THE FLORIDA STATE UNIVERSITY
COLLEGE OF ARTS AND SCIENCES

Calibration of a Local Volatility Surface using Tikhonov Regularization

Jian Geng

Directed by Dr. I. Michael Navon

A Candidacy Exam submitted to the
Department of Mathematics
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Contents

Chapter 1

Introduction

Based on the assumption of a constant volatility, the celebrated Black-Scholes model can be used to evaluate European style options easily and quickly by using an estimated volatility or forecast volatility as an input[5]. If this model is perfectly realistic, then the implied volatility for all options under the same underlying with different strike and maturities would be the same. However, in the market, it is usually observed that Black-Scholes implied volatility of the same underlying varies with strikes and maturity, which are respectively known as the smile effect or sometimes skew effect and the term structure[11].

There have been various attempts to extend the Black-Scholes theory to account for the volatility effect and the term structure. One class of models introduces a non-traded source of risk such as jumps[24] and stochastic volatility[27]. Another class considers the volatility term as a deterministic volatility function that depends on the spot price and time, which is usually called the local volatility function. For this assumption, Black-Scholes model remains a one parameter model and thus retains the completeness of the model:the ability to hedge options with the underlying asset, is maintained.

In this candiate exam dissertation, we will just focus on calibration of the local volatility functions. There may be disagreement concerning whether the local volatility function is the best parameterization for specifying the underlying process. We'll temporarily bypass this subject and leave it for future research. Crepey[26] sheds some light on this subject in his paper showing that the local delta(the delta of an option with local volatility) provides a better hedge than the implied delta from Black-Scholes model using both simulated and real time-series of equity-index data.

Calibration of parameters in a general sense is important because how well a model behaves usually depends on the extent of successful specification of its parameters. The approach used to estimate parameters varies according to the intended purpose. For the Black-Scholes pricing model, one way of choosing the volatility parameter is based on a forecast of future behavior of the underlying, probably also guided by statistical information from past behavior. Another way to estimate the volatility parameters is to calibrate the model with respect to the observed market price of actively traded

derivatives. This market calibration relies on the anticipation of the trading agents reflected in the price of traded options and thus the volatility parameters estimated are consistent with the market. This approach is typically used to price and hedge exotic derivatives.

There have been a series of studies about calibration of local volatility models of European style options. Rubinstein[17], Dupire[1] and Derman and Kani[4] all have constructed a discrete approximation to the risk neutral process for the underlying in the form of a bi/trinomial trees. These "implied" trees are devised to accommodate the implied volatility smile and term structure. They are effective for consistently pricing an entire class of standard European options. Nevertheless, in all cases, some ad hoc interpolation or extrapolation from the market prices is needed to permit unique determination of the local volatility function.

It is established in [1] that the local volatility function can be uniquely determined from the European options of all strikes and maturities. However, the fact is that there are only a limited number of European options with different strikes and maturities available. Therefore the problem of determining the local volatility function can be considered as an approximation problem from a finite data set with a nonlinear observation functional. Due to the limited number of available option prices, this problem is ill-posed, which means a small perturbation in the data may lead to a large change in model output. So artificially interpolating available option prices is not a good idea because false information in mispriced options would be transmitted.

Many inverse problems are ill-posed. Bouchouev and Isakov[8],[9] establish the calibration of local volatility as an inverse problem and prove the uniqueness and stability issue associated with it. Following this line, most of further studies consider the calibration problem as an inverse problem. To deal with the ill-posedness, a common technique called Tikhonov regularization is used to stabilize the problem[18]. There are several other papers in the literature which prove theoretically the uniqueness and stability issue of this ill-posed problem using Tikhonov regularization [15],[25],[7].

Lagnado and Osher[22],[23] numerically solve the inverse problem in a non-parametric space by solving a PDE to compute the update for local volatility in each step. However, this technique is very computationally demanding due to the large dimension of the problem. Bodurtha and Jermakyan [13] solve the inverse problem in a non-parametric space based on a small parameter expansion of the option value function.

Coleman and Li [2] use a cubic spline to both fit the local volatility surface and reduce the dimension of the problem. Achdou and Pironneau [30] solve the inverse problem using the Dupire equation but also some linear or cubic spline fitting is assumed. Turinici[29] proposes to calibrate the local volatility using variance of implied volatility. But again interpolation is also used in the study. All of these studies essentially consider the inverse problem in a parametric space, which renders it more tractable by artificially reducing the dimension of the problem.

Avellaneda et al[21] use a dynamic programming approach and minimize the entropy function. Crespy[25] solves this inverse problem using Tikhonov regularization in the

framework of a trinomial tree.

In this candidate exam, we still propose to solve the inverse problem in a non-parametric space, i.e., we do not implement any interpolation or extrapolation to fit the volatility surface. Since most of the inverse problems involve solving a constrained minimization problem by using the gradient information of a cost function. We also propose a new way to easily compute the gradient of the cost function without deriving the adjoint model analytically. This feature can enhance this method's capability to adapt to other calibration problems, like exotic options when the analytical adjoint model may not be easily derived. In addition, we'll attempt to study how to efficiently quantify Tikhonov regularization parameter for the inverse problem. Last, most of previous studies use a relatively small number of options to calibrate the local volatility function. This study will also investigate the feasibility of calibrating with respect to a large number of options existing in the market if it's not too ambitious..

Chapter 2

Formulation of the Problem

2.1 Local Volatility Model

The local volatility model assumes the price S of an underlying following a general diffusion process of the form:

$$\frac{dS}{S} = \mu dt + \sigma(S,t)dW_t \tag{2.1}$$

where μ is the drift, W_t is a standard Brownian motion, and the local volatility σ is a deterministic function that may depend on both the price S of the underlying and time t.

2.2 The inverse problem

Suppose we are given the market prices of European options(calls, puts, or both) spanning a set of expiration dates T_1,..., T_N . Assume that for each expiration date T_i, there are a set of options with strikes spanning from K_{i1}, ..., K_{iM_i}, where the number of strikes M_i and their prices may be different for each expiration date. Let V_{ij}^a and V_{ij}^b denote the bid and ask prices, respectively, at the present time $t = 0$ for an option with maturity T_i and strike K_{ij}.

Assume that the underlying follows the general stochastic process specified in equation (2.1). Let $V(S,t,K,T,\sigma)$ denote the theoretical price of an option with strike K and maturity T at time t and the underlying's price is S. If we also assume that the price S of the underlying follows the stochastic process specified in equation(2.1), the price function V satisfies the following generalized Black-Scholes PDE:

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

where r is the risk-free continuously compounded interest rate and q is the continuous dividend yield of the underlying. We just assume r and q are deterministic and constant in this article.

If the functional form of σ is specified, then the price of $V(S_0, 0, K, T, \sigma)$ corresponding to a spot price S_0 at the present time $t = 0$ can be uniquely determined by solving equation (2.2)together with appropriate initial and boundary conditions.

The inverse problem of option pricing or calibration to the market involves finding a local volatility function σ such that the solution of (2.2) falls between the corresponding bid and ask market quotes for any option(K_{ij}, T_i), i.e.,

$$V_{ij}^b \leq V(S_0, 0, K_{ij}, T_j, \sigma) \leq V_{ij}^a$$

for $i = 1, \ldots, N$, and $j = 1, \ldots, M_i$.

This problem is usually solved by minimizing a functional of the form

$$G(\sigma) = \sum_{i=1}^{N} \sum_{j=1}^{M_i} [V(S_0, 0, K_{ij}, T_i, \sigma) - \overline{V}_{ij}]^2 \tag{2.3}$$

where $\overline{V}_{ij} = (V_{ij}^b + V_{ij}^a)/2$ is the mean of the bid and ask prices. This $G(\sigma)$ reasonably quantifies the misfit between model predicted option prices and market prices of options. By minimizing this functional G, the model prediction would best fit the market.

