ManiPerp: A Simple Pool-Based AMM for Perpetual Futures

Stephen Grugett
Manifold Markets, MNX

February 2026

Abstract

We present ManiPerp, a minimal automated market maker (AMM) for perpetual futures contracts. The mechanism uses dual liquidity pools—one for long positions and one for short—with a symmetric funding rate that simultaneously haircuts the dominant side and scales up the minority side. All positions are opened and closed at an externally provided oracle price; the system performs no internal price discovery. Solvency is maintained as a continuous invariant: after every oracle update, the system processes liquidations and then performs pro-rata auto-deleveraging (ADL) if needed, ensuring the opposing pool can always cover all profitable positions. We define the core operations (opening, closing, funding, liquidation, and auto-deleveraging) as state transitions over the pool and position state. The design is intended for play-money prediction markets such as Manifold, where simplicity and support for thinly traded, user-created markets are more important than capital efficiency.

1 Introduction

Perpetual futures (perps) are derivative contracts with no expiry date. Unlike traditional futures, which settle on a fixed date, perps use a periodic *funding rate* to anchor the contract price to an external oracle. When a perp trades above the oracle price, long position holders pay short holders, creating downward pressure; the symmetric case applies when the perp trades below the oracle. This mechanism, introduced by BitMEX and now dominant in cryptocurrency markets, produces a simple instrument for gaining leveraged exposure to any continuously quoted index.

Orderbook-based perp exchanges (e.g., dYdX, Hyperliquid) require sufficient liquidity on both sides of the book to function. Pool-based designs such as GMX and Ostium replace the orderbook with shared liquidity pools that act as counterparty to all traders, making them better suited to long-tail markets with low trading volume. ManiPerp follows this approach but with a substantially simpler mechanism, trading capital efficiency for clarity and ease of implementation.

The primary motivation is to extend prediction market platforms like Manifold beyond binary and multi-choice questions to continuous time-series: asset prices, temperature indices, approval ratings, AI benchmark scores, and any other quantity with a regularly updated data feed.

2 Model

2.1 State

We define the system state as a tuple $(L, S, \mathcal{P})$, where L and S are the mana balances of the Long and Short liquidity pools respectively, and $\mathcal{P}$ is the set of open positions. Each position $p \in \mathcal{P}$ is a tuple (d, q, c, P_e) where:

- $d \in \{\texttt{long}, \texttt{short}\}$ is the direction,
- q is the position size (in notional mana),
- c is the cost basis (margin deposited),

- P_e is the entry price.

The effective leverage of a position is $\ell = q/c$, and the liquidation price is:

$$P_{\text{liq}} = \begin{cases} \left(1 - \frac{1}{\ell}\right) P_e & \text{if } d = \texttt{long} \\ \left(1 + \frac{1}{\ell}\right) P_e & \text{if } d = \texttt{short} \end{cases} \tag{1}$$

Let P denote the current oracle price, an externally provided value updated at regular intervals by a data source outside the system (e.g., an API feed for asset prices, a weather service for temperature indices). The system does not perform price discovery—all positions are opened and closed at the prevailing oracle price. The AMM's role is solely to manage liquidity, leverage, and solvency.

A market is initialized with $L_0 = S_0 > 0$ and $\mathcal{P} = \emptyset$.

2.2 Opening a Position

A trader opens a position by specifying a direction d, a mana amount m, and a leverage $\ell \leq \ell_{\max}$. The position is entered at the current oracle price P.

Definition 2.1 (Open)**.** $\texttt{open}(d, m, \ell)$ takes as input a direction, mana amount, and leverage, and updates the state:

$$(L, S, \mathcal{P}) \xrightarrow{\texttt{open}(d,m,\ell)} (L', S', \mathcal{P}') \tag{2}$$

where, if $d = \texttt{long}$:

$$L' = L + m, \qquad S' = S, \qquad \mathcal{P}' = \mathcal{P} \cup \{(\texttt{long},\ m \cdot \ell,\ m,\ P)\} \tag{3}$$

and symmetrically for $d = \texttt{short}$.

