Exploring the Autonomous Converter Contract: Risk-Free Arbitrage in Metronome 1.0

Luca Nicoli

January 2025

Introduction

The purpose of this report is to analyze the *Ethereum-based Autonomous Converter Contract (ACC)*[1] that is a key component of the Metronome 1.0 ecosystem. This contract facilitates the conversion between the Metronome token (MET) and Ether (ETH) at market-driven rates, providing a liquidity mechanism and stabilizing prices by adjusting exchange rates based on the supply of MET and ETH. The Metronome 1.0 ecosystem introduces an innovative approach to cryptocurrency that emphasizes longevity, self-governance, and cross-blockchain portability. The ecosystem operates through a set of autonomous smart contracts that oversee various functions, such as token issuance, governance, and transaction execution. In particular, the ACC acts as a fallback marketplace to ensure continuous liquidity and mitigate slippage in the MET-ETH exchange.
This report explores the purpose of the ACC within the broader Metronome protocol, explains the mechanics of the system, and examines whether there exists a risk-free arbitrage opportunity. Additionally, we will discuss how such an opportunity could be implemented in practice, including mathematical formulations and optimal strategies to outperform competitors in exploiting these arbitrage chances.

Overview of Metronome 1.0

Metronome (MET) is a cryptocurrency designed for longevity, self-governance, and cross-blockchain portability. It operates via autonomous smart contracts that handle all token issuance, governance, and transactions. Its token supply begins with an initial auction distributing 10 million MET, of which 20% is retained by founders under a gradual release schedule, and 80% is sold through a Descending Price Auction. Subsequent Daily Price Auctions (DPA) mint new tokens at the greater of 2,880 MET per day or 2% of the existing supply annually. Proceeds from all auctions are directed to a Proceeds Contract, which funds an Autonomous Converter Contract (ACC) enabling MET-ETH conversions at market-driven rates, mitigating slippage via a dynamic supply balance. Portability between blockchains is enabled through export-import mechanisms that burn tokens on the source chain and mint them on the target, with validators ensuring supply integrity and security. This structure ensures decentralization, predictable token economics, and resilience against governance disputes, aiming to create a stable, enduring currency ecosystem.

Flow of the Metronome Protocol

Metronome operates through four autonomous smart contracts, each with a distinct role in ensuring the system's decentralization, predictability, and functionality:

1. Token Contract

Purpose: Implements the MET cryptocurrency as an ERC-20 token with additional functionalities for enhanced security and cross-blockchain portability.
Functions:

- Handles token balances, transfers, and approvals.
- Supports batch transfers (`multiTransfer`) and enables custom porters for blockchain migration.
- Manages minting and burning of tokens during auctions and blockchain export/import processes.

2. Auctions Contract

Purpose: Manages token distribution through descending price auctions (DPAs).
Functions:

[1] https://etherscan.io/address/0x686e5ac50d9236a9b7406791256e47feddb26aba

- Executes the *Initial Supply Auction*, distributing 80% of the initial MET supply.
- Conducts *Daily Supply Lots*, minting new tokens at a predictable rate (2,880 MET/day or 2% of the existing supply annually, whichever is greater).
- Sets auction parameters such as start time, initial price, price decrement rate, and floor price.
- Sends 100% of auction proceeds to the Proceeds Contract.

3. Proceeds Contract

Purpose: Stores auction proceeds and ensures liquidity for MET-ETH conversions.
Functions:

- Retains ETH collected from auctions, ensuring all proceeds remain decentralized and on-chain.
- Transfers 0.25% of its total balance daily to the Autonomous Converter Contract.
- Provides a buffer to smooth out fluctuations in daily auction proceeds.

4. Autonomous Converter Contract

Purpose: Facilitates MET-to-ETH and ETH-to-MET conversions at market-driven rates using dynamic supply balancing.
Functions:

- Maintains MET and ETH reserves to ensure continuous liquidity.
- Adjusts prices automatically based on reserve balances.
- Serves as a fallback marketplace for MET outside of auctions.
- Enables arbitrage opportunities to stabilize MET prices relative to ETH.

