Dexy - A Stablecoin With Algorithmic Bank Interventions

Alexander Chepurnoy, Amitabh Saxena

June 1, 2022

Abstract

In this paper, we consider a new stablecoin design called *Dexy*.

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 believed to be more or less stable with time (e.g. 1 gram of gold).

Stable pricing could be useful for:

1. securing fundraising; a project can be sure that funds collected during fundraising will have stable value in the mid- and long-term.

2. doing business with predictable results. For example, a hosting provider can be sure that payments collected at the beginning of the month would be about the same at the end of the month when the provider will need to pay the bills.

3. 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 KYC procedures are often cumbersome or not possible, it would be better to buy fiat and commodity substitues in form of stablecoins on decentralized exchanges.

4. lending and other decentralized finance applications. Stability of asset value is critical for many applications.

Algorithmic stablecoins are different from centralized stablecoins, such as USDT and USDC, which are convertible into underlying (pegged asset) via a trusted party. In case of an algorithmic stablecoin, its pegging is done via rebasement of total supply, or via imitating the trusted party, which holds.

2 General Design

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 Dexy tokens and

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

3 Stability

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

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.

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.

Note that current price in the pool is $\frac{u}{e}$.

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: $E = E_1 + E_2 = \sqrt{\frac{e*u}{s}} - e + \frac{O}{s}$.

5 Minting Rules

- Free mint
- Arbitrage mint

6 Implementation

7 Simulations

8 Extensions

Acknowledgments

Authors would like to thank.