Bull. Korean Math. Soc. **59** (2022), No. 2, pp. 507–528
https://doi.org/10.4134/BKMS.b210391
pISSN: 1015-8634 / eISSN: 2234-3016

# TIME STEPWISE LOCAL VOLATILITY

Hyeong-Ohk Bae and Hyuncheul Lim

Abstract. We propose a path integral method to construct a time stepwise local volatility for the stock index market under Dupire's model. Our method is focused on the pricing with the Monte Carlo Method (MCM). We solve the problem of randomness of MCM by applying numerical integration. We reconstruct this task as a matrix equation. Our method provides the analytic Jacobian and Hessian required by the nonlinear optimization solver, resulting in stable and fast calculations.

## 1. Introduction

We propose a new practical method for the series of the local volatility (LV) curves under Dupire's model. Our method is specialized in the practical usage. It is designed for MCM with non equidistant time intervals and complex contingency claims.

**Problems arising from the calibration of the local volatility.** The following typical problems appear in LV generation:

1. Inverse Problem of LV: Calibration of LV is known as a mathematical inverse problem. This is because LV provided as a PDE coefficient has been set to very small grid sizes $dT$ and $dK$, but in practice, the market call option prices are recovered for relatively large $dT$ and $dK$ grids. Moreover, call option prices not only change sensitively depending on LV, but are given insufficiently.
2. Finite Difference Method: The Finite Difference Method (FDM) solves Dupire's PDE (2b) directly. Even though there are many papers and results about this, the accuracy stability issue still remains: a state

Received May 17, 2021; Accepted July 28, 2021.
2010 *Mathematics Subject Classification.* 91G20, 91G30.
*Key words and phrases.* Time stepwise local volatility, Dupire's model, Monte Carlo, transition density function, path integral method.
Bae is supported by the Basic Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education and Technology (NRF-2018R1D1A1A09082848), Lim by (NRF-2019R1I1A3A03059382), and BK21 FOUR (Fostering Outstanding Universities for Research, NO.5120200913674) funded by the Ministry of Education(MOE, Korea) and National Research Foundation of Korea(NRF).

space should be tightly discretized to get reasonable accuracy. To obtain stable results, a pre-processing work is required to fine-tune the call option price curves used as an input to FDM to fit the grid structure of FDM (see Appendix A).

3. Monte-Carlo Method: MCM is most popular for pricing derivatives since its flexibility and simplicity allows high-speed calculation of derivatives with complex contingency claims. This method is performed by calculating the expectated values of the future price scenarios generated from Dupire's model (1). LV made from FDM are not well suitable for use in MCM since the values appropriate for the discrete space of the FDM are different from the values required by MCM. We note that when MCM calculates the price, it uses the MC integration, which takes the average of the prices along random paths. For that reason, MCM's randomness causes difficulty to calibrate LV.

**Our contribution.** We introduce the time stepwise LV suitable for MCM and a new calibration method. Instead of creating MC scenarios, we use the integral equation introduced in [6] and numerical integration. And it provides analytic Jacobian and Hessian matrices for the nonlinear optimization problem of the time stepwise LV. While implementing this idea, we proceed in the following order: 1) to construct a precise discretized state space, 2) to use the cubic spline approximation of the call option price curves, 3) to do the calibration process of Arrow-Debreu prices, and finally, 4) to compute the time stepwise LV.

**Organization.** In Section 2, introduces the time stepwise LV. We explain the relationship between the transition density function inherent in Dupire's model, and the Arrow-Debreu price and market call option prices to be used for our method. We review results in [5, 6], which are main theoretical basis of our method, and sketch a brief idea for calculation of the time stepwise LV. In Section 3, we implement our theory based on the path integral formula, and mechanically discretize the needed contents for applying the theorem. In Section 4, conclusion and a practical example are presented.

## 2. Time stepwise local volatility

### 2.1. Dupire's model and partial differential equation

Dupire's model is the following ([9, 10]):

$$\frac{dS(t)}{S(t)} = (r(t) - q(t))dt + \sigma(S(t), t)dW(t). \tag{1}$$

Here, $S(t)$ is a stock price at time $t > 0$, $S(0) = S_0$ its initial price, $r(t)$ and $q(t)$ are instantaneous interest and dividend rates, and $W(t)$ is the Wiener process with a risk neutral measure. The coefficient of the Wiener process, $\sigma(S(t), t)$, is a function of time $t$ and $S(t)$, called the local volatility (LV). The existence and the uniqueness of a solution to Dupire's stochastic differential equation (1)

and its associated transition density function $P$ are well known [13, Theorem 5.2.1].

We denote by $\mathcal{S} := \{(K,T) \mid K > 0, T > 0\}$ the state space consisting of all possible future strikes and maturities which can be reached by the process (1). The transition density function $P = P(K,T)$ starting from the origin $(S_0, 0)$ and reaching $(K,T)$ is a solution of the Kolmogorov forward equation (2a). The call option prices with different maturities and strikes of the same index satisfy Dupire's partial differential equation (PDE) (2b),

$$\frac{\partial P}{\partial T} = \frac{1}{2}\frac{\partial^2(\sigma^2 K^2 P)}{\partial K^2} - \frac{\partial((r(T)-q(T))KP)}{\partial K} \qquad \text{Kolmogorov Equation,} \tag{2a}$$

$$\frac{\partial C}{\partial T} = \frac{\sigma^2 K^2}{2}\frac{\partial^2 C}{\partial K^2} - (r(T)-q(T))K\frac{\partial C}{\partial K} - q(T)C \quad \text{Dupire's PDE,} \tag{2b}$$

where $C := C(K,T)$ is a call option price and $\sigma := \sigma(K,T)$ is an instantaneous LV with strike $K$ and maturity $T$.

### 2.2. Implied and transition density function

The transition density function $P(S_T,T)$ arriving at $(S_T,T)$ is, in particular, called an implied probability density function of the market. The analytic form of $P(S_T,T)$ is not known. The probability function $P(S_T,T)$ associated to (1) satisfies the following Chapman-Kolmogorov equation [11, Eqs. 2.23, 2.24],

$$P(S_T,T) = \int_0^\infty p(S_T,T \mid S_t,t;\sigma)P(S_t,t)dS_t, \tag{3}$$

where $p(end \mid start;\sigma)$ is a transition density function from *start* to *end* with volatility $\sigma$. From a practical point of view, since Dupire's model is handled in a discretized space, the time and stock price ranges are discretized to the 2-dimensional grid. In this case, as shown in Figure 2, it is assumed that the instantaneous LV of the Dupire model is constant from the starting point of the grid to all subsequent grid points. Let us refer to this as a time stepwise LV, which connects one point of the state space to all other points of the next time. If the next time of $t$ is $T$, then the transition density function $p$ connecting a starting point $(S_t,t)$ to an ending point $(S_T,T)$ is obtained from (1) with a constant volatility $\sigma$,

$$p(S_T,T \mid S_t,t;\sigma) = \frac{1}{S_T\sigma\sqrt{2\pi(T-t)}}\exp\left(-\frac{(\ln S_T - \ln S_t - \bar{\mu})^2}{2\sigma^2(T-t)}\right), \tag{4}$$

where $\bar{\mu} := (\bar{r} - \bar{q} - \frac{1}{2}\sigma^2)(T-t)$, $\bar{r} := \int_t^T r(s)ds/(T-t)$, $\bar{q} := \int_t^T q(s)ds/(T-t)$, $T > t \geq 0$.

### 2.3. Arrow-Debreu prices

The discounted implied density function $e^{-\int_0^T r(t)dt}P(S_T,T)$ of (3) is called the Arrow-Debreu (AD) price [4]. AD price is the present value of the security

$AD(S_T, T)$, whose payoff function $\text{Payoff}_{AD}(S_T, T)$ is 1 when the stock price $S$ at time $T$ is $S_T$ and 0 otherwise, that is,

$$\text{Payoff}_{AD}(S_T, T) = \delta(S_T - S) := \begin{cases} 1, \text{ if } S = S_T, \\ 0, \text{ otherwise.} \end{cases} \tag{5}$$

As in the case of $p(S_T, T \mid S_t, t)$, AD price starting from time $t \geq 0$ is directly inferred from its definition:

$$AD(S_T, T) := e^{-\int_0^T r(t)dt} P(S_T, T) \quad \text{for } t = 0, \tag{6a}$$

$$ad(S_T, T \mid S_t, t; \sigma) := e^{-\int_t^T r(u)du} p(S_T, T \mid S_t, t; \sigma) \quad \text{for } t > 0. \tag{6b}$$

In [5], authors have obtained the relationship between transition (probability) density function (or AD price) and the price of the European call options in the following:

$$C(K, T) = e^{-\int_0^T r(t)dt} \int_K^\infty (S - K)\, P(S, T) dS, \tag{7a}$$

$$\frac{\partial C}{\partial K} = -e^{-\int_0^T r(t)dt} \int_K^\infty P(S, T) dS \leq 0, \tag{7b}$$

$$\frac{\partial^2 C}{\partial K^2} = e^{-\int_0^T r(t)dt} P(K, T) \geq 0, \tag{7c}$$

where $C(K, T)$ is a current value of a call option price with strike $K$ and maturity $T$. With similar notations to (3) and (6), we denote by $c(K, T \mid S_t, t)$ the $(S_t, t)$ start call option price with strike $K$ and maturity $T$, clearly $C(K, T) = c(K, T \mid S_0, 0)$. The specific method of generating a precise AD price from the call option price curve is described in Appendix A.