Arbitrage Opportunity

Arbitrage in Metronome arises from the price differential between the daily descending price auction (DPA) and the Autonomous Converter Contract (ACC). Specifically:

- If the price of MET in the DPA is lower than the equivalent price in the ACC (as determined by the MET/ETH ratio in the ACC), an arbitrage opportunity exists.
- You can buy MET in the DPA at a lower cost, then sell it in the ACC at a higher price to obtain ETH, profiting from the price difference.

This mechanism relies on the predictable and transparent pricing structure of the auction and the dynamic market-driven rates in the ACC. Arbitrageurs play a crucial role in stabilizing prices across these platforms, as their activity balances supply and demand discrepancies. To understand how to catch the arbitrage opportunity, we first need to define what we mean by arbitrage. Let us assume that the initial amount of ETH is e_0 and MET m_0 and that after certain operations, we end up with our final amounts e_1 and m_1. For simplicity, we assume $m_0 = m_1 = 0$. Then we want:

$$e_1 > e_0 \tag{1}$$

Meaning that under the condition of zero MET asset variation, we obtained a net profit $e_1 - e_0 > 0$. If we define the *Price function* $\mathbb{P}^x$ the amount of token y received for a unit of token x, then assuming that we are selling ETH (e_d) for MET (m_d) from the DPA at price $\mathbb{P}_D^{met}$ and selling all these METs to the ACC for ETH e_a at price $\mathbb{P}_A^{eth}$, ignoring all transaction fees we have that:

$$m_d = \mathbb{P}_D^{met} e_d \tag{2}$$

$$e_a = \mathbb{P}_A^{eth} m_d \tag{3}$$

Working the equations out, we have that:

$$\frac{e_a}{e_d} = \mathbb{P}_A^{eth} \mathbb{P}_D^{met} \tag{4}$$

And since we require $\frac{e_a}{e_d} > 1$, then we need that

$$\mathbb{P}_A^{eth} > \frac{1}{\mathbb{P}_D^{met}} = \mathbb{P}_D^{eth} \tag{5}$$

When this condition is met, then there is an arbitrage opportunity.
Let us now get a closer look at the price functions (See Appendix A for the derivation of the price function of the ACC):

$$\mathbb{P}_D^{eth}(t) = \mathbb{P}_{D,0}^{eth} \cdot (0.99)^{\frac{t}{60}} \tag{6}$$

$$\mathbb{P}_A^{eth}(m_d) = \frac{E_{A,0}}{M_{A,0} + 2m_d} \tag{7}$$

Where $\mathbb{P}_{D,0}^{eth}$ is the initial ETH price in the Daily auction, $M_{A,0}$, and $E_{A,0}$ are the initial MET and ETH amounts in the ACC respectively, and m_d is the amount of MET sold to the ACC. The initial ETH price of the daily auction is twice the previous auction closing price. If zero MET is sold in the previous DPA, the price of the following day's DPA will begin at 1/100th of the last price at which MET was purchased.In Fig. 1 we report the two price functions (Eq. 6 – left, and Eq. 7 – right).