By now, calibration to the market is changed into a problem of minimizing G over some general space of admissible functions. This type of problem is ill-posed because the set of price observations is discrete and finite. The function σ cannot be uniquely determined with guaranteed continuous dependence on the market price observations, which means small perturbation in the price data can result in large changes in the minimizing function. To render the problem well-posed, regularization should be imposed on the functional G. One common regularization technique in literature assumes the form of

$$F(\sigma) = G(\sigma) + \lambda \parallel \bigtriangledown \sigma \parallel^2 \tag{2.4}$$

where λ is the Tikhonov regularization parameter. This method is actually a first-order Tikhonov regularization. Another way of regularizing the problem has the form of :

$$F(\sigma) = G(\sigma) + \lambda \parallel \sigma - \sigma_0 \parallel^2 \tag{2.5}$$

where the regularization is a zeroth-order Tikhonov regularization.

2.3 Tikhonov Regularization

Tikhonov Regularization is important in solution of inverse problems in that it is the most commonly used method of regularization of ill-posed inverse problems manifested

by the fact that a small perturbation of the observation data may be leading to a big change in estimated parameters or input. In fact, every observation is prone to have observation errors, for example, errors from measuring instruments. In this candidate presentation, the observation errors mean the errors from mispriced options. There is also truncation error in our numerical scheme. These errors will all affect the amount of true information we can get from the system. As we will see in the following example, Tikhonov regularization serves as a good compromise between what we can get from the system and the degree of errors in the system. Here we will use a simple example to illustrate the idea of Tikhonov Regularization. Let us consider a linear inverse problem:

$$GM = D \tag{2.6}$$

where G is a model operator, M is the input or parameters characterizing a model and D consists of observation data. The forward problem is to find D given G. The inverse problem is to find M given D. For simplicity, we also just view G as a matrix of m rows and n columns; M and D are vectors of size of n, m respectively.

If there are less observations points than model parameters (m<n), then the number of constraints is less than the number of unknowns. (2.6) is likely to be a rank-deficient problem, which means (2.6) may have infinite solutions. If there are more observations points than model parameters (m>n), then the number of constraints is greater than the number of unknowns. Because of the random errors in observation data, (2.6) may be an inconsistent system, which means there might not be an exact solution.

Another reasonable approach to finding the best approximate solution of (2.6) is to find M that minimizes the misfit, or residual, between the observation data and theoretical prediction of the forward problem. A traditional strategy is to minimize the L2-norm of the residual:

$$\| GM - D \|_2$$

This is a general linear least square problem. A method of analyzing and solving least squares problem that is of particular interest in ill-conditioned and/or rank deficient systems is the sigular value decomposition, or SVD. In SVD, G is factored into the form of

$$G = USV^T$$

Where U is an m by m unitary matrix and V is an n by n unitary matrix, and S is an m by n diagonal matrix with diagonal elements called singular values. The singular values along the diagonal of S are customarily arranged in decreasing size, $s_1 \geq s_2 \geq \cdots \geq s_{min(m,n)} \geq 0$. Note some of the singular values may be zero.

Using pseudo inverse of G, the solution to (2.6) can be expressed as:

$$M_\dagger = V_p S_p^{-1} U_p^T D = \sum_{i=1}^{p} \frac{U_{.,i}^T D}{s_i} V_{.,i} \tag{2.7}$$

We can see that the inverse solution can become extremely unstable when one or more of the singular values, s_i, is small.

For a generalized linear least square problem, if we consider that observation data contain noise and that there is no point fitting noise exactly, it becomes evident that there can be many solutions that adequately fit the data in the sense that $\| GM - D \|_2$ is small enough. The question is how to choose a good solution.

One way of Tikhonov Regularization selects a good solution by

$$min \| M \|_2 \tag{2.8}$$

with M is one of solutions of $\| GM - D \|_2 < \delta$

Another way of Tikhonov Regularization chooses a good solution by :

$$min \| GM - D \|_2 \tag{2.9}$$

when $\| M \|_2 < \epsilon$

A third way of Tikhonov Regularization has the form of :

$$min \| GM - D \|_2^2 + \alpha^2 \| M \|_2^2 \tag{2.10}$$

where α is a regularization parameter. It can be shown that by properly choosing the δ, ϵ,α, the three problems(2.8),(2.9),(2.10) yield the same solution. We will just concentrate on interpreting and solving the damped least square problem (2.10).

The solution to (2.10) can be obtained by augmenting the least squares problem for GM=D in the following way:

$$min \left\| \begin{bmatrix} G \\ \alpha I \end{bmatrix} m - \begin{bmatrix} d \\ 0 \end{bmatrix} \right\|_2^2$$

Using the pseudo inverse of $\begin{bmatrix} G \\ \alpha I \end{bmatrix}$, the solution to (2.10) is:

$$M_\alpha = \sum_{i=1}^{k} \frac{s_i^2}{s_i^2 + \alpha_i^2} \frac{U_{.,i}^T D}{s_i} V_{.,i} \tag{2.11}$$

We neglect the derivation process here, see [3] for more details.

The objects

$$f_i = \frac{s_i^2}{s_i^2 + \alpha_i^2}$$

are called filter factors. For $s_i \gg \alpha, f_i \approx 1$, and for $s_i \ll \alpha, f_i \approx 0$. For singular values between these two extremes, as the s_i decreases, the f_i produces a monotonically decreasing contribution of corresponding model space vectors, $V_{.,i}$

For each α, there is an optimal value for both $\| M \|_2$ and $\| GM - D \|_2$. We can plot the curve of the logarithm of the optimal values of $\| M \|_2$ versus $\| GM - D \|_2$. This curve frequently takes on the characteristic L shape, as illustrated in the following figure, which is borrowed from [3]. This curve is called L-curve. It happens because $\| M \|_2$ is a strictly decreasing function of α and $\| GM - D \|_2$ is a strictly increasing function of α. The sharpness of the corner varies from problem to problem. But it is usually well-defined. One popular criterion for picking the value of α is to choose α that gives the solution closest to the corner of L-curve, which is called L-curve criterion.

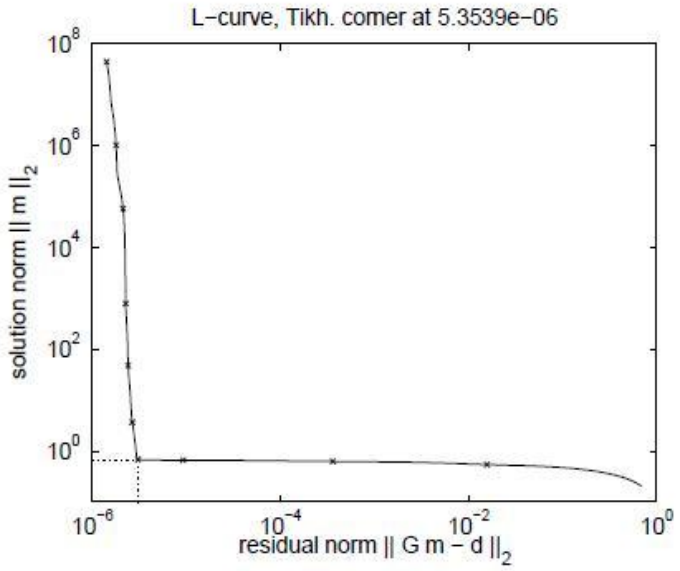

Figure 5.3: L–curve for the Shaw problem.

2.4 Solving the inverse Problem

For in-depth analysis of the stability, uniqueness and convergence of the inverse problem, we refer to studies such as [9],[25],[7].

The minimization of (2.4) or (2.5) is usually obtained with unconstrained large-scale optimization methods. Using the gradient of cost functional information to perform the minimization is a popular method. However, finding the gradient of cost function $F(\sigma)$ is not an easy task. Many studies use an adjoint method to derive the gradient. However, this method might not be applicable when the model is complicated, e.g. exotic options. This article proposes to use automatic differentiation to find the gradient of F, a method which can be adapted to more complicated models and will be discussed in detail in next section. Here, some computational issues related to minimizing cost function $F(\sigma)$ will be discussed first.

