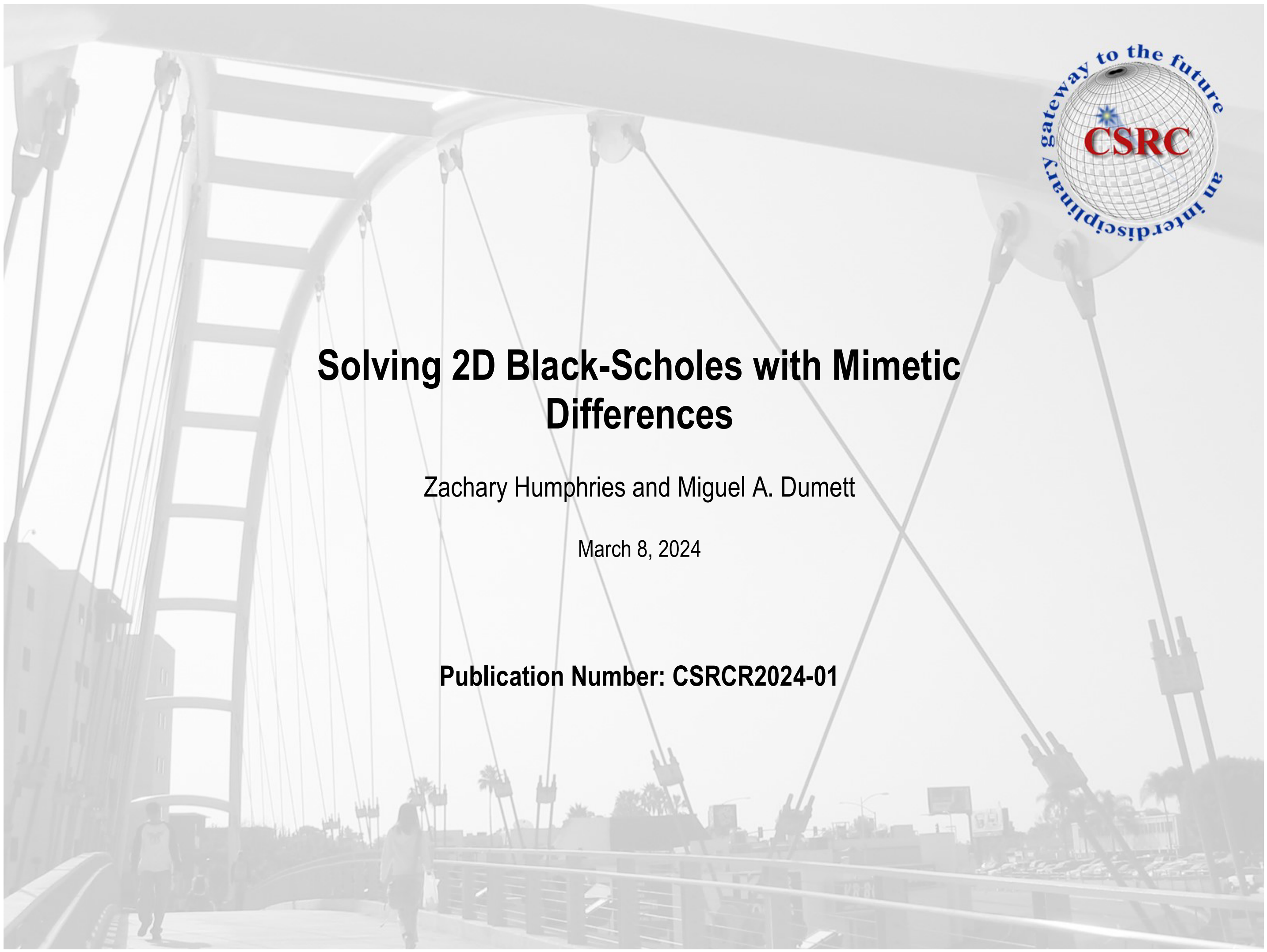


Solving 2D Black-Scholes with Mimetic Differences

Zachary Humphries and Miguel A. Dumett

March 8, 2024

Publication Number: CSRCR2024-01

Solving 2D Black-Scholes with Mimetic Differences

Zachary Humphries [*] and Miguel A. Dumett [†‡]

March 8, 2024

Abstract

The Black-Scholes equation is a mathematical model from which one can obtain a theoretical estimate of the price of European-style options. In this paper, mimetic differences are utilized to numerically solve the 2D version of it.

