This is a joint work by Finn Casey-Fierro (Nethermind Research), Conor McMenamin (Nethermind Research), Lorenzo Feroleto (Chainbound) & Fra Mosterts (Chainbound). Many thanks to Lin Oshitani for his detailed review.

This is the final article in a series (part 1 and part 2 here) on the economic viability of preconfirmations. This series has received funding from a LEGO grant.

Introduction

This article focuses on the pricing of inclusion preconfirmations (preconfs); credible promises within a blockchain protocol that guarantee a transaction will be included in a block. We introduce a pricing methodology for inclusion preconf tips, that can apply to inclusion preconf protocols like Chainbound's Bolt V1, ETHGas, Luban's Taiyi, and Manifold Finance's XGA.

In a previous article in the series, we identified that the revenue from offering inclusion preconfs throughout the slot is greater than or equal to the expected revenue from not offering inclusion preconfs. Building upon that result, this article builds a model for estimating the compensation required by a proposer to provide an inclusion preconf.

Over a sample of historical Ethereum block data, we fit a logarithmic regression modelling blocks' cumulative proposer rewards against cumulative gas within a block to discover a model for the expected cumulative proposer rewards for any gas usage. With this model, we are able to calculate the marginal value of each part of the block for the proposer. In this model, we argue that the lowest value block-space approximates the value that should be required by a proposer to provide an inclusion preconf. The intuition here is that a proposer gives up this lowest value block-space when providing an inclusion preconf, although their is nuance to this statement, which we discuss when introducing the model.

Summarised in the graph below, per-gas inclusion preconf tips scale up with two factors: already preconfirmed gas amount and transaction size. A typical 21k gas ETH transfer sees inclusion preconf tips from 0.61 GWEI (0 already preconfirmed gas) to 1.17 GWEI (15M already preconfirmed gas), and beyond. With this model in hand, proposers can begin to reasonably determine the price of an incoming inclusion preconf.

Beyond this, we also outline a series of practical considerations that should be incorporated by any proposer intending to put our theoretical model to practical use.

[

image

1680×1102 89.2 KB

](https://europe1.discourse-cdn.com/flex013/uploads/lido/original/2X/f/f403a6de51ff19665dfd8ed1fcea69aad21a0898.jpeg)

Pricing curve for a preconfirmed inclusion tip, taking into consideration all possible amounts of pre-existing preconfirmed gas.

Organisation of the article

We start by providing the necessary context and definitions to reason about pricing of inclusion preconfirmations. We describe the data used to create our pricing model, as well as the assumptions used when creating our model. We then outline a method for defining proposer rewards, calculating the expected cumulative proposer rewards function within a block, before creating a predictive model for cumulative proposer rewards. This is followed by section on practical considerations for pricing using our outlined model. This section ends with a brief Case-Study from Chainbound where we put our theoretical model in the context of a company deploying an inclusion preconf protocol. We finish with a brief conclusion.

Preliminaries

This section introduces the terminology, data and assumptions that are used throughout the article.

Ethereum Block Construction

This article develops a pricing model in the context of Ethereum, although it can be adapted to any blockchain that shares discrete time progression, transaction dependent proposer rewards, and ahead of time proposer election. In Ethereum, transactions use specific amounts of gas

dependent on the combination of state machine operations it uses. At time of writing, each block in Ethereum can contain transactions using up to 30 million gas.

Proposer Rewards

The expected proposer rewards for a specific cumulative gas used within the block forms the foundation of our pricing model. Proposer rewards are the earnings validators receive specifically when chosen to propose a new block. When a validator becomes a block proposer, they gain temporary but complete control over which transactions to include and how to order them. This control allows validators to capture not just standard block rewards and transaction fees, but also what's known as Maximal Extractable Value (MEV) - additional value extracted through strategic transaction ordering and inclusion.

