FOLLOWING THE BAYES PATH TO OPTION PRICING

MARCO AVELLANEDA, ANDREA CARELLI AND FABIO STELLA

Conventional modeling techniques for option pricing have systematic biases resulting from the assumption of constant volatility (homoskedasticity) for the price of the underlying asset. Nevertheless, practitioners seldom use stochastic volatility models since the latter require making unverifiable assumptions about the price process. A different approach consists of ''letting the data speak for itself", i.e. to make a few general assumptions about the process to be modeled, and to exploit the information available from the prices of traded options. In this paper we develop of a non-parametric model for specifying the volatility of the underlying asset based on Feedforward Neural Networks and a Bayesian learning approach. We then develop of an option-pricing model based on this volatility specification. Numerical experiments are presented for the case of the USD/DEM options, accompanied by a graphical analysis of the resulting smiles.

1. INTRODUCTION

Options are some of the most important financial instruments traded in the markets today. The technical aspects of option-pricing theory and practice have attracted the attention of mathematicians, statisticians, physicists and computer scientists.

According to Rubinstein [1985] and Eales et al. [1990], the celebrated Black & Scholes option-pricing formula has systematic and persistent biases. These biases depend upon both the "time to maturity" and the "option moneyness ratio" which is the relationship between the spot and the strike price. Among several plausible reasons accounting for these biases is the one introduced in Hull and White [1987] where the conventional models bias is shown to be consistent with the theory of heteroscedastic volatility.

Models that take into account the stochastic nature of volatility may therefore give rise to improved option trading strategies. The stochastic volatility approach has been pursued by many researchers (see, for example, Heston [1993], Dupire [1994], Avellaneda et al. [1997], and Fouque et al. [1998] and the bibliography therein). An alternative for modeling volatility in the presence of small samples can be developed using neural networks. Nonparametric models offer the advantage of reducing both the number and the complexity of the assumptions concerning the process to be modeled.

Recently, Feedforward Neural Networks (FNNs) have received attention due to their theoretical properties and computational efficiency. The main property of FNNs is that they are ''universal approximators" in the sense described by White [1989]. This means that FNNs, with as many as one hidden layer, are capable of approximating *any* mapping between independent variables and response variables as the size of the data set describing the Data Generating Process (DGP) and the number of neurons of the hidden layer both tend to infinity. At the present time, the limitation in the number of neurons is less keenly felt due to technological enhancement. However, the problem of paucity of data remains. Indeed, in the case of volatility modeling, data is difficult and expensive to get. For this reason it is important that the FNN model is data-efficient, i.e. that it extracts as much information as possible from the data.

The universal approximation property enjoyed by FNNs is an asymptotic result. Unfortunately, it is of little help in estimation problems where increasing in the number of parameters often leads to over-fitting. Among several approaches presented in the literature to deal with this problem, one which seems promising for the task of the volatility modeling is Bayesian learning in FNNs [see Neal 1996].

The aims of this paper are twofold. First, we show how a special class of non-parametric models, the class of FNNs, can be exploited for modeling the heteroscedastic nature of the volatility. Second, we develop a new option-pricing model based on the non-parametric approach. This model also allows for the pricing of more general contingent claims.

Submitted to the *Journal of Computational Intelligence in Finance* (8/4/99)

The rest of the paper is organized as follows. Section 2 provides a brief introduction to FNNs and to the problem of learning on such structures. In Section 3, a new pricing scheme, which exploits the Bayesian learning in FNNs, is introduced. Finally, in Section 4, the proposed pricing scheme is numerically evaluated using data from the over-the-counter USD/DEM currency options market.

2. BAYESIAN LEARNING IN FEEDFORWARD NEURAL NETWORKS

This section introduces FNNs, their graphical and computational structures, and their theoretical properties. Furthermore the Bayesian framework, for solving the learning problem on FNNs, is described.

2.1 FEEDFORWARD NEURAL NETWORKS

A FNN (Figure I) is a layered structure consisting of computing units named artificial neurons (circles) and connections between neurons named synapses (directed links). Artificial neurons, or simply neurons, of the input layer are associated with independent variables or input variables; while the output neurons are associated with response variables or output variables. Without loss of generality, we consider FNNs with one response variable. Synapses are oriented connections linking neurons from the input layer to neurons of the hidden layer and neurons from the hidden layer to output neurons. This means that the information flows from neurons of lower layers to neurons of higher layers and cannot flow between neurons of the same layer or from higher layer to lower layer neurons.

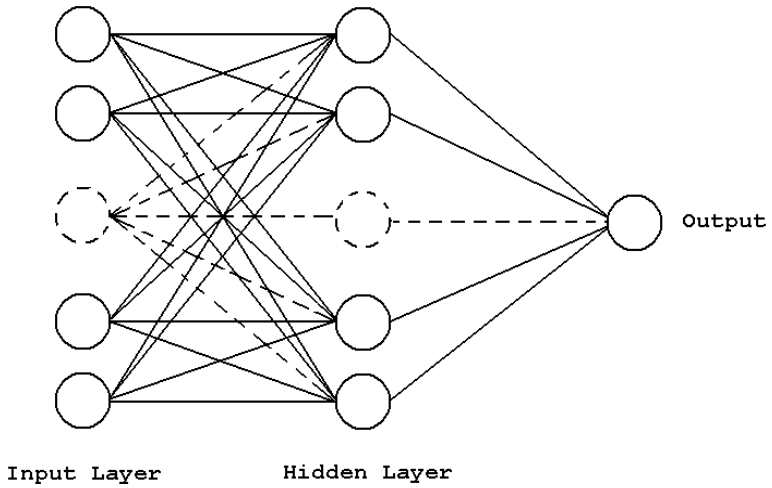

FIGURE I. Single Layer Feedforward Neural Network – A single layer feedforward neural network with a single output neuron.

The FNN graphical structure dictates that for any input the corresponding output value is obtained through propagation from the input layer neurons to the hidden layer neurons and then from the hidden layer neurons to the output neuron. The strength of the synapse from neuron *i* to neuron *j* is determined by means of a real value w_{ij}, named weight. Furthermore, each neuron *j* from the hidden layer, and eventually the output neuron, are associated with a real value b_j named the neuron's bias or threshold and with a nonlinear function, named the transfer or activation function.

Let us now formally describe the structure of FNNs and show how, given an input vector, the network's output is computed. To this extent, consider the single layer FNN with *K* input variables and *H* hidden neurons depicted in Figure II and let:

- $\underline{x} = (x_1, ..., x_K)$, be the input vector (associated with the independent variables),
- $w_{ij}^{(1)}$, be the weight for the synapse from the i^{th} input to the j^{th} hidden neuron,
- $b_j^{(1)}$, be the bias associated with the j^{th} neuron of the hidden layer,
- $w_j^{(2)}$, be the weight for the synapse from the j^{th} hidden neuron to the output neuron,
- $b_j^{(1)}$, be the bias associated with the output neuron.

Furthermore we let $G^{(1)}(\bullet)$ and $G^{(2)}(\bullet)$ respectively be the activation function associated with hidden neurons and the output neuron. Typical choices for the activation function include the logistic function $G^{(1)}(z)=\frac{1}{1+\exp(-z)}$ and the hyperbolic tangent $G^{(1)}(z)=\frac{\exp(z)-\exp(-z)}{\exp(z)+\exp(-z)}$.

If we assume that the activation functions $G^{(1)}(\bullet)$ and $G^{(2)}(\bullet)$ are given and let the vector $\underline{\theta}=\left(w_{11}^{(1)},...,w_{1H}^{(1)},...,w_{K1}^{(1)},...,w_{KH}^{(1)},b_1^{(1)},...,b_H^{(1)},...,w_1^{(2)},...,w_H^{(2)},b^{(2)}\right)$, then the network in Figure II computes the following function of the input vector $\underline{x}$:

$$\hat{y}(\underline{x},\underline{\theta})=G^{(2)}\left(\sum_{j=1}^{H}w_j^{(2)}G^{(1)}\left(\sum_{i=1}^{K}w_{ij}^{(1)}x_i-b_j^{(1)}\right)-b^{(2)}\right) \tag{1}$$

FNNs are *data-driven models* in the sense that their adjustable parameters are selected in such a way as to minimize some performance measure on a given set of training data $\chi=\{(\underline{x}^1,\underline{y}^1),...,(\underline{x}^N,\underline{y}^N)\}$, where $\underline{x}^n\in\Re^K$ and $\underline{y}^n\in\Re$. In particular the task concerned with the selection of the optimal values for the adjustable parameters, exploiting the available data set χ, is called the learning problem.

