Addendum A: Vectorized Rebalancing & Dynamic Capacity Logic

Sonia Kolasinska, IndexMaker Labs

January 21, 2026

Abstract

This addendum details the vector-based rebalancing mechanism of the Decentralised Index Maker (DeIndex) protocol. Unlike simple atomic swaps, the DeIndex rebalance routine utilizes a custom Vector Intermediate Language (VIL) to perform a deterministic, dual-sided auction entirely on-chain. We formalize the two-stage process: (1) The derivation of *Target Drift* based on index weight changes, and (2) The *Restoring Force Execution*, which dynamically clamps rebalancing actions against a Capacity Limit (CL) derived from real-time asset liquidity and system margin. This approach ensures that index adjustments mathematically guarantee system solvency via saturating arithmetic and atomic state transitions.

1 Introduction

Rebalancing is the mechanism by which the Index aligns its composition with a new target weight distribution. In standard EVM implementations, this is often an iterative, gas-intensive process. In the DeIndex VM^2 environment, rebalancing is treated as a single atomic vector operation.

The logic is split into two distinct VIL routines:

1. **Target Derivation (`update_rebalance`):** Calculates the net drift in asset quantities required to match the new weight distribution.
2. **Capacity & Execution (`execute_rebalance`):** Calculates the maximum safe execution quantity (CL) based on the "Restoring Force" principle and commits the new state.

2 Stage 1: Target Drift Derivation

The first phase calculates the *Rebalance Vectors* (R_{long}, R_{short}), which represent the quantity of each asset that must be bought or sold to achieve the new portfolio structure.

2.1 Total Supply Calculation

The VIL first computes the net active supply (S_{total}) by strictly netting the Bid and Ask inventory states.

$$S_{total} = M_{bid} - (C_{ask} + S_{ask}) \tag{1}$$

Where M represents minted, C represents committed, and S represents spent Index tokens and the corresponding underlying inventory.

2.2 Weight Delta (ΔW)

The protocol identifies the shift in weights between the old configuration (W_{old}) and the new configuration (W_{new}). Utilizing the VIL's `LUNION` (Label Union) and `JUPD` (Join Update) instructions, vectors are aligned to a superset domain $\mathcal{U}$.

The weight drift is split into mutually exclusive Long and Short components using the VIL's Saturating Subtraction (`SSB`) instruction ($\ominus$), where $a \ominus b = \max(0, a - b)$:

$$\Delta W_{short} = W_{old}^{\mathcal{U}} \ominus W_{new}^{\mathcal{U}} \tag{2}$$

$$\Delta W_{long} = W_{new}^{\mathcal{U}} \ominus W_{old}^{\mathcal{U}} \tag{3}$$

2.3 Rebalance Vector Accumulation

The system calculates the new target quantities by scaling the weight delta by the total supply and accumulating it into the existing rebalance vectors.

$$R'_{long} = R_{long} + (S_{total} \cdot \Delta W_{long}) \tag{4}$$

$$R'_{short} = R_{short} + (S_{total} \cdot \Delta W_{short}) \tag{5}$$

Finally, the system normalizes the rebalance vectors to ensure that for any asset i, the protocol is not simultaneously buying and selling (netting internal crossings):

$$R_{long,final} = R'_{long} \ominus R'_{short} \tag{6}$$

$$R_{short,final} = R'_{short} \ominus R'_{long} \tag{7}$$

3 Stage 2: The Restoring Force & Capacity Limits

The execution phase (`execute_rebalance`) is the system's risk engine. It does not blindly execute the target R; instead, it calculates a *Capacity Limit* (CL) that ensures market stability.

3.1 The Capacity Limit (CL) Formula

The protocol employs a "Restoring Force" logic: exposure that reduces the system's Net Delta (Δ) is allowed up to the full liquidity limit, while exposure that increases Delta is tightly constrained by the Margin (M).

Let L_{iq} be the available market liquidity. The Capacity Limit vectors are derived as:

$$CL_{long} = \Delta_{long} + \min\left((M \ominus \Delta_{short}), L_{iq}\right) \tag{8}$$

$$CL_{short} = \Delta_{short} + \min\left((M \ominus \Delta_{long}), L_{iq}\right) \tag{9}$$

Interpretation:

- The term Δ_{long} implies that we can always "close" an existing long position.
- The term $\min(M \ominus \Delta_{short}, L_{iq})$ represents the *new* capacity we can open, bounded by either the remaining system margin or the physical market liquidity.

3.2 Execution Capping

The actual executed quantity (E) is determined by scaling the Capacity Limit by a governance factor (K). The primary function of K is to preserve a portion of the available margin for user orders, preventing the rebalance mechanism from monopolizing system liquidity. The factor K is strictly constrained to the range $K \in (0, 1]$.

$$E_{long} = \min(R_{long}, K \cdot CL_{long}) \tag{10}$$
$$E_{short} = \min(R_{short}, K \cdot CL_{short}) \tag{11}$$

This ensures that the protocol never attempts to move more assets than the market can absorb while guaranteeing that user-initiated flow retains access to liquidity.

4 Stage 3: State Update & Atomic Commit

Once the execution quantities (E) are determined, the VIL performs an atomic update of the Market Demand and Net Delta vectors.

4.1 Demand Vector Update

The executed rebalance acts as a modifier to the existing market demand (D).

1. **Update Short Demand:** Executed assets satisfy existing Short Demand (D_{short}) first.

$$D_{short,new} = D_{short} \ominus E \tag{12}$$

2. **Residuals to Long Demand:** Any execution quantity exceeding Short Demand flows into Long Demand (D_{long}).

$$D_{long,new} = D_{long} \oplus (E \ominus D_{short}) \tag{13}$$

4.2 Delta (△) Finalization

The final Net Exposure (Delta) is recalculated from the new Total Supply (T) states. This step is critical for maintaining the invariant that Δ_{long} and Δ_{short} are mutually exclusive.

$$\Delta_{long} = (S_{long} + D_{short,new}) \ominus (S_{short} + D_{long,new}) \tag{14}$$
$$\Delta_{short} = (S_{short} + D_{long,new}) \ominus (S_{long} + D_{short,new}) \tag{15}$$

5 Conclusion

The VIL implementation of the rebalance logic demonstrates a novel approach to on-chain financial engineering. By replacing conditional control flow (if/else) with **Saturating Arithmetic** and **Vectorized Clamping**, the protocol achieves high-dimensional portfolio adjustments with $O(N_I + N_M)$ complexity. The mathematical formulation of the Capacity Limit (CL) guarantees that rebalancing operations act as a stabilizing force, strictly adhering to the system's solvency constraints.