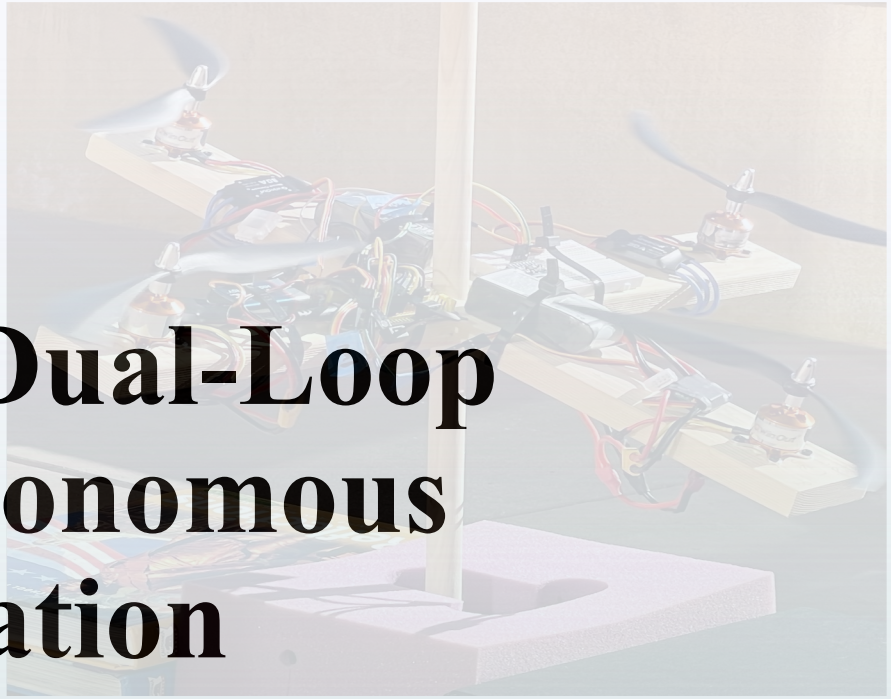


A Novel Fault Tolerant Dual-Loop Control System for Autonomous Quadcopter Navigation

Anish Anand

Palos Verdes Peninsula High School

Rancho Palos Verdes, CA, USA

No. of Cells:	2 - 3 Li-Poly 6 - 10 NiCd/NiMH
Kv:	1000 RPM/V
Max Efficiency:	80%
Max Efficiency Current:	4 - 10A (>75%)
No Load Current:	0.5A @10V
Resistance:	0.090 ohms
Max Current:	13A for 60S
Max Watts:	150W
Weight:	52.7 g / 1.86 oz
Size:	28 mm dia x 28 mm bell length

Introduction

WHY QUADCOPTERS:

- Lightweight and Easy Maneuverability
- Versatile and Quick
- Broad Potential Usage: Transportation between hospitals (Organs) or packages, Search and Rescue, Wireless 5G mobile base station

STABILITY CHALLENGES:

- Underactuated System: 4 Inputs to Control 6 DOF
- Complex Rotational & Translational Dynamics → Difficult to Stabilize
- Real world Conditions such as Weather (Heat) or Humidity → Faulty Actuator/motor
  - No Redundancy
- Result: High Risk for Crashes

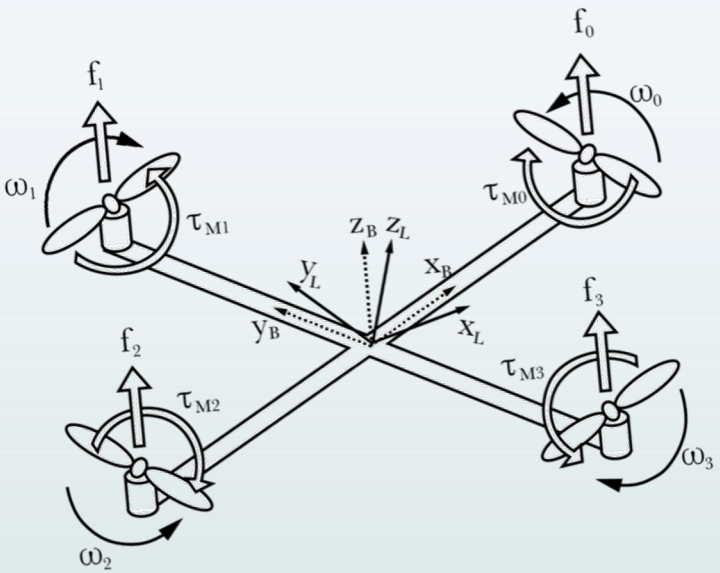
SOLUTION: Control System to Correct for Fault in Actuator

