4-17-2019

A Model-Free Control Algorithm Based on the Sliding Mode Control Method with Applications to Unmanned Aircraft Systems

Adarsh Raj Kadungoth Sreeraj
ak3092@rit.edu

R.I.T

A Model-Free Control Algorithm Based on the Sliding Mode Control Method with Applications to Unmanned Aircraft Systems

By: Adarsh Raj Kadungoth Sreeraj

A Thesis Submitted in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Mechanical Engineering

Department of Mechanical Engineering

Kate Gleason College of Engineering

ROCHESTER INSTITUTE OF TECHNOLOGY

Rochester, NY

April 17th, 2019

A Model-Free Control Algorithm Based on the Sliding Mode Control Method with Applications to Unmanned Aircraft Systems

By: Adarsh Raj Kadungoth Sreeraj

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Mechanical Engineering

Department of Mechanical Engineering

Kate Gleason College of Engineering

Rochester Institute of Technology

Approved By:

Dr. Agamemnon Crassidis *Date*
Thesis Advisor
Department of Mechanical Engineering

Dr. Amitabha Ghosh *Date*
Department Representative, Thesis Committee Member
Department of Mechanical Engineering

Dr. Mark Kempski *Date*
Thesis Committee Member
Department of Mechanical Engineering

Dr. Jason Kolodziej *Date*
Thesis Committee Member
Department of Mechanical Engineering

Dr. Daniel Kaputa *Date*
Thesis Committee Member
Department of Electrical, Computer and Telecommunications Engineering Technology

ABSTRACT

Control methods require the use of a system model for the design and tuning of the controllers in meeting and/or exceeding the control system performance objectives. However, system models contain errors and uncertainties that also may be complex to develop and to generalize for a large class of systems such as those for unmanned aircraft systems. In particular, the sliding control method is a superior robust nonlinear control approach due to the direct handling of nonlinearities and uncertainties that can be used in tracking problems for unmanned aircraft system. However, the derivation of the sliding mode control law is tedious since a unique and distinct control law needs to be derived for every individual system and cannot be applied to general systems that may encompass all classifications of unmanned aircraft systems. In this work, a model-free control algorithm based on the sliding mode control method is developed and generalized for all classes of unmanned aircraft systems used in robust tracking control applications. The model-free control algorithm is derived with knowledge of the system's order, state measurements, and control input gain matrix shape and bounds and is not dependent on a mathematical system model. The derived control law is tested using a high-fidelity simulation of a quadrotor-type unmanned aircraft system and the results are compared to a traditional linear controller for tracking performance and power consumption. Realistic type hardware inputs from joysticks and inertial measurement units were simulated for the analysis. Finally, the model-free control algorithm was implemented on a quadrotor-type unmanned aircraft system testbed used in real flight experimental testing. The experimental tracking performance and power consumption was analyzed and compared to a traditional linear-type controller. Results showed that the model-free approach is superior in tracking performance and power consumption compared to traditional linear-type control strategies.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
SMC	Sliding Mode Control
MFSMC	Model Free Sliding Mode Control
IMU	Inertial Measurement Unit
PID	Proportional Integral Derivate
NED	North-East-Down
B	Control Input Gain
λ	Slope of the Sliding Surface
K	Switching Gain
φ	Boundary Layer Thickness
Φ	Roll Angle (°)
Θ	Pitch Angle (°)
Ψ	Yaw Angle (°)
u	Control Input
x_d	Desired Tracking

1. INTRODUCTION

There has been a tremendous rise in the field of control systems used in system automation that is becoming more versatile for control highly complex systems. Control systems can be divided into linear and nonlinear control-type categories depending on the differential equations that govern the system behavior. Nonlinear systems are more complex and exhibit unpredictable behavior, requiring complex control algorithms [1]. A simple approach for nonlinear controls is to invert the nonlinear dynamics and replace it with the desired linear dynamics but requires the exact knowledge of the system model. If not known exactly, the inversion may lead to unstable residual dynamics [2]. In addition, effects of unmodeled dynamics (sensors and actuator dynamics) and external noise are also present and therefore, any nonlinear control schemes are required to be robust to such unknown perturbations.

Proportional-Integral Derivative (PID) control is the most popular theory used widely in industry applications. It is also used in complex systems like autonomous cars and Unmanned Aerial Vehicles (UAVs). The PID theory controls the systems states by compensating for the errors. But unfortunately, PID control strategies require definitive knowledge of the system model for proper tuning and are limited to linear and linearizable systems restricting its performance. Additionally, the PID controller needs to be properly tuned specific to each system that it is applied to resulting in a cumbersome and time consuming process.

Another approach is the Sliding Mode Control (SMC) scheme [1]. Due to its robustness, it can be applied to both linear and nonlinear systems with modelling uncertainties. SMC theory transforms the control system into a first-order problem which is easier to control and hence good tracking performance can be achieved. The method has two phases: a reaching phase and a sliding phase.

The reaching phase drives the system towards a “sliding surface” and the sliding phase “slides” the states towards an equilibrium point. Lyapunov’s method is used to ensure asymptotic stability during the reaching phase. A discontinuous term is added to the control law to compensate for the system uncertainties and disturbances. However, the method requires knowledge of the mathematical model of the system and hence it is unique to each system which restricts its use. Therefore, there is a clear need to develop a Model-Free control algorithm based on the Sliding Mode Control (MFSMC) which derives the control law from previous control inputs, system measurements, control input gains and system’s order. An initial MFSMC control strategy was first proposed for Single-Input-Single-Output (SISO) systems [3] and then extended to Multi-Input-Multi-Output (MIMO) systems [4]. The goal of this research is to successfully integrate the proposed control algorithm on an Unmanned Aircraft System (UAS), e.g. quadrotor drone, for motion control and path tracking. In this work, a comparison in the performance of the proposed MFSMC algorithm to a traditional PID controller for tracking precision, power consumption and tuning time is presented.

The integration of the MFSMC algorithm on a UAS would allow easy, accurate and robust control tracking for engineering design purposes compared to model-based robust nonlinear control methods that are complex to implement. MFSMC algorithms saves time and reduces overall development costs in developing autonomous controls systems used in UASs for search and rescue type operations, surveillance and transportation as an example. This work proposes a unique MFSMC approach applied to UASs used in real-world type applications.

2. LITERATURE REVIEW

The literature review presents important conclusions of previous studies related to SMC, MFSMC and related control applications to UAS.

2.1 Generic SMC Schemes

Laghrouche et al. [5] introduced a higher-order sliding mode controller on optimal linear quadratic control applied to a minimum-phase nonlinear SISO systems. The problem was divided into three steps. First, a higher-order sliding mode problem was created to eliminate the chattering effect, followed by characterizing the nonlinear uncertainties as bounded non-structured parametric uncertainties considering the system as an uncertain linear system. Lastly, an optimal sliding mode controller was derived by minimizing a quadratic cost function over finite amount of time. The SMC was tested on a kinematic model of an automobile for steering controls from an initial position to a trajectory defined by the user. A sliding mode control of fourth-order was used with a time varying sliding surface. The system achieved perfect tracking with the error converging to zero with no chattering.

Hai Yu and Ozguner [6] developed a new sliding mode control scheme called Adaptive Seeking Sliding Mode control (ASSM control) for a particular set of nonlinear systems. The work addressed the issue of high feedback control effort when the system faces system disturbances and uncertainties. The method had a floating control gain adjusted adaptively to handle unknown disturbances and uncertainties thus reducing the chattering effect. It was used on the cruise control system of an off-road vehicle for velocity reference tracking control with minor errors and negligible chattering.

