Master

Mathematical Finance

Master's Final Work

Dissertation

The Use of Radial Basis Functions in the Numerical Solution of Option Pricing Problems

Beatriz Malheiros Leal

August 2018

Master

Mathematical Finance

Master's Final Work

Dissertation

The Use of Radial Basis Functions in the Numerical Solution of Option Pricing Problems

Beatriz Malheiros Leal

Supervisor:
João Paulo Vicente Janela

August 2018

Resumo

Esta dissertação tem como objetivo a implementação de uma abordagem numérica moderna à solução da equação diferencial parcial de Black-Scholes, usada para calcular o preço de opções financeiras usando um método sem grelha baseado em funções de base radial. Estas equações são normalmente resolvidas através de métodos numéricos tradicionais tal como o método das diferenças finitas, elementos finitos ou volumes finitos. Mais ainda, o cálculo do preço de opções pode ser feito através de modelos binomiais e/ou simulação de Monte Carlo.

A interpolação de pontos utilizando funções de base radial (RBPI) é muito útil quando o número de derivados financeiros é elevado, por exemplo no caso das "basket options" cujo lucro depende do valor de um conjunto de derivados (portefólio). Este método permite concentrar os graus de liberdade do problema nas regiões do domínio mais relevantes, distribuindo os esforços computacionais.

Esta dissertação apresenta a implementação do RBPI em vários problemas teste e uma análise de convergência dos problemas relativamente à sua solução exata bem como os respectivos tempos computacionais.

Foi possível concluir que o método é válido e que os resultados obtidos são consistentes. No futuro, consideraremos problemas de maior dimensão bem como outras implementações deste método.

Palavras-Chave: Valorização de Opções, Funções de Base Radial, Equação de Black-Scholes, Métodos Numéricos.

Abstract

This dissertation aims at implementing a modern numerical approach to the solution of the Black-Scholes partial differential equation, used for pricing financial options, by using a mesh-free method based on radial basis functions. These equations are normally solved by standard numerical methods like finite differences, finite elements or finite volumes. Additionally, option pricing can also be performed using binomial models and/or Monte Carlo Simulation.

Radial Basis Point Interpolation (RBPI) is very useful when the number of derivatives is high, for example in basket options where the pay-off depends on the value of a portfolio (or basket) of derivatives. This method allows to concentrate the degrees of freedom on the most relevant regions in the domain, distributing the computational efforts.

This dissertation presents the implementation of the radial basis point interpolation in several test problems and an analysis of the convergence of the discrete problems to the exact solutions, including computational times.

We conclude that the method is valid and the obtained results are consistent. In the future, we will consider problems in higher dimensions as well as parallel implementations of this method.

Keywords: Option Pricing, Radial Basis Functions, Black-Scholes equation, Numerical Methods.

Acknowledgements

I would like to thank my supervisor João Janela for guiding me in this dissertation, giving me the needed space to develop it on my own and for supporting me in the moments when I doubted myself. I also owe gratitude to professor Fernando Gonçalves for the opportunities and knowledge he has been providing me, and for helping me trace my academical path lately.

To my colleagues who have been accompanying and supporting this stage of my life - specially to Alina Teles, with whom I have learned so much - thank you for all the steps we took together. To my friends, thank you for the encouragement and moments of leisure that contributed to the strength I needed for everything I've accomplished.

Last but not least, I'd like to thank my parents and brother for their love and for supporting me in every decision I've been taking since I was a child, and to my boyfriend for all the comprehension and encouragement.

Contents

Resumo iii

Abstract iv

Acknowledgements v

1 Introduction 1

2 Option Pricing 3

2.1 Black Scholes Equation 4

2.1.1 European Put Option Pricing 7

2.2 Black Scholes Formula 9

2.3 Other pricing models 10

3 Numerical Methods 12

3.1 Finite Difference Methods 13

3.1.1 Euler Implicit Scheme 15

3.2 Interpolation Methods 16

3.2.1 Polynomial Interpolation 16

3.2.2 Radial Basis Point Interpolation Method 17

3.2.3 RBPI implementation in the Black Scholes model for European put options 20

4 Numerical Results 23

4.1 Interpolation: Polynomial vs. Radial Based 23

4.2 Black-Scholes PDE discretization with RBPI 26

4.2.1 The finite difference method vs. The RBPI method 30
4.3 Convergence 30
4.4 Introduction of a source term in the IBVP 31

5 Conclusion **35**

Bibliography **37**

A MATLAB Code **38**
A.1 Radial Basis Point Interpolation 38
A.2 RBPI in the Black Scholes Model 39
A.3 Finite Difference Method 42
A.4 RBPI with the introduction of a source term 42
A.5 Black Scholes Formula 45
A.6 Auxiliar functions 46

List of Figures

1 Graphic of the function to be approximated 23
2 Radial Basis Point Interpolation 24
3 Polynomial Interpolation 24
4 Value of the put option and payoff function 27
5 Position of the centers s_i 28
6 Value of the put option varying in time and space 28
7 Exact and Numerical Solutions 33

List of Tables

1 Errors associated to the different interpolations 25
2 Error associated to each radial basis function 26
3 Errors associated to different space divisions 29
4 FD method and RBPI method errors 30
5 Convergence order evolution 31
6 Errors associated to the experiment with a source term 34
7 Convergence order evolution 34

Chapter 1

Introduction

The idea of what an option is comes from the 14th century, when insurance companies would make promises of buying ships or cargoes when they'd fail to arrive in order to provide against that danger [12], but its main growth has happened since 1950.

Everyday, all around the world, options on securities are exchanged. An option is a type of derivative, which is a contract between two parties: the writer and the holder. Call options give the holder the right (but not the obligation) to buy an underlying asset S by a certain date called maturity T for a certain price called the strike price K. Put options give the holder the right to sell the underlying asset, on the same conditions as the Call Option. The most used options are called European and American and the difference between them is that the first can only be exercised at maturity whereas the second can be exercised at any moment from the establishment of the contract until the maturity date [11].

Fisher Black and Myron Scholes were two economists that in the year of 1973 published a paper [3] where they presented a formula which valuates an option in terms of the price of the underlying stock. This formula is very well known and still used nowadays. For more general option contracts there are no closed form solutions and the valuation must be made solving certain partial differential equations with the aid of numerical methods.

For instance, finite difference methods allow to approximate the solution of partial differential equations numerically by transforming them in a set of difference

equations, which are solvable by iteration [11]. The advantage of this numerical method, and others, is that their solutions can be computationally derived [10].

The finite difference method has some limitations, namely the usage of a constant space-time grid. To approximate the solution of partial differential equations using mesh-free methods (without the need of a constant space-time grid), usually it is discretized in nodes by collocation methods [14].

When the data is scattered, radial basis point interpolation methods are a good approach since they allow to interpolate the solution of the PDE between the data sites as a linear combination of radial basis functions and monomials. A scheme with finite differences is used to approximate the derivatives in time but in space the derivatives are a linear combination of the Radial Basis Functions and Polynomials used.

In the following chapters you may find a theoretical introduction on Option Pricing, focused on the Black Scholes Model, followed by the presentation of the numerical methods that are used afterwards. The numerical results present comparisons between the method in study and others, with a study of convergence to the exact solutions. Finally, some conclusions are drawn. In the appendices are available the MATLAB programs used to obtain the presented results.

Chapter 2

Option Pricing

Derivatives play an increasingly important role in finance all over the world. This way, it is very important to understand how they work, how can they be used and, most importantly, how to price them. Hull [11] defines a derivative as "a financial instrument whose value depends on (or derives from) the values of other, more basic, underlying variables" such as prices of traded assets. For example, the price of a stock option depends heavily on the price of a stock, but there are many factors (some of them not so evident) influencing the price of a derivative.

An option is a contract between two parties: the holder and the writer. There are two types of options:

- Call Option - gives the holder the right to buy the underlying asset at a specific date for a specific price
- Put Option - gives the holder the right to sell the underlying asset at a specific date for a specific price

Note that the holder has the right to (buy or sell) but not the obligation to do it, which means that he may decide not to exercise his right. The factors that affect the price of the option are the price of the stock S, the strike price K which is the amount that the holder must pay in order to exercise his option, the maturity of the contract T which is the date where the contract expires, the volatility of the market σ and the risk free interest rate r.

The decision of the holder to exercise or not the option, obviously, depends on its profit. The interpretation of the meaning of the strike price depends on the nature of the option. If the option is a call, the strike price is the price at which the option holder can purchase the underlying asset, therefore in this case he will exercise if the stock price is greater than the strike price, earning a profit equal to $S - K$. If the option is a put, the strike price is the price at which the option holder can sell the underlying asset, then he will exercise if the stock price is lower than the strike price, earning a profit equal to $K - S$.

The timing of this decision depends on whether the option is European or American. In the first case, the holder is allowed to exercise only at the maturity date but in the second, he may exercise whenever he pleases, from the date when the contract is established until the maturity date. There are other types of options, but they will not be object of study in this dissertation.

2.1 Black Scholes Equation

The Black Scholes model published in 1973 [3] has been explored by several authors. In the introduction of his famous book in Arbitrage Theory [2], Björk covers the basic Black Scholes theory by following the arguments of Merton. We will now highlight the most relevant steps, starting with the assumptions of the model:

1. The price of a stock follows a Geometric Brownian Motion
2. There is a continuous trading
3. The market is efficient (arbitrage free) and frictionless (there are no transaction costs)
4. Short positions are allowed
5. The volatility σ is constant
6. The yield curve is flat

7. The portfolio is locally riskless

8. The derivative which integrates the portfolio can be bought and sold on a market

Let there be a market with a price process B that refers to the price of a risk free asset with the dynamics

$$dB_t = rB_t dt, \tag{2.1}$$

where r is the short interest rate and the stock price S given by

$$dS_t = \alpha S_t dt + \sigma_t S_t dW_t, \tag{2.2}$$

where α is the mean rate of return of S, σ is the (stochastic) volatility of S and W is a Wiener process.

Consider a financial derivative X with maturity T of the form $X = \Phi(S(T))$ and price process $\Pi(t, X)$. If the case is of an European put option, then $\Phi(x) = \max(K - x, 0)$. The value of the option at maturity is quite simple to understand:

- if the price of the derivative is lower than the strike price, the holder has interest in exercising the option since he will be selling the derivative to the writer at a higher price than in the market and gain a profit equal to $K - S(T)$;

- if the opposite situation occurs, then it wouldn't make sense to exercise, since the holder could sell the underlying asset at a higher price in the market, and therefore the profit of this contract is 0.

This way, we can derive $\Pi(T) = \max(S(T) - K, 0)$.

Now that we know the behavior of the price of the option at maturity, we must derive its price for all time $t \in [0, T[$.

