Smiling at Evolution *

D.A. Bloch †, C.A. Coello Coello ‡
Mizuho Securities §, CINVESTAV-IPN ¶

Working Paper

21st of May 2010

Abstract

We generate a reliable implied volatility surface without arbitrage in space and in time by parametrising a mixture of shifted lognormal densities under constraints and use a Differential Evolution algorithm to calibrate the model's parameters to a finite set of option prices. It is used for marking options not directly visible as well as for computing a proper deterministic local volatility. To do so, we devise an evolutionary algorithm handling constraints in a simple and efficient way. Using some of the improvements made to the DE algorithm and taking advantage of the specific structure of our objective function, we use special operators to help satisfy the equality constraints together with feasibility rules to handle the inequality constraints. Finally, we propose a modified algorithm for solving our optimisation problem under constraints which, after testing on real market data, greatly improves its performances.

1 Introduction

Since the Black-Scholes model in [BS73], market prices of index options and foreign exchange options have reached a high degree of liquidity such that they became the benchmark for marking to market or calibrating option pricing models when pricing and hedging exotic options. Deterministic local volatility models (see Dupire in [Du94]) are widely used because they give perfect fit to market prices allowing for simple hedging of exotic options. In order to implement the deterministic local volatility, Dupire's formula requires the knowledge of call prices for all strikes and maturities. However, in practice we can only observe a few market prices from standard strikes and maturities. These points do not necessary satisfy the no-arbitrage conditions, can have wide or narrow spreads depending on the liquidity on the market and the volume traded. As a result, the market is incomplete and there are more than one acceptable price (volatility) surfaces satisfying the no-arbitrage conditions.

The recent reform of the over-the-counter (OTC) market is seen as an evolutionary step leading bilateral trades to a shift towards central counterparties (CCP) to mitigate the counterparty risks. Wood in [Woo10] reports that about 50% to 70% of the market will be eligible for clearing, standardising trades within the next few years. As it becomes easier and safer to trade, volume will increase while profit will decline. As a

*We are gratefull to Nicole El Karoui, Monique Jeanblanc, Mark Davis, Rama Cont, the GRO and the GRM for their useful comments and suggestions. All mistakes are ours.

†daniel.bloch@mizuho-sc.com
‡ccoello@cs.cinvestav.mx
§Otemachi First Square, 1-5-1 Otemachi, Chiyoda-ku, Tokyo 100-0004
¶Depto. de computacion, Av. Instituto Politecnico Nacional No. 2508, Col. San Pedro Zacatenco, Mexico, D.f. 07300

result, demand will increase for options for which prices are not directly visibe, that is vanilla options with longer matutities and strikes further in and out of the money. Hence, the need for obtaining a reliable smile surface from market prices.

A standard approach to compute local volatility is to interpolate and extrapolate market prices or volatilities to complete the market. Practitioners use the Black-Scholes implied volatility and smooth the prices in a parabolic way to generate the missing prices. However, direct interpolation of implied volatility surfaces does not guarantee a resulting smooth risk-neutral density, hence a proper local volatility surface. Among the different techniques proposed for obtaining from market prices a smooth volatility surface, Rebonato et al in [RC04] argued that modeling directly the density was the most desirable approach. Alternatively, the no-arbitrage conditions on the local volatility being simply that it must be positive and Lipschitz continuous, one can assume a particular functional form for the local volatility and fit it to market prices. For example, in the case of deterministic rates, Ben Hamida et al in [BC04] introduced the parametrisation of a set of admissible local volatility surface. However, in markets with long maturity products and discrete dividends such as the Japanese market, it is important for model pricing to obtain a reliable volatility surface satisfying the no-arbitrage constraints not only in space but also in time. Therefore, we are going to assume a functional form for the market prices and solve a constrained numerical optimisation problem (CNOP), generating a surface without arbitrage in time and in space as closely as possible to market data.

We are going to consider an Evolutionary Algorithm (EA) to calibrate a local volatility diffusion model to a finite set of option prices, devising an algorithm that handle constraints in a simple and efficient way. Evolutionary algorithms introduced by Fogel in [Fog66] and Holland in [Hol75] are robust and efficient optimisation algorithms based on the theory of evolution proposed by Darwin in [Dar82], where a biological population evolves over generations to adapt to an environment by mutation, recombination and selection. They search from multiple points in space instead of moving from a single point like gradient-based methods do. Moreover, they work on function evaluation alone (fitness) and do not require derivatives or gradients of the objective functions. Among the different EA's commonly used to solve CNOPs such as evolutionary programming, evolution strategies, genetic algorithm and many more, differential evolution (DE) became very popular. DE is a population-based approach to function optimisation generating a new position for an individual by calculating vector differences between other randomly selected members of the population. The DE algorithm is found to be a powerful evolutionary algorithm for global optimisation in many real problems. As a result, since the original article of Storn and Price in [StPr95] many authors improved the DE model to increase the exploration and exploitation capabilities of the DE algorithm when solving optimisation problems.

Since EAs are search engines working in unconstrained search spaces they lacked until recently of a mechanism to deal with the constraints of the problems. The first attempts to handle the constraints were either to incorporate methods from mathematical programming algorithms within EAs such as penalty functions or to exploit the mathematical structure of the constraints. Then, a considerable amount of research proposed alternative methods to improve the search of the feasible global optimum solution. Most of the research on DE focused on solving CNOPs by using a sole DE variant, a combination of variants or combining DE with another search method. One of the most popular constraint handling mechanisms is the use of the three feasibility rules proposed by Deb in [Deb00] on genetic algorithms. Using some of the improvements to the DE algorithm combined with simple and robust constraint handling mechanisms we propose a modified algorithm for solving our optimisation problem under constraints which greatly improves its performances.

We present in Section (2) the dynamics of our underlying asset together with its local volatility. We describe in Section (3) our volatility model with constraints and discuss in Section (4) the DE algorithm for solving a nonlinear programming problem under constraints. Using the specific structure of our objective

function, we propose in Section (5) a constraint handling approach that combines a generic method together with a specific one by using special operators to help satisfy the equality constraints together with feasibility rules to handle inequality constraints. Dividing the population vector into subvectors, we apply the mutation and recombination operators independently to each subvector followed by a global selection method. Finally, we propose a robust algorithm for solving our optimisation problem under constraints with fast convergence and provide results showing the effectiveness of our algorithm.

2 The underlying asset

We consider the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ where $\mathcal{F}_t$ is a right continuous filtration including all $\mathbb{P}$ negligible sets in $\mathcal{F}$. For simplicity of exposition we let the market be complete and assume that there exist an equivalent martingale measure denoted $\mathbb{Q}$.

2.1 The dynamics

We let the underlying process $(S_t)_{t\geq 0}$ be a one-dimensional Ito process valued in the open subset D with dynamics under the risk-neutral measure $\mathbb{Q}$ being

$$\frac{dS_t}{S_t} = \mu_t dt + \sigma(t, S_t) dW_S(t) \tag{2.1}$$

where the drift $\mu : D \to \mathbb{R}$ as well as the diffusion $\sigma : D \to \mathbb{R}$ are regular enough to have a unique strong solution valued in D. We let the drift be $\mu_t = r_t - q_t$ where $(r_t)_{t\geq 0}$ is the deterministic spot rate and $(q_t)_{t\geq 0}$ is a deterministic repo rate. Using the concept of absence of arbitrage opportunities (AAO) (see Harrison and Kreps in [HK79]), contingent claims can be valued by taking expectation of their discounted payoffs under the risk-neutral measure.

