Learning-Based Minimally-Sensed Fault-Tolerant Adaptive Flight Control

Michael O'Connell*, Joshua Cho*, Mattew Anderson, and Soon-Jo Chung

*Equal contribution


Michael O'Connell


Joshua Cho


Matthew Anderson


Soon-Jo Chung

ARCLab
Autonomous Robotics and Control Lab

Caltech

With multiple actuators we achieve redundancy, but how do we model and respond to failures optimally in real-time?

Previous Work

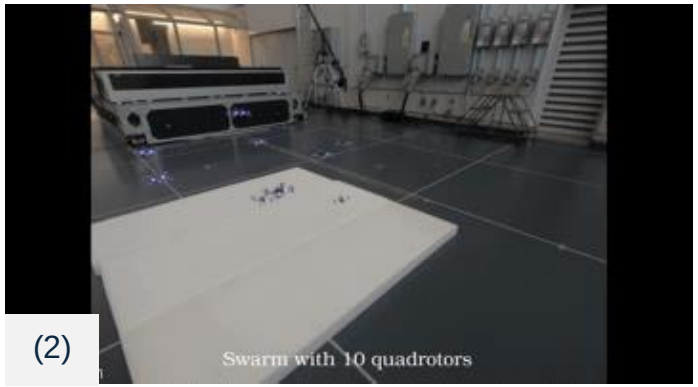
Neural-Lander (1) and Neural-Swarm (2): Deep learning-based control in a fixed environment

$$\dot{x} = f(x) + B_0 u + \underbrace{g(x,u)}_{\text{unknown}} + d(t)$$

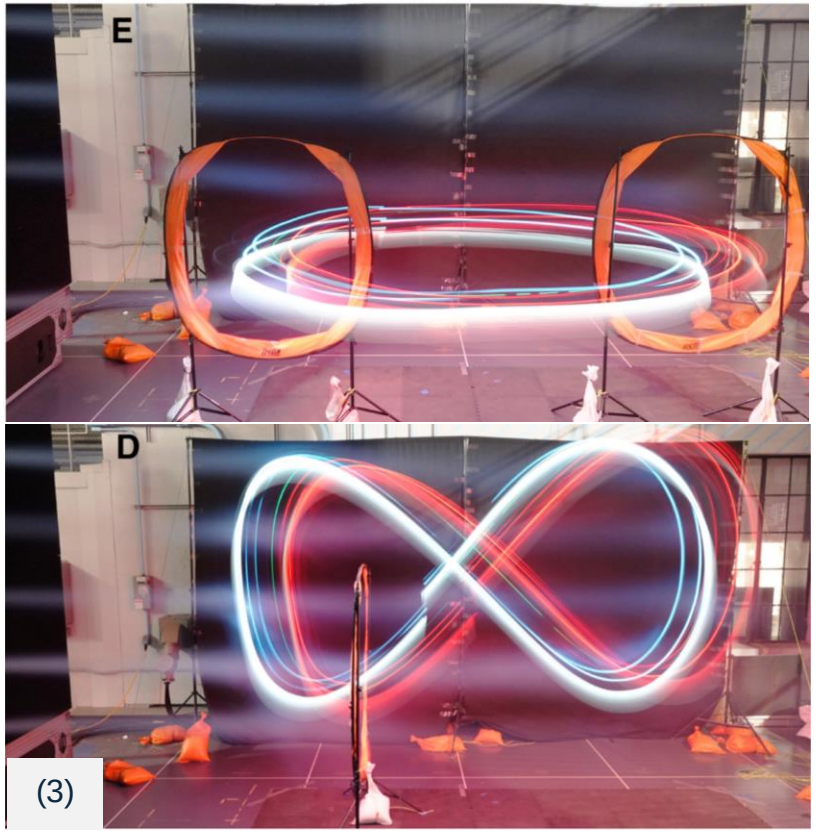
Neural-Fly (3): Learn and adapt to changing environments (e.g., different wind conditions)

$$\dot{x} = f(x) + B_0 u + \underbrace{g(x,u,t)}_{\text{unknown}} + d(t)$$

(1)



(2)



(3)

Neural-Fly

general robot dynamics model:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + g(q) = u + f(q,\dot{q},w)$$

A model of the unknown dynamics is built in the form:

$$f(q,\dot{q},w) \approx \phi(q,\dot{q})a(w)$$

ϕ is a learned basis function shared by all wind conditions and a is a set of linear coefficients that is updated for each condition.

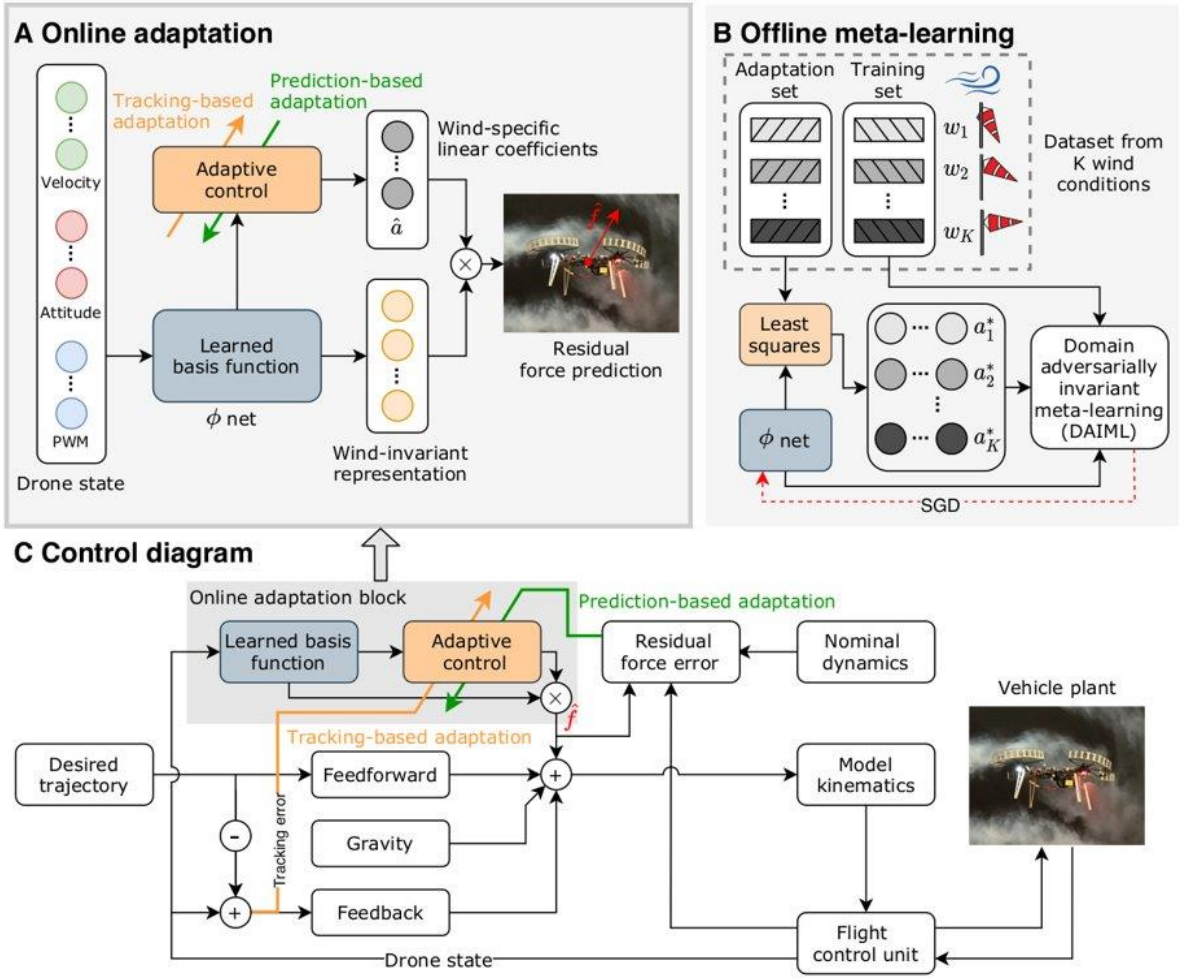
Neural-Fly's DAIML algorithm solves the following adversarial optimization problem offline:

$$\max_{h} \min_{\phi, a_1, \cdots, a_K} \sum_{k=1}^{K} \sum_{i=1}^{N_k} \left(\left\| y_k^{(i)} - \phi(x_k^{(i)}) a_k \right\|^2 - \alpha \cdot \text{loss}\left(h(\phi(x_k^{(i)})), k \right) \right)$$

Optimal a is computed using the following Kalman-filter based composite adaptation law and covariance update equations

$$\dot{\hat{a}} = \underbrace{-\lambda \hat{a}}_{\text{regularization term}} \underbrace{-P\phi^\top R^{-1}(\phi\hat{a} - y)}_{\text{prediction error term}} \underbrace{+P\phi^\top s}_{\text{tracking error term}}$$

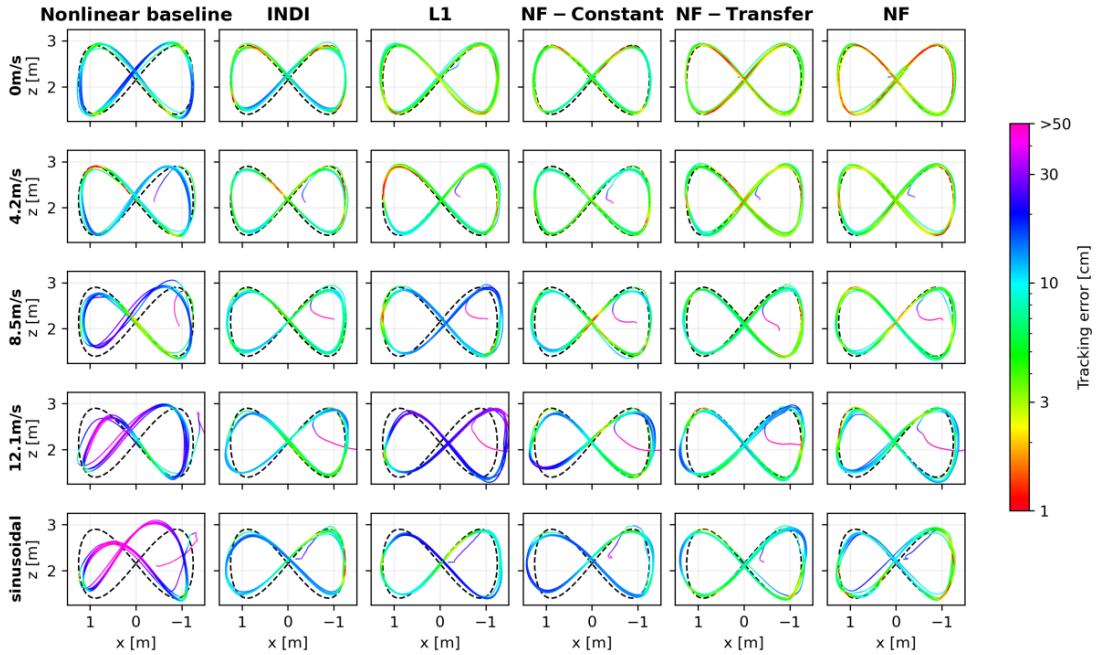
$$\dot{P} = -2\lambda P + Q - P\phi^\top R^{-1}\phi P$$

