Solving the N-Dimensional Black-Scholes Model with Mimetic Methods

A Final Project

Presented to the

Faculty of

San Diego State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computational Science

with a Concentration in

Data Science

by

Zachary Fraser Humphries

Spring 2024

SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Final Project of Zachary Fraser Humphries:

Solving the N-Dimensional Black-Scholes Model with Mimetic Methods

Miguel Dumett, Chair
Computational Science Research Center

Jose Castillo
Computational Science Research Center

Barbara Bailey
Department of Mathematics and Statistics

Approval Date

ABSTRACT OF THE FINAL PROJECT

Solving the N-Dimensional Black-Scholes Model with Mimetic Methods
by
Zachary Fraser Humphries
Master of Science in Computational Science with a Concentration in Data Science
San Diego State University, 2024

The Black-Scholes model is a partial differential equation that includes mixed second-order partial derivatives, developed by Fisher Black and Myron Scholes [4], to evaluate the underlying price of European options. An option is an agreement where someone can reserve to buy (call) or sell (put) a stock at a specific time. Unlike American options, European options can only be exercised at the maturity date. Finite differences and Radial Basis Functions have been utilized for solving the Black-Scholes model in two-dimensions. The aim of this work is to compare mimetic methods with those two techniques in two dimensions and to exhibit simple mimetic discrete analogs that can evaluate European options for a general portfolio.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1
Introduction

The Black-Scholes model, first developed by economists Fischer Black and Myron Scholes in 1973[4], is a mathematical model used for calculating the present value of European-style put and call options.

The model's one-dimensional differencial equation is described by the equation:

$$\frac{\partial F}{\partial t} = -rS\frac{\partial F}{\partial S} - \frac{1}{2}\Sigma^2 S^2 \frac{\partial^2 F}{\partial S^2} + rF \tag{1.1}$$

where:

- F is the discounted present value (price) of the option as a function of time (t) and the underlying asset price (S).
- $\frac{\partial F}{\partial t}$ is the partial derivative of F with respect to time.
- $\frac{\partial F}{\partial S}$ is the partial derivative of F with respect to the underlying asset price (S).
- $\frac{\partial^2 F}{\partial S^2}$ is the second partial derivative of F with respect to the underlying asset price (S).
- Σ is the volatility of the underlying asset.
- r is the risk free interest rate.

Because the value of the option is discounted back to the present, the initial condition for a European call option is the following payoff function at maturity time (T):

$$F(T, S) = \Phi(S) = \max(S - K, 0) \tag{1.2}$$

where:

- S is the underlying asset price.
- K is the strike price.

The model is risk-neutral and so it incorporates the risk-free interest rate (r) as the discount factor for future cash flows, assuming that investors require a return equivalent to the risk-free rate for taking on the risk associated with the underlying assets. The Black-Scholes model is expressed as a stochastic partial differential equation (SPDE), which accounts for the random and unpredictable nature of the underlying asset's price movements. The Black-Scholes equation is a PDE is derived from the SPDE.

In a multi-dimensional context, where there are n underlying assets, the Black-Scholes PDE is expressed as a sum of coupled partial derivatives. The extension to n-dimensions is particularly relevant in the context of options on baskets of assets, indices, or other derivatives dependent on multiple factors. The n-dimensional Black-Scholes PDE can be represented in a general form as follows:

$$\frac{\partial F}{\partial t} = -\sum_{i=1}^{n} r_i S_i \frac{\partial F}{\partial S_i} - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \sigma_i \sigma_j \rho_{ij} S_i S_j \frac{\partial^2 F}{\partial S_i \partial S_j} + rF \tag{1.3}$$

with the payoff function:

$$F(T,S) = \Phi(S) = \max\left(\left(\sum_{i=1}^{n} \frac{S_i}{n}\right) - K, 0\right) \tag{1.4}$$

where:

- F is the discounted present value (price) of the call option as a function of time (t) and the n underlying asset prices (S_1,S_2,. . .,S_n).
- $\frac{\partial F}{\partial t}$ is the partial derivative of F with respect to time.
- $\frac{\partial F}{\partial S_i}$ is the partial derivative of F with respect to the i-th underlying asset price (S_i).
- $\frac{\partial^2 F}{\partial S_i S_j}$ is the partial second derivative of F with respect to the i-th and j-th underlying asset prices (S_i and S_j).
- σ_i and σ_j are the volatilities of the corresponding underlying assets.
- ρ_{ij} is the correlation between assets i and j.
- r is is the risk-free interest rate applicable to the entire basket of assets.

Since the parameters σ_i, σ_j, and ρ_{ij} are multiplied together, the root of the product is often presented as a square n by n matrix as Σ

The model has made a significant impact on modern financial markets, particularly in the options trading and risk management sectors. For options traders and investors, the PDE has provided them with a quantitative tool to assess option prices, contributing to the growth and efficiency of options markets. Additionally, for risk management sectors, by providing a framework for pricing options, it aids financial institutions in managing their exposure to market volatility. The model has been increasingly important in the context of modern advanced financial instruments and portfolios.

1.1 Problem Statement

Mimetic differences are a method for numerically approximating spatial operators and when coupled to a time discretization method it produces mimetic schemes. Mimetic differences preserve symmetries, conservation laws, and other properties of partial differential equations. In addition, the discrete mimetic analogs achieve uniform high-order accuracy over the whole computational domain (including the boundaries). Mimetic difference operators are constructed to satisfy a discrete version of the integration by parts formula with high-order of accuracy. They attain this property by utilizing weighted inner products [2].

For solving the Black-Scholes model, it is necessary to consider appropriate boundary conditions, as is pointed out by Sundvall and Trang [7].

Different numerical techniques have been utilized for solving Black-Scholes in two-dimensions, more recently, Radial Basis Functions (RBF).

1.2 Proposed Solution

The proposed solutions to the previous section are as follows:

1. Numerically solve the two-dimension (2D) Black-Scholes model utilizing Mimetic Differences.
2. Replicate the RBF solution of the 2D Black-Scholes model according to [6] and [5].
3. Compare RBF and Mimetic Differences results.
4. Compare of Mimetic Differences model to actual data.
5. Extend the mimetic differences approach for the n-dimensional case in general.

1.3 Justification

The Black-Scholes model forms the basis of modern day options markets. Providing a way to solve the model using mimetic operators would have great use by expanding upon the existing implementations of the MOLE library and its application to financial markets.

CHAPTER 2
Problem Background

In [7], Sundvall and Trang explored how different boundary conditions impacted the results of the 2D Black-Scholes model.

Using the two-dimensional Black-Scholes PDE ([1.3] where n=2)

$$F_t = -rxF_x - ryF_y - \frac{1}{2}x^2\Sigma^2_{(1,1)}F_{xx} - \frac{1}{2}y^2\Sigma^2_{(2,2)}F_{yy} - xy\Sigma^2_{(1,2)}F_{xy} + rF \tag{2.1}$$

and the payoff function at the maturity time is

$$F(T,x,y) = \Phi(x,y) = \left(\frac{x+y}{2} - K\right)^+ \tag{2.2}$$

where

- r : Risk-free investment rate
- Σ : Volitility Correlation Matrix

$$\Sigma = \begin{bmatrix} \Sigma_{(1,1)} = \sqrt{\sigma_1\sigma_1\rho_{11}} & \Sigma_{(1,2)} = \sqrt{\sigma_1\sigma_2\rho_{12}} \\ \Sigma_{(2,1)} = \sqrt{\sigma_2\sigma_1\rho_{21}} & \Sigma_{(2,2)} = \sqrt{\sigma_2\sigma_2\rho_{22}} \end{bmatrix},$$

$$\Sigma_{(1,2)} = \Sigma_{(2,1)}, \rho_{11} = \rho_{22} = 1, \rho_{12} = \rho_{21}$$

- T : Final Maturity Time
- K : Strike Price

The payoff function used in the paper is

$$F(T,x,y) = \Phi(x,y) = \max\left(\frac{x+y}{2}, 0\right) \tag{2.3}$$

which serves as the initial conditions.

Due to the nature of options having a minimum value of zero (lower domain) but no maximum price, they divided the boundary conditions into close-field and far-field boundary conditions.

Figure 2.1 the close-field (as diamonds) and far-field (as stars) boundary conditions for the 2D model as defined in the paper. The triangular points where the close-field boundary conditions and meet the far-field bound- ary conditions can be either of the two, however, Sundvall and Trang defined as them far-field boundary conditions.

