MTH 600- Final Project

Name: Sophia Rybnik

Date: April 2024

Project Background

A typical practice in finance is to estimate an option pricing model from market prices of options that are traded frequently (in other words, liquid options). The calibrated model can be used to hedge liquid options or price and hedge other (illiquid) exotic options on the same underlying. In this project, we will consider the calibration of a local volatility function (LVF) model in practice.

The LVF model assumes the stock price as

$$\mathrm{dS} = r \cdot \mathrm{Sdt} + \sigma(S,t)\mathrm{SdW}_t,$$

where $\sigma(S,t) = \max\left(0.0, x_1 + x_2 S + x_3 S^2\right)$. The European call option value is then the solution of the PDE

$$V_\tau - \frac{\sigma^2(S,\tau)}{2} S^2 V_{\mathrm{SS}} - \mathrm{rSV}_S + \mathrm{rV} = 0, \text{ (1)}$$

where $\tau = T - t$ is the time to maturity.

I) Determining the Initial and Boundary Conditions

We will first introduce some simple notation:

V = the option value; a function of current value of the underlying asset and time, i.e. $V(S,t)$.

The value of the option also depends on:

K = the exercise price;

T = the expiry time;

$\sigma^2(S,\tau)$ = the local volatility of the underlying asset;

r = the risk-free interest rate

For a European call option at maturity T, the payoff is determined by the choice to exercise or not exercise the contract. In terms of T, the terminal condition is given by $V(S,T) = (S-K)^+$, where S is the price of the underlying at T. Considering the transformation $\tau = T - t$, the initial condition is then $V(S,0) = (S-K)^+$.

To solve the Black Scholes Local Volatilty Model numerically, the domain must be discretized, i.e. $[0, S_{\max}] \times [0, T]$. Although the price of the underlying asset has an unbounded upside, i.e. $[0, \infty)$, an artificial limit $S_{\max}$ is introduced for the purposes of discretization. If the price of the underlying is zero at maturity, the option contract will not be exercised. On the other hand, as the price tends to infinity (i.e. $S_{\max}$ in discretized space), the contract will be exercised and the payoff is discounted to today at the risk free rate.

Thus, in discretized space, the boundary conditions are:

$$\begin{cases} V(0,t) = 0 \\ V(S_{\max},t) = (S_{\max} - K) \times e^{-rt} \end{cases}$$

II) Computational Pricing

i. Monte Carlo Method

Monte Carlo simulation can be used to price options by repeatedly taking random samples from an underlying distribution. Monte Carlo relies on risk-neutral valuation (see https://en.wikipedia.org/wiki/Risk-neutral_measure). Under the risk-neutral measure $\mathbb{Q}$, the fair price of an option is the expectation of the payoff discounted at the risk-free rate.

The Monte Carlo technique can be applied to value an option, dependent on a single underlying S, as follows:

1. Generate a large number of random price paths of the underlying, where S is assumed to follow a Geometric Brownian motion.
2. For each path, calculate the exercise value/payoff of the option.
3. Take the mean of the sample payoffs to get an estimate of the expected payoff in a risk-neutral world.
4. Discount the expected payoff to today using the risk-free rate.

The result is an estimate of the value of the option today. In pricing the option via the Monte Carlo method, anthithetic variates can be used as a variance reduction technique. In essence, the technique reduces the variance of the sample mean by introducing negative correlation between pairs of observations. This is done by taking the antithetic path of each sample and caculating the average expectation of the antithetic pair to reduce the variance of the sample mean (see https://en.wikipedia.org/wiki/Antithetic_variates or https://www.math.arizona.edu/~tgk/mc/book_chap5.pdf).

```
clear;
clc;

% define parameters
S0 = 1; % initial spot price
K = 1; % strice price
T = 0.25; % time to maturity
r = 0.03; % risk-free rate
x1 = [0.2, 0.001, 0.003]; % parameters for local volatility model
M_mc = 10000; % number of monte carlo simulations
N = 100; % number of time steps

% monte carlo method
V0_mc = Eur_Call_LVF_MC(S0, K, T, r, x1, M_mc, N)
```

```
V0_mc = 0.0455
```

ii. Finite Difference: Explicit Method

To numerically solve a continuous differential equation, such as the Black-Scholes PDE, finite differences can be used to approximate the continuous time (and price) derivatives. Many published papers further

explain how we can arrive at the discretized Black-Schole's PDE (see https://www.researchgate.net/publication/243112801_Numerical_Approximation_of_Black-Scholes_Equation).