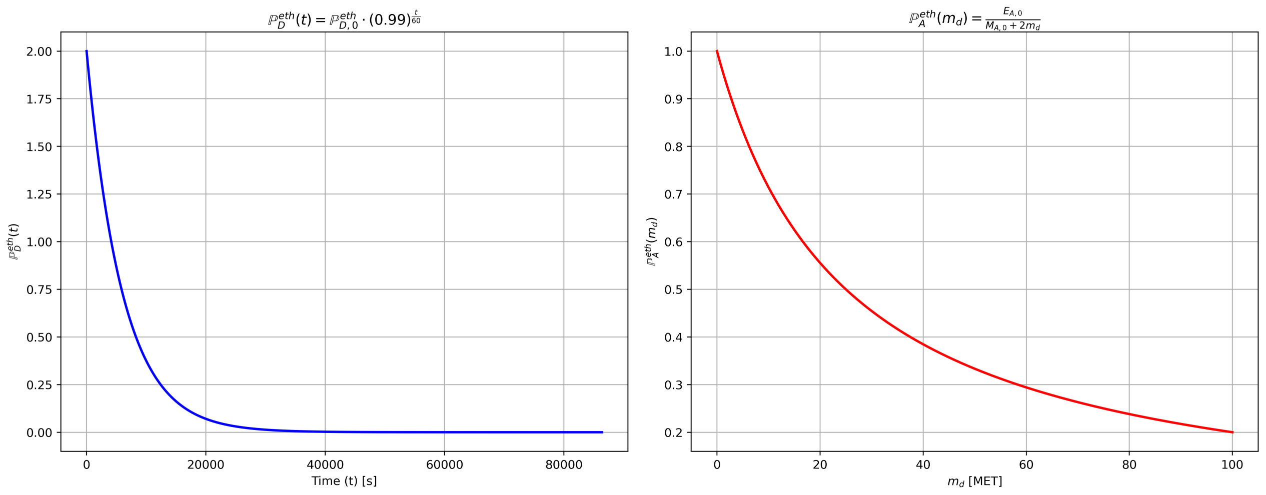


Figure 1: (left) DPA $\mathbb{P}_D^{eth}(t)$ price as a function of time throughout the day. (right) ACC price $\mathbb{P}_A^{eth}(m_d)$ as a function of the MET bought at the DAC and sold to the pool m_d.

To leverage the arbitrage, we can substitute $\mathbb{P}_D^{eth}$, and $\mathbb{P}_A^{eth}$ into Eq. 5:

$$\frac{E_{A,0}}{M_{A,0} + 2m_d} > \mathbb{P}_{D,0}^{eth} \cdot (0.99)^{\frac{t}{60}} \tag{8}$$

This leads to the following arbitrage condition for m_d:

$$\frac{1}{2}\left[\frac{E_{A,0}}{\mathbb{P}_{D,0}^{eth}}(0.99)^{\frac{-t}{60}} - M_{A,0}\right] - \epsilon \geq m_d \geq \epsilon \tag{9}$$

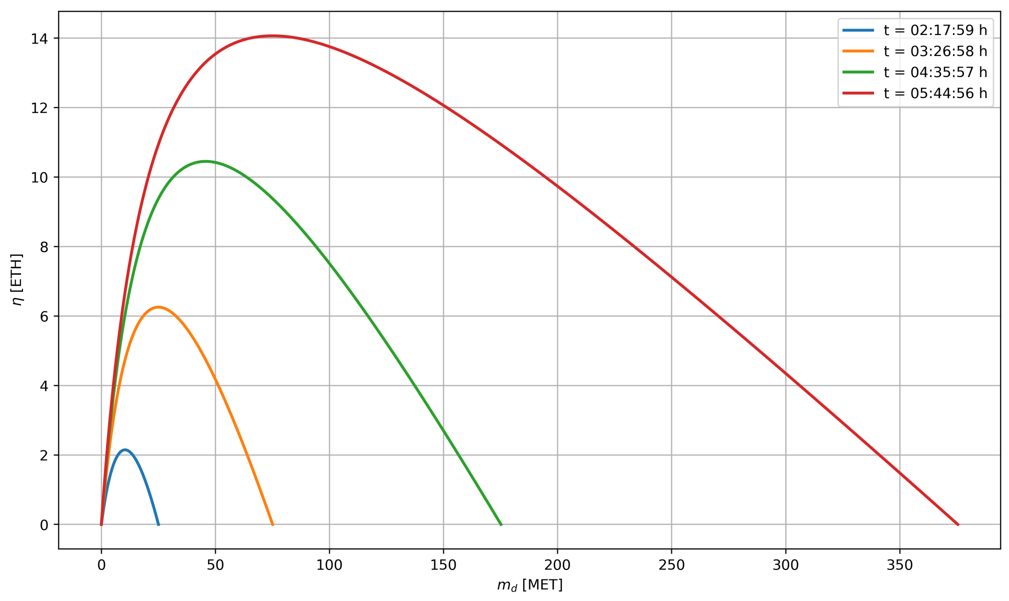


Figure 2: Profit η as a function of the amount of the amount MET (m_d) bought during the DPA for different times $t \in [t^*, T]$. For this example, we employed $M_{A,0} = 50$, $D_{A,0} = 50$, $P^{eth}_{D,0} = 2$, and therefore the computed $t^* \approx$ 01:08:59 hours for which conditions in Eq. 10 hold.

