| **Metric** | **Traditional EKF** | **Full‑Batch KD dEKF** | **Selective KD dEKF** |
| --- | --- | --- | --- |
| **MSE after training** | 0.100 | 0.050 | ~0.045 |
| **Knowledge Absorbed** | 0 % | 56 % | ~60 % |
| **Risk of Over‑tuning / Drift** | n/a | Medium | Low |
| **Susceptibility to Plateau** | High | High once tuned | Low—each batch on fresh outliers |

Here are concise problem‑statement formulations for the **DSP** and **Model‑Based Design** aspects of your project:

**1. DSP Problem Statement**

**“Design and implement a real‑time digital signal processing front‑end on an ESP32 that conditions 6‑DoF IMU data for downstream knowledge‑distillation filtering. Specifically:**

* **a. Noise Filtering:** Develop IIR/FIR filters (e.g. low‑pass and notch) to remove sensor noise and mechanical vibration from 6‑axis accel/gyro at 200 Hz.
* **b. Feature Extraction & Event Detection:** Compute real‑time statistics (e.g. variance, spectral energy) and implement an “outlier” detector that flags significant motion events (roll > θ, jerk spikes) for selective KD triggers.
* **c. Fixed‑Point Optimization:** Port all DSP routines to CMSIS‑DSP using Q15/Q31 arithmetic to meet a worst‑case compute budget of < 1 ms per sample under FreeRTOS.”

**2. Model‑Based Design (MBD) Problem Statement**

**“Use a model‑based design workflow to specify, simulate, and auto‑generate embedded C code for the differentiable EKF (dEKF) L1 filter on the ESP32, ensuring bit‑exact behavior between simulation and target hardware. Specifically:**

* **a. System Modeling:** Build a Simulink (or equivalent) model that encapsulates the IMU kinematics (f), measurement model (h), and trainable covariance updates (Q, R) in block form.
* **b. Verification & Validation:** Perform MIL/SIL/PIL testing to verify that the simulated dEKF output matches the generated code running on the ESP32 within ±1 % numerical error, and that real‑time deadlines (200 Hz) are met.
* **c. Automatic Code Generation:** Configure code‑generation settings to produce optimized, maintainable C for FreeRTOS—complete with generated parameter‑tuning hooks for micro‑SGD-based covariance updates.”

These two problem statements crisply capture the **DSP** and **MBD** challenges in your project scope, and directly tie into your multi‑path KD architecture.

***Hybrid : Differentiable + Adaptive***

**In the sequential architecture; there are four approaches: -**

**1) Adaptive First: The raw 6-DoF IMU data streams thru the adaptive filter (AF) first, then the differentiable filter (DF). So, the distillation loss calculated between L1 and L2 accounts for AF inference. The LMS algo runs agnostic of DF performance.**

**2) Differentiable First: The raw 6-DoF IMU data streams thru the DF first, then the AF. So, the LMS algo on the AF accounts for DF performance. The KD on DF runs agnostic of AF performance.**

**3) Bistate Inference: Periodically, or smartly, sequence of the filters is switched to optimize the 10 L1 Evaluation Criteria (10L1EC, henceforth).**

**4) Serial Inference Parallel Learning (SIPL): the final filtering is passed thru both, but the individual learning of both DF and AF is mutually exclusive.**