Stable Local Volatility Calibration Using Kernel Splines

by

Cheng Wang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2008

Abstract

This thesis proposes an optimization formulation to ensure accuracy and stability in the local volatility function calibration. The unknown local volatility function is represented by kernel splines. The proposed optimization formulation minimizes calibration error and an L_1 norm of the vector of coefficients for the kernel splines. The L_1 norm regularization forces some coefficients to be zero at the termination of optimization. The complexity of local volatility function model is determined by the number of nonzero coefficients. Thus by using a regularization parameter, the proposed formulation balances the calibration accuracy with the model complexity. In the context of the support vector regression for function based on finite observations, this corresponds to balance the generalization error with the number of support vectors. In this thesis we also propose a trust region method to determine the coefficient vector in the proposed optimization formulation. In this algorithm, the main computation of each iteration is reduced to solving a standard trust region subproblem. To deal with the non-differentiable L_1 norm in the formulation, a line search technique which allows crossing nondifferentiable hyperplanes is introduced to find the minimum objective value along a direction within a trust region. With the trust region algorithm, we numerically illustrate the ability of proposed approach to reconstruct the local volatility in a synthetic local volatility market. Based on S&P 500 market index option data, we demonstrate that the calibrated local volatility surface is smooth and resembles in shape the observed implied volatility surface. Stability is illustrated by considering calibration using market option data from nearby dates.

Acknowledgements

First of all, I would like to thank my supervisor, Prof. Yuying Li. Her remarkable guidance led me through every step of thesis work. She came upon the interesting idea, patiently imparted research methodologies and principles, involved in the research investigation, and even devoted much time to instructing thesis writing. I feel very lucky to be able to spend two years with such a responsible and considerate professor.

I would also like to thank my thesis readers, Prof. Peter Forsyth and Prof. Justin Wan for taking time to review my thesis carefully and providing valuable comments, corrections and advice.

I would like to thank my husband, son and parents for their endless love and support, also for their forgiveness for the insufficient time that I spent with them.

Lastly many thanks should be given to all my schoolmates in the Scientific Computation Lab. They gave me a lot warmhearted help and support, and made my graduate study such a pleasant journey.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

One of the main objectives of this thesis is to formulate an optimization problem to calibrate the local volatility function stably and accurately. We extend a trust region algorithm for bound constrained minimization problem to solve the calibration problem which includes a 1-norm of the unknown vector of coefficients in the objective function. The stability and accuracy of the model calibration is controlled by regularization based on the 1-norm of unknown coefficients.

In this chapter, we provide financial and theoretical background for this thesis work. The standard Black-Scholes model and its drawbacks, illustrated by the volatility smile, various proposals and discussions by researchers regarding this issue, and finally an outline of the proposed LVF model calibration approach using kernel splines are described in the following sections.

1.1 Black-Scholes Model and Volatility Smile

A financial derivative is a contract whose value depends on a specified underlying asset like a stock, bond, commodity, interest or exchange rate. The well-known classical Black-Scholes model [4] has long been applied to price options and other derivative securities. Under the Black-Scholes model, the behavior of the underlying asset price S is modeled by the following stochastic differential equation:

$$\frac{dS}{S} = \mu dt + \sigma dW_t.$$

This equation describes that the asset return follows a diffusion process, where W_t is a standard Brownian motion, μ is the expected return of the asset and σ is the volatility of the underlying asset. The stochastic behavior of the value of a derivative $V(S,t)$ can be derived from the Ito's lemma:

$$dV = (\frac{\partial V}{\partial t} + \mu S\frac{\partial V}{\partial S} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2})dt + \sigma S\frac{\partial V}{\partial S}dW_t.$$

Under the Black and Scholes model, it is possible to construct a portfolio of the derivative and the underlying which is instantaneously riskless. Under the assumption of no-arbitrage opportunity the portfolio must earn the same rate of return as other short-term risk-free securities. By taking this equality, the following Black-Scholes equation is obtained:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0.$$

The Black-Scholes differential equation is independent of the risk preferences as μ does not appear in it. In the Black-Scholes equation, the only parameter which cannot be observed from the market directly is the volatility σ. In fact, σ is a measure of the uncertainty about the return provided by the underlying asset. It can be estimated empirically from historical asset prices. In reality finance engineers usually work with what is known as implied volatility. These are the volatilities implied by the market option data. They are obtained by inverting Black-Scholes formula with observed option prices. If the Black-

	strike(% of spot)							
maturity(yrs)	85%	90%	95%	100%	105%	110%	115%	120%
.425	.177	.155	.138	.125	.109	.103	.1	.114
.695	.172	.157	.144	.133	.118	.104	.1	.101
.94	.171	.159	.149	.137	.127	.113	.106	.103
1	.171	.159	.15	.138	.128	.115	.107	.103
1.5	.169	.16	.151	.142	.133	.124	.119	.113

Table 1.1: Implied volatilities for S&P 500 Index Options in Oct, 1995

Scholes model truly describes the underlying price dynamically, they would be the same constant for all options on the underlying with different strikes and maturities. However the observed implied volatilities show a strong dependence of implied volatility on the option strikes (volatility skew) and maturities (volatility term structure). This dependence is collectively referred to as the volatility smile. For example, in October 1995, the implied volatility of at-the-money S&P 500 Index Option for a maturity of 1 year is 13.8% [1]. While the out-of-the-money option with strike of 110% spot asset price has implied volatility 11.5%. Table 1.1 lists the S&P 500 Index Option implied volatilities of strikes at % of the spot price in Oct, 1995 [1]. Figure 1.1 shows the variation of implied volatility with respect to strikes and maturities. We can find that the implied volatilities are lower for out-of-the-money options and for options with shorter maturities. In addition, in October 1987, the two-month S&P 500 futures price fell 29% [20]. This would only happen with probability of 10^{-160} under the lognormal diffusion process hypothesis, which is the standard constant volatility assumption. These examples reveal that the Black-Scholes model is not satisfactory. In practice, the finance traders use different implied volatilities to price options with different strikes and maturities. Unfortunately it is difficult to price exotic options since these options are very sensitive to specification of volatility. Thus it is unreasonable to use any constant volatility to price many types of exotic options.

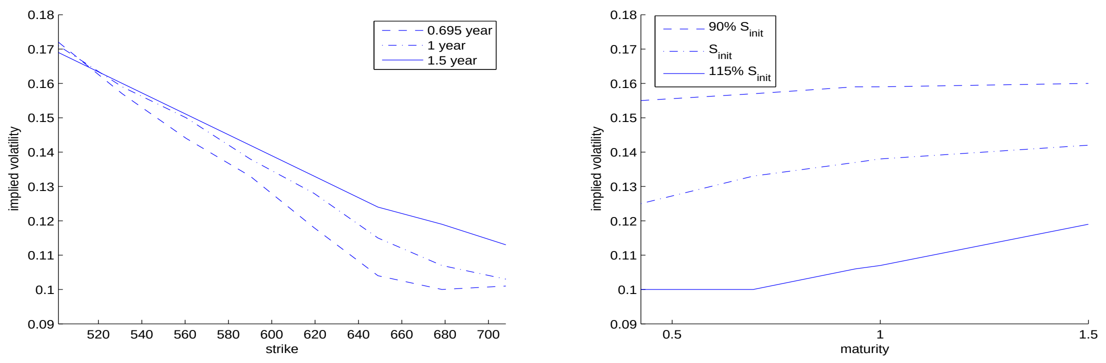


Figure 1.1: Implied volatilities for S&P 500 Index Options in Oct, 1995 with respect to strikes and maturities

1.2 Models for Volatility Smile

Researchers have attempted to improve the Black-Scholes model to fit the volatility smile. Merton [23] suggests a model where the asset price has jumps superimposed upon a geometric Brownian motion. These jumps follow a Poisson process. Hull and White [18] propose a stochastic volatility model for the asset price. Under this model, options prices can be obtained using a Monte Carlo method; simple European options can also be priced with an analytic formula. In these two models, a second random factor, is added. Thus a non-traded source of risk, such as jumps and stochastic volatility is added, which can lose the completeness of the model when hedging portfolios by adding options.

Dupire [15] introduces a one-factor model of a risk-neutral process for the spot asset:

$$\frac{dS_t}{S_t} = (r - q)dt + \sigma(S_t, t)dW_t \tag{1.1}$$

where S_t is the stock price at time t, $r > 0$ is the risk free interest, $q > 0$ is a constant dividend yield, and $\sigma(S_t, t)$ is the local volatility function (LVF), which now depends deterministically on S_t and t. The local volatility function $\sigma(S_t, t)$ can be chosen so that the model prices of all European options are consistent with the market. This model is known as the LVF model. Similar to the Black-Scholes model, it only has one source of randomness. Under this model, any option on the underlying can be priced and hedged using the underlying asset. One can also use deterministic local volatility model to price and hedge an exotic option. It is shown, see e.g., Dupire [15] and Andersen and Brotherton-Ratcliffe [1], that $\sigma(S_t, t)$ can be calculated analytically below:

$$\sigma^2(K,T) = 2\frac{\partial V^0/\partial T + qV^0 + (r-q)K\partial V^0/\partial K}{K^2(\partial^2 V^0/\partial K^2)} \tag{1.2}$$

where $V^0(K,T)$ is the initial price of an option with strike price K and maturity T. This equation can be used to estimate the LVF $\sigma(S_t, t)$ if option prices for all strikes and maturities are available in the market. Unfortunately in reality option prices are available only for a discrete set of strikes and maturities. Thus the calibration of the local volatility function becomes recovering the local volatility from a finite set of market option prices. This is an inverse problem and is ill-posed [6], e.g., there are more than one local volatility surfaces that can give theoretical prices matching market prices. A simple solution would be to use interpolation and extrapolation to create options data of all strikes and maturities. But it is not safe to interpolate option prices because it may create arbitrage opportunities and add erroneous information to the market options.

1.3 Methods for LVF Model Calibration

Researchers have been making efforts towards solving this ill-posed problem in a stable way. Dupire [15], Derman & Kani [13] and Rubinstein [20] propose implied binomial/trinomial tree algorithms for pricing options. They introduce a method to infer risk-neutral probabilities on the tree from market option prices. Then the local volatility structure at lattices is implied by these risk-neutral probabilities. However, the methods in [20, 13] force the well-posedness by constructing a complete set of market options using interpolation and extrapolation of available market option prices.

Avellaneda [2] constructs a representation for $\sigma(S_t, t)$ with a relative-entropy method. However, the local volatility surface constructed with their method consists of "humps" and "troughs" near strike/expiration date. This local volatility irregularity may be unrealistic and bring difficulty to discretise the space-time domain in a finite difference or finite element pricing method.

Smoothness and stability have been the main objective of local volatility reconstruction. Methods have been proposed by selecting a LVF to match option prices with the model values. These methods involve solving minimization problem:

$$\min_{\sigma(S_t,t)} \sum_{j=1}^{m} (V_j^0(\sigma(S_t, t)) - \bar{V}_j^0)^2$$

where $\bar{V}_j^0$, $1 \leq j \leq m$, are observed m initial market option prices, and $\{V_j^0\}_{j=1}^m$ are the model option values with local volatility function $\sigma(S_t, t)$ at the same strikes and maturities as the market options $\{\bar{V}_j^0\}_{j=1}^m$. To recover a local volatility function stably, numerical methods that focus on adding a regularization criterion with the above minimization problem have been considered. Lagnado & Osher [21] present a regularized method which minimizes the L_2 norm of the gradient of the local volatility to fit the volatility smile using a finite difference method. This approach does not require explicit interpolation or extrapolation of observed market prices. However it has disadvantage of high computation cost. The calculation of derivatives in the gradient descent minimization requires extensive computation of solutions to partial differential equations.

Splines have been known to play an important role in approximating smooth curves and surfaces [14]. They have also been used as a tool for regularizing ill-posedness of function approximations from finite known data [29]. Jackson & Süli [19] present an algorithm which chooses the local volatility function to be a 2-dimensional spline constructed by nodal points in price-time space. The representation of $\sigma(S_t, t)$ is therefore guaranteed to be a smooth function. An adaptive finite element method is used to solve the partial differential equation. The LVF $\sigma(S_t, t)$ approximation is fulfilled by using regularization strategy which minimizes a cost function of partial derivative of local volatility with respect to time and underlying price at nodal points. However this method still requires relatively heavy computation.

Coleman, Li and Verma [10] propose a spline functional approach. The local volatility function $\sigma(S_t, t)$ is represented by a cubic spline constructed with a fixed set of spline knots and end condition. The method chooses the number of spline knots to be no greater than the number of available option prices. The local volatility is approximated by solving a constrained nonlinear optimization problem to match the market option prices. Unlike the proposal of Jackson and Süli that uses nodal points over price-time domain, this approach chooses only a few knots to construct the spline. The stability of the LVF is controlled by the number and placement of these knots. Since the degree of freedom is the same as the number of spline knots, the dimension of the optimization problem is typically small and the local volatility computation is cost-efficient. However this approach still has some disadvantages:

- The choice and placement of spline knots is judged by a rule of thumb. Inappropriate choices may affect the effectiveness of local volatility approximation.
- If the number of knots is not chosen appropriately, calibration is not sufficiently robust against the inevitable noise of market options data. Slight difference in the market data will give deviated volatility at low or high spot prices. This causes difficulty in pricing exotic options accurately.
- The LVF calibration from data on consecutive dates can be quite different.
- The calibrated LVF from market options data seems to have unrealistic oscillations.

Based on this survey of numerical methods for the LVF model calibration, we propose an efficient, accurate, robust and stable approach. By stability we mean the property that an algorithm identifies genuine relations of the data source [24]. If we rerun the algorithm on a new sample from the same source it should identify a similar relation. While robustness is the property that an algorithm is insensitive against the noise perturbation of the sampled data. In this thesis, we propose a kernel spline $\sigma(S,t;x)$ in contrast to cubic spline to represent the unknown local volatility function. This kernel spline is formulated by the inner product of linear splines with infinite number of knots. The unknown coefficients $x \in \Re^n$ in the kernel spline representation are determined by solving an optimization problem. The kernel coefficients $x \in \Re^n$ are computed to make the local volatility produce option prices that match the given market data. Ill-posedness of the problem is solved by adding regularization criterion of L_1 norm of coefficient vector. Specifically, our optimization problem becomes:

$$\min_{x\in\Re^n} \left(\sum_{j=1}^{m} \left(V_j^0(\sigma(S,t;x)) - \bar{V}_j^0 \right)^2 \right) + \rho \parallel x \parallel_1$$

where $\rho > 0$ is a regularization parameter. This proposed formulation is motivated by support vector regression for function estimation based on a few observations. Finite difference method is used to determine the option prices by solving a partial differential equation.

The above proposed formulation balances the calibration accuracy with the model complexity based on the regularization parameter ρ. Minimizing the L_1 norm of coefficient vector x for the kernel spline forces some coefficients to be zero [16]. This corresponds to minimizing the number of support vectors in the context of the support vector regression for function estimation based on finite observations. The larger the regularization parameter ρ is, the larger the calibration error is allowed. Thus the less the number of nonzero coefficients x exists, and the simpler the complexity of calibration model is, which makes it more stable.

Our LVF calibration formulation overcomes the disadvantages of previous LVF calibration approach by cubic splines. Our computational results indicate that the calibrated LVF surface is stable and smooth for market S&P 500 Index Options on different dates. A slight change of market options will not cause unrealistic oscillation of LVF surface. In addition, the model calibration is insensitive against the number and placement of the training points we choose in price-time domain. Since the kernel spline used is smooth, our approach keeps the advantage of smoothness of cubic spline approach. Because the number of degrees of freedom of the problem is the number of unknown coefficients x, the dimension of the optimization is reasonably small. Finally this method does not introduce any erroneous information due to the interpolation and extrapolation of the available market data.

The proposed calibration problem has the following form:

$$\min_{x \in \Re^n} f(x) + \parallel x \parallel_1$$

where $f : \Re^n \rightarrow \Re^1$ is a smooth function. In this thesis, we present an interior trust region algorithm to solve this nonlinear function plus one-norm variable optimization problem. The main part of this algorithm is reduced to a standard trust region quadratic subproblem. In this algorithm we introduce a line search technique to deal with the problem of non-differentiability at points with zero components. Experiment results demonstrate the effectiveness and efficiency of this algorithm for our LVF calibration problem.

The presentation of the thesis is organized as follows. We first describe the local volatility approximation with a kernel spline in Chapter 2. Next we present the trust region algorithm and a simple function estimation example in Chapter 3. Performance and computational discussion of a synthetic option example and a real market option example are described in Chapter 4. Finally, Chapter 5 presents concluding remarks.

Chapter 2

An L_1 Optimization Formulation for a Stable LVF Calibration

Our objective is to estimate a local volatility function (LVF) based on a finite number of market option prices which can be regarded as indirect measurement of LVF. This problem is very challenging. Let us consider a simpler problem first. Imagine that we want to determine a LVF based on a finite number of direct function observations, i.e., local volatility values at some sample points of price-time domain. Support vector regression (SVR) can be employed to identify the LVF efficiently and stably. Notions of support vector and kernel method in classification problem help to explain the essential properties of the SVR algorithms. With the indirect measurement of LVF, we propose an L_1 optimization formulation to calibrate the LVF model. This approach is proposed to achieve the corresponding properties of SVR method for function estimation with direct measurement. In this chapter, we first present the overview of the LVF model calibration problem. Next we explain the notions of support vectors and kernel function for support vector machines for classification problems. Then we propose the kernel spline representation of a local volatility function $\sigma(S_t, t)$ and the L_1 optimization formulation for the LVF model calibration. We motivate our proposed formulation which attempts to achive the key properties of the SVR method.

2.1 LVF Model Calibration

Dupire [15] first proposes a local volatility function model for option pricing. Under this model, a risk-neutral process for the underlying asset price in the form of a one-factor diffusion is:

$$\frac{dS_t}{S_t} = (r-q)dt + \sigma(S_t,t)dW_t \tag{2.1}$$

where S_t is the stock price at time t, constants $r > 0$ and $q > 0$ are the risk free interest and the dividend yield, W_t is a standard Brownian motion, and $\sigma(S_t,t)$ is the LVF which depends deterministically on S_t and t. Under the no-arbitrage assumption, the following partial differential equation can be derived:

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2(S,t)S^2\frac{\partial^2 V}{\partial^2 S} + (r-q)S\frac{\partial V}{\partial S} - rV = 0 \tag{2.2}$$

where $V(S,t)$ is the model option price for asset price S at time t. The boundary condition can be found as the final payoff of V at $t = T$. If the local volatility function $\sigma(S,t)$ is known, the option value V can be uniquely determined by the above backward parabolic equation (2.2). American option prices can be determined from a partial differential complementarity equation. Furthermore the local volatility $\sigma(S,t)$ can be utilized for other path-dependent options pricing and constructing dynamic hedging portfolios.

Unfortunately, the local volatility $\sigma(S,t)$ cannot be observed directly from the market. In addition, it cannot be obtained analytically with equation (1.2) described in the previous chapter because a complete set of market option prices is normally not available. Our problem becomes to determine local volatility $\sigma(S,t)$ as stably as possible to match limited market information accurately. Suppose that we have m market option prices $\{\bar{V}_j^0\}$, $1 \leq$

$j \leq m$ with strikes $\bar{K}_j$ and maturities $\bar{T}_j$. We want to find the $\sigma(S,t)$ such that the solutions to equation (2.2) at these strikes and maturities satisfy

$$V_j(S_0, 0; \bar{K}_j, \bar{T}_j) = \bar{V}_j^0(\bar{K}_j, \bar{T}_j), \quad \forall \;\, 1 \leq j \leq m. \tag{2.3}$$

This is referred as model calibration problem. A good calibrated model should price assets and derivatives consistently with the market. In addition, the calibration should be stable. Specifically, similar set of market option prices should lead to similar calibrated local volatility. The model calibration can be used to predict the future behavior of the underlying asset in (2.1) with the available market option prices. With the calibrated local volatility function one can price path-dependent exotic options.

Dupire [15] introduces an adjoint partial differential equation for pricing European options:

$$\frac{\partial V^0}{\partial T} - \frac{1}{2}\sigma^2(K,T)K^2\frac{\partial^2 V^0}{\partial K^2} + (r-q)K\frac{\partial V^0}{\partial K} + qV^0 = 0 \tag{2.4}$$

where $\sigma(K,T)$ is equivalent to $\sigma(S,t)$ with variables S and t replaced with K and T. Function $V^0(K,T)$ is the initial ($t=0$) price of the option with strike K and maturity T. The boundary condition of V^0 can be found from the payoff function at $T=0$. In contrast to the backward parabolic Black-Scholes equation (2.2), this is a forward parabolic equation since the computation direction is from $T=0$ to $T=T_{max}$. Actually the equation (1.2) that gives the analytic solution of $\sigma(K,T)$ is obtained by solving this equation. The adjoint equation allows us to price a range of options of different strikes and maturities by solving only one PDE. To compute m option values it only takes one pde calculation, which is $\frac{1}{m}$ computational cost of using Black-Scholes equation. Thus the calibration problem becomes to find a local volatility function $\sigma(K,T)$ such that the solutions to the forward parabolic equation (2.4) satisfy condition (2.3).

2.2 Function Estimation with SVR

Determining a LVF by solving (2.3) in the previous section is an ill-posed inverse problem. There exist more than one local volatility functions which can price available market options accurately. We have described various intensive efforts [1, 6, 2, 21, 19, 10] towards solving this problem in Chapter 1. However since there are still some drawbacks in these methods, developing a more efficient, stable and accurate calibration method is our ultimate goal.

One of the key problems in the LVF calibration is a proper mechanism to balance calibration accuracy and stability of the calibrated model. Accuracy requires the model to be sufficiently complex to match all given data, while stability demands the model to be sufficiently simple so that a slight change of data does not cause large change in the model. In [10], the local volatility function is represented by a cubic spline with a fixed number of spline knots and end conditions. The conflicting goals of achieving accuracy and stability is balanced by choosing a minimum number of spline knots (making a model simple) to match option prices accurately. Unfortunately this process is difficult to automate as described in the previous chapter.

