma_- \otimes \sigma_+^{\otimes l}) + a(\sigma_+ \otimes \sigma_-^{\otimes l}\right) \right) \\ + c(\sigma_- \otimes \sigma_+^{\otimes n}) + a(\sigma_+ \otimes \sigma_-^{\otimes n}) \quad (3.1)$$

$$\Rightarrow bI^{\otimes k+1} + \sum_{l=0}^{k-1} I^{\otimes (k+1)-1-l} \otimes \left(c(\sigma_- \otimes \sigma_+^{\otimes l-1}) + a(\sigma_+ \otimes \sigma_-^{\otimes l-1})\right) \\ + c(\sigma_- \otimes \sigma_+^{\otimes k}) + a(\sigma_+ \otimes \sigma_-^{\otimes k}) \quad (3.2)$$

$$\Rightarrow bI^{\otimes k+1} + \sum_{l=0}^{k-1} I^{\otimes (k+1)-1-l} \otimes \left(c(\sigma_- \otimes \sigma_+^{\otimes l-1}) + a(\sigma_+ \otimes \sigma_-^{\otimes l-1})\right) \\ \underbrace{I^{\otimes (k+1)-1-k} \left(c(\sigma_- \otimes \sigma_+^{\otimes k}) + a(\sigma_+ \otimes \sigma_-^{\otimes k})\right)}_{(k+1)^{th}\text{term}} \quad (3.3)$$

$$\downarrow$$

$$bI^{\otimes k+1} \;+\; \sum_{l=0}^{(k+1)-1} I^{\otimes (k+1)-1-l} \;\otimes\; \left(c(\sigma_- \otimes \sigma_+^{\otimes l}) + a(\sigma_+ \otimes \sigma_-^{\otimes l}\right) \quad := \quad T_{k+1} \quad (3.4)$$

We will return to the importance of this decomposition later when examining the Hamiltonian simulation subroutine as part of the HHL algorithm.

3.1.3 Dirichlet Boundary conditions in the one dimensional case

Consider again the difference matrix associated with evaluating the second derivative.

$$\begin{pmatrix} V''(x_1) \\ V''(x_2) \\ \vdots \\ \\ V''(x_{N-1}) \end{pmatrix} = \frac{1}{h^2} \begin{pmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & & \vdots \\ 0 & 1 & -2 & \ddots & \\ \vdots & & \ddots & \ddots & 1 \\ 0 & \dots & & 1 & -2 \end{pmatrix} \begin{pmatrix} V(x_1) \\ V(x_2) \\ \vdots \\ \\ V(x_{N-1}) \end{pmatrix}$$

There is one subtlety that has not been addressed yet in this formulation. There is an issue here as we have not discussed how the derivative is evaluated on the interior gridpoints adjacent to the boundaries. Explicit computation of $V''(x_1)$ and $V''(x_{N-1})$ gives:

$$\begin{aligned} V''(x_1) &= \frac{-2V(x_1) + V(x_2)}{h^2} \\ V''(x_{N-1}) &= \frac{V(x_{N-2}) - 2V(x_{N-1})}{h^2} \end{aligned}$$

We are missing information here namely $V(x_0)$ and $V(x_N)$ to evaluate the derivatives. However, if $V(x_0) = 0$ and $V(x_N) = 0$ the previous equations can be reinterpreted as:

$$\begin{aligned} V''(x_1) &= \frac{0 - 2V(x_1) + V(x_2)}{h^2} = \frac{V(x_0) - 2V(x_1) + V(x_2)}{h^2} \\ V''(x_{N-1}) &= \frac{V(x_{N-2}) - 2V(x_{N-1}) + 0}{h^2} = \frac{V(x_{N-2}) - 2V(x_{N-1}) + V(x_N)}{h^2} \end{aligned}$$

Hence, the difference matrix above can be reinterpreted as approximating the derivative of a function $V(x)$ where the function is held at zero at the endpoints. (This is analogous to Poisson's equation with the boundary values set to zero).

The boundary values can be adjusted to some other value or function easily.

If $V(t,x_0) = L(t)$, $V(t,x_N) = U(t)$, then substituting yields:

$$\begin{aligned} V''(t,x_1) &= \frac{L(t) - 2V(t,x_1) + V(t,x_2)}{h^2} \\ V''(t,x_{N-1}) &= \frac{V(t,x_{N-2}) - 2V(t,x_{N-1}) + U(t)}{h^2} \end{aligned}$$

The previous matrix equation can be modified to reflect this change in boundary conditions as:

$$\begin{pmatrix} V''(t,x_1) \\ V''(t,x_2) \\ \vdots \\ \\ V''(t,x_{N-1}) \end{pmatrix} = \frac{1}{h^2} \begin{pmatrix} -2 & 1 & 0 & \dots & 0 \\ 1 & -2 & 1 & & \vdots \\ 0 & 1 & -2 & \ddots & \\ \vdots & & \ddots & \ddots & 1 \\ 0 & \dots & & 1 & -2 \end{pmatrix} \begin{pmatrix} V(t,x_1) \\ V(t,x_2) \\ \vdots \\ \\ V(t,x_{N-1}) \end{pmatrix} + \frac{1}{h^2} \begin{pmatrix} L(t) \\ 0 \\ \vdots \\ 0 \\ U(t) \end{pmatrix}$$

However for the case of the difference matrix that corresponds to approximating the first derivative, there is a sign change that must be accounted for. If $V(t,x_0) = L(t)$, $V(t,x_N) = U(t)$, this can be accounted for as:

$$\begin{pmatrix} V'(t,x_1) \\ V'(t,x_2) \\ \vdots \\ \\ V'(t,x_{N-1}) \end{pmatrix} = \frac{1}{2h} \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ -1 & 0 & 1 & & \vdots \\ 0 & -1 & 0 & \ddots & \\ \vdots & & \ddots & \ddots & 1 \\ 0 & \dots & & -1 & 0 \end{pmatrix} \begin{pmatrix} V(t,x_1) \\ V(t,x_2) \\ \vdots \\ \\ V(t,x_{N-1}) \end{pmatrix} + \frac{1}{2h} \begin{pmatrix} -L(t) \\ 0 \\ \vdots \\ 0 \\ U(t) \end{pmatrix}$$

$$\begin{aligned} V'(t,x_1) &= \frac{-L(t) + V(t,x_2)}{2h} \Leftrightarrow \frac{-V(t,x_0) + V(t,x_2)}{2h} \\ V'(t,x_{N-1}) &= \frac{-V(t,x_{N-2}) + U(t)}{2h} \Leftrightarrow \frac{-V(t,x_{N-2}) + V(t,x_N)}{2h} \end{aligned}$$

3.1.4 Dirichlet boundary conditions in the multi-dimensional case

If we consider a d-dimensional mesh $\Omega_{I_d \times \ldots \times I_1}$ where each interval I_j has $N+1$ gridpoints ($I_j = [(x_j)_0, (x_j)_2, .., (x_j)_N]$), the *lower* boundary conditions are:

$$
\begin{aligned}
V(t,(x_d)_{j_d},..,(x_2)_{j_2},(x_1)_0) &= L_{x_1}(t,(x_d)_{j_d},..,(x_2)_{j_2},\widehat{(x_1)}_{j_1}) \\
V(t,(x_d)_{j_d},..,(x_2)_0,(x_1)_{j_1}) &= L_{x_2}(t,(x_d)_{j_d},..,\widehat{(x_2)}_{j_2},(x_1)_{j_1}) \\
&\vdots \\
V(t,(x_d)_0,..,(x_2)_{j_2},(x_1)_{j_1}) &= L_{x_d}(t,\widehat{(x_d)}_{j_d},..,(x_2)_{j_2},(x_1)_{j_1})
\end{aligned}
$$

The *upper* boundary conditions are:

$$
\begin{aligned}
V(t,(x_d)_{j_d},..,(x_2)_{j_2},(x_1)_{N-1}) &= U_{x_1}(t,(x_d)_{j_d},..,(x_2)_{j_2},\widehat{(x_1)}_{j_1}) \\
V(t,(x_d)_{j_d},..,(x_2)_{N-1},(x_1)_{j_1}) &= U_{x_2}(t,(x_d)_{j_d},..,\widehat{(x_2)}_{j_2},(x_1)_{j_1}) \\
&\vdots \\
V(t,(x_d)_{N-1},..,(x_2)_{j_2},(x_1)_{j_1}) &= U_{x_d}(t,\widehat{(x_d)}_{j_d},..,(x_2)_{j_2},(x_1)_{j_1})
\end{aligned}
$$

The caret symbol here, $\widehat{\ }$ denotes that this argument is omitted from the function domain. To declutter the following equations, we will condense the notation in a similar fashion as [35] such that points within the mesh will be represented as:

$$((x_d)_{j_d},(x_{d-1})_{j_{d-1}},...,(x_1)_{j_1}) \leftrightarrow (j_d,j_{d-1},...,j_1)$$

Let us now consider how the boundary conditions can be imposed. For the d dimensional case, if the first derivative is being taken with respect to the axis (x_a), we must correct the value of $V(t,j_d,j_{d-1},...,j_1)$ by:

$$-L_{x_a}(t,j_d,j_{d-1},..,\widehat{j_a},..,j_1) \quad \text{at the point} \quad (j_d,j_{d-1},..,\underbrace{\quad 1 \quad}_{a^{th}\text{position}},..,j_1)$$

$$U_{x_a}(t,j_d,j_{d-1},..,\widehat{j_a},..,j_1) \quad \text{at the point} \quad (j_d,j_{d-1},..,\underbrace{N-1}_{a^{th}\text{position}},..,j_1)$$

For the case of taking the second derivative along the axis (x_a), the expression is:

$$L_{x_a}(t, j_d, j_{d-1}, .., \widehat{j_a}, .., j_1) \quad \text{at the point} \quad (j_d, j_{d-1}, .., \underbrace{1}_{a^{th}\text{position}}, .., j_1)$$

$$U_{x_a}(t, j_d, j_{d-1}, .., \widehat{j_a}, .., j_1) \quad \text{at the point} \quad (j_d, j_{d-1}, .., \underbrace{N-1}_{a^{th}\text{position}}, .., j_1)$$

Finally for the case of mixed partial derivatives, if we are evaluating the second derivative along the $(x_a), (x_b)$ axes, the boundary conditions are:

$$-L_{x_a,x_b}(t, j_d, .., j_b, .., \widehat{j_a}, .., j_1) \quad \text{at the point} \quad (j_d, .., j_b, .., \underbrace{1}_{a^{th}\text{position}}, .., j_1)$$

$$U_{x_a,x_b}(t, j_d, .., j_b, .., \widehat{j_a}, .., j_1) \quad \text{at the point} \quad (j_d, .., j_b, .., \underbrace{N-1}_{a^{th}\text{position}}, .., j_1)$$

$$-L_{x_a,x_b}(t, j_d, .., \widehat{j_b}, .., j_a, .., j_1) \quad \text{at the point} \quad (j_d, .., \underbrace{1}_{b^{th}\text{position}}, .., j_a, .., j_1)$$

$$U_{x_a,x_b}(t, j_d, .., \widehat{j_b}, .., j_a, .., j_1) \quad \text{at the point} \quad (j_d, .., \underbrace{N-1}_{b^{th}\text{position}}, .., j_a, .., j_1)$$

The goal now is to define the vector for each differential operator that will impose the corresponding boundary conditions. The index function $\mathcal{N}(j_d, ..., j_1) = \sum_{m=1}^{d}(N-1)^{m-1}(j_m - 1) + 1$ can be used to translate from the overall index in the $(N-1)^d$ dimensional vector to the location along each of the individual axes. To impose the boundary condition for the first derivative along some axis (x_c), the vector $\vec{B}_{\partial x_c}(t)$ will be added whose l^{th} component is:

$$(\vec{B}_{(\partial x_c)}(t))_l = \frac{1}{2h} \sum_{\substack{j_1, j_2, .., j_d \\ j_i \neq j_c}} \Big[-\delta_{l, \mathcal{N}(j_d, ., \underset{\overset{\uparrow}{c^{th}}}{1}, ., j_1)} L_{x_c}(t, j_d, ., \widehat{j_t}, ., j_1) \\ + \delta_{l, \mathcal{N}(j_d, ., \underset{\overset{\uparrow}{c^{th}}}{N-1}, ., j_1)} U_{x_c}(t, j_d, ., \widehat{j_t}., j_1) \Big]$$

where the values 1 and $N-1$ are at the c^{th} argument in the indexing function. Similarly for the case of the second derivative along the same axis (x_c), the vector

$\vec{B}_{(\partial^2 x_c)}(t)$ will be added whose l^{th} component is:

$$(\vec{B}_{(\partial^2 x_c)}(t))_l = \frac{1}{h^2} \sum_{\substack{j_1, j_2, .., j_d \\ j_i \neq j_c}} \Big[\delta_{l, \mathcal{N}(j_d, ., \underset{\underset{c^{th}}{\uparrow}}{1}, ., j_1)} L_{x_c}(t, j_d, ., \widehat{j_t}, ., j_1)$$

$$+ \delta_{l, \mathcal{N}(j_d, ., \underset{\underset{c^{th}}{\uparrow}}{N-1}, ., j_1)} U_{x_c}(t, j_d, ., \widehat{j_t}., j_1) \Big]$$

Finally, for the case of the mixed partial derivatives along the axes $(x_a), (x_b)$, the vector $\vec{B}_{(\partial x_a, \partial x_b)}$ may be expressed as:

$$(\vec{B}_{(\partial x_a, \partial x_b)}(t))_l = \frac{1}{4h^2} \sum_{\substack{j_1, j_2, .., j_d \\ j_i \neq j_a \\ j_i \neq j_b}} \Big[-\delta_{l, \mathcal{N}(j_d, ., \underset{\underset{a^{th}}{\uparrow}}{1}, ., j_1)} L_{x_a}(t, j_d, .., \widehat{j_a}, .., j_1)$$

$$- \delta_{l, \mathcal{N}(j_d, ., \underset{\underset{b^{th}}{\uparrow}}{1}, ., j_1)} L_{x_b}(t, j_d, .., \widehat{j_b}, .., j_d)$$

$$+ \delta_{l, \mathcal{N}(j_d, ., \underset{\underset{a^{th}}{\uparrow}}{N-1}, ., j_1)} U_{x_a}(t, j_d, .., \widehat{j_a}, .., j_1)$$

$$+ \delta_{l, \mathcal{N}(j_d, ., \underset{\underset{b^{th}}{\uparrow}}{N-1}, ., j_1)} U_{x_b}(t, j_d, .., \widehat{j_b}, .., j_1) \Big]$$

We will now consider an elementary example of a PDE in two dimensions. Consider the following mesh:

$$\Omega_{I_1 \times I_2} = I_1 \times I_2$$

$$I_1 = [(x_1)_0, (x_1)_1, ..., (x_1)_N]$$

$$I_2 = [(x_2)_0, (x_2)_1, ..., (x_2)_N]$$

and the following PDE with respective boundary conditions:

$$\frac{\partial^2 V}{\partial x_2^2} + \frac{\partial V}{\partial x_1} = \frac{\partial V}{\partial t}$$

$$V(t, (x_2)_{j_2}, (x_1)_0) = L_{x_1}(t, (x_2)_{j_2}) \quad V(t, (x_1)_N, (x_2)_{j_2}) = U_{x_1}(t, (x_2)_{j_2})$$

$$V(t, (x_2)_0, (x_1)_{j_1}) = L_{x_2}(t, (x_1)_{j_1}) \quad V(t, (x_1)_{j_1}, (x_2)_N) = U_{x_2}(t, (x_1)_{j_1})$$

The boundary conditions can be re-expressed in the condensed notation as:

$$\begin{aligned} V(t, j_2, 0) &= L_{x_1}(t, j_2) \quad V(t, j_2, N) = U_{x_1}(t, j_2) \\ V(t, 0, j_1) &= L_{x_2}(t, j_1) \quad V(t, N, j_1) = U_{x_2}(t, j_1) \end{aligned}$$

The differential operator will be represented as the following sum of tensor products:

$$D_2 \otimes I + I \otimes D_1$$

and the governing equation on the interior $\tilde{\Omega}_{I_1 \times I_2}$ is given by:

$$(D_2 \otimes I + I \otimes D_1) \begin{pmatrix} V(t, (x_2)_1, (x_1)_1) \\ V(t, (x_2)_1, (x_1)_2) \\ V(t, (x_2)_1, (x_1)_3) \\ \dots \end{pmatrix} = \frac{\partial}{\partial t} \begin{pmatrix} V(t, (x_2)_1, (x_1)_1) \\ V(t, (x_2)_1, (x_1)_2) \\ V(t, (x_2)_1, (x_1)_3) \\ \dots \end{pmatrix}$$

$$(D_2 \otimes I + I \otimes D_1) \begin{pmatrix} V(t, (1, 1)) \\ V(t, (1, 2)) \\ V(t, (1, 3)) \\ \dots \end{pmatrix} = \frac{\partial}{\partial t} \begin{pmatrix} V(t, (1, 1)) \\ V(t, (1, 2)) \\ V(t, (1, 3)) \\ \dots \end{pmatrix}$$

Considering just the right hand side of the previous expression and expanding it yields:

$$(D_2 \otimes I) \begin{pmatrix} V(t, (1, 1)) \\ V(t, (1, 2)) \\ V(t, (1, 3)) \\ \dots \end{pmatrix} + (I \otimes D_1) \begin{pmatrix} V(t, (1, 1)) \\ V(t, (1, 2)) \\ V(t, (1, 3)) \\ \dots \end{pmatrix}$$

$$\to \begin{pmatrix} \partial_{x_2}^2 V(t, (1, 1)) \\ \dots \\ \partial_{x_2}^2 V(t, (1, N-1)) \\ \partial_{x_2}^2 V(t, (2, 1)) \\ \dots \end{pmatrix} + \begin{pmatrix} \partial_{x_1} V(t, (1, 1)) \\ \dots \\ \partial_{x_1} V(t, (1, N-1)) \\ \partial_{x_1} V(t, (2, 1)) \\ \dots \end{pmatrix}$$

Adding the boundary conditions to the original matrix equation yields:

$$\begin{pmatrix} \partial^2_{x_2} V(t, \overbrace{(1,1)}^{(x_2,x_1)}) \\ ... \\ \partial^2_{x_2} V(t,(1,N-1)) \\ \partial^2_{x_2} V(t,(2,1)) \\ ... \end{pmatrix} + \underbrace{\begin{pmatrix} L_{x_2}(t,1) \\ ... \\ L_{x_2}(t,N-1) \\ 0 \\ 0 \downarrow \end{pmatrix}}_{\vec{B}_{(\partial x_2)}} + \begin{pmatrix} \partial_{x_1} V(t, \overbrace{(1,1)}^{(x_2,x_1)}) \\ ... \\ \partial_{x_1} V(t,(1,N-1)) \\ \partial_{x_1} V(t,(2,1)) \\ ... \end{pmatrix} + \underbrace{\begin{pmatrix} -L_{x_1}(t,1) \\ 0 \updownarrow \\ U_{x_1}(t,1) \\ -L_{x_1}(t,2) \\ 0 \downarrow \end{pmatrix}}_{\vec{B}_{(\partial x_1)}}$$

Combining the terms $\vec{B}_{(\partial x_1)}(t), \vec{B}_{(\partial x_1)}(t)$ into some vector $\vec{B}(t)$ yields the final form of the linear system of ODEs to be solved:

$$\vec{B}(t) = \vec{B}_{(\partial x_1)} + \vec{B}_{(\partial x_2)} \tag{3.5}$$

$$\frac{\partial V}{\partial t} = (D_2 \otimes I + I \otimes D_1) V + \vec{B}(t) \tag{3.6}$$

3.2 Numerical methods for partial differential equations

We will now examine how a partial differential equation can be expressed in the form of (3.5) and (3.6).

3.2.1 Multi-dimensional Black-Scholes Equation

The single asset Black-Scholes equation, which governs the dynamics of option V which derives its values from an asset S is given by:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

with final payoff condition at maturity T at a strike price K given by:

$$V(T,S) = \max(p(K-S),0); \qquad p = \begin{cases} +1 & \text{call} \\ -1 & \text{put} \end{cases}$$

Assuming an European call option, the initial and boundary conditions are:

$$\begin{aligned} V(T,S) &= \max(S-K,0) \quad \forall S \\ V(t,0) &= 0 \quad \forall t \\ V(t,S) &\approx S-Ke^{-r(T-t)} \quad \text{as} \quad S\to\infty \end{aligned}$$

Similarly, for an European put option, the initial boundary conditions are:

$$\begin{aligned} V(T,S) &= \max(K-S,0) \quad \forall S \\ V(t,0) &= Ke^{-r(T-t)} \quad \forall t \\ V(t,S) &\approx 0 \quad \text{as} \quad S\to\infty \end{aligned}$$

The d-dimensional Black-Scholes equation which governs the value of multi asset option V contingent on assets $S_1, S_2, ..., S_d$ can be expressed as:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sum_{i,j=1}^{d} \rho_{ij}\sigma_i\sigma_j S_i S_j \frac{\partial^2 V}{\partial S_i \partial S_j} + \sum_{i=1}^{d} rS_i \frac{\partial V}{\partial S_i} - rV = 0$$

with a payoff condition at maturity T:

$$V(T,(S_1,S_2,...,S_d)) = f(S_1,S_2,...,S_d)$$

The nature of f depends on the option type. For 'worst of' option types for example, the payoff function for a call option is given by:

$$f(S_1,S_2,...,S_d) = \max(K - \min(S_1,S_2,...,S_d),0)$$

In either the single or multi-asset case, the evaluation of the option price amounts to finding the present value of the option:

$$\underbrace{V(0,S_{\text{now}})}_{\text{price of single-asset option}} \qquad \underbrace{V(0,(S_1)_{\text{now}},(S_2)_{\text{now}},...,(S_d)_{\text{now}})}_{\text{price of multi-asset option}}$$

3.2.2 Reduction to linear partial differential equation

The ordinary/partial differential equation in its current form is nonlinear. As the quantum algorithm is explicitly for solving linear systems of ordinary differential equations (ODEs), we have to linearize the equation by making the following variable changes [53]:

$$x_i := \ln S_i; \qquad \tau := T - t$$
$$(S_i = e^{x_i} : \qquad t = T - \tau)$$

Hence the derivatives with respect to x and S can be rewritten as:

$$\frac{\partial}{\partial S_i} = \frac{dx_i}{dS_i}\frac{\partial}{\partial x_i} \qquad \frac{\partial}{\partial t} = \frac{d\tau}{dt}\frac{\partial}{\partial \tau}$$

$$\frac{\partial}{\partial S_i} = \frac{1}{S_i}\frac{\partial}{\partial x_i} \qquad \frac{\partial}{\partial t} = -\frac{\partial}{\partial \tau}$$

Substituting these definitions into the Black-Scholes equation yields:

$$-\frac{\partial V}{\partial \tau} + \frac{1}{2}\sum_{i,j=1}^{d} \rho_{ij}\sigma_i\sigma_j S_i S_j \left(\frac{1}{S_i}\frac{\partial}{\partial x_i}\left(\frac{1}{S_j}\frac{\partial V}{\partial x_j}\right)\right) + \sum_{i=1}^{d} r S_i \left(\frac{1}{S_i}\frac{\partial V}{\partial x_i}\right) - rV = 0$$

$$-\frac{\partial V}{\partial \tau} + \frac{1}{2}\sum_{i,j=1}^{d} \rho_{ij}\sigma_i\sigma_j S_j \frac{\partial}{\partial x_i}\left(\frac{1}{S_j}\frac{\partial V}{\partial x_j}\right) + \sum_{i=1}^{d} r \left(\frac{\partial V}{\partial x_i}\right) - rV = 0$$

$$-\frac{\partial V}{\partial \tau} + \frac{1}{2}\sum_{i,j=1}^{d} \rho_{ij}\sigma_i\sigma_j S_j \left(\frac{1}{S_j}\frac{\partial^2 V}{\partial x_i \partial x_j} + \frac{\partial}{\partial x_i}\left(\frac{1}{S_j}\right)\frac{\partial V}{\partial x_j}\right) + \sum_{i=1}^{d} r \frac{\partial V}{\partial x_i} - rV = 0 \quad (3.7)$$

As $\frac{1}{S_i} = e^{-x_i}$

$$\frac{\partial}{\partial x_i}\left(\frac{1}{S_j}\right) = -\delta_{ij}\frac{1}{S_i} \quad (3.8)$$

Substituting (3.8) into (3.7) yields:

$$\Rightarrow -\frac{\partial V}{\partial \tau} + \frac{1}{2}\sum_{i,j=1}^{d} \rho_{ij}\sigma_i\sigma_j S_j \left(\frac{1}{S_j}\frac{\partial^2 V}{\partial x_i \partial x_j} - \delta_{ij}\frac{1}{S_i}\frac{\partial V}{\partial x_j}\right) + \sum_{i=1}^{d} r \frac{\partial V}{\partial x_i} - rV = 0$$

$$-\frac{\partial V}{\partial \tau} + \frac{1}{2}\sum_{i,j=1}^{d} \rho_{ij}\sigma_i\sigma_j \frac{\partial^2 V}{\partial x_i \partial x_j} - \frac{1}{2}\sum_{i=1}^{d} \rho_{ii}\sigma_i^2 \frac{\partial V}{\partial x_i} + \sum_{i=1}^{d} r \frac{\partial V}{\partial x_i} - rV = 0$$

$$-\frac{\partial V}{\partial \tau} + \frac{1}{2}\sum_{i,j=1}^{d} \rho_{ij}\sigma_i\sigma_j \frac{\partial^2 V}{\partial x_i \partial x_j} + \sum_{i=1}^{d} \left(r - \frac{1}{2}\sigma_i^2\right)\frac{\partial V}{\partial x_i} - rV = 0$$

$$\frac{\partial V}{\partial \tau} = \frac{1}{2}\sum_{i,j=1}^{d} \rho_{ij}\sigma_i\sigma_j \frac{\partial^2 V}{\partial x_i \partial x_j} + \sum_{i=1}^{d} \left(r - \frac{1}{2}\sigma_i^2\right)\frac{\partial V}{\partial x_i} - rV$$

The final substitution is given by creating a new variable W such that:

$$W := e^{r\tau}V \qquad (V = e^{-r\tau}W)$$

As the term $e^{-r\tau}$ features no dependence on spatial variable x_i, W can be substituted in V for any of the spatial derivatives as:

$$\frac{\partial W}{\partial x} = e^{r\tau}\frac{\partial V}{\partial x}$$

However for the temporal derivative, the derivative becomes:

$$\frac{\partial V}{\partial \tau} = \frac{\partial}{\partial \tau}\left(e^{-r\tau}W\right) = e^{-r\tau}\frac{\partial W}{\partial \tau} - re^{-r\tau}W = e^{-r\tau}\frac{\partial W}{\partial \tau} - rV$$

Replacing into the above equation yields the final form of the Black-Scholes equation that we will use throughout the thesis:

$$e^{-r\tau}\frac{\partial W}{\partial \tau} - rV = \frac{1}{2}\sum_{i,j=1}^{d}\rho_{ij}\sigma_i\sigma_j\frac{\partial^2(e^{-r\tau}W)}{\partial x_i \partial x_j} + \sum_{i=1}^{d}\left(r - \frac{1}{2}\sigma_i^2\right)\frac{\partial(e^{-r\tau}W)}{\partial x_i} - rV$$

$$e^{-r\tau}\frac{\partial W}{\partial \tau} = e^{-r\tau}\frac{1}{2}\sum_{i,j=1}^{d}\rho_{ij}\sigma_i\sigma_j\frac{\partial^2 W}{\partial x_i \partial x_j} + e^{-r\tau}\sum_{i=1}^{d}\left(r - \frac{1}{2}\sigma_i^2\right)\frac{\partial W}{\partial x_i}$$

Hence, we arrive at the final linear partial differential equation:

$$\frac{\partial W}{\partial \tau} = \frac{1}{2}\sum_{i,j=1}^{d}\rho_{ij}\sigma_i\sigma_j\frac{\partial^2 W}{\partial x_i \partial x_j} + \sum_{i=1}^{d}\left(r - \frac{1}{2}\sigma_i^2\right)\frac{\partial W}{\partial x} \tag{3.9}$$

3.2.3 Method of lines

The reason for the change of variables in the above fashion is that the resulting differential equation is linear. This means that if we discretize and express the partial differential equation on a grid, the partial differential equation can be converted into a system of ordinary differential equations of the form:

$$\frac{d\vec{W}}{d\tau} = A\vec{W}$$

$$\vec{W} = (W(x_1), W(x_2), ..., W(x_N))$$

where A is some matrix and the indexing l refers to all gridpoints that arise from the discretization, that has been previously defined. If we consider the PDE (3.9) in its above form, using the method of lines, it may be re-expressed as a linear algebraic problem. First of all, we need to split up the partial derivatives of second order in to mixed (cross derivatives) and unmixed:

$$\frac{\partial W}{\partial \tau} = \frac{1}{2}\sum_{i,j=1}^{d} \rho_{ij}\sigma_i\sigma_j \frac{\partial^2 W}{\partial x_i \partial x_j} + \sum_{i=1}^{d}\left(r - \frac{1}{2}\sigma_i^2\right)\frac{\partial W}{\partial x}$$

$$\downarrow$$

$$\frac{\partial W}{\partial \tau} = \frac{1}{2}\sum_{i=1}^{d}\sigma_i^2\frac{\partial^2 W}{\partial x_i^2} + \frac{1}{2}\sum_{\substack{i,j=1\\ i\neq j}}^{d} \rho_{ij}\sigma_i\sigma_j \frac{\partial^2 W}{\partial x_i \partial x_j} + \sum_{i=1}^{d}\left(r - \frac{1}{2}\sigma_i^2\right)\frac{\partial W}{\partial x_i}$$

Due to the symmetry of the partial derivatives, $\left(\frac{\partial^2}{\partial x_i \partial x_j} = \frac{\partial^2}{\partial x_j \partial x_i}\right)$, the second summation can be reindexed to avoid double counting, picking up a factor of two:

$$\frac{\partial W}{\partial \tau} = \frac{1}{2}\sum_{i=1}^{d}\sigma_i^2\frac{\partial^2 W}{\partial x_i^2} + \sum_{i=1}^{d}\sum_{j>i}^{d} \rho_{ij}\sigma_i\sigma_j \frac{\partial^2 W}{\partial x_i \partial x_j} + \sum_{i=1}^{d}\left(r - \frac{1}{2}\sigma_i^2\right)\frac{\partial W}{\partial x_i}$$

Using the finite difference scheme alluded to previously, this can be expressed in the tensor product notation as:

$$\frac{d\vec{W}}{d\tau} = \frac{1}{2}\sum_{i=1}^{d}\sigma_i^2(I^{\otimes d-l}\otimes D_2 \otimes I^{\otimes l-1})\vec{W}$$

$$+\sum_{\substack{i,j=1\\ j>i}}^{d}\rho_{ij}\sigma_i\sigma_j(I^{\otimes d-i}D_1\otimes I^{\otimes j-i-1}\otimes D_1\otimes I^{\otimes j-1})\vec{W}$$

$$+\sum_{i=1}^{d}\left(r - \frac{\sigma_i^2}{2}\right)(I^{\otimes d-i}\otimes D_1\otimes I^{\otimes i-1})\vec{W}$$

Using the following notation, the above equation can be condensed further. Collecting all the boundary condition terms in to the vector $\vec{B(t)}$ yields:

$$\begin{aligned} A_{(\partial x_i)} &= I^{\otimes d-i} \otimes D_1 \otimes I^{\otimes i-1} \qquad\qquad A_{(\partial^2 x_i)} = I^{\otimes d-l} \otimes D_2 \otimes I^{\otimes l-1} \\ A_{(\partial x_i, \partial x_j)} &= I^{\otimes d-i} D_1 \otimes I^{\otimes j-i-1} \otimes D_1 \otimes I^{\otimes j-1} \end{aligned}$$

$$\frac{d\vec{W}}{d\tau} = \frac{1}{2}\sum_{i=1}^{d} \sigma_i^2 A_{(\partial^2 x_i)} \vec{W} + \sum_{\substack{i,j=1 \\ j>i}}^{d} \rho_{ij}\sigma_i\sigma_j A_{(\partial x_i, \partial x_j)} \vec{W} + \sum_{i=1}^{d} \left(r - \frac{\sigma_i^2}{2} \right) A_{(\partial x_i)} \vec{W} + \vec{B}(\tau)$$

$$\Rightarrow \left(\frac{1}{2}\sum_{i=1}^{d} \sigma_i^2 A_{(\partial^2 x_i)} + \sum_{\substack{i,j=1 \\ j>i}}^{d} \rho_{ij}\sigma_i\sigma_j A_{(\partial x_i, \partial x_j)} + \sum_{i=1}^{d} \left(r - \frac{\sigma_i^2}{2} \right) A_{(\partial x_i)} \right) \vec{W} + \vec{B}(\tau)$$

At last, we have the matrix differential equation:

$$\frac{d\vec{W}}{d\tau} = A\vec{W} + \vec{B}(\tau) \tag{3.10}$$

3.3 Matrix formulation of problem

We now proceed with showing how (3.10) may be expressed and therefore solved in a matrix formulation.

3.3.1 Analytic solution to matrix differential equation

The analytic solution for the case of constant $\vec{B}(\tau) = \vec{B}_0$ in the above problem:

$$\frac{d\vec{X}}{d\tau} = A\vec{X} + \vec{B}_0 \qquad \vec{X}(0) = \vec{X}_0$$

may be expressed as:

$$\vec{X}(T) = e^{AT}\vec{X}_0 + \left(e^{AT} - I \right) A^{-1}\vec{B}_0$$

Currently, there exists a quantum algorithm that re-expresses the above solution as the inverse of an auxilliary matrix [5]. However, the requirement for constant $\vec{B}_0$ reflects the assumption that the boundary conditions only vary with asset price. This may be realistic for barrier types within options however it would be preferrable to have some possibility of time dependence within the boundary conditions. Namely, we seek a quantum algorithm that can solve the problem:

$$\frac{d\vec{X}}{d\tau} = A\vec{X} + \vec{B}(\tau) \qquad \vec{X}(0) = \vec{X}_0$$

Although there has been recent work examining solutions to this problem with time-varying matrix A using Dyson series methods [4], we expand the original authors approach in [5] for the case of constant matrix A and time-varying vector $\vec{B}(\tau)$:

$$\vec{X}(T) = e^{AT}\vec{X}_0 + e^{AT}\left(\int_0^T e^{-As}\vec{B}(s)ds\right)$$

To simplify the analysis, consider a Taylor expansion of $\vec{B}(\tau)$:

$$\vec{B}(\tau) \approx \vec{B}_0 + \tau\vec{B}_1 + \frac{\tau^2}{2!}\vec{B}_2 + \mathcal{O}(\tau^3)$$

$$\vec{B}(\tau) = \sum_{l=0}^{\infty}\frac{\tau^l}{l!}\vec{B}_l \tag{3.11}$$

Substituting (3.11) into the above equation for the solution above yields:

$$\vec{X}(T) = e^{AT}\vec{X}_0 + e^{AT}\left(\int_0^T e^{-At}\left(\sum_{l=0}^{\infty}\frac{t^l}{l!}\vec{B}_l\right)dt\right) \tag{3.12}$$

$$\vec{X}(T) = e^{AT}\vec{X}_0 + \sum_{l=0}^{\infty}\frac{e^{AT}}{l!}\left(\int_0^T t^l e^{-At}dt\right)\vec{B}_l$$

$$\begin{aligned} I_n(T) := \int_0^T t^n e^{-At}dt &= t^n(-A^{-1}e^{-At})\Big|_0^T - n\int_0^T t^{n-1}(-A^{-1}e^{-At}) \\ &= \left(-T^nA^{-1}e^{-AT} + 0\right) + nA^{-1}\underbrace{\int_0^T t^{n-1}e^{-At}dt}_{I_{n-1}} \end{aligned}$$

$$I_n(T) = A^{-1}\left(nI_{n-1} - T^ne^{-AT}\right) \qquad I_0(T) = A^{-1}\left(I - e^{-AT}\right)$$

The recursive relationship has the following exact form:

$$I_n = n!A^{-(n+1)}\left(I - e^{-AT}\sum_{k=0}^{n}\frac{(AT)^k}{k!}\right) \tag{3.13}$$

This can be seen as:

$$\begin{aligned}
I_n &= A^{-1}\left(nI_{n-1} - T^n e^{-AT}\right)\\
&= A^{-1}\left[n\left((n-1)!A^{-(n)}\left(I - e^{-AT}\sum_{k=0}^{n-1}\frac{(AT)^k}{k!}\right)\right)\right] - A^{-1}T^n e^{-AT}\\
&= nA^{-1}\left((n-1)!A^{-n}e^{-AT}\left(e^{AT} - \sum_{l=1}^{n-1}\frac{(AT)^l}{l!}\right)\right) - A^{-1}T^n e^{-AT}\\
&= n!A^{-(n+1)}\left(I - e^{-AT}\sum_{k=0}^{n-1}\frac{(AT)^k}{k!}\right) - A^{-1}T^n e^{-AT}\\
&= n!A^{-(n+1)}\left(I - e^{-AT}\sum_{k=0}^{n-1}\frac{(AT)^k}{k!}\right) - n!A^{-(n+1)}\left(\frac{A^nT^ne^{-AT}}{n!}\right)\\
&= n!A^{-(n+1)}\left(I - e^{-AT}\sum_{k=0}^{n-1}\frac{(AT)^k}{k!} - \frac{e^{-AT}(AT)^n}{n!}\right)\\
&= n!A^{-(n+1)}\left(I - e^{-AT}\sum_{k=0}^{n}\frac{(AT)^k}{k!}\right) := I_n
\end{aligned}$$

Returning to the solution to the matrix differential equation and substituting (3.13) for the integral $I_l(T)$ yields:

$$\vec{X}(T) = e^{AT}\vec{X}_0 + \sum_{l=0}^{\infty}\frac{e^{AT}}{l!}\left(l!A^{-(l+1)}\left(I - e^{-AT}\sum_{k=0}^{l}\frac{(AT)^k}{k!}\right)\right)\vec{B}_l$$

$$\vec{X}(T) = e^{AT}\vec{X}_0 + \sum_{l=0}^{\infty}A^{-(l+1)}\left(e^{AT} - \sum_{k=0}^{l}\frac{(AT)^k}{k!}\right)\vec{B}_l \tag{3.14}$$

$$\vec{X}(T) = e^{AT}\vec{X}_0 + \sum_{l=0}^{\infty}A^{-(l+1)}\left(\sum_{k=l+1}^{\infty}\frac{(AT)^k}{k!}\right)\vec{B}_l$$

$$\vec{X}(T) = e^{AT}\vec{X}_0 + \sum_{l=1}^{\infty}A^{-l}\left(\sum_{k=l}^{\infty}\frac{(AT)^k}{k!}\right)\vec{B}_{l-1}$$

$$\vec{X}(T) = e^{AT}\vec{X}_0 + \sum_{l=1}^{\infty}T^l\left(\sum_{k=l}^{\infty}\frac{(AT)^{k-l}}{k!}\right)\vec{B}_{l-1}$$

$$\Rightarrow \vec{X}(T) = e^{AT}\vec{X}_0 + \sum_{l=1}^{\infty} T^l \left(\sum_{k=0}^{\infty} \frac{(AT)^k}{(k+l)!} \right) \vec{B}_{l-1}$$

Incorporating the initial state $\vec{X}_0$ into the summation, the exponential term can be expressed as:

$$\Rightarrow \vec{X}(T) = T^0 \left(\sum_{k=0}^{\infty} \frac{(AT)^k}{(k+0)!} \right) \vec{X}_0 + \sum_{l=1}^{\infty} T^l \left(\sum_{k=0}^{\infty} \frac{(AT)^k}{(k+l)!} \right) \vec{B}_{l-1}$$

If we define a vector V_l:

$$\vec{V}_l = \begin{cases} \vec{X}_0 & l = 0 \\ \vec{B}_{l-1} & l > 0 \end{cases}$$

The solution can be written as:

$$\vec{X}(T) = \sum_{l=0}^{\infty} T^l \left(\sum_{k=0}^{\infty} \frac{(AT)^k}{(k+l)!} \right) \vec{V}_l$$

Considering only the dependence of T, the above solution can also be represented as:

$$\begin{aligned} \vec{X}(T) &= \sum_{l=0}^{\infty} T^l \left(\sum_{k=0}^{\infty} \frac{(AT)^k}{(k+l)!} \right) \vec{V}_l \\ &= \sum_{l=0}^{\infty} \sum_{k=0}^{\infty} \frac{T^{l+k}}{(k+l)!} A^k \vec{V}_l \\ &= \sum_{\lambda=0}^{\infty} \frac{T^\lambda}{\lambda!} \left(\sum_{\substack{k,l \\ k+l=\lambda}} A^k \vec{V}_l \right) = \sum_{\lambda=0}^{\infty} \frac{T^\lambda}{\lambda!} \left(\sum_{k=0}^{\lambda} A^{\lambda-k} \vec{V}_k \right) \end{aligned} \tag{3.15}$$

3.3.2 Representation of solution as matrix inversion

For the following section, assume that each of the components of the vector $\vec{B}(\tau)$, are a polynomial of degree at most d. This is equivalent to the statement that $\vec{V}_l = \vec{0}, \quad \forall l > d+1$ (This can also be thought of as truncating the Taylor expansion of $\vec{B}(\tau)$) if $\vec{B}(\tau)$ does not consist strictly of poly(t) components). Returning to the solution (3.15) yields:

$$\vec{X}(\Delta t) = \sum_{\lambda=0}^{\infty} \frac{(\Delta t)^\lambda}{\lambda!} \left(\sum_{k=0}^{\lambda} A^{\lambda-k} \vec{V}_k \right)$$

If we define $\vec{x}_{l+1} = \frac{(\Delta t)^l}{l!}\left(\sum_{k=0}^{l} A^{l-k}\vec{V}_k\right)$, a recursive relationship can be defined:

$$\begin{aligned}
\vec{x}_{l+1} &= \frac{(\Delta t)^l}{l!}\left(\sum_{k=0}^{l-1} A^{l-k}\vec{V}_k\right) + \frac{h^l}{l!}\vec{V}_{l-1} \\
&= \frac{Ah}{l}\left(\frac{(\Delta t)^{l-1}}{(l-1)!}\sum_{k=0}^{l-1} A^{l-k}\vec{V}_k\right) + \frac{(\Delta t)^l}{l!}\vec{V}_{l-1} \\
&:= \frac{A(\Delta t)}{l}\vec{x}_l + \frac{(\Delta t)^l}{l!}\vec{V}_{l-1}
\end{aligned}$$

To see how this recursive relation can be expressed in matrix form, we express the first few terms of the sequence for a small period of evolution, $T = \Delta t$:

$$\vec{X}(\Delta t) = \underbrace{\left(\vec{V}_0\right)}_{\vec{x}_1} + \underbrace{\Delta t\left(\vec{V}_1 + A\vec{V}_0\right)}_{\vec{x}_2} + \underbrace{\frac{(\Delta t)^2}{2}\left(\vec{V}_2 + A\vec{V}_1 + A^2\vec{V}_0\right)}_{\vec{x}_3} + \mathcal{O}((\Delta t)^3)$$

$$\begin{aligned}
\vec{x}_1 &= \vec{V}_0 \\
\vec{x}_2 &= A\Delta t\vec{V}_0 + \Delta t\vec{V}_1 = A\Delta t\vec{x}_1 + \Delta t\vec{V}_1 \\
\vec{x}_3 &= \frac{A\Delta t}{2}\left(A\Delta t\vec{V}_0 + \Delta t\vec{V}_1\right) + \frac{\Delta t^2}{2}\vec{V}_2 = \frac{A\Delta t}{2}\vec{x}_2 + \frac{\Delta t^2}{2}\vec{V}_2 \\
&\vdots \\
\vec{x}_d &= \frac{A\Delta t}{(d-1)}\vec{x}_{d-1} + \frac{\Delta t^{d-1}}{(d-1)!}\vec{V}_{d-1} \\
\vec{x}_{d+1} &= \frac{A\Delta t}{d}\vec{x}_d \\
&\vdots \\
\vec{x}_n &= \frac{A\Delta t}{n-1}\vec{x}_{n-1}
\end{aligned}$$

The recursive relationship previously defined emerges. Solving for each of these vectors $\{\vec{x}_i\}$ can be achieved by encoding the recursive relationship above into the following linear system:

$$\begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & & \dots & \mathbf{0} \\ -\mathbf{A}(\Delta t) & \mathbf{I} & \mathbf{0} & & \vdots & & & \vdots \\ \mathbf{0} & -\frac{\mathbf{A}(\Delta t)}{2} & \mathbf{I} & \ddots & & & & \\ \vdots & & \ddots & \ddots & \mathbf{0} & & & \\ \mathbf{0} & \dots & & -\frac{\mathbf{A}(\Delta t)}{d-1} & \mathbf{I} & \mathbf{0} & & \\ \vdots & & & & -\frac{\mathbf{A}(\Delta t)}{d} & \mathbf{I} & \ddots & \vdots \\ & & & & & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \dots & & & \dots & & -\frac{\mathbf{A}(\Delta t)}{n-1} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \vec{x}_3 \\ \vdots \\ \vec{x}_d \\ \vec{x}_{d+1} \\ \vdots \\ \vec{x}_n \end{pmatrix} = \begin{pmatrix} \vec{V}_0 \\ (\Delta t)\vec{V}_1 \\ \frac{(\Delta t)^2}{2}\vec{V}_2 \\ \vdots \\ \frac{(\Delta t)^{d-1}}{(d-1)!}\vec{V}_{d-1} \\ \vec{0} \\ \vdots \\ \vec{0} \end{pmatrix}$$

In the above and following matrices, we denote block matrices with boldface. The terms in each of the series solution for $\vec{X}(\Delta t)$ are encoded in each of the $\{\vec{x}_i\}$. In order to find an approximation of the solution, we have to sum each of the terms in the Taylor expansion, $\vec{x}_{n+1} = \sum_{i=1}^{n} \vec{x}_i$. This can be accomplished by adding one extra row and column to the original matrix:

$$\begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & & \dots & & \mathbf{0} \\ -\mathbf{A}(\Delta t) & \mathbf{I} & \mathbf{0} & & \vdots & & & & \vdots \\ \mathbf{0} & -\frac{\mathbf{A}(\Delta t)}{2} & \mathbf{I} & \ddots & & & & & \\ \vdots & & \ddots & \ddots & \mathbf{0} & & & & \\ \mathbf{0} & \dots & & -\frac{\mathbf{A}(\Delta t)}{d-1} & \mathbf{I} & \mathbf{0} & & & \\ \vdots & & & & -\frac{\mathbf{A}(\Delta t)}{d} & \mathbf{I} & \ddots & & \\ & & & & & \ddots & \ddots & \mathbf{0} & \\ & & & & & & -\frac{\mathbf{A}(\Delta t)}{n-1} & \mathbf{I} & \mathbf{0} \\ -\mathbf{I} & \dots & & & & & \dots & -\mathbf{I} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \vec{x}_3 \\ \vdots \\ \vec{x}_d \\ \vec{x}_{d+1} \\ \vdots \\ \vec{x}_n \\ \vec{x}_{n+1} \end{pmatrix} = \begin{pmatrix} \vec{V}_0 \\ (\Delta t)\vec{V}_1 \\ \frac{(\Delta t)^2}{2}\vec{V}_2 \\ \vdots \\ \frac{(\Delta t)^{d-1}}{(d-1)!}\vec{V}_{d-1} \\ \vec{0} \\ \vdots \\ \vec{0} \\ \vec{0} \end{pmatrix}$$

At this final step, the vector $\vec{x}_n$ will be approximately equal to:

$$\vec{X}(\Delta t) = \sum_{\lambda=0}^{n-1} \frac{(\Delta t)^\lambda}{\lambda!} \left(\sum_{k=0}^{\lambda} A^{\lambda-k} \vec{V}_k \right) \approx \sum_{\lambda=0}^{\infty} \frac{(\Delta t)^\lambda}{\lambda!} \left(\sum_{k=0}^{\lambda} A^{\lambda-k} \vec{V}_k \right)$$

for sufficiently small Δt (error $\sim \mathcal{O}((\Delta t)^3)$). However, we have only approximated the evolution over one timestep, Δt. In order to approximate the evolution over a larger time period, the matrix encoding procedure has to be redone and solved again, with $\vec{V}_0 = \vec{X}(\Delta t)$ rather than the original V_0. To illustrate this further, consider the example of $T = 2\Delta t$:

$$\begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \cdots & & & \mathbf{0} \\ -\mathbf{A}\Delta t & \mathbf{I} & \mathbf{0} & & \vdots & & & & \vdots \\ \mathbf{0} & -\frac{\mathbf{A}\Delta t}{2} & \mathbf{I} & \ddots & & & & & \\ \vdots & & \ddots & \ddots & \mathbf{0} & & & & \\ \mathbf{0} & \cdots & & -\frac{\mathbf{A}\Delta t}{d-1} & \mathbf{I} & \mathbf{0} & & & \\ \vdots & & & & -\frac{\mathbf{A}\Delta t}{d} & \mathbf{I} & & & \\ & & & & & \ddots & \ddots & \mathbf{0} & \\ & & & & & & -\frac{\mathbf{A}\Delta t}{n-1} & \mathbf{I} & \mathbf{0} \\ -\mathbf{I} & \cdots & & & & & \cdots & -\mathbf{I} & \mathbf{I} \end{pmatrix} \begin{pmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \vec{x}_3 \\ \vdots \\ \vec{x}_d \\ \vec{x}_{d+1} \\ \vdots \\ \vec{x}_n \\ \vec{x}_{n+1} \end{pmatrix} = \begin{pmatrix} \vec{X}((\Delta t)) \\ (\Delta t)\vec{V}_1 \\ \frac{(\Delta t)^2}{2}\vec{V}_2 \\ \vdots \\ \frac{(\Delta t)^{d-1}}{(d-1)!}\vec{V}_{d-1} \\ \vec{0} \\ \vdots \\ \vec{0} \\ \vec{0} \end{pmatrix}$$

Solving this linear system with modified $\vec{V}_0$ can be reimagined as solving a larger auxilliary system where the previous matrix is effectively 'concatenated' with itself again:

$$\left(\begin{array}{cccc|cccc|c} \mathbf{I} & & & & & & & & \\ -\mathbf{A}\Delta t & \ddots & & & & & & & \\ & \ddots & & & & & & & \\ & & -\frac{\mathbf{A}\Delta t}{n-1} & I & & & & & \\ -\mathbf{I} & \cdots & & -\mathbf{I} & \mathbf{I} & & & & \\ & & & & -\mathbf{A}\Delta t & \ddots & & & \\ & & & & & \ddots & & & \\ & & & & & & -\frac{\mathbf{A}\Delta t}{n-1} & \mathbf{I} & \\ & & & & -\mathbf{I} & \cdots & & -\mathbf{I} & \mathbf{I} \end{array}\right) \left(\begin{array}{c} \vec{x}_1 \\ \vec{x}_2 \\ \vdots \\ \vec{x}_n \\ \hline \vec{x}_{n+1} \\ \vec{x}_{n+2} \\ \vdots \\ \vec{x}_{2n-1} \\ \hline \vec{x}_{2n} \end{array}\right) = \left(\begin{array}{c} \vec{V}_0 \\ (\Delta t)\vec{V}_1 \\ \vdots \\ \vec{0} \\ \hline \vec{0} \\ (\Delta t)\vec{V}_1 \\ \vdots \\ \vec{0} \\ \hline \vec{0} \end{array}\right) \tag{3.16}$$

After solving the linear system, $\vec{x}_{2n}$ will be equal to:

$$\sum_{\lambda=0}^{n-1} \frac{(2\Delta t)^\lambda}{\lambda!} \left(\sum_{k=0}^{\lambda} \mathbf{A}^{\lambda-k}\vec{V}_k \right)$$

which again is approximately the analytic solution, $\vec{X}(2\Delta t)$, for sufficiently small time step Δt:

$$\sum_{\lambda=0}^{n-1} \frac{(2\Delta t)^\lambda}{\lambda!} \left(\sum_{k=0}^{\lambda} \mathbf{A}^{\lambda-k}\vec{V}_k \right) \approx \sum_{\lambda=0}^{\infty} \frac{(2\Delta t)^\lambda}{\lambda!} \left(\sum_{k=0}^{\lambda} \mathbf{A}^{\lambda-k}\vec{V}_k \right) = \vec{X}(2\Delta t)$$

This process can be repeated until the desired time evolution has been achieved. The result can be extracted from the final component of the vector, namely $\vec{x}_{(l)n}$ where l refers to how many timesteps Δt that we have evolved by. For classical algorithms, we solve (5.20) and extract the relevant component. Quantum algorithms, however are fundamentally probabilistic in nature and this fact must be taken into account. The success probability of extracting the relevant component can be increased by repeating the component in the overall solution vector. If we consider the following matrix equation:

$$\begin{pmatrix} -\mathbf{I} & \mathbf{I} & & & \\ & -\mathbf{I} & \mathbf{I} & & \\ & & -\mathbf{I} & \ddots & \\ & & & \ddots & \mathbf{I} \\ & & & & -\mathbf{I} \end{pmatrix} \begin{pmatrix} \vec{v} \\ \vec{w}_1 \\ \vec{w}_2 \\ \vdots \\ \vec{w}_p \end{pmatrix} = \begin{pmatrix} \vec{0} \\ \vec{0} \\ \vec{0} \\ \vdots \\ \vec{0} \end{pmatrix} \tag{3.17}$$

The solution to (3.17) encodes $\vec{v}$ as :

$$\begin{aligned} -\vec{v} + \vec{w}_1 &= 0 \quad && \Rightarrow \vec{w}_1 = \vec{v} \\ -\vec{w}_1 + \vec{w}_2 &= 0 \quad && \Rightarrow \vec{w}_2 = \vec{w}_1 = \vec{v} \\ &\vdots && \\ -\vec{w}_{p-1} + \vec{w}_p &= 0 \quad && \Rightarrow \vec{w}_p = \vec{w}_{p-1} = \vec{v} \end{aligned}$$

Appending this above matrix structure into (5.20) yields:

$$
\left(\begin{array}{ccccccccccc}
\mathbf{I} & & & & & & & & & & \\
-\mathbf{A}\Delta t & \ddots & & & & & & & & & \\
 & \ddots & & & & & & & & & \\
 & & -\frac{\mathbf{A}\Delta t}{n-1} & \mathbf{I} & & & & & & & \\
-\mathbf{I} & \dots & & -\mathbf{I} & \mathbf{I} & & & & & & \\
 & & & & -\mathbf{A}\Delta t & \ddots & & & & & \\
 & & & & & \ddots & & & & & \\
 & & & & & & -\frac{\mathbf{A}\Delta t}{n-1} & \mathbf{I} & & & \\
 & & & & -\mathbf{I} & \dots & & -\mathbf{I} & \mathbf{I} & & \\
 & & & & & & & & -\mathbf{I} & \mathbf{I} & \\
 & & & & & & & & & \ddots & \ddots \\
 & & & & & & & & & -\mathbf{I} & \mathbf{I}
\end{array}\right)
\left(\begin{array}{c}
\vec{x}_1 \\ \vec{x}_2 \\ \vdots \\ \vec{x}_n \\ \hline \vec{x}_{n+1} \\ \vec{x}_{n+2} \\ \vdots \\ \vec{x}_{2n-1} \\ \hline \vec{x}_{2n} \\ \vec{w} \\ \vdots \\ \vec{w}
\end{array}\right)
=
\left(\begin{array}{c}
\vec{V}_0 \\ (\Delta t)\vec{V}_1 \\ \vdots \\ \vec{0} \\ \hline \vec{0} \\ (\Delta t)\vec{V}_1 \\ \vdots \\ \vec{0} \\ \hline \vec{0} \\ \vec{0} \\ \vdots \\ \vec{0}
\end{array}\right)
\tag{3.18}
$$

Where we denote the block matrix of (3.18) as the Λ matrix.

