Bond Arb Project Report #2: Improving on existing Spread Simulator

Quantitative Trading Club at UIC

February 21, 2025

Abstract

In this report, I outline the improvements I made to my simulated spread for the bond arbitrage strategy. The upgraded model now integrates a Heston-type stochastic volatility (using the Milstein scheme), jump components, a dynamic short-term mean from the Hull–White model, and a yield curve adjustment via the Nelson–Siegel–Svensson framework. In addition, a Markov-switching regime indicator adapts the model to changing market conditions. This report details the methodology, provides breakdowns of the key formulas with simple examples, and includes code excerpts.

1 Introduction

My original simulation used a basic Ornstein–Uhlenbeck (OU) process with jumps and fixed parameters. Although it was a good starting point, it did not capture the nuances of a high-volatility bond arbitrage market. To improve this, I have made several updates:

- I replaced constant volatility with a stochastic volatility process based on a Heston model, updated using the Milstein scheme.
- I introduced a dynamic short-term mean using the Hull–White model.
- I combined the Hull–White mean with a longer-term yield obtained via the Nelson–Siegel–Svensson model; the NSS–Svensson parameters are selected according to market regime.
- I added a Markov-switching framework to alternate between different market conditions (calm versus turbulent).

These enhancements yield a more realistic simulation of the spread, capturing both short-term fluctuations and longer-term trends.

2 Model and Parameter Updates

Below are the scripts I used for the simulation setup and simulation of the spread.

```python
T = 2000
dt = 1
theta = 1.0

jump_intensity = 0.001
jump_mean = 0.0
jump_std = 0.01

kappa_hw = 0.3
theta_hw = 0.0
sigma_hw = 0.005
exp_factor_hw = np.exp(-kappa_hw * dt)
std_factor_hw = np.sqrt((sigma_hw**2 / (2 * kappa_hw)) * (1 - np.exp(-2 *
    kappa_hw * dt)))

v0 = 0.0001
kappa_v = 1.5
theta_v = 0.0001
sigma_v = 0.005
rho = -0.5

tau_nss = 5.0

spread = np.zeros(T)
mu_t_arr = np.zeros(T)
effective_mu_arr = np.zeros(T)
variance = np.zeros(T)
volatility = np.zeros(T)
regime_arr = np.zeros(T, dtype=int)

spread[0] = 0.0
mu_t_arr[0] = theta_hw
ns_yield0 = nelson_siegel_svensson(tau_nss, **nss_params_regime0)
effective_mu_arr[0] = theta_hw + ns_yield0
variance[0] = v0
volatility[0] = np.sqrt(v0)
regime_arr[0] = 0
```

Listing 1: Simulation Parameters and Model Setup

```python
for t in range(1, T):
    current_regime = regime_arr[t-1]
    # Generate a random number to determine next regime
    if np.random.rand() < P[current_regime, 1 - current_regime]:
        regime = 1 - current_regime
    else:
        regime = current_regime
    regime_arr[t] = regime

    Z_hw = np.random.normal()
    mu_t = theta_hw + (mu_t_arr[t-1] - theta_hw) * exp_factor_hw + std_factor_hw * Z_hw
    mu_t_arr[t] = mu_t

    if regime == 0:
        nss_params = nss_params_regime0
    else:
        nss_params = nss_params_regime1
    ns_yield = nelson_siegel_svensson(tau_nss, **nss_params)

    # Combine the short-term H u l l White mean with the NSS yield to form effective
    effective_mu = mu_t + ns_yield
    effective_mu_arr[t] = effective_mu

    Z1 = np.random.normal()
    Z2 = np.random.normal()
    Z_ou = Z1  # for the OU process
    Z_v = rho * Z1 + np.sqrt(1 - rho**2) * Z2  # for variance update

    v_prev = variance[t-1]
    sqrt_v_prev = np.sqrt(max(v_prev, 0))
    # Milstein correction: (sigma_v^2 / 4) * dt * (Z_v^2 - 1)
    dv = kappa_v * (theta_v - v_prev) * dt \
         + sigma_v * sqrt_v_prev * np.sqrt(dt) * Z_v \
         + (sigma_v**2 / 4) * dt * (Z_v**2 - 1)
    v_t = v_prev + dv
    v_t = max(v_t, 0)
    variance[t] = v_t
    volatility[t] = np.sqrt(v_t)

    exp_factor_ou = np.exp(-theta * dt)
    std_factor_ou = np.sqrt((v_prev / (2 * theta)) * (1 - np.exp(-2 * theta * dt)))
    X_t = effective_mu + (spread[t-1] - effective_mu) * exp_factor_ou + std_factor_ou * Z_ou

    if np.random.rand() < jump_intensity:
        jump = np.random.normal(jump_mean, jump_std)
    else:
        jump = 0.0
    spread[t] = X_t + jump
```

