Large Step Local Volatility

Romano Trabalzini
CID:00500474

Imperial college of Science Technology and Medicine
Department of Mathematics

September 19, 2011

"We have not succeeded in answering all our problems. The answers we have found only serve to raise a whole set of new questions. In some ways we feel we are as confused as ever, but we believe we are confused on a higher level and about more important things"

from Berndt Øksendal, Stochastic Differential Equations, [30]

Acknowledgment

I am deeply indebted to Royal Bank of Scotland and the team of Hybrid Quantitative Analytics for giving me the opportunity to work there this summer in an incredibly friendly environment where I had a pleasant and rewarding experience. I learned a new way of looking at things, which will stay with me: it feels like wearing new, more powerful spectacles. I would like to thank in particular William McGhee, Stephen Smith and Hicham Fadil for the patience in answering my questions and for the support shown during all these 12 weeks of internship. I owe a debt of gratitude to Professor Mark H. Davis for his advice, support and patience in giving up his time to help me complete this dissertation.

Contents

Introduction. Problem Statement and Objectives

In this work, we propose to elaborate on analytical and numerical techniques to reconstruct an accurate and stable local volatility function.

The topic does not stem from a purely academic urgency: the industry-driven problem that this thesis intends to address is the lack of any viable implementation of local volatility function when pricing, in a MonteCarlo setup, long-dated FX trades. The issues at stake are, on the one hand, the strong path-dependency of those: hence an accurate local volatility function is paramount; on the other, the large time step that a long-dated product imposes (say six months over 10 to 20 years): hence the difficulty of employing traditional discretization schemes becomes manifest.

It is no secret that the Black-Scholes constant volatility assumption is at most inappropriate in valuing path-dependent products, where averages may come into the fore and multiple strikes, which result in multiple implied volatilities, are involved. The concept of Smile will be defined and a brief literature review will be given on how the industry and the academia have reacted to the theoretical challenges it creates.

Once the simplifying assumption of constant volatility is removed, we will first of all pay attention to different analytical schemes to handle the trade-off between a better fit of the smile itself and an acceptable degradation in performance that such a procedure imposes on any computational strategy. In particular: we will employ a SABR model of market implied volatility, known for its simplicity and accuracy and we will target the reconstruction of an analytically and numerically stable Local Volatility Function. On the one hand we will use Dupire's formula, inserting the SABR implied vol into the Black-Scholes option pricing function; on the other hand, we will derive it from the market implied volatility itself, after a few analytical rearrangements. A detailed analytical derivation of both ways of proceeding will be given along with a discussion of each other's benefits.

A recurrent thread of our discussion will be the apparent gap in the industry's model space across different asset classes: on the one hand, in the realm of *Short Dated FX/Equity Market*, the tool of choice is usually PDE modeling through Finite Difference Schemes where Local Volatility is modeled through small steps. Here the emphasis lies on modeling a continuous density over a short time horizon. On the other hand, in *Rates/Hybrids Domain*, the relevance of (stochastic) interest rates increases quite substantially the dimensionality of the problems and PDE schemes, even if ADI/ADE, do not stand comparison with MonteCarlo. Here what happens is that, in the context of long dated products (say 20 years), it is the coupon density only that can reasonably be modeled, whereas a finer time grid (finer than 3 to 6 months) would impose too big a computational burden on the MonteCarlo sampler. Here the Local Volatility is modeled through large time steps.

Our work will therefore approach a Local Volatility reconstruction from these two computational perspectives at the same time. Multiple analytical techniques will be devised, analyzed and successively deployed in both a (Crank-Nicolson) Finite Difference scheme and in a MonteCarlo setup to better match the curvature of the surface. In particular, polynomial fits and extrapolation schemes will be discussed for FDM schemes, while log-space Euler and predictor-corrector schemes will be explored and deployed in the MC setup.

All the above infrastructure, focusing of course on the Local Volatility Function, will finally be deployed into the very computational bottleneck, i.e. a large step MonteCarlo framework that uses a non-linear solver to perform calibration during simulation. In the bank's current scenario, there is no Local Volatility information feeding into the solver: the volatility function is reconstructed by the solver itself, in fact discovering the process while moving forward. We propose to use our reconstructed Local Volatility Function above to give the solver a better prior, thus potentially reducing the number of iterations that the Steepest Descent is forced to perform without any sensible first guess.

What happens at the moment is that the solver, by construction, reprices a vanilla contract via minimizing the squared error of difference between analytical and MonteCarlo price: but that is of no warranty for any adequate fit of the transition density. On the contrary it may happen that the local volatility vector that the solver produces at each time step is too noisy to be of any practical value for the pricing of any exotic we may be interested in.

As regards existing literature, the problem is the following: the finite difference scheme for Local Volatility -although traditionally considered an ill-posed problem- has received some attention, in particular as regards Implicit Schemes. Not so much the Large Step Local Volatility computation in a path-driven sampling framework. The results and tools we will talk about will venture into an unexplored terrain.

Beyond the actual results achieved, we would like to drive home another point here: Feynman-Kač representation, as known, expresses the solution of a parabolic PDE as the expected value of a certain functional of a Brownian motion. This representation, striking as it is *analytically*, is given even more significance by the fact that we can *numerically* recover essentially the same results across Finite Difference schemes, i.e. strategies to approximate the differential operator of PDE world, and MonteCarlo schemes, i.e. strategies to approximate the expectation operator of Stochastic Process world. This is an hint to a more fundamental idea, which from time-to-time we will try to unearth in this or that computational strategy: the oneness of the diffusive phenomena we look at from analytically (and hence numerically) different viewpoints. We are effectively contemplating (and acting upon) the same substance from different perspectives.

Finally, a remark: all the code that corresponds to the above analytics has been implemented from scratch in industry-standard C++; when needed, easily available open source libraries have been used in code base to avoid reinventing the wheel.

1 Modeling of Volatility

1.1 Introduction and Scope

In this chapter we will first of all explore the analytics and the actual calibration of the Market Implied Volatility Model we have adopted in this work, i.e. the SABR engine. Successively, we will work on deriving a Local Volatility Function from either Option Prices in the spirit of Dupire or from Market Implied Volatilities. The reason we used SABR is purely because of its analytical tractability and its nice closed form solution as given in [31]. We will explore this point in due course. As regards the issue of deriving Local Volatility from either Implied Volatility or Option Prices, it is really a specious dichotomy: in both cases what drives the derivation is a Market Implied Volatility (for us coming from SABR) that in the former case is used directly while in the latter is channeled through a non-linear function, i.e. Black-Scholes. We will see how this essentially uniqueness of structure will dictate a uniqueness of behavior modulo the truncation error implicit in a non-linear function. As regards the above two well-established strategies of deriving local vol, we will give full analytical detail. It is not necessarily the case that those same formulas have an equally widespread use in the industry, but nonetheless their acceptance is common practice (at least among exotics derivatives desks) and we have based our calculations on both those derivations. We will see how, in order to address the same problem, those two strategies yield essentially similar results (according to a metric that we will specify in due course).

1.2 A brief excursus on the Smile

Amongst the many merits of Black-Scholes model and definitely contributing to its early adoption and subsequent massive influence in the financial industry, there is the fact that the price of a derivative contract is obtained via inserting into an analytical formula five parameters, only one of which is unobserved (see for example [24]). The unobserved one is of course volatility, and BS formula assumes it to be a constant. It was soon noticed by academics and practitioners alike that volatility is hardly a constant throughout strike and maturity domain and it was quickly realized that *Volatility clustering*[1] and *Excess Kurtosis*[2] can lead to severe distortions when prices of exotics options are based on the Black-Scholes assumptions (see Fig. 1)[3].

Thus, market practitioners started to analyze

- the sensitivity of (market) implied volatility to strike level
- the sensitivity of (market) implied volatility to time-to-maturity of at-the-money options

The first concept was given the name of *Volatility Skew*, the second the name of *Volatility Term Structure*. Collectively, the concept of *Volatility Smile* started to identify the peculiar way (market) implied volatility changes across strikes and time-to-maturity.

Early attempts were made to accommodate what was recognized as some form of volatility dynamics. At the beginning, different dynamics than the Black-Scholes' were advocated and found some use: CEV model being an illustrious example of that way of proceeding. In the progress of time *stochastic volatility* models started to come out in the literature, thus recognizing volatility itself as a random variable. Possibly the first stochastic volatility model was Hull & White's [19], whereas Heston model followed closely [17]. For a through recognition, we point the reader to [34]. In the words of Gatheral [15], whose neat and informed book will guide us through some of the present work,

[1]Volatility Clustering corresponds to the empirical remark that large changes in prices have a tendency to cluster togheter

[2]In other words, fatter tails than a normal distribution

[3]Image and estimation from `http://blog.gillerinvestments.com`

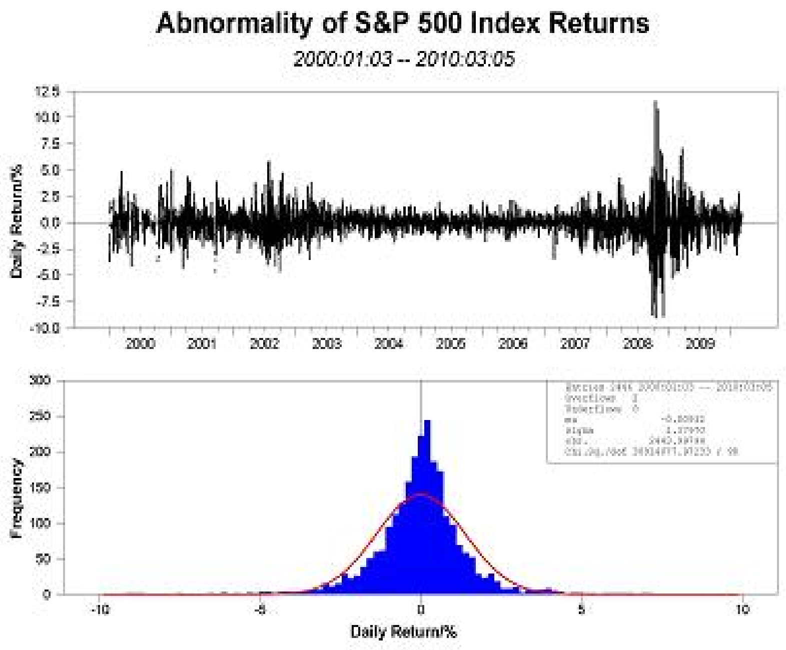

Figure 1: Vol Clustering and Excess Kurtosis

> "Fat tails and the high central peak are characteristics of mixture of distributions with different variances. This motivates us to model variance as a random variable. The volatility clustering feature implies that volatility (or variance) is auto-correlated. In the model, this is a consequence of mean-reversion of volatility."

At the same time, and here the story starts to become more relevant for the present work, the concept of Local Volatility made its entrance in literature. It was the pioneering work of Breeden and Litzenberger [3] that made available to a broader spectrum of academics the (practitioners') intuition that the risk neutral density[4] could be derived from the market prices of European options. But it was only a few years later that Dupire [13] and Derman and Kani [9] realized that there was a unique diffusion capable of accommodating these distributions.

1.3 SABR: a model for Market Implied Volatility

1.3.1 Mechanics of reproducing the surface

We have noted above that he industry was quick to drop the Black-Scholes model assumption of constant volatility in favor of alternative dynamics. In order to fit the implied volatility, i.e. the parameter that has to be put into the BS formula to recover the market price of an option, Hagan and coauthors [31] devised a parsimonious model whose parameters could immediately relate to observable market volatilities[5]. The SABR (Stochastic Alpha, Beta, Rho) model describes the Forward Price F and the Instantaneous Volatility α by the following system of SDEs

[4]We will be working throughout in a pricing setup, so we will be using exclusively a martingale measure. No reference will be made to the physical (objective/statistical) measure.

[5]In a sense that we will make clear below

$$\begin{aligned} dF(t) &= \alpha F^{\beta}(t) dW^{f}(t), \quad F(0) = F, \\ d\alpha(t) &= \nu \alpha(t) dW^{\alpha}(t), \quad \alpha(0) = \alpha, \end{aligned}$$

where ρ in $d < W^{\alpha}, W^{f} >= \rho dt$ is the correlation of the two Wiener processes, $\nu > 0$ is the volatility of volatility parameter and β is a leverage coefficient. Here is how they relate to observable market volatilities:

1. $\alpha(t)$ can be seen as a measure of overall At-The-Money (ATM) Forward volatility;
2. ν is a measure of convexity[6], i.e. the stochastic nature of $\alpha(t)$;
3. ρ and β both measures of the skew
   - $\beta = 1 \Rightarrow$ *pure stochastic volatility model* with log-normal dynamics
   - $\beta \neq 1 \Rightarrow dF_t = \alpha F_t^{\beta-1} F_t dW_t^{f}$, i.e. *local-stochastic volatility model*
   - $\rho < 0 \Rightarrow$ negative skew
   - $\rho > 0 \Rightarrow$ inverse skew
   - $\rho = 0$ (given $\beta = 1$) $\Rightarrow$ symmetric volatility smile

As it is easily seen, the SABR model uses CEV-dynamics[7] for the Forward Price and log-normal dynamics for the instantaneous volatility $\alpha(t)$.

Based on the above, and for given Strike K and maturity T, the SABR engine returns an approximation of the Black-Scholes Implied Volatility. This is the formula that, when adequately calibrated, governs the Market Implied Volatility throughout all our exercise:

$$\begin{aligned} \sigma_{impl} = & \frac{\alpha}{(FK)^{\frac{1-\beta}{2}} \left(1 + \frac{(1-\beta)^2}{24} \ln^2 F/K + \frac{(1-\beta)^4}{1920} \ln^4 F/K\right)} \left(\frac{z}{x(z)}\right) \times \\ & \left(1 + \left(\frac{(1-\beta)^2}{24} \frac{\alpha^2}{(FK)^{1-\beta}} + \frac{1}{4} \frac{\rho\beta\nu\alpha}{(FK)^{(1-\beta)/2}} + \frac{2 - 3\rho^2}{24} \nu^2\right) T\right) \end{aligned} \tag{1}$$

where we have used

$$z = \frac{\nu}{\alpha} (FK)^{(1-\beta)/2} \ln F/K, \quad x = \ln\left(\frac{\sqrt{1 - 2\rho z + z^2} + z - \rho}{1 - \rho}\right)$$

A point often forgotten in many implementations is the importance of the requirement $\nu^2 T << 1$. That results in a term structure of the vol of vol $\nu(T)$ parameter which decreases in T (See Fig 2). The reason why this requirement has to be observed is the asymptotic nature of the Hagan formula above and the fact that in SABR model volatility does not mean revert.

Two particular cases are of relevance in our set-up:

(a) the log-normal model ($\beta = 1$) case

[6]Recall that a function $\phi : \mathbb{R} \to \mathbb{R}$ is said to be convex when, $\forall \lambda \in [0, 1]$, x,y $\in \mathbb{R}$,

$$\phi(\lambda x + (1 - \lambda)y) \leq \lambda\phi(x) + (1 - \lambda)\phi(y)$$

Modeling of Convexity plays a crucial rôle in finance, Jensen Inequality being one of the analytical tools that surface all the time in stochastic analysis. Moreover, *Convexity Adjustment* is a pervasive concept across different models and asset-classes.

[7] Moving from some early intuitions by Feller [14], subsequent major rework done by Cox and Ross [6] led to the CEV model, where the dynamics evolve according to

$$dS = \mu(S, t)dt + \sigma S^{\beta} dW_t$$

with $\beta \geq 0$. But CEV models are parametric and cannot match the implied volatility surface [8].

(b) the ATM case

In case (a), formula (1) simplifies considerably, as in the one below:

$$\sigma_{impl}(T,K) = \alpha \left(\frac{y}{x(y)}\right) \left(1 + \left(\frac{1}{4}\rho\nu\alpha + \frac{2-3\rho^2}{24}\nu^2\right) T\right)$$

with $y = \frac{\nu}{\alpha} \ln F/K$.

The ATM case above, instead, results in:

$$\sigma_{impl} = \alpha \left(1 + \left(\frac{1}{4}\rho\nu\alpha + \frac{2-3\rho^2}{24}\nu^2\right) T\right)$$

As an additional point, it is worth mentioning that *SABR domain of applicability is practically limited.* We did not explore this issue ourselves since the parameters where given to us, but for some parameterizations, the formula may lead to regions of negative density. This inconsistency is discussed at length in [28].

1.3.2 Calibration to Market Data: the term-structure

Without going into further detail on the mechanics of the SABR model, we can now start talking about the effective choice of parameters for our exercise. In particular, for long dated FX products we are interested in modeling the skew and both β and ρ govern the aforementioned feature of the surface. We can then use this degree of freedom to fix $\beta = 1$ and then delegate to the correlation parameter the modeling of the skew. Other teams in the bank (for instance Interest Rates) are indeed using SABR with $\beta \neq 1$: that corresponds to a peculiar choice of dynamics for the underlying process. But in our case the issue is vastly different: we are not concerned about real dynamics and we have chosen SABR because of its tractability, simplicity of use and well-definiteness (except, as said previously, for regions of negative density).

To sum it up: the market furnishes us with a discrete set of (K,T)-Implied Volatilities. We want to use equation (1) to retrieve from the SABR engine a continuum of Implied Volatilities. The missing step is of course the actual fitting of SABR parameters to the existing Market Volatility Surface. That will not impinge on us here, since SABR parameters calibration has been left outside the scope of our work and we have indeed used a precomputed term structure.

Table (1) contains the reference set of parameters for our SABR term structure.

