

上海交通大学

SHANGHAI JIAO TONG UNIVERSITY

学士学位论文

THESIS OF BACHELOR



论文题目：THE APPLICATION OF STATISTICAL LEARNING
IN FINANCIAL DATA FORECASTS

学生姓名：袁航

学生学号：5111209048

专 业：金融学（试点班）

指导教师：罗俊

学院(系)：安泰经济管理学院

THE APPLICATION OF STATISTICAL LEARNING IN FINANCIAL DATA FORECASTS

---IMPLICATIONS IN DERIVATIVE PRICING AND HEDGING

ABSTRACT

I propose a nonparametric approach to derivative pricing and dynamic hedging. I extend earlier study by Hutchinson *et al.* with Geometric Brownian Motion assumption to a broader scenario of Stochastic Volatility Model. Data are simulated according to the pricing process, and option contracts are assigned exactly following CBOE rules. I make a comparison among multiple-layer perceptron, radial basis function and parametric methods. Two metrics are proposed for comparison. The first is a static measure of out-sample prediction error. The second is a dynamic measure of dynamic hedging error. For the latter one, I introduce a well-designed minimum-variance dynamic hedging analysis of SVM with theoretical background of parameter estimation. Parameter estimation is run through an optimization scenario, and I offer a refined version of instantaneous volatility estimation which can further improve the optimizing result. Dynamic hedging results of parametric and nonparametric methods are provided and discussed within each moneyness-maturity category. The thesis ends with an empirical analysis of data from Chinese option market.

Key Words: nonparametric method, multiple-layer perceptron, dynamic hedging, stochastic volatility model

摘要

我参考了一种用于衍生品定价及动态对冲的非参数方法，对 Hutchinson 等人基于几何布朗运动假设下的该方法进行了推广，在随机波动率模型的框架下进行了推导和试验。我对多层感知器、径向基网络等方法与参数方法进行比较。我使用两种指标来衡量模型的准确度。其一是静态指标，即指样本外的预测误差。其二是动态指标，即动态对冲误差，相比前者复杂度更高，但也有更多经济含义。我参考了随机波动率模型下的参数估计方法来辅助动态对冲。特别地，我使用了一种针对随机波动率模型修正的瞬时波动率估计方法，提高了精度。我对于不同到期日-执行价的组合进行了结果的分析。最后，我利用上述方法对中国期权市场进行实证研究。

关键词: 非参数方法, 多层感知器, 动态对冲, 随机波动率模型

CONTENT

1	Introduction.....	1
2	Methodology.....	3
	2.1 Projection Pursuit Regression and Neural Network	3
	2.2 Issues in Training and Prediction	5
3	Data Generation and In-sample Fitting Result.....	7
	3.1 Simulation and Pricing under Geometric Brownian Motion Scheme	7
	3.1.1 Data Generation	7
	3.1.2 Option Contracts Generation Rule	7
	3.1.3 Training and Performance Evaluation	8
	3.2 Simulation and Pricing under Stochastic Volatility Scheme	12
	3.2.1 Introduction	12
	3.1.2 Option Contracts Generation Rule	14
4	Static and Dynamic Measures of Model Performance.....	18
	4.1 Introduction: Discrete Time Dynamic Hedging	19
	4.2 Dynamic Hedging Strategy Formulation and Simulation: Constant Volatility	19
	4.3 Parameter Fitting with Stochastic Volatility Sequence.....	21
	4.3.1 Parameter Estimation	21
	4.3.2 Optimization Problem Formulation	23
	4.3.3 Optimizing Scenario.....	24
	4.4 Out-sample Pricing Approximation: Static Approach	27
	4.5 Dynamic Hedging Strategy Formulation and Simulation: Stochastic Volatility	33
	4.5.1 Minimum-Variance Hedging	34
	4.5.2 Delta-neutral Hedging	35
	4.6 Results.....	36
	4.6.1 Results from Constant Volatility Model	36
	4.6.2 Results from Stochastic Volatility Model	37
5	Empirical Results.....	41
	5.1 Data Description.....	41
	5.2 Out-sample Pricing Approximation: Static Approach	41
	5.3 Out-sample Pricing Approximation: Static Approach	42
6	Conclusions	43
	Reference.....	44
	Appendix.....	44

1 Introduction

The pioneering paper by Black, Scholes and Merton (1973) has greatly contributed to the prosperity of option markets. In the paper, they derived the closed-end option pricing formula through a dynamic hedging approach under no-arbitrage assumption. However, it has indicated some biases with empirical data, where it shows biases across moneyness and maturity, known as the “smile”. The evidence is consistent with the empirical distribution of stock returns which is negatively skewed with high kurtosis than log-normal distribution within the assumption of BS model. Therefore, BS model has been widely generalized and applied in the years to come. The new model serves to release some conditions in Black-Scholes formula. All these models vary with the assumption on asset price. This includes the stochastic interest rate model which assumes the stochastic interest-rate mean-reversion spread; the jump-diffusion model which allows for jumps; the stochastic volatility model which further account for the variation in variance; and the stochastic volatility and stochastic interest model which accounts for both stochastic natures in interest and volatility. There could actually be infinitely many of such models with each subtle assumption subject to change on the nature of asset price and related features.

In the empirical option pricing and hedging, however, such a variety of models could possibly lead to misspecification. A model as simple as Black-Scholes is likely to stop short of exploring all features and require too stringent conditions. However, for a model as complicated as SVSI (stochastic volatility stochastic interest) it can be hard to specify all the requiring parameters and run the risk of overfitting in-sample data and deteriorating out-sample performance. The success of an option pricing model heavily depends on the identification of underlying asset’s pricing process. Therefore, the selection and evaluation of option pricing models have always been of intriguing interest to academia and practitioners.

As a substitution to the traditional approach, which I will denote as parametric approach, Hutchinson *et al.*[1] (1994) proposed a nonparametric approach to pricing and hedging derivatives. They took the primary variables that affect derivative prices as inputs, and labeled the option prices as single output, then trained learning nets with simulated datasets. The intension is to learn the pricing formula through a learning net, then utilize the net to pricing and hedging. They applied the nonparametric method to the pricing and hedging of S&P 500 Futures Options, and witnessed an improvement from the traditional Black-Scholes approach.

This data-driven method has several advantages over traditional parametric pricing models. First, the restrictions of lognormality or sample-path continuity can be removed for the data-driven method. Such assumption is vital to parametric pricing models. Second, they are adaptive to the structural changes in the data-generating process. We can always use close-to-date data to train network to keep the model informative of present data. Also, the method is applicable to a wide range of asset price dynamics without the need for specification of these dynamics and its corresponding parameter estimation. For a specific example, Black-Scholes pricing is based on a diffusion process with constant volatility, which has been rejected by empirical studies. Pricing scheme can be generalized under a stochastic volatility scheme, but it may suffer from over-fitting

of historical data when estimating parameters and therefore lack a good indication of future data. The flexibility of a data-driven method makes it preferable in real-world pricing and hedging. On the other hand, as a shortcoming of the method, the data-driven method shall require a large amount of data available, therefore hard to apply to thinly-traded derivatives. The prediction accuracy of this method may, to a great extent, depend on the designing of a problem-specific model (the topological structure of a multiple-layer perceptron for instance), and the delicate selection of model can be a demanding task. Also it will be dominated by analytical formulae if the underlying asset process is well understood and parameter can be estimated soundly.

My study will extend that of Hutchinson's, which basically focused on the nonparametric pricing when underlying assets follows a Geometric Brownian Motion. I further study the case where underlying asset process is stochastic volatility, where the training and hedging scenario are quite different.

The central issue in the prediction of stochastic volatility model lies in parameter estimation. Different from Black-Scholes model where only volatility is required to be estimated, there is a sequence of parameters for estimation in stochastic volatility model. An optimization approach will be applied here. This can be done both locally and globally. Local approach requires optimization on each trading day's data. Global one can be run with a wider range of data, however it also requires the pre-estimation of volatility. I offer two approaches for estimating instantaneous volatility. The first one applies implied volatility from at-the-money close-to-maturity options. The second one is a refined estimation of local volatility with implied volatility. It views implied volatility as the integration of instantaneous volatility over the life of the option. It takes the inverse of this process, and extracts instantaneous volatility from implied volatility. I will show that this trick can improve the estimation of local volatility, and finally improve the result of optimization.

With the knowledge about parametric estimation, I give two metrics to measure out-sample performance. The first is a static one. I just use mean squared error or mean absolute error here. It is a quite straightforward one. I'll subtly adjust nonparametric methods for making out-sample prediction, and present a comparison among different methods. The second one is a dynamic one built on dynamic hedging strategy. It is more implicit and complicated than the static measure, but is more meaningful economically and practically. The hedging scenario is simpler for Black-Scholes model. Delta shares of underlying asset are implemented to hedge against the option price's exposure to the risk of underlying asset's price. It could be more subtle for stochastic volatility, since the option price has two risk exposures: one to asset price and the other to asset volatility. I offer two options about the hedging here. The first one is the generally-known minimum variance hedging, in which only underlying asset is applied to minimize the variance of replicating portfolio. This hedging cannot perfectly hedge since the volatility cannot be hedged with merely underlying asset. The second one is delta-neutral hedging which implements multiple assets to hedge against all exposures to risk. To hedge against volatility, one additional option with different strike or maturity will be applied.

The thesis is outlined as following. In Chapter 2, I will introduce the statistical learning

methodologies heavily implemented in latter sections. I will give brief introduction of theoretical background and tips for applications. In Chapter 3, I describe the process of data generation and provide some in-sample fitting result. I start by generating simulated datasets and option contracts. In the first exploration, the price sequences are generated according to Geometric Brownian Motion, and option contracts are assigned to the sequence according to CBOE rules with some simplifications. Parametric and nonparametric methods will be applied to the sequence of data. The in-sample performance will be justified. Specially, I will give some details about the training and specification of radial basis function model and multiple layer perceptron. Then I further extend this approach to where underlying asset pricing process is stochastic volatility. I will provide a brief overview of the nature and pricing of stochastic volatility model, and proceed to the pricing with parametric and nonparametric methods with perfect understanding of model parameters. In Chapter 4, I present the out-of-sample performance of parametric and nonparametric methods with respect to both GBM and SVM. I begin by providing details about parameter estimation for stochastic volatility model, which is central to derive both metrics. Two measures are offered to evaluate performance. The first is a static metric, for which I use out-of-sample prediction error. The second is a dynamic metric, which is denoted as the tracking error of dynamic hedging strategy over option's life. I will formulate the dynamic hedging problem in detail. For SVM, I give to possible hedging strategies. They are minimum-variance hedging and delta-neutral hedging. In the final section, I will provide an analysis of results. In Chapter 5, I will provide empirical results on Chinese option market with the pre-analyzed methods. Finally in Chapter 6, conclusions are made and the expectation of further studies is discussed.

2 Methodology

I will introduce the main statistical learning methods applied in this thesis. Two methods are heavily implemented. The first is radial basis function, and the second is multiple-layer perceptron. Both can be viewed as a specification of the renowned neural network algorithm, which is one of the most prevalent nonparametric statistical learning algorithm. In the following context, I will briefly introduce the theoretical background and application details of the two methodologies.

2.1 Projection Pursuit Regression and Neural Network

Multiple-layer perceptron is the most typical type of neural network model. The central idea of neural network is to form linear combinations of inputs as a feature, and then map the feature to the target with a nonlinear function. I first introduce the idea of projection pursuit regression, which is highly correlated to the modeling of neural network.

The projection pursuit regression takes the following form:

$$f(X) = \sum_{i=1}^M g_m(\omega_m^T X)$$

The model is additive on the nonlinear maps of linear combinations of inputs. The model is very general in that it can represent very complex combinations of inputs. For example, we can write the product $X_1 X_2$ as $\frac{1}{4} \left[(X_1 + X_2)^2 - (X_1 - X_2)^2 \right]$. Such generalized feature gives it very strong predictive power. Meanwhile, it could be hard to form a good interpretation of the model.

The fitting of a PPR model given training dataset $(x_i, y_i), i = 1, \dots, N$ is formulated as minimization of summed squared error:

$$\min_{g_m(\cdot), \omega_m} \sum_{i=1}^N \left[y_i - \sum_{m=1}^M g_m(\omega_m^T x_i) \right]^2.$$

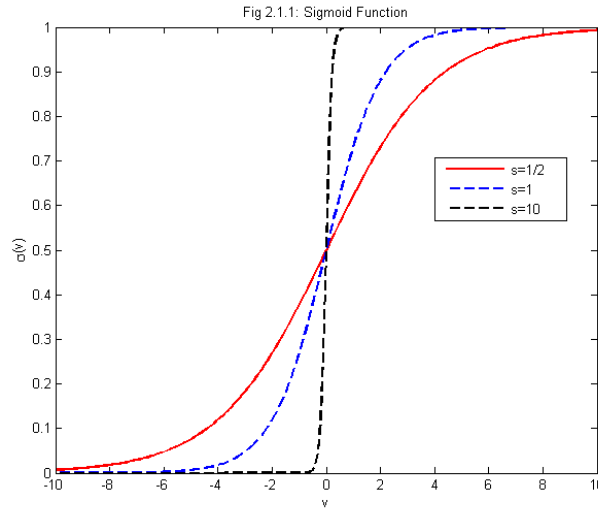
The optimization has two steps. In step one, we minimize over ω given g . In step two, we minimize over g given ω . The two steps are iterated until convergence. More details about this estimation is available in T.Hastie *et al.*[2](2009).

Neural network inherits many characteristic of PPR. I begin by introducing “Vanilla” Neural Net, sometimes named single hidden layer backward-propagation network. It is actually a two-stage model for regression or classification. Suppose we have a training dataset with input vectors X of length M , and output vectors Y of length K . Then the target can be modeled as a function of linear combinations of intermediate input denoted as hidden layer, which is again a function of linear combinations of initial inputs. A mathematical formulation is given as:

$$\begin{aligned} Z_m &= \sigma(\alpha_{0m} + \alpha_m^T X), m = 1, 2, \dots, M \\ T_k &= \beta_{0k} + \beta_k^T Z, k = 1, 2, \dots, K \\ f_k(X) &= g_k(T), k = 1, 2, \dots, K \end{aligned}$$

The formulation can be vividly visualized as a network diagram, which accounts for the name “neural network”. The conventional selection of activation function is sigmoid function

$\sigma(v) = \frac{1}{1 + e^{-v}}$. To explore the nature of this function, I plot $\sigma(sv)$ for different values of s , illustrated in **Fig 2.1.1**.



The function flattens as input goes to extreme values, and has greater slope for small values of input. The scaling effect of input will approximate the function to a Heaviside function. There are plenty of other choices for activation function. When Gaussian radial basis function is applied, the corresponding model is called *radial basis function network*. This is also one type of kernel

smoother with kernel function $K_{\lambda}(\xi, x) = D\left(\frac{\|x - \xi\|}{\lambda}\right)$ used as basis function. Here function D

denotes the Gaussian density function. And g_k are usually set to be identity function for regression problems. *Multiple-layer perceptron*, as its literal meaning suggests, represents neural network with multiple hidden layers.

We can easily find out that one-hidden-layer neural network model shares the same form as projection pursuit function model. The difference lies in the selection of activation function, where PPR uses nonparametric functions while neural network model implements a function simpler in form. This makes vanilla neural net easier to fit than PPR, and can actually contain more terms.

The fitting of neural network parameters α and β is again a minimization over summed squared error. The approach of optimization is by gradient descent, and it is called back-propagation. It is performed by a forward and backward transition of errors over the network. I will not refer to the detailed procedure here. However, we don't require global minimizer here since this could possibly overfit out-sample data. Therefore, we can control the optimization by adding a penalized term or implementing early stop.

2.2 Classic Issues in Training and Prediction

There are many details to cover in the training of neural network due to its flexibility. I will discuss about some of the most important issues.

1) Starting Value

From the property of sigmoid function we know that the operative part is approximately linear if weights are close to zero. Usually the starting values of weights are assigned to random values near zero. Therefore the model begins as a linear one, and gains nonlinear feature as it is trained with larger weights.

2) Overfitting

The global minimization of SSE in weight estimation will overfit the data. We can design an early-stop method to stop before global minimum is reached, therefore avoiding the overfitting issue. Another method is called weight decay, which adds a penalty term to the error function.

3) Scaling of Inputs

The standardization of inputs can ensure all inputs to be treated equally in the regularization.

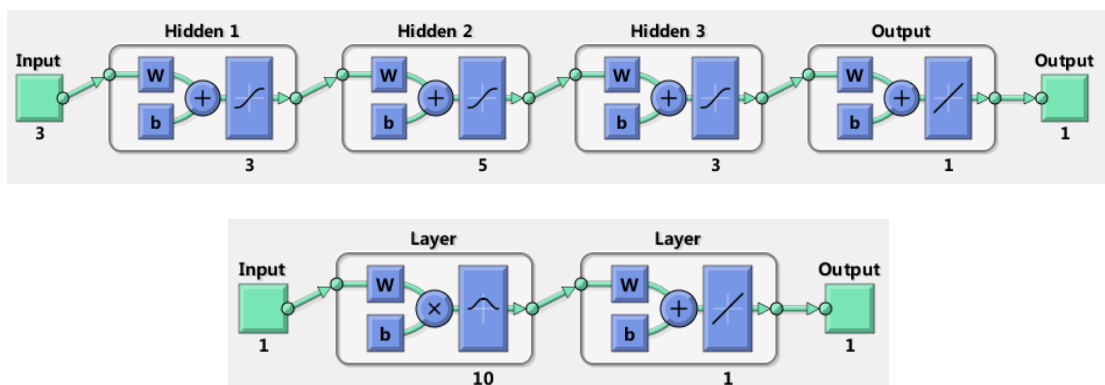
4) Number of hidden units and layers

In general, more hidden units can not harm the model's performance. Typically, the number of hidden units increases with the number of inputs as well as the size of training data. A good approach is to use a large number of hidden layers and then train them with regularization. The regularization parameter can be chosen based on a cross-validation set.

