mathematics

MDPI

Review

Finite Difference Method for the Multi-Asset Black–Scholes Equations

Sangkwon Kim [1], Darae Jeong [2], Chaeyoung Lee [1] and Junseok Kim [1,*]

[1] Department of Mathematics, Korea University, Seoul 02841, Korea; ksk8863@korea.ac.kr (S.K.); chae1228@korea.ac.kr (C.L.)
[2] Department of Mathematics, Kangwon National University, Gangwon-do 24341, Korea; tinayoyo@kangwon.ac.kr
* Correspondence: cfdkim@korea.ac.kr

Received: 13 January 2020; Accepted: 6 March 2020; Published: 10 March 2020

Abstract: In this paper, we briefly review the finite difference method (FDM) for the Black–Scholes (BS) equations for pricing derivative securities and provide the MATLAB codes in the Appendix for the one-, two-, and three-dimensional numerical implementation. The BS equation is discretized non-uniformly in space and implicitly in time. The two- and three-dimensional equations are solved using the operator splitting method. In the numerical tests, we show characteristic examples for option pricing. The computational results are in good agreement with the closed-form solutions to the BS equations.

Keywords: operator splitting method; Black–Scholes equations; option pricing; finite difference method

1. Introduction

The well-known Black–Scholes (BS) partial differential equation (PDE) [1,2] is an accurate and efficient mathematical model for option pricing. The pioneers F. Black, M. Scholes, and R. Merton made a great breakthrough in the option pricing; and M. Scholes and R. Merton received in 1997 the Nobel Prize for their discovery of the BS equation. A European call option is a contract that gives the owner the right to buy an asset with a strike price K at an expiration date T. Let $u(s_1, s_2, \ldots, s_n, t)$ be the value of the option, where s_i is the underlying i-th asset value. We consider the following generalized n-asset BS equation [3,4]:

$$\frac{\partial u(\mathbf{s},t)}{\partial t} + \frac{1}{2}\sum_{i,j=1}^{n} \sigma_i \sigma_j \rho_{ij} s_i s_j \frac{\partial^2 u(\mathbf{s},t)}{\partial s_i \partial s_j} + r\sum_{i=1}^{n} s_i \frac{\partial u(\mathbf{s},t)}{\partial s_i} = r u(\mathbf{s},t), \tag{1}$$

for $(\mathbf{s},t) = (s_1, s_2, \ldots, s_n, t) \in \mathbf{R}_+^n \times [0,T)$. The final condition is

$$u(\mathbf{s},T) = u_T(\mathbf{s}). \tag{2}$$

Here, r is an interest rate, σ_i is a constant volatility of the i-th asset, and ρ_{ij} is the correlation coefficient between i-th and j-th underling assets.

The finite difference method (FDM) has been used for numerically solving the BS equations. In FDM, a PDE is replaced by discrete equations by using the Taylor series approximations to the partial derivatives. Then, we solve the resulting discrete equations by using various numerical methods [5]. Jeong et al. [6] developed the FDM without the far-field boundary condition. Hout and Valkov [7] obtained the European two-asset options by the FDM based numerical method considering non-uniform grids. For high-order option pricing schemes, there are the Crandall–Douglas method [8],

the Crank–Nicolson method [9,10], and second-order in time and fourth-order in space method [11]. In [12], Zhao and Tian investigated FDMs for solving the fractional BS equation, and studied the stability and convergence of the proposed first- and second-order implicit numerical schemes. Chen and Wang [13] presented a second-order Crank–Nicolson alternating direction implicit (ADI) scheme for solving a 2D spatial fractional BS equation. Sawangtong et al. [14] proposed the BS equation with two assets based on the Liouville–Caputo fractional derivative. In [15], Zhan et al. developed numerical methods for the time fractional BS equation. Hu and Gan [16] developed a fourth-order scheme for the BS PDE. In [17], Hendricks et al. combined high-order FDM with an ADI scheme for the BS equation in a sparse grid setting. In [18], Rao proposed an FDM for a generalized BS equation (non-constant interest rate and volatility) and proved that the scheme is second-order accurate spatially and temporally. In [19], Ullah presented numerical solution of three dimensional Heston–Hull–White (HHW) model using a high order FDM. The HHW model applies to pricing options arising from the probabilistic nature of asset prices, volatility, and risk-free interest rates. In [9], Ankudinova and Ehrhardt presented numerical methods for the nonlinear BS equation [20].

The main contribution of this paper is to present the detailed FDM for the BS equations for pricing derivative securities and provide the MATLAB codes for the one-, two-, and three-dimensional numerical implementation so that the beginners can use the provided codes for their research projects without wasting time for debugging the implementation [21].

The outline of this paper is as follows. The numerical solutions of the one-, two-, and three-dimensional BS equation are briefly described in Section 2. In Section 3, we present the numerical results of cash-or-nothing options in order to show the accuracy and efficiency. We finalize the paper with the conclusion in Section 4. In the Appendix A, we provide the MATLAB codes for the numerical implementation for one-, two-, and three-dimensions.

2. Numerical Solutions

By changing the variable with $\tau = T - t$, Equation (1) becomes

$$\frac{\partial u(\mathbf{s},\tau)}{\partial \tau} = \frac{1}{2}\sum_{i,j=1}^{n} \sigma_i \sigma_j \rho_{ij} s_i s_j \frac{\partial^2 u(\mathbf{s},\tau)}{\partial s_i \partial s_j} + r\sum_{i=1}^{n} s_i \frac{\partial u(\mathbf{s},\tau)}{\partial s_i} - r u(\mathbf{s},\tau). \tag{3}$$

The initial condition is $u(\mathbf{s},0) = u_T(\mathbf{s})$. Let us consider the numerical solution algorithm for the three-dimensional BS equation. The algorithms for the one- and two-dimensional BS equations are similarly defined. Let $x = s_1$, $y = s_2$, and $z = s_3$. Let $\mathcal{L}_{BS}$ be the following operator:

$$\begin{aligned} \mathcal{L}_{BS}u = {} & \frac{1}{2}\sigma_x^2 x^2 \frac{\partial^2 u}{\partial x^2} + \frac{1}{2}\sigma_y^2 y^2 \frac{\partial^2 u}{\partial y^2} + \frac{1}{2}\sigma_z^2 z^2 \frac{\partial^2 u}{\partial z^2} + \rho_{xy}\sigma_x\sigma_y xy \frac{\partial^2 u}{\partial x \partial y} \\ & + \rho_{yz}\sigma_y\sigma_z yz \frac{\partial^2 u}{\partial y \partial z} + \rho_{zx}\sigma_z\sigma_x zx \frac{\partial^2 u}{\partial z \partial x} + rx\frac{\partial u}{\partial x} + ry\frac{\partial u}{\partial y} + rz\frac{\partial u}{\partial z} - ru. \end{aligned} \tag{4}$$

Then, the BS equation can be written as

$$\begin{aligned} \frac{\partial u}{\partial \tau} &= \mathcal{L}_{BS}u \quad \text{for } (x,y,z,\tau) \in \Omega \times (0,T], \\ u(x,y,z,0) &= u_T(x,y,z), \end{aligned} \tag{5}$$

where $\tau = T - t$. We truncate the original infinite domain into a finite domain $\Omega = (0,L) \times (0,M) \times (0,N)$. We discretize Ω with a non-uniform space step $h_i^x = x_{i+1} - x_i, h_j^y = y_{j+1} - y_j$, $h_k^z = z_{k+1} - z_k$ for $i = 0, \ldots, N_x - 1$, $j = 0, \ldots, N_y - 1$, and $k = 0, \ldots, N_z - 1$. Here, $x_0 = y_0 = z_0 = 0$, $x_{N_x} = L$, $y_{N_y} = M$, and $z_{N_z} = N$. A time step is $\Delta\tau = T/N_\tau$. Figure 1 is an example of 3D non-uniform grid.

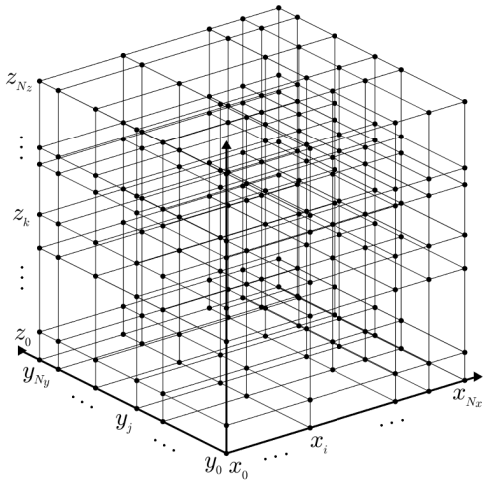

Figure 1. 3D non-uniform grid.

