# Stochastic Volatility

Tyler J. Brough

November 26, 2017

## Stochastic Volatility

In this document we review the stochastic volatility model of the Heston (1993) model. Relative to Black-Scholes we now have a bivariate system of stochastic differential equations:

$$dS_t = rS_t dt + \sqrt{V_t} S_t dW_{1,t}$$
$$dV_t = \kappa(\theta - V_t)dt + \sigma\sqrt{V_t} dW_{2,t}$$

with $E[dW_{1,t}, dW_{2,t}] = \rho dt$.

In the original paper, Heston was concerned with deriving a semi-closed form solution for the option pricing problem. Our concern is how to use Monte Carlo simulation to simulate from this system of equations under the risk-neutral density.

The first problem we face is how to simulate from these equations. There are a few options:

1. For the Heston model, Glasserman and Kaya (2004) show how to do this for the exact transition law of the process. It turns out though that this involves integration of a characteristic function expressed in terms of Bessel functions. It is very time consuming, and only works in the special case of Heston. That is, many other related stochastic volatility models offer no way to do exact simulation.

We therefore, turn to discretization schemes:

2. Euler discretization.
3. Milstein discretization.

See chapter 5 of Glasserman (2004).

Let's take a look at how this works:

1. Take the case of $dS_t = \mu(S_t, t)dt + \sigma(S_t, t)dW_t$, where $W_t$ is a Brownian motion.
2. We simulate over the time interval $[0, T]$, which we assume to be discretized as $0 = t_1 < t_2 < t_3 < \ldots < t_m = T$. Where the time increments are equally spaced with width $dt$.
3. We can integrate $dS_t$ from $t$ to $t + dt$ and get

$$S_{t+dt} = S_t + \int_t^{t+dt} \mu(S_u, u)du + \int_t^{t+dt} \sigma(S_u, u)dW_u$$

This equation is the starting point for any discretization scheme. At time $t$, the value of $S_t$ is known, and we wish to obtain the next value $S_{t+dt}$.

## The Euler Scheme

The simplest way to discretize the above integral equation is called the Euler approximation. This is equivalent to approximating the integrals using the left-point rule. Hence, the first integral is approximated as the product of the integrand at time $t$, and the integration range $dt$

$$\int_t^{t+dt} \mu(S_u, u)du \approx \mu(S_t, t) \int_t^{t+dt} du = \mu(S_t, t)dt$$

**NB:** recall the the left-point rule states that:

$$L_n = \int_a^b f(x)dx \approx \sum_{i=1}^{n} f(x_{i-1})\Delta x_i$$

We can use the left-point rule since at time $t$ the value of $\mu(S_t, t)$ is known.

Similarly, the second integral is approximated as

$$\begin{aligned}\int_t^{t+dt} \sigma(S_u, u)dW_u &\approx \sigma(S_t, t) \int_t^{t+dt} dW_u \\ &= \sigma(S_t, t)(W_{t+dt} - W_t) \\ &= \sigma(S_t, t)\sqrt{dt}Z\end{aligned}$$

Since $W_{t+dt} - W_t$ and $\sqrt{dt}Z$ are indentical in distribution where $Z \sim N(0, 1)$.

Thus the Euler discretization for

$$dS_t = \mu(S_t, t)dt + \sigma(S_t, t)dW_t$$

is

$$S_{t+dt} = S_t + \mu(S_t, t)dt + \sigma(S_t, t)\sqrt{dt}Z$$

Now let's look at the Euler discretization of the Black-Scholes model:

1. We have $dS_t = rS_t dt + \sigma S_t dW_t$
2. Applying the above we get:

$$S_{t+dt} = S_t + rS_t dt + \sigma S_t \sqrt{dt}Z$$

An alternative is to work with $x_t = \ln(S_t)$. Then by Itô's Lemma we have

$$dx_t = (r - \frac{1}{2}\sigma^2)dt + \sigma dW_t$$

Euler discretization gives us

$$x_{t+dt} = x_t + (r - \frac{1}{2}\sigma^2)dt + \sigma\sqrt{dt}Z$$

So that

$$S_{t+dt} = S_t \exp\left[(r - \frac{1}{2}\sigma^2)dt + \sigma\sqrt{dt}Z\right]$$

where $dt = t_i - t_{i-1}$.

