Zephyflyer: Flying Drones Under the Wind

24774 ACSI Final Presentation

Hanze Liu, Nathan Sun, Yunliang Zhao

Fall 2025, Team 5

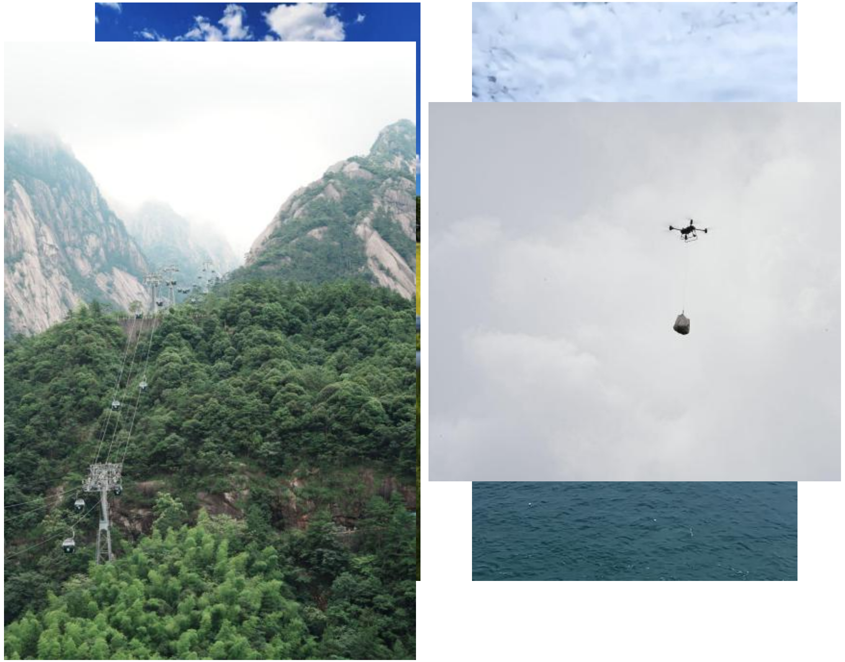
Problem Definition

Drones when flying, especially at higher altitudes, can face disturbances such as crosswinds, which can cause the drone to deviate from its path.

Implement controller(s) to recover the drone from the unknown disturbance to its planned trajectory.

Goal:

Return to the planned original trajectory and goal with minimal drift and fast recovery from disturbance



Relevant Literature

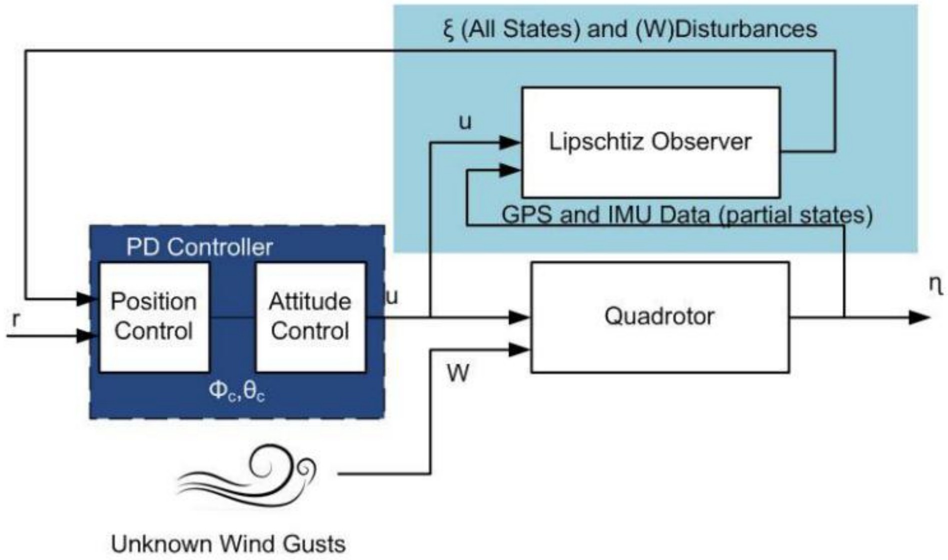


FIGURE 6. Proposed system architecture, the aircraft pose information within the observer provides accurate information to the controller for precise position.

Precise Trajectory Tracking of Multi-Rotor UAVs using Wind Disturbance Rejection Approach

- Lipschitz Stability + Real-Time UIO to form LUIO for active wind rejection
- Requires GPS sensors

https://ris.cdu.edu.au/ws/portalfiles/portal/86661900/Precise_Trajectory_Tracking_of_Multi_Rotor_UAVs_Using_Wind_Disturbance_Rejection_Approach_1_.pdf

A

Approach

🛠️ Hardware

- Built and validated Crazyflie + Flowdeck
- Set up reliable firmware flashing
- Ran PID and PID+ESO on hardware

🖥️ Simulation

- Built Webots environment (linearized dynamics)
- Implemented PID, LQR, TinyMPC, PID + ESO, LQR + ESO
- Evaluated robustness under wind disturbances

📊 Results

- Benchmarked all controllers (sim + hardware)
- Compared nominal vs. wind-disturbed performance
- Analyzed sources of sim–hardware mismatch

System Dynamics

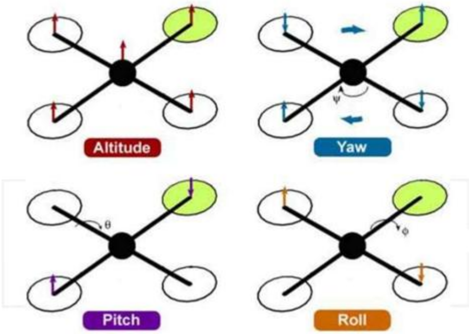
Force & Moments:

- Each Motor i produces:

$$F_i = K_f \omega_i^2, \quad M_i = K_m \omega_i^2$$

- Total control inputs:

$$\begin{aligned} U_1 &= F_1 + F_2 + F_3 + F_4 && \text{(Total thrust)} \\ U_2 &= (F_2 - F_4)L && \text{(Roll moment)} \\ U_3 &= (F_3 - F_1)L && \text{(Pitch moment)} \\ U_4 &= (M_1 - M_2 + M_3 - M_4) && \text{(Yaw moment)} \end{aligned}$$



Equation of Motion:

$$m\ddot{\mathbf{r}} = m\mathbf{g} + R\,[0, 0, -U_1]^T$$

$$I\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (I\boldsymbol{\omega}) = [U_2, U_3, U_4]^T$$

Hover Simplification:

