Local volatility calibration with optimal control in a Lagrangian framework

Love Lindholm

Abstract

We develop a Lagrangian based method for solving the calibration problem of identifying a local volatility function that makes the solution to Dupire’s pricing equation match observed market quotes on options. Our method uses well established techniques from the field of inverse problems, but differs from other published methods in that our formulation makes it possible to use a Newton method with the analytic Jacobian of the system corresponding to first order optimality conditions of our problem. We give a numerical example with market data of options on the Euro Stoxx 50 index and find that our method is efficient and robust in its ability to identify a volatility function that fit observed data.

1 Introduction

Since the so called local volatility model was introduced in 1994 (Dupire [8], Derman and Kani [6]) it has become one of the most extensively used models in derivatives pricing across all asset classes. In the case of an equity stock or index $S$, the price dynamics in the local volatility model under the risk neutral measure are given as

$$dS_t = (r_t - q_t)S_t dt + \sigma(t, S_t)S_t dW_t,$$

where $S_0 > 0$ is given, $t$ is the time, $W_t$ is a Brownian motion, $r_t$ is the risk free interest rate and $q_t$ is a continuous dividend yield. The squared local volatility $\sigma^2$ thus gives the instantaneous variance of the logarithm of $S$ as a deterministic function of the time $t$ and the spot value $S_t$. Under the dynamics (1), the prices of call options $c(t, x)$ written on $S$, of time to maturity $t$ and strike $x$, can be expressed (see e.g. Pironneau [18]) as the solution to a parabolic partial differential equation known as Dupire’s equation,

$$\frac{\partial c(t, x)}{\partial t} = \frac{1}{2} \sigma^2(t, x)x^2 \frac{\partial^2 c(t, x)}{\partial x^2} - q(t, x)\frac{\partial c(t, x)}{\partial x} - (r_t - q_t)x \frac{\partial c(t, x)}{\partial x}, \quad (t, x) \in \mathbb{R}_+ \times \mathbb{R}_+,$$

$$c(0, x) = (S_0 - x)_+, \quad c(t, 0) = S_0 e^{-\int_0^t q_u du}, \quad c(t, \infty) = 0.$$}

The problem of “calibrating” the local volatility model to fit market data consists in choosing the function $\sigma$ so that the solution to (2) is as close as possible to the option prices observed in the market. Suppose that $\bar{u} \in \mathbb{R}^n$ represents a vector of observed market quotes on put- and call options. Since Dupire’s equation gives us only call option prices, we need to use put-call parity,

$$p(t, x) = c(t) - b(t)(fw(t) - x),$$

1
to obtain the put option prices $p$ corresponding to the solution $c$ to (2) given the zero-coupon bond $b$ and forward price $f w$ defined by

$$b(t) = e^{-\int_0^t r_s \, ds}, \quad f w(t) = S_0 e^{\int_0^t (r_s - q_s) \, ds}.$$ (4)

Let $(t_k, x_k)$ be the maturity and strike of the element $\bar{u}_k$, $k = 1, \ldots, n$, and suppose, for a given $\sigma$, that $u(\sigma) \in \mathbb{R}^n$ is a vector such that the element $u_k = c(t_k, x_k)$ if $\bar{u}_k$ is a quoted call option price and $u_k = p(t_k, x_k)$ if $\bar{u}_k$ represents a put option price. The problem of finding the volatility function $\sigma$ that makes the model prices from Dupire’s equation match observed data can then be stated as

$$\min_{\sigma \in \Sigma} \| u(\sigma) - \bar{u} \|^2$$ (5)

where $\Sigma$ is an appropriate function space and $\| \cdot \|$ some norm in $\mathbb{R}^n$. The minimization (5) is an example of a parameter estimation problem for partial differential equations, and as such it belongs to a class of inverse problems that in general are ill-posed. This particular parameter estimation problem - estimating a local volatility function to make the solution to Dupire’s equation fit observed option market data - has been the topic of several studies, many of which employ tools from the theory of inverse problems. We will mention some of these previous studies, without pretending to be exhaustive. Lagnado and Osher in [14] add a Tikhonov regularization to the objective function and solve the problem by a gradient method in which a partial differential equation for the gradient is obtained and solved numerically. Jackson et al [13] exploit the ideas of Lagnado and Osher to calculate a local volatility defined in a space $\Sigma$ of spline functions and employ a gradient based quasi-Newton optimization technique. Coleman et al. [4] also let the space $\Sigma$ consist of spline functions defined on a grid but use a Jacobian based optimization algorithm in which the Jacobian is numerically evaluated. A similar approach is used by Glover and Ali in [11] but now with $\Sigma$ consisting of a space of radial basis functions rather than splines. Achdou and Pironneau [1] regularize the objective function by adding penalties on the derivatives of the squared volatility $\sigma^2$ and find an equation for the gradient of the objective function which is used to solve the problem on a successively refined grid with a finite element method. Avellaneda et al. [3] do not use an objective function based on a squared norm as in (5), but pose the calibration of a local volatility to market data as the problem of minimizing a certain entropy of the volatility under the constraint that the model prices match market prices. This approach leads to a Lagrange-formulation and a resulting minimization problem in the Lagrangian multiplier that is solved with a gradient method. Andreasen and Huge [2] develop a fast algorithm for solving (5) with a fully implicit finite difference scheme on a coarse grid (corresponding to the set of observable maturities and strikes) and use the result to interpolate prices in time between the maturities with observable data. Lipton and Sepp [16] use a transform method to find a semi-analytical solution to (2) for the special case of a volatility $\sigma$ that is piecewise constant in $(t, x)$ and use that solution to optimize $\sigma$ in a “bootstrapping manner” to match data for one maturity at the time.

The approach used in this paper has elements in common with many of the above mentioned previous studies. We recognize the need for a regularization of the objective function in (5) and will use both a standard Tikhonov penalty (as in e.g. [9]) and a penalty on the second derivative of the squared volatility (as in e.g. [1]). We will also use a “continuation principle” and iteratively solve a sequence of successively harder problems by reducing the size of the regularization penalties from on round of solving to the next. See [9] and references therein.
for an analysis of this technique. We optimize for a volatility one maturity at the time, thus splitting the problem into a sequence of smaller problems, as is done in, e.g., [16]. This is mostly made to speed up calculations, since solving the sequence of smaller problems is faster than solving the full problem. By splitting the problem into subproblems, we formally lose the possibility of finding an optimal solution to the global optimization problem. This is not an issue of great concern, though, since as we will see, we will still be able to fit market data on options to a satisfactory level. Like many authors (e.g. [1], [4]), we recognize the need to have fewer degrees of freedom for the volatility than we use in the discretization of the Dupire equation (2) itself. We also, like in [16], define our volatility to be piecewise constant in time and space. In our case, this means that the value of the volatility at a given node in the grid will only have a local effect, thus making the resulting system we solve sparser, with fewer connections between the variables than if some basis functions with larger support (as, e.g., the radial basis functions used in [11]) were to be used in the volatility definition. Something we do believe is strongly beneficial, but that does not seem to be done in other studies, is to set up the system which yields necessary optimality conditions for our optimization in a fashion which allows for a natural use of Newton based optimization algorithms. The cited studies use either gradient based optimization, quasi Newton methods or Newton methods in which the Jacobian is calculated numerically. (In [15][Paper I], we provide a setup for local volatility optimization using a Hamilton-Jacobi-Bellman formulation which also uses a Newton method in the solving phase.) The main contribution of this report is therefore to illustrate how elements from well established theory of inverse problems and optimal control can be brought together with a Newton based optimization to form a robust and efficient algorithm for solving the local volatility calibration problem.

