Local Volatility Calibration using the "Most Likely Path"

19 December 2006

Computational Methods in Finance

Professor Ali Hirsa/ Paris Pender

Submitted by:

Kwasi Danquah

Saurav Kasera

Brian Lee

Sonky Ung

TABLE OF CONTENTS

INTRODUCTION

Why are we interested in calibrating a local volatility surface?
It is mostly to price dependent exotic options which are not very liquid and the proper market quotes are not available. Local volatility also gives rise to some interesting relative value trading strategies.

There are a number of local volatility calibration methods available, but no one agrees on the "perfect" calibration method. There is no single correct answer. Perhaps that is the reason why calibration is not only a science but also an art. The problem of calibration requires creativity and that is the reason why there is no perfect calibration model. There is always room for improvement in these kinds of problems.

Below we discuss a robust local volatility calibration method that appeared in a 2-page *Risk Magazine* article in April 2006[1]. The method is based a fixed-point iteration, and on the concept of the "most likely path" first observed by Jim Gatheral [2]. He calls it the "most probable path".

We start with a market implied volatility surface and then by an iterative process construct our local volatility surface. We have taken special care in making sure that our market option prices have been interpolated using liquid options as it can be very tempting to neglect the quality of data in these kinds of projects. Filtering data is again a mixture or art and sciences. The process then entails forming a most likely path and then taking the time average of the volatility along that path. We then price options using this surface and compare it with the Monte Carlo prices generated by our Monte Carlo pricer.

OUR METHOD

The "most likely path" approach works by an iterative scheme that starts out by making a guess of what the local volatility surface should look like. We take the market implied volatility surface as our initial guess and put it through a series of iterations till in converges. The process of going from some market implied volatility to our final local volatility surface is defined as the phi-inverse operator: $\Phi^{-1} : \Sigma \to \sigma$

Phi – inverse: Going from Market Implied Vol to Calibrated Local Vol

The Grid

Given an implied volatility data for a finite set of (K,T), we interpolate onto a "calibration grid" using Matlab "*grid fit*" interpolator. Once a surface interpolated, we can obtain a value for the implied volatility for any of the grid points. The figure below shows the red circles as the market implied volatility data points and the calibration grid as the black dots.