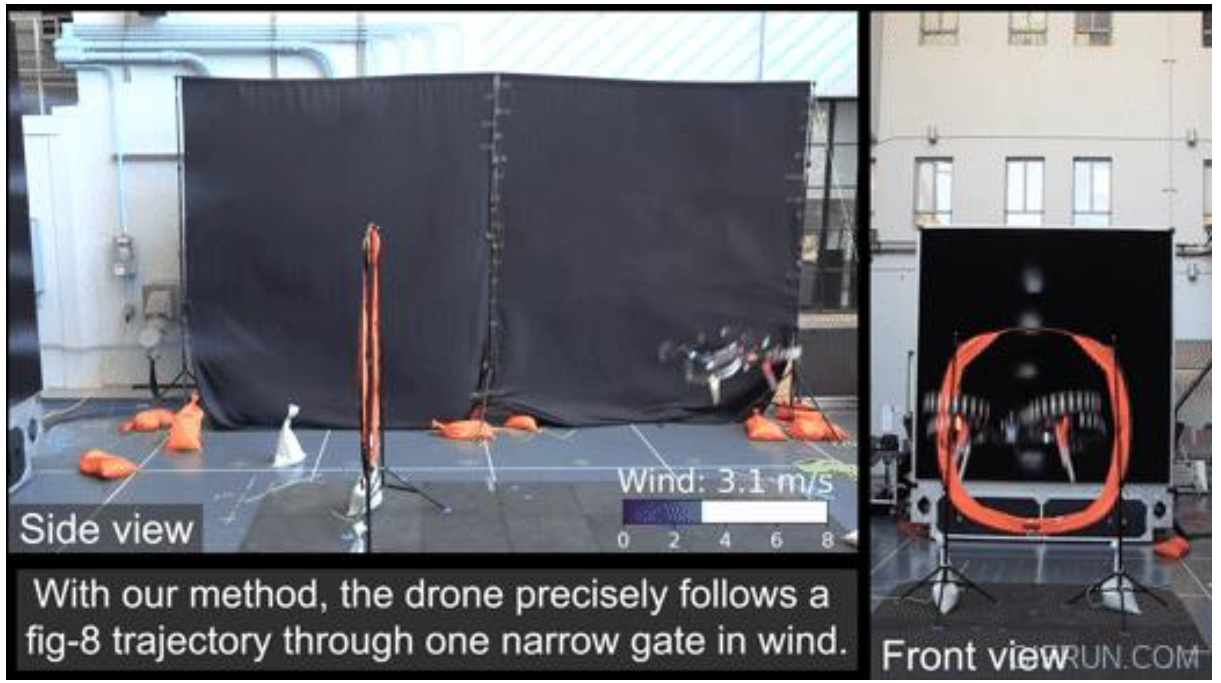


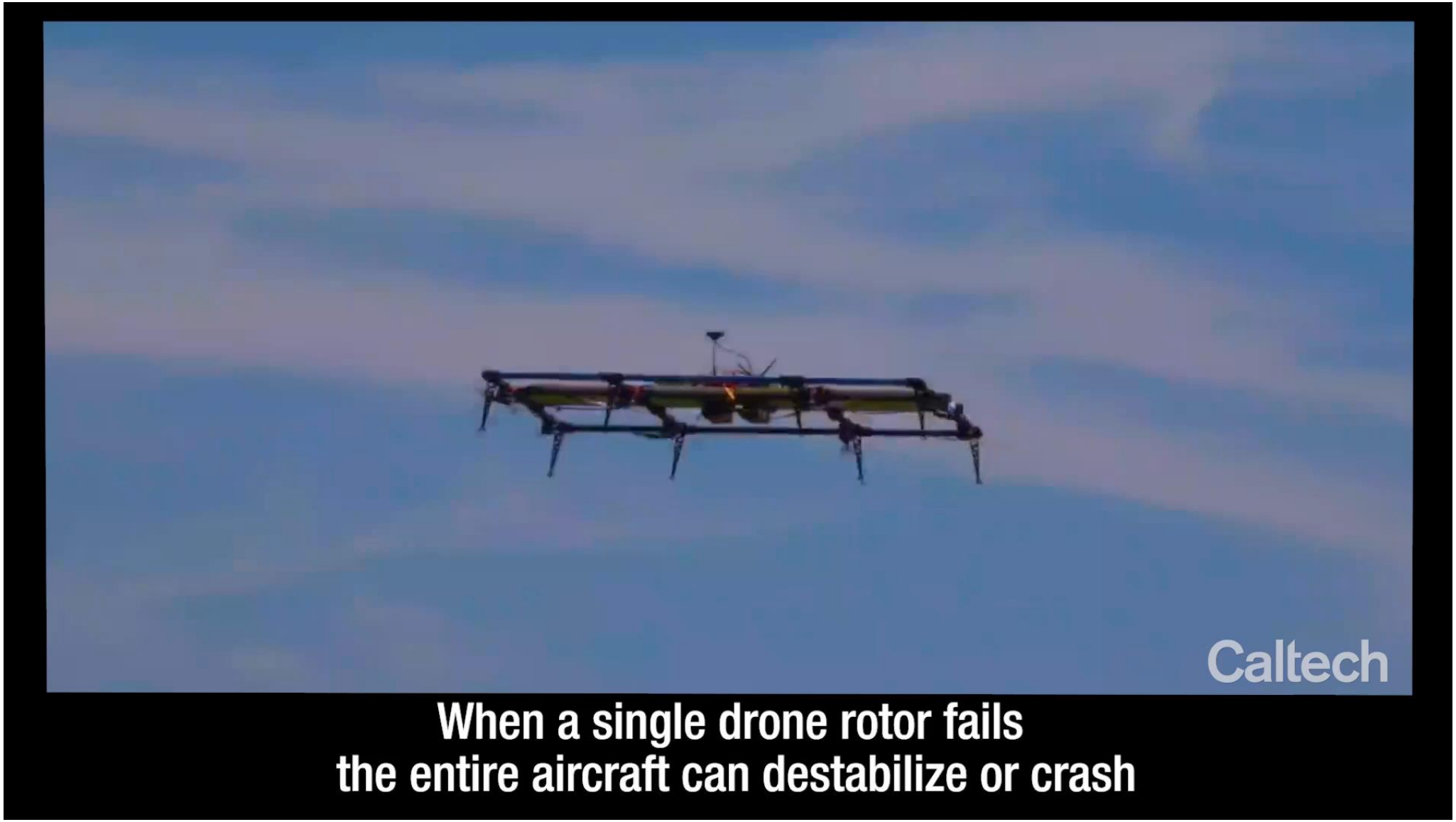
Previous Work

Neural-Fly control law:

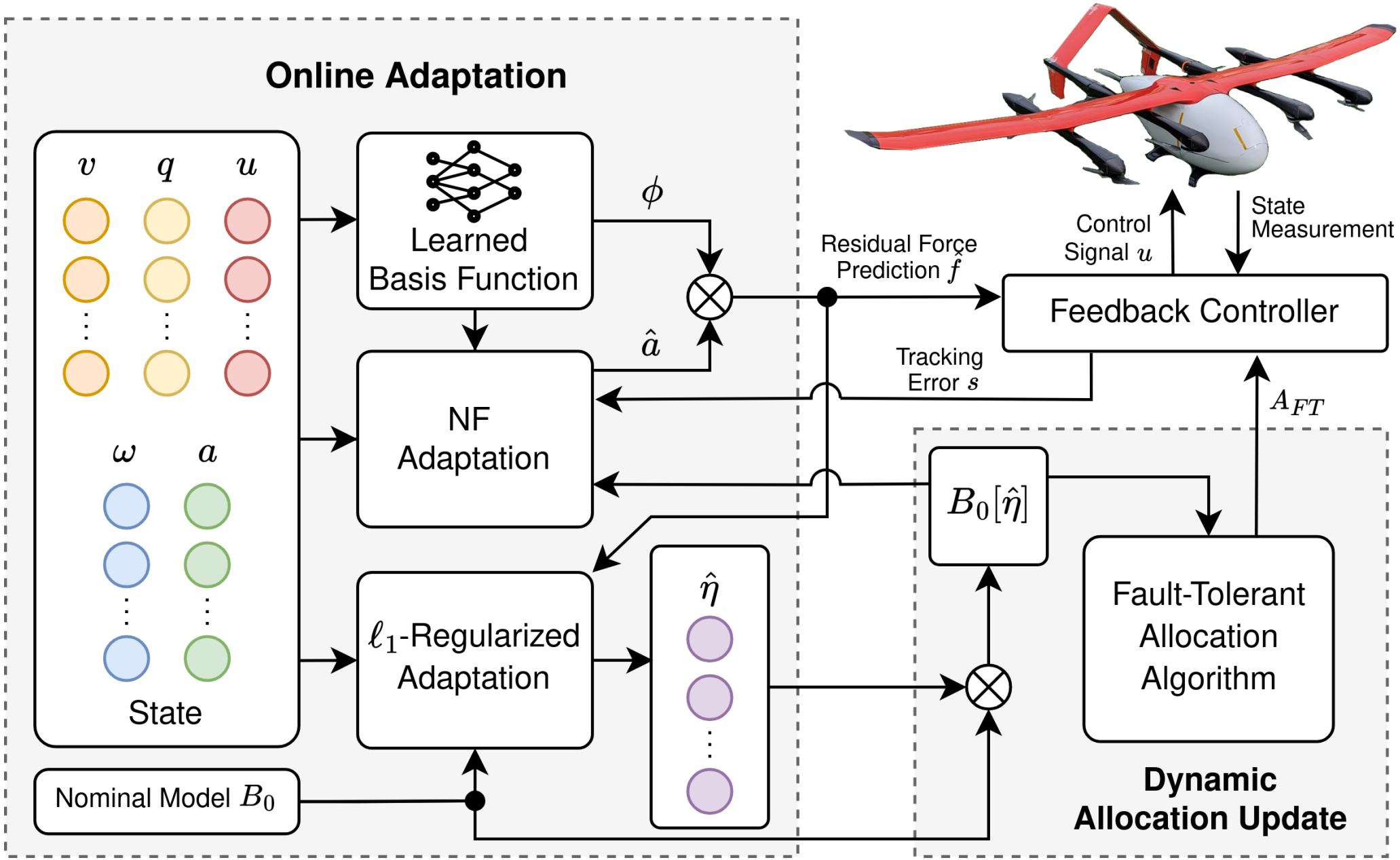
$$u_{\mathrm{NF}} = \underbrace{M(q)\ddot{q}_r + C(q,\dot{q})\dot{q}_r + g(q)}_{\text{nominal model feedforward terms}} \quad \underbrace{-Ks}_{\text{PD feedback}} \quad \underbrace{-\phi(q,\dot{q})\hat{a}}_{\text{learning-based feedforward}}$$







https://www.caltech.edu/about/news/virtual-sensors-help-aerial-vehicles-stay-aloft-when-rotors-fail


Online Adaptation
v q u
ω a
State
Learned Basis Function
ϕ
$\hat{a}$
NF Adaptation
ℓ_1-Regularized Adaptation
$\hat{\eta}$
Nominal Model B_0
Residual Force Prediction $\hat{f}$
Control Signal u
State Measurement
Feedback Controller
Tracking Error s
A_{FT}
$B_0[\hat{\eta}]$
Fault-Tolerant Allocation Algorithm
Dynamic Allocation Update

System Dynamics

Consider the following aircraft dynamics:

$$\dot{x} = f(x) + B_0 u + g(x, u, t) + d(t)$$

When $g(x, u, t) \approx 0$ this system is exponentially stabilized by the feedback linearization control law,

$$u = B_0^{-R}(-K\tilde{x} + \dot{x}_d - f(x))$$

Under actuator faults, $g(x, u, t)$ becomes $g(x, u, t) = B_0([\eta] - I)u + r(x, u, t)$

$$[\eta] = \begin{bmatrix} \eta_1(t) & 0 & \dots & 0 \\ 0 & \eta_2(t) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \eta_m(t) \end{bmatrix}$$

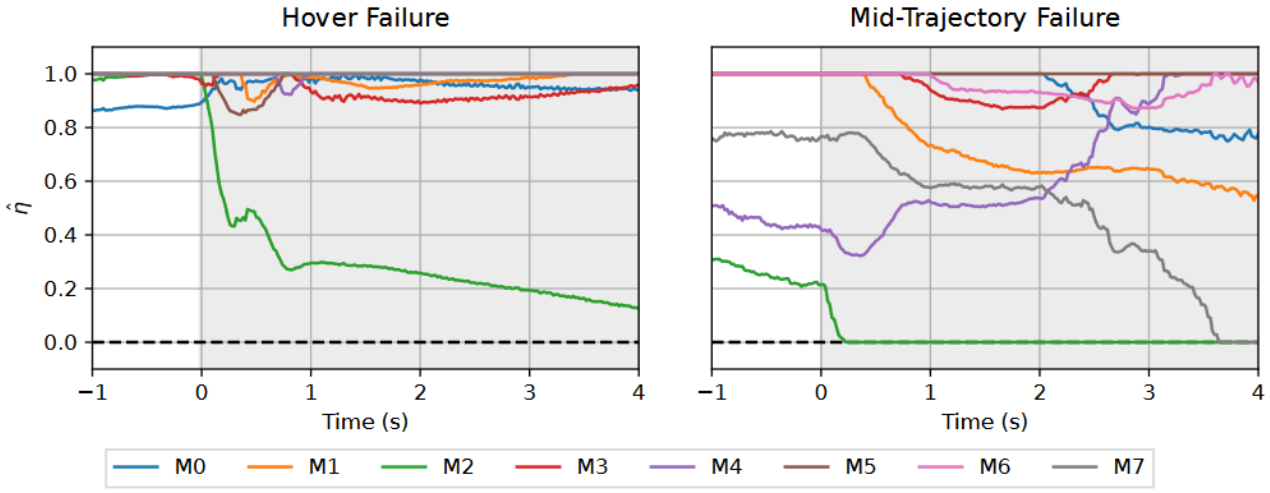
where $\eta_i(t) \in [0, 1]$ is the effectiveness factor for the *i*th actuator, and $r(x, u, t)$ denotes remaining force not captured.