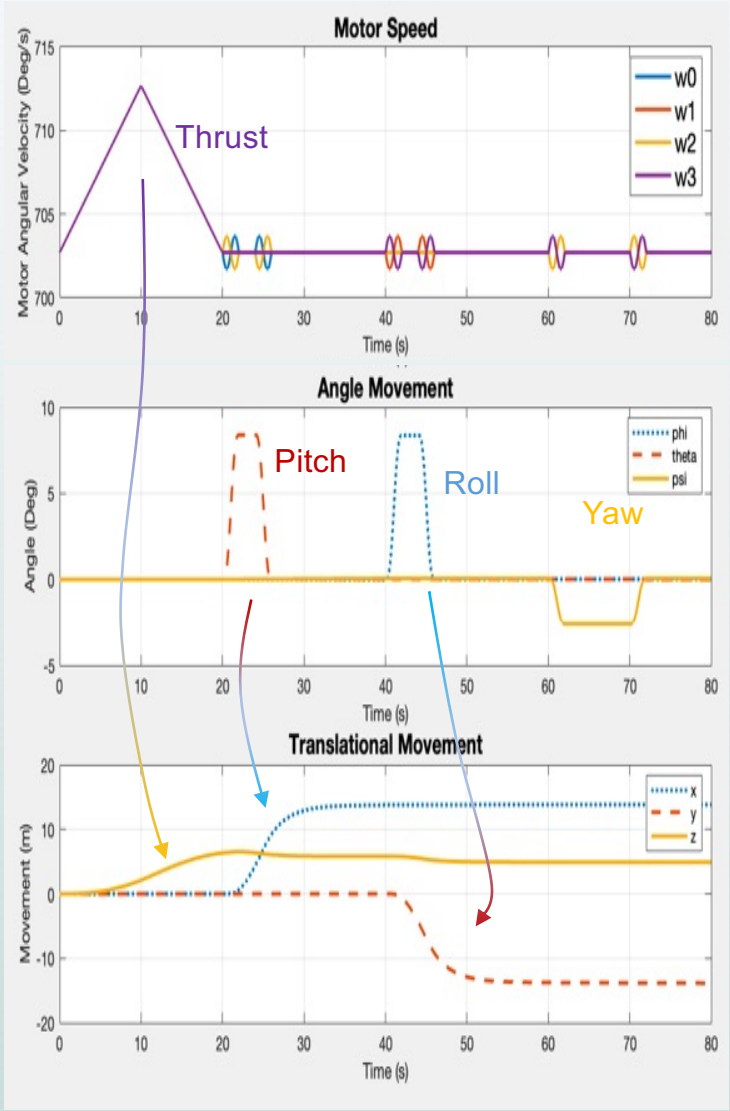
- Novel Dual-Loop Fault Tolerant Control System
  - Determines level of fault and if fault is recoverable or non-recoverable
- Recoverable Fault
  - FTC Systems adjusts Motor speeds to overcome fault
- Non-recoverable Fault
  - FTC Systems adjusts Motor speeds and Motor speed to minimize Quadcopter Yaw and land safely

Quadcopter Dynamics

- Thrust, $\boldsymbol{T_B}$, and Torque, $\boldsymbol{\tau_B}$, determined by the Angular velocities of the motors (quad arm length, l, lift constant, k, and drag constant, b)

- Acceleration, $\boldsymbol{a}$, and Angular Acceleration, $\boldsymbol{\alpha}$, determined by the Thrust and Torque (gravitiy, g, mass, m, and Rotational Inertia, I)





$$T_B = \begin{bmatrix} 0 \\ 0 \\ T = \sum_{i=1}^{4} f_i = k\sum_{i=1}^{4} \omega_i^2 \end{bmatrix}$$

$$\tau_B = \begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} lk(-\omega_2^2 + \omega_4^2) \\ lk(-\omega_1^2 + \omega_3^2) \\ \sum_{i=1}^{4} \tau_{Mi} = b\sum_{i=1}^{4} (-1)^i \omega_i^2 \end{bmatrix}$$

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = -g\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{T}{m}\begin{bmatrix} cos\psi sin\theta cos\phi + sin\psi sin\phi \\ sin\psi sin\theta cos\phi - cos\psi sin\phi \\ cos\theta cos\phi \end{bmatrix}$$

$$\begin{bmatrix} \alpha_{B\phi} \\ \alpha_{B\theta} \\ \alpha_{B\psi} \end{bmatrix} = \begin{bmatrix} \tau_\phi / I_{xx} \\ \tau_\theta / I_{yy} \\ \tau_\psi / I_{zz} \end{bmatrix}$$

Current Quadcopter Feedback Control Systems

How to determine the New motor speeds based on the physical state errors?

