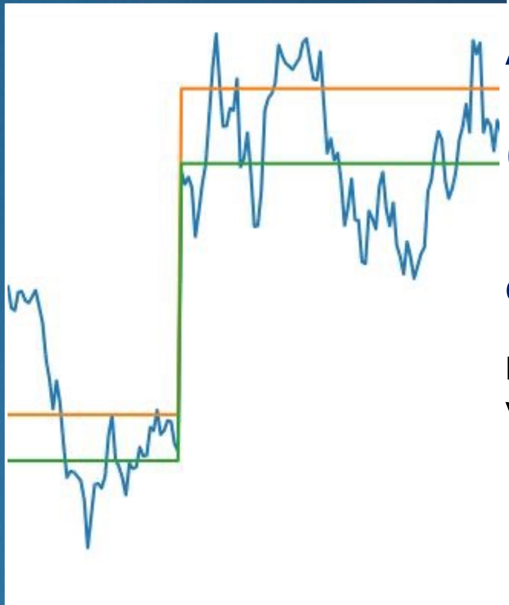


Algorithmic Trading Division

Ornstein-Uhlenbeck Pair Trading

Optimal Mean Reversion Trading

Brabec, Bronauer, Kolev, Shkola

Vienna, January, 2025

Team Overview

Algorithmic Trading

Tomislav Kolev

Associate

- Strategy
- Python Coding

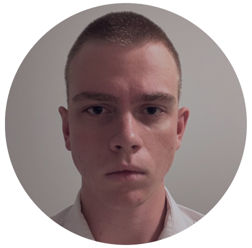
BSc. BBE – 5th Sem.

Emma Bronauer

Analyst

- Strategy Research
- Slidesdeck

BSc. BBE – 3rd Sem.



Denys Shkola

Fellow Analyst

- Python Coding
- Strategy
- Project Planning

BSc. Mathematics – 3rd Sem.

Karel Brabec

Fellow Analyst

- Macro Research
- Strategy

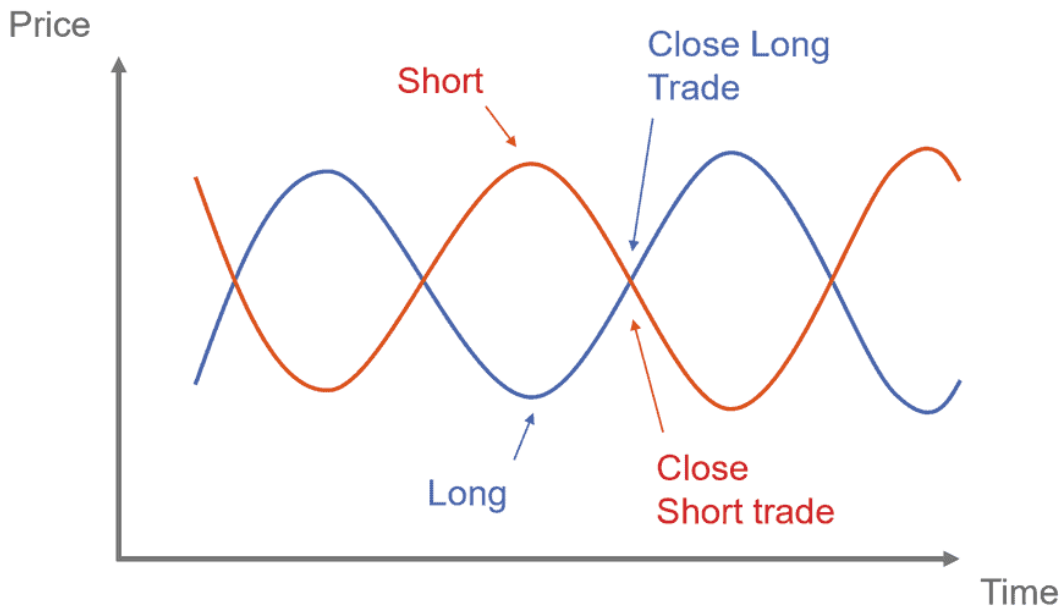
MSc. Qfin – 3rd Sem.
BSc. Finance

Introduction to Pairs Trading and Mean Reversion

Basics of pairs trading and mean reversion.