We assume that we know at time t a continuum of market prices $C(t, S_t, T, K)$ such that given the family $\{C(T, K); (T, K) \in \mathbb{R}^+ \times \mathbb{R}^+\}$ the application $K \to C(T, K)$ is convex and decreasing continuously. Dupire showed in [Du94] that the deterministic local volatility can be estimated from market call and put prices of all strikes and maturities. Considering call prices as function of strike K and maturity T, he solved the forward equation

$$\partial_T C = \frac{1}{2}\sigma^2_{DLV}(T, K)K^2\partial_{KK}C - \mu_T K\partial_K C \tag{2.2}$$

with initial condition $C(t, S_t, K, t) = (S_t - K)^+$ obtaining a unique solution to the function of volatility

$$\sigma^2_{DLV}(T, K) = \frac{\partial_T C(t, S_t, K, T) + K(r_T - q_T)\partial_K C(t, S_t, K, T) + q_T C(t, S_t, K, T)}{\frac{1}{2}\partial_{KK}C(t, S_t, T, K)K^2}$$

called the deterministic local variance DLV. This result is crucial because it allows perfect fit to the market prices leading to a complete modification of the equity market. The Black-Scholes implied volatility is no longer the only form of observable volatility and one can directly use the local volatility for calibration. So, in principle one can directly diffuse a process S_t with stochastic instantaneous volatility provided that its local variance LV matches that of the DLV. That is, we may impose any dynamics such that the equality holds and the local variance stays non-negative. This is very appealing because we get perfect fit to the market prices but it is in general difficult to implement in practice. Hence, in the rest of this article we are going to concentrate on computing a smooth and arbitrage-free deterministic local volatility.

2.2 The spot model with discrete dividends

We are now going to introduce discrete dividends to the stock price with dynamics given in Equation (2.1) as well as to describe our framework. With the aim of obtaining closed-form solutions to European option prices we consider the Spot Model described by Frishling in [Fri02]. We let $C(0,t) = \frac{Re(0,t)}{B(0,t)} = e^{\int_0^t \mu_s ds}$ be the capitalisation from time zero until time t so that the forward capitalisation factor is

$$C(t,T) = \frac{Re(t,T)}{B(t,T)}$$

with $C(t,t) = 1$. Given a set of discrete dividends $(d_i)_{i\in\mathcal{N}}$, we let $D_t = D(0,t)$ be the accumulated dividends at time t in the interval $[0,t]$, that is the dividends paid from $t_0 = 0$ till time t and capitalised at t is given by

$$D_t = \sum_{i=0}^{\infty} \mathcal{H}(t - t_{d_i}) d_i C(t_{d_i}, t)$$

where $\mathcal{H}(.)$ is the heaviside function and $D_0 = 0$. We consider the process $(Z_t)_{t\geq 0}$ to be the asset price S_t plus the forward value of all dividends paid from today up to time t. In that setting, the stock price and the dynamics of the diffusion process under the risk-neutral measure $\mathbb{Q}$ becomes

$$\begin{aligned} S_t &= Z_t - D_t \\ dZ_t &= \mu_t Z_t dt + \sigma_Z(t, Z_t) Z_t dW_t \\ Z_0 &= S_0 \end{aligned} \tag{2.3}$$

This model is popular mainly due to the fact that adding the already paid dividends to the strike, the market price of a call option on the stock price seen at time $t_0 = 0$ is

$$C(t_0, S_0; K, T) = E^Q_{t_0}[e^{-\int_{t_0}^T r_s ds}(Z_T - K')^+]$$

where $K' = K + D_T$. Similarly, we can express the price of a call option on the diffusion process Z in terms of a call option on the stock price

$$C_Z(t_0, Z_0; k, T) = \frac{1}{B(t_0,T)} C(t_0, S_0; k - D_T, T) \tag{2.4}$$

To recover market prices we use Equation (2.4) to compute the local volatility with discrete dividends, getting

$$\sigma^2_Z(T,k) = \frac{\partial_T C(t, S_t, k - D(t,T), T) + q_T C(t, S_t, k - D(t,T), T) + (r_T - q_T) k \partial_k C(t, S_t, k - D(t,T), T)}{\frac{1}{2}\partial_{kk} C(t, S_t, T, k - D(t,T)) k^2}$$

which correspond to the local volatility on the stock price without discrete dividends given in Section (2.1).

3 The choice of a volatility model

Pricing options at strikes not observed in the market is necessary when pricing and hedging exotic derivative products dependent on the entire risk-neutral probability density function of the underlying. We saw earlier that Dupire's formula requires the knowledge of call prices for all strikes and maturities. However, in practice we can only observe a few market prices from standard strikes and maturities. For an index, one can easily obtain more than 100 points (prices or implied volatility) where some of them are listed while the others

are produced by brokers. On the other hand, there are only a few points for single stock (from 0 to 30) produced only by brokers. These points do not necessary satisfy the no-arbitrage conditions, can have wide or narrow spreads depending on the liquidity on the market and the volume traded. As a result, the market is incomplete and there are more than one acceptable price (volatility) surfaces satisfying the no-arbitrage conditions.

3.1 The standard approach

The standard approach is to interpolate and extrapolate market prices or volatilities to complete the market. Practitioners use the Black-Scholes implied volatility and smooth the prices in a parabolic way to generate the missing prices. As described by Daglish, Hull and Suo in [DHS06], a natural choice would be to perform a Taylor expansion up to the second order of the implied volatility surface around the money forward level. For example Malz in [Mal97] interpolated the smile within the region of observed prices with a polynomial and cut the volatility outside that region. Alternatively, one can fit with little control a parametric form for the implied volatility derived from a model which is usually the result of an asymtotic expansion of a stochastic volatility model, see for examples Hagan et al in [HKLW02] or Bloch in [Blo09]. Recently, Benaim et al in [BDK08] directly extrapolated the prices outside the range of observed prices with a simple convex function of strikes matching the market prices at the two boundaries as well as their first two derivatives. However, direct interpolation and extrapolation of implied volatility surfaces does not guarantee a resulting smooth risk-neutral density, hence a proper local volatility surface. Moreover, they rely on some interpolation and extrapolation of parameters in time with no guarantee of satisfying the calendar spreads. As explained by Overhaus et al in [OBBFJL07], satisfying these conditions when expressed in terms of implied volatilities is non trivial. Alternatively, the no-arbitrage conditions on the local volatility being simply that it must be positive and Lipschitz continuous, one can assume a particular functional form for the local volatility and fit it to market prices. For example, Ben Hamida et al introduced in [BC04] the parametrisation of a set of admissible local volatility surface by considering a cubic spline with ten spline nodes per maturity. Instead, we are directly going to assume a functional form for the market prices and solve a non-linear programming problem under constraints.

3.2 The parametric model

In a market with a limited number of prices, a model of interpolation and extrapolation of the volatility surface should have few parameters with the ability of mapping a large family of surfaces. Among the different techniques proposed for obtaining a smooth volatility surface from market prices, Rebonato et al in [RC04] argued that modeling directly the density was the most desirable approach. They extended the mixture of normals approach proposed by Alexander in [Ale01], obtaining a density with non-zero skew and satisfying the risk-neutral forward condition while retaining an unconstrained numerical search. However, in markets with long maturity products and discrete dividends such as the Japanese market, it is important for model pricing to obtain a reliable volatility surface satisfying the no-arbitrage constraint not only in space but also in time. So, we intend to generate a surface without arbitrage in time and in space as closely as possible to market data. We are going to use a model which was first initiated by the GRM and improved over time, see Baude et al in [BR02].

Given the framework described in Section (2.2) to represent the stock price with discrete dividends, we are going to use a parametric representation of the market call prices in order to smooth the data and get nice probability distribution function (pdf). We let $\Psi \subset \mathbb{R}^N$ be a vector of model parameters and consider a weighted sum of interpolation functions taken in a parametric family. We want each function to satisfy the no-free lunch constraints in such a way that the constraints are preserved in the weighted sum. Consequently, in our parametric model, the market option price $C_M(K,T)$ of strike K and maturity T is estimated at time $t_0 = 0$ by the weighted sum

$$C_M(t_0, S_0, B(t_0,T), R(t_0,T), D(t_0,T); K, T) = \sum_{i=1}^{n} a_i(T) F_i(t_0, S_0, P_T, R_T, D_T; K, T)$$

where $a_i(t)$ are the weights and where $R_t = Re(0,t)$ is the repo factor between $[0,t]$, $P_t = B(t_0,t) = e^{-rt}$ is the zero-coupon bond price, $C_t = C(t_0,t) = \frac{R_t}{P_t}$ is the cost of carry and $D_t = D(t_0,t)$ is the compounded sum of discrete dividends between $[0,t]$. Several families F_i can satisfy the No-Free-Lunch constraints and we choose a sum of shifted log-normal distributions, that is, using the Black-Scholes formula with shifted strike (modified by the parameters $\mu_i(t)$) as an interpolation function

