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

Anish Anand
Palos Verdes Peninsula High School
Rancho Palos Verdes, CA, USA
anishmail2007@gmail.com

***Abstract*— As industrial and commercial UAV adoption grows, robust fault-tolerant control systems are required. This paper investigates a novel fault tolerant dual loop control system, which possesses the flexibility to function even when a complete fault occurs in the actuator, while maintaining the robustness of a classical control system. This novel fault tolerant control system changes the motors' spin directions and speed when a non-recoverable actuator fault is detected, thereby creating stability in the pitch, roll, and yaw despite adjacent motors having different motor speeds. The fault tolerant dual-loop control system is optimized in MATLAB. Its robustness is tested on a quadcopter prototype with MicroPython coding on a Raspberry Pico.**

Keywords—quadcopters, control systems, fault tolerant control system, sliding-method controller, PID controller

I. Introduction

Autonomous unmanned vehicles (UAVs) are growing in popularity for both commercial and military applications. Quadcopters have gained interest because of their high versatility and maneuverability, as well as their structural simplicity. These attributes lead to multiple applications in delivery, aerial surveillance, search and rescue, as well as communication base stations [1,2]. However, the complexity of quadcopter dynamics makes navigation and stability challenging. Further, real world conditions like wind or humidity can cause fault in the actuators or sensors, creating a high potential for the quadcopter to crash. Therefore, there is growing interest in developing a fault-tolerant UAV control system, which would allow for greater adoption of quadcopters.

Stabilizing an autonomous quadcopter while hovering or in flight is complex, particularly in the presence of a failing actuator, as there is no redundancy in the quadcopter's actuators. Current literature describes potential solutions for fault tolerant control systems but assume a fixed spin rate or a constant thrust after developing a faulty actuator, which are ideal conditions that limit applicability [3]. Other literature use Reinforcement Learning (RL) techniques to provide fault tolerance, but considerable assumptions are used in the RL environment, reducing the control solution's applicability [4,5]. This paper investigates a novel fault tolerant dual loop control system that is generalizable in overcoming a partial to complete failure of an actuator.

As an overview, the novel fault tolerant dual loop control system consists of two independent loops. The first control loop provides the quadcopter navigation control, while the second is a hover loop used to maintain stability by bringing the quadcopter back closer to a linear, stable, hover position in between navigation steps. The linear hover loop is also used for the fault detection. The sliding method control algorithm provides the key motor inputs for navigation.

Integrated in the hover loop is a fault tolerant control system (FTC). If a fault is perceived, the fault detector determines the magnitude of the actuator failure to evaluate if the fault is recoverable or non-recoverable. In the event of a recoverable fault, the FTC system adjusts the motor speeds to overcome the actuator failure. However, if the fault condition is non-recoverable, the FTC system runs a separate arm of the control loop which changes motor spins and motor speeds to counteract yaw acceleration. Traditional control systems do not change the motor spins with a motor failure, resulting in rapidly escalating yaw velocity, leading to a quadcopter that spins out of control. However, by adjusting the motor speeds and changing the motor spins in the event of a non-recoverable failure, this FTC creates a stable yaw, allowing the quadcopter to hover and quickly maneuver to a rest position, preventing significant damage to the drone.

II. Quadcopter Dynamics

Quadcopter dynamics are defined with two frames of reference: inertial and body frame. The inertial or earth frame provides the linear positioning of x, y, z, and the three Euler angles, ϕ, θ, ψ, which respectively represent the pitch, roll, and yaw angle. The body frame is used to determine the quadcopter's torque and thrust dynamics created by motor speeds. The relationship of the forces and torques in two frames of reference are presented in Fig. 1.

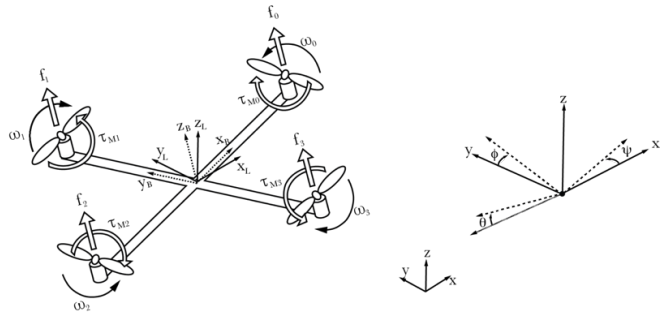

Fig.1. The inertial and the body frame of the conventional quadcopter

The quadcopter operates by rotating the four motors shown in Fig. 1. The angular velocity, ω_i, of each motor creates an upward force f_i in the z-axis of the body frame of the

