I want to thank Fernando Martinelli (Balancer), Mike B (Balancer), Felix Leupold (CoW DAO), and Felix Henneke (CoW DAO), Hayden Tsutsui (Blockworks) for insightful discussions that led to the writing of this piece, and for providing comments on its initial version.

Summary

If we believe that prices do not have a systematic trend component, then eliminating LVR does more than “solving” impermanent loss because it also reduces volatility relative to holding a portfolio of assets. Hence, If you LP in an AMM that is LVR resistant (like CoW AMM), then:

- Even in the absence of fees from retail traders, your expected impermanent loss is zero (i.e., zero up to some random, unpredictable noise),
- The volatility of your portfolio is lower than that of the holding strategy,
- You earn trading fees.

Hence, LPing on an LVR-resistant AMM delivers higher expected returns and lower volatility than the equivalent holding strategy.

If we believe that prices do have a systematic trend component (which may be a reasonable assumption over long periods of time, for example, I may expect ETH to increase in value in the next 2 to 4 years), then by LPing in an AMM that is LVR resistant (like CoW AMM):

- You maximize your risk-adjusted rewards
- You earn trading fees

Preliminary

This research note connects AMM design (and associated concepts like Impermanent Loss and Loss-vs-Rebalancing) and portfolio theory. Portfolio theory is the branch of finance that studies how people should design their portfolios. At a super high level, it says that:

- People should care both about expected returns and risk.
- Holding multiple assets instead of a single asset (i.e., diversification) is valuable because it reduces risk.
- The problem with diversification is deciding how much to invest in each asset, and several theories exist on how to do so. But broadly speaking, an asset should constitute a higher share (or weight) of the portfolio if its expected future return is high, its volatility is low, and its correlation with the rest of the portfolio is also low.

For our purposes, it is important to note that if the elements that determine the weights of the different assets do not change, then the portfolio weights also do not change. However, prices change all the time. Hence, with some frequency, the portfolio should rebalance to maintain its target allocation. To say it differently, prices change constantly for no fundamental reason.* A portfolio that does not rebalance ends up overweighting/underwriting assets based purely on random price movements. Portfolio theory, instead, states that you should rebalance after each price movement to re-establish the correct weights.

Below, I call a portfolio with a target weight for each asset a rebalancing portfolio

.

Impermanent Loss (IL), Loss-Vs-Rebalancing (LVR), FM-AMM, and CoW AMM: a brief history of how we got here

For a long time, people knew that contributing liquidity to an AMM could lead to losses. This intuition was formalized in the concept of Impermanent (or divergence) Loss, defined as

IL = value_of_LP_position_if_zero_fee - value_of_holding

Where value_of_LP_position_if_zero_fee

is the value of contributing liquidity to an AMM charging zero fees, and value_of_holding

is the value of holding the same assets contributed to the AMM but outside the AMM (and never rebalancing). The return for liquidity providers (relative to holding) is determined by the fees earned by the AMM and the impermanent loss.

The key observation is that in a constant function AMM, IL is always positive. The reason is that whenever there is a

rebalancing event, the AMM trades at an outdated price with arbitrageurs. Hence, the AMM loses money whenever prices move.

Building on this intuition, Milionis, Moallemi, Roughgarden & Zhang (2024) (aka the LVR paper) propose to decompose the profitability of an LP position as:

return_from_LP_position = return_from_rebalancing_strategy - LVR + trading_fees

For a standard constant product AMM, return_from_rebalancing_strategy

is the percentage increase in value of a 50/50 rebalancing portfolio. LVR (or Loss-vs-Rebalancing) instead measures precisely how much the AMM loses to arbitrageurs whenever prices change.

The LVR paper was extremely influential because it provided a clear target for people interested in designing better AMMs. In June 2023, Robin and I released the first version of the FM-AMM paper, in which we show that you can eliminate LVR if: (i) the AMM receives trades from a batch and (ii) the AMM implements a specific pricing function. We called this new AMM design FM-AMM. Because LVR is eliminated, then LPing in an FM-AMM can be decomposed into:

return_from_LP_position = return_from_rebalancing_strategy + trading_fees

Following that paper, CoW DAO built the first LVR-resistant AMM currently in production: CoW AMM. The beta version of CoW AMM was launched in February 2024 (a description of how it worked and an evaluation of its performance can be found here) and then re-launched in August 2024 in collaboration with Balancer.