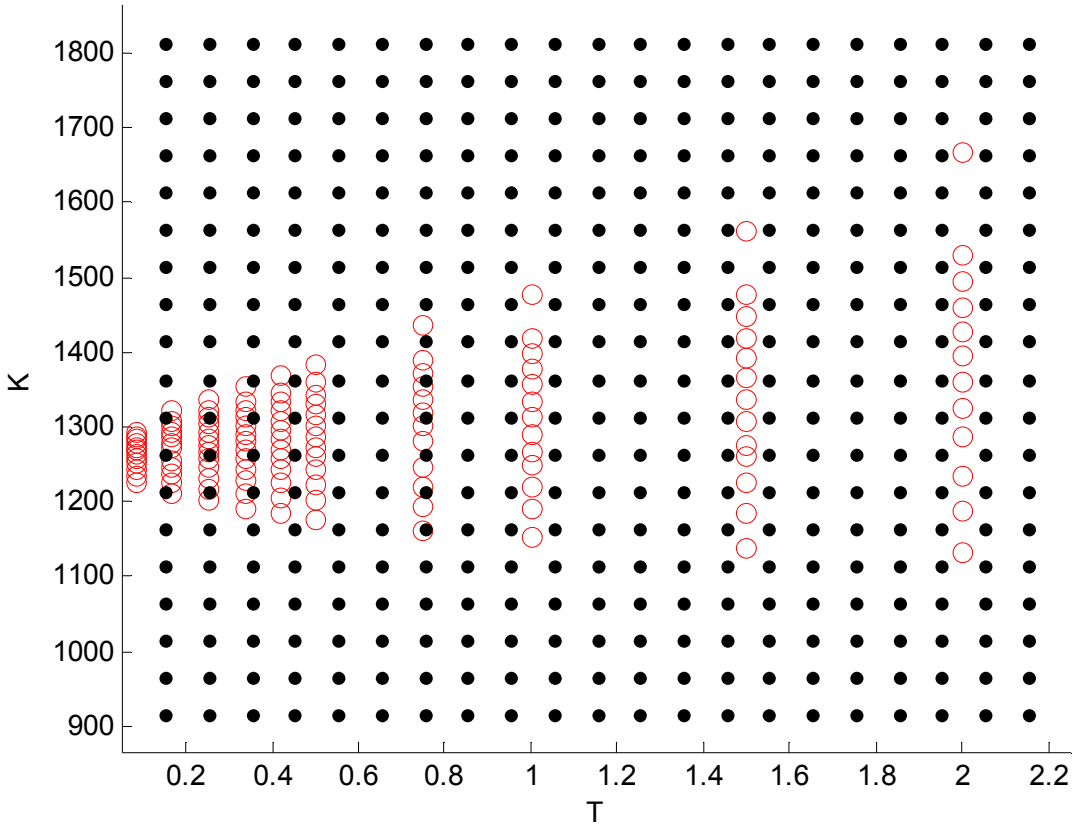

Figure 1

The Process

To go from the i^{th} estimate of the local volatility surface to the $(i+1)^{th}$ estimate, we first generate an implied volatility surface from the current local volatility surface. This is done point-by-point (done for each point on the local volatility grid) using the definition of the implied volatility proxy. This concept is discussed in the next section and we define an operator phi that performs the entire operation of going from some local volatility surface to an implied volatility surface. Once we generate the implied volatility surface from the local volatility, we use the definition below to generate the next estimate of the local volatility surface.

$$\{\sigma^{i+1}\} = \{\sigma^{i}\} \cdot \frac{\{\Sigma_{market}\}}{\{\Phi(\sigma^{i})\}}$$

This operation is performed point wise; meaning, we evaluate it for every single point on the local volatility surface. So, for some S and t on the current local volatility surface, the next local volatility estimate for that point is the previous value times a factor. The factor is the ratio of the corresponding implied volatility on the market implied volatility surface divided by the implied volatility from the surface generated by our operator phi.

This process repeats till we the nth local volatility surface converges. We check for convergence after each step by comparing the L1 norm of the new local volatility surface and the previous local volatility surface. If the difference is less than some threshold, we stop. In our case we use ***0.01*** as the convergence limit.

Phi Going from Local Volatility to Implied Volatility Surface

$\Phi : \sigma \to \Sigma$

Given a local volatility surface (i.e. evaluation by linearly interpolating on a grid), this function generates an implied volatility surface (i.e. points on a grid). For each point (K, T) on Σ calibration grid we do the following:

1. Generate the most likely path iteratively, assume the following price dynamics:
   $$\frac{dS_t}{S_t} = r \cdot dt + \sigma(t) \cdot dW_t$$

2. Evaluate $\Sigma(K,T)$ using the implied volatility proxy

TWO KEY CONCEPTS

The following two key concepts comprise the core of our calibration method. The power of our calibration method directly depends on the accuracy of the *implied volatility proxy* based on the idea of the *most likely path* which we discuss below.

Most Likely Path

The concept of the most likely path is very intuitive and has been discussed by many authors. The idea is to figure out the path of the spot if you know its terminal point $S_T = K$. We need to calculate the conditional expectation $E[S_t \mid S_T = K]$. This is a difficult calculation if we use the original dynamic model:

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

Therefore it has to be tackled starting with a simper version.

$$\frac{dS_t}{S_t} = \mu_t dt + \underbrace{\sigma(E(S_t \mid S_T = K), t)}_{\sigma_*(t)} dW_t$$

$$\frac{dS_t}{S_t} = \mu_t dt + \sigma_*(t) dW_t$$

In this simpler version, the stochastic volatility part does not depend on S_t any more. It now only depends on the time t. And this dynamic generates a log-normal process. It is straightforward to calculate the most likely path knowing the end-point of this dynamic $S_T = K$. Using the simplified model we can end up with the following closed form formula:

$$\mathrm{E}[S_t \mid S_T = K] = F_t F_T^{-\alpha_{t,T}} K^{\alpha_{t,T}} e^{\frac{1}{2}\alpha_{t,T}\int_t^T \sigma^2(s)ds}$$