979-8-3503-0965-2/23/$31.00 ©2023 IEEE

quadcopter and a torque τ_{Mi} around each motor. The combined forces create the thrust T_B and the torque for the quadcopter τ_B as seen in Eq. (1) and (2) where k is the lift constant, b is the drag constant, and l is the length of the of the quadcopter. The torque in the yaw direction is based on the conventional Clockwise-Counterclockwise-Clockwise-Counterclockwise (CK–CCK–CK–CCK) motor rotation.

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

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

Since all the forces and torques are easily provided for the quadcopter's body frame, Newton-Euler rotational matrices are used to go back and forth from the body frame and the inertial frame. Basic geometry is used to determine this relationship in Eq. (3). The translational motion is based on the gravitational force and thrust of the quadcopter in the inertial frame. An additional term is then added: Ax, Ay, Az (based on the drag coefficient of the air in the *x, y, z* direction) is multiplied by the velocity of the quadcopter.

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} = -g \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \frac{T_B}{m} \begin{bmatrix} cos\psi sin\theta cos\phi + sin\psi sin\phi \\ sin\psi sin\theta cos\phi - cos\psi sin\phi \\ cos\theta cos\phi \end{bmatrix} + \frac{1}{m} \begin{bmatrix} Ax & 0 & 0 \\ 0 & Ay & 0 \\ 0 & 0 & Az \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \tag{3}$$

The rotational dynamics determined in the body frame are then rotated to the inertial frame. In the body frame, the external torque is equal to the angular inertia, the centripetal forces, and the gyroscopic forces as depicted in Eq. (4) and (5). In these equations, I_{xx}, I_{yy}, and I_{zz} are the moments of inertia in the *x*, *y*, and *z* directions for the quadcopter, I_r is the moment of inertia for the rotor, ω_B is the angular velocity in the body frame for the ϕ, θ, ψ angles, and $\omega_\Gamma = \omega_1 - \omega_2 + \omega_3 - \omega_4$ which is the gyroscopic angular velocity.

$$\begin{bmatrix} \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} I_{xx}\alpha_\phi \\ I_{yy}\alpha_\theta \\ I_{zz}\alpha_\psi \end{bmatrix} + \begin{bmatrix} (I_{zz} - I_{yy})\omega_{B\theta}\omega_{B\psi} \\ (I_{xx} - I_{zz})\omega_{B\phi}\omega_{B\psi} \\ (I_{yy} - I_{xx})\omega_{B\theta}\omega_{B\phi} \end{bmatrix} + I_r \begin{bmatrix} \omega_{B\theta} \\ -\omega_{B\phi} \\ 0 \end{bmatrix} \omega_\Gamma \tag{4}$$

$$\begin{bmatrix} \alpha_{B\phi} \\ \alpha_{B\theta} \\ \alpha_{B\psi} \end{bmatrix} = \begin{bmatrix} \tau_\phi / I_{xx} \\ \tau_\theta / I_{yy} \\ \tau_\psi / I_{zz} \end{bmatrix} + \begin{bmatrix} (I_{yy} - I_{zz})\omega_{B\theta}\omega_{B\psi}/I_{xx} \\ (I_{zz} - I_{xx})\omega_{B\phi}\omega_{B\psi}/I_{yy} \\ (I_{xx} - I_{yy})\omega_{B\theta}\omega_{B\phi}/I_{zz} \end{bmatrix} - I_r \begin{bmatrix} \omega_{B\theta}/I_{xx} \\ -\omega_{B\phi}/I_{xx} \\ 0 \end{bmatrix} \omega_\Gamma \tag{5}$$

With these equations, the quadcopter's location, orientation, acceleration, and velocity are specified based on the speeds applied to the four motors.

III. Novel Fault Tolerant Dual-Loop Control

Feedback control systems operate to stabilize the states of the quadcopter when presented with noise or disturbance. The control system uses the errors between the desired states and the measured states to determine the correction in the quadcopter motor speeds. In Fig 2, a novel dual-loop control system is presented where the outer loop (green) is used for determining quadcopter control inputs for navigation and the inner hover loop (purple) moves the quadcopter states closer to a hover state. Fault detection takes place based in this hover loop. The error between the desired state and the measured output from the gyroscope and accelerometer sensors determine the control input from the control algorithm methodology.