2.3 Funding

Every funding period, we compute the imbalance between the two pools and transfer mana from the dominant side to the minority side. Funding is *symmetric*: positions on the dominant side are haircut, while positions on the minority side are scaled up in proportion to the mana they receive.

Definition 2.2 (Imbalance Function)**.** Let $r = L/S$. The imbalance function I is defined as:

$$I(r) = \frac{r - 1}{r - 1 + k}, \qquad r \geq 1 \tag{4}$$

where $k > 0$ is a sensitivity parameter. I satisfies:

1. $I(1) = 0$,
2. I is monotonically increasing,
3. $\lim_{r \to \infty} I(r) = 1$,
4. $I(1 + k) = \frac{1}{2}$—i.e., the funding rate reaches half its maximum when the imbalance ratio exceeds equilibrium by k.

The funding rate for the period is $f = I(r) \cdot f_{\max}$, where $f_{\max}$ is a market parameter.

Definition 2.3 (Fund)**.** $\texttt{fund}()$ updates the state as follows. If $L > S$, let $\Delta = f \cdot L$ be the mana transferred:

$$(L, S, \mathcal{P}) \xrightarrow{\texttt{fund}()} (L', S', \mathcal{P}') \tag{5}$$

where:

$$L' = L - \Delta = (1 - f) \cdot L \tag{6}$$
$$S' = S + \Delta \tag{7}$$

Long positions are haircut: for each $p = (\texttt{long}, q, c, P_e) \in \mathcal{P}$:

$$p' = \big(\texttt{long},\ (1 - f) \cdot q,\ (1 - f) \cdot c,\ P_e\big) \tag{8}$$

Short positions are scaled up by the proportional increase to the Short pool. Let $g = \Delta/S = fL/S$. For each $p = (\texttt{short}, q, c, P_e) \in \mathcal{P}$:

$$p' = \big(\texttt{short},\ (1 + g) \cdot q,\ (1 + g) \cdot c,\ P_e\big) \tag{9}$$

The symmetric case applies when $S > L$.

Remark 2.1. Funding is symmetric: positions on the dominant side are haircut by a factor of f, while positions on the minority side are scaled up by a factor of $g = fL/S$ (or fS/L in the opposite case). Since both pools and their respective positions are scaled by the same factor, each pool can always return the cost bases of its participants. The minority side is actively rewarded for providing balance to the market.

2.4 Oracle Update: Liquidation and Auto-Deleveraging

When the oracle price updates to a new value P, the system performs two operations in sequence: first liquidation, then auto-deleveraging (ADL). Together, these maintain the invariant that the system is always solvent—i.e., every position can be closed at full value.

Definition 2.4 (Liquidate)**.** `liquidate`(P) takes a new oracle price and processes liquidations. For each long position $p = (\texttt{long}, q, c, P_e) \in \mathcal{P}$ with $P_{\text{liq}} \geq P$:

$$q' = 0, \qquad c' = 0 \tag{10}$$

The position's margin remains in the Long pool (L is unchanged). The symmetric case applies for short positions where $P_{\text{liq}} \leq P$.

After liquidations, profitable positions on one side may claim more equity than the opposing pool can cover. Auto-deleveraging restores solvency by proportionally scaling down these positions.

Definition 2.5 (Auto-Deleverage)**.** `adl`(P) is applied after `liquidate`(P). For the long side, compute the solvency factor:

$$s_L = \frac{S - C_S}{E_L} \tag{11}$$

where:

- S is the current balance of the Short pool,
- $C_S = \sum_{p_j \in \mathcal{P}_S} \min(c_j,\ v_j)$ is the reserved cost bases for open short positions, where v_j is the current value of position p_j,
- $E_L = \sum_{\substack{p_i \in \mathcal{P}_L \\ \pi_i > 0}} \pi_i$ is the total unrealized equity of profitable long positions, with $\pi_i = \frac{P - P_{e,i}}{P_{e,i}} \cdot q_i$.

If $s_L < 1$, then for each profitable long position $p = (\texttt{long}, q, c, P_e)$ with $\pi > 0$:

$$p' = \big(\texttt{long},\ s_L \cdot q,\ c,\ P_e\big) \tag{12}$$

Only the position size is reduced; the cost basis is unchanged, as it is backed by the trader's own pool. This reduces the effective leverage $\ell = q/c$, which in turn moves the liquidation price further from the entry price—correctly reflecting the position's reduced risk. Total long equity is scaled down to match the available funds in the Short pool.