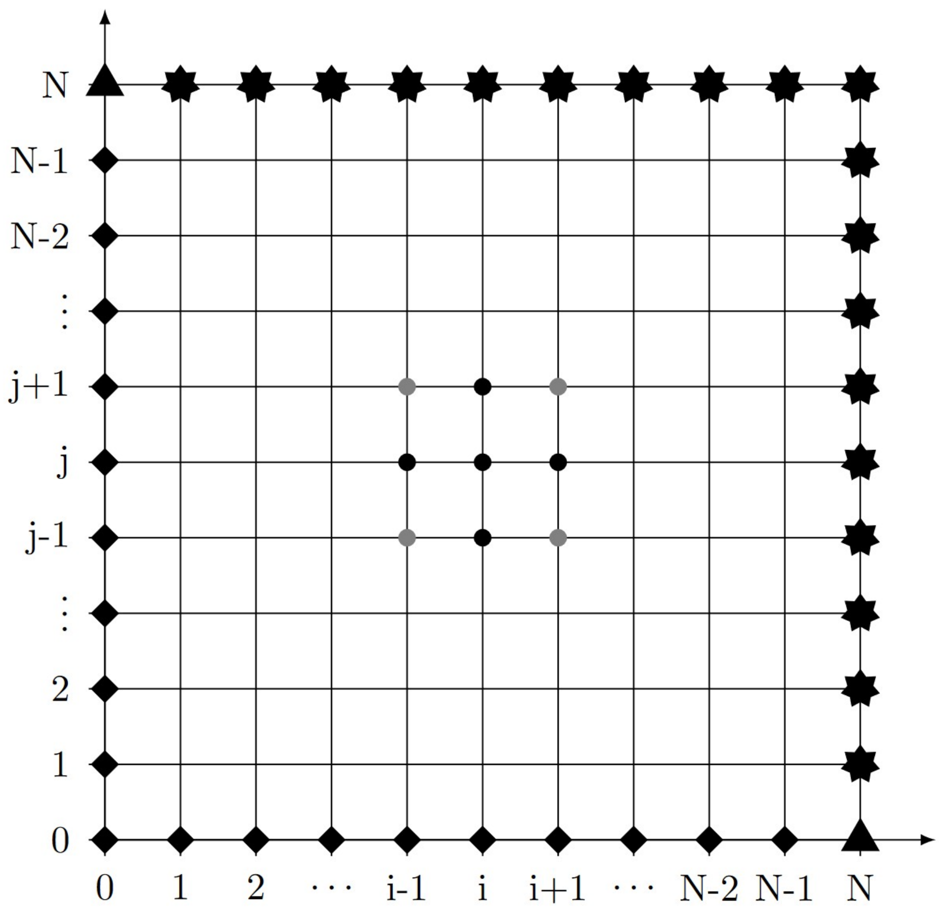


Figure 2.1: Close-Field and Far-Field Boundary Conditions

For the far-field boundaries, Sundvall and Trang explored Dirichlet, linearity condition, and one-sided differences boundary conditions. For example, they derived the Dirichlet boundary condition by extrapolating the 1D case of the limit as the payoff function goes to infinity into 2D as:

$$F(t, x_{max}, y) = \tfrac{x_{\max}+y}{2} - Ke^{-r(T-t)}$$
$$F(t, x, y_{max}) = \tfrac{x+y_{\max}}{2} - Ke^{-r(T-t)} \tag{2.4}$$

The paper notes, as a ”rule of thumb”, to limit the upper domain by setting the maximum value for each asset ($x_{\max}$ and $y_{\max}$ in this case) to be to 4K to 6K times the number of spatial dimensions, which is two.

For the close-field boundaries, Sundvall and Trang explored Dirichlet one-sided differences, linearity condition, and no boundary conditions. For example, for the no boundary condition case, due to $y = 0$ on the x-axis and $x = 0$ on the y-axis, they noted that the x and y axes simplified as:

$$F_t = -rxF_x - \tfrac{1}{2}x^2\Sigma^2(1,1)F_{xx} + rF$$
$$F_t = -ryF_y - \tfrac{1}{2}y^2\Sigma^2(2,2)F_{yy} + rF \tag{2.5}$$

and the value at the origin would thus be $F(t,0,0) = 0$.

Figure 2.2 shows the error results for the Dirichlet boundary coundition for the far-field and no boundary counditions for the close-field. They specifically focused on the region where $\frac{K}{3} \leq x \leq \frac{5K}{3}$ and $\frac{K}{3} \leq y \leq \frac{5K}{3}$

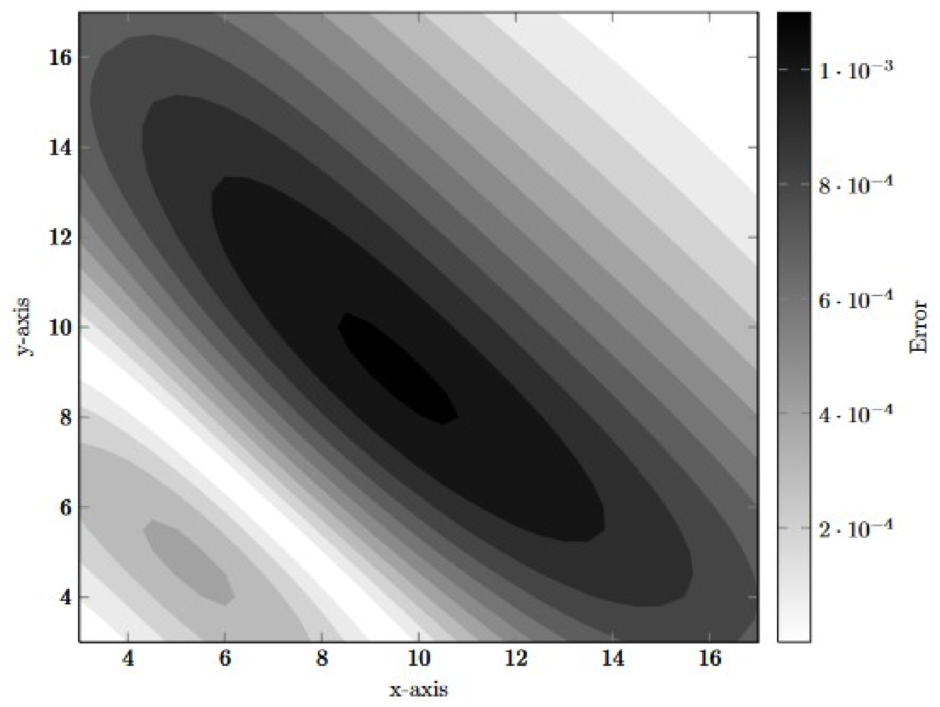


Figure 2.2: Closeup error plot with the far-field Dirichlet boundary condition and no boundary on the close-field

The project will have relevance to the fields of finance and applied mathematics, as well as the growing subjects of Black-Scholes models and mimetic differences.

CHAPTER 3

Solving 2D Black-Scholes Model with Mimetic Methods

3.1 Implementing Mimetic Methods Using the MOLE Library

This section will cover the creation of F_t through the use of mimetic difference functions in the MOLE Library.

3.1.1 Mimetic Difference Functions: G, D, I^{FC}, and I^{CF}

The *grad2D* function in the MOLE library returns a matrix that takes the gradient of a vectorized 2D mesh.

The resulting matrix from *grad2D* is:

$$G = \begin{bmatrix} G_x \\ G_y \end{bmatrix}$$

The application of G to a scalar field of nodal (center) points results in a vector field of discrete edge (face) points. The graph below 3.1 displays an example of the center points (grey) that become face points (red) with the use of G.

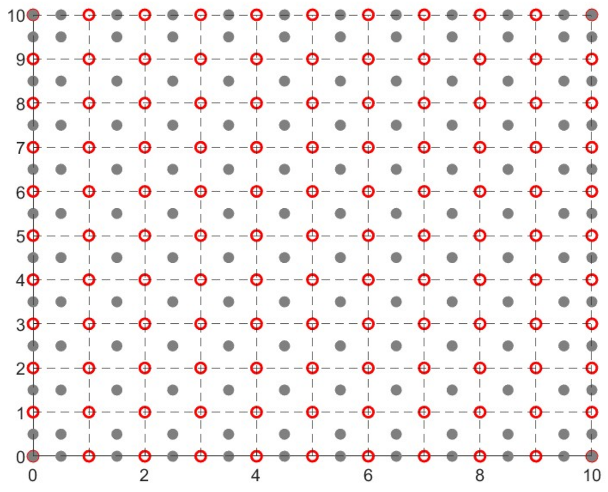


Figure 3.1: Center Points (Grey) and Face Points (Red)

In order to have the face points to return back to the center points for further use, the interpolation function, *interpolFacesToCentersG2D*, is implemented.