What is Pairs Trading?

- Pairs trading involves buying one asset (long) and selling another (short) simultaneously.
- Targets highly correlated assets with similar price patterns.
- Aims to minimize exposure to market-wide trends while profiting from temporary price deviations.
- Relies on identifying pairs with stable historical relationships.



What is Mean Reversion?

- Mean reversion is the theory that prices or returns tend to return to their long-term average over time.
- Significant deviations from the mean create opportunities to buy undervalued assets and sell overvalued ones.
- It underpins strategies like pairs trading and statistical arbitrage.
- Common indicators include moving averages, Bollinger Bands, and RSI.

Mean Reversion in Pairs Trading

Objective: Profit from the convergence of the price ratio to its historical mean.

Execution:

- Take a long position in the undervalued asset (below the mean).
- Take a short position in the overvalued asset (above the mean).

Challenges:

- Requires precise pair selection.
- Market corrections may take longer than anticipated, impacting returns.

The Ornstein-Uhlenbeck (OU) Process

Understanding the stochastic process behind the strategy.

What is the OU Process?

- The OU process is a **stochastic model** used to describe **mean-reverting behavior**. It describes how a variable X_t fluctuates around a long-term mean θ, influenced by unpredictable market fluctuations (random noise).
- The process for modeling portfolio value is defined as:

$$dX_t = \mu(\theta - X_t)\,dt + \sigma\,dB_t$$

  - X_t = Value of process at time t
  - θ = Long-term mean
  - μ = Rate of mean reversion
  - σ = Volatility
  - B_t = Standard Brownian Motion

Understanding the Dynamics:

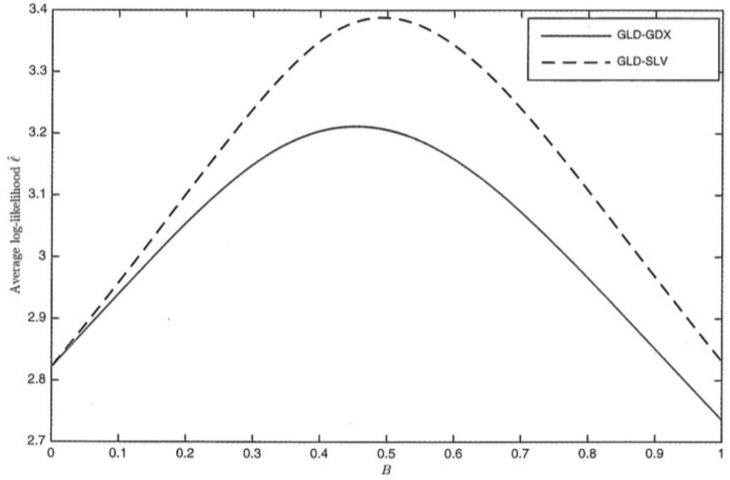
- The OU process illustrates how **price spreads tend to revert** to their average over time, making it foundational for pairs trading strategies.
- Maximum Likelihood Estimation is used to estimate the parameters μ,θ and σ, optimizing portfolio decisions based on observed mean-reversion behavior.
- Mean Reversion:
  - When $X_t > \theta$, the term $\mu(\theta - X_t)$ becomes negative, pulling X_t back toward θ
  - When $X_t < \theta$, the term $\mu(\theta - X_t)$ becomes positive, pushing X_t back toward θ

Portfolio

- A **mean-reverting portfolio** is constructed by holding a long position in α shares of asset $S^{(1)}$ and a short position in β shares of asset $S^{(2)}$:

$$X_t^{\alpha,\beta} = \alpha S_t^{(1)} - \beta S_t^{(2)}$$

- α = Long position in $S^{(1)}$ and β = Short position in $S^{(2)}$
- X_t = The value of the portfolio at time t
- **Scaling the Portfolio**: Fixing $\alpha = 1$ simplifies adjustments to β.
- **Example**: Setting $\beta = B / S_0^{(2)}$ optimizes the portfolio based on available cash B.