Let $u_{ijK}^n = u\left(x_i, y_j, z_k, n\Delta\tau\right)$, where $i = 0, \ldots, N_x$, $j = 0, \ldots, N_y$, $k = 0, \ldots, N_z$, and $n = 0, \ldots, N_\tau$. We use the zero Dirichlet boundary condition: $u(0, y, z, \tau) = u(x, 0, z, \tau) = u(x, y, 0, \tau) = 0$ for $0 \le x \le L$, $0 \le y \le M$, $0 \le z \le N$, $0 \le \tau \le T$, i.e., $u_{0jk}^n = 0$ for $j = 0, \ldots, N_y$, $k = 0, \ldots, N_z$, $u_{i0k}^n = 0$ for $i = 0, \ldots, N_x$, $k = 0, \ldots, N_z$, and $u_{ij0}^n = 0$ for $i = 0, \ldots, N_x$, $j = 0, \ldots, N_y$; and the homogeneous Neumann boundary condition: $\frac{\partial u}{\partial x}(L, y, z, \tau) = \frac{\partial u}{\partial y}(x, M, z, \tau) = \frac{\partial u}{\partial z}(x, y, N, \tau) = 0$ for $0 \le x \le L$, $0 \le y \le M$, $0 \le z \le N$, $0 \le \tau \le T$, that is, $u_{N_x+1,jk}^n = u_{N_x,jk}^n$ for $j = 0, \ldots, N_y$, $k = 0, \ldots, N_z$, $u_{i,N_y+1,k}^n = u_{i,N_y,k}^n$ for $i = 0, \ldots, N_x$, $k = 0, \ldots, N_z$, and $u_{ij,N_z+1}^n = u_{ij,N_z}^n$ for $i = 0, \ldots, N_x$, $j = 0, \ldots, N_y$.

Now, we apply the operator splitting (OS) method [22,23] to solve Equation (5). We consider the following three discrete equations:

$$\frac{u_{ijk}^{n+\frac{1}{3}} - u_{ijk}^n}{\Delta\tau} = \left(\mathcal{L}_{BS}^x u\right)_{ijk}^{n+\frac{1}{3}}, \tag{6}$$

$$\frac{u_{ijk}^{n+\frac{2}{3}} - u_{ijk}^{n+\frac{1}{3}}}{\Delta\tau} = \left(\mathcal{L}_{BS}^y u\right)_{ijk}^{n+\frac{2}{3}}, \tag{7}$$

$$\frac{u_{ijk}^{n+1} - u_{ijk}^{n+\frac{2}{3}}}{\Delta\tau} = \left(\mathcal{L}_{BS}^z u\right)_{ijk}^{n+1}, \tag{8}$$

for $1 \le i \le N_x$, $1 \le j \le N_y$, and $1 \le k \le N_z$. Here, the discrete difference operators $\mathcal{L}_{BS}^x$, $\mathcal{L}_{BS}^y$, and $\mathcal{L}_{BS}^z$ are defined by

$$\begin{aligned}\left(\mathcal{L}_{BS}^x u\right)_{ijk}^{n+\frac{1}{3}} &= \frac{\left(\sigma_x x_i\right)^2}{2} D_{xx} u_{ijk}^{n+\frac{1}{3}} + r x_i D_x u_{ijk}^{n+\frac{1}{3}} + \frac{1}{3}\sigma_x\sigma_y\rho_{xy} x_i y_j D_{xy} u_{ijk}^n \\ &\quad + \frac{1}{3}\sigma_y\sigma_z\rho_{yz} y_j z_k D_{yz} u_{ijk}^n + \frac{1}{3}\sigma_z\sigma_x\rho_{zx} z_k x_i D_{zx} u_{ijk}^n - \frac{1}{3} r u_{ijk}^{n+\frac{1}{3}},\end{aligned} \tag{9}$$

$$\begin{aligned}\left(\mathcal{L}_{BS}^y u\right)_{ijk}^{n+\frac{2}{3}} &= \frac{\left(\sigma_y y_j\right)^2}{2} D_{yy} u_{ijk}^{n+\frac{2}{3}} + r y_j D_y u_{ijk}^{n+\frac{2}{3}} + \frac{1}{3}\sigma_x\sigma_y\rho_{xy} x_i y_j D_{xy} u_{ijk}^{n+\frac{1}{3}} \\ &\quad + \frac{1}{3}\sigma_y\sigma_z\rho_{yz} y_j z_k D_{yz} u_{ijk}^{n+\frac{1}{3}} + \frac{1}{3}\sigma_z\sigma_x\rho_{zx} z_k x_i D_{zx} u_{ijk}^{n+\frac{1}{3}} - \frac{1}{3} r u_{ijk}^{n+\frac{2}{3}},\end{aligned} \tag{10}$$

$$\begin{aligned}\left(\mathcal{L}_{BS}^z u\right)_{ijk}^{n+1} &= \frac{\left(\sigma_z z_k\right)^2}{2} D_{zz} u_{ijk}^{n+1} + r z_k D_z u_{ijk}^{n+1} + \frac{1}{3}\sigma_x\sigma_y\rho_{xy} x_i y_j D_{xy} u_{ijk}^{n+\frac{2}{3}} \\ &\quad + \frac{1}{3}\sigma_y\sigma_z\rho_{yz} y_j z_k D_{yz} u_{ijk}^{n+\frac{2}{3}} + \frac{1}{3}\sigma_z\sigma_x\rho_{zx} z_k x_i D_{zx} u_{ijk}^{n+\frac{2}{3}} - \frac{1}{3} r u_{ijk}^{n+1},\end{aligned} \tag{11}$$

where the first and second derivatives are defined in [24] as

$$D_x u_{ijk} = -\frac{h_i^x}{h_{i-1}^x(h_{i-1}^x + h_i^x)} u_{i-1,jk} + \frac{h_i^x - h_{i-1}^x}{h_{i-1}^x h_i^x} u_{ijk} + \frac{h_{i-1}^x}{h_i^x(h_{i-1}^x + h_i^x)} u_{i+1,jk}, \tag{12}$$

$$D_y u_{ijk} = -\frac{h_j^y}{h_{j-1}^y(h_{j-1}^y + h_j^y)} u_{i,j-1,k} + \frac{h_j^y - h_{j-1}^y}{h_{j-1}^y h_j^y} u_{ijk} + \frac{h_{j-1}^y}{h_j^y(h_{j-1}^y + h_j^y)} u_{i,j+1,k}, \tag{13}$$

$$D_z u_{ijk} = -\frac{h_k^z}{h_{k-1}^z(h_{k-1}^z + h_k^z)} u_{ij,k-1} + \frac{h_k^z - h_{k-1}^z}{h_{k-1}^z h_k^z} u_{ijk} + \frac{h_{k-1}^z}{h_k^z(h_{k-1}^z + h_k^z)} u_{ij,k+1}, \tag{14}$$

$$D_{xx} u_{ijk} = \frac{2}{h_{i-1}^x(h_{i-1}^x + h_i^x)} u_{i-1,jk} - \frac{2}{h_{i-1}^x h_i^x} u_{ijk} + \frac{2}{h_i^x(h_{i-1}^x + h_i^x)} u_{i+1,jk}, \tag{15}$$

$$D_{yy} u_{ijk} = \frac{2}{h_{j-1}^y(h_{j-1}^y + h_j^y)} u_{i,j-1,k} - \frac{2}{h_{j-1}^y h_j^y} u_{ijk} + \frac{2}{h_j^y(h_{j-1}^y + h_j^y)} u_{i,j+1,k}, \tag{16}$$

$$D_{zz} u_{ijk} = \frac{2}{h_{k-1}^z(h_{k-1}^z + h_k^z)} u_{ij,k-1} - \frac{2}{h_{k-1}^z h_k^z} u_{ijk} + \frac{2}{h_k^z(h_{k-1}^z + h_k^z)} u_{ij,k+1}, \tag{17}$$

$$D_{xy} u_{ijk} = \frac{u_{i+1,j+1,k} - u_{i-1,j+1,k} - u_{i+1,j-1,k} + u_{i-1,j-1,k}}{h_i^x h_j^y + h_{i-1}^x h_j^y + h_i^x h_{j-1}^y + h_{i-1}^x h_{j-1}^y}, \tag{18}$$

$$D_{yz} u_{ijk} = \frac{u_{i,j+1,k+1} - u_{i,j-1,k+1} - u_{i,j+1,k-1} + u_{i,j-1,k-1}}{h_j^y h_k^z + h_{j-1}^y h_k^z + h_j^y h_{k-1}^z + h_{j-1}^y h_{k-1}^z}, \tag{19}$$

$$D_{zx} u_{ijk} = \frac{u_{i+1,j,k+1} - u_{i+1,j,k-1} - u_{i-1,j,k+1} + u_{i-1,j,k-1}}{h_k^z h_i^x + h_{k-1}^z h_i^x + h_k^z h_{i-1}^x + h_{k-1}^z h_{i-1}^x}. \tag{20}$$

