Fast Method for Black-Scholes Equation of European Options

Haiyan Song, Haoxi Jia

***Abstract*—The Black-Scholes equation is a well known mathematic model in option pricing theory and for European options it is a diffusion PDE with solution $V(S,t)$, a function of current value of the underlying asset S and time t. In theory, the range of S is $S \in (0,\infty)$ and in practical computation we truncate the infinity interval into a finite one, such as $S \in (0, S_{\max})$, where $S_{\max}$ is often a large quantity. In this paper, we use the Crank-Nicolson method to compute the numerical solution, which consists of a centered finite difference method with mesh size ΔS for the derivatives V_S and V_{SS} and a trapezoidal rule to the time derivative V_t. We have to solve a triangular linear system at each time step, which would be a serious computation burden when $N = S_{\max}/\Delta S$ is large. To accelerate the computation we use the Crout factorization of the triangular matrix, by using the diagonal dominance. Numerical results for both the put option and the call option are given, which illustrate the advantage of the Crout factorization in terms of CPU time, compared to the built-in command `inv` in Matlab.**

***Index Terms*—Black-Scholes equation, European options, Crank-Nicolson method, Crout factorization, diagonal dominance.**

I. Introduction

In the financial market, pricing an option is an important problem in the view of both theory and practice. The Black–Scholes (BS) model, which was proposed in 1973 by Black and Scholes [3] and Merton [18], provides an approximate description of the behavior of the underlying asset. Most remarkably, the BS model leads to a boom in options trading because of its simplicity and clarity in obtaining the price of the option [24]. There are two classes of options, the European options and the American options. For the former, the BS equation is a boundary value problem of a diffusion equation, while for the American options the BS equation is a free boundary value problem. In this paper, we focus on the BS model for the European options and a generalization of our algorithm to the American options will be discussed in our forthcoming paper.

The BS equation can be solved by both analytically and numerically. Black and Scholes (1973) first found the solution based on previous research on option pricing that gave an idea of what the solution would look like. In [15] Mellin transformation was utilized to explain this model. Such a transformation did not require variable change or explaining dispersion condition. Company, Gonzalez and Jodar [5] solved the BS model which was modified with discrete dividend. They utilized a delta-characterizing grouping of generalized Dirac-Delta function and connected the Mellin transformation to get an integral formula. However, in these studies the closed form for the analytic solutions is only available for BS equations with constant coefficients. For BS model with *time-dependent* coefficients [11,12,14,20] there is no closed form for the analytic solution and we have to rely on numerical computation [4,6,9,10,19] .

For the European options, the BS model is the following partial differential equation (PDE) about the unknown function $V(S,t)$ with $(S,t) \in (0,\infty) \times (0,T)$:

$$V_t + \frac{\sigma^2 S^2}{2} V_{SS} + rSV_S - rV = 0, \tag{1}$$

together with final-value condition $V(S,T) = V_T(S)$ and suitable boundary conditions. This PDE is backward propagation in time and it is not convenient to deal with. Usually, we can make a variable change to transform it to a forward propagation problem. To this end, we let

$$\tilde{t} = T - t,$$

and by noticing $t = T - \tilde{t}$ we have

$$V_{\tilde{t}} = \frac{\partial V}{\partial \tilde{t}} = \frac{\partial V}{\partial t}\frac{\partial t}{\partial \tilde{t}} = -\frac{\partial V}{\partial t}, \text{ i.e., } V_t = -V_{\tilde{t}}.$$

Substituting this into (1) gives

$$V_{\tilde{t}} - \frac{\sigma^2 S^2}{2} V_{SS} - rSV_S + rV = 0,$$

and the final-value conditions is changed to $V(S,\tilde{t}) = V_T(S)$ for $\tilde{t} = 0$. For uniformness, we still use the variable t to replace $\tilde{t}$ and focus on the following PDE

$$V_t - \frac{\sigma^2 S^2}{2} V_{SS} - rSV_S + rV = 0, \tag{2}$$

together with $V(S,0 = V_T(S)$ (initial-value condition) and some corresponding boundary conditions. For the BS model (2), the notations are explained as follows. The quantity V is the option value: when the distinction is important we use $C(S,t)$ to denote a call option and $P(S,t)$ to denote a put option. This value is a function of current value of the underlying asset S and time t. The value of the option also depends on the parameters σ (the volatility of the underlying asset), E (the exercise price), r (the interest rate) and T (the expiry time).

In a practical application, the initial and boundary conditions for the BS model are specified by the put option or call option. For an European put option, the symbol V is usually denoted by P, i.e.,

$$P_t - \frac{\sigma^2 S^2}{2} P_{SS} - rSP_S + rP = 0, \tag{3a}$$

together with initial and boundary conditions

$$\begin{aligned} &P(S,0) = \max\{E - S, 0\}, && S \in (0,\infty), \\ &P(0,t) = Ee^{-rt}, \ \lim_{S\to\infty} \frac{P(S,t)}{S} = 0, && t \in (0,T). \end{aligned} \tag{3b}$$

Manuscript received June 21, 2022; revised October 10, 2022.

Haiyan Song is a lecturer at School of Computer and Data Engineering, NingboTech University, Ningbo 315100, China. E-mail: *haiyansong@nbt.edu.cn*.

