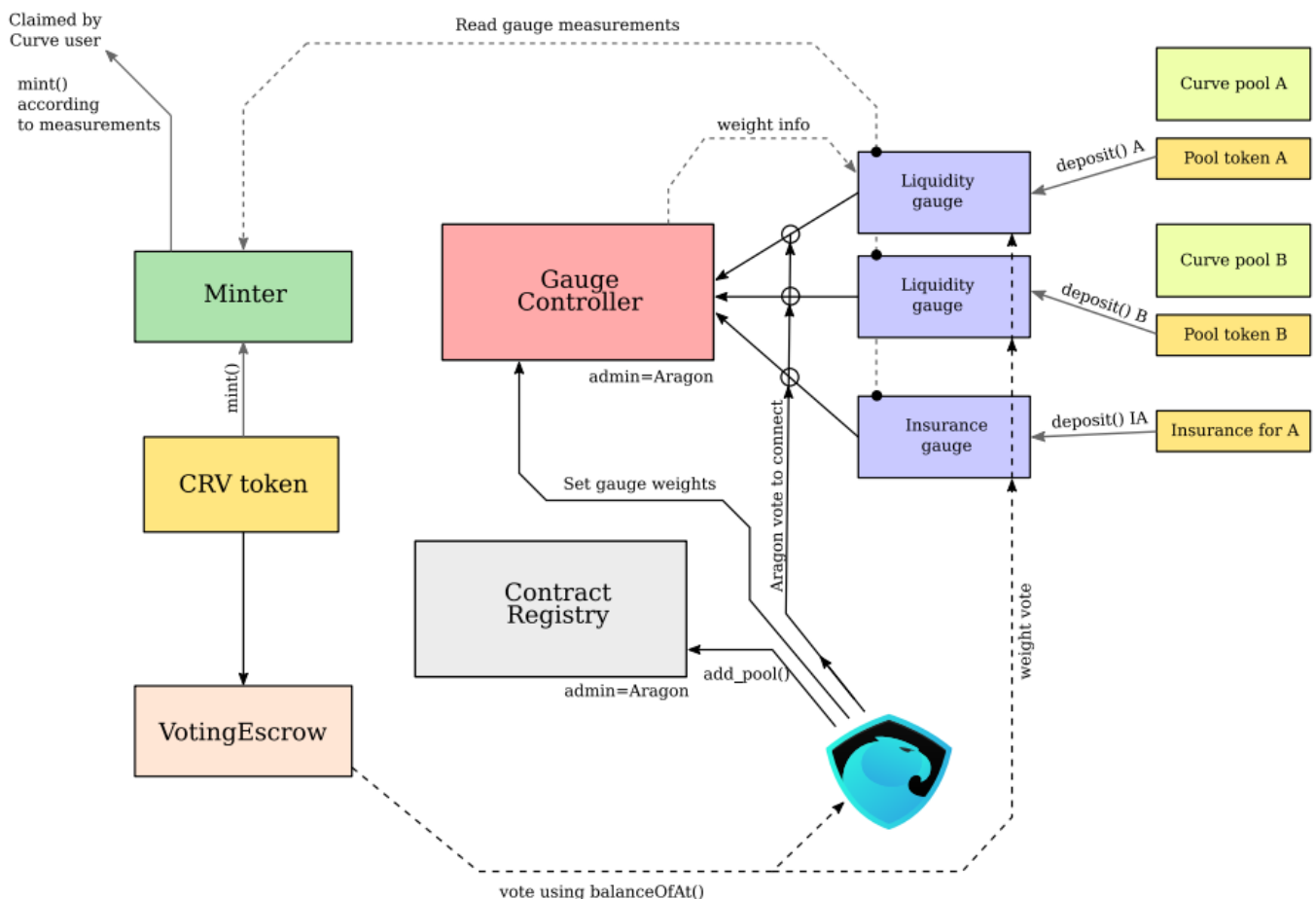


커브(Curve)

해당 번역본은 이용자가 디지털자산 관련 정보를 편리하게 확인할 수 있도록
단순 참고용으로 번역된 것이며, 투자 권유를 목적으로 하지 않습니다.
그 내용이 정확하지 않을 수 있으므로 정확한 정보 습득을 위해서는
원문을 참고하시거나 원문 작성 주체 측에 문의하시기 바랍니다.
두나무는 해당 번역본 내용의 진실성 및 정확성을 보장하지 않고,
제공된 정보에 의한 투자 결과에 대하여 어떠한 책임도 지지 않으며,
투자 손실은 투자자에게 귀속됩니다.

커브 DAO

커브 DAO는 Aragon에 의해 연결된 여러 스마트 컨트랙트로 구성되어 있다. 이와 별개로, 표준 Aragon의
“1 토큰 = 1 표” 방식은 아래에 설명될 잠금 기간에 비례한 투표 가중치 방식으로 대체된다.



커브 DAO는 거버넌스와 가치 축적을 모두 위해 사용되는 토큰 CRV를 보유하고 있다.

시간 가중 투표. VotingEscrow에 락업된 투표용 토큰

커브 DAO에서는 단순히 토큰 수량 a 로 투표하는 대신, 토큰을 *VotingEscrow*에 선택 가능한 잠금 기간 t_l 로 잠글 수 있으며, 이때 $t_l < t_{max}$ 이고 $t_{max} = 4$ 년이다. 토큰을 잠금 후에는 잠금 해제까지 남은 시간이 $t \leq t_l$ 이 된다. 투표 가중치는 다음과 같다:

$$w = a \frac{t}{t_{max}}.$$

즉, 투표는 수량뿐 아니라 시간에도 비례하며, 여기서 시간은 앞으로 토큰을 이동할 수 없는 기간을 의미한다.

토큰을 잠그는 계정은 스마트 컨트랙트일 수 없으며(거래 가능하거나 토큰화될 수 있기 때문), 널리 사용되는 다중 서명 지갑 등 화이트리스트에 등록된 스마트 컨트랙트의 경우에만 예외로 허용된다.

*VotingEscrow*는 Aragon의 Minime 토큰을 모방하려 한다. 가장 중요한 점은 `balanceOf()` / `balanceOfAt()` 및 `totalSupply()` / `totalSupplyAt()` 함수가 각각 시간 가중 투표 가중치 w 와 그 모든 가중치의 합 $W = \sum w_i$ 를 반환한다는 것이다. Aragon은 *VotingEscrow*를 일반적인 거버넌스 토큰처럼 인터페이스할 수 있다.

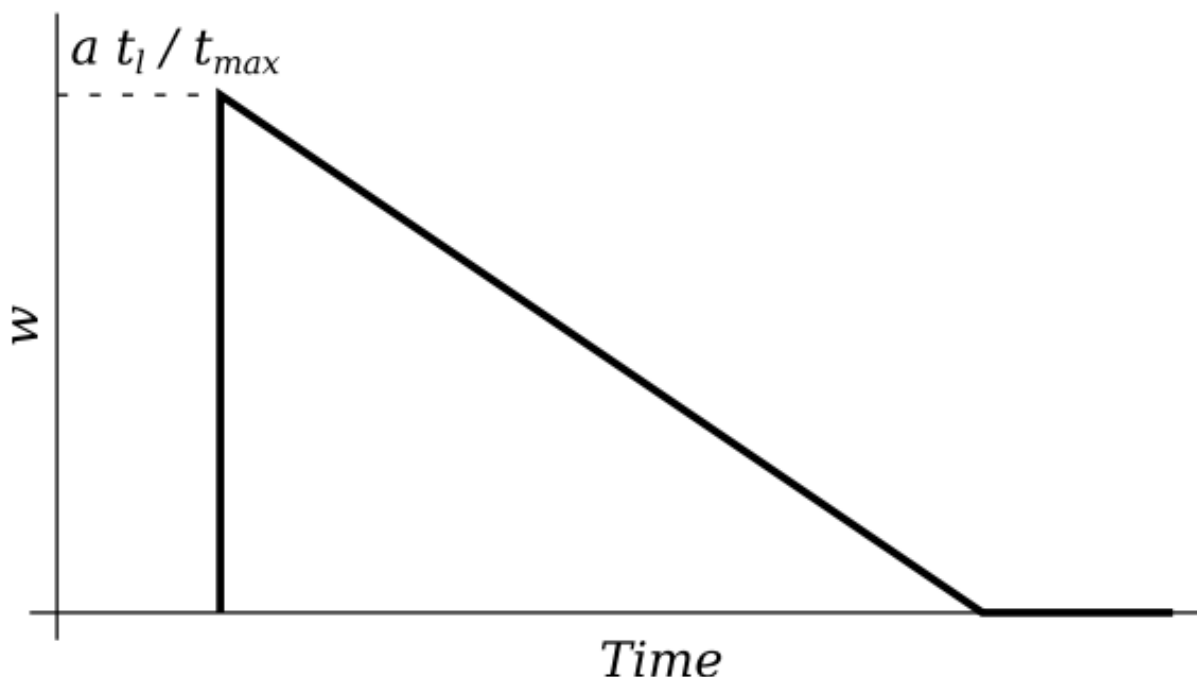


그림 2: 락업된 투표용 토큰의 투표 가중치

락은 `create_lock()`으로 생성할 수 있고, `increase_unlock_time()`으로 잠금 기간을 연장하거나 `increase_amount()`로 토큰 수량을 늘릴 수 있다. 또한 잠금이 만료되면 `withdraw()`를 통해 에스크로에서 토큰을 인출할 수 있다.

구현 세부 사항

사용자의 투표력 W_i 는 잠금 시점부터 선형적으로 감소하며, 전체 투표력 W 도 동일하게 감소한다. 주기적인 확인을 피하기 위해, 사용자가 예치하거나 인출하거나 잠금 시간을 변경할 때마다 선형 함수 $W_i(t)$ 에 대한 사용자의 기울기(*slope*)와 편향(*bias*)을 `user_point_history`에 기록한다. 전체 투표력 $W(t)$ 의 기울기와 편향 또한 변경되어 `point_history`에 기록된다. 추가로, 사용자의 잠금이 종료될 예정일 때에는 미래의 $W(t)$ 기울기 변화를 `slope_changes`에 예약한다. 각 변경 시마다 *epoch*이 1 증가한다.

이 방식으로 우리는 $W(t)$ 가 얼마나 변해야 하는지를 계산하기 위해 모든 사용자를 순회할 필요가 없으며, 사용자가 주기적으로 확인할 필요도 없다. 단, 사용자의 잠금 종료 시점은 1주 단위로 반올림하여 제한된다.

기울기와 편향은 사용자가 거버넌스 토큰을 예치하고 잠글 때, 그리고 잠금 기간이 만료될 때 모두 변경된다. 가능한 모든 만료 시점은 주 단위로 반올림되며, 이는 블록체인에서 읽어야 하는 횟수가 사용자의 수(매우 많을 수 있음)가 아닌 최대 누락된 주의 수에 비례하도록 하기 위함이다.

인플레이션 일정. ERC20CRV

ERC20CRV 토큰은 구간별 선형 인플레이션 일정을 허용하는 ERC20 토큰이다. 인플레이션은 매년 $2^{1/4}$ 의 비율로 감소한다. *Minter* 컨트랙트만이 인플레이션으로 정의된 한도 내에서 ERC20CRV를 직접 발행할 수 있다. 인플레이션이 변경될 때마다 새로운 마이닝 에폭이 시작된다.