$$\begin{aligned} F_i(t_0, S_0, P_T, R_T, D_T; K, T) &= Call_{BS}(t_0, S_0, R_T, P_T, K^{'}(K,T)(1+\mu_i(T)), T, \sigma_i(T)) \\ &= P_T Call_{Black}(t_0, S_0 C_T, K^{'}(K,T)(1+\mu_i(T)), T, \sigma_i(T)) \end{aligned}$$

where $K^{'}(K,t) = K + D_t$ with Black-Scholes price $Call_{BS}(t_0, S_0, R_T, P_T, K, T, \sigma) = C(t_0, S_0, K, T)$ and such that the time function $t \to \sigma_i(t)$ is regular enough. The time dependent parameters $a_i(t)$ and $\mu_i(t)$ are used to recover the time structure of the volatility surface. The simplest way to ensure that condition is to take the same time dependency for each μ, that is, $\mu_i(t) = \mu_i f(t)$ where μ_i is a constant and

$$f(t,\beta) = 1 - \frac{2}{1 + (1 + \frac{t}{\beta})^2}$$

with $\beta > 0$ for $f(t,.)$ to be positive. To increase control of the model we let each function F_i have its own parameter beta so that the function f becomes $f_i = f(t,\beta_i)$ for $i = 1,..,n$. Moreover, to keep manageable the no-free lunch constraints, ensuring that the weights sum to one, we let the weight $a_i(t)$ be proportional to $\frac{a_i^0}{f(t,\beta_i)}$ for some constant $a_i^0 > 0$, getting the representation

$$\mu_i(t) = \mu_i^0 f(t,\beta_i) \text{ and } a_i(t) = \frac{a_i^0}{f(t,\beta_i) \times norm}$$

where $norm = \sum_{i=1}^{n} \frac{a_i^0}{f(t,\beta_i)}$. Making the weights and the shift parameter time-dependent to fit a large class of volatility surfaces leads, for all time t, to the following no-free lunch constraints

- $a_i(t) \geq 0$ to get convexity of the price function
- $\sum_{i=1}^{n} a_i(t) = 1$ to get a normalised risk-neutral probability
- $\sum_{i=1}^{n} a_i(t)\mu_i(t) = 0$ to keep the martingale property of the induced risk-neutral pdf
- $\mu_i(t) \geq -1$ to get non-degenerate functions

where the pdf is normalised by construction. Fortunately, with seperable functions of time the above constraints simplify to

$$\begin{aligned} a_i^0 &\geq 0 \\ \sum_{i=1}^{n} a_i^0 \mu_i^0 &= 0 \\ \mu_i^0 &\geq -1 \end{aligned} \tag{3.5}$$

for all i. The model being invariant when multiplying all the terms a_i^0 with the same factor, we impose the normalisation constraint $\sum_{i=1}^{n} a_i^0 = 1$ to avoid different parameter sets to give the same model. Given the

n parameters β_i and assuming a constant volatility $\sigma_i(t) = \sigma_i^0$, there are $N = 4n$ elements in the vector Ψ but only $4n - 2$ free parameters since we can always write $a_1^0 = 1 - \sum_{i=2}^n a_i^0$ and $a_1^0\mu_1^0 = -\sum_{i=2}^n a_i^0\mu_i^0$.

As such, this model does not allow for the control of the long term volatility surface. Therefore, for our model to be complete we need to specify the time-dependent volatility $\sigma_i(t)$ to capture the term structure of the implied volatility surface. For simplicity and good performances, we choose the function

$$\sigma_i(t) = a_i e^{-c_i t} + d_i$$

for c_i, a_i and d_i positive constants. In that setting, the number of elements in the vector Ψ becomes $N = 6n$. As a result, we can therefore write at time $t_0 = 0$ the parametric model for a call option price of maturity t as

$$C_M(t_0, S_0, P_t, R_t, D_t; K, t) = \frac{1}{norm} \sum_{i=1}^{n} \overline{a}_i(t) Call_{BS}(t_0, S_0, R_t, P_t, \overline{K}(K,t), t, \sigma_i(t)) \tag{3.6}$$

where $\overline{K}(K,t) = K^{'}(K,t)(1 + \mu_i(t))$ and $\overline{a}_i(t) = \frac{a_i^0}{f(t,\beta_i)}$. Computation of the Greeks and the local volatility can be found in [Blo10].

4 Nonlinear programming problems with constraints

4.1 The calibration problem

Since the Black-Scholes model in [BS73], market prices of index options and foreign exchange options have reached a high degree of liquidity such that they became the benchmark to mark to market or calibrate option pricing models for pricing and hedging exotic options. More formally, using market prices, we need to estimate the vector Ψ of model parameters in order to price exotic options. This amount to solving the following inverse problem.

***Problem* 1** *Given prices $C_t(T_i, K_i)$ for $i \in I$ where I is the total number of market prices considered, find the vector Ψ of model parameters such that the discounted asset price $\hat{S}_t = e^{-rt}S_t$ is a martingale and the observed option prices are given by their risk-neutral expectations*

$$\forall i \in I \ , \ C_t(T_i, K_i) = e^{-r(T-t)} E^{\Psi}[(S(T_i) - K_i)^+ | S_t = S]$$

That is, we need to retrieve the risk-neutral process and not just the conditional densities which is equivalent to a moment problem for the process S. However, as explained in Section (3.2), in practice we do not know the call and put prices for all strike prices but only for a finite number of them so that extrapolation and interpolation is needed, resulting in solutions at best approximately verifying the constraints. It is typically an ill posed problem as there may be either no solution at all or an infinite number of solutions, see Cont and Tankov in [CT02] for more details. Therefore, one need to use additional criterions for choosing a solution. It means that we need to reformulate the calibration as an approximation problem, for instance minimising the in-sample quadratic pricing error

$$\begin{aligned} \Psi^* &= \arg\inf_{\Psi} \mathcal{J}(\Psi) \\ \mathcal{J}(\Psi) &= \sum_{i=1}^{n} w_i \left| P_t(T_i, K_i) - C_t(T_i, K_i; \Psi) \right|^2 \end{aligned} \tag{4.7}$$

where w_i is a weight associated to each market option price. Market practice is to solve the optimisation problem with a gradient-based minimisation method to locate the minima. In that case we can always find a solution but the minimisation function is not convex and the gradient descent may not succeed in locating the minimum. In order to circumvent these difficulties various authors proposed different regularisation methods all consisting in adding to the objective function a penalisation criterion. As a result, it makes the problem well-posed and allow for gradient-based optimisation algorithm. We will not discuss these criterions in detail and will refer the readers to Tankov in [Tan05] for more details on regularisation techniques.

In an incomplete market, a deterministic optimisation method will at best locate one of the local minima of the fitting criterion but will not guarantee the global minima and will not acknowledge the multiplicity of solutions of the initial calibration problem. To overcome these issues, Ben Hamida et al in [BC04] introduced a stochastic optimisation algorithm which generates a random sample from the set of global minima of the in-sample error and allow for the existence of multiple global minima. The population of model parameters is updated through cycles of independent random moves followed by selection according to pricing performance. Parametrising a set of admissible local volatility surface, they used an Evolutionary Algorithm (EA) to calibrate a local volatility diffusion model to a finite set of option prices. Instead, as explained in Section (3.2) we chose to parametrise a mixture of shifted lognormal densities under constraints. Hence, we are going to detail an alternative approach to solving non-linear programming problems under constraints that do not require computing the gradient of the model. We will need to devise an evolutionary algorithm that handle constraints in a simple and efficient way.