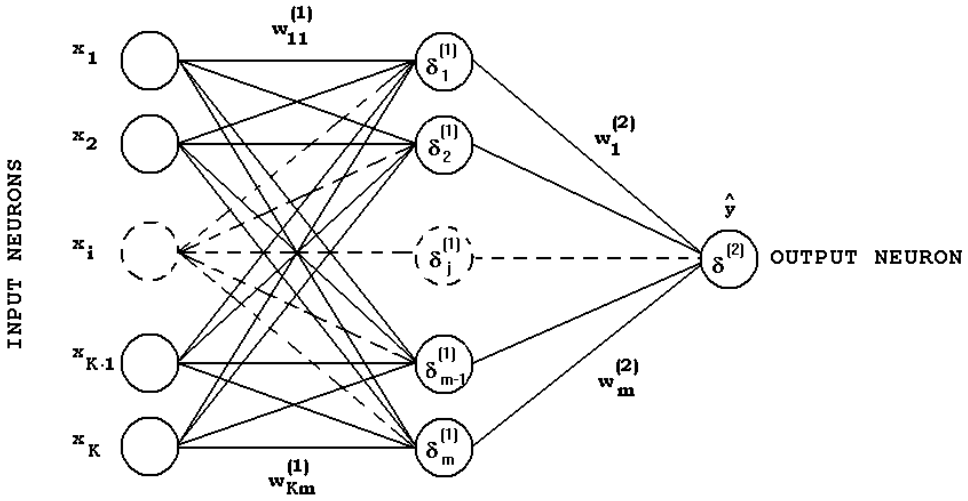

FIGURE II. Single Layer Feedforward Neural Network – A single layer feedforward neural network with the corresponding adjustable parameters.

2.2 THE BAYESIAN FRAMEWORK FOR LEARNING IN FEEDFORWARD NEURAL NETWORKS

Assume that the number of hidden layers, the number of neurons associated to each hidden layer and the nature of the activation functions are all fixed. The problem of learning in FNNs consists of selecting the adjustable parameter vector $\underline{\theta}$ in such a way as to optimize some performance measure dependent upon the available dataset χ.

Conventional maximum likelihood estimation solves the learning problem by selecting a single 'best' set of values $\hat{\underline{\theta}}$ in such a way as to minimize some suitable error function. Within this framework a frequently used measure of error is the sum of squared errors (SSE). This effectively reduces the problem of learning in FNNs to a nonlinear least-squares problem.

Indeed, in such a case, the learning problem consists of selecting the optimal value $\hat{\underline{\theta}}$ for the parameter vector $\underline{\theta}$. We select $\hat{\underline{\theta}}$ such that

$$SSE(\hat{\underline{\theta}}) = \min_{\theta} \sum_{n=1}^{N} \left(\hat{y}^n (\underline{x}, \underline{\theta}) - y^n \right)^2 \tag{2}$$

In the Bayesian framework we consider a probability distribution over the parameter vector $\hat{\underline{\theta}}$ which prior to any received data is expressed through a prior distribution $p(\underline{\theta})$. Once the dataset χ has been observed the posterior probability distribution $p(\underline{\theta} \mid \chi)$ for the parameter vector $\underline{\theta}$ can be written, according to the Bayes' theorem, as follows:

$$p(\underline{\theta} \mid \chi) = \frac{p(\chi \mid \underline{\theta}) p(\underline{\theta})}{p(\chi)} \tag{3}$$

where $p(\chi \mid \underline{\theta})$ represents the likelihood of the data given the parameter. The denominator of (3), $p(\chi) = \int p(\chi \mid \underline{\theta}) p(\underline{\theta}) d\underline{\theta}$, represents a normalization factor, which ensures that the left-hand side of (3) integrates to one over the parameter space. Let us describe learning in the Bayesian framework. The procedure starts from some prior distribution $p(\underline{\theta})$ over the parameter vector $\underline{\theta}$, which expresses a general property such as smoothness for the function to be approximated. Once the observed data χ is available, the prior distribution $p(\underline{\theta})$ can be converted to a posterior distribution $p(\underline{\theta} \mid \chi)$ by means of Bayes's Theorem (3). This task is extremely complex and can be solved through both approximated methods or direct sampling from the posterior distribution through the Markov Chain Monte Carlo (MCMC) approach.

Following McKay [1992], we conclude this section by discussing why the Bayesian framework is useful for data modeling with FNNs. First, it provides a unifying framework for data modeling and allows us to develop probabilistic models that are well matched to the data and therefore to make optimal predictions. Probability theory forces us to make explicit all of our modeling assumptions. Bayesian methods are ``mechanistic": once the model space has been defined, the rules of probability theory give a unique answer which takes into account all the given information. Bayesian inference satisfies the likelihood principle in that our inferences depend only on the probabilities assigned to the data but not on properties of other data sets which could have occurred but did not. Probabilistic modeling also handles uncertainty in a natural manner. There is a unique prescription for incorporating uncertainty about parameters into our predictions of other variables, namely marginalization. Finally, Bayesian model comparison embodies "Occam's razor", the principle that states a preference for simple models over complex ones.

3. A PRICING SCHEME FOR GENERAL CONTINGENT CLAIMS

In this section we propose a new option pricing scheme which allows us to price general contingent claims (plain vanilla, exotic and options traded on illiquid markets). The proposed pricing scheme exploits the universal approximation property and the Bayesian learning framework in FNNs in order to develop a non-parametric model for implied volatility. The importance of using both FNNs and the Bayesian learning framework to model implied volatility could be described as follows. First, we want to exploit all the available market data. Second, we want to make few loose assumptions about DGP which originated the available data set in such a way as to allow the data to speak for themselves. According to this line of thinking the choice for the FNNs, as non-parametric models, and the choice of the Bayesian learning framework are naturally motivated. Indeed while FNNs does not allow us to constrain the maximum complexity of the function to be approximated (i.e. the implied volatility), the Bayesian learning framework allows us to use only the needed degrees of freedom offered by means of FNNs.

The option pricing scheme, described in this section, exploits the two main results of Dupire's Formula (Dupire [1994]) and an implementation of stochastic volatility models on a trinomial tree (lattice) proposed in Avellaneda and Paras [1996]. Dupire's formula allows us to determine a risk-neutral process for the security *S* of the form

$$\frac{dS}{S} = \mu(t)dt + \sigma(S,t)dW \tag{4}$$

where $\sigma(S,t)$ represents the local volatility, from a function that describes the prices of options with a continuum of strikes and expiration dates. The volatility $\sigma(S,t)$ is assumed to be a deterministic function of both the time *t* and the underlying

level price *S*, and $\mu(t)$ is the cost-of-carry. Dupire [1994] shows that the conditional probability distribution for the price for the price and the ''local volatility surface" can be computed as explicit functions of the derivatives of the price function with respect to the strike price and the time to maturity. Dupire's formula, can be viewed as a tool for computing the local volatility surface starting from option prices. Let *K* be the strike price and $C(S,K,T)$ be the call option price associated with the underlying price *S*, the strike price *K* and the time to maturity *T* and assume that the cost-of-carry is zero. Dupire's Formula
states that

$$\sigma(S,t)=\sqrt{\frac{\frac{\partial}{\partial T}C(S,K,T)}{\frac{1}{2}K^2\frac{\partial^2}{\partial K^2}C(S,K,T)}} \tag{5}$$

A discrete computational model for the continuous process described by (4) can be generated using a trinomial lattice.

The pricing scheme originates from the need for a computational model based primarily on option prices. Furthermore the pricing scheme should be such that the resulting price process satisfies the following two conditions:

- compatibility with the implied volatility smile,
- model completeness.

The pricing scheme (Figure III) consists of the following five steps:

1. Implied Volatility Modeling (**IVM**); which is accomplished though FNNs jointly with the Bayesian learning approach,
2. Option Price Computation (**OPC**); which is accomplished through Black's formula (Hull [1995]),
3. Local Volatility Computation (**LVC**); exploits the results from Dupire and is performed by means of (6),
4. Model Discretization (**MD**); which is performed through the Trinomial Stochastic Volatility Model,
5. General Contingent Claims Pricing (**GCCP**); which concerns the computation of option prices for general contingent claims and the consequent comparison between them and the original option prices.

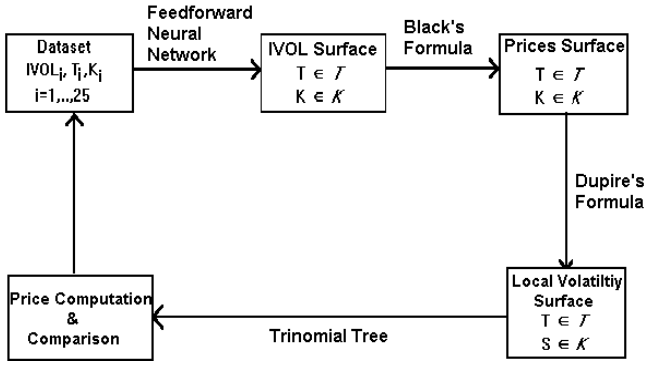

FIGURE III. Pricing Scheme – The five steps procedure for pricing general contingent claims starting from market data.