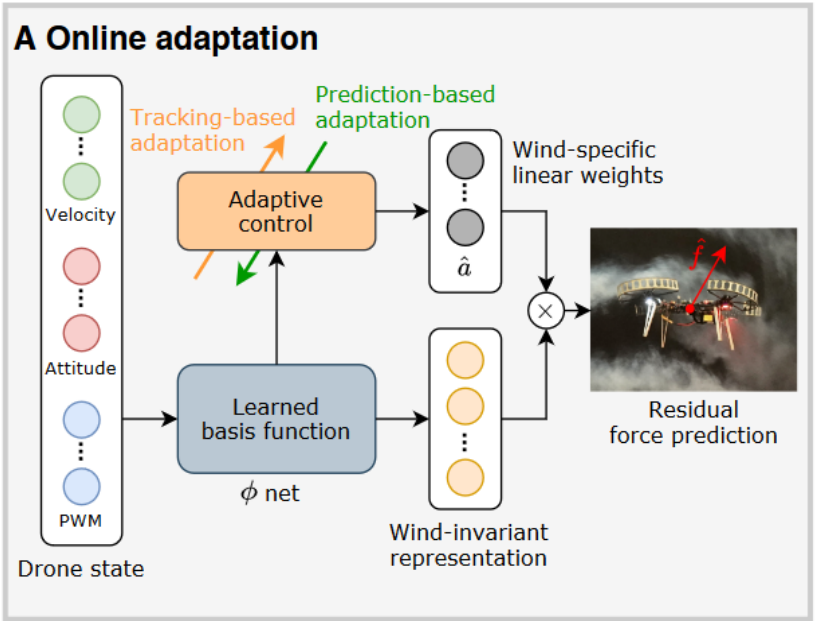
Motor Effectiveness Estimation

The unknown residual dynamics are modeled as:

$$\hat{g}_{\mathrm{NFFT}}(x,u,t) = \underbrace{B_0([\hat{\eta}]-I)u}_{\text{Estimate of model mismatch due to actuator faults}} + \underbrace{\phi(x,u)\hat{a}(t)}_{\text{Learned residual dynamics (Neural-Fly)}}$$

Effectiveness estimate without learned residual dynamics following rear left outer motor failure





Motor Effectiveness Estimation

Consider the following least squares loss function:

$$J_k(\hat{\eta}) = \sum_{i=0}^{k} e^{-\omega_f (t_k - t_i)} \left\| y - \hat{g}_{\mathrm{NFFT}} \right\|_2^2 + \gamma \left\| \hat{\eta} - 1 \right\|_1$$

Defining $\bar{\eta} = \hat{\eta} - 1$, we expand the loss function:

$$\begin{aligned} J_k(\bar{\eta}) &= \sum_{i=0}^{k} e^{-\omega_f (t_k - t_i)} \left(y_i - \hat{g}\right)^\top \left(y_i - \hat{g}\right) + \gamma \left\| \bar{\eta} \right\|_1 \\ &= \sum_{i=0}^{k} e^{-\omega_f (t_k - t_i)} \left(y_i^\top y_i - 2 y_i^\top \hat{g} + \hat{g}^\top \hat{g}\right) + \gamma \left\| \bar{\eta} \right\|_1 \\ &= \left(\sum_{i=0}^{k} e^{-\omega_f (t_k - t_i)} y_i^\top y_i \right) - 2 \left(\sum_{i=0}^{k} e^{-\omega_f (t_k - t_i)} y_i^\top B_0 U_i \right) \bar{\eta} \\ &\quad + \bar{\eta}^\top \left(\sum_{i=0}^{k} e^{-\omega_f (t_k - t_i)} U_i B_0^\top B_0 U_i \right) \bar{\eta} + \gamma \left\| \bar{\eta} \right\|_1 \end{aligned}$$

Where $\hat{g} = \hat{g}_{\mathrm{NFFT}}$. This is a convex function of $\bar{\eta}$, so the optimal $\bar{\eta}$ can be solved.

Control Allocation Problem

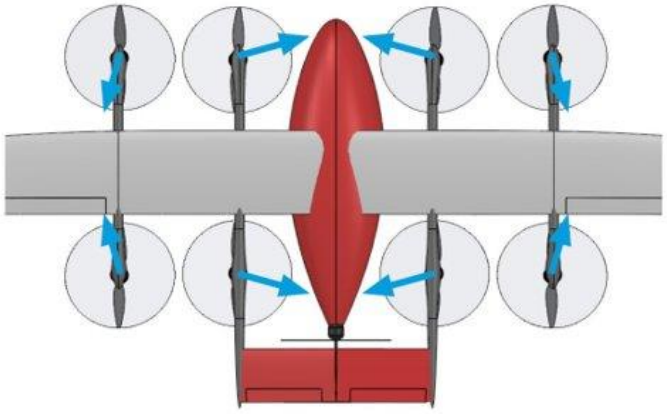
Must find a mapping from the desired control force, τ_d, to the desired actuator commands, u_d , such that the desired force is realized. Thus, the control allocation problem reduces to:

$$\begin{aligned} & u_d := A\tau_d \\ & \text{s.t. } B_0[\hat{\eta}]u_d = \tau_d \end{aligned} \implies B_0[\hat{\eta}]A = I$$

Our proposed allocation algorithm produces maximum control authority while maintaining the nominal performance characteristics of the system

$$\bar{A}_{\mathrm{FT}} = \underset{A}{\operatorname{argmax}} \sum_{i\in\{R,P,Y\}} B_i(hA_T + A_i) - \sum_{i\in\{R,P,Y\}} B_i(hA_T - A_i) + B_T(hA_T) - \|A - A_0\|_F$$

