# Terra Constant-Product Swaps Proposal (WIP)

Nicholas Platias
8/7/2019

The following is a proposal for a constant-product market maker to implement Terra/Luna swaps. We assume familiarity with constant-product market making and uniswap.

**Design Objective**

It is essential to be clear on the difference in objective between our rendition of the constant product model and the original: we are designing a pegged currency market for *Terra* that uses a *Luna* price feed, not a standalone Terra/Luna market. The difference is subtle but key, and pertains to the question: when do we want the market to have an incentive to trade?

In a traditional uniswap contract, say ETH/ZRX, the market is invited to express an opinion on ETH vs ZRX, so if ZRX demand surges relative to ETH you would expect a lot of ETH to be sold for ZRX. Contrast to the market we are designing: the objective is to keep Terra at the peg, and **we want the market to trade when its opinion on Terra's price has changed relative to the peg, *not* relative to Luna**. For instance: if Luna demand surges relative to Terra, but Terra demand relative to the peg (say SDR) does *not* change, we should *not* be offering an incentive to trade relative to off-chain markets. Conversely, say both Terra and Luna double in price relative to the SDR. In that case we certainly *do* want strong incentives to trade, even though the market's relative desire for Terra vs Luna has *not* changed.

The implication is that the incentives to trade in the market we are designing need to be independent of Luna's price in fiat terms. This is of course the job of the oracle. It is therefore most accurate to reason about price and spread in terms of Terra vs the peg, not in terms of Luna. For instance, a 2% spread on either side is best interpreted as the market offering to buy Terra at 0.98 SDR and sell Terra at 1.02 SDR.

**Liquidity Pools**

The protocol maintains virtual liquidity pools of Terra and Luna.

- The size of the Terra pool is initialized at a target liquidity.
- The size of the Luna pool is initialized to offer a Terra price at the peg, i.e. (size of Terra pool)/(Luna price).

Target Terra liquidity is a function of Terra supply (say 1%) and determines the maximum amount of Terra that can be issued or burned in any 24h period. The cap is set relative to Terra rather than Luna supply given the large asymmetry between the market capitalizations of the two (Luna's being much larger). Both empirically and theoretically, Luna's market capitalization is expected to be larger than Terra's, meaning that a cap relative to Terra supply serves as an effective cap on Luna's supply — the opposite would not be the case.

**Constant Product**

The invariant that governs the pools is that the product (**size of Terra pool)*(fiat value of Luna pool) is constant**. Notice the subtle difference between this invariant and uniswap's: we use the *fiat* value of the Luna pool, not its size per se. What this accomplishes is that changes in Luna's *price* do *not* change the product, rather they change the *size* of the Luna pool.

Swaps are served using the constant product equation. Say T, L are sizes of the Terra and Luna pools at some point in time, and f() gives the fiat value of a pool.

- Buying x Terra costs y SDR worth of Luna, where (T-x)*(f(L)+y) = T*f(L)
- Selling x Terra pays y SDR worth of Luna, where (T+x)*(f(L)-y) = T*f(L)