Haoxi Jia is a master student at Western Bank, Management School, The University of Sheffield, Sheffield S10 2TN, the Untied Kingdom. E-mail: *haoxij@hotmail.com*

For the call option, the BS model is specified by a PDE with solution $C(S,t)$:

$$C_t - \frac{\sigma^2 S^2}{2} C_{SS} - rSC_S + rC = 0, \tag{4a}$$

together with initial and boundary conditions

$$\begin{aligned} &C(S,0) = \max\{S-E,0\}, && S \in (0,\infty), \\ &C(0,t) = 0, \ \lim_{S\to\infty} \frac{C(S,t)}{S} = 1, && t \in (0,T). \end{aligned} \tag{4b}$$

To make a practical computation, we have to restrict ourself to a finite range for S, i.e., $S \in [0, S_{\max}]$. The quantity $S_{\max}$ should be as large as possible and in practice it stands for the maximal price of the underlying asset and according to [16] a simple rule for fixing $S_{\max}$ is to let $S_{\max}$ be around four times of the exercise price E, i.e., $S_{\max} = 4E$. With a finite range of S, we have to put suitable boundary conditions at $S = S_{\max}$ to replace $\lim_{S\to\infty} \frac{P(S,t)}{S} = 0$ and $\lim_{S\to\infty} \frac{C(S,t)}{S} = 1$. According to the analysis in [16], we choose the following boundary condition at $S = S_{\max}$ for the put and call options:

$$P(S_{\max},t) = 0, \ C(S_{\max},t) = S_{\max} - Ee^{-rt}. \tag{5}$$

Normally, this choice of boundary condition leads to a negligible error in the value of the option. Clearly, for the put and call options the governing PDE is the same and therefore a discretization formula (2) is applicable to both (3a)-(3b) and (4a)-(4b). The numerical method studied in this paper is also applicable to the following BS equations with time-varying coefficients

$$V_t - \frac{\sigma^2(t) S^2}{2} V_{SS} - r(t) S V_S + r(t) V = 0, \tag{6}$$

which is the so-called generalized BS equation[12,20] and is found useful in many application fields, such as the variable volatility driven BS option pricing [8] and double barrier option pricing [9].

In this paper, we use the Crank-Nicolson method to solve the BS equation (6) (together with the boundary conditions specified in (4a)-(4b) at $S = 0$ and (5) at $S = S_{\max}$), which consists of applying a centered finite difference scheme to the first and second derivatives V_S and V_{SS} and applying the trapezoidal rule to the time derivative V_t. At each step of the Crank-Nicolson method, we have to solve a linear system with a coefficient matrix of triangular structure. For long time computation, solving this linear system would be the major computation burden, especially when the number of mesh sizes, i.e., $N = S_{\max}/\Delta S$, is large. By proving the diagonal dominance of the coefficient matrix, we show that this matrix permits a stable Crout factorization, by which we can handle the large scale linear system via fast forward and backward substitutions. Numerical results indicate that such an algorithm is more efficient than the built-in command `inv` in Matlab, in terms of stability and CPU time.

The remainder of this paper is organized as follows. In Section II, by using the trapezoidal rule as the time discretization and the centered finite difference formula as the space discretization, we consider a fully discretized version of (2) and present the discrete formula. At the discrete level, we have to a linear system at each time point (the major computation cost) and in Section III we describe a fast Thomas method based on the Crout factorization for such a system. Our numerical results are given in Section IV, where we will consider concrete data for both the put model (3a)-(3b) and the call model (4a)-(4b). We finish this paper in Section V with some concluding remarks.

II. The Crank-Nicolson discretization

In this section, we establish numerical formula for solving (6). The main point is to discretize the time and space derivatives, for which we use the trapezoidal rule and the centered finite difference formula, respectively. Such a Crank-Nicolson scheme is successfully applied to many other problems, such as Fisher-Kolmogorov equation [25], fractional problems[7,17] and transient diffusion convection reaction problems [2]. To setup the space and time discretizations, we first partition the computation domain $[0, S_{\max}] \times [0, T]$ by mesh sizes ΔS and Δt and denote an arbitrary grid on this domain by $(n\Delta S, m\Delta t)$, where $n = 0, 1, \ldots, N$ and $m = 0, 1, \ldots, M$.

We first consider the time discretization via the trapezoidal rule

$$\begin{aligned} &V^{[m+1]}(S) - V^{[m]}(S) = \frac{\Delta t}{2}(L^{[m]}(S) + L^{[m+1]}(S)), \\ &m = 0, 1, \ldots, M-1, \end{aligned} \tag{7}$$

where $L^{[l]}(S) = \frac{\sigma_l^2 S^2}{2} V_{SS}^{[l]}(S) + r_l S V_S^{[l]}(S) - r_l V^{[l]}(S)$ $(l = m, m+1)$ with $V^{[l]}(S)$ being the approximate solution of $V(S,t)$ and $t = t_l$, $\sigma_l = \sigma(t_l)$ and $r_l = r(t_l)$. Next, we discretize the spatial derivatives $V_{SS}^{[l]}(S)$ and $V_S^{[l]}(S)$ at $S = S_n$ by the centered finite difference method:

$$\begin{aligned} V_S^{[l]}(S_n) &= \frac{V^{[l]}(S_{n+1}) - V^{[l]}(S_{n-1})}{2\Delta S} + \tau_n, \ \tau_n = \mathcal{O}(\Delta S^2), \\ V_{SS}^{[l]}(S_n) &= \frac{\frac{V^{[l]}(S_{n+1}) - V^{[l]}(S_n)}{\Delta S} - \frac{V^{[l]}(S_n) - V^{[l]}(S_{n-1})}{\Delta S}}{\Delta S} + \tau_n \\ &= \frac{V^{[l]}(S_{n+1}) - 2V^{[l]}(S_n) + V^{[l]}(S_{n-1})}{\Delta S^2} + \tau_n, \end{aligned}$$

where $\tau_n = \mathcal{O}(\Delta S^2)$ is the truncation error. Dropping this truncation error in the above formulas gives the approximations as

$$\begin{aligned} V_S^{[l]}(S_n) &\approx \frac{V_{n+1}^{[l]} - V_{n-1}^{[l]}}{2\Delta S}, \\ V_{SS}^{[l]}(S_n) &\approx \frac{V_{n+1}^{[l]} - 2V_n^{[l]} + V_{n-1}^{[l]}}{\Delta S^2}. \end{aligned} \tag{8}$$

At $S = S_n$, the semi-discrete BS model is

$$V^{[m+1]}(S_n) - V^{[m]}(S_n) = \frac{\Delta t}{2}(L^{[m]}(S_n) + L^{[m+1]}(S_n)),$$

which after replacing $V_S^{[l]}(S_n)$ and $V_{SS}^{[l]}(S_n)$ by the centered finite difference approximation in (8) gives

$$\begin{aligned} &V_n^{[m+1]} - V_n^{[m]} = \frac{\Delta t}{2}\left(L_n^{[m]} + L_n^{[m+1]}\right), \\ &m = 0, 1, \ldots, M-1, \ n = 1, 2, \ldots, N-1, \\ &L_n^{[l]} := \frac{\sigma_l^2 S_n^2}{2} \frac{V_{n+1}^{[l]} - 2V_n^{[l]} + V_{n-1}^{[l]}}{\Delta S^2} + \\ &\qquad r_l S_n \frac{V_{n+1}^{[l]} - V_{n-1}^{[l]}}{2\Delta S} - r_l V_n^{[l]}, \ l = m, m+1. \end{aligned}$$

Let

$$\begin{aligned}\alpha_n^{[l]} &= \frac{\Delta t\sigma_l^2 S_n^2}{4\Delta S^2} + \frac{r_l \Delta t S_n}{4\Delta S},\\ \beta_n^{[l]} &= \frac{\Delta t\sigma_l^2 S_n^2}{4\Delta S^2} + \frac{r_l \Delta t}{4},\\ \gamma_n^{[l]} &= \frac{\Delta t\sigma_l^2 S_n^2}{4\Delta S^2} - \frac{r_l \Delta t S_n}{4\Delta S}.\end{aligned} \tag{9}$$

Then, the discrete BS-model is

$$\begin{aligned}&V_n^{[m+1]} - V_n^{[m]} = \alpha_n^{[m]} V_{n+1}^{[m]} - 2\beta_n^{[m]} V_n^{[m]} + \gamma_n^{[m]} V_{n-1}^{[m]} +\\ &\alpha_n^{[m+1]} V_{n+1}^{[m+1]} - 2\beta_n^{[m+1]} V_n^{[m+1]} + \gamma_n^{[m+1]} V_{n-1}^{[m+1]},\end{aligned} \tag{10}$$

where $n = 1, 2, \ldots, N-1$. For $n = 1$ and $n = N-1$, we have to use the boundary conditions to specify $V_0^{[l]}$ and $V_N^{[l]}$ $(l = m, m+1)$:

$$V_0^{[l]} = V_a(t_l),\ V_0^{[l]} = V_b(t_l).$$

Define the following notations

$$\boldsymbol{V}^{[m]} = \left((V_1^{[m]})^\top, (V_2^{[m]})^\top, \ldots, (V_{N-1}^{[m]})^\top\right)^\top,$$

$$\boldsymbol{c}_{m,m+1} = \begin{bmatrix} \gamma_1^{[m]} V_a(t_m) + \gamma_1^{[m+1]} V_a(t_{m+1}) \\ 0 \\ \vdots \\ 0 \\ \alpha_{N-1}^{[m]} V_b(t_m) + \alpha_{N-1}^{[m+1]} V_b(t_{m+1}) \end{bmatrix},$$

$$A^{[l]} = \begin{bmatrix} -2\beta_1^{[l]} & \alpha_1^{[l]} & & & \\ \gamma_2^{[l]} & -2\beta_2^{[l]} & \alpha_2^{[l]} & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{N-2}^{[l]} & -2\beta_{N-2}^{[l]} & \alpha_{N-2}^{[l]} \\ & & & \gamma_{N-1}^{[l]} & -2\beta_{N-1}^{[l]} \end{bmatrix},$$

where $l = m, m+1$. From (10) we have

$$(I - A^{[m+1]})\boldsymbol{V}^{[m+1]} = (I + A^{[m]})\boldsymbol{V}^{[m]} + \boldsymbol{c}_{m,m+1}. \tag{11}$$