1 Introduction

Mimetic differences [2], a method for numerically solving PDEs via matrix representations of differential operators of vector calculus, provide a more user-friendly experience for constructing numerical schemes for PDEs. Instead of needing to create custom functions that return matrices for the specific differential equation, the MOLE library [3] uses mimetic operators to create matrices for common differential operators: gradient, divergence, and Laplacian. By inputting the dimensions of the scheme and the desired order of accuracy into one of the mimetic MOLE functions, the user receives a matrix for the difference operations. Not having to construct the matrices by hand greatly reduced the time invested in obtaining the corresponding numerical estimates.

However, there are a few shortcomings of mimetic operators. Mimetic differences require a general understanding of the linear transformations represented and how each operation affects the movement of points on the grid. Although the MOLE library functions provide flexibility for the orders of accuracy, interpolation is often needed to return nodal points back to their original position. One of the unexpected difficulties while trying to construct a discrete analog of a second-order mixed derivative operator was that it required interpolations in both the X and Y directions. The MOLE library does come with its own interpolation functions, which eases the difficulty with interpolation. Once the user overcomes the initial learning curve, mimetic operators add a powerful and flexible tool for programmers to obtain numerical solutions of PDE models.

[*] Computational Science Master Program at San Diego State University (zhumphries0117@sdsu.edu).
[†] Jose E. Castillo:Editor
[‡] Computational Science Research Center at San Diego State University (mdumett@sdsu.edu).

2 Background

The Black-Scholes Model [1] is a partial differential equation (PDE) developed by Fisher Black and Myron Scholes 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 specific time. Unlike American options, European options can only be exercised at the maturity date.

The motivation to deal with this PDE was the experience the author had with the equation while working as a pricing analyst in the natural gas industry, but never really understood the equation. The Risk Management team that worked alongside the Pricing team would mitigate risk (hedge) by buying call options on future consumption of natural gas by customers, whose contracts were needed to price.

The research paper *Examination of Impact from Different Boundary Conditions on the 2D Black-Scholes Model: Evaluating Pricing of European Call Options* by Tomas Sundvall and David Trång[1] is utilized to obtain the parameter values, boundary and initial condition for our investigation.

3 Black-Scholes PDE

The Black-Scholes PDE is given by:

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

The payoff function at the maturity time is:

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

The parameters in the equation are:

- r : Risk-free Investment (often US bonds)
- σ : Volitility Correlation Matrix

$$\sigma = \begin{pmatrix} \sigma_{xx} & \sigma_{yx} \\ \sigma_{xy} & \sigma_{yy} \end{pmatrix}, \qquad \sigma_{yx} = \sigma_{x)}, \qquad \sigma_{xx} = \sigma_{yy}$$

- T : Final Maturity Time
- K : Strike Price (Call Option Premium)

3.1 Parameters

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

- $r = 0.1$
- $\sigma_{(1,1)} = 0.3$
- $\sigma_{(1,2)} = 0.05$

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

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

3.2 Initial Conditions

Since the value of the option is discounted to the present, the initial condition is the payoff function:

$$F(T, x, y) = \Phi(x, y) = \left(\frac{x+y}{2} - K\right)^{+} \tag{3}$$

3.3 Dirichlet Boundary Conditions

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

In Figure 1, close-field (as diamonds) and far-field (as stars) boundary conditions are exhibited. The triangular points where the close-field boundary conditions and meet the far-field boundary conditions can be either of the two, however, in this project, they are defined as far-field boundary conditions. The close-field and far-field boundary conditions will be updated with each time step.