The local volatility function estimation is clearly a challenging problem. Consider first the following simpler problem. Assume that the local volatility function only depends on underlying asset price S, i.e., $\sigma(S,t) \equiv \sigma(S)$. In addition, assume that we actually have direct observations of the local volatility $\bar{\sigma}_j = \sigma(S_j)$ at S_j, $j = 1, 2, ..., m$. The problem of determining a function $\sigma(S)$ from the observations $(S_j, \bar{\sigma}_j)$, $j = 1, 2, ..., m$, is a known statistical learning problem. In particular, the support vector regression offers a solution approach, see e.g., Vapnik [28]. The support vector (SV) method was discovered in the early 1960s for constructing separating hyperplanes for the pattern recognition problem [27, 26]. Then in 1992-1995, it was generalized for constructing nonlinear separating functions with kernel methods [5, 12]. In 1995, it was generalized for estimating real-valued functions [25], called support vector regression. SVR has shown to give good generalization performance on a wide variety of problems. It generalizes a number of well-known learning models such as neural networks and radial basis function networks [28].

Our proposed formulation for the LVF calibration is motivated by examining the properties of the solution to the support vector regression. To illustrate this, we first review a classification problem with SV method and then a classical function estimation problem with SVR. Suppose we want to separate two classes of data with a linear function $h : \Re^N \rightarrow \{\pm 1\}$ using input-output training data

$$(s_1, y_1), ..., (s_l, y_l) \in \Re^N \times \{\pm 1\}$$

such that h will correctly classify unseen data points (s, y). We consider the class of hyperplanes

$$(z \cdot s) + b = 0, \quad z \in \Re^N, \ b \in \Re,$$

and the corresponding decision function

$$h(s) = \text{sign}((z \cdot s) + b)$$

where $\cdot$ denotes the inner product of two vectors. Figure 2.1 shows a hyperplane $(z \cdot s) + b = 0$ and two marginal hyperplanes $(z \cdot s) + b = \pm 1$ that separate two classes of data. Suppose s_1 and s_2 are points on these two hyperplanes respectively, the margin of the classification problem is the distance between these two hyperplanes, i.e., $\frac{z}{\|z\|_2} \cdot (s_1 - s_2) = \frac{2}{\|z\|_2}$. The optimal hyperplane is the one that has the maximum margin of separation between the classes:

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\|z\|_2^2 \\ &\text{subject to} \quad y_i * ((z \cdot s_i) + b) \geq 1, \quad i = 1, ..., l. \end{aligned}$$

This constrained optimization problem is solved by introducing Lagrange multipliers $\beta_i \geq 0$, $1 \leq i \leq l$, and a Lagrangian function

$$L(z, b, \beta) = \frac{1}{2}\|z\|^2 - \sum_{i=1}^{l} \beta_i (y_i * ((s_i \cdot z) + b) - 1).$$

By minimizing L with respect to the primal variables z and b and maximizing L with respect to the dual variables β_i, one can obtain the dual of the optimization problem (Appendix A):

$$\begin{aligned} &\text{maximize} \quad \sum_{i=1}^{l} \beta_i - \frac{1}{2} \sum_{i,j=1}^{l} \beta_i \beta_j y_i y_j (s_i \cdot s_j) \\ &\text{subject to} \quad \sum_{i=1}^{l} \beta_i y_i = 0, \\ &\qquad\qquad\quad \beta_i \geq 0, \quad i = 1, ..., l. \end{aligned}$$

The decision function can be written as

$$h(s) = \text{sign}(\sum_{i=1}^{l} y_i \beta_i * (s \cdot s_i) + b)$$

where b is obtained by applying a data point with its corresponding $\beta_j \neq 0$ in the KKT complementarity condition:

$$\beta_j * [y_j * ((s_j \cdot z) + b) - 1] = 0$$

where the solution vector z is an expansion of those training points whose β_i is non-zero:

$$z = \sum_{i=1}^{l} \beta_i y_i s_i.$$

Those training points are called Support Vectors. By the KKT complementarity conditions, all SVs lie on the marginal hyperplanes. All remaining data points are irrelevant to the optimal solution since they do not appear in the solution vector z. In Figure 2.1 dark points are SVs.

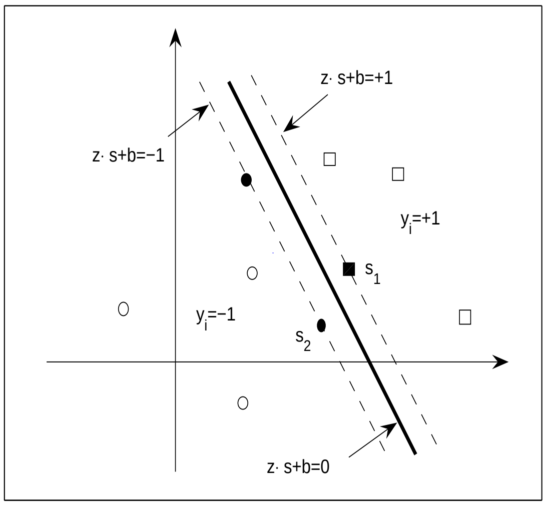


Figure 2.1: An optimal hyperplane separates two data classes

The above method only constructs linear optimal hyperplane. How can we deal with the case of a decision function which has a nonlinear relation with the input data? To construct SV machines for nonlinear decision functions, kernel methods are used to extend the optimal hyperplane algorithm. The key aspect of a kernel method is that data items are embedded in a vector space called the feature space and with some learning algorithm

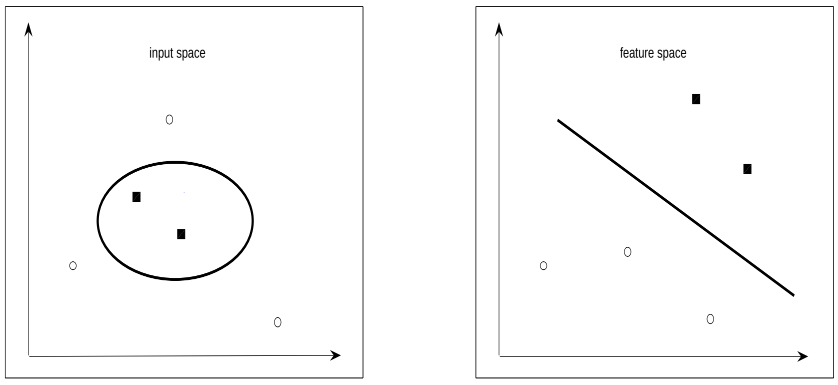


Figure 2.2: Mapping data from input space to feature space

corresponding relations are sought among the mapped data in the feature space. Using kernel functions, only the pairwise inner products of mapped data is used in determining the decision function. The coordinates of the data in feature space are not used. The pairwise inner products actually are computed efficiently directly from the data in input space using a kernel function. These methods enable researchers to analyze nonlinear relations in a high-dimensional feature space and yet preserve the efficiency of linear algorithms. Figure 2.2 shows an example of mapping data to a feature space, where an optimal hyperplane can be constructed. The mapped data in the feature space can be in high dimensionality. It might be difficult to write the nonlinear mapping function $\Phi(s)$ in an explicit form. Note that both the optimization problem and the decision function require only the evaluation of dot products of $\Phi(s)$. The dot product of $\Phi(s)$ can be represented as a kernel function with variables s in the input space. Thus to construct SV machines, we replace each s with

$\Phi(s)$. The quadratic problem and the decision function become:

$$\begin{aligned} &\text{maximize} \quad \sum_{i=1}^{l}\beta_i - \frac{1}{2}\sum_{i,j=1}^{l}\beta_i\beta_j y_i y_j F(s_i, s_j) \\ &\text{subject to} \quad \sum_{i=1}^{l}\beta_i y_i = 0 \\ &\qquad\qquad\quad \beta_i \geq 0, \quad i = 1, ..., l \end{aligned}$$

and

$$h(s) = \text{sign}(\sum_{i=1}^{l} y_i\beta_i * F(s, s_i) + b)$$

where we denote the kernel function $F(s_i, s_j) = \Phi(s_i)\cdot\Phi(s_j)$. However, noise data can cause overlaps of the data classes; a strictly separating hyperplane may not exist. In this case, slack variables $\xi_i \geq 0$, $i = 1, ..., l$, and corresponding relaxed constraints $y_i * ((z \cdot s_i) + b) \geq 1 - \xi_i$, $i = 1, ..., l$, are introduced to allow for the class data overlaps. In this case, we maximize the margin via $\|z\|_2$ and minimize the number of outliers, i.e., minimizing an objective function

$$\min_{z,\xi}\frac{1}{2}\|z\|_2^2 + C\sum_{i=1}^{l}\xi_i.$$

Constant $C > 0$ controls the trade-off between the two factors. By rewriting it with respect to the Lagrange multipliers (Appendix B), we find that the quadratic optimization problem is similar to the separable cases except that the constraints become

$$\sum_{i=1}^{l}\beta_i y_i = 0 \text{ and } 0 \leq \beta_i \leq C, \; i = 1, ..., l.$$

We can see that the Lagrange multiplier β_i is upper bounded by C. The decision function has the same form of separable cases with b computed by the fact that for all SVs with $\beta_i < C$, the slack variable ξ_i is zero, i.e., $\sum_{j=1}^{l} y_i\beta_j \cdot F(s_i, s_j) + b = 1$ from complementary

KKT conditions.

The learning algorithm illustrated above, support vector machine (SVM), uses the concept of margin which is specific to classification problems in pattern recognition. Later the SVM method is extended to function estimation [28]. An $\epsilon-$insensitive loss function below is introduced to construct a marginal hyperplane:

$$|y - h(s)|_\epsilon = \max\{0, |y - h(s)| - \epsilon\}. \tag{2.5}$$

Any y data with error less than ϵ can be considered within a tube of radius ϵ centered

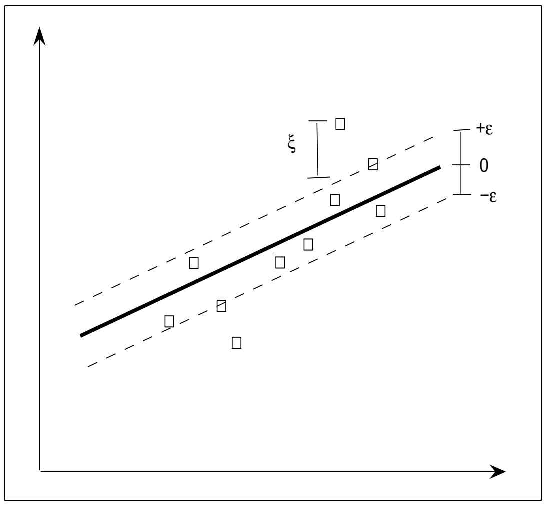


Figure 2.3: An ϵ tube introduced with loss function for function estimation

around function values. Figure 2.3 depicts a separating hyperplane for function estimation. Slack variables ξ are introduced for function values outside of the ϵ tube. For a linear regression case, to estimate $h(s) = (z \cdot s) + b$ with precision ϵ one needs to minimize

$$\frac{1}{2}\|z\|_2^2 + C\sum_{i=1}^{l} |y_i - h(s_i)|_\epsilon.$$

where $C > 0$ is a constant priori. Therefore a constrained optimization problem can be constructed

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\|z\|_2^2 + C\sum_{i=1}^{l}(\xi_i + \bar{\xi}_i) \\
\text{subject to} \quad & ((z \cdot s_i) + b) - y_i \le \epsilon + \xi_i, \quad \text{for all } i = 1, ..., l \\
& y_i - ((z \cdot s_i) + b) \le \epsilon + \bar{\xi}_i, \quad \text{for all } i = 1, ..., l \\
& \xi_i, \bar{\xi}_i \ge 0, \quad \text{for all } i = 1, ..., l.
\end{aligned}$$

Incorporating kernel functions, we represent the optimization problem in terms of the Lagrange multipliers (Appendix C):

$$\begin{aligned}
\text{maximize} \quad & -\epsilon\sum_{i=1}^{l}(\bar{\beta}_i + \beta_i) + \sum_{i=1}^{l}(\bar{\beta}_i - \beta_i)y_i \\
& -\frac{1}{2}\sum_{i,j=1}^{l}(\bar{\beta}_i - \beta_i)(\bar{\beta}_j - \beta_j)F(s_i, s_j) \\
\text{subject to} \quad & \sum_{i=1}^{l}(\beta_i - \bar{\beta}_i) = 0, \\
& 0 \le \beta_i, \bar{\beta}_i \le C, \quad i = 1, ..., l.
\end{aligned} \tag{2.6}$$

The regression estimation has the form:

$$h(s) = \sum_{i=1}^{l}(\bar{\beta}_i - \beta_i)F(s_i, s) + b,$$

where b is obtained by using the fact that for all SVs the constraints take equality and have $\xi_i = 0$ if $0 < \beta_i < C$ and $\bar{\xi}_i = 0$ if $0 < \bar{\beta}_i < C$. The SVs for function estimation problem are those training vectors on margins or outside of the ϵ tube. All other training vectors within the tube are irrelevant to the problem solution. Since a training vector can only land on either outer side of the ϵ tube, β_i and $\bar{\beta}_i$ cannot both be nonzero. Therefore SV can be given as below:

$$\begin{aligned} SV &\stackrel{\text{def}}{=} \{i : \text{either } \beta_i \neq 0 \text{ or } \bar{\beta}_i \neq 0\} \\ &= \{i : \bar{\beta}_i - \beta_i \neq 0\}. \end{aligned}$$

The number of SVs is affected by the accuracy level ϵ. Suppose we want to approximate a function $h(s)$ with the accuracy level ϵ. An ϵ tube is constructed to find the estimation function. It is a nonlinear tube since we introduce kernel mapping for input data. Since the ϵ tube tends to be flat, it will touch some sample points. The axis of the ϵ tube constructs the estimation function $h(s)$ and those sample points that touch the ϵ tube becomes the support vectors. For some points lying outside of the tube (with the introduction of slack variables ξ and $\bar{\xi}$), their β ($\bar{\beta}$) parameters are nonzero and should be included in the final estimation function. Therefore these points are also included as support vectors. It is easy to see that the wider the ϵ tube the lower the number of touching points, i.e., the fewer the support vectors. It can be shown that the number of SVs controls the balance of generalization accuracy and stability. The larger the ϵ level is allowed, the less the number of existing SVs is, the simpler the model complexity is, and therefore the more stable the generalization model is.

2.3 Kernel Spline Representation of LVF

In the simple example with observed local volatility values at sampled training points, we can form the quadratic optimization problem in the SVR framework to estimate a local volatility function. However when observing the indirect measurements of $\sigma(S,t)$, i.e., a finite number of available market option prices, we are not able to form the classical SVR quadratic optimization problem (2.6). Instead we motivate our L_1 optimization formulation to attempt to achieve the advantageous properties of SVR.

First we utilize kernel splines to represent the estimation function, i.e., $\sigma(S,t)$ according to the formulation in SVR. Spline kernels can be used to approximate nonlinear functions. Splines are created with either a fixed number of knots or an infinite number of knots. In this thesis we will use an infinite number of knots which construct an infinite dimensionality piecewise linear approximation of a function. Suppose that a one-dimensional function defined on the interval $[0, s_b]$, $0 < s_b < \infty$, is approximated by splines of order $d \geq 0$ with infinite number of knots: $\{t_i\}$, $1 \leq i < \infty$. A one-dimensional variable s is mapped into an infinite-dimensional vector u:

$$s \rightarrow u = (1, s, ..., s^d, (s-t_1)_+^d, ..., (s-t_i)_+^d, ...)$$

where

$$(s-t_k)_+^d = \begin{cases} 0 & \text{if } s \leq t_k, \\ (s-t_k)^d & \text{if } s > t_k. \end{cases}$$

Then the spline has the form:

$$h(s) = \sum_{i=0}^{d} a_i s^i + \int_0^{s_b} a(t)(s-t)_+^d dt,$$

where a_i, $i = 0, ..., d$, are unknown coefficients and $a(t)$ is an unknown function that constructs the expansion using an infinite number of knots. One attractive feature of kernel methods is that only a pairwise inner product is considered. And the inner products are computed eventually from data items in input space. Therefore coefficients a_i and function $a(t)$ for $h(s)$ are not our concern. Instead we consider the kernel function that generates the inner products of variable s and training input data s_i in the feature space as follows:

$$\begin{aligned} F(s, s_i) &= \int_0^{s_b} (s-t)_+^d (s_i - t)_+^d dt + \sum_{k=0}^{d} s^k s_i^k \\ &= \int_0^{(s \wedge s_i)} (s-t)^d (s_i - t)^d dt + \sum_{k=0}^{d} s^k s_i^k \\ &= \int_0^{(s \wedge s_i)} (u)^d (u + |s - s_i|)^d du + \sum_{k=0}^{d} s^k s_i^k \\ &= \sum_{k=0}^{d} \frac{C_d^k}{2d-k+1} (s \wedge s_i)^{2d-k+1} |s - s_i|^k + \sum_{k=0}^{d} s^k s_i^k \end{aligned}$$

where $s \wedge s_i$ denotes the minimum value between s and s_i, i.e., $\min(s, s_i)$, and C_d^k is the number of choices for choosing k samples from d samples. For the linear spline with $d = 1$ in particular, we have the following function after integration:

$$F(s, s_i) = 1 + s s_i + \frac{1}{2} \mid s - s_i \mid (s \wedge s_i)^2 + \frac{(s \wedge s_i)^3}{3}$$

where s_i is a training input data in interval $[0, s_b]$. The above kernel function can be extended with the estimated function defined on interval $[-s_b, +\infty)$, $0 \leq s_b < \infty$. Actually, we have

$$\begin{aligned} F(s,s_i) &= \int_{-s_b}^{+\infty} (s-t)_+^d (s_i-t)_+^d dt + \sum_{k=0}^{d} s^k s_i^k \\ &= \int_{-s_b}^{(s\wedge s_i)} (s-t)^d (s_i-t)^d dt + \sum_{k=0}^{d} s^k s_i^k \\ &= \int_{0}^{(s\wedge s_i+s_b)} (s-t+s_b)^d (s_i-t+s_b)^d dt + \sum_{k=0}^{d} s^k s_i^k \end{aligned}$$

Therefore for the linear spline $d=1$ the kernel function becomes

$$F(s,s_i) = 1 + ss_i + \frac{1}{2} \mid s - s_i \mid (s \wedge s_i + s_b)^2 + \frac{(s\wedge s_i + s_b)^3}{3}$$

It can be proven that the above kernel function is twice differentiable. Actually one can easily show that for all s_i, $1 \le i \le l$:

$$\begin{aligned} \lim_{s\to s_i^-} \frac{d(F(s,s_i))}{ds} &= \lim_{s\to s_i^+} \frac{d(F(s,s_i))}{ds}, \\ \lim_{s\to s_i^-} \frac{d^2(F(s,s_i))}{ds^2} &= \lim_{s\to s_i^+} \frac{d^2(F(s,s_i))}{ds^2}. \end{aligned}$$

Using this kernel, we represent an estimation function as

$$h(s;x) = \sum_{i=1}^{l} x^{(i)} F(s,s_i) + x^{(0)}$$

where s and training input data s_i, $1 \le i \le l$, are defined in $[-s_b, +\infty)$; this corresponds to $x^{(i)} = \bar{\beta}_i - \beta_i$, for $1 \le i \le l$, and $x^{(0)} = b$ in the quadratic optimization problem described in (2.6). The above kernel spline expansion is analogous to the linear SV approximation where a function is represented in the form:

$$h(s;x) = \sum_{i=1}^{l} x^{(i)}(s \cdot s_i) + x^{(0)}.$$

By replacing the inner product of input space data (s, s_i) with a kernel function $F(s, s_i)$, a learning algorithm with a kernel method has the same computation complexity as with a linear method. For $N-$dimensional space cases, the kernel splines are defined using tensor products of one-dimensional kernel functions:

$$F(s, s_i) = \prod_{k=1}^{N} F(s^{(k)}, s_i^{(k)}),$$

where we use $s = (s^{(1)}, ..., s^{(N)})$ to denote a variable in a $N-$dimensional space.

In this thesis we use this kernel spline to represent a LVF:

$$\sigma((K,T); x) = \left| \sum_{i=1}^{l} x^{(i)} F((K,T),(K_i,T_i)) + x^{(0)} \right|, \tag{2.7}$$

with

$$\begin{aligned} F((K,T),(K_i,T_i)) &= [1 + KK_i + \frac{1}{2} \mid K - K_i \mid (K \wedge K_i + K_b)^2 + \frac{(K \wedge K_i + K_b)^3}{3}] \times \\ & [1 + TT_i + \frac{1}{2} \mid T - T_i \mid (T \wedge T_i + T_b)^2 + \frac{(T \wedge T_i + T_b)^3}{3}] \end{aligned}$$

where (K,T) denotes a variable in two-dimensional strike-maturity space, and F is the tensor product of the kernel functions in these two dimensions. The function variables (K,T) and l training variables (K_i, T_i) are defined in interval $[-K_b, +\infty) \times [-T_b, +\infty)$. In (2.7), we use absolute value of the linear expansion of kernel splines since the local volatility is positive in finance practice.

However local volatility values $\sigma(K_i, T_i)$ at training vectors $\{K_i, T_i\}_{i=1}^{l}$ are not observed in the market. We only have market option prices $\{\bar{V}^0\}_{j=1}^m$ available. The quadratic optimization problem based on SVR cannot be formulated directly in this situation.

The goals of calibration remain the same. We first want to minimize the calibration error, which can be measured as

$$\sum_{j=1}^{m} w_j \left(V^0\left((\bar{K}_j, \bar{T}_j); x\right) - \bar{V}_j^0\right)^2$$

where each model option value $V^0(\bar{K}_j, \bar{T}_j; x)$ is uniquely determined by the local volatility function (2.7) specified by the unknown coefficients x. We have included the weights $\{w_j \geq 0\}$ in the formulation, which can be used to achieve desired accuracy when option values are of significantly different magnitudes.

In addition to control calibration error, we want to keep the local volatility function simple to reduce the generalization error of the local volatility function based on finite observations. We attempt to achieve this by pushing unnecessary coefficients of the kernel function representation (2.7) of the local volatility function to have zero values. Then the complexity of the model becomes simple enough. Combining these two objectives together, we solve the optimization problem:

$$\min_{x \in \Re^{l+1}} \sum_{j=1}^{m} w_j \left(V^0\left((\bar{K}_j, \bar{T}_j); x\right) - \bar{V}_j^0\right)^2 + \rho \sum_{i=0}^{l} |x^{(i)}| \tag{2.8}$$

where the constant $\rho > 0$ is a regularization parameter balancing the tradeoff between the two objectives. When the parameter ρ is large, the calibration error is large but the calibration local volatility function has more zero coefficients and tends to be simpler. This corresponds to the property in SVR solution: if we have fewer support vectors left in the estimation function with SVR approach, the function is an expansion of less complexity and thus simpler and is likely to be more stable.