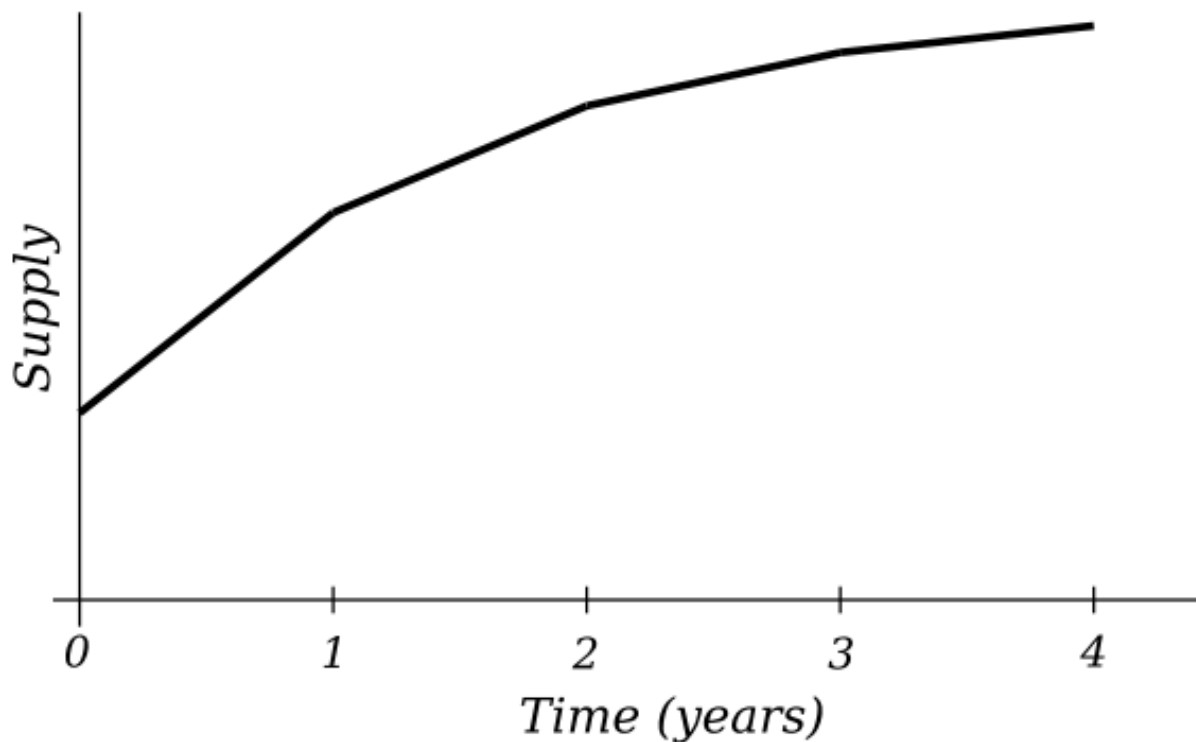


그림 3: CRV 토큰 인플레이션 일정

CRV의 초기 공급량은 12억 7,300만 개로, 이는 최종 $(t \rightarrow \infty)$ 공급량 약 30억 3,000만 개의 42%에 해당한다. 모든 초기 토큰은 블록마다 점진적으로 베스팅된다. 위 인플레이션 일정을 지원하는 초기 인플레이션율은 $r = 22.0\%$ (연간 2억 7,960만 개)이다. 모든 인플레이션은 *gauge*에서 측정된 값에 따라 Curve 사용자에게 분배된다. 첫해 동안 유통 공급량으로 유입되는 CRV의 대략적인 양은 하루 약 200만 개이며, 0에서 시작한다.

게이지 시스템. LiquidityGauge 및 GaugeController

Curve에서는 인플레이션이 실제로 Curve를 사용하는 사용자에게 분배된다. 사용량은 *Gauges*로 측정된다. 현재는 사용자가 제공하는 유동성의 양을 측정하는 *LiquidityGauge*만 존재한다. 동일한 유형의 게이지는 보험에 제공된 “유동성”을 측정하는 데에도 사용할 수 있다.

*LiquidityGauge*가 사용자의 유동성을 시간에 따라 측정하려면, 사용자는 `deposit()`을 통해 LP 토큰을 게이지에 예치하고, `withdraw()`를 통해 인출할 수 있다.

게이지가 받는 코인 비율은 현재 인플레이션율과 게이지 *유형 가중치*(Aragon에서 투표로 결정됨)에 따라 달라진다. 각 사용자는 자신이 락업한 LP 토큰의 양에 비례하여 인플레이션을 받는다. 또한 사용자가 *VotingEscrow*에 Curve 거버넌스용 토큰을 투표 락업하면 보상이 최대 2.5배까지 부스트될 수 있다.

사용자는 주기적으로 확인할 필요가 없다. 이러한 메커니즘이 어떻게 구현되는지는 기술적 세부 사항에서 설명된다.

*GaugeController*는 게이지와 그 유형의 목록을 유지하며, 각 게이지와 유형의 가중치를 관리한다.

게이지는 풀 단위로 존재하며, 각 풀마다 개별 게이지가 있다.

LiquidityGauge 구현 세부 사항

인플레이션을 r 이 각 에폭(1년)마다 변경되고, 게이지 가중치 w_g 및 게이지 유형 가중치 w_t 가 있다고 가정하자. 그러면 모든 게이지는 인플레이션 흐름을 $r' = w_g w_t r$ 의 비율로 처리하며, w_g , w_t , 또는 마이닝 에폭이 변경될 때마다 이를 업데이트할 수 있다.

사용자의 공정한 r' 몫을 계산하기 위해, 본질적으로 다음 적분을 계산해야 한다:

$$I_u = \int \frac{r'(t)b_u(t)}{S(t)} dt,$$

여기서 $b_u(t)$ 는 사용자가 제공한 잔액(LP 토큰 단위로 측정됨)이며, $S(t)$ 는 시간 t 에 따라 변하는 전체 유동성이다. 값 I_u 는 사용자가 발행받아야 할 토큰의 양을 나타낸다. 사용자의 잔액 b_u 는 사용자 u 가 예치 또는 인출할 때마다 변경되고, S 는 어느 사용자가 예치하거나 인출할 때마다 변경된다(따라서 S 는 사용자 u 의 두 이벤트 사이에도 여러 번 변할 수 있다). *LiquidityGauge* 컨트랙트에서는 I_u 의 값이 사용자별로 `integrate_fraction` 맵에 기록된다.

모든 사용자가 주기적으로 체크포인트를 수행할 필요가 없도록 하기 위해, 다음 적분의 값을 계속 기록한다 (컨트랙트에서는 `integrate_inv_supply`로 명명됨):

$$I_{is}(t) = \int_0^t \frac{r'(t)}{S(t)} dt.$$

I_{is} 의 값은 사용자가 예치하거나 인출할 때, 그리고 비율 r' 이 변경될 때(가중치 변경 또는 마이닝 에폭 변경에 의해) 기록된다.

사용자가 예치 또는 인출을 수행할 때, I_u 의 변화는 현재(사용자 행동 전)의 I_{is} 값을 사용자의 행동 전 잔액과 곱하여 계산하며, 사용자 잔액 전체에 대해 합산한다:

$$I_u(t_k) = \sum_k b_u(t_k) [I_{is}(t_k) - I_{is}(t_{k-1})].$$

사용자별 적분을 이 합으로 대체할 수 있는 이유는 $b_u(t)$ 가 t_{k-1} 과 t_k 사이의 모든 시간 동안 변하지 않기 때문이다.

사용자가 거버넌스에 참여하도록 유도하고, 동시에 유동성 유지력을 높이기 위해 다음 메커니즘을 구현한다. *LiquidityGauge*에서 계산되는 사용자의 잔액은 *VotingEscrow*에 CRV 토큰을 락업한 정도(투표 가중치 w_i)에 따라 부스트된다:

$$b_u^* = \min \left(0.4b_u + 0.6S \frac{w_i}{W}, b_u \right).$$

w_i 의 값은 사용자가 어떤 행동(예치, 인출, 발행된 CRV 토큰 인출)을 수행하는 시점에서 측정되며, 이 사용자가 다음 행동을 수행할 때까지 적용된다.

사용자들이 CRV를 투표 락업하지 않거나(혹은 단순히 보유하지 않은 경우), 인플레이션은 각 사용자가 제공한 유동성 b_u 에 비례하여 단순히 분배된다. 그러나 사용자가 충분한 양의 CRV를 스테이킹하면, 자신의 CRV 보상을 최대 2.5배까지 부스트할 수 있으며, 이는 해당 행동을 하지 않는 사용자들의 몫을 약간 줄이는 효과를 낸다.

구현 세부 사항상, 사용자는 마지막 행동 또는 체크포인트 시점에서 실제 부스트를 적용받는다. 투표력이 시간이 지남에 따라 감소하기 때문에, 사용자는 부스트를 적용한 뒤 추가 행동을 하지 않고 있다가 더 많은 토큰을 투표 락업하는 것이 유리하다. 그러나 투표 락업이 만료되면, 다른 모든 사용자가 그 사용자를 위해 체크포인트를 생성함으로써 그 사용자를 “킥(kick)”할 수 있으며, 이때 사용자가 이미 투표력을 잃었다면 부스트는 사실상 초기화된다.

마지막으로, 게이지는 인플레이션의 전체 1년치를 놓치지 않도록 설계되어 있다(예: 1년 동안 게이지에 어떤 상호작용도 없을 경우). 만약 이런 일이 발생한다면, 해당 방치된 게이지는 더 적은 CRV를 받게 된다.

게이지에 대한 가중치 투표

Aragon에서 단순히 가중치 변경에 투표하는 대신, 사용자는 자신이 투표 락업한 토큰을 특정 *게이지*(풀)에 할당할 수 있다. 해당 풀은 자신에게 할당된 투표 락업 토큰의 양에 비례하여 발행되는 CRV 토큰의 일부를 받게 된다. *VotingEscrow*에 토큰을 보유한 각 사용자는 언제든지 자신의 선호도를 변경할 수 있다.

사용자가 새로운 가중치 투표를 적용하면, 그 투표는 다음 주의 첫 번째 날(정확히 한 주의 시작 시점)에만 반영된다(이는 확장성 문제를 고려한 설계이다). 동일한 게이지에 대한 가중치 투표는 10일에 한 번보다 자주 변경할 수 없다.

GaugeController 구현 세부 사항

가중치 투표를 구현하기 위해 *GaugeController*는 각 사용자가 가진 투표력의 선형 특성을 처리할 수 있는 매개변수를 포함해야 한다.

*VotingEscrow*에서의 구현 방식과 유사하게, *GaugeController*는 각 게이지별로 `vote_points`에 포인트(편향 `bias` + 기울기 `slope`)를 기록하고, `vote_bias_changes` 및 `vote_slope_changes`에 해당 포인트의 편향 및 기울기 변경을 예약한다. 이러한 변경은 매 라운드 주 단위로 발생하며, 각 사용자별 게이지에 대한 현재 기울기는 `vote_user_slopes`에 저장된다. 이와 함께 사용자가 사용한 투표력과 그 투표 락업이 종료되는 시점도 기록된다. 게이지별 투표 가중치에 대한 기울기와 편향의 총합, 그리고 유형별 합계는 다음 주에 대해 예약기록되며, 일부 사용자의 락업 만료 시 투표력이 0이 되는 시점 또한 함께 관리된다.

사용자가 자신의 선호도를 변경하면, 게이지 가중치의 변경은 즉시 적용되지 않고 다음 주 라운드에 예약된다. 이는 각 사용자가 수행해야 하는 블록체인 읽기의 수를 줄이기 위한 조치로, 다른 사용자의 상호작용 횟수가 아니라 마지막 변경 이후 지난 주의 수에 비례하게 된다.

*GaugeController*는 시스템의 핵심 구성 요소 중 하나이므로, 반드시 DAO에 의해 제어되어야 한다. 중앙화된 관리자가 이를 통제해서는 안 되며, 누구에게도 유형 가중치를 일방적으로 변경할 권한이 주어져서는 안 된다.

수수료 소각기 (Fee burner)

모든 풀은 관리자가 `withdraw_admin_fees`를 사용하여 수수료를 수집할 수 있도록 허용한다. Aragon은 관리 계정으로 이러한 수수료를 수집하고, 자유 시장이 존재하게 되면 해당 수수료를 사용해 CRV를 매수 및 소각할 수 있어야 한다. 이러한 작업은 투표 없이도 누구나 수행할 수 있어야 한다.