Suppose there is a self financing portfolio (i.e. a collection of assets where the purchase of determined assets is financed through the sale of others) with value V such that

$$dV_t = kV_t dt. \tag{2.3}$$

If the market is free of arbitrage, i.e. there are no opportunities of taking advantage from a difference of prices, then we must have $k = r$ (remember the price process of B from equation (2.1).

Assume that the price of the derivative has the form $\Pi_t(X) = F(t, S_t)$, i.e. is a function depending on time and on the stock price, then by using Itô's formula we may obtain the dynamics of Π_t in terms of the unknown function F:

$$\begin{aligned} d\Pi_t &= dF(t, S_t) \\ &= \frac{\partial F}{\partial t}dt + \frac{\partial F}{\partial S}dS + \frac{1}{2}\frac{\partial^2 F}{\partial S^2}dS^2 \\ &= \left(\frac{\partial F}{\partial t} + \alpha S\frac{\partial F}{\partial S} + \frac{\sigma^2}{2}S^2\frac{\partial^2 F}{\partial S^2}\right)dt + \sigma S\frac{\partial F}{\partial S}dW. \end{aligned} \tag{2.4}$$

Simplifying, we get

$$d\Pi_t = \alpha_\pi \Pi dt + \sigma_\pi \Pi dW \tag{2.5}$$

with

$$\alpha_\pi = \frac{\frac{\partial F}{\partial t} + \alpha S\frac{\partial F}{\partial S} + \frac{\sigma^2}{2}S^2\frac{\partial^2 F}{\partial S^2}}{F} \quad \text{and} \quad \sigma_\pi = \frac{\sigma S\frac{\partial F}{\partial S}}{F}.$$

Let there be a self financing portfolio with value V based on the underlying stock and the derivative asset where u_s is the proportion of the investment in the stock and u_π is the proportion of the investment in the derivative, such that $u_s + u_\pi = 1$. Then, the dynamics of the portfolio is given by

$$\begin{aligned} dV &= V\left\{u_s\frac{dS}{S} + u_\pi\frac{d\Pi}{\Pi}\right\} \\ &= V\left\{u_s(\alpha dt + \sigma dW) + u_\pi(\alpha_\pi dt + \sigma_\pi dW)\right\} \\ &= V\left\{(\alpha u_s + \alpha_\pi u_\pi)dt + (\sigma u_s + \sigma_\pi u_\pi)dW\right\}. \end{aligned} \tag{2.6}$$

We must choose a proportion of investment such that the Brownian term dW disappears in order to ensure a locally riskless portfolio. This way the condition $\sigma u_s + \sigma_\pi u_\pi = 0$ must hold. Adding the obvious condition $u_s + u_\pi = 1$ then we have enough to determine the proportions u_s and u_π.

$$\begin{cases} \sigma u_s + \sigma_\pi u_\pi = 0 \\ u_s + u_\pi = 1 \end{cases} \Leftrightarrow \begin{cases} u_s = \dfrac{\sigma_\pi}{\sigma_\pi - \sigma} \\ u_\pi = -\dfrac{\sigma}{\sigma_\pi - \sigma} \end{cases} \tag{2.7}$$

Substituting the expression of σ_π from (2.5), then

$$u_s = \frac{S\frac{\partial F}{\partial S}}{S\frac{\partial F}{\partial S} - F} \text{ and } u_\pi = -\frac{F}{S\frac{\partial F}{\partial S} - F}. \tag{2.8}$$

Recalling the arbitrage condition, then we must also have $\alpha u_s + \alpha_\pi u_\pi = r$. Now, substituting the expression of α_π from (2.5) and the proportions from (2.8) in the arbitrage condition, we have

$$\begin{aligned} \alpha\frac{S\frac{\partial F}{\partial S}}{S\frac{\partial F}{\partial S} - F} + \frac{\frac{\partial F}{\partial t} + \alpha S\frac{\partial F}{\partial S} + \frac{\sigma^2}{2}S^2\frac{\partial^2 F}{\partial S^2}}{F}\left(-\frac{F}{S\frac{\partial F}{\partial S} - F}\right) = r \\ \Leftrightarrow \frac{\partial F}{\partial t} + rS\frac{\partial F}{\partial S} + \frac{\sigma^2}{2}S^2\frac{\partial^2 F}{\partial S^2} - rF = 0. \end{aligned} \tag{2.9}$$

Since at maturity $\Pi(T) = \Phi(S(T)) = F(T, S)$ then, finally, we have that S can take any value and F has to satisfy the PDE

$$\begin{cases} \frac{\partial}{\partial t}F(t,S) + rS\frac{\partial}{\partial S}F(t,S) + \frac{\sigma^2}{2}S^2\frac{\partial^2}{\partial S^2}F(t,S) - rF(t,S) = 0 \\ F(T,S) = \Phi(S) \end{cases} \tag{2.10}$$

This result (2.10) is the so-called Black-Scholes equation that allows to price the derivative in a relative way, i.e. as a function of the price of the underlying asset. This means that the underlying price process(es) must be known.

2.1.1 European Put Option Pricing

We will now consider the specific case of an European Put Option. The price of a put option with strike price K and maturity T on an underlying asset S is given by $V(S,t)$ where the Black-Scholes operator is given by

$$\mathcal{L}V(S,t) = \frac{\partial}{\partial t}V(S,t) + \frac{\sigma^2}{2}S^2\frac{\partial^2}{\partial S^2}V(S,t) + rS\frac{\partial}{\partial S}V(S,t) - rV(S,t) \tag{2.11}$$

The problem is written, for $S \in]0, +\infty[$ and $t \in [0, T[$:

$$\begin{cases} \mathcal{L}V(S,t) = 0 \\ V(S,T) = \max(K - S, 0) \\ V(0,t) = Ke^{-r(T-t)} \\ \lim_{S\to\infty} V(S,t) = 0 \end{cases}. \tag{2.12}$$

Since the value of the underlying asset S lies in the interval $[0, +\infty[$, the change of variables

$$x(S) = 1 - e^{-\frac{S}{L}}, \tag{2.13}$$

where L is a constant parameter, transforms the unbounded S-domain in the bounded x-domain $[0, 1[$ which allows us to choose a finite number of RBPI centers inside this interval [15]. Equivalently, we have $S(x) = -L\ln(1-x)$. With this transformation, $S = 0 \Rightarrow x(S) = 0$ and $\lim_{S\to\infty} x(S) = 1$. Furthermore,

$$\frac{\mathrm{d}x}{\mathrm{d}S} = -\Big(-\frac{1}{L}\Big)e^{-\frac{S}{L}} = \frac{1}{L}(1-x) = \frac{1-x}{L} \quad \text{for} \quad 0 < x < 1. \tag{2.14}$$

Defining $U(x,t) = V(S(x),t)$ then we must modify the BS operator (2.11) according with this change of variable:

$$\begin{aligned}
\widetilde{\mathcal{L}}U(x,t) &= \frac{\partial}{\partial t}V(S(x),t) + \frac{\sigma^2}{2}S(x)^2\frac{\partial^2}{\partial S(x)^2}V(S(x),t) + rS(x)\frac{\partial}{\partial S(x)}V(S(x),t) \\
&\quad - rV(S(x),t) \\
&= \frac{\partial}{\partial t}U(x,t) + \frac{\sigma^2}{2}(-L\ln(1-x))^2\Bigg(\left(\frac{1-x}{L}\right)^2\frac{\partial^2}{\partial x^2}U(x,t) \\
&\quad + \frac{x-1}{L^2}\frac{\partial}{\partial x}U(x,t)\Bigg) + r(-L\ln(1-x))\frac{\partial}{\partial x}U(x,t)\left(\frac{1-x}{L}\right) - rU(x,t) \\
&= \frac{\partial}{\partial t}U(x,t) + \frac{\sigma^2}{2}(1-x)^2\ln^2(1-x)\frac{\partial^2}{\partial x^2}U(x,t) + \Bigg(\frac{\sigma^2}{2}(x-1)\ln^2(1-x) \\
&\quad - r(1-x)\ln(1-x)\Bigg)\frac{\partial}{\partial x}U(x,t) - rU(x,t)
\end{aligned}$$

using the chain rule to calculate the following partial derivatives:

$$\frac{\partial V}{\partial S} = \frac{\partial U}{\partial x}\frac{\mathrm{d}x}{\mathrm{d}S} = \frac{\partial U}{\partial x}\left(\frac{1-x}{L}\right)$$

$$\begin{aligned}
\frac{\partial^2 V}{\partial S^2} &= \frac{\partial^2 U}{\partial x^2}\frac{\mathrm{d}x}{\mathrm{d}S}\left(\frac{1-x}{L}\right) + \frac{\partial U}{\partial x}\frac{\partial}{\partial S}\left(\frac{1-(1-e^{-\frac{S}{L}})}{L}\right) \\
&= \frac{\partial^2 U}{\partial x^2}\left(\frac{1-x}{L}\right)^2 + \frac{\partial U}{\partial x}\left(-\frac{1}{L^2}e^{-\frac{S}{L}}\right) = \frac{\partial^2 U}{\partial x^2}\left(\frac{1-x}{L}\right)^2 + \frac{\partial U}{\partial x}\left(\frac{x-1}{L^2}\right)
\end{aligned}$$

Finally,

$$\widetilde{\mathcal{L}}U(x,t) = \frac{\partial}{\partial t}U(x,t) + \mathcal{A}(x)\frac{\partial^2}{\partial x^2}U(x,t) + \mathcal{B}(x)\frac{\partial}{\partial x}U(x,t) - rU(x,t) \tag{2.15}$$

where

$$\mathcal{A}(x) = \frac{\sigma^2}{2}(1-x)^2\ln^2(1-x) \text{ and } \mathcal{B}(x) = \frac{\sigma^2}{2}(x-1)\ln^2(1-x) - r(1-x)\ln(1-x).$$

Then, with the change of variables (2.13), the initial problem (2.12) now writes

$$\begin{cases} \widetilde{\mathcal{L}}U(S,t) = 0 \\ U(x,T) = \max(K + L\ln(1-x), 0) \\ U(0,t) = Ke^{-r(T-t)} \\ U(1,t) = 0 \end{cases} . \tag{2.16}$$

2.2 Black Scholes Formula

In their paper [3], Black and Scholes derived a formula from the partial differential equation that is known as the Black Scholes formula. We will skip its derivation which can be found in [3].

The price of a European call option with strike price K and time of maturity T is given by

$$C(t,S) = SN[d_1(t,S)] - e^{-r(T-t)}KN[d_2(t,S)], \tag{2.17}$$

where N is the Normal cumulative distribution function and

$$d_1(t,S) = \frac{1}{\sigma\sqrt{T-t}}\left(\ln\left(\frac{S}{K}\right) + \left(r + \frac{\sigma^2}{2}\right)(T-t)\right),$$
$$d_2(t,S) = d_1(t,S) - \sigma\sqrt{T-t}.$$

The value of a Put Option $P(S,t)$ may be derived from the value of a Call Option $C(t,S)$ using a known formula called Put-Call Parity

$$C(t,S) - P(t,S) = S - Ke^{-r(T-t)} \Leftrightarrow P(t,S) = C(t,S) - S + Ke^{-r(T-t)}. \tag{2.18}$$

Note that this formula is suitable in a very limited context. Any changes in the model hypothesis or in the contract conditions obliges the numerical solution of the Black Scholes equation instead of using this closed formula. Nevertheless, this formula allows to validate the numerical solution of the equation since it provides an exact solution (in the terms fixed).