We emphasize the use of the 1-norm of the coefficients $\|x\|_1$ in the objective function of the proposed formulation. We can think of this term as the exact penalty function for the constraints $x^{(i)} = 0$, $0 \le i \le l$. Indeed, with a finite $\rho > 0$, typically some subset of constraints will be satisfied. In addition, for each constraint $x^{(i)} = 0$, there exists a finite lower bound for ρ such that when ρ is greater than this bound, the constraint $x^{(i)} = 0$ will be satisfied. If we use the quadratic penalty function $\|x\|_2^2$ in the objective function, even though the objective function becomes smooth, the coefficients $\{x^{(i)}\}$ are typically nonzero at the solution [16]. The optimization problem (2.8) has the following equivalent constrained optimization formulation:

$$\begin{aligned}
\text{minimize} \quad & \sum_{j=1}^{m} w_j (V^0((\bar{K}_j, \bar{T}_j); x) - \bar{V}_j^0)^2 + \rho \sum_{i=0}^{l} \bar{x}^{(i)} \\
\text{subject to} \quad & \bar{x}^{(i)} - x^{(i)} \ge 0, \text{ for all } i = 0, ..., l \\
& \bar{x}^{(i)} + x^{(i)} \ge 0, \text{ for all } i = 0, ..., l \\
& \bar{x}^{(i)} \ge 0, \text{ for all } i = 0, ..., l.
\end{aligned}$$

Chapter 3

A Trust Region Algorithm

We now develop a trust region method based on proper affine scaling for the proposed LVF calibration problem. The LVF estimation problem (2.8) can be formulated as a slightly more general problem which minimizes a nonlinear function $f(x)$ plus the one-norm of the vector x as follows:

$$\min_{x \in \Re^n} f(x) + \|x\|_1 \tag{3.1}$$

where $f : \Re^n \to \Re^1$ is a twice continuously differentiable function, and $\|x\|_1 = \sum_{i=1}^n |x^{(i)}|$. We use $x^{(i)}$ to denote the ith component of the vector x.

3.1 Algorithm Outline

The objective function (3.1) has a smooth component $f(x)$ and a piecewise linear component $\|x\|_1$. In [8] by Coleman and Li, an interior trust region method is proposed to solve the bound-constrained minimization problem,

$$\min_{x \in \Re^n} f(x), \quad l \leq x \leq u$$

where $l \in \{\Re \cup \{-\infty\}\}^n$, $u \in \{\Re \cup \{\infty\}\}^n$, $l < u$, and $f : \Re^n \to \Re^1$ is a smooth function. At each iteration, the main computation is a trust region subproblem based on appropriate scaling, which depends on the first order KKT optimality condition as well as closeness of the current iterate to the constraints. In addition, a reflective technique is used to accelerate convergence, more details can be found in [7, 9].

One of the main component of solving (3.1) is to identify which variables have zero values at the solution (or which constraints become active at a solution). We outline subsequently a similar trust region method for solving (3.1). The trust region subproblem is based on appropriate affine scaling. The affine scaling matrices D_k and C_k are given naturally by examining the first-order KKT conditions for (3.1) (Appendix D): if a point x is a local minimizer, then

$$x^{(i)}[(\nabla f(x))^{(i)} + \text{sign}(x^{(i)})] = 0 \text{ and } \mid (\nabla f(x))^{(i)} \mid \leq 1 \text{ for } 1 \leq i \leq n \tag{3.2}$$

where

$$\text{sign}(x^{(i)}) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } x^{(i)} \geq 0, \\ -1 & \text{otherwise.} \end{cases}$$

This condition can be expressed as the nonlinear system of equations

$$D(x)(\nabla f(x) + \text{sign}(x)) = 0 \tag{3.3}$$

where

$$D(x) \overset{\text{def}}{=} \text{diag}(v(x)),$$

and the vector $v(x) \in \Re^n$ is defined below: for each $1 \le i \le n$,

$$v^{(i)} \overset{\text{def}}{=} \begin{cases} \mid x^{(i)} \mid & \text{if } |(\nabla f(x))^{(i)}| \le 1, \\ 1 & \text{otherwise} . \end{cases} \tag{3.4}$$

In (3.4) we use $|x^{(i)}|$ instead of $x^{(i)}$ so that we could introduce an affine scaling matrix $D_k^{-\frac{1}{2}}$ later. Actually using $|x^{(i)}|$ also meets the first-order KKT condition (3.2). A vector x satisfies equations (3.3) if and only if the first-order KKT conditions of (3.1) hold at x. To see this, let us suppose that the first-order KKT conditions hold at x. Then for $1 \le i \le n$, either $x^{(i)} = 0$ and $\mid (\nabla f(x))^{(i)} \mid \le 1$ or $(\nabla f(x))^{(i)} + \text{sign}(x^{(i)}) = 0$. Naturally we have either $v^{(i)} = 0$ or $(\nabla f(x))^{(i)} + \text{sign}(x^{(i)}) = 0$ for $1 \le i \le n$. Thus the nonlinear equations (3.3) hold at x. On the other hand, if the equations (3.3) hold at x we have either $v^{(i)} = 0$ or $(\nabla f(x))^{(i)} + \text{sign}(x^{(i)}) = 0$ for $1 \le i \le n$. It leads to either $x^{(i)} = 0$ and $\mid (\nabla f(x))^{(i)} \mid \le 1$ or $(\nabla f(x))^{(i)} + \text{sign}(x^{(i)}) = 0$ for $1 \le i \le n$, which gives $x^{(i)}[(\nabla f(x))^{(i)} + \text{sign}(x^{(i)})] = 0$ and $\mid (\nabla f(x))^{(i)} \mid \le 1$ for $1 \le i \le n$, i.e., the first-order KKT conditions.

Assume that, at the iteration k, $x_k^{(i)} \ne 0$, $1 \le i \le n$. A Newton step for (3.3) at the kth iteration satisfies

$$(J_k^v \text{diag}(g_k) + \text{diag}(v_k)\nabla^2 f(x_k))d_k = -\text{diag}(v_k)g_k \tag{3.5}$$

where $g_k \overset{\text{def}}{=} \nabla f(x_k) + \text{sign}(x_k)$. Here $J^v(x) \in \Re^{n \times n}$ is a diagonal matrix which corresponds to the Jacobian of $\mid v(x) \mid$. Each diagonal element equals to zero or ± 1.

Equation (3.5) can be written below:

$$\hat{M}_k s_k = -\hat{g}_k, \tag{3.6}$$

where

$$
\begin{aligned}
& s_k = D_k^{-\frac{1}{2}} d_k \\
& \hat{g}_k = D_k^{\frac{1}{2}} g_k = \operatorname{diag}(v_k^{\frac{1}{2}}) g_k \\
& \hat{M}_k = D_k^{\frac{1}{2}} * \nabla^2 f(x_k) * D_k^{\frac{1}{2}} + \operatorname{diag}(g_k) * J_k^v
\end{aligned}
$$

The Newton step (3.6) suggests an affine scaling transformation: $s_k = D_k^{-\frac{1}{2}} d_k$. With the transformation, our nonlinear plus one-norm minimization problem (3.1) can be locally reduced to a typical nonlinear unconstrained problem. A natural way to improve x_k is to solve the trust region subproblem

$$
\min_{s \in \Re^n} \{\hat{\psi}_k(s) : \|s\|_2 \leq \Delta_k\}, \tag{3.7}
$$

where

$$
\hat{\psi}_k(s) = \hat{g}_k^T s + \frac{1}{2} s^T \hat{M}_k s.
$$

The subproblem (3.7) has its counterpart in the original variable space:

$$
\min_{d \in \Re^n} \{\psi_k(d) : \|D_k^{-\frac{1}{2}} d\|_2 \leq \Delta_k\} \tag{3.8}
$$

where

$$
\begin{aligned}
& \psi_k(d) = g_k^T d + \frac{1}{2} d^T M_k d \\
& C_k = D_k^{-\frac{1}{2}} * \operatorname{diag}(g_k) * J_k^v * D_k^{-\frac{1}{2}} \\
& M_k = \nabla^2 f_k + C_k \\
& d_k = D_k^{\frac{1}{2}} s_k.
\end{aligned}
$$

The affine scaling matrix D_k defines the shape of the ellipsoid created by $\|D_k^{-\frac{1}{2}}d\|_2 = \Delta_k$. With this choice, the ellipsoid is short in directions corresponding to components of $x^{(i)}$ close to zero and $|(\nabla f(x))^{(i)}| \leq 1$, and long in directions corresponding to relatively large $| x^{(i)} |$ or $|(\nabla f(x))^{(i)}| > 1$. In this way the quadratic model (3.8) takes small steps along the direction with components of $x^{(i)}$ close to zero and $|(\nabla f(x))^{(i)}| \leq 1$.

The nonlinear system (3.3) derived from the KKT conditions is not differentiable when $x^{(i)} = 0$. We define a differentiable region in which all points have no zero component: $\mathcal{F} \overset{\text{def}}{=} \{x : x \in \Re^n, x^{(i)} \neq 0, \text{ for all } 1 \leq i \leq n\}$.

For any given direction d, we consider the following piecewise quadratic approximation of objective function (3.1):

$$\phi_k(d) \overset{\text{def}}{=} \nabla f_k^T d + \frac{1}{2} d^T \nabla^2 f_k d + \|x_k + d\|_1 - \|x_k\|_1 + \frac{1}{2} d^T C_k d \tag{3.9}$$

As an approximation to the change in objective function (3.1), $\phi_k(d)$ has at least accuracy of linear order $O(\|d\|)$. Note that without $\frac{1}{2}d^T C_k d$, $\phi_k(d)$ has quadratic accuracy order $O(\|d\|^2)$. But $\frac{1}{2}d^T C_k d$ is added to match our quadratic model $\psi_k(d)$. Since the differentiable region in (3.1) is not connected, and since we do not know in which region the optimal point lies, our algorithm must allow crossing points of nondifferentiability. Note that $\psi_k(d)$ is derived from (3.5) based on the Newton step for the KKT condition; thus it may not be a good approximation to the change of the objective function far away from a solution. Because $\phi_k(d)$ approximates the change of the objective function in (3.1) even though it is not differentiable everywhere, it gives a better way than $\psi_k(d)$ to represent quadratic approximation of objective function.

If $[x_k, x_k + d] \subset \mathcal{F}$, there is no sign change from x_k to $x_k + d$. Quadratic approximation $\phi_k(d)$ equals the objective value $\psi_k(d)$ of the trust region subproblem. This can be seen

Proposed Trust Region Algorithm. Let $0 < \mu < 1$.
For $k = 0, 1, ...$

Step 1. Compute f_k, g_k, D_k, M_k and C_k; define the quadratic model

$$\psi_k(d) = g_k^T d + \frac{1}{2} d^T M_k d.$$

Step 2. Compute a step d_k such that $x_k + d_k \in \mathcal{F}$, based on subproblem:

$$\min_d \{\psi_k(d) : \|D_k^{-\frac{1}{2}} d\|_2 \leq \Delta_k\}.$$

Step 3. Compute

$$\rho_k^f = \frac{f(x_k + d_k) - f(x_k) + \|x_k + d_k\|_1 - \|x_k\|_1 + \frac{1}{2} d_k^T C_k d_k}{\phi_k(d_k)}$$

Step 4. If $\rho_k^f > \mu$, then set $x_{k+1} = x_k + d_k$. Otherwise set $x_{k+1} = x_k$.

Step 5. Update Δ_k as specified below.

Updating Trust Region Size Δ_k
$0 < \mu < \eta < 1$, $\Lambda_U > 0$, $0 < \gamma_3 < 1$ and $0 < \gamma_0 < \gamma_1 < 1 < \gamma_2$

1. If $\rho_k^f < 0$ then set $\Delta_{k+1} = min(\Delta_k \gamma_0, \Lambda_U)$

2. If $0 < \rho_k^f < \mu$ then set

$$\Delta_{k+1} = \min(\Delta_k \gamma_k, \Lambda_U)$$

$$\gamma_k = \max(\gamma_0, \gamma_1 \|D_k^{-\frac{1}{2}} d_k\|_2 / \Delta_k)$$

3. If $\rho_k^f \geq \mu$ then

    If $\|D_k^{-\frac{1}{2}} d_k\|_2 / \|p_k\|_2 < \gamma_3$ then

$$\gamma_k = \gamma_0$$

    Otherwise

        If $\rho_k^f \geq \eta$ then

$$\gamma_k = \max(1, \gamma_2 \|D_k^{-\frac{1}{2}} d_k\|_2 / \Delta_k)$$

        Otherwise $\gamma_k = 1$

4. Set $\Delta_{k+1} = \min(\Delta_k \gamma_k, \Lambda_U)$

Figure 3.1: A trust region algorithm for the optimization problem (3.1)

easily from below:

$$\begin{aligned}
\phi_k(d) &\overset{\text{def}}{=} \nabla f_k^T d + \frac{1}{2} d^T \nabla^2 f_k d + \|x_k + d\|_1 - \|x_k\|_1 + \frac{1}{2} d^T C_k d \\
&= \nabla f_k^T d + \|x_k + d\|_1 - \|x_k\|_1 + \frac{1}{2} d^T M_k d \\
&= \nabla f_k^T d + \operatorname{sign}(x_k)^T d + \frac{1}{2} d^T M_k d \\
&= g_k^T d + \frac{1}{2} d^T M_k d.
\end{aligned}$$

In the proposed algorithm, we maintain differentiability for all iterates $\{x_k\}$. Assume that d_k is the solution to the trust region subproblem (3.8). It is possible that non-differentiability occurs from x_k to $x_k + d_k$, i.e., some of variables may become zero. For any descent direction d, let $\phi_k^*[d]$ denote the minimum value of $\phi_k(d)$ along d within the trust region, i.e.,

$$\phi_k^*[d] \overset{\text{def}}{=} \min_{\|\alpha D_k^{-\frac{1}{2}} d\|_2 \le \Delta_k, \alpha \ge 0} \phi_k(\alpha d).$$

A simple backtracking technique used in many interior points can similarly be used to avoid landing exactly on the points of non-differentiability.

The proposed trust region algorithm is summarized in Figure 3.1. Similar to the bound-constrained trust region method [7], global convergence can be ensured if, at each iteration,

$$\phi_k(d_k) \le \beta_g \phi_k^*[-D_k g_k] \tag{3.10}$$

where $0 < \beta_g < 1$ is a given constant. In addition, under some regularity conditions, local quadratic convergence can be achieved if

$$\phi_k(d_k) \le \beta_p \phi_k^*[p_k] \tag{3.11}$$

where $0 < \beta_p < 1$ is a given constant, and p_k is the solution of trust region subproblem (3.8). To satisfy both convergence conditions, we should allow crossing nondifferentiable hyperplanes when searching for minimizer along both steepest descent direction $-D_k g_k$ and trust region solution p_k. Condition (3.10) ensures the global convergence condition. However searching along this direction to reach a solution is quite slow in practice. On the other hand, the trust region solution p_k can lead to rapid convergence when started close enough to a solution. Therefore we choose the better improvement along these two directions for each iteration.

The trust region size is adjusted at each iteration to ensure a sufficient decrease of the objective function. The size of the trust region is critical to the effectiveness of each step. If the region is too small, the iteration misses a chance to take a larger step to move closer to the minimizer of the objective function. If too large, the minimizer of the model may be far from the minimizer of the objective function; in this case we have to reduce the size of the region and try again. In practice, we choose the size of the trust region according to the performance of the algorithm during previous iterations. Basically we increase the size of trust region, i.e., Δ_k, to take more ambitious steps if the quadratic approximation well represents the objective function reduction, i.e., $\rho_k^f > \eta$. On the contrary, we decrease Δ_k if our quadratical model is an inadequate representation of the objective function. Similar to the algorithm proposed by Coleman and Li for bound constrained minimization problem, the trust region method also allows possible trust region size reduction even when $\rho_k^f > \eta$. Specifically, when the chosen step d_k is much smaller than the trust region solution p_k, even if $\rho_k^f > \eta$, the trust region size Δ_k should still be reduced. This situation would happen when the current iterate is very close to a hyperplane $x^{(i)} = 0$, $1 \leq i \leq n$, and the gradient after the hyperplane becomes ascent. Backtracking steps within differentiable region $\mathcal{F}$ of both trust region direction p_k and steepest descent direction $-D_k g_k$ are typically very small. In addition, line search steps allowing crossing are also small as the gradient after $x^{(i)} = 0$

hyperplane becomes ascent. Therefore the chosen step d_k is eventually very small compared with trust region step p_k.

3.2 Two Line Search Variations: TR1 and TR2

To ensure the global convergence and local quadratic convergence, we introduce a line search technique allowing crossing points of non-differentiability. In the methods proposed by Coleman and Li [7, 9] for bound-constrained optimization, a line search with a reflective technique accelerates the optimization convergence. Reflection on bounds along s_k allows further decrease of the objective function and improve performance significantly. Similar to the reflective technique, the line search technique which permits crossing makes a difference in the computational performance. In the proposed algorithm, we illustrate two types of line search techniques which are referred to as TR1 and TR2. The difference between these two methods is that TR1 allows crossing only one hyperplane $x^{(i)} = 0$ and TR2 allows crossing as many hyperplanes as possible within the trust region. We want to investigate how different line search techniques can affect the algorithm performance.

To illustrate the two line search techniques, we describe how a direction d is divided by the hyperplanes. We also describe backtracking along this direction. Suppose direction d of infinite length is divided into m segments $(\alpha_1 - \alpha_0)d$, $(\alpha_2 - \alpha_1)d$,... $(\alpha_m - \alpha_{m-1})d$ by $m-1$ breakpoints $(x_k)_1$,... $(x_k)_{m-1}$ as in Figure 3.2. In particular, break points and segments can be calculated below:

$$(x_k)_i = x_k + \alpha_i d, \ \ \alpha_1 \le \alpha_2 \le, ... \le \alpha_{m-1}, \ \ 1 \le i \le m-1,$$

$$\alpha_i \in \{\alpha : \alpha > 0, \ \alpha = -x_k^{(j)}/d^{(j)}, \ 1 \le j \le n\}, \ \ 1 \le i \le m-1, \ \ \alpha_0 = 0, \ \ \alpha_m = +\infty.$$

The objective function approximation $\phi_k(\alpha_l d)$, $1 \leq l \leq m-1$, equals to the sum of $\varphi_k^i((\alpha_i - \alpha_{i-1})d)$, $1 \leq i \leq l$, of all segments:

$$\phi_k(\alpha_l d) = \sum_{i=1}^{l} \varphi_k^i((\alpha_i - \alpha_{i-1})d), \tag{3.12}$$
$$\varphi_k^i((\alpha_i - \alpha_{i-1})d) = (\alpha_i - \alpha_{i-1})(g_k)_{i-1}^T d + (\alpha_i - \alpha_{i-1})^2 \frac{1}{2} d^T M_k d$$

where $(g_k)_i$ is the gradient immediately after crossing the i^{th} breakpoint $(x_k)_i$:

$$(g_k)_0 = g_k, \quad (g_k)_i = (g_k)_{i-1} + (\alpha_i - \alpha_{i-1})M_k d - 2\text{sign}(x_k^{(j)})e_n^j, \ \ 1 \leq i \leq m-1$$

where j is the zero component index of $(x_k)_i$, e_n^j is a vector of size n with all zeros but 1 at the index j. Figure 3.2 shows how d is divided. Note that for any iterate, $x_k^{(j)} \neq 0$, $1 \leq j \leq n$.

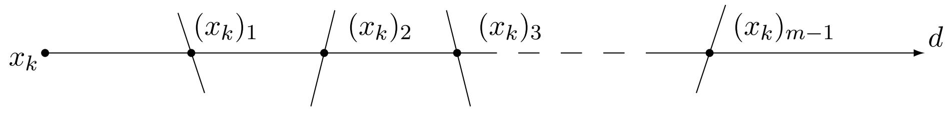


Figure 3.2: Breakpoints $(x_k)_i$ along direction d

To see why equation

$$\phi_k(\alpha_l d) = \sum_{i=1}^{l} \varphi_k^i((\alpha_i - \alpha_{i-1})d)$$

holds, we simply show the case with $l = 2$ here. We use α_2^- to denote a value immediately before α_2. Since $x_k + \alpha_1 d$ is zero at the index of sign change from x_k to $x_k + \alpha_2^- d$, we have $\text{sign}(x_k + \alpha_2^- d)^T(x_k + \alpha_1 d) = \text{sign}(x_k)^T(x_k + \alpha_1 d)$. Then with the fact that M_k is a symmetric matrix, we obtain the following:

$$\begin{aligned}
\phi_k(\alpha_2 d) &= \alpha_2 \nabla f^T d + \|x_k + \alpha_2 d\|_1 - \|x_k\|_1 + \frac{1}{2}\alpha_2^2 d^T M_k d \\
&= \alpha_1 \nabla f^T d + (\alpha_2 - \alpha_1)\nabla f^T d + \text{sign}(x_k + \alpha_2^- d)^T (x_k + \alpha_2 d) - \text{sign}(x_k)^T x_k \\
&\quad + \frac{1}{2}\alpha_1^2 d^T M_k d + \frac{1}{2}(\alpha_2 - \alpha_1)^2 d^T M_k d + \alpha_1(\alpha_2 - \alpha_1) d^T M_k d \\
&= \alpha_1(\nabla f^T + \text{sign}(x_k)^T) d + (\alpha_2 - \alpha_1)(\nabla f^T + \text{sign}(x_k + \alpha_2^- d)^T + \alpha_1 d^T M_k) d \\
&\quad + \frac{1}{2}\alpha_1^2 d^T M_k d + \frac{1}{2}(\alpha_2 - \alpha_1)^2 d^T M_k d \\
&= \alpha_1 (g_k)_0^T d + \frac{1}{2}\alpha_1^2 d^T M_k d + (\alpha_2 - \alpha_1)(g_k)_1^T d + \frac{1}{2}(\alpha_2 - \alpha_1)^2 d^T M_k d \\
&= \varphi_k^1(\alpha_1 d) + \varphi_k^2((\alpha_2 - \alpha_1) d).
\end{aligned}$$

For the cases when $l > 2$, the above equation can also be verified similarly.

