Pricing model calibration through Stochastic Optimization

Ivan Perez, Kei Nemoto, Tian Cai, Motahare Mounesan

Advisor: Professor Felisa Vázquez-Abad

January 6, 2020

0.1 Introduction

Market makers are the arbiters of risk neutral pricing in derivatives markets. While their models are proprietary, they must work in balance with market participants (e.g., market-takers and boutique traders) to accurately price financial products. Model calibration is an initial step practitioners in algorithmic trading must undertake to implement trading strategies. The Black-Scholes-Merton model is the most well known, and is built into as a functionality of many financial information databases, and data analysis tools. The Heston Model, also well known, sparked an interest in understanding stochastic volatility, and still has an active community of researchers and practioners studying its methods of facile parameterization and implementation.

For robust and widespread parameterization of financial derivative instruments we attempt to apply stochastic optimization to estimate parameters of the Heston Model. Herein, we use the Black-Scholes-Merton and Heston Stochastic differential equations (SDE) as an assumed observed differential equation whose parameters we aim to estimate from market price dynamics of observed derivative products.

We attempt implement infinitesimal perturbation analysis (IPA) to estimate the gradient of the expectation of the cost function $\nabla_\theta J(\theta)$. and minimize $J(\theta)$ through the iterative methods presented in this class and commonly found in optimization and numerical analysis literature.

The structure of the paper is as follows. Section 2 explains the financial framework of Black-Scholes-Merton and Heston models. Section 3 outlines the methodology, and the Stochastic Optimization framework. Section 4 demonstrates the application of theorems to the parameter estimation problem. Section 5 illustrates the implementation scheme following with section 6 describing the data. Section 7 discusses the results of experiments on synthetic and real options data. The paper concludes with sections 7, and 8, titled Conclusion and Future Work respectively.

0.2 Financial Aspect Background

0.2.1 Black-Scholes-Merton Formula

Black-Scholes-Merton (BSM) model was proposed by Fischer Black, Myron Scholes and Robert Merton in 1970 as a theoretical estimation model for European-style call options. To simplify the task, we take the Black-Scholes-Merton closed form solution. It describes asset prices moving through time with drift component μ and volatility component σ. The stochastic differential equation for the rate of return is given below:

$$S_t = S_t \mu dt + \sigma S_t dW$$

Applying the **Independence lemma (2.3.4), Shreve**, the stock price movement under the risk neutral measure is described below.

$$S(t) = S(0) \exp\left(\sigma \tilde{W}(t) + \left(r - \frac{1}{2}\sigma^2\right) t\right)$$

Where $\tilde{W}$ is an a Brownian motion in a probability space $(\Omega, \mathfrak{F}, \mathbb{P})$.

The final result of the BSM Equation describes the price process of a call option under the risk neutral measure $\tilde{\mathbb{P}}$:

$$C_{t,K_N} = S_t N(d_1) - e^{-r\tau} K N(d_2)$$

$$d_1 = \frac{1}{\sigma\sqrt{\tau}} \left[\log \frac{S_t}{K} + \left(r + \frac{\sigma^2}{2}\right)\tau\right] \text{ and } d_2 = d_1 - \sigma\sqrt{\tau}$$

Where $N(\cdot)$ is the cumulative distribution function (CDF) of the Gaussian Kernel, T is time to maturity, for $0 \leq t \leq T$, and $\tau = T - t$.

0.2.2 Heston model

The Heston model relaxes the constant σ parameter of BSM. The model introduces stochastic volatility v_t as a Cox-Ingersol-Ross (CIR) process. The differential equations are:

$$dS_t = \mu S_t dt + \sqrt{v_t} S_t dW_t^S$$
$$dv_t = a(b - v_t)dt + \xi\sqrt{v_t} dW_t^v$$

and the price of a Call Option is described as the expectation of being above its strike, K.

$$C_{t,K_N} = \mathbb{E}[(S_t(\mu, v_t(a, b, \xi)) - K_N)_+], N = 1, 2, ..., 25$$

While the Heston model better captures the idea of a dynamically changing volatility, it is not as straight forward to solve as there is no readily accessible closed form solution available. **GOT TO HERE**

0.3 Methodology

Under Stochastic Optimization framework, we first assume the target company is using the BSM model.

0.3.1 Stochastic Optimization Framework

To align with notations used in the course and to formally put the problem into stochastic optimization framework, we have the following setting:

Non-Random Observations: Call prices set by Market Makers (e.g., Jane Street, State Street, Citadel) are observed over a 30 Day period at 25 unique strike prices enumerated $K_1, ..., K_25$. They are assumed to follow either the BSM or Heston model dynamics outlined previously. We describe the vector of these observed prices as X_t .

$$X_t = \begin{bmatrix} C_{t,K_1} \\ \vdots \\ C_{t,K_{25}} \end{bmatrix}$$

We denote our call option estimation as an expectation of the Future stock price, $\mathbb{E}[(S_t(\theta;\omega) - K_N)_+] = X_t(\theta;\omega)$ is defined as a vector of the Call Option price estimated using parameters guessed by our algorithm for some time point t before maturity T and a constant strike price K_N where N enumerates a unique strike price and ranges from 1 to 25. C_{t,n,K_N}. To align with the notation in Part II Gradient Estimation, we further clarify $\{X_t\}$ to denote observations:

$$\hat{X}_t(\theta;\omega) = \begin{bmatrix} \mathbb{E}[(S_t(\theta;\omega) - K_1)_+] \\ \vdots \\ \mathbb{E}[(S_t(\theta;\omega) - K_25)_+] \end{bmatrix}$$

We define our cost function, $J(\theta)$, to be the estimation of the Mean Squared Error of our estimated Call option prices and published Call option prices.

$$J(\theta) = J(\mu, a, b, \xi) = \frac{1}{2T}\sum_{t=0}^{T-1} \mathbb{E}\left[((S_t(\mu, v_i(a, b, \xi)) - K_N)_+ - C_{t,K_N,obs})^2\right]$$

After IPA Gradient Estimation, the recursion will be:

$$\theta_{n+1} = \theta_n + \epsilon_n Y_n$$

$$Y_n = \nabla_\theta^{IPA} J(\theta_n)$$

The Optimization objective is to minimize the expected error of estimated prices, denoted as $\hat{X}_t$, and their non-random published observations X_t.

$$\min_{\theta \in (R^n)_+} J(\theta) = E[(X_n - \hat{X_n})^2]$$

Min-batch Heston

Since the initial and optimal values of each parameter are usually quite small in the Heston model, the process is vulnerable to extreme values. To address this issue, we decided to use the batch method proposed in Chapter 5 and 6, where we generate a certain number of estimations as a batch, then take the average of them. This method is justified by the Central Limit Theorem and useful to reduce the influence of unexpected estimations. In our model, we applied this method to the generation of a sequence of price estimations in our Heston calibration model.

$$\frac{1}{N}\sum_{i=1}^{N} S_{t+\Delta t,i} \tag{1}$$

, where N is batch size. The detail of the algorithm step is described in section 7.

0.4 Applied Theorems and Proofs

0.4.1 Black-Scholes-Merton Deterministic Optimization

Objective function

BSM

$$\arg\min_{\theta \in \mathbb{R}^d} J(\theta)$$

we define our object function $J(\theta)$ as the difference between our expectation of the call payoff $E[]$ and the pay off expected by market maker $C_{K_N,obs}$

$$J(r,\sigma) = \frac{1}{2T}\sum_{t=0}^{T}(C_{K_N}(t,S_t) - C_{K_N,obs})^2$$

$$C_{K_N}(t,S_t,r,\sigma) = S_t N(d_1) - Ke^{-r\tau}N(d_2)$$

$$d_1 = \frac{1}{\sigma\sqrt{\tau}}\left[\log\left(\frac{S_t}{K}\right) + \left(r + \frac{1}{2}\sigma^2\right)\tau\right]; d_2 = d_1 - \sigma\sqrt{\tau}$$

Demonstration of Convexity through Gradient and Hessian

First denote the n times continuously differential mapping from $\mathbb{R}^d_+ \to \mathbb{R}^d_+$ by $\mathcal{C}^n$. For $J \in \mathcal{C}^1$, we denote the $\nabla J(\cdot)$. and the hessian of $J(\cdot)$ by $HJ(\cdot) = \nabla^2 J(\cdot)$.

Definition 1.2 can be applied to show convexity of $J(r,\sigma)$

The Gradient is:

$$\nabla_\theta J(\theta) = \nabla_{r,\sigma} J(r,\sigma) = \begin{bmatrix} \frac{\partial}{\partial r}J(r,\sigma) \\ \frac{\partial}{\partial \sigma}J(r,\sigma) \end{bmatrix}$$

The Hessian is:

$$HJ(r,\sigma) = \begin{bmatrix} \frac{\partial^2}{\partial r^2}J(r,\sigma) & \frac{\partial^2}{\partial r \partial \sigma}J(r,\sigma) \\ \frac{\partial^2}{\partial \sigma \partial r}J(r,\sigma) & \frac{\partial^2}{\partial \sigma^2}J(r,\sigma) \end{bmatrix}$$

Individual partial derivatives with respect to r of the BSM Call formula:

$$\frac{\partial}{\partial r}C(r,\sigma) = S_t\frac{1}{\sqrt{2\pi}}e^{-\frac{d_1^2}{2}}\frac{\sqrt{\tau}}{\sigma} + K\pi e^{-r\tau}N(d_2) - Ke^{-r\tau}\frac{1}{\sqrt{2\pi}}e^{-\frac{d_2^2}{2}}\frac{\sqrt{\tau}}{\sigma}$$

$$\frac{\partial^2}{\partial r^2}C(r,\sigma) = [K\pi N(d_2) - K\frac{1}{\sqrt{2\pi}}e^{-\frac{d_2^2}{2}}\frac{\sqrt{\tau}}{\sigma}](-\tau)e^{-r\tau}$$

$$\frac{\partial^2}{\partial r\partial\sigma}C(r,\sigma) = (S_t e^{-\frac{d_1^2}{2}} - Ke^{-r\tau-\frac{d_2^2}{2}})\frac{\sqrt{\tau}}{\sqrt{2\pi}}(-1)\sigma^{-2}$$

Combine to yield elements of the gradient and hessian matrices:

$$\frac{\partial}{\partial r}J(r,\sigma) = \frac{1}{2T}\sum_{t=0}^{T}[2(C_{K_N}(t,S_t) - C_{K_{N,obs}})\frac{\partial C}{\partial r}]$$

$$\frac{\partial^2}{\partial r^2}J(r,\sigma) = \frac{1}{2T}\sum_{t=0}^{T}[2(\frac{\partial C}{\partial r})^2 + 2(C_{K_N}(t,S_t) - C_{K_{N,obs}})\frac{\partial^2 C}{\partial r^2}]$$

$$\frac{\partial^2}{\partial r\partial\sigma}J(r,\sigma) = \frac{1}{2T}\sum_{t=0}^{T}[2\frac{\partial C}{\partial r}\frac{\partial C}{\partial \sigma} + 2(C_{K_N}(t,S_t) - C_{K_{N,obs}})\frac{\partial^2 C}{\partial r^2}]$$

Similarly individual partial derivatives with respect to σ of the BSM Call formula:

$$\frac{dd_1}{d\sigma} = \frac{1}{\sqrt{\tau}}\log\left(\frac{S_t}{K}\right)\frac{-1}{\sigma^2} + \frac{\sqrt{\tau}}{2}$$

$$\frac{\partial}{\partial\sigma}C(r,\sigma) = S_t\frac{1}{\sqrt{2\pi}}e^{-\frac{d_1^2}{2}}\frac{dd_1}{d\sigma} - Ke^{-r\tau}\frac{1}{\sqrt{2\pi}}e^{-\frac{d_2^2}{2}}(\frac{dd_1}{d\sigma} - \sqrt{\tau})$$

$$\frac{\partial^2}{\partial\sigma^2}C(r,\sigma) = [S_t\frac{1}{\sqrt{2\pi}}e^{-\frac{d_1^2}{2}} - Ke^{-r\tau}\frac{1}{\sqrt{2\pi}}e^{-\frac{d_2^2}{2}}]\frac{1}{\sqrt{\tau}}\log\left(\frac{S_t}{K}\right)2\sigma^{-3}$$

Combine to yield elements of the gradient and hessian matrices:

$$\frac{\partial}{\partial r}J(r,\sigma) = \frac{1}{2T}\sum_{t=0}^{T}[2(C_{K_N}(t,S_t) - C_{K_{N,obs}})\frac{\partial C}{\partial\sigma}]$$

$$\frac{\partial^2}{\partial\sigma^2}J(r,\sigma) = \frac{1}{2T}\sum_{t=0}^{T}[2(\frac{\partial C}{\partial\sigma})^2 + 2(C_{K_N}(t,S_t) - C_{K_{N,obs}})\frac{\partial^2 C}{\partial\sigma^2}]$$

fig 1 the map of r vs sigma and J for some fixed S, K ? fig 2 the map of dr, and ds surface
proof/experiments in mathematica
With almost sufficient evidence we apply **Theorem 1.1**. *LetJ* $\in \mathcal{C}$:

• *A local minimum (local maximum)* θ^* *of J is a stationary point, that is, it satisfies the first order optimality condition*:

$$\nabla J(\theta^*) = 0$$

i.e.

$$\frac{\partial}{\partial r}C(r,\sigma) = \frac{\partial}{\partial\sigma}C(r,\sigma) = 0$$

• *A decision value* θ^* *is a local miniumum of J if in addition to (1.4), the following is also satisfied*

$$\nabla^2 J(\theta^*) > 0$$

Equation (1.5) is called the second order optimality condition.