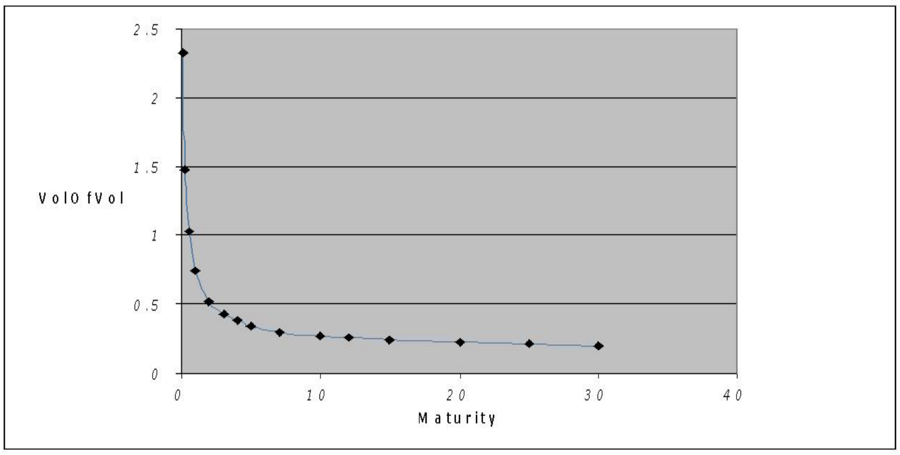

Figure 2: USDJPY Vol of Vol in SABR Term Structure (as of 08 Jul 2011)

It is interesting to notice that this term structure, which corresponds to US Dollar vs. Japan Yen, presents a pronounced negative correlation. The reason why we chose

USDJPY	α	Vol of Vol	ρ
1m	8.25%	232.80%	-8.66%
3m	9.29%	147.91%	-13.89%
6m	10.47%	103.48%	-17.80%
1y	11.94%	74.74%	-19.88%
2y	13.10%	52.02%	-23.86%
3y	13.80%	43.30%	-25.51%
4y	14.30%	38.47%	-27.93%
5y	14.90%	34.41%	-32.31%
7y	16.50%	29.72%	-38.52%
10y	18.10%	26.70%	-47.53%
12y	18.85%	26.13%	-46.44%
15y	19.90%	24.49%	-45.61%
20y	20.30%	22.60%	-44.85%
25y	20.55%	21.26%	-44.74%
30y	20.80%	19.92%	-44.62%

Table 1: SABR Term Structure - USDJPY as of 08 Jul 2011

those is essentially business-driven: Dollar Yen is the main driver of the Long Dated FX space we are in[8]. As stated above, the pronounced negative correlation results in negative skew. While the actual mechanics of the calibration, essentially an L^2-minimization of the differences between model and implied volatilities, are orthogonal to the main topic of this project, a more elaborate discussion needs to be made w.r.t. the interpolation used to fit a continuous function between the above parameters: function whose independent variable is the time horizon we are at. We will see in later chapters how the Local Volatility Function, however derived, needs a $\mathbb{C}^1$ market volatility function to continuously furnish the calculation with volatility values. The decision about how to interpolate the discrete set of calibration parameters for SABR term structure is quite important: following a pattern that seems quite usual in the domain of term structure modeling, we did not choose a *global fit*, like a cubic spline : common perception is that what happens at short time horizon needs not bear relation to what happens at longer maturities. Therefore, we have used a *local fit*, in the form of a Lagrange polynomial. In so doing, we are aware that a growing literature exists on how best to interpolate the SABR parameters but for our problem the terminal density at each time step which is reflected in the SABR parameters does not necessitate any more complex interpolating procedure.

Before going to explore local volatility, a final remark: SABR interpolation is sometimes quite tricky to accomplish and there is still quite a lot of research in this area. We could not go much further into that direction and we leave the topic here. More details on issues and pitfalls can be found in [28].

1.4 Local Volatility

1.4.1 Preliminary Considerations

Upon considering, at the beginning of this chapter, the issues arising from assuming flat volatility within Black-Scholes, we pointed out how academics and practitioners alike started exploring ways to use the information contained in the smile. One very simple approach was to give $\sigma(S, t)$ a particular parametric form. Another approach, the one we will explore

[8]It is worth mentioning that, while in the process of choosing our market calibrated SABR term structure looking for a negative correlation, we experienced that the relation EUR-CHF (Swiss Frank), which has always been one of positive correlation (almost to the point of using the EUR correlation matrices to adjust for missing entries in CHFvs. other currencies correlation matrix), suddenly - around mid July - moved in the realm of negative correlation, because of the market fly to quality.

more closely in the following, consisted in "deducing $\sigma(S,t)$ numerically from the smile"[9]. The attempts to introduce some more sources of randomness to account for the particular structure of volatility opened up new areas of research. In particular, Hull-White original stochastic volatility model [19] or Merton's Jump approach [29] resulted in the introduction of non-diversifiable risks within the model itself. Clearly, losing completeness leads, on the one hand, to the loss of hedging possibility; on the other hand it results in multi-factors finite difference schemes, hence in more complicated discretizations. If we consider the particular structure of Black-Scholes model, though, we can perceive, as Dupire and Derman-Kani, that there is a one-to-one relationship between volatility and price ([13] and [9]). The function can therefore be inverted. Unfortunately, in doing that, we are immediately placed in the realm of inverse problems which have traditionally given rise in literature to a wealth of analytical and numerical techniques to solve the intrinsic ill-posedness (for a recent example, see [1]).

What Dupire (in the continuous setup) and Derman-Kani (in the lattice framework, so in the discrete setup) realized was that, if we are ready to place some restrictions, say if the drift is somehow imposed, the conditional laws should permit us to recover the full diffusion. We cannot expect that the spot actually follows the diffusion we have found: but in a sense we can expect, in Dupire's words, that "the market prices European options *as if* the process was this diffusion" [13].

The difficulties that we will encounter from now on during our exercise in reconstructing a Local Volatility Function by means of some interpolation scheme will result in noisy surfaces; that can be read as a confirmation that our model is in a sense too restrictive. We will explore those issues going forward.

Before starting with derivation of Local Volatility Function from either Option Prices or from Market Implied Volatility, let us fix some standard concepts.

Kolmogorov Backward Equation Given a Probability Transition Density p(s,x;t,y), it can be shown that it satisfies *Kolmogorov Backward Equation*

$$\begin{cases} \frac{\partial}{\partial s}p(s,x;t,y) + \frac{1}{2}\sigma^2(s,x)\frac{\partial^2}{\partial x^2}p(s,x;t,y) + b(s,x)\frac{\partial}{\partial x}p(s,x;t,y) = 0 \\ \lim_{s\to t} p(s,x;t,y)dy = \delta_x(dy) \end{cases} \tag{2a}$$

Kolmogorov Forward Equation Given a Probability Transition Density p(s,x;t,y), it can be shown that it satisfies *Kolmogorov Forward Equation*

$$\begin{cases} \frac{\partial}{\partial t}p(s,x;t,y) - \frac{1}{2}\frac{\partial^2}{\partial y^2}\left(\sigma^2(t,y)p(s,x;t,y)\right) + \frac{\partial}{\partial y}\left(b(t,y)p(s,x;t,y)\right) = 0 \\ \lim_{t\to s} p(s,x;t,y)dy = \delta_x(dy) \end{cases} \tag{3a}$$

where δ_x is Dirac measure[9] at x

1.4.2 First Derivation: Local Volatility from Option Prices

Let us assume that the risk neutral dynamics for underlying S is given by the following SDE:

$$dS(t) = (r(t) - d(t))S(t)dt + \sigma_{loc}(S,t)S(t)dW(t), \quad S(0) = S_0 \tag{4}$$

where σ_{loc} is local volatility (as function of asset level and time) while (r(T) - d(T)) is drift, expressed (depending on contexts) as either difference of domestic and foreign component

[9] Recall Dirac Delta Function:

$$\delta(x) = \begin{cases} 0, & x \neq 0 \\ \infty, & x = 0 \end{cases}$$

where

$$\int_{-\infty}^{\infty} \delta(x)dx = 1$$

or as difference of risk free rate and dividend yield. For us it does not make difference which interpretation to choose. Via no arbitrage conditions, the value of the European Option V(S,t;K,T) with Strike K and Maturity T at any time before T satisfies the following backward parabolic equation[10]

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma_{loc}^2(S,t)S^2\frac{\partial^2 V}{\partial S^2} + (r(t) - d(t))S\frac{\partial V}{\partial S} - r(t)V = 0$$
$$V(S,T;K,T) = max\{\chi(S-K),0\}$$

where

$$\chi = \begin{cases} 1, & if\ call \\ -1, & if\ put \end{cases}$$

By results in previous section about Kolmogorov Equations, the risk neutral transition density $p = p(S,t;y,T)$ associated with the asset price dynamics in (4) satisfies the following Forward Equation, also known in literature as Fokker-Planck Equation:

$$\frac{\partial}{\partial T}p = -\frac{\partial}{\partial y}\left((r(T) - d(T))yp\right) + \frac{\partial^2}{\partial y^2}\left(\frac{1}{2}\sigma^2(y,T)y^2 p\right) \tag{5}$$

Dupire's result can now be enunciated and proved.

Theorem: Dupire's Local Volatility Formula
If the price of the underlying follows SDE in (4) Call option price in terms of Strike K is given by

$$\frac{\partial C}{\partial T} = \frac{\sigma_{loc}^2}{2}K^2\frac{\partial^2 C}{\partial K^2} + (r(T) - d(T))\Big(C - K\frac{\partial C}{\partial K}\Big) \tag{6}$$

where $C := C(S,t;K,T)$ is the value of a European Call option.

Proof
Under standard martingale measure, we can write the time t value of a call as

$$\begin{aligned} C = \mathbb{E}\left[\exp^{-r(T)(T-t)}(S_T - K)^+\right] &= \exp^{-\int_t^T r(\xi)d\xi}\int_0^\infty (y-K)^+ p(S,t;y,T)dy \qquad (7) \\ &= \exp^{-\int_t^T r(\xi)d\xi}\int_K^\infty (y-K)p(S,t;y,T)dy \end{aligned}$$

Differentiating the integral according to formula below:

$$\frac{\partial}{\partial x}\int_0^x f(t,x)dt = f(x,x) + \int_0^x \frac{\partial f(t,x)}{\partial x}dt$$

where the interchange of the two limit processes can be justified by *Monotone Convergence Theorem*, we get

$$\begin{aligned} \frac{\partial}{\partial K}C(S,t;K,T) &= \exp^{-\int_t^T r(\xi)d\xi}\frac{\partial}{\partial K}\int_K^\infty (y-K)p(S,t;y,T)dy \\ &= -\exp^{-\int_t^T r(\xi)d\xi}\frac{\partial}{\partial K}\int_\infty^K (y-K)p(S,t;y,T)dy \\ &= -\exp^{-\int_t^T r(\xi)d\xi}\left((K-K) - \int_\infty^K \frac{\partial}{\partial K}(y-K)p(S,t;y,T)dy\right) \\ &= \exp^{-\int_t^T r(\xi)d\xi}\int_\infty^K p(S,t;y,T)dy \end{aligned}$$

[10]It may be helpful to recall that a second order parabolic equation is said to be backward if, when at the same side of the equation, the derivative w.r.t. time and the second derivative w.r.t. space have same sign; forward when the sign is different. Fokker-Planck equation is obviously forward in time (see [32])

Differentiating again, we obtain

$$\frac{\partial^2}{\partial K^2}C(S,t;K,T) = \exp^{-\int_t^T r(\xi)d\xi} p(S,t;y,T) \tag{8}$$

and hence we recover

$$p(S,t;y,T) = \exp^{\int_t^T r(\xi)d\xi} \frac{\partial^2}{\partial K^2}C(S,t;K,T) \tag{9}$$

The meaning of the above formula is the following: if we are given a collection of market prices for European Calls at different strikes and maturities (collection that we may interpolate to get a continuum of values) we possess automatically the risk-neutral probability density. It remains to be established what should be the scheme to interpolate this discrete collection to a continuum of values: we will talk at length about this issue when building our Local Volatility Functions (see [Chap. 2]).

Differentiating equation (7) w.r.t. T, we get

$$\begin{aligned}\frac{\partial}{\partial T}C(S,t;K,T) &= -r(T)C(S,t;K,T) \\ &\quad + \exp^{-\int_t^T r(\xi)d\xi} \int_K^\infty (y-K)\frac{\partial}{\partial T}p(S,t;y,T)dy = (\star)\end{aligned}$$

where again interchange of integration and differentiation comes from Monotone Convergence. By inserting the Fokker-Planck equation (5) in the above, i.e. by substituting $\frac{\partial p(S,t;y,T)}{\partial T}$ with RHS of (5), we obtain:

$$\begin{aligned}(\star) &= -r(T)C(S,t;K,T) + \exp^{-\int_t^T r(\xi)d\xi} \times \\ &\quad \int_K^\infty (y-K)\left[-\frac{\partial}{\partial y}\left((r(T)-d(T))yp\right) + \frac{\partial^2}{\partial y^2}\left(\frac{1}{2}\sigma^2(y,T)y^2p\right)\right]dy \end{aligned}\tag{10}$$

Second component of LHS in (10), using integration by parts, can be rewritten as:

$$\begin{aligned}\int_K^\infty (y-K)\frac{\partial^2}{\partial y^2}\left(\frac{1}{2}\sigma^2(y,T)y^2p(S,t;y,T)\right)dy &= (y-K)\frac{\partial}{\partial y}\left(\frac{1}{2}\sigma^2(y,T)y^2p(S,t;y,T)\right)\Big|_K^\infty \\ &\quad -\int_K^\infty \frac{\partial}{\partial y}\left(\frac{1}{2}\sigma^2(y,T)y^2p(S,t;y,T)\right)dy \\ &= \frac{1}{2}\sigma^2(K,T)K^2p(S,t;K,T) \\ &= \frac{1}{2}\sigma^2(K,T)K^2\exp^{\int_t^T(r(\xi)d\xi)}\frac{\partial^2}{\partial K^2}C(S,t;K,T)\end{aligned}$$

where last equivalence comes from (9)

Again in second component of LHS in (10), integrating by parts once more, we can rewrite:

$$\begin{aligned}\int_K^\infty (y-K)\frac{\partial}{\partial y}((r(T)-d(T))yp(S,t;y,T))dy &= -(r(T)-d(T))\int_K^\infty yp(S,t;y,T)dy \\ &= -(r(T)-d(T))\left[\int_K^\infty (y-K)p(S,t;y,T)dy + \int_K^\infty Kp(S,t;y,T)dy\right] \\ &= -(r(T)-d(T))\left[\exp^{\int_t^T r(\xi)d\xi}C(S,t;K,T) - K\frac{\partial}{\partial K}(S,t;K,T)\right]\end{aligned}$$

If we insert this last equation in (10) and make the necessary rearrangements, we recover the Dupire equation (6). □

1.4.3 Second Derivation: Local Vol from Implied Volatility

Let us start by reviewing Black's 1976 formula, which is particularly used in FX markets [2]. Let $C_{BS}(K,T,\sigma_{BS})$ be the price of an European Call in the Black-Scholes model where

$$F_{0,T} = S_0 \exp\left(\int_0^T (r_t - d_t)dt\right)$$

is the current Forward Price at T of the underlying. Black's formula gives us the option price as

$$C_{BS}(K,T,\sigma_{BS}) = DF\left(F_{0,T}\Phi(d_+) - K\Phi(d_-)\right)$$

where

$$d_\pm = \frac{\ln(F_{0,T}/K) \pm \frac{1}{2}\sigma^2 T}{\sigma\sqrt{T}}$$

$(F_{0,T}/K) = \frac{1}{(K/F_{0,T})}$ is the inverse of moneyness $(K/F_{0,T})$ and DF is the discount factor.

Now we want to derive Local Volatility from Market Implied Volatility. We will see that, as in Dupire's case, it is the Fokker-Planck equation for transition density that drives the analytics. To accomplish this lengthy derivation, we will proceed like the following:

1. perform a first change of variables, $(K,T) \to (y,T)$, where $y = \ln(K/F_{0,T})$ (Black log-moneyness). After that, rewrite Dupire equation (6) in the new variables;

2. perform a second change of variables, $(y,T) \to (y,w)$, where $w = \sigma^2_{BS}(K,T)T$ After that, rewrite once again Dupire equation (6) in these new variables;

3. Feed Black-Scholes Call formula into Forward Equation (Dupire). Calculate derivatives and rearrange.

Let us then start with the first change, and compute derivatives in the new variables. For (K,T) = $(F_{0,T}\exp(y),T)$, the partial derivatives of C(K,T) = C$((F_{0,T}\exp(y),T)$ become:

$$\begin{aligned}
\frac{\partial C(K,T)}{\partial T} &= \frac{\partial C(y,T)}{\partial T} + \frac{\partial C(y,T)}{\partial y}\frac{\partial y}{\partial T} \\
&= \frac{\partial C(y,T)}{\partial T} - (r(T) - d(T))\frac{\partial C(y,T)}{\partial y} \\
\frac{\partial C(K,T)}{\partial K} &= \frac{\partial C(y,T)}{\partial y}\frac{\partial y}{\partial K} = \frac{1}{K}\frac{\partial C(y,T)}{\partial y} \\
\frac{\partial^2 C(K,T)}{\partial K^2} &= \frac{\partial}{\partial K}\left(\frac{1}{K}\frac{\partial C(y,T)}{\partial y}\right) \\
&= -\frac{1}{K^2}\frac{\partial C(y,T)}{\partial y} + \frac{1}{K}\frac{\partial}{\partial K}\left(\frac{\partial C(y,T)}{\partial y}\right) \\
&= \frac{1}{K}\left(\frac{\partial^2 C(y,T)}{\partial y^2}\frac{\partial y}{\partial K}\right) - \frac{1}{K^2}\frac{\partial C(y,T)}{\partial y} \\
&= \frac{1}{K^2}\left(\frac{\partial^2 C(y,T)}{\partial y^2} - \frac{\partial C(y,T)}{\partial y}\right)
\end{aligned}$$

In the new variables (y,T), Dupire's equation (6), after inserting the equivalences just found, becomes:

$$\frac{\partial C(y,T)}{\partial T} = \frac{\sigma^2(K,T)}{2}\left(\frac{\partial^2 C(y,T)}{\partial y^2} - \frac{\partial C(y,T)}{\partial y}\right) + (r_T - d_T)C(y,T) \tag{11}$$

Denoting by $\sigma_{BS}(K,T)$ the Implied Volatility Surface, we can now perform the above second change of variables, from (y,T) to (y,w) where w = $\sigma^2_{BS}(K,T)T$ is the total Black-Scholes Implied Variance. The call price becomes C(K,T) = C$\Big(F_{0,T}\exp(y), \frac{w}{\sigma^2_{BS}}\Big)$.