MEV-Boost Proposer Rewards

The complexity of MEV strategies led Flashbots to developMEV-Boost. MEV-Boost, which is now adopted by 90% of all validators, allows validators to outsource their block-building responsibilities to specialised builders through an auction system managed by trusted intermediaries called relayers. The builders in MEV-Boost receive compensation through two primary mechanisms: priority fees and builder tips. Builder tips are direct monetary transfers from transaction originators that can often serve as a complete alternative to traditional priority fees when compensating block builders, as occurs in this case. In the auction stage, builders compete to win the right to construct a block by submitting bids. The winning builder pays a transfer transaction directly to the block proposeras the bid of the auction. Our analysis thus examines how the proposer rewards are structured at the individual transaction level.

We model for each transaction indexed i ∈ {0,\ldots,n-1}

in the winning MEV-Boost block with n

transactions, the builder rewards for transaction i

as \pi_{i, \text{builder}}

. Note, we only focus on the winning builder as this is the only block that is seen per Ethereum slot. We define \pi_{i, \text{builder}} = g_i p_i +b_i

, where g_i

represents the total gas used by the transaction's execution, p_i

is the priority fee of per gas and b_i

is the non-priority fee value received by the builder for the transaction. The total builder reward \Pi_{\text{builder}}

received by the winning builder will therefore be the sum of all the individual builder rewards in the transactions, \Pi_{\text{builder}} = \sum_{i=0}^{n-1} \pi_{i, \text{builder}}

. This value is not necessarily the same as the value paid to the proposer, which we represent by \Pi_{\text{proposer}}

. This value received by the proposer is represented in the winning MEV-Boost block as a transfer placed at the bottom of the block.

Non-MEV Boost Proposer Rewards

The alternative - that roughly 10% of validators use - is non-MEV-Boost 'traditional' block building. Although we do not know the MEV captured by the proposers in this method, we define proposer rewards in a transaction i

as solely the priority tips:

\pi_{i, \text{proposer}} = g_i p_i

Our analysis examines and calculates proposer rewards for a transaction for both the MEV-Boost and non-MEV-Boost block-building approaches.

Inclusion Preconfirmations

This article is focused on pricing inclusion preconfs, a subset of the broader class of proposer commitments. Inclusion preconfs guarantee only that a transaction will be included in a block, but without specifying its exact order, final output, or whether it will revert. Inclusion preconfs are most suitable for straightforward, order-independent state updates like oracle writes, token transfers or rollup batches, which do not require execution guarantees. In a MEV-Boost implementation, proposers can run a MEV-Boost auction with the added restriction that builders must adhere to these inclusion preconfirmations for blocks to be valid in the MEV-Boost auction.

For example, proposers can integrate Bolt through the introduction of theCommitment and Constraint API either via running a light-fork of mev-boost or by running commit-boost. This API enables proposers to commit to including specific transactions and subsequently constraint block builders by requiring them to accept blocks that prove the inclusion of the specified transactions. Builders must provide a Merkle proof of inclusion attached to their bids as evidence. Such proof will be verified in the chosen software that supports the Constraints API.

The Data

The pricing function we propose requires us to model the expected proposer rewards as a function of gas usage. To extract this model, we utilise Ethereum transactions from Dune Analytics of 1,000-blocks (blocks 20425813 to 20426813) that were published on July 31, 2024. The specifically chosen 1,000 blocks is arbitrary and serves as a sample that balances having enough data for meaningful analysis, while being recent and relevant. The range specifically represents a three-hour window from 10:03 to 13:24 UTC and contains approximately 170,000 transactions. Each transaction containing the priority tips, a potential builder tip, and gas usage that we will be using to predict expected proposer rewards. We provide the complete source code and dataset for this study at the following GitHub repository: GitHub - FinnCF/PreconfInclusionPricing: A Repository of Resources for Inclusion Preconfirmation Tip Pricing. Prospective inclusion preconfirmers or users can incorporate any dataset that they wish to tailor the model to their needs.