First of all, this inverse problem could easily be considered a large scale optimization

problem. For large scale optimization problems, there may be many local minima of the cost function that makes finding the global minimum a challenging problem.

The local volatility function $\sigma(S,t)$ that we want to calibrate is actually a volatility surface. To completely specify the volatility surface, we'll have to determine infinite number of points on the surface, which is impossible. But since we need to discretize the problem to solve equation (2.2), the number of σ on the surface $\sigma(S,t)$ that we want to calibrate depends on how fine the discretization is.

Say the computational domain is discretized $(0, S_{max}) \times (0, T_{max})$ into N_x segments in the S dimension and N_t segments in the t dimension, where S_{max} is the largest S allowed in the S direction, and T_{max} is the furthest maturity, then we'll have $(N_x - 1) \times N_t$ components of $\sigma(S_i, t_j)$ to estimate $(i = 1, 2, \ldots, N_x; j = 1, 2, \ldots, N_t)$.

Let $N_x = 100$, $N_t = 10$, there will be 990 parameters to estimate, which is a big number. But the discretization in this case is really not very fine. For example, the current SPX is around 1000. To compute its option price, we discretize the S domain into $N_x = 100$ segments uniformly. With this discretization, the Crank-Nicholson scheme stilll have big truncation error where we conservatively assume S_{max} to be 1000. If Smax is assumed to be greater, then the truncation error would be greater too. But increasing N_x and N_t to improve the accuracy of the solution to equation (2.2) will result in an increase in the dimension of the optimization problem.

Second, it is not computationally feasible to compute the gradient of $F(\sigma)$ with respect to σ using finite difference in that $\frac{\partial}{\partial \sigma}\overline{V}_{ij}$ needs to be computed for each option $\overline{V}_{ij}$, where σ is a vector of dimension $(N_x \times N_t)$. If there is a total of M options, then the Black-Scholes model needs to be run for $M \times (N_x - 1) \times N_t$ times.

Third, with the adjoint method introduced in section 3, we can reduce the problem to running the Black-Scholes model just M times in order to compute the gradient of $F(\sigma)$. But when M is not small, this process still takes time, which is why we introduce calibration in the framework of the dual of Black-Scholes Equation, the celebrated Dupire Equation in the next section.

Fourth, how to reasonably choose the regularization parameter in (2.4) and (2.5) has not been studied. Hopefully we'll be able to come up with a scheme to choose a suitable regularization parameter.

2.5 Calibration of local volatility surface using Dupire Equation

The reason this article starts setting up the inverse function using Black-Scholes equation is that Black-Scholes model is very familiar to everybody in mathematics of finance. Nevertheless, it is not the best model in this case to solve the calibration problem.

When the price of underlying is fixed at current time t, Dupire Equation establishes

the price of options as a function of strike (K) and maturity (T). Calling S_0 the current spot price, the price of an European call option $C(K,\tau) = C(S_0, 0, K, \tau)$ is a solution to the following Dupire Equation:

$$\frac{\partial C}{\partial \tau} - \frac{1}{2}k^2\sigma^2(K,\tau)\frac{\partial^2 C}{\partial^2 k} + (r-q)K\frac{\partial C}{\partial k} + qC = 0$$

with boundary and initial conditions as:

$$\begin{aligned} C(K,0) &= (S_0 - K)_+, K \in (0, \bar{K}), \\ C(0,\tau) &= S_0 e^{-q\tau}, \tau \in (0, \bar{\tau}], \\ C(\bar{K},\tau) &= 0, \tau \in (0, \bar{\tau}]. \end{aligned}$$

We refer to [1] for the derivation of Dupire Equation. Using the Dupire Equation, the theoretical price of all the options with different strikes and maturities can be computed by solving Dupire Equation only once. Consequently, using the adjoint method introduced in the following section to compute the gradient of cost function G only requires running the Dupire Equation only once too. Observing the similarity between the Dupire Equation and Black-Scholes Equation,the numerical code used to solve the Black-Scholes equation can also be used to solve Dupire Equation with slight modification.

Chapter 3

Finding the gradient of cost function using the adjoint model

The adjoint method has recently gained popularity in quantitative finance field . For example, Giles and Glasserman [16] use it to speed up the calculation of the sensitivity of Greeks by Monte Carlo simulation. Capriotti and Giles[14] introduce the idea of adjoint algorithm differentiation to compute the correlation Greeks. Adjoint models are powerful tools for inverse modeling and they've been already widely used in other fields for sensitivity studies, data assimilation and parameter estimation, such as [19]. Adjoint models are essentially built using automatic differentiation (AD) tools. The adjoint algorithm for obtaining the gradient of the cost functional has its advantage when the number of inputs of a model is larger than the number of outputs and the model is expensive to run. It can compute the sensitivity of output with respect to input in a much less costly fashion. In this section, we'll cover both adjoint algorithm derivation and also establish its relationship with the gradient of a cost function when the cost function is in the form of a sum of squares. We'll introduce first how to use adjoint model to compute the gradient of cost function with respect to the control variables and then discuss adjoint algorithm derivation.

3.1 Definition of adjoint operator

Suppose H is a Hilbert space with inner product. Consider a continuous linear operator $A : H \to H$.

Using the Riesz representation theorem, one can show that there exists a unique continuous linear operator $A^* : H \to H$ with the following property:

$$< AX, Y >=< X, A^*Y > \qquad for \quad all \quad X \quad and \quad Y$$

This operator A^* is the adjoint of A. If X and Y are real vectors, and A and A^* are real matrices. Then A^* is just the transpose of A. A model that uses adjoint operator

to compute the gradient of a cost function is called an adjoint model.

3.2 Adjoint Models

Consider a general dynamical system and a model describing this system. Let $D \in R^m$ be a set of observations of the system and suppose that the model can compute the values $Y \in R^m$ corresponding to these observations.

To quantify the misfit we introduce a cost function:

$$J = \frac{1}{2}(Y - D, Y - D)$$

by the choice of an appropriate inner product $(.,.)$. This implies that least-squares-fitting is intended: the smaller J is, the better the model fits the data.

Assuming the model is in the form of

$$\begin{aligned} F: \quad & R^n \to R^m \\ & X \to Y \end{aligned}$$

where $X \in R^n$ is the input or control parameters of the model. Thus, J can be expressed in terms of X by

$$J(X) = \frac{1}{2}(Y - D, Y - D) = \frac{1}{2}(F(X) - D, F(X) - D) \tag{3.1}$$

This cost function is a legitimate assessment of the misfit between the model prediction and observations. By finding the minimum of this cost function, we're looking for control parameters X that fit the model best with the observation. Namely we adjust the misfit between model output and observations subject to the model serving as a strong constraint. To find the minimum of J, the gradient of cost function J with respect to X is desirable.

If we use Taylor expansion on the cost function

$$J(X) = J(X_i) + (\bigtriangledown_X J(X_i), X - X_i) + o(\mid X - X_i \mid)$$

If we neglect the higher order terms, we have

$$\delta J = (\bigtriangledown_X J(X_i), X - X_i) = (\bigtriangledown_X J(X_i), \delta X_i) \tag{3.2}$$

Now let's suppose F is sufficiently smooth, then for each perturbation of X_i, we can approximate

$$\delta Y = (A(X_i), \delta X_i) \tag{3.3}$$

where A is the Jacobian of $F(X)$ at X_i, which is usually called the tangent linear model of F. It will approximately compute δY for arbitrage perturbation δX_i at X_i. Here we are essentially linearizing the model F.