The solvency factor for the short side, s_S, is computed symmetrically using L, C_L, and E_S, and the same scaling is applied to profitable short positions if $s_S < 1$.

Remark 2.2. After `liquidate` and `adl`, the system satisfies $s_L \geq 1$ and $s_S \geq 1$. This invariant guarantees that every position can be closed at its full current value at any time. Auto-deleveraging serves the same role as ADL in centralized perp exchanges, but is applied continuously and proportionally rather than selectively targeting individual positions.

2.5 Closing a Position

Because solvency is maintained after every oracle update, closing a position always pays the full position value.

Definition 2.6 (Close)**.** `close`(p) for a long position $p = (\texttt{long}, q, c, P_e)$ at the current oracle price P updates the state as follows. The position is exited at P—no slippage or price impact is applied.

The position equity is:

$$\pi = \frac{P - P_e}{P_e} \cdot q \tag{13}$$

If $\pi \leq 0$ (the position is at a loss), the payout is $\max(c + \pi,\ 0)$, drawn entirely from the Long pool:

$$L' = L - \max(c + \pi,\ 0), \qquad S' = S, \qquad \mathcal{P}' = \mathcal{P} \setminus \{p\} \tag{14}$$

If $\pi > 0$ (the position is profitable), the payout is $c + \pi$:

$$L' = L - c, \qquad S' = S - \pi, \qquad \mathcal{P}' = \mathcal{P} \setminus \{p\} \tag{15}$$

The cost basis is returned from the Long pool, and the profit is drawn from the Short pool. The solvency invariant guarantees the Short pool has sufficient funds.

2.6 Liquidity Provision

Multiple liquidity providers (LPs) can co-fund a market. Each LP's ownership is tracked as a fractional share of each pool.

Definition 2.7 (Add Liquidity)**.** `addLiquidity`(Δ_L, Δ_S) for an LP depositing Δ_L and Δ_S into the Long and Short pools respectively updates the state:

$$(L, S) \xrightarrow{\texttt{addLiquidity}(\Delta_L, \Delta_S)} (L + \Delta_L,\ S + \Delta_S) \tag{16}$$

The LP's ownership shares are:

$$\omega_L = \frac{\Delta_L}{L + \Delta_L}, \qquad \omega_S = \frac{\Delta_S}{S + \Delta_S} \tag{17}$$

An LP may withdraw their proportional share of each pool at any time. In practice, the protocol may wish to restrict withdrawals to amounts that preserve the solvency invariant—i.e., ensuring that $s_L \geq 1$ and $s_S \geq 1$ remain satisfied after the withdrawal, so that no withdrawal can push the system into a state requiring auto-deleveraging.

3 Market Parameters

To instantiate a ManiPerp market, the creator specifies the parameters listed in Table 1.

Parameter	**Description**
Oracle	An external index, updated at regular intervals
$\ell_{\max}$	Maximum leverage
$f_{\max}$	Maximum funding rate per period
k	Funding sensitivity
Funding period	Frequency of funding events (e.g., hourly, daily)
(L_0, S_0)	Initial liquidity pool subsidy

Table 1: Market parameters required to instantiate a ManiPerp market.

4 Discussion

ManiPerp is deliberately minimal. All trades execute at the oracle price with no slippage or price impact—the funding rate is the sole mechanism that discourages imbalanced positioning. The design omits several features present in production perp protocols—including price-impact curves, dynamic fees, and internal price discovery—in favor of a system that is easy to reason about, implement, and explain to non-expert users. Two mechanisms guarantee safety: symmetric funding actively rewards the minority side, discouraging persistent imbalance, while continuous auto-deleveraging maintains solvency as an invariant after every oracle update. Together, they ensure the system can always honor full payouts without complexity.

The design opens a large space of continuous-valued markets that are difficult to express as binary or multi-choice prediction questions: weather indices, economic indicators, approval ratings, benchmark scores, and more. Any quantity with a reliable, regularly updated data feed is a candidate for a ManiPerp market.