When observing block proposer rewards, we found a log-normal distribution with a significant right skew. To address potential biases and distortions within our later regression from extreme values, we apply a standard statistical method for outlier detection, the 1.5x Interquartile Range (IQR) method, which in our case defines outliers as rewards exceeding 0.115 ETH. We find that 66 blocks (6.6% of the sample) satisfy this criteria. These abnormal blocks exhibit a mean reward of 0.232 ETH (standard deviation: 0.229 ETH), a significant difference from the remaining average, which exhibits a mean of 0.046 ETH and a standard deviation of 0.024 ETH. Alternative approaches could incorporate these high-variance observations, at a deliberate trade-off between data inclusion and statistical model reliability: that is, incorporating these data points leads to a poor fit in the regression.

Assumptions

In our model, we make the following assumptions:

1. We assume that, at any given time, the proposer does not have complete information about the mempool or all private order flows. Transactions arrive and are pre-confirmed irrespective of the time remaining in the slot. The distribution of proposer rewards within the block thus remains constant, regardless of the time within the slot or the expected block fullness. This simplifying assumption allows our model to exclude any dependence on time remaining in the slot.

2. Aiding in our models calculation of cumulative proposer rewards, we assume MEV-Boost proposer rewards per transaction are distributed identically to builder rewards per transaction. Proposer rewards for each transaction \pi_{i, \text{proposer}}

can thus be determined by scaling the transaction's builder down rewards \pi_{i, \text{builder}}

with the ratio between the total proposer reward and total builder reward \frac{\Pi_{\text{proposer}}}{\Pi_{\text{builder}}}

.

1. Transactions are independent, identically distributed (IID). This assumption is typical in the regression models we use, and is made to avoid overfitting of our pricing model on a sample of block reward data. As we show in the Trend and Autocorrelation Adjustments to Pricing section, this assumption is unlikely to hold in reality. That being said, incorporating dependency assumptions in block reward training data should be carefully done. Our IID assumption and corresponding model will help sanity check any more advanced pricing assumptions.

A Methodology for Inclusion Preconfirmation Pricing

This section outlines the methodology for determining the pricing of an inclusion preconf tip. We first detail a method for calculating the cumulative proposer rewards across a block's gas usage. We then describe how to use these variables in a logarithmic regression. The final section applies this regression model to estimate the average inclusion preconfirmation tip per gas unit, taking into account a specific transaction's gas usage and existing preconfirmations.

Discovering Cumulative Proposer Rewards and Gas Usage

Our method requires us to first discover the expected proposer rewards per unit gas within a block. In the MEV-boost blocks, the proposer reward is represented as a single builder payment, while the builders rewards are the the sum of transactions' priority fees and builder tips. Therefore, we use Assumption 2 to proportionally decrease the builder reward across transactions to estimate individual proposer rewards for each transaction. Therefore:

- If MEV-Boost is used by the validator, the proposer rewards in a transaction is:

\pi_{i,\text{proposer}} = \frac{\Pi_{\text{proposer}}}{\Pi_{\text{builder}}} \cdot \pi_{i, \text{builder}}

- If traditional block building is used by the validator, the proposer rewards in a transaction is:

\pi_{i,\text{proposer}} = g_i p_i

From here, we can discover the cumulative proposer rewards for each transaction S_i

as the running total of all proposer rewards preceding and including transaction i

.

S_{i} = \sum_{j=0}^{i} \pi_{j,\text{proposer}}

Focusing on each transaction's gas used, g_i

, we further discover the cumulative gas used G_i

for each transaction within the block as the sum of all gas that came before, and including, transaction i

:

G_i = \sum_{j=0}^{i} g_j

A Model for Cumulative Proposer Rewards

With our cumulative gas and proposer reward datapoints (S_i, G_i)

, we have the precise observations for how proposer reward accumulates within block-space. The following figure illustrates these discrete data points. Block gas usage exhibits expected variation across different block paths.

[

image

1830×1154 98.3 KB

](https://europe1.discourse-cdn.com/flex013/uploads/lido/original/2X/f/fee2b45e0d3821d4575a0868592f9742999bbd94.jpeg)

Figure 1. Cumulative Proposer Rewards vs Cumulative Gas for the blocks in our dataset.