**Euler Scheme for Heston's Stochastic Volatility**

Again, for Heston dynamics are described by the bivariate process:

$$\begin{aligned} dS_t &= rS_t dt + \sqrt{V_t}S_t dW_{1,t} \\ dV_t &= \kappa(\theta - V_t)dt + \sigma\sqrt{V_t}dW_{2,t} \\ E(dW_{1,t}, dW_{2,t}) &= \rho dt \end{aligned}$$

The stochastic differential equation (SDE) in integral form is:

$$V_{t+dt} = V_t + \int_t^{t+dt} \kappa(\theta - V_t)du + \int_t^{t+dt} \sigma\sqrt{V_u}dW_{2,u}$$

Again we use the left-point rule

$$\begin{aligned}\int_t^{t+dt} \kappa(\theta - V_u)du &\approx \kappa(\theta - V_t)dt \\ \int_t^{t+dt} \sigma\sqrt{V_u}dW_{2,u} &\approx \sigma\sqrt{V_t}(W_{t+dt} - W_t) \\ &= \sigma\sqrt{V_t dt}Z_v\end{aligned}$$

where $Z_v \sim N(0,1)$.

This gives us

$$V_{t+dt} = V_t + \kappa(\theta - V_t)dt + \sigma\sqrt{V_t dt}Z_v$$

To avoid negative values, we can set $V_t$ to $V_t = \max 0, V_t$. This is called the "full truncation scheme." The "reflection scheme" is an alternative, which sets $V_t$ to its absolute value $|V_t|$.

The process for $S_t$ is:

$$S_{t+dt} = S_t + r\int_t^{t+dt} S_u du + \int_t^{t+dt} \sqrt{V_u}S_u dW_u$$

Euler discretization leads to:

$$\begin{aligned}\int_t^{t+dt} S_u du &\approx S_t dt \\ \int_t^{t+dt} dW_{1,t} &\approx \sqrt{V_t}S_t(W_{t+dt} - W_t) \\ &= \sqrt{V_t dt}S_t Z_s\end{aligned}$$

where $Z_s \sim N(0,1)$. We end up with:

$$S_{t+dt} = S_t + rS_t dt + \sqrt{V_t dt}S_t Z_s$$

with $Corr(Z_v, Z_s) = \rho$.

**The Process for** $\ln S_t$

Let $x_t = \ln S_t$. Then

$$dx_t = \left(r - \frac{1}{2}V_t\right) dt + \sqrt{V_t} dW_{1,t}$$

Or in integral form

$$x_{t+dt} = x_t + \int_0^t \left(r - \frac{1}{2}V_u\right) du + \int_0^t \sqrt{V_u} dW_{1,u}$$

Euler discretization gives us

$$\begin{aligned} x_{t+dt} &= x_t + \left(r - \frac{1}{2}V_t\right) dt + \sqrt{V_t}\left(W_{1,t+dt} - W_{1,t}\right) \\ &= x_t + \left(r - \frac{1}{2}V_t\right) dt + \sqrt{V_t dt} Z_s \end{aligned}$$

Hence, for $S_t$ we have

$$S_{t+dt} = S_t \exp\left[\left(r - \frac{1}{2}V_t\right) dt + \sqrt{V_t dt} Z_s\right]$$

Again, to avoid negative variances we need to replace $V_t$ with either $V_t^+$ or $|V_t|$.

To generate $Z_v$ and $Z_s$ with correlation $\rho$, we do the following:

1. First generate two independent standard normals $Z_1$ and $Z_2$.
2. Set $Z_v = Z_1$.
3. Set $Z_s = \rho Z_1 + \sqrt{(1-\rho^2)} Z_2$

Let's see how to do this in Python:

```
import numpy as np
rho = 0.85
z1 = np.random.normal(size=10000)
z2 = np.random.normal(size=10000)
zv = z1
zs = rho * z1 + np.sqrt(1 - (rho * rho)) * z2
np.corrcoef(zv,zs)
```

Now we can use Python to actually carry out the Monte Carlo simulation for a stochastic volatility model:

```
import numpy as np
```