# Advanced Programming and Project (PRAP)

# Project Report

Alessio BALDINI

# Contents

# Chapter 1

# Introduction

The Black-Scholes equation is widely used in financial applications to determine the price of various types of assets. In this report, we study the application of the equation to European Vanilla Options, specifically a Call and a Put. These financial instruments are contracts signed at time $t = 0$, granting the owner the right to sell (Put option) or buy (Call option) an asset at a predetermined strike price $K$ at a future time $T$.
To address the problem of option pricing and to define the price of an option at time $t$, the following inputs are required:

- Expiration date of the option ($T$);
- Value of the asset at time $t$ ($S_t$);
- Market interest rate ($r$);
- Maximum value that the strike price can take ($L$);
- Volatility of the asset ($\sigma$);
- Strike price ($K$).

The price of an option at time $t$ for an underlying asset with value $S$ is defined as $C(S, t)$, or more generally, as $C$.

# Chapter 2

# Methodology

## 2.1 Discretization of the Domain

Before delving into the problem, the time range $[0, T]$ and asset value range $[0, L]$ are discretized into $M$ and $N$ intervals of length $\Delta t$ and $\Delta S$, represented as $t_i = i \cdot \Delta t$ and $S_j = j \cdot \Delta S$, where $\Delta t = \frac{T}{M}$ and $\Delta S = \frac{L}{N}$.
This approach creates a matrix where each column represents a time step, and each row represents an asset value step. The matrix contains prices $C(S, t)$ for $t \in [0, T]$ and $S \in [0, L]$, calculated using the Black-Scholes equation and Ito's formula.

## 2.2 Complete Equation Analysis

The first method involves analyzing the asset price using the partial differential equation derived from the Black-Scholes equation with Ito's formula:

$$\frac{\partial C}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} + rS\frac{\partial C}{\partial S} - rC = 0 \tag{2.1}$$

This equation can be rewritten as:

$$\frac{\partial C}{\partial t} = -\frac{1}{2}\sigma^2 S^2 \frac{\partial^2 C}{\partial S^2} - rS\frac{\partial C}{\partial S} + rC \tag{2.2}$$

Using Taylor's approximation, the partial derivatives in Equation (2.1) can be expressed in discrete terms.

Considering the Taylor series expansion up to the second order, a function $f(x)$ can be approximated in the neighborhood of a point $x_0$ as follows:

$$f(x) = f(x_0) + f'(x_0) \cdot \Delta x + \frac{f''(x_0) \cdot \Delta x^2}{2}.$$

Focusing first on the first-order term of the Taylor approximation:

$$f(x) - f(x_0) = f'(x_0) \cdot \Delta x, \quad \text{which implies } f'(x_0) = \frac{f(x) - f(x_0)}{\Delta x},$$

and applying it to $C(S,t)$, we obtain:

$$\frac{\partial C}{\partial t} = \frac{C(S, t + \Delta t) - C(S,t)}{\Delta t}.$$

Now, analyzing the problem from the perspective of the asset's value and incorporating both the first and second derivatives, we start with:

$$f(x) - f(x_0) = f'(x_0) \cdot \Delta x + \frac{f''(x_0) \cdot \Delta x^2}{2},$$

which leads to:

$$C(S + \Delta S, t) - C(S,t) = \frac{\partial C}{\partial S} \cdot \Delta S + \frac{\partial^2 C}{\partial S^2} \cdot \frac{\Delta S^2}{2},$$
$$C(S - \Delta S, t) - C(S,t) = \frac{\partial C}{\partial S} \cdot (-\Delta S) + \frac{\partial^2 C}{\partial S^2} \cdot \frac{\Delta S^2}{2}.$$

By subtracting these equations term by term, we derive:

$$C(S + \Delta S, t) - C(S - \Delta S, t) = \frac{\partial C}{\partial S} \cdot \Delta S + \frac{\partial C}{\partial S} \cdot \Delta S,$$
$$C(S + \Delta S, t) - C(S - \Delta S, t) = 2 \cdot \left( \frac{\partial C}{\partial S} \cdot \Delta S \right),$$

from which it follows:

$$\frac{\partial C}{\partial S} = \frac{C(S + \Delta S, t) - C(S - \Delta S, t)}{2\Delta S}.$$