For both line search techniques of TR1 and TR2, we use $\phi_k[d]$ to denote the minimum value $\phi_k(\tau_k^* d)$ that we search along d, in which $\tau_k^* \in [0, +\infty)$, denotes a minimizer along d within the trust region. For TR1 method, we allow crossing at most one hyperplane $x^{(i)} = 0$, $1 \le i \le n$:

$$\phi_k[d] \overset{\text{def}}{=} \phi_k(\tau_k^* d) \overset{\text{def}}{=} \min\{\phi_k(\tau d) : \|\tau D_k^{-\frac{1}{2}} d\|_2 \le \Delta_k, 0 \le \tau \le \alpha_2\}.$$

Since τ_k^* is a minimizer within the trust region, it can be obtained as shown in Figure 3.3.

Note that $\tau = -\frac{(g_k)_1^T d}{d^T M_k d}$ solves $(\tau (g_k)_1^T d + \frac{1}{2}\tau^2 d^T M_k d)'_\tau = 0$, thus gives an optimal point along d if $(g_k)_1^T d < 0$. And τ is bounded by $\alpha_2 - \alpha_1$ to make sure the optimal point is within the interval $[x_k, x_k + (\alpha_2 - \alpha_1) d]$. Inequality $(g_k)_1^T d \ge 0$ means that $\varphi_k^2((\alpha_2 - \alpha_1) d)$ becomes ascent along d. In this case, τ_k^* corresponds to the optimal point in interval $[x_k, x_k + \alpha_1 d]$. A special condition we should consider is when $d^T M_k d \le 0$. Under this condition, the objective value approximation ϕ_k is always decreasing along d as long as $(g_k)_i^T d < 0$, $0 \le i \le m-1$. Therefore an optimal point within an interval $[x_k + \alpha_i d, x_k + \alpha_{i+1} d]$ is simply at the end of

```
if  $(g_k)_1^T d \geq 0$ or $\|D_k^{-\frac{1}{2}} \alpha_1 d\|_2 \geq \Delta_k$
    if  $d^T M_k d \leq 0$
        $\tau = \alpha_1$
    else
        $\tau = \min(-\frac{g_k^T d}{d^T M_k d}, \alpha_1)$
    end if
    $\tau_k^* \|D_k^{-\frac{1}{2}} d\|_2 = \min(\Delta_k, \|\tau D_k^{-\frac{1}{2}} d\|_2)$
else
    if  $d^T M_k d \leq 0$
        $\tau = \alpha_2 - \alpha_1$
    else
        $\tau = \min(-\frac{(g_k)_1^T d}{d^T M_k d}, \alpha_2 - \alpha_1)$
    end if
    $\tau_k^* \|D_k^{-\frac{1}{2}} d\|_2 = \min(\Delta_k, \|(\alpha_1 + \tau) D_k^{-\frac{1}{2}} d\|_2)$
end if
```

Figure 3.3: Solve τ_k^* in TR1 method

the interval, $x_k + \alpha_{i+1} d$. With the calculation, it can be guaranteed that $x_k + \tau_k^* d$ is within the trust region and is an optimal point within interval $[x_k, x_k + \alpha_2 d]$.

For TR2 method, we allow crossing as many hyperplanes $x^{(i)} = 0$, $1 \le i \le n$, as possible. We define:

$$\phi_k[d] \overset{\text{def}}{=} \phi_k(\tau_k^* d) \overset{\text{def}}{=} \min\{\phi_k(\tau d) : \|\tau D_k^{-\frac{1}{2}} d\|_2 \le \Delta_k, \tau \ge 0\}.$$

To illustrate the computation steps for τ_k^*, we let $(\tau)_i$ denote the minimizer of ϕ_k that we search along d allowing going across breakpoints until we reach $x_k + \alpha_i d$.

$$(\tau)_i \overset{\text{def}}{=} \operatorname{argmin}_\tau\{\phi_k(\tau d) : \|\tau D_k^{-\frac{1}{2}} d\|_2 \le \Delta_k, 0 \le \tau \le \alpha_i\}.$$

A minimizer τ_k^* which permits crossing as many breakpoints as possible can be obtained as shown in Figure 3.4. With the calculation, we can guarantee that τ_k^* finds the minimizer along d within the trust region.

To avoid landing exactly on a point at which the objective function becomes nondifferentiable, we backtrack slightly as follows. If $x_k + \tau_k^* d$ lands at the lth, $1 \le l \le m-1$, breakpoint, we have $\tau_k^* = \alpha_l$. We use $B_k^*[d]$ to denote a possible step-back from d to keep strict differentiablity. Define

$$B_k^*[d] \overset{\text{def}}{=} \begin{cases} \tau_k^* d & \text{if } x_k + \tau_k^* d \in \mathcal{F}, \\ \alpha_{l-1} d + \theta_k(\tau_k^* - \alpha_{l-1}) d & \text{if } \tau_k^* = \alpha_l, \quad 1 \le l \le m-1. \end{cases}$$

For our algorithm implementation, we use

$$\theta_k = \max(0.95, 1 - VGnorm), \quad VGnorm = \|D(x_k)(\nabla f(x_k) + \operatorname{sign}(x_k))\|_\infty \tag{3.13}$$

where for any vector $x \in \Re^n$

$$\|x\|_\infty = \max_i |x^{(i)}|.$$

```
$(\tau)_0 = 0$
for   $i = 1$ to $m$
    if   $(g_k)_{i-1}^T d \geq 0$ or $\|D_k^{-\frac{1}{2}} \alpha_{i-1} d\|_2 \geq \Delta_k$
        $(\tau)_i = (\tau)_{i-1}$
    else
        if   $d^T M_k d \leq 0$
            $\tau = \alpha_i - \alpha_{i-1}$
        else
            $\tau = \min(-\frac{(g_k)_{i-1}^T d}{d^T M_k d}, \alpha_i - \alpha_{i-1})$
        end if
        $(\tau)_i \|D_k^{-\frac{1}{2}} d\|_2 = \min(\Delta_k, \|(\alpha_{i-1} + \tau) D_k^{-\frac{1}{2}} d\|_2)$
        if   $\phi_k((\tau)_i d) \geq \phi_k((\tau)_{i-1} d)$
            $(\tau)_i = (\tau)_{i-1}$
        end if
    end if
end for
$\tau_k^*(d) = (\tau)_m$
```

Figure 3.4: Solve τ_k^* in TR2 method

Variable $VGnorm$ is a criteria for convergence. It approaches zero when current iterate approaches an optimal solution. We take very cautious backtracking when the iteration converges. With above updates on d, we can guarantee that $x_k + B_k^*[d]$ is a point very close to $x_k + \tau_k^* d$ and does not land exactly on a breakpoint.

For both TR1 and TR2 in Step 2 in Figure 3.1, we compute d_k from

$$\phi_k(d_k) = \min(\phi_k[-D_k g_k], \phi_k[p_k])$$

to satisfy the global convergence requirement (3.10) and local quadratic convergence requirement (3.11). In the following chapters we will demonstrate the performance for algorithm TR1 and TR2 respectively. Intuitively, TR2 method takes fewer iterations than TR1 method as TR2 allows more objective reduction for each iteration. Since line search along a direction d has linear computation complexity, TR2 does not add much cpu time compared with TR1 for one iteration. Therefore TR2 is expected to outperform TR1 in speed. We will investigate how large the difference is computationally.

3.3 A Function Estimation Example

In this section, we will give a simple function estimation example using kernel splines and optimization formulation (3.1). With this example, we illustrate the problem formulation, extended trust region algorithm performance, and several computation issues.

3.3.1 Problem Formulation

As described in the previous chapter, support vector learning methods can be applied to function estimation based on a few sample observations. In our LVF model calibration problem, the sampled input training data are the strikes and maturities $\{(K_i, T_i)\}_{i=1}^{l}$ in the price-time domain. However the function observations, e.g., the real local volatilities

$\{\bar{\sigma}(K_i, T_i)\}_{i=1}^{l}$ at these input points, are not observed. Our local volatility function estimation has to be based on the collected market option data $\{\bar{V}_j^0\}_{j=1}^{m}$, which are indirect observations of local volatility function. To apply support vector regression in estimating the local volatility function, we propose an idea that uses the kernel splines to represent the σ function in (2.7)

$$\begin{aligned}
\sigma((K,T);x) &= \left| \sum_{i=1}^{l} x^{(i)} F((K,T),(K_i,T_i)) + x^{(0)} \right|, \\
F((K,T),(K_i,T_i)) &= [1 + KK_i + \frac{1}{2} \mid K - K_i \mid (K \wedge K_i + K_b)^2 + \frac{(K \wedge K_i + K_b)^3}{3}] \times \\
& \quad [1 + TT_i + \frac{1}{2} \mid T - T_i \mid (T \wedge T_i + T_b)^2 + \frac{(T \wedge T_i + T_b)^3}{3}]
\end{aligned}$$

and then unknown x coefficients are determined by the formulation (2.8)

$$\min_{x \in \Re^{l+1}} \sum_{j=1}^{m} w_j \left(V^0 \left((\bar{K}_j, \bar{T}_j); x \right) - \bar{V}_j^0 \right)^2 + \rho \sum_{i=0}^{l} |x^{(i)}|$$

which we described in Chapter 2. Minimizing the calibration error has similar effect as minimizing the loss function $|\bar{\sigma}(K,T) - \sigma(K,T)|_\epsilon$, where $\bar{\sigma}(K,T)$ stands for the local volatility function that prices the market options. In addition, one-norm regularization on x plays a key role in capacity control and forcing some x coefficients to be zero, yielding a small number of nonzero x.

To demonstrate our problem formulation, we first consider a simple function estimation problem. Suppose that we have observation data $(\bar{s}_1, \bar{y}_1), ..., (\bar{s}_m, \bar{y}_m)$ collected according to:

$$\bar{y}_j = \sin(\bar{s}_j), \quad 1 \le j \le m.$$

We want to estimate a function based on the above data, which is as close to the samples as possible. We assume that the unknown function is represented by a spline kernel

$$h(s;x) = \sum_{i=1}^{l} x^{(i)} F(s,s_i) + x^{(0)},$$
$$F(s,s_i) = 1 + ss_i + \frac{1}{2} \mid s - s_i \mid (s \wedge s_i + s_b)^2 + \frac{(s \wedge s_i + s_b)^3}{3}.$$

The coefficient x in the kernel expansion is obtained by solving the optimization problem

$$\min_{x \in \Re^{l+1}} \sum_{j=1}^{m} (\bar{y}_j - h(\bar{s}_j; x))^2 + \rho \sum_{i=0}^{l} |x^{(i)}| \tag{3.14}$$

with constant $\rho > 0$. Training points $\{s_i\}_{i=1}^{l}$ are chosen to be evenly placed in function definition interval. They are different from the input data $\{\bar{s}_j\}_{j=1}^{m}$ of observations to simulate a regression with indirect measurement of estimated function.

3.3.2 Computation Results and Issues

For the above estimation problem, we let the observations be $\{\bar{y}_j\}_{j=1}^{m} = \sin([0 : 0.4 : 2.8])$ at $\{\bar{s}_j\}_{j=1}^{m} = [0 : 0.4 : 2.8]$, in which $[0 : 0.4 : 2.8]$ is a Matlab command that represents an array starting with 0, ending at 2.8 and with a step 0.4. The training points are placed at $\{s_i\}_{i=1}^{l} = [0.2 : 0.4 : 2.6]$. In Chapter 2, we mention that the kernel function $F(s,s_i)$ for generating splines with an infinite number of knots is twice differentiable. The estimated function is represented by a linear expansion of kernel functions. Therefore the estimated function should be smooth and twice differentiable as well. Figure 3.5 shows the kernel splines $F(s,s_i)$ with respect to different s_i with s and s_i defined in $[0,3]$. In the figure each curve represents a basis function $F(s,s_i)$ for a specific s_i. We can see that curve $F(s,s_i)$ is above $F(s,s_j)$ when $s_i > s_j$.

We compare two different algorithms, TR1 and TR2 trust region methods. Algorithm

TR2 is expected to be faster than TR1 since it allows crossing as many $x^{(i)} = 0$ hyperplanes as possible in each iteration. Thus we first consider the TR2 algorithm to solve the optimization problem (3.14). The exact trust region updating parameters in Figure 3.1 are chosen as

$$\mu = 0.25, \quad \eta = 0.75, \quad \Lambda_U = 0.5(\sqrt{\|x_0\|_1} + 1), \quad \Lambda_L = 10^{-5}$$
$$\gamma_0 = 0.0625, \quad \gamma_1 = 0.5, \quad \gamma_2 = 2, \quad \Delta_0 = \min(0.1 * \|\nabla f(x_0) + \text{sign}(x_0)\|_2, \Lambda_U)$$

where x_0 denotes the initial value of x variable. Our experiments are carried out with the Matlab. The stopping criteria for this problem is

$$\begin{aligned} &\text{either} \quad VGnorm \leq tol, \\ &\text{or} \qquad \phi_k(d_k) \leq qtol \end{aligned}$$

where $VGnorm$ is defined in (3.13). We define the tolerance for $VGnorm$, $tol = 10^{-4}$, and the tolerance for change of objective function approximation, $qtol = 10^{-7}$. We also impose an upper bound of 600 on the number of iterations.

In trust region algorithm implementation, the gradient and Hessian matrix of $f(x)$ should be evaluated. For the example of sin function estimation, $f(x)$ in optimization problem (3.1) is formulated as:

$$f(x) = \frac{1}{\rho} \sum_{j=1}^{m} (\bar{y}_j - h(\bar{s}_j; x))^2.$$

To express the gradient and Hessian matrix with some ease, we define a vector-valued residual function $R : \Re^{l+1} \rightarrow \Re^m$ where component j of R is given by $(\frac{2}{\rho})^{\frac{1}{2}}(\bar{y}_j - h(\bar{s}_j; x))$,

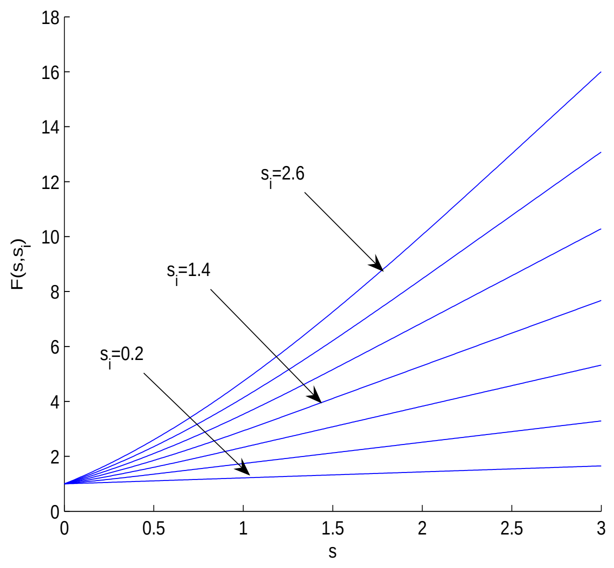


Figure 3.5: Kernel function $F(s, s_i)$ for $s_i = [0.2 : 0.4 : 2.6]$

for $j = 1, ..., m$. Therefore we have:

$$f(x) = \frac{1}{2}R(x)^T R(x).$$

We define $J(x)$ as the Jacobian matrix of R with respect to x:

$$J(x) = \begin{bmatrix} \left(\nabla R^{(1)}\right)^T \\ \vdots \\ \left(\nabla R^{(m)}\right)^T \end{bmatrix}.$$

In our optimization algorithm, we use automatic differentiation [11] to compute the $m \times (l+1)$ matrix $J(x)$. The gradient of $f(x)$ is:

$$\nabla f(x) = \sum_{j=1}^{m} R^{(j)}(x) \nabla R^{(j)}(x) = J(x)^T R(x).$$

ρ	1	10^{-2}	10^{-4}	10^{-6}
iterations	16	14	22	42
error	0.817	$5.9 * 10^{-3}$	$1.87 * 10^{-4}$	$1.25 * 10^{-6}$
# of "SVs"	2	2	3	7

Table 3.1: sin function estimation performance with TR2 method

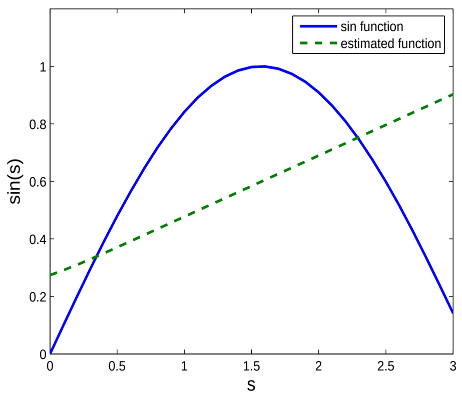
Furthermore the Hessian matrix of $f(x)$ can be expanded:

$$\nabla^2 f(x) = \sum_{j=1}^{m} \nabla R^{(j)}(x) \nabla R^{(j)}(x)^T + R^{(j)}(x)\nabla^2 R^{(j)}(x) = J(x)^T J(x) + \sum_{j=1}^{m} R^{(j)}(x)\nabla^2 R^{(j)}(x).$$

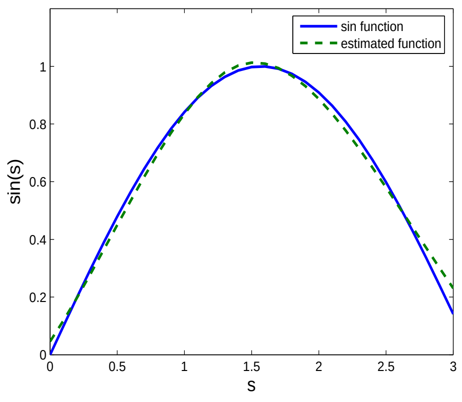
Note that our optimization problem is expected to be a small-residual problem, i.e., $R(x^*) \approx 0$, when approaching an optimal solution x^*. Therefore $\sum_{j=1}^{m} R^{(j)}(x)\nabla^2 R^{(j)}(x)$ is omitted in the Hessian matrix of $f(x)$ because it is much smaller compared with the other term $J(x)^T J(x)$. Thus we have the approximate Hessian matrix of $f(x)$:

$$\nabla^2 f(x) \approx J(x)^T J(x). \tag{3.15}$$

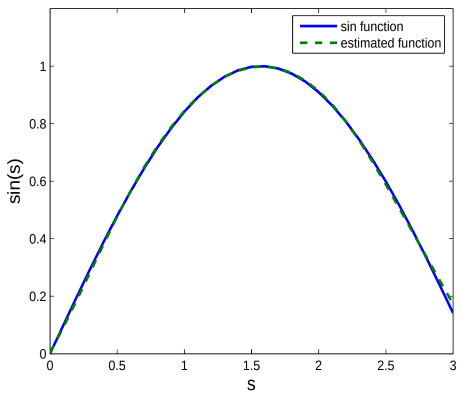
We first demonstrate the results of the sin function estimation with starting $x_0^{(i)} = 0$, $0 \leq i \leq l$. Table 3.1 shows the TR2 algorithm performance, calibration errors $\sum_{j=1}^{m}(\bar{y}_j - h(\bar{s}_j; x))^2$ and number of "support vectors" for different choices of parameter ρ. We consider those training vectors that have nonzero coefficients $x^{(i)}$ as "support vectors", although they are not strictly support vectors in SVR. Figure 3.6 shows plots of estimated functions compared with sin function. In general, computation with larger ρ takes fewer iterations, yields larger calibration errors and leads to smaller number of "SVs". The "support vectors" are judged by examining the corresponding $x^{(i)}$, $1 \leq i \leq l$, parameters of training points. Table 3.2 lists the $x^{(i)}$ values after optimization termination when we choose $\rho = 1$. It may not be obvious for us to see if a $x^{(i)}$ coefficient is zero. We decide the "SVs" in a standard way. The largest absolute value of $x^{(i)}$ is $|x^{(2)}| = 0.27412513459983$ in Table 3.2. We consider



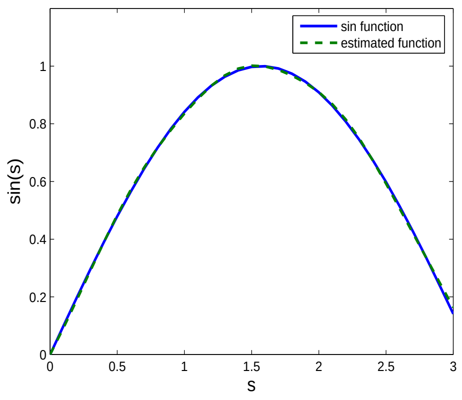
(a) $\rho = 1$



(b) $\rho = 10^{-2}$



(c) $\rho = 10^{-4}$



(d) $\rho = 10^{-6}$

Figure 3.6: sin function estimation for different ρ

point	x coefficient
0.2	0.00000105763502
0.6	0.27412513459983
1	-0.00000000063126
1.4	-0.00000000008146
1.8	-0.00000000002170
2.2	0.00000000000003
2.6	-0.00018903984863

Table 3.2: Coefficients x after optimization for sin function estimation with training points $[0.2 : 0.4 : 2.6]$ and $\rho = 1$

those $|x^{(i)}|$ which is less than $10^{-4} \times |x^{(2)}|$ as zeros. According to this criteria, in Table 3.2 we find only $x^{(2)}$ and $x^{(7)}$ are non-zeros. The training vectors of non-zero $x^{(i)}$ are referred to as "support vectors". Therefore we have two "SVs", which are training points at $s_2 = 0.6$ and $s_7 = 2.6$.

As analyzed previously, the TR2 algorithm is expected to be more efficient than the original TR1 algorithm in that the line search in TR2 allows going across as many breakpoints as possible to search the minimizer along trust region solution. Normally the TR2 method can obtain larger objective value reduction than TR1 method for one iteration under the same computation condition. Therefore we expect that using TR2 algorithm costs fewer iterations than TR1 algorithm. By setting initial $x_0^{(0)} = 0$, 0.5 and 1, and $x_0^{(i)} = 0$ for all $1 \leq i \leq l$, we run the optimization problem with TR1 and TR2 algorithms respectively. The same optimal points are found for the same parameter settings, but TR2 method outperforms the TR1 method in CPU time. Table 3.3 lists the number of iterations for these two methods. We find that all TR2 computations take fewer iterations than TR1. Especially when $x_0^{(0)} = 1$, TR2 converges much faster than TR1 method.

