Almost all algorithmic stablecoins are good in mitigating periods of high demand but fail to offer sustainable stability mechanism during periods of low demand

The core idea is to shift the demand in time in order to achieve a low volatile crypto native unit of account

. That is; temporarly disable selling if price is low and temporarly disable buying if price is high.

Example Implementation:

Say PP coin (aka purchasing power coin) targets 1 DAI as a price target

Say PP is mainly bought from or sold to PP-DAI pool (highest liquidity market)

We define the 2 types of txns to contract

Buy txn = Send DAI to pool Recieve PP

Sell txn = Send PP to pool Recieve DAI

Target Realized Cap (TRC) = Σ [Target price of PP in units of DAI (= 1) x PP amount in realized buy/sell txn over z blocks ]

Actual Realized Cap (ARC) = Σ [Actual price of PP in units of DAI at the time of txn x PP amount in realized buy/sell txn over z blocks ]

if

TRC > ARC

then

//Slow down sell txns

Disable sell txns up to certain block height (determined by the difference between TRC and ARC. If ARC = TRC/10 then sell txns become temporarily disabled for 10*y blocks where y is coefficient of slowdown )

else if

TRC < ARC

//Slow down buy txns

Disable sell txns up to certain block height

Obviously this is a very simplistic high level view. What are your thoughts? How can it be improved…