The rest of this paper gives a formulation of the volatility calibration problem with a Lagrangian multiplier in Section 2 and presents some details regarding the implementation in Section 3. We show results of a calibration to option market data on the Euros Stoxx 50 equity index in Section 4. Section 5 concludes with a short discussion.

2 Local volatility as optimal control

Market listed options have a prescribed set of possible maturities. In our optimization, we will sequentially search for a local volatility defined on the time interval between one observable maturity and the next. This is a common practice [16] - feasible because of the forward direction in time the Dupire equation (2) - which allows to divide the optimization into a set of smaller subproblems at the cost of formally losing the possibility of finding the best global solution. In our numerical schemes, we will need a discretization of the partial differential equation (2). We will also discretize the volatility function $\sigma$, but on a different grid in space and time than what we use for the discretization of the Dupire equation (2) itself. Before we define our optimization problem, we need some notation related to these discretizations. Set $\tau^0 = 0$ and let

$$\mathcal{T} := \{\tau^1, \ldots, \tau^{m_\tau} : \tau^{i-1} < \tau^i, i = 1, \ldots, m_\tau\}$$

(6)
be the maturities for which we can observe market prices on options. We can split the equation (2) into a sequence of partial differential equation for \( t = 1, \ldots, m_r \) according to

\[
\partial_t c^\ell(t, x) = \frac{1}{2} \sigma^2(t, x) x^2 \partial_{xx} c^\ell(t, x) - q(t) x \partial_x c^\ell(t, x), \quad (t, x) \in (\tau^{\ell-1}, \tau^{\ell}] \times \mathbb{R}_+ \\
c^\ell(\tau^{\ell-1}, x) = c^{\ell-1}(\tau^{\ell-1}, x), \quad x \in \mathbb{R}_+ \\
c^\ell(t, 0) = S_0 e^{-\int_0^t q(s) ds}, \quad c^\ell(t, \infty) = 0, \quad t \in (\tau^{\ell-1}, \tau^\ell],
\]

(7)

where we let \( c^0(0, x) = (S_0 - x)_+ \) where \( S_0 \) was the given spot price. We then have that

\[
c(t, x) = c^\ell(t, x), \quad t \in (\tau^{\ell-1}, \tau^\ell], \quad \ell = 1, \ldots, m_r,
\]

(8)

where \( c \) solves (2) and \( c^\ell \) solves (7). We will search for the local volatility function \( \sigma \) on each interval \((\tau^{\ell-1}, \tau^\ell]\), starting at \( \ell = 1 \) and proceeding from \( \ell \) to \( \ell + 1 \) until we reach \( \ell = m_r \). It will thus be natural to work with the sequence of equations defined in (7) rather than the equation (2) defined on the full interval \((0, \tau^{m_r}]\). For ease of notation we will, however, sometimes drop the superscript \( \ell \) and write \( c \) rather than \( c^\ell \), bearing in mind that we refer to a function on the interval \((\tau^{\ell-1}, \tau^\ell]\) as in (7).

We now discretize the partial differential equation (7) in the space variable to obtain an ordinary differential equation. Below we will formulate an optimal control problem based on this ordinary differential equation, before discretizing the time dimension to allow for a numerical solution of the problem. Let us thus introduce a grid for the space dimension \( x \in \mathbb{R}_+ \) in (7) according to

\[
\chi := \{x_0, \ldots, x_{n+1} : x_0 = 0, \quad x_{j-1} < x_j, \quad j = 1, \ldots, n_x + 1\},
\]

(9)

and approximate the differential operators in (7) by finite difference operators given by

\[
\partial_x c(t, x_j) \approx D^1_j c_j := \frac{1}{\mu^-_j + \mu^+_j} \left[ \frac{\mu^-_j}{\mu^+_j} (c_{j+1} - c_j) + \frac{\mu^+_j}{\mu^-_j} (c_j - c_{j-1}) \right] \\
\partial_{xx} c(t, x_j) \approx D^2_j c_j := \frac{2}{\mu^-_j + \mu^+_j} \left[ \frac{c_{j-1} - c_{j+1}}{\mu^-_j + \mu^+_j} - \left( \frac{1}{\mu^-_j} + \frac{1}{\mu^+_j} \right) c_j \right],
\]

(10)

where \( c_j \) is shorthand for \( c(t, x_j) \) and \( \mu^-_j = x_j - x_{j-1}, \quad \mu^+_j = x_{j+1} - x_j \). We make this choice of operators since we will use a non-uniform grid in order to refine the mesh at strikes \( x \) close to \( S_0 \). A Taylor expansion applied to a smooth function yields that the operator \( D^1_j \) is \( O(\mu^-_j \mu^+_j) \) and \( D^2_j \) is \( O(\max(|\mu^-_j - \mu^+_j|, \mu^-_j \mu^+_j)) \). In other words, \( D^1_j \) is formally second order accurate and also \( D^2_j \) provided \( |\mu^-_j - \mu^+_j| \sim \mu^-_j \mu^+_j \).