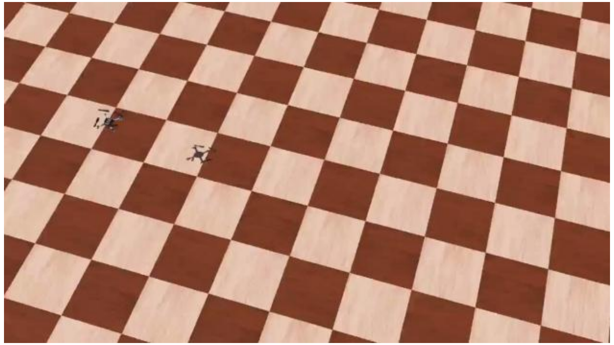
$$\ddot{x} \approx -\frac{U_1}{m}\theta, \quad \ddot{y} \approx \frac{U_1}{m}\phi, \quad \ddot{z} \approx -g + \frac{U_1}{m}$$

Circular Trajectory:

$$x(t) = x_c + R\cos(\omega t), \quad y(t) = y_c + R\sin(\omega t), \quad z = const$$

Simulation & Environment Setup

- Impulse Force Mode
  - Single short disturbance, like a gust of wind.
  - Direction and strength adjustable.
- **Global Wind Mode - 0.03 N**
  - Continuous wind across the whole environment.
  - Drag force proportional to wind speed squared.
  - Simulates steady airflow; tests long-term stability.
- Local Wind Mode
  - Wind active only in a defined spatial region.
  - Same drag computation applied locally.
  - Simulates tunnels or uneven wind zones.



$$F_{\text{drag}} = \frac{1}{2} \rho C_d A v^2$$

$$m\mathbf{a} = \mathbf{T} - \mathbf{F}_{\text{drag}}$$

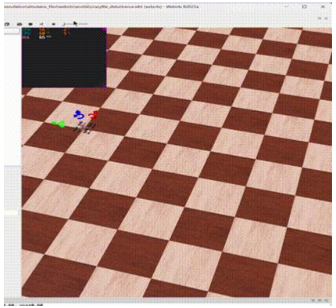
Controls: ESO Implementation

Sprint 1 ESO:

- None-state-space model based
- Decoupled thrust, attitude, and acc.
    - Ignored rotation-thrust coupling structure

Inaccurate output and representation of forces

Untunable and unstable in practice



Rebuilt ESO:

- State-space based 12-State ESO
    - [xyz, rpy, linear vel, disturbance forces]
- Observable, linearized at hover observer gain L
- Dynamics model

Controls: ESO Design Approach

Phase 1
Input & Sensing

- 1 — Att. State Meas. [x, y, z, r, p, y]
- 2 — Initialize ESO states
- 3.1 — Read control inputs (thrust in PWM)
- 3.2 — Read body rates (Gyro Ang. Vel.)

Phase 2
Model Prediction

- 4 — Quadrotor (CT) dynamics prediction
- 5 — State propagation
- 6 — Measurement prediction

Phase 3
Correction & Estimation

- 7 — Innovation Computation
- 8 — Observer Correction
- 9 — Disturbance force extraction

4 Quadrotor (CT) Dynamics Pred.

$$\dot{z} = f(z, u, \omega)$$

$$\dot{p} = v$$

$$\dot{v} = -ge_3 + \frac{1}{m} R(\phi, \theta, \psi) \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} + d$$

$$\dot{\eta} = E(\eta)\,\omega$$

$$\dot{d} = 0$$

z: full ESO state
p: position
v: lin. vel.
η: attitude rpy

e3: [0 0 1]^T
m: mass
R: rot. matrix
d: distur. acc. vec.
E: Euler kine. Mat.
ω: ang. vel.

- Predicts nominal motion from states
- Thrust–attitude–acceleration coupling
- Separates known dynamics from unknown disturbances

7 Innovation Computation

$$r = y - \hat{y}$$

$$y = \begin{bmatrix} p \\ \eta \end{bmatrix} = [x, y, z, \phi, \theta, \psi]$$

$$\hat{y} = \begin{bmatrix} \hat{p}^- \\ \hat{\eta}^- \end{bmatrix}$$

r: innovation
y: measurement vec.
y^: predicted meas.

- Quantifies model mismatch
- Encodes wind, thrust bias, unmodeled dynamics…

8 Observer correction

$$\hat{z}^+ = \hat{z}^- + Lr$$
$$\hat{d}^+ = \hat{d}^- + L_d r$$

$$A^\top P + PA - PC^\top R^{-1} CP + Q = 0$$
$$L_d = P_d C^\top R^{-1}$$

z^+: pred. state
z^-: corrected state
Ld: disturbance gain
P: Riccati solution

- Maps output error into unmeasured states
- Infers disturbance acceleration indirectly
- Stable observer error dynamics

9 Disturbance force extraction

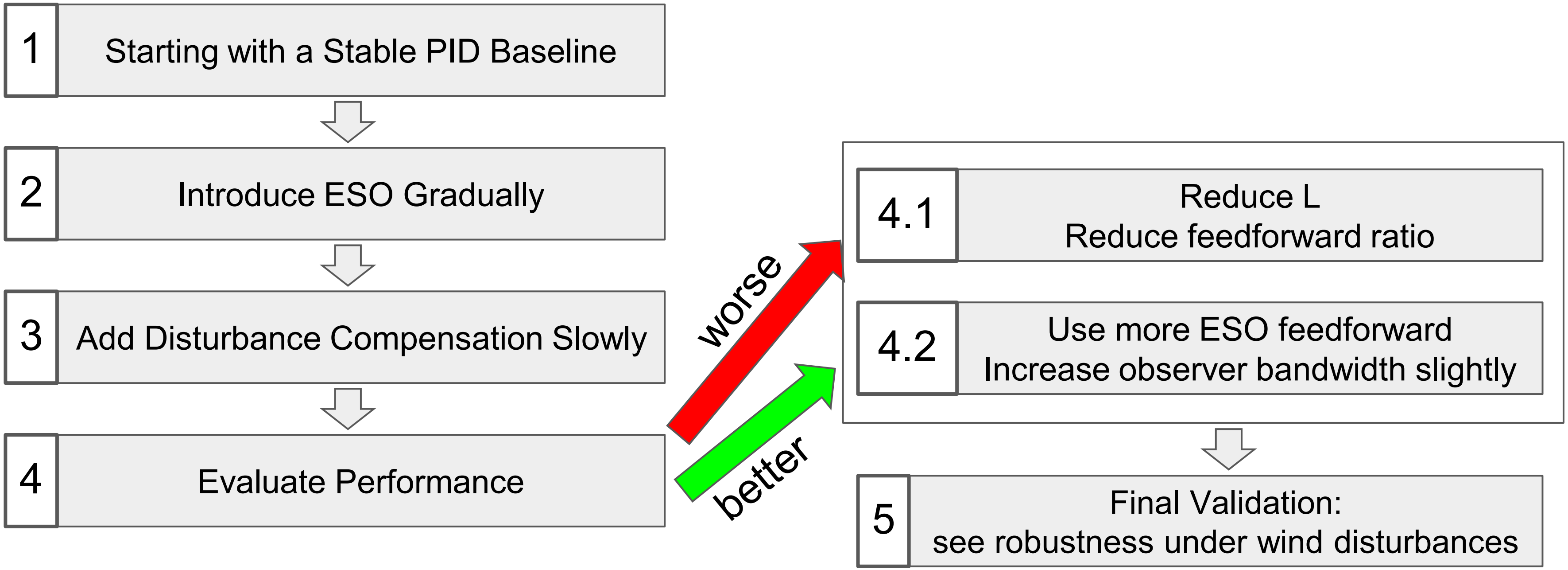
$$\begin{bmatrix} \hat{p}^+ \\ \hat{v}^+ \\ \hat{\eta}^+ \\ \hat{d}^+ \end{bmatrix} = \begin{bmatrix} \hat{p}^- \\ \hat{v}^- \\ \hat{\eta}^- \\ \hat{d}^- \end{bmatrix} + \begin{bmatrix} L_p \\ L_v \\ L_\eta \\ L_d \end{bmatrix} r$$