Python Implementation

Key Functions in the Trading Workflow.

Calculate Likelihood

Objective: Estimate optimal OU model parameters (β,θ,μ,σ) by maximizing log-likelihood using historical data.

1. **Optimize Portfolio Weights (β)**: Identify the hedge ratio that maximizes the model's likelihood.
2. **Maximize Log-Likelihood**: Select the β that results in the highest likelihood, ensuring the model is most aligned with historical data.
3. **Parameter Refinement**: After finding the optimal β, recalculate the OU parameters (μ,θ,σ) for precise modeling.
4. **Output Optimal Parameter**s: Generate optimized parameters and log-likelihood metrics.

Set Thresholds

Objective: Define critical trading thresholds to signal entry and exit points based on market conditions.

1. **Threshold Calculation:** Compute upper/lower bounds for the spread using θ, μ, σ.
2. **Account for Transaction Costs:** Incorporate real-world constraints when setting thresholds.
3. **Set Trading Thresholds**

Setup Model

Objective: Initialize the model with historical price data and compute key parameters for the OU trading strategy.

1. Establish the **initial price levels** of both assets for spread calculations.
2. Estimate the OU process **parameters (*μ, σ, θ*)** to model the mean-reverting behavior.
3. Compute the **hedge ratio (*β*)** for optimizing the portfolio.
4. Set the **upper and lower trading thresholds** based on statistical properties of the spread.

Recalculate Amounts

Objective: Dynamically adjust portfolio holdings based on asset price changes to maintain a mean-reverting hedge.

1. **Transaction Cost Management:** Deduct transaction costs from the capital, ensuring realistic portfolio management.
2. **Reallocation of Capital:** Calculate the optimal number of shares to hold for each asset to maintain the portfolio's hedged position.

Python Implementation

Key Functions in the Trading Workflow.

W U T I S

Trading Function

```
def trade(self, prices):
    """
    Execute trading logic based on the current prices of the assets.

    Args:
        prices (list): Current prices of the two assets.

    Raises:
        Exception: If the model is not set up with initial prices.
    """
    signal = False
    if self.init_prices is None:
        raise Exception("Model not setup")

    # Calculate the spread index based on initial prices and hedge ratio
    self.index = prices[0] / self.init_prices[0] - self.beta * prices[1] / self.init_prices[1]

    # Update portfolio value based on price changes and current positions
    if self.amount_a is not None and self.amount_b is not None:
        if self.is_long:  # Long position
            self.p_a += (prices[0] - self._last_a) * self.amount_a
            self.p_b += (self._last_b - prices[1]) * self.amount_b
        else:  # Short position
            self.p_a += (self._last_a - prices[0]) * self.amount_a
            self.p_b += (prices[1] - self._last_b) * self.amount_b

    # Update total capital
    if self.update_capital and self.p_a is not None and self.p_b is not None:
        self.capital = self.p_a + self.p_b

    # Check if trading thresholds are crossed and adjust positions
    if self.index <= self.l_threshold and (self.is_long is None or not self.is_long): # buy signal (long)
        self.recalculate_amounts(prices)
        signal = True
        self.is_long = True
    if self.index >= self.u_threshold and (self.is_long is None or self.is_long):
        self.recalculate_amounts(prices)
        signal = True
        self.is_long = False

    # Store the last prices
    self._last_a, self._last_b = prices

    # New variable to track order signals
    return signal
```

Purpose and Core Objectives

Objective: Implement a robust trading strategy for mean-reverting pairs using the Ornstein-Uhlenbeck (OU) process.

