End-user's instruction

April 2022

1 Introduction

This is an end-user guidance document. It is intended to guide end-users to the correct usage of this implementation, select functions, configure parameters, and interpret the displayed results. Another intention of this document is to give a brief explanation about the Barrier (up-and-out) call option price of a local volatility model and methods used to calculate the price, including the selection of relevant parameters.

What contain in the zip file

When you unzip the file you will find a number of files which are five MATLAB scripts, two documents, the developer documentation and end-user's instruction (this document). The MATLAB scripts (.m files) consist of the following files: *explicit.m*, *implicit.m*, *crank.m*, *monte_carlo.m* and *main.m*

The first four scripts are the implementations (finite difference methods and Monte Carlo simulation) and the last file (main.m) is the main script that calls the implemented functions and displays the computed result and the corresponding plot.

2 A brief description of the implemented methods.

2.1 the local volatility Black-Scholes model

Given the interest rate r, the dividend yield d and a local volatility surface $\sigma(S,t) = 0.25e^{-t}(100/S_t)^{\alpha}$ when α is a local volatility parameter. Assume that the stock price S follows the process

$$dS_t = (r-d)S_t dt + \sigma(S_t,t)S_t dW_t,$$

where $W = (W(t) : t \geq 0)$ is a standard Brownian motion.

Let $V(S,t)$ be the option price on the above stock satisfies the Black-Scholes (BS) equation:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma(S,t)^2 S^2 + (r-d)S\frac{\partial V}{\partial S} = rV,$$

where $-\infty \leq S \leq \infty$ and $0 \leq t \leq T$. And with final condition $V(S,T) = f(S)$.

One can use the change of variables $\tau = T - t$ to simplify BS equation, we have

$$\frac{\partial V}{\partial \tau} = \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\sigma(S,\tau)^2 S^2 + (r-d)S\frac{\partial V}{\partial S} - rV$$

and $V(S,0) = f(S)$.

2.2 the Barrier options

In our implementation, we use an up-and-out call option with zero rebate which has a payoff;

$$f(S_T) = \begin{cases} \max(0, S_T - K) & \text{if } \max\{S_t : 0 \leq t \leq T\} < B, \\ 0 & \text{otherwise,} \end{cases}$$

where K is a strike(or exercise) price and B is a barrier level. Hence the initial condition at $\tau = 0 (t = T)$ is

$$V(S,0) = \max\{S-K, 0\},$$

when $S < B$ and $V(S,0) = 0$ otherwise.

The boundary conditions for an up-and-out call option are, for all $\tau \in [0, T]$,

$$V(S = 0, \tau) = 0,$$

due to it is call option, the payoff is 0. And because in a barrier call option the payoff is 0 when the asset price exceeds the barrier level,

$$V(S = B, \tau) = 0.$$

That is, it has zero boundary conditions.

Moreover, to solve the above problem via the Finite Difference Methods, we will consider the domain

$$D = \{0 \leq S \leq B; 0 \leq \tau \leq T\}$$

which can be discretized as:

$$D = \{S_j = (j-1)\Delta S; \tau_k = (k-1)\Delta\tau\}$$

$\forall j = 1, \dots, N+1, \forall k = 1, \dots, M+1$, when $\Delta S = \frac{B}{N}$ and $\Delta\tau = \frac{T}{M}$.

2.3 the Finite Difference Methods (FDM)

To use the concept of FDM, we first define the constants:

$$\alpha_{j,k} = \frac{\sigma_{j,k}^2 S_j^2}{2}\frac{\Delta\tau}{\Delta S^2}, \qquad \beta_{j,k} = \frac{(r-d)S_j}{2}\frac{\Delta\tau}{\Delta S}$$