3.1 IMPLIED VOLATILITY MODELING (IVM)

This computational step exploits the Bayesian learning framework in FNNs for solving the problem of non-parametric estimation of the option-implied volatility. This task is extremely complex for due to fact that dataset coming from the market consists of few observations. We adopt the concept of "infinite networks" described in Neal [1996]. Accordingly, the prior distribution associated with the FNN parameter vector $\underline{\theta}$ is a Gaussian distribution of the form:

$$p(\underline{\theta}) = \frac{1}{Z_{\underline{\theta}}} \exp\left(-\frac{1}{2} \alpha \|\underline{\theta}\|_2^2 \right) \tag{6}$$

where $Z_{\underline{\theta}} = \int \exp\left(-\frac{1}{2} \beta \|\underline{\theta}\|_2^2 \right) d\underline{\theta}$ is a normalization factor which ensures that $\int p(\underline{\theta}) d\underline{\theta} = 1$.

Furthermore we adopt a Gaussian noise model for the likelihood, i.e.:

$$p(\chi \mid \underline{\theta}) = \frac{1}{Z_\chi} \exp\left(-\frac{1}{2} \beta \sum_{n=1}^{N} \left(\hat{y}^n(\underline{x}, \underline{\theta}) - y^n \right)^2 \right) \tag{7}$$

where $Z_\chi = \int \exp\left(-\frac{1}{2} \beta \sum_{n=1}^{N} \left(\hat{y}^n(\underline{x}, \underline{\theta}) - y^n \right)^2 \right) d\chi$ again represents a normalization factor.

It is now possible to write down the expression for the posterior distribution $p(\underline{\theta} \mid \chi)$ of the parameter vector $\underline{\theta}$.
To this extent by substituting (6) and (7) in (3) we obtain the following expression for the posterior distribution:

$$p(\underline{\theta} \mid \chi) = \frac{1}{Z_S} \exp\left(-\frac{1}{2} \beta \sum_{n=1}^{N} \left(\hat{y}^n(\underline{x}, \underline{\theta}) - y^n \right)^2 - \frac{1}{2} \alpha \|\underline{\theta}\|_2^2 \right) \tag{8}$$

where $Z_S = \int \exp\left(-\frac{1}{2} \beta \sum_{n=1}^{N} \left(\hat{y}^n(\underline{x}, \underline{\theta}) - y^n \right)^2 - \frac{1}{2} \alpha \|\underline{\theta}\|_2^2 \right) d\underline{\theta}$.

The distribution in (8) can be computed using a MCMC (Markov chain Monte Carlo) sampling scheme (Neal [1996]).

Usually the available implied volatility data depends upon other market parameters such as the cost-of-carry (interest rate minus dividend rate). It is recommended to transform the data in such a way that the drift of the interest rate and the cost of carry are both zero.

To clarify this point we let p_i be the mid-market option price, k_i be the strike price, T_i be the maturity (expressed in years) and $\Sigma_i = \Sigma_{BS}(k_i, T_i)$ be the implied volatility. The latter is computed based on mid-market prices by means of the Black & Scholes formula. We denote respectively by $\tilde{k}_j$ and T_w the strike price and the time to maturity after having applied this transformation.

According to such a transformation we shall refer, here and in the rest of the paper, to the transformed implied volatility associated with a given transformed strike price $\tilde{k}_j$ and with a given transformed time to maturity T_w with $\tilde{\Sigma}(\tilde{k}_j, T_w)$.

Returning to the implied volatility modeling task we'll use a Single Layer FNN having two input neurons, respectively associated with the transformed strike price $\tilde{k}_j$ and the transformed time to maturity T_w, and one output neuron associated with the approximated transformed implied volatility $\hat{\Sigma}(\tilde{k}_j, T_w)$.
According to this modeling choice the FNN architecture for the implied volatility modeling is depicted in Figure IV.

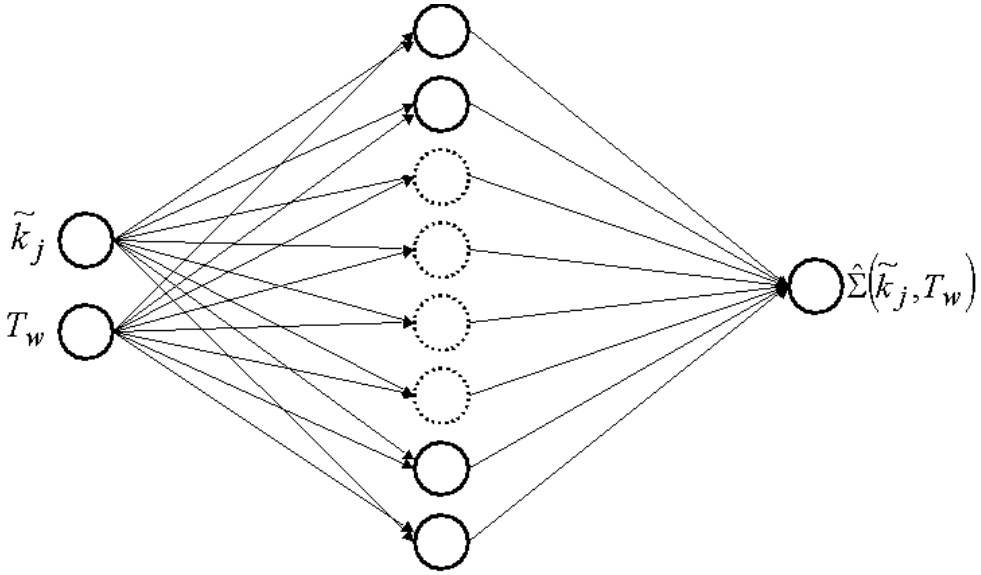

FIGURE IV. Feedforward Neural Network with two input neurons associated respectively with the adjusted strike price $\tilde{k}_j$ and the maturity T_w, and one output neuron associated with the approximated implied volatility $\hat{\Sigma}(\tilde{k}_j, T_w)$.

3.2 OPTION PRICES COMPUTATION (OPC)

This step allows us to go from the implied volatility $\hat{\Sigma}(\tilde{k}_j, T_w)$, associated with a given strike price $\tilde{k}_j$ and a time to maturity T_w, to the corresponding call option price $c_{j,w}$. For computational purposes the input pairs $(\tilde{k}_j, T_w)$ are constrained to belong to a discrete $d \times d$ squared grid which can be defined as follows.

Let $\tilde{K} = \left\{ \tilde{k} \in \Re_+ : \tilde{k}_{\min} = \min_i(\tilde{k}_i) \le \tilde{k} \le \tilde{k}_{\max} = \max_i(\tilde{k}_i) \right\}$ and $\mathrm{T} = \left\{ T \in \Re_+ : 0 \le T \le T_{\max} = \max_i(T_i) \right\}$ where the index i ranges from 1 to 25 (i.e. the number of available data points).

Then the grid is $\tilde{K} \otimes \mathrm{T}$, i.e. the set consisting of all the ordered pairs $(\tilde{k}_j, T_w)$ where $j, w = 1, \ldots . d$.
According to the discretization grid the computation of the call option prices are performed on the $d \times d$ grid through the Black's formula. Formally, for each pair of the grid $(\tilde{k}_j, T_w) \in \tilde{K} \otimes \mathrm{T}$ the corresponding call option price is computed according to the following formula:

$$c_{j,w} = \bar{S}\Phi(d_1) - \tilde{k}_j \Phi(d_2) \qquad \forall j = 1, \ldots . d, \qquad \forall w = 1, \ldots, d \tag{9}$$

where

$$d_1 = \frac{\ln(\bar{S} / \tilde{k}_j) + \frac{T_w}{2}(\hat{\Sigma}_{j,w})^2}{\hat{\Sigma}_{j,w}\sqrt{T_w}} \qquad d_2 = \frac{\ln(\bar{S} / \tilde{k}_j) - \frac{T_w}{2}(\hat{\Sigma}_{j,w})^2}{\hat{\Sigma}_{j,w}\sqrt{T_w}} = d_1 - \hat{\Sigma}_{j,w}\sqrt{T_w} \tag{10}$$

Notice that in (9) and (10) $\bar{S}$ represents the market's spot price associated with the underlying security, while $\Phi(\bullet)$ represents the standard normal cumulative distribution function.

3.3 LOCAL VOLATILITY COMPUTATION (LVC)

According to the pricing scheme (Figure III) this step concerns the computation of the local volatility starting from the call option prices obtained through the OPC step. This task is accomplished by applying the Dupire's Formula to compute the local volatility $\sigma_{j,w}$ associated with all the pairs $(\tilde{k}_j, T_w) \in \tilde{K} \otimes \mathrm{T}$. Formally, for each pair $(\tilde{k}_j, T_w) \in \tilde{K} \otimes \mathrm{T}$ we compute the following quantity:

$$\sigma_{j,w} \equiv \sigma(S_j, t_w) = \sqrt{\frac{g_{j,w}}{\frac{1}{2}\tilde{k}_j^2 f_{j,w}}} \qquad \forall j = 1, \ldots, d, \qquad \forall w = 1, \ldots, d \tag{11}$$

where $g_{j,w}$ and $f_{j,w}$ are respectively the first derivative of the call option price with respect to the time and second derivative of the call option price with respect to the strike price. The above derivatives are estimated through a standard finite difference scheme.