3.4 Discussion

We presented the algorithm, originally described in [5], and developed it for the case of time dependent $\vec{B}(\tau)$. The authors in [5] were able to bound the error between the first two components of their analytic solution:

$$\vec{X}(T) = \underbrace{e^{AT}\vec{X}_0}_{\text{first term}} + \underbrace{A^{-1}\left(e^{AT} - I\right)\vec{B}_0}_{\text{second term}}$$

with their respective truncated series representation (for some integers m_0 and m_1):

$$\vec{X}(T) = \underbrace{\left(\sum_{k=0}^{m_0} \frac{(AT)^k}{k!}\right)\vec{X}_0}_{\text{first term}} + \underbrace{\left(\sum_{k=0}^{m_1} \frac{A^k T^{k+1}}{(k+1)!}\right)\vec{B}_0}_{\text{second term}}$$

One could expand upon their complexity analysis by bounding the error between subsequent analytic terms and their truncated series representation. Specifically, it is possible to bound the error between the $(l+1)^{th}$ term in the analytic expression (3.14) which was derived from (3.12):

$$A^{-l}\left(e^{AT} - \sum_{k=0}^{l-1} \frac{(AT)^k}{k!}\right)\vec{V}_l$$

with its corresponding series representation, truncated at order m_l:

$$\left(\sum_{k=0}^{m_l} \frac{A^k T^{k+l}}{(k+l)!}\right) \vec{V_l}$$

We believe that the argument presented in [5] can be extended, however the corresponding analysis would amount to another thesis on its own right. We hope to continue this work to bound the error between higher order analytic expressions that appear in (3.14) and their respective truncated series representation in future.

Chapter 4

The Quantum Algorithm for Linear Systems of Equations

From the previous chapter, we have reformulated the problem of multi-asset option pricing as matrix inversion. Arising from this it is noted, that the dimension of the final matrix may be extremely large, at least growing exponentially in the number of the assets comprising the option and at least polynomially in the spatial discretization for each asset. We also note that a certain periodicity and large degree of sparseness is present in the block matrix Λ. Needless to state, an efficient method for solving large scale linear systems of equations is a crucial requirement for the realisation of the currently outlined method for multi-asset option pricing. The quantum algorithm for solutions to linear systems of equations as proposed by Harrow, Hassidim and Lloyd [22] may aid in this computationally intensive task of matrix inversion. We first outline this algorithm in the following pages as well as addressing the issues of data loading and its classical pre-processing required before deployment of the quantum algorithm. As systems of linear equations are ubiquitous in many disciplines, a quantum algorithm that can expedite this computational task would be transformative. The promise of this algorithm is its ability to return the solution to a linear system, encoded in the amplitudes of a quantum state, in a time that scales at most logarithmically in the dimension of the problem under certain assumptions. As the best classical algorithms scale linearly with dimension, exponential outperformance of classical algorithms may be possible with HHL.

However, quantum computers face obstacles that classical computers do not and the previous statement needs further qualification as noted in [1]. For example, loading a quantum representation of an arbitrary normalised vector $|b\rangle$ for which you want to solve the system $\vec{x} = \mathbf{A}^{-1}\vec{b}$ will take at least as many operations as the dimension of the problem. On the first step we have eliminated any hope of exponential speedup, even polynomial speedup. Another issue arises once a quantum state representation of the solution $|x\rangle$ has been returned. Estimating each of the amplitudes will take at least as many steps in the size of the problem or worse, extracting the value of one of the amplitudes reliably will take exponentially many measurements. In this case, deployment of the HHL algorithm for the task of matrix inversion would be much slower compared to solving with a classical algorithm.
The consequence of these two points is that quantum algorithms, especially HHL are **not** solving the exact classical problem. However, often in many computational tasks, matrix inversion is only a subroutine from which we only need to sample global information about the solution. These related but not directly dependent matrix inversion problems are where the HHL algorithm is believed to achieve quantum advantage. In order to understand the existence and or nature of quantum speedup, exact end-to-end complexity analysis from the nature of the inputs to what information needs to be extracted has to be considered. For these specific cases where optimality of quantum algorithms has been shown, quantum computers will provide immense speedup.

Although the time complexity of HHL depends logarithmically on dimension, linear dependence on other parameters such as sparsity and condition number, characteristic of the linear system to be solved, feature in the overall complexity. This implies that these parameters may have at most polylogarithmic scaling with the dimensions of the problem if an overall exponential speed up is desired. The difference matrices that arise from the Black Scholes and linear PDEs *may* offer this desirable scaling between these characteristic parameters and dimension.

4.1 Overview of algorithm

The HHL algorithm has four subroutines, quantum phase estimation (QPE), ancilla rotation (AR), postselection (PS) and inverse quantum phase estimation (QPE†). The quantum phase estimation subroutine is an established algorithm, whose most well known use is in Shor's factoring algorithm [45]. We now proceed with the description of the algorithm.

4.2 Description of algorithm

Consider the following linear systems of equations with $\mathbf{A} \in \mathbb{C}^{N\times N}$, $\vec{x}, \vec{b} \in \mathbb{C}^N$ and $\mathbf{A} = \mathbf{A}^\dagger$:

$$\mathbf{A}\vec{x} = \vec{b}$$

Consider the eigendecomposition of $\mathbf{A}$ in some eigenbasis $\{\vec{u_j}\}_{j=0,\ldots,N-1}$ along with the representations of $\vec{x}$ and $\vec{b}$:

$$\mathbf{A} = \sum_{j=0}^{N-1} \lambda_j \vec{u_j}^\dagger \vec{u_j}; \qquad \vec{x} = \sum_{j=0}^{N-1} x_j \vec{u_j}; \qquad \vec{b} = \sum_{j=0}^{N-1} b_j \vec{u_j}$$

The solution $\vec{x} = \mathbf{A}^{-1}\vec{b}$ may be expressed succinctly in this notation as:

$$\vec{x} = \mathbf{A}^{-1}\vec{b} = \sum_{j=0}^{N-1} \frac{b_j}{\lambda_j} \vec{u_j}$$

The quantum algorithm for systems of linear equations broadly emulates the above process by expressing the solution in terms of the eigenbasis of $\mathbf{A}$. There are some caveats in this approach that need to be addressed. One of the many constraints that arises when using quantum systems to solve problems is the requirement of state normalisation, namely that all physical quantum states must have unit norm. Therefore, in general, we can only initialize, manipulate and extract quantum states that are **proportional** to classical data. In the case of $\vec{b}$, its corresponding quantum state representation will be:

$$|b\rangle = \frac{1}{\sqrt{\sum_{j=0}^{N-1} |b_j|^2}} \sum_{j=0}^{N-1} b_j |u_j\rangle$$

In some sense, the initial normalisation of $\vec{b}$ has been 'forgotten' by the quantum algorithm. Often this normalisation is critical for extracting useful information from the output at the end, we will address this issue in due course. A second constraint that has to be considered with quantum algorithms is the physical insignificance of global phase. Practically speaking, this implies for example that classical data such as the vectors $\vec{v}$ and $(e^{i\theta})\vec{v}$ will have the same exact quantum state representation $|v\rangle$. For the case of real-valued data, this reduces to the existence of a fundamental indeterminacy to whether the quantum state $|v\rangle$ corresponds to $-\vec{v}$ or $\vec{v}$. Unless there is *a priori* knowledge about what the solution should look like (i.e the solution corresponds to some physical process which is necessarily positively valued, etc), we must remain aware of this uncertainty. For a Hermitian matrix $\mathbf{A} \in \mathbb{C}^{N\times N}$, and a suitably normalised vector $\vec{b} \in \mathbb{C}^N$, the quantum algorithm returns another suitably normalised vector $\vec{x} \in \mathbb{C}^N$ such that:

$$\vec{x} = \mathbf{A}^{-1}\vec{b}$$

We can express the quantum state representation of $|b\rangle$ in the eigenbasis of $\mathbf{A}$ $\{|u_j\rangle\}_{j=0,1,2,.,N-1}$ with the corresponding eigenvalues$\{\lambda_j\}_{j=0,1,2,.,N-1}$ as:

$$|b\rangle = \sum_{j=0}^{N-1} b_j|u_j\rangle \quad ; \quad \mathbf{A}\,|u_j\rangle = \lambda_j|u_j\rangle \tag{4.1}$$

Upon successful execution of the algorithm, the quantum representation of the solution $\vec{x}$ will be stored in the final register:

$$|x\rangle = \sum_{j=0}^{N-1} \frac{b_j}{\lambda_j}|u_j\rangle \tag{4.2}$$

This is the solution to the systems of the linear equations as:

$$\mathbf{A}|x\rangle = \mathbf{A}\left(\sum_{j=0}^{N-1} \frac{b_j}{\lambda_j}|u_j\rangle\right) \Rightarrow \sum_{j=0}^{N-1} \frac{b_j}{\lambda_j}\mathbf{A}|u_j\rangle \Rightarrow \sum_{j=0}^{N-1} \frac{b_j}{\lambda_j}\lambda_j|u_j\rangle \Rightarrow \sum_{j=0}^{N-1} b_j|u_j\rangle = |b\rangle \tag{4.3}$$

4.2.1 Quantum Phase Estimation

The original quantum phase estimation algorithm (QPE)[36] is a key subroutine in many quantum algorithms and features predominantly in the HHL algorithm.

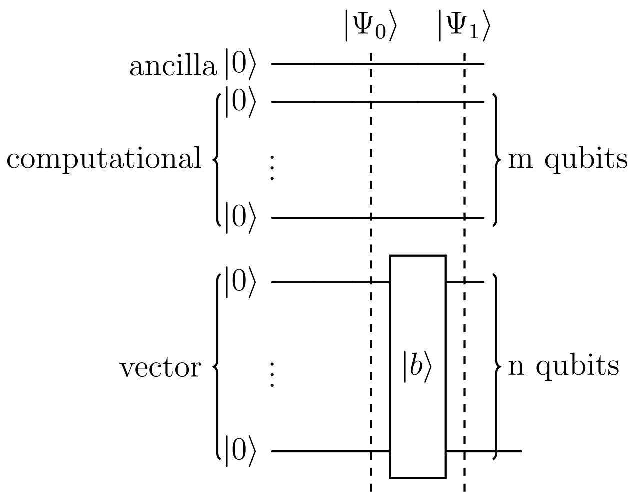


Figure 4.1: Quantum circuit diagram of initial state loading

The HHL quantum algorithm requires three registers, one to store the quantum representation of $|b\rangle$, another register to store the binary approximation of the eigenvalues $\{\lambda_j\}_{j=0,1,2,.,,N-1}$ that have been extracted during quantum phase estimation, and a final ancilla register upon which the ancilla rotation subroutine will act. The first step is loading the state $|b\rangle$ into the first register.

$$|\Psi_1\rangle = |b\rangle_v |0\rangle_c^{\otimes m} |0\rangle_a \quad ; \quad |b\rangle = \frac{1}{\|\vec{b}\|}\sum_{j=0}^{N-1} b_j \,|u_j\rangle$$

The quantum circuit diagrams such as figure (4.1) are drawn with the Quantikz2 package [33]. Here we denote the subscripts v, c and a as corresponding to the *vector* register, the *computational* register and the *ancilla* register respectively (which we will sometimes refer to as the v-register, c-register and a-register respectively). We will work throughout in the standard computational basis.

Conditional unitary evolution

The algorithm proceeds with the first state of QPE which is placing the *computational* register in an equal superposition of basis states:

$$|\Psi_2\rangle = \frac{1}{\sqrt{M}}\sum_{k=0}^{M-1} |b\rangle|k\rangle|0\rangle = \frac{1}{\sqrt{M}\|\vec{b}\|}\sum_{j=0}^{N-1}\sum_{k=0}^{M-1} b_j|u_j\rangle|k\rangle|0\rangle \tag{4.4}$$

Throughout the description of this algorithm, we will use binary notation to describe the computational basis for multi-qubit systems. In the case of the two-qubit system this would correspond to:

$$
\begin{aligned}
|0\rangle|0\rangle &\leftrightarrow |2^1(0)+2^0(0)\rangle = |0\rangle \\
|0\rangle|1\rangle &\leftrightarrow |2^1(0)+2^0(1)\rangle = |1\rangle \\
|1\rangle|0\rangle &\leftrightarrow |2^1(1)+2^0(0)\rangle = |2\rangle \\
|1\rangle|1\rangle &\leftrightarrow |2^1(1)+2^0(1)\rangle = |3\rangle
\end{aligned}
$$

In general, the basis state $|k_1\rangle|k_2\rangle .. |k_{m-1}\rangle|k_m\rangle$ where each $k_i \in \{0,1\}$ will correspond to the state $|k\rangle$ where k is defined as:

$$
k = \sum_{i=1}^{m} k_i 2^{m-i} \tag{4.5}
$$

To see how state (4.4) is produced, we can directly compute the action of the quantum circuit:

$$
\begin{aligned}
|\Psi_2\rangle &= H^{\otimes m}\left(|b\rangle|0\rangle^{\otimes m}|0\rangle\right) \\
&= |b\rangle(H|0\rangle)^{\otimes m}|0\rangle \\
&= |b\rangle\left(\frac{1}{\sqrt{2}}(|0\rangle+|1\rangle)\right)^{\otimes m}|0\rangle \\
&= \frac{1}{\sqrt{2^m}}|b\rangle\bigotimes_{j=1}^{m}\left(\sum_{k_j=0}^{1}|k_j\rangle\right)|0\rangle \\
&= \frac{1}{\sqrt{M}}|b\rangle\left(\sum_{k_m=0}^{1}\cdots\sum_{k_1=0}^{1}\bigotimes_{j=1}^{m}|k_j\rangle\right)|0\rangle
\end{aligned}
$$

Recalling the definition of (4.5) yields:

$$
\begin{aligned}
|\Psi_2\rangle &= \frac{1}{\sqrt{M}}|b\rangle\left(\sum_{k=0}^{2^m-1}|k\rangle\right)|0\rangle \\
&= \frac{1}{\sqrt{M}}\sum_{k=0}^{M-1}|b\rangle|k\rangle|0\rangle
\end{aligned}
$$

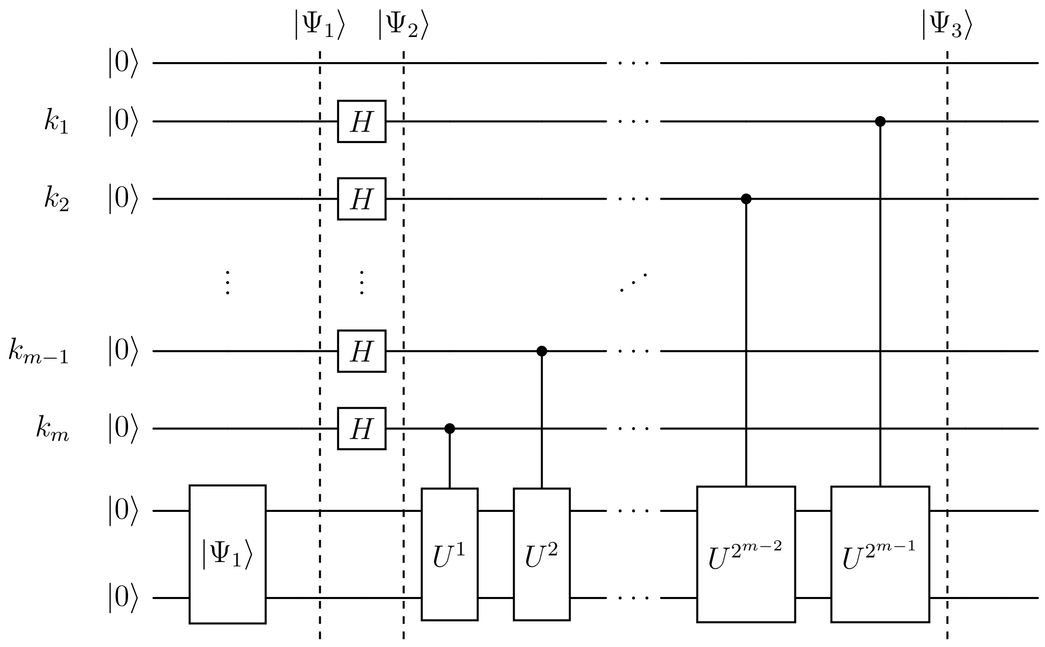


Figure 4.2: Quantum circuit diagram of conditional unitary evolution

Finally, inserting the eigendecomposition of $|b\rangle$ (4.1), we arrive at the final form of (4.4):

$$|\Psi_2\rangle = \frac{1}{\sqrt{M}\|\vec{b}\|}\sum_{k=0}^{M-1}\left(\sum_{j=0}^{N-1} b_j\,|u_j\rangle\right)|k\rangle\,|0\rangle$$

$$|\Psi_2\rangle = \frac{1}{\sqrt{M}\|\vec{b}\|}\sum_{j=0}^{N-1}\sum_{k=0}^{M-1} b_j|u_j\rangle|k\rangle|0\rangle$$

The third step of the algorithm is evolution of the eigenvectors $|u_j\rangle$ conditioned on the state in the c-register $|k\rangle$. The unitary operator that will drive this evolution is given by:

$$U = \exp\left(\frac{i\mathbf{A}t}{M}\right)$$

However, quantum phase estimation returns a binary approximation θ of the eigenvalues $|\lambda\rangle$ that correspond to the eigenvalue equation:

$$U|\lambda\rangle = \exp(2\pi i\theta)|\lambda\rangle = \exp\left(\frac{2\pi i\tilde{\theta}}{M}\right)|\lambda\rangle$$

where $0 \leq \theta < 1$, $\tilde{\theta} \in \{0, 1, ..., 2^m - 1\}$ and is the actual quantity that we would potentially measure in the c-register immediately after QPE ($\frac{\tilde{\theta}}{M}$ is the best m-bit approximation of θ). So in order for QPE to be used for matrix inversion, we need to be able to translate from phase to eigenvalues unambiguously. Returning to the algorithm, after conditional evolution with the unitary $U = e^{iAt/M}$ with t chosen to satisfy the above criterion, the quantum state will be:

$$|\Psi_3\rangle = \frac{1}{\sqrt{M}\|\vec{b}\|} \sum_{j=0}^{N-1} \sum_{k=0}^{M-1} b_j \exp\left(\frac{ik\lambda_j t}{M}\right) |u_j\rangle |k\rangle |0\rangle \tag{4.6}$$

To see this from the circuit model, we sequentially apply the controlled gates $C - U^{2^{(\cdot)}}$ to $|\Psi_2\rangle$ and so forth. We apply the first controlled unitary $C - U^{2^0}$:

$$\begin{aligned}\left(C - U^{2^0}\right) |\Psi_2\rangle &= \left(C - U^{2^0}\right) \left(\frac{1}{\sqrt{M}\|\vec{b}\|} \sum_{k=0}^{M-1} \sum_{j=0}^{N-1} b_j |u_j\rangle |k\rangle |0\rangle\right) \\ &= \frac{1}{\sqrt{M}\|\vec{b}\|} \sum_{k=0}^{M-1} \sum_{j=0}^{N-1} b_j \left(\left(C - U^{2^0}\right) |u_j\rangle |k\rangle\right) |0\rangle \end{aligned} \tag{4.7}$$

The nature of a controlled gate is that the associated unitary is applied to one qubit if and only if the other qubit is in the $|1\rangle$ state, otherwise both states are left unaffected. For the case at hand the unitary U^{2^0} is applied if the control qubit $|k_m\rangle$, which is one of the qubits that comprise the actual quantum state $|k\rangle = |k_1\rangle |k_2\rangle .. |k_m\rangle$, is in the $|1\rangle$ state. This operation can be represented in a condensed expression as:

$$\left(C - U^{2^0}\right) |u_j\rangle |k_m\rangle = \left(U^{k_m 2^0} |u_j\rangle\right) |k_m\rangle$$

Returning into (4.7) and inserting the result yields:

$$\begin{aligned}\left(C - U^{2^0}\right) |\Psi_2\rangle &= \frac{1}{\sqrt{M}\|\vec{b}\|} \sum_{k=0}^{M-1} \sum_{j=0}^{N-1} b_j \left(\left(U^{k_m 2^0} |u_j\rangle\right) |k\rangle\right) |0\rangle \\ |\Psi_2'\rangle &= \frac{1}{\sqrt{M}\|\vec{b}\|} \sum_{k=0}^{M-1} \sum_{j=0}^{N-1} b_j U^{k_m 2^0} |u_j\rangle |k\rangle |0\rangle \end{aligned}$$

where we denote this intermediate state after applying $C - U^{2^0}$ as $|\Psi_2'\rangle$. Repeating this process above with the next controlled unitary in the circuit $C - U^{2^1}$ to $|\Psi_2'\rangle$

yields:

$$\left(C-U^{2^1}\right)|\Psi_2'\rangle = \frac{1}{\sqrt{M}\|\vec{b}\|}\sum_{k=0}^{M-1}\sum_{j=0}^{N-1} b_j\left(C-U^{2^1}\right)\left(\left(U^{k_m 2^0}|u_j\rangle\right)|k\rangle\right)|0\rangle$$
$$= \frac{1}{\sqrt{M}\|\vec{b}\|}\sum_{k=0}^{M-1}\sum_{j=0}^{N-1} b_j\left(\left(U^{k_{m-1}2^1}U^{k_m 2^0}|u_j\rangle\right)|k\rangle\right)|0\rangle$$
$$|\Psi_2''\rangle = \frac{1}{\sqrt{M}\|\vec{b}\|}\sum_{k=0}^{M-1}\sum_{j=0}^{N-1} b_j U^{k_{m-1}2^1}U^{k_m 2^0}|u_j\rangle|k\rangle|0\rangle$$

Repeating this for all remaining controlled unitaries for this stage of the circuit yields:

$$|\Psi_3\rangle = \frac{1}{\sqrt{M}\|\vec{b}\|}\sum_{k=0}^{M-1}\sum_{j=0}^{N-1} b_j\prod_{q=1}^{m} U^{k_q 2^{m-q}}|u_j\rangle|k\rangle|0\rangle$$
$$= \frac{1}{\sqrt{M}\|\vec{b}\|}\sum_{k=0}^{M-1}\sum_{j=0}^{N-1} b_j U^{\left(\sum_{q=1}^{m} k_q 2^{m-q}\right)}|u_j\rangle|k\rangle|0\rangle$$
$$= \frac{1}{\sqrt{M}\|\vec{b}\|}\sum_{k=0}^{M-1}\sum_{j=0}^{N-1} b_j U^{k}|u_j\rangle|k\rangle|0\rangle \tag{4.8}$$

As $U = \exp\left(\frac{i\mathbf{A}t}{M}\right)$, $U^k = \exp\left(\frac{ik\mathbf{A}t}{M}\right)$. Furthermore, as $|u_j\rangle$ is an eigenvector of $\mathbf{A}$ with eigenvalue λ_j, $|u_j\rangle$ will also be an eigenvector of U^k with eigenvalue $\exp\left(\frac{ik\lambda_j t}{M}\right)$.

$$U^k|u_j\rangle = \exp\left(\frac{ik\lambda_j t}{M}\right)|u_j\rangle \tag{4.9}$$

Inserting into (4.8), equation (4.6) is recovered:

$$|\Psi_3\rangle = \frac{1}{\sqrt{M}\|\vec{b}\|}\sum_{j=0}^{N-1}\sum_{k=0}^{M-1} b_j\exp\left(\frac{ik\lambda_j t}{M}\right)|u_j\rangle|k\rangle|0\rangle$$

Inverse Quantum Fourier Transform

The next step in the algorithm is applying the Inverse Quantum Fourier transform [10] on the *c*-register. The Inverse Quantum Fourier Transform is defined [36] as:

$$|k\rangle \xrightarrow{\text{IQFT}} \frac{1}{\sqrt{M}}\sum_{l=0}^{M-1}\exp\left(-\frac{2\pi ikl}{M}\right)|l\rangle \tag{4.10}$$

To see how the circuit model enacts this transformation, we outline the derivation from [36]. The gate R_p, known as a *phase-shift* gate, transforms the basis states

$|0\rangle, |1\rangle$ as:

$$R_p |0\rangle = |0\rangle \qquad R_p |1\rangle = \exp\left(-\frac{2\pi i}{2^p}\right) |1\rangle$$

The controlled version of the phase shift gate where the second qubit is the control qubit can be represented succinctly as:

$$\begin{aligned}(C - R_p) |0\rangle |k_j\rangle &= |0\rangle |k_j\rangle \\ (C - R_p) |1\rangle |k_j\rangle &= \exp\left(-\frac{2\pi i k_j}{2^p}\right) |1\rangle |k_j\rangle\end{aligned}$$

Similarly the action of the Hadamard gate can also be represented succinctly as:

$$H |k_j\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + (-1)^{k_j} |1\rangle\right) = \frac{1}{\sqrt{2}} \left(|0\rangle + \exp\left(-\frac{2\pi i k_j}{2}\right) |1\rangle\right)$$

$$|\Psi_3'\rangle = (H |k_1\rangle) |k_1\rangle |k_2\rangle ... |k_m\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + \exp\left(-\frac{2\pi i k_1}{2}\right) |1\rangle\right) |k_2\rangle ... |k_m\rangle$$

Applying the $C - R_2$ gate to $|\Psi_3'\rangle$ yields:

$$\begin{aligned}|\Psi_3''\rangle = (C - R_2) |\Psi_3'\rangle &= \frac{1}{\sqrt{2}} ((C - R_2) |0\rangle |k_2\rangle) |k_3\rangle .. |k_m\rangle \\ &\quad + \frac{1}{\sqrt{2}} \exp\left(-\frac{2\pi i k_1}{2}\right) ((C - R_2) |1\rangle |k_2\rangle) |k_3\rangle ... |k_m\rangle \\ |\Psi_3''\rangle &= \frac{1}{\sqrt{2}} |0\rangle |k_2\rangle |k_3\rangle ... |k_m\rangle \\ &\quad + \frac{1}{\sqrt{2}} \exp\left(-\frac{2\pi i k_1}{2}\right) \exp\left(-\frac{2\pi i k_2}{2^2}\right) |1\rangle |k_2\rangle |k_3\rangle ... |k_m\rangle \\ |\Psi_3''\rangle &= \frac{1}{\sqrt{2}} \left(|0\rangle + \exp\left(-\frac{2\pi i}{2^2} \left(2^1 k_1 + 2^0 k_2\right)\right)\right) |k_2\rangle ... |k_m\rangle\end{aligned}$$

Repeating this process up to $C - R_m$ transforms the first qubit into the state:

$$\begin{aligned}|\bar{\Psi}_3\rangle &= \frac{1}{\sqrt{2}} \left(|0\rangle + \exp\left(-\frac{2\pi i}{2^m} \left(\sum_{q=1}^{m} k_q 2^{m-q}\right)\right)\right) |k_2\rangle ... |k_m\rangle \\ |\bar{\Psi}_3\rangle &= \frac{1}{\sqrt{2}} \left(|0\rangle + \exp\left(-\frac{2\pi i k}{2^m}\right)\right) |k_2\rangle ... |k_m\rangle\end{aligned}$$

Repeating this process for the remaining qubits yields the state:

$$|\Psi_4\rangle = \frac{1}{\sqrt{M}} \bigotimes_{z=0}^{m-1} \left(|0\rangle + \exp\left(-\frac{2\pi i k}{2^{m-z}}\right) |1\rangle\right)$$

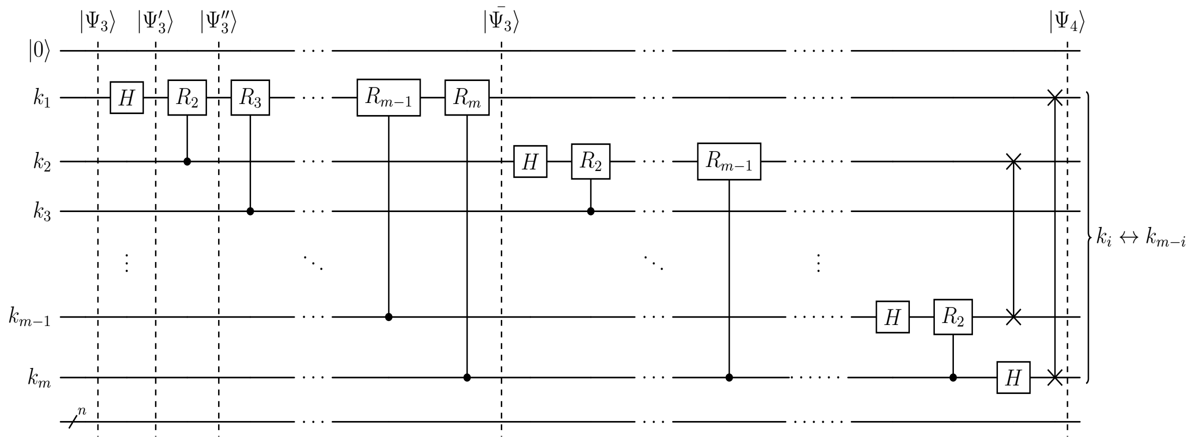


Figure 4.3: Quantum circuit diagram of the Inverse Quantum Fourier Transform (IQFT)

Finally, as a matter of convention, we perform a series of pairwise swaps (swapping the (k_i) qubit with the (k_{m-i}) qubit) of all the qubits to bring the state into the standard form that will correspond to the IQFT:

$$|\Psi_4\rangle = \frac{1}{\sqrt{M}} \bigotimes_{z=1}^{m} \left(|0\rangle + \exp\left(-\frac{2\pi i k}{2^z} \right) |1\rangle \right) \tag{4.11}$$

To conclude this section, we now show how (4.10) is equivalent to (4.11):

$$\begin{aligned}
|k\rangle \xrightarrow{\text{IQFT}} &= \frac{1}{\sqrt{M}} \sum_{l=0}^{M-1} \exp\left(-\frac{2\pi i k l}{2^m} \right) |l\rangle \\
&= \frac{1}{\sqrt{M}} \sum_{l_1=0}^{1} ... \sum_{l_m=0}^{1} \exp\left(-\frac{2\pi i k}{2^m} \left(\sum_{z=1}^{m} l_z 2^{m-z} \right) \right) |l_1\rangle ... |l_m\rangle \\
&= \frac{1}{\sqrt{M}} \sum_{l_1=0}^{1} ... \sum_{l_m=0}^{1} \exp\left(-2\pi i k \left(\sum_{z=1}^{m} l_z 2^{-z} \right) \right) |l_1\rangle ... |l_m\rangle \\
&= \frac{1}{\sqrt{M}} \sum_{l_1=0}^{1} ... \sum_{l_m=0}^{1} \left(\prod_{z=1}^{m} \exp\left(-\frac{2\pi i k l_z}{2^z} \right) \right) \bigotimes_{z=1}^{m} |l_z\rangle \\
&= \frac{1}{\sqrt{M}} \sum_{l_1=0}^{1} ... \sum_{l_m=0}^{1} \bigotimes_{z=1}^{m} \left(\exp\left(-\frac{2\pi i k l_z}{2^z} \right) |l_z\rangle \right) \\
&= \frac{1}{\sqrt{M}} \bigotimes_{z=1}^{m} \left(\sum_{l_z=0}^{1} \exp\left(-\frac{2\pi i k l_z}{2^z} \right) |l_z\rangle \right) \\
&= \frac{1}{\sqrt{M}} \bigotimes_{z=1}^{m} \left(|0\rangle + \exp\left(-\frac{2\pi i k}{2^z} \right) |1\rangle \right) = |\Psi_4\rangle
\end{aligned}$$

Returning to the overall algorithm, applying IQFT to the c-register yields:

$$|\Psi_4\rangle = \frac{1}{\sqrt{M}\|\vec{b}\|}\sum_{j=0}^{N-1}\sum_{k=0}^{M-1} b_j \exp\left(\frac{ik\lambda_j t}{M}\right)|u_j\rangle\left(\text{IQFT}\,|k\rangle\right)|0\rangle$$

$$|\Psi_4\rangle = \frac{1}{\sqrt{M}\|\vec{b}\|}\sum_{j=0}^{N-1}\sum_{k=0}^{M-1} b_j \exp\left(\frac{ik\lambda_j t}{M}\right)|u_j\rangle \otimes \left(\frac{1}{\sqrt{M}}\sum_{l=0}^{M-1}\exp\left(\frac{-2\pi ikl}{M}\right)|l\rangle\right)|0\rangle$$

$$|\Psi_4\rangle = \frac{1}{M\|\vec{b}\|}\sum_{j=0}^{N-1}\sum_{k=0}^{M-1}\sum_{l=0}^{M-1} b_j \exp\left(\frac{ik\lambda_j t}{M}\right)\exp\left(\frac{-2\pi ikl}{M}\right)|u_j\rangle|l\rangle|0\rangle$$

$$|\Psi_4\rangle = \frac{1}{\|\vec{b}\|}\sum_{j=0}^{N-1}\sum_{l=0}^{M-1} b_j \underbrace{\left(\frac{1}{M}\sum_{k=0}^{M-1}\exp\left(\frac{ikt}{M}\left(\lambda_j - \frac{2\pi l}{t}\right)\right)\right)}_{\gamma_j(l)}|u_j\rangle|l\rangle|0\rangle \tag{4.12}$$

If we examine (4.12), we see that possible eigenvalues we could potentially extract from QPE are given by $\frac{2\pi l}{t}$ and the actual eigenvalues we are trying to approximate is λ_j. To see this more clearly, we proceed to analyse the term $\gamma_j(l)$ and specifically $|\gamma_j(l)|^2$:

$$\left|\frac{1}{M\|\vec{b}\|}\sum_{k=0}^{M-1}\exp\left(\frac{ikt}{M}\left(\lambda_j - \frac{2\pi l}{t}\right)\right)\right|^2 = \left|\frac{1-\exp\left(ikt(\lambda_j - \frac{2\pi l}{t})\right)}{1-\exp\left(\frac{ikt}{M}\left(\lambda_j - \frac{2\pi l}{t}\right)\right)}\right|^2$$

For the case of the exact representation of the phases associated with the eigenvalue λ_j, we get:

$$\lambda_j = \frac{2\pi\tilde{l}_j}{t}, \quad \tilde{l}_j \in \{0, 1, ..., M-1\}$$

In the case of exact representation of the phases, the function $\gamma_j(l)$ behaves like a Kronecker delta function $\delta(x, y)$. To see this, if $l = \tilde{l}_j \Leftrightarrow (\lambda_j - \frac{2\pi\tilde{l}_j}{t} = 0)$ we have:

$$\Rightarrow |\gamma_j(l)|^2 = |\gamma_j(\tilde{l}_j)|^2 = \left|\frac{1}{M}\sum_{k=0}^{M-1}\exp\left(\frac{ikt}{M}\left(\lambda_j - \frac{2\pi\tilde{l}_j}{t}\right)\right)\right|^2$$
$$= \left|\frac{1}{M}\sum_{k=0}^{M-1}\exp\left(\frac{ikt}{M}(0)\right)\right|^2 = \left|\frac{1}{M}\sum_{k=0}^{M-1}1\right|^2 = 1$$

However, if $l \neq \tilde{l}_j$

$$\begin{aligned}
&\rightarrow \quad l = \tilde{l}_j + a; \quad a \in \mathbb{Z} \\
&\rightarrow \quad \lambda_j - \frac{2\pi l}{t} = \lambda_j - \frac{2\pi(\tilde{l}_j + a)}{t} = \underbrace{\left(\lambda_j - \frac{2\pi \tilde{l}_j}{t}\right)}_{=0} - \frac{2\pi a}{t} \\
&\rightarrow \quad = ikt\left(\lambda_j - \frac{2\pi l}{t}\right) = ikt\left(-\frac{2\pi a}{t}\right) = -2\pi ika
\end{aligned}$$

Returning to the definition of $|\gamma_j(l)|^2$, we see that:

$$\begin{aligned}
\Rightarrow |\gamma_j(l)|^2 = \left| \frac{1 - \exp\left(ikt(\lambda_j - \frac{2\pi l}{t})\right)}{1 - \exp\left(\frac{ikt}{M}\left(\lambda_j - \frac{2\pi l}{t}\right)\right)} \right|^2 &= \left| \frac{1 - \exp(-2\pi ika)}{1 - \exp\left(\frac{ikt}{M}\left(\lambda_j - \frac{2\pi l}{t}\right)\right)} \right|^2 \\
&= \left| \frac{1 - 1}{1 - \exp\left(\frac{ikt}{M}\left(\lambda_j - \frac{2\pi l}{t}\right)\right)} \right|^2 = 0
\end{aligned}$$

Summing over l extracts $\tilde{l}_j$ and stores the result in the c-register and concludes the final step of QPE:

$$|\Psi_4\rangle = \frac{1}{||\vec{b}||} \sum_{j=0}^{N-1} \sum_{l=0}^{M-1} b_j \delta(\tilde{l}_j, l) |u_j\rangle |l\rangle |0\rangle \rightarrow \frac{1}{||\vec{b}||} \sum_{j=0}^{N-1} b_j |u_j\rangle |\tilde{l}_j\rangle |0\rangle$$

So we can see that from this analysis that $\tilde{l}_j$ is an approximation of the quantity $\frac{t\lambda_j}{2\pi}$.

$$g_{\phi \rightarrow \lambda}(l) := \frac{2\pi l}{t}$$

Naively embedding the phase from the eigenvalues, such as $\theta_j = e^{i\lambda_j}$ (corresponding to the unitary $U = e^{iA}$ in QPE), will imply eigenvalues outside the range $[0, 2\pi)$ cannot be reliably distinguished (only up to modulo(2π)). To circumvent this issue, the unitary $U = e^{iAt/M}$ is implemented instead, with a predetermined value t to compress' the spectrum so that we can faithfully extract the eigenvalues. We now have two free parameters t and M to assign values to. Firstly, there are two cases for the spectrum of a Hermitian matrix $\mathbf{A}$. For the case that all the eigenvalues share the same sign, we can spread out the phases we assign to eigenvalues amongst the entire complex unit circle. For the second case whereby the eigenvalues do not all share the same sign, we have to effectively half the complex unit circle dedicating the upper half of the unit circle $(e^{i\phi}, \phi \in [0, \pi])$

to the positive eigenvalues and the bottom half ($e^{i\phi}, \phi \in [\pi, 2\pi]$) to the negative eigenvalues. The latter description will work for either case so we outline that procedure below. The first step in addressing the values of t and M is determining the spectrum of $\mathbf{A}$, $\sigma(\mathbf{A})$. For convenience, we define the following two quantities:

$$\lambda_{\text{max}} = \max_i \{|\lambda_i| : \lambda_i \in \sigma(\mathbf{A})\}$$

$$\lambda_{\text{min}} = \min_i \{|\lambda_i| : \lambda_i \in \sigma(\mathbf{A})\}$$

This allows to bound $\sigma(\mathbf{A})$ as:

$$\sigma(A) \subseteq [-\lambda_{\text{max}}, -\lambda_{\text{min}}] \cup [\lambda_{\text{min}}, \lambda_{\text{max}}]$$

If we consider re-scaling the whole linear system to be solved as the original authors in [22] assume, (i.e. solve $\frac{1}{\lambda_{\text{max}}}\mathbf{A}\vec{x} = \frac{1}{\lambda_{\text{max}}}\vec{b}$ rather than $\mathbf{A}\vec{x} = \vec{b}$), this will effectively compress the spectrum into the new interval given by:

$$\begin{aligned} \sigma(\mathbf{A}) \ &\subseteq \left[-1, -\tfrac{\lambda_{\text{min}}}{\lambda_{\text{max}}}\right] \bigcup \left[\tfrac{\lambda_{\text{min}}}{\lambda_{\text{max}}}, 1\right] \\ &= \left[-1, -\tfrac{1}{\kappa}\right] \bigcup \left[\tfrac{1}{\kappa}, 1\right] \end{aligned}$$

The parameter κ refers to the *condition number* of the matrix $\mathbf{A}$ and characterizes the numerical stability of the linear system to be solved [49]. It is defined formally as:

$$\kappa := \max_{s_i, s_j \in s(\mathbf{A})} \frac{s_i}{s_j}$$

where $s(\mathbf{A})$ refers to the singular values of the matrix $\mathbf{A}$. For the case of a Hermitian matrix, the above definition reduces to:

$$\kappa = \frac{\lambda_{\text{max}}}{\lambda_{\text{min}}}$$

Going forward we assume this scaling has been done as a pre-processing step. Depending on the matrix, there may exist lower bounds or analytic expressions for the spectral norm of the matrix $\|\mathbf{A}\|$ which is equal to λ_{max} in the case that $\mathbf{A}$ is Hermitian. In the following steps, we slightly modify the approach of [39]. In order to ensure an injective mapping from the phases to the eigenvalues, we split the spectrum up into its positive and negative components. We take

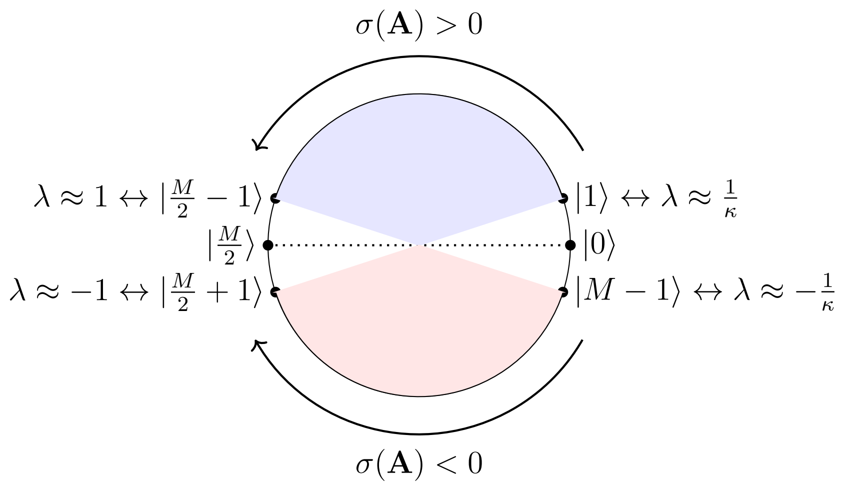

$\sigma(\mathbf{A}) > 0$
$\lambda \approx 1 \leftrightarrow |\frac{M}{2} - 1\rangle$
$|1\rangle \leftrightarrow \lambda \approx \frac{1}{\kappa}$
$|\frac{M}{2}\rangle$
$|0\rangle$
$\lambda \approx -1 \leftrightarrow |\frac{M}{2} + 1\rangle$
$|M - 1\rangle \leftrightarrow \lambda \approx -\frac{1}{\kappa}$
$\sigma(\mathbf{A}) < 0$

Figure 4.4: Mapping of phases to eigenvalues with the blue shaded section corresponding to eigenvalues $\in [\frac{1}{\kappa}, 1]$ and red for phases corresponding to eigenvalues $\in [-1, -\frac{1}{\kappa}]$. The basis states $|0\rangle$ and $|\frac{M}{2}\rangle$ are excluded in QPE.

the case of positive eigenvalues first. If $\lambda_j \in [\frac{1}{\kappa}, 1]$, we require the associated phase to lie somewhere within the upper part of the complex unit circle, more specifically that it lies somewhere between the phases corresponding to $|1\rangle$ and $|\frac{M}{2} - 1\rangle$ of the Fourier basis states of the computational register. This implies the eigenvalues associated to $|1\rangle$ and $|\frac{M}{2} - 1\rangle$ should approximately be equal to $\frac{1}{\kappa}$ and 1 respectively:

$$g^{+}_{\phi\to\lambda}(1) = \frac{2\pi}{t}(1) \approx \frac{1}{\kappa} \qquad g^{+}_{\phi\to\lambda}\left(\frac{M}{2} - 1\right) = \frac{2\pi}{t}\left(\frac{M}{2} - 1\right) \approx 1$$

In order to ensure that all eigenvalues are contained within the image of $g^{+}_{\phi\to\lambda}(l)$, we demand that the smallest positive eigenvalue that can be estimated is less than the actual smallest positive eigenvalue, namely $\frac{1}{\kappa}$:

$$\frac{2\pi}{t} \leq \frac{1}{\kappa} \tag{4.13}$$

We also require that the largest eigenvalue that can be estimated be is greater than the largest actual eigenvalue which is one 1. However, we do not want to verge into the lower half of the complex unit circle so we demand instead that the largest possible eigenvalue is greater than a value slightly less than one, namely:

$$\frac{2\pi}{t}\left(\frac{M}{2} - 1\right) \geq \frac{2}{M}\left(\frac{M}{2} - 1\right) \tag{4.14}$$

Manipulating (4.13) and (4.14) yields:

$$t \geq 2\pi\kappa \qquad t \leq \pi M \tag{4.15}$$

Making these inequalities (4.15) to be strict equalities, we get the following values for t and M:

$$t = 2\pi\kappa \qquad M = 2\kappa$$

For the case of negative eigenvalues, the mapping from phases to eigenvalues is given by:

$$g^{-}_{\phi\to\lambda}(l) := \frac{2\pi}{t}\,(l - M)$$

To see this we notice that if $\lambda < 0$, the associated phase $\frac{\lambda t}{M}$ will be negative. We can circumvent this issue by adding 2π to the phase and then extracting the eigenvalue as:

$$\begin{aligned}
\exp\left(i\left(\frac{\lambda t}{M}\right)\right) = \exp\left(i\left(\frac{\lambda t}{M}\right) + 2\pi i\right) &= \exp\left(\frac{2\pi l}{M}\right) \\
\frac{\lambda t}{M} + \frac{2\pi M}{M} &= \frac{2\pi l}{M} \\
\frac{\lambda t}{M} &= \frac{2\pi}{M}\,(l - M) \\
\lambda &= \frac{2\pi}{t}\,(l - M)
\end{aligned}$$

The mapping from the phases to eigenvalues $g(l)$ is given by:

$$g(l) = \begin{cases} g^{+}_{\phi\to\lambda}(l) & 0 < l < \frac{M}{2} \\ g^{-}_{\phi\to\lambda}(l) & \frac{M}{2} < l < M \end{cases} = \begin{cases} \frac{2\pi l}{t} & 0 < l < \frac{M}{2} \\ \frac{2\pi}{t}(l - M) & \frac{M}{2} < l < M \end{cases}$$

To understand how this works, say that the spectrum of an operator $\sigma(\mathbf{A}) = \{-1, -\frac{1}{2}, -\frac{1}{4}, \frac{1}{3}, \frac{3}{4}, 1\}$ (where we already have implicitly scaled the linear system so that $\lambda_{\max}(A) = 1$). As $\kappa(\mathbf{A}) = 4$, we get that $t = 2\pi(4) = 8\pi$ and $M = 2(4) = 8$. In case that M is not a power of 2, we can just take the next power of 2 above it:

$$M = 2^{\lceil \log_2(2\kappa) \rceil}$$

where $\lceil(.)\rceil$ denotes the ceiling function. So in this example $g(l)$ with $M = 8$, $t = 8\pi$ takes the form:

$$g(l) = \begin{cases} \frac{2\pi l}{8\pi} & 0 < l < \frac{8}{2} \\ \frac{2\pi}{8\pi}(l-8) & \frac{8}{2} < l < 16 \end{cases} \quad \rightarrow \quad g(l) = \begin{cases} \frac{l}{4} & 0 < l < 4 \\ \frac{l-8}{4} & 4 < l < 8 \end{cases}$$

For final illustration of this point, we will examine how a positive and negative eigenvalue can be extracted. Take the eigenvalue $\lambda = \frac{1}{4}$ with the corresponding unitary $e^{i\mathbf{A}t/M} = e^{\frac{8\pi i}{8}\mathbf{A}}$. This will correspond to a phase of $e^{\pi i(\frac{1}{4})} \implies e^{2\pi i(\frac{1}{8})}$. Upon successful QPE, the basis state $|1\rangle$ will be measured with high probability, indicating $l = 1$ and that the eigenvalue is $g(1) = \frac{1}{4}$. Thus, the correct eigenvalue has been recovered. For the case of negative eigenvalues, take the eigenvalue $\lambda = -\frac{1}{2}$. This will correspond to a phase of $e^{i\pi(-\frac{1}{2})} = e^{2\pi i(\frac{-2}{8})} \implies e^{2\pi i(\frac{6}{8})}$ (The Fourier basis states are strictly positive, so the phase that will be measured must also be positive too). Upon successful QPE, the basis state $|6\rangle$ will be measured with high probability, indicating $l = 6$ and the eigenvalue is $g(6) = \frac{6-8}{4} = -\frac{1}{2}$. So again, the correct eigenvalue has been recovered. However, the eigenvalue $\frac{1}{3}$ does not have an exact representation above so there would be unavoidable error in determining the eigenvalue in that instance.

4.2.2 Ancilla rotation

The second central subroutine of the HHL algorithm is ancilla rotation which performs the following map:

$$\frac{1}{||\vec{b}||}\sum_{j=0}^{N-1} b_j|u_j\rangle|\tilde{l}_j\rangle|0\rangle \xrightarrow{\text{Ancilla rotation}} \frac{1}{||\vec{b}||}\sum_{j=0}^{N-1} b_j|u_j\rangle|\tilde{l}_j\rangle\left(\sqrt{1-f(\tilde{l}_j)^2}|0\rangle + f(\tilde{l}_j)|1\rangle\right)$$

For the HHL algorithm, the function $f(x)$ is chosen such that:

$$f(x) = \frac{C}{g(x)}$$

$$f(\tilde{l}_j) = \frac{C}{g(\tilde{l}_j)} = \frac{C}{\lambda_j}$$

The normalisation constant C is chosen such that $\frac{C}{\min|\sigma(\mathbf{A})|} \leq 1$. If the overall matrix is scaled such that $\max|\sigma(\mathbf{A})| = 1$, C can be chosen such that $C \leq \frac{1}{\kappa}$.