Where $\epsilon = 10^{-18}$ is the minimum possible amount of MET considerable. While $M_{A,0}+2\epsilon$ is a constant, the other term on the left-hand side grows in time. Thus, there might exist a minimum time for which the condition expressed by Eq. 9 holds, and thus, we can make arbitrage.
More formally:

$$\begin{aligned} &\text{if } \exists\ t^* \in [0,T] \text{ s.t. } \frac{E_{A,0}}{\mathbb{P}^{eth}_{D,0}}(0.99)^{\frac{-t^*}{60}} = M_{A,0} + 2\epsilon, \text{ then} \\ &\forall t \in [t^*, T]\ \exists\ m^*_d \geq \epsilon, \text{ s.t. } \forall m_d \in [\epsilon, m^*_d], \text{ then } \frac{e_a}{e_d} > 1 \text{ holds} \end{aligned} \tag{10}$$

Where T is the DPA duration, i.e. 24 hours. The condition expressed in the first line of Eq. 10 can be easily checked at the beginning of each day, i.e. when we know the values of:

1. $M_{A,0}$, and $E_{A,0}$. The latter will increase at the beginning of each day due to the injection in the ACC of 0.25% of the total accumulated balance of the Proceeds Contract.
2. $P^{eth}_{D,0}$ the DPA's starting price.

If the condition is met at time t^* then for each subsequent time we can perform arbitrage. Nevertheless, we are still facing another problem. Indeed, when we make arbitrage we'd like to maximize our profit ($\eta = [e_a - e_d]$).

$$\max_{m_d \in [\epsilon, m^*_d]} \eta = \max_{m_d \in [\epsilon, m^*_d]} \left[\mathbb{P}^{eth}_A m_d - \mathbb{P}^{eth}_D m_d\right] = \max_{m_d \in [\epsilon, m^*_d]} \left[\frac{E_{A,0} m_d}{M_{A,0} + 2m_d} - m_d P_{D,0}(0.99)^{\frac{t}{60}}\right] \tag{11}$$

In Fig. 2 we report the shape of the profit η as a function of the amount of MET m_d bought during the DPA for different times $t \in [t^*, T]$. As it can be noticed, the profit function η has a maximum in the window $m_d \in [\epsilon, m^*_d]$, and this maximum grows in time. In Fig. 3 is depicted the maximum of the profit ($\max \eta$) and its derivative as a function of the time $t \in [t^*, T]$.
It is interesting to notice that the maximum profit converges to a maximum value by the end of the