Where,

$$\alpha_{t,T} = \frac{\int_0^t \sigma^2(s)ds}{\int_0^T \sigma^2(s)ds}$$

is a regression coefficient between 0 and 1 and

$$F_t = S_0 \exp \int_0^t \mu_s ds$$

is the natural forward

Then through the following iterative algorithm, the most likely path for the original dynamic $m_t(K,T) = E[S_{t|} \mid S_T = K]$ can be obtained. First, by looking at the forward $\sigma_t^0 = \sigma(F_{t,}t)$ we initialize particular term structure volatility and compute for this initial term structure local volatility the most likely path as described in the initial log-normal process. If we note the following:

$$E_{K,T}^i(\bullet) = E\left[\bullet \mid S_T = K, \frac{dS_t}{S_t} = \mu_t dt + \sigma_t^i dW_t\right]$$

Then,

$$m_t^0(K,T) = E_{K,T}^0(S_t)\sigma_t^1 = E_{K,T}^0(\sigma(t,S_t))$$

Now we update our estimated most likely path by the formula: $m_t^1(K,T) = E_{K,T}^0(S_t)$
More generally, the algorithm repeats the following two steps:

- $\sigma_t^{i+1} = E_{K,T}^i(\sigma(t,S_t))$
- $m_t^{i+1}(K,T) = E_{K,T}^{i+1}(S_t)$

This algorithm converges very fast and gives the most likely path given local volatility. Figure.2 shows that our engine is very robust and it converges to most likely path in a few iterations. The green line is the initial forward stock price path $S_t = S_0 e^{rt}$ and the black line is the most likely path with the parameters S_0 = 1265.65, r = 5.25%, K=1400, T=1.0.

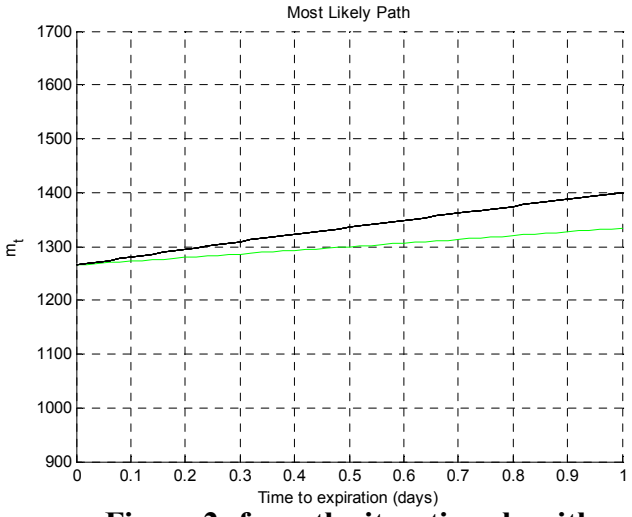

Figure 2: from the iterative algorithm

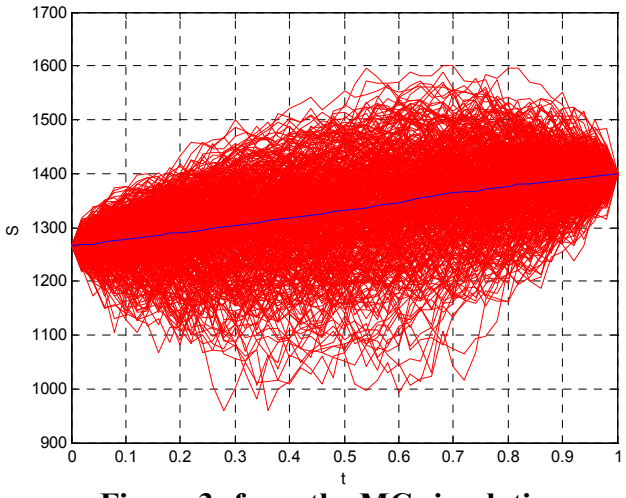

Figure 3: from the MC simulation

The average of the generated red sample paths whose $S_T = K$ gives very similar results for the most likely path using our algorithm. See figure 4. And figure 5 is the zoomed version of figure 4 changing the vertical axis value, stock price.