• *If J is a convex(concave) function, then (1.4) is necessary and sufficient for* θ^* *being a global minimum (maximum).*

• see appendix for surface of of $J(\theta)$. While their exists a set of 0's for θ^*, a concise form using BSM Call formula is impractical.

Theorem 1.4 can be applied to find a local minimum of θ with constant step size, as used in our experiments.

• while the Lipschitz constant L exists for most *rands* except for cases where it crosses the Strike price. See Appendix for Lipschitz L surface for $\delta \in [0.0.0001]$.

While demonstrated in the paper, the appendix does also hold some preliminary experiments on the convergence of θ to some local minimum θ with sufficiently small ϵ.

0.4.2 Heston Stochastic Volatility Asset Pricing Model

The Objective Function

The Objective function $J(\theta)$ as the mean squared error of our expectation of the call payoff $\mathbb{E}[(S-K)_+]$ and the expected payoff of market maker $C_{K_N,obs}$. Herein, the parameters of θ as μ, a, b, and ξ. The describe the volatility process

$$J(\theta) = J(\mu, a, b, \xi) = \frac{1}{2T}\sum_{t=0}^{T-1} \mathbb{E}\left[((S_t(\mu, v_i(a, b, \xi)) - K_N)_+ - C_{t,K_N,obs})^2\right]$$

For now, one can assume convexity and a coercive vector field for this $J(\theta)$ as we attempt to build the IPA gradient estimator. The nature of the assumed vector field will reveal itself as we further explore the underlying processes S_t and v_t, and take their partial derivatives. We begin by taking initializing the IPA framework with Definition 8.1, and Theorem 8.1 and their post-ambles, restated below:

Definition 8.1 *Let $\Theta \subset \mathbb{R}$ be an open connected set, such that $X(\theta)$, for $\theta \in \Theta$, is a real-valued random variable defined on a common underlying probability space $(\Omega, \mathfrak{F}, \mathbb{P})$. Let θ be an interior point of Θ. We say that $h(X(\theta))$ is almost sureflly (locally) Lipschitz continuous on Θ if the set $\mathcal{N} \subset \Omega$ of realization such that*

$$|h(X(\theta_1;\omega)) - h(X(\theta_0;\omega))| \leq |\theta_1 - \theta_0|K(\omega)$$

has probability one, i.e., $\mathbb{P}(\mathcal{N} = 1)$ and $\mathbb{E} < \infty$, where measurability of $\mathcal{N}$ is assumed.

In the following we often write $\frac{d}{d\theta}f(\theta_0)$ for the derivative of a differentiable mapping f at a point θ, i.e., we let

$$\frac{d}{d\theta}\bigg|_{\theta=\theta_0} f(\theta) = \frac{d}{d\theta}f(\theta_0)$$

Theorem 8.1. Let $\Theta \subset \mathbb{R}$ *be and open connected set, such that $X(\theta)$ is a measurable mapping on a common underlying probability space $(\Omega, \mathfrak{F}, \mathbb{P})$ Let θ_0 inΘ If,*

(i) *the sample path(or stochastic) derivative of $dX(\theta)/d\theta$ exists with probability one at θ*

(ii) *the mapping $h : S \to \mathbb{R}$ is differentiable,*

(iii) *the mapping $h(X(\theta))$ is Lipschitz continuous on Θ with probability one, then*

$$\frac{d}{d\theta}\mathbb{E}\left[h(X(\theta))\right]\bigg|_{\theta=\theta_0} = \mathbb{E}\left[\frac{d}{d\theta}X(\theta)h'(X(\theta))\bigg|_{\theta=\theta_0}\right]$$

where $h'(x)$ denotes the derivative of $h(x)$ with respect to x.

Beginning with assumption (i) we must show that for some CDF of $S(\theta)$ denoted as $F_\theta(S(\theta))$ we can apply **Lemma 8.1** to $dS(\theta)/d\theta$. from the given pricing process $S(\theta)$ has an iterative form, which will be discussed in the next subsection. However, for now we know it is exponentially distributed with parameter $\lambda(\theta, t)$. While the exponential CDF exists, a closed form is unknown to us. However, the result that can be taken away is that the stochastic derivative exists with probability 1. We proceed with this knowledge to further describe this stochastic derivative.

Summation forms of S_t and v_t and their derivative processes $S'_T(\theta)$ and $Z_t(\theta)$.

Beginning with that assumption (i) may hold, we use the iterative forms of S_t and v_t to develop derivative processes for v_t as $Z_t(\theta)$ and $S'_t(\theta)$.

We take well known recursive forms of the price process S_t and v_t assuming a $\Delta t = 1$:

$$S_t = S_{t-1} \exp\left[\sqrt{v_{t-1}} W_2 + \mu - \frac{1}{2} v_{t-1}\right]$$

$$v_t = \left(\sqrt{v_{t-1}} + \frac{\xi}{2} W_1\right)^2 - a(v_{t-1} - b) - \frac{\xi^2}{4}$$

Finding the Derivative process of v_T; $Z_T(\theta)$.

Begin by expanding brackets of v_t and rearranging to afford a recursive form with 3 components:

$$v_t = v_{t-1}(1-a) + \xi W_1 \sqrt{v_{t-1}} + \frac{\xi^2}{4}(W_1^2 - 1) + ab$$

The summation form of for some terminal T is shown below:

$$v_T(\theta) = \sum_{i=0}^{T-1} v_i(\theta)(1-a) + \sum_{i=0}^{T-1} \xi W_1 \sqrt{v_i} + T\left(\frac{\xi^2}{4}(W_1^2 - 1) + ab\right)$$

One can similarly take the derivative process with respect to a:

$$\frac{\partial}{\partial a} v_t = \frac{\partial}{\partial a} v_{t-1} \left(1 - a + \frac{\xi W_1}{2\sqrt{v_{t-1}}}\right) - v_{t-1} + b$$

Similarly for b one observes:

$$\frac{\partial}{\partial b} v_t = \frac{\partial}{\partial b} v_{t-1} \left(1 - a + \frac{\xi W 1}{2\sqrt{v_t - 1}}\right) + a$$

And finally for ξ:

$$\frac{\partial}{\partial \xi} v_t = \frac{\partial}{\partial \xi} v_{t-1} \left(1 - a + \frac{\xi W 1}{2\sqrt{v_t - 1}}\right) + W_1 \sqrt{v_{t-1}} + \frac{\xi}{2}(W_1^2 - 1)$$

The generalized derivative iterative process $Z_t(\theta)$, for some θ, is:

$$Z_t(\theta) = \frac{dv_t(\theta)}{d\theta} = \frac{dv_{t-1}(\theta)}{d\theta} B(v_{t-1}(\theta)) + C(v_{t-1}(\theta))$$

Therefore we can write the derivative process $Z_t(\theta)$ in summation form, with special attention to each partial derivative being taken:

$$Z_t(\theta) = \frac{dv_t(\theta)}{d\theta} = \sum_{i=0}^{t-1} \frac{dv_i(\theta)}{d\theta} B(v_i(\theta)) + \sum_{i=0}^{t-1} C(v_i(\theta))$$

$$B(x) = 1 - a + \frac{\xi W_1}{2\sqrt{x}}$$

https://www.overleaf.com/project/5dcc6e56308c4700019e9a28

$$C(x) = \begin{cases} b - x & \frac{\partial}{\partial a} \\ a & \frac{\partial}{\partial b} \\ W_1\sqrt{x} + \frac{\xi}{2}(W_1^2 - 1) & \frac{\partial}{\partial \xi} \end{cases}$$

Finding the derivative process of $S_t(\theta)$; $S'_t(\theta)$.

Begin by finding a product representation of $S_t(\theta)$, and then taking the log to get to a summation form.

$$S_t(\theta) = S_0 \prod_{i=0}^{t-1} \exp\left[\sqrt{v_i}W_2 + \mu + \frac{1}{2}v_i\right]$$

$$\log S_t(\theta) = S_0 + \sum_{i=0}^{t-1}\left[\sqrt{v_i}W_2 + \mu + \frac{1}{2}v_i\right]$$

Taking the derivative with respect to θ:

$$\frac{d}{d\theta}\log S_t(\theta) = \sum_{i=0}^{t-1}\frac{d}{d\theta}\left[\sqrt{v_i}W_2 + \mu + \frac{1}{2}v_i\right]$$

Special consideration for the cases of which partial derivative in θ is being taken:

$$\frac{d}{d\theta}\left[\sqrt{v_i}W_2 + \mu + \frac{1}{2}v_i\right] = \begin{cases} \frac{1}{2}\frac{\partial v_i}{\partial \theta}\left(\frac{W_2}{\sqrt{v_i}} - 1\right) & \theta \in \{a, b, \xi\} \\ 1 & \theta = \mu \end{cases}$$

Which can be recast using the derivative process $Z_t(\theta)$:

$$\frac{d}{d\theta}\log S_t(\theta) = \frac{1}{S_t(\theta)}S_t'(\theta) = \sum_{i=0}^{t-1} Z_i(\theta)\frac{1}{2}\left(\frac{W_2}{\sqrt{v_i}} - 1\right)$$

$$S_t'(\theta) = S_t(\theta)\sum_{i=0}^{t-1} Z_i(\theta)\frac{1}{2}\left(\frac{W_2}{\sqrt{v_i}} - 1\right)$$

Following this result condition (i) is satisfied providing the stochastic derivatives $Z_t(\theta)$ and $S_t'(\theta)$.

Examining the unbiasedness of the Stochastic derivatives $Z_t(\theta)$ and $S_t'(\theta)$:

While a formal proof is not provided, would have been easy to assume the unbiasedness of an exponentially distributed random variable, however, given it's explicit dependenence on past iterations, the derivative processes, and as a result the IPA estimator are likely biased estimators.

Examining Continuity of $h(x)$ justifying assumptions (ii) and (iii) of Theorem 8.1.

Restating **Definition 8.1 +** *Let $\Theta \subset \mathbb{R}$ be an open connected set, such that $X(\theta)$, for $\theta \in \Theta$, is a real-valued random variable defined on a common underlying probability space $(\Omega, \mathfrak{F}, \mathbb{P})$. Let θ be an interior point of Θ. We say that $h(X(\theta))$ is almost surefully (locally) Lipschitz continuous on Θ if the set $\mathcal{N} \subset \Omega$ of realization such that*

$$|h(X(\theta_1;\omega)) - h(X(\theta_0;\omega))| \leq |\theta_1 - \theta_0|K(\omega)$$

has probability one, i.e., $\mathbb{P}(\mathcal{N} = 1)$ and $\mathbb{E} < \infty$, where measurability of $\mathcal{N}$ is assumed.

$h(S_t(\theta;\omega))$ and use dummy variable $x = S_t(\theta_0;\omega)$ and $x + \delta = S_t(\theta_1;\omega)$for ease of calculation for the **locally** Lipschitz random variable $L =: K(\omega)$.

$$h(x) = ([x - K]_+)^2$$

while the function is continuous and twice differentiable upon inspection we write $dh(x)/dx$ as $h'(x)$:

$$h'(x) = 2((x - K)_+)\mathbf{1}_{x>K}$$

examining Lipschitz continuity:

$$|h(x + \delta) - h(x)| \leq L\delta$$

$$h(x+\delta) = \begin{cases} ((x+\delta) - K)^2 = (x+\delta)^2 - 2(x+\delta)K + K^2 & x > K \\ (\delta^2) & x = K \\ ((x+\delta) - K)^2 = (x+\delta)^2 - 2(x+\delta)K + K^2 & x < K \text{ and } x + \delta > K \\ 0 & x < K \text{ and } x + \delta \leq K \end{cases}$$

Extracting the δ for the four cases:

$$|h(x+\delta)-h(x)| = \begin{cases} \delta(2x+\delta-2K) & x > K \\ \delta^2 & x = K \\ (x+\delta)^2 - 2(x+\delta)K + K^2 & x < K \text{ and } x+\delta > K \\ 0 & x < K \text{ and } x+\delta \leq K \end{cases}$$

from this we all cases except $x < K$ and $x + \delta > K, x \neq K$ is h Lipschitz continuous. One can argue, that the probability of the Stock price finishing at such a price where a small move δ would put it above the Strike price is measure 0. With these conclusions (ii) and (iii) are shown to be satisfied.