소각 대신 동일한 인터페이스를 사용하는 다른 메커니즘이 작동할 수도 있다. 어떤 경우이든, 이 기능은 즉시 적용되지 않는다.

거래량 및 거버넌스 투표 보상을 위한 게이지

투표와 거래는 모두 이산적(discrete) 이벤트이므로, 동일한 유형의 게이지를 사용할 수 있다. 그 개념은 각 이벤트가 시간이 지남에 따라 지속적으로 감소하는 가중치를 가진다는 것이다.

사용자가 Aragon에서 투표할 때마다 게이지 컨트랙트를 호출할 수 있어야 한다.

StableSwap - 안정적인 코인 유동성을 위한 효율적 메커니즘

Michael Egorov

2019년 11월 10일

초록

StableSwap은 “레버리지가 적용된 Uniswap”이라 부를 수 있는 방식으로 스테이블코인 간의 교차 시장을 생성하는 메커니즘을 제공한다. 이는 매우 작은 가격 슬리피지를 가진 완전 자율적 스테이블코인 마켓메이커이며, 동시에 유동성 공급자에게는 효율적인 “법정화폐형 저축 계좌” 역할을 한다.

본 문서는 세부 사항(특히 StableSwap 불변식)에 대한 모든 내용을 포함하지 않은 간략 버전이다.

서론

스테이블코인은 최근 매우 인기를 끌고 있다. 중앙화된 USDC, USDT, BUSD, PAX, TrueUSD뿐만 아니라 탈중앙화된 DAI도 포함된다. 그러나 이들(특히 탈중앙화된 스테이블코인)은 가격 안정성과 유동성 문제를 겪고 있다. 이는 DeFi 차익거래(Arbitrage)에 특히 큰 부담으로 작용한다. 예를 들어, MakerDAO가 안정화 수수료를 5.5%로 낮췄을 때, Compound의 사용자 중 다수(당시 이자율 11%)는 여전히 Compound에 머물렀다. 그 이유는 DAI로 대출을 받았으며, DAI와 USDC 간 변환이 매우 비용이 많이 드는 작업이었기 때문이다.

한편, 많은 DeFi 사용자는 5% APR의 수익을 얻기 위해 스테이블코인을 대출에 활용하고자 한다. 이는 전통 금융권이 제공하는 금리보다 훨씬 높기 때문이다. 그러나 이들은 “이익을 보장한다”고 주장하는 트레이딩 회사에 동일한 자금을 맡기는 것에는 불안함을 느낀다.

이 연구에서는 *StableSwap* — 스테이블코인을 위한 자동화 유동성 공급자를 제안한다. 수요 측면에서는 매우 낮은 가격 슬리피지를 가진 Uniswap 유사 자동화 거래소를 제공하며(보통 100배 이상 작다), 공급 측면에서는 다중 스테이블코인 기반 “저축 계좌”를 제공한다. 시뮬레이션에 따르면, 트레이더가 지난 반년 간의 거래량 및 스테이블코인 가격 데이터를 기준으로 스마트 컨트랙트와 기존 거래소 간의 차익거래를 수행한다고 가정할 때, 최대 300% APR을 달성할 수 있다. 이 과정에서 거래를 책임지는 중개자(예: 거래소 운영자, 오더북, 인간 마켓메이커)는 존재하지 않는다.

작동 원리

먼저, 일정한 가격을 유지하는 유동성 공급자를 상상해보자. 두 개의 코인 $\$X\$$ 와 $\$Y\$$ 가 있을 때, 예를 들어 코인 $\$X\$$ 를 $\$dx\$$ 만큼 판매하면 코인 $\$Y\$$ 를 $\$-dy = dx\$$ 만큼 구매하게 된다. 이는 다음과 같은 “선형” 불변식을 통해 임의의 개수의 코인 $\$X_i\$$ 로 일반화할 수 있다:

$$\sum x_i = \text{const.}$$

가격은 $\$-dx_i/dx_j\$$ 로 결정되며, 이 경우 항상 정확히 1이 된다. 하지만 이러한 방식은 시장이 변동할 때 지속적인 가격 조정 없이는 작동하지 않는다. 가격 오라클을 통해 이를 수행할 수 있으나, 이는 위험하며 탈중앙화 정도가 낮다. 더 나은 방법이 존재한다.

Uniswap, Bancor, Kyber와 같은 프로토콜은 본질적으로 변동성이 크고 가격이 불안정한 시장에서 작동하므로, 다른 접근법을 사용한다. 이들은 가격을 조정하여 “포트폴리오”(보통 두 개의 코인으로 구성됨)가 균형을 이루도록 만든다. 즉, 유동성 풀 내에서 코인 $\$X\$$ 와 $\$Y\$$ 의 가치를 동일한 통화 단위로 환산했을 때 동일하게 유지되도록 한다. 이러한 조건은 유동성 풀 내 코인 수량의 곱이 일정하게 유지될 때 자동으로 달성된다:

$$xy = \text{const.}$$

더 나아가, Balancer에서 훌륭하게 구현된 것처럼, 이 불변식을 임의의 비율을 가진 다수의 코인으로 일반화할 수도 있다:

$$\prod x_i^{w_i} = \text{const.}$$

이 방식은 ETH나 일반 토큰과 같은 자산에는 적합하지만, 안정적이어서 하는 자산에는 적절하지 않다. 문제는 가격 슬리피지가 매우 크며, 의미 있는 유동성을 유지하려면 막대한 자금이 필요하다는 것이다. 반대로, 예를 들어 DAI와 USDC를 Uniswap의 유동성 풀에 예치하면 수익률은 매우 작으며(연간 수 퍼센트 수준) 그친다.

StableSwap에서는 중간 수준의 불변식이 발견되었다(그림 1). 예상대로, 가격(미분값과 동일)은 코인의 수량이 균형에 가까울 때 1에서 아주 약간만 벗어난다.

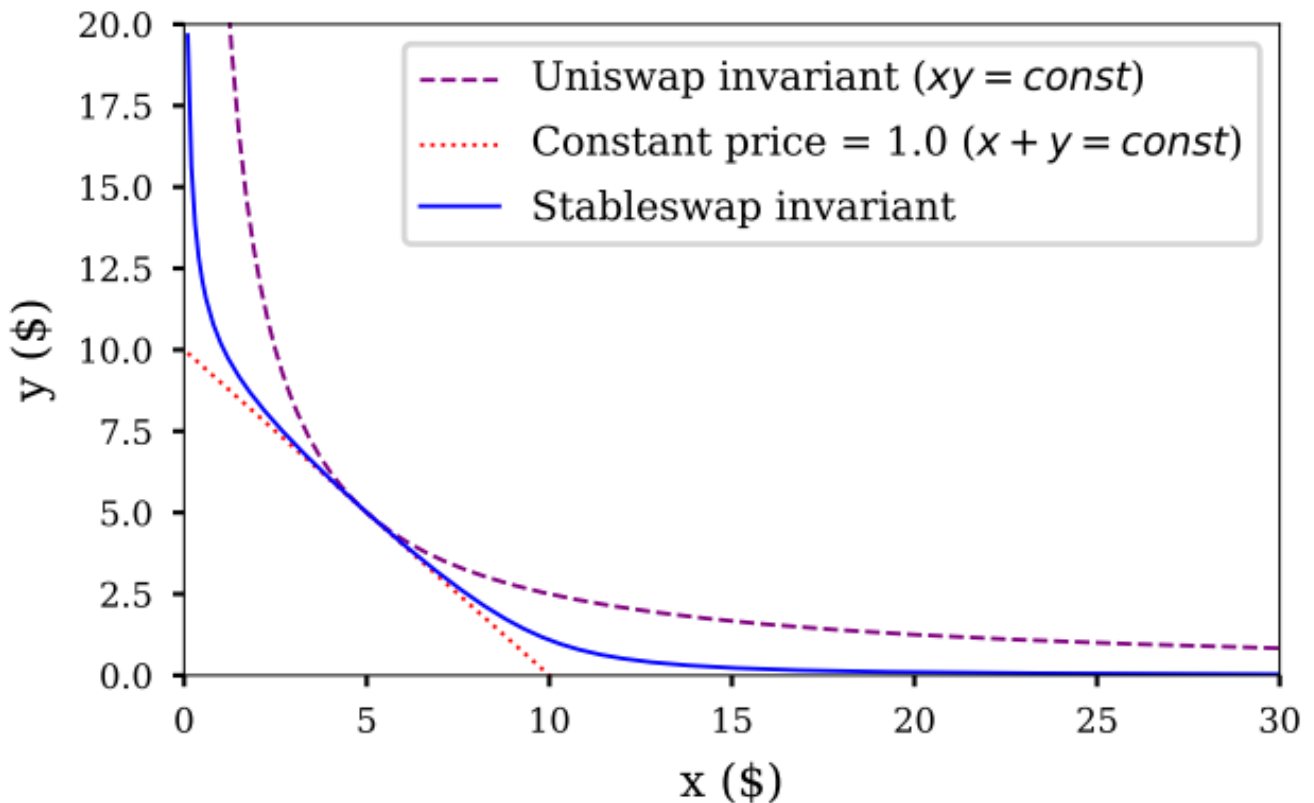


그림 1: StableSwap 불변식과 Uniswap(상수곱) 및 상수 가격 불변식의 비교. 포트폴리오는 “이상적인” 가격 1.0을 가진 코인 $\$X\$$ 와 $\$Y\$$ 로 구성되어 있다. 초기에는 $\$x = 5\$$, $\$y = 5\$$ 코인이 로드되어 있다. $\$x\$$ 가 감소함에 따라 $\$y\$$ 가 증가하며, 가격은 $\$dy/dx\$$ 의 미분값으로 정의된다.

상수곱 불변식과 비교할 때 가격 슬리피지(그림 2)는 훨씬 작다.

StableSwap 불변식에는 “증폭 계수”(amplification coefficient) 매개변수가 존재하며, 그 값이 낮을수록 불변식은 상수곱 형태에 가까워진다. 슬리피지를 계산할 때는 실용적인 값 $\$A = 100\$$ 을 사용한다. 이는 Uniswap을 100배 레버리지로 사용하는 것과 어느 정도 유사하다.

