Sequential Calibration of Asset Pricing Models to Option Prices

Joel Oagile

A dissertation submitted to the Faculty of Commerce, University of Cape Town, in partial fulfilment of the requirements for the degree of Master of Philosophy.

June 15, 2018

MPhil in Mathematical Finance,
University of Cape Town.

Declaration

I declare that this dissertation is my own, unaided work. It is being submitted for the Degree of Master of Philosophy to the University of Cape Town. It has not been submitted before for any degree or examination to any other university.

Signed by candidate

Joel Oagile
June 15, 2018

Abstract

This paper implements four calibration methods on stochastic volatility models. We estimate the latent state and parameters of the models using three non-linear filtering methods, namely the extended Kalman filter (EKF), iterated extended Kalman filter (IEKF) and the unscented Kalman filter (UKF). A simulation study is performed and the non-linear filtering methods are compared to the standard least square method (LSQ). The results show that both methods are capable of tracking the hidden state and time varying parameters with varying success. The non-linear filtering methods are faster and generally perform better on validation. To test the stability of the parameters, we carry out a delta hedging study. This exercise is not only of interest to academics, but also to traders who have to hedge their positions. Our results do not show any significant benefits resulting from performing delta hedging using parameter estimates obtained from non-linear filtering methods as compared to least square parameter estimates.

Acknowledgements

My sincere gratitude goes to my supervisor Associate Professor Peter Ouwehand. This dissertation would not have reached its present form without his patience and guidance. I would also like to thank my family who supported me throughout my postgraduate studies. Lastly, I thank my friend Amogelang Rapelang for dedicating sleepless nights proof-reading.

Contents

List of Figures

List of Tables

List of Abbreviations

Black-Scholes: BS
Stochastic differential equation: SDE
Geometric Brownian motion: GBM
Kalman filter: KF
Extended Kalman filter: EKF
Iterated extended Kalman filter: IEKF
Least Square: LSQ
Weighted Least Square: WLS
Penalized Weighted Least Square: PWLS
Unscented Kalman filter: UKF
Normal inverse Gaussian-Cox-Ingersoll Ross: NIG-CIR
Barndorff Nielsen Shephard: BNS
Cosine Expansion pricing: COS method
Parameter: Par
Validation: Val
Estimation: Est
In sample fit: IS
Mean absolute error: MAE
Mean square error: MSE
Relative mean square error: RMSE
Root mean square error: RMSE

Chapter 1

Introduction

Since its inception, the Black and Scholes (1973) model has received strong criticism regarding its restrictive assumptions and its inability to capture features of observed option prices (He *et al.*, 2005). As a result, variants of option pricing models were developed that try to relax some assumptions of the model. Such improvements include introducing a jump term in modelling the underlying (stock) price process, treating volatility as a stochastic quantity, adding jumps to the stochastic differential equation (SDE) of volatility.

Treating volatility as a stochastic quantity is a sensible assumption particularly after the stock market crash of 1987 and lately 2008 (Gatheral and Lynch, 2002). Lindström *et al.* (2008) support this idea and argue that market properties change over time because of change in macro-economic events, investor preferences etc. Thus, realistic option pricing models are required for risk management and hedging under the real world probability measure $\mathbb{P}$ and for pricing purposes under the so called risk-neutral probability measure $\mathbb{Q}$. Common models for option pricing include stochastic volatility, stochastic interest rates, local volatility, jump diffusion and Lévy models. Kovachev (2014) calibrates some of these models and Cont and Tankov (2004) explore these models in pricing and risk management purposes. Bakshi *et al.* (1999) state that every model has to make three basic assumptions regarding the underlying price process, the market factor risks and interest rate process. These assumptions make the models to choose from virtually infinite.

Once a model is chosen based on assumptions made, its parameters must be estimated by calibrating the model using observed option prices or fitting to historical data. Considering the overwhelming number of models already developed for pricing and risk management, it is not easy to choose among the competing models. An analysis can be made on the performance of different models and calibration techniques based on metric measures like the following.

1. Global fit measures:

(a) Root mean square error (RMSE): RMSE is the square root of mean square error which measures the squared difference between model option prices and observed option prices. We expect to have a lower RMSE if our parameter estimates are very close to the true values. In other words, the model option prices will be very close to observed option prices

(b) Mean absolute error (MAE): This statistic also measures the variation between model prices and market prices. Willmott and Matsuura (2005) compare RMSE and MAE and conclude that all model validations should use MAE as it is unambiguous. The authors go further and suggest that all past model validation that used RMSE are questionable.

(c) In-sample fit: Counts the number of model prices that fall in the bid ask spread. The model prices are calculated using estimated parameters (the parameters estimated using data reserved for parameter estimation).

(d) Out-sample fit: This measure is similar to the above statistic except that we use data from the validation set. In other words, we calculate model prices for options that were not used in parameter estimation and count prices falling in the bid ask spread.

2. Pricing and hedging options

3. Run time: We present the average time taken to do one calibration/iteration for that particular day.

One approach to calibration of asset pricing models to option prices is the least square method. Its advantages are that it is simple to understand and calibrate but it has its drawbacks too which include over-fitting among others. Researchers have introduced more sophisticated techniques to address the shortcomings of the least square method. Among such calibration techniques are non-linear filtering methods that form the focus of this dissertation. This paper compares these two methods of calibration based on the metrics above.

1.1 Literature review

Before parameters can be estimated, a model has to be selected. Black and Scholes (1973) laid down the foundation for option pricing models and ever since, several papers were published trying to address the shortcomings of the Black-Scholes (BS) model. Today there are many models that try to capture the characteristics of the stock price process as observed in the market. Schoutens *et al.* (2003) argue that models used can give different results in pricing of exotic options even though

they all agree almost perfectly on the prices of vanilla options. This finding is not isolated as Jessen and Poulsen (2013) also show that different models give different prices for barrier options.

Calibration can be performed using techniques like least squares, method of moments and maximum likehood estimation. Cont and Tankov (2002) analyse the performance of non-linear least square estimation on calibrating jump diffusion models. They argue that the standard least square method has some challenges dealing with the inverse problem of finding the best parameters given option prices. In the paper, Cont and Tankov (2002) argue that the LSQ gradient descent optimisation routine can fail to identify the minimum as the objective function (quadratic pricing error) is non-convex. As a solution, the authors suggest a non-parametric method to address some weaknesses of the LSQ. The new technique involves adding a penalisation term to the LSQ objective function. This will ensure uniqueness and stability of parameter estimates.

Gagliardini *et al.* (2011) explore the extended method of methods (XMM) in parameter estimation. The method is an extension of the general method of moments (GMM) but unlike the GMM, the XMM can in addition to uniform moment restrictions, deal with local moment restrictions. This is of particular interest when spot returns and cross sectional option prices are used in calibration. The authors argue that the spot returns and cross sectional option data have uniform and local restrictions respectively.

Gilli and Schumann (2010) argue that calibration is often taken for granted. Gilli and Schumann (2010) stress the point that many researchers hardly discuss the impact of changing starting values on parameter estimates of the optimisation but this can lead to totally different results. The authors argue that gradient based optimisation often fail for the kinds of optimisation problem discussed in their paper and redoing the calibration with different starting values leads to very different parameter estimates.

Unlike the calibration methods discussed above, the sequential calibration method of Lindström *et al.* (2008) does not involve optimisation routines. Lindström *et al.* (2008) carry out a study to compare the performance of non-linear filtering and least square methods using S&P 500 option data. Lindström *et al.* (2008) calibrate Heston, Bates and NIG-CIR models using non-linear filtering (extended Kalman filter and iterated extended Kalman filter), and least square methods (weighted least square and penalized weighted least square). The four calibration techniques performed differently depending on the metrics chosen. The WLS performed better than the three calibration techniques in estimating the parameters of the Bates

model from simulated data. However, the WLS performed poorly on out of sample forecasts indicating over fitting. The filter methods (EKF and IEKF) performed very well in terms of parameter RMSE compared to the least square methods (WLS and PWLS).

The least square methods involve some form of optimisation routine and it is a well-known fact that most routines are only able to determine local extrema. Thus, WLS can sometimes fail to identify the global minimum as noted by Lindström *et al.* (2015). This led to WLS sometimes giving unreasonable parameter estimates in the Bates model for Lindström *et al.* (2008) empirical study. Overall, the least square methods performed poorly as compared to the non-linear filtering methods. This can be traced back to the fact that the least square methods only use cross sectional data as prior period observations are completely ignored. This methodology is questionable as captured by Lindström *et al.* (2015) in the following exclamatory.

"If yesterday's data are of little use today, then today's data will be of little use tomorrow!"[1]

It is found that non-linear filtering methods outperform the least square methods and Lindström *et al.* (2008) strongly recommend the former in calibrating asset returns models for option pricing as the techniques make optimal use of the data.

The aim of this paper is to calibrate stochastic volatility jump diffusion models to a time series of options prices using non-linear filtering methods and compare the results to the standard least square method. A hedging study is performed to see whether there are any benefits in hedging options using parameter estimates from non-linear filtering methods. The rest of the paper is structured as follows. In chapter two, we thoroughly discuss non-linear filtering and how they can be used to jointly estimate the state variables and model parameters. Chapter three gives an overview of the models to be calibrated together with their characteristic functions. Characteristic functions are at the heart of derivative pricing for these complicated models. Chapter four discusses the least square method and sequential calibration of Lindström *et al.* (2008). In chapter five, we carry out a simulation study and also perform discrete delta hedging. In the last chapter, we conclude.

[1] Lindström *et al.* (2015)

Chapter 2

Non-linear filtering

Filtering techniques are widely used across all disciplines like science, engineering and economics. In engineering, filtering can be used to track the motion of aircrafts and missiles. In economics, filtering methods are used in interest rate modelling, asset allocation and risk management among others. The techniques date back to the first publication by Kalman (1960). Suppose we have measured/observed some variables and we want to estimate the underlying state process that drives the observations. Using the observed variables, we can use filtering to estimate the conditional distribution of the state given the noisy observations that we have measured. To do this, the filtering techniques goes through a two-step process, namely the prediction step and the update step. Suppose we have a vector of the hidden state x_t and a vector of observed variables y_t recorded at time t. In the prediction stage, we use $\hat{x}_{t-1}$ to predict $\hat{x}_{t|t-1}$. In the update stage, $\hat{x}_{t|t-1}$ is adjusted using the observed value value y_t to get $\hat{x}_{t|t}$.

Assume that the state transition equation has the following form:

$$x_t = f(x_{t-1}, w_t) \tag{2.1}$$

In our formulation, the function f is non-linear and the state noise is given by w_t. It is clear that the vector of hidden state does not depend on the measurement but only depends on the previous hidden state and some noise.

Consider the following equation:

$$y_t = h(u_t, x_t, \nu_t) \tag{2.2}$$

The above equation is termed the measurement equation in the non-linear filtering literature. It is also clear that the measurements are driven by an unobserved process x_t and some noise ν_t. In this paper, u_t is a vector of known variables like strike price, maturity, risk free rate etc. To estimate the vector of hidden state x_t, we use Bayesian approach and continuously update our estimate according to the

arrival of new information. The conditional probability density functions is then given by the following Bayes formula:

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})} \tag{2.3}$$

To estimate the parameters of the models, we need to first choose our starting point. Lindström *et al.* (2008) suggest using the asymptotic estimates from the WLS method above as this will ensure that the filter is not worse off compared to the WLS. The filter estimates the hidden state process (volatility) together with the parameters of the model from observed options prices. This procedure is termed dual estimation of the parameters and the hidden state. We say that the estimation is done online. The algorithm for Kalman filter is summarised below.

- Given the initial starting point (x_0, P_0) and the first data points (option variables), the filter predicts the mean ($\hat{x}_{t|t-1}$) and covariance (P_t^-).
- The Kalman gain (K) is calculated using the estimate and measurement errors.
- The next step is to recalculate the updated mean ($\hat{x}_{t|t}$) and covariance (P_t). using the Kalman gain (K).
- This procedure simultaneously estimates the parameters of the model and the latent state (volatility).

Our transition function f is key to Kalman filtering. It is used to predict the parameters (estimates) of the model by using the prior estimates and transition noise. The parameters are given random walk dynamics. The measurement function h is basically a pricing function in this paper. In the GBM framework, it is the well known BS formula. The function takes estimated parameters and returns option prices. The function is crucial in calculating the Kalman gain. The Kalman gain (K) is a weighting factor. To better understand it, it is advisable to consider the classical Kalman filter. If the measurement error is very small compared to the error in the estimate, the Kalman gain will be closer to 1 and more weight is given to the measurement. On the other hand, if the measurement error is to high, the Kalman gain will be close to 0 and more weight will be given to the estimate (predicted state). The Kalman gain is a scalar ranging from 0 to 1 in the case of the classical Kalman filter. This idea can be extended to non-linear filtering although the Kalman gain would now be a matrix.

Below we go through the non-linear filtering methods to be covered in this paper. We also present the algorithm to be implemented using either observed or simulated data.

2.1 Extended Kalman filter

The classical Kalman filter is only applicable when our measurement and transition functions are both linear in the variables of interest. The extended Kalman filter was developed to address some shortfalls of the classical Kalman filter. Unlike in the case of Kalman filtering where the system equations are assumed to be linear, no such assumptions are made in EKF (Julier and Uhlmann, 1996). Since our measurement and state equations are non-linear, this method is appropriate. Lindström *et al.* (2008) posit that linearization is possible when the system is fairly linear and in some cases the filter may diverge for systems that are too non-linear.

Suppose we have a dynamical system given by the following two equations;

$$x_t = f(x_{t-1}) + w_t \tag{2.4}$$

$$y_t = h(u_t, x_t) + \nu_t \tag{2.5}$$

As already explained, the first equation is called the state transition equation and the second one is known as the measurement equation. In our modelling exercise, the noise for the two equations are additive which turn some matrices that we are going to deal with into identity matrices. The EKF starts by initialising the state transition density.

Initialisation stage: $p(x_0) \sim N(\hat{x}_0, P_0)$

The next step is to use the two equations given above to predict the distribution of the state.