To understand the relationship between these variables we compared the R^2

of different linear and non-linear regressions. Using Ordinary Least Squares (OLS), we identified a logarithmic function best represents the above relationship. Specifically, our regression leads us to the following function to describe expected cumulative proposer rewards as a function of cumulative gas used:

\hat S(G) =0.019\ln(1.02 \cdot 10^{-6} G + 1)

This model achieves a logarithmic regression fit of $R^2 = 0.26$

. With our regression, we have transformed our discrete observations into a continuous model (in reality, gas usage and rewards remain discrete in actual blockchain transactions). For the remainder of this article we therefore transition from individual i

transaction notation to a continuous function G

, S

that describes how cumulative proposer rewards scale with cumulative block gas usage.

Our analysis of the distribution of proposer rewards reveals that transactions positioned at the top of the block have a disproportionately large share of the total value relative to those in the median and lower segments. The data indicates that a substantial majority of rewards are accumulated within the initial 10 million gas units. This distribution pattern aligns with our expectations, given that transactions at the block's top execute first, and competitive state-dependent operations—particularly those involving CEX-DEX and general arbitrage transactions - demonstrate a natural preference for priority positioning.

Pricing Inclusion Preconfs with Predicted Cumulative Proposer Rewards

This section utilises the predicted cumulative rewards per unit gas model of the previous sections, \hat S(G)

, to deduce a price above which a proposer should be happy to provide a preconfirmation. To extract a price from this model, we must consider what the user is receiving with an inclusion preconfirmation. User's are buying inclusion at any order or position in the block. More specifically, in the presence of a rational proposer, inclusion preconfs are buying the least valuable available blockspace.

In our logarithmic model, the marginal value of the block is decreasing in the gas used. Therefore, at all times, an inclusion preconf should be at most valued equal to the least valuable position of the block. This “at most” comes from the fact that the proposer at block-building time can put transactions that have been inclusion preconfirmed anywhere. In other words, the proposer can extract MEV from the transaction. For uncontentious transactions, transactions most suited to inclusion preconfs, there is likely no MEV to be extracted. Therefore, we expect the true value of an inclusion preconf using IG

gas to be well-approximated by the value of the least valuable IG

gas segment in our model. Given UG

gas has already been preconfirmed, this least valuable IG

is the segment [30M-UG-IG, 30M - UG]

.

Let’s define \hat V()

, the expected value required by a proposer for an inclusion preconf using IG

gas given UG

gas has already been preconfirmed as follows:

\hat V(IG, UG) = \hat S(30M - UG) - \hat S(30M - UG - IG)

Illustrated below is the scenario where a 2M

-sized inclusion-guaranteed transaction within a block already containing 13M

gas of inclusion preconfirmed transactions. Expanding on the intuition for this, non-preconfirmed transactions would be expected to accumulate progressively from the top of the block (the leftmost section), gradually filling the highest-available value block space. In contrast, inclusion preconf transactions would take the lowest-available value block space, approximated by taking the leftmost segment that hasn’t already been taken by another preconf.

[

image

2048×1244 162 KB

](https://europe1.discourse-cdn.com/flex013/uploads/lido/original/2X/9/92e8388547acdf59c2fc97e67327886aa39ed557.jpeg)

Figure 2. We identify the expected remuneration \hat V(IG,UG)

required for a 2M gas inclusion preconfirmation when 13M gas is already inclusion preconfirmed, reading the used gas from right to left, difference between an evaluation at both points on the curve.