As shown in part I, for numerical purposes the option price is modelled by a lattice in discrete dimensions: time runs from 0 to maturity and price runs from 0 to a high value, given by $S_{\max}$. Recall that the initial and boundary conditions are given by:

$$\begin{cases} V(S,0) = (S-K)^+ \\ V(0,t) = 0 \\ V(S_{\max},t) = (S_{\max}-K) \times e^{-\mathrm{rt}} \end{cases}$$

The option price can be solved for in different ways by modifying the approximation of the time derivative. We can solve the price explicitly if using the forward difference approximation, implicitly using the backward difference or semi implicitly by equal weighting the implicit and explicit methods (i.e. the Crank–Nicolson method). These techniques work such that the value at each lattice point is a function of the value at later and adjacent lattice points. After the PDE is discretized per the technique chosen, the values at the remaining lattice points are calculated by iterating backwards in time from maturity ($\rightarrow \tau = 0$) and inwards from the boundary prices ($\rightarrow S = 0$).

More practically speaking, an option can be priced today using the explicit finite difference method as follows:

1. Use a forward difference approximation for the time derivative V_τ and a symmetric central difference approximation for the second order price derivative V_{SS}. The first order price derivative V_S is approximated by either central or forward difference approximation to ensure positive coefficient discretization (to promote stability).
2. After plugging the finite difference approximations into (1), the resulting equation can be represented as a matrix of coefficients A.
3. The matrix of option prices V is initialized accounting for the implied initial and boundary conditions. Then, working backwards in time from $\tau = T$ to $\tau = 0$, we can solve the system $AV_i^{n+1} = V_i^n$. Given that A is a tri-diagonal matrix, the LU decomposition algorithm provides a fast and efficient method to solve for V_i^{n+1}.
4. Then, the value of the option today given the spot price of the underlying can be interpolated from the lattice.

```
M_fd = 30; % number of stock price steps
Smax=3; % upper bound of the stock price

% finite difference method
V0_fd = Eur_Call_LVF_FD(S0, K, T, r, x1, Smax, M_fd, N)
```

```
V0_fd = 0.0489
```

III) Model Calibration

i. Purpose

In the Black-Scholes model (1), the unknown variable is $\sigma^2(S,\tau)$. The remaining parameters are deterministic and easily observed. In part II, the option prices are computed using some given values of $x=[x_1,x_2,x_3]$ as input for the local volatility function. We want our model to obtain the best fit between the generated prices and some observed market prices. In order to do so, we need to fit the volatility function to market data to determine the optimal parameters for x. By optimizing the parameters of the volatility function, we can get our model to generate prices which are more closely consistent to the prices of liquidly traded options.

ii. Method

Assume that the initial market prices $V_0^{\text{mkt}}(K_j,T_j)$, $j=1,2,\ldots,m$, are available; here (K_j,T_j) denotes the strike and expiry of the j^{th} option.

Assume that $V_0(K_j,T_j;x)$ denotes the initial value of an option with strike K_j and T_j under a LVF model described by a set of parameters $x=[x_1,x_2,\ldots,x_n]$. In the quadratic local volatility model (1), the model parameters are $[x_1,x_2,x_3]$.

The model which best fits the market option prices can be estimated by solving the following non linear least squares problem

$$\min_{x\in\Re^n}\frac{1}{2}\sum_{j=1}^{m}\left(V_0(K_j,T_j;x)-V_0^{\text{mkt}}(K_j,T_j)\right)^2, \quad (2)$$

Let $F(x)$ denote the vector of model option value errors from the market prices:

$$F(x)=\begin{bmatrix} V_0(K_1,T_1;x)-V_0^{\text{mkt}}(K_1,T_1)\\ \vdots \\ V_m(K_m,T_m;x)-V_0^{\text{mkt}}(K_m,T_m)\end{bmatrix}$$

Vector $F(x):\ \Re^n\to\Re^m$ is a nonlinear function of the model parameter $x\in\Re^n$ and the calibration problem (2) is equivalently formulated as

$$\min_{x\in\Re^n}\left(f(x)\overset{\text{def}}{=}\frac{1}{2}\|F(x)\|_2^2\right), \quad (3)$$

A nonlinear least squares problem (3) can be solved by a special optimization method such as the Levenberg-Marquardt method, the Gauss Newton method, or a general method for nonlinear unconstrained optimization problems.

