# Accelerated Grids

Optimizing Solvers for Financial Partial
Differential Equations

**Mustafa Berke Erdis, ID 180883925**

Supervisor: Dr. Sebastian del Bano Rollin



A thesis presented for the degree of
Master in Sciences in *Mathematical Finance*

School of Mathematical Sciences
and *School of Economics and Finance*
Queen Mary University of London

# Declaration of original work

This declaration is made on August 11, 2019.

**Student's Declaration:** I, Mustafa Berke Erdis, hereby declare that the work in this thesis is my original work. I have not copied from any other students' work, work of mine submitted elsewhere, or from any other sources except where due reference or acknowledgement is made explicitly in the text, nor has any part been written for me by another person.

Referenced text has been flagged by:

1. Using italic fonts, **and**

2. using quotation marks ".....", **and**

3. explicitly mentioning the source in the text.

This work is dedicated to my family.

# Acknowledgements

Here you thank people that have helped you in the journey.

Vivamus vehicula leo a justo. Quisque nec augue. Morbi mauris wisi, aliquet vitae, dignissim eget, sollicitudin molestie, ligula. In dictum enim sit amet risus. Curabitur vitae velit eu diam rhoncus hendrerit. Vivamus ut elit. Praesent mattis ipsum quis turpis. Curabitur rhoncus neque eu dui. Etiam vitae magna. Nam ullamcorper. Praesent interdum bibendum magna. Quisque auctor aliquam dolor. Morbi eu lorem et est porttitor fermentum. Nunc egestas arcu at tortor varius viverra. Fusce eu nulla ut nulla interdum consectetuer. Vestibulum gravida. Morbi mattis libero sed est.

# Abstract

Here you write a short summary, around 10 lines, of your work.
Vivamus vehicula leo a justo. Quisque nec augue. Morbi mauris wisi, aliquet vitae, dignissim eget, sollicitudin molestie, ligula. In dictum enim sit amet risus. Curabitur vitae velit eu diam rhoncus hendrerit. Vivamus ut elit. Praesent mattis ipsum quis turpis. Curabitur rhoncus neque eu dui. Etiam vitae magna. Nam ullamcorper. Praesent interdum bibendum magna. Quisque auctor aliquam dolor. Morbi eu lorem et est porttitor fermentum. Nunc egestas arcu at tortor varius viverra. Fusce eu nulla ut nulla interdum consectetuer. Vestibulum gravida. Morbi mattis libero sed est.

# Preface

Here you write a summary of the work. A paragraph on the motivation, previous work, then maybe a brief chapter by chapter summary.

Vivamus vehicula leo a justo. Quisque nec augue. Morbi mauris wisi, aliquet vitae, dignissim eget, sollicitudin molestie, ligula. In dictum enim sit amet risus. Curabitur vitae velit eu diam rhoncus hendrerit. Vivamus ut elit. Praesent mattis ipsum quis turpis. Curabitur rhoncus neque eu dui. Etiam vitae magna. Nam ullamcorper. Praesent interdum bibendum magna. Quisque auctor aliquam dolor. Morbi eu lorem et est porttitor fermentum. Nunc egestas arcu at tortor varius viverra. Fusce eu nulla ut nulla interdum consectetuer. Vestibulum gravida. Morbi mattis libero sed est.

Queen Mary University of London
12$^{\text{th}}$ August 2019

# Contents

6

# Chapter 1

# Introduction

In Ancient Greece, Thales was scorned for his poverty. Later that year, Thales utilized his skills in astrology to forecast an increase in olive yields. Using his limited capital, he rented oil presses in winter. Months later, over the oil making season, many people rushed to the presses because of the high yields that Thales predicted. As he rented the presses over the winter, he forced the terms he pleased. Thales showed it was easy for philosophers to be rich if they chose it and practically used the first financial derivative product [1].

In the modern world, financial derivatives are contracts between two or more parties. The value of the contract depends on one or several underlying assets. Commonly the assets are currencies, equities, bonds, interest rates, market indices or commodities. The vanilla call option gives the right but not the obligation to buy the underlying asset at the expiry date at a previously agreed strike price. Essentially, Thales bought call options for oil presses. If the olive yields didn't come as Thales expected he didn't have the obligation to use the olive presses. On the other hand, the vanilla put option gives the right but not the obligation to sell the underlying asset at the expiry date at a previously agreed strike price. Practical applications of the options include hedging or speculating the future asset price. Hence, accurately pricing the

options is crucial for an efficient and mature financial market.

Merton and Scholes received the 1997 Nobel Prize in Economic Science for this work [12].

## 1.1 Motivation of the Project

Derivative pricing in the real world is a computationally intensive task. The existing numerical methods for partial differential equations are all constrained by the computational complexity. Being fast when evaluating new information is critical for the operations of hedge funds and investment banks. Therefore optimizing the existing numerical methods with hardware and software that can be installed on a trading floor is crucial. Goal of the project is to provide efficient methods for pricing options.

Purpose of this project to optimize numerical solutions of parabolic PDEs by testing high performance computing techniques and comparing compilers/os/32bit/64bit. The idea of this project is to study how to take advantage of this parallelism and explore how much faster we can make these calculations.

## 1.2 Literature Review

Bakılacak paperlar ekle.

## 1.3 Contents of the Thesis

Included in your 'Introduction' section should be a clear summary of what you have achieved in the project work presented, such as any new results, generalisations, corollaries, examples, new connections, or computer investigations. The thesis is organized as follows. Chapter 2 presents an introduction to financial derivatives, Black-Scholes model and finite difference models.

Chapter3 extends the 2d and gives examples of two dimensional heat equation. Chapter 4 develops the numerical methods and techniques considered to solve the PDE-based models for the option pricing problems. In the same chapter, finite difference method with improved algorithms to solve a large tridiagonal systems is discussed. Chapter 5 shows the numerical results of our numerical methods with several examples of. Chapter 6 concludes the thesis.