The main contribution of our proposed L_1 norm optimization formulation is to control the conflict between minimizing the calibration error and making a stable calibration model. We expect that the larger the regularization parameter ρ is, the larger the calibration error

Parameter\Method		TR2	TR1
$x_0^{(0)} = 0$	$\rho = 1$	16	16
	$\rho = 10^{-2}$	14	18
	$\rho = 10^{-4}$	22	26
	$\rho = 10^{-6}$	42	46
$x_0^{(0)} = 0.5$	$\rho = 1$	17	18
	$\rho = 10^{-2}$	10	14
	$\rho = 10^{-4}$	16	19
	$\rho = 10^{-6}$	27	31
$x_0^{(0)} = 1$	$\rho = 1$	35	33
	$\rho = 10^{-2}$	10	30
	$\rho = 10^{-4}$	15	33
	$\rho = 10^{-6}$	24	44

Table 3.3: Comparison between TR2 and TR1 for different parameter ρ and starting x in numbers of iterations

$\epsilon \setminus \rho$	1	10^{-2}	10^{-4}	10^{-6}
$\pm 2\%$	2	2	4	7
$\pm 5\%$	2	2	5	7
$\pm 10\%$	2	3	5	7

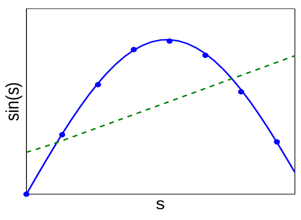
Table 3.4: Number of "support vectors" for different ρ and noise level ϵ

is allowed. Thus the lower the number of "SVs" is, and the simpler and more stable the calibrated model is. To demonstrate how stability is affected by the choice of parameter ρ, we introduce noise to the collected data in the sin function estimation example. We add $\epsilon = \pm 2\%$, $\pm 5\%$, and $\pm 10\%$ uniform distribution error to the observed data respectively. Particularly, we have the corrupted data as

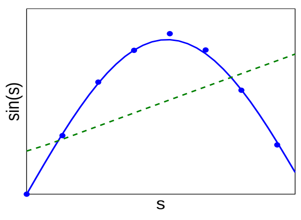
$$s = [0 : 0.4 : 2.8], \;\; y_i = \sin(s_i) * (1 + \kappa * (\text{rand} - 0.5)), \;\; \kappa = 0.04, \; 0.1, \; 0.2, \quad 1 \leq i \leq 7$$

where rand returns a uniform random sample in $[0, 1]$.

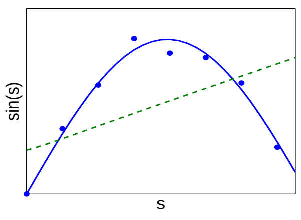
Figure 3.7 lists the estimated functions for different choices of regularization parameter ρ and noise level. We find that larger ρ yields larger calibration error but more robust



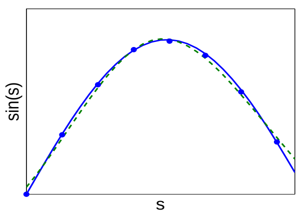
(a) $\rho = 1, \epsilon = \pm 2\%$



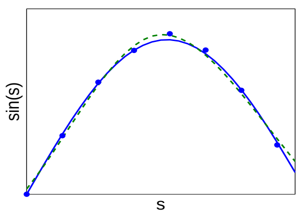
(b) $\rho = 1, \epsilon = \pm 5\%$



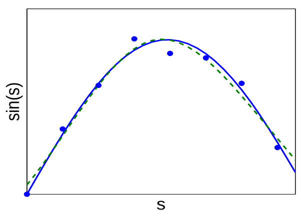
(c) $\rho = 1, \epsilon = \pm 10\%$



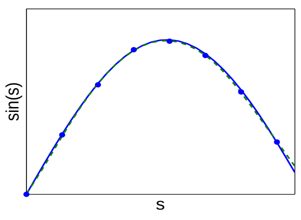
(d) $\rho = 10^{-2}, \epsilon = \pm 2\%$



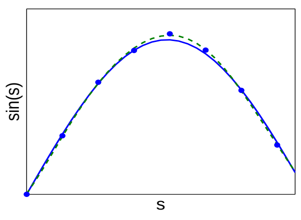
(e) $\rho = 10^{-2}, \epsilon = \pm 5\%$



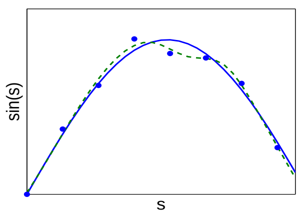
(f) $\rho = 10^{-2}, \epsilon = \pm 10\%$



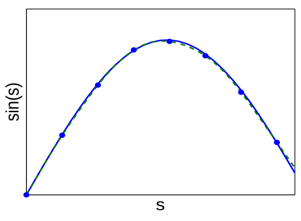
(g) $\rho = 10^{-4}, \epsilon = \pm 2\%$



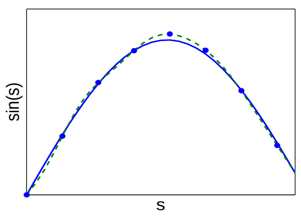
(h) $\rho = 10^{-4}, \epsilon = \pm 5\%$



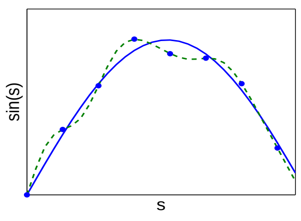
(i) $\rho = 10^{-4}, \epsilon = \pm 10\%$



(j) $\rho = 10^{-6}, \epsilon = \pm 2\%$



(k) $\rho = 10^{-6}, \epsilon = \pm 5\%$



(l) $\rho = 10^{-6}, \epsilon = \pm 10\%$

Figure 3.7: Robustness of sin function estimation with different ρ and noise level ϵ Rows from top to bottom: $\rho = 1,\ 10^{-2},\ 10^{-4},\ 10^{-6}$. Columns from left to right: $\epsilon = \pm 2\%,\ \pm 5\%,\ \pm 10\%$. (solid line: sin function; dashed line: estimation function; circle: sample data)

the estimated function, i.e., the function is less affected by the noise. Table 3.4 shows the number of "support vectors" for the corresponding implementations. It also verifies that the larger parameter ρ, the smaller the number of "SVs" and thus leads to more robust estimation $h(s)$.

Chapter 4

LVF Calibration Examples

To illustrate accuracy and stability of the proposed method for LVF model calibration, we investigate a numerical example with a synthetical option data and then a real option data in more details. Various computational issues such as data centering for kernel expression, initial guess of a local volatility function, and influence of regularization parameter ρ are discussed in this chapter.

4.1 Computation Set-up

In Chapter 2, we introduce a kernel spline approach (2.7) to represent local volatility function with appropriately chosen training vectors $\{(K_i, T_i)\}_{i=1}^{l}$ in the strike-maturity domain. Coefficient vector $x \in \Re^{l+1} = (x^{(0)}, ..., x^{(l)})$ in (2.7) is determined by the proposed optimization problem (2.8).

There are a variety of numerical techniques used for option pricing such as binomial/trinomial trees, Monte Carlo methods, and numerical methods (finite difference or finite element methods) for solving partial differential equations. We use Crank-Nicolson finite difference method to determine initial model option values $\{V_j^0\}_{j=1}^m$ by solving the adjoint equation:

$$\frac{\partial V^0(K,T)}{\partial T} - \frac{1}{2}\sigma^2(K,T)K^2\frac{\partial^2 V^0(K,T)}{\partial K^2} + (r-q)K\frac{\partial V^0(K,T)}{\partial K} + qV^0(K,T) = 0. \quad (4.1)$$

Without loss of generality, we assume available initial prices are call option prices and use the following boundary conditions:

$$\begin{aligned}
&\lim_{K\to+\infty} V^0(K,T) = 0\\
&\frac{\partial V^0(K,T)}{\partial T} + qV^0(K,T) = 0, \text{ at } K = 0\\
&V^0(K,T) = \max(S_0 - K, 0), \ \ at \ \ T = 0.
\end{aligned}$$

Computation domain for the adjoint equation is $\mathcal{D} = [0, K_{max}) \times [0, T_{max}) \subset \Re^2$, where $K_{max} = 2S_0$ and T_{max} is the maximum maturity among all given market options, and S_0 is the initial underlying price. A uniform grid discretization is used in $\mathcal{D}$ with $N \times M$ nodal points:

$$\begin{aligned}
K_i^N &= i\tfrac{K_{max}}{N-1}, \quad i = 0, ..., N-1,\\
T_j^M &= j\tfrac{T_{max}}{M-1}, \quad j = 0, ..., M-1.
\end{aligned} \quad (4.2)$$

To construct the kernel spline representation of a local volatility function, l training vectors need to be selected in the region $\mathcal{D}$. Since the option price observations rather than local volatility function value observations are given, these training vectors do not necessarily have to correspond to the strikes and maturities of the option price observations. In addition, the total number of training vectors does not have to correspond to the total number of observations. The total number of nonzero coefficients, i.e., relevant training vectors, can be determined by minimizing $\|x\|_1$ component in the objective function. However a large number of training vectors l can increase computational cost of the optimization, we choose the number of the vectors l to be approximately the number of market option price observations m. In the following test examples, we choose $l < m$ in particular. Later on we will further investigate the robustness of model calibration if we choose $l \gg m$ training points.

The model option price depends on the unknown local volatility surface. However this dependence is not uniform in the computation region $\mathcal{D}$. The option price depends little on the local volatility values at very small or large K far from initial underlying asset S_0. We denote $\mathcal{D}_S = [0.7S_0, 1.3S_0] \times [0, T_{max}]$ as a region centered around S_0 in which the local volatility value is significant in option pricing and hedging. Therefore in our experiment we place training vectors of kernel spline uniformly in this significant region.

For kernel methods, data centering in a feature space can improve the performance of a kernel algorithm. Basically, in a kernel method, a Gram matrix is formed with the pairwise inner product of mapped data. The Gram matrix will be ill-conditioned if the origin is far away from the training data [22, 3]. Therefore data centering preprocess is used for the numerical reason in support vector learning algorithms. Our LVF calibration approach follows some properties of SVR and in our computation context the strike and maturity values are not around zero. Particularly the strikes are around S_0 which are far away form zero. Therefore in our method, we shift the data of strikes and maturities in the significant region $\mathcal{D}_s$. We use constants $C_K > 0$ and $C_T > 0$ to denote the centers for strike K and maturity T, and $Q_K > 0$ and $Q_T > 0$ to denote the scales for K and T. We want the strikes and maturities to be translated and scaled so that they are centered around 0 and have deviation from center around 1. Note that the training points and testing points should be centered consistently. The training points in kernel expansion (2.7) are (K_i, T_i); the testing points are variables (K, T). Using data centering, the kernel spline representation of local volatility is as follows:

$$\sigma((K,T);x) = \left| \sum_{i=1}^{l} x^{(i)} F((K_C, T_C), (K_{Ci}, T_{Ci})) + x^{(0)} \right|,$$

with

$$F((K_C, T_C), (K_{Ci}, T_{Ci})) =$$
$$\left(1 + K_C K_{Ci} + \frac{1}{2} \mid K_C - K_{Ci} \mid (K_C \wedge K_{Ci} + K_b)^2 + \frac{(K_C \wedge K_{Ci} + K_b)^3}{3}\right) \times$$
$$\left(1 + T_C T_{Ci} + \frac{1}{2} \mid T_C - T_{Ci} \mid (T_C \wedge T_{Ci} + T_b)^2 + \frac{(T_C \wedge T_{Ci} + T_b)^3}{3}\right)$$

in which (K_C, T_C) and (K_{Ci}, T_{Ci}) are centered and scaled 2-dimensional variables for (K, T) and (K_i, T_i) in $\mathcal{D}_s$. And these centered and scaled training variables are defined in interval $[-K_b, +\infty) \times [-T_b, +\infty)$. The centering and scaling of a point (K, T) is given below:

$$K_C = \frac{K - C_K}{Q_K}, \quad T_C = \frac{T - C_T}{Q_C}.$$

For minimizing the nonlinear objective problem (2.8), we employ the trust region algorithm TR2 described in Chapter 3. In this algorithm, the gradient and Hessian matrix of $f(x)$ should be evaluated. Similarly as in Chapter 3, we introduce a residual function R, use the automatic differentiation to compute the Jacobian J of the residual function and approximate Hessian matrix with $J^T J$. For our LVF model calibration in particular, $f(x)$ in the objective function (3.1) of the trust region algorithm is formulated as:

$$f(x) = \frac{1}{\rho} \sum_{j=1}^{m} w_j (V^0(\bar{K}_j, \bar{T}_j; x) - \bar{V}_j)^2$$

where $w_j > 0$, $j = 1, ..., m$, are weights which can be used to ensure desired accuracy for option values calibration. The vector-valued residual function is $R : \Re^{l+1} \rightarrow \Re^m$, where component $R^{(j)}$ is given by $(\frac{2w_j}{\rho})^{\frac{1}{2}} (V^0(\bar{K}_j, \bar{T}_j; x) - \bar{V}_j^0)$, for $j = 1, ..., m$. Therefore we have $f(x) = \frac{1}{2} R(x)^T R(x)$. Then we similarly approximate the Hessian matrix with $J^T J$ since the residual function is close to zero, i.e. $R(x^*) \approx 0$, when the optimization computation is

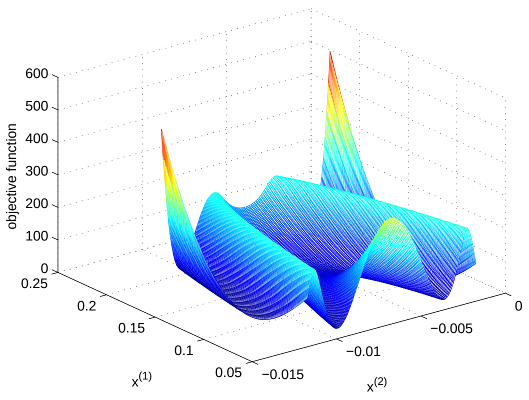


Figure 4.1: Nonlinearity of objective function $f(x) + \|x\|_1$ with 2-variable vector x

close to an optimal solution x^*. In fact finite-difference technique can also approximate the Hessian matrix of $f(x)$.

The functional $f(x)+\|x\|_1$ can be highly nonlinear with respect to the unknown variable x. Figure 4.1 shows the mesh plot of $f(x)+\|x\|_1$ against $(x^{(1)}, x^{(2)})$ when only two training points are placed and the constant term $x^{(0)}$ is set to zero for simplicity. The parameter $\rho = 1$ and weights $w_j = 1$ for $1 \leq j \leq m$. Figure 4.1 can be accounted for by the nonlinearity of the option value with respect to the local volatility surface as well as the nonlinearity of the local volatility with respect to the parameters x due to the kernel spline representation. Therefore the performance of the optimization algorithm can be highly influenced by a good priori of initial x_0. We choose x_0 such that the local volatility surface given by x_0 resembles the dependence given by the volatility smile. In particular, suppose m market options have implied volatilities $\{\tilde{\sigma}_j\}_{j=1}^m$, we choose the initial local volatility surface such that its values at $\{(\bar{K}_j, \bar{T}_j)\}_{j=1}^m$ are as close to $\{\tilde{\sigma}_j\}_{j=1}^m$ as possible. Specifically, we choose x_0 to satisfy the following as much as possible:

$$\sigma((\bar{K}_j, \bar{T}_j); x_0) \equiv \sum_{i=1}^{l} x_0^{(i)} F((\bar{K}_j, \bar{T}_j), (K_i, T_i)) + x_0^{(0)} = \tilde{\sigma}_j, \quad 1 \leq j \leq m. \tag{4.3}$$

(4.3) is a set of m linear equations for vector x_0 of $l+1$ unknown variables. If $l+1 < m$, it is an overdetermined problem and can be easily solved. On the other hand, we minimize the one norm of x in the objective function. Thus it would be more efficient if the initial x_0 are not too large. Therefore we solve the following minimization problem which minimizes the 2-norm of difference of the left side and right side of (4.3) with constraints on x_0:

$$\begin{aligned} &\min_{x_0 \in \Re^{l+1}} \sum_{j=1}^{m} \|\sigma((K_j, T_j); x_0) - \tilde{\sigma}_j\|_2^2 \\ &l_b \le x_0 \le u_b \end{aligned} \tag{4.4}$$

where $l_b < 0$ and $u_b > 0$ are bounds for x_0. For our calibration example, this linear optimization problem can be solved efficiently with Matlab toolbox within one second cpu time.

Finally in the trust region algorithm, we set the stopping criteria in Figure 3.1 to be $tol = 2 * 10^{-3}$ or $qtol = 10^{-7}$. In the following experiments we will see the results give accurate approximation of the local volatility function with the chosen stopping criteria.

4.2 Synthetic Data Examples

In order to illustrate computation issues such as data centering and scaling, choice of regularization parameter, good priori choice of x and model robustness against data noise, we consider the following calibration examples used in [10].

Suppose that the underlying asset follows an absolute diffusion process (1.1) in which the local volatility function σ^* is a function of the underlying asset price only:

$$\sigma^*(S,t) = \frac{\sigma_c}{S} \tag{4.5}$$

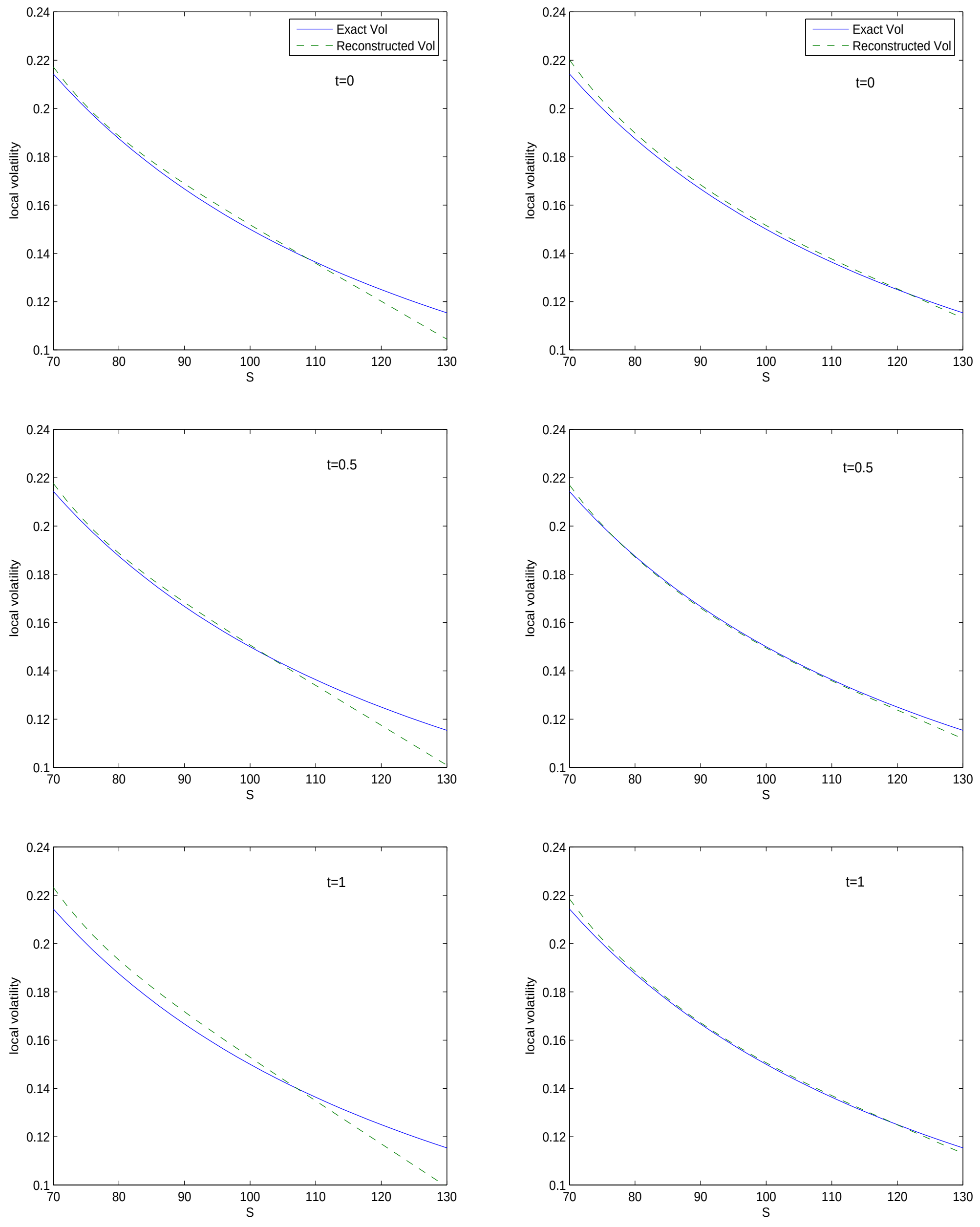
with $\sigma_c = 15$. In addition, we let the initial asset price $S_0 = 100$, the risk free interest rate $r = 0.05$, and the dividend rate $q = 0.02$. We assume that 22 European call options values generated from the model (4.5) are market option prices. Half of the options have 0.5 year maturity with strikes $[90 : 2 : 110]$ and the other half have 1 year maturity with the same strikes. The 22 option prices are obtained from adjoint equation (4.1) with Crank-Nicolson method, in which the local volatility function $\sigma(K, T) = \frac{\sigma_c}{K}$. The discretization parameters are set as $M = 101$ and $N = 51$ in (4.2) for strikes and maturities respectively.

For a good priori of x, we set $l_b = -0.8$ and $u_b = 0.8$ for lower and upper bounds in (4.4) respectively. Unless specified explicitly, weight w_j for each option is unity.

We let the number of training vectors $l = 18$ and place these points evenly in the significant region $\mathcal{D}_s$ of price-time domain. First we place them on the grid of $[80 : 5 : 120] \times [0.25, 0.75]$. The centering parameters are set at $C_K = S_0 = 100$, $Q_K = 20$ for strike and $C_T = 0.5$, $Q_T = 1$ for maturity. For regularization parameter $\rho = 1$, the optimization method takes 28 iterations and the calibration error, i.e., the sum of squares of the option price residuals $\sum_{j=1}^{m}(V_j^0 - \bar{V}_j^0)^2$ is 3.3×10^{-4}.