- The errors are manipulated to include derivative, proportional, and/or integral terms and weighted to generate control inputs

$$e(t) = x_d(t) - x(t)$$

$$u(t) = K_P e(t) + K_I \int e(t)dt + K_D \frac{de(t)}{dt}$$

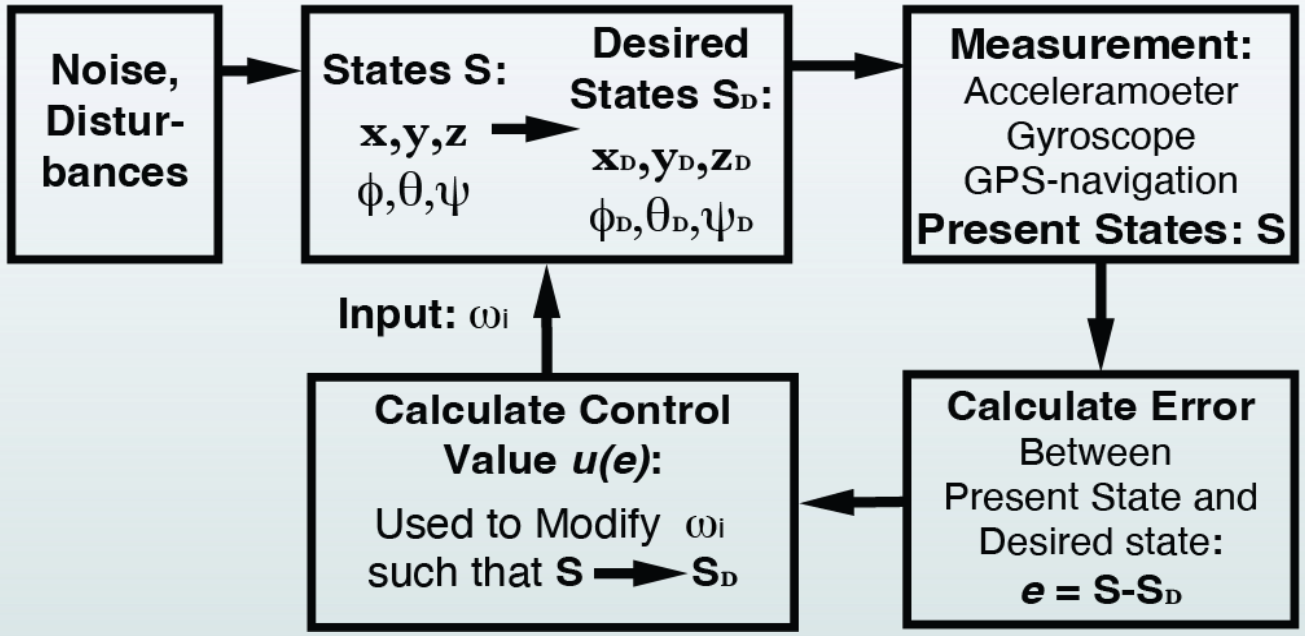
- The control input then need to be modified tor Thrust and Torque control units so that updates for the motor speed can be made based on Quadcopter Dynamics

$$U_T = (g + u_z)\frac{m}{cos\theta cos\phi}$$

$$U_{\tau\phi} = I_{xx}u_\phi \qquad U_{\tau\theta} = I_{yy}u_\theta \qquad U_{\tau\psi} = I_{zz}u_\psi$$

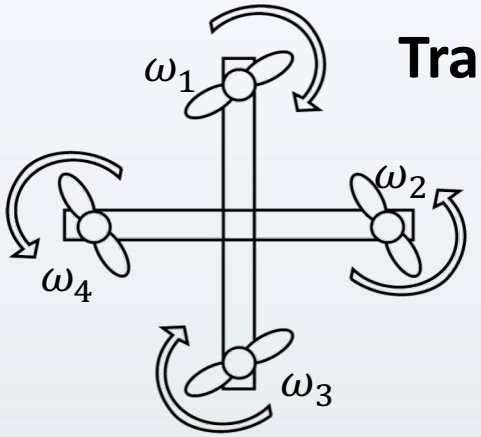
$$\begin{bmatrix} U_T \\ U_{\tau\phi} \\ U_{\tau\theta} \\ U_{\tau\psi} \end{bmatrix} = \begin{bmatrix} k & k & k & k \\ 0 & -kl & 0 & kl \\ -kl & 0 & kl & 0 \\ b & -b & b & -b \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}$$

$$\begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = \begin{bmatrix} \frac{1}{4k} & 0 & -\frac{1}{2kl} & \frac{1}{4b} \\ \frac{1}{4k} & -\frac{1}{2kl} & 0 & -\frac{1}{4b} \\ \frac{1}{4k} & 0 & \frac{1}{2kl} & \frac{1}{4b} \\ \frac{1}{4k} & \frac{1}{2kl} & 0 & -\frac{1}{4b} \end{bmatrix} \begin{bmatrix} U_T \\ U_{\tau\phi} \\ U_{\tau\theta} \\ U_{\tau\psi} \end{bmatrix}$$



Credit: Author

Development of Novel Non-Recoverable Control Policy



Traditional Spin Design

$$T_B = k(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2)$$

$$\tau_\phi = lk(-\omega_2^2 + \omega_4^2)$$

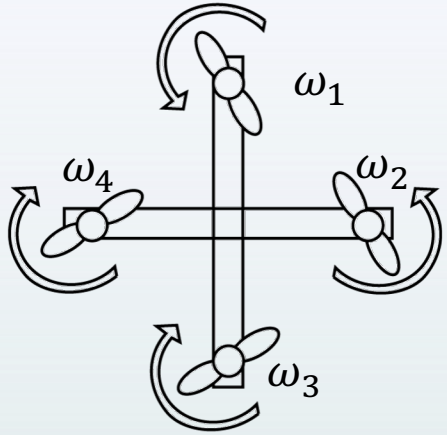
$$\tau_\theta = lk(-\omega_1^2 + \omega_3^2)$$

$$\tau_\psi = b(+\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2)$$

Pros: Independent Roll, Pitch, and Yaw Controls
Intuitive Design
Less prone to calculation errors for motor speeds