3.4 Close-Field Boundary Conditions

Since $y = 0$ on the x-axis and $x = 0$ on the y-axis, the x and y axes Black-Scholes equation reduces to:

$$\begin{aligned} F_t &= -rxF_x - \tfrac{1}{2}x^2\sigma^2_{(1,1)}F_{xx} + rF, \quad y = 0 \\ F_t &= -ryF_y - \tfrac{1}{2}y^2\sigma^2_{(2,2)}F_{yy} + rF, \quad x = 0 \end{aligned} \tag{4}$$

The value at the origin $F(t, 0, 0) = 0$.

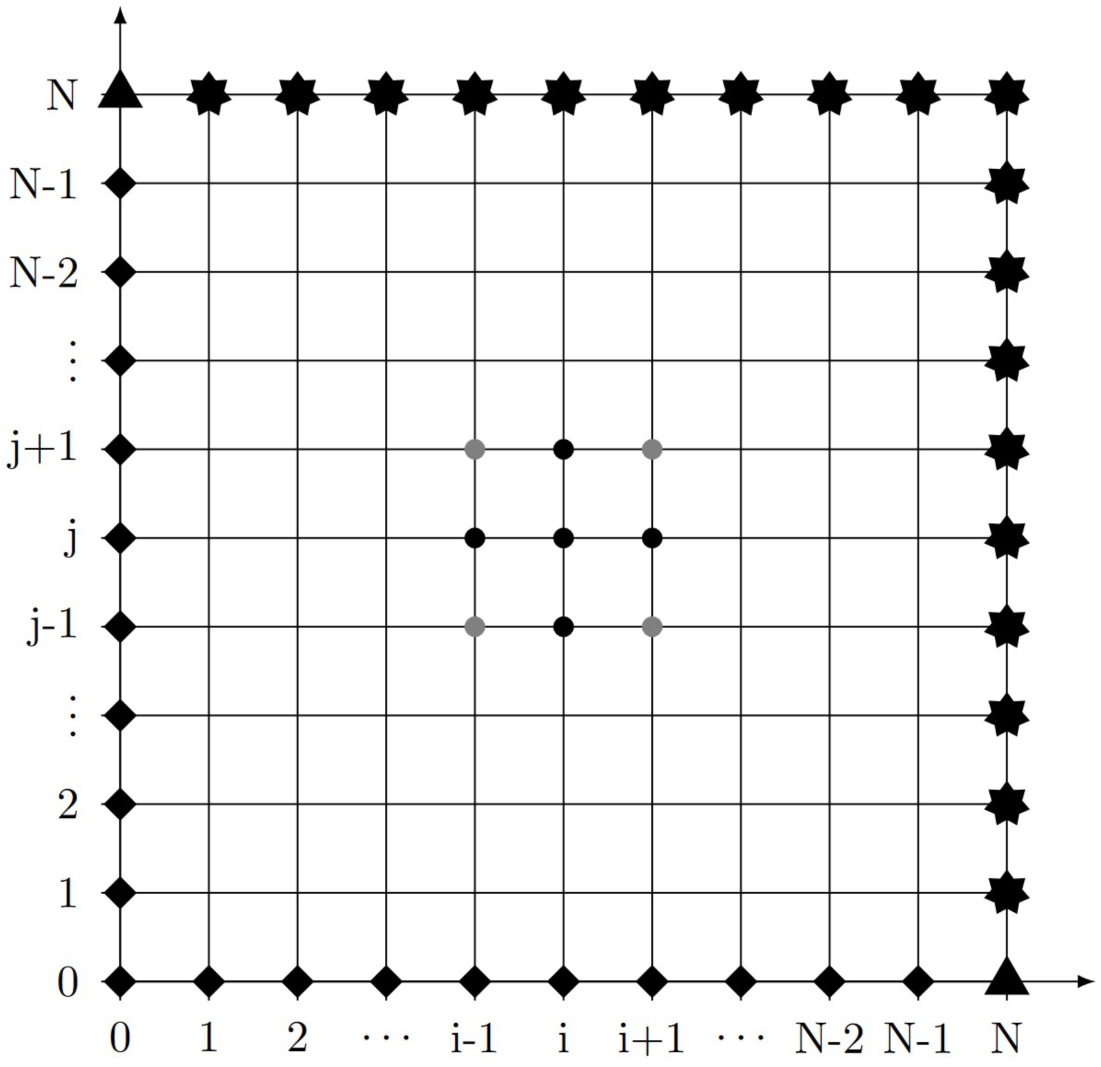


Figure 1: Close-field (as diamonds) and far-field (as stars) boundary conditions.

3.5 Far-Field Boundary Conditions

Extrapolating the 1D case of the limit as the payoff function goes to infinity into 2D, the paper [1] defined the far-field boundary conditions as:

$$\begin{aligned} F(t, x_{max}, y) &= \tfrac{x_{\max}+y}{2} - Ke^{-r(T-t)} \\ F(t, x, y_{max}) &= \tfrac{x+y_{\max}}{2} - Ke^{-r(T-t)} \end{aligned} \tag{5}$$

The paper [1] recommends, as a "rule of thumb", to limit the upper domain by setting the $x_{\max}$ and $y_{\max}$ to $4K$ to $6K$ times the number of spatial dimensions (two in this case). I have chosen to use $10K$ as the upper limit for both x and y.

4 Implementation using the MOLE Library

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

4.1 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 2 displays an example of the center points (grey) that become face points (red) with the use of G.

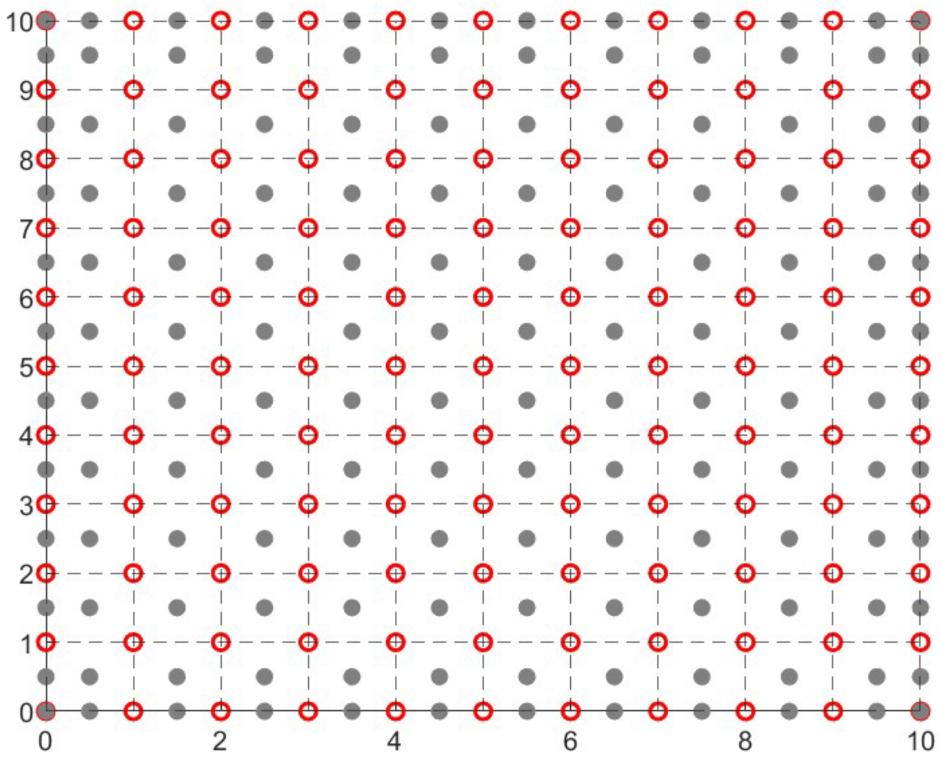


Figure 2: Center Points (Grey) and Face Points (Red)