### 2.4. Path integral formula

Feynman-Kač stochastic representation theorem [13] state that:

**Theorem 2.1** (Feynman–Kač)**.** *Assume that $C = C(K, T \mid S, t)$ is a $(S, t)$ value (solution) of the boundary value problem of a call option with strike $K$ and maturity $T$.*

$$\frac{\partial C}{\partial t} + \mu(S, t)\frac{\partial C}{\partial S} + \frac{1}{2}\sigma(S, t)^2 S^2 \frac{\partial^2 C}{\partial S^2} - rC = 0,$$
$$C(K, T \mid S, T) = \max(S - K, 0),$$

*where $C = C(K, T \mid S, t)$.*
*Assume furthermore that the process $\sigma(S, t)^2 S^2 \frac{\partial C}{\partial S}(u, S_u)$ is in $\mathcal{L}^2$ [1], where $S$ is defined* (1)*. Then $C$ has the representation*

$$C(K, T \mid S, t) = \mathbb{E}_{S,t}\left[\max(S(T) - K, 0)\right],$$

[1]Function $f \in \mathcal{L}^2[a, b]$ implies $\int_a^b \mathbb{E}[f^2(S(u))]du < \infty$ and $f(S(t))$ is adapted to a filtration generated by the process $S(u), u \leq t$ up to time $t$.

*where $\mathbb{E}_{S,t}$ implies expectation under the filtration generated by the process $S$ up to time $t$ with $S(t) = S$.*

*Proof.* From reference Oksendal [13, Theorem 8.2.1]. □

Since $S$ is a stochastic variable, and $u < T$, then we have the law of iterated expectations result.

$$C(K, T \mid S_0, 0) = \mathbb{E}_{S_0,0}\big[\mathbb{E}_{S,u}\left[\max(S(T) - K, 0)\right]\big],$$

where $C(K, T \mid S_0, 0) = C(K, T)$ from our previous notation, which can be represented as an integral form.

**Theorem 2.2** (Path integral call options formula)**.** *A European call option $C(K, T)$ maturing at time $T$ and strike $K$ relates to the start time $u$, $0 < u < T$ value of a continuum of forward start European call options with strike $K$ and matures at $T$.*

$$\begin{aligned} C(K, T \mid S_0, 0) &= \int_0^\infty \max(S - K, 0) AD(S, T) dS \\ &= \int_0^\infty \underbrace{\left(\int_0^\infty \max(S - K, 0) ad(S, T \mid S_u, u; \sigma) dS\right)}_{c(K,T \mid S,u;\sigma)} AD(S_u, u) dS_u, \end{aligned} \tag{8a}$$

$$c(K, T \mid S, u; \sigma) = e^{-\int_u^T q(s)ds} N(d^1) - e^{-\int_u^T r(s)ds} K N(d^2), \tag{8b}$$

*The formula in underbrace is a $(S_u, u)$ value of a call option, which could be calculated from Black-Scholes formula, where $\sigma = \sigma(S, u)$,* (8b) *is the Black-Scholes formla for the call option price,*

$$d^1 := \frac{\ln(K/S) + \int_u^T (r(s) - q(s))\, ds}{\sigma\sqrt{T-u}} + \frac{1}{2}\sigma\sqrt{T-u}, \quad d^2 := d^1 - \sigma\sqrt{T-u}$$

*and $AD(S_u, u) = \frac{\partial^2}{\partial S_u^2} C(S_u, u)$ from Breeden's* (7)*. This is sometimes called "path integral formula" in mathematical finance, and play a key role in our method.*

*Proof.* trivial. □

Equation (8a) implies the replication of a market call option price using the discounted implied density $AD(S, u)$ and equation (8b) implies the strike $K$ and maturity $T$ call option value at $\sigma$ at $(S, u)$. It should be noted that the volatility $\sigma$ in (8a) is a fixed at $(S, u)$ and is applied up to all strike prices $K$ at time $T$. Therefore, $c(K, T \mid S, u; \sigma)$ is simply expressed by using the Black-Scholes formula (8b).

### 2.5. Path generation using time stepwise local volatility

Consider the case of creating a scenario of stock prices at times $T_1, \ldots, T_m$. The simulated stock prices $S^0, S^1, \ldots, S^m$ [2] should satisfy (9a), which is an

[2]Super subscripts were used to distinguish them from discretized nodes in stock prices.

exact solution of (1). It must be equivalent to (9b) which uses the constant volatility $\bar{\sigma}_i$,

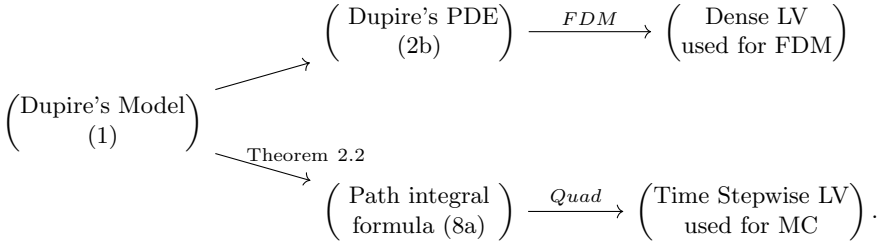
$$S^{i+1} = S^i \cdot \exp\left(\int_{T_i}^{T_{i+1}} (r(u)-q(u)-\frac{1}{2}\sigma(S,u)^2)du+\int_{T_i}^{T_{i+1}} \sigma(S,u)dW(u)\right) \tag{9a}$$

$$= S^i \cdot \exp\left(\int_{T_i}^{T_{i+1}} (r(u)-q(u)-\frac{1}{2}\bar{\sigma}_i^2)du+\bar{\sigma}_i W(\tau_i)\right), \tag{9b}$$

where $\tau_i := T_{i+1}-T_i$ and $\bar{\sigma}_i := \bar{\sigma}(S^i, T_i)$ is the volatility that is fixed at $(S^i, T_i)$. We call it a time stepwise LV.

### 2.6. Comparison of dense and time stepwise LV

From (1), we obtain Dupire's PDE (2b) and a path integral formula (8a) (Theorem 2.2). Calculation diagram of (2b) and (8a) is described in the following:




Here, *FDM* means calibration using FDM as its pricing scheme. The local volatility $\sigma(S,u)$ obtained by *FDM* exists on a dense mesh which FDM requires. The method *Quad* is the calibration using the numerical integration(quadrature) as its pricing scheme, the time stepwise volatility $\bar{\sigma}(S(t_i))$ obtained.

### 2.7. Overview of the time stepwise local volatility calibration

**Step 1**. For each maturity $T$, assume that the possible stock price range is discretized to $S_1, \ldots, S_N$, and we have call option prices $C(S_i, T)$ and piecewise linear AD price curve $\bar{AD}(S_i, T)$ at the discretized stock price nodes is derived from the cubic spline approximation of call option price curve, which is explained in Appendix A.

**Step 2**. The equation (8a) is rewritten as the quadrature rule such that

$$C(K_i,T) \approx \sum_{l=1}^{N} w_l \, \underline{c(K_i, T \mid S_l, u; \bar{\sigma}_l)} \, \bar{AD}(S_l, u) =: f_i(\bar{\boldsymbol{\sigma}}), \tag{10}$$

where $u < T$, $w_l$ is a weight for specific quadrature rule.[3] The right-hand side is a function $f_i$ of $\bar{\boldsymbol{\sigma}} = (\bar{\sigma}_1, \ldots, \bar{\sigma}_N)$. For each $i$, $C(K_i, T)$ is a given market call option price.

**Step 3**. We construct the least square problem for finding $\bar{\boldsymbol{\sigma}}$ of loss function $F$:

$$F(\bar{\boldsymbol{\sigma}}) := \min_{\bar{\boldsymbol{\sigma}}} \sum_{i=1}^{N} |v_i(C(K_i, T) - f_i(\bar{\boldsymbol{\sigma}}))|^2.$$

Commonly, this nonlinear least squares optimization problem is solved by using TRF method [14]. Since the method we present easily calculates analytic Jacobian and Hessian, it guarantees superlinear convergence by applying the TRF-Newton method. The weighting factor $v_i > 0$ is introduced for enhancing the accuracy for the relatively small value term.[4]

## 3. Implementation of time stepwise local volatility

Based on Theorem 2.2, the process of reassembling the equation (8a) in a discrete state space is described. At this time, the market volatility, which is the data required for input, and the call option price obtained from it must satisfy the following conditions.

### 3.1. Requirements and necessary conditions

1. Piecewise Linear Approximation of Arrow-Debreu Price Curves: In Appendix A, we show that a piecewise linear approximation of the AD price curve is derived from the cubic spline approximation of the call option price curve. AD price is a discounted probability density function, so the integral domain is limited to finite intervals.
   **Note**: Even if we do not use this part, our method works well as long as a call option price is provided that does not violate the no-arbitrage conditions. Method A systematically corrects the arbitrage opportunity for a given call option price as input and generates Arrow-Debreu price[5].
2. Conditions for Call Option Price Curves: Let's look at Breeden's relation (7) again. The first equation is negative and the second is positive. Adding no calendar arbitrage condition (11c) gives the necessary conditions for the call option price surface, which are important conditions to prevent negative transition probability density of (1),

(11a) $$\frac{\partial C(K,T)}{\partial K}\Big|_{K_i} < 0, \qquad \text{negative slope for strike derivative}$$

[3]We use trapzoidal and Simpson's rule for exact matching the regular strike nodes.

[4]The time values of the in/out of the money options are very small, resulting in a minor contribution to the total error. From the view point of root finding, it gives relatively large learning rate for speeding up.

[5]This method is included in the author's paper, yet to be published.

(11b) $\left.\frac{\partial^2 C(K,T)}{\partial K^2}\right|_{K_i} > 0,$ convexity for 2nd derivatives for strike

(11c) $C(K,T) < C(K^*,U),$ no calendar arbitrage

where $T < U$ and $Ke^{\int_T^U (r(s)-q(s))ds} = K^*$.

