THE FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCES

CALIBRATION OF LOCAL VOLATILITY MODELS AND PROPER ORTHOGONAL DECOMPOSITION REDUCED ORDER MODELING FOR STOCHASTIC VOLATILITY MODELS

By

JIAN GENG

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

Degree Awarded:
Summer Semester, 2013

Jian Geng defended this dissertation on June 14, 2013.

The members of the supervisory committee were:

Ionel Michael Navon
Professor Directing Dissertation

Bettye Anne Case
Professor Co-Directing Dissertation

Rob Contreras
University Representative

Giray Ökten
Committee Member

Alec N. Kercheval
Committee Member

Brian Ewald
Committee Member

The Graduate School has verified and approved the above-named committee members, and certifies that the dissertation has been approved in accordance with the university requirements.

This thesis is dedicated to my wife Yingyun, my parents, my brother, and our expected daughter.

ACKNOWLEDGMENTS

First of all, I would like to thank my major advisor professor I. Michael Navon for patiently mentoring me for the last five years. He introduced me to the area of optimal control, parameter calibration, and POD reduced order modeling. Without his guidance, I would not have gotten to the stage where I am right now. His profound knowledge in optimal control, hardworking style, and passion for research has benefited and inspired my PhD research. I am very fortunate to be one of his PhD students. It was a pleasant experience to work with him.

Second, I want to express my gratitude to my co-advisor professor Bettye A. Case. Dr. Case was the first professor that I met in the Mathematics department in FSU. I still have a vivid impression of that date of the graduate orientation. Ever since then, Dr. Case was closely involved in every step of my maturing from a graduate student to a PhD, on and off school. She genuinely cares about students especially about me. Working for her as the administrative TA was a great experience and I learned a lot from it. I was blessed to have Dr. Case as my co-advisor and to be a student of her own.

I want to extend my thanks to professor Giray Ökten, professor Alec N. Kercheval, and Dr. Brian Ewald for being my PhD committee members and for what I learned from their courses. I also want to thank professor Rob Contreras for joining my Ph.D. committee as the University Representative. I thank Dr. Kopriva and Dr. Nichols for the various courses that I signed up with them. I want to thank Dr. Kirby for training us to be excellent TAs. The wonderful TA training programs and the solo classes, which I did not like in the first place, gave me a great opportunity to hone my presentations skills, which was proved to be very useful during my job hunting. I also want to specially thank Dr. Mike Mesterton-Gibbons, my wife's advisor, and his family, for their nice friendship and hospitality. Dr. Xiao Chen, a previous student of Dr. Navon, helped me overcome a problem when it could have gotten stagnant, which I sincerely appreciate.

I want to thank the Department of Mathematics in general for providing a good environment for students like me to study financial mathematics.

Last, I owe my thanks to my family members especially my beloved wife Yingyun, for their support during my PhD study. They are the light and meaning of my life.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

There are two themes in this thesis: local volatility models and their calibration, and Proper Orthogonal Decomposition (POD) reduced order modeling with application in stochastic volatility models, which has a potential in the calibration of stochastic volatility models.

In the first part of this thesis (chapters II-III), the local volatility models are introduced first and then calibrated for European options across all strikes and maturities of the same underlying. There is no interpolation or extrapolation of either the option prices or the volatility surface. We do not make any assumption regarding the shape of the volatility surface except to assume that it is smooth. Due to the smoothness assumption, we apply a second order Tikhonov regularization. We choose the Tikhonov regularization parameter as one of the singular values of the Jacobian matrix of the Dupire model. Finally we perform extensive numerical tests to assess and verify the aforementioned techniques for both local volatility models with known analytical solutions of European option prices and real market option data.

In the second part of this thesis (chapters IV-V), stochastic volatility models, POD reduced order modeling are introduced first respectively. Then POD reduced order modeling is applied to the Heston stochastic volatility model for the pricing of European options. Finally, chapter VI summaries the thesis and points out future research areas.

CHAPTER 1

INTRODUCTION

"At its core, the study of finance is fundamentally about the trade-off between risk and expected return. Various measures have been proposed to operationalize the risk component of this trade-off, ..." (Peter Carr on volatility derivatives [16]). Based on the assumption of a constant volatility, the celebrated Black-Scholes model [11] can be used to evaluate European style options easily by using an estimated volatility or forecast volatility as an input. If this model is true, then the implied volatility for all options under the same underlying with different strikes and maturities would be the same. However, in the market, it is usually observed that Black-Scholes implied volatility varies with strikes and maturity, which is known as the "volatility smile" [42].

There have been various attempts to extend the Black-Scholes theory to account for the volatility effect. One class of models considers the volatility term as a deterministic volatility function that depends on the spot price and time; these are usually referred to as local volatility models. For this class of models, the volatility contains only one source of randomness and retains completeness, i.e., the ability to hedge options with the underlying asset. Another class of models introduces another source of risk, such as stochastic volatility [40] or jumps [51]. In this thesis, we focus on the calibration of local volatility models and a proper orthogonal decomposition (POD) reduced order modeling approach, which has the potential to speed up the calibration of stochastic volatility models. The importance of calibration is obvious in that how well a volatility model can be utilized to price a financial derivative depends on how well the model can be calibrated. This chapter consists of two parts: one reviewing the research on the calibration of local volatility models, and the other introducing POD reduced order modeling.

1.1 Calibration of local volatility models

The local volatility model is an extension of the Black-Scholes constant volatility model [11] aimed at explaining the volatility smiles observed in the market. It assumes the volatility term is a deterministic function of both stock price and time. Dupire [27] established that the local volatility function can be uniquely derived from European option prices given the existence of European options with all strikes and maturities. However, in the market, there are only a limited number of available European options with discrete strikes and maturities. Up to the current date, there have been a large number of studies addressing the

reconstruction of the local volatility function from the limited number of options available in the market.

Lagnado and Osher [48] first solved the calibration problem in a PDE framework without assuming any shape of the local volatility function, ie, a non-parametric approach. They used the first order derivatives of the volatility surface to regularize the inverse problem. Most subsequent research followed the same regularization approach, see [13, 14]; [12]; [47, 46]; [25]; [28]; [39]; [1] and [65]. However, the recovered volatility surface is usually very rough and it is hard to discern any patterns. [18]; [1] and [65] solved the calibration problem using a parametric approach: the volatilities at several specially chosen points on the volatility surface are computed first, and then the volatility surface is constructed from those points using either linear interpolation or cubic splines. By parameterizing the volatility surface, this approach reduces the dimension of the calibration problem. It works well when the chosen points can represent well the key regions of the true volatility surface. However, it runs the danger of allowing too few degrees of freedom to explain the data. The recovered volatility surface is still either too rough (for the linear interpolation case) or subject to extreme values (for the cubic spline case), especially for the market data.

In this thesis, we still use a non-parametric approach for the calibration of the local volatility surface with the only assumption that a smooth volatility surface is preferable to a non-smooth volatility surface. This assumption is inspired by "Occam's razor", a principle from the 14^{th} century philosopher William of Ockham, who argued that simpler explanations should be preferred to more complicated explanations. The same idea was introduced for solving nonlinear inverse problems by [19]. We hope that by imposing the smoothness assumption a simpler solution can be obtained from which some patterns may be detectable. The smoothness preference assumption has already been adopted in the approach in [18]; [1] and [65] when cubic splines are used to connect the volatility surface. Another reason for the smoothness assumption is due to the fact that this calibration problem is an underdetermined problem. A second order regularization, which follows from the smoothness assumptions, adds more constraints to the problem than a first order regularization.

There have been some theoretical studies about the stability, uniqueness and convergence of the calibration problem such as [13, 14]; [53]; [47]; [25]; [28] and [39]. However, to the author's knowledge, there is no conclusive answer as yet. We will not address these issues in this thesis. We demonstrate the robustness of our calibration approach by extensive numerical tests with both theoretical local volatility models with known analytical solutions for European option prices and the real market option prices. The novelty of the calibration method in this thesis consists in the use of second-order Tikhonov regularization and the way we choose the Tikhonov regularization parameter. (See also [22] and [3])

1.2 Proper orthogonal decomposition reduced order modeling with application in stochastic volatility models

Although local volatility models are still very popular in the industry due to their simplicity, the dynamics of local volatility models are inconsistent with the market, an

example of which will be introduced in chapter 2. Stochastic volatility models assume that the volatility of the asset return is also a random process, which is usually correlated with the asset price. They have better dynamics of the volatility surface compared to the local volatility models, see [5]. For some specific stochastic volatility models, an analytical solution for vanilla options exists, such as the Heston volatility model [40] and the stochastic alpha beta rho (SABR) volatility model [37]. Because there are two sources of randomness, computing the price of exotic financial derivatives under the stochastic volatility models is usually expensive since there are two random processes to be simulated using a Monte Carlo approach. The PDE approach usually involves a 2 dimensional partial differential equation.

In this thesis, we explore a proper orthogonal decomposition (POD) reduced order modeling approach to approximate the solution, in which only a small dimensional ODE would be required to compute the solution. Reduced order modeling has already found applications in a number of fields, such as meteorology, oceanography, electric circuit design, etc. It seeks to find a subspace to approximate the dynamics of a usually large dimension, nonlinear dynamic system, for which the computational cost and storage requirements are demanding. The necessity of reduced order modeling is even more obvious when it comes to the inverse problem of a large-scale, nonlinear dynamic problem. Reduced order modeling began to gain some interest in computational finance recently, such as [58], [20], [56], [59]. We applied the POD reduced order modeling to the Heston model for pricing European options. Being capable of computing the option prices using a much simpler model, the POD reduced model has a potential to speed up the calibration for stochastic volatility models.

This thesis can be broadly divided into three parts: an introductory part, introduction and calibration of local volatility models, and introduction of POD reduced order modeling with application to stochastic volatility models. Specifically, it is organized in the following manner. Chapter 1 consists of the introduction. In Chapter 2, the local volatility model including the constant elasticity of variance (CEV) models and the Dupire equation, which will be used in Chapter 3, are introduced. In Chapter 3, we address in detail a non parametric approach for the calibration of a local volatility surface using a second order Tikhonov regularization. In this chapter, we cover several thorny issues of the non-parametric calibration of the local volatility surface, such as using automatic differentiation tools to derive the adjoint code required to compute the gradient of the cost function with respect to the volatility surface, ill-posedness of the calibration problem, and proposing a method to select the Tikhonov regularization parameter. In Chapter 4, we present stochastic volatility models including the popular Heston volatility model and SABR volatility model. In Chapter 5, POD reduced order modeling is introduced as an efficient way of computing the option prices. Finally, the thesis concludes with a summary and conclusions chapter.

CHAPTER 2

LOCAL VOLATILITY MODEL

The local volatility model assumes that the price S of an asset follows a general diffusion process:

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

where μ_t is the risk neutral drift rate, W_t is a standard Brownian motion process, and the local volatility σ is a deterministic function that may depend on both the asset price S and the time t. Typically, $\mu_t = r_t - q_t$ with r_t being the risk-free continuously compounded interest rate and q_t being the continuous dividend yield of the asset. Without losing generality, we'll assume q_t is zero and r_t is a constant unless they are specified otherwise. Due to its simplicity and the availability of analytical solutions for certain cases, the local volatility model is still very popular in the financial industry. In the following, we'll introduce some of the most popular local volatility models. Due to the existence of an analytical formula for European options prices, they will also serve as ideal cases for testing our proposed calibration methods.

2.1 Constant elasticity of variance (CEV) model

The CEV model assumes the form of:

$$\frac{dS}{S} = (r-q)dt + \kappa S^{p-1} dW_t \tag{2.2}$$

where κ and p are constant coefficients. When $p = 0$, (2.2) corresponds to the Ornstein-Uhlenbeck process or is referred as the absolute CEV model; when $p = 1$, it is the Black-Scholes model; when $p = 0.5$, it corresponds to a square root process; when $p = 2$, it is a special case of quadratic volatility model. When $0 < p < 1$, the volatility decreases as the asset price S increases. This creates a probability distribution function (pdf) of the asset return with a heavy left tail and less heavy right tail, for example, in the equity market. When $p > 1$, the volatility increases as S increases. This creates a probability distribution of the asset return with a heavy right tail and less heavy left tail, which corresponds to a volatility smile where the implied volatility is an increasing function of the strike price. This type of volatility smile is sometimes observed for options on futures [42].

The CEV model acquired its name from the fact that the elasticity of the variance of the rate of return of S with respect to S is a constant. The elasticity of variable y with respect to variable x is defined as :

$$E_{y,x} = \left| \frac{dy}{y} \Big/ \frac{dx}{x} \right|$$

The elasticity of the variance of the rate of return of S with respect to S is

$$\begin{aligned}
\frac{\mathrm{dVar}(dS/S)}{\mathrm{Var}(dS/S)} \Big/ \frac{dS}{S} &= \frac{\mathrm{dVar}(dS/S)}{dS} \Big/ \frac{\mathrm{Var}(dS/S)}{S} \\
&= \kappa^2(2p-2)S^{2p-3}dt \Big/ \kappa^2 \frac{S^{2p-2}dt}{S} \\
&= 2p-2
\end{aligned}$$

where we have used

$$\mathrm{Var}(dS/S) = \kappa^2 S^{2p-2}dt$$

The constant elasticity of variance means that the ratio of the relative change of the variance of the rate of return of S and the relative change of S is a constant, i.e., for every 10% change of S, the change of the variance of the rate of return will also be 10%. In the CEV model, σ changes monotonically with respect to S. So (2.2) is most appropriate for markets where the volatility smile is close to a monotonic function of strike K. For a detailed discussion of the properties of CEV models, please see [5] .

In this thesis, we'll use the analytical solution of the cases $p = 0$, $p = 1$, and $p = 2$ as test cases respectively for the proposed calibration method discussed in the following chapters. So we list their analytical solutions here.

When $0 < p < 1$, the price of an European call option with strike K and maturity T under the CEV model is:

$$C = S_0 e^{-qT}\left[1 - \chi^2(a, b+2, c)\right] - Ke^{-qT}\chi^2(c, b, a) \tag{2.3}$$

when $p > 1$, the pricing formula is:

$$C = S_0 e^{-qT}\left[1 - \chi^2(a, -b, c)\right] - Ke^{-qT}\chi^2(c, 2-b, a) \tag{2.4}$$

where

$$\begin{aligned}
a &= \frac{\left[Ke^{-(r-q)T}\right]^{2(1-p)}}{(1-p)^2\nu} \\
b &= \frac{1}{1-p} \\
c &= \frac{S_0^{2(1-p)}}{(1-p)^2\nu} \\
\nu &= \frac{\kappa^2}{2(r-q)(p-1)}\left[e^{2(r-q)(p-1)T} - 1\right]
\end{aligned}$$

and $\chi^2(z;k,v)$ is the cumulative distribution function for a noncentral Chi-squared distribution with noncentrality parameter v and k degrees of freedom evaluated at z.

When $p = 0$, its pricing formula is :

$$C = (S_0 e^{-qT} - Ke^{-rT})N(y_1) + (S_0 e^{-qT} + Ke^{-rT})N(y_2) + \nu\left[n(y_1) - n(y_2)\right]$$

where

$$\begin{aligned}
\nu &= \kappa\left(\frac{e^{-2qT} - e^{-2rT}}{2(r-q)}\right)^{\frac{1}{2}} \\
y_1 &= \frac{S_0 e^{-qT} - Ke^{-rT}}{\nu} \\
y_2 &= \frac{-S_0 e^{-qT} - Ke^{-rT}}{\nu}
\end{aligned}$$

and $N(\cdot)$ is the cumulative distribution function of a standard normal variable, $n(\cdot)$ is the probability density function of a standard normal distribution. For a detailed derivation of the above formulas, see [23] and [61].

2.2 Quadratic volatility model

The CEV model is limited in that it assumes the volatility smile is a monotonic function with respect to strikes K. In reality it is not uncommon to see convex volatility smiles. In order to introduce a volatility smile that's easy to fit with arbitrary convexity, one of the most popular deterministic approach is to use a quadratic volatility model. The quadratic volatility model assumes the form:

$$dS = (r-q)dt + (\alpha + \beta S + \gamma S^2)dW_t \tag{2.5}$$

For simplicity and without loss of generality, we will consider the process:

$$dS = (\alpha + \beta S + \gamma S^2)dW_t \tag{2.6}$$

Equation (2.6) can be derived from (2.5) through a change of measure. The behavior of stochastic process of (2.6) depends strongly on the root configuration of the quadratic equation $\alpha + \beta S + \gamma S^2 = 0$. For example, if the two roots are l and u with S_0 is sandwiched between them, then S will be bounded between l and u. As S gets close to either l and u, the volatility of (2.6) will approach zero, thus S will stop changing.

The analytical formula of European put option price under the quadratic volatility model (2.6) is summarized in [6]. When S_0 lies to the left or right of the two solutions l and u, the local volatility is still a monotonic function with respect to S. But the reason we choose the quadratic volatility model as a test case is its ability in generating convexed volatility smiles. For this reason, we only adopt the case from [6] when there is no real roots for equation $\alpha + \beta S + \gamma S^2 = 0$, in which case the volatility smile is a convex curve. For completeness, we list the analytical solution for European put prices from [6] in the following. We only list the solution computed using the method of images.

(2.6) can be reformulated as:

$$dS = b\left(1 + \left(\frac{S-a}{b}\right)^2\right) dW_t \tag{2.7}$$

Then the price of a European put option under the (2.7) is

$$P = \sqrt{bA(S_0)}e^{\frac{1}{2}T}e_1 + \sqrt{\frac{A(S_0)}{A(S_L)}}(K - S_L)e_2 \tag{2.8}$$

where

$$\begin{aligned}
A(x) &= b\left[1 + ((x-a)/b)^2\right] \\
e_1 &= \frac{2}{m_U - m_L}\sum_{n=1}^{\infty} e^{-\alpha_n T}\sin(-a_n)(\tilde{K}I_n^{(c)} - I_n^{(s)}) \\
e_2 &= \frac{m_U}{m_U - m_L} - \frac{1}{(m_U - m_L)^2}\sum_{n=1}^{\infty} n\pi sin(-a_n)\frac{e^{-(\alpha_n - \frac{1}{2})T}}{\alpha_n - \frac{1}{2}} \\
\alpha_n &= \frac{n^2\pi^2}{2(m_U - m_L)^2} \\
a_n &= n\pi\frac{m_L}{m_U - m_L} \\
m &= z - z_0 \\
m_U &= z_U - z_0 \\
m_L &= z_L - z_0 \\
z &= \arctan\frac{S-a}{b} \\
z_0 &= \arctan\frac{S_0 - a}{b} \\
I_n^{(c)} &= \frac{m_L}{2}\left[\frac{\cos(c_n^-)}{m_L - a_n} - \frac{\cos(c_n^+)}{m_L + a_n} + 2\frac{\cos(z_L)a_n}{a_n^2 - m_L^2}\right] \\
I_n^{(s)} &= \frac{m_L}{2}\left[\frac{\sin(c_n^-)}{m_L - a_n} - \frac{\sin(c_n^+)}{m_L + a_n} + 2\frac{\sin(z_L)a_n}{a_n^2 - m_L^2}\right] \\
c_n^{\pm} &= b_n^{\pm}k \mp a_n + z_0 \\
b_n^{\pm} &= 1 \pm a_n/m_L \\
k &= \arctan(\tilde{K}) - z_0 \\
\tilde{K} &= \frac{K-a}{b}
\end{aligned}$$

2.3 Dupire equation

Dupire in his seminal work [27] established that the local volatility function can be uniquely derived from European option prices given the existence of European options with

all strikes and maturities. Since there are only a limited number of options available in the market, the analytical solution can not be directly used to compute the volatility surface. However, the Dupire PDE provides a way to compute all the prices of European options with different strikes and maturities for a fixed S_0 by solving the PDE only once. It suits well our purpose of calibration of the local volatility model in that we do not have to solve the Black-Scholes equation multiple times. We include the derivation of Dupire equation as follows.

For a vanilla European call option with strike K and maturity T, its price C can be expressed as:

$$C(K,T) = e^{-rT}\int_0^\infty (S_T-K)^+\phi(S_T,T;S_0,t_0)dS_T = e^{-rT}\int_K^\infty (S_T-K)\phi(S_T,T;S_0,t_0)dS_T \tag{2.9}$$

where ϕ is the risk-neutral transition probability density function of the final spot price S_T at time T given the current spot price S_0 at t_0.

The first and second partial derivatives of C with respect to K are given by

$$\frac{\partial C}{\partial K} = -e^{-rT}\int_K^\infty \phi(S_T,T;S_0,t_0)dS_T \tag{2.10}$$

$$\frac{\partial^2 C}{\partial K^2} = e^{-rT}\phi(K,T;S_0,t_0) \tag{2.11}$$

The partial derivative of C with respect to T is given by

$$\frac{\partial C}{\partial T} = e^{-rT}\int_K^\infty (S_T-K)\left(-r\phi(S_T,T;S_0,t_0) + \frac{\partial\phi(S_T,T;S_0,t_0)}{\partial T}\right)dS_T \tag{2.12}$$

The transition probability density ϕ for the stochastic model (2.1) satisfies the following Kolmogorov forward equation, which is also known as the Fokker-Planck equation.

$$\frac{\partial\phi(S_T,T;S_0,t_0)}{\partial T} = -\frac{\partial}{\partial S_T}\left((r-q)S_T\phi(S_T,T;S_0,t_0)\right) + \frac{1}{2}\frac{\partial^2}{\partial S_T^2}(\sigma^2(S_T,T)S_T^2\phi(S_T,T;S_0,t_0)) \tag{2.13}$$

In contrast to the Kolmogorov backward equation, in which T and S_T are held constant and the variables are S_0 and t_0, the Kolmogorov forward equation holds S_0 and t_0 as constant and uses S_T and T as the variables. The variables S_T and T are called forward variables. Please note the function $\sigma(\cdot,\cdot)$ in equation (2.13) has the same function form as the function $\sigma(\cdot,\cdot)$ in equation (2.1).

