See also: ideas near the end of this thread that led to this concrete proposal.

Consider a version of EIP 1559 that works as follows. The protocol maintains a parameter, excess_gas_issued

. The protocol has a pricing function:

eth_qty(gas_qty) = exp(gas_qty / TARGET / ADJUSTMENT_QUOTIENT)

Where exp(x) \approx 2.71828^x

. In an actual implementation, this would of course be replaced with an integer-math-friendly approximation, but it's mathematically cleaner to think of the above formula.

When a block proposer wants to make a block that contains gas_in_block

gas, they need to pay a "burn fee" equal to eth_qty(excess_gas_issued + gas_in_block) - eth_qty(excess_gas_issued)

:

After the block is processed, we set excess_gas_issued = max(0, excess_gas_issued + gas_in_block - TARGET)

.

The analogy here is that the blockchain manages a pool of gas, which block producers have to "buy" according to a constant-function automated-market-maker curve. Block producers can buy a maximum of SLACK_COEFFICIENT * TARGET

gas in a block (eg. SLACK_COEFFICIENT = 2

). Additionally, in every block, a virtual agent "sells" an extra TARGET

gas.

This creates a mechanism very similar to EIP 1559. The "basefee" equivalent here is the derivative of the curve at the current excess_gas_issued

value, so basefee = eth_qty(excess_gas_issued) / (TARGET * ADJUSTMENT_QUOTIENT)

(remember basic calculus for how \frac{d}{dx} e^{kx}

is computed).

But this mechanism is superior in two key respects:

1. We have a hard invariant for how to compute the basefee

as a function of excess_gas_issued

. This ensures that over time, excess_gas_issued

is bounded, meaning that long-run gas usage is guaranteed to be very close to TARGET * (number of blocks)

. There is no fancy trick that you can pull with moving transactions between blocks to make long-run average usage per block be anything other than TARGET

.

1. It is resistant to the instability scenario where there are many transactions whose max-basefee is N

and the chain jumps between 2x-full blocks with a basefee of N * 0.9

and empty blocks with a basefee of N * 1.0125

. This is because even the formula within a block

is nonlinear, so if the gasprice starts off at N * 0.9

with an ADJUSTMENT_QUOTIENT

of 8, you would expect to see one block with TARGET * 0.843

gas followed by a chain of blocks with TARGET

gas.

Note that because of the nonlinearity of the burn, EIP 1559 would need to be adjusted. There are a few options:

1. The proposer pays the burn and the full fees from the transactions (including the basefee component) go to the proposer. Note that this still requires an algorithm to determine how high the basefee

is considered to be in transactions that specify their gasprice in the form basefee + tip

.

1. The transaction origin pays a basefee equal to the maximum it could pay (that is, the basefee at excess_gas_issued + TARGET * SLACK_COEFFICIENT

), and then at the end

of block execution, everyone gets refunded so that the end result is that everyone pays the implied average basefee (the "implied average basefee" is (eth_qty(excess_gas_issued + gas_in_block) - eth_qty(excess_gas_issued)) / gas_in_block

; the refund is the difference between the originally paid basefee and this amount)