Exponential Time Integration and Second-Order Difference Scheme for a Generalized Black-Scholes Equation

A Project Report submitted

for the Course

MA473: Computational Finance

by

Shashank Goyal	**(180123042)**
Tejus Singla	**(180123061)**
Jay Sabale	**(180123019)**

under

Dr Natesan Srinivasan

Professor

Department of Mathematics

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI - 781039, INDIA

November 2021

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Derived securities, or contingent claims, are of great practical significance when it comes to pricing and hedging them. Option is an example of a derivative. Call options allow the owner the right, but not the obligation, to purchase an underlying asset, such as a stock, for a specified price, known as the exercise price or strike price, on or before an expiration date.

Using the time horizon and underlying price as inputs, Black and Scholes show that these option prices satisfy a second order partial differential equation. As a result, it is now known as the Black-Scholes equation but it can only be solved exactly with constant coefficients or space-independent coefficients. Euler transformations are commonly used to solve the Black-Scholes equation. This may lead to computational errors, however, as the left-hand side of the domain is truncated to artificially remove the degeneracy. The uniform mesh will also cause the grid points to concentrate around x = 0 unnecessarily on the transformed interval. In addition, when a problem is space-dependent, this transformation is not possible, and hence the Original Black-Scholes equation must be solved.

Here, by changing the grid spacing, the author implemented a piece-wise uniform mesh which is constructed so that the central difference is applied everywhere. It is easy to implement explicit schemes for time discretization, but they have instability issues. It is well-known that some well-known second-order implicit schemes, such as Crank-Nicolson, are susceptible to spurious oscillations unless the time step size is no greater than two times the time step size for an explicit method. Due to recent innovations in efficient matrix exponential computation methods, exponential time integration is likely to be a popular choice for solving large semi-discrete systems arising in various numerical computations.

On a piece-wise uniform mesh with respect to the spatial dimension, we apply a time integration scheme combining exponential and central differences. The scheme is found to be unconditionally stable. This conclusion is supported by numerical results.

1.1 Method description

We consider the following generalized Black-Scholes equation

$$\frac{\partial v}{\partial t} - \frac{1}{2}\sigma^2(x,t)x^2\frac{\partial^2 v}{\partial x^2} - r(t)x\frac{\partial v}{\partial x} + r(t)v = 0, \quad (x,t) \in \mathbb{R}^+ \times (0,T)$$

$$v(x,0) = \max(x-K,0), \quad x \in \mathbb{R}^+$$

$$v(0,t) = 0, \quad t \in [0,T]$$

Here v(x, t) is the European call option price at asset price x and at time to maturity t, K is the exercise price, T is the maturity, r(t) is the risk-free interest rate, and $\sigma(x,t)$ represents the volatility function of underlying asset. Here, we assume that $\sigma^2 \geq \alpha \gneqq 0$ and $\beta_* \geq r \geq \beta \gneqq 0$. When σ and r are constant functions, it becomes the classical Black-Scholes model.

The finite difference scheme may fail to converge as a result of the initial conditions not being smooth.

1.1.1 Smoothing the initial condition

Define $\pi_\epsilon(y)$ as

$$\pi_\varepsilon(y) = \begin{cases} y, & y \geq \varepsilon \\ c_0 + c_1 y + \cdots + c_9 y^9, & -\varepsilon < y < \varepsilon \\ 0, & y \leq -\varepsilon \end{cases}$$

where $0 \leq \epsilon$ <<1 is a transition parameter and $\pi_\epsilon(y)$ is a function which smooths out the original max(y, 0) around y = 0. This requires that $\pi_\epsilon(y)$ satisfies

$$\pi_\varepsilon(-\varepsilon) = \pi_\varepsilon'(-\varepsilon) = \pi_\varepsilon''(-\varepsilon) = \pi_\varepsilon'''(-\varepsilon) = \pi_\varepsilon^{(4)}(-\varepsilon) = 0$$

$$\pi_\varepsilon(\varepsilon) = \varepsilon, \pi_\varepsilon'(\varepsilon) = 1, \pi_\varepsilon''(\varepsilon) = \pi_\varepsilon'''(\varepsilon) = \pi_\varepsilon^{(4)}(\varepsilon) = 0$$