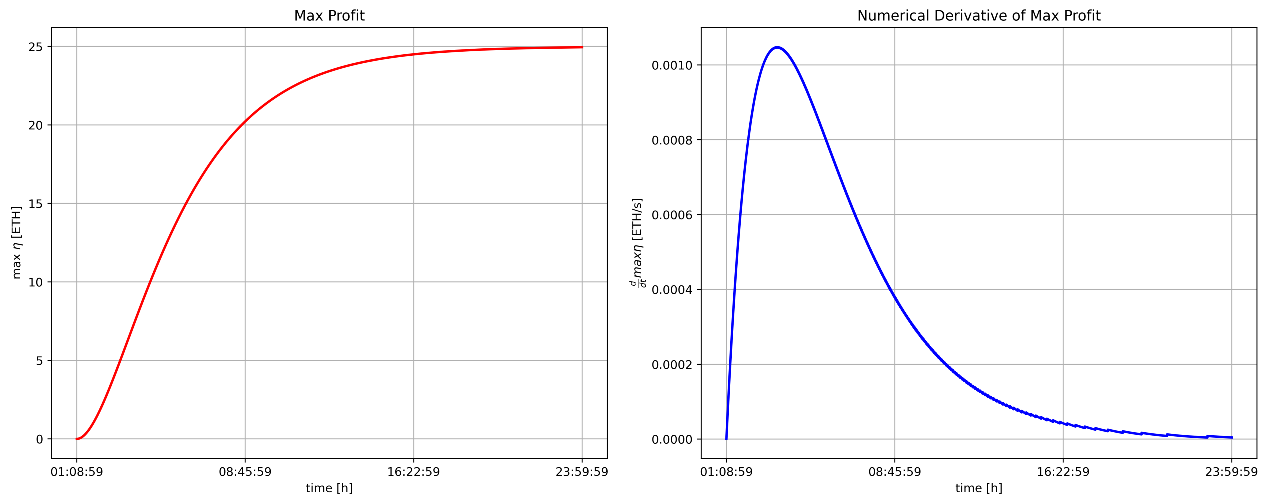


Figure 3: Maximum of the profit function (left – $\max_{m_d} \eta$) and its time derivative (right – $\frac{d}{dt} \max_{m_d} \eta$) of as a function of the time $t \in [t^*, T]$. For this example, we employed $M_{A,0} = 50$, $D_{A,0} = 50$, $P^{eth}_{D,0} = 2$, and therefore the computed $t^* \approx 01:08:59$ hours for which conditions in Eq. 10 hold.

day. However, such a high-profit condition is unlikely to happen since we are in a strongly adversarial environment, and plenty of competitors will try to leverage the arbitrage opportunity as we do. Most probably, the arbitrage opportunity is attacked before the maximum profit has the inflection (the maximum of its derivative) since, after that point, waiting time will reduce the increase in profit per unit time whilst the probability of an adversarial trader leveraging the arbitrage opportunity increases. In addition to this, we also need to consider that the reservoirs in the ACC might change during the day due to others trading (and thus, we need to keep our conditions updated), but also we need to consider that to leverage higher profits we need to buy more MET (as it can be seen from Fig. 2). However, only a limited amount of MET can be acquired during the day. In particular, the amount of tokens in each DPA is the greatest between (i) 2,880 MET per day or (ii) an annual rate equal to 2.0000% of the then-outstanding supply per year.

Therefore, under the conditions that the ACC liquidity pool has not changed its reservoirs and that there are still enough MET to be bought in the DPA such that we can leverage our maximum profit, the more we wait after the time t^*, the higher our profit but the lower our chance to catch it. Techniques like real-time data monitoring, game-theoretic models, and adaptive risk management could be employed to optimize the timing and execution of arbitrage trades. To develop an effective quantitative strategy, we should (i) model the likelihood of other users purchasing MET during the DPA and (ii) model the probability that these users will attempt to leverage the arbitrage opportunity. With these quantities or reasonable approximations thereof, we can identify the "sweet spot" where we can maximize our profit while also increasing the likelihood of successfully executing the arbitrage trade before competitors, ensuring that there is still enough MET available in the DPA to make this move. Achieving this balance is a complex task, as it requires continuously adapting to dynamic market conditions and the actions of other traders.

However, I feel that all these considerations point towards the idea that *"the quicker, the better"* to maximize our chance of effectively leveraging this arbitrage opportunity. One could keep on making small profit trades right after the time t^* at high frequency:

1. Given the initial reserves in the ACC compute t^* for which $\max[\eta(t^*)] > 0$ and the corresponding t_{max} such that the profit $\max_{m_d}[\eta(t_{max})] > c \cdot (1 + f)$, where $c > 0$ is a parameter that accounts for the amount of profit more than the transaction fee (f) we are willing to get;