Black's formula becomes

$$C(y,w) = F_{0,T}\left[\Phi\left(\frac{-y}{\sqrt{w}} + \frac{\sqrt{w}}{2}\right) - \exp(y)\Phi\left(\frac{-y}{\sqrt{w}} - \frac{\sqrt{w}}{2}\right)\right]$$

We want to transform the Dupire equation to the new coordinates. We start by calculating the derivatives in the new variables:

$$\begin{aligned}
\frac{\partial C(y,T)}{\partial y} &= \frac{\partial C(y,w)}{\partial y} + \frac{\partial C(y,w)}{\partial w}\frac{\partial w}{\partial y} \\
\frac{\partial^2 C(y,T)}{\partial y^2} &= \frac{\partial}{\partial y}\left(\frac{\partial C(y,T)}{\partial y}\right) \\
&= \frac{\partial}{\partial y}\left(\frac{\partial C(y,w)}{\partial y} + \frac{\partial C(y,w)}{\partial w}\frac{\partial w}{\partial y}\right) \\
&= \frac{\partial^2 C(y,w)}{\partial y^2} + 2\frac{\partial C(y,w)}{\partial y \partial w}\frac{\partial w}{\partial y} + \frac{\partial^2 C(y,w)}{\partial w^2}\left(\frac{\partial w}{\partial y}\right)^2 + \frac{\partial C(y,w)}{\partial w}\frac{\partial^2 w}{\partial y^2} \\
\frac{\partial C(y,T)}{\partial T} &= \frac{\partial C(y,w)}{\partial T} + \frac{\partial C(y,w)}{\partial w}\frac{\partial w}{\partial T}
\end{aligned}$$

In this new set of variables, Dupire equation (6) as rearranged in (11) becomes:

$$\begin{aligned}
\frac{\partial C(y,w)}{\partial T} + \frac{\partial C(y,w)}{\partial w}\frac{\partial w}{\partial T} = \; & \frac{\sigma^2(K,T)}{2}\Big(\frac{\partial^2 C(y,w)}{\partial y^2} + 2\frac{\partial C(y,w)}{\partial y \partial w}\frac{\partial w}{\partial y} \\
& + \frac{\partial^2 C(y,w)}{\partial w^2}\left(\frac{\partial w}{\partial y}\right)^2 + \frac{\partial C(y,w)}{\partial w}\frac{\partial^2 w}{\partial y^2} - \frac{\partial C(y,w)}{\partial y} \\
& - \frac{\partial C(y,w)}{\partial w}\frac{w}{\partial y}\Big) + (r_T - d_T)C(y,w) \qquad (12)
\end{aligned}$$

By what we said about Forward Equation, we know that $C_{BS}(y,w)$ satisfies this equation as well. Recalling that we let $w = \sigma^2_{BS}(K,T)T$ ($\Rightarrow \sigma_{BS} = \sqrt{\frac{w}{T}}$) and $y = \ln(K/F_{0,T})$ ($\Rightarrow K = F_{0,T}\exp(y)$), we may now compute the partial derivatives to be inserted in (12):

$$\begin{aligned}
\frac{\partial C_{BS}(y,w)}{\partial w} &= \frac{\partial C_{BS}(y,w)}{\partial \sigma}\frac{\partial \sigma}{\partial w} = \frac{F_{0,T}\phi(d_+)}{2\sqrt{w}} \\
\frac{\partial^2 C_{BS}(y,w)}{\partial w^2} &= F_{0,T}\phi(d_+)\left[-\frac{1}{4\sqrt{w^3}} - \frac{1}{2\sqrt{w}}d_+\frac{\partial d_+}{\partial w}\right] \\
&= \frac{F_{0,T}\phi(d_+)}{2\sqrt{w}}\left[-\frac{1}{2w^2} + \left(\frac{y}{\sqrt{w}} - \frac{\sqrt{w}}{2}\right)\left(\frac{2}{\sqrt{w^3}} + \frac{1}{4\sqrt{w}}\right)\right] \\
&= \frac{\partial C_{BS}(y,w)}{\partial w}\left[-\frac{1}{8} - \frac{1}{2w} + \frac{y^2}{2w^2}\right]
\end{aligned}$$

$$\begin{aligned}
\frac{\partial C_{BS}(y,w)}{\partial y} &= \frac{\partial C_{BS}(y,w)}{\partial K}\frac{\partial K}{\partial y} = -F_{0,T}\exp(y)\Phi(d_-) \\
\frac{\partial^2 C_{BS}(y,w)}{\partial y^2} &= -F_{0,T}\exp(y)\Phi(d_-) - F_{0,T}\phi(d_-)\frac{\partial d_-}{\partial y} \\
&= -F_{0,T}\exp(y)\Phi(d_-) - F_{0,T}\phi(d_-)\left(-\frac{1}{\sigma\sqrt{T}}\right) \\
&= -F_{0,T}\exp(y)\Phi(d_-) - F_{0,T}\phi(d_-)\left(-\frac{1}{\sqrt{w}}\right) \\
&= -F_{0,T}\exp(y)\Phi(d_-) + \frac{F_{0,T}\phi(d_+)}{\sqrt{w}}
\end{aligned}$$

In the above we have used: $\phi(\cdot)$ for normal pdf and $\Phi(\cdot)$ for normal Cdf. We can rearrange the above and we can rewrite:

$$\begin{aligned}
\frac{\partial^2 C_{BS}(y,w)}{\partial y^2} - \frac{\partial C_{BS}(y,w)}{\partial y} &= \frac{F_{0,T}\phi(d_+)}{\sqrt{w}} = 2\frac{\partial C_{BS}(y,w)}{\partial w} \\
\frac{\partial^2 C_{BS}(y,w)}{\partial y \partial w} &= -F_{0,T}\exp(y)\phi(d_-)\frac{\partial d_-}{\partial w} \\
&= -F_{0,T}\phi(d_+)\left(\frac{y}{2\sqrt{w^3}} - \frac{1}{4\sqrt{w}}\right) \\
&= \frac{\partial C_{BS}(y,w)}{\partial w}\left(\frac{1}{2} - \frac{y}{w}\right) \\
\frac{\partial C_{BS}(y,w)}{\partial T} &= \frac{\partial F_{0,T}}{\partial T}\left[\Phi\left(-\frac{y}{\sqrt{w}} + \frac{\sqrt{w}}{2}\right) - \exp(y)\Phi\left(-\frac{y}{\sqrt{w}} - \frac{\sqrt{w}}{2}\right)\right] \\
&= (r_T - d_T)C_{BS}(y,w)
\end{aligned}$$

If we now insert the formulas above into (12), $(r_T - d_T)C_{BS}(y,w)$ terms cancel out and we obtain

$$\frac{\partial C(y,w)}{\partial w}\frac{\partial w}{\partial T} = \frac{\sigma^2(K,T)}{2}\frac{\partial C_{BS}(y,w)}{\partial w}\left[2 + 2\left(\frac{1}{2} - \frac{y}{w}\frac{\partial w}{\partial y}\right) + \left(-\frac{1}{8} - \frac{1}{2w} + \frac{y^2}{2w^2}\right)\left(\frac{\partial w}{\partial y}\right)^2 + \frac{\partial^2 w}{\partial y^2} - \frac{\partial w}{\partial y}\right]$$

Cancelling out the factor $\frac{\partial C_{BS}(y,w)}{\partial w}$ and simplifying yields:

$$\frac{\partial w}{\partial T} = \frac{\sigma^2(K,T)}{2}\left[1 - \frac{y}{w}\frac{\partial w}{\partial y} + \frac{1}{4}\left(-\frac{1}{4} - \frac{1}{w} + \frac{y^2}{w^2}\right)\left(\frac{\partial w}{\partial y}\right)^2 + \frac{1}{2}\frac{\partial^2 w}{\partial y^2}\right]$$

Remembering that $w = \sigma^2_{BS}(K,T)$, the Black-Scholes Implied Volatility, we can see that the formula above can easily be inverted to obtain an expression for the Local Variance $\sigma^2(K,T)$ in terms of the Implied Volatility. In particular, we can rewrite:

$$\sigma^2(K,T) = \frac{\frac{\partial w}{\partial T}}{\left[1 - \frac{y}{w}\frac{\partial w}{\partial y} + \frac{1}{4}\left(-\frac{1}{4} - \frac{1}{w} + \frac{y^2}{w^2}\right)\left(\frac{\partial w}{\partial y}\right)^2 + \frac{1}{2}\frac{\partial^2 w}{\partial y^2}\right]} \tag{13}$$

We can observe now that when the volatility smile is flat for each maturity, the partial derivatives with respect to log-moneyness y disappear and the local variance simplifies to

$$\sigma^2(T) = \frac{\partial w}{\partial T}$$

1.5 Concluding remarks

We would like to stress two facts in this final section.

The first is that Kolmogorov Forward Equation (3) underpins both the derivations given, the one coming from Option Prices and the one coming from Implied Volatility: in a sense, therefore, the resulting formulas should be similar in spirit. But since, as shown, the analytics of those derivations are different, quite a different shape of the two resulting local volatility functions may come out of the grids, in particular in boundary regions. Considering the issue from a different viewpoint, it is immediate to realize that while *Market Implied Volatility is the common ingredient of both* the latter uses it directly, whilst the former goes through the meanders of Black-Scholes non-linear formula, thus potentially introducing further instability[11].

We will explain in detail in [Chap. 2] how the numerical fit of Local Volatility is performed in a Lattice framework. At that point we will give figures about the shapes of the two resulting surfaces (see Figs. 3, 4,5,6).

The other point we would like to make is that, even if apparently weaker in terms of results, Dupire's formula in the limit (as discretization step of differential operator tends to zero) should recover the Implied Volatility formula, which is better behaved only because of the inevitable machine epsilon that surfaces because of the additional non-linearity in Dupire's case.

[11]That seems to be the reason why in the industry it is the *latter* that people use

2 First Setup: Finite Differences for One Factor PDE

2.1 Introduction and Scope

We will start this chapter by giving a few coordinates on Finite Difference Schemes to solve Initial and Boundary Value Problems (IBVP). The mechanics of various interpolation algorithms that drive our discretization of the differential operator will be quickly reviewed. Explicit, Implicit and Crank-Nicolson schemes will be mentioned since our implementation permits the runtime to parametrically switch among all three of those.

The second part of this chapter [Section 2.3] delves into the internals of the two local volatility implementations discussed in [Chap. 1].

Finally a list of strategies for finer tuning is given in [Section 2.4] -some of them deployed in code base and some only explored for future reference.

2.2 Brief excursus on finite difference schemes

In a number of finance modeling domains, where the dimensionality of the problem does not reach past three or possibly four[12], Finite Difference Schemes prove very robust and effective: their use is well established amongst the quantitative finance professionals and we have started our implementation of Local Volatility Function in a FDM setup.

A few points are worth mentioning right at the beginning. In real analysis, amongst the two limit processes -integration and differentiation, the former is the difficult one (to the point where some functions cannot be analytically integrated), whilst the second is trivially accomplished for each and every function. In numerical analysis, the pattern is the exact reciprocal: integration or quadrature is the simple procedure[13], while differentiation, as an approximation method, is considered unstable and best avoided when possible. In the process of solving partial differential equations, though, where very seldom an analytical solution is available, numerical differentiation becomes the leading ingredient and as such its quirks must be understood and somehow dominated.

As it is known, for a function $f\colon \mathbb{R} \to \mathbb{R}$, the analytical derivative at $x_0 \in \mathbb{R}$ is

$$f^{'}(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}$$

.

As in the case of numerical integration, the point here is deciding which interpolating engine to use in order to discretize the differential operator. We start interpolating $f \in \mathbb{C}^2[a, b]$ by a first order Lagrange polynomial, where setting $x_1 = x_0 + h$ we will have: [14]

$$\begin{aligned} f(x) &= P_{0,1} + \frac{(x - x_0)(x - x_1)}{2!} f^{''}(\xi(x)) \\ &= f(x_0)\frac{x - x_0 - h}{-h} + f(x_0 + h)\frac{x - x_0}{h} + \frac{(x - x_0)(x - x_0 - h)}{2!} f^{''}(\xi(x)) \end{aligned}$$

where Taylor Theorem gives $\xi(x) \in (a, b)$

By differentiating the above and fixing x to be equal to x_0 we have:

$$f^{'}(x_0) \approx \frac{f(x_0 + h) - f(x_0)}{h} - \frac{h}{2} f^{''}(\xi)$$

[12]We will discuss this issue in more detail in [Chap. 3]

[13]At least in one dimension. For higher dimensional problems, difficulties may arise in particular w.r.t. the region of integration

[14]Recall:

$$P_{0,1} = f(x_0)\frac{x - x_1}{x_0 - x_1} + f(x_1)\frac{x - x_0}{x_1 - x_0}$$

where, if $f^{''}(x)$ is uniformly bounded on the function domain - so that $\exists M \in \mathbb{N}$ s.t $|f^{''}(x)| \leq M \ \forall x \in [a,b]$, then the truncation error to be had by using the approximation

$$f^{'}(x_0) \approx \frac{f(x_0+h) - f(x_0)}{h} \tag{14}$$

is at most $\frac{M|h|}{2}$.

Standard terminology for (14) reads:

- *Forward Difference* if $h > 0$;
- *Backward Difference* if $h < 0$

The reason why the numerical approximation of derivatives is an unstable process is easily seen from equation (14): a better fit of the function is obtained by reducing the interval $\Delta x = [x_0, x_0 + h]$ (which means by reducing $|h|$) because the finite difference $\Delta f = |f(x_0 + h) - f(x_0)|$ is closer and closer in the limit to the differential operator and that will produce a better local approximation of the functional form f. At the same time, the denominator will also shrink and that will increase the numerical round-off error because we are effectively dividing by increasingly smaller numbers. So the trade-off amongst discretization error and round-off error is what makes numerical differentiation unstable. Another trade-off that has clearly played a role in this exercise is between the number of points used to compute the finite difference and the round-off error itself. For, consider having a $\mathbb{C}^{n+1}[a,b]$ function f, with $\{x_0, ..., x_n\}$ (n+1) points $\in [a,b]$. By differentiating the n-th order Lagrange polynomial that interpolates f at those points we will have the *(n+1)-order formula*:

$$f^{'}(x_j) = \sum_{k=0}^{n} f(x_k) L_k^{'}(x_j) + \frac{f^{(n+1)}(\xi(x_j))}{(n+1)!} \prod_{k=0, k\neq j}^{n} (x_j - x_k)$$

The issue here is the trade-off between the better fit coming from using more interpolation points and the increasing of round-off error when more functional evaluations are performed[15]. The formulas we have constantly used in our polynomial fit are the *three points* and *five points* formulas. In particular, for the three point case we have:

$$f^{'}(x_0) = \frac{1}{2h}[-3f(x_0) + 4f(x_0+h) - f(x_0+2h)] + \frac{h^2}{3} f^{(3)}(\xi_0), \quad \xi_0 \in (x_0, x_0+2h) \tag{15}$$

and

$$f^{'}(x_0) = \frac{1}{2h}[f(x_0+h) - f(x_0-h)] - \frac{h^2}{6} f^{(3)}(\xi_1), \quad \xi_1 \in (x_0 - h, x_0 + h) \tag{16}$$

The central difference scheme in (16) is the approximation we have used most frequently in our FDM computation of the Local Volatility Function. The fact that its error is $O(x^2)$ and that the functional evaluations happen only in two points (instead of three) make the scheme reliable and fast.

Another numerical scheme we have used to fit the target function is the five points formula:

$$f^{'}(x_0) = \frac{1}{12h}[f(x_0-2h) - 8f(x_0-h) + 8f(x_0+h) - f(x_0+2h)] \tag{17}$$

$$+\frac{h^4}{30} f^{(5)}(\xi_2), \quad \xi_2 \in (x_0 - 2h, x_0 + 2h) \tag{18}$$

where the error term is $O(x^4)$.

[15] We are considering here only the numerical point of view of the accuracy in the approximation: if we were to consider also performance, more evaluations would *ceteris paribus* slow down our computation. We will see that such considerations are not off-topic when building pricing algorithms

In our setup, we want to be able to test results for Explicit, Implicit and Mixed (i.e., Crank-Nicolson) schemes. The possibility of being able to switch among all three schemes (with a θ parameter taking values in the continuum of [0,1], with Explicit= 1, Implicit = 0, CN = 0.5) is not only of theoretical interest. Clearly, the base case for which we have provided results is Crank-Nicolson. It is quite standard in Numerical Analysis books (see [5]) to prove why, besides being *unconditionally Stable*[16], it also enjoys second order convergence no matter what is the volatility structure. But in the cases relevant for Financial Mathematics, i.e. when either the payoff is a discontinuous function or it has some non-differentiability, the scheme may be shown *not* to achieve its full potential (see in particular [10] and [11]). Therefore on the one hand we have coded remedies to smooth function payoffs in order to recover Crank-Nicolson's full potential. On the other hand, we have allowed the runtime to switch among schemes because, even if no time was left to pursue this route numerically, we have done some research on the viability of having Implicit Schemes at the beginning and Crank-Nicolson afterwards in order to smooth spurious oscillations (more on that in [Section 2.4]).

2.3 Various polynomial fits for Local Volatility Function

The generic IBVP (Initial and Boundary Value Problem) for a second order parabolic PDE can be illustrated, fox example in the case of the *heat equation*, as follows. Let u(x,t) a real valued function, $\alpha \in \mathcal{R}$

$$\frac{\partial u}{\partial t}(x,t) = \alpha^2 \frac{\partial^2 u}{\partial x^2}(x,t), \quad 0 < x < l, \quad t > 0 \tag{19}$$

subject to conditions

$$\begin{aligned} u(x,0) &= f(x), \quad 0 \leq x \leq l \\ u(0,t) &= g(t), \quad g(t)\ a\ given\ function \\ u(l,t) &= h(t), \quad h(t)\ a\ given\ function, \quad t > 0 \end{aligned}$$

We have given here an example to terminology. The equation is first order in time, so we need one Initial Condition only. Moreover, since it is second order in space, we need two Boundary Conditions: in our case we have given function value at end points, so we apply the Dirichlet boundary conditions[17]

To characterize the evolution of the PDE in the grid, fix a time-step size k and a space mesh-size $h = \frac{l}{m}$ (where $m \in \mathbb{N}, m > 0$). The resulting generic grid point will be: (x_i, t_j), where $x_i = ih$, for i $\in \{0, ..., m\}$, and $t_j = jk$, for j=0,... At each node, we will compute the local volatility using

- in time domain, point t_j and t_{j+1};
- in log spot domain, a subset[18] of the collection of mesh points with x_i as pivot;

Our discussion here will explore the two following cases which we have analytically derived in the previous chapter, i.e.

(i) computing the Local Volatility Function from Call Option Prices;

(ii) computing the Local Volatility Function from Market Implied Volatility.

For each one of them, we will give details of three different polynomial fits.

[16]We have called numerical differentiation an *unstable* algorithm, by which we mean that small changes in the initial input may result in big changes in the final result. If this obviously undesirable property holds only for some collection of input data, the algorithm is called *conditionally stable*. Otherwise it is *stable*.

[17]We could have given derivatives of function at end points, in the form of Newman Boundary Conditions. Combining both, we would have got Robin's Boundary Conditions

[18]We will see in the spline case, the subset coincides with the entire collection of discretization points

(a) a base case fit;

(b) a quadratic polynomial;

(c) a (natural/clamped) cubic spline fit.

2.3.1 Dupire case

While Implied Volatility ($\sigma_{impl}(K,T)$), as a function of Strike and Maturity, is a *Market Number*, Local Volatility ($\sigma_{loc}(S_t,t)$), as a function of Asset Level (S_t) and time t, is a *Model Number.* To compute Local Volatility, then, we have performed the following steps (details of linear, quadratic and spline fits are given):

(1) compute Forward Price at t_j and t_{j+1};

(2) use the log-spot mesh points (sorted in ascending order), with pivot x_i, to compute strike's vector;

(3) compute SABR Market Implied volatility (through look-up of term-structure point) as a function of (K,T,Fwd) for each of the strikes in previous step at respectively t_j and t_{j+1};

(4) from Black-Scholes Option Pricing Formula, derive a collection of Option Prices to be used in approximating $\frac{\partial C}{\partial T}$, $\frac{\partial C}{\partial K}$, $\frac{\partial^2 C}{\partial K^2}$: where each Forward Price is passed as the spot, each spot is passed as the strike, and the previously calculated implied volatilities are passed as Black-Scholes volatilities;

(5) Time Derivative is then the difference

$$\frac{C_{t_{j+1}} - C_{t_j}}{t_{j+1} - t_j}$$

(6) First and Second Order Space Derivatives are computed through the fit of

   (i) a linear polynomial

   (ii) a quadratic Lagrange polynomial

$$\begin{aligned}&f(x_0)\left(\frac{x-x_m}{x_0-x_m}\right)\left(\frac{x-x_p}{x_0-x_p}\right)\\&+f(x_m)\left(\frac{x-x_0}{x_m-x_0}\right)\left(\frac{x-x_p}{x_m-x_p}\right)\\&+f(x_p)\left(\frac{x-x_m}{x_p-x_m}\right)\left(\frac{x-x_0}{x_p-x_0}\right)\end{aligned}$$

   (iii) a cubic (natural/clamped) spline

(7) the final first or second space-derivative is computed by a weighted average of the corresponding polynomial fit at each of the two time steps t_j and t_{j+1}. The weight is set in order to preserve symmetry in time of space discretization.