Using these ten conditions we can easily find that

$$\begin{array}{llll} c_0 = \frac{35}{256}\varepsilon, & c_1 = \frac{1}{2}, & c_2 = \frac{35}{64\varepsilon}, & c_4 = -\frac{35}{128\varepsilon^3}, \\ c_6 = \frac{7}{64\varepsilon^5}, & c_8 = -\frac{5}{256\varepsilon^7}, & c_3 = c_5 = c_7 = c_9 = 0. & \end{array}$$

In order to apply the numerical method we need to truncate the infinite domain. Wilmott et al.'s estimated that the upper bound of the asset price is typically three or four times the strike price.

1.1.2 The scheme

When a uniform mesh is used to compute a solution, central difference schemes may create nonphysical oscillations. Piece-wise uniform meshes are used to overcome these oscillations.

$$x_i = \begin{cases} h & i = 1 \\ h\left[1 + \frac{\alpha}{\beta^*}(i-1)\right] & i = 2, \ldots, \frac{N}{4} - 1 \\ K & i = \frac{N}{4} \\ K + \varepsilon & i = \frac{N}{4} + 1 \\ K + \varepsilon + \frac{S_{\max} - K - \varepsilon}{3N/4 - 1}(I - N/4 - 1) & i = \frac{N}{4} + 2, \ldots, N \end{cases}$$

where

$$h = \frac{K - \varepsilon}{1 + (\alpha/\beta^*)\,(N/4 - 2)}$$

For treating the non smoothness of the payoff function, we have used a refined mesh near x = K. On the piece-wise uniform mesh above, we discretize the generalized Black-Scholes operator by using a central difference scheme:

$$L^N U_i(t) = \frac{dU_i(t)}{dt} - \frac{\sigma_i^2(t)x_i^2}{h_i + h_{i+1}}\left(\frac{U_{i+1}(t) - U_i(t)}{h_{i+1}} - \frac{U_i(t) - U_{i-1}(t)}{h_i}\right)$$
$$-r(t)x_i\frac{U_{i+1}(t) - U_{i-1}(t)}{h_i + h_{i+1}} + r(t)U_i(t)$$

for i = 1, . . . , N - 1. This discretization leads to an initial value problem of the form

$$\frac{d\mathbf{U}}{dt} = \mathbf{A}(t)\mathbf{U}(t) + \mathbf{f}(t), \quad \mathbf{U}(0) = \pi_\varepsilon(\mathbf{x} - \mathbf{K})$$

where $U(t) = (U_1(t), ..., U_{N-1}(t))^T$, the matrix A(t) of order (N - 1) is given by

$$\mathbf{A}(t)=\begin{bmatrix} b_1 & c_1 & 0 & & \cdots & 0 \\ a_2 & b_2 & c_2 & \cdots & & \\ \vdots & a_3 & b_3 & c_3 & \cdots & \\ & \vdots & \vdots & \vdots & \vdots & \\ & \cdots & a_{N-2} & b_{N-2} & c_{N-2} & \\ 0 & \cdots & & & a_{N-1} & b_{N-1} \end{bmatrix}$$

where

$$a_i(t)=\frac{\sigma_i^2(t)x_i^2}{\left(h_i+h_{i+1}\right)h_i}-\frac{r(t)x_i}{h_i+h_{i+1}},\quad b_i(t)=-\frac{\sigma_i^2(t)x_i^2}{h_ih_{i+1}}-r(t)$$

$$c_i(t)=\frac{\sigma_i^2(t)x_i^2}{\left(h_i+h_{i+1}\right)h_{i+1}}+\frac{r(t)x_i}{h_i+h_{i+1}}\qquad \text{for } i=1,\ldots,N-1$$