Runcharoon and Srichatrapimuk [7] presented an altitude control strategy for a quadrotor using a SMC system. The Euler angles was used to define the altitude (φ = roll, θ = pitch, ψ = yaw) and describe the orientation of the quadrotor. A PD controller was used to control the altitude (z) and the position (x and y) while the equations characterizing the positon and altitude were linearized to quantify the PD control gains. Euler angle assumptions $\varphi \approx \psi \approx 0$ on the x-axis, $\theta \approx \psi \approx 0$ on the y-axis and $\varphi \approx \theta \approx 0$ were applied. A boundary layer was added to the dynamic equations to eliminate the control chattering about the sliding surface. The simulation was able to drive the quadrotor to the desired position and desired orientation and prove the stability of the system and the control inputs. The above authors [7] used a SMC and PD to achieve a stable closed-loop system. The strategy showed an improvement compared to a general PID control system, but it limited the full potential of a SMC which does not required linearization. The presence of under-actuated systems led to the use of two different control methods as it requires manipulation if used with the SMC process severely hindering the application of the proposed control scheme.

Sen et al. [8] introduced an adaptive method SMC for quadrotor helicopters. The work dealt with the estimation of the system uncertainties and perturbation bounds. As these bounds are unknown, they are overestimated thus leading to large gains and subsequently excessive control. The gain is directly proportional to the magnitude of chattering, and hence by estimating these bounds and updating the control law, the magnitude can be reduced. The method was implemented on a quadrotor helicopter. A stable closed-loop system with perfect position tracking with no chattering was achieved. The uncertainties bounds were unknown, which were later estimated.

Xu and Ozguner [9] presented a method to stabilize under-actuated systems using SMC. It uses the method proposed by Olfati – Saber [10] to transform the system into a cascade normal form. Xu and Ozguner applied this method on two nonlinear under-actuated MIMI systems, a

Translational Oscillator with Rotational Actuator (TORA) and a quadrotor UAS. The quadrotor system was similar to the one used by Runcharoon and Srichatrapimuk [7] while the TORA was controlled using a rate bounded PID controller and a sliding mode controller [11]. A stable closed-loop convergence to the desired position was shown to be achieved.

Pai [12] demonstrated that the SMC showed robust tracking in discrete time systems by applying a discrete-time integral SMC on uncertain linear systems. An auxiliary control function was introduced to define the discrete-time sliding mode controller to stabilize the system. The switching surface was designed by extending the integral switching function from continuous to discrete time SMC ensuring quasi-sliding mode is reached [13, 14]. In practice, only the switching surface is approached by discrete-time SMC systems and hence quasi-sliding mode is assured. The method was applied to a discrete-time system and it showed excellent tracking performance with the presence of uncertainties along with stability of the closed-loop system. The integral switching surface in the design process eliminated the reaching phase and chattering was absent due to the absence of a switching gain.

Lee [15] also introduced a discrete-time SMC using a fast output sampling. In the work, the system’s closed-loop eigenvalues are arbitrarily defined while designing the control system focusing on stability and transient response. A boundary layer was introduced in the control law to reduce the chattering effect. The proposed method was tested on a discrete-time controller for a continuous time plant model with a serial type lightly damped resonance. Outstanding step response tracking was achieved which eventually proved that the system’s closed-loop eigenvalues can be arbitrarily assigned.

Ferrara [16] presented the problem of applying SMC in systems with saturating actuators. A sub-optimal sliding mode controller with modifications was used to avoid control input saturation. The problem was the uncertainty in convergence of the sliding variable to zero in a finite amount of time when saturation occurs during reaching phase. The proposed modification decreases the control input magnitude once it reaches the saturation value (reaching phase). The control input magnitude increases again if the switching value was not reached implying the control input remains at the saturation value until a new switching value is reached. It is also proved that the system states converge to the origin in finite time and was reaffirmed with an example while avoiding the saturation limits.

2.2 Model–Free Sliding Mode Control Schemes

The limitation with strategies referenced previously is a system model is required for control development that is typically time consuming to design and also hinders performance in presence of model uncertainties. As dynamical systems become exceeding complex, a simplification is possible by developing a model-free control algorithm based on the sliding mode control.

Martinez-Guerra et al. [17] presented a Sliding Mode Observer (SMO) called master-slave synchronization to determine certain synchronization problem with chaotic behavior. The scheme required an accurate knowledge of the nonlinear system dynamics. Hence a model-free SMO with a promotional correction of the sign function of the synchronization error was introduced. The method was successful applied to the control of a Lorenz system (a nonlinear system exhibiting chaotic behavior when tuned to certain gains).

Salgado-Jimenez et al. [18] applied a model-free high-order SMC on a one degree-of-freedom underwater vehicle for position control. The proposed new method used only the exponential convergence of the desired trajectory, eliminating the need to know the system dynamics or parameters. Chattering was avoided to restrict damage to the actuators lifetime by using a higher-order SMC. However, the controller is integrated to a PD control whose desired gain values and performances requiring tuning. Two trajectories were tested: 1. sine wave; 2. triangular wave. A smooth response was achieved in both cases while the vehicle followed the desired path.

Raygosa-Barahona et al. [19] introduced a model-free back stepping technique with integral SMC to develop a model-free SMC system for under-actuated underwater Remotely Operated Vehicles (ROVs). A model-free controller was obtained by designing a regressor free second-order sliding mode controller as the auxiliary input control at each iteration. The sliding mode is integrated with a PID control and is applied to a ROV to track a helix trajectory. The vehicle states converged to the desired trajectory with no chattering. However, the PID controller requires tuning to achieve the desired performance.

Munoz-Vasquez [20] introduced a method to control the position of a quadrotor using passive Velocity Field (VF) navigation with unknown system dynamics. The controller has three subsystems: 1. model-free control subsystem – responsible for maintaining the sliding mode condition all the time; 2. velocity field subsystem – responsible for designing the velocity field to define the desired path and; 3. sliding surface subsystem – responsible for assembling invariant manifolds of position and orientating sliding surfaces. The controller was tested in a 3D environment without and with obstacles to prove the effectives off the VF to navigate around the obstacles in cluttered environments. The system displayed perfect tracking with no chattering however, the velocity field needs to be designed in order to derive the controller scheme.

Crassidis and Mizov [21, 22] presented a model-free control algorithm based on pure sliding mode control scheme to achieve perfect tracking for linear and nonlinear systems along with asymptotic tracking stability. The controller is designed based on previous control inputs, state measurements and the knowledge of the system order. A boundary layer was introduced into the control law to remove the excessive control chattering effect. The boundary layer reduced the tracking precision and gave a smooth control effort which is required. The method was tested on a first and second-order linear and nonlinear system. All the systems showed perfect tracking using identical controllers and outstanding asymptotic tracking stability was observed.

Crassidis and Reis [3] derived a similar model-free control algorithm based on SMC and extended the applications into SISO systems with non-unitary input gains. The effect of noise and system inaccuracies was also investigated. Firstly, a nonlinear mass-spring damper system with non-unitary control input gain without sensor noise was simulated followed by a state measurement noise using a Gaussian distribution of noise. The variance, mean and probability distribution was obtained from the sensor’s datasheet. Perfect tracking was obtained in both cases and chattering was eliminated by using a boundary layer.

Crassidis and Fares [4] further extended the model-free control algorithm based on the SMC method for MIMO systems and examined the effects of an actuator-induced time delay. The derivation and implementation for square MIMO systems was similar to the one mentioned in [18]. For under-actuated MIMO systems a transformation matrix was introduced to square the control input gain matrix to derive the control law. Perfect tracking was achieved for squared MIMO systems while only certain outputs, as expected, achieved perfect tracking in the under-actuated cases. The transformation diffeomorphism allowed the control of the output for tracking. The method was further applied to single-input nonlinear two mass spring damper system and a quadrotor. The first system achieved perfect tracking on all states with the control effort maximized. But the latter observed perfect tracking on certain outputs while the control efforts and certain outputs displayed high frequency activity. This is due to the aggressiveness of the controller to track the required trajectory entirely. The presence of an actuator time delay had an adverse effect when it exceeded 0.1 seconds. Hence the control law needs to be modified to account for the presence of this time delay. The modified control law handled the time delay inaccuracies but was inconsistent at 0.1 seconds of time delay in some cases.