Prediction stage:

$$\hat{x}_{t|t-1} = f(\hat{x}_{t-1}) + 0 \tag{2.6}$$

$$P_t^- = F(\hat{x}_{t-1}, 0)P_{t-1}F^T(\hat{x}_{t-1}, 0) + G(\hat{x}_{t-1}, 0)\mathbf{Q}(\hat{x}_{t-1}, 0)G^T(\hat{x}_{t-1}, 0) \tag{2.7}$$

Update stage: update the mean and the covariance using the Kalman gain K.

$$K = P_t^- H^T(\hat{x}_{t|t-1})(H(\hat{x}_{t|t-1})P_t^- H^T(\hat{x}_{t|t-1}) + \mathbf{R})^{-1} \tag{2.8}$$

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K(y_t - h((u_t, \hat{x}_{t|t-1})) \tag{2.9}$$

$$P_t = (I - KH(\hat{x}_{t|t-1}))P_t^- \tag{2.10}$$

Where

$F(x, w) = \frac{df}{dx_t}|_{(\hat{x}_{t-1},0)}$
$G(x, w) = \frac{df}{dw_t}|_{(\hat{x}_{t-1},0)}$
$H(x) = \frac{dh}{dx_t}|_{(\hat{x}_{t|t-1},0)}$
$\nu_t \sim \mathrm{N}(\mathbf{0}, \mathbf{R})$
$w_t \sim \mathrm{N}(\mathbf{0}, \mathbf{Q})$

The above procedure is summarised below in the form of algorithm.

Algorithm 1 Extended Kalman Filter

---

1. Initialization stage: $\hat{x_0} = \mathbb{E}[x_0]$ and $P_0 = \mathbb{E}[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T]$

2. for t=1 to N

predict stage

$\hat{x}_{t|t-1} = f(\hat{x}_{t-1}) + 0$

$P_t^- = F_t P_{t-1} F_t^T + G_t \mathbf{Q}_t G_t^T$

Update stage

$K = P_t^- H_t^T (H_t P_t^- H_t^T + \mathbf{R})^{-1}$

$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K(y_t - h((u_t, \hat{x}_{t|t-1}))$

$P_t = (I - KH_t)P_t^-$

3. end for

Tab. 2.1: EKF algorithm

As an example, we are going to use the geometric Brownian motion to simulate stock prices. Using the simulated stock prices, we can calculate European call options prices. Remember our SDE is given by

$$dS_t = rS_t d_t + \sigma S_t dW_t \tag{2.11}$$

Solving the above equation we get

$$S_{t+\Delta t} = S_t e^{(r-0.5\sigma^2)\Delta t + \sigma W_{\Delta t}} \tag{2.12}$$

Which can also be written as

$$S_{t+\Delta t} = S_t e^{(r-0.5\sigma^2)\Delta t + \sigma\sqrt{\Delta t}Z} \tag{2.13}$$

Where Z is standard normal random variable. Because of the additive property of logarithms, we covert the above equation to log form to get

$$\log S_{t+\Delta t} = \log S_t + (r - 0.5\sigma^2)\Delta t + \sigma\sqrt{\Delta t}Z \tag{2.14}$$

Now we can simulate log stock prices and calculate European call options under risk neutral measure using the following parameters:
S_0=100
r=0.1
$dt = \frac{1}{365}$
$T = \frac{1}{2}$
Strike=90,91,92,...,109
$t = 0, \frac{1}{365}, \frac{2}{365}, \ldots \frac{182}{365}$
σ=0.2

Using the log stock prices and the above parameters, we calculate a matrix of European call option prices using the Black-Scholes formula. All option prices are for half year maturity, calculated using the corresponding stock price. We **convert** from log stock price to stock price when convenient.

Let the matrix of option prices be y (20 (options per day) by 183(days)) and let our measurement function $h(u_t, x_t)$ be the Black-Scholes formula. In our measurement function, u_t is a vector of variables strike price, maturity (T) and risk free rate (r) whilst x_t is the state variable to be estimated. We are now going to **pretend** that we never knew the simulated stock prices and we are going to use the EKF algorithm to recover them.Our state transition equation f is now given by

$x_{t+\Delta t} = f(x_t) + w_t = \Big[\log S_t + (r - 0.5\sigma_t^2)\Delta t\Big] + \Big[\sigma_t^2 \Delta t\Big] = \Big[X_t + (r - 0.5\sigma^2)\Delta t\Big] + \Big[\sigma_t^2 \Delta t Z\Big]$

where Z is a standard normal random variable.

For this example, we chose our variance of the measurement noise to be $\mathbf{R} = 1/16$. We have to add this random noise (one fourth as standard deviation) to the matrix of simulated option prices.

Note also that our state noise variance is given by
$\mathbf{Q} = \sigma^2 \Delta t$

We also need to calculate the Jacobian matrices and we get

$F(x, w) = \frac{df}{dx_t}|_{(\hat{x}_{t-1},0)} = 1$
$G(x, w) = \frac{df}{dw_t}|_{(\hat{x}_{t-1},0)} = 1$
$H(x) = \frac{dh}{dx_t}|_{(\hat{x}_{t|t-1},0)}$

In this case, H is a 20 by 1 column vector where each entry is the sensitivity of the option to the changes in the log stock price. To get this vector we can use finite difference methods like central difference which has the following form:

$\frac{h(X+\epsilon,u_t)-h(X-\epsilon,u_t)}{2\epsilon}$

Where X is the log stock price, h is still the BS formula and other variables are held constant. For the BS formula, we have analytical formula which we can use but for other models to be discussed later, it is not easy to derive the analytical formula.

For demonstration, we are going to show the output of the first iteration in the EKF algorithm.

1. Initialization stage

$\hat{x_0} = 90$ and $P_0 = 1$

For this step, it is advisable to make a good guess and the filter will quickly zoom in to the true value. Although P_0 =var(x_0), the covariance matrix (P_0) is normally set as the identity matrix in practice.

2. Prediction stage

$\hat{x}_{t|t-1} = f(\hat{x}_{t-1}, 0) = \log(90) + (0.1 - 0.5(0.2)^2) * \frac{1}{365} = 90.0197$

$P = 1 * 1 * 1' + 1 * (0.2)^2 \frac{1}{365} * 1' = 1.0001$

Similarly, the other matrices are calculated by subbing in the relevant quantities as per the algorithm.

3.Update stage

H and K are vectors so we are not going to show them here. Our main interest

is on the updated estimate which is $x_1 = 103.9938$. On the other hand, the true state variable (stock price) is given by

$x_1 = 100$

Comparing the two, it is quite clear that the estimate is not far off, even on the first iteration. The rest of the estimates are given on the graph below.

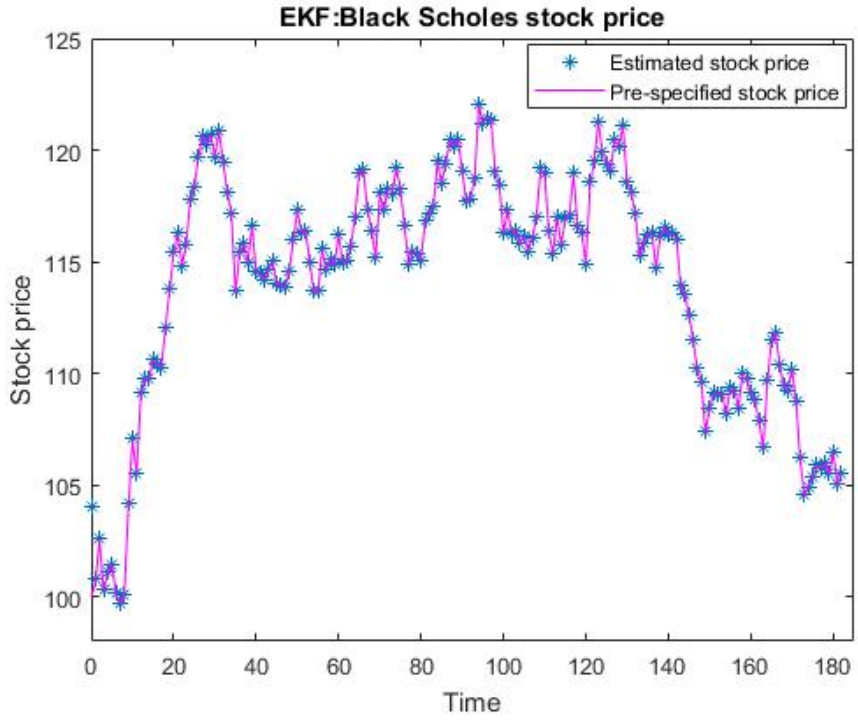

Fig. 2.1: EKF:Estimating the stock price

2.2 Iterated extended Kalman filter

A natural extension of the EKF is the iterated extended Kalman filter (IEKF). This method greatly reduces the 'burn in period' for the EKF as it can quickly recover/estimate the parameters early on. The algorithm is similar to the EKF except in the update step which is replaced by the following set of equations.

$$K_{i+1} = P_t^- H^T(m_i)(H(m_i)P_t^- H^T(m_i) + \mathbf{R})^{-1} \tag{2.15}$$

$$m_{i+1} = \hat{x}_{t|t-1} + K(y_t - h(u_t, m_i) - H^T(m_i)(\hat{x}_{t|t-1} - m_i)) \tag{2.16}$$

$$P_t = (I - KH(m_{i+1}))P_t^- \tag{2.17}$$

To implement this, one has to introduce an inner loop inside the EKF algorithm and iterate until convergence or a fixed number of times i. In other words, the iterations are stopped when there is no significant difference between two consecutive estimates or the iterations have reached the maximum iterations (i) specified in the algorithm.

For demonstration purposes, we will stick to the EKF example above. We only show the update step as everything else stay the same. The estimate we get from the first iteration is given by

$x_1 = 100.0121$

It is quite clear that the IEKF is an improvement to the EKF. This is supported by just analysing the first estimate from the iteration. The rest of the estimates are presented graphically below.

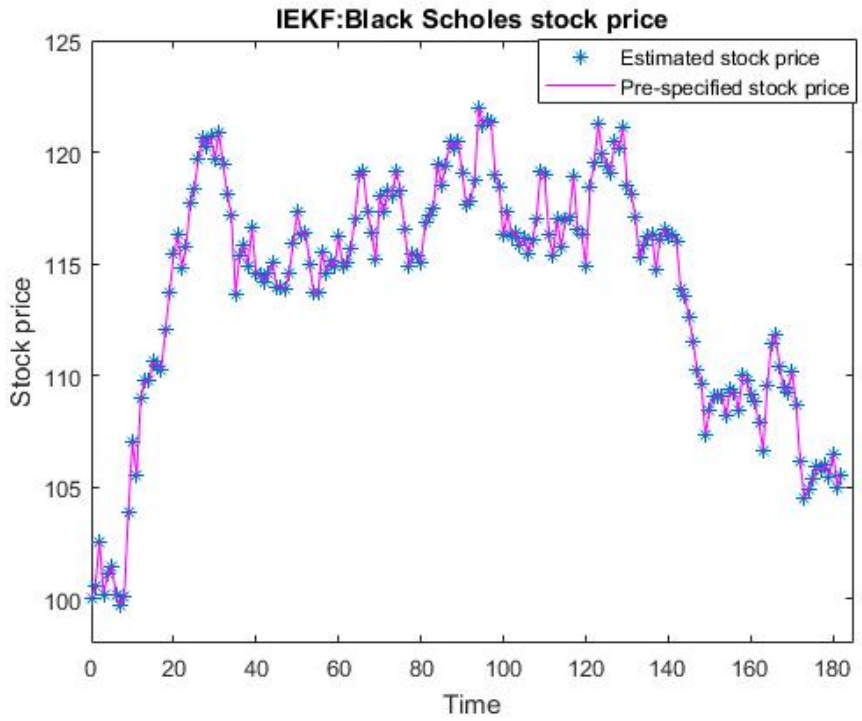

Fig. 2.2: IEKF:Estimating the stock price

2.3 Unscented Kalman filter

When the state and measurement functions f and h above in the EKF are highly non-linear, the EKF can give poor estimates (Hirsa, 2012), resulting from the fact

that only the mean is propagated through the non-linearity. The EKF equations obtained by linearising the system using Taylor series expansion to the first degree can lead to large errors is the posterior mean and covariance and this can lead to the system diverging (Wan and van der Merwe, 2001). The Kalman gain, as we saw in the above formulation of the EKF, depends on the covariance estimates. So, any procedure that significantly over/under estimates the covariance will greatly affect the Kalman gain and this will in turn affect the estimation of the transition matrix. These shortcomings of the EKF are addressed by the UKF. The idea behind UKF is that instead of huge cloud of points, rather use carefully selected points (X_i) called sigma points that better capture the mean and the covariance to third degree which is a better approximation compared to the EKF (Wan and van der Merwe, 2001).skip 0.2 cm

To illustrate how the UKF is performed, we illustrate by way of abstract example, numerical example will follow later. Suppose we have a non-linear function given by

$$y = h(x) \tag{2.18}$$

To compute the conditional density function of y given x we proceed as follows.

Let L =dim(x) where x has mean $\hat{x}$ and the associated covariance matrix is denoted by P_x. We proceed by generating $2L+1$ weighted sigma points $X(i)$ given by

$X(0) = \hat{x}$

$X_{(i=1,2,..,L)} = \hat{x} + (\sqrt{(L+\lambda)P_x})_i$

$X_{(i=1,2,..,L)} = \hat{x} + (\sqrt{(L+\lambda)P_x})_{i-L}$

In the above system, λ is a scaling parameter defined as

$\lambda = \alpha^2(L+\kappa) - L$.

The parameter λ determines the spread of sigma points around x and in the literature, it is advisable to set it as low as possible, say $10^-4 \leq \lambda \leq 1$. The other parameter κ is normally set to $\kappa = 3 - L$ where L is defined as above. Now each set of sigma points $X(i)$ is associated two weights defined as

$W^m(0) = \frac{\lambda}{L+\lambda}$

$W^c(0) = \frac{\lambda}{L+\lambda} + 1 - \alpha^2 + \beta$

$W^m_{(i=1,..,2L)} = W^c_{(i=1,..,2L)} = \frac{\lambda}{2(L+\lambda)}$

The parameter β is normally set to β=2 for Gaussian distributions. We can now employ the function given by equation (3.15) to propagate the sigma points.

$Y_{(i=0,1,...,2L)} = q(X(i))$ with mean and covariance of y given by

$\hat{y} = \sum_{i=0}^{2L} W_i^m Y(i)$

$P_{y_t y_t} = \sum_{i=0}^{2L} W_i^c (Y(i) - \hat{y})(Y(i) - \hat{y})^T$

Now to generate the sigma points and propagating them using state and measurement equations we proceed as follows.

Define

N_x =dim(x)

N_w =dim(w)

N_y =dim(y)

N_v =dim(v)

The above equations are the dimensions for the state process, state noise, measurement process, measurement noise respectively.

Let $L = N_v + N_w + N_x$

Now define column vector $x_t^a = [x_t, w_t, v_t]^T$ with dimensions L.

We proceed by constructing an L by $2L+1$ matrix of sigma points with columns given by

$\chi_0 = \bar{x}$

$\chi_{(i=1,..,L)} = \bar{x} + (\sqrt{(L+\lambda)P_x})_i$

$\chi_{(i=L+1,..2L)} = \bar{x} - (\sqrt{(L+\lambda)P_x})_{i-L}$

It is easy to see that the matrix of sigma points is given by

$$\chi^a = \begin{bmatrix} \chi^x \\ \chi^w \\ \chi^v \end{bmatrix}$$

with the dimensions for χ^x being N_x by $2L+1$, for χ^w being N_w by $2L+1$ and for χ^v being N_v by $2L+1$.