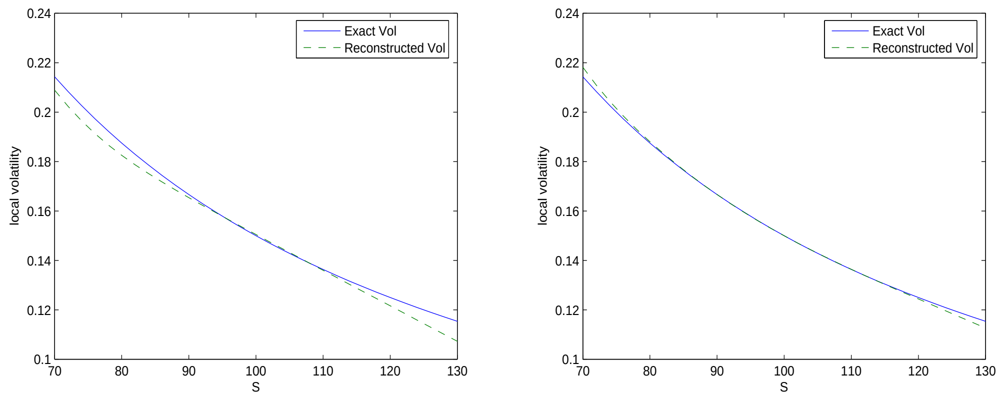
Figure 4.2 shows the reconstructed local volatility compared with the model local volatility (4.5) when $\rho = 1$ and $\rho = 10^{-2}$. It demonstrates that the local volatility is reconstructed accurately in the region of $[70, 130] \times [0, 1]$. Three plots for $t = 0$, $t = 0.5$ and $t = 1$ show the reconstruction is more accurate when t is close to 1. Compared with the plots of $\rho = 1$, $\rho = 10^{-2}$ gives better reconstruction with less calibration error at 1.6×10^{-6}.

We now consider the local volatility function as a deterministic function of underlying price only for simplicity. In this case, the kernel spline representation of volatility is



Left plot: $\rho = 1$ Right plot: $\rho = 0.01$

Figure 4.2: 2-dimensional LVF reconstruction for noise-free data with uniform weights



Left plot: $\rho = 1$ Right plot: $\rho = 0.01$

Figure 4.3: 1-dimensional LVF reconstruction for noise-free data with uniform weights

simplified as:

$$\sigma((K,T);x) = \left| \sum_{i=1}^{l} x^{(i)} F(K,K_i) + x^{(0)} \right|,$$
$$F(K,K_i) = (1 + KK_i + \frac{1}{2} \mid K - K_i \mid (K \wedge K_i + K_b)^2 + \frac{(K \wedge K_i + K_b)^3}{3}).$$

We place $l = 9$ training data points on grid $[80 : 5 : 120]$ since the local volatility function estimation is based on strike only. The parameter set-up of data centering and the initial guess x_0 is similar to the two-dimensional experiment except that we only use parameters related to strikes. Figure 4.3 shows the one-dimensional local volatility reconstruction when $\rho = 1$ and $\rho = 0.01$. In general the one-dimensional local volatility function reconstruction is computationally easier. It takes 20 and 41 iterations for $\rho = 1$ and $\rho = 0.01$ respectively. Parameter $\rho = 0.01$ gives more accurate reconstruction of local volatility than $\rho = 1$.

Table 4.1 illustrates the performance of our approach with different training point placement. In general, one-dimensional local volatility reconstruction is more efficient since it has less number of unknowns, therefore less computation complexity. A larger ρ means we require smaller $\|x\|_1$ and allow larger calibration error. Therefore we have fewer "support vectors" and the model calibration is more robust. This can be further seen when we introduce noise to the option data. Usually larger ρ also takes fewer iterations.

The "support vectors" are decided by examining the corresponding x parameters of training vectors as illustrated in the last chapter. Table 4.2 lists the x values at termination when we choose one-dimensional training vectors $[80 : 5 : 120]$ and $\rho = 1$. The largest coefficient in magnitude is $|x^{(1)}| = 0.08341018779593$. Therefore we consider those $|x^{(i)}|$ less than $10^{-4} * |x^{(1)}|$ as zeros. The training vectors of non-zero $x^{(i)}$ are referred to as "support vectors". Therefore training points with strikes at 80, 105 and 120 are "support vectors" in Table 4.2.

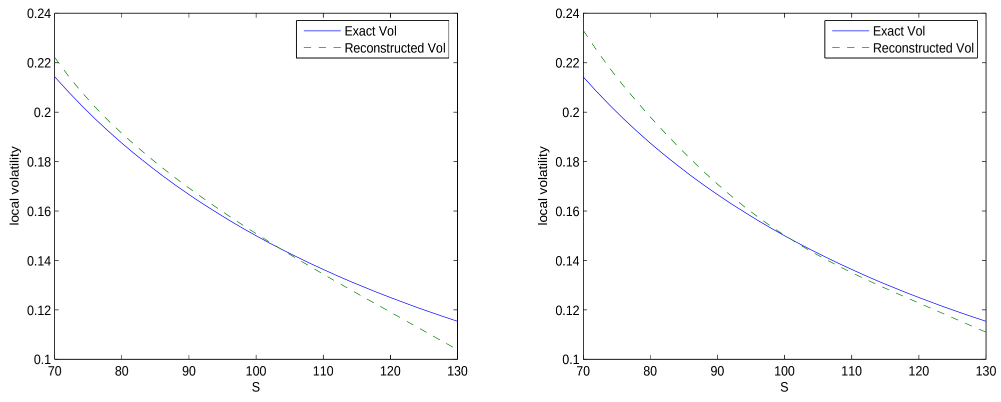
	$[80 : 5 : 120]$		$[80 : 5 : 120] \times [0.25, 0.75]$	
	$\rho = 1$	$\rho = 0.01$	$\rho = 1$	$\rho = 0.01$
iteration	20	41	28	50
error	$5.6 * 10^{-4}$	$1.9 * 10^{-6}$	$3.3 * 10^{-4}$	$1.6 * 10^{-6}$
# of "SVs"	3	5	11	14

Table 4.1: LVF reconstruction performance for synthetic option example

point	x coefficient
80	0.08341018779593
85	0.00000013710573
90	0.00000034895936
95	0.00000477462809
100	0.00000000000990
105	0.01582733622649
110	-0.00000000001277
115	0.00000000055154
120	0.00453346330431

Table 4.2: Parameter x at optimization termination for 1-dimensional reconstruction on training points $[80 : 5 : 120]$

In practice the market data is collected within a bid-ask spread. We investigate the in-



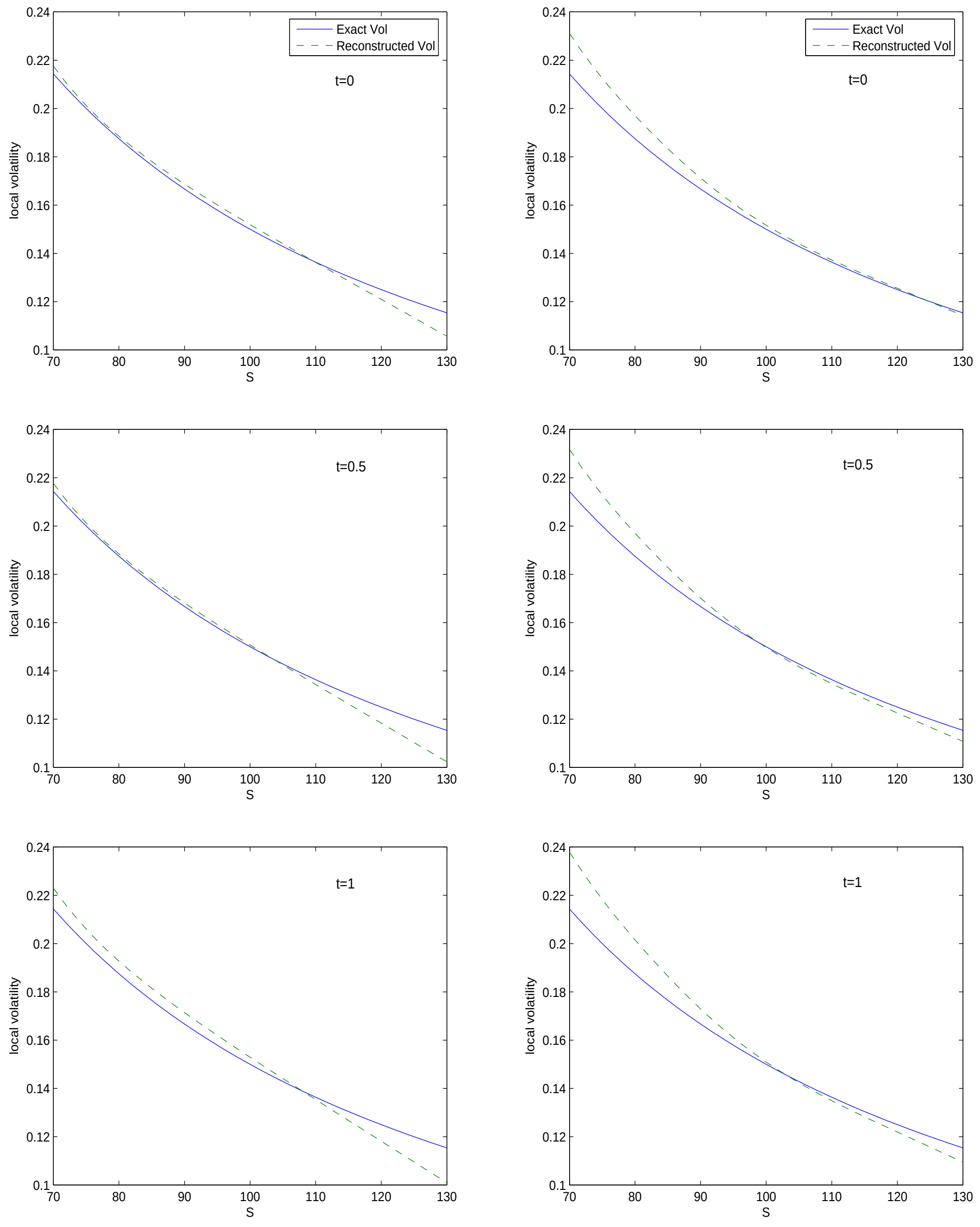
Left plot: $\rho = 1$ Right plot: $\rho = 0.01$

Figure 4.4: 1-dimensional LVF reconstruction for noisy data with uniform weights

fluence of data noise by adding uniformly distributed noise to the synthetic market option prices. Considering that OTM option prices are less reliable in the market, we add larger perturbation for these options. In particular, the option price used in the previous example is multiplied by:

$$\begin{cases} (1 + 0.04 * (\text{rand} - 0.5)), & \text{if } (\bar{K}, \bar{T}) = (108, 0.5), (110, 0.5), (110, 1); \\ (1 + 0.02 * (\text{rand} - 0.5)), & \text{otherwise.} \end{cases}$$

We choose training points $[80 : 5 : 120] \times [0.25, 0.75]$ and $[80 : 5 : 120]$ for two-dimensional and one-dimensional experiments respectively. Other parameters setting is the same as the noise-free examples. The optimization algorithm takes similar number of iterations as before, but the calibration error is larger. Figure 4.4 and 4.5 demonstrate the reconstructed local volatility function with one-dimensional and two-dimensional kernel spline representation. From these plots we find that the reconstructed local volatility is still smooth for the noisy option data. Larger ρ seems to give more robust model calibration, i.e., the local volatility function has less variation with noise perturbation.



Left plot: $\rho = 1$ Right plot: $\rho = 0.01$

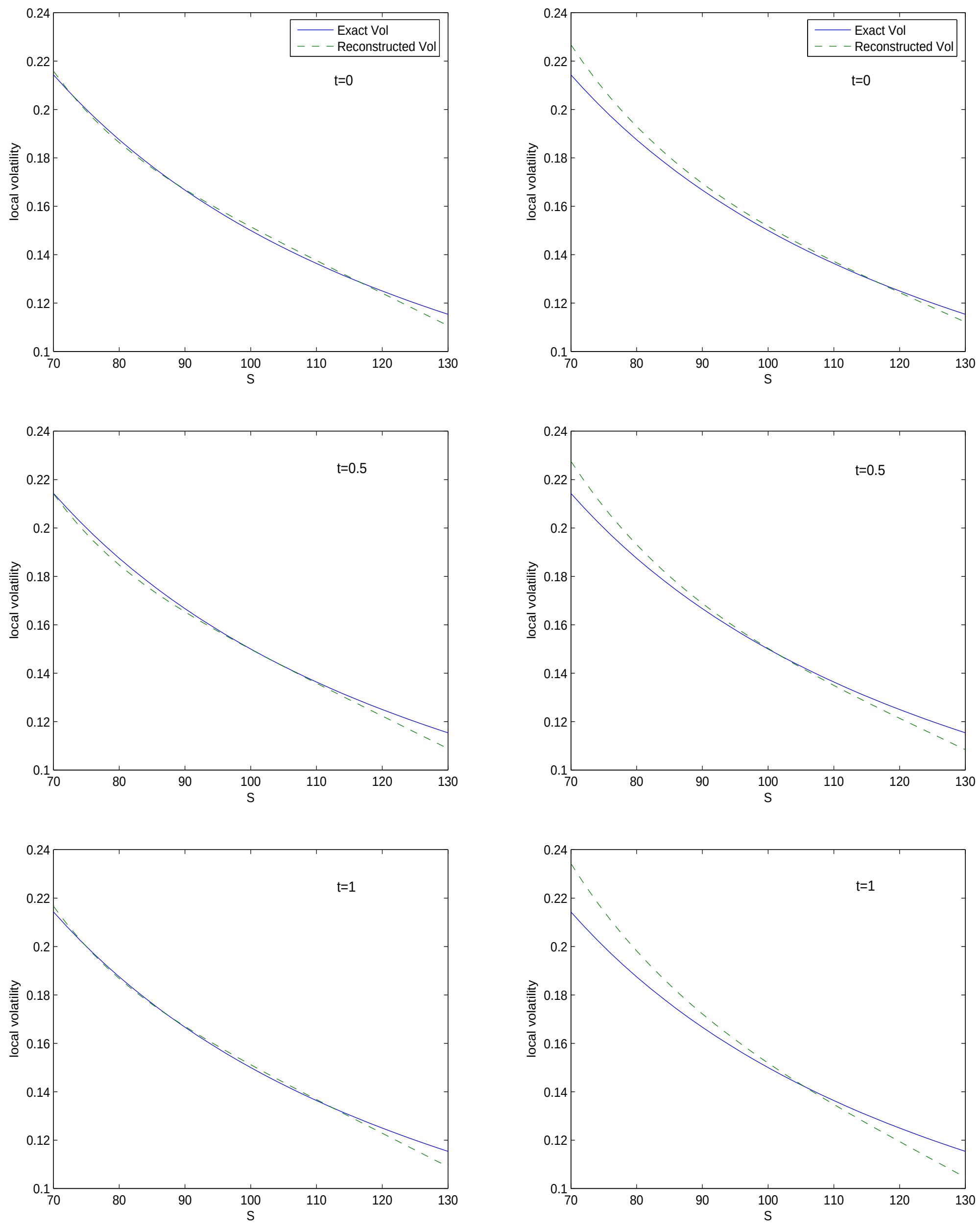
Figure 4.5: 2-dimensional LVF reconstruction for noisy data with uniform weights

More experiments are implemented with different training points placement. We place 18 points on grid $[76 : 6 : 124] \times [0.25, 0.75]$ for two-dimensional cases and 9 points on grid $[76 : 6 : 124]$ for one-dimensional cases. Good reconstruction of local volatility function is also obtained for noise-free data and the parameter ρ influences in the same way as the previous experiments. Moreover, we experiment with more training points on grid $[70 : 5 : 130] \times [0.2 : 0.2 : 0.8]$. We choose the parameter $\rho = 1$ and compute with both error-free data and data with the same noise as previous experiments. The optimization takes a similar number of iterations and calibration errors are close to previous two-dimensional examples. The reconstructed local volatility surfaces are graphed in Figure 4.6. It can be observed that it demonstrates the accuracy of the function approximation and robustness with noise information added. Furthermore, the appearance of local volatility surface is fairly close to the previous example. This experiment shows that our model calibration is relatively insensitive against the number and placement of training points.

4.3 LVF Calibration from S&P 500 Index Option Data

In spite of its attractiveness (such as market completeness), it is unlikely that a local volatility function model exactly specifies an underlying market price. Some of the criticisms of the local volatility model is that the calibrated local volatility often have unreasonable oscillations. In addition, the calibrated local volatilities calibrated within a small time window seem to have unreasonably large change.

The previous synthetic option example reveals that the reconstructed local volatility function is smooth and robust against some data perturbation. We now consider the local volatility calibrated from our proposed method using the S&P 500 Index Option data. Specifically, we are interested in the characteristic and stability of the calibrated local volatility surface.



Left plot: noise-free data Right plot: noisy data

Figure 4.6: 2-dimensional LVF reconstruction with uniform weights and training points more than available options

Maturity \ Strike	85%	90%	95%	100%	105%	110%	115%	120%
0.695	.172	.157	.144	.133	.118	.104	.100	.101
1	.171	.159	.150	.138	.128	.115	.107	.103
1.5	.169	.160	.151	.142	.133	.124	.119	.113

Table 4.3: Implied volatilities of S&P 500 Index Options in Oct 95 of maturities (year) and strikes (% of underlying price)

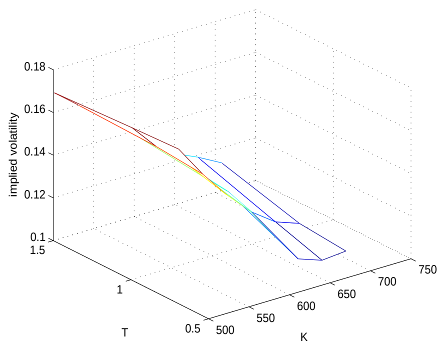
Maturity \ Strike	85%	90%	95%	100%	105%	110%	115%	120%
0.695	101.9	76.26	52.76	32.75	16.47	6.02	1.93	.62
1	108	83.6	61.55	41.57	25.41	12.75	5.5	2.13
1.5	117.2	94.37	73.14	53.97	37.33	23.68	14.3	7.65

Table 4.4: Prices of S&P 500 Index Options in Oct 95 of maturities (year) and strikes (% of underlying price)

In this section we first calibrate a LVF model with a real market option data set in Oct 1995. We show that the calibrated local volatility function resembles the implied volatility surface. And we add some random noise to the option prices and our experiment shows the LVF calibration is robust against noisy data. In addition we calibrate LVF with market options on two close dates, March 02, 2004 and April 05, 2004. We find that the two calibrated local volatility surfaces appear fairly similar. We also demonstrate effects of introducing individual weights in calibration.

4.3.1 Calibration from Data in Oct 1995

Table 4.3 shows the implied volatilities for S&P 500 Index Options in October 1995. This data is also used in [10]. The prices for these options are listed in Table 4.4. The option pricing parameters are: initial asset $S_0 = \$590$, interest rate $r = 0.06$ and dividend rate $q = 0.0262$. The discretization parameters are $M = 101$ for the strike axis and $M = 76$ for the maturity axis. The market option prices are computed from implied volatilities with Matlab *blsprice* function based on the constant volatility Black-Scholes formula. We



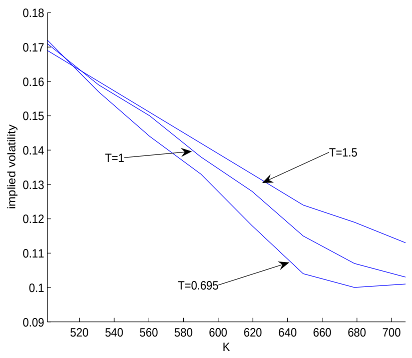
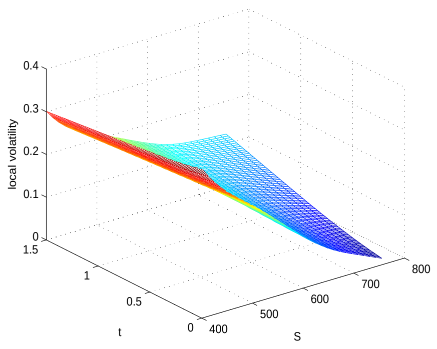
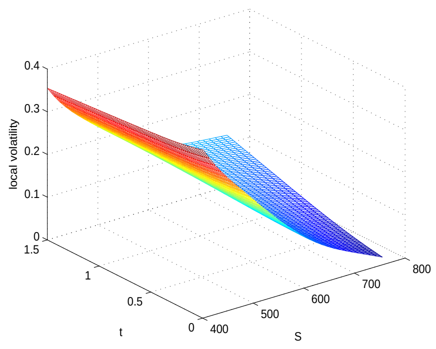


Figure 4.7: Implied volatility surface of S&P 500 Index Options in Oct 95



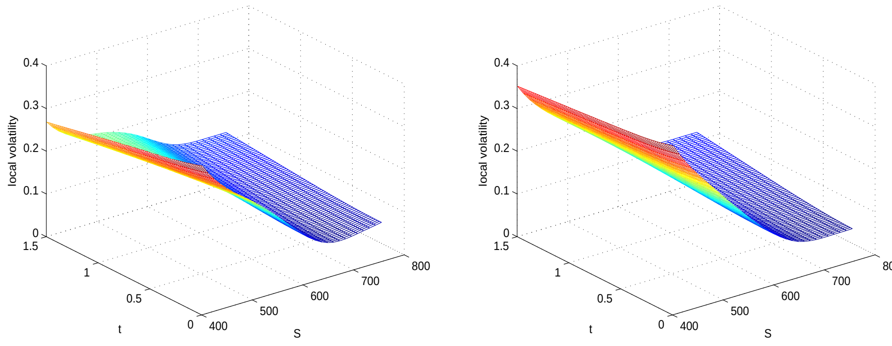


Left plot: collected market data without noise   Right plot: perturbed market data

Figure 4.8: 3-D plot of calibrated LVF from S&P 500 Index Options in Oct 95 with uniform weights

Maturity \ Strike	85%	90%	95%	100%	105%	110%	115%	120%
0.695	-.10	0.19	.66	.06	3.55	11.4	-2.37	-51.79
1	-.10	-.06	-.72	-1.08	-2.53	-.76	-2.35	-19.37
1.5	.17	.11	.07	-.10	.19	1.37	-.26	.12

Table 4.5: Relative calibration error in % for S&P 500 Index Options in Oct 95 of maturities (year) and strikes (% of underlying price) with uniform weights



Left plot: collected market data   Right plot: perturbed market data

Figure 4.9: 3-D plot of calibrated LVF from S&P 500 Index Options in Oct 95 with weights (4.6)

Maturity \ Strike	85%	90%	95%	100%	105%	110%	115%	120%
0.695	-.09	0.20	.61	-.23	2.78	11.95	11.11	-13.39
1	-.11	-.02	-.64	-1.07	-2.79	-.71	2.94	2.74
1.5	.09	.15	.25	.1	.04	.45	-1.29	1.5

Table 4.6: Relative calibration error in % for S&P 500 Index Options in Oct 95 of maturities (year) and strikes (% of underlying price) with weights (4.6)

(For a smooth LVF calibration, we have to allow large errors for some low-price options. The largest error 11.95%, 11.11%, and -13.39 are for options priced at 6.02, 1.93 and .62.)

choose 18 training points on grid $[0.8S_0 : 0.05S_0 : 1.2S_0] \times [0.5, 1]$. Centering and scaling parameters are set as: $C_K = S_0$, $Q_K = 50$, $C_T = 0.75$ and $Q_T = 1$. These choices make the strikes and maturities centered around zero and with a deviation from center around one. The regularization parameter $\rho = 10$ is used. Note that the calibration error is expected to be much larger than the error in the previous test examples because the option prices are almost 6 time larger. Therefore we choose a larger $\rho = 10$ to ensure more appropriate balance of the values in the objective function. The optimization takes 51 iterations and the calibration error is 2.4. The reconstructed local volatility surface is shown as the left 3-D mesh plot in Figure 4.8. It can be observed that the local volatility surface is smooth within the significant region $[.7S_0, 1.3S_0] \times [0, 1.5]$. Moreover, the local volatility surface resembles the implied volatility surface shown in Figure 4.7. The relative errors of calibrated option prices $\frac{\bar{V}^0(\bar{K}_i,\bar{T}_i)-V^0(\bar{K}_i,\bar{T}_i)}{\bar{V}^0(\bar{K}_i,\bar{T}_i)}$ in % are listed in Table 4.5. For most options, the errors are within $\pm1\%$. However for out-of-the-money call options with short maturities, the relative errors are larger because their prices are quite smaller; thus their errors have relatively smaller weights in the sum of the squared errors. The calibration errors for out-of-the-money (OTM) options can be decreased if we allocate larger weights w_j for them. For example, we set the weights as below:

$$w_j = \begin{cases} 45 & j = 8 \\ 4 & j = 7, 16 \\ 2 & j = 15 \\ 1 & \text{otherwise.} \end{cases} \tag{4.6}$$

We order the options $\{V_j^0\}_{j=1}^{24}$ by strikes and maturities in this way: $j = 1, ...8$ for strikes from low to high at maturity 0.695; similarly $j = 2, ...16$ and $j = 17, ...24$ for maturity at 1 and 1.5. Table 4.6 shows the calibration relative errors for each option and we can see that errors for OTM options are lower than the uniform weight case.