4.2 Defining the problems

We consider a system with the real-valued properties

$$g_m \text{ for } m = 0, .., P-1$$

making the objectives of the system to be optimised. Given a N-dimensional vector of real-valued parameter $X \subset \mathbb{R}^N$ the optimisation problem can always be written as

$$\min f_m(X)$$

where $f_m(.)$ is a function by which g_m is calculated and where each element $X(i)$ of the vector is bounded by lower and upper limits $L_i \leq X(i) \leq U_i$ which define the search space $\mathcal{S}$. We follow Lueder in [Lu90] who showed that all functions $f_m(.)$ can be combined in a single objective function $H : X \subset \mathbb{R}^N \to \mathbb{R}$ expressed as the weighted sum

$$H(X) = \sum_{m=1}^{P} w_m f_m(X)$$

where the weighting factors w_m define the importance of each objective of the system. Hence, the optimisation problem becomes

$$\min H(X)$$

so that all the local and global minima (when the region of eligibility in X is convex) can be found. However, since the problems of calibration in finance involves a single objective function, the optimisation function simplifies. Most complex search problems such as optimisation problems are constrained numerical problem (CNOP) more commonly called general nonlinear programming problems with constraints given by

$$\begin{aligned} g_i(X) &\leq 0 \ , \ i = 1, .., p \\ h_j(X) &= 0 \ , \ j = 1, .., q \end{aligned}$$

Equality constraints are usually transformed into inequality constraints by

$$|h_j(X)| - \epsilon \leq 0$$

where ϵ is the tolerance allowed. Given the search space $\mathcal{S} \subset \mathbb{R}^N$, we let $\mathcal{F}$ be the set of all solutions satisfying the constraints of the problems called the feasible region. It is defined by the intersection of $\mathcal{S}$ and the set of $p+q$ additional constraints. At any point $X \in \mathcal{F}$, the constraints $g_i(.)$ that satisfy $g_i(X) = 0$ are active constraints at X while equality constraints $h_j(.)$ are active at all points of $\mathcal{F}$. Many practical problems have objective functions that are non-differentiable, non-continuous, non-linear, noisy, multi-dimensional and have many local minima. This is the case of the calibration problem defined in Section (4.1). Evolutionary algorithms (EAs) introduced by Holland in [Hol62] and later in [Hol75] as well as by Fogel in [Fog66] are robust and efficient optimisation algorithms based on the theory of evolution proposed by Darwin in [Dar82], where a biological population evolves over generations to adapt to an environment by mutation, recombination and selection. They search from multiple points in space instead of moving from a single point like gradient-based methods do. Moreover, they work on function evaluation alone (fitness) and do not require derivatives or gradients of the objective functions. Since EAs are search engines working in unconstrained search spaces they lacked until recently of a mechanism to deal with the constraints of the problems. The first attempts to handle the constraints was to incorporate methods from mathematical programming algorithms within EAs such as penalty functions. Then, a considerable amount of research proposed alternative methods to improve the search of the feasible global optimum solution. Among the different EA's commonly used to solve CNOPs such as evolutionary programming, evolution strategies, genetic algorithm and many more, differential evolution (DE) became very popular. DE is a population-based approach to function optimisation generating a new position for an individual by calculating vector differences between other randomly selected members of the population. Most of the research on DE focused on solving CNOPs by using a sole DE variant, a combination of variants or combining DE with another search method. One of the most popular constraint handling mechanisms is the use of the three feasibility rules proposed by Deb in [Deb00] on genetic algorithms.

4.3 The DE algorithm

The Differential Evolution (DE) proposed by Storn and Price in [StPr95] is an algorithm that can find approximate solutions to nonlinear programming problems. It is a parallel direct search method which uses NP parameter vector

$$X_{i,G} \text{ for } i = 0, .., NP - 1$$

as a population for each generation G like any evolutionary algorithms. Each of the NP parameter vectors undergoes mutation, recombination and selection.

4.3.1 The mutation

The role of mutation is to explore the parameter space by expanding the search space. For a given parameter vector $X_{i,G}$ called the Target vector, the DE generates a Donor vector V made of three or more independent parent vectors $X_{r_l,G}$ for $l = 1, 2, ..$ where r_l is an integer chosen randomly from the interval $[1, NP]$ and different from the running index i. In the spirit of Wright in [Wri91], the main idea is to perturbate a Base vector $\hat{V}$ with a weighted difference vector (called differential vectors)

$$V = \hat{V} + F \sum_{l=1} \left(X_{r_{2l-1},G} - X_{r_{2l},G} \right)$$

where the mutation factor F is a constant taking values in $[0, 2]$ and scaling the influence of the set of pairs of solutions selected to calculate the mutation value. Most of the time, the Base vector is defined as the arithmetical crossover operator

$$\hat{V} = \lambda X_{best,G} + (1-\lambda) X_{r_1,G}$$

where $\lambda \in [0,1]$ allows for a linear combination between the best element $X_{best,G}$ of the parent population vectors and a randomly selected vector $X_{r_1,G}$. It is called a global selection when $\lambda = 1$ while when $\lambda = 0$ the base vector is the same as the target vector, $X_{r_1,G} = X_{i,G}$ and we get a local selection. In the special case where the mutation factor is set to zero, the mutation operator becomes a crossover operator.

4.3.2 The recombination

Recombination incorporates successful solutions from the previous generation. That is, according to a rule, we combine elements of the Target vector $X_{i,G}$ with elements of the Donor vector $V_{i,G}$ to create an offspring called the Trial vector $U_{i,G}$. In order to increase the diversity of the parameter vectors, elements of the Donor vector enter the Trial vector with probability CR. In the DE algorithm, each element of the Trial vector satisfies

$$\begin{aligned} U_{i,G}(j) &= V_{i,G}(j) \text{ for } j = n_r \mod dim, (n_r+1) \mod dim, ..., (n_r+L-1) \mod dim \\ &= X_{i,G}(j) \text{ for all other } j \in [0, .., NP-1] \end{aligned}$$

where dim is the dimension of the vector V (here $dim = N$) and the starting index n_r is a randomly chosen integer from the interval $[0, dim-1]$. Hence, a certain sequence of the element of U is equal to the element of V while the other elements get the original element of $X_{i,G}$. We only choose a subgroup of parameters for recombination, enhancing the search in parameter space. The integer L denotes the number of parameters that are going to be exchanged and is drawn from the interval $[1, dim]$ with probability

$$P(L > \nu) = (CR)^{\nu} \ , \ \nu > 0$$

The random decisions for both n_r and L are made anew at each new generation G. The term $CR \in [0,1]$ is the crossover factor controlling the influence of the parent in the generation of the offspring. A higher value means less influence from the parent. Most of the time, the mutation operator in Section (4.3.1) is sufficient and one can directly set the Trial vector equal to the Donor vector.

4.3.3 The selection

The tournament selection only needs part of the whole population to calculate an individual selection probability where subgroups may contain two or more individuals. In the DE algorithm, the selection is deterministic between the parent and the child. The best of them remain in the next population. We compute the objective function with the original vector $X_{i,G}$ and the newly created vector $U_{i,G}$. If the value of the latter is smaller than that of the former, the new Target vector $X_{i,G+1}$ is set to $U_{i,G}$ otherwise $X_{i,G}$ is retained

$$\begin{aligned} X_{i,G+1} &= U_{i,G} \text{ if } H(U_{i,G}) \leq H(X_{i,G}) \ , \ i = 0, .., NP-1 \\ &= X_{i,G} \text{ otherwise} \end{aligned}$$

Mutation, recombination and selection continue until some stopping criterion is reached. The mutation-selection cycle is similar to the prediction-correction step in the EM algorithm or in the filtering problems.

4.3.4 Convergence criterions

We allow for different convergence criterion in such a way that if one of them is reached, the algorithm terminates. We let f_{min} be the fittest design in the population and define $f_{a,G} = \frac{1}{NP} \sum_{i=0}^{NP-1} f(X_{i,G})$ as

the average objective value in a generation. Then, when the percentage difference between the average value and the best design reaches a specified small value ϵ_1

$$\frac{|f_{a,G} - f_{min}|}{|f_{a,G}|} \times 100 \leq \epsilon_1$$

we terminate the algorithm. Also, we let f_{minOld} be the fittest design in the previous generation and consider as a criterion the difference

$$f_{minOld} - f_{min} < \epsilon_2$$

where ϵ_2 is user defined. In that case, the DE algorithm will continue until there is no appreciable improvement in the minimum fitness value or some predefined maximum number of iterations is reached.

4.3.5 Pseudocode

We now present the pseudo code of a standard DE algorithm.

```
Initialise vectors of the population NP
Evaluate the cost of each vector
    for i=0 to Gmax do
        repeat
            Select some distinct vectors randomly
            Perform mutation
            Perform recombination
            Perform selection
                if offspring is better than main parent then
                replace main parent in the population
                end if
        until population is completed
        Apply convergence criterions
    next i
```

4.4 Improvements

The DE algorithm is found to be a powerful evolutionary algorithm for global optimisation in many real problems. As the DE algorithm performs mutation based on the distribution of the solutions in a given population, search directions and possible step sizes depend on the location of the individuals selected to calculate the mutation values. As a result, since the original article of Storn and Price in [StPr95] many authors improved the DE model to increase the exploration and exploitation capabilities of the DE algorithm when solving optimisation problems. We are going to review a few changes to the DE algorithm which greatly improved the performances of our problem.

