Simulation Methods Comparison and Parameter Estimation for a Fractional Stochastic Volatility Model with Application in Stock Price Analysis

By

Fengmei Wu

Submitted to the graduate degree program in Mathematics and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Arts.

_____

Chairperson：David Nualart

_____

Yaozhong Hu

_____

Bozenna Pasik-Duncan

Date Defended: May 13, 2013

The Thesis Committee for Fengmei Wu

certifies that this is the approved version of the following thesis:

Simulation Methods Comparison and Parameter Estimation for a Fractional Stochastic Volatility Model with Application in Stock Price Analysis

_____

Chairperson: David Nualart

Date approved: May 13, 2013

Simulation Methods Comparison and Parameter Estimation for a Fractional Stochastic Volatility Model with Application in Stock Price Analysis

Fengmei Wu
Advisor: David Nualart
University of Kansas

Abstract: This paper studies continuous-time stock pricing models with stochastic volatility driven by fractional Brownian motion. We compare two ways for simulating the paths of stochastic volatility and stock price when the Hurst parameter of fractional Brown motion is between 0.5 and 1. The first approach, is to use truncated fractional Brownian motion to approximate the fractional Brownian motion and estimate the volatility by Monte Carlo integral and symbolic integral. In the second one, Euler method is employed in simulation, without truncating the fractional Brownian process. Simulating the fractional Brownian motion in the second approach, we use spectral representation. Simulation results show that the latter is more efficient than using the symbolic integral and Monte Carlo integral is the worst. The application of the stochastic model is illustrated through real financial data.

Keyword: fractional Brownian motion, truncated fractional Brownian motion, stochastic volatility, Monte Carlo Integral, spectral representation

Acknowlegements

Grateful thanks to Professor David Nualart for thought-provoking supervision, constructive suggestions on weekly meeting for this thesis, and his endless patience to tutor me to read a series of books on mathematical finance. Prof. Nualart has showed me by living example how to be a dedicated scholar.

Special thanks to Prof. Bozenna Pasik-Duncan for her encouragements and supports, in both tangible and intangible way. I have learned from her to be a positive person, to pursue goals relentlessly, and to stretch talents to extremes.

I am indebted to my family members, who are always there for me. They encourage me to become the person I dream to be.

I am lucky to have built up friendship with smart and nice people when I worked for my degree, on intellectual and/or spiritual base.

The challenging while rewarding study experience in the department will always be part of me.

Table of Contents

1 Introduction

To make investment decisions in financial markets, critical concerns are the returns and the risks. To capture the feature of risks, one measure is to calculate the volatilities. For security prices, especially bond prices, existence of long range dependence in volatilities is well-documented. With a slight abuse of concept, in this paper we will use stock price to represent security price. Backus and Zin (1995) used fractional difference model to model the features of the volatility of long term interest rate and introduced discrete bond pricing model. Continuous-time stochastic differential equations (SDEs) are most frequently employed in finance to quantitatize assets returns, interest rates and investment risks, and are extended to capture the feature of long dependence in the volatilities of stock prices.

In the basic case, we assume the interest rate and the volatility of stock price are time-invariant. In Black-Scholes-Merton (BSM) model for stock prices, with constant interest rate r and constant volatility σ, the stock price S_t is modeled by a geometric Brownian motion (GBM)

$$dS_t = rS_t dt + \sigma S_t dW_t$$

where W_t is a standard Brownian motion(SBM).

The stock price is

$$S_t = S_0 \exp\{\sigma W_t + (r - \frac{1}{2}\sigma^2)t\}.$$

Generally, the interest rate and volatility are time-varying. Thus, with instantaneous rate of return $r(t)$ and volatility $\sigma(t)$, both deterministic, one obtains a generalized geometric Brownian motion model

$$dS_t = r(t)S_t dt + \sigma(t)S_t dW_t.$$

The stock price is given by

$$S_t = S(0)\exp\Big\{\int_0^t \sigma(s)dW_s + \int_0^t (r(s) - \frac{1}{2}\sigma^2(s))ds\Big\}.$$

In complicated stock markets, time-evolution volatilities, more often than not, are non-deterministic. In literature, stochastic differentiation equations are frequently applied to model continuous-time volatilities in finance. Hull and White (1987), Scott(1987) as well as Melino and Turnbull(1990) proposed a standard stochastic differentiation volatility model, in which the random part is driven by a standard Brownian motion. In reality, a standard stochastic differential process doesn't capture the characteristics of all volatilities. To model the volatility exhibiting long range dependence, it is natural to extend the standard stochastic differential volatility model to be fractional stochastic differential volatility(FSDV) model, which is driven by a fractional Brownian motion. For simplification, Comte and Renault (1998), proposed truncated fractional stochastic differential volatility (TFSDV) model for modeling the dependence.

The main focus of this research, is to investigate the quantitative aspects of a class of stock price models with fBm driven volatilities and improve the simulating method for the trajectories of stock prices and volatilities. We compare two simulating methods. One is to adopt truncated fractional Brownian motion to approximate fractional Brownian motion as the model in Comte and Renault (1998). We specify how Monte Carlo integral works in implementing the process. Also, we also use symbolic integral to calculate the integral in truncated fBm. The other approach is to employ Euler algorithm in simulating the paths of stock price and volatility, which is showed to be an improvement for the first method.

Another motivation of this research is to apply the stock price model driven by fractional Brownian motion to analyze real financial data. We estimate the parameter in fractional Brownian motion by periodgram method.

The paper is organized as follows. Stock price models with standard stochastic volatility model, fractional stochastic differential volatility (FSDV) model and truncated stochastic differentiation volatility (TFSDV) model are investigated in Section 2. Section 3 studies specific simulation proce-

dure for paths of stock price and volatility which is driven by truncated fBm. Section 4 specifies Euler method to improve the simulation by simulating fBm directly. Section 5 presents the methods for parameter estimation and applies the method to analyze real financial dataset. In section 6, we try to explore the trajectory of European call price.

2 Model

2.1 Fractional Brownian Motion

2.1.1 Standard Brownian Motion

Definition 2.1.1: Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space. A continuous real-valued process $(W_t)_{t\geq 0}$ is called a standard Brownian motion or Wiener process if

(i) $W_0 = 0$, and the sample paths of W_t are continuous function of t.

(ii)W_t has independent increments, i.e. for any sequence of times $0 \leq t_1 < t_2 < \cdots < t_n$, the increments $W_{t_1} - W_{t_0}, W_{t_2} - W_{t_1}, \cdots, W_{t_n} - W_{t_{n-1}}$ are independent random variables; W_t is time homogeneous, i.e. for $t \geq s$ the distribution of $W_t - W_s$ depends only on $t - s$, and $W_t - W_s \stackrel{d}{=} W_{t-s}$.

(iii) For $0 \leq s < t$,

$$W_t - W_s \sim N(0, t - s).$$

Definition 2.1.1'(n-dimensional): An n-dimensional stochastic process $(W_t)_{t\geq 0}$ is called a standard Brownian motion if

$$(W_t)_{t\geq 0} = (W_t^1, W_t^2, \cdots, W_t^n)_{t\geq 0},$$

where the processes $(W_t^i)_{t\geq 0}, i = 1, 2, \cdots, n$ are independent standard Brownian motions.

2.1.2 Fractional Brownian Motion

Definition 2.1.2: A continued-time Gaussian stochastic process W_t^H with Hurst parameter H, is called fraction Brownian motion (fBm) if

$$W_t^H = \frac{1}{\Gamma(H+\frac{1}{2})}\Big(\int_{-\infty}^{0}[(t-s)^{H-\frac{1}{2}} - (-s)^{H-\frac{1}{2}}]dW_s + \int_0^t (t-s)^{H-\frac{1}{2}}dW_s\Big),$$

where $\Gamma(\alpha) = \int_0^\infty x^{\alpha-1}exp(-x)dx$, $H \in (0,1)$ is called Hurst parameter, and $W = (W_t)_{t\geq 0}$ is a standard Brownian motion.

The covariance function of fBm is given by

$$\mathrm{Cov}_{W_H}(s,t) = \frac{1}{2}(t^{2H} + s^{2H} - |t-s|^{2H})$$

for every $s, t \in [0,T]$.

The fractional Brownian motion is a continuous Gaussian process with stationary increment and is self-similar. W_t^H is H self-similar, which means that $(\frac{1}{u^H}W_{ut}^H)_{t\geq 0}$ and $(W_t^H)_{t\geq 0}$ have identical probability distributions, for any $u \in \mathbb{R}^+$.

2.1.3 Truncated Fractional Brownian Motion, a Proxy to W_t^H

Definition 2.1.3: Let W_t be standard Brownian motion, and $\alpha = H - \frac{1}{2}$. Define

$$W_t^\alpha = \int_0^t \frac{(t-s)^\alpha}{\Gamma(1+\alpha)}dW_s.$$

Then W_t^α is a truncated fractional Brownian motion of order α.
Being said truncated, means that the first integral term in fractional Brownian motion $W_H(t)$ is chopped. $W_\alpha(t)$ is a simplified approximate form to W_t^H when $\alpha \neq 0$. If $\alpha = 0$, i.e. $H = \frac{1}{2}$, $W_t^\alpha = W_t^H = W_t$, a standard Brownian motion.

Unlike Brownian motion, W_t^α does not have stationary increments. However, the truncated Brownian motion has conditional stationary increments. Let $\mathcal{F}_t$ be a filtration generated by W_t:

$$\mathcal{F}_t = \sigma(W_s, 0 \leq s \leq t).$$

Following is to calculate the mean and the variance of increment for W_t^α conditional on $\mathcal{F}_t$.

Conditional on $\mathcal{F}_t$, the expectation of the increment $W_{t+h}^\alpha - W_t^\alpha$ is given by

$$\begin{aligned}
&\mathbb{E}\Big[W_{t+h}^\alpha - W_t^\alpha|\mathcal{F}_t\Big] \\
&= \mathbb{E}\Big[\int_0^{t+h}\frac{(t+h-s)^\alpha}{\Gamma(1+\alpha)}dW_s - \int_0^t\frac{(t-s)^\alpha}{\Gamma(1+\alpha)}dW_s\Big|\mathcal{F}_t\Big] \\
&= \mathbb{E}\Big[\int_t^{t+h}\frac{(t+h-s)^\alpha}{\Gamma(1+\alpha)}dW_s + \int_0^t\frac{(t+h-s)^\alpha}{\Gamma(1+\alpha)}dW_s - \int_0^t\frac{(t-s)^\alpha}{\Gamma(1+\alpha)}dW_s\Big|\mathcal{F}_t\Big] \\
&= \mathbb{E}\Big[\int_t^{t+h}\frac{(t+h-s)^\alpha}{\Gamma(1+\alpha)}dW_s\Big|\mathcal{F}_t\Big] + \mathbb{E}\Big[\int_0^t\frac{(t+h-s)^\alpha-(t-s)^\alpha}{\Gamma(1+\alpha)}dW_s\Big|\mathcal{F}_t\Big] \\
&= \mathbb{E}\Big[\int_0^t\frac{(t+h-s)^\alpha-(t-s)^\alpha}{\Gamma(1+\alpha)}dW_s\Big|\mathcal{F}_t\Big] \\
&= \int_0^t\frac{(t+h-s)^\alpha-(t-s)^\alpha}{\Gamma(1+\alpha)}dW_s.
\end{aligned}$$