Now, we describe the numerical algorithm for Equations (6)–(8). Given u_{ijk}^n, Equation (6) is rewritten as follows:

$$\alpha_i u_{i-1,jk}^{n+\frac{1}{3}} + \beta_i u_{ijk}^{n+\frac{1}{3}} + \gamma_i u_{i+1,jk}^{n+\frac{1}{3}} = f_{ijk}, \tag{21}$$

where

$$\alpha_i = -\frac{(\sigma_x x_i)^2}{h_{i-1}^x(h_{i-1}^x + h_i^x)} + r x_i \frac{h_i^x}{h_{i-1}^x(h_{i-1}^x + h_i^x)}, \tag{22}$$

$$\beta_i = \frac{1}{\Delta\tau} + \frac{(\sigma_x x_i)^2}{h_{i-1}^x h_i^x} - r x_i \frac{h_i^x - h_{i-1}^x}{h_{i-1}^x h_i^x} + \frac{r}{3}, \quad \gamma_i = -\frac{(\sigma_x x_i)^2}{h_i^x(h_{i-1}^x + h_i^x)} - r x_i \frac{h_{i-1}^x}{h_i^x(h_{i-1}^x + h_i^x)}, \tag{23}$$

$$f_{ijk}^n = \frac{1}{3}\sigma_x\sigma_y\rho_{xy} x_i y_j D_{xy} u_{ijk}^n + \frac{1}{3}\sigma_y\sigma_z\rho_{yz} y_j z_k D_{yz} u_{ijk}^n + \frac{1}{3}\sigma_x\sigma_z\rho_{zx} x_i z_k D_{zx} u_{ijk}^n - \frac{1}{\Delta\tau} u_{ijk}^n. \tag{24}$$

For fixed indices j and k; and $f_{1:N_x,jk}^n = [f_{1jk}^n \; f_{2jk}^n \; \cdots \; f_{N_x,jk}^n]^T$, the solution vector $u_{1:N_x,jk}^{n+\frac{1}{3}} = [u_{1jk}^{n+\frac{1}{3}} \; u_{2jk}^{n+\frac{1}{3}} \; \cdots \; u_{N_x,jk}^{n+\frac{1}{3}}]^T$ can be found by solving the tridiagonal system

$$A_x u_{1:N_x,jk}^{n+\frac{1}{3}} = f_{1:N_x,jk}^n, \tag{25}$$

where A_x is a matrix obtained from Equation (21) with the zero Dirichlet (i.e., $u_{0jk}^{n+\frac{1}{3}} = 0$ at $x = 0$) and the one-sided difference for the homogeneous Neumann (i.e., $u_{N_x+1,jk}^{n+\frac{1}{3}} = u_{N_x,jk}^{n+\frac{1}{3}}$ at $x = L$) boundary conditions, that is,

$$A_x = \begin{pmatrix} \beta_1 & \gamma_1 & 0 & \cdots & 0 & 0 \\ \alpha_2 & \beta_2 & \gamma_2 & \cdots & 0 & 0 \\ 0 & \alpha_3 & \beta_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \beta_{N_x-1} & \gamma_{N_x-1} \\ 0 & 0 & 0 & \cdots & \alpha_{N_x} & \beta_{N_x}+\gamma_{N_x} \end{pmatrix}. \tag{26}$$

To generate a tridiagonal matrix A_x of Equation (26) in MATLAB, see Listing 1, we use the $diag(V, n)$ function which returns a square matrix with the elements of input vector V on the n-th diagonal. For example, option $n = 0, n = 1$, and $n = -1$ place the elements of input vector V on the main-, first super-, and first sub-diagonal, respectively.

Listing 1: Generating a tridiagonal matrix A_x.

```
ax=(r*x(3:Nx).*hx(3:Nx)-(sigx*x(3:Nx)).^2)./(hx(2:Nx-1).*(hx(2:Nx-1)+hx(3:Nx)));
bx=1/dt+((sigx*x(2:Nx)).^2-r*x(2:Nx).*(hx(2:Nx)-hx(1:Nx-1)))./(hx(2:Nx).*hx(1:Nx-1));
gx=(-r*x(2:Nx).*hx(1:Nx-1)-(sigx*x(2:Nx)).^2)./(hx(2:Nx).*(hx(1:Nx-1)+hx(2:Nx)));
bx(Nx-1)=bx(Nx-1)+gx(Nx-1);
A=diag(ax,-1)+diag(bx,0)+diag(gx(1:Nx-2),1);
Ax=A+eye(Nx-1)*r/3;
```

Similarly, Equation (7) is rewritten as matrix form:

$$\alpha_j u_{i,j-1,k}^{n+\frac{2}{3}} + \beta_j u_{ijk}^{n+\frac{2}{3}} + \gamma_j u_{i,j+1,k}^{n+\frac{2}{3}} = f_{ijk}^{n+\frac{1}{3}}, \tag{27}$$

where

$$\alpha_j = -\frac{(\sigma_y y_j)^2}{h_{j-1}^y (h_{j-1}^y + h_j^y)} + r y_j \frac{h_j^y}{h_{j-1}^y (h_{j-1}^y + h_j^y)}, \tag{28}$$

$$\beta_j = \frac{1}{\Delta\tau} + \frac{(\sigma_y y_j)^2}{h_{j-1}^y h_j^y} - r y_j \frac{h_j^y - h_{j-1}^y}{h_{j-1}^y h_j^y} + \frac{r}{3}, \quad \gamma_j = -\frac{(\sigma_y y_j)^2}{h_j^y (h_{j-1}^y + h_j^y)} - r y_j \frac{h_{j-1}^y}{h_j^y (h_{j-1}^y + h_j^y)}, \tag{29}$$

$$f_{ijk}^{n+\frac{1}{3}} = \frac{1}{3}\sigma_x\sigma_y\rho_{xy}x_i y_j D_{xy} u_{ijk}^{n+\frac{1}{3}} + \frac{1}{3}\sigma_y\sigma_z\rho_{yz}y_j z_k D_{yz} u_{ijk}^{n+\frac{1}{3}} + \frac{1}{3}\sigma_z\sigma_x\rho_{zx}z_k x_i D_{zx} u_{ijk}^{n+\frac{1}{3}} - \frac{1}{\Delta\tau} u_{ijk}^{n+\frac{1}{3}}. \tag{30}$$

For fixed indices i and k; and $f_{i,1:N_y,k}^{n+\frac{1}{3}} = [f_{i1k}^{n+\frac{1}{3}}\ f_{i2k}^{n+\frac{1}{3}}\ \cdots\ f_{i,N_y,k}^{n+\frac{1}{3}}]^T$, the solution vector $u_{i,1:N_y,k}^{n+\frac{2}{3}} = [u_{i1k}^{n+\frac{2}{3}}\ u_{i2k}^{n+\frac{2}{3}}\ \cdots\ u_{i,N_y,k}^{n+\frac{2}{3}}]^T$ can be obtained by

$$A_y u_{i,1:N_y,k}^{n+\frac{2}{3}} = f_{i,1:N_y,k}^{n+\frac{1}{3}}, \tag{31}$$

where A_y is a matrix obtained from Equation (27) with the zero Dirichlet (i.e., $u_{i0k}^{n+\frac{2}{3}} = 0$ at $y = 0$) and the one-sided difference for the homogeneous Neumann (i.e., $u_{i,N_y+1,k}^{n+\frac{2}{3}} = u_{i,N_y,k}^{n+\frac{2}{3}}$ at $y = M$) boundary conditions, that is,

$$A_y = \begin{pmatrix} \beta_1 & \gamma_1 & 0 & \cdots & 0 & 0 \\ \alpha_2 & \beta_2 & \gamma_2 & \cdots & 0 & 0 \\ 0 & \alpha_3 & \beta_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \beta_{N_y-1} & \gamma_{N_y-1} \\ 0 & 0 & 0 & \cdots & \alpha_{N_y} & \beta_{N_y}+\gamma_{N_y} \end{pmatrix}. \tag{32}$$

The process of generating a tridiagonal matrix A_y is similar to that of generating A_x and the code is as follows Listing 2.

