Comity Challenge Writeup

1. How long did you spend working on the problem? What difficulties, if any, did you run into along the way?

The problem took four and half hours to solve, including understanding the problem requirements, implementing the solution, debugging, and bonus features. One difficulty was ensuring point-in-time accuracy for spread and variance calculations.

2. The slowest step in your program is most likely the loop where we run the portfolio optimization for each operating day in the backtest period. How could we speed this up to accelerate research and development velocity?

To speed up this step, parallel processing can be used to run the optimization for multiple days concurrently. Additionally, memoization or caching can be employed to avoid recalculating spreads and variances for frequently accessed dates.

3. For this challenge, we provided you with a fixed value of the risk tolerance factor. How would you go about setting the risk tolerance value if this was not provided to you?

If the risk tolerance value was not provided, it could be determined by backtesting with different values and selecting the one that maximizes the risk-adjusted return *(Sharpe Ratio)*. Alternatively, domain expertise and historical performance data could inform the appropriate risk tolerance level.

4. What happens to the PnL when you increase the maximum daily volume limit while keeping the risk tolerance factor fixed? What does this imply about how you would choose the risk tolerance parameter?

Increasing the maximum daily volume limit generally increases both potential returns and risks. The PnL could become more volatile. This implies that the risk tolerance parameter should be carefully chosen to balance the tradeoff between risk and return, considering the volume limit and desired risk profile.

5. How would you productionize your code so that the strategy could run on a daily basis and submit portfolios to the market? How can you structure the code and what other precautions can you take to ensure that the core strategy code which generates your backtest results is the same as the code that runs on production?

In order to productionize the code, it should be modularized, with clear separation between data processing, optimization, backtesting, and portfolio submission. Automated tests should be implemented to ensure code correctness. A continuous integration/continuous deployment (CI/CD) pipeline can be set up to automate code deployment. Regular monitoring and logging should be added to detect and resolve issues promptly.

Depending on the ISO, portfolio submission through virtual (IND/DEC) bids can be submitted through the programmatic API *(XML using HTTP requests in Python or C# for PJM)* and monitoring of the portfolio submissions through the Markets Gateway User Interface.

Bonus Features

1. Added Sharpe ratio as an additional metric.
2. Added charts for daily and cumulative profit and loss.
3. Added risk tolerance factor function for returning optimal risk tolerance based on highest Sharpe ratio.