2.3 Other pricing models

The Black-Scholes model is based on the Brownian Motion (or Wiener process) that follows a Normal Distribution. One of the drawbacks of this model is that empirical evidence goes against the Normal distribution of the returns. This way, other more general processes were considered, for instance Lévy processes, which use an infinitely divisible distribution, maintaining the good properties of the Wiener process (independent and stationary increments) [17]. Furthermore, Lévy processes have jumps, which is another important property to consider in the price process that the Black Scholes model does not include.

Consider that the dynamics of an underlying asset have the form $S_t = S_0 e^{-rt+X_t}$, where X_t is a Lévy process. Then, the risk-netural dynamics of S_t are given by

$$S_t = S_0 + \int_0^t rS_{u-}du + \int_0^t S_{u-}\sigma dW_u + \int_0^t \int_{-\infty}^{+\infty} (e^x - 1)S_{u-}\tilde{J}_X(du, dx) \tag{2.19}$$

where $\tilde{J}_X$ is the compensated random measure describing the jumps of X, S_t is a Markov process with infinitesimal generator

$$L^S f(x) = rx\frac{\partial f}{\partial x} + \frac{\sigma^2 x^2}{2}\frac{\partial^2 f}{\partial x^2} + \nu(dy)\left[f(xe^y) - f(x) - x(e^y - 1)\frac{\partial f}{\partial x}\right]. \tag{2.20}$$

Using a proper change of variable we obtain a equation for the value of a European option $C(t, S)$:

$$\begin{cases} \frac{\partial C}{\partial t} + L^S C(t, S) - rC(t, S) = 0 \\ C(T, S) = \Phi(S) \end{cases}. \tag{2.21}$$

This is similar to the Black-Scholes equation mentioned before, with the difference in the operator: instead of a second order differential operator we now have a integro-differential operator [5].

Another argument against the hypothesis of the Black Scholes model is that volatility is not constant. First, let us define what is the implied volatility. Suppose we know the price of an option and the price of its underlying for a certain maturity and strike price with today's risk-free asset. Then, we may implicitly obtain the value of the market's volatility - this is called the implied volatility. Now, if the assumption of constant volatility was to be right, then collecting data concerning

different options with the same maturity on the same underlying and plotting the implied volatility against the strike price, it should present a straight line. Instead, if we do so, we obtain a curve with the shape of a "smile" - that is called the volatility smile [2]. This proves that contrarily to the assumption of the Black Scholes model, volatility depends on the strike price.

In spite of the limitations of the Black-Scholes model, since we will focus on the RBPI method, we will not use more elaborated models, although they could be considered.

Chapter 3

Numerical Methods

The need to solve complex problems in many fields in science and engineering, along with the computational solution provided the development of numerical analysis. The first recorded use of finite differences to approximate derivatives was in Euler's method in 1768, which consists of a procedure to solve a first order differential equation [1]. Since then, this approach has been extended and improved to higher order problems.

Finite difference methods allow to numerically solve partial differential equations such as the Black Scholes equation, using differences to approximate the partial derivatives, based on a discrete grid on space and time. They start with an initial or final condition and solve the problem at each node in time and space by iteration.

Meshless methods are a very powerfull tool to solve partial differential equations since they avoid the generation of a grid and can be applied with any node distribution [15]. The radial basis point interpolation that will be explored later on is one possible mesh-free approach. They allow to reconstruct unknown functions that may be solutions of the PDE from known data.

Finite element methods provide data dependent spaces, i.e. the space furnishing the "trial" functions is fixed in advance. The reconstruction of multivariate functions cannot be done this way, that's why meshless methods have been suggested: they avoid triangulations, re-meshing and other geometric programming [16].

Another way to value derivatives is to use binomial methods. The binomial option pricing model assumes time to be discrete and divided in periods and calculates

the price of an option by iteration between the time of maturity and the valuation date. Assuming that the interest rate is constant and that the price of a derivative follows a binomial process over each period of time, if the current price of a stock is S then at the end of one period its price will be either uS if it goes up (with probability q) or dS if it goes down (with probability $1-q$) where $u > 1$ and $d < 1$ [6]. In the end of that period the same happens and so on.

It is possible to simulate the trajectory of a stochastic process using Monte Carlo Method which samples outcomes of the process randomly. This is useful in the pricing of derivatives since they depend on the evolution of the underlying stock. The method generates a random path for S and calculates the pay-off of the derivative associated to that stock price. Then it repeats this process many times and calculate the mean of the obtained payoffs. Discounting this average pay-off at the risk-free rate, a value for the derivative is obtained [11]. This is a heavy method since it requires a high number of simulations.

We will not go into detail on binomial models and Monte Carlo simulations since our focus is to price options using the numerical solution of the BS equation.

3.1 Finite Difference Methods

The value of the derivative of a function in a specific point can be approximated by the difference of values of the function in that point and in a point in its neighbourhood. There are three different ways to approximate it [8]: forward difference (3.1), backward difference (3.2) and centered difference (3.3)

$$f'(a) \approx D_+ f(a) \equiv \frac{f(a+h) - f(a)}{h}, \tag{3.1}$$

$$f'(a) \approx D_- f(a) \equiv \frac{f(a) - f(a-h)}{h}, \tag{3.2}$$

$$f'(a) \approx D_0 f(a) \equiv \frac{f(a+h) - f(a-h)}{2h}. \tag{3.3}$$

The approximate formulas presented above may be used to solve Partial Differential Equations in two independent variables such as the Black Scholes Equation. By convention, those variables are usually x (space) and t (time).

Example 1 *Suppose we have a simple PDE like the heat equation to solve using finite differences*

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} \tag{3.4}$$

We consider a grid of mesh points $(x,t) = (jh, nk)$ *where* h *and* k *are the space and time step of the problem (which are small, tending to zero) and* j *and* n *are natural numbers. The approximated solution of the problem in those points,* U_j^n *will be obtained replacing the partial derivatives by finite differences [18]. According to the forward, backward and centered differences, respectively, we define:*

- *in space*

$$D_+U^n(x) = \frac{U_{j+1}^n - U_j^n}{h}, \quad D_-U^n(x) = \frac{U_j^n - U_{j-1}^n}{h}, \quad D_0U^n(x) = \frac{U_{j+1}^n - U_{j-1}^n}{2h}.$$

- *in time*

$$D_+U_j(t) = \frac{U_j^{n+1} - U_j^n}{k}, \quad D_-U_j(t) = \frac{U_j^n - U_j^{n-1}}{k}, \quad D_0U_j(t) = \frac{U_j^{n+1} - U_j^{n-1}}{2k}.$$

For the second order derivative in space, we use the three-point formula

$$D_+D_-U^n(x) = \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{h^2}. \tag{3.5}$$

This way, equation (3.4) may be written in differences

$$\frac{U_j^{n+1} - U_j^n}{k} = \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{h^2}. \tag{3.6}$$

When we consider the full discretization, in both time and space, two types of schemes can be used: the explicit or implicit, corresponding to the specific method used for time integration. The differences between them are that the explicit methods, as the name implies, expresses the solution at $t = (n+1)k$ explicitly in terms of the values in the previous time step $t = nk$, whereas in the implicit schemes a system of equations must be solved in order to compute the solution at time $t = (n+1)k$. Among the implicit schemes, a very popular one is the Crank Nicholson scheme, providing second order accuracy in both space and time. The finite difference scheme used in (3.6) is the explicit.

3.1.1 Euler Implicit Scheme

We will now focus on the fully implicit finite difference scheme [19] which is the one that will be used later on. It uses the backward difference scheme for the partial derivative in time, the centered difference scheme for the first order derivative in space and the three point formula for the second order derivative in space.

In finite difference methods one should define a space-time grid in order to discretize the partial derivatives in time and space. Let l, h, k be constants such that $l > 0$, $h \in (0, 1]$ is the space step, $k \in (0; 1]$ is the time step and $M = l/k$ is the number of time discretizations.

$$Q_{h,k} = \{(x,t) : x = jh, t = nk, j = -M, \ldots, M, n = 0, \ldots, 1\} \tag{3.7}$$

Let U be the function that represents the solution to problem (2.16) in $Q_{h,k}$:

$$U(jh, nk) = U_j^n. \tag{3.8}$$

Using the most usual change of variables when trying to solve the Black Scholes equation $S = e^x$ [9] then, setting $U(x,t) = V(S,t)$, problem (2.12) is equivalent to

$$\begin{cases} \frac{\partial}{\partial t}U(x,t) + \frac{\sigma^2}{2}\frac{\partial^2}{\partial x^2}U(x,t) + \left(r - \frac{\sigma^2}{2}\right)\frac{\partial}{\partial x}U(x,t) - rU(x,t) = 0 \\ U(x,T) = \max(K - e^x, 0) \\ U(0,t) = Ke^{-r(T-t)} \\ U(-l,t) = U(l,t) = 0 \end{cases} . \tag{3.9}$$

By means of discretization using the implicit scheme, this system can be approximated by

$$\begin{cases} \frac{U_j^{n+1} - U_j^n}{\Delta t} + \frac{\sigma^2}{2}\left(\frac{U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}}{(\Delta x)^2}\right) \\ \qquad + \left(r - \frac{\sigma^2}{2}\right)\frac{U_{j+1}^{n+1} - U_{j-1}^{n+1}}{2\Delta x} - rU_j^{n+1} = 0 \\ U_j^N = \max(K - e^x, 0) \\ U_{-M}^n = U_M^n = 0 \end{cases} . \tag{3.10}$$

Or, in matrix form,

$$AU^{n+1} = U^n \tag{3.11}$$

where

$$A = \begin{bmatrix} b & c & & & \\ a & b & c & & \\ & \ddots & \ddots & \ddots & \\ & & a & b & c \\ & & & a & b \end{bmatrix}, \; U^{n+1} = \begin{bmatrix} U^{n+1}_{-M+1} \\ U^{n+1}_{-M+2} \\ \vdots \\ U^{n+1}_{M-2} \\ U^{n+1}_{M-1} \end{bmatrix}, \; U^{n} = \begin{bmatrix} U^{n}_{-M+1} \\ U^{n}_{-M+2} \\ \vdots \\ U^{n}_{M-2} \\ U^{n}_{M-1} \end{bmatrix},$$

with

$$a = -\frac{\sigma^2}{2}\frac{\Delta t}{(\Delta x)^2} + \left(r - \frac{\sigma^2}{2}\right)\frac{\Delta t}{2\Delta x},$$
$$b = 1 + \Delta t\left(r + \frac{\sigma^2}{(\Delta x)^2}\right),$$
$$c = -\frac{\sigma^2}{2}\frac{\Delta t}{(\Delta x)^2} - \left(r - \frac{\sigma^2}{2}\right)\frac{\Delta t}{2\Delta x}.$$

At each step, the linear system presented above must be solved numerically.