Conditional on $\mathcal{F}_t$, the variance of the increment $W_{t+h}^\alpha - W_t^\alpha$ is given by

$$\begin{aligned}
&\mathbb{V}\text{ar}(W_{t+h}^\alpha - W_t^\alpha|\mathcal{F}_t) \\
&= \mathbb{E}\Big[\Big(W_{t+h}^\alpha - W_t^\alpha - \mathbb{E}(W_{t+h}^\alpha - W_t^\alpha|\mathcal{F}_t)\Big)^2\Big|\mathcal{F}_t\Big] \\
&= \mathbb{E}\Big[\Big(\int_0^{t+h}\frac{(t+h-s)^\alpha}{\Gamma(1+\alpha)}dW_s - \int_0^t\frac{(t-s)^\alpha}{\Gamma(1+\alpha)}dW_s - \int_0^t\frac{(t+h-s)^\alpha-(t-s)^\alpha}{\Gamma(1+\alpha)}dW_s\Big)^2\Big|\mathcal{F}_t\Big] \\
&= \mathbb{E}\Big[\Big(\int_0^{t+h}\frac{(t+h-s)^\alpha}{\Gamma(1+\alpha)}dW_s - \int_0^t\frac{(t+h-s)^\alpha}{\Gamma(1+\alpha)}dW_s\Big)^2\Big|\mathcal{F}_t\Big] \\
&= \mathbb{E}\Big[\Big(\int_t^{t+h}\frac{(t+h-s)^\alpha}{\Gamma(1+\alpha)}dW_s\Big)^2\Big|\mathcal{F}_t\Big] \\
&= \int_t^{t+h}\frac{(t+h-s)^{2\alpha}}{\Gamma(1+\alpha)^2}ds \\
&= \frac{h^{2\alpha+1}}{(2\alpha+1)\Gamma(1+\alpha)^2}.
\end{aligned}$$

2.2 Stock Price Model with Volatility Driven by Standard Brownian Motion

Hull and White (1987), Scott (1987), as well as Melino and Turnbull (1990) proposed a stock price model, by assuming that the volatility not only depends on time t but also an adapted stochastic process, which is the standard stochastic volatility(SSV) model

$$\begin{cases}\frac{dS(t)}{S(t)} = r(t,S(t))dt + \sigma(t)dW_t^1 \\ d(\ln\sigma(t)) = k(\theta - \ln\sigma(t))dt + \gamma dW_t^2,\end{cases} \tag{1}$$

where (W_t^1, W_t^2) is a bivariate Brownian process.

2.3 Stock Price Model with Volatility Driven by Fractional Brownian Motion

The above model can be extended, by replacing W_t^2 in the volatility equation by a fractional Brownian motion with Hurst parameter H, as follows.

$$\begin{cases} \frac{dS(t)}{S(t)} = r(t, S(t))dt + \sigma(t)dW_t^1 \\ d(\ln\sigma(t)) = k(\theta - \ln\sigma(t))dt + \gamma dW_t^H. \end{cases} \tag{2}$$

As such, the model accommodates to a wider range of volatility features.

2.4 Stock Price Model with Volatility Driven by Truncated Fractional Brownian Motion

By substituting the standard Brownian process W_t^2 with a truncated fractional Brownian motion(fBm) W_t^α of order α, Comte and Renault (1998) extended model (1) to the following model with volatility driven by truncated fractional Brownian motion.

$$\begin{cases} \frac{dS(t)}{S(t)} = r(t, S(t))dt + \sigma(t)dW_t^1 \\ d(\ln\sigma(t)) = k(\theta - \ln\sigma(t))dt + \gamma dW_t^\alpha. \end{cases} \tag{3}$$

For $-\frac{1}{2} < \alpha \leq 0$, the volatility is of short memory. With $0 < \alpha < \frac{1}{2}$, the model captures the persistent stochastic feature, or long memory feature of stochastic volatility.

3 Numerical Estimate of Stock Price Model with Volatility Driven by W_t^α

This section presents procedures to simulate the paths of the stock and truncated fBm driven volatility. In particular, Monte Carlo integral is specified.

3.1 Long-memory Volatility Driven by Truncated Fractional Brownian Process

Consider first-order fractional stochastic differential equation with long-memory

$$dX(t) = -kX(t)dt + \gamma dW_t^{\alpha}, x(0) = 0, k > 0, 0 < \alpha < \frac{1}{2}.$$

The solution is, $X(t) = \int_0^t e^{-k(t-s)}\gamma dW_t^{\alpha}$.

According to Comte and Renault (1996),

$$X(t) = \int_0^t A(t-s)dW_s,$$

where $A(t-s)$ is given by

$$A(x) = \frac{\gamma}{\Gamma(1+\alpha)}\frac{d}{dx}\Big[\int_0^x e^{-kx}(x-u)^{\alpha}du\Big] = \frac{\gamma}{\Gamma(1+\alpha)}\Big(x^{\alpha} - ke^{-kx}\int_0^x e^{ku}u^{\alpha}du\Big).$$

3.2 Approximate the Volatility Process

In the same spirit of Comte and Remault (1998), to approximate the volatility $X(t)$ process, discrete time numerical method is employed to estimate the integrals, specifically, using step functions to approximate integrands. Let $0 = t_0 < t_1 < \cdots < t_j < \cdots < t_{\frac{[nt]}{n}} \leq t$ be a partition of $[0,t]$. $j = 0, 1, \cdots, [nt]$, where $[nt]$ is the greatest integer not exceeding nt. In particular, in the rest of this paper, we divide $[0, \frac{[nt]}{n}]$ into equal subintervals with length $t_j - t_{j-1} = \frac{j}{n}$.

$$\begin{aligned} X(t) &= \int_0^t A(t-s)dW_s \\ &\approx \sum_{j=1}^{[nt]} A\Big(t - \frac{j-1}{n}\Big)\Delta W_{\frac{j}{n}} + A\Big(t - \frac{[nt]}{n}\Big)\Big(W_t - W_{\frac{[nt]}{n}}\Big), \end{aligned}$$

where

$$\Delta W_{j/n} = W_{j/n} - W_{(j-1)/n}.$$

For large n, $t - \frac{[nt]}{n}$ is very small, so that the last term in approximating expression of $X(t)$ is negligible. Thus,

$$X(t) \approx \widetilde{X}_n(t) = \sum_{j=1}^{[nt]} A\Big(t - \frac{j-1}{n}\Big)\Delta W_{\frac{j}{n}}.$$

3.3 Monte Carlo Method to Approximate $A(t - \frac{j-1}{n})$

To approximate above $X(t)$, we need to estimate the integral in $A(t-\frac{j-1}{n})$.

$$A(t - \frac{j-1}{n}) = \frac{\gamma}{\Gamma(1+\alpha)}\Big((t - \frac{j-1}{n})^{\alpha} - ke^{-k(t-\frac{j-1}{n})}\int_0^{t-\frac{j-1}{n}} e^{ku}u^{\alpha}du\Big).$$

Unless $\alpha = 0$, it is impossible to derive the integration analytically. An alternative approach to deal with the integral in $A(t-\frac{j-1}{n})$ is to using Monte Carlo Integration.

To compute $\theta = \int_0^1 f(x)dx$, observe that $\theta = E[f(\mathrm{U})]$, where $\mathrm{U} \sim \mathrm{U}(0,1)$. With this,

(i) Generate $\mathrm{U}_1, \mathrm{U}_2, \cdots, \mathrm{U}_m \sim \mathrm{U}(0,1)$ i.i.d.;

(ii) Estimate θ with

$$\hat{\theta}_m := \frac{f(\mathrm{U}_1) + f(\mathrm{U}_2) + \cdots f(\mathrm{U}_m)}{m}.$$

$\hat{\theta}_m$ is a good estimator of θ, since

(i)$\hat{\theta}_m$ is unbiased, i.e. $E[\hat{\theta}_m] = \theta$ and

(ii)$\hat{\theta}_m$ is consistent, i.e., $\hat{\theta}_m \to \theta$ with probability 1 as $m \to \infty$, following strong law of large numbers.

For more general case, $\theta = \int_a^b f(x)dx = \int_a^b h(x)w(x)dx$,
where $h(x) = f(x)(b-a), w(x) = \frac{1}{b-a}, w(x)$is the p.d.f of$\mathrm{U}(a,b)$.

$$\hat{\theta}_m = \frac{h(\mathrm{U}_1) + h(\mathrm{U}_2) + \cdots h(\mathrm{U}_m)}{m}, \mathrm{U}_i \sim \mathrm{U}(a,b).$$

To estimate $\int_0^{t-\frac{j-1}{n}} e^{ku}u^{\alpha}du$, one writes
$\theta = \int_0^{t-\frac{j-1}{n}} e^{ku}u^{\alpha}du = (t-\frac{j-1}{n})E[\exp(k\mathrm{U})\mathrm{U}^{\alpha}]$, with $\mathrm{U} \sim \mathrm{U}(0, t-\frac{j-1}{n})$.

Generate m i.i.d. U(0,1), convert them to $\mathrm{U}(0, t-\frac{j-1}{n})$ multiplying by the coefficient $t-\frac{j-1}{n}$, and then take the average of the sum of the integrand at the converted random values.

3.4 Symbolic Integral to Estimate $A(t-\frac{j-1}{n})$

To estimate $\int_0^{t-\frac{j-1}{n}} e^{ku}u^{\alpha}du$, use the function int() in MATLAB, as follows.
$\int_0^{t-\frac{j-1}{n}} e^{ku}u^{\alpha}du = \mathrm{int}(e^{ku}u^{\alpha}, u, 0, t-\frac{j-1}{n})$, where u is specified as symbolic variable by syms u.

3.5 Approximate Stock Price, with Volatility Driven by W_t^{α}

With the above preparation steps, we can simulate the stock price path. Suppose that $\ln S(t)$ is a martingale and Let $Y(t) = \ln S(t)$. We are interested in the paths of stock price and volatility the model

$$\begin{cases} dY(t) = \sigma(t)dW_t^1 \\ d(\ln\sigma(t)) = -k\ln\sigma(t)dt + \gamma dW_t^{\alpha}, \end{cases} \tag{4}$$

where

$$W_t^{\alpha} = \int_0^t \frac{(t-s)^{\alpha}}{\Gamma(1+\alpha)}dW_t^2,$$

and W_t^1, W_t^2 are two independent standard Brownian processes.

In analogue to volatility, $Y(t)$ is estimated by

$$Y(t) = \int_0^t \sigma(t)dW_t^1 \tag{5}$$

$$\approx \sum_{j=1}^{[nt]} \sigma\Big(\frac{j-1}{n}\Big)\Delta W_{\frac{j}{n}}^1 + \sigma\Big(\frac{[nt]}{n}\Big)\Big(W_t^1 - W_{\frac{[nt]}{n}}^1\Big), \tag{6}$$

in which $\Delta W^1_{j/n} = W^1_{j/n} - W^1_{(j-1)/n}$. For large n, the last term is neglected. Note that, in (6), $\sigma(\cdot)$ is estimated. Use the previous proxy $\widetilde{X}_n(t)$, then

$$\widetilde{\sigma}_n(t) = \exp\Big[\sum_{j=1}^{[nt]} A\Big(t - \frac{j-1}{n}\Big)\Delta W^2_{j/n}\Big],$$

$$\widetilde{Y}_n(t) = \sum_{j=1}^{[nt]} \widetilde{\sigma}_n\Big(\frac{j-1}{n}\Big)\Delta W^1_{j/n}.$$

3.6 Experiment to Simulate the Paths of Volatility and Stock Price

Simulate a standard Brownian motion W_t in Matlab:

(a) Set $t_0 = 0, W_{t_0} = 0$.

(b) For $j = 1, 2, 3, \cdots, [nt]$, generate $\Delta W_{t_j} \sim N(0, t_j - t_{j-1})$,

i.e. $\Delta W_{t_j} = \text{normrnd}(0, (t_j - t_{j-1})/n)$.

(c) Set $W_{t_j} = W_{t_{j-1}} + \Delta W_{t_j}$.