3. Conditions for Risk Neutral Pricing: Since the call option price function is approximated by using the cubic splines in Section A, the second derivative with respect to the exercise price becomes a piecewise linear function which represents an AD Price. The following three conditions should be satisfied: 1) The Price Matching Condition that restores the call option price observed at the regular strike price node point, 2) The discounted density function, which is the condition of the distribution function, and 3) The forward risk neutral condition:

(12a)

Price Matching Condition: $$\int_0^\infty \text{Max}(S - K_i, 0) AD(S,T) dS = C(K_i, T),$$

(12b)

Discounted Density Function: $$\int_0^\infty AD(S,T) dS = e^{-\int_0^T r(s) ds},$$

(12c)

Forward Risk Neutrality: $$\int_0^\infty S \cdot AD(S,T) dS = e^{-\int_0^T q(s) ds} S(0),$$

where $K_i,\ i = 1, \ldots, N$, is a regular exercise node for the stock index options markets and $C(K_i, T)$ is observed call option price at strike $K_i$ and maturity $T$. For condition (12b), the trapezoidal rule is applied, and for conditions (12a) and (12c), Simpson's rule is used for numerical integrals. Since the integration domain is a semi-infinite and the observed values are matched only at the given regular strike price node, the integral range is properly truncated and variably discretized to include the strike price nodes.

### 3.2. Discrete state space construction

**3.2.1.** *Technical considerations.* This paragraph describes the technical points that pays attention for implementation.

1. *Numerical Integration Rule*
   We use trapezoidal and Simpson's rules, which are simple non-Gaussian type integration methods. Although the Gaussian quadrature is generally an effective method, the reasons for choosing it are as follows: (1) The strike prices should be a subset of node points, called abscissas for the integration. (2) The AD price we generate is given as a piecewise linear function, a linear function between the nodes. (3) AD price,

which is a piecewise linear function with positive value, must have a bounded region of integration.

2. *Grid Refinment for Optimization and Numerical Integration*
For simplicity, we use the unitized stock and strike prices, which are called the spot moneyness. Define the unitized nodes (spot moneyness) as $\boldsymbol{x}$ given by,

$$\boldsymbol{x} = \{x_1, \ldots, x_N\}, \tag{13}$$

where $x_1 = 0$ and initial starting point $x_o = 1$, for some $o = i \in \{1, \ldots, N\}$. The number of index $N > n$, which is more finer than unitized regular strike nodes: $\{K_1/S_0, \ldots, K_n/S_0\} \subset \boldsymbol{x}$. The reason $N > n$ is that the number of nodes must be large for the accuracy of numerical integration. Next, we truncate the semi-infinite region to $x_{\max} = x_N$ such that $\int_{x_{\max}}^{\infty} P(x)dx < \epsilon$. However, since $P(x)$ is a non-negative piecewise linear function (23), it must be zero for $x > x_{\max}$ for all of the observed maturities $T_j$, $j = 0, \ldots, m$.

3. *Refinment of the First Time Node*
Since the call option prices at $T_1$ have different implied volatilities, they cannot be created using one fixed volatility at the origin. We assume that the process (1) starting at $T_0 = 0$ with a fixed volatility $\sigma_0$ reflects the diffusion over the intermediate point and then moves to $T_1$ to generate a call option prices with different implied volatilities at $T_1$. This is important for pricing with LV, especially when using MCM. Therefore, let's mark the mid-point of $[0, T_1]$ with $T_{1/2}$. The initial volatility $\sigma_0$ must also meet the calendar arbitrage condition $bscall(x, T_{1/2}; \sigma_0) < C(x^{\text{forward}}, T_1)$ for $x \in \boldsymbol{x}$ where $C(x^{\text{forward}}, T_1)$ is the strike $x^{\text{forward}} = xe^{\int_{T_{1/2}}^{T_1}(r(s)-q(s))ds}$ call option price with maturity $T_1$, and $bscall(x, T_{1/2}; \sigma_0)$ is the Black-Scholes call option price with strike $x$ and maturity $T_{1/2}$.

**3.2.2.** *Assembly of discrete state space.* The practical LV following Dupire's model is a function on the state space $\mathcal{S}$. And the mesh we construct has a variable time interval, and discretized spot-moneyness $\boldsymbol{x}$.
The followings are variables in discrete state spaces.

(14a)
$\mathcal{S}$ : discretized state space, $\{x_i^j := (x_i, T_j) \,|\, i \in I, j \in J\}$,

(14b)
$\sigma_i^j$ : stepwise volatility which is fixed from $x_i^j$ to $x_l^{j+1}$, $\forall l \in I$,

(14c)
$a_{i,l}^j$ : one time step Arrow-Debreu price from $x_i^j$ to $x_l^{j+1}$ with $\sigma_i^j$,

(14d)
$A_i^j$ : Arrow-Debreu price from the origin $x_o$ to $x_i^j$,

(14e)
$P_i^j$ : probability(marginal) Density from $x_o$ to $x_i^j$,

(14f)
$w_i$ : weight for the Simpson or Trapzoidal numerical integration scheme,

(14g)
$c_{l,i}^j$ : strike $x_i$, maturity $T_{j+1}$ call option price starting at point $x_l^j$ with $\sigma_l^j$,

(14h)
$C_i^j$ : strike $x_i$, maturity $T_j$ call option price at $x_o$ with implied volatility $\Sigma_i^j$,

(14i)
$\sigma_0$ : initial starting volatility from $x_o(=1)$ to $x_l^1$, $\forall l \in I$.

Here, $I = \{1, \ldots, N\}, J = \{0, 1/2, 1, \ldots, m\}$. The AD price starting at $x_o$ arriving $x_i^j$ is $A_i^j = e^{-\int_0^{T_j} r(s)ds} P_i^j$, which comes from the effect of all different stepwise local volatilities $\sigma_l^k$, $l \in I$, $k < j$ applied to the dynamics $x(t) = S(t)/S_0$ starting at $x_o$ and reaching $x_i^j$. Assuming that we know $A_i^j$, then our purpose is to compute each of the $\sigma_l^k$, $k < j$ from (3), (8). A discrete relationships between $C_i^{j+1}$ and $\{c_{l,i}^j, a_{l,i}^j\}$, between $A_i^{j+1}$ and $\{A_l^j, a_{l,i}^j\}$, which corresponds to (8a) and (3), are generated by the following integration rule in the next subsection.

### 3.3. Quadrature rule

Simpson's method uses a formula to obtain an area surrounded by a quadratic equation passing through three points. If $f(x)$ is a quadratic equation and for given three $x$ points $a < c < b$, values $f(a), f(c), f(b)$ are known, then the Lagrange interpolation is the following:

$$f(x) = f(a)\frac{(x-c)(x-b)}{(a-c)(a-b)} + f(c)\frac{(x-a)(x-b)}{(c-a)(c-b)} + f(b)\frac{(x-a)(x-c)}{(b-a)(b-c)}.$$

If $c = \frac{a+b}{2}$, then $\int_a^b f(x)dx = \frac{b-a}{6}\left(f(a) + 4f(c) + f(b)\right)$.

**Lemma 3.1.** *If $f(x)$ and $g(x)$ are both piecewise linear functions with node points $x_1, \ldots, x_N$, and their function values at node points are $f_1, \ldots, f_N$ and $g_1, \ldots, g_N$, respectively, then*

$$\begin{aligned}\int_{x_1}^{x_N} f(x)g(x)dx = \sum_{i=1}^{N-1} \frac{x_{i+1}-x_i}{6}\Bigg(f(x_i)g(x_i) + 4f\left(\frac{x_i+x_{i+1}}{2}\right)g\left(\frac{x_i+x_{i+1}}{2}\right) \\ + f(x_{i+1})g(x_{i+1})\Bigg) \\ = w_1 f_1 + \underline{w_2 f_2 + \cdots + w_{N-1} f_{N-1}} + w_N f_N,\end{aligned}$$

$$\text{where } w_i = \begin{cases} (2g_1 + g_2)\Delta x_1 \text{ for } i = 1, \\ (2g_i + g_{i+1})\Delta x_i + (2g_i + g_{i-1})\Delta x_{i-1} \text{ for } i = 2, \ldots, N-1, \\ (2g_N + g_{N-1})\Delta x_{N-1} \text{ for } i = N, \end{cases}$$

*and* $\Delta x_i = x_{i+1} - x_i,\ i = 1, \ldots, N-1$.

*Proof.* For piecewise linear function $f$, it is clear $f\left(\frac{x_i+x_{i+1}}{2}\right) = \frac{f(x_i)+f(x_{i+1})}{2}$, $i = 1, \ldots, N-1$, the same as $g$. □

In our cases for (8), $AD(S, u)$, which corresponding to $g$, is a piecewise linear function, and we approximate $c$ as a piecewise linear function of $\boldsymbol{x}$.

### 3.4. Backward and forward matrix equations

**Lemma 3.2** (Discrete Version of the path integral call options with call options)**.** *By applying Lemma 3.1, the discrete version of Theorem* 2.2 *is established,*

$$C_i^{j+1} = \sum_{l=1}^{N} w_l c_{l,i}^j A_l^j, \tag{15a}$$

$$c_{l,i}^j = bscall(x_l, x_i, \tau_j, \sigma_l^j), \tag{15b}$$

*where* $c_{l,i}^j = c_{l,i}^j(\sigma_l^j)$ *is a functions of the volatility* $\sigma_l^j$*, which can be easily calculated by Black-Scholes call option formula,*

$$c_{l,i}^j = bscall(x_l, x_i, \tau_j, \sigma_l^j) = dq^j x_l N(d_{l,i}^1) - df^j x_i N(d_{l,i}^2), \tag{16}$$

*where* $df^j, dq^j, d_{l,i}^1, d_{l,i}^2$ *are discretized version of* (8b) *for the time interval* $[T_j, T_{j+1}]$.