The resulting matrix from *interpolFacesToCentersG2D* is:

$$I^{FC} = \begin{bmatrix} I_x^{FC} & \\ & I_y^{FC} \end{bmatrix}$$

$I^{FC}G$ results in a gradient operation on each center point interpolated back to that point. Thus,

$$I^{FC}G = \begin{bmatrix} I_x^{FC} & \\ & I_y^{FC} \end{bmatrix} \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix}$$

Similar to the *grad2D* function, the *div2D* function in the MOLE library returns a matrix that takes the divergence of a vectorized 2D mesh.

The resulting matrix from *div2D* is:

$$D = \begin{bmatrix} D_x & D_y \end{bmatrix}$$

The application of D to a vector field of discrete edge (face) points results in a scalar field of nodal (center) points. However, since the relevant point lay on the center points, the face points must be interpolated to the center points using the function *interpolCentersToFacesD2D*.

The resulting matrix from *interpolCentersToFacesD2D* is:

$$I^{CF} = \begin{bmatrix} I_x^{CF} & \\ & I_y^{CF} \end{bmatrix}$$

DI^{CF} results in a interpolation of center points to face points, which then allow for a divergence operation that then return the face points to center points. Thus,

$$DI^{CF} = \begin{bmatrix} D_x & D_y \end{bmatrix} \begin{bmatrix} I_x^{CF} & \\ & I_y^{CF} \end{bmatrix} = \begin{bmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{bmatrix}$$

3.1.2 Discretization of F_t

The Black-Scholes PDE can be rewritten as

$$F_t = \mathcal{L}F$$

where

$$\mathcal{L} = -rx\frac{\partial}{\partial x} - ry\frac{\partial}{\partial y} - \frac{1}{2}x^2\Sigma_{(1,1)}^2\frac{\partial^2}{\partial x^2} - \frac{1}{2}y^2\Sigma_{(2,2)}^2\frac{\partial^2}{\partial y^2} - xy\Sigma_{(1,2)}^2\frac{\partial^2}{\partial x\partial y} + rI$$

Since $\frac{\partial^2}{\partial x \partial y} = \frac{\partial^2}{\partial y \partial x}$ and $\Sigma_{(1,2)} = \Sigma_{(2,1)}$,

$$\mathcal{L} = -r(x\frac{\partial}{\partial x}+y\frac{\partial}{\partial y})-\frac{1}{2}(x^2\Sigma^2_{(1,1)}\frac{\partial^2}{\partial x^2}+xy\Sigma^2_{(1,2)}\frac{\partial^2}{\partial x \partial y}+yx\Sigma^2_{(2,1)}\frac{\partial^2}{\partial y \partial x}+y^2\Sigma^2_{(2,2)}\frac{\partial^2}{\partial x^2}+xy\Sigma^2_{(1,2)}\frac{\partial^2}{\partial x \partial y})+rI$$

with $x_{\text{diag}} = \text{diag}(x)$, $y_{\text{diag}} = \text{diag}(y)$, and $\mathbf{I} = \begin{bmatrix} I & \\ & I \end{bmatrix}$, further combining and simplifying produces

$$\mathcal{L} = -r \begin{bmatrix} x_{\text{diag}} \\ y_{\text{diag}} \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{bmatrix} - \frac{1}{2} \begin{bmatrix} x\frac{\partial}{\partial x} & y\frac{\partial}{\partial y} \end{bmatrix} \begin{bmatrix} \Sigma^2_{(1,2)}I & \Sigma^2_{(1,2)}I \\ \Sigma^2_{(2,1)}I & \Sigma^2_{(2,2)}I \end{bmatrix} \begin{bmatrix} x\frac{\partial}{\partial x} \\ y\frac{\partial}{\partial y} \end{bmatrix} + r\mathbf{I}$$

$$\mathcal{L} = -r \begin{bmatrix} x_{\text{diag}} \\ y_{\text{diag}} \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{bmatrix} - \frac{1}{2} \begin{bmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{bmatrix} \begin{bmatrix} x_{\text{diag}} & \\ & y_{\text{diag}} \end{bmatrix} \begin{bmatrix} \Sigma^2_{(1,2)}I & \Sigma^2_{(1,2)}I \\ \Sigma^2_{(2,1)}I & \Sigma^2_{(2,2)}I \end{bmatrix} \begin{bmatrix} x_{\text{diag}} & \\ & y_{\text{diag}} \end{bmatrix} \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix} + r\mathbf{I}$$

substituting $I^{FC}G = \begin{bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{bmatrix}$ and $DI^{CF} = \begin{bmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{bmatrix}$

$$\mathcal{L} = -r \begin{bmatrix} x_{\text{diag}} \\ y_{\text{diag}} \end{bmatrix} I^{FC}G - \frac{1}{2} DI^{CF} \begin{bmatrix} x_{\text{diag}} & \\ & y_{\text{diag}} \end{bmatrix} \begin{bmatrix} \Sigma^2_{(1,2)}I & \Sigma^2_{(1,2)}I \\ \Sigma^2_{(2,1)}I & \Sigma^2_{(2,2)}I \end{bmatrix} \begin{bmatrix} x_{\text{diag}} & \\ & y_{\text{diag}} \end{bmatrix} I^{FC}G + r\mathbf{I} \tag{3.1}$$

3.1.3 Close-Field Boundary Condition

As described earlier [2.5], the close-field boundary conditions operate where $x = 0$ and $y = 0$, thus simplifying each equation into a one-dimensional version of the Black-Scholes Model for each asset respectively.

Similar to the previous section, we can rewrite each 1D boundary condition as...

$$F_t = \mathcal{L}_x F, \quad y = 0$$

$$F_t = \mathcal{L}_y F, \quad x = 0$$

where

$$\mathcal{L}_x = -rx\frac{\partial}{\partial x} - \tfrac{1}{2}x^2\Sigma^2_{(1,1)}\frac{\partial^2}{\partial x^2} + r$$

$$\mathcal{L}_y = -ry\frac{\partial}{\partial y} - \tfrac{1}{2}y^2\Sigma^2_{(2,2)}\frac{\partial^2}{\partial y^2} + r$$

To find $\mathcal{L}_x$ and $\mathcal{L}_y$, we much implement the following MOLE Library functions:

1. *grad*: Applies a gradient operation to a 1D scalar array of nodal (center) points which results in a 1D vector array of discrete edge points.
2. *interpolFacesToCentersG1D*: Interpolates the result of *grad* back to the 1D scalar array of nodal (center) points
3. *lap*: Applies a laplacian operation to a 1D scalar array of nodal (center) points which results in a 1D scalar array of nodal (center) points. No interpolation is needed.

With

1. the 1D gradients $grad = G_x = G_y$,
2. the 1D interpolations $interpolFacesToCentersG1D = I_x^{FC} = I_y^{FC}$, and
3. the 1D Laplacians $lap = \nabla_x^2 = \nabla_y^2$,

$$I^{FC}G = \frac{\partial}{\partial x} = \frac{\partial}{\partial y}$$

$$\nabla^2 = \frac{\partial^2}{\partial x^2} = \frac{\partial^2}{\partial y^2}$$

and with the diagonal matrices $x_{\text{diag}} = \text{diag}(x)$ and $y_{\text{diag}} = \text{diag}(y)$

$$\mathcal{L}_x = -rx_{\text{diag}}I_x^{FC}G_x - \frac{1}{2}x_{\text{diag}}^2\Sigma_{(1,1)}^2\nabla_x^2 + rI$$

$$\mathcal{L}_y = -ry_{\text{diag}}I_y^{FC}G_y - \frac{1}{2}y_{\text{diag}}^2\Sigma_{(2,2)}^2\nabla_y^2 + rI$$

3.1.4 Far-Field Boundary Condition

The far-field boundary conditions [2.4] are of Dirichlet type and can be implemented directly on the mesh grid.

3.2 Discretization of Time

As the value of the option is discounted in time back to the present, we must note that time starts at $t = T$ and ends at $t = 0$. This discounting means that the index n represents $T - n\,\Delta t$, thus $n + 1$ is a step backwards in time from n.