With the discretization above, we can approximate the solution \( c(t, x_j) \), of Dupire’s equation (7) on an interval \((\tau^{\ell-1}, \tau^\ell]\) at the grid points \( x_j, \quad 1 \leq j \leq n_x \) by a function \( c(t) = (c_1(t), \ldots, c_{n_x}(t)) : \mathbb{R}_+ \rightarrow \mathbb{R}^{n_x} \), solution to the following ordinary differential equation

\[
c_j'(T) = \frac{1}{2} \sigma^2_j(T) x_j^2 D^2_j c_j(t) - q(t) x_j D^1_j c_j(t), \quad (t, j) \in (\tau^{\ell-1}, \tau^\ell] \times \{1, \ldots, n_x\}
\]

(11)

\[
c_j(\tau^{\ell-1}) = c_{j-1}^{\ell-1}(\tau^{\ell-1}), \quad j \in \{1, \ldots, n_x\}, \quad c_0(t) = S_0 e^{-\int_0^t q(s) ds}, \quad c_{n_x+1}(t) = 0, \quad t \in (\tau^{\ell-1}, \tau^\ell],
\]
where
\[ \sigma_j(t) := \sigma(t, x_j) \] (12)
and where \( c_{t-1} \) denotes the solution to the corresponding equation on the previous interval \((\tau^{t-2}, \tau^{t-1}]\). Now let the vector valued function
\[ f = (f_1, \ldots, f_{n_x}) : \mathbb{R}_+ \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_x} \to \mathbb{R}^{n_x} \] (13)
be defined by the spatial finite differences in (11) above, so that for \( c, \sigma \in \mathbb{R}^n \) we have
\[ f_j(t, c, \sigma) := \frac{1}{2} \sigma_j^2 x_j^2 D_j^2 c_j - q_t c_j - (r_t - q_t) x_j D_j^1 c_j, \quad j = 1, \ldots, n_x, \] (14)
where we use the boundary conditions for \( c_0 \) and \( c_{n_x+1} \) defined in (11) for the difference operators at the boundaries \( j = 1 \) and \( j = n_x \). If we set \( \sigma(t) := (\sigma_1(t), \ldots, \sigma_{n_x}(t)) \), we can now write (11) on vector form according to
\[ c'(t) = f(t, c(t), \sigma(t)), \quad t \in (\tau^{t-1}, \tau^t], \quad c(\tau^{t-1}) = c_{t-1}^{(t-1)}. \] (15)

The market data on options we will try to replicate includes both put- and call options, and since the Dupire equation (2) gives call option prices, we will calculate put option prices via put-call parity. The values \( c_j(t) \) in (11) represent the model price of a call option with maturity \( t \) and strike \( x_j \) for a given local volatility function \( \sigma \). We will denote by \( p_j(t) \) the model put option price corresponding to the same maturity, strike and volatility function. From put-call parity, we then have
\[ p_j(t) = c_j(t) - b(t) (f w(t) - x_j), \quad j = 0, \ldots, n_x + 1 \] (16)
where \( f w(t) = S_0 e^{\int_0^t (r_s - q_s) ds} \) is the forward price of \( S \) and \( b(t) = e^{-\int_0^t r_s ds} \) is the zero coupon bond.

We will restrict our search of a local volatility \( \sigma \) to functions that are piecewise constant over rectangles in a grid in space and time. As mentioned, the discretization used for the volatility will not coincide with the discretization used for the partial differential equations - for which a space grid was defined in (9) - and we will therefore need some additional notation. The volatility will be constant over the times in the grid \( T \) that defines the observed maturities of the market’s option quotes. In the space dimension, we define a grid \( K \)
\[ K_i := \{ \kappa_0, \ldots, \kappa_{n_k} : \kappa_0 = 0, \kappa_{k-1} < \kappa_k, k = 1, \ldots, n'_k \}. \] (17)
and then consider functions \( \sigma^t : (t, x) \in (\tau^{t-1}, \tau^t] \times (0, \infty) \to \mathbb{R}_+ \), \( t = 1, \ldots, m_t \), that are piecewise constant over the rectangles \((\tau^{t-1}, \tau^t] \times (\kappa_{k-1}, \kappa_k) \) for \( k \in \{1, \ldots, n_k\} \). For some vector \( \xi^t \in \mathbb{R}^n_{+} \), we can then write the square of \( \sigma \) from (7) as
\[ \sigma^2(t, x) = \xi_k^t, \quad (t, x) \in (\tau^{t-1}, \tau^t] \times (\kappa_{k-1}, \kappa_k), \quad \forall (t, k) \in \{1, \ldots, m_t\} \times \{1, \ldots, n_k\}. \] (18)
Our discretized volatility function defined in (12) is then componentwise given by
\[ \sigma^2_j(t) = \xi_k^t \] for \( t \in (\tau^{t-1}, \tau^t] \) and \( j \in T'(k), \quad j = 1, \ldots, n_x, \) (19)
where the index set \( T'(k) \) is defined as
\[ T'(k) := \{ j \in \{1, \ldots, n_x\} : x_j \in (\kappa_{k-1}, \kappa_k) \}, \quad k = 1, \ldots, n_k'. \] (20)
Existence of a solution to the stochastic differential equation defining the process $S$ in (1) for this choice of $\sigma$ is guaranteed by [17]. In our optimization problem below, we will search for a vector $\xi^t$ that yields a solution to Dupire’s equation that is as close as possible to observed market data. The actual optimization algorithm we will employ will not use any constraints on the variables optimized for. The vector $\xi^t$, though, is defined to have positive elements. To avoid the potential issue with negative squared volatility values, we let the elements of $\xi^t$ be a positive function of the elements of another vector $\alpha \in \mathbb{R}^{n^t}$. We thus set

$$\xi^t_k(\alpha_k) = \frac{1}{10} \ln \cosh(10\alpha_k), \ k = 1, \ldots, n^t,$$

so that the value of $\xi^t_k$ is a mollified version of the modulus $|\alpha_k|$. We can now define an optimization problem on the form

$$\min_{\alpha \in \mathbb{R}^{n^t}} \ g(c(\tau^t)) + h(\xi^t(\alpha))$$

subject to: $c$ and $\sigma$ satisfy (15) with $\sigma$ given by (19) for $\xi^t$ defined by (21),

for functions $g : \mathbb{R}^n \rightarrow \mathbb{R}_+$ and $h : \mathbb{R}^{n^t} \rightarrow \mathbb{R}_+$. The function $g$ is a terminal cost, which will be defined as a distance between model values $c$ and the data we want to replicate and $h$ will be used to regularize the problem by introducing Tikhonov type penalizations [10] on the vector $\xi^t$. Specifically, we take $h$ to contain a penalty on the values of the vector $\xi^t$ as well as a penalty on the second order finite differences of these values. We let $D_k^j \xi^t_k$, $k = 1, \ldots, n^t$ denote the finite difference operator from (10), but now defined with respect to the grid $K^t$ from (17) rather than $\chi$, so that $\mu_k^- = \kappa_k - \kappa_{k-1}$, $\mu_k^+ = \kappa_{k+1} - \kappa_k$, $k = 2, \ldots, n^t_k - 1$, and set

$$h(\xi^t) = \epsilon_1 \sum_{k=1}^{n^t_k} (\xi^t_k - \bar{\sigma}^2_k)^2 + \epsilon_2 \sum_{k=2}^{n^t_k-1} (D_k^2 \xi^t_k)^2.$$

Here $\bar{\sigma} \in \mathbb{R}^{n^t}$ is a vector of reference volatility values that we will define in Section 4 and $\epsilon_1$, $\epsilon_2$ are positive constants.