1. **Spread index Calculation**: Computes the relative price difference between assets, adjusted by the hedge ratio (β).
2. **Portfolio Value Update**: Reflects gains or losses based on price changes and current positions (long or short).
3. **Signal Generation**: Identifies trading opportunities based on upper/lower thresholds.
4. **Capital Tracking**: Monitors portfolio capital to reflect real-time price changes..

Trading Logic and Execution

- Trading Signal Logic:
  - **Buy Signal (Long)**: Triggered when the spread index crosses below the lower threshold, indicating an undervalued condition.
  - **Sell Signal (Short)**: Triggered when the spread index exceeds the upper threshold, indicating an overvalued condition.
- Output: Returns True if a trade signal is generated, otherwise False.

Python Implementation

Real-Time Data Handling and Trading Execution using Alpaca.

Connect to API, Search Pairs and Stream Data

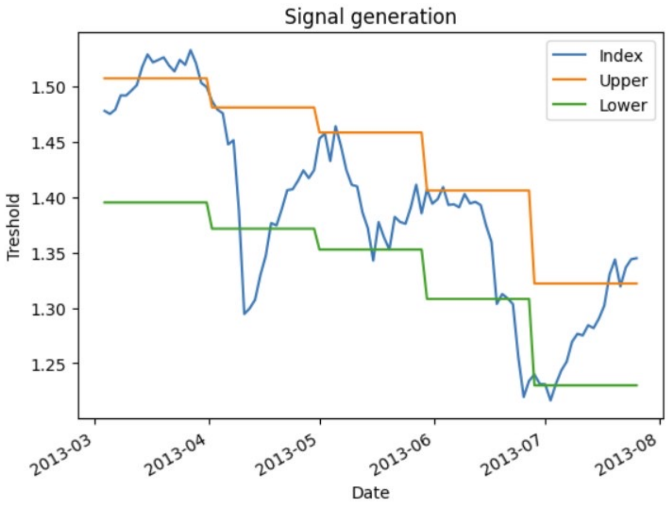
```python
if __name__ == '__main__':

    with open("config.yaml") as stream:
        try:
            data = yaml.safe_load(stream)
            api_key = data['api_key']
            secret_key = data['secret_key']
        except yaml.YAMLError as exc:
            print(exc)
    trading_client = TradingClient(api_key=api_key,
                                   secret_key=secret_key,
                                   paper=True,
                                   url_override=base_url)
    account = trading_client.get_account()
    print('Account cash:',account.cash)
    model = OU_Trading_Model(int(account.cash)*0.1)

    search_params = GetAssetsRequest(asset_class=AssetClass.US_EQUITY)
    assets = trading_client.get_all_assets(search_params)

    symbols = [asset.symbol for asset in assets if asset.tradable and asset.shortable]
    print(symbols)
    price_data = get_price_data(symbols,api_key,secret_key,test=True)
    result = analyze_tickers(price_data,dt)[:10]

    for i,out in enumerate(result):
        print(f"{i+1}. {out['t1']}-{out['t2']}: {out['likl']}")

    ind = input('Choose pair:')
    try:
        ind = int(ind)-1
        print(f"Trading pair: {result[ind]['t1']}-{result[ind]['t2']}")

        pair = [result[ind]['t1'],result[ind]['t2']]
        stream = StockDataStream(api_key, secret_key)
        stream.subscribe_bars(quote_data_handler, *pair)
        stream.run()
    except ValueError:
        print('Not an integer.')
```

