Algorithmic Trading – Tutorial #3

Name: Paolo Bussacchini
Student Number: 717119

Due Date: January 14, 2022

Introduction

It has been widely observed that many asset prices exhibit mean reversion, for example foreign exchange rate as well as commodities. In industry, hedge fund managers and investors often attempt to construct mean-reverting prices by simultaneously taking positions in two highly correlated or co-moving assets. The advent of exchange-traded funds (ETFs) has further facilitated this pairs trading approach since some ETFs are designed to track identical or similar indexes and assets.
Given the price dynamics of some risky assets, one important problem commonly faced by individual and institutional investors is to determine when to open and close a position. While observing the prevailing market prices, a speculative investor can choose to enter the market immediately or wait for a future opportunity. After completing the first trade, the investor will need to decide when is best to close the position.
We would like to better understand how mean-reverting Ornstein-Uhlenbeck (OU) processes work, and how they can be used to model trading strategies.
Let us introduce the continuous Ornstein-Uhlenbeck process, which is defined by the following dynamics:

$$d\xi_t = \kappa(\theta - \xi_t)dt + \sigma dW_t$$

where W_t is a standard Brownian motion, σ is the volatility, θ is the level of mean reversion and $\kappa > 0$ is its speed.
First of all we discretize the continuous Ornstein-Uhlenbeck process and simulate a random path checking its mean-reverting property. Then we address the model calibration using a simple linear regression, trying to infer the model parameters given the simulated data. In part III we formulate a trading strategy for a certain portfolio of financial assets which can be modeled by the OU process, which allows us to maximize our profits by buying and selling ξ at proper times. Finally we will try to predict the best entry and exit points of the trade by means of a numerical solution of the associated free boundary problems or variational inequalities.

Question 1.

We set a discretization step $\delta t > 0$ and note that the OU process satisfies:

$$\begin{cases} d\xi_u = \kappa(\theta - \xi_u)du + \sigma dW_u \\ \forall u \in [t, t + \delta t] \end{cases} \tag{1}$$

Applying the Ito formula to the the change of variable $\widetilde{\xi_u} = e^{\kappa u}\xi_u - \theta e^{\kappa u}$ we compute:

$$d\widetilde{\xi_u} = (\kappa e^{\kappa u}\xi_u - \kappa\theta e^{\kappa u})du + e^{\kappa u}d\xi_u$$

where $d\xi_u$ is given by the process dynamics:

$$d\widetilde{\xi_u} = (\kappa e^{\kappa u}\xi_u - \kappa\theta e^{\kappa u})du + e^{\kappa u}(\kappa(\theta - \xi_u)du + \sigma dW_u) = e^{\kappa u}\sigma dW_u$$

Integrating between t and $t + \delta t$:

$$\int_t^{t+\delta t} d\widetilde{\xi_u} = \int_t^{t+\delta t} e^{\kappa u}\sigma dW_u$$

$$\widetilde{\xi_{t+\delta t}} - \widetilde{\xi_t} = \sigma \int_t^{t+\delta t} e^{\kappa u} dW_u$$

Using basic properties of the stochastic integral such as the Ito isometry we can infer that:

$$\widetilde{\xi_{t+\delta t}} - \widetilde{\xi_t} \sim \mathcal{N}\left(0, \sigma^2 \int_t^{t+\delta t} e^{2\kappa u} du\right) \sim \mathcal{N}\left(0, \sigma^2 \frac{e^{2\kappa(t+\delta t)}}{2\kappa}(1 - e^{-2\kappa\delta t})\right)$$

which, substituting the values of $\widetilde{\xi_u}$, leads to:

$$e^{\kappa(t+\delta t)}\xi_{t+\delta t} - \theta e^{\kappa(t+\delta t)} = e^{\kappa t}\xi_t - \theta e^{\kappa t} + \sigma \int_t^{t+\delta t} e^{\kappa u} dW_u$$

$$e^{\kappa t}e^{\kappa\delta t}\xi_{t+\delta t} = e^{\kappa t}\xi_t - \theta e^{\kappa t}(1 - e^{\kappa\delta t}) + \sigma \int_t^{t+\delta t} e^{\kappa u} dW_u$$

$$\xi_{t+\delta t} = e^{-\kappa\delta t}\xi_t + \theta(1 - e^{-\kappa\delta t}) + \sigma e^{-\kappa(t+\delta t)} \int_t^{t+\delta t} e^{\kappa u} dW_u$$

where the last term is distributed as $\mathcal{N}\left(0, \sigma^2 \frac{1-e^{-2\kappa\delta t}}{2\kappa}\right)$