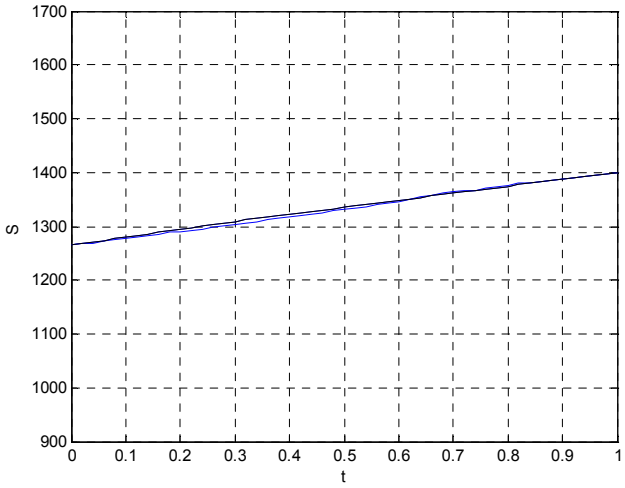

Figure 4

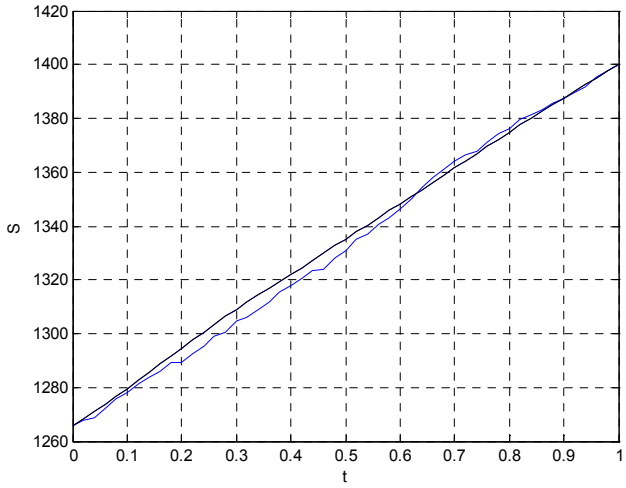

Figure 5

Implied volatility Proxy

The *most likely path* concept enables us to derive an accurate approximation formula for the implied volatility. Indeed, the implied volatility of a call option maturing at T with a strike K can be well approximated by the following integral:

$$\sigma_{BS}{}^2(K,T) = \Sigma^2(K,T) \approx \frac{1}{T}\int_0^T \sigma^2(m_t,t)\cdot dt$$

It says that the Black-Scholes implied variance of an option with strike K is given approximately by the integral from valuation date (t =0) to the expiration date (t =T) of the local variances along the *most likely path.* Of course in practice, it's not easy to compute the m_t. Nevertheless, this method gives us a very intuitive picture for the meaning of Black-Scholes implied variance of a European option with a given strike and expiration: This approximation states that implied volatility is the time average of local volatility along the *most likely path* conditional on the stock price at expiration being the strike price of the option.

These two concepts are based on rigorous theoretical probability theory. For detailed derivations we refer the reader to [2] and Implied volatility: Statics, Dynamics, and Probabilistic Interpretation, 2002 [3].

OPTION DATA EXTRACTION

We use **"OptionMetrics"** from the WRDS (Wharton Data Research Services) database as our data source. OptionMetrics is a comprehensive source of historical price and implied volatility data for the US equity and index options markets. They have a special facility called "Volatility Surface" data.

Volatility Surface contains the interpolated volatility surface for each security on each day, using a methodology based on kernel smoothing algorithm. This file contains information on standard options, both calls and puts, with expirations of 30, 60, 91, 152, 182, 273, 365, 547, 730 calendar days, at deltas of 0.2, 0.3, 0.4, 0.5, 0.6, 0.7,and 0.8 (negative deltas for puts). A standard option is only included if there exists enough option price data on that date to accurately interpolate the required values.