We use the implicit Backward Differentiation Formula of second-order (BDF2), with A the mimetic difference discrete analog of operator $\mathcal{L}$. One can think of the implicit BDF2 used as the Forward Differentiation Formula of second-order but with Δt being negative due to discounting. The BDF2 results in the following discretization of F_t:

$$AF_{n+1} + b = \frac{F_n - F_{n+1}}{\Delta t} - \frac{1}{2}\frac{F_{n+1} - 2F_n + F_{n-1}}{\Delta t}$$

where b is the Dirichlet far-boundary condition. However, we will ignore b as the boundary condition can just be imposed and saved onto the resulting F_{n+1}.

$$\Delta t A F_{n+1} = F_n - F_{n+1} - \tfrac{1}{2}F_{n+1} + F_n - \tfrac{1}{2}F_{n-1}$$

$$\tfrac{3}{2}F_{n+1} + \Delta t A F_{n+1} = 2F_n - \tfrac{1}{2}F_{n-1}$$

Finally, leaving

$$\left(I + \frac{2}{3}\Delta t A\right) F_{n+1} = -\frac{4}{3}F_n + \frac{1}{3}F_{n-1} \qquad (3.2)$$

which requires information from the previous two time step. To start the backward discrete evolution, we utilize the BDF1 method as suggested in [?], which reads as

$$(I + \Delta t A)F_1 = F_0$$

3.3 Parameters

As used in the paper by Sundvall and Trång the project, I will be using the parameters:

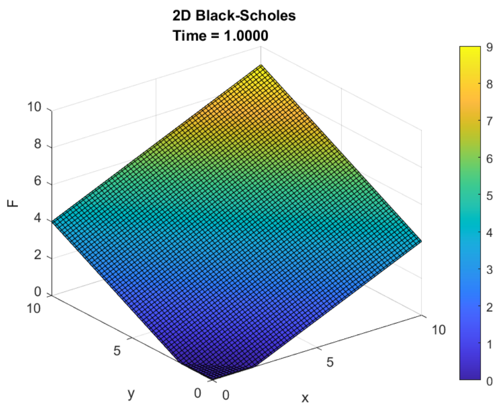
- $r = 0.1$
- $\Sigma = \begin{bmatrix} 0.3 & 0.05 \\ 0.05 & 0.3 \end{bmatrix}$

Being undefined in the paper, I will also be using the parameters:

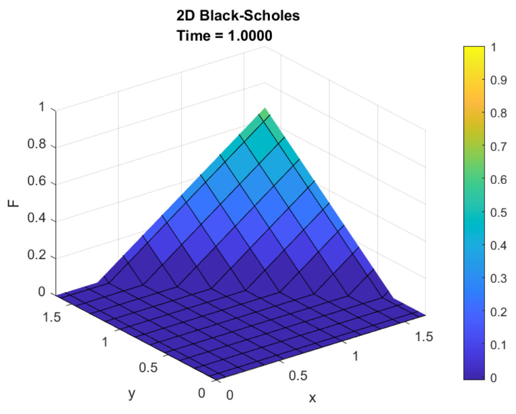
- $T = 1$
- $K = 1$

3.4 Results

The plots of the initial conditions [2.3] for the entire field and the field of interest ($\left[0 \leq x \leq \frac{5K}{3}\right]$ and $\left[0 \leq y \leq \frac{5K}{3}\right]$) are



(a) Entire Field



(b) Field of Interest

Figure 3.2: Initial Conditions

The plot of the results for the entire field using mimetic methods is

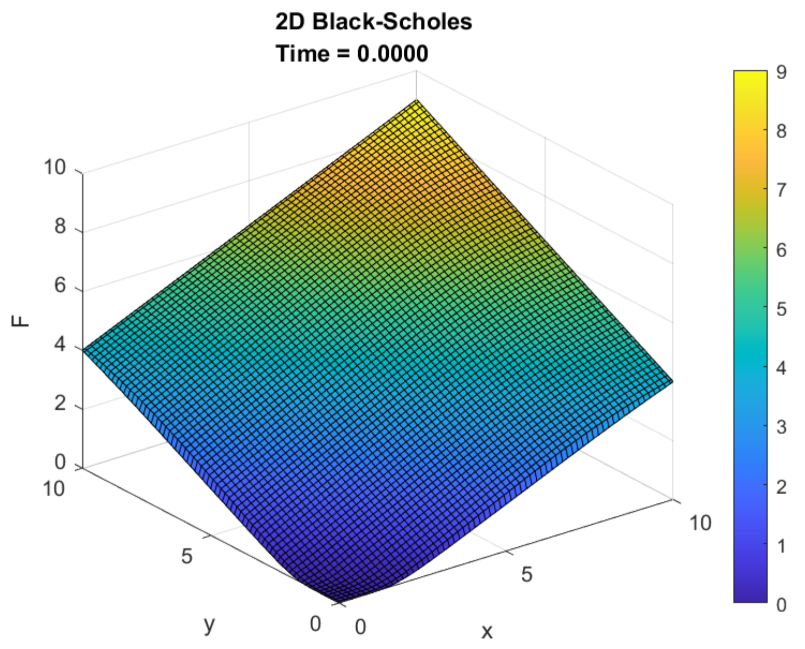


Figure 3.3: Results: Entire Field

and the plot of the results in the region of interest is

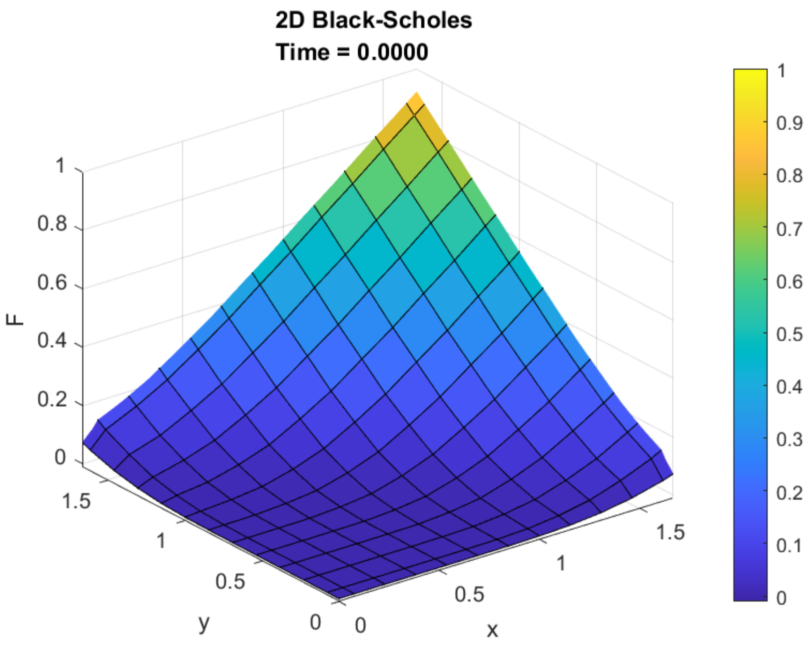


Figure 3.4: Results: Field of Interest

CHAPTER 4
Radial Basis Function Model (RBF-PU)

In the paper, Shcherbakov and Larsson use radial basis functions (RBF), specifically the partition of unity (PU) method, to solve the Black-Scholes model in two dimensions for American call options. The 2D RBF-PU model is implemented in the MATLAB file $RBFPUMamCallpenalty2D.m$

4.1 Modifications to 2D PBF-PU Model in $RBFPUMamCallpenalty2D.m$

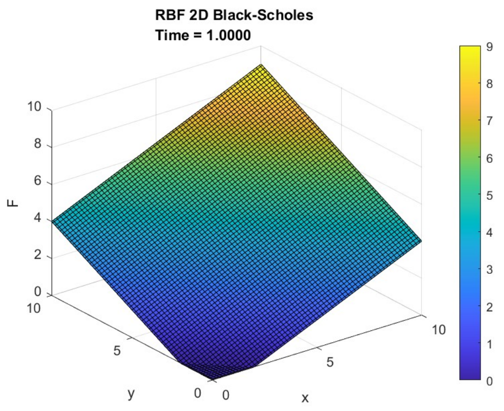
Since American call options can be exercised at any date before the maturity time, Shcherbakov and Larsson follow the convention of applying a penality parameter, Q ,to the result each timestep in the model. They also include a parameter, d_i, subtracted from r representing the continuous dividend yield of the ith asset. We can just set both of these to zero to have the model represent a European call option with no dividend.

The paper and MATLAB file uses a uniform grid for the RBF-PU model so, in order to more easily compare to the Mimetic Differences model, a staggard grid is used instead. RBF-PU model are able to work with non-uniform grids so a staggard grid will not impact the results.