4.4.1 Ageing

The DE selection is based on local competition only. The number of children that may be produced to compete against the parent $X_{i,G}$ should be chosen sufficiently high so that a sufficient number of child will enter the new population. Otherwise, it would lead to survival of too many old population vectors that may induce stagnation. To prevent the vector $X_{i,G}$ from surviving indefinitely, Storn in [Sto96] used the concept of ageing. One can define how many generations a population vector may survive before it has to be replaced due to excessive age. If the vector $X_{i,G}$ is younger than Num generations it remains unaltered otherwise it is replaced by the vector $X_{r_3,G}$ with $r_3 \neq i$ being a randomly chosen integer in $[1, NP]$.

4.4.2 Constraints on parameters

Given the parent vector $X_{i,G}$ for $i = 0, .., NP-1$ we define upper and lower bounds for each initial parameters as

$$L(j) \leq X_{i,G_0}(j) \leq U(j)$$

and we randomly select the initial parameter values uniformly on the interval $[L(j), U(j)]$ as

$$X_{i,G_0}(j) = L(j) + U(0,1)\big(U(j) - L(j)\big)$$

where $U(0,1)$ generates a random number in the range $[0,1]$ with a uniform distribution. Obviously, as the number of generation G increases, the DE algorithm will generate elements of the vector outside of the limits established (lower and upper) by an amount. Following Mezura Montes et al in [MezCoeTun04] this amount is substracted or added to the limit violated to shift the value inside the limits. If the shifted value is now violating the other limit, a random value inside the limits is generated.

4.4.3 Convergence

In order to accelerate the convergence process, when a child replaces its parent, Mezura-Montes et al in [MezCoeTun04] copied its value both into the new generation and into the current generation. It allows the new child, which is a new and better solution, to be selected among the r_l solutions and create better solutions. Therefore, a promising solution does not need to wait for the next generation to share its genetic code. Similarly, to improve performance and to accelerate the convergence process, Storn in [Sto96] explored the idea of allowing a solution to generate more than one offspring. Once a child is better than its parent, the multiple offspring generation ends. Following the same idea, Coello Coello and Mezura-Montes in [CoeMez03] and then Mezura-Montes et al in [MezVelCoe06] allowed for each parent at each generation to generate $k > 0$ offspring. Among these newly generated solutions, the best of them is selected to compete against its parent, increasing the chances to generate fitter offspring.

4.4.4 Self-adaptive parameters

Balamurugan et al in [BaSu07] considered the key parameters of control in DE algorithm such as the crossover CR and the weight applied to random differential F to be self-adapted. That is, the control parameters are not required to be pre-defined and can change during the evolution process. These control parameters are applied at the individual levels in the population so that better values lead to better individuals producing better offspring and hence better values. F is a scaling factor controlling the amplification of the difference between two individuals to avoid search stagnation. At generation $G = 1$ the amplification factor $F_{k,G}$ for the kth individual is generated randomly in the range $[0.1, 1.0]$. Then, at the next generations the control parameter is given by

$$F_{k,G+1} = \begin{cases} F_L + U(0,1)F_U \text{ if } U(0,1) < \tau_1 \\ F_{k,G} \text{ otherwise} \end{cases}$$

where $F_L = 0.1$, $F_U = 0.9$ and τ_1 represent the probability to adjust the parameter F.

4.5 Handling the constraints

We saw above that EAs in general and DE in particular lacked a mechanism to deal with the constraints of the problems. Recently, various academics worked on solving that problem, and one of the most popular constraint handling mechanisms was proposed by Deb in [Deb00] on genetic algorithms who used the three feasibility rules. In a single-objective optimisation problem, the traditional approach for handling constraints

is the penalty function method. The fitness of a candidate is based on a scale function F which is a weighted sum of the objective function value and the amount of design constraint violation

$$F(X) = f_1(X) + \Big(\sum_{k=1}^{p} \omega_k \max\left(g_k(X), 0\right) \sum_{k=p+1}^{q} \omega_k |h_k(X)|\Big)$$

where ω_k are positive penalty function coefficients and such that the kth constraint $g_k(.)$ and $h_k(.)$ should be normalised. This method requires a careful tuning of the coefficients ω_k to obtain satisfactory design, that is a balance between the objective function and the constraints but also between the constraints themselves. To overcome this problem, Deb proposed a penalty function approach based on the non-dominance concept, ranking candidates using the definition of domination between two candidates.

Definition 4.1 *A solution i is said to dominate a solution j if both of the following conditions are true*

1. *solution i is no worse than solution j in all objective*

$$\forall f_m(X_i) \leq f_m(X_j)$$

2. *solution i is strictly better than solution j in at least one objective*

$$\exists f_m(X_i) < f_m(X_j)$$

The constrained domination approach ranks candidates according to the following definition

Definition 4.2 *A solution i is said to constrained-dominate a solution j if any of the following conditions is true*

1. *solutions i and j are feasible and solution i dominates solution j.*
2. *solution i is feasible and solution j is not.*
3. *both solutions i and j are infeasible but solution i has a smaller constraint violation.*

He let the fitness function be

$$F(X) = \begin{cases} f(X) \text{ if } g_k(X) \leq 0 \ \forall k = 1, 2, .. \\ f_{max} + TACV \text{ otherwise} \end{cases}$$

where f_{max} is the objective value with the worst feasible solution in the population and (TACV) is the total amount of constraint violation

$$TACV = \sum_{k=1}^{p+q} \max\left(g_k(X), 0\right)$$

Therefore, solutions are never directly compared in terms of both objective function and constraint violation information. However, the high selection pressure generated by tournament selection will induce the use of additional procedure to preserve diversity in the population such as niching or sharing. Clearly, there is no tuning of the penalty function coefficients when the number of constraint is one. But, when multiple constraints are considered some considerations must be taken to relate constraints together. Again, many different approaches were proposed, for instance Coello Coello in [Coe00] modified the definition of the constrained domination approach given in Definition (4.2) such that if the individuals are infeasible he compares the number of constraints violated first and only in the case of a tie would he use the total amount of constraint violation. Going one step further, Oyama et al in [OSF05] introduced dominance in constraint

space. In that setting, any non-dominance ranking can be applied to feasible designs and infeasible designs separately. As a result, in a single-objective constrained optimisation problem, Bloch in [Blo10] modified the dominance-based tournament selection of Coello and Mezura with the non-dominance concept of Oyama et al, getting

Definition 4.3 *The new dominance-based tournament selection is*

1. *if solutions i and j are both feasible and solution i dominates in objective function solution j then solution i wins.*
2. *if solution i is feasible and solution j is not, solution i wins.*
3. *if solutions i and j are both infeasible and solution i dominates in constraint space solution j then solution i wins.*
4. *if solutions i and j are infeasible and non-dominated in constraint space, if solution i violates less number of constraints than solution j then solution i wins.*
5. *if solutions i and j are both infeasible, non-dominated in constraint space and violating the same number of constraints but solution i has a smaller TACV than solution j then solution i wins.*

5 Boundary search

Many techniques experience difficulties in solving real-world problems which include non-trivial constraints. For such problems, very often the global solution lies on the boundary of the feasible region. This is the case for our optimisation problem, since market prices evolves stochastically over time, leading to marking to market our pricing model on a constantly changing volatility surface. As a result, depending on the market prices and the shape of the volatility surface, some inequality constraints in Equation (3.5) might become active. Hence, the importance of problem specific operators to search the boundary in an efficient way. Even though some of our inequality constraints are active, we can not use the general approach for boundary search proposed by Leguizamon et al in [LeguizamonCoe09] since most of our constraints remain inactive. However, taking advantage of the structure of our objective function, we combines specific operators together with feasibility rules to handle simultaneously the equality and inequality constraints. Dividing the population vector into subvectors, we apply the mutation and recombination operators independently to each subvector followed by a global selection method.