Remark 1: When we generate a standard Brownian motion, actually, we generate a correlated random vector $(W_{t_1}, W_{t_2}, \cdots, W_{t_{[nt]}})$.

Simulate bivariate standard Brownian motion $W(t) = (W^1_t, W^2_t)$:

(a) Set $t_0 = 0, W_{t_0} = (W^1_{t_0}, W^2_{t_0}) = (0, 0)$.

(b) For $j = 1, 2, 3, \cdots, [nt]$, generate $\Delta W_{t_j} = \text{normrnd}(0, t_j - t_{j-1}, [1, 2])$. ΔW_{t_j} is a 1 by 2 matrix, containing two independent random numbers from normal distribution $N(0, t_j - t_{j-1})$.

(c) Set $W_{t_j} = W_{t_{j-1}} + \Delta W_{t_j}$.

By above construction process, it is obvious, $(W^1_{t_1}, \cdots, W^1_{t_{[nt]}})'$ and $(W^2_{t_1}, \cdots, W^2_{t_{[nt]}})'$ are two independent random vectors, while each vector itself is correlated.

Given n and $[0, t]$, for evenly partitioned $[0, \frac{[nt]}{n}]$, here is the procedure to estimate stock price and volatility.

Step 1, generate bivariate standard Brownian motion $W_t = (W^1_t, W^2_t)$ as follows.

(a) Set $W_0 = (W^1_0, W^2_0) = (0, 0)$.

(b) For $j = 1, 2, 3, \cdots, [nt]$, generate $\Delta W_{j/n} = \text{normrnd}(0, j/n, [1, 2])$.

(c) Set $W_{j/n} = W_{(j-1)/n} + \Delta W_{j/n}$.

Step 2, Estimate each $A(t - \frac{j-1}{n})$

Step 3, Estimate $\sigma_n(t)$

Step 4, Estimate $\hat{Y}_n(t)$

Consider on time interval $[0, 20]$, with discretion step size $h = 0.02$, $n = 1000$, $(\alpha, k, \gamma) = (0.1, 1, 0.05)$. To implement above embedded algorithm on a computer with core 2 Duo CPU T64000 @ 2.GHz, it took 144 hours to produce 98 simulated data of stock price. In practice, it is inefficient and impractical to conduct the simulation. Another shortcoming of the simulation is that, in each step calculating $A(t)$, it introduces approximating errors. As a result, the simulation is not only tremendously computing intensive, but less precise due to accumulated errors.

By symbolic integral, we obtain the paths of logvolatility and log stock price as the Hurst parameter changes from 0.6 to 0.99. Here are parameter setups for experiments 1-5 to simulate the paths for logvolatility and log-stockprice by employing symbolic integral.

Simulation 1: $k = 1, \gamma = 0.01, H = 0.6$.

Simulation 1: $k = 1, \gamma = 0.01, H = 0.7$.

Simulation 3: $k = 1, \gamma = 0.01, H = 0.8$.

Simulation 4: $k = 1, \gamma = 0.01, H = 0.9$.

Simulation 5: $k = 1, \gamma = 0.01, H = 0.99$.

It is worth noticing that, for each parameter setup, 1000 paths of logvolatility are generated. Then, we take the sample mean of the paths and produce the graph. Similar averaging work is done for logstock price. Simulation results are displayed as graphs 1-5 in appendix. Based on the ranges shown on the graphs, as H increases, logvolatility becomes smoother, although graphs appear similar shape.

4 Euler Algorithm

In this section, a modified method for simulating the paths of stock price and fractional Brownian motion driven volatility is presented. Instead of using truncated fBm as an approximate process to fBm, fBm is simulated from spectral density. Also, rather than finding an integral expression for volatility, we use Euler method.

4.1 The Dzhaparidze and van Zanten Method to Simulate fBm

Based on spectral theory, Dzhaparidze and van Zanten proposed to use series representations to simulate fBm.

Definition 4.1: The differential equation

$$z^2\frac{d^2y}{dz^2} + z\frac{dy}{dz} + (z^2 - \nu^2)y = 0$$

for a real constant ν, is called Bessel's differential equation of order ν, and its solutions are called Bessel functions.

Remark:

1. The order ν must be real. The argument z can be complex. The solutions are real if z is positive.

2. Bessel's differential equation is of second-order differential equation, thus there are two linearly independent solutions, that is, two Bessel functions.

3. Bessel functions of the first genre is

$$J_\nu(z) = \left(\frac{z}{2}\right)^\nu \sum_{k=0}^{\infty} \frac{\left(\frac{-z^2}{4}\right)^k}{k!\Gamma(v+k+1)} \ ,$$

where $\Gamma(\cdot)$ is the gamma function.

Bessel functions of the second genre can be expressed by the first genre as

$$Y_\nu(z) = \frac{J_\nu(z)\cos(\nu\pi) - J_{-\nu}(z)}{\sin(\nu\pi)} \ ,$$

for noninteger v. For integer order n, $Y_n(z) = \lim_{v \to n} Y_v(z)$.

The procedures to simulate fBm by spectral method are as follows.

Case 1: For $t \in [0,1]$, the fractional Brownian motion $W_H(t)$ can be simulated according to following steps.

Given Hurst parameter H,

(a) Find the positive zeros of Bessel function $J_{-H}(z)$, $x_1 < x_2 < \cdots < \cdots$, and the positive zeros of Bessel function $J_{1-H}(z)$, $y_1 < y_2 < \cdots$.(Note, $J_{-H}(z) = 0$ has infinite solutions and so does for $J_{1-H}(z) = 0$.)

(b) Generate independent Gaussian random variables, $X_1, X_2, X_3, \cdots$ and $Y_1, Y_2, \cdots$, such that $E(X_i) = E(Y_i) = 0$, and

$$VarX_i = 2c_H^2 x_i^{-2H} J_{1-H}^{-2}(x_n), VarY_i = 2c_H^2 y_n^{-2H} J_{-H}^{-2}(y_i),$$

where $c_H^2 = \pi^{-1}\Gamma(1+2H)\sin\pi H$.

(c) The fractional Brownian motion is calculated as

$$W_H(t) = \sum_{i=1}^{\infty} \frac{\sin x_i t}{x_i} X_i + \sum_{i=1}^{\infty} \frac{1-\cos y_i t}{y_i} Y_i \ .$$

The two series above is absolutely and uniformly convergent for $t \in [0,1]$. In simulation, we don't really sum up to infinity items. Instead, we take a finite number N, then calculate

$$W_H(t) = \sum_{i=1}^{N} \frac{\sin x_i t}{x_i} X_i + \sum_{i=1}^{N} \frac{1-\cos y_i t}{y_i} Y_i \ .$$

Case 2: For $t \in [0,T]$, $T > 1$, to guarantee W_t^H is convergent, use the following procedure.

Given Hurst parameter H,

(a) Find the positive zeros of first genre Bessel function J_{1-H}, $\omega_1 < \omega_2 < \cdots$.

(b) $X, Y_1, Y_2, \cdots, Z_1, Z_2, \cdots$, are i.i.d. standard Gaussian distributed random variables.

(c) For $n \in \mathcal{N}$, calculate σ_n, which is defined by

$$\sigma_n^{-1} = \frac{1-H}{H} \frac{\Gamma^2(1-H)\Gamma(\frac{3}{2}-H)}{\Gamma(H+\frac{1}{2})\Gamma(3-2H)} \left(\frac{\omega_n}{2}\right)^{2H} J_{-H}^2(\omega_n) T^{2-2H}.$$

(d) Fractional Brownian motion with Hurst parameter H is represented as

$$W_t^H = \frac{t}{\sqrt{2-2H}}X + \sum_{n=1}^{\infty} \frac{sin(2\omega_n t/T)}{\omega_n/T}\sqrt{\sigma_n}Y_n + \sum_{n=1}^{\infty} \frac{cos(2\omega_n t/T)-1}{\omega_n/T}\sqrt{\sigma_n}Z_n,$$

which converges uniformly for $t \in [0, T]$. Choose N large, sum up finite N terms instead of infinite terms in above formula.

4.2 Euler Algorithm

Following is an improved algorithm.

Step 1: We simulate the fractional Brownian motion by Dzhaparidze and van Zanten method by Case 2, instead of using the explicit integral expression for $X(t)$ in Section 3.

Step 2. Simulate standard Brownian process W_t.

Step 3: Use Euler method to simulate $X(t)$ in formula (4):

$X(t_{n+1}) - X(t_n) = -kX(t_n)(t_{n+1} - t_n) + \gamma(W^H_{t_{n+1}} - W^H_{t_n})$

Step 4: $\sigma(t_n) = \exp(X(t_n))$

step 5: $Y(t_{n+1}) - Y(t_n) = \sigma(t_n)(W_{t_{n+1}} - W_{t_n})$

sept 6: $S(t_n) = \exp(Y(t_n))$

In simulation, set $t_{n+1} - t_n$ to be equal distance $\frac{j}{n}$.

Here are parameter setups for experiments 6-10 to simulate the paths for fractional Brownian motion, standard Brownian motion, logvolatility and logstockprice.

Simulation 6: $k = 1, \gamma = 0.01, H = 0.6$.

Simulation 7: $k = 1, \gamma = 0.01, H = 0.7$.

Simulation 8: $k = 1, \gamma = 0.01, H = 0.8$.

Simulation 9: $k = 1, \gamma = 0.01, H = 0.9$.

Simulation 10: $k = 1, \gamma = 0.01, H = 0.99$.

Refer to graphs 6-10 in appendix for the simulation results by Euler method. Simulations 6-10 show that, as H increases, the tendency of

logvolatility depends more on the past. The path of logstockprice is a martingale, as we assumed. Logvolatility and logstockprice don't have similar path, since the former is driven by fBm, and the latter by SBM, which is independent of the SBM of the fBm.

5 Parameter Estimation

This section explores methods to estimate volatility and Hurst parameter in fractional Brownian motion giving a series of observed stock price. For illustration, we analyze the case of Google.

5.1 Estimate $\sigma^2(t)$, Using Quadratic Variation Method

Given stock price, estimate the volatility $\sigma(t)$, at given time t.

Following Comte and Renault(1998), $\int_0^t \sigma^2(s)ds =< Y >_t$, that is to say,

$$P(\lim_{\text{stepsize}\to 0}\sum_{k=1}^{m}(Y_{t_k}-Y_{t_{k-1}})^2 =< Y >_t) = 1$$

where step=$\max_{1\le i\le m}\{|t_i - t_{i-1}|\}$. Thus, $\lim_{h\to 0}\frac{<Y>_t-<Y>_{t-h}}{h} = \sigma^2(t)$ a.s.

Let $[0,T]$ be the observation time interval, $t_k = \frac{kT}{N}$, and $Y_{t_k}, k = 0,\cdots,N$ is the observed centered natural log stock prices. Set $N = np$, i.e, the observations are separated to n blocks with p observation of each block. There is trade-off between n and p and one suggested method in statistical reference is to set both to be of order $\sqrt{N}$.

$\widehat{< Y >_t}^{(N)} = \sum_{k=0}^{[(tN)/T]}(Y_{t_k}-Y_{t_{k-1}})^2$, and thus, in practice, estimate $\sigma^2(t)$ with

$$\hat{\sigma}^2_{n,p}(t) = \frac{n}{T}\sum_{k=[\frac{tN}{T}]-p+1}^{[\frac{tN}{T}]}(Y_{t_k}-Y_{t_{k-1}})^2$$

Take Google's stock prices between Oct. 16 2005 and Oct.15th 2012 as observation data sample of size 1261. In this case, the observation is on daily base, set $T = 1, N = 1261$. $\sigma^2_{n,p}(t)$ measures the the estimated centered

log-prices variation on $[0, T]$. $n = \sqrt{1261} \approx 35$, $\frac{kT}{N} = k, k = 0, \cdots, N$,

Note that, we can still calculate $\sigma^2_{n,p}(t)$ for each day, despite the breaking of blocks, this is an exciting phenomenon.