Substituting equation (2.13) into the right hand side of equation (2.12) and integrating by parts, we obtain three integrals:

$$e^{-rT}\int_K^\infty -(S_T-K)r\phi(S_T,T;S_0,t_0)dS_T = -rC$$

$$
\begin{aligned}
& e^{-rT}\int_K^\infty -(S_T-K)\frac{\partial}{\partial S_T}\left((r-q)S_T\phi(S_T,T;S_0,t_0)\right)dS_T \\
=& -(r-q)e^{-rT}\int_K^\infty (S_T-K)\frac{\partial}{\partial S_T}\left(S_T\phi(S_T,T;S_0,t_0)\right)dS_T \\
=& -(r-q)e^{-rT}\left[(S_T-K)S_T\phi(S_T,T;S_0,t_0)|_{S_T=K}^{S_T=\infty}-\int_K^\infty S_T\phi(S_T,T;S_0,t_0)dS_T\right] \\
=&(r-q)e^{-rT}\int_K^\infty S_T\phi(S_T,T;S_0,t_0)dS_T \\
=&(r-q)(C-K\frac{\partial C}{\partial K})
\end{aligned}
$$

where we have used equation (2.10) and the following assumption (2.14).

$$\lim_{S\to\infty}(S_T-K)S_T\phi(S,T;S_0,t_0)=0 \tag{2.14}$$

$$
\begin{aligned}
& e^{-rT}\int_K^\infty (S_T-K)\frac{1}{2}\frac{\partial^2}{\partial S_T^2}(\sigma^2(S_T,T)S_T^2\phi(S_T,T;S_0,t_0))dS_T \\
=&\frac{1}{2}e^{-rT}\int_K^\infty (S_T-K)\frac{\partial^2}{\partial S_T^2}(\sigma^2(S_T,T)S_T^2\phi(S_T,T;S_0,t_0))dS_T \\
=&\frac{1}{2}e^{-rT}\left[(S_T-K)\frac{\partial}{\partial S_T}(\sigma^2(S_T,T)S_T^2\phi(S_T,T;S_0,t_0))|_{S_T=K}^{S_T=\infty}-\int_K^\infty \frac{\partial}{\partial S_T}(\sigma^2(S_T,T)S_T^2\phi(S_T,T;S_0,t_0))dS_T\right] \\
=&\frac{1}{2}e^{-rT}\left[0-\sigma^2(S_T,T)S_T^2\phi(S_T,T;S_0,t_0)|_{S_T=K}^{S_T=\infty}\right] \\
=&\frac{1}{2}e^{-rT}\sigma^2(K,T)K^2\phi(K,T;S_0,t_0) \\
=&\frac{1}{2}\sigma^2(K,T)K^2\frac{\partial^2 C}{\partial K^2}
\end{aligned}
$$

where we have used equation (2.11), assumptions (2.14) and (2.15).

$$\lim_{S_T\to\infty}\frac{\partial}{\partial S}\sigma^2(S_T,T)S_T^2\phi(S_T,T;S_0,t_0)=0 \tag{2.15}$$

Combining the above three integrals with the left hand side of equation (2.12), we obtain

$$\frac{\partial C}{\partial T}=\frac{1}{2}\sigma^2(K,T)K^2\frac{\partial^2 C}{\partial K^2}-(r-q)K\frac{\partial C}{\partial K}-qC \tag{2.16}$$

This is the famous Dupire equation, which gives an analytical solution for the local volatility surface σ given the existence of European options across all strikes and maturities:

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

When the drift term in (2.1) vanishes due to a change of measure, σ assumes a compact form given by

$$\sigma^2(K,T) = \frac{\frac{\partial C}{\partial T}}{\frac{1}{2}K^2\frac{\partial^2 C}{\partial K^2}}$$

2.4 Dynamics of local volatility model

While the local volatility model is easier to apply for computing option prices and is still a popular model in the financial industry, it has a major drawback due to its non-stationary dynamics, which is usually not in agreement with the actual market. This issue can be illustrated by considering the following simple example.

Consider a local volatility model assuming the following form:

$$\sigma(S) = (c + (S - S_0)^2) \tag{2.17}$$

where $c > 0$ is a constant. According to (2.1), the asset follows the process:

$$dS = \mu S dt + S(c + (S - S_0)^2)dW_t$$

For this local volatility model, when $S = S_0$ at $t = 0$, the volatility surface has the shape of a parabola, or a "smile". When $t > 0$, if $S_t > S_0$, then the volatility surface at that moment will be a monotonic increasing function with respect to S. On the contrary, if $S_t < S_0$, then at time t, the volatility surface will decrease monotonically with respect to S. So the shape of the volatility surface will depend strongly on the asset price S, which is often in conflict with the market behavior. In order to overcome this issue, stochastic volatility models are proposed as an alternative, which will be discussed in the second part of this thesis.

CHAPTER 3

CALIBRATION OF LOCAL VOLATILITY MODELS

3.1 Description of the calibration problem

For consistency, the local volatility model is defined as in [48]. The local volatility model assumes that the price s of an underlying follows a general diffusion process:

$$\frac{ds}{s} = (r - q)dt + \sigma(s, t)dW_t \tag{3.1}$$

where r is the risk-free continuously compounded interest rate, q is the continuous dividend yield of the asset, W_t is a standard Brownian motion process, and the local volatility σ is a deterministic function that may depend on both the asset price s and the time t. r and q are assumed to be constant in this thesis. Let $V(s_0, 0, K, T, \sigma)$ denote the theoretical price of an European option with strike K and maturity T at reference time 0 for an asset with spot price s_0 following the process in (3.1). Let T_1,..., T_N be the set of maturities of the European options available in the market for the asset. For each maturity T_i, the strikes available range from K_{i1}, ..., K_{iM_i}.

The calibration of the local volatility surface to the market is to find a local volatility surface $\sigma(s, t)$ such that the theoretical option price computed using this volatility surface lies between the corresponding bid and ask prices 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 $j = 1, \ldots, M_i$. V_{ij}^a and V_{ij}^b denote the bid and ask prices respectively for an option with maturity T_i and strike K_{ij} at the time $t = 0$.

This problem is usually solved by solving the following optimization problem:

$$\min_{0<\sigma\leq 1} \quad G(\sigma) = \sum_{i=1}^{N} \sum_{j=1}^{M_i} [(V(s_0, 0, K_{ij}, T_i, \sigma) - \overline{V}_{ij})w_{ij}]^2 \tag{3.2}$$

where $\overline{V}_{ij} = (V_{ij}^b + V_{ij}^a)/2$ is the mean of the bid and ask prices. w_{ij} is a scaling factor to reflect the relative importance of different options. This scaling factor could play an important role in the calibration problem especially when option prices contain "noises" since a small amount of relative noise from deep-in-the-money option prices could easily mask the

"signals" implied by out-of-money option prices. One way of choosing the above weights is based on a liquidity argument. The weights are computed as inversely proportional to the square of the bid-ask spreads to give more weight to the liquid options. [21] suggested computing the weights as the inverse of square of the Black-Scholes vegas evaluated at the implied volatilities of the market option prices. They showed that it is approximately equivalent to minimizing the differences of the Black-Scholes implied volatilities between the market prices and the model computed prices. The weighting scheme is derived as an efficient way of generating approximation errors proportional to the bid-ask spreads and it also works for the case when the bid-ask spreads are not available. In this thesis, we set w_{ij} to be one for all cases except for the case where we deliberately add enough artificial noise to show the significance of the weighting scheme, when the weighting scheme in [21] is used. The reason we set $w_{ij} = 1$ is that either we know the prices are true prices or $w_{ij} = 1$ was used in other papers for the same market data used in the present thesis. We retain the same weighting scheme in order to be able to compare our result with the previous studies.

When there is more than one maturity, the theoretical option price V can be efficiently computed by solving the Dupire equation (3.3). The Dupire equation establishes the option prices as a function of strike K and maturity T for a fixed asset price s_0 at reference time $t = 0$. By solving the following Dupire equation (3.3) just once, we can obtain the theoretical prices for all the European options of the same underlying asset at s_0:

$$\frac{\partial V}{\partial T} - \frac{1}{2}K^2\sigma^2(K,T)\frac{\partial^2 V}{\partial^2 K} + (r-q)K\frac{\partial V}{\partial K} + qV = 0 \tag{3.3}$$

Notice in (3.3), σ is a function of K and T instead of s and t. We just point out that the function form of σ is not changed, and that the K, T, s, or t are all just dummy variables, details of which are in [27].

Before attempting to solve the optimization problem in (3.2), we want to point out some aspects of the problem that render it complicated. The optimization problem in (3.2) is a large scale nonlinear under-determined inverse problem. (a) The number of parameters to estimate is very large. To estimate the volatility surface, we want to find the volatility at each grid point. While similar to other archival material as well as we find that only the section of volatility surface near the money can be estimated from market prices, the number of parameters to estimate is still quite large. (b) The Dupire or Black-Scholes equation is a nonlinear operator in σ or σ^2. (c) The total number of options available is usually much less than the number of parameters to be estimated. Thus it is also an under-determined problem. (d) As for most inverse problems, it is ill-posed in the sense that small changes in the option prices may lead to big changes in the volatility surface. When noises are included in the option prices, which is usually the case in reality, the reconstructed volatility surface tends to be unstable. To resolve the issues (c) and (d), we propose use of a second order Tikhonov regularization, details of which will be introduced in later sections.

To deal with the issues of (a) and (b), a gradient-based optimization routine is usually used. Most papers [13, 14, 47, 46, 25, 28, 39, 1, 65] derive the gradient of cost function G in (3.2) with respect to σ by solving the adjoint model of the Dupire model. By using an adjoint approach, the gradient can be computed by integrating the adjoint model backwards in time just once. In all of these papers, the adjoint model of the Dupire model was derived first and then solved numerically. This way of using the adjoint belongs to the *differentiate-*

then-discretize approach, i.e., one differentiates the partial differential equations (along with initial and boundary conditions), takes the adjoint of the results, and then discretizes the continuous system of adjoint equations.

There is an alternative way of deriving the adjoint, namely the *discretize-then-differentiate* approach, see for example [30], in which one first discretizes the original model and then obtains a system of adjoint equations of the discretized model. Both approaches yield a set of discrete equations for the adjoint variables. But the discretization and differentiation operators do not commute. [36] found that the gradient derived using the *differentiate-then-discretize* approach can be inconsistent with the true gradient. The inconsistency can result in a serious difficulty for minimizing the cost function. In this thesis, we will adopt the *discretize-then-differentiate* approach: we first discretize the Dupire model using a finite difference method and then differentiate the discrete version of the Dupire model to obtain its adjoint model. In the step of differentiation of the discrete Dupire model, automatic differentiation in reverse mode can be utilized to generate the discrete adjoint model. In the following section, we set up the derivation of the gradient in a general framework so that the same technique can be used for calibration of other models or with respect to exotic options.

3.2 Gradient of the cost function

Algorithmic differentiation has already been employed in the quantitative finance field. For example, [32], [15] used it to speed up the calculation of Greeks. It has long been established in other domains of research such as computational fluid dynamics that the gradient of a cost function in the form of (3.2) can also be computed by using automatic differentiation, such as displayed in [31]. We'll just list some results for the sake of completeness of this thesis. For a more general formulation, see [17], [53].

Let M be a general model such that

$$\frac{\partial \boldsymbol{X}}{\partial t} = M(\boldsymbol{X}, \boldsymbol{\alpha}) \tag{3.4}$$

where $\boldsymbol{X} \in \mathbf{R}^m$ is a vector containing the state variables of the model, $\boldsymbol{\alpha} \in \mathbf{R}^n$ denotes the model parameters. A typical cost function used for parameter calibration assumes the form

$$J(\boldsymbol{X}, \boldsymbol{\alpha}) = \frac{1}{2} \int_{t_0}^{t_\tau} \langle \boldsymbol{W}(\boldsymbol{X} - \boldsymbol{X}^{obs}), \boldsymbol{W}(\boldsymbol{X} - \boldsymbol{X}^{obs}) \rangle dt \tag{3.5}$$

where $[t_0, t_\tau]$ is the observation window, $\boldsymbol{W}$ is a weighting factor to reflect the relative importance of each observation. $\boldsymbol{X}^{obs}$ is the observation vector. It can be shown [50] that the gradient of the cost function with respect to the parameters $\boldsymbol{\alpha}$ is given by

$$\nabla_{\boldsymbol{\alpha}} J = \int_{t_0}^{t_\tau} (-[\frac{\partial M}{\partial \boldsymbol{\alpha}}]^T \boldsymbol{P}) dt \tag{3.6}$$

where $\boldsymbol{P} \in \mathbf{R}^m$ is adjoint variable of the state variables and is governed by the following system:

$$\begin{cases} \frac{\partial \boldsymbol{P}}{\partial t} + [\frac{\partial M}{\partial \boldsymbol{X}}]^T \boldsymbol{P} = \boldsymbol{W}(\boldsymbol{X} - \boldsymbol{X}^{obs}) \\ \boldsymbol{P}(t_\tau) = 0 \end{cases} \tag{3.7}$$

where $[\frac{\partial M}{\partial \boldsymbol{X}}]^Tand[\frac{\partial M}{\partial \boldsymbol{\alpha}}]^T$ represent the transpose of the Jacobian matrix of the model with respect to state variables and model parameters respectively in the discrete case. When $\boldsymbol{P}$ is known by integrating backward in time the system described by (3.8), all the components of the gradient J with respect to $\boldsymbol{\alpha}$ can be computed using equation (3.6).

Equations (3.6) and (3.8) show that we can compute the gradient of cost function $J(\boldsymbol{\alpha})$ by running the adjoint model only once. [34] shows that the required numerical operations will require only $2-5$ times the CPU time required for the forward cost function.

In this thesis, $[\frac{\partial M}{\partial \boldsymbol{X}}]^Tand[\frac{\partial M}{\partial \boldsymbol{\alpha}}]^T$ are obtained using automatic differentiation tools. A complete detailed discussion of the rationale of automatic differentiation is beyond the scope of this thesis. In the following, we will use a simple example to briefly explain how to implement automatic differentiation. See [35] for a detailed discussion of automatic differentiation.[1]

3.2.1 Derivation of the adjoint code

The models $[\frac{\partial M}{\partial \boldsymbol{X}}]^Tor[\frac{\partial M}{\partial \boldsymbol{\alpha}}]^T$ are referred to as adjoint model. In this section, we provide a simple example to demonstrate the concept of the adjoint model, the meaning of adjoint variables and how to code an adjoint model in straightforward fashion. It is not intended to cover extensively the general principles behind the reverse mode of automatic differentiation.

Let us consider a model given by a function $F = X * sin(Y^2)$, where we assume X,Y, and F all depend on time t. To get the adjoint of this model $F(X, Y)$, we first generate its Jacobian matrix (or sometimes called 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, $n-1$ denote subsequent time levels. Given an 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

$$\begin{pmatrix} ADF \\ ADY \\ ADX \end{pmatrix}^{n-1} = \begin{pmatrix} 0 & 0 & 0 \\ cos((Y^{n-1})^2)2Y^{n-1} & 1 & 0 \\ sin((Y^{n-1})^2) & 0 & 1 \end{pmatrix} \begin{pmatrix} ADF \\ ADY \\ ADX \end{pmatrix}^n \tag{3.8}$$

ADF, ADY and ADX represents the adjoint variable of F, Y and X respectively. (3.8) can be written as:

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

[1]There are several free automatic differentiation tools available, whose details are to be found on the website www.autodif.org. Automatic differentiation can help speed up the process of developing the numerical code of an adjoint model especially for complicated models. However, some debugging and verification is usually necessary for checking the validity of the code generated by the free automatic differentiation tools.

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.

This adjoint model computes the derivative of $\frac{\partial F}{\partial X}$, $\frac{\partial F}{\partial Y}$ (or sensitivity) in reverse direction. When integrated backward in time, this adjoint model would enable us the find out the source area that leads to the final perturbation of ADF. Classical methods are slow at computing the partial derivatives of a function with respect to many inputs, as is needed for gradient-based optimization algorithms. Automatic differentiation solves all of these problems.

3.2.2 Gradient test

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 adjoint identity, see Navon et al. [54]

$$< \mathbf{A}\delta\boldsymbol{X}, \mathbf{A}\delta\boldsymbol{X} >=< \delta\boldsymbol{X}, \mathbf{A}^T(\mathbf{A}\delta\boldsymbol{X}) > \quad \text{for any } \delta\boldsymbol{X} \tag{3.9}$$

where $\delta\boldsymbol{X}$ is an arbitrary perturbation vector. It is also the input of code $\mathbf{A}$. $\mathbf{A}$ represents the tangent linear code or a segment of it, say a subroutine, a do loop or even a single statement. $\mathbf{A}^T$ is the adjoint of $\mathbf{A}$. If (3.9) holds within machine accuracy, it can be said that the adjoint is correct versus the tangent linear code.

1. Test of the accuracy of the Tangent Linear Model (TLM)
   To use (3.9) test the correctness of adjoint model, the tangent linear model $\mathbf{A}$ is used. Thus first we need to test the correctness of tangent linear model $\mathbf{A}$. 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 $\mathbf{A}$, we use the fact that $\mathbf{A}$ is a linearization of the model F:

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

   where $\delta\boldsymbol{X}$ is an arbitrary perturbation around $\boldsymbol{X}$, α is a nonzero scalar.

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

   $$r = \frac{F(\boldsymbol{X} + \alpha * \delta\boldsymbol{X}) - F(\boldsymbol{X})}{A(\alpha * \delta\boldsymbol{X})} = 1 + O(\alpha) \tag{3.10}$$

   will approach one 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 adjoint identity:
   $$< \mathbf{A}\delta\boldsymbol{X}, \mathbf{A}\delta\boldsymbol{X} >=< \delta\boldsymbol{X}, \mathbf{A}^T(\mathbf{A}\delta\boldsymbol{X}) > \quad \text{for any } \boldsymbol{X} \tag{3.11}$$
   where $\delta\boldsymbol{X}$ is a small perturbation around $\boldsymbol{X}$. If (3.11) 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.

   Suppose the initial X is perturbed by a small amount $\alpha\boldsymbol{h}$, where α is a small scalar and $\boldsymbol{h}$ is a vector of unit length. According to Taylor expansion, we get the cost function:

   $J(\boldsymbol{X} + \alpha\boldsymbol{h}) = J(\boldsymbol{X}) + \alpha\boldsymbol{h}^T \bigtriangledown J(\boldsymbol{X}) + O(\alpha^2)$

   We can define a function of α as:

   $$\Phi(\alpha) = \frac{J(\boldsymbol{X} + \alpha\boldsymbol{h}) - J(\boldsymbol{X})}{\alpha\boldsymbol{h}^T \bigtriangledown J(\boldsymbol{X})} = 1 + O(\alpha) \tag{3.12}$$

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

3.3 Tikhonov regularization

3.3.1 The second order Tikhonov regularization

To deal with the ill-posedness of the calibration problem, regularization is usually required. Tikhonov regularization is one of the most popular regularization methods for ill-posed inverse problems [63]. In addition to minimizing the cost function, it seeks to minimize some measure of the solution, for example, the size of the solution or the norm of the first and second derivative of the solution. It usually assumes the following form.

$$J(\sigma) = G(\sigma) + \lambda \parallel \mathbf{L}\sigma \parallel_2^2 \tag{3.13}$$

where $G(\sigma)$ is as defined in (3.2) and λ is the regularization parameter. $\mathbf{L}$ is an operator on σ. When $\mathbf{L}$ is the identity matrix, it is called the zeroth order Tikhonov regularization. When $\mathbf{L}$ is an operator approximating the first or second derivative of σ with respect to s and t, it is called the first or second order Tikhonov regularization respectively. As mentioned in the introduction, most papers on the calibration of local volatility surfaces used the following first order Tikhonov regularization, see [48], [13, 14]; [12]; [47, 46]; [25]; [28]; [39]; [1] and [65].

$$J(\sigma) = G(\sigma) + \lambda(\| \frac{\partial \sigma}{\partial s} \|_2^2 + \| \frac{\partial \sigma}{\partial t} \|_2^2) \tag{3.14}$$

However, the volatility surface generated by the first order Tikhonov regularization is usually rough. Assuming the volatility surface is smooth, we propose to use a second order Tikhonov regularization: the regularization term $\| \mathbf{L}\sigma \|_2^2$ would be a measure of the norm of the second derivatives of σ with respect to s and t. Since the calibration problem is under-determined, a second order Tikhonov regularization also imposes more constraints on the calibration problem than the first order Tikhonov regularization. Our regularization term $\| \mathbf{L}\sigma \|_2^2$ is an approximation of the following :

$$\|\frac{\partial^2 \sigma}{\partial s^2}\|_2^2 + \|\frac{\partial^2 \sigma}{\partial t^2}\|_2^2 + \|\frac{\partial^2 \sigma}{\partial t \partial s}\|_2^2 \tag{3.15}$$

If σ were just an one dimensional vector of size n, the exact form of $\mathbf{L}$ could be written as follows:

$$\begin{bmatrix} 1 & -2 & 1 & & & & 0 \\ & 1 & -2 & 1 & & & \\ & & & \cdots & & & \\ & & & & 1 & -2 & 1 \\ & 0 & & & & 1 & -2 & 1 \end{bmatrix}_{(n-2)\times n} \tag{3.16}$$

At each grid point, the second derivative of σ is approximated by a second order accurate finite difference scheme up to a constant. Due to the fact that σ is a two dimensional surface rather than a vector, the explicit matrix form of operator $\mathbf{L}$ could not be easily written down since we have to approximate three second derivatives at each grid point. For our computation, we actually do not need the explicit form of $\mathbf{L}$, since we just need the term $\| \mathbf{L}\sigma \|_2^2$. The following simple algorithm (1) describes the computation of $\| \mathbf{L}\sigma \|_2^2$ and the update of gradient of J with respect to σ when the regularization part is added. In the algorithm (1), $\partial^2\sigma/\partial s^2$, $\partial^2\sigma/\partial t^2$, $\partial^2\sigma/\partial t\partial s$ are all approximated by a second order accurate finite difference scheme up to a constant.

Algorithm 1 Compute $\lambda(\|\frac{\partial^2\sigma}{\partial s^2}\|_2^2 + \|\frac{\partial^2\sigma}{\partial t^2}\|_2^2 + \|\frac{\partial^2\sigma}{\partial t\partial s}\|_2^2)$ and update gradient

```
//Compute ||∂²σ/∂s²||²
norm1 = 0.0
for i = 1 to n_t do
  for j = a + 1 to b − 1 do
    temp = σ_{j+1,i} + σ_{j−1,i} − 2σ_{j,i}
    norm1 = norm1 + temp * temp
    g(j + 1, i) = g(j + 1, i) + temp * alpha
    g(j − 1, i) = g(j − 1, i) + temp * alpha
    g(j, i) = g(j, i) − 2 * temp * alpha
  end for
end for
// Compute ||∂²σ/∂t²||²
norm2 = 0.0
for i = a to b do
  for j = 2 to n_t − 1 do
    temp = σ_{i,j+1} − 2σ_{i,j} + σ_{i,j−1}
    norm2 = norm2 + temp * temp
    g(i, j + 1) = g(i, j + 1) + temp * alpha
    g(i, j) = g(i, j) − 2.0 * temp * alpha
    g(i, j − 1) = g(i, j − 1) + temp * alpha
  end for
end for
// Compute ||∂²σ/∂t∂s||²₂
norm3 = 0.0
for i = a to b do
  for j = 2 to n_t − 1 do
    temp = σ_{i+1,j+1} + σ_{i−1,j−1} − σ_{i+1,j−1} − σ_{i−1,j+1}
    norm3 = norm3 + temp * temp
    g(i + 1, j + 1) = g(i + 1, j + 1) + temp * alpha
    g(i − 1, j + 1) = g(i − 1, j + 1) − temp * alpha
    g(i + 1, j − 1) = g(i + 1, j − 1) − temp * alpha
    g(i − 1, j − 1) = g(i − 1, j − 1) + temp * alpha
  end for
end for
Lsigma = norm1 + norm2 + norm3
f = f+ λ*Lsigma
```

Since only the section of the volatility surface that is near the money is sensitive to option prices and can be recovered, the regularization is just applied to the part of volatility surface $\sigma(s,t)$ for which the ratio between s and spot s_0 lies within the interval [0.8, 1.2].

For the regions of the volatility surface outside the interval defined above, no regularization is performed. Since the components of the gradient vector corresponding to volatilities at these regions are zero, the volatilities at these regions cannot be updated by a gradient

based optimization routine and are thus kept constant throughout the optimization. The constant is the initial guess of the local volatility surface.

In the algorithm (1), $\sigma(n_x, n_t)$ is a two dimensional matrix representing $\sigma(s,t)$ where n_x, n_t are the number of intervals along the s and t direction, respectively. a and b are the indices that correspond to $0.8s_0$ and $1.2s_0$ along the s direction. f and g are inputted respectively as the cost function and gradient before any regularization, and then returned as the regularized cost function and the gradient of the regularized cost function with respect to σ, respectively.