- Extract the Ld terms

Get disturbance forces in x y z directions

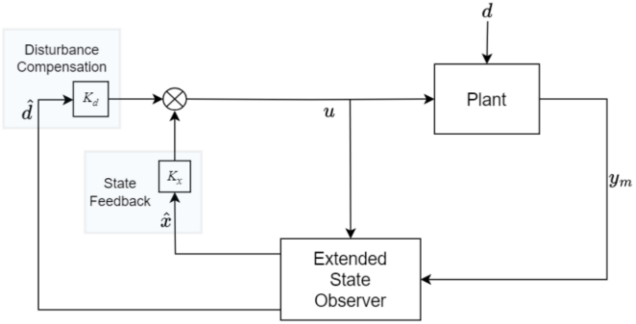
N

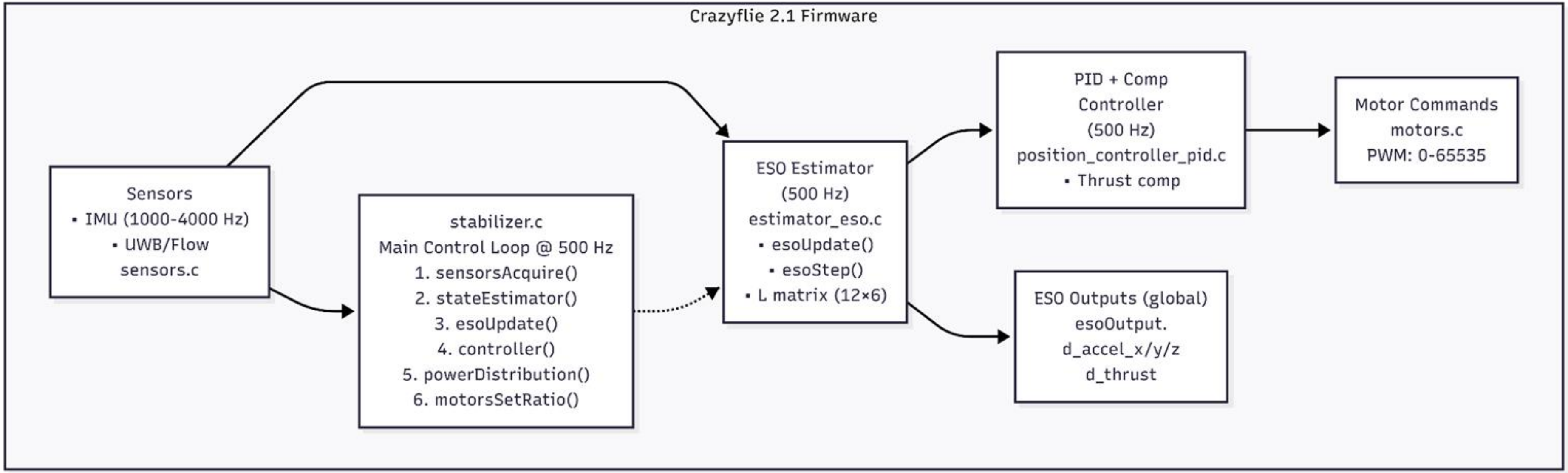
ESO Tuning Strategy



A

ESO + PID Firmware Structure





N

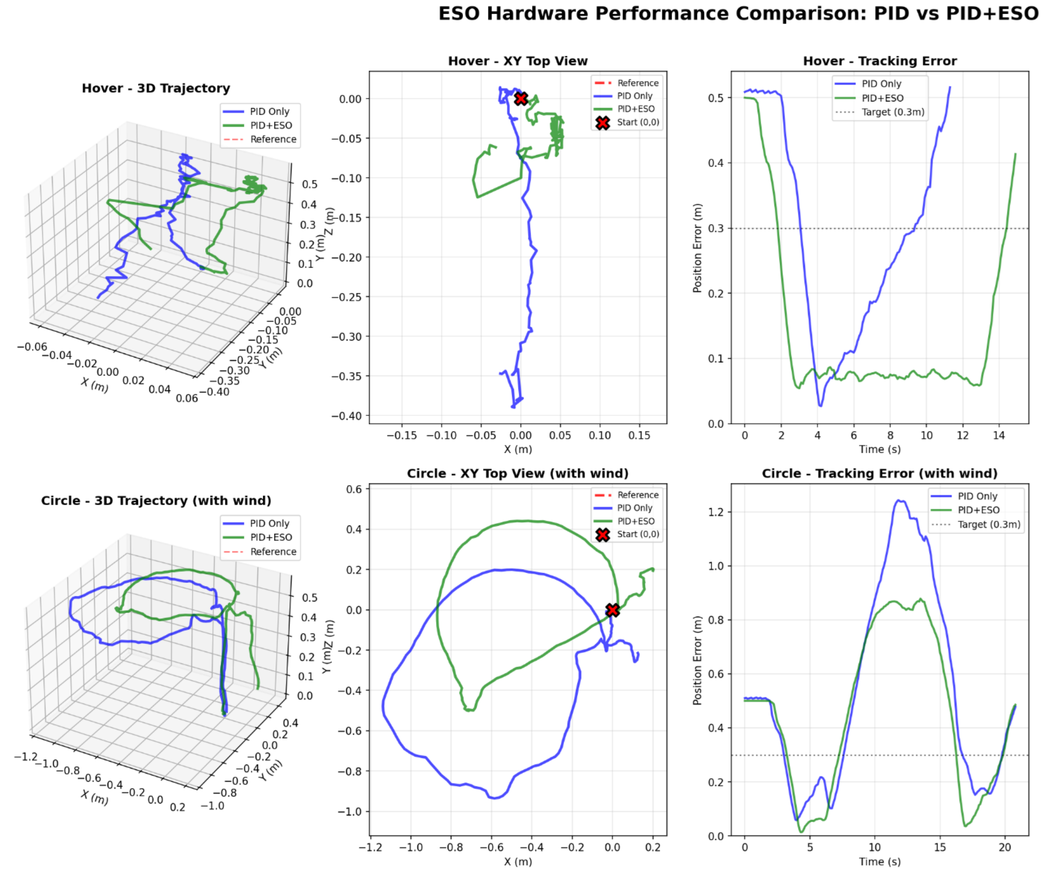
ESO Usages and Limitations

Powerful ADRC tool, helpful for drone in unknown, disruptive environments

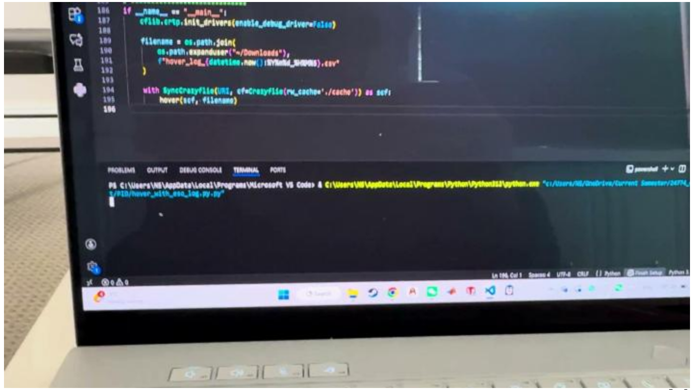
12 state: can observe x y disturbances, can output thrust as vertical disturbance compensation, lacks ability to perform x y compensation due to lack of ang. vel. disturbance evaluation.

15 state: require more computation power, may be dangerous for real time CF calculations, more lag in tracking.

Benchmarking: PID with ESO on Hardware



No ESO, hover



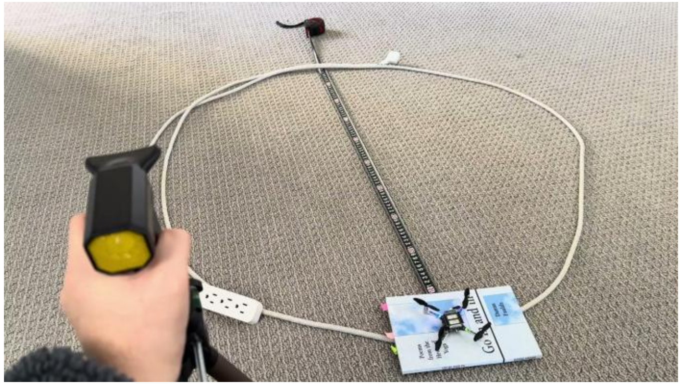
ESO, hover

N

Benchmarking: PID with ESO on Hardware

Metric	PID Only	PID+ESO	Improvement
Mean Err. (m)	0.534	0.447	16.2%
Max Err. (m)	1.244	0.878	29.4%
Std Dev. (m)	0.370	0.289	21.7%
RMSE (m)	0.649	0.532	17.9%

No ESO, 1m circle



ESO, 1m circle

LQR Design & Setup

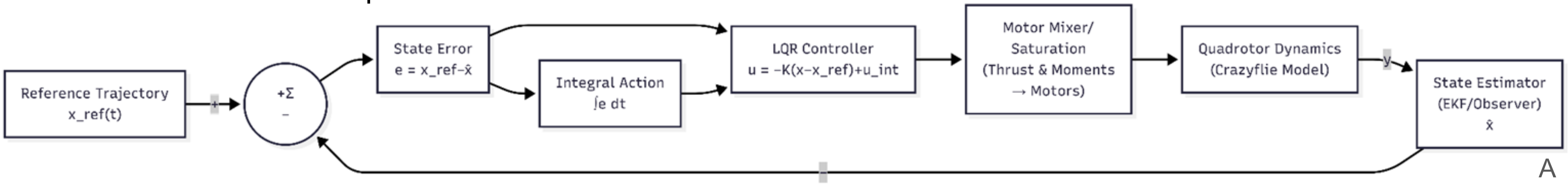