# Chapter 2

# Pricing Financial Derivatives

## 2.1   The Risk Neutral Approach

The Black-Scholes framework is a theoretical valuation formula for options. It reveals the relationship between the prices of the options and the underlying assets. Since almost all corporate liabilities can be viewed as combinations of options, the formula is applicable to common stocks and corporate bonds [2]. The Black-Scholes model makes the following assumptions:

- There does not exist any arbitrage opportunity in the financial market. The traders can't make instantaneous profit without any risk.

- The underlying asset value follows a geometric Brownian Motion $dS = \mu S dt + \sigma S dB$ where $\mu$ denotes the average rate of growth of the underlying assets, $\sigma$ denotes the volatility of the asset price and B is a Brownian Motion.

- The market is frictionless. This means there are no transaction fees, the interest rates for borrowing and lending money from and to the bank are the same, every party in market has immediate information and all entities are available at anytime and in any size.

- No dividend will paid on the underlying asset $S$.

### 2.1.1   Black-Scholes Partial Differential Equation

The original model is used to price the vanilla option, which is the simplest type of option. Under the assumptions of Black-Scholes framework, the call or put option price satisfies the parabolic partial differential equation:

$$\frac{\partial V}{\partial t} = rS\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} - rV \tag{2.1}$$

Framework shows the price V(t,S) of a European option driven by one underlying asset that satisfies the PDE where $r$, $\sigma$, $t$, $S$ respectively denotes the risk-free interest rate, volatility, time and the underlying price. It is assumed that $r$ and $\sigma$ are constants. In more complicated models such as stochastic volatility models they can be a function. In order to price a vanilla call option the PDE needs to satisfy the following boundary and initial conditions.

$$C(0,t) = 0, \ C(S_{\max},t) = S_{\max} - Ke^{-r(T-t)}, \quad 0 \le t \le T \tag{2.2}$$

$$C(S,T) = \max(S-K,0), \quad 0 \le S \le S_{\max} \tag{2.3}$$

### 2.1.2   Derivation of the Black-Scholes Equation

Black-Scholes model takes advantage of the properties of the geometric Brownian motion and Itô's lemma.

**Definition 2.1.1.** Brownian Motion Brownian motion (also known as Wiener Process) was discovered by botanist Robert Brown as he observed a chaotic motion of particles suspended in water [16]. A Brownian motion, $B(t)$, is a continuous-time stochastic process with the following properties:

- $B(0) = 0$.

- $B(t)$ is a continuous function of t.

- For $0 \leq s < t$ the increment $B(t) - B(s)$ has normal distribution $\mathcal{N}(0, t - s)$.

Brownian motion is the basic building block in stochastic calculus and geometric Brownian motion is used to model the stock prices in Black-Scholes model.

**Lemma 2.1.2.** *Itô's Lemma Let $B(t)$ be a Brownian motion and $X(t)$ be an Ito process which satisfies the stochastic differential equation:*

$$dX(t) = \mu(X(t), t)dt + \sigma(X(t), t)dB(t) \tag{2.4}$$

*If $f(x, t) \in C^2(\mathbb{R}^2, \mathbb{R})$ then $f(X(t), t)$ is also an Ito drift-diffusion process, with its differential given by:*

$$d(f(X(t), t)) = \frac{\partial f}{\partial t}(X(t), t)dt + f'(X(t), t)dX + \frac{1}{2}f''(X(t), t)dX(t)^2 \tag{2.5}$$

*With $dX(t)^2$ given by: $dt^2 = 0$, $dtdB(t) = 0$ and $dB(t)^2 = dt$.*

**Theorem 2.1.3.** *Assume that the asset price $S$ follows a geometric Brownian motion. Under the assumptions of Black-Scholes framework, the call or put option price $V(t, S)$ satisfies the parabolic partial differential equation*

$$\frac{\partial V}{\partial t} = rS\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} - rV \tag{2.6}$$

*Proof.* Suppose an investor sets up a self-financing portfolio, $X(t)$, comprising one option and an $\Delta$ amount of the underlying asset. Therefore, value of the portfolio at time t is $X(t) = V(t) + \Delta S(t)$. Since the self-financing trading strategy has no capital influx or consumption, the value of portfolio change can be written as

$$dX = dV + \Delta dS \tag{2.7}$$

Applying the Itô's Lemma to the option price V(t,S)

$$dV = \frac{\partial V}{\partial t}dt + \frac{\partial V}{\partial S}(S,t)dS + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}(S,t)dS^2 \tag{2.8}$$

Since the Black-Scholes model assumes that the stock price under the "market probability measure" follows a gBM.

$$dS = \mu S dt + \sigma S dW \tag{2.9}$$

Putting (1.4) and (1.6) together yields

$$dV = (\frac{\partial V}{\partial t} + \mu S\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2} + \Delta\mu S)dt + (\sigma S\frac{\partial V}{\partial S} + \Delta\sigma S)dW \tag{2.10}$$

The fact that portfolio is risk-free implies that the second term involving the Brownian Motion, $dW$, must be zero. This technique is known as delta-hedging , otherwise, we would have an arbitrage opportunity. Thus, $\Delta = -\frac{\partial V}{\partial S}$. Hence, the growth rate of the portfolio must be the risk free rate which can be summarized as $dX = rXdt$. Substituting $\Delta$ and $dX$ yields

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2} = r(V - S\frac{\partial V}{\partial S}) \tag{2.11}$$

Rearranging the equation to get famous Black-Scholes equation:

$$\frac{\partial V}{\partial t} = rS\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2\frac{\partial^2 V}{\partial S^2} - rV \tag{2.12}$$