We also define

$$l_{j,k} = \alpha_{j,k} - \beta_{j,k}, \qquad d_{j,k} = 1 - r\Delta\tau - 2\alpha_{j,k}, \qquad u_{j,k} = \alpha_{j,k} + \beta_{j,k},$$

$$\hat{l}_{j,k} = \beta_{j,k} - \alpha_{j,k}, \qquad \hat{d}_{j,k} = 1 + r\Delta\tau + 2\alpha_{j,k}, \qquad \hat{u}_{j,k} = -(\alpha_{j,k} + \beta_{j,k}),$$

$$A_k^E = \begin{bmatrix} d_{2,k} & u_{2,k} & & & & \\ l_{3,k} & d_{3,k} & u_{3,k} & & & \\ & \cdots & \cdots & \cdots & & \\ & & l_{N-1,k} & d_{N-1,k} & u_{N-1,k} \\ & & & l_{N,k} & d_{N,k} \end{bmatrix}, \ A_k^I = \begin{bmatrix} \hat{d}_{2,k+1} & \hat{u}_{2,k+1} & & & \\ \hat{l}_{3,k+1} & \hat{d}_{3,k+1} & \hat{u}_{3,k+1} & & \\ & \cdots & \cdots & \cdots & \\ & & \hat{l}_{N-1,k+1} & \hat{d}_{N-1,k+1} & \hat{u}_{N-1,k+1} \\ & & & \hat{l}_{N,k+1} & \hat{d}_{N,k+1} \end{bmatrix}$$

and $V_k = \begin{bmatrix} V_{2,k} & V_{3,k} & \dots & V_{N-1,k} & V_{N,k} \end{bmatrix}^T$.

2.3.1 explicit scheme

We can perform the recursive explicit discretization scheme to obtain the approximation solution as

$$V_{k+1} = A_k^E V_k.$$

Note: this scheme will be stable if all the eigenvalues of A_k^E have absolute values less than 1.

2.3.2 implicit scheme

We can perform the recursive implicit discretization scheme to obtain the approximation solution as

$$V_{k+1} = (A_k^I)^{-1} V_k.$$

Note: this scheme will be stable if all the eigenvalues of $(A_k^I)^{-1}$ have absolute values less than 1.

2.3.3 Crank-Nicolson scheme

We can perform the recursive Crank-Nicolson discretization scheme to obtain the approximation solution as

$$V_{k+1} = (A_k^I + I)^{-1}(A_k^E + I)V_k.$$

Note: this scheme will be stable if all the eigenvalues of $(A_k^I + I)^{-1}(A_k^E + I)$ have absolute values less than 1.

2.4 the Monte Carlo simulation

Consider the price of a call option with strike price K and maturity T

$$C = \mathbb{E}[e^{-rT}\max\{S(T) - K, 0\}],$$

when the stock price is a geometric Brownian motion

$$S(t) = S(0)\exp\{(r - d - \frac{\sigma^2}{2})t + \sigma W(t)\}$$

where W is a standard Brownian motion.

In order to perform the Monte Carlo simulation for the option payoff, we will repeat the follows process N times:

1. generate Z_i from N(0, 1),
2. compute $S_i = S(0)\exp\{(r - d - \frac{\sigma^2}{2})T + \sigma\sqrt{T}Z_i\}$,
3. set $X_i = e^{-rT}f(S_i)$ where f is an up-and-out payoff function.

Then we can compute the estimate $\hat{C} = \frac{1}{N}\sum_{i=1}^{N} X_i$.

2.4.1 antithetic sampling

Antithetic Sampling is a variance reduction technique which introducing samples that are negatively correlated. That is, repeat the above process and using the same Z_i to compute $S_i^a = S(0)\exp\{(r - d - \frac{\sigma^2}{2})T - \sigma\sqrt{T}Z_i\}$ and set $Y_i = e^{-rT}f(S_i^a)$ then we can calculate the estimate $\hat{C} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{X_i+Y_i}{2}\right)$.

