Hip Large の時のズレについて

A slight offset in the hip angle for the larger perturbation is **not necessarily a bug**—it can arise quite naturally in a multi-link system for several reasons:

1. **Relative vs. Absolute Angle Convention**
    - If q2 is the **relative** rotation between link 2 and link 1, then a large initial deflection of link 1 can shift link 2's motion in an asymmetric way.
    - Meanwhile, the ankle joint (q1) may appear more "centered" if q1 is measured from a fixed global reference (or if that link is more directly under feedback control).
2. **Dynamics Coupling and Reference**
    - The second link's behavior depends on both link 1's motion and any coupling terms in your dynamics. A bigger initial lean can cause the second link to "drift" into a slightly shifted equilibrium or limit cycle—especially if you have no explicit damping or friction to enforce perfect symmetry around zero.
    - Even if your reference state is [q1,q2,dq1,dq2]=0[q1,q2,dq1,dq2]=0, the actual closed-loop system can settle into a small offset if there's any unmodeled effect or bias.
3. **Process/Measurement Noise and EKF Estimation**
    - With higher noise or larger transients, the EKF can accumulate a small bias in the estimated angle for the hip if it sees repeated asymmetrical corrections.
    - This typically appears as a consistent offset or swirl in the phase portrait for the second joint, especially under bigger disturbances.
4. **Friction or Model Mismatch**
    - If there is even a tiny friction or torque bias (e.g.\ from joint weight distribution, initial posture, or unaccounted damping), link 2 may not come back perfectly to zero.
    - Also check if `I2` or `com2` was approximated. A mismatch can cause small offset errors for large motions.

Is It "Normal"?

Yes—if your system or estimation model allows any small bias or offset to persist, especially under a **big** perturbation, seeing a shifted or spiraling phase portrait for the hip is quite normal in a double-link system. You can often reduce that offset by:

- **Adding a small damping** or friction term in `doubleLinkDynamics`.
- **Improving the EKF tuning** (Qekf,Rekf) if the offset appears estimation-related.
- **Double-checking the sign convention** for q2 if it's meant to be absolute vs. relative to q1.

If everything is otherwise stable and converges near zero velocity, then having a slightly shifted average angle for the second link (compared to the first) is not unusual. It just indicates the second link's posture or the EKF's estimate is consistently leaning one way under the larger excitation.

EKF 後の Pareto について

Whether you **need** to rerun the Pareto optimization after adding the EKF depends on what, exactly, you are optimizing and how the paper's methodology is structured. Here are some practical guidelines:

---

1. **If the Paper Treats EKF as Just a State-Estimator "Plug-In"**

- Often, in real-time control, the controller (Q,R) design is done **first** under the assumption of full state feedback.
- Then, to handle noisy partial measurements, one **adds** an EKF (or some observer) so the controller can operate with $\hat{x}$ in place of x.
- In that scenario, the authors might **not** re-run any multi-objective optimization for (Q,R). They typically keep the same (Q,R) in the MPC cost function and simply evaluate how well the system performs when the states are estimated instead of measured perfectly.

If that's the structure the paper follows, **you do not** strictly re-run the Pareto optimization. You just show that the existing (Q,R) controller can still perform well once you incorporate state estimation.

---

2. **If the Paper Ties Estimator & Controller Tuning Together**

- In some studies, especially advanced or integrated "co-design" approaches, the authors treat observer and controller design as a **joint** optimization (for example, optimizing both the controller's (Q,R) and the filter's $(Q_\mathrm{ekf}, R_\mathrm{ekf})$ in a single multi-objective procedure).
- In that case, **yes**, you would repeat the Pareto optimization with the EKF in the loop, because your cost function presumably accounts for the real closed-loop performance under partial/noisy measurements.

If that's the structure the paper uses, you **do** re-run the optimization. The cost function is now a function of the entire closed-loop system with estimation in the loop, so you want the best (Q,R) given that your states come from the EKF.

---

3. **Practical Recommendation**

1. **Check the paper's methodology**:
   - Did they first present the MPC design (with (Q,R) found via Pareto) in an ideal full-state scenario, *then* simply say "We use an EKF to get $\hat{x}$" with no mention of re-optimizing?

- Or do they explicitly say “We performed the same Pareto optimization but under measurement noise with an observer in the loop”?

2. **If the paper does not re-run it**:
   - You can just keep your existing \((Q,R)\).
   - Show results that the controller is robust enough to handle partial/noisy state info (via the EKF).

3. **If you want thoroughness** and your cost function (in `paretoObjective`) is truly measuring the “real” performance (i.e., the real state vs. reference, plus input energy) under measurement noise and an EKF, **then** re-running the multi-objective GA can yield different \((Q,R)\) that might be better for the noisy scenario.

Short Answer

It depends on how the paper sets up the observer+controller.

- **If** they design the MPC (with \((Q,R)\)) under perfect state feedback and later bolt on an EKF for implementation, then **no**—they likely do **not** re-run the multi-objective optimization.
- **If** their approach is an integrated design that accounts for noisy measurements in the cost function from the start, then **yes**—you should re-run the Pareto optimization with the EKF in the loop.