02 March 2004

Maturity \ Strike	1025	1050	1100	1125	**1150**	1200	1250	1300
.58	.197	.1872	.1645	.1588	.1538	.1398	.1323	.1257
.84	.194	.1801	.1709	.1595	.1576	.1448	.1344	.1324
1.34	.1976	.1908	.1782	.1725	.1649	.1577	.1503	.1402

05 April 2004

Maturity \ Strike	1025	1050	1100	1125	**1150**	1200	1250	1300
.5	.1952	.1852	.1597	.1582	.1471	.1305	.1228	.1164
1	.201	.1936	.1797	.1689	.1628	.152	.1473	.1394
1.25	.2097	.196	.1894	.1785	.1776	.1673	.1584	.1511

Table 4.7: Implied volatilities for S&P 500 Index Options on 02 Mar 2004 and 05 Apr 2004

In order to illustrate the robustness of the calibrated local volatility function, we add uniformly distributed noise to the market index option prices. Similar to the previous synthetic data example, we add larger perturbation for some OTM options. In particular, the option price is multiplied by:

$$\begin{cases} (1 + 0.06 * (\text{rand} - 0.5)), & \text{if } (\bar{K}, \bar{T}) = (1.15S_0, .695), (1.2S_0, .695), (1.2S_0, 1); \\ (1 + 0.02 * (\text{rand} - 0.5)), & \text{otherwise.} \end{cases}$$

We use the same experiment parameters as the noise-free option example. The optimization algorithm takes 64 iterations and finally gives the calibration error at 4.53. The calibrated local volatility surface is illustrated in the right plot of Figure 4.8, which is close to the previous one calibrated from noise-free option data. Using unequal weight $\{w_j\}_{j=1}^{24}$ in (4.6), Figure 4.9 shows the 3-D mesh plot of calibrated local volatility surfaces for options data with and without noise. We see that adding larger weights on some out-of-the-money options still gives a smooth volatility surface and the shape of the surface does not change much. As a whole, this example reveals that our LVF model calibration with kernel spline method is relatively robust against the option data noise.

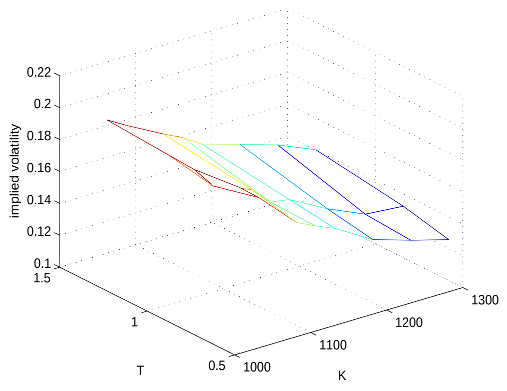
02 March 2004

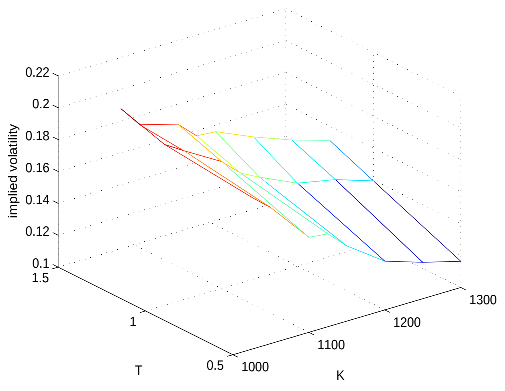
Maturity \ Strike	1025	1050	1100	1125	**1150**	1200	1250	1300
.58	141	120.4	81	65	50.9	26.9	12.7	5
.84	148.7	127.1	93	75	62.2	37.4	20	10.9
1.34	164.2	145.8	111.8	96.4	81	57.6	38.6	23

05 April 2004

Maturity \ Strike	1025	1050	1100	1125	**1150**	1200	1250	1300
.5	138.9	117.9	77.2	62.2	46	21.7	8.8	2.8
1	157.1	138.1	103	85.2	70.6	45.9	29.2	16.3
1.25	167.7	146.4	115.2	97.3	85.3	60.2	40.3	25.6

Table 4.8: Prices for S&P 500 Index Options on 02 Mar 2004 and 05 Apr 2004





Left plot: options on 02 Mar 2004 Right plot: options on 05 Apr 2004

Figure 4.10: Implied volatility surface of S&P 500 Index Options in 2004

02 March 2004

Maturity \ Strike	1025	1050	1100	1125	**1150**	1200	1250	1300
.58	-.9	-.88	.92	.12	-1.34	.48	-2.35	-5.51
.84	-.16	1.41	-.38	1.86	-.86	.57	3.7	-6.59
1.34	.02	.0	-2.45	-.46	.76	-.92	-1.68	3.57

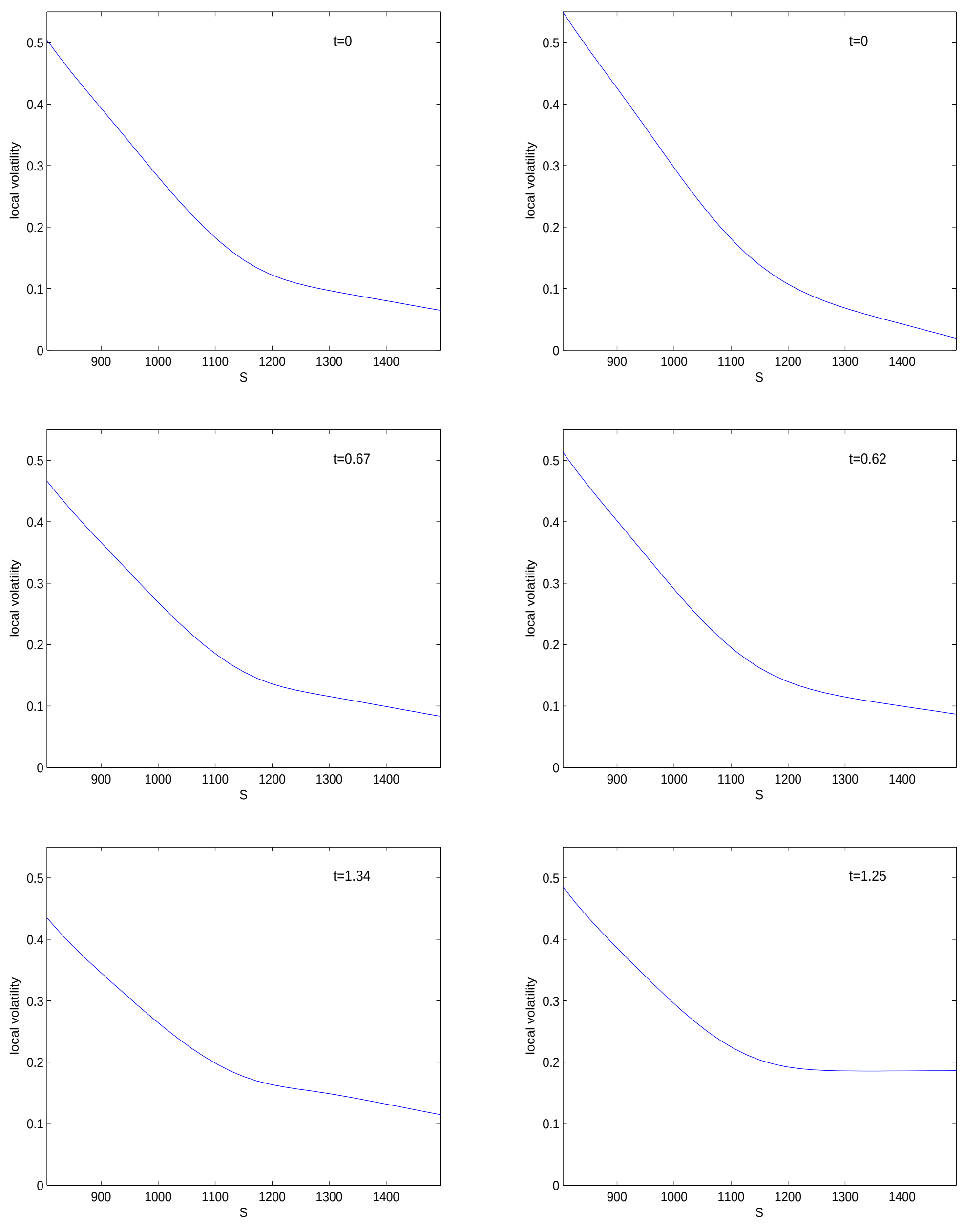
05 April 2004

Maturity \ Strike	1025	1050	1100	1125	**1150**	1200	1250	1300
.5	-.42	-.48	1.66	-1.57	-.29	2.92	-3.55	-16.19
1	.09	-.06	-.68	1.13	1.13	1.7	-2.79	-2.09
1.25	-.54	1.29	-1.11	1.07	-1.52	-1.16	.37	3.06

Table 4.9: Relative calibration error in % for S&P 500 Index Options on 02 Mar 2004 and 05 Apr 2004 with uniform weights

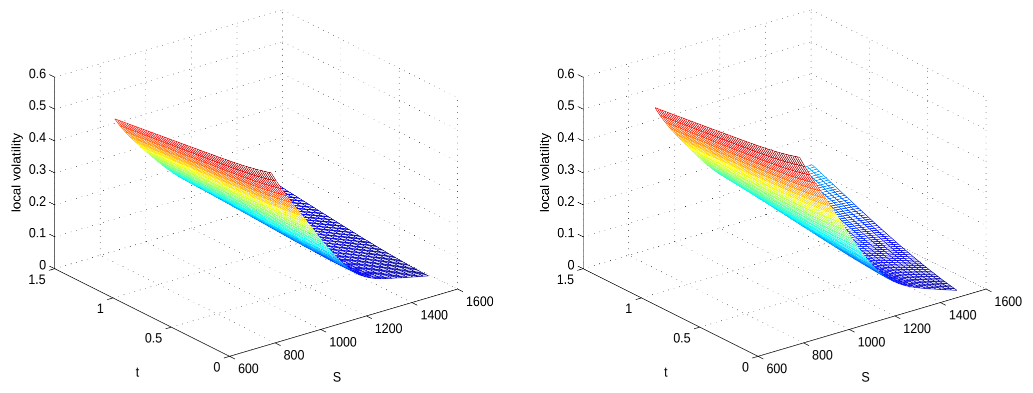
4.3.2 Stability of Proposed LVF Calibration Method

A stable calibration model should yield similar local volatility surfaces for option data on nearby dates. Therefore two sets of S&P 500 Index Options data are collected. They are chosen on two close dates, 02 March 2004 and 05 April 2004, which are the closest dates collected from $http://www.marketdataexpress.com/dataEODSumOverview.aspx?u = SPX$. The underlying price is 1149.1 on 02 March 2004 and 1150.57 on 05 Apr 2004. The interest rate $r = 0.01$ and dividend rate $q = 0.016$ are the same for two days. Table 4.7 and 4.8 list the implied volatilities and prices for the collected options. Similarly 18 training points on grid $[0.8S_0 : 0.05S_0 : 1.2S_0] \times [0.5, 1]$ are chosen and the scaling parameters are: $C_K = S_0$, $Q_K = 120$, $C_T = 0.75$ and $Q_T = 1$. We use the regularization parameter $\rho = 10$. Finally we list the relative calibrated error in % in Table 4.9. Two dimensional plots and three dimensional mesh plots for both examples are illustrated in Figure 4.11 and 4.12. These plots demonstrate that both calibrated local volatility surfaces are smooth in region of $[0.7S_0, 1.3S_0]$. Furthermore the shapes of surfaces look quite similar and resemble their implied volatility surfaces shown in Figure 4.10. They give higher local volatility for lower S&P 500 index. As time goes by the local volatility is higher for high index values.



Left plots: options on 02 Mar 2004 Right plots: options on 05 Apr 2004

Figure 4.11: LVF calibration from S&P 500 Index Options in 2004 with uniform weights



Left plot: options on 02 Mar 2004 Right plot: options on 05 Apr 2004

Figure 4.12: 3-D plot of calibrated LVF from S&P Index Options in 2004 with uniform weights

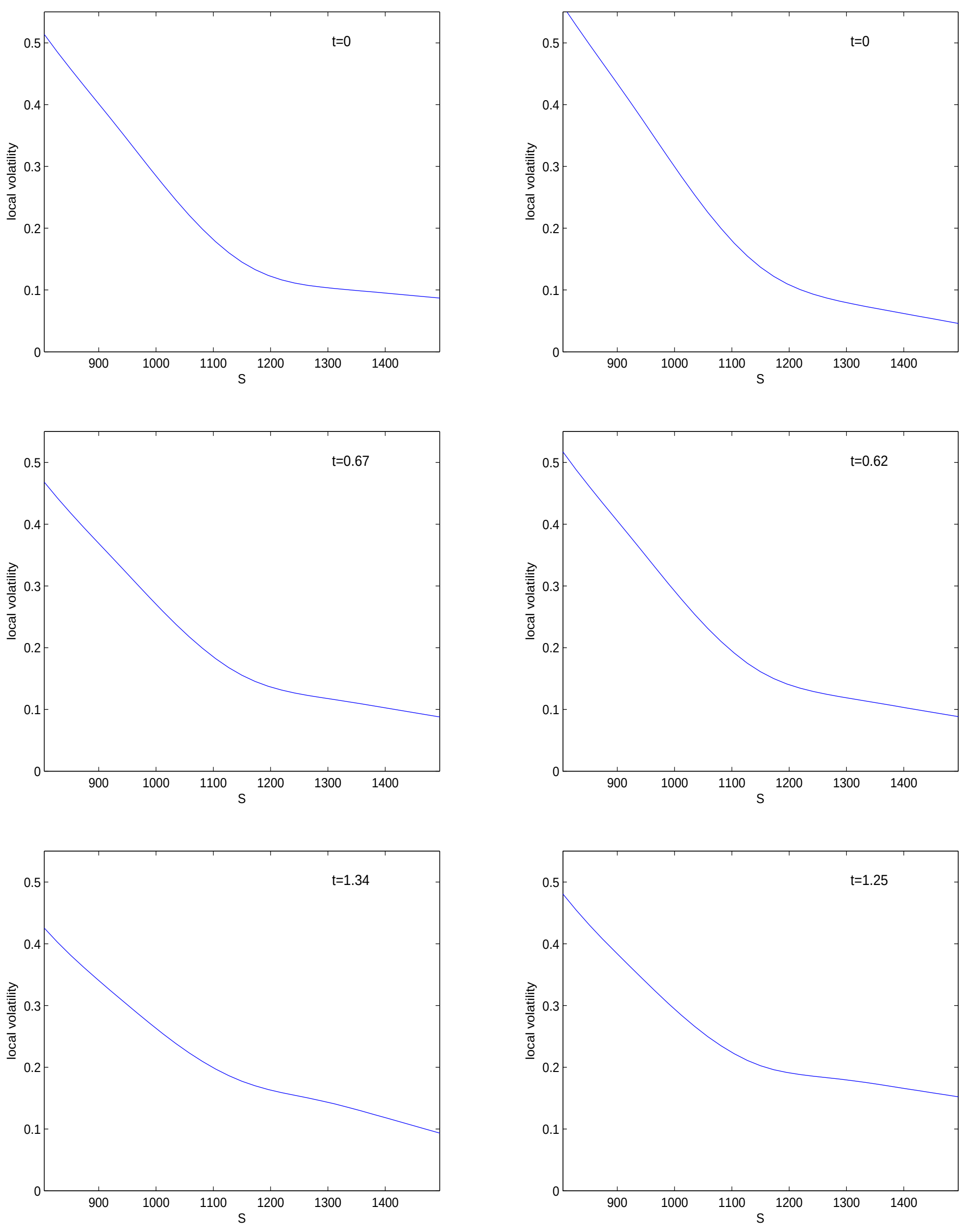
02 March 2004

Maturity \ Strike	1025	1050	1100	1125	**1150**	1200	1250	1300
.58	-.92	-.93	.78	-.05	-1.51	.63	-.87	-.66
.84	-.14	1.42	-.38	1.86	-.82	.84	4.64	-4.45
1.34	.02	.02	-.19	-.38	.83	-.99	-2.2	1.88

05 April 2004

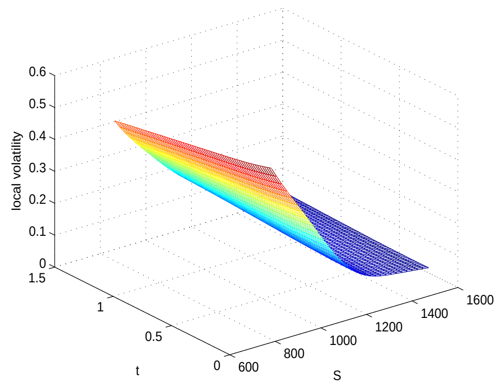
Maturity \ Strike	1025	1050	1100	1125	**1150**	1200	1250	1300
.5	-.45	-.54	1.5	-1.76	-.44	3.72	.95	-3.56
1	.12	-.02	-.62	1.21	1.27	2.13	-1.92	-.84
1.25	-.54	1.29	-1.11	1.06	-1.56	-1.31	-.2	1.24

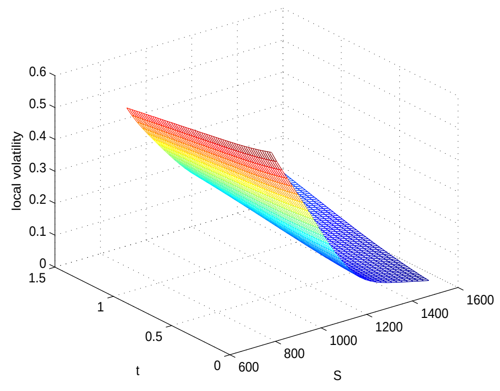
Table 4.10: Relative calibration error in % for S&P 500 Index Options on 02 Mar 2004 and 05 Apr 2004 with weights (4.7)



Left plots: options on 02 Mar 2004 Right plots: options on 05 Apr 2004

Figure 4.13: LVF calibration from S&P 500 Index Options in 2004 with weights (4.7)





Left plot: options on 02 Mar 2004 Right plot: options on 05 Apr 2004

Figure 4.14: 3-D plot of calibrated LVF from S&P 500 Index Options in 2004 with weights (4.7)

To enhance the model accuracy for those out-of-the-money options, we carry out the same experiment as above with individual weights assigned as

$$w_j = \begin{cases} 9 & j = 8 \\ 4 & j = 16, 24 \\ 2 & j = 7, 15, 23 \\ 1 & \text{otherwise.} \end{cases} \tag{4.7}$$

Similarly We order the options $\{V_j^0\}_{j=1}^{24}$ by strikes and maturities from low to high. Particularly $j = 1, ...8$, $j = 2, ...16$ and $j = 17, ...24$ are for options with maturities at 0.58, 0.84 and 1.34 on 02 March 2004 and 0.5, 1 and 1.25 on 05 April 2004. Table 4.10 gives the relative calibration error and Figure 4.13 and 4.14 give the 2-D and 3-D local volatility plots. We can see that calibrated errors for some OTM options are improved. In addition, the calibrated local volatility surfaces still look similar for two nearby dates, 02 Mar 2004 and 05 Apr 2004. Moreover the shapes in the region of high spot prices are even slightly smoother and closer between two dates than the uniform weights case.