IPA derivative estimator

Given that assumptions (i), (ii) and (iii) of **Theorem 8.1** are satisfied, one can reasonably exchange derivative and expectation operators:

$$\frac{d}{d\theta}\mathbb{E}\left[h(X(\theta))\right] = \mathbb{E}\left[\frac{d}{d\theta}h(X(\theta))\right] = \mathbb{E}\left[\frac{\partial}{\partial\theta}X(\theta)h'(X(\theta))\right]$$

Restating our functions:

$$J(\theta) = \frac{1}{2T}\sum_{t=0}^{T-1}\mathbb{E}\left[(S_t(\theta;\omega)-K)_+ - C_{t,K_N,obs})^2\right]$$

$$X_t(\theta) = S_t(\theta)$$

$$h(x) = ((x-K)_+ - C_{K,obs})^2$$

$$h'(X(\theta)) = 2((x-K)_+)\mathbf{1}_{x>K} = 2(x-K)_+$$

$$\frac{\partial}{\partial\theta}X_t(\theta) = S'_t(\theta) = \begin{cases} S_t(\theta)\sum_{i=0}^{t-1} Z_i(\theta)\frac{1}{2}\left(\frac{W_2}{\sqrt{v_i}} - 1\right) & \theta = \{a, b, \xi\} \\ S_t(\theta) =: S_{t-1}\exp\left[\sqrt{v_{t-1}}W_2 + \mu - \frac{1}{2}v_{t-1}\right] & \theta = \mu \end{cases}$$

Therefore our IPA gradient estimator is:

$$\nabla_\theta^{IPA} J(\theta) = \frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[S'_t(\theta)(S_t(\theta)-K_N)_+\right]$$

0.4.3 the coerciveness of the vector field $J(\theta)$

The vector field was found to be coercive in the BSM case. Data presented has shown evidence of a well known volatility smile, increased convexity, for OTM Options. We can expect that the vector field may also be coercive for the Heston model. See Appendix for brief experiments on continuity of $J(\theta)$, and coerciveness of the vector field.

0.5 Implementation

Using the mathematical models explained in the previous sections, we designed 2 algorithms, one for each of the methods. Algorithm 1 present the steps of the Black-Schole-Merton model algorithm in detail. As earlier mentioned, this algorithm searches for a tuple (r, σ) -hyperparameters of the model- which fits best to the observations. Similarly, Algorithm 2 search for the optimal hyperparameters of the Heston model which is a quad-tuple(μ, a, b, ξ). We used python3 for the programming part of this project.

Algorithm 1: Back-Shole model Algorithm

Input: N(number of iterations), ϵ, r, σ, T, O Set of observations
Output: σ,r

1 Initialize the target parameters: r and σ, the number of iterations N, and the maturity period T.
2 Generate estimations over T period with current r and sigma values.
3 Calculate the Gradient of *Cost function* for r and *sigma* using the observations and current estimations.
4 Update the value of r and *sigma* using the Gradient values.
5 Check the non-negative constraint. If any of values get negative value, project it onto 0.0001.
6 Store the new values in lists. Repeat *Steps 2 to 6* N times, until it converges to optimal values.

Algorithm 2: Heston Model Algorithm

Input: N(number of iterations), ϵ, μ, a, b and ζ, T, O Set of observations
Output: μ, a, b and ξ

1 1: Initialize the target parameters: μ, a, b and ξ, the number of iterations N, the maturity period T, and the batch-size.
2 2: Generate estimations with current μ, a, b and ξ values by the number of batch-size as we described in equation (6).
3 3: Take the average of a batch and use it as an estimation with current parameter values.
4 4: Take the average of the IPA Gradients for each parameter at each time point τ up to T as a surrogate gradient (equation (7)).
5 5: Update the value of μ, a, b and ξ using the Gradient values.
6 6: Check the non-negative constraint. If any of values get negative, project it onto 0.0001.
7 7: Store the new values in lists.
8 8: Repeat *Step 2 - 6* N times.

0.6 Data Description

0.6.1 Real Data

The data examined is a set of historical stock price data at the close of the trading day in USD for $AMD, $CHTR. Call option price data observed is at the historical close of the trading day and is provided by Bloomberg LP. Both data sets are recorded for 24 days between Oct 22, 2019 to Nov 22, 2019. The record for each day includes date, strike, bid, ask, IVM and S_{obs}. The random vector $\hat{X}_t(\theta;\omega)$ contains the estimated stock price $S_t(\theta)$, and a the estimation of the set conjugate call prices $C_{t,K_N}(S_t(\theta))$ at different strike prices, K_N.

Under the assumption of constant volatility and interest rates, we take observations during a period where market volatility σ is stable. While it is well known that interest rates do change dynamically, both BSM and Heston models assume a constant interest rate r, μ over the estimation period.

$$X_t = \begin{bmatrix} S_t \\ C_{t,K_1} \\ \vdots \\ C_{t,K_N} \end{bmatrix}$$

However, the Bloomberg price data showed bid and ask, and for consistency of calculation, the best ask was used as the option price. Features such as last trade price, and Black-Scholes implied volatility were ignored due to sparsity, or coherent with out model assumptions.

Synthetic Data

As introduced, to better understand the power and limits of our method using both Black-Scholes and Heston models, before jumping into the real data set, we create synthetic data sets using the configurations followed.

Black-Scholes-Merton Model Synthetic data for the BSM model, was generated using a target pair of interest rate and volatility r^*, σ^*. using the risk neutral pricing formula presented in Section 1 and a Brownian noise generated by $W_t = \mathcal{N}(0,1)$. the successive price is generated using the scheme below

$$S_{t+\Delta t} = S_t \exp\left[(\mu - \frac{1}{2}\sigma^2)\Delta t + \sqrt{\sigma \Delta t} W_t\right]$$

Heston Model

- We initialize a set of ideal parameters $\theta^* = \{\mu^* = 0, a^* = 1, b^* = 0.55, \xi^* 0.02\}$, and seed values for stock price, $S_0 = 35$, and volatility v_0.
- Using the Milstein discretization schemes, with $\Delta t = 1$,

$$v_{t+\Delta t} = (\sqrt{v_t} + \frac{1}{2}\xi\sqrt{\Delta t} W_1)^2 - a(v_t - b)\Delta t - \frac{\xi^2}{4}\Delta t$$

$$S_{t+\Delta t} = S_t exp[(\mu - \frac{1}{2}v_t)\Delta t + \sqrt{v_t \Delta t} W_2]$$

- To generate W_1 and W_2 with correlation ρ, we first generate two independent standard normal variables Z_1 and Z_2, set $W_1 = Z_1$, then

$$W_2 = \rho Z_1 + \sqrt{1 - \rho^2} Z_2$$

0.7 Experiments

To evaluate the efficiency of our model and have a clearer sense of how our method is working, we start with synthetic data that perfectly follows the models instead of jumping into the real data set. Running the experiment on synthetic data tells us how reliable the program is and also how accurate the results are in the best-case scenario. The real data set is tested with both the Black-Scholes and Heston model afterward. In each set of experiments, we tried different sets of initial values, step size, and the number of iterations to evaluate the reliability of the model and its convergence rate in different settings respectively.

Black-Sholes

We ran a set of experiments with the different input values including the initial values of $\theta = (r, \sigma)$, the number of iteration N and stepsize ϵ on synthetic data generated based on Black-Sholes model and described in section 6.1.1. The program converged to the target values σ and r used in generating the data. Figure 4 plots the values of r and σ against iteration step. Furthermore, Figure 3 plotted the curve generated with the optimal values of r and σ and the call prices from the data.

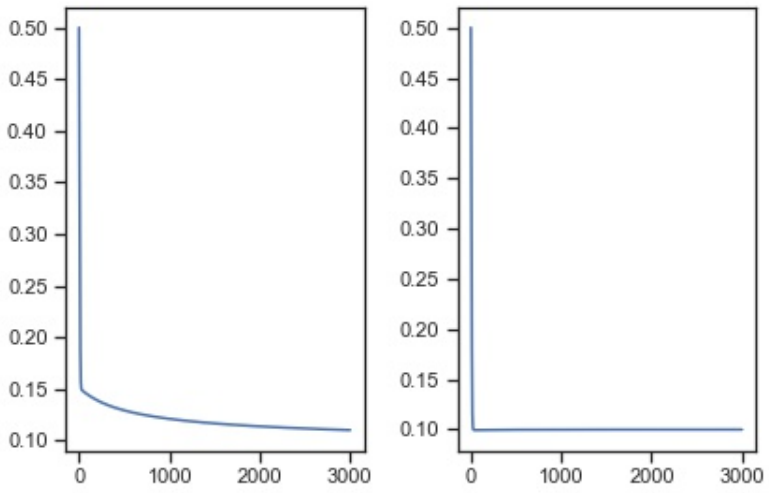


Figure 1: Shows the sequence of r and σ with the input values of $\theta = (0.1, 0.4)$, $S_0 = 31.51$ extracted from data, $k = 34.0$, $\epsilon = 0.001$, N = 500 with $T = 24$. The subplot on the left represents the values of σ converged to 0.0313 and the plot on the right is the sequence of rs converged to 0.012.

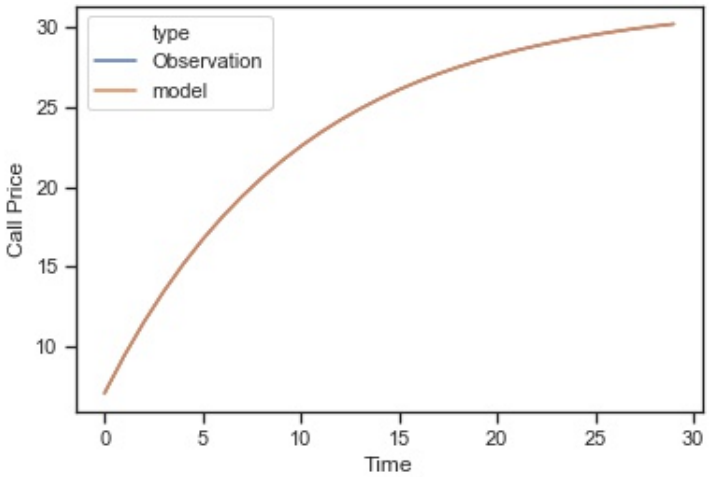


Figure 2: Shows the sequence of r and σ with the input values of $\theta = (0.1, 0.4)$, $S_0 = 31.51$ extracted from data, $k = 34.0$, $\epsilon = 0.001$, N = 500 with $T = 24$. The subplot on the left represents the values of σ converged to 0.0313 and the plot on the right is the sequence of rs converged to 0.012.

Using the Bloomberg dataset, we ran a set of experiments with the different input values including the initial values of $\theta = (r, \sigma)$, the number of iteration N and stepsize ϵ. The program converged to the optimal values of 0.0313 and 0.012 for σ and r respectively. Figure 4 plots the values of r and σ against iteration step. Furthermore, Figure 3 plotted the curve generated with the optimal values of r and σ and the call prices from the data.

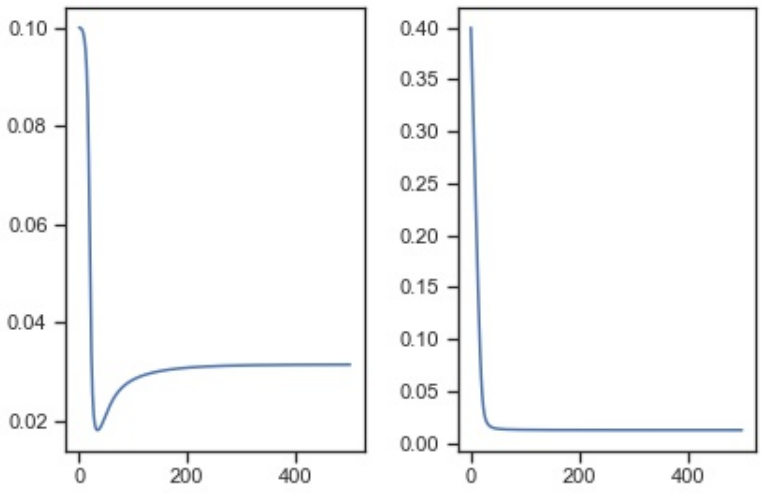


Figure 3: Shows the sequence of r and σ with the input values of $\theta = (0.1, 0.4)$, $S_0 = 31.51$ extracted from data, $k = 34.0$, $\epsilon = 0.001$, N = 500 with $T = 24$. The subplot on the left represents the values of σ converged to 0.0313 and the plot on the right is the sequence of rs converged to 0.012.

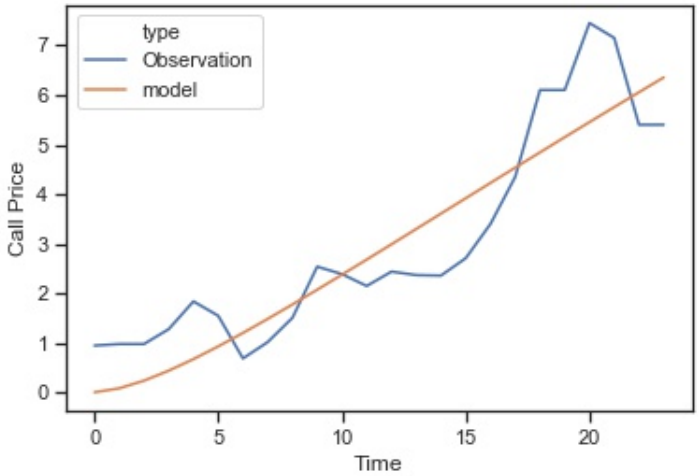


Figure 4: The call prices from the data plotted as observation(the blue line) and the model generated using the optimal values of r and σ in Figure 4 are represented.

Fixing all the other inputs we test the sensitivity of program convergence to step size. The results are plotted in Figure 5 in which as expected the smaller ϵs converges slower but smoother in comparison with larger ones. On the other hand, for σ values, the larger epsilons find the proximity of solution so fast but cannot find the solution with high accuracy. Furthermore, it is clear in Figure 5 that with smaller ϵs larger iteration number N required to make sure the sequence converges. In order to prove the insensitivity of program to initial values of σ and r, we ran a set of experiments with 9 different starting point. Figure **??** represent the convergence of r and σ with different initial points.