We now return to the function $g$ that will be used to measure the distance between model prices and market data. Suppose thus that for a given maturity $\tau^t$, $t \in \{1, \ldots, m^\tau\}$, we can observe market quotes on put and call options at the sets of strikes

$$K^t_p := \left\{ \bar{x}^p_1, \ldots, \bar{x}^p_{n^t_p} \right\}, \ K^t_c := \left\{ \bar{x}^c_1, \ldots, \bar{x}^c_{n^t_c} \right\}$$

respectively. We denote by $\bar{c} := \{\bar{c}_1, \ldots, \bar{c}_{n^t_c}\}$ the quoted prices on call options corresponding to the strikes in $K^t_c$ and by $\bar{p} := \{\bar{p}_1, \ldots, \bar{p}_{n^t_p}\}$ the quoted put option prices for the strikes in $K^t_p$. Market quotes come as pairs of bid- and ask prices, and we will take the prices $\bar{c}_k$ and $\bar{p}_k$ to be the “mid prices”, i.e., the average of the bid- and ask price for a given maturity and strike. Now, the observable market strikes will not necessarily be included in the space grid $\chi$ from (9) at which we obtain model prices. We will therefore approximate the model price at a market strike $\bar{x}^t_k$ as the linear interpolation between the model prices at adjacent grid points according to

$$\bar{c}_k := \frac{x_j - \bar{x}^t_k}{x_j - x_{j-1}} c_{j-1}(\tau^t) + \frac{\bar{x}^t_k - x_{j-1}}{x_j - x_{j-1}} c_j(\tau^t),$$

(25)
for $j$ such that $\tilde{x}_k^c \in (x_{j-1}, x_j]$. We use the put-call parity relation (16) to obtain

$$\hat{p}_k := \frac{x_j - \tilde{x}_k^p}{x_j - x_{j-1}} (c_{j-1}(\tau^i) + b(\tau^i)x_{j-1}) + \frac{\tilde{x}_k^p - x_{j-1}}{x_j - x_{j-1}} (c_j(\tau^i) - b(\tau^i)x_j) - b(\tau^i)fw(\tau^i),$$

for the analogous interpolation of model put prices at $\tilde{x}_k^p$ for for $j$ such that $\tilde{x}_k^p \in (x_{j-1}, x_j]$.

We now have the notation to define the terminal cost function $g$ from (22) as a norm of the difference between the market prices $\bar{c}$ and $\bar{p}$ at $\tau^i$ and the model prices $c$ from (15) $\tau^i$ according to

$$g(c) := \sum_{k=1}^{n^i_p} \omega_k^p (\hat{p}_k - \bar{p}_k)^2 + \sum_{k=1}^{n^i_c} \omega_k^c (\bar{c}_k - c_k)^2.$$

The weights $\omega_k^p$, $\omega_k^c$ are taken to be

$$\omega_k^p := (\hat{p}_k^{bid} - \hat{p}_k^{ask})^{-1}, \quad \omega_k^c := (\bar{c}_k^{ask} - \bar{c}_k^{bid})^{-1},$$

where $\hat{p}_k^{bid}$, $\hat{p}_k^{ask}$ and $\bar{c}_k^{ask}$, $\bar{c}_k^{bid}$ are the bid and ask prices of the observed put- and call options of strike $\tilde{x}_k^p$ and $\tilde{x}_k^c$ respectively. We thus attribute a higher weight to prices given with a small spread, which in a sense corresponds to putting higher weights on data given with less incertitude.

In order to solve our optimization problem (22) we need to discretize the time-dimension of the ordinary differential equation (11). To this end, we define a grid with constant time-step on the interval $[\tau_0, \tau^i]$ according to

$$t^i = \tau_0 + i\Delta t, \quad i = 0, \ldots, m_t, \quad \Delta t = \frac{\tau^i - \tau_0}{m_t}.$$  

We use the notation

$$\sigma_j^t = \sigma_j(t^i),$$

with $t^i$ as in (29) and the $\sigma$ on the right-hand side as in (19). We choose a Crank-Nicolson scheme in time for (15). Define thus a point-wise function $c_j^t$, $(i, j) \in \{1, \ldots, m_t^i\}$ that satisfy, for $\theta = \frac{1}{2}$,

$$0 = c_j^i - c_j^{i-1} - [(1 - \theta)f_j(t^{i-1}, c^{i-1}, \sigma^{i-1}) + \theta f_j(t^i, c^i, \sigma^i)] \Delta t,$$

$$(i, j) \in \{1, \ldots, m_t^i\} \times \{1, \ldots, n_x^i\}, \quad c_j^0 = c_j^{i-1, 0}, \quad j \in \{1, \ldots, n_x\},$$

where $c_j^{i-1, 0}, j = 1, \ldots, n$, denotes a solution to the corresponding scheme on the interval $[\tau^{i-2}, \tau^{i-1}]$ with $c_j^{0, 0} = \max \{0, x_j - S_0\}$. The function $f$ is as in (15) and we use the notation $c^t = (c_j^1, \ldots, c_j^{n_x})'$ and $\sigma^t = (\sigma_1^t, \ldots, \sigma_{n_x}^t)'$, where $'$ denotes the transpose. The equivalent of the optimization problem (22), with the time-continuous $c$ replaced by the solution to our Crank-Nicolson scheme, then reads

$$\min_{\alpha \in \mathbb{R}^{n_x}} g(c^m \alpha) + h(\xi^t(\alpha))$$

subject to: $c$ and $\sigma$ satisfy (31) with $\sigma$ given by (30).
If we introduce a Lagrangian multiplier $\lambda_j^i$, $(i,j) \in \{0, \ldots, m_i^j - 1\} \times \{1, \ldots, n_x\}$, we can define the Lagrangian corresponding to the minimization problem (32) as

$$L(\lambda, c, \alpha) = g(c^{\alpha}) + \sum_{i=1}^{m_i^j} \sum_{j=1}^{n_x} \lambda_j^i \left( c_j^i - c_j^i - \left[ (1 - \theta) f_j^i - \theta f_j^i \right] \Delta t \right)$$

(33)

where we have used the abbreviation $f_j^i = f_j(t^i, c^i, \sigma^i)$. Optimality conditions for (32) can then be written as

$$\partial_{c_j^i} L(\lambda, c, \alpha) = 0, \ (i,j) \in \{1, \ldots, m_i^j\} \times \{1, \ldots, n_x\}$$

$$\partial_{\lambda_j^i} L(\lambda, c, \alpha) = 0, \ (i,j) \in \{0, \ldots, m_i^j - 1\} \times \{1, \ldots, n_x\}$$

$$\partial_{\alpha_k} L(\lambda, c, \alpha) = 0, \ k \in \{1, \ldots, n_k^\alpha\}.$$ 