III. Crout factorization for the triangular system

Both the put and call modes are defined for $S \in (0, \infty)$ and in a practical convenience we truncate the infinity range to a finite one $(0, S_{\max})$ with $S_{\max}$ being a large quantity (see our comments in Section I). On the other hand, to get an accurate numerical solution the mesh size ΔS should be small. This implies that the number of the discrete points for S, i.e., $N = \frac{S_{\max}}{\Delta S}$, is of often large. Indeed, for a computation with $S_{\max} = \mathcal{O}(10^3)$ and $\Delta S = \mathcal{O}(10^{-3})$ we have to solve the linear system (11) at each time step of size $N = \mathcal{O}(10^6)$. For a regular desk computer, this is rather time expensive and is an unrealistic task for long time computation, even though the matrix $A^{[m+1]}$ is spare with triangular structure. So, it is necessary and valuable to look for a fast algorithm to handle the triangular system (11). To this end, we consider the Crout factorization in this section.

Theorem 1: Let $S_n = n\Delta S (n = 1, 2, \ldots, N-1)$ and $t_m = m\Delta t (m = 1, 2, \ldots, M)$ with $\Delta S > 0$ and $\Delta t > 0$. Assume that the problem parameters $\{\sigma_l,\ r_l\}_{l=1,2,\ldots,M}$ and the discretization parameter Δt satisfy

$$1 + \frac{\Delta t r_l}{2}\left(1 - \frac{r_l}{4\sigma_l^2}\right) \geq 0. \tag{12}$$

Then, the matrix $I - A^{[l]}$ is invertible and it holds

$$I - A^{[l]} = L^{[l]} U^{[l]}, \tag{13}$$

where $l \geq 0$ denotes the index of the discrete time point $t = t_l$ and

$$L^{[l]} = \begin{bmatrix} l_1 & & & & & \\ m_2 & l_2 & & & & \\ & m_3 & l_3 & & & \\ & & \ddots & \ddots & & \\ & & & m_{N-2} & l_{N-2} & \\ & & & & m_{N-1} & l_{N-1} \end{bmatrix},$$

$$U^{[l]} = \begin{bmatrix} 1 & u_1 & & & & \\ & 1 & u_2 & & & \\ & & 1 & u_3 & & \\ & & & \ddots & \ddots & \\ & & & & 1 & u_{N-2} \\ & & & & & 1 \end{bmatrix}.$$

The quantities $\{l_n\}_{n=1}^{N-1}$, $\{m_n\}_{n=2}^{N-1}$ and $\{u_n\}_{n=1}^{N-2}$ are determined recursively by

$$\begin{cases} m_n = -\gamma_n^{[l]},\ n = 2, 3, \ldots, N-1, \\ l_1 = 1 + 2\beta_1^{[l]},\ u_1 = \frac{-\alpha_1^{[l]}}{l_1}, \\ l_n = 1 + 2\beta_n^{[l]} - m_n u_{n-1},\ n = 2, 3, \ldots, N-1, \\ u_n = \frac{-\alpha_n^{[l]}}{l_n},\ n = 2, 3, \ldots, N-2. \end{cases}$$

Proof: According to [13, Theorem 2.1, pp. 51], it is sufficient to prove that one of the following two conditions holds:
1. the matrix $I - A^{[l]}$ is strict diagonally dominant;
2. the matrix $I - A^{[l]}$ is diagonally dominant and

$$\alpha_n^{[l]} \neq 0 (\forall n = 1, 2, \ldots, N-2),\ |\gamma_{N-1}^{[l]}| < 1 + 2\beta_{N-1}^{[l]}. \tag{14}$$

We prove that under the assumptions stated by the theorem the second condition holds. We have

$$I - A^{[l]} = \begin{bmatrix} 1 + 2\beta_1^{[l]} & -\alpha_1^{[l]} & & & & \\ -\gamma_2^{[l]} & 1 + 2\beta_2^{[l]} & -\alpha_2^{[l]} & & & \\ & \ddots & \ddots & \ddots & & \\ & & -\gamma_{N-2}^{[l]} & 1 + 2\beta_{N-2}^{[l]} & -\alpha_{N-2}^{[l]} & \\ & & & -\gamma_{N-1}^{[l]} & 1 + 2\beta_{N-1}^{[l]} \end{bmatrix}.$$

According to (9), it is clear that $\alpha_n^{[l]} > 0$ for $n = 1, 2, \ldots, N-2$ and $\beta_n^{[l]} > 0$ for $n = 1, 2, \ldots, N-1$. We next prove

$$1 + 2\beta_n^{[l]} \geq \begin{cases} \alpha_n^{[l]}, & n = 1, \\ |\gamma_n^{[l]}| + \alpha_n^{[l]}, & n = 2, 3, \ldots, N-2, \\ |\gamma_{N-1}^{[l]}|, & n = N-1. \end{cases}$$

For $n = 1$, by noticing $S_1 = \Delta S$ we have

$$\begin{aligned} &1 + 2\beta_1^{[l]} - \alpha_1^{[l]} \\ &= 1 + \frac{\Delta t\sigma_l^2 S_1^2}{2\Delta S^2} + \frac{r_l \Delta t}{2} - \frac{\Delta t\sigma_l^2 S_1^2}{4\Delta S^2} - \frac{r_l \Delta t S_1}{4\Delta S} \\ &= 1 + \frac{\Delta t\sigma_l^2 S_1^2}{4\Delta S^2} - \frac{r_l \Delta t S_1}{4\Delta S} + \frac{r_l \Delta t}{2} \\ &= 1 + \frac{\Delta t\sigma_l^2}{4} - \frac{r_l \Delta t}{4} + \frac{r_l \Delta t}{2} \\ &= 1 + \frac{\Delta t\sigma_l^2}{4} + \frac{r_l \Delta t}{4} \geq 0, \end{aligned}$$

where we have used $S_1 = \Delta S$.