From (3.1), the variation of cost function J around X_i is:

$$\delta J = \frac{1}{2}(\delta Y, F(X_i) - D) + \frac{1}{2}(F(X_i) - D, \delta Y) = (\delta Y, F(X_i) - D)$$

because inner product is commutative.

Using (3.3) and definition of adjoint operator, we have

$$\delta J = (A(X_i)\delta X_i, F(X_i) - D) = (A^*(X_i)(F(X_i) - D), \delta X_i) \tag{3.4}$$

where the operator A^* is the adjoint of linear operator A. $A^*(X)$ is called the adjoint model of F.

Compare (3.4) with (3.2), we have:

$$\bigtriangledown_X J(X_i) = A^*(X_i)(F(X_i) - D). \tag{3.5}$$

Equation (3.5) establishes that we can compute the gradient of cost function $J(x)$ using adjoint model of F. But why do we want to use the adjoin model to compute the gradient? The reason is that when the dimension of the input X is really large, we'll have to run $n + 1$ times the model F if using finite difference scheme to compute the gradient of cost function $J(X)$. This becomes computationally very expensive when the model F is large and complicated. By using (3.5) we can just run the adjoint model once to compute the gradient of cost function $J(X)$. A detailed analysis of the required numerical operations yields that this computation takes only $2-5$ times the computation of the cost function[6]. (3.5) is usually used to derive gradient when the dimension of the input n is greater than the dimension of the output m.

Both linear operators A and A^* depend on point X_i, at which the linearization takes place. According to (3.5), the misfit $(F(X_i) - D)$ represents the forcing of the adjoint model.

3.3 Adjoint code construction

Now the question is how to find tangent linear model A and its adjoint A^*? Suppose we want to simulate a dynamical system numerically. The development of a numerical simulation program is usually done in three steps. First, analytical differential equations are formulated. Then a discretization scheme is chosen and the discrete equations are constructed. The last step is to implement an algorithm that solves the discrete equations in a programming language. The construction of the tangent linear and adjoint code may start after any of these three steps. So there are three ways to construct the adjoint code.

If using the first methods, the analytical adjoint operator needs to be derived first.Then the analytical adjoint model needs to be discretized and solved using a numerical algorithm. However, since the commutativity rule doesn't apply to discrete operators, one has to be careful in constructing the discrete adjoint operators. If constructing the adjoint model from discrete model equations,no adjoint operator needs to be constructed explicitly. However, the coding required is both extensive and cumbersome.

This article is concerned with the third method, where the adjoint code is developed directly from the numerical code of the model. For one thing, analytical adjoint operator need not be derived analytically, which makes it possible for application to complicated models. For another thing, the methodology of automatic algorithm differentiation can be used here to speed up the coding process. Adjoint model is actually just the reverse mode of automatic differentiation(AD) for a model, while tangent linear model is just the forward mode of AD for a model. To understand it, we need to first know what the forward mode of AD for a model is. A numerical model is an algorithm that can be viewed as a composition of differentiable functions F, each representing a statement in the numerical code:

$$Y = F(X) = (F_k \circ F_{k-1} \circ \ldots \circ F_1 \circ F_0)(X)$$

where the l-th step has an explicit representation:

$$\begin{array}{rl} F_l: & R^{n_{l-1}} \to R^{n_l} \quad (l = 1, \ldots, k) \\ & Z^{l-1} \to Z^l \end{array}$$

The vector Z^l holds n_l intermediate results that are valid after the lth step of the algorithm. Using chain rule, the Jacobian matrix of $F(X)$ or tangent linear model can be expressed as:

$$A = F'(X) = F'_k \mid_{Z^{k-1}} \cdot F'_{k-1} \mid_{Z^{k-2}} \cdot \ldots \cdot F'_1 \mid_{Z^1} \cdot F'_0 \mid_X \tag{3.6}$$

Since matrix product is associative, this product can be evaluated in any order. When evaluated in forward mode, the intermediate derivatives are computed in the same order as the model computes the composition.

A^* is the transpose of A. (3.6) says that A^* is a product of the transpose of intermediate Jacobian matrices in (3.6). Now that we have an idea of the explicit expression of adjoint model A^*, to fully understand the logic behind constructing the adjoint codes automatically, knowledge of adjoint variable, variables that serve as input or intermediate results for the adjoint model A^*, is necessary.

Consider a simple case when the mapping F is: $F : R^n \to R^1$, and let the decomposition of F be:

$$F = \bigodot_{l=1}^{K} F_l$$

For an intermediate result Z^l,

$$Z^l = \overset{l}{\underset{i=1}{\odot}} F_l(X) = F_l(Z^{l-1}) \qquad (1 \leq l \leq K)$$

The intermediate variation depends on the previous intermediate variation by

$$\delta Z^l = \frac{\partial F_l(Z^{l-1})}{\partial Z^{l-1}} \mid_{Z^{l-1}} \delta Z^{l-1} \tag{3.7}$$

The adjoint of an intermediate result δZ^l is defined as the gradient of F with respect to the intermediate result:

$$\delta^* Z^l = \bigtriangledown_{Z^l} \overset{K}{\underset{i=l+1}{\odot}} F_i \mid_{Z^l} \tag{3.8}$$

In a moment, it will be clear why it is defined this way. This definition also explains the meaning of adjoint variables.

From the definition of gradient of (3.2), we obtain

$$\delta F = (\delta^* Z^l, \delta Z^l) \tag{3.9}$$

(3.9) holds for every level of l, so

$$\begin{aligned} (\delta^* Z^{l-1}, \delta Z^{l-1}) &= (\delta^* Z^l, \delta Z^l) \\ &= (\delta^* Z^l, \tfrac{\partial F_l(Z^{l-1})}{\partial Z^{l-1}} \mid_{Z^{l-1}} \delta Z^{l-1}) \quad using(3.7) \\ &= ((\tfrac{\partial F_l(Z^{l-1})}{\partial Z^{l-1}})^* \mid_{Z^{l-1}} \delta^* Z^l, \delta Z^{l-1}) \end{aligned}$$

This holds for all δZ^{l-1}, so that

$$\delta^* Z^{l-1} = (\frac{\partial F_l(Z^{l-1})}{\partial Z^{l-1}})^* \mid_{Z^{l-1}} \delta^* Z^l \tag{3.10}$$

where $(\frac{\partial F_l(Z^{l-1})}{\partial Z^{l-1}})^* \mid_{Z^{l-1}}$ is the transpose of $\frac{\partial F_l(Z^{l-1})}{\partial Z^{l-1}} \mid_{Z^{l-1}}$. Equation (3.10) just illustrates that adjoint model A^* is a product of the transpose matrix of $\frac{\partial F_l(Z^{l-1})}{\partial Z^{l-1}} \mid_{Z^{l-1}}$ evaluated in reverse order, which is consistent with what's said right after (3.6). This also explains why the adjoint variable is defined as (3.8).

Knowing what adjoint model is and what adjoint variable stands for, we're ready to code the adjoint model. The general idea is that the adjoint code is written backward from the tangent linear model line by line, subroutine by subroutine. This could be a cumbersome process. Fortunately, for each kind of statement in a code, simple rules exist for constructing its corresponding adjoint statements, which makes AD in reverse mode possible[20]. In constructing the adjoint code, a piece of adjoint code fragment will be created for each code statement in the model code. The adjoint code fragments are then composed in reverse order compared to the model code. A simple example is given in the following section.

3.4 Example of constructing the adjoint code for a simple model

The simple example provided here is just used in order to demonstrate the concept of the adjoint model of a model, the meaning of adjoint variables and how to code adjoint model easily. It is not intended to cover exhaustively the general principles behind the reverse mode of automatic differentiation.

$F = X * sin(Y^2)$, F is a function of variable X and Y, and assume X,Y, and F all depend on time t.