3.2 Interpolation Methods

There is the need to use interpolation whenever we deal with discrete data and want to obtain an approximation for specific point (or set of points) that is not available. Linear interpolation allows to determine an unknown value between two given values by tracing the line that connects those two points and localizing the point of interest in that line. This is the simplest method of interpolation but there are many others.

3.2.1 Polynomial Interpolation

The advantages of using polynomials in approximation are their easy evaluation, differentiation and integration in a finite number of steps with the use of basic arithmetic operations [7].

A polynomial of order n (and n degrees of freedom) is a function of the type

$$p(x) = \beta_1 + \beta_2 x + \beta_2 x^3 + ... + \beta_n x^{n-1} = \sum_{i=1}^{n} \beta_i x^{i-1}. \tag{3.12}$$

The Lagrange method uses n different points $x_1, ...x_n$ in a determined interval [7]. The ith Lagrange polynomial for the data sites nodes is given by

$$l_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}. \tag{3.13}$$

This way, if $g(x_1), \ldots, g(x_n)$ are the known values for the data sites $x_1, \ldots, x_n$ then there is one function p verifying $p(x_i) = g(x_i)$ where

$$p = \sum_{i=1}^{n} g(x_i) l_i(x). \tag{3.14}$$

Then the linear system

$$\beta_1 + \beta_2 x_i + \ldots + \beta_n x_i^{n-1} = b_i \;\; i = 1, \ldots, n \tag{3.15}$$

has a unique solution for arbitrary values of b_i which makes p the only interpolant from the set of all polynomials of order n to g at the points x_i for $i = 1, \ldots, n$.

3.2.2 Radial Basis Point Interpolation Method

A radial basis function is a function which value depends on the distance of the argument from the origin (or any other center), therefore any rotations of it have no effect in its value [4]. This way, if φ is a radial basis function, then $\varphi(x) = \varphi(\|x\|)$.

There are several different radial basis functions, for example

- $\varphi_1(x) = x^2 \ln(x)$ (thin-plate splines)
- $\varphi_2(x) = \sqrt{x^2 + c^2}$ (multiquadratics) where c is a positive parameter
- $\varphi_3(x) = e^{-cr^2}$ (Gaussian)

This functions, apart from its radial symmetry, are smooth and have good Fourier transform properties.

When there is need to approximate scattered data (i.e. points that have no relationship with each other in space) in more than one dimension, the radial basis function method is a very common approach [4]. It has the advantage of the possibility to select the nodes for the interpolation [13].

Given known values y of an unknown "function" in the points $\xi \in S$ such that $y = f(\xi)$ we wish to interpolate the known values in order to construct the function f.

For approximations using RBF, the approximants s are normally finite linear combinations of translates of a radially symmetric basis function [4] (which means that any rotations of the point make no difference in its value) and have general form

$$s(x) = \sum_{\xi \in S} \lambda_\xi \varphi(\|x - \xi\|), \tag{3.16}$$

with coefficients $\lambda_\xi \in \mathbb{R}$.

The algorithm for obtaining the coefficients will now be described.

Let u be the function we want to obtain. By interpolation, the value of u at any given point x is approximated at $n+1$ scattered nodes $x_0, x_1, ..., x_n \in S$. There are multiple ways of performing this interpolation, depending on the chosen functions. We will focus on the Radial Basis Point Interpolation which uses a combination of polynomials and radial basis functions. Let us denote the function that interpolates u by u_{RBPI}. Then

$$u_{RBPI}(x) = \sum_{i=0}^{n} \alpha_i \varphi_i(x) + \sum_{j=0}^{m} \alpha_{n+1+j} \upsilon_j(x), \tag{3.17}$$

where φ_i is the radial basis function in the node x_i such that

$$\varphi_i(x) = \varphi(|x - x_i|) \text{ for } i = 0, 1, ..., n, \tag{3.18}$$

υ_j is the monomial of order j such that

$$\upsilon_j(x) = x^j \text{ for } j = 0, 1, ..., m \tag{3.19}$$

and α_i are the $n+m+2$ coefficients of the radial basis function (at the $n+1$ centers) and the $m+1$ monomials, respectively.

The presence of the monomials in the interpolation ensures the non-singularity of the interpolation matrix in case of need. Furthermore, while the radial basis functions have a local effect concentrated in the point, the monomials have a global effect in the whole domain.

Using the function (3.17) to interpolate the nodes x_i for $i = 0, 1, ..., n$ we obtain a system of equations to determine the coefficients $\alpha_0, \alpha_1, ..., \alpha_{n+m+2}$:

$$\begin{cases} \sum_{i=0}^{n} \alpha_i \varphi_i(x_0) + \sum_{j=0}^{m} \alpha_{n+1+j} \upsilon_j(x_0) = u_0 \\ \sum_{i=0}^{n} \alpha_i \varphi_i(x_1) + \sum_{j=0}^{m} \alpha_{n+1+j} \upsilon_j(x_1) = u_1 \\ \vdots \\ \sum_{i=0}^{n} \alpha_i \varphi_i(x_n) + \sum_{j=0}^{m} \alpha_{n+1+j} \upsilon_j(x_n) = u_n \end{cases} . \tag{3.20}$$

Note that this system is undetermined since we have $n+1$ equations to determine $n+m+2$ unknowns. This way, to have a determined system, if $m \neq 0$ we add more equations, for example the orthogonality condition:

$$\sum_{i=0}^{n} \alpha_i \upsilon_j(x_i) = 0 \quad \text{for} \quad j = 0, 1, ..., m. \tag{3.21}$$

Gathering the systems of equations (3.20) and (3.21) we have the linear system

$$GA = U \Leftrightarrow A = G^{-1}U, \tag{3.22}$$

where

$$G = \begin{bmatrix} R & P \\ P^T & 0 \end{bmatrix}, \tag{3.23}$$

$$R = \begin{bmatrix} \varphi_0(x_0) & \varphi_1(x_0) & \dots & \varphi_n(x_0) \\ \varphi_0(x_1) & \varphi_1(x_1) & \dots & \varphi_n(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_0(x_n) & \varphi_1(x_n) & \dots & \varphi_n(x_n) \end{bmatrix}, \; P = \begin{bmatrix} \upsilon_0(x_0) & \upsilon_1(x_0) & \dots & \upsilon_m(x_0) \\ \upsilon_0(x_1) & \upsilon_1(x_1) & \dots & \upsilon_m(x_1) \\ \vdots & \vdots & \ddots & \vdots \\ \upsilon_0(x_n) & \upsilon_1(x_n) & \dots & \upsilon_m(x_n) \end{bmatrix},$$

$$A = \begin{bmatrix} \alpha_0 & \alpha_1 & \dots & \alpha_n & 0 & \dots & 0 \end{bmatrix}^T,$$

$$U = \begin{bmatrix} u_0 & u_1 & \dots & u_n & 0 & \dots & 0 \end{bmatrix}^T.$$

After obtaining the vector of the coefficients A we may now obtain an approximation for any value of x using expression (3.17) or, equivalently,

$$u_{RBPI}(x) = \begin{bmatrix} R^T(x) & P^T(x) \end{bmatrix} A. \tag{3.24}$$

3.2.3 RBPI implementation in the Black Scholes model for European put options

In order to solve problem (2.16) numerically, one must discretize it. We will divide the interval of time $[0,T]$ in $M+1$ moments in time equally spaced $t_0, t_1, ..., t_M$ where $t_k = k\Delta t$ and $\Delta t = \frac{T}{M}$. This way, $U^k(x) = U(x, k\Delta t)$, $k = 0, 1, ..., M$. The partial derivative of $U(x,t)$ with respect to time may be approximated by

$$\begin{aligned}\frac{\partial}{\partial t}U(x,t) &\approx \frac{U(x,t+h)-U(x,t)}{h} \approx \frac{1}{\Delta t}(U(x,k\Delta t+\Delta t)-U(x,k\Delta t)) \\ &\approx \frac{1}{\Delta t}(U^{k+1}(x)-U^k(x)).\end{aligned} \tag{3.25}$$

Additionally, we must define a criteria to choose the value of the constant parameter L which constitutes the change of variable proposed in (2.13). Following the criteria used in [15], recall that the change of variable does an equivalence between the nodes x_i in the domain $[0,1]$ and the nodes s_i in the domain $[0,+\infty)$ such that $s(x) = -L\ln(1-x)$. We want the majority of the nodes s_i to be in the interval $[0, 2K]$. If we choose the parameter L in a way that 70% of the nodes x_i are in the interval $[0, 1-e^{-\frac{K}{L}}]$ then we have 70% of the nodes s_i in the interval $[0,K]$ and therefore almost all of them are in the interval $[0,2K]$ as intended.

Thus,

$$1-e^{-\frac{K}{L}} = w\Delta x \Leftrightarrow L = -\frac{K}{\ln(1-w\Delta x)}, \tag{3.26}$$

where $w = \frac{7}{10}\Delta x$.

With this approximation, using an implicit scheme, we may apply it to the Black Scholes operator (2.15) for each moment in time

$$\widetilde{\mathcal{L}}U^k(x) = \frac{1}{\Delta t}(U^{k+1}(x)-U^k(x)) + \mathcal{A}(x)\frac{\partial^2}{\partial x^2}U^k(x) + \mathcal{B}(x)\frac{\partial}{\partial x}U^k(x) - rU^k(x). \tag{3.27}$$

Using a Radial Basis Point Interpolation, we consider $n+1$ equally spaced points in the interval $[0,1]$ and approximate $U^k(x)$ with

$$U^k_{RBPI}(x) = \sum_{j=0}^{n} \lambda_j^k \phi_j(x) \quad \text{for} \quad k = 0, 1, ..., M, \tag{3.28}$$

where $\phi_0(x), \ldots, \phi_n(x)$ are the shape functions.

In order to get numerical results involving this method, one should construct a numerical code, deriving the necessary matrices to solve the approximation of the solution for each moment in time.

Considering equation (3.27), we want to solve $\widetilde{\mathcal{L}}U^k(x) = 0$, using $U^k_{RBPI}(x)$ as an approximation of $U^k(x)$ in order to perform the discretization in time. This way

$$\frac{1}{\Delta t}(U^{k+1}_{RBPI}(x) - U^k_{RBPI}(x)) + \mathcal{A}(x)\frac{\partial^2}{\partial x^2}U^k_{RBPI}(x) + \mathcal{B}(x)\frac{\partial}{\partial x}U^k_{RBPI}(x) - rU^k_{RBPI}(x) = 0 \tag{3.29}$$

with final condition $U^M = \max(K + L\ln(1-x), 0)$.