The calibration problem now assumes the form of a constrained minimization problem:

$$\min_{0<\sigma(s,t)\leq 1} \quad J(\sigma) = G(\sigma) + \lambda \parallel \mathbf{L}\sigma \parallel_2^2 \tag{3.17}$$

3.3.2 Strategy for selecting the Tikhonov regularization parameter λ

A Tikhonov regularization solution of an inverse problem depends critically on a suitable selection of the regularization parameter λ. How to suitably choose a regularization parameter is still at the stage of active research. For linear inverse problems, λ is usually selected by either the L-curve method or by generalized cross validation theory, see [38], [9], [22] and [3]. For nonlinear inverse problems, the L-curve method is still applicable to select the optimal λ. The L-curve method plots the cost function $G(\sigma)$ with respect to $\parallel \mathbf{L}\sigma \parallel_2^2$. This plot usually assumes an L shape. The corner of the L-curve is considered the best compromise point between the size of the solution and the magnitude of the cost function. The λ at the corner of L-curve is considered the optimal regularization parameter λ. However, we found from our numerical tests that we cannot generate an L-shaped curve for this nonlinear inverse problem since λ chosen close to any sort-of L-corner generates a volatility surface far away from the true volatility surface.

As many nonlinear problems are solved iteratively by solving a linear problem at each iteration, we will adopt an iterative regularization strategy to solve the nonlinear inverse problem, in which a suitable regularization parameter λ is selected at each iteration. By linearizing the problem at each iteration, some of the analysis for linear inverse problems can be applied.

To determine how to select a suitable regularization parameter at each iteration, we carry out the following analysis to see how ill-posedness occurs.

We are actually solving for a vector $\boldsymbol{X}$ from

$$\mathbf{F}\boldsymbol{X} = \tilde{\boldsymbol{Y}} \tag{3.18}$$

where $\mathbf{F}$ is a nonlinear model, $\boldsymbol{X}$ is the input of model $\mathbf{F}$ and $\tilde{\boldsymbol{Y}}$ consists of observation data.

This problem can not be solved directly due to its nonlinearity. Instead, it is solved by minimizing a cost function of the form (3.5). Usually a gradient based optimization scheme is used to minimize the cost function in (3.5) iteratively. At each iteration, it attempts to find a better estimate $\boldsymbol{X}_{k+1}$ from the current estimate $\boldsymbol{X}_k$ using the gradient information of the cost function at $\boldsymbol{X}_k$. This process is in fact equivalent to solving the following linearized problem:

$$\mathbf{F}\boldsymbol{X}_{k+1} = \mathbf{F}\boldsymbol{X}_k + \mathbf{A}(\delta\boldsymbol{X}) + o(\delta\boldsymbol{X}) = \tilde{\boldsymbol{Y}} \tag{3.19}$$

where $\mathbf{A}$ is the Jacobian matrix of nonlinear operator $\mathbf{F}$ at $\boldsymbol{X}_k$, $\delta\boldsymbol{X}$ represents the changes from $\boldsymbol{X}_k$ to $\boldsymbol{X}_{k+1}$. $\boldsymbol{X}_{k+1}$ is then updated by $\boldsymbol{X}_{k+1} = \boldsymbol{X}_k + \delta\boldsymbol{X}$. If equation (3.19) is not well-posed, then the optimization routine is much likely to find an unstable solution. If equation (3.19) is well posed, the optimization routine has a better chance to find stable solutions.

Equation (3.19) can be reformulated as:

$$\mathbf{A}(\delta\boldsymbol{X}) = \tilde{\boldsymbol{Y}} - \mathbf{F}\boldsymbol{X}_k \tag{3.20}$$

Considering $\boldsymbol{X}_{k+1} = \boldsymbol{X}_k + \delta\boldsymbol{X}$, (3.20) is equivalent to:

$$\mathbf{A}\boldsymbol{X}_{k+1} = \tilde{\boldsymbol{Y}} - \mathbf{F}\boldsymbol{X}_k + \mathbf{A}\boldsymbol{X}_k \tag{3.21}$$

Let $\boldsymbol{B} = \tilde{\boldsymbol{Y}} - \mathbf{F}\boldsymbol{X}_k + \mathbf{A}\boldsymbol{X}_k$, then

$$\mathbf{A}\boldsymbol{X}_{k+1} = \boldsymbol{B} \tag{3.22}$$

Let matrix $\mathbf{A}$ be an m by n matrix. In our case, n is the number of parameters to estimate; m is the number of options. $\mathbf{A}$ can be reduced to the following form using Singular Value Decomposition (SVD).

$$\begin{aligned} \mathbf{A}_{mn} &= \mathbf{U}_{mm}\mathbf{S}_{mn}\mathbf{V}_{nn}^T \\ &= [\mathbf{U}_p, \mathbf{U}_0] \begin{bmatrix} \mathbf{S}_p & 0 \\ 0 & 0 \end{bmatrix} [\mathbf{V}_p, \mathbf{V}_0]^T \\ &= \mathbf{U}_p\mathbf{S}_p\mathbf{V}_p^T \end{aligned}$$

where p is the number of non-zero singular values s_i of matrix $\mathbf{A}$. Since m is less than n in our problem, $p \leq m$. $\mathbf{U}_{mm}, \mathbf{V}_{nn}$ are orthogonal matrices. $\mathbf{S}_{mn}$ is a diagonal matrix. $\mathbf{U}_p, \mathbf{V}_p$ are the first p columns of matrices $\mathbf{U}$ and $\mathbf{V}$ respectively. $\mathbf{S}_p$ is a diagonal matrix containing all the non-zero singular eigenvalues s_i. The singular values s_i are all positive and gradually decrease to zero.

The solution to (3.22) then can be written as in [9] :

$$\boldsymbol{X}_{k+1} = \mathbf{X}_\dagger + \tilde{\boldsymbol{X}} = \mathbf{V}_p\mathbf{S}_p^{-1}\mathbf{U}_p^T\boldsymbol{B} + \tilde{\boldsymbol{X}} = \sum_{i=1}^{p} \frac{(\mathbf{U}_{\cdot,i})^T\boldsymbol{B}}{s_i}\mathbf{V}_{\cdot,i} + \sum_{i=p+1}^{n} \alpha_i\mathbf{V}_{\cdot,\mathbf{i}} \tag{3.23}$$

where $\mathbf{V}_{\cdot,\mathbf{i}}$ is the ith column of matrix $\mathbf{V}$.

From (3.23) we can see that the solution $\boldsymbol{X}_{k+1}$ is composed of two parts: $\boldsymbol{X}_\dagger$ and $\tilde{\boldsymbol{X}}$. $\boldsymbol{X}_\dagger$ is the solution obtained from solving $\mathbf{U}_p\mathbf{S}_p\mathbf{V}_p^T\boldsymbol{X} = \boldsymbol{B}$ while $\tilde{\boldsymbol{X}} = \sum_{i=p+1}^{n} \alpha_i\mathbf{V}_{\cdot,\mathbf{i}}$ is a vector that lies in the null space of matrix $\mathbf{A}$. The existence of $\tilde{\boldsymbol{X}}$ shows the under-determined nature of this inverse problem.

For the solution $\boldsymbol{X}_\dagger = \Sigma_{i=1}^{p} (\mathbf{U}_{\cdot,i})^T\boldsymbol{B}/s_i\mathbf{V}_{\cdot,i}$, if $(\mathbf{U}_{\cdot,i})^T\boldsymbol{B}$ does not decay as fast as s_i, $\boldsymbol{X}_{k+1}$ will become unstable as s_i tends to zero, since a small amount of noise from $\boldsymbol{B}$ will be amplified by the small singular values.

After diagnosing where the ill-posedness originates from, we propose to regularize the ill-posedness by eliminating the effects of the small singular values s_i. The addition of Tikhonov

regularization at each iteration is equivalent to solving the following over-determined linear problem:

$$\begin{bmatrix} \mathbf{A} \\ \lambda \mathbf{L} \end{bmatrix} \boldsymbol{X}_{k+1} = \begin{bmatrix} \boldsymbol{B} \\ \mathbf{0} \end{bmatrix} \tag{3.24}$$

When $\mathbf{L}$ represents a higher order Tikhonov regularization operator, as in our case, the analytical solution of (3.24) can be obtained by applying a generalized singular value decomposition (GSVD) of the matrix pair $[\mathbf{A}^T, \mathbf{L}^T]^T$, see [38] and [9] for details. But GSVD of the matrix pair is computationally expensive especially since the dimension of our problem is large. In addition, most GSVD packages require the explicit form of matrices $\mathbf{A}$ and $\mathbf{L}$, neither of which is generated explicitly in our method. Extra computation and storage are necessary to generate and store the matrices $\mathbf{A}$ and $\mathbf{L}$ in order to use the GSVD packages. Furthermore, if we need to carry out a GSVD to find the regularization parameter λ at each iteration, the total computational cost of the minimization of (3.17) becomes very expensive.

To avoid using GSVD, we consider the special case when $\mathbf{L}$ is the identity matrix in order to gain insight of the problem. When $\mathbf{L}$ is the identity matrix, the regularized solution of (3.24) assumes the following analytical form of (3.25), see [38] for the derivation.

$$\boldsymbol{X}_\lambda = \sum_{i=1}^{m} \frac{{s_i}^2}{\lambda^2 + {s_i}^2} \frac{(\mathbf{U}_{\cdot,i})^T \boldsymbol{B}}{s_i} \mathbf{V}_{\cdot,i} \tag{3.25}$$

When $\lambda \gg s_i$, the weighting factor $f = \frac{{s_i}^2}{\lambda^2 + {s_i}^2}$ is about 0. When $\lambda \ll s_i$, f is about 1. Choosing a λ that is smaller than the leading singular values and greater than the smallest singular values eliminates the ill-posedness caused by small singular values, yet does not affect the information represented by the large singular values. The under-determined part $\tilde{\mathbf{X}}$ is also eliminated. Guided by this insight, we choose our regularization parameter λ to be one of the singular values of $\mathbf{A}$ determined by the truncation level defined in the following:

$$\frac{\sum_{k=1}^{i} s_i}{\sum_{k=1}^{m} s_i} = \text{truncation level} = 50\% \tag{3.26}$$

where the singular values are sorted in order of magnitude such as $s_1 \geq s_2 \geq \ldots \geq s_m \geq 0$, where m is the total number of singular values.

Now the question is how to compute the singular values of $\mathbf{A}$ at each iteration? We need an algorithm that can compute the singular values without requiring the explicit form of matrix $\mathbf{A}$. For this purpose we will use the package ARPACK which meets this requirement. All it requires is the product of matrices $\mathbf{A}$ and $\mathbf{A}^T$ with a vector. For our problem, the tangent linear code and adjoint code derived from the automatic differentiation tools readily compute these two products. ARPACK is based upon an algorithmic variant of the Arnoldi process called the Implicitly Restarted Arnoldi Method (IRAM). See [49] for details.

Computing the singular values at each iteration to determine λ is the most computationally expensive part of our algorithm. We found out from numerical experiments,to be detailed in the following section, that the λ selected in this manner does not change much throughout the minimization. By using a fixed λ, the reconstructed volatility surface remains the same as the one reconstructed by repeatedly updating λ at each iteration. However the computational time is significantly reduced by using a fixed λ. If we assume that λ

selected according to (3.26) is almost a constant during the minimization of (3.17), an alternative and efficient strategy of choosing λ consists in using a constant λ selected according to (3.26) throughout the minimization process of (3.17). This assumption is valid in the case where the Jacobian matrix $\mathbf{A}$ does not change significantly during the minimization process. If we assume that the initial guess $\boldsymbol{X}_0$ is not far away from the optimal solution $\boldsymbol{X}^*$, then we can assume that $\mathbf{A}$ is almost constant. In our calibration problem, the initial guess $\boldsymbol{X}_0$ is set as a constant volatility surface obtained by averaging the Black-Scholes implied volatilities of the ATM options across different maturities. If we assume the true local volatility surface does not deviate much from the average of the Black-Scholes implied volatility surface of ATM options, then the assumption that $\mathbf{A}$ is constant is reasonable. In this case, we can assume λ is constant. However, for a general model when $\boldsymbol{X}_k$ changes significantly across iterations, we have to choose a λ at each iteration. For this reason, we still exhibit the pseudo-algorithm for the general case in algorithm (2) on the following page.

With the gradient obtained from the previous section and the regularization parameter λ ready, we can use a constrained optimization routine to find the optimal σ of (3.17). We use the algorithm L-BFGS-B to carry out the optimization. For details of L-BFGS-B, see [66]. This is a robust algorithm for bound-constrained minimization. Prior to discussing our numerical tests, we summarize our pseudo-algorithm description in the following.

Algorithm 2 Main algorithm to reconstruct the local volatility surface

1. Initialize volatility surface $\sigma_0(s,t)$.
2. Use (3.3) to compute option prices V_{cmpt} and cost function G in (3.2).
3. Feed the difference between V_{cmpt} and V_{obs} into the adjoint model $\mathbf{A}^T$, using (3.6) and (3.8), to compute the gradient of G with respect to $\sigma(s,t)$.
4. Use ARPACK to compute the singular values of Jacobian matrix $\mathbf{A}$ and select the regularization parameter λ according to (3.26).
5. Compute the regularized cost function J of (3.17) and update the gradient after the regularization.
6. Insert the cost function J and its gradient into L-BFGS-B routine to obtain the next estimate $\sigma_{k+1}(s,t)$. $\quad k = 0, 1, 2, \cdots$
7. When either the stopping criterion of L-BFGS-B is satisfied or the number of function calls of the cost function exceeds a preset limit, stop. Otherwise, go back to step 2.

For theoretical volatility models, the limit on the number of function calls is 1500 while for the case of real market data the limit is 250. We allow more iterations for the theoretical volatility models since the true volatility surfaces are known. The recovered volatility surface actually displays the general features of the true volatility surface after 250 function calls, which is why we set the upper limit of function calls for the real market data as 250.

3.4 Numerical tests

For all of our numerical tests, the initial guess σ_0 is the average of Black-Scholes implied volatilities for the ATM options across different maturities. We scale the spot price of the underlying to 100 and then the option prices are scaled accordingly. The scaling reduces the calibration problem for different underlying instruments into the same problem. It has the additional benefit that λ can be precomputed and applied to different problems when we assume the regularization parameter λ is constant and when r and q do not change significantly.

The Dupire equation (3.3) is solved using the backward Euler scheme in time and a centered finite difference scheme in space direction. The computation domain $\begin{bmatrix} 0 & \bar{T} \end{bmatrix} \times \begin{bmatrix} 0 & \bar{K} \end{bmatrix}$ is set as $\bar{K} = 2s_0$ as in [48] while $\bar{T}$ is the longest maturity. The space and time domain are divided into $n_x = 200$ and $n_t = 100$ intervals respectively. Since only the section of volatility surface $\sigma(s,t)$ for which the ratio s/s_0 lies in $[0.8, 1.2]$ can be recovered, the total number of parameters to calibrate is $0.2 \times 200 \times 100 = 4000$. The lower and upper bound for σ is set to be 0.00001 and 1, respectively. We perform two kinds of numerical tests: one for volatility models, whose analytical solution for European options are known, and the other one for the real market options data.

3.4.1 Tests with theoretical volatility models

We start with the constant elasticity of variance (CEV) model, for which the analytical form of European option prices can be found in [23]. The CEV model assumes the following form:

$$ds(t) = \mu s dt + \kappa s^p dW(t)$$

According to our definition of local volatility in (3.1), the local volatility for the CEV model is :

$$\sigma(s,t) = \kappa s^{p-1} \tag{3.27}$$

We will test three cases: $p = 0$, $p = \frac{1}{2}$ and $p = 2$. When $p = 0$, it corresponds to the Bachelier model. When $p = \frac{1}{2}$, it corresponds to the square root process. When $p = 2$, it is a special case of quadratic volatility model. The first case was used as a test case in both [48] and [18]. Specifically, we test the following three cases:

$$\sigma(s,t) = \frac{15}{s} \tag{3.28}$$

$$\sigma(s,t) = \frac{2}{\sqrt{s}} \tag{3.29}$$

$$\sigma(s,t) = 0.002s \tag{3.30}$$

The constant κ in (3.27) is chosen in order that $\sigma(s,t)$ is contained in the interval of (0, 1) for all s and t. Twenty two European call option prices are generated using the closed-form solution for two maturities $T = 0.5$ and $T = 1.0$. For each maturity, we select eleven options whose strikes range from 90.0 to 110.0 with an increment of 2.0. These option prices

are used to recover the volatility surface for (3.28-3.30). Similar to the study of [48]; [18], $s_0 = 100$, the risk free interest rate $r = 0.05$ and the dividend yield $q = 0.02$ for all three cases.

Figures 3.1-3.3 show the recovered volatility surface and the true volatility surface. For all the three cases, the recovered volatility surfaces approximate the true volatility surfaces very well. The relative errors of the computed option prices with respect to the true option prices are of the order of 10^{-4}. Figure 3.4 shows the plot of relative errors and option prices with respect of the number of options for the case (3.30). For the other two cases, the plots of relative errors exhibit similar patterns.

In order to compare the difference between the first order Tikhonov regularization and the second order Tikhonov regularization, we exhibit Figures 3.5 and 3.6 which show the recovered volatility surface by using the first order Tikhonov regularization for two CEV models. We can see that even for these two simple CEV models, the first order Tikhonov regularization could not match the true volatility surface as precisely as the second order Tikhonov regularization.

For all of the above CEV models, $\sigma(s,t)$ are monotonic functions of s. Next, we deliberately choose a quadratic volatility model that is not monotonically changing as our test case. [6] summarizes the analytical solution of European option prices for different quadratic volatility models. The following quadratic volatility model is taken from his paper.

$$\sigma(s,t) = 0.1(1 + \frac{s_0}{s} + \frac{(s-s_0)^2}{100s}) \tag{3.31}$$

A total of 22 European put options with the same set of maturities and strikes as the previous tests are computed as market data. s_0 is set to 100, the risk free interest rate r and the dividend yield q are both zero. (When the drift is not zero, a change of measure can reduce the drift to zero). Figure 3.7 plots the true volatility surface as well as the recovered volatility surface. We observe that the recovered volatility surface approximates the true volatility surface fairly well. Figure 3.8 displays the relative errors of computed option prices with respect to the true prices. We notice that the relative errors are of order of 10^{-4}. Figure 3.9 displays the cost function J with respect to iteration number. Figure 3.10 shows the decrease of the norm of the projected gradient as the number of iterations increases.

To test the stability of our methods, we add noise perturbation to the true option prices to assess whether we can still recover the volatility surface. The noise perturbation is introduced as in ([18]):

$$\tilde{v_i} = v_i + 0.02\epsilon_i$$

where v_i is the true price of the ith option, ϵ_i is a uniformly distributed random number between 0 and 1. The noises are introduced as absolute errors rather than relative errors. The plot of the reconstructed volatility surfaces using noisy option prices and noise-free option prices is shown in Figure 3.11 for the quadratic volatility model. We notice that the two volatility surfaces are indistinguishable from each other with the maximum absolute difference of the order of 10^{-3}. It means that our method exhibit stability with respect to a small amount of perturbation.

Next, we test the case when the noises are introduced as relative errors:

$$\tilde{v_i} = v_i(1 + 0.02\epsilon_i)$$

2% of uniformly distributed noises are added as relative errors to the option prices. A direct calibration without any weighting of the noisy option prices fails to recover the true volatility surface. When relative errors are introduced, a proper weighting scheme needs to be introduced so as to reflect the relative importance of different options. We adopt the weighting method as in [21] to scale the noisy prices in this case, which is defined as :

$$w_i = \frac{1}{vega(I_i)^2}$$

where I_i is the Black-Scholes implied volatility of the ith noisy option price, $vega()$ is the Black-Scholes vega evaluated as a function of implied volatility. Figure 3.12 displays the recovered the volatility surface with respect to the true volatility surface. We notice that the recovered volatility surface approximates the true volatility surface very well.

The total CPU time required for each of the previous six tests lies between 332 and 480 seconds using a Dell Vostro 1720 with Intel Core Duo CPU @2.2G HZ and 2GB RAM.

The above calibration updates the regularization parameter λ at each iteration. Figure 3.13 displays λ against the number of iterations for the quadratic volatility model. At the beginning, λ is set to zero. We notice that λ does not vary much throughout iterations and that it almost stays constant after a number of iterations. The same phenomenon is observed for other test cases as well as the real market data cases in the following section. Based on this observation, we use a constant λ during the optimization. Figure 3.14 shows the recovered volatility surface by using a constant λ vs an updated λ for the quadratic volatility model with noise free prices. The two constructed volatility surfaces are indistinguishable from each other with the maximum absolute difference being of the order of 10^{-3}. By using a constant λ the total CPU time required for each of the previous six tests is now just between 13 and 20 seconds.

3.4.2 Tests with market data

All our market data are obtained from previous studies on the calibration of local volatility surface. Our first test uses option prices as in [18]; [4] and [65]. The options are European call options on S&P 500 index in October 1995. There are a total of 57 options with seven maturities. The initial index, interest rate, and dividend yield are provided in the footnotes of Figure 3.15. Figure 3.15 shows the optimal volatility obtained. Contrary to previous studies, the volatility surface exhibits an obvious skew structure as expected for the equity market. The volatility surface is also smoother. Furthermore, the recovered volatility surface is in a range between 0.08 and 0.30 without local extreme values. The relative errors of computed prices with respect to observed prices are plotted in Figure 3.16. The relative errors are mostly close to zero except for options whose prices are close to zero. This is acceptable since the bid and ask spreads for out of money options are usually much higher than or comparable with the option prices. In other words, out of money option prices allow a much higher degree of approximation errors. The mean absolute relative error is 4.7%.

Excluding the seven options with big absolute relative errors, the mean absolute relative error is as small as 0.2%.

The second test uses data set from [7], which contains 155 European options on the Eurostoxx (SX5E) index spanning 12 maturities. The shortest maturity was about one week ($T = 0.025$) and the longest maturity was about 5.8 years ($T = 5.778$). Since the original data only had market data in terms of implied volatilities without the interest rate structure, we computed the option prices just using these implied volatilities under the assumption the interest rates were zero. This assumption is reasonable since the local volatility model (3.1) can be changed into a driftless process by a change a measure while the local volatility term keep the same during the change. The recovered volatility surface is shown in Figure 3.17. Figure 3.18 shows the absolute errors and the option prices in terms of implied volatilities as in [7]. Figure 3.19 plots the relative errors and the option prices in terms of prices. Figure 3.17 displays an obvious skew structure although there are a lot of fluctuations of the local volatility surface close to the region when $T = 0.025$. The computed data does not match the market data very well either for that maturity. However, one of the possible reasons is that our finite difference scheme does not have enough resolution at $T = 0.025$. By setting $\bar{T} = 5.778$, $nt = 100$ and using a uniform grid, $\Delta t = 0.058 > 0.025$. The situation is alleviated by a finer mesh grid. Ignoring the 15 options with maturity $T = 0.025$, Figures 3.18 and 3.19 demonstrate a very good fit of the market prices. Again, high relative errors occur when the option prices are close to zero (Figure 3.19). For the remaining 140 options, the mean relative error is 2% and the mean absolute difference in terms of implied volatility is 0.6%. Next, we refine the mesh by setting $nt = 500$. Figure 3.20 shows the recovered volatility surface. Compared to Figure 3.17, the volatility surface does not change significantly. Anyway, the region when $T \leq 0.025$ just occupies a small section of the volatility surface. Figure 3.21 exhibits the absolute difference in terms of implied volatility. Compared to Figure 3.18, we can see that there are some improvements in terms of matching the prices for the options with $T = 0.025$. The CPU time in this case is very high, namely 32 minutes when λ is selected iteratively. A non-uniform grid should be used to reduce the computational cost when it is necessary to resolve cases like this with the maximum maturity $\bar{T}$ large yet the shortest maturity being very small.