IL in Rebalancing Strategies: do CoW AMMs expose their LPs to impermanent loss?

Let us now study the return of providing liquidity to a CoW AMM in relation to both impermanent loss and portfolio theory. To do so, I make two assumptions:

1. One of the central assumptions in finance is that price movements cannot be predicted, at least in the short term. Otherwise, deep-pocketed speculators could make an infinite amount of money! Mathematically, this means that the expected future price is always equal to the current price (or, more formally, prices are martingales), an assumption I maintain throughout this section. Now, something to keep in mind (and to which we'll return later) is that this assumption may not be appropriate in the long run: we may expect the value of ETH to go up in the next 3 to 5 years. But in this case, a simple risk/reward argument implies that ETH must also be more risky than, say, T-bills.
2. We omit trading fees for simplicity. This implies that to understand whether LPs on CoW AMM suffer impermanent loss, we must establish how the rebalancing strategy relates to the holding strategy.

The introductory discussion on portfolio theory already hinted that the rebalancing strategy is better than holding. Nonetheless, it is useful to illustrate this point with a simple mathematical analysis and then with a simulation.

Mathematical analysis

Suppose Anna contributes 1 USD to a 50/50 CoW AMM pool in block t,

and then removes it in block t+2

. For simplicity, suppose that one of the assets in the CoW AMM pool is the numeraire (for example, USDC) so that we only need to keep track of the price of the other asset. Bob invests 1 USD in a 50/50 holding strategy on the same assets. He also invests in block t

and liquidates his position in block t+2

. We check block by block how the value of their positions evolved.

Block t

,

Anna and Bob have the exact same portfolio: half of their wealth is invested in one asset, and half in the other.

Block t+1

, in between block t and block t+1

prices have moved. For the sake of the argument, suppose that the price of the non-numeraire asset went up. Bob is going to hold the same allocation of assets from period 1, so he is going to have more than 50% of his wealth invested in the non-numeraire asset. Anna will instead rebalance her portfolio. Crucially, because her position is in a CoW AMM and not in a

constant product AMM, Anna does not suffer LVR: the AMM will sell the non-numeraire asset to buy the other asset to maintain a 50/50 allocation, and it will do so at the correct market price. So, Anna and Bob's portfolio will have the same value, but not the same composition.

Mathematically, call w_b>0.5

the portfolio weight of the non-numeraire asset in Bob's portfolio. Call p_{t+1}

the price of the non-numeraire asset in blockt+1

. Call V

the value of their portfolios. We have that, at the end of block t+1

Bob holds (w_b * V)/p_{t+1}

of the non-numeraire asset and (1-w_b)V

of the numeraire asset, while Anna holds 0.5 * V)/p_{t+1}

of the non-numeraire asset and 0.5V

of the numeraire asset.

Block t+2

is when things get interesting. Remember that at the end of the previous block, Anna and Bob held portfolios of the same value but different compositions: Bob has a higher proportion of his wealth invested in the asset that previously increased in price. But the price changed again between blocks t+1

and t+2

. As a result, Bob's portfolio value is now

V_b = (p_{t+2} * w_b * V)/p_{t+1} + (1-w_b)V

And Anna's portfolio value is now

V_a = (p_{t+2} * 0.5 * V)/p_{t+1} + 0.5 * V

We, therefore, have two possibilities: if p_{t+2}>p_{t+1}

(i.e., the price that previously increased in value increases again) then Bob becomes richer than Anna; else p_{t+2}<p_{t+1}

(i.e, prices invert their trajectory) and Anna becomes richer than Bob.

But let's now take a step back and evaluate the expressions V_b

and V_a

from the point of view of the previous block. Remember that the best predictor of the future price is the current price, meaning that E[p_{t+2}]=p_{t+1}.

Knowing this, we can compute

E[V_b]= E[V_a]=V

From block t viewpoint, therefore, the holding strategy and the rebalancing strategy have the same expected return E[V].

This also implies that the expected impermanent loss is zero.

How about the difference in volatility between Anna and Bob's investment strategies? Now, providing a simple mathematical intuition as to why Anna's investment strategy has lower volatility than that of Bob is not straightforward, and I'll brute-force this point from the simulation below. However, as an intuition, note that, in the holding strategy, as prices move the volatility of the portfolio may become driven by a single asset, that is, it is possible to more or less lose the benefit of diversification. Instead, the rebalancing strategy makes sure that the portfolio remains diversified.