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,

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

4.2 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 $DI^{CF} = \begin{bmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{bmatrix}$

$$\mathcal{L} = -r\begin{bmatrix} x_{\text{diag}} \\ y_{\text{diag}} \end{bmatrix} I^{FC}G - \frac{1}{2}DI^{CF}\begin{bmatrix} x_{\text{diag}} & \\ & y_{\text{diag}} \end{bmatrix}\begin{bmatrix} \sigma^2_{(1,2)}I & \sigma^2_{(1,2)}I \\ \sigma^2_{(2,1)}I & \sigma^2_{(2,2)}I \end{bmatrix}\begin{bmatrix} x_{\text{diag}} & \\ & y_{\text{diag}} \end{bmatrix} I^{FC}G + r\mathbf{I} \quad (6)$$

4.3 MATLAB Implementation

Section 4 is implemented in the following code.

```matlab
%% operator matrix
small_I = speye((m+2)*(m+2), (n+2)*(n+2));
I = speye((m+2)*(m+2)+(n+2)*(n+2), (m+2)*(m+2)+(n+2)*(n+2));

XI = spdiags(Xvec, 0, (m+2)*(m+2), (m+2)*(m+2));
YI = spdiags(Yvec, 0, (n+2)*(n+2), (n+2)*(n+2));

XYI = spdiags([Xvec; Yvec],0,(m+2)*(m+2)+(n+2)*(n+2),(m+2)*(m+2)+(n+2)*(n+2));

XY = [XI,YI];

O = [omega(1,1)*small_I, omega(1,2)*small_I; omega(2,1)*small_I, omega(2,2)*
    small_I]; % MD

%% Setting Up Gradient and Divergence Matricies
G = grad2D(k, m, dx, n, dy);
D = div2D(k, m, dx, n, dy);

%% Setting Up Interpolation Matrices

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

%% Combine for Black-Scholes Matrix
A = -(r*XY*IFC*G) - 0.5*(D*ICF*XYI*O*XYI*IFC*G) + r*small_I;
```

Implementation of F_t

4.4 Close-Field Boundary Condition

As described earlier, 3.4, 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

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

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

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* $= \nabla_x^2 = \nabla_y^2$,

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

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

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

$$\mathcal{L}_x = -rx_{\text{diag}}I_x^{FC}G_x - \tfrac{1}{2}x^2_{\text{diag}}\sigma^2_{(1,1)}\nabla_x^2 + rI$$

$$\mathcal{L}_y = -ry_{\text{diag}}I_y^{FC}G_y - \tfrac{1}{2}y^2_{\text{diag}}\sigma^2_{(2,2)}\nabla_y^2 + rI$$

4.5 Far-Field Boundary Condition

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

5 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

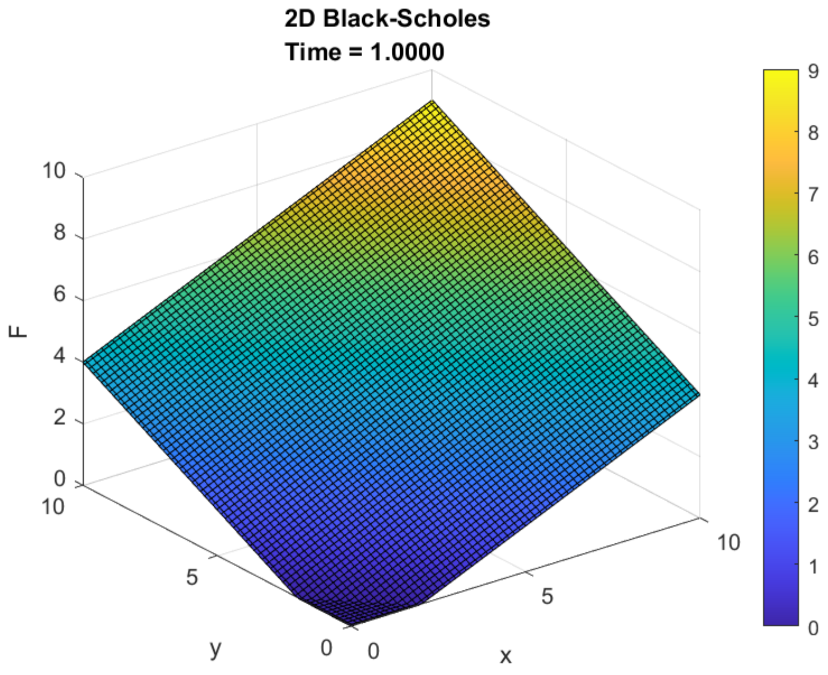
$$\left(I + \frac{2}{3}\Delta t A\right) F_{n+1} = -\frac{4}{3}F_n + \frac{1}{3}F_{n-1} \tag{7}$$