After this transformation, the quantum state is given by:

$$|\Psi_5\rangle = \frac{1}{\|\vec{b}\|}\sum_{j=0}^{N-1} b_j|u_j\rangle|\tilde{l}_j\rangle\left(\sqrt{1-\frac{C^2}{\lambda_j^2}}|0\rangle + \frac{C}{\lambda_j}|1\rangle\right) \tag{4.16}$$

After postselection for the state $|1\rangle$ on the ancilla register, we can see the essence of HHL emerge (referring back to (4.2)):

$$|\Psi_5\rangle = \frac{1}{\|\vec{b}\|}\sum_{j=0}^{N-1} b_j|u_j\rangle|\tilde{l}_j\rangle\left(\sqrt{1-\frac{C^2}{\lambda_j^2}}|0\rangle + \frac{C}{\lambda_j}|1\rangle\right)$$
$$\xrightarrow{\text{postselection}} \frac{1}{\mathcal{N}}\sum_{j=0}^{N-1}\frac{Cb_j}{\lambda_j}|u_j\rangle|\tilde{l}_j\rangle|1\rangle$$

where the ‘new’ normalisation constant $\mathcal{N}$ arising from the measurement of the ancilla qubit:

$$\mathcal{N} = \sqrt{\sum_{j=0}^{N-1}\frac{C^2|b_j|^2}{|\lambda_j|^2}}$$

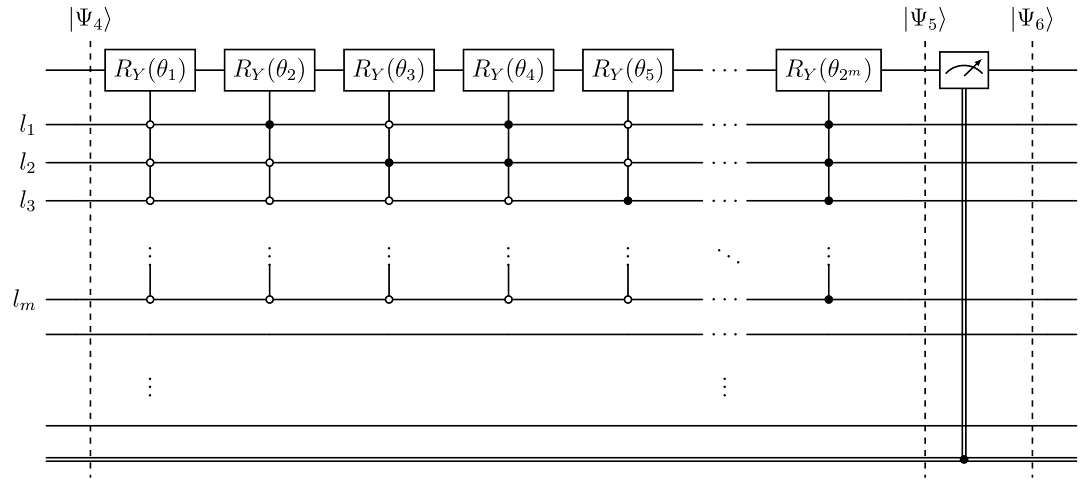


Figure 4.5: Quantum circuit representation of Ancilla rotation subroutine

At last, we see the probability amplitudes are now proportional to the solution $\mathbf{A}^{-1}b$. The final step is uncomputation of QPE, however the critical step of matrix inversion' has been completed before this step. Although the quantum circuit corresponding to QPE is quite familiar, the steps required to realise a quantum circuit for ancilla rotation (AR) are less understood. We proceed with deriving the circuit. The first step is choosing the function $f(x)$. In the original HHL paper [22], the authors use two ‘filter functions’ $f_1(x)$ and $f_2(x)$ and interchange the ancilla qubit for an ancilla qutrit (three dimensional Hilbert space):

$$\frac{1}{||\vec{b}||}\sum_{j=0}^{N-1} b_j|u_j\rangle|\tilde{l}_j\rangle\left(f_1(\tilde{l}_j)|0\rangle + f_2(\tilde{l}_j)|1\rangle + \sqrt{1-f_1(\tilde{l}_j)^2 - f_2(\tilde{l}_j)^2}|2\rangle\right)$$

The idea of these filter functions is to ameliorate the errors associated with inverting an eigenvalue with small magnitude. The errors associated with rotating by the reciprocal of an eigenvalue much smaller comparably in magnitude versus the rest would completely dominate over the other errors. The filter functions split the subspace into a well-conditioned and ill-conditioned subspace, the ill-conditioned subspace is the eigenspace corresponding to the eigenvalues whose absolute value is less than a predetermined threshold. Upon postselection after AR, measurement of $|0\rangle$ implies that matrix inversion has taken place and that the original vector $|b\rangle$ resided in the well-conditioned subspace, measurement of $|1\rangle$ implies a pseudo-matrix inversion has occurred as a result of $|b\rangle$ residing in the ill-conditioned subspace and finally a result of $|2\rangle$ implies no matrix inversion has occurred and the previous steps have to be repeated. The authors in [22] also utilise amplitude amplification [8] to increase the postselection probability for the 'correct state'. However for the purposes of explication, we will only consider the binary case of successful or non-successful matrix inversion at postselection. We will also not consider the amplitude amplification subroutine as it does not alter the key fact that the complexity scales polynomially with the condition number κ. If we consider the general mapping again:

$$\frac{1}{||\vec{b}||}\sum_{j=0}^{N-1} b_j|u_j\rangle|\tilde{l}_j\rangle|0\rangle \rightarrow \frac{1}{||\vec{b}||}\sum_{j=0}^{N-1} b_j|u_j\rangle|\tilde{l}_j\rangle\left(\sqrt{1-f(\tilde{l}_j)^2}|0\rangle + f(\tilde{l}_j)|1\rangle\right)$$

The unitary evolution of the ancilla qubit can be thought of as a controlled Pauli Y rotation (RY) by an appropriately chosen angle θ_j for each eigenvalue λ_j:

$$RY(\theta_j) = \exp\left(-\left(\frac{i\theta_j}{2}\right)\sigma_Y\right) = \begin{pmatrix}\cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right)\\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right)\end{pmatrix}$$

This rotation is controlled by the state of the c-register:

$$\frac{1}{||\vec{b}||d}\sum_{j=0}^{N-1} b_j|u_j\rangle(C-R_y)(|\tilde{l}_j\rangle|0\rangle)$$
$$\xrightarrow{\text{Ancilla rotation}} \frac{1}{||\vec{b}||}\sum_{j=0}^{N-1} b_j|u_j\rangle|\tilde{l}_j\rangle\left(\cos\left(\frac{\theta_j}{2}\right)|0\rangle+\sin\left(\frac{\theta_j}{2}\right)|1\rangle\right) \tag{4.17}$$

To match the desired eigenvalue inversion, we must equate:

$$\frac{C}{\lambda_j}=\sin\left(\frac{\theta_j}{2}\right)$$

$$\Rightarrow \theta_j = 2\arcsin\left(\frac{C}{\lambda_j}\right)=2\arcsin\left(\frac{C}{g(\tilde{l}_j)}\right)$$

The explicit function is given as:

$$\theta(l)=\begin{cases}2\arcsin\left(\frac{Ct}{2\pi l}\right) & 0<l<\frac{M}{2}\\ 2\arcsin\left(\frac{Ct}{2\pi(l-M)}\right) & \frac{M}{2}<l<M\end{cases}$$

Therefore if we rotate by $\tilde{\theta}_j=\theta(\tilde{l}_j)$, we get:

$$|\tilde{l}_j\rangle|0\rangle \Rightarrow |\tilde{l}_j\rangle\left(e^{-\frac{i\tilde{\theta}_j\sigma_Y}{2}}|0\rangle\right)\Rightarrow|\tilde{l}_j\rangle\left(\sqrt{1-\frac{C^2}{\lambda_j}}|0\rangle+\frac{C}{\lambda_j}|1\rangle\right) \tag{4.18}$$

The definition above of $\theta(l)$ is rather abstract in its current definition, we would rather consider the actual circuit implementation in terms of gates. Consider the representation of a particular Fourier basis state $|\tilde{l}_j\rangle$ in the computational basis again:

$$|\tilde{l}_j\rangle=|(\tilde{l}_j)_{m-1}\rangle|(\tilde{l}_j)_{m-2}\rangle\dots|(\tilde{l}_j)_0\rangle \quad ; \quad \tilde{l}_j=\sum_{q=0}^{m-1}(\tilde{l}_j)_q 2^q$$

$$(\tilde{l}_j)_k\in\{0,1\},\quad \forall k\in 0,1,...,m-1$$

Adjoining the ancilla qubit, the desired evolution can be seen as:

$$|(\tilde{l}_j)_{m-1}\rangle|(\tilde{l}_j)_{m-2}\rangle\dots|(\tilde{l}_j)_0\rangle|0\rangle$$
$$\xrightarrow{\text{Ancilla rotation}} |(\tilde{l}_j)_{m-1}\rangle|(\tilde{l}_j)_{m-2}\rangle\dots|(\tilde{l}_j)_0\rangle\left(\exp\left(-\frac{i\sigma_y}{2}\theta\left(\sum_{q=0}^{m-1}(\tilde{l}_j)_q 2^q\right)\right)|0\rangle\right)$$

This evolution can be achieved by defining a multi-qubit controlled rotation gate $C-U$, conditioned on qubit 0 in the c-register being in the state $(\tilde{l}_j)_0$, qubit 1 being in state the state$(\tilde{l}_j)_1$ and so forth such that if all qubits are in the 'correct state', a Pauli Y rotation through an angle:

$$\tilde{\theta_j} = \theta\left(\sum_{q=0}^{m-1}(\tilde{l}_j)_q 2^q\right)$$

is performed on the ancilla qubit. These rotation coefficients are computed classically ahead of time for the algorithm, a different rotation coefficient for each possible Fourier basis state in the c-register (i.e 2^m). For the general case, this requires an exponential amount of pre-processing before deployment of the algorithms. However, the initial computational cost can be reduced if Taylor approximations to the analytic form of $\theta(l)$ are considered. Various schemes exist for 'quantum arithmetic' circuits [51] which could be composed together to implement (4.18) without an exponentially increasing circuit depth with the number of qubits in the c-register.

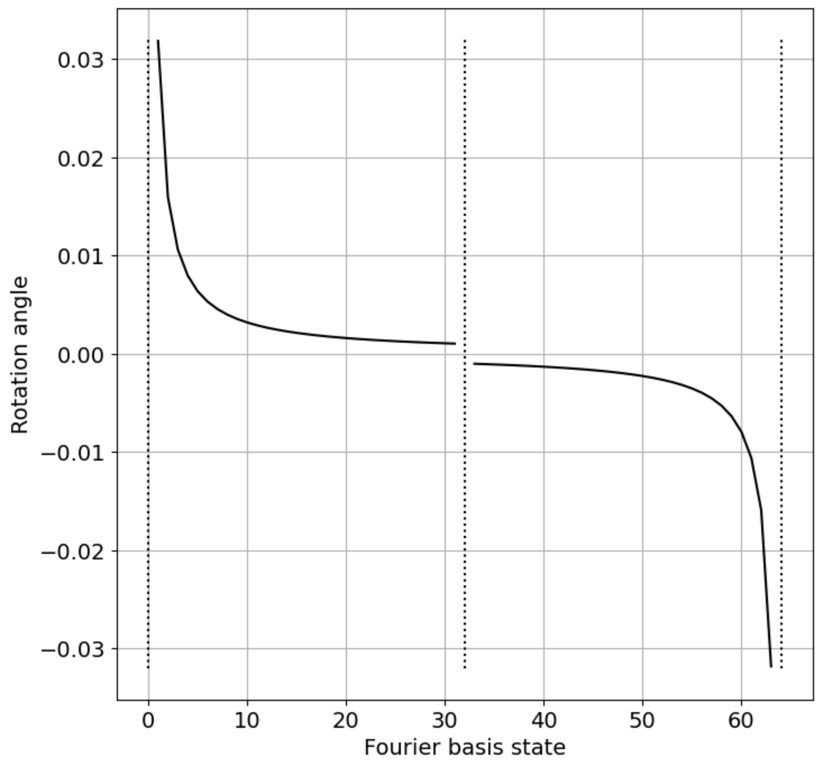


Figure 4.6: Plot of rotation angles versus Fourier basis states with parameters $M = 64$, $t = 0.5$ and $C = 0.2$. The three dotted lines from right to left denote the basis states $l = 0, 32, 64$ for which the rotation angle is not defined.

4.2.3 Uncomputation of the QPE subroutine

After successful postselection of the ancilla qubit, the current state will be :

$$|\Psi_6\rangle = \frac{1}{\mathcal{N}} \sum_{j=0}^{N-1} \frac{b_j}{\lambda_j} |u_j\rangle |\tilde{l}_j\rangle |1\rangle$$

The final step in the HHL algorithm is inverse quantum phase estimation (QPE†) which simply returns the c-register to its initial state, namely $|0\rangle$:

$$|\Psi_7\rangle = \frac{1}{\mathcal{N}} \sum_{j=0}^{N-1} \frac{b_j}{\lambda_j} |u_j\rangle \, |0\rangle^{\otimes m} \, |1\rangle$$

$$|\Psi_7\rangle = \underbrace{\left(\frac{1}{\mathcal{N}} \sum_{j=0}^{N-1} \frac{b_j}{\lambda_j} |u_j\rangle \right)}_{\propto \mathbf{A}^{-1}\vec{b}} |0\rangle^{\otimes m} \, |1\rangle$$

With the final step of the HHL algorithm, a quantum state in v-register whose amplitudes are proportional to the solution of the original linear system has been reproduced.

4.2.4 Measurement of solution $|x\rangle$

At the end of the algorithm, the solution of the linear system of equations $\vec{x} = \mathbf{A}^{-1}\vec{b}$ should be approximately encoded into the amplitudes of a quantum state $|x\rangle$. Furthermore as $|x\rangle$ is a quantum state, the solution encoded in $|x\rangle$ will be necessarily normalized whereas the 'actual' solution $\vec{x}$ does not have to be necessarily normalised. In addition, only relative phases between quantum states are physically meaningful, implying that the state $|x\rangle$ could correspond to the family of 'possible' states:

$$|x\rangle \in \left\{ e^{i\theta} \frac{\vec{x}}{\|\vec{x}\|} \mid \quad \theta \in [0, 2\pi) \right\}$$

In the case of real-valued $\mathbf{A}$ and $\vec{b}$, this reduces to $|x\rangle$ corresponding to either $-\frac{\vec{x}}{\|\vec{x}\|}$ or $\frac{\vec{x}}{|\vec{x}\|}$. Often for useful information to be extracted from the solution $\vec{x}$ the norm $\|\vec{x}\|$ will have to recovered. We will detail how this can be done in the next section. We could estimate the magnitude of each of the components of the solution by running the HHL circuit multiple times and measuring each qubit which has support for the solution $|x\rangle$. However, with this approach we would not recover the sign of each of the components of the solution and furthermore it would take exponentially increasing number of measurements of the solution (and thus executions of the circuit) to faithfully recover the magnitude of each of these components. Another option is to measure the inner product of the solution vector $\vec{x}$ with another chosen vector $\vec{w}$. We now outline the relatively simple quantum circuit that can be utilised to estimate inner products or overlaps between two quantum states.

4.2.5 SWAP **test**

If we denote $\vec{x} = ||\vec{x}||\vec{\tilde{x}}$ and $\vec{w} = ||\vec{w}||\vec{\tilde{w}}$, we can express the inner product of the two vectors as an inner product of their normalized versions scaled by their respective norms:

$$\langle \vec{w}, \vec{x} \rangle = ||\vec{w}||||\vec{x}||\langle \vec{\tilde{w}}, \vec{\tilde{x}} \rangle$$

As $\vec{\tilde{w}}, \vec{\tilde{x}}$ are normalized vectors, they can potentially correspond to quantum states and thus we can construct a circuit to calculate their overlap.

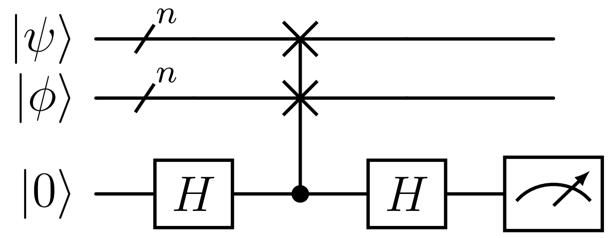


Figure 4.7: Quantum circuit representation of SWAP test for two n qubit states

To examine the effect of the circuit, we evaluate the action of the first Hadamard:

$$|\psi\rangle\,|\phi\rangle\,(H\,|0\rangle) = \frac{1}{\sqrt{2}}\,|\psi\rangle\,|\phi\rangle\,|0\rangle + \frac{1}{\sqrt{2}}\,|\psi\rangle\,|\phi\rangle\,|1\rangle$$

Applying the controlled SWAP yields:

$$\begin{aligned}
&= \frac{1}{\sqrt{2}}\,(\text{C-SWAP})\,(|\psi\rangle\,|\phi\rangle\,|0\rangle) + \frac{1}{\sqrt{2}}\,(\text{C-SWAP})\,(|\psi\rangle\,|\phi\rangle\,|1\rangle) \\
&= \frac{1}{\sqrt{2}}\,|\psi\rangle\,|\phi\rangle\,|0\rangle + \frac{1}{\sqrt{2}}\,|\phi\rangle\,|\psi\rangle\,|1\rangle
\end{aligned}$$

Applying the final Hadamard gate yields:

$$\begin{aligned}
&= \frac{1}{\sqrt{2}} |\psi\rangle |\phi\rangle (H|0\rangle) + \frac{1}{\sqrt{2}} |\phi\rangle |\psi\rangle (H|1\rangle) \\
&= \frac{1}{2} |\psi\rangle |\phi\rangle |0\rangle + \frac{1}{2} |\psi\rangle |\phi\rangle |1\rangle + \frac{1}{2} |\phi\rangle |\psi\rangle |0\rangle - \frac{1}{2} |\phi\rangle |\psi\rangle |1\rangle \\
&= \left(\frac{1}{2} (|\psi\rangle |\phi\rangle + |\phi\rangle |\psi\rangle)\right) |0\rangle + \left(\frac{1}{2} (|\psi\rangle |\phi\rangle - |\phi\rangle |\psi\rangle)\right) |1\rangle \qquad (4.19)
\end{aligned}$$

For an arbitrary quantum state $|\Psi\rangle = |\psi_0\rangle |0\rangle + |\psi_1\rangle |1\rangle$, the probability of measuring the state $|1\rangle$ can be calculated from the expectation value of the projector $\Pi_{|1\rangle} := I \otimes |1\rangle \langle 1|$:

$$\begin{aligned}
\text{Prob}(|1\rangle) &= \langle \Psi | \Pi_{|1\rangle} |\Psi\rangle \\
&= (\langle \psi_0| \langle 0| + \langle \psi_1| \langle 1|) (I \otimes |1\rangle \langle 1|) (|\psi_0\rangle |0\rangle + |\psi_1\rangle |1\rangle) \\
&= (\langle \psi_0| \langle 0| + \langle \psi_1| \langle 1|) (|\psi_1\rangle |1\rangle) \\
&= \langle \psi_1 | \psi_1 \rangle \qquad (4.20)
\end{aligned}$$

Therefore for the SWAP test, the probability of measuring $|1\rangle$ is given by:

$$\begin{aligned}
\text{Prob}(|1\rangle) &= \left(\frac{1}{2} (\langle \psi| \langle \phi| - \langle \phi| \langle \psi|)\right) \left(\frac{1}{2} (|\psi\rangle |\phi\rangle - |\phi\rangle |\psi\rangle)\right) \\
&= \frac{1}{4} (\langle \psi|\psi\rangle \langle \phi|\phi\rangle - \langle \psi|\phi\rangle \langle \phi|\psi\rangle - \langle \phi|\psi\rangle \langle \psi|\phi\rangle + \langle \phi|\phi\rangle \langle \psi|\psi\rangle) \\
&= \frac{1}{4} \left(2 - 2|\langle \psi|\phi\rangle|^2\right) = \frac{1}{2} \left(1 - |\langle \psi|\phi\rangle|^2\right)
\end{aligned}$$

Rearranging this relationship, we can calculate the absolute value of the overlap of two states in terms of the probability as:

$$|\langle \psi|\phi\rangle| = \sqrt{1 - 2\,\text{Prob}(|1\rangle)}$$

Recovery of solution norm $\|\vec{x}\|$

We now return to (4.16):

$$|\Psi_5\rangle = \frac{1}{\|\vec{b}\|} \sum_{j=0}^{N-1} b_j |u_j\rangle |\tilde{l}_j\rangle \left(\sqrt{1 - \frac{C^2}{\lambda_j^2}} |0\rangle + \frac{C}{\lambda_j} |1\rangle\right)$$

As previously emphasised, we will need some method to recover the norm of the solution $\vec{x}$. One method is to examine the postselection success probability of the ancilla qubit. Expressing the above state in the same format as (4.19) yields:

$$|\Psi_5\rangle = \left(\sum_{j=0}^{N-1} \frac{b_j}{\|\vec{b}\|}\sqrt{1-\frac{C^2}{\lambda_j^2}}\,|u_j\rangle\,|\tilde{l}_j\rangle\right)|0\rangle + \left(\sum_{j=0}^{N-1} \frac{b_j C}{\|\vec{b}\|\lambda_j}\,|u_j\rangle\,|\tilde{l}_j\rangle\right)|1\rangle$$

We can apply (4.20) again to determine the probability of successful postselection as:

$$\text{Prob}(|1\rangle) = \sum_{j=0}^{N-1}\left|\frac{b_j C}{\|\vec{b}\|\lambda_j}\right|^2 = \frac{C^2}{\|\vec{b}\|^2}\sum_{j=0}^{N-1}\left|\frac{b_j}{\lambda_j}\right|^2$$

Recalling the definition of the classical solution $\vec{x}$ in the eigenbasis of $\mathbf{A}$:

$$\|\vec{x}\| = \left\|\sum_{j=0}^{N}\frac{b_j}{\lambda_j}\vec{u_j}\right\| = \sqrt{\sum_{j=0}^{N-1}\left|\frac{b_j}{\lambda_j}\right|^2}$$

Substituting back into (4.20) we see that:

$$\text{Prob}\left(|1\rangle\right) = \frac{C^2\|\vec{x}\|^2}{\|\vec{b}\|^2}$$

$$\|\vec{x}\| = \frac{\|\vec{b}\|\sqrt{\text{Prob}(|1\rangle)}}{C} \tag{4.21}$$

In order to use the above equality (4.21) to determine the norm of the solution, we assume either a priori knowledge of of $\|\vec{b}\|$ or some oracle to calculate this value. If $\vec{b}$ is mostly sparse or has some other specific structure, the normalisation **may** be determined efficiently, i.e. that the determination of this normalisation constant would not be bottleneck in the overall runtime of the algorithm.

4.3 Performance against classical matrix inversion techniques

The field of classical linear solvers is vast as detailed in [49], [13]. Direct methods include but are not limited to the LU decomposition and QR decomposition, Cholesky decomposition and Gaussian elimination. Iterative techniques include the Jacobi method, Gauss-Seidel method etc. The conjugate gradient method [44] for solving linear systems is often thought to be one of the fastest methods for solving linear systems. The performance of this algorithm for solving a system

$\mathbf{A}\vec{x} = \vec{b}$ scales as:

$$\mathcal{O}\left(Ns\kappa \log\left(\frac{1}{\epsilon}\right)\right)$$

where s is the sparsity (maximum number of non-zero elements in any one row or column), N is the dimension of the problem ($N = \dim(\mathbf{A})$) and ϵ is the error tolerance. The complexity of the HHL algorithm [22] is given by:

$$\mathcal{O}\left(\frac{\log(N)s^2\kappa^2}{\epsilon}\right)$$

Therefore although we have achieved exponential better dependence on the dimension of the problem with respect to the complexity (logarithmic scaling in the quantum algorithm vs linear scaling in the classical algorithm), we have traded this for an exponentially worse dependence on the error tolerance parameter (linear scaling in the quantum algorithm vs logarithmic scaling in the classical algorithm). The exponentially worse dependence on the error tolerance has been reduced with the development of new matrix inversion algorithms, among many achieving $\mathcal{O}(\kappa(\log(\kappa)/\epsilon)^3)$ with an adiabatic randomization method in [47] and achieving a complexity of $\mathcal{O}(\kappa \text{polylog}(\kappa/\epsilon))$ with a zero eigenstate filtering method in [34]. Finally, an algorithm attaining the theoretical optimal scaling for each of these parameters was described with $\mathcal{O}(\kappa \log(1/\epsilon))$ in [11]. A table detailing all of these complexity results can also be found in [11]. The theoretical optimal scaling of $\mathcal{O}\left(\kappa \log\left(1/\epsilon\right)\right)$ for the algorithm arises from the widely believed conjecture that BQP $\subset$ PSPACE. The equivalence between these two statements was shown in the original HHL paper [22].

Chapter 5

Analysis of Λ matrix

Here we analyse the Λ matrix described in equation (3.18). From the previous chapter outlining the quantum algorithm of linear systems for equations, the sparsity and condition number of the matrix describing the system are critical factors in determining the runtime of the algorithm. In this context, The sparsity is defined as the maximum number of non-zero rows in any one row or column and condition number refers to the ratio of the largest singular value to the smallest singular value. For example, if the resources required to engineer the unitary associated with the matrix scales linearly with the problem size or if the condition number of the matrix constrains the probability of successful post-selection in the quantum algorithm, no speed up will be achieved. In this chapter, we address both of these questions for the proper Black-Scholes differential operator as well as an 'improper' Black-Scholes differential operator which we define. A previous work [5] showed that the runtime of the algorithm in this instance for the application of the HHL for matrix differential equations depends not on the condition number of the matrix $\mathbf{A}$ but rather the matrix V that diagonalises it $\mathbf{A} = VDV^{-1}$. More specifically, consider a $N \times N$ matrix $\mathbf{A} = VDV^{-1}$ where $D = \mathrm{diag}\left(\lambda_0, ..., \lambda_{N-1}\right)$ such that $\mathrm{Re}(\lambda_j) \leq 0$ for all $j \in \{0, 1, ..., N-1\}$ and has at most s non-zero elements in any one row or column. The authors of [5] showed that the query complexity for producing a state $\epsilon-$close to $\vec{x}(T)/\left\|\vec{x}(T)\right\|$ in the ℓ^2 norm where $\vec{x}(T)$ corresponds to the evolution of the state $\vec{x}(0) = \vec{x}_{in}$ under $\frac{d\vec{x}}{dt} = \mathbf{A}\vec{x} + \vec{b}$ for a time period T is given by:

$$\mathcal{O}\left(\kappa_V sgT \left\|\mathbf{A}\right\| \times \mathrm{polylog}\left(\kappa_V sg\beta T \left\|\mathbf{A}\right\| / \epsilon\right)\right)$$

where $g := \max_{t\in[0,T]} \|\vec{x}(t)\| / \|\vec{x}(T)\|$, $\beta := \left(||\vec{x}_{in}|| + T||\vec{b}||\right) / \|\vec{x}(T)\|$ and $\kappa_V := \|V\| \, \|V^{-1}\|$. They also prove that the gate complexity is larger than the query complexity by a factor of $\mathrm{polylog}\left(\kappa_V sg\beta T \|\mathbf{A}\| / \epsilon\right)$. A second paper [35] applied this algorithm to derive an end-to-end description of multi-asset option pricing algorithm. The authors in [35] address comprehensively the many technical subtleties and complexity assumptions associated with this algorithm. Our motivation for this thesis was a description of the problem of option pricing along with the simplest implementation of the algorithm with a corresponding analysis of the scaling of these numerical characteristics that determine the runtime of the algorithm. Taking our main inspiration from this paper [35], we tried to implement the most basic form of this algorithm which includes the HHL algorithm and our goal is to derive upper bounds for the parameters κ_V, $||\tilde{\mathbf{A}}||$ and s for this newly defined improper differential operator $\tilde{\mathbf{A}}$. Alongside providing some upper bounds for the sparsity s and spectral norm of $\tilde{\mathbf{A}}$, $||\tilde{\mathbf{A}}||$, we show that $\tilde{\mathbf{A}}$ is unitarily diagonalisable implying an optimal condition number of 1. We also show that the real part of the spectrum of $\tilde{\mathbf{A}}$ is less than or equal to zero as required by the above algorithm. For the case of the standard Black-Scholes differential operator, we provide numerical evidence of unfavourable scaling in this parameter in the dimension of the Black-Scholes differential operator.

5.1 Condition number of V and spectral norm of $\mathbf{A} = VDV^{-1}$

We now return to the problem of identifying the condition number of the relevant matrix. In order to determine the condition number of V, we have to either find the matrix norm of V and its respective inverse:

$$\kappa(V) = \|V\|_2 \|V^{-1}\|_2 \tag{5.1}$$

where $\|.\|_2$ refers to spectral norm. Alternatively, the condition number can be defined as ratio of the largest to smallest singular values:

$$\kappa(V) = \max_{s_i, s_j \in s(V)} \frac{|s_i|}{|s_j|} \qquad s(V) := \sqrt{\sigma(V^\dagger V)} \tag{5.2}$$

We first proceed with examining if we can find an eigenbasis for $\mathbf{A}$. For the case of the matrices of interest in this thesis, tridiagonal matrices $\in \mathbb{R}^{N\times N}$, there is an analytic expression for the spectrum [37]. The l^{th} eigenvalue and eigenvector is given by:

$$\lambda_l = b + 2\sqrt{ac}\cos\left(\frac{\pi l}{N+1}\right), \quad l \in \{1, 2, ..., N\} \tag{5.3}$$

Where a is the value along the subdiagonal, b along the diagonal and c along the superdiagonal. Assuming that $ac \neq 0$, the associated right l^{th} eigenvector is given by the expression:

$$\vec{v_l} = \begin{pmatrix} (\frac{c}{a})^{\frac{1}{2}}\sin(\frac{l\pi}{N+1}) \\ (\frac{c}{a})^{\frac{2}{2}}\sin(\frac{2l\pi}{N+1}) \\ \vdots \\ (\frac{c}{a})^{\frac{N}{2}}\sin(\frac{Nl\pi}{N+1}) \end{pmatrix} \tag{5.4}$$

For the case of the D_1 matrix, $\frac{c}{a} = -1 \implies \frac{c}{a} = e^{\pi i}$ .We show that for the D_1 matrix, the eigenvectors listed above form an eigenbasis, by showing the stronger condition that each of the eigenvectors are orthogonal with respect to the usual norm on $\mathbb{C}^N$. Taking two vectors $\vec{v_l}, \vec{v_k}$, we see their inner product is:

$$\vec{v_l}^\dagger\vec{v_k} = \begin{pmatrix} e^{\frac{-\pi i}{2}}\sin(\frac{l\pi}{N+1}) & e^{-\pi i}\sin(\frac{2l\pi}{N+1}) & \dots & e^{\frac{-N\pi i}{2}}\sin(\frac{Nl\pi}{N+1}) \end{pmatrix} \begin{pmatrix} e^{\frac{\pi i}{2}}\sin(\frac{k\pi}{N+1}) \\ e^{\pi i}\sin(\frac{2k\pi}{N+1}) \\ \vdots \\ e^{\frac{N\pi i}{2}}\sin(\frac{Nk\pi}{N+1}) \end{pmatrix}$$

$$\Rightarrow \vec{v_l}^\dagger\vec{v_k} = \sum_{\gamma=1}^{N}\sin\left(\frac{\gamma l}{N+1}\right)\sin\left(\frac{\gamma k}{N+1}\right) = \left(\frac{N+1}{2}\right)\delta_{l,k}$$

Where in the last line, we have invoked the orthogonality of discrete sine functions on a closed interval to insert the Kronecker delta function. As for the D_2 matrix, its Hermiticity guarantees that its eigenvectors will necessarily form an eigenbasis. Returning to the discretized representation of A:

$$\mathbf{A} = \left(\frac{1}{2}\sum_{i=1}^{d}\sigma_i^2 A_{(\partial^2 x_i)} + \sum_{\substack{i,j=1 \\ j>i}}^{d}\rho_{ij}\sigma_i\sigma_j A_{(\partial x_i, \partial x_j)} + \sum_{i=1}^{d}\left(r - \frac{\sigma_i^2}{2}\right)A_{(\partial x_i)}\right) \tag{5.5}$$

where the submatrices $A_{(\partial x_i)}$, $A_{(\partial x_i, \partial x_j)}$ and $A_{(\partial^2 x_i)}$ are defined as:

$$A_{(\partial x_i)} = I^{\otimes d-i} \otimes D_1 \otimes I^{\otimes i-1} \qquad A_{(\partial^2 x_i)} = I^{\otimes d-i} \otimes D_2 \otimes I^{\otimes i-1}$$

$$A_{(\partial x_i, \partial x_j)} = I^{\otimes d-j} \otimes D_1 \otimes I^{\otimes j-i-1} \otimes D_1 \otimes I^{\otimes i-1}$$

We can now immediately deduce that for the matrices $A_{(\partial x_i)}, A_{(\partial x_j, \partial x_k)}$, they have the following eigenbasis:

$$\vec{v}_{l_1,l_2,\ldots,l_d} := \vec{v}_{l_1} \otimes \vec{v}_{l_2} \otimes \ldots \otimes \vec{v}_{l_d} \qquad \forall l_1, l_2, ..l_N \in \{1, 2, ..., N\} \tag{5.6}$$

$$\vec{v}_{l_i} = \begin{pmatrix} e^{\frac{\pi i}{2}} \sin(\frac{l\pi}{N+1}) \\ e^{\pi i} \sin(\frac{2l\pi}{N+1}) \\ \vdots \\ e^{\frac{N\pi i}{2}} \sin(\frac{Nl\pi}{N+1}) \end{pmatrix}$$

For matrices of the type $A_{(\partial^2 x_i)}$, their corresponding eigenbasis will be:

$$\vec{v}_{l_1,l_2,\ldots,l_d} := \vec{v}_{l_1} \otimes \vec{v}_{l_2} \otimes \ldots \otimes \vec{v}_{l_d} \qquad \forall l_1, l_2, ..l_N \in \{1, 2, ..., N\} \tag{5.7}$$

$$\vec{v}_{l_i} = \begin{pmatrix} \sin(\frac{l\pi}{N+1}) \\ \sin(\frac{2l\pi}{N+1}) \\ \vdots \\ \sin(\frac{Nl\pi}{N+1}) \end{pmatrix}$$

It can be seen from (5.6) and (5.7) that the eigenvectors from the matrices $A_{(\partial x_i)}$ and $A_{(\partial^2 x_i)}$ are distinct. This implies that these matrices do not commute and hence the straightforward analysis to determine the simultaneous eigenbasis will not be applicable here. Alternatively if $\mathbf{A}$ is Hermitian, we can immediately deduce that $\mathbf{A}$ is diagonalised by a unitary matrix V ($\mathbf{A} = VDV^{-1}$). The singular values of V and thus the condition number of V in this instance is trivial as:

$$s(V) = \sqrt{\sigma(V^\dagger V)} = \sqrt{\sigma(I)} = 1$$

$$\kappa(V) = \max_{s_i, s_j \in s(V)} \frac{s_i}{s_j} = 1$$

However A is not Hermitian. This can be seen as the matrix D_1 is anti-Hermitian.

$$D_1^\dagger = \frac{1}{2h}\begin{pmatrix} 0 & -1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & & -1 \\ & & 1 & 0 \end{pmatrix} = -\frac{1}{2h}\begin{pmatrix} 0 & 1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & & 1 \\ & & -1 & 0 \end{pmatrix} = -D_1$$

Corollary of this is that the matrices of the form of $A_{(\partial x_i)}$ are anti-Hermitian and the matrices $A_{(\partial x_i,\partial x_j)} := A_{(\partial x_i)}A_{(\partial x_j)}$ are Hermitian.

$$\begin{aligned} A^\dagger_{(\partial x_i)} &= I\otimes\cdots\otimes D_1^\dagger\otimes\cdots\otimes I \to I\otimes\cdots\otimes(-D_1)\otimes\cdots\otimes I \\ &\to -I\otimes\ldots D_1\cdots\otimes I = -A_{(\partial x_i)} \end{aligned} \tag{5.8}$$

$$\begin{aligned} A^\dagger_{(\partial x_i,\partial x_j)} &= \left(A_{(\partial x_i)}A_{(\partial x_j)}\right)^\dagger \to A^\dagger_{(\partial x_j)}A^\dagger_{(\partial x_i)} \\ &\to \left(-A_{(x_j)}\right)\left(-A_{(x_i)}\right) = A_{(\partial x_i)}A_{(\partial x_j)} = A_{(\partial x_i,\partial x_j)} \end{aligned} \tag{5.9}$$

Where in for the last equality in (5.9), we have used the fact that $[A_{(\partial x_i)}, A_{(\partial x_i)}] = 0 \quad \forall i,j$ to commute the matrices through. As the matrices of the type $A_{(\partial x_i)}$ form part of the sum that defines **A** in (5.5) and the remaining terms are Hermitian, **A** itself cannot be Hermitian. The second option is to consider the weaker condition of **A** being normal, which would imply that **A** can be unitarily diagonalized. This is equivalent to the statement:

$$[\mathbf{A},\mathbf{A}^\dagger] = 0$$

It can be shown that **A** does not satisfy the normality condition. However if we consider the $A_{(\partial^2 x_i)}$ matrix as the composition of the $A_{(\partial x_i)}$ matrix with itself, we can achieve normality in the overall matrix **A**.

$$A_{(\partial^2 x_i)} \Leftrightarrow \left(A_{(\partial x_i)}\right)^2$$

This now approximately corresponds to the operator $A_{(\partial^2 x_i)}$ with stepsize equal to $2h$ rather than h. Having two different step-sizes for different differential operators will unavoidably lead to additional error in the approximation as well as an additional minor perturbation in the top left and bottom right elements of the matrix respectively leading to a reconsideration of how boundary conditions are imposed. However, this would have the benefit of every differential operator (i.e.$A_{(\partial x_i)}$,$A_{(\partial^2 x_i)}$,etc) in the larger Black-Scholes differential operator $\tilde{\mathbf{A}}$ commuting pairwise with one another, allowing a straightforward determination of normality of the 'improper' differential operator $\tilde{\mathbf{A}}$. We proceed with this approximation now and examine in due course the practicality of this approximation with numerical simulations. Reinserting $\rho_{ii} = 1$ in each of the terms of the first summand and re-expressing $A_{(\partial x_i, \partial x_j)}$ as $A_{(\partial x_i)}A_{(\partial x_i)}$ yields:

$$\tilde{\mathbf{A}} = \left(\frac{1}{2} \sum_{i=1}^{d} \rho_{ii} \sigma_i^2 A_{(\partial x_i)}^2 + \sum_{\substack{i,j=1 \\ j>i}}^{d} \rho_{ij} \sigma_i \sigma_j A_{(\partial x_i)} A_{(\partial x_j)} + \sum_{i=1}^{d} \left(r - \frac{\sigma_i^2}{2} \right) A_{(\partial x_i)} \right) \tag{5.10}$$

We now can recombine the first two summands as follows. The first thing to notice is that switching the dummy variables i, j in the second summand, does not change the sum due to the symmetry of the correlation term ρ_{ij} and the fact that the matrices $A_{(\partial x_i)}, A_{(\partial x_j)}$ commute. Therefore the second summand can be represented as:

$$\frac{1}{2} \left(\sum_{\substack{i,j=1 \\ j>i}}^{d} \rho_{ij} \sigma_i \sigma_j A_{(\partial x_i)} A_{(\partial x_j)} + \sum_{\substack{j,i=1 \\ i>j}}^{d} \rho_{ji} \sigma_j \sigma_i A_{(\partial x_j)} A_{(\partial x_i)} \right) \Leftrightarrow$$

$$= \frac{1}{2} \sum_{\substack{i,j=1 \\ i \neq j}}^{d} \rho_{ij} \sigma_i \sigma_j A_{(\partial x_i)} A_{(\partial x_j)} \tag{5.11}$$

Inserting (5.11) yields:

$$\tilde{\mathbf{A}} = \left(\frac{1}{2} \sum_{i=1}^{d} \rho_{ii} \sigma_i^2 A_{(\partial x_i)}^2 + \frac{1}{2} \sum_{\substack{i,j=1 \\ i \neq j}}^{d} \rho_{ij} \sigma_i \sigma_j A_{(\partial x_i)} A_{(\partial x_j)} + \sum_{i=1}^{d} \left(r - \frac{\sigma_i^2}{2} \right) A_{(\partial x_i)} \right)$$

$$\tilde{\mathbf{A}} = \left(\frac{1}{2} \sum_{i,j=1}^{d} \rho_{ij} \sigma_i \sigma_j A_{(\partial x_i)} A_{(\partial x_j)} + \sum_{i=1}^{d} \left(r - \frac{\sigma_i^2}{2} \right) A_{(\partial x_i)} \right) \tag{5.12}$$

To determine $[\tilde{\mathbf{A}}, \tilde{\mathbf{A}}^\dagger]$, we decompose $\tilde{\mathbf{A}}$ into its Hermitian and anti-Hermitian components.

$$\tilde{\mathbf{A}} = M_{\text{Herm}} + M_{\text{anti-Herm}}$$

Where M_{Herm} and $M_{\text{anti-Herm}}$ are defined as:

$$M_{\text{Herm}} = \frac{1}{2}\sum_{i,j=1}^{d} \rho_{ij}\sigma_i\sigma_j A_{(\partial x_i)}A_{(\partial x_j)} \qquad M_{\text{anti-Herm}} = \sum_{i=1}^{d}\left(r - \frac{\sigma_i^2}{2}\right) A_{(\partial x_i)}$$

These components can be seen as Hermitian and anti-Hermitian from equations (5.8) and (5.9). Each of these respective components also commute as:

$$\begin{aligned}
[M_{\text{Herm}}, M_{\text{anti-Herm}}] &= \left[\frac{1}{2}\sum_{i,j=1}^{d} \rho_{ij}\sigma_i\sigma_j A_{(\partial x_i)}A_{(\partial x_j)}, \sum_{k=1}^{d}\left(r - \frac{\sigma_k^2}{2}\right) A_{(\partial x_k)}\right] \\
&\to \frac{1}{2}\sum_{i,j=1}^{d}\sum_{k=1}^{d} \rho_{ij}\sigma_i\sigma_j\left(r - \frac{\sigma_k^2}{2}\right)\left[A_{(\partial x_i)}A_{(\partial x_j)}, A_{(\partial x_k)}\right]
\end{aligned}$$

Using the identity $[AB, C] = A[B, C] + [A, C]B$, we can see that the last commutator vanishes as:

$$\left[A_{(\partial x_i)}A_{(\partial x_j)}, A_{(\partial x_k)}\right] = A_{(\partial x_i)}\underbrace{\left[A_{(\partial x_j)}, A_{(\partial x_j)}\right]}_{=0} + \underbrace{\left[A_{(\partial x_i)}, A_{(\partial x_k)}\right]}_{=0} A_{(\partial x_j)}$$

Therefore $[M_{\text{Herm}}, M_{\text{anti-Herm}}] = 0$. We can show now that $[\tilde{A}, \tilde{A}^\dagger]$ is zero:

$$\begin{aligned}
[\tilde{\mathbf{A}}, \tilde{\mathbf{A}}^\dagger] &= [M_{\text{Herm}} + M_{\text{anti-Herm}}, M_{\text{Herm}} - M_{\text{anti-Herm}}] \\
&= [M_{\text{Herm}}, M_{\text{Herm}}] - [M_{\text{Herm}}, M_{\text{anti-Herm}}] \\
&+ [M_{\text{anti-Herm}}, M_{\text{Herm}}] - [M_{\text{anti-Herm}}, M_{\text{anti-Herm}}] \\
&\Rightarrow -2[M_{\text{Herm}}, M_{\text{anti-Herm}}] = 0
\end{aligned}$$

Therefore V is unitary and thus we arrive at the result:

$$\kappa(V) = 1$$

Two other numerical characteristics of the matrix $\mathbf{A}$ that determines the overall runtime complexity is the spectral norm of $\mathbf{A}$ and the real part of all the eigenvalues of $\mathbf{A}$ being negative.

$$||\mathbf{A}||_{\text{Spec}} := \sqrt{\lambda_{\max}\left(\mathbf{A}^{\dagger}\mathbf{A}\right)}$$

Using a similar argument from commutation relations, $\tilde{\mathbf{A}}^{\dagger}\tilde{\mathbf{A}}$ has a simple representation in terms of the M_{Herm} and $M_{\text{anti-Herm}}$:

$$\begin{aligned}\tilde{\mathbf{A}}^{\dagger}\tilde{\mathbf{A}} &= \left(M_{\text{Herm}} - M_{\text{anti-Herm}}\right)\left(M_{\text{Herm}} + M_{\text{anti-Herm}}\right)\\ &= M_{\text{Herm}}^2 + \underbrace{\left[M_{\text{Herm}}, M_{\text{anti-Herm}}\right]}_{=0} - M_{\text{anti-Herm}}^2\\ &= M_{\text{Herm}}^2 - M_{\text{anti-Herm}}^2\end{aligned}$$

Returning to equation (5.6), we now determine the spectrum of the symmetric and anti-symmetric components. We first proceed with demonstrating the negativity of the real part of $\sigma(\tilde{\mathbf{A}})$. Taking the eigenvector $\vec{v}_{l_1,l_2,\ldots,l_d}$, we see that the eigenvalues of the Hermitian component are:

$$\underbrace{\left(\frac{1}{2}\sum_{i,j=1}^{d}\rho_{ij}\sigma_i\sigma_j A_{(\partial x_i)}A_{(\partial x_j)}\right)}_{M_{\text{Herm}}}\vec{v}_{l_1,l_2,\ldots,l_d} = \frac{1}{2}\sum_{i,j=1}^{d}\rho_{ij}\sigma_i\sigma_j\left(A_{(\partial x_i)}A_{(\partial x_j)}\vec{v}_{l_1,l_2,\ldots,l_d}\right)$$

$$\Leftrightarrow \frac{1}{2}\sum_{i,j=1}^{d}\rho_{ij}\rho_i\rho_j\left(-\frac{4}{4h^2}\cos\left(\frac{\pi l_i}{N+1}\right)\cos\left(\frac{\pi l_j}{N+1}\right)\right)\vec{v}_{l_1,l_2,\ldots,l_d}$$

$$\Leftrightarrow \underbrace{\left(-\frac{1}{2h^2}\sum_{i,j=1}^{d}\rho_{ij}\sigma_i\sigma_j\cos\left(\frac{\pi l_i}{n+1}\right)\cos\left(\frac{\pi l_j}{N+1}\right)\right)}_{\lambda_{l_1,l_2,\ldots,l_d}}\vec{v}_{l_1,l_2,\ldots,l_d} \tag{5.13}$$

Similarly for the anti-Hermitian component, the eigenvalues are:

$$\underbrace{\left(\sum_{k=1}^{d}\left(r - \frac{\sigma_k^2}{2}\right)A_{(\partial x_k)}\right)}_{M_{\text{anti-Herm}}}\vec{v}_{l_1,l_2,\ldots,l_d} = \sum_{k=1}^{d}\left(r - \frac{\sigma_k^2}{2}\right)\left(A_{(\partial x_k)}\vec{v}_{l_1,l_2,\ldots,l_d}\right)$$

$$\Leftrightarrow \sum_{k=1}^{d}\left(r - \frac{\sigma_k^2}{2}\right)\left(\frac{2i}{2h}\cos\left(\frac{\pi l_k}{N+1}\right)\right)\vec{v}_{l_1,l_2,\ldots,l_d}$$

$$\Leftrightarrow \underbrace{\left(\frac{i}{h}\sum_{k=1}^{d}\left(r-\frac{\sigma_k^2}{2}\right)\cos\left(\frac{\pi l_k}{N+1}\right)\right)}_{\lambda_{l_1,l_2,\ldots,l_d}}\vec{v}_{l_1.l_2,\ldots,l_d} \tag{5.14}$$

Hence the eigenvalues of $\tilde{A}$ are:

$$(\tilde{\mathbf{A}})\vec{v}_{l_1,l_2,\ldots,l_d} = (M_{\text{Herm}})\,\vec{v}_{l_1,l_2,\ldots,l_d} + (M_{\text{anti-Herm}})\,\vec{v}_{l_1,l_2,\ldots,l_d}$$

$$= \left(-\frac{1}{2h^2}\sum_{i,j=1}^{d}\rho_{ij}\sigma_i\sigma_j\cos\left(\frac{\pi l_i}{N+1}\right)\cos\left(\frac{\pi l_j}{N+1}\right)\right)\vec{v}_{l_1,l_2,\ldots,l_d} + \left(\frac{i}{h}\sum_{k=1}^{d}\left(r-\frac{\sigma_k^2}{2}\right)\cos\left(\frac{\pi l_k}{N+1}\right)\right)\vec{v}_{l_1,l_2,\ldots,l_d} \tag{5.15}$$

The expression can be re-expressed as a *pseudo*-quadratic programming problem. If we revisit the notation for the covariance matrix $(\hat{\sigma})_{ij} = \rho_{ij}\sigma_i\sigma_j$ as well as introduce the following notation:

$$(\vec{w})_k := \cos\left(\frac{\pi l_k}{N+1}\right) \qquad (\vec{u})_k := r - \frac{\sigma_k^2}{2}$$

The eigenvalue equation of (5.15) can be rewritten as:

$$(\tilde{\mathbf{A}})\vec{v}_{l_1,l_2,\ldots,l_d} = \left(-\frac{1}{2h^2}(\vec{w}^\dagger\hat{\sigma}\vec{w}) + \frac{i}{h}(\vec{u}^\dagger\vec{w})\right)\vec{v}_{l_1,l_2,\ldots,l_d} \tag{5.16}$$

We can deduce from (5.16) that the real part of the spectrum of $\tilde{\mathbf{A}}$ will be negative due to the semi-positive definiteness of the covariance matrix $\hat{\sigma}$:

$$\text{Re}(\sigma(\tilde{\mathbf{A}})) = -\frac{1}{2h^2}\underbrace{(\vec{w}^\dagger\hat{\sigma}\vec{w})}_{>0} < 0$$

We now address the issue of determining $||\mathbf{A}||_{\text{Spec}}$. Applying the eigenvalues equations of (5.13) and (5.14) to $\tilde{\mathbf{A}}^\dagger\tilde{\mathbf{A}}$ yields:

$$\left(\tilde{\mathbf{A}}^\dagger\tilde{\mathbf{A}}\right)\vec{v}_{l_1,l_2,\ldots,l_d} = M_{\text{Herm}}^2\vec{v}_{l_1,l_2,\ldots,l_d} - M_{\text{anti-Herm}}^2\vec{v}_{l_1,l_2,\ldots,l_d}$$

Utilising the condensed notation of equation yields:

$$\left(\tilde{\mathbf{A}}^\dagger\tilde{\mathbf{A}}\right)\vec{v}_{l_1,l_2,\ldots,l_d} = \left(-\frac{1}{2h^2}\vec{w}^\dagger\hat{\sigma}\vec{w}\right)^2\vec{v}_{l_1,l_2,\ldots,l_d} - \left(\frac{i}{h}\vec{u}^\dagger\vec{w}\right)^2\vec{v}_{l_1,l_2,\ldots,l_d}$$

$$\left(\tilde{\mathbf{A}}^\dagger \tilde{\mathbf{A}}\right) \vec{v}_{l_1,l_2,\ldots,l_d} = \frac{1}{4h^4}\left[\left(\vec{w}^\dagger \hat{\sigma} \vec{w}\right)^2 + 4h^2 \left(\vec{u}^\dagger \vec{w}\right)^2\right] \vec{v}_{l_1,l_2,\ldots,l_d} \tag{5.17}$$

To determine the largest value of (5.17), we first focus on the quadratic term. Re-expressing the term $\vec{w}^\dagger \hat{\sigma} \vec{w}$ in terms of the Rayleigh quotient, $\vec{w}^\dagger \hat{\sigma} \vec{w} = B(\hat{\sigma}, \vec{w})$, and the fact that $B(\hat{\sigma}, \vec{w}) \leq \lambda_{\max}(\hat{\sigma})$ yields:

$$(\vec{w}^\dagger \hat{\sigma} \vec{w})^2 \leq \left(\lambda_{\max}(\hat{\sigma})||\vec{w}||\right)^2 = \lambda_{\max}(\hat{\sigma})^2 ||\vec{w}||^2 \tag{5.18}$$

Examining the form of $\vec{w}$, we have the simple bound on $||\vec{w}||$ as:

$$||\vec{w}|| \leq \sqrt{\sum_{k=1}^{d} \cos\left(\frac{\pi l_k}{N+1}\right)^2} \leq \sqrt{\sum_{k=1}^{d} 1} = \sqrt{d}$$

This bounds (5.18) as:

$$\left(\vec{w}^\dagger \hat{\sigma} \vec{w}\right)^2 < \lambda_{\max}(\hat{\sigma})^2 \left(\sqrt{d}\right)^2 = d\lambda_{\max}(\hat{\sigma})^2$$

Turning to the linear programming aspect of (5.17), we can bound the quantity using the Cauchy-Schwarz inequality.

$$|\vec{u}^\dagger \vec{w}| \leq ||\vec{u}|| \cdot ||\vec{w}|| \tag{5.19}$$

We bound $||\vec{u}||$ as:

$$\begin{aligned} ||\vec{u}|| = \sqrt{\sum_{k=1}^{d} (\vec{u}_k)^2} &= \sqrt{\sum_{k=1}^{d} \left(r - \frac{\sigma_k^2}{2}\right)^2} \\ &\leq \sqrt{\sum_{k=1}^{d} \left(|r| + \frac{|\lambda_{\max}(\hat{\sigma})|}{2}\right)^2} = \left(|r| + \frac{\lambda_{\max}(\hat{\sigma})}{2}\right)\sqrt{d} \end{aligned}$$

This leads to a bound for (5.19) as:

$$(\vec{u}^\dagger \vec{w})^2 \leq \left(\left(|r| + \frac{\lambda_{\max}(\hat{\sigma})}{2}\right)\sqrt{d}\right)^2 \left(\sqrt{d}\right)^2 \leq d^2 \left(|r| + \frac{\lambda_{\max}(\hat{\sigma})}{2}\right)^2$$

We can now provide an upper bound for $||\mathbf{A}||_{\text{Spec}}$. The final parameter to be bounded is the stepsize h. As $h \sim \mathcal{O}\left(\frac{1}{N}\right)$, we will assume the upper bound $h \leq \frac{c}{N}$, where c is some constant independent of N or d. Combining it all together gives:

$$
\frac{1}{4h^4}\left((\vec{w}^\dagger \hat{\sigma} \vec{w})^2 + 4h^2(\vec{u}^\dagger \vec{w})^2\right)
$$

$$
\leq \frac{N^4}{4c^4}\left(d\lambda_{\max}(\hat{\sigma})^2 + \frac{4c^2}{N^2}d^2\left(|r| + \frac{\lambda_{\max}(\hat{\sigma})}{2}\right)^2\right)
$$

$$
\Rightarrow \frac{\lambda_{\max}(\hat{\sigma})^2}{4c^4}N^4 d + \frac{1}{c^2}\left(|r| + \frac{\lambda_{\max}(\hat{\sigma})}{2}\right)^2 N^2 d^2 \sim \mathcal{O}\left(\lambda_{\max}(\hat{\sigma})^2 \max\left(N^4 d, N^2 d^2\right)\right)
$$

5.2 Numerical plots of condition number and spectral norm for both differential operators

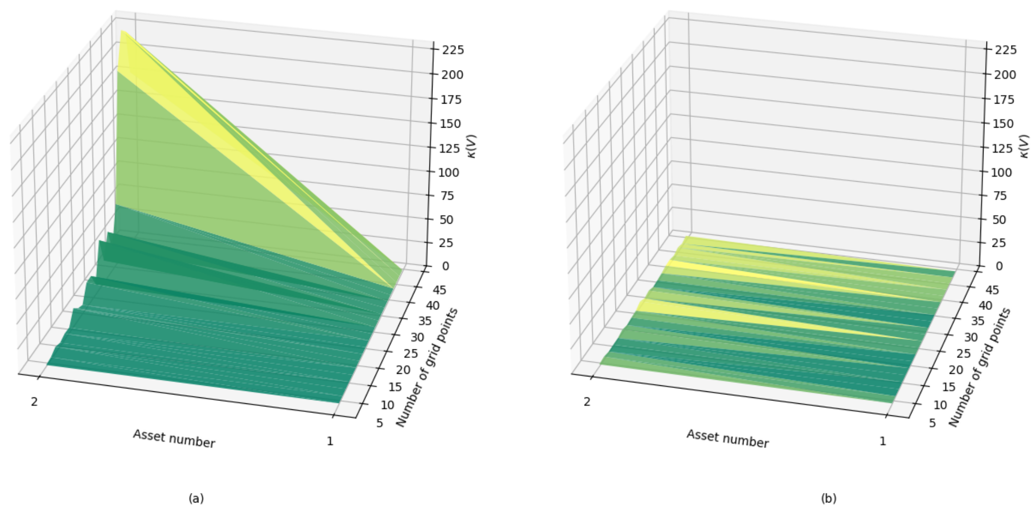


Figure 5.1: Plot of condition number of $\kappa\left(V\right)$ against number of underlying assets and number of gridpoints where $\mathbf{A} = VDV^{-1}$ for the case of (a) the proper Black-Scholes differential operator $\mathbf{A}$ and (b) improper Black-Scholes differential operator $\tilde{\mathbf{A}}$ for $r = 0.1$, $\sigma_i = 0.2$, $\Delta x = 0.2$ and no correlation between the assets.