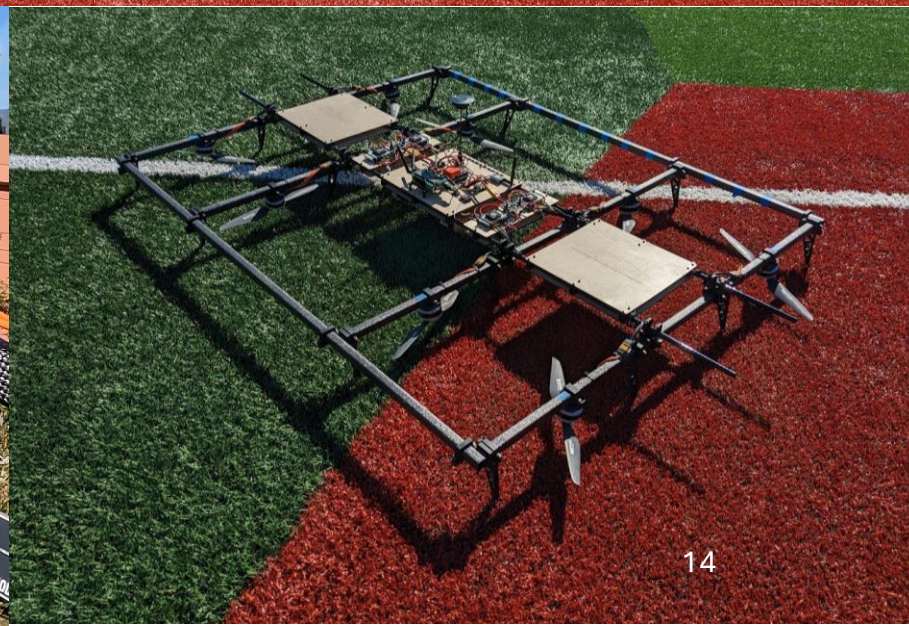
$$\begin{aligned}
\text{s.t.} \quad & B_{R,P,Y}A_T = 0,\ 0 \le A_T \le 1, && \left.\right\} \text{thrust constraint} \\
& B_{T,P,Y}(hA_T + A_R) = [h,0,0]^\top, && \left.\right\} \text{roll constraint} \\
& 0 \le (hA_T \pm A_R) \le \mathbb{1}, && \\
& B_{T,R,Y}(hA_T + A_P) = [h,0,0]^\top, && \left.\right\} \text{pitch constraint} \\
& 0 \le (hA_T \pm A_P) \le \mathbb{1}, && \\
& B_{T,R,P}(hA_T + A_Y) = [h,0,0]^\top, && \left.\right\} \text{yaw constraint} \\
& 0 \le (hA_T \pm A_Y) \le \mathbb{1}, &&
\end{aligned}$$