Trading function triggered by API-Stream

```python
async def quote_data_handler(data):

    global pair,work_data,dt_from_start,model,status,past_data,trading_client,capital

    with open("config.yaml") as stream:
        try:
            data = yaml.safe_load(stream)
            interval = data['interval']
            window = data['window']
        except yaml.YAMLError as exc:
            print(exc)

    tickers = list(work_data.keys())

    if not data.symbol in tickers:
        work_data[data.symbol] = [data.close]
    else:
        work_data[data.symbol].append(data.close)

    status[data.symbol] = True

    if len(tickers) == 2 and all(list(status.values())):
        n_closes = len(past_data)
        dt_from_start += 1

        prices = [work_data[tickers[0]][-1],work_data[tickers[1]][-1]]
        print(prices)
        past_data.append(prices)

        model.capital = trading_client.get_account().portfolio_value

        if n_closes >= window and dt_from_start % interval == 0:
            print('setup',past_data)
            model.setup(past_data)
        try:
            signal = model.trade(prices)
            if signal:
                spread(pair,model.is_long,[model.amount_a,model.amount_b])
            print('trade',signal,model.is_long)
        except:
            pass

        for ticker in tickers:
            status[ticker] = False
```

Results for GLD-SLV

Analysing Strategy Performance: Results and Insights.



Implementation

- The strategy was backtested using historical price data for GLD and SLV.
- **Timeframe**: From March 2013 to August 2024.
- **Parameters** recalculated every 20 days.
- **Source**: Historical ETF data, processed through the OU trading model.
- **Trades Executed**: 110 trades were executed during the backtesting period.

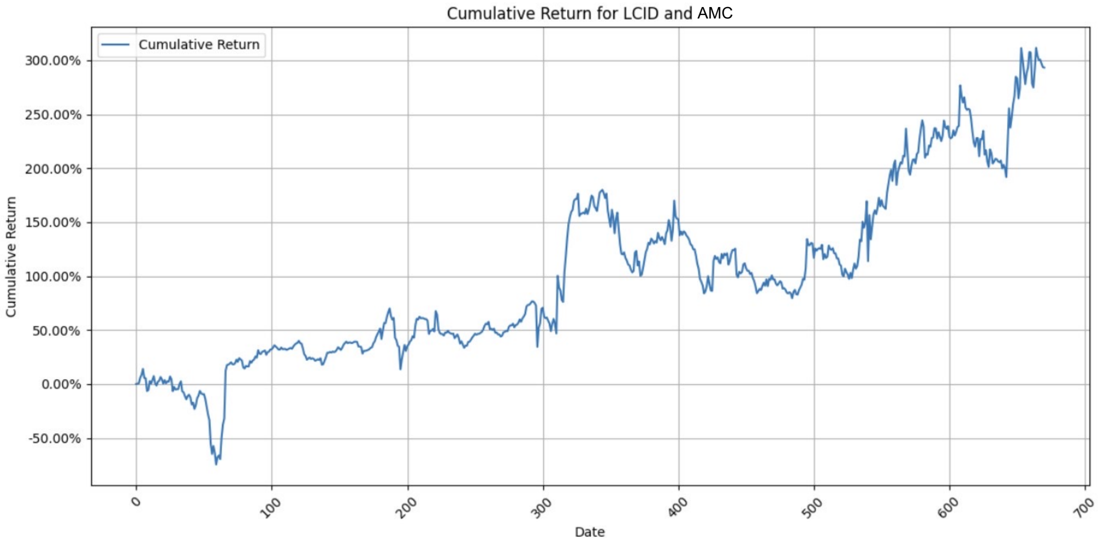
Results

- **Annualized Return:** 6.11% (reasonable returns)
- **Annualized Volatility:** 13.01% (standard volatilty)
- **Strengths**: Effective signal generation and steady capital growth.
- **Limitations**: Returns were modest compared to benchmarks like broader indices.

Results for LCID-AMC

Analysing Strategy Performance: Results and Insights.