Notice that even if the two new variables S_j and t_w irange over the same set $\tilde{K} \otimes \mathrm{T}$ they assume a completely different meaning. Indeed they represent, respectively, the level of underlying S (no more the strike price $\tilde{k}_j$) and the time (no more than the time to maturity T_w).

This transformation allows us to shift from a measure of global volatility (implied volatility) to a measure of local volatility. This enables us to distinguish each node of the grid according to its local volatility value and to describe the evolution of the underlying asset.

3.4 MODEL DISCRETIZATION (MD)

This step uses a trinomial tree for the price of the underlying asset. It is aimed at computing the prices of a general contingent claim, whose underlying process can be described by (5), starting from the local volatility $\sigma_{j,w}$. The main feature of such a computational model is that it allows one to associate the nodes of the tree different local volatilities and drifts. The elementary structure of the trinomial tree is depicted in Figure V:

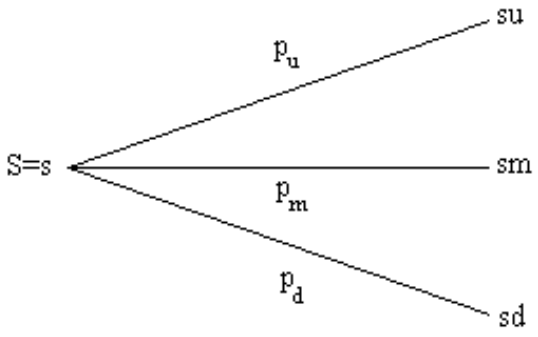

FIGURE V. Representation of the trinomial tree (one period).

The stability, calibration and convergence conditions (see Paras [1995]) allow for the definition of the parameters of the trinomial tree in such a way as to guarantee that the discrete model of the process effectively assigns to each distinct level of *S* and *t* the input volatility value. The parameters of computation, for the specific market we are dealing with, are expressed through the following formulas:

$$p_u = \frac{p}{2}\left(1 - \frac{\sigma_{\max}\sqrt{\Delta t}}{2}\right) \qquad p_d = \frac{p}{2}\left(1 + \frac{\sigma_{\max}\sqrt{\Delta t}}{2}\right) \qquad p_m = 1 - p \tag{12}$$

and

$$u = e^{\sigma_{\max}\sqrt{\Delta t}} \qquad m = 1 \qquad d = e^{-\sigma_{\max}\sqrt{\Delta t}} \tag{13}$$

where Δt represents a (small) interval of time between successive shocks, measured in years, and $\sigma_{\max} \equiv \max_{j,w}\{\sigma_{j,w}\}$ is the maximum value of local volatility which has to be modeled.

In order to better clarify the contents of the MD step let us describe in detail the basic sub-steps which allow us to obtain the option price starting from the local volatility. To this extent we first characterize the contract in terms of:

- time to maturity *T*, (measured in years);
- underlying price *s*;
- number of periods *N*;
- strike price *K*.

Then, we can compute the quantity $\Delta t = T/N$ and build a new $(2M+1)\times(N+1)$ grid, where $M = \lfloor 6\sqrt{N} \rfloor$, which describes the dynamic of prices from the current time *t=0* and the maturity *t=T* according to the formulas (12) and (13). The grid is function of two indexes: $\eta = 1,..., N+1$ and $\lambda = 1,...,2M+1$. The first index determines the time t_η on the grid, while the second index identifies the prices s_λ of the underlying **security(?).** Each node of the grid is distinguished by a pair (s_λ, t_η), where the values of the variables are defined as:

$$s_1 = e^{-M\sigma_{\max}\sqrt{\Delta t}} < \cdots < s_\lambda = e^{-(M-\lambda+1)\sigma_{\max}\sqrt{\Delta t}} < \cdots < s_{2M+1} = e^{M\sigma_{\max}\sqrt{\Delta t}} \tag{14}$$

and

$$t_1 = 0 < \cdots < t_\eta = (\eta-1)\Delta t < \cdots < t_{N+1} = T\,. \tag{15}$$

To each node of the grid must be assigned the corresponding local volatility computed previously. Given that nodes of the two grids are not necessarily coincident, a linear interpolation between local volatility values $\sigma_{j,w} = \sigma(s_j, t_w)$ (j =1,...,d; w=1,...,d) computed on the $d \times d$ grid is needed for the mapping on the values $\hat{\sigma}_{\lambda,\eta} = \sigma(S_\lambda, t_\eta)$ $(\lambda = 1,\ldots,2M+1; \eta = 1,\ldots,N+1)$ to assign them to the $(2M+1)\times(N+1)$ grid for pricing purposes. We have already introduced the extrapolation problem for the local volatility surface in the price direction. In fact, the levels of the underlying price s_λ are not *a priori* controllable because they are defined using the formula (14) and so they can be outside the computational domain. This problem is different with respect to the time extrapolation problem we had on the implied volatility surface, which we solved with a direct extrapolation on the forecasting function defined by the neural network.
Extrapolation of the local volatility in the price direction is required for nodes that that lie outside from the domain $\tilde{K}$. In fact, the numerical example shows that initial prices are contained in the interval [1.362152,1.649869], whereas the trinomial lattice price levels range from $e^{-\lfloor 6\sqrt{N} \rfloor \sigma_{\max}\sqrt{T/N}}$ to $e^{\lfloor 6\sqrt{N} \rfloor \sigma_{\max}\sqrt{T/N}}$. Therefore, they can assume very small values (0.5) as well "big" values (3-4) with respect the initial domain. We decided to use a simple solution, which consists in assuming constant volatility levels for prices outside the domain $\tilde{K}$. We do this to avoid the introduction of "subjective" biases.
Pricing contingent claims on the trinomial tree is a straightforward application of the backward induction rule. The value of the option at a generic time t_η for a specific underlying value s_λ is computed by discounting expected value, with respect to the probability measure p_u, p_m and p_d, of the payoff of the option at time $t_{\eta+1}$ and for the underlying prices $s_{\lambda+1}$, s_λ and $s_{\lambda-1}$.
The probabilities p_u, p_m and p_d, are functions of the local volatility of the node $\hat{\sigma}_{\lambda,\eta} = \sigma(s_\lambda, t_\eta)$ as seen in the following formulas:

$$p_u = \frac{\hat{\sigma}^2\lambda,\eta}{2\sigma^2\max}\left(1 - \frac{\sigma\max\sqrt{\Delta t}}{2}\right) \qquad p_d = \frac{\hat{\sigma}^2\lambda,\eta}{2\sigma^2\max}\left(1 + \frac{\sigma\max\sqrt{\Delta t}}{2}\right) \qquad p_m = 1 - \frac{\hat{\sigma}^2\lambda,\eta}{\sigma^2\max} \tag{16}$$

Starting from η = N to η = 1 we determine the contingent claim price $V_1^{M+1} = V(s_{M+1}, t_1, K)$ according to the following recursive formula:

$$V_{\eta}^{\lambda} = e^{-r_{DEM}\Delta t}\left[V_{\eta+1}^{\lambda+1}p_u + V_{\eta+1}^{\lambda}p_m + V_{\eta+1}^{\lambda-1}\right] = e^{-r_{DEM}\Delta t}\left[V_{\eta+1}^{\lambda+1}\frac{\hat{\sigma}_{\lambda,\eta}}{2\sigma_{\max}}\left(1-\frac{\sigma_{\max}\sqrt{\Delta t}}{2}\right)+V_{\eta+1}^{\lambda}\left(1-\frac{\hat{\sigma}_{\lambda,\eta}}{\sigma_{\max}}\right)+V_{\eta+1}^{\lambda-1}\frac{\hat{\sigma}_{\lambda,\eta}}{2}\left(1+\frac{\sigma_{\max}\sqrt{\Delta t}}{2}\right)\right]$$

where $V_{N+1}^{\lambda} = V(s_{\lambda}, t_{N+1}, K)$ for $\lambda = 1, \ldots, 2M+1$ represents payoff to maturity for the considered contingent claim for strike K and underlying price s_{λ}.

For a call option:

$$V_{N+1}^{\lambda} = (s_{\lambda} - K)^{+} \; \forall \lambda = 1,\ldots,2M+1$$

and, in a similar way, for a put option:

$$V_{N+1}^{\lambda} = (K - s_{\lambda})^{+} \; \forall \lambda = 1,\ldots,2M+1$$

For nodes that are on the border of the grid, where the tree is binomial, the value is computed as:

$$V_{\eta}^{1} = 2V_{\eta}^{2} - V_{\eta}^{3} \; \forall \eta = 1,\ldots,N$$

$$V_{\eta}^{2M+1} = 2V_{\eta}^{2M} - V_{\eta}^{2M-1} \; \forall \eta = 1,\ldots,N$$

In the recursive scheme the discounting factor $e^{-r_{DEM}\Delta t}$ appears given that with the last adjustment we are in the real market.