5) Multiple Minima

In most cases, the error function is nonconve with many local minima. In these cases, we may end up with local optima. A good approach is to use the average predictions over a series of trained network as the final prediction.

To end up this section, I briefly introduce MatLab Neural Network Toolbox, which will be of great help to my pursuing studies. MatLab(R2012b) has a well established neural network toolbox. It can achieve the training and prediction of a wide variety of neural network models, including multiple-layer perceptron and radial basis function network. I give an illustration of typical MLP and RBF configuration with this toolbox. The upper one is the configuration of a neural network with three hidden layers. The lower one is a typical radial basis function with 10 hidden units. One may refer to the user's guide for detailed operations of this toolbox. All implementation of neural network models in the following context will be supported by this toolbox.



3 Data Generation and In-sample Fitting Result

3.1 Simulation and Pricing under Geometric Brownian Motion Scheme

In this chapter, I shall generate the price sequences according to different assumptions of underlying processes. I will add option contracts to the price sequences according to CBOE (Chicago Board Options Exchange) rules. Finally, different learning net methods will trained on the datasets.

3.1.1 Data Generation

Black-Scholes formula lays on the basic assumption of GBM, specified as following:

$$\frac{dS(t)}{S(t)} = \mu dt + \sigma dW(t) \quad (2.1.1)$$

where: volatility σ , return μ , expiration T .

Given $S(0)$ and a separation defined by $\{t_i | 0 = t_0 < t_1 < \dots < t_n = T\}$, the simulation goes as following:

$$S(t_{i+1}) = S(t_i) \exp\left[\left(\mu - \frac{1}{2}\sigma^2\right)(t_{i+1} - t_i) + \sigma\sqrt{t_{i+1} - t_i}Z_{i+1}\right] \quad i = 0, 1, \dots, n-1 \quad (2.1.2)$$

For the pricing procedure, I refer to the Black-Scholes formula:

$$\begin{aligned} \hat{C}_{BS} &= f_{BS}(S(t), T-t, K) \\ &= S(t)\Psi(d_1) - Ke^{-r(T-t)}\Psi(d_2) \end{aligned} \quad (2.1.3)$$

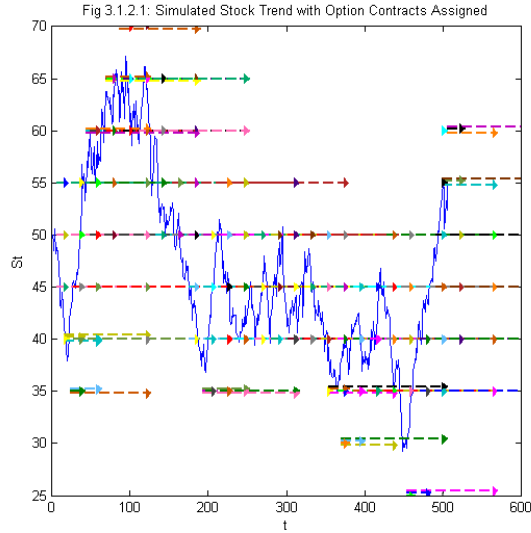
$$\text{where: } d_1 = \frac{\ln\left(\frac{S(t)}{K}\right) + \left(r + \frac{1}{2}\sigma^2\right)(T-t)}{\sigma(T-t)}, d_2 = d_1 - \sigma(T-t)$$

3.1.2 Option Contracts Generation Rule

Given a simulated series of stock prices, corresponding option paths are introduced according to CBOE rules. I summarize it as following:

- At any time, options with certain strike price on a particular stock have four expiration dates: the current month, the following month, the following two expirations on a quarterly scale.
- When current options expire, new options will be introduced with strike price at multiples of 5 around current stock price. When option price is close to one of the two strike prices, a third strike price will be attached to the options.
- If stock price moves out of the current strike price range, a new strike price will be added to the current sets of options.

Sample paths of stock price and the corresponding option contracts are illustrated in **Fig 3.1.2.1**. The figure displays a typical process of asset price (blue curve). The colorful arrows are attached to this curve, pointing from the beginning to the ending of an option contract in its x-value, remaining constant with the value of strike price on y-axis. Parameters are set to be: $S(0) = 50, \mu = 0.10, \sigma = 0.20, T = 2, N = 506$. The mean and standard deviation are set to be typical value of a frequently traded stock, and trading days are set to be 253 days per year, which is also consistent to reality.



3.1.3 Training and Performance Evaluation

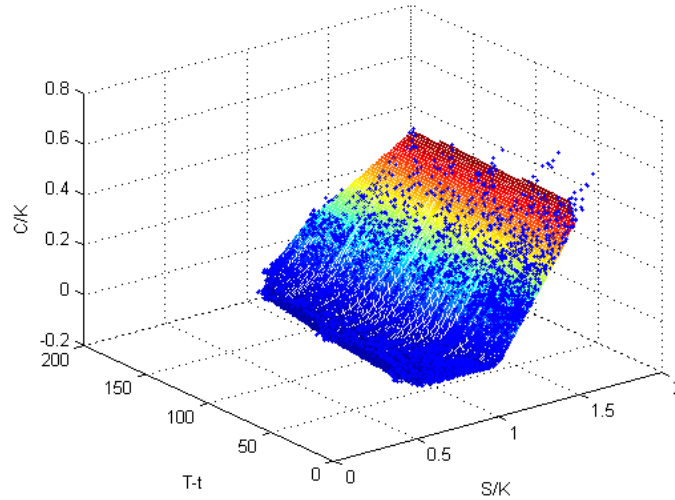
Now I shall apply statistical learning methods to simulate the pricing procedure and apply them to independent sample paths. More explicitly, given a training dataset $D = \{(\mathbf{X}_i, \hat{C}_i)\}_{i=1}^m$, we want to

define $f: \mathcal{d} \rightarrow \mathcal{+}$ that minimize $\text{MSE} = \frac{1}{|J|} \sum_{j \in J} (f(\mathbf{X}_j) - \hat{C}_j)^2$ on a cross-validation set.

Multiple linear regression, radial basis function network and multiple-layer neural network can be applied here. Several indicators can be used to evaluate the performance, including out of sample R-squared and dynamic hedging test error.

We begin with the case where underlying process is Geometric Brownian Motion. First, I generate 5 sample paths and attach option contracts according to previous procedures. The observations will be priced according to pricing formulae. Then I will randomly sample to form the training dataset. **Fig 3.1.3.1** is a 3-d plot of our simulated data and how they fit the Black-Scholes surface. The figure displays scatter plot of our generated samples against a fitting Black-Scholes surface. The surface shows the property of B-S pricing formula: it is monotone and concave. And we can see from the scatter plot that more observations are generated approximately at-the-money.

Fig 3.1.2.2: Scatterplot of Simulated Prices



Notice that I decrease the dimensions of input data from 3 to 2. This is doable by transforming (2.1.3) to $\hat{C}_{BS} / K = f_{BS}(S(t) / K, T-t, 1)$, since C is of linear relationship with S.

Then I generate the training set and cross validation set. For the training set, I randomly generate 5 sample paths, and randomly select 2000 observations from each sample path. Selected observations together form a training set of 10000 observations. Then another 5 sample paths are generated, and 1000 observations are selected from each of them. This shall form the cross validation set with 5000 observations. Both sets will be applied throughout the training and analysis of three methods. See **Table 2.2.1** for a detailed description of regression result. The result is obtained with prediction from a fitted linear model on cross-validation set. The overall R-Squared is 0.926. By further dividing the dataset into ‘In the money’ and ‘Out of the money’ part, we can observe the dispersion in prediction accuracy on these two parts. The performance is promising for ‘In the money’ observations since the pricing formula has relatively more linear relationship with inputs when $S > K$. MLP captures the linear relationship, but it fails to deliver the pricing procedure for $S < K$ observations.

Table 3.1.3.1 LR Performance

SSE	MSE	SST	SSR	RSQUARED	TYPE
20.99	0.004	193.221	153.554	0.891	'Whole Sample'
0.168	0	136.348	133.78	0.999	'In the money'
0.233	0	0.684	0.503	0.659	'Out of the money'

Since linear structure may not be a good approximation, a simple idea is to approximate with higher orders of inputs. I expand the linear regression model to a polynomial regression with quadratic form. An explicit form is $f(x_1, x_2) = \alpha_1 x_1 + \alpha_2 x_1^2 + \alpha_3 x_2 + \alpha_4 x_2^2 + \alpha_5 x_1 x_2$.

The result for prediction on cross-validation set is summarized in **Table 3.1.3.2**. We can see that the model has better performance on the whole sample, though the performance remains similar or

even worth when we treat the sample separately with respect to in/out of the money.

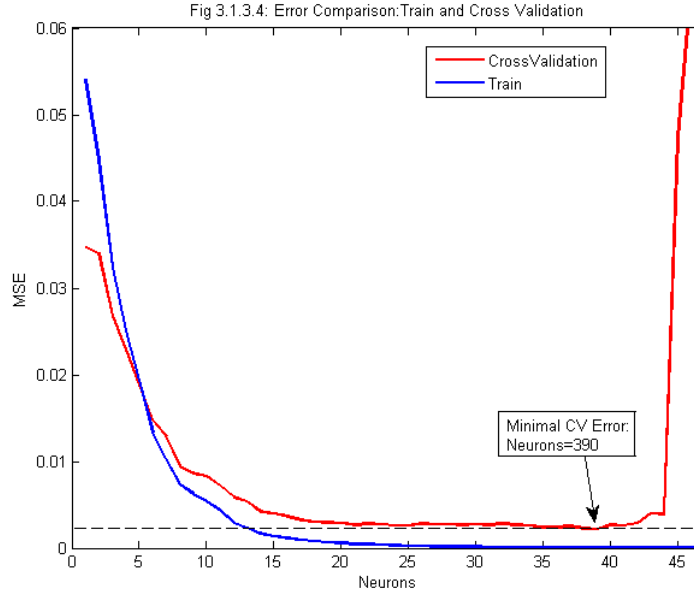
Table 3.1.3.2 PR Performance

SSE	MSE	SST	SSR	RSQUARED	TYPE
20.71	0.004	193.221	267.556	0.893	'Whole Sample'
0.709	0	136.348	140.165	0.995	'In the money'
0.373	0	0.684	0.333	0.455	'Out of the money'

Then I proceed to apply learning net to the identification of pricing formulae. I will apply radial-basis function and multiple-layer network here. As I have mentioned, learning nets are trained on a randomly separated training set and evaluated on a cross-validation set. The max iterations (or minimal error) shall be deliberately chosen to avoid over-fitting issues (normally, a penalized term is included in the objective function to avoid over-fitting). For the selection of neurons, I apply the rule to optimize cross-validation error, as shown in **Fig 3.1.3.4**. The mean squared error (MSE) value is plotted against the number of neurons in the radial basis function model. Two curves are plotted. The blue one indicates the training result, where we shall expect to see a monotonously declining value in mean squared error. The red curve stands for the result when a trained model is applied to predict cross-validation set. The curve forms a U-shape: it drops in the beginning, reaches its minimum and then bounces back. In the language of statistical learning, we say that the left end shows high bias, which indicates under-fitting; the right end shows high variance, which indicates over-fitting. To achieve best prediction accuracy and avoid over-fitting, the value of neurons is set to the one that minimizes cross-validation MSE. In this figure, the number is 390. Detailed amount of related statistical quantities are illustrated in **Table 3.1.3.3**. Compared with **Table 3.1.3.2**, the RBF has slightly better performance with respect to R-Squared.

Table 3.1.3.3 RBF Performance

SSE	MSE	SST	SSR	RSQUARED	EPOCHS	SIZE	TYPE
0.133	0	103.16	103.027	0.999	390	1500	'Train'
8.68	0.002	135.875	134.944	0.936	390	1500	'CrossValidation'



The case with multiple-layer perceptron is similar but slightly more complicated in the sense that the network setting is more flexible in terms of network structure, train methods, transition functions and etc. I take the most common setting with three hidden layers (3,5,3) , Levenberger-Marquardt method and sigmoid transition function. The performance on training set and cross-validation set is shown in **Table 3.1.3.4** and **Fig 3.1.3.5**, which apparently outperforms pervious methods. This displays the strong predictability using a simple network.

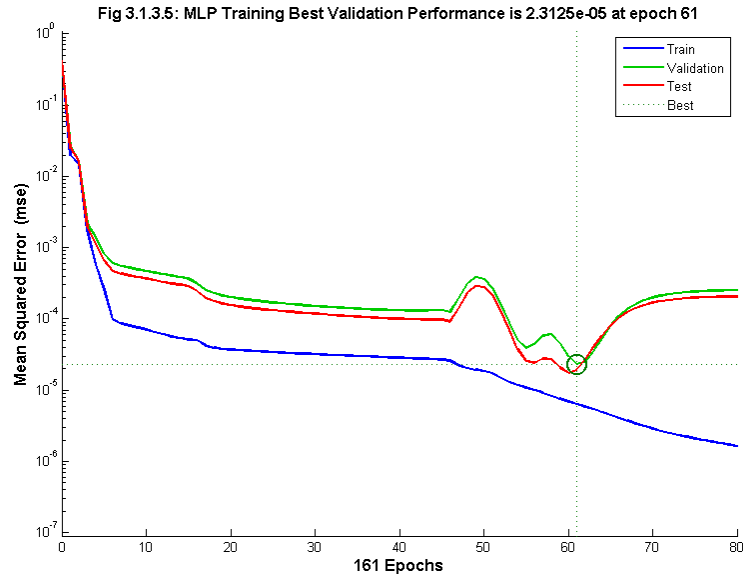
Table 3.1.3.4: MLP Performance

SSE	MSE	SST	SSR	RSQUARED	TYPE
0.063	0.000	275.006	274.773	1.000	'Train'
0.108	0.000	193.221	194.505	0.999	'CrossValidation'

```

\begin{tabular}{c|c|c|c|c|c}
\hline
\multicolumn{6}{c}{Table 3.1.3.3 RBF Performance}
\hline
SSE&MSE&SST    &SSR    &R^2&TYPE
\hline
0.133& 0& 103.16& 103.027&0.999& 'Train'
\hline
8.68& 0.002& 135.875&134.944&0.936& 'CrossValidation'
\hline
\multicolumn{6}{c}{Table 3.1.3.4: MLP Performance}
\hline
SSE& MSE& SST& SSR& R^2&TYPE
\hline
0.063& 0.000& 275.006&274.773&1.000& 'Train'
\hline
0.108& 0.000& 193.221&194.505&0.999& 'CrossValidation'
\hline
\end{tabular}

```



Measures as R-Squared and MSE can be good indicators of prediction accuracy. But in this specific case, the dynamic hedging error with underlying prediction function can be a better proxy with its real-world utilization. The amount can be calculated using the “tracking error” of replicating portfolios used to delta-hedge an option position. To be more specific, suppose we sell a call option at time 0 and apply dynamic trading strategies in stocks and bonds to hedge the option during its life according to its recognized price. Then the difference between the terminal value of option and our trading portfolio is the approximation of tracking error. A formal formulation of the dynamic hedging problem and my simulation procedures and results will be provided in Chapter 4.

3.2 Simulation and Pricing under Stochastic Volatility Scheme

In this section I will extend the previous network pricing method to the pricing under stochastic volatility scheme. I will start by providing some background scenario of stochastic volatility model, preceded by the theoretical pricing formula. Training paths and testing paths will be generated as done before to apply the learning net to the pricing of European style options with stochastic volatility assumption.

3.2.1 Introduction

Under the Black-Scholes model, the diffusion process of asset price is assumed to have constant volatility. However, empirical results have shown conditional heteroskedasticity in the returns of financial time series. The implied volatility smile is of immense interest to researchers and practitioners. There are other interesting fact with volatility and price, say leverage effect, known as the negative relationship between asset price and volatility. Stochastic volatility model can account for these facts with a pre-assumed stochastic process for underlying asset’s volatility.

The stochastic volatility model is actually a generalized type of Black-Scholes process, with the assumption that the underlying volatility follows Cox, Ingersoll and Ross[3]. The model extends Black-Scholes in the sense that volatility is assumed to follow another stochastic process that is correlated to asset price with certain coefficient. It can be used to model certain features of stock returns, including semi-heavy tails, volatility clustering, and the leverage effect (the negative correlation between changes in volatility and stock prices). The model is meaningful based on following observations:

- a. There is significant variability over the historical volatility of traded assets. The non-constant volatility shall be taken into account for hedging and evaluation of hedging results.
- b. Some products have been built on volatility, such as variance swaps and skew swaps. This presumes that volatility can be treated as a tradable risky asset.

The process can be represented as following:

$$\begin{aligned} dS(t) &= \mu S(t)dt + \sqrt{v(t)}S(t)dZ_1 \\ dv(t) &= \kappa(\theta - v(t))dt + \sigma\sqrt{v(t)}dZ_2 \\ \text{corr}(Z_1, Z_2) &= \rho \end{aligned} \quad (2.3.1.1)$$

The model makes assumption about the nature of volatility of underlying asset. The volatility evolves according to a mean-reversion spread. κ stands for the rate of mean reversion. θ is the long-term rate of volatility. σ is the volatility of volatility.