3 Example

3.1 How to input the data

The following is an example of using my implementation for calculating barrier option prices with Finite Difference Methods (FDM) and Monte Carlo simulation. To use my developed functions for finding the call option price, one need to set the input for the function which are stock price (S0), strike price (K), barrier level (B), time to maturity (T), interest rate (r), dividend yield (q) and the local volatility surface parameter (α). Including the number of grid points for the space (N) and time intervals (M) when using Finite Difference Methods or the number of simulation (N_sim) for using Monte Carlo simulation.

```
%% the problem parameters
S0 = 100;              % spot price (in British Pound)
K = 90;                % strike price (in British Pound)
B = 130;               % barrier level (in British Pound)
r = 3;                 % risk-free rate (in %)
q = 5;                 % dividend yield (in %)
T = 0.5;               % time to maturity (years)
vola_alpha = 0.35;     % the local volatility alpha

%% the model parameters
% FDM: Set the number of grid points
N = 50;            % For the space interval [a,b]
M = 500;           % For the time interval [0,T]

% Monte Carlo: Set the number of simulations
N_sim = 10000;  % Number of simulations
```

3.2 How to use the implemented functions

In the follows example, it takes values for each parameters as its shown in the code snippet, one will obtains the call option price corresponding to the given stock price (S0). In the case of FDM, the implemented function will also return the option prices with the whole FDM grid in $S \in [0, B]$ and $t \in [0, T]$. On the other hand, Monte Carlo simulation will returns only the call price and its standard error.

```
%% solving the Black-Scholes PDE using explicit FDM
[call , V] = explicit(S0,K,B,T,r,q,vola_alpha,N,M);

%% solving the Black-Scholes PDE using implicit FDM
[call , V] = implicit(S0,K,B,T,r,q,vola_alpha,N,M);

%% solving the Black-Scholes PDE using Crank-Nicolson FDM
[call , V] = crank(S0,K,B,T,r,q,vola_alpha,N,M);

%% Monte Carlo simulation
[call , se_call] = monte_carlo(S0,K,B,T,r,q,vola_alpha,N_sim);
```

3.3 How to display the results

```
% plot of FDM solution
subplot(2,1,1)
dS = B/N;
S = (0+dS:dS:B-dS)';
plot(S,V(:,M+1),'LineWidth',2)
hold on
plot(S0,call,'r*') % mark a call option price on the plot
title('European Call price, Crank-Nicolson - BS formula')
xlabel('Stock price')
ylabel('Call price')

subplot(2,1,2)
t = T-(0:T/M:T);
[X,Y] = meshgrid(t,S);
surf(X,Y,V,'LineStyle','none');
xlabel('t')
ylabel('S(t)')
zlabel('V(S,t)')
```

To illustrate the result from FDMs (e.g. Crank-Nicolson scheme), one may use the above snippet for making graphs. The first plot is the option price at time 0 plot against the stock spot price which the red mark is the call price at S0. The second graph is the option prices plot with the whole FDM grid in $S(t)$ and t.

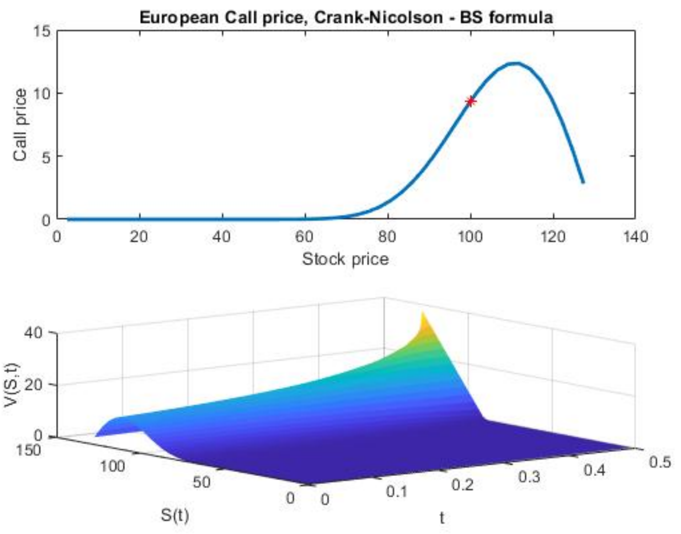

Figure 1: (Top) the option at time 0 plot against the stock spot price. (Bottom) the option prices plot with the whole FDM grid in $S(t)$ and t.