(34)

The second line, with the partial derivatives $\partial_{\lambda_j^i} L$, yields exactly the Crank-Nicolson scheme in $c$ from (31). The derivatives with respect to $c_j^i$ yield the following finite difference scheme for the multiplier $\lambda,$

$$0 = \lambda_j^{i-1} - \lambda_j^i - \partial_{c_j^i} \left[ \theta \left( \lambda_{j-1}^{i-1} f_{j-1}^i + \lambda_j^{i-1} f_j^i + \lambda_{j+1}^{i-1} f_{j+1}^i \right) + (1 - \theta) \left( \lambda_j^{i-1} f_j^i + \lambda_{j+1}^{i-1} f_{j+1}^i \right) \right] \Delta t,$$

$$(i,j) \in \{1, \ldots, m_i^j - 1\} \times \{2, \ldots, n_x - 1\}$$

(35)

and the $\alpha$-derivative in (34) gives the equations

$$0 = \sum_{i=1}^{m_i^j} \sum_{j=1}^{n_x} \lambda_j^i (1 - \theta) \partial_{\alpha_k} f_j^i - \theta \partial_{\alpha_k} f_j^i \Delta t, \ k \in \{1, \ldots, n_k^\alpha\}.$$ 

(36)

We can develop the derivatives of $f$ in (35) and (36) to obtain explicit expressions for these equations. Note that $\lambda_j^i$ has not been defined for $j = 0$ and $j = n_x + 1$, but let us introduce the boundary values

$$\lambda_j^i = 0, \ (i,j) \in \{0, \ldots, m_i^j - 1\} \times \{0, n_x + 1\}.$$ 

(37)

If we use the notation

$$\nu_j^i = \frac{1}{2} \frac{x_j^2}{\sigma_j^i} (\sigma_j^i)^2, \ (i,j) \in \{1, \ldots, m_i^j\} \times \{1, \ldots, n_x\}$$

$$\nu_j^i = 0, \ (i,j) \in \{1, \ldots, m_i^j\} \times \{0, n_x + 1\},$$

(38)
we obtain the following equations from (35)

\[ 0 = \lambda_j^i - \lambda_j^{i-1} \]

\[ + \left\{ \theta \left( \frac{2}{\mu_j^- + \mu_j^+} \left[ \frac{\nu_j^i}{\mu_j^-} \lambda_j^{i-1} - \lambda_j^{i-1} \right] + \left( \frac{1}{\mu_j^-} + \frac{1}{\mu_j^+} \right) \nu_j^i \lambda_j^{i-1} + \frac{1}{\mu_j^+} \nu_{j+1}^i \lambda_{j+1}^{i-1} \right) \right. 

\[ - (r^i - q^i) \frac{1}{\mu_j^- + \mu_j^+} \left[ \frac{\mu_j^-}{\mu_j^+} (\lambda_j^{i-1} - \lambda_j^{i-1}) + \frac{\mu_j^-}{\mu_j^+} (\lambda_j^{i-1} - \lambda_j^{i-1}) \right] - q^i \lambda_j^{i-1} \right) \]

\[ + (1 - \theta) \left( \frac{2}{\mu_j^- + \mu_j^+} \left[ \frac{\nu_j^i}{\mu_j^-} \lambda_j^{i-1} + \left( \frac{1}{\mu_j^-} + \frac{1}{\mu_j^+} \right) \nu_j^i \lambda_j^{i-1} + \frac{1}{\mu_j^+} \nu_{j+1}^i \lambda_{j+1}^{i-1} \right) \right. 

\[ - (r^i - q^i) \frac{1}{\mu_j^- + \mu_j^+} \left[ \frac{\mu_j^-}{\mu_j^+} (\lambda_j^{i-1} - \lambda_j^{i-1}) + \frac{\mu_j^-}{\mu_j^+} (\lambda_j^{i-1} - \lambda_j^{i-1}) \right] - q^i \lambda_j^{i-1} \right) \right\} \Delta t, \quad (39) \]

\[ (i, j) \in \{1, \ldots, m_i^k - 1\} \times \{1, \ldots, n_x\} \]

\[ 0 = \partial_{c_j} g(c^i) + \lambda_j^{i-1} \]

\[ - \theta \left( \frac{2}{\mu_j^- + \mu_j^+} \left[ \frac{\nu_j^i}{\mu_j^-} \lambda_j^{i-1} - \lambda_j^{i-1} \right] + \left( \frac{1}{\mu_j^-} + \frac{1}{\mu_j^+} \right) \nu_j^i \lambda_j^{i-1} + \frac{1}{\mu_j^+} \nu_{j+1}^i \lambda_{j+1}^{i-1} \right) \]

\[ - (r^i - q^i) \frac{1}{\mu_j^- + \mu_j^+} \left[ \frac{\mu_j^-}{\mu_j^+} (\lambda_j^{i-1} - \lambda_j^{i-1}) + \frac{\mu_j^-}{\mu_j^+} (\lambda_j^{i-1} - \lambda_j^{i-1}) \right] - q^i \lambda_j^{i-1} \right) \Delta t, \quad (i, j) \in \{m_i^k\} \times \{1, \ldots, n_x\}. \]

Equation (36) for the derivatives in \( \alpha \) can be developed into

\[ 0 = \partial_{\alpha_k} \xi_k \sum_{i=1}^{m_i^k} \sum_{j \in \mathbb{Z}^+ (k)} \lambda_j^{i-1} \left[ \frac{1}{2} \nu_j^2 \right] \left( \frac{1}{2} D_j^2 c_j^{i-1} + \theta D_j^2 c_j^i \right) \Delta t, \quad k \in \{1, \ldots, n_i^\kappa\}, \quad (40) \]

where \( \xi \) was defined in (21). The three equations (31), (39) and (40) constitute a system of equations in the three variables \( c, \lambda \) and \( \alpha \) whose solution represent first order optimality conditions for our problem (32). If we define a variable \( w \in \mathbb{R}^{2m_i^k(n_x+2)+n_i^\kappa} \) according to

\[ w_{i+1}^{j} := \lambda_j^i, \quad (i, j) \in \{1, \ldots, m_i^k\} \times \{0, \ldots, n_x + 1\} \]

\[ w_{i+2}^{j} := c_j^i, \quad (i, j) \in \{1, \ldots, m_i^k\} \times \{0, \ldots, n_x + 1\} \]

\[ w_{2m_i^k} := \alpha_k, \quad k \in \{1, \ldots, n_i^\kappa\}, \quad (41) \]
we can write the equations (31), (39) and (40) with associated boundary conditions as

\[
\begin{align*}
F_{j+2(i-1)(n_x+2)}(w) &:= \lambda_{j}^{-1}, \\
F_{j+2(i-1)(n_x+2)}(w) &:= \text{rhs of (39), 1st eq.,} \\
F_{j+2(i-1)(n_x+2)}(w) &:= \text{rhs of (39), 2nd eq.,} \\
F_{j+2(i-1)(n_x+2)}(w) &:= c^s_{j} - S_{0} e^{-\int_{t_i}^{t} r_s ds}, \\
F_{j+2(i-1)(n_x+2)}(w) &:= \bar{c}_{j}, \\
F_{2m_{1}(n_x+2)+k}(w) &:= \text{rhs of (40)},
\end{align*}
\]

where \( F \), then, is a function \( \mathbb{R}^{(2m_{1}+1)(n_{x}+2)+n_{s}} \rightarrow \mathbb{R}^{(2m_{1}+1)(n_{x}+2)+n_{s}} \). Our optimality condition now simply reads

\[
F(w) = 0.
\]