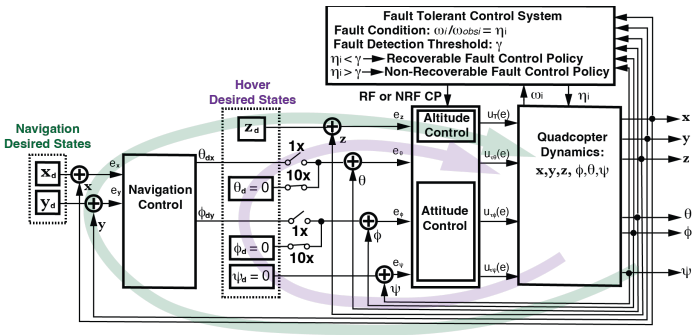

Fig 2. Dual-Loop Quadcopter Feedback control system

The fault detector determines the level of fault for each motor. The fault detector determines if the fault is a partial fault that can be recovered with the current motor capability or if the fault is so high such that the motor cannot correct for the faulty condition (non-recoverable fault). Fault condition threshold γ is used to determine which control policy is used for the fault control system. In the event of a recoverable actuator fault, which is lower than the fault threshold γ, a fault gain η_i is multiplied to the motor gain to accommodate for the fault condition. If the fault condition is greater than γ, a nonrecoverable fault, the maximum possible motor gain is applied to the faulty motor and a similar gain to the opposite motor, while the two adjacent motors increase their gain to make up for the remaining deficit, producing the correct total thrust. This new control algorithm, along with a change in the motor rotations, described later, is applied to move the quadcopter to a stable hover state and land without damage.

IV. Dual-Loop Control System Optimization

Three control algorithms are investigated to determine which algorithm is optimal: proportional-integral-derivative (PID), back-stepping control (BSC), or sliding-method control (SMC). The control algorithm determines the error used for correction and the input need to fix the error.

Once the control input is calculated based on the error terms and the control algorithm, u_z, u_θ, u_ϕ, u_ψ, are used to create control units which in turn determine the required motor speed.

The control unit for thrust and the desired value for theta and phi are defined as follows in Eq. (6),

$$\begin{aligned} U_T &= (g + u_z)\frac{m}{cos\theta cos\phi} \\ \theta_D &= \frac{m}{U_T}u_x \\ \phi_D &= \frac{m}{U_T}u_y \end{aligned} \tag{6}$$

and the rest of the control units for the torques are shown below in Eq. (7).

$$\begin{aligned} U_{\tau\phi} &= I_{xx}u_\phi \\ U_{\tau\theta} &= I_{yy}u_\theta \\ U_{\tau\psi} &= I_{zz}u_\psi \end{aligned} \tag{7}$$

The control unit is then related to the motor speeds as follow in Eq. (8a) and its matrix form in Eq. (8b).

$$\begin{aligned} U_T &= k(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) \\ U_{\tau\phi} &= kl\,(-\omega_1^2 + \omega_3^2) \\ U_{\tau\theta} &= kl\,(-\omega_2^2 + \omega_4^2) \\ U_{\tau\psi} &= b\,(\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2) \end{aligned} \tag{8a}$$

$$\begin{bmatrix} U_T \\ U_{\tau\phi} \\ U_{\tau\theta} \\ U_{\tau\psi} \end{bmatrix} = \begin{bmatrix} k & k & k & k \\ 0 & -kl & 0 & kl \\ -kl & 0 & kl & 0 \\ b & -b & b & -b \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} \tag{8b}$$

Multiplying both sides by the inverse matrix, the motor speeds are calculated by the conventional control policy Eq (9).

$$\begin{aligned} \omega_1^2 &= \frac{U_T}{4k} - \frac{U_{\tau\theta}}{2k} + \frac{U_{\tau\psi}}{4b} \\ \omega_2^2 &= \frac{U_T}{4k} - \frac{U_{\tau\phi}}{2k} - \frac{U_{\tau\psi}}{4b} \\ \omega_3^2 &= \frac{U_T}{4k} + \frac{U_{\tau\theta}}{2k} + \frac{U_{\tau\psi}}{4b} \\ \omega_4^2 &= \frac{U_T}{4k} + \frac{U_{\tau\phi}}{2k} - \frac{U_{\tau\psi}}{4b} \end{aligned} \tag{9}$$