Optimization methods for nonlinear least squares problems typically exploit the special structure of the gradient and Hessian of the function $\|F(x)\|_2^2$. They typically require computation of the Jacobian matrix of $F(x)$

$$J(x) = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \cdots & \frac{\partial F_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial x_1} & \frac{\partial F_m}{\partial x_2} & \cdots & \frac{\partial F_m}{\partial x_n} \end{bmatrix}$$

One difficulty in solving the nonlinear least squares calibration problem (2) is that it is not easy to compute the Jacobian matrix $J(x)$ explicitly. Instead, a finite difference approximation of the Jacobian matrix can be computed.

iii. Results

The calibration problem will be solved using the Matlab function lsqnonlin with LevenbergMarquardt as the choice of the optimization method. The Monte Carlo method implemented above is used to evaluate $V_0(K, T, x)$.

The market prices used for model calibration are:

K	0.80	0.85	0.90	0.95	1.00	1.05	1.10
V_0	0.3570	0.2792	0.2146	0.1747	0.1425	0.1206	0.0676

The options have expiry in three months and the risk-free rate remains at 3%.

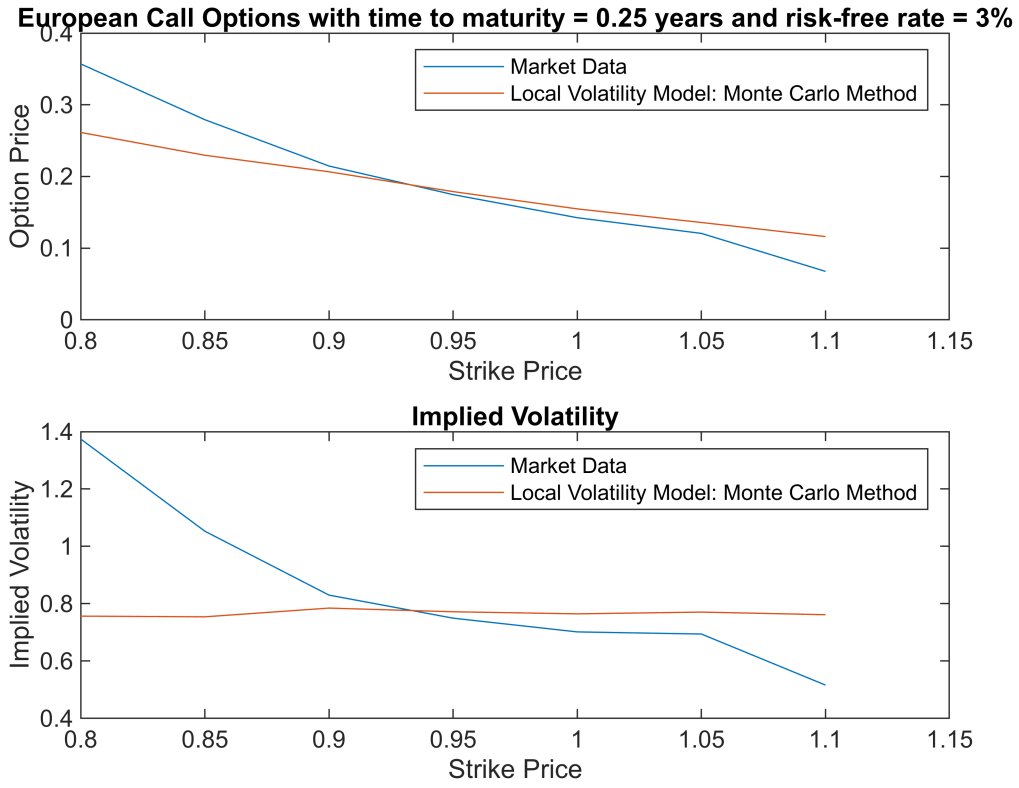
```
x0_1 = [0.2; 0.0; 0.0]; % initial parameter guess
[x1, resnorm1] = calibrate_parameters(x0_1) % Invoke optimizer
```

```
                                        First-order                    Norm of
 Iteration  Func-count     Resnorm      optimality       Lambda          step
     0          1        0.0793308            0.22         0.01
     1          2        0.0425347           0.174        0.001       0.215092
     2          3        0.0246344           0.155       0.0001        0.19827
     3          4        0.0182943          0.0338        1e-05      0.0892564
     4          5        0.0150406          0.0188        1e-06       0.081454
     5         12        0.0136408          0.0146            1      0.0218929
     6         13        0.0132235          0.0204          0.1      0.0244999
     7         20                                                  3.7116e-07

Local minimum possible.
lsqnonlin stopped because the relative size of the current step is less than
the value of the step size tolerance.

<stopping criteria details>
x1 = 3×1
    0.6597
    0.1069
         0
resnorm1 = 0.0132
```