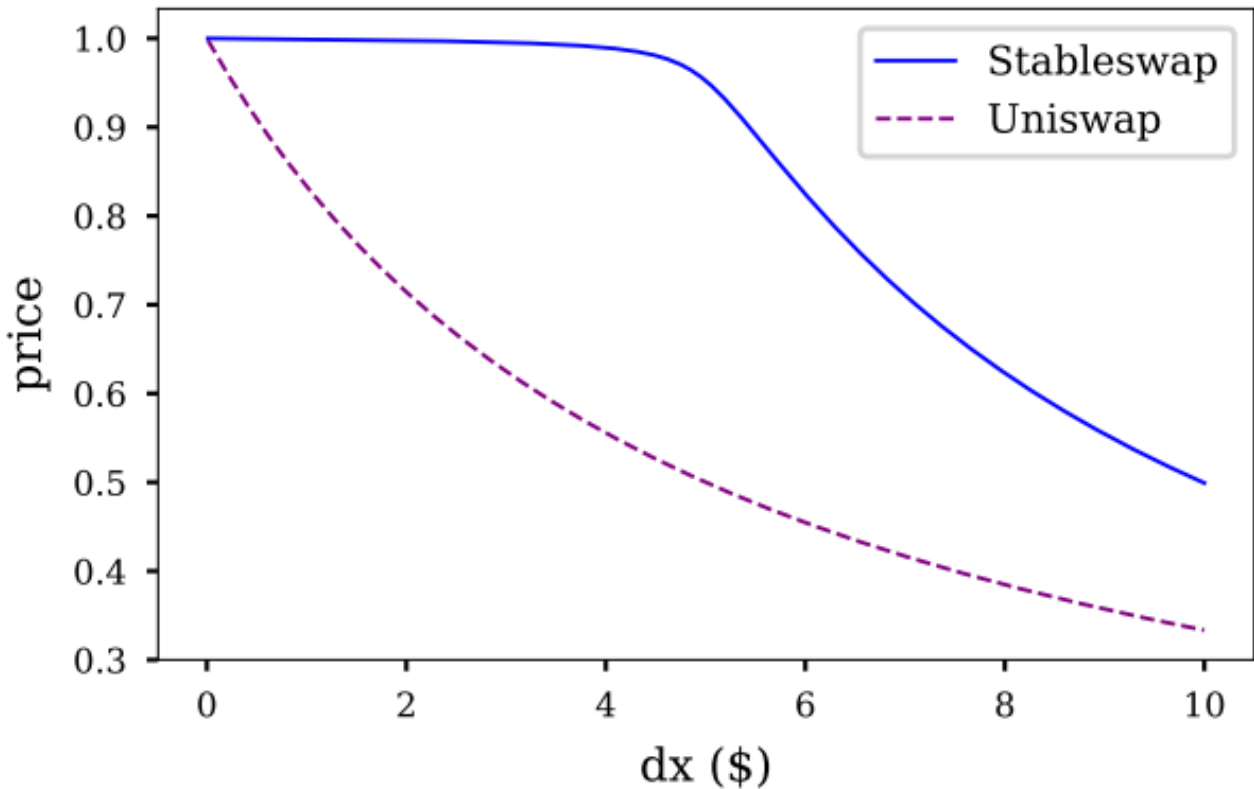


그림 2: 가격 슬리피지 비교: Uniswap 불변식(점선) vs StableSwap(실선)

가격이 균형점(1.0)에서 벗어나 보이더라도, 불변식은 비최적 지점에서 작동하기 시작하지만 여전히 일정 수준의 유동성을 제공한다(대부분의 경우 최적의 $\$A\$$ 가 정확히 설정되었다면 상수곱 불변식보다 더 큰 유동성을 제공한다). 어떤 가격에서도 이 불변식은 상수곱 불변식과 마찬가지로 일정 유동성을 제공하며, 상수합 불변식과는 달리 유동성이 완전히 사라지지 않는다.

StableSwap 불변식의 구성

그림 1에서 보듯, 상수 가격 불변식은 직선(또는 두 개 이상의 코인이 있을 경우 초평면)을 형성한다. 상수곱 불변식은 쌍곡선을 형성한다.

가격은 그래프상의 기울기로 정의된다. 우리는 균형 근처에서 비교적 평평한(즉, 가격 변화가 느리고 그래프가 직선에 가깝거나 “확대된” 쌍곡선 형태인) 불변식을 찾고자 한다. 그러나 포트폴리오가 불균형해질수록(예: 축에 가까워질수록) 상수곱 불변식에 가까워지도록 해야 한다.

다음은 $\$i\$$ 로 인덱싱된 $\$n\$$ 개의 코인에 대해 일반화된 상수합(상수 가격) 및 상수곱 불변식이다:

$$\sum x_i = D;$$

$$\prod x_i = \left(\frac{D}{n}\right)^n.$$

상수 χD 는 모든 코인의 가격이 동일할 때의 총 코인 수량을 의미한다.

이제 “증폭된” 불변식을 상상해보자. 낮은 가격 슬리피지를 가지려면 곡률이 작아야 한다. “제로 슬리피지” 불변식은 무한대의 레버리지에 해당한다. 그러나 제로 슬리피지 불변식은 상수 가격, 즉 상수합 불변식이다! 따라서 상수곱이 “제로 레버리지”, 상수합이 “무한 레버리지”라고 가정하면, 그 중간을 구성할 수 있다. 레버리지를 χ 로 나타내자. 상수합 불변식을 χ 배하고 상수곱 불변식에 더하면, $\chi = 0$ 일 때 상수곱 불변식이 되고, $\chi = \infty$ 일 때 상수합 불변식이 된다. 우리가 원하는 성질이다. 단, χ 는 코인 수에 의존하지 않는 무차원 파라미터여야 한다.

따라서 상수합 불변식을 χD^{n-1} 로 곱하고 두 번째 불변식에 더하면 다음과 같다:

$$\chi D^{n-1} \sum x_i + \prod x_i = \chi D^n + \left(\frac{D}{n}\right)^n.$$

이 식이 항상 성립하면, 우리는 레버리지 χ 를 가진 거래를 수행할 수 있다. 그러나 이 식은 가격이 이상적인 1.0에서 멀어질 때를 지원하지 못한다. 불변식은 어떤 가격에서도 작동해야 하므로(항상 일정한 유동성을 보장하기 위해) χ 를 동적으로 만들어야 한다.

이를 위해 포트폴리오가 완전한 균형 상태에 있을 때 χ 는 상수 A 와 같으며, 균형에서 멀어질수록 0으로 감소한다:

$$\chi = \frac{A \prod x_i}{(D/n)^n}.$$

이를 위의 “레버리지 적용” 불변식에 대입하면 StableSwap 불변식을 얻는다:

$$A n^n \sum x_i + D = A D n^n + \frac{D^{n+1}}{n^n \prod x_i}.$$

코인 포트폴리오 $\{x_i\}$ 가 주어졌을 때, 우리는 χD 를 계산해야 하며, 거래(예: x_i 를 x_j 로 교환)를 수행할 때 이 식이 항상 성립해야 한다. 이를 위해 모든 다른 변수가 주어졌을 때 χD 나 x_j 에 대해 반복적이고 수렴하는 해를 찾는다.

시뮬레이션 및 성능

이 알고리즘의 성능은 3개의 스테이블코인(DAI, USDC, USDT)에 대한 유동성 제공을 가정하고 평가 및 최적화되었다. 가격 데이터는 6개월간(2019년 5월~10월) Coinbase Pro(DAI/USDC), Binance(USDC/USDT), HitBtc(USDT/DAI)에서 가져왔다. 시뮬레이션에서는 컨트랙트 내 총 유동성을 \$30,000으로 가정하였다. 거래는 충분한 가격 변동이 있을 때만 수행되었다. 그 결과는 다음과 같다:

- 최적 “증폭 계수”(“레버리지”): $A = 85$
- 최적 수수료: 거래당 0.06%
- 최적 매개변수에서의 유동성 공급자 수익률: 연 312% (APR)

구현

다중 스테이블코인 컨트랙트는 Vyper로 구현되었다. StableSwap 불변식을 사용하는 방정식의 해는 스마트 컨트랙트 내부에서 반복적 방법으로 계산되며, 정수 연산만을 사용한다. 브라우저 UI(그림 3)는 순수 클라이언트 사이드 자바스크립트로 구현되었다.

기타 응용

스테이블코인 유동성 외에도, 동일한 방식은 이자 발생 자산(cDAI)이나 스테이킹 가능한 암호화폐의 토큰화된 스테이크에 유동성을 제공하는 데 적용할 수 있다. 필자의 견해로, 이 방법은 미래 DeFi 인프라의 중요한 구성 요소가 될 것이다.

이 방법을 스테이블코인에 적용하면 실전 검증이 가능해지고, 탈중앙화(비수탁형) 스테이블코인의 사용성을 높일 수 있다.

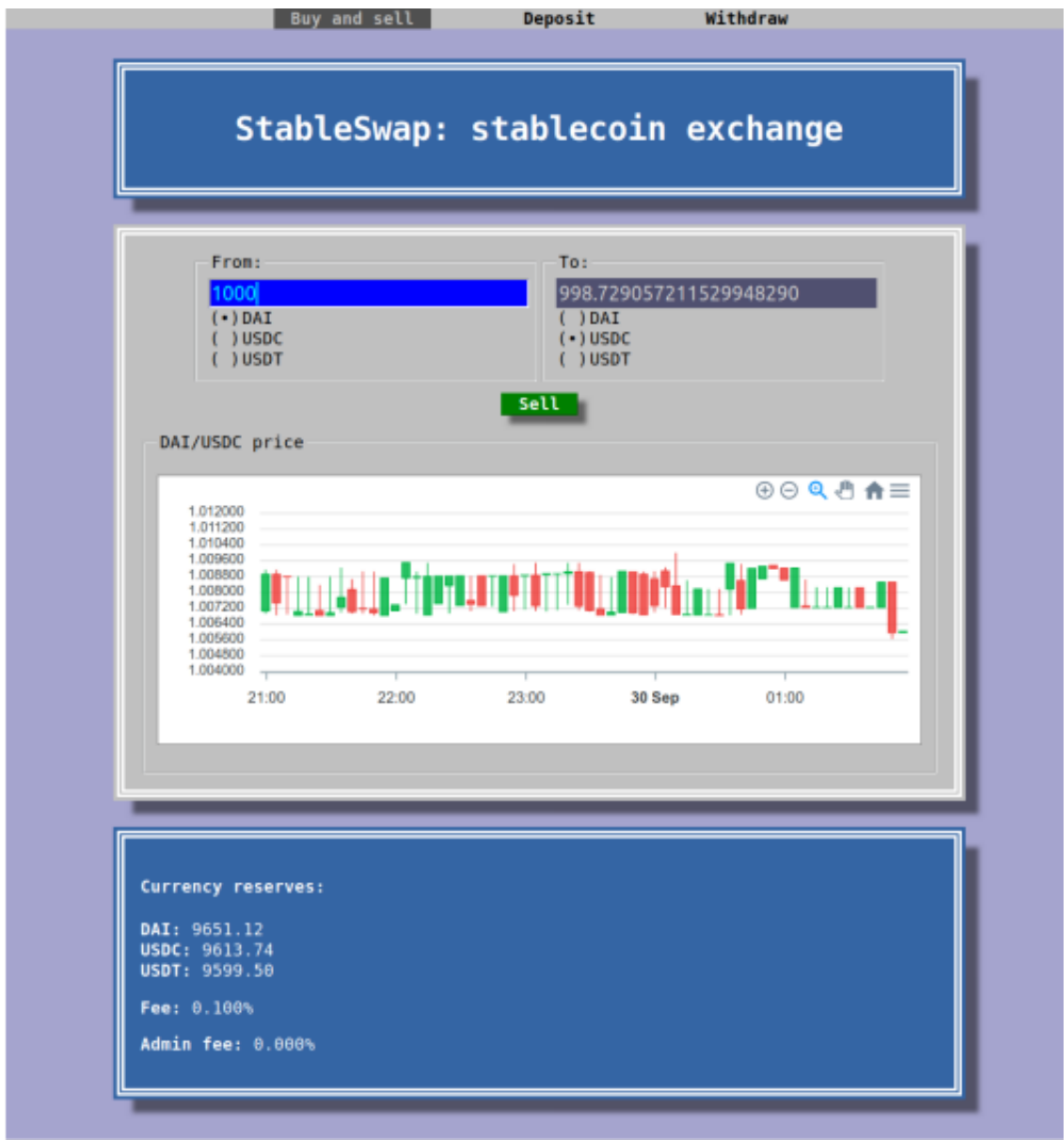


그림 3: StableSwap 사용자 인터페이스(UI)

동적 페그(Dynamic Peg)를 이용한 자동 시장 조성

Michael Egorov, Curve Finance (Swiss Stake GmbH)

2021년 6월 9일

서론

자동 시장 조성자(AMM, Automatic Market-Makers)는 탈중앙화 금융이 가져온 주요 혁신 중 하나이다. 먼저 Uniswap은 기초 자산의 가격에 대한 어떠한 가정도 하지 않고 유동성을 모든 가격 구간에 균등하게 분포시키는 $x \cdot y = k$ 불변식을 기반으로 시장을 도입했다. 이후 우리는 대부분의 유동성을 가격 1.0(또는 임의의 특정 가격) 주변에 집중시킬 수 있도록 하는 stableswap 불변식을 도입했으며, 이는 스테이블코인 간 유동성을 형성하는 데 매우 유용한 기능이였다.