	Capital	A	B	AmountA	AmountB	Upper	Lower	Index	Signal	Side
0	99.993000	23.046109	76.946891	1.291099	0.596488	0.983223	0.804373	0.591541	True	True
175	137.636758	16.616438	121.027320	6.235893	1.041394	0.402247	0.345539	0.443125	True	False
180	151.367571	24.411304	126.963267	4.814368	1.569484	0.525254	0.462156	0.458459	True	True
286	162.481054	6.646287	155.841767	6.979786	2.526683	0.326311	0.271202	0.328970	True	False
290	173.340200	15.231424	158.115775	7.747187	2.804482	0.326311	0.271202	0.251462	True	True
322	261.310898	12.364966	248.952932	32.445746	5.126428	0.417822	0.381480	0.424699	True	False
336	274.604568	39.294933	235.316635	40.362250	6.377235	0.417822	0.381480	0.372656	True	True
494	207.348142	-59.592581	266.947723	63.721395	3.950726	0.430543	0.338437	0.438699	True	False
503	225.114944	-46.211086	271.333030	79.143212	0.079143	0.435836	0.383774	0.380093	True	True
539	269.370259	-1.890873	271.268133	65.792171	8.750359	0.467823	0.406938	0.508930	True	False
556	293.324538	24.426002	268.905536	76.062738	13.158854	0.542477	0.489303	0.479593	True	True
568	336.651383	68.542384	268.115999	64.083304	21.403823	0.584676	0.532346	0.642206	True	False
608	376.776485	125.576516	251.206969	101.638644	24.494913	0.510514	0.471838	0.419022	True	True
653	411.297025	159.117260	252.186765	110.745913	18.383822	0.618523	0.555039	0.670356	True	False



Implementation

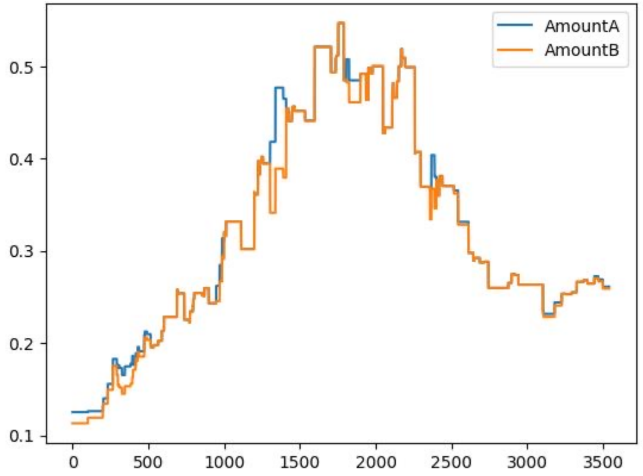
- The strategy was backtested using historical price data for LCID and AMC.
- **Timeframe:** From May 2022 to January 2025.
- **Parameters** recalculated every 20 days.
- **Source:** Historical ETF data, processed through the OU trading model.
- **Trades Executed:** 14 trades were executed during the backtesting period.

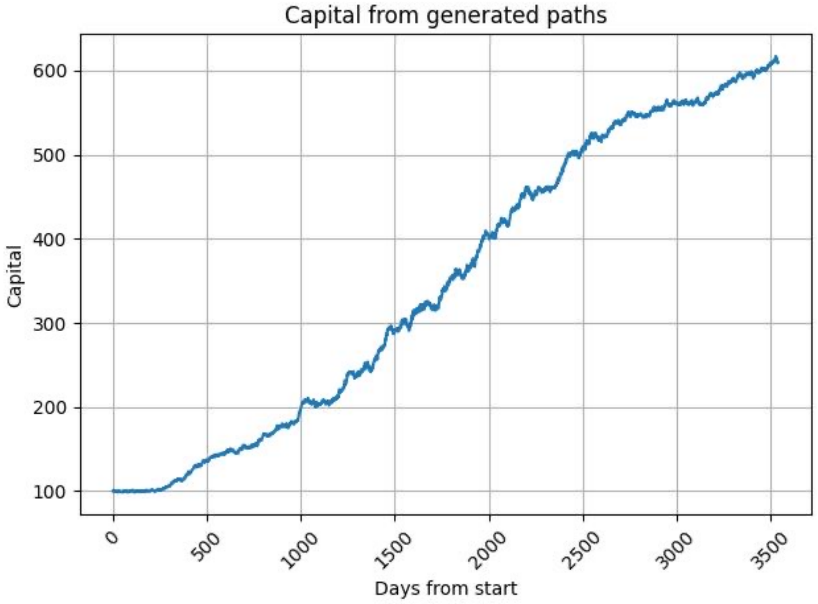
Results

- **Annualized Return:** 67.37% (exceptional returns)
- **Annualized Volatility:** 93.55% (extremely high risk)
- **Strengths**: The strategy showed strong growth, with signal generation capturing profits during volatility.
- **Limitations**: High volatility risks profitability, with returns sensitive to parameters and market conditions.