Listing 2: Generating a tridiagonal matrix A_y.

```
ay=(r*y(3:Ny).*hy(3:Ny)-(sigy*y(3:Ny)).^2)./(hy(2:Ny-1).*(hy(2:Ny-1)+hy(3:Ny)));
by=1/dt+((sigy*y(2:Ny)).^2-r*y(2:Ny).*(hy(2:Ny)-hy(1:Ny-1)))./(hy(2:Ny).*hy(1:Ny-1));
gy=(-r*y(2:Ny).*hy(1:Ny-1)-(sigy*y(2:Ny)).^2)./(hy(2:Ny).*(hy(1:Ny-1)+hy(2:Ny)));
by(Ny-1)=by(Ny-1)+gy(Ny-1);
B=diag(ay,-1)+diag(by,0)+diag(gy(1:Ny-2),1);
Ay=B+eye(Nx-1)*r/3
```

Finally, Equation (8) is rewritten as matrix form:

$$\alpha_k u_{ij,k-1}^{n+1} + \beta_k u_{ijk}^{n+1} + \gamma_k u_{ij,k+1}^{n+1} = f_{ijk}^{n+\frac{2}{3}}, \tag{33}$$

where

$$\alpha_k = -\frac{(\sigma_z z_k)^2}{h_{k-1}^z(h_{k-1}^z + h_k^z)} + r z_k \frac{h_k^z}{h_{k-1}^z(h_{k-1}^z + h_k^z)}, \tag{34}$$

$$\beta_k = \frac{1}{\Delta\tau} + \frac{(\sigma_z z_k)^2}{h_{k-1}^z h_k^z} - r z_k \frac{h_k^z - h_{k-1}^z}{h_{k-1}^z h_k^z} + \frac{r}{3}, \quad \gamma_k = -\frac{(\sigma_z z_k)^2}{h_k^z(h_{k-1}^z + h_k^z)} - r z_k \frac{h_{k-1}^z}{h_k^z(h_{k-1}^z + h_k^z)}, \tag{35}$$

$$f_{ijk}^{n+\frac{2}{3}} = \frac{1}{3}\sigma_x\sigma_y\rho_{xy}x_i y_j D_{xy} u_{ijk}^{n+\frac{2}{3}} + \frac{1}{3}\sigma_y\sigma_z\rho_{yz}y_j z_k D_{yz} u_{ijk}^{n+\frac{2}{3}} + \frac{1}{3}\sigma_z\sigma_x\rho_{zx}z_k x_i D_{zx} u_{ijk}^{n+\frac{2}{3}} - \frac{1}{\Delta\tau} u_{ijk}^{n+\frac{2}{3}}. \tag{36}$$

For fixed indices i and j; and $f_{ij,1:N_z}^{n+\frac{2}{3}} = [f_{ij1}^{n+\frac{2}{3}}\ f_{ij2}^{n+\frac{2}{3}}\ \cdots\ f_{ij,N_z}^{n+\frac{2}{3}}]^T$, the solution vector $u_{ij,1:N_z}^{n+1} = [u_{ij1}^{n+1}\ u_{ij2}^{n+1}\ \cdots\ u_{ij,N_z}^{n+1}]^T$ can be found by solving the tridiagonal system

$$A_z u_{ij,1:N_z}^{n+1} = f_{ij,1:N_z}^{n+\frac{2}{3}}, \tag{37}$$

where A_z is a matrix obtained from Equation (33) with the zero Dirichlet (i.e., $u_{ij0}^{n+1} = 0$ at $z = 0$) and the one-sided difference for the homogeneous Neumann (i.e., $u_{ij,N_z+1}^{n+1} = u_{ij,N_z}^{n+1}$ at $z = N$) boundary conditions, that is,

$$A_z = \begin{pmatrix} \beta_1 & \gamma_1 & 0 & \cdots & 0 & 0 \\ \alpha_2 & \beta_2 & \gamma_2 & \cdots & 0 & 0 \\ 0 & \alpha_3 & \beta_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \beta_{N_z-1} & \gamma_{N_z-1} \\ 0 & 0 & 0 & \cdots & \alpha_{N_z} & \beta_{N_z}+\gamma_{N_z} \end{pmatrix}. \tag{38}$$

The code for generating a tridiagonal matrix A_z is as follows, Listing 3.

Listing 3: Generating a tridiagonal matrix A_z.

```
az=(r*z(3:Nz).*hz(3:Nz)-(sigz*z(3:Nz)).^2)./(hz(2:Nz-1).*(hz(2:Nz-1)+hz(3:Nz)));
bz=1/dt+((sigz*z(2:Nz)).^2-r*z(2:Nz).*(hz(2:Nz)-hz(1:Nz-1)))./(hz(2:Nz).*hz(1:Nz-1));
gz=(-r*z(2:Nz).*hz(1:Nz-1)-(sigz*z(2:Nz)).^2)./(hz(2:Nz).*(hz(1:Nz-1)+hz(2:Nz)));
bz(Nz-1)=bz(Nz-1)+gz(Nz-1);
C=diag(az,-1)+diag(bz,0)+diag(gz(1:Nz-2),1);
Az=C+eye(Nz-1)*r/3;
```

Then, Equations (6)–(8) are implemented with the following Algorithm 1.

Algorithm 1 Implicit scheme and OSM for three-dimensional Black–Scholes (BS) equation.

Require: Previous data u^n.
procedure FIND THE SOLUTION $u^{n+\frac{1}{3}}$ for Equation (6)
for $j = 1; j \leq N_y; j{+}{+}$ **do**
for $k = 1; k \leq N_z; k{+}{+}$ **do**
for $i = 1; i \leq N_x; i{+}{+}$ **do**
Generate A_x and set f^n_{ijk} by using Equations (22)–(24)
end for
Solve $A_x u^{n+\frac{1}{3}}_{1:N_x,jk} = f^n_{1:N_x,jk}$
end for
end for
end procedure
procedure FIND THE SOLUTION $u^{n+\frac{2}{3}}$ for Equation (7)
for $k = 1; k \leq N_z; k{+}{+}$ **do**
for $i = 1; i \leq N_x; i{+}{+}$ **do**
for $j = 1; j \leq N_y; j{+}{+}$ **do**
Generate A_y and set $f^{n+\frac{1}{3}}_{ijk}$ by using Equations (28)–(30)
end for
Solve $A_y u^{n+\frac{2}{3}}_{i,1:N_y,k} = f^{n+\frac{1}{3}}_{i,1:N_y,k}$
end for
end for
end procedure
procedure FIND THE SOLUTION u^{n+1} for Equation (8)
for $j = 1; j \leq N_y; j{+}{+}$ **do**
for $i = 1; i \leq N_x; i{+}{+}$ **do**
for $k = 1; k \leq N_z; k{+}{+}$ **do**
Generate A_z and set $f^{n+\frac{2}{3}}_{ijk}$ by using Equations (34)–(36)
end for
Solve $A_z u^{n+1}_{ij,1:N_z} = f^{n+\frac{2}{3}}_{ij,1:N_z}$
end for
end for
end procedure

To solve the tridiagonal systems found in Equations (25), (31), and (37), we used the backslash operator in MATLAB. Also, *interp3()* function, which returns interpolated value from a function of three variables at the query points, is used to determine the value at specific query point using the calculated u. The code for Algorithm 1 is as follows, Listing 4:

Listing 4: Code for Algorithm 1.

```
v=u;fx=zeros(Nx-1,1);fy=zeros(Ny-1,1);fz=zeros(Nz-1,1);
for n=1:Nt
for j=2:Ny
for k=2:Nz
for i=2:Nx
fx(i-1)=(1/3)*(rhoxy*sigx*sigy*x(i)*y(j)...
*(u(i+1,j+1,k)-u(i+1,j-1,k)-u(i-1,j+1,k)+u(i-1,j-1,k))...
/(hx(i-1)*hy(j)+hx(i)*hy(j)+hx(i)*hy(j-1)+hx(i-1)*hy(j-1))...
+rhoxz*sigx*sigz*x(i)*z(k)*(u(i+1,j,k+1)-u(i+1,j,k-1)-u(i-1,j,k+1)+u(i-1,j,k-1))...
/(hx(i-1)*hz(k)+hx(i)*hz(k)+hx(i)*hz(k-1)+hx(i-1)*hz(k-1))...
+rhoyz*sigy*sigz*y(j)*z(k)*(u(i,j+1,k+1)-u(i,j+1,k-1)-u(i,j-1,k+1)+u(i,j-1,k-1))...
/(hy(j-1)*hz(k)+hy(j)*hz(k)+hy(j)*hz(k-1)+hy(j-1)*hz(k-1)))+u(i,j,k)/dt;
end
v(2:Nx,j,k)=Ax\fx;
end
end
v(2:Nx,2:Ny,Nz+1)=v(2:Nx,2:Ny,Nz);
v(Nx+1,2:Ny,2:Nz+1)=v(Nx,2:Ny,2:Nz+1);
v(2:Nx+1,Ny+1,2:Nz+1)=v(2:Nx+1,Ny,2:Nz+1);
for k=2:Nz
for i=2:Nx
for j=2:Ny
fy(j-1)=(1/3)*(rhoxy*sigx*sigy*x(i)*y(j)...
*(v(i+1,j+1,k)-v(i+1,j-1,k)-v(i-1,j+1,k)+v(i-1,j-1,k))...
/(hx(i-1)*hy(j)+hx(i)*hy(j)+hx(i)*hy(j-1)+hx(i-1)*hy(j-1))...
+rhoxz*sigx*sigz*x(i)*z(k)*(v(i+1,j,k+1)-v(i+1,j,k-1)-v(i-1,j,k+1)+v(i-1,j,k-1))...
/(hx(i-1)*hz(k)+hx(i)*hz(k)+hx(i)*hz(k-1)+hx(i-1)*hz(k-1))...
+rhoyz*sigy*sigz*y(j)*z(k)*(v(i,j+1,k+1)-v(i,j+1,k-1)-v(i,j-1,k+1)+v(i,j-1,k-1))...
/(hy(j-1)*hz(k)+hy(j)*hz(k)+hy(j)*hz(k-1)+hy(j-1)*hz(k-1)))+v(i,j,k)/dt;
end
u(i,2:Ny,k)=Ay\fy;
end
end
u(2:Nx,2:Ny,Nz+1)=u(2:Nx,2:Ny,Nz);
u(Nx+1,2:Ny,2:Nz+1)=u(Nx,2:Ny,2:Nz+1);
u(2:Nx+1,Ny+1,2:Nz+1)=u(2:Nx+1,Ny,2:Nz+1);
for j=2:Ny
for i=2:Nx
for k=2:Nz
fz(k-1)=(1/3)*(rhoxy*sigx*sigy*x(i)*y(j)...
*(u(i+1,j+1,k)-u(i+1,j-1,k)-u(i-1,j+1,k)+u(i-1,j-1,k))...
/(hx(i-1)*hy(j)+hx(i)*hy(j)+hx(i)*hy(j-1)+hx(i-1)*hy(j-1))...
+rhoxz*sigx*sigz*x(i)*z(k)*(u(i+1,j,k+1)-u(i+1,j,k-1)- u(i-1,j,k+1)+u(i-1,j,k-1))...
/(hx(i-1)*hz(k)+hx(i)*hz(k)+hx(i)*hz(k-1)+hx(i-1)*hz(k-1))...
+rhoyz*sigy*sigz*y(j)*z(k)*(u(i,j+1,k+1)-u(i,j+1,k-1)-u(i,j-1,k+1)+u(i,j-1,k-1))...
/(hy(j-1)*hz(k)+hy(j)*hz(k)+hy(j)*hz(k-1)+hy(j-1)*hz(k-1)))+u(i,j,k)/dt;
end
v(i,j,2:Nz)=Az\fz;
end
end
v(2:Nx,2:Ny,Nz+1)=v(2:Nx,2:Ny,Nz);
v(Nx+1,2:Ny,2:Nz+1)=v(Nx,2:Ny,2:Nz+1);
v(2:Nx+1,Ny+1,2:Nz+1)=v(2:Nx+1,Ny,2:Nz+1);
u=v;
end
threeD_price=interp3(x,y,z,u(1:Nx,1:Ny,1:Nz),x0,y0,z0,'linear')
```