Levant [23] also presented a unique method of model-free sliding mode control based on Higher Order Slide Mode [HOSM] theory. The controller form is based usually on an insignificant relay controller $u = -Ksgn(S)$ where $S = x - x_d$ satisfies the higher-order sliding modes. No information about the plant and only the relative order (r) is required for the controller since S and its derivatives are based only on the states. The control effort is calculated by integration eliminating the chattering effect without the need for a smoothing step. The method eliminates chattering in the ideal scenario, but chattering may still be present due to the excitation of parasitic dynamics [24]. The parasitic dynamics are unmodeled dynamics such as actuator or sensor delays. The control law was successfully used to steer a four-wheeled vehicle onto a desired trajectory.

Precup [25] developed two distinct methods of model-free sliding mode control based on dynamic data-driven linear estimation of the system model. For a first-order system, the sliding surface is defined as:

$$S = \tilde{x}(t) + \int_0^t \tilde{x}(\tau)d\tau$$

The system model is assumed to be:

$$\dot{y}(t) = F(t) + \alpha u(t)$$

where α is a tunable parameter. It is selected to keep the magnitude of $\dot{y}$ and u same. $F(t)$ is approximated to:

$$\hat{F}(t) = \hat{y}(t) - u(t)$$

where $\hat{y}(t)$ is determined from y by a first-order derivative and a low-pass filter. Replacing the discontinuous sign function and adding a thickness boundary layer and a saturation function, the SMC control law was derived as:

$$u = \alpha^{-1}\left(-\hat{F}(t) + \dot{x}_d(t) - \lambda^{-1}\tilde{x}(t) - e_{est\,max} - \lambda^{-1} sat\left({}^{S}/_{\varphi}\right)\right)$$

where $e_{est\,max}$ satisfies the following inequality:

$$\varphi^{-1}|S(t)|\eta > 2\lambda e_{est\,max}$$

The method is similar to those used in [3, 4, and 21] as it is the adaption of conventional SMC with the system model estimated from only the states and inputs. However, in this method the algebraic loops are avoided using a differentiator rather than a direct state measurement.

2.3 Sliding Mode Control for Under-actuated System

Many practical applications require the SMC to control MIMO systems. Dehghani and Menhaj [26] developed a state-space model for a leader-follower system which omits the effects of flight dynamics. The control inputs are considered translational acceleration in three dimensions. This removed the problem of cross-coupling of control and the under-actuated nature of a conventional aircraft wing. Resolving problems arising from developing control law for under-actuated MIMO systems is an important aspect and numerous solutions are offered.

Qian, Yi and Zhao [27] developed a multi-surface SMC for a single-input multi-output under-actuated system. The proposed method was based on nested sliding variables including all system states. The number of sliding surfaces depends on the number of states. It allows tracking of multiple outputs with a single input. Tunable coefficients were weighted to the outputs to lower leveled sliding surfaces when constructing higher leveled sliding surfaces. The method was validated through a simulation effort on a single and double inverted pendulum for stabilization. The effect for chattering was not mentioned nor handled.

Schkoda [28] developed a squaring transformation diffeomorphism using optimal control theory to decide the weighting of different outputs in the transformation. The method is similar to the approach developed by Raygosa [19]. Virtual control inputs were mapped to the actual control inputs through a transformation diffeomorphism. The transformation matrix in [28] led to a square input matrix which is inverted to derive the SMC control law for MIMO systems.

One major area of application of under-actuated systems is in Unmanned Aircraft Systems (UASs). There are two main configurations of UASs, fixed-wing and quadrotor. Fixed-wing UASs are similar in nature to conventional aircraft. The four traditional control inputs are the rudder, elevator, ailerons, and forward thrust (engine). A quadrotor is a type of helicopter with four equal-sized rotors distributed equally in horizontal plane around the center-of-mass. Typical the rotors employ a fixed blade pitch and implies that torque is the only input. The next section reviews some methods used to develop SMC for UAS addressing the issue of under-actuation.

2.4 Sliding Mode Control for Unmanned Aircraft Systems

Norton et al. [29] developed a fixed-wing UAS system with 12 state outputs but only 4 inputs (rudder, elevator, aileron deflections and thrust). The under-actuation is eliminated by applying a diffeomorphism to the differential equations of the systems. After coordinate transformations, the differential equations are given as four three dimensional equations z_i, with four sliding variables S_i defined as:

$$S_i = z_i - z_{i_d}, \text{ i =1, 2, 3, 4}$$

In the approach, $z_{i_d}$is the desired trajectory to be tracked in the transformed coordinate system. Similar to classic SMC, the sliding variables are differentiated and substituted in the system model. The control laws are developed for thrust and surface deflections along with three virtual controllers to compensate modal uncertainties.

Abdulhamitbilal [30] also developed 6 DOF state-space model for a fixed-wing UAS with 12 states. Unlike assigning a sliding surface to each state as proposed in [14], only four sliding surfaces were developed, one for each input (rudder, elevator, aileron deflections and thrust). The two 6-dimensional state variables (position and velocity) are transformed into a 4-dimensional space with the sliding surface. Transformation matrix using weights for individual states like the one used in [28] is used.

Duan, Mora-Camino and Miquel [2] compared the performance of a decoupled longitudinal fixed-wing UAS model using dynamic inversion and backstepping methods. A full 6 DOF aircraft model with actuator dynamics was simulated but only the longitudinal results were examined. The backstepping method gave smoother responses but the control law was extremely complex and cumbersome to implement.

Brezoescu, Lozano, and Casillo [31] worked on the tracking control in the lateral direction of a fixed-wing aircraft where the single input was the derivative of the yaw rate. The outputs were yaw (heading) angle and orthogonal distance from the required path. Even though the system was under-actuated, the proposed control law was able to regulate both outputs effectively.

Villanueva et al. [32] developed a 6 DOF state-space model for a quadrotor with 12 states. Four different control modes (manual, altitude hold, position hold, and waypoint following) was derived using the super-twisting method of SMC. The under-actuation of the quadrotor was resolved by the addition of pseudo-control inputs to roll and pitch that are dependent on positions in the horizontal plane. The approach was the same method used by Munoz-Vazquez et al. [20] where pitch and roll were removed as the explicit outputs of the system and the remaining 4 states and 4 inputs transform the problem to a fully actuated system. The proposed method is different to the method followed in [4] where all 6 DOF were retained and a transformation matrix was used for the under-actuated system using tracking weights.

Derafa, Benallegue, and Fridman [33] also applied super-twisting on a quadrotor and tested the system in a real-world application. The controller was developed for attitude tracking and stabilization. The desired values are given as functions of desired positions. The system modelled in this method becomes a fully actuated system.

Crassidis and Schulken [34] applied the MFSMC algorithm on a quadrotor and the control algorithm was the same developed by [3, 4, and 21]. The tracking performance was compared to a traditional PID controller. The altitude, roll, pitch and yaw were tracked by both the MFSMC and PID controller. Simulation results showed that the MFSMC and PID controller had similar tracking performance however, without the tedious tuning process required for the PID controller compared to MFSMC algorithm. Real-world flight testing proved that the MFSMC controller can control real-world hardware but the performance was unacceptable. Large tracking errors were observed for both the PID and MFSMC controller. Improved state estimation and system characterization was suggested to improve the tracking performance and reduce tracking error.

The previous works demonstrate there is a clear need for a truly model-free control approach relaying solely on system measurements that can be applied to both linear and nonlinear systems which is the goal of this effort. In addition, the approach should handle modelling uncertainties for both SISO and MIMO using a realistic control effort with minimal control effort chattering for implantation on real-world systems, e.g., quadcopter-type UASs.

3. MODEL FREE SMC ALGORITHM

In this chapter, the derivation for the MFSMC algorithm as proposed in [3, 4] and its application on a second-order nonlinear system is presented. The first section describes the stability of nonlinear systems, autonomous and non-autonomous systems, and square and non-square systems. In the next section, the model-free control algorithm for squared second-order systems followed by the application of the control algorithm on nonlinear systems is developed.