A. PID Control Algorithm

The error for each state and control input for the PID control algorithm is as follows in Eq. (10), where the error term is the desired state minus the measured state from a sensor and the control input includes a proportional, integral, and derivative term with corresponding K_i gain parameters.

$$\begin{aligned} e(t) &= x_d(t) - x(t) \\ u(t) &= K_P e(t) + K_I \int e(t)dt + K_D \frac{de(t)}{dt} \end{aligned} \tag{10}$$

The proportional part of the controller corrects for large errors, but can be limited by transient and steady state error. The transient error is the undershoot, overshoot, and the oscillatory behavior of the states before reaching the desired states. The derivative part of the controller helps reduce this transient error, but if given too much weight it can overcompensate for noise and disturbances. The instability is due to the amplification of the variation in the signal when there is noise or disturbances. Steady state errors arise from constant imbalance during the control process, which does not change. The main drawback of a PID controller is the slow settling time, but later simulations show the slow settling is beneficial in non-recoverable fault conditions.

B. BSC Algorithm

A Back-Stepping Controller assumes a local stable point by defining the error term to be stable. The error term includes an error for the state and its derivative as seen in Eq (11). Back-stepping overcorrects for the error and then removes the overcorrection in the next step. The Back-Stepping Controller has fast settling and better tolerance to noise but requires very high dynamic ranges for the motor speeds, which is an undesirable trait for partial fault conditions.

$$\begin{aligned} e(t) &= x_d(t) - x(t) \\ e_s(t) &= (\dot{x}_d(t) - \dot{x}(t)) + K_1\big(x_d(t) - x(t)\big) \\ u(t) &= K_1\big(e_s(t) + K_I e(t) - K_2 e_s(t-1)\big) \end{aligned} \tag{11}$$

C. SMC Algorithm

Similar to the BSC, the SMC controller also assumes a local stable point by changing the error term to always be stable. The sliding method uses a signum (or sign function) to always make the correction in the opposite direction of the error. The settling time is fast like the BSC, but the dynamic range of the motor speed is much less, making this an ideal solution for partial fault conditions. The error terms and control input are shown in Eq. (12).

$$\begin{aligned} e(t) &= x_d(t) - x(t) \\ e_s(t) &= (\dot{x}_d(t) - \dot{x}(t)) + K_1\big(x_d(t) - x(t)\big) \\ u(t) &= K^2 e(t) + K_1\, sgn\big(e_s(t)\big) + K_2 e_s(t) \end{aligned} \tag{12}$$

V. Fault Dectection and Novel FTC System

The FTC system determines how to correct quadcopter instability when faced with actuator faults. The fault tolerant algorithm compares the expected motor speed, ω_i, from the control policy to the observed motor speed, ω_{obs}, based on the measured states in the hover loop. If η_i (expected motor speed ω_i divided by the observed motor speed ω_{obs}) is below the threshold of γ, this state is deemed a recoverable fault (Fig 3). The motor speed is then multiplied by the gain η_i so the new inputted motor speed will achieve the desired speed.

A non-recoverable fault condition occurs when η_i exceeds the threshold γ (Fig. 3). In this case, the required motor speed is greater than the maximum possible motor speed in the faulty actuator ($\eta_i\omega_i > \omega_{max}$). The observed motor speed cannot reach the desired speed in the faulty motor. To maintain quadcopter stability, the FTC creates matching observed motor speeds between the faulty motor and the opposite motor, so that the pitch and roll angles are stable. To create enough total thrust, the adjacent two motors additionally increase their observed speed to recover the remaining thrust deficit.

Additionally, in the event of a non-recoverable fault, a change in the motor rotations is required to maintain stability with the imbalance of motor speeds. With the conventional control policy seen in Eq (9), if the motor speeds in the pitch and

roll directions are imbalanced, the yaw angle exhibits an acceleration that will quickly destabilize the quadcopter, likely resulting in a crash. To counteract the instability created when the four motor speeds are uneven, the FTC system creates a new hardware state for motor rotation, so there is no acceleration in the yaw angle. The conventional rotation plan of CK-CCK-CK-CCK changes to CCK-CCK-CK-CK so adjacent motors can endure being imbalanced. The flow chart (Fig. 3) below details the fault detection and FTC algorithm.