5.1 Combining generic and specific method

Classical optimisation algorithms distinguish two main groups of constraint handling methods, the generic methods and the specific methods exploiting mathematical structure of the constraint. We considered in Section (4.5) the generic approach by using feasibility rules combined with a penalty function method after converting equality constraints into inequality ones. However, even if we generate initial parameters inside the domain of definition as explained in Section (4.4.2), it is very unlikely that the equality constraint derived in Equation (3.5) will be satisfied, making the initial vector guess X_{G_0} infeasible. Hence, if we handle our constraints using the dominance-based tournament selection introduced in Definition (4.3) we will rely sole on non-dominance ranking applied to infeasible designs. That is, our algorithm will rely on the relevance of the penalty function chosen as well as the fine tuning of its coefficients, slowing convergence. One way forward is to add some domain knowledge to our DE algorithm at initialisation time as Landa Becerra et al did in [LandaCoe06]. To do so, we let our population space consists of the vector $X^1_{G_0}$ generated as in Section (4.4.2) while our belief space that consists of the vector $X^2_{G_0}$ is generated in such way as to enforce the AAO constraints derived in Equation (3.5) so that it becomes feasible (see Annexe (A)). Given a probability

CR_0 taking values in $[0,1]$ we will compute our ith initial vector guess X_{i,G_0} from either the population vector or the belief vector. As a result, depending on the probability chosen, part of our initial NP vectors will be feasible and the dominance-based tournament selection will be applied to both feasible and infeasible designs.

Alternatively, we can consider specific methods which are only applicable to special types of constraints. For instance, Michalewicz et al in [MNM96] proposed a geometrical crossover and a special mutation operator to generate offspring lying on the boundary between the feasible and infeasible search space. Consequently, the search space is reduced as the exploration considers only the boundary of the feasible search space. Instead, we are going to propose a third approach by combining the generic method together with the specific one, using special operators to help satisfy the equality constraint. Using the specific structure of our objective function together with its constraints, we will conserve the penalty function method when handling inequality constraints, but we will be using a set of closed operators converting feasible solution into another feasible solution for equality constraints.

5.2 The structure of our optimisation problem

The DE algorithm described in Section (4.3) is a powerful and robust optimisation method when the model parameters are relatively homogeneous. However, the structure of our model parameters is peculiar in that our model price in Equation (3.6) is a weighted sum of interpolation functions taken in a parametric family. That is, we have n times the same parameters such that we can choose to gather similar parameters into subvectors when entered into the parameter vector, getting $X_{(.)} = [X^1_{(.)}, .., X^k_{(.)}]^\top$ where k is the total number of parameters per function and $X^l_{(.)}$ for $l \in [1,k]$ is a subvector of size n. Moreover, the constraints in Equation (3.5) do not directly applies to the vector $X_{(.)}$ but to its subvectors so that we can consider the constraints to be given by

$$g_i(X^i_{(.)}) \leq 0 \ , \ i = 1, .., k$$
$$h_j(X^j_{(.)}) = 0 \ , \ j = 1, .., k$$

As a result, we consider exploring the search space by applying the mutation and recombination operators independently to each subvector $X^l_{(.)}$.

5.3 Handling inequality constraints

We are now going to reconsider the DE algorithm described in Section (4.3) focussing first on the mutation operator. Because our Target vector decomposes into k subvectors, we can also decompose the Donor vector together with the Base vector into k subvectors, getting

$$V_l = \hat{V}_l + F_l \sum_{j=1} \big(X^l_{r_{2j-1},G} - X^l_{r_{2j},G} \big) \text{ for } l = 1, .., k$$

where F_l is the mutation factor for the lth subvector. In the case of inequality constraints applied to the l-subvector, the Base subvector is defined as

$$\hat{V}_l = \lambda X^l_{best,G} + (1-\lambda) X^l_{r_1,G}$$

with F_l taking values in $[0,2]$. Then, we use the recombination operator in the subvectors, getting each element of the Trial vector to satisfy

$$\begin{aligned} U^l_{i,G}(j) &= V^l_{i,G}(j) \text{ for } j = n_r \mod dim, (n_r+1) \mod dim, ..., (n_r+L-1) \mod dim \\ &= X^l_{i,G}(j) \text{ for all other } j \in [1,..,n] \end{aligned}$$

where $dim = n$

5.4 Handling equality constraints

5.4.1 The Mutation operators

In our optimisation problem we have to handle two different types of constraints. In the first type, we are given the subvector $X^l_{(.)}$ of size n where $l \in [1,k]$ with the constraint $\sum_{i=1}^n X^l_{(.)}(i) = \delta$. We let p be a random variable taking values in $[0,1]$ and consider two integers $i \neq j$ when the the ith component is selected for mutation such that

$$\begin{aligned} \overline{X}^l_{(.)}(i) &= pX^l_{(.)}(i) \\ \overline{X}^l_{(.)}(j) &= (1-p)X^l_{(.)}(i) + X^l_{(.)}(j) \end{aligned}$$

In the second type, if we let $m \in [1,k]$, set $m \neq l$ and consider the subvectors X^l and X^m together with the constraint $\sum_{i=1}^n X^l(i)X^m(i) = \delta$, then for $i \neq j$ we get

$$\begin{aligned} \overline{X}^m_{(.)}(i) &= pX^m_{(.)}(i) \\ \overline{X}^m_{(.)}(j) &= (1-p)\frac{X^l_{(.)}(i)X^m_{(.)}(i)}{X^l_{(.)}(j)} + X^m_{(.)}(j) \end{aligned}$$

5.4.2 The recombination operators

Given the first type of equality constraints, we consider two parent subvectors X^l_1 and X^l_2 and want to generate the offspring X^l_3 such that it satisfies the constraint. It can be done with the arithmetical crossover by letting X^l_3 be a linear combination of its two parents

$$X^l_3(i) = \alpha X^l_1(i) + (1-\alpha)X^l_2(i)$$

for $\alpha \in [0,1]$ since

$$\sum_{i=1}^n X^l_3(i) = \alpha\delta + (1-\alpha)\delta = \delta$$

In the second type, if we let $m \in [1,k]$, set $m \neq l$ and consider the subvectors $X^l_{(.)}$ and $X^m_{(.)}$ together with the constraint $\sum_{i=1}^n X^l_{(.)}(i)X^m_{(.)}(i) = \delta$ we can reproduce the above procedure, getting

$$X^l_3(i)X^m_3(i) = \alpha X^l_1(i)X^m_1(i) + (1-\alpha)X^l_2(i)X^m_2(i)$$

and the sum becomes

$$\sum_{i=1}^n X^l_3(i)X^m_3(i) = \alpha\delta + (1-\alpha)\delta = \delta$$

Therefore, given the element $X_3^l(i)$ we can construct $X_3^m(i)$ such that the constraint is satisfied

$$X_3^m(i) = \frac{\alpha X_1^l(i) X_1^m(i) + (1-\alpha) X_2^l(i) X_2^m(i)}{X_3^l(i)}$$

In the DE algorithm, it is equivalent to setting the mutation factor to zero $F_l = 0$, forcing the Donor subvector to be generated sole with crossover operator. In the first type of equality constraints, the Base subvector satisfies

$$\hat{V}_l = \alpha X_{s_1,G}^l + (1-\alpha) X_{s_2,G}^l$$

where s_1 and s_2 are integers chosen randomly from the interval $[1, NP]$ and different from the running index. In the second type of equality constraints, the Base subvector satisfies for each element i the relation

$$\hat{V}_m(i) = \frac{\alpha X_{s_1,G}^l(i) X_{s_1,G}^m(i) + (1-\alpha) X_{s_2,G}^l(i) X_{s_2,G}^m(i)}{\hat{V}_l(i)} \ , \ i = 1, .., n$$

where $\hat{V}_l$ is already computed.