Hence:

$$\xi_{t+\delta t} = e^{-\kappa\delta t}\xi_t + \theta(1 - e^{-\kappa\delta t}) + \sigma\sqrt{\frac{1 - e^{-2\kappa\delta t}}{2\kappa}}\varepsilon_t$$

where $\varepsilon_t \sim \mathcal{N}(0, 1)$.

Question 2.

We implement the function to generate the random realization of ξ_k:

- Z is the array containing the random values sampled from $\mathcal{N}(0, 1)$
- X is the array containing the discretized values of ξ_k
- dt is the timestep $\delta t = \frac{T}{N}$

```
def GeneratePaths(NoOfPaths,NoOfSteps,T,sigma,xi_0, theta, kappa):
    # Fixing random seed
    np.random.seed(1)

    Z = np.random.normal(0.0,1.0,[NoOfSteps])
    X = np.zeros([NoOfSteps+1])

    time = np.zeros([NoOfSteps+1])

    X[0] = xi_0

    dt = T / float(NoOfSteps) #delta_t

    for i in range(0,NoOfSteps):
        # making sure that samples from normal have mean 0 and variance
                                                    1
        if NoOfPaths > 1:
            Z[i] = (Z[i] - np.mean(Z[i])) / np.std(Z[i])

        X[i+1] = np.exp(-kappa * dt)*X[i] + theta * (1 - np.exp(-kappa
                                                    * dt)) + sigma * np.power((
                                                    1 - np.exp(-kappa * dt))/(2
                                                    *kappa), 0.5)*Z[i]
        time[i+1] = time[i] +dt


    paths = {"time":time,"X":X}
    return paths
```

We plot one realization of ξ_k for $T = N = 1000$, $\sigma = 0.5$, $\theta = 1$, $\kappa = 0.5$, which as we can see in the graph has the desired mean-reverting property around the level θ.