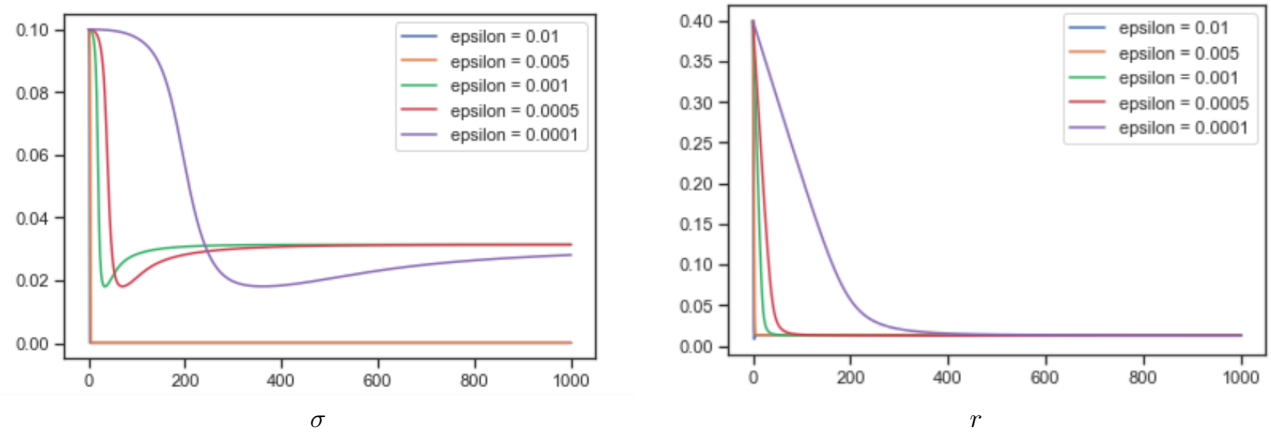


Figure 5: Shows the convergence of r and σ with different values of ϵ.

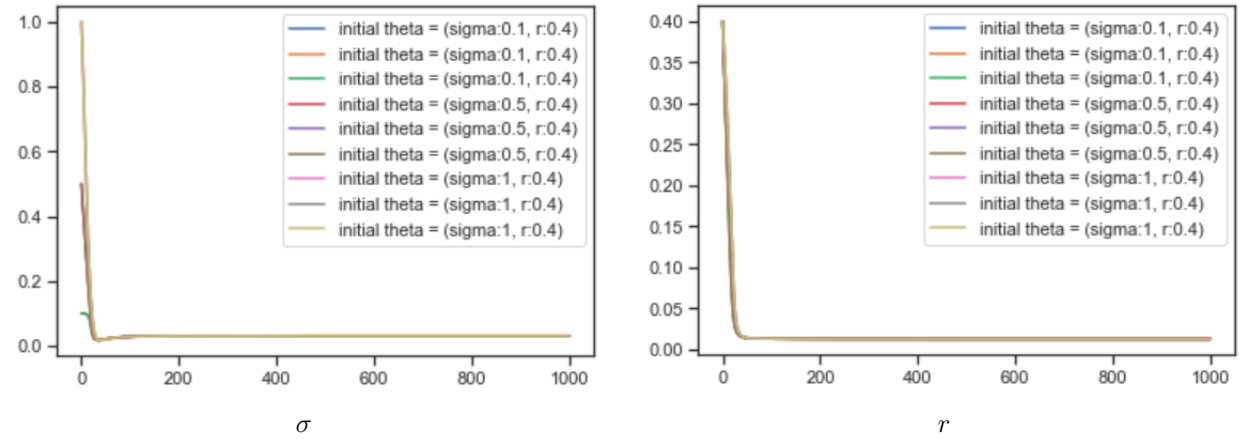


Figure 6: Shows the convergence of r and σ with different sets of initial points $\theta = (\sigma, r)$.

Min-batch Heston

Check the convergence of sequence In this experiment, we set the target parameters as μ and b, and see the convergence of sequence for those parameters regardless of initial values. For the sake of simplicity, we simulated the stock price. Since we simulate stock price data and calibrate the Heston model on our data, we don't have to worry about the uncertainty of stock price. Also, this makes the experiment simpler because we don't have to worry about the positive function in our cost function (i.e.) $(\mathrm{St}(\theta) - \mathrm{Kt})_+$.

For the Min-batch Heston with synthetic data, according to the lectures and Chapters 7 and 8 of the textbook, they would converge to the optimal values. As a result, we chose the time period of 10, assuming that the target companies like Jean Street update the parameters of their model every 2 weeks excluding weekdays. (Result) Min-batch Heston

Initial input values: step-size = 0.001, # of iteration = 10000, T = 10, batch size = 30, s0 = 20, v0 = 0.05

Initial target values: b = 3 and 1, μ = 0.03 and 0.07 Optimal target values: b = 2, μ = 0.05

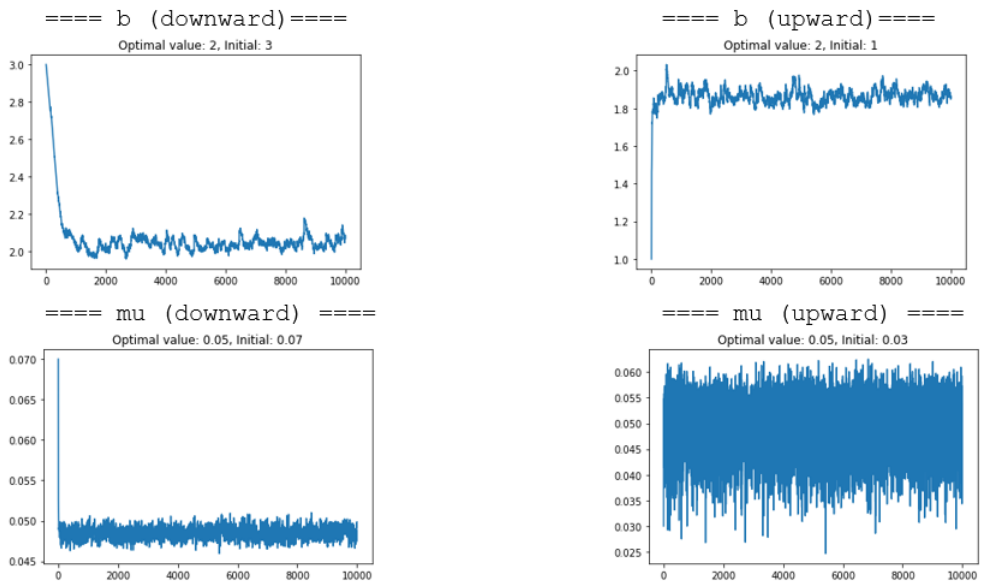
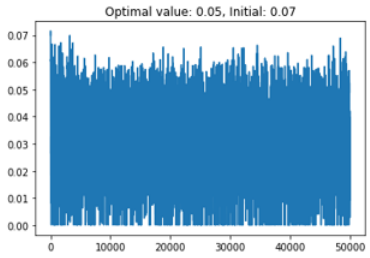


Figure 7: Caption

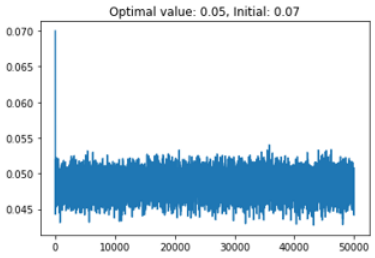
As the graphs show, the process converges to the optimal value of μ regardless of its initial value. Also, the process converges near the optimal value but not quite exactly.
Try different batch-sizes
In this experiment, we compare the behavior of Min-batch Heston model with different batch-sizes. According to the lectures, the smaller batch-sizes yield the higher variance but faster execution time and vice versa. In this experiment, we compared them by computation time and variance in plots.
(Result) Min-batch Heston with different batch-sizes
Initial input values: step-size = 0.0000001, # of iteration = 50000, T = 10, batch size = [1, 50, 100, 200], a = 1, b = 0.005, xi = 0.02, s0 = 20, v0 = 0.05
Initial target values: $\mu = 0.07$, a = 0.01, b = 0.03, $\zeta = 0.03$
Optimal target values: $\mu = 0.05$, a = 0.02, b = 0.02, $\zeta = 0.02$

==== batch_size=1 ====



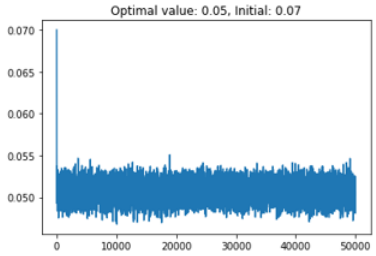
Computation time= 7sec

==== batch_size=50 ====



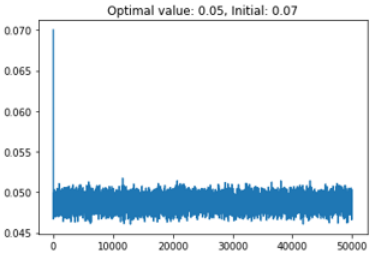
Computation time= 2min 23sec

==== batch_size=100 ====



Computation time= 9min 29sec

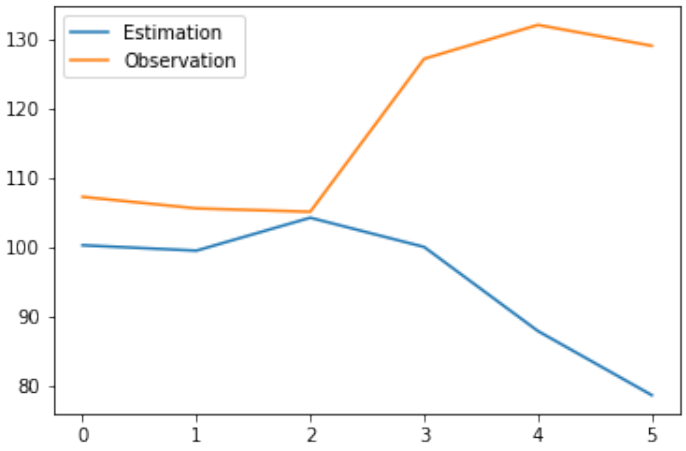
==== batch_size=200 ====



Computation time= 14min 36sec

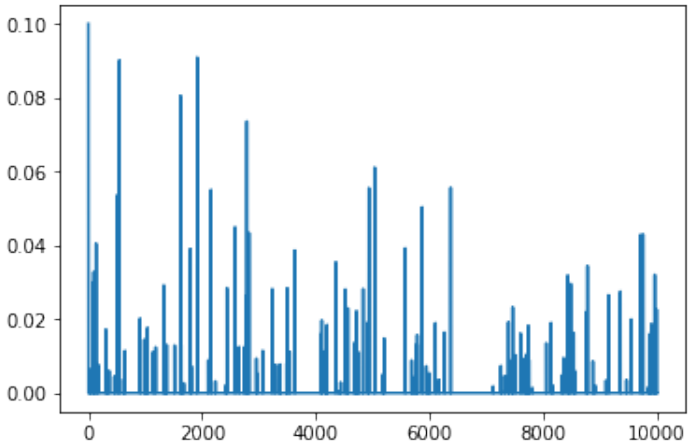
(2)

As we learned from the lectures, the graph clearly shows that the variance dramatically can be reduced by increasing the number of batches. However, the computation time also significantly increases. It takes almost 15 minutes to finishes the whole process with batch size of 200, though the variance seems not much different from the variance with batch size of 100. As Chris mentioned in his presentation, comparing the decreasing rate of variance along with computation time would be an interesting research question. Applying Heston model on real data: As we described in the Black-Scholes experiment, we use a dataset from Bloomberg. We used the call price of $CHTR given by a market maker such as Jane Street. The price observation and estimations given by the model are the following:



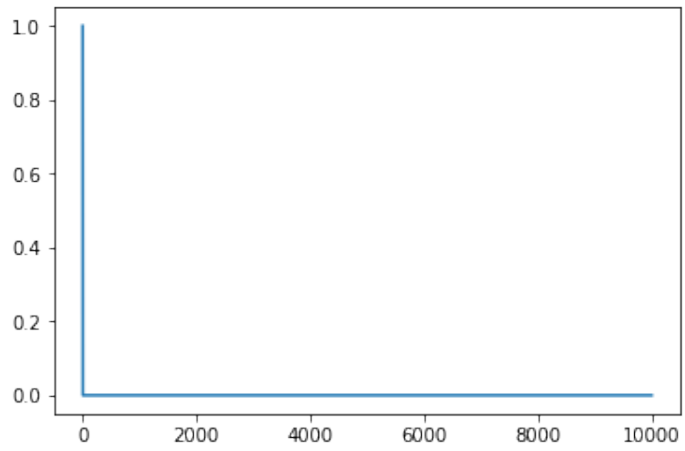
(3)

The process of b is the following:



(4)

The process of μ is the following:



(5)

Although the optimal value of b converges around 0.02, the value of μ becomes 0. The estimation is close to the observation, but they fall apart as the time goes by.