Now, it may happen that the Local Volatility computation results in an extremely small positive number or even in a negative number. For a number of reasons that will explore later on[19] we decided to put a lower bound on σ_{loc} to be equal to 10E-4.

With respect to parameterization in Table 2.3.1, the Figures below show the resulting Local Volatility Surface derived from Option Prices:

[19]See [Section 2.4]

Parameter	Value
Expiry	3y
Spot	95
Strike	100
DriftRate	0.0
DiscountRate	0.02

Table 2: European Call Parameters

(i) when a Quadratic Fit (Fig. 3) is employed;

(ii) when a Natural Cubic Spline Fit (Fig. 4) is employed.

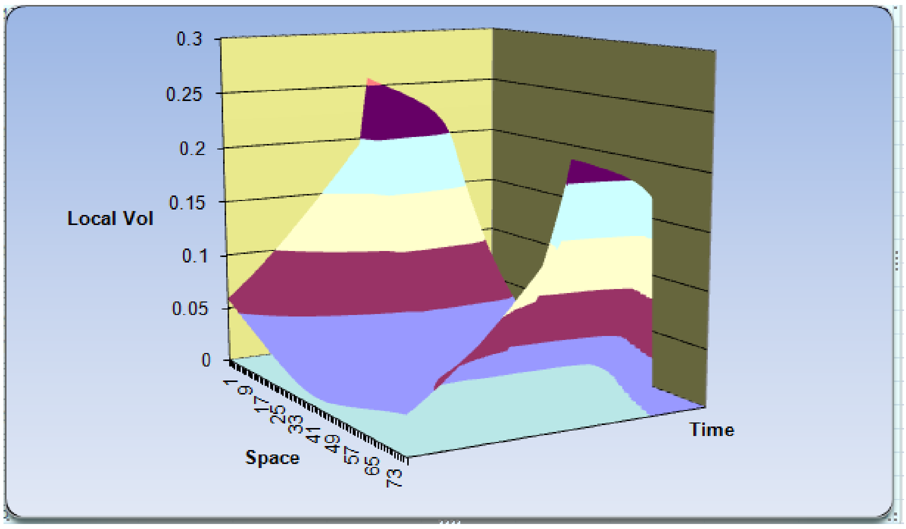

Figure 3: Local Vol Surface from Call Options Price: quadratic fit for Grid (X,T)=(80,64)

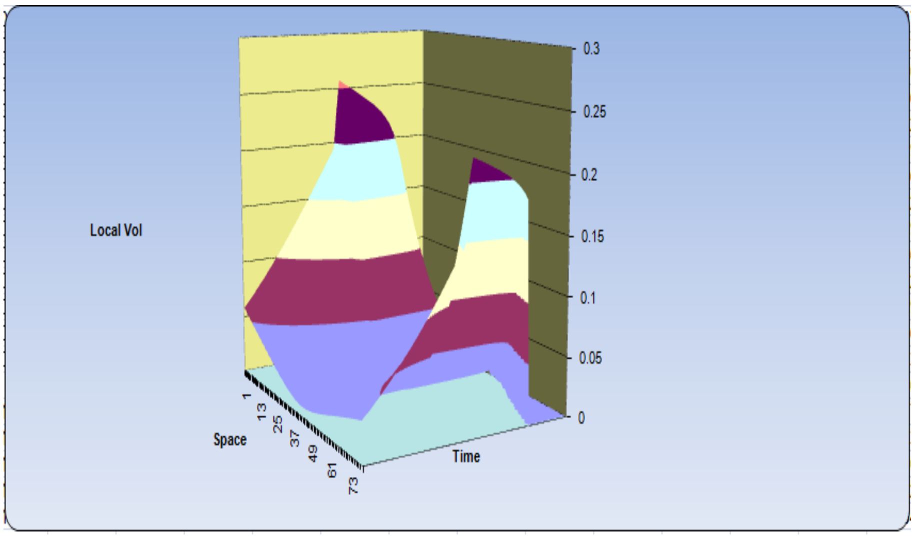

Figure 4: Local Vol Surface from Call Options Prices: spline fit for Grid (X,T)=(80,64)

2.3.2 Implied Volatility case

To compute Local Volatility as a Function of Implied Volatility, those are the relevant steps (details of linear, quadratic and spline fits are given):

(1) compute Forward Price at t_j and t_{j+1} and at $\frac{t_j+t_{j+1}}{2}$;

(2) use the log-spot mesh points (sorted in ascending order), with pivot x_i, to compute strike's vector;

(3) calculate SABR Market Implied volatility (through look-up of term-structure point) as a function of (K,T,Fwd) for each of the strikes in previous step at respectively t_j, t_{j+1} and $\frac{t_j+t_{j+1}}{2}$;

(4) from Implied Volatility Formula, derive a collection of Implied Volatilities to be used in approximating $\frac{\partial \sigma_{impl}}{\partial T}$, $\frac{\partial \sigma_{impl}}{\partial K}$, $\frac{\partial^2 \sigma_{impl}}{\partial K^2}$;

(5) Time Derivative is then the difference

$$\frac{\sigma_{impl}(t_{j+1}, x_i, fwd_{t_{j+1}}) - \sigma_{impl}(t_j, x_i, fwd_{t_j})}{t_{j+1} - t_j}$$

(6) First and Second Order Space Derivatives are computed through the fit of

   (i) a linear polynomial

   (ii) a quadratic Lagrange polynomial

$$\begin{aligned} & f(x_0)\left(\frac{x - x_m}{x_0 - x_m}\right)\left(\frac{x - x_p}{x_0 - x_p}\right) \\ & +f(x_m)\left(\frac{x - x_0}{x_m - x_0}\right)\left(\frac{x - x_p}{x_m - x_p}\right) \\ & +f(x_p)\left(\frac{x - x_m}{x_p - x_m}\right)\left(\frac{x - x_0}{x_p - x_0}\right) \end{aligned}$$

   (iii) a cubic (natural/clamped/tension) spline

As in previous case, we have set a lower bound on volatility ($\sigma_{impl} \geq 10E{-}4$) because our scheme would have probably suffered from low granularity in the evolution of the diffusion component.

Again as previously for Dupire, here we show the Local Volatility Surface coming from Grid in case of same parameterization as before (see 2.3.1) for:

(i) with Quadratic Fit (Fig. 5)

(ii) with Natural Cubic Spline Fit (Fig. 6)

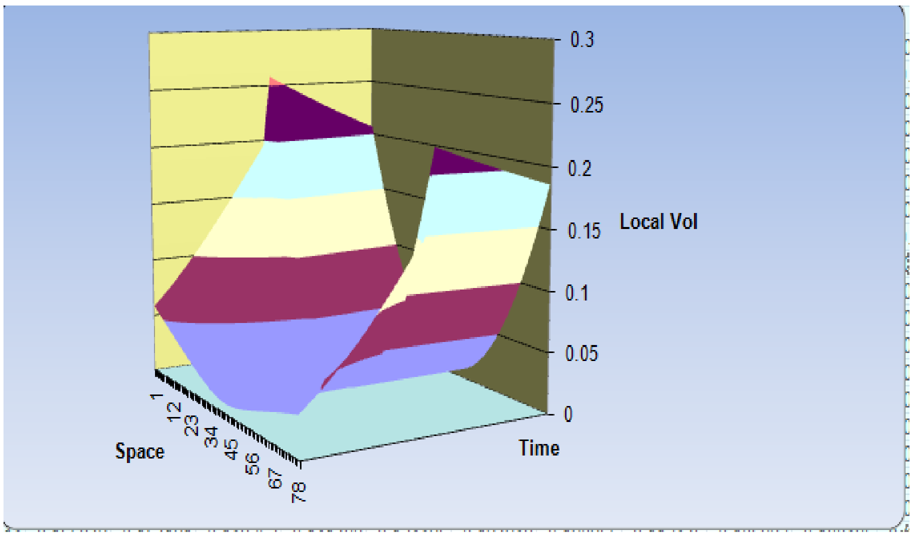

Figure 5: Local Vol Surface from Implied Vol: quadratic fit for Grid (X,T)=(80,64)

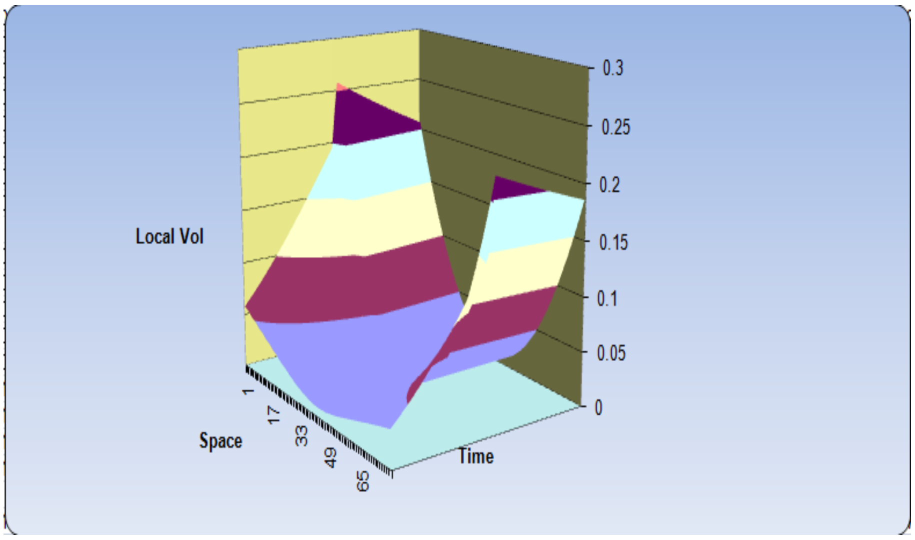

Figure 6: Local Vol Surface from Implied Vol: spline fit for Grid (X,T)=(80,64)

2.4 Implementation Issues and Tuning

Discretizing a differential operator by means of finite difference schemes is a well established practice in the finance community[20]. In any case, a vanilla framework may not work for more complicated setups, like ours, where a local volatility function exhibits numerical instability: that is why in literature there exist manifold implementation strategies and multiple fine-tuning recipes. The target of this section is to explain the reasons behind the choice among those and to document the benefits coming from each one of the adopted.

At various stages, the following strategies have found their way into our runtime:

(i) Log-Space Transform: removing state-space dependency

(ii) Boundary Conditions: how to recapture distribution

[20]For references on the history of FDM in finance, see [4]

(iii) Optimal place of Strike on grid:

(iv) Choice of discretization mesh: sampling same points

(v) Controlling PDE granularity

(vi) Smoothing the terminal payoff

(vii) "Black-Scholes Approximation"

2.4.1 Log-Space Transform

In general, the stability of a Finite Difference Scheme (of whichever type) for a parabolic equation of the Black-Scholes type is increased if a Log-Transform of the equation itself is performed. To see why, consider:

$$\frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} + \frac{\partial V}{\partial T} - rV = 0 \tag{20}$$

and perform the substitution $y = \ln(S)$. Writing $W(y,T) = V(S,T)$ and applying chain rule we get:

- $\frac{\partial V(S,T)}{\partial S} = \frac{\partial W(y,T)}{\partial y}\frac{\partial y}{\partial S} = \frac{1}{S}\frac{\partial W(y,T)}{\partial y}$
- $\frac{\partial^2 V(S,T)}{\partial S^2} = \frac{\partial}{\partial y}\left(\frac{\partial W(y,T)}{\partial y}\frac{\partial y}{\partial S}\right) = \frac{\partial}{\partial y}\left(\frac{1}{S}\frac{\partial W(y,T)}{\partial y}\right) = -\frac{1}{S^2}\frac{\partial W(y,T)}{\partial y} + \frac{1}{S}\frac{\partial^2 W(y,T)}{\partial y^2}\frac{\partial y}{\partial S}$
- $\frac{\partial V(S,T)}{\partial T} = \frac{\partial W(y,T)}{\partial T}$

Upon simplification, original equation (20) becomes:

$$\frac{1}{2}\sigma^2\frac{\partial^2 W}{\partial y^2} + \left(r - \frac{1}{2}\sigma^2\right)\frac{\partial W}{\partial y} + \frac{\partial W}{\partial T} - rW = 0 \tag{21}$$

Equation (21), unlike (20), is a constant coefficients PDE: that is exactly what makes the numerical analysis of it more stable [4].

2.4.2 Boundary Conditions

During the work that led to our implementation, we have dedicated quite a long time to fine tune the upper and lower boundary conditions for the Local Volatility Finite Differencing. Suppose we are simulating the evolution of a PDE and assume we know, as in the BS case, that the actual density of the diffusion is a Gaussian. In measure theoretic terms, let $(\Omega, \mathcal{F}, \mu)$ be a measure space and f a real valued function defined on Ω s.t. f $\in m\mathcal{F}$, f $\in L^2(\Omega)$. Then, $\forall t > 0$, by *Chebyschëv Theorem* we have:

$$\mu\left(\{x \in \Omega : |f(x)| \geq t\}\right) \leq \frac{1}{t^2}\int_\Omega |f^2| d\mu$$

By virtue of the above[21], and given the Gaussian density, we know that setting the boundaries in the PDE evolution to k*stdev we will be sampling $1 - \frac{1}{k^2}$ of the total probability support. That holds for a Gaussian: in case we do not have the distribution, we may be deluded by a crude ±k*stdev sample space limit (for upper and lower boundaries). In our case, we can still recover distribution. Indeed, by Breeden-Litzenberger (8) we know

$$\frac{\partial^2 C}{\partial K^2} = p$$

[21] In probabilistic terms that reads: let X be a random variable s.t. $\mathbb{E}[X] = \mu$ and $\mathbb{E}[(X-\mu)^2] < \infty$. Then $\forall k > 0$

$$Pr\left(|X - \mu| \geq k\sigma\right) \leq \frac{1}{k^2}$$

where C the call price, K the Strike and p the probability density function. Integrating the density over the whole relevant support Ω

$$\int_{\Omega} p = \int_{\Omega} \frac{\partial^2 C}{\partial K^2} = \frac{\partial C}{\partial K}$$

we easily recover the Cumulative Distribution Function.

2.4.3 Optimal place of Strike on grid

It is well known that discretization schemes may yield more or less accurate results depending on how the strike is positioned on the Grid. Take for instance a Barrier Option: positioning lattice's nodes on the barrier is quite a common practice to improve convergence and speed up runtime (see [18]). Tavella and Randall [35] recommend placing strike halfway between two neighboring node points and there exists a literature that explores the optimal configuration for different products to be priced. In our set up, because working in log-spot space instead of in standard coordinates, some more work was needed to adjust suggestions of Tavella and Randall. Analytically we tried to replicate their reasoning in log-space, but it was only heuristically that we were able to refine the factor by which to shift grid nodes. Here, the optimal factor seemed to change along with Strike.

2.4.4 Controlling PDE granularity

Contrary to what happens in Path Driven Sampling [Chap. 3], where the diffusion parameter may become zero without harming the results, in the case of lattices schemes, the disappearance of the diffusion coefficient may prevent the difference operator from achieving enough granularity to match the differential operator in the limit of mesh size going to zero. In other words, PDE may not be able to evolve. Besides, it is well known in the PDE literature that reduction of a second order parabolic to a first order hyperbolic (*convection equation*) results in some numerical issues. In fact, while in parabolic PDE discontinuities of initial conditions tend to decay as time goes on, in the hyperbolic case the smoothness of solution is determined by the continuity/discontinuity of the initial condition and this will be propagated indefinitely.

For all the reasons above, we have imposed a lower bound on σ_{loc} equal to 10E-4. True, for particular combinations of Strike/Time the local volatility fit may result in a negative value but while that may be recorded and used to explore convergence and stability issues, it is nonetheless evident that by construction volatility cannot be negative.

As a last remark, *drift dominated flow*, i.e. the setup in which the diffusion coefficient tends to disappear, needs to be handled in quite some more careful way: "upwinding" or "downwinding" techniques may be used [11].

2.4.5 Choice of discretization mesh

One of the marvels of mathematics is the continuous overlap of threads amongst various areas apparently disconnected, as for example, Combinatorics and Functional Analysis, Group Theory and Quantum Physics or Number Theory and Complex Analysis, to name just a few. The point we would like to stress here is the interplay between deterministic and random entities, or better between concepts that inhere those two areas. This is just another example of that underlying oneness of the substance we are contemplating and acting upon that we were stressing in the introduction w.r.t. Feynman-Kač.

When solving PDE through Finite Differences, the differential operator is discretized through a difference operator. In a sense, that is similar to the process of Riemann integration, where the continuous map is replaced by a step-wise continuous one. Now consider MonteCarlo integration: if function boundary is complicated and the integrand is not strongly peaked in very small regions then sampling points randomly distributed in a region that encloses the integration domain is a viable strategy. The way to do that can take any shape in between the two extremal configurations of:

- producing (pseudo-)random numbers that spread in the integration domain at large, with no pattern[22];
- producing a quadrature scheme on a predetermined Cartesian Grid.

In between sits the possibility of refining the mesh as we go along, i.e. producing points that have low-discrepancy on the domain we are sampling[23]. In our setup we are doing something similar in a way: having a sequence of meshes that do not overlap, i.e. using discretizations whose meshes are not refinements each of the next one, is intuitively similar to sampling points at random; on the other hand evolving the mesh such that we are effectively reusing the points we have already sampled within our previous time/space grid helps the result to converge faster. To be more explicit, the evolution of our meshes (in time and space) is something along these lines:

n $\to$ 2n $\to$ 2^2n $\to$ 2^3n $\to$ ... $\to$ 2^kn

If we were to do something different, for instance progressing like 10 to 15 and so on, we would be sampling different points and our convergence would be slower.

2.4.6 Smoothing the terminal payoff

Another improvement we have adopted in our various refinements of the Finite Difference Scheme is something that was originally advocated by Heston and Zhou in [16]. The idea is the following. When working in particular with Crank-Nicolson schemes, the payoff function is clearly sampled at discrete node points. In case of either non $\mathbb{C}^0$ payoff functions (think to binary case) or, like in our case, non $\mathbb{C}^1$ payoff functions, although the point of discontinuity/non differentiability has Lebesgue measure zero, *quantization error* may nonetheless arise because we are not working in a continuum of mesh points. To get away with it, instead of taking the terminal payoff function as it is, say $C_T(S)^j$ sampled at node point j, we may average the the payoff function over the adjacent node points, like this:

$$C_j = \frac{1}{\Delta S}\int_{S_j-\frac{\Delta S}{2}}^{S_j+\frac{\Delta S}{2}} C_T(S)dS$$

Moreover, the smoother convergence thus achieved makes room for the use of Richardson Extrapolation, which we did not have time to implement but that is of proven reliability in FDM schemes (see 2.4.8).