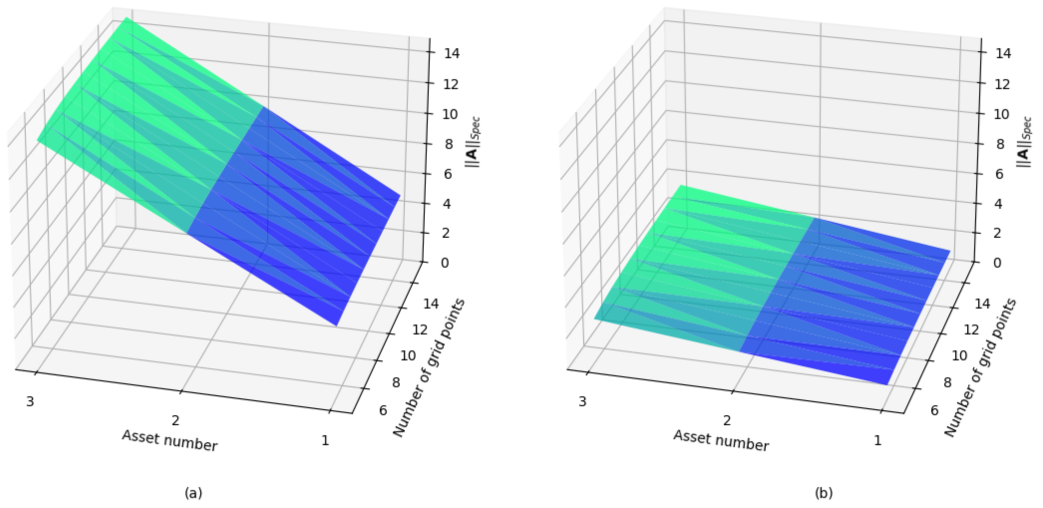


Figure 5.2: Plot of spectral norm of matrix $||(.)||_{Spec}$ against number of underlying assets and number of gridpoints for the case of (a) the proper Black-Scholes differential operator $\mathbf{A}$ and (b) improper Black-Scholes differential operator $\tilde{\mathbf{A}}$ for $r = 0.1$, $\sigma_i = 0.2$, $\Delta x = 0.2$ and no correlation between the assets.

5.2.1 Discussion of condition number and spectral norm

In this section, we have provided an analysis of some of the numerical characteristics associated with the proper and improper Black-Scholes differential operators associated with the Black-Scholes equation. By considering the improper Black-Scholes differential operator $\tilde{\mathbf{A}}$, we were able to find an explicit eigenbasis for the matrix which enabled us to derive analytic bounds on the condition number of the matrix that diagonalises $\mathbf{A}$ (namely V) as well as determine the spectral norm of $\tilde{\mathbf{A}}$ and the sign of real components of its spectrum $\text{Re}\left(\sigma(\tilde{\mathbf{A}})\right)$. As to the best of our knowledge the eigenbasis and associated spectrum of $\mathbf{A}$ cannot be expressed in analytic manner as with the case of $\tilde{\mathbf{A}}$ and therefore the above method to bound these parameters cannot be utilised. We also tried other methods such as examining inequalities regarding quantities such as the sub-multiplicativity of the condition number ($\kappa(AB) \leq \kappa(A)\kappa(B)$), however, we were not able to derive any analytic expressions for bounds on these variables. Summarized, the non-normality of the proper Black-Scholes differential operator $\mathbf{A}$ proves quite difficult for stating anything conclusive for the scaling of these parameters with problem size. Finally, we provided some numerical evidence for the scaling of the condition numbers associated with both the proper and improper Black-Scholes differential operators, indicating a far more rapid scaling of the condition number associated with the proper operator versus the improper operator. We also show favourable scaling of the spectral norm for the improper operator versus the proper operator. Although we showed analytically that the condition number is 1 for the case of the improper operator, the plot indicates some change in this value for increasing N and d. We believe this is due to a numerical error of the program rather then a genuine phenomenon.

5.3 Decomposition and sparsity determination of matrix

An indispensable procedure required for many algorithms, critical within the field of quantum chemistry and quantum dynamics [16] is Hamiltonian simulation. Broadly speaking, this procedure addresses for a given Hamiltonian $H \in \mathbb{C}^{2^n \times 2^n}$, how can the unitary evolution generated by H, $U = e^{-iHt}$, be encoded into a quantum circuit with a standard universal gate set. One of the first techniques developed is Trotter Suzuki decomposition [23]. The Trotter Suzuki decomposition is derived from the Lie product formula [21].

$$e^{A+B} = \lim_{n\to\infty} \left(e^{A/n} e^{B/n}\right)^n$$

To see how this works in practice take a Hamiltonian $\hat{H}$ and its decomposition into a sum of *simpler* Hamiltonians $\hat{H}_i$ that are easier to simulate in terms of a universal gate set (i.e. Clifford group + T-Gate)[36].

$$\hat{H} = \sum_{i=1}^{n} \hat{H}_i \qquad \text{such that} \qquad \underbrace{U_k(t) := e^{-i\hat{H}_k t}}_{\text{Circuit representation easy / known}}$$

Simpler in this context implies that either the circuit representation of $\hat{H}_i$ is known before hand or each Hamiltonian is *k-local*, implying it acts on at most k qubits. Suppose we want to simulate unitary evolution generated by $\hat{H}$ for a time duration t. Fix n sufficiently large and define the timestep $h := \frac{t}{n}$.

$$U = e^{-i\hat{H}t} = e^{\left(\sum_{k=1}^{n} -i\hat{H}_k t\right)} \approx \left(\prod_{k=1}^{n} e^{-\frac{i\hat{H}_k t}{n}}\right)^n = \left(\prod_{k=1}^{n} U_k\left(\frac{t}{n}\right)\right)^n$$

The first step in using this algorithm is choosing a certain set of Hamiltonians $\{H_i\}_{i\in I}$. We will decompose the matrix of interest into a 'pseudo'-Pauli basis in the next section. In [5] and [35], the sparsity of the matrix plays a fundamental role in the overall complexity of the algorithm. Sparsity is defined in this context as the maximum number of non zero elements in any one row or column:

$$\mathrm{Spar}(A) := \max\left(\max_j\left(\sum_k \mathbb{1}_{[a_{jk}\neq 0]}\right), \max_j\left(\sum_k \mathbb{1}_{[a_{kj}\neq 0]}\right)\right)$$

where $\mathbb{1}_{[(.)]}$ refers to the Iverson bracket:

$$\mathbb{1}_{[x]} = \begin{cases} 1 & \text{if } x \text{ is True} \\ 0 & \text{if } x \text{ is False} \end{cases}$$

From this definition of sparsity, we use the following inequalities which will be useful in bounding the sparsity of the overall matrix to be inverted with the HHL algorithm:

$$\mathrm{Spar}(A\otimes B) \leq \mathrm{Spar}(A)\mathrm{Spar}(B)$$

$$\mathrm{Spar}(A+B) \leq \mathrm{Spar}(A) + \mathrm{Spar}(B)$$

We provide a proof of these two inequalities in appendix (B). We will now proceed with the Pauli decomposition decomposing the matrix $\tilde{\Lambda}$. This will enable us to use the two sparsity inequalities above to ascertain an upper bound for $\mathrm{Spar}(\tilde{\Lambda})$.

5.3.1 Pauli decomposition of $\tilde{\Lambda}$

If we return to the matrix to be inverted:

$$\left(\begin{array}{ccccccccccc}
I &&&&&&&&&& \\
-A\Delta t & \ddots &&&&&&&&& \\
&& \ddots &&&&&&&& \\
&& -\frac{A\Delta t}{2^m-1} & I &&&&&&& \\
-I & \dots && -I & I &&&&&& \\
&&&& -A\Delta t & \ddots &&&&& \\
&&&&&& \ddots &&&& \\
&&&&&& -\frac{A\Delta t}{2^m-1} & I &&& \\
&&&& -I & \dots && -I & I && \\
&&&&&&&& -I & I & \\
&&&&&&&&& \ddots & \ddots \\
&&&&&&&&& -I & I
\end{array}\right)
\left(\begin{array}{c}
\vec{x}_1 \\ \vec{x}_2 \\ \vdots \\ \vec{x}_n \\ \hline \vec{x}_{n+1} \\ \vec{x}_{n+2} \\ \vdots \\ \vec{x}_{2n-1} \\ \hline \vec{x}_{2n} \\ \vec{w} \\ \vdots \\ \vec{w}
\end{array}\right)
=
\left(\begin{array}{c}
\vec{V}_0 \\ (\Delta t)\vec{V}_1 \\ \vdots \\ \vec{0} \\ \hline \vec{0} \\ (\Delta t)\vec{V}_1 \\ \vdots \\ \vec{0} \\ \hline \vec{0} \\ \vec{0} \\ \vdots \\ \vec{0}
\end{array}\right) \tag{5.20}$$

For the matrix above, $\Lambda_{[2^r;2^p,2^m,2^n]}$ where $\dim(\mathbf{A}) = 2^n$, 2^m refers to the order which the analytic expression is approximated in each block, $2^p - 2^r$ refers to how many timesteps Δt the solution is evolved forward and 2^r refers to how many of those blocks are dedicated to repeating the final solution. If the solution is not repeated, then we take $2^r = 0$. The dimension of the matrix $\Lambda_{[2^r;2^p,2^m,2^n]}$ is 2^{p+m+n}. Furthermore, the matrix $\Lambda_{[2^r;2^p,2^m,2^n]}$, in its current form is not Hermitian and hence can not be realised as a valid unitary operator. However, if we represent the linear system above to the equation $\left(\Lambda_{[2^r;2^p,2^m,2^n]}\right)\vec{x} = \vec{b}$, we can solve a 'Hermitian' form of the matrix above:

$$\underbrace{\begin{pmatrix} \mathbf{0} & \Lambda_{[2^r;2^p,2^m,2^n]} \\ \Lambda^{\dagger}_{[2^r;2^p,2^m,2^n]} & \mathbf{0} \end{pmatrix}}_{\tilde{\Lambda}_{[2^r;2^p,2^m,2^n]}} \begin{pmatrix} \vec{0} \\ \vec{x} \end{pmatrix} = \begin{pmatrix} \vec{b} \\ \vec{0} \end{pmatrix}$$

$$\tilde{\Lambda}_{[2^r;2^p,2^m,2^n]} = \frac{1}{2}\left((\sigma_1 + i\sigma_2) \otimes \Lambda_{[2^r;2^p,2^m,2^n]} + (\sigma_1 - i\sigma_2) \otimes \Lambda^{\dagger}_{[2^r;2^p,2^m,2^n]}\right)$$

$$\tilde{\Lambda}_{[2^r;2^p,2^m,2^n]} = \sigma_+ \otimes \Lambda_{[2^r;2^p,2^m,2^n]} + \sigma_- \otimes \Lambda^{\dagger}_{[2^r;2^p,2^m,2^n]}$$

We now seek to identify the decomposition of this matrix in terms of the Pauli basis. However, as finding a decomposition for the matrix $\mathbf{A}$ will automatically lead to a decomposition for $\tilde{\mathbf{A}}$, we proceed with finding a decomposition for the original matrix $\Lambda_{[2^r;2^p,2^m,2^n]}$. For all $M \in M_n(\mathbb{C})$, there exists a decomposition of the form:

$$M = \frac{1}{2^n} \sum_{\vec{K} \in \{I,X,Y,Z\}^{\otimes n}} \mathrm{Tr}(\sigma_{\vec{K}} M)\sigma_{\vec{K}} \tag{5.21}$$

However, computationally this requires a super-exponential number of trace calculations ($\mathcal{O}(n4^n)$).This can be seen as we need to 4^n trace for the matrix, we have to define a smaller classes of matrix and 'build' them up to the matrix $\Lambda_{[2^r;2^p,2^m,2^n]}$. To proceed we define same basic 2×2 matrices:

$$\widehat{\Pi}_0 = \frac{1}{2}(\sigma_0 + \sigma_3) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad \widehat{\Pi}_1 = \frac{1}{2}(\sigma_0 - \sigma_3) = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

$$\widehat{\sigma}_+ = \frac{1}{2}(\sigma_1 + i\sigma_2) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \quad \widehat{\sigma}_- = \frac{1}{2}(\sigma_1 - i\sigma_2) = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$$

$\mathbf{C}_1$ matrix

The first matrix that we need to be able to construct is given by:

$$(\mathbf{C}_1)_{[2^n]}(a_0, a_1, ..., a_{2^n-2}) := \underbrace{\begin{pmatrix} 0 & & & & \\ a_0 & 0 & & & \\ 0 & a_1 & \ddots & & \\ & & \ddots & & \\ & & & a_{2^n-2} & 0 \end{pmatrix}}_{2^n}$$

This matrix has the following analytic form:

$$\begin{aligned}
&(\mathbf{C}_1)_{[2^n]}(a_0, a_1, ..., a_{2^n-2}) = \\
&\qquad + \sum_{k_1=0}^{1} \cdots \sum_{k_{n-1}=0}^{1} a_{\left(\sum_{t=1}^{n-1} k_t 2^t\right)} \left(\bigotimes_{t=1}^{n-1} \widehat{\Pi}_{k_{n-t}} \right) \otimes \sigma_- \\
&\qquad + \sum_{l=2}^{n-1} \left(\sum_{k_l=0}^{1} \cdots \sum_{k_{n-1}=0}^{1} a_{\left(2^{l-1}-1+\sum_{t=l}^{n-1} k_t 2^t\right)} \left(\bigotimes_{t=1}^{n-l} \widehat{\Pi}_{k_{n-t}} \right) \otimes \sigma_-^{\otimes(l-1)} \otimes \sigma_+ \right) \\
&\qquad + a_{2^{n-1}-1}(\sigma_-^{\otimes(n-1)} \otimes \sigma_+)
\end{aligned}$$

The recursive relationship between these matrices is given by:

$$(\mathbf{C}_1)_{[2^{n+1}]}(a_0, a_1, ..., a_{2^{n+1}-2}) = \tag{5.22}$$

$$+\sum_{k_n=0}^{1} \widehat{\Pi}_{k_n} \otimes \big((\mathbf{C}_1)_{[2^n]}(a_{0+k_n 2^n}, a_{1+k_n 2^n}, ..., a_{2^n-2+k_n 2^n})\big) \tag{5.23}$$

$$+\, a_{2^n-1}(\sigma_-^{\otimes n} \otimes \sigma_+) \tag{5.24}$$

In matrix form, this recursive relationship has the form:

$$(\mathbf{C}_1)_{[2^{n+1}]}(a_0, a_1, ..., a_{2^{n+1}-2}) := \underbrace{\begin{pmatrix} 0 & & & & \\ a_0 & 0 & & & \\ 0 & a_1 & \ddots & & \\ & & \ddots & & \\ & & & a_{2^{n+1}-2} & 0 \end{pmatrix}}_{\text{dim}=2^{n+1}} =$$

$$\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \otimes \underbrace{\begin{pmatrix} 0 & & & & \\ a_0 & 0 & & & \\ 0 & a_1 & \ddots & & \\ & & \ddots & & \\ & & & a_{2^n-2} & 0 \end{pmatrix}}_{(\mathbf{C}_1)_{[2^n]}(a_0, a_1, ..., a_{2^n-2})} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \otimes \underbrace{\begin{pmatrix} 0 & & & & \\ a_{2^n} & 0 & & & \\ 0 & a_{2^n+1} & \ddots & & \\ & & \ddots & & \\ & & & a_{2^{n+1}-2} & 0 \end{pmatrix}}_{(\mathbf{C}_1)_{[2^n]}(a_{2^n}, a_{2^n+1}, ..., a_{2^{n+1}-2})}$$

$$+ \quad a_{2^n-1} \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}^{\otimes n} \otimes \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} \tag{5.25}$$

Using the above recursion relation, we can verify the correctness of the closed form formula of (5.22) (where we highlight in blue the $(2^n)^{th}$ closed form expression inserted into the recursive relationship):

$$
\begin{aligned}
&(\mathbf{C}_1)_{[2^{n+1}]}(a_0, a_1, ..., a_{2^{n+1}-2}) = \\
&\quad = \sum_{k_n=0}^{1} \widehat{\Pi}_{k_n} \otimes \left(\sum_{k_1=0}^{1} ... \sum_{k_{n-1}=0}^{1} a_{\left(\sum_{t=1}^{n-1} k_t 2^t + k_n 2^n\right)} \left(\bigotimes_{t=1}^{n-1} \widehat{\Pi}_{k_{n-t}} \right) \otimes \sigma_- \right. \\
&\quad + \sum_{l=2}^{n-1} \Big(\sum_{k_l=0}^{1} ... \sum_{k_{n-1}=0}^{1} a_{\left(2^{l-1}-1+\sum_{t=l}^{n-1} k_t 2^t + k_n 2^n\right)} \left(\bigotimes_{t=1}^{n-l} \widehat{\Pi}_{k_{n-t}} \right) \otimes \sigma_-^{\otimes(l-1)} \otimes \sigma_+ \Big) \\
&\quad \left. + a_{(2^{n-1}-1+k_n 2^n)} (\sigma_-^{\otimes(n-1)} \sigma_+) \right) + a_{2^n-1}(\sigma_-^{\otimes n} \otimes \sigma_+)
\end{aligned}
\tag{5.26}
$$

Using the identities:

$$
\widehat{\Pi}_{k_n} \otimes \left(\left(\bigotimes_{t=1}^{n-1} \widehat{\Pi}_{k_{n-t}} \right) \otimes (...) \right) = \left(\bigotimes_{t=1}^{(n+1)-1} \widehat{\Pi}_{k_{(n+1)-t}} \right) \otimes (...)
$$

$$
\widehat{\Pi}_{k_n} \otimes \left(\left(\bigotimes_{t=1}^{n-l} \widehat{\Pi}_{k_{n-t}} \right) \otimes (...) \right) = \left(\bigotimes_{t=1}^{(n+1)-l} \widehat{\Pi}_{k_{(n+1)-t}} \right) \otimes (...)
$$

$$
\sum_{t=1}^{n-1} k_t 2^t + k_n 2^n = \sum_{t=1}^{(n+1)-1} k_t 2^t
$$

Applying these identities to (5.26) yields:

$$
\begin{aligned}
= \; & \sum_{k_n=0}^{1} \sum_{k_1=0}^{1} ... \sum_{k_{n-1}=0}^{1} a_{\left(\sum_{t=1}^{(n+1)-1} k_t 2^t\right)} \left(\bigotimes_{t=1}^{(n+1)-1} \widehat{\Pi}_{k_{(n+1)-t}} \right) \otimes \sigma_- \\
+ \; & \sum_{k_n=0}^{1} \sum_{l=2}^{n-1} \left(\sum_{k_l=0}^{1} ... \sum_{k_{n-1}=0}^{1} a_{\left(2^{l-1}-1+\sum_{t=l}^{(n+1)-1} k_t 2^t\right)} \left(\bigotimes_{t=1}^{(n+1)-l} \widehat{\Pi}_{k_{(n+1)-t}} \right) \right) \otimes \sigma_-^{\otimes(l-1)} \otimes \sigma_+ \\
+ \; & \sum_{k_n=0}^{1} a_{(2^{n-1}-1+k_n 2^n)} \left(\widehat{\Pi}_{k_n} \otimes \sigma_-^{\otimes n-1} \otimes \sigma_+ \right) + a_{2^n-1}(\sigma_-^{\otimes n} \otimes \sigma_+)
\end{aligned}
$$

Moving the summation $\sum_{k_n=0}^{1}(\_)$ inside the first parentheses yields :

$$
\begin{aligned}
= & \sum_{k_1=0}^{1} \cdots \sum_{k_n=0}^{1} a_{\left(\sum_{t=1}^{(n+1)-1} k_t 2^t\right)} \left(\bigotimes_{t=1}^{(n+1)-1} \widehat{\Pi}_{k_{(n+1)-t}} \right) \otimes \sigma_- \qquad (5.27)\\
+ & \sum_{l=2}^{n-1} \left(\sum_{k_l=0}^{1} \cdots \sum_{k_{n-1}=0}^{1} \sum_{k_n=0}^{1} a_{\left(\sum_{t=l}^{(n+1)-1} k_t 2^t + 2^{l-1} - 1\right)} \left(\bigotimes_{t=1}^{(n+1)-l} \widehat{\Pi}_{k_{(n+1)-t}} \right) \right) \otimes \sigma_-^{\otimes(l-1)} \otimes \sigma_+ \\
+ & \sum_{k_n=0}^{1} a_{(2^{n-1}-1+k_n 2^n)} \left(\widehat{\Pi}_{k_n} \otimes \sigma_-^{\otimes n-1} \otimes \sigma_+ \right) + a_{2^n-1}(\sigma_-^{\otimes n} \otimes \sigma_+)
\end{aligned}
$$

To complete the proof, we need to show how the term:

$$
\sum_{k_n=0}^{1} a_{(2^{n-1}-1+k_n 2^n)} \left(\widehat{\Pi}_{k_n} \otimes \sigma_-^{\otimes n-1} \otimes \sigma_+ \right)
$$

can be interpreted as the n^{th} term in the second summation (5.28):

$$
\sum_{l=2}^{n-1} \left(\sum_{k_l=0}^{1} \cdots \sum_{k_{n-1}=0}^{1} \sum_{k_n=0}^{1} a_{\left(\sum_{t=l}^{(n+1)-1} k_t 2^t + 2^{l-1} - 1\right)} \left(\bigotimes_{t=1}^{(n+1)-l} \widehat{\Pi}_{k_{(n+1)-t}} \right) \right) \otimes \sigma_-^{\otimes(l-1)} \otimes \sigma_+
$$

If we take $l = n$, we can see equality for the index value $(\_)$ in $a_{(\_)}$

$$
\begin{aligned}
& \sum_{t=n}^{(n+1)-1} k_t 2^t + 2^{l-1} - 1 = 2^{n-1} - 1 + k_n 2^n \\
& \Rightarrow a_{\left(\sum_{t=n}^{(n+1)-1} k_t 2^t + 2^{l-1} - 1\right)} = a_{(2^{n-1}-1+k_n 2^n)}
\end{aligned}
$$

Focusing on the tensor product term, we can also see equality:

$$
\Rightarrow \left(\bigotimes_{t=1}^{(n+1)-n} \widehat{\Pi}_{k_{(n+1)-t}} \right) \otimes \sigma_-^{\otimes(l-1)} \otimes \sigma_+ = \widehat{\Pi}_{k_n} \otimes \sigma_-^{\otimes n-} \otimes \sigma_+
$$

Therefore, we can conclude that it is in fact the n^{th} term in the summation:

$$
\underbrace{\sum_{k_n=0}^{1} a_{(2^{n-1}-1+k_n 2^n)} \left(\widehat{\Pi}_{k_n} \otimes \sigma_-^{\otimes n-} \otimes \sigma_+ \right)}_{n^{th} \text{ term}}
$$

Hence, the penultimate term can be included in the second summation to yield:

$$
\begin{aligned}
= & \sum_{k_1=0}^{1} \cdots \sum_{k_{(n+1)-1}=0}^{1} a_{\left(\sum_{t=1}^{(n+1)-1} k_t 2^t\right)} \left(\bigotimes_{t=1}^{(n+1)-1} \widehat{\Pi}_{k_{(n+1)-t}} \right) \otimes \sigma_- \\
+ & \sum_{l=2}^{(n+1)-1} \left(\sum_{k_l=0}^{1} \cdots \sum_{k_n=0}^{1} a_{\left(\sum_{t=l}^{(n+1)-1} k_t 2^t + 2^{l-1} - 1\right)} \left(\bigotimes_{t=1}^{(n+1)-l} \widehat{\Pi}_{k_{(n+1)-t}} \right) \right) \otimes \sigma_-^{\otimes(l-1)} \otimes \sigma_+ \\
+ & a_{2^n-1} (\sigma_-^{\otimes n} \otimes \sigma_+)
\end{aligned}
$$

$\mathbf{C}_2$ matrix

The last matrix that we will have to construct is the G matrix which we define as:

$$
(\mathbf{C}_2)_{[2^n]} := \underbrace{\begin{pmatrix} 1 & 1 & \dots & 1 \\ 0 & 0 & \dots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix}}_{\text{dim}=2^n} = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}^{\otimes n} = \left(\widehat{\Pi}_0 + \sigma_+ \right)^{\otimes n}
$$

Construction of $\mathbf{C}_3$, $\mathbf{C}_4$ & $\mathbf{C}_5$ matrices

Assume that for the initial matrix $\mathbf{A}$, $\dim(\mathbf{A}) = 2^n$. This will be a necessary constraint as we are assuming that we are dealing with a quantum circuit comprising solely of qubits (2-dimensional Hilbert space). Consider the matrix:

$$
(\mathbf{C}_3)_{[2^m,2^n]} := -(\mathbf{C}_1)_{[2^m]} \left(\Delta t, \frac{\Delta t}{2}, ..., \frac{\Delta t}{2^m - 1} \right) \otimes \mathbf{A}_{2^n} \tag{5.28}
$$

$$
\underbrace{\begin{pmatrix} \mathbf{0} & & & & \\ -\mathbf{A}_{2^n}\Delta t & \mathbf{0} & & & \\ & -\frac{\mathbf{A}_{2^n}\Delta t}{2} & \ddots & & \\ & & \ddots & & \\ & & & -\frac{\mathbf{A}_{2^n}\Delta t}{2^m-1} & \mathbf{0} \end{pmatrix}}_{\substack{\text{dim}=2^{m+n} \\ \text{No of blocks}=2^m}} := - \underbrace{\begin{pmatrix} \mathbf{0} & & & & \\ \Delta t & \mathbf{0} & & & \\ \mathbf{0} & \frac{\Delta t}{2} & \ddots & & \\ & & \ddots & & \\ & & & \frac{\Delta t}{2^m-1} & \mathbf{0} \end{pmatrix}}_{\text{dim}=2^m} \otimes \underbrace{\mathbf{A}_{2^n}}_{\text{dim}=2^n}
$$

$(\mathbf{C}_1)_{[2^m]}(\Delta t, \frac{\Delta t}{2}, ..., \frac{\Delta t}{2^m-1}) =$

$$
\begin{aligned}
\to\ & \sum_{k_1=0}^{1} \cdots \sum_{k_{m-1}=0}^{1} \frac{\Delta t}{\left(\sum_{t=1}^{m-1} k_t 2^t + 1\right)} \left(\bigotimes_{t=1}^{m-1} \widehat{\Pi}_{k_{m-t}} \right) \otimes \sigma_- \\
+\ & \sum_{l=2}^{m-1} \left(\sum_{k_l=0}^{1} \cdots \sum_{k_{m-1}=0}^{1} \frac{h}{\left(\sum_{t=l}^{m-1} k_t 2^t + 2^{l-1}\right)} \left(\bigotimes_{t=1}^{m-l} \widehat{\Pi}_{k_{m-t}} \right) \otimes \sigma_-^{\otimes(l-1)} \otimes \sigma_+ \right) \\
+\ & \frac{\Delta t}{2^{m-1}} (\sigma_-^{\otimes(m-1)} \otimes \sigma_+)
\end{aligned}
$$

Hence $(\mathbf{C}_3)_{[2^m,2^n]} =$

$$
\begin{aligned}
-\ & \sum_{k_1=0}^{1} \cdots \sum_{k_{m-1}=0}^{1} \frac{\Delta t}{\left(\sum_{t=1}^{m-1} k_t 2^t + 1\right)} \left(\bigotimes_{t=1}^{m-1} \widehat{\Pi}_{k_{m-t}} \right) \otimes \sigma_- \otimes \mathbf{A}_n \\
-\ & \sum_{l=2}^{m-1} \left(\sum_{k_l=0}^{1} \cdots \sum_{k_{m-1}=0}^{1} \frac{\Delta t}{\left(\sum_{t=l}^{m-1} k_t 2^t + 2^{l-1}\right)} \left(\bigotimes_{t=1}^{m-l} \widehat{\Pi}_{k_{m-t}} \right) \right) \otimes \sigma_-^{\otimes(l-1)} \otimes \sigma_+ \otimes \mathbf{A}_n \\
-\ & \frac{\Delta t}{2^{m-1}} (\sigma_-^{\otimes(m-1)} \otimes \sigma_+) \otimes \mathbf{A}_n
\end{aligned}
$$

Here we omit the Pauli decomposition of the matrix $\mathbf{A}_n$, however as shown earlier in the thesis $\mathbf{A}$ is composed of a sum of tensor products of tridiagonal Toeplitz matrices, each of which has an analytic formula for their decomposition in the Pauli basis as provided earlier. Hence, an analytic form for the complete description exists. For the sake of brevity, we omit the description here. Returning to the original assumption that the matrix $\mathbf{A}$ has dimension 2^n. Consider the matrix:

$$(\mathbf{C}_4)_{[2^m,2^n]} = (\mathbf{C}_2)_{[2^m]} \otimes \mathbf{I}_{2^n} \tag{5.29}$$

$$(\mathbf{C}_4)_{[2^m,2^n]} = \left(\widehat{\Pi}_0 + \sigma_+\right)^{\otimes m} \otimes \sigma_0^{\otimes n}$$

$$
\underbrace{\begin{pmatrix} \mathbf{I}_{2^n} & \mathbf{I}_{2^n} & \dots & \mathbf{I}_{2^n} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & & & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \end{pmatrix}}_{\substack{\text{dim}=2^{m+n} \\ \text{No of blocks}=2^m}} := \begin{pmatrix} 1 & 1 & \dots & 1 \\ 0 & 0 & \dots & 0 \\ \vdots & & & \vdots \\ 0 & 0 & \dots & 0 \end{pmatrix} \otimes \mathbf{I}_{2^n}
$$

Finally, we can start to reproduce the actual matrix. Let the $(\mathbf{C}_5)_{[2^m,2^n]}$ matrix be defined as:

$$(\mathbf{C}_5)_{[2^m,2^n]} = (\mathbf{C}_3)_{[2^m,2^n]} + \underbrace{\mathbf{I}_{2^m} \otimes \mathbf{I}_{2^n}}_{\sigma_0^{\otimes(m+n)}} \tag{5.30}$$

$$(\mathbf{C}_5)_{[2^m,2^n]} := \underbrace{\begin{pmatrix} \mathbf{I}_{2^n} & & & & \\ -\mathbf{A}_{2^n}\Delta t & \mathbf{I}_{2^n} & & & \\ & -\frac{\mathbf{A}_{2^n}\Delta t}{2} & & & \\ & & \ddots & \ddots & \\ & & & -\frac{\mathbf{A}_{2^n}\Delta t}{2^m-1} & \mathbf{I}_{2^n} \end{pmatrix}}_{\substack{\text{dim}=2^{m+n}\\ \text{No of blocks}=2^m}}$$

$$\Rightarrow \underbrace{\begin{pmatrix} \mathbf{0} & & & & \\ -\mathbf{A}_{2^n}\Delta t & \mathbf{0} & & & \\ & -\frac{\mathbf{A}_{2^n}\Delta t}{2} & \ddots & & \\ & & \ddots & & \\ & & & -\frac{\mathbf{A}_{2^n}\Delta t}{2^m-1} & \mathbf{0} \end{pmatrix}}_{\substack{\text{dim}=2^{m+n}\\ \text{No of blocks}=2^m}} + \underbrace{\begin{pmatrix} \mathbf{I}_{2^n} & & & & \\ & \mathbf{I}_{2^n} & & & \\ & & \mathbf{I}_{2^n} & & \\ & & & \ddots & \\ & & & & \mathbf{I}_{2^n} \end{pmatrix}}_{\substack{\text{dim}=2^{m+n}\\ \text{No of blocks}=2^m}}$$

$(\mathbf{C}_5)_{[2^m,2^n]} =$

$$\begin{aligned} \rightarrow\ & \sum_{k_1=0}^{1}\cdots\sum_{k_{m-1}=0}^{1} \frac{\Delta t}{\left(\sum_{t=1}^{m-1} k_t 2^t + 1\right)} \left(\bigotimes_{t=1}^{m-1} \widehat{\Pi}_{k_{m-t}}\right) \otimes \sigma_- \otimes \mathbf{A}_{2^n} \\ +\ & \sum_{l=2}^{m-1} \left(\sum_{k_l=0}^{1}\cdots\sum_{k_{m-1}=0}^{1} \frac{\Delta t}{\left(\sum_{t=l}^{m-1} k_t 2^t + 2^{l-1}\right)} \left(\bigotimes_{t=1}^{m-l} \widehat{\Pi}_{k_{m-t}}\right) \otimes \sigma_-^{\otimes(l-1)} \otimes \sigma_+ \otimes \mathbf{A}_{2^n} \right) \\ +\ & \frac{\Delta t}{2^{m-1}} (\sigma_-^{\otimes(m-1)} \otimes \sigma_+) \otimes \mathbf{A}_{2^n} + \sigma_0^{\otimes(m+n)} \end{aligned}$$

Construction of the Λ matrix without repetition of solution

Consider the block matrix $(\mathbf{C}_6)_{[2^2,2^m,2^n]}$ for the case of 2^2 'blocks' along the diagonal, each of dimension 2^{m+n}:

$$(\mathbf{C}_6)_{[2^2,2^m,2^n]}\left((\mathbf{C}_5)_{[2^m,2^n]},(\mathbf{C}_4)_{[2^m,2^n]}\right):=$$

$$\begin{pmatrix} (\mathbf{C}_5)_{[2^m,2^n]} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ (\mathbf{C}_4)_{[2^m,2^n]} & (\mathbf{C}_5)_{[2^m,2^n]} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & (\mathbf{C}_4)_{[2^m,2^n]} & (\mathbf{C}_5)_{[2^m,2^n]} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & (\mathbf{C}_4)_{[2^m,2^n]} & (\mathbf{C}_5)_{[2^m,2^n]} \end{pmatrix}$$

This block structure is encountered often in our construction. Its shorthand notation appears as follows:

$$(\mathbf{C}_6)_{[2^p,2^m,2^n]}\left(\mathbf{G}_{2^{m+n}},\mathbf{H}_{2^{m+n}}\right):=\mathbf{I}_{2^p}\otimes\mathbf{G}_{2^{m+n}}+(\mathbf{C}_2)_{[2^p]}\otimes\mathbf{H}_{2^{m+n}} \tag{5.31}$$

$$=\underbrace{\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix}}_{\text{dim}=2^p}\otimes\mathbf{G}_{2^{m+n}}-\underbrace{\begin{pmatrix} 0 & & & \\ 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{pmatrix}}_{\text{dim}=2^p}\otimes\mathbf{H}_{2^{m+n}}$$

$$=\begin{pmatrix} \mathbf{G}_{2^{m+n}} & & & \\ & \mathbf{G}_{2^{m+n}} & & \\ & & \ddots & \\ & & & \mathbf{G}_{2^{m+n}} \end{pmatrix}-\begin{pmatrix} \mathbf{0} & & & \\ \mathbf{H}_{2^{m+n}} & \mathbf{0} & & \\ & \ddots & \ddots & \\ & & \mathbf{H}_{2^{m+n}} & \mathbf{0} \end{pmatrix}$$

$$=\begin{pmatrix} \mathbf{G}_{2^{m+n}} & & & \\ -\mathbf{H}_{2^{m+n}} & \mathbf{G}_{2^{m+n}} & & \\ & \ddots & \ddots & \\ & & -\mathbf{H}_{2^{m+n}} & \mathbf{G}_{2^{m+n}} \end{pmatrix}$$

where $\mathbf{G}_{2^{m+n}}$, $\mathbf{H}_{2^{m+n}}$ are dummy matrices. Replacing these matrices with the matrices $(\mathbf{C}_5)_{[2^m,2^n]}, (\mathbf{C}_4)_{[2^m,2^n]}$ respectively yields the block matrix of current interest:

$$(\mathbf{C}_7)_{[2^p,2^m,2^n]} := (\mathbf{C}_6)_{[2^p,2^m,2^n]}\left((\mathbf{C}_5)_{[2^m,2^n]}, (\mathbf{C}_4)_{[2^m,2^n]}\right)$$

which explicitly in matrix notation appears as:

$$= \underbrace{\begin{pmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix}}_{\text{dim}=2^p} \otimes \underbrace{\begin{pmatrix} \mathbf{I}_{2^n} & & & \\ -\mathbf{A}_{2^n}\Delta t & \mathbf{I}_{2^n} & & \\ & \ddots & \ddots & \\ & & -\frac{\mathbf{A}_{2^n}\Delta t}{2^m-1} & \mathbf{I}_{2^n} \end{pmatrix}}_{\substack{\text{dim}=2^{m+n}\\ \text{No of blocks}=2^m}} - \underbrace{\begin{pmatrix} 0 & & & \\ 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{pmatrix}}_{\text{dim}=2^p} \otimes \underbrace{\begin{pmatrix} \mathbf{I}_{2^n} & \mathbf{I}_{2^n} & \dots & \mathbf{I}_{2^n} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \\ \vdots & & & \vdots \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} \end{pmatrix}}_{\substack{\text{dim}=2^{m+n}\\ \text{No of blocks}=2^m}}$$

Explicitly, in equation form, is given as: $(\mathbf{C}_7)_{[2^p,2^m,2^n]} =$

$$\begin{aligned}
\to\ & \sum_{k_1=0}^{1}\cdots\sum_{k_{m-1}=0}^{1} \frac{\Delta t}{\left(\sum_{t=1}^{m-1} k_t 2^t + 1\right)} \sigma_0^{\otimes p} \otimes \left(\bigotimes_{t=1}^{m-1} \widehat{\Pi}_{k_{m-t}}\right) \otimes \sigma_- \otimes \mathbf{A}_{2^n} \\
+\ & \sum_{l=2}^{m-1} \sigma_0^{\otimes p} \left(\sum_{k_l=0}^{1}\cdots\sum_{k_{m-1}=0}^{1} \frac{\Delta t}{\left(\sum_{t=l}^{m-1} k_t 2^t + 2^{l-1}\right)} \left(\bigotimes_{t=1}^{m-l} \widehat{\Pi}_{k_{m-t}}\right)\right) \otimes \sigma_-^{\otimes(l-1)} \otimes \sigma_+ \otimes \mathbf{A}_{2^n} \\
+\ & \frac{\Delta t}{2^{m-1}}\left(\sigma_0^{\otimes p} \otimes \sigma_-^{\otimes(m-1)} \otimes \sigma_+\right) \otimes \mathbf{A}_{2^n} + \sigma_0^{\otimes(m+n+p)} \\
+\ & \sum_{l=0}^{n-1} \sigma_0^{\otimes(n-1-l)} \otimes \sigma_- \otimes \sigma_+^{\otimes l} \otimes \left(\widehat{\Pi}_0 + \sigma_+\right)^{\otimes m} \otimes \sigma_0^n
\end{aligned}$$

To aid in understanding the structure of the matrix, we colour it as:

$$(\mathbf{C}_7)_{[2^p,2^m,2^n]} = (\mathbf{C}_6)_{[2^p,2^m,2^n]}\left((\mathbf{C}_5)_{[2^m,2^n]}, (\mathbf{C}_4)_{[2^m,2^n]}\right)$$

$$\begin{aligned}
\to\ & \sum_{k_1=0}^{1}\cdots\sum_{k_{m-1}=0}^{1} \frac{\Delta t}{\left(\sum_{t=1}^{m-1} k_t 2^t + 1\right)} \sigma_0^{\otimes p} \otimes \left(\bigotimes_{t=1}^{m-1} \widehat{\Pi}_{k_{m-t}}\right) \otimes \sigma_- \otimes \mathbf{A}_{2^n} \\
+\ & \sum_{l=2}^{m-1} \sigma_0^{\otimes p} \otimes \left(\sum_{k_l=0}^{1}\cdots\sum_{k_{m-1}=0}^{1} \frac{\Delta t}{\left(\sum_{t=l}^{m-1} k_t 2^t + 2^{l-1}\right)} \left(\bigotimes_{t=1}^{m-l} \widehat{\Pi}_{k_{m-t}}\right)\right) \otimes \sigma_-^{\otimes(l-1)} \otimes \sigma_+ \otimes \mathbf{A}_{2^n} \\
+\ & \frac{\Delta t}{2^{m-1}}\left(\sigma_0^{\otimes p} \otimes \sigma_-^{\otimes(m-1)} \otimes \sigma_+\right) \otimes \mathbf{A}_{2^n} + \sigma_0^{\otimes p} \otimes \sigma_0^{(m+n)} \\
+\ & \sum_{l=0}^{n-1} \sigma_0^{\otimes(n-1-l)} \otimes \sigma_- \otimes \sigma_+^{\otimes l} \otimes \left(\widehat{\Pi}_0 + \sigma_+\right)^{\otimes m} \otimes \sigma_0^n
\end{aligned}$$

The last step is to augment the the lower right corner of the matrix so that the solution is ‘repeated’. In matrix notation, this can represented as:

$$(\mathbf{C}_8)_{[2^r;2^p,2^m,2^n]} =$$

$$\left(\begin{array}{cccc|cccc}
(\mathbf{C}_5)_{[2^m,2^n]} & & & & & & & \\
(\mathbf{C}_4)_{[2^m,2^n]} & (\mathbf{C}_5)_{[2^m,2^n]} & & & & & & \\
 & \ddots & \ddots & & & & & \\
 & & (\mathbf{C}_4)_{[2^m,2^n]} & (\mathbf{C}_5)_{[2^m,2^n]} & & & & \\ \hline
 & & & (\mathbf{C}_4)_{[2^m,2^n]} & \mathbf{I}_{2^{m+n}} & & & \\
 & & & & -\mathbf{I}_{2^{m+n}} & \mathbf{I}_{2^{m+n}} & & \\
 & & & & & \ddots & \ddots & \\
 & & & & & & -\mathbf{I}_{2^{m+n}} & \mathbf{I}_{2^{m+n}}
\end{array}\right)$$

$$= \underbrace{\left(\begin{array}{cccc|cccc}
(\mathbf{C}_5) & & & & & & & \\
(\mathbf{C}_4) & (\mathbf{C}_5) & & & & & & \\
 & \ddots & \ddots & & & & & \\
 & & (\mathbf{C}_4) & (\mathbf{C}_5) & & & & \\ \hline
 & & & (\mathbf{C}_4) & (\mathbf{C}_5) & & & \\
 & & & & (\mathbf{C}_4) & (\mathbf{C}_5) & & \\
 & & & & & \ddots & \ddots & \\
 & & & & & & (\mathbf{C}_4) & (\mathbf{C}_5)
\end{array}\right)}_{(\mathbf{C}_7)_{[2^p,2^m,2^n]}}$$

$$
-\left(\begin{array}{cccc|cccc}
\mathbf{0} & & & & & & & \\
\mathbf{0} & \mathbf{0} & & & & & & \\
 & \ddots & \ddots & & & & & \\
 & & \mathbf{0} & \mathbf{0} & & & & \\
\hline
 & & & \mathbf{0} & (\mathbf{C}_5)-\mathbf{I} & & & \\
 & & & & (\mathbf{C}_4)+\mathbf{I} & (\mathbf{C}_5)-\mathbf{I} & & \\
 & & & & & \ddots & \ddots & \\
 & & & & & & (\mathbf{C}_4)+\mathbf{I} & (\mathbf{C}_5)-\mathbf{I}
\end{array}\right)
$$

To position an arbitrary matrix $\mathbf{A}$ with dimension $2^r \times 2^r$ at the bottom right corner of a matrix with zero entries everywhere of dimension $2^p \times 2^p$ to produce a matrix $\tilde{\mathbf{A}}$, we can express it as:

$$
\tilde{\mathbf{A}} = \left(\widehat{\Pi}_1\right)^{\otimes p-r} \otimes \mathbf{A}
$$

Or in matrix notation as:

$$
\tilde{\mathbf{A}} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}^{\otimes p-r} \otimes \mathbf{A} = \left(\begin{array}{c|c}
\begin{matrix}\mathbf{0} \\ (2^p-2^r \times 2^p-2^r)\end{matrix} & \begin{matrix}\mathbf{0} \\ (2^p-2^r \times 2^r)\end{matrix} \\
\hline
\begin{matrix}\mathbf{0} \\ (2^r \times 2^p-2^r)\end{matrix} & \begin{matrix}\mathbf{A} \\ (2^r \times 2^r)\end{matrix}
\end{array}\right)
$$

As the matrix $\mathbf{A}$ for the current case is:

$$
\mathbf{A} = \begin{pmatrix}
(\mathbf{C}_5)_{[2^m,2^n]} - \mathbf{I}_{2^{m+n}} & & & \\
(\mathbf{C}_4)_{[2^m,2^n]} + \mathbf{I}_{2^{m+n}} & (\mathbf{C}_5)_{[2^m,2^n]} - \mathbf{I}_{2^{m+n}} & & \\
 & \ddots & \ddots & \\
 & & (\mathbf{C}_4)_{[2^m,2^n]} + \mathbf{I}_{2^{m+n}} & (\mathbf{C}_5)_{[2^m,2^n]} - \mathbf{I}_{2^{m+n}}
\end{pmatrix}
$$

This can be expressed in an equation form adopting the previous notation of (5.31):

$$
\mathbf{A} = (\mathbf{C}_6)_{[2^r,2^m,2^n]} \left((\mathbf{C}_5)_{[2^m,2^n]} - \mathbf{I}_{2^{m+n}}, (\mathbf{C}_4)_{[2^m,2^n]} + \mathbf{I}_{2^{m+n}}\right)
$$

Wrapping it all together yields:

$$\begin{aligned}\Lambda_{[2^r;2^p,2^m,2^n]} := (\mathbf{C}_8)_{[2^r;2^p,2^m,2^n]} = (\mathbf{C}_7)_{[2^p,2^m,2^n]} - (\widehat{\Pi}_1)^{\otimes p-r} \\ \otimes (\mathbf{C}_6)_{[2^p,2^m,2^n]} \big((\mathbf{C}_5)_{[2^m,2^n]} - \mathbf{I}_{2^{m+n}}, (\mathbf{C}_4)_{[2^m,2^n]} + \mathbf{I}_{2^{m+n}}\big)\end{aligned}$$

Sparsity from pseudo-Pauli decomposition

We begin effectively backwards decomposing the matrix from the $(\mathbf{C}_8)$ matrix:

$$\begin{aligned}&\mathrm{Spar}\left((\mathbf{C}_8)_{[2^r;2^p,2^m,2^n]}\right) \leq \mathrm{Spar}\left((\mathbf{C}_7)_{[2^p,2^m,2^n]}\right) \\ &+ \mathrm{Spar}\left(\widehat{\Pi}_1\right)^{p-r} \mathrm{Spar}\left((\mathbf{C}_6)_{[2^p,2^m,2^n]}\left((\mathbf{C}_5)_{[2^m,2^n]} - \mathbf{I}_{2^{m+n}}, (\mathbf{C}_4)_{[2^m,2^n]} + \mathbf{I}_{2^{m+n}}\right)\right)\end{aligned}$$

Substituting that $\mathrm{Spar}\left(\widehat{\Pi}_1\right) = 1$ and recalling the definition of the $(\mathbf{C}_7)_{[2^p,2^m,2^n]}$ yields:

$$\begin{aligned}\mathrm{Spar}\left((\mathbf{C}_8)_{[2^r;2^p,2^m,2^n]}\right) &\leq \mathrm{Spar}\left((\mathbf{C}_6)_{[2^p,2^m,2^n]}\left((\mathbf{C}_5)_{[2^m,2^n]}, (\mathbf{C}_4)_{[2^m,2^n]}\right)\right) \\ &+ \mathrm{Spar}\left((\mathbf{C}_6)_{[2^p,2^m,2^n]}\left((\mathbf{C}_5)_{[2^m,2^n]} - \mathbf{I}_{2^{m+n}}, (\mathbf{C}_4)_{[2^m,2^n]} + \mathbf{I}_{2^{m+n}}\right)\right) \quad (5.32)\end{aligned}$$

Recalling the definition of the $(\mathbf{C}_6)_{[2^p,2^m,2^n]}$ matrix in (5.31), we can bound its sparsity as:

$$\begin{aligned}&\mathrm{Spar}\left((\mathbf{C}_6)_{[2^p,2^m,2^n]}(\mathbf{G}_{2^{m+n}}, \mathbf{H}_{2^{m+n}})\right) \leq \\ &\quad \mathrm{Spar}(\mathbf{I}_{2^p})\,\mathrm{Spar}(\mathbf{G}_{2^{m+n}}) + \mathrm{Spar}\left((\mathbf{C}_{[2^p]})\right)\mathrm{Spar}(\mathbf{H}_{2^{m+n}}) \\ &\quad = \mathrm{Spar}(\mathbf{G}_{2^{m+n}}) + \mathrm{Spar}(\mathbf{H}_{2^{m+n}})\end{aligned}$$