In matrix notation, considering the boundary conditions presented in (2.12), we get the following system

$$A\Phi''\lambda^k + B\Phi'\lambda^k - \left(r + \frac{1}{\Delta t}\right)\Phi\lambda^k + C\lambda^k = -\frac{1}{\Delta t}\Phi\lambda^{k+1} + D^k \tag{3.30}$$

$$\Leftrightarrow (A\Phi'' + B\Phi' - \left(r + \frac{1}{\Delta t}\right)\Phi + C)\lambda^k = -\frac{1}{\Delta t}\lambda^{k+1} + D^k, \tag{3.31}$$

solving recursively backwards for $k = M-1, \ldots, 1, 0$ starting from

$$U^M = \max(K + L\ln(1-x), 0) \Leftrightarrow (\Phi + C)\lambda^M = \max(K + L\ln(1-x), 0), \tag{3.32}$$

where

$$A = \begin{bmatrix} 0 & & & \\ & \mathcal{A}(x_1) & & 0 \\ & & \ddots & \\ & 0 & & \mathcal{A}(x_{n-1}) \\ & & & & 0 \end{bmatrix}, \; B = \begin{bmatrix} 0 & & & \\ & \mathcal{B}(x_1) & & 0 \\ & & \ddots & \\ & 0 & & \mathcal{B}(x_{n-1}) \\ & & & & 0 \end{bmatrix},$$

$$C = \begin{bmatrix} \varphi_0(0) & \varphi_1(0) & \ldots & \varphi_{n-1}(0) & \varphi_n(0) \\ 0 & 0 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 0 & 0 \\ \varphi_0(1) & \varphi_1(1) & \ldots & \varphi_{n-1}(1) & \varphi_n(1) \end{bmatrix}, \; D^k = \begin{bmatrix} Ke^{-r(T-k\Delta t)} \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

$$\Phi = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ \varphi_0(x_1) & \varphi_1(x_1) & \dots & \varphi_{n-1}(x_1) & \varphi_n(x_1) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \varphi_0(x_{n-1}) & \varphi_1(x_{n-1}) & \dots & \varphi_{n-1}(x_{n-1}) & \varphi_n(x_{n-1}) \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix},$$

$$\Phi' = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ \varphi_0'(x_1) & \varphi_1'(x_1) & \dots & \varphi_{n-1}'(x_1) & \varphi_n'(x_1) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \varphi_0'(x_{n-1}) & \varphi_1'(x_{n-1}) & \dots & \varphi_{n-1}'(x_{n-1}) & \varphi_n'(x_{n-1}) \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix},$$

$$\Phi'' = \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ \varphi_0''(x_1) & \varphi_1''(x_1) & \dots & \varphi_{n-1}''(x_1) & \varphi_n''(x_1) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \varphi_0''(x_{n-1}) & \varphi_1''(x_{n-1}) & \dots & \varphi_{n-1}''(x_{n-1}) & \varphi_n''(x_{n-1}) \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}.$$

Solving the system we obtain the coefficients λ^k for $k = 0, 1, ..., M$ that may be stored in a matrix of size $(n+1) \times (M+1)$:

$$\Lambda = \begin{bmatrix} \lambda_0^0 & \lambda_0^1 & \dots & \lambda_0^{M-1} & \lambda_0^M \\ \lambda_1^0 & \lambda_1^1 & \dots & \lambda_1^{M-1} & \lambda_1^M \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \lambda_{n-1}^0 & \lambda_{n-1}^1 & \dots & \lambda_{n-1}^{M-1} & \lambda_{n-1}^M \\ \lambda_n^0 & \lambda_n^1 & \dots & \lambda_n^{M-1} & \lambda_n^M \end{bmatrix}. \tag{3.33}$$

Then, we may able to determinate an approximate solution for a certain value of $x = 1 - e^{-\frac{S}{L}}$ using expression (3.28).

Constructing a cycle that goes through each column k of the matrix Λ and multiplies each element in the line j for $j = 0, 1, \dots, n$ by the shape functions on x we obtain a vector of solutions U_{RBPI} where each element correspond to the value of the option in time k.

Chapter 4

Numerical Results

The numerical results were performed on a PC Laptop Intel(R) Core(TM) i7-6500U CPU 3.1GHz 8GB RAM and the programs were developed and run under Matlab R2017a, 64-bit.

4.1 Interpolation: Polynomial vs. Radial Based

Let us select a known function as a reference for interpolation, i.e. to compute values for a grid of selected points to do an interpolation between those values with the aim of comparing the interpolated function with the original one, using both polynomials and radial basis functions.

Suppose that this function has the expression

$$f(x) = sin(8\pi x). \tag{4.1}$$

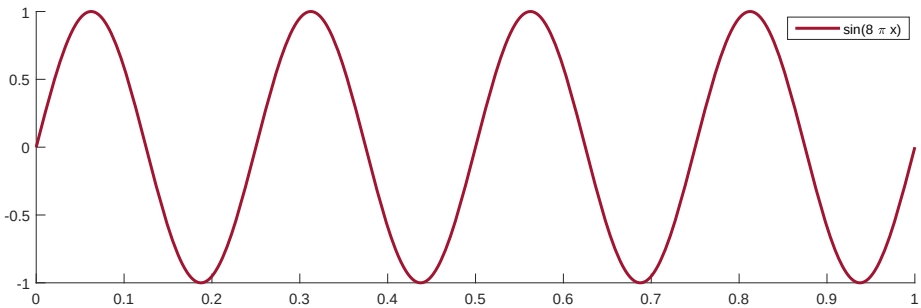

Figure 1: Graphic of the function to be approximated

Selecting randomly a set S of 10 points in the interval $[0, 1]$ and determining the images of the interpolation points $f(S)$, we may perform both interpolations using a 9^{th} order polynomial function $p(x) = \sum_{i=0}^{9} \beta_i x^i$ and a radial basis functions of the form $\varphi(x) = x^4 \ln(x)$, separately.

Starting this comparison with the interpolation using a RBF, we use the points $x_i \in S$ to implement the method described in (3.2.2) ignoring the polynomial terms, and obtain the coefficients α_i that for a given value x will be multiplied by the Radial Basis Function $\varphi_i(x)$ for $i = 0, 1, ..., 9$.

Similarly, we implement a polynomial interpolation using the same points $x_i \in S$ and obtain its coefficients β.

Then, we plot both interpolations against 100 equally spaced points in the interval $[0, 1]$.

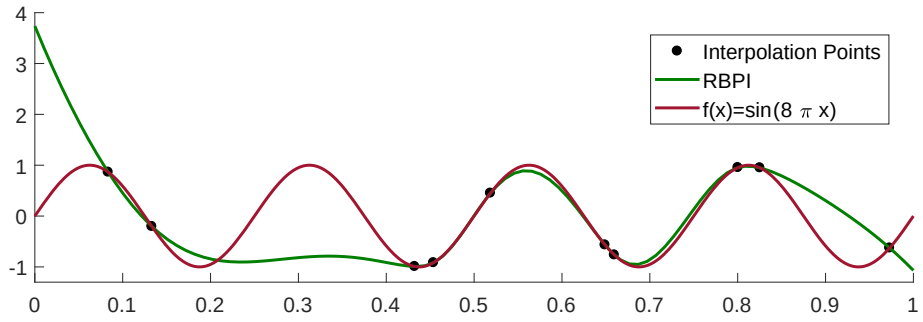

Figure 2: Radial Basis Point Interpolation

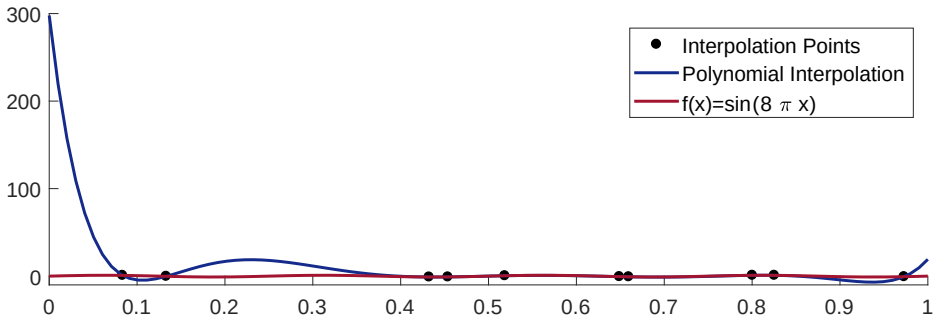

Figure 3: Polynomial Interpolation

Taking a look at both graphics, it is possible to observe that the radial basis point interpolation, in figure 2, gets much closer to the graphic of $f(x)$ than the polynomial interpolation, in figure 3. Let us define the maximum error and root mean square error:

$$MaximumError = \max_{i=0,1,\ldots,n} |\tilde{f}(x_i) - f(x_i)|, \tag{4.2}$$

$$RMSE = \frac{1}{n}\sqrt{\sum_{i=0}^{n}(\tilde{f}(x_i) - f(x_i))^2} \tag{4.3}$$

where $\tilde{f}$ is either the approximant using RBPI or polynomial interpolation.

Calculating this values, we observe in table 1 that we obtain a smaller error for the interpolation using the radial basis point interpolation method.

	Maximum Error	**RMSE**	**Time (s)**
Radial Basis Point Interpolation	3.7372	0.6634	0.057363
Polynomial Interpolation	297.8186	47.2143	0.155770

Table 1: Errors associated to the different interpolations

Recalling the types of radial basis functions presented before, we may now apply the RBPI method to approximate values of the same function $f(x)$ using different examples of radial basis functions to observe the associated error. In all of them, we try to implement the method using radial basis functions only and adding monomials.

We observe in table 2 that, in this experiments, the Radial Basis Function that presents a smaller associated error is $\varphi(x) = x^4 \ln(x)$ and that the addition of the polynomial term does not improve the convergence of the approximation to the exact solution. The computational times are very similar for every function and order of polynomials.

Function	Polynomial Order	Maximum Error	RMSE	Time Elapsed
$\sqrt{x^2+9}$	-	1.0408	0.1005	0.023328
	0	1.0405	0.1005	0.026673
	1	1.1330	0.0993	0.015527
	2	1.1462	0.0967	0.014248
	3	1.1505	0.0960	0.015118
	4	1.4300	0.1011	0.014430
e^{-3x^2}	-	1.2048	0.0984	0.015672
	0	1.1988	0.0985	0.016577
	4	1.4215	0.1010	0.016137
$x^2\ln(x)$	-	0.2540	0.0060	0.016625
	0	0.2538	0.0060	0.016814
	4	0.4064	0.0095	0.016944
$x^4\ln(x)$	-	0.0745	0.0016	0.016042
	0	0.0904	0.0020	0.016169
	4	0.0778	0.0017	0.020683

Table 2: Error associated to each radial basis function

4.2 Black-Scholes PDE discretization with RBPI

Let us consider a market with risk-free interest rate $r = 5\%$ and an European option with maturity $T = 2$, strike price $K = 80$ on an underlying asset with volatility $\sigma = 30\%$.