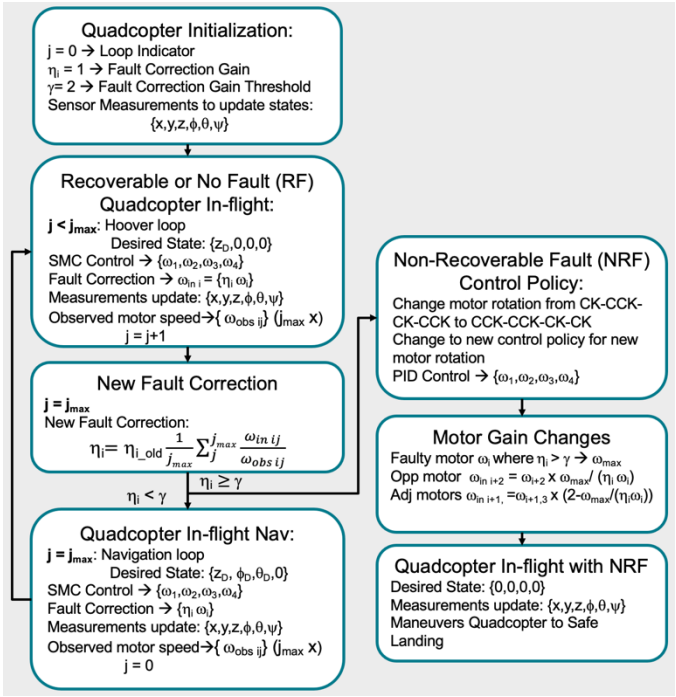

Fig 3. Dual-Loop Quadcopter Control System Flow Chart

To determine if a fault is present, the observed motor speeds are calculated based on the measured states and the control policy matrix in Eq (9), that results in Eq (13). These observed motor speeds are determined by measured averaged values from the accelerometer and gyroscope, during the hover loop of the dual-loop quadcopter control. Calculating the observed motor speed in the hover loop allows for a more accurate reading since the quadcopter states are converging to a stable value.

$$\begin{bmatrix}\omega_{obs1}^2\\ \omega_{obs2}^2\\ \omega_{obs3}^2\\ \omega_{obs4}^2\end{bmatrix} = \begin{bmatrix}\frac{1}{4k} & 0 & -\frac{1}{2kl} & \frac{1}{4b}\\ \frac{1}{4k} & -\frac{1}{2kl} & 0 & -\frac{1}{4b}\\ \frac{1}{4k} & 0 & \frac{1}{2kl} & \frac{1}{4b}\\ \frac{1}{4k} & \frac{1}{2kl} & 0 & -\frac{1}{4b}\end{bmatrix}\begin{bmatrix}(g+a_z)\frac{m}{cos\theta cos\phi}\\ \frac{(\omega_{B\phi n}-\omega_{B\phi(n-1)})}{t_d}I_{xx}\\ \frac{(\omega_{B\theta n}-\omega_{B\theta(n-1)})}{t_d}I_{yy}\\ \frac{(\omega_{B\psi n}-\omega_{B\psi(n-1)})}{t_d}I_{zz}\end{bmatrix} \quad (13)$$

For a non-recoverable fault condition, the motor spins are changed to CCK-CCK-CK-CK to accommodate different torques in the pitch and roll directions. With the new spin directions, a new torque definition for yaw is required (Eq. 14) as opposed to what is presented in Eq (2).

$$\tau_\psi = b(-\omega_1^2 - \omega_2^2 + \omega_3^2 + \omega_4^2) \quad (14)$$

Using the new torques (Eq. 14), a newly derived inverse matrix is the basis for the novel control policy. The motor inputs are calculated with the novel control policy (Eq. 15), thereby leading to a stable yaw. Despite a non-recoverable fault in the actuator and imbalances between the adjacent motors, this novel FTC system allows the quadcopter to function even when one motor is completely broken.

$$\begin{bmatrix}\omega_1^2\\ \omega_2^2\\ \omega_3^2\\ \omega_4^2\end{bmatrix} = \begin{bmatrix}\frac{1}{4k} & \frac{b^2}{2kl(K_F)} & -\frac{K_F-b^2}{2kl(K_F)} & -\frac{1}{b(K_F)}\\ \frac{1}{4k} & -\frac{K_F-b^2}{2kl(K_F)} & \frac{b^2}{2kl(K_F)} & -\frac{1}{b(K_F)}\\ \frac{1}{4k} & -\frac{b^2}{2kl(K_F)} & \frac{K_F-b^2}{2kl(K_F)} & \frac{1}{b(K_F)}\\ \frac{1}{4k} & \frac{K_F-b^2}{2kl(K_F)} & -\frac{b^2}{2kl(K_F)} & \frac{1}{b(K_F)}\end{bmatrix}\begin{bmatrix}U_T\\ U_{\tau\phi}\\ U_{\tau\theta}\\ U_{\tau\psi}\end{bmatrix} \quad (15)$$

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

Because of the higher accumulation of error (chattering) in the conventional SMC control system in the FTC, the PID control algorithm is instead used for non-recoverable faults.