Applying to (5.32) yields:

$$\begin{aligned}(\mathbf{C}_8) &\leq \mathrm{Spar}\left((\mathbf{C}_5)_{[2^m,2^n]}\right) + \mathrm{Spar}\left((\mathbf{C}_5)_{[2^m,2^n]}\right) \\ &+ \mathrm{Spar}\left((\mathbf{C}_5)_{[2^m,2^n]} - \mathbf{I}_{2^{m+n}}\right) + \mathrm{Spar}\left((\mathbf{C}_4)_{[2^m,2^n]} + \mathbf{I}_{2^{m+n}}\right) \\ &= 2\mathrm{Spar}\left((\mathbf{C}_5)_{[2^m,2^n]}\right) + 2\mathrm{Spar}\left((\mathbf{C}_4)_{[2^m,2^n]}\right) + 2\end{aligned}$$

Recalling the definitions of $(\mathbf{C}_5)$ matrix in (5.30) and $(\mathbf{C}_4)$ in (5.29) bounds the sparsity as:

$$\begin{aligned}\text{Spar}\left((\mathbf{C}_5)_{[2^m,2^n]}\right) &\leq \text{Spar}\left((\mathbf{C}_3)_{[2^m,2^n]}\right) + 1\\ \text{Spar}\left((\mathbf{C}_4)_{[2^m,2^n]}\right) &\leq \text{Spar}\left((\mathbf{C}_2)_{[2^m]}\right)\end{aligned}$$

Therefore:

$$(\mathbf{C}_8)_{[2^r;2^p,2^m,2^n]} \leq 2\text{Spar}\left((\mathbf{C}_3)_{[2^m,2^n]}\right) + 2\text{Spar}\left((\mathbf{C}_2)_{[2^m]}\right) + 4$$

As the matrix $(\mathbf{C}_2)_{[2^m]}$ consists of $2^m \quad 1'$s all along its top row as well as recalling the definition of $(\mathbf{C}_3)_{[2^m,2^n]}$ in (5.28) yields:

$$(\mathbf{C}_8)_{[2^r;2^p,2^m,2^n]} \leq 2\text{Spar}(A) + 2^{m+1} + 4 \tag{5.33}$$

If we consider the case of the proper Black-Scholes differential operator:

$$\mathbf{A} = \left(\frac{1}{2}\sum_{i=1}^{d}\sigma_i^2 A_{(\partial^2 x_i)} + \sum_{\substack{i,j=1\\ j>i}}^{d}\rho_{ij}\sigma_i\sigma_j A_{(\partial x_i,\partial x_j)} + \sum_{i=1}^{d}\left(r - \frac{\sigma_i^2}{2}\right)A_{(\partial x_i)}\right)$$

$$\text{Spar}(\mathbf{A}) = d\left(\text{Spar}(A_{(\partial^2 x_i)})\right) + \frac{d(d-1)}{2}\left(\text{Spar}(A_{(\partial x_i,\partial x_j)})\right) + d\left(\text{Spar}(A_{(\partial x_i)})\right)$$

Recalling the definition of the matrices $A_{(\partial x_i)}$,$A_{(\partial^2 x_i)}$ and $A_{(\partial x_i,\partial x_j)}$:

$$A_{(\partial x_i)} = I^{\otimes d-i} \otimes D_1 \otimes I^{\otimes i-1} \qquad A_{(\partial^2 x_i)} = I^{\otimes d-i} \otimes D_2 \otimes I^{\otimes i-1}$$

$$A_{(\partial x_i,\partial x_j)} = I^{\otimes d-j} \otimes D_1 \otimes I^{\otimes j-i-1} \otimes D_1 \otimes I^{\otimes i-1}$$

Therefore, we can bound the sparsities as:

$$\text{Spar}\left(A_{(\partial x_i)}\right) \leq \text{Spar}\left(D_1\right) = \text{Spar}\left(\frac{1}{2h}\begin{pmatrix} 0 & -1 & & \\ 1 & 0 & \ddots & \\ & \ddots & \ddots & -1 \\ & & 1 & 0 \end{pmatrix}\right) = 2$$

$$\text{Spar}\left(A_{(\partial^2 x_i)}\right) \leq \text{Spar}\left(D_2\right) = \text{Spar}\left(\frac{1}{h^2}\begin{pmatrix} -2 & 1 & & \\ 1 & -2 & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{pmatrix}\right) = 3$$

$$\text{Spar}\left(A_{(\partial x_i, \partial x_j)}\right) \leq \left(\text{Spar}\left(D_1\right)\right)^2 \leq 4$$

Therefore, we bound the sparsity of the proper Black-Scholes differential operator as:

$$\text{Spar}(\mathbf{A}) \leq d(3) + \frac{d(d-1)}{2}(4) + d(2) = 2d^2 + 3d$$

Considering now the case of the improper Black-Scholes differential operator:

$$\tilde{\mathbf{A}} = \frac{1}{2}\sum_{i=1}^{d} \sigma_i^2 A^2_{(\partial x_i)} + \sum_{\substack{i,j=1 \\ j>i}}^{d} \rho_{ij} A_{(\partial x_i, \partial x_j)} + \sum_{i=1}^{d}\left(r - \frac{\sigma_i^2}{2}\right) A_{(\partial x_i)}$$

Proceeding in the same manner as before yields:

$$\begin{aligned} \text{Spar}\left(\tilde{\mathbf{A}}\right) &\leq d\left(\text{Spar}\left(A^2_{(\partial x_i)}\right)\right) + \frac{d(d-1)}{2}(4) + d(2) \\ &= d\left(\text{Spar}\left(A^2_{(\partial x_i)}\right)\right) + 2d^2 \end{aligned}$$

For arbitrary matrices, the sparsity measure Spar(_) is not sub-multiplicative (i.e. the product of two sparse matrices need not be sparse itself). Therefore we cannot bound $\text{Spar}\left(A^2_{(\partial x_i)}\right)$ in terms of $\text{Spar}\left(A_{(\partial x_i)}\right)$. The first bound we can place on $\text{Spar}\left(A^2_{(\partial x_i)}\right)$ is :

$$\text{Spar}\left(A^2_{(\partial x_i)}\right) = \text{Spar}\left(I^{\otimes d-i} \otimes D_1^2 \otimes I^{\otimes i-1}\right) \leq \text{Spar}\left(D_1^2\right)$$

However as D_1 is sparse and has a periodic structure, we can directly compute its square to see the general structure of the matrix:

$$(D_1)^2 = \left(\frac{1}{2h}\begin{pmatrix} 0 & -1 & & \\ 1 & 0 & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & 1 & 0 \end{pmatrix}\right)^2 = \frac{1}{4h^2}\begin{pmatrix} -1 & 0 & 1 & & \\ 0 & -2 & 0 & \ddots & \\ 1 & 0 & -2 & \ddots & 1 \\ & \ddots & \ddots & \ddots & 0 \\ & & 1 & 0 & -1 \end{pmatrix}$$

$$\text{Spar}\left(D_1^2\right) = 3$$

Therefore we can bound the sparsity of the improper Black-Scholes differential operator as:

$$\text{Spar}\left(\tilde{\mathbf{A}}\right) \leq 2d^2 + 3d$$

Therefore for either case of the improper or proper Black-Scholes differential operator, the sparsity of the $\mathbf{C}_8$ matrix in (5.33) can be finally bounded as:

$$\Lambda_{[2^r;2^p,2^m,2^n]} = (\mathbf{C}_8) \leq 4d^2 + 6d + 2^{m-1} + 4$$

As the final matrix to be inverted $\tilde{\Lambda}_{[2^r;2^p,2^m,2^n]}$ is just:

$$\tilde{\Lambda}_{[2^r;2^p,2^m,2^n]} = \begin{pmatrix} \mathbf{0} & \Lambda_{[2^r;2^p,2^m,2^n]} \\ \Lambda^\dagger_{[2^r;2^p,2^m,2^n]} & \mathbf{0} \end{pmatrix}$$

We have that $\text{Spar}\left(\tilde{\Lambda}_{[2^r;2^p,2^m,2^n]}\right) = \text{Spar}\left(\Lambda_{[2^r;2^p,2^m,2^n]}\right)$ and thus:

$$\text{Spar}\left(\tilde{\Lambda}_{[2^r;2^p,2^m,2^n]}\right) \leq 4d^2 + 6d + 2^{m-1} + 4$$

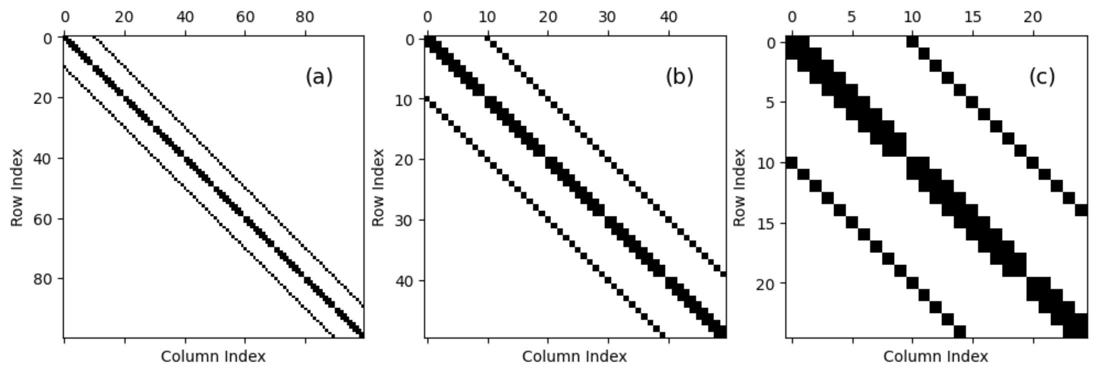


Figure 5.3: Sparsity structure of two dimensional Black-Scholes differential operator with 100 gridpoints at different levels of resolution (a), (b) and(c)

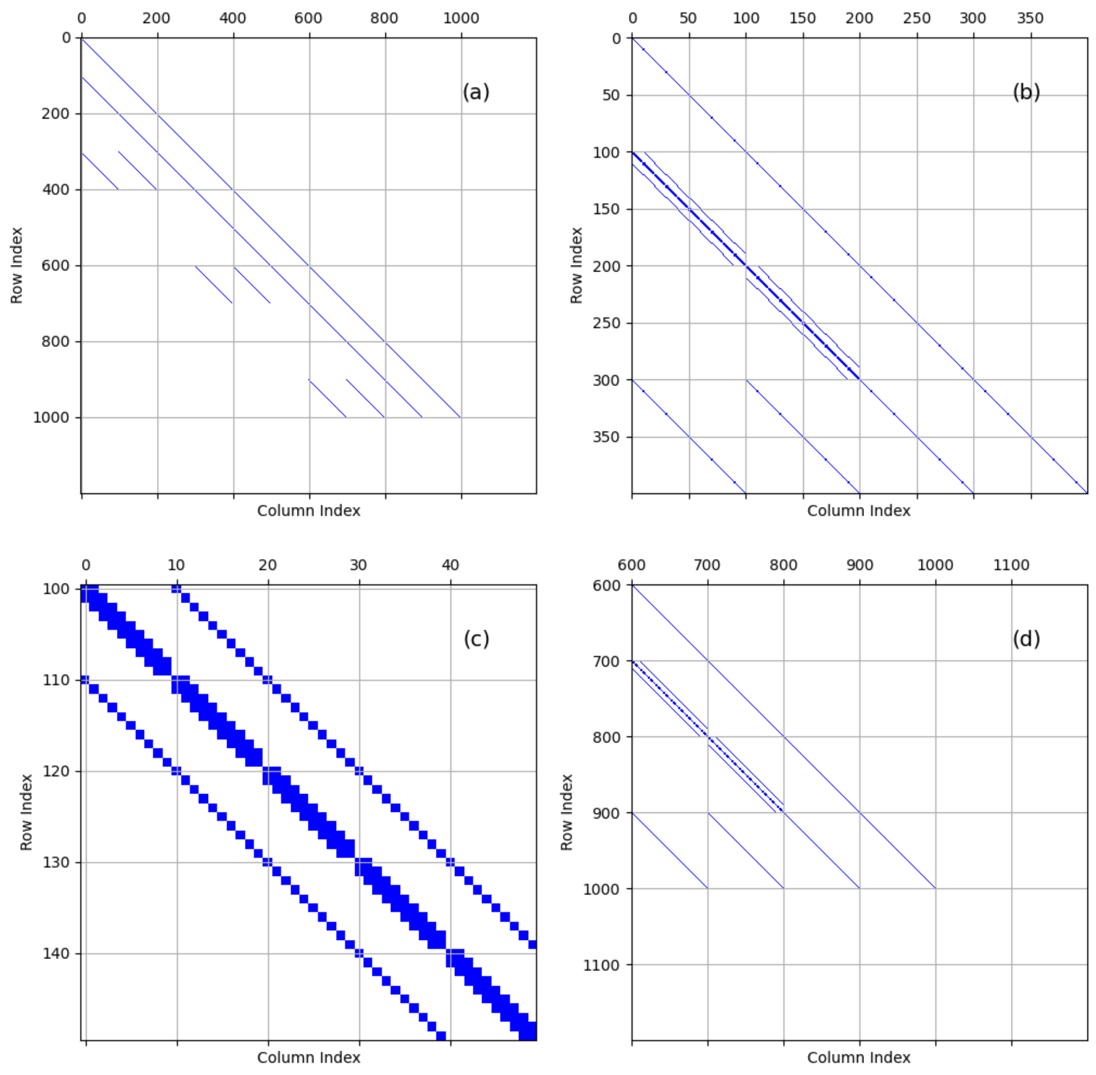


Figure 5.4: Sparsity structure of Λ matrix with parameters $2^p - 2^r = 2^1$ timesteps, $2^m = 2^1$ order of Taylor expansion and $2^r = 0$ repeats of the solution at different levels of resolution (a), (b), (c) and (d)

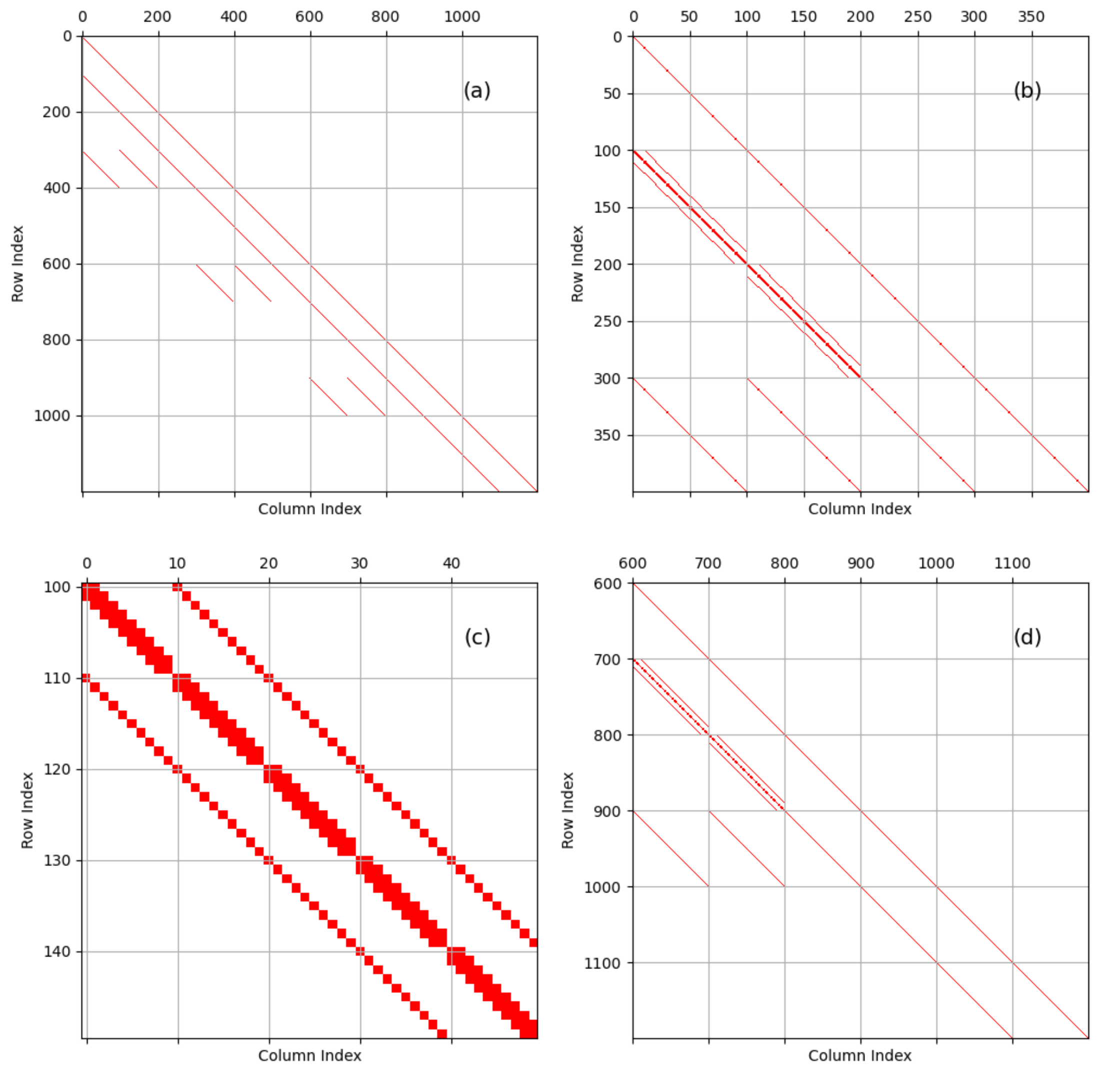


Figure 5.5: Sparsity structure of Λ matrix with parameters $2^p - 2^r = 2^1$ timesteps, $2^m = 2^1$ order of Taylor expansion, and $2^r = 2^1$ repeats of the solution at different levels of resolution (a), (b), (c) and (d)

5.3.2 Discussion of Sparsity and Decomposition

In this section, we attempted to determine both the sparsity and Pauli decomposition of the matrix $\tilde{\Lambda}$ corresponding to both the proper and improper differential operator associated to the matrices $\mathbf{A}$ and $\tilde{\mathbf{A}}$ respectively. Although due to the nature of the matrix $\tilde{\Lambda}_{[2^r;2^p,2^m,2^n]}$, we were only able to derive a pseudo-Pauli decomposition which involves tensors products of sums of Pauli matrices rather than an explicit sum where each term is a simple tensor product of Pauli matrices. With the decomposition we derived, the next step would be to potentially use a computer program to to expand products such as $\sigma_+^{\otimes l}$ and $\left(\widehat{\Pi}_0 + \sigma_+\right)^{\otimes m}$ in terms of the actual Pauli basis and recombine like terms. We think that although this explicit decomposition may be too cumbersome and or complicated to express in a concise format, we believe that it may assist in the decomposition required for Hamiltonian simulation of this matrix in the sense that a brute force evaluation of the terms of (5.21) may not be required. We leave this task for future work. The decomposition still did prove useful for determining the sparsity of $\Lambda_{[2^r;2^p,2^m,2^n]}$ which was the same for the case of $\mathbf{A}$ or $\tilde{\mathbf{A}}$. We found that the sparsity in either case had polynomial dependence on the number of assets d (d^2) and exponential dependence on the parameter m (2^{m-1}). However, as the order at which the analytic solution is truncated in each block of the matrix (5.20) is itself 2^{m-1}, it is more representative to say that the sparsity has a linear dependence on the order of truncation in each of these blocks. We do not see any dependence on the number of gridpoints N in the final complexity however it may be case that there is a hidden dependence on N in terms of d and m if desired error between the approximated solution and actual solution is to be less than some fixed ϵ. Incorporating this possible hidden dependence into the sparsity we leave for future work.

Chapter 6

Numerical Simulations

We will now present the results of our numerical simulations which follow two objectives. The first is to examine the performance of the improper Black-Scholes differential operator in evaluating the price of the option compared to the canonical proper Black-Scholes differential operator. We restrict our examination to single-asset and multi-asset options which permit closed-form expressions for their value, allowing us to benchmark the two numerical methods in terms of their accuracy. For the second half of this chapter, we present the complete proof-of-principle implementation of a quantum algorithm for the case of single-asset option pricing.

In the first part, we present two cases where we confirm that the improper differential operator faithfully approximates the solutions given by the proper differential operator. We first examine the single-asset case and then the multi-asset case and within each of these cases we examine whether the error introduced with improper differential operator can be ameliorated by imposing carefully 'modified' boundary conditions. For the case of the single-asset option, we simulate a vanilla European put option and for the case of the multi-asset option, we simulate a European exchange option. We chose the European exchange option for the multi-asset option as it is one of the few multi-asset options which has a closed form expression from which we can benchmark the numerical solutions. With regards to determining the price of the option, there exists two methods to do so. The first technique is to examine the relevant component from the solution vector directly, say after a given simulation time T_0. The other technique available is to simulate the option up to a time T_0 and then take a conditional expectation with respect to probability distribution of price movements at a time T_1 later, producing the value of the option at a time $T_0 + T_1$. We compare the results from both of these approaches in determining the value of each option. For the latter half of the thesis, we focus on an end-to-end implementation for the pricing of a single-asset. We first provide a detailed overview of the algorithm, which consists of the HHL algorithm and SWAP test. We conclude the description with an expression for the value of the option in terms of probabilities of measuring a subset of qubits of the associated circuit. We simulate two single-asset options with the circuit and provide calculations of option value for each in terms of the count statistics. We conclude by discussing potential sources of errors and any open questions to be addressed in future work.

6.1 1D Black-Scholes

We first consider the case of single-asset option pricing. Consider the Black-Scholes equation:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0 \tag{6.1}$$

with the following boundary and initial conditions for the European put option:

$$\begin{aligned} V(T,S) &= \max(K-S,0) \\ V(t,0) &= Ke^{-r(T-t)} \quad \forall \quad t \\ V(t,S) &\sim 0 \quad \text{as} \quad S \to \infty \end{aligned} \tag{6.2}$$

where the variables T corresponds to maturity, K for the strike price, r for risk-free interest rate, S for spot or current price of the underlying asset and V for the value of the option. We now proceed with how this option can be priced via the finite difference method. Using the spatial and temporal variable changes:

$$S \to x := \ln(S) \qquad\qquad t \to \tau := T - t$$

the differential equation (6.1) along with the boundary and initial conditions (6.2) becomes:

$$\frac{\partial V}{\partial \tau} = \frac{1}{2}\sigma^2 \frac{\partial^2 V}{\partial x^2} + \left(r - \frac{\sigma^2}{2}\right)\frac{\partial V}{\partial x} - rV \tag{6.3}$$

$$
\begin{aligned}
V(\tau = 0, x) &= \max(K - e^x, 0) \\
V(T - \tau, x = -\infty) &= Ke^{-r\tau} \quad \forall \quad \tau \\
V(T - \tau, x = \infty) &\sim 0 \quad as \quad x \to \infty
\end{aligned}
$$

Finally, the second variable change before the finite difference method can be applied is given by:

$$
V \to W := e^{r\tau}V
$$

$$
\frac{\partial W}{\partial \tau} = \frac{1}{2}\sigma^2 \frac{\partial^2 W}{\partial W^2} + \left(r - \frac{\sigma^2}{2}\right)\frac{\partial W}{\partial x} \tag{6.4}
$$

$$
\begin{aligned}
W(\tau = 0, x) &= \max(K - e^x, 0) \\
W(T - \tau, x = -\infty) &= K \quad \forall \quad \tau \\
W(T - \tau, x = \infty) &\sim 0 \quad \text{as} \quad x \to \infty
\end{aligned}
$$

6.1.1 Imposition of Boundary conditions

We now discuss the imposition of two different sets of boundary conditions. We refer to the ‘standard’ boundary conditions as the canonical boundary conditions associated with the usual finite difference method. The second set of boundary conditions we impose, which we denote as ‘modified’ boundary conditions, can be thought of as a correction to the canonical boundary conditions to account for the perturbation we introduced with the use of the improper Black-Scholes differential operator. As previously outlined, we examine the improper Black-Scholes differential operator for the sake of better scaling associated with the numerical characteristics of the matrix compared to the proper Black-Scholes differential operator. We define the proper Black-Scholes differential operator as:

$$
\mathbf{A} = \left(\frac{1}{2}\sum_{i=1}^{d} \sigma_i^2 A_{(\partial^2 x_i)} + \sum_{\substack{i,j=1 \\ j>i}}^{d} \rho_{ij}\sigma_i\sigma_j A_{(\partial x_i, \partial x_j)} + \sum_{i=1}^{d} \left(r - \frac{\sigma_i^2}{2}\right) A_{(\partial x_i)} \right) \tag{6.5}
$$

and the improper differential Black-Scholes operator as:

$$
\tilde{\mathbf{A}} = \left(\frac{1}{2}\sum_{i,j=1}^{d} \rho_{ij}\sigma_i\sigma_j A_{(\partial x_i)} A_{(\partial x_j)} + \sum_{i=1}^{d} \left(r - \frac{\sigma_i^2}{2}\right) A_{(\partial x_i)} \right)
$$

where the submatrices $A_{(\partial x_i)}$, $A_{(\partial x_i, \partial x_j)}$ and $A_{(\partial^2 x_i)}$ are defined as:

$$A_{(\partial x_i)} = I^{\otimes d-i} \otimes D_1 \otimes I^{\otimes i-1} \qquad A_{(\partial^2 x_i)} = I^{\otimes d-i} \otimes D_2 \otimes I^{\otimes i-1}$$

$$A_{(\partial x_i, \partial x_j)} = I^{\otimes d-j} \otimes D_1 \otimes I^{\otimes j-i-1} \otimes D_1 \otimes I^{\otimes i-1}$$

and the explicit forms of the matrices D_1 and D_2 are given by:

$$D_1 = \frac{1}{2h}\begin{pmatrix} 0 & 1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & -1 & 0 \end{pmatrix} \quad D_2 = \frac{1}{h^2}\begin{pmatrix} -2 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{pmatrix} \tag{6.6}$$

In summary, we impose standard Dirichlet boundary conditions for the proper Black-Scholes differential operator and impose modified Dirichlet boundary conditions for the improper Black-Scholes differential operator. We will also examine the case if the standard Dirichlet boundary conditions are imposed for the improper Black-Scholes differential operator to see the effect on the solution.

Standard Dirichlet boundary conditions

As we are dealing with a finite discretized system, we have to place upper and lower limits on the simulation region. The standard boundary conditions for the 1D case will be given by:

$$\vec{V}_1 = \frac{\sigma^2}{2}\begin{pmatrix} L(t) \\ \vdots \\ U(t) \end{pmatrix} + \left(r - \frac{\sigma^2}{2}\right)\begin{pmatrix} -L(t) \\ \vdots \\ U(t) \end{pmatrix} = \begin{pmatrix} (\sigma^2 - r)L(t) \\ \vdots \\ rU(t) \end{pmatrix} \Rightarrow \begin{pmatrix} (\sigma^2 - r)K \\ \vdots \\ 0 \end{pmatrix} \tag{6.7}$$

where we have used the fact that at x_{lower}, $L(t) = K$ and at x_{upper}, $U(t) = 0$

Modified Dirichlet boundary conditions

We now discuss the issues of boundary conditions for the case of approximating the second-order central difference matrix as the composition of the first-order central difference matrix with itself.

$$D_2 \approx (D_1)^2 \tag{6.8}$$

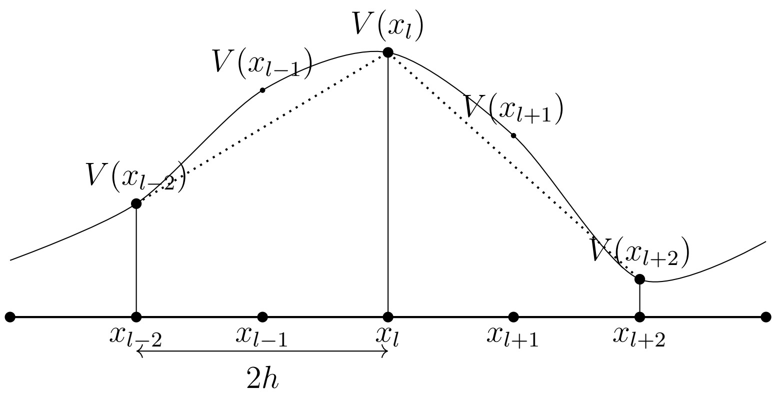
As we stated earlier, this approximation needs to be further examined. We first begin by squaring the D_1 matrix and then seeing how the boundary conditions can be modified to get the appropriate behaviour. A simple calculation yields that the matrix structure for $(D_1)^2$ is given as:

$$D_1^2 = \frac{1}{(2h)^2} \begin{pmatrix} -1 & 0 & 1 & & \\ 0 & -2 & \ddots & \ddots & \\ 1 & \ddots & \ddots & \ddots & 1 \\ & \ddots & \ddots & -2 & 0 \\ & & 1 & 0 & -1 \end{pmatrix} \tag{6.9}$$

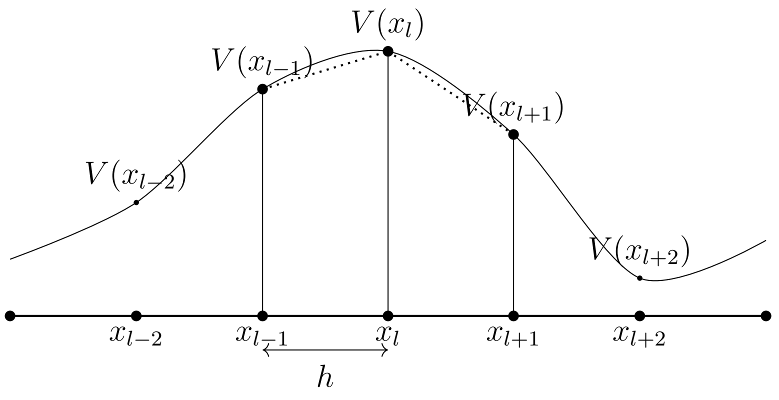
The first thing to notice is that in the bulk region of the simulation area, the matrix (6.9) does in fact compute the second-order central difference albeit with a stepsize $2h$ rather than h:

$$\frac{1}{(2h)^2} \begin{pmatrix} & & & & & & \\ \ddots & \ddots & \ddots & \ddots & \ddots & & \\ \dots & 1 & 0 & -2 & 0 & 1 & \dots \\ & & \ddots & \ddots & \ddots & \ddots & \ddots \\ & & & & & & \end{pmatrix} \begin{pmatrix} \cdot \\ V(x_{l-2}) \\ V(x_{l-1}) \\ V(x_l) \\ V(x_{l+1}) \\ V(x_{l+2}) \\ \cdot \end{pmatrix} = \begin{pmatrix} \cdot \\ \cdot \\ \cdot \\ \frac{V(x_{l-2})-2V(x_l)+V(x_{l+2})}{(2h)^2} \\ \cdot \\ \cdot \\ \cdot \end{pmatrix} \approx \begin{pmatrix} \cdot \\ \cdot \\ \cdot \\ V''(x_l) \\ \cdot \\ \cdot \\ \cdot \end{pmatrix}$$

Pictorially, this can be represented in the following diagram as:



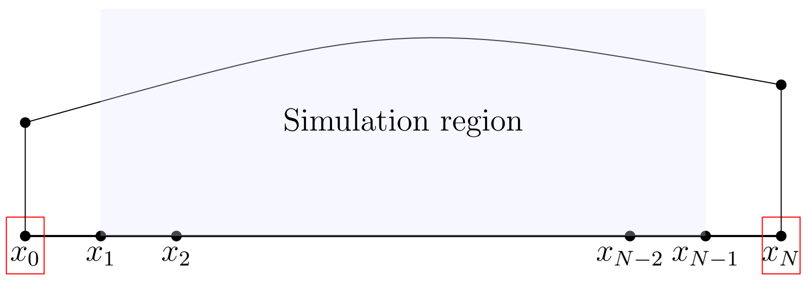
(a) Second-order central difference with $2h$



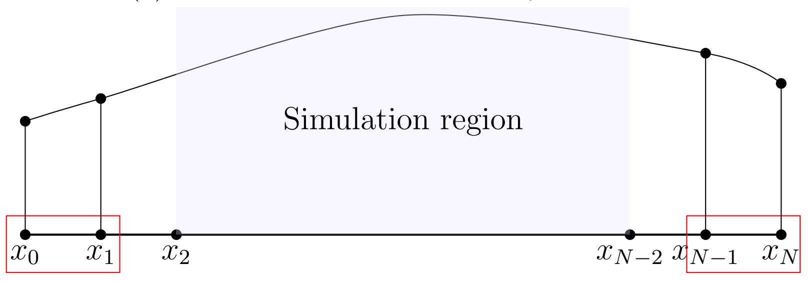
(b) Second-order central difference with h.

Figure 6.1: Difference between stepsize spacing due to approximation of D_2 matrix as D_1 squared.

So therefore the matrix (6.9) is effectively calculating the second-order central difference (within the bulk) with twice the step. However, this implies that if we need to evaluate the second-order central difference at the point x_1 or x_{N-1}, we are missing the necessary points at 'x_{-1}' and 'x_{N+1}' (which don't exist) at the leftmost and rightmost ends of the grid respectively. To counteract this, we effectively have to decrease the number of grid points again by two in each direction.



(a) Standard Dirichlet boundary conditions



(b) Modified Dirichlet boundary conditions

Figure 6.2: Difference between boundary regions for Standard and Modified Dirichlet boundary conditions

In term of matrix arithmetic, we now proceed with the direct computation of the effect of the $(D_1)^2$ matrix on an arbitrary vector so that we can impose the correct boundary conditions:

$$\frac{1}{(2h)^2}\begin{pmatrix} -1 & 0 & 1 & & & & \\ 0 & -2 & \ddots & \ddots & & & \\ 1 & \ddots & \ddots & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & 1 \\ & & & \ddots & \ddots & -2 & 0 \\ & & & & 1 & 0 & -1 \end{pmatrix}\begin{pmatrix} V(x_2) \\ V(x_3) \\ V(x_4) \\ \vdots \\ V(x_{N-4}) \\ V(x_{N-3}) \\ V(x_{N-2}) \end{pmatrix} = \frac{1}{(2h)^2}\begin{pmatrix} -V(x_2)+V(x_4) \\ -2V(x_3)+V(x_5) \\ V(x_2)-2V(x_4)+V(x_6) \\ \vdots \\ V(x_{N-6})-2V(x_{N-4})+V(x_{N-2}) \\ V(x_{N-5})-2V(x_{N-3}) \\ V(x_{N-4})-V(x_{N-2}) \end{pmatrix} \tag{6.10}$$

We need to add in the following vector $\vec{\tilde{B}}$ so that right hand side of (6.10) actually approximates the second-order central difference:

$$\frac{1}{(2h)^2}\begin{pmatrix} -V(x_2)+V(x_4) \\ -2V(x_3)+V(x_5) \\ V(x_2)-2V(x_4)+V(x_6) \\ \vdots \\ V(x_{N-6})-2V(x_{N-4})+V(x_{N-2}) \\ V(x_{N-5})-2V(x_{N-3}) \\ V(x_{N-4})-V(x_{N-2}) \end{pmatrix} + \underbrace{\frac{1}{(2h)^2}\begin{pmatrix} V(x_0)-V(x_2) \\ V(x_1) \\ 0 \\ \vdots \\ 0 \\ V(x_{N-1}) \\ -V(x_{N-2})+V(x_N) \end{pmatrix}}_{\vec{\tilde{B}}} = \begin{pmatrix} V''(x_2) \\ V''(x_3) \\ V''(x_4) \\ \vdots \\ V''(x_{N-4}) \\ V''(x_{N-3}) \\ V''(x_{N-2}) \end{pmatrix}$$

The issue now is that we have the presence of the terms $V(x_2)$ and $V(x_{N-2})$ in $\vec{\tilde{B}}$ in the first and last components, as these are the points that we are actually trying to solve in the first place. One method to approximate these terms is by considering a first-order approximation to the function V near the boundary points using the definition of the central first-order difference:

$$\begin{aligned} V(x_0)-V(x_2) &= -\left(V(x_1+h)-V(x_1-h)\right) \approx -2hV'(x_1) \\ -V(x_{N-2})+V(x_N) &= V(x_{N-1}+h)-V(x_{N-1}-h) \approx 2hV'(x_{N-1}) \end{aligned}$$

There exists two options now with reference to assigning values to $V'(x_1)$ and $V'(x_{n-1})$. The first option is to calculate analytically the derivative of V at the boundary points or alternatively we can take the first-order backwards and forwards finite difference approximation of the derivatives at the points x_1, x_{N-1} respectively.

$$V'(x_1) \approx \frac{V(x_1)-V(x_0)}{h} \quad V'(x_{N-1}) \approx \frac{V(x_N)-V(x_{N-1})}{h}$$

In this case, we will default to the analytic evaluation of the derivative at each point. In this case, we can rewrite (6.10) as:

$$\frac{1}{(2h)^2}\begin{pmatrix} -1 & 0 & 1 & & & & \\ 0 & -2 & \ddots & \ddots & & & \\ 1 & \ddots & \ddots & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & 1 \\ & & & \ddots & \ddots & -2 & 0 \\ & & & & 1 & 0 & -1 \end{pmatrix}\begin{pmatrix} V(x_2) \\ V(x_3) \\ V(x_4) \\ \vdots \\ V(x_{N-4}) \\ V(x_{N-3}) \\ V(x_{N-2}) \end{pmatrix} + \frac{1}{(2h)^2}\begin{pmatrix} -2hV'(x_1) \\ V(x_1) \\ 0 \\ \vdots \\ 0 \\ V(x_{N-1}) \\ 2hV'(x_{N-1}) \end{pmatrix} \approx \begin{pmatrix} V''(x_2) \\ V''(x_3) \\ V''(x_4) \\ \vdots \\ V''(x_{N-4}) \\ V''(x_{N-3}) \\ V''(x_{N-2}) \end{pmatrix} \tag{6.11}$$

For completeness, we also present the D_1 matrix along with its appropriate boundary conditions:

$$\frac{1}{2h}\begin{pmatrix} 0 & 1 & & & \\ -1 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & -1 & 0 \end{pmatrix}\begin{pmatrix} V(x_2) \\ V(x_3) \\ \vdots \\ V(x_{N-3}) \\ V(x_{N-2}) \end{pmatrix} + \frac{1}{2h}\begin{pmatrix} -V(x_1) \\ 0 \\ \vdots \\ 0 \\ V(x_{N-1}) \end{pmatrix} \approx \begin{pmatrix} V'(x_2) \\ V'(x_3) \\ \vdots \\ V'(x_{N-3}) \\ V'(x_{N-2}) \end{pmatrix} \tag{6.12}$$

We notice now that the grid points x_0 or x_N do not appear anywhere in equations (6.11), (6.12). Hence, we can think of the current equations as modified Dirichlet boundary conditions, completely analogous to standard Dirichlet boundary conditions. Therefore we can apply all the same numerical analysis as before in solving the equations. We proceed with some further re-indexing, increasing the number of grid points by two $N \to N' = N + 2$ and shifting all the indices of the grid points down one $(x')_l = (x)_{l+1}$. This transforms the matrices in to the same indexing convention as before. In addition, we re-introduce the implicit time dependence within the function V as well as notation that $L(t) = V(x_0, t)$ and $U(t) = V(x_N, t)$. Finally, we use the following notation for the derivatives at these boundaries:

$$\partial_x L(t) := V'(x_0, t) \qquad \partial_x U(t) := V'(x_N, t)$$

$$\frac{1}{(2h)^2}\begin{pmatrix} -1 & 0 & 1 & & & & \\ 0 & -2 & \ddots & \ddots & & & \\ 1 & \ddots & \ddots & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & 1 \\ & & & \ddots & \ddots & -2 & 0 \\ & & & & 1 & 0 & -1 \end{pmatrix}\begin{pmatrix} V(t,x_1) \\ V(t,x_2) \\ V(t,x_3) \\ \vdots \\ V(t,x_{N-3}) \\ V(t,x_{N-2}) \\ V(t,x_{N-1}) \end{pmatrix} + \frac{1}{(2h)^2}\begin{pmatrix} -2h\partial_x L(t) \\ L(t) \\ 0 \\ \vdots \\ 0 \\ U(t) \\ 2h\partial_x U(t) \end{pmatrix} \approx \begin{pmatrix} \partial_x^2 V(t,x_1) \\ \partial_x^2 V(t,x_2) \\ \partial_x^2 V(t,x_3) \\ \vdots \\ \partial_x^2 V(t,x_{N-3}) \\ \partial_x^2 V(t,x_{N-2}) \\ \partial_x^2 V(t,x_{N-1}) \end{pmatrix} \tag{6.13}$$

$$\frac{1}{2h}\begin{pmatrix} 0 & 1 & & & \\ -1 & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & -1 & 0 \end{pmatrix}\begin{pmatrix} V(t,x_1) \\ V(t,x_2) \\ \vdots \\ V(t,x_{N-2}) \\ V(t,x_{N-1}) \end{pmatrix} + \frac{1}{2h}\begin{pmatrix} -L(t) \\ 0 \\ \vdots \\ 0 \\ U(t) \end{pmatrix} \approx \begin{pmatrix} \partial_x V(t,x_1) \\ \partial_x V(t,x_2) \\ \vdots \\ \partial_x V(t,x_{N-2}) \\ \partial_x V(t,x_{N-1}) \end{pmatrix} \tag{6.14}$$

We will refer interchangeably between the terms such as proper **scheme** and proper **Black-Scholes differential operator** as well as vice versa for the improper case. Scheme will only refer to the type of differential operator used and not which type of boundary conditions are imposed. For determining the value of the option as a condition expectation, we use the following equation:

$$\text{Value of option} = e^{-rT}\int_0^\infty V(T,S)\mathbb{P}(S|S_0)dS \tag{6.15}$$

where the underlying conditional probability distribution $\mathbb{P}(S|S_0)$ is given by:

$$\mathbb{P}(S_t \in (\tilde{S}, \tilde{S}+\delta\tilde{S})) \approx \frac{1}{\tilde{S}\sigma\sqrt{2\pi t}}\exp\left(-\frac{\left(\ln(\tilde{S}) - (r-\frac{\sigma^2}{2})t - \ln(S_0)\right)^2}{2\sigma^2 t}\right)\delta\tilde{S}$$

Transforming to the integration variable $x = \ln(S)$ transforms (6.15) to:

$$\text{Value of option} = e^{-rT}\int_{-\infty}^\infty V(T,e^x)\mathbb{P}(e^x|S_0)dx \tag{6.16}$$

As the j^{th} component of the solution will approximately be value of the option for a given asset price of $S = \exp(x_j)$ (namely the quantity $V\left(T,\exp(x_j)\right)$) and assuming the integrand is only appreciably different from zero within the simulation region we can approximate (6.16) as:

$$\begin{aligned}
\text{Value of option} &\approx e^{-rT}\int_{x_{\text{lower}}}^{x_{\text{upper}}} V(T,e^x)\mathbb{P}(e^x|S_0)dx \\
&\approx e^{-rT}\sum_{j=1}^{N-1} \underbrace{V(T,\exp(x_j))}_{j^{th}\text{component of solution}} \tilde{\mathbb{P}}(x_j|x_0)\Delta x \qquad (6.17)
\end{aligned}$$

where the new conditional probability distribution $\tilde{\mathbb{P}}(x|x_0 := \ln(S_0))$ is given by:

$$\tilde{\mathbb{P}}(x|x_0) = \frac{1}{\sigma\sqrt{2\pi T_1}}\exp\left(-\frac{\left(x-(r-\frac{\sigma^2}{2})T_1 - x_0\right)^2}{2\sigma^2 T_1}\right) \qquad (6.18)$$

We estimate the price of the option with the conditional expectation using (6.17).

6.1.2 Simulation parameters

We summarize the free parameters in the Black-Scholes equation and discretization used in the simulations. We now assign some values for each of the variables that characterize the option. A volatility of $\sigma = 0.1$ and and risk-interest rate $r = 0.05$ are typical values for these parameters looking at S&P 500 and U.S. Treasury securities. The spot price S_0 and strike price K were chosen randomly along with the two time periods T_0 and T_1. We chose a symmetric interval for the simulation region with an upper possible value of $e^7 \approx 1100$ which is sufficiently 'far away' from the initial value S_0 and strike price K to avoid unwanted effects associated with being too close to the boundary conditions. We chose a grid number of $N_{\text{gridnumber}} = 300$ as we found that it was optimal in terms of having a sufficiently dense grid without running into floating point errors associated with evaluating derivatives over small distances. We also found that having too many timesteps $N_{\text{timesteps}}$ lead to floating point errors in the final solution.

Parameter	Value
S_0	80
K	100
σ	0.1
r	0.05
T_0	1
T_1	1
T	2

Table 6.1: Parameters relating to the option.

Parameter	Value
x_{lower}	-7
x_{upper}	7
$N_{\text{gridpoints}}$	300
Δx	0.047
Δt	0.2
$N_{\text{timesteps}}$	5
Order of Taylor approximation of analytic sol. for each step.	5

Table 6.2: Parameters relating to the discretization

6.1.3 Results

We now plot the numerical solutions arising from calculating the 'time-compounded' value of the option $W = e^{r\tau}V$ versus the natural logarithm of the underlying asset price $x = \ln(S)$ for the case of both the proper and improper Black-Scholes (B-S) differential operators. In figure (6.3), we provide a graphical comparison between the two simulations for the case of standard boundary conditions imposed for both the proper and improper differential operator. We remark a more pronounced numerical instability at the lower boundary condition for the simulation of the improper B-S differential operator compared to the proper B-S differential operator. In figure (6.4), we also provide a graphical comparison between the solutions, however with modified boundary conditions rather than standard boundary conditions imposed for the improper B-S differential operator. We remark that there seems to be no discernible difference between the two plots of the improper differential B-S operator for the different sets of boundary conditions.

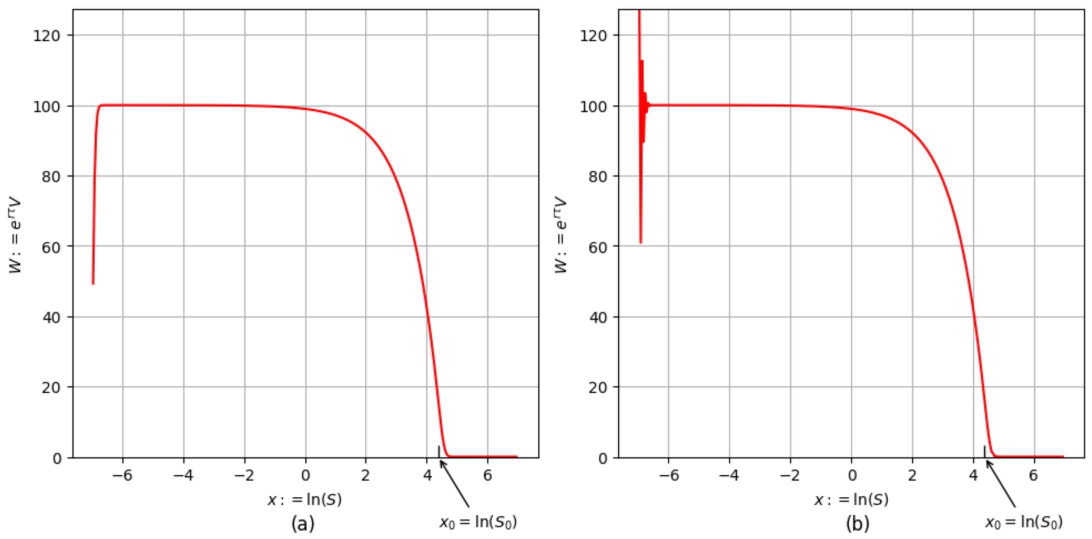


Figure 6.3: Plot of 'time-compounded' value of the option $W = e^{r\tau}V$ where $\tau = T_0$ versus the natural logarithm of the underlying asset price $x = \ln(S)$ for the (a) proper and the (b) improper schemes with **standard** Dirichlet boundary conditions imposed for **both** simulations for a single asset. The natural logarithm of the spot price $x_0 \approx 4.4$ is also indicated in the diagram.

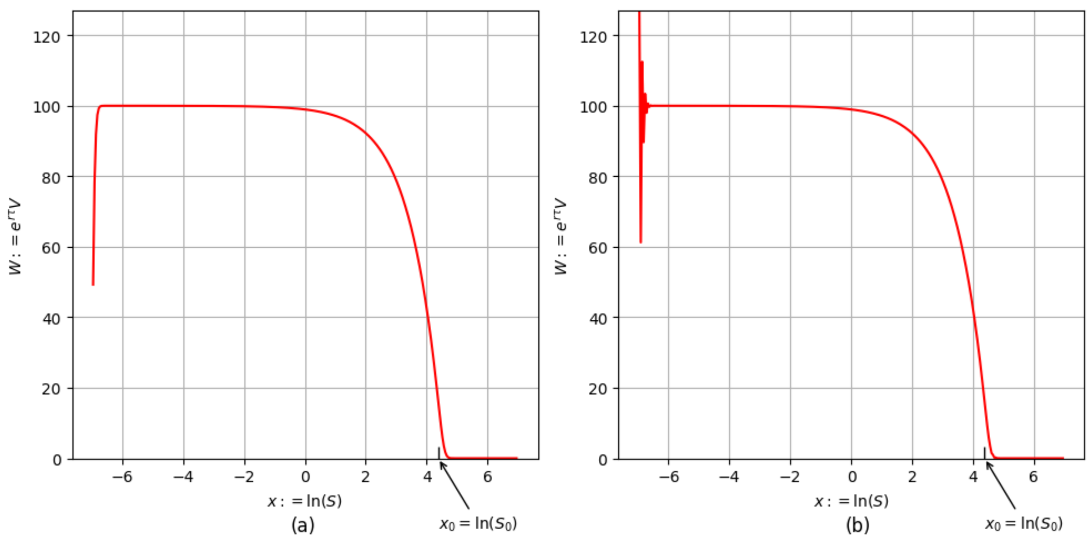


Figure 6.4: Plot of 'time-compounded' value of the option $W = e^{r\tau}V$ where $\tau = T_0$ versus the natural logarithm of the underlying asset price $x = \ln(S)$ for the (a) proper scheme with standard Dirichlet boundary conditions and the (b) improper scheme with modified Dirichlet boundary conditions imposed. The natural logarithm of the spot price $x_0 \approx 4.4$ is also indicated in the diagram. As previously stated, there is no discernible difference between imposing standard or modified Dirichlet boundary conditions for the improper scheme. We include the plot of the proper scheme with standard Dirichlet boundary conditions again for purposes of direct comparison.

We also plot the probability density function associated with potential movements in the natural logarithm of the of the underlying asset price $x = \ln(S)$ at a time T_1 later. As outlined in Chapter 2, this corresponds to a normal probability distribution. We plotted the probability distribution for $N_{grid} = 300$ (which is the grid used in this simulation) and for the case of $N_{grid} = 1000$ (which we take to be the continuum for this and other plots). As to be expected, the graphs seem indiscernible given the large number of grid points, however in some cases for coarse grids, there will be noticeable difference.

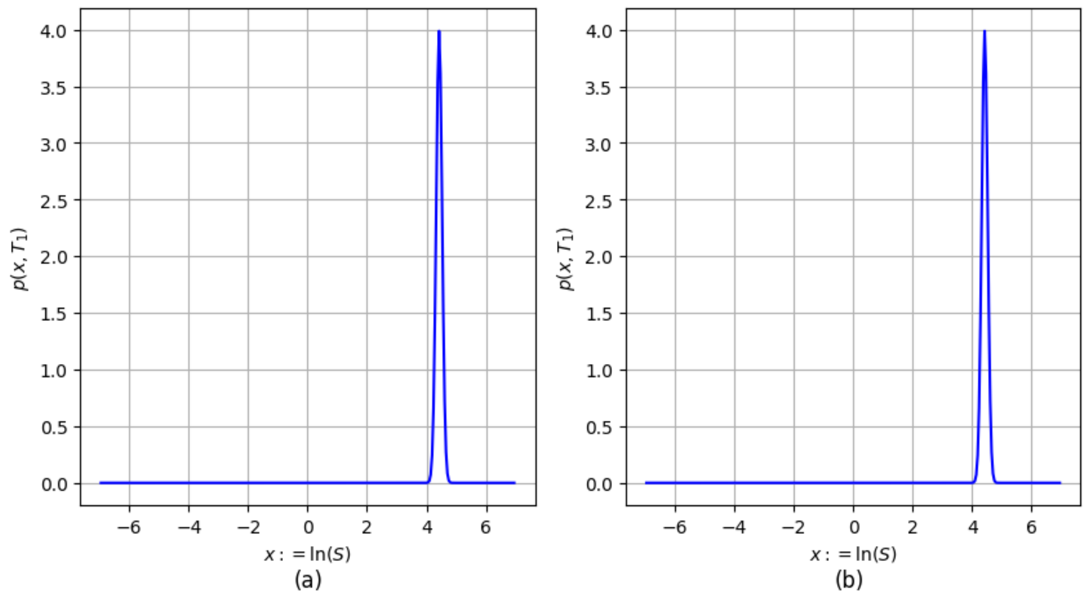


Figure 6.5: Plot of probability distribution of natural logarithm of the price of the underlying asset as defined on (a) the grid vs (b) continuum. As we are dealing with $N_{\text{grid}} = 300$, the graphs are essentially identical.