- **Linearize** quadrotor dynamics around **hover** equilibrium
- 12-state model (position, velocity, attitude, angular rates)

$$x = [p_x, p_y, p_z,\ v_x, v_y, v_z,\ \phi, \theta, \psi,\ p, q, r]^T$$

- **LQR Input** (state error) $e = x - x_{ref}$
- **LQR Output** (virtual thrust N + body torques Nm) $u = [\Delta T,\ \tau_\phi,\ \tau_\theta,\ \tau_\psi]^T$
  - Integral action added externally to remove steady-state bias $u_{\text{int}} = -K_i \int (p - p_{\text{ref}})\, dt$
- Map Thrust and Torque -> Motor Velocities
- Cost Function $J = \int (x^T Q x + u^T R u)\, dt$

Q: state tracking weights
R: control effort penalty

- Discrete-time implementation at fixed simulation control rate



A

LQR Performance & Limitations

What We Tested

- Hover stabilization
- Circular trajectory tracking with ESO

Observed Performance

- Stable hover under nominal conditions
- Smooth tracking for circle trajectory tracked better than PID

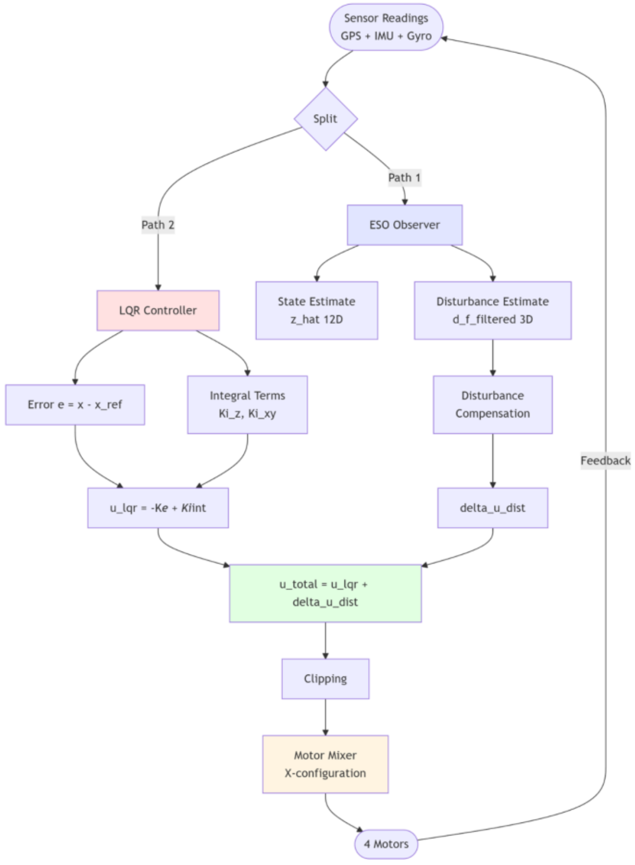
Key Limitations

- Sensitive to wind and unmodeled disturbances
- Steady-state bias without integral compensation
- Performance depends on model accuracy