Since we have analytical expressions for \( F \) according to (42), it is straightforward to calculate the Jacobian matrix of \( F \) with respect to \( w \), which allows for Newton based methods to be employed when solving \( F(w) = 0 \) numerically. We will return to our choice of optimization method and provide some details about our implementation below in Section 3, before we present numerical results in Section 4.

3 Implementation

Estimating the parameters in a partial differential equation from noisy data is a notoriously ill-posed problem [20], [10]. A key to success in our problem formulation is therefore the Tikhonov regularization [20], [10] provided by the function \( h(\alpha) \) defined in (23). In general, the resulting system (43) is easier to solve for higher levels of regularization, i.e., for higher values of the constants \( \epsilon_{1} \) and \( \epsilon_{2} \) in (23). On the other hand, while a high level of regularization facilitates the solution of the problem and yields smoother solutions, the resulting function will typically be further away from the data that is to be replicated. For this reason, we solve \( F(w) = 0 \) for successively smaller values of the regularizing parameters \( \epsilon_{1}, \epsilon_{2} \), and check after each round if the obtained solution is “close enough” to the input market prices. Let us redefine \( \hat{c} \) and \( \hat{p} \) from (25) and (26) to be the analogous linear interpolations of \( c^{m_{i}}, \) i.e, the discrete solution to the Crank-Nicolson scheme (31) at the last time-step \( m_{i} \). Our stopping criterion in the optimization will be that all model values are inside a certain fraction of the observed bid-ask spread on the quoted put- and call options. With the notation from (27), this can be written

\[
\begin{align*}
|\hat{p}_{k} - \bar{p}_{k}| < \eta \left( p^{\text{ask}}_{k} - p^{\text{bid}}_{k} \right), & \quad k \in \{1, \ldots, n_{p}\}, \\
|\hat{c}_{k} - \bar{c}_{k}| < \eta \left( c^{\text{ask}}_{k} - c^{\text{bid}}_{k} \right), & \quad k \in \{1, \ldots, n_{c}\},
\end{align*}
\]

where we have chosen \( \eta = 0.4 \). Our iterative procedure with successively smaller regularization parameters is summarized in Algorithm 1. The first step in the optimization is to choose an initial guess. In addition to the three variables \( c, \lambda \) and \( \alpha \), we need an initial guess for the reference volatility \( \tilde{\sigma} \) in (23). We have used a local volatility constructed from a so called affine stochastic volatility model [7] as our \( \tilde{\sigma} \). Specifically, we use spot process with a Heston volatility [12], multiplied by an exponential Meixner Lévy process [19]. Some further details
on our construction of a local volatility from this affine model is given in [15][Paper III]. Once \( \tilde{\sigma} \) is fixed, we set \( \alpha = \tilde{\sigma} \) as our initial guess. An initial guess for \( c \) is then readily obtained by solving the linear Crank-Nicolson scheme (31) for \( \sigma \) given by (19) and (21). As for the \( \lambda \)-variable, we use \( \lambda = 0 \) as initial guess.

Algorithm 1

1: Choose an initial guess \( w_0 \) for \( w \) in (42) and a reference volatility \( \tilde{\sigma} \) in (23).
2: Define \( \bar{\epsilon}_1 := 1, \bar{\epsilon}_2 := 10^{-5} \).
3: for \( n = 0 \to 5 \) do
4:   Set \( \epsilon_1 = \bar{\epsilon}_1 10^{3(1-\frac{n}{5})}, \epsilon_2 = \bar{\epsilon}_2 10^{3(1-\frac{n}{5})} \) in \( h \) from (23).
5:   Find \( w \) that solves \( F(w) = 0 \) given the initial guess \( w_0 \).
6: if (44) is satisfied then
7:   break
8: end if
9: Set \( w_0 = w \).
10: end for

The solving step on line 5 of Algorithm 1 is performed by means of a trust-region Newton method. This class of methods is designed to robustify the convergence properties of the Newton method while preserving its advantageous quadratic convergence close to an optimal point [5]. The exact algorithm we use is given in [15][Introduction, Algorithm 2]. The most computationally heavy step in a Newton based optimization algorithm typically is the calculation of the Newton step \( \Delta w \), which in our case is given by the solution to \( J_w \Delta w = -F(w) \), where \( J_w \) is the Jacobian matrix of \( F \) from (42). Our implementation of the Algorithm 1 is made in C++ and for the solution of this linear system we use the PARDISO solver, provided as part of the MKL package, which is an efficient and threaded direct linear solver.

An important feature of our approach is that the function \( \sigma \) is defined to be piecewise constant on a grid in time and space that is coarser than the grid used for the discretization of the actual differential equation. The bid-ask price spread on options prices can often be on the order of a basis point of the underlying spot price. A finite difference scheme of the type (31) will require a fairly dense grid, typically with over \( 10^5 \) grid points, to achieve sufficient accuracy in relation to this precision demanded by the data. In contrast, the option market on a major equity index will at best provide a few hundred price quotes at a given moment in time. If we were to define the volatility on the same grid as the one used in the numerical scheme, we would therefore be faced with an optimization problem with approximately a thousand unknown volatility values to fit for every observed data point. To avoid such a severely underdetermined problem, we choose a coarser discretization for the volatility than used in the discrete scheme (31). We opt for a grid which essentially corresponds to the points in space and time for which we observe data. As we have seen, our grid for the piecewise constant \( \sigma \) is taken to be the maturities for the observed option quotes. To avoid such a severely underdetermined problem, we choose a coarser discretization for the volatility than used in the discrete scheme (31). We opt for a grid which essentially corresponds to the points in space and time for which we observe data. As we have seen, our grid for the piecewise constant \( \sigma \) is taken to be the maturities for the observed option quotes. In space we will, for each maturity \( \tau^\iota, \iota = 1, \ldots, n_\tau \), simply form the union \( K_p^\iota \cup K_c^\iota \) of the grids from (24), and add a few grid points to the left of the lowest and the right of the highest strike at which market data is observed.