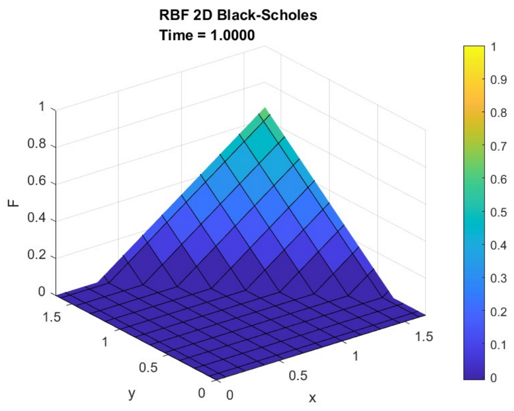
The Dirichlet far-boundary conditions [2.4] in the Mimetic Differences model are also applied to the RBF-PU model at each timestep. The close-field boundaries were not able to be implemented which might minimally skew the results.

4.2 Results of 2D RBF-PU

The plots of the initial conditions [2.3] for the entire field and the field of interest ($\left[0 \leq x \leq \frac{5K}{3}\right]$ and $\left[0 \leq y \leq \frac{5K}{3}\right]$) are



(a) Entire Field



(b) Field of Interest

Figure 4.1: Initial Conditions

Applying the modifications described in the previous section and using the same parameters as before produce the resulting plots

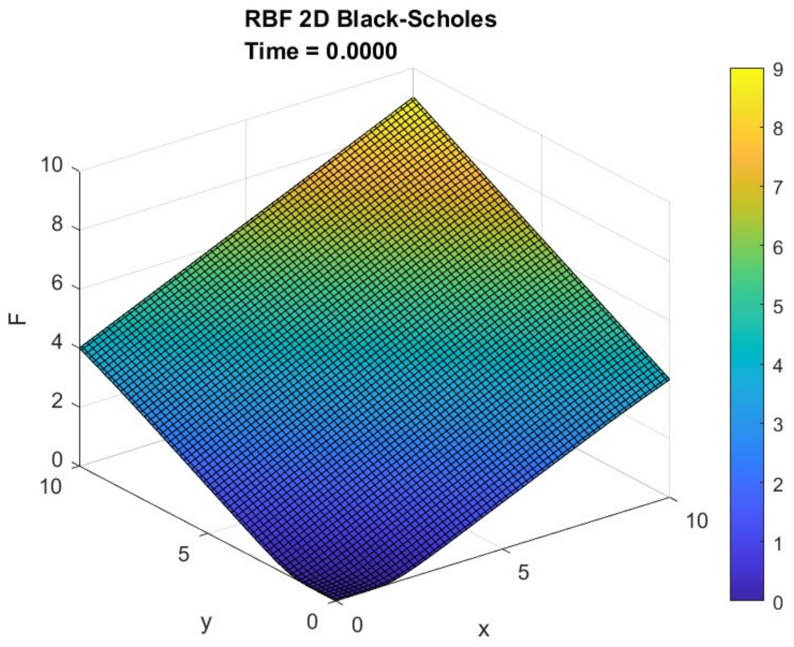


Figure 4.2: Results for 2D RBF-PU: Entire Field

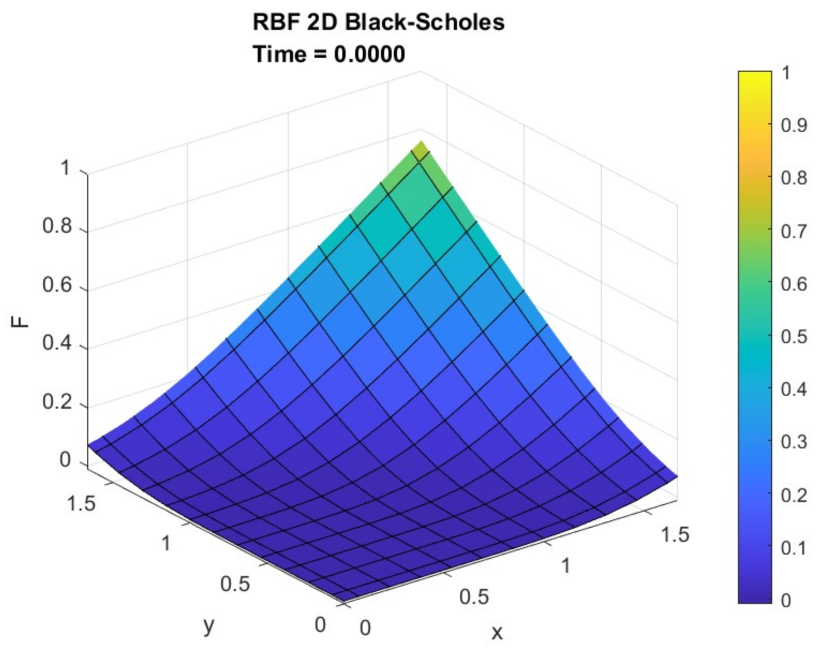


Figure 4.3: Results for 2D RBF-PU: Field of Interest

4.3 Comparison to Mimetic Difference Model

The following plots display the absolute difference between the Mimetic Difference and RBF-PU models