The vectors f(t) and $\pi_\epsilon(x-K)$ are the corresponding boundary and initial conditions:

$$\mathbf{f}(t)=\begin{pmatrix} a_1U_0(t) \\ 0 \\ \vdots \\ 0 \\ c_{N-1}U_N(t) \end{pmatrix},\quad \pi_\varepsilon(\mathbf{x}-\mathbf{K})=\begin{pmatrix} \pi_\varepsilon\left(x_1-K\right) \\ \pi_\varepsilon\left(x_2-K\right) \\ \vdots \\ \pi_\varepsilon\left(x_{N-1}-K\right) \end{pmatrix}$$

Solving the initial value problem and using a numerical integration rule for evaluating the quadrature gives:

$$\mathbf{U}(t+l)\approx e^{l\mathbf{A}(t)}\left[\mathbf{U}(t)+\int_t^{t+l}e^{-(s-t)\mathbf{A}(t)}\mathbf{f}(s)\mathrm{d}s\right]$$

$$=e^{l\mathbf{A}(t)}\mathbf{U}(t)+\int_t^{t+l}e^{(t+l-s)\mathbf{A}(t)}\mathbf{f}(s)\mathrm{d}s$$

$$\approx e^{l\mathbf{A}(t)}\mathbf{U}(t) + \int_t^{t+l} e^{(t+l-s)\mathbf{A}(t)} \left[l^{-1}(t+l-s)\mathbf{f}(t) + l^{-1}(s-t)\mathbf{f}(t+l)\right] \mathrm{d}s$$

$$= e^{l\mathbf{A}(t)}\mathbf{U}(t) + (l\mathbf{A}(t))^{-1} \left[le^{l\mathbf{A}(t)} + \mathbf{A}^{-1}(t) - \mathbf{A}^{-1}(t)e^{l\mathbf{A}(t)}\right] \mathbf{f}(t)$$

$$+(l\mathbf{A}(t))^{-1} \left[\mathbf{A}^{-1}(t)e^{l\mathbf{A}(t)} - lI - \mathbf{A}^{-1}(t)\right] \mathbf{f}(t+l), \quad t = 0, l, 2l, \ldots$$

Now the problem is how to approximate $e^{lA(t)}$ to get numerical solution. The numerical method to be employed here is based on the use of the following second-order rational approximation:

$$e^z \approx \frac{1+(1-c)z}{1-cz+(c-1/2)z^2}$$

The recursion becomes

$$U(t+l) = R(l\mathbf{A}(t))U(t) + \frac{l}{2}[V(l\mathbf{A}(t))\mathbf{f}(t) + W(l\mathbf{A}(t))\mathbf{f}(t+l)]$$

for t = 0, l, 2l, ..., M, where

$$V(l\mathbf{A}(t)) = \left[I - cl\mathbf{A}(t) + \left(c - \frac{1}{2}\right) l^2\mathbf{A}^2(t)\right]^{-1}$$

$$W(l\mathbf{A}(t)) = \left[I - cl\mathbf{A}(t) + \left(c - \frac{1}{2}\right) l^2\mathbf{A}^2(t)\right]^{-1} \left[I - 2\left(c - \frac{1}{2}\right) l\mathbf{A}(t)\right]$$

It can be shown that the truncation error of the difference scheme with 1/2 <c <2$-\sqrt{2}$ is $O(h^2 + l^2)$.

Chapter 2

Numerical results

Since we don't have the exact solution, we have implemented the implicit Euler method given in paper '*A robust and accurate finite difference method for a generalized Black-Scholes equation*' (which the author have also used) and we take this as exact solution with N = 1024 and M = 1024, since we know this method converges. Because we only know "the exact solution" on mesh points, we use the linear interpolation to get solutions at other points. The file 'exact_robust.m' contains the implementation of implicit Euler method used as exact solution. Exponential time integration and second order difference scheme is implemented with **'Exp_1.m'** and **'Exp_2.m'** files.

The file 'Exp_1.m' plots European option price with parameters $\sigma = 0.2 + 0.2(1 - t)((x/25 - 1.2)^2/((x/25)^2 + 1.44))$, r = 0.06, T = 1, K = 25 and $S_{max} = 100$ and calculates error and convergence rate. Output of the program is given in fig 2.1 and table 2.1.