The standardized option implied volatilities in the Volatility Surface are calculated using a kernel smoothing technique [4]. The data is first organized by the log of days to expiration and by "call-equivalent delta" (delta for a call, one plus delta for a put). A kernel smoother is then used to generate a smoothed volatility value at each of the specified interpolation grid points. At each grid point *j* on the volatility surface, the smoothed volatility $\boldsymbol{\sigma_i}$ is calculated as a weighted sum of option implied volatilities:

$$\hat{\sigma}_j = \frac{\sum_i V_i \sigma_i \Phi\left(x_{ij}, y_{ij}, z_{ij}\right)}{\sum_i V_i \Phi\left(x_{ij}, y_{ij}, z_{ij}\right)}$$

where *i* is indexed over all the options for that day, V_i is the vega of the option, $\boldsymbol{\sigma_i}$ is the implied volatility, and Φ(.) is the kernel function:

$$\Phi\left(x, y, z\right) = \frac{1}{\sqrt{2\pi}} e^{-\left[\left(x^2/2h_1\right)+\left(y^2/2h_2\right)+\left(z^2/2h_3\right)\right]}$$

The parameters to the kernel function, x_{ij}, y_{ij}, and z_{ij} are measures of the "distance" between the option and the target grid point:

$$x_{ij} = \log\left(T_i / T_j\right)$$
$$y_{ij} = \Delta_i - \Delta_j$$
$$z_{ij} = I_{\{CP_i = CP_j\}}$$

where T_i (T_j) is the number of days to expiration of the option (grid point); Δ_i (Δ_j) is the "call-equivalent delta" of the option (grid point); CP_i (CP_j) is the call/put identifier of the option (grid point); and I{.} is an indicator function (=0 if the call/put identifiers are equal, or 1 if they are different).
The kernel "bandwidth" parameters were chosen empirically, and are set as h_1=0.05, h_2=0.005, and h_3=0.001.

We download the following fields from the database:

1. Days to Expiration.
2. Interpolated Implied Volatility
3. Implied Strike Price
4. Implied Premium.
5. Spot price.

The above data is downloaded for both calls and puts.

We have designed a data processing module in Matlab that pulls this data in Matlab vectors and then fed into out local volatility processing engine. The Matlab vectors contain implied volatility data only for OTM calls and puts. We filter away the in and at the money options in our data processing module.

RESULTS
Our method applied to SPX

We start out with a finite set of 126 points {strikes, maturities, implied vol} for SPX on 8-Feb-2006. Using this finite set of "126 points", we interpolate to a smooth implied vol surface shown below on Figure 6.

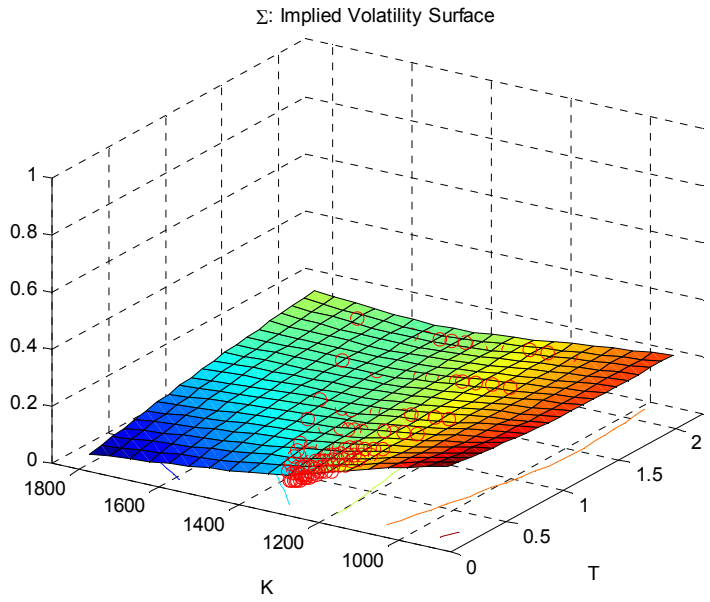

Figure 6

We then calibrate a local volatility surface based on this market data using Reghai's [1] fixed-point algorithm. Below is the resulting local volatility surface just after 5 iterations as shown on Figure 7.