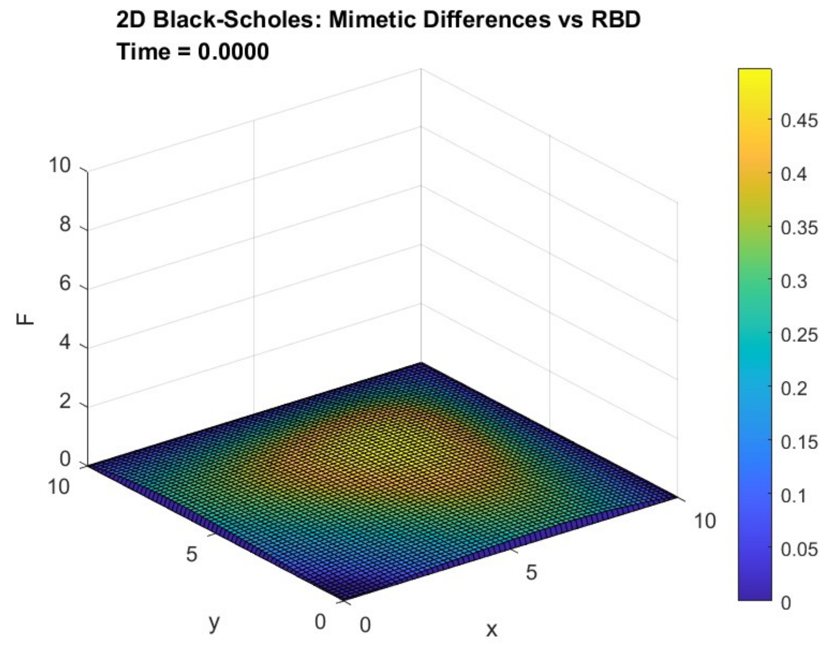


Figure 4.4: Mimetic Differences vs RBF-PU Models: Entire Field

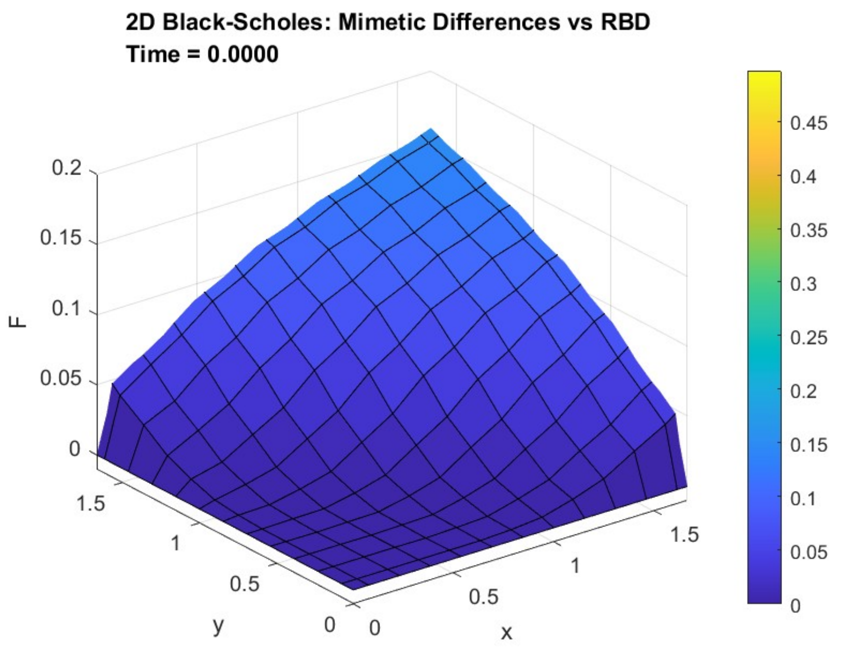


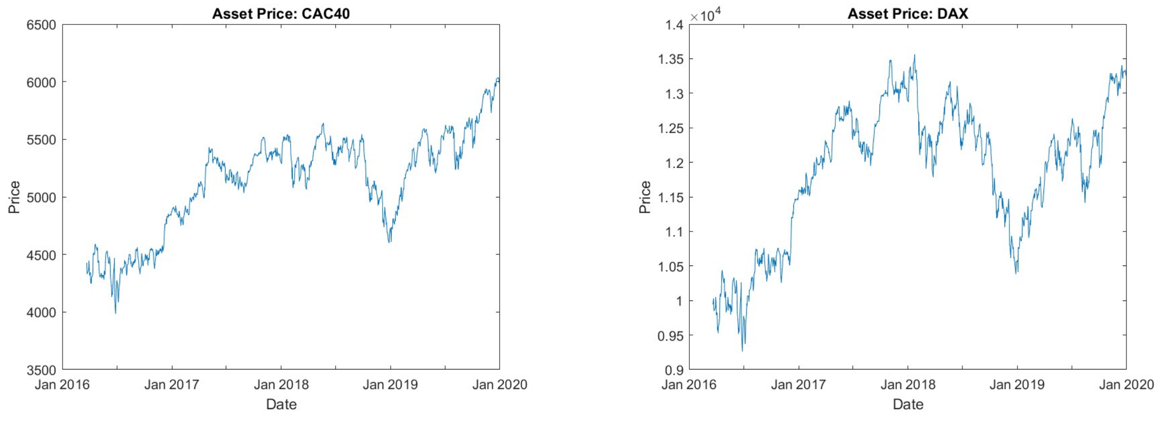
Figure 4.5: Mimetic Differences vs RBF-PU Models: Field of Interest

CHAPTER 5
Comparison to Real Data

When comparing a multivariate model to real data, knowing the value of call option at each possible price combination of assets is unfeasible. However, by analyzing historical price data, we can estimate the parameters needed to make a forcasting model for our mimetic difference model. We can then start a simulation starting at the first date with data recorded and track how that compares to how the value of the real assests moved during that time.

5.1 Data

The two assets used to compare to the mimetic difference model are the daily index price data of CAC40 (France) and of the DAX (Germany) from March 2016 to December 2019.



(a) CAC40 (b) DAX

Figure 5.1: Index Price (March 2016 to December 2019)

To normalize the data, the daily log returns were used instead of nominal prices. The daily log returns were calculated by

$$r_t = \log\left(\frac{P_t}{P_{t-1}}\right)$$

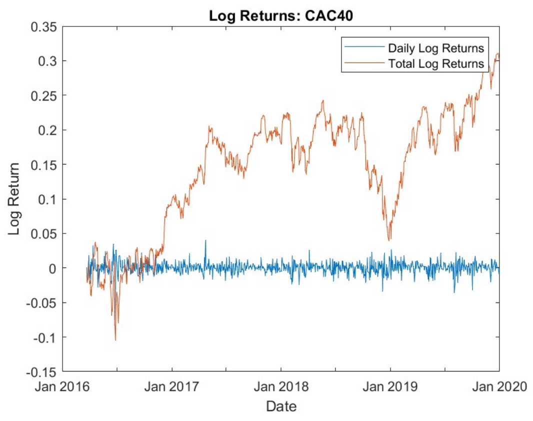
where P_t is the price of the index at time, t, with $t-1$ being the trading day beforehand.

Total log returns were calculated by

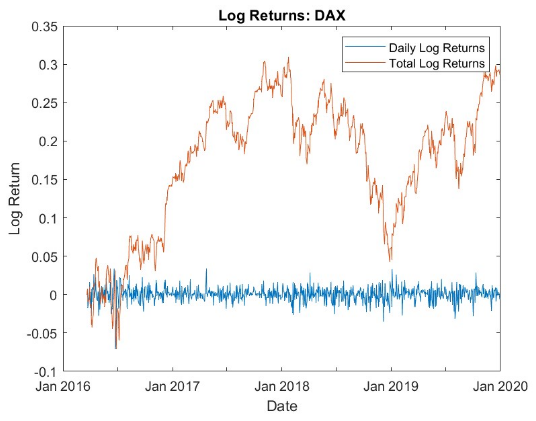
$$r_t = \log\left(\frac{P_t}{P_0}\right)$$

where P_0 is the initial price of the index.

The daily and total log returns for both indexes are displayed below



(a) CAC40



(b) DAX

Figure 5.2: Index Daily and Total Log Returns (March 2016 to December 2019)

5.2 Parameters

The following parameters will be used for the

- $r = 0.1$ (assumed risk-free interest rate)
- $T = 3.7728$ (March 2016 to December 2019 in terms of years)
- $K = 1$

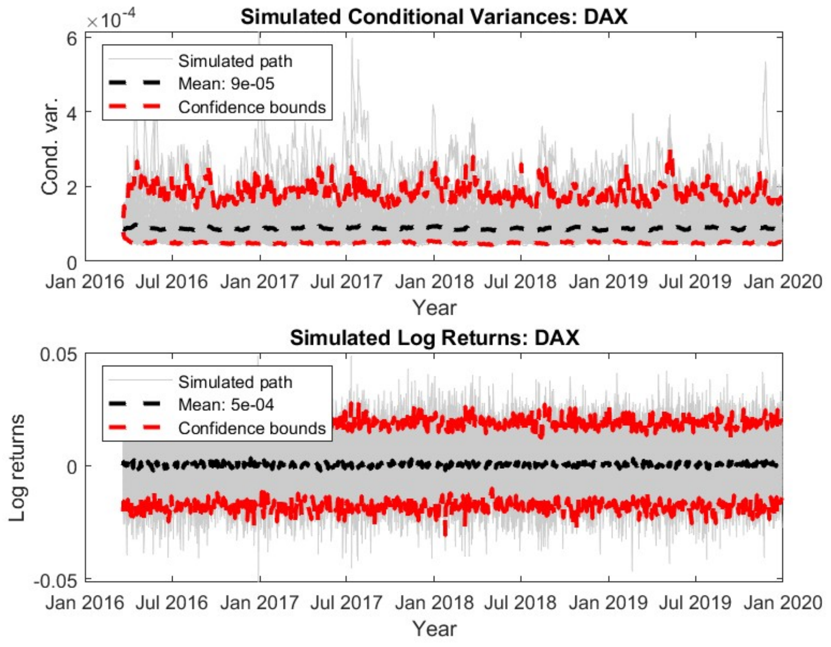
5.3 Volatility and Correlation

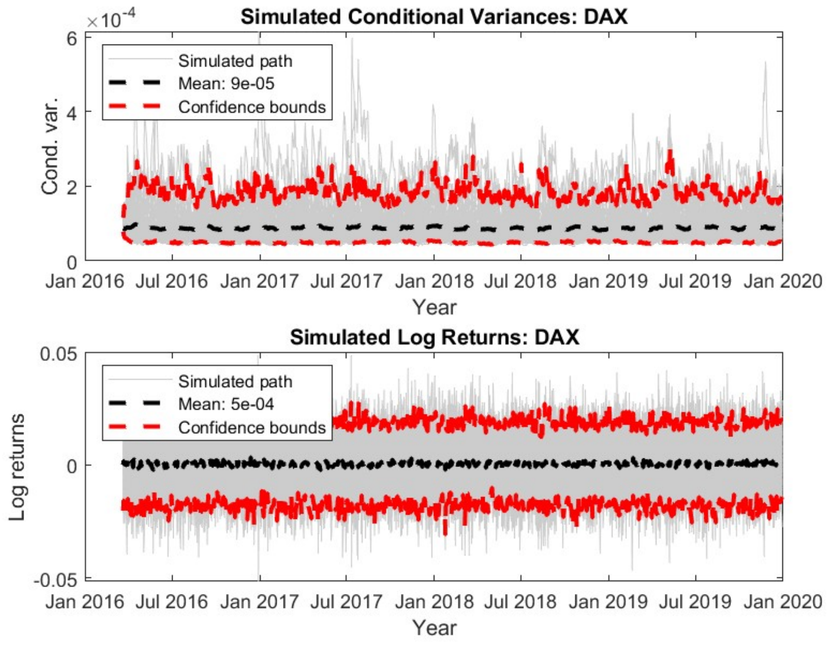
As shown in 2, the correlation volatility matrix is comprised of volatility coefficients (σ) for each asset and a correlation coefficient (ρ) relating both assets.