4. THE USD/DEM OPTION PRICING

In this section we illustrate how the pricing scheme, described through Section 3, can be used when the USD/DEM over-the-counter option, for the date of August 23, 1995, is considered. The case under study has been performed by means of the software environment named BLINNBOP. This software package consists of the following three main modules:

1. Bayesian Learning In Neural Networks (**BLINN**); this module, written in C language, implements the Markov Chain Monte Carlo (MCMC) solution procedure for the Bayesian learning problem on FNNs allowing one to perform the IVM step.
2. Bayesian Option Pricer (**BOP**); this module, which has been written in MATLAB, receives the input from the BLINN module and implements the computational steps (OPC, LVC and MD) to obtain the final option prices.
3. Pricing Analyzer (**PA**); this module, which has been written in MATLAB, allows one to compare the original option prices with the option prices obtained through the option pricing scheme.

The three BLINNBOP software modules should be used in a strictly sequential fashion. The BLINN module allows one to approach the IVM computational task and to perform several statistical tests for checking the FNN model significance. Once the FNN model is considered statistically significant it is automatically linked to the BOP software module. The BOP module exploits the FNN model developed through the use of the BLINN module and allows one to perform three of the five steps of the option pricing scheme, namely the OPC, the LVC and the MD computational steps. At this stage, the BLINNBOP Graphical User Interface (GUI) allows for the pricing of general contingent claims. For each contract to be priced, the user is must specify the option type, underlying spot price, strike price, number of periods and time to maturity. Finally, the PA module, which offers its own GUI, compares the computed option prices with the input prices from the market.

4.1 THE INPUT DATASET

The input data set (Table I) that we will consider for the volatility-modeling task is borrowed from Avellaneda and Paras [1996]. It corresponds to US Dollar-Deutschemark (USD/DEM) over-the-counter options for the date of August 23, 1995. The data set consists of 25 option prices corresponding respectively to 20, 25, and 50-delta puts and calls, with expiration dates of 30, 60, 90, 180 and 270 days.

Maturity	Type	Strike	Bid	Offer	Mid	IVOL
	Call	1.5421	0.0064	0.0076	0.0070	14.9
	Call	1.5310	0.0086	0.0100	0.0093	14.8
30 days	Call	1.4872	0.0230	0.0238	0.0234	14.0
	Put	1.4479	0.0085	0.0098	0.0092	14.2
	Put	1.4371	0.0063	0.0074	0.0069	14.4
	Call	1.5621	0.0086	0.0102	0.0094	14.4
	Call	1.5469	0.0116	0.0135	0.0126	14.5
60 days	Call	1.4866	0.0313	0.0325	0.0319	13.8
	Put	1.4312	0.0118	0.0137	0.0128	14.0
	Put	1.4178	0.0087	0.0113	0.0100	14.2
	Call	1.5764	0.0101	0.0122	0.0112	14.1
	Call	1.5580	0.0137	0.0160	0.0149	14.1
90 days	Call	1.4856	0.0370	0.0385	0.0378	13.5
	Put	1.4197	0.0141	0.0164	0.0153	13.6
	Put	1.4038	0.0104	0.0124	0.0114	13.6
	Call	1.6025	0.0129	0.0152	0.0141	13.1
	Call	1.5779	0.0175	0.0207	0.0191	13.1
180 days	Call	1.4823	0.0494	0.0515	0.0505	13.1
	Put	1.3902	0.0200	0.0232	0.0216	13.7
	Put	1.3682	0.0147	0.0176	0.0162	13.7
	Call	1.6297	0.0156	0.0190	0.0173	13.3
	Call	1.5988	0.0211	0.0250	0.0226	13.2
270 days	Call	1.4793	0.0586	0.0609	0.0598	13.0
	Put	1.3710	0.0234	0.0273	0.0254	13.2
	Put	1.3455	0.0173	0.0206	0.0190	13.2

TABLE I. USD/DEM OTC Options – August 1995

In the last column, the implied volatility (IVOL) associated with each option is reported. Notice that the data listed in Table I depends on the market parameters (Table II).

Description	Symbol	Value
Spot Price	S	1.4885 DEM
DEM Interest rate	r_{DEM}	4.27 %
US Interest rate	r_{USD}	5.91 %

TABLE II. Market parameters

According to what is presented in subsection 3.1 we have to make a transformation to obtain the implied volatility data related to a zero interest rate market. This can be done by means of the following transformation:

$$\tilde{k}_i = k_i e^{(r_{USD} - r_{DEM})T_i} \tag{17}$$

The adjusted dataset, obtained through transformation (17), is reported in Table III.

Maturity	Type	Adj. Strike	Bid	Offer	Mid	IVOL
	Call	1.544209	0.0064	0.0076	0.0070	14.9
	Call	1.533094	0.0086	0.0100	0.0093	14.8
30 days	Call	1.489234	0.0230	0.0238	0.0234	14.0
	Put	1.449880	0.0085	0.0098	0.0092	14.2
	Put	1.439065	0.0063	0.0074	0.0069	14.4
	Call	1.566376	0.0086	0.0102	0.0094	14.4

	Call	1.551134	0.0116	0.0135	0.0126	14.5
60 days	Call	1.490669	0.0313	0.0325	0.0319	13.8
	Put	1.435117	0.0118	0.0137	0.0128	14.0
	Put	1.421681	0.0087	0.0113	0.0100	14.2
	Call	1.582877	0.0101	0.0122	0.0112	14.1
	Call	1.564401	0.0137	0.0160	0.0149	14.1
90 days	Call	1.491703	0.0370	0.0385	0.0378	13.5
	Put	1.425533	0.0141	0.0164	0.0153	13.6
	Put	1.409567	0.0104	0.0124	0.0114	13.6
	Call	1.615695	0.0129	0.0152	0.0141	13.1
	Call	1.590892	0.0175	0.0207	0.0191	13.1
180 days	Call	1.494505	0.0494	0.0515	0.0505	13.1
	Put	1.401647	0.0200	0.0232	0.0216	13.7
	Put	1.379465	0.0147	0.0176	0.0162	13.7
	Call	1.649869	0.0156	0.0190	0.0173	13.3
	Call	1.618587	0.0211	0.0250	0.0226	13.2
270 days	Call	1.497608	0.0586	0.0609	0.0598	13.0
	Put	1.387967	0.0234	0.0273	0.0254	13.2
	Put	1.362152	0.0173	0.0206	0.0190	13.2

TABLE III. USD/DEM OTC Options – August 1995 (zero rate)

4.2 IMPLIED VOLATILITY MODELING (IVM)

In order to approach the IVM modeling task we performed several numerical experiments, that nature of which we skip for brevity. The IVM modeling results are presented and discussed below.

In Table IV a comparison between the network's forecasts and the original dataset is reported. Furthermore in Figure VI the residual analysis. **(Commento sui residui).**

Maturity	Type	Adj. Strike	Market IVOL	Computed IVOL	Diff	Diff %
	Call	1.544209	14.9			
	Call	1.533094	14.8			
30 days	Call	1.489234	14.0			
	Put	1.449880	14.2			
	Put	1.439065	14.4			
	Call	1.566376	14.4			
	Call	1.551134	14.5			
60 days	Call	1.490669	13.8			
	Put	1.435117	14.0			
	Put	1.421681	14.2			
	Call	1.582877	14.1			
	Call	1.564401	14.1			
90 days	Call	1.491703	13.5			
	Put	1.425533	13.6			
	Put	1.409567	13.6			
	Call	1.615695	13.1			
	Call	1.590892	13.1			
180 days	Call	1.494505	13.1			
	Put	1.401647	13.7			
	Put	1.379465	13.7			
	Call	1.649869	13.3			
	Call	1.618587	13.2			
270 days	Call	1.497608	13.0			
	Put	1.387967	13.2			

	Put	1.362152	13.2			

TABLE IV USD/DEM OTC Options – August 1995 (zero rate)

In Figure VII the implied volatility surface, computed via the FNN, is shown.