To get the adjoint of this model $F(X, Y)$, we first generate the Tangent linear model

$$\delta F^n = sin((Y^{n-1})^2)\delta X^{n-1} + X^{n-1} * cos((Y^{n-1})^2)2Y^{n-1}\delta Y^{n-1}$$

Using the Jacobian, this can be expressed as:

$$\begin{pmatrix} \delta F \\ \delta Y \\ \delta X \end{pmatrix}^n = \begin{pmatrix} 0 & cos((Y^{n-1})^2)2Y^{n-1} & sin((Y^{n-1})^2) \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \delta F \\ \delta Y \\ \delta X \end{pmatrix}^{n-1}$$

where n denotes time step. Given initial perturbation of δX and δY, this model computes how the perturbation would be transmitted after linearizing the model F.

Thus the adjoint model would be

$$\begin{pmatrix} ADF \\ ADY \\ ADX \end{pmatrix}^{n-1} = \begin{pmatrix} 0 & 0 & 0 \\ cos((Y^{n-1})^2)2Y^{n-1} & 1 & 0 \\ sin((Y^{n-1})^2) & 0 & 1 \end{pmatrix} \begin{pmatrix} ADF \\ ADY \\ ADX \end{pmatrix}^n \tag{3.11}$$

ADF, ADY and ADX represents the adjoint variable of F, Y and X respectively.

(3.11) can be written as:

$$\begin{aligned} ADY &= ADY + ADF * X * \cos(Y * Y) * 2 * Y \\ ADX &= ADX + ADF * \sin(Y * Y) \\ ADF &= 0.0 \end{aligned}$$

The assignment of $ADF = 0.0$ must be the last one since its previous value is used by all other corresponding adjoint assignments. That ADF needs to be set to zero in the end is because the previous value of F is overwritten by executing the assignment. Consequently, the previous value has no influence on the function output considering ADF measures the gradient of output with respect to F.

In practice, it is not necessary to write each piece of code in this way to find out its adjoint counterpart. Since generating adjoint model is essentially finding derivative

in reverse order, some standard procedures can be followed to speed up the process. Some AD tools also exist to do this job though some afterward debugging is very necessary, such as the Tangent linear and Adjoint Model Compiler($TAMC$), available at http://www.autodiff.com/tamc.

This adjoint model computes the derivative of $\frac{\partial F}{\partial X}$, $\frac{\partial F}{\partial Y}$ (or sensitivity) reversely. When integrated backward in time, this adjoint model would enable us the find out source area that leads to the final perturbation of ADF.

3.5 Test of accuracy of the adjoint code

The generation of correct adjoint code for the simple example above is trivial. But for a complicated system, it is not an easy task and thus it becomes especially necessary to test the correctness of adjoint code. The adjoint code is tested using the following identity, see Navon et al[12]

$$< A\delta X, A\delta X >=< \delta X, A^*(A\delta X) > \quad for \quad any \quad \delta X \tag{3.12}$$

where δX is an arbitrary perturbation vector. It is also the input of code A. A represents the tangent linear code or a segment of it, say a subroutine, a do loop or even a single statement. A^* is the adjoint of A. If (3.12) holds within machine accuracy, it can be said that the adjoint is correct versus the tangent linear code.

1. Test of accuracy of the Tangent Linear Model(TLM)
   To use (3.12) test the correctness of adjoint model, the tangent linear model A is used. Thus first we need to test the correctness of tangent linear model A. The accuracy of tangent linear model determines the accuracy of the adjoint model and the accuracy of the gradient of cost function with respect to the control variables.

   To verify A, we use the fact of (3.3) that A is linearization of the model F:

   $$F(X + alpha * \delta X) - F(X) = A(alpha * \delta X) + O(alpha^2)$$

   where δX is an arbitrary perturbation around X.

   We compare the result of TLM with the difference of the twice model call, with and without perturbation respectively. If the TLM is right, then the ratio between these two,

   $$r = \frac{F(X + alpha * \delta X) - F(X)}{A^*(alpha * \delta X)} = 1 + O(alpha) \tag{3.13}$$

   will approach zero as α gets close to zero.

2. Verification of Adjoint Model
   After verifying the TLM, We can then use it to test the adjoint model using the adjoin identity:

$$< A\delta X, A\delta X >=< \delta X, A^*(A\delta X) > \quad for \quad any \quad X \tag{3.14}$$

   where δX is a perturbation around X. If (3.14) holds within machine precision, then it can be said that the adjoint code is correct with respect to tangent linear model.

3. Verification of Gradient
   Even though the TLM and adjoint models are correct, the gradient generated by the adjoint code needs to be verified because the accuracy of the adjoint gradient not only depends on the accuracy of the tangent linear and adjoint model, but also on the approximation involved in linearizing the cost function (3.3).

   Suppose the initial X has a perturbation αh, where α is a small scalar and h is a vector of unit length. According to Taylor expansion, we get the cost function:

   $J(X + \alpha h) = J(X) + \alpha h^T \bigtriangledown J(x) + O(\alpha^2)$

   We can define a function of α as:

$$\Phi(alpha) = \frac{J(X + \alpha h) - J(X)}{\alpha h^T \bigtriangledown J(X)} = 1 + O(\alpha) \tag{3.15}$$

   So as α tends to zero but not close to machine precision, this ratio Φ should be close to 1. Here the gradient of $\bigtriangledown J(X)$ is calculated by the adjoint model.

   This test is usually called α-test.

Chapter 4

Numerical Results

In this example, we use the closed option price for SPX on Apr 2ND as our raw data to estimate the volatility surface. We will calibrate the local volatility function using Dupire Equation, i.e., we consider local volatility as a function of strike and maturity. To simplify the problem, we only look at options with maturity on June 18th, and also assume volatility is just a function of strike.

There are a total of 121 options available. But we just use 45 of those options that are near the money ($0.9S < K < 1.1S$) for two reasons. First, calibration to the market is usually performed with respect to liquid instruments in the market. Trading volumes for options that are deep in or out of the money are really low. Second, in our numerical test, we find out that the price of options that are deep in or out of the money do not depend on volatility very much: the dependence is small and they cannot pass neither the tangent linear test nor gradient test.

The SPX on Apr 2ND was 1178.10. The options that we selected for calibration had strikes ranging from 1070 to 1300. The interest rate r was 0.0015 which was attained from 13-week treasury bill and the dividend rate q was assumed to be zero. The computational domain$(0, K_{max}) \times (0, T)$ is discretized uniformly into 2000 grid points in the K direction and 10 levels in the T direction withK_{max} = 2000 and T = maturity = 0.21096.

From here we also see the large scale nature of the problem. Dividing the K direction of computational domain into 2000 segments means there is a total of 1999 parameters that should be estimated because sigma is a function of K. For each K, there is a sigma corresponding to it. If sigma is assumed to be a function of both K and T, the number of parameters to be estimated is even larger.

After constructing the code to solve the Dupire Equation, we use TAMC automatic differentiation to generate the code for its tangent linear and adjoint model. After debugging the code, we use the following test to test the validity of the tangent linear code and adjoint code.

4.1 Verification of the Tangent Linear Model

Using (3.5.3), X is volatility in our case. X is constructed by multiplying a random number from (0,1) to each component of X. The ratio r with respect to alpha is listed in table1 and plotted differently in Figure 1 and Figure 2. We can see that as α decreases from 10^{-2} to 10^{-11}, the ratio r approaches 1 with high accuracy. When α is between 10^{-6} to 10^{-8}, the ratio r is closest to 1.