The sigma points are now ready to be propagated using the state transition process.

Below is a summarised version of the above is a form of algorithm.

Algorithm 2 Unscented Kalman Filter

---

1. Initialization stage:

$\hat{x_0} = \mathbb{E}[x_0]$

$P_0 = \mathbb{E}[(x_0 - \hat{x}_0)(x_0 - \hat{x}_0)^T]$

$$\hat{x}_0^a = \mathbb{E}[x_0^a] = \begin{bmatrix} \hat{x}_0 \\ 0 \\ 0 \end{bmatrix}$$

$$P_0^a = \mathbb{E}[(x_0^a - \hat{x}_0^a)(x_0^a - \hat{x}_0^a)^T] = \begin{bmatrix} P_t & 0 & 0 \\ 0 & P_w & 0 \\ 0 & 0 & P_v \end{bmatrix}$$

2. for t=1 to N_y

Calculate the sigma points

$\chi_{t-1}^a(0) = \hat{x}_{t-1}^a$

$\chi_{t-1}^a(i = 1, .., L) = \hat{x}_{t-1}^a + (\sqrt{(L+\lambda)P_{t-1}^a})_i$

$\chi_{t-1}^a(i = L+1, .., 2L) = \hat{x}_{t-1}^a - (\sqrt{(L+\lambda)P_{t-1}^a})_{i-L}$

$$\chi_{t-1}^a = \begin{bmatrix} \chi_{t-1}^x \\ \chi_{t-1}^w \\ \chi_{t-1}^v \end{bmatrix}$$

3. prediction stage

sigma points for x_t

$\chi_{t|t-1}(i) = f(\chi_{t-1}^x(i), \chi_{t-1}^w(i))$

$\hat{x}_{t|t-1} = \sum_{i=0}^{2L} W_i^m \chi_{t|t-1}(i)$

$P_t^- = \sum_{i=0}^{2L} W_i^c (\chi_{t|t-1}(i) - \hat{x}_{t|t-1})(\chi_{t|t-1}(i) - \hat{x}_{t|t-1})^T$

sigma points for y_t

$Y_{t|t-1}(i) = h(\chi_{t-1}(i), \chi^v_{t-1}(i))$

$\hat{y}_t = \sum_{i=0}^{2L} W_i^m Y_{t|t-1}(i)$

$P_{y_t y_t} = \sum_{i=0}^{2L} W_i^c (Y_{t|t-1}(i) - \hat{y}_t)(Y_{t|t-1}(i) - \hat{y}_t)^T$

covariance of x_t y_t is given by

$P_{x_t y_t} = \sum_{i=0}^{2L} W_i^c (\chi_{t|t-1}(i) - \hat{x}_t)(Y_{t|t-1}(i) - \hat{y}_t)^T$

4. Update stage

Calculate the Kalman gain

$K = P_{x_t y_t} P_{y_t y_t}^{-1}$

$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K(y_t - \hat{y}_t)$

$P_t = P_t^- - K P_{y_t y_t} K^T$

$$\hat{x}_t^a = \mathbb{E}[x_t^a] = \begin{bmatrix} \hat{x}_{t|t} \\ 0 \\ 0 \end{bmatrix}$$

$$P_t^a = \mathbb{E}[(x_t^a - \hat{x}_t^a)(x_t^a - \hat{x}_t^a)^T] = \begin{bmatrix} P_t & 0 & 0 \\ 0 & P_w & 0 \\ 0 & 0 & P_v \end{bmatrix}$$

5. end for

Tab. 2.2: UKF algorithm

We are going to use the same example from the EKF section so that we can easily compare the results. Since the algorithm is too long, we will show only challenging calculations from the algorithm. The sigma points from the implemented algorithm are

$$\chi_{t|t-1}(1) = \begin{bmatrix} 90.00 & 99.46 & 81.43 \end{bmatrix}$$

Other calculations involve matrices and its best not to show them here. The updated estimate from the first iteration is $x_t = 99.7686$. The estimate is very close to the true initial stock price of 100.00 and the rest of the estimates are depicted on the next page.

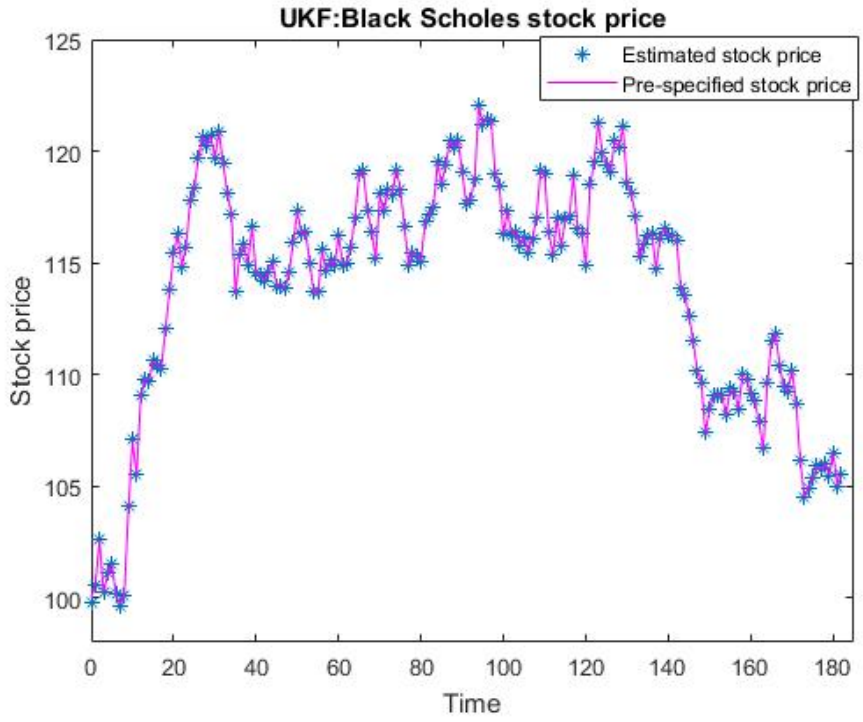

Fig. 2.3: UKF:Estimating the stock price

The three graphs clearly demonstrate the success of non-linear filters in estimating the system state. In chapter 5, we will thoroughly analyse them when exposed to many states and parameters to estimate.

Chapter 3

Stochastic volatility jump diffusion models

Many authors agree that the variance of the underlying (stock) plays a vital role in option valuation as evidenced in the BS model (Lee *et al.*, 2010). Thus, various stochastic volatility models have been introduced that assume that the volatility of the underlying can be modelled by an independent diffusion process. Prominent stochastic volatility models include Bates (1996), Scott (1997), Heston (1993) etc. These models try to explain in a self-consistent way why European options with varying maturities and strike prices can have different BS implied volatilities, termed volatility smile in option pricing literature (Gatheral and Lynch, 2002). Below, we give a brief overview of the models to be calibrated.

3.1 The Heston model

Heston (1993) develops a model to price European options when the underlying asset (stock price process) has stochastic volatility. Under this formulation, the stock price and volatility have their own SDEs. One interesting achievement of this formulation is that a closed form solution for option prices is available, unlike other models where the solution is obtained through numerical techniques like Monte Carlo methods. Also, the Heston (1993) model allows the underlying and the stochastic volatility to be correlated.

Under the Heston model, the risk-neutral dynamics of the stock price process and the volatility of the stock price are given as follows:

$$dS_t = rS_t d_t + \sqrt{\nu_t} S_t dW_t^1 \tag{3.1}$$

$$d\nu_t = \kappa(\theta - \nu_t)d_t + \sigma\sqrt{\nu_t} dW_t^2 \tag{3.2}$$

In the above models, W_t^1 and W_t^2 are two correlated standard Brownian motion processes such that $d[W^1, W^2]_t = \rho dt$. r is the risk free rate and ν_t is the stochastic volatility. Unlike the BS model, volatility is assumed to be a stochastic process with dynamics given by the Cox-Ingersoll-Ross process. This allows volatility to be mean reverting, similar to interest rates modelling using a CIR process. This assumption (stochastic volatility) is in line with observed financial data. The parameter κ is the mean reversion speed while θ is the long run mean of the stochastic volatility. σ is the volatility of the volatility, since now volatility is a stochastic process. This parameter also controls the kurtosis of the returns with low values associated with low kurtosis. Notice also that $\sigma = 0$ renders the stochastic volatility model being deterministic. On the other hand, the parameter ρ controls the skewness of the returns with high values of ρ associated with high volatility. Heston (1993) argues that when the mean reversion is positive, the variance has a steady state distribution with mean θ. Returns over long periods of time will have asymptotically normal distributions and this explains the better performance of the BS model for long dated options Heston (1993).

The graphs below show the effect of θ on the volatility process. We varied θ to

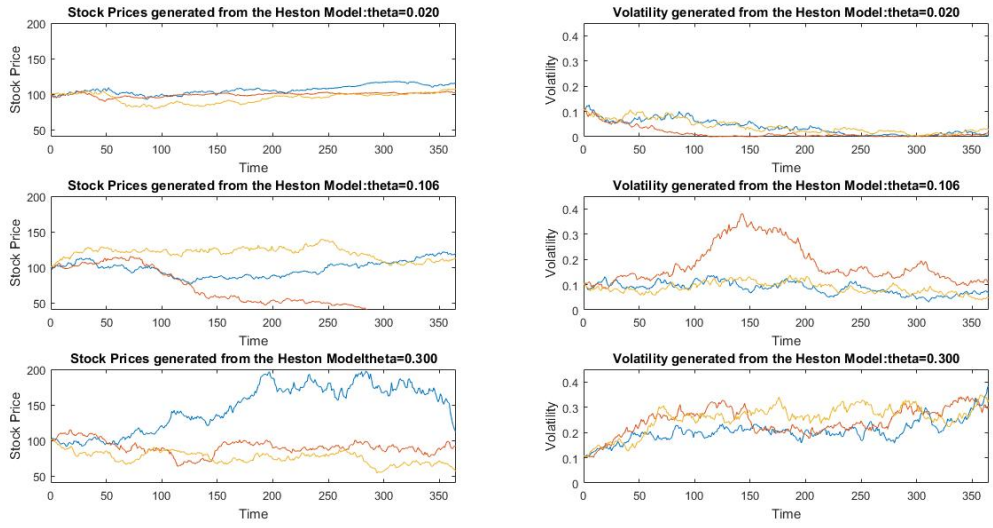

Fig. 3.1: Stock prices and volatility generated from the Heston Model. The parameters used for these graphs are S_0=100, ν_0=0.106, κ=4.03, σ=0.38, ρ=-0.5, T=1, r=0.03, steps=365.

see how the volatility process will evolve. In the first row, we chose a lower value of θ (lower than ν) and it is evident that indeed the parameter θ is the long run mean for volatility. This can also be observed in the other two remaining graphs.

In the middle graph, our initial volatility ν is the same as the long run mean. The last graph, we chose a higher value of θ, that is why volatility is rising. Thus, we can conclude that volatility is governed by a mean reverting process. The Feller condition is given by $2\kappa\theta > \sigma^2$ and the argument is that if the condition is not met, then it's possible to have negative volatility when the CIR process is discretised using Euler or Milstein scheme (Feller, 1951). To guard against this, we normally take the absolute value or reflect the generated/simulated volatility. Another way is to take the maximum of zero and the generated volatility. We implement the latter throughout the simulation exercise. For the Heston parameters above, it is clear that the Feller condition is met. Andersen (2007) develops a more efficient way of simulating the Heston model. The new technique is industry standard but we did not use it for this paper.

3.2 The Bates model

The dynamics of the stock price process S_t under the risk neutral measure are given by

$$dS_t = (r - \lambda\mu_J)S_t d_t + \sqrt{\nu_t}S_t dW_t^1 + J_t S_t dN_t \tag{3.3}$$

$$d\nu_t = \kappa(\theta - \nu_t)d_t + \sigma\sqrt{\nu_t}dW_t^2 \tag{3.4}$$

N_t is an independent Poisson process with parameter λ. J_t is the percentage jump size conditional on the jump occurring and has the following distribution.

$\log(1+J_t) \sim \text{N}(\log(1+\mu_J)\text{-}\frac{\sigma_J^2}{2}, \sigma_J^2)$

W_t^1 and W_t^2 are still two correlated standard Brownian motion processes such that $d[W^1, W^2]_t = \rho dt$.

Since the model is an extension of the Heston model, all other the parameters have their usual meaning and effect on the stock price process and volatility. The significant contribution of the model is to allow the stock price process to be discontinuous. The parameter λ in the above bivariate system of SDEs represents the average frequency of jumps per year. The mean (μ_J) of the percentage jump size (J_t) drives the skewness of the distribution of the returns with positive values of μ_J associated with positively skewed distribution and negative values of μ_J leading to negative skewness. In the BS model, the stock price process is governed by geometric Brownian motion (GBM) which has continuous trajectories. This assumption is not entirely true as stock prices have a tendency to jump in response to major events. The Bates (1996) model rectifies this by including the jump component in the formulation of the model.

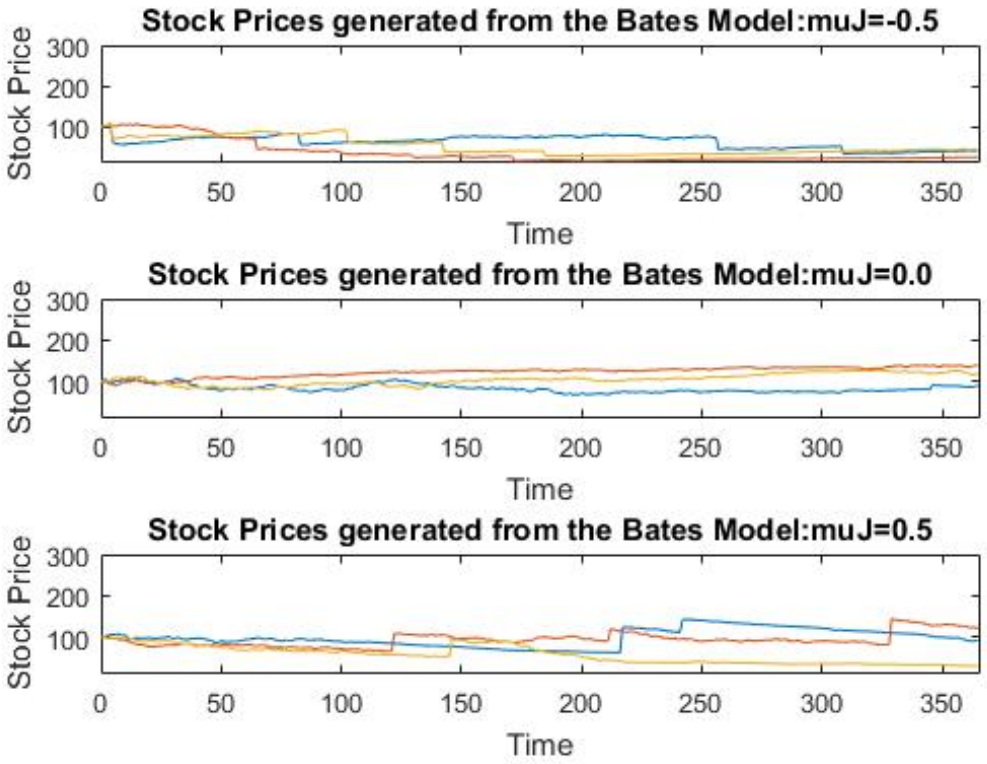

Fig. 3.2: Stock prices generated from the Bates Model. The parameters used for these graphs are S_0=100, ν_0=0.206, κ=4.03, θ=0.04, σ=0.50, ρ=-0.5, λ=2, σ_J=0.17, T=1, r=0.1, steps=365.