5.2 Estimating the Hurst Parameter in fBm by Periodogram Method

This section use the spectral density of fBm, which is a Fourier transform of the density of fBm, to estimate the Hurst parameter.

5.2.1 Spectral Density of Fractional Brownian Motion Process

Fourier transform converts a time-domain function into a frequency-domain function without losing any information. And the reserve process is realized by inverse Fourier transform. For stationary processes, it is believed that, autocovariance function contains all the frequency information. The spectral density for frequencies $-\pi \leq x \leq \pi$ is

$$f(x) = \sum_{j=-\infty}^{\infty} d(j)\exp(ijx),$$

where $d(\cdot)$ denotes the autocovariance function. By reverse process, the autocovariance function is

$$d(j) = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x)\exp(-ijx)dx.$$

By Sinai(1976) and Fox and Taqqu (1986), the spectral density of fractional Brownian motion $f(x, H)$ is given by

$$f(x, H) = 2\sin(\pi H)\Gamma(2H + 1)(1 - \cos x)[|x|^{-2H-1} + B(x, H)];$$

where $\Gamma(\cdot)$ is Gamma function and

$$B(x, H) = \sum_{j=1}^{\infty} \{(2\pi j + x)^{-2H-1} + (2\pi j - x)^{-2H-1}\}$$

for $-\pi \leq x \leq \pi$.

Since $1 - cosx = \frac{x^2}{2} + O(x^4)$ as $|x| \to 0$, then

$$f(x, H) \sim |x|^{1-2H} \text{ as } |x| \to 0.$$

5.2.2 Periodogram

To approximate the spectral density of fBm, we use the truncated sum in $B(x, H)$.

Definition 5.2.2: The periodogram of a spectral density is defined to be

$$I(x) = \sum_{j=-N}^{N} \hat{d}(j)\exp(ijx),$$

with the autocovariance

$$\hat{d}(j) = \frac{1}{N}\sum_{k=0}^{N-|j|}(X_k - \bar{X})(X_{k+|j|} - \bar{X}), \bar{X} = \frac{\sum_{j=1}^{N} X_j}{N}.$$

Priestley (1981), Fox and Taqqu (1986) have showed that, the above definition of periodgram is equivalent to

$$I_N(x) = \frac{1}{2\pi N}\Big|\sum_{j=1}^{N} e^{ijx}(X_j - \overline{X})\Big|^2, \overline{X} = \frac{\sum_{j=1}^{N} X_j}{N}.$$

Use $I_N(x)$ approximate the $f(x, H)$. With $f(x, H) \sim x^{1-2H}$ as shown in section 5.2.1, one obtains $\log(I_N(x)) \sim (1 - 2H)\log(x)$.

In data analysis, x is the estimated σ, $I_N(x)$ is calculated above. $1 - 2H$ is the estimated by fitting a linear regression to the data. It's worth noting that, the estimator of Hurst parameter obtained from periodogram is not of high-quality. Modified periodogram methods in literature show no much better in precision.

For Google's stock price dataset, Hurst parameter H is estimated to be 0.5739 with log-Periodogram regression , which is between 0.5 and 1, with weak long-range dependence.

6 Option Pricing

With stock price and volatility model, it is possible to track the European call price? This topic is open for further exploration. Following are some adventurous attempts.

Let $(\Omega, \mathcal{F}, \mathbb{P})$ Be the fundamental probability space.$(\mathcal{F}_t)_{t\in[0,T]}$ denotes the $\mathbb{P}$-augmentation of the filtration. By Harrison and Kreps(1981)there exists a probability distribution $\mathbf{Q}$ on $(\Omega, \mathcal{F})$, equivalent to $\mathbb{P}$, under which the discounted price processes are martingales.

Let C_t be the call premium, i.e. the price at an European call option on the financial asset of price S_t at $t \leq T$, with strike K and maturing at time T.

Assume the instantaneous interest rate is $r(t)$, a deterministic function of at time t, then the price at time t of a zero coupon bond of maturity T is $B(t,T) = exp(-\int_t^T r(u)du)$

With the defined $\mathbf{Q}$, on $(\Omega, \mathbb{P}, \mathbf{Q})$, the call option price is

$$C_t = B(t,T)\mathbb{E}^{\mathbf{Q}}[Max(0, S_T - K)|\mathcal{F}_t]$$

. By Comte and Renault(1998), conditioned on the volatility path,

$$C_t = S(t)\Big\{\mathbb{E}_t^{\mathbf{Q}}\Big[\mathbf{\Phi}(\frac{m_t}{U_{t,T}} + \frac{U_{t,T}}{2})|\mathcal{F}_t\Big] - e^{-m_t}\mathbb{E}_t^{\mathbf{Q}}\Big[\mathbf{\Phi}(\frac{m_t}{U_{t,T}} - \frac{U_{t,T}}{2})|\mathcal{F}_t\Big]\Big\}$$

where $m_t = \ln(\frac{S(t)}{KB(t,T)})$, $U_{t,T} = \sqrt{\int_t^T \sigma(u)^2 du}$, and$\Phi(u) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{u} e^{-t^2/2}dt$.

In above conditional expectation, only $U_{t,T}$ contains the future. It would be very interesting to explore how to calculate the conditional expectation.

Let $(Z_t : t \in (-\infty, \infty))$ denote the normalized fractional Brownian motion with parameter $\alpha \in (0, \frac{1}{2})$. So far, the theory available is to predict $Z_h(h > 0)$ on the values obtained by Z_t in an interval $(-T, 0)$, in another word, to predict the conditional expectation of difference $Z_{t+h} - Z_t$ on the basis of the differences $Z_s - Z_t, s \in (t-T, t)$. By G. Gripenberg and I.Normmos (1996), the predictor $\hat{Z}_{h,T} = \mathbf{E}[Z_h|Z_s, s \in (-T, 0)] = \int_{-T}^{0} g_T(h,t)dZ_t$, where,

for $T < \infty, t \in (0, T)$,

$$g_T(h, -t) = \frac{sin(\pi\alpha)}{\pi} t^{-\alpha}(T-t)^{-\alpha} \int_0^h \frac{\sigma^\alpha(\sigma+T)^\alpha}{\sigma+t} d\sigma$$

for $T = \infty, t > 0$

$$g_T(h, -t) = \frac{sin(\pi\alpha)}{\pi} t^{-\alpha}(T-t)^{-\alpha} \int_0^h \frac{\sigma^\alpha}{\sigma+t} d\sigma$$

The function $g_T(h, .)$ is a solution of the integral equation

$$2\alpha \int_0^T g_T(h, -t)|t-s|^{2\alpha-1} dt = (h+s)^{h\alpha} - s^{2\alpha}, s \in (0, T),$$

with the scaling property $g_T(h, t) = g_{T/h}(1, t/h)$.

7 Conclusion and Further Topic

The main contribution of this paper is to propose a modified method, Euler method, to simulate the stock price model with stochastic volatility driven by fractional Brownian motion. Euler method, together with spectral series representation of fBm, speeds up the simulation quickly. Comte and Renault (1998)'s method is very computational intensive when using Monte Carlo integral for the integration in truncated fBm, and introduces errors. Symbolic integral for truncated fBm is practical, but still cost more time than Euler method.

The fractional Brownian volatility is applied to model real stock prices. For stock prices with feature of persistence, by periodogram method, we identify the Hurst parameter based on data. A topic of interest for further exploration is to find the trajectories of European call option prices.

References

[1] Backus and Zin . Long-memory inflation uncertainty: Evidence from the term structure of interest rates, *Journal of Money, Credit and Banking*, Vol 25, 681-700, 1995.

[2] F. Comte and E. Renault. Long Memory Continuous Time Models. *Journal of Econometrics*, Vol. 73, pp. 101-149, 1996.

[3] F. Comte and E. Renault. Long Memory in Continuous-Time Stochastic Volatility Models. *Mathematical Finance*, Vol. 8, No. 4, pp. 291-323, 1998.

[4] G. Fripenberg and I. Norros. On the Prediction of Fractional Brownian Motion. *Journal of Applied Probability*, Vol. 33, No.2, p. 400-410, 1996.

[5] Kacha Dzhaparidze and Harry van Zanten (2004). A series expansion of fractional Brownian motion. *Probability Theory and Related Fields*, Vol. 130, No.1, p.39-55.

[6] M.B. Priestley. *Spectral analysis and time series, Vol. 1.* Academic Press, 1981.

[7] P. Cheridito, H. Kawaguchi and M. maejima. Fractional Ornstein-Uhlenbeck Processes. *Electronic Journal of Probability*, Vol.8, page 1-14, 2003.

[8] Steven E. Shreve. *Stochastic Calculus for Finance II, Continuous-Time Models.* Springer, 2004.

Appendices

Figure 1

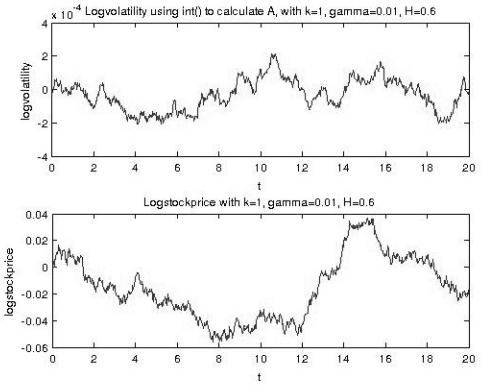


Figure 2

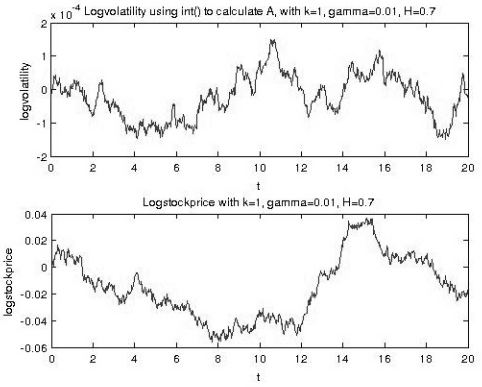


Figure 1 contains sample paths for logvolatility and logstockprice, on time interval [0,20]. The function int() in matlab is used to calculate the integral part in logvolatility, with parameter k=1, γ = 0.01, H=0.6. In simulation, 1000 paths of logvolatility and logstockprice are generated separately, then we take the mean of the 1000 paths respectively to form the graphs.

Similarly, Figure 2 to Figure 5 contain the paths for logvolatility and logstockprice with H=0.7, 0.8, 0.9, 0.99, respectively. Due to averaging process, the shape of the graphs are the same. However, as H increases from 0.6 to 0.99, the graphs of logvolatility become smoother and smoother, as can be seen that the range of logvolatility shrinks from $[-2\times10^{-4}, 2\times10^{-4}]$ to $[-5.5\times10^{-5}, 5.5\times10^{-5}]$.