본 연구에서는 $x \cdot y = k$ 불변식보다 효율적으로 서로 페그되지 않은 자산 간의 유동성을 생성하는 방법을 제시한다. 우리는 현재의 “내부 오라클(internal oracle)” 가격을 중심으로 유동성을 집중시키되, 시스템이 얻는 이익이 손실보다 클 때에만 그 가격을 이동시킨다. 이 방식은 Uniswap 불변식보다 5~10배 높은 유동성을 제공하며, 유동성 공급자에게 더 큰 수익을 제공한다.

이 방법의 일반적 원리, 선택된 불변식, 적용 한계, 그리고 과거 가격 데이터를 기반으로 한 시뮬레이션 결과를 기술한다.

변환된 페그 불변식

가격 공간을 변환하여 모든 가격이 어떤 목표값으로부터 1에 가까운 영역으로 변환되도록 하자. 이 변환을 $T()$ 라 하며, 이는 이미 curve.fi의 compound 및 y 풀에서 사용되고 있다. 스마트 컨트랙트 내 잔액 벡터를 $\mathbf{b} = (b_0, b_1, \dots)$ 로 정의하고, 여기서 b_i 는 풀에서 지원되는 i 번째 토큰의 잔액이다. 컨트랙트는 코드 상에서 *price_scale*이라 불리는 내부 계수 집합 $\mathbf{p} = (p_0, p_1, \dots)$ 를 유지한다. 실제 잔액 $\mathbf{b}$ 와 변환된 잔액 $\mathbf{b}'$ 는 다음과 같이 상호 변환할 수 있다:

$$\mathbf{b} = T(\mathbf{b}', \mathbf{p}) = (b'_0 p_0, b'_1 p_1, \dots);$$

$$\mathbf{b}' = T^{-1}(\mathbf{b}, \mathbf{p}) = \left(\frac{b_0}{p_0}, \frac{b_1}{p_1}, \dots \right).$$

불변식은 다음과 같이 정의되는 초곡면(차원이 2일 경우 곡선)으로 표현할 수 있다:

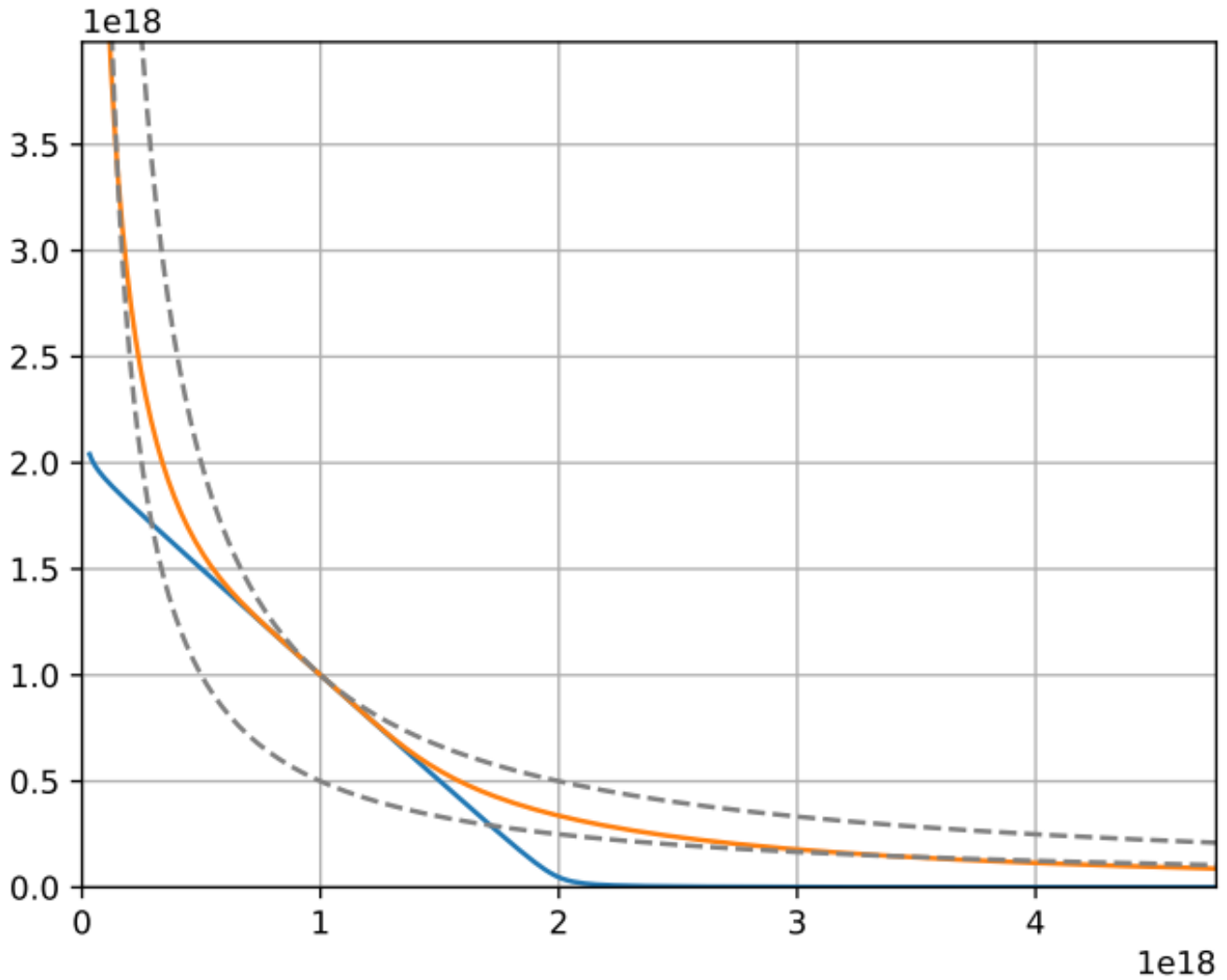


그림 1: AMM 불변식 비교: 상수곱(점선), stableswap(파란색), 본 연구에서 제시한 불변식(주황색)

$$I(\mathbf{b}') = 0.$$

불변식 함수는 $p_0 = 1$ 이 되도록 선택하는 것이 편리하다(예: 0번째 자산이 USD이고, 모든 가격이 USD 기준으로 측정된다고 가정). 우리가 고려하는 불변식들은 다음 조건을 만족하도록 구성된다:

$$\forall x : I(x, x, \dots) = 0.$$

이러한 불변식의 한 예는 stableswap 불변식이며, 또 다른 예는 본 문서에서 이후 논의될 불변식이다. 두 불변식 모두 그림 1에 표시되어 있다.

특정 곡선은 풀 내 예치금에 따라 달라진다. 동일한 곡선에서 상수로 유지되는 불변식 D 를 정의하는 것이 편리하며, 이는 풀의 예치금이 균형 상태일 때의 총 예치금과 같다:

$$\mathbf{x}_{eq} = (x_{eq}, x_{eq}, \dots),$$

$$I(\mathbf{x}_{eq}, D) = 0,$$

$$D = Nx_{eq}.$$

D 는 곡선을 매개화하는 역할을 하므로, 균형점 x_{eq} (벡터 $\mathbf{p}$ 가 페그되는 지점)를 구하는 것이 간단해진다. 여기서 N 은 코인의 개수이다.

리페깅 손실의 정량화

이익이나 손실을 정량화하기 위해서는 견고한 수익 측정 기준이 필요하다. 이를 위해 우리는 *균형점* (equilibrium point)에서의 상수곱 불변식(constant-product invariant)의 값을 선택한다. 계산의 편의를 위해 손실/이익 함수를 1제곱(balance power of 1) 형태로 변환한다. 그 결과, 노이즈 없이 포트폴리오의 가치를 정량화하는 함수는 다음과 같다:

$$X_{cp} = \left(\prod \frac{D}{Np_i} \right)^{\frac{1}{N}}.$$

$\mathbf{p}$ 가 변경되면 가격 페그(price peg)는 바뀌지만 잔액은 변하지 않는다. 새로운 $\mathbf{b}'$ 값에 대해 새로운 D 를 계산하고, 그 D 와 p_i 를 대입하여 X_{cp} 를 계산할 수 있다. 우리는 X_{cp} 의 감소를 허용하지만, 그 손실이 지금까지 얻은 수익의 절반을 초과하지 않도록 제한한다(이는 X_{cp} 의 증가를 추적함으로써 측정된다).

CurveCrypto 불변식

여기서 사용하는 불변식은 stableswap 불변식에서 영감을 받았다:

$$KD^{N-1} \sum x_i + \prod x_i = KD^N + \left(\frac{D}{N} \right)^N,$$

그러나 K 의 정의는 약간 다르다:

$$K_0 = \frac{\prod x_i N^N}{D^N}, \quad K = AK_0 \frac{\gamma^2}{(\gamma + 1 - K_0)^2},$$

여기서 A 는 증폭 계수(amplification coefficient)이고, $\gamma > 0$ (보통 작은 값)은 그림 1에서 점선 두 개 사이의 거리 의미를 가진다. 이 불변식은 상수곱 불변식과 stableswap 불변식의 중첩(superposition)으로 근사적으로 작동한다.

이 불변식을 x_j 또는 D 에 대해 풀기 위해서는 다음 형태로 정의할 필요가 있다: $F(\mathbf{x}, D) = 0$

$$F(\mathbf{x}, D) = K(\mathbf{x}, D)D^{N-1} \sum x_i + \prod x_i - K(\mathbf{x}, D)D^N - \left(\frac{D}{N} \right)^N.$$

스왑 알고리즘은 먼저 $F(\mathbf{x}, D) = 0$ 식을 D 에 대해 풀 다음, 코인 x_i 가 교환으로 인해 증가한 만큼의 금액을 반영한 후 x_i 가 주어졌을 때 x_j 를 계산한다(이는 stablecoin용 curve/stableswap 알고리즘과 동일하다). 이를 위해 우리는 뉴턴 방법(Newton's method)을 사용한다. 먼저 다음과 같이 D 를 반복 계산한다:

$$D_{k+1} = D_k - \frac{F(\mathbf{x}, D_k)}{F'_D(\mathbf{x}, D_k)}$$

그리고 다음과 같이 x_i 를 계산한다:

$$x_{i,k+1} = x_{i,k} - \frac{F(x_{i,k}, \dots, D)}{F'_{x_i}(x_{i,k}, \dots, D)}.$$

모든 로직은 EVM 상에 구현되어 있으며, 각 계산은 최적화되어 있다. 따라서 D 나 x_i 의 해를 찾는 데 약 35,000 가스(gas)가 소모된다.

이 함수는 단조(monotonic)가 아니기 때문에, 뉴턴 방법의 정확한 수렴을 위해 초기값 설정이 매우 중요하다. 우리는 다음의 초기값이 가장 좋은 결과를 준다는 것을 발견했다:

$$D_0 = N \left(\prod x_k \right)^{\frac{1}{N}},$$

$$x_{i,0} = \frac{D^{N-1}}{\prod_{k \neq i} x_k N^{N-1}}.$$

초기값 설정 외에도, 우리는 퍼징(fuzzing, Hypothesis 프레임워크 사용)을 통해 위 수학적 모델의 적용 한계를 결정하였다. 그 결과, x_i 의 수렴을 위한 안전한 값은 다음과 같다 (모든 비정수 값은 10^{18} 단위를 기준으로 함):

- $0.1 \leq D \leq 10^{15}$ [USD]
- $5 \cdot 10^{-3} < x_i / D < 200$
- $10^{-8} \leq \gamma \leq 10^{-2}$ (일반적으로 10^{-4})

D 의 수렴을 위한 값은 다음과 같다:

- $1 \leq A \leq 10000$
- $10^{-9} \leq x_0 \leq 10^{15}$
- $10^{-5} \leq x_i / x_0 \leq 10^5$
- $10^{-8} \leq \gamma \leq 10^{-2}$

리페킹 알고리즘

우리는 각 교환 또는 예치 시점마다 X_{cp} 를 추적한다. 각 연산 후 xcp_profit 변수를 $X_{cp,after} / X_{cp,before}$ 로 곱하여 업데이트하며, 초기값은 1.0이다. 또한 $\mathbf{p}$ 조정 후의 손실을 추적하기 위해 xcp_profit_real 변수를 유지한다. 만약 $\mathbf{p}$ 조정으로 인해 $xcp_profit_real - 1$ 이 $xcp_profit - 1$ 의 절반보다 낮아진다면, 해당 조정을 되돌린다.