[22]Or with a pattern that is indistinguishable (modulo the statistical test of randomness we are employing) from no-pattern at all. See [A], where the talk is about "DieHard" and other tests of the good quality of the chosen Random Sequence Generator (in our case, primarily Mersenne Twister)

[23]On this respect, I cannot refrain here from quoting the full paragraph in Numerical Recipes where the point is made clear [36]: We have just seen that choosing N points uniformly randomly in an n-dimensional space leads to an error term in MonteCarlo integration that decreases as $\frac{1}{\sqrt{N}}$. In essence, each new point sampled adds linearly to an accumulated sum that will become the function average, and also linearly to an accumulated sum of squares that will become the variance. The estimated error comes from the square root of this variance, hence the power $N^{-\frac{1}{2}}$. Just because this square-root convergence is familiar does not, however, mean that it is inevitable. A simple counterexample is to choose sample points that lie on a Cartesian grid, and to sample each grid point exactly once (in whatever order). The MonteCarlo method thus becomes a deterministic quadrature scheme albeit a simple one whose fractional error decreases at least as fast as N^{-1} (even faster if the function goes to zero smoothly at the boundaries of the sampled region or is periodic in the region). The trouble with a grid is that one has to decide in advance how fine it should be. One is then committed to completing all of its sample points. With a grid, it is not convenient to "sample until" some convergence or termination criterion is met. One might ask if there is not some intermediate scheme, some way to pick sample points "at random", yet spread out in some self-avoiding way, avoiding the chance clustering that occurs with uniformly random points. A similar question arises for tasks other than MonteCarlo integration. We might want to search an n-dimensional space for a point where some (locally computable) condition holds. Of course, for the task to be computationally meaningful, there had better be continuity, so that the desired condition will hold in some finite ndimensional neighborhood. We may not know a priori how large that neighborhood is, however. We want to "sample until" the desired point is found, moving smoothly to finer scales with increasing samples. Is there any way to do this that is better than uncorrelated, random samples? The answer to the above question is "yes". Sequences of n-tuples that fill nspace more uniformly than uncorrelated random points are called quasi-random sequences. That term is somewhat of a misnomer, since there is nothing "random" about quasi-random sequences: they are cleverly crafted to be, in fact, subrandom. The sample points in a quasi-random sequence are, in a precise sense, "maximally avoiding" of each other.

2.4.7 "Black-Scholes Approximation"

This last improvement is quite well-understood and frequently found in the industry. It consists in using the analytical Black-Scholes values in the next-to-last sequence of node points and use the finite difference everywhere else.

At first sight we could argue that the above strategy, powerful as it is, is only viable when an analytical solution exists. On more careful consideration, it appears that even if we *do not know the shape of distribution a whole transition slice back from horizon*, we do actually have that *within a transition slice there are multiple time slices*, so for small time slice dt we can always recover our distribution via simple analytical considerations. With reference to the instability that plagues SABR model for some parameterizations, it is worth stressing that all the remarks above are applicable *modulo regions of negative density* (see [Section 1.3.1] and [28]).

2.4.8 Other Techniques: references

In literature, there exist various techniques that may either improve accuracy, accelerate convergence or both. We did not have time to embed them into a stable and accurate local volatility computation. More experiments and a better grasp of the underlying may be needed to transform the theoretical framework into a fully working C++ implementation. In particular, further investigation may prove fruitful along the following lines:

- consider *Richardson Extrapolation* in combination with Implicit Euler [11]: we could rearrange error term, if depending on step size, in a form that achieves high accuracy given the use of a low order (Taylor)-expansion. This technique needs a smooth convergence upstream: so we would apply it in conjunction with either the Heston-Zhou algorithm [2.4.6] or with the "Black-Scholes Approximation" [2.4.7];

- consider *Tikhonov Regularization Techniques* [7]: we could rewrite Fokker-Planck as a linear system of equations and use regularization with discrete smoothing to solve for the unknown volatilities;

- consider *Non-Uniform Meshes* [20]: if we consider in particular time discretization, from a business perspective a one week step (say) today is not equivalent to a one week step a year down the line. That may suggest to have a non-uniform mesh for the time discretization, with progressively larger steps.

  More generally, we may rephrase this issue as follows: in regions where strike lies, it is meaningful to make the discretization more dense, while everywhere else to space the points more. It may be possible to have an automatically adjusting scheme that rearranges the mesh depending on how "important" is the region we are sampling. We did not have time to spend on that but this way of proceeding may provide links with the concept of "importance sampling" in MonteCarlo methods.

- consider *Mixed Schemes* ([10] and [11]): while Explicit methods are notoriously only conditionally stable, Implicit methods are unconditionally stable and can prove useful in preventing spurious oscillations in Crank-Nicolson at the beginning. So a combination of fully Implicit Euler for first few time steps and Crank-Nicolson afterwards may end up improving our convergence. Many authors have shown that Crank-Nicolson is only asymptotically second order in space and in particular for discontinuous or non-differentiable payoffs the second-order convergence cannot be achieved;

- Investigate/analytically circumscribe rôle of *Péclet/Reinolds Numbers* in affecting results of arbitrary second-order parabolic PDEs

2.5 Concluding remarks

Both the above formulas, although their derivation is analytically sound, are quite intricate to use in practice, as our setup has shown and as many practitioners recognize [27]. Perhaps surprisingly, both the resulting Finite Difference Schemes converge quite fast to the true analytical price, with minor regions of slightly more pronounced error (see [Section 4.2.3]. All the enhancements we have spent time deploying have a beneficial impact on the convergence of schemes: results will show a sub-basis point price match also in case of quite crude grids. The relevance of *crude grids* is paramount for us since we need accuracy in particular for few -i.e. large- steps in time domain.

The local volatility surfaces displayed in Figs.(3),(4),(5),(6) above, for grids (X,T)=(80,64), contain some points of interest. They show surfaces from Option Prices and from Implied Volatilities in case of both Quadratic and natural Cubic Spline fits: results correspond to mildly OTM options. In *that* setup, i.e. with *that much* granularity, there are no apparent differences between Quadratic and Spline fits in both derivations: that in turn tends to confirm our intuition that, since Fokker-Planck has been derived by discarding higher order terms, Spline fit does not really improve on Quadratic fit. On the contrary: in region of high skew, the fit of a cubic polynomial may indeed miss the structure of curvature and perform worst than a quadratic fit.

The case of crude meshes is quite different: spline has not enough points to produce an accurate result and very often overshoots.

An additional point is even more interesting: surfaces derived from Implied Volatilities tend to decay more smoothly at the boundaries (in particular time boundaries) whilst surfaces coming from Dupire's setup tend to show pronounced discontinuities there. Our hypothesis is that machine epsilon may play a rôle, where Dupire's formula may indeed smear out its abrupt behavior if finer meshes (i.e less discretization error) were to be deployed. As we were suggesting in [Chapter 1], the similarity of formulas hides the fact that Dupire's derivation goes through an additional non-linear function (the Black-Scholes Option Pricing Formula), thus potentially magnifying instabilities.

Our reconstruction would not be complete if we would not mention that, in order to have a better grasp of how accurate our local volatility computation is, we have generated *through the same FDM engine* a PDE evolution with constant (i.e. flat) volatility as per Black-Scholes case[24]. Then we will use its results to partition the total error[25] of the Local Volatility schemes into the inaccuracy feeding through the grid *because of the skew* and the inaccuracy coming from *replacing the differential operator with a difference operator*[26]. We will comment on all of this in [Chap. 4] where figures will be given for all the different polynomial fits across Dupire, Implied Vol and flat BS frameworks.

[24]A Flat Volatility Black-Scholes Finite Difference scheme consists simply, as regards the *diffusion* coefficient, in multiplying the second order difference operator times a constant volatility, a parameter that comes as input from a Market Volatility Engine - in our case, the SABR. For details, see [Chap. 1]

[25]As we will see later, for a target (80x80) grid, less than a basis point

[26]Notice the similarity of this setup *mutatis mutandis* with the Control Variate idea of making calculations only of what is truly unknown.

3 Second Setup: Process Driven Sampling and the non-linear solver

3.1 Introduction and Scope

This chapter is best divided into three parts.

The first one, [Section 3.2], is a reconstruction of the analytics and numerics of Path-driven Sampling as employed in our exercise. Thus, after a brief account of the domain terminology we will explore the discretization schemes actually used, with particular emphasis on log-space one-step marching Euler-Maruyama and multi-steps Predictor Corrector. To accelerate runtime of notoriously slowly-converging MonteCarlo a few variance reduction schemes that have been implemented in code base will also be discussed.

The second part [Section 3.3] is where the specific nature of our work starts to surface: in pricing long-dated FX or some other exotic where the Smile/Skew plays a crucial rôle, MonteCarlo evolves with large steps in time domain. We will see how, *if no information on local volatility is available*, a non-linear solver can be deployed at each time step in order to reconstruct the missing local volatility information. Therefore we will delve into the mechanics of the Levenberg-Marquardt minimizer and we will see how at each time step our the objective function should consist in matching the analytical with the MonteCarlo price of a vanilla call across all sample trajectories. The issues of such a constrained[27] minimization of local volatilities will be explored. In particular we will comment on the fact that, even if solver recovers analytical price of vanilla, that does not as such imply the resulting local volatilities to be appropriate when used in pricing an exotic. Indeed, the problem structure is such that we will show how different solver reconstructed σ_{loc} vectors correspond to different initial guesses.

In the third part [Section 3.4], the work done in [Chap. 2] about reconstructing through lattices a local volatility function will be brought to fruition. In particular, we will focus on

- using (pseudo-)analytic local volatility as initial guess in solver

A more advanced formulation of the problem would involve

- constraining MonteCarlo σ_{loc} in order to adhere to a functional form centered on lattice's σ_{loc}

Finally a few more analytical improvements to the problem formulated in [section 3.3] will be discussed, in particular

- to weight the importance of strikes in solver to profile for Vega
- to automatically adjust the weighting function to fit tails incrementally

As regards the use of Local Volatility Function into Solver, a metric on how accurate is the pseudo analytic reconstruction may be offered by the computational time employed in the heavily-CPU intense minimization. Moreover, a possible test on how noisy are the resulting volatilities coming from the solver could be the pricing of a strongly path-dependent exotic, say a multidigital.

3.2 Schemes for SDE Discretization

Our implementation of the Large Time Step MonteCarlo algorithm had to choose among various possible discretization schemes. Some general terminology is here appropriate.

A Stochastic Initial Value Problem (SIVP) for scalar autonomous[28] Itô SDE can be expressed as:

$$\begin{cases} dX(t) = a(X(t))dt + b(X(t))dW(t), & t \in [0,T] \\ & X(0) = x \end{cases}$$

[27] Variance cannot be negative

[28] With no dependency of either drift or diffusion coefficients on time variable

where W(t) represent a standard Wiener process and the initial value x is fixed.

We define as usual:

(i) Trajectory Stability

(ii) Moment Stability

Letting X(T) be exact solution, $Y_\delta(T)$ the time-discretized approximation, δ the supremum of step refinements, c, β and γ scalar constants, reference is (in the first case above) to each and every sample path and in this respect we are intuitively contiguous to what happens in ODE world. Indeed, in order for Trajectory Stability to hold we require that for any time horizon T:

$$\mathbb{E}\left[|X(T) - Y_\delta(T)|\right] \leq c\delta^\gamma$$

In the case of Moment Stability, we require instead that for any time horizon T

$$|\mathbb{E}\left[g(X(T))\right] - \mathbb{E}\left[g(Y_\delta(T))\right]| \leq c\delta^\beta$$

where in particular $g(\cdot)$ is a $C^{2(\beta+1)}$ function (see [21] and [33]).

3.2.1 One-Step marching: Euler-Maruyama and Milstein

We have adopted the standard Euler-Maruyama discretization for our exercise. In particular, we have followed well-understood practice of transforming standard coordinates into log-space coordinates. To perform sampling, from initial GBM

$$\frac{dS}{S} = (r - d)dt + \sigma dW \tag{23}$$

by routine application of Itô's lemma, we get to

$$d\ln S = (r - d - \frac{1}{2}\sigma^2)dt + \sigma dW \tag{24}$$

Since the Stochastic term is no longer dependent on State Variable, the numerics of it are again more stable. Notice the similarity here with what Brennan and Schwartz [4] advocated for discretizing PDE. That is another instance of the underlying sameness of apparently different numerical procedures, as we were saying in many points of this work.

Defining: $X \equiv \ln(S)$, and having, for N$\equiv$ number of time steps, $\Delta t \equiv \frac{T}{N}$, in (24), $\forall n \in \{0, N-1\}$ Euler-Maruyama scheme consists in:

$$X(t_{n+1}) = X(t_n) + (r - d - \frac{1}{2}\sigma^2)\Delta t + \sigma\sqrt{\Delta t}\mathcal{N}(0,1) \tag{25}$$

Notice that we are working here

- with *Uniform mesh*
- and *One Step Marching*

Another point is worth mentioning. It is quite easy to show that transforming the original Geometric Brownian Motion SDE in (23) into log-space coordinates makes the scheme exact up to order $\mathcal{O}(\Delta t)$. That is the reason why we have decided to use log-space Euler for one-step marching and we did not try any higher order Taylor Schemes, like e.g. Milstein (see [12], [26])

3.2.2 Predictor-Corrector Scheme

In the case of Multiple-Step Marching, the idea is quite similar to what we did in Crank-Nicolson as opposed to pure Explicit Scheme for PDE discretization. We are essentially taking an Explicit Euler Step as *predictor* and subsequently *correcting* it with an Implicit Step. That is based on the consideration that both drift and diffusion coefficients in (24) evolve during step Δt and performing an average of the above two values results in more accuracy than a single Euler Explicit Step as in (25).
Indeed, having two weigth coefficients α, $\eta \in [0,1]$, Predictor-Corrector scheme used is:

$$\begin{aligned}\tilde{X}(t_{n+1}) &= X(t_n) + (r - d - \frac{1}{2}\hat{\sigma}^2)\delta t + \hat{\sigma}\sqrt{\delta t}\mathcal{N}(0,1) && (26)\\ X(t_{n+1}) &= X(t_n) + (r - d - \frac{1}{2}\sigma(\tilde{X_{t_n}})^2)\delta t + \sigma(\tilde{X_{t_n}})\sqrt{\delta t}\mathcal{N}(0,1) && (27)\end{aligned}$$

where the first equation is the Euler Explicit Step (i.e. the Predictor), while the second is the Corrector Step.

3.2.3 Variance Reduction Schemes

Standard MonteCarlo convergence is quite slow. Even if industry standard Mersenne 19937 Twister algorithm is used to generate random sequences[29], convergence is still $\mathcal{O}\left(\frac{1}{\sqrt{N}}\right)$ where N is the number of trajectories.

Here we will give some details on the adopted Variance Reduction ideas employed. In particular we will briefly discuss:

(i) Antithetic sampling;

(ii) Sobol fed Brownian Bridge;

Antithetic Sampling is quite simple to implement and results *ceteris paribus* in a net win in terms of computational time. It consists in reusing a vector of draws by playing with the random variables involved. In particular, for Gaussian Variates, if $\mathcal{N}(0,1)$ is normal, also $-\mathcal{N}(0,1)$ is and it is possible to show that the standard error behaves predictably provided some additional restrictions are imposed [21]. In our setup it may be recalled that we are working *not along trajectories* but *along time steps.* The standard strategy of using rearranged sequence of Gaussian of previous path every other one cannot be employed here lest we want to introduce some spurious correlations. What we do is then the following: at each time step, we generate only half the cardinality of required random numbers and we use them in the antithetic way to produce noise for the other half of the trajectories for that same time step.

Another improvement to the raw speed we have accomplished is using a *Brownian Bridge* algorithm to back out Wiener increments from the paths (where of course ΔW_i is obtained by difference of two adjacent entries in the sample path that the bridge constructs). The interesting bit here is that the input noise for the Bridge algorithm in our implementation comes from a (Sobol) low-discrepancy sequence generator[30]. It may seem quite counter-intuitive to start with a quasi-random sequence and end up with a pseudo-random one, after a laborious transformation. But apart from the fact that Sobol construction is fast and hence the whole algorithm is still winning against a purely pseudo-random one, the structure inherent in Brownian Bridge may help convergence[31].

[29]We have employed Boost library implementation of it, which is solid and reliable, see also A.1
[30]For our implementation of Sobol, we relied on the code provided by [12]
[31]For further reference, see [21]

3.3 Performing Calibration during Simulation: the non-linear solver

The discretization schemes explored in the previous section will be used now in a setup that to the best of our knowledge is new in literature.

We may be interested in pricing path-dependent options, where an accurate, i.e. not noisy, local volatility function is paramount. Our tool of choice is process driven sampling[32].

After creating a time-mesh, we evolve *all the trajectories one step forward in time*, say from T_{i-1} to T_i where $i \in \{1, ..., n\}$. With no Local Volatility information, what we are effectively doing is performing calibration during simulation. All we have is Market Implied Volatility which gives us a terminal distribution at each T_i point. We have no information regarding the transition density from T_{i-1} to T_i. So we create an objective function that *at each time-step* should collect the T_i value of each trajectory and match, by traversing the parameter space in an intelligent way, vanilla prices and MonteCarlo prices. The parameter space is a variable size collection of node points whose cardinality we have arbitrarily chosen[33]. The non linear solver should work its way in the parameter space by returning exactly those diffusion coefficients that minimize the squared error between analytical and MonteCarlo prices.

Convergence, as a function of number of draws, should always happen: but again by very construction, the analytical price should be recovered from the MonteCarlo embedded in solver even for a *small* number of trajectories.

A more subtle point is worth mentioning before exploring the mechanics of our objective function: the solver converges to analytical price *no matter what is the starting guess.* In order to match underlying process, we are effectively numerically recovering *some* diffusion coefficients .

It is debatable whether that is a powerful feature of the minimizer or else it is quite an explicit admission of the fact that we are blindly going along, reconstructing through clever numerics the distribution at each T_i *with no real intelligence of what the diffusion looks like at that point across the strike space.*

3.3.1 Internal mechanics of the solver

The standard minimizer in the industry to tackle non-linear least squares problems is the Levenberg-Marquardt algorithm [25] and that is what we have used in our exercise. It is an iterative technique that locates minimum of a multivariate function expressed as sum of squares of real valued functions. It works as a combination of an always converging but slow *Steepest Descent* and a faster but not always converging *Gauss-Newton.* When the current solution is far from the correct one, steepest descent ensures the convergence will be eventually reached at some point. When the current solution is close to correct one, the Gauss Newton component takes in. The algorithm is adaptive because it controls its own damping factor between the two strategies. Before exploring the details of our objective function, a few technical points are appropriate. Letting:

- $\underline{p}$:= parameter vector $\equiv \{\sigma_i\}$
- $\underline{x}$:= measurements vector $\equiv \underline{0}$

Levenberg-Marquardt algorithm works its way in the solution space like this. Choosing

$$f : \mathbb{R}^m \to \mathbb{R}^n, \ n \geq m$$

[32]The reason why that is so are manifold: in particular, in a *Hybrids* space, stochastic interest rates play a big rôle in increasing the problem dimensionality. Schemes for two dimensional PDEs (like ADI/ADE) are available in literature and quite easily implemented. Growing past that threshold, the advantages of Lattices may quickly disappear. As Jäckel [21], puts it

> The concept of MonteCarlo integration becomes ever more appealing as the dimensionality of the domain increases.[...] the MonteCarlo method breaks even with lattice techniques as of *d=4* and outperforms regular grid methods for *d >4*.