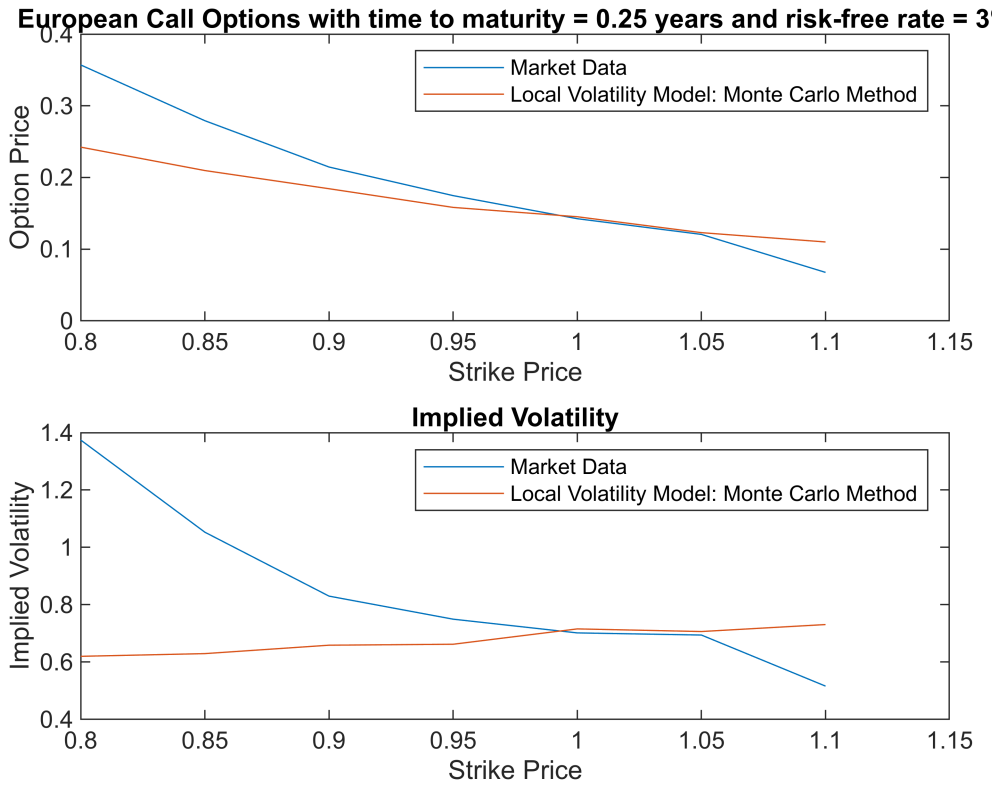
```
x0_2 = [0.2; 0.1; 0.01]; % initial parameter guess
[x2, resnorm2] = calibrate_parameters(x0_2) % Invoke optimizer
```

```
                                         First-order                    Norm of
 Iteration  Func-count     Resnorm          optimality       Lambda          step
     0          1        0.0608349               0.264         0.01
     1          2        0.0190914              0.0666        0.001      0.281599
     2          4         0.019086              0.0627         0.01      0.112277
     3          7        0.0181408              0.0255            1     0.0474496
     4         11        0.0179806              0.0333         1000    3.63104e-05
     5         13        0.0177727              0.0477        10000    5.06054e-06
     6         16                                                     4.81396e-08

Local minimum possible.
lsqnonlin stopped because the relative size of the current step is less than
the value of the step size tolerance.

<stopping criteria details>
x2 = 3×1
    0.3748
    0.2140
    0.1323
resnorm2 = 0.0178
```

```
impliedvolplots(x1) % graph
```




```
impliedvolplots(x2) % graph
```



From only looking at the value of the sum of the squared residuals from the calibration output, we cannot determine the extent to which the matlab program is successful in solving the optimization problem. We would have to further investigate the mean and distribution of the residuals, if they are uncorrelated with the independent and dependent variables, etc. However, from looking at the plots of the option prices and implied volatilities generated by our model, we can see that they significantly deviate from observed market values. The source of error may come from both the model itself and the calibration of its parameters.