Estimator Tuning Guide

This document describes how to tune the joint state estimator in the MARS Hexapod firmware.

The estimator runs in the main 166 Hz control loop, per joint, and produces:

- Estimated joint position: `pos_cd` (centidegrees)
- Estimated joint velocity: `vel_cds` (centidegrees/second)

The estimator is designed to be lightweight but robust under sparse feedback: each tick you send commands to all servos and read feedback from exactly one servo (round-robin).

Estimator Overview

Per joint, the estimator maintains:

- State:
  - Estimated position: $\hat{p}$ (centidegrees)
  - Estimated velocity: $\hat{v}$ (centidegrees/second)
- Inputs per tick:
  - Effective command: the post-COMP/IMP joint target (centidegrees)
  - Sparse measurement: `g_meas_pos_cd[leg][joint]` when that joint is in the round-robin slot

The update has two conceptual stages:

1. **Predict**
   - Integrate velocity: $\hat{p} \leftarrow \hat{p} + \hat{v} \cdot \Delta t$
   - Pull toward the effective command using `est.cmd_alpha_milli`.
2. **Correct (when measurement arrives)**
   - Pull position toward the measured position using `est.meas_alpha_milli`.
   - Update velocity from the residual using `est.meas_vel_alpha_milli` and $(\text{meas} - \hat{p})/\Delta t$.

The estimator is intentionally simple: no covariance matrices, no dynamic gains. The three scalar gains below control its behavior.

Tunable Gains

All gains are configured in `milli` units (0..1000 or 0..2000) and can be set via `/config.txt` or the `EST` console commands.

1. Command pull: `est.cmd_alpha_milli`

- Console: `EST CMD_ALPHA <milli 0..1000>`

- Config key: `est.cmd_alpha_milli`

Controls how strongly the estimator position is pulled toward the **effective command** in the predict step.

- Low values → estimator behaves more like a filtered measurement; between feedback ticks it does not strongly anticipate command motion.
- Higher values → estimator tracks the command trajectory more aggressively between feedback ticks.

Guideline: keep `CMD_ALPHA` **smaller than** `MEAS_ALPHA` so that reality (measurements) still win long-term.

2. Measurement blend: `est.meas_alpha_milli`

- Console: `EST MEAS_ALPHA <milli 0..1000>`
- Config key: `est.meas_alpha_milli`

Controls how strongly the estimator position is pulled toward the **measured joint position** when feedback arrives.

- Low values → sluggish convergence to the true servo position.
- High values → estimator closely copies measurements with little smoothing.

This is the primary knob for trading off responsiveness vs. noise.

3. Velocity correction: `est.meas_vel_alpha_milli`

- Console: `EST MEAS_VEL_ALPHA <milli 0..2000>`
- Config key: `est.meas_vel_alpha_milli`

Controls how strongly the estimator updates its **velocity** from the residual between measurement and prediction.

- Low values → velocity estimate is small and slow to respond (can look "dead").
- High values → velocity estimate reacts quickly but can amplify measurement noise.

This is the main knob for making `vel_cds` useful without being too noisy.

What Good Behavior Looks Like

On a bench (single leg or whole robot, lightly loaded):

- Estimated position $\hat{p}$:
  - Closely tracks the measured position when feedback is available.
  - Between feedback ticks, interpolates smoothly along the commanded motion.

- Estimated velocity $\hat{v}$:
  - Smooth, roughly proportional to how fast a joint moves.
  - No large spikes at rest; small noise is acceptable.
- Control behavior (PID/COMP/IMP, which now use the estimator):
  - Similar or slightly smoother behavior than using raw measured positions.
  - No obvious extra lag, overshoot, or oscillation when you change estimator gains within reasonable ranges.

If PID shadow telemetry is enabled, you should see:

- `est_pos` closely following `meas_pos`, with at most 1–2 ticks of lag.
- `est_vel` resembling a smoothed derivative of `meas_pos`.

Recommended Starting Values

For a 166 Hz loop with sparse feedback, reasonable starting values are:

```
EST CMD_ALPHA 200
EST MEAS_ALPHA 700
EST MEAS_VEL_ALPHA 400
EST LIST
STATUS
```

In `/config.txt`:

```
est.cmd_alpha_milli=200
est.meas_alpha_milli=700
est.meas_vel_alpha_milli=400
```

These values assume relatively clean servo position feedback and are a good baseline for tripod test mode and slow step motions.

Tuning Procedure

The tuning process is structured in three steps: tune measurement blending, then velocity, then command pull.

Step 0: Setup

1. Put the robot in a safe, stable posture (e.g., STAND or a supported tuck).
2. Enable PID shadow telemetry and/or logging for:
   - Commanded joint position
   - Measured joint position
   - Estimated joint position
   - Estimated joint velocity

3. Use simple motions first:
   - Single-joint tests via `RAW3`.
   - Slow periodic motions before trying full tripod gaits.

Step 1: Tune `est.meas_alpha_milli` (MEAS_ALPHA)

Goal: make `est_pos` follow the measured position with mild smoothing.