0.8 Conclusion

Conclusions from out initial optimization tests showed that for the BSM model, parameters μ and σ converged to optimal values in synthetic data. Estimation of constant parameters showed some correlation with Call prices offered for $AMD.

Results from the Heston model showed that it is possible to estimate parameters μ and b provided a and ξ are known. While the convergence happened quickly, ¡500 iterations, the variability of the steady state stochastic process begs the question of variance reduction through batched iterations. Results from the min-batch experiments did attribute to significant variance reduction but as expected significantly increased computation.

0.9 Future Work

Immediate future work includes setting confidence intervals for the Heston parameter estimation algorithm. If this model were to be further developed for trading, bounds on estimation error would need to be addressed.

Convergence speed would have to be improved without sacrificing variance. This could be done using variance reduction techniques found in chapter 6, as well as starting from initial values of μ, a, b and ξ derived from historic data (i.e., guessing initial b, by estimating the long term variance over a a year, and guessing initial a from the frequency at which the observed call price crosses the BSM call price estimation).

Other future work includes finding converging algorithms for a, and ξ, and their appropriate confidence interval after convergence. This framework can be expanded to other models found in mathematical finance, and given the robustness of the IPA estimator; it is a powerful tool for facile model parameterization.

References

1. S.E. Shreve, Stochastic Calculus for Finance II: Continuous-Time Models, Springer, New York, 2004

2. J. Gatheral , The Volatility Surface: A Practicioner's Guide, Wiley, New Jerset, 2006, p. 15-24

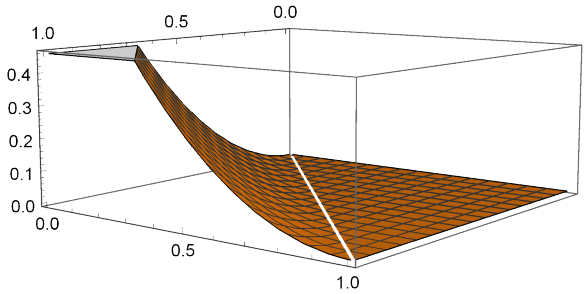
3. Vasquez-Abad, Felisa. "Stochastic Optimization" Felisa Vasquez-Abad, Hunter College (CUNY), 12/23/2019, http://cs.hunter.cuny.edu/ felisav/St-Opt/course-material/index.html.

4. Bloomberg L.P., "Options Chain of AMD, CHTR. 10/22/2019 to 11/23/2019," Bloomberg 2019.

Appendix

In[ ]:=
```
(*lipshitz of h*)
h[x_, K_] = (Max[x - K, 0])^2;
Plot3D[h[x, K], {x, 0, 1}, {K, 0, 1}]
```

Out[ ]=



In[ ]:=
```
(* showing lipschitz continuity *)
(* define the iterative price process of Heston *)
Z1 = RandomVariate[NormalDistribution[]];
Z2 = RandomVariate[NormalDistribution[]];
W1 = Z1;
W2 = ρ * Z1 + Sqrt[1 - ρ^2] * Z2;
S0 = 1;
v0 = 0.05;
dt = 0.01;
μ0 = 0.07;
ρ = 0;
S0pdt[μ0] = S0 * Exp[Sqrt[v0] * W2 + (μ0 - (1 / 2) * v0) * dt]
S0pdtmps[μ0, s_] = S0 * Exp[Sqrt[v0] * W2 + ((μ0 + s) - (1 / 2) * v0) * dt]

Plot[Abs[S0pdt[μ0] - S0pdtmps[μ0, s]], {s, 0, 0.00000000001}]
Reduce[Abs[S0pdt[μ0] - S0pdtmps[μ0, s]] ≤ s, {s}]
```

Out[ ]= 0.740177

Out[ ]= $e^{-0.301315+0.01\ (0.045+s)}$

Reduce: Reduce was unable to solve the system with inexact coefficients. The answer was obtained by solving a corresponding exact system and numericizing the result.

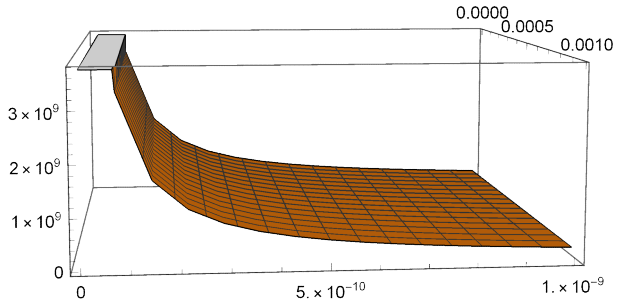
Out[ ]= $0.005516 \le s \le 681.954$

In[ ]:=
```
Clear[μ]
S0pdt[μ_] = S0 * Exp[Sqrt[v0] * W2 + (μ - (1 / 2) * v0) * dt]
S0pdtmps[μ_, s_] = S0 * Exp[Sqrt[v0] * W2 + ((μ + s) - (1 / 2) * v0) * dt]
Plot3D[(Abs[S0pdtmps[μ, s] - S0pdt[μ]]) / s, {μ, 0, 0.001}, {s, 0, 0.000000001}]
```

Out[ ]= $e^{-0.301315+0.01\ (-0.025+\mu)}$

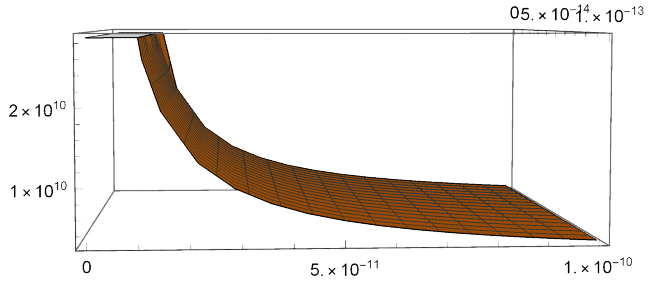
Out[ ]= $e^{-0.301315+0.01\ (-0.025+s+\mu)}$

Out[ ]=



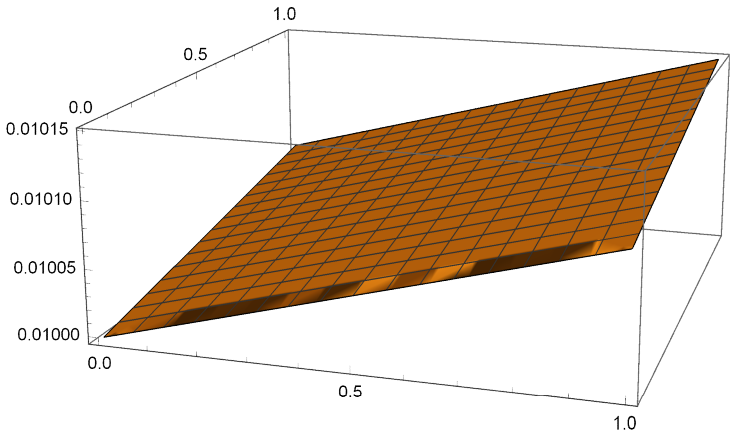
In[ ]:=
```
Plot3D[Abs[Log[(S0pdtmps[μ, s] / S0)] - Log[(S0pdt[μ] / S0)]] / s,
 {μ, 0, 0.0000000000001}, {s, 0, 0.0000000001}]
```

Out[ ]=




```
(* doing an expecation*)
ES0pdt[μ_] = S0 * Exp[ (μ - (1 / 2) * v0) * dt];
ES0pdtmps[μ_, s_] = S0 * Exp[((μ + s) - (1 / 2) * v0) * dt];
Plot3D[Abs[ES0pdt[μ] - ES0pdtmps[μ, s]] / s, {μ, 0, 1}, {s, 0, 1}]
```

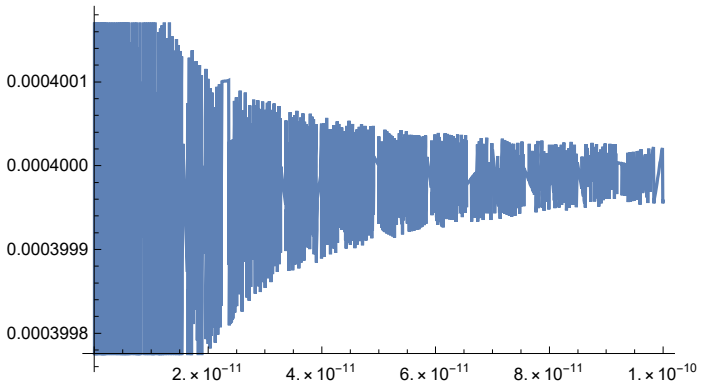
Out[ ]=



In[ ]:=
```
(* trying for variance procedures *)
Clear[s]
b = 0.01;
ξ = 0.01;
v0pdt1[a_] = ((Sqrt[v0] + (ξ / 2) * Sqrt[dt] W1) ^ 2) - a * (v0 - b) * dt - (dt / 4) * (ξ ^ 2);
v0pdt1[1]
v0pdtps[a_, s_] =
  ((Sqrt[v0] + (ξ / 2) * Sqrt[dt] W1) ^ 2) - (a + s) * (v0 - b) * dt - (dt / 4) * (ξ ^ 2);
Plot[Abs[v0pdt1[0.05] - v0pdtps[0.05, s]] / s, {s, 0, 0.0000000001}]
```

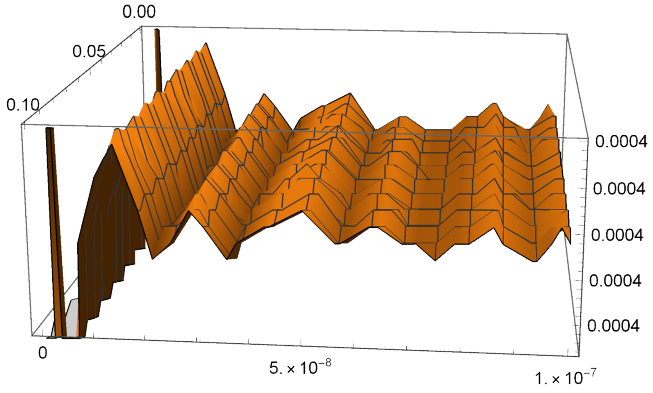
Out[ ]=
```
0.049657
```

Out[ ]=



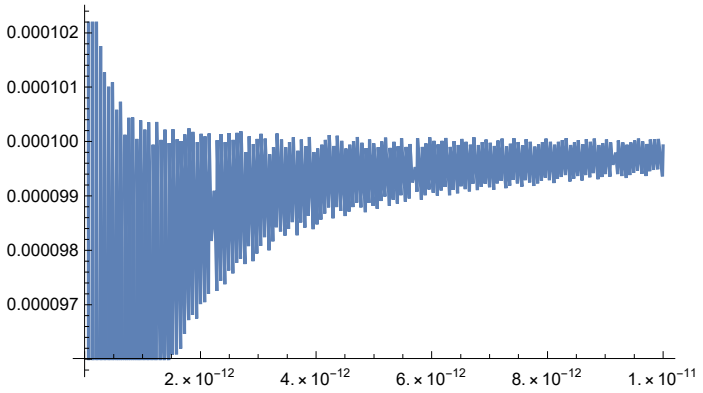
In[ ]:=
```
(* doing the lipshitz continuity of the variance process and a *)
Plot3D[Abs[v0pdtps[a, s] - v0pdt1[a]] / s, {a, 0.0001, 0.1}, {s, 0, 0.0000001}]
```

Out[ ]=



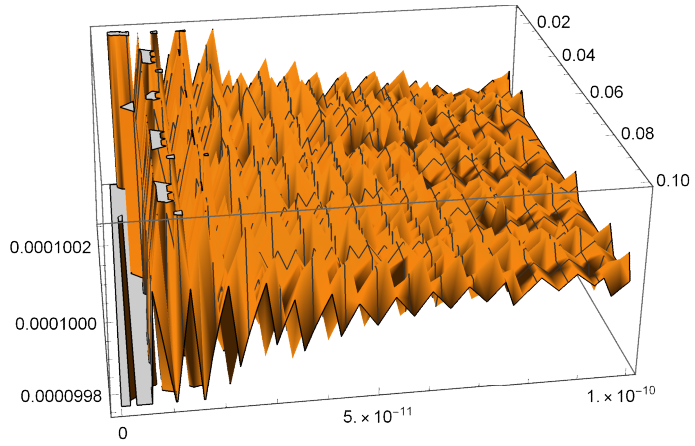
```
(* doing the lipshitz continuity of the variance process and b*)
Clear[b]
a = 0.01;
ξ = 0.01;
v0pdt2[b_] = ((Sqrt[v0] + (ξ / 2) * Sqrt[dt] W1) ^ 2) - a * (v0 - b) * dt - (dt / 4) * (ξ^2);
v0pdt2ps[b_, s_] =
  ((Sqrt[v0] + (ξ / 2) * Sqrt[dt] W1) ^ 2) - a * (v0 - (b + s)) * dt - (dt / 4) * (ξ^2);
Plot[Abs[v0pdt2[0.01] - v0pdt2ps[0.01, s]] / s, {s, 0, 0.00000000001}]
```

Out[ ]=



In[ ]:=
```
(*lets try varying s and b at the same time*)
Plot3D[Abs[v0pdt2ps[b, s] - v0pdt2[b]] / s, {b, 0.01, 0.1}, {s, 0, 0.0000000001}]
```