Figure 3

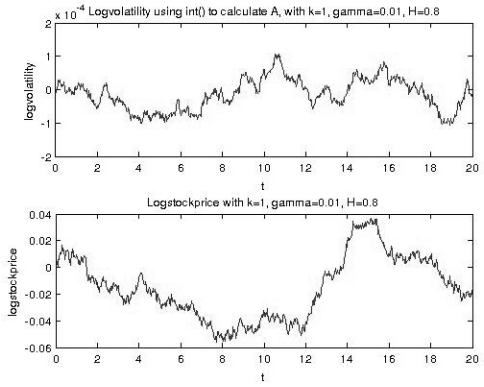


Figure 4

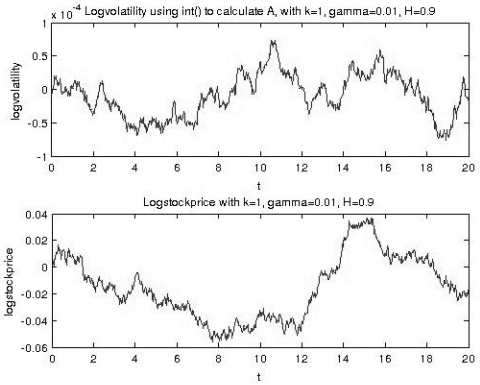


Figure 5

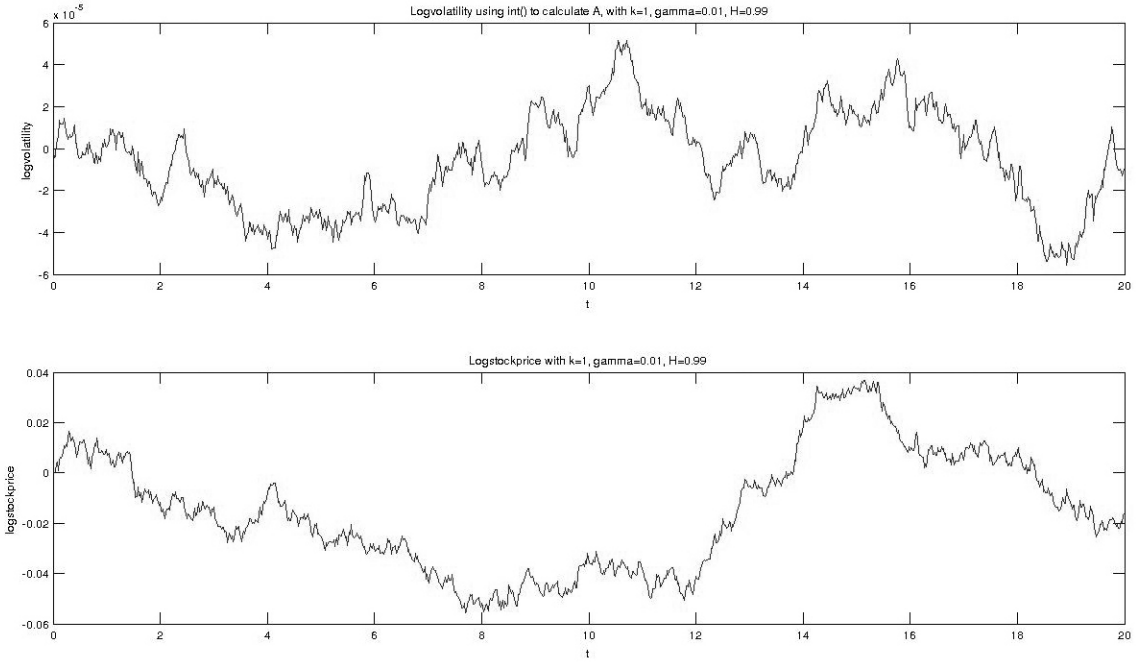
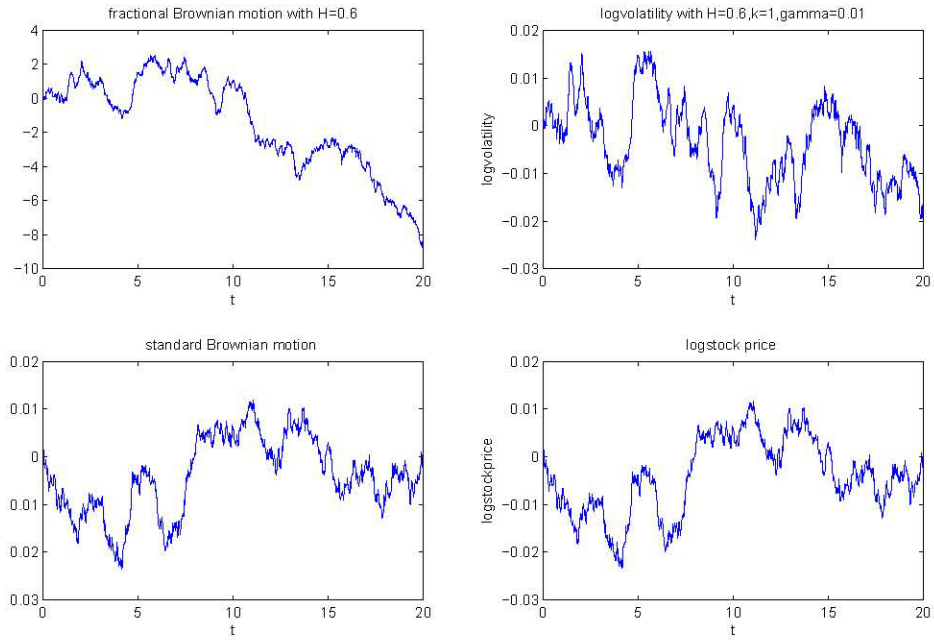


Figure 6: Paths with parameters $k = 1, \gamma = 0.01, H = 0.6$, by Euler method



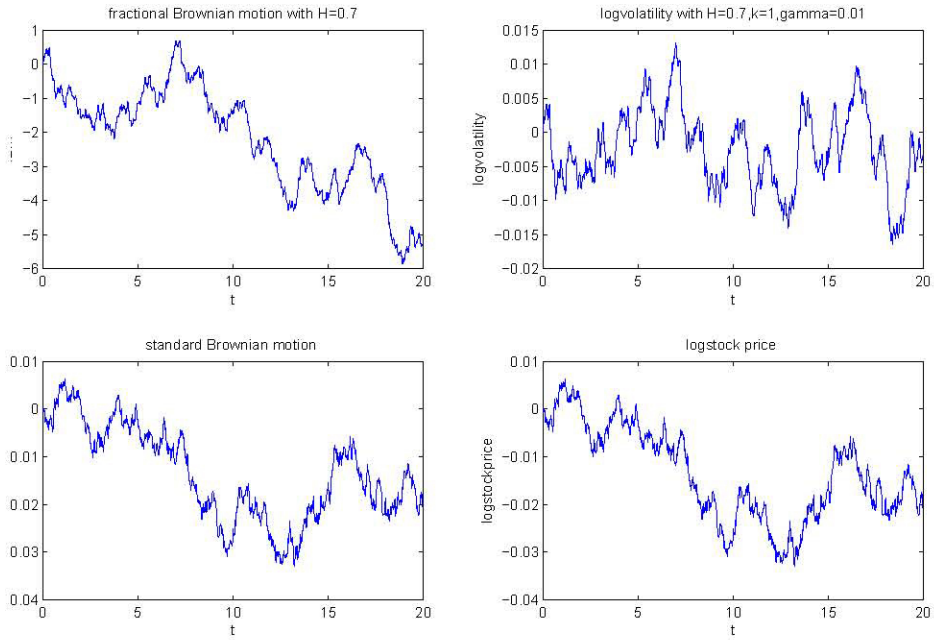
Above are sample paths for SBM, FBM, logvolatility and logstock-price,on time interval [0,20], using Euler method, with parameter k=1, $\gamma = 0.01$, H=0.6. In simulation, FBM is simulated by spectral method, in which the number of finite summation N=1000.

Note that, FBM with Hurst parameter H is represented as

$$W_t^H = \frac{t}{\sqrt{2-2H}}X + \sum_{n=1}^{\infty} \frac{sin(2\omega_n t/T)}{\omega_n/T}\sqrt{\sigma_n}Y_n + \sum_{n=1}^{\infty} \frac{cos(2\omega_n t/T)-1}{\omega_n/T}\sqrt{\sigma_n}Z_n,$$

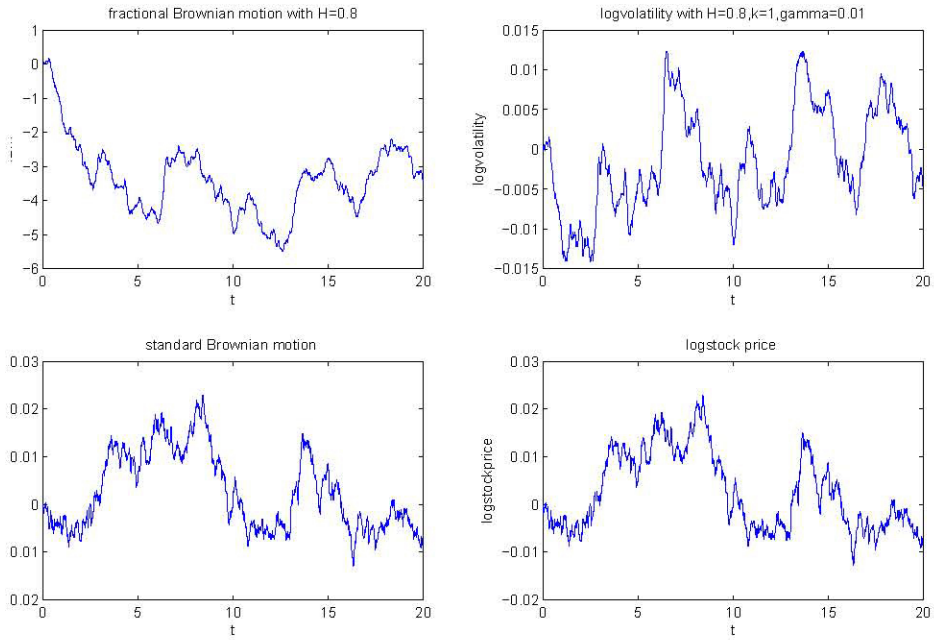
which converges uniformly for $t \in [0, T]$. Choose N large, sum up finite N terms instead of infinite terms in above formula.

Figure 7: Paths with parameters $k = 1, \gamma = 0.01, H = 0.7$, by Euler method



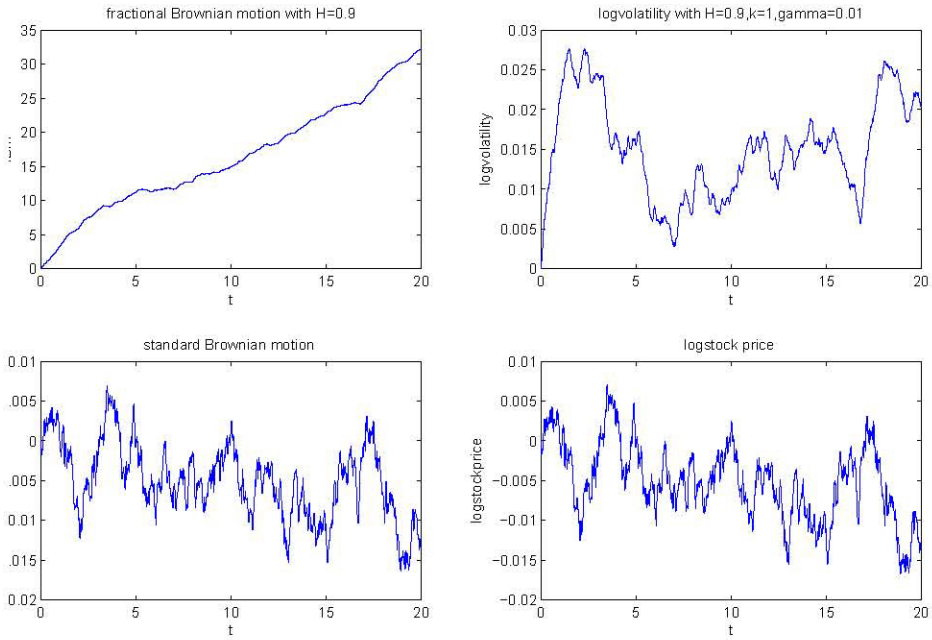
Above are sample paths for SBM, FBM, logvolatility and logstockprice, on time interval [0,20], using Euler method, with parameter k=1, $\gamma = 0.01$, H=0.7. In simulation, FBM is simulated by spectral method, in which the number of finite summation N=1000.