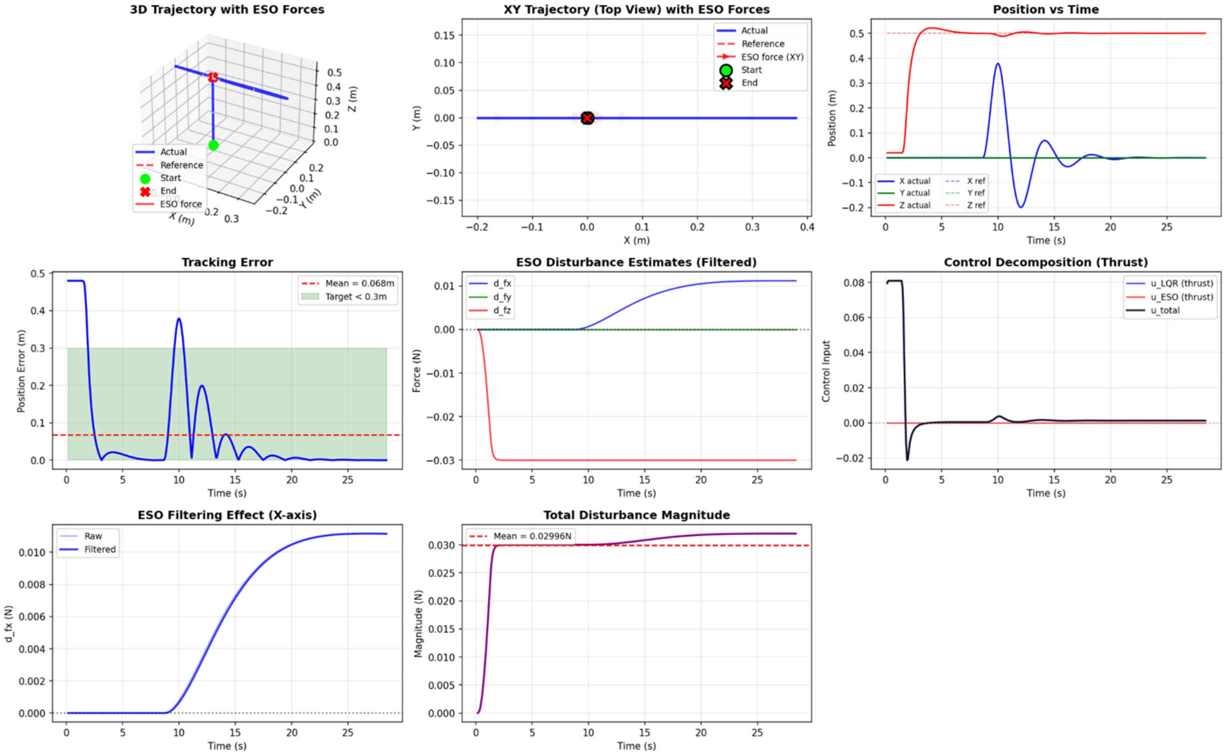
Takeaway

- LQR works well **when the model is accurate**
- Motivates the need for **disturbance-aware control (ESO)**

ESO + LQR



ESO+LQR Trajectory | Mean Error: 0.068m | Yaw Drift: 0.00°



TinyMPC Design & Setup

- Cost Function

$$J = \sum_{k=0}^{N-1} \left[(x_k - x_k^{ref})^T Q (x_k - x_k^{ref}) + u_k^T R u_k \right] + (x_N - x_N^{ref})^T Q_f (x_N - x_N^{ref})$$

- Control inputs constraints (normalized units)

$$-5.00\text{E-}02 \le \Delta T_k \le +5.00\text{E-}02$$
$$-1.50\text{E-}02 \le \tau_{x,k}, \tau_{y,k} \le +1.50\text{E-}02$$
$$-5.00\text{E-}02 \le \tau_{z,k} \le +5.00\text{E-}02$$

- State Constraints

State

$$x_k = \begin{bmatrix} p_x & p_y & p_z & v_x & v_y & v_z & \phi & \theta & \psi & p & q & r \end{bmatrix}^T$$

Velocity bounds

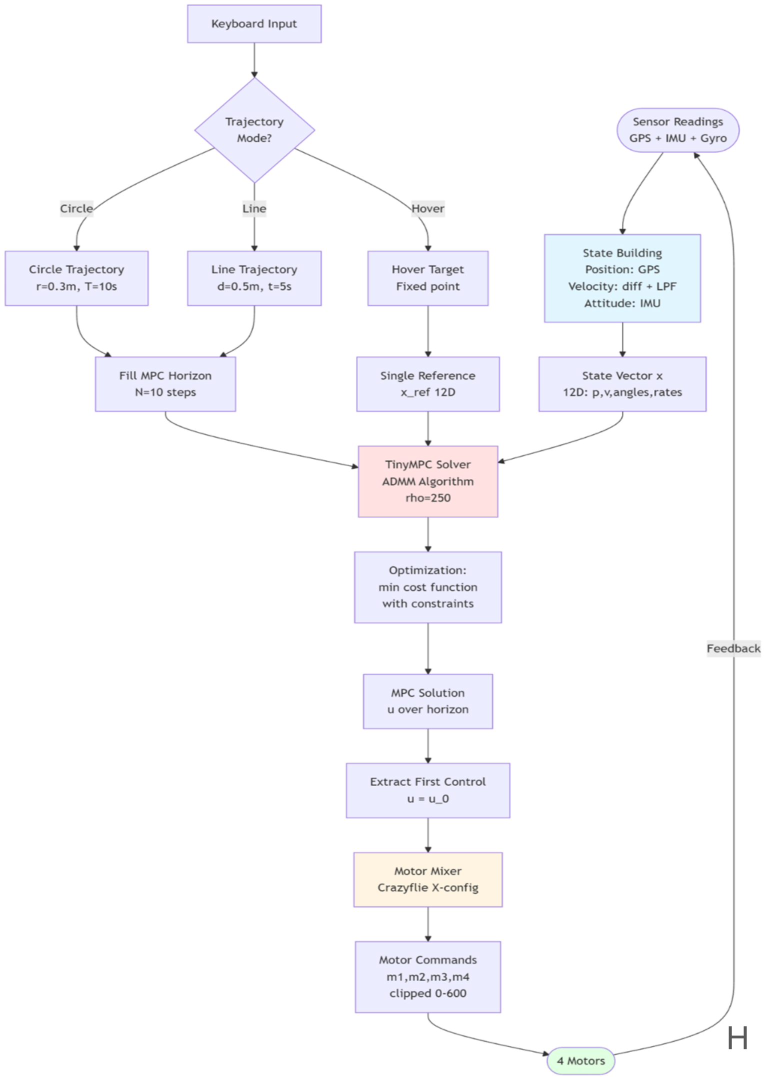
$$-2.00\text{E+}00 \le v_x, v_y, v_z \le +2.00\text{E+}00 \text{ [m/s]}$$