α	r
$0.1000000000E+00$	$0.1032539117E+01$
$0.1000000000E-01$	$0.1003277163E+01$
$0.1000000000E-02$	$0.1000327956E+01$
$0.1000000000E-03$	$0.1000032798E+01$
$0.1000000000E-04$	$0.1000003282E+01$
$0.1000000000E-05$	$0.1000000326E+01$
$0.1000000000E-06$	$0.1000000720E+01$
$0.1000000000E-07$	$0.1000000248E+01$
$0.1000000000E-08$	$0.9999700354E+00$
$0.1000000000E-09$	$0.1000381809E+01$
$0.1000000000E-10$	$0.9990800710E+00$
$0.1000000000E-11$	$0.1064862770E+01$
$0.1000000000E-12$	$0.1665776365E+01$
$0.1000000000E-13$	$0.3622784142E+02$
$0.1000000000E-14$	$0.4005300720E+04$

Table1

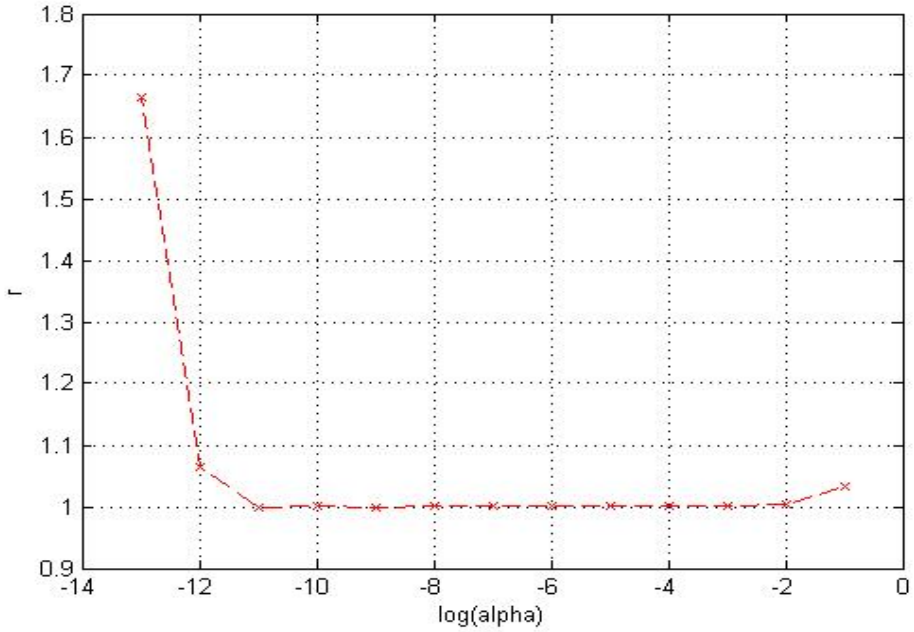

Figure1

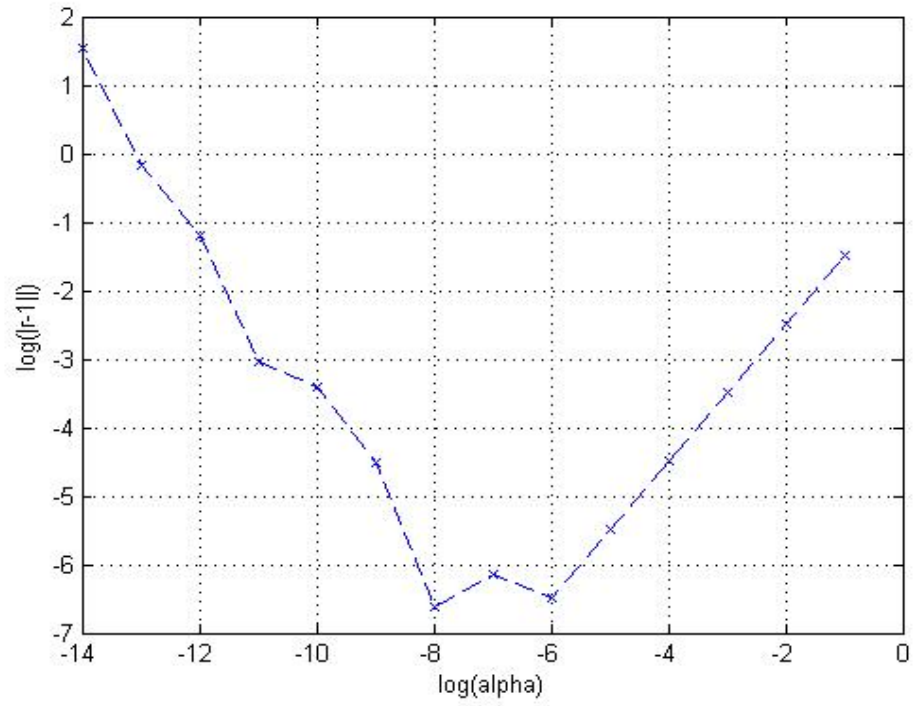

Figure 2

4.2 Transpose Test

$$< A\delta X, A\delta X >= 1630954.41614603$$

$$< \delta X, A^*(A\delta X) >= 1630954.41614618$$

We checked the adjoint model segment by segment, do loop by do loop, and subroutine by subroutine. With double precision, the identity (3.12) is accurate within 13 digits. This verified the correctness of adjoint model against TLM.

4.3 Gradient Verification

Instead of minimizing (2.3), the cost function is constructed as a weighted sum of squares of the difference:

$$J(\sigma) = \sum_{i=1}^{N} \sum_{j=1}^{M_i} [(V(S_0, 0, K_{ij}, T_i, \sigma) - \overline{V}_{ij})w(ij)]^2 \tag{4.1}$$

where $V(S_0, 0, K_{ij}, T_i, \sigma)$ is the computed option price, $\overline{V}_{ij} = (V_{ij}^b + V_{ij}^a)/2$ is the mean of the bid and ask price, $w(ij)$ is a weight that represents our confidence in the difference between computed price and observed price. This weight is necessary if we want to give equal weight to price difference between $V(S_0, 0, K_{ij}, T_i, \sigma)$ and $\overline{V}_{ij}$ for different options. A natural choice of the weight function that describes our confidence in an option price is the bid and ask spread of the option. So in our numerical experiment, we use 1/(bid-ask) as our weight function.

α-test with respect to the above cost function is performed and the result is listed in table 2. Figures 3 and 4 show the variation of $\Phi(\alpha)$ and $\log| \ 1 - \Phi(\alpha) \ |$. One can see that as α decreases form 10^{-5} to 10^{-15}, $\Phi(\alpha)$ approaches unity with high accuracy until α is close to 10^{-15}. This agrees with our expectation that as α gets close to machine precision, we don't expect the α-test still holds. This also means that for perturbations within the range of 10^{-5} to 10^{-15}, the gradient calculated from the adjoint model is reliable.

α	$\Phi(\alpha)$
0.100000000000000	$9.531840348585277E-004$
$1.000000000000000E-002$	$2.191575720234569E-002$
$1.000000000000000E-003$	0.594217183916692
$1.000000000000000E-004$	1.14053859511066
$1.000000000000000E-005$	1.01354376144459
$1.000000000000000E-006$	1.00134884779927
$1.000000000000000E-007$	1.00013482926537
$1.000000000000000E-008$	1.00001348318852
$1.000000000000000E-009$	1.00000136139467
$1.000000000000000E-010$	1.00000018208732
$9.999999999999999E-012$	1.00000002313088
$1.000000000000000E-012$	1.00000458015879
$1.000000000000000E-013$	1.00002898391480
$1.000000000000000E-014$	0.999068189779104
$1.000000000000000E-015$	0.995200111444452
$1.000000000000000E-016$	1.10314937913147
$1.000000000000000E-017$	1.90083677818741
$1.000000000000000E-018$	1.20358660627587
$1.000000000000000E-019$	16.6011945693224

