OUJ + Bayesian Stats in Bond Arb Strategy

Quantitative Trading Club at UIC

February 17, 2025

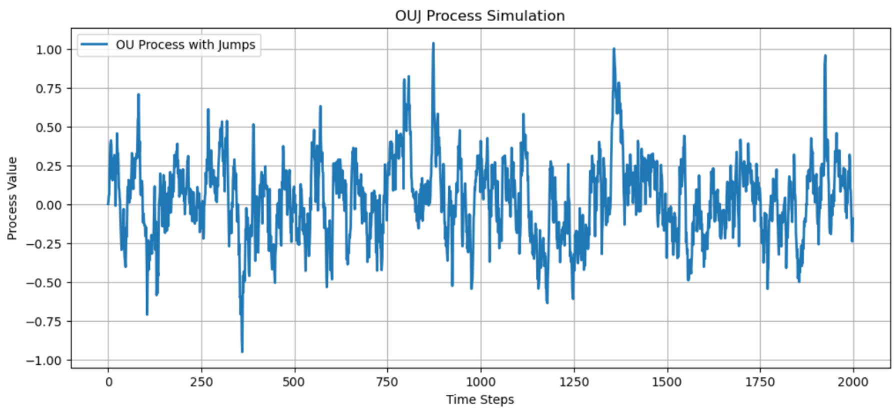
1 Simulating and graphing the OUJ Process

The script below essentially simulates the spread of two assets (bonds) with "Jumps" (sudden and abnormal price movements).

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import norm

# Simulate OU Process with Jumps
# Total number of time steps
# Time increment
# Mean reversion speed
# Long-run mean
# Diffusion volatility

T = 2000
dt = 1.0
theta = 0.1
mu = 0.0
sigma = 0.1

# Jump parameters (for jump diffusion)
# Probability of a jump occurring at any time step
# Mean jump size
# Jump volatility

jump_intensity = 0.02
jump_mean = 0.0
jump_std = 0.25

spread = np.zeros(T)
spread[0] = 0.0  # Starting value

for t in range(1, T):
    # Diffusion (OU) component: mean reversion plus normal shock
    diffusion = theta * (mu - spread[t-1]) * dt + sigma * np.sqrt(
        dt) * np.random.normal()

    # Jump component: add a jump if one occurs
    jump = np.random.normal(jump_mean, jump_std) if np.random.rand
        () < jump_intensity else 0.0

```

```
    # Update the process with both components
    spread[t] = spread[t-1] + diffusion + jump

plt.figure(figsize=(12, 5))
plt.plot(spread, lw=2, label='OU Process with Jumps')
plt.xlabel("Time Steps")
plt.ylabel("Process Value")
plt.title("Enhanced OU Process Simulation with Jumps")
plt.legend()
plt.grid(True)
plt.show()
```

1.1 Output



2 Applying Bayes stats to the OUJ output

The script below will analyze the spreads in the time series above to identify trading opportunities depending on each trading opportunity's z-score. When there's hard evidence of a potential downturn or upturn (high or high-negative z-score), then it will enter the position. It will then finally print out its trading log.

```
# Bayesian Backtesting with Adjusted z-score Thresholds
# Number of observations for Bayesian update
# Prior mean for the spread
# Prior variance (uncertainty about the mean)
# Known variance of observations
# cost per trade per leg (entry and exit), so total cost per trade
   = 2 * transaction_cost
# Exit long when z > n_1
# Exit short when z < -n_1

window_size = 40
mu0 = 0.0
```

```python
tau2 = 1.0
sigma2 = sigma**2
transaction_cost = 0.004
entry_threshold = 2
long_exit_threshold = 2
short_exit_threshold = -2

position = None
trades = []

print("Starting backtest...\n")

for t in range(window_size, T):
    # Bayesian Update: Use the most recent 'window_size' observations
    window_data = spread[t-window_size:t]
    n = len(window_data)
    X_bar = np.mean(window_data)

    sigma_post2 = 1.0 / (n / sigma2 + 1.0 / tau2)
    mu_post = sigma_post2 * ((n * X_bar / sigma2) + (mu0 / tau2))
    sigma_post = np.sqrt(sigma_post2)

    current_value = spread[t]
    # Compute the z-score relative to our Bayesian posterior estimate
    z_score = (current_value - mu_post) / sigma_post

    # If no position is open, check for entry signals
    if position is None:
        # Long entry: current spread is significantly lower than estimated mean.
        if z_score < -entry_threshold:
            position = {
                'type': 'long',
                'entry_time': t,
                'entry_value': current_value,
                'entry_z': z_score,
                'mu_post': mu_post,
                'sigma_post': sigma_post
            }
            print(f"Time {t}: LONG entry triggered. Spread = {current_value:.4f}, "
                  f"Posterior mean = {mu_post:.4f}, z-score = {z_score:.2f}")
        # Short entry: current spread is significantly higher than estimated mean.
        elif z_score > entry_threshold:
            position = {
                'type': 'short',
                'entry_time': t,
                'entry_value': current_value,
                'entry_z': z_score,
                'mu_post': mu_post,
                'sigma_post': sigma_post
            }
            print(f"Time {t}: SHORT entry triggered. Spread = {
```