Simulation

I wrote a Python script that does four things:

1. It simulates two prices over 1000 periods (i.e., blocks), computed as:

for t in range(1, time_periods):

price1[t] = price1[t-1] * (1 + np.random.normal(0, 0.07))

price2[t] = price2[t-1] * (1 + np.random.normal(0, 0.07))

These prices can be interpreted as the discrete version of two geometric Brownian motions (the most common stochastic process used to model price movements, which is however in continuous time and needs to be discretized in a simulation). The parameter 0.7 determines the volatility of prices. At the start, both prices are equal to 1.

1. It uses the two simulated prices to compute two investment strategies: (1) investing half in each asset and holding and (2) investing half in each asset and rebalancing each period.

2. It repeats the simulation 10000 times.

3. It uses the results of the simulation to compute the per-period standard deviation of each investment strategy. It also computes the per-period expected difference between the two strategies, together with its 68% confidence intervals if this difference (corresponding to average ± standard deviation)

[

1000×600 37.8 KB

](https://europe1.discourse-cdn.com/flex013/uploads/cow/original/2X/b/b939612ce4741355b32878f357791647483a3f56.png)

The above figure reports the standard deviation of the two investment strategies and clearly shows that the rebalancing strategy is less risky than the holding strategy.

[

1000×600 34.1 KB

](https://europe1.discourse-cdn.com/flex013/uploads/cow/original/2X/3/382502e13beeb48055891e455b109d6c35ccdccd.png)

The above figure instead shows that the difference between the expected values of the two strategies is a precisely-estimated zero (which we see by looking at the confidence intervals).

I have re-run the simulation changing the volatility of the second price, including the case in which its volatility is 0 (i.e., asset two is the numeraire). The figures look almost identical to those above and hence are omitted. Also, the interested reader can find the code below, and play with it as he/she pleases. But the results seem clear: the rebalancing portfolio has the same expected value but lower risk than holding.

Once we account for the fact that liquidity providers to a CoW AMM run a rebalancing strategy and also earn trading fees, this result implies that LPing on a CoW AMM generates higher expected returns than holding, while also reducing risk.

Trending prices

I now change our fundamental assumption about prices by considering longer time horizons, over which we may expect some prices to systematically increase (if all prices systematically increase by the same amount, then we are back to the previous case).

Let us imagine that there is a risk-free asset (i.e., cash or T-bills) and several risky but promising assets that we expect to appreciate but are also subject to risk. Now, if all we care about is expected returns, the optimal thing is to invest 100% of our wealth in the risky but promising asset with the highest expected return. If instead we also care about risk, then portfolio theory (in particular the one-fund theorem) tells us that to maximize the risk-adjusted return (i.e., the expected return of the portfolio divided by its volatility) we should:

1. Build the optimal fund using the risky assets: in the simple case in which the various risky assets are uncorrelated with each other, each risky asset is included in the portfolio with a weight proportional to its risk-adjusted return. If these assets are correlated, then constructing these weights is more complicated. But in the end, they depend on expected return, correlation, and volatility. As long as these quantities do not change, the weights also do not change. The optimal fund can, therefore, be implemented in a multi-token CoW AMM with appropriate weights.

2. Decide how much to invest in the optimal fund vs the risk-free asset. This decision is driven by the investor's risk appetite.

To summarize, an appropriately weighted CoW AMM maximizes risk-adjusted returns. Holding the two assets outside the AMM is always dominated by either holding them in a CoW AMM (for someone who cares about risk-adjusted return) or investing everything in the assets with the highest returns (for someone who only cares about expected returns).

Conclusions

Portfolio theory is one of the cornerstones of finance theory. Its significance is highlighted by the popularity of simple, passive investment strategies that track the S&P 500 or similar indexes, which, by some accounts, constitute more than 30% of the entire stock market. Note also that these strategies have outperformed more complex investment strategies run by sophisticated hedge funds, see the famous Warren Buffet bet.