The last example is for European call options in the foreign exchange market. The option data were studied by both [10] and [65]. There are 15 European call options for the US dollar/Deutsche mark with 5 maturities, which are computed from 20, 25 and 50 delta risk-reversals quoted on Aug 23, 1995. The spot price and interest rates are shown in the captions of Figure 3.22. The optimal volatility surface and relative errors are plotted in Figures 3.22 and 3.23 respectively. The volatility surface has a shape similar to the smile shape as expected for volatilities in the foreign exchange market. The mean absolute relative error is as small as 1.9%.

There may still be some instability in the volatility surface recovered, for example the reconstructed volatility surface for the last example. We attribute this issue partially to the assumption that every option is equally important. The amount of noises in the market option prices is unknown. A proper weighting scheme is necessary to reflect the relative importance of different options. This will constitute an interesting follow-up future research area.

For the above three numerical tests, the required CPU time is 158, 232, and 12 seconds

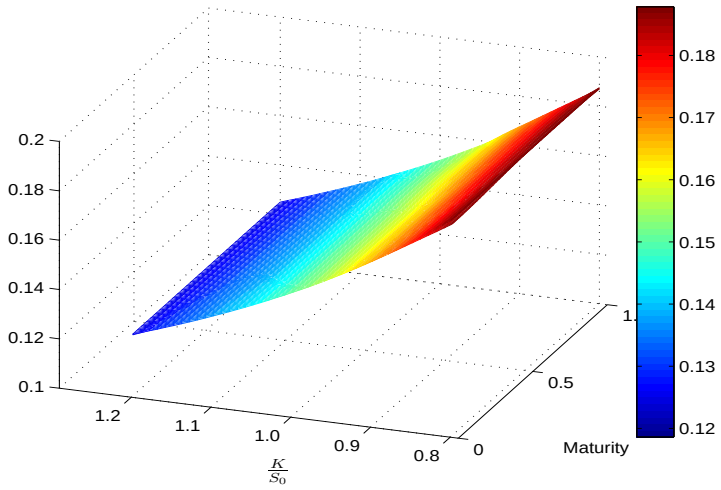

Figure 3.1: The true volatility surface and optimal volatility surface for the volatility model $\sigma(s,t) = \frac{15}{s}$

respectively, using a Dell Vostro 1720 with Intel Core Duo CPU @2.2G HZ and 2GB RAM. Again, when we use a constant λ, the CPU time required is just as small as 3.4, 3.6, 3.6 seconds respectively. The changes of the relative errors and recovered implied volatility surface are again very small compared to those obtained using an updated λ. From here we can see that when using a constant λ, the CPU time is independent of the number of the options. When $nt = 500$, the CPU time for the data set from [7] is 18.9 seconds. From this example, we observe that when using a constant λ the CPU time grows linearly as the number of parameters increases, which results from the linear dependence of computational cost of an adjoint model on the number of parameters, as mentioned by [32].

The only parameter that is subject to change in our algorithm is the **truncation level**. It is fixed at 50% throughout our numerical tests. Other truncation levels were also tested. The relative error and the general shape of the optimal volatility surface did not change significantly overall when the truncation level was less than 0.9 although as the truncation level gets lower the volatility surface tends to be smoother. This means this method is fairly robust for different choices of truncation levels as long as the regularization parameter selected is not close to the smallest singular values at the end of the singular values spectrum. The fact that we used the same truncation level for all numerical tests also serves as an indication that the calibration is not very sensitive to the truncation levels.

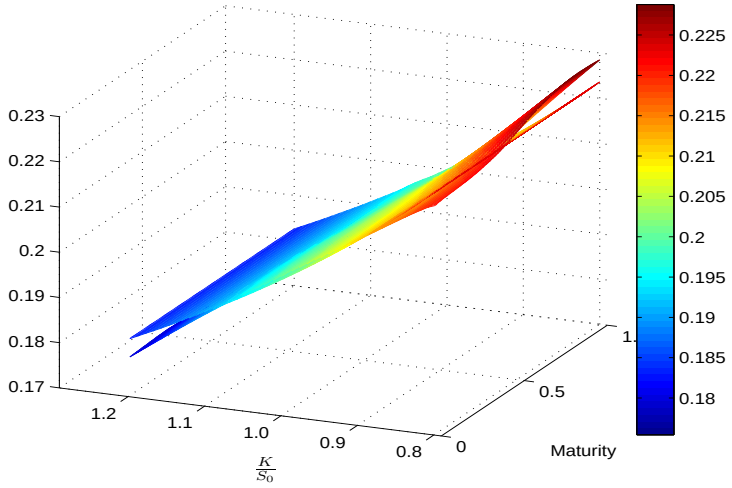

Figure 3.2: The true volatility surface and optimal volatility surface for the volatility model $\sigma(s,t) = \frac{2}{\sqrt{s}}$

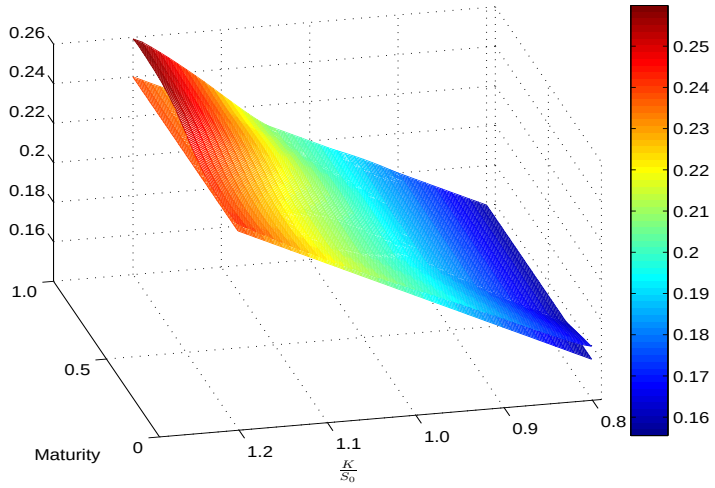

Figure 3.3: The true volatility surface and optimal volatility surface for the volatility model $\sigma(s,t) = 0.002s$

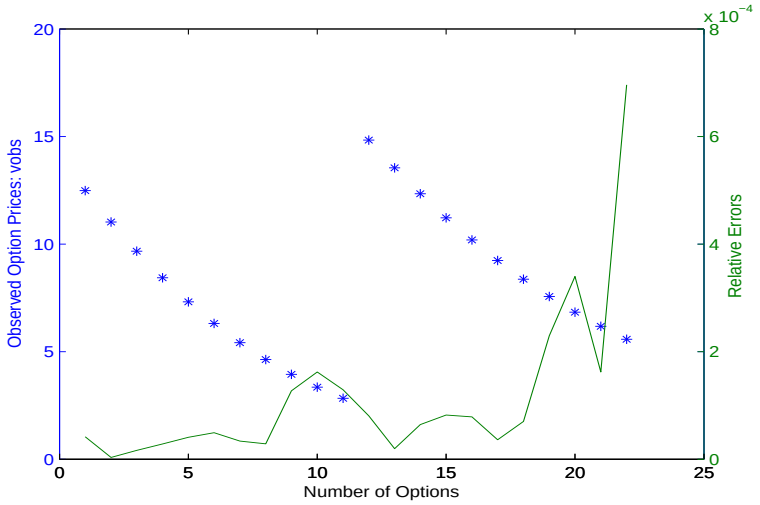

Figure 3.4: $*$: option prices; _ : relative errors = |vobs-vcmpt|/vobs. Left axis: The true option prices. Right axis: The relative errors of the computed option prices using optimal volatility surface with respect to the true prices for the volatility model $\sigma(s,t) = 0.002s$

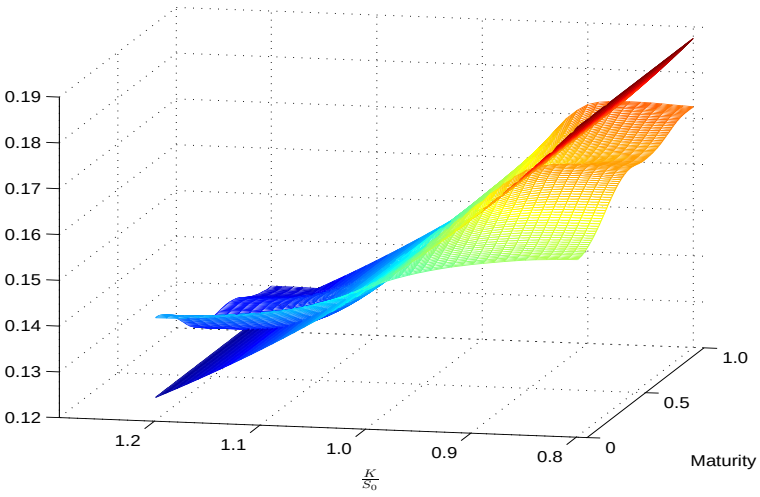

Figure 3.5: The true volatility surface and optimal volatility surface for the volatility model $\sigma(s,t) = \frac{15}{s}$ using the first order Tikhonov regularization

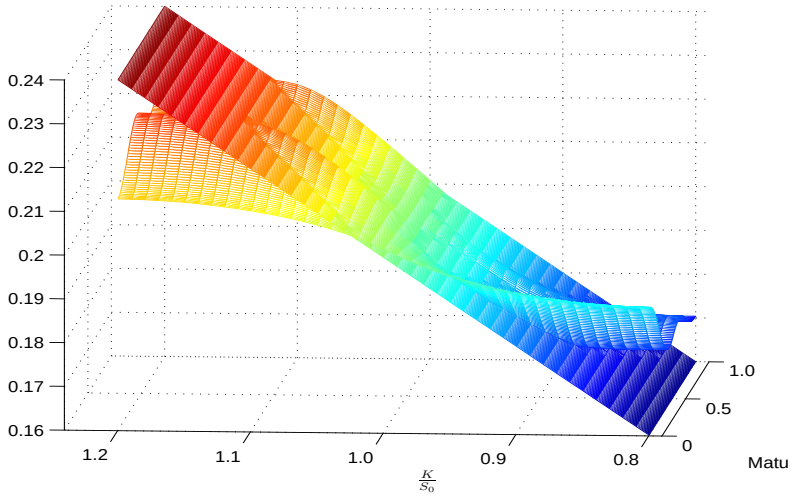

Figure 3.6: The true volatility surface and optimal volatility surface for the volatility model $\sigma(s,t) = 0.002s$ using the first order Tikhonov regularization

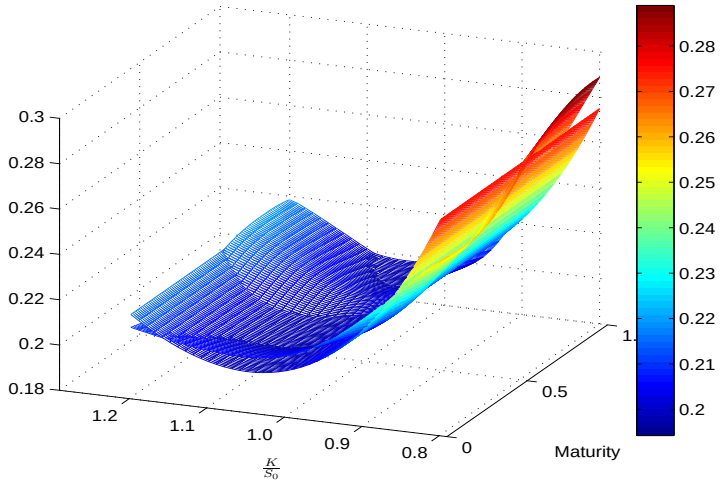

Figure 3.7: The true volatility surface and optimal volatility surface for the volatility model $\sigma(s,t) = 0.1(1 + \frac{s_0}{s} + \frac{(s-s_0)^2}{100s})$

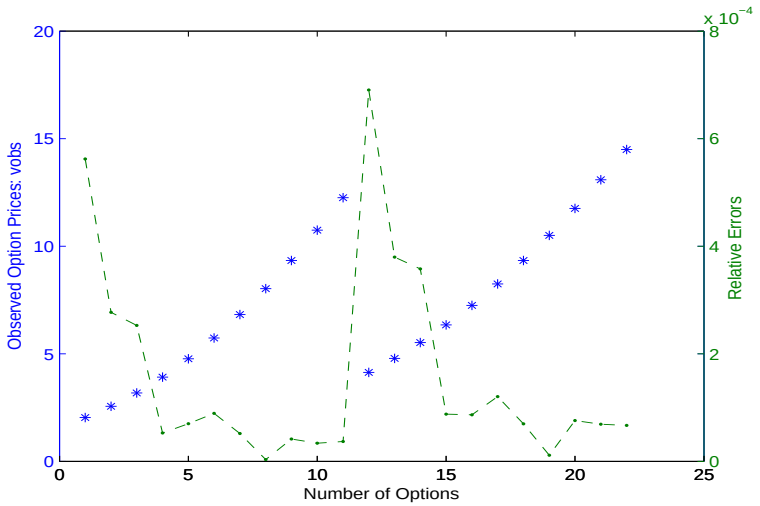

Figure 3.8: $*$: option prices; _ : relative errors. Left axis: The true option prices. Right axis: The relative errors of the computed option prices using optimal volatility surface with respect to the true prices for the volatility model $\sigma(s,t) = 0.1(1 + \frac{s_0}{s} + \frac{(s-s_0)^2}{100s})$

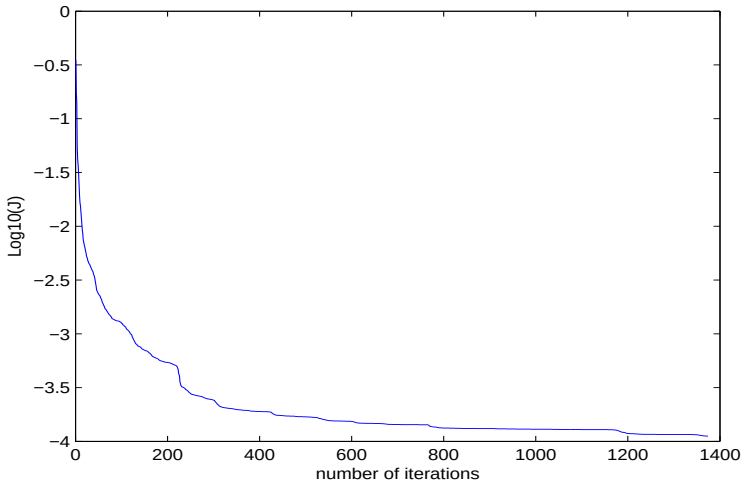

Figure 3.9: The evolution of the regularized cost function versus the number of minimization iterations for the volatility model $\sigma(s,t) = 0.1(1 + \frac{s_0}{s} + \frac{(s-s_0)^2}{100s})$

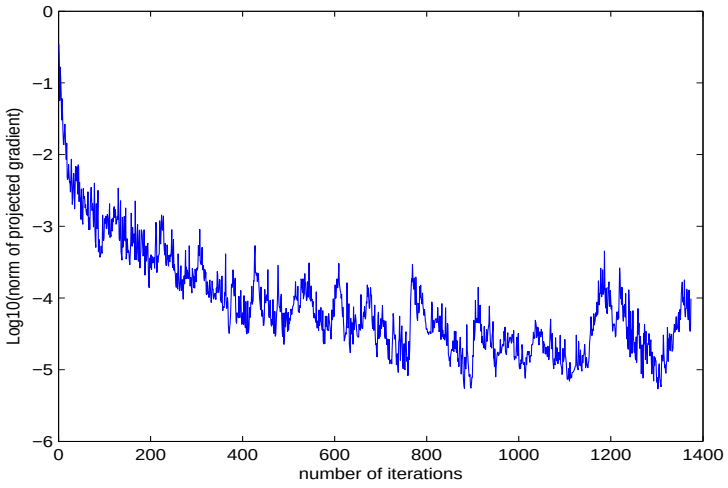

Figure 3.10: Variation of the norm of the projected gradient of the cost function vs the number of iterations for the volatility model $\sigma(s,t) = 0.1(1 + \frac{s_0}{s} + \frac{(s-s_0)^2}{100s})$

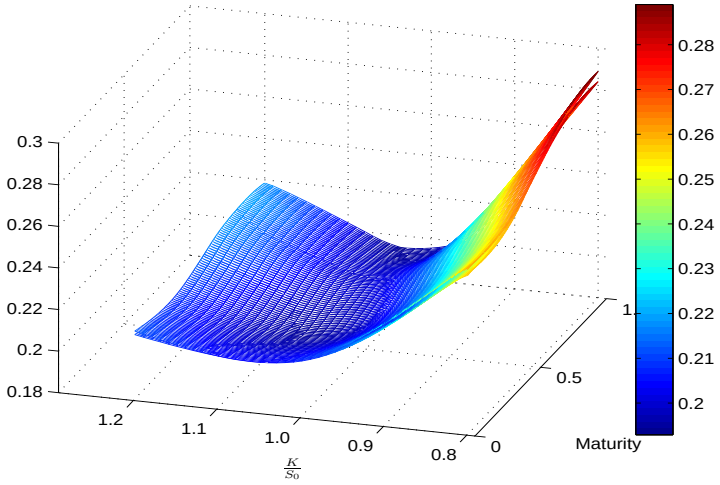

Figure 3.11: The optimal volatility surfaces obtained from the non-noisy and noisy option prices(2% uniformly distributed absolute noise) for the volatility model $\sigma(s,t) = 0.1(1 + \frac{s_0}{s} + \frac{(s-s_0)^2}{100s})$

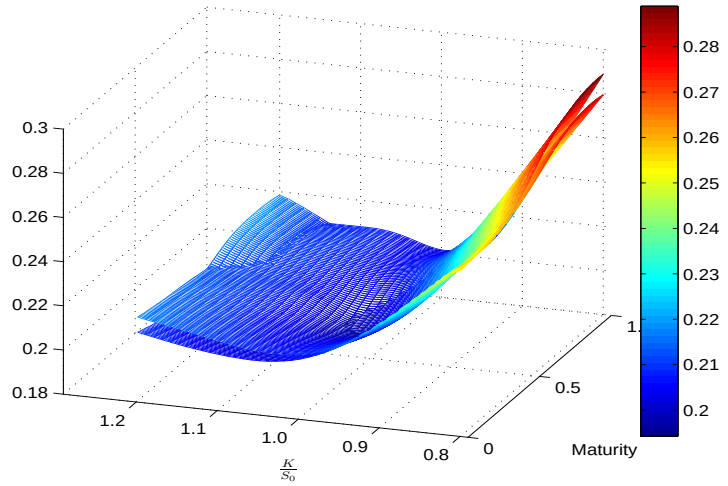

Figure 3.12: The true volatility surface and the optimal volatility surface obtained from the noisy option prices(2% uniformly distributed relative noise) for the volatility model $\sigma(s,t) = 0.1(1 + \frac{s_0}{s} + \frac{(s-s_0)^2}{100s})$

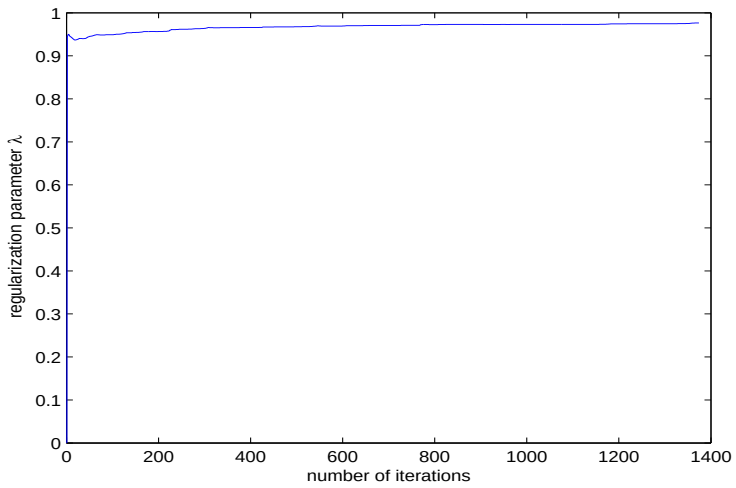

Figure 3.13: The regularization parameter λ computed at each iteration for volatility model $\sigma(s,t) = 0.1(1 + \frac{s_0}{s} + \frac{(s-s_0)^2}{100s})$

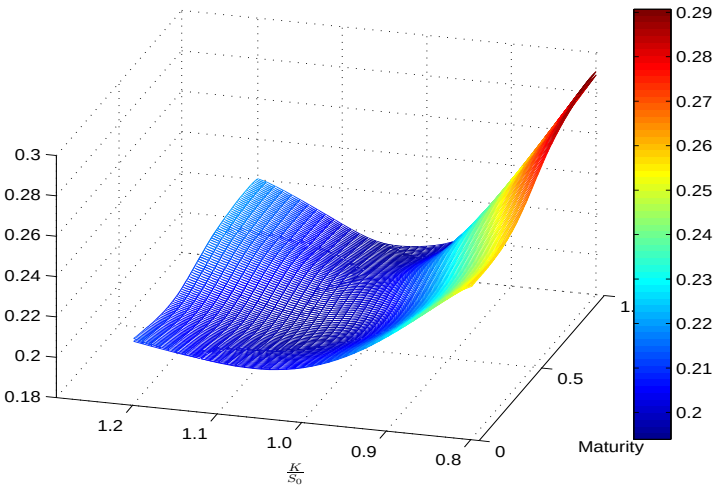

Figure 3.14: The optimal volatility surfaces obtained by using a constant $\lambda = 0.94$ and a λ updated at each iteration for volatility model $\sigma(s,t) = 0.1(1 + \frac{s_0}{s} + \frac{(s-s_0)^2}{100s})$

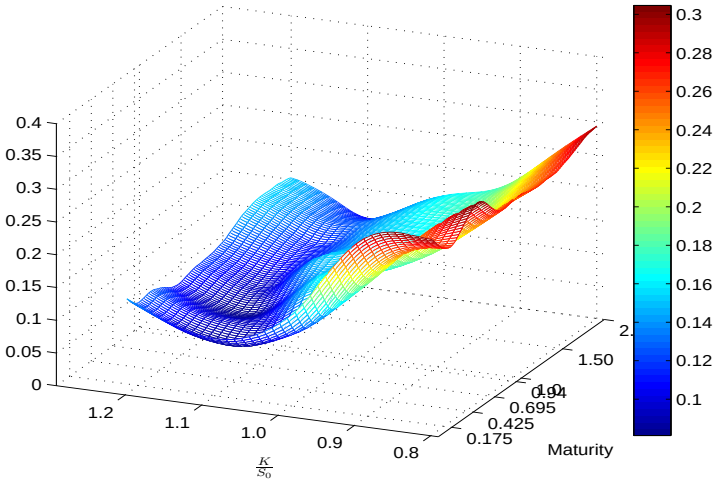

Figure 3.15: The optimal volatility surface obtained for S&P 500 index European call options in October 1995. s_0 =\$ 590, r=0.06, q=0.0262. Note: the available maturities are plotted on the T axis in units of years.