The model can also be simulated by discretizing on $\{t_i \mid 0 = t_0 < t_1 < \dots < t_n = T\}$.

$$S(t_{i+1}) = S(t_i) \exp\left(\mu - \frac{1}{2}v(t_i)\right)(t_{i+1} - t_i) + \sigma\sqrt{t_{i+1} - t_i}Z_{i+1}^{(1)} \quad i = 0, 1, \dots, n-1$$

$$v(t_{i+1}) = v(t_i) + \kappa(\theta - v(t_i))(t_{i+1} - t_i) + \sigma\sqrt{v(t_i)}\sqrt{t_{i+1} - t_i}Z_{i+1}^{(2)} \quad i = 0, 1, \dots, n-1$$

$$\text{where: } Z_{i+1}^{(2)} = \rho Z_{i+1}^{(1)} + \sqrt{1 - \rho^2}u_{i+1}$$

I display some typical paths of stochastic volatility to show the changes of its configurations with changing underlying parameters. I use the same set of random normal variables to generate price and volatility paths under different sets of parameters. The original setting is:

$$r = .05, v = .1, \kappa = .2, \theta = .1, \lambda = 0, \sigma = .1, \rho = 0, N = 1000, T = 4, S = 100$$

Heston[4] proposed the semi-closed analytical result to the model. As the case with Black-Scholes, the pricing of European options under stochastic volatility assumption can be derived through hedging with objective assets. The difference lies in the requirement of another option contract together with underlying asset to hedge for the two stochastic terms Z_1 and Z_2 in the formula (2.3.1.1). Suppose we want to value a European-style option V_t with underlying asset S_t . We

want to hedge the option with S_t and another option V_{1t} of different expiration and strike price. We hope to form a risk free portfolio by longing one option and shorting respective shares of assets and another option:

$$\Pi_t = V_t - \Delta_1 S_t - \Delta_2 V_{1t}$$

We are interested in finding Δ_1 and Δ_2 to form the hedging where Π_t is risk-less over $[t, t + dt]$. We can do this by calculating $d\Pi_t = dV_t - \Delta_1 dS_t - \Delta_2 dV_{1t}$ with Ito's Lemma, and let the coefficient of stochastic terms to be zero. Then we derive:

$$\Delta_2 = \frac{\frac{\partial V}{\partial v}}{\frac{\partial V_{1t}}{\partial v}}$$

$$\Delta_1 = \frac{\partial V}{\partial S} - \frac{\frac{\partial V}{\partial v}}{\frac{\partial V_{1t}}{\partial v}} \frac{\partial V_{1t}}{\partial S}$$

From this ratio we can derive the partial differential equation satisfied by option price. A semi-closed solution is provided by Heston. The option price depends on the contemporaneous value of asset price and volatility. The result can be derived through numerical integration, and the MatLab code for calculating this value can be found in **Appendix I**. I will not refer to the explicit form of solution due to complexity. Rather, I denote the solution as:

$$\hat{C}_{HS} = f_{HS}(S(t), v(t), T - t, K).$$

A closed-end formulation of Heston price can also be found in Appendix I, together with the PDE formulation of the pricing problem. More details of this PDE derivation are available in Rouah[5] (2013) and LiShang Jiang *et al.*[6] (2008). Due to the homogeneity in asset price and strike price, we can simply divide strike price on both sides of the equation and reduce the dimension of this approximation by 1:

$$\hat{C}_{HS} / K = f_{HS}(S(t) / K, v(t), T - t, 1)$$

A typical surface for stochastic volatility model is given as following. In order to visualize the plot, I keep the instantaneous volatility constant as 0.10.

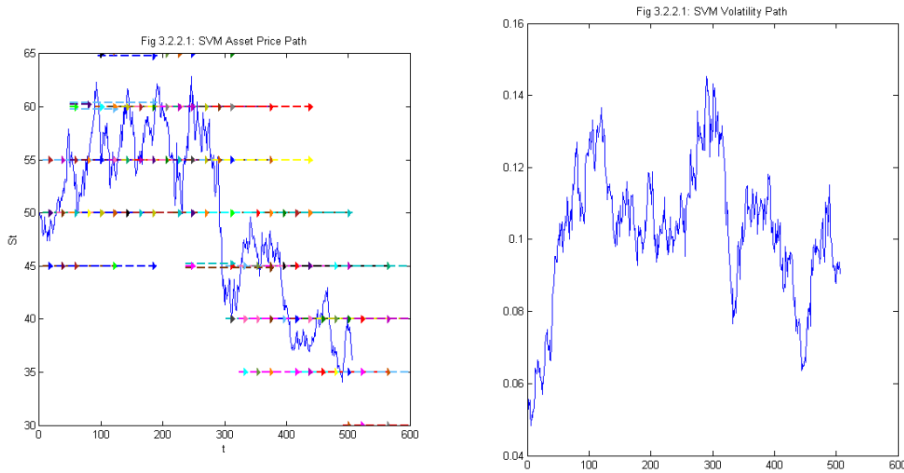
3.2.2 Data Generation and In-sample Performance

I use a typical parameter setting of Stochastic Volatility model to generate data sequences:

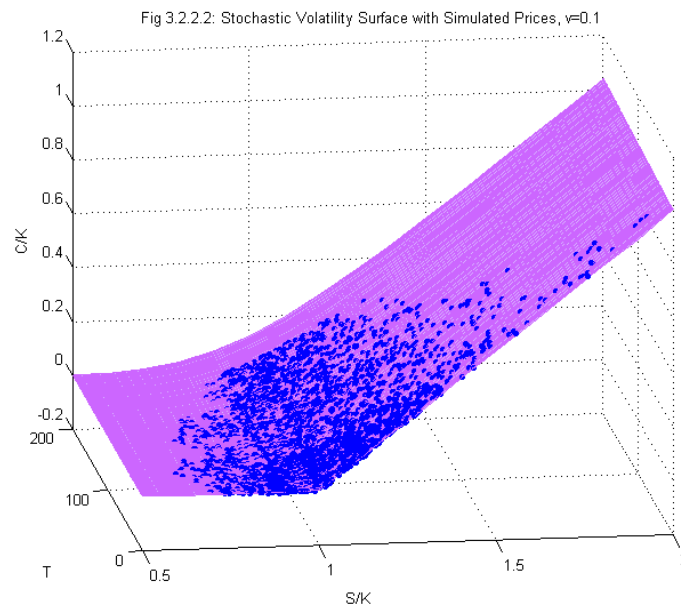
$$S(0) = 50, v(0) = 0.05, \kappa = 5, \theta = 0.10, \lambda = 0, \mu = 0.10, \sigma = 0.20, \rho = 0, T = 2, N = 506.$$

Option contracts are assigned to the generated paths according to the previously-introduced CBOE

laws. I give a typical asset price path with its corresponding contracts labeled in **Fig 3.2.2.1**. The left panel is the underlying asset price, and the right panel denotes the volatility's process, which clearly follows a mean-reverting spread.



The number of data points varies for each price path. It approximately has a mean of 8354 (16.51 contracts not yet expired at each trading day). Based on this generation, I form the whole training dataset by generating 5 paths and randomly sampling 2,000 observations from each of the paths. Therefore, the training set consists of 10,000 observations from 5 price sequences (around 40,000 overall observations). Each observation has four inputs: instantaneous price divided by strike price, instantaneous volatility, time left to expiration. Then the objective is given by the Heston price with the proposed values of inputs. For the cross-validation set, 1000 observations are sampled from each path with a total size of 5000 observations. I plot the simulated prices with the Heston surface. Since the difficulty to plot all four dimensions with price against moneyness, maturity and volatility, I further shrink the inputs by 1 by denoting $v=0.1$. The scatters are plotted for those observations where instantaneous volatility lies between 0.09 and 0.11. Although there is deviations in volatility, the observations still fit the surface quite well.



I will apply the previously-implemented methods of linear regression, polynomial regression, MLP and RBF to the approximation of in-sample and cross-validation data. Here I assume the instantaneous volatility input is well-identified with its true value. This is not the case in reality, and I will specially talk about parameter fitting and volatility estimation in next Chapter. However, here we just want to identify the fitting power of these different methods, therefore I will save the trouble of estimating and directly apply original volatility data. Notice that this will lead to a higher prediction accuracy than should be.

The linear regression model is first applied here. The corresponding results are provided in **Table 3.2.2.1**. The RSquared value is around 0.92 for in-sample and cross-validation. The value is approximately the same with that of constant volatility assumption. And as we can see on the subset 'In the money' and 'Out the money', there is great dispersion with respect to moneyness. The linear regression model mainly accounts for in-the-money observations, and leaves the other part unexplained. This is understandable from the configuration of **Fig 3.2.2.2**, as the linear model shall lay more weight on the in-the-money slope, therefore engender misleading prediction for out-the-money observations.

A direct extension to the simple linear model is second-order polynomial approximation. The regressors contain all the first and second order terms. The significant improvement with regard to R-squared reveals the explanatory power of a second-ordered model. Moreover, the result indicates better performance compared with the pricing of Black-Scholes model. There does exist great dispersion when we separate the dataset into 'in the money' and 'out of the money' section. The model can mainly account for the observations 'in the money', but leaves the 'out of the money' part unexplained. We then turn to the network models in order to find a better approximation over the whole sample space.

Table 3.2.2.1: LR and PR Performance, CV

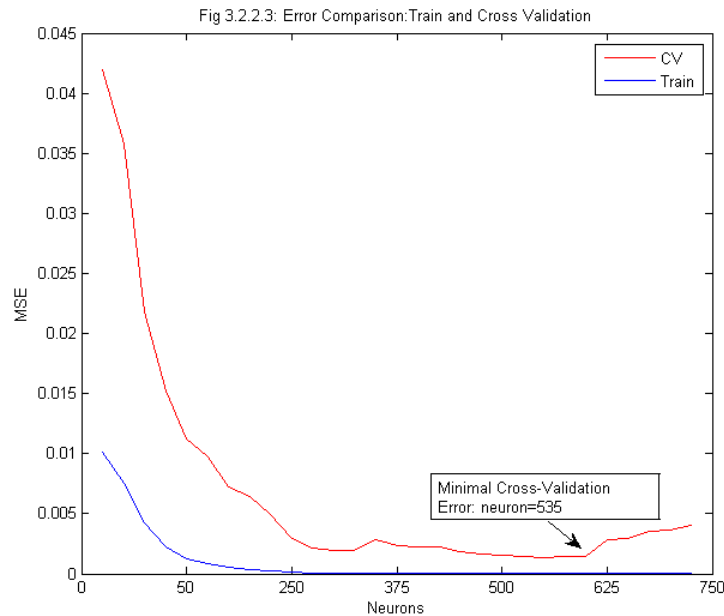
METHOD	SSE	MSE	SST	SSR	RSQUARED	TYPE
LR	5.785	0.001	70.535	60.572	0.918	'Whole Sample'
	3.206	0.001	41.940	23.265	0.924	'In the money'
	2.578	0.001	1.787	5.914	-0.443	'Out of the money'
PR	0.985	0.000	70.535	67.490	0.986	'Whole Sample'
	0.660	0.000	41.940	39.864	0.984	'In the money'
	0.325	0.000	1.787	2.382	0.818	'Out of the money'

The main concern with a radial basis network model is the selection of hidden layers. Previously, I select the number of layers to minimize cross-validation error. I generate 50 paths for cross-validation, and calculate the mean and standard deviation of prediction result of 5 models: radial basis function with 1500, 1000, 500, 300 and 100 hidden layers respectively. We can see from the result that the performance again forms a U-shaped trend, with best performance for a 500-layer model. A typical training process is available in **Fig 3.2.2.3**. The pattern is similar to the previous training of GBM model. A larger number of hidden layers is required to achieve minimal MSE in cross-validation set. This indicates the greater complexity of the model to fit stochastic volatility pricing. Table 3.2.2.2 gives descriptive statistics of the cross-validation performance

with respect to number of layers in RBF.

Table 3.2.2.2 RBF Performance(Cross-Validation)

Num of Layers	1500	1000	500	300	100
RSQUARED	0.600	0.600	0.993	0.979	0.724
MSE	0.010	0.010	0.000	0.000	0.004



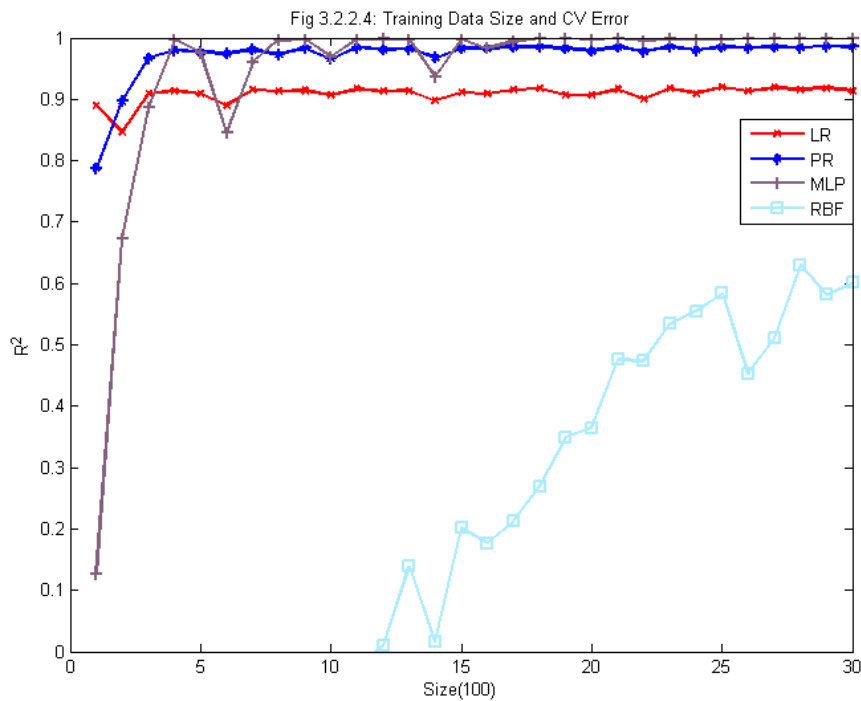
Finally, I provide the result of training and cross-validation through a multiple-layer perceptron model. The configuration of the network is set to be 3 layers with neurons' number 3,5,3 respectively. The structure is displayed in the following figure. The training function applied here is Levenberg-Marquardt backpropagation. The training and cross-validation dataset is applied as implemented with previous models. The results given in the **Table 3.2.2.3** show promising predictive power in terms of R-Squared both in-sample and in cross-validation.

Table 3.2.2.3: MLP Performance

SSE	MSE	SST	SSR	RSQUARED	TYPE
0.002	0.000	62.098	62.095	1.000	'Train'
0.001	0.000	25.609	25.598	0.997	'CrossValidation'

One key issue in applying statistical learning methods is the training data size. We would like to see how many training data should be sufficient to make accurate prediction on cross-validation set. With a small training data size, the model tends to have very good in-sample prediction performance, but will lack consistency on cross-validation set since it overfits. But a training set being too large may just end up being a waste of consumption of time. I would like to suggest the appropriate data size for making cross-validation prediction. I run the following procedures. Given the original training dataset, I random sample a certain amount of observations from it to form a new training set. Then I will train the methods on this set, and check the prediction accuracy on

cross-validation set with respect to the size of training set. **Fig 3.2.2.4** plots the RSquared data on cross-validation set against the size of training data (in 100). We can see that parametric estimation methods remain stable for small size. MLP's performance improves significantly with training data when size is small (<500). The performance is nice but unstable for training data size between 500 and 2000. And finally, the performance is stable and promising for larger size. Further enlarging training dataset won't improve it any more. Therefore a training data of a size around 2000 shall be sufficient to train MLP. As shown in the figure, RBF requires a larger size of data to achieve good prediction. The training effect will also depend on the diversification of training dataset. If the training dataset consists of options with rather similar characteristics in moneyness and maturity, then the statistical learning methods trained on it will lack predictive power in data of different moneyness or maturity. Therefore in providing training data we should guarantee that the dataset is diversified enough.



4 Static and Dynamic Measures of Model Performance

I will propose two measures for the out-of-sample error of parametric and nonparametric methods in this chapter. The static measure includes statistics measuring out-sample pricing accuracy. The dynamic measure will focus on dynamic hedging scheme. The context in this chapter will be outlined as following. In 4.1 I give some analysis about dynamic hedging and its discrete version. In 4.2, I formulate the dynamic hedging strategy for the constant volatility (BS) case. In 4.3, I provide analysis of parameter estimation approaches required for calculating metrics in stochastic volatility scheme. In 4.4, I will provide detailed data and error analysis of static measure. In 4.5, I

formulate two dynamic hedging strategies in stochastic volatility scenario. In 4.6, I provide results from dynamic hedging.

4.1 Introduction: Discrete Time Dynamic Hedging

The dynamic hedging strategy following the derivation in Black-Scholes or Heston formula is based on continuous time. The strategy is expected to have zero error for replicating portfolio. However, it is impossible to hedge continuously in real world. Applying continuous-time derived deltas to hedge in discrete time will certainly engender errors. Hayashi, T. et al.[7] (2005) proposed an asymptotic approach to evaluate hedging errors due to the discreteness of trading. Also difficulty arises with the parameter estimation in continuous time. Therefore, data-driven method has its prevalence in discrete time hedging since we reinforce learning on discrete datasets. Therefore, a discrete dynamic hedging strategy should be efficient in evaluating the performance of a data-driven nonparametric method. Some of the studies on data-driven discrete hedging methods includes Bossaerts, P. et al.[8] (1997) and Bossaerts, P. et al.[9] (2003). All the analysis in the following context will be done on a discrete time grid.