We now outline the results from solving the Black-Scholes equation up to the time T_0 and extracting the value of the option by examining the relevant component of the solution vector that corresponds (or closest) to the spot price S_0. As this component actually corresponds to the quantity $e^{rT_0} V_{\text{option price}}$, we multiply the value of this component by the 'time-discount' factor of e^{-rT_0} to get the price of the option $V_{\text{option price}}$. We detail the values of the option as calculated by the proper scheme with standard boundary conditions along with the improper scheme for the two types of boundary conditions:

Description	Value
Proper scheme with standard Dirichlet boundary conditions	15.618
Improper scheme with standard Dirichlet boundary conditions	15.661
Improper scheme with modified Dirichlet boundary conditions	15.661
Analytic value at T_0	15.271

Table 6.3: Value of the option at a time T_0 as calculated by examining the relevant component from the solution vector for the proper scheme with standard boundary conditions and the improper scheme with standard and modified Dirichlet boundary conditions imposed. We also include the analytical value for the option from the Black-Scholes formula

Description	Value
Conditional expectation of proper scheme with *std* Dirichlet B.C.s	11.770
Conditional expectation of improper scheme with *std* Dirichlet B.C.s	11.773
Conditional expectation of improper scheme with *mod* Dirichlet B.C.s	11.773
Analytic value at $T_0 + T_1$	11.757

Table 6.4: Value of the option at a time $T_0 + T_1$ later as calculated from the conditional expectation using the proper scheme with standard boundary conditions and the improper scheme with standard and modified Dirichlet boundary conditions imposed. We also include the analytical value for the option as calculated from the Black-Scholes formula.

For the case of a single asset, numerical simulations agree nearly perfectly with the analytic value, both with direct extraction at a time T_0 and extraction via conditional expectation at a time $T_0 + T_1$. We also note that the performance of both the improper and proper schemes for evaluating the option price via both direct extraction and extraction via conditional expectation agree to at least three decimal places. For evaluating the option via direct extraction at a time T_0, the relative error between the analytic value and the proper scheme is approximately 2.272% whereas the difference between analytic value and improper scheme (for both standard and modified boundary conditions) is approximately 2.554%. We suspect that the agreement between values for the option under (a) improper scheme with standard boundary conditions and (b) improper scheme with modified boundary conditions is as a result of the larger number of grid points used in the simulation. For the case of extraction via conditional expectation, we also see very close agreement between the analytic value and proper scheme (relative error of 0.111%) and the improper scheme (relative error of 0.136%).

6.2 2D Black-Scholes

Consider the Black-Scholes equation in two dimensions:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S_1^2 \frac{\partial^2 V}{\partial S_1^2} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S_2^2} + \rho\sigma_1\sigma_2 \frac{\partial V}{\partial S_1 \partial S_2} + rS_1 \frac{\partial V}{\partial S_1} + rS_2 \frac{\partial V}{\partial S_2} - rV = 0$$

with the associated boundary and initial conditions for the European exchange option:

$$V(T, S_1, S_2) = \max(S_1(T) - S_2(T), 0) \tag{6.19}$$

Near side boundary conditions:

$$\begin{aligned} V(t, 0, S_2) &= 0 \quad \forall t \\ V(t, S_1, 0) &= \max(S_1, 0) \quad \forall t \end{aligned}$$

Far side boundary conditions:

$$\begin{aligned} V(t, (S_1)_{upper}, S_2) &= \max((S_1)_{upper} - S_2, 0) \quad \forall t \\ V(t, S_1, (S_2)_{upper}) &= \max(S_1 - (S_2)_{upper}, 0) \quad \forall t \end{aligned}$$

where the variables T corresponds to maturity, r for interest rate, σ_1, σ_2 for volatilities of the two assets, S_1, S_2 for the prices of the two underlying assets, $(S_1)_{\text{upper}}, (S_2)_{\text{upper}}$ for the far side boundary values for assets S_1, S_2, and V for the value of the option. The analytic value for the European Exchange option with terminal payoff condition (6.19) is given by Margrabe's formula:

$$V_{\text{Exchange}} = S_1(0)\Phi(d_1) - S_2(0)\Phi(d_2) \tag{6.20}$$

where d_1, d_2 are defined as:

$$d_1 = \frac{\ln\left(\frac{S_1(0)}{S_2(0)}\right) + \frac{\tilde{\sigma}^2 T}{2}}{\tilde{\sigma}\sqrt{T}} \qquad d_2 = d_1 - \sigma\sqrt{T}$$

where $S_1(0)$, $S_2(0)$ refer to the current (spot) price of the first and second underlying asset. We will use the above closed form expression (6.20) to benchmark the values of the option as calculated from the simulations. For the European exchange option, the risk-free interest rate r is set to zero as the option comprises of two options that are both subject to the same compounding interest. However, we present the following derivations of the Black-Scholes differential equation and associated initial and boundary conditions for the more general case of $r \neq 0$. Taking the following variables changes:

$$S_1 \to x_1 := \ln(S_1) \quad S_2 \to x_2 := \ln(S_2) \quad t \to \tau := T - t$$

the Black-Scholes differential equation transforms as :

$$\frac{\partial V}{\partial \tau} = \frac{1}{2}\sigma_1^2 \frac{\partial^2 V}{\partial x_1^2} + \frac{1}{2}\sigma_2^2 \frac{\partial^2 V}{\partial x_2^2} + \rho\sigma_1\sigma_2 \frac{\partial^2 V}{\partial x_1 \partial x_2} + \left(r - \frac{\sigma_1^2}{2}\right)\frac{\partial V}{\partial x_1} + \left(r - \frac{\sigma_2^2}{2}\right)\frac{\partial V}{\partial x_2} - rV$$

The boundary and initial conditions for the Black-Scholes equation can be transformed using the new variables as follows:

Initial conditions

$$V(\tau = 0, x_1, x_2) = \max(e^{x_1} - e^{x_2}, 0)$$

Boundary conditions - Near side

$$V(T - \tau, (x_1)_{\text{lower}}, x_2) = 0 \quad \forall \tau$$
$$V(T - \tau, x_1, (x_2)_{\text{lower}}) = \max(e^{x_1}, 0) = e^{x_1} \quad \forall \tau$$

Boundary conditions - Far side

$$V(T - \tau, (x_1)_{\text{upper}}, x_2) = \max(e^{(x_1)_{\text{upper}}} - e^{x_2}, 0) \quad \forall \tau$$
$$V(T - \tau, x_1, (x_2)_{\text{upper}}) = \max(e^{x_1} - e^{(x_2)_{\text{upper}}}, 0) \quad \forall \tau$$

For the last substitution $W = e^{r\tau}V$, we get the final form of the Black-Scholes equation along with the corresponding boundary and initial conditions:

$$V \to W := e^{r\tau}V$$

$$\frac{\partial W}{\partial \tau} = \frac{1}{2}\sigma_1^2 \frac{\partial^2 W}{\partial x_1^2} + \frac{1}{2}\sigma_2^2 \frac{\partial^2 W}{\partial x_2^2} + \rho\sigma_1\sigma_2 \frac{\partial^2 V}{\partial x_1 \partial x_2} + \left(r - \frac{\sigma_1^2}{2}\right)\frac{\partial W}{\partial x_1} + \left(r - \frac{\sigma_2^2}{2}\right)\frac{\partial W}{\partial x_2}$$

Initial conditions

$$W(\tau = 0, x_1, x_2) = \max(e^{x_1} - e^{x_2}, 0)$$

Boundary conditions - Near side

$$W(\tau, (x_1)_{\text{lower}}, x_2) = 0 \quad \forall \tau$$
$$W(\tau, x_1, (x_2)_{\text{lower}}) = e^{r\tau + x_1} \quad \forall \tau$$

Boundary conditions - Far side

$$W(\tau, (x_1)_{\text{upper}}, x_2) = e^{r\tau} \max(e^{(x_1)_{\text{upper}}} - e^{x_2}, 0) \quad \forall \tau$$
$$W(\tau, x_1, (x_2)_{\text{upper}}) = e^{r\tau} \max(e^{x_1} - e^{(x_2)_{\text{upper}}}, 0) \quad \forall \tau$$

6.2.1 Imposition of boundary conditions

Standard Dirichlet boundary conditions

Consider the transformed Black-Scholes equation again:

$$\frac{\partial W}{\partial \tau} = \frac{1}{2}\sigma_1^2 \frac{\partial^2 W}{\partial x_1^2} + \frac{1}{2}\sigma_2^2 \frac{\partial^2 W}{\partial x_2^2} + \rho\sigma_1\sigma_2 \frac{\partial^2 V}{\partial x_1 \partial x_2} + \left(r - \frac{\sigma_1^2}{2}\right)\frac{\partial W}{\partial x_1} + \left(r - \frac{\sigma_2^2}{2}\right)\frac{\partial W}{\partial x_2}$$

The boundary conditions are given by:

$$\vec{B}(\tau) = \frac{\sigma_1^2}{2}\vec{B}_{\partial^2 x_1}(\tau) + \frac{\sigma_2^2}{2}\vec{B}_{\partial^2 x_2}(\tau) + \rho\sigma_1\sigma_2 \vec{B}_{(\partial x_1 \partial x_2)}(\tau) + \left(r - \frac{\sigma_1^2}{2}\right)\vec{B}_{(\partial x_1)}(\tau) + \left(r - \frac{\sigma_2^2}{2}\right)\vec{B}_{(\partial x_2)}(\tau)$$

where each subcomponent is defined as:

$$\left(\vec{B}_{(\partial x_1)}(\tau)\right)_l = \frac{1}{2h}\sum_{j_2}\left[-\delta_{l,\mathcal{N}(j_2,1)} L_{x_1}(\tau, j_2) + \delta_{l,\mathcal{N}(j_2,N-1)} U_{x_1}(\tau, j_2)\right]$$

$$\left(\vec{B}_{(\partial x_2)}(\tau)\right)_l = \frac{1}{2h}\sum_{j_1}\left[-\delta_{l,\mathcal{N}(1,j_1)}L_{x_2}(\tau,j_1) + \delta_{l,\mathcal{N}(N-1,j_1)}U_{x_2}(\tau,j_1)\right]$$

$$\left(\vec{B}_{(\partial^2 x_1)}(\tau)\right)_l = \frac{1}{h^2}\sum_{j_2}\left[\delta_{l,\mathcal{N}(j_2,1)}L_{x_1}(\tau,j_2) + \delta_{l,\mathcal{N}(j_2,N-1)}U_{x_1}(\tau,j_2)\right]$$

$$\left(\vec{B}_{(\partial^2 x_2)}(\tau)\right)_l = \frac{1}{h^2}\sum_{j_1}\left[\delta_{l,\mathcal{N}(1,j_1)}L_{x_2}(\tau,j_1) + \delta_{l,\mathcal{N}(N-1,j_1)}U_{x_2}(\tau,j_1)\right]$$

$$\left(\vec{B}_{(\partial x_1,\partial x_2)}(\tau)\right)_l = \frac{1}{4h^2}\sum_{j_2,j_1}\Big[-\delta_{l,\mathcal{N}(j_2,1)}L_{x_1}(\tau,j_2) - \delta_{l,\mathcal{N}(1,j_1)}L_{x_2}(\tau,j_1) + \delta_{l,\mathcal{N}(j_2,N-1)}U_{x_1}(\tau,j_2) + \delta_{l,\mathcal{N}(N-1,j_1)}U_{x_2}(\tau,j_1)\Big]$$

For the current case, the explicit boundary conditions are given by:

$$\begin{aligned}
L_{x_1}(\tau,j_2) &= 0 && \forall\tau, j_2\\
U_{x_1}(\tau,j_2) &= e^{r\tau}\max(e^{(x_1)_{\text{upper}}} - e^{x_2(j_2)}, 0) && \forall\tau, j_2\\
L_{x_2}(\tau,j_1) &= e^{r\tau + x_1(j_1)} && \forall\tau, j_1\\
U_{x_2}(\tau,j_1) &= e^{r\tau}\max(e^{x_1(j_1)} - e^{(x_2)_{\text{upper}}}, 0) && \forall\tau, j_1
\end{aligned}$$

where we define the following functions that goes from the indices j_1, j_2 to the corresponding values of x_1, x_2:

$$x_1(j_1) = \left(\frac{(x_1)_{\text{upper}} - (x_1)_{\text{upper}}}{N+1}\right)(j_1+1) + (x_1)_{\text{lower}}$$

$$x_2(j_2) = \left(\frac{(x_2)_{\text{upper}} - (x_2)_{\text{upper}}}{N+1}\right)(j_2+1) + (x_2)_{\text{lower}}$$

Modified Dirichlet boundary conditions

Consider the Black-Scholes equation again in two dimensions:

$$\frac{\partial W}{\partial\tau} = \frac{1}{2}\sigma_1^2\frac{\partial^2 W}{\partial x_1^2} + \frac{1}{2}\sigma_2^2\frac{\partial^2 W}{\partial x_2^2} + \rho\sigma_1\sigma_2\frac{\partial^2 V}{\partial x_1\partial x_2} + \left(r - \frac{\sigma_1^2}{2}\right)\frac{\partial W}{\partial x_1} + \left(r - \frac{\sigma_2^2}{2}\right)\frac{\partial W}{\partial x_2}$$

However, this time we shall impose the modified Dirichlet boundary conditions along each axis x_1 and x_2 for the case of the improper Black-Scholes differential operator corresponding to the second partial spatial derivatives along each of these axes. As we saw in the one dimensional case this implies the following new boundary conditions along each axis:

$$\frac{1}{(2h)^2}\begin{pmatrix} -1 & 0 & 1 & & & & \\ 0 & -2 & \ddots & \ddots & & & \\ 1 & \ddots & \ddots & \ddots & \ddots & & \\ & \ddots & \ddots & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots & \ddots & 1 \\ & & & \ddots & \ddots & -2 & 0 \\ & & & & 1 & 0 & -1 \end{pmatrix}\begin{pmatrix} V(t,x_1) \\ V(t,x_2) \\ V(t,x_3) \\ \vdots \\ V(t,x_{N-3}) \\ V(t,x_{N-2}) \\ V(t,x_{N-1}) \end{pmatrix} + \frac{1}{(2h)^2}\begin{pmatrix} -2h\partial_x L(t) \\ L(t) \\ 0 \\ \vdots \\ 0 \\ U(t) \\ 2h\partial_x U(t) \end{pmatrix} \approx \begin{pmatrix} \partial_x^2 V(t,x_1) \\ \partial_x^2 V(t,x_2) \\ \partial_x^2 V(t,x_3) \\ \vdots \\ \partial_x^2 V(t,x_{N-3}) \\ \partial_x^2 V(t,x_{N-2}) \\ \partial_x^2 V(t,x_{N-1}) \end{pmatrix} \tag{6.21}$$

Where the new 'subcomponents' for $\partial_{x_1}^2, \partial_{x_2}^2$ operators are defined respectively as:

$$\left(\vec{\tilde{B}}_{(\partial^2 x_1)}(\tau)\right)_l = \frac{1}{(2h)^2}\sum_{j_2}\Big[-\delta_{l,\mathcal{N}(j_2,1)}2h\partial_{x_1}L_{x_1}(\tau,j_2) + \delta_{l,\mathcal{N}(j_2,2)}L_{x_1}(\tau,j_2)$$
$$+ \delta_{l,\mathcal{N}(j_2,N-2)}U_{x_1}(\tau,j_2) + \delta_{l,\mathcal{N}(j_2,N-1)}2h\partial_{x_1}U_{x_1}(\tau,j_2)\Big]$$

$$\left(\vec{\tilde{B}}_{(\partial^2 x_2)}(\tau)\right)_l = \frac{1}{(2h)^2}\sum_{j_1}\Big[-\delta_{l,\mathcal{N}(1,j_1)}2h\partial_{x_2}L_{x_2}(\tau,j_1) + \delta_{l,\mathcal{N}(2,j_1)}L_{x_2}(\tau,j_1)$$
$$+ \delta_{l,\mathcal{N}(N-2,j_1)}U_{x_2}(\tau,j_1) + \delta_{l,\mathcal{N}(N-1,j_1)}2h\partial_{x_2}U_{x_2}(\tau,j_1)\Big]$$

The boundary conditions are now given by:

$$\vec{B}(\tau) = \frac{\sigma_1^2}{2}\vec{\tilde{B}}_{\partial^2 x_1}(\tau) + \frac{\sigma_2^2}{2}\vec{\tilde{B}}_{\partial^2 x_2}(\tau) + \rho\sigma_1\sigma_2\vec{B}_{(\partial x_1 \partial x_2)}(\tau)$$
$$+ \left(r - \frac{\sigma_1^2}{2}\right)\vec{B}_{(\partial x_1)}(\tau) + \left(r - \frac{\sigma_2^2}{2}\right)\vec{B}_{(\partial x_2)}(\tau)$$

where each of the remaining subcomponents is the same as the standard Dirichlet boundary conditions:

$$\left(\vec{B}_{(\partial x_1)}(\tau)\right)_l = \frac{1}{2h}\sum_{j_2}\left[-\delta_{l,\mathcal{N}(j_2,1)}L_{x_1}(\tau,j_2) + \delta_{l,\mathcal{N}(j_2,N-1)}U_{x_1}(\tau,j_2)\right]$$

$$\left(\vec{B}_{(\partial x_2)}(\tau)\right)_l = \frac{1}{2h}\sum_{j_1}\left[-\delta_{l,\mathcal{N}(1,j_1)}L_{x_2}(\tau,j_1) + \delta_{l,\mathcal{N}(N-1,j_1)}U_{x_2}(\tau,j_1)\right]$$

$$\left(\vec{B}_{(\partial x_1,\partial x_2)}(\tau)\right)_l = \frac{1}{4h^2}\sum_{j_2,j_1}\Big[-\delta_{l,\mathcal{N}(j_2,1)}L_{x_1}(\tau,j_2) - \delta_{l,\mathcal{N}(1,j_1)}L_{x_2}(\tau,j_2)$$
$$+ \delta_{l,\mathcal{N}(j_2,N-1)}U_{x_1}(\tau,j_2) + \delta_{l,\mathcal{N}(N-1,j_1)}U_{x_2}(\tau,j_1)\Big]$$

For the case at hand, the explicit boundary conditions are given by:

$$L_{x_1}(\tau,j_2) = 0 \qquad \forall\tau, j_2$$
$$U_{x_1}(\tau,j_2) = e^{r\tau}\max(e^{(x_1)_{\text{upper}}} - e^{x_2(j_2)}, 0) \qquad \forall\tau, j_2 \tag{6.22}$$
$$L_{x_2}(\tau,j_1) = e^{r\tau + x_1(j_1)} \qquad \forall\tau, j_1$$
$$U_{x_2}(\tau,j_1) = e^{r\tau}\max(e^{x_1(j_1)} - e^{(x_2)_{\text{upper}}}, 0) \qquad \forall\tau, j_1 \tag{6.23}$$

In addition, we need to evaluate the partial derivatives at each of these boundary conditions. Using the fact:

$$\frac{\partial}{\partial x_i}\max(f(x_1,...,x_n),0) = \frac{\partial_{x_i}f(x_1,...,x_n)}{f(x_1,...,x_n)}\max(f(x_1,...,x_n),0) \qquad \text{if} \quad f(x_1,..,x_n) \neq 0 \tag{6.24}$$

As $(x_1)_{\text{lower}} < x_1(j_1) < (x_1)_{\text{upper}} \quad \forall j_1$ and vice versa for $x_2(j_2)$, the difference of two exponential functions within the max(.) function for (6.22) and (6.23) will never be zero. Therefore we can apply (6.24) to yield the following results:

$$\partial x_1 L_{x_1}(\tau,j_2) = 0 \qquad \forall\tau, j_2$$
$$\partial x_1 U_{x_1}(\tau,j_2) = \left(\frac{e^{(x_1)_{\text{upper}}+r\tau}}{e^{(x_1)_{\text{upper}}} - e^{x_2(j_2)}}\right)\max(e^{(x_1)_{\text{upper}}} - e^{x_2(j_2)}, 0) \qquad \forall\tau, j_2$$
$$\partial x_2 L_{x_2}(\tau,j_1) = 0 \qquad \forall\tau, j_1$$
$$\partial x_2 U_{x_2}(\tau,j_1) = \left(\frac{e^{(x_2)_{\text{upper}}+r\tau}}{e^{(x_2)_{\text{upper}}} - e^{x_1(j_1)}}\right)\max(e^{x_1(j_1)} - e^{(x_2)_{\text{upper}}}, 0) \qquad \forall\tau, j_1$$

where we use the following 'indexing' functions that maps from the indices j_1, j_2 to the corresponding values of x_1, x_2:

$$x_1(j_1) = \left(\frac{(x_1)_{\text{upper}} - (x_1)_{\text{lower}}}{N+1}\right)(j_1+1) + (x_1)_{\text{lower}}$$

$$x_2(j_2) = \left(\frac{(x_2)_{\text{upper}} - (x_2)_{\text{lower}}}{N+1}\right)(j_2+1) + (x_2)_{\text{lower}}$$

For determination of the price of the option with the conditional expectation, we use the following integral expression (in a similar manner to 6.17):

$$\text{Value of option} = e^{-rT} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} V\left(T, (e^{x_2}, e^{x_1})\right) \tilde{\mathbb{P}}\left[(x_2, x_1) | ((x_2)_{\text{now}}, (x_1)_{\text{now}})\right] dx_1 dx_2 \quad (6.25)$$

where the parameters $(x_1)_{\text{now}} := \ln(S_1)$, $(x_2)_{\text{now}} := \ln(S_2)$ are the natural logarithms of the initial prices of the two assets S_1, S_2 respectively. The integral expression (6.25) can be approximated in the same manner as before to yield:

$$e^{-rT} \sum_{k,l=1}^{N-1} V\left(T, \left(e^{(x_2)_k}, e^{(x_1)_l}\right)\right) \tilde{\mathbb{P}}\left[((x_2)_k, (x_1)_l) \,|\, ((x_2)_{\text{now}}, (x_1)_{\text{now}})\right] \Delta x_1 \Delta x_2 \quad (6.26)$$

where the term $V\left(T, \left(e^{(x_2)_k}, e^{(x_1)_l}\right)\right)$ corresponds to the $[(N-1)(k-1)+l]^{th}$ component of the solution and $\tilde{\mathbb{P}}\left((x_2, x_1) | ((x_2)_{\text{now}}, (x_1)_{\text{now}})\right)$ corresponds to the bivariate equivalent of the univariate distribution in (6.18). We will use (6.26) to calculate the value of the option from the conditional expectation.

6.2.2 Simulation parameters

We now choose some parameters relating to the option and discretization.

Parameter	Value
S_1	170
S_2	90
r	0
ρ	0.1
σ_1	0.3
σ_2	0.4
T_0	1
T_1	1
T	2

Table 6.5: Parameters relating to option

Parameter	Value
$(x_1)_{\text{lower}}$	-8
$(x_1)_{\text{upper}}$	8
$(x_2)_{\text{lower}}$	-8
$(x_2)_{\text{upper}}$	8
N_{x_1}	30
N_{x_2}	30

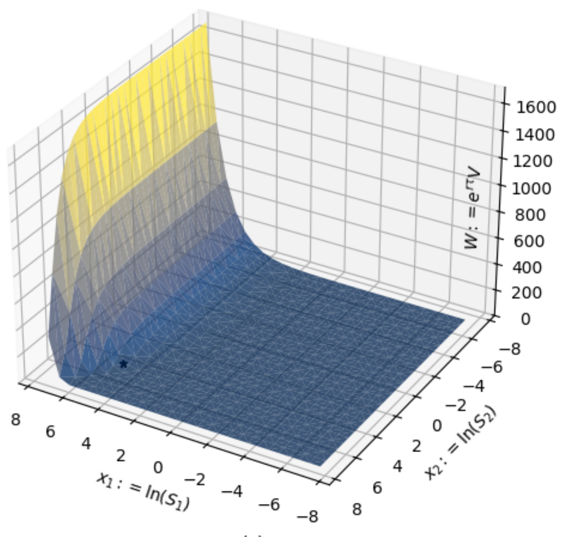
Parameter	Value
$N_{x_1 \times x_2}$	900
Δx_1	0.533
Δx_2	0.533
$N_{\text{timesteps}}$	2
Δt	0.5
Order of Taylor approximation of analytic sol. for each step.	3

Table 6.6: Parameters relating to discretization

As this is a European Exchange option, the risk-free interest rate r is set to zero. As multi-asset options typically involve riskier underlying assets, we chose higher volatilities of $\sigma = 0.3,\ 0.4$. The two time periods T_1, T_2 were chosen to be the same as previously. The initial underlying asset prices S_0, S_1 were again chosen at random. As this option comprised of riskier underlying assets with one of the assets S_1 nearly twice as large as the previous single-asset option, we chose a lager symmetric simulation region of $[-8, 8]$ compared to $[-7, 7]$ for the previous option. For selection of the number grid points along each direction N_{x_1}, N_{x_2}, we were limited by the computational resources required to invert the final matrix. The above issue also constrained us with selection of the number of time steps $N_{timesteps}$ and the order of the Taylor approximation of the analytical solution for each time step, as larger values for these parameters increased the size of the matrix to be inverted. Therefore, we varied those parameters until we saw close agreement between the analytic value and the proper scheme with standard boundary conditions. Once we had these parameters selected, we then benchmarked the improper scheme with the two sets of boundary conditions against the analytic solution and the proper scheme with standard boundary conditions.

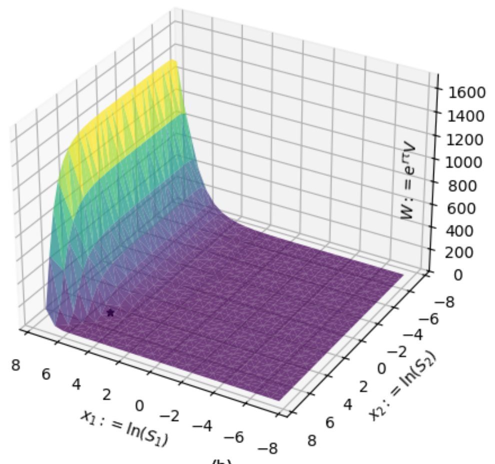
6.2.3 Results

We now plot the numerical solutions arising from calculating the 'time-compounded' value of the option $W = e^{r\tau}V$ versus the natural logarithm of the two underlying asset price $x_1 = \ln(S_1)$, $x_2 = \ln(S_2)$ for the case of both the proper and improper schemes. As the risk-free interest rate r is zero for the European Exchange option, the 'time-compounded' value W and value of the option V will coincide. In figure (6.6), we plot the analytic value of the option V versus the natural logarithm of the two underlying assets $x_1 = \ln(S_1)$, $x_2 = \ln(S_2)$ as calculated from Margrabe's pricing formula. In figure (6.7), we provide a graphical comparison between the two simulations of the option V versus the parameters x_1, x_2 for the case of the proper and improper scheme with standard Dirichlet boundary conditions imposed for both simulations. We note a divergence between the two solutions at the upper boundary condition for the variable x_1 for the simulation of the improper B-S differential operator compared to the proper B-S differential operator. However, as the initial point $((x_2)_{\text{now}}, (x_1)_{\text{now}}) \approx (5.1, 4.5)$, we would not anticipate much of a effect on the calculated price of the solution if the aforementioned divergence is simply a localised boundary effect. In figure (6.8), we also provide a graphical comparison between the absolute error between the two solutions and the analytic value for the price of the option. We remark that as we approach the upper boundary condition associated with the variable x_1, we see quite pronounced deviations from the numerical value of the option for as calculated by proper and improper scheme (with standard boundary conditions) and the analytic value for the price of the option. Although the deviations occur around the same value of $x_1 \approx 6 \sim 7$, the error seems more pronounced surprisingly in the case of the proper scheme with standard Dirichlet boundary conditions.

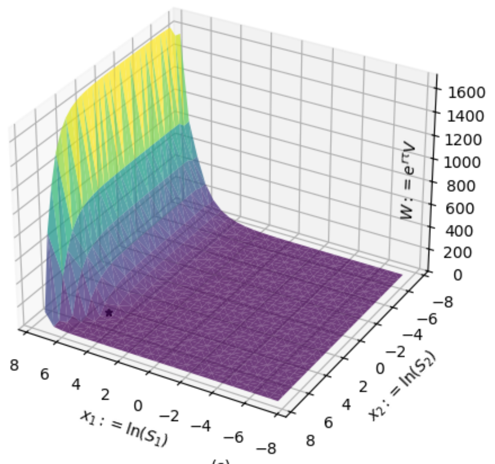


(a)

Figure 6.6: (a) Plot of analytic value of the European exchange option option V versus the natural logarithms of the underlying asset prices $x_2 = \ln(S_2)$, $x_1 = \ln(S_1)$ using Margrabe's formula up to time T_0.



(b)



(c)

Figure 6.7: Plot of 'time-compounded' value of the option $W = e^{r\tau}V$ where $\tau = T_0$ versus the natural logarithms of the underlying asset prices $x_2 = \ln(S_2)$, $x_1 = \ln(S_1)$ for the (b) proper and the (c) improper schemes with standard Dirichlet boundary conditions imposed for both simulations for two assets. The natural logarithms of the two spot prices of the underlying asset prices $((x_2)_{\text{now}}, (x_1)_{\text{now}}) \approx (5.1, 4.5)$ is also indicated in the diagram with a star. We remark that as $r = 0$ for a European Exchange option, the time-compounded value of the option and the value of the option coincide (i.e. $W = V$).

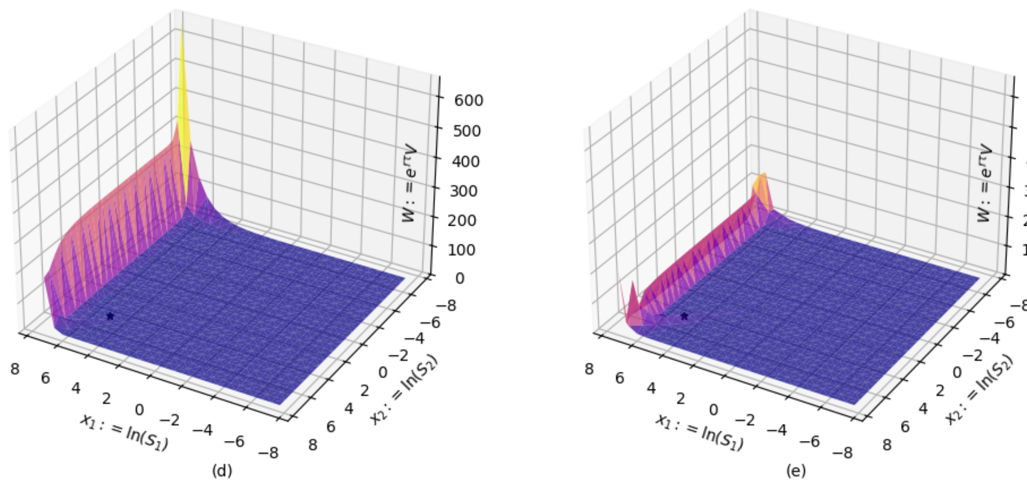
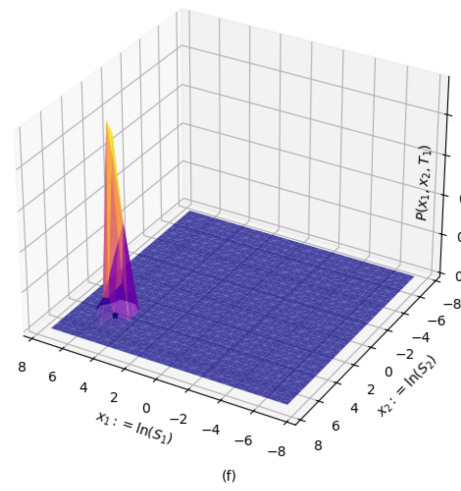


Figure 6.8: Plot of absolute error between the analytic value of the option V versus the natural logarithms of the underlying asset prices $x_2 = \ln(S_2)$, $x_1 = \ln(S_1)$ for the (d) proper and the (e) improper schemes with standard Dirichlet boundary conditions imposed for both simulations for two assets. The natural logarithms of the two spot prices of the underlying asset prices $((x_2)_{\text{now}}, (x_1)_{\text{now}}) \approx (5.1, 4.5)$ is also indicated in the diagram with a star.We remark that as $r = 0$ for a European Exchange option, the time-compounded value of the option and the value of the option coincide (i.e $W = V$).

Description	Value
Proper scheme with standard Dirichlet boundary conditions	83.103
Improper scheme with standard Dirichlet boundary conditions	89.375
Improper scheme with modified Dirichlet boundary conditions	89.375
Analytic value	82.421

Table 6.7: Value of the option at a time T_0 as calculated by examining the relevant component from the solution vector for the proper scheme with standard boundary conditions and the improper scheme with standard and modified Dirichlet boundary conditions imposed. We also include the analytical value for the option as calculated from Margrabe's formula.



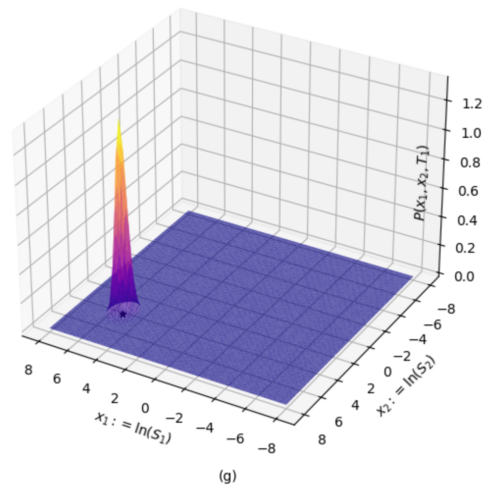


Figure 6.9: Plot of the bivariate normal distribution $\tilde{\mathbb{P}}\left((x_1, x_2) \,|\, ((x_1)_{\text{now}}, (x_2)_{\text{now}})\right)$ for the natural logarithm of the price of the underlying assets $x_1 = \ln(S_1)$, $x_2 = \ln(S_2)$ at a time T_1 later defined (a) on the grid $N_{x_1 \times x_2}$ and (b) on the continuum (i.e. $N_{1000\times 1000}$).

Description	Value
Conditional expectation of proper scheme with *std* Dirichlet B.C.s	86.863
Conditional expectation of improper scheme with *std* Dirichlet B.C.s	88.319
Conditional expectation of improper scheme with *mod* Dirichlet B.C.s	88.319
Analytic value at $T_0 + T_1$	87.406

Table 6.8: Value of the option at a time $T_0 + T_1$ later as calculated from the conditional expectation using the proper scheme with standard boundary conditions and the improper scheme with standard and modified Dirichlet boundary conditions imposed. We also include the analytical value for the option as calculated from Margrabe's formula.

For the case of multi-asset pricing, we begin to notice a deviation between the price of the option at a time T_0 as calculated between the proper and improper scheme when determining the value of the option via examining the relevant component from the solution vector. For the case of the proper scheme with standard boundary conditions, we calculate an option value of 83.103 leading to a relative error of 0.827% whereas we calculate an option value of 89.375 for the improper scheme under both sets of boundary conditions, leading to a corresponding relative error of 8.437%. We believe that this tenfold discrepancy in relative error between the two schemes is due to insufficient number of gridpoints (as previously highlighted for the case of the improper scheme, the second central difference is approximated with effectively twice the stepsize compared to the proper scheme). As we will see in due course, the improper scheme seems to converge to the proper scheme with sufficiently dense set of gridpoints. However, considering the calculation of the price via the conditional expectation, the relative errors between two approaches compared to the analytic value at time $T_0 + T_1$ seem roughly the same with a relative error between the proper scheme and analytic value at time $T_0 + T_1$ of 0.621% compared to a relative error of 0.936% with the improper scheme. We note also agreement to three decimal places between the value of the option as calculated by the improper scheme (via examining the relevant component of the solution vector and calculation via the conditional probability distribution) irrespective of whether we impose standard or modified boundary conditions. The agreement between these two approaches may relate to the fact that the correction terms for the modified boundary conditions is proportional to the stepsize $h \approx \mathcal{O}(1/N_{\text{gridnumber}})$. Therefore, for sufficiently large gridnumber, these additional terms may be negligible and we are effectively implementing the same boundary conditions. We leave further analysis of this artefact for future work.

6.2.4 Numerical stability of solutions

We now proceed with examining and comparing the stability of numerical simulations of the proper scheme with standard boundary conditions versus the improper scheme with modified Dirichlet boundary conditions. We first examine the consequence of increasing the number of grid points along each axis from 15 to 45 in increments of 15. We stopped at 45 grid points to due excessive computational resources required to solve the matrix associated to, for example, a 60×60 grid. As we have already plotted for the case of 30 grid points along each axis, we shall omit that case below and examine the case of 15 and 45 grid points along each axis. Apart from number of grid points, every parameter related to the simulation has the exact same value as previously stated in table (6.5).

Grid number and stability

We note that calculating the value of the option from examining the relevant component of the solution vector leads to considerably worse accuracy in the value of the option then taking the conditional expectation of the numerical solution with the bivariate normal distribution $\tilde{\mathbb{P}}\left(((x_2, x_1)) \,|\, ((x_2)_{\text{now}}, (x_1)_{\text{now}})\right)$. However, we do notice that the accuracy improves slightly as we increase the grid number but there is still substantial relative error in both the proper and improper schemes. One reason for this is an insufficient grid number $N_{\text{gridnumber}}$. As we are uniformly dividing the grid in the logarithmic value of the underlying assets $x_i = \ln(S_i)$, that implies the spacing in the actual value of the underlying assets will increase exponentially as we move towards the far-side boundary of the grid:

$$(x_1)_{i+1} - (x_1)_i = \Delta x$$

$$e^{(x_1)_{i+1}} - e^{(x_1)_i} = \underbrace{e^{(x_1)_i}}_{\text{Exponentially increasing}} \underbrace{\left(e^{\Delta x} - 1\right)}_{\substack{\text{Constant arising} \\ \text{from discretization}}}$$

As we can only examine the value of the option at the grid point that is closest to the logarithmic value of the underlying asset prices, this may lead to large error. However, by taking the conditional expectation value with the bivariate distribution , we are, in some sense, 'interpolating' between the grid points, hence leading to a considerably higher accuracy. Interestingly, we do not see as a pronounced numerical instability in the improper scheme with modified boundary conditions.

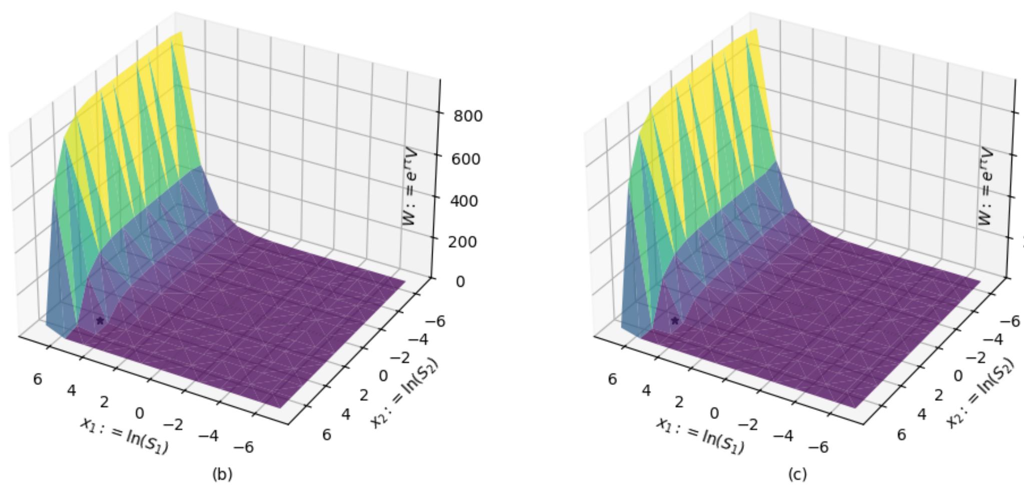


Figure 6.10: Plot of 'time-compounded' value of the option $W = e^{r\tau}V$ where $\tau = T_0$ versus the natural logarithms of the underlying asset prices $x_2 = \ln(S_2)$, $x_1 = \ln(S_1)$ for the (a) proper scheme with standard Dirichlet boundary conditions and (b) the improper scheme with modified Dirichlet boundary conditions for two assets with 15 gridpoints along each axis $N_{x_1} = N_{x_2} = 15$. The natural logarithms of the two spot prices of the underlying asset prices $((x_2)_{\text{now}}, (x_1)_{\text{now}}) \approx (5.1, 4.5)$ is also indicated in the diagram with a star.

Description	Value
Proper scheme with standard Dirichlet boundary conditions	7.987
Improper scheme with modified Dirichlet boundary conditions	6.271
Analytic value at T_0	82.421
Conditional expectation of proper scheme with *std* Dirichlet B.C.s	46.562
Conditional expectation of improper scheme with *mod* Dirichlet B.C.s	76.848
Analytic value at $T_0 + T_1$	87.406

Table 6.9: The first set of three values corresponds to the values of the option at a time T_0 as calculated by examining the relevant component from the solution vector for the proper scheme with standard boundary conditions, the improper scheme with modified Dirichlet boundary conditions and the analytic value for the option from Margrabe's formula. The second set of values refer to the values of the option at a time $T_0 + T_1$ later as calculated from the conditional expectation using the proper scheme with standard boundary conditions, the improper scheme with modified Dirichlet boundary conditions and the analytic value for the option from Margrabe's formula. Both sets of values refers to a simulation with a set of 15 gridpoints along each axis ($N_{x_1} = N_{x_2} = 15$).

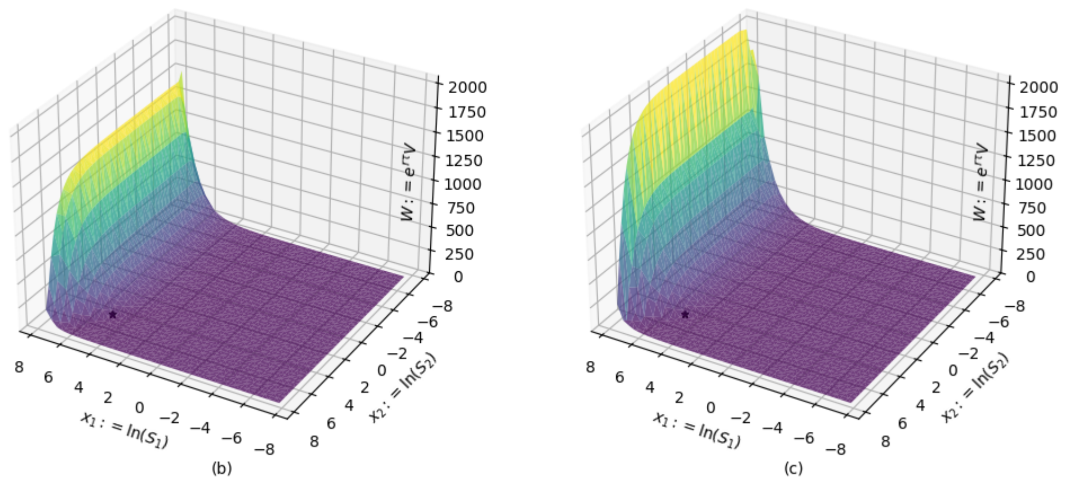


Figure 6.11: Plot of 'time-compounded' value of the option $W = e^{r\tau}V$ where $\tau = T_0$ versus the natural logarithms of the underlying asset prices $x_2 = \ln(S_2)$, $x_1 = \ln(S_1)$ for the (a) proper scheme with standard Dirichlet boundary conditions and (b) the improper scheme with modified Dirichlet boundary conditions for two assets with 45 gridpoints along each axis $N_{x_1} = N_{x_2} = 45$. The natural logarithms of the two spot prices of the underlying asset prices $((x_2)_{\text{now}}, (x_1)_{\text{now}}) \approx (5.1, 4.5)$ is also indicated in the diagram with a star.

Description	Value
Proper scheme with standard Dirichlet boundary conditions	85.142
Improper scheme with modified Dirichlet boundary conditions	85.417
Analytic value at T_0	82.421
Conditional expectation of proper scheme with *std* Dirichlet B.C.s	86.805
Conditional expectation of improper scheme with *mod* Dirichlet B.C.s	87.370
Analytic value at $T_0 + T_1$	88.406

Table 6.10: The first set of three values corresponds to the values of the option at a time T_0 as calculated by examining the relevant component from the solution vector for the proper scheme with standard boundary conditions, the improper scheme with modified Dirichlet boundary conditions and the analytic value for the option from Margrabe's formula. The second set of values refer to the values of the option at a time $T_0 + T_1$ later as calculated from the conditional expectation using the proper scheme with standard boundary conditions, the improper scheme with modified Dirichlet boundary conditions and the analytic value for the option from Margrabe's formula. Both sets of values refers to a simulation with a set of 15 gridpoints along each axis ($N_{x_1} = N_{x_2} = 45$).

Covariance matrix and stability

We now examine how the numerical stability is affected by varying statistical parameters associated with the covariance matrix describing the distribution of the underlying assets comprising the option. We will return to having 30 gridpoints along each axis ($N_{x_1} = N_{x_2} = 30$). Apart from the covariance matrix, every parameter related to the simulation has the exact same value as previously stated in table (6.5). We first begin by changing the variance between the correlation ρ between the two assets from being moderately correlated with $\rho = 0.5$ to strongly correlated with $\rho = 0.9$. We then set the correlation back to its previous value of $\rho = 0.1$ and then vary the volatility σ_1 associated with the first underlying asset S_1. We chose to vary the volatility of the first asset rather than the second asset as we remarked previously, that there seems is a more pronounced deviation between the simulations for the two schemes at the upper boundary condition associated with $(x_1)_{\text{upper}}$. We want to examine what, if any effect, does increased volatility have on the deviation between the two simulations and associated calculated prices for the option.

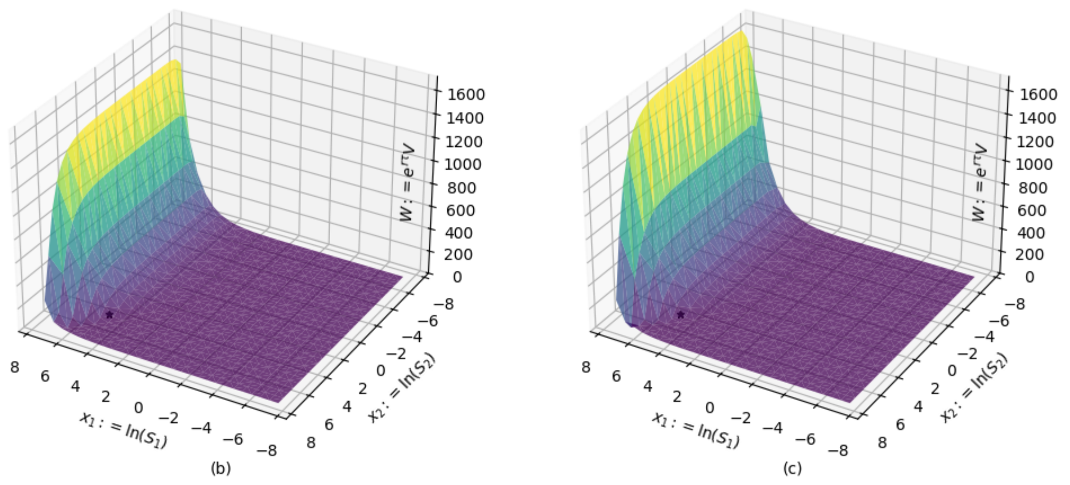


Figure 6.12: Plot of 'time-compounded' value of the option $W = e^{r\tau}V$ where $\tau = T_0$ versus the natural logarithms of the underlying asset prices $x_2 = \ln(S_2)$, $x_1 = \ln(S_1)$ for the (a) proper scheme with standard Dirichlet boundary conditions and (b) the improper scheme with modified Dirichlet boundary conditions for two assets with a correlation of $\rho = 0.5$. The natural logarithms of the two spot prices of the underlying asset prices $((x_2)_{\text{now}}, (x_1)_{\text{now}}) \approx (5.1, 4.5)$ is also indicated in the diagram with a star.

Description	Value
Proper scheme with standard Dirichlet boundary conditions	80.040
Improper scheme with modified Dirichlet boundary conditions	86.675
Analytic value at T_0	80.688
Conditional expectation of proper scheme with *std* Dirichlet B.C.s	82.318
Conditional expectation of improper scheme with *mod* Dirichlet B.C.s	84.942
Analytic value at $T_0 + T_1$	83.138

Table 6.11: The first set of three values corresponds to the values of the option at a time T_0 as calculated by examining the relevant component from the solution vector for the proper scheme with standard boundary conditions, the improper scheme with modified Dirichlet boundary conditions and the analytic value for the option from Margrabe's formula with $\rho = 0.5$. The second set of values refer to the values of the option at a time $T_0 + T_1$ later as calculated from the conditional expectation using the proper scheme with standard boundary conditions, the improper scheme with modified Dirichlet boundary conditions and the analytic value for the option from Margrabe's formula, also with $\rho = 0.5$.

The first thing to notice is that the correlation between the underlying assets seems to have little to no effect on the stability of the numerical solutions. We see some numerical instability for the case of the improper scheme with modified boundary conditions at the intersection (corner) of both far-side boundary conditions. When determining the price of the option via examination of the relevant component of the solution vector, we see a large relative error of 7.42% for the improper scheme versus 0.803% for the proper scheme. However, in a similar way to the previous simulations, we record more comparable relative errors between the two schemes when determining the price of the option via the conditional probability distribution. We note a relative error of 2.17% for the improper scheme and 0.986% for the proper scheme when using the conditional probability distribution to determine the price of the option.

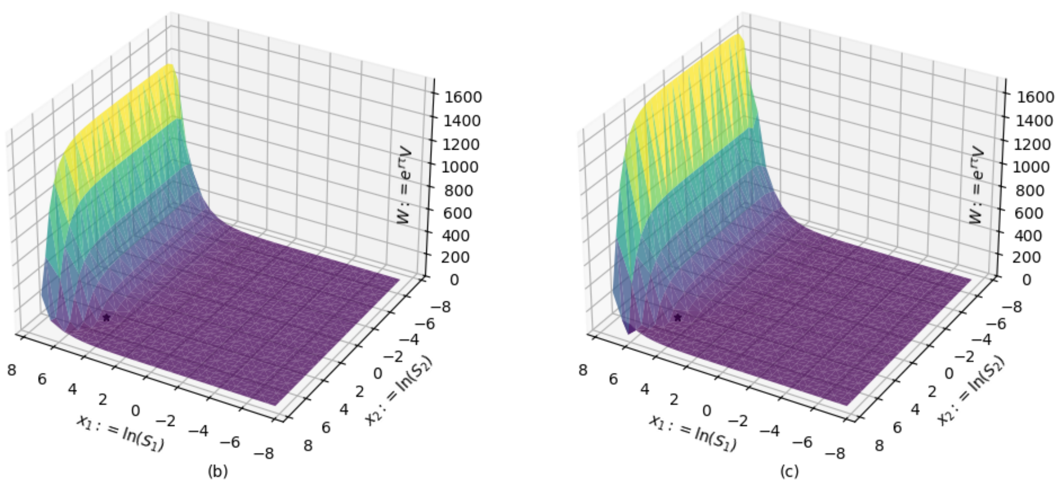


Figure 6.13: Plot of 'time-compounded' value of the option $W = e^{r\tau}V$ where $\tau = T_0$ versus the natural logarithms of the underlying asset prices $x_2 = \ln(S_2)$, $x_1 = \ln(S_1)$ for the (a) proper scheme with standard Dirichlet boundary conditions and (b) the improper scheme with modified Dirichlet boundary conditions for two assets with a correlation of $\rho = 0.9$. The natural logarithms of the two spot prices of the underlying asset prices $((x_2)_{\text{now}}, (x_1)_{\text{now}}) \approx (5.1, 4.5)$ is also indicated in the diagram with a star.

Description	Value
Proper scheme with standard Dirichlet boundary conditions	76.815
Improper scheme with modified Dirichlet boundary conditions	83.853
Analytic value at T_0	80.002
Conditional expectation of proper scheme with *std* Dirichlet B.C.s	74.638
Conditional expectation of improper scheme with *mod* Dirichlet B.C.s	80.800
Analytic value at $T_0 + T_1$	80.077

Table 6.12: The first set of three values corresponds to the values of the option at a time T_0 as calculated by examining the relevant component from the solution vector for the proper scheme with standard boundary conditions, the improper scheme with modified Dirichlet boundary conditions and the analytic value for the option from Margrabe's formula with $\rho = 0.9$. The second set of values refer to the values of the option at a time $T_0 + T_1$ later as calculated from the conditional expectation using the proper scheme with standard boundary conditions, the improper scheme with modified Dirichlet boundary conditions and the analytic value for the option from Margrabe's formula, also with $\rho = 0.9$.