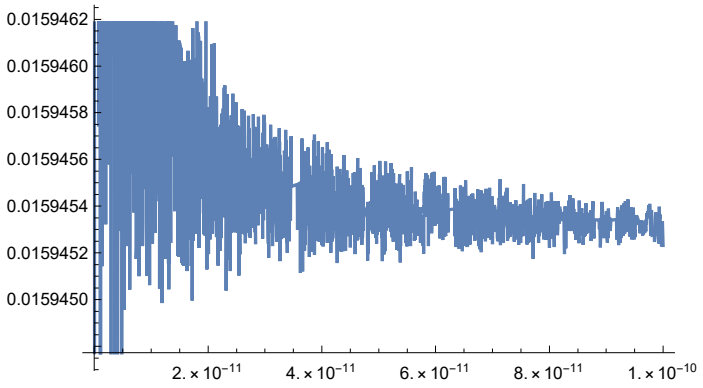
Out[ ]=



In[ ]:=
```
(*doing the lipshitz continuity of the variance process and ξ*)
Clear[ξ, a, b, μ]
μ = 0.07;
a = 0.02;
b = 0.02;
v0pdt3[ξ_] = ((Sqrt[v0] + (ξ / 2) * Sqrt[dt] W1) ^ 2) - a * (v0 - b) * dt - (dt / 4) * (ξ^2);
v0pdt3ps[ξ_, s_] =
  ((Sqrt[v0] + ((ξ + s) / 2) * Sqrt[dt] W1) ^ 2) - a * (v0 - b) * dt - (dt / 4) * ((ξ + s) ^ 2);
```

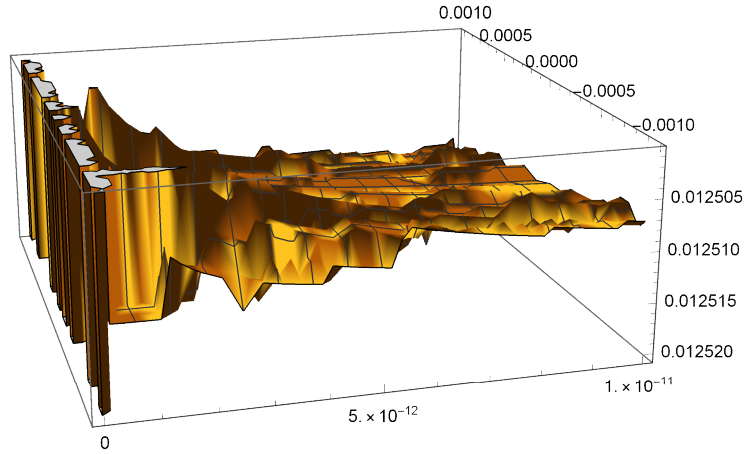
In[ ]:=
```
(*lets plot s for some ξ=1*)
ξ0 = 1;
Plot[Abs[v0pdt3[ξ0] - v0pdt3ps[ξ0, s]] / s, {s, 0, 0.0000000001}]
```

Out[ ]=



In[ ]:=
```
(* lets plot for varying ξ and s *)
Plot3D[Abs[v0pdt3ps[ξ, s] - v0pdt3[ξ]] / s, {ξ, -0.001, 0.001}, {s, 0, 0.00000000001}]
```

Out[ ]=



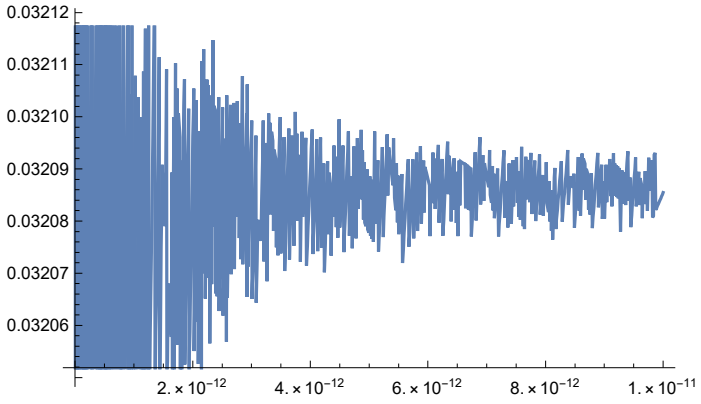
In[ ]:=
```
(* showing lipschitz continuity in time*)
(* define the iterative price process of Heston *)
Z1 = RandomVariate[NormalDistribution[]];
Z2 = RandomVariate[NormalDistribution[]];
W1 = Z1;
W2 = ρ * Z1 + Sqrt[1 - ρ^2] * Z2;
S0 = 1;
v0 = 0.05;
dt = 0.01;
μ0 = 0.07;
ρ = 0;
St[t_] = S0 * Exp[Sqrt[v0] * W2 + (μ0 - (1 / 2) * v0) * t]
Stps[t_, s_] = S0 * Exp[Sqrt[v0] * W2 + (μ0 - (1 / 2) * v0) * (t + s)]
```

Out[ ]= $e^{-0.383191+0.045\,t}$

Out[ ]= $e^{-0.383191+0.045\,(s+t)}$

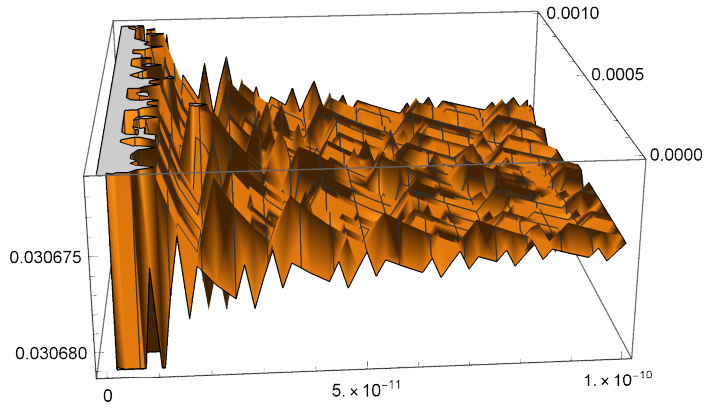
In[ ]:=
```
(*lets try plotting for some t*)
tbar = 1;
Plot[Abs[St[tbar] - Stps[tbar, s]] / s, {s, 0, 0.00000000001}]
```

Out[ ]=



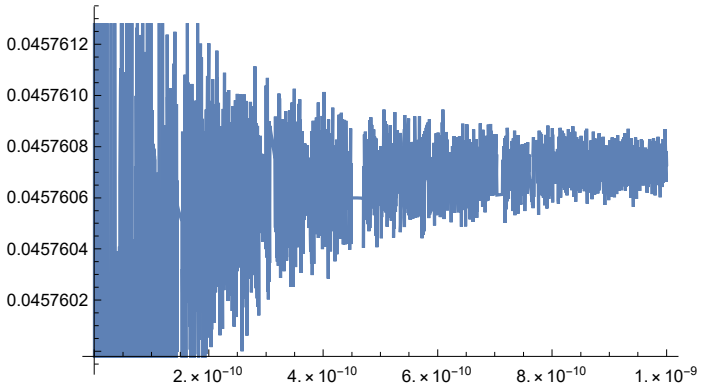
In[ ]:=
```
(* lets try plotting for some varyin t and s*)
Plot3D[Abs[St[t] - Stps[t, s]] / s, {t, 0, 0.001}, {s, 0, 0.0000000001}]
```

Out[ ]=



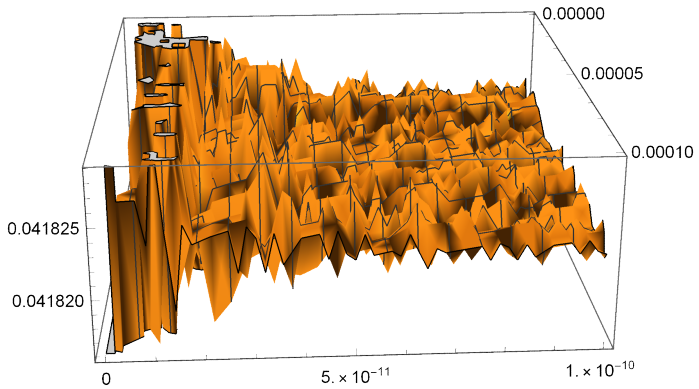
In[ ]:=
```
(* since it works for the price process in the
 context of the project lets try doing it for J theta *)
h[x_] = x^2;
Plot[Abs[h[St[tbar]] - h[Stps[tbar, s]]] / s, {s, 0, 0.000000001}]
```

Out[ ]=



In[ ]:=
```
(*now varying both t and s *)
Plot3D[Abs[h[St[t]] - h[Stps[t, s]]] / s, {t, 0, 0.0001}, {s, 0, 0.0000000001}]
```

Out[ ]=



In[ ]:=
```
(* showing lipschitz continuity *)
(* define the iterative price process of Heston *)
Z1 = RandomVariate[NormalDistribution[]];
Z2 = RandomVariate[NormalDistribution[]];
W1 = Z1;
W2 = ρ * Z1 + Sqrt[1 - ρ^2] * Z2;
S0 = 1;
v0 = 0.05;
dt = 0.01;
μ0 = 0.07;
ρ = 0;
S0pdt[μ0] = S0 * Exp[Sqrt[v0] * W2 + (μ0 - (1 / 2) * v0) * dt]
S0pdtmps[μ0, s_] = S0 * Exp[Sqrt[v0] * W2 + ((μ0 + s) - (1 / 2) * v0) * dt]

Plot[Abs[S0pdt[μ0] - S0pdtmps[μ0, s]], {s, 0, 0.00000000001}]
Reduce[Abs[S0pdt[μ0] - S0pdtmps[μ0, s]] ≤ s, {s}]
```

Out[ ]= 0.740177

Out[ ]= $e^{-0.301315+0.01\ (0.045+s)}$

Reduce: Reduce was unable to solve the system with inexact coefficients. The answer was obtained by solving a corresponding exact system and numericizing the result.

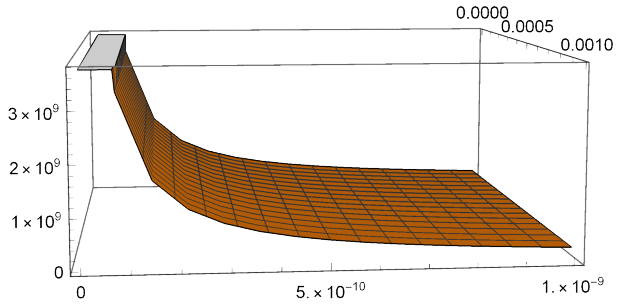
Out[ ]= $0.005516 \le s \le 681.954$

In[ ]:=
```
Clear[μ]
S0pdt[μ_] = S0 * Exp[Sqrt[v0] * W2 + (μ - (1 / 2) * v0) * dt]
S0pdtmps[μ_, s_] = S0 * Exp[Sqrt[v0] * W2 + ((μ + s) - (1 / 2) * v0) * dt]
Plot3D[(Abs[S0pdtmps[μ, s] - S0pdt[μ]]) / s, {μ, 0, 0.001}, {s, 0, 0.000000001}]
```

Out[ ]= $e^{-0.301315+0.01\ (-0.025+\mu)}$

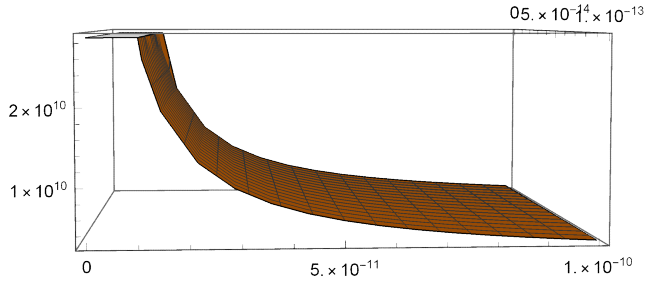
Out[ ]= $e^{-0.301315+0.01\ (-0.025+s+\mu)}$

Out[ ]=



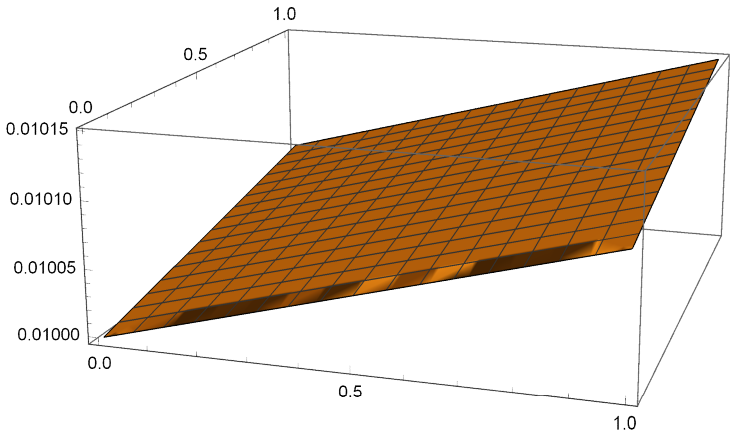
In[ ]:=
```
Plot3D[Abs[Log[(S0pdtmps[μ, s] / S0)] - Log[(S0pdt[μ] / S0)]] / s,
 {μ, 0, 0.0000000000001}, {s, 0, 0.0000000001}]
```