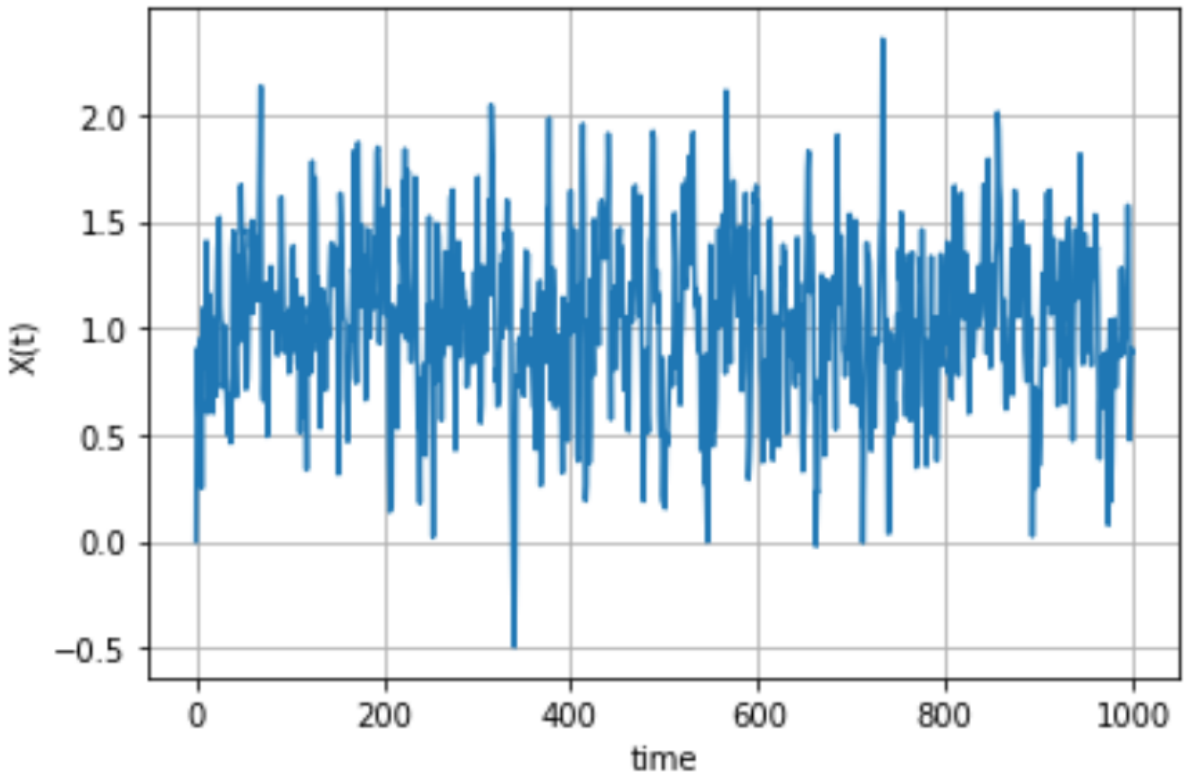

Figure 1: Mean-reverting behavior of the process

Question 3.

We regress ξ_{k+1} on ξ_k importing the function `LinearRegression()` from the package `sklearn`.

$$\xi_{k+1} = a\xi_k + b + \eta_k$$

From the model above, which we have just simulated, we infer that:

$$a = e^{-\kappa\delta t}$$

$$b = \theta(1 - e^{-\kappa\delta t})$$

And the error term is:

$$\sigma\sqrt{\frac{1 - e^{-2\kappa\delta t}}{2\kappa}}\varepsilon_t$$

Therefore, since $\delta t = 1$, we get the formulas for κ and θ from the slope and the intercept:

$$\kappa = -\log(a)$$

$$\theta = \frac{b}{1 - e^{-\kappa}}$$

The error term η_k is distributed as a $\mathcal{N}\left(0, \sigma^2 \frac{1-e^{-2\kappa}}{2\kappa}\right)$, so taking its standard deviation and dividing by $\sqrt{\frac{1-e^{-2\kappa}}{2\kappa}}$ we obtain σ.

We apply these explicit formulas in order to calibrate the model and we obtain the following values, which are close to the real parameters:

- $\kappa = 0.54$
- $\theta = 1.03$
- $\sigma = 0.56$

Question 4.

Applying the same function written for the previous question to the simulated data we obtain the following values:

- $\kappa = 0.48$
- $\theta = 1.04$
- $\sigma = 0.73$

There is a mistake in the computation of σ (unless the σ used to generate the data was closer to 0.7). Trying to change this parameter in our simulation doesn't seem to affect the robustness of the model calibration algorithm, which maintains close estimates.

Question 5.

We implement the function to simulate the strategy:

```
def simulateStrategy(entryValue, exitValue, T, X):
    c = 0.01
    pos = 0
    cash = 0
    cumpnl = 0
    numberTrades = 0
    for t in range(1, T+1):
        if pos == 0 and X[t] <= entryValue:
            numberTrades += 1
            pos = 1
            cash = cash - X[t] - c
        elif pos == 1 and X[t] >= exitValue:
            pos = 0
            cash = cash + X[t] - c
        else:
            pos = pos
            cash = cash
        cumpnl = cash + pos*X[t]
    average_pnl = cumpnl/numberTrades
    return average_pnl
```

Question 6.

Setting $T = N = 1000000$ and using the code in question 5, we compute the average pnl per trade for the specified entry and exit points, producing the following table:

entry \ exit	0.5	1	1.5	2
-0.4	1.25	1.68	2.13	2.59
0	0.85	1.29	1.75	2.22
0.4	0.51	0.92	1.37	1.84

Table 1: Average pnl per trade as a function on entry and exit points

The average pnl per trade is maximized with the pair entry_value = -0.4, exit_value = 2.