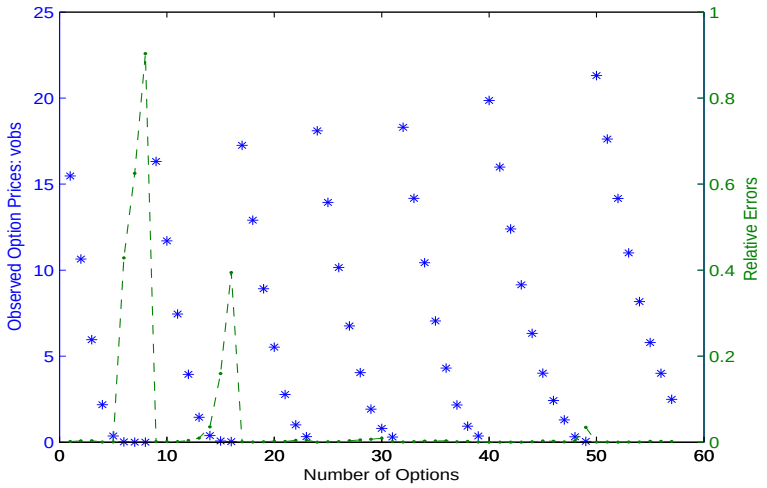

Figure 3.16: $*$: scaled option prices; _ : relative errors. Left axis: The scaled prices of S&P 500 index European call options in October 1995 [4]. Right axis: The relative errors of computed option prices with respect to observed price. Option prices are plotted in an order of increasing maturities.

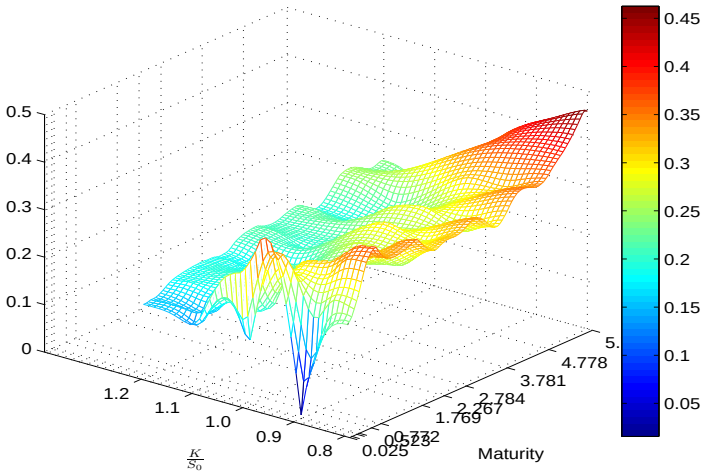

Figure 3.17: The optimal volatility surface obtained for Eurostoxx 50(SX5E) equity options on March 1, 2010, as studied in [7]. Note: the available maturities are plotted on the T axis in units of years.

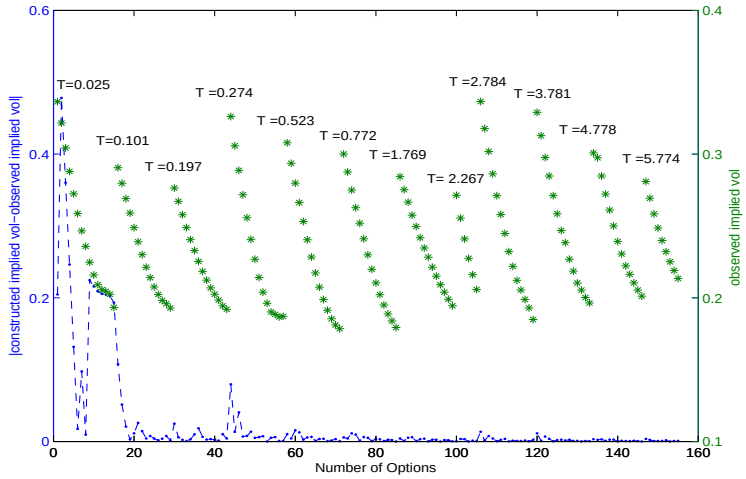

Figure 3.18: $*$: option prices in terms of implied volatilities; $\_$: absolute difference of the implied volatilities between market data and reconstructed option prices. Left axis: The absolute difference of implied volatilities between market data and reconstructed data. Right axis: The option prices for SX5E in terms of implied volatilities on March 1, 2010.

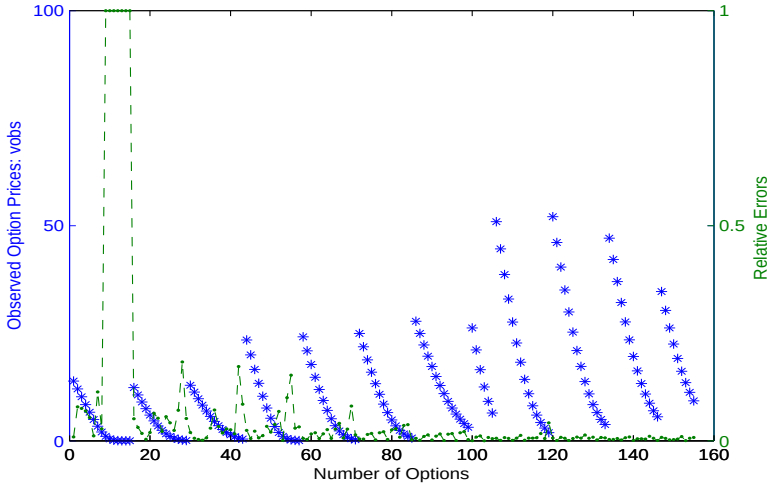

Figure 3.19: $*$: scaled option prices ; $\_$: relative error between market data and reconstructed option prices. Left axis: The scaled option prices for SX5E on March 1, 2010. Right axis: The relative error between market data and reconstructed data. Option prices are plotted in an order of increasing maturities.

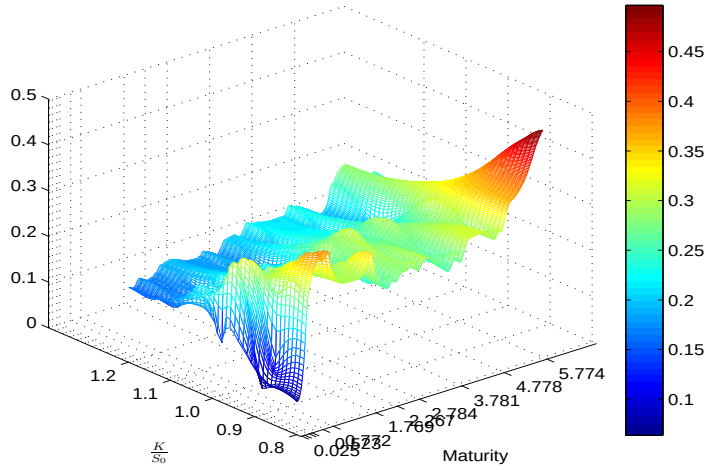

Figure 3.20: The optimal volatility surface obtained when nt= 500 for Eurostoxx 50 (SX5E) equity options on March 1, 2010, as studied in [7]. Note: the available maturities are plotted on the T axis in units of years.

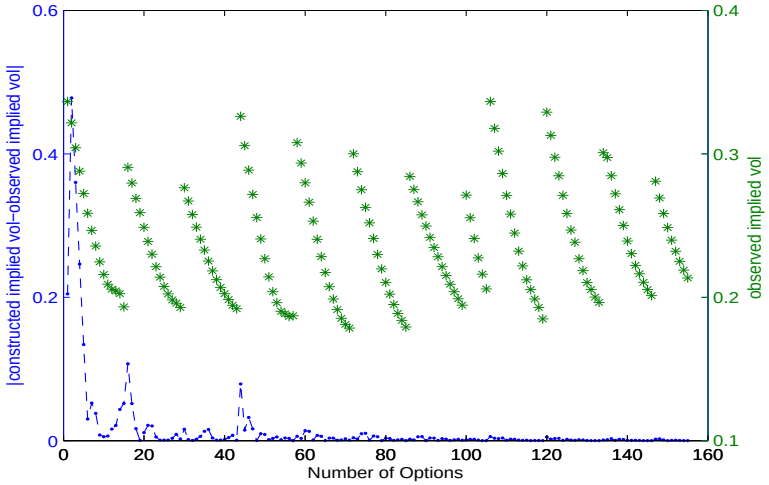

Figure 3.21: $*$: option prices in terms of implied volatilities; _ : absolute difference of the implied volatilities between market data and reconstructed option prices. Left axis: The absolute difference of implied volatilities between market data and reconstructed data when nt=500. Right axis: The option prices for SX5E in terms of implied volatilities on March 1, 2010. Option prices are plotted in an order of increasing maturities.

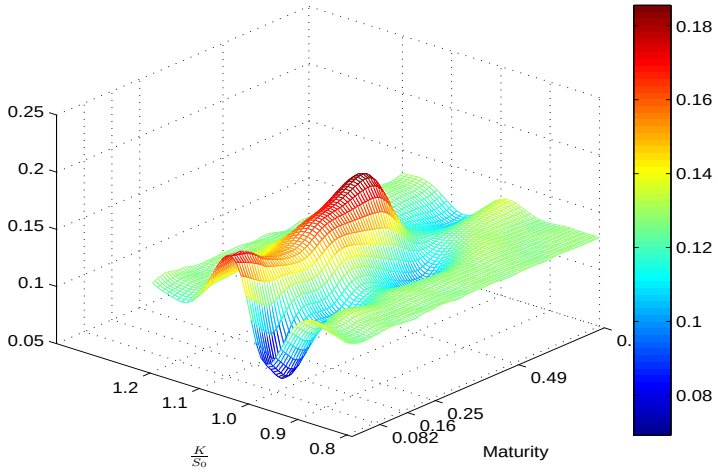

Figure 3.22: The optimal volatility surface obtained for European call options of US dollar/ Deutsche mark rate. The spot price was s_0 =1.48875; US dollar interest rate was r_{USdollar} = 5.91%; Deutsche mark rate was $r_{\text{Deutschemark}}$= 4.27%. Note: the available maturities are plotted on the T axis in units of years.

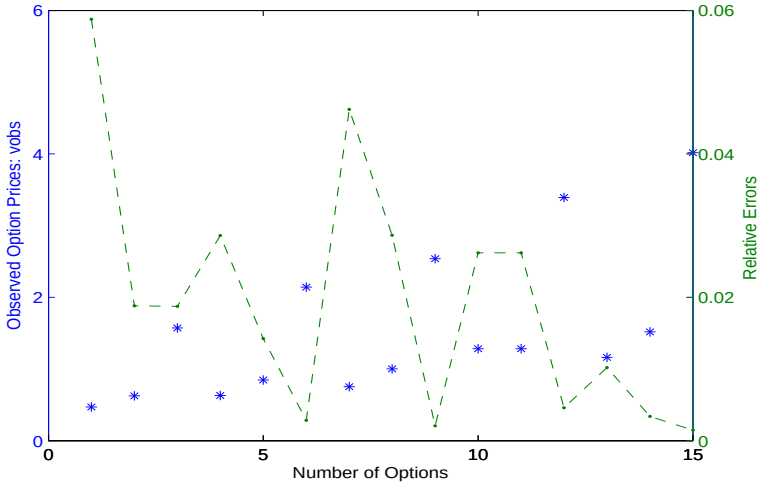

Figure 3.23: $*$: scaled option prices; _ : relative errors. Left axis: The scaled prices of European call options on US dollar/Deutsche mark rate recovered from 20, 25, and 50 delta risk reversals [10]. Right axis: The relative errors of computed option price with respect to observed price. Option prices are plotted in an order of increasing maturities.

CHAPTER 4

STOCHASTIC VOLATILITY MODELS

4.1 Description of stochastic volatility models

A general stochastic volatility model follows the form of :

$$\begin{aligned} dS_t &= \mu_S(S_t, V_t, t)dt + \sigma_S(S_t, V_t, t)dW_t^1 \\ dV(t) &= \mu_V(S_t, V_t, t)dt + \sigma_V(S_t, V_t, t)dW_t^2 \\ dW_t^1 dW_t^2 &= \rho \end{aligned} \tag{4.1}$$

where S_t denotes the asset price, V_t is the variance of return of the asset, and the two standard Brownian motion processes W_t^1, W_t^2 are correlated with correlation ρ. μ_S, σ_S, μ_V, and σ_V are usually deterministic functions of S_t and V_t. Given all the conditions that (4.1) needs to make it suitable to simulate asset prices, for example, non-negative asset price and non negative variance, (4.1) is usually used as an extension of local volatility model to price derivatives. Compared to the local volatility model, (4.1) has better dynamics of the volatility surface, see [5]. For some specific models of (4.1), analytical solution for vanilla options exists. However, for exotic derivatives, an analytical formula is usually not available and other methods are needed to compute the prices. The Monte Carlo method is one of the most popular methods. However, it converges slowly. Since (4.1) has two random processes, the computation can be expensive. Another alternative is a Partial Differential Equation (PDE) approach that relates the price of a derivative to a two dimensional convection diffusion equation.

4.2 PDE method

4.2.1 PDE formulation

Let U denotes the price of a derivative with the underlying asset S_t satisfying the process (4.1). Depending on whether using physical measure or risk neutral measure, there is some subtle difference in the derivation of the PDE for U. For example, if using the physical measure, the market price of risk is usually involved in the derivation, such as [40]. Since our focus is pricing derivatives, we'll use the risk neutral measure, which means $\mu_S = rS$. The following derivation closely follows [33]. Assuming U just depends on the terminal value

S_t, then U is a function of S_t, V_t. In notation, $U = U(S_t, V_t, t)$. Applying Itô's formula [44] to derivative price U, we have:

$$dU = \frac{\partial U}{\partial t}dt + \frac{\partial U}{\partial S}dS_t + \frac{\partial U}{\partial V}dV_t + \frac{1}{2}\frac{\partial^2 U}{\partial S^2}(dS_t)^2 + \frac{1}{2}\frac{\partial^2 U}{\partial V^2}(dV_t)^2 + \rho\frac{\partial^2 U}{\partial S \partial V}dS_t dV_t \tag{4.2}$$

Substituting (4.1)into (4.2) and ignoring all the higher order terms with respect to dt, we have:

$$\begin{aligned} dU &= \frac{\partial U}{\partial t}dt + \mu_S\frac{\partial U}{\partial S}dt + \mu_S\frac{\partial U}{\partial S}dW_t^1 + \mu_V\frac{\partial U}{\partial V}dt + \mu_V\frac{\partial U}{\partial V}dW_t^2 \\ &\quad + \frac{1}{2}\frac{\partial^2 U}{\partial S^2}\sigma_S^2 dt + \frac{1}{2}\frac{\partial^2 U}{\partial V^2}\sigma_V^2 dt + \rho\sigma_S\sigma_V\frac{\partial^2 U}{\partial S\partial V}dt \\ &= \left[\frac{\partial U}{\partial t} + \frac{1}{2}\sigma_S^2\frac{\partial^2 U}{\partial S^2} + \mu_S\frac{\partial U}{\partial S} + \frac{1}{2}\sigma_V^2\frac{\partial^2 U}{\partial V^2} + \mu_V\frac{\partial U}{\partial V} + \rho\sigma_S\sigma_V\frac{\partial^2 U}{\partial S\partial V}\right]dt \\ &\quad + \mu_S\frac{\partial U}{\partial S}dW_t^1 + \mu_V\frac{\partial U}{\partial V}dW_t^2 \end{aligned} \tag{4.3}$$

For the sake of compact notation, let us denote

$$L(U) = \frac{\partial U}{\partial t} + \frac{1}{2}\sigma_S^2\frac{\partial^2 U}{\partial S^2} + \mu_S\frac{\partial U}{\partial S} + \frac{1}{2}\sigma_V^2\frac{\partial^2 U}{\partial V^2} + \mu_V\frac{\partial U}{\partial V} + \rho\sigma_S\sigma_V\frac{\partial^2 U}{\partial S\partial V}$$

then

$$dU = L(U)dt + \mu_S\frac{\partial U}{\partial S}dW_t^1 + \mu_V\frac{\partial U}{\partial V}dW_t^2 \tag{4.4}$$

The following derivation is similar in method to the derivation of the PDE for Black-Scholes constant volatility model. We want to construct a portfolio Π to hedge the risk of U. Since (4.1) has two sources of randomness, in addition to including asset S, we need another asset U_1 whose payoff just depends on S in the portfolio Π in order to able to hedge U.

Let Π be a self financing portfolio consisting of one derivative U, $-\Delta$ share of S and $-\Delta_1$ share of U_1.

$$\Pi = U - \Delta S - \Delta_1 U_1 \tag{4.5}$$

The change in Π over a small amount of time dt is:

$$\begin{aligned} d\Pi &= dU - \Delta dS - \Delta_1 dU_1 \\ &= \left[L(U)dt + \mu_S\frac{\partial U}{\partial S}dW_t^1 + \mu_V\frac{\partial U}{\partial V}dW_t^2\right] \\ &\quad - \Delta\left[\mu_S dt + \sigma_S dW_t^1\right] \\ &\quad - \Delta_1\left[L(U_1)dt + \mu_S\frac{\partial U_1}{\partial S}dW_t^1 + \mu_V\frac{\partial U_1}{\partial V}dW_t^2\right] \end{aligned} \tag{4.6}$$

Collecting all the terms with respect to dt, dS and $dV2$ results in :

$$\begin{aligned} d\Pi = & \left[L(U) - \Delta\mu_S - \Delta_1 L(U_1)\right] dt \\ & + \left[\mu_S \frac{\partial U}{\partial S} - \Delta\mu_S - \Delta_1 \mu_S \frac{\partial U_1}{\partial S}\right] dW_t^1 \\ & + \left[\mu_V \frac{\partial U}{\partial V} - \Delta_1 \mu_V \frac{\partial U_1}{\partial V}\right] dW_t^2 \end{aligned} \tag{4.7}$$

To get rid of the randomness of $d\Pi$, Δ_1 and Δ_2 should satisfy the following system:

$$\begin{aligned} \mu_S \frac{\partial U}{\partial S} - \Delta\mu_S - \Delta_1 \mu_S \frac{\partial U_1}{\partial S} = 0 \\ \mu_V \frac{\partial U}{\partial V} - \Delta_1 \mu_V \frac{\partial U_1}{\partial V} = 0 \end{aligned}$$

Solving this yields:

$$\begin{aligned} \Delta_1 &= \left(\frac{\partial U}{\partial V}\right) / \left(\frac{\partial U_1}{\partial V}\right) \\ \Delta &= \frac{\partial U}{\partial S} - \frac{\partial U_1}{\partial S}\left(\frac{\partial U}{\partial V}\right) / \left(\frac{\partial U_1}{\partial V}\right) \end{aligned} \tag{4.8}$$

Using the above hedge strategy, Π is a risk less portfolio with

$$d\Pi = \left[L(U) - \Delta_1 L(U_1)\right] dt$$

To be arbitrage free,

$$d\Pi = r\Pi dt$$

where r is the risk free interest rate. So we have

$$r\Pi = L(U) - \Delta_1 L(U_1)$$

Inserting (4.5), (4.8), and $\mu_S = rS$ into the above equation and collecting all the terms with respect to U and U_1 results in:

$$\frac{L(U) - rU}{\frac{\partial U}{\partial V}} = \frac{L(U_1) - rU_1}{\frac{\partial U_1}{\partial V}}$$

Since this equation satisfies any choices of U_1 and U, both the left hand side and the right hand side should equal to a function $\lambda(S, V, t)$ independent of U and U_1. Setting λ equal to 0 results in :

$$\frac{\partial U}{\partial t} + \frac{1}{2}\sigma_S^2 \frac{\partial^2 U}{\partial S^2} + \mu_S \frac{\partial U}{\partial S} + \frac{1}{2}\sigma_V^2 \frac{\partial^2 U}{\partial V^2} + \mu_V \frac{\partial U}{\partial V} + \rho\sigma_S\sigma_V \frac{\partial^2 U}{\partial S \partial V} - rU = 0 \tag{4.9}$$

This is a two dimensional convection-diffusion equation with mixed partial derivatives. Together with initial and boundary conditions, the PDE can be solved numerically. For some special choices of μ_S, μ_V, σ_S, σ_V, an analytical solution of U can be found. For example, the popular Heston [40] and SABR models [37] which will be introduced in the following sections.

4.2.2 Heston PDE

The Heston model [40] assumes following dynamics for the risky asset S

$$dS_t = rS_t dt + \sqrt{V_t} S_t dW_t^1 \tag{4.10}$$
$$dV_t = \kappa(\theta - V_t)dt + \sigma\sqrt{V_t} dW_t^2 \tag{4.11}$$
$$dW_t^1 dW_t^2 = \rho dt \tag{4.12}$$

where κ is the mean reversion rate, θ is the long run variance, σ is the volatility of variance, ρ is the correlation parameter. Compared to (4.1), we have

$$\begin{aligned}
\mu_S &= rS_t \\
\sigma_S &= \sqrt{V_t} S_t \\
\mu_V &= \kappa(\theta - V_t) \\
\sigma_V &= \sigma\sqrt{V_t}
\end{aligned}$$

Inserting them into (4.9) and ignoring the subscript t, we obtain the PDE for U under the Heston model,

$$\frac{\partial U}{\partial t} + \frac{1}{2}S^2 V \frac{\partial^2 U}{\partial S^2} + rS\frac{\partial U}{\partial S} + \frac{1}{2}\sigma^2 V \frac{\partial^2 U}{\partial V^2} + \kappa(\theta - V)\frac{\partial U}{\partial V} + \rho\sigma SV\frac{\partial^2 U}{\partial S \partial V} - rU = 0 \tag{4.13}$$

The above derivation assumes the asset does not pay any dividends. When the dividends are not zero, the above equation needs to be modified by replacing r with $r - r_f$, where r_f is the continuous dividend rate. In order for the variance to be positive, the Feller condition

$$2\kappa\theta > \sigma^2 \tag{4.14}$$

should be satisfied.

4.2.3 SABR PDE

The SABR model was introduced by [37] and and its acronym stands for Stochastic Alpha Beta Rho. It is defined on the forward price of asset. The model assumes the following form:

$$\begin{aligned}
df_t &= \alpha_t f_t^\beta dW_t^1 \\
d\alpha_t &= \sigma\alpha_t dW_t^2 \\
dW_t^1 dW_t^2 &= \rho
\end{aligned} \tag{4.15}$$

where f_t is the forward price. α is used to denote the variance. In order to change the above process with respect to f_t into a process about spot price S_t so that we may use the generic equation (4.9), we make use of the following relationship between f_t and S_t:

$$S_t = f_t D(t,T) = e^{-r(T-t)} f_t$$

Using Ito calculus together with (4.15) we obtain:

$$\begin{aligned} dS_t &= rS_t dt + D^{1-\beta}\alpha_t S_t^{\beta} dW_t^1 \\ d\alpha_t &= \sigma\alpha_t dW_t^2 \\ dW_t^1 dW_t^2 &= \rho dt \end{aligned} \tag{4.16}$$

Following the same argument as for the Heston model, we have

$$\begin{aligned} \mu_S &= rS_t \\ \sigma_S &= D^{1-\beta}\alpha_t S_t^{\beta} \\ \mu_V &= 0 \\ \sigma_V &= \sigma\alpha_t \end{aligned}$$

Substituting them into the generic PDE (4.9) and noting V is referred to as α, the PDE of derivative U for SABR model is:

$$\frac{\partial U}{\partial t} + \frac{1}{2}D^{2(1-\beta)}\alpha^2 S^{2\beta}\frac{\partial^2 U}{\partial S^2} + rS\frac{\partial U}{\partial S} + \frac{1}{2}\sigma^2\alpha^2\frac{\partial^2 U}{\partial \alpha^2} + \rho\sigma D^{1-\beta}\alpha^2 S^{\beta}\frac{\partial^2 U}{\partial S \partial \alpha} - rU = 0 \tag{4.17}$$

4.3 An alternating direction implicit (ADI) approach

The Heston and SABR models are usually calibrated with respect to liquid vanilla options and then are used to price exotic derivatives. The numerical cost of calibrating the Heston or SABR models depends critically on the computational cost and numerical performance of solving the forward pricing model (4.13) and (4.17), since the calibration process usually involves solving the forward model for a number of different parameter set. Consequently, the faster and less computational cost demanding is the forward model, the faster and less expensive would be the calibration process. In this section, we'll introduce an alternation direction (ADI) method to solve PDE (4.13) and (4.17) using non-uniform grids.