4.2 Dynamic Hedging Strategy Formulation and Simulation: Constant Volatility

Let $V_O(t), V_S(t), V_B(t)$ denote the value of option, stock and bond at time t , and

$V(t) = V_O(t) + V_S(t) + V_B(t)$ shall denote the dollar value of replicating portfolio. Suppose we sell 1 share of European call option at $t=0$, then delta shares of stocks shall be bought at period t to

offset the returns from call option with the formula $\Delta_t = \frac{\partial C(t)}{\partial S}$. For our different estimation of

C , we shall get different delta and different result from delta-hedging. Suppose radial basis function is applied here, then at time 0,

$$\begin{aligned} V_C(0) &= -f_{BS}(0) \\ V_S(0) &= S(0)\Delta_{RBF}(0) = S(0)\frac{\partial f_{RBF}(0)}{\partial S} \\ V_B(0) &= -V_S(0) - V_C(0) \end{aligned}$$

Here f_{BS} is just Black-Scholes price and can be directly calculated with B-S formula. f_{RBF} is our estimated form, and $\Delta_{RBF}(0)$ can be calculated with numerical approximation

$$\Delta_{RBF}(t) = \frac{f_{RBF}(S(t) + \varepsilon) - f_{RBF}(\varepsilon)}{\varepsilon}.$$

Then at time $t, 1 \leq t \leq T$, we rebalance the account as

¹ An alternative is to calculate this analytically through the expression of f_{rbf} .

$$V_C(t) = -f_{BS}(t)$$

$$V_S(t) = S(t)\Delta_{RBF}(t) = S(t)\frac{\partial f_{RBF}(t)}{\partial S}$$

$$V_B(t) = e^{r\Delta}V_B(t-\Delta) - S(t)(\Delta_{RBF}(t) - \Delta_{RBF}(t-\Delta))$$

I offer several indicators for the tracking error. The first is most straightforward. This measures the expectation of tracking error over many test paths:

$$\gamma = e^{-rT}E[|V(T)|]$$

There can be other types of definitions. By taking the variance into consideration, we have the prediction error:

$$\eta = e^{-rT}\sqrt{E^2[|V(T)|] + \text{Var}[V(T)]}$$

It takes into account the variance of tracking error. This makes sense since we want the tracking strategy to be consistent over different asset price paths. A tracking strategy with small error but large dispersion is still not desirable.

Another measure is the tracking error over the whole life of the derivative. This indicates the match of our duplicating portfolio over the time period. It is written as:

$$\lambda = \frac{1}{T}\int_0^T e^{-rt} |V(t)| dt$$

in continuous time, and

$$\lambda = \frac{1}{n}\sum_{i=1}^n e^{-rt_i} |V(t_i)|$$

is the corresponding discrete time representation.

We can compare the result with the benchmark of hedging with Black-Scholes formula. For

Black-Scholes hedging, we can just calculate the hedging ratio $\frac{\partial C}{\partial S} = N(d_1)$, where

$$d_1 = \frac{\ln(S_0 / K) + (r + \sigma^2 / 2)(T - t)}{\sigma\sqrt{T - t}}.$$

Notice that the volatility parameter here may not be correctly learned, but only estimated from historical data. In our simulation, this value is estimated as the volatility of training data. Although the estimation can be made quite accurately under a constant volatility assumption, there is still certain error engendered. One benefit about the network-fitting hedging is that no volatility term is required to be estimated (or rather, the approximation model itself contains sufficient information for delta hedging).

4.3 Parameter Fitting with Stochastic Volatility Sequence

4.3.1 Parameter Estimation

I have already made parameter estimation in Black-Scholes hedging problem, where the volatility parameter is estimated from historical training data. Here in the hedging problem of stochastic volatility model, we still encounter the problem of unobservable parameters. This includes the instantaneous volatility $v(t)$, variance of volatility σ , long-term volatility θ , mean-reverting spread of volatility process κ and correlation coefficient ρ . Other terms are intrinsic properties of an option and can be directly quoted from the market, including strike and time to maturity. To estimate these parameters of the stochastic process, there are several approaches applicable here. We can apply econometric methodologies to achieve required estimation (like generalized methods of moments and maximum likelihood). But these methods are demanding on historical data, which makes it less preferable to our problem. Therefore I refer to the local optimization method here, as applied in Bakshi, *et al.* [10] (1997). This method doesn't have stringent requirement for dataset, and have proved to be advantageous in its estimation result. The logic of this method is simple. We want to select parameter sets to minimize summed squared error term of our estimated option price from the quoted market price. The optimization can be run on daily frequency. It can also be extended to a wider span of time period. To be more specifically, the concrete steps of this procedure can be formulated in following paragraph.

Firstly, we collect a series of option data for a specific trading day t . Our dataset has the following components for a number of options: time to maturity T , strike price K , instantaneous price of underlying asset $S(t)$ and corresponding quoted option price C . We write the price as $C_i = C(S(t), T_i, K_i), i = 1, \dots, N_t$, for the N_t options available on day t . Then with our estimating dataset $\Theta = [\sigma, \theta, \kappa, \rho]$ and instantaneous volatility $v(t)$ we can calculate the estimated option price with Heston pricing formula: $\hat{C}_i = \hat{C}(S(t), T_i, K_i, v(t), \Theta), i = 1, \dots, N_t$. Our purpose is to minimize the error between estimated value and real market value. We can define the distance between estimated and quoted value as following:

$$\varepsilon_i(v(t), \Theta) = \hat{C}_i(v(t), \Theta) - C_i, i = 1, \dots, N_t$$

Then the summed squared error for this specific trading day is defined as:

$$SSE_t(v(t), \Theta) = \sum_{i=1}^{N_t} [\varepsilon_i(v(t), \Theta)]^2$$

For each day, we run the optimizer to access a desired value of local volatility and parameter set to achieve minimal error. The optimization problem is formulated as:

$$\min_{v(t), \Theta} SSE_t(v(t), \Theta)$$

Then we can loop over the steps to achieve parameter estimation for each trading day with data

available. We can calculate the mean and standard error of our estimated values of instantaneous volatility and parameter set.

There are some further claims we can make about this estimation. From the nature of SSE function, we have larger weights on more expensive options, that is to say, in-the-money options and options with long maturity. Correspondingly, we have less weight assigned to cheap options, meaning out-of-the-money and short maturity options. Another approach is to minimize over the squared sum of percentage of error:

$$\varepsilon'_i(v(t), \Theta) = \frac{\hat{C}_i(v(t), \Theta) - C_i}{C_i}, i = 1, \dots, N_t$$

$$\min_{v(t), \Theta} SSPE_t(v(t), \Theta) = \sum_{i=1}^{N_t} [\varepsilon'_i(v(t), \Theta)]^2$$

But this estimation could still be biased in a more favorable position on cheaper options. We can make some adjustment by assigning weight to the summation that makes the term more balanced through different option contracts. Other adjustment including making optimization on different subsets of dataset and taking arithmetic average of estimated value.

I offer several alternatives to the local optimization method. Since the parameters of stochastic volatility are assumed to be constant over time, the estimation over the time is actually reasonable and achievable. The only time-varying unobservable parameter is the instantaneous volatility. Therefore we can first make an estimation of instantaneous volatility via the implied volatility of Black-Scholes model. I use the method implied by Ait-Sahalia *et al.* (2007). The approach is to use the at-the-money, close-to-maturity option's implied volatility as a proxy for instantaneous volatility. To be more specific, at each trading day, we consider an option that is at-the-money with 30 days to expiration. We denote its price as $C^R = C^{HS}(v(t), \dots)$, which is a Heston price priced with the previously-claimed parameter setting. Then we find the Black-Scholes implied volatility of this price:

$$C^{BS}(\sigma^*(t), \dots) = C^R$$

Then this backwardly solved implied volatility can be viewed as the proxy for the real instantaneous volatility. Denote the optimized value as $v^*(t) = [\sigma^*(t)]^2$. Then I will run the optimization of the rest parameter set over the whole sample with the locally optimized value of instantaneous volatility:

$$\min_{\Theta} SSE_t(\Theta) = \sum_{t=1}^T \sum_{i=1}^{N_t} [\varepsilon_i(v^*(t), \Theta)]^2$$

Then the optimized parameter shall be consistent over the whole sample space.

We must notice that this implied volatility is biased in approximating true instantaneous volatility.

The Black-Scholes implied volatility is actually the integration of true instantaneous volatility over the rest of the option's life. Due to the mean-reversion spread, this approximation would be too high for low instantaneous volatility, since it tends to bounce back shortly afterwards. With the same logic, it could be too low for high instantaneous volatility. There can some refinement made to fit this scenario. Consider the Black-Scholes implied variance represented as:

$$V(t, T) = \frac{1}{T-t} \int_t^T v(t) dt$$

We notice that v has a drift term of $\kappa\theta - \kappa v(t)$ in stochastic volatility model. Then we can calculate the expected value of this implied variance term:

$$E_t[V(t, T)] = \left[\frac{e^{-\kappa(T-t)} - 1}{-\kappa(T-t)} \right] [v(t) - \theta] + \theta$$

Solve this backwardly, we will have the approximation for instantaneous volatility:

$$v(t) \approx \frac{-\kappa(T-t)V_{imp}(t, T) + \kappa\theta(T-t)}{e^{-\kappa(T-t)} - 1} + \theta$$

This formula will fit the biased-estimation with Black-Scholes implied volatility. The effect of this refinement will be illustrated in the next section. With this formula, I actually represent the instantaneous volatility with respect to two other parameters: kappa and theta. Then in our optimization problem, we can denote $v(t) = v_t(\kappa, \theta)$, therefore we have removed the volatility term in our optimization problem. We can optimize universally with respect to four global parameters.

$$\min_{\Theta} SSE_t(\Theta) = \sum_{t=1}^T \sum_{i=1}^{N_t} [\varepsilon_i(v_t(\Theta), \Theta)]^2$$

4.3.2 Optimizing Scenario

MatLab provides abundant tools for linear and nonlinear optimization. Many of them are available to our problem formulation. I apply the function `fmincon` in the problem. There are several specifications worth mentioning about the algorithm:

1) Algorithm

`fmincon` offers four algorithm options: 'trust-region-reflective', 'interior-point', 'sqp' and 'active-set'. 'interior-point' can handle large-scale problem; The algorithm can use special techniques for large-scale problems. 'sqp' satisfies bounds at all iterations. It is not a large-scale algorithm; 'active-set' can take larger steps, which will add up speed. The algorithm is effective on some problems with nonsmooth constraints. It is not a large-scale algorithm; 'trust-region-reflective' requires you to provide a gradient, and allows only bounds or linear equality constraints, but not both. Within these limitations, the algorithm handles both large sparse problems and small dense problems efficiently. Our problem is not a large-scale problem, and there is difficulty in supplying

the gradient. Therefore, ‘active-set’ algorithm is applied.

2) Bounds

The problem doesn’t have linear or nonlinear constraints, but it does have a bound. For instantaneous volatility, standard deviation of volatility, long-term volatility, the bound is set to be $[0,1]$, which is a reasonable bound empirically. For mean-reverting spread, it is set to be $[0,10]$.

The bound for correlation coefficient can be $[-1,1]$ or $[-1,0]$. The latter makes sense for leverage effect in empirical study.

3) Initial Values

In order to make the optimization problem more economical in time consumption and accurate in output, the selection of initial value should be reasonable and not far from the true value.

Ait-Sahalia and Kimmel [11] (2007) implemented the implied volatility of 30-day at-the-money options to be an estimation for unobserved instantaneous volatility. Therefore we can suggest this value for the initialization of instantaneous volatility in the optimization.

We apply the ‘Smart Parameter’ Gauthier and Rivaille [12] (2009) to find initial values of ρ and σ . The fundamental underlying idea is the approximation of option price through expansion approach, as proposed by Benhamou *et al.* [13] (2010). I will not go deep into the details.

4.3.3 Simulation Results

I test the algorithmic efficiency on a simulated dataset. The dataset is generated with the frequently-implemented method in pervious context. The initial parameter is set as following:

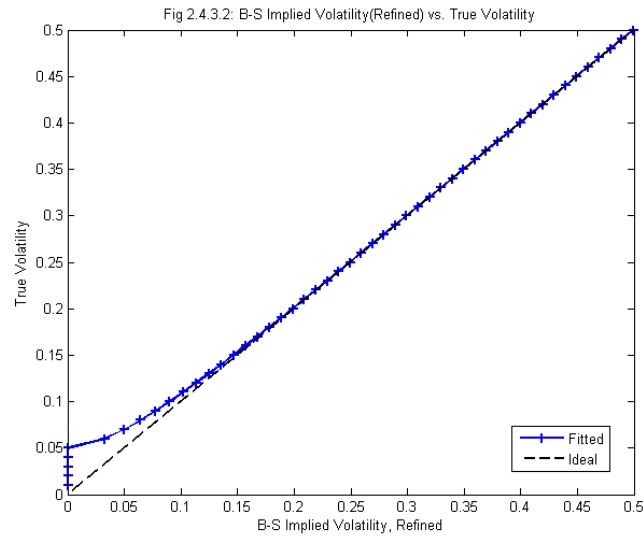
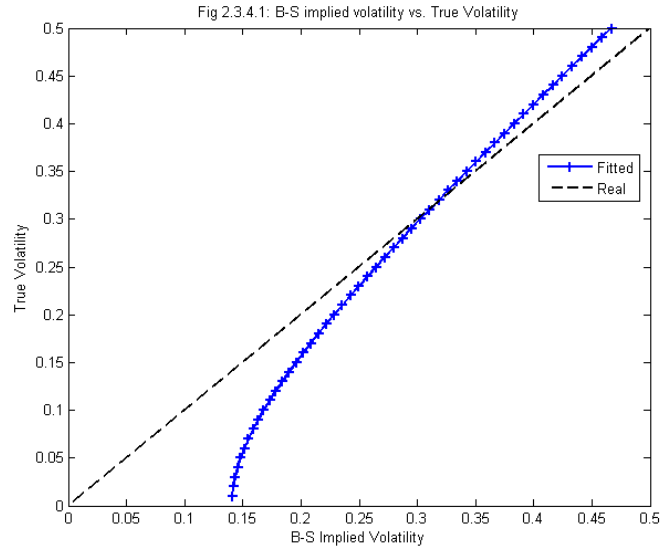
$$\begin{aligned} \mu &= .1; \quad r = .05; \quad n = 253; \quad T = 2; \quad S_0 = 50; \\ v_0 &= .05; \quad \kappa = 5; \quad \theta = .1; \quad \text{sig} = .2; \quad \rho = -.5; \quad \lambda = 0; \end{aligned}$$

There is on average 7438 contracts assigned to the price sequence, with an average of 14.7 contracts per trading day.

First, I provide result for volatility fitting with B-S implied volatility. Table 2.4.3.1 contains the average approximation error and its standard error on 2-years’ simulated data (506 trading day observations). Fig 2.3.4.1 and Fig 2.3.4.2 provide the plot of fitted-value curve. I plot the Black-Scholes implied volatility against true volatility. If the implied volatility fits perfectly, the fitted curve should be a line-segment with 45 degree tangent. We can see from the figure our fitting result (the blue curve) and the ideal fitting (the black curve). In the case where original Black-Scholes implied volatility is directly used as proxy, the B-S implied volatility tends to under-fit for low volatility and over-fit for high volatility. This is natural from the fact of mean-reversion spread in volatility. After the refinement, we can see from Fig 2.3.4.2 that the line fits ideally. There is no bias at two ends, with only certain error with low implied volatility.

Table 4.3.3.1 Volatility Estimation

	B-S Implied	B-S Implied Refined
AE	0.003	3.73E-04
SE	3.36E-06	2.05E-07



I apply different types of methodologies to estimate the parameters. In general, there are local optimization method and whole sample optimization (WSO) method. For the local optimization problem, I optimize the parameters for each trading day, and take the arithmetic mean and standard deviation of parameter estimation. There are about 14.7 contracts available each trading day, which will be the average number of quadratic terms in our optimization formulation. For the whole sample space estimation, I offer three different variations. For the first formation, I run the optimization based on real volatility data. For the second one, I estimate the volatility via implied volatility of at-the-money short-maturity option. Finally, I implement the refined estimation of instantaneous volatility. All these methods have been explained in detail in previous section. One thing worth noticing is that the large complexity of a summed squared error term over whole space

leaves the optimization problem intractable with regular optimization method. Therefore I randomly select a certain amount of contracts from the whole sample space and optimize over this representation. I try different number of contracts to be selected into the objective function (50,100, 200 and 300), and again take the arithmetic mean and standard error of the estimations over a number of tries (20 replications). Detailed data with respect to the previously-mentioned methods is provided in the following table.

In **Table 4.3.3.1**, there are four panels with respect to methodologies. The first column indicates the method applied, where WSO is the abbreviation for ‘Whole-Sample-Optimization’. The second to fifth column gives parameter estimation and standard error. The sixth column gives the absolute error and standard error in each optimization. The benchmark differs with respect to the size of optimization problem. To provide a more uniform index, I provide the relative error, which is defined as summed percentage error from estimated parameter set to benchmark parameter set:

$$RE = \sum_{i=1}^4 \left| \frac{\Theta_i - \hat{\Theta}_i}{\Theta_i} \right|.$$

I will make a brief analysis of the result. Whole-sample-optimization method with volatility hint has the best performance, with the advantage of given volatility knowledge. Other terms are significantly less robust in terms of relative and absolute error. This indicates the key position of volatility estimation. Of other methods, the WSO-IV method has great improvement with the refinement in volatility. It has a good fitting of kappa and theta, but is less accurate in estimating sigma and rho. The latter one is also advantageous with respect to standard error. It is more robust in the fitting. The local estimation has good performance in its estimation error, but the standard error is significantly larger.