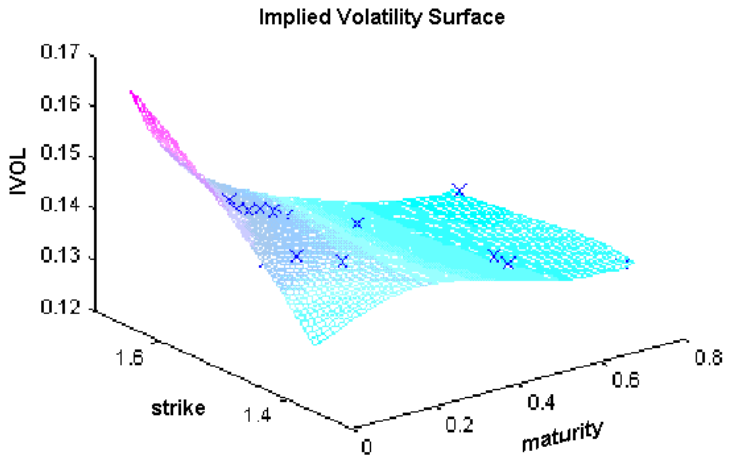

FIGURE VII. FNN implied volatility surface for the USD/DEM option market (mesh) and corresponding input data set (crosses).

It is interesting to analyze the sections of the implied volatility surface in order to highlight the systematic behaviors of the market with respect to term structure (changes in volatility due to maturity fluctuations) and strike structure (changes in volatility due to strike price fluctuations). The behavior of the implied volatility with respect time to maturity, for different sections of the surface in Figure VII, is depicted in the Figure VIII.

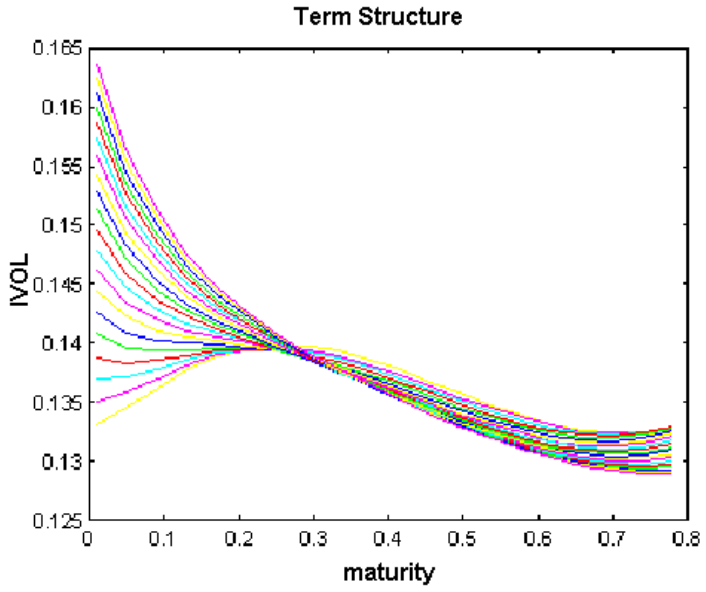

FIGURE VIII. Term structure for FNN approximated implied volatility surface for the USD/DEM option market. The term structure highlights a particular shape. Long term implied volatility is, in general, lower then the short term one. Moreover, short term implied volatility shows a higher unsteadiness then the long term one due to big changes in volatility values with respect to different strike prices.

Another interesting cross section, of the implied volatility surface reported in Figure VII, is the one reported in Figure IX. Such a cross section allows us to graphically investigate the behavior of implied volatility with respect to the strike price.

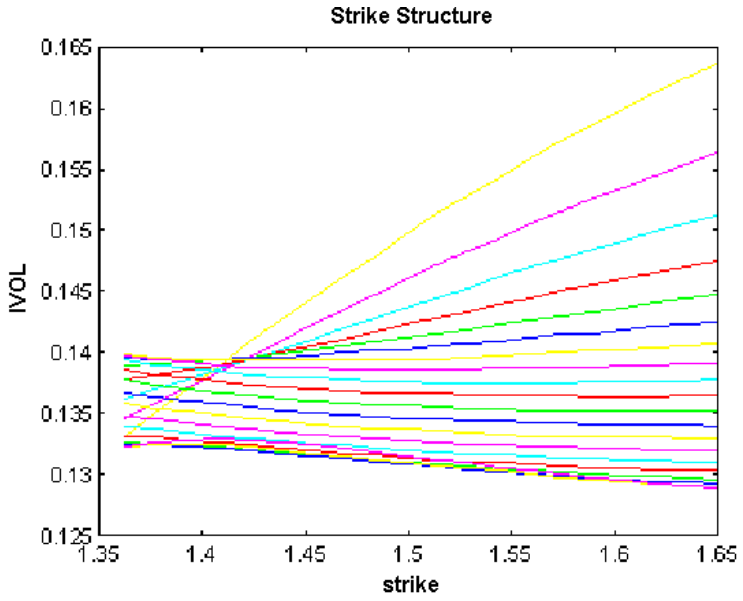

FIGURE IX. Strike structure for FNN implied volatility surface corresponding to the USD/DEM option market. The strike structure is characterized by a light smile effect, indeed the implied volatility decreases when the strike price increases.

4.3 OPTION PRICES COMPUTATION (OPC)

According to this reasoning we begin the description of the case study directly from the OPC computational step. In particular the output of this computational step is the so called "Prices Surface" which is reported in Figure X.

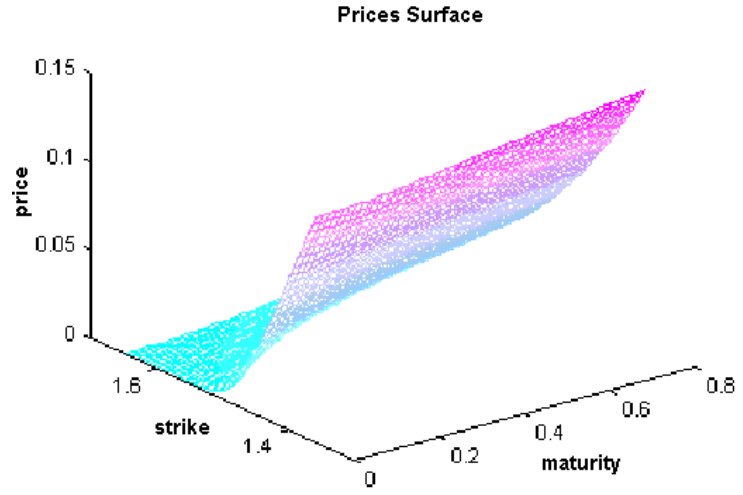

FIGURE X. Prices Surface – It is obtained by means of the BOP software module jointly with the FNN input provided by means of the BLINN module.

Si dovrebbe inserire qualche commento sulla figura.

The spot price *S* used in the computation is fixed at 1.4885 and was observed on the market synchronously with the other data.

4.4 LOCAL VOLATILITY COMPUTATION (LVC)

The second computational step (LVC) performed by means of the BOP module allows us to compute the local volatility surface. In particular the BOP module allows us to collect additional information about this computational step. Indeed the BOP module allows us to compute two derivatives which bring important information about the numerical properties of the current solution. In Figure XI the behavior of the first derivative of the call option price $C(S,K,T)$ with respect to time to maturity T is depicted.

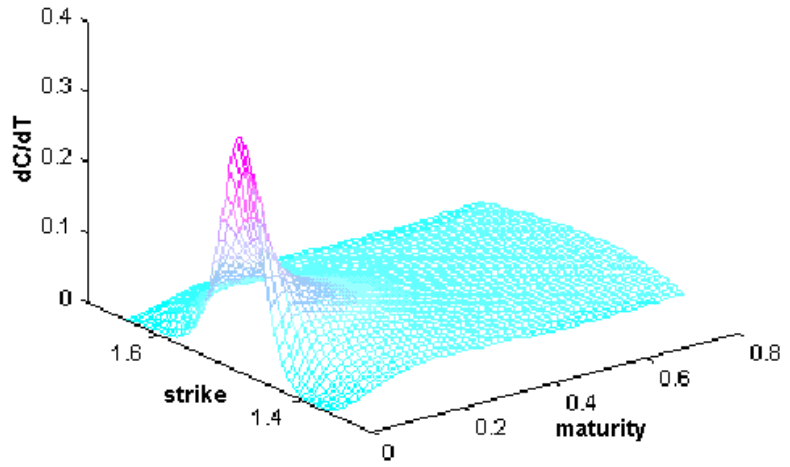

FIGURE XI. Call Option Price First Derivative - First derivative of the call option price $C(S,K,T)$ with respect to the time to maturity T.

The behavior of the second order derivative of the call option price $C(S,K,T)$ with respect to the strike price K is reported in Figure XII.