We'll use the Heston PDE (4.13) as an example. The ADI method introduced can also be extended to the SABR PDE (4.17). The ADI method is a widely-used approach for solving PDEs in higher dimensions. By solving the PDE implicitly along one spatial dimension at each time, it significantly reduces the computational cost by only requiring the inversion of a small dimension fixed-width banded matrix at each time step instead of the inversion of a large dimension sparse matrix, details of which will be explained in the following. They are several ADI schemes for solving the 2D Heston PDE (4.13), we will adopt the *Craig-Sneyd scheme*. The following derivation including notations follows closely the paper by [45].

For succinctness, we change the capital case characters in (4.13) into their low case. So the Heston PDE is:

$$\frac{\partial u}{\partial t} + \frac{1}{2}s^2 v\frac{\partial^2 u}{\partial s^2} + rs\frac{\partial u}{\partial s} + \frac{1}{2}\sigma^2 v\frac{\partial^2 u}{\partial v^2} + \kappa(\theta - v)\frac{\partial u}{\partial v} + \rho\sigma sv\frac{\partial^2 u}{\partial s \partial v} - ru = 0$$

After a change the variable $t = T - \tau$, where T is the maturity date of a derivative, this PDE becomes an initial boundary value problem. Let t denote τ and we consider the general case when the asset pays dividend, the above PDE becomes

$$\frac{\partial u}{\partial t} = \frac{1}{2}s^2 v\frac{\partial^2 u}{\partial s^2} + (r - r_f)s\frac{\partial u}{\partial s} + \frac{1}{2}\sigma^2 v\frac{\partial^2 u}{\partial v^2} + \kappa(\theta - v)\frac{\partial u}{\partial v} + \rho\sigma sv\frac{\partial^2 u}{\partial s \partial v} - (r - r_f)u \quad (4.18)$$

where r_f is the continuous dividend rate.

The initial value of the above PDE is the payoff of the derivative. For a European call option, it is:

$$u(s, v, 0) = max(0, s - K) \quad (4.19)$$

where K is the strike of the European option.

At $s = 0$, the boundary condition is

$$u(0, v, t) = 0 \quad (4.20)$$

At $v = \infty$, the boundary condition is

$$u(s, \infty, t) = se^{-r_f t} \qquad 0 \le t \le T \quad (4.21)$$

At $S = \infty$, the boundary condition is

$$\frac{\partial u(\infty, v, t)}{\partial s} = e^{r_f t} \qquad 0 \le t \le T \quad (4.22)$$

At $v = 0$, no condition is specified. It will be solved using the Heston PDE (4.18).

To solve (4.18)-(4.22) numerically, we first restrict the spatial domain to a bounded set $[0, s_{max}] \times [0, v_{max}]$, where $s_{max} = 8K$ and $v_{max} = 5$. "This yields a negligible modeling error compared to the unbounded spatial domain for a wide range of parameter values" [45]. In order to use a small number of mesh points, yet achieve high accuracy when solving (4.18), we adopt a non-uniform mesh grid. To select the grid points along the s and v directions, the variables s, v are usually transformed into another two variables, which will be discretized uniformly in the new space, and then the discretized mesh points in the new space will be transformed back to get the mesh points along s and v direction. We employ the non-uniform mesh as in [64].

In the s direction, the transform is defined by : $\sinh(\xi) = (s - K)/c$, where $c > 0$ is a constant. We choose $c = K/5$. Since $0 \le s \le s_{max}$, $-K/c \le \xi \le (s_{max} - K)/c$. Assuming that along the s direction, the grid points are $0 = s_0 < s_1 < \cdots < s_m = s_{max}$, where $m > 0$ is an integer, then s_i is defined according to the following transformation:

$$s_i = K + c\,\sinh(\xi_i) \quad (0 \le i \le m) \quad (4.23)$$

where

$$\begin{aligned} \xi_i &= \xi_0 + i * \Delta\xi \quad (0 \le i \le m) \\ \Delta\xi &= (\xi_{max} - \xi_0)/m \\ \xi_0 &= \sinh^{-1}(-K/c) \\ \xi_{max} &= \sinh^{-1}((s_{max} - K)/c) \end{aligned}$$

In the v direction, the transform is : $\sinh(\eta) = v/d$, where $d > 0$ is a constant. We choose $d = v_{max}/500$. Let $0 = v_0 < v_1 < v_2 < \cdots < v_n = v_{max}$ be the mesh points along the v direction, where $n > 0$ is an integer, then v_i is defined from the following transformation:

$$v_i = \sinh(\eta_i) \quad (0 \le i \le n) \tag{4.24}$$

where

$$\eta_i = i\Delta\eta \quad (0 \le i \le n)$$
$$\Delta\eta = \frac{1}{n}\sinh^{-1}(v_{max}/d)$$

The spatial grid defined by (4.23) and (4.24) assigns relatively more points close to $s = K$ and $v = 0$ respectively. The parameter c controls the number of grid points around $s = K$ since

$$\Delta s_i \approx c\ \Delta\xi \text{ whenever } s_i \approx K$$

with $\Delta s_i = s_i - s_{i-1}$. Similarly, the parameter d controls the number of grid points around $v = 0$ since

$$\Delta v_i \approx d\ \Delta\eta \text{ whenever } v_i \approx 0$$

with $\Delta v_i = v_i - v_{i-1}$. Figure 4.1 shows the grids when $m = 30$ and $n = 30$.

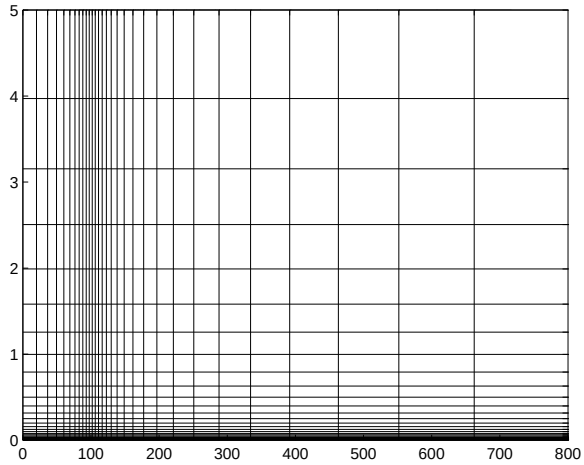

Figure 4.1: Sample grid defined by (4.23) and (4.24) when $m = n = 30$, $K = 100$.

To use a finite difference scheme to solve (4.18), we need to approximate the spatial derivatives on the right hand side of the equation with a non-uniform mesh. In each spatial direction, there is one first order partial derivative and one second order partial derivative. There is also a mixed second order partial derivative when $\rho \neq 0$. Regardless of the spatial direction s or v, the following scheme can be used to approximate the partial derivatives.

Let $f : \mathbb{R} \to \mathbb{R}$ be a smooth function, let $x_0 < x_1 < x_2 \cdots < x_m$ be a set of discrete points, $\Delta x_i = x_i - x_{i-1}$, then $f'(x_i)$ can be approximated using any of the following finite difference scheme:

$$f'(x_i) \approx \alpha_{i,-2} f(x_{i-2}) + \alpha_{i,-1} f(x_{i-1}) + \alpha_{i,0} f(x_i) \tag{4.25}$$
$$f'(x_i) \approx \beta_{i,-1} f(x_{i-1}) + \beta_{i,0} f(x_i) + \beta_{i,1} f(x_{i+1}) \tag{4.26}$$
$$f'(x_i) \approx \gamma_{i,0} f(x_i) + \gamma_{i,1} f(x_{i+1}) + \gamma_{i,2} f(x_{i+2}) \tag{4.27}$$

where

$$\alpha_{i,-2} = \frac{\Delta x_i}{\Delta x_{i-1}(\Delta x_{i-1} + \Delta x_i)}, \qquad \alpha_{i,-1} = \frac{-\Delta x_{i-1} - \Delta x_i}{\Delta x_{i-1}\Delta x_i}, \quad \alpha_{i,0} = \frac{\Delta x_{i-1} + 2\Delta x_i}{\Delta x_i(\Delta x_{i-1} + \Delta x_i)}$$
$$\beta_{i,-1} = \frac{-\Delta x_{i+1}}{\Delta x_i(\Delta x_i + \Delta x_{i+1})}, \qquad \beta_{i,0} = \frac{\Delta x_{i+1} - \Delta x_i}{\Delta x_i \Delta x_{i+1}}, \qquad \beta_{i,1} = \frac{\Delta x_i}{\Delta x_{i+1}(\Delta x_i + \Delta x_{i+1})}$$
$$\gamma_{i,0} = \frac{-2\Delta x_{i+1} - \Delta x_{i+2}}{\Delta x_{i+1}(\Delta x_{i+1} + \Delta x_{i+2})}, \qquad \gamma_{i,1} = \frac{\Delta x_{i+1} + \Delta x_{i+2}}{\Delta x_{i+1}\Delta x_{i+2}}, \quad \gamma_{i,2} = \frac{-\Delta x_{i+1}}{\Delta x_{i+2}(\Delta x_{i+1} + \Delta x_{i+2})}$$

$f''(x_i)$ can be approximated by the following scheme:

$$f''(x_i) \approx \delta_{i,-1} f(x_{i-1}) + \delta_{i,0} f(x_i) + \delta_{i,1} f(x_{i+1}) \tag{4.28}$$

where

$$\delta_{i,-1} = \frac{2}{\Delta x_i(\Delta x_i + \Delta x_{i+1})}, \qquad \delta_{i,0} = \frac{-2}{\Delta x_i \Delta x_{i+1}}, \qquad \delta_{i,1} = \frac{2}{\Delta x_{i+1}(\Delta x_i + \Delta x_{i+1})}$$

Schemes (4.25-4.28) can be proved to be second order accurate provided f is a smooth function and the mesh consisting of x_i is smooth. Both (4.26) and (4.28) are *centered* schemes. (4.25) and (4.27) are *upwind* schemes.

Let $f : \mathbb{R}^2 \to \mathbb{R}$ be a function of two variables x and y, let $y_0 < y_1 < \cdots < y_n$ be a set of discrete mesh points along the y direction with $\Delta y_j = y_j - y_{j-1}$. x_i is still as defined above. $f_{xy}(x_i, y_j)$ can be approximated using the following finite difference scheme:

$$\frac{\partial^2 f}{\partial x y}(x_i, y_j) \approx \sum_{k,\ l=-1}^{1} \beta_{i,\ k} \hat{\beta}_{j,\ l} f(x_{i+k}, y_{j+l}) \tag{4.29}$$

where $\hat{\beta}_{j,l}$ is defined analogous to the coefficients β in (4.26) but relevant to the y direction. (4.29) can also be proved to be a second order accurate scheme provided f and the mesh grids along x and y direction satisfy some smoothness conditions.

After the spatial finite difference approximation and including the boundary conditions, the initial boundary value problem (4.18)-(4.22) is transformed into an ordinary initial value problem:

$$\frac{dU}{dt} = \mathbf{A}U + b, \qquad U(0) = U_0 \tag{4.30}$$

where U is a vector of dimension $p = m \times n$, $\mathbf{A}$ is a matrix of dimension $p \times p$. b is a vector of dimension p. U_0 and b are obtained from the initial condition of (4.19) and the boundary conditions (4.20)-(4.22) respectively.

A direct solution of the above initial boundary value problem can be summarized by the following implicit-explicit(IMEX) scheme:

$$\frac{U^{k+1} - U^k}{\Delta t} = \theta(\mathbf{A}^{n+1}U^{n+1} + b^{n+1}) + (1-\theta)(\mathbf{A}^n U^n + b^n) \tag{4.31}$$

(4.31) is also called θ-method. When $\theta = 0$, it is the forward Euler scheme; when $\theta = 1$, it corresponds to the backward Euler scheme; when $\theta = 1/2$, it is the Crank-Nicolson(CN) scheme. The forward Euler scheme is an explicit scheme and it does not involve any matrix inversion when computing the solution at the next time step. However, it is not stable. When $\theta \neq 0$, the θ-method is an implicit scheme. It is stable. However, solving it requires the inversion of the matrix $\mathbf{I} - \theta\Delta t\mathbf{A}$ of size $p = m \times n$ at each time step, which can be computationally demanding. The reason for this is the matrix $\mathbf{I} - \theta\Delta t\mathbf{A}$, and hence the matrices of LU decomposition, possesses a bandwidth proportional to $\min(m, n)$ [45]. In addition, if the mesh grid is refined by a factor of 10 in each direction, the dimension of the matrix $\mathbf{I} - \theta\Delta t\mathbf{A}$ will increase 100 times.

To improve the efficiency, the ADI approach, which stands for alternating direction implicit, is introduced. It decomposes the spatial operator $\mathbf{A}$ into several sub-spatial operators, each of which contains only the spatial derivatives along one direction, and treats only one sub-spatial operator implicitly at one time while keeping other sub-spatial operators explicit. By doing this, it reduces the problem from a 2-D problem into a 1-D problem during the implicit step, for which the matrix inversion is proportional to a fixed band width independent of the dimension of the problem, see [29].

$\mathbf{A}$ is split into three matrices:

$$\mathbf{A} = \mathbf{A}_0 + \mathbf{A}_1 + \mathbf{A_2}$$

where $\mathbf{A}_0$ represents the mixed spatial derivative operator, $\mathbf{A}_1$ denotes the spatial operator in the s direction, and $\mathbf{A}_2$ denotes the spatial operator in the v direction. The ru term is distributed equally into $\mathbf{A}_1$ and $\mathbf{A}_2$. Accordingly, we split b into three vectors in the same way as for $\mathbf{A}$:

$$b = b_0 + b_1 + b_2$$

where b_0, b_1, b_2 correspond to the mixed spatial component, the s component, and the v component respectively. We can then rewrite (4.30) as follows:

$$\frac{dU}{dt} = \sum_{i=0}^{2} \mathbf{A}_i U + b_i$$

For simplicity of notation, define $F(t, U) = \mathbf{A}Y + b(t)$, where $U \in \mathbb{R}^p,\ b \in \mathbb{R}^n$, and $\mathbf{A}$ is as defined above. Define $F_j(t, U) = \mathbf{A}_j U + b_j$, where $U \in \mathbb{R}^p,\ b_j \in \mathbb{R}^n$, and $\mathbf{A}_j$ is as defined above. There are several different versions of ADI schemes for solving the PDE (4.18), such as the *Douglas(Do) scheme* [26], *Craig-Sneyd(CS) scheme* [24], *Modified Craig-Sneyd(MCS)*

scheme, and the *Hundsorfer-Verwer(HV) scheme* [43]. Among them, the *Douglas scheme* is simplest as listed in the following.

Douglas(Do) scheme

$$\begin{aligned} Y_0 &= U_{n-1} + \Delta t F(t_{n-1}, U_{n-1}) \\ Y_j &= Y_{j-1} + \theta \Delta t (F_j(t_n, Y_j) - F_j(t_{n-1}, U_{n-1})) \quad (j = 1, 2) \\ U_n &= \tilde{Y}_2 \end{aligned} \tag{4.32}$$

where θ is a parameter that controls the degree of implicity. For the Douglas scheme, at each step, it first computes an estimate of the solution using an explicit scheme, then the estimate is corrected in two consecutive substeps by solving the PDE implicitly along only one spatial direction at each substep.

Craig-Sneyd(CS) scheme

$$\begin{aligned} Y_0 &= U_{n-1} + \Delta t F(t_{n-1}, U_{n-1}) \\ Y_j &= Y_{j-1} + \theta \Delta t (F_j(t_n, Y_j) - F_j(t_{n-1}, U_{n-1})) \quad (j = 1, 2) \\ \tilde{Y}_0 &= Y_0 + \frac{1}{2} \Delta t (F_0(t_n, Y_2) - F_0(t_{n-1}, U_{n-1})) \\ \tilde{Y}_j &= \tilde{Y}_{j-1} + \theta \Delta t (F_j(t_n, \tilde{Y}_j) - F_j(t_{n-1}, U_{n-1})) \quad (j = 1, 2) \\ U_n &= \tilde{Y}_2 \end{aligned} \tag{4.33}$$

The *Craig-Sneyd scheme* as listed above performs the "predictor-corrector" step twice. The first three steps of the CS scheme is the same as the Do scheme. After the first round of predictor-corrector operation, the CS scheme computes another estimate followed by two steps of corrections. In both of these two schemes, the mixed operator F_0 is treated explicitly. The computational savings result from the fixed band width of the matrix to be inverted during the correction step and that the band width is independent of m or n. The matrices to be inverted in the correction step corresponding to the spatial operators F_1 and F_2 are diagonal and pentadiagonal [52] respectively. As a result, the total floating point operations of the above two schemes is directly proportional to p, which is more efficient than an IMEX scheme.

For the Do scheme, the order of accuracy is 1 for any θ and for general F_0, F_1, and F_2. For the CS scheme, the order of accuracy is 2 when and only when $\theta = 1/2$ for general F_0, F_1, and F_2. When $\theta \geq 1/2$, the Do scheme is unconditional stable. The CS scheme is unconditionally stable for any θ. The MCS and HV schemes can attain second order of accuracy for any θ. For details, see [45]. In this thesis, we adopt the CS scheme by fixing $\theta = 1/2$.

4.4 Test of Craig-Sneyd(CS) scheme

To verify the correctness of the *Craig-Sneyd(CS) scheme*, we carried out the following numerical test. The parameters for the Heston model are $\kappa = 1.5$, $\theta = 0.04$, $\sigma = 0.3$, $\rho = -0.9$, $r = 0.025$, $r_f = 0$ and $s_0 = 100$. The parameters are the same as [2] and [45].

The European option has maturity $T = 1.0$ and strike $K = 100$. We are interested in the price of the option for the region defined in the following.

$$\{u(s,t) \mid \frac{1}{2}K < s < \frac{3}{2}K,\ 0.1 < v < 1.0\}$$

There are two reasons why we choose this region: first, most options in the market lie within this region, second, we use the non-uniform grid defined in (4.23) and (4.24) for computation. The absolute error between the computed option prices using the *CS scheme* and the analytical solution when $m = 200 = 2n = 200$ is displayed in Figure 4.2. For this specific configuration of the Heston model, the absolute error is within one cent.

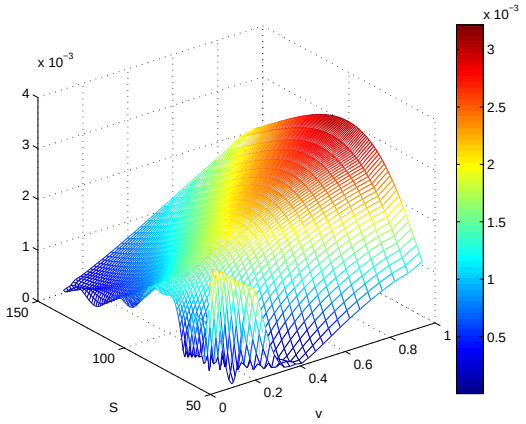

Figure 4.2: The absolute error between the computed option prices using *CS scheme* and the analytical solution

Next, we verify the correctness of *Craig-Sneyd(CS) scheme* by checking the order of convergence as in [45]. We use the maximum norm as a measure of the *global spatial discretization error* at time $t = T$:

$$e(m,n) = max\{|u(s_i,v_j,T) - U_k(T)| : \frac{1}{2}K < s_i < \frac{3}{2}K, 0.1 < v_j < 1.0\}$$

where $u(s_i,v_j,T)$ is the numerical solution of the Heston PDE (4.18-4.22) at grid point (s_i,v_j) when $t = T$, $U_k(T)$ is the corresponding true price of the option at the grid point (s_i,v_j) when $t = T$. Figure 4.3 exhibits the plot of *global spatial discretization error* as $m = 2n$ increases. The absolute value of the slope of the line fitted to the plot is 2.1, which suggests that the Heston PDE is quadratically convergent.

Similarly, we define the *global temporal discretization error* at time $t = T = nt\Delta t$ by

$$\hat{e}(nt;m,n) = max\{|u(s_i,v_j,T) - U_k(T)| : \frac{1}{2}K < s_i < \frac{3}{2}K, 0.1 < v_j < 1.0\}$$

where nt is an integer. Here m,n are the number of mesh points along the s and v direction. They are fixed in the test with $m = 2n = 100$. Again the maximum norm is used as a measure of the error. Figure 4.4 plots the *global temporal discretization error* as nt increases. The absolute value of the slope of the line fitted to the data is 1.9, which indicates a quadratic order of convergence in the time coordinate.

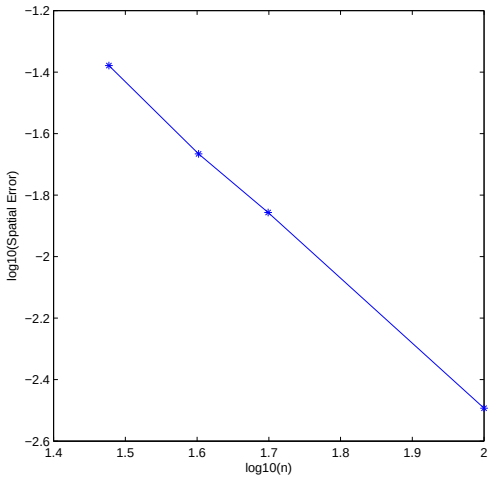

Figure 4.3: $\log_{10}(e(2n, n))$ versus $\log_{10}(n)$ as $n = 30, 40, 50, 100$

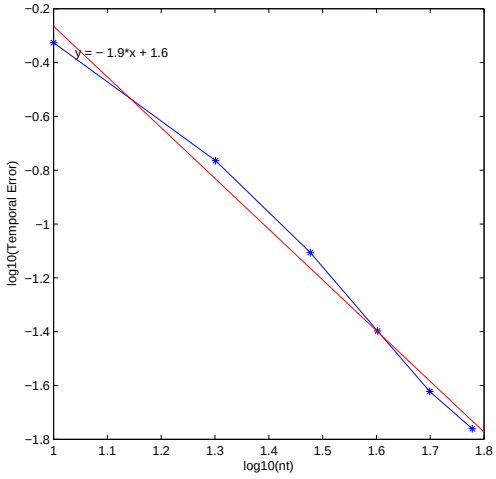

Figure 4.4: $\log_{10}(\hat{e}(nt))$ versus $\log_{10}(nt)$ as $nt = 10, 20, 30, 40, 50, 60$. Blue curve: plot of the data points. Red line: a line fitted to the data points

CHAPTER 5

PROPER ORTHOGONAL DECOMPOSITION (POD) REDUCED ORDER MODELING

5.1 Introduction

In practice, traders who use models to price financial instruments usually go through two processes: first, a model is calibrated to the market; second, with the parameters obtained from the first step, the model is used to price a financial instrument. In this way, the price of the instrument is considered to be consistent with the market. Traders especially high frequency traders, never stop seeking algorithms that can achieve the above two process as fast as possible. Accuracy, speed, cost are also constantly the three major issues in computational finance. Typically, given the parameters of a model to compute the output(price) of a model is called the forward problem. Calibration of a model is called the inverse problem. The inverse problem is much more complex compared to the forward problem and depends closely on the solution of the forward problem. The faster and less expensive of the forward problem, the higher the possibility of a faster and less expensive solution of the inverse problem. This is because the inverse problem usually involves solving the forward problem a number of times for different parameter sets. While the non-uniform grid ADI approach in the previous chapter is very efficient, we seek a method to speed up the process of solving the forward problem even more in this chapter. The method we will apply can be categorized in the framework of reduced order modeling. In particular, we will adopt the method Proper Orthogonal Decomposition (POD) reduced order modeling for our purposes.