5.4 Volatility and Correlation Calculations

To calculate volatility, the multivariate GARCH method will be used. The GARCH method uses a ordinary least squares analysis of the daily log returns and runs multiple simulations to produce an approximation for the daily volatility σ^* for each asset.

The following graphs show the simulations for both CAC40 and DAX

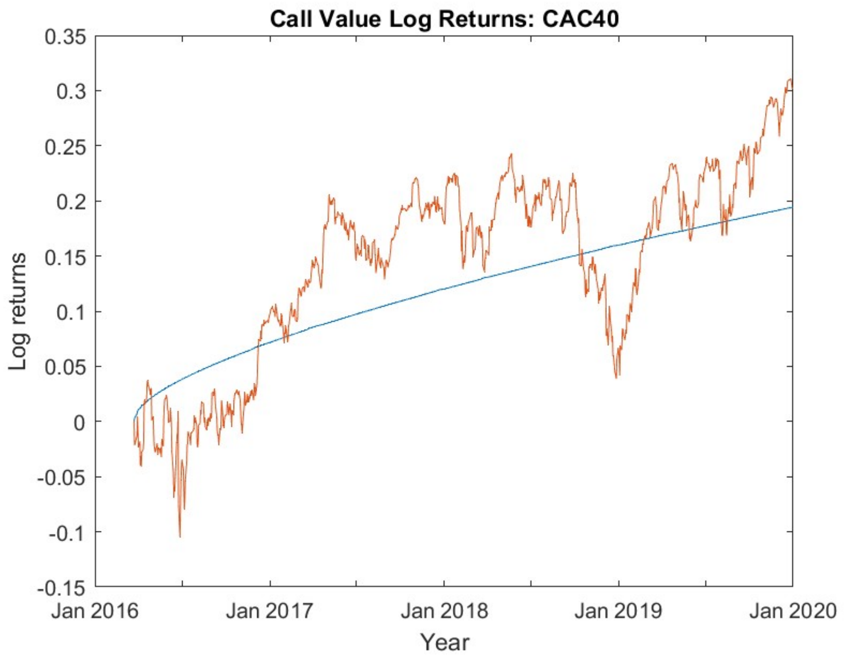


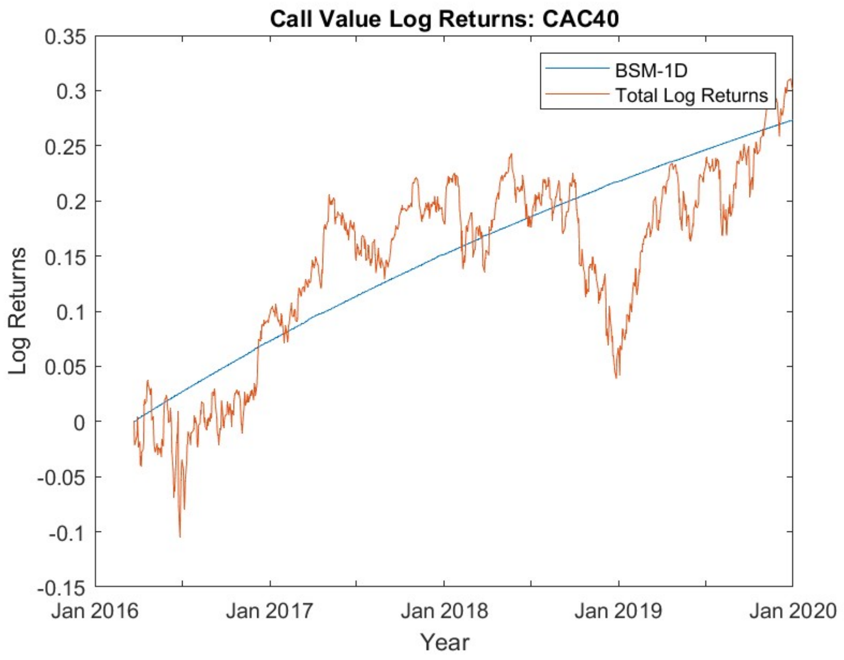


To ease calculation, we will assume that both volatility and correlation will remain constant over time. Thus, we can take the mean of the daily conditional volatility. Note that after taking the mean daily conditional volatility, we will need to adjust for the length of time the model will be running by multiplying by the square root of the number of trading days in that time period. There are about 252 trading days per year.

$$\sigma_i = \sigma_i^* \sqrt{252T}$$

After getting the value of the mean daily volatility, we can check its accuracy by plotting a simulated 1D Black-Scholes model using the σ value against the total log returns shown below





Correlation is just the correlation between DAX and CAC40's log returns. In our case, this ended up being quite close to 1.

Combining together, we get

$$\Sigma = \begin{bmatrix} 0.0029 & 0.0027 \\ 0.0027 & 0.0027 \end{bmatrix}$$

5.5 Results

Using the parameters and same initial conditions, we can use the estimated Σ to put into the 2D Black-Scholes model. We can then compare the Mimetic Differences model result to the simulated 1D version of the Black-Scholes model for each asset.

I have chosen to compare the points where the initial condition $\Phi(x, y) = 0$ rather than expanding the field as we are only comparing the log returns. The values end up being normalized.