For $n = 2, 3, \ldots, N-2$ we have

$$1 + 2\beta_n^{[l]} - (|\gamma_n^{[l]}| + \alpha_n^{[l]}) = 1 + \frac{\Delta t \sigma_l^2 S_n^2}{2\Delta S^2} + \frac{r_l \Delta t}{2} - \left(\frac{\Delta t \sigma_l^2 S_n^2}{4\Delta S^2} + \frac{r_l \Delta t S_n}{4\Delta S} + \left| \frac{\Delta t \sigma_l^2 S_n^2}{4\Delta S^2} - \frac{r_l \Delta t S_n}{4\Delta S} \right| \right).$$

If $\frac{\Delta t \sigma_l^2 S_n^2}{4\Delta S^2} \geq \frac{r_l \Delta t S_n}{4\Delta S}$, i.e., $n\sigma_l^2 \geq r_l$, it holds

$$\frac{\Delta t \sigma_l^2 S_n^2}{4\Delta S^2} + \frac{r_l \Delta t S_n}{4\Delta S} + \left| \frac{\Delta t \sigma_l^2 S_n^2}{4\Delta S^2} - \frac{r_l \Delta t S_n}{4\Delta S} \right| = \frac{\Delta t \sigma_l^2 S_n^2}{2\Delta S^2}.$$

This implies $1 + 2\beta_n^{[l]} - (|\gamma_n^{[l]}| + \alpha_n^{[l]}) = 1 + \frac{r_l \Delta t}{2} > 0$. If $\frac{\Delta t \sigma_l^2 S_n^2}{4\Delta S^2} < \frac{r_l \Delta t S_n}{4\Delta S}$ (i.e., $n\sigma_l^2 < r_l$), it holds

$$\frac{\Delta t \sigma_l^2 S_n^2}{4\Delta S^2} + \frac{r_l \Delta t S_n}{4\Delta S} + \left| \frac{\Delta t \sigma_l^2 S_n^2}{4\Delta S^2} - \frac{r_l \Delta t S_n}{4\Delta S} \right| = \frac{r_l \Delta t S_n}{2\Delta S},$$

and thus by using $S_n = n\Delta S$ we get

$$\begin{aligned} 1 + 2\beta_n^{[l]} - (|\gamma_n^{[l]}| + \alpha_n^{[l]}) &= 1 + \frac{\Delta t \sigma_l^2 S_n^2}{2\Delta S^2} + \frac{r_l \Delta t}{2} - \frac{r_l \Delta t S_n}{2\Delta S} \\ &= 1 + \frac{n^2 \Delta t \sigma_l^2}{2} + \frac{r_l \Delta t}{2} - \frac{n r_l \Delta t}{2} \\ &= 1 + \frac{\Delta t}{2} \left(n^2 \sigma_l^2 - n r_l + r_l \right). \end{aligned}$$

By regarding $1 + \frac{\Delta t}{2}\left(n^2\sigma_l^2 - nr_l + r_l\right)$ as a function of n for $n \geq 2$, we know that the minimum of this function is

$$\begin{cases} 1 + \frac{\Delta t}{2}\left(4\sigma_l^2 - r_l\right), & \text{if } \frac{r_l}{2\sigma_l^2} \leq 2, \\ 1 + \frac{\Delta t}{2}\left(r_l - \frac{r_l^2}{4\sigma_l^2}\right), & \text{if } \frac{r_l}{2\sigma_l^2} > 2. \end{cases}$$

For $\frac{r_l}{2\sigma_l^2} \leq 2$, i.e., $\frac{r_l}{4\sigma_l^2} \leq 1$ we have $1 + \frac{\Delta t}{2}\left(4\sigma_l^2 - r_l\right) = 1 + 4\sigma_l^2 \times \frac{\Delta t}{2}\left(1 - \frac{r_l}{4\sigma_l^2}\right) > 0$. For the other case $\frac{r_l}{2\sigma_l^2} > 2$, i.e., $\frac{r_l}{4\sigma_l^2} > 1$, we have $1 + \frac{\Delta t}{2}\left(r_l - \frac{r_l^2}{4\sigma_l^2}\right) = 1 + \frac{\Delta t r_l}{2}\left(1 - \frac{r_l}{4\sigma_l^2}\right)$ and under the assumption (12) we have $1 + \frac{\Delta t}{2}\left(r_l - \frac{r_l^2}{4\sigma_l^2}\right) \geq 0$ as well. The above analysis implies $1 + 2\beta_n^{[l]} - (|\gamma_n^{[l]}| + \alpha_n^{[l]}) \geq 0$ for all $n = 2, 3, \ldots, N-2$.