$\square$

**Definition 2.1.4.** The resulting partial differential equation can be solved for the analytical solution. The boundary conditions and initial conditions

for the call option is

$$\frac{\partial C}{\partial t} = rS\frac{\partial C}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} - rC \tag{2.13}$$

$$C(0,t) = 0, \ C(S_{\max}, t) = S_{\max} - Ke^{-r(T-t)}, \quad 0 \le t \le T \tag{2.14}$$

$$C(S,T) = \max(S - K, 0), \quad 0 \le S \le S_{\max} \tag{2.15}$$

Solving the equations, the formulae [18] for European call is

$$C = S\Phi(d_1) - Ke^{-r(T-t)}\Phi(d_2) \tag{2.16}$$

$$d_1 = \frac{log(S/K) + (r + \sigma^2/2)(T - t)}{\sigma\sqrt{T - t}} \tag{2.17}$$

$$d_2 = d_1 - \sigma\sqrt{T - t} \tag{2.18}$$

**Remark 2.1.5.** Untradable assets

Modern financial engineering created derivatives using untradable assets as an underlying such as multi asset derivatives like equity baskets, weather derivatives and non-deliverable forwards. Non-deliverable forwards are for offshore investors that want to trade non-convertible currencies such as Brazilian Real, South Korean Won. The Black-Scholes model is still used in these cases [9] but not entirely applicable to assets that cannot be hedged.

## 2.2 Solving Partial Differential Equations

Since the foundation of the world humanity tried to understand and model the nature. Differential equations serves this purpose by enabling us to describe natural phenomena for instance, heat, sound and fluid flow. General

form of 2nd order PDEs of two independent variables is

$$au_{xx} + bu_{xy} + cu_{yy} + du_x + eu_y + fu + g = 0$$

If the equation satisfies the condition $b^2 - 4ac = 0$ it is considered a parabolic partial differential equation. Generally, financial partial differential equations can be classified as parabolic partial differential equations.

Generally, partial differential equations are too complicated to work out an analytical solution. Thus, we need to achieve a numerical solution to the problem. Numerical partial differential equations includes components in the areas of applications, mathematics and computers. The most common framework is finite difference method which tries to find approximate solutions to the problem at a discrete set of points, normally on a rectangular grid of points. Instead we try to find approximate solutions of the problem at a discrete set of points in the (x, t) plane, normally a rectangular grid of points. It is simple to construct and analyse but can compromise performance because of increased computational complexity when there are high dimensions.

Feynman-Kac theorem [10], establishes a link between parabolic partial differential equations and stochastic processes by writing the solution as a conditional expectation. Thanks to the theorem, Monte Carlo method is also utilized to find the numerical solutions to the partial differential equations. Monte Carlo method is preferred when the dimensions are too high !!Glassermann MC book rough calculation of error MC vs PDE grids(1-2 pages)!!

## 2.2.1 Heat Equation

Most basic case for parabolic differential equations is the heat equation. In the experiments the following initial and boundary conditions will be used.

$$u_t(x,t) \quad = \quad u_{xx}(x,t) \tag{2.19}$$

$$u(0,t) \quad = \quad u(x_{max},t) = 0, \quad 0 \le t \le T \tag{2.20}$$

$$u(x,0) \quad = \quad sin(\pi x), \quad 0 \le x \le x_{\max} \tag{2.21}$$

**Definition 2.2.1.** Analytical Solution of Heat Equation

Certain kinds of partial differential equations allows us to find an analytical solution with the help of the Separation of Variables technique [5].

$$u(x,t) = X(x)T(t) \tag{2.22}$$

$$u_{xx}(x,t) = X''(x)T(t) \tag{2.23}$$

$$u_t(x,t) = X(t)T'(t) \tag{2.24}$$

Using the partial derivatives the equation $u_t = u_{xx}$ becomes

$$\frac{T'(t)}{T(t)} = \frac{X''(x)}{X(x)} \tag{2.25}$$

Right hand side only depends on x and the left hand side depends only on t. Therefore, the equation is valid only when each side is equal to a constant, which we set to $\lambda$. Rearranging terms gives us the following equations:

$$\frac{T'(t)}{T(t)} = \frac{X''(x)}{X(x)} = \lambda \tag{2.26}$$

$$X''(x) = \lambda X(x), \quad T'(t) = \lambda T(t) \tag{2.27}$$

$$X(0) = X(1) = 0 \tag{2.28}$$

Solving for X(x) is an example case of Sturm-Liouville problem [11]. However the $\lambda < 0$ and $\lambda = 0$ cases result in trivial solutions, thus they are discarded. Solving for $\lambda > 0$ yields

$$X(x) = c_1 sin(kx) + c_2 cos(kx) \tag{2.29}$$

The boundary conditions leads to $c_2 = 0, c_1 sin(k) = 0 \to k = 0, \pi, 2\pi, ...n\pi$ where n is an integer.

Solving for $T(t)$ gives the solution

$$T(t) = c_3 exp(-n^2\pi^2 t) \tag{2.30}$$

$$u(x,t) = \sum_{n=1}^{\infty} b_n exp(-n^2\pi^2 t) sin(n\pi x) \tag{2.31}$$

where we have set $b_n = c_1 c_3$. The initial condition gives

$$u(x,0) = sin(\pi x) = \sum_{n=1}^{\infty} b_n sin(n\pi x) \tag{2.32}$$

which is a Fourier sine series. Thus, the coefficient $b_n$ is chosen such that

$$b_n = 2 \int_0^1 sin(\pi x) sin(\pi n x) dx = \frac{2 sin(\pi n)}{\pi - \pi n^2} \tag{2.33}$$

Combining the solutions

$$u_n(x,t) = \sum_{n=1}^{\infty} \frac{2sin(\pi n)}{\pi - \pi n^2} exp(-n^2\pi^2 t)sin(n\pi x) = exp(-\pi t)sin(\pi x) \quad (2.34)$$