We now define the grids in time and space used for the Crank-Nicolson scheme (31). As defined in (29), the time grid has a constant step \( \Delta t = \frac{\tau^\iota - \tau^{\iota-1}}{m^\iota} \) between any two maturities.

\(^{1}\text{Intel® Math Kernel Library, MKL 11.2} \)}}
The space grid \( \chi \) from (9) is chosen according to

\[
\bar{\chi} := \{ x_0, \ldots, x_{n\chi+1} \}, \quad x_j = S_0 \left[ 1 + \tan \left( \frac{j}{n\chi+1} \right) d_- + \frac{j}{n\chi+1} d_+ \right],
\]

for \( n\chi = 375 \), which gives a total of nodes such that \( x_j - x_{j-1} \approx \frac{S_0}{200} \) for \( x_j \approx S_0 \) and \( x_j - x_{j-1} \approx \frac{S_0}{100} \) for \( x_j \approx \frac{S_0}{5} \).

Finally a note on market data. We have used market data\(^2\) on options written on the European equity index EurStoxx 50. The market quotes represent a snapshot of the market at a given time, and we have used all available data points. In addition to the option prices, we need the interest rate \( r \) and the dividend yield \( q \) in (1). We have used a zero coupon curve\(^2\) with values given at the option maturities from (6) and then calculated an interest rate \( r \) using the expression for \( b \) from (4) that is piecewise constant and left continuous between maturities. The dividend yield \( q \) is taken to be piecewise constant in the same fashion and is - given the interest rate \( r \) - calculated from the forward price \( f w \) from (4). The forward price we use for the dividend calculation is obtained by applying put-call parity (3) to the pair of put- and call option price quotes with strike closest to the spot price \( S_0 \). We provide the values of the interest rate \( r \) and dividend yield \( q \) used on each interval in Table 2. We have not verified whether or not the dividend yields obtained are realistic in that they reflect reasonable values for dividends actually paid by the individual stocks that comprise the Euro Stoxx 50 index. However, the displayed dividend values, along with the interest rate values, do match the “implicit forward prices” obtained from the put call parity relation.

<table>
<thead>
<tr>
<th>( t )</th>
<th>16</th>
<th>44</th>
<th>79</th>
<th>170</th>
<th>261</th>
<th>352</th>
<th>443</th>
<th>625</th>
</tr>
</thead>
<tbody>
<tr>
<td>( r )</td>
<td>0.000968</td>
<td>0.0017393</td>
<td>0.0025088</td>
<td>0.0022378</td>
<td>0.0025194</td>
<td>0.0030292</td>
<td>0.0029132</td>
<td>0.0037395</td>
</tr>
<tr>
<td>( q )</td>
<td>0.066573</td>
<td>0.22451</td>
<td>0.049823</td>
<td>0.012355</td>
<td>0.022658</td>
<td>0.012288</td>
<td>0.10236</td>
<td>0.012212</td>
</tr>
</tbody>
</table>

Table 2: The interest rate \( r \) and dividend yield \( q \) used for in the calculations. Both \( r \) and \( q \) are taken to be piecewise constant and left continuous over the time intervals defined by the values for \( t \) given in the table, where the unit is in days.

4 Numerical results

The option quotes on the Euro Stoxx 50 that we consider in this example were observed on the 3rd of April 2013 at around 13:07 CET. The spot price at this time was \( S_0 = 2660.08 \) euros. We list the bid- and ask quotes on the put options in Table 4 and those on the call

\(^2\)The option market data and the zero-coupon values were provided by Svenska Handelsbanken AB.
options in Table 5. We have 284 put option quotes and 248 call option quotes, giving a total of 532 bid-ask pairs.

The piecewise constant, left continuous local volatility resulting from our algorithm is displayed in Figure 1 and the volatility values at the nodes of the grid are provided in Table 3. We observe that the volatility has some unsmooth features, particularly for the shorter maturities. In general, prices for shorter maturities are harder to replicate. As is the general case for inverse problems, there is a conflict between the smoothness of the control parameter and the ability to replicate observed data, and higher values for the penalty constants $\epsilon_1$ and $\epsilon_2$ in (23) typically yields a smoother volatility but a worse fit to observed prices.

For the data set under consideration, our calibration algorithm yields a volatility function such that the model prices fit within the spread of all the quoted prices in tables 4 and 5. The model put- and call prices are given in Table 6 and Table 7 respectively.

The total solving time used in Algorithm 1 for the the example case is around 211 seconds when executed on a an Intel i7 processor of 2.93 GHz. This execution time should be short enough to make the algorithm useful also in practice. The execution time, of course, depends on the number of variables used for the discretization of the partial differential equations. As can be seen in Table 1, we have used a fairly high number of grid points in time. Reducing the number of time steps will reduce the execution time significantly.

5 Discussion

We have developed a Lagrangian method for calibrating a local volatility function from observed market data. In the example case we gave above, we find a volatility such that a numerical solution of Dupire’s solution with this volatility fits within the bid-ask spread of all observed option quotes. When we test our algorithm for other sets of option prices on the same index observed around the same period, we obtain equally good results, with none or very few option prices not replicated within the spread. The algorithm thus seems robust in its ability to accurately replicate market data. It is also robust in the sense that our trust-region Newton algorithm solves the system $F(u) = 0$ from (43) with a small residual error. We also believe that the execution time is sufficiently short to make it useful for practitioners.
Figure 1: The volatility function obtained from Algorithm 1 for the market data in Table 4 and Table 5. The volatility is piecewise constant and left continuous in the time variable $t$ and space variable $x$. The spot price $S_0$ was 2660.08 euros.
Table 3: Resulting volatility function. The volatility is left continuous on the rectangles in time and space defined by the grid in $t$ on the first row and in $x$ in the first column. The unit for $t$ is days and the values for $x$ are in euros.
### Table 4: Put bid- and ask prices on Euro Stoxx 50 from April 3 2013. The unit for \( t \) is days and the values for \( x \) are in euros. The spot price was \( S_0 = 2660.08 \) euros.