3.1 Lyapunov Stability Theory

Russian mathematician Alexandr Lyapunov introduced the Lyapunov theory to analyze system stability for linear and nonlinear systems [35]. The theory introduces two methods: 1. the Linearization Method and; 2. the Direct Method. The first method linearizes a nonlinear system around an operating point condition and analyzes the stability of the system at that point. The Direct Method uses the concept of energy to quantify stability. The sliding mode control utilizes the Direct Method ensure tracking stability the derived closed-loop control system.

3.1.1 Autonomous and Non-Autonomous Systems

A nonlinear system is said to be autonomous if the system parameters do not depend explicitly on time (Slotine [1]), and is defined as:

$$x_p^n = f_p(x; u_m) \tag{1}$$

Non-autonomous systems are defined as systems that are dependent on time. The state trajectory of an autonomous system is independent of the initial time. Realistically, most system parameters do not vary quickly over time and can therefore be assumed to be autonomous.

3.1.2 Equilibrium Points

An equilibrium point is defined as the point where the system's state trajectories converge to a steady-state condition. Linear systems have one equilibrium point at the origin while nonlinear systems can have several or infinite equilibrium points. The equilibrium point is defined as:

$$f_p(x_{p_e}) = 0 \tag{2}$$

Consider the nonlinear equation of motion of a pendulum:

$$MR^2\ddot{\theta} + b\dot{\theta} + MgR\,sin\,\theta = 0 \tag{3}$$

where, M is mass of the pendulum, R is the length of the pendulum, Θ is the angle between the pendulum and the vertical, b is coefficient of friction, and g is the acceleration due to gravity.

The state-space equation for the system is:

$$\dot{x}_1 = x_2 \tag{4}$$

$$\dot{x}_2 = -\frac{b}{MR^2}x_2 - \frac{g}{R}sin\,x_1 \tag{5}$$

where, $x_1 = n\pi$ are the infinite number of equilibrium points for the pendulum.

3.1.3 Concepts of Stability

An equilibrium point is stable if given a spherical region with radius $R > 0$, there exists $r > 0$, such that if $\|x(0)\| < r$, then $\|x(t)\| < R$ for all $t \geq 0$. Otherwise, the equilibrium point is unstable. The is referred to as Lyapunov stability and outlined in Slotine [1].

An equilibrium point (x_e) is said to be asymptotically stable if it is stable as defined above and there exists some $r > 0$ such that $\|x(0)\| < r$ and $x(t) \to x_e$ as $t \to \infty$. If there exist a point $\|x_2\| < R$ and $x(t) \to x_2$ as $t \to \infty$, then the point is marginally stable. If doesn't meet either of these conditions, then the point is unstable [1].

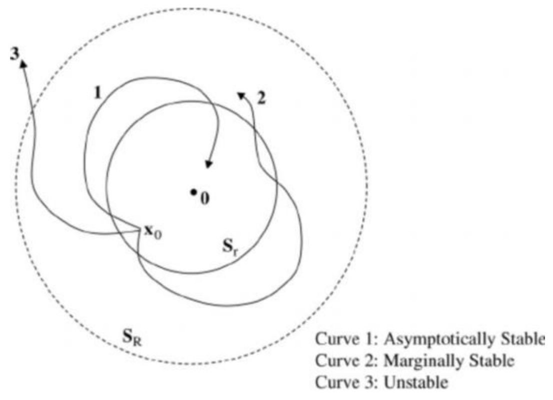


Figure 1: *Concepts of Stability [1]*

Apart from convergence to the equilibrium point, estimates of how fast the trajectories converge to the equilibrium point are also important. An equilibrium point is said to be exponentially stable if there exist two positive numbers α and δ such that for $t > 0, \|x(t)\| \leq \alpha\|x(0)\|e^{-\delta t}$, inside of S_R. The concept proves that the state vector is converging faster than the exponential function to the equilibrium point.

The equilibrium point is said to be globally asymptotically or exponentially stable if asymptotic or exponential stability hold for any and all initial states in the large.

3.1.4 Lyapunov's Direct Method

Lyapunov's Direct Method concept is introduced in this section. The method analyzes the energy of the system. If the total energy of the system is continuously dissipated, be it mechanical or electrical, then the system must settle to an equilibrium point eventually, be it a linear or nonlinear system. The energy variation of the system can be used to study the stability feasibility of the system, hence a function that describes the energy of the system must be defined. The function is referred to as the candidate Lyapunov function, $V(x)$.

The energy function has certain rules to be considered. The function $V(x)$ must be strictly positive for all x and $\dot{x}$ excluding the origin. The function $V(x)$ is said to be positive definite if $V(x) > 0$ for any $x \neq 0$. $\dot{V}(x)$ is said to be negative semi-definite if $\dot{V}(x) \leq 0$. If these conditions are met then the equilibrium point is said to be stable.

The equilibrium point is said to be asymptotically stable if $V(x)$ is strictly positive definite and $\dot{V}(x)$ is strictly negative definite. The equilibrium point is globally asymptotically stable, if $V(0) = 0$, $V(x) > 0$ for any x, $\dot{V}(x) < 0$ for any x, and $V(x)$ is radially unbounded.

3.2 Model Free Sliding Mode Control Scheme

Reis and Crassidis [3] derived a model-free SMC law for SISO linear and nonlinear first and second-order systems with unitary and non-unitary gains. The developed control law testing using simulations for various systems and perfect state tracking was shown to be achieved. El Tin and Crassidis [4] extended the previous work to MIMO linear and nonlinear first and second-order squared and non-squared systems with non-unitary gains. Perfect state tracking and asymptotic tracking stability was achieved in the simulations. In this section, the definitions of squared and non-squared MIMO systems are defined and the MFSMC algorithm for second-order squared MIMO system is derived.

3.2.1 Square and Non-square MIMO Systems

To derive the control law for MIMO systems, the various types of MIMO systems should be considered. Also, the control input gain matrix $[B]$ needs to be invertible for the control law to be derived.

An n^{th}- order autonomous system is defined:

$$x_p^n = f_p(x) + [B]_{p\times m}u_m \tag{6}$$

where *p* and *m* are the number of outputs and inputs respectively, x is the system states and output, $f_p(x)$ is the autonomous nonlinear character in x, *B* is the *p* x *m* control input matrix gains, and *u* is the control input.

A system whose number of inputs is equal to the number of outputs (i.e., $m = p$) is called a square or fully actuated system. Each output system has its own controller and perfect control can be achieved assuming the system is controllable.

When the number of inputs is greater than output ($m > p$) the system is referred to as a non-square system, and is considered over-actuated. In this case, we have an abundance of control inputs and one controller can be replaced by another one.

The non-square system where the number of inputs is less than outputs ($m < p$) is referred to as an under-actuated system. In the non-square systems, the control input matrix [*B*] is not invertible hence further manipulation is required to derive the model-free control algorithm based on the SMC.

3.2.2 System Description for Squared MIMO Systems

Consider the *n*th order autonomous system:

$$x_p^n = f_p(x) + [B]_{p\times m}u_m \tag{7}$$

The system description can be rewritten as:

$$x_p^n = x_p^n + [B]u_m - [B]u_{m_{k-1}} - [B]u_m + [B]u_{m_{k-1}} \tag{8}$$

where p and m are the number of inputs and outputs respectively, x is the system states and output, [*B*] is a square m x m matrix of control input gains, and $u_{m_{k-1}}$ is the previous control input. The error between the control input and the previous control input is defined as:

$$\varepsilon_m = u_{m_{k-1}} - u_m \tag{9}$$

To avoid an algebraic loop within the controller algorithm and to compute the control law, a control input error estimation function is required and is defined as

$$\hat{\varepsilon}_m = u_{m_{k-1}} - u_{m_{k-2}} \tag{10}$$

where, $u_{m_{k-2}}$ is the previous control input of the previous control input. The control input error is not exactly known but it is assumed to be within the following bounds:

$$(1 - \sigma_l)\hat{\varepsilon}_m \leq \hat{\varepsilon}_m \leq (1 + \sigma_l)\hat{\varepsilon}_m \tag{11}$$

where σ_u is the upper bound and σ_l is the lower bound of the control input estimation error.