## 2.2.2   Two Dimensional Heat Equation

2 Dimensional Heat Equation to test ADI method:

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \qquad (2.35)$$

Initial and boundary condition

$$u(x,y,0) = 1, \quad 0 \le x \le x_{\max}, \quad 0 \le y \le y_{\max} \qquad (2.36)$$

$$u(x,0,t) = u(x,y_{max},t) = 0, \quad 0 \le t \le T \qquad (2.37)$$

$$u(x,0,t) = u(x,1,t) = 0, \quad 0 \le t \le T \qquad (2.38)$$

**Definition 2.2.2.** Analytical Solution of Two Dimensional Heat Equation
Similarly, applying separation of variables method to the equation

$$u(x,t) = X(x)Y(y)T(t) \qquad (2.39)$$

$$X^{''}(x) - BX(x) = 0 \qquad (2.40)$$

$$Y^{''}(y) - C(y) = 0 \qquad (2.41)$$

$$T^{'}(t) - (B+C)T(t) = 0 \qquad (2.42)$$

$$X(0) = X(1) = 0, \quad Y(0) = Y(1) = 0 \qquad (2.43)$$

We have solved for X(x) and Y(y) in Analytical Solution of Heat Equa-

tion:

$$X_n(x) = sin(n\pi x), \quad B = n\pi Y_m(x) = sin(m\pi y), \quad C = m\pi \qquad (2.44)$$

Solving for $T(t)$ gives the solution

$$T(t) = exp(-\pi\sqrt{m^2 + n^2}) \qquad (2.45)$$

$$u(x, y, t) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} A_{mn} sin(n\pi x) sin(m\pi y) exp(-\pi\sqrt{m^2 + n^2}) \qquad (2.46)$$

The initial condition gives

$$u(x, y, 0) = 1 = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} A_{mn} sin(n\pi x) sin(m\pi y) \qquad (2.47)$$

which is a double Fourier sine series. Thus, the coefficient $A_{mn}$ is chosen such that

$$A_{mn} = 4\int_0^1\int_0^1 sin(\pi n x) sin(\pi n y) dx dy = \frac{4(cos(\pi n) - 1)(cos(\pi m) - 1)}{\pi^2 mn}$$
$$(2.48)$$

Combining the solutions

$$u_{mn}(x, y, t) = \sum_{m=1}^{\infty}\sum_{n=1}^{\infty} \frac{4(cos(\pi n) - 1)(cos(\pi m) - 1)}{\pi^2 mn}$$

$$sin(n\pi x) sin(m\pi y) exp(-\pi\sqrt{m^2 + n^2}) = 0 \quad (2.49)$$

## 2.3    Finite Difference Methods

### 2.3.1    Discretization

First, we need to discretize the PDE for numerically solving parabolic PDEs such as heat equation using FDMs, the space dimension(s) and the time dimension are handled differently.
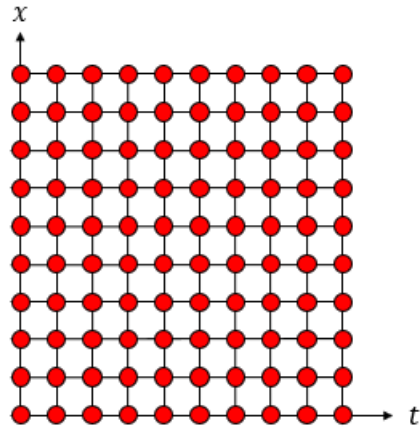
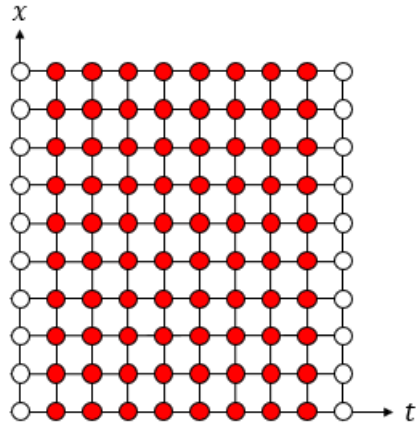

Figure 2.1: 10 x 10 discretization.



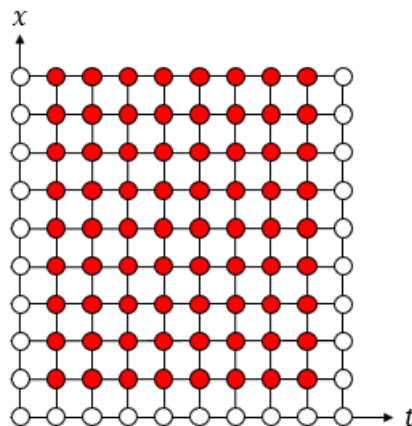Figure 2.2: Boundary conditions.

Figure 2.3: Initial condition.

## 2.3.2 Explicit Method

Explicit method is a forward time, central space scheme . 1st order Forward Difference

$$\frac{\partial u}{\partial t} = \frac{u_i^{n+1} - u_i^n}{\Delta t}$$

2nd order Central Difference

$$\frac{\partial^2 u}{\partial x^2} = \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n}{(\Delta x)^2}$$

The parabolic partial differential equation can be generalized by applying the forward difference to the time derivative and the centred second difference.

$$u_j^{n+1} = \alpha u_{j-1}^n + \beta u_j^n + \gamma u_{j+1}^n$$

For heat equation the coefficients become

$$\alpha = r, \beta = 1 - 2r, \gamma = r$$

Black-Scholes partial differential equation the coefficients become:

$$\alpha = \frac{\sigma^2 j^2 \Delta t}{2} - \frac{rj\Delta t}{2}, \beta = 1 - \sigma^2 j^2 \Delta t - r\Delta t, \gamma = \frac{\sigma^2 j^2 \Delta t}{2} - \frac{rj\Delta t}{2}$$
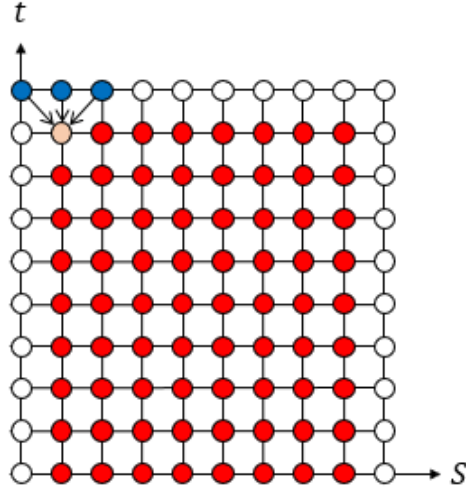
where $r = \frac{\delta t}{\delta x^2}$.