VI. Simulation of Quadcopter with Accuator Faults

The novel FTC dual-loop system presented in Fig 3 is simulated using MATLAB and then coded in MicroPython for implementation on a quad-axis UAV prototype. All three control algorithms presented in Section IV were coded to evaluate their performance. For the conventional control policy, compared to BSC and PID, the SMC algorithm performed best due to fastest settling time and lowest requirements on motor speed range. The SMC also worked well for recoverable fault conditions (initiated at time 30 sec). However, during a non-recoverable fault condition (initiated at time 60 sec), the conventional control policy exhibits significant instability that can lead to a crash (Fig. 4).

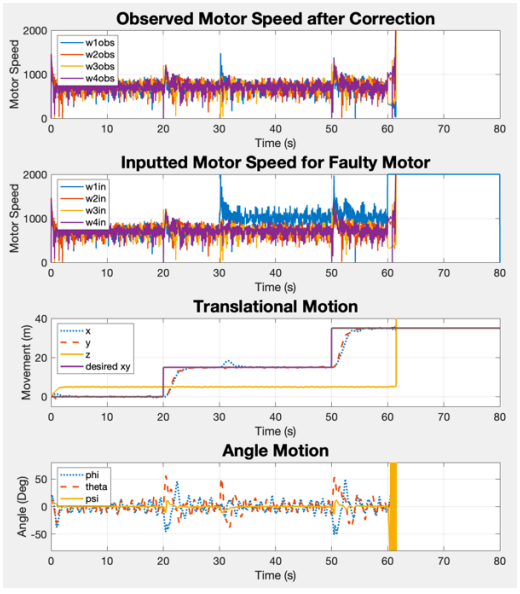

Fig 4. Simulations for Quadcopter with recoverable fault at 30 sec, Motor 1 speed is reduced to 70% at 30 sec, and non-recoverable fault at 60 sec, Motor 1 at max speed setting is effectively only 50% of required speed. Non-recoverable Fault using conventional control algorithm shows instability.

Once the new control policy is implemented, the quadcopter can maintain stability during a non-recoverable fault (Fig 5).

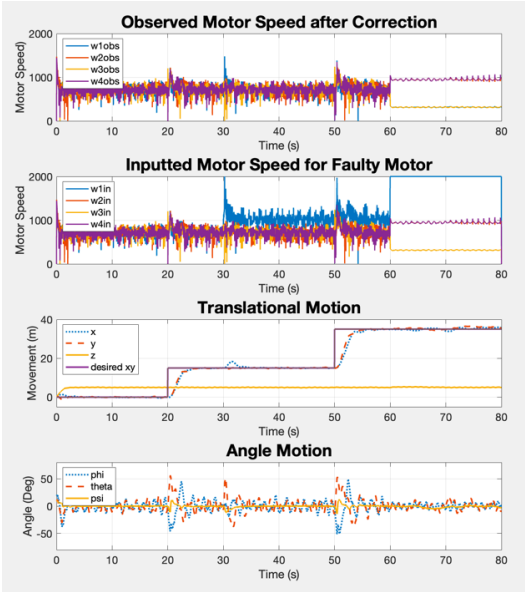

Fig 5. Same simulation condition as Fig 4 but the novel Non-recoverable fault Control Policy with PID algorithm is used.

In the event of a complete actuator fault, the new FTC system stabilizes the quadcopter for short periods of time (Fig 6), allowing for a safe landing.

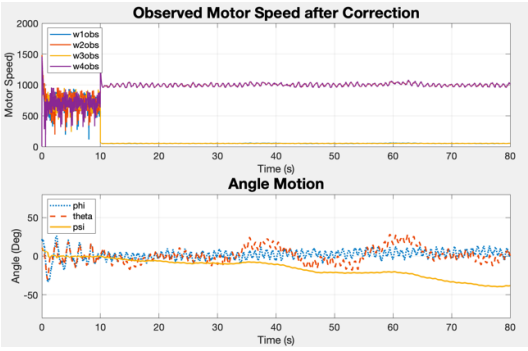

Fig 6. A non-recoverable fault in Motor 1 with 0% motor speed exhibits stability for ~60sec with new fault tolerant control policy.