The control input gain [*B*] is unknown but is assumed to be bounded, i.e.,:

$$b_{lower} \leq b_{mm} \leq b_{upper} \tag{12}$$

where b_{lower} is the lower most bound, b_{upper} is the upper bound and b_{mm} is an element of the control input gain matrix [*B*].

3.2.3 Second Order MIMO System Control Law

A possible time varying sliding surface can be defined as [1]:

$$s_m = (\frac{d}{dt} + \lambda_m)^{n-1}\tilde{x}_p \tag{13}$$

where λ_m is a positive constants, *n* is the order of the system, and $\tilde{x}_p = x_p - x_{d_p}$ defines the tracking error.

The sliding surface for a second-order MIMO system is then defined as:

$$\vec{s} = \dot{\tilde{\vec{x}}} + \lambda\tilde{\vec{x}} \tag{14}$$

By setting the derivative of the sliding surface is to zero ensures the state trajectories remain on the sliding surface on the trajectories have reached the surface, i.e.,:

$$\dot{\vec{s}} = \ddot{\tilde{\vec{x}}} + \lambda\dot{\tilde{\vec{x}}} = \vec{0} \tag{15}$$

Substituting Eq. (8) into Eq. (13) results in:

$$\dot{\vec{s}} = \ddot{\vec{x}} + [B]\vec{u} - [B]\vec{u}_{k-1} + [B]\vec{\varepsilon} - \ddot{\vec{x}}_d + \lambda\dot{\tilde{\vec{x}}} = \vec{0} \tag{16}$$

Rearranging the equation for the control input and adding the discontinuous term to eliminate the system uncertainties results in:

$$\vec{u} = -[B]^{-1}\left[\ddot{\vec{x}} - \ddot{\vec{x}}_d + \lambda\dot{\tilde{\vec{x}}} + \eta sgn(\vec{s})\right] + \vec{u}_{k-1} - \vec{\varepsilon} \tag{17}$$

where η is assumed to be a small positive constant.

3.2.4 Asymptotic Stability of the Control Law

Lyapunov's Direct Method is used to achieve asymptotic stability of the control law during the reaching phase. A candidate Lyapunov function that is strictly positive definite (satisfying one of the Direct Method stability criterion) can be defined as:

$$\vec{V}(\vec{x}) = \frac{1}{2}\vec{s}^2 \tag{18}$$

Differentiating Eq. (16) results in:

$$\dot{\vec{V}}(\vec{x}) = \dot{\vec{s}}\vec{s} \tag{19}$$

Substituting Eq. (14) into Eq. (1) and setting Eq. (17) to be strictly negative to ensure global asymptotic stability results in:

$$\dot{\vec{V}}(\vec{x}) = \vec{s}\left(\ddot{\vec{x}} + [B]\vec{u} - [B]\vec{u}_{k-1} + [B]\vec{\varepsilon} - \ddot{\vec{x}}_d + \lambda\dot{\tilde{\vec{x}}}\right) \leq 0 \tag{20}$$

Substituting the control law Eq. (15) into Eq. (18) yields:

$$\dot{\vec{V}}(\vec{x}) = \vec{s}\left(\ddot{\vec{x}} + [B]\left(-[B]^{-1}\left[\ddot{\vec{x}} - \ddot{\vec{x}}_d + \lambda\dot{\vec{x}} + \eta sgn(\vec{s})\right] + \vec{u}_{k-1} - \vec{\varepsilon}\right) - [B]\vec{u}_{k-1} + [B]\vec{\varepsilon} - \ddot{\vec{x}}_d + \lambda\dot{\vec{x}}\right) \leq 0 \tag{21}$$

Simplifying Eq. (19) results in:

$$\dot{\vec{V}}(\vec{x}) = \vec{s}\big(-\eta sgn(\vec{s})\big) \leq 0 \tag{22}$$

The signum function gives negative unity when the sliding surface is negative and positive unity when the sliding surface is positive. The term " $\vec{s}\big(sgn(\vec{s})\big)$" can be replaced with "$|\vec{s}|$" yielding:

$$\dot{\vec{V}}(\vec{x}) = -\eta|\vec{s}| \leq 0 \tag{23}$$

which is always satisfied as η is strictly positive. This ensures the derivative of the Lyapunov function as negative definite and therefore the asymptotic stability of the system is achieved.

3.2.5 The Switching Gain

The control law Eq. (15) is formally now updated with the switching gain, i.e.,:

$$\vec{u} = -[\hat{B}]^{-1}\left[\ddot{\vec{x}} - \ddot{\vec{x}}_d + \lambda\dot{\vec{x}} + \vec{K}sgn(\vec{s})\right] + \vec{u}_{k-1} - \vec{\varepsilon} \tag{24}$$

where the switching gain $\vec{K}$ ensures that the state trajectories are asymptotically stable during the reaching phase and $[\hat{B}]$ is the estimate of the control input gain matrix.

Using the sliding condition defined in Eq. (21), the asymptotic stability of the closed-loop system is ensured so that:

$$\dot{\vec{s}}\vec{s} \leq -\eta|\vec{s}| \tag{25}$$

The upper bounds of the error estimates are assumed to be conservative and an auxiliary variable is introduced. To ensure the controller handles the most extreme case of inequality and thus solving for $\vec{K}$ yields:

$$\vec{K} = \left|\ddot{\vec{x}} - \ddot{\vec{x}}_d\right|\left|[\beta] - 1\right| + \lambda\left|\dot{\tilde{\vec{x}}}\right|\left|[\beta] - 1\right| + \left|[\hat{B}]\sigma_u(\vec{u}_{k-2} - \vec{u}_{k-1})\right| + [\beta]\eta \tag{26}$$

The control law in Eq. (22) is updated as:

$$\vec{u} = -[\hat{B}]^{-1}\left[\ddot{\vec{x}} - \ddot{\vec{x}}_d + \lambda\dot{\tilde{\vec{x}}} + \vec{K}sgn(\vec{s})\right] + 2\vec{u}_{k-1} - \vec{u}_{k-2} \tag{27}$$

The estimated control input gain is given by the geometric mean of the upper and lower bounds:

$$\left[\hat{\vec{B}}\right] = \sqrt{b_{upper}b_{lower}} \tag{28}$$

The auxiliary variable β is defined as:

$$\beta = \hat{b}\hat{b}^{-1} = \sqrt{\frac{b_{upper}}{b_{lower}}} \tag{29}$$

3.2.6 The Boundary Layer

The discontinuous term in the control law shown in Eq. (25) causes high frequency chattering of the control effort [3, 4] causing damage to the actuators and motors in the real-world. Hence, a smoothing boundary layer is added into to the control law and the algorithm is formulated as:

$$\vec{u} = [B]^{-1}\left[-\ddot{\vec{x}} + \ddot{\vec{x}}_d - \lambda\dot{\tilde{\vec{x}}} - (\vec{K} - \dot{\vec{\Phi}})sat\left(\frac{\vec{s}}{\vec{\Phi}}\right)\right] + 2\vec{u}_{k-1} - \vec{u}_{k-2} \tag{30}$$