**Bid and ask prices**

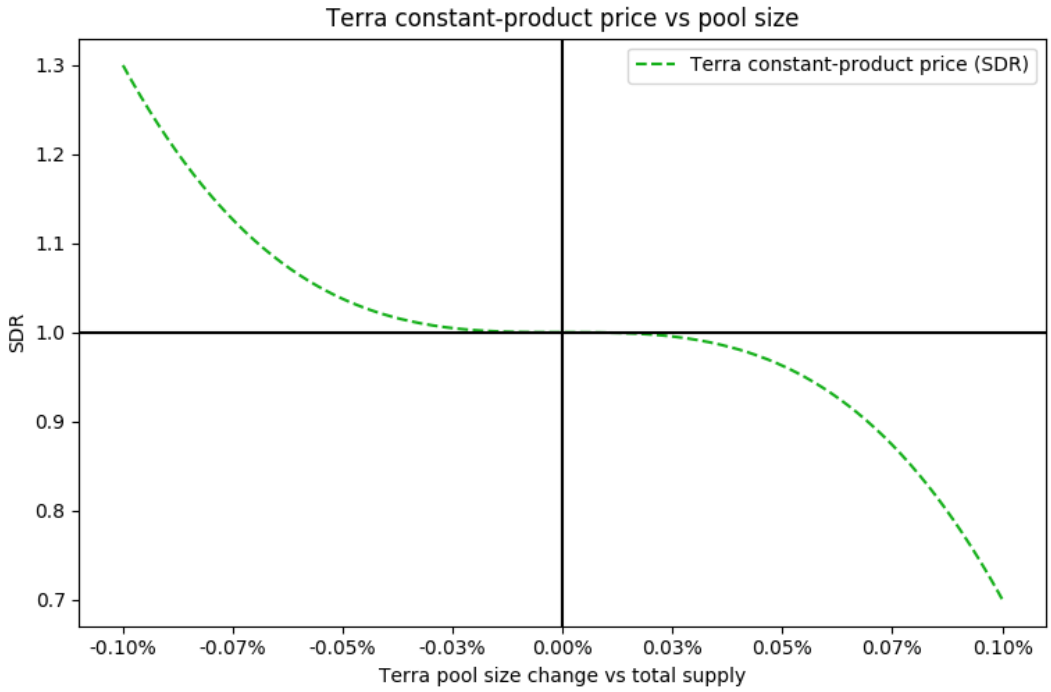
There is one problem with the above formulation: it offers the *same* price to buyers and sellers (for infinitesimal quantities). This is desirable for uniswap's use case, but not for ours. For instance, say that the Terra price offered by the rule above is 0.95 SDR. Then the protocol should

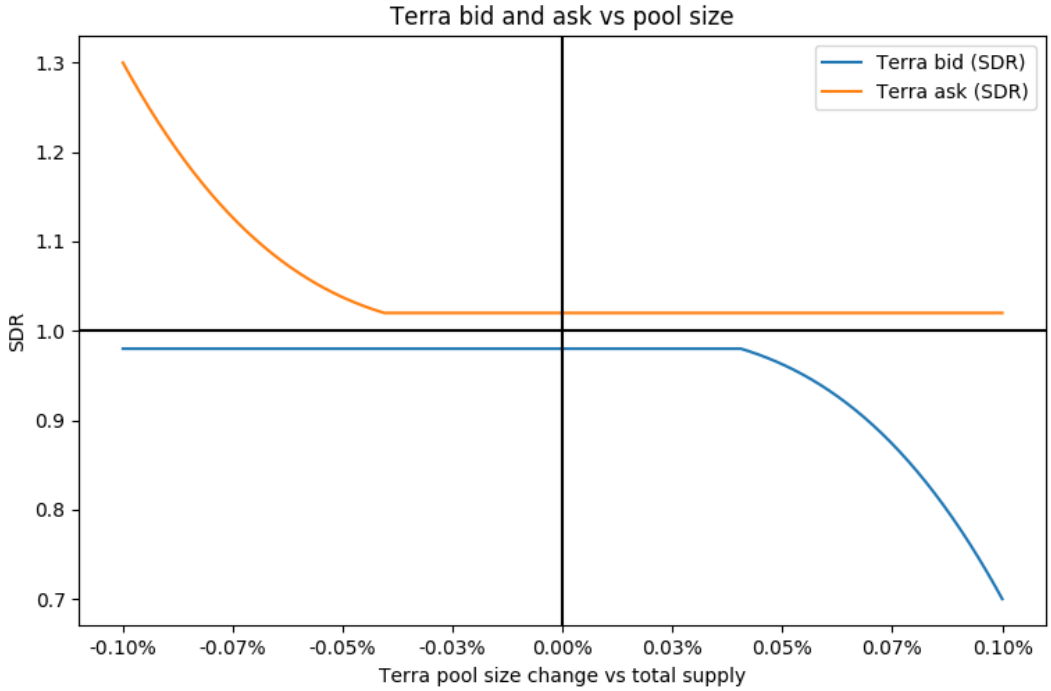
be willing to buy Terra at 0.95 SDR (implying a 5% spread on the bid side), but it should certainly not be *selling* Terra at 0.95 SDR (it should be selling at 1 SDR + some minimum ask spread). Therefore, **our swap mechanism needs to be able to offer different prices to buyers and sellers to serve the peg correctly**. This is a key difference between swaps for a regular market (e.g. uniswap) and swaps for a currency peg.