Our main interest in the graphs above is the effect of the parameter μ_J on the the stock price. In the first row, we chose a negative value of μ_J and whenever they are jumps, there are in the downward direction. In the middle graph, it can be seen that there are no jumps while in the last graph, we have upward jumps. The model tries to addresses one of the limitations of the BS model which does not cater for jumps.

3.3 Characteristic functions and option pricing

As we have already noted that the LSQ method finds the best parameters that minimise the error between observed prices and model prices. To achieve this, we have to come up with a way to compute the option prices (model prices). Also the measurement function in non-linear filtering is basically a pricing function that takes relevant inputs and outputs the price. Therefore, it is necessary to give an

overview of how pricing is performed under these complicated models.

The pricing of options can be a daunting task particularly when we do not have an analytical formula to use. In the GBM world, things are rather easier as we have a closed form solution that we can employ. This explains the popularity of the model in the investment community. The variants of the BS models mostly do not have the analytical formulas that can be used for option valuation. Nevertheless, numerical methods can be used to calculate the price under the relaxed assumptions. The popular methods include

- Direct integration method of Bakshi and Madan (2000)
- The FFT method of Carr and Madan (1999)
- The COS method of Fang and Oosterlee (2008)
- The Convolution method of Lord *et al.* (2008)
- The fractional FFT method of Chourdakis (1999)

In this paper, we will employ the COS method to calculate option prices, thus it is necessary that we briefly discuss this option pricing method.

3.3.1 The COS method

Fang and Oosterlee (2008) noted that the FFT method can be quite challenging as one has to calculate the damping factor $e^{-\alpha k}$. Carr and Madan (1999) provide guidance on how α should be computed but different values have been suggested in the literature some as low as 0.75 depending on the model used. The COS method does not rely on applying the damping factor. The formulation is based on scaled log stock price s_T =log(S_T/K).

To calculate the option prices using the cosine method, the following formula was derived by Fang and Oosterlee (2008).

$$V_0 = e^{-rT} \sum_{n=0}^{N-1}{}' Re[\phi_{s_T}(\frac{n\pi}{b-a}) e^{-in\pi\frac{a}{b-a}}]\upsilon_n \tag{3.5}$$

where

$\sum'$ indicates that the first term in V_0 is multiplied by $\frac{1}{2}$.

N is the number of terms to be summed.

a and b are the limits of integration.

ϕ_{s_T} is our characteristic function.

$\upsilon_n = \frac{2}{b-a}K(\chi_n(a,b) - \psi_n(a,b))$

$\chi_n(c,d) = \int_c^d e^s \cos(n\pi \frac{s-a}{b-a})ds$

$\psi_n(c,d) = \int_c^d \cos(n\pi \frac{s-a}{b-a})ds$

The COS method introduces two errors in the evaluation of the option prices.

- The truncation of the integral to $[a, b]$.
- The summation of only N terms.

To deal with the first error, Fang and Oosterlee (2008) recommend using the following to compute the new bounds of integration.

$a = c_1 - \sqrt{c_2\sqrt{c_4}}$

$b = c_1 + \sqrt{c_2\sqrt{c_4}}$

where $c_n = \frac{d}{dt^n}\log\mathbb{E}[e^{ts}]$, the nth cumulant of s =log(S_T/K).

The method performs better than other pricing methods which are slow and sometimes produce inaccurate results (Fang and Oosterlee, 2008). Therefore, I will use it for pricing in this paper. The set up above is general and one has to adapt υ_n in order to price European call, put and binary options.

Below, we compare the accuracy of this method using the BS model. We calculate European call option price using the BS analytical formula for comparison. Remember the characteristic function for the scaled log price following the geometric Brownian motion is given by

$\phi_{S_T}(u)$=e(iu$(log(S_0/K) + (r - 0.5\sigma^2)T) - 0.5\sigma^2 T u^2)$

We can see that by 15^{th} term $(N = 15)$, the COS method price is the same as the BS analytical price. So, the truncation error can be reduced by using a larger value of N.

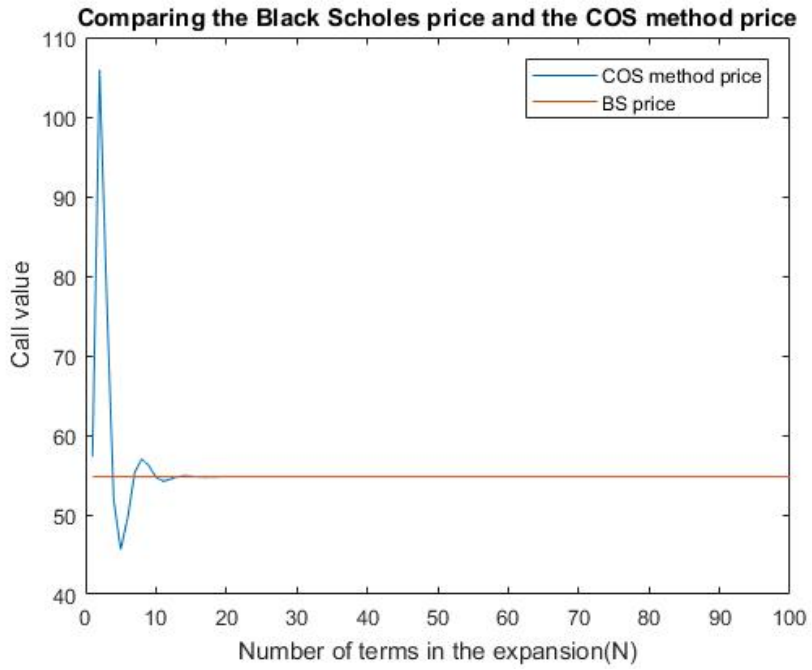

Fig. 3.3: COS method option pricing example. The parameters used for pricing are S_0=100, σ=0.2, T=1, r=0.1, K=50.

Now we are ready to price using the COS method but we need the characteristic functions for our models.

3.3.2 The Heston model characteristic function

In the original paper, Heston (1993) derived a characteristic function for the model which was later found to have some drawbacks under certain conditions. In 2007, Albrecher *et al.* (2007) discuss two formulations of the Heston model characteristic function and favoured the following formulation.

$$\phi_{S_T}(u,t) = E[e^{iulog(S_t)}|S_0,\nu_0^2] = e^{A(u,t)+B(u,t)+C(u,t)} \tag{3.6}$$

where

$$A(u,t) = iu(\log(S_0)+rt)$$

$$B(u,t) = \theta\kappa\sigma^{-2}[(\kappa-\rho\sigma iu-d(u))t - 2\log(\tfrac{1-g(u)e^{-dt}}{1-g(u)})]$$

$$C(u,t) = \tfrac{\nu_0\sigma^{-2}[(\kappa-\rho\sigma iu-d(u))(1-e^{-dt})]}{1-g(u)e^{-dt}}$$

$$d(u) = \sqrt{(\rho\sigma iu-\kappa)^2+\sigma^2(iu+u^2)}$$

$g(u) = \frac{\kappa - \rho\sigma iu - d(u)}{\kappa - \rho\sigma iu + d(u)}$

The graphs below show plots of volatility smile generated by varying ρ and σ.

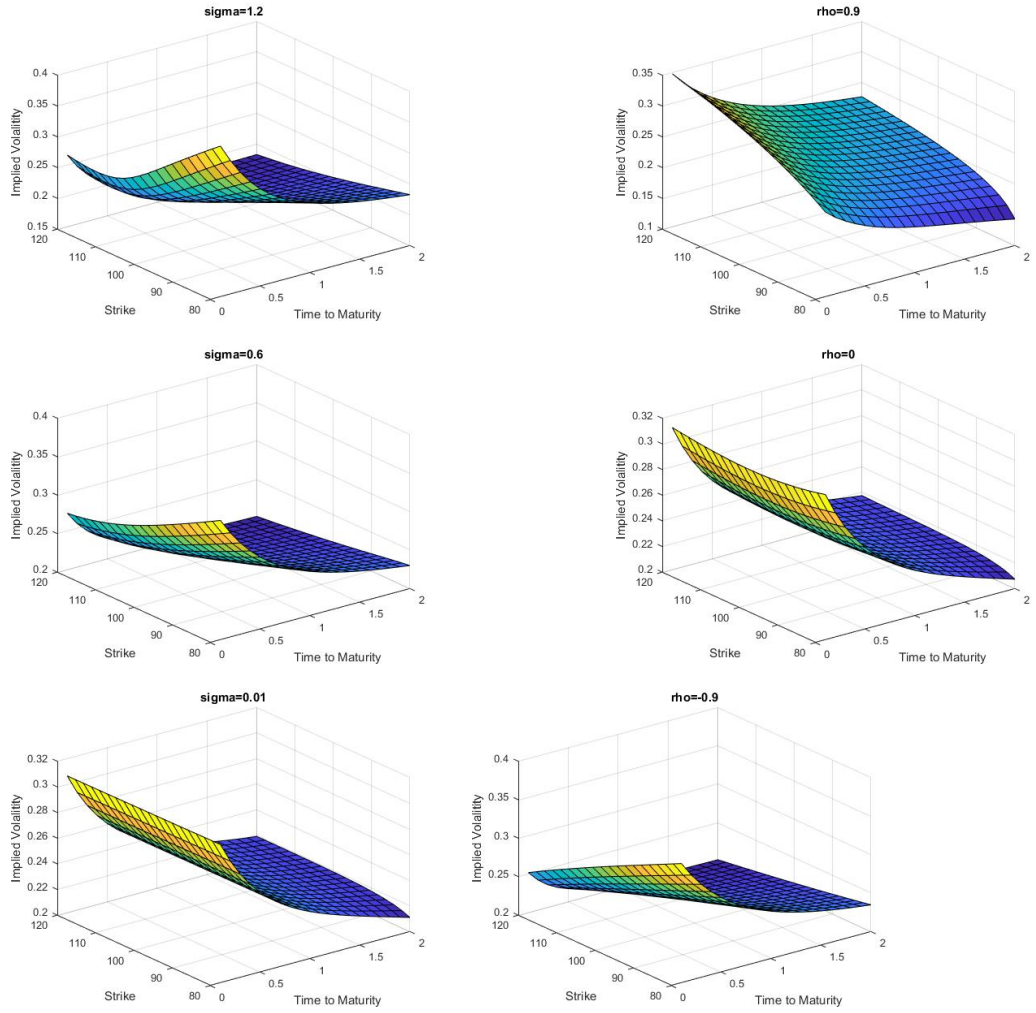

Fig. 3.4: The effect of σ and ρ on the volatility surface. Our parameters are S_0=100, ν_0=0.106, κ=4.03, θ=0.04, σ=0.38, ρ=-0.5, r=0.03, $K = \Big[80, 81, ..., 119, 120\Big]$, $T = \Big[0.1, 0.2, ..., 1.9, 2\Big]$.

A matrix of call option prices was generated using the COS method for varying maturity and strike. The MATLAB built in function called the *fzero* was used to find the implied volatility by calling the Black-Scholes formula.

3.3.3 The Bates model characteristic function

Since the model is similar to the Heston model, the characteristic function also looks similar. The characteristic for the model is given by

$$\phi_{S_T}(u,t) = E[e^{iu\log(S_t)}|S_0, \nu_0^2] = e^{(A(u,t)+B(u,t)+C(u,t)+D(u,t))} \tag{3.7}$$

where

$A(u,t) = iu(\log(S_0) + rt)$

$B(u,t) = \theta\kappa\sigma^{-2}[(\kappa - \rho\sigma iu - d(u))t - 2\log(\frac{1-g(u)e^{-dt}}{1-g(u)})]$

$C(u,t) = \frac{\nu_0\sigma^{-2}[(\kappa-\rho\sigma iu-d(u))(1-e^{-dt})]}{1-g(u)e^{-dt}}$

$D(u,t) = -\lambda\mu_J iut + \lambda t[(1+\mu_J)^{iu}e^{\frac{1}{2}\sigma_J^2 iu(iu-1)} - 1]$

$d(u) = \sqrt{(\rho\sigma iu - \kappa)^2 + \sigma^2(iu + u^2)}$

$g(u) = \frac{\kappa-\rho\sigma iu-d(u)}{\kappa-\rho\sigma iu+d(u)}$

We also show the volatility surface generated by varying μ_J and λ on the next page.