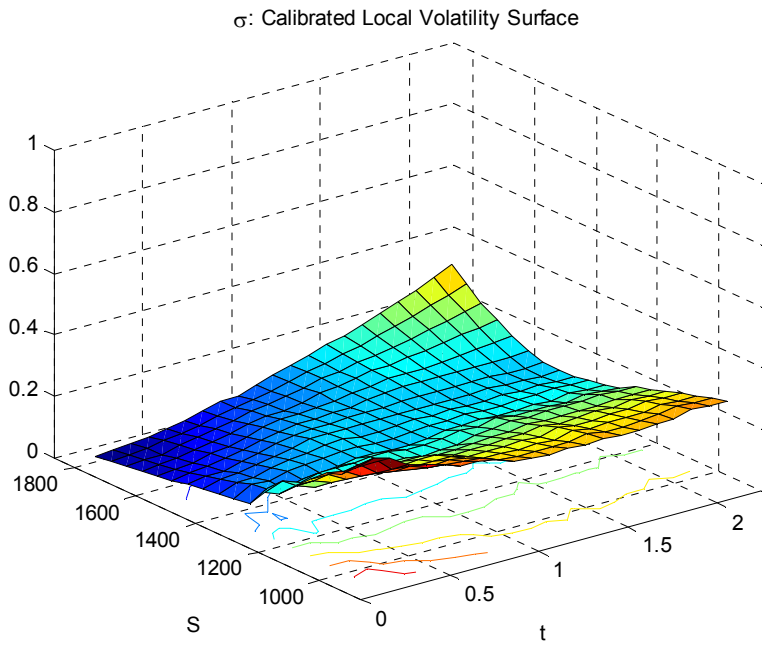

Figure 7

Total calibration time was 3 minutes (using Matlab)! Shown below on Figure 8 is the evolution of the local volatility surface in the iterations of the calibration process.

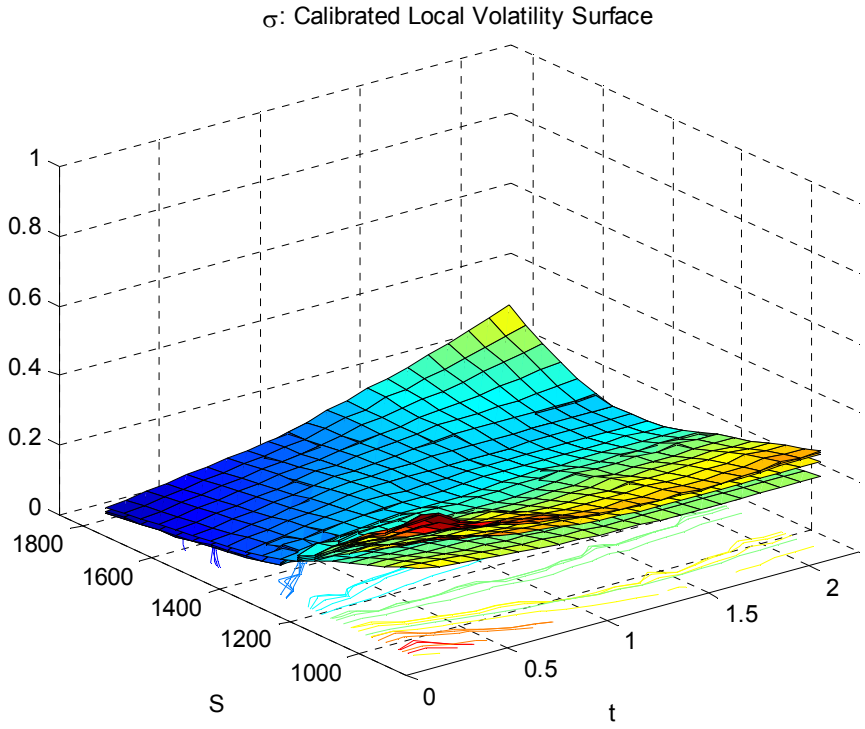

Figure 8

As a check, we use the Φ function to see whether we get back our original implied vol surface. Of course, we do! (That's why the algorithm converged). Below is the reconstructed implied vol surface, using the *newly* calibrated local vol surface.