Volatilities $\sigma_i^j,\ i \in I$ can be obtained by optimization that minimizes the sum of squares of the differences between reconstructed call option values and real values by setting volatilities as unknown. After finding $\sigma_i^j$, the time $T_{j+1}$, AD prices $AD_i^{j+1}, i \in I$ are calculated by using the discrete version of Chapman-Kolmogorov equation (3),

$$A_i^{j+1} = \sum_{l=1}^{N} w_l A_l^j a_{l,i}^j, \tag{17a}$$

$$a_{l,i}^j = \frac{df^j}{x_l \sigma \sqrt{2\pi\tau}} \exp\left(-\frac{(\ln x_l - \ln x_i + \mu)^2}{2\sigma^2\tau}\right), \tag{17b}$$

where $\sigma = \sigma_l^j, \mu = -\ln df^j + \ln dq^j$ and $\tau = T_{j+1} - T_j$.

**Theorem 3.3.** *Combining the discussions we have discussed with lemmas so far, we can construct two matrix equations with backward and forward schemes that generate stepwise LV and use them to reconstruct the next level of Arrow-Debreu price.*

- *Reconstruction of Call Option*
  *The equation* (15a) *is rewritten as a matrix form*

$$\boldsymbol{C}^{j+1} = [\boldsymbol{c}^j] \cdot (\boldsymbol{w} \odot \boldsymbol{A}^j), \ (\textbf{backward}) \tag{18}$$

*where* $[\boldsymbol{c}^j]$ *is the matrix of forward start one step call option prices with rows*: *starting nodes, cols*: *target node, and* $\odot$ *implies Hadamard product,*

$$\underbrace{\begin{bmatrix} C_1^{j+1} \\ C_2^{j+1} \\ \vdots \\ C_N^{j+1} \end{bmatrix}}_{\boldsymbol{C}^{j+1}} = \underbrace{\begin{bmatrix} c_{11} & c_{21} & \dots & c_{N1} \\ c_{12} & c_{22} & \dots & c_{N2} \\ \vdots & \vdots & \vdots & \vdots \\ c_{1N} & c_{2N} & \dots & c_{NN} \end{bmatrix}}_{[\boldsymbol{c}^j]} \underbrace{\begin{bmatrix} w_1 A_1^j \\ w_2 A_2^j \\ \vdots \\ w_N A_N^j \end{bmatrix}}_{\boldsymbol{w} \odot \boldsymbol{A}^j}.$$

*For example, the first row in matrix representation of* (18) *is a local forward start call option vector with element* $c_{l,i}^j$ *which starts at* $x_l^j$ *with volatility* $\sigma_l^j$, *maturity* $T_{j+1}$, *strike* $x_1$. *Thus, the* $N \times N$ *matrix* $[\mathbf{c}^j]$ *is a function of* $\boldsymbol{\sigma}^j = (\sigma_1^j, \dots, \sigma_N^j)^\top$. *By the inductive assumption for the maturity time* $T_j$, *we have AD price vector* $\boldsymbol{A}^j$, *and the call option price vector* $\boldsymbol{C}^{j+1}$ *from the refined implied volatility surface. Our objective is to find a matrix* $[\mathbf{c}^j]$ *as a function of* $\boldsymbol{\sigma}^j$. *In this step,* (**backward**) (18) *means that this matrix equation is calculated by the backward scheme, given left and right most vectors,* $\boldsymbol{C}^{j+1}$ *and* $\boldsymbol{w} \odot \boldsymbol{A}^j$, *which is a main topic for the next section. After finding* $\boldsymbol{\sigma}^j$, *we obtain, at time* $T_{j+1}$, *AD price vector* $\boldsymbol{A}^{j+1}$ *by the formula* (19) *using the calculated* $\boldsymbol{\sigma}^j$.

- *Reconstruction of Arrow-Debreu Price*

$$\boldsymbol{A}^{j+1} = [\boldsymbol{a}^j] \cdot (\boldsymbol{w} \odot \boldsymbol{A}^j), \ (\textbf{forward}), \tag{19}$$

$$\underbrace{\begin{bmatrix} A_1^{j+1} \\ A_2^{j+1} \\ \vdots \\ A_N^{j+1} \end{bmatrix}}_{\boldsymbol{A}^{j+1}} = \underbrace{\begin{bmatrix} a_{11} & a_{21} & \dots & a_{N1} \\ a_{12} & a_{22} & \dots & a_{N2} \\ \vdots & \vdots & \vdots & \vdots \\ a_{1N} & a_{2N} & \dots & a_{NN} \end{bmatrix}}_{\boldsymbol{a}^j} \underbrace{\begin{bmatrix} w_1 A_1^j \\ w_2 A_2^j \\ \vdots \\ w_N A_N^j \end{bmatrix}}_{\boldsymbol{w} \odot \boldsymbol{A}^j}.$$

*The colored first row in matrix representation of* (19) *is a vector* $(\boldsymbol{a}^j)_{l \to 1}^\top$. (**forward**) *implies this matrix equation is normally calculated using multiplicative two known right hand side terms to obtain* $\boldsymbol{A}^{j+1}$.

### 3.5. Calibration

Equation (18) is a system of nonlinear equations, and left and right most hand side vectors are already known, $[\boldsymbol{c}^j]$ is a function of $\boldsymbol{\sigma}^j$. In fact, this is

a nonlinear minimization problem of dimension $m$. We simplify the equation (18) and define the loss function $\boldsymbol{F}$ as a function of $\boldsymbol{\sigma}(=\boldsymbol{\sigma^j})$ as

$$\boldsymbol{F}(\boldsymbol{\sigma}) = ||\boldsymbol{y} - \boldsymbol{f}(\boldsymbol{\sigma})||^2_{2,\boldsymbol{v}} = \sum_{i=1}^{N} |v_i(y_i - f_i(\boldsymbol{\sigma}))|^2, \tag{20}$$

where $\boldsymbol{y} = \boldsymbol{C}^{j+1}$, $\boldsymbol{f}(\boldsymbol{\sigma}) = [\mathbf{c}^j] \cdot (\boldsymbol{w} \odot \boldsymbol{A}^j)$ and $\boldsymbol{F} : \mathbb{R}^N \to \mathbb{R}^1$ is an $N$-dimensional scalar function. The weight term $\boldsymbol{v} = (v_1, v_2, \ldots, v_N)$, $v_i = 1/\max(vega(x_i), \epsilon)$ for some $\epsilon > 0$, $vega(x_i)$ is same as (21c), but is calculated using average market volatility. It is the optional correction factor that multiplied by the values that come out too small to match [3].

### 3.6. Analytic Jacobian and Hessian matrices

The representative algorithm for solving the multivariate nonlinear least squares problem is the hybrid algorithm in [14], which uses the Trust Region dogleg method [12] and Levenberge-Marquardt algorithm [12] using the damping factor. These methods solve approximated quadratic equation for every stage. In this process, methods need Jacobian or Hessian or pseudo Hessian $(\boldsymbol{J_f}^\top \boldsymbol{J_f})$ matrices for the accuracy and stability as well as for the computational speed. If Jacobian and Hessian are not provided to the optimization solver, the calculations are very slow, and stability and accuracy problems occur seriously. In the presented method, these derivatives can be calculated in analytic form (21c). This is one of the biggest advantages of using the proposed method.

**Lemma 3.4** (Analytic Jacobian and Hessian for the Loss function $\boldsymbol{F}$)**.** *The Jacobian and Hessian of the loss function $\boldsymbol{F}$ is calculated as follows*:

$$\nabla \boldsymbol{F}(\boldsymbol{\sigma}) = \sum_{i=1}^{N} f_i(\boldsymbol{\sigma}) \nabla f_i(\boldsymbol{\sigma}) = \boldsymbol{J_f}^\top(\boldsymbol{\sigma}) \boldsymbol{f}(\boldsymbol{\sigma}),$$

$$\nabla^2 \boldsymbol{F}(\boldsymbol{\sigma}) = \underbrace{\sum_{i=1}^{N} \nabla f_i(\boldsymbol{\sigma}) \nabla f_i(\boldsymbol{\sigma})^\top}_{\boldsymbol{J_f}^\top \boldsymbol{J_f}} + \sum_{i=1}^{N} f_i(\boldsymbol{\sigma}) \nabla^2 f_i(\boldsymbol{\sigma}).$$

*Proof.* We omit the time $j$-superscription for brevity, the $i$-component $f_i$ of $\boldsymbol{f}$,

$$f_i(\boldsymbol{\sigma}) = \sum_{l=1}^{N} w_l A_l c_{l,i}(\sigma_l), \tag{21a}$$

$$\frac{\partial f_i}{\partial \sigma_l} = w_l A_l \frac{\partial c_{l,i}(\sigma_l)}{\partial \sigma_l}, \tag{21b}$$

$$\frac{\partial^2 f_i}{\partial \sigma_k \partial \sigma_l} = w_l A_l \frac{\partial^2 c_{l,i}(\sigma_l)}{\partial^2 \sigma_l} \quad \text{for } k = l, \qquad 0 \text{ otherwise}, \tag{21c}$$

where $l \in I$, $d^1_{l,i}$, $dq = dq^j$, $\tau = \tau_j$ are the same as in (16), and $\phi$ is the standard normal probability function. The partial derivatives are calculated to the following:

$$\frac{\partial c_{l,i}(\sigma_l)}{\partial \sigma_l} = dq\, x_l \phi(d^1_{l,i})\sqrt{\tau}, \tag{22a}$$

$$\frac{\partial^2 c_{l,i}(\sigma_l)}{\partial^2 \sigma_l} = dq\, x_l \phi(d^1_{l,i})\sqrt{\tau}\frac{d^1_{l,i}d^2_{l,i}}{\sigma_l} \quad \text{for } l \in I. \tag{22b}$$

Form (21) and (22), the Jacobian matrix of $\boldsymbol{f}$ with respect to $\boldsymbol{\sigma}$ can be expressed as a $N \times N$ matrix,

$$\begin{aligned}\boldsymbol{J_f} &= (\nabla f_1, \nabla f_2, \ldots, \nabla f_N)^\top \\ &= \left[ w_l A_l \frac{\partial c_{l,i}(\sigma_l)}{\partial \sigma_l} = dq\, w_l x_l A_l \phi(d^1_{l,i})\sqrt{\tau} \right]_{i,l=1,\ldots,N},\end{aligned}$$

where $d^1_{l,i}$ is in (16). The Gradient and Hessian matrices of the loss function $\boldsymbol{F}$ are as in the lemma. □

### 3.7. Using only the real strike nodes volatilities

It is more efficient to matching the values to the nodes with real strikes than to use all the $x$-grids $\boldsymbol{x}$. However, the reconstruction of the call option price needs entire node points of $\boldsymbol{x}$, so the volatility values of the remaining nodes applies linear interpolation. Both ends of the entire node and the two left and right most ends of the used nodes can be left in a fixed value or a straight line with a slope. The results are shown in Figure 1.