Figure 2.4: Computational stencil of explicit method

## 2.3.3 Crank-Nicholson Method

The crank nicolson method was introduced [3]

$$u(t,x) \approx \frac{1}{2}(u_i^{n+1} + u_i^n)$$

$$\frac{\partial u}{\partial t} \approx \frac{u_i^{n+1} - u_i^n}{\Delta t}$$

$$\frac{\partial u}{\partial x} \approx \frac{u_{i+1}^n - u_{i-1}^n + u_{i+1}^{n+1} - u_{i-1}^{n+1}}{4\Delta x}$$

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{u_{i+1}^n - 2u_i^n + u_{i-1}^n + u_{i+1}^{n+1} - 2u_i^{n+1} + u_{i-1}^{n+1}}{2(\Delta x)^2}$$

$$A = a(t,x)\frac{\Delta t}{\Delta x^2}, B = b(t,x)\frac{\Delta t}{4\Delta x}, C = c(t,x)\frac{\Delta t}{2}, D = d(t,x)\Delta t$$

$$
\begin{aligned}
(-A - B)u_{i+1}^{n+1} + (1 + 2A - C)u_i^{n+1} + (-A + B)u_{i-1}^{n+1} = \\
= (A + B)u_{i+1}^n + (1 - 2A + C)u_i^n + (A - B)u_{i-1}^n + D \quad (2.50)
\end{aligned}
$$

The left hand side groups the unknowns and the right hand side groups knowns. The system of equations can be represented by a tridiagonal matrix system. Most commonly these system are solved by Thomas Algorithm.

### 2.3.4 Rannacher Trick

Using the Crank nicholson

### 2.3.5 Alternating Direction Implicit Method

The alternate direction implicit (ADI) method is used to numerically solve two dimensional parabolic PDEs. ADI schemes give us advantages of implicit finite difference method and computationally only requires to solve tridiagonal matrices.

ADI Peaceman and Rachford in 1955 [14]

Operator splitting is a natural and old idea. When a PDE or system of PDEs contains different terms expressing different physics, it is natural to use different numerical methods for different physical processes. This can optimize and simplify the overall solution process. The idea was especially popularized in the context of the Navier-Stokes equations and reaction-diffusion

PDEs. Common names for the technique are operator splitting, fractional step methods, and split-step methods. We shall stick to the former name. In the context of nonlinear differential equations, operator splitting can be used to isolate nonlinear terms and simplify the solution methods. [4] A related technique, often known as dimensional splitting or alternating direction implicit (ADI) methods, is to split the spatial dimensions and solve a 2D or 3D problem as two or three consecutive 1D problems, but this type of splitting is not to be further considered here. [17].

$$\delta x^2 u_{i,j}^n = \frac{u_{i+1,j}^n - 2u_{i,j}^n + u_{i-1,j}^n}{\Delta x^2} \tag{2.51}$$

$$\frac{u_{i,j}^{n+1} + u_{i,j}^n}{\Delta t} = \delta x^2 u_{i,j}^n + \delta y^2 u_{i,j}^n \tag{2.52}$$

$$\frac{u_{i,j}^{n+1} + u_{i,j}^n}{\Delta t} = \delta x^2 u_{i,j}^{n+1} + \delta y^2 u_{i,j}^{n+1} \tag{2.53}$$

Divide each time step in half Implicit x, Explicit y:

$$\frac{u_{i,j}^{n+1/2} + u_{i,j}^n}{0.5\Delta t} = \frac{\delta x^2 u_{i,j}^{n+1/2}}{\Delta x^2} + \frac{\delta y^2 u_{i,j}^n}{\Delta y^2} \tag{2.54}$$

Explicit x, Implicit y:

$$\frac{u_{i,j}^{n+1} + u_{i,j}^{n+1/2}}{0.5\Delta t} = \frac{\delta x^2 u_{i,j}^{n+1/2}}{\Delta x^2} + \frac{\delta y^2 u_{i,j}^{n+1}}{\Delta y^2} \tag{2.55}$$
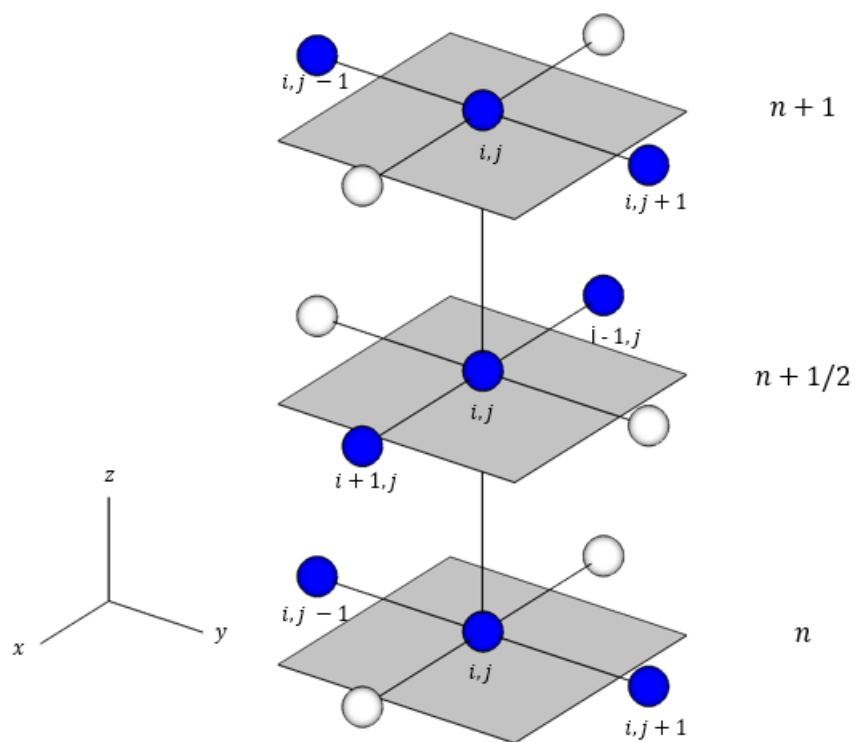
Figure 2.5: Computational stencil of alternating direction implicit method