It’s also interesting and informative to look within each panel. We’d like to see how many terms shall be sufficient in optimization formulation. More terms means a more complex optimization formulation, which is more computationally expensive. It should lead to a more accurate estimation, but as we can see this is not always the case. As we add the terms into WSO-IV and WSO-Refined-IV methods, we can see that the error doesn’t decrease monotonously. There is an improvement from 50 to 100. But when terms are further included, the error gets magnified. This obvious paradox could possibly due to the optimizer’s inability to handle large-scale optimizing problem. It may possibly end up with a less preferable result. Based on the consideration of this effect as well as time consumption, I regard 100 as a reasonable selection, and will continue to apply this number to study in latter part. The WSO with refined IV and 100 terms have the best performance over all the methods applied.

Table 4.3.3.1 provides more details about the out-of-sample prediction error. With the common practice, I divide the sample space into subspaces with respect to moneyness and maturity. For moneyness, I generate three subsets: in-the-money set, at-the-money set and out-of-the-money set. For maturity, I generate another three subsets: short maturity, medium maturity and long maturity. More details about the separation can be found in previous section.

Table 4.3.3.1 Parameter Estimation Results

	Kappa	Theta	sigma	Rho	A.E	R.E
WSO with Volatility Hint(100)	5.00 0	0.100	0.204	-0.489	5.713E-07	0.044
	1.734E-04	3.027E-05	0.004	0.008		
WSO with Volatility Hint(200)	5.000	0.100	0.209	-0.480	7.720E-06	0.083
	3.724E-04	1.057E-05	0.002	0.005	1.930E-06	
WSO with IV Estimation(50)	3.868	0.101	0.243	-0.394	0.005	0.663
	0.044	1.339E-04	0.014	0.029	0.002	
WSO with IV Estimation(100)	3.917	0.101	0.208	-0.446	0.008	0.373
	0.016	7.752E-05	0.010	0.019	0.001	
WSO with IV Estimation(200)	3.909	0.101	0.217	-0.428	0.017	0.454
	0.016	5.231E-05	0.006	0.015	0.005	
WSO with IV Estimation(300)	3.893	0.101	0.230	-0.401	0.026	0.577
	0.012	6.923E-05	0.007	0.012	0.008	
WSO with Refined IV Estimation(50)	5.066	0.100	0.164	-0.624	1.224E-04	0.440
	0.005	0.000	0.008	0.034	3.870E-05	
WSO with Refined IV Estimation(100)	5.065	0.100	0.184	-0.546	2.559E-04	0.184
	0.005	2.208E-05	0.005	0.014	8.094E-05	
WSO with Refined IV Estimation(200)	5.060	0.100	0.167	-0.602	4.627E-04	0.381
	0.003	1.953E-05	0.004	0.013	1.463E-04	
Local Optimization	4.055	0.103	0.212	-0.507	3.728E-04	0.289
	0.257	0.001	0.017	0.047	1.934E-04	

4.4 Out-sample Pricing Approximation: Static Approach

With the well-prepared parametric estimation method, I will try to give an exhaustive result about out-sample prediction with the parametric method based on ideal Heston formula. Especially, this result will be compared with nonparametric result applied in Chapter 3.

The out-sample fit is run on a deliberately-designed data sequence. It is generated as a three-year option price sequence priced with Heston formula. Options are attached according to CBOE. With this data prepared, I will go into parameter estimation, as well as parametric and nonparametric prediction of out-sample data. The procedure is summarized within the following steps

- 1) Generate a three-year pricing sequence with pre-given parameters.
- 2) Estimate instantaneous volatility on the pricing sequence.
- 3) Extract first two years' data as a subset to learn parameters (I will name it parameter-training set in the following context).

I apply three methods to estimate parameter: local optimization method, global optimization method with implied volatility and global optimization method with refined estimation of implied volatility.

- 1) Local Optimization method: This doesn't require estimation on prepared-subset.
- 2) Global optimization method with implied volatility: This requires the optimization on parameter-training set with the implementation of implied volatility data.
- 3) Global optimization method with refined estimation of implied volatility: This also requires the optimization on parameter-training set with the implementation of implied volatility data.
- 4) Implement the parameter-training data to train nonparametric models.

MLP and RBF will be trained on this subset. This requires the conventional three inputs and single output as processed in Chapter 3. But notice that we have several options about the input of instantaneous volatility. I will list the overall 6 choices:

- 1) Current Implied Volatility
- 2) Current Refined Implied Volatility
- 3) Current Real Volatility
- 4) Lagged Implied Volatility
- 5) Lagged Refined Implied Volatility
- 6) Lagged Real Volatility

Each of them is meaningful to some extent. The current data is intended to achieve higher in-sample prediction accuracy. But notice that we can only use lagged data in implied volatility when pricing out-sample data. Therefore a training based on lagged data is sensible to engender higher out-sample prediction accuracy. Also I consider three types of volatility estimation. The implied volatility is the most direct and widely-used approximation. The refined implied volatility approximation is intended to give a more robust estimation. We should notice that the acquisition of this data is dependent on the estimation of global parameters, therefore the pre-estimation of global parameters is regarded as a necessity for the acquirement of this data. Finally there is the true volatility. This is not achievable with empirical data, but achievable here with our simulated data. I implement it to form a comparison with other methods.

I implement the final one-year data as out-sample data. Global parameters are inherited from the pre-estimation on parameter-estimation subset.

In the Heston formula pricing approach with global parameters, out-sample price sequence is estimated via Heston Formula with pre-estimated global parameters. Since we can only have access to previous trading day's implied volatility in real world, the instantaneous volatility input is replaced with implied/refined implied volatility lagged for 1 trading day. This will lead to some misspecification error in the prediction.

In the Heston formula pricing approach with local parameter estimation, I run a local optimization on previous trading day's option data to give the local estimation, and provide a prediction of current day's option price with this set of parameters.

In the nonparametric approach, the prediction is quite straightforward. The previously-trained network models are simulated with three inputs: moneyness, maturity and lagged volatility. Two types of lagged volatility are implemented here: the normal version and the refined version. Several metrics are provided to evaluate the pricing accuracy. The first is the arithmetic mean of

absolute pricing error on out-sample dataset. The second is the arithmetic mean of pricing error. The third is the arithmetic mean of absolute percentage error in pricing. Detailed data are provided in **Table 4.4.1-3**.

In **Table 4.4.1-2**, I provide results with MLP prediction. The M1-M6 label in the column denotes the 6 different volatility inputs I used to train network. The first, third and fifth row indicates average error, average absolute error and average percentage error respectively. The second, fourth and sixth row are the standard error of the values. The left panel shows predictive results from implied volatility. The right panel shows predictive results from refined implied volatility. From the out-sample performance, we can also find out some facts about the result. Firstly, network trained with lagged data performs better than others. This is quite intuitive since we run the prediction with lagged data. Secondly, we have the best performance in M4 for implied volatility prediction, and M5 for refined implied volatility prediction. This result is also quite intuitive. The network trained with implied volatility ought to have better performance with implied volatility as input. The same is true for refined implied volatility. Moreover, we can see that M3 and M6 don't necessarily perform better with a volatility hint.

Table 4.4.1 MLP Out-Sample Approximation Error, IV							Table 4.4.2 MLP Out-Sample Approximation Error, RIV						
	M1	M2	M3	M4	M5	M6		M1	M2	M3	M4	M5	M6
A.E.	-0.018	-0.013	-0.011	-0.003	-0.017	-0.012	A.E.	-0.005	0.004	0.002	0.013	-0.003	0.000
(s.e.)	0.001	0.001	0.001	0.001	0.001	0.001	(s.e.)	0.001	0.001	0.001	0.001	0.001	0.001
A.A.E	0.055	0.066	0.069	0.043	0.045	0.060	A.A.E	0.052	0.067	0.067	0.046	0.042	0.057
(s.e.)	0.001	0.001	0.001	0.001	0.001	0.001	(s.e.)	0.001	0.001	0.001	0.001	0.001	0.001
A.P.E	0.471	0.580	0.678	0.250	0.267	0.494	A.P.E	0.463	0.582	0.672	0.255	0.267	0.485
(s.e.)	0.221	0.242	0.269	0.058	0.079	0.218	(s.e.)	0.218	0.243	0.266	0.055	0.076	0.215

Table 4.4.3 contains the result from the pricing with Heston formula and estimated parameters. Column 1 shows the result with globally optimized parameter via implied volatility. Column 2 is the result with globally optimized parameter via refined implied volatility. Column 3 is the result from locally optimized parameter. Column 4 is the result where there is a volatility hint in global optimization. This is only for reference purpose, and is not truly achievable. From the result, we can see that in spite of N4's perfect performance, N1-N3 also have very robust prediction. N2 and N3 have equivalently good performance in terms of average and standard error.

Table 4.4.3 Heston Pricing with Estimated Parameter				
	N1	N2	N3	N4
A.E.	0.014	-0.003	0.001	0.000
(s.e.)	0.001	0.000	0.000	0.000
A.A.E	0.024	0.016	0.016	0.001
(s.e.)	0.000	0.000	0.000	0.000
A.P.E	0.017	0.015	0.015	0.001
(s.e.)	0.001	0.001	0.001	0.000

We can then make a comparison between Heston pricing with learnt parameters and nonparametric pricing. We are mainly concerned with the best-performing nonparametric method,

which is MLP trained on lagged volatility input. The two are very close in terms of average error. Nonparametric method is slightly outperformed in terms of average absolute error. It is dominated when we look at average percentage error. This is due to the misspecification of low-value options leading to a magnification of percentage error.

To sum up, the pricing formula approach has its advantage over nonparametric method based on simulated data. But the nonparametric method is still efficient in tracking the pricing formula, and it is expected to have better performance when empirical data is applied, which is not priced perfectly according to the pricing formula, and global parameter cannot be very well identified as in the simulation.

It is worth noticing the large difference in percentage error while the average error of two methods doesn't vary too much. As I have said, this can be caused by the error on small-value option prices. The MLP method should have equally large error for small-value term, therefore leads to the large percentage error, while the pricing formula approach will have shrunk error on those small-value terms, keeping the overall percentage error constrained. To better realize the structure of pricing, I implement a partition of the dataset to see the performance of different methods on this partition of dataset.

Following the common practice, I divide the sample space into subspaces with respect to moneyness and maturity. First I plot the distribution of moneyness and maturity. **Fig 4.4.1** and **Fig 4.4.2** provides the histogram of moneyness and maturity distribution. Most observations of moneyness are ranged within 0.8 and 1.2. The distribution is close to normal with mean 1. The distribution of maturity is monotonously decreasing, means that there are fewer long-maturity options. There is an obvious change of slope at 40. Based on the configuration, I consider 6 subsets with respect to moneyness, separated with an interval of 0.10 around at-the-money position. For maturity, I generate another three subsets: short maturity, medium maturity and long maturity. A subset contains those positions within either one of the 6 moneyness range, and either one of the three maturity range. Therefore there are 18 subsets available, covering from deep-in-the-money to deep-out-the-money, from short maturity to long maturity. The number of observations within each moneyness-maturity subset is given in **Table 4.4.4**. There are more observations of close-to-money short-maturity options. Number decreases as maturity lengthens and price deviates from strike.

Finally, in **Table 4.4.5**, I provide result of prediction error with several typical methods on this partition of subsets. There are several panels in this table. Each panel denotes the prediction error from one method. There are four methods in all applied. The first two are two best performing MLP network. They are trained on lagged/refined lagged implied volatility data and make prediction with lagged/refined lagged implied volatility data. The following four are all the Heston pricing methods with different parameter estimation. Within each panel, I provide results on each subset of moneyness(maturity) and maturity(column). There are two categories of indexes used. The first is average of absolute error. The second is average of absolute percentage error.

I shall proceed to make some interpretations of the result. First let's look at the absolute error. There are some similar observations over different methods. There are different patterns for

formula pricing and neural net pricing models. For the formulaic pricing results, the error is usually higher for short-maturity at-the-money option, and is better controlled when the option is deep-in or out of the money. The error usually increases with maturity. This could be partly explained with the nature of optimization where highly-valued options are naturally weighted more. Therefore deep-in-the-money option with short maturity (which is most highly valued) has the least approximation error. What's counter-intuitive is well-controlled minor error when option is deep-out-of-the-money. Such options are cheap in value and therefore not highly weighted in the optimization formulation. But the prediction error at this end is actually the best. This could be due to the pricing nature of Heston formula, which varies little with parametric change at low moneyness and maturity. Therefore the overall error could be controlled. The percentage pattern is similar to some extent. It's worth noticing that the percentage pricing error is still minor for cheapest options. The trend for network pricing model shares some similarity but also has several differences. The absolute error panel is less informative with no significant pattern. However, the percentage data shows certain trend. It is monotonous with respect to moneyness within each maturity class. Especially for short maturity options, the percentage error increases explosively to over 100%. This fact shows that, different from the approach of pricing formula where pricing error is on proportional to option price, the error is approximately on the same scale for all options with the network pricing model. Therefore we can witness the surprisingly large percentage error for cheaply-priced options.

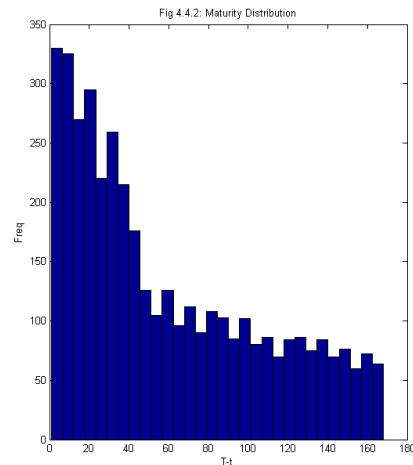
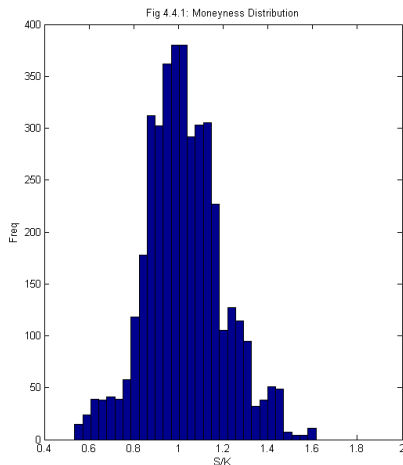


Table 4.4.4: Sample Size

	Short	Medium	Long
$S/K \geq 1.20$	271	249	65
$1.10 \leq S/K < 1.20$	297	218	151
$1 \leq S/K < 1.10$	409	327	184
$0.90 \leq S/K < 1$	465	351	185
$0.80 \leq S/K < 0.90$	318	249	44
$S/K \leq 0.80$	154	113	0

Table 4.4.5: Out-sample Prediction Absolute Error and Percentage Error on Subsets

		Short	Medium	Long	Short	Medium	Long
MLP Prediction Trained with Lagged iv.	S/K>=1.20	0.0623	0.0355	0.1799	0.44%	0.24%	1.03%
	1.10<=S/K<1.20	0.0306	0.0220	0.0420	0.46%	0.26%	0.42%
	1<=S/K<1.10	0.0470	0.0344	0.0340	2.39%	0.65%	0.47%
	0.90<=S/K<1	0.0559	0.0357	0.0360	8.90%	1.31%	0.80%
	0.80<=S/K<0.90	0.0344	0.0516	0.0416	107.78%	6.49%	1.64%
	S/K<=0.80	0.0326	0.0257	NaN	130.74%	77.08%	NaN
MLP Prediction Trained with Lagged riv.	S/K>=1.20	0.0437	0.0363	0.2030	0.29%	0.23%	1.14%
	1.10<=S/K<1.20	0.0337	0.0242	0.0631	0.50%	0.29%	0.64%
	1<=S/K<1.10	0.0566	0.0278	0.0467	3.04%	0.52%	0.65%
	0.90<=S/K<1	0.0513	0.0350	0.0335	11.88%	1.41%	0.73%
	0.80<=S/K<0.90	0.0421	0.0394	0.0340	180.84%	4.44%	1.33%
	S/K<=0.80	0.0209	0.0126	NaN	89.73%	21.56%	NaN
Heston Pricing with Global Parameter Optimized on iv.	S/K>=1.20	0.0018	0.0176	0.0426	0.01%	0.14%	0.29%
	1.10<=S/K<1.20	0.0079	0.0316	0.0602	0.12%	0.37%	0.60%
	1<=S/K<1.10	0.0170	0.0353	0.0614	0.55%	0.64%	0.86%
	0.90<=S/K<1	0.0184	0.0346	0.0563	2.48%	1.22%	1.23%
	0.80<=S/K<0.90	0.0070	0.0251	0.0396	5.06%	2.47%	1.54%
	S/K<=0.80	0.0008	0.0032	NaN	0.51%	3.34%	NaN
Heston Pricing with Global Parameter Optimized on riv.	S/K>=1.20	0.0017	0.0091	0.0151	0.01%	0.07%	0.10%
	1.10<=S/K<1.20	0.0079	0.0199	0.0205	0.12%	0.24%	0.20%
	1<=S/K<1.10	0.0170	0.0249	0.0251	0.55%	0.47%	0.35%
	0.90<=S/K<1	0.0180	0.0272	0.0257	2.29%	1.00%	0.57%
	0.80<=S/K<0.90	0.0068	0.0204	0.0258	5.03%	2.24%	1.03%
	S/K<=0.80	0.0007	0.0036	NaN	0.70%	3.87%	NaN
Heston Pricing with Locally Optimized Parameter	S/K>=1.20	0.0017	0.0094	0.0151	0.01%	0.07%	0.10%
	1.10<=S/K<1.20	0.0079	0.0203	0.0205	0.12%	0.24%	0.20%
	1<=S/K<1.10	0.0168	0.0252	0.0252	0.55%	0.47%	0.35%
	0.90<=S/K<1	0.0178	0.0277	0.0257	2.27%	1.03%	0.57%
	0.80<=S/K<0.90	0.0068	0.0204	0.0252	5.54%	2.23%	1.01%
	S/K<=0.80	0.0007	0.0033	NaN	1.37%	3.91%	NaN
Heston Pricing with Volatility Hint	S/K>=1.20	0.0000	0.0004	0.0009	0.00%	0.00%	0.01%
	1.10<=S/K<1.20	0.0003	0.0013	0.0017	0.00%	0.02%	0.02%
	1<=S/K<1.10	0.0005	0.0014	0.0015	0.01%	0.03%	0.02%
	0.90<=S/K<1	0.0003	0.0006	0.0005	0.05%	0.02%	0.01%
	0.80<=S/K<0.90	0.0005	0.0013	0.0010	0.48%	0.20%	0.05%
	S/K<=0.80	0.0001	0.0008	NaN	0.35%	1.13%	NaN