Reduced order modeling has already found its applications in a number of research fields, such as meteorology, oceanography, electric circuit design and etc. It seeks to find a subspace to approximate the dynamics of a usually large dimension, nonlinear dynamic system, for which the computational cost and storage requirements are demanding. The necessity of reduced order modeling is even more obvious when it comes to the inverse problem of a large-scale, nonlinear dynamic problems.

The original idea of proper orthogonal decomposition can be traced back to [55]. It has been applied under other names such as Karhunen-Loève decomposition, principal component analysis, factor analysis, or total least-squares estimation. To project a high dimension dynamic system to a low dimension system, POD accomplishes this by first finding an orthogonal basis that spans the a given data set of the original high dimension system, then

retaining only several bases selected by a certain measure, and then forming a reduced model in the subspace spanned by the bases retained to approximate the dynamics of original full model. The number of bases retained constitutes the dimension of the reduced model. How well the dynamics of the original full model can be represented by the reduced model depends clearly on how well we choose the data set that is used to construct the orthogonal basis. Here we use the *method of snapshots* introduced by [62]to generate the data set. The full model is simulated up to a certain time T, during which a number of snapshots of the dynamic system is taken and collected as the data to represent the dynamics of the full model. Each snapshot contains the spatial information of the system at a certain time $t \in [0, T]$.

As shown in the following, the method of POD reduced order modeling requires only standard matrix operations. Compared to the finite element method or Fourier transform method, the POD reduced order modeling selects a basis that is relevant to the dynamics of a system. By doing this, it is more likely to find a small number of bases or features that best describe the dynamics of a system. In the following, we will introduce in detail the method of POD reduced order modeling with an application for the Heston model. Application to the SABR model can be derived similarly. The description of the POD is analogous to [57] and [41].

5.2 Proper orthogonal decomposition (POD)

5.2.1 Construction of POD basis

We will limit our consideration for only finite dimension dynamic system. For infinite dimensional problems, we assume it will be truncated and solved as a finite dimension problem, as for the Heston model (4.18). Let us assume there is a data set describing the dynamic system.

$$\mathbf{Y} = [y_1, y_2, \cdots, y_m]$$

where $y_i \in \mathbb{R}^n$, $1 \leq i \leq m$. n is the dimension of the dynamic system and it typically equals to the number of mesh points. m is the number of snapshots. We want to find a subspace of fixed rank d so that when $\mathbf{Y}$ is projected into this subspace, the residual is minimized, i.e.,

$$\min_{\{\phi_1, \phi_2, \cdots, \phi_d\}} \sum_{i=1}^{m} \parallel y_i - \Pi_d y_i \parallel^2 = \sum_{i=1}^{m} \parallel y_i - \sum_{j=1}^{j=d} \alpha_j \phi_j \parallel^2 \tag{5.1}$$

where $\| \cdot \|^2$ is the L_2 norm,

$$\Pi_d y_i = \sum_{j=1}^{j=d} \alpha_j \phi_j$$

is the projection of y_i into the subspace spanned by the orthonormal basis $[\phi_1, \phi_2, \cdots, \phi_d]$, where $\phi_j \in \mathbb{R}^n$, $j = 1, 2, \cdots, d$. $\alpha_j =< y_i, \phi_j >$, where $< \cdot, \cdot >$ is an inner product defined for the vector space of $\mathbb{R}^n$.

The solution of above problem turns out to be directly related to the solution of the following eigenvector problem:

$$\mathbf{K} u_j = \lambda_j u_j \qquad j = 1, 2, \cdots, n \tag{5.2}$$

and

$$< u_i, u_j >= \begin{cases} 0 & \text{if} \quad i \neq j \\ 1 & \text{if} \quad i = j \end{cases} \tag{5.3}$$

where $\mathbf{K}$ is the covariance matrix given by $\mathbf{K} = \mathbf{Y}\mathbf{Y}^T$. Since $\mathbf{K}$ is always a positive semidefinite square matrix, the eigenvalues λ_j satisfy: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$. u_j is the eigenvector of $\mathbf{K}$ corresponding to eigenvalue λ_j.

The optimal subspace is the space spanned by bases $u_1, u_2, \cdots, u_d$, which we denote as V_d. The vectors $u_j,\ j = 1, 2, \cdots, d$ are the bases ϕ_i to find and are called POD modes. Furthermore, the residual of the projection equals the sum of the remaining eigenvalues of $\mathbf{K}$ [57] :

$$\min_{\{\phi_1, \phi_2, \cdots, \phi_d\}} \sum_{i=1}^{m} \parallel y_i - \Pi_d y_i \parallel^2 = \sum_{i=n+d-1}^{n} \lambda_i \tag{5.4}$$

Once the POD basis have been found, we seek to simulate the full model by constructing a reduced model in the subspace of V_d.

$$y^{\text{POD}} = \sum_{i=1}^{d} \alpha_i(t)\phi_i \tag{5.5}$$

where $\alpha_i(t)$ is to be found and is utilized to simulate the time evolution of the dynamic system.

The reduced model can be also formulated by removing the mean of data first and then projecting the remaining data into a subspace. The reduced model then has the form of

$$y^{\text{POD}} = \bar{y} + \sum_{i=1}^{d} \alpha_i(t)\phi_i \tag{5.6}$$

where $\bar{y} = \sum_{i=1}^{m} y_i$, ϕ_i is the normalized eigenvector of the matrix $\mathbf{K} = \mathbf{Y}\mathbf{Y}^T$, and $\mathbf{Y} = [y_1 - \bar{y}, y_2 - \bar{y}, \cdots, y_m - \bar{y}]$. This form applies to cases when the mean of y is not the major interest, for example, the Karman's vortex street, which is a superposition of a mean flow and a vortex structure, or cases when the boundary conditions are not homogeneous in terms of the spatial variables.

5.2.2 Choosing the dimension

One important question for POD is how to choose the dimension d of the subspace? Equation (5.4) gives us some guidance for this question. Since the eigenvalues λ_i are sorted in a decreasing order, we can choose d such that the sum of the remaining eigenvalues is close to zero. It makes sense since in practice large singular values correspond to major modes of a dynamic system while small singular values correspond to small perturbations of a dynamic system. We define the relative information content as follows.

$$I(d) = \frac{\sum_{i=1}^{d} \lambda_i}{\sum_{i=1}^{n} \lambda_i} \tag{5.7}$$

The goal is to choose d so that $I(d)$ is close to 1 to keep the most information of the data. This is achieved by fixing a percentage γ close to 1 to represent the amount of information to retain, then choose d such that

$$d = \arg\min \{I(d) : I(d) > \gamma\}$$

5.2.3 Method of snapshots

The size of the matrix $\mathbf{K}$ is $n \times n$, where n is the dimension of a dynamic system. If the dimension n is really large, for example, of the order of 10^6, a direct eigenvalue decomposition of the matrix $\mathbf{K}$ to compute the POD basis is not effective any more. In cases like this, the number of snapshots m is usually less than n. We still have data

$$\mathbf{Y} = [y_1 - \bar{y}, y_2 - \bar{y}, \cdots, y_m - \bar{y}]$$

where $\bar{y} = \sum_{i=1}^{m} y_i$ is still the average of the ensemble of snapshots.

In the method of snapshots, instead of computing the eigenvectors directly of the covariance matrix $\mathbf{K} = \mathbf{Y}\mathbf{Y}^T$, we first solve the eigenvectors of matrix $\tilde{\mathbf{K}} = \mathbf{Y}^T\mathbf{Y}$, which is of size $m \times m$. m is usually much less than n. It is easy to compute the eigenvectors and eigenvalues λ_j of $\tilde{\mathbf{K}}$.

$$\mathbf{Y}^T\mathbf{Y}\nu_j = \lambda_j\nu_j \qquad j = 1, 2, \cdots, m \tag{5.8}$$

Then the POD basis ϕ_i can be computed in the following way:

$$\phi_i = \frac{1}{\sqrt{\lambda_i}}\mathbf{Y}\nu_j \qquad j = 1, 2, \cdots, m \tag{5.9}$$

The reason why we can use (5.9) to compute POD modes ϕ_i is detailed in the following section, which is based singular value decomposition(SVD).

5.2.4 POD and singular value decomposition (SVD)

Equation (5.9) can be easily proved by using singular value decomposition(SVD) since SVD can be applied to any rectangular matrix while the eigenvalue decomposition just applies to square matrices. For a general matrix of the $\mathbf{Y} \in \mathbb{R}^{n\times m}$ with rank d, the singular value decomposition of $\mathbf{Y}$ is in the form of:

$$\begin{aligned} \mathbf{Y}_{nm} &= \mathbf{U}_{nn}\mathbf{S}_{nm}\mathbf{V}_{mm}^T \\ &= [\mathbf{U}_d, \mathbf{U}_0]\begin{bmatrix}\mathbf{S}_d & 0\\ 0 & 0\end{bmatrix}[\mathbf{V}_d, \mathbf{V}_0]^T \\ &= \mathbf{U}_d\mathbf{S}_d\mathbf{V}_d^T \end{aligned} \tag{5.10}$$

where $\mathbf{U}_{nn}$ and $\mathbf{V}_{mm}$ are all orthonormal matrices. $\mathbf{S}$ is a diagonal matrix with only d nonzero diagonal entries. $\mathbf{S}_d = \text{diag}(\lambda_1, \lambda_2, \cdots, \lambda_d)$ with $\lambda_1 \geq \lambda_2 \geq \cdots \lambda_d > 0$.

From (5.10), we can obtain the following by matrix multiplication:

$$\mathbf{K} = \mathbf{Y}\mathbf{Y}^T = \mathbf{U}\mathbf{S}\mathbf{V}^T\mathbf{V}\mathbf{S}^T\mathbf{U} = \mathbf{U}\mathbf{S}^2\mathbf{U}^T$$

where $\mathbf{S}^2 = \text{diag}(\lambda_1^2, \lambda_2^2, \cdots, \lambda_d^2)$. For simplicity, we have also dropped the subscript of d for all the $\mathbf{U}, \mathbf{S}, \mathbf{V}$ terms. Multiplying $\mathbf{U}$ to both side of the above equation, we obtain:

$$\mathbf{KU} = \mathbf{S}^2\mathbf{U} \tag{5.11}$$

From the above equation, we can deduce that the matrix $\mathbf{K}$ has only d nonzero eigenvalues, which is the square of the λ_i defined in (5.10). The eigenvectors of $\mathbf{K}$ or the POD basis are just the columns of $\mathbf{U}$.

Similarly, we can obtain

$$\tilde{\mathbf{K}} = \mathbf{Y}^T\mathbf{Y} = \mathbf{V}^T\mathbf{S}\mathbf{U}\mathbf{U}^T\mathbf{S}\mathbf{V} = \mathbf{V}^T\mathbf{S}^2\mathbf{V}$$

thus

$$\tilde{\mathbf{K}}\mathbf{V}^T = \mathbf{S}^2\mathbf{V}^T$$

So $\tilde{\mathbf{K}}$ has the same eigenvalues as $\mathbf{K}$, and its eigenvectors are the columns of matrix $\mathbf{V}^T$.

When $n \gg m$, i.e. the dimension of a system is much larger than the number of snapshots, it is computationally efficient to find the eigenvalues and eigenvectors of $\tilde{\mathbf{K}}$ first, then plug them into (5.10) to solve for the POD basis $\mathbf{U}$. Note the λ defined in (5.9) and (5.10) are different with the former being the square of the latter.

5.2.5 Galerkin projection

Assume a system given by,

$$\begin{aligned} \frac{\partial y}{\partial t} &= L(y, p, t) \\ \text{with} \quad y(0) &= y_0 \end{aligned} \tag{5.12}$$

where L is a spatial operator, y represents the multi-dimensional state variables, and p represents all the parameters. There are usually additional boundary conditions imposed. When discretized, $y \in \mathbb{R}^n$ represents the state variables for a mesh grid, where n is the number of mesh points; L would be an $n \times n$ matrix representing the discretized spatial operator. The boundary conditions are already considered in the discretized spatial operator. So (5.12) represents a discretized dynamic system. It is an initial value problem.

Reduced order modeling is to find a reduced model, as in (5.6), so that y^{POD} constructed in a small dimension subspace will approximate the full model y in higher dimension within tolerable error.

Substituting (5.6) into (5.12), we obtain:

$$\begin{aligned} \frac{\partial(\bar{y} + \sum_{i=1}^d \alpha_i(t)\phi_i)}{\partial t} &= L(\bar{y} + \sum_{i=1}^d \alpha_i(t)\phi_i, p, t) \\ \text{with} \quad \bar{y} + \sum_{i=1}^d \alpha_i(0)\phi_i &= y_0 \end{aligned} \tag{5.13}$$

In order to find the coefficients $\alpha_i(t)$, we use the Galerkin projection. Galerkin projection is to find an approximate solution y^{POD} so that the approximation error projected into the reduced space is zero, see [8].

The approximation error is:

$$Error_{appox} = \frac{\partial(\bar{y} + \sum_{i=1}^{d} \alpha_i(t)\phi_i)}{\partial t} - L(\bar{y} + \sum_{i=1}^{d} \alpha_i(t)\phi_i, p, t) \tag{5.14}$$

Galerkin projection requires:

$$< Error_{appox}, \phi_j >= 0 \qquad \text{for } j = 1, 2, \cdots, d \tag{5.15}$$

which is equivalent to

$$< \frac{\partial(\bar{y} + \sum_{i=1}^{d} \alpha_i(t)\phi_i)}{\partial t}, \phi_j >=< L(\bar{y} + \sum_{i=1}^{d} \alpha_i(t)\phi_i, p, t), \phi_j > \qquad \text{for } j = 1, 2, \cdots, d \tag{5.16}$$

Since $\bar{y}$ and the basis ϕ_i are independent of t, the above equation can be reduced to:

$$< \sum_{i=1}^{d} \frac{d\alpha_i(t)}{dt}\phi_i, \phi_j >=< L(\bar{y} + \sum_{i=1}^{d} \alpha_i(t)\phi_i, p, t), \phi_j > \qquad \text{for } j = 1, 2, \cdots, d \tag{5.17}$$

The linearity of the inner product results in:

$$\sum_{i=1}^{d} \frac{d\alpha_i(t)}{dt} < \phi_i, \phi_j >=< L(\bar{y} + \sum_{i=1}^{d} \alpha_i(t)\phi_i, p, t), \phi_j > \qquad \text{for } j = 1, 2, \cdots, d \tag{5.18}$$

ϕ_i being an orthonormal basis leads to:

$$\dot{\alpha}_i(t) =< L(\bar{y} + \sum_{i=1}^{d} \alpha_i(t)\phi_i, p, t), \phi_i > \tag{5.19}$$

where $\dot{\alpha}_i = \frac{d\alpha_i}{dt}$ for $i = 1, 2, \cdots, d$. In analogous to above derivation process, the initial condition of $\alpha_i(0)$ can be found by:

$$\alpha_i(0) =< y_0 - \bar{y}, \phi_i > \tag{5.20}$$

for $i = 1, 2, \cdots, d$.

In summary, the coefficients α_i of the POD reduced model in the form of (5.6) is governed by the following dynamics:

$$\begin{aligned} \dot{\alpha}_i(t) &=< L(\bar{y} + \sum_{i=1}^{d} \alpha_i(t)\phi_i, p, t), \phi_i > \\ \alpha_i(0) &=< y_0 - \bar{y}, \phi_i > \end{aligned} \tag{5.21}$$

for $i = 1, 2, \cdots, d$. This is a set of ordinary differential equations with a dimension of d.

5.3 Application to the Heston model

In this section, we test three scenarios: strong negative correlation, strong positive correlation, and no correlation. In all the three scenarios we assume $r_f = 0$. However, the POD method can be extended to the case when $r_f \neq 0$.

	Case I	Case II	Case III
κ	1.5	0.6067	2.0
θ	0.04	0.0707	0.2
σ	0.3	0.25	0.3
ρ	-0.9	0.7	0.0
r	0.025	0.05	0.03
T	1.0	3.0	1.0
K	100	100	100

All of the three cases satisfy the Feller condition (4.14). Case I has been used in Chapter 4 for the verifying the *CS* scheme. Case II is from [45] and [60], where we use σ =0.25. Case III represents the cases when $\rho = 0$. In all three cases, $m = 2n = 100$, and 40 snapshots distributed evenly in time are taken. Figure 5.1 shows the decay of eigenvalues defined in (5.2) for all three cases. We retain only the five modes corresponding the first five eigenvalues, i.e, we choose $d = 5$. This leads to the relative information content $I(d)$ defined in (5.7) to be more than 0.9999.

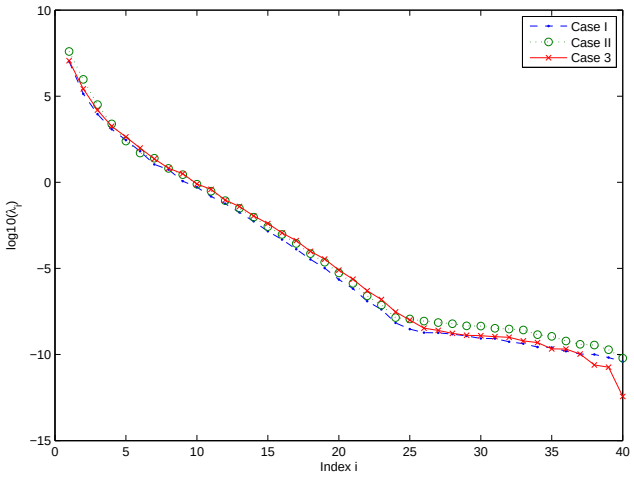

Figure 5.1: $\log_{10}(\lambda_i)$ where λ_i is the eigenvalues defined in (5.2)

Case I. Case I corresponds to the case when there is strong negative correlation. Figure 5.2 displays the solution $u(s,t)$ of the full model (4.18-4.22) for the region when $0 \leq s \leq 200$ and $0 \leq v \leq 1.0$. The price of u ranges from 0 to more than 100 depending on the initial asset price s and volatility v. Figure 5.3 shows the absolute errors of the option prices $u(s,t)$ between the full model and the POD reduced model (5.21) in this

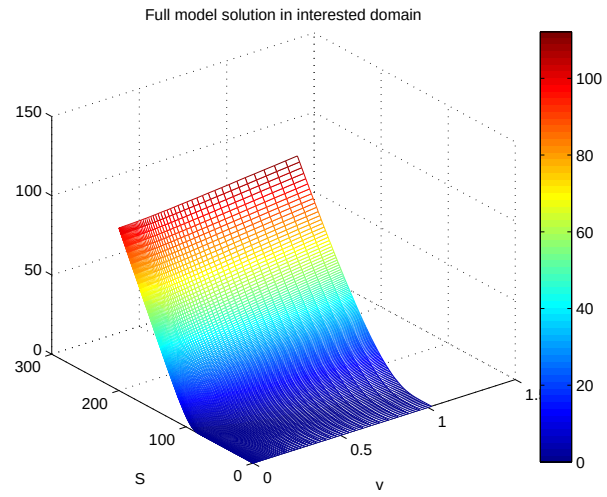

Figure 5.2: Full model solution of $\{u(s,t) \mid 0 \leq s \leq 200,\ 0 \leq v \leq 1\}$ for the parameter set in Case I.

region. The absolute errors are bounded by 10 cents for option prices ranging from $0 - 100$ dollars. Figure 5.4 exhibits the relative errors of $u(s,t)$ between the full model and the POD reduced model for the region when $70 \leq s \leq 100$ and $0.1 \leq v \leq 1.1$. We plot the relative errors only in this region since this region contains most of the interesting option prices. We can see that most of the relative errors are below 1%. Big relative errors occur at the lower left corner when both s and v are small, where the option prices are close zero and allow for higher tolerance of relative errors due to the bid and ask spreads. We do not plot the relative errors for $\{u(s,v) \mid 0 \leq s \leq 70, 0 \leq v \leq 0.1\}$ since the relative errors are high in this region and they will mask the relative errors in other regions if they are plotted. Furthermore, the option prices in this region are almost to zero, which we will not be interested in.

Case II. Case II corresponds to the case when there is strong positive correlation. Figure 5.5 displays the solution $u(s,t)$ of the full model (4.18-4.22) for the region when $0 \leq s \leq 200$ and $0 \leq v \leq 1.0$. The price of u ranges from 0 to more than 120 depending on the initial asset price s and volatility v. Figure 5.6 shows the absolute errors of the option prices $u(s,t)$ between the full model and the POD reduced model (5.21) in this region. The absolute errors are bounded by 40 cents for option prices ranging from $0 - 120$ dollars. Figure 5.7 exhibits the relative errors of $u(s,t)$ between the full model and the POD reduced model for the region when $70 \leq s \leq 100$ and $0.1 \leq v \leq 1.1$. We can observe that the relative errors are in the order of 0.1%. Big relative errors again happen at the lower left corner where both s and v are small, where the option prices are close zero and allow for higher tolerance of relative errors. For the same reason as in Case I, we do not plot the relative errors for $\{u(s,v) \mid 0 \leq s \leq 70, 0 \leq v \leq 0.1\}$.

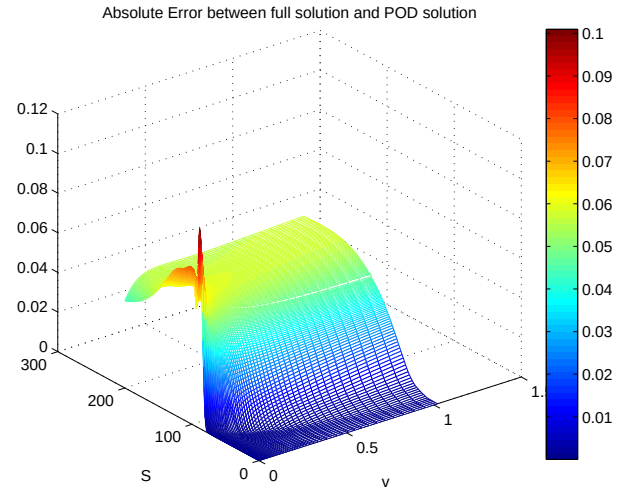

Figure 5.3: Absolute errors of the option prices $\{u(s,t) \mid 0 \leq s \leq 200, 0 \leq v \leq 1.0\}$ between the full model and POD reduced model for the parameter set in Case I.

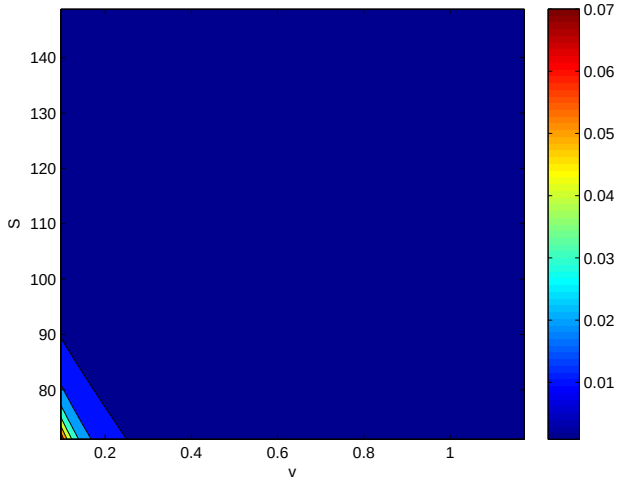

Figure 5.4: Relative errors of the option prices $\{u(s,t) \mid 70 \leq s \leq 150, 0.1 \leq v \leq 1.1\}$ between the full model and POD reduced model for the parameter set in Case I.

