Dexy - A Stablecoin With Algorithmic Central Bank

Alexander Chepurnoy, Amitabh Saxena

June 25, 2022

Abstract

In this paper, we consider a new stablecoin protocol design called Dexy. The protocol has two mandatory components, reference market (in form of liquidity pool where anyone can trade base currency against stable coins), and an algorithmic central bank, which is responsible for new stablecoins issuance, and also for stabilizing stablecoin price in the reference market.

1 Introduction

Algorithmic stablecoins is a natural extension of cryptocurrencies, trying to solve problems with volatility of their prices by pegging stablecoin price to an asset which price is considered to be more or less stable with time (e.g. gold).

Having an asset with a stable value could be useful for:

- securing fundraising; a project can be sure that funds collected during fundraising will have stable value in the mid- and long-term.
- doing business with predictable results. For example, a shop can be sure that funds collected from sales will be about the same when the shop is ordering goods from warehouses (otherwise, the shop may go bankrupt if margin is not that big).
- shorting: when cryptocurrency prices are high, it is desirable for investors to rebalance their portfolio by increasing exposure to fiat currencies (or traditional commodities). However, as fiat currencies and centralized exchanges impose significant risks, it would be better to buy fiat and commodity substitues in form of stablecoins on decentralized exchanges.
- lending and other decentralized finance applications. Stability of collateral value is critical for many applications.

Algorithmic stablecoins are different from centralized stablecoins, such as USDT and USDC, for which there is a trusted party doing conversions to a pegged asset. In case of an algorithmic stablecoin, the pegging is done via rebasement of total supply (as in Ampleforth), or via imitating the trusted party, which holds reserves and doing market interventions when it is needed for getting exchange rate back to the peg. Imititating the trusted party is usually done by allowing anyone on the blockchain creating over-collateralized financial instruments, such as collateralized debt positions (as in DAI) or zero-coupon bonds (as in the Yield protocol).

Alex notes : add links to the paragraph above

In this work we present Dexy, a stablecoin protocol where the bank presented explicitly as a contract with few predefined rules. The bank is trying to stabilize stablecoin value on the markets, using a liquidity pool as a reference market, by providing stablecoin liquidity, when stablecoin is over the peg, or injecting base currency from its reserves, when the stablecoin is under the peg. In extreme case, when bank reserves are depleted and stablecoin is still under the peg, its value is restored by burning stablecoin in the liquidity pool.

2 Dexy Design in General

Unlike popular algorithmic stablecoins based on two tokens (instruments), Dexy is based on one token but two protocols. In the first place, there is market where trading of Dexy vs the base currency (ERG) happens. In the second place, if market price is way too different from target price (reported by an oracle), there is an algorithmic central bank which makes interventions. The central bank can also mint new Dexy tokens, selling them for ERG. The bank is using reserves in ERG it is buying for interventions then.

As a simple solution for the market, we are using constant-factor Automated Market Maker (CF-AMM) liquidity pool, similar to ErgoDEX and UniSwap. The pool has ERG on one side and Dexy on another. For CF-AMM pool, multiplication of ERG and Dexy amounts before and after a trade should be preserved, so $e * u = e' * u'$, where e and u are amounts of ERG and Dexy in the pool before the trade, and e' and u' are amounts of ERG and Dexy in the pool after the trade. It is also possible to put liquidity into the pool, and remove liquidity from it.

The bank has two basic operations. It can mint new stablecoin tokens per request, by accepting ERG in its reserves. It also can intervene into markets by providing ERG from reserves when needed (namely, when price in the pool $\frac{u}{e}$ is significantly different from price on external markets p which reported by oracle).

Now we are going to consider how to put restrictions and design rules for the system to have stable price of Dexy.

3 Notation

We start with introducing notation:

- T - period before intervention starts. After one intervention the bank can start another one after T.
- p - price reported by the oracle at the moment (for example, 20 USD per ERG)
- s - price which the bank should stand in case of price crash. For example, we can assume that $s = \frac{p}{4}$ (so if p is 20 USD per ERG, then s is 5 USD per ERG, means the bank needs to have enough reserves to save the markets when the price is crashing from 20 to 5 USD per ERG)
- R - ratio between p and s, $R = \frac{p}{s}$
- e - amount of ERG in the liquidity pool
- u - amount of stablecoin in the liquidity pool
- O - amount of stablecoin outside the liquidity pool. The distribution in O is not known for the Dexy protocol.
- E - amount of ERG in the bank.
- r - ratio between p, and price in the pool, which is $\frac{u}{e}$, thus $r = \frac{p}{\frac{u}{e}} = \frac{p*e}{u}$

4 Worst Scenario and Bank Reserves

The bank is doing interventions when the situation is far from normal on the markets, and enough time passed for markets to stabilize themselves with no interventions. In our case, the bank is doing interventions based on stablecoin price in the liquidity pool in comparison with oracle provided price (we can assume that price on other markets is similar to liquidity poolś due to arbitrage). The bankś intervention then is about injecting its ERG reserves into the pool.