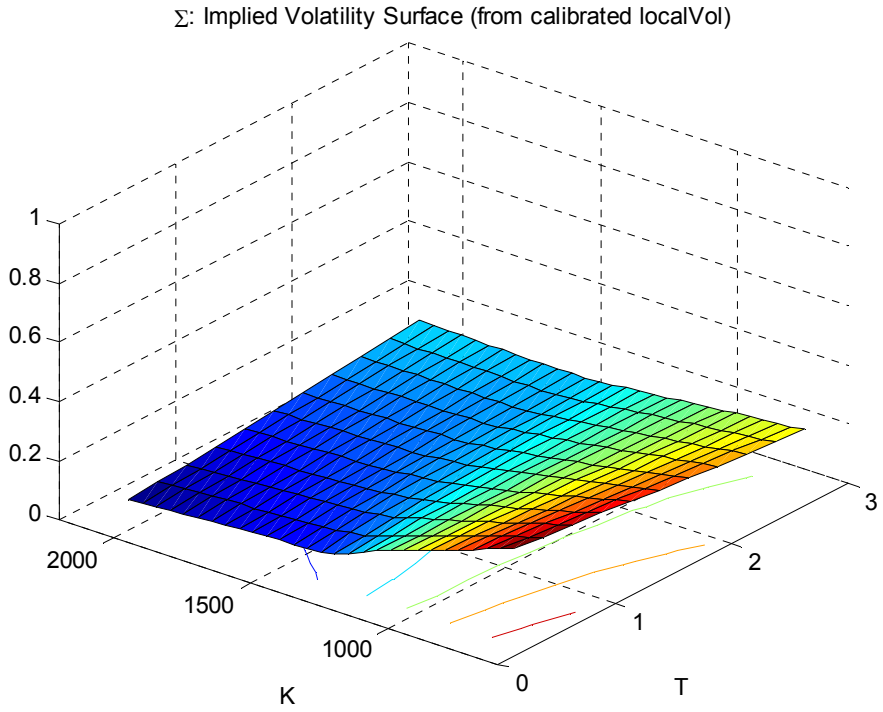

Figure 9

This gives us confidence in our calibration method. Now we validate our calibrated local volatility surface using a Monte-Carlo local pricer under local volatility price dynamics.

MONTE CARLO VALIDATION

To check the validity of our local volatility surface we compare the prices we generate by using the surface to prices we get when we plug the market implied volatility into the Black-Scholes formula. If the two results match up nicely then we can confirm that our local volatility surface has been calibrated accurately.

To get the prices via the local volatility surface we use a Monte Carlo simulation. Assuming that price follows the following local volatility dynamics:

$$\frac{dS_t}{S_t} = r \cdot dt + \sigma(S_t, t) \cdot dW_t$$

We discretize the process as follows,

$$S_{i+1} = S_i + (r * dt * S_i) + sigma\ (S_i,t) * S_i * dZ$$

We read $sigma\ (S_i,t)$ from our calibrated local volatility surface and generate dZ as iid standard normal at each step.
For variance reduction we apply the antithetic variates method. This means generating an alternate path that uses a negatively correlated random normals, dW = -dZ to generate changes in price. We then take the average payoff of the two paths as the final payoff.

Using the parameters below, we come up with a
Number of simulations = 2500;
dt = 0.002;
S_0 = 1265.65; % SPX closing price on 8-Feb-2006

The tables below compare Black-Scholes analytical prices, with Monte-Carlo simulation prices.

For this example, we calibrated on a grid with dT=0.1, dK=50, and evaluated by interpolation on a finer grid of dT=0.02, dK=25.