Cons: One Point of Equilibrium for stable Roll, Pitch, and Yaw
All motor torques must be equal
<u>If not equal→ uncontrollable yaw acceleration</u>

$$\begin{bmatrix}\omega_1^2\\ \omega_2^2\\ \omega_3^2\\ \omega_4^2\end{bmatrix} = \begin{bmatrix}\frac{1}{4k} & 0 & -\frac{1}{2kl} & \frac{1}{4b}\\ \frac{1}{4k} & -\frac{1}{2kl} & 0 & -\frac{1}{4b}\\ \frac{1}{4k} & 0 & \frac{1}{2kl} & \frac{1}{4b}\\ \frac{1}{4k} & \frac{1}{2kl} & 0 & -\frac{1}{4b}\end{bmatrix}\begin{bmatrix}U_T\\ U_{\tau\phi}\\ U_{\tau\theta}\\ U_{\tau\psi}\end{bmatrix}$$



Non-Recoverable Fault Tolerant Spin Design

$$T_B = k(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2)$$

$$\tau_\phi = lk(-\omega_2^2 + \omega_4^2)$$

$$\tau_\theta = lk(-\omega_1^2 + \omega_3^2)$$

$$\boxed{\tau_\psi = b(-\omega_1^2 - \omega_2^2 + \omega_3^2 + \omega_4^2)}$$

Pros: **<u>Several Points of Equilibrium for Roll, Pitch and Yaw</u>**
<u>Only ω_1 and ω_3 have to be equal and independently</u>
<u>ω_2 and ω_4 have to be equal →Neutral Yaw</u>