It remains to prove the last inequality $1 + 2\beta_{N-1}^{[l]} \geq |\gamma_{N-1}^{[l]}|$. Since our proof for $1 + 2\beta_n^{[l]} - (|\gamma_n^{[l]}| + \alpha_n^{[l]}) \geq 0$ holds for all $n \geq 2$, it is clear that

$$1 + 2\beta_{N-1}^{[l]} - |\gamma_{N-1}^{[l]}| > 1 + 2\beta_{N-1}^{[l]} - (|\gamma_{N-1}^{[l]}| + \alpha_{N-1}^{[l]}) \geq 0,$$

which ends the whole proof. ■

A routine calculation implies that the condition (12) is equivalent to

$$\sigma_l \geq \frac{r_l}{2}\sqrt{\frac{\Delta t}{2 + \Delta t r_l}}. \tag{15}$$

Hence, we can select a small Δt to let $I - A^{[l]}$ invertible and to let the Crout factorization (13) applicable. Of course, this is only a sufficient condition and in many cases we find that the coefficient matrix $I - A^{[l]}$ is invertible (and thus the Crout factorization (13) is applicable) even though (15) does not hold.

IV. Numerical Results

In this section, we consider some concrete examples and compute the numerical solution via the CN scheme and the Crout factorization (for handing the linear system at each step of the CN scheme). All numerical results are implemented by Matlab R2016b installed in a desk computer with Mac OS and 2.7 GHz Intel Core i5.

A. Put model

We first consider the put model (3a)-(3b) with the following data:

$$E = 2,\ S_{\max} = 10,\ r = 0.02 + 0.04t,\ \sigma = \frac{1 + e^t}{4}, \tag{16}$$

where r and σ are the same functions used in [11]. The profile of the solution $P(S,t)$ is illustrated in Figure 1 for three different values of the expiry time T. We see that there is bump near the left boundary $S = 0$ in the time interval $[0, 10]$ and as time grows the bump disappears and the solution rapidly and uniformly decays to 0.

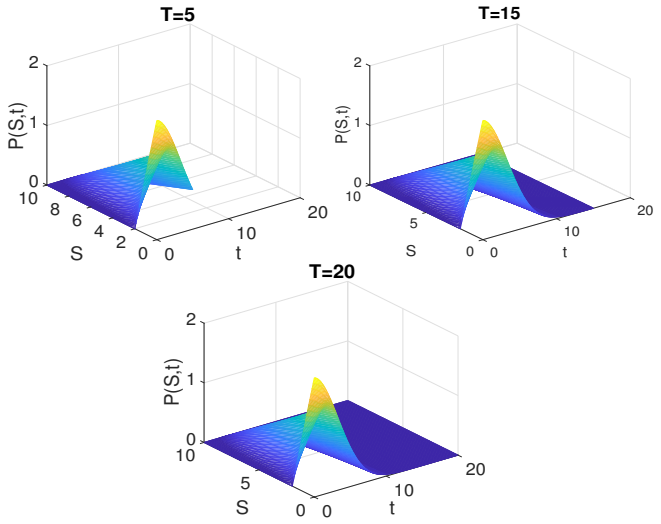

Fig. 1. Solution $P(S,t)$ of the put model with data (16) and different expiry time T: $T = 5$, $T = 15$ and $T = 20$.

As we mentioned at the begin of Section III, each step of the CN scheme lies in solving a large scale tridiagonal system, which is the major computation cost. We now solve this linear system for each CN step by two solvers: the built-in command 'backslash' (i.e., the `inv` command) in Matlab and the Crout factorization introduced in Section III. In Figure 2, we plot the maximal error at each discrete time point between the numerical solutions obtained by these two solvers, where we see clearly that the error is very small. This implies that both solvers lead to the same numerical solutions if we neglect the roundoff error.

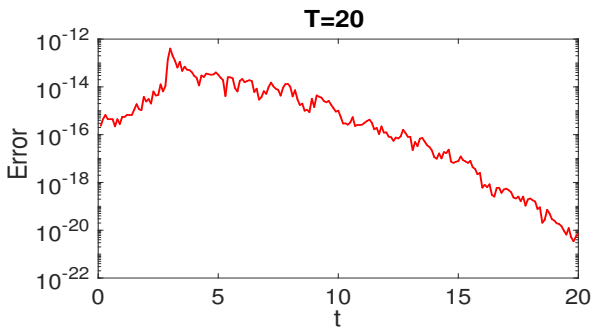

Fig. 2. For the put model with data (16), the maximal error at each discrete time point between the numerical solutions obtained by the Matlab's built-in command 'backslash' (i.e., the `inv` command) and the Crout factorization introduced in Section III.

In Table I, we compare the CPU time of solving the put model by using these two solvers. The CPU time is measured

by the `tic` and `toc` commands in Matlab. Clearly, the Crout factorization needs approximately a half less CPU time compared to the backslash command.

TABLE I
PUT MODEL: COMPARISON OF CPU TIME (IN SECONDS) FOR TWO LINEAR SOLVERS

(M, N)	(200, 100)	(400, 200)	(800, 400)	(2000,1000)
backslash (`inv`)	0.4395	0.9888	3.1621	56.5350
LU-factorization	0.3720	0.4949	1.8848	31.6852

B. Call model

We next consider the call model (4a)-(4b) with data

$$E = 2,\ S_{\max} = 10,\ r = \frac{t}{1+t},\ \sigma = 1 + \ln(1+t). \quad (17)$$

We plot in Figure 3 the solution $C(S,t)$ for different expiry time T. Different from the put model, we see that the solution of the call model evolves smoothly as t increases and does not decay to zero uniformly.