3. Numerical Experiments

We consider one-, two-, and three-asset cash-or-nothing options as test problems. The codes of the payoff function and the closed-form formula are provided below. Unless otherwise specified, we use the following parameters in numerical tests: maturity time $T = 1$, time step $\Delta\tau = 0.5/365$, risk-free interest rate $r = 0.03$, and cash $c = 100$. We conduct numerical tests on three non-uniform grids:

$$\begin{aligned}\Omega_1 &= [0, 1.5, 5.5, 9.5, \cdots, 77.5, 80.5, 83.5, \cdots, 122.5, 126.5, 130.5 \cdots, 298.5, 300],\\ \Omega_2 &= [0, 1, 4, 7, \cdots, 79, 81, 83, \cdots, 121, 124, 127, \cdots, 298, 300],\\ \Omega_3 &= [0, 0.5, 2.5, 4.5, \cdots, 80.5, 81.5, 82.5, \cdots, 120.5, 122.5, 124.5, \cdots, 298.5, 300].\end{aligned}$$

Here, error is defined by the discrete L^2-norm of relative error between the numerical and the closed-form solutions:

$$e_{L^2} = \sqrt{\frac{1}{\aleph}\sum_i\sum_j\sum_k \left(\frac{u_{ijk}^{N_\tau} - u(x_i, y_j, z_k, T)}{u(x_i, y_j, z_k, T)}\right)^2}, \text{ for } (x_i, y_j, z_k) \in (80, 120) \times (80, 120) \times (80, 120), \quad (39)$$

where $\aleph$, $u_{ijk}^{N_\tau}$ and $u(x_i, y_j, z_k, T)$ are the number of grid points within the interval $(80, 120) \times (80, 120) \times (80, 120)$, the numerical and the closed-form solutions of the cash-or-nothing option, respectively. Here, accuracy is important at the interesting region $(80, 120) \times (80, 120) \times (80, 120)$ because the payoff of cash-or-nothing is non-smooth and a large volume of trades take place on the neighborhood of the strike K. One- and two-dimensional errors are defined similarly.

3.1. One-Asset Option Test

For the pricing of the one-asset cash-or-nothing option, we use strike price $K = 100$, volatility of underlying asset $\sigma = 0.3$, and computational domain $\Omega = [0, 300]$. The payoff function of cash-or-nothing option is

$$u_T(x) = \begin{cases} c, & \text{if } x \geq K, \\ 0, & \text{otherwise.} \end{cases} \quad (40)$$

The following is the code for the payoff function of the one-asset cash-or-nothing option, see Listing 5:

Listing 5: Pay off of the one-asset cash-or-nothing option.

```
L=300;x=[0 1.5:4:77.5 80.5:3:119.5 122.5:4:L-1.5 L];Nx=length(x);K=100;c=100;
u=zeros(Nx,1);
for i=1:Nx
if x(i)>=K
u(i)=c;
end
end
```

The closed-form formula for the cash-or-nothing option is as follows [25]:

$$u(x, \tau) = ce^{-r\tau}N(d), \quad d = \frac{\ln\left(\frac{x}{K}\right) + \left(r - \frac{\sigma^2}{2}\right)\tau}{\sigma\sqrt{\tau}}, \quad (41)$$

where the cumulative normal distribution $N(d) = (1/\sqrt{2\pi})\int_{-\infty}^{d} e^{-0.5\xi^2} d\xi$.

The code of the closed-form formula for the cash-or-nothing option is as follows, Listing 6:

Listing 6: Closed-form solution for the one-asset cash-or-nothing option.

```
x0=100;K=100;c=100;sig=0.3;r=0.03;T=1;
d=(log(x0/K)+(r-0.5*sig^2)*T)/(sig*sqrt(T));
oneD_price=c*exp(-r*T)*normcdf(d)
```

Here, *normcdf(d)* function returns the cumulative normal distribution and is evaluated at the value d.

Figure 2 shows the closed-form solution and numerical solution on the grid Ω_3.

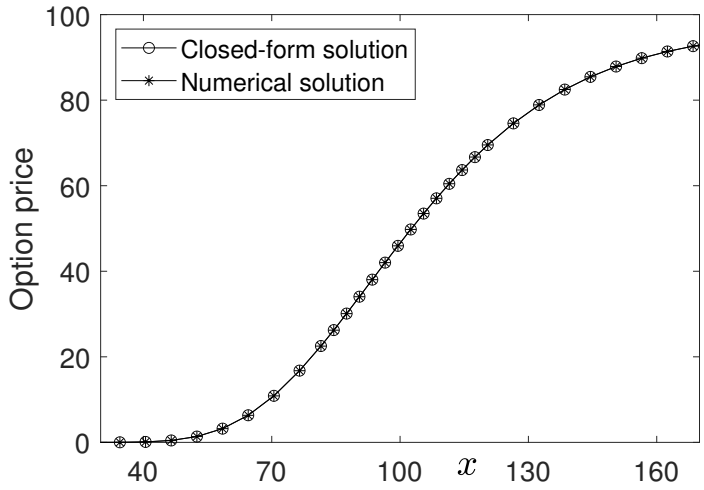

Figure 2. Comparison between the closed-form and the numerical solutions of cash-or-nothing option with Ω_3.

Table 1 lists the numerical solutions of cash-or-nothing option at underlying asset $x = 100$ and its corresponding error e_{L^2} with grid Ω_i for $i = 1, 2, 3$.

Table 1. Numerical prices and errors on different grids for one-asset option at $x = 100$. Here, $u(x, T) = 46.58732417$.

Grid	Numerical Solution	e_{L^2}
Ω_1	46.57902712	0.00096356
Ω_2	46.58536682	0.00049427
Ω_3	46.58834737	0.00025289

3.2. Two-Asset Option Test

For pricing two-asset option, we use strike prices $K_1 = 100, K_2 = 100$, volatilities of underlying assets $\sigma_x = 0.3$, $\sigma_y = 0.3$, the correlation coefficient between two underling assets $\rho = 0.5$, and computational domain $\Omega = [0, 300] \times [0, 300]$. The payoff function of cash-or-nothing option with two assets is

$$u_T(x, y) = \begin{cases} c, & \text{if } x \geq K_1,\ y \geq K_2, \\ 0, & \text{otherwise.} \end{cases} \tag{42}$$

The code of the payoff function for two-asset cash-or-nothing option is as follows, Listing 7:

Listing 7: Pay off of the two-asset cash-or-nothing option.

```
L=300;x=[0 1.5:4:77.5 80.5:3:119.5 122.5:4:L-1.5 L];Nx=length(x);K=100;c=100;y=x;Ny=Nx;
u=zeros(Nx+1,Ny+1);
for i=1:Nx
for j=1:Ny
if x(i)>=K && y(j)>=K
u(i,j)=c;
end
end
end
u(2:Nx,Ny+1)=u(2:Nx,Ny); u(Nx+1,2:Ny+1)=u(Nx,2:Ny+1);
```