2. At the time t_{max} we perform the transactions exchanging the computed m_d, thus perturbing the reserves in the ACC.

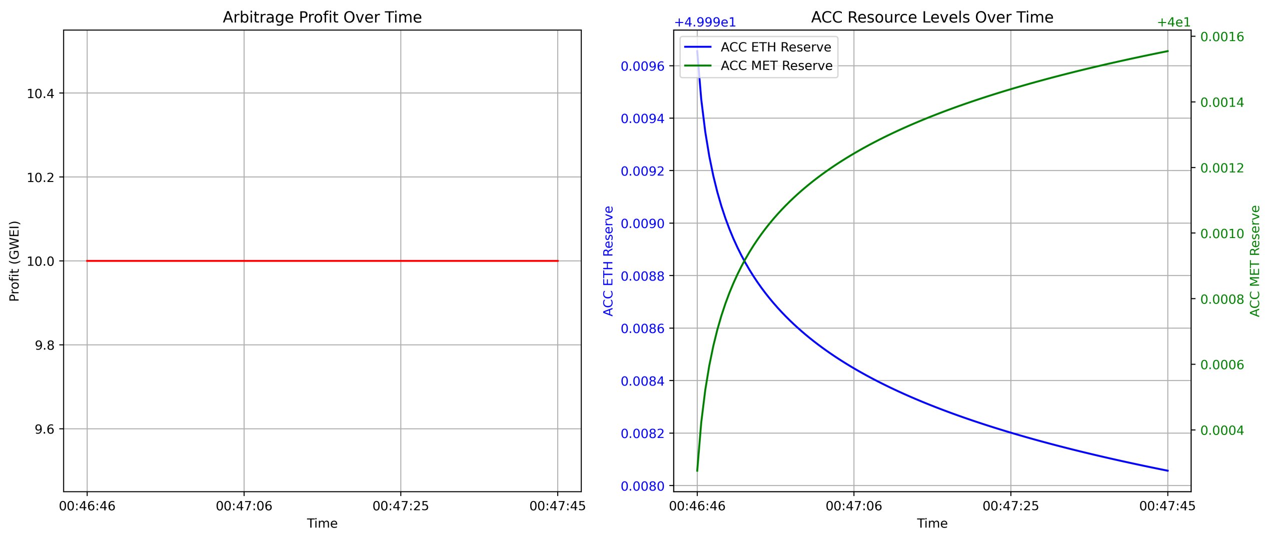


Figure 4: (left) Net profit of the strategy in GWEI as a function of time. (right) ACC reserves in time.

3. Recompute the new t^* for which $\max[\eta(t^*)] > 0$ and t_{max} given the updated reserves in the ACC.

These trades should be packed in an atomic bundle so we can effectively perform the arbitrage if all the trades can be executed. We should keep an eye on the mempool to see if other MEV searchers are trying to leverage the arbitrage opportunity, although, with a high probability, they might also use private atomic bundles, sending them directly to the block-builders and thus skipping the public mempool. In Fig. 4, I am plotting the results of such a simulated strategy over a time window of 1 min after the first t^*. I have set $c = 50\%$, and 50 ETH and 40 MET as the initial reserves in the ACC. The timestep to find the optimal t_{max} is set to 0.25 s, although it's not optimal (probably if it would be possible, it would be better to perform trades at higher frequency). The fee is set to $f = 20$ GWEI. Nevertheless, we can still see that the net profit, i.e., $\eta - f$ [GWEI], is constant. The code implemented to simulate this strategy is simple and should be optimized and numerically stabilized. In particular, the time at which we perform the trades is set to be precisely the time for which the net profit is $\max_{m_d} \eta(t^*) = c \cdot (1 + f)$. This might be challenging due to numerical precision, and thus, in order to avoid weird fluctuations in the profit, I had to hardcode its value and smooth the curve out. You can find more details about this in appendix B. However, being this a proof of concept, I believe this can already give a hint of the results of the strategy.

Simultaneously, we might also pursue a more aggressive approach. We should keep on performing our atomic trades each time while looking to the mempool for possible trades with the ACC. These trades can be sandwiched to reach two different goals: (i) earn ETH from the sandwich attack and (ii) apply a veto to large interactions with the ACC. Indeed, if we are capable of spotting a transaction with the ACC that would not perturb the ETH/MET distribution too much, i.e. we can keep on with our DPA-ACC arbitrage, then we could still profit from this transaction by sandwiching it with small transactions. Most users might have set a slippage tolerance limit on their transaction Tx. Thus, if we sandwich Tx with small transactions, then there is a high chance of Tx landing and us profiting. If on the other hand, we spot transactions with the ACC that aim at balancing out the ETH/MET spread in the ACC and thus reducing our time-window for the arbitrage $[t^*, T]$, then it might be profitable to perform a sandwich attack with large transactions, thus inducing a large slippage and most likely resulting in the abortion of the user's transaction. Of course, in this case, we need to be very careful since the cost might outweigh the benefits. All these considerations are valid, considering the price we would need to pay to create a private atomic transaction bundle and the price needed to have our bundle accepted by block-builders.