[33]To be 19

we compute

$$\underline{p} \; s.t. \; f(\underline{p}) \approx \underline{x}$$

We are dealing with an L^2 minimization, i.e. we are computing

$$\underset{\underline{p}}{\text{argmin}} \; ||\underline{e}||_2 = \underset{\underline{p}}{\text{argmin}} \; ||\underline{x} - f(\underline{p})||_2 \tag{28}$$

Now a few points regarding our implementation:

(i) we do not furnish the minimizer with an analytic Jacobian but we use central differences to approximate it

(ii) we perform a Constrained Minimization where

- lower bound: σ^i_{loc} = 10E-4
- *no* upper bound [34]

(iii) termination condition is that $||e||_2$ becomes smaller than our tolerance, currently fixed at well under a thousandth of a basis point

The following is the pseudocode for our objective function:

```
∀ TStep ∈ 1,···,TSteps:
    1. produce Gaussian noise → ($\#\underline{noise} \equiv \#\underline{draws}$)
    2. have an initial guess for local volatility parameter vector $\underline{p}=\{\sigma_i\}$ → ($\#\{\sigma_i\} \equiv 19$)
    3. perform one step in MonteCarlo evolution across all draws with diffusion coefficient coming from local vol guess
    4. choose asset levels based on paths
    5. $\forall i \in \{1, ..., 19\}$, compute i-th Black-Scholes price, $C^i_{BS}$, where Local Vol is passed values specified at asset levels equidistant in an implied probability space
    6. $\forall i \in \{1, ..., 19\}$, compute i-th MonteCarlo price, $C^i_{MC}$, where Local Vol is passed same values as in previous step
    7. minimize function: $f(C^i_{BS}, C^i_{MC})$, consisting for us in:
        • $\forall i \in \{1, ..., 19\}$, create polynomial in $\underline{x}$ where each $x_i = (C^i_{BS} - C^i_{MC}) * W_i$
```

W_i in the above is the weight coefficients vector. The weights are determined in an attempt to perform some sensible profiling for Vega. We will explain that in [Section 3.4].

3.3.2 Convergence of vanilla solver: facts and figures

Let us explore convergence of our solver in the case of uninformed initial guess with the help of some figures.

With respect to parameterization in Table 3,

[34] It is debatable whether a possible upper bound could be a scalar factor times the Implied Market Volatility at the point. No underlying business intuition could support that, so we decided not to have any upper bound

Parameter	Value
Expiry	1y
Spot	95
Strike	100
DriftRate	0.01
DiscountRate	0.02
Time Steps	4
Initial Guess	0.1

Table 3: European Call Parameters

we have now a quarterly time-step, i.e. by our standards, a *large* one. That is consistent with what happens in long dated FX space. The following table shows in first column the number of iterations and in second the percentage difference, as a ratio of Spot, of Analytical Price(2.9632) and MonteCarlo Price:

Trajectories	% Difference
16,000	-0.0002%
32,000	-0.0001%
64,000	0.0002%
128,000	0.0001%
256,000	-0.0001%

Table 4: Converge of vanilla non-linear solver: facts

The two figures below show the evolution of local volatilities as reconstructed by vanilla solver across the relevant time steps of our exercise for 16,000 and 256,000 trajectories:

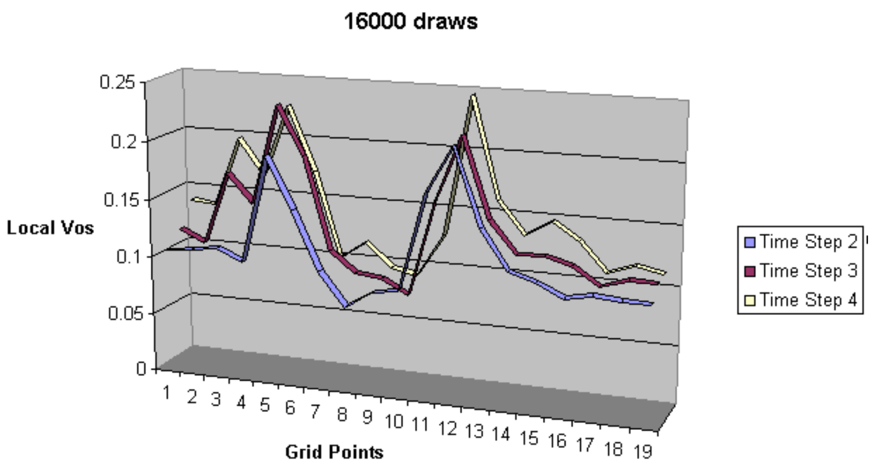

Figure 7: 16,000 draws: solver reconstructed local vols

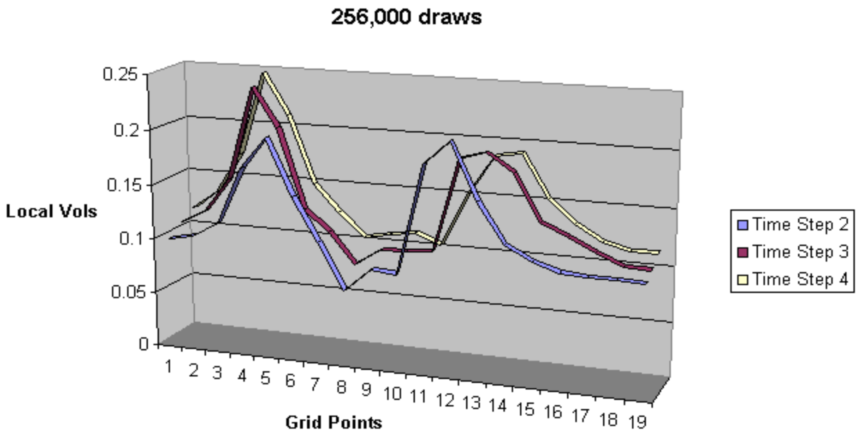

Figure 8: 256,000 draws: solver reconstructed local vols

We said previously that by construction solver *does recover analytical price in any case* so the figures in Table (4) are no big surprise. What may not necessarily have been expected is that a similar shape of the local volatility vector is apparent across time steps in each figure. That may be explained by observing that the solver has a standard marching strategy in fitting the points (currently from left to right). Moreover here we equipped the solver with a weight function that implies *fitting less aggressively the tails and giving full weight to the center of distribution.* In order to have a better grasp of how we implemented internals of object function, it may also be argued that the functional form of local volatility at time step T_i is not going to be completely different, in most cases, than the form of it at *previous* step T_{i-1}: that is why here we are giving figures for a continuously fed non-linear solver, i.e. a minimizer that at each step *takes as initial guess the previous step local volatilities vector*[35].

3.3.3 Issues associated with Solver

As Dupire taught us, European option prices are equivalent to densities whilst path-dependent option-prices correspond to the full diffusion process. As we mentioned previously, when we feed a *some particular* initial guess into the solver, because of the non-uniqueness of solution, we may bias the solver towards traversing solution space in one particular direction.

Consider for example the following set of tables, which show calibration of Local Volatilities in solver for two initial guesses (10% and 20%) with increasing number of iterations. Except for initial guess, parameters are the same as in previous section, Table (3).

8,000	**Initial Guess**		*16,000*	**Initial Guess**	
$K\backslash\sigma$	10%	20%	$K\backslash\sigma$	10%	20%
96	14.53%	17.28%	96	13.27%	16.27%
98	6.27%	5.77%	98	6.34%	5.61%
100	7.25%	5.58%	100	7.63%	7.02%
102	10.02%	13.20%	102	10.65%	11.55%
104	16.24%	15.88%	104	15.18%	15.74%

Table 5: σ_{loc} for different initial guesses

[35]Except for the first step where initial guess is given

32,000	**Initial Guess**		*64,000*	**Initial Guess**	
$K\backslash\sigma$	10%	20%	$K\backslash\sigma$	10%	20%
96	13.64%	16.37%	96	13.77%	16.58%
98	5.58%	5.32%	98	6.21%	5.53%
100	7.76%	6.51%	100	7.62%	6.72%
102	10.15%	12.40%	102	10.24%	13.49%
104	14.72%	14.75%	104	15.80%	15.37%

Table 6: σ_{loc} for different initial guesses: continued

Time steps here are quite large: still they are not larger than what may happen at calibration of long dated FX products in a production scenario[36]. In that case, simply having a path-driven sampling algorithm to perform small steps will make the computation infeasible. What Table (5) and (5) show is that with steps thus large, the objective function of matching a discrete set of option prices seems insufficient to provide a unique large step local volatility function. The terminal distributions will match by construction but forward transition densities will remain *different* for *different* initial conditions. The full diffusion contains much more information than the conditional laws where distinct diffusions may generate identical conditional laws and uniqueness of solution seems to be forever lost.

What we face here is a subtle epistemological issue: whether we should consider sensible or not that the solver recovers *in any case* some diffusion coefficients whilst it goes forward, time step after time step, sampling probability space: in a sense, our issue here is similar to what happens in local stochastic volatility modeling, where the local volatility function is chosen to match the input vanilla option surface *irrespective of the underlying stochastic volatility model.*

We may now point out that as a large time step Δt tends to a small time step δt the whole system will collapse onto the unique volatility function of Dupire: but working with a crude time mesh, we may need to constrain the problem further in order to find a "good" solution. That is what we will try to explore in the next section.

[36]With horizon of 10 to 20 years

3.4 Wiring Lattices and Path-Driven Sampling: Tuning

We have reconstructed a viable -i.e. numerically stable, accurate and fast- Local Volatility Function in various incarnations. We have shown that a path driven sampling algorithm to price long-dated FX products, if no Local Volatility function is provided, is able via a solver to compute the diffusion coefficients that, for a given reconstruction of terminal distribution around spot at each time slice, minimizes away L^2 error between analytical and MonteCarlo price.

What we want now, first of all, is to be able to use (pseudo)-analytic local vol as initial guess in solver. Successively we could even feed local vol at each iteration.

In order to accomplish that, we clearly need a concept of space discretization in a MonteCarlo world, where there is none. A possible way to achieve this arbitrary discretization in sample space, where no such concept as grid exists for that dimension[37], is to specify a collection of points in (log-)spot space bracketing the target (log-)spot itself.

We can play with distributions at each time step: theoretically, we could even use our arbitrary grid point cardinality *to sample more densely* regions closer to strike than regions away from it. In our setup, we have decided to work with an equally spaced (log-)spot grid centered on Strike.

All that said, we are in principle ready to perform our experiment: to see whether, notwithstanding the uncomfortable results of [Section 3.3.3], by feeding the solver with an initial guess coming from lattice's σ_{loc} we could improve on the runtime of the algorithm, reducing the amount of time the solver takes to find a viable local volatility parameter vector.

As an additional experiment, we may feed the solver with our previously created local volatility function *at each time step*: that will hopefully reduce the time it takes to reconstruct the surface while going along.

The last point we would like to explore is the following: within the objective function, we weight the space-points differently according to the closeness of the sampled point to the target strike itself. The underlying rationale is that deep OTM options have low Vega sensitivity so, if equal weight is given, solver expends too much effort on improving them.

We deployed various polynomial functions to fit strikes more or less aggressively depending on Vega.

We will investigate the impact of one such weight function on solver runtime: a Gaussian shaped function, with weights of value ≈ 1.0 in target Strike region and decaying symmetrically like $\exp\left(-\frac{x^2}{2}\right)$ along a standard deviation path away from strike

3.4.1 Large Δt: quarterly step

Consider the parameterization in Table (7):

Parameter	Value
Expiry	1y
Spot	95
Strike	100
DriftRate	0.02
DiscountRate	0.03
Time Steps	4
Initial Guess	0.1

Table 7: European Call Parameters

We employ here the symmetrical exponential smoothing of Strike space points in objective function as discussed above.

[37]It does of course exists for time instead

8,000 draws	T2	T3	T4
#	255	164	126
1	0.100001	0.107153	0.112596
2	0.106047	0.128893	0.138029
3	0.132099	0.13218	0.161968
4	0.177644	0.190069	0.213228
5	0.193166	0.23614	0.240238
6	0.186408	0.191059	0.171044
7	0.0798008	0.113082	0.118469
8	0.0636907	0.0938862	0.118215
9	0.0977353	0.0784082	0.101539
10	0.061404	0.0842616	0.08626
11	0.214728	0.181769	0.110198
12	0.132268	0.163534	0.176231
13	0.132609	0.120854	0.139734
14	0.0974354	0.140163	0.183109
15	0.121977	0.145724	0.140804
16	0.0844916	0.102099	0.115172
17	0.100202	0.12508	0.129367
18	0.0988453	0.106471	0.124262
19	0.1	0.0957336	0.112951

Table 8: LevMar reconstructed σ_{loc}: vanilla case (initial guess 0.1)

8,000 draws	T2	T3	T4
#	224	90	55
1	0.21678	0.222847	0.20604
2	0.172493	0.179171	0.178096
3	0.217127	0.22048	0.226921
4	0.212317	0.216512	0.215838
5	0.192549	0.197305	0.197037
6	0.167875	0.180863	0.16881
7	0.077306	0.0928133	0.0905416
8	0.0721695	0.089327	0.116096
9	0.0913672	0.105097	0.119218
10	0.0708143	0.0874441	0.117365
11	0.206468	0.16668	0.151365
12	0.139804	0.119165	0.111558
13	0.144439	0.124302	0.111868
14	0.169506	0.153596	0.129609
15	0.184673	0.17655	0.170478
16	0.18722	0.186879	0.172527
17	0.199638	0.194202	0.191251
18	0.215473	0.213085	0.204821
19	0.233649	0.22544	0.218693

Table 9: LevMar reconstructed σ_{loc}: feeding pseudo analytical Local Volatility

Tables (8),(9) above display figures for two runs, the first with the solver traversing parameter space with random initial guess (=0.1), the second with local volatility function furnishing solver with an informed guess. Second row in both tables display total iterations performed by Levenberg Marquardt at each time step. A table of summary figures follows:

Initial Guess	Total Time	%Error
Flat 0.1	15438 millis	0.000126%
σ_{loc}	9907 millis	-0.000168%

Table 10: Summary for runs in Tables (8),(9)

The run where solver is given at each step an informed guess based on σ_{loc} performs clearly less iterations, as second row of Table (8) and (9) show.

3.4.2 Large Δt: six months step

Parameters of simulations are now:

Parameter	Value
Expiry	5y
Spot	95
Strike	100
DriftRate	0.02
DiscountRate	0.03
Time Steps	10
Initial Guess	0.1

Table 11: European Call Parameters

What we are doing is evolving a long dated (5ys) product over 10 six-monthly time steps.

	T2	T3	T4	T5	T6	T7	T8	T9	T10
#	161	111	55	33	130	57	93	59	93
1	0.141314	0.167151	0.170708	0.15742	0.194463	0.174073	0.173432	0.245696	0.270224
2	0.125507	0.133398	0.139901	0.136094	0.152325	0.140265	0.148825	0.178196	0.181591
3	0.203276	0.220218	0.224211	0.222262	0.243989	0.238506	0.251209	0.271743	0.275578
4	0.233692	0.247704	0.250506	0.244479	0.266593	0.263669	0.269816	0.29488	0.29947
5	0.225183	0.23155	0.227432	0.220027	0.245951	0.233494	0.234129	0.259401	0.26663
6	0.175864	0.181859	0.180735	0.176198	0.189037	0.17519	0.19486	0.238205	0.246354
7	0.157749	0.16739	0.164912	0.164481	0.180242	0.170901	0.168978	0.205623	0.216758
8	0.0770942	0.0919448	0.0954021	0.0894135	0.110211	0.101287	0.103127	0.127635	0.131737
9	0.0866553	0.0880101	0.0824594	0.0876465	0.0888038	0.0994335	0.0998971	0.119367	0.119513
10	0.0790048	0.0651707	0.0704146	0.0676514	0.0612827	0.0791848	0.0894498	0.0806548	0.0712884
11	0.105132	0.0699908	0.0781145	0.0960067	0.114571	0.104854	0.11478	0.110644	0.108704
12	0.226769	0.208468	0.208491	0.198166	0.199038	0.183265	0.191326	0.23359	0.244911
13	0.20426	0.188318	0.184153	0.198665	0.199085	0.189772	0.199558	0.221695	0.225863
14	0.136381	0.13439	0.143148	0.152599	0.164547	0.161931	0.163387	0.187787	0.186841
15	0.139215	0.131264	0.13262	0.13951	0.134859	0.139805	0.124764	0.12292	0.123347
16	0.117749	0.108575	0.107023	0.111662	0.138676	0.149282	0.163549	0.169138	0.174551
17	0.119787	0.111994	0.115675	0.112562	0.135068	0.140282	0.134686	0.140932	0.144665
18	0.0853715	0.0905634	0.0854347	0.0862749	0.0951518	0.0975518	0.119181	0.11372	0.122654
19	0.101746	0.0969998	0.0981446	0.102664	0.1265	0.128723	0.136646	0.139781	0.155851

Table 12: LevMar reconstructed σ_{loc}: vanilla case (initial guess 0.1)

	T2	T3	T4	T5	T6	T7	T8	T9	T10
#	196	42	67	39	57	53	60	39	32
1	0.260048	0.227131	0.21535	0.17842	0.209181	0.199255	0.198427	0.255063	0.281298
2	0.249996	0.237905	0.237066	0.224762	0.242655	0.239271	0.237947	0.255074	0.261344
3	0.256595	0.244489	0.233497	0.218135	0.236947	0.225911	0.221857	0.244396	0.253567
4	0.220663	0.203111	0.194394	0.181787	0.190133	0.184769	0.182129	0.208014	0.217438
5	0.219526	0.192887	0.187856	0.163331	0.180209	0.175082	0.172236	0.2016	0.21209
6	0.173095	0.147713	0.142767	0.121859	0.172209	0.158337	0.154463	0.179683	0.189594
7	0.131123	0.113778	0.124932	0.115991	0.145214	0.147715	0.149673	0.187391	0.199163
8	0.0710737	0.110886	0.117076	0.108983	0.121706	0.12777	0.126805	0.159648	0.171988
9	0.103239	0.108545	0.121109	0.121383	0.135094	0.129724	0.135481	0.163117	0.175166
10	0.0855599	0.0827847	0.102699	0.107913	0.105501	0.090132	0.102099	0.107876	0.117504
11	0.0983787	0.0803654	0.093753	0.113918	0.124342	0.12448	0.127107	0.14489	0.151906
12	0.194594	0.15464	0.154936	0.154078	0.152779	0.137059	0.142269	0.168653	0.175991
13	0.211691	0.179306	0.171728	0.164643	0.170614	0.174931	0.181601	0.184895	0.197526
14	0.143026	0.098936	0.0814906	0.0802302	0.0776772	0.0757661	0.0820418	0.113662	0.117322
15	0.170052	0.138635	0.127341	0.119517	0.133505	0.138303	0.141943	0.158087	0.161167
16	0.225481	0.199048	0.190232	0.185877	0.210266	0.211432	0.210291	0.221068	0.22663
17	0.173836	0.165063	0.165154	0.154468	0.159352	0.157601	0.14948	0.170007	0.171275
18	0.184808	0.174395	0.164423	0.15446	0.161149	0.1552	0.153325	0.158723	0.160254
19	0.190941	0.14391	0.123326	0.105656	0.125889	0.143202	0.130968	0.142678	0.128369