The closed-form formula for the cash-or-nothing option [25] is

$$u(x,y,\tau) = ce^{-r\tau}B(d_x,d_y;\rho),\ d_x = \frac{\ln\left(\frac{x}{K_1}\right) + \left(r - 0.5\sigma_x^2\right)\tau}{\sigma_x\sqrt{\tau}},\ d_y = \frac{\ln\left(\frac{y}{K_2}\right) + \left(r - 0.5\sigma_y^2\right)\tau}{\sigma_y\sqrt{\tau}}, \quad (43)$$

where the bivariate cumulative normal distribution function $B(\alpha,\beta;\rho)$ [26] is

$$B(d_x,d_y;\rho) = \frac{1}{2\pi\sqrt{1-\rho^2}}\int_{-\infty}^{d_x}\int_{-\infty}^{d_y} e^{-\frac{\xi_1^2 - 2\rho\xi_1\xi_2 + \xi_2^2}{2(1-\rho^2)}} d\xi_2 d\xi_1. \quad (44)$$

The code of the closed-form formula for the two-asset cash-or-nothing option is as follows, Listing 8:

Listing 8: Closed-form solution for the two-asset cash-or-nothing option.

```
x0=100;K=100;c=100;r=0.03;T=1;
y0=x0;rho=0.5;sigx=0.3;sigy=sigx;
corr=[1,rho;rho,1];
dx=(log(x0/K)+(r-sigx^2/2)*T)/(sigx*sqrt(T));
dy=(log(y0/K)+(r-sigy^2/2)*T)/(sigy*sqrt(T));
twoD_price=c*exp(-r*T)*mvncdf([dx,dy],[0,0],corr)
```

In the code, the *mvncdf(X, μ, σ)* function returns the multivariate cumulative normal distribution and is evaluated at each row of X. For more information, see the document in MATLAB and [26].

Figure 3a,b show the closed-form and numerical solutions with grid $\Omega_3 \times \Omega_3$, respectively.

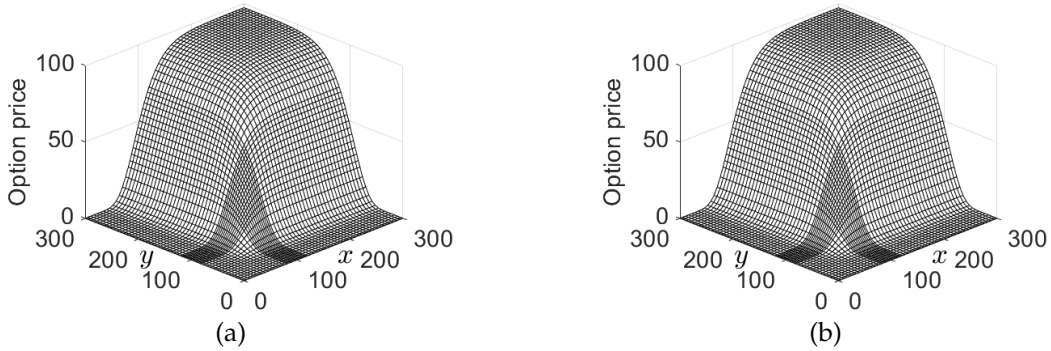

Figure 3. Closed-form and numerical solutions of cash-or-nothing option with $\Omega_3 \times \Omega_3$: (**a**) closed-form solution and (**b**) numerical solution.

Table 2 lists the numerical solutions of cash-or-nothing option at underlying assets $x = y = 100$ and its corresponding error e_{L^2} on grid $\Omega_i \times \Omega_i$ for $i = 1, 2, 3$.

Table 2. Numerical prices and errors on different grids for two-asset option at $x = y = 100$. Here, $u(x, y, T) = 30.43550958$.

Grid	Numerical Solution	e_{L^2}
$\Omega_1 \times \Omega_1$	30.40026164	0.00136876
$\Omega_2 \times \Omega_2$	30.42419734	0.00066143
$\Omega_3 \times \Omega_3$	30.43889746	0.00030173

3.3. Three-Asset Test

For pricing the three-asset option, we use strike prices $K_1 = K_2 = K_3 = 100$, volatilities of underlying assets $\sigma_x = \sigma_y = \sigma_z = 0.3$, the correlation coefficients $\rho_{xy} = \rho_{yz} = \rho_{zx} = 0.5$, and computational domain $\Omega = [0, 300] \times [0, 300] \times [0, 300]$. The payoff function of the cash-or-nothing option with three assets is

$$u_T(x, y, z) = \begin{cases} c, & \text{if } x \geq K_1,\ y \geq K_2,\ z \geq K_3, \\ 0, & \text{otherwise.} \end{cases} \tag{45}$$

The setting of the initial condition for the three-asset option is similar to the one- and two-asset option, with the following codes , Listing 9:

Listing 9: Pay off of the three-asset cash-or-nothing option.

```
L=300;x=[0 1.5:4:77.5 80.5:3:119.5 122.5:4:L-1.5 L];Nx=length(x);K=100;c=100;
y=x;z=x;Ny=Nx;Nz=Nx; u=zeros(Nx+1,Ny+1,Nz+1);
for i=1:Nx
for j=1:Ny
for k=1:Nz
if x(i)>=K && y(j)>=K && z(k)>=K
u(i,j,k)=c;
end
end
end
end
u(2:Nx,2:Ny,Nz+1)=u(2:Nx,2:Ny,Nz);
u(Nx+1,2:Ny,2:Nz+1)=u(Nx,2:Ny,2:Nz+1);
u(2:Nx+1,Ny+1,2:Nz+1)=u(2:Nx+1,Ny,2:Nz+1);
```

The closed-form formula for the cash-or-nothing option with three underlying assets [25] is as follow:

$$u(x, y, z, \tau) = ce^{-r\tau} M(d_x, d_y, d_z; R), \tag{46}$$

where

$$d_x = \frac{\ln\left(\frac{x}{K_1}\right) + \left(r - 0.5\sigma_x^2\right)\tau}{\sigma_x\sqrt{\tau}},\ d_y = \frac{\ln\left(\frac{y}{K_2}\right) + \left(r - 0.5\sigma_y^2\right)\tau}{\sigma_y\sqrt{\tau}},\ d_z = \frac{\ln\left(\frac{z}{K_3}\right) + \left(r - 0.5\sigma_z^2\right)\tau}{\sigma_z\sqrt{\tau}}, \tag{47}$$

and the trivariate cumulative normal distribution function $M(d_x, d_y, d_z; R)$ is defined in [26] by

$$M(d_x, d_y, d_z; R) = \frac{1}{\sqrt{|R|(2\pi)^3}} \int_{-\infty}^{d_x} \int_{-\infty}^{d_y} \int_{-\infty}^{d_z} e^{-0.5\mathbf{d}R^{-1}\mathbf{d}^T} d\xi_3 d\xi_2 d\xi_1, \tag{48}$$

where $\mathbf{d} = (d_x, d_y, d_z)$ and a correlation matrix R is

$$R = \begin{pmatrix} 1 & \rho_{xy} & \rho_{zx} \\ \rho_{xy} & 1 & \rho_{yz} \\ \rho_{zx} & \rho_{yz} & 1 \end{pmatrix}. \tag{49}$$

The code of the closed-form formula for the three-asset cash-or-nothing option is as follows, Listing 10:

Listing 10: Closed-form solution for the three-asset cash-or-nothing option.

```
x0=100;K=100;c=100;r=0.03;T=1;
y0=x0;z0=x0;rhoxy=0.5;rhoyz=0.5;rhoxz=0.5;
sigx=0.3;sigy=sigx;sigz=sigx;
corr=[1,rhoxy,rhoxz;rhoxy,1,rhoyz;rhoxz,rhoyz,1];
dx=(log(x0/K)+(r-sigx^2/2)*T)/(sigx*sqrt(T));
dy=(log(y0/K)+(r-sigy^2/2)*T)/(sigy*sqrt(T));
dz=(log(z0/K)+(r-sigz^2/2)*T)/(sigz*sqrt(T));
threeD_price=c*exp(-r*T)*mvncdf([dx,dy,dz],[0,0,0],corr)
```

Table 3 lists the numerical solution of cash-or-nothing option at underlying assets $x = y = z = 100$ and its corresponding error e_{L^2} on grid $\Omega_i \times \Omega_i \times \Omega_i$ for $i = 1, 2, 3$. From these results, we can confirm that FDM for option pricing is accurate.

Table 3. Numerical prices and errors on different grids for three-asset option at $x = y = z = 100$. Here, $u(x, y, z, T) = 22.52919331$.

Grid	Numerical Solution	e_{L^2}
$\Omega_1 \times \Omega_1 \times \Omega_1$	22.48442671	0.00170747
$\Omega_2 \times \Omega_2 \times \Omega_2$	22.51504195	0.00074917
$\Omega_3 \times \Omega_3 \times \Omega_3$	22.53434245	0.00031189

We note that there are many boundary conditions for the BS equation such as Dirichlet, Neumann, linear, PDE, and payoff-consistent boundary conditions. See [27,28] for the detailed numerical treatment of the above-mentioned boundary conditions. Furthermore, there are no-boundary condition [6] and hybrid boundary condition [22].