Figure 8: Paths with parameters $k = 1, \gamma = 0.01, H = 0.8$, by Euler method



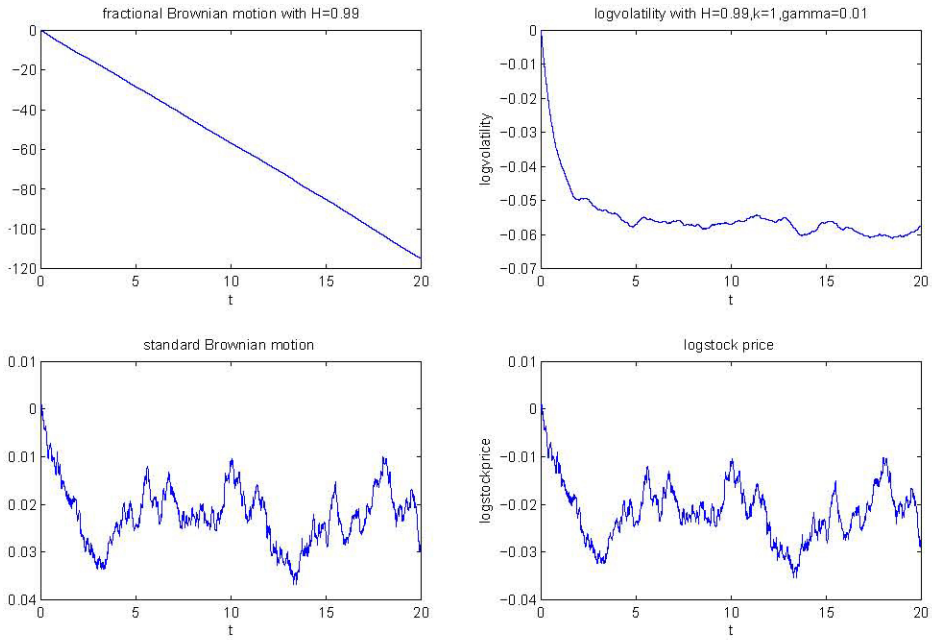
Above are sample paths for SBM, FBM, logvolatility and logstockprice, on time interval [0,20], using Euler method, with parameter k=1, $\gamma = 0.01$, H=0.8. In simulation, FBM is simulated by spectral method, in which the number of finite summation N=1000.

Figure 9: Paths with parameters $k = 1, \gamma = 0.01, H = 0.9$, by Euler method



Above are sample paths for SBM, FBM, logvolatility and logstockprice, on time interval [0,20], using Euler method, with parameter k=1, $\gamma = 0.01$, H=0.9. In simulation, FBM is simulated by spectral method, in which the number of finite summation N=1000.

Figure10: Paths with parameters $k = 1, \gamma = 0.01, H = 0.99$, by Euler method