5.5 The proposed algorithm

Using the improvements in Section (4.4) together with a particular mutation operator and our combined way of handling the equality and inequality constraints, the pseudo code for the DE algorithm with constraints becomes

```
Begin
 G = 0  and  Age_i,G = 0 ∀ i, i = 0,..,NP − 1
Create a random initial population  X_i,G ∀ i, i = 0,..,NP−1  from feasible and infeasible vectors
Evaluate  f(X_i,G) ∀ i, i = 0,..,NP − 1
while  niter < max_iter  and  G < Gmax  do
    fmin_old = fmin
   for  i = 0  to  NP − 1  do
      for  k = 1  to  N_K  do
         Loop subvectors for  l = 1,..,k
         if equality constraint then
            Do specific mutation and recombination
            Get  U_i,l,G(j)
         else if inequality constraint then
        Select randomly  r_1 ≠ r_2 ≠ r_3 ≠ i    j_r = U(1,N)
        for  j = 1  to  n  do
           if  U(0,1) < CR  or  j = j_r  then
               U_i,l,G(j) = X_r_3,l,G(j) + F_l (X_r_1,l,G(j) − X_r_2,l,G(j))
               else
               U_i,l,G(j) = X_i,l,G(j)
               end if
            end for
         end if
         end loop
         if  k > 1  then
            if  U_i,G(j)  is better than  U_i_best,G(j)  based on five selection criteria then
                U_i_best,G(j) = U_i,G(j)
         else
```

```
            U_i_best, G(j) = U_i, G(j)
        end of for
        Apply selection criterions :
        if U_i_best, G is better than X_i, G based on five selection criteria then
            X_i, G + 1 = X_i, G = U_i_best, G
        else
            if Age_i, G < N_A or i = i_best then
                X_i, G + 1 = X_i, G
            else
                Select randomly r_4 ≠ i
                X_i, G + 1 = X_r_4, G
                Age_i, G = 0
            end if
        end if

        if fmin > f(X_i, G)
            fmin = f(X_i, G)
            i_best = i
        end if
    end for
    Apply convergence criterions :
    if fmin_old − f_min < precision
        niter = niter + 1
    else
        niter = 0
    end if
     G = G + 1
end while
End
```

5.6 The results

We proposed in Section (3.2) a model producing an arbitrage-free implied volatility surface both in time and in space and discussed in Section (4.1) the calibration problem together with a global measure of fit for estimating the model parameters. More precisely, we considered in Equqtion (4.7) a weighted least square estimator (WLSE) as our prediction error method. That is, after calibration of the model to market prices we obtain the optimised objective function $\mathcal{J}(\Psi^*)$ with optimal model parameter Ψ^*. We are now going to discuss the in-sample performance of the model using the root mean square error (RMSE) defined by

$$RMSE = \sqrt{\sum_{i=1}^{n} w_i \big(P_t(T_i, K_i) - C_t(T_i, K_i; \Psi^*)\big)^2}$$

where the difference between market price and model price is the prediction error. To measure the effectiveness of our algorithm, we test it against a quasi-Newton method called Broyden-Fletcher-Goldfarb -Shanno (BFGS) with penalty function (see Press et al in [PTVF92]) and a DE algorithm with feasibility rules where equality constraints are transformed into inequality constraints (see Bloch in [Blo10]). Tests are performed on the Nikkei 225 index over the last two years on a monthly basis. For illustration purpose, we will only reproduce a single day, that is, the evaluation day is the 21st of December 2009 with index value being 10197.81 yen. The calibration of the model to market prices is performed on 8 maturities such that for all strikes less than

the forward price we use put option while for all strikes above the forward price we use call options. The maturities are 2010/1/8 with strikes $\{8500, 8750, 9000, 9250, 9491, 9740, 10000, 10250, 10490, 10750, 11000, 11250\}$, 2010/2/12 with strikes $\{7000, 7250, 7500, 7550, 8000, 8250, 8500, 8750, 9000, 9241, 9500, 9750, 10000, 10039, 10250, 10500, 10750, 11000, 11250, 11500, 11750, 12000, 12250\}$, 2010/3/12 with strikes $\{6000, 6500, 7000, 7500, 7750, 8000, 8250, 8492, 8500, 8750, 9000, 9250, 9500, 9750, 10000, 10250, 10490, 10500, 10750, 11000, 11239, 11250, 11489, 11750, 12000, 12250, 12500\}$, 2010/4/9 with strikes $\{5000, 5500, 6000, 6500, 7000, 7500, 8000, 8500, 9000, 9500, 9990, 10000, 10490, 11000, 11500, 12000, 12500, 13000\}$, 2010/6/11 with strikes $\{4500, 5000, 6500, 7493, 7992, 8492, 9491, 11500, 12000, 12500, 13000\}$, 2010/9/10 with strikes $\{5000, 5500, 6000\}$, 2010/12/10 with strikes $\{4500, 5000, 6000\}$, 2011/6/10 with strikes $\{10500, 11988\}$. For these maturities and strikes, the mid-market prices range from 2.5 yen to 585 yen, so that the WLSE and the RMSE are expressed in yen. However, following market practice, we express the WLSE in volatility by replacing the model and market prices in Equqtion (4.7) with their implied volatility. We generate 50 independent trials and compute the optimised objective function, displaying in Table (1) the best and worst values together with the sample mean and sample standard deviation.

	BFGS	**DE**	**Modified DE**
Best	$3.932\ 10^{-3}$	$3.483\ 10^{-5}$	$2.716\ 10^{-6}$
Worst	-	0.013547	$6.139\ 10^{-4}$
Mean	-	0.00144455	$3.247\ 10^{-5}$
SdDev	-	0.0034326	$4.521\ 10^{-4}$

Table 1: WLSE for the three optimisation methods expressed in volatility

The proposed algorithm outperforms the two other methods with the lowest objective function on average and provide a much lower standard deviation when compared with the other DE algorithm. The RMSE results expressed in price for the three optimisation methods in the best case are given in Table (2).

	BFGS	**DE**	**Modified DE**
RMSE	20.76	18.93	15.35

Table 2: RMSE for the three optimisation methods expressed in price

In all the tests performed, our modified algorithm returned the smallest prediction errors with smallest weighted differences between market prices and model prices. Hence, our modified approach on handling equality constraints with special operators can find better optimums than transforming them into inequalities and using the feasibility rules, improving calibration to market prices. As a result, market prices being correctly matched, our model pricer can be used more pertinently to quote prices not directly visible in the market and to generate local volatility.

6 Conclusions and future work

To mark options not directly visible in the market as well as to compute a proper deterministic local volatility for pricing exotic options, we proposed a reliable implied volatility surface without arbitrage both in space and in time. We considered a parametric mixture of shifted lognormal densities under constraints and used a Differential Evolution algorithm to calibrate the model's parameters to a finite set of option prices. Our improved algorithm focused on handling constraints in a simple and efficient way by taking advantage of the specific structure of our objective function and used special operators to help satisfy the equality constraints together with feasibility rules to handle the inequality constraints. That is, dividing the population vector

into subvectors, we applied the mutation and recombination operators independently to each subvector followed by a global selection method. Finally, we tested the algorithm on real market data and showed that, compared to other DE method, our approach for handling constraints led on average to better optimums with smaller standard deviations. One possibility for potential future work would be to handle both equality and inequality constraints independently from the model pricer in a more general way. That would allow for the choice of other model pricers to be considered.

Annexes

A Enforcing constraints on parameters

We are presenting a simple way of enforcing the constraints on model parameters given in Equation (3.5). We first take the absolute value of the parameters a_i^0 for all i and consider the normalisation constraint $\sum_{i=1}^n a_i^0 = 1$. We set $sum_1 = \sum_{i=1}^n a_i^0$ and create a new vector with element $\overline{a}_i^0 = \frac{a_i^0}{sum_1}$ so that the new constraint $\sum_{i=1}^n \overline{a}_i^0 = 1$ is satisfied. We then consider the constraint $\sum_{i=1}^n a_i^0 \mu_i^0 = 0$ together with $\mu_i^0 \geq -1$. Again, we set $sum_2 = \sum_{i=1}^n a_i^0 |\mu_i^0|$ and consider the modified parameter

$$\overline{\mu}_i^0 = -1 + \frac{|\mu_i^0|}{sum_2}$$

and since $sum_2 > 0$ and $\sum_{i=1}^n a_i^0 = 1$ then the new constraint

$$\sum_{i=1}^n a_i^0 \overline{\mu}_i^0 = 0$$

is satisfied. Note, when $sum_2 \to 0$ the parameter $\overline{\mu}_i^0$ is not defined but in that case $\sum_{i=1}^n a_i^0 \mu_i^0 \approx 0$. Therefore, when $sum_2 < \epsilon$ for a small enough parameter ϵ we directly assume that the constraint is satisfied.