Adding the two equations term by term instead, we obtain:

$$C(S + \Delta S, t) - 2C(S,t) + C(S - \Delta S, t) = 2 \cdot \left( \frac{\partial^2 C}{\partial S^2} \cdot \frac{\Delta S^2}{2} \right),$$
$$C(S + \Delta S, t) - 2C(S,t) + C(S - \Delta S, t) = \frac{\partial^2 C}{\partial S^2} \cdot \Delta S^2,$$

from which we can define:

$$\frac{\partial^2 C}{\partial S^2} = \frac{C(S + \Delta S, t) - 2C(S,t) + C(S - \Delta S, t)}{\Delta S^2}.$$

The following approximations are obtained:

- $$\frac{\partial C}{\partial t} = \frac{C(S, t + \Delta t) - C(S, t)}{\Delta t} \tag{2.3}$$

- $$\frac{\partial C}{\partial S} = \frac{C(S + \Delta S, t) - C(S - \Delta S, t)}{2\Delta S} \tag{2.4}$$

- $$\frac{\partial^2 C}{\partial S^2} = \frac{C(S + \Delta S, t) - 2C(S, t) + C(S - \Delta S, t)}{\Delta S^2} \tag{2.5}$$

### 2.2.1 Crank-Nicholson Scheme

The Crank-Nicholson scheme is applied to further discretize the continuous partial differential equation. This method combines explicit and implicit schemes:

- **Explicit Method**: Directly computes future values from current values but is conditionally stable and computationally costly for small time steps.
- **Implicit Method**: Computes values by solving a system of equations, ensuring unconditional stability but at a higher computational cost.

The Crank-Nicholson method averages these two methods to create a semi-implicit scheme. It is unconditionally stable and achieves second-order accuracy in both time and space.

To work backward from $T$ to $t = 0$, a change of variable is applied: $t = T - q$ (where $q = T - t$). Using the chain rule:

$$\frac{\partial C}{\partial t} = -\frac{\partial C}{\partial q}, \quad \frac{\partial^2 C}{\partial t^2} = \frac{\partial^2 C}{\partial q^2}$$

Let $C(S, q) = C_q^S$. By the definition of the scheme:

$$\frac{C_{t+\Delta t}^S - C_t^S}{\Delta t} = \frac{1}{2}\left[F\left(C_{(t+\Delta t)}\right) + F\left(C_{(t)}\right)\right],$$

Applying the change of variable:

$$\frac{C_{q+\Delta q}^S - C_q^S}{\Delta q} = -\frac{1}{2}\left[F\left(C_{(q+\Delta q)}\right) + F\left(C_{(q)}\right)\right],$$

from which it is possible to define

$$C^S_{q+\Delta q} + \frac{\Delta q}{2} F\left(C_{(q+\Delta q)}\right) = C^S_q - \frac{\Delta q}{2} F\left(C_{(q)}\right). \tag{2.6}$$

Where $F(C_q) = \frac{\partial C}{\partial q} = \frac{C^S_{q+\Delta q} - C^S_q}{\Delta q}$ is defined substituting approximation of partial derivatives (2.4), (2.5) in the Black-Scholes partial differential equation (2.2):

$$F(C_q) = -\frac{rS}{2\Delta S} \cdot \left(C^{S+\Delta S}_q - C^{S-\Delta S}_q\right) - \frac{\sigma^2 S^2}{2\Delta S^2} \cdot \left(C^{S+\Delta S}_q - 2C^S_q + C^{S-\Delta S}_q\right) + rC^S_q \tag{2.7}$$

The result obtained up to this point is the relation between the price of the option between two time steps. The intuition is to develop this relation to be able to define the price $C^S_{q+\Delta q}$ knowing $C^S_q$. It is now possible to define the two members of the equation (2.6), by substituting F with its definition (2.7). For the first member we get the following result:

$$C^S_{q+\Delta q} + \frac{\Delta q}{2}\left[-\frac{rS}{2\Delta S}\left(C^{S+\Delta S}_{q+\Delta q} - C^{S-\Delta S}_{q+\Delta q}\right) - \frac{\sigma^2 S^2}{2\Delta S^2}\left(C^{S+\Delta S}_{q+\Delta q} - 2C^S_{q+\Delta q} + C^{S-\Delta S}_{q+\Delta q}\right) + rC^S_{q+\Delta q}\right]$$

$$= C^{S-\Delta S}_{q+\Delta q}\left[-\frac{\Delta q}{4}\left(\frac{\sigma^2 S^2}{\Delta S^2} - \frac{rS}{\Delta S}\right)\right] + C^S_{q+\Delta q}\left[1 + \frac{\Delta q}{2}\left(r + \frac{\sigma^2 S^2}{\Delta S^2}\right)\right] + C^{S+\Delta S}_{q+\Delta q}\left[-\frac{\Delta q}{4}\left(\frac{\sigma^2 S^2}{\Delta S^2} + \frac{rS}{\Delta S}\right)\right]$$

.