A Derivations of the ACC price formula

As discussed in the Metronome's *Owner Manual*[2] we have that:

$$t = s_0 \left[\sqrt{1 + \frac{e}{e_0 + e}} - 1 \right] \tag{12}$$

$$r = r_0 \left[1 - \left(1 - \frac{t}{s_0 + t} \right)^2 \right] \tag{13}$$

Where s_0 is the Smart Token Supply, e is the reserve tokens received in exchange for smart tokens t, e_o r_0 are the initial reserve token supplies.

$$\begin{aligned}
y(x) &= y_0 \left[1 - \left(1 - \frac{s_0 \left(\sqrt{1 + \frac{x}{x_0+x}} - 1 \right)}{s_0 + s_0 \left(\sqrt{1 + \frac{x}{x_0+x}} - 1 \right)} \right)^2 \right] \\
&= y_0 \left[1 - \left(1 - \frac{\left(\sqrt{1 + \frac{x}{x_0+x}} - 1 \right)}{\sqrt{1 + \frac{x}{x_0+x}}} \right)^2 \right] \\
&= y_0 \left[1 - \left(\frac{1}{\sqrt{1 + \frac{x}{x_0+x}}} \right)^2 \right] \\
&= y_0 \left[1 - \frac{1}{1 + \frac{x}{x_0+x}} \right] \\
&= y_0 \left(\frac{\frac{x}{x_0+x}}{1 + \frac{x}{x_0+x}} \right) \\
&= y_0 \left(\frac{x}{x_0 + 2x} \right)
\end{aligned} \tag{14}$$

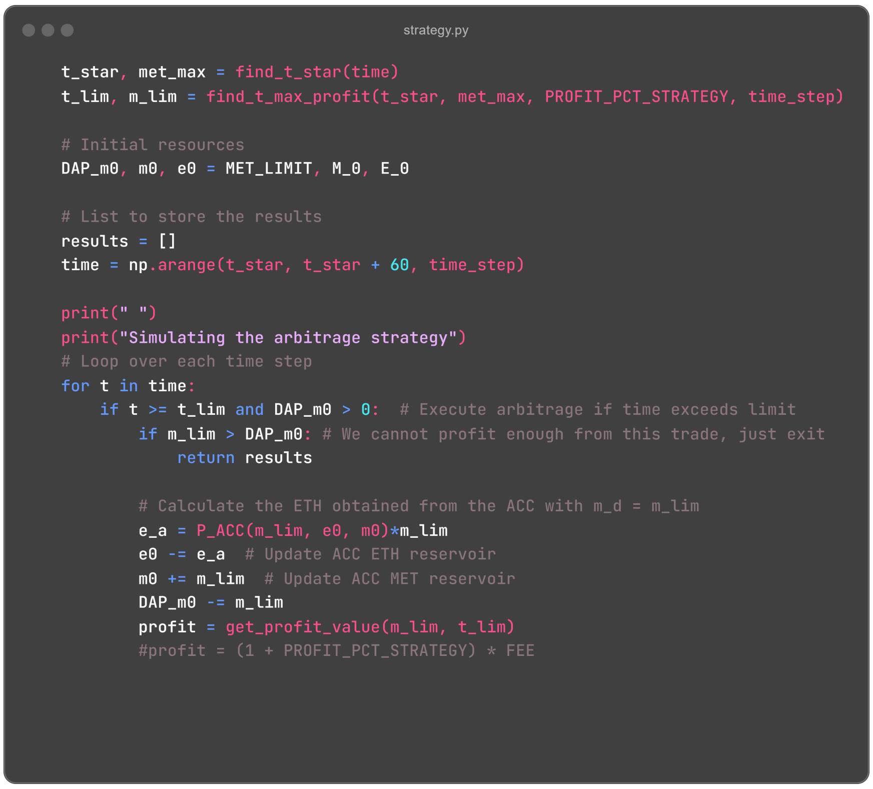
[2] https://github.com/autonomoussoftware/documentation/blob/master/owners_manual/owners_manual.md

B Numerical issues with numerical simulation of the arbitrage strategy

As stated in the main text, I am trying to perform this strategy to overcome the other adversarial MEV players.