Autonomous Flying Ambulance 4.0

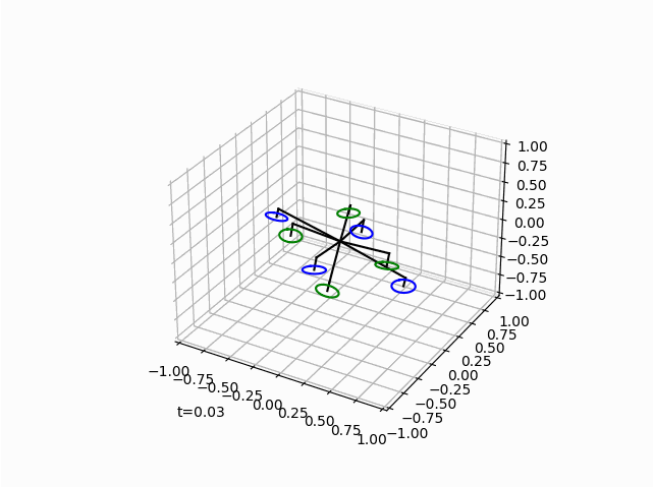

CALTECH

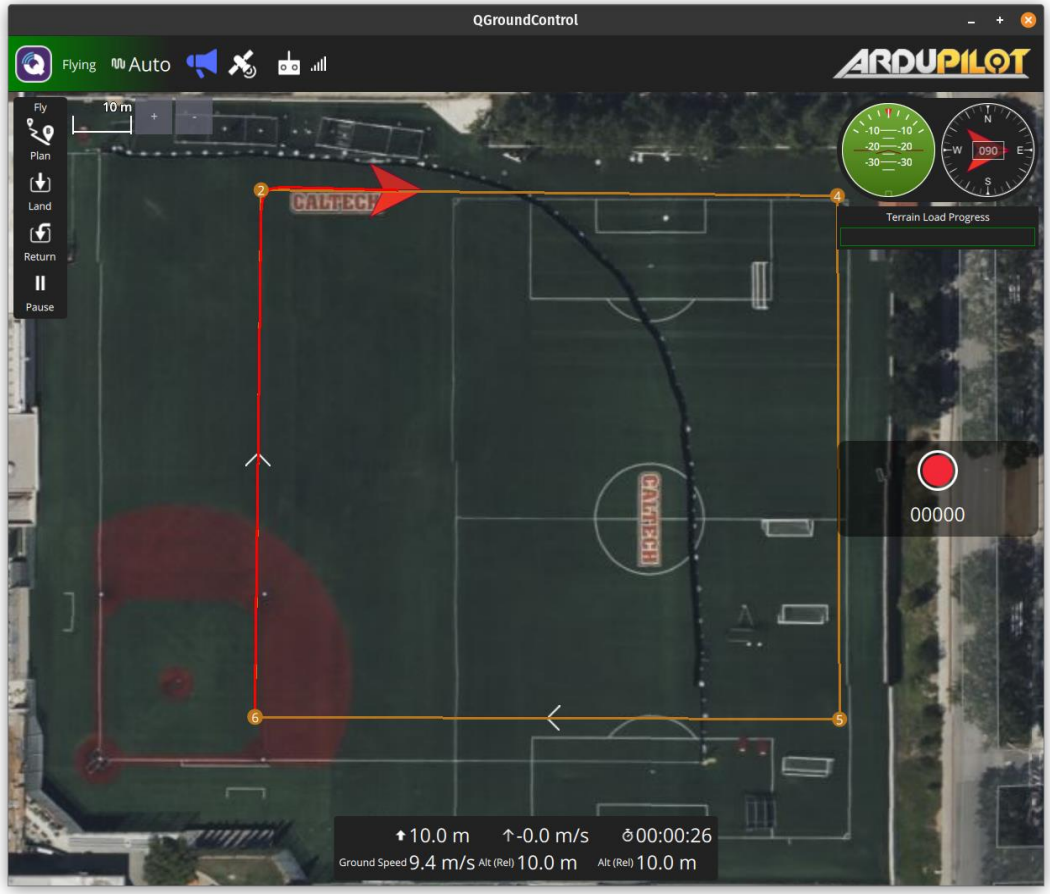
Test Platform - AFA 4.1



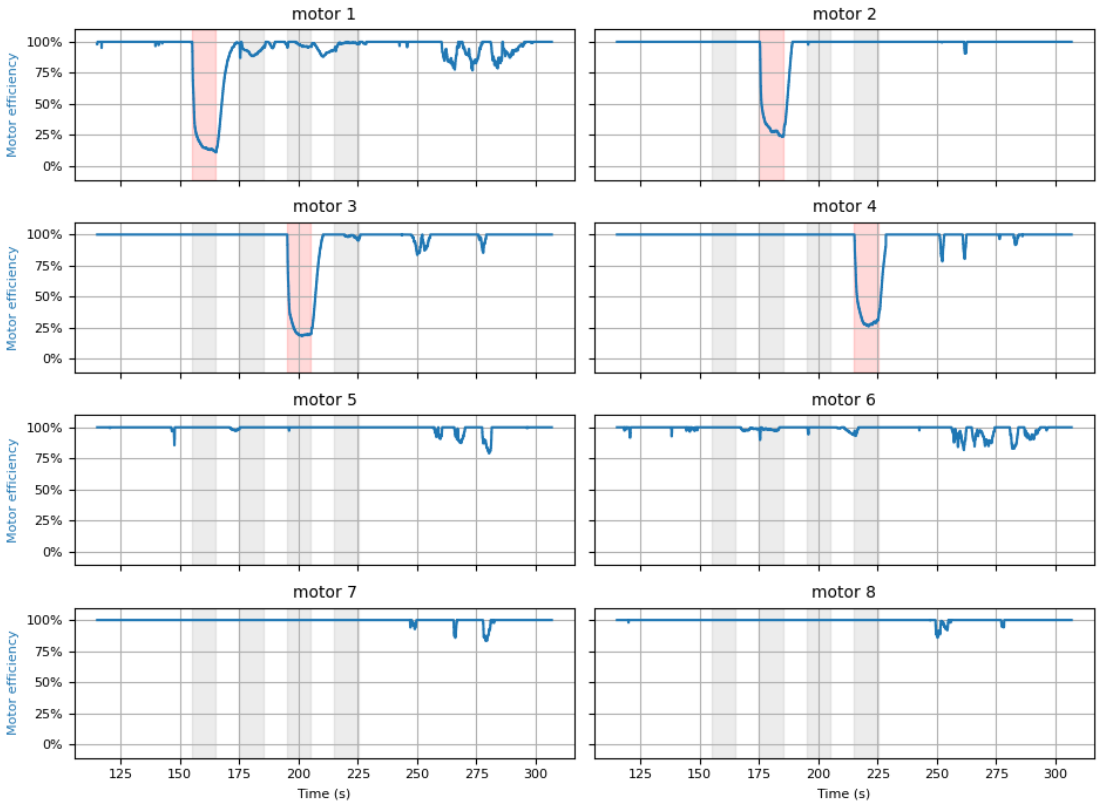
Simulation Environment

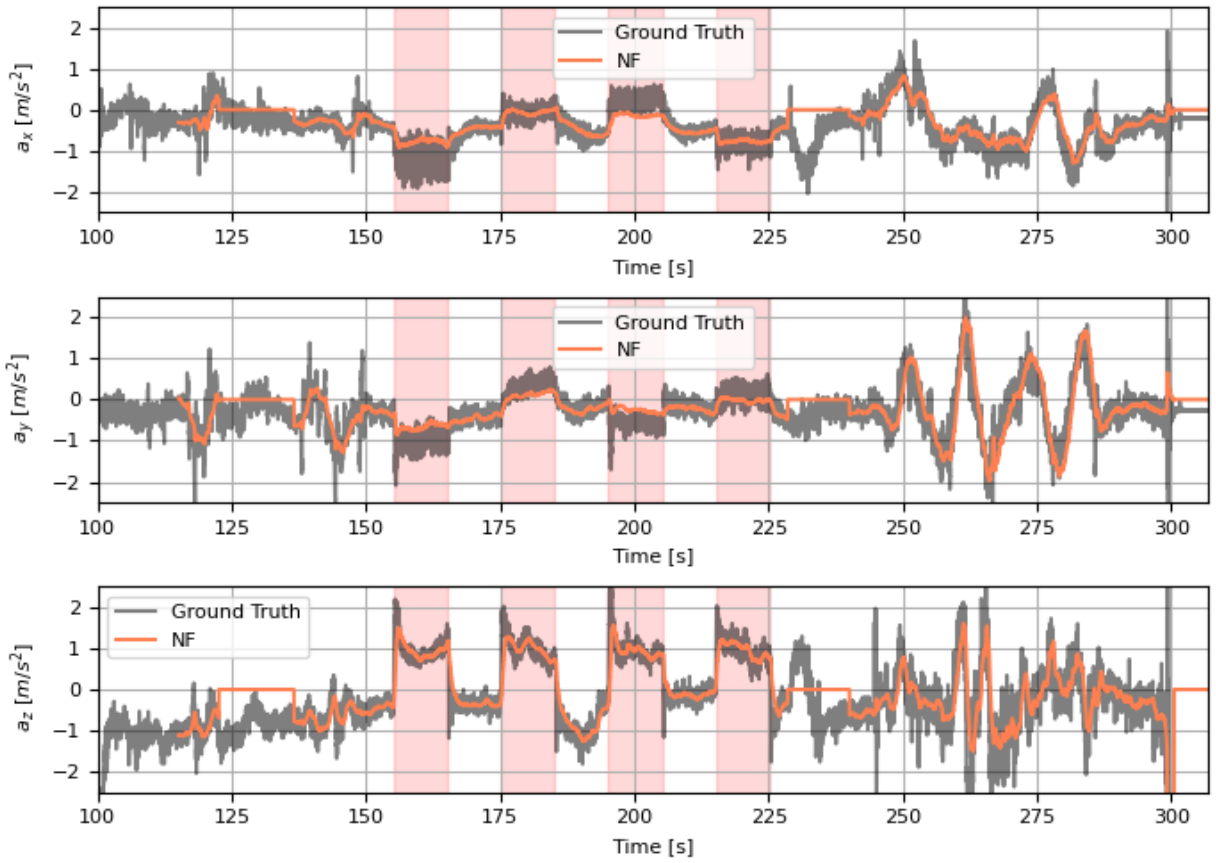
- ArduPilot Software-in-the-loop simulator
  - Octocopter dynamics model
  - Motor response time
  - aerodynamic drag, disturbances
  - Trajectory planning
  - Integration with ROS