For the second member we get the following result:

$$C^S_q - \frac{\Delta q}{2}\left[-\frac{rS}{2\Delta S}\left(C^{S+\Delta S}_q - C^{S-\Delta S}_q\right) - \frac{\sigma^2 S^2}{2\Delta S^2}\left(C^{S+\Delta S}_q - 2C^S_q + C^{S-\Delta S}_q\right) + rC^S_q\right] =$$

$$= C^{S-\Delta S}_q\left[\frac{\Delta q}{4}\left(\frac{\sigma^2 S^2}{\Delta S^2} - \frac{rS}{\Delta S}\right)\right] + C^S_q\left[1 - \frac{\Delta q}{2}\left(r + \frac{\sigma^2 S^2}{\Delta S^2}\right)\right] + C^{S+\Delta S}_q\left[\frac{\Delta q}{4}\left(\frac{\sigma^2 S^2}{\Delta S^2} + \frac{rS}{\Delta S}\right)\right]$$

.

Given that $\frac{S_j}{\Delta S} = j$, for $j \in [0; N]$, it's possible to write values inside square brackets as:

- $\alpha = \frac{\Delta q}{4}\left(\frac{\sigma^2 S^2}{\Delta S^2} - \frac{rS}{\Delta S}\right) = \frac{\Delta q}{4}\left(\sigma^2 j^2 - rj\right)$
- $\beta = -\frac{\Delta q}{2}\left(r + \frac{\sigma^2 S^2}{\Delta S^2}\right) = -\frac{\Delta q}{2}\left(r + \sigma^2 j^2\right)$
- $\gamma = \frac{\Delta q}{4}\left(\frac{\sigma^2 S^2}{\Delta S^2} + \frac{rS}{\Delta S}\right) = \frac{\Delta q}{4}\left(\sigma^2 j^2 + rj\right)$

We get the equation that defines the relation between prices in two any successive time steps. Consequently, for $q \in [0, T - \Delta q]$:

$$C_{q+\Delta q}^{S-\Delta S}(-\alpha) + C_{q+\Delta q}^{S}(1-\beta) + C_{q+\Delta q}^{S+\Delta S}(-\gamma) = C_{q}^{S-\Delta S}(\alpha) + C_{q}^{S}(1+\beta) + C_{q}^{S+\Delta S}(\gamma) \tag{2.8}$$

This system of equations can be represented in matrix form as $AC_{q+\Delta q} = BC_q$. Since the pricing problem as a terminal condition for the partial differential equation and B is the tridiagonal matrix with $(1 + \beta)$ on the principal diagonal, $\alpha$ in the lower diagonal and $\gamma$ in the upper one, that are known parameters, it is possible to compute $BC_q = b$. Therefore, the system defined by equation (2.8) can be represented in matrix form as $AC_{q+\Delta q} = b$:

$$\begin{pmatrix} (1-\beta_0) & -\gamma_0 & 0 & \cdots & \cdots & 0 \\ -\alpha_1 & (1-\beta_1) & -\gamma_1 & \cdots & \cdots & 0 \\ 0 & -\alpha_2 & (1-\beta_2) & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & (1-\beta_{L-1}) & -\gamma_{L-1} \\ 0 & 0 & 0 & \cdots & -\alpha_L & (1-\beta_L) \end{pmatrix} \begin{pmatrix} C_{q+\Delta q}^0 \\ C_{q+\Delta q}^1 \\ C_{q+\Delta q}^2 \\ \vdots \\ C_{q+\Delta q}^{L-1} \\ C_{q+\Delta q}^L \end{pmatrix} = \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{L-1} \\ b_L \end{pmatrix}$$

The matrix A is square tridiagonal with parameters defined in the equation (2.8). $C_{q+\Delta q}$ is the vector of prices of the option at time $q + \Delta q$ and b is the vector of the solutions of $BC_q$, both for each value S, with $S \in [0, L]$.

The result of the computations made up to this point is a linear system with which is possible to compute $C_q$ for any $q \in [0, T]$, that is a column of the matrix obtained by the discretization of the domain. The interest is now in computing, given $C_0$ by the terminal condition, all the $C_q$ up to $C_T$, which corresponds to $C - t = C_0$, the vector of option's prices for any S at time 0 (present time).