1. Given the initial reserves in the ACC compute t^* and the corresponding t_{lim} such that the profit $\max_{m_d} \eta(t_{lim}) > c \cdot (1+f)$, where $c > 0$ is a variable that accounts for the amount of profit more than the transaction fee (f) we are willing to get;
2. At the time t_{lim} we perform the transactions exchanging the computed m_{lim}, thus perturbing the reserves in the ACC.
3. Iterate with the new reserves in the ACC

A pseudo-snippet of the code implemented to do this is in Fig. 5, where it is visible the original function used to compute the profit at the trade time t_{lim} associated with a purchased m_{lim} METs from the DAC. This t_{lim} was chose to be such that $\max_{m_d} \eta(t_{lim}) = c \cdot (1+f)$. However, due to numerical accuracy, we often couldn't match the equality conditions, finding $\max_{m_d} \eta(t_{lim}) > c \cdot (1+f)$, and resulting in fluctuations in the computed net profit (see Fig. 6). To solve this problem I hardcoded the expected value of the profit (the commented line in the snippet).



strategy.py

```python
t_star, met_max = find_t_star(time)
t_lim, m_lim = find_t_max_profit(t_star, met_max, PROFIT_PCT_STRATEGY, time_step)

# Initial resources
DAP_m0, m0, e0 = MET_LIMIT, M_0, E_0

# List to store the results
results = []
time = np.arange(t_star, t_star + 60, time_step)

print(" ")
print("Simulating the arbitrage strategy")
# Loop over each time step
for t in time:
    if t >= t_lim and DAP_m0 > 0:  # Execute arbitrage if time exceeds limit
        if m_lim > DAP_m0: # We cannot profit enough from this trade, just exit
            return results

        # Calculate the ETH obtained from the ACC with m_d = m_lim
        e_a = P_ACC(m_lim, e0, m0)*m_lim
        e0 -= e_a  # Update ACC ETH reservoir
        m0 += m_lim  # Update ACC MET reservoir
        DAP_m0 -= m_lim
        profit = get_profit_value(m_lim, t_lim)
        #profit = (1 + PROFIT_PCT_STRATEGY) * FEE
```

Figure 5: Pseudo-snippet of the code implementing my strategy

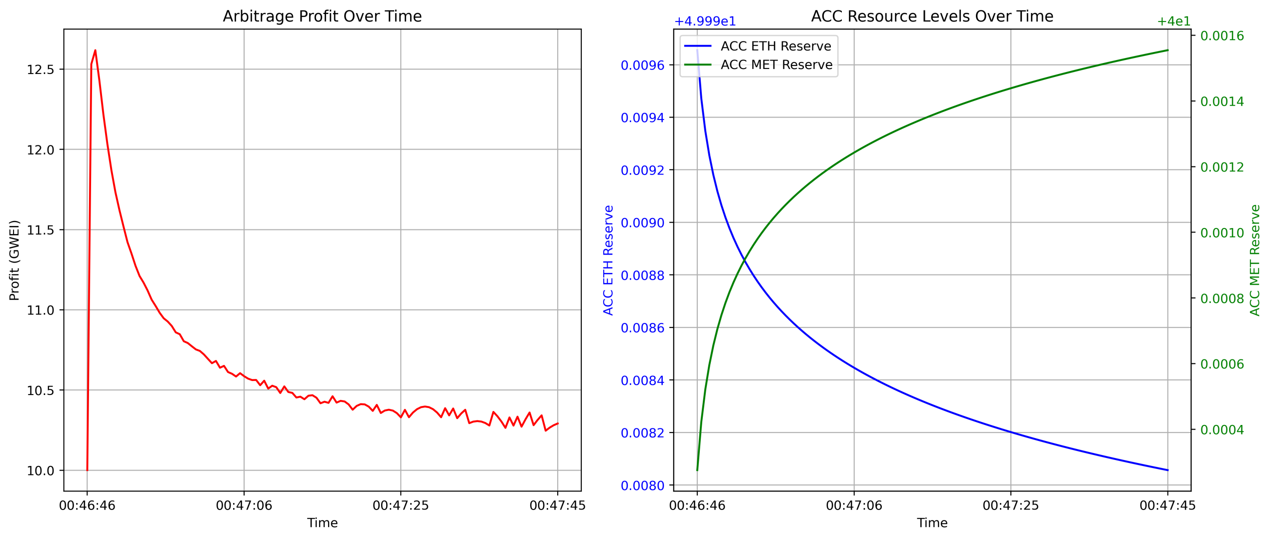


Figure 6: (left) Net profit of the strategy in GWEI as a function of time. (right) ACC reserves in time.