# Chapter 3

# Optimizing Solvers

Numerical analysis and computer simulations will be undertaken to put theory and observation together to gain insight into the workings of numerical solutions of partial differential equations. First step was deriving toy examples that can be calculated by hand and Excel. Following the verifications, porting the toy examples in C++ and utilize high performance computing principles.

This section documents the performance of attempted optimizations. Experiments are conducted at W307 computer laboratory, Queen Mary University of London. Each computer has Windows 10 Enterprise 64 bit, 16 GB of RAM, Intel Core i7-6700 CPU with 4 cores clocked at 3.40 GHz. The source code is written in C++ and compiled with Microsoft Visual Studio Enterprise 2017, Version 15.3.3 in the release mode. External tools utilized in the tests include Intel Compiler, version 18.0.3 and Intel Math Kernel Library, version X .

## 3.1   Timing the Code

Measuring execution time intervals accurately is an important aspect to compare the efficiency and speed of different environments and implementations.

### 3.1.1 Windows Application Programming Interface

Windows Application Programming Interface (API) is the lowest level of interaction between applications and the windows operating system. Thus every program is built upon the API. Mostly, the interaction is hidden, the runtime and support libraries manage it in the background [15]. The APIs can be used in the C++ environment. Runtime can be calculated by "QueryPerformanceCounter" or "QueryPerformanceFrequency" functions. Respectively, the functions retrieve a high resolution time stamp and the frequency of the performance counter.

### 3.1.2 Chrono Library

Using the Windows API for just timing the code is slightly excessive given the amount of work it takes. Luckily, Chrono library was introduced part of the C++11's standard library. Timers and clocks might differ on distinct systems, thus Chrono library is designed to work effortlessly with date and time. The "high resolution clock" provides the smallest possible tick period and with the "now" method, returns a value corresponding to the call's point in time. Once the start and end time of the code is recorded , the duration::count method is used to get the elapsed time.

## 3.2 Optimization Experiments

Main optimization techniques that can be implemented can be listed as follows Intel Compiler, VS Compiler, gcc

### 3.2.1 Tridiagonal Solvers

Implementing Crank Nicolson and Alternating Direction Implicit methods requires to solve tridiagonal systems which is the most computationally intensive part of the program. Therefore, choosing efficient tridiagonal solvers

is crucial for the speed of the solver. In this experiment three different algorithms will be tested.

**Thomas Algorithm**

Thomas Algorithm is the most commonly used method for solving tridiagonal system of equations. The method is used to solve a tridiagonal matrix system invented by Llewellyn Thomas [17]. The system equations can be written as

$$
\begin{bmatrix}
b_1 & c_1 & 0 & 0 & ... & 0 \\
a_2 & b_2 & c_2 & 0 & ... & 0 \\
0 & a_3 & b_3 & c_3 & 0 & 0 \\
. & . & & & & . \\
. & . & & & & . \\
. & . & & & & c_{k-1} \\
0 & 0 & 0 & 0 & a_k & b_k
\end{bmatrix}
\begin{bmatrix}
f_1 \\ f_2 \\ f_3 \\ . \\ . \\ . \\ f_k
\end{bmatrix}
=
\begin{bmatrix}
d_1 \\ d_2 \\ d_3 \\ . \\ . \\ . \\ d_k
\end{bmatrix}
$$

The method begins by forming coefficients $c_i^*$ and $d_i^*$ in place of $a_i$, $b_i$ and $c_i$ as follows:

$$
c_i^* =
\begin{cases}
\frac{c_1}{b_1} & ; i = 1 \\
\frac{c_i}{b_i - c_{i-1}^* a_i} & ; i = 2, 3, ..., k - 1
\end{cases}
$$

$$
d_i^* =
\begin{cases}
\frac{d_1}{b_1} & ; i = 1 \\
\frac{d_i - d_{i-1}^* a_i}{b_i - c_{i-1}^* a_i} & ; i = 2, 3, ..., k - 1
\end{cases}
$$

With these new coefficients, the matrix equation can be rewritten as:

$$
\begin{bmatrix}
1 & c_1^* & 0 & 0 & ... & 0 \\
0 & 1 & c_2^* & 0 & ... & 0 \\
0 & 0 & 1 & c_3^* & 0 & 0 \\
. & . & & & & . \\
. & . & & & & . \\
. & . & & & & c_{k-1}^* \\
0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
f_1 \\ f_2 \\ f_3 \\ . \\ . \\ . \\ f_k
\end{bmatrix}
=
\begin{bmatrix}
d_1^* \\ d_2^* \\ d_3^* \\ . \\ . \\ . \\ d_k^*
\end{bmatrix}
$$

The algorithm for the solution of these equations is now straightforward and works 'in reverse':

$$
f_k = d_k^*, \qquad f_i = d_k^* - c_i^* x_{i+1}, \qquad i = k-1, k-2, ..., 2, 1
$$

### Intel Math Kernel Library

Intel Math Kernel Library implements routines for solving systems of linear equations from the standard LAPACK library. Variety of matrix types are supported by the routines. Specifically gtsv function is utilized from the package. Using Gaussian elimination with partial pivoting, gstv computes the solution to the system of linear equations with a tridiagonal coefficient matrix [8].