내부적으로는 N 차원 가격 공간에서 지수 이동 평균(EMA, Exponential Moving Average)을 적용한 가격 오라클을 사용한다. 마지막 보고된 가격이 $\mathbf{p}_{last}$ 이고, 업데이트가 t 초 전에 이루어졌으며 EMA의 반감기가 $T_{1/2}$ 일 경우, 오라클 가격 $\mathbf{p}^*$ 는 다음과 같이 계산된다:

$$\alpha = 2^{-\frac{t}{T_{1/2}}},$$
$$\mathbf{p}^* = \mathbf{p}_{last}(1 - \alpha) + \alpha \mathbf{p}_{prev}^*.$$

우리는 로그 공간에서 가격 벡터를 $\mathbf{p}^\wedge$ 방향으로 조정하지만, 계산 단순화를 위해 근사치를 사용한다. 따라서 조정 방향은 이상적인 $(\mathbf{p}^\wedge - \mathbf{p})$ 벡터와 약간 다를 수 있으며, 상대 가격 변화 스텝을 s 로 정의할 때, i 번째 구성 요소에 대한 관계는 다음과 같다:

$$\frac{p_i}{p_{i,prev}} = 1 + \frac{s}{\sqrt{\sum \left(\frac{p_j^*}{p_{j,prev}} - 1 \right)^2}} \left(\frac{p_i^*}{p_{i,prev}} - 1 \right).$$

동적 수수료(Dynamic Fees)

우리는 f_{mid} 에서 f_{out} 범위의 수수료 f 를 사용하며, 이는 다음과 같이 정의된다:

$$g = \frac{\gamma_{fee}}{\gamma_{fee} + 1 - \frac{\prod x_i}{(\sum x_i / N)^N}},$$
$$f = g \cdot f_{mid} + (1 - g) \cdot f_{out}.$$

커브 스테이블코인 설계

Michael Egorov, Curve Finance

2022년 10월 9일

개요

이 스테이블코인의 설계는 여러 개념으로 구성된다. 그중 가장 중요한 것은 대출-청산 AMM 알고리즘 (LLAMMA), PegKeeper, 그리고 통화정책(Monetary Policy)이다. 그러나 핵심 아이디어는 LLAMMA에 있으며, 이는 청산(liquidation) 과정을 특수 목적의 AMM으로 대체하는 것이다.

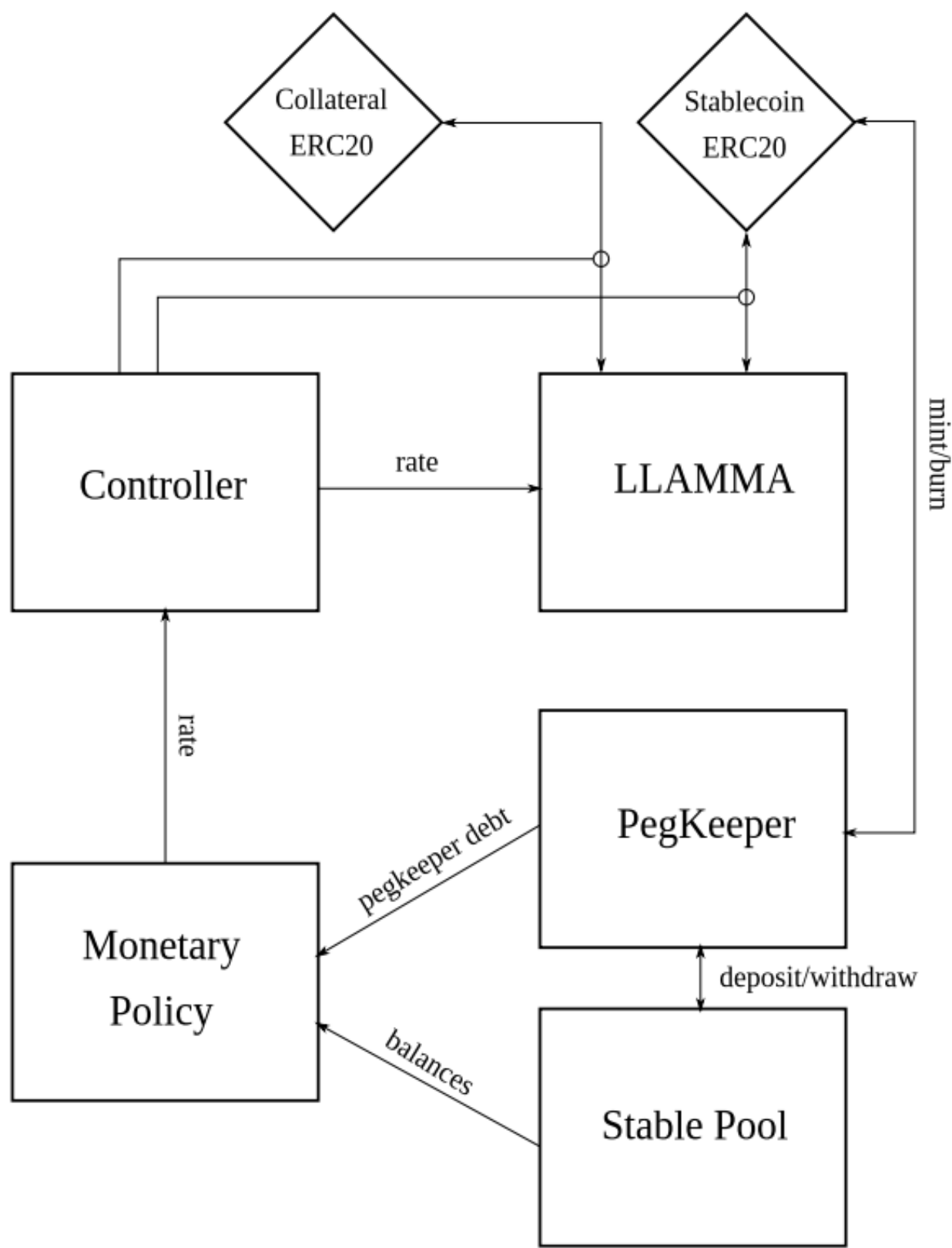


그림 1: 전체 개요도

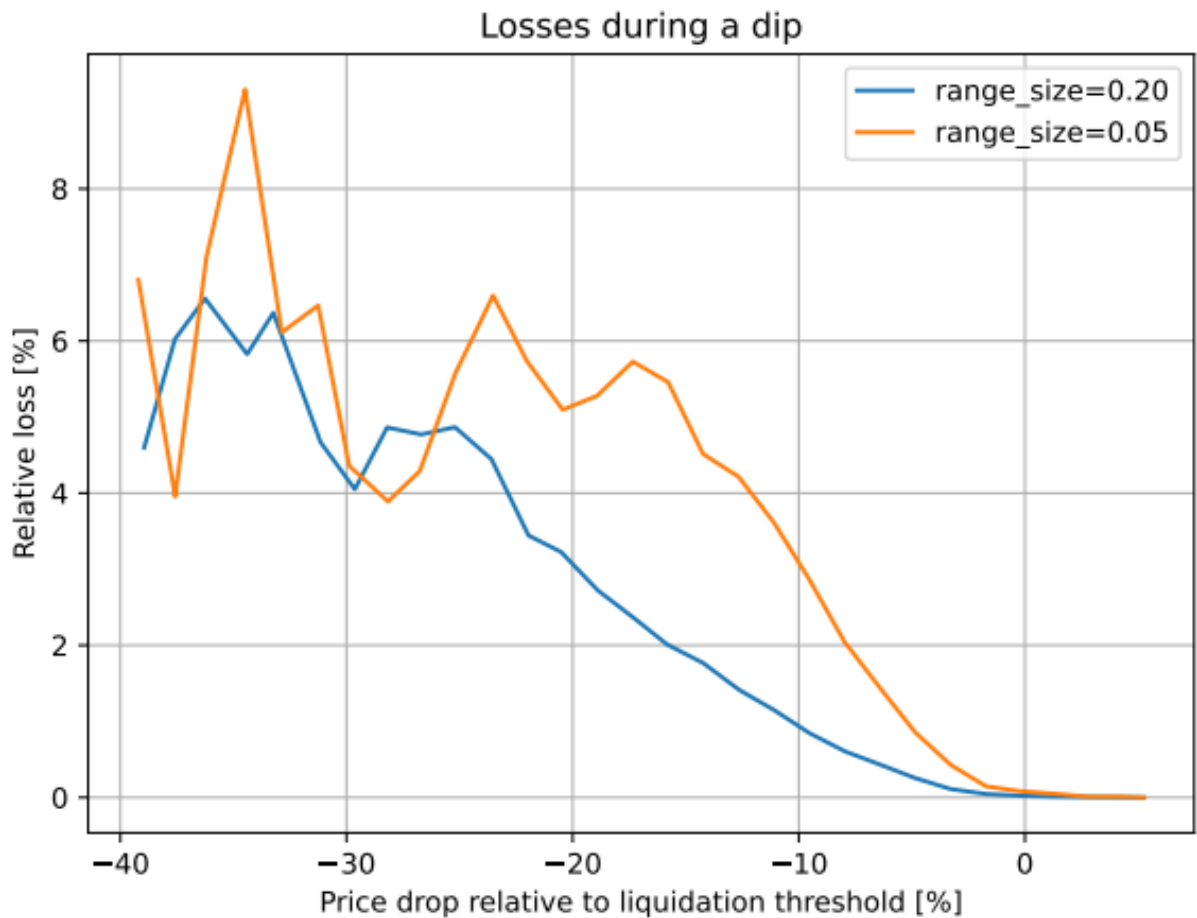


그림 2: 청산 임계값 대비 가격 변동에 따른 손실의 관계. 관찰 기간은 3일.

이 설계에서는 담보를 기반으로 대출을 받은 사용자가 청산 임계값에 도달하더라도, 담보 자산의 가격이 하락 후 반등할 경우 큰 손실이 발생하지 않는다. 예를 들어, 2017년 9월 이후 ETH/USD의 과거 데이터를 사용한 시뮬레이션에 따르면, CDP를 3일 동안 방치했을 때 가격이 청산 임계값보다 10% 하락하더라도 담보의 손실은 1%에 불과하다.

연속적 청산/역청산을 위한 AMM (LLAMMA)

스테이블코인 설계의 핵심은 대출-청산 AMM 알고리즘(Lending-Liquidating AMM Algorithm, LLAMMA)이다. 이 알고리즘의 개념은 담보 자산(예: ETH)과 스테이블코인(여기서는 USD로 표현)을 상호 변환하는 것이다. 담보의 가격이 높을 때는 사용자의 예치금이 모두 ETH로 구성되지만, 가격이 낮아질수록 USD로 점진적으로 전환된다. 이는 기존의 AMM 설계에서 USD가 상단에, ETH가 하단에 위치하는 구조와 매우 다르다.

아래의 설명은 완전한 수학적 증명이라기보다는 개념적 수준의 구성이다. 특히 불변식(invariant)은 차원 분석(dimensional consideration)을 기반으로 도출되었으며, 완전한 수학적 기술을 위해서는 추가 연구가 필요할 수 있다. 그러나 다음의 설명은 실제 구현에 충분한 수준으로 간주된다.