Above are sample paths for SBM, FBM, logvolatility and logstockprice, on time interval [0,20], using Euler method, with parameter k=1, $\gamma = 0.01$, H=0.99. In simulation, FBM is simulated by spectral method, in which the number of finite summation N=1000.

```
% Consider stochastic differential logvolatility driven by truncated fBm
% This is the procedure to simulate logvolatility and logstockprice by
% using symbolic integration  int() in matlab for H=0.6  .

H=0.6; N=1000;
   % N is the number of paths for logvolatility and logstockprice.
   % We generate N paths separately then take the mean.

 log_sigma=zeros(1,1001);  % The initial value is a vector of zeros. 1001 is
% the same as the length of the vector t=[0:0.02:T] when T=20
     % It will be updated by the formula before the last "end"
 log_sto_price=zeros(1,1001);
 for p=1:N   % Utmost loop, in order to calculate the mean of different paths
% for logvolatility and logstockprice

  k=1;
  r=0.01;   % note, r here represent \gamma in the volatility model
%dX(t)=kX(t)dt+\gammadW^\alpha_t
          % Avoid using gamma as the name of gamma function as variable.
  alpha=H-1/2;

 n=50;
 T=20;
%[0,T]=[0,20] is time interval we would like to
%observefunction[]=direct_int_method(H,N)
 t=[0:0.02:T]; % discretize the interval
DBm=normrnd(0,1/n,[n*T,2]); % generate n*T by 2 Gaussian random variables,
%with mean 0, variance j/n-(j-1)/n=1/n

X=[]; % X here respresents the logsigma matrix.
X(1)=0; % since t(1)=0
A=[];
for i=2:n*T+1

    for j=floor(n*t(i-1))+1:floor(n*t(i))
        syms u

A(j)=r/gamma(1+alpha)*((t(i)-(j-1)/n)^alpha-k*exp(-k*(t(i)-(j-1)/n))*int(
exp(k*u)*u^alpha,0,t(i)-(j-1)/n));
         format short
         A(j)=round(A(j)*10000)*0.0001; % round A to four decimals
    end

    X(i)=A*DBm(1:floor(n*t(i)),2);
```

```
end

  % Following is to simulate the path of logstock price
  sigma=exp(X);


  Y_est=[];
  t=[0:0.02:T];
  Y_est(1)=0; % according to the integral form of logstockprice
  Y_est(2)=0;
     for i=3:T*n+1
         Y_est(i)=sigma(1:floor(n*t(i)))*DBm(1:floor(n*t(i)),1);
     end

  log_sigma=1/p*((p-1)*log_sigma+X); % take the mean of the paths of log_sigma.
%This is designed to update log_sigma
  log_sto_price=1/p*((p-1)*log_sto_price+Y_est); % take the mean of
%log_sto_price. This is designed to update log_sto_price
  p  % print out p, the number of sample paths.

 end   % end of utmost loop.


   subplot(2,1,1); plot([0:0.02:20],log_sigma)
    xlabel('t')
   ylabel('logvolatility')
   title(' Logvolatility using int() to calculate A, with k=1, gamma=0.01,
H=0.6')

   subplot(2,1,2); plot([0:0.02:20],log_sto_price)
   xlabel('t')
  ylabel('logstockprice')
  title('Logstockprice with k=1, gamma=0.01, H=0.6')
```

```
% Consider stochastic differential logvolatility driven by truncated fBm
% This is the procedure to simulate logvolatility and logstockprice by
% using symbolic integration  int() in matlab for H=0.7.

H=0.7; N=1000;
   % N is the number of paths for logvolatility and logstockprice.
   % We generate N paths separately then take the mean.

 log_sigma=zeros(1,1001);  % The initial value is a vector of zeros. 1001 is
% the same as the length of the vector t=[0:0.02:T] when T=20
  % It will be updated by the formula before the last "end"
 log_sto_price=zeros(1,1001);
 for p=1:N   % Utmost loop, in order to calculate the mean of different paths
% for logvolatility and logstockprice

  k=1;
  r=0.01;   % note, r here represent \gamma in the volatility model
%dX(t)=kX(t)dt+\gammadW^\alpha_t
          % Avoid using gamma as the name of gamma function as variable.
  alpha=H-1/2;

 n=50;
 T=20; %[0,T]=[0,20] is time interval we would like to
%observefunction[]=direct_int_method(H,N)
 t=[0:0.02:T]; % discretize the interval
DBm=normrnd(0,1/n,[n*T,2]); % generate n*T by 2 Gaussian random variables,
%with mean 0, variance j/n-(j-1)/n=1/n

X=[]; % X here respresents the logsigma matrix.
X(1)=0; % since t(1)=0
A=[];
for i=2:n*T+1

    for j=floor(n*t(i-1))+1:floor(n*t(i))
        syms u

A(j)=r/gamma(1+alpha)*((t(i)-(j-1)/n)^alpha-k*exp(-k*(t(i)-(j-1)/n))*int(
exp(k*u)*u^alpha,0,t(i)-(j-1)/n));
        format short
        A(j)=round(A(j)*10000)*0.0001; % round A to four decimals
    end

    X(i)=A*DBm(1:floor(n*t(i)),2);
```

```
end

 % Following is to simulate the path of logstock price
  sigma=exp(X);


  Y_est=[];
  t=[0:0.02:T];
  Y_est(1)=0; % according to the integral form of logstockprice
  Y_est(2)=0;
     for i=3:T*n+1
         Y_est(i)=sigma(1:floor(n*t(i)))*DBm(1:floor(n*t(i)),1);
     end

  log_sigma=1/p*((p-1)*log_sigma+X); % take the mean of the paths of log_sigma.
%This is designed to update log_sigma
  log_sto_price=1/p*((p-1)*log_sto_price+Y_est); % take the mean of
%log_sto_price. This is designed to update log_sto_price
  p  % print out p, the number of sample paths.

 end   % end of utmost loop.


   subplot(2,1,1); plot([0:0.02:20],log_sigma)
    xlabel('t')
   ylabel('logvolatility')
   title(' Logvolatility using int() to calculate A, with k=1, gamma=0.01,
H=0.7')

   subplot(2,1,2); plot([0:0.02:20],log_sto_price)
   xlabel('t')
  ylabel('logstockprice')
  title('Logstockprice with k=1, gamma=0.01, H=0.7')
```

```
% Consider stochastic differential logvolatility driven by truncated fBm
% This is the procedure to simulate logvolatility and logstockprice by
% using symbolic integration  int() in matlab for H=0.8.

H=0.8; N=1000;
   % N is the number of paths for logvolatility and logstockprice.
   % We generate N paths separately then take the mean.

 log_sigma=zeros(1,1001);  % The initial value is a vector of zeros. 1001 is
%the same as the length of the vector t=[0:0.02:T] when T=20
                        % It will be updated by the formula before the
                        % last "end"
 log_sto_price=zeros(1,1001);
 for p=1:N   % Utmost loop, in order to calculate the mean of different paths
%for logvolatility and logstockprice

  k=1;
  r=0.01;    % note, r here represent \gamma in the volatility model
%dX(t)=kX(t)dt+\gammadW^\alpha_t
  alpha=H-1/2;

 n=50;
 T=20;   %[0,T]=[0,20]   is   time   interval   we   would   like   to
%observefunction[]=direct_int_method(H,N)
 t=[0:0.02:T]; % discretize the interval
DBm=normrnd(0,1/n,[n*T,2]); % generate n*T by 2 Gaussian random variables,
%with mean 0, variance j/n-(j-1)/n=1/n

X=[]; % X here respresents the logsigma matrix.
X(1)=0; % since t(1)=0
A=[];
for i=2:n*T+1

    for j=floor(n*t(i-1))+1:floor(n*t(i))
        syms u

A(j)=r/gamma(1+alpha)*((t(i)-(j-1)/n)^alpha-k*exp(-k*(t(i)-(j-1)/n))*int(
exp(k*u)*u^alpha,0,t(i)-(j-1)/n));
        format short
        A(j)=round(A(j)*10000)*0.0001; % round A to four decimals
    end

    X(i)=A*DBm(1:floor(n*t(i)),2);
```

```
end

 % Following is to simulate the path of logstock price
  sigma=exp(X);


  Y_est=[];
  t=[0:0.02:T];
  Y_est(1)=0; % according to the integral form of logstockprice
  Y_est(2)=0;
     for i=3:T*n+1
         Y_est(i)=sigma(1:floor(n*t(i)))*DBm(1:floor(n*t(i)),1);
     end

  log_sigma=1/p*((p-1)*log_sigma+X); % take the mean of the paths of log_sigma.
% This is designed to update log_sigma
  log_sto_price=1/p*((p-1)*log_sto_price+Y_est);  %  take  the  mean  of
%log_sto_price. This is designed to update log_sto_price
  p  % print out p, the number of sample paths.

 end   % end of utmost loop.


   subplot(2,1,1); plot([0:0.02:20],log_sigma)
    xlabel('t')
   ylabel('logvolatility')
   title(' Logvolatility using int() to calculate A, with k=1, gamma=0.01,
H=0.8')

   subplot(2,1,2); plot([0:0.02:20],log_sto_price)
   xlabel('t')
  ylabel('logstockprice')
  title('Logstockprice with k=1, gamma=0.01, H=0.8')
```

```
% Consider stochastic differential logvolatility driven by truncated fBm
% This is the procedure to simulate logvolatility and logstockprice by
% using symbolic integration  int() in matlab for H=0.9.

H=0.9; N=1000;
   % N is the number of paths for logvolatility and logstockprice.
   % We generate N paths separately then take the mean.

 log_sigma=zeros(1,1001);  % The initial value is a vector of zeros. 1001 is
%the same as the length of the vector t=[0:0.02:T] when T=20
                        % It will be updated by the formula before the
                        % last "end"
 log_sto_price=zeros(1,1001);
 for p=1:N   % Utmost loop, in order to calculate the mean of different paths
%for logvolatility and logstockprice

  k=1;
  r=0.01;    % note, r here represent \gamma in the volatility model
%dX(t)=kX(t)dt+\gammadW^\alpha_t
  alpha=H-1/2;

 n=50;
 T=20;   %[0,T]=[0,20]   is   time   interval   we   would   like   to
%observefunction[]=direct_int_method(H,N)
 t=[0:0.02:T]; % discretize the interval
DBm=normrnd(0,1/n,[n*T,2]); % generate n*T by 2 Gaussian random variables,
%with mean 0, variance j/n-(j-1)/n=1/n

X=[]; % X here respresents the logsigma matrix.
X(1)=0; % since t(1)=0
A=[];
for i=2:n*T+1

    for j=floor(n*t(i-1))+1:floor(n*t(i))
        syms u

A(j)=r/gamma(1+alpha)*((t(i)-(j-1)/n)^alpha-k*exp(-k*(t(i)-(j-1)/n))*int(
exp(k*u)*u^alpha,0,t(i)-(j-1)/n));
        format short
        A(j)=round(A(j)*10000)*0.0001; % round A to four decimals
    end

    X(i)=A*DBm(1:floor(n*t(i)),2);
```

```
end


   % Following is to simulate the path of logstock price
  sigma=exp(X);


  Y_est=[];
  t=[0:0.02:T];
  Y_est(1)=0; % according to the integral form of logstockprice
  Y_est(2)=0;
     for i=3:T*n+1
         Y_est(i)=sigma(1:floor(n*t(i)))*DBm(1:floor(n*t(i)),1);
     end

  log_sigma=1/p*((p-1)*log_sigma+X); % take the mean of the paths of log_sigma.
%This is designed to update log_sigma
  log_sto_price=1/p*((p-1)*log_sto_price+Y_est);  %  take  the  mean  of
%log_sto_price. This is designed to update log_sto_price
  p  % print out p, the number of sample paths.

 end   % end of utmost loop.


   subplot(2,1,1); plot([0:0.02:20],log_sigma)
    xlabel('t')
   ylabel('logvolatility')
   title(' Logvolatility using int() to calculate A, with k=1, gamma=0.01,
H=0.9')

   subplot(2,1,2); plot([0:0.02:20],log_sto_price)
   xlabel('t')
  ylabel('logstockprice')
  title('Logstockprice with k=1, gamma=0.01, H=0.9')
```

```
% Consider stochastic differential logvolatility driven by truncated fBm
% This is the procedure to simulate logvolatility and logstockprice by
% using symbolic integration  int() in matlab for H=0.99.

H=0.99; N=1000;
   % N is the number of paths for logvolatility and logstockprice.
   % We generate N paths separately then take the mean.

 log_sigma=zeros(1,1001);  % The initial value is a vector of zeros. 1001 is
%the same as the length of the vector t=[0:0.02:T] when T=20
                        % It will be updated by the formula before the
                        % last "end"
 log_sto_price=zeros(1,1001);
 for p=1:N   % Utmost loop, in order to calculate the mean of different paths
%for logvolatility and logstockprice

  k=1;
  r=0.01;    % note, r here represent \gamma in the volatility model
%dX(t)=kX(t)dt+\gammadW^\alpha_t
  alpha=H-1/2;

 n=50;
 T=20;   %[0,T]=[0,20]   is   time   interval   we   would   like   to
%observefunction[]=direct_int_method(H,N)
 t=[0:0.02:T]; % discretize the interval
DBm=normrnd(0,1/n,[n*T,2]); % generate n*T by 2 Gaussian random variables,
%with mean 0, variance j/n-(j-1)/n=1/n

X=[]; % X here respresents the logsigma matrix.
X(1)=0; % since t(1)=0
A=[];
for i=2:n*T+1

    for j=floor(n*t(i-1))+1:floor(n*t(i))
        syms u

A(j)=r/gamma(1+alpha)*((t(i)-(j-1)/n)^alpha-k*exp(-k*(t(i)-(j-1)/n))*int(
exp(k*u)*u^alpha,0,t(i)-(j-1)/n));
        format short
        A(j)=round(A(j)*10000)*0.0001; % round A to four decimals
    end

    X(i)=A*DBm(1:floor(n*t(i)),2);
```

```
end

  % Following is to simulate the path of logstock price
  sigma=exp(X);


  Y_est=[];
  t=[0:0.02:T];
  Y_est(1)=0; % according to the integral form of logstockprice
  Y_est(2)=0;
     for i=3:T*n+1
         Y_est(i)=sigma(1:floor(n*t(i)))*DBm(1:floor(n*t(i)),1);
     end

  log_sigma=1/p*((p-1)*log_sigma+X); % take the mean of the paths of log_sigma.
%This is designed to update log_sigma
  log_sto_price=1/p*((p-1)*log_sto_price+Y_est);  %  take  the  mean  of
%log_sto_price. This is designed to update log_sto_price
  p  % print out p, the number of sample paths.