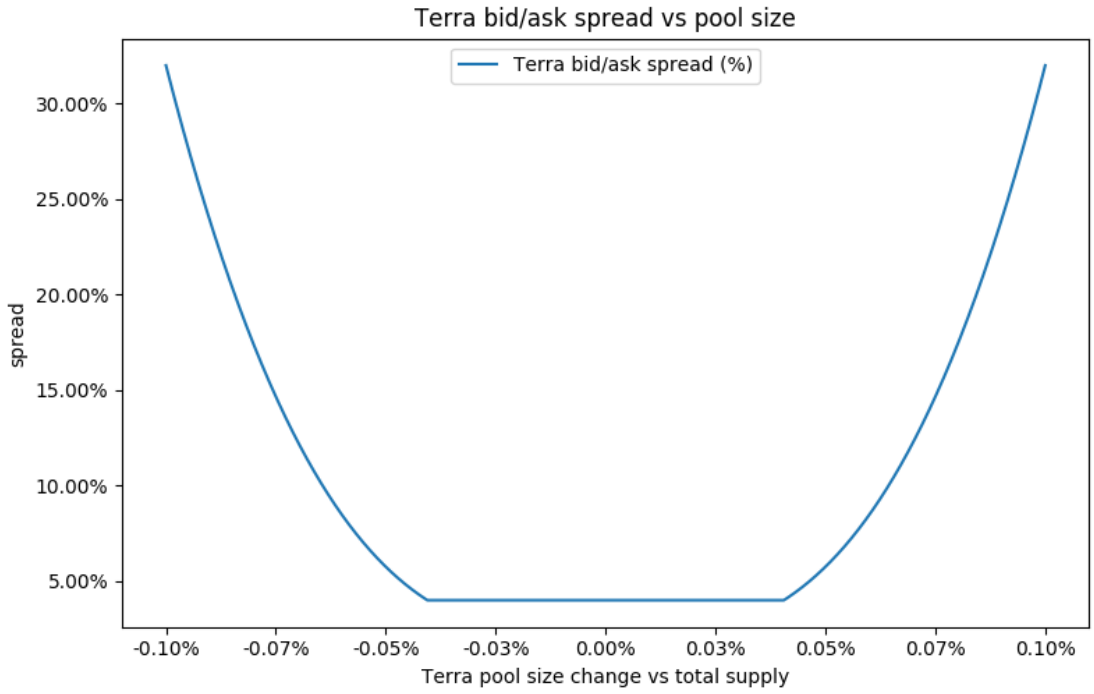
So how do we use the constant product rule to determine bid and ask prices? First, we need to choose a minimum spread to charge on both bids and asks, say 2%. The protocol therefore buys Terra at 0.98 SDR or less, and sells Terra at 1.02 SDR or more. To enforce this, we restrict the bid and ask prices determined by the constant product rule to those ranges:

- If the constant product bid price is more than 0.98 SDR, adjust the bid price to 0.98 SDR.
- If the constant product ask price is less than 1.02 SDR, adjust the ask price to 1.02 SDR.