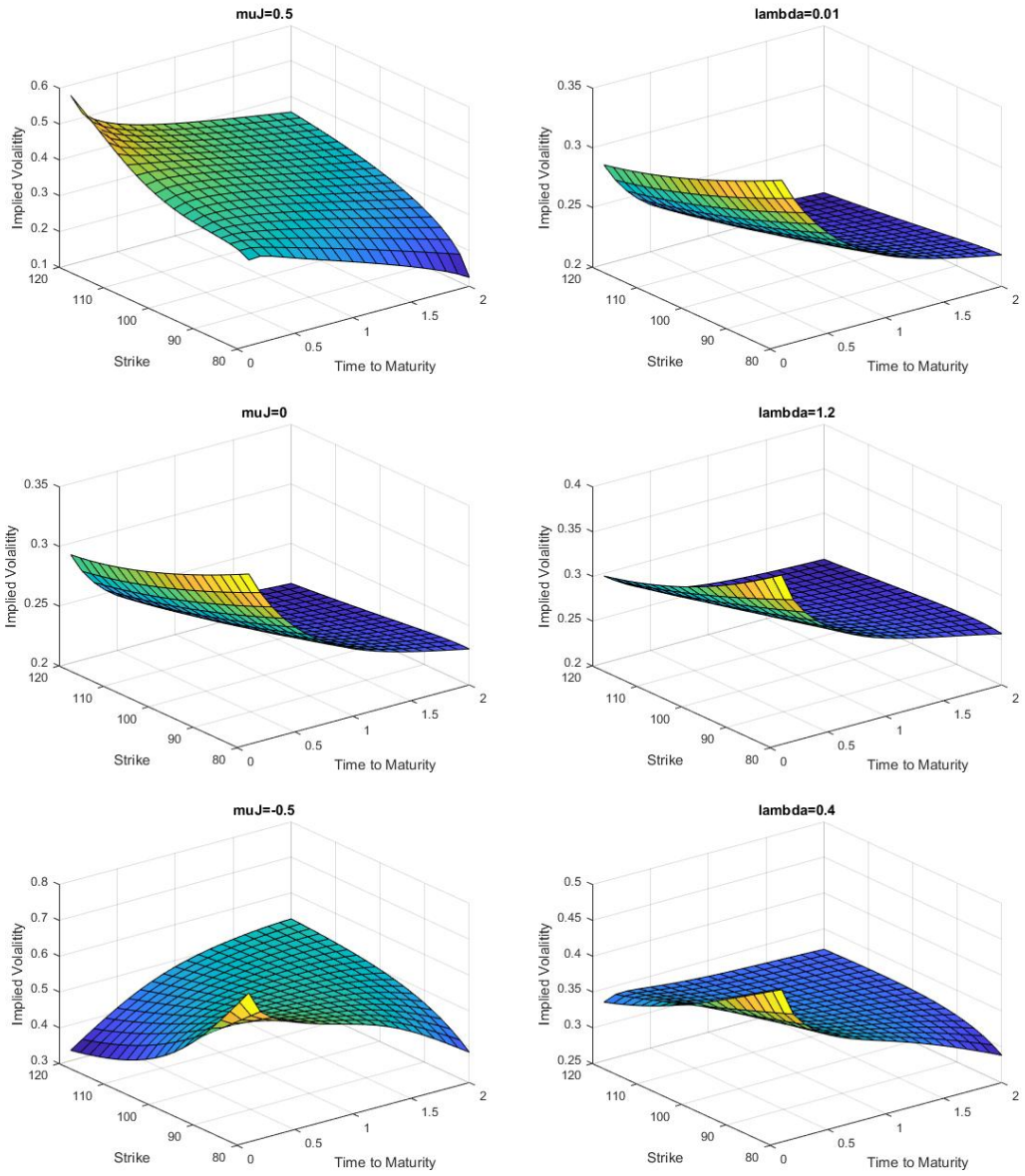

Fig. 3.5: The effect of μ_J and λ on the volatility surface. Our parameters are S_0=100, ν_0=0.106, κ=2.03, θ=0.04, σ=0.38, ρ=-0.7, λ=0.59, μ_J=-0.05, σ_J=0.07, r=0.03, $K = \Big[80, 81, ..., 119, 120\Big]$, $T = \Big[0.1, 0.2, ..., 1.9, 2\Big]$.

A matrix of call option prices was generated using the COS method for varying maturity and strike. The MATLAB built in function called the *fzero* was used to find the implied volatility by calling the Black-Scholes formula.

3.4 Discrete delta hedging

The purpose of a model in derivatives management according to Guo (2013) is to calibrate, price and hedge derivative contracts. A trader can hedge a financial contract until maturity to minimise potential losses.

Consider a trader who has sold a vanilla European call option. The trader can perfectly hedge the option using the underlying and the risk free asset in the Black-Scholes when re-balancing is done in continuous time . Let $V(t, S_t)$ be the value of the derivative at time t. Assume that the contract can be hedged using only the risk free asset and the stock. Then, the value of a hedged portfolio is given by

$$P(t, S_t) = \theta_s S(t) + \theta_B B(t) \tag{3.8}$$

In the above equation θ_s and θ_B are the weights of the stock and the risk free asset (bank account). They are chosen in such a way that the value of the hedged portfolio $P(t, S_t)$ best replicates the value of the derivative $V(t, S_t)$. For delta hedging, things are simpler as we can set up a self-financing portfolio such that

$$\frac{\partial P(t, S_t)}{\partial S} = \frac{\partial V(t, S_t)}{\partial S} \tag{3.9}$$

$$P(t, S_t) = V(t, S_t) \tag{3.10}$$

The vector of weights used in replicating portfolio is given by

$$\theta_s = \frac{\partial V(t, S_t)}{\partial S} \tag{3.11}$$

$$\theta_B = P(t, S_t) - \theta_s S_t \tag{3.12}$$

In BS, θ_s is given by $N(d_+) = N(\frac{\log \frac{S_0}{K} + (r - q + 0.5\sigma^2)T}{\sigma\sqrt{T}})$. For this paper, I used the COS method for pricing and we derived the following which can be used for computing the delta.

$$\frac{\partial \phi_{ST}(u)}{\partial S} = \phi_{ST}(u)\frac{iu}{S} \tag{3.13}$$

where ϕ_{ST} is the characteristic function.

Finite difference schemes also give good approximations to the Greeks. In this paper, we are going to employ delta hedging to the stochastic models under study.

It is well-known fact that it is not easy to replicate the option using only the underlying and the risk free asset. In the literature for hedging in stochastic volatility setting, adding another derivative contract of longer maturity than the option being hedged is the norm (He *et al.*, 2005). This will ensure that the hedged portfolio is able to deal with jumps in the underlying.

Chapter 4

Calibration techniques

4.1 The least square method

The least square method was first introduced by Carl Friedrich Gauss and ever since, there have been developments to improve the technique. The method is frequently used in regression analysis to find the model parameter that minimise the error in least square sense. The error is usually defined to be the squared difference between the model option prices and observed option prices. The technique uses optimisation routines like gradient descent, particle swarm optimisation, genetic algorithm. One has to decide on the objective function to be used and the common one is the root mean square error (RMSE). The idea behind this LSQ method is to estimate the best parameters that minimise the error (RMSE).

Suppose we have observed market option prices (European call options for instance) $C_i^{market}(K_i, \tau_i)$ and estimated option prices from the proposed model $C_i^{model}(K_i, \tau_i; \theta)$, we can estimate the parameters of the model (θ) that minimizes the squared residuals. For our case, K_i represents a vector of strike prices and similarly τ_i is a vector of maturities.

$$L_t(\hat{\theta}) = \sum_{s=1}^{t} \sum_{i=1}^{N_s} (C_i^{market}(K_i, \tau_i)) - C_i^{model}(K_i, \tau_i; \hat{\theta}))^2 \tag{4.1}$$

In the above objective function, the first sum is over time and the second sum is the sum of the number of options per time period. For the LSQ method, traders normally use today's data for parameter estimation, unlike non-linear filtering methods to be discussed later. So, t in the summation is normally set to 1. To minimise the error, an objective function must be chosen. Other objective functions normally used in practice are briefly discussed below.

The mean square error : $\frac{1}{N_t} \sum_{i=1}^{N_t} (C_i^{market}(K_i, \tau_i)) - C_i^{model}(K_i, \tau_i; \hat{\theta}))^2$

The relative mean square error : is closely related to the statistic above, its defined as,

$\frac{1}{N_t}\sum_{i=1}^{N_t}\frac{(C_i^{market}(K_i,\tau_i))-C_i^{model}(K_i,\tau_i;\hat{\theta}))^2}{(C_i^{market}(K_i,\tau_i))}$

These loss functions are easy to implement but have some challenges. Short dated options and deep out of the money options contribute less to the sum when using the mean square error loss function. One solution is to use the relative mean square error, but it also has some challenges, as options with low market value value contribute more to the sum because of the divisor ($C_i^{market}(K_i,\tau_i)$). One can resort to other loss/objective functions like the following:

The average relative percentage error (ARPE): $\frac{1}{t}\sum_{s=1}^{t}\sum_{i=1}^{N_s}\frac{|C_i^{market}(K_i,\tau_i))-C_i^{model}(K_i,\tau_i;\hat{\theta})|}{C_i^{market}(K_i,\tau_i))}$

The relative percentage error (RPE):

$\frac{1}{t}\sum_{s=1}^{t}\frac{N_s}{\sum_{i=1}^{N_s}C_i^{market}(K_i,\tau_i)}\sum_{i=1}^{N_s}|C_i^{market}(K_i,\tau_i))-C_i^{model}(K_i,\tau_i;\hat{\theta})|$

The least square methods have some challenges though.

1. Unless initial starting conditions are already close to the global minimum, there is a possibility of convergence to some undesirable values (Strutz, 2011).

2. Unlike the non-linear filtering methods discussed below, the least square methods use only cross sectional data. Lindström *et al.* (2008) argue that using only current data to predict future prices implies that past information is not useful in making inference about the future.

To illustrate how state variable (stock price) and parameter (volatility) are estimated using this method, I will use the BS model. The parameters used for this example are

S_0=100
r=0.1
$dt = \frac{1}{365}$
$T = \frac{1}{2}$
$K = \left[90, \quad 91, \quad 92, ..., 109\right]$
$t = \left[0, \quad \frac{1}{365}, \quad \frac{2}{365}, ...\frac{182}{365}\right]$
σ=0.15

We used the MATLAB built function called *lsqnonlin*. This function deals with non-linear optimisation problems and uses trust region methodology to find the parameter estimates. The function implicitly minimises mean square error. At the heart of this optimisation function are three sets of vectors called the start, upper

bound (ub) and lower bound (lb) vectors. The lower and upper bound vector act as bounds for estimates and the *lsqnonlin* function will look for estimates within this boundary. The start vector should have initial guess obeying the boundary vectors. Below are the three vectors for this example. In all the vectors, the first element is the stock price and the second element is volatility. Our three crucial vectors are,

start= $\begin{bmatrix} 80 & 0.05 \end{bmatrix}$
lb= $\begin{bmatrix} 5 & 00.0001 \end{bmatrix}$
ub= $\begin{bmatrix} 200 & 0.5 \end{bmatrix}$

So our initial guess for the stock price is 80 but its constrained between 5 and 200 etc.

Unlike sequential calibration methods to be discussed below, the algorithm for the LSQ is much shorter. Below is a MATLAB pseudo code for the first estimate.

lsqnonlin(@(par) (BScall(par(1),par(2),Strike)-y(:,1)),start,lb,ub)= $\begin{bmatrix} 99.99 & 0.1499 \end{bmatrix}$

In the above code, BSCall is the BS price for the call option and y is the vector of simulated prices. The algorithm will search for estimates (par) that are within the acceptable bounds. The algorithm implicitly calculates the mean square error until the error is less than the *lsqnonlin* function tolerance. The LSQ method is able to recover the parameters well since since only two parameters have to be estimated. The parameter estimates are graphically encapsulated below.

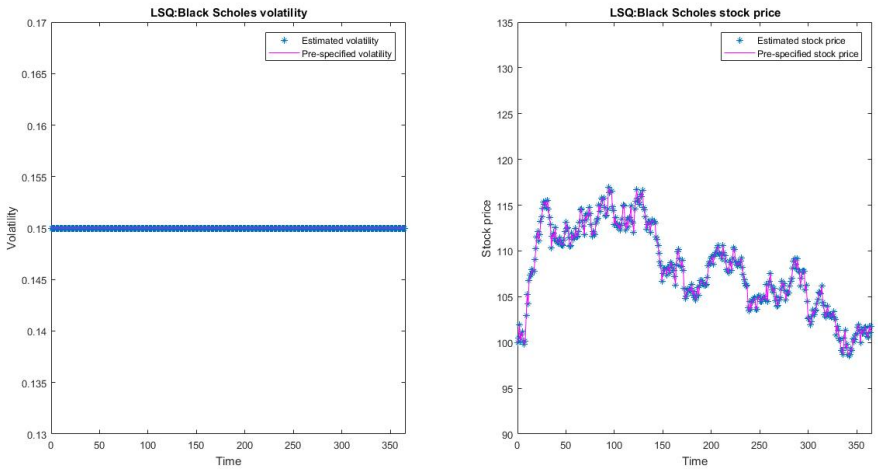

Fig. 4.1: LSQ:Parameter estimation-stock price and volatility.

4.2 Sequential calibration

Lindström *et al.* (2008) show how non-linear filtering methods can be used to jointly estimate the state variable and parameters of a model. Sequential calibration expands the state variable vector by adding the parameters to the state variables, and gives the parameters random walk dynamics that contain some noise, so as to explore the parameter space. The techniques use a set of equations and consecutive data inputs to estimate parameters of a model as the measured values have random error. As the number of iterations/steps increases, the estimates will converge to the true parameter values.

Below we show how non-linear filtering can be used for state and parameter estimation.

4.2.1 Extended Kalman filter

As we have already presented much of the work in chapter two, we will leave out much detail to avoid repetition. To illustrate EKF sequential calibration, I will continue with BS example from chapter two. In this chapter however, we are going to estimate the state variable (stock price) and the parameter of the model, σ. Sticking with parameters from the LSD section we can simulate stock prices and calculate European option prices. The option prices are stored in a matrix y. Now we **pretend** that we never knew the simulated stock price and the pre-specified σ. We can follow the algorithm from chapter two to recover the stock price (state variable) and σ (parameter). For clarity, we are going to use the same notation as per the algorithm.

Our state transition equation is given as

$$x_{t+\Delta t} = f(x_t) + w_t = \begin{bmatrix} \log S_t + (r - 0.5\sigma^2)\Delta t \\ \sigma_t \end{bmatrix} + \begin{bmatrix} \sigma_t^2 \Delta t \\ q^2 \Delta t \end{bmatrix} = \begin{bmatrix} X(1)_t + (r - 0.5X(2)^2)\Delta t \\ X(2)_t \end{bmatrix} + \begin{bmatrix} X(2)_t^2 \Delta t \\ q^2 \Delta t \end{bmatrix}$$

We chose our variance for the measurement noise to be $\mathbf{R} = 1/16$, thus one fourth (standard deviation) has to be added as random noise to simulated option prices. In financial markets, option prices have a bid ask spread, thus it is not easy to arrive at the true option price. The variance is added because of this phenomenon.

Note also that our variance for the state noise is given by

$$\mathbf{Q} = \begin{bmatrix} X(2)_t^2 \Delta t & 0 \\ 0 & q^2 \Delta t \end{bmatrix}$$

The first entry in the above matrix is the variance of the random part $(\sigma\sqrt{\Delta t}Z)$ of the log stock price. The second non zero entry can be determined using optimisation techniques.

The parameter noise is important as it allows the filter to search for the true state. But there is a trade-off, adding more noise can lead to noisy estimates that hardly converge to the true parameters. On the other hand, choosing a lower noise can lead to the filter failing to find the state.