As we increase the strength of the correlation between the two assets to $\rho = 0.9$, we see the aforementioned instability at the corner of the two far-side boundaries become more pronounced. When determining the price of the option via examination of the relevant component of the solution vector, we see a relative error of 4.812% for the improper scheme and 3.984% for the proper scheme. In determining the value of the option via the conditional probability distribution, we record a relative error of 0.903% for the improper scheme and 6.792% for the proper scheme.

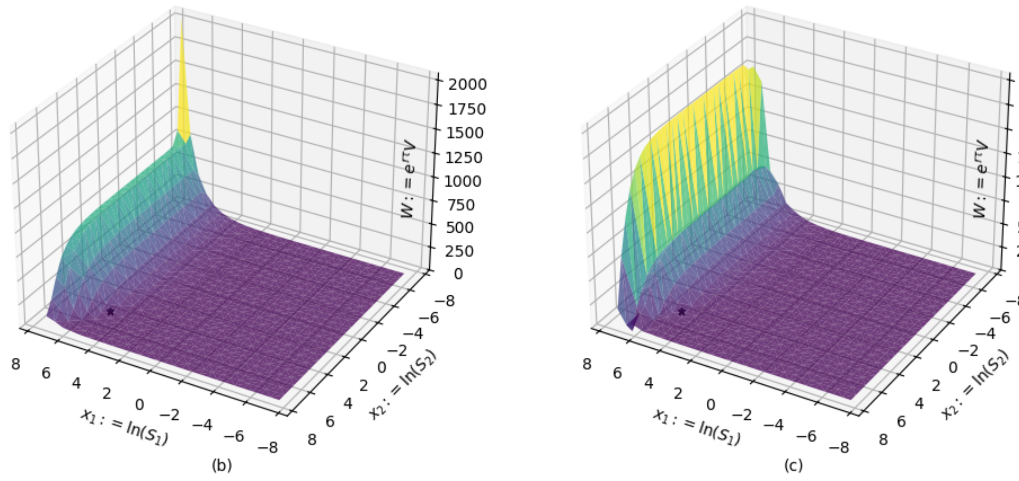
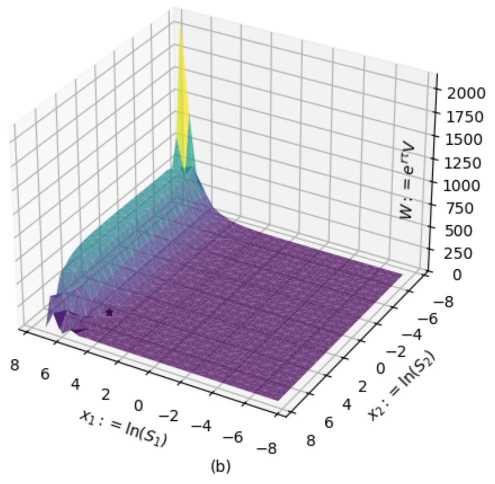


Figure 6.14: Plot of 'time-compounded' value of the option $W = e^{r\tau}V$ where $\tau = T_0$ versus the natural logarithms of the underlying asset prices $x_2 = \ln(S_2)$, $x_1 = \ln(S_1)$ for the (a) proper scheme with standard Dirichlet boundary conditions and (b) the improper scheme with modified Dirichlet boundary conditions for two assets with a volatility of the first asset $\sigma_1 = 0.7$. The natural logarithms of the two spot prices of the underlying asset prices $((x_2)_{\text{now}}, (x_1)_{\text{now}}) \approx (5.1, 4.5)$ is also indicated in the diagram with a star.

Description	Value
Proper scheme with standard Dirichlet boundary conditions	81.885
Improper scheme with modified Dirichlet boundary conditions	104.387
Analytic value at T_0	90.487
Conditional expectation of proper scheme with *std* Dirichlet B.C.s	96.100
Conditional expectation of improper scheme with *mod* Dirichlet B.C.s	95.636
Analytic value at $T_0 + T_1$	101.592

Table 6.13: The first set of three values corresponds to the values of the option at a time T_0 as calculated by examining the relevant component from the solution vector for the proper scheme with standard boundary conditions, the improper scheme with modified Dirichlet boundary conditions and the analytic value for the option from Margrabe's formula with $\sigma_1 = 0.7$. The second set of values refer to the values of the option at a time $T_0 + T_1$ later as calculated from the conditional expectation using the proper scheme with standard boundary conditions, the improper scheme with modified Dirichlet boundary conditions and the analytic value for the option from Margrabe's formula, also with $\sigma_1 = 0.7$.



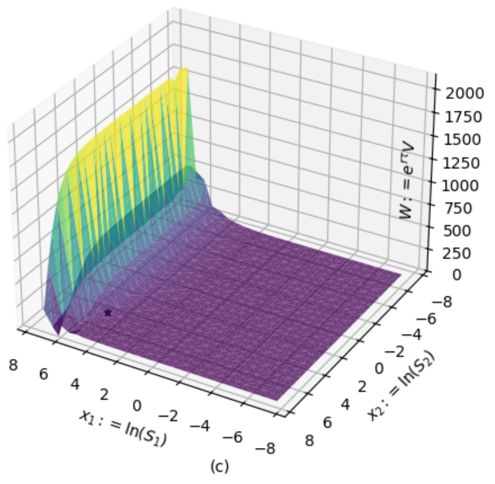


Figure 6.15: Plot of ‘time-compounded’ value of the option $W = e^{r\tau}V$ where $\tau = T_0$ versus the natural logarithms of the underlying asset prices $x_2 = \ln(S_2)$, $x_1 = \ln(S_1)$ for the (a) proper scheme with standard Dirichlet boundary conditions and (b) the improper scheme with modified Dirichlet boundary conditions for two assets with a volatility of the first asset $\sigma_1 = 0.9$. The natural logarithms of the two spot prices of the underlying asset prices $((x_2)_{\text{now}}, (x_1)_{\text{now}}) \approx (5.1, 4.5)$ is also indicated in the diagram with a star.

Description	Value
Proper scheme with standard Dirichlet boundary conditions	119.286
Improper scheme with modified Dirichlet boundary conditions	127.823
Analytic value at T_0	96.521
Conditional expectation of proper scheme with *std* Dirichlet B.C.s	92.827
Conditional expectation of improper scheme with *mod* Dirichlet B.C.s	94.225
Analytic value at $T_0 + T_1$	110.525

Table 6.14: The first set of three values corresponds to the values of the option at a time T_0 as calculated by examining the relevant component from the solution vector for the proper scheme with standard boundary conditions, the improper scheme with modified Dirichlet boundary conditions and the analytic value for the option from Margrabe’s formula with $\sigma_1 = 0.9$. The second set of values refer to the values of the option at a time $T_0 + T_1$ later as calculated from the conditional expectation using the proper scheme with standard boundary conditions, the improper scheme with modified Dirichlet boundary conditions and the analytic value for the option from Margrabe’s formula, also with $\sigma_1 = 0.9$.

We first notice that increasing the volatility of the first underlying asset σ_1 of the first asset drastically reduces the numerical stability of the solutions. However, taking the conditional expectation and comparing to the analytic solution $T_0 + T_1$ ameliorates this issue greatly. For the case of $\sigma_1 = 0.7$, we note a relative error of approximately $\approx 6\%$ between the analytic value of the option and either scheme when calculating the value of the option through the conditional expectation technique. For the case of $\sigma_1 = 0.9$, we see a larger relative error of $\approx 15\%$ between either scheme and the analytic value. For larger variances, there is a higher probability that the underlying assets may 'wander' towards the boundaries, leading to very large errors. Setting the boundary conditions further away from the natural logarithm of the initial price of the assets may reduce this error.

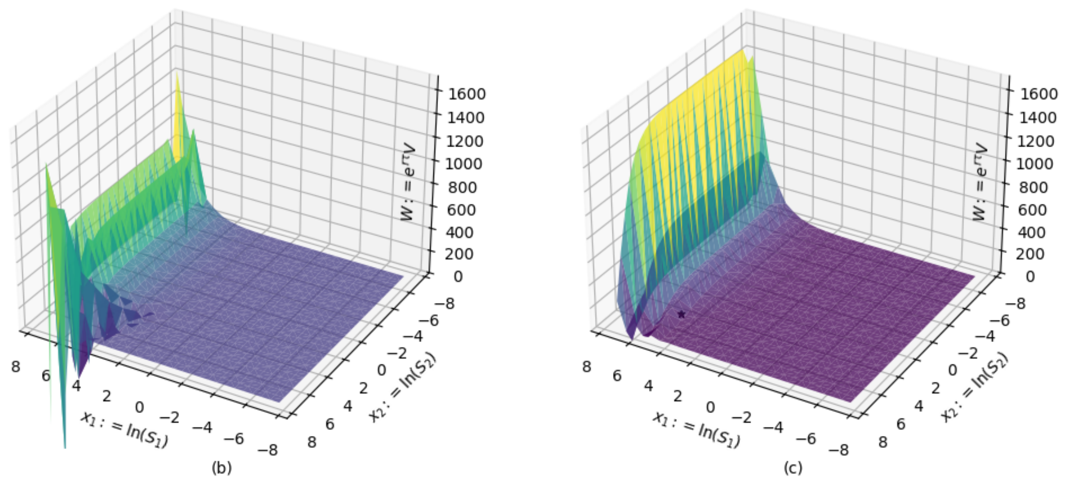


Figure 6.16: Plot of 'time-compounded' value of the option $W = e^{r\tau}V$ where $\tau = T_0$ versus the natural logarithms of the underlying asset prices $x_2 = \ln(S_2)$, $x_1 = \ln(S_1)$ for the (a) proper scheme with standard Dirichlet boundary conditions and (b) the improper scheme with modified Dirichlet boundary conditions for two assets with a volatility of the first asset $\sigma_1 = 0.9$ and simulation time of $T_0 = 2$. The natural logarithms of the two spot prices of the underlying asset prices $((x_2)_{\text{now}}, (x_1)_{\text{now}}) \approx (5.1, 4.5)$ is also indicated in the diagram with a star.

Description	Value
Proper scheme with standard Dirichlet boundary conditions	277.695
Improper scheme with modified Dirichlet boundary conditions	148.447
Analytic value at T_0	101.592
Conditional expectation of proper scheme with *std* Dirichlet B.C.s	95.389
Conditional expectation of improper scheme with *mod* Dirichlet B.C.s	96.119
Analytic value at $T_0 + T_1$	110.340

Table 6.15: The first set of three values corresponds to the values of the option at a time T_0 as calculated by examining the relevant component from the solution vector for the proper scheme with standard boundary conditions, the improper scheme with modified Dirichlet boundary conditions and the analytic value for the option from Margrabe's formula with $\sigma_1 = 0.9$ and $T_0 = 2$. The second set of values refer to the values of the option at a time $T_0 + T_1$ later as calculated from the conditional expectation using the proper scheme with standard boundary conditions, the improper scheme with modified Dirichlet boundary conditions and the analytic value for the option from Margrabe's formula, also with $\sigma_1 = 0.9$ and $T_0 = 2$.

6.3 Black-Scholes option pricing with HHL

We now present a working proof of principle example of a pricing a one dimensional European put option. As we are simulating this algorithm with a classical simulator, we are constrained by the depth and number of qubits of the circuit. With this fact in mind, we hope to demonstrate a single time step (with a second order Taylor approximation of the analytic solution for said time step) for the finite difference method and then estimate the price of the option by performing a SWAP test between the two quantum states. The price of an option (with no path-dependent requirements to consider) for a current asset price of S_0 can be expressed as:

$$\text{Value of option} = e^{-rT} \int_0^\infty V(T,S)\mathbb{P}(S|S_0)dS \tag{6.27}$$

The overall idea of the algorithm is to encode the amplitudes of $V(T,S)$ in a quantum state through execution of the HHL algorithm and then approximating the integral of (6.27) through a SWAP test between the previous state and another quantum state whose amplitudes encode the conditional probability distribution

$P(S|S_0)$.

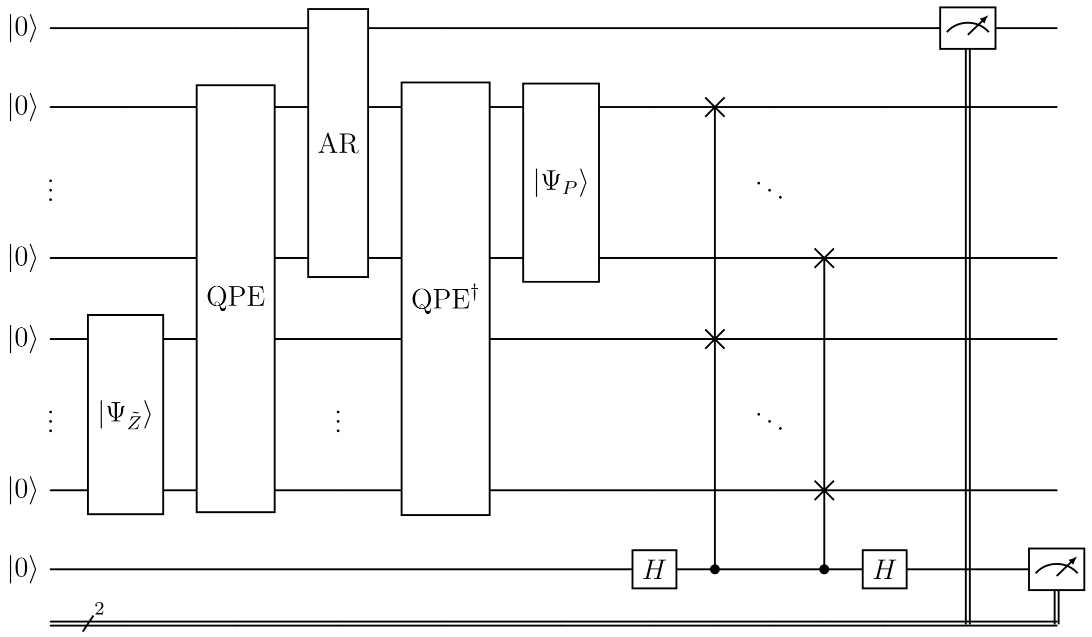


Figure 6.17: Quantum circuit for option pricing incorporating both the HHL algorithm and SWAP test.

Consider the following matrix equation:

$$\mathbf{M}\vec{Z} = \vec{V}$$

$$\underbrace{\begin{pmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\mathbf{A}\Delta t & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\frac{\mathbf{A}\Delta t}{2} & \mathbf{I} & \mathbf{0} \\ -\mathbf{I} & -\mathbf{I} & -\mathbf{I} & \mathbf{I} \end{pmatrix}}_{\mathbf{M}} \underbrace{\begin{pmatrix} \vec{z}_1 \\ \vec{z}_2 \\ \vec{z}_3 \\ \vec{z}_4 \end{pmatrix}}_{\vec{Z}} = \underbrace{\begin{pmatrix} \vec{V}_0 \\ \Delta t \vec{V}_1 \\ \vec{0} \\ \vec{0} \end{pmatrix}}_{\vec{V}} \tag{6.28}$$

The matrices **A** and **I** will each have dimension 16. The matrix **A** will correspond to the Black-Scholes operator differential operator with 16 points arising from a spatial discretization $[x_0 := x_{\text{lower}}, x_1, x_2, ..., x_{15} := x_{\text{upper}}]$ of the variable x which is the natural logarithm of the underlying asset price S ($x := \ln(S)$). The price of the option at time increment Δt later will be approximately encoded in each of the amplitudes of the $\vec{z}_4$ vector **only**. Furthermore, the vector $\vec{y}_4$ will be proportional to a vector where the j^{th} component will approximately be value of the option for a given asset price of $S = \exp(x_j)$ The matrix **M** is not Hermitian therefore we instead solve a 'Hermitian' variant of (6.28):

$$\tilde{\mathbf{M}}\vec{\tilde{Z}} = \vec{\tilde{V}}$$

$$\underbrace{\begin{pmatrix} \mathbf{0} & \mathbf{M} \\ \mathbf{M}^\dagger & \mathbf{0} \end{pmatrix}}_{\tilde{\mathbf{M}}} \underbrace{\begin{pmatrix} \vec{0} \\ \vec{Z} \end{pmatrix}}_{\vec{\tilde{Z}}} = \underbrace{\begin{pmatrix} \vec{V} \\ \vec{0} \end{pmatrix}}_{\vec{\tilde{V}}} \tag{6.29}$$

Reinserting the definition of (6.28) for the matrix $\mathbf{M}$ in terms of A and I transforms (6.29) as:

$$\underbrace{\begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{A}\Delta t & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\frac{\mathbf{A}\Delta t}{2} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{I} & -\mathbf{I} & -\mathbf{I} & \mathbf{I} \\ \mathbf{I} & -\mathbf{A}^\dagger\Delta t & \mathbf{0} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & -\frac{\mathbf{A}^\dagger\Delta t}{2} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & -\mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix}}_{\tilde{\mathbf{M}}} \underbrace{\begin{pmatrix} \vec{0} \\ \vec{0} \\ \vec{0} \\ \vec{0} \\ \vec{z}_1 \\ \vec{z}_2 \\ \vec{z}_3 \\ \vec{z}_4 \end{pmatrix}}_{\vec{\tilde{Z}}} = \underbrace{\begin{pmatrix} \vec{V}_0 \\ \Delta t\vec{V}_1 \\ \vec{0} \\ \vec{0} \\ \vec{0} \\ \vec{0} \\ \vec{0} \\ \vec{0} \end{pmatrix}}_{\vec{\tilde{V}}}$$

As discussed with the chapter discussing the HHL algorithm, we assume the spectrum of the matrix that we are inverting has a maximum absolute value of any of its eigenvalues to be one. Therefore, the actual matrix equation that will be solved by the HHL algorithm will be:

$$\frac{1}{|\lambda|_{\max}}\tilde{\mathbf{M}}\vec{\tilde{Z}} = \frac{1}{|\lambda|_{\max}}\vec{\tilde{V}} \qquad |\lambda|_{\max} := \{\max_i |\lambda_i| : \lambda_i \in \sigma(\tilde{\mathbf{M}})\}$$

With this caveat in mind,we proceed with the post-selection step of the HHL algorithm. Upon successful post-selection of the ancilla qubit in the HHL algorithm, we should have on the vector register of the quantum algorithm approximately the quantum state:

$$|\Psi_{\tilde{Z}}\rangle = \frac{1}{\sqrt{\sum_{i=1}^{4} \|\vec{z}_i\|^2}} \begin{pmatrix} \vec{0} \\ \vec{0} \\ \vec{0} \\ \vec{0} \\ \vec{z}_1 \\ \vec{z}_2 \\ \vec{z}_3 \\ \vec{z}_4 \end{pmatrix} \tag{6.30}$$

Before we proceed, we will need to estimate the norm of the solution $\vec{\tilde{Z}}$ before normalisation, (i.e. estimate the value of $\sqrt{\sum_{i=1}^{4} \|\vec{z}_i\|^2}$). This will be required as ultimately as we will see that the value of an option can be expressed as the time-discounted inner product of two not necessarily normalised vectors $\vec{P}$ and $\vec{z}_4$. The vector $\vec{P}$ here refers to a vector whose j^{th} component encodes the probability that the underlying asset price S (whose present value we assume is S_0) will undergo a movement in its price attaining a final value $S = \exp(x_j)$ at a time increment Δt_2 later. Recalling from the previous chapter (4.2.5), we can estimate the norm of the solution $\vec{x} = \mathbf{A}^{-1}\vec{b}$ with the HHL algorithm as:

$$\|\vec{x}\| = \frac{\left\|\vec{b}\right\| \sqrt{\text{Prob}\left(|1\rangle_{\text{Ancilla}}\right)}}{C} \tag{6.31}$$

Applied to the current case yields:

$$\left\|\vec{\tilde{Z}}\right\| = \sqrt{\sum_{i=1}^{4} \|\vec{z}_i\|^2} = \frac{\left\|\vec{\tilde{V}}\right\| \sqrt{\text{Prob}\left(|1\rangle_{\text{Ancilla}}\right)}}{C} \tag{6.32}$$

We assume a-priori that $\|\vec{\tilde{V}}\|$ is known. The parameter C is chosen beforehand based on the spectrum of the matrix to be inverted. Considering we need to estimate the inner product of $\vec{z}_4$ with $\vec{P}$, we need to estimate the quantity $\|\vec{z}_4\|$. This can be achieved by doing three further measurements on the state $|\Psi_{\tilde{Z}}\rangle$ which is the output of the HHL algorithm. To see this explicitly, consider the

Dirac notation representation of (6.30):

$$|\Psi_{\tilde{Z}}\rangle = \frac{1}{\sqrt{\sum_{i=1}^{4}\|\vec{z}_i\|^2}}\Big[|100\rangle \otimes \vec{z}_1 + |101\rangle \otimes \vec{z}_2 + |110\rangle \otimes \vec{z}_3 + |111\rangle \otimes \vec{z}_4\Big]$$

By post-selecting on the 'last' three qubits being simultaneously measured to be $|111\rangle$, we can reproduce a quantum state $|z_4\rangle$ that is proportional to the vector $\vec{z}_4$. The probability of measuring the state $|111\rangle$ is given by:

$$\text{Prob}\left(|111\rangle\right) = \left\| \frac{\vec{z}_4}{\sqrt{\sum_{i=1}^{4}\|\vec{z}_i\|^2}} \right\|^2$$

Or equivalently:

$$\|\vec{z}_4\| = \left(\sqrt{\sum_{i=1}^{4}\|\vec{z}_i\|^2}\right)\sqrt{\text{Prob}\left(|111\rangle\right)}$$

By inserting the definition of (6.32) for the value of $\sqrt{\sum_{i=1}^{4}\|\vec{z}_i\|^2}$ yields:

$$\begin{aligned}\|\vec{z}_4\| &= \frac{\left\|\vec{V}\right\|}{C}\sqrt{\text{Prob}(|111\rangle)}\sqrt{\text{Prob}\left(|1\rangle_{\text{Ancilla}}\right)} && (6.33)\\ &= \frac{\left\|\vec{V}\right\|}{C}\sqrt{\text{Prob}(|1111\rangle)} && (6.34)\end{aligned}$$

where we have dropped the explicit denoting of the ancilla qubit. Therefore, we can estimate the norm of the component we care about from the solution, namely $\vec{z}_4$ from examining the post-selection probabilities and the norm of the input vector $\left\|\vec{V}\right\|$ (which encodes the initial conditions and boundary conditions). After partial measurement of the state $|\Psi_{\tilde{Z}}\rangle$ in the vector register, we proceed with loading the state encoding the probability distribution of underlying asset price movement, namely $|P\rangle$. We assume some sort of oracle access for the preparation of both this state and the state encoding the initial and boundary conditions $|\Psi_{\tilde{V}}\rangle$. Once $|\Psi_P\rangle$ has been loaded, we proceed with the SWAP test between the states $|\Psi_{z_4}\rangle$ and $|\Psi_P\rangle$. As previously stated we can estimate the absolute value of the overlap as:

$$|\langle\Psi_P|\Psi_{z_4}\rangle| = \sqrt{1 - 2\,\text{Prob}\left(|1\rangle_{\text{SWAP}}\right)}$$

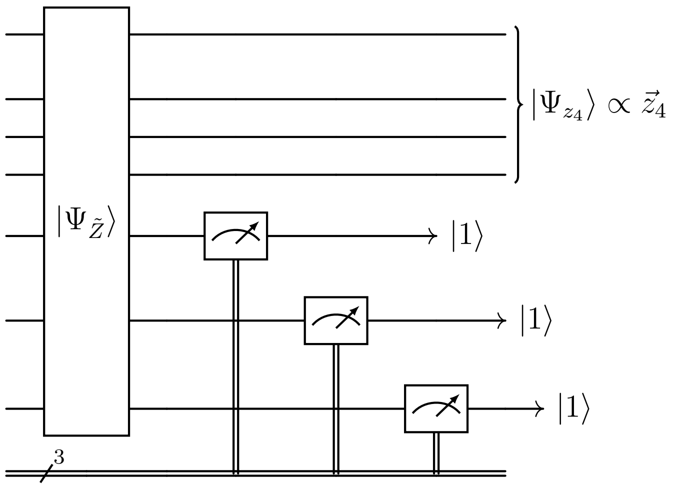
where the notation SWAP denotes the SWAP qubit that will ultimately be measured to determine the overlap between the two states. We can now calculate the inner-product of the two non-normalized states $\vec{P}$ and $\vec{z}_4$ as:

$$\left|\langle \vec{P}, \vec{z}_4 \rangle\right| = \left\|\vec{P}\right\| \|\vec{z}_4\| \, |\langle \Psi_P | \Psi_{z_4} \rangle| \tag{6.35}$$

$$= \left\|\vec{P}\right\| \left(\frac{\left\|\vec{\tilde{V}}\right\|}{C} \sqrt{\text{Prob}(|1111\rangle)} \sqrt{1 - 2\,\text{Prob}(|1\rangle_{\text{SWAP}})} \right) \tag{6.36}$$

As the value of the option is just the time discounted factor of (6.35), the value of the option will be:

$$\boxed{\text{Value of option} = \left(\frac{e^{-r(T_0+T_1)} \left\|\vec{P}\right\| \left\|\vec{\tilde{V}}\right\|}{C} \right) \sqrt{\text{Prob}(|1111\rangle)} \sqrt{1 - 2\,\text{Prob}\left(|1\rangle_{\text{SWAP}}\right)}}$$



$|\Psi_{\tilde{Z}}\rangle$
$|\Psi_{z_4}\rangle \propto \vec{z}_4$
$|1\rangle$
$|1\rangle$
$|1\rangle$
3

Figure 6.18: Partial measurement of $|\Psi_{\tilde{Z}}\rangle$ to determine the normalisation of $|z_4\rangle$

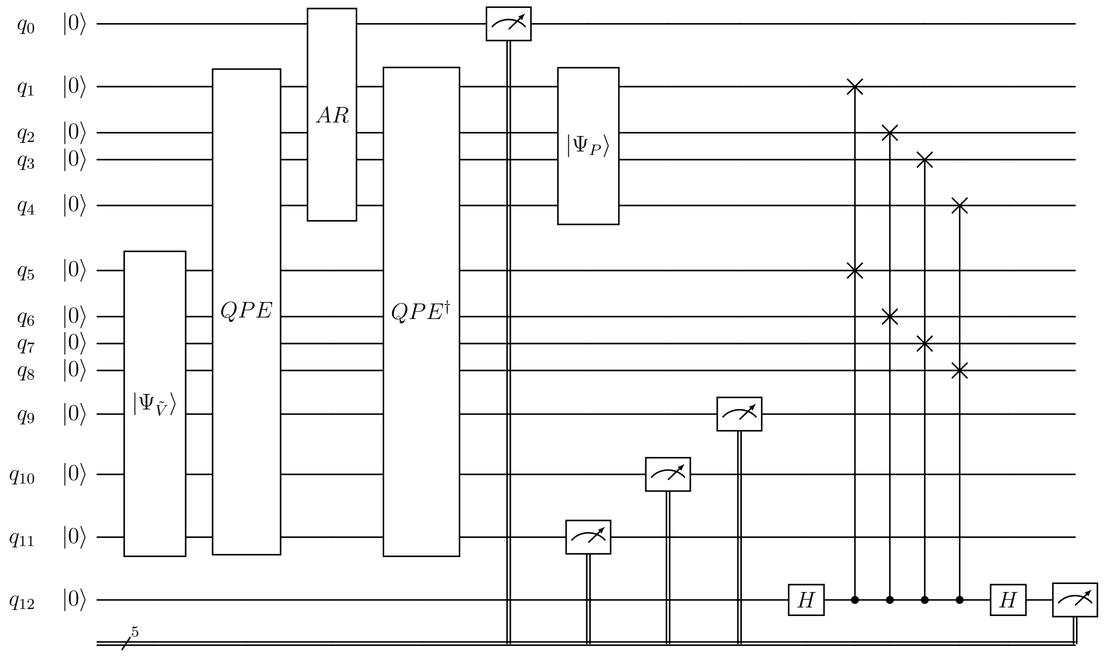


Figure 6.19: Full quantum circuit for current example. The block circuits denoted by $|\Psi_{\tilde{V}}\rangle$ and $|\Psi_P\rangle$ represent the loading of the associated 'normalised version' of the states $\vec{\tilde{V}}$ and $\vec{P}$ respectively.

6.3.1 Simulation parameters and option setup

We now select parameters relating to the option, discretization and quantum algorithm.

Option 1				
Parameter	Value	—	Parameter	Value
S_0	15	—	n_{grid}	16
K	40	—	$n_{\text{timesteps}}$	1
r	0.1	—	x_{lower}	0
σ	0.5	—	x_{upper}	5
T_0	0.25	—	Δx	0.3125
T_1	0.25	—	Δt	0.25
T_{total}	0.5	—	Order of Taylor approximation of analytic sol. for each step.	2

Table 6.16: Parameters relating to option pricing for the first option. S_0 denotes the spot price, K for the strike price of the asset, r for the risk-free interest rate, σ for the volatility, T_0 for the simulation period of the option with the HHL algorithm, T_1 for the time parameter associated with the probability distribution of possible underlying asset price movements.

Option 2				
Parameter	Value	—	Parameter	Value
S_0	30	—	n_{grid}	16
K	55	—	$n_{\text{timesteps}}$	1
r	0.05	—	x_{lower}	0
σ	0.45	—	x_{upper}	6
T_0	0.25	—	Δx	0.3125
T_1	0.25	—	Δt	0.25
T_{total}	0.5	—	Order of Taylor approximation of analytic sol. for each step.	2

Table 6.17: Parameters relating to option pricing for the second option. S_0 denotes the spot price, K for the strike price of the asset, r for the risk-free interest rate, σ for the volatility, T_0 for the simulation period of the option with the HHL algorithm, T_1 for the time parameter associated with the probability distribution of possible underlying asset price movements.

We choose larger than typical volatilities of $\sigma = 0.45, 0.5$ to increase the theoretical overlap $|\langle \Psi_P | \Psi_{z_4} \rangle|$ we should measure (which we calculated beforehand by directly solving the matrix). As we will see in due course, for an overlap ϵ, the number of executions of the circuit required to faithfully determine $|\langle \Psi_P | \Psi_{z_4} \rangle|$ scales as $\mathcal{O}(1/\epsilon^2)$. As each execution of the circuit took a relatively long time of approximately 5 seconds, we chose a larger volatility to reduce the total number of executions we would need to faithfully determine the overlap and thus price of the option. As we were effectively limited to a single timestep for simulation due to limited computational resources, we chose the simulation time T_0 itself to be small (comparable to a single timestep itself). We were also restricted to a coarse grid of just $n_{\text{grid}} = 16$ gridpoints due to limited computational resources. Therefore, unlike before where we selected the simulation region to be a symmetric interval of the form $[-c, c]$, we needed to chose an interval that was 'concentrated' around the initial price of the asset. We chose the interval $[x_{\text{lower}}, x_{\text{upper}}]$ to be of the form of $[\ln(S_0) - c, \ln(S_0) + c]$. If we had the luxury of a larger number of gridpoints, we could have just set the simulation region to be a symmetric interval as done previously. We determined c by classically solving the system for different values of c and comparing the resulting accuracy of the solution against the true price of the option.

Similar to the previous section for the case of single-asset option pricing, we need to create the vectors $\vec{V_0}$ and $\vec{V_1}$ which correspond to the initial and boundary conditions respectively. We reintroduce a simple function that translates from component index to grid points arising from the discretization of the variable x:

$$x(j) = \left(\frac{x_{\text{upper}} - x_{\text{lower}}}{n_{\text{grid}} + 1}\right)(j+1) + x_{\text{lower}}; \quad j \in [0, N-1]$$

We encode the initial condition which is just they payoff function evaluated at each of the grid points. As we are pricing a European put option, the payoff function P is $P(S) = \max(K - S, 0)$ and therefore the j^{th} component of the vector V_0 will be (assuming the first component of the vector is indexed by $j = 0$):

$$(\vec{V_0})_j = \max(K - \exp(x(j)), 0)$$

For the case of the boundary conditions, $\vec{V_1}$ will be the same as in (6.7):

$$(\vec{V_1})_0 = \left(\sigma^2 - r\right) K$$

Finally, the last piece of classical information that will be loaded into the quantum algorithm is $\vec{P}$ which encodes the probability distribution surrounding possible price movements in the underlying asset S. More precisely, the j^{th} component of $\vec{P}$ roughly corresponds to the probability that the underlying asset price S which has a price S_0 at $t = 0$, will attain the price $\exp(x(j))$ at time T later. Or equivalently with change of variables, the j^{th} component of $\vec{P}$ corresponds to the probability that the natural logarithm of the asset price x, will attain the value $x(j)$ at a time $t = T$ later given its initial value of $x_0 := \ln(S_0)$ at $t = 0$. As seen in chapter 2, the natural logarithm of the price of an underlying asset undergoing geometric Brownian motion is lognormally distributed with mean $\ln(S_0) + \left(r - \frac{\sigma^2}{2}\right)T$ and variance $\sigma^2 T$. Therefore the j^{th} component of $\vec{P}$ is defined as:

$$\left(\vec{P}\right)_j = \frac{1}{\sigma\sqrt{2\pi T}} \exp\left(-\frac{1}{2}\left(\frac{x(j) - \ln(S_0) - \left(r - \frac{\sigma^2}{2}\right)T}{\sigma\sqrt{T}}\right)^2\right)\Delta x$$

where Δx is the spacing of the grid.

Option 1				
Parameter	Value	—	Parameter	Value
$\dim(\tilde{\mathbf{M}})$	128	—	C	0.198
M_l	16	—	$\text{Ancilla}_{\text{qubits}}$	[0]
M	16	—	$\text{QPE}_{\text{qubits}}$	[1,2,3,4]
t_l	31.7	—	$\lvert\Psi_{\tilde{\text{V}}}\rangle$	[5,6,7,8,9,10,11]
t_u	50.3	—	$\lvert\Psi_{\text{P}}\rangle$	[1,2,3,4]
t	31.7	—	$\text{SWAP}_{\text{qubits}}$	[12]
$\left\lVert\vec{\tilde{V}}\right\rVert$	53.415	—	$\left\lVert\vec{P}\right\rVert$	0.742

Table 6.18: Parameters relating to the circuit to calculate the price of the first option. The parameter M denotes the No. of Fourier basis states, M_l denotes the lower bound for this No. The parameters t_l, t_u and t refer to the lower, upper bounds and actual selected time parameter t in $\exp(iAt/M)$ respectively. Finally, the parameter C refers to the ancilla normalisation constant from the HHL algorithm. Each set of integers in the table indicates on what qubits a subroutine takes place or where a quantum state is initially loaded

Option 2				
Parameter	Value	—	Parameter	Value
$\dim(\tilde{\mathbf{M}})$	128	—	C	0.191
M_l	16	—	$\text{Ancilla}_{\text{qubits}}$	[0]
M	16	—	$\text{QPE}_{\text{qubits}}$	[1,2,3,4]
t_l	34.7	—	$\lvert\Psi_{\tilde{\text{V}}}\rangle$	[5,6,7,8,9,10,11]
t_u	50.3	—	$\lvert\Psi_{\text{P}}\rangle$	[1,2,3,4]
t	34.7	—	$\text{SWAP}_{\text{qubits}}$	[12]
$\left\lVert\vec{\tilde{V}}\right\rVert$	59.448	—	$\left\lVert\vec{P}\right\rVert$	0.655

Table 6.19: Parameters relating to the circuit to calculate the price of the second option. The parameter M denotes the No. of Fourier basis states, M_l denotes the lower bound for this No. The parameters t_l, t_u and t refer to the lower, upper bounds and actual selected time parameter t in $\exp(iAt/M)$ respectively. Finally, the parameter C refers to the ancilla normalisation constant from the HHL algorithm. Each set of integers in the table indicates on what qubits a subroutine takes place or where a quantum state is initially loaded

6.3.2 Results

We use the Qiskit[28] `AerSimulator()` to classically simulate the circuit. Due to the extremely large depth of the transpiled circuit of approximately 1.1×10^6 and average gate fidelities for hardware, it was not possible to run the algorithm on actual hardware. We use the default `method='automatic'` of this simulator as the simulation style for the circuit. We simulate 500 executions (`shots=500`) of the circuit for option 1 and 750 executions (`shots=750`) for option 2. As we have not specified a noise model for the simulator, the simulation is noiseless. Before execution of the circuit, the `AerSimulator()` requires a transpilation of the circuit. We choose an `optimization_level=1` to strike a balance between light optimization of the circuit for faster execution times of the circuit while still faithfully implementing the circuit accurately. We chose a larger number of counts for the second option due to anticipated smaller overlap which would require more samples to estimate reliably. It would have been useful during execution of the circuit to halt simulation of the circuit if post-selection is unsuccessful (no matrix inversion) to reduce the simulation time of each of the shots. We examine this issue more thoroughly in the discussion section. For the case of implementing the unitary $\exp\left(\frac{iAt}{M}\right)$ that appears in the QPE step we utilized the `UnitaryGate()` method which constructs a circuit which approximates the matrix representation of the unitary operator U we are trying to implement. We initially had planned to implement a Suzuki-Trotter method given the explicit 'pseudo'-matrix decomposition we derived in chapter 5 however due to time concerns we were not able to directly implement it, we leave this project for future work. For the sake of clarity, we only present here the counts after successful post-selection of the ancilla qubit in the HHL algorithm subroutine.

Option 1					
First post-selection			—	Second post-selection	
No	Basis state $\|q_{12}q_{11}q_{10}q_9\rangle \|q_0\rangle$	Counts	—	Basis state $\|q_{12}\rangle \|q_{11}q_{10}q_9q_0\rangle$	Counts
0	$\|0000\rangle \|1\rangle$	3	—	-	-
1	$\|0001\rangle \|1\rangle$	0	—	-	-
2	$\|0010\rangle \|1\rangle$	2	—	-	-
3	$\|0011\rangle \|1\rangle$	0	—	-	-
4	$\|0100\rangle \|1\rangle$	72	—	-	-
5	$\|0101\rangle \|1\rangle$	0	—	-	-
6	$\|0110\rangle \|1\rangle$	1	—	-	-
7	$\|0111\rangle \|1\rangle$	53	—	$\|0\rangle \|1111\rangle$	53
8	$\|1000\rangle \|1\rangle$	0	—	-	-
9	$\|1001\rangle \|1\rangle$	1	—	-	-
10	$\|1010\rangle \|1\rangle$	1	—	-	-
11	$\|1011\rangle \|1\rangle$	0	—	-	-
12	$\|1100\rangle \|1\rangle$	53	—	-	-
13	$\|1101\rangle \|1\rangle$	2	—	-	-
14	$\|1110\rangle \|1\rangle$	0	—	-	-
15	$\|1111\rangle \|1\rangle$	42	—	$\|1\rangle \|1111\rangle$	42
Sum of counts		230	-	Sum of counts	95
Total executions of circuit				500	

Table 6.20: Counts for the circuit to calculate the value of the first option as provided by the Qiskit `AerSimulator()`. Although five qubits are ultimately measured, qubit q_0 corresponds to either successful or unsuccessful matrix inversion. Therefore, we only present counts where matrix inversion was successful ($q_0 = +1$). Furthermore, for the SWAP subroutine, we need to collapse the quantum state $|\Psi_{\tilde{Z}}\rangle$ to the quantum state $|\Psi_{z_4}\rangle$ by post-selecting for the state $|q_{11}q_{10}q_9\rangle = |111\rangle$. We indicate the post-selected states in the second column.

Option 1			
No. of $\|(.)\rangle \|1111\rangle$	Shots	Prob($\|1111\rangle$)	$\\|\vec{z}_4\\|$
95	500	0.190	117.621
No. of $\|0\rangle_{\text{SWAP}}$	No. of $\|1\rangle_{\text{SWAP}}$	Prob ($\|1\rangle_{\text{SWAP}}$)	$\|\langle\Psi_P\|\Psi_{z_4}\rangle\|$
53	42	0.442	0.340
$e^{-r(T_0+T_1)}$	$\text{Price}_{\text{Class}}$	$\text{Price}_{\text{Quantum}}$	Rel. error
0.951	27.805	28.228	1.521%

Table 6.21: Calculation of the value of the first option from count statistics.

Option 2					
First post-selection			—	Second post-selection	
No	Basis state $\lvert q_{12}q_{11}q_{10}q_9\rangle \lvert q_0\rangle$	Counts	—	Basis state $\lvert q_{12}\rangle \lvert q_{11}q_{10}q_9q_0\rangle$	Counts
0	$\lvert 0000\rangle \lvert 1\rangle$	4	—	-	-
1	$\lvert 0001\rangle \lvert 1\rangle$	2	—	-	-
2	$\lvert 0010\rangle \lvert 1\rangle$	0	—	-	-
3	$\lvert 0011\rangle \lvert 1\rangle$	0	—	-	-
4	$\lvert 0100\rangle \lvert 1\rangle$	90	—	-	-
5	$\lvert 0101\rangle \lvert 1\rangle$	2	—	-	-
6	$\lvert 0110\rangle \lvert 1\rangle$	0	—	-	-
7	$\lvert 0111\rangle \lvert 1\rangle$	81	—	$\lvert 0\rangle \lvert 1111\rangle$	81
8	$\lvert 1000\rangle \lvert 1\rangle$	3	—	-	-
9	$\lvert 1001\rangle \lvert 1\rangle$	0	—	-	-
10	$\lvert 1010\rangle \lvert 1\rangle$	0	—	-	-
11	$\lvert 1011\rangle \lvert 1\rangle$	0	—	-	-
12	$\lvert 1100\rangle \lvert 1\rangle$	61	—	-	-
13	$\lvert 1101\rangle \lvert 1\rangle$	0	—	-	-
14	$\lvert 1110\rangle \lvert 1\rangle$	1	—	-	-
15	$\lvert 1111\rangle \lvert 1\rangle$	69	—	$\lvert 1\rangle \lvert 1111\rangle$	69
Sum of counts		313	-	Sum of counts	150
Total executions of circuit				750	

Table 6.22: Counts for the circuit to calculate the value of the second option as provided by the Qiskit `AerSimulator()`. Although five qubits are ultimately measured, qubit q_0 corresponds to either successful or unsuccessful matrix inversion. Therefore, we only present counts where matrix inversion was successful ($q_0 = +1$). Furthermore, for the SWAP subroutine, we need to collapse the quantum state $\lvert\Psi_{\tilde{Z}}\rangle$ to the quantum state $\lvert\Psi_{z_4}\rangle$ by post-selecting for the state $\lvert q_{11}q_{10}q_9\rangle = \lvert 111\rangle$. We indicate the post-selected states in the second column.

Option 2			
No. of $\lvert(.)\rangle \lvert 1111\rangle$	Shots	Prob($\lvert 1111\rangle$)	$\lVert\vec{z}_4\rVert$
150	750	0.2	139.193
No. of $\lvert 0\rangle_{\text{SWAP}}$	No. of $\lvert 1\rangle_{\text{SWAP}}$	Prob ($\lvert 1\rangle_{\text{SWAP}}$)	$\lvert\langle\Psi_P\lvert\Psi_{z_4}\rangle\rvert$
81	69	0.46	0.283
$e^{-r(T_0+T_1)}$	$\text{Price}_{\text{Class}}$	$\text{Price}_{\text{Quantum}}$	Rel. error
0.975	23.811	25.173	5.72%

Table 6.23: Calculation of the value of the second option from count statistics.

6.3.3 Intermediate shot counts

We now provide intermediate count statistics for both the first and second option.

Option 1			
Batch No.	1		
Shots for run	100	Accumulated shots	100
Accumulated results			
No. of $\lvert 01111\rangle$	No. of $\lvert 11111\rangle$	Prob ($\lvert.\rangle \lvert 1111\rangle$)	Prob ($\lvert 1\rangle_{\text{SWAP}}$)
12	10	0.22	0.455
$\lvert\langle\Psi_P\lvert\Psi_{z_4}\rangle\rvert$	$\lVert\vec{z}_4\rVert$	$\text{Price}_{\text{Class}}$	$\text{Price}_{\text{Quantum}}$
0.302	122.145	27.805	26.036
		Rel. error	6.361 %

Table 6.24: Aggregated results for first circuit after 100 shots

Option 1			
Batch No.	2		
Shots for run	100	Accumulated shots	200
Accumulated results			
No. of $\|01111\rangle$	No. of $\|11111\rangle$	Prob $(\|.\rangle\,\|1111\rangle)$	Prob $(\|1\rangle_{\text{SWAP}})$
22	19	0.205	0.463
$\|\langle\Psi_P\|\Psi_{z_4}\rangle\|$	$\\|\vec{z}_4\\|$	$\text{Price}_{\text{Class.}}$	$\text{Price}_{\text{Quant.}}$
0.271	122.145	27.805	23.363
		Rel. error	15.976%

Table 6.25: Aggregated results for first circuit after 200 shots

Option 1			
Batch No.	3		
Shots for run	100	Accumulated shots	300
Accumulated results			
No. of $\|01111\rangle$	No. of $\|11111\rangle$	Prob $(\|.\rangle\,\|1111\rangle)$	Prob $(\|1\rangle_{\text{SWAP}})$
32	26	0.193	0.448
$\|\langle\Psi_P\|\Psi_{z_4}\rangle\|$	$\\|\vec{z}_4\\|$	$\text{Price}_{\text{Class.}}$	$\text{Price}_{\text{Quant}}$
0.322	118.52	27.805	26.935
		Rel. error	3.129%

Table 6.26: Aggregated results for first circuit after 300 shots

Option 1			
Batch No.	4		
Shots for run	100	Accumulated shots	400
Accumulated results			
No. of $\|01111\rangle$	No. of $\|11111\rangle$	Prob $(\|.\rangle\,\|1111\rangle)$	Prob $(\|1\rangle_{\text{SWAP}})$
42	35	0.193	0.455
$\|\langle\Psi_P\|\Psi_{z_4}\rangle\|$	$\\|\vec{z}_4\\|$	$\text{Price}_{\text{Class.}}$	$\text{Price}_{\text{Quant.}}$
0.302	118.362	27.805	25.23
		Rel. error	9.261%

Table 6.27: Aggregated results for first circuit after 400 shots

Option 1			
Batch No.	**5**		
Shots for run	100	Accumulated shots	500
Accumulated results			
No. of $\|01111\rangle$	No. of $\|11111\rangle$	$\text{Prob}\left(\|.\rangle\,\|1111\rangle\right)$	$\text{Prob}\left(\|1\rangle_{\text{SWAP}}\right)$
53	42	0.19	0.442
$\|\langle\Psi_P\|\Psi_{z_4}\rangle\|$	$\\|\vec{z}_4\\|$	$\text{Price}_{\text{Class.}}$	$\text{Price}_{\text{Quant.}}$
0.34	117.591	27.805	28.219
		Rel. error	1.521%

Table 6.28: Aggregated results for first circuit after 500 shots

Option 2			
Batch No.	**1**		
Shots for run	150	Accumulated shots	150
Accumulated results			
No. of $\|01111\rangle$	No. of $\|11111\rangle$	$\text{Prob}\left(\|.\rangle\,\|1111\rangle\right)$	$\text{Prob}\left(\|1\rangle_{\text{SWAP}}\right)$
15	18	0.22	0.545
$\|\langle\Psi_P\|\Psi_{z_4}\rangle\|$	$\\|\vec{z}_4\\|$	$\text{Price}_{\text{Class}}$	$\text{Price}_{\text{Quantum}}$
NaN	NaN	23.811	NaN
		Rel. error	NaN %

Table 6.29: Aggregated results for second circuit after 150 shots. We use the term ‘NaN’ to indicate that a complex number was calculated for that quantity and therefore omit it.

Option 2			
Batch No.	2		
Shots for run	150	Accumulated shots	300
Accumulated results			
No. of $\|01111\rangle$	No. of $\|11111\rangle$	$\text{Prob}\,(\|.\rangle\,\|1111\rangle)$	$\text{Prob}\,(\|1\rangle_{\text{SWAP}})$
31	32	0.21	0.508
$\|\langle\Psi_P\|\Psi_{z_4}\rangle\|$	$\|\|\vec{z}_4\|\|$	$\text{Price}_{\text{Class.}}$	$\text{Price}_{\text{Quant.}}$
NaN	NaN	23.811	NaN
		Rel. error	NaN%

Table 6.30: Aggregated results for second circuit after 300 shots. We use the term 'NaN' to indicate that a complex number was calculated for that quantity and therefore omit it.

Option 2			
Batch No.	3		
Shots for run	150	Accumulated shots	450
Accumulated results			
No. of $\|01111\rangle$	No. of $\|11111\rangle$	$\text{Prob}\,(\|.\rangle\,\|1111\rangle)$	$\text{Prob}\,(\|1\rangle_{\text{SWAP}})$
52	43	0.211	0.453
$\|\langle\Psi_P\|\Psi_{z_4}\rangle\|$	$\|\|\vec{z}_4\|\|$	$\text{Price}_{\text{Class.}}$	$\text{Price}_{\text{Quant}}$
0.308	143.008	23.811	28.129
		Rel. error	18.135%

Table 6.31: Aggregated results for second circuit after 450 shots

Option 2			
Batch No.	**4**		
Shots for run	150	Accumulated shots	600
Accumulated results			
No. of $\|01111\rangle$	No. of $\|11111\rangle$	Prob $(\|.\rangle \|1111\rangle)$	Prob $(\|1\rangle_{\mathrm{SWAP}})$
68	61	0.215	0.473
$\|\langle\Psi_P\|\Psi_{z_4}\rangle\|$	$\|\|\vec{z}_4\|\|$	$\mathrm{Price}_{\mathrm{Class.}}$	$\mathrm{Price}_{\mathrm{Quant.}}$
0.233	144.319	23.811	21.470
		Rel. error	9.832%

Table 6.32: Aggregated results for second circuit after 600 shots

Option 2			
Batch No.	**5**		
Shots for run	150	Accumulated shots	750
Accumulated results			
No. of $\|01111\rangle$	No. of $\|11111\rangle$	Prob $(\|.\rangle \|1111\rangle)$	Prob $(\|1\rangle_{\mathrm{SWAP}})$
81	69	0.2	0.46
$\|\langle\Psi_P\|\Psi_{z_4}\rangle\|$	$\|\|\vec{z}_4\|\|$	$\mathrm{Price}_{\mathrm{Class.}}$	$\mathrm{Price}_{\mathrm{Quant.}}$
0.283	139.193	23.811	25.143
		Rel. error	5.72%

Table 6.33: Aggregated results for second circuit after 750 shots

For the case of the first option, we see the relative error jump quite significantly from a maximum relative error of $\approx 15\%$ and a minimum relative error 1.5%. As the `AerSimulator()` is noiseless, this result is unexpected. We should expect the relative error to approximately decrease with increasing shot number. However, for the case of the second option we note that after the 3^{rd} batch of shots, the relative error consistently falls from 18.135% to 9.382% to a final value of 5.72% with subsequent batches.

6.3.4 Discussion

We now proceed with commenting on the results and identifying potential sources of error within the algorithm that could effect the results.