Table2

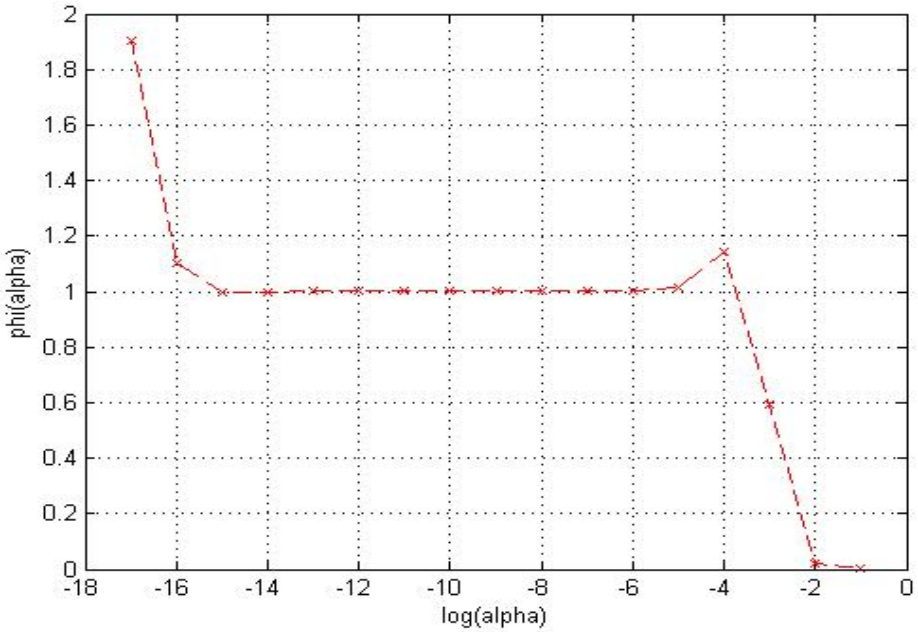

Figure3

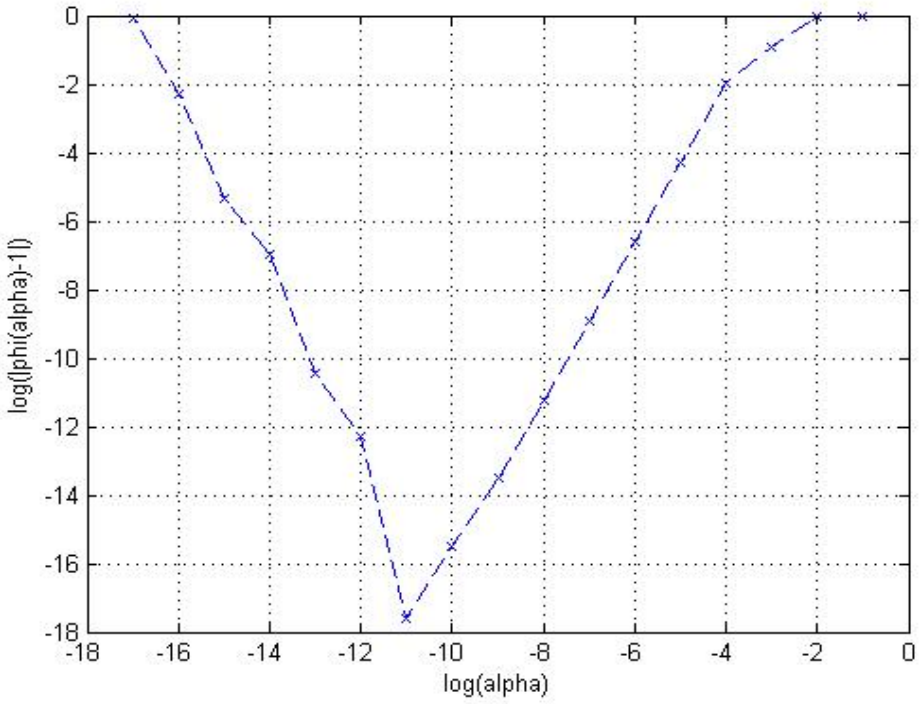

Figure4

4.4 Minimization and Discussion

Up to this point, we haven't performed any regularization yet. We just minimized the cost function (4.1). Setting the initial $\alpha = 0.5$, we performed the minimization using LBFGSB. The gradient of the cost function is derived from adjoint model. The cost function is reduced from an order of 10^5 to an order of 10^2 within 2 minutes. The option prices computed using the optimal volatility are plotted in figure 5. We can see that for options whose strikes are not immediately around the stock price ($0.9S < K < 0.95S, 1.05S < K < 1.1S$), the computed option prices approach the market price well. But for options whose strikes are in immediate vicinity of the stock price($0.95S < K < 1.05S$), the computed option prices are about two or three times the bid-ask spread away from the market prices. The optimal volatility $\sigma(K)$ is plotted with respect to K/S_0 in figure 6. We can see that the curve of $\sigma(K)$ has the property of a smile: the volatility near the money is smaller than the volatility that's deep in or out of money. When K is either far away from the money or deep in the money, the optimal volatility doesn't change too much from the initial value.

We also tested the impact of minimization using different initial σ. However, the result is not as good. The cost function could still be minimized significantly by a magnitude of 10^3. But the cost function is much greater than the optimal cost function obtained above. The optimal $\sigma(K)$ obtained is also not as good

by exhibiting some negative values at some region. This means that the success of this large scale minimization problem depends on the initial input, which is in another way of saying that the large scale minimization may get trapped at some local minima instead of a global minimum. We also tested the minimization using other optimization algorithms such as Sequential Quadratic Programming and Conjugate Gradient method. No better results were obtained either. This may be due to the fact that the cost function is not as yet regularized, or something more fundamental, the identifiability of the local volatility, which we just realize recently. These questions lead to our future research.

4.5 Future Research

At the moment this research has not as yet addressed the Tikhonov regularization. The next step will consist in implementing a regularization on the cost function to find out if we can further reduce the cost function and possibly achieve convergence to the global minimum.

We observe from Figure 6 that $\sigma(K)$ for K that is deep in or out of the money doesn't change too much from the initial value. One way to reduce the dimensionality of the problem may be by introducing nonuniform discretization the computational domain: to have coarse grids when K is deep in or out of the money and to have fine grids when K is near the money. Another alternative for reducing the dimension may be to use adaptive grid generation techniques.

One issue that is probably more fundamental and important comes into our sight recently: the identifiablity of the local volatility parameters in this inverse problem. The problem of identifiablity in parameter identification is related to the uniqueness problem in parameter identification. Whether a parameter is identifiable or not has important implications in an inverse problem of parameter identification. When a parameter is identifiable, one can try to develop an algorithm that gives the unique exact solution of the inverse problem. When parameters are not identifiable, one must expect that a procedure of solution of the inverse problem will give a nonunique solution, generally dependent on some parameters or initializers for iterative algorithms. It is possible sometimes to find methods that lead to a unique solution of the inverse problem, which in general could be different from the correct solution. Here we just want to briefly introduce the definition of identifiability from [10].

The parameter λ is said to be identifiable if from

$$\lambda, \lambda' \in \wedge_{ad} \qquad h = H_f(\lambda) \qquad h' = H_f(\lambda') \qquad h' = h$$

it follows that

$$\lambda = \lambda'$$

where $\wedge_{ad}$ is the admissible parameter subset, and H_f is a mapping from $\wedge_{ad}$ to the observation space. It requires that for every different observation, there is a

different set of paramters. This condition, however, cannot be achieved in any practical problem due to modeling and observation errors.

Relaxation or extension of the definition of identifiablity comes in many forms depending on the purpose of parameter identification, properties of observation data, and the nature of the underlying model. Good examples can be found in [28]. Identifiablity of this inverse problem will also be another important topic for our future research.