4. Conclusions

In this paper, we briefly reviewed the FDM for the numerical solution of the BS equations and provided the MATLAB codes for numerical implementation. The FDM has been used in a variety of ways for extensive study to solve partial differential equations. The BS equations were discretized non-uniformly in space and implicitly in time. This paper mainly focused on the pricing of the one-, two-, and three-dimensional BS equations using FDM. The two- and three-dimensional equations were solved using operator splitting method. In the numerical tests, we performed characteristic examples such as cash-or-nothing option pricing. The computational results were in good agreement with the closed-form solutions to the BS equations.

Author Contributions: All authors contributed equally. All authors have read and agreed to the published version of the manuscript.

Funding: The corresponding author (J.S. Kim) was supported by the Brain Korea 21 Plus (BK 21) from the Ministry of Education of Korea.

Acknowledgments: The authors are grateful to the reviewers for constructive and helpful comments on the revision of this article.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

The following MATLAB codes are available from the corresponding author's webpage: http://elie.korea.ac.kr/~cfdkim/codes/.

FDM Code for Cash-or-Nothing

```
% Numerical solution for cash-or-nothing
clear;x0=100;L=300;x=[0 1.5:4:77.5 80.5:3:119.5 122.5:4:L-1.5 L];
Nx=length(x);hx=diff(x);hx=[hx,hx(end)];
K=100;c=100;sigx=0.3;r=0.03;T=1;dt=0.5/365;Nt=round(T/dt);
ax=(r*x(3:Nx).*hx(3:Nx)-(sigx*x(3:Nx)).^2)./(hx(2:Nx-1).*(hx(2:Nx-1)+hx(3:Nx)));
bx=1/dt+((sigx*x(2:Nx)).^2-r*x(2:Nx).*(hx(2:Nx)-hx(1:Nx-1)))./(hx(2:Nx).*hx(1:Nx-1));
gx=(-r*x(2:Nx).*hx(1:Nx-1)-(sigx*x(2:Nx)).^2)./(hx(2:Nx).*(hx(1:Nx-1)+hx(2:Nx)));
bx(Nx-1)=bx(Nx-1)+gx(Nx-1);
A=diag(ax,-1)+diag(bx,0)+diag(gx(1:Nx-2),1);
dimension=3;
if dimension==1
Ax=A+eye(Nx-1)*r;
u=zeros(Nx,1);
for i=1:Nx
if x(i)>=K
u(i)=c;
end
end
for n=1:Nt
f=u(2:Nx)/dt;  u(2:Nx)= Ax\f;
end
oneD_price=interp1(x,u,x0,'linear')
else
y0=x0;y=x;Ny=Nx;hy=hx;rho=0.5;sigy=sigx;
ay=(r*y(3:Ny).*hy(3:Ny)-(sigy*y(3:Ny)).^2)./(hy(2:Ny-1).*(hy(2:Ny-1)+hy(3:Ny)));
by=1/dt+((sigy*y(2:Ny)).^2-r*y(2:Ny).*(hy(2:Ny)-hy(1:Ny-1)))...
./(hy(2:Ny).*hy(1:Ny-1));
gy=(-r*y(2:Ny).*hy(1:Ny-1)-(sigy*y(2:Ny)).^2)./(hy(2:Ny).*(hy(1:Ny-1)+hy(2:Ny)));
by(Ny-1)=by(Ny-1)+gy(Ny-1);
B=diag(ay,-1)+diag(by,0)+diag(gy(1:Ny-2),1);
Ax=A+eye(Nx-1)*0.5*r;Ay=B+eye(Ny-1)*0.5*r;
if dimension==2
u=zeros(Nx+1,Ny+1);
for i=1:Nx
for j=1:Ny
if x(i)>=K && y(j)>=K
u(i,j)=c;
end
end
end
u(2:Nx,Ny+1)=u(2:Nx,Ny); u(Nx+1,2:Ny+1)=u(Nx,2:Ny+1);
v=u;fx=zeros(Nx-1,1);fy=zeros(Ny-1,1);
for n=1:Nt
for j=2:Ny
for i=2:Nx
fx(i-1)=u(i,j)/dt+0.5*rho*sigx*sigy*x(i)*y(j)...
*(u(i+1,j+1)+u(i-1,j-1)-u(i-1,j+1)-u(i+1,j-1))...
/(hx(i-1)*hy(j)+hx(i)*hy(j)+hx(i)*hy(j-1)+hx(i-1)*hy(j-1));
end
v(2:Nx,j)=Ax\fx;
end
v(2:Nx,Ny+1)=v(2:Nx,Ny); v(Nx+1,2:Ny+1)=v(Nx,2:Ny+1);
for i=2:Nx
for j=2:Ny
fy(j-1)=v(i,j)/dt+0.5*rho*sigx*sigy*x(i)*y(j)*...
 (v(i+1,j+1)+v(i-1,j-1)-v(i-1,j+1)-v(i+1,j-1))...
```

```
/(hy(j-1)*hx(i)+hy(j)*hx(i)+hy(j)*hx(i-1)+hy(j-1)*hx(i-1));
end
u(i,2:Ny)=Ay\fy;
end
u(2:Nx,Ny+1)=u(2:Nx,Ny); u(Nx+1,2:Ny+1)=u(Nx,2:Ny+1);
end
twoD_price=interp2(x,y,u(1:Nx,1:Ny),x0,y0,'linear')
else
z0=x0;z=x;Nz=Nx;hz=hx;rhoxy=0.5;rhoyz=0.5;rhoxz=0.5;sigz=sigx;
az=(r*z(3:Nz).*hz(3:Nz)-(sigz*z(3:Nz)).^2)./(hz(2:Nz-1).*(hz(2:Nz-1)+hz(3:Nz)));
bz=1/dt+((sigz*z(2:Nz)).^2-r*z(2:Nz).*(hz(2:Nz)-hz(1:Nz-1)))./(hz(2:Nz).*hz(1:Nz-1));
gz=(-r*z(2:Nz).*hz(1:Nz-1)-(sigz*z(2:Nz)).^2)./(hz(2:Nz).*(hz(1:Nz-1)+hz(2:Nz)));
bz(Nz-1)=bz(Nz-1)+gz(Nz-1);
C=diag(az,-1)+diag(bz,0)+diag(gz(1:Nz-2),1);
Ax=A+eye(Nx-1)*r/3;Ay=B+eye(Ny-1)*r/3;Az=C+eye(Nz-1)*r/3;
u=zeros(Nx+1,Ny+1,Nz+1);
for i=1:Nx
for j=1:Ny
for k=1:Nz
if x(i)>=K && y(j)>=K && z(k)>=K
u(i,j,k)=c;
end
end
end
end
u(2:Nx,2:Ny,Nz+1)=u(2:Nx,2:Ny,Nz);
u(Nx+1,2:Ny,2:Nz+1)=u(Nx,2:Ny,2:Nz+1);
u(2:Nx+1,Ny+1,2:Nz+1)=u(2:Nx+1,Ny,2:Nz+1);
v=u;fx=zeros(Nx-1,1);fy=zeros(Ny-1,1);fz=zeros(Nz-1,1);
for n=1:Nt
for j=2:Ny
for k=2:Nz
for i=2:Nx
fx(i-1)=(1/3)*(rhoxy*sigx*sigy*x(i)*y(j)...
*(u(i+1,j+1,k)-u(i+1,j-1,k)-u(i-1,j+1,k)+u(i-1,j-1,k))...
/(hx(i-1)*hy(j)+hx(i)*hy(j)+hx(i)*hy(j-1)+hx(i-1)*hy(j-1))...
+rhoxz*sigx*sigz*x(i)*z(k)*(u(i+1,j,k+1)-u(i+1,j,k-1)-u(i-1,j,k+1)+u(i-1,j,k-1))...
/(hx(i-1)*hz(k)+hx(i)*hz(k)+hx(i)*hz(k-1)+hx(i-1)*hz(k-1))...
+rhoyz*sigy*sigz*y(j)*z(k)*(u(i,j+1,k+1)-u(i,j+1,k-1)-u(i,j-1,k+1)+u(i,j-1,k-1))...
/(hy(j-1)*hz(k)+hy(j)*hz(k)+hy(j)*hz(k-1)+hy(j-1)*hz(k-1)))+u(i,j,k)/dt;
end
v(2:Nx,j,k)=Ax\fx;
end
end
v(2:Nx,2:Ny,Nz+1)=v(2:Nx,2:Ny,Nz);
v(Nx+1,2:Ny,2:Nz+1)=v(Nx,2:Ny,2:Nz+1);
v(2:Nx+1,Ny+1,2:Nz+1)=v(2:Nx+1,Ny,2:Nz+1);
for k=2:Nz
for i=2:Nx
for j=2:Ny
fy(j-1)=(1/3)*(rhoxy*sigx*sigy*x(i)*y(j)...
*(v(i+1,j+1,k)-v(i+1,j-1,k)-v(i-1,j+1,k)+v(i-1,j-1,k))...
/(hx(i-1)*hy(j)+hx(i)*hy(j)+hx(i)*hy(j-1)+hx(i-1)*hy(j-1))...
+rhoxz*sigx*sigz*x(i)*z(k)*(v(i+1,j,k+1)-v(i+1,j,k-1)-v(i-1,j,k+1)+v(i-1,j,k-1))...
/(hx(i-1)*hz(k)+hx(i)*hz(k)+hx(i)*hz(k-1)+hx(i-1)*hz(k-1))...
+rhoyz*sigy*sigz*y(j)*z(k)*(v(i,j+1,k+1)-v(i,j+1,k-1)-v(i,j-1,k+1)+v(i,j-1,k-1))...
/(hy(j-1)*hz(k)+hy(j)*hz(k)+hy(j)*hz(k-1)+hy(j-1)*hz(k-1)))+v(i,j,k)/dt;
end
u(i,2:Ny,k)=Ay\fy;
end
end
u(2:Nx,2:Ny,Nz+1)=u(2:Nx,2:Ny,Nz);
u(Nx+1,2:Ny,2:Nz+1)=u(Nx,2:Ny,2:Nz+1);
```

```
u(2:Nx+1,Ny+1,2:Nz+1)=u(2:Nx+1,Ny,2:Nz+1);
for j=2:Ny
for i=2:Nx
for k=2:Nz
fz(k-1)=(1/3)*(rhoxy*sigx*sigy*x(i)*y(j)...
*(u(i+1,j+1,k)-u(i+1,j-1,k)-u(i-1,j+1,k)+u(i-1,j-1,k))...
/(hx(i-1)*hy(j)+hx(i)*hy(j)+hx(i)*hy(j-1)+hx(i-1)*hy(j-1))...
+rhoxz*sigx*sigz*x(i)*z(k)*(u(i+1,j,k+1)-u(i+1,j,k-1)- u(i-1,j,k+1)+u(i-1,j,k-1))...
/(hx(i-1)*hz(k)+hx(i)*hz(k)+hx(i)*hz(k-1)+hx(i-1)*hz(k-1))...
+rhoyz*sigy*sigz*y(j)*z(k)*(u(i,j+1,k+1)-u(i,j+1,k-1)-u(i,j-1,k+1)+u(i,j-1,k-1))...
/(hy(j-1)*hz(k)+hy(j)*hz(k)+hy(j)*hz(k-1)+hy(j-1)*hz(k-1)))+u(i,j,k)/dt;
end
v(i,j,2:Nz)=Az\fz;
end
end
v(2:Nx,2:Ny,Nz+1)=v(2:Nx,2:Ny,Nz);
v(Nx+1,2:Ny,2:Nz+1)=v(Nx,2:Ny,2:Nz+1);
v(2:Nx+1,Ny+1,2:Nz+1)=v(2:Nx+1,Ny,2:Nz+1);
u=v;
end
threeD_price=interp3(x,y,z,u(1:Nx,1:Ny,1:Nz),x0,y0,z0,'linear')
end
end
```