State preparation

One of the first issues to be addressed with this algorithm is the time complexity of loading the quantum states $|\Psi_{\tilde{V}}\rangle$ and $|\Psi_P\rangle$ which correspond to the boundary/initial conditions of the simulation and probability distribution of possible underlying price movements respectively. As the amplitudes of $|\Psi_P\rangle$ are proportional to a normal distribution, itself being a smooth differentiable function, there exists quantum algorithms such as [19] and [26] which may permit efficient loading of states corresponding to said distribution. The norm of this quantum state can also be efficiently computed as follows. If $f(x)$ corresponds to the standard normal distribution:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

then:

$$\begin{aligned}
\int_{-\infty}^{\infty} f(x)^2 dx &= \int_{-\infty}^{\infty} \frac{1}{(2\pi)\sigma^2} e^{-\left(\frac{x-\mu}{\sigma}\right)^2} dx \\
\tilde{\sigma} &:= \frac{\sigma}{\sqrt{2}} \\
&= \int_{-\infty}^{\infty} \frac{1}{(2\tilde{\sigma}^2)2\pi} e^{-\frac{(x-\mu)^2}{2\tilde{\sigma}^2}} dx = \frac{1}{\sqrt{8\pi}\tilde{\sigma}} \underbrace{\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\tilde{\sigma}}\right)^2} dx}_{1} \\
&= \frac{1}{\sqrt{8\pi}\tilde{\sigma}}
\end{aligned}$$

For sufficiently many qubits representing the state $\vec{P}$, we can estimate its norm as:

$$\left\|\vec{P}\right\| = \sqrt{\sum_{i=0}^{2^n-1} f(x_i)^2 \Delta x} \approx \sqrt{\int_{-\infty}^{\infty} f(x)^2 dx} = \frac{1}{\sqrt{\tilde{\sigma}\sqrt{8\pi}}} \tag{6.37}$$

We now consider the more general case of the vector $\vec{\tilde{V}}$ being equal to the vector $\vec{V}$ (which is the general form of the vector for encoding the initial and boundary conditions). The vector $\vec{V}$ has the following form:

$$\vec{V} = \left(\begin{array}{c}
\vec{V}_0 \\
(\Delta t)\vec{V}_1 \\
\vdots \\
\frac{(\Delta t)^{2^{m-1}}}{(2^{m-1})!}\vec{V}_{2^{m-1}} \\
\vec{0} \\
\vdots \\
\vec{0} \\
\hline
\vec{0} \\
(\Delta t)\vec{V}_1 \\
\vdots \\
\frac{(\Delta t)^{2^{m-1}}}{(2^{m-1})!}\vec{V}_{2^{m-1}} \\
\vec{0} \\
\vdots \\
\vec{0} \\
\hline
\vdots \\
\hline
\vec{0} \\
(\Delta t)\vec{V}_1 \\
\vdots \\
\frac{(\Delta t)^{2^{m-1}}}{(2^{m-1})!}\vec{V}_{2^{m-1}} \\
\vec{0} \\
\vdots \\
\vec{0} \\
\hline
\vec{0} \\
\vec{0} \\
\vdots \\
\vec{0}
\end{array}\right)$$

For the structure of the vector $\vec{V}$, we may able to efficiently estimate its norm $||\vec{V}||$. If the boundary and initial conditions are repeated 2^{p-r} times, where each term has a Taylor order approximation of the analytic solution of order $= 2^{m-1}$, then the norm of $\left\|\vec{V}\right\|$ can be estimated as:

$$\left\|\vec{V}\right\| = \sqrt{\left\|\vec{V_0}\right\|^2 + 2^{p-r}\left(\sum_{j=1}^{2^{m-1}} \frac{(\Delta t)^{2j}}{(j!)^2}\left\|\vec{V_j}\right\|^2\right)}$$

For the case of time independent boundary conditions, this expression simplifies as:

$$\left\|\vec{V}\right\| = \sqrt{\left\|\vec{V_0}\right\|^2 + 2^{p-r}(\Delta t)^2\left\|\vec{V_1}\right\|^2}$$

Without further information about the nature of each $\vec{V_j}$ for $j \in [1, \text{order}]$, it is difficult to quantity the circuit depth and or complexity of loading these quantum states. We leave this for future work.

Quantum Phase Estimation

The second caveat to be examined is the gate complexity of the unitary $\exp\left(\frac{i\tilde{\mathbf{M}}t}{m}\right)$ in the QPE subroutine of the HHL algorithm. The process of `UnitaryGate()` method decomposing the 128×128 matrix $\exp\left(\frac{i\tilde{\mathbf{M}}t}{m}\right)$ into a circuit dominated the computational resources required to simulate the circuit. This computational cost is what limited us to a grid of no more than 16 points and a single time step with a second order approximation of the analytical solution. Although as we ultimately calculated option through a conditional expectation, we believe that the underlying normal distribution enabled effectively an 'interpolation' between the grid points. After execution of the HHL algorithm, the value of the option for present underlying asset price of $\exp(x_i)$ should be approximately encoded at the i^{th} grid point. For only 16 grid points, this is quite a coarse grid to directly extract the price from. By calculating the price from considering the overlap from the output of HHL and a normal distribution (whose center we can position continuously), we are able to estimate the value of the option for underlying asset prices which is not 'close' to any of the 16 grid points listed above. Our results derived from this simple contrived example for pricing a single asset European option provide proof of the working principle of this algorithm. One thing we noticed in the execution of this algorithm was that the expected ancilla post-selection probability and measured ancilla post-selection probability within the HHL subroutine were slightly different. We can manipulate (6.31) to express the success probability in terms of C, $\left\|\vec{V}\right\|$ and $\left\|\vec{\tilde{Z}}\right\|$:

$$\text{Prob}(|1\rangle_{\text{Ancilla}}) = \frac{C^2 \left\|\vec{\tilde{Z}}\right\|^2}{\left\|\vec{V}\right\|^2}$$

Classically solving for $\vec{\tilde{Z}}$ yields a success probability of 41.2% whereas we measured the ancilla qubit with a success probability of 46%, a relative error between the two probabilities of 11.65%. We suspect this source of error arises from two primary sources. The first source being that the QPE subroutine of the algorithm is unavoidably probabilistic and may err in extracting the correct Fourier basis state which corresponds to the phase that is nearest to $\exp\left(\frac{i\lambda_j t}{M}\right)$. Here λ_j denotes the eigenvalues that we are trying to 'extract'. Notwithstanding the probabilistic element of the algorithm, there is only a finite number of Fourier basis states ($M = 2^m$) that can be used to encode all possible phases. Therefore, we can only represent the eigenvalues with finite precision (m bits namely) which will also introduce an error.

Ancilla Rotation

The second source of error may arise from AR subroutine. For the current proof of concept, we constructed the AR circuit such that for every possible Fourier basis state $|l\rangle$, we have appended a controlled Y-rotation conditioned on $|l\rangle$ where the rotation angle is analytically calculated from the formula $\theta(l) = 2\arcsin\left(\frac{C}{g(l)}\right)$. Although in the current case this would imply an exponentially increasing circuit depth with the number of qubits m, we proceeded in this manner as to potentially eliminate this source of error in the final result. However for the general case, one would consider a polynomial approximation to the function $\theta(l)$ and so called 'quantum arithmetic' circuits [51] to implement the polynomial approximation, in the process avoiding an exponentially increasing circuit depth with the number of qubits m in the computational register. Qiskit also includes a class `qiskit.PolyPauliRotations()` which can implement a quantum arithmetic circuit given a polynomial (in this instance the Taylor approximation of the function $\theta(l) = 2\arcsin\left(\frac{C}{g(l)}\right)$). Although we also construct the HHL circuit with this quantum arithmetic circuit, we did not use this circuit for collection of results we document. Other additional sources of error may arise with improper state preparation where we have used the native `qiskit.initialize()` function. As previously stated, the simulator backend will default to noiseless evolution and therefore this should not be another possible source of error. After successful ancilla post-selection and QPE†, we should have returned the computational register to the state $|0\rangle^{\otimes m}$ and the state $|\Psi_{\tilde{Z}}\rangle$ on the vector register. The return of the computational register to the canonical state $|0\rangle^{\otimes m}$ is crucial as the next step of the algorithm is to initialize the state $|\Psi_P\rangle$ for the subsequent SWAP test. To avoid possible sources of error due to imperfect 'uncomputation', we reset the computational qubits to the state $|0\rangle^{\otimes m}$ before initialization of $|\Psi_P\rangle$. Before the SWAP test, we have to perform a measurement on the vector register to 'collapse' the state $|\Psi_{\tilde{Z}}\rangle$ supported on seven qubits to the state $|\Psi_{z_4}\rangle$ supported on the top four qubits of the vector register. This collapse can be achieved by post-selecting for the state $|111\rangle$ on the bottom three qubits of the vector register. The success probability for measuring the state $|111\rangle$ also provides critical information about the normalisation of $\vec{z}_4$ (along with the success probability of the ancilla qubit in the HHL algorithm, utltimately measuring the state $|1111\rangle$). For the first option, classically computing $\|\vec{z}_4\|$ yields a value of 121.222 whereas a value of 117.498 is estimated for $\|\vec{z}_4\|$ from the success probability of measuring the state $|1111\rangle$, leading to a relative error of 3.072%. For the second option, the classically computed value of $\|\vec{z}_4\|$ is 139.243 compared to a value of 139.234 from the success probability of measuring $|1111\rangle$, with a relative error of 0.006%.

SWAP test

As the SWAP test can be represented exactly by controlled swap gates and Hadamard gates, there should not be any error associated with the circuit itself. The final step of the algorithm is measuring the SWAP qubit to determine the overlap of the two quantum states $|\langle\Psi_P|\Psi_{z_4}\rangle| = \sqrt{1 - 2\,\mathrm{Prob}\,(|1\rangle_{\mathrm{SWAP}})}$. The formula above roughly states that as the overlap between the two quantum states approaches zero, we should roughly measure the basis states $|0\rangle$ and $|1\rangle$ equiprobably and as the overlap reaches its maximum value of one, we should measure the state $|0\rangle$ with certainty. There is one subtlety to be addressed for the case of near-zero overlap. We can see from the above formula that the probability of measuring $|1\rangle$ should be less than or equal to 50% as the number of samples (executions and subsequent measurements of circuit) tends to infinity. However for a finite number of samples there is no guarantee that this criterion will be met and therefore it is possible to record probabilities that imply nonsensical (purely imaginary) values for the absolute value for the overlap between the two states $|\Psi_{z_4}\rangle$ and $|\Psi_P\rangle$. For example if we say the overlap between the states is ϵ then:

$$|\langle\Psi_P|\Psi_{z_4}\rangle| = \epsilon \Leftrightarrow \hat{p} := \text{Prob}(|1\rangle) = \frac{1-\epsilon^2}{2}$$

Then by considering the Wald confidence interval for the population probability p in terms of the sample probability $\hat{p}$ from n samples [50], we should have it lay completely inside the interval $[0, 0.5]$ to ensure an interpretable (real) overlap:

$$p \in \left[\hat{p} - z_\alpha\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}, \hat{p} + z_\alpha\sqrt{\frac{\hat{p}(1-\hat{p})}{n}}\right] \subseteq [0, 0.5]$$

where z_α relates to the z-score or confidence of the interval. As the above interval will first encounter the barrier of $\frac{1}{2}$ to the right of $\hat{p}$, it is sufficient to ensure the right endpoint of the confidence interval is less than $\frac{1}{2}$:

$$\begin{aligned}\hat{p} + z_\alpha\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} &\leq \frac{1}{2}\\ \frac{1-\epsilon^2}{2} + z_\alpha\sqrt{\frac{\left(\frac{1-\epsilon^2}{2}\right)\left(1-\left(\frac{1-\epsilon^2}{2}\right)\right)}{n}} &\leq \frac{1}{2}\\ z_\alpha\sqrt{\frac{(1-\epsilon^2)(1+\epsilon^2)}{4n}} &\leq \frac{\epsilon^2}{2}\\ \frac{z_\alpha}{\epsilon^2}\sqrt{1-\epsilon^4} &\leq n\end{aligned}$$

For small ϵ, we can approximate the square root as just one leaving:

$$\frac{z_\alpha}{\epsilon^2} \leq n \tag{6.38}$$

Therefore in order to approximate an overlap of ϵ reliably , we need at least $\mathcal{O}\left(\frac{1}{\epsilon^2}\right)$ samples. For example if we want to get an estimate of an overlap $\epsilon = 0.1$ with 95% confidence that it will correspond to meaningful result for the probability $\hat{p}$, we will need at least $1.96/(0.1)^2 \approx 200$ samples. Therefore, one criterion we suspect for this algorithm to offer speedup is having a non-negligible (or bounded away from zero) value for the quantity $|\langle\Psi_P|\Psi_{z_4}\rangle|$. We have seen some numerical evidence that for the overlap is generally larger for options that are out of the money versus in the money. We also note a larger overlap for higher volatilities, (which is characteristic of multi-asset options generally, they often comprise volatile underlying assets). For the first option, the classically computed value for $|\langle\Psi_P|\Psi_{z_4}\rangle|$ is 0.326 with a quantum calculated value of 0.340, yielding a relative error of 4.294%. For the second option, the classically computed value for $|\langle\Psi_P|\Psi_{z_4}\rangle|$ is 0.265 with a quantum calculated value of 0.283, yielding a relative error of 6.792%. We also highlight for the first round and second round of shots for the second option, we record a probability of greater than 50% for measuring the SWAP qubit in the $|1\rangle$ state, leading to a nonsensical (complex) value for the overlap. From (6.38), we should estimate that for an overlap of 0.265, approximately $\frac{1.96}{(0.265)^2} \approx 28$ samples would be sufficient. However, by the end of the second round of shots we have accumulated 63 samples which should be more than enough to provide a probability of less than 0.5 for measuring $|1\rangle$ (that being said the overlap as calculated from the counts in round 2 is only nonsensical due to the presence of a single shot being in the favour of $|1\rangle$ rather than $|0\rangle$).

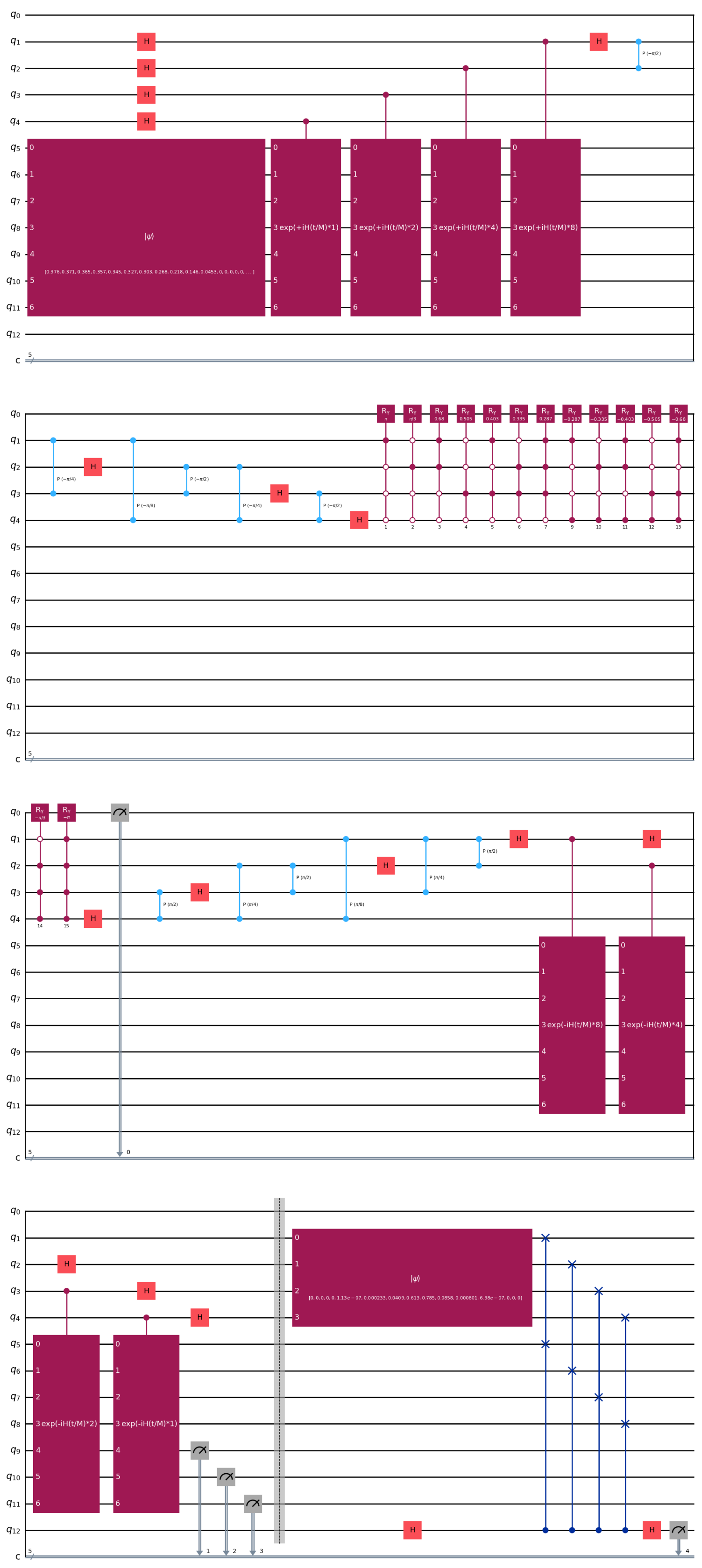


Figure 6.20: Quantum circuit to price the second option. The corresponding Qiskit code to generate these circuits is available from Github repository upon request.

Chapter 7

Conclusions

In this thesis, we examined the application of quantum algorithm for linear systems of equations (HHL) to option pricing within the area of quantitative finance. We took a first principles approach in terms of explicating the mathematical framework surrounding the problem of option pricing and the associated dynamics through the language of stochastic calculus. Guided by the governing equations that determine the price of an option, we then proceeded with examining the associated Black-Scholes partial differential equation and associated techniques to numerically approximate the solution via encoding in large matrices. Once equipped with the matrix formulation of the problem, we outlined the quantum algorithm for linear systems of equations (HHL) which may offer enormous computational advantage in solving linear systems of equations given certain caveats. We believe our major contribution in this thesis is the characterization of parameters related to numerical linear algebra and simulation of a newly-defined 'improper' Black-Scholes differential operator $\tilde{\mathbf{A}}$. We show that that this operator can be unitarily diagonalized, implying an optimal condition number of 1 for the matrix that diagonalizes it. We also place bounds on the spectral norm of the matrix, the spectrum itself and sparsity of the matrix. All of the above parameters, in particular the condition number parameter, are critical to determining the potential speed-up compared to classical methods with the HHL algorithm. Having shown the above results, we then proceeded with comparing the performance of this algorithm in terms of numerically simulating option pricing for single-asset and multi-asset options as well as outlining how the boundary conditions can be corrected for the case of the improper Black-Scholes differential operator $\tilde{\mathbf{A}}$. For the case of the single-asset option, we see virtually exact agreement between both the ‘proper’ and ‘improper’ schemes (related to the proper and improper Black-Scholes differential operator). However, for the case of the multi-asset option, we see some discrepancy between the two schemes when calculating the option price through direct extraction from the solution vector. In spite of this, we see an improvement in this error when extracting the price via the conditional expectation technique. The significance of this result is that the quantum algorithm broadly replicates the latter process and therefore we could potentially price options faithfully on a quantum computer with much more desirable scaling then previously thought. We also remark that, in some instances, the correction term in the boundary conditions for the improper Black-Scholes differential operator leads to no difference compared to when just the standard Dirichlet boundary conditions are imposed. We found this behaviour to be unexpected and we intend to examine this further in future work. We tested the performance of the two approaches and their dependence on parameters related both to the option and discretization setup. We leave for future work a more comprehensive analysis of dependence of the performance between the two approaches and various possible option and discretization setups.

Finally, we provided a novel proof of principles simulation of single-asset option pricing with the HHL algorithm and SWAP test. We simulated two single-asset options. We note in either case that with increasing shot number, the value of either option is calculated within 1.5% and 5.6% respectively. Due to limits on available computational resources, we were only able to simulate a single-asset option for a single timestep with a coarse grid of 16 gridpoints. We hope to extend simulations to multi-asset options with more realistic discretization setups in the future with access to greater computational resources.

In conclusion, this thesis has explored the feasibility of modifying the Black-Scholes differential operator yielding more favourable complexity scaling without compromising numerical accuracy. Finally in this thesis, we presented a novel end-to-end implementation of a quantum algorithm for option pricing. We believe this to be an important step for a potential realization of this algorithm in an industry setting.

Appendices

Appendix A

Derivations relevant to stochastic calculus

A.1 *Box calculus* rules for Wiener process

We now proceed with a derivation of the multiplication rules or *Box calculus* associated with the Wiener process:

$$dW dt = 0 \quad dW^2 = dt \quad dt^2 = 0$$

We proceed as follows:

$$dW = W_{t+\Delta t} - W_t \quad dt = \Delta t$$

$$\mathbb{E}\left((W_{t+\Delta t} - W_t)^2\right) = \mathbb{E}(W_{t+\Delta t}^2) - 2\mathbb{E}(W_{t+\Delta t}W_t) + \mathbb{E}(W_t^2)$$

As $\mathrm{Var}(X) = \mathbb{E}(X^2) - \mathbb{E}(X)^2$

$$\mathbb{E}(W_{t+\Delta t}^2) = \underbrace{\mathrm{Var}(W_{t+\Delta t})}_{t+\Delta t} + \underbrace{\mathbb{E}(W_{t+\Delta t})^2}_{0} = t + \Delta t$$

$$\mathbb{E}(W_t^2) = \underbrace{\mathrm{Var}(W_t)}_{t} + \underbrace{\mathbb{E}(W_t)^2}_{0} = t$$

For the case of the term $\mathbb{E}(W_{t+\Delta t}W_t)$:

$$\mathbb{E}(W_{t+\Delta t}W_t) = \mathbb{E}\left([W_{t+\Delta t} - W_t + W_t]\, W_t\right)$$

$$\rightarrow \mathbb{E}((W_{t+\Delta t} - W_t)\, W_t) + \underbrace{\mathbb{E}(W_t^2)}_{t}$$

By the independence of the increments, the above expression can be rewritten as:

$$\mathbb{E}(W_{t+\Delta t} - W_t) \underbrace{\mathbb{E}(W_t)}_{0} + t \Rightarrow t$$

Hence, this implies:

$$\Delta W^2 = \mathbb{E}((W_{t+\Delta t} - W_t)^2) = t + \Delta t - 2t + t = \Delta t$$

$$dW^2 = dt$$

In the limit as $\Delta t \rightarrow 0$, the final multiplication rule is arrived at. For proof of the second multiplication rule $dWdt = 0$, we can first see that:

$$dW = \mathbb{E}(W_{t+\Delta t} - W_t) = \underbrace{\mathbb{E}(W_{t+\Delta t})}_{0} - \underbrace{\mathbb{E}(t)}_{0} = 0$$

Reusing the independence of increments again yields:

$$dWdt = \mathbb{E}((W_{t+\Delta t} - W_t)\Delta t) = \underbrace{\mathbb{E}(W_{t+\Delta t} - W_t)}_{0} \mathbb{E}(\Delta t) = 0$$

A.2 Joint probability distribution for multiple correlated assets

$$dS_i(t) = rS_i(t)dt + \sigma_i S_i(t)dW_i(t) \qquad \forall i \in I := \{1, 2, ..., d\} \tag{A.1}$$

However, due to non-independence of the Brownian motions $\{W_1, W_2, ..., W_d\}$, we must effectively change to a new set of independent Brownian motions $\{Z_1, Z_2, ..., Z_3\}$. This will permit us to describe the joint probability distribution of the assets. We will express each correlated Brownian motion as a linear combination of the uncorrelated Brownian motions:

$$\begin{aligned}
W_1 &= a_{11}Z_1 + a_{12}Z_2 + ... + a_{1d}Z_d \\
W_2 &= a_{21}Z_1 + a_{22}Z_2 + ... + a_{2d}Z_d \\
&\vdots \\
W_d &= a_{d1}Z_1 + a_{d2}Z_2 + ... + a_{dd}Z_d
\end{aligned}$$

As each of $\{W_i\}$ are Brownian motions in their own right, their variances must each be equal to the time elapsed:

$$\text{Var}(W_i) = t \qquad \forall i \in I$$

Secondly, there are additional constraints due to the pairwise correlations between $\{W_i\}_{i\in I}$. These constraints can be imposed recursively. Namely, for the n^{th} Brownian motion W_n, we demand that it satisfies the correlation constraints for all indices less than n:

$$\text{Corr}(W_n, W_i) = \rho_{ni} \qquad 1 \leq i \leq n$$

These two conditions are sufficient to determine each of the coefficients a_{ij}. To see how, consider the first Brownian motion W_1. Using the properties of the variance one can see:

$$\begin{aligned}
\text{Var}(W_1) &= \sum_{i=1}^{d} a_{1i}^2 \underbrace{\text{Var}(Z_i)}_{=t} + 2\sum_{i<j} \underbrace{\text{Cov}(Z_i, Z_j)}_{=0} = t \\
t &= \sum_{i=1}^{d} a_{1i}^2 t \Rightarrow \sum_{i=1}^{d} a_{1i}^2 = 1
\end{aligned}$$

The pairwise covariance of each the Brownian motions is zero by construction. In terms of correlations, the only condition to be satisfied is $\text{Corr}(W_1, W_1) = \rho_{11} := 1$. The previous statement is somewhat vacuous as we do not have another distinct Brownian motion to impose a correlation constraint on (in fact, yields the exact same constraint as the previous one). Hence, by convention we choose the first coefficient $a_{11} = 1$ and the rest to be equal to zero:

$$W_1 = Z_1$$

Considering the second Brownian motion, we obtain the same constraint for the sum of the squares of the coefficients due to the requirement $\mathrm{Var}(W_2) = t$:

$$\sum_{i=1}^{d} a_{2i}^2 = 1 \tag{A.2}$$

However, a non-trivial correlation constraint now exists:

$$\mathrm{Corr}(W_2, W_1) = \rho_{21} \qquad \mathrm{Corr}(X, Y) := \frac{\mathrm{Cov}(X, Y)}{\sqrt{\mathrm{Var}(X)\mathrm{Var}(Y)}}$$

$$\begin{aligned}
\mathrm{Corr}(W_2, W_1) &= \frac{\mathrm{Cov}(W_2, W_1)}{\sqrt{\mathrm{Var}(W_2)W}} \\
\rho_{21} &= \frac{\mathrm{Cov}(\sum_{i=1}^{d} a_{2i}Z_i, Z_1)}{\sqrt{t^2}} = \frac{\sum_{i=1}^{d} a_{2i}\mathrm{Cov}(Z_i, Z_1)}{t} \\
\rho_{21} &= \frac{\sum_{i=1}^{d} a_{2i}\delta_{i1}t}{t} = a_{21}
\end{aligned}$$

Returning to equation (A.2) we see that:

$$\underbrace{a_{21}^2}_{\rho_{21}} + \sum_{i=2}^{d} a_{2i}^2 = 1 \Rightarrow \sum_{i=2}^{d} a_{2i} = 1 - \rho_{21}^2$$

By convention, we set $a_{22} = \sqrt{1 - \rho_{21}^2}$ and the rest of the coefficients zero.

$$\begin{aligned}
W_1 &= Z_1 \\
W_2 &= \rho_{21}Z_1 + \sqrt{1 - \rho_{21}^2}Z_2 \\
&\vdots \\
W_d &= a_{d1}Z_1 + a_{d2}Z_2 + ... + a_{dd}Z_d
\end{aligned}$$

For the sake of clarity of how this procedure generalises, we explicitly show how the term W_3 can be calculated:

$$\begin{aligned}
\rho_{31} &= \text{Corr}(W_3, W_1) \\
\rho_{31}t &= \sum_{i=1} a_{3i}\text{Cov}(Z_i, Z_1) = a_{31}t \\
\implies a_{31} &= \rho_{31}
\end{aligned}$$

$$\begin{aligned}
\rho_{32} &= \text{Corr}(W_3, W_2) \\
\rho_{32}t &= \text{Cov}(\rho_{21}Z_1 + \sqrt{1-\rho_{21}^2}Z_2, \sum_{i=1}^{d} a_{3i}Z_i) \\
\rho_{32}t &= \sum_{i=1}^{d} a_{3i}\rho_{21}\text{Cov}(Z_1, Z_i) + \sum_{i=1}^{d} a_{3i}\sqrt{1-\rho_{21}^2}\text{Cov}(Z_2, Z_i) \\
\rho_{32} &= a_{31}\rho_{21} + a_{32}\sqrt{1-\rho_{21}^2} = \rho_{31}\rho_{21} + a_{32}\sqrt{1-\rho_{21}^2} \\
\implies a_{32} &= \frac{\rho_{32} - \rho_{31}\rho_{21}}{\sqrt{1-\rho_{21}^2}}
\end{aligned}$$

Finally, considering the total variance constraint of each of the Brownian motions, and continuing with the convention that the coefficients $a_{ki} = 0, \forall i > k$ we get:

$$\begin{aligned}
\sum_{i=1}^{d} a_{3i}^2 &= \rho_{31}^2 + \left(\frac{\rho_{32} - \rho_{31}\rho_{21}}{\sqrt{1-\rho_{21}^2}}\right)^2 + \sum_{i=3}^{d} a_{3i}^2 \\
1 &= \rho_{31}^2 + \left(\frac{\rho_{32} - \rho_{31}\rho_{21}}{\sqrt{1-\rho_{21}^2}}\right)^2 + a_{33}^2 \\
a_{31} &= \sqrt{1 - \rho_{31}^2 - \left(\frac{\rho_{32} - \rho_{31}\rho_{21}}{\sqrt{1-\rho_{21}^2}}\right)^2}
\end{aligned}$$

Finally, we can begin to see the relationship between the correlated Brownian motions and uncorrelated Brownian motions:

$$\begin{aligned}
W_1 &= Z_1 \\
W_2 &= \rho_{21}Z_1 + \sqrt{1-\rho_{21}^2}Z_2 \\
W_3 &= \rho_{31}Z_1 + \frac{\rho_{32}-\rho_{31}\rho_{21}}{\sqrt{1-\rho_{21}^2}}Z_2 + \sqrt{1-\rho_{31}^2-\left(\frac{\rho_{32}-\rho_{31}\rho_{21}}{\sqrt{1-\rho_{21}^2}}\right)^2}Z_3 \\
&\vdots \\
W_d &= a_{d1}Z_1 + a_{d2}Z_2 + ... + a_{dd}Z_d
\end{aligned}$$

If we define $\vec{\mathbf{W}} := (W_1, W_2, ..., W_d)$ and $\vec{\mathbf{Z}} := (Z_1, Z_2, ..., Z_d)$. The above relationships can be represented in matrix form as:

$$\begin{pmatrix} W_1 \\ W_2 \\ \vdots \\ W_d \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ \rho_{21} & \sqrt{1-\rho_{21}^2} & & \vdots \\ \vdots & & \ddots & \\ a_{d1} & \dots & & a_{dd} \end{pmatrix} \begin{pmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_d \end{pmatrix} \tag{A.3}$$

$$\vec{\mathbf{W}} = \mathrm{Chol}(\boldsymbol{\rho})\vec{\mathbf{Z}} \tag{A.4}$$

The notation, Chol($\boldsymbol{\rho}$), in equation (A.4) refers to the Cholesky decomposition of the correlation matrix $\boldsymbol{\rho}$ and is defined by:

$$\mathrm{Chol}(\boldsymbol{\rho})\left(\mathrm{Chol}(\boldsymbol{\rho})\right)^T = \boldsymbol{\rho}$$

$$\begin{pmatrix} 1 & \rho_{12} & \dots & \rho_{1d} \\ \rho_{12} & 1 & & \\ \vdots & & \ddots & \\ \rho_{1d} & & & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ \rho_{12} & \sqrt{1-\rho_{12}^2} & & \vdots \\ \vdots & & \ddots & \\ \rho_{1d} & \dots & & \end{pmatrix} \begin{pmatrix} 1 & \rho_{21} & \dots & \rho_{1d} \\ 0 & \sqrt{1-\rho_{12}^2} & & \\ \vdots & & \ddots & \\ 0 & \dots & & \end{pmatrix} \tag{A.5}$$

A heuristic proof to explain the presence of the Cholesky decomposition in the description of correlated variables is detailed in [41]. The components of $\vec{\mathbf{W}}$ consist wholly of standard Brownian motions, namely $\mathrm{Var}(W_i) = t, \quad \forall i$. However, different assets may have a variety of volatilities as to how they evolve, reflected in the various σ_i in the d SDEs in equation (A.1). To capture the differing volatilities in the matrix equation (A.3), we define a volatility matrix $\boldsymbol{\sigma}$:

$$\boldsymbol{\sigma} = \begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_d \end{pmatrix} \tag{A.6}$$

Hence, the modified Brownian motion 'vector' that captures the individual volatilities is simply $\boldsymbol{\sigma}\vec{\mathbf{W}}$. As each quantity $\ln(S_i)$ has a respective mean $(r - \frac{\sigma_i^2}{2})t$, the final matrix equation can be represented as:

$$\ln(\vec{\mathbf{S}}) = \vec{\boldsymbol{\mu}} + \boldsymbol{\sigma}\,\mathrm{Chol}(\boldsymbol{\rho})\vec{\mathbf{Z}}$$

Where $\vec{\boldsymbol{\mu}} = \Big((r - \frac{\sigma_1^2}{2})t, (r - \frac{\sigma_2^2}{2})t, ..., (r - \frac{\sigma_d^2}{2})t\Big)$ and $\ln(\vec{\mathbf{S}})$ refers to taking the logarithm component-wise (i.e $(\ln(S_1), \ln(S_2), ..., \ln(S_d))$). Finally, we can convert from this current formulation to the standard normal distribution using the following theorem on multivariate distributions from [20]. If $\vec{\mathbf{X}} := (X_1, X_2, ..., X_d)$ and $\vec{\mathbf{Z}} := (Z_1, Z_2, ..., Z_d)$, it can be shown that:

$$\begin{gathered} \mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \Leftrightarrow \exists \quad \boldsymbol{\mu} \in \mathbb{R}^d, \quad \mathbf{A} \in \mathbb{R}^{d\times d} \\ \text{such that} \quad \mathbf{X} = \boldsymbol{\mu} + \mathbf{A}\mathbf{Z}; \quad \text{with} \quad \boldsymbol{\Sigma} := \mathbf{A}\mathbf{A}^T; \quad Z_i \sim \mathcal{N}(0,1) \end{gathered}$$

As each of the Z_i in the current formulation are standard Brownian motions (implying $\mathrm{Var}(Z_i) = t$), a multiplicative factor of $\frac{1}{\sqrt{t}}$ is needed to ensure the standard normal distribution requirements of unit variance for each of the variables Z_i are satisfied:

$$\ln(\vec{\mathbf{S}}) = \vec{\boldsymbol{\mu}} + \sqrt{t}\boldsymbol{\sigma}\,\mathrm{Chol}(\boldsymbol{\rho}) \underbrace{\left(\frac{1}{\sqrt{t}}\vec{\mathbf{Z}}\right)}_{\sim\mathcal{N}(0,1)}$$

We can now deduce that $\mathbf{\Sigma}$ is given by:

$$\begin{aligned}
\mathbf{\Sigma} &= \left(\sqrt{t}\boldsymbol{\sigma}\,\mathrm{Chol}(\boldsymbol{\rho})\right)\left(\sqrt{t}\boldsymbol{\sigma}\,\mathrm{Chol}(\boldsymbol{\rho})\right)^T \\
&= t\left(\boldsymbol{\sigma}\,\mathrm{Chol}(\boldsymbol{\rho})\,\mathrm{Chol}(\boldsymbol{\rho})^T\boldsymbol{\sigma}^T\right) \\
&= t\boldsymbol{\sigma}\boldsymbol{\rho}\boldsymbol{\sigma}
\end{aligned}$$

Or in matrix notation:

$$\mathbf{\Sigma} = \begin{pmatrix} t\sigma_1^2 & t\rho_{12}\sigma_1\sigma_2 & \dots & t\rho_{1d}\sigma_1\sigma_d \\ t\rho_{12}\sigma_1\sigma_2 & t\sigma_2^2 & & \vdots \\ \vdots & & \ddots & \\ t\rho_{1d}\sigma_1\sigma_d & \dots & & t\sigma_d^2 \end{pmatrix} =$$

$$\begin{pmatrix} \sqrt{t}\sigma_1 & & & \\ & \sqrt{t}\sigma_2 & & \\ & & \ddots & \\ & & & \sqrt{t}\sigma_d \end{pmatrix}\begin{pmatrix} 1 & \rho_{12} & \dots & \rho_{1d} \\ \rho_{12} & 1 & & \vdots \\ \vdots & & \ddots & \\ \rho_{1d} & \dots & & 1 \end{pmatrix}\begin{pmatrix} \sqrt{t}\sigma_1 & & & \\ & \sqrt{t}\sigma_2 & & \\ & & \ddots & \\ & & & \sqrt{t}\sigma_d \end{pmatrix}$$

Hence, the joint probability distribution for $\ln(\vec{\mathbf{S}})$ is given by:

$$\ln(\vec{\mathbf{S}}) \sim \mathcal{N}(\vec{\boldsymbol{\mu}}, \mathbf{\Sigma})$$
$$f(\vec{\mathbf{x}}) = \frac{1}{\sqrt{(2\pi)^d\det(\mathbf{\Sigma})}}\exp\left(-\frac{1}{2}(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}})^T\mathbf{\Sigma}^{-1}(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}})\right)$$

Or equivalently:

$$\mathbb{P}\left(\ln(\vec{\mathbf{S}}) \in \Pi_{i=1}^d(x_i, x_i+\delta x_i)\right) \approx \frac{\exp\left(-\frac{1}{2}(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}})^T\mathbf{\Sigma}^{-1}(\vec{\mathbf{x}}-\vec{\boldsymbol{\mu}})\right)}{\sqrt{(2\pi)^d\det(\mathbf{\Sigma})}}\left(\Pi_{i=1}^d\delta x_i\right)$$

Appendix B

Derivation of sparsity inequalities

Consider the sparsity of a matrix which is defined as:

$$\mathrm{Spar}(A) := \max\left(\max_j\left(\sum_k \mathbb{1}_{[a_{jk}\neq 0]}\right), \max_j\left(\sum_k \mathbb{1}_{[a_{kj}\neq 0]}\right)\right)$$

where $\mathbb{1}_{[(.)]}$ refers to the Iverson bracket:

$$\mathbb{1}_{[x]} = \begin{cases} 1 & \text{if } x \text{ is True} \\ 0 & \text{if } x \text{ is False} \end{cases}$$

We now provide a proof of the two sparsity results stated previously:

$$\mathrm{Spar}(A \otimes B) \leq \mathrm{Spar}(A)\mathrm{Spar}(B)$$

$$\mathrm{Spar}(A + B) \leq \mathrm{Spar}(A) + \mathrm{Spar}(B)$$

To see this, consider the two square matrices A, B where $\dim(A) = n, \dim(B) = m$.

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{pmatrix} \qquad B = \begin{pmatrix} b_{11} & \dots & b_{1m} \\ \vdots & & \vdots \\ b_{m1} & \dots & b_{mm} \end{pmatrix}$$

Consider the tensor product of these matrices $A \otimes B$ and the $(jm + j')^{th}$ row.

$$\mathrm{Row}_{(jm+j')}(A \otimes B) =$$

$$\Big(a_{j1}(b_{j'1}, b_{j'2}, ..., b_{j'm}) \quad a_{j2}(b_{j'1}, b_{j'2}, ..., b_{j'm}) \quad \dots \quad a_{jn}(b_{j'1}, b_{j'2}, ..., b_{j'm})\Big)$$

The number of nonzero elements in this row will simply be the number of non zero elements in the $(j')^{th}$ row of B times the number of non zero elements in the j^{th} row of A.

$$\begin{aligned} &= \left(\sum_{k=1}^{n} \mathbb{1}_{[a_{jk}\neq 0]}\right)\left(\sum_{k=1}^{m} \mathbb{1}_{[b_{j'k}\neq 0]}\right) \\ &\leq \left(\sum_{k=1}^{n} \mathbb{1}_{[a_{jk}\neq 0]}\right) \mathrm{Spar}(B) \end{aligned}$$

where the inequality arises from the fact that the sparsity of B is defined as the maximum of the number of non zero elements in any one row or column however we have just considered the rows of B in the above example. Taking the max over all rows in $A \otimes B$ is:

$$\max_{j,j'}\left(\left(\sum_{k=1}^{n} \mathbb{1}_{[a_{jk}\neq 0]}\right) \mathrm{Spar}(B)\right) = \left(\max_{j}\left(\sum_{k=1}^{n} \mathbb{1}_{[a_{jk}\neq 0]}\right)\right) \mathrm{Spar}(B)$$

$$\Rightarrow \# \binom{\text{Non zero elements in}}{\text{any one row of } A \otimes B} \leq \left(\max_{j}\left(\sum_{k=1}^{n} \mathbb{1}_{[a_{jk}\neq 0]}\right)\right) \mathrm{Spar}(B)$$

The exact same result can be derived analogously for the number of non zero elements in any one column:

$$\# \binom{\text{Non zero elements in}}{\text{any one col of } A \otimes B} \leq \left(\max_{j}\left(\sum_{k=1}^{n} \mathbb{1}_{[a_{kj}\neq 0]}\right)\right) \mathrm{Spar}(B)$$

Therefore we can finally conclude :

$$\mathrm{Spar}(A \otimes B) = \max\left(\# \binom{\text{Non zero elements in}}{\text{any one row of } A \otimes B}, \# \binom{\text{Non zero elements in}}{\text{any one col of } A \otimes B}\right)$$

$$\leq \max\left[\left(\max_{j}\left(\sum_{k=1}^{n} \mathbb{1}_{[a_{jk}\neq 0]}\right)\right) \mathrm{Spar}(B) \right.$$
$$\left. , \left(\max_{j}\left(\sum_{k=1}^{n} \mathbb{1}_{[a_{kj}\neq 0]}\right)\right) \mathrm{Spar}(B)\right]$$

$$= \max\left[\max_j\left(\sum_{k=1}^{n}\mathbb{1}_{[a_{jk}\neq 0]}\right), \max_j\left(\sum_{k=1}^{n}\mathbb{1}_{[a_{kj}\neq 0]}\right)\right]\text{Spar}(B)$$

$$:= \text{Spar}(A)\text{Spar}(B)$$

To see the case of the inequality for Spar$(A+B)$, consider again the case of the j^{th} row of $A+B$ where $\dim(A) = \dim(B) = n$:

$$\text{Row}_{(j)}(A+B) = (a_{j1}+b_{j1}, a_{j2}+b_{j2}, ..., a_{jn}+b_{jn})$$

We now review two small lemmas required for the next statement. If we have two statements p and q such that $p \to q$ then:

$$\mathbb{1}_{[p]} \leq \mathbb{1}_{[q]}$$

The second lemma that we will use relates to an inequality regarding the disjunction of two statements p and q to the addition of their associated Iverson brackets:

$$\mathbb{1}_{[p\vee q]} \leq \mathbb{1}_{[p]} + \mathbb{1}_{[q]}$$

where $p \vee q$ is equivalent to p OR q. Now consider the statement that if two numbers are zero so is there sum and its associated contrapositive statement:

$$\begin{aligned}
&\text{if}\,((a_{jk}=0)\wedge(b_{jk}=0)) \to (a_{jk}+b_{jk}) = 0\\
\Leftrightarrow&(a_{jk}+b_{jk}) \neq 0 \to \neg\,((a_{jk}=0)\wedge(b_{jk}=0))\\
\Leftrightarrow&(a_{jk}+b_{jk}) \neq 0 \to (a_{jk}\neq 0)\vee(b_{jk}\neq 0)
\end{aligned}$$

Therefore:

$$\begin{aligned}
\mathbb{1}_{[(a_{jk}+b_{jk})\neq 0]} &\leq \mathbb{1}_{[(a_{jk}\neq 0)\vee(b_{jk}\neq 0)]}\\
&\leq \mathbb{1}_{[a_{jk}\neq 0]} + \mathbb{1}_{[b_{jk}\neq 0]}
\end{aligned}$$

We now return to bounding the sparsity of $A + B$. Consider the number of non-zero elements in the k^{th} row of $(A + B)$:

$$\sum_{k=1}^{n} \mathbb{1}_{[(a_{jk}+b_{jk})\neq 0]} \leq \sum_{k=1}^{n} \mathbb{1}_{[a_{jk}\neq 0]} + \sum_{k=1}^{n} \mathbb{1}_{[b_{jk}\neq 0]}$$

$$\max_j \left(\sum_{k=1}^{n} \mathbb{1}_{[(a_{jk}+b_{jk})\neq 0]} \right) \leq \max_j \left(\sum_{k=1}^{n} \mathbb{1}_{[a_{jk}\neq 0]} \right) + \max_j \left(\sum_{k=1}^{n} \mathbb{1}_{[b_{jk}\neq 0]} \right)$$

Again the exact same result can be shown for the columns of the matrix by considering the transpose of $A + B$ (which does not alter sparsity).

$$\max_j \left(\sum_{k=1}^{n} \mathbb{1}_{[(a_{kj}+b_{kj})\neq 0]} \right) \leq \max_j \left(\sum_{k=1}^{n} \mathbb{1}_{[a_{kj}\neq 0]} \right) + \max_j \left(\sum_{k=1}^{n} \mathbb{1}_{[b_{kj}\neq 0]} \right)$$

Therefore:

$$\text{Spar}(A+B) = \max \left[\max_j \left(\sum_{k=1}^{n} \mathbb{1}_{[(a_{jk}+b_{jk})\neq 0]} \right), \max_j \left(\sum_{k=1}^{n} \mathbb{1}_{[(a_{kj}+b_{kj})\neq 0]} \right) \right]$$

$$\leq \max \left[\max_j \left(\sum_{k=1}^{n} \mathbb{1}_{[a_{jk}\neq 0]} \right) + \max_j \left(\sum_{k=1}^{n} \mathbb{1}_{[b_{jk}\neq 0]} \right) \right.$$
$$\left. , \max_j \left(\sum_{k=1}^{n} \mathbb{1}_{[a_{kj}\neq 0]} \right) + \max_j \left(\sum_{k=1}^{n} \mathbb{1}_{[b_{kj}\neq 0]} \right) \right]$$

$$\leq \max \Big[\text{Spar}(A) + \text{Spar}(B), \text{Spar}(A) + \text{Spar}(B) \Big] = \text{Spar}(A) + \text{Spar}(B)$$

Bibliography

[1] Scott Aaronson. “Read the fine print”. In: *Nature Physics* 11.4 (2015), pp. 291–293.

[2] Krishna B Athreya and Soumendra N Lahiri. *Measure theory and probability theory*. Vol. 19. Springer, 2006.

[3] Christian Bayer et al. “Quasi-Monte Carlo for Efficient Fourier Pricing of Multi-Asset Options”. In: *arXiv preprint arXiv:2403.02832* (2024).

[4] Dominic W Berry and Pedro CS Costa. “Quantum algorithm for time-dependent differential equations using Dyson series”. In: *Quantum* 8 (2024), p. 1369.

[5] Dominic W Berry et al. “Quantum algorithm for linear differential equations with exponentially improved dependence on precision”. In: *Communications in Mathematical Physics* 356 (2017), pp. 1057–1081.

[6] Fischer Black and Myron Scholes. “The pricing of options and corporate liabilities”. In: *Journal of political economy* 81.3 (1973), pp. 637–654.

[7] Zvi Bodie and Alex Kane. *Investments*. McGraw Hill, 2020.

[8] Gilles Brassard and Peter Hoyer. “An exact quantum polynomial-time algorithm for Simon’s problem”. In: *Proceedings of the Fifth Israeli Symposium on Theory of Computing and Systems*. IEEE. 1997, pp. 12–23.

[9] Geon Ho Choe et al. *Stochastic analysis for finance with simulations*. Springer, 2016, p. 225.

[10] Don Coppersmith. “An approximate Fourier transform useful in quantum factoring”. In: *arXiv preprint quant-ph/0201067* (2002).

[11] Pedro CS Costa et al. “Optimal scaling quantum linear-systems solver via discrete adiabatic theorem”. In: *PRX quantum* 3.4 (2022), p. 040303.

[12] John C Cox, Stephen A Ross, and Mark Rubinstein. “Option pricing: A simplified approach”. In: *Journal of financial Economics* 7.3 (1979), pp. 229–263.

[13] James W Demmel. *Applied numerical linear algebra.* SIAM, 1997.

[14] Monroe David Donsker. *An invariance principle for certain probability limit theorems.* 1951.

[15] Frank J Fabozzi. *Handbook of finance, financial markets and instruments.* Vol. 1. John Wiley & Sons, 2008.

[16] Richard P Feynman. “Simulating physics with computers”. In: *Feynman and computation.* cRc Press, 2018, pp. 133–153.

[17] Forrest Flesher. “Stochastic Processes and the Feynman-Kac Theorem”. In: (2021).

[18] Robert Geske and Herb E Johnson. “The American put option valued analytically”. In: *The Journal of Finance* 39.5 (1984), pp. 1511–1524.

[19] Lov Grover and Terry Rudolph. “Creating superpositions that correspond to efficiently integrable probability distributions”. In: *arXiv preprint quant-ph/0208112* (2002).

[20] Allan Gut. *An Intermediate Course in Probability.* 2nd. Springer Publishing Company, Incorporated, 2009. ISBN: 1441901612.

[21] Brian C Hall and Brian C Hall. *Lie groups, Lie algebras, and representations.* Springer, 2013.

[22] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. “Quantum algorithm for linear systems of equations”. In: *Physical review letters* 103.15 (2009), p. 150502.

[23] Naomichi Hatano and Masuo Suzuki. “Finding exponential product formulas of higher orders”. In: *Quantum annealing and other optimization methods.* Springer, 2005, pp. 37–68.

[24] J. Hull. *Options, Futures, and Other Derivatives.* Prentice Hall, 2012. ISBN: 9780132164948.

[25] John C Hull and Sankarshan Basu. *Options, futures, and other derivatives.* Pearson Education India, 2016.

[26] Jason Iaconis, Sonika Johri, and Elton Yechao Zhu. “Quantum state preparation of normal distributions using matrix product states”. In: *npj Quantum Information* 10.1 (2024), p. 15.

[27] Kiyosi Itô. “On a formula concerning stochastic differentials”. In: *Nagoya Mathematical Journal* 3 (1951), pp. 55–65.

[28] Ali Javadi-Abhari et al. *Quantum computing with Qiskit.* 2024. DOI: `10.48550/arXiv.2405.08810`. arXiv: `2405.08810 [quant-ph]`.

[29] Lishang Jiang and Canguo Li. *Mathematical modeling and methods of option pricing.* World Scientific, 2005.

[30] Herb Johnson. “Options on the maximum or the minimum of several assets”. In: *Journal of Financial and Quantitative analysis* 22.3 (1987), pp. 277–283.

[31] Mark Kac. “On distributions of certain Wiener functionals”. In: *Transactions of the American Mathematical Society* 65.1 (1949), pp. 1–13.

[32] Ioannis Karatzas and Steven E. Shreve. *Brownian motion and stochastic calculus.* 2nd ed. Springer, 1991.

[33] Alastair Kay. “Tutorial on the quantikz package”. In: *arXiv preprint arXiv:1809.03842* (2018).

[34] Lin Lin and Yu Tong. “Near-optimal ground state preparation”. In: *Quantum* 4 (2020), p. 372.

[35] K. Miyamoto and K. Kubo. “Pricing Multi-Asset Derivatives by Finite-Difference Method on a Quantum Computer”. In: *IEEE Transactions on Quantum Engineering* 3.01 (Jan. 2022), pp. 1–25. ISSN: 2689-1808. DOI: `10.1109/TQE.2021.3128643`.

[36] Michael A Nielsen and Isaac L Chuang. *Quantum computation and quantum information.* Vol. 2. Cambridge university press Cambridge, 2001.

[37] Silvia Noschese, Lionello Pasquini, and Lothar Reichel. “Tridiagonal Toeplitz matrices: properties and novel applications”. In: *Numerical linear algebra with applications* 20.2 (2013), pp. 302–326.

[38] Peter Ouwehand and Graeme West. “Pricing rainbow options”. In: *Wilmott magazine* 5 (2006), pp. 74–80.

[39] Xinbo Li Christopher Phillips. "Detailed Error Analysis of the HHL Algorithm". In: *arXiv preprint arXiv:2401.17182* (2024).

[40] George F Pinder. *Numerical methods for solving partial differential equations: a comprehensive introduction for scientists and engineers.* John Wiley & Sons, 2018.

[41] Steven Roman. *Introduction to the Mathematics of Finance: Arbitrage and Option Pricing.* 2nd ed. Undergraduate Texts in Mathematics. Springer-Verlag New York, 2012. ISBN: 1461435811,9781461435815.

[42] Peter S Rose. *Money and capital markets: The financial system in the economy.* Business Publications, Incorporated, 1986.

[43] Christoph Schwab, N Hilber, and C Winter. "Computational methods for quantitative finance". In: *Lecture Notes-ETHZ* (2007).

[44] Jonathan Richard Shewchuk et al. "An introduction to the conjugate gradient method without the agonizing pain". In: (1994).

[45] Peter W Shor. "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". In: *SIAM review* 41.2 (1999), pp. 303–332.

[46] Dirk Sierag and Bernard Hanzon. "Pricing derivatives on multiple assets: recombining multinomial trees based on Pascal's simplex". In: *Annals of Operations Research* 266 (2018), pp. 101–127.

[47] Yiğit Subaşı, Rolando D Somma, and Davide Orsucci. "Quantum algorithms for systems of linear equations inspired by adiabatic quantum computing". In: *Physical review letters* 122.6 (2019), p. 060504.

[48] Endre Süli. *A Brief Introduction to the Numerical Analysis of PDEs.*

[49] Lloyd N Trefethen and David Bau. *Numerical linear algebra.* SIAM, 2022.

[50] Sean Wallis. "Binomial confidence intervals and contingency tests: mathematical fundamentals and the evaluation of alternative methods". In: *Journal of quantitative linguistics* 20.3 (2013), pp. 178–208.

[51] Siyi Wang et al. "A Comprehensive Study of Quantum Arithmetic Circuits". In: *arXiv preprint arXiv:2406.03867* (2024).

[52] Paul Wilmott. *Paul Wilmott on quantitative finance*. John Wiley & Sons, 2013.

[53] Paul Wilmott, Sam Howison, and Jeff Dewynne. *The mathematics of financial derivatives: a student introduction*. Cambridge university press, 1995.

[54] Jinsha Zhao. “American option valuation methods”. In: *International Journal of Economics and Finance* 10.5 (2018).