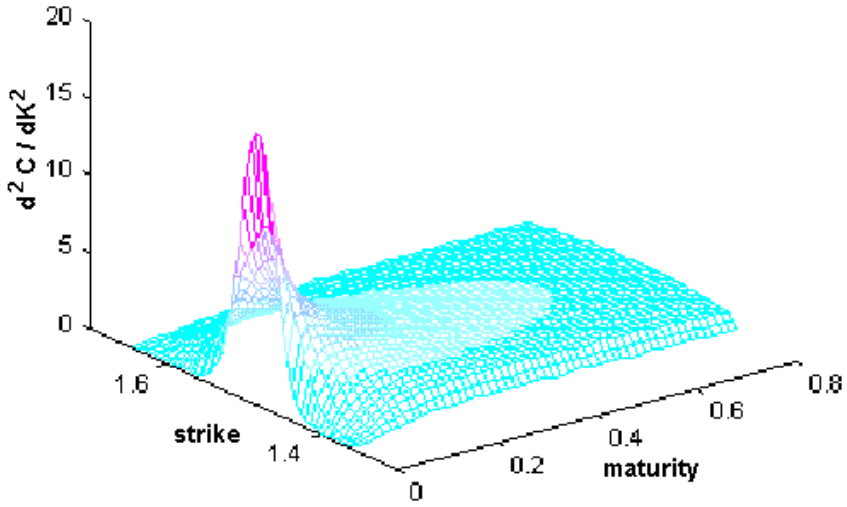

FIGURE XII. Call Option Price Second Derivative - Second derivative of the call option price $C(S,K,T)$ with respect to the time to maturity K.

From the above two figures (Figure XI and Figure XII) it is possible to conclude that the computational model utilized for the LVC step shares nice numerical properties. Indeed the two figures show two smoothed surfaces indicating that the derivatives estimated through the finite difference scheme are a reliable approximation of the true derivatives.

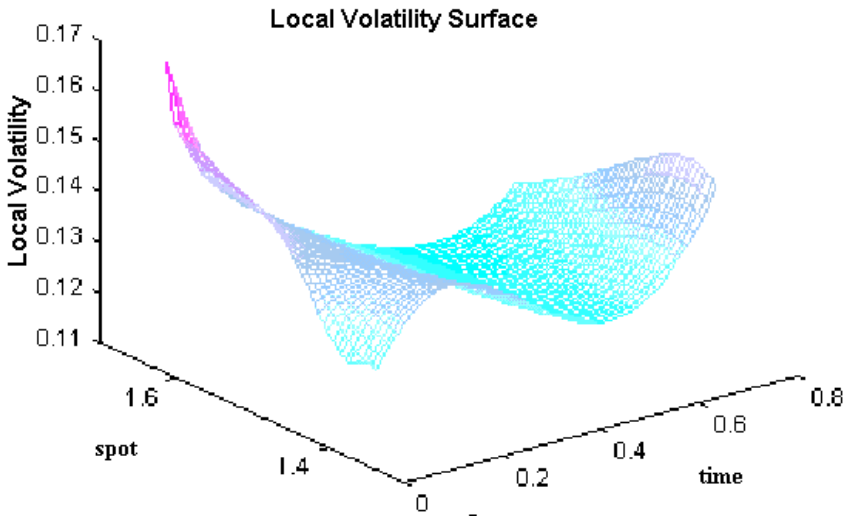

FIGURE XIII. Local Volatility Surface – This figure illustrates the approximated local volatility surface obtained through the BOP software module.

A further graphical analysis about the nature of the Local Volatility Surface can be performed by exploiting the features of the BOP software module. Indeed this module offers the possibility to plot several sections of the Local Volatility Surface, with respect to the input variables, which generates valuable information about the numerical properties of the utilized computational model.

Accordingly to this observation it is interesting to analyze the behavior of the term structure (Figure XIV) and the behavior of the spot structure (Figure XV), where the strike prices are replaced by the underlying spot prices, of the local volatility surface.

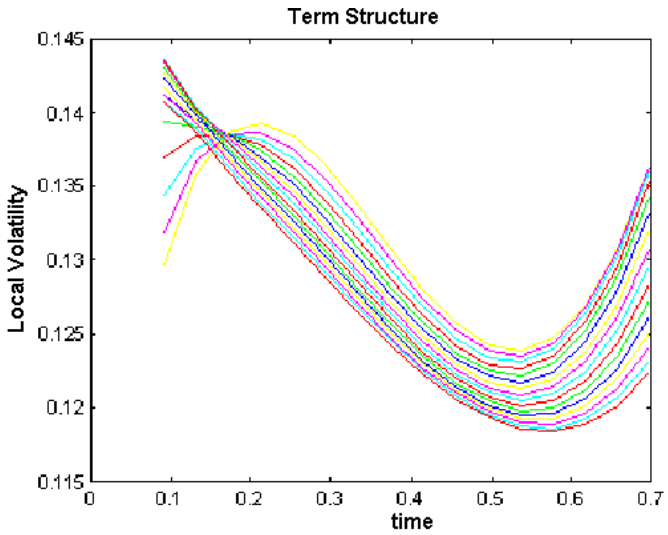

FIGURE XIV. Term Structure – Cross section of the term structure for the implied volatility model depicted in Figure XI.

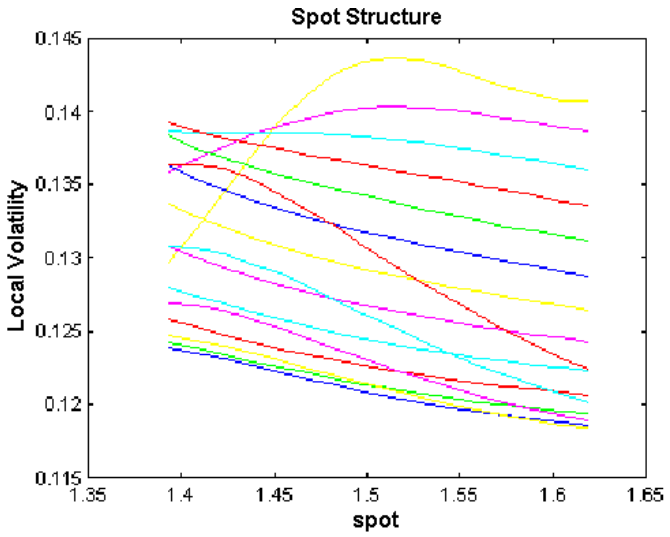

FIGURE XV. Spot Structure – Cross section of the spot structure for the implied volatility model depicted in Figure VII.

4.5 MODEL DISCRETIZATION (MD)

The third, and final step (MD), performed through the BOP module allows us to obtain the approximated Local Volatility Surface as shown in Figure XVI.

Using this extrapolation method, the local volatility surface mapped on the new grid shows the following behavior:

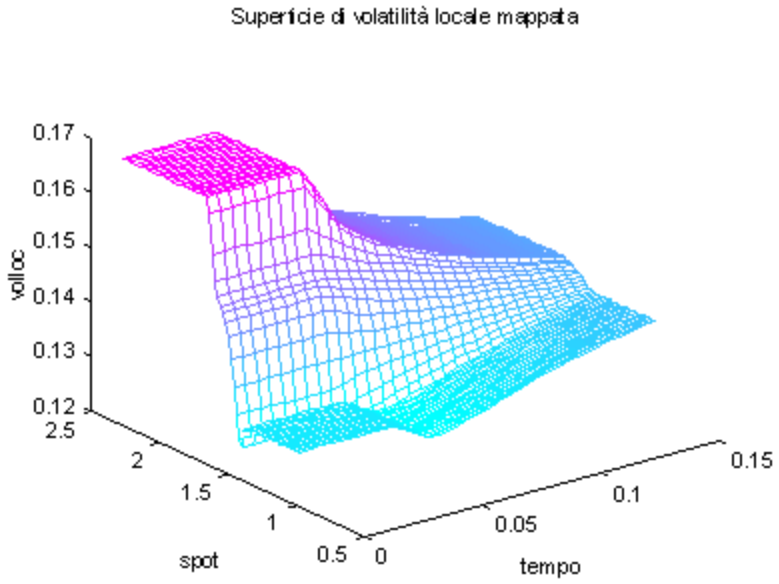

FIGURE XVI. Local volatility surface mapped on Trinomial Stochastic Volatility Model. Extrapolation is required for values which can be deeply outside the domain $\tilde{K}$. In fact, while initial prices are contained in the interval [1.362152,1.649869], the prices computed by the trinomial tree range from $e^{-\lfloor 6\sqrt{N} \rfloor \sigma_{\max}\sqrt{T/N}}$ to $e^{\lfloor 6\sqrt{N} \rfloor \sigma_{\max}\sqrt{T/N}}$. Hence, they can assume very small values (0.5) as well as "big" values (3-4) with respect to the initial domain. In such a situation any extrapolation, without adding "subjective" information on market behavior, is not satisfactory. We decided to use a simple solution which consists of assuming constant volatility for prices outside the domain $\tilde{K}$ for which the local volatility is undefined.