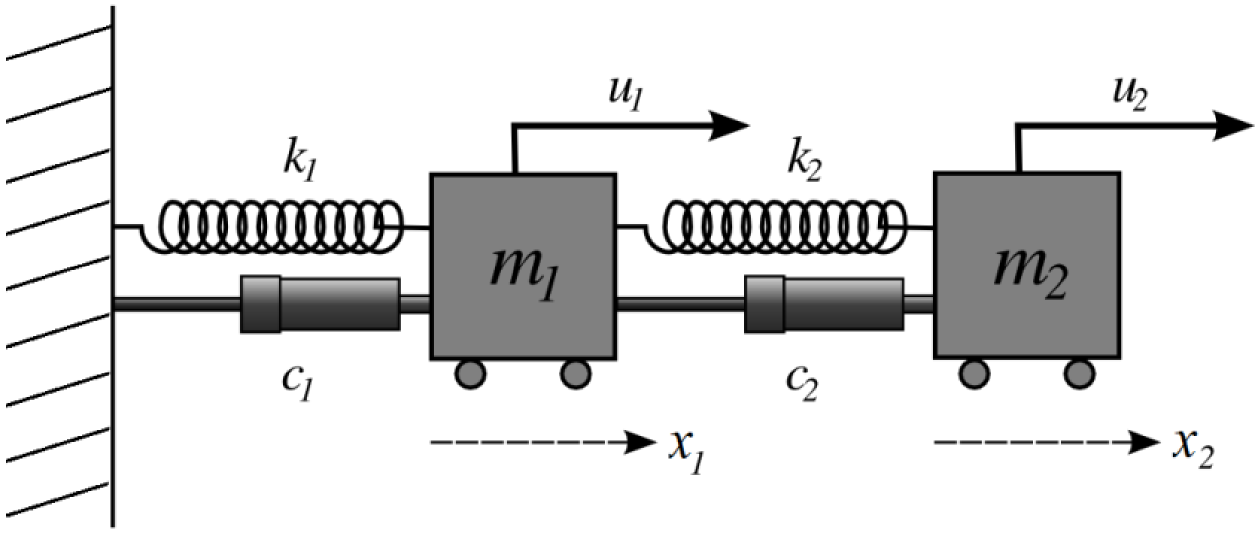
where the boundary layer dynamics are defined as follows (Slotine [1]):

$$\dot{\vec{\Phi}} + \lambda\vec{\Phi} = \vec{K}(\vec{x}_d) \tag{31}$$

with $\vec{\Phi}(0) = \eta/\lambda$. The control effort smoothing approach is similar to applying a "perfect" first-order filter with zero phase loss to the control effort.

3.3 Second Order System Illustration

The above SMC law was applied to a fully-actuated nonlinear mass-spring-damper system. A schematic of the nonlinear mass-spring-damper system is given below:



The equation of motion of system was defined by:

$$m_1\ddot{x}_1 = u_1 + c_2(\dot{x}_2 - \dot{x}_1)|\dot{x}_2 - \dot{x}_1| + k_2(x_2 - x_1) - \beta_2(x_2 - x_1)^3 - c_1\dot{x}_1|\dot{x}_1| - k_1x_1 + \beta_1{x_1}^3$$

$$m_2\ddot{x}_2 = u_2 - c_2(\dot{x}_2 - \dot{x}_1)|\dot{x}_2 - \dot{x}_1| - k_2(x_2 - x_1) - \beta_2(x_2 - x_1)^3$$

where m_i is the mass, x_i is the state output, u_i is the input, c_i ($\vec{c} = [5{,}8]'$) is the damping coefficient, k_i ($\vec{k} = [3{,}7]'$) is the spring constant, β_i ($\vec{\beta} = [-1.5, -3]'$) is the spring stiffening coefficient and *i* is the number of masses, springs and dampers.

The control input gain is [*B*] = [1/*m1*, 1/*m2*] ′ with known bounds:

$$5 \le m_1 \le 15$$

$$15 \le m_2 \le 25$$

The signals to be tracked are:

$$x_{1_d}(t) = \sin\left(\frac{\pi}{2}t\right)$$

$$x_{2_d}(t) = \sin\left(\frac{\pi}{2}t\right)$$

The same controller parameters used in [4] was used and are given below:

Table 1: *Controller Parameters for Second-Order Nonlinear MIMO System*

Parameter	Value
σ_u	0.5
λ	20
η	0.1

The closed-loop simulation results with the derived control law are given below:

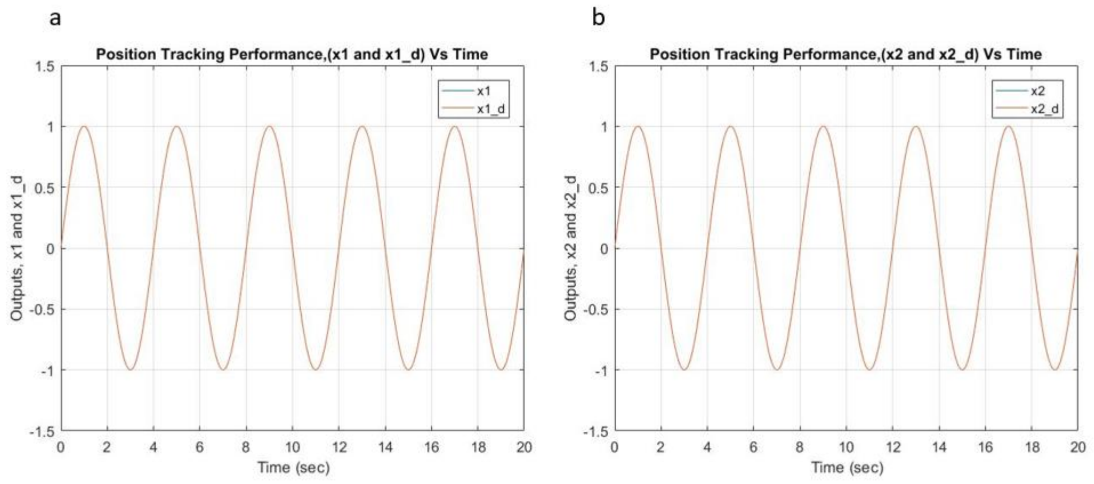


Figure 2: *Position Tracking of (a) X1, (b) X2*

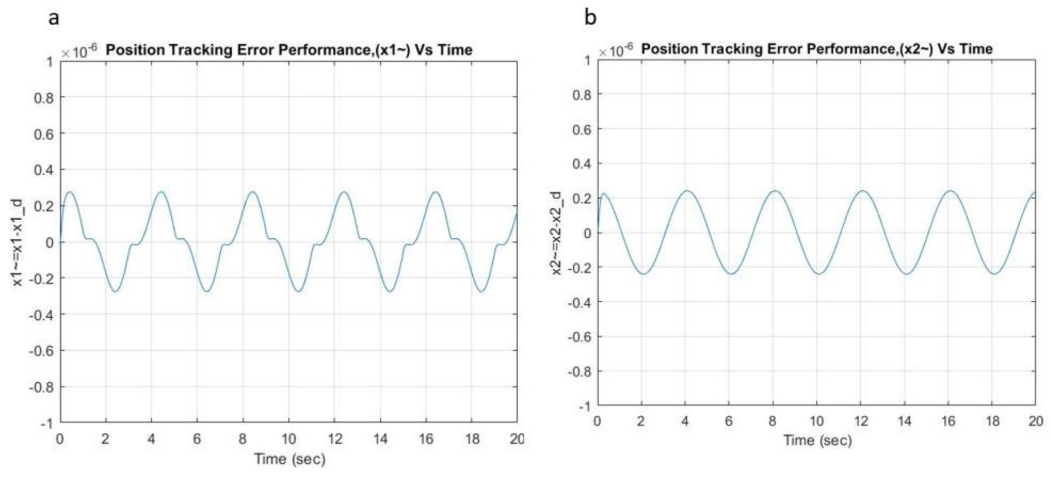


Figure 3: *Position Tracking Error of (1) X1, (b) X2*

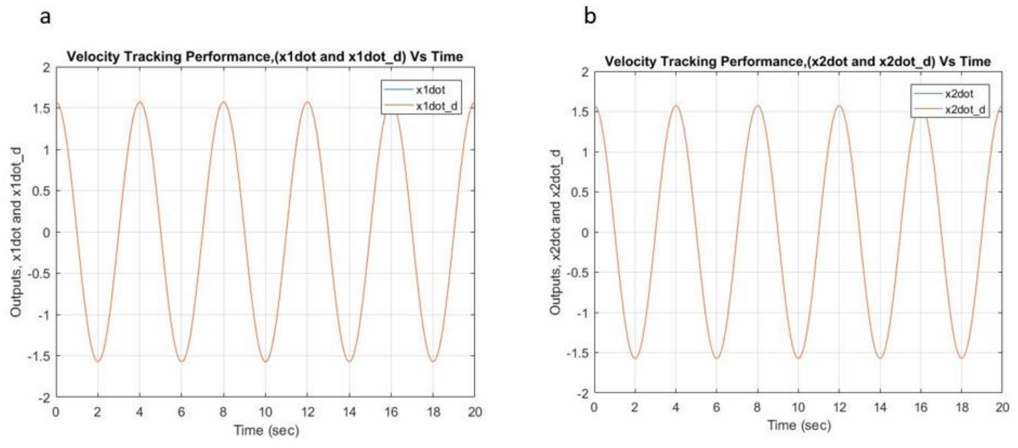


Figure 4: *Velocity Tracking of (a) X1, (b) X2*

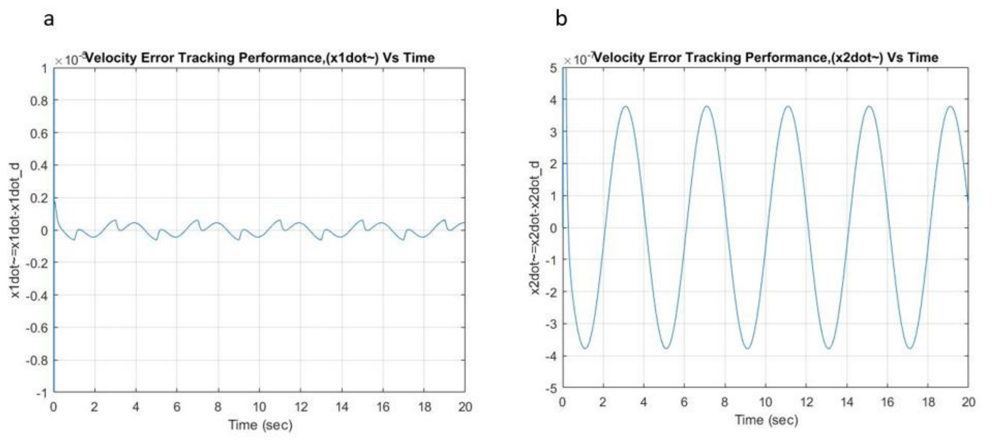


Figure 5: *Velocity Tracking Error of (a) X1, (b) X2*

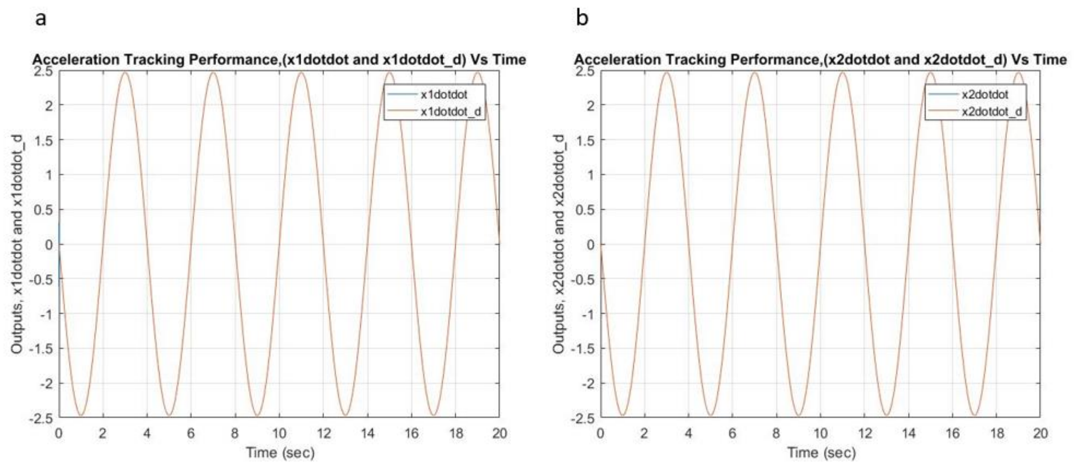


Figure 6: *Acceleration Tracking of (a) X1, (b) X2*

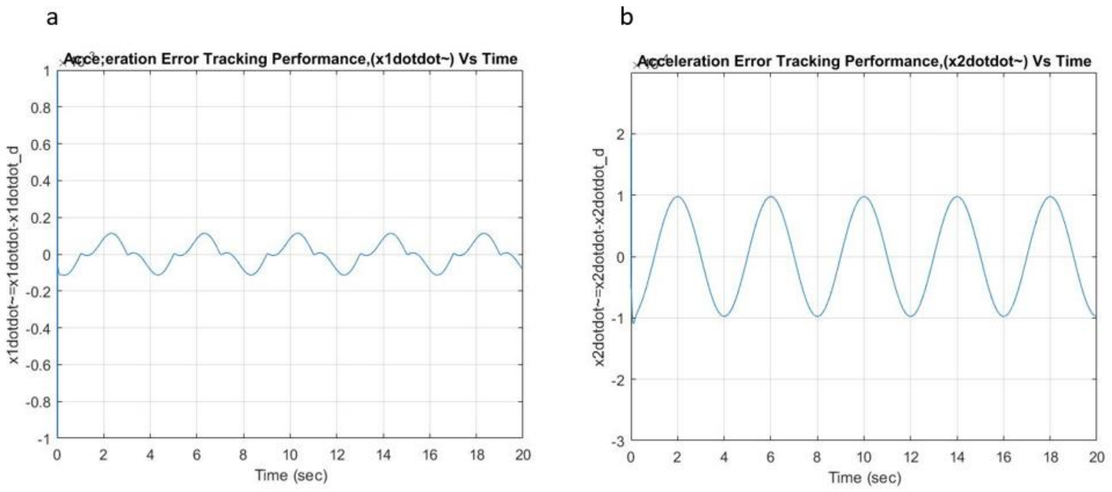


Figure 7: *Acceleration Tracking Error of (a) X1, (b) X2*

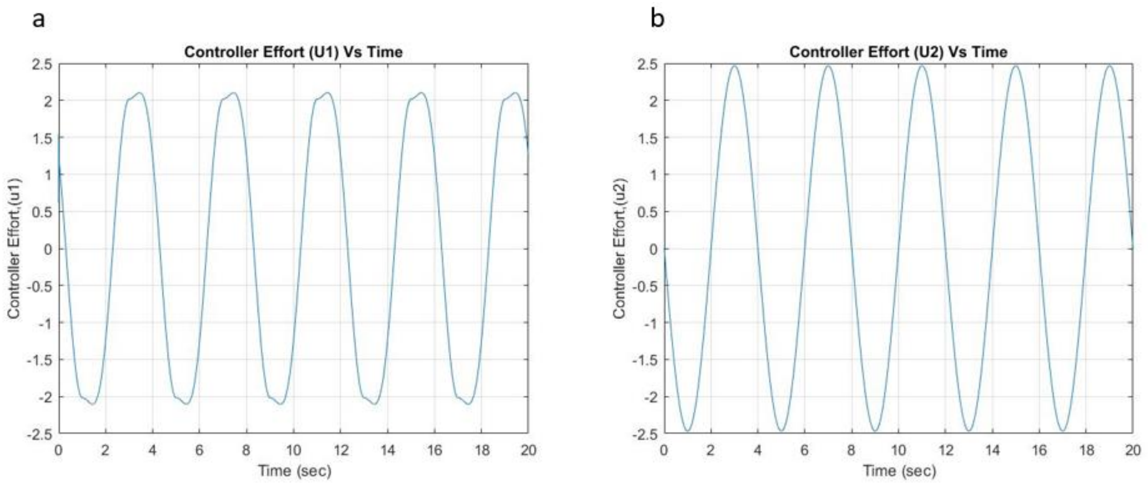


Figure 8: *Controller Effort (a) U1, (b) U2*

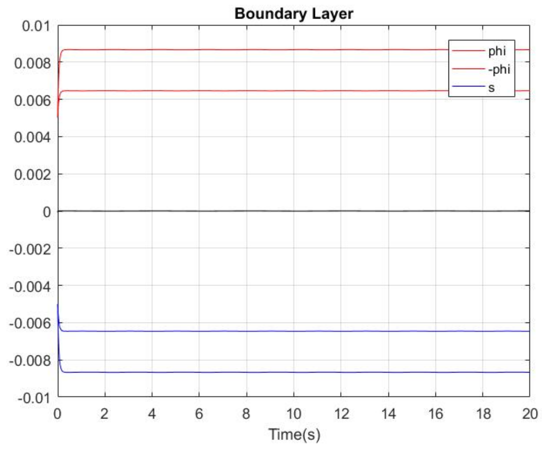


Figure 9: *Sliding Condition Constraint*

From the above results, perfect state trajectory tracking (as shown in Figures 2, 4, and 6) is achieved with negligible error of the order of 10^{-3} for the higher-order system. Perfect tracking for both the mass tracking states is achieved as expected since the system is square. The inclusion of the boundary layer ensures no control chattering in the system as shown in Figure 8. An initial spike in the error is present for the higher-order states due to the inability to initialize the higher-order states. Asymptotic tracking stability of the system is also achieved by satisfying the sliding condition as shown in Figure 9.

4. METHODOLOGY

The objective of this work is to implement the exclusive and unique model-free control algorithm on real-world hardware and to compare the performance to a traditional PID controller with respect to tracking precision and power consumption. The Root-Mean-Square (RMS) of the tracking error (desired trajectory – actual trajectory) is used to compare the tracking performance. As shown in Section 2, to date there has been no work that has used the MFSMC on a real hardware successfully. Schulken [34] implemented the developed MFSMC on an actual hardware but the performance was unacceptable. A quadcopter was selected for hardware implementation in this study. More details about the hardware are mentioned later in Chapter 6.

In Chapter 5, the simulation study of the quadcopter with both PID and MFSMC law is presented. Simulations are used to study system behavior and performance prior to implementing the newly developed control strategy on actual hardware. This identifies critical system (model errors, actuator dynamics, Inertial Measurement Unit (IMU) errors and sensor noise) behavior that could cause damage or lead to hardware damage during flight testing and saves valuable time and effort during hardware development. The simulation study was conducted on a basic quadcopter dynamic model followed with including hardware components such as actuator dynamics, sensor delays, state estimation, IMU sensor noise and controller sampling frequency.

After simulation results are analyzed and studied, destructive system behaviors are identified and the control law is revised for implementation. The hardware study and results are described in Chapter 6. Hardware tests were performed on a gimbal platform with autonomous control for both pitch and roll. Controller parameters were amended to optimize the tracking precision. The system performance was compared with a PID controller for tracking precision and power consumed. Finally, the conclusions and future work for this study are summarized.

5. SIMULATION STUDY