References

[Ale01] C. Alexander, *Market models - a guide to financial data analysis.*, John Wiley and Sons, New York, (2001).

[BaSu07] R. Balamurugan, S. Subramanian, *Self-adaptive differential evolution based power economic dispatch of generators with valve-point effects and multiple fuel options.*, International Journal of Computer Science and Engineering, Winter (2007).

[BR02] S. Baude, CH. Roubinet, *GRM smile model for the pricing of regular and exotic options on equities.*, Working Paper, GRM, Credit Lyonnais, October (2002).

[BDK08] S. Benaim, M. Dodgson, D. Kainth, *An arbitrage-free method for smile extrapolation.*, Working Paper, QuaRC, Royal Bank of Scotland, (2008).

[BC04] S. Ben Hamida, R. Cont, *Recovering volatility from option prices by evolutionary optimization.*, Working Paper, CMAP, (2004).

[BS73] F. Black, and M. Scholes, *The pricing of options and corporate liabilities*, Journal of Political Economics, **81**, 637-659 (1973).

[Blo09] D. Bloch, *A note on calibration of Markov processes.*, Working Paper, University of Paris 6, April, (2009).

[Blo10] D. Bloch, *A practical guide to implied and local volatility.*, Working Paper, University of Paris 6, January, (2010).

[BGS03] R. Bos, A. Gairat and A. Shepeleva, *Dealing with discrete dividends.*, Risk, January, p109-112 (2003).

[Coe99] C.A. Coello Coello, *A comprehensive survey of evolutionary-based multiobjective optimization techniques.*, Knowledge and Information Systems, An International Journal, **1** (3), p269-p308, (1999).

[Coe00] C.A. Coello Coello, *Constraint-handling using an evolutionary multiobjective optimization technique.*, Civil Engineering and Environmental Systems, Vol. **17**, p319-p346, (2000).

[CoeMez02] C.A. Coello Coello, E. Mezura-Montes, *Constraint-handling in genetic algorithms through the use of dominance-based tournament selection.*, Advanced Engineering Informatic, Vol. **16**, p193-p203, (2002).

[CoeMez03] C.A. Coello Coello, E. Mezura-Montes, *Increasing successful offspring and diversity in differential evolution for engineering design.*, Advanced Engineering Informatic, Vol. **16**, p193-p203, (2003).

[CT02] R. Cont, P. Tankov, *Calibration of jump-diffusion option-pricing models : a robust non-parametric approach*, CMAP, **42**, September (2002).

[CC88] C.E. Curtis, and G.L. Carriker, *Estimating implied volatility directly from "nearest-to-the-money" commodity option premiums.*, Working Paper 081588, Clemson University, (1988).

[DHS06] T. Daglish, J. Hull and W. Suo, *Volatility surfaces : theory, rules of thumb and empirical evidence*, Working Paper, August (2006).

[Dar82] C.R. Darwin, *The variation of animals and plants under domestication.*, Murray, London, second edition (1882).

[Deb00] K. Deb, *An efficient constraint handling method for genetic algorithms.*, Computer Methods in Applied Mechanics and Engineering, **186** (2/4), 311-338 (2000).

[Du94] B. Dupire, *Pricing with a smile*, Risk, **7**, pp. 18-20, (1994).

[Fog66] L.J. Fogel, *Artificial intelligence through simulated evolution.*, John Wiley, New York, (1966).

[Fri02] V. Frishling, *A discrete question.*, Risk, January, p115-116 (2002).

[HKLW02] P.S. Hagan, D. Kumar, A.S. Lesniewski and D.E. Woodward, *Managing smile risk.*, Wilmott Magazine, September, 84-108 (2002).

[HK79] J.M. Harrison, D. Kreps, *Martingale and arbitrage in multiperiods securities markets.*, Journal of Economic Theory, **20**, 381-408 (1979).

[Har68] J. Hart, *Computer approximations.*, Wiley, Algorithm 5666 for the error function, (1968).

[Hol62] J.H. Holland, *Outline for a logical theory of adaptive systems.*, Journal of the Association for Computing Machinery, **9**, pp 297-314 (62).

[Hol75] J.H. Holland, *Adaptation in natural and artificial systems.*, University of Michigan Press, Ann Arbor, (1975).

[LandaCoe06] R. Landa Becerra, C.A. Coello Coello *Cultured differential evolution for constrained optimization.*, Computer Methods in Applied Mechanisand Engineering, **195**, Jully, pp 4303 - 4322 (2006).

[LeguizamonCoe09] G. Leguizamon, C.A. Coello Coello *Boundary search for constrained numerical optimization problems with an algorithm inspired by the ant colony metaphor.*, IEEE Transactions on Evolutionary Computation, **13**, No2, April, pp 350 - 368 (2009).

[Lu90] E. Luede, *Optmization of circuits with a large number of parameters*, Archiv f. Elektr. u. Uebertr., Band (44), Heft (2), pp 131-138, (1990).

[Mal97] A. Malz, *Estimating the probability distribution of the future exchange rate from option prices.*, Journal of Derivatives, Winter, pp 18 - 36, (1997).

[MezCoeTun04] E. Mezura-Montes, C.A. Coello Coello and E.I. Tun-Morales *Simple feassibility rules and differential evolution for constrained optimization.*, Third Mexican International Conference on Artificial Intelligence, MICAI, Lecture Notes in Artificial Intelligence, p707-p716, (2004).

[MezCoe04] E. Mezura-Montes, C.A. Coello Coello *A study of mechanisms to handle constraints in evolutionary algorithms.*, Workshop at the Genetic and Evolutionary Computation Conference, Seattle, Washington, ISGEC, (2004).

[MezVelCoe06] E. Mezura-Montes, J. Velazquez-Reyes and C.A. Coello Coello *Modified differential evolution for constrained optimization.*, IEEE Congress on Evolutionary Computation, IEEE Press, p332-p339, (2006).

[Mi95] Z. Michalewicz, *Gnetic algorithms, numerical optimization and constraints.*, L. Eshelman, ed., Proceeding of the Sixth International Conference on Genetic Algorithms, San Mateo, 151-158, (1995).

[MNM96] Z. Michalewicz, G. Nazhiyath and M. Michalewicz *A note on usefulness of geometrical crossover for numerical optimization problems.*, Proc. 5th Annu. Conf. Evolut. Prog., L.J. Fogel, P.J. Angeline and T. Back, Eds., Cambridge, MA, pp 305-311 (1996).

[OBBFJL07] M. Overhaus, A. Bermudez, H. Buehler, A. Ferraris, C. Jordinson and A. Lamnouar, *Equity hybrid derivatives*, Wiley Finance, (2007).

[OSF05] A. Oyama, K. Shimoyama and K. Fujii, *New constraint-handling method for multi-objective multi-constraint evolutionary optimization and its application to space plane design.*, Evolutionary and Deterministic Methods for Design, (2005).

[PTVF92] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery, *Numerical recipes, 2nd ed. Cambridge*, Cambridge University Press, (1992).

[RC04] R. Rebonato, M.T. Cardoso, *Unconstrained fitting of implied volatility surfaces using a mixture of normals.*, Working Paper, QUARC and Oxford University, July 6 (2004).

[SanCoe05] L.V. Santana-Quintero, C.A. Coello Coello, *An algorithm based on differential evolution for multi-objective problems.*, International Journal of Computational Intelligence Research, **1**, ISSN 0973-1873, pp 151 - 169 (2005).

[StPr95] R. Storn and K. Price, *Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces.*, International Computer Science Institute, Berkeley, **TR-95-012**, (1995).

[Sto96] R. Storn, *System design by constraint adaptation and differential evolution.*, International Computer Science Institute, Berkeley, **TR-96-039**, (1996).

[Tan05] P. Tankov, *Calibration de modeles et couverture de produits derives.*, Working Paper, Universite Paris VII, (2005).

[Wri91] A.H. Wright, *Genetic algorithms for real parameter optimization.*, In Foundation of Genetic Algorithms, ed. G. Rawlins, First Workshop on the Foundation of Gen. Alg. and Classified Systems, Los Altos, CA, pp 205-218 (1991).

[Woo10] D. Wood, *Future options.*, Risk Magazine, March, pp 23-25 (2010).