 end   % end of utmost loop.


   subplot(2,1,1); plot([0:0.02:20],log_sigma)
    xlabel('t')
   ylabel('logvolatility')
   title(' Logvolatility using int() to calculate A, with k=1, gamma=0.01,
H=0.99')

   subplot(2,1,2); plot([0:0.02:20],log_sto_price)
   xlabel('t')
  ylabel('logstockprice')
  title('Logstockprice with k=1, gamma=0.01, H=0.99')
```

```
% This is Matlab code for the simulation of BM, fBm logvolatility
% and logstock price. Bessel function is used to generate fBm with H=0.6.
H=0.6;
k=1;
r=0.01;
n=1000;
s=0.02;
h=0.02;

% Generate fractional Brown motion on [0,20]by by spectral method
 T=20;
N_0=2000;%N_0 is the number of steps for approximating fBm

Wb=zerobess('J', 1-H, N_0);
Wb=Wb'; % change Wb to be 1 by N matrix
% Generate N positive zeros of first kind bessel function J_{1-H}
B=besselj(-H,Wb); % calculate the bessel function of first kind at each
                  %element of vector W.

X_0=normrnd(0,1);
Yg=normrnd(0,1,1,N_0); % generate 1 by N standard Gaussian random numbers
Zg=normrnd(0,1,1,N_0);

% calculate the inverse of standard variance
inv_sta_var=[];
sr_sta_var=[];
for i=1:N_0

inv_sta_var(i)=(1-H)/H*gamma(1-H)*gamma(1-H)*gamma(1.5-H)/(gamma(H+0.5)*g
amma(3-2*H))*(Wb(i)/2)^(2*H)*B(i)^2*T^(2-2*H);
end

sta_var=1./inv_sta_var;
sr_sta_var=sqrt(sta_var);

%  t=[0:0.02:20] <---> j=1:1001, calculate fBm(t), actually, we calcualate
%  discretized fBm
t=[0:0.02:T];
fBm=[];
fBm(1)=0; % Conclude from the series expression. Note, since when t=0,
% the numerators and denominators in the series are 0.
for j=2:1001
```

```
fBm(j)=t(j)/sqrt(2-2*H)*X_0+sum((sin(2*Wb*t(j)/T))./(Wb/T).*sr_sta_var.*Y
g)+sum((cos(2*Wb*t(j)/T)-1)./(Wb/T).*sr_sta_var.*Zg);
end
DfBm=[];
DfBm(1)=0;
for j=2:1000
    DfBm(j)=fBm(j)-fBm(j-1);
end

  subplot(2,2,1); plot([0:0.02:T],fBm);
     title('fractional Brownian motion with H=0.6')
     xlabel('t')
     ylabel('fBm')

% now use Euler algorithm to calculate logvolatility price
X=[];
X(1)=0;
for j=2:floor(n)
    X(j)=X(j-1)-k*X(j-1)*h+r*DfBm(j);
end

subplot(2,2,2);plot([0:0.02:T-0.02],X);
title('logvolatility with H=0.6,k=1,gamma=0.01')
xlabel('t')
ylabel('logvolatility')

% generate a Brownian motion, which is used to simulate the logstock price
 W=[];
W(1)=0;
for j=2:n
    W(j)=W(j-1)+normrnd(0,1/n);
end

subplot(2,2,3); plot([0:0.02:T-0.02],W)    % plot W against t
title('standard Brownian motion ')
xlabel('t')
ylabel('W(t)')

% simulate the logstock price model
DBm=[];
DBm(1)=W(1);
for j=2:n;
    DBm(j)=W(j)-W(j-1);
end
```

```
sigma_est=exp(X); % sigma estimated

Y=[];
Y(1)=0;
for j=2:n
Y(j)=Y(j-1)+sigma_est(j-1)*DBm(j);
end
subplot(2,2,4); plot([0:0.02:T-0.02],Y);
title('logstock price ')
xlabel('t')
ylabel('logstockprice')
```

```
% This is Matlab code for the simulation of BM, fBm logvolatility
% and logstock price. Bessel function is used to generate fBm with H=0.7.
H=0.7;
k=1;
r=0.01;
n=1000;
s=0.02;
h=0.02;

% Generate fractional Brown motion on [0,20]by by spectral method
T=20;
N_0=2000;%N_0 is the number of steps for approximating fBm

Wb=zerobess('J', 1-H, N_0);
Wb=Wb'; % change Wb to be 1 by N matrix
% Generate N positive zeros of first kind bessel function J_{1-H}
B=besselj(-H,Wb); % calculate the bessel function of first kind at each
                  %element of vector W.

X_0=normrnd(0,1);
Yg=normrnd(0,1,1,N_0); % generate 1 by N standard Gaussian random numbers
Zg=normrnd(0,1,1,N_0);

% calculate the inverse of standard variance
inv_sta_var=[];
sr_sta_var=[];
for i=1:N_0

inv_sta_var(i)=(1-H)/H*gamma(1-H)*gamma(1-H)*gamma(1.5-H)/(gamma(H+0.5)*g
amma(3-2*H))*(Wb(i)/2)^(2*H)*B(i)^2*T^(2-2*H);
end

sta_var=1./inv_sta_var;
sr_sta_var=sqrt(sta_var);

%  t=[0:0.02:20] <---> j=1:1001, calculate fBm(t), actually, we calcualate
%  discretized fBm
t=[0:0.02:T];
fBm=[];
fBm(1)=0; % Conclude from the series expression. Note, since when t=0,
% the numerators and denominators in the series are 0.
for j=2:1001

fBm(j)=t(j)/sqrt(2-2*H)*X_0+sum((sin(2*Wb*t(j)/T))./(Wb/T).*sr_sta_var.*Y
```

```
g)+sum((cos(2*Wb*t(j)/T)-1)./(Wb/T).*sr_sta_var.*Zg);
end
DfBm=[];
DfBm(1)=0;
for j=2:1000
    DfBm(j)=fBm(j)-fBm(j-1);
end

  subplot(2,2,1); plot([0:0.02:T],fBm);
     title('fractional Brownian motion with H=0.7')
     xlabel('t')
     ylabel('fBm')

% now use Euler algorithm to calculate logvolatility price
X=[];
X(1)=0;
for j=2:floor(n)
    X(j)=X(j-1)-k*X(j-1)*h+r*DfBm(j);
end

subplot(2,2,2);plot([0:0.02:T-0.02],X);
title('logvolatility with H=0.7,k=1,gamma=0.01')
xlabel('t')
ylabel('logvolatility')

% generate a Brownian motion, which is used to simulate the logstock price
W=[];
W(1)=0;
for j=2:n
    W(j)=W(j-1)+normrnd(0,1/n);
end

subplot(2,2,3); plot([0:0.02:T-0.02],W)    % plot W against t
title('standard Brownian motion ')
xlabel('t')
ylabel('W(t)')

% simulate the logstock price model
DBm=[];
DBm(1)=W(1);
for j=2:n;
    DBm(j)=W(j)-W(j-1);
end
```

```
sigma_est=exp(X); % sigma estimated

Y=[];
Y(1)=0;
for j=2:n
Y(j)=Y(j-1)+sigma_est(j-1)*DBm(j);
end
subplot(2,2,4); plot([0:0.02:T-0.02],Y);
title('logstock price ')
xlabel('t')
ylabel('logstockprice')
```

```
% This is Matlab code for the simulation of BM, fBm logvolatility
% and logstock price. Bessel function is used to generate fBm with H=0.8.
H=0.8;
k=1;
r=0.01;
n=1000;
s=0.02;
h=0.02;

% Generate fractional Brown motion on [0,20]by by spectral method
 T=20;
N_0=2000;%N_0 is the number of steps for approximating fBm

Wb=zerobess('J', 1-H, N_0);
Wb=Wb'; % change Wb to be 1 by N matrix
% Generate N positive zeros of first kind bessel function J_{1-H}
B=besselj(-H,Wb); % calculate the bessel function of first kind at each
                  %element of vector W.

X_0=normrnd(0,1);
Yg=normrnd(0,1,1,N_0); % generate 1 by N standard Gaussian random numbers
Zg=normrnd(0,1,1,N_0);

% calculate the inverse of standard variance
inv_sta_var=[];
sr_sta_var=[];
for i=1:N_0

inv_sta_var(i)=(1-H)/H*gamma(1-H)*gamma(1-H)*gamma(1.5-H)/(gamma(H+0.5)*g
amma(3-2*H))*(Wb(i)/2)^(2*H)*B(i)^2*T^(2-2*H);
end

sta_var=1./inv_sta_var;
sr_sta_var=sqrt(sta_var);

%  t=[0:0.02:20] <---> j=1:1001, calculate fBm(t), actually, we calcualate
%  discretized fBm
t=[0:0.02:T];
fBm=[];
fBm(1)=0; % Conclude from the series expression. Note, since when t=0,
% the numerators and denominators in the series are 0.
for j=2:1001
```

```
fBm(j)=t(j)/sqrt(2-2*H)*X_0+sum((sin(2*Wb*t(j)/T))./(Wb/T).*sr_sta_var.*Y
g)+sum((cos(2*Wb*t(j)/T)-1)./(Wb/T).*sr_sta_var.*Zg);
end
DfBm=[];
DfBm(1)=0;
for j=2:1000
    DfBm(j)=fBm(j)-fBm(j-1);
end

  subplot(2,2,1); plot([0:0.02:T],fBm);
     title('fractional Brownian motion with H=0.8')
     xlabel('t')
     ylabel('fBm')

% now use Euler algorithm to calculate logvolatility price
X=[];
X(1)=0;
for j=2:floor(n)
    X(j)=X(j-1)-k*X(j-1)*h+r*DfBm(j);
end

subplot(2,2,2);plot([0:0.02:T-0.02],X);
title('logvolatility with H=0.8,k=1,gamma=0.01')
xlabel('t')
ylabel('logvolatility')

% generate a Brownian motion, which is used to simulate the logstock price
W=[];
W(1)=0;
for j=2:n
    W(j)=W(j-1)+normrnd(0,1/n);
end

subplot(2,2,3); plot([0:0.02:T-0.02],W)    % plot W against t
title('standard Brownian motion ')
xlabel('t')
ylabel('W(t)')

% simulate the logstock price model
DBm=[];
DBm(1)=W(1);
for j=2:n;
    DBm(j)=W(j)-W(j-1);
end
```

```
sigma_est=exp(X); % sigma estimated

Y=[];
Y(1)=0;
for j=2:n
Y(j)=Y(j-1)+sigma_est(j-1)*DBm(j);
end
subplot(2,2,4); plot([0:0.02:T-0.02],Y);
title('logstock price ')
xlabel('t')
ylabel('logstockprice')
```

```
% This is Matlab code for the simulation of BM, fBm logvolatility
% and logstock price. Bessel function is used to generate fBm with H=0.9.
H=0.9;
k=1;
r=0.01;
n=1000;
s=0.02;
h=0.02;

% Generate fractional Brown motion on [0,20]by by spectral method
T=20;
N_0=2000;%N_0 is the number of steps for approximating fBm

Wb=zerobess('J', 1-H, N_0);
Wb=Wb'; % change Wb to be 1 by N matrix
% Generate N positive zeros of first kind bessel function J_{1-H}
B=besselj(-H,Wb); % calculate the bessel function of first kind at each
                  %element of vector W.

X_0=normrnd(0,1);
Yg=normrnd(0,1,1,N_0); % generate 1 by N standard Gaussian random numbers
Zg=normrnd(0,1,1,N_0);

% calculate the inverse of standard variance
inv_sta_var=[];
sr_sta_var=[];
for i=1:N_0

inv_sta_var(i)=(1-H)/H*gamma(1-H)*gamma(1-H)*gamma(1.5-H)/(gamma(H+0.5)*g
amma(3-2*H))*(Wb(i)/2)^(2*H)*B(i)^2*T^(2-2*H);
end

sta_var=1./inv_sta_var;
sr_sta_var=sqrt(sta_var);

%  t=[0:0.02:20] <---> j=1:1001, calculate fBm(t), actually, we calcualate
%  discretized fBm
t=[0:0.02:T];
fBm=[];
fBm(1)=0; % Conclude from the series expression. Note, since when t=0,
% the numerators and denominators in the series are 0.
for j=2:1001
```

```
fBm(j)=t(j)/sqrt(2-2*H)*X_0+sum((sin(2*Wb*t(j)/T))./(Wb/T).*sr_sta_var.*Y
g)+sum((cos(2*Wb*t(j)/T)-1)./(Wb/T).*sr_sta_var.*Zg);
end
DfBm=[];
DfBm(1)=0;
for j=2:1000
    DfBm(j)=fBm(j)-fBm(j-1);
end

  subplot(2,2,1); plot([0:0.02:T],fBm);
     title('fractional Brownian motion with H=0.9')
     xlabel('t')
     ylabel('fBm')

% now use Euler algorithm to calculate logvolatility price
X=[];
X(1)=0;
for j=2:floor(n)
    X(j)=X(j-1)-k*X(j-1)*h+r*DfBm(j);
end

subplot(2,2,2);plot([0:0.02:T-0.02],X);
title('logvolatility with H=0.9,k=1,gamma=0.01')
xlabel('t')
ylabel('logvolatility')

% generate a Brownian motion, which is used to simulate the logstock price

W=[];
W(1)=0;
for j=2:n
    W(j)=W(j-1)+normrnd(0,1/n);
end

subplot(2,2,3); plot([0:0.02:T-0.02],W)    % plot W against t
title('standard Brownian motion ')
xlabel('t')
ylabel('W(t)')

% simulate the logstock price model
DBm=[];
DBm(1)=W(1);
for j=2:n;
    DBm(j)=W(j)-W(j-1);
```

```
end

sigma_est=exp(X); % sigma estimated

Y=[];
Y(1)=0;
for j=2:n
Y(j)=Y(j-1)+sigma_est(j-1)*DBm(j);
end
subplot(2,2,4); plot([0:0.02:T-0.02],Y);
title('logstock price ')
xlabel('t')
ylabel('logstockprice')
```

```
% This is Matlab code for the simulation of BM, fBm logvolatility
% and logstock price. Bessel function is used to generate fBm with H=0.99.
H=0.99;
k=1;
r=0.01;
n=1000;
s=0.02;
h=0.02;

% Generate fractional Brown motion on [0,20]by by spectral method
T=20;
N_0=2000;%N_0 is the number of steps for approximating fBm

Wb=zerobess('J', 1-H, N_0);
Wb=Wb'; % change Wb to be 1 by N matrix
% Generate N positive zeros of first kind bessel function J_{1-H}
B=besselj(-H,Wb); % calculate the bessel function of first kind at each
                  %element of vector W.

X_0=normrnd(0,1);
Yg=normrnd(0,1,1,N_0); % generate 1 by N standard Gaussian random numbers
Zg=normrnd(0,1,1,N_0);

% calculate the inverse of standard variance
inv_sta_var=[];
sr_sta_var=[];
for i=1:N_0

inv_sta_var(i)=(1-H)/H*gamma(1-H)*gamma(1-H)*gamma(1.5-H)/(gamma(H+0.5)*g
amma(3-2*H))*(Wb(i)/2)^(2*H)*B(i)^2*T^(2-2*H);
end

sta_var=1./inv_sta_var;
sr_sta_var=sqrt(sta_var);

%  t=[0:0.02:20] <---> j=1:1001, calculate fBm(t), actually, we calcualate
%  discretized fBm
t=[0:0.02:T];
fBm=[];
fBm(1)=0; % Conclude from the series expression. Note, since when t=0,
% the numerators and denominators in the series are 0.
for j=2:1001
```

```
fBm(j)=t(j)/sqrt(2-2*H)*X_0+sum((sin(2*Wb*t(j)/T))./(Wb/T).*sr_sta_var.*Y
g)+sum((cos(2*Wb*t(j)/T)-1)./(Wb/T).*sr_sta_var.*Zg);
end
DfBm=[];
DfBm(1)=0;
for j=2:1000
    DfBm(j)=fBm(j)-fBm(j-1);
end

  subplot(2,2,1); plot([0:0.02:T],fBm);
     title('fractional Brownian motion with H=0.99')
     xlabel('t')
     ylabel('fBm')

% now use Euler algorithm to calculate logvolatility price
X=[];
X(1)=0;
for j=2:floor(n)
    X(j)=X(j-1)-k*X(j-1)*h+r*DfBm(j);
end

subplot(2,2,2);plot([0:0.02:T-0.02],X);
title('logvolatility with H=0.99,k=1,gamma=0.01')
xlabel('t')
ylabel('logvolatility')

% generate a Brownian motion, which is used to simulate the logstock price
W=[];
W(1)=0;
for j=2:n
    W(j)=W(j-1)+normrnd(0,1/n);
end

subplot(2,2,3); plot([0:0.02:T-0.02],W)    % plot W against t
title('standard Brownian motion ')
xlabel('t')
ylabel('W(t)')

% simulate the logstock price model
DBm=[];
DBm(1)=W(1);
for j=2:n;
    DBm(j)=W(j)-W(j-1);
end
```

```
sigma_est=exp(X); % sigma estimated

Y=[];
Y(1)=0;
for j=2:n
Y(j)=Y(j-1)+sigma_est(j-1)*DBm(j);
end
subplot(2,2,4); plot([0:0.02:T-0.02],Y);
title('logstock price ')
xlabel('t')
ylabel('logstockprice')
```