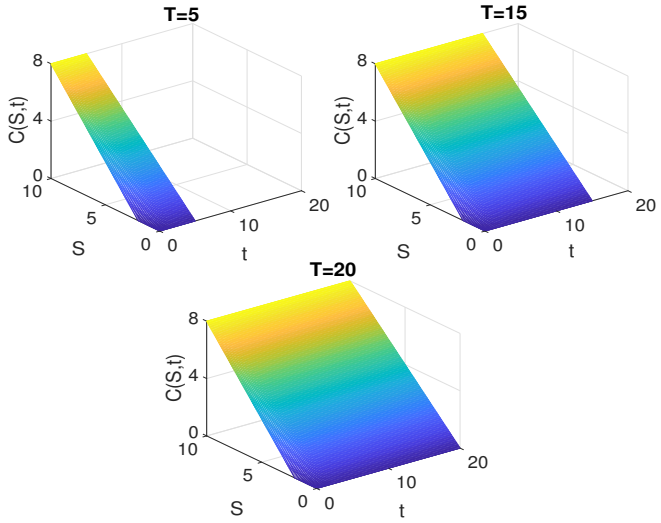

Fig. 3. Solution $C(S,t)$ of the call model with data (17) and different expiry time T: $T = 5$, $T = 15$ and $T = 20$.

We now compare the two linear solvers for each CN step, i.e., the built-in command 'backslash' (i.e., the `inv` command) in Matlab and the Crout factorization introduced in Section III. Similar to Figure 2 we plot in Figure 4 the maximal error between the numerical solutions computed by these two solvers. Again, both solvers lead to the same numerical solutions if we neglect the roundoff error.

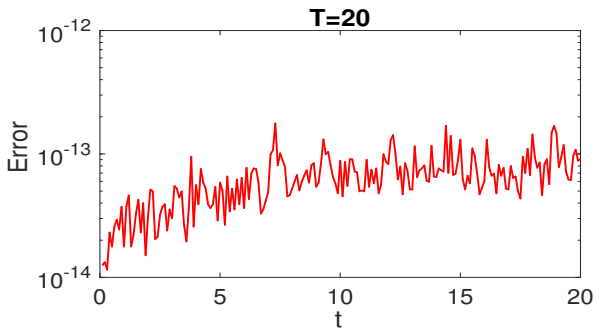

Fig. 4. For the call model with data (17), the maximal error at each discrete time point between the numerical solutions obtained by the Matlab's built-in command 'backslash' (i.e., the `inv` command) and the Crout factorization introduced in Section III.

At last, in Table II we compare the CPU time of solving the call model by using these two solvers. Similar to the put model, it is clear that the Crout factorization needs approximately a half less CPU time compared to the backslash command.

TABLE II
CALL MODEL: COMPARISON OF CPU TIME (IN SECONDS) FOR TWO LINEAR SOLVERS

(M, N)	(200, 100)	(400, 200)	(800, 400)	(2000,1000)
backslash (`inv`)	0.3171	0.8395	3.7288	55.4824
LU-factorization	0.2599	0.5003	1.8370	34.2013

V. CONCLUSION

The Black-Scholes equation is a class of fundamental mathematic models in finance. These are partial differential equations and numerical solutions play an important role in the study of these equations, especially when the problem parameters are time-dependent (because there are no exact solutions in this case). In this paper, we consider the Crank-Nicolson (CN) scheme as discretization, which is of second-order accuracy. In each CN step, we have to handle a large scale linear system (with coefficient matrix changing from step to step) and solving this system is the major computation burden. By looking insight into the relationship between the coefficients of the CN scheme, we propose a Crout factorization of the matrix, which leads to fast computation of the linear system via forward and backward substitutions. Numerical results indicate that the Crout factorization needs approximately a half less CPU time compared to the built-in command `inv` in Matlab. Such a Crout factorization can be used to handle linear systems arising from other problems, such as the adaptive fuzzy funnel control [22], pollutant spread in water [1] and parameter estimation [21].

REFERENCES

[1] N. Y. Ashar, and I. Solekhudin, "A numerical study of steady pollutant spread in water from a point source," *Engineering Letters*, Vol 29, pp. 840–848, 2021.

[2] M. I. Azis, "A numerical simulation for transient diffusion convection reaction problems of anisotropic quadratically graded media with incompressible flow," *Lecture Notes in Engineering and Computer Science: Proceedings of the World Congress on Engineering*, 7-9 July, 2021, London, U.K., pp. 1–6, 2021.

[3] F. Black and M. Scholes, "The pricing of options and corporate liabilities," *Journal of Political Economy*, Vol. 81, pp. 637–659, 1973.

[4] R. Company, E. Navarro, J. R. Pintos, and E. Ponsoda, "Numerical solution of linear and nonlinear black–scholes option pricing equations," *Comput. Math. Appl.*, Vol. 56, pp. 813–821, 2008.

[5] R. Company, A. L. Gonzalez, and L. Jodar, "Numerical solution of modified Black-Scholes equation pricing stock options with discrete dividend," *Mathematical and Computer Modelling*, Vol. 44, pp. 1058–1068, 2006.