References

1. Black, F.; Scholes, M. The pricing of options and corporate liabilities. *J. Political Econ.* **1973**, *81*, 637–654. [CrossRef]
2. Merton, R. Theory of Rational Option Pricing. *Bell J. Econ.* **1973**, *4*, 141–183. [CrossRef]
3. Bustamante, M.; Contreras, M. Multi-asset Black–Scholes model as a variable second class constrained dynamical system. *Phys. A* **2016**, *457*, 540–572. [CrossRef]
4. Khodayari, L.; Ranjbar, M. A computationally efficient numerical approach for multi-asset option pricing. *Int. J. Comput. Math.* **2019**, *96*, 1158–1168. [CrossRef]
5. Burden, R.L.; Faires, J.D. *Numerical Analysis*, 6th ed.; Brooks/Cole Publishing: Pacific Grove, CA, USA, 1997.
6. Jeong, D.; Yoo, M.; Kim, J. Finite difference method for the Black–Scholes equation without boundary conditions. *Comput. Econ.* **2018**, *51*, 961–972. [CrossRef]
7. Hout, K.I.T.; Valkov, R. Numerical solution of a two-asset option valuation PDE by ADI finite difference discretization. *AIP Conf. Proc.* **2015**, *1648*, 020007.
8. McCartin, B.J.; Labadie, S.M. Accurate and efficient pricing of vanilla stock options via the Crandall–Douglas scheme. *Appl. Math. Comput.* **2003**, *143*, 39–60. [CrossRef]
9. Ankudinova, J.; Ehrhardt, M. On the numerical solution of nonlinear Black–Scholes equations. *Comput. Math. Appl.* **2008**, *56*, 799–812. [CrossRef]
10. Tangman, D.Y.; Gopaul, A.; Bhuruth, M. Numerical pricing of options using high-order compact finite difference schemes. *J. Comput. Appl. Math.* **2008**, *218*, 270–280. [CrossRef]
11. Koleva, M.N.; Mudzimbabwe, W.; Vulkov, L.G. Fourth-order conmpact finite schemes for a parabolic-ordinary system of European option pricing liquidity shock model. *Numer. Algorithms* **2017**, *74*, 59–75. [CrossRef]
12. Zhao, H.; Tian, H. Finite difference methods of the spatial fractional Black–Schloes equation for a European call option. *IMA J. Appl. Math.* **2017**, *82*, 836–848. [CrossRef]
13. Chen, W.; Wang, S. A 2nd-order ADI finite difference method for a 2D fractional Black–Scholes equation governing European two asset option pricing. *Math. Comput. Simul.* **2019**. [CrossRef]
14. Sawangtong, P.; Trachoo, K.; Sawangtong, W.; Wiwattanapataphee, B. The Analytical Solution for the Black–Scholes Equation with Two Assets in the Liouville–Caputo Fractional Derivative Sense. *Mathematics* **2018**, *6*, 129. [CrossRef]
15. Zhang, H.; Liu, F.; Turner, I.; Yang, Q. Numerical solution of the time fractional Black–Scholes model governing European options. *Comput. Math. Appl.* **2016**, *71*, 1772–1783. [CrossRef]

16. Hu, J.; Gan, S. High order method for Black–Scholes PDE. *Comput. Math. Appl.* **2018**, *75*, 2259–2270. [CrossRef]
17. Hendricks, C.; Heuer, C.; Ehrhardt, M.; Gunther, M. High-order ADI finite difference schemes for parabolic equations in the combination technique with application in finance. *J. Comput. Appl. Math.* **2017**, *316*, 175–194. [CrossRef]
18. Rao, S.C.S. Numerical solution of generalized Black–Scholes model. *Appl. Math. Comput.* **2018**, *321*, 401–421.
19. Ullah, M.Z. Numerical Solution of Heston–Hull–White Three-Dimensional PDE with a High Order FD Scheme. *Mathematics* **2019**, *7*, 704. [CrossRef]
20. Gulen, S.; Popescu, C.; Sari, M. A New Approach for the Black–Scholes Model with Linear and Nonlinear Volatilities. *Mathematics* **2019**, *7*, 760. [CrossRef]
21. Jeong, D.; Kim, Y.; Hwang, H.; Yoo, M.; Kim, J. *Derivative Programming (Korean Edition)*; Kyeongmunsa: Seoul, Korea, 2015; Volume 1, ISBN 9788961058940.
22. Jeong, D.; Yoo, M.; Yoo, C.; Kim, J. A hybrid monte carlo and finite difference method for option pricing. *Comput. Econ.* **2019**, *53*, 111–124. [CrossRef]
23. Pak, D.; Han, C.; Hong, W.T. Iterative Speedup by Utilizing Symmetric Data in Pricing Options with Two Risky Assets. *Symmetry* **2017**, *9*, 12. [CrossRef]
24. Bodeau, J.; Riboulet, G.; Roncalli, T. Non-Uniform Grids for PDE in Finance. *SSRN Electron. J.* **2000**. [CrossRef]
25. Haug, E.G. *The Complete Guide to Option Pricing Formulas*; McGraw-Hill: New York, NY, USA, 2007; Volume 2.
26. Genz, A. Numerical computation of rectangular bivariate and trivariate normal and t probabilities. *Stat. Comput.* **2004**, *14*, 251–260. [CrossRef]
27. Choi, Y.; Jeong, D.; Kim, J.; Kim, Y.R.; Lee, S.; Seo, S.; Yoo, M. Robust and accurate method for the Black–Scholes equations with payoff-consistent extrapolation. *Commun. Korean Math. Soc.* **2015**, *30*, 297–311. [CrossRef]
28. Jeong, D.; Seo, S.; Hwang, H.; Lee, D.; Choi, Y.; Kim, J. Accuracy, robustness, and efficiency of the linear boundary condition for the Black–Scholes equations. *Discrete Dyn. Nat. Soc.* **2015**, *2015*, 359028. [CrossRef]