Results for Synthetic Data

Analysing Strategy Performance: Results and Insights.





Implementation

- The strategy was implemented using a generated synthetic dataset to simulate market conditions.
- The Simulated stocks are almost perfectly correlated and follow the OU process.
- **Timeframe:** The 3500 days represent approximately 14 years of trading days.
- **Parameters** recalculated every 100 days.
- **Source:** Self-created.
- **Trades Executed:** 102 trades were executed during the backtesting period.

Results

- **Annualized Return:** 13.74% (high returns)
- **Annualized Volatility:** 6,56% (high risk)
- **Strengths**: Capital grew steadily, showcasing the trading model's effectiveness in a controlled environment.
- **Limitations**: Simulated results might not reflect real-world complexities.

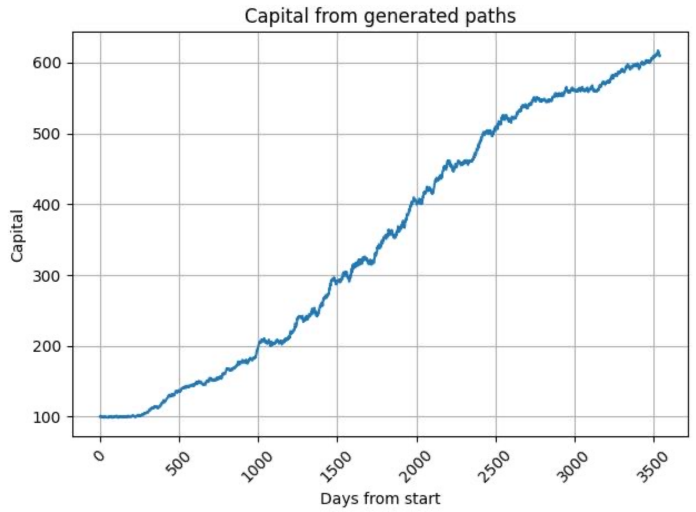
Conclusion

Key Insights and Recommendations.

Key Strengths

The Ornstein-Uhlenbeck trading model demonstrated **strong performance and steady capital growth** across various datasets, including real-world pairs (GLD-SLV, LCID-AMC) and simulated data.

- **GLD-SLV**: Generated stable returns with a 6.11% annualized return and low volatility.
- **LCID-AMC**: Delivered high returns (67.37%) despite notable volatility, highlighting strong performance in volatile environments.
- **Simulated Data**: Showed consistent capital growth, validating the model's robustness in controlled conditions.



Critical Considerations

- **Pair Selection**: Ensuring the selected pair follows the OU process is critical for success. Thorough analysis of historical data is essential.
- **Assumptions**: The model's reliance on mean-reversion limits its effectiveness to pairs exhibiting this behavior.
- **Adaptability**: Static thresholds and parameters may hinder performance in dynamic or rapidly changing market conditions.

Recommendations for Improvements

- **Pair Selection**: Focus on selecting pairs with strong historical evidence of mean-reversion to align with the OU process.
- **Parameter Adjustment:** Conduct market research about the chosen secuurities and carefully assess how often to recalculaate the parameters of the OU process.