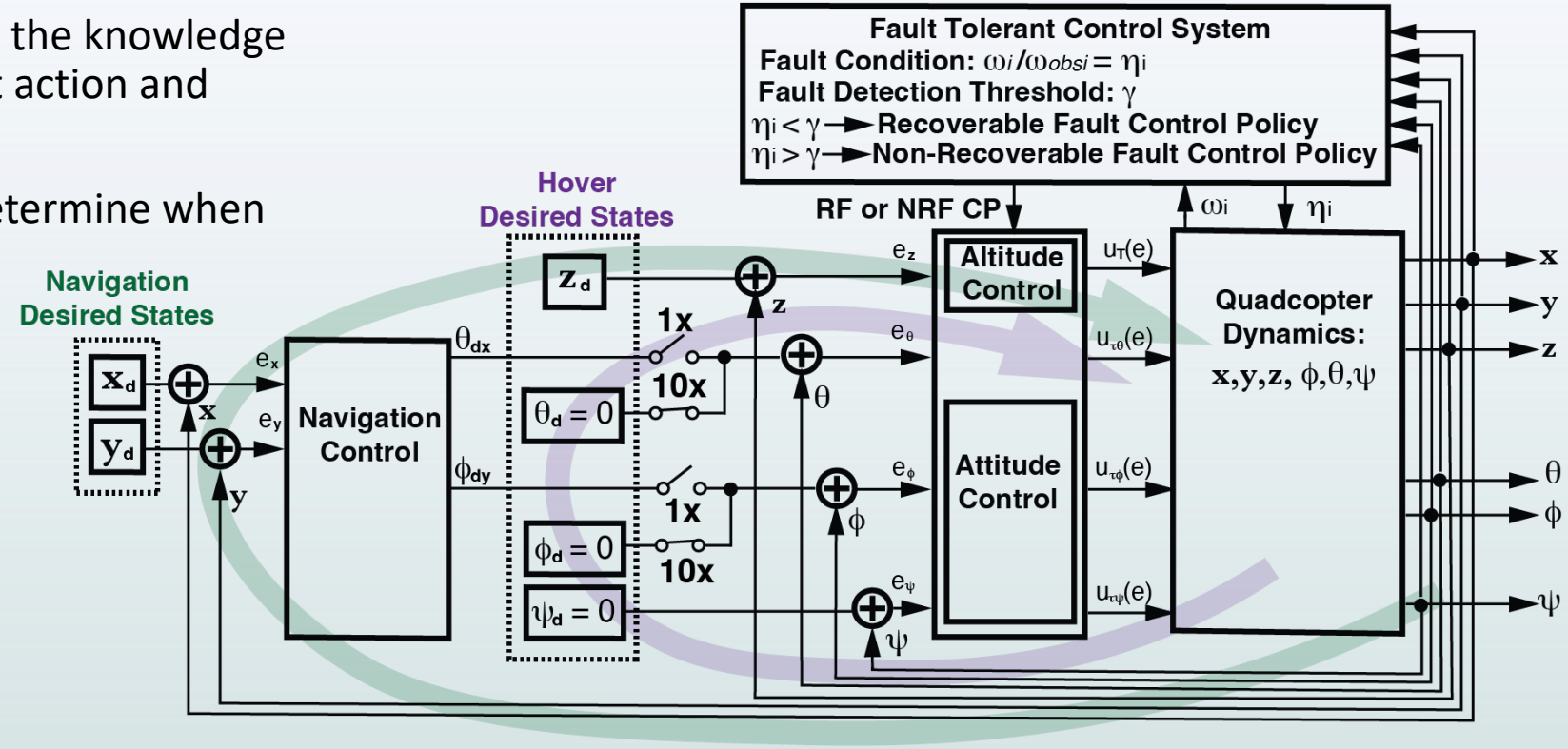
Cons: None of the Traditional Design

$$\begin{bmatrix}\omega_1^2\\ \omega_2^2\\ \omega_3^2\\ \omega_4^2\end{bmatrix} = \begin{bmatrix}\frac{1}{4k} & \frac{b^2}{2kl(K_F)} & -\frac{K_F - b^2}{2kl(K_F)} & -\frac{1}{b(K_F)}\\ \frac{1}{4k} & -\frac{K_F - b^2}{2kl(K_F)} & \frac{b^2}{2kl(K_F)} & -\frac{1}{b(K_F)}\\ \frac{1}{4k} & -\frac{b^2}{2kl(K_F)} & \frac{K_F - b^2}{2kl(K_F)} & \frac{1}{b(K_F)}\\ \frac{1}{4k} & \frac{K_F - b^2}{2kl(K_F)} & -\frac{b^2}{2kl(K_F)} & \frac{1}{b(K_F)}\end{bmatrix}\begin{bmatrix}U_T\\ U_{\tau\phi}\\ U_{\tau\theta}\\ U_{\tau\psi}\end{bmatrix}$$

$$where\ K_F = 4b^2 + 2(kl)^2$$

Implementing FTC into Dual-Loop Control System

- Fault Tolerant Control Policy requires the knowledge of the Fault Level to generate correct action and correct stable hover settings
- A Dual Loop system is designed to determine when a Fault has occurred
- The Dual-Loop system has two independent loops for Navigation and Hover.
  - The Navigation Loop, which is used to move the quadcopter to the desired state
  - The Hover Loop, that brings the Quadcopter closer to a stable point of equilibrium