## 2.3 Reduced Equation Analysis

### 2.3.1 Change of variables

The second method involves analyzing the asset price using the partial differential equation derived from the Black-Scholes equation with a variable change that transforms our complete equation to a reduced one, akin to the heat equation like this:

$$\frac{d\tilde{C}}{d\tilde{t}} = \mu \frac{\partial^2 \tilde{C}}{\partial \tilde{s}^2} \tag{2.9}$$

With $\mu = 1$

With the change of variables :

$$\begin{cases} \tilde{t} = \frac{\sigma^2}{2}(T-t) \\ \tilde{s} = \ln(S/K) \\ C = Ku(\tilde{t}, \tilde{s}) \\ \tilde{C}(\tilde{t}, \tilde{s}) = e^{\alpha\tilde{s}+\beta\tilde{t}}u(\tilde{s}, \tilde{t}) \end{cases}$$

This gives the new derivative according to the chain rule :

$$\frac{\partial C}{\partial t} = K\frac{\partial u}{\partial \tilde{t}}\frac{\partial \tilde{t}}{\partial t} = -K\frac{\partial u}{\partial \tilde{t}}$$

$$\frac{\partial C}{\partial S} = K\frac{\partial u}{\partial \tilde{s}}\frac{\partial \tilde{s}}{\partial S} = K\frac{\partial u}{\partial s}\frac{1}{S} = e^{-x}\frac{\partial u}{\partial s}$$

$$\frac{\partial^2 C}{\partial S^2} = \frac{\partial}{\partial S}(\frac{\partial C}{\partial S}) = \frac{e^{-x}}{K}\frac{\partial}{\partial \tilde{s}}(e^{-x}\frac{\partial u}{\partial \tilde{s}}) = \frac{e^{-2x}}{K}(\frac{\partial^2 u}{\partial^2 \tilde{s}^2} - \frac{\partial u}{\partial s})$$

Thus :

$$\frac{\partial \tilde{C}}{\partial \tilde{t}} = e^{\alpha\tilde{s}+\beta\tilde{t}}(\beta u + \frac{\partial u}{\partial \tilde{t}})$$

$$\frac{\partial \tilde{C}}{\partial \tilde{s}} = e^{\alpha\tilde{s}+\beta\tilde{t}}(\alpha u + \frac{\partial u}{\partial \tilde{s}})$$

$$\frac{\partial^2 \tilde{C}}{\partial \tilde{s}^2} = e^{\alpha\tilde{s}+\beta\tilde{t}}(\alpha^2 u + 2\alpha\frac{\partial u}{\partial \tilde{s}} + \frac{\partial^2 u}{\partial \tilde{s}^2})$$

Which gives :

$$\beta u + \frac{\partial \tilde{C}}{\partial \tilde{t}} = \alpha^2 u + 2\alpha\frac{\partial \tilde{C}}{\partial \tilde{s}} + \frac{\partial^2 \tilde{C}}{\partial \tilde{s}^2} + (k-1)(\alpha u + \frac{\partial \tilde{C}}{\partial \tilde{s}}) - ku$$

with $k = 2r/\sigma^2$. To bring it to the heat equation we then require that the terms in $\tilde{C}$ and $\frac{\partial \tilde{C}}{\partial \tilde{s}}$ become 0.

Then :

$$\alpha = \frac{-1}{2}(k-1)$$
$$\beta = \frac{-1}{4}(k+1)^2$$

$$C(S,t) = Ke^{\frac{(1-k)\tilde{s}}{2} - \frac{\tilde{t}(k+1)^2}{4}}\tilde{C}(\tilde{s},\tilde{t})$$

Which is now a function that will verify the heat equation :

$$\frac{d\tilde{C}}{d\tilde{t}} = \frac{\partial^2 \tilde{C}}{\partial \tilde{s}^2} \tag{2.10}$$

This new differential equation is much simpler to manage now, and we will apply to it the implicit method of the finite differences. Let :

$$\Delta\tilde{s} = \ln\frac{L}{N}$$
$$\Delta\tilde{t} = -\frac{\sigma^2}{2}\Delta t$$

### 2.3.2 Implicit Finite Differences Scheme

Once the equation is reduced to the form of the heat equation, we can apply the finite differences method on it. The finite differences method consists in approaching $\frac{d\tilde{C}}{d\tilde{t}}$ and $\frac{d^2\tilde{C}}{d\tilde{s}^2}$ with their respective $\Delta\tilde{s}$ and $\Delta\tilde{t}$.

$$\frac{\tilde{C}_{n,j+1} - \tilde{C}_{n,j}}{\Delta\tilde{t}} = \frac{\tilde{C}_{n+1,j+1} - 2\tilde{C}_{n,j+1} + \tilde{C}_{n-1,j+1}}{\Delta\tilde{s}^2}$$