이 구조는 외부 가격 오라클을 통해서만 가능하다. 요약하자면, 일반적인 AMM(예: 쌍곡선 형태의 본딩 커브)을 만들고, 그 “중심 가격(center price)”을 아래에서 위로 점진적으로 이동시키면, 토큰들이 USD에서 ETH로, 또는 그 반대로 서서히 변환되며 양방향 유동성을 제공한다(그림 3). 이는 양자물리학의 비선형 천이(Landau–Zener 전이) 현상과 유사하지만, 수학적 기술은 전혀 다를 수 있다.

유동성이 집중되는 구간을 *밴드(band)*라 하며, 일정한 p_o 에서 밴드는 p_{cd} 에서 p_{cu} 까지 유동성을 가진다. 우리는 $p_{cd}(p_o)$ 와 $p_{cu}(p_o)$ 가 p_o 의 함수이되, 선형보다 가파르게 증가하는 함수를 찾고자 한다(그림 4). 또한 $p_{\downarrow}$ 와 $p_{\uparrow}$ 를 정의하여 각각 $p_{\downarrow}(p_o) = p_o$, $p_{\uparrow}(p_o) = p_o$ 를 만족하도록 하며, 이는 아디아바틱 한계(adiabatic limit)에서 밴드의 경계를 결정한다(즉, $p = p_o$ 일 때를 의미한다).

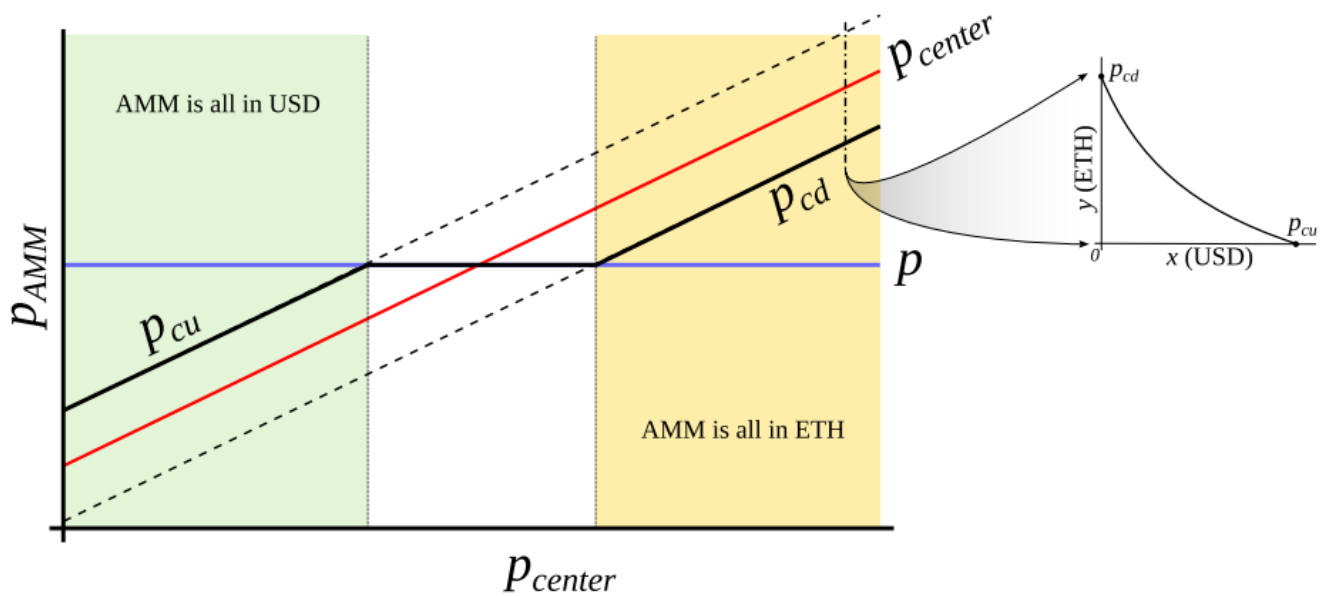


그림 3: “외부 가격 소스가 있는 AMM”의 동작. 외부 가격 p_{center} 는 유동성이 형성되는 중심 가격을 결정한다. AMM은 p_{cd} 에서 p_{cu} 까지의 가격 구간에 유동성을 집중시키며, $p_{cd} < p_{center} < p_{cu}$ 를 만족한다. 현재 가격 p 가 p_{cd} 와 p_{cu} 사이에 있을 때 AMM 가격은 p 와 동일하다. p_{cu} 에 도달하면 모든 자산이 스테이블코인으로, p_{cd} 에 도달하면 모든 자산이 담보로 전환된다.

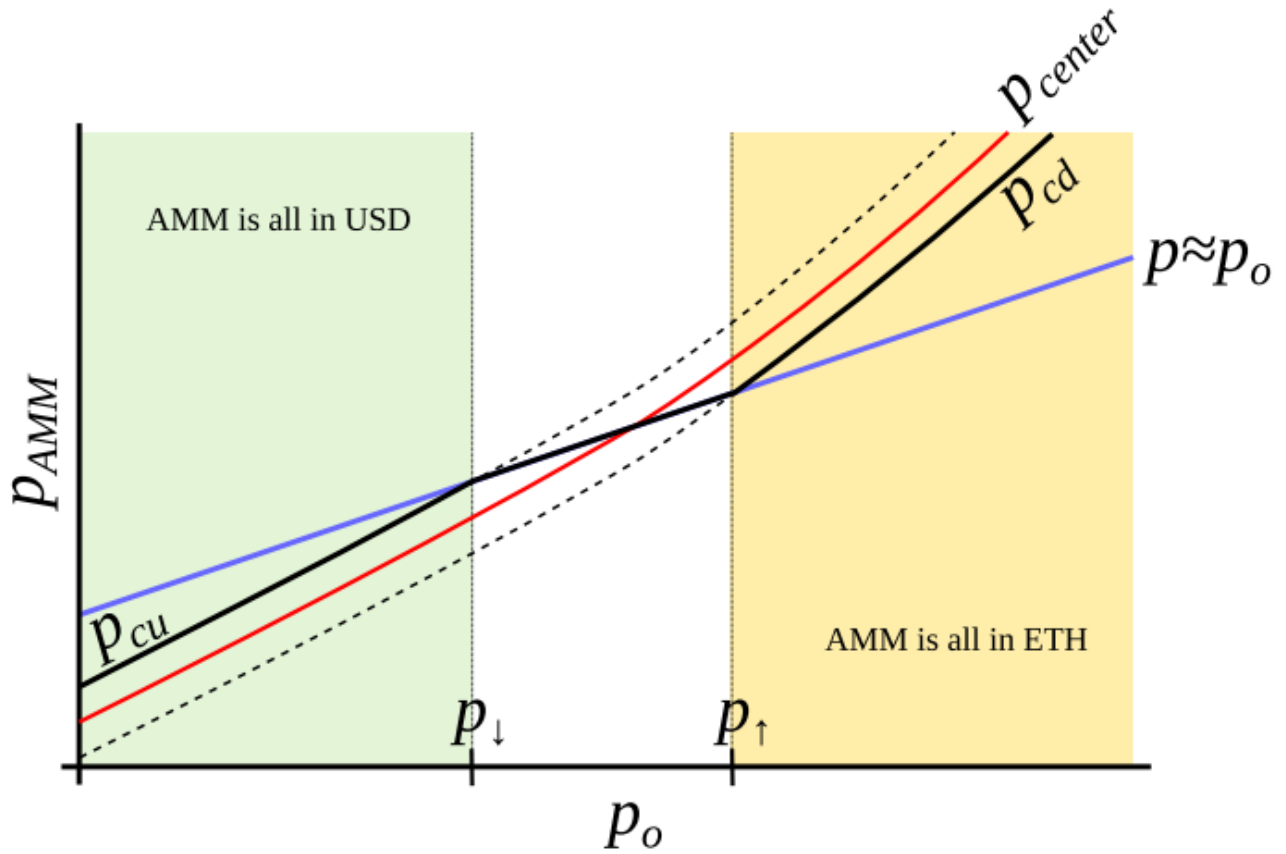


그림 4: 목표 AMM 곡선의 동작. 우리는 p_{cd} 와 p_{cu} 가 p_o 의 함수로서 p_o 가 증가할 때 더 빠르게 증가하는 AMM을 구축하고자 한다. 이 경우, ETH가 비쌀 때 AMM은 전부 ETH로 구성되고, ETH가 저렴할 때는 전부 USD로 구성된다.

우리는 여러 개의 밴드(bands)에서 시작하며, Uniswap v3와 유사하게 본딩 커브(bonding curve)의 쌍곡선 형태를 가상 잔액(virtual balances)을 추가함으로써 유지한다. USD의 양을 x , ETH의 양을 y 라고 하면, “증폭된” 상수곱 불변식은 다음과 같다:

$$I = (x + f)(y + g). \quad (1)$$

또한 $x' \equiv x + f$, $y' \equiv y + g$ 로 정의하면, 불변식은 익숙한 형태인 $I = x'y'$ 로 쓸 수 있다.

그러나 f 와 g 는 상수가 아니며, 외부 가격 오라클(external price oracle)에 따라 변한다(따라서 불변식 I 도 변한다). 즉, 불변식은 오직 오라클 가격 p_o 가 변하지 않을 때만 유지된다. 주어진 p_o 에서 f 와 g 는 밴드 내에서는 일정하다. 앞서 언급한 바와 같이, $p_{\uparrow}$ 는 밴드의 상단 가격을, $p_{\downarrow}$ 는 밴드의 하단 가격을 의미한다. 우리는 유동성의 집중 정도를 나타내는 매개 변수 A 를 다음과 같이 정의한다:

$$\frac{p_{\downarrow}}{p_{\uparrow}} = \frac{A - 1}{A}. \quad (2)$$

우리가 원하는 특성은 더 높은 가격 p_o 가 동일한 잔액에서 더 높은 가격을 유도하도록 하는 것이다. 이렇게 되면 현재 시장 가격(평균적으로 p_o 를 따르게 됨)은 그보다 낮게 유지되어, 밴드가 ETH로 전환 되는 방향으로 거래가 진행된다(반대의 경우에는 USD 방향으로 전환된다). 이를 만족시키는 방법은 여러 가지가 있지만, 우리는 다음과 같은 형태를 채택한다:

$$f = \frac{p_o^2}{p_{\uparrow}} A y_0, \quad g = \frac{p_{\uparrow}}{p_o} (A - 1) y_0, \quad (3)$$

여기서 y_o 는 현재 밴드 내 예치금의 양을 ETH 단위로 나타내는 p_o 의존적 척도이다. 이는 현재 가격 p , $p_{\uparrow}$, 그리고 p_o 가 모두 동일할 때 $y = y_o$ 이고 $x = o$ 이 되도록 정의된다 (그림 4에서 $p_o = p_{\uparrow}$ 지점을 참조). 이 시점의 y 값을 대입하면:

$$I = p_o A^2 y_o^2. \quad (4)$$

가격은 dx'/dy' 로 정의되며, 상수곱 불변식의 경우 다음과 같다:

$$p = \frac{dx'}{dy'} = \frac{x'}{y'} = \frac{f + x}{g + y}. \quad (5)$$

$p_o = p_{\uparrow}$ 또는 $p_o = p_{\downarrow}$ 인 경우 각각 $x = o$ 또는 $y = o$ 을 대입하면 위의 식들이 자기 일관성을 가지고 있음을 확인할 수 있다.