Attitude bounds

$$-1.50\text{E-}01 \le \phi, \theta \le +1.50\text{E-}01 \text{ [rad]}$$

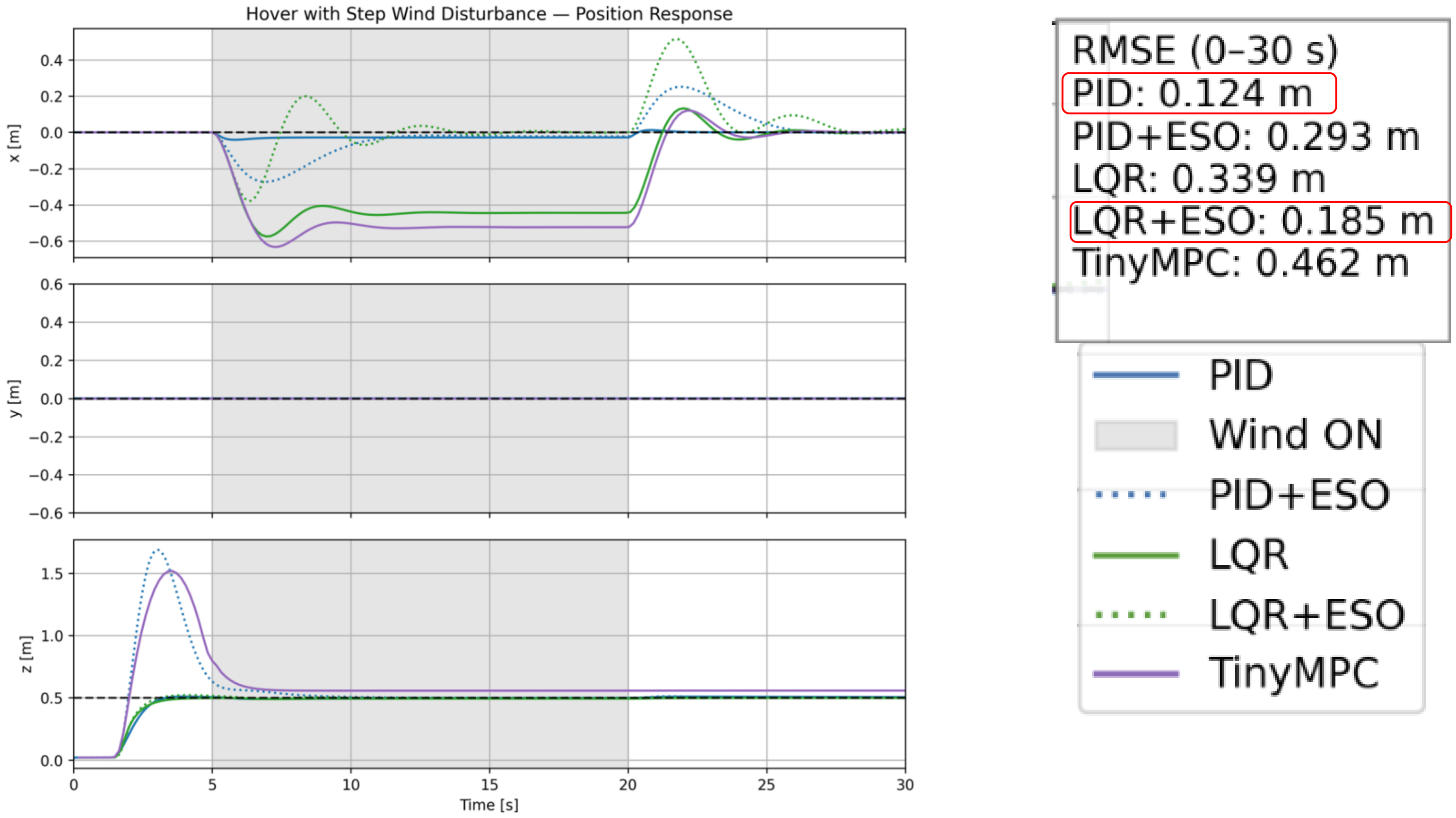
- TniyMPC (condensed QP Formulation)

$$\min_{\mathbf{u}} \frac{1}{2} \mathbf{u}^T H \mathbf{u} + f^T \mathbf{u} \quad \text{s.t. } u_{\min} \le u_k \le u_{\max}$$