Listing 2: Applying the Siegle-Svensson model and Markov-switching framework alongisde our existing OUJ foundation

I also used the following code for plotting the results:

```
plt.figure(figsize=(14, 12))

plt.subplot(3, 1, 1)
plt.plot(spread, label="Spread")
plt.plot(effective_mu_arr, label="Effective    (HW + NSSS)", linestyle='--')
plt.xlabel("Time Steps")
plt.ylabel("Value")
plt.title("OU Process with Stochastic Volatility, Jumps, and Effective
    (HW + NSS-Svensson)")
plt.legend()
plt.grid(True)

plt.subplot(3, 1, 2)
plt.plot(volatility, label="Volatility (sqrt(v))", color='orange')
plt.xlabel("Time Steps")
plt.ylabel("Volatility")
plt.title("Stochastic Volatility Process (Heston, Milstein Scheme)")
plt.legend()
plt.grid(True)

plt.subplot(3, 1, 3)
plt.step(range(T), regime_arr, where='post', label="Regime", color='purple
    ')
plt.xlabel("Time Steps")
plt.ylabel("Regime (0=Calm, 1=Turbulent)")
plt.title("Markov-Switching Regime Indicator")
plt.legend()
plt.grid(True)

plt.tight_layout()
plt.show()
```

Listing 3: Plotting Results

3 Detailed Explanation of the New Implementations

3.1 Hull–White Update for the Short-Term Mean

The Hull–White model updates the short-term mean μ_t as:

$$\mu_t = \theta_{hw} + (\mu_{t-1} - \theta_{hw})e^{-\kappa_{hw}\,dt} + \sqrt{\frac{\sigma_{hw}^2}{2\kappa_{hw}}\left(1 - e^{-2\kappa_{hw}\,dt}\right)}\,Z_{hw}$$

where:

- μ_t is the updated short-term mean.
- θ_{hw} is the long-term mean level (here 0).
- μ_{t-1} is the previous short-term mean (e.g., 0.1).
- κ_{hw} is the speed at which the process reverts to the long-term mean.
- dt is the time increment.
- σ_{hw} is the volatility of the short-term mean process; a value of 0.005 indicates that the random fluctuations are on the order of 0.5%.

- Z_{hw} is a standard normal random variable.

Example: Suppose $\theta_{hw} = 0$, $\mu_{t-1} = 0.1$, $\kappa_{hw} = 0.3$, $dt = 1$, and $\sigma_{hw} = 0.005$. Then:

$$e^{-0.3} \approx 0.74, \qquad \sqrt{\frac{0.005^2}{0.6}(1 - e^{-0.6})} \approx 0.005.$$

For $Z_{hw} = 1$, we obtain:

$$\mu_t \approx 0 + 0.1 \times 0.74 + 0.005 \times 1 \approx 0.074 + 0.005 = 0.079.$$

This means the updated mean is 0.079, showing a reversion from the previous value (0.1) toward the long-term level (0) with a small noise contribution (0.005). In practical terms, 0.079 represents the new average level around which the process is expected to oscillate.