일반적으로 하나의 밴드에 대해서는 $p_{\uparrow}$ 가 주어지고, 따라서 $p_{\downarrow}$, p_o , 상수 A , 그리고 현재 밴드의 예치금 x 와 y 도 알고 있다고 가정할 수 있다. 모든 값을 계산하기 위해서는 y_o 를 찾아야 하며, 이는 다음의 불변식에 대한 2차 방정식을 풀어 얻을 수 있다:

$$\left(\frac{p_o^2}{p_{\uparrow}} A y_o + x \right) \left(\frac{p_{\uparrow}}{p_o} (A - 1) y_o + y \right) = p_o A^2 y_o^2. \quad (6)$$

이 식은 y_o 에 대한 2차 방정식으로 정리된다:

$$p_o A y_o^2 - y_o \left(\frac{p_{\uparrow}}{p_o} (A - 1) x + \frac{p_o^2}{p_{\uparrow}} A y \right) - x y = 0. \quad (7)$$

스마트 컨트랙트에서는 이 2차 방정식을 `get_y0` 함수 내에서 계산한다.

오라클 가격 p_o 가 일정하게 유지되는 동안, AMM은 정상적인 방식으로 작동한다. 즉, 가격이 상승하면 ETH를 판매하고, 가격이 하락하면 ETH를 매수한다. “현재 하단 가격(current down price)” p_{cd} 의 경우 $x = o$, “현재 상단 가격(current up price)” p_{cu} 의 경우 $y = o$ 을 각각 불변식에 대입하면, AMM의 현재 p_o 와 $p_{\uparrow}$ 에서의 가격은 다음과 같음을 확인할 수 있다:

$$p_{cd} = \frac{p_o^3}{p_{\uparrow}^2}, \quad p_{cu} = \frac{p_o^3}{p_{\downarrow}^2}. \quad (8)$$

또 하나의 실질적으로 중요한 문제는 가격이 매우 천천히 변하여 오라클 가격 p_o 가 이를 *아디아바틱* (adiabatic)하게 따라갈 수 있을 때, 즉 가격이 상승하면 밴드가 ETH를 얼마나($y \uparrow$), 가격이 하락하면 USD를 얼마나($x \downarrow$) 가지게 되는지를 구하는 것이다. 현재 값 x , y , 그리고 시작점 $p = p_o$ 에서 이를 수학적으로 정확히 푸는 것은 간단하지 않지만, 수치 계산 결과는 다음과 같은 단순한 형태로 나타났다:

$$y_{\uparrow} = y + \frac{x}{\sqrt{p_{\uparrow}p}}, \quad (9)$$

$$x_{\downarrow} = x + y\sqrt{p_{\downarrow}p}. \quad (10)$$

이 결과는 대출의 안정성을 평가하거나 AMM의 잠재적 손실을 분석할 때 사용된다.

이제 하나의 밴드가 정의되었다. 우리는 전체 가격 공간을 여러 밴드로 분할하며, 각 밴드는 서로 맞닿는 $p_{\downarrow}$ 와 $p_{\uparrow}$ 값을 가진다. 기준 가격 p_{base} 와 밴드 번호 n 을 설정하면 다음과 같다:

$$p_{\uparrow}(n) = \left(\frac{A-1}{A}\right)^n p_{base}, \quad p_{\downarrow}(n) = \left(\frac{A-1}{A}\right)^{n+1} p_{base}. \quad (11)$$

식 (7)과 식 (5)의 해를 임의의 밴드에 대해 구하면 다음과 같은 관계가 성립함을 증명할 수 있다:

$$p(x=0, y>0, n) = p_{cd}(n) = p_{cu}(n-1), \quad (12)$$

$$p(x>0, y=0, n) = p_{cu}(n) = p_{cd}(n+1), \quad (13)$$

이는 밴드 간에 어떠한 간격(gap)도 존재하지 않음을 보여준다.

거래는 식 (1)의 불변식을 유지하면서 이루어지지만, AMM 내부의 현재 가격은 오라클 가격 p_o 가 변함에 따라 이동한다. 즉, 식 (8)에서 볼 수 있듯이 p_o 가 하락하면 AMM 가격은 상승하고, p_o 가 상승하면 AMM 가격은 하락한다. 이러한 관계는 세제곱(cubic) 비례 관계로 표현된다.

LLAMMA와 스테이블코인의 비교

스테이블코인은 변동성이 큰 담보(예: ETH 등 암호화폐)를 기반으로 스테이블코인을 대출받는 CDP(Collateralized Debt Position) 형태를 취한다. 담보는 LLAMMA 내에서 특정 가격 범위(즉, 특정 밴드)에 로드되며, 담보의 가격이 비교적 느리게 하락할 경우 ETH가 점진적으로 스테이블코인으로 변환되어 CDP를 청산할 수 있을 만큼 충분한 스테이블코인이 생성된다. 이는 자가 청산(self-liquidation)을 통해 이루어질 수도 있고, 담보 비율이 위험 수준에 근접하거나 너무 멀리 벗어날 경우 외부 청산자(external liquidator)에 의해 이루어질 수도 있다. 또한, 가격이 반등하기를 기다리는 동안 청산이 일어나지 않을 수도 있다.

사용자가 담보를 예치하고 스테이블코인을 대출받으면, LLAMMA 스마트 컨트랙트는 담보를 배치할 밴드 위치를 계산한다. 담보의 가격이 변할 때마다 ETH는 스테이블코인으로 변환되기 시작한다. 시스템이 “수중 상태(underwater)”에 있을 때, 사용자는 이미 대출 상환을 위한 충분한 USD를 보유하고 있다. 획득 가능한 스테이블코인의 양은 공개 메서드 `get_x_down`을 통해 계산할 수 있다. 만약 이 값이 청산 임계값에 너무 근접한다면, 외부 청산자가 개입할 수 있다. 하지만 일반적으로 담보 가격이 하락 후 일정 기간 횡보하거나 빠르게 반등하는 경우, 이러한 청산은 며칠 또는 수주 내에 발생하지 않거나 아예 발생하지 않을 수 있다. `health` 메서드는 `get_x_down`과 부채의 비율 및 담보 가격 상승에 따른 가치 증가를 반환하며, 이는 “청산” 구간보다 충분히 높은 가격일 때 사용자의 포지션 건전성을 평가하는 데 사용된다.

스테이블코인이 이자를 부과하는 경우, 이는 AMM에도 반영되어야 한다. 이를 위해 모든 가격 그리드를 조정한다. 즉, 스테이블코인이 이자율 r 을 부과하면 AMM의 전체 가격 그리드가 동일한 비율만큼 상승하며, 이는 `base_price` 승수를 통해 구현된다. 따라서 부과된 이자율이 양수일 경우, 해당 승수는 시간이 지남에 따라 지속적으로 증가한다.

`get_x_down` 또는 `get_y_up`을 계산할 때는, 먼저 현재 가격이 p_o 로 이동한다고 가정했을 때 얻을 수 있는 스테이블코인 및 담보의 양 x^* , y^* 를 계산한다. 그 후, 오라클 가격 p_o 가 아디아바틱(adiabatic)하게 변하여 가장 낮은 밴드의 최저 가격 또는 가장 높은 밴드의 최고 가격에 도달한다고 가정하고, 그때 얻을 수 있는 스테이블코인 또는 담보의 양을 계산한다. 이를 통해 현재 순간적인 가격에 의존하지 않는 형태로 획득 가능한 스테이블코인의 양을 측정할 수 있으며, 이는 샌드위치 공격(sandwich attack)에 대한 저항성을 확보하는 데 중요하다.

LLAMMA는 가격 소스로 ETH/USD 가격으로 정의된 p_o 를 사용한다는 점이 중요하다. 스테이블코인은 페그 아래($p_s < 1$)에서 거래될 수도 있고, 페그 위($p_s > 1$)에서 거래될 수도 있다. 만약 $p_s < 1$ 이라면 LLAMMA 내의 가격은 $p > p_o$ 가 된다.

아디아바틱 근사(adiabatic approximation)에서 $p = p_o / p_s$ 가 되며, 모든 담보→스테이블코인 전환은 더 높은 오라클 가격에서 일어나거나, 혹은 오라클 가격이 더 낮고 다음과 같은 값일 때와 동일하게 동작한다:

$$p'_o = p_o \sqrt{\frac{p_o}{p}} = p_o \sqrt{p_s}. \tag{14}$$

이 가격에서 전환 시 얻는 스테이블코인의 양은 $1 / p_s$ 배 더 많다(단, $p_s < 1$ 일 경우). 반대로, $p_s > 1$ 인 상태가 장기간 지속되는 것은 바람직하지 않으며, 이를 방지하기 위해 다음 절에서 설명할 안정화 장치(stabilizer)를 사용한다.

자동 안정화 장치(Automatic Stabilizer) 및 통화정책(Monetary Policy)

$p_s > 1$ 인 경우(예: 스테이블코인 수요 급증으로 인한 경우), 스테이블코인과 상환 가능한 기준 코인 또는 LP 토큰 간의 비대칭 예치(asymmetric deposit)를 통해 스테이블스왑(Curve) 풀 내에 페그 유지 준비금(peg-keeping reserve)이 형성된다. $p_s > 1$ 이 되면, PegKeeper 컨트랙트는 담보 없이 스테이블코인을 발행(mint)하고, 이를 스테이블스왑 풀에 단일 방향(single-sided)으로 예치할 수 있다. 단, 예치 후 최종 가격이 1보다 낮아지지 않아야 한다. 반대로, $p_s < 1$ 일 때 PegKeeper는 비대칭 인출을 수행하고 스테이블코인을 소각(burn)할 수 있다.

이러한 조치는 p_s 가 1보다 높을 때 빠르게 하락하도록, 1보다 낮을 때 상승하도록 유도한다. 이는 비대칭 예치와 인출이 가격에 영향을 미치기 때문이다. 비록 발행된 스테이블코인은 명시적 담보 없이 생성되지만, 스테이블스왑 풀의 유동성에 의해 암묵적으로 담보된 형태를 띤다. 전체 발행/소각 주기(mint/burn cycle)는 결과적으로 안정성을 제공하면서 수익성 또한 유지된다.

안정화 장치(stabilizer)에 발행된 스테이블코인의 양(부채)을 d_{st} 라 하고, 스테이블스왑 AMM의 get_dx 함수를 $f_{dx}()$ 로 하여 스테이블코인을 구매하기 위해 필요한 상환 가능한 USD 양을 계산한다고 하자. 이때, 지나치게 큰 준비금이 쌓이지 않도록 하기 위해, 차입 이율 r 을 변동시켜 완만한 안정화(slow stabilization)를 수행한다:

$$p_s = \frac{f_{dx}(d_{st})}{d_{st}}, \quad (15)$$

$$r = r_0 \cdot 2^{-\frac{p_s - 1}{h}}, \quad (16)$$

여기서 h 는 p_s 가 얼마만큼 변할 때 이율 r 이 2배 변화하는지를 나타내는 파라미터이다 (p_s 가 높을수록 r 은 낮아진다). 안정화 장치의 부채 d_{st} 는 $p_s = 1$ 일 때의 기준 이율 r_o 에 따라 서로 다른 값에서 평형을 이루게 된다. 따라서 시스템은 수동으로 r_o 를 설정하는 대신, d_{st} / supply 가 일정 목표값(예: 5%)을 초과할 때 r_o 를 점진적으로 낮추어 차입자가 스테이블코인을 대출 후 매도하도록 유도해 가격을 낮추고, 시스템이 d_{st} 를 소각하도록 만든다. 반대로, d_{st} / supply 가 낮으면 r_o 를 높여 차입자들이 대출을 상환하도록 유도하고, 스테이블코인의 가격을 상승시켜 시스템이 부채 d_{st} 및 안정화 예치를 증가시키도록 한다.

결론

제시된 메커니즘은 스테이블코인 발행 및 대출 과정에서 발생하는 청산 위험을 완화할 수 있는 가능성을 제시한다. 또한, 안정화 장치와 자동 통화정책 메커니즘은 과도하게 큰 PSM(Peg Stability Module) 없이도 페그 유지(peg-keeping)를 지원할 수 있다.