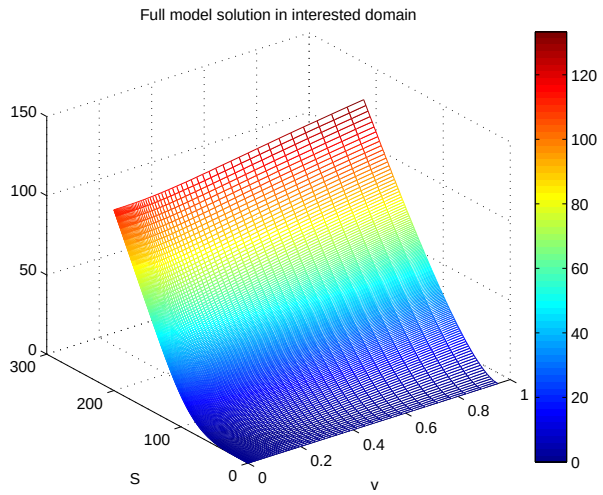

Figure 5.5: Full model solution of $\{u(s,t) \mid 0 \leq s \leq 200,\ 0 \leq v \leq 1\}$ for the parameter set in Case II.

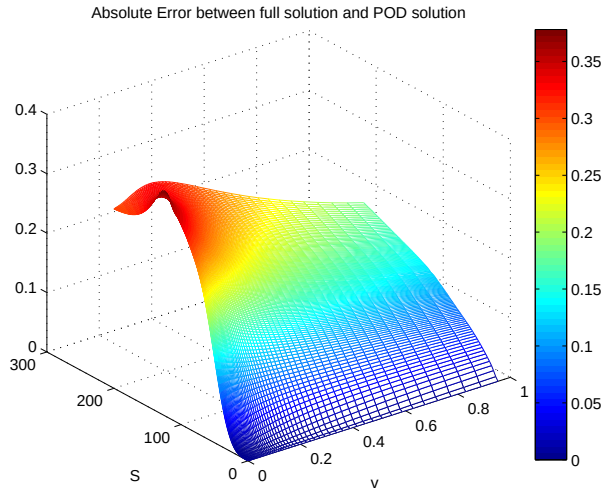

Figure 5.6: Absolute errors of the option prices $\{u(s,t) \mid 0 \leq s \leq 200, 0 \leq v \leq 1.0\}$ between the full model and POD reduced model for the parameter set in Case II.

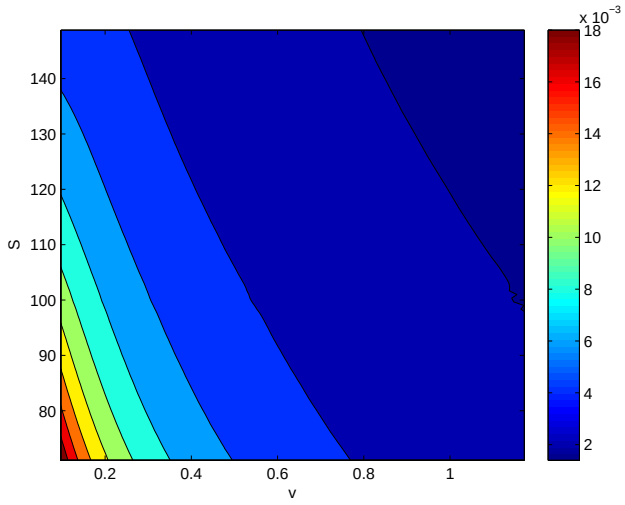

Figure 5.7: Relative errors of the option prices $\{u(s,t) \mid 70 \leq s \leq 150, 0.1 \leq v \leq 1.1\}$ between the full model and POD reduced model for the parameter set in Case II.

Case III. Case III corresponds to the case when there is no correlation. Figure 5.8 displays the solution $u(s,t)$ of the full model (4.18-4.22) for the region when $0 \leq s \leq 200$ and $0 \leq v \leq 1.0$. The price of u ranges from 0 to more than 100 depending on the initial asset price s and volatility v. Figure 5.9 shows the absolute errors of the option prices $u(s,t)$ between the full model and the POD reduced model (5.21) in this region. The absolute errors are bounded by 14 cents for option prices ranging from $0 - 100$ dollars. Figure 5.10 exhibits the relative errors of $u(s,t)$ between the full model and the POD reduced model for the region when $70 \leq s \leq 100$ and $0.1 \leq v \leq 1.1$. We plot the relative errors only in this region since this region contains most of the interested option prices. We can see that the relative errors are in the order of 0.1%. Big relative errors again happen at the lower left corner when both s and v are small, where the option prices are close zero and allow for higher tolerance of relative errors. For the same reason as Cases I and II, we do not plot the relative errors for $\{u(s,v) \mid 0 \leq s \leq 70, 0 \leq v \leq 0.1\}$

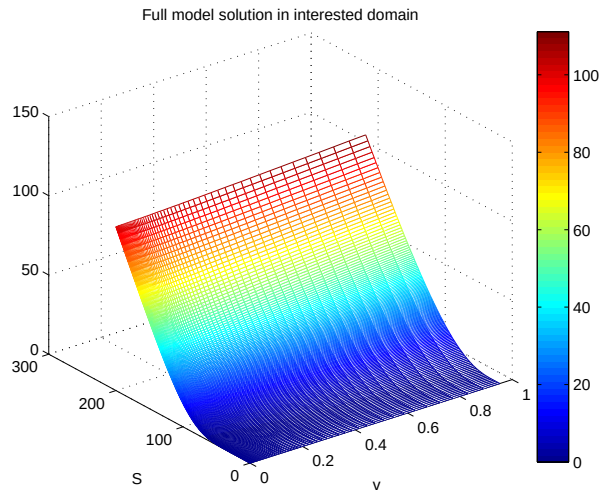

Figure 5.8: Full model solution of $\{u(s,t) \mid 0 \leq s \leq 200,\ 0 \leq v \leq 1\}$ for the parameter set in Case III.

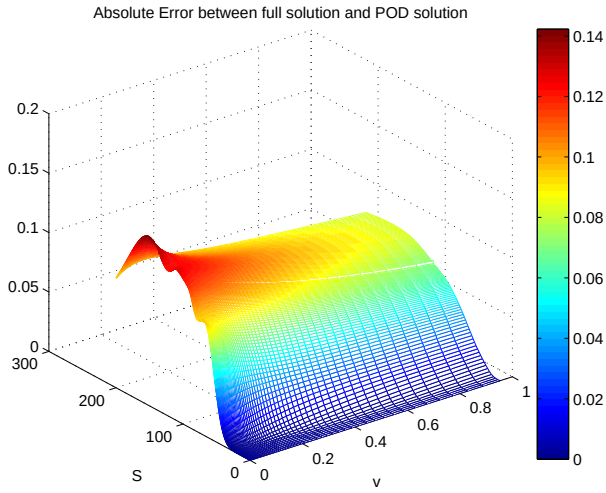

Figure 5.9: Absolute errors of the option prices $\{u(s,t) \mid 0 \leq s \leq 200, 0 \leq v \leq 1.0\}$ between the full model and POD reduced model for the parameter set in Case III.

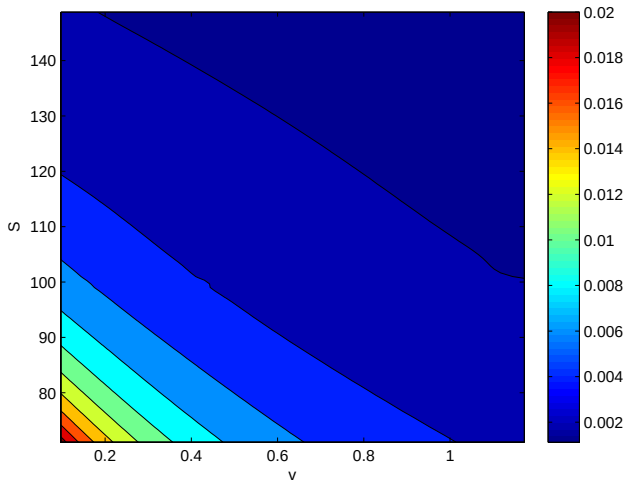

Figure 5.10: Relative errors of the option prices $\{u(s,t) \mid 70 \leq s \leq 150, 0.1 \leq v \leq 1.1\}$ between the full model and POD reduced model for the parameter set in Case III.

Figures 5.11-5.13 displays the basis of the subspace of POD reduced order modeling for the parameter set in case I.

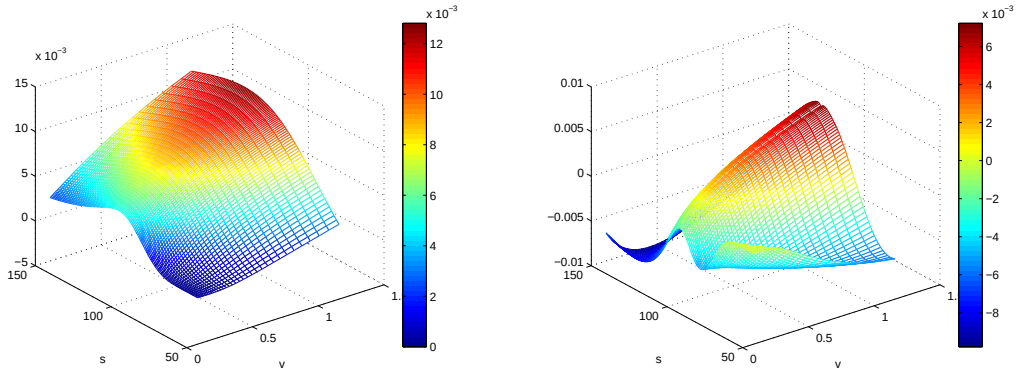

Figure 5.11: The first two modes of the reduced model space. Left figure: the first mode; Right figure: the second mode

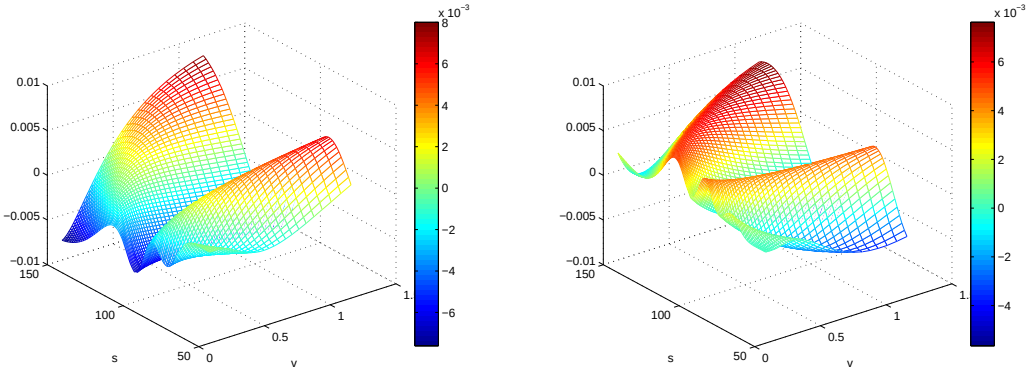

Figure 5.12: The third and fourth modes of the reduced model space. Left figure: the third mode; Right figure: the fourth mode

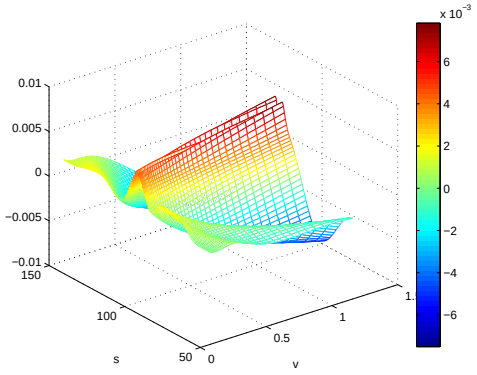

Figure 5.13: The fifth mode of the reduced model space

CHAPTER 6

SUMMARY AND FUTURE WORK

In chapters II and III, we addressed the issue of solving the calibration of the local volatility surface for European options in a non-parametric approach by using a second order Tikhonov regularization. We selected one of the singular values of the Jacobian matrix of the Dupire model as the regularization parameter. For the theoretical volatility models with known analytical solution for European option prices, the proposed method recovers almost exactly the true volatility surface. This method was also tested and proves to be stable for a small amount of noises in the option prices. We also showed the significance of the weighting of option prices when the option prices contain a significant amount of noises.

This method also performs reasonably well for real market data. The observed option prices can be matched very well. The obtained volatility surface lies in a reasonable range with nice general pattern, for example, the skew structure in the equity market and the smile structure in the foreign exchange market. Some instability may still persist in the volatility surface recovered. We attribute this partially to the noises in market data and our assumption that every option is equally important in the market data. A proper weighting scheme may prove to be necessary to reflect the relative importance of different options when they exhibit different amount of noises. When using a constant regularization parameter, the total CPU time is as small as 3-4 seconds for market data.

Although the techniques introduced above focus on calibration for local volatility model for European options, the calibration technique proposed here is developed in a very general framework so that it can be generalized to explore the calibration of other models such as the hybrid local-stochastic volatility models or calibration with respect to other options, such as American options.

In chapters IV, we presented an ADI approach to solve the Heston PDE for European options using a non-uniform mesh as a cheap method of computing the European option prices. In Chapter V, POD reduced order modeling with application in the Heston stochastic volatility model was introduced as a method to reduce the CPU time required to compute the European option prices even further. The original two dimensional convection-diffusion PDE used to pricing European options is approximated by a small dimension ODE. The relative errors are well within the bid-ask spreads. One future research area would consist in extending POD reduced order modeling to the pricing of other exotic derivatives. The other interesting area would be to use POD reduced order modeling to speed up the calibration of stochastic volatility models.

BIBLIOGRAPHY

[1] Y. Achdou and O. Pironneau. *Computational Methods for Option Pricing*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 2005.

[2] H. Albrecher, P. Mayer, W. Schoutens, and J. Tistaert. The little Heston trap. *Wilmott Magazine*, January 2007.

[3] A. K. Alekseev and I. M. Navon. The analysis of an ill-posed problem using multi-scale resolution and second-order adjoint techniques. *Computer Methods in Applied Mechanics and Engineering*, 190:1937–1953, 2001.

[4] L. Andersen and R. Brotherton-Ratcliffe. The equity option volatility smile: an implicit finite difference approach. *The Journal of Computational Finance*, 1:37–64, 1998.

[5] L. Andersen and V. V. Piterbarg. *Interest Rate Modeling*, volume I. Atlantic Financial Press, London, New York, 2010.

[6] L. Anderson. Option pricing with quadratic volatility: a revisit. *Finance and Stochastics*, 15:191–219, 2011.

[7] J. Andreasen and B. Huge. Volatility interpolation. *Risk*, page 7679, Mar 2011.

[8] A. C. Antoulas. *Approximation of Large-Scale Dynamical Systems*. Society for Industry and Applied Mathematics (SIAM), Philadelphia, 2005.

[9] R. C. Aster, B. Borchers, and C. Thurber. *Parameter Estimation and Inverse Problems*. Elsevier Academic Press, Burlington, 2005.

[10] M. Avellaneda, C. Fridman, R. Holems, and D. Samperi. Calibrating volatility surface via relative entropy minimization. *Applied Mathematical Finance*, 4:667–686, 1997.

[11] F. Black and M. Scholes. The pricing of options and corporate liabilities. *Journal of Political Economy*, 81:637–659, 1973.

[12] J. N. Bodurtha,Jr. and M. Jermakyan. Nonparametric estimation of an implied volatility surface. *The Journal of Computational Finance*, 2:29–61, 1999.

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

[14] 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.

[15] L. Capriotti and M. Giles. Fast correlation greeks by adjoint algorithmic differentiation. *Risk*, 23:79–83, 2010.

[16] P Carr and R Lee. Volatility derivatives. *Annual Review of Financial Economics*, 1:319–339, October 2009.

[17] W. Castaings, D. Dartus, F. X. Le Dimet, and G.-M. Saulnier. Sensitivity analysis and parameter estimation for the distributed modeling of infiltration excess overland flow. *Hydrology and Earth System Sciences Discussions*, 4:363–405, 2007.

[18] T. F. Coleman, Y. Li, and A. Verma. Reconstructing the unknown local volatility function. *The Journal of Computational Finance*, 2:77–100, 1999.

[19] C. S. Constable, L. R. Parker, and G. C. Constable. Occam's inversion: A practical algorithm for generating smooth models from electromagnetic sounding data. *Geophysics*, 52(3):289–300, 1987.

[20] R. Cont, N. Lantos, and O. Pironneau. A reduced basis for option pricing. *SIAM Journal on Financial Mathematics*, 2:287–316, 2011.

[21] R. Cont and P. Tankov. Non-parametric calibration of jumpdiffusion option pricing models. *Journal of Computational Finance*, 7(3):1–49, 2004.

[22] L. Cordier, B. Abou El Majd, and J. Favier. Calibration of POD reduced-order models using Tikhonov regularization. *International Journal for Numerical Methods in Fluids*, 63(2):269–296, 2010.

[23] J. C. Cox and S. A. Ross. The valuation of options for alternative stochastic processes. *Journal of Financial Economics*, 3:145–166, 1976.

[24] I. J. D. Craig and A. D. Sneyd. An alternating-direction implicit scheme for parabolic equations with mixed derivatives. *Comp. Math. Appl.*, 16:341–350, 1988.

[25] S. Crepey. Calibration of the local volatility in a generalized Black-Scholes model using Tikhonov regularization. *Journal of Mathematical Analysis on SIAM*, 34(5):1183–1206, 2003.

[26] J. Douglas and H. H. Rachford. On the numerical solution of heat conduction problems in two and three space variables. *Trans. Amer. Math. Soc.*, 82:421–439, 1956.

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

[28] H. Egger and H. W. Engl. Tikhonov regularization applied to the inverse problem of option pricing: convergence analysis and rates. *Inverse Problems*, 21:1027–1045, 2005.

[29] G. Fairweather and I. M. Navon. A linear ADI method for the shallow-water equations. *Journal of Computational Physics*, 37:1–18, 1980.

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

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

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

[33] C. S. L. Graaf de. Finite difference methods in derivatives pricing under stochastic volatility models. Master's thesis, Universiteit Leiden, 2012.

[34] 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.

[35] A. Griewank and A. Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation,second edition*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 2008.

[36] M. Gunzburger. Adjoint equation-based methods for control problems in incompressible, viscous flows. *Flow, Turbulence and Combustion*, 65:249–272, 2000.

[37] P. S. Hagan, D. Kumar, Lesniewski A. S., and Woodward D. E. Managing smile risk. *Willmot Magazine*, pages 84–108, 2002.

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

[39] T. Hein. Some analysis of Tikhonov regularization for the inverse problem of option pricing in the price dependent case. *Journal for Analysis and its Applications*, 24(3):593–609, 2005.

[40] 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.

[41] P. Holmes, J. L. Lumley, G. Berkooz, and C. W. Rowley. *Turbulence, Coherent Structures, Dynamical Systems and Symmetry*. Cambridge University Press, Cambridge, UK, second edition, 2012.

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

[43] W. Hundsdorfer and J. G. Verwer. *Numerical solution of time-dependent advection-diffusion-reaction equations*. Springer, Berlin, 2003.

[44] K. Itô. Stochastic integral. *Proc. Imp. Acad. Jap.*, 20(8):519–524, 1944.

[45] IN'T Hout K. J. and S. Foulon. ADI finite difference schemes for option pricing in the Heston model with correlation. *International Journal of numerical analysis and modeling*, 7(2):303–320, 2010.

[46] 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.

[47] L. Jiang and Y. Tao. Identifying the volatility of underlying assets from option prices. *Inverse Problems*, 17:137–155, 2001.

[48] R. Lagnado and S. Osher. Reconciling difference. *Risk*, 10:79–83, 1997.

[49] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK Users'Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 1998.

[50] J. Lions. *Optimal control of systems governed by partial differential equations*. Springer-Verlag, 1968.

[51] R. Merton. Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3:125–44, 1976.

[52] I. M. Navon. Pent: a periodic pentadiagonal systems solver. *Communications in Applied Numerical Methods*, 3:63–69, 1987.

[53] I. M. Navon. Practical and theoretical aspects of adjoint parameter estimation and identifiability in meteorology and oceanography. *Dynamics of Atmospheres and Oceans*, 27:55–59, 1998.

[54] 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.

[55] K. Pearson. On lines and planes of closest to points in space. *Philosophical Magazine*, 1901.

[56] O. Pironneau. Proper orthogonal decomposition for pricing options. *The Journal of Computational Finance*, 16(1):33–46, Fall 2012.

[57] Pinnau R. Model reduction via proper orthogonal decomposition. In *Model Order Reduction: Theory, Research Aspects and Applications*, pages 95–109. Springer, 2008.

[58] E. W. Sachs and M. Schu. Reduced order models (POD) for calibration problems in finance. In *Numerical Mathematics and Advanced Applications,ENUMATH 2007*, pages 735–742, Heidelberg, 2008. Springer-Verlag.

[59] E. W. Sachs and M. Schu. A priori error estimates for reduced order models in finance. *ESAIM: Mathematical Modelling and Numerical Analysis*, 47:449–469, March 2013.

[60] W. Schoutens, E. Simons, and J. Tistaert. A perfect calibration ! now what ? *Wilmott Magazine*, March 2004.

[61] M. Schroder. Computing the constant elasticity of variance option pricing formula. *The Journal of Finance*, XLIV(1):211–219, March 1989.

[62] L. Sirovich. Turbulence and the dynamics of coherent structures. i-iii. *Quarterly of Applied Mathematics*, 45(3):561–590, 1987.

[63] A. N. Tikhonov and V. Y. Arsenin. *Solution of Ill-posed Problems*. Winston and Sons, Washington, 1977.

[64] D. Travella and C. Randall. *Pricing Financial Instruments*. Wiley, 2000.

[65] G. Turinici. Calibration of local volatility using the local and implied instantaneous variance. *The Journal of Computational Finance*, 13(2):1–18, 2009.

[66] C. Zhu, R. H. Byrd, and J. Nocedal. L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.

BIOGRAPHICAL SKETCH

Jian Geng

EDUCATION

Florida State University
Department of Mathematics
M.S. in Financial Mathematics
2010

Ocean University of China
Department of Marine Science
B.S. in Physical Oceanography
2005

PUBLICATION

Non parametric calibration of the local volatility surface for European options using a second order Tikhonov regularization by J. Geng, I.M. Navon and X.Chen. Accepted by Quantitative Finance in June 2013.

PROFESSIONAL EXPERIENCE

Southern Company, Atlanta, May 2012-Aug 2012
Intern, Risk Analysis Service (RAS)
Research Institute of HuaChuang Securities, Beijing, May 2010-Jul 2010
Intern, Financial Engineering Group

TEACHING EXPERIENCE

Calculus III solo Instructor (Spring 2013, Fall 2012)
Calculus II solo Instructor (Summer 2011)
Pre Calculus solo Instructor (Spring 2010, Fall 2010)

RESEARCH INTERESTS

Local and stochastic volatility modeling and calibration, Interest rates modeling and calibration, POD Reduced Order Modeling

AWARDS

The second prize in the poster competition in the 14th Annual Financial Mathematics Festival, FSU, 2012;
Distinguished Teaching Assistant, Mathematics Department, FSU, 2013;