Which follows the identity :

$$\tilde{C}_{n,j} = \frac{\Delta\tilde{t}}{\Delta\tilde{s}^2}\left(-\tilde{C}_{n+1,j+1} + 2\tilde{C}_{n,j+1} - \tilde{C}_{n-1,j+1}\right) + \tilde{C}_{n,j+1}$$

Or :

$$\tilde{C}_{n,j} = \alpha\left(-\tilde{C}_{n+1,j+1} + 2\tilde{C}_{n,j+1} - \tilde{C}_{n-1,j+1}\right) + \tilde{C}_{n,j+1}$$

with $\alpha = \frac{\Delta\tilde{t}}{\Delta\tilde{s}^2}$

Then follows the matrix approach similar to what we did for the Cranck-Nicholson

method, giving a tridiagonal matrix with the complete formula but this time with only one parameter $\alpha$. This matrix A will be then separated with the LU decomposition to be more computationally efficient.

$A\tilde{C}_{j+1} = \tilde{C}_j$:

$$\begin{pmatrix} (1+2\theta) & -\theta & 0 & \cdots & \cdots & 0 \\ -\theta & (1+2\theta) & -\theta & \cdots & \cdots & 0 \\ 0 & -\theta & (1+2\theta) & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & (1+2\theta) & -\theta \\ 0 & 0 & 0 & \cdots & -\theta & (1+2\theta) \end{pmatrix} \begin{pmatrix} \tilde{C}^0_{j+1} \\ \tilde{C}^1_{j+1} \\ \tilde{C}^2_{j+1} \\ \vdots \\ \tilde{C}^{L-1}_{j+1} \\ \tilde{C}^L_{j+1} \end{pmatrix} = \begin{pmatrix} \tilde{C}^0_j \\ \tilde{C}^1_j \\ \tilde{C}^2_j \\ \vdots \\ \tilde{C}^{L-1}_j \\ \tilde{C}^L_j \end{pmatrix}$$

In this context, the interest is in determining $\tilde{C}_{j+1}$.

### 2.3.3 Boundary conditions determination

To use and perform an analysis on our new equation, we have to modify the boundary accordingly. As $\tilde{C}$ is defined for every $\tilde{s}$, we have only to focus on the boundary condition of time $\tilde{t}$. Terminal conditions for the complete equation are written as :

$$C(T, s) = max(0, K - s)$$

$$C(T, s) = max(0, s - K)$$

Respectively for a Call and a Put. We have also :

$$t = T \iff \tilde{t} = 0$$

$$\tilde{s} = \ln(S/K)$$

According to these variables change, the two conditions become:

- Call :

$$u(0, \tilde{s}) = \max(0, \exp(\tilde{s})) - 1$$

- Put :

$$u(0, \tilde{s}) = \max(0, 1 - \exp(\tilde{s}))$$

Applying the change of variable

$$\tilde{C}(\tilde{t}, \tilde{s}) = e^{\alpha\tilde{s}+\beta\tilde{t}}u(\tilde{s}, \tilde{t})$$

we get the terminal conditions for the heat equation version of Black-Scholes PDE:

- Call : $u(0, \tilde{s}) = \max(0, e^{\tilde{s}} - 1)$

$$\tilde{C}(0, \tilde{s}) = e^{\alpha x}u(0, \tilde{s}) \iff \tilde{C}(0, \tilde{s}) = \max(0, e^{\frac{1}{2}(k+1)\tilde{s}} - e^{\frac{1}{2}(k-1)\tilde{s}})$$

- Put : $u(0, \tilde{s}) = \max(0, 1 - e^{\tilde{s}})$

$$\tilde{C}(0, \tilde{s}) = e^{\alpha x}u(0, \tilde{s}) \iff \tilde{P}(0, \tilde{s}) = \max(0, e^{\frac{1}{2}(k-1)\tilde{s}} - e^{\frac{1}{2}(k+1)\tilde{s}})$$

Those equations are then the terminal conditions for pricing problem. Other boundary conditions are not taken into account to avoid conflicts with the asymptotic behaviour of the logarithm.

## 2.4 LU factorization

To solve a linear system in matrix form it has been implemented the LU factorization. It is a technique used to decompose a square matrix $A$ into the product of two triangular matrices: a lower triangular matrix $L$ and an upper triangular matrix $U$. This decomposition is useful for solving systems of linear equations, where the system $Ax = b$ is redefined according to $A = LU \rightarrow Ax = LUx = b$. Therefore, we can solve the system:

$$\begin{cases} Ly = b \\ Ux = y \end{cases}$$

.