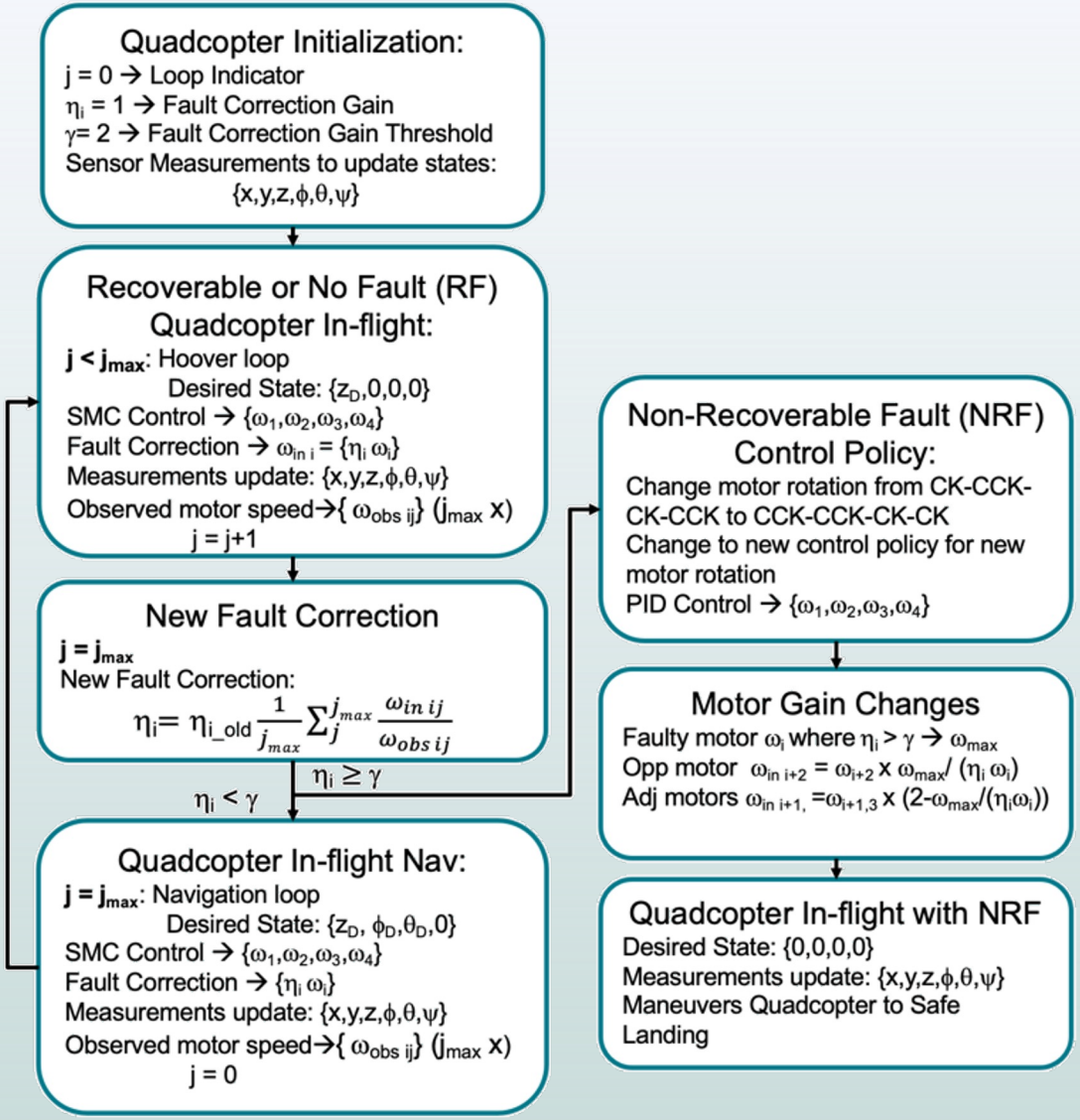


- **<u>FTC during the Hover Loop</u>**: the observed motor speeds, ω_{obs}, are determined based on averaging the states from the sensors.
- **The inputted motor speed based on the control policy, ω_i, is compared to $\omega_{obs\,i}$ to determine the Fault Level**

$$\begin{bmatrix}\omega_{obs1}^2\\ \omega_{obs2}^2\\ \omega_{obs3}^2\\ \omega_{obs4}^2\end{bmatrix} = \begin{bmatrix}\frac{1}{4k} & 0 & -\frac{1}{2kl} & \frac{1}{4b}\\ \frac{1}{4k} & -\frac{1}{2kl} & 0 & -\frac{1}{4b}\\ \frac{1}{4k} & 0 & \frac{1}{2kl} & \frac{1}{4b}\\ \frac{1}{4k} & \frac{1}{2kl} & 0 & -\frac{1}{4b}\end{bmatrix}\begin{bmatrix}(g+a_z)\frac{m}{cos\theta cos\phi}\\ \frac{(\omega_{B\phi n}-\omega_{B\phi(n-1)})}{t_d}I_{xx}\\ \frac{(\omega_{B\theta n}-\omega_{B\theta(n-1)})}{t_d}I_{yy}\\ \frac{(\omega_{B\psi n}-\omega_{B\psi(n-1)})}{t_d}I_{zz}\end{bmatrix}$$

Complete Fault Tolerant Control System

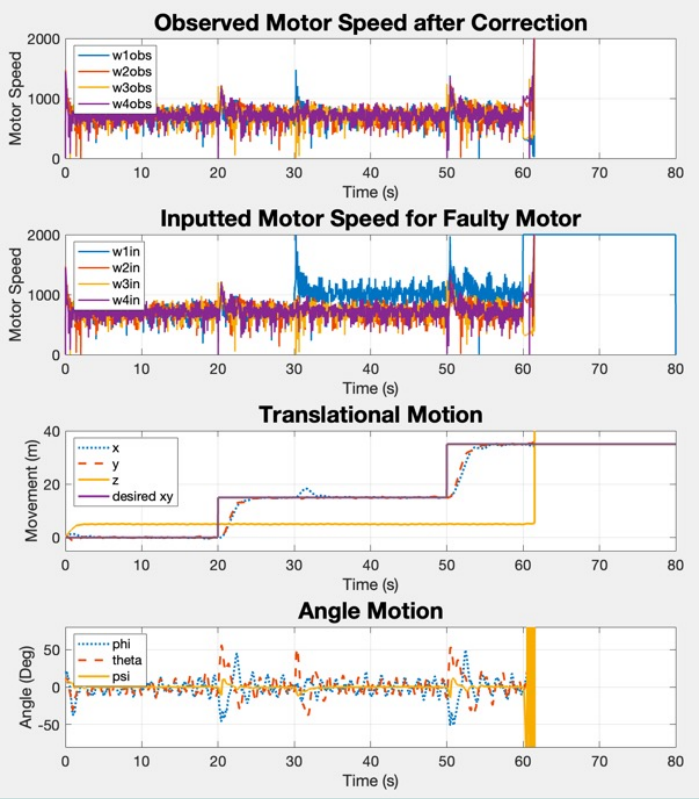
- Fault Tolerant Control Uses the Fault Level to determine the Fault Tolerant actions and adjustments are required to apply to the Navigation State $$Fault\ Level = \frac{\omega_{\mathrm{i}}}{\omega_{\mathrm{obs\ i}}}$$
- **Recoverable Fault:** Fault Level $\frac{\omega_{\mathrm{i}}}{\omega_{\mathrm{obs\ i}}} < \gamma$ such that additional gain to the actuator **can** make up for the fault. Gain based on Fault level is adjusted to the inputted motor speed for the faulty motor.
- **Non-Recoverable Fault:** Fault Level $\frac{\omega_{\mathrm{i}}}{\omega_{\mathrm{obs\ i}}} \geq \gamma$ such that additional gain to the actuator **cannot** make up for the fault.
  - The Quadcopter spin directions and control policies are changed to the FTC based settings
  - The faulty motor is set to its max level which a lower motor speed than desired. The opposite motor is set to match the motor speed of the faulty motor speed.
  - The other two motor speeds must increase to maintain the same adjusted thrust
- Once Fault is Non-Recoverable, main objective for Quadcopter is safe landing