The results are shown below

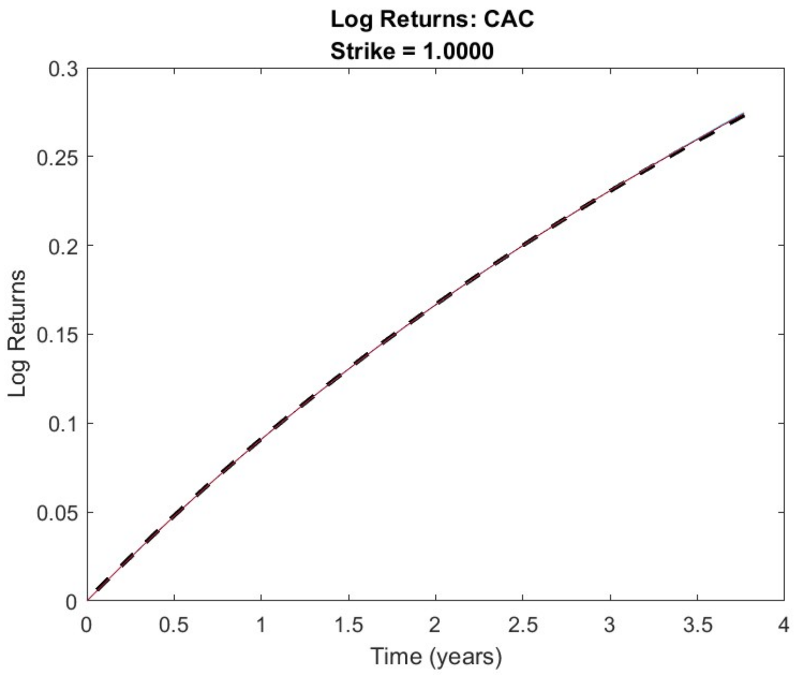


Figure 5.3: Call Option Value (Black) vs Mimetic Difference Model: CAC40

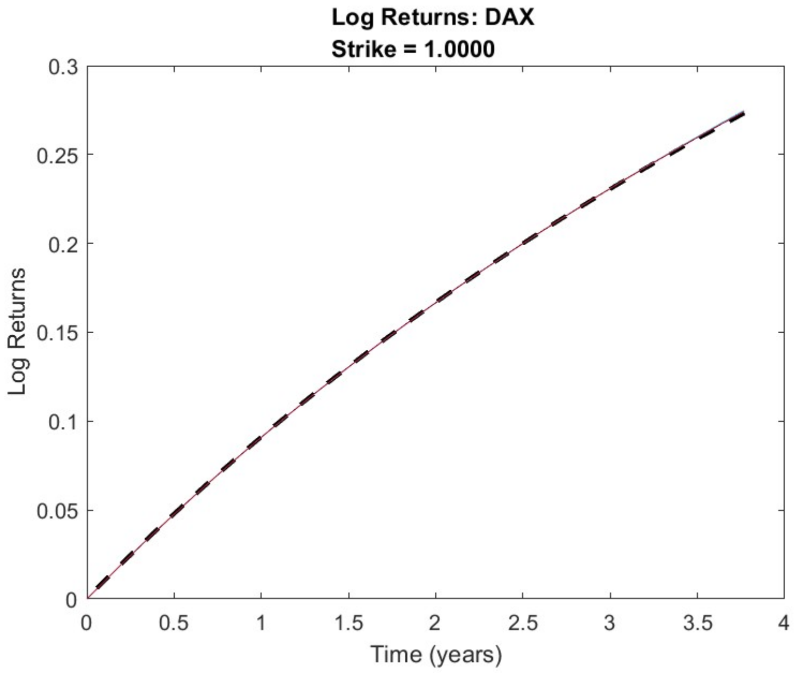


Figure 5.4: Call Option Value (Black) vs Mimetic Difference Model: DAX

CHAPTER 6
N-Dimensional Mimetic Model

The following code is a template to expand the boundary-less Black-Scholes model using mimetic methods. The only missing parts are the flexible mimetic operators that work for multiple dimensions and the boundary conditions, which can be expanded upon in the future.

```
%% N-D Black-Scholes PDE
% Zachary Humphries

clear all;
close all;
clc;


%% Parameters
T = 1;                                          % Time
strike = 1;                                     % Strike Price
r = 0.1;                                        % Risk free interest rate

k = 4;                                          % Order of accuracy

%% Omega Matrix (must be square and diagonal)

omega = [0.1 0.1 0.1;
     0.1 0.1 0.1;
     0.1 0.1 0.1];

%% Do not edit code below

%% Set up Dimensions
dimensions = length(omega);
n = (2*k)^2 +1;                                 % 2*k+1 = Minimum number of
    cells to attain the desired accuracy

n = 40;

minValue = 0;
maxValue = 5*strike*dimensions;
```

```
dn = (maxValue-minValue)/n;

grid1D = [minValue minValue+dn/2 : dn : maxValue-dn/2 maxValue];

NdArrays = cell(dimensions, 1);  %create cell array to receive outputs of
    ndgrid

[NdArrays{:}] = ndgrid(grid1D);  %fills all N Nd arrays

%% Time

dt = 0.05*(dimensions*(1/dn)^2)^(-1);        % Von Neumann stability
    criterion for explicit scheme dx^2/(4*alpha)
dt = -dt;

%% Initial Conditions

Fzeros = zeros(size(NdArrays{1}));

Fsum = Fzeros;
for ii = 1:dimensions

    Fsum = NdArrays{ii} + Fsum;

end

F = max((Fsum./dimensions) - strike, Fzeros);
Fvec = reshape(F, (n+2)^dimensions, 1);

NdVec = cell(dimensions, 1);

for ii = 1:dimensions

    NdVec{ii} = reshape(NdArrays{ii}, (n+2)^dimensions, 1);

end

%% Plot Initial Conditions

plotBlackScholes(Fvec, size(F), NdArrays, n, T, minValue, maxValue, strike)
    ;

%% operator maticies
```

```
small_I = speye((n+2)^(dimensions), (n+2)^(dimensions));
I = speye(dimensions*((n+2)^(dimensions)), dimensions*((n+2)^(dimensions)))
    ;

NdI = cell(dimensions, 1);
for ii = 1:dimensions

    NdI{ii} = NdVec{ii} .* small_I; % length is (n+2)^(dimensions)

end

NDI = [cat(1, NdVec{:})] .* I;  % length is dimensions*(n+2)^(dimensions)

ND = [NdI{:}];

%O = sparse([], length(omega)*length(small_I), length(omega)*length(small_I
    ));
O = kron(omega.*omega, small_I);

%% Setting Up Gradient and Divergence Matricies

% G = grad2D(k, m, dx, n, dy);
% D = div2D(k, m, dx, n, dy);

G = [];
D = [];

%% Setting Up Interpolation Matrices

% IFC = interpolFacesToCentersG2D(k, m, n);
% ICF = interpolCentersToFacesD2D(k, m, n);

IFC = [];
ICF = [];

%% Combine for Black-Scholes Matrix

A1 = -(r*ND*IFC*G);
A2 = -0.5*(D*ICF*NDI*O*NDI*IFC*G);
A3 = r*small_I;

A = A1+A2+A3;


```

```
%% auxiliary variables
len = length(T : dt : 0);
Fsol = zeros(numel(Fvec), len); % to store solutions
Fsol(:,1) = Fvec;

Faux = Fvec; % to jump start time discretization

%% Forward Euler (First Order)

Faux = (small_I - dt*A)\(Fsol(:,1));

Faux(1) = 0;

Fsol(:,2) = Faux;

%% Forward Euler
count = 2;
for t = T+(2*dt) : dt : 0

    Faux = (small_I - ((2*dt)/3)*A)\((4/3)*Fsol(:,count) - (1/3)*Fsol(:,
    count-1));


%    plotBlackScholes(Faux, X, Y, m, n, t, a, b, c, d, strike);

    count = count+1;
    Fsol(:,count) = Faux;

end

figure(2)
plotBlackScholes(Faux, X, Y, m, n, 0, a, b, c, d, strike);


function plotBlackScholes(Fvec, sizeFvec, NdArrays, n, t, minValue,
    maxValue, strike)
    F = reshape(Fvec, sizeFvec);

    tempNdArrays = cell(length(NdArrays)-1, 1);

    sliderValues = [NdArrays{3}(1,1,:)];

    Fslice = squeeze(F(:,:, 1));

    for ii = 1:(length(NdArrays)-1)
        tempNdGrid = NdArrays{ii};
        tempNdArrays{ii} = squeeze(tempNdGrid(:,:, 1));
    end

    currFig = surf(tempNdArrays{1}, tempNdArrays{2}, Fslice);
%     set(currFig, 'title', ['2D Black-Scholes \newlineTime = ' num2str(t,
    '%1.4f')],...
%         'xlabel', 'x', 'ylabel', 'y', 'zlabel', 'F', 'axis', [minValue
    maxValue minValue maxValue])
    title(['2D Black-Scholes \newlineTime = ' num2str(t, '%1.2f') ', z = '
    num2str(sliderValues(1), '%2.2f')]);
    xlabel('x');
    ylabel('y');
    zlabel('F');
    colorbar;
    caxis([minValue maxValue])
    axis([minValue maxValue minValue maxValue minValue maxValue]);
    drawnow

    uicontrol('Style', 'slider', 'Min', 1, 'Max', length(sliderValues), ...
        'Value', 1, 'Position', [400 20 130 20], ...
        'Callback', @react_to_slider);

    drawnow

        function react_to_slider(source, event)   %nested !!
            val = round(get(source, 'Value'));
            if val >= length(sliderValues)
                val = length(sliderValues);
            end
            set(source, 'Value', val);
            plotGraph(val)


            function plotGraph(val)
                Fslice = squeeze(F(:,:, val));

                for jj = 1:(length(NdArrays)-1)
                    tempNdGrid = NdArrays{jj};
                    tempNdArrays{jj} = squeeze(tempNdGrid(:,:, val));
                end

```

```
                    set(currFig, 'XData', tempNdArrays{1}, 'YData',
    tempNdArrays{2}, 'ZData', Fslice);
                    title(['2D Black-Scholes \newlineTime = ' num2str(t, '%1.2f
    ') ', z = ' num2str(sliderValues(val), '%2.2f')]);
                    drawnow

               end

          end

end
```

N-D Black-Schole Mimetic Differences Model

CHAPTER 7

Conclusion and Personal Reflection

The Black-Scholes PDE is quite versitile in the ways it can be approximated. In future updates to this project, I would recommend the exploration of different boundary conditions on the PDE, the creation of a better means of comparison to real world data, and the addition of the N-D mimetic operators to the N-D Black-Scholes model.

I am grateful to my committee for their patience with this project, as well as, my friends and family for their love and support throughout graduate school.

BIBLIOGRAPHY

[1] J. Corbino, *Mimetic operators library enhanced (mole).* https://github.com/csrc-sdsu/mole.

[2] J. Corbino and J. Castillo, *High-order mimetic finite-difference operators satisfying the extended gauss divergence theorem*, Journal of Computational and Applied Mathematics, (2020).

[3] R. K. R. L. J. Matti and H. Gong, *Estimating the volatility in the black-scholes formula*, (2015).

[4] Z. B. R. C. Merton and D. L. Cleeton, *Financial Economics*, vol. 2, Pearson Education, Inc, Upper Saddle River, New Jersey, 2006.

[5] S. Milovanovic, *Radial basis functions generated finite differences to solve high-dimensional pdes in finance*, (2014).

[6] V. Shcherbakov and E. Larsson, *Radial basis function partition of unity methods for pricing vanilla basket options*, (2018).

[7] T. Sundvall and D. Trång, *Examination of impact from different boundary conditions on the 2d black-scholes model: Evaluating pricing of european call options*, (2014).