Table 13: LevMar reconstructed σ_{loc}: feeding pseudo analytical Local Volatility

Initial Guess	Total Time	%Error
Flat 0.1	22283 millis	-0.000316%
σ_{loc}	16720 millis	0.000000%

Table 14: Summary for runs

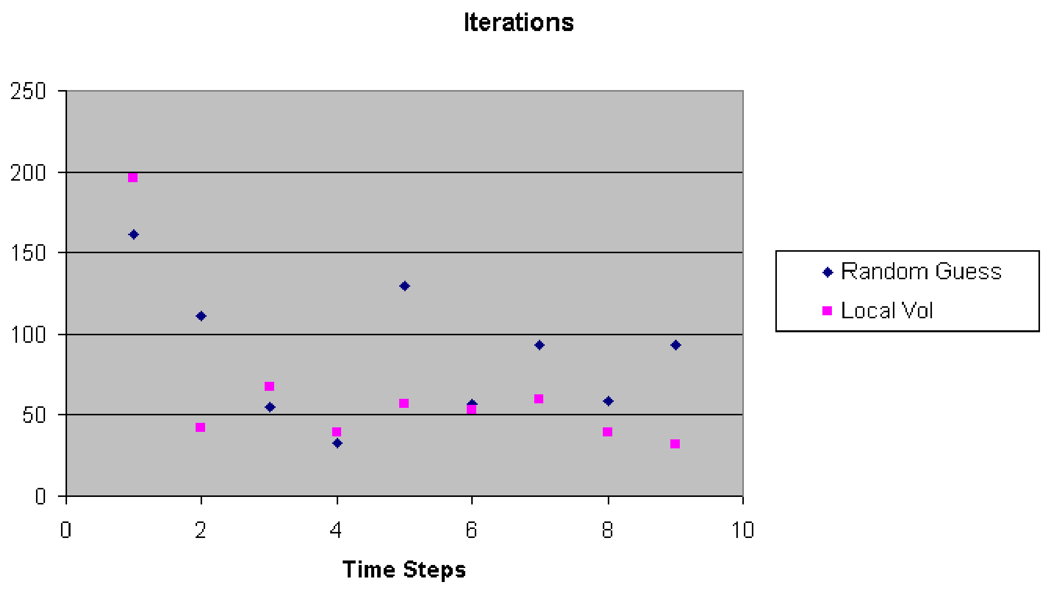

Figure 9: Solver Iterations with/without Local Vol first guess

The above tables and figures deserve a brief comment: Table (12) shows σ_{loc} vectors at each time step as coming from unaided solver, with random initial guess = 0.1, whereas Table (13) shows the same results in case of a solver where Local Vol is given as initial guess. The number of iterations performed by Levenberg Marquardt at each time step are displayed in second row of each table and the total cost of runtime is given in Table (14) along with percentage difference, as ratio of Spot, of analytical and solver price. Figure (9) gives visual comparison of number of solver iterations at each time step.

Further analysis would be necessary to compare vectors of both tables across time steps. Only a few considerations are offered: solver moves in left-to-right marching but we have imposed a particular structure of fit in objective function that is meant to start fitting more aggressively the center and to smooth out tails. That is probably reflected in the resulting calibrated σ_{loc} at each step. There is more: it seem that, notwithstanding results in [Section 3.3.3], shapes of terminal vectors at T10 are closer than shapes of (pairwise) previous steps vectors. A proper L^2 analysis would be necessary to substantiate those consideration and we have to leave question open.

3.5 Concluding remarks

“Wovon man nicht sprechen kann, darüber muss man schweigen“[38]

Ludwig Wittgenstein

The famed Proposition 7.0 of Wittgenstein’s “Tractatus Logico-Philosophicus” [37] fits the solver’s approach quite nicely.

On the other hand, the results of [Section 3.4.1 and 3.4.2] seem to be positive, in the sense that our Local Volatility Function does possess some information that is truly useful to the otherwise agnostic minimizer. One may, however, argue rather differently on the premise that the target of our investigation can be twofold.

On the one hand, we may be interested in helping the solver to speed up the identification of a functional form for the local volatility: this target is broadly equivalent to the confession that as a large step Δt is information lossy in any case, by feeding some reasonable guess we are helping the algorithm traverse the solution space more quickly with no further theoretical utterances.

On the other hand, we may want to be bolder and, as we were saying when discussing solver’s issues, try to constrain the problem so as to find not only a quick solution but also a “good” one.

As regards minimalistic target of runtime efficiency, results seem to comfort idea that having an accurate and fast Local Volatility reconstruction, well tested and tuned in lattices, *does speed up computation reducing overhead in MonteCarlo solver.*

The bolder target is not assisted *as yet* by many results: the fact that terminal densities do not allow us to recover full diffusion and, on the contrary, the fact that uniqueness is *a priori* lost in our setup, do cast a dim light on the whole exercise. More investigation than we had time to perform is needed to achieve some intermediate target, i.e. constraining the Local Volatility in a form like: “Lattice σ_{loc} function + error term”. In any case, what we have produced already may be further examined along the following lines:

- replicating computations in Tables (12) and (13) multiple times *with different initial seed in random sequence generator*;
- comparing standard deviations of terminal σ_{loc} vector in both setups;

The hypothesis that we did not have time to test is that *even if playing in the probabilistic ground and by the MonteCarlo rules*, the results coming from the solver where a lattice σ_{loc} is fed as guess may prove to be less noisy in an L^2 sense than results of an unaided minimization. This will confirm our idea that, even if multiple transitions correspond to same diffusion, *there exist ways of constraining the problem in the direction of a less noisy outcome if we posses an analytic Local Volatility Function.*

We did not have time to investigate another interesting facet of the problem at hand: we could regress results on *multiple noise structures* and see how different coverings of sample space may constrain variance of terminal σ_{loc} vector even more. At the same time, in so doing we may possibly introduce some spurious correlations in need of further analysis, especially if low-discrepancy numbers are used with no prior reflection on how they remap, in affine translations, the hypercube to be sampled.

[38] “Whereof one cannot speak, thereof one must be silent”

4 Results and further comments

4.1 Introduction and Scope

In this final section we will discuss the results of our work showing figures about how accurate is our Local Volatility reconstruction in the Finite Difference framework. Besides Black-Scholes, numbers of percentage error w.r.t. analytical price will be given for all four its incarnations i.e.:

(i) σ_{loc} from Option Prices: quadratic fit;

(ii) σ_{loc} from Option Prices: cubic spline fit;

(iii) σ_{loc} from Market Implied Vols: quadratic fit;

(iv) σ_{loc} from Market Implied Vols: cubic spline fit;

We have built some more interpolating routines and algorithms in code base; our impression is that the above 4 incarnations (besides the obvious benchmark, i.e. Black-Scholes) are the most representative and so results will be given for those only.

[Section 4.2.1] will explore a mildly OTM call option case while [Section 4.2.2] will give results for deeply OTM call option case.

Various points will be raised and discussed in both of them:

- accuracy of schemes without/with Grid Point shift to accommodate Strike;
- various discretization (from crudest (X,T) = (5,2) up to (X,T) + (80,64));
- peculiar PDE phenomenon of P$\acute{e}$clet/Reinold Number;
- differences across fits (Quadratic vs. Cubic Spline) and across Derivations (Dupire vs. Implied Vol)

[Section 4.2.3] will fix a discretization in all five schemes and it will show figures of how the fit of each one of them changes across Strike-Space.

Finally, [Section 4.2.4] will explore the slightly orthogonal point of how fast the routines above run.

4.2 Lattices Approximation of Local Volatility

The following tables show results of FDM methods with varying time and space mesh. All results in tables below are given as

$$\frac{(Analytical\ Price-Finite\ Difference\ Price)}{Spot}$$

in percentage points. As said previously we are interested in crude Time/Space grids, so we will start by a (X,T)=(5,2) scheme and grow up to a (X,T)=(80,64) by refining our mesh in Time and Space via sampling the same points plus some more at each refinement.

4.2.1 Mildly OTM Option case

Let us start with the following case of a mildly OTM Call:

Parameter	Value
Expiry	1y
Spot	95
Strike	100
DriftRate	0.01
DiscountRate	0.02

Table 15: European Call Parameters

Analytical price is 2.963202

$T\backslash X$	5	10	20	40	80
2	-0.80596%	-0.19718%	0.10005%	-0.22441%	-0.11993%
4	-0.81077%	-0.23790%	-0.01761%	-0.06713%	-0.02464%
8	-0.81125%	-0.24161%	-0.02577%	-0.02532%	0.00462%
16	-0.81134%	-0.24226%	-0.02718%	-0.02676%	0.00042%
32	-0.81135%	-0.24240%	-0.02748%	-0.02707%	0.00012%
64	-0.81136%	-0.24243%	-0.02755%	-0.02714%	0.00005%

Table 16: Black-Scholes Flat Vol case

$T\backslash X$	5	10	20	40	80
2	1.40784%	-0.35459%	-0.09809%	-0.23572%	-0.06542%
4	1.41022%	-0.36013%	-0.05656%	-0.08956%	-0.02242%
8	1.41035%	-0.35724%	-0.02962%	-0.04615%	-0.01321%
16	1.41063%	-0.34753%	-0.02212%	-0.03365%	-0.00634%
32	1.41068%	-0.34689%	-0.02041%	-0.02826%	-0.00173%
64	1.41069%	-0.34672%	-0.01938%	-0.02674%	0.00096%

Table 17: Errors in Dupire's case: Quadratic Fit

$T\backslash X$	5	10	20	40	80
2	-0.82663%	-0.13349%	0.03860%	-0.26978%	-0.16149%
4	-0.84283%	-0.18207%	-0.04220%	-0.06966%	-0.01991%
8	-0.84353%	-0.18248%	-0.04448%	-0.03674%	0.00065%
16	-0.84362%	-0.18251%	-0.04489%	-0.03692%	-0.00097%
32	-0.84363%	-0.18252%	-0.04498%	-0.03695%	-0.00100%
64	-0.84364%	-0.18252%	-0.04499%	-0.03696%	-0.00101%

Table 18: Errors in Implied Vol case: Quadratic Fit

$T\backslash X$	5	10	20	40	80
2	0.34505%	-0.69969%	-0.36376%	-2.88986%	-2.91848%
4	0.32893%	-0.72381%	-0.24370%	-0.25148%	-0.37099%
8	0.33636%	-0.72347%	-0.22045%	-0.11514%	-0.12170%
16	0.33863%	-0.71954%	-0.20848%	-0.07712%	-0.04430%
32	0.34099%	-0.71935%	-0.20396%	-0.08330%	-0.01836%
64	0.35631%	-0.71713%	-0.20175%	-0.07429%	-0.00949%

Table 19: Errors in Dupire's case: Cubic Spline Fit

$T\backslash X$	5	10	20	40	80
2	-0.82631%	-0.14734%	0.02133%	-0.27273%	-0.16215%
4	-0.84429%	-0.19490%	-0.05164%	-0.07089%	-0.02010%
8	-0.84412%	-0.19327%	-0.05218%	-0.03775%	0.00050%
16	-0.84410%	-0.19295%	-0.05232%	-0.03786%	-0.00110%
32	-0.84410%	-0.19288%	-0.05235%	-0.03788%	-0.00113%
64	-0.84410%	-0.19287%	-0.05235%	-0.03789%	-0.00113%

Table 20: Errors in Implied Vol case: Cubic Spline Fit

Let us comment on the above results stressing a few points.

Sub-basis point agreement also for crude Time meshes The first three tables show that, given enough *Space discretization*, a sub-basis point accuracy can be achieved already for 8 time steps in Implied Vol case, or 16 time steps in Dupire case. We do not seem to be in such a good shape in case of Spline Fit where, notwithstanding space discretization, the crude time mesh is likely to affect results in the wrong sense, thus making them less accurate.

Quadratic Fit wins over Spline Fit As previous point was effectively stressing, Quadratic Fit seems to be better than Spline Fit *in all cases* considered. Both Dupire and Implied Vol result in a better agreement with the curvature of Vol Surface in the Quadratic Fit case than in the Cubic Fit case. We have argued that this ostensible supremacy is to be credited to the fact we are discarding higher order terms in the Taylor Expansion that leads to the Fokker-Planck equation; in any case, even disregarding the less accurate fit produced, Spline algorithm[39] is still more than an order of magnitude *slower* than Lagrange quadratic. We will give more precise estimates in [Section 4.2.4].

Dupire's derivation becomes inaccurate at edges Ultimately, both Dupire and Implied Vol formulas use the Market Implied Volatility as coming from our reference generator, i.e. SABR. But Dupire channels it through an additional non-linear function (Black-Scholes for Option Prices) which magnifies oscillation or rough behavior on the edges; something that is acutely reflected in Spline case which becomes grossly inaccurate also for quite fine space meshes. The same inaccuracy does not seem to surface in Implied Vol case, where the derivation, although analytically more involved, is numerically better behaved. In our reconstruction of things, the non-linear function amplification of inaccuracy may be attributed to machine epsilon: were we to posses Turing machines that work over Cantor's $\mathbb{R}$ instead of simply $\mathbb{Q}$, we could possibly avoid such inconsistencies[40].

[39]We have optimized Spline Fits down to the bare metal by coding it in procedural C, not even in object-oriented C++

[40]Pythagoreans suffered a big stroke when they discovered the incommensurability of the side and the

Less Sensitivity in Time than Space Another fact that can easily be noticed from tables is the minor sensitivity of results to Time discretization than to Space discretization: in most of the cases, i.e. with the exception of particularly noisy Spline Fit in Dupire's case, the results do not show a pronounced sensitivity to time discretization refinement. It may be recalled that the differential operator we are discretizing is second order in Space and first order in Time: that according to us is the reason why there is far more sensitivity to Space Grid refinements.

P$\acute{e}$clet Number Another interesting fact from data is the emerging pattern of non monotonicity of convergence improvement across refinements. In other words it may happen that a *finer* discretization results in *less accurate* figures. In the literature of PDE, a similar phenomenon is often labeled as *Péclet/Reinold Number* of a differential operator: there exists boundaries for drift and diffusion coefficients outside which increasing the discretization does not necessarily cascade in increasing the accuracy of result. That seems to be the case here.

We have spent considerable time tuning our discretization. We cannot talk of all the improvements that we have deployed: it is enough to spend some words commenting results in case of adjusting Grid to capture Strike as Tavella and Randall [35] suggest.

The following two tables display results for Quadratic Fit in Dupire's case and Implied Volatility case. Contrary to the suggestions in [35], the coefficient by which the grid has been shifted is *different* for each one of them: we could not replicate the above authors reasoning in regions of high curvature with a non-flat volatility, so we had to progress heuristically.

$T\backslash X$	5	10	20	40	80
2	1.49684%	0.10594%	-0.09158%	-0.18373%	-0.06877%
4	1.51855%	0.10122%	-0.03620%	-0.04593%	-0.02404%
8	1.51795%	0.10856%	-0.00788%	-0.00918%	-0.01469%
16	1.51924%	0.11363%	-0.00005%	0.00363%	-0.00750%
32	1.51949%	0.11501%	0.00160%	0.00930%	-0.00273%
64	1.51951%	0.11509%	0.00214%	0.01097%	0.00004%

Table 21: Errors in Dupire's case: Quadratic Fit with shift of Grid Points

$T\backslash X$	5	10	20	40	80
2	2.03081%	0.31761%	0.09134%	-0.15421%	-0.16069%
4	2.06528%	0.27751%	0.00792%	-0.02140%	-0.01965%
8	2.06992%	0.27561%	0.00528%	0.00352%	0.00125%
16	2.07077%	0.27543%	0.00481%	0.00327%	-0.00050%
32	2.07095%	0.27540%	0.00471%	0.00322%	-0.00053%
64	2.07100%	0.27539%	0.00469%	0.00321%	-0.00054%

Table 22: Errors in Implied Vol case: Quadratic Fit with shift of Grid Points

A few remarks are appropriate:

Effectiveness of Adjusting Grid Points for Strike We achieve an overall better result in all three explored cases (two only displayed). Dupire's improvement is striking: it results in a figure which is an order of magnitude more accurate than the unoptimized

diagonal of a square; the great mathematician George Cantor was ultimately driven out of Mathematics by the *Continuum Hypothesis*. Real Numbers are dangerous, but unavoidable: "Nobody will expel us from the paradise that Cantor has created for us" was Hilbert's position.

version of the same algorithm. Implied Volatility derived Local Vol improves as well, although not as hugely: still it is quite impressive how such a simple strategy delivers such good results (half an order of magnitude better in this case).

Heuristic result We could not develop a mathematically satisfying (i.e. predictive) theory about how the Grid should be shifted in all three cases: we worked out a set of viable possibilities and tested them sequentially until some pattern emerged. We noticed that the same Grid Points shift does not necessarily result in an improvement of both the incarnations of Local Vol, i.e. their internal mechanics can be such that a particular strategy works in Dupire's case say but fails to give any improvement in Implied Volatility's case.