1. Temporarily neutralize other influences:

   ```
   EST CMD_ALPHA 0
   EST MEAS_VEL_ALPHA 0
   ```

2. Start with a moderate measurement gain:

   ```
   EST MEAS_ALPHA 200
   ```

3. Slowly move a leg joint back and forth and observe `est_pos` vs `meas_pos`.

4. Adjust:

   - If `est_pos` noticeably lags `meas_pos` → increase `MEAS_ALPHA`.
   - If `est_pos` is almost identical but noisy → decrease `MEAS_ALPHA`.

5. Typical good range: **500–800**.

   Example:

   ```
   EST MEAS_ALPHA 700
   ```

Step 2: Tune `est.meas_vel_alpha_milli` (MEAS_VEL_ALPHA)

Goal: obtain a stable, useful velocity estimate.

1. Keep `CMD_ALPHA` at 0 and use the tuned `MEAS_ALPHA` from Step 1.

2. Start with a modest velocity gain:

   ```
   EST MEAS_VEL_ALPHA 200
   ```

3. Apply a ramp motion: move a joint from low to high and back at a steady speed.

4. Observe `est_vel`:

   - If `est_vel` barely moves → `MEAS_VEL_ALPHA` too low.
   - If `est_vel` spikes or is very noisy at rest → `MEAS_VEL_ALPHA` too high.

5. Increase in steps of ~100 until `est_vel` tracks motion well but remains relatively quiet at rest.

   Typical good range: **300–600**.

   Example:

```
EST MEAS_VEL_ALPHA 400
```

Step 3: Tune `est.cmd_alpha_milli` (CMD_ALPHA)

Goal: between sparse feedback ticks, make `est_pos` follow the command trajectory without drifting away from reality.

1. Use the tuned `MEAS_ALPHA` and `MEAS_VEL_ALPHA` from Steps 1–2.
2. Start low:

   ```
   EST CMD_ALPHA 100
   ```

3. Run a simple gait (e.g., tripod test mode) or repeated stepping motion.
4. Observe:
   - If `est_pos` noticeably lags the command and only catches up when measurements arrive → `CMD_ALPHA` too low.
   - If `est_pos` closely follows command but drifts away from `meas_pos` over time → `CMD_ALPHA` too high.
5. Increase `CMD_ALPHA` in small increments (50–100) until you see clear improvement between feedback ticks but no long-term drift.

   Guideline: keep `CMD_ALPHA` roughly **1/2 to 1/3** of `MEAS_ALPHA`.

   Example with `MEAS_ALPHA 700`:

   ```
   EST CMD_ALPHA 200
   ```

Interpretation Cheatsheet

CMD_ALPHA (`est.cmd_alpha_milli`)

- **Too low (< 50)**: estimator behaves like a filtered measurement; doesn't anticipate commanded motion.
- **Good (100–300)**: estimator follows the command smoothly between feedback ticks while still being corrected by measurements.
- **Too high (> 500 or » MEAS_ALPHA)**: estimator chases the command more than reality; can drift when servos saturate or lag.

MEAS_ALPHA (`est.meas_alpha_milli`)

- **Too low (< 200)**: sluggish adaptation to true servo motion.
- **Good (500–800)**: fast convergence with mild smoothing.
- **Too high (~1000)**: estimator is nearly identical to raw measurements; noisy if feedback is noisy.

MEAS_VEL_ALPHA (`est.meas_vel_alpha_milli`)

- **Too low (< 100)**: velocity estimate is weak and often near zero.
- **Good (300–600)**: velocity tracks motion and is reasonably smooth.
- **Too high (> 800–1000)**: velocity becomes noisy and can show large spikes.

Practical Tuning Session Script

When tuning on hardware:

1. Reset to safe defaults:

   ```
   EST CMD_ALPHA 100
   EST MEAS_ALPHA 700
   EST MEAS_VEL_ALPHA 300
   EST LIST
   ```

2. Run a single-joint test (e.g., tibia of LF) using `RAW3` and watch/log:

   - `meas_cd`
   - `est_pos_cd`
   - `est_vel_cds`

3. Adjust `MEAS_ALPHA` until `est_pos` ≈ `meas_pos` with mild smoothing.

4. Adjust `MEAS_VEL_ALPHA` until `est_vel` is stable and responsive.

5. Adjust `CMD_ALPHA` while running a tripod gait, choosing the lowest value that clearly improves between-feedback behavior without introducing long-term drift.

Safety Notes

- Always tune in a safe posture with limited range of motion and, ideally, with the robot supported.
- If you see large oscillations in `est_vel` at rest or sustained bias between `est_pos` and `meas_pos`:
  - Reduce `MEAS_VEL_ALPHA` and/or `CMD_ALPHA`.
  - Re-verify that `MEAS_ALPHA` is high enough for measurements to dominate long-term.

Once tuned, the estimator should make PID/COMP/IMP layers more consistent under sparse feedback without adding noticeable lag or instability.