We discussed how LPing in a CoW AMM is equivalent to running a rebalancing strategy, plus earning extra yields whenever the liquidity in the AMM is used to fulfill a user order. For now, this is limited to simple, fixed-weight rebalancing strategies, but in the future, more complex passive investment strategies will be possible. For example, a multi-token CoW AMM could have a “manager” who can change its weights and add and remove assets.

The end game is, therefore, a market mechanism allowing retail investors who want to run simple passive investment strategies to do so by contributing liquidity to AMMs, while earning additional returns. If traditional finance is an indication, these AMMs could attract up to ⅓ of the total market cap of various assets as liquidity, providing super-efficient trading at close to zero slippage. Decentralized finance will finally provide better market mechanisms than traditional finance, opening the door to rebuilding the financial system on blockchain rails.

Notes:

*Note that some price movements may have information content and should lead to a change in the portfolio weights (and there are some theories that account for that). The point is that not all price movements do.

The code of the simulation:

import numpy as np import matplotlib.pyplot as plt

Simulation parameters

num_simulations = 10000 time_periods = 1000 initial_price = 1

Array to store the values for all simulations

price1_all = np.zeros((num_simulations, time_periods)) price2_all = np.zeros((num_simulations, time_periods)) no_rebalance_all = np.zeros((num_simulations, time_periods)) rebalance_all = np.zeros((num_simulations, time_periods))

Run simulations

for sim in range(num_simulations): # Generate two random price series as discrete version of a geometric Brownian motion with zero drift price1 = np.zeros(time_periods) price2 = np.zeros(time_periods) price1[0] = initial_price price2[0] = initial_price

```
for t in range(1, time_periods):
    price1[t] = price1[t-1] * (1 + np.random.normal(0, 0.07))
    price2[t] = price2[t-1] * (1 + np.random.normal(0, 0.00))

# Calculate no-rebalance portfolio and store its value
no_rebalance_portfolio = 0.5 * price1 + 0.5 * price2
no_rebalance_all[sim, :]=no_rebalance_portfolio

 # Calculate rebalanced portfolio
rebalance_portfolio = np.zeros(time_periods)
rebalance_portfolio[0] = 1
holdings1 = 0.5
holdings2 = 0.5
for t in range(1, time_periods):
    value1 = holdings1 * price1[t]
    value2 = holdings2 * price2[t]
    total_value = value1 + value2
    holdings1 = (total_value / 2) / price1[t]
    holdings2 = (total_value / 2) / price2[t]
    rebalance_portfolio[t] = total_value
rebalance_all[sim, :]=rebalance_portfolio

#store prices of each simulation
price1_all[sim, :]=price1
price2_all[sim, :]=price2
```

compute the relative hedge of the rebalance strategy relative to the no-rebalance strategy

relative_all = rebalance_all - no_rebalance_all

Compute per-period standard deviation across all simulations

std_dev_relative = np.std(relative_all, axis=0) std_dev_no_rebalance = np.std(no_rebalance_all, axis=0) std_dev_rebalance = np.std(rebalance_all, axis=0) std_dev_price1 = np.std(price1, axis=0) std_dev_price2 = np.std(price2, axis=0)

Compute per-period average deviation across all simulations

avg_relative = np.mean(relative_all, axis=0) avg_no_rebalance = np.mean(no_rebalance_all, axis=0) avg_rebalance = np.mean(rebalance_all, axis=0) avg_price1 = np.mean(price1, axis=0) avg_price2 = np.mean(price2, axis=0)

confidence intervals

u_ci= avg_relative + std_dev_relative l_ci= avg_relative - std_dev_relative

Plot the volatility of the different strategies

plt.figure(figsize=(10,6)) plt.plot(std_dev_no_rebalance, label='No Rebalance Portfolio', color='orange') plt.plot(std_dev_rebalance, label='Rebalanced Portfolio', color='green') plt.title('Volatility of the different investment strategies (10000 Simulations)') plt.xlabel('Time') plt.ylabel('Standard Deviations') plt.legend() plt.show()

Plot the expected difference in the return of the rebalancing and no rebalancing strategy, with confidence intervals

plt.figure(figsize=(10,6)) plt.plot(u_ci, linestyle='--') plt.plot(l_ci, linestyle='--') plt.plot(avg_relative, color='orange') plt.title('Rebalancing strategy - No Rebalancing strategy, with 68% confidence intervals (10000 Simulations)') plt.xlabel('Time') plt.legend() plt.show()