VII. Quad Axis Prototype Development

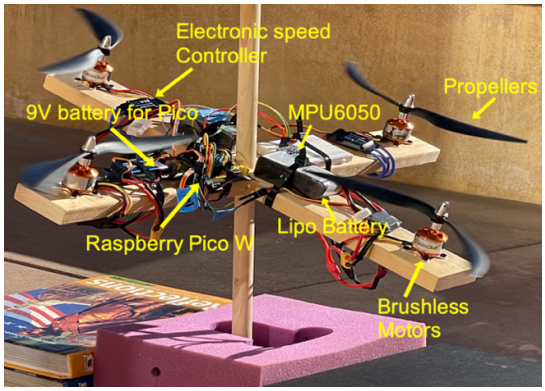

Fig 7. Dual-Loop Quadcopter Feedback control system

A quadcopter prototype (Fig 7) 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, while restricting movement in the x and y direction. A Raspberry Pico is used to implement the fault tolerant control system with the motors oriented in a CCK-CCK-CK-CK manner using MicroPython. Flight data was collected using the novel fault tolerant control policy (Fig. 8). Simulations in MATLAB used the estimated value of mass, length, inertia, and lift and drag constants of the prototype.

Prototype data shows the quadcopter can be stable for a limited amount of time with the new fault tolerant control (FTC) system.

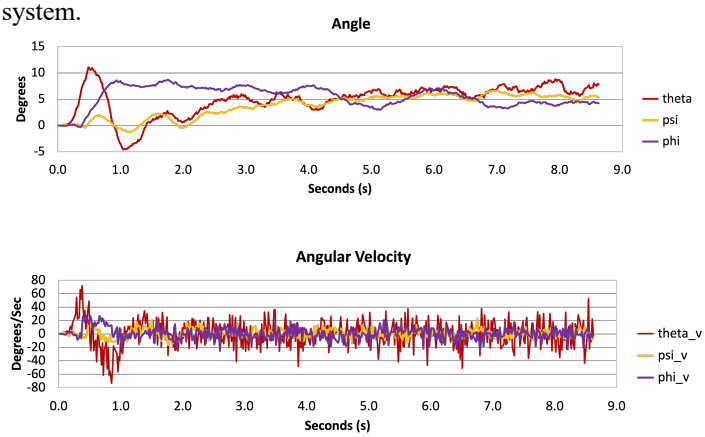

Fig 8. In-flight Data from Quadcopter prototype using fault-tolerant control policy.

VIII. Conclusion

A novel Fault Tolerant Dual Loop Control System is designed and validated with MATLAB simulations and prototype data. Current fault tolerant control algorithms do not minimize yaw acceleration and rate in many conditions, causing the quadcopter to become unstable and crash. 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. This control system works in conditions even when one motor becomes completely non-functional.

References

[1] J. Castillo-Zamora, K. A. Camarillo-GóMez, G. I. PéRez-Soto and J. RodríGuez-ReséNdiz, "Comparison of PD, PID and Sliding-Mode Position Controllers for V–Tail Quadcopter Stability," in IEEE Access, vol. 6, pp. 38086-38096, 2018.

[2] S. Bouabdallah and R. Siegwart, "Backstepping and Sliding-mode Techniques Applied to an Indoor Micro Quadrotor," Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain, 2005, pp. 2247-2252.

[3] F. Sharifi, M. Mirzaei, B. W. Gordon and Y. Zhang, "Fault tolerant control of a quadrotor UAV using sliding mode control," 2010 Conference on Control and Fault-Tolerant Systems (SysTol), Nice, France, 2010, pp. 239-244.

[4] Y. Sohège, M. Quiñones-Grueiro and G. Provan, "A Novel Hybrid Approach for Fault-Tolerant Control of UAVs based on Robust Reinforcement Learning," 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 2021, pp. 10719-10725

[5] S. Mallavalli and A. Fekih, "Adaptive Fault Tolerant Control Design for Actuator Fault Mitigation in Quadrotor UAVs," 2018 IEEE Conference on Control Technology and Applications (CCTA), Copenhagen, Denmark, 2018, pp. 193-198.