Chapter 5

Concluding Remarks

We propose a stable local volatility surface calibration method based on SVR for function estimation with kernel generating splines. Determining a local volatility function from a finite number of market prices is an ill-posed inverse problem. One of the key problems in the LVF calibration is to control the balance between calibration accuracy and stability of the calibration. SVR has good generalization performance since the ϵ level of regression adjusts the number of SVs, and therefore controls the balance between these two factors. Unfortunately we only have a set of market option prices available which is the indirect measurement of the local volatility function. The classical SVR quadratic optimization problem cannot be directly used. Instead, by examining the good generalization properties of SVR we motivate our calibration formulation (2.8):

$$\min_{x \in \Re^{l+1}} \sum_{j=1}^{m} w_j (V^0((\bar{K}_j, \bar{T}_j); \sigma((K,T);x)) - \bar{V}_j^0)^2 + \rho \sum_{i=0}^{l} |x^{(i)}|$$

in which LVF $\sigma((K,T);x)$ is represented by a linear expansion of kernel generating splines:

$$\sigma((K,T);x) = \left| \sum_{i=1}^{l} x^{(i)} F((K,T),(K_i,T_i)) + x^{(0)} \right|,$$

with

$$\begin{aligned} F((K,T),(K_i,T_i)) &= [1 + KK_i + \frac{1}{2} \mid K - K_i \mid (K \wedge K_i + K_b)^2 + \frac{(K \wedge K_i + K_b)^3}{3}] \times \\ & [1 + TT_i + \frac{1}{2} \mid T - T_i \mid (T \wedge T_i + T_b)^2 + \frac{(T \wedge T_i + T_b)^3}{3}]. \end{aligned}$$

The proposed calibration approach has the following features:

- The local volatility function representation follows the format of estimation function in the SVR framework. The only unknowns we need to determine is the coefficient vector x. In this representation, a kernel spline constructed by an infinite number of knots gives the kernel function of input variables (K,T). In SVR, kernel methods allow us to analyze nonlinear relations in a high dimensional feature space while preserve the efficiency of linear algorithms.

- The proposed approach minimizes the calibration error, which has the similar effect of minimizing the loss function of regression estimation in SVR approach.

- We emphasize minimizing one-norm of the coefficients $\|x\|_1$ in the proposed formulation. It forces some coefficients $x^{(i)}$ to be zero at the termination of optimization. Thus the local volatility function is formed by an expansion of kernel splines associated with nonzero coefficients of x. This corresponds to minimizing the number of SVs in the SVR framework.

- The regularization parameter ρ controls the trade-off between the calibration accuracy and model stability. Large ρ allows large calibration error, small number of "SVs", and therefore obtains a simple and stable calibration model. This achieves the corresponding property in SVR solution: a large ϵ level in loss function allows large regression error, leading to small number of SVs and a stable estimation function.

The proposed calibration formulation (2.8) becomes a problem of minimizing a smooth nonlinear function plus one-norm of the variables. In this thesis we also propose a trust region method to determine the coefficient vector x in the proposed optimization formulation. In this algorithm, the main computation of each iteration is reduced to solving a standard trust region subproblem. A line search technique which allows crossing nondifferentiable $x^{(i)} = 0$ hyperplanes is introduced to find the minimum objective value along a direction within a trust region. To ensure global convergence and local quadratic convergence, a minimizer for each iteration is chosen from the minimum of objective values along trust region subproblem solution and steepest descent direction. We compare two methods TR1 and TR2 which are slightly different in that TR1 allows crossing only one $x^{(i)} = 0$ hyperplane and TR2 allows crossing as many hyperplanes as possible in the line search. TR2 is expected to converge faster than TR1 since it attempts to achieve larger objective value reduction at each iteration.

Two local volatility calibration examples are implemented in this thesis. The first example uses synthetical call options computed with a known LVF model. The second example experiments with real S&P 500 Index European options. For better optimization performance, we try to obtain a priori of vector x. Issues such as data centering and placement of training points in price-time domain are also illustrated. The first example demonstrates the accuracy of reconstructed LVF and robustness with noise added to the market options. Parameter ρ tunes the balance between these two factors. Larger ρ allows larger reconstruction errors and leads to a more robust model, i.e., the reconstructed LVF is less affected by the noise added to the data. One-dimensional LVF reconstruction is computed by assuming that the LVF only depends on underlying asset price. In this case the computation is easier than two-dimensional reconstruction which assumes that LVF depends on both the underlying price and time. In the synthetic data example we experiment on the number of chosen training points in price-time domain. Results show that the LVF reconstruction is relatively insensitive against the number and placement of the training points.

The second example calibrates a LVF first based on market S&P 500 Index Options in October 1995, and then on options on two close dates, March 02, 2004 and April 05, 2004. With appropriate choices of parameters, a calibrated LVF is smooth and resembles the shape of the implied volatility surface. Moreover calibration is robust against the uniform distributed noise added to October 1995 option data. Calibration from options on two close dates reveals that the proposed approach is stable for slightly different option data samples. However if we use uniform weights for all option data in the calibration formulation, the calibration errors for out-of-the-money (OTM) options are larger than those of other options. This is due to the fact that the OTM options have lower prices and are less important in the calibration formulation. By using larger weights for OTM options, we obtain better calibration accuracy for OTM options. With the specified individual weights, the relative calibration error is within $\pm 1\%$ for most options and within $\pm 5\%$ for other (OTM) options. Moreover the calibrated LVF for two nearby dates looks closer with individual weights, especially in large index region.

Appendix A

SVM Optimization Formulation for Separable Classification

Suppose we want to find a linear function $h : \Re^N \rightarrow \{\pm 1\}$ to separate training data

$$(s_1, y_1), ..., (s_l, y_l) \in \Re^N \times \{\pm 1\}.$$

The following optimization problem determines the optimal hyperplane:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|z\|_2^2, \quad z \in \Re^N \\ \text{subject to} & y_i * ((z \cdot s_i) + b) \geq 1, \quad i = 1, ..., l, \quad b \in \Re, \end{array}$$

with decision function

$$h(s) = \text{sign}(z \cdot s + b).$$

To solve the constrained optimization problem, one can introduce a Lagrangian

$$L(z, b, \beta) = \frac{1}{2}\|z\|^2 - \sum_{i=1}^{l} \beta_i [y_i * ((s_i \cdot z) + b) - 1] \tag{A.1}$$

with Lagrange multipliers $\beta_i \geq 0$. The Lagrangian L has to be minimized with respect to the primal variables z and b and maximized with respect to the dual variables β_i. Therefore at the saddle point, one has

$$\frac{\partial}{\partial b}L(z,b,\beta) = 0, \;\; \frac{\partial}{\partial z}L(z,b,\beta) = 0,$$

which leads to

$$\sum_{i=1}^{l} \beta_i y_i = 0 \;\text{ and }\; z = \sum_{i=1}^{l} \beta_i y_i s_i. \tag{A.2}$$

By substituting (A.2) into (A.1), one obtains the dual optimization problem: find multipliers β_i which solves

$$\begin{array}{ll} \text{maximize} & \displaystyle\sum_{i=1}^{l} \beta_i - \frac{1}{2}\sum_{i,j=1}^{l} \beta_i \beta_j y_i y_j (s_i \cdot s_j) \\ \text{subject to} & \displaystyle\sum_{i=1}^{l} \beta_i y_i = 0, \text{ and } \beta_i \geq 0, \; i = 1, ..., l. \end{array}$$

Since the Lagrangian L has to be maximized with respect to the dual variables β_i, for all constraints which are not met as equalities, i.e., $y_i * (z \cdot s_i + b) - 1 > 0$, the corresponding $\beta_i = 0$. This gives the KKT complementarity conditions

$$\beta_i * [y_i(z \cdot s_i + b) - 1] = 0, \;\; i = 1, ..., l.$$

The hyperplane decision function can be written as

$$h(s) = \text{sign}(\sum_{i=1}^{l} y_i \beta_i (s \cdot s_i) + b)$$

where b is computed using KKT complementarity condition with a $\beta_j \neq 0$:

$$\beta_j * [y_j((s_j \cdot z) + b) - 1] = 0$$

with

$$z = \sum_{i=1}^{l} \beta_i y_i s_i.$$

Appendix B

SVM Optimization Formulation for Nonseparable Classification

Suppose we have training data

$$(s_1, y_1), ..., (s_l, y_l) \in \Re^N \times \{\pm 1\}$$

and there is no hyperplane which can separate these two classes. We want to find an optimal hyperplane that allows some overlaps of class data. By introducing slack variables ξ_i, $1 \leq i \leq l$, the optimal hyperplane solves:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\|z\|_2^2 + C\sum_{i=1}^{l}\xi_i, \quad z \in \Re^N \\ \text{subject to} \quad & y_i * ((z \cdot s_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \quad i = 1, ..., l, \quad b \in \Re, \end{aligned}$$

where $C > 0$ is a constant. The decision function is

$$h(s) = \mathrm{sign}(z \cdot s + b).$$

To solve the constrained optimization problem, one can introduce a Lagrangian

$$L(z, b, \xi, \beta, \beta^*) = \frac{1}{2}\|z\|^2 + C\sum_{i=1}^{l} \xi_i - \sum_{i=1}^{l} \beta_i(y_i * (s_i \cdot z + b) - 1 + \xi_i) - \sum_{i=1}^{l} \beta_i^* \xi_i \qquad \text{(B.1)}$$

with Lagrange multipliers $\beta_i \geq 0$ and $\beta_i^* \geq 0$. The Lagrangian L has to be minimized with respect to the primal variables z, b and ξ and maximized with respect to the dual variables β_i and β_i^*. Therefore at the saddle point, the derivative to the primal variables leads to

$$\sum_{i=1}^{l} \beta_i y_i = 0, \;\; z = \sum_{i=1}^{l} \beta_i y_i s_i, \;\text{ and }\; C - \beta_i - \beta_i^* = 0, \;\; 1 \leq i \leq l. \qquad \text{(B.2)}$$

By substituting (B.2) into (B.1), one obtains the dual optimization problem: find multipliers β_i which solves

$$\begin{array}{ll} \text{maximize} & \displaystyle\sum_{i=1}^{l} \beta_i - \frac{1}{2} \sum_{i,j=1}^{l} \beta_i \beta_j y_i y_j (s_i \cdot s_j) \\ \text{subject to} & \displaystyle\sum_{i=1}^{l} \beta_i y_i = 0, \text{ and } C \geq \beta_i \geq 0, \; i = 1, ..., l. \end{array}$$

Since the Lagrangian L has to be maximized with respect to the dual variables β_i and β_i^*, for the following constraints which are not met as equalities, i.e.,

$$y_i * (z \cdot s_i + b) - 1 + \xi_i > 0,$$

one has the corresponding $\beta_i = 0$. This gives the KKT complementarity conditions

$$\beta_i * [y_i(z \cdot s_i + b) - 1 + \xi_i] = 0, \quad i = 1, ..., l.$$

For the other constraints which are not met as equalities, i.e., $\xi_i > 0$, one has the corresponding $\beta_i^* = 0$. Thus we can see that β_i is bounded by C and equals to C when $\xi_i > 0$.

The hyperplane decision function can be written as

$$h(s) = \text{sign}(\sum_{i=1}^{l} y_i \beta_i * (s \cdot s_i) + b)$$

where b is computed using KKT complementarity conditions

$$\beta_j * [y_j(\sum_{i=1}^{l} y_i \beta_i * (s_j \cdot s_i) + b) - 1] = 0, \quad \text{for} \quad \beta_j < C, \quad j = 1, ..., l.$$

Appendix C

SVM Optimization Formulation for Function Estimation Regression

Suppose we have training data

$$(s_1, y_1), ..., (s_l, y_l) \in \Re^N \times \Re.$$

We want to estimate a linear regression

$$h(s) = z \cdot s + b, \;\; z \in \Re^N, \;\; b \in \Re.$$

By using Vapnik's $\epsilon-$insensitive loss function

$$|y - h(s)|_\epsilon \overset{\text{def}}{=} \max\{0, |y - h(s)| - \epsilon\},$$

one can minimize

$$\frac{1}{2}\|z\|_2^2 + C\sum_{i=1}^{l} |y_i - h(s_i)|_\epsilon$$

with accuracy $\epsilon > 0$ and priori constant $C > 0$.

By introducing slack variables $\xi \geq 0$ and $\bar{\xi} \geq 0$ for $1 \leq i \leq l$, one rewrites this as a constrained optimization problem:

$$\begin{aligned}
\text{minimize} \quad & \frac{1}{2}\|z\|_2^2 + C\sum_{i=1}^{l}(\xi_i + \bar{\xi}_i) \\
\text{subject to} \quad & ((z \cdot s_i) + b) - y_i \leq \epsilon + \xi_i \\
& y_i - ((z \cdot s_i) + b) \leq \epsilon + \bar{\xi}_i \\
& \xi_i, \bar{\xi}_i \geq 0, \quad \text{for all } \ i = 1, ..., l.
\end{aligned}$$

To solve the constrained optimization problem, one can introduce a Lagrangian

$$\begin{aligned}
L(z, b, \xi, \bar{\xi}, \beta, \bar{\beta}, \beta^*, \bar{\beta}^*) \quad = \quad & \frac{1}{2}\|z\|^2 + C\sum_{i=1}^{l}(\xi_i + \bar{\xi}_i) \\
& - \sum_{i=1}^{l}\beta_i(y_i - (s_i \cdot z + b) + \epsilon + \xi_i) - \sum_{i=1}^{l}\beta_i^*\xi_i \\
& - \sum_{i=1}^{l}\bar{\beta}_i(-y_i + (s_i \cdot z + b) + \epsilon + \bar{\xi}_i) - \sum_{i=1}^{l}\bar{\beta}_i^*\bar{\xi}_i
\end{aligned} \tag{C.1}$$

with Lagrange multipliers $\beta_i \geq 0$, $\bar{\beta}_i \geq 0$, $\beta_i^* \geq 0$ and $\bar{\beta}_i^* \geq 0$. The Lagrangian L has to be minimized with respect to the primal variables z, b, ξ and $\bar{\xi}$ and maximized with respect to the dual variables β_i, $\bar{\beta}_i$, β_i^* and $\bar{\beta}_i^*$. Therefore at the saddle point, derivative to primal variables leads to

$$\begin{aligned}
& \sum_{i=1}^{l}\beta_i - \bar{\beta}_i = 0 \\
& z = \sum_{i=1}^{l}\beta_i s_i - \bar{\beta}_i s_i \\
& C - \beta_i - \beta_i^* = 0, \ \ 1 \leq i \leq l \\
& C - \bar{\beta}_i - \bar{\beta}_i^* = 0, \ \ 1 \leq i \leq l.
\end{aligned} \tag{C.2}$$

By substituting (C.2) into (C.1), one obtains the dual optimization problem: find multipliers β_i, $\bar{\beta}_i$ which solves

$$\begin{aligned} \text{maximize} \quad & -\epsilon\sum_{i=1}^{l}(\bar{\beta}_i+\beta_i)+\sum_{i=1}^{l}(\bar{\beta}_i-\beta_i)y_i \\ & -\frac{1}{2}\sum_{i,j=1}^{l}(\bar{\beta}_i-\beta_i)(\bar{\beta}_j-\beta_j)(s_i\cdot s_j) \\ \text{subject to} \quad & \sum_{i=1}^{l}(\beta_i-\bar{\beta}_i)=0, \text{ and } C\geq\beta_i,\bar{\beta}_i\geq 0,\ i=1,...,l. \end{aligned}$$

Since the Lagrangian L has to be maximized with respect to the dual variables β_i, $\bar{\beta}_i$, β_i^* and $\bar{\beta}_i^*$, we can find the KKT complementarity conditions

$$\begin{aligned} &\beta_i * [y_i-(s_i\cdot z+b)+\epsilon+\xi_i]=0,\ \ 1\leq i\leq l \\ &\bar{\beta}_i * [-y_i+(s_i\cdot z+b)+\epsilon+\bar{\xi}_i]=0,\ \ 1\leq i\leq l \end{aligned}$$

and

$$\beta_i^*\xi_i=0 \text{ and } \bar{\beta}_i^*\bar{\xi}_i=0,\ \ 1\leq i\leq l.$$

Thus we can see that β_i ($\bar{\beta}_i$) is bounded by C and equals to C when $\xi_i>0$ ($\bar{\xi}_i>0$).

The hyperplane decision function can be written as

$$h(s)=\text{sign}(\sum_{i=1}^{l}y_i\beta_i*(s\cdot s_i)+b)$$

where b is computed using KKT complementarity conditions with a point with $\beta_i<C$ ($\xi_i=0$) or $\bar{\beta}_i<C$ ($\bar{\xi}_i=0$).

Appendix D

First-order Optimality Conditions for Nonlinear L_1 Optimization

Suppose an optimization is defined below:

$$\min_{x \in \Re^n} f(x) + \|c(x)\|_1 \tag{D.1}$$

where $f : \Re^n \rightarrow \Re^1$ is a twice continuously differentiable function, $c : \Re^n \rightarrow \Re^m$ is a continuously differentiable function, and $\|c(x)\|_1 = \sum_{i=1}^m |c^{(i)}(x)|$. We let the sign function be defined:

$$\mathrm{sign}(c^{(i)}) \overset{\text{def}}{=} \begin{cases} 1 & \text{if } c^{(i)} \geq 0, \\ -1 & \text{if } c^{(i)} < 0. \end{cases}$$

And denote $J \overset{\text{def}}{=} J(x)$ with $J(x) \overset{\text{def}}{=} [\nabla c^{(1)}(x), ..., \nabla c^{(m)}(x)] \in \Re^{n \times m}$.

The following first order necessary conditions can be obtained from Theorem 14.2.1, 14.2.2, 14.2.3 and Lemma 14.3.1 in [17]. If x is a local minimizer of a nonlinear L_1 problem (D.1), then there exists $y \in \Re^m$ with $|y| \leq 1$ and $y^{(i)} = \mathrm{sign}(c^{(i)})$ if $c^{(i)} \neq 0$, such that $\nabla f + Jy = 0$.

If $c(x) = x$, the first-order condition can be written as following. If x is a local minimizer of a nonlinear L_1 problem

$$\min_{x \in \Re^n} f(x) + \|x\|_1, \tag{D.2}$$

then there exists $y \in \Re^n$ with $|y| \leq 1$ and $y^{(i)} = \text{sign}(x^{(i)})$ if $x^{(i)} \neq 0$, such that $\nabla f + y = 0$. Therefore we obtain the first-order KKT conditions for (D.2): if a point x is a local minimizer, then

$$x^{(i)}[(\nabla f(x))^{(i)} + \text{sign}(x^{(i)})] = 0 \text{ and } \mid (\nabla f(x))^{(i)} \mid \leq 1 \text{ for } 1 \leq i \leq n.$$

Bibliography

[1] L. B. G Andersen and R. Brotherton-Ratcliffe. The equity option volatility smile: An implicit finite difference approach. *Journal of Computational Finance*, 1(2):5–37, 1998.

[2] M. Avellaneda, C. Friedman, R. Holmes, and D. Samperi. Calibrating volatility surfaces via relative entropy minimization. *Applied Mathematical Finance*, 4, 1997.

[3] K. P. Bennett and M. J. Embrechts. An opitmization perspective on kernel partial least squares regression. In *Advances in Learning Theory: Methods, Models and Applications.* IOS Press Amsterdam.

[4] F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81(1):637–654, 1973.

[5] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory.* ACM Press, 1992.

[6] I. Bouchouev and V. Isakov. Uniqueness, stability, and numerical methods for the inverse problem arises in financial markets. *Inverse Problems*, 15:R95–R116, 1999.

[7] T. F. Coleman and Y. Li. On the convergence of reflective newton methods for large-scale nonlinear minimization on bounds. *Mathematical Programming*, 67, 1994.

[8] T. F. Coleman and Y. Li. An interior, trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization*, 6(2):418–445, 1996.

[9] T. F. Coleman and Y. Li. A reflective newton method for minimizing a quadratic function subject to bounds on the variables. *SIAM Journal on Optimization*, 6(4): 1040–1058, 1996.

[10] T. F. Coleman, Y. Li, and A. Verma. Reconstructing the unknown local volatility function. *Journal of Computational Finance*, 2(3):77–102, 1999.

[11] T. F. Coleman and A. Verma. Structure and efficient jacobian calculation. In Martin Berz, Christian Bischof, George Corliss, and Andreas Griewank, editors, *Computational Differentiation: Techniques, Applications, and Tools*, pages 149–159. SIAM, Philadelphia, Penn., 1996. URL `citeseer.ist.psu.edu/coleman96structure.html`.

[12] C. Cortes and V. Vapnik. Support vector network. *Machine Learning*, 20, 1995.

[13] E. Derman and I. Kani. Riding on a smile. *Risk*, 7, 1994.

[14] Paul Dierckx. *Curve and Surface Fitting with Splines.* Oxford Science Publications, 1993.

[15] B. Dupire. Pricing with a smile. *Risk Magazine*, 7(1):18–20, 1994.

[16] R. Fletcher. *Practical Methods of Optimization: Unconstrained Optimization.* John Wiley and Sons, 1981.

[17] R. Fletcher. *Practical Methods of Optimization: Volume 2, Constrained Optimization.* John Wiley and Sons, 1981.

[18] J. C. Hull and A. White. The pricing of options on assets with stochastic volatilities. *Journal of Finance*, 42:281–300, June 1987.

[19] N. Jackson, E. Süli, and S. Howison. Computation of deterministic volatility surface. *Journal of Mathematical Finance*, 1998.

[20] J. C. Jackwerth and M. Rubinstein. Recovering probability distributions from option prices. *Journal of Finance*, 51, 1996.

[21] R. Lagnado and S. Osher. A technique for calibrating derivative security pricing models: numerical solution of an inverse problem. *Journal of Computational Finance*, 1(1):13–25, 1997.

[22] M. Meila. Data centering in feature space. Technical Report 421, University of Washington, 2002. URL `citeseer.ist.psu.edu/meila02data.html`.

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

[24] J. Shawe-Taylor and N. Cristianini. *Kernel methods for pattern analysis.* Cambridge University Press, 2004.

[25] V. Vapnik. *The nature of Statistical Learning Theory.* Springer Verlag, 1982.

[26] V. Vapnik and A. Chervonenkis. A note on one class of perceptrons. *Automation and Remote Control*, 25, 1964.

[27] V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24, 1963.

[28] Vladimir N. Vapnik. *Statistical Learning Theory.* AT&T Labs-Research, London University, 1998.

[29] G. Wahba. Spline models for observational data. In *Series in Applied Mathematics*, 1990.