In the case of a tridiagonal matrix, which is indeed the type of system that has to be solved using this method, the structure of $A$, $L$, and $U$ can be written as follows. The matrix of coefficients $A$ is:

$$A = \begin{pmatrix} a_0 & g_0 & 0 & \cdots & 0 \\ c_1 & a_1 & g_1 & \cdots & 0 \\ 0 & c_2 & a_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_n \end{pmatrix}$$

.

The LU decomposition splits $A$ into the product of:

$$L = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ low_1 & 1 & 0 & \cdots & 0 \\ 0 & low_2 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}, \quad U = \begin{pmatrix} up_0 & g_0 & 0 & \cdots & 0 \\ 0 & up_1 & g_1 & \cdots & 0 \\ 0 & 0 & up_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & up_n \end{pmatrix}$$

.

For $i \in [0, n]$, the elements of the lower diagonal $low_i$ and those of the upper diagonal $up_i$ can be computed, basing on their definition, as follows:

$$\begin{cases} up_o = a_o \\ c_i = low_i \cdot u_{i-1} \rightarrow low_i = \frac{c_i}{u_{i-1}}, \qquad i \in [1, n] \\ a_i = low_i \cdot g_{i-1} + up_i \rightarrow up_i = a_i - low_i \cdot g_{i-1}, \qquad i \in [1, n] \end{cases}$$

.

Using this decomposition and knowing elements of matrix A, it's possible to compute all the elements needed to solve the system:

- from $Ly = b$ it's obtained vector y:

$$\begin{cases} y_o = b_o \\ low_i \cdot y_{i-1} + y_i = b_i \rightarrow y_i = b_i - low_i \cdot y_{i-1}, \qquad i \in [1, n] \end{cases}$$

.

- then after having computed the vector y, from $Ux = y$ it's possible to find the vector unknown in the original system:

$$\begin{cases} up_n \cdot x_n = y_n \rightarrow x_n = \frac{y_n}{up_n} \\ up_j \cdot x_j + g_j \cdot x_{j+1} = y_j \rightarrow x_j = \frac{y_j - g_j \cdot x_{j+1}}{up_j}, \qquad j \in [n-1, 0] \end{cases}$$

.

The resulting vector x is the vector of prices. In it's computation, it must be kept into account boundaries, for which $x_n$ will be indeed substituted by the terminal condition, without taking into account its solution given by the system, as well as $x_0$, for which boundary solutions hold.

This decomposition simplifies the computation: the advantage of LU factorization lies in its efficiency. Once the LU decomposition is computed, it can be reused for solving multiple systems with the same matrix $A$ but different right-hand sides $b$.

# Chapter 3

# Technical Solutions

As the analysis is separated into two problems, for the complete and reduced equations, the structure of the program has been divided into three branches: the first, dealing with the complete PDE, using Crank-Nicolson Scheme, the second related to the reduced form of the PDE, applying variables change and the implicit finite differences method and the third, giving instruments to manage SDL2 library.