H

Benchmarking - Simulation Controllers - Hover

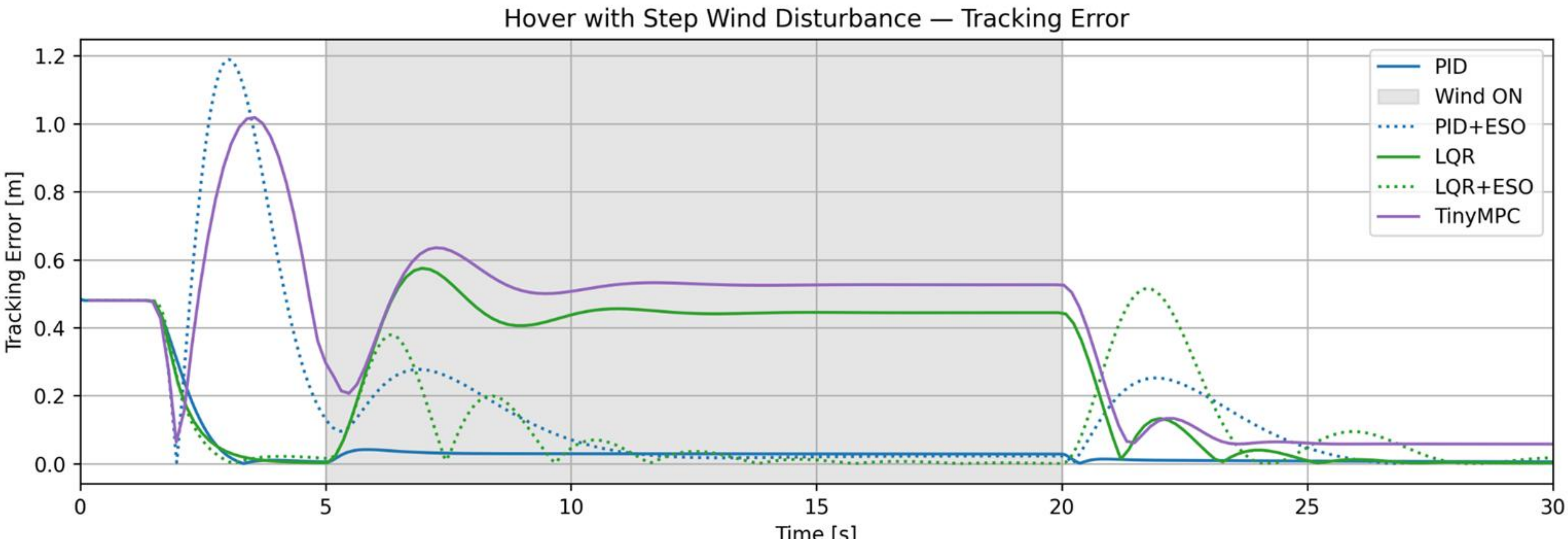


A

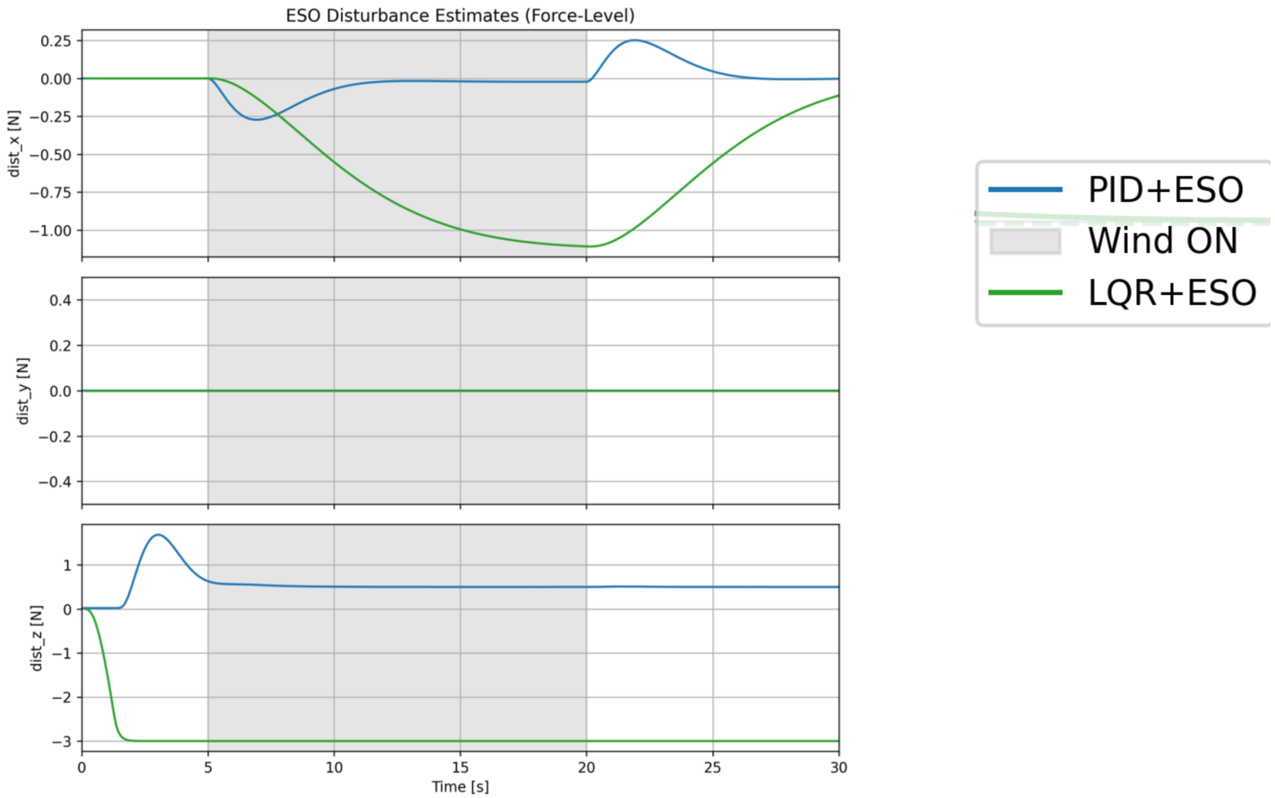
Benchmarking - Simulation Controllers - Hover

RMSE (0–30 s)
PID: 0.124 m
PID+ESO: 0.293 m
LQR: 0.339 m
LQR+ESO: 0.185 m
TinyMPC: 0.462 m