3.2 Heston Model with Milstein Scheme for Stochastic Volatility

The variance process in the Heston model is given by:

$$dv_t = \kappa_v(\theta_v - v_t)\,dt + \sigma_v\sqrt{v_t}\,dW_t^v,$$

and using the Milstein scheme, it becomes:

$$v_{t+1} = v_t + \kappa_v(\theta_v - v_t)\,dt + \sigma_v\sqrt{v_t}\,\sqrt{dt}\,Z_v + \frac{\sigma_v^2}{4}\,dt\,(Z_v^2 - 1).$$

Here:

- v_t is the variance at time t.
- θ_v is the long-run variance level.
- κ_v is the rate of mean reversion for the variance.
- σ_v is the volatility of volatility; a value of 0.005 indicates small but significant random fluctuations.
- dt is the time increment.
- Z_v is a standard normal random variable.

Example: Let $v_t = 0.0001$, $\theta_v = 0.0001$, $\sigma_v = 0.005$, $dt = 1$, and $Z_v = 1$. Then:

$$\text{Diffusion term} = 0.005 \times \sqrt{0.0001} = 0.005 \times 0.01 = 0.00005.$$

The Milstein correction is zero if $Z_v^2 - 1 = 0$. Thus, $v_{t+1} \approx 0.0001 + 0.00005 = 0.00015$. This increase represents a small upward adjustment in variance due to random shocks, indicating how volatility can gradually change.

3.3 Nelson–Siegel–Svensson Yield Curve

The NSS–Svensson formula is:

$$y(\tau) = \beta_0 + \beta_1 \frac{1 - e^{-\tau/\lambda_1}}{\tau/\lambda_1} + \beta_2 \left(\frac{1 - e^{-\tau/\lambda_1}}{\tau/\lambda_1} - e^{-\tau/\lambda_1} \right) + \beta_3 \left(\frac{1 - e^{-\tau/\lambda_2}}{\tau/\lambda_2} - e^{-\tau/\lambda_2} \right).$$

Here:

- $y(\tau)$ is the yield for a maturity τ.
- β_0 sets the base yield (level).
- β_1 controls the slope.
- β_2 and β_3 adjust the curvature.
- λ_1 and λ_2 are decay factors determining how quickly the effects of β_1, β_2, and β_3 diminish with τ.

Example: Using $\beta_0 = 0.02$, $\beta_1 = -0.01$, $\beta_2 = 0.01$, $\beta_3 = 0$, $\lambda_1 = 1.0$, $\lambda_2 = 2.0$, and $\tau = 5$: The base yield is 0.02 (or 2%). The second term roughly contributes -0.002 (since $\frac{1-e^{-5}}{5}$ is near 0.2), so $y(5) \approx 0.02 - 0.002 = 0.018$ (or 1.8%), with minor adjustments from the curvature terms. This tells me the yield for a 5-year maturity under these parameters.

3.4 Markov-Switching Regime

I use a two-state Markov chain with the transition matrix:

$$P = \begin{bmatrix} 0.95 & 0.05 \\ 0.10 & 0.90 \end{bmatrix},$$

where P_{ij} is the probability of moving from regime i to regime j. For example, if I am in the calm regime (state 0), there is a 95% chance to stay calm and a 5% chance to switch to the turbulent regime (state 1). This framework allows the model to adjust yield parameters and other dynamics based on market conditions.

4 Results and Discussion

The results from my simulation are visualized in the following graphs:

- **Spread and Effective Mean:** The top graph (Figure 1) shows the simulated spread alongside the effective mean (the sum of the Hull–White mean and the NSS–Svensson yield). This dynamic effective mean adapts with market regimes, providing a realistic drift for the spread.
- **Stochastic Volatility Process:** The middle graph (Figure 2) displays the evolution of the volatility process. The Heston-type variance is updated using the Milstein scheme, ensuring accurate and non-negative volatility.

- **Markov-Switching Regime Indicator:** The bottom graph (Figure 3) presents the regime indicator, which toggles between calm (0) and turbulent (1) conditions, highlighting the impact of market regimes.

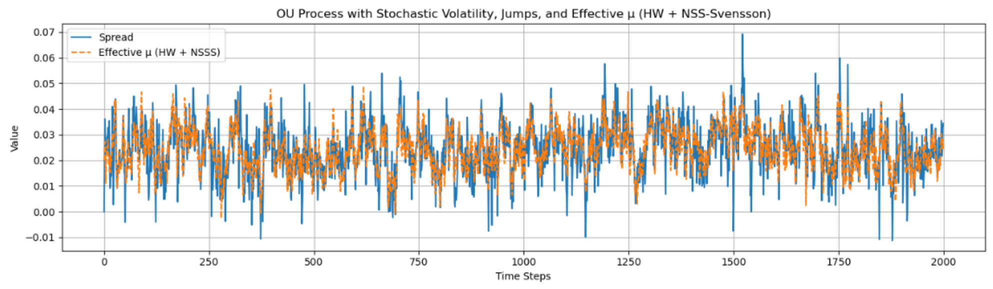

Figure 1: Spread and Effective Mean (HW + NSS-Svensson)

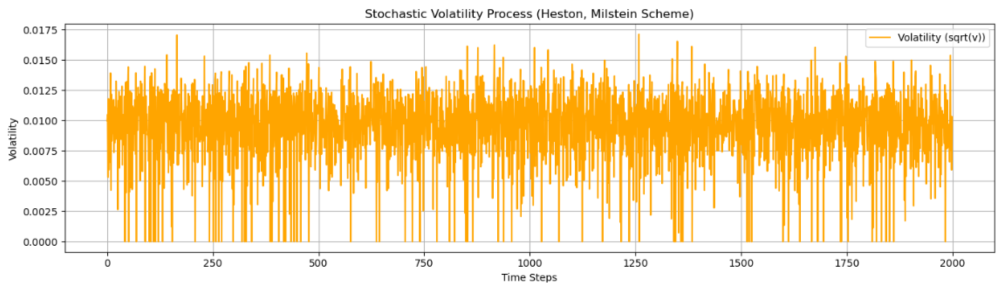

Figure 2: Stochastic Volatility Process (Heston, Milstein Scheme)

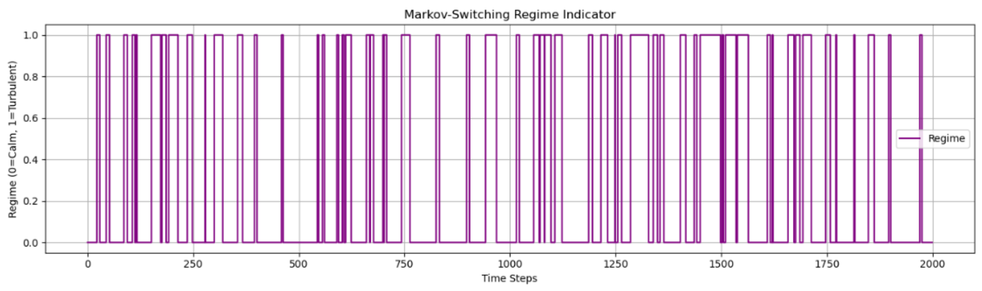

Figure 3: Markov-Switching Regime Indicator

5 Conclusion

In summary, my enhanced model now integrates several dynamic features into our current model to enhance our backtesting results compared to live market data. This is also an interesting topic for aspiring developers out there.

- **Stochastic Volatility:** The Heston model, updated with the Milstein scheme, allows the volatility to vary over time accurately.

- **Dynamic Mean:** The Hull–White model provides a short-term mean that I augment with a yield from the Nelson–Siegel–Svensson framework to capture both immediate and long-term trends.
- **Regime Adaptation:** A Markov-switching framework enables the model to adjust its parameters between calm and turbulent market conditions.

These enhancements provide a richer, more realistic picture of the spread dynamics in a high-volatility bond arbitrage environment.

In terms of next steps, I plan on implementing this into our existing trading strategy that uses Bayes Stats to estimate and trade based on the mean change from μ.

I hope this report clarifies the improvements and the impact of each component. If you have any questions or need further adjustments, feel free to reach out at 765-398-9053. I only check Discord at night.