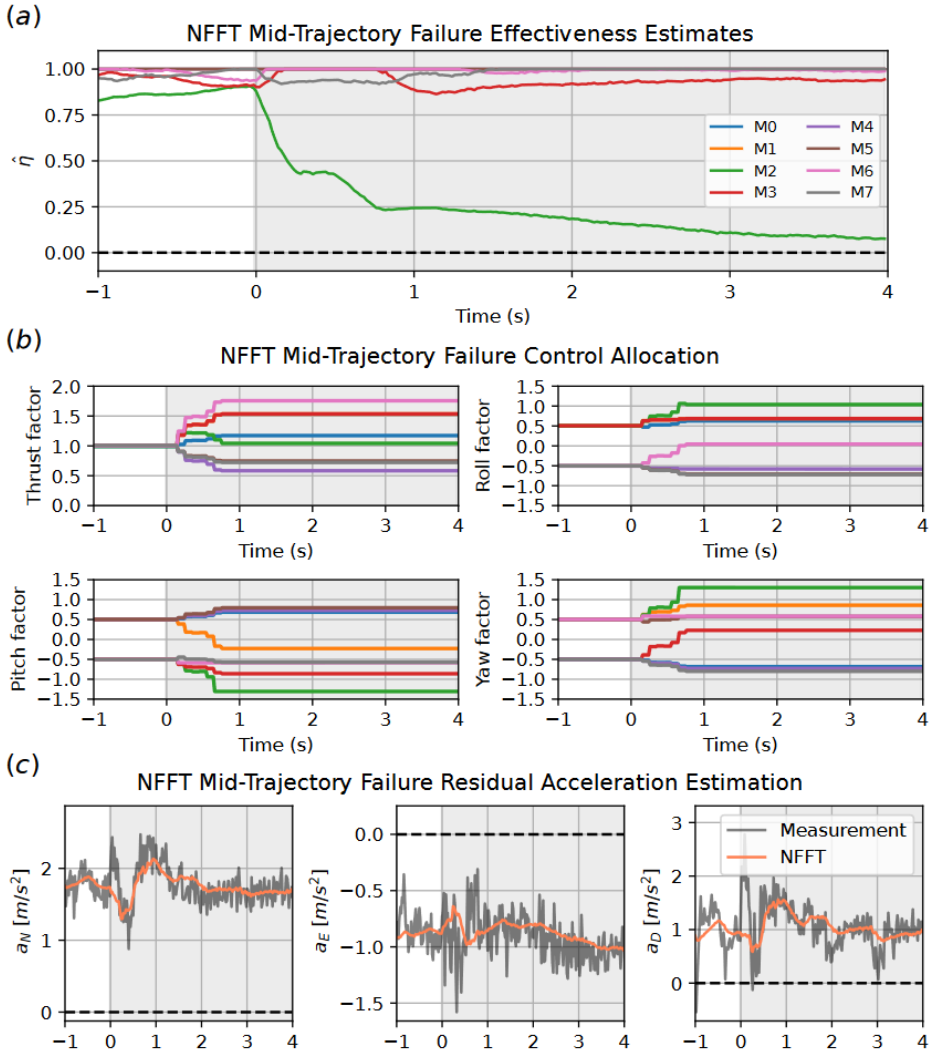
NFFT Performance - Hover



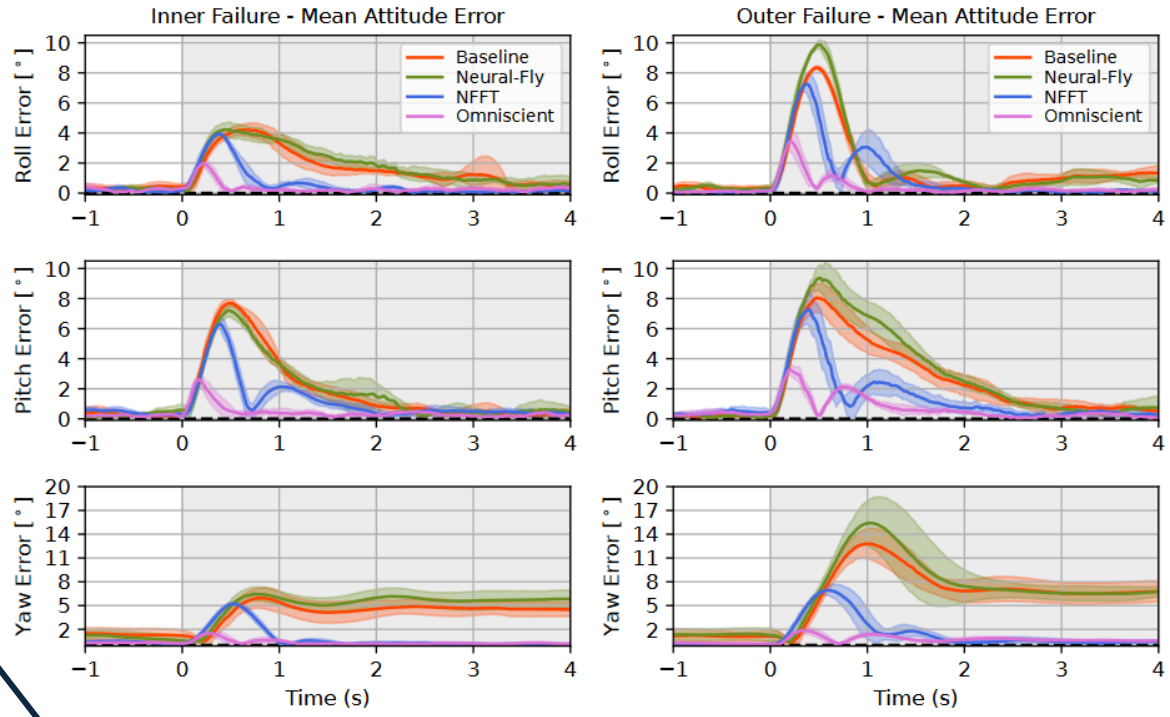


Neural-Fly residual + NFFT efficiency adaptation during hover/sequential actuator failures

NFFT Performance – Forward Flight



Attitude tracking performance is improved, both in magnitude of the initial deviation, and recovery time



Steady state allocation factor in 0.66 seconds

NFFT Performance – Forward Flight