Maturity	Type	Adj. Strike	Market Price	Computed Price	Diff	Diff %
	Call	1.544209	0.0070	0.0069720	2.80E-05	0.4
	Call	1.533094	0.0093	0.0091977	1.02E-04	1.1
30 days	Call	1.489234	0.0234	0.0235170	-1.17E-04	-0.5
	Put	1.449880	0.0092	0.0091448	5.52E-05	0.6
	Put	1.439065	0.0069	0.0068034	9.66E-05	1.4
	Call	1.566376	0.0094	0.0094094	-9.40E-06	-0.1
	Call	1.551134	0.0126	0.0125370	6.30E-05	0.5
60 days	Call	1.490669	0.0319	0.0321233	-2.23E-04	-0.7
	Put	1.435117	0.0128	0.0127360	6.40E-05	0.5
	Put	1.421681	0.0100	0.0099400	6.00E-05	0.6
	Call	1.582877	0.0112	0.0111440	5.60E-05	0.5
	Call	1.564401	0.0149	0.0147063	1.94E-04	1.3
90 days	Call	1.491703	0.0378	0.0382158	-4.16E-04	-1.1
	Put	1.425533	0.0153	0.0152694	3.06E-05	0.2
	Put	1.409567	0.0114	0.0115482	-1.48E-04	-1.3
	Call	1.615695	0.0141	0.0140436	5.64E-05	0.4
	Call	1.590892	0.0191	0.0193674	-2.67E-04	-1.4
180 days	Call	1.494505	0.0505	0.0507525	-2.53E-04	-0.5
	Put	1.401647	0.0216	0.0214272	1.73E-04	0.8
	Put	1.379465	0.0162	0.0160380	1.62E-04	1
	Call	1.649869	0.0173	0.0174038	-1.04E-04	-0.6
	Call	1.618587	0.0226	0.0225322	6.78E-05	0.3
270 days	Call	1.497608	0.0598	0.0606372	-8.37E-04	-1.4
	Put	1.387967	0.0254	0.0257302	-3.30E-04	-1.3
	Put	1.362152	0.0190	0.0192280	-2.28E-04	-1.2

TABLE V USD/DEM OTC Options – August 1995 (zero rate)

Comparison results are summarized in the following graphics:

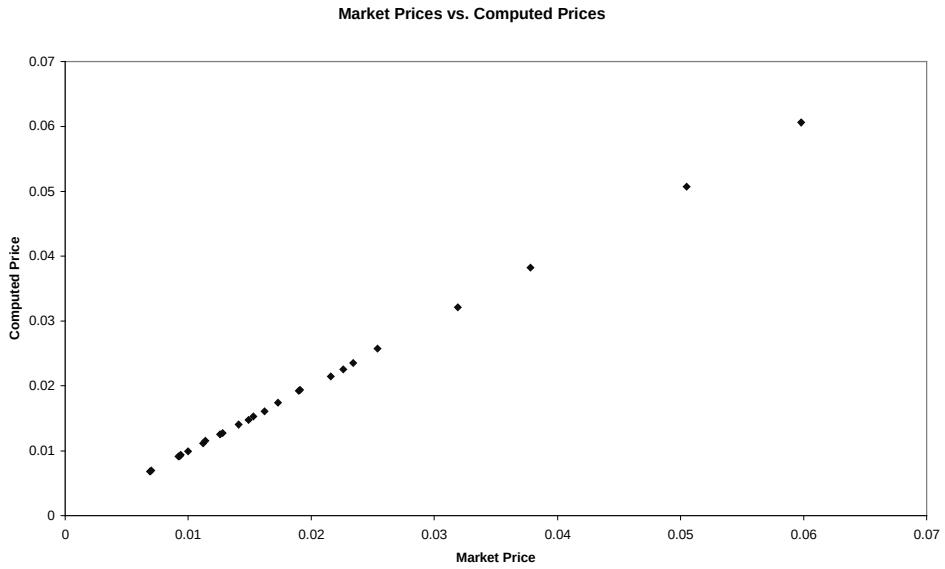

FIGURE XVII. Market Prices vs. Computed Prices – Scatter Plot

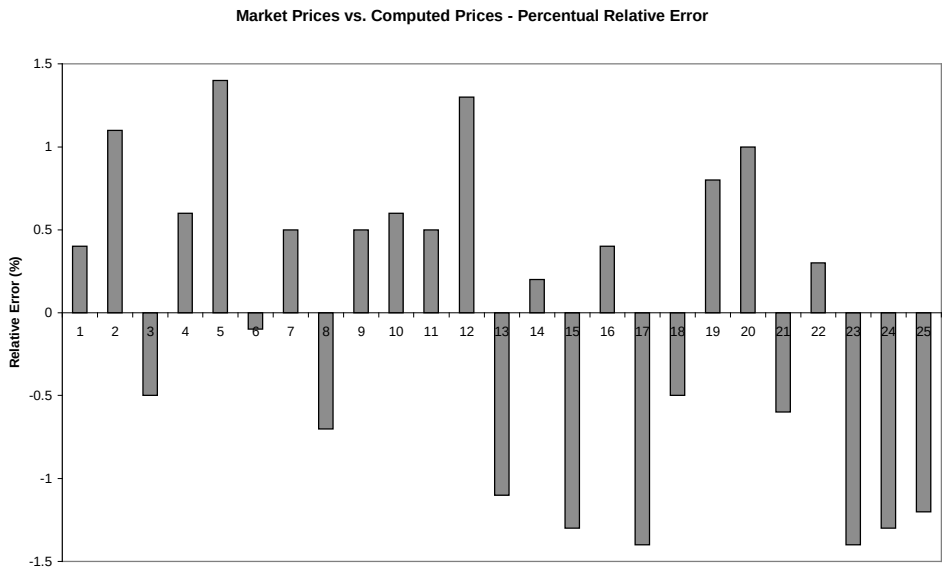

FIGURE XVIII. Market Prices vs. Computed Prices – Percentage Relative Error

It is easy to verify how the pricing system is able to correctly compute option prices. Indeed, computed values are very close to the market ones and they are always contained in the bid-ask interval, reflecting the uncertainty of the market. We have to remember that the scope of the pricing system is to globally describe the "sentiment" of the market about future volatilities (regression), instead of to exactly recompute market prices (interpolation). Therefore, the proposed pricing system allows us to compute option prices on the entire domain with the same precision we obtained on the initial market prices. The precision, as we observe in Figure XVIII, is characterized by the relative error in range (-1.4%, +1.4%).

In this paper we have shown that it is possible to pass from a set of liquid option prices to a pricing system by means of which we can value other derivatives whose prices are not readily available from the market, i.e. illiquid European options, American options and Exotic options. We are also certain that the system is valuing all the instruments consistently with the market. For example, the proposed pricing system can be used for pricing Barrier options, where the probability of striking the barrier is sensitive to the shape of the smile, for creating static hedge portfolios for Exotic options or for generating Monte Carlo distributions for valuing path-dependent options.

REFERENCES

Avellaneda, M., Friedman, C., Holmes, R., and Samperi, D. [1997], Calibration of volatility surfaces via relative-entropy minimization, *Applied Mathematical Finnace*, Vol. 4

Avellaneda, M. and Paras, A. [1996], Managing the volatility risk of portfolios of derivative securities: the Lagrangian Uncertain Volatility Model, *Applied Mathematical Finance*, Vol. 3, pp. 21-52

Derman and Kani [1998], Stochastic implied trees: arbitrage pricing with stochastic term and strike structure of volatility, *IJTAF*, Vol. 1, No. 1, pp. 61-110

Dupire, B. [1994], Pricing with a smile, *RISK* 7(1), pp. 18-20

Eales, J. and Hauser R. [1990], Analyzing biases in valuation models of options on futures, *Journal of Future Markets*, Vol.10**,** No.3, pp. 415-447.

Fouque, J., Papanicolaou, G., and Sircar, R. [1998] , Mean-Reverting Stochastic Volatility, *International Journal of Theoretical and Applied Finance,* forthcoming

Heston, S. [1993], A closed-form solution for options with Stochastic Volatility with applications to bond and currency options, *Review of Financial Studies*, Vol. 6, No. 2, pp. 327-343

Hull, J. and White A.. [1987], Nonparametric tests of alternative option pricing models, *Journal of Finance*, Vol.17**,** No.2, pp. 415-447.?????????

McKay, D.J.C. [1992], Bayesian interpolation, *Neural Computation*, Vol.4**,** No.3, pp. 415-447.

Neal, R.M. [1996], Bayesian learning for Neural Networks, *Lecture Notes in Statistics*, Springer-Verlag.

Rubinstein, M. [1994], Implied binomial trees, *Journal of Finance*, Vol. LXIX, No. 3, pp. 771-818

Rubinstein, M. [1985], Nonparametric tests of alternative option pricing models, *Journal of Finance*, Vol.40**,** No.2, pp. 455-480

Scott, L. [1987], Option Pricing when the Variance changes randomly: Theory, Estimation, and an Application, *Journal of Financial and Quantitative Analysis*, Vol. 22, No. 4, pp. 419-438

White, H. [1989], Learning in Artificial Neural Networks: A statistical perspective, *Neural Computation*, Vol.1**,** No.4, pp. 425-464

Wiggins, J. [1987], Option Values under Stochastic Volatility, *Journal of Financial Economics*, Vol. 19, No. 2, pp. 351-372