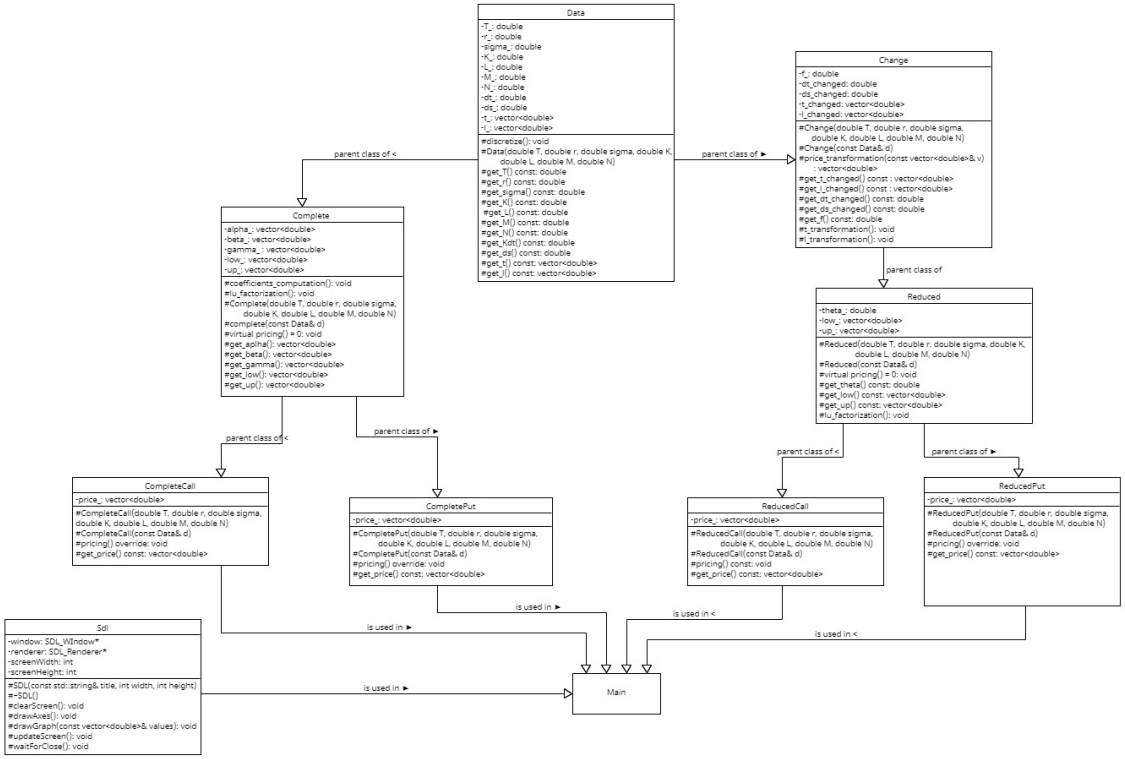


Figure 3.1: The UML diagram with every classes

The Data class has been designed to store every global variable and to discretize time and space in order to define the matrix that will contain prices for each step in time and underlying asset's value.
The Change class is specific for the reduced problem and it's crucial as it operates and it stores change of variables for the time, price and time and space steps.
The SDL Class calls the library and uses methods to print out the results.

The core of the code resides in the abstract class Complete and its children, and the abstract class Reduced and its children, where all the computations in order to get prices are done.

## 3.1 Complete Classes

The Complete section of the code is made of to two classes for the distinction between the case of Put and Call, and their parent Complete Class.
The Complete Class is asbtract: it contains the virtual method for pricing that is implemented by its children classes. It also declares the method for the LU factorization which is key to the computation and efficiency of the program, as well as the accessor for the parameters $\alpha, \beta, \gamma$ which are the coefficients attached to the diagonals of the matrix that define the problem's system.
CompleteCall and CompletePut are concrete classes that use the mathematical methods described previously to solve the system obtained through the LU factorization. The difference between them is not on the method itself, that works in the same way, but in their boundary conditions, that are specific for the type of Option. The price vector that results form the algorithm is then used to print the curve and to do comparison with results obtained through the reduced method.

## 3.2 Reduced Classes

The Reduced Class is the counterpart for the Complete Class for the reduced equation, but has one key difference: it is the children of the class Change contains the

methods used to apply the transformations to variables, in order to get the reduced version of the PDE. It contains also the method to change the values of the price vector $\tilde{C}$ in order to get C.
The Reduced class has a similar purpose of to the Complete one, with the exception of having only one parameter, $\theta$, that is the unique parameter of the matrix to be solved using the implicit finite differences method. The pricing() method is once again central in both children classes ReducedPut and ReducedCall. It computes prices after the transformation of the variables using Change class' methods.

## 3.3 Possible Amelioration

The two main challenges that has been faced in the development of the program are related to the management of the sdl library, particularly managing many windows together, and to the algorithm to compute the option price using the reduced PDE. The behavior of the Call option price for the reduced equation is unexpected: the algorithm is able to compute properly the price for almost the entire domain in $\tilde{s}$, but for values close to its upper bound we get an unexpected change towards zero of prices. This behavior is not observed on the Put function.
One of the main difficulties in the mathematical part is to find the correct change of variables, because several change of variables are valid, bringing to different results.

## 3.4 Printing Results

It can be observed that for the Put functions, we have a small discrepancy, which is expected, while for the Call function the sudden decrease of the reduced version of the Call (the green curve) is not justified by the model and could be the consequence of local problems in the algorithm.