The graphs below demonstrate how the constant-product price, bid price, ask price and bid/ask spread change depending on the size of the Terra pool. In all graphs, the x-axis shows the change in the size of the Terra pool as a percentage of Terra's total supply. In the right half the change is net positive, meaning that Terra sell pressure exceeds buy pressure, and so the constant-product price should be below the peg. In the left half the change is net negative, meaning that Terra buy pressure exceeds sell pressure, and so the constant-product price should be above the peg. The first graph shows simulated constant-product prices as a function of Terra pool size. The second graph shows Terra bid and ask prices following adjustment to the constant-product price: bid price is at most 0.98 SDR, and ask price is at least 1.02 SDR. As we would expect, when the constant-product price is in the range (0.98, 1.02), bid and ask are both at their respective upper/lower limits. The third graph shows the resulting Terra bid/ask spread. As we can see, the spread is constant at 4% when the constant-product price is in the (0.98, 1.02) range, and grows fast as the price diverges either above or below.


Terra constant-product price vs pool size
Terra constant-product price (SDR)
SDR
1.3
1.2
1.1
1.0
0.9
0.8
0.7
-0.10%
-0.07%
-0.05%
-0.03%
0.00%
0.03%
0.05%
0.07%
0.10%
Terra pool size change vs total supply


Terra bid and ask vs pool size
Terra bid (SDR)
Terra ask (SDR)
SDR
1.3
1.2
1.1
1.0
0.9
0.8
0.7
-0.10%
-0.07%
-0.05%
-0.03%
0.00%
0.03%
0.05%
0.07%
0.10%
Terra pool size change vs total supply




Note that to maintain the constant product invariant after this modification, the size of the Luna pool will need a small adjustment whenever the constant-product price is different from the price at which a trade is executed. For instance, if the constant product ask price is 0.95 SDR and someone buys Terra he/she will instead be charged 1.02 SDR. The Luna surplus needs to be burned to maintain the constant product.

**Oracle price update**

An implication of our constant product definition is that an update in Luna's oracle price needs to adjust the size of the Luna pool *to maintain its previous fiat price.* For instance, if Luna's price doubles between two consecutive price updates, the size of the Luna pool is cut in half relative to its size immediately prior. It should be clear why this is necessary: if Luna's fiat price changed but the size of the Luna pool remained the same, the *fiat* price of Terra offered by the market would have changed. We obviously don't want that: the fiat price of Terra offered needs to depend strictly on changes in Terra demand, not changes in Luna demand.

**Liquidity Replenishment**

We explained how the pools are initialized and how trades change their size. We also need a way to gradually *replenish* liquidity to prevent the spread on either side from growing too large. Note that at equilibrium (no recent trades), the Terra pool is at target liquidity and the pools are sized so that the price of Terra offered is at the peg. Replenishing liquidity simply means increasing the size of either the Terra or Luna pool, depending on whether the price of Terra offered is more on less than the peg, until the target Terra pool size is reached and the price offered is at the peg. If recent buying pressure exceeds selling pressure, then the size of the Terra pool is decreasing, and the price of Terra offered is increasing. Replenishing liquidity means gradually increasing the size of the Terra pool/decreasing the size of the Luna pool until the price of Terra returns to the peg. Conversely, if recent selling pressure exceeds buying pressure, then the size of the Terra pool is increasing, and the price of Terra offered is decreasing. Replenishing liquidity means gradually decreasing the size of the Terra pool/increasing the size of the Luna pool until the price of Terra returns to the peg.

Potential approaches include resetting liquidity to the target every 24h, or adjusting liquidity continuously based on trailing 24h trades. Both approaches suffer from the problem that they may create liquidity jumps, where the price/spread adjusts abruptly. The problem is most evident in the former approach, though the latter is also susceptible when a large trade drops out of the trailing 24h window (i.e., when liquidity adjusts 24h + 1 block after a large trade). A simple alternative that solves the jump problem is to replenish liquidity every block at a constant rate until equilibrium is reached. This needs to be done in accordance with the constant product rule: either increase or decrease the size of the Terra pool by x Terra on every block, depending on whether it is above or below target liquidity, and simultaneously adjust the size of the Luna pool to maintain constant product. This process gradually reverts the price of Terra offered towards the peg.