We also need to calculate the Jacobian matrices and we get

$$F(x,w) = \frac{df}{dx_t}|_{(\hat{x}_{t-1},0)} = \begin{bmatrix} 1 & -X(2)_t \Delta t \\ 0 & 1 \end{bmatrix}$$

$$G(x,w) = \frac{df}{dw_t}|_{(\hat{x}_{t-1},0)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$H(x) = \frac{dh}{dx_t}|_{(\hat{x}_{t|t-1},0)}$$

The matrix H can be calculated using finite difference methods where the first column is the sensitivity of option prices to the stock and the second column is the sensitivity of the option prices to volatility. Note that in the case of the BS model, we have analytical formulas. For more complicated models to be discussed in the next chapter, we have to resort to finite difference methods.

Now we are ready to do the filtering.

1. Initialization stage: $\hat{x_0} = \begin{bmatrix} 90 & 0.25 \end{bmatrix}$ and $P_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

2. Predict stage:

$\hat{x}_{t|t-1} = f(\hat{x}_{t-1}, 0) = \begin{bmatrix} 90.0081 & 0.25 \end{bmatrix}$

$$P = FPF' + GQG' = \begin{bmatrix} 1.0001 & -0.0006 \\ -0.0006 & 1.00002 \end{bmatrix}$$

Similarly, the other matrices are calculated by subbing in the relevant quantities as per the algorithm.

3. Update stage:

We can also sub in the relevant quantities to get

$x_1 = \begin{bmatrix} 105.39 & 0.1150 \end{bmatrix}$

True estimates

$$x_1 = \begin{bmatrix} 100.00 & 0.15 \end{bmatrix}$$

The rest of the estimates are given on the graph.

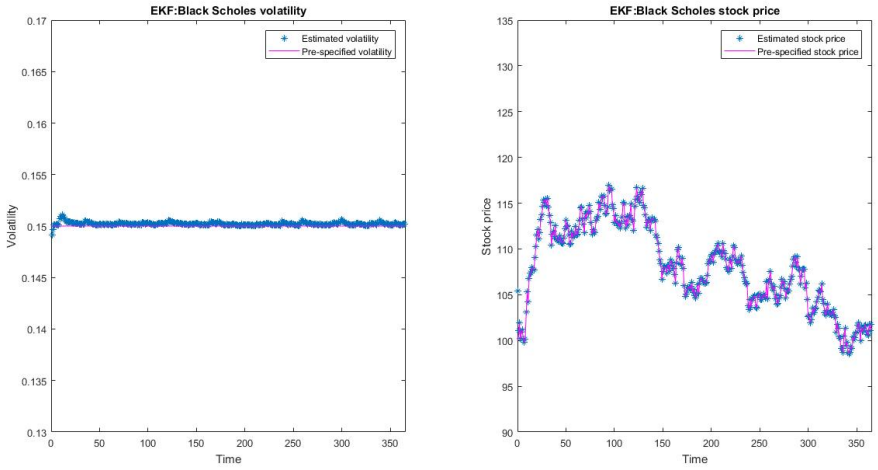

Fig. 4.2: EKF:Parameter estimation-stock price and volatility.

4.2.2 Iterated extended Kalman filter

A natural extension of the EKF is the iterated extended Kalman filter where we replace the update step by an inner loop. In this loop, we iterate a until convergence or a fixed number of times. Remember the update step under IEKF is given by

$$K_{i+1} = P_t^- H^T(m_i)(H(m_i)P_t^- H^T(m_i) + \mathbf{R})^{-1}$$

$$m_{i+1} = \hat{x}_{t|t-1} + K(y_t - h(u_t, m_i) - H^T(m_i)(\hat{x}_{t|t-1} - m_i))$$

$$P_t = (I - KH(m_{i+1}))P_t^-$$

The good thing about this method is that the burn in period is greatly reduced. The first estimate from the iteration is given as

$$x_1 = \begin{bmatrix} 99.99 & 0.15 \end{bmatrix}$$

Comparing these to the pre-specified parameters, we can see clearly that the IEKF performs much better than EKF.

The remaining estimates are given shown on the graph on the next page.

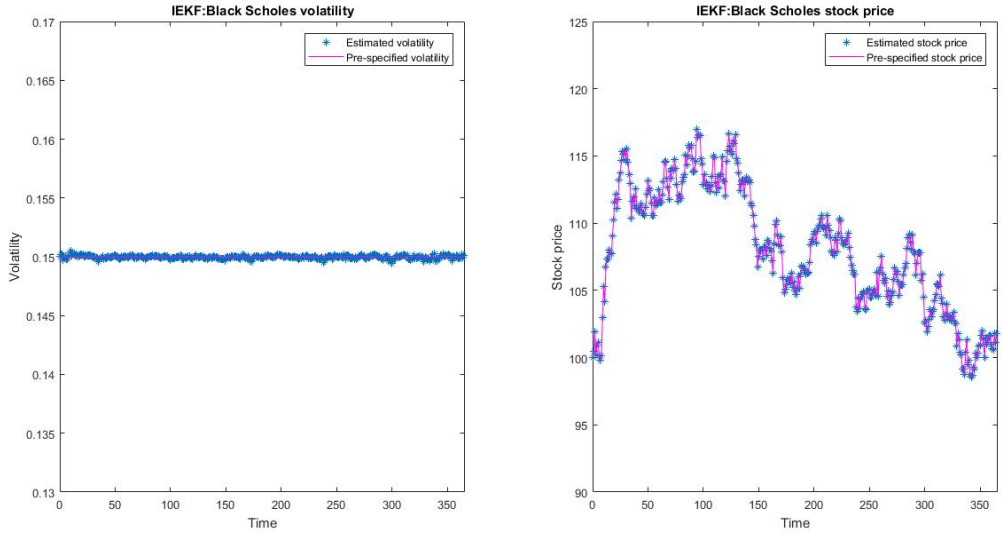

Fig. 4.3: IEKF:Parameter estimation-stock price and volatility.

4.2.3 Unscented Kalman filter

This non-linear filtering method was developed to address some shorfalls of EKF Julier and Uhlmann (1996). Instead of calculating Jacobian matrices, a set of point called sigma points are calculated. The algorithm is much longer than the two above.

The first estimate from this filtering method is

$$x_1 = \begin{bmatrix} 107.18 & 0.1377 \end{bmatrix}$$

It is clear that IEKF performs better than the two non-linear filtering methods in this example.

The remaining estimates are depicted on the graphs on the the next page.

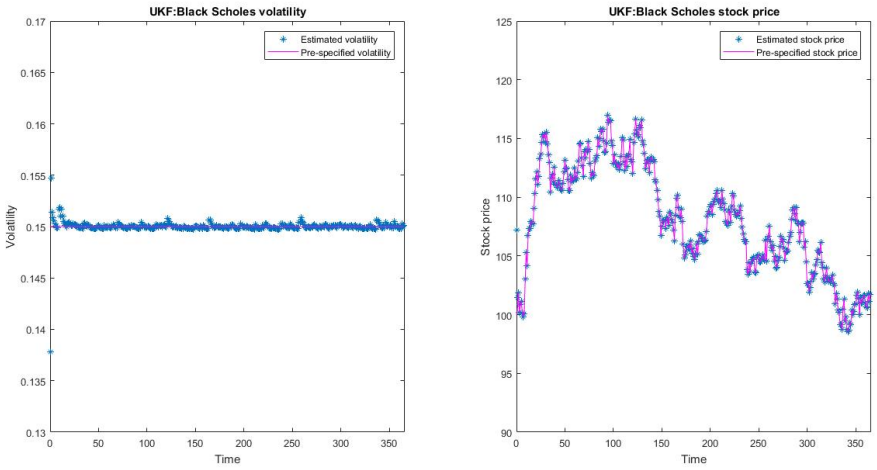

Fig. 4.4: UKF:Parameter estimation-stock price and volatility.

One thing worth mentioning is that the measurement function h and the state transition equations are all the same in these three methods. Unlike, most calibration techniques, sequential calibration of Lindström *et al.* (2008) does not involve any optimisation procedure and this lead to faster parameter estimation than most methods. We will see this when we analyse the methods in the next chapter.

Chapter 5

A simulation study

As this paper closely follows the approach adopted by Lindström *et al.* (2008), we maintain most of the assumptions and only making minor adjustments to improve the estimates. It is a well-known fact that option prices in the market are quoted with a bid ask spread. Lindström *et al.* (2008) argue that the bid ask spread for their simulated S&P 500 option prices is about \$1.00. They hyphothesised that 95% of true option prices will lie within this spread. To closely, capture this, Lindström *et al.* (2008) add a noise of $\mathbf{R} = \frac{1}{16}$ for the simulated option price in their paper. This ensures that a 95% confidence interval has a width of \$1.00. However, for parameter recovery, we found the noise to be too high as all the filters missed the true parameters and we had to reduce it to 0.1 even though we adopted Lindström *et al.* (2008) initial stock price (S_0) and strike prices .

For the Heston model, the results of parameter recovery using the four calibration techniques can be found in the appendix. For the purpose of illustration, I will use the Bates model as it has many parameters. The dynamics of the Bates model are given by equation (2.4) and (2.5). I have made a slight adjustment by including two time varying parameters being ρ_t and θ_t. This relaxed formulation of the Bates dynamics is to ensure that the filters can still cope with time varying parameter like all other calibration methods.

The simulated data was obtained using the following pre-specified parameters.

These parameter values are the same ones used by Lindström *et al.* (2008) and to properly replicate his paper, we stick to them. The parameters were chosen based on Bakshi *et al.* (1999) paper as they mimic the attributes of S&P stock index

Parameter	S_0	r	ν_0	κ	θ_0	σ	ρ_0	λ	μ_J	σ_J
Value	1000	0.03	0.0576	2.03	0.04	0.38	-0.7	0.59	-0.05	0.07

Tab. 5.1: Pre-specified Bates parameters

(Lindström *et al.*, 2008). For each day, we generate 40 call option prices using the COS method. We use 8 different strike prices ranging from R800 to R1200 and maturity vector given by τ=[1/12, 4/12, 9/12, 15/12, 18/12].

5.1 The standard least square calibration

The least square methods are optimization routines. For this paper, I used the MATLAB built in function called the *lsqnonlin*. This function uses trust region optimisation procedure to find a vector of parameters that minimise the objective function. This function implicitly computes the mean square error. At the heart of the *lsqnonlin* are the 3 vectors being the starting point, the lower bound and the upper bound for the parameters to be estimated. Lindström *et al.* (2008) suggest using the true parameter as the starting values. The LSQ method will then have to track the time varying parameters and the hidden state. This procedure greatly increases the speed unlike using initial guess which might be far off.

5.2 The non-linear filtering methods

Lindström *et al.* (2008) calculated the variance for the state noise for the continuous parameters and jump parameters and they obtained $\mathbf{Q^C}$=0.005 and $\mathbf{Q^J}$=0.0001 using quasi-maximum likelihood estimation. We decided to adjust them to ensure that on the last day/step, the noise has reduced by 50%. For instance, we used $\mathbf{Q^J} = ((k + 0.5 - 0.5i)/k)$ for the jump part, where $k = 365$ steps/days and i is the i^{th} step/day. We believe the noise dies out as the number of steps increases and this adjustment exactly implements this idea.

Just like the LSQ method, the filters need a vector of starting values at the start of the algorithm. It is well understood in the literature that the non-linear filtering methods quickly zoom in to the true parameters regardless of how bad the initial vector was. But Lindström *et al.* (2008) suggest using parameter estimates obtained from the LSQ as initial vector.

The parameters have to be transformed and we use the following function to transform all of our parameters.

$X = -\log \frac{b-x}{x-a}$

where

X=transformed parameter

a=lower bound for untransformed parameter

b=upper bound for untransformed parameter
x=untransformed parameter

Lindström *et al.* (2008) argue that its possible for the algorithm to break down if the parameters are not transformed. To untransform the parameter, we can invert the above equation to get

$$x = \frac{b-a}{1+e^{-X}} + a$$

Notice how the untransformation will ensure that the parameters are always within the acceptable range. If the transformed parameter is too high, the divisor in the above equation evaluates to one and we get $x = b$. The following are the bounds for the simulation study:

$$a = \begin{bmatrix} 0 & 0 & 0 & -1 & 0 & -1 & 0 \end{bmatrix}$$
$$b = \begin{bmatrix} 10 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The parameters are arranged as κ, θ, σ, ρ, λ, μ_J, σ_J. The bounds have to make sense too. For instance, we expect ρ to be between -1 and 1. The same bounds were also used in the LSQ method.

I have already explained how to estimate parameters using non-linear filtering. The major difference here is that our measurement function h and our process function f takes transformed parameters as inputs. Sequential calibration gives parameters a random walk dynamics. However, a random walk can go from -∞ to ∞, whereas the parameters may need to be constrained. To constrain the parameters, we transform them as indicated. This means that we have to write the option price in the measurement equation h and state transition equation f as a function of transformed expanded state variables. To get, our estimates, we can untransform the transformed parameters. Our estimates will also be within the bounds given by vectors **a** and **b**. Our state transition equation is now given by

$$x_{t+\Delta t} = f(x_t) + w_t = \begin{bmatrix} X(1)_t + X(2)_t(X(3)_t - X(1)_t^2)\Delta t \\ X(2)_t \\ X(3)_t \\ X(4)_t \\ X(5)_t \\ X(6)_t \\ X(7)_t \\ X(8)_t \end{bmatrix} + \begin{bmatrix} X(4)_t^2 X(1)_t \Delta t Z \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \\ \varepsilon_7 \\ \varepsilon_8 \end{bmatrix}$$

where Z is a standard normal random variable.

The Bates model has seven parameters and the hidden state (volatility). The estimates can be presented in a vector.

$$\theta = \Big[\nu_t, \kappa, \theta, \sigma, \rho, \lambda, \mu_J, \sigma_J\Big] = \Big[x(1), x(2), x(3), x(4), x(5), x(6), x(7), x(8)\Big]$$

The rest of vectors are given as

$\mathbf{Q}$=8 by 8 diagonal matrix with errors on the main diagonal.

$\mathbf{G}$=8 by 8 identity matrix.

$\mathbf{P_0}$=8 by 8 identity matrix.

5.3 Simulation results

The graphs below shows parameter estimates from the standard least square method and the three non-linear filtering methods. It is clear that all methods are able to recover the parameters well.