Out[ ]=




```
(* doing an expecation*)
ES0pdt[μ_] = S0 * Exp[ (μ - (1 / 2) * v0) * dt];
ES0pdtmps[μ_, s_] = S0 * Exp[((μ + s) - (1 / 2) * v0) * dt];
Plot3D[Abs[ES0pdt[μ] - ES0pdtmps[μ, s]] / s, {μ, 0, 1}, {s, 0, 1}]
```

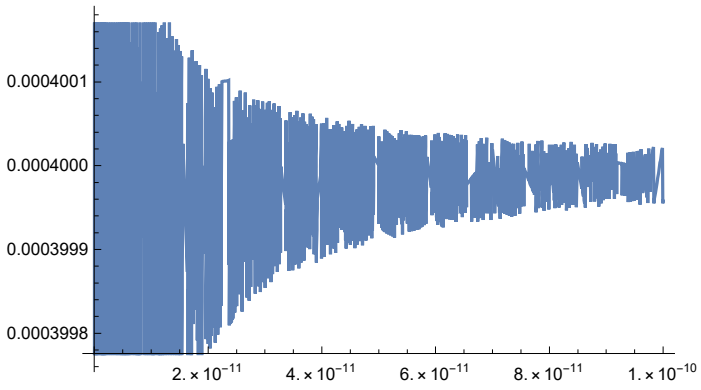
Out[ ]=



In[ ]:=
```
(* trying for variance procedures *)
Clear[s]
b = 0.01;
ξ = 0.01;
v0pdt1[a_] = ((Sqrt[v0] + (ξ / 2) * Sqrt[dt] W1) ^ 2) - a * (v0 - b) * dt - (dt / 4) * (ξ ^ 2);
v0pdt1[1]
v0pdtps[a_, s_] =
  ((Sqrt[v0] + (ξ / 2) * Sqrt[dt] W1) ^ 2) - (a + s) * (v0 - b) * dt - (dt / 4) * (ξ ^ 2);
Plot[Abs[v0pdt1[0.05] - v0pdtps[0.05, s]] / s, {s, 0, 0.0000000001}]
```

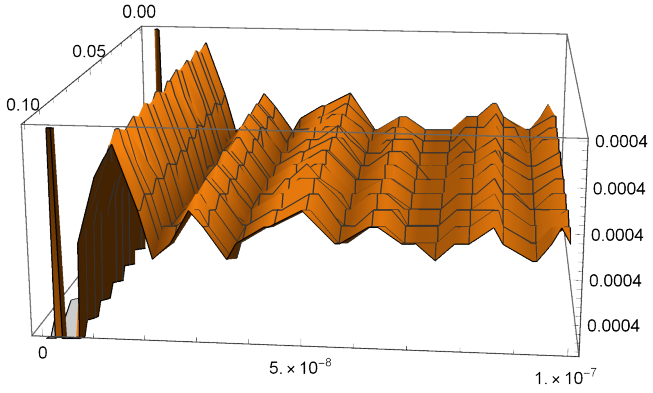
Out[ ]=
```
0.049657
```

Out[ ]=



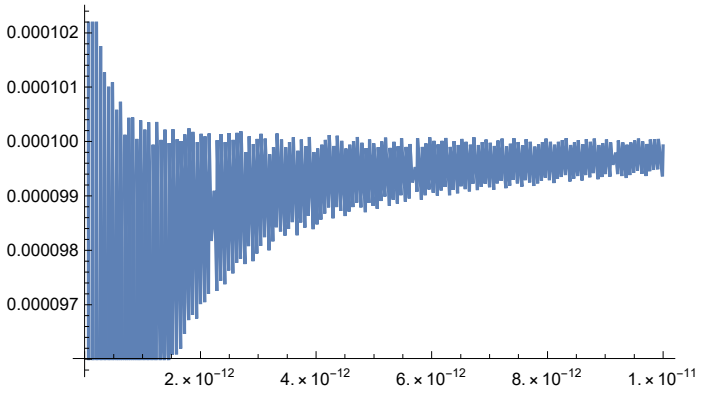
In[ ]:=
```
(* doing the lipshitz continuity of the variance process and a *)
Plot3D[Abs[v0pdtps[a, s] - v0pdt1[a]] / s, {a, 0.0001, 0.1}, {s, 0, 0.0000001}]
```

Out[ ]=



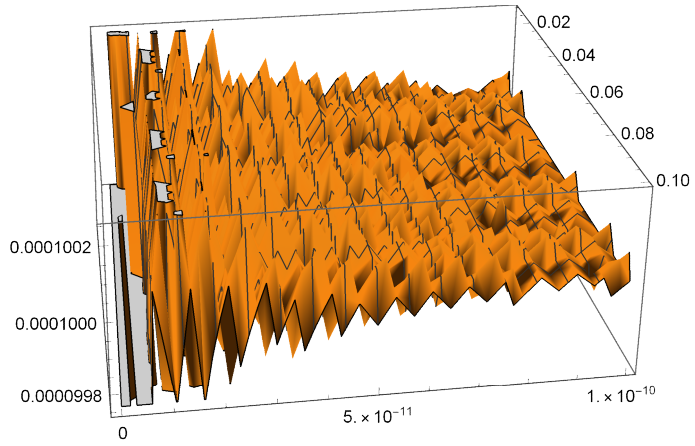
```
(* doing the lipshitz continuity of the variance process and b*)
Clear[b]
a = 0.01;
ξ = 0.01;
v0pdt2[b_] = ((Sqrt[v0] + (ξ / 2) * Sqrt[dt] W1) ^ 2) - a * (v0 - b) * dt - (dt / 4) * (ξ^2);
v0pdt2ps[b_, s_] =
  ((Sqrt[v0] + (ξ / 2) * Sqrt[dt] W1) ^ 2) - a * (v0 - (b + s)) * dt - (dt / 4) * (ξ^2);
Plot[Abs[v0pdt2[0.01] - v0pdt2ps[0.01, s]] / s, {s, 0, 0.00000000001}]
```

Out[ ]=



In[ ]:=
```
(*lets try varying s and b at the same time*)
Plot3D[Abs[v0pdt2ps[b, s] - v0pdt2[b]] / s, {b, 0.01, 0.1}, {s, 0, 0.0000000001}]
```

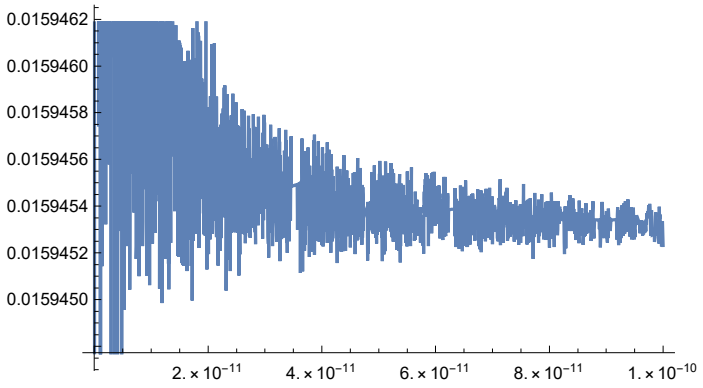
Out[ ]=



In[ ]:=
```
(*doing the lipshitz continuity of the variance process and ξ*)
Clear[ξ, a, b, μ]
μ = 0.07;
a = 0.02;
b = 0.02;
v0pdt3[ξ_] = ((Sqrt[v0] + (ξ / 2) * Sqrt[dt] W1) ^ 2) - a * (v0 - b) * dt - (dt / 4) * (ξ^2);
v0pdt3ps[ξ_, s_] =
  ((Sqrt[v0] + ((ξ + s) / 2) * Sqrt[dt] W1) ^ 2) - a * (v0 - b) * dt - (dt / 4) * ((ξ + s) ^ 2);
```

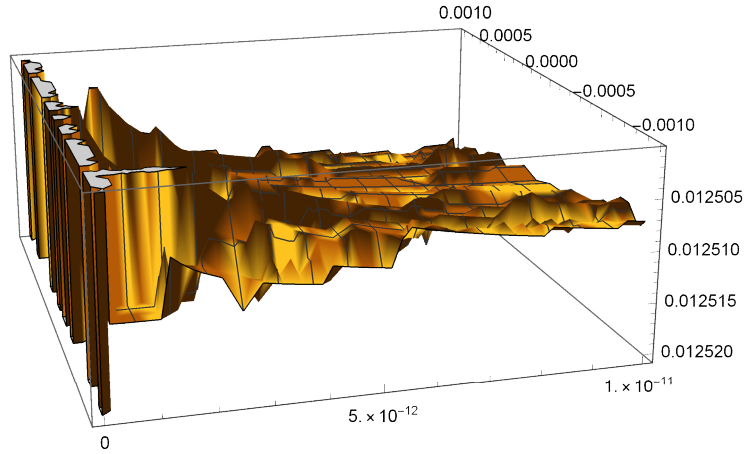
In[ ]:=
```
(*lets plot s for some ξ=1*)
ξ0 = 1;
Plot[Abs[v0pdt3[ξ0] - v0pdt3ps[ξ0, s]] / s, {s, 0, 0.0000000001}]
```

Out[ ]=



In[ ]:=
```
(* lets plot for varying ξ and s *)
Plot3D[Abs[v0pdt3ps[ξ, s] - v0pdt3[ξ]] / s, {ξ, -0.001, 0.001}, {s, 0, 0.00000000001}]
```

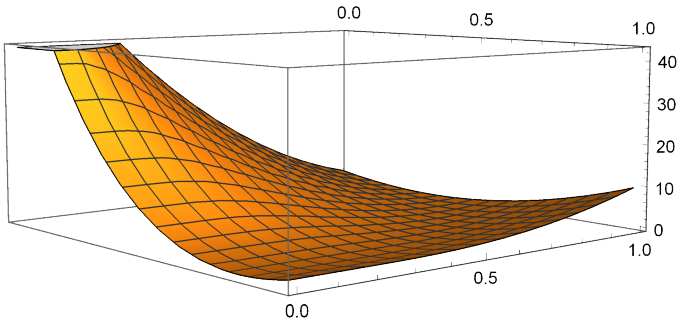
Out[ ]=



In[97]:=
```
(*black scholes merton conceptual proof of Theorem 1.1*)
S0 = 35;
K0 = 35;
C0 = 12;
τ = 0.5
d1[r_, σ_] = (1 / (σ Sqrt[τ])) (Log[S0 / K0] + (r + 0.5 σ^2) τ);
d2[r_, σ_] = d1[r, σ] - σ Sqrt[τ];
No[z_] = CDF[NormalDistribution[], z];
(*trying to find a surface *)
C01[r_, σ_] = S0 * No[d1[r, σ]] - K0 * Exp[-r * τ] No[d2[r, σ]];
J[r_, σ_] = 0.5 (C01[r, σ] - C0)^2;
Plot3D[J[r, σ], {r, 0, 1}, {σ, 0, 1}]
Gr[r_, σ_] = D[J[r, σ], {r}];
Gσ[r_, σ_] = D[J[r, σ], {σ}];
```

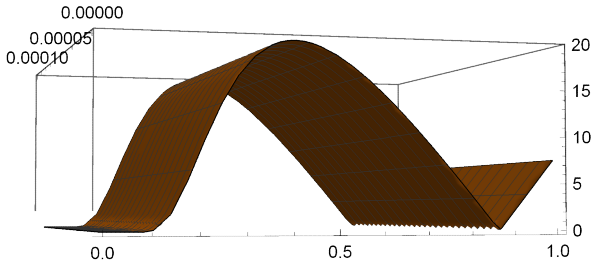
Out[100]= 0.5

Out[106]=



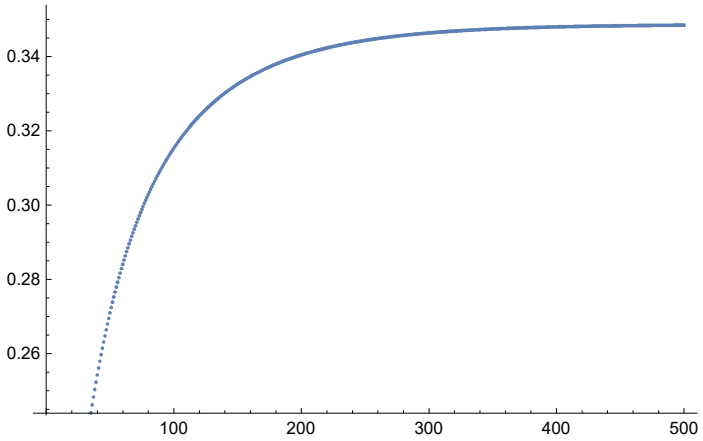
In[109]:=
```
(* theorem 1.4 sketch of lipschitz continuity*)
r0 = 0.5;

Plot3D[Abs[J[r0, (σ + δ)] - J[r0, σ]] / δ, {σ, 0, 1}, {δ, 0, 0.0001}]
```