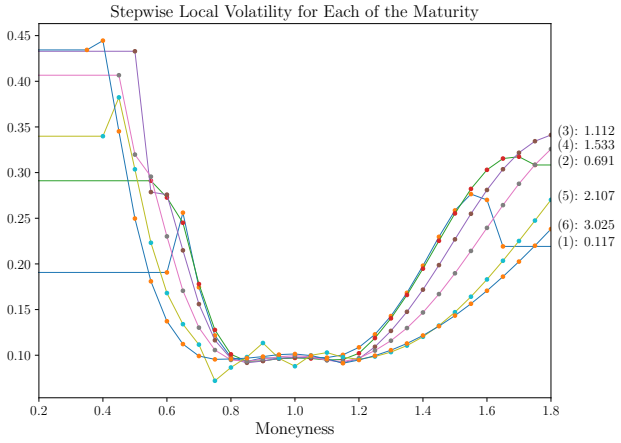



FIGURE 1. Generated Local Volatility S&P500, Date: 07/11/2019

## 4. Practical example and conclusion

We have shown the method of calibrating the LV surface based on the integral equation. The popularly used method is FDM, which suffers from discretization error. To obtain an available LV, the state space needs to be closely discretized. These make a difference when applying MCM to derivatives pricing with local volatilities which are calibrated from FDM. To handle this inconsistency, we adopt the quadrature method for the pricing tool of calibration. The important characteristic of our methodology is to use AD prices as a building block.

Table 1 and Figure 1 show S&P500 time stepwise LV at 07/11/2019, values, and piecewise linear interpolated curves for each option maturities. Dotted points are real strike nodes for used in optimization. Table 2 compare call option values using the implied volatility and resulting time stepwise LV. The price is multiplied by 100 to see the value on a percent basis. Figure 2 illustrates piecewise linear AD price at maturity as described in Section A. Figure 3 shows a picture of the call option reconstruction method presented. The process of restoring prices by embedded volatility by LV is similar to a multi-layer functional neural network.

Remaining options maturity dates are 43, 253, 407, 561, 771, 1107, 1499 and left most points of moneyness are 0.6, 0.55, 0.5, 0.45, 0.45, 0.35, 0.35 and rightmost points 1.65, 1.75, 1.85, 2.0, 2.1, 2.2. The unitized strike nodes were subdivided into 0.025 intervals, of which the used node points were 0.05 intervals. The total time spent on optimization tasks with seven maturities and initial additional processing at $t_{1/2}$ is 5.78 seconds. In addition, the processing time for generating piecewise linear AD price curves described in Appendix is 2.43 seconds in Python version 3.6 with equipment: 1.1GHz dual core Intel Core M CPU notebook.

## Appendix A. Constrained calibration of Arrow-Debreu price

Least Squares Cubic Spline approximation. Let $C(x)$ be a cubic spline polynomial with knots $x_1, \ldots, x_N$,

$$C(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \tag{23}$$

where $x \in [x_i, x_{i+1}]$, $i = 1, \ldots, N-1$. The polynomial $C(x)$ with the condition $c_1 = c_N = 0$ at the end points $x_1, x_N$ is called the natural cubic spline. Following [8, Ch. XIV], the relation of $\boldsymbol{a} = (a_1, \ldots, a_N)^\top$ and $\boldsymbol{c} = (c_1, \ldots, c_N)^\top$ is written in the matrix form

$$\boldsymbol{R}\boldsymbol{c} = 3\boldsymbol{Q}^\top \boldsymbol{a}, \tag{24}$$

where $c_1 = c_N = 0$, $\boldsymbol{R}$ is a symmetric tridiagonal matrix of dim $n \times n$ of which $i = 2, \ldots, N-1$ rows are $\Delta x_{i-1}$, $2(\Delta x_{i-1} + \Delta x_i)$, $\Delta x_i$. As a result, $\boldsymbol{R}$ is

TABLE 1. S&P500 VOLATILITY 07/11/2019, Spot: 3085.18

| Maturity(days) | $t_{1/2}$ | 43 | 253 | 407 | 561 | 771 | 1107 | 1499 |
|---|---|---|---|---|---|---|---|---|
| ZeroRate | 0.01708 | 0.01708 | 0.01795 | 0.01727 | 0.01685 | 0.01633 | 0.01604 | 0.01592 |
| DividendRate | 0.01510 | 0.01510 | 0.01648 | 0.01668 | 0.01645 | 0.01626 | 0.01627 | 0.01633 |
| 0.50 | 0.2317 | 0.1907 | 0.2911 | 0.4329 | 0.3197 | 0.3036 | 0.2497 | 0.2162 |
| 0.55 | 0.2317 | 0.1907 | 0.2911 | 0.2787 | 0.2956 | 0.2232 | 0.1809 | 0.1581 |
| 0.60 | 0.2317 | 0.1907 | 0.2727 | 0.2759 | 0.2301 | 0.1681 | 0.1372 | 0.1227 |
| 0.65 | 0.2317 | 0.2561 | 0.2450 | 0.2148 | 0.1706 | 0.1340 | 0.1122 | 0.1038 |
| 0.70 | 0.3368 | 0.1744 | 0.1780 | 0.1560 | 0.1302 | 0.1116 | 0.0991 | 0.0952 |
| 0.75 | 0.4035 | 0.1216 | 0.1277 | 0.1164 | 0.1057 | 0.0720 | 0.0955 | 0.0922 |
| 0.80 | 0.2224 | 0.0976 | 0.1010 | 0.0964 | 0.0948 | 0.0866 | 0.0959 | 0.1027 |
| 0.85 | 0.1407 | 0.0931 | 0.0934 | 0.0917 | 0.0933 | 0.0980 | 0.0968 | 0.1032 |
| 0.90 | 0.0993 | 0.0972 | 0.0938 | 0.0937 | 0.0957 | 0.1134 | 0.0985 | 0.0972 |
| 0.95 | 0.0500 | 0.0967 | 0.0962 | 0.0965 | 0.0983 | 0.0966 | 0.1007 | 0.0930 |
| 1.00 | 0.0500 | 0.0970 | 0.0973 | 0.0978 | 0.0995 | 0.0879 | 0.1014 | 0.1004 |
| 1.05 | 0.0507 | 0.0974 | 0.0964 | 0.0974 | 0.0983 | 0.0999 | 0.0996 | 0.1060 |
| 1.10 | 0.1051 | 0.0974 | 0.0948 | 0.0951 | 0.0944 | 0.1029 | 0.0959 | 0.0962 |
| 1.15 | 0.1696 | 0.1004 | 0.0953 | 0.0919 | 0.0930 | 0.0978 | 0.0912 | 0.0986 |
| 1.20 | 0.0936 | 0.1086 | 0.1021 | 0.0960 | 0.0967 | 0.0955 | 0.0948 | 0.0982 |
| 1.25 | 0.4045 | 0.1228 | 0.1188 | 0.1092 | 0.1051 | 0.0986 | 0.0995 | 0.0971 |
| 1.30 | 0.1954 | 0.1429 | 0.1404 | 0.1266 | 0.1159 | 0.1034 | 0.1055 | 0.0972 |
| 1.35 | 0.4301 | 0.1683 | 0.1660 | 0.1476 | 0.1298 | 0.1106 | 0.1129 | 0.0992 |
| 1.40 | 0.4656 | 0.1981 | 0.1948 | 0.1718 | 0.1468 | 0.1202 | 0.1216 | 0.1034 |
| 1.45 | 0.1440 | 0.2297 | 0.2253 | 0.1987 | 0.1670 | 0.1324 | 0.1318 | 0.1096 |
| 1.50 | 0.0968 | 0.2586 | 0.2553 | 0.2269 | 0.1897 | 0.1471 | 0.1434 | 0.1179 |