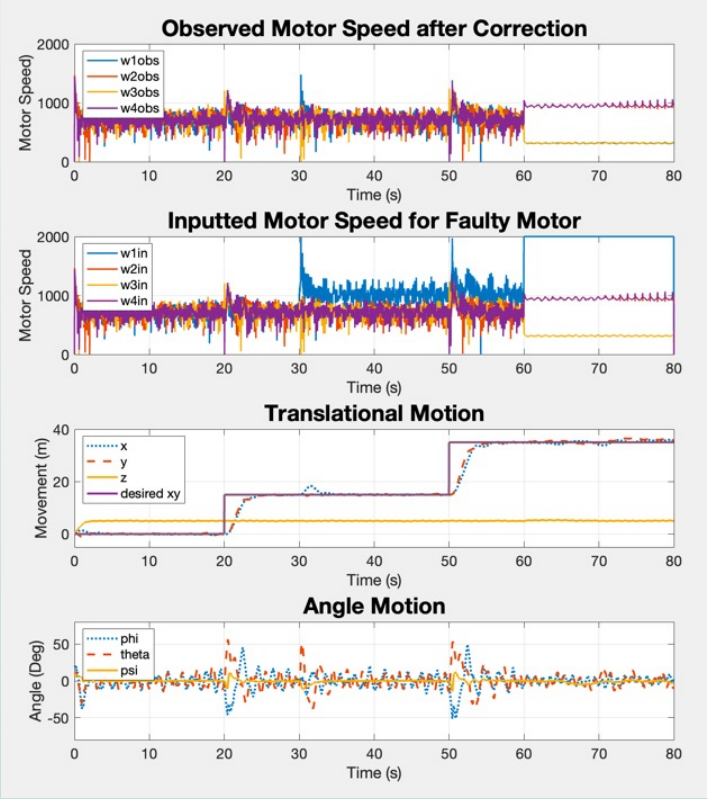
Results from Simulations

- (1) is a transient simulation with a **Recoverable Fault at 30 sec and a Non-recoverable Fault at 60 sec** using only the Traditional Design with no change in Spin Direction for Non-recoverable Fault. The Recoverable Fault is corrected for by the algorithm, but the non-recoverable fault cause the Quadcopter yaw to accelerate and spin out of control.
- (2) is a transient simulation with a **Recoverable Fault at 30 sec and a Non-recoverable Fault (not complete failure) at 60 sec** using the FTC Design. Both the Recoverable Fault and the Non-recoverable Fault are corrected for by the algorithm.