First of all, let's assume that price crashed from p to s sharply and stands there, and before the crash there were e of ERG and u of stablecoin, respectively, with liquidity pool price being p. The worst case is when no liqudity put into the pool during the period T. With large enough T and large enough R this assumption is not very realistic probably: liquidity will be put into the pool by arbitrage players, price is failing with swings where traders will mint stablecoin by putting ERG into bank reserves, and so on. However, it would be reasonable to consider worst-case scenario, then in the real world Dexy will be even more durable than in theory.

In this case, the bank must intervene after T units of time, as the price differs significantly, and restore the price in the pool, so set it to s. We denote amounts of ERG and stablecoin in the pool after the intervention as e' and u', respectively. Then:

- $e * u = e' * u'$
- as the bank injects E_1 ergs into the pool, $e' = e + E_1$

- $\frac{u'}{e'} = s$, thus $u' = s * (e + E_1)$
- from above, $E_1 = \sqrt{\frac{e*u}{s}} - e$

So by injecting E_1 ERG, the bank recovers the price. However, this is not enough, as now there are O stablecoin units which can be injected into the pool from outside. Again, in the real life it is not realistic to assume that all the O stablecoin would be injected, as some of them are simply lost. However, we need to assume worst-case scenario. We also assume that those O tokens are being sold in very small batches not significantly affecting price in the pool, and after each batch seller of a new batch is waiting for a bank intervention to happen (so for T units of time), and sells only after the intervention. In this case all the O tokens are being sold at price close to s, so the bank should have $E_2 = \frac{O}{s}$. We note that this scenario is also not realistic and takes very long time. However, as before we assume the absolutely worst case.

Summing up E_1 and E_2, we got ERG reserves the bank should have to be ready for worst-case scenario: $E_w = E_1 + E_2 = \sqrt{\frac{e*u}{s}} - e + \frac{O}{s}$.

It is simple to see why this scenario is worst-case for the bank. In this scenario, the bank is buying all the O of external *stablecoin* at the worst possible price s, and get to this price by burning its own reserves only.

5 Bank and Pool Rules

Based on needed reserves for worst-case scenario estimation, we can consider minting rules accordingly. Similarly to SigUSD (a Djed instantiation), we can, for example, target for security in case of 4x price drop, so to consider $R = \frac{p}{s} = 4$, and allow to mint stablecoin while there are enough ERG in reserves, so while there are not less than E_w ERG in reserves. However, in this case most of time stablecoin would be non-mintable, and only during moments of ERG price going up significantly it will be possible to mint.

Thus we leave worst-case scenario for UI, so dapps working with the Dexy may show level of reserves, in comparison with worst-case scenario estimations. In this case, having on-chain data analysis, more precise estimations of reserve quality can be made (by considering stablecoin locked in DeFi protocols, likely forgotten, etc).

We are proposing following minting rules.

- Arbitrage mint: if price reported by the oracle is higher than in the pool, i.e. $p > \frac{u}{e}$, we allow to print enough stablecoin to bring the price to p. That is, the bank allows to mint up to $\delta_u = \sqrt{p * e * u} - u$ stablecoin by accepting up to $\delta_e = \frac{\delta_u}{p}$ ERGs into reserves.

  To instantiate the rule, we can allow for arbitrage minting if the price p is more than $\frac{u}{e}$ by at least 1% for time period T_{arb} (e.g. 1 hour), also, the bank is chargin 1% fee for the operation.

- *Free mint: we allow to mint up to* $\frac{u}{100}$ *stablecoins within time period* T_{free}. To instantiate the rule, we propose to have bank fee of 1%, and allow for free mint if $0.98 < r < 1.02$.

In addition to minting actions, which increase bank reserves only, we define following two actions which decrease them:

- Intervention: if reported by the oracle is lower than in the pool by significantly enough margin, i.e. $\frac{p*e}{u} < r$, where r is some constant which is hard-wired into the protocol, then the bank is providing ERGs. to restore the price. *To instantiate the rule, we propose to allow the bank to intervene if* $r <= 0.98$ *for time period* T_{int}

- *Payout: if bank has too much reserves, so* $E > E_w$*, we can allow for paying excess reserves out. There could be different ways to do this. E. g. extra reserves can be paid to holders of liquidity pool LP tokens.*

We also state following rules for the liquidity pool (which, otherwise, acts as ordinary CP-AMM liquidity pool):

- Stopping withdrawals: if r is below some threshold, withdrawals are stopped. *To instantiate the rule, we propose to stop withdrawals immediately if* $r <= 0.98$.

- *Burn: if the bank if empty, and* r *is below some threshold, it is allowed to burn stablecoins in the pool.* To instantiate the rule, we propose to stablecoin burn if $r <= 0.95$ for time period T_{burn}. T_{burn} must be quite big

6 Stability

What provides stability for the stablecoin when we have the design sketched in the previous sections?

Alex notes : finish the section

7 Implementation

Alex notes : put contracts here

8 Simulations

We made simulations. Alex notes : finish

9 Extensions

Acknowledgments

Authors would like to thank.