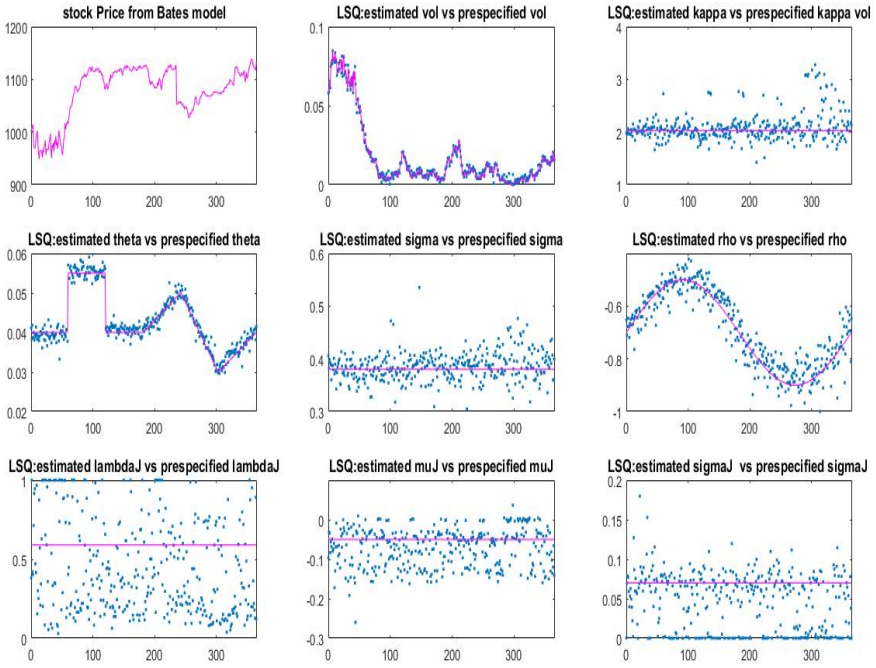

Fig. 5.1: Parameter estimation for the Bates model using LSQ. The graphs above show parameter estimates for the Bates model for a period of 365 days. The pre-specified values are from Table 5.1

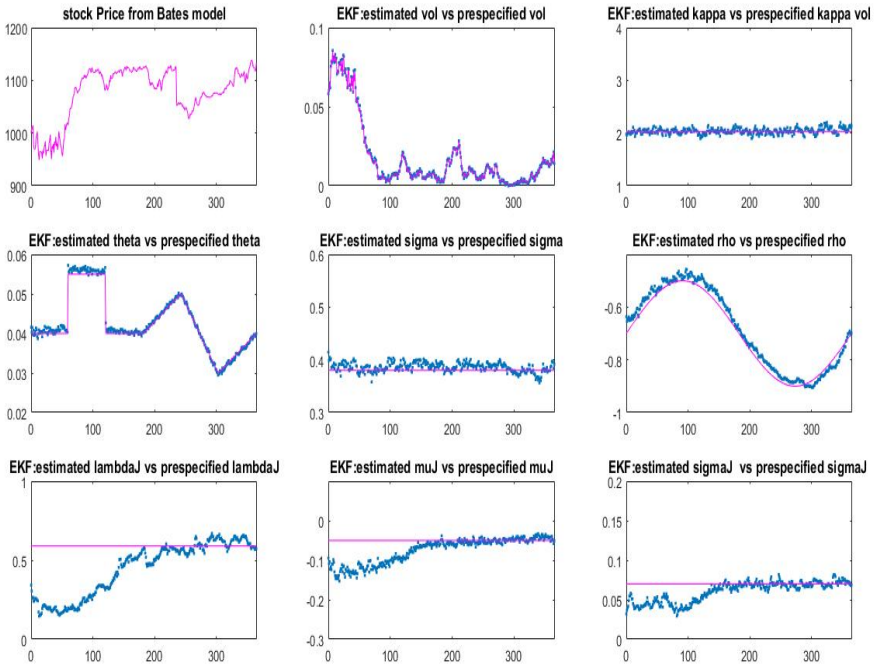

Fig. 5.2: Parameter estimation for the Bates model using EKF. The graphs above show parameter estimates for the Bates model for a period of 365 days. The pre-specified values are from Table 5.1

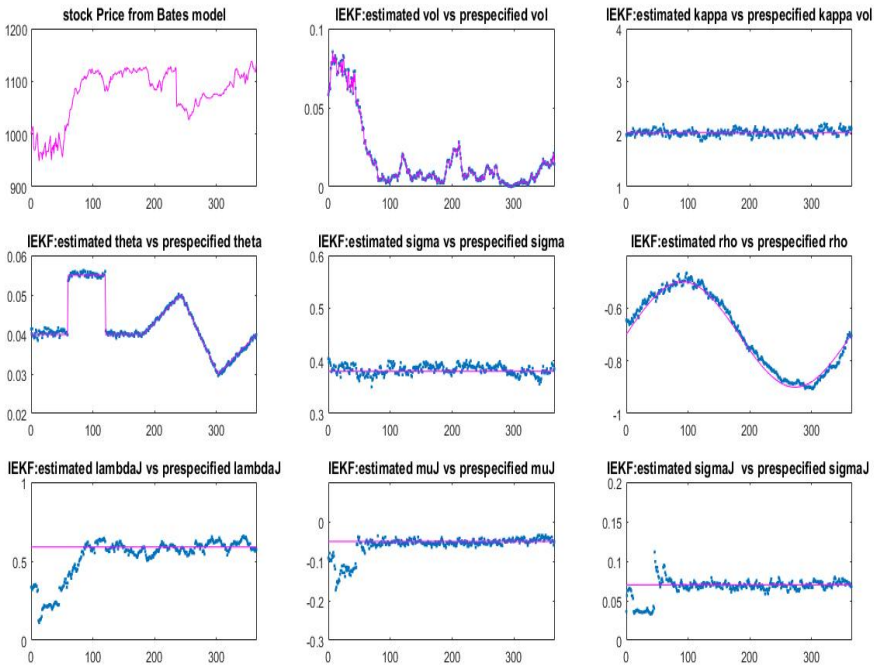

Fig. 5.3: Parameter estimation for the Bates model using IEKF. The graphs above show parameter estimates for the Bates model for a period of 365 days. The pre-specified values are from Table 5.1

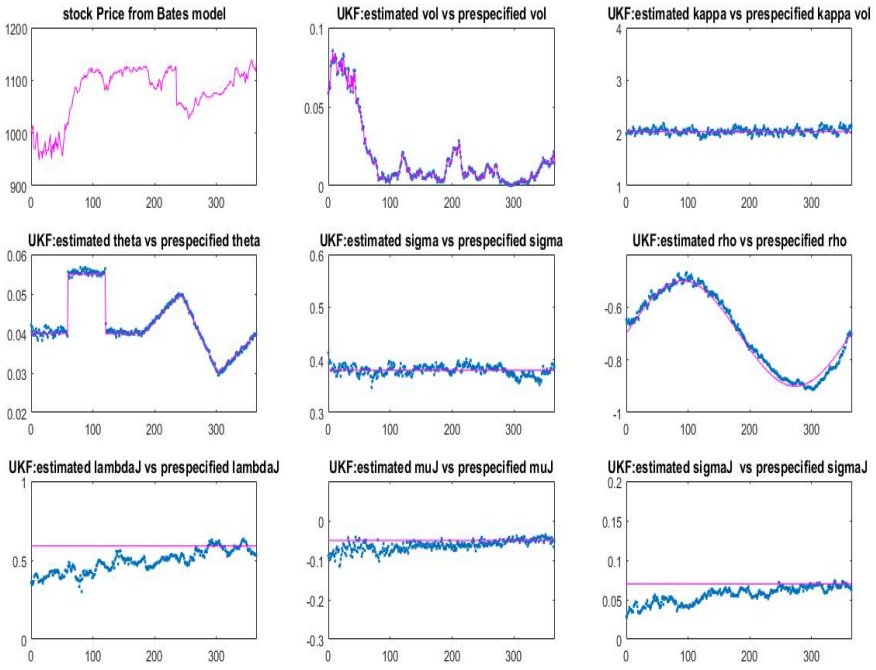

Fig. 5.4: Parameter estimation for the Bates model using UKF. The graphs above show parameter estimates for the Bates model for a period of 365 days. The pre-specified values are from Table 5.1

The LSQ method had the largest variation in the parameter estimates as it can be observed from the graph unlike earlier on when we had to estimate few parameters. For the Bates model, there are 8 parameters and the LSQ algorithm has to search for parameters that minimise the residual error in multi-dimension grid. For the non-linear filtering methods, the initial parameter estimates were far off from the pre-specified parameters. However, with each iteration, the filters were able to recover the pre-specified parameters.All the four calibration methods were able to recover the latent state well as per the graphs above.

The following table is a thorough analysis of all the four methods. The data is made up of 40 options per day for 365 days. Every day, 20 options were use for calibration and the remaining 20 were reserved for validation. To make the procedure as bias free as possible, the options were selected randomly every day and the same options were used for all the four methods.

Method	AE Par	ARPE Est	ARPE Val	IS Est(%)	IS Val(%)	MAE Est	MAE Val	Time(s)
EKF	150.363	1.107	0.885	87.810	81.263	0.077	0.099	0.254
IEKF	113.531	1.098	0.815	89.640	87.020	0.067	0.091	3.543
LSQ	264.296	1.060	5.364	91.642	84.032	0.064	0.129	24.741
UKF	123.785	1.209	0.834	88.602	85.023	0.072	0.096	0.537

Tab. 5.2: Summary statistics: Bates calibration

Before we analyse the table, it is better to explain what each summary statistic mean.

Absolute error (AE Par): This is the absolute difference between the pre-specified parameters and the estimated parameters. Since the calibration is performed for 365 days, we had 365 absolute differences for each parameter. The figures appearing in the table are the grand total. The best parameter estimates should give us the lowest AE Par.

Absolute relative percentage error (ARPE Val,ARPE Val): Here, we calculate the absolute difference between model and simulated prices. We also divide the difference by the simulated option prices to get a relative measure. Since we divided the data into validation and estimation set, we calculate the statistic for both sets. Again, we expect a lower ARPE for good calibration methods.

In sample fit (IS Val,IS Est): This measure counts the number of model prices falling in the bid ask spread bracket. A high value of IS is a sign of good parameter estimates.

Mean absolute error (MAE): This summary statistic is similar to ARPE. The only difference is that MAE is a absolute measure. One of the disadvantages of ARPE is that lower prices contribute more to the sum. This can be understood clearly if we consider the fact that we get a high value if the divisor is small. Willmott and Matsuura (2005) strongly advocate for this measure against other measure like root mean square error. Now we are ready to make sense of the figures in the table.

The LSQ method had the largest absolute error (AE) for the parameters. The best calibration under this heading was IEKF. We also calculated the absolute relative percentage error for the European call options that were used for calibration. Since the data was split into estimation set and validation set, we were able to calculate statistics for the two sets. We expect to see the lowest values in the ARPE in the estimation data compared to validation set. Indeed the same is confirmed by the table above. In their simulation study, Lindström *et al.* (2008) find that the LSQ suffers from over-fitting which is normally notable from the lowest ARPE calculated from the estimation data. The results I obtained in the table above confirm this fact. The LSQ methods performs badly on validation and the results we obtained confirm this as LSQ method has the highest ARPE-val. This is attributed to over-fitting by the LSQ.

When it comes to speed, the non-linear filtering methods are much faster. This is because there is no optimisation involved while the LSQ methods suffers from a curse of dimensionality. For the Heston model, the LSQ has to search for the parameters that minimise the error in a 2^5 grid. This process considerably lead to the slow performance of LSQ especially when the initial value is far off from the true parameters.

We present a similar table from the Heston model calibration using the four methods. In the same spirit, we reach the same conclusion. Maybe one observation worth mentioning is that the calibration is much faster in the Heston model simulation study. This is because of fewer parameters to estimate.

Method	AE Par	ARPE Est	ARPE Val	IS Est(%)	IS Val(%)	MAE Est	MAE Val	Time(s)
EKF	185.724	1.122	1.3463	87.83	83.40	0.087	0.117	0.156
IEKF	131.965	0.967	1.007	89.08	87.23	0.072	0.106	0.646
LSQ	214.674	0.876	22.7	91.233	84.30	0.069	0.127	1.271
UKF	143.203	0.987	1.107	88.40	85.13	0.0755	0.112	0.162

Tab. 5.3: Summary statistics: Heston calibration

5.4 A hedging study

Below are the histograms showing the performance of the four calibration methods on hedging a call option under the Bates model. The option has maturity of a half a year. We simulated 200 stock paths. For each stock paths, we estimated the parameters and hedging was done using those paths specific parameter estimates. The Profit and loss (PnL) was calculated as a way appraising the four methods.

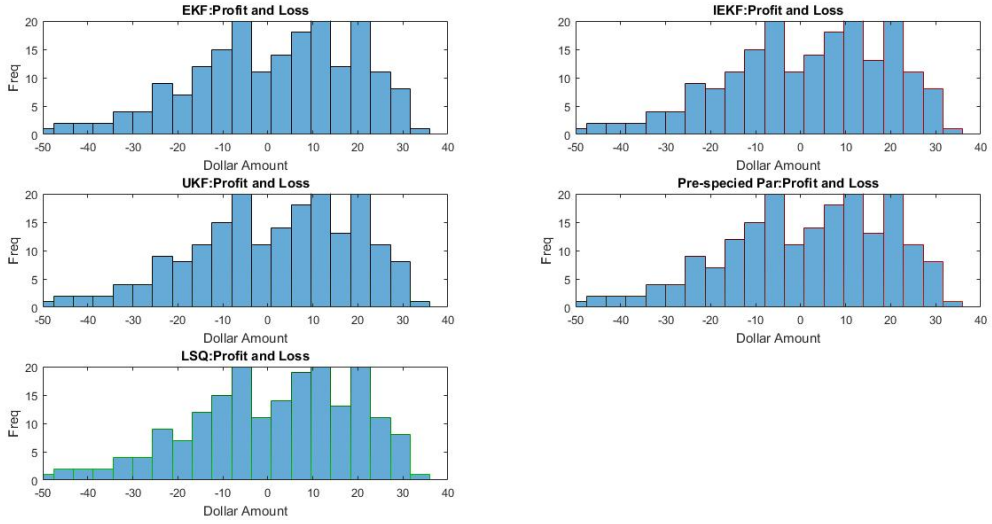

Fig. 5.5: Histograms for a simulation study

The five graphs above have a general shape. This is supported by summary statistics table below. I have also included a hedging study using the true parameter values as a benchmark. The results are similar to the parameter estimates obtained from the four calibration methods.

Method	Mean (PnL)	Std(PnL)	Sum of Delta
EKF	0.6403	19.5655	789.1582
IEKF	0.6392	19.5641	788.9413
UKF	0.6396	19.5678	788.9022
LSQ	0.6396	19.5659	789.3787
Pre-specified	0.080	19.5637	788.9157

Tab. 5.4: Analysis of a hedging study

The mean profit and loss for the 4 cases is almost the same except for the pre-specified parameters. However, the standard deviation is the same for all the 5

cases. We also calculated absolute changes in the delta to gauge the extent of portfolio re-balancing. Portfolio re-balancing can be used as a proxy for transaction cost but in this analysis, the sum of absolute changes in the delta is almost the same.