### Cyclic Reduction

Cyclic reduction was proposed by R. W. Hockney in the 1960s for solving the resulting linear systems from the discretization of the Poisson equation [7].

### 3.2.2   32 bit and 64 bit

In practical terms, you would use X64 when: You need to directly address more than 4GB of memory, or You need very fast (native) processing of 64 bit numerical quantities (including double-precision floating-point numbers) X86 (32 bit) is suitable for most everything else.

## 3.2.3   Visual Studio Optimization Switches

## 3.2.4   Optimal Grid Size

The resulting analytical solutions are utilised to calculate errors for the solutions using different grid sizes, the grid with the lowest error is used for further tests. Figure X and X demonstrates the error compared to the grid sizes. The optimal grid size for heat equation and Black-Scholes equation is 50 by 50. why uniform olunca daha iyi oluyor araştır??
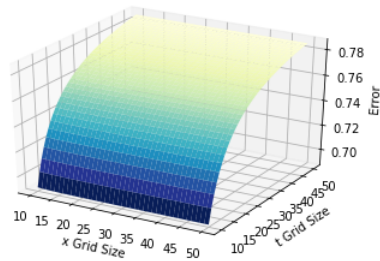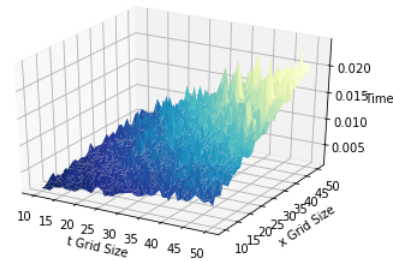


Figure 3.1: Error of grid sizes.          Figure 3.2: Timing grid sizes.

After choosing the optimal grid size the resulting grids can be visualized as

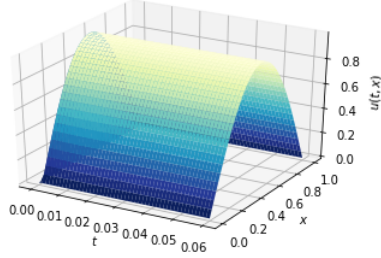After choosing the optimal grid size the resulting grids can be visualized as
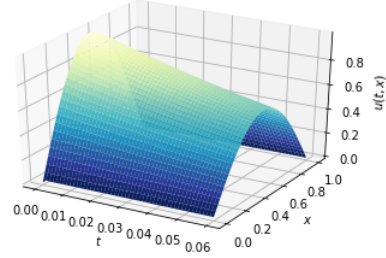
Figure 3.3: Explicit method.        Figure 3.4: Crank nicolson method.

| Parameter | Value |
|---|---|
| Strike Price | 1.0 |
| Volatility | 20 % |
| Risk Free Rate | 5 % |
| Time to Expiry | 2.0 |
| Maximum Share Price | 2.0 |

Table 3.1: Black-Scholes model testing parameters.

| Parameter | Value |
|---|---|
| $T$ | 0.075 |
| $x_{max}$ | 1.0 |

Table 3.2: One dimensional heat equation testing parameters.

| Parameter | Value |
|---|---|
| $T$ | 0.075 |
| $x_{max}$ | 1.0 |
| $y_{max}$ | 1.0 |

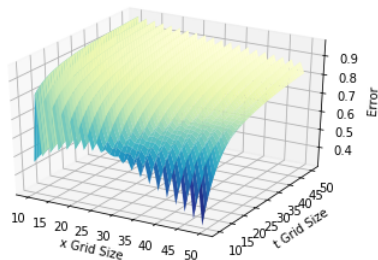Table 3.3: Two dimensional heat equation testing parameters.
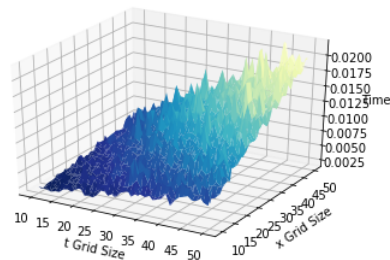
Figure 3.5: Error of grid sizes.
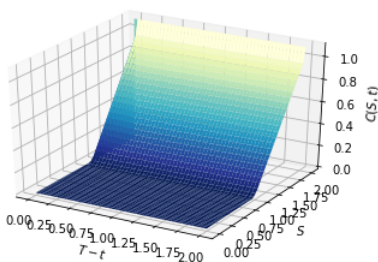


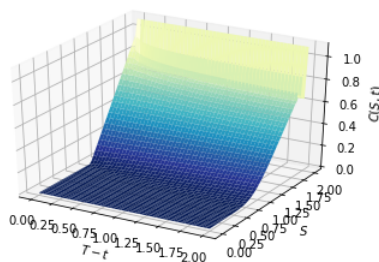Figure 3.6: Timing grid sizes.



Figure 3.7: Explicit method.



Figure 3.8: Crank nicolson method.