[6] J. Dewynne and W. T. Shaw, "Differential equations and asymptotic solutions for arithmetic Asian options: 'Black-Scholes formulae' for Asian rate calls," *European Journal of Applied Mathematics*, Vol. 19, pp. 353–391, 2008.

[7] S. Doley, A. V. Kumar, K. R. Singh, and L. Jino, "Study of time fractional Burgers' equation using Caputo, Caputo-Fabrizio and Atangana-Baleanu fractional derivatives," *Engineering Letters*, Vol. 30, pp. 1017–1024, 2022.

[8] S. O. Edeki, O. O. Ugbebor, G. O. Akinlabi, and O. Gonzalez-Gaxiola, "Approximate solutions of a variable volatility driven black-scholes option pricing model". In: *2nd International Conference on Knowledge Engineering and Applications (ICKEA)*, 2017.

[9] R. Farnoosh, A. Sobhani, H. Rezazadeh, and M. H. Beheshti, "Numerical method for discrete double barrier option pricing with time-dependent parameters," *Computers and Mathematics with Applications*, Vol. 70, pp. 2006–2013, 2015.

[10] P. Forsyth, K. Vetzal, and R. Zvan, "A finite element approach to the pricing of discrete lookbacks with stochastic volatility," *Applied Mathematical Finance*, Vol. 6, pp. 87–106, 1999.
[11] S. G. Georgiev and L. G. Vulkov, "Fast reconstruction of time-dependent market volatility for European options," *Computers and Mathematics with Applications*, 40: 30, 2021.
[12] G. R. Goldstein, J. A. Goldstein, and M. Kaplin, "The chaotic Black–Scholes equation with time-dependent coefficients," *Archiv der Mathematik*, Vol. 115, pp. 1–12, 2020.
[13] M. H. Holmes, *Introduction to mumerical methods in differential equations*. Springer New York, 2009.
[14] Y. Jin, J. Wang, S. Kim, Y. Heo, C. Yoo, Y. Kim, J. Kim, and D. Jeong, "Reconstruction of the time-dependent volatility function using the Black–Scholes model," *Discrete Dynamics in Nature and Society*, 3093708, 2018.
[15] L. Jodar, P. Sevilla-Peris, J. C. Cortes, and R. Sala, "A new direct method for solving the Black-Scholes equation," *Applied Mathematics Letters*, Vol. 18, pp. 29–32, 2005.
[16] R. Kangro and R. Nicolaides, "Far field boundary conditions for black–scholes equations," *SIAM Journal on Numerical Analysis*, Vol. 38, pp. 1357–1368, 2001.
[17] L. Li, Z. Wei, and Q. D. Huang, "A numerical method for solving fractional variational problems by the operational matrix based on Chelyshkov polynomials," *Engineering Letters*, Vol 28, pp. 486–491, 2020.
[18] R. C. Merton, "Theory of rational option pricing," *The Bell Journal of Economics and Management Science*, Vol. 4, pp. 141–183, 1973.
[19] O. Pironneau and F. Hecht, "Mesh adaption for the black and scholes equations," *East West Journal of Numerical Mathematics*, Vol. 8, pp. 25–35, 2000.
[20] M. R. Rodrigo and R. S. Mamon, "An alternative approach to solving the BlackScholes equation with time-varying parameters," *Applied Mathematics Letters*, Vol. 19, pp. 398–402, 2006.
[21] S. Ryota, I. Jun, O. Yukihiko, and Y. Kyosuke, "Numerical study of the effect of measurement noise on the accuracy of bridge parameter estimation in VBI system identification," *Lecture Notes in Engineering and Computer Science: Proceedings of the World Congress on Engineering*, 7-9 July, 2021, London, U.K., pp. 10–15, 2021.
[22] F. R. Shi, N. N. Zhao, X. Y. Ouyang, H. B. Xu, and Y. Q. Zhou, "Adaptive fuzzy funnel control for pure-feedback nonlinear system with input constraint," *IAENG International Journal of Computer Science*, Vol 48, pp. 334–342, 2021.
[23] P. Wilmott, J. Dewynne, and S. Howison, *Option pricing: mathematical models and computation*. Oxford Financial Press, Oxford, UK, 1994.
[24] P. Wilmott, S. Howison, and J. Dewynne, *The mathematics of financial derivatives. a student introduction*. Cambridge University Press, Cambridge, 1995.
[25] J. M. Zuo, "New compact finite difference schemes with fourth-order accuracy for the extended Fisher-Kolmogorov equation," *Engineering Letters*, Vol. 30, pp. 131–139, 2022.

Haiyan Song The first author has obtained Doctor of Science degree in Mathematics and Statistics from Wuhan University, in 2020. She is a lecturer of Computer and Data Engineering, NingboTech University since September 2020.

Haoxi Jia The second author was born in Yuci District, Jinzhong City, Shanxi Province, China, in August 11, 1993. In June 2015, he graduated with Bachelor's Degree of Management at Changchun Normal University, Changchun City, Jilin Province, China. During this period, he obtained the Accounting Qualification Certificate. From November 2015 to December 2019, he completed Master of Professional Accounting at Monash University and Holmes Institute, Melbourne, Australia. From October 2020 to September 2021, he completed MSc Finance & Accounting at the University of Sheffield, Sheffield, UK.