<table>
<thead>
<tr>
<th>( x/t )</th>
<th>16</th>
<th>44</th>
<th>79</th>
<th>170</th>
<th>261</th>
<th>352</th>
<th>443</th>
<th>625</th>
</tr>
</thead>
<tbody>
<tr>
<td>bid/ask</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1150</td>
<td>0.1</td>
<td>0</td>
<td>0.9</td>
<td>1.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1500</td>
<td>0.2</td>
<td>0</td>
<td>2.5</td>
<td>3.4</td>
<td>4.3</td>
<td>10.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1800</td>
<td>0.5</td>
<td>0</td>
<td>3.4</td>
<td>5.5</td>
<td>7.1</td>
<td>13.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2100</td>
<td>0.8</td>
<td>1</td>
<td>4.9</td>
<td>6.3</td>
<td>8.9</td>
<td>15.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2400</td>
<td>1.0</td>
<td>1</td>
<td>6.2</td>
<td>7.2</td>
<td>10.9</td>
<td>17.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2700</td>
<td>1.4</td>
<td>1</td>
<td>7.5</td>
<td>8.7</td>
<td>15.8</td>
<td>17.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3000</td>
<td>1.8</td>
<td>2</td>
<td>8.9</td>
<td>10.2</td>
<td>18.8</td>
<td>20.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3300</td>
<td></td>
<td>3</td>
<td>4.5</td>
<td>5.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3600</td>
<td></td>
<td>4</td>
<td>6.2</td>
<td>7.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3900</td>
<td></td>
<td>5</td>
<td>7.9</td>
<td>8.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4200</td>
<td></td>
<td>6</td>
<td>9.6</td>
<td>9.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4500</td>
<td></td>
<td>7</td>
<td>11.3</td>
<td>11.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4800</td>
<td></td>
<td>8</td>
<td>13.0</td>
<td>13.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5100</td>
<td></td>
<td>9</td>
<td>14.7</td>
<td>14.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5400</td>
<td></td>
<td>10</td>
<td>16.4</td>
<td>16.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5700</td>
<td></td>
<td>11</td>
<td>18.1</td>
<td>18.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6000</td>
<td></td>
<td>12</td>
<td>19.8</td>
<td>19.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6300</td>
<td></td>
<td>13</td>
<td>21.5</td>
<td>21.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6600</td>
<td></td>
<td>14</td>
<td>23.2</td>
<td>23.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6900</td>
<td></td>
<td>15</td>
<td>24.9</td>
<td>24.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7200</td>
<td></td>
<td>16</td>
<td>26.6</td>
<td>26.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7500</td>
<td></td>
<td>17</td>
<td>28.3</td>
<td>28.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7800</td>
<td></td>
<td>18</td>
<td>30.0</td>
<td>30.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8100</td>
<td></td>
<td>19</td>
<td>31.7</td>
<td>31.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8400</td>
<td></td>
<td>20</td>
<td>33.4</td>
<td>33.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Notes:**

- The data covers bids and asks for the put option on the Euro Stoxx 50 index.
- The unit for time \( t \) is days.
- Values for \( x \) are in euros.
- The spot price on April 3 2013 was \( S_0 = 2660.08 \) euros.
Table 5: Call bid- and ask prices Euro Stoxx 50 from April 3 2013. The unit for $t$ is days and the values for $x$ are in euros. The spot price was $S_0 = 2660.08$ euros.
<table>
<thead>
<tr>
<th>$x/t$</th>
<th>16</th>
<th>44</th>
<th>79</th>
<th>170</th>
<th>261</th>
<th>352</th>
<th>443</th>
<th>625</th>
</tr>
</thead>
<tbody>
<tr>
<td>1550</td>
<td>0.32</td>
<td>4.98</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1550</td>
<td>0.45</td>
<td>6.16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1600</td>
<td>0.60</td>
<td>7.57</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1650</td>
<td>0.78</td>
<td>9.25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1700</td>
<td>0.99</td>
<td>11.22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1750</td>
<td>1.24</td>
<td>13.53</td>
<td>34.72</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1800</td>
<td>1.55</td>
<td>16.21</td>
<td>40.45</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1850</td>
<td>1.96</td>
<td>19.35</td>
<td>46.87</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1900</td>
<td>0.66</td>
<td>11.44</td>
<td>22.96</td>
<td>54.00</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1950</td>
<td>0.74</td>
<td>13.69</td>
<td>27.12</td>
<td>61.91</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2000</td>
<td>0.83</td>
<td>16.44</td>
<td>31.90</td>
<td>46.14</td>
<td>70.66</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2050</td>
<td>1.21</td>
<td>19.79</td>
<td>37.39</td>
<td>53.49</td>
<td>80.32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2100</td>
<td>1.61</td>
<td>23.83</td>
<td>43.70</td>
<td>61.74</td>
<td>90.95</td>
<td>128.85</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2150</td>
<td>2.18</td>
<td>28.69</td>
<td>50.97</td>
<td>70.98</td>
<td>102.66</td>
<td>143.07</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2200</td>
<td>3.01</td>
<td>34.53</td>
<td>59.34</td>
<td>81.28</td>
<td>115.50</td>
<td>158.29</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2250</td>
<td>4.21</td>
<td>41.57</td>
<td>68.99</td>
<td>92.78</td>
<td>129.62</td>
<td>174.63</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2300</td>
<td>5.98</td>
<td>49.94</td>
<td>80.04</td>
<td>105.58</td>
<td>145.10</td>
<td>192.14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2350</td>
<td>8.56</td>
<td>59.85</td>
<td>92.59</td>
<td>119.78</td>
<td>160.03</td>
<td>210.89</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2400</td>
<td>10.27</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2450</td>
<td>12.21</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2500</td>
<td>14.42</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2550</td>
<td>16.83</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2600</td>
<td>19.33</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2650</td>
<td>21.92</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2700</td>
<td>24.72</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2750</td>
<td>27.72</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2800</td>
<td>30.92</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2850</td>
<td>34.32</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2900</td>
<td>37.92</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2950</td>
<td>41.72</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3000</td>
<td>45.82</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3050</td>
<td>49.32</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3100</td>
<td>53.12</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3150</td>
<td>57.22</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3200</td>
<td>61.62</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3250</td>
<td>66.42</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3300</td>
<td>71.62</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3350</td>
<td>77.22</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3400</td>
<td>83.12</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3450</td>
<td>89.32</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3500</td>
<td>95.92</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3550</td>
<td>102.82</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3600</td>
<td>109.92</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3650</td>
<td>117.32</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3700</td>
<td>124.92</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3750</td>
<td>132.82</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3800</td>
<td>140.92</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3850</td>
<td>149.22</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3900</td>
<td>157.72</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3950</td>
<td>166.42</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4000</td>
<td>175.32</td>
<td>65.51</td>
<td>106.81</td>
<td>135.51</td>
<td>180.52</td>
<td>231.01</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6: Put model prices corresponding to the market quotes given in Table 4. The unit for $t$ is days and the values for $x$ are in euros. The spot price was $S_0 = 2660.08$ euros.
Table 7: Call model prices corresponding to the market quotes given in Table 5. The unit for \( t \) is days and the values for \( x \) are in euros. The spot price was \( S_0 = 2660.08 \) euros.
References