Table 2.1: Numerical results for Test 1

M	N	Exp Error	Exp Rate	Implicit Error	Implicit rate
16	64	0.37941	2.3351	0.14668	1.0995
32	128	0.032271	1.2515	0.068448	1.2224
64	256	0.02959	–	0.029335	–

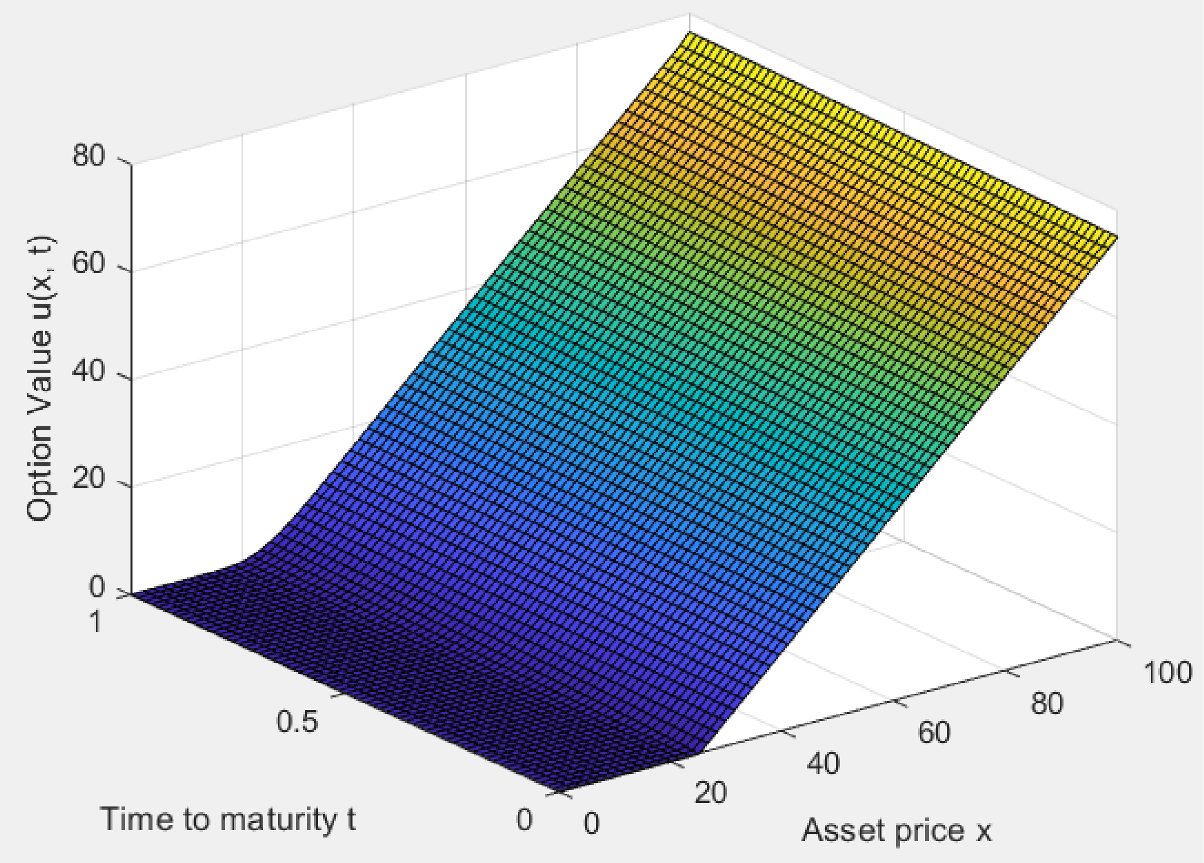

Figure 2.1: Computed option value for Test 1

The file **'Exp_2.m'** plots the European option price with parameters $\sigma = 0.2(1 + (0.1(1-t))(x/(1+x)))$, r = 0.06, T = 1, K = 25 and $S_{max} = 100$ and also calculates the error and convergence rate. Output of the program is given in fig 2.2 and table 2.2.