which requires information from the previous two time step. To start the backward discrete evolution, we utilize the BDF1 method as suggested in [1], which reads as

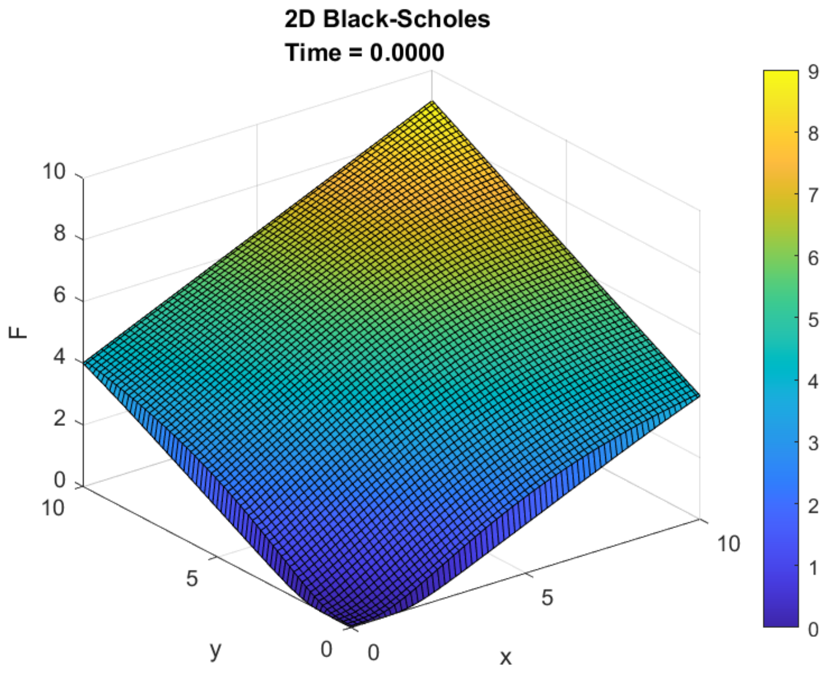
$$(I + \Delta t A)F_1 = F_0$$

6 Results

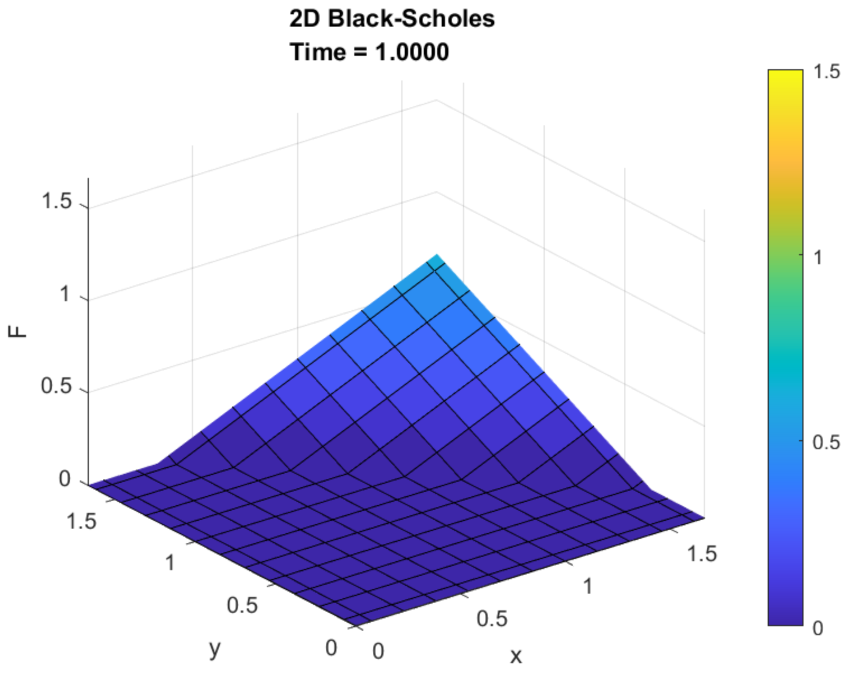
A plot of the initial condition [3] for the entire field is