Taking into consideration the results obtained in the previous section, we will perform the radial basis point interpolation method with functions of the the type $\varphi(x) = x^4\ln(x)$ and monomials will not be used, unless the matrix of the coefficients becomes singular which obliges the addition of monomials.

Let the number of divisions of the space and time intervals, respectively, be $n = 100$ and $M = 1000$ which originates a space step $\Delta x = 0.01$ and a time step $\Delta t = 0.002$.

Using the methodology explained in section 3.2.3, we built a numerical code

presented in appendix A.2 that constructs the necessary matrices in order to solve a matricial system that gives as output a matrix of size $(n+1) \times (M+1)$ with the coefficients λ where each column refers to a moment in time t_j, $j = 0, 1, ..., M$ and has n coefficients λ_i, $i = 0, 1, ..., n$. With the use of an auxiliary function that associates the coefficients of this matrix to the radial basis function $\varphi(x)$ in the argument $\|x - \xi(i)\|$ we are now able to obtain a vector of solutions of size M for a specific value of x that represents the prices of the put option with the characteristics presented before at the different moments in time and for a specific price of the underlying asset $S = -L\ln(1-x)$.

Generating a vector of 100 points x_i in the interval $[0, 1]$ we may now calculate the value of the option for each one of the different prices S and, registering the first element of each obtained vector, we get to observe the evolution of the value of the put option at the establishment date $(t = 0)$ for different values of S in figure 4.

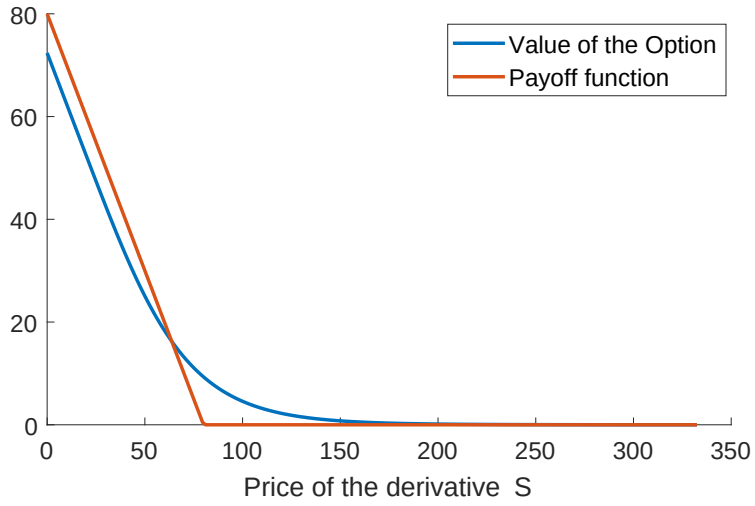

Figure 4: Value of the put option and payoff function

We may also observe the distribution of the centers $s_0, s_1, ...s_{99}$ in figure 5. Note that the node s_{100} represents $S \to \infty$ (since it is given, using the change of variables, by $x_100 = 1$).

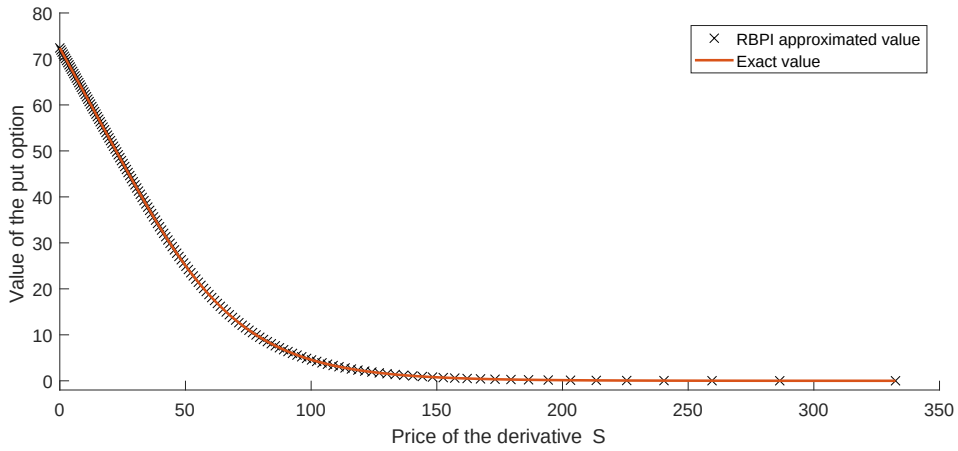

Figure 5: Position of the centers s_i

Furthermore, since we have a grid in time and space, we may plot a surface of the matrix of solutions $U^k(x)$ for $\forall k, x$. See figure 6.

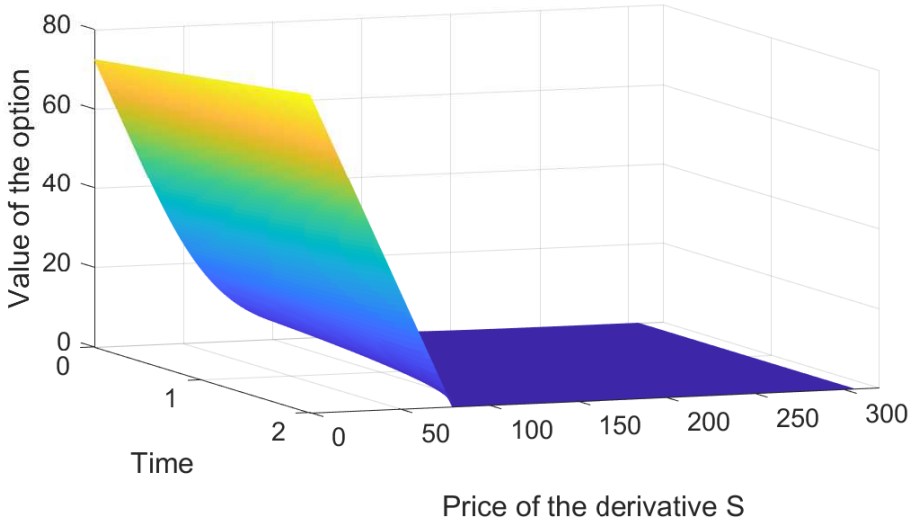

Figure 6: Value of the put option varying in time and space

Let $U_{RBPI}(x)$ and U denote the European put option price approximated by the RBPI method and calculated exactly using the Black Scholes Formula (presented in section 2.2), respectively.

As it have been done previously, we will use as reference the establishment date ($t = 0$). Settling the fixed parameters and changing the number of RBPI centers we may observe the evolution of the error with respect to the and time of computation of the approximated price.

We will compute both the maximum error and root mean square error between the approximated and exact price for each center x_i for $i = 0, 1, ..., n$.

$$MaximumError = \max_{i=0,1,...,n} |U_{RBPI}(x_i, 0) - U(x_i, 0)| \tag{4.4}$$

$$RMSE = \frac{1}{n}\sqrt{\sum_{i=0}^{n}(U_{RBPI}(x_i, 0) - U(x_i, 0))^2} \tag{4.5}$$

Let us fix the time step $M = 100$.

n	**Maximum Error**	**RMSE**	**Elapsed time** (s)
10	0.4619	0.0767	0.045508
25	0.0514	0.0035	0.089983
50	0.0346	0.0022	0.276126
75	0.0137	0.0009	0.539480
100	0.0214	0.0010	0.851010
150	0.0190	0.0007	2.246037
200	0.0182	0.0006	3.852224
500	0.0173	0.0004	36.866097

Table 3: Errors associated to different space divisions

4.2.1 The finite difference method vs. The RBPI method

Using the methodology presented in 3.1.1 we constructed a program (see A.3) to obtain the same results of the previous section but using the finite difference method (in particular the implicit scheme). Doing so, we may perform a comparison between both algorithms and finding the advantages of each one.

Let us divide space in $n = 500$ points and time in $M = 1000$ points.

Even though the algorithms constructed for the Finite Difference Method and for the Radial Basis Point Interpolation Method are based on different changes of variables from S to x, their outputs are comparable since after they derived in terms of the original variable S.

	Maximum Error	**RMSE**	**Elapsed Time (s)**
Finite Difference Method	0.5368	0.0065	0.134898
RBPI Method	0.0182	0.0006	4.404781

Table 4: FD method and RBPI method errors

It is possible to observe in table 4 that there is an advantage of accuracy in the results of the solution to the Black-Scholes equation using the RBPI method, when compared to the FD method. The computational time in order to derive the result is higher using the RBPI method since the matrices involved are much more complex than in the Finite Difference method, therefore the computational effort is bigger.

4.3 Convergence

We may now try some experiments changing the space and time steps in order to observe the evolution of the error comparing the approximated solution using the RBPI method with the value of the exact solution of the Black Scholes equation in the same conditions.

One can estimate the convergence order α by calculating the error derived from an experiment with two different space steps. From now on let us denote the space step Δx by h. Then, if h_1, h_2 are two different space steps and $RMSE(h_1)$,

$RMSE(h_2)$ are their associated root mean square errors, then the convergence order is given by

$$\frac{RMSE(h_1)}{RMSE(h_2)} = \left(\frac{h_1}{h_2}\right)^{\alpha} \Leftrightarrow \alpha = \frac{log\left(\frac{RMSE(h_1)}{RMSE(h_2)}\right)}{log\left(\frac{h_1}{h_2}\right)} \tag{4.6}$$

	$\Delta t = 0.1$			$\Delta t = 0.01$		
	RMSE	α	Time (s)	RMSE	α	Time (s)
$\Delta x = 0.1$	0.0829	1.2751	0.170502	0.0760	2.1233	0.058604
$\Delta x = 0.01$	0.0044		0.377598	5.722×10^{-4}		2.021670
$\Delta x = 0.05$	0.0197	0.9314	0.056961	0.0138	1.9777	0.124012
$\Delta x = 0.01$	0.0044		0.377598	5.722×10^{-4}		2.021670

Table 5: Convergence order evolution

The verified convergence order variation is mainly related with the discretization in time since it doesn't change much with the choice of different space steps.

4.4 Introduction of a source term in the IBVP

In the case of the Black Scholes equation, we have, in some cases, an exact solution that allows us to compare if the numerical method built is correct by comparing the obtained approximated solutions with the exact value. When the exact solution is not available the numerical method must be tested in some other way.

In order to do so, we fix a solution to the problem satisfying the final and boundary conditions, substituting it in the PDE and deriving a source term. Running the numerical method, the computed solution should be close to the exact one.