Question 7.

While in question 5 we simulated a strategy to compute average pnl per trade for a given pair of entry and exit points we will now try to predict what the optimal entry/exit thresholds are.

We know that given a long position in a mean-reverting portfolio ξ following an OU process, the optimal profit from entering a long position, holding, and then exiting the long position (optimal entry/exit problem for a long position) satisfies the variational inequality known as obstacle problem:

$$\max\{(\mathcal{L} - \rho)G(\xi), \max(H_-(\xi) + \xi - c, H_+(\xi) - \xi - c) - G(\xi)\} = 0$$

Where:

- $\max(H_-(\xi) + \xi - c, H_+(\xi) - \xi - c)$ is the obstacle function
- $\mathcal{L}$ is the differential operator associated to the OU process and $c > 0$ is a transaction cost
- H_+ is the optimal exit function, which satisfies:

  $$\max\{(\mathcal{L} - \rho)H_+(\xi), (\xi - c) - H_+(\xi)\} = 0$$

  where ρ is a discount factor and $\psi_+ = \xi - c$ is the obstacle function

- H_- is the optimal entry function, which satisfies :

  $$\max\{(\mathcal{L} - \rho)H_-(\xi), (-\xi - c) - H_-(\xi)\} = 0$$

  where $\psi_- = -\xi - c$ is the obstacle function

Our goal is now to compute numerically H_+, H_-, G, and deduce the optimal entry and exit level of the trading strategy. Assuming $\xi_{min} = -2, \xi_{max} = 2, n = 1000, \kappa = 0.5, c = 0.01, \rho = 0.01, \sigma = 0.5, \theta = 1$ we first compute ξ_k using a uniform discretization grid with $n > 0$ points:

$$\xi_k = \xi_{min} + \frac{k}{n-1}(\xi_{max} - \xi_{min})$$

Then, we solve the obstacle problems for H_+, H_- and G using the function `SolveObstacleProblem(kappa, rho, sigma, theta, xi, phi)` where ξ is the numpy array of ξ_k, and φ is the numpy array for the obstacle function, with $0 \leq k \leq n-1$. The function SolveObstacleProblem returns the solution for each of the three problems computed on the grid in the form of a numpy array. Now that we have computed $G_k := G(\xi_k)$, we know that the optimal entry and exit points for the trading strategy ξ_- and ξ_+ are the points where G 'lifts off from the obstacle' ('smooth pasting principle'), i.e.: ξ_{k0} and ξ_{k1}, where $k0$ is the first index from the left (i.e. starting with $k = 0$) for which $G_k > \varphi_k + \varepsilon$ and $k1$ is the first index from the right (i.e. starting with $k = n-1$) for which $G_k > \varphi_k + \varepsilon$.
The values we found are:

$$\xi_{k0} = -0.37, \ \xi_{k1} = 1.92$$

which are the closest to the optimal entry/exit values computed in question 6.
The conclusion we can draw from this is that the entry/exit thresholds computed solving the obstacle problem are very close to the ones computed maximizing the average pnl per trade, which gives us an indication of how well the numerical solution of the variational inequality is able to predict the optimal entry/exit points in the trading strategy.

Question 8.

Taking the value in between ξ_{k0} and ξ_{k1} we obtain 0.775, which is fairly close to $\theta = 1$. This is probably due to the fact that it is optimal to entry (exit) the position when the asset is at its lowest (highest), therefore farther from the mean (and hence symmetric being the process mean-reverting).

Conclusion

The Ornstein-Uhlenbeck process is a stochastic process that has a tendency to drift towards its mean function. In the financial markets, the investors can observe assets prices reverting back to their long-run mean. Other examples of mean-reversion are visible in the dynamics of the rate and volatility, which are both fundamental subjects that are extensively researched in finance.
We have seen how to discretize the model and simulate a random path, calibrate the model using linear regression, compute an optimal trading strategy by means of the average pnl per trade and solving numerically the free boundary problem associated to predicting the best entry/exit points in our trading strategy.
Results show that the obstacle problem allow us to successfully predict the optimal thresholds from entering a long position, holding, and then exiting the long position.