(1)—Traditional + Recoverable Fault Policy

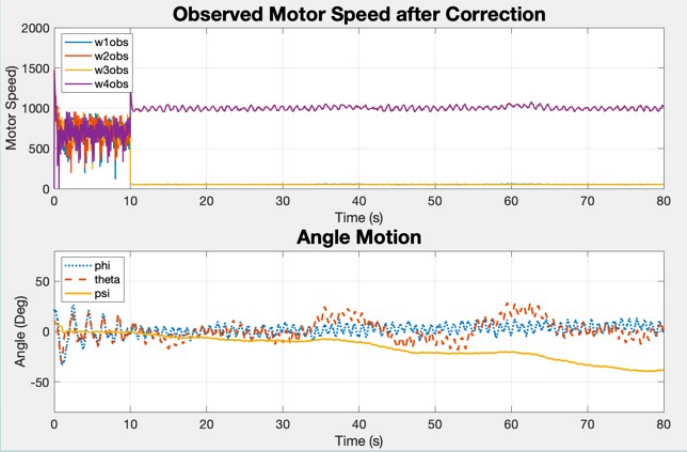


(2)—Fault Tolerant Control System



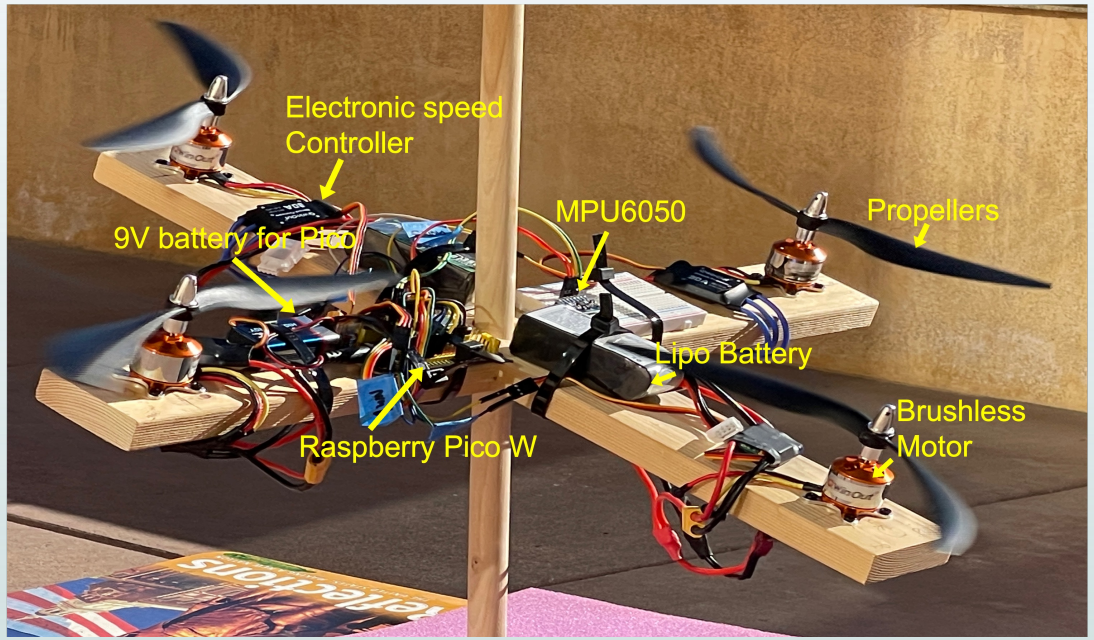
- (3) is a transient simulation of a Non-recoverable Fault with one **actuator being completely broken** and the Quadcopter is still able to hover and move.

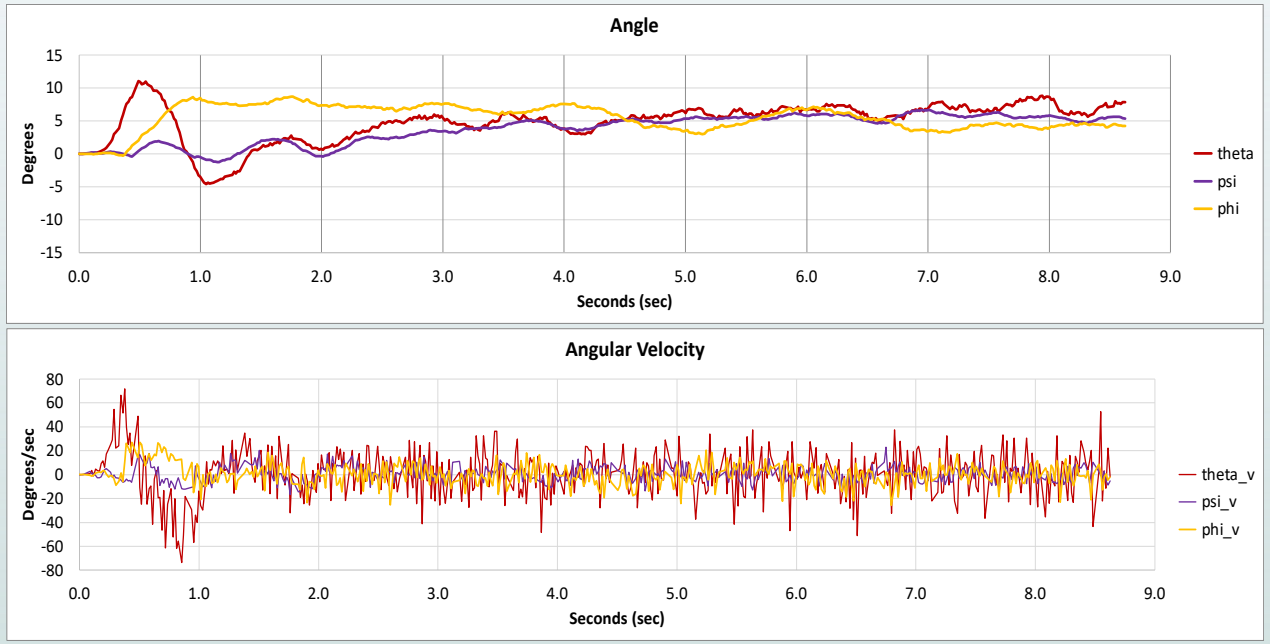
(3)—Complete Motor Failure



Results from Prototypes and Conclusion

- A quadcopter prototype was designed to **validate the new fault tolerant control policy**. The dowel in the center allows the model to move freely in the ϕ, θ, ψ, and z direction
- Prototype data shows the quadcopter can be stable for a limited amount of time with the new fault tolerant control system





- A novel Fault Tolerant Dual Loop Control System is designed and validated with MATLAB simulations and prototype data
- **The solution presented here showcases a control system that maintains no yaw in the event of motor actuator fault, giving stability for a safe landing**