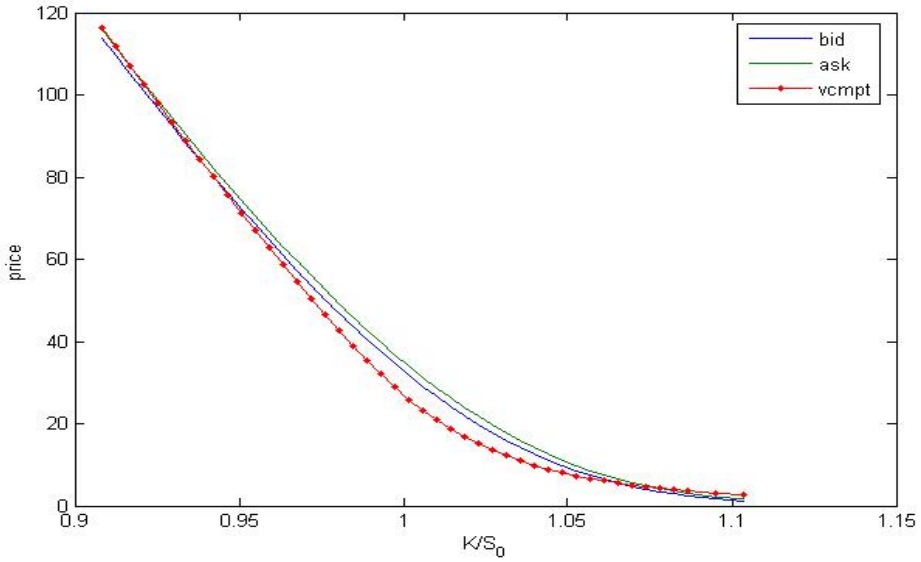

Figure5

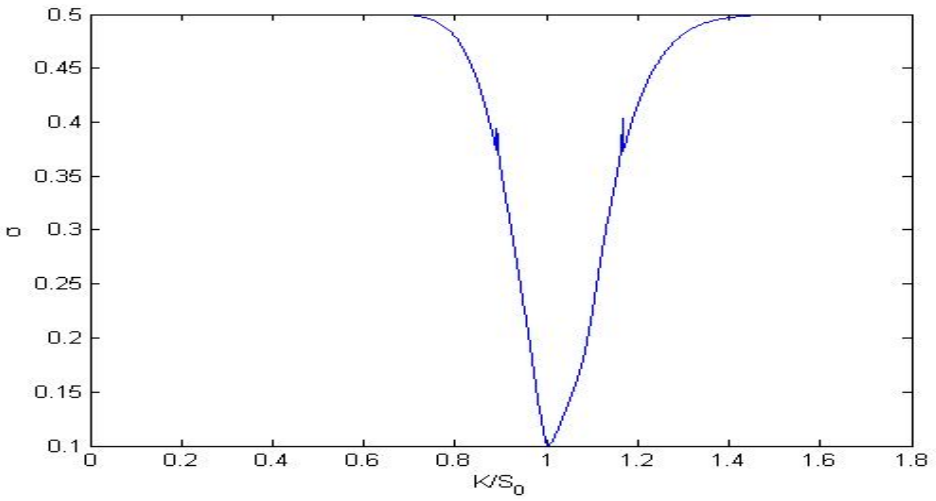

Figure6

Bibliography

[1] B.DUPIRE. Pricing with a smile. *Risk*, 7:18–20, 1994.

[2] T.F. COLEMAN and A.VERMA. Reconstructing the unknown local volatility function. *The Journal of Computational Finance*, 2:77–100, 1999.

[3] R.C.ASTER B.BORCHERS C.THURBER. *Parameter Estimation and Invese Problems*. Elesvier Academic Press, Burlington, 2005.

[4] E.DERMAN and I.KANI. Riding on a smile. *Risk*, 7:32–39, 1994.

[5] F.BLACK and M.SCHOLES. The pricing of options and corporate liabilities, journal of political economy. *Journal of Political Economy*, 81:637–659, 1973.

[6] A. GRIEWANK. On automatic differentiation. In M.IRI and K.TANABE, editors, *Mathematical programming: Recent Developments and Applications*, pages 83–108. Kluwer Academic Publishers, Dordrecht, 1989.

[7] H.EGGER and H.W.ENGL. Tikhonov regularization applied to the inverse problem of option pricing: convergence analysis and rates. *Inverse Problems*, 2005.

[8] I.BOUCHOUEV and V.ISAKOV. The inverse problem of option pricing. *Inverse Problems*, 13:L11–L17, 1997.

[9] I.BOUCHOUEV and V.ISAKOV. Uniqueness,stability and numerical methods for the inverse problem that arises in financial markets. *Inverse Problems*, 15:R95–116, 1999.

[10] I.M.NAVON. Practical and theoretical aspects of adjoint parameter estimation and identifiabiliy in meteorology and oceanography. *Dynamics of Atmospheres and Oceans*, 27:55–59, 1997.

[11] J.C.HULL. *Options,futures, and other derivatives*. Pearson Education, New Jersey, 2009.

[12] I.M.NAVON X.ZOU J.DERBER and J.SELA. Variational data assimilation with an adiabatic version of the nmc spectral model. *Monthly Weather Review*, 120:1433–1446, 1992.

[13] J.JR.BODURTHA and M.JERMAKYAN. Nonparametric estimation of an implied volatility surface. *The Journal of Computational Finance*, 2:29–61, 1999.

[14] L.CAPRIOTTI and M.GILES. Fast correlation greeks by adjoint algorithmic differentiation. *Risk*, 23:79–83, 2010.

[15] L.JIANG Q.CHEN L.WANG and J.E. ZHANG. A new well-posed algorithm to recover implied local volatility. *Quantitative Finance*, 3:451–457, 2003.

[16] M.GILES and P.GLASSMAN. Smoking adjoints: Fast calculation of greeks in monte carlo calculation. *Technical Report*, NA-05/15, 2005.

[17] M.RUBINSTEIN. Implied binomial trees. *The Journal of Finance*, 49:771–818, 1994.

[18] P.C.HANSEN. *Rank-Deficient and Discrete Ill-Posed Problems: numerical aspects of linear inversion*. Society for Industrial and Applied Mathematics(SIAM), Philadelphia, 1998.

[19] R.GIERING. Tangent linear and adjoint biogeochemical models. In P.Rayner N.Mahowald R.G.Prinn D.E.Hartley P.Kasibhatla, M.Heimann, editor, *Inverse methods in global biogeochemical cycles*, pages 33–47. American Geophysical Union, Washington DC, 2000.

[20] R.GIERING and T.KAMINSKI. Recipes for adjoint code construction. *ACM on Transactions on Mathematical Software*, 24:437–474, 1998.

[21] M.AVELLANEDA C.FRIDMAN R.HOLEMS and D.SAMPERI. Calibrating volatility surface via relative entropy minimization. *Applied Mathematical Finance*, 4:667–686, 1997.

[22] R.LAGNADO and S.OSHER. Reconciling difference. *Risk*, 10:79–83, 1997.

[23] R.LAGNADO and S.OSHER. A technique for calibrating derivative security pricing models: numerical solution of an inverse problem. *The Journal of Computational Finance*, 1:13–25, 1998.

[24] R.MERTON. Option pricing when underlying stock returns are discontinuous. *Journal of Financial Econommics*, 3:125–44, 1976.

[25] S.CREPEY. Calibration of the local volatility in a generalized black-scholes model using tikhonov regularization. *SIAM on Journal of Mathematical Analysis*, (5), 2003.

[26] S.CREPEY. Delta-hedging vega risk. *Quantitative Finance*, 4(5), 2004.

[27] S.L.HESTON. A closed-form solution for options with stochastic volatility with application to bond and currency options. *The Review of Financial Studies*, 6:327–343, 1993.

[28] N.Z. SUN and W.W.-G. YEH. Coupled inverse problems in groundwater modeling 2. identifiability and experimental design. *Water Resourses Research*, 26(10):2527–2540, 1990.

[29] G. TURINICI. Calibration of local volatility using the local and implied instantaneous variance. *The Journal of Computational Finance*, 13(2), 2009.

[30] Y.ACHDOU and O.PIRONNEAU. *Computational Methods for Option Pricing*. Society for Industrial and Applied Mathematics(SIAM), Philadelphia, 2005.