4.5 Dynamic Hedging Strategy Formulation and Simulation: Stochastic Volatility

The scenario could be different under stochastic volatility scheme. Firstly, the hedging strategy is more complicated from the formation of stochastic volatility hedging. Different from the constant volatility case, where we can achieve theoretical perfect hedging by holding only bond and one stock (underlying asset), the hedging with stochastic volatility requires an additional asset (one option) to hedge against the change in volatility. Suppose we want to hedge an option V with underlying stock S and another option U written on the same asset with different expiration and strike price. We form portfolio by longing V and shorting respective shares of S and U :

$$\Pi = V + \Delta S + \varphi U .$$

Then the change in portfolio value is formulated as:

$$d\Pi = dV + \Delta dS + \varphi dU .$$

We would like the change to be zero against change in asset price:

$$\frac{\partial \Pi}{\partial S} = \frac{\partial V}{\partial S} + \Delta + \varphi \frac{\partial U}{\partial S} = 0$$

and change in volatility:

$$\frac{\partial \Pi}{\partial v} = \frac{\partial V}{\partial v} + \varphi \frac{\partial U}{\partial v} = 0$$

Solving the two equations yields:

$$\varphi = - \frac{\partial V / \partial v}{\partial U / \partial v}$$

$$\Delta = - \frac{\partial V}{\partial S} - \varphi \frac{\partial U}{\partial S}$$

This gives the same result as previous derivation with Ito's Lemma. In practice, a simplified approach is to hedge only against price change. Therefore, we have the same scenarios as Black-Scholes hedging. In the next context, I will implement two different types of hedging approach.

The hedging strategy can be formulated as done before, but there are differences in strategic formulation, especially for the delta-hedging case. Also the discussion and application will be based on the previously proposed methods of parameter estimation, which has been assessed in detail in previous section.

For the most part of analysis here, I will inherit the dynamic hedging formulation and notation that has been applied to constant volatility case. I will employ two types of hedging strategy. The first is minimum-variance hedge, which implement only the underlying asset as our hedging instrument. It is biased with the stochastic volatility case, where two sources of risk are in existence: price and volatility. Therefore, we give the second hedging strategy: delta-neutral hedge. It will implement additional shares of options to hedge against volatility risk.

4.5.1 Minimum-Variance Hedging

In this first part, let's consider the simplified hedging with only one asset and one bond to hedge price risk, and discard the volatility risk since the uncertainty of changes in option price with regard to volatility cannot be hedged merely with underlying asset. Although this approach may look inferior here, but the issue of transaction cost and the factor of model misspecification make it reliable in practice. We still use $V_O(t), V_S(t), V_B(t)$ to denote the value of option, underlying stock and bond at time t , where $V_S(t) = X_S(t)S(t), V_O(t) = -C(t)$ (suppose we short one position of option). Then $V(t) = V_O(t) + V_S(t) + V_B(t)$ denote the dollar value of replicating portfolio. Solving the minimum-variance hedging problem $\min_{X_S(t)} \text{var}[-C(t) + X_S(t)S(t)]$ will give the share of underlying assets to hold as:

$$X_S(t) = \Delta_S(t) + \frac{\rho\sigma\Delta_V(t)}{S}$$

Now we just need to specify the corresponding delta values. In the constant volatility case, I only need to estimate volatility to calculate delta term to perform parametric hedging, and no estimation is required in the nonparametric method. Now I need to make estimation of all parameters to make parametric hedging, and need to estimate instantaneous volatility for nonparametric hedging. I denote the two hedging ratios as

$$\Delta_S^{HS} = \frac{\partial f_{HS}(T-t, S(t), K, \hat{\Theta}, \hat{v}(t))}{\partial S}, \Delta_V^{HS} = \frac{\partial f_{HS}(T-t, S(t), K, \hat{\Theta}, \hat{v}(t))}{\partial v} \text{ and}$$

$$\Delta_S^{MLP} = \frac{\partial f_{MLP}(T-t, S(t), k, \hat{v}(t))}{\partial S}, \Delta_V^{MLP} = \frac{\partial f_{MLP}(T-t, S(t), k, \hat{v}(t))}{\partial v}$$

respectively. In this representation, f_{HS} and f_{MLP} stands for the theoretical Heston pricing formula and trained network for pricing respectively. We have closed-end formula for the Heston Delta and Vega (see **Appendix II** for closed-end formula), and finite difference method is also applicable to derive the two Greeks. For the neural net approximation, a finite-difference method is more convenient to apply, written as:

$$\Delta_S^{MLP}(t) = \frac{f_{MLP}(S(t) + \varepsilon) - f_{MLP}(S(t))}{\varepsilon}.$$

The hedging problem formulation is based on continuous time assumption but only discrete time hedging is achievable in reality. Let's consider a discrete time hedging with a time interval of Δt . Suppose the option to hedge has a life of $T = M\Delta t$. Now I begin to formulate the entire procedure of dynamic minimum-variance hedging on discrete time spot $t = k\Delta t, k = 1, \dots, M$.

Starting from $t=0$, we short one share of option, long $S(0)$ shares of underlying asset, and

purchase/borrow short-term bond to fill up the rest. We have a replicating portfolio with zero initial value.

$$\begin{aligned} V_C(0) &= -f_{HS}(0) \\ V_S(0) &= S(0)X_s^{HS/MLP}(0) \\ V_B(0) &= -V_S(0) - V_C(0) \end{aligned}$$

At time $t - \Delta t$, we make an estimation of Heston parameters $\hat{\Theta}_{t-\Delta t}$ and instantaneous volatility $\hat{v}(t - \Delta t)$. Then at time t we rebalance the account as:

$$\begin{aligned} V_C(t) &= -f_{BS}(t) \\ V_S(t) &= S(t)X_s^{HS}(t, \hat{\Theta}_{t-\Delta t}, \hat{v}(t - \Delta t)) \text{ or } S(t)X_s^{MLP}(t, \hat{v}(t - \Delta t)). \\ V_B(t) &= e^{r\Delta t}V_B(t - \Delta t) - S(t)(X_s^{HS/MLP}(t) - X_s^{HS/MLP}(t - \Delta t)) \end{aligned}$$

Basically, I rebalance the stock account according to the Greeks estimated with previous day's estimated parameter. And the risk-free asset is rebalanced accordingly. The strategy shall be self-financing. The same set of indicators as used in 4.2 can be used here to evaluate tracking error.

Then the tracking error prior to next rebalance can be calculated as:

$$H(t + \Delta t) = X_S(t)S(t + \Delta t) + e^{r\Delta t}V_B(t) + V_C(t + \Delta t)$$

4.5.2 Delta-neutral Hedge

Now consider multiple assets to hedge the option. In this case, we can control the risk engendered by not only underlying asset's price movement but also volatility change. In order to hedge against volatility risk, an additional option with different expiration or strike is implemented. Thus our replicating portfolio contains one short position in option, and long positions in underlying stock, additional option and risk-free bond. We have the following formulation:

$$\begin{aligned} V(t) &= V_o(t) + V_s(t) + V_B(t) + V_C(t) \\ &= -C(t, T, K) + X_S(t)S(t) + V_B(t) + X_C(t)C(t, T, \bar{K}) \end{aligned}$$

Recall the formerly-derived hedging ratio, and we have:

$$\begin{aligned} X_C(t) &= \frac{\Delta_V(t, T, K)}{\Delta_V(t, T, \bar{K})} \\ X_S(t) &= \Delta_S(t, T, K) - \Delta_S(t, T, \bar{K})X_C(t) \end{aligned}$$

The Greeks here can be calculated as before. Then I do the same formulation from 0 to T. At $t=0$:

$$\begin{aligned} V_o(0) &= -f_{BS}(0, T, K) \\ V_C(0) &= f_{HS}(0, T, \bar{K})X_C^{HS/MLP}(0) \\ V_S(0) &= S(0)X_s^{HS/MLP}(0) \\ V_B(0) &= -V_o(0) - V_C(0) - V_S(0) \end{aligned}$$

At time $t - \Delta t$, we make an estimation of Heston's instantaneous volatility. For local optimization approach, we need to estimate other parameters at the same time. And then I rebalance the account at time t as:

$$\begin{aligned} V_O(t) &= -f_{HS}(t, T, K) \\ V_C(t) &= f_{HS}(t, T, \bar{K}) X_C^{HS/MLP}(t, \hat{\Theta}_{t-\Delta t}, \hat{v}(t - \Delta t)) \\ V_S(t) &= S(t) X_S^{HS/MLP}(t, \hat{\Theta}_{t-\Delta t}, \hat{v}(t - \Delta t)) \\ V_B(t) &= e^{r\Delta t} V_B(t - \Delta t) - S(t)(X_S(t) - X_S(t - \Delta t)) - f_{HS}(t, T, \bar{K})(X_C(t) - X_C(t - \Delta t)) \end{aligned}$$

Again, we look up to the terminal value of replicating portfolio for the tracking error.

4.6 Results

4.6.1 Results from Constant Volatility Model

I compare RBF, MLP with BS through a dynamic hedging formulation in a constant volatility case. First, RBF and MLP nets are trained on simulated dataset as before, and then independent paths are generated to run the delta hedging strategy. 100 paths will be generated within different moneyness-maturity category, and the performance metrics of hedging errors will be calculated with the paths in each category. The mean tracking error is one metric I'm concerned with. In **Table 4.6.1.1** I briefly report the results with respect to each category. I omit RBF in this table since its results are much inferior to other methods. I provide only the result of mean terminal tracking error.

We can find that MLP prediction results are inferior to BS within all combinations of moneyness and maturity. In some subsets the values are close (at or around-the-money options with short-maturity), but as the price deviates from the strike or maturity gets longer, we can observe significantly poorer performance by MLP. This pattern is similar to the results in next section, and I'll try to give some explanations.

Table 4.6.1.1: Dynamic Hedging Error, GBM

S/K	Model	Maturity(T)						
		0.25	0.5	0.75	1	1.25	1.5	1.75
0.8	BS	0.005	0.040	0.014	0.042	0.094	0.040	0.057
	MLP	0.090	0.543	0.990	1.561	2.615	4.244	2.610
0.9	BS	0.082	0.097	0.053	0.115	0.101	0.124	0.075
	MLP	0.102	0.211	1.410	1.152	1.324	2.146	3.913
1	BS	0.189	0.133	0.156	0.164	0.132	0.110	0.163
	MLP	0.189	0.149	0.945	1.170	0.762	2.132	2.639
1.1	BS	0.089	0.194	0.258	0.134	0.099	0.199	0.157
	MLP	0.130	0.225	0.266	0.261	0.532	1.987	3.667
1.2	BS	0.043	0.088	0.180	0.163	0.175	0.188	0.236
	MLP	0.081	0.111	0.513	0.287	0.682	2.685	13.268

4.6.2 Results from Stochastic Volatility Model

I will provide results from simulations of dynamic hedging strategies under stochastic volatility assumption. I will try different combinations of moneyness and maturity in the simulation. Also different hedging approaches are compared. I have applied the parametric method with pricing formula, the nonparametric approach with multiple-layer perceptron. For both methods I will use refined volatility estimation to form hedging. There are several steps with the simulation. First, I prepare the models for hedging. I generate two-years' option pricing data as training dataset. Global parameters are learnt from the optimization over dataset based on refined volatility estimation. Neural network is trained on the same dataset with lagged data of refined volatility estimation. Models are trained on a two-year option price sequence. Then I generate an additional option price sequence according to the specific moneyness and maturity. Dynamic hedging is run over the life of this option. I will replicate the hedging process of this specific moneyness and maturity for 50 times, and calculate the expectation of terminal error as one of the indicator of hedging accuracy. I will also provide values of other meaningful indicators. In addition to the parametric and nonparametric pricing, I give the result of hedging from Black-Scholes formula. This result should be indicative of the case where the underlying pricing process is misspecified. A detailed review of the results is given in the following tables, together with diagrams illustrating the hedging process.

Fig 4.6.1-2 gives two typical illustrations of hedging process. In each figure, the red curve indicates the value of option until its expiration. In both cases, the value converges to zero, meaning the terminal stock price is lower than strike. The blue and green curves indicate the value of self-financing replicating portfolio to track the option. We can see that their value evolves over time in pretty similar trend as option price, and the terminal error is small enough. In the first figure, the option is at-the-money with half year to expiration at the initial date. We can see the MLP method fits the trend better. In the second figure, the option is deep in the money with 1.75 years to expiration. The fitting is quite stable over the long term, and tracking error is well controlled. I also add the Black-Scholes hedging result to the figure (the black one). Obviously, it generates much larger error than the other two approaches.

Table 4.6.2.1 provides result with respect to expected terminal error of each moneyness-maturity category. Row names are assigned as moneyness, and column names are assigned to be maturity. The moneyness varies from 0.8 to 1.2 with a step of 0.1. The maturity varies from 0.25 to 1.75 with a step of 0.25. Within each moneyness-maturity, results from three models are provided. The first is Heston pricing formula with globally estimated parameters. The second is nonparametric pricing method with multiple-layer perceptron, trained with refined volatility data. The third is Black-Scholes formula. This is applied here to indicate misspecification error. **Table 4.6.2.2** provides statistics of prediction error that accounts for variance in result as proposed earlier, and **Table 4.6.2.3** provides statistics of summed error over the hedging period. The MatLab code I wrote to derive dynamic hedging error is given in **Appendix III**.

First we look at error in terminal value in **Table 4.6.2.1**. The error is larger for longer-maturity. It also seems to be larger for at-the-money options. The trend is almost homogeneous to all three

models. We are more concerned with the relative performance over three models. I have highlighted the observations where MLP outperforms Heston. There are some very interesting results in the table. We can see MLP outperforms SVM for all at-the-money options, and performs less well as moneyness deviates from 1. I try to offer some explanations for this effect. MLP is a data-driven method, and its performance depends to a great extent on the training data size. As is usually the case, from a randomly generated sample path, a greater portion of options assigned are at- or around- the money, especially when underlying asset's pricing process has a small mean or standard deviation. This usually leads to a larger sample size of at-the-money options in the training dataset. For those options that are deep in or out of the money, the size of data is not sufficient to provide sound training of MLP model. Therefore we can see it outperformed by Heston model. Still, results from this table show quite promising predictive power of the data-driven method, which can do better than ideal pricing formula that the asset price sequence follows exactly. Black-Scholes data here cannot offer many implications, since it is dominated by the two methods over all the categories. This can only show that the application of Black-Scholes is rather restricted with the condition of heteroscedasticity.

We can also make comparison between the relative performance under GBM and SVM assumptions. Recall that the data-driven method is almost surely dominated by Black-Scholes formula in 4.6.1. But here it can outperform the pricing formula method on most occasions. The difference in parameter estimation shall account for this difference. In the constant volatility case, we only have one parameter of volatility constant to estimate. This is easily achieved based on adequate historical data, and since volatility remains constant, this estimation remains to be right almost surely. With stochastic volatility model, however, we have to estimate the whole dynamics of volatility, which makes an accurate estimation much harder to achieve. And that's why I have devoted a whole section to discuss parameter estimation approach with stochastic volatility model. With the difficulty to make parameter estimation for formulaic pricing model, the data-driven model gains its preference. It's natural to have a result with better performance in MLP.

As we refer to Table 4.6.2.2 and Table 4.6.2.3, the implications are similar. We have better performance with MLP for at-the-money observations. The result is the same as we account for the variance in the output. We only need to notice that the performance has a greater preference over MLP when we account for the summed error. This indicates that MLP is more consistent in the tracking.

And finally, we can make a comparison between MLP results with constant and stochastic volatility assumption. The performance with stochastic volatility model is significantly better than the other case, and this indicates the advantage of data-driven method in recognizing more complicated patterns compared over the parametric pricing model which fail to learn parameter perfectly in a complicated case.