Converting Total Preconf Value to a Per-Gas Fee

The expected ETH value of an inclusion preconfirmation to a validator does not provide any meaningful insights for the transactor when interacting with their wallet, as transaction fees are typically quoted per-unit gas. We therefore derive the average inclusion tip per gas \hat T(IG, UG)

, by dividing the inclusion preconf value function of the transaction by the gas the transaction uses. Specifically:

\hat T(IG, UG) = \frac{\hat V(IG, UG)}{IG} = \frac{\hat S(30M - UG) - \hat S(30M - UG - IG)}{IG}

Substituting in \hat S(G)

, we arrive at a final closed-form pricing function for an inclusion preconf tip:

\hat{T}(IG, UG) = \frac{0.019 \ln\left(\frac{1.02 \cdot 10^{-6}(30M - UG) + 1}{1.02 \cdot 10^{-6}(30M - UG - IG) + 1}\right)}{IG}

[

image

1680×1102 89.2 KB

](https://europe1.discourse-cdn.com/flex013/uploads/lido/original/2X/f/f403a6de51ff19665dfd8ed1fcea69aad21a0898.jpeg)

Figure 3. Pricing curve for a preconfirmed inclusion tip for any given transaction size, taking into consideration any pre-existing pre-confirmed gas usage.

We plot our three variables as a surface in Figure 3. At the theoretical minimum, with zero preconfirmed gas and a transaction size of 21,000 (a transfer of ETH), the inclusion tip per gas is 0.61 GWEI. As preconfirmed gas increases, the inclusion tip per gas rises exponentially, reaching an average of 1.17 GWEI when the target rate of 15M gas is achieved. In the most expensive scenario, our model suggests an average inclusion tip per gas reaches an average of 15 GWEI per gas. This being said our model becomes less reliable as the amount of gas preconfirmed approaches 30M. There are many reasons for this, but one important reason is that we omit large outlier transactions to improve average fit, which disproportionately affects the most valuable transactions.

Practical Considerations

While the previous sections constructed a model for a theoretical intrinsic price per inclusion preconf, the actual market price that emerges will be shaped by many factors not included in our model including varying expected proposer rewards, changing demand for block-space, and proposer risk tolerance to name but a few. This section explores such considerations towards a reliable real-world model.

Intrinsic Pricing vs Market-Based Pricing

As proposers issue preconfirmation transactions (and if these preconfirmations are public), an ‘implied expected proposer rewards’ curve will emerge. This market-implied curve will supersede any intrinsic value model, such as the one we have presented. [This effect can be seen in financial markets where option derivatives are priced according to market supply and demand, and not always according to Black-Scholes.]

Constant Pricing: Practicality vs Accuracy

Proposers may opt for a simpler fixed minimum inclusion preconfirmation tip at configuration, accepting transactions where the inclusion tip per gas unit exceeds a predetermined threshold. This simplicity can be seen as beneficial for user experience and adoption.

Figure 4 shows expected proposer rewards under different constant minimum tip thresholds of T_{\text{const}}

equal to 0.5, 1.0

, and 1.5

GWEI.

[

image

1516×986 183 KB

](https://europe1.discourse-cdn.com/flex013/uploads/lido/original/2X/2/202c79ea3d8a2e705f1eee816e95381664f74ccc.jpeg)

[

image

1568×1036 171 KB

](https://europe1.discourse-cdn.com/flex013/uploads/lido/original/2X/5/559e09942717c7f8ee8f79e95fac714ab74b47e8.jpeg)

Figure 4. Comparison of our proposer rewards over differing amounts of inclusion preconfs. Our model is compared to three constant inclusion tip pricing models, with the lower figure displaying the difference between the constant model and ours.

At a constant 0.5 GWEI inclusion tip per gas, a rate lower than our model’s minimum of 0.61 GWEI, expected proposer rewards remain worse off than using our model. In other words, it is never appropriate in expectation to accept a preconfirmation tip lower than 0.61 GWEI. Setting a 1.5 GWEI tip, the proposer can potentially profit across inclusion tips up to 27M gas. Unfortunately for the proposer, such high fees will likely exceed the amount user’s are willing to pay for inclusion tip. Any fee, including a constant fee, must consider this tradeoff between over-pricing and under-pricing. Generally, this is addressed by pricing according to demand, as discussed in the previous section.