Table 2.2: Numerical results for Test 2

M	N	Exp Error	Exp Rate	Implicit Error	Implicit rate
16	64	0.38644	2.2826	0.14668	1.0995
32	128	0.032989	1.2232	0.068448	1.2224
64	256	0.030307	–	0.029335	–

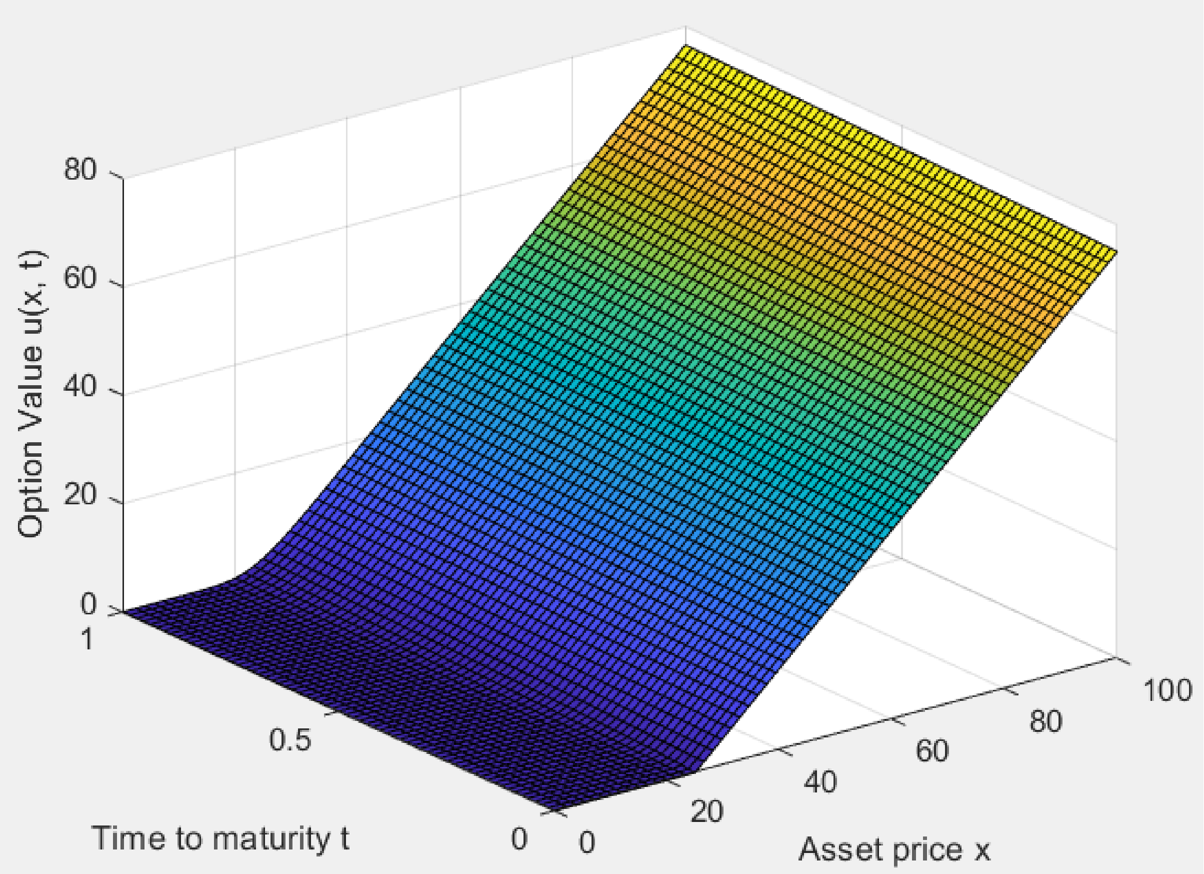

Figure 2.2: Computed option value for Test 2

By looking at the figures, it is evident that the numerical solutions by the method are non-oscillatory. Tables 2.1 and 2.2 show that the exponential time integration method converges more rapidly than the implicit Euler method. Nevertheless, exponential time integration method requires more computer time than implicit Euler method for the same M and N. Although it cannot be reached by large M and N, exponential time integration rate is close to 2 due to the fact that there is no actual ”exact solution”.