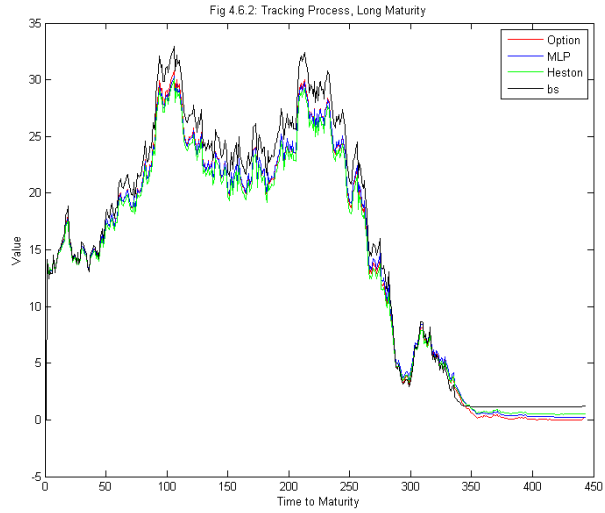
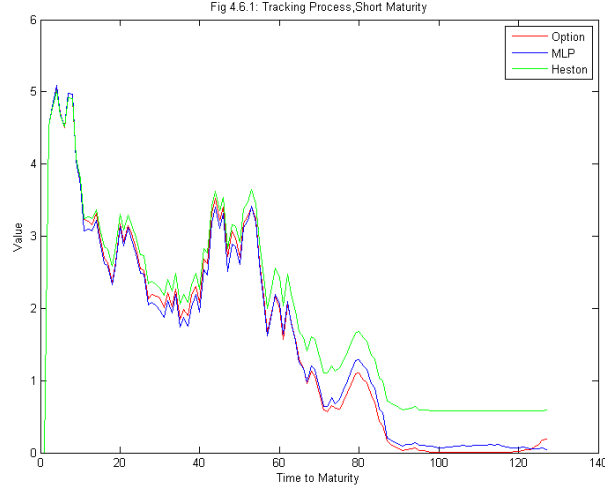


Table 4.6.2.1: Dynamic Hedging Error I, SVM

S/K	Model	Maturity(T)						
		0.25	0.5	0.75	1	1.25	1.5	1.75
0.8	HS	0.282	0.365	0.441	0.527	0.808	0.892	1.164
	MLP	0.309	0.406	0.417	0.511	0.637	0.546	1.226
	BS	0.376	0.972	1.912	2.615	2.783	2.565	3.438
0.9	HS	0.342	0.392	0.553	0.728	0.818	0.823	1.117
	MLP	0.390	0.362	0.425	0.520	0.654	0.648	1.063
	BS	1.060	1.724	2.163	2.502	1.985	2.335	3.248
1	HS	0.366	0.459	0.584	0.596	0.738	0.780	1.185
	MLP	0.308	0.348	0.378	0.461	0.738	0.715	0.664
	BS	1.085	1.517	2.037	1.969	1.948	2.957	2.394
1.1	HS	0.276	0.286	0.507	0.528	0.627	0.721	0.793
	MLP	0.284	0.298	0.429	0.531	0.641	0.799	0.777
	BS	0.773	1.350	1.484	1.855	2.351	2.239	2.432
1.2	HS	0.246	0.257	0.273	0.488	0.456	0.524	0.791

	MLP	0.226	0.303	0.426	0.368	0.563	1.458	0.715
	BS	0.305	0.864	1.394	1.578	1.572	1.608	1.826

Table 4.6.2.2: Dynamic Hedging Error II, SVM

S/K	Model	Maturity(T)						
		0.25	0.5	0.75	1	1.25	1.5	1.75
0.8	HS	0.354	0.517	0.609	0.654	0.917	1.058	1.232
	MLP	0.392	0.564	0.522	0.700	0.786	0.611	1.289
	BS	0.584	1.288	2.294	2.917	3.343	2.802	4.053
0.9	HS	0.473	0.456	0.692	0.826	0.935	0.930	1.236
	MLP	0.492	0.421	0.554	0.609	0.748	0.747	1.217
	BS	1.293	2.016	2.511	2.947	2.304	2.744	3.481
1	HS	0.431	0.537	0.716	0.691	0.833	0.835	1.151
	MLP	0.368	0.444	0.517	0.529	1.549	0.800	0.794
	BS	1.286	1.749	2.412	2.372	2.299	3.408	2.585
1.1	HS	0.326	0.364	0.569	0.615	0.702	0.765	0.842
	MLP	0.350	0.345	0.524	0.653	0.766	1.058	0.861
	BS	0.840	1.585	1.773	2.123	2.540	2.595	2.750
1.2	HS	0.264	0.337	0.333	0.553	0.499	0.590	0.823
	MLP	0.253	0.387	0.532	0.446	0.715	3.978	0.918
	BS	0.370	1.290	1.575	1.832	1.634	1.840	1.934

Table 4.6.2.3: Dynamic Hedging Error III, SVM

S/K	Model	Maturity(T)						
		0.25	0.5	0.75	1	1.25	1.5	1.75
0.8	HS	0.064	0.178	0.261	0.353	0.464	0.635	0.794
	MLP	0.072	0.167	0.244	0.312	0.408	0.406	0.892
	BS	0.199	0.802	1.410	1.896	2.055	2.349	2.817
0.9	HS	0.185	0.228	0.377	0.405	0.594	0.560	0.767
	MLP	0.160	0.195	0.254	0.252	0.452	0.540	0.837
	BS	0.776	1.281	1.642	1.708	1.699	1.707	2.359
1	HS	0.191	0.262	0.393	0.401	0.473	0.593	0.776
	MLP	0.133	0.195	0.212	0.264	0.362	0.502	0.494
	BS	0.808	1.030	1.450	1.548	1.483	2.029	1.557
1.1	HS	0.146	0.193	0.287	0.382	0.448	0.496	0.579
	MLP	0.135	0.126	0.220	0.309	0.356	0.329	0.436
	BS	0.586	0.974	1.179	1.483	1.684	1.608	1.788
1.2	HS	0.064	0.136	0.206	0.322	0.376	0.418	0.533
	MLP	0.071	0.135	0.255	0.291	0.270	0.438	0.470
	BS	0.220	0.653	1.023	1.145	1.244	1.331	1.415

5. Empirical Results

In this chapter, I will apply the previous methods to the hedging of 50ETF options in Shanghai Stock Exchange. The Chinese Option Market is initiated on Feb 9th. Options written on 50 ETF (510050.OF) are the first to be publically traded. I access the first three months' trading data of the options, and try to form dynamic hedging with previously-discussed method. Compared with overseas derivative markets, option market in China certainly lacks liquidity at this very beginning, and most institutions are speculating. This illiquidity will lead to a misspecification of pricing formula, which may seem a preference over our method. On the other hand, the lack of trading data would also deteriorate the model performance. We shall see in the following context how the models perform in this empirical case.

5.1 Data Description

The data collected covers 50 ETF options from 2015.02.09 to 2015.05.14. There are 204 option contracts written over this period in total. I will focus on call options in the study. In this case, there are 102 option contracts with an average of 49.6 contracts per trading day. And there are 3076 observations (one trading day's data for a specific option) in total. Each option contract is accompanied with daily close price, underlying asset close price, time to maturity and strike price. To calculate time to maturity, I account for the number of trading days before date of maturity. This amount is divided by 244(number of trading days in 2015) to be scaled into year. Besides these, I still need to estimate volatility. I have given several methods for this in previous chapters. Here I can use either the standard deviation close-to-date returns, or implied volatility from option contracts. The previous one is backward looking and the latter one is forward looking. To give close-to-date estimation, we can use last thirty days' return data. To give implied volatility, there can be more choices. There are many option contracts each day, and they give distinct implied volatility, possibly forming a volatility smile. A sensible way is to choose that value of at-the-money close-to-maturity option. I select the closest-to-money options, than find the one with shortest maturity among them. The implied volatility by this option is implemented as the volatility estimated for that trading day. Notice that we can only have access to last trading day's data on each day, therefore, the volatility estimation is lagged for one day. Also there is the need for interest rate data. I implement a very conventional approach by applying 1-year SHIBOR rate.

The integrated data is separated into two sections on the timeline. The first is the set for training and in-sample fitting, and the latter is for out-sample fitting and dynamic hedging checking. Data from Feb 9th to April 22nd is used for training, with 2086 observations in total. The rest from April 23rd to May 14th are applied for testing, with 990 observations in total.

5.2 Out-sample Prediction Error

Four methods are applied to the pricing and hedging. Firstly we have two parametric methods: Black-Scholes pricing and Heston Pricing. Then come the nonparametric methods of multiple layer perceptron of Heston and Black-Scholes model. The difference lies in the number of inputs. Heston MLP has a volatility input while Black-Scholes MLP doesn't require.

The nonparametric models are trained on the training dataset. All four models are used to price options in test set. **Table 6.2** gives the results. We can see that Heston MLP has the best performance, followed by Black-Scholes pricing and Black-Scholes MLP. It shows how a data-driven method can beat pricing formula with proper training on sufficient related features. And we can also see that volatility does contain important information for pricing, since a MLP model without volatility input has only inferior performance to the one with it. Also, The fact that Heston formula cannot outperform simple Black-Scholes model is surprising. One possible explanation is that the complicity of the model might lead to overfitting of in-sample data, and therefore the out-sample performance is inferior. Also the inaccuracy in parameter estimation may harm the prediction accuracy of the model.

	SSE	MSE	SST	SSR	RSQUARED
MLP_HS	0.096	0.000	13.138	12.651	0.993
BS	0.102	0.000	13.138	13.661	0.992
MLP_BS	0.394	0.000	13.138	14.134	0.970
HS	2.348	0.002	13.138	15.708	0.821

5.3 Dynamic Hedging Error

I also use the previously formed hedging strategy to form dynamic hedging to compare tracking error performance. There are 62 option contracts in the test dataset. Some of them are not yet expired at the end of the time period, and I will hedge them over the active period. Also some of the options have too short a life to form an insightful result of hedging, therefore I will drop them.

Table 6.3 provides results from dynamic hedging strategy over all option contracts. The table gives result on different categories of data. The left column indicates the type. We can see that data-driven methods has a better performance overall. However, MLP1 (with volatility input) and MLP2 (without volatility input) don't have much difference worth noticing. The result indicates that data-driven method has an improvement for hedging result, but the adjustment with volatility information doesn't further improve the result. We can further look at the performance at some extreme cases, listed from row 2 to row 5. We can see the performance worsens for the extreme cases. This is consistent with the previous assumption that data size matters prediction results for MLP method. A key drawback to our approach is the lack of data and its diversity. We'd expect an improvement in data-driven methods with the expansion of data size.

TYPE	BS	MLP1	MLP2
Overall	0.032	0.027	0.026
Deep in	0.040	0.054	0.045
Deep out	0.029	0.032	0.038
Long	0.030	0.036	0.038
Short	0.027	0.027	0.023

6. Conclusion

The thesis mainly introduces and validates the application of a specific data-driven method to the pricing and hedging of derivatives, with an emphasis on European-style options under stochastic volatility process. Data are simulated according to GBM and SVM assumptions, and option contracts are designed with the rule of CBOE. In-sample error is measured with cross-validation set, and I offer some details and implications with statistical learning's perspective. To measure out-sample error, I mainly study the two metrics for performance analytics. The first one is the static measure of out-sample prediction error. The second is the tracking error of dynamic hedging strategy. In order to develop a reasonable setting in dynamic hedging, I introduce an optimization scenario for parameter estimation for stochastic volatility model, and offer a refined estimation of instantaneous volatility. For the dynamic hedging strategy itself, I provide two settings of hedging for SVM. The first is minimum variance hedging which is partial to volatility risk. The second is delta-neutral hedging which can perfectly hedge all the exposures to risk. I provide results for different moneyness-maturity categories, and make analysis for the results. I account for the difference between parametric and nonparametric methods with respect to hedging performance in both constant and stochastic volatility case.

In the constant volatility case, the nonparametric approach is inferior to B-S pricing with respect to both prediction error and dynamic hedging error. In the stochastic volatility case, I propose a multiple-layer perceptron model trained on the refined instantaneous volatility estimation input. The approach achieves better performance in dynamic hedging than the parametric pricing model in most of moneyness-maturity categories. This result is amazing since the prices are simulated strictly according to Heston pricing formula. The difficulty in parameter estimation makes the parametric approach less favorable to data-driven method, which requires less information about parameters. While in the constant volatility case, there is no such difficulty of estimation, therefore the data-driven method is dominated by parametric method. Also, I expect this advantage of data-driven method to be magnified in empirical data which doesn't necessarily evolve according to pricing formulas, and may go through structural changes in parameters. The results from each category indicate that MLP method is likely to have better performance for at-the-money options, and I propose a possible explanation to this interesting phenomenon from the perspective of training-data size.

The study with respect to this topic doesn't merely end up here. More interesting facts and implications can be found in further studies. I offer some ideas that may initiate further researches. The first is a refinement of methodologies. I have applied the most typical RBF and MLP, but there have been tremendous development in machine learning, and there could be more powerful tools that can facilitate the work. One of the most popular choices is the deep-learning net. Also there can be extensions in the model, such as one with structural changes in parameters. There can also be more to study with empirical data, like a comparison across markets. All these topics are of intriguing interest to carry out, and I will possibly follow them to further the study.

REFERENCE

- [1] Hutchinson, J., Lo, A. and Poggio, T. (1994) A nonparametric approach to pricing and hedging derivative securities via learning networks, *The Journal of Finance*, July 1994
- [2] Hastie, T., Tibshirani, R. and Friedman J. (2009) The Elements of Statistical Learning, second edition, *Springer Series in Statistics*
- [3] Cox, J., Ingersoll, J. and Ross, S. (1985) A theory of the term structure of interest rates, *Econometrica*, 53: 385-407
- [4] Heston, S.(1993) A closed-form solution for options with stochastic volatility with applications to bond and currency options, *Review of Financial Studies*, 6, 2: 327-43
- [5] Rouah, F., Heston, S.(2013) The Heston Model and Its Extensions in MatLab and C#, *Wiley*
- [6] 姜礼尚(2008) 期权定价的数学模型和方法, 高等教育出版社, 37-39
- [7] Hayashi, T., Mykland, P.(2005) Evaluating Hedging Errors: An Asymptotic Approach, *Mathematical Finance*, April 2005
- [8] Bossaerts, P., Hillion, P.(1997) Local parametric analysis of hedging in discrete time, *Journal of Econometrics*, 81 (1997) 243-372.
- [9] Bossaerts, P., Hillion, P.(2003) Local parametric analysis of derivative pricing and hedging. *Journal of Financial Markets*, 6 (2003):573-605.
- [10] Bakshi, G., Cao, C., and Z. Chen. (1997). Empirical Performance of Alternative Option Pricing Models. *Journal of Finance*, 52(5):2033–49.
- [11] Ait-Sahalia, Y., and R. Kimmel. (2007). “Maximum Likelihood Estimation of Stochastic Volatility Models.” *Journal of Financial Economics*, 83:413–52.
- [12] Gauthier, P., and D. Possamaï. (2010). “Efficient Simulation of the Double Heston Model.” Working Paper, Pricing Partners (www.pricingpartners.com).
- [13] Benhamou, E., Gobet, E., and M. Miri. (2010). Time Dependent Heston Model. *SIAM Journal on Financial Mathematics*, 1:289–325.