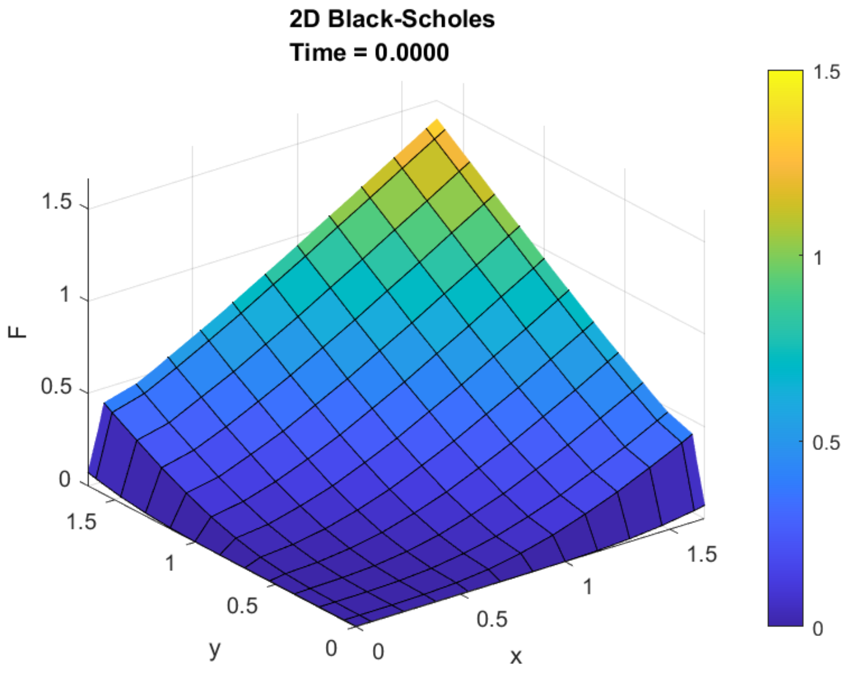


and the plot of the result for the entire field is



As done in the reference paper, the following graphs displays the initial conditions and the subsequent results of utilizing RBF2 for the discretization in time for the region of interest $\left[0 \leq x \leq \frac{5K}{3}\right]$ and $\left[0 \leq y \leq \frac{5K}{3}\right]$





The graph shows the expected curve for the Black-Scholes model expanded into two dimensions.

References

[1] Sundvall, T., and Trång, D., Examination of Impact from Different Boundary Conditions on the 2D Black-Scholes Model: Evaluating Pricing of European Call Options, Uppsala Universites, B.Sc. thesis, 2014.

[2] Corbino, J., and Castillo, J.E., High-order mimetic difference operators satisfying the extended Gauss divergence theorem, Journal of Computational and Applied Mathematics, 364 (2020).

[3] Corbino, J., and Castillo, J.E., MOLE: Mimetic Operators Library Enhanced: The Open-Source Library for Solving Partial Differential Equations using Mimetic Methods, 2017. https://github.com/csrc-sdsu/mole.