TABLE 2. Call Option Price from Implied Volatility and Local Volatility, S&P500, 7/11/2019

| Moneyness | $t_1 = 0.1175$ | IV, LV | $t_2 = 0.691$ | IV, LV | $t_3 = 1.112$ | IV, LV | $t_4 = 1.533$ | IV, LV | $t_5 = 2.107$ | IV, LV | $t_6 = 3.025$ | IV, LV | $t_7 = 4.096$ | IV, LV |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0.500 | 49.484 | 49.484 | 49.113 | 49.113 | 48.786 | 48.786 | 48.324 | 48.325 | 47.567 | 47.568 | 46.688 | 46.691 | 45.845 | 45.852 |
| 0.525 | 47.015 | 47.015 | 46.661 | 46.661 | 46.349 | 46.349 | 45.909 | 45.909 | 45.185 | 45.187 | 44.348 | 44.352 | 43.546 | 43.556 |
| 0.550 | 44.546 | 44.546 | 44.208 | 44.208 | 43.913 | 43.913 | 43.493 | 43.494 | 42.804 | 42.807 | 42.009 | 42.015 | 41.252 | 41.264 |
| 0.575 | 42.076 | 42.076 | 41.756 | 41.756 | 41.477 | 41.477 | 41.078 | 41.079 | 40.424 | 40.428 | 39.675 | 39.682 | 38.966 | 38.979 |
| 0.600 | 39.607 | 39.607 | 39.304 | 39.304 | 39.041 | 39.041 | 38.663 | 38.665 | 38.046 | 38.051 | 37.346 | 37.355 | 36.692 | 36.705 |
| 0.625 | 37.138 | 37.138 | 36.851 | 36.851 | 36.604 | 36.605 | 36.249 | 36.252 | 35.673 | 35.677 | 35.029 | 35.038 | 34.436 | 34.448 |
| 0.650 | 34.669 | 34.669 | 34.399 | 34.399 | 34.169 | 34.170 | 33.836 | 33.841 | 33.306 | 33.309 | 32.727 | 32.736 | 32.203 | 32.213 |
| 0.675 | 32.200 | 32.200 | 31.946 | 31.947 | 31.734 | 31.736 | 31.428 | 31.434 | 30.952 | 30.951 | 30.449 | 30.456 | 30.003 | 30.011 |
| 0.700 | 29.731 | 29.731 | 29.495 | 29.495 | 29.301 | 29.305 | 29.027 | 29.035 | 28.617 | 28.610 | 28.203 | 28.207 | 27.844 | 27.850 |
| 0.725 | 27.261 | 27.261 | 27.044 | 27.045 | 26.874 | 26.879 | 26.641 | 26.648 | 26.311 | 26.297 | 25.998 | 26.000 | 25.735 | 25.742 |
| 0.750 | 24.793 | 24.793 | 24.597 | 24.599 | 24.458 | 24.463 | 24.277 | 24.284 | 24.046 | 24.029 | 23.848 | 23.846 | 23.687 | 23.695 |
| 0.775 | 22.325 | 22.325 | 22.160 | 22.161 | 22.063 | 22.067 | 21.950 | 21.953 | 21.834 | 21.820 | 21.762 | 21.759 | 21.708 | 21.720 |
| 0.800 | 19.862 | 19.862 | 19.740 | 19.741 | 19.702 | 19.703 | 19.675 | 19.675 | 19.691 | 19.685 | 19.753 | 19.750 | 19.809 | 19.823 |
| 0.825 | 17.410 | 17.410 | 17.356 | 17.356 | 17.394 | 17.393 | 17.470 | 17.468 | 17.631 | 17.637 | 17.832 | 17.831 | 17.998 | 18.012 |
| 0.850 | 14.986 | 14.986 | 15.030 | 15.028 | 15.165 | 15.161 | 15.358 | 15.354 | 15.670 | 15.687 | 16.009 | 16.011 | 16.280 | 16.294 |
| 0.875 | 12.617 | 12.617 | 12.793 | 12.791 | 13.041 | 13.036 | 13.358 | 13.355 | 13.822 | 13.845 | 14.293 | 14.299 | 14.663 | 14.675 |
| 0.900 | 10.348 | 10.347 | 10.683 | 10.681 | 11.050 | 11.046 | 11.490 | 11.491 | 12.098 | 12.121 | 12.690 | 12.700 | 13.151 | 13.161 |
| 0.925 | 8.231 | 8.231 | 8.736 | 8.736 | 9.217 | 9.217 | 9.770 | 9.775 | 10.506 | 10.524 | 11.205 | 11.217 | 11.744 | 11.753 |
| 0.950 | 6.326 | 6.326 | 6.986 | 6.988 | 7.564 | 7.568 | 8.211 | 8.220 | 9.053 | 9.062 | 9.839 | 9.853 | 10.445 | 10.453 |
| 0.975 | 4.681 | 4.682 | 5.456 | 5.460 | 6.104 | 6.110 | 6.820 | 6.830 | 7.740 | 7.741 | 8.594 | 8.606 | 9.252 | 9.261 |
| 1.000 | 3.327 | 3.328 | 4.159 | 4.162 | 4.842 | 4.850 | 5.598 | 5.607 | 6.566 | 6.561 | 7.467 | 7.475 | 8.163 | 8.174 |
| 1.025 | 2.267 | 2.267 | 3.092 | 3.093 | 3.776 | 3.781 | 4.540 | 4.546 | 5.529 | 5.519 | 6.454 | 6.458 | 7.175 | 7.187 |
| 1.050 | 1.479 | 1.479 | 2.242 | 2.241 | 2.894 | 2.896 | 3.640 | 3.640 | 4.621 | 4.611 | 5.551 | 5.549 | 6.283 | 6.296 |
| 1.075 | 0.924 | 0.924 | 1.585 | 1.582 | 2.181 | 2.179 | 2.885 | 2.880 | 3.834 | 3.826 | 4.752 | 4.743 | 5.483 | 5.495 |
| 1.100 | 0.552 | 0.552 | 1.094 | 1.090 | 1.616 | 1.610 | 2.261 | 2.252 | 3.159 | 3.154 | 4.048 | 4.036 | 4.768 | 4.779 |
| 1.125 | 0.316 | 0.316 | 0.737 | 0.733 | 1.179 | 1.170 | 1.753 | 1.742 | 2.585 | 2.584 | 3.434 | 3.419 | 4.133 | 4.142 |
| 1.150 | 0.174 | 0.174 | 0.485 | 0.483 | 0.846 | 0.838 | 1.344 | 1.335 | 2.102 | 2.104 | 2.900 | 2.885 | 3.571 | 3.579 |
| 1.175 | 0.092 | 0.092 | 0.312 | 0.312 | 0.598 | 0.593 | 1.021 | 1.015 | 1.698 | 1.703 | 2.439 | 2.427 | 3.077 | 3.083 |
| 1.200 | 0.046 | 0.046 | 0.196 | 0.199 | 0.416 | 0.416 | 0.768 | 0.767 | 1.364 | 1.372 | 2.043 | 2.038 | 2.643 | 2.648 |
| 1.225 | 0.023 | 0.023 | 0.121 | 0.126 | 0.286 | 0.291 | 0.572 | 0.578 | 1.089 | 1.099 | 1.705 | 1.707 | 2.264 | 2.269 |
| 1.250 | 0.011 | 0.011 | 0.073 | 0.079 | 0.194 | 0.203 | 0.422 | 0.434 | 0.864 | 0.878 | 1.418 | 1.429 | 1.935 | 1.941 |
| 1.275 | 0.005 | 0.005 | 0.043 | 0.050 | 0.129 | 0.142 | 0.309 | 0.326 | 0.683 | 0.699 | 1.175 | 1.196 | 1.649 | 1.657 |
| 1.300 | 0.002 | 0.002 | 0.025 | 0.032 | 0.085 | 0.100 | 0.224 | 0.246 | 0.536 | 0.555 | 0.970 | 1.002 | 1.402 | 1.414 |
| 1.325 | 0.001 | 0.001 | 0.014 | 0.020 | 0.056 | 0.071 | 0.161 | 0.185 | 0.419 | 0.440 | 0.799 | 0.839 | 1.190 | 1.205 |
| 1.350 | 0.000 | 0.000 | 0.008 | 0.013 | 0.036 | 0.050 | 0.115 | 0.140 | 0.326 | 0.350 | 0.656 | 0.704 | 1.007 | 1.027 |
| 1.375 | 0.000 | 0.000 | 0.004 | 0.009 | 0.023 | 0.036 | 0.082 | 0.107 | 0.253 | 0.278 | 0.537 | 0.592 | 0.851 | 0.876 |
| 1.400 | 0.000 | 0.000 | 0.002 | 0.006 | 0.014 | 0.026 | 0.058 | 0.081 | 0.195 | 0.221 | 0.438 | 0.498 | 0.718 | 0.747 |
| 1.425 | 0.000 | 0.000 | 0.001 | 0.004 | 0.009 | 0.019 | 0.040 | 0.062 | 0.150 | 0.176 | 0.357 | 0.420 | 0.604 | 0.638 |
| 1.450 | 0.000 | 0.000 | 0.001 | 0.003 | 0.006 | 0.014 | 0.028 | 0.048 | 0.115 | 0.141 | 0.290 | 0.356 | 0.508 | 0.546 |
| 1.475 | 0.000 | 0.000 | 0.000 | 0.002 | 0.003 | 0.010 | 0.019 | 0.037 | 0.088 | 0.114 | 0.235 | 0.301 | 0.426 | 0.468 |
| 1.500 | 0.000 | 0.000 | 0.000 | 0.001 | 0.002 | 0.008 | 0.013 | 0.029 | 0.067 | 0.092 | 0.190 | 0.256 | 0.357 | 0.403 |

strictly diagonally dominant.[6] The $n \times n$ matrix $\boldsymbol{Q}^\top$ is tridiagonal, and its $i = 2, \ldots, N-1$ rows are $\frac{1}{\Delta x_{i-1}}$, $-\frac{1}{\Delta x_{i-1}} - \frac{1}{\Delta x_i}$, $\frac{1}{\Delta x_i}$, and the other elements of $\boldsymbol{R}, \boldsymbol{Q}$ are zeros.

[6]As clear from the expression, in this case, $\boldsymbol{R}$ is strictly positive definite and the matrix equation (24) has a unique solution $\boldsymbol{c}$.