Table of Call Prices

K \ T	*0.1*	*0.50*	*0.75*	*1.0*	*1.50*	*2.00*
800	$469.84 $469.91	**$486.38** **$486.58**	**$496.54** **$497.32**	**$506.57** **$507.75**	**$526.23** **$528.74**	$545.39 $549.07
1000	$270.90 $270.95	$294.34 $293.47	$309.03 $308.39	$324.03 $323.53	$352.99 $352.14	$381.56 $380.43
1100	$171.60 $171.42	$201.27 $200.16	$218.77 $218.06	$236.63 $235.54	$270.02 $269.09	$301.90 $299.97
1150	$122.46 $121.99	$156.83 $155.60	$175.86 $175.23	$195.14 $193.99	$230.61 $229.75	$263.78 $263.33
1200	$75.02 $73.94	$114.99 $113.94	$135.38 $134.85	$155.81 $154.22	$192.98 $191.42	$227.00 $225.04
1250	$33.38 $31.91	$77.28 $76.52	$98.33 $97.98	$119.36 $118.78	$157.45 $155.67	$191.78 $190.46
1300	$7.07 $6.44	$45.86 $45.75	$66.17 $66.44	$86.82 $86.98	$124.50 $125.54	$158.53 $158.71
1350	$0.29 $0.25	$23.01 $22.30	$40.67 $41.91	$59.54 $59.85	$95.00 $95.98	$128.02 $129.87
1400	$0.00 $0.00	$9.28 $9.08	$22.88 $23.49	$38.52 $38.63	$69.97 $73.01	$100.96 $103.07
1600	$0.00 $0.00	$0.00 $0.00	$0.50 $0.38	$3.34 $3.29	$17.58 $17.21	$36.29 $38.37

Table of Put Prices

	0.1	*0.5*	*0.75*	*1.0*	*1.5*	*2.0*
800	***$0.00*** ***$0.00***	***$0.00*** ***$0.20***	***$0.00*** ***$0.54***	***$0.00*** ***$1.07***	***$0.00*** ***$2.32***	***$0.00*** ***$3.82***
1000	**$0.01** **$0.00**	$2.78 $2.38	$4.77 $4.31	$7.24 $6.19	$11.61 $10.60	$16.24 $14.11
1100	**$0.19** **$0.05**	$7.12 $6.15	$10.64 $10.26	$14.72 $13.74	$21.07 $19.90	$26.61 $23.81
1150	**$0.79** **$0.32**	$11.38 $9.66	$15.81 $15.40	$20.67 $19.66	$27.87 $27.54	$33.50 $32.25
1200	**$3.09** **$1.85**	$18.25 $17.58	$23.40 $23.03	$28.79 $27.28	$36.46 $34.79	$41.74 $40.09
1250	**$11.19** **$9.81**	$29.24 $28.39	$34.41 $33.80	$39.78 $39.25	$47.14 $45.45	$51.53 $51.02
1300	$34.61 $33.87	$46.53 $46.68	$50.33 $50.91	$54.68 $54.99	$60.40 $61.63	$63.31 $63.71
1350	$77.57 $77.55	$72.38 $71.19	$72.90 $74.29	$74.84 $75.25	$77.12 $78.57	$77.81 $80.28
1400	$127.02 $127.14	$107.36 $107.06	$103.18 $104.15	$101.27 $100.99	$98.30 $101.86	$95.76 $97.90
1600	$325.97 $325.90	$292.90 $292.89	$272.95 $272.54	$255.81 $255.36	$230.40 $230.43	$211.16 $213.58

The areas highlighted in red indicate that the BS prices were outside the 95% confidence interval of the Monte-Carlo prices.

CONCLUSION

Unlike other finite difference calibration methods that are directly dependent on the smoothness of the interpolated price surface, our "most likely path" calibration method does not depend on this smoothness. This gives this calibration method robustness. The method is also time efficient as it took less than 3 minutes to calibrate in Matlab. Furthermore, it converges in a few iterations. We have confidence that our calibrated local volatility surface, as Black-Scholes and Monte-Carlo prices differ within $1 across the entire (K, T) surface on average. Further work can be done by experimenting how calibrating on a finer grid affects the calibrated local volatility surface.

REFERENCES

1. Reghai, "*The Hybrid Most Likely Path*". Risk Magazine. April 2006.
2. J. Gatheral, The Volatility Surface: A Practitioners Guide. Wiley 2006.
3. R. Lee, "*Implied Volatility: Statics, Dynamics, and Probabilistic Interpretation*". Recent Advances in Applied Probability, 2005.
4. http://wrds.wharton.upenn.edu/ds/optionm/vsurfd/doc.shtml