Out[110]=



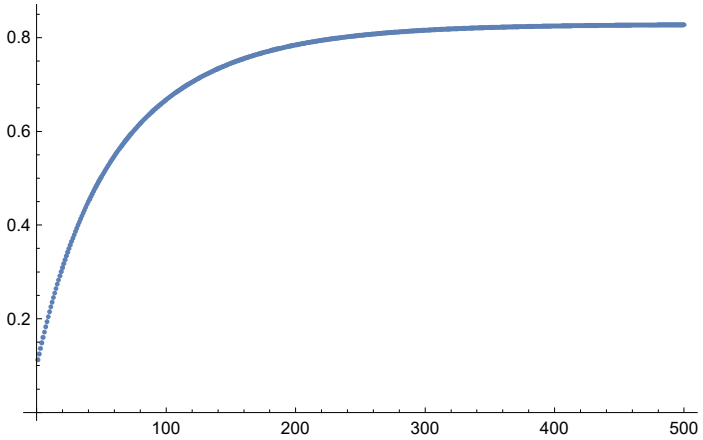
```
rslist = {};
r0 = 0.1;
σ0 = 0.1;
ϵ = 0.0001;
iterlimit  =  500;
For[i = 0, i < iterlimit, i++,
  rn = r0 - ϵ * Gr[r0, σ0];
  σn = σ0 - ϵ * Gσ[r0, σ0];
  r0 = rn;
  σ0 = σn;
  AppendTo[rslist, {i, σ0, r0}];
 ];
(*plotting r*)
ListPlot[rslist[[1 ;; 500, 2]]]
```

Out[159]=




```
(*plotting sigma*)
ListPlot[rslist[[1 ;; 500, 3]]]
```

Out[161]=



In[ ]:=
```
(* heston derivative tests*)
v0 = 0.5;
a0 = 1;
b0 = 0.54;
ξ0 = 0.01;
W1 = RandomVariate[NormalDistribution[]]
```

Out[ ]=
```
-0.231193
```

In[ ]:=
```
A[x_] = ξ * W1 * Sqrt[x];
EA[x_] = 0;
B[x_] = 1 - a + 0.5 * ξ * W1 * (1 / Sqrt[x]);
EB[x_] = 1 - a;
F1[x_] =  b - x;
EF1[x_] = b - x;
F2[x_] = a;
EF2[x_] = a;
F3[x_] = W1 * Sqrt[x] + 0.5 * ξ * (W1^2 - 1);
EF3[x_] = 0;
```

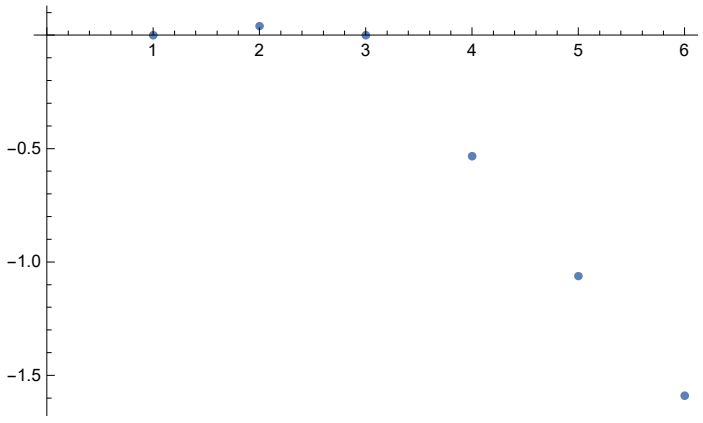
In[ ]:=
```
v1[a_, b_, ξ_] = v0 (1 - a) A[v0] + ((ξ^2) / 4) (W1^2 - 1) + a * b;
v2[a_, b_, ξ_] = v1[a, b, ξ] + A[v1[a, b, ξ]] - ((ξ^2) / 4) (W1^2 - 1) + a * b;
v3[a_, b_, ξ_] = v2[a, b, ξ] + A[v2[a, b, ξ]] - ((ξ^2) / 4) (W1^2 - 1) + a * b;
v4[a_, b_, ξ_] = v3[a, b, ξ] + A[v3[a, b, ξ]] - ((ξ^2) / 4) (W1^2 - 1) + a * b;
v5[a_, b_, ξ_] = v4[a, b, ξ] + A[v4[a, b, ξ]] - ((ξ^2) / 4) (W1^2 - 1) + a * b;
With[{a = a0, b = b0, ξ = ξ0},
 {v0, v1[a, b, ξ], v2[a, b, ξ], v3[a, b, ξ], v4[a, b, ξ], v5[a, b, ξ]}]
```

Out[ ]=
```
{0.5, 0.539976, 1.0783, 1.61592, 2.15301, 2.68964}
```

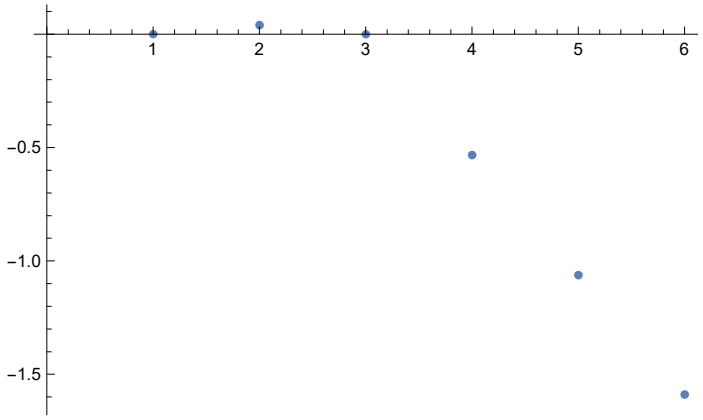
```
(*derivative process*)
(*wrt a*)
Z0 = 0;
Z1[a_, b_, ξ_] = Z0 * B[v0] + F1[v0];
Z2[a_, b_, ξ_] = Z1[a, b, ξ] * B[v1[a, b, ξ]] + F1[v1[a, b, ξ]];
Z3[a_, b_, ξ_] = Z2[a, b, ξ] * B[v2[a, b, ξ]] + F1[v2[a, b, ξ]];
Z4[a_, b_, ξ_] = Z3[a, b, ξ] * B[v3[a, b, ξ]] + F1[v3[a, b, ξ]];
Z5[a_, b_, ξ_] = Z4[a, b, ξ] * B[v4[a, b, ξ]] + F1[v4[a, b, ξ]];
dalist = With[{a = a0, b = b0, ξ = ξ0},
  {Z0, Z1[a, b, ξ], Z2[a, b, ξ], Z3[a, b, ξ], Z4[a, b, ξ], Z5[a, b, ξ]}]
ListPlot[dalist]
EZ1[]
EZ2
```

Out[ ]=
```
{0, 0.04, -0.000241171, -0.533363, -1.06211, -1.5893}
```

In[ ]:=



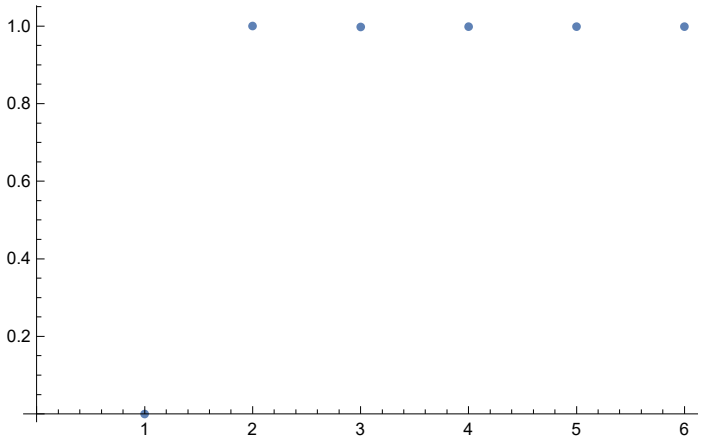
Out[ ]=



In[ ]:=
```
(*derivative process*)
(*wrt b*)
Z0 = 0;
Z1[a_, b_, ξ_] = Z0 * B[v0] + F2[v0];
Z2[a_, b_, ξ_] = Z1[a, b, ξ] * B[v1[a, b, ξ]] + F2[v1[a, b, ξ]];
Z3[a_, b_, ξ_] = Z2[a, b, ξ] * B[v2[a, b, ξ]] + F2[v2[a, b, ξ]];
Z4[a_, b_, ξ_] = Z3[a, b, ξ] * B[v3[a, b, ξ]] + F2[v3[a, b, ξ]];
Z5[a_, b_, ξ_] = Z4[a, b, ξ] * B[v4[a, b, ξ]] + F2[v4[a, b, ξ]];
dblist = With[{a = a0, b = b0, ξ = ξ0},
  {Z0, Z1[a, b, ξ], Z2[a, b, ξ], Z3[a, b, ξ], Z4[a, b, ξ], Z5[a, b, ξ]}]
ListPlot[
 dblist]
```

Out[ ]=
```
{0, 1, 0.997708, 0.998381, 0.998676, 0.998853}
```

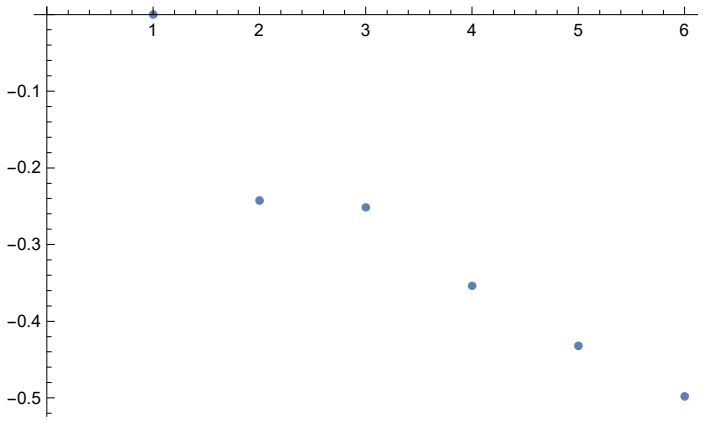
Out[ ]=



In[ ]:=
```
(*derivative process*)
(*wrt ξ*)
Z0 = 0;
Z1[a_, b_, ξ_] = Z0 * B[v0] + F3[v0];
Z2[a_, b_, ξ_] = Z1[a, b, ξ] * B[v1[a, b, ξ]] + F3[v1[a, b, ξ]];
Z3[a_, b_, ξ_] = Z2[a, b, ξ] * B[v2[a, b, ξ]] + F3[v2[a, b, ξ]];
Z4[a_, b_, ξ_] = Z3[a, b, ξ] * B[v3[a, b, ξ]] + F3[v3[a, b, ξ]];
Z5[a_, b_, ξ_] = Z4[a, b, ξ] * B[v4[a, b, ξ]] + F3[v4[a, b, ξ]];
dξlist = With[{a = a0, b = b0, ξ = ξ0},
  {Z0, Z1[a, b, ξ], Z2[a, b, ξ], Z3[a, b, ξ], Z4[a, b, ξ], Z5[a, b, ξ]}]
ListPlot[
 dξlist]
```

Out[ ]=
```
{0, -0.242657, -0.251442, -0.35374, -0.43198, -0.497906}
```

Out[ ]=



In[ ]:=
```
(* lets try to show that for synthetic data it will have a negative gradient *)
astar = 1;
bstar = 0.7;
ξstar = 0.2;
v0 = 0.5;
```

In[ ]:=
```
(*therefore the ideal variance process*)
vobs = With[{a = astar, b = bstar, ξ = ξstar},
  {v0, v1[a, b, ξ], v2[a, b, ξ], v3[a, b, ξ], v4[a, b, ξ], v5[a, b, ξ]}]
```

Out[ ]=
```
{0.5, 0.713108, 1.14326, 1.50508, 1.81899, 2.09584}
```

In[ ]:=
```
(* getting a J *)
(*1/2(5)sum0→5 E[(vguess - voptimal)^2]*)
vs = {0.5, 0.6911350203351362, 1.3439838486718274,
   1.9747348347545628, 2.588913718419046, 3.1893633907455308};
Exp
GJa[a_, b_, ξ_] =
 (1 / 3) * (Re[Total[{v0, v1[a, b, ξ], v2[a, b, ξ]}]] - (vs[[1]] + vs[[2]] + vs[[3]])) *
  Re[Total[{Z0, Z1[a, b, ξ], Z2[a, b, ξ], Z3[a, b, ξ], Z4[a, b, ξ], Z5[a, b, ξ]}]]

GJav1[a_, b_, ξ_] = (1 / 3) *
    (v1[a, b, ξ] + Re[v2[a, b, ξ]] - vs[[1]] - vs[[2]] - vs[[3]]) * (Z1[a, b, ξ] + Z2[a, b, ξ]);

alist = {};
a0 = 0.1;
ϵ = 0.01;
GJa[1, 1, 1]
For[i = 1, i < 1000, i++,
  an = a0;
  anp1 = an + ϵ * GJav1[an, 0.1, -0.1];
  a0 = anp1;
  AppendTo[alist, anp1]
 ];
ListPlot[alist]
Print[alist];
```

Out[ ]= Exp

Out[ ]=
$$\frac{1}{3}\left(-2.03512+\text{Re}\left[3\,a\,b-0.163478\,(1-a)\,\xi-0.236637\,\xi^2-0.231193\,\xi\sqrt{a\,b-0.0817392\,(1-a)\,\xi-0.236637\,\xi^2}\,\right]\right)$$
$$\text{Re}[Z0+Z1[a,b,\xi]+Z2[a,b,\xi]+Z3[a,b,\xi]+Z4[a,b,\xi]+Z5[a,b,\xi]]$$

Out[ ]= 0.175416 Re[Z0 + Z1[1, 1, 1] + Z2[1, 1, 1] + Z3[1, 1, 1] + Z4[1, 1, 1] + Z5[1, 1, 1]]