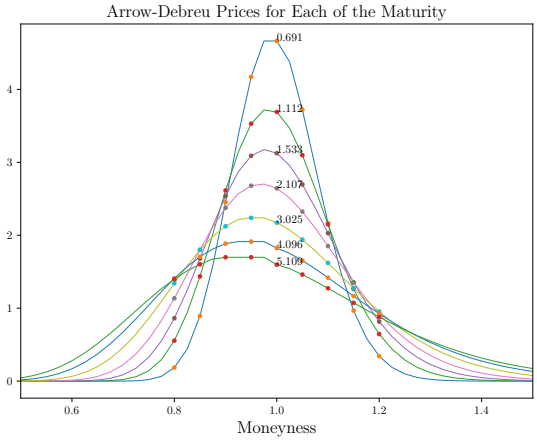



FIGURE 2. Calibrated Arrow-Debreu Prices S&P500, Date: 07/11/2019

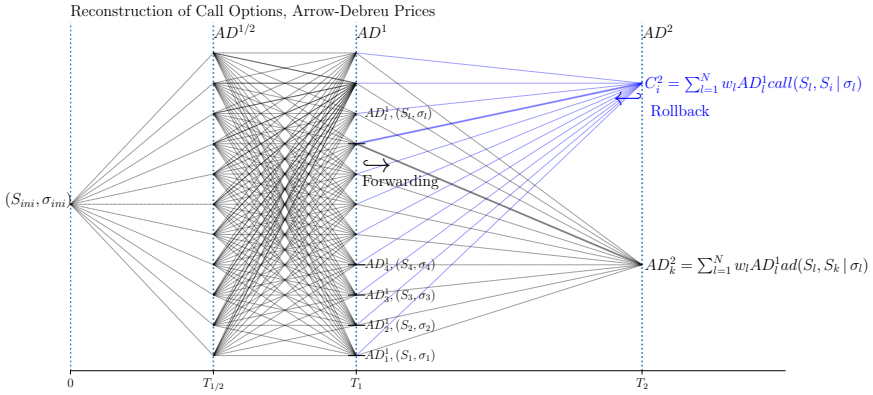



FIGURE 3. Reconstruction of Call Option Price

The smoothing spline method is a spline approximation that has a penalty term and allows some observational errors. The general form of the penalty is to minimize the summation of the square of second derivatives over the whole range and the spline function by a natural cubic spline:

$$\boldsymbol{J}(f) = p\sum_{i=1}^{n}\left[w_i(y_i - C(x_i))\right]^2 + (1-p)\int_{x_1}^{x_N}\left(C''(x)\right)^2 dx, \tag{25}$$

where $y_i = g(x_i) + \epsilon_i$ for a smooth function $g$ at data $x_1, \ldots, x_N$, and $\frac{1}{w_i}$ is the standard deviation in $y_i$ for estimation errors $\epsilon_i$, and $\lambda = \frac{1-p}{p}$ is called

the roughness of the penalty. Since $C''(x)$ is a linear function, the integral is calculated using Simpson's rule,

$$\int_{x_1}^{x_N} \left(C''(x)\right)^2 dx = \frac{4}{3}\sum_{i=1}^{n-1} \Delta x_i \left(c_i^2 + c_i c_{i+1} + c_{i+1}^2\right) = \frac{2}{3}\boldsymbol{c}^\top \boldsymbol{R}\boldsymbol{c}. \tag{26}$$

Now, we construct the quadratic programming problem to find $\boldsymbol{a}$. From (24), we can express $\boldsymbol{c}$ using $\boldsymbol{a}$,

$$\boldsymbol{c} = 3\left(\boldsymbol{R}\backslash\boldsymbol{Q}^\top\right)\boldsymbol{a} =: 3\boldsymbol{S}\boldsymbol{a}, \tag{27}$$

where $\boldsymbol{S} = \boldsymbol{R}\backslash\boldsymbol{Q}^\top$ and the backslash operator "\" implies solving the matrix equation $\boldsymbol{R}\boldsymbol{S} = \boldsymbol{Q}^\top$ without using the inverse and ignoring the first and last rows of the $n \times n$ matrix $\boldsymbol{S}$, which are all 0. The integral part is $\frac{2}{3}\boldsymbol{c}^\top\boldsymbol{R}\boldsymbol{c} = 6\boldsymbol{S}^\top\boldsymbol{R}\boldsymbol{S} = 6\boldsymbol{M}$, where $\boldsymbol{M} = \boldsymbol{S}^\top\boldsymbol{R}\boldsymbol{S}$. Then, (25) can be transformed to the matrix form:

$$\boldsymbol{J}(f)/p = \boldsymbol{a}^\top\left(\boldsymbol{W} + 6\lambda\boldsymbol{M}\right)\boldsymbol{a} - 2\boldsymbol{y}^\top\boldsymbol{W}\boldsymbol{a} + \boldsymbol{y}^\top\boldsymbol{W}\boldsymbol{y}, \tag{28}$$

where $\boldsymbol{W}$ is the diagonal matrix with elements $w_1, \ldots, w_N$. Since the last term is constant, the first two terms are the objective functions of the minimization problem. Because at knots $x_1, \ldots, x_{N-1}$, the value, the first and second derivatives of $C(x)$, must be the same by the left and right segments,

$$\begin{aligned} b_i &= \frac{1}{\Delta x_i}(a_{i+1} - a_i) - \frac{\Delta x_i}{3}(2c_i + c_{i+1}), \\ d_i &= \frac{1}{3\Delta x_i}(c_{i+1} - c_i),\ i = 1, \ldots, N-1. \end{aligned} \tag{29}$$

At $x_N$,

$$\begin{aligned} b_N = C'(x_N) &= \lim_{x \to x_N -} C'(x_N) \\ &= \frac{1}{\Delta x_{N-1}}(a_N - a_{N-1}) + \frac{\Delta x_{N-1}}{3}(2c_N + c_{N-1}), \end{aligned}$$

and additional assumption of $d_N = 0$. In a matrix form, we have

$$\boldsymbol{c} = 3\boldsymbol{S}\boldsymbol{a},\ \boldsymbol{b} = \boldsymbol{B}\boldsymbol{a},\ \boldsymbol{d} = \boldsymbol{D}\boldsymbol{a}, \tag{30}$$

where $\boldsymbol{B} = \boldsymbol{B}_1 - \boldsymbol{B}_2\boldsymbol{S}$, $\boldsymbol{D} = \boldsymbol{B}_1\boldsymbol{S}$. The matrices $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ are the $n \times n$ matrixes whose $(i,i)$, $(i,i+1)$-th elements are $(-\frac{1}{\Delta x_i}, \frac{1}{\Delta x_i})$, $(2\Delta x_i, \Delta x_i)$, $i = 1, \ldots, N-1$, and 0 elsewhere.

Discounted Density Function Condition (12b). Since the second-order differentiation of the cubic spline function $C$ is a piecewise linear function, by the trapezoidal quadrature we have

$$\int_{x_1}^{x_N} C''(x)dx = 2\boldsymbol{F}^\top\boldsymbol{c} = 6\boldsymbol{F}^\top\boldsymbol{S}\boldsymbol{a} = df, \tag{31}$$

where $\boldsymbol{F} = \frac{1}{2}\big(\Delta x_1, (\Delta x_1 + \Delta x_2), \ldots, (\Delta x_{N-2} + \Delta x_{N-1}), \Delta x_{N-1}\big)^\top$ and $C''(x_i) = 2c_i$, $i = 1, \ldots, N$.

Forward Risk Neutrality Condition. The risk-neutral assumption of model (1) is simply

$$\mathbb{E}[\mathbb{X}_T] = \int_{x_1}^{x_N} xC''(x)dx = df\, R = dq, \tag{32}$$

where $\mathbb{X}_T$ is the random variable with probability density function $\boldsymbol{P}(x,T)$ for given maturity $T > 0$ and forwarding factor $R = R(0,T)$ is defined as

$$R(T_1,T_2) = dq(T_1,T_2)/df(T_1,T_2) \tag{33}$$

for brevity, where $df(T) = df(0,T)$, $dq(T) = dq(0,T)$, $R(T) = R(0,T)$. Since

$$\begin{aligned}\int_{x_i}^{x_{i+1}} xC''(x)dx &= \int_{x_i}^{x_{i+1}} x(2c_i + 6d_i(x - x_i))dx \\ &= (x_{i+1}^2 - x_i^2)c_i + (2x_{i+1}^3 + x_i^3 - 3x_i x_{i+1}^2)d_i\end{aligned}$$

and $\boldsymbol{c} = 3\boldsymbol{S}\boldsymbol{a}, \boldsymbol{d} = \boldsymbol{D}\boldsymbol{a}$, the equation (32) becomes

$$\begin{aligned}\int_{x_1}^{x_N} xC''(x)dx &= (x_2^2 - x_1^2, \ldots, x_N^2 - x_{N-1}^2, 0) \cdot \boldsymbol{c} \\ &\quad + (2x_2^3 + x_1^3 - 3x_1x_2^2, \ldots, 2x_N^3 + x_{N-1}^3 - 3x_{N-1}x_N^2,\ 0) \cdot \boldsymbol{d} \\ &= \boldsymbol{G}^\top \boldsymbol{a} = df\, R = dq, \end{aligned} \tag{34}$$

where $\boldsymbol{G}$ is an $n$ dimensional vector.

Optional Constraint for the Volatility Smile. In a market with evident volatility smile, the risk-neutral implied distribution shows a monotone increase and monotone decrease centering on the forward price. However, depending on the optimization method and specific solvers, the computed implied distribution can have much whipsaws, especially for quadratic optimization. These whipsaws can be reduced by allowing negligible numerical errors. In this case, artificial monotone conditions can be imposed. Considering this environmental situation of practitioners, the following are described as optional conditions. Since

$$\int_{x_1}^{x_N} (x - R)C''(x)dx = \int_{x_1}^{R} (x - R)C''(x)dx + \int_{R}^{x_N} (x - R)C''(x)dx = 0$$

and $C'' > 0$, constraint (32) has the same role that discounted implied distribution $C''$ has in the area left and right of $R$. From this point of view, we can add artificial monotonic conditions to $C''$, where $C''$ increases monotonically before $R$ and decreases monotonically after $R$. This does not affect the accuracy of the recovering power of $C''$ when a volatility smile is present. This condition dramatically reduces the whipsaw and peak of $C''$ at the knots.[7]

[7]This condition, which has smoothing effects on the distribution, is very important for a feasible calculation of local volatilities from the call option prices or implied distribution $C''$ itself. More specifically, in [2] use implied distribution as the building block in their FDM and in [3] use call option prices as their finite difference.