```
                  current_value:.4f}, "
                  f"Posterior mean = {mu_post:.4f}, z-score = {z_score:.2f}")
    else:
        # If a position is open, check for the symmetric exit conditions.
        if position['type'] == 'long' and z_score > long_exit_threshold:
            exit_value = current_value
            position['exit_time'] = t
            position['exit_value'] = exit_value
            profit = (exit_value - position['entry_value']) - (2 * transaction_cost)
            position['profit'] = profit
            trades.append(position)
            print(f"Time {t}: Exiting LONG position opened at time {position['entry_time']} "
                  f"(entry spread = {position['entry_value']:.4f}). Exit spread = {exit_value:.4f}, "
                  f"Profit = {profit:.4f}\n")
            position = None  # Reset the position
        elif position['type'] == 'short' and z_score < short_exit_threshold:
            exit_value = current_value
            position['exit_time'] = t
            position['exit_value'] = exit_value
            profit = (position['entry_value'] - exit_value) - (2 * transaction_cost)
            position['profit'] = profit
            trades.append(position)
            print(f"Time {t}: Exiting SHORT position opened at time {position['entry_time']} "
                  f"(entry spread = {position['entry_value']:.4f}). Exit spread = {exit_value:.4f}, "
                  f"Profit = {profit:.4f}\n")
            position = None  # Reset the position

# If a position remains open at the end, close it at the final time step.
if position is not None:
    exit_value = spread[-1]
    position['exit_time'] = T - 1
    position['exit_value'] = exit_value
    if position['type'] == 'long':
        profit = exit_value - position['entry_value']
    else:
        profit = position['entry_value'] - exit_value
    position['profit'] = profit
    trades.append(position)
    print(f"Time {T-1}: Exiting remaining {position['type']} position opened at time {position['entry_time']} "
          f"(entry spread = {position['entry_value']:.4f}). Exit spread = {exit_value:.4f}, "
          f"Profit = {profit:.4f}\n")
```

2.1 Output

```
    Starting backtest...

Time 40: LONG entry triggered. Spread = -0.2092, Posterior mean = 0.1376, z-score = -21.93
Time 52: Exiting LONG position opened at time 40 (entry spread = -0.2092). Exit spread = 0.0630, Profit = 0.2642

Time 53: SHORT entry triggered. Spread = 0.1116, Posterior mean = -0.0008, z-score = 7.11
Time 87: Exiting SHORT position opened at time 53 (entry spread = 0.1116). Exit spread = 0.0645, Profit = 0.0391

Time 88: LONG entry triggered. Spread = 0.0584, Posterior mean = 0.1958, z-score = -8.69
Time 124: Exiting LONG position opened at time 88 (entry spread = 0.0584). Exit spread = 0.0025, Profit = -0.0639

Time 125: SHORT entry triggered. Spread = 0.1147, Posterior mean = -0.1650, z-score = 17.69
Time 127: Exiting SHORT position opened at time 125 (entry spread = 0.1147). Exit spread = -0.2953, Profit = 0.4019

Time 129: SHORT entry triggered. Spread = -0.1127, Posterior mean = -0.1948, z-score = 5.20
Time 130: Exiting SHORT position opened at time 129 (entry spread = -0.1127). Exit spread = -0.5863, Profit = 0.4656

...

Time 1973: SHORT entry triggered. Spread = 0.1872, Posterior mean = 0.1327, z-score = 3.44
Time 1976: Exiting SHORT position opened at time 1973 (entry spread = 0.1872). Exit spread = 0.1071, Profit = 0.0720

Time 1977: LONG entry triggered. Spread = 0.1002, Posterior mean = 0.1472, z-score = -2.98
Time 1978: Exiting LONG position opened at time 1977 (entry spread = 0.1002). Exit spread = 0.2234, Profit = 0.1152

Time 1979: LONG entry triggered. Spread = -0.0450, Posterior mean = 0.1611, z-score = -13.03
Time 1987: Exiting LONG position opened at time 1979 (entry spread = -0.0450). Exit spread = 0.2133, Profit = 0.2502

Time 1988: LONG entry triggered. Spread = 0.1065, Posterior mean = 0.1784, z-score = -4.55
Time 1990: Exiting LONG position opened at time 1988 (entry spread = 0.1065). Exit spread = 0.3182, Profit = 0.2037

Time 1991: SHORT entry triggered. Spread = 0.2820, Posterior mean = 0.1840, z-score = 6.20
Time 1992: Exiting SHORT position opened at time 1991 (entry spread = 0.2820). Exit spread = 0.1015, Profit = 0.1725

```