Introducing a source term in the problem (2.16), instead of a homogeneous PDE we have

$$\frac{\partial}{\partial t}U(x,t) + \mathcal{A}(x)\frac{\partial^2}{\partial x^2}U(x,t) + \mathcal{B}(x)\frac{\partial}{\partial x}U(x,t) - rU(x,t) = f(x,t) \tag{4.7}$$

Suppose that the solution to this problem is

$$U(x,t) = Ke^{-r(T-t)}(x-1)^2 \tag{4.8}$$

This solution verifies the boundary conditions in (2.16) since

$$U(0,t) = Ke^{-r(T-t)}(0-1)^2 = Ke^{-r(T-t)} \text{ and } U(1,t) = Ke^{-r(T-t)}(1-1)^2 = 0$$

We must change the final condition since, assuming the solution (4.8) then

$$U(x,T) = Ke^{-r(T-T)}(x-1)^2 = K(x-1)^2 \tag{4.9}$$

To derive the expression of $f(x,t)$, we must substitute the solution (4.8) in the PDE (4.7)

Starting with the calculus of the partial derivatives

$$\begin{aligned} \frac{\partial}{\partial t}U(x,t) &= rKe^{-r(T-t)}(x-1)^2, \\ \frac{\partial}{\partial x}U(x,t) &= 2Ke^{-r(T-t)}(x-1), \\ \frac{\partial^2}{\partial x^2}U(x,t) &= 2Ke^{-r(T-t)}, \end{aligned} \tag{4.10}$$

we now plug them in (4.7):

$$\begin{aligned} rKe^{-r(T-t)}(x-1)^2 + \mathcal{A}(x)2Ke^{-r(T-t)} + \mathcal{B}(x)2Ke^{-r(T-t)}(x-1) \\ - rKe^{-r(T-t)}(x-1)^2 = f(x,t) \\ \Leftrightarrow 2Ke^{-r(T-t)}(\mathcal{A}(x) + (x-1)\mathcal{B}(x)) = f(x,t) \end{aligned}$$

Substituting also the expressions for $\mathcal{A}(x)$ and $\mathcal{B}(x)$ written in (2.1.1), we get:

$$\begin{aligned} 2Ke^{-r(T-t)}\Bigg(\frac{\sigma^2}{2}(1-x)^2\ln^2(1-x) + (x-1) \\ \left(\frac{\sigma^2}{2}(x-1)\ln^2(1-x) - r(1-x)\ln(1-x)\right)\Bigg) = f(x,t). \end{aligned}$$

Finally, setting

$$f(x,t) = 2Ke^{-r(T-t)}\ln(1-x)(\sigma^2\ln(1-x) + r)(x-1)^2, \tag{4.11}$$

the exact solution is (4.8).

In order to implement this method numerically, suppose the same parameters used in the previous section: $\sigma = 30\%$, $r = 5\%$, $T = 2$ and $K = 80$. Let the time and space steps be $\Delta t = 0.002$ and $\Delta x = 0.01$.

The numerical procedure will be very similar to the one presented in (3.2.3) except for the introduction of the function in the equation to be solved numerically. This way equation (3.30) will now be written

$$\left(A\Phi'' + B\Phi' - \left(r + \frac{1}{\Delta t}\right)\Phi + C\right)\lambda^k = -\frac{1}{\Delta t}\lambda^{k+1} + D^k + F^k \tag{4.12}$$

where

$$F^k = \begin{bmatrix} 0 \\ f(x_1, t_k) \\ \vdots \\ f(x_{n-1}, t_k) \\ 0 \end{bmatrix}, \tag{4.13}$$

and all the other matrices are defined as in section 4.2. The matrix of the coefficients Λ will be obtained in the same way, solving recursively backwards for $k = 0, 1, ..., M-1$ starting from the final condition:

$$U^M = K(x-1)^2 \Leftrightarrow (\Phi + C)\lambda^M = K(x-1)^2. \tag{4.14}$$

Afterwards, we may now use the auxiliary function once again to obtain the value of the solutions for every time and space step and plot the numerical solution simultaneously with the exact solution and calculate the associated errors.

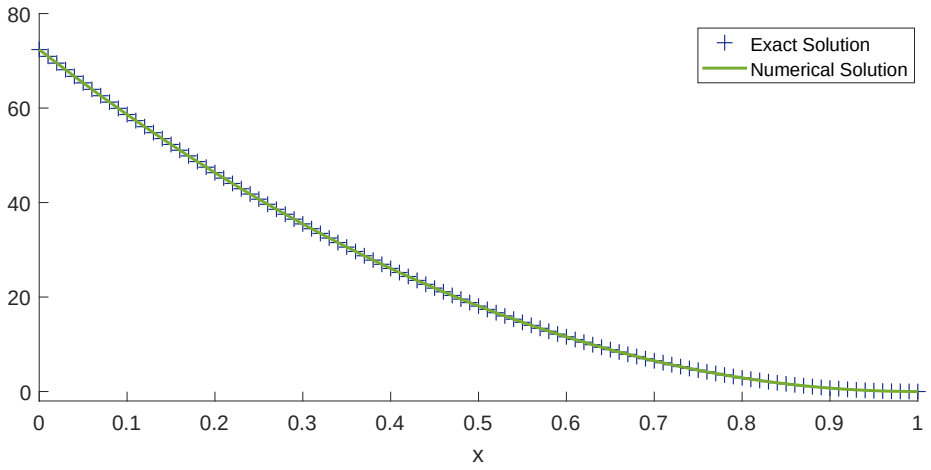

Figure 7: Exact and Numerical Solutions

Maximum Error	**RMSE**	**Elapsed Time (s)**
1.8×10^{-3}	2.4×10^{-5}	9.216839

Table 6: Errors associated to the experiment with a source term

Observing the obtained results in table 6, it is possible to conclude that the method is working with precision since the errors between the approximated and exact values tend to 0.

	$\Delta t = 0.1$			$\Delta t = 0.02$		
	RMSE	α	Time (s)	RMSE	α	Time (s)
$\Delta x = 0.1$	0.0204	1.4065	0.036503	0.0208	2.9379	0.304140
$\Delta x = 0.01$	0.0008		0.734559	2.4×10^{-5}		9.216839
$\Delta x = 0.05$	0.0033	0.8805	0.042677	0.0031	3.0204	0.467194
$\Delta x = 0.01$	0.0008		0.734559	2.4×10^{-5}		9.216839

Table 7: Convergence order evolution

Performing a convergence study, in table 7, once again we observe that the verified convergence order variation is mainly related with the discretization in time, although in this case there is a higher difference in its value changing the space step than in the previous section.

Chapter 5

Conclusion

We discretized the Black Scholes partial differential equation using the radial basis point interpolation to obtain its derivatives in space along with an implicit scheme for the discretization in time.

The method has been proven to be more accurate than the finite difference method in the discretization in space, since using the same number of time and space steps we obtain an approximation closer to the exact result using this proposed method.

We conclude that the method is working with accuracy, therefore it may be adapted to other problems. Since there is a closed formula to solve the problems that were used in this dissertation, it is interesting to try it on more complex types of options such as basket options where the number of underlying assets is bigger. Pricing this type of options cannot be done using closed formulas or even the finite difference method. The advantage of the radial basis point interpolation resides in the fact that it is possible to concentrate the computational efforts in the important points instead of distributing the efforts equally in the whole grid.

Future work in this field could also be done in terms of the relaxation of the limited assumptions of the Black-Scholes model such as constant volatility and absence of jumps in the price of the underlying asset.

Bibliography

[1] Ames, W. F. (2014). *Numerical methods for partial differential equations*. Academic press.

[2] Björk, T. (2009). *Arbitrage theory in continuous time*. Oxford university press.

[3] Black, F. and Scholes, M. (1973). The pricing of options and corporate liabilities. *Journal of political economy*, 81(3):637–654.

[4] Buhmann, M. D. (2003). *Radial basis functions: theory and implementations*, volume 12. Cambridge university press.

[5] Cont, R. and Voltchkova, E. (2005). Integro-differential equations for option prices in exponential lévy models. *Finance and Stochastics*, 9(3):299–325.

[6] Cox, J. C., Ross, S. A., Rubinstein, M., et al. (1979). Option pricing: A simplified approach. *Journal of financial Economics*, 7(3):229–263.

[7] De Boor, C., De Boor, C., Mathématicien, E.-U., De Boor, C., and De Boor, C. (1978). *A practical guide to splines*, volume 27. Springer-Verlag New York.

[8] Duffy, D. J. (2006). *Finite Difference Methods in Financial Engineering: A Partial Differential Equation Approach*. John Wiley & Sons Ltd.

[9] Gonçalves, F. (2017). 5. f-d schemes for european options. Lecture notes in Numerical Methods in Finance, Instituto Superior de Economia e Gestão.

[10] Grossinho, M. (2009). Métodos numéricos em finanças. *Texto de Apoio, ISEG*.

[11] Hull, J. C. (2015). *Options, Futures and Other Derivatives*. Pearson, 9th edition.

[12] Kohn, M. (1999). Risk instruments in the medieval and early modern economy.

[13] Liu, G.-R. (2002). *Mesh free methods: moving beyond the finite element method.* CRC press.

[14] Liu, G.-R. and Gu, Y.-T. (2005). *An introduction to meshfree methods and their programming.* Springer Science & Business Media.

[15] Rad, J. A., Parand, K., and Ballestra, L. V. (2015). Pricing european and american options by radial basis point interpolation. *Applied Mathematics and Computation*, 251:363–377.

[16] Schaback, R. (2007). A practical guide to radial basis functions. *Electronic Resource*, 11.

[17] Schoutens, W. (2003). *Lévy processes in finance.* Wiley.

[18] Thomée, V. (1990). Finite difference methods for linear parabolic equations. *Handbook of numerical analysis*, 1:5–196.

[19] Wilmott, P., Howison, S., and Dewynne, J. (1995). *The mathematics of financial derivatives: a student introduction.* Cambridge university press.