APPENDIX

Appendix I: Closed-end Heston formula and MatLab Code

Consider a European call option written on an underlying stock $S(t)$ with time-to-expiration $T-t$ and strike price K . The asset price follows stochastic volatility process with parameter $\kappa, \theta, \sigma, \rho$. Denote instantaneous volatility $v(t)$. Then the option price at time t $C(t, S(t), v(t))$ must solve:

$$\frac{1}{2} v S^2 \frac{\partial^2 C}{\partial S^2} + r S \frac{\partial C}{\partial S} + \rho \sigma v S \frac{\partial^2 C}{\partial S \partial V} + \frac{1}{2} \sigma^2 v \frac{\partial^2 C}{\partial V^2} + [\theta - \kappa v] \frac{\partial C}{\partial V} + \frac{\partial C}{\partial t} - r C = 0$$

subject to:

$$C(T) = \max \{ S(T) - K, 0 \}$$

The closed-end formulation of call price is given as:

$$C(t) = S(t) \Pi_1(t; S, V, T) - K e^{-r(T-t)} \Pi_2(t; S, V, T)$$

where:

$$\Pi_j(t; S(t), V(t), T) = \frac{1}{2} + \frac{1}{\pi} \int_0^\infty \operatorname{Re} \left[\frac{e^{-i\phi \ln(K)} f_j(t, S(t), V(t); \phi, T)}{i\phi} \right] d\phi$$

$$f_1(t; \phi, T) = \exp \left\{ -\frac{\theta}{\sigma^2} \left[2 \ln \left(1 - \frac{[\xi - \kappa + (1+i\phi)\rho\sigma](1-e^{-\xi(T-t)})}{2\xi} \right) \right] \right. \\ \left. - \frac{\theta}{\sigma^2} [\xi - \kappa + (1+i\phi)\rho\sigma](T-t) + i\phi \ln[S(t)] \right. \\ \left. + \frac{i\phi(1+i\phi)(1-e^{-\xi(T-t)})}{2\xi - [\xi - \kappa + (1+i\phi)\rho\sigma](1-e^{-\xi(T-t)})} V(t) \right\}$$

$$f_2(t; \phi, T) = \exp \left\{ -\frac{\theta}{\sigma^2} \left[2 \ln \left(1 - \frac{[\xi^* - \kappa + i\phi\rho\sigma](1-e^{-\xi^*(T-t)})}{2\xi^*} \right) \right] + [\xi^* - \kappa + i\phi\rho\sigma](T-t) \right. \\ \left. + i\phi \ln[S(t)] - \ln[e^{-r(T-t)}] \right. \\ \left. + \frac{i\phi(i\phi-1)(1-e^{-\xi^*(T-t)})}{2\xi^* - [\xi^* - \kappa + (1+i\phi)\rho\sigma](1-e^{-\xi^*(T-t)})} V(t) \right\}$$

where: $\xi = \sqrt{[\kappa - (1+i\phi)\rho\sigma]^2 - i\phi(i\phi+1)\sigma^2}$, $\xi^* = \sqrt{[\kappa - i\phi\rho\sigma]^2 - i\phi(i\phi-1)\sigma^2}$

The following is the MatLab implementation of this pricing formula. The integration is dealt with numerically.

```
function fj=CF_SVj(xt,vt,tau,mu,a,uj,bj,rho,sig,phi)
%PURPOSE: implements the CF fj of Heston. Uses Heston's notations.
xj = bj - rho.*sig.*phi.*i;
dj = sqrt( xj.^2 - (sig.^2).*( 2.*uj.*phi.*i - phi.^2 ) );
gj = ( xj+dj )./( xj-dj );
D = ( xj+dj )./(sig.^2).*( 1-exp(dj.*tau) )./( 1-gj.*exp(dj.*tau) );
xx = ( 1-gj.*exp(dj.*tau) )./( 1-gj );
C = mu.*phi.*i.*tau + a./( sig.^2 ) .* ( (xj+dj) .* tau - 2.*log(xx) );
fj = exp( C + D.*vt + i.*phi.*xt );
end
```

```
function C=HestonCall(St,K,r,T,vt,kap,th,sig,rho,lda)
%PURPOSE: Computes the option price using Heston's model.
%INPUT: St - scalar or vector, price of underlying at time t
%       K - scalar or vector, strike price
%       r - scalar or vector, continuously compound risk free rate expressed
as a positive decimal number.
%       sig- scalar or vector, volatility of the volatility of the
%       underlying(same time units as for r)
```

```

% T - scalar or vector, time to maturity (same time units as for r)
% vt - scalar or vector, instantaneous volatility
% kap - scalar or vector, is the rate at which vt reverts to th
% th - scalar or vector, is the long vol, or long run average price
% volatility; as t tends to infinity, the expected value of ?t tends
to ?
% lda - scalar or vector, risk premium for volatility
% rho - scalar or vector, correlation between underlying and
% volatility (rho<0 generates the leverage effect)
%-----
-----
%OUTPUT: C - scalar or vector, Heston's model call option price
%-----
-----
dphi=0.01;
maxphi=50;
phi=(eps:dphi:maxphi)';
f1 = CF_SVj(log(St),vt,T,0,kap*th,0.5,kap+lda-rho*sig,rho,sig,phi);
P1 = 0.5+(1/pi)*sum(real(exp(-i*phi*log(K)).*f1./(i*phi))*dphi);
f2 = CF_SVj(log(St),vt,T,0,kap*th,-0.5,kap+lda,rho,sig,phi);
P2 = 0.5+(1/pi)*sum(real(exp(-i*phi*log(K)).*f2./(i*phi))*dphi);
C = St*P1 -K*exp(-r*T)*P2;
end

```

Appendix II: Closed-end formula for Heston Delta and Vega

$$\Delta_S(t) = \frac{\partial C(t)}{\partial S} = \Pi_1 \geq 0$$

$$\Delta_V(t) = \frac{\partial C(t)}{\partial V} = S(t) \frac{\partial \Pi_1}{\partial V} - Ke^{-r(T-t)} \frac{\partial \Pi_2}{\partial V}$$

$$\text{where } \frac{\partial \Pi_j}{\partial V} = \frac{1}{\pi} \int_0^\infty \text{Re} \left[(i\phi)^{-1} e^{-i\phi \ln(K)} \frac{\partial f_j}{\partial V} \right] d\phi.$$

Appendix III: Function for Dynamic Hedging Error

```

function [E,F] =
SVHedgingStrat(T,money,net,netin,param,type,method,local,m)
%The function calculates the stochastic volatility hedging error from a
%dynamic hedging approach. It takes the following inputs:
%T: Time to maturity(in year)
%money: moneyness of the option at initial date(S0/K)
%net: Trained network input. If 0, will be automatically trained.
%param: Globally estimated parameters. If 0, will run local optimization.
%type: If 'iv', apply implied volatility; If 'riv', apply refined implied
%volatility.
%m: Number of simulations to run

```

```

%method: 1 for minimum variance, 2 for delta-neutral
%local: 1 for local optimization, 0 for global.
%% Define Global Params
if nargin<1, T=.5;money=1;net=0;type='riv';method=1;local=0;m=500;end
mu=.1;
n=253;
S0=50;
v0=.05;
kappa=5;
theta=.1;
sig=.2;
N=floor(n*T);
lambda=0;
r=.05;
rho=-.5;
eps=.0001;
%% Net Training
if netin==0
    load('Train3.mat');
    MLP=Data;
    clearvars Data;
    if strcmp(type,'ivr')
        param_ivr=param;
        MLP(:,7)=MLP(:,8);

MLP(:,8)=(-param_ivr(1)*15/n*MLP(:,8)+param_ivr(1)*param_ivr(2)*15/n)
/(exp(-param_ivr(1)*15/n)-1)+param_ivr(2);
        end
        MLP(MLP(:,5)==1,9)=nan;
        for k=2:length(MLP)
            MLP(MLP(:,5)==k,9)=mean(MLP(MLP(:,5)==(k-1),8));
        end

MLPTrain=[MLP(:,3)./MLP(:,1),MLP(:,2),MLP(:,9),MLP(:,6)./MLP(:,1)];%U
se lagged implied vol
MLPTrain=MLPTrain(MLP(:,5)<=2*n,:);
MLPTrain=MLPTrain(~isnan(MLPTrain(:,3)),:);
net=feedforwardnet([3 5 3]);
net=configure(net,MLPTrain(:,1:3)',MLPTrain(:,4)');
net.trainFcn='trainlm';
net=init(net);
net.performParam.regularization=0;
net.trainParam.epochs=100;
net.trainParam.max_fail=9;

```

```

[netnn, tr]=train(net,MLPTrain(:,1:3)',MLPTrain(:,4)');
plotperf(tr);

MLPSim=[MLP(:,3)./MLP(:,1),MLP(:,2),MLP(:,9),MLP(:,6)./MLP(:,1)];%Use
lagged refined iv to predict
MLPSim=MLPSim(MLP(:,5)>2*n,:);
MLPSimResult=sim(netnn,MLPSim(:,1:3)');
Ks=MLP(MLP(:,5)>2*n,1);
OSError=mean(abs(MLPSimResult-MLPSim(:,4)).*Ks);
disp(OSError);
else
    netnn=net;
end

%% Simulate Price for Hedging Test
%clearvars -except netnn S0 T mu v0 kappa theta lambda sig rho N money
param param_ivr r n m eps;
E=zeros(m,6);
for w=1:m
[S,V]=GenerateHSequence(S0,T,mu,v0,kappa,theta,lambda,sig,rho,N);%S(1)
is one day before initial date of option.
close all;
K=S(2)/money;
TT=30;tt=15;
IV=zeros(length(S),1);
for k=1:length(S)

Price=HestonCall(S(k),S(k),r,(TT-tt)/n,V(k),kappa,theta,sig,rho,lambda);
IV(k) = (blsimpv(S(k),S(k),r,(TT-tt)/n,Price))^2;
end
if strcmp(type,'ivr')
    param_ivr=param;

IV=(-param_ivr(1)*15/n*IV+param_ivr(1)*param_ivr(2)*15/n)/(exp(-param_ivr(1)*15/n)-1)+param_ivr(2);
end

e=1e-5;
lb=[e e e e -.999];
ub=[1 10 1 1 0];
a=[.1 4 .1 .1 -.2];
if method==1
    Vc=zeros(length(S),1);
    DeltaHeston=zeros(length(S),1);

```

```

Xs_Heston=zeros(length(S),1);
Vs_Heston=zeros(length(S),1);
Deltamp=zeros(length(S),1);
Vegamp=zeros(length(S),1);
Xs_mlp=zeros(length(S),1);
Vs_mlp=zeros(length(S),1);
Vb_Heston=zeros(length(S),1);
Vb_mlp=zeros(length(S),1);
err_Heston=zeros(length(S),1);
err_mlp=zeros(length(S),1);
Xs_bs=zeros(length(S),1);
Vs_bs=zeros(length(S),1);
Vb_bs=zeros(length(S),1);
err_bs=zeros(length(S),1);
    for i=2:length(S)
        if local
            [param,~]=fmincon(@(x)
ErrFun1(x,Datatemp),a,[],[],[],[],lb,ub);
        end
        iv=IV(i-1);

Vc(i)=-HestonCall(S(i),K,r,(N-i+1)/n,V(i),kappa,theta,sig,rho,lambda)
;

DeltaHeston(i)=HestonDelta(S(i),K,r,(N-i+1)/n,iv,param(1),param(2),pa
ram(3),param(4),lambda);

Xs_Heston(i)=DeltaHeston(i)+param(3)*param(4)/S(i)*HestonVega(S(i),K,
r,(N-i+1)/n,iv,param(1),param(2),param(3),param(4),lambda,eps);
        Vs_Heston(i)=Xs_Heston(i)*S(i);
        p_hs=sim(netnn,[S(i))/K;N-i+1;iv]);

Deltamp(i)=(sim(netnn,[S(i)+eps)/K;N-i+1;iv])-p_hs)/eps*K;
        Vegamp(i)=(sim(netnn,[S(i)/K;N-i+1;(iv+eps)])-p_hs)/eps*K;
        Xs_mlp(i)=Deltamp(i)+param(3)*param(4)/S(i)*Vegamp(i);
        %Vc_mlp(i)=sim(netnn,[S(i)+eps)/K;N-i+1;iv])*K;
        Vs_mlp(i)=Xs_mlp(i)*S(i);
        Xs_bs(i)=blsdelta(S(i),K,r,max((N-i+1)/n,1e-3),iv);
        Vs_bs(i)=Xs_bs(i)*S(i);
        if i==2
            Vb_Heston(i)=-Vc(i)-Vs_Heston(i);
            Vb_mlp(i)=-Vc(i)-Vs_mlp(i);
            Vb_bs(i)=-Vc(i)-Vs_bs(i);
        else

```

```

Vb_Heston(i)=exp(r/n)*Vb_Heston(i-1)-S(i)*(Xs_Heston(i)-Xs_Heston(i-1));

Vb_mlp(i)=exp(r/n)*Vb_mlp(i-1)-S(i)*(Xs_mlp(i)-Xs_mlp(i-1));
    Vb_bs(i)=exp(r/n)*Vb_bs(i-1)-S(i)*(Xs_bs(i)-Xs_bs(i-1));
end

err_Heston(i)=exp(-r*i/n)*abs(Vc(i)+Vb_Heston(i)+Vs_Heston(i));
err_mlp(i)=exp(-r*i/n)*abs(Vc(i)+Vb_mlp(i)+Vs_mlp(i));
err_bs(i)=exp(-r*i/n)*abs(Vc(i)+Vb_bs(i)+Vs_bs(i));
disp(i);
end

E(w,1)=err_Heston(end);
E(w,2)=err_mlp(end);
E(w,3)=err_bs(end);
E(w,4)=sum(abs(err_Heston))/(length(err_Heston)-1);
E(w,5)=sum(abs(err_mlp))/(length(err_mlp)-1);
E(w,6)=sum(abs(err_bs))/(length(err_bs)-1);
figure,clf;
    plot(-Vc,'r');
    hold on;
    plot(Vs_mlp+Vb_mlp,'b');
    plot(Vs_Heston+Vb_Heston,'g');
    plot(Vs_bs+Vb_bs,'k');
    legend('Option','MLP','Heston','bs');
    xlabel('Time to Maturity');
    ylabel('Value');
    hold off;
    %saveas(gcf,sprintf('figure%d.fig',w));
    disp('====');
    disp(w);
    disp('====');

F(:,1)=mean(E(:,1:3),1)';
F(:,2)=min(E(:,1:3))';
F(:,3)=max(E(:,1:3))';
F(1,4)=nan;
F(2,4)=sum(E(:,2)<E(:,1))/m;
F(3,4)=sum(E(:,3)<E(:,1))/m;
F(:,5)=(exp(-r*T)*sqrt(mean(E(:,1:3),1).^2+std(E(:,1:3),1).^2))';

```

```

F(:,6)=mean(E(:,4:6),1)';
F(:,7)=min(E(:,4:6))';
F(:,8)=max(E(:,4:6))';
F(1,9)=nan;
F(2,9)=sum(E(:,5)<E(:,4))/m;
F(3,9)=sum(E(:,6)<E(:,4))/m;
F(:,10)=money;
F(:,11)=T;
else
    for i=2:length(S)
        iv=IV(i-1);

Vc(i)=-HestonCall(S(i),K,r,(N-i)/n,V(i),kappa,theta,sig,rho,lambda);

Vcc(i)=HestonCall(S(i),KK,r,(N-i)/n,V(i),kappa,theta,sig,rho,lambda);

Xcc_Heston(i)=HestonVega(S(i),K,r,(N-i)/n,iv,param(1),param(2),param(
3),param(4),lambda,eps)/HestonVega(S(i),KK,r,(N-i)/n,iv,param(1),para
m(2),param(3),param(4),lambda,eps);

Xs_Heston(i)=HestonDelta(S(i),K,r,(N-i)/n,iv,param(1),param(2),param(
3),param(4),lambda)-HestonDelta(S(i),KK,r,(N-i)/n,iv,param(1),param(2
),param(3),param(4),lambda)*Xcc_Heston(i);
        Vs_Heston(i)=S(i)*Xs_Heston(i);
        Vcc_Heston(i)=Vcc(i)*Xcc_Heston(i);
        p_hs=sim(netnn,[(S(i))/K;N-i+1;iv]);
        p_hs_kk=sim(netnn,[(S(i))/KK;N-i+1;iv]);

Xcc_mlp(i)=(sim(netnn,[(S(i)+eps)/K;N-i+1;iv])-p_hs)/(sim(netnn,[(S(i
)+eps)/KK;N-i+1;iv])-p_hs_kk)*KK/K;

Xs_mlp(i)=(sim(netnn,[(S(i)+eps)/K;N-i+1;iv])-p_hs)/eps*K-(sim(netnn,
[(S(i)+eps)/KK;N-i+1;iv])-p_hs_kk)/eps*KK*Xcc_mlp(i);
        Vs_mlp(i)=Xs_mlp(i)*S(i);
        Vcc_mlp(i)=Vcc(i)*Xcc_mlp(i);
        if i==2
            Vb_Heston(i)=-Vc(i)-Vs_Heston(i)-Vcc_Heston(i);
            Vb_mlp(i)=-Vc(i)-Vs_mlp(i)-Vcc_mlp(i);
        else

Vb_Heston(i)=exp(r/n)*Vb_Heston(i-1)-S(i)*(Xs_Heston(i)-Xs_Heston(i-1
))-Vcc(i)*(Xcc_Heston(i)-Xcc_Heston(i-1));

Vb_mlp(i)=exp(r/n)*Vb_mlp(i-1)-S(i)*(Xs_mlp(i)-Xs_mlp(i-1))-Vcc(i)*(X

```

```
cc_mlp(i)-Xcc_mlp(i-1));  
    end  
  
err_Heston(i)=exp(-r*i/n)*abs(Vc(i)+Vb_Heston(i)+Vs_Heston(i));  
    err_mlp(i)=exp(-r*i/n)*abs(Vc(i)+Vb_mlp(i)+Vs_mlp(i));  
    disp(i);  
    end  
end  
end  
end
```

谢辞

如今回首这漫长的投身毕设的时间，最深刻的记忆是整日研习理论，编写代码的艰辛。我会想起那挑灯夜战代码的许多夜晚。但这种艰辛中无疑也是蕴含着喜悦与享受的。模型构建到编程实现仿真的这个漫长的过程，任何一个细小环节的差池都会导致结果的巨大误差，即是做到纤毫无差也会因为模型本身的缺陷得到并不理想的结果。正因此，一个理想的结果所带来的欣喜是发自内心的。这种快乐也正是激励我的源动力。回首整篇文章，一些当时让我欣喜若狂的结果此时看来并不那么成熟，很多段落尚显稚嫩，但却无疑都蕴含着巨大的付出，是以我无比珍重这篇文章。我时常会迷失在复杂的模型或方法论中。方法诚然重要，但任何研究的核心在于对问题的本质形成有力的见解。

我需要感谢在此过程中帮助到我的所有人。我想感谢我的导师罗俊老师。他总是于百忙中抽空给予我适时的指导，让我每周进行成果的汇报，也在我有所松懈的时候进行督促。毕设之外，老师还为我提出学习、升学的指导，给我提供机会参加会议，这些都对我有积极的影响，也真正地开拓了我的视野，让我对学界、业界有了更深入的了解。我真心感谢老师对我的帮助。我想感谢蓝海老师。他精彩的授课为我打下了夯实的基础。感谢杨义虎老师、王立河老师，他们的课程同样令我获益匪浅。我要感谢乐向楠同学，我总能在同他的交流中获得启迪，他精深的学术能力也令我钦佩。我也要感谢父母一直以来对我的关心，以及对我一系列决定的支持。一路走来，我接受到那么多人的帮助，恕不能一一列举。

最后，感谢评阅老师的耐心审阅。

袁航

15.5.17 于上海