We describe this as cumulative distribution. Consider the first-order derivatives of $C$ at $x_i$,

$$C'(x_i) - C'(0) = \int_0^{x_i} C''(x)dx = df \cdot \mathrm{cdf}(x_i),\ i = 1, \ldots, N. \tag{35}$$

Since $C'(0) = -df$ for this relation, the cumulative distribution function satisfies $\mathrm{cdf}(x) = C'(x)/df + 1$. The inflection point of $\mathrm{cdf}(x)$ is the point of sign changes for the second derivatives of $C'(x)/df + 1$, which equals $C'''(x)$.

$$\begin{aligned} &\frac{\partial^3 C}{\partial x^3}\Big|_{x_i} \geq 0,\ i = 1, \ldots, l, \\ &\frac{\partial^3 C}{\partial x^3}\Big|_{x_i} < 0,\ i = l+1, \ldots, N, \end{aligned} \tag{36}$$

where $|x_l - R| < |x_i - R|$, $l \neq i, i = 1, \ldots, N$. For maturity $T$, $df = df(T), R = R(T)$.

Linear Programming for Implied Distributions. We can obtain a series of cubic spline coefficients that is used to recover the call option price curves as well as implied distributions. The call option price is represented as an integration of the product of its payoff and the Arrow-Debreu prices (discounted distribution function), both of which are piecewise linear functions, and calculated using Simpson's rule for polynomials of order 2. Based on this, we construct a more direct optimization problem for distribution $C''$. This is the final step to find the implied distributions. This gives a call option price as a summation of the product of its payoff and discounted probability distribution at knot points, both of which are linear. We obtain series of probability density functions each of which recovers call option prices for the given maturity and satisfies no arbitrage constraints under $\ell^1$ minimization scheme.

**Theorem A.1** ($\ell^1$-Minimization for Arrow-Debreu Prices)**.** *There exists a linear programming problem for the construction of the Arrow-Debreu prices $C_j''(\boldsymbol{x})$ for each $t_j, j = 1/2, 1, \ldots, m$, based on the cubic spline approximation of the call option prices.*

*Proof.* Step 1: Call option price vector as a function of $C_j''(\boldsymbol{x})$:

$$\begin{aligned} C(x_k, t_j) &= \int_{x_1}^{x_N} \max(x - x_k, 0) C''(x)dx \\ &= \sum_{i=k}^{n-1} \int_{x_i}^{x_{i+1}} (x - x_k) l(x \,|\, 2c_i, 2c_{i+1})dx \to P_i(x) := (x - x_k) l(x \,|\, 2c_i, 2c_{i+1}) \\ &= \sum_{i=k}^{n-1} \frac{x_{i+1} - x_i}{6} \left[ P_i(x_i) + 4P_i\left(\frac{x_{i+1} + x_i}{2}\right) + P_i(x_{i+1}) \right] \to \text{Simpson's rule} \\ &= \sum_{i=\max(k,2)}^{n-1} \frac{1}{3} \underline{\left[ -x_{i-1}^2 + x_{i+1}^2 - x_{i-1}x_i + x_i x_{i+1} + x_{i-1}x_k - x_{i+1}x_k \right]} c_i, \end{aligned} \tag{37}$$

where $k = 1, \ldots, N-1$, and $l(x \,|\, 2c_i, 2c_{i+1})$ is a linear segment connecting $(x_i, 2c_i)$ to $(x_{i+1}, 2c_{i+1})$. In a matrix form, we have $\frac{1}{3}\boldsymbol{Cc} = \boldsymbol{CSa} = \boldsymbol{call}$, where $\boldsymbol{S}$ in (27), $\boldsymbol{call}$ is the call option price vector with $k-$th element $call(x_k, t_j)$ and $\boldsymbol{C}$ is the $(n-1) \times n$ matrix whose $(k, i)$ element is the underlined term in (37).

Step 2: $\ell^1$-minimization problem for $C_j''(\boldsymbol{x})$.

Here, the objective function is

$$\min_{\boldsymbol{a}} \sum_{i=1}^{n} \left| w_i (y_i - (\boldsymbol{CSa})_i) \right|,$$

where $(\boldsymbol{CSa})_i$ indicates the $i$-th element of vector $\boldsymbol{CSa}$. By introducing the new slack variables $z_i = |w_i(y_i - (\boldsymbol{CSa})_i)|$, $i = 1, \ldots, N-1$, we can change this into a linear programming problem (38), without loss of generality, assuming that $w_i = 1$, we have

(38)

$$\begin{aligned}
&\min_{\boldsymbol{a}} \left( \sum_{i=1}^{n-1} z_i = [\boldsymbol{0}, \boldsymbol{1}] \cdot \begin{bmatrix} \boldsymbol{a} \\ \boldsymbol{z} \end{bmatrix} \right), \\
&\text{s.t.} \quad -z_i \le 0, \\
&\qquad -(\boldsymbol{CSa})_i - z_i \le -y_i, \\
&\qquad +(\boldsymbol{CSa})_i - z_i \le +y_i, \ i = 1, \ldots, N-1, \\
&\qquad C_j(x_i) \ge 0, \\
&\qquad \left. \frac{\partial C_j}{\partial x} \right|_{x_i} \le 0 \longrightarrow (11a), \\
&\qquad \left. \frac{\partial^2 C_j}{\partial x^2} \right|_{x_i} \ge 0 \ i = 1, \ldots, N, \longrightarrow (11b), \\
&\qquad \int_{x_1}^{x_N} C_j''(x) dx = df \longrightarrow (31), \\
&\qquad \int_{x_1}^{x_N} x C_j''(x) dx = dq \longrightarrow (32), \\
&\qquad \lambda_j = 0 \ \text{ for } t_j < t_{\text{short}}, \text{ or using optional constraint} \\
&\qquad \textbf{optional constraint for volatility smile:} \\
&\qquad \left[ \left. \frac{\partial^3 C_j}{\partial x^3} \right|_{x_i} \ge 0, \ i = 1, \ldots, l, \ \left. \frac{\partial^3 C_j}{\partial x^3} \right|_{x_i} \le 0, \ i = l+1, \ldots, N \right] \longrightarrow (36),
\end{aligned}$$

where the time 0 call option price is $C_0(x) = \max(1 - x, 0)$ and index $l = \underset{i}{\operatorname{argmin}} |x_i - R_j|, R_j = R(0, t_j)$. Since $\frac{\partial^3 C_j}{\partial x^3} = 6d_i$ at $x = x_i, i = 1, \ldots, N$ for each $t_j$, the last two condition can be expressed as $\boldsymbol{D}_1 \boldsymbol{a} \ge \boldsymbol{0}$ and $\boldsymbol{D}_2 \boldsymbol{a} < \boldsymbol{0}$, where $\boldsymbol{D}_1$ is a sub matrix of $\boldsymbol{D}$ consisting of rows 1 to $l$, and $\boldsymbol{D}_2$ is made up of the remaining rows. □

## References


[1] L. B. G. Andersen, J. Andreasen, and D. Eliezer, *Static replication of barrier options: some general results*, J. Comput. Finance **5** (2002), 1–25.

[2] L. B. G. Andersen and R. Brotherton-Ratcliffe, *The equity option volatility smile: an implicit finite-difference approach*, J. Comput. Finance **1** (1998), no. 2, 5–37.

[3] J. Andreasen and B. N. Huge, *Volatility interpolation*, Available at SSRN 1694972, 2010.

[4] K. J. Arrow and G. Debreu, *Existence of an equilibrium for a competitive economy*, Econometrica **22** (1954), 265–290. https://doi.org/10.2307/1907353

[5] D. T. Breeden and R. H. Litzenberger, *Prices of state-contingent claims implicit in option prices*, The Journal of Business **51** (1978), no. 4, 621–651.

[6] P. Carr and L. Wu, *Static hedging of standard options*, J. Financial Econometrics **12** (2014), 1–44.

[7] S. L. Chung, P. T. Shih, and W. C. Tsai, *A modified static hedging method for continuous barrier options*, The Journal of Futures Markets **30** (2010), no. 12, 1150–1166.

[8] C. de Boor, *A practical guide to splines*, Applied Mathematical Sciences, 27, Springer-Verlag, New York, 1978.

[9] B. Dupire, *Pricing with a smile*, Risk **7** (1994), no. 1, 18–20.

[10] J. Gatheral, *The volatility surface: a practitioner's guide*, Volume 357, John Wiley & Sons, 2011.

[11] V. Linetsky, *The path integral approach to financial modeling and options pricing*, Computational Economics **11** (1998), 129–163.

[12] J. Nocedal and S. J. Wright, *Numerical optimization*, second edition, Springer Series in Operations Research and Financial Engineering, Springer, New York, 2006.

[13] B. Øksendal, *Stochastic differential equations*, sixth edition, Universitext, Springer-Verlag, Berlin, 2003. https://doi.org/10.1007/978-3-642-14394-6

[14] M. J. D. Powell, *A hybrid method for nonlinear equations*, in Numerical methods for nonlinear algebraic equations (Proc. Conf., Univ. Essex, Colchester, 1969), 87–114, Gordon and Breach, London, 1970.



Hyeong-Ohk Bae
Department of Financial Engineering
Ajou University
Suwon 16499, Korea
*Email address*: hobae@ajou.ac.kr

Hyuncheul Lim
Department of Mathematics
Chonnam National University
Gwangju 61186, Korea
*Email address*: limhc@jnu.ac.kr