## 3.3 Results of the Numerical Experiments

Timing the tridiagonal systems in a vanilla setup. Timing the different compilers and Solution Platforms Timing 32 bit vs 64 bit + VS vs Intel compiler Timing changing the optimization switches with the most optimal one.

| Environment | Explicit Method | Crank Nicolson |
|---|---|---|
| Visual Studio Compiler x86 | 0.022081 | 0.02035 |
| Visual Studio Compiler x64 | 0.018627 | 0.01888 |
| Intel Compiler x86 | 0.023165 | 0.023723 |
| Intel Compiler x64 | 0.019535 | 0.017451 |

Table 3.4: Averaging 1000 timings for Black-Scholes.

| Testing | Explicit Method Timing | Crank Nicolson Timing |
|---|---|---|
| Visual Studio Compiler x86 | 0.022625 | 0.022441 |
| Visual Studio Compiler x64 | 0.018933 | 0.016687 |
| Intel Compiler x86 | 0.023076 | 0.023719 |
| Intel Compiler x64 | 0.017271 | 0.017271 |

Table 3.5: Averaging 1000 timings for Black-Scholes and adding random $\epsilon < 10^{-7}$ to step size in order to avoid compiler optimizations.

# Chapter 4

# Conclusions

## 4.1  Further Work

aaaaa [13], cloud functions [6].

Threading High performance computing techniques that can be implemented for multithreading with Open Multi-Processing(OpenMP) and compiler intrinsics.

AVX and Intrinsics CPUs are pipelining and use of SSE/SIMD kusswurm registers with Advanced Vector Extensions(AVX 512),

GPGPU In the case of General Purpose GPUs, CUDA or Open Computing Language(OpenCL) can be utilized but can be challenging because of the requirement of delicate memory management.

# Appendix A

# Usage of chrono class

Should code example be in appendix or stay here?

```
auto start = std::chrono::high_resolution_clock::now();
    Portion of code to be timed
auto finish = std::chrono::high_resolution_clock::now();
std::chrono::duration<double> elapsed = finish - start;
std::cout << "Elapsed time: " << elapsed.count() << " s\n";
```

# Appendix B

# Implementation of the `FiniteDifferenceMethod` **class**

Parabolic partial differential equation can be denoted as

$$\frac{\partial u}{\partial t} = a(t,x)\frac{\partial^2 u}{\partial x^2} + b(t,x)\frac{\partial u}{\partial x} + c(t,x)u(t,x) + d(t,x)$$

$a(t,x)$ denotes diffusion coefficient, $b(t,x)$ convection coefficient, $c(t,x)$ reaction coefficient, $d(t,x)$ source coefficient

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

# Appendix C

# Additional details on the Gundermanian determinant

Suspendisse vitae elit. Aliquam arcu neque, ornare in, ullamcorper quis, commodo eu, libero. Fusce sagittis erat at erat tristique mollis. Maecenas sapien libero, molestie et, lobortis in, sodales eget, dui. Morbi ultrices rutrum lorem. Nam elementum ullamcorper leo. Morbi dui. Aliquam sagittis. Nunc placerat. Pellentesque tristique sodales est. Maecenas imperdiet lacinia velit. Cras non urna. Morbi eros pede, suscipit ac, varius vel, egestas non, eros. Praesent malesuada, diam id pretium elementum, eros sem dictum tortor, vel consectetuer odio sem sed wisi.

# Bibliography

[1] Saeed Amen. *Trading Thalesians: What the Ancient World Can Teach Us About Trading Today*, pages 39–60. Palgrave Macmillan UK, London, 2014.

[2] Fischer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(3):637–654, 1973.

[3] J. Crank and P. Nicolson. A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type. *Advances in Computational Mathematics*, 6(1):207–226, Dec 1996.

[4] Jim Douglas, Jr. Alternating direction methods for three space variables. *Numer. Math.*, 4(1):41–63, December 1962.

[5] D.J. Duffy. *Finite Difference Methods in Financial Engineering: A Partial Differential Equation Approach*. The Wiley Finance Series. Wiley, 2006.

[6] Sadjad Fouladi, Francisco Romero, Dan Iter, Qian Li, Shuvo Chatterjee, Christos Kozyrakis, Matei Zaharia, and Keith Winstein. From laptop to lambda: Outsourcing everyday jobs to thousands of transient functional containers. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 475–488, Renton, WA, July 2019. USENIX Association.

[7] R. W. Hockney. A fast direct solution of poisson's equation using fourier analysis. *J. ACM*, 12(1):95–113, January 1965.

[8] Intel. Intel math kernel library for c, gtsv. https://software.intel.com/en-us/mkl-developer-reference-c-gtsv, 2019. [Online; accessed 7-August-2019].

[9] Stephen Jewson and Mihail Zervos. The black-scholes equation for weather derivatives. *SSRN Electronic Journal*, 09 2003.

[10] F.C. Klebaner. *Introduction to Stochastic Calculus with Applications*. Introduction to Stochastic Calculus with Applications. Imperial College Press, 2005.

[11] Matthew J. Hancock. The 1-d heat equation (18.303, linear partial differential equations). https://ocw.mit.edu/courses/mathematics/18-303-linear-partial-differential-equations-fall-2006/lecture-notes/heateqni.pdf, 2006. [Online; accessed 7-August-2019].

[12] Robert C. Merton. Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3(1):125 – 144, 1976.

[13] Samuel Palmer. Accelerating implicit finite difference schemes using a hardware optimised implementation of the thomas algorithm for fpgas. *arXiv preprint arXiv: 1402.5094, 2014*, 2014.

[14] D. W. Peaceman and H. H. Rachford. The numerical solution of parabolic and elliptic differential equations. *Journal of the Society for Industrial and Applied Mathematics*, 3(1):28–41, 1955.

[15] Charles Petzold. *Programming Windows, Fifth Edition*. Microsoft Press, Redmond, WA, USA, 5th edition, 1998.

[16] Sriram Ramaswamy. Pollen grains, random walks and einstein. *Resonance*, 5, 03 2000.

[17] Llewellyn Hilleth Thomas. Elliptic problems in linear difference equations over a network. *Watson Sci. Comput. Lab. Rept., Columbia University, New York*, 1, 1949.

[18] Paul Wilmott. *Paul Wilmott Introduces Quantitative Finance*. Wiley-Interscience, New York, NY, USA, 2 edition, 2007.