Appendix

Sources:

- https://corporatefinanceinstitute.com/resources/career-map/sell-side/capital-markets/pairs-trading/
- https://www.investopedia.com/terms/m/meanreversion.asp#:~:text=Mean%20reversion%20is%20a%20financial,long%2Dterm%20average%20over%20time.
- https://blog.quantinsti.com/pairs-trading-basics/
- https://alpaca.markets/learn/pairs-trading
- https://cdn.shortpixel.ai/spai/q_lossy+w_949+to_webp+ret_img/algotrading101.com/learn/wp-content/uploads/2020/09/pairs-trading.png

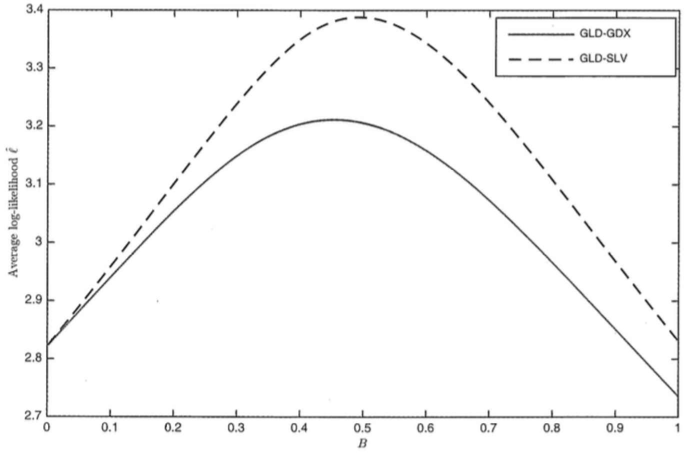
Mathematical Framework for the Strategy

Key formulas and parameter estimation.

Maximum Likelihood Estimation

- Purpose: Estimate the parameters μ, θ and σ of the OU process from observed portfolio values X_t
- Maximized Average log-likelihood defined by:

$$\begin{aligned}
&\ell(\theta, \mu, \sigma | x_0^{\alpha,\beta}, x_1^{\alpha,\beta}, \dots, x_n^{\alpha,\beta}) \\
&:= \frac{1}{n}\sum_{i=1}^{n} \ln f^{OU}\left(x_i^{\alpha,\beta} | x_{i-1}^{\alpha,\beta}; \theta, \mu, \sigma\right) \\
&= -\frac{1}{2}\ln(2\pi) - \ln(\tilde{\sigma}) - \frac{1}{2n\tilde{\sigma}^2}\sum_{i=1}^{n}[x_i^{\alpha,\beta} - x_{i-1}^{\alpha,\beta}e^{-\mu\Delta t} - \theta(1-e^{-\mu\Delta t})]^2
\end{aligned}$$



Parameter Estimation

- The Optimal parameter estimates under the OU model are given explicitly by:

$$\begin{aligned}
\theta^* &= \frac{X_y X_{xx} - X_x X_{xy}}{n(X_{xx} - X_{xy}) - (X_x^2 - X_x X_y)}, \\
\mu^* &= -\frac{1}{\Delta t}\ln\frac{X_{xy} - \theta^* X_x - \theta^* X_y + n(\theta^*)^2}{X_{xx} - 2\theta^* X_x + n(\theta^*)^2}, \\
(\sigma^*)^2 &= \frac{2\mu^*}{n(1-e^{-2\mu^*\Delta t})}(X_{yy} - 2e^{-\mu^*\Delta t}X_{xy} + e^{-2\mu^*\Delta t}X_{xx} \\
&\quad - 2\theta^*(1-e^{-\mu^*\Delta t})(X_y - e^{-\mu^*\Delta t}X_x) + n(\theta^*)^2(1-e^{-\mu^*\Delta t})^2)
\end{aligned}$$

Portfolio Optimization

- Adjust α and β to optimize the portfolio for maximum likelihood of mean-reverting behavior
- For any α, we choose the strategy (α, β*), where

$$\beta^* = \arg\max_{\beta} \hat{\ell}(\theta^*, \mu^*, \sigma^* | x_0^{\alpha,\beta}, x_1^{\alpha,\beta}, \dots, x_n^{\alpha,\beta})$$