The statistics from the above table do not show any measure significant difference between the 4 sets of parameter estimates. Delta hedging stochastic volatility models using only two assets is not sufficient. This is mainly because under these models, the market is incomplete and there are many equivalent measures. To eliminate risk, one can use carefully selected options like in the study by He *et al.* (2005). They show that ten options in addition to the underlying and risk free asset can greatly decrease the risk to almost zero. But one has to note that in the real world, there are some constraints to the number of option one can trade. Transaction costs have to be factored in.

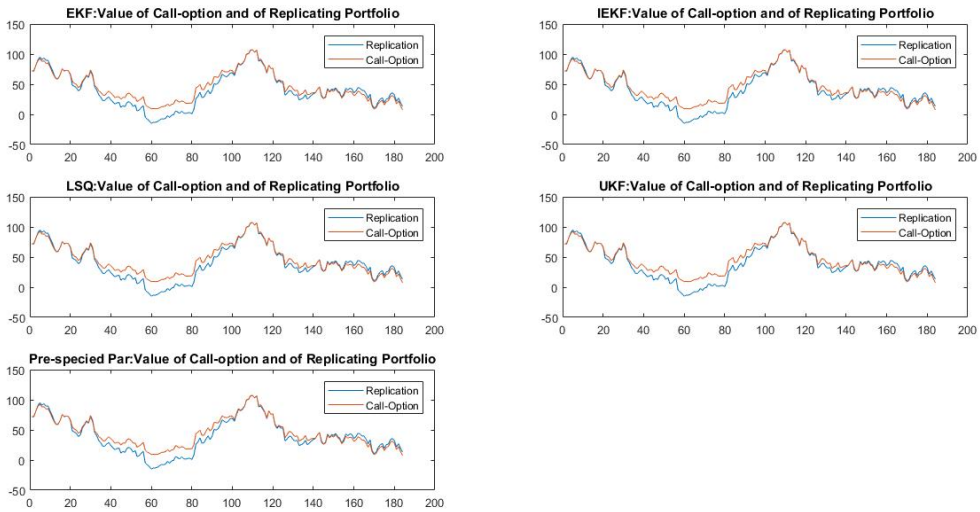

Fig. 5.6: The value of a call option and replicating portfolio

The graphs above show the value of replicating portfolio and a call option being hedged from the first day until maturity. Like the summary statistics already shown above, the path followed is similar. One has to mention that other paths will produce a wider difference because of jumps in the stock price. This is supported by the high variance in the summary statistics table.

Chapter 6

Conclusion

Stochastic volatility models are a new area of research in option pricing. These models treat volatility as a stochastic process and there are many reasons for modelling volatility. For governments around the world, they have to come up with policies to absorb shocks in the market. For traders, they have to forecast the sensitivity of their positions from random market movement. For the fund manager, volatility should be considered in their investment policy statements especially for investment of short horizons. We implemented four calibration methods and analysed their performance on estimating parameters of stochastic volatility models. These models are not easy to calibrate as they have a latent state (volatility) that has to be inferred from the quoted option prices.

Our simulation study results show that the non-linear filtering methods are computationally efficient compared to the standard least square method. Moreover, they are able to estimate the latent state and time-varying parameters better than the standard LSQ method. One method which performed particularly well on all metrics is the IEKF. It had the lowest errors (ARPE and MAE). Because of the iteration loop, IEKF lagged behind on computational efficiency even though it still outperformed the standard least square methods.

A simulation study using delta hedging does not show any significant benefit resulting from performing delta hedging using parameter estimates obtained from non-linear filtering methods as compared to least square parameter estimates.

Our results show that the non-linear filtering methods provide robust parameter estimates and are computationally efficient. The techniques also make optimal use of the data unlike the standard least square which only uses today's data for prediction. Thus, we suggest that standard least square method should possibly be replaced by non-linear filtering methods as a new way of calibrating asset pricing models to option prices. However, filtering gives the real-world parameters, whereas for pricing, the risk-neutral parameters are needed . It is therefore neces-

sary to also specify and estimate a market price of risk. But this is for future work.

Bibliography

Albrecher, H., Mayer, P., Schoutens, W. and Tistaert, J. (2007). The little Heston trap, *Journal of Financial Economics* **7**(3): 83–92.

Andersen, L. (2007). Efficient simulation of the Heston stochastic volatility model. **URL:** *&+#http://www.ressources-actuarielles.net/ext/FILE/LeifAndersenHeston.pdf*

Bakshi, G., Cao, C. and C, Z. (1999). Empirical performance of alternative option pricing models, *Journal of Finance* **52**(5): 2003–2049.

Bakshi, G. and Madan, D. (2000). Spanning and derivative security valuation, *Journal of Financial Economics* **55**(2): 205–238.

Bates, D. S. (1996). Jumps and stochastic volatility: Exchange rate processes implicit in Deutsche mark options, *The Review of Financial Studies* **9**: 67–107.

Black, F. and Scholes, M. S. (1973). The pricing of options and corporate liabilities, *Journal of Political Economy* **81**(3): 229–263.

Carr, P. and Madan, D. B. (1999). Option valuation using the fast Fourier transform, *Journal of Computational Finance,* **2**(4): 61–73.

Chourdakis, K. (1999). Option pricing using the fractional FFT, *Journal of Computational Finance* **8**(2): 1–18.

Cont, R. and Tankov, P. (2002). Calibration of jump-diffusion option-pricing models:a robust non-parametric approach, *Rapport Interne CMAP* **490**: 72–82.

Cont, R. and Tankov, P. (2004). *Financial Modelling with Jump Processes*, Chapman and Hall/ICRC.

Fang, F. and Oosterlee, C. W. (2008). A novel pricing method for european options based on Fourier-cosine series expansions, *SIAM Journal of Scientific Computing* **31**(2): 826–848.

Feller, W. (1951). Two singular diffusion problems, *Annals of Mathematics* **54**(1): 173–182.

Gagliardini, P., Gourierouxy, C. and Renaultz, E. (2011). Efficient derivative pricing by the extended method of moments, *Journal of Economic Society* **79**(4): 11811232.

Gatheral, J. and Lynch, M. (2002). Lecture 1: Stochastic volatility and local volatility. **URL:** *http://www.math.ku.dk/ rolf/teaching/ctff03/Gatheral.1.pdf*

Gilli, M. and Schumann, E. (2010). Calibrating option pricing models with heuristics, *Uppsala University* .
URL: *https://link.springer.com/chapter/10.1007/978-3-642-23336-4*$_2$

Guo, J. (2013). Simultenous calibration and hedging of options.
URL: *http://portal.research.lu.se/portal/en/publications/simultaneous-calibration-and-quadratic-hedging-of-options(e87d3df9-7d9d-45ee-9936-8e095d731bed).html*

He, C., Kennedy, J., Coleman, T., Li, Y. and Vetzal, K. (2005). Calibration and hedging under jump diffusion, *Review of Derivatives Research*, **9**(1): 1–35.

Heston, S. L. (1993). A closed-form solution for options with stochastic volatility with applications to bond and currency options, *The Review of Financial Studies* **6**(2): 327–343.

Hirsa, A. (2012). *Computational Methods In Finance*, CRC Press.

Jessen, C. and Poulsen, R. (2013). Empirical performance of models for barrier option valuation, *Quantitative Finance* **13**(1): 1–11.

Julier, S. J. and Uhlmann, J. K. (1996). A general method for approximating nonlinear transformations of probability distributions. tech rep,rrg, department of engineering science, university of oxford.
URL: *https://pdfs.semanticscholar.org/523a/865ffabb50d10f85d141963d40528e952760.pdf*

Kalman, R. (1960). A new approach in linear filtering and prediction problems, *Rapport Interne CMAP* .

Kovachev, Y. (2014). Calibration of stochastic volatility models.
URL: *https://uu.diva-portal.org/smash/get/diva2:729886/FULLTEXT01.pdf*

Lee, C.-F., Lee, A. C. and Lee, J. (2010). *Handbook of Quantitative Finance and Risk Management*, New York : Springer.

Lindström, E., Madsen, H. and Nielsen, J. N. (2015). *Statistics for Finance*, Chapman and Hall/CRC.

Lindström, E., Ströjby, J., Brodèn, M., Wiktorsson, M. and Holst, J. (2008). Sequential calibration of options, *Computational Statistics and Data Analysis* **52**(6): 2877–2891.

Lord, R., Fang, F., Bervoets, F. and Oosterlee, C. W. (2008). A fast and accurate FFT-based method for pricing early-exercise options under Lévy processes, *SIAM Journal of Scientific Computing* **30**: 229–263.

Schoutens, W., Simons, E. and Tistaertz, J. (2003). A perfect calibration ! Now what ?, *Wilmott magazine* **2**: 66–78.

Scott, L. O. (1997). Pricing stock options in a jump-diffusion model with stochastic volatility and interest rates: Application of Fourier inversion methods, *Mathematical Finance* **7**(4): 413–426.

Strutz, T. (2011). *Data Fitting and Uncertainty*, Vieweg+Teubner Verlag.

Wan, E. A. W. and van der Merwe, R. (2001). The unscented Kalman filter for nonlinear estimation, *Journal of Financial Economics* pp. 153–158.

Willmott, C. J. and Matsuura, K. (2005). Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance, *Center for Climatic Research* **30**: 72–82.

Appendix A

Appendix: Heston parameter estimation

A.1 EKF Heston model parameter estimates

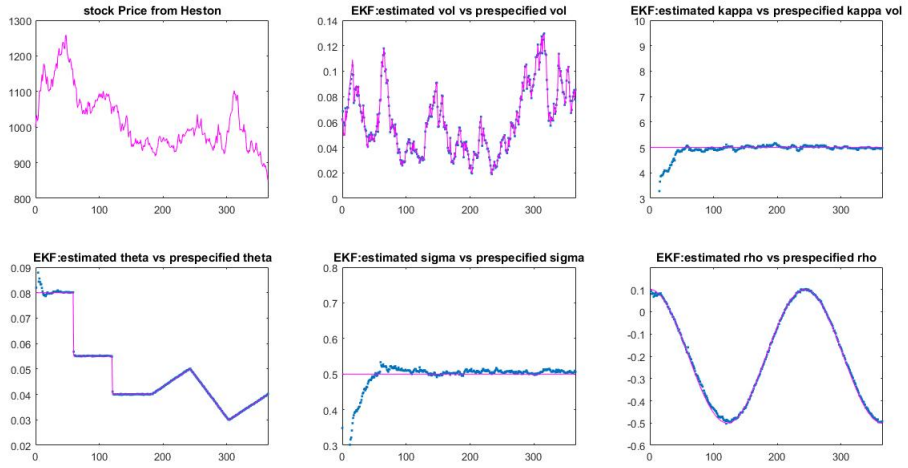

Fig. A.1: Parameter estimation for the Heston model using EKF

A.2 UKF Heston model parameter estimates

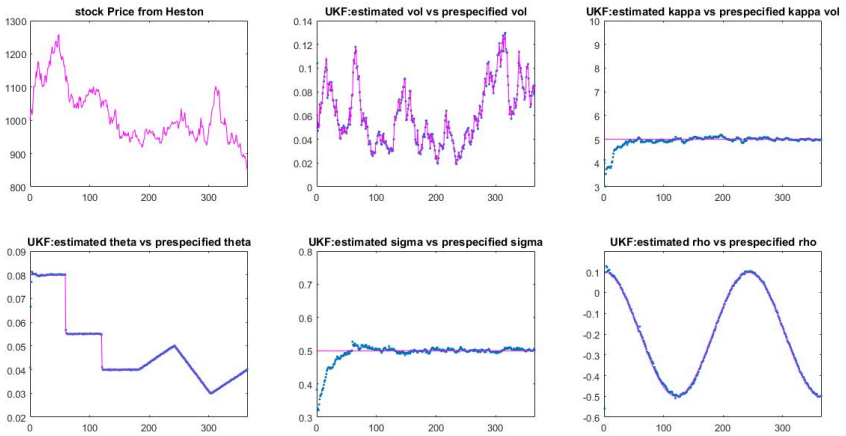

Fig. A.2: Parameter estimation for the Heston model using UKF

A.3 IEKF Heston model parameter estimates

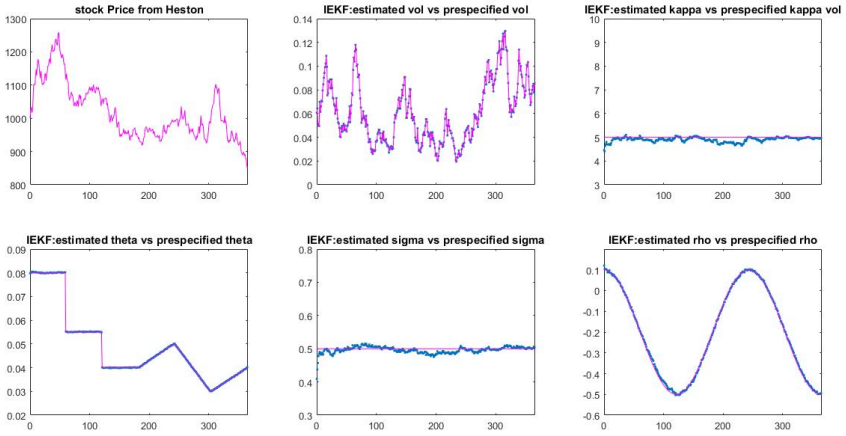

Fig. A.3: Parameter estimation for the Heston model using IEKF

A.4 LSQ Heston model parameter estimates

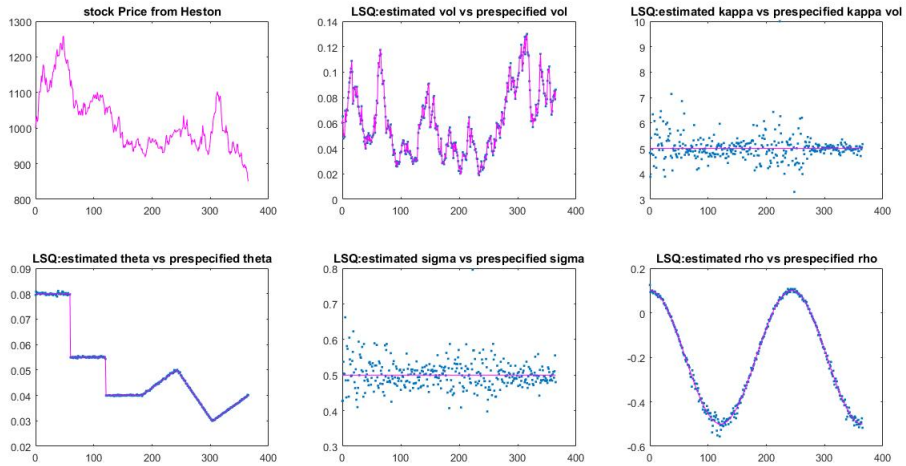

Fig. A.4: Parameter estimation for the Heston model using LSQ