||x − x_ref|| vs time

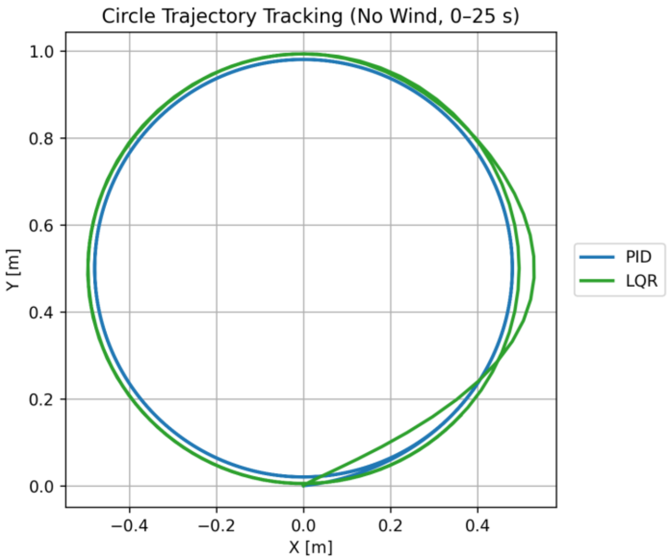


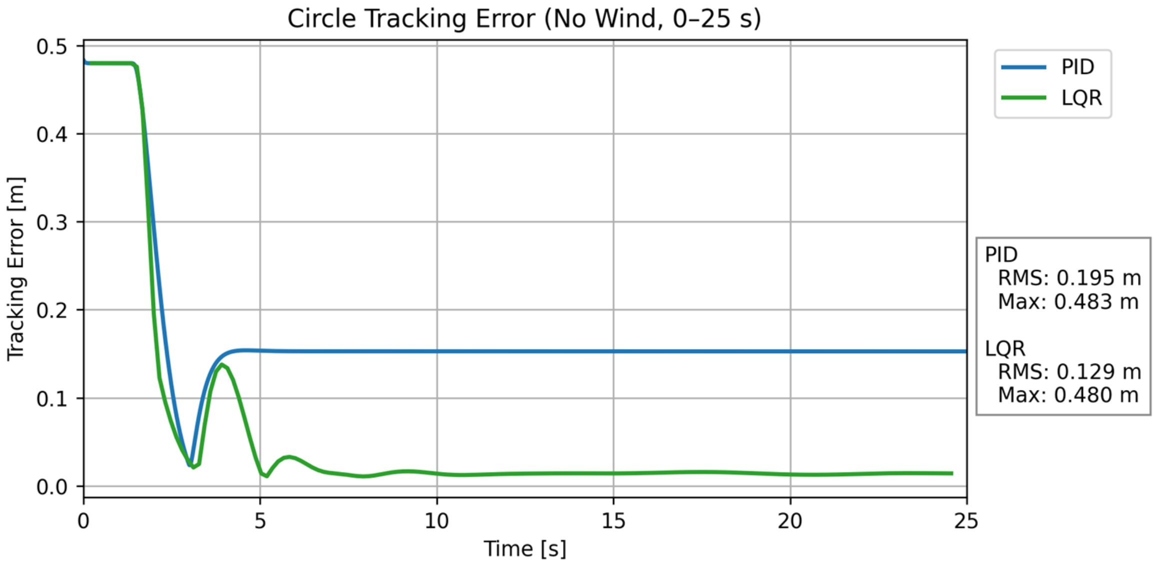
Benchmarking - Simulation Controllers - ESO_Sim



A

Benchmarking - Simulation Controllers - Circle





A

Findings & Conclusions

Key Findings

- **PID** shows smoother tracking behavior but has higher steady-state error
- **LQR** achieves lower tracking error but is more aggressive and oscillatory
- **ESO** significantly improves disturbance rejection for both controllers, effective in hardware

Controller Comparison

- **Best robustness under wind: PID + ESO**
- **Best tracking accuracy: LQR + ESO**

Overall Conclusion

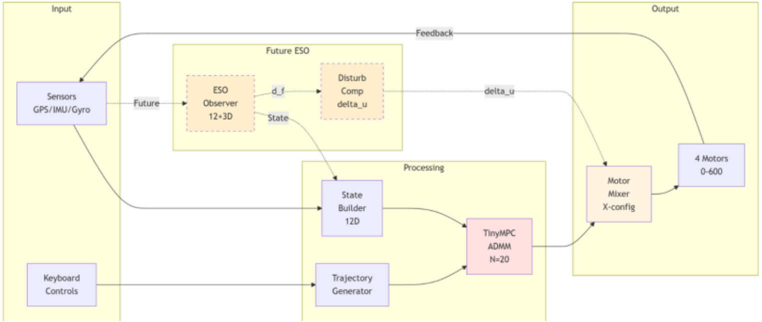
- **ESO provides a modular improvement** without redesigning the base controller
- **LQR + ESO offers the best performance–robustness trade-off** in simulation

A

Future Work

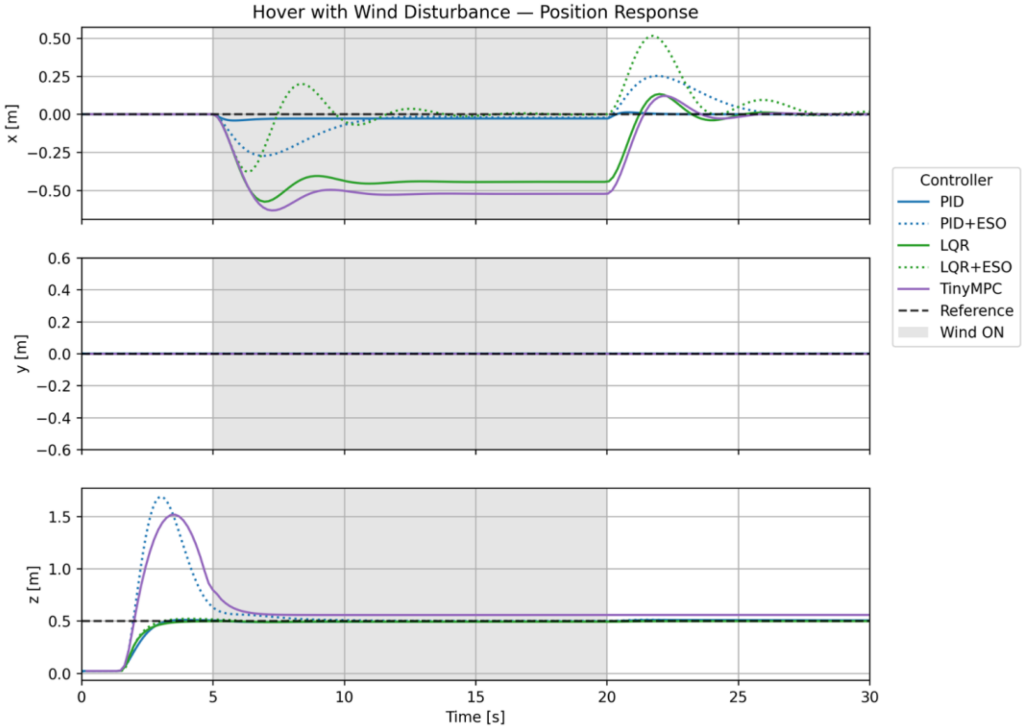
1. ESO
   a. Develop 15-state, horizontal disturbance compensation version for full ADRC
   b. Perform hardware measured force disturbance and accurately map to ESO

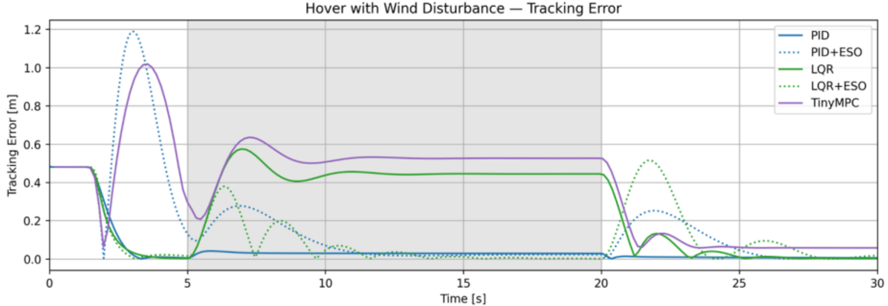
2. LQR+ESO
   a. Tune on firmware, currently spins horizontally
   b. Test More Trajectory
3. MPC+ESO
   a. Combine ESO with current MPC controller



N

Thank you!





Hover with Wind Disturbance — Quantitative Performance Metrics

	RMSE [m]	Max Error [m]	Std Dev [m]
PID	0.124	0.483	0.112
PID+ESO	0.293	1.189	0.244
LQR	0.339	0.575	0.209
LQR+ESO	0.185	0.515	0.151
TinyMPC	0.462	1.019	0.247