4.2.2 Deeply OTM Option case

Let us now change parameters and consider the following case of deeply OTM Call:

Parameter	Value
Expiry	1y
Spot	95
Strike	130
DriftRate	0.01
DiscountRate	0.02

Table 23: European Call Parameters

Analytical price is 0.1610039

$T\backslash X$	5	10	20	40	80
2	0.70440%	0.10180%	0.04155%	0.02554%	0.01946%
4	0.70886%	0.16710%	0.03952%	0.00708%	-0.00218%
8	0.70931%	0.17293%	0.04322%	0.01100%	0.00194%
16	0.70939%	0.17395%	0.04386%	0.01178%	0.00278%
32	0.70940%	0.17416%	0.04400%	0.01195%	0.00296%
64	0.70941%	0.17421%	0.04403%	0.01199%	0.00300%

Table 24: Errors in Black-Scholes Flat Vol case

$T\backslash X$	5	10	20	40	80
2	1.05870%	-0.01777%	-0.03338%	-0.01115%	-0.00510%
4	1.06557%	0.02123%	-0.02587%	-0.01653%	-0.00775%
8	1.06843%	0.02859%	-0.01501%	-0.01278%	-0.00686%
16	1.06858%	0.03868%	-0.01317%	-0.00908%	-0.00486%
32	1.06886%	0.04047%	-0.01249%	-0.00779%	-0.00339%
64	1.06892%	0.04086%	-0.01210%	-0.00761%	-0.00261%

Table 25: Errors in Dupire's case: Quadratic fit

$T\backslash X$	5	10	20	40	80
2	0.59917%	-0.08474%	0.09318%	0.08856%	0.08473%
4	0.57854%	-0.13418%	0.02913%	0.01651%	0.00958%
8	0.57729%	-0.13523%	0.02382%	0.01005%	0.00311%
16	0.57709%	-0.13538%	0.02288%	0.00926%	0.00241%
32	0.57705%	-0.13541%	0.02269%	0.00910%	0.00227%
64	0.57705%	-0.13542%	0.02265%	0.00906%	0.00224%

Table 26: Errors in Implied Vol case: Quadratic fit

$T\backslash X$	5	10	20	40	80
2	0.78215%	0.13061%	-0.13831%	-0.14972%	-0.15628%
4	0.79766%	0.12913%	0.04600%	-0.04755%	-0.09181%
8	0.80090%	0.12878%	0.04781%	0.01275%	-0.02946%
16	0.80244%	0.12904%	0.05301%	0.02328%	-0.00359%
32	0.80432%	0.12902%	0.05257%	0.02754%	0.00492%
64	0.80375%	0.12903%	0.05147%	0.02713%	0.00722%

Table 27: Errors in Dupire's case: cubic Spline fit

$T\backslash X$	5	10	20	40	80
2	0.99030%	-0.17637%	0.10082%	0.09092%	0.08532%
4	0.92438%	-0.15742%	0.03745%	0.01984%	0.01043%
8	0.90970%	-0.15317%	0.03441%	0.01389%	0.00410%
16	0.90663%	-0.15237%	0.03383%	0.01319%	0.00342%
32	0.90588%	-0.15220%	0.03371%	0.01305%	0.00329%
64	0.90568%	-0.15216%	0.03369%	0.01302%	0.00326%

Table 28: Errors in Implied Vol case: cubic Spline fit

These following three display figures for Strike adjusted Grids:

$T\backslash X$	5	10	20	40	80
2	-1.29974%	0.03312%	0.02318%	0.01971%	0.01824%
4	-1.29336%	0.06709%	0.01280%	-0.00058%	-0.00411%
8	-1.29272%	0.07090%	0.01547%	0.00296%	-0.00011%
16	-1.29261%	0.07158%	0.01599%	0.00372%	0.00073%
32	-1.29258%	0.07172%	0.01610%	0.00388%	0.00090%
64	-1.29258%	0.07175%	0.01613%	0.00392%	0.00094%

Table 29: Errors in Black-Scholes Flat Vol case: shift of Grid Points

$T \backslash X$	5	10	20	40	80
2	-2.29062%	-0.09830%	-0.02848%	-0.00795%	-0.00009%
4	-2.22982%	-0.12429%	-0.03002%	-0.01001%	-0.00035%
8	-2.22087%	-0.12429%	-0.02789%	-0.00849%	-0.00019%
16	-2.22041%	-0.12356%	-0.02712%	-0.00723%	0.00008%
32	-2.22037%	-0.12275%	-0.02638%	-0.00674%	0.00027%
64	-2.22036%	-0.12262%	-0.02622%	-0.00659%	0.00038%

Table 30: Errors in Dupire case: Quadratic fit with shift of Grid Points

$T \backslash X$	5	10	20	40	80
2	-0.58055%	-0.08120%	0.07865%	0.08058%	0.08698%
4	-0.49965%	-0.00640%	0.01934%	0.01128%	0.00997%
8	-0.49044%	-0.00374%	0.01763%	0.00528%	0.00316%
16	-0.48874%	-0.00312%	0.01705%	0.00457%	0.00240%
32	-0.48837%	-0.00299%	0.01693%	0.00442%	0.00225%
64	-0.48828%	-0.00295%	0.01691%	0.00439%	0.00222%

Table 31: Errors in Implied Vol case: Quadratic fit with shift of Grid Points

Similar considerations as in previous case apply here as well. Some more remarks are necessary:

Non monotonicity of increase in accuracy across Space refinements *ceteris paribus* The phenomenon we have labeled previously as "effect of Péclet number" seems to be at work here for time discretizations in the Black-Scholes case, both for vanilla version and for Strike-adjusted-Grid case.

Dupire and Implied Vol beat flat BS in region of high curvature We are considering here the version where there grids are not Strike-adjusted. Contrary to what happens in previously examined mildly OTM case, here the regions of high curvature make the flat case approximation actually slightly worse than the Local Volatility Case, where an attempt is made by the framework to capture the curvature.

Adjusting Grid for Strike: a global improvement Past the initial really crude space-grids and facing some more resistance to improvements in Dupire's case, the pairwise comparison of Tables (24) and (29), Tables (25) and (30), Tables (26) and (31) shows that adjustments of grid for strike result in global improvements across all combinations of Space/Time for which we have provided figures.

4.2.3 Fits when Moving Across Strike Space

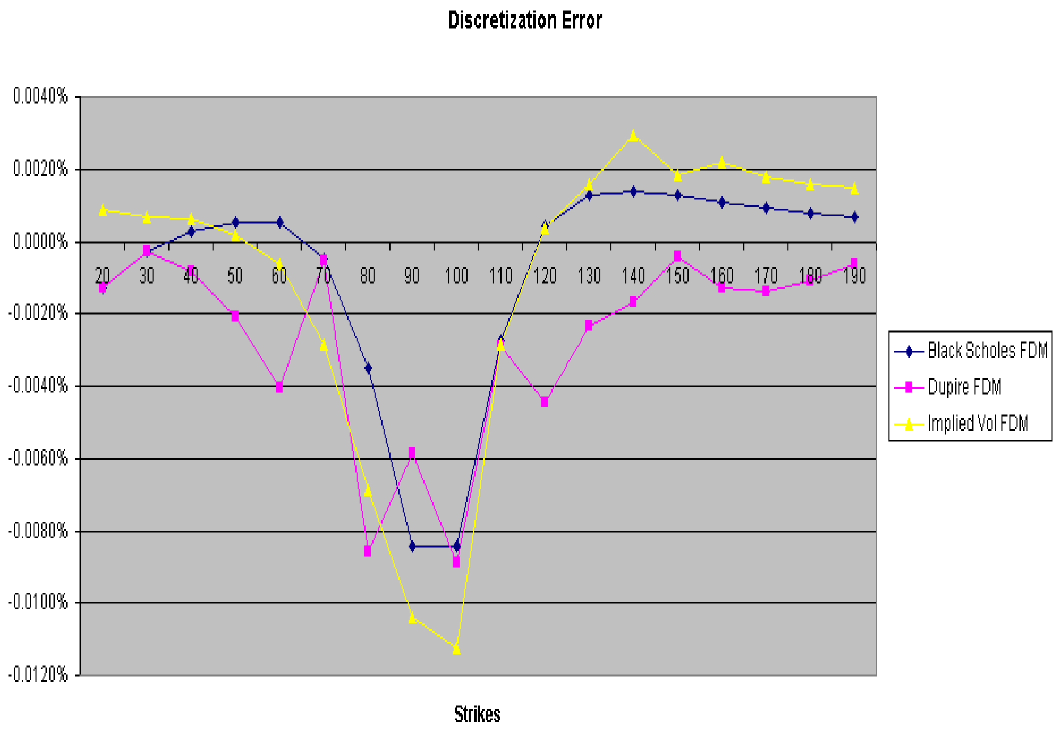

Figure 10: Discretization error in Grids (X,T)=(80,64) across Schemes

Figure 10 shows how Finite Difference Schemes perform in Strike Space for all three the volatility structures we possess (i.e. Flat, Dupire's Local Vol or Implied Vol Local Vol). Parameterization of underlying runs is the one in Table (15) and there is no tuning of Vanilla Scheme. The lines show percentage difference, as a ratio of Spot, of analytical price and finite difference price.

By examining the graph we may get an intuition about how big is the portion of total error due to *discretization of differential operator* and how big is due to *skew structure degrading our fit*. We did not have time to perform an analytical disaggregation of the two components. Suffice it to notice that, in regions of proximity to Strike (here: K=100) the discretization error is more pronounced (but still under a basis point in almost all cases: tuning may be necessary to reduce it in cases where it is not).

The Implied Vol Derivation, because of the better numerical tractability explored somewhere else[41], appears to follow more closely the flat volatility case.

4.2.4 Remarks on Computational Time and Efficiency

From a computational perspective, the Black-Scholes formula requires no more than two Normal CDF evaluations. Easily available and robust formulas exist to perform the integral and overhead of non-linear function is almost negligible.

If we are comparing, though, the flat volatility case with the local volatility case, whilst the former setup uses a scalar parameter given as input, the latter, even in the case of simplest fits, has to perform a sometimes quite laborious calculation to approximate partial derivatives.

[41]See [Section 4.2.1]

We have expended a considerable effort in making our runtime not only *accurate* but *fast*. The language, as we will explore in Appendix, has helped us considerably: all sorts of optimizations are possible within a C++ compiler. Still, taking for instance the natural cubic spline (whose results we have shown before) the computation on a 80 points Space grid, say, involves fitting a cubic polynomial through all the points at each step in the discretization and the cost of such operations may easily escalate. All sorts of strategies have been attempted to make the code fast: removing invariants from loops, replacing branching statements with hierarchies of objects from computationally demanding segments, rewriting the STL container-based computations in a C-style pointer-based dialect[42]. Virtual dispatching of methods has been kept to a minimum while preserving good architectural design.

We cannot enter the talk here about computational complexity, for reason of time and space constraints. A few figures will suffice to inform on the runtime cost of our setup[43].

Local Vol	Milliseconds
Processor	Intel(R) Xeon 2.79 GHz, 2 Gb of RAM
OS	Microsoft Windows XP (SP3)
Compiler	Microsoft Visual Studio C++ 2008: Version 9.0
Configuration	Release mode

Table 32: Runtime

Sequentially producing the 20 prices contained in first 4 rows of tables in [Section 4.2.1] requires:

Volatility Approximation	Millis
Flat Black-Scholes	≈ 60
Dupire Quad. Fit	≈ 200
Implied Vol Quad. Fit	≈ 200
Dupire Spline Fit	≈ 6000
Implied Vol Spline Fit	≈ 4500

Table 33: Speed of Local Volatility Calculation

As the above figures show, while Quadratic fit in both cases is not *vastly* more expensive than flat case vol in Black-Scholes case, Spline fit is one order of magnitude *slower* than Quadratic fits case. It may be argued that tables involve calculations for crude as well as fine grids (from (X,T)=(5,2) to (X,T)=(80,16)and that it is only when finer grids are traversed that Spline performance quickly degrades.

A simple way out of this would be to let Spline fit only a subset of points and *not* the whole discretized collection. If, for example, we have a 80 space nodes grid, the points far out the target computation region in grid are *not* contributing much to the global shape of cubic polynomial. It may then be sensible to restrict Spline fit to a subset of closer points, thus recovering performance on a par with Quadratic Fit and ultimately Black-Scholes case.

[42]In order to adhere to bank standards, we did not use C++ template metaprogramming and compile-time polymorphism.

[43]Figures refer to averages of 5 runs

A Appendix

A.1 Implementation details and Libraries used

This section will briefly talk about

- the software landscape used to build the project
- the library architecture

A.1.1 C++, STL and Boost

Despite a lot of prophesying about the death of it, C++ seems as vital and vibrant a language as ever. Some of the most recent developments, incorporating in the Standard Library a wealth of mathematical infrastructure, like Orthogonal Polynomials, Probability Distributions and even an enhanced Random Number support, can only be read as an hint that the language intends to maintain its entrenched position as the platform of choice for numerical and scientific computing. In finance, where the code base of major industry players counts sometimes in the million of lines, the position of C++ seems hardly at stake and surely newer generation languages, in particular virtual machine based like Java and C# do not seem to us in the position to challenge its rôle[44]. That is the reason why we have adopted C++ as the language of our project. Moreover, STL and in particular Boost libraries (`www.boost.org`) have been used extensively.

A.1.2 LevMar and CBLAS

Levenberg -Marquardt algorithm is easily available from Numerical Recipes book [36], but it is reputed to lack robustness[45]. That is why we have decided to adopt a different library, the open source *levmar* (`http://www.ics.forth.gr/~lourakis/levmar`). Written in C but easily linkable to C++ runtime, it enjoys large community and broad industry usage.

The other piece of software we have used is a linear algebra package, commonly known by the acronym BLAS. CBLAS (`http://www.netlib.org/clapack`), a Fortran-to-C implementation, is intended to perform matrix operations during minimization. Both levmar and CBLAS resulted reliable and well-documented, as free and open-source software usually is.

A.1.3 Random Sequence Generators tests: DieHard

To test the random sequence generators, essentially Boost::mersenne twister and Sobol Brownian Bridge, we have used:

- DieHard (`http://www.stat.fsu.edu/pub/diehard`)

The issue here is of course the normality of the draws. Eventually a standardization procedure may be needed to recenter the sequences to match first and second central moment. More complicated the issue of Skew and Kurtosis, which we have not considered

A.2 GOF patterns

A few standard Design Patterns have been used in code base to help traverse the domain complexity in a professional, non write-only way. Standard reference is [22]; for quantitative finance in particular, [23] is also useful. In our library we have:

[44]Speed matters of course: although virtual machines by means of Just-In-Time compilation and runtime-profiling based optimization seem to improve at high pace, compiled languages are likely to remain faster. We are comparing here languages of the same nature and paradigm, i.e. Object Oriented. A slightly orthogonal topic would be analyzing *functional* languages where the language model may theoretically be more suited to heavily parallel computations, which is the trend in the processor industry, and hence, down the valley, in software engineering.

[45]See for instance: `http://www.lysator.liu.se/c/num-recipes-in-c.html`

Decorator : is used to layer Antithetic Variates on top of Standard Noise Generator;

Factory : has been employed for Random Sequences Generator. Factory returns a RNG Strategy Object -either implemented through Mersenne Twister or Brownian Bridge - depending on runtime input;

Strategy : is used throughout the Local Volatility Function implementations to let different interpolations act on same underlying data structures;

Singleton : is used to have unique SABR term-structure in environment;

Separation of Concerns : Controller Objects (Workers) are in charge of consuming Business Logic Services without being exposed to internal implementation of algorithms.

References

[1] Sergei Kucherenko Balazs Feil and Nilay Shah. Volatility calibration using spline and high dimensional model representation models. *Wilmott Journal*, 1(4), 2009.

[2] Fisher Black. The pricing of commodity contracts. *Journal of Financial Economics*, 3, 1976.

[3] D. Breeden and R. Litzenberger. Prices of state-contingent claims implicit in option prices. *The Journal of Business*, 51, 1979.

[4] Michael J. Brennan and Eduardo S. Schwartz. Finite difference methods and jump processes arising in the pricing of contingent claims: A synthesis. *The Journal of Financial and Quantitative Analysis*, 13, 1978.

[5] Richard L. Burden and J. Douglas Faires. *Numerical Analysis*. Brooks-Cole, 9th edition, 2010.

[6] J. C. Cox and S. A. Ross. The valuations of options for alternative stochastic processes. *Journal of Financial Economics*, 3, 1976.

[7] S. Crepey. Calibration of the local volatility in a generalized black-scholes model with tikhonov regularization. *SIAM Journal on Mathematical Analysis*, 2003.

[8] Emanuel Derman. Modeling the volatility smile. 2006.

[9] Emanuel Derman and Iraj Kani. Riding on a smile. *Risk*, 7:32–39, 1994.

[10] K.R. Vetzal D.M. Pooley and P.A. Forsyth. Convergence remedies for non-smooth payoffs in option pricing. *Journal of Computational Finance*, 6:25–40, 2003.

[11] Daniel J. Duffy. *Finite difference Methods in financial engineering*. John Wiley & Sons, Inc., New York, 2006.

[12] Daniel J. Duffy and Joerg Kienitz. *Monte Carlo Framework: Building customizable high-performance C++ applications*. John Wiley & Sons, Inc., New York, 2009.

[13] Bruno Dupire. Pricing with a smile. *Risk*, 7(1), 1994.

[14] William Feller. Two singular diffusion problems. *The Annals of Mathematics*, 55(1):3–34, 1951.

[15] Jim Gatheral. *The Volatility Surface*. John Wiley & Sons, Inc., New York, 2006.

[16] S. Heston and G. Zhou. On the rate of convergence of discrete-time contingent claims. *Mathematical Finance*, 10(1), 2000.

[17] S. L. Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies*, 6:327–343, 1993.

[18] John C. Hull. *Option Futures and Other Derivatives*. Pearson Education, Upper Saddle River, New Jersey, 8th edition, 2007.

[19] John C. Hull and Allen White. The pricing of options on assets with stochastic volatilities. *Journal of Finance*, 42:281–300, 1987.

[20] G. Riboulet J. Bodeau and T. Roncalli. Non uniform grids for pde in finance. *Groupe de Recherche Opérationnelle,Crédit Lyonnais*, 2000.

[21] Peter Jäckel. *Monte Carlo Methods in Finance*. John Wiley & Sons, Inc., New York, 2002.

[22] Eric Gamma Richard Helm Ralph Johnson and John Vlissides. *Design patterns: elements of reusable object-oriented software.* Addison Wesley, 1994.

[23] Mark Joshi. *C++ Design Patterns and Derivatives Pricing.* Cambridge University Press, Cambridge, 2004.

[24] Mark Joshi. *The Concepts and Practice of Mathematical Finance.* Cambridge University Press, Cambridge, 2nd edition, 2008.

[25] H.B. Nielsen K. Madsen and O. Tingleff. Methods for non-linear least squares problems, 2004.

[26] Peter E. Kloeden and Eckhard Platen. *Numerical Solutions of Stochastic Differential Equations.* Springer, Berlin, 3rd edition, 1999.

[27] Alexander Lipton. *Mathematical Methods for Foreign Exchange: A Financial Engineer's Approach.* World Scientific Publishing Co Pte Ltd, Singapore, 2001.

[28] William A. McGhee. An efficient implementation of stochastic volatility by the method of conditional integration. ICBI Global Derivatives, Paris, 2011.

[29] Robert C. Merton. Option pricing when the underlying stock returns are discontinuous. *Journal of Financial Economics*, 3:125–144, 1976.

[30] Berndt Øksendal. *Stochastic Differential Equations.* Springer, Berlin, 5 edition, 2003.

[31] Andrew S. Lesniewski Patrick S. Hagan, Deep Kumar and Diana E. Woodward. Managing smile risk. *Wilmott Magazine*, pages 327–343, 2002.

[32] Jeff Dewynne Paul Wilmott and Sam Howison. *Option Pricing: Mathematical Models and Computations.* Oxford Financial Press, Oxford, illustrated edition, 1994.

[33] Yoshihiro Saito and Taketomo Mitsui. Stability analysis of numerical schemes for stochastic differential equations. *SIAM Journal of Numerical Analysis*, 33:2254–2267, 1996.

[34] E. Stein and J. Stein. Stock price distributions under stochastic volatility: An analytic approach. *Review of Financial Studies*, 4, 1991.

[35] D. Tavella and C. Randall. *Pricing Financial Instruments: the finite difference method.* John Wiley & Sons, Inc., New York, 2000.

[36] William H. Press Saul A. Teukolsky William T. Vetterling and Brian P. Flannery. *Numerical Recipes The Art of Scientific Computing.* Cambridge University Press, Cambridge, 3rd edition, 2007.

[37] Ludwig Wittgenstein. *Tractatus logico-philosophicus, Werkausgabe Band I.* Suhrkamp Verlag, Frankfurt am Main, 1st edition, 1984.