This chapter describes the simulation studies conducted on the dynamic model of a quadcopter with the proposed MFSMC algorithm and the PID controller. The chapter is divided into six sections. The first section describes the dynamic model of the quadcopter and delves into the plant model states, the reference frames, the various quadcopter orientation configurations, the plant model equations, the thrust equations, and the motor dynamics of a basic quadcopter. The second section characterizes the actuator dynamics required in the proposed MFSMC algorithm followed by the basic simulation testing (shown in the third section) of the MFSMC algorithm and the predesigned PID controller on the quadcopter model developed in first section of the chapter. The design of the control law in the MFSMC algorithm to actuator efforts is important as mentioned by El Tin [4]. The fourth section describes state estimation that is required for hardware implementation followed by state conversion into body-reference frame components outlined in the fifth section of this chapter. Noise characterization of signals sensed by the IMU is illustrated in final section with MFSMC algorithm parameter selection as mentioned by Reis [3]. Finally, a simulation study of real-world hardware is carried out under the MFSMC law and the performance is compared to a traditional PID controller for a quadcopter.

5.1 Quadcopter Plant Model

5.1.1 Plant Model States

The quadcopter is a 6 DOF system that travels in the *X*, *Y* and *Z* directions and rotates about the *X* (Roll), *Y* (Pitch), and *Z* (Yaw) axes. The state variables define the physical dynamics of the quadcopter which are the angles of rotation and absolute positions. The local frame is assumed to be the reference coordinate frame (Figure 10) for the quadcopter simulation.

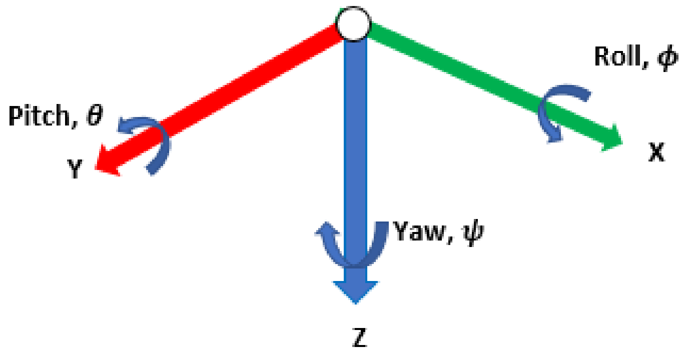


Figure 10: *Quadcopter State Coordinate Visualization* [36]

Quadcopters have various sensors that measure the different states of the quadcopter. The sensors include accelerometers, gyroscopes, vision, GPS etc. A state that can be directly measured using a sensor is referred to as a fully observable state. The quadcopter used in this study uses an IMU which comprises of a full 3-axis accelerometer and gyroscope. The states of the quadcopter are described in Table 2.

Table 2: *Quadcopter States [36]*

Symbol	**Description**	**Unit**	**Observability**
θ	Pitch Euler Angle	rad	Stereo Vision
$\dot{\theta}$	Pitch Euler Angular Velocity	rad/s	Gyroscope
$\ddot{\theta}$	Pitch Euler Angular Acceleration	$rad\ /s^2$	-
ϕ	Roll Euler Angle	rad	Stereo Vision
$\dot{\phi}$	Roll Euler Angular Velocity	rad/s	Gyroscope
$\ddot{\phi}$	Roll Euler Angular Acceleration	$rad\ /s^2$	-
ψ	Yaw Euler Angle	rad	Stereo Vision
$\dot{\psi}$	Yaw Euler Angular Velocity	rad/s	Gyroscope
$\ddot{\psi}$	Yaw Euler Angular Acceleration	$rad\ /s^2$	-
X	Position along X	m	Stereo Vision

$\dot{X}$	Velocity along X	m/s	-
$\ddot{X}$	Acceleration along X	m/s^2	Accelerometer
Y	Position along Y	m	Stereo Vision
$\dot{Y}$	Velocity along Y	m/s	-
$\ddot{Y}$	Acceleration along Y	m/s^2	Accelerometer
Z	Position along Z	m	Stereo Vision
$\dot{Z}$	Velocity along Z	m/s	-
$\ddot{Z}$	Acceleration along Z	m/s^2	Accelerometer
$\Omega1$	Motor 1 Angular Velocity	rad/s	Voltage/Current/Optical
$\Omega2$	Motor 2 Angular Velocity	rad/s	Voltage/Current/Optical
$\Omega3$	Motor 3 Angular Velocity	rad/s	Voltage/Current/Optical
$\Omega4$	Motor 4 Angular Velocity	rad/s	Voltage/Current/Optical

5.1.2 Reference Frames and Orientations

The two reference frames are assumed to be: 1. local NED (North – East – Down) reference frame and; 2. body-reference frame. The NED reference is an inertial frame while the body-reference frame is a non-inertial frame and is assumed fixed on the UAS and moves along with the UAS as the UAS maneuvers in a 3D space.

In the NED reference frame, the X axis is pointed towards north, Y axis is pointed to the east and the Z axis is pointed downwards normal to the X and Y plane, a right handed frame. Positive roll angle translates the quadcopter towards the east and positive pitch angle translates the quadcopter south. Positive yaw rotates the quadcopter clockwise. Figure 11 describes the local NED frame.