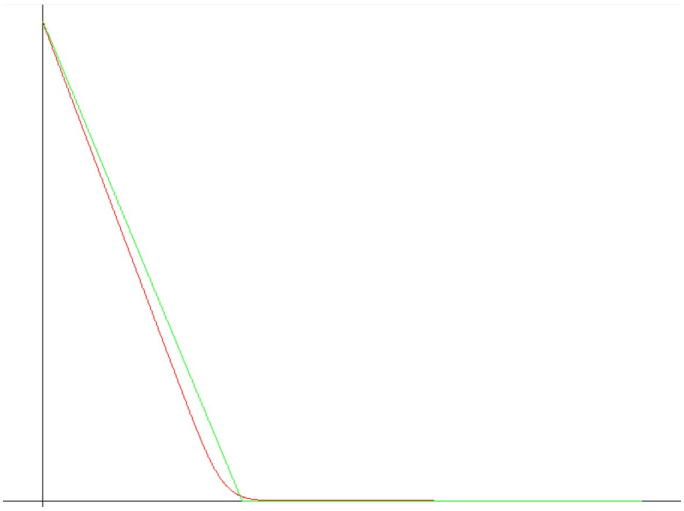

Figure 3.2: Both curves for a Put (scaleless)

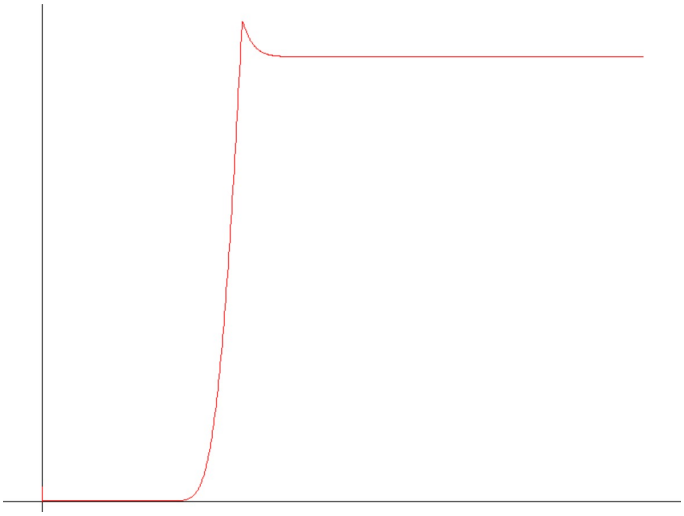

Figure 3.3: The difference between both Puts (scaleless)

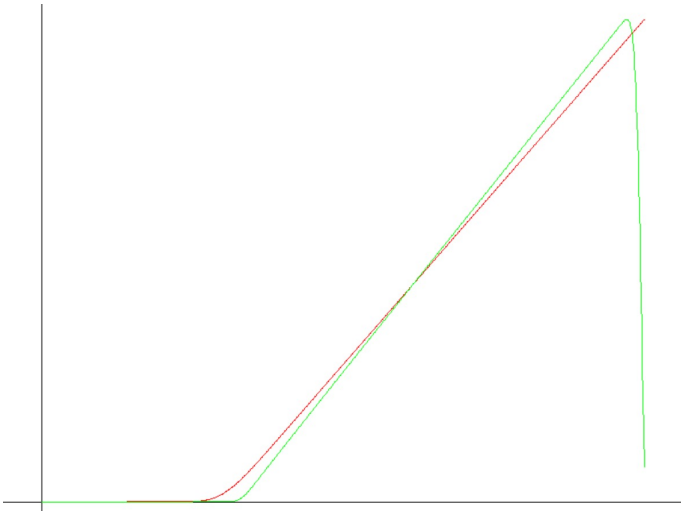

Figure 3.4: Both curves for a Call (scaleless)

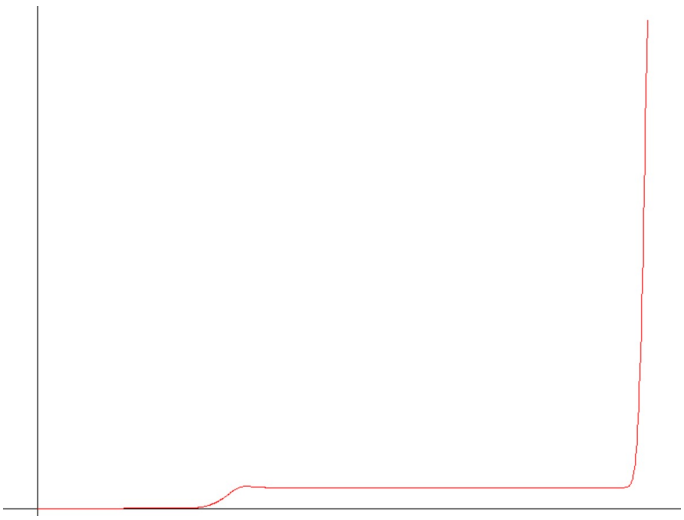

Figure 3.5: The difference between both Calls (scaleless)