Trend and Autocorrelation Adjustments to Pricing

Predictive variables can be incorporated to improve the accuracy of our model. Our current method makes the assumption (Assumption 3) that each block's proposer rewards come from the same probability distribution, regardless of when they occur. This serves as a useful starting point, but overlooks patterns in how blocks actually arrive. Block rewards might show trends over time, correlation with previous blocks' rewards (autocorrelation), and patterns that lend to the next blocks data being able to be predicted from previous ones. Our full methodology for this section is shown in the Appendix, including a method for adjusting how inclusion tip per gas are calculated.

Verifying this in our dataset, in Figure 5 we observe that median priority fee per gas, a large factor to proposer rewards, exhibits a statistically significant upwards trend, increasing throughout our 1000 block sample. The next median priority fee per gas, and subsequent curve \hat S(G)

should therefore take this into account, and be increased to reflect it.

[

image

2048×844 138 KB

](https://europe1.discourse-cdn.com/flex013/uploads/lido/original/2X/6/6d30390dce268d158ca1baacb346c2011135a2dc.jpeg)

Figure 5. A comparison of per-block metrics and trend: median proposer rewards versus their trend, and median priority fee per gas versus their trend. Here we see an statistically significant increasing trend (p=0.000009)

, with data points climbing steadily from left to right.

[

image

2048×683 129 KB

](https://europe1.discourse-cdn.com/flex013/uploads/lido/original/2X/f/f07290c9858a67c15b5a8fe47c779972545f5124.jpeg)

Figure 6. A comparison of autocorrelation and partial autocorrelation for median priority fee per gas and proposer reward. Data points extending above the blue confidence bands show statistically significant correlations at those lag periods. While the autocorrelation function shows persistent significant correlation across all lags, the partial autocorrelation function drops to insignificance after the first few lags.

After removing the deterministic trend to prevent it being interpreted as between-block relationships and validating stationarity, we validate the presence of autocorrelation. Figure 6 reveals statistically significant correlations for median priority fee extending to the 48th lag, with partial autocorrelation remaining significant up to 6 lags. This suggests that there is autocorrelation, and the most recent values contain the most predictive information for median priority fee per gas.

Any short-term predictive power we identify could lead to an array of possible adjustments. This justifies exploration of optimal historical windows, developing more sophisticated prediction models, understanding different market regimes, and studying how various proposers might incorporate these time-series signals into their own unique pricing strategies.

Adjust the Training Data

While our analysis uses an arbitrary 3-hour sample of 1000 blocks, market participants may prefer different historical windows to calculate expected proposer rewards. Some proposers will likely favour recent, short-term data for their regression, while others may rely on more stable long-term historical trends.

Incorporate Real-Time Data into the Model

In a system where the proposer has real-time information about the demand for block-space during their slot e.g. observing the public mempool, the proposer can update their pricing model accordingly. That being said, the increasing prevalence of private orderflow puts the average proposer at a significant informational disadvantage when trying to include such estimators.

Chainbound Case-Study: Learnings from Node Operators and Proposers

- Node operators indicate that an additional fixed fee is justifiable for inclusion preconfirmations, given they reduce confirmation time from ~7 seconds to 200ms. Users and dApps likely have less elastic demand than for normal transactions due to this UX improvement, allowing proposers to charge this additional fixed fee.
- In Chainbound model development, the following were explored:

- Fixed pricing: Setting a static preconf tip at configuration time
- Dynamic pricing: Setting a dynamic price by calculating the 60th percentile of priority fees from the previous 20 blocks, plus a fixed additional fee.
- The Chainbound dynamic pricing model aligned with this study's findings, achieving an average 1 gwei dynamic fee compared to the study's 0.67-1.15 gwei range
- The Chainbound dynamic pricing model aligned with this study's findings, achieving an average 1 gwei dynamic fee compared to the study's 0.67-1.15 gwei range
- Fixed pricing: Setting a static preconf tip at configuration time
- Dynamic pricing: Setting a dynamic price by calculating the 60th percentile of priority fees from the previous 20 blocks, plus a fixed additional fee.
- The Chainbound dynamic pricing model aligned with this study's findings, achieving an average 1 gwei dynamic fee compared to the study's 0.67-1.15 gwei range
- The Chainbound dynamic pricing model aligned with this study's findings, achieving an average 1 gwei dynamic fee compared to the study's 0.67-1.15 gwei range
- Currently, the Chainbound pricing approach hasn't accounted for block space scarcity. Chainbound's model initially assumed that block space is abundant and that inclusion preconf prices remain constant throughout slot progression.
- This article, along with operator feedback, demonstrates that block space is indeed scarce. Even with increased block limits, we'll face upper bounds that must be factored into preconf pricing logic.
- Recommendation to Node Operators:

The pricing methodology outlined in this article proves highly accurate, and we confidently endorse it as the latest development in precise pricing. While it represents a substantial optimization, some limitations remain, many of which have been outlined earlier in this section.

Conclusion

We present a model for estimating the price of inclusion preconfs. This model stands as a reference point for both proposers selling preconfs, and users buying preconfs. It is presented in a simple yet expressive closed form formula that can be easily adjusted to incorporate new training data, new parameters and/or additional predictive information. The practical considerations we outline are generally applicable to any preconf pricing strategy, and should be considered by both buyers and sellers of preconfs. We are excited to see how preconf pricing evolves as the preconf market matures.

Appendix

We provide here a more in depth analysis of trend, autocorrelation and seasonality of rewards within blocks. Specifically, we describe a method to use these predictive variables to shift our pricing function \hat S(G)

and their effect on subsequent inclusion, and analyse the variables of median priority tips and proposer rewards.

The components of our defined proposer rewards per transaction: priority fees, builder tips and the final proposer payment, are all likely to affected by influences from recent real-time factors like market activity, network congestion, and changing demand for block space. For example, comparing overarching proposer rewards to median priority tips, Figure 7 below plots a simple linear OLS regression discovers this upward-sloping deterministic trend in proposer rewards within our sample. Our regression is statistically significant to 1% correlation coefficient, with p-value of 0.004. We observe the same for median priority fee per gas, with statistical significant to 1% and a p-value of 0.000009. We can therefore deduce that a trend is present in our 3 hour sample, that can be deterministically added into any prediction.

[

image

2048×860 153 KB

](https://europe1.discourse-cdn.com/flex013/uploads/lido/original/2X/d/d19981e7ac9d88888706768898b3c8190d20d1a6.jpeg)

[

image

2048×844 138 KB

](https://europe1.discourse-cdn.com/flex013/uploads/lido/original/2X/6/6d30390dce268d158ca1baacb346c2011135a2dc.jpeg)

Figure 7. A comparison of per-block metrics and trend: median proposer rewards versus their trend, and median priority fee per gas versus their trend.

We then remove these trends to avoid skewing our results. We proceed to study how our two metrics relate to their own past values using auto-correlation (which measures how a value relates to its previous values) and partial auto-correlation (which shows direct relationships between values at different intervals). As shown in Figure 8, these metrics behave quite differently: proposer rewards have a “short memory,” being strongly influenced only by the immediately previous block and showing significant correlation only at lag 1, with some effect at lag 8. In contrast, median priority fees have a “longer memory” with effects persisting across about 40 blocks, taking about 7 blocks to return to normal after unexpected changes, unlike proposer rewards which adjust within one block.

[

image

2048×666 105 KB

](https://europe1.discourse-cdn.com/flex013/uploads/lido/original/2X/e/eb976a8e87e92d1eb936181599312adcd468dbc1.jpeg)

[

image

2048×683 129 KB

](https://europe1.discourse-cdn.com/flex013/uploads/lido/original/2X/f/f07290c9858a67c15b5a8fe47c779972545f5124.jpeg)

Figure 8. A comparison of autocorrelation and partial autocorrelation for median priority fee per gas and proposer reward.