Appendix A

MATLAB Code

A.1 Radial Basis Point Interpolation

```
%% Radial Basis Point Interpolation
y=@(x) sin(8*pi*x);
xx=rand(10,1)';
n=length(xx)-1;
R=@(x,xx)(x-xx).^4*log(abs(x-xx));
P=@(x,j)x^j;
MatrixR=zeros(n+1,n+1);
for i=0:n
    for j=0:n
        MatrixR(i+1,j+1)=R(xx(i+1),xx(j+1));
    end
end
for i=0:n
    MatrixR(i+1,i+1)=0;
end
m=0;
MatrixP=zeros(n+1,m+1);
for i=0:n
    for j=0:m
        MatrixP(i+1,j+1)=P(xx(i+1),j);
    end
end
```

```
G=[MatrixR,MatrixP;transpose(MatrixP),zeros(m+1,m+1)];
U=[y(xx),zeros(1,m+1)];
coefs=G\U';
x=linspace(0,1,100);
z=zeros(1,length(x));
for i=1:length(x)
    z(i)=RBFinterp(x(i),n,m,xx,coefs,R,P);
end
figure()
plot(x,z);
hold on
scatter(xx,y(xx));
%% Polynomial Interpolation
p=polyfit(xx,y(xx),9);
x1=linspace(0,1);
y1=polyval(p,x1);
figure
plot(xx,y(xx),'o')
hold on
plot(x1,y1)
```

A.2 RBPI in the Black Scholes Model

```
sig=0.3;r=0.05;T=2;M=1000;K=80;
xi=linspace(0,1,101)';
n=length(xi)-1; dx=xi(2)-xi(1); dt=T/M;
t=zeros(1,M+1);
for k=0:M
    t(k+1)=k*dt;
end
w=7/(10*dx);L=-K/log(1-w*dx);
A=Diag0(@(x)(sig^2/2)*(1-x)^2*(log(1-x))^2,xi);
B=Diag0(@(x)(sig^2/2)*(x-1)*(log(1-x))^2-r*(1-x)*log(1-x),xi);
phi=@(x,xx)(x-xx).^4*log(abs(x-xx));
Phi=zeros(n+1,n+1);
for i=1:n+1
    for j=1:n+1
```

```
15          if  i==j
16              Phi(i,j)=0;
17          elseif  i==1
18              Phi(i,j)=0;
19          elseif  i==n+1
20              Phi(i,j)=0;
21          else
22          Phi(i,j)=phi(xi(i),xi(j));
23          end
24      end
25  end
26  phi1=@(x,xx)(x-xx).^3*(4*log(abs(x-xx))+1);
27  Phi1=zeros(n+1,n+1);
28  for  i=1:n+1
29      for  j=1:n+1
30          if  i==j
31              Phi1(i,j)=0;
32          elseif  i==1
33              Phi1(i,j)=0;
34          elseif  i==n+1
35              Phi1(i,j)=0;
36          else
37          Phi1(i,j)=phi1(xi(i),xi(j));
38          end
39      end
40  end
41  phi2=@(x,xx)(x-xx).^2*(12*log(abs(x-xx))+7);
42  Phi2=zeros(n+1,n+1);
43  for  i=1:n+1
44      for  j=1:n+1
45          if  i==j
46              Phi2(i,j)=0;
47          elseif  i==1
48              Phi2(i,j)=0;
49          elseif  i==n+1
50              Phi2(i,j)=0;
51          else
```

```
52            Phi2(i,j)=phi2(xi(i),xi(j));
53            end
54        end
55    end
56    BC1=zeros(n+1,n+1);
57    for i=1:n+1
58        if i==1
59            BC1(1,i)=0;
60        elseif i==n+1
61            BC1(n+1,i)=0;
62        else
63            BC1(1,i)=phi(0,xi(i));
64            BC1(n+1,i)=phi(1,xi(i));
65        end
66    end
67    UBC=K*exp(-r*(T-t));LBC=zeros(1,M+1);
68    BC2=zeros(n+1,M+1);BC2(1,:)=UBC;BC2(n+1,:)=LBC;
69    LambdaM=(Phi+BC1)\max(K+L*log(1-xi),0);
70    LambdaOld=LambdaM;
71    LAMBDA(:,M+1)=LambdaM;
72    for k=M:-1:1
73        LambdaNew=(A*Phi2+B*Phi1-(1/dt+r)*Phi+BC1)\(-(1/dt)*Phi*LambdaOld+
              BC2(:,k));
74        LAMBDA(:,k)=LambdaNew;
75        LambdaOld=LambdaNew;
76    end
77    %% Solution at the establishment date (t=0)
78    W=zeros(n+1,1);
79    for i=1:n+1
80        aux=BSinterp(xi(i),xi,phi,LAMBDA);
81        W(i)=aux(1);
82    end
83    plot(-L*log(1-xi),W)
84    hold on
85    plot(-L*log(1-xi),max(K+L*log(1-xi),0))
```

A.3 Finite Difference Method

```
sig=0.3;r=0.05;T=2;M=1000;K=80;n=500;
Xmax=log(2*K); Xmin=-Xmax; dx=(Xmax-Xmin)/(n-1); xi=(Xmin:dx:Xmax)';
dt=T/M;
t=zeros(1,M+1);
for k=0:M
    t(k+1)=k*dt;
end
alpha=(r-0.5*sig^2);
a=-0.5*sig^2*dt/dx^2+alpha*dt/(2*dx);
b=1+r*dt+dt*sig^2/dx^2;
c=-0.5*sig^2*dt/dx^2-alpha*dt/(2*dx);
A=Tridiag1(n,b,c,a);
A(1,1)=1; A(1,2)=0;
A(n,n-1)=0; A(n,n)=1;
UBC=K*exp(-r*(T-t));LBC=zeros(1,M);
USol=zeros(n,M);
USol(:,M)=max(K-exp(xi),0);
Uold=USol(:,M);
for k=M-1:-1:1
    Uold(1)=UBC(k);Uold(n)=LBC(k);
    Unew=A\Uold;
    USol(:,k)=Unew;
    Uold=Unew;
end
plot(exp(xi),USol(:,1))
hold on
plot(exp(xi),max(K-exp(xi),0))
```

A.4 RBPI with the introduction of a source term

```
sig=0.3;r=0.05;T=2;M=1000;K=80;
F=@(x,t)2*K*exp(-r*(T-t))*log(1-x)*(log(1-x)*sig^2+r)*(x-1).^2;
xi=linspace(0,1,101)';
n=length(xi)-1;
dx=xi(2)-xi(1);
```

```
dt=T/M;
t=zeros(1,M+1);
for k=0:M
    t(k+1)=k*dt;
end
w=7/(10*dx);
L=-K/log(1-w*dx);
A=Diag0(@(x)(sig^2/2)*(1-x)^2*(log(1-x))^2,xi);
B=Diag0(@(x)(sig^2/2)*(x-1)*(log(1-x))^2-r*(1-x)*log(1-x),xi);
phi=@(x,xx)(x-xx).^4*log(abs(x-xx));
Phi=zeros(n+1,n+1);
for i=1:n+1
    for j=1:n+1
        if i==j
            Phi(i,j)=0;
        elseif i==1
            Phi(i,j)=0;
        elseif i==n+1
            Phi(i,j)=0;
        else
        Phi(i,j)=phi(xi(i),xi(j));
        end
    end
end
phi1=@(x,xx)(x-xx).^3*(4*log(abs(x-xx))+1);
Phi1=zeros(n+1,n+1);
for i=1:n+1
    for j=1:n+1
        if i==j
            Phi1(i,j)=0;
        elseif i==1
            Phi1(i,j)=0;
        elseif i==n+1
            Phi1(i,j)=0;
        else
        Phi1(i,j)=phi1(xi(i),xi(j));
        end
```

```
43      end
44  end
45  phi2=@(x,xx)(x-xx).^2*(12*log(abs(x-xx))+7);
46  Phi2=zeros(n+1,n+1);
47  for i=1:n+1
48      for j=1:n+1
49          if i==j
50              Phi2(i,j)=0;
51          elseif i==1
52              Phi2(i,j)=0;
53          elseif i==n+1
54              Phi2(i,j)=0;
55          else
56          Phi2(i,j)=phi2(xi(i),xi(j));
57          end
58      end
59  end
60  BC1=zeros(n+1,n+1);
61  for i=1:n+1
62      if i==1
63          BC1(1,i)=0;
64      elseif i==n+1
65          BC1(n+1,i)=0;
66      else
67          BC1(1,i)=phi(0,xi(i));
68          BC1(n+1,i)=phi(1,xi(i));
69      end
70  end
71  UBC=K*exp(-r*(T-t));LBC=zeros(1,M+1);
72  BC2=zeros(n+1,M+1);BC2(1,:)=UBC;BC2(n+1,:)=LBC;
73  LambdaM=(Phi+BC1)\(K*(xi-1).^2);
74  LambdaOld=LambdaM;
75  LAMBDA(:,M+1)=LambdaM;
76  Faux=zeros(n+1,M+1);
77  for i=1:n+1
78      for j=1:M+1
79          if i==1
```

```
80                  Faux(i,j)=0;
81              elseif  i==n+1
82                  Faux(i,j)=0;
83              else
84                  Faux(i,j)=F(xi(i),t(j));
85              end
86          end
87      end
88      for  k=M:-1:1
89          LambdaNew=(A*Phi2+B*Phi1-(1/dt+r)*Phi+BC1)\(-(1/dt)*Phi*LambdaOld+
                BC2(:,k)+Faux(:,k));
90          LAMBDA(:,k)=LambdaNew;
91          LambdaOld=LambdaNew;
92      end
93      W=zeros(n+1,1);
94      for  i=1:n+1
95          aux=BSinterp(xi(i),xi,phi,LAMBDA);
96          W(i)=aux(1);
97      end
98      plot(xi,W,'+')
99      hold  on
100     plot(xi,K*(xi-1).^2*exp(-r*(T)),'r')
```

A.5 Black Scholes Formula

```
1   function  [call,put]=BSFormula(S0,K,r,sig,t0,T)
2   tau=T-t0;
3   if  tau==0;
4       call=max(S0-K,0);
5       put=max(K-S0,0);
6   else
7       d1=(log(S0/K)+(r+sig^2/2)*tau)/(sig*sqrt(tau));
8       d2=d1-sig*sqrt(tau);
9       call=S0*normcdf(d1)-normcdf(d2)*K*exp(-r*tau);
10      put=call+K*exp(-r*tau)-S0;
11  end
12  end
```

A.6 Auxiliar functions

```
function [u_RBPI]=RBFinterp(x,n,m,xx,coefs,R,P)
u_RBPI=0;
dim=n+m+2;
for i=0:n
    if x==xx(i+1)
        aux=0;
    else
        aux=R(x,xx(i+1));
    end
    u_RBPI=u_RBPI+aux*coefs(i+1);
end
for i=0:m
    u_RBPI=u_RBPI+P(x,i)*coefs(n+i+2);
end
end
```

```
function [M]=Diag0(u,xi)
M=zeros(length(xi),length(xi));
for i=2:length(xi)-1
    M(i,i)=u(xi(i));
end
end
```

```
function [USol]=BSinterp(x,xx,RBF,coefs)
[~,M]=size(coefs);
M=M-1;
n=length(xx)-1;
USol=zeros(n,1);
for k=1:M+1
    sum=0;
    for j=1:n+1
        if x==xx(j)
            aux1=0;
        else
            aux1=RBF(x,xx(j));
        end
```

```
14              aux2=coefs(j,k)*aux1;
15              sum=sum+aux2;
16          end
17          USol(k)=sum;
18      end
19  end
```