```
Time 1993: LONG entry triggered. Spread = 0.0873, Posterior mean = 0.1765, z-score = -5.64
Time 1999: Exiting remaining long position opened at time 1993 (entry spread = 0.0873). Exit spread = -0.0936, Profit = -0.1809
```

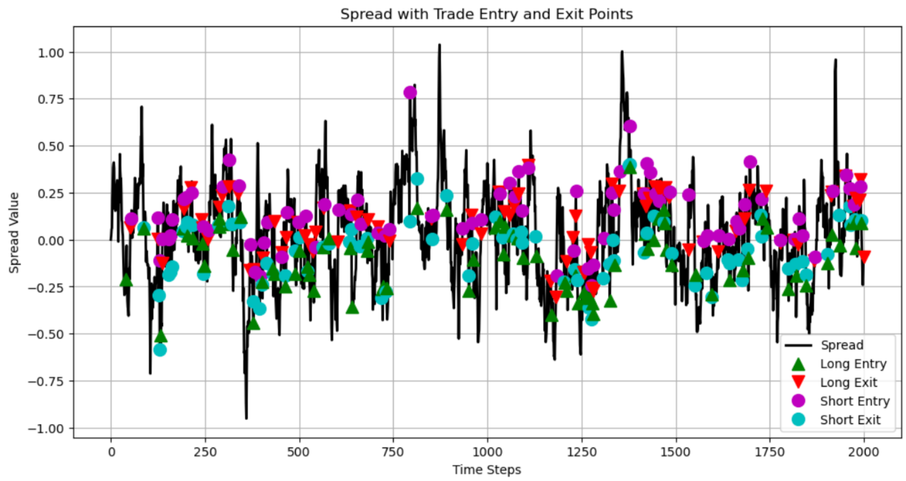
3 Visualizing trades

This script will help you visualize the trades as seen in the script above's outputs.

```python
    plt.figure(figsize=(12, 6))
plt.plot(spread, label="Spread", lw=2, color="black")

# For avoiding duplicate legend entries, we'll keep track of what labels we have already plotted.
plotted_labels = set()

for trade in trades:
    entry_time = trade['entry_time']
    exit_time = trade['exit_time']
    entry_val = trade['entry_value']
    exit_val = trade['exit_value']

    if trade['type'] == 'long':
        if "Long Entry" not in plotted_labels:
            plt.plot(entry_time, entry_val, 'g^', markersize=10, label="Long Entry")
            plotted_labels.add("Long Entry")
        else:
            plt.plot(entry_time, entry_val, 'g^', markersize=10)

        if "Long Exit" not in plotted_labels:
            plt.plot(exit_time, exit_val, 'rv', markersize=10, label="Long Exit")
            plotted_labels.add("Long Exit")
        else:
            plt.plot(exit_time, exit_val, 'rv', markersize=10)

    elif trade['type'] == 'short':
        if "Short Entry" not in plotted_labels:
            plt.plot(entry_time, entry_val, 'mo', markersize=10, label="Short Entry")
            plotted_labels.add("Short Entry")
        else:
            plt.plot(entry_time, entry_val, 'mo', markersize=10)

        if "Short Exit" not in plotted_labels:
            plt.plot(exit_time, exit_val, 'co', markersize=10, label="Short Exit")
            plotted_labels.add("Short Exit")
        else:
            plt.plot(exit_time, exit_val, 'co', markersize=10)

plt.xlabel("Time Steps")
plt.ylabel("Spread Value")
plt.title("Spread with Trade Entry and Exit Points")
```

```
plt.legend()
plt.grid(True)
plt.show()
```

3.1 Output



4 Final Results and Conclusion

To visualize and see how this strategy performed overall, use the script below to figure that out! More specifically, I wrote a script to find our P&L, Margin (P&L if accounted for transaction costs), total amount of trades, Turnover, and Fitness.

```
total_profit = sum(trade['profit'] for trade in trades)
print("Analysis:")
print(f"Total number of trades executed: {len(trades)}")
print(f"Total Profit from strategy: {total_profit:.4f}")

# Calculate total turnover: sum of the absolute differences between entry and exit values for all trades.
total_turnover = sum(abs(trade['exit_value'] - trade['entry_value']) for trade in trades)

# Define a simple fitness metric as net profit divided by total turnover.
fitness = total_profit / total_turnover if total_turnover != 0 else 0.0

print(f"Total Turnover (Margin Used): {total_turnover:.4f}")
print(f"Strategy Fitness (Profit per Unit of Turnover): {fitness:.4f}")
```

4.1 Output

```
Analysis:
Total number of trades executed: 165
Total Profit from strategy: 28.9988
Total Turnover (Margin Used): 33.0670
Strategy Fitness (Profit per Unit of Turnover): 0.8770
```

Overall, the strategy is still very much in progress as our Fitness needs to be above 1 (ideally). Our Margin could also be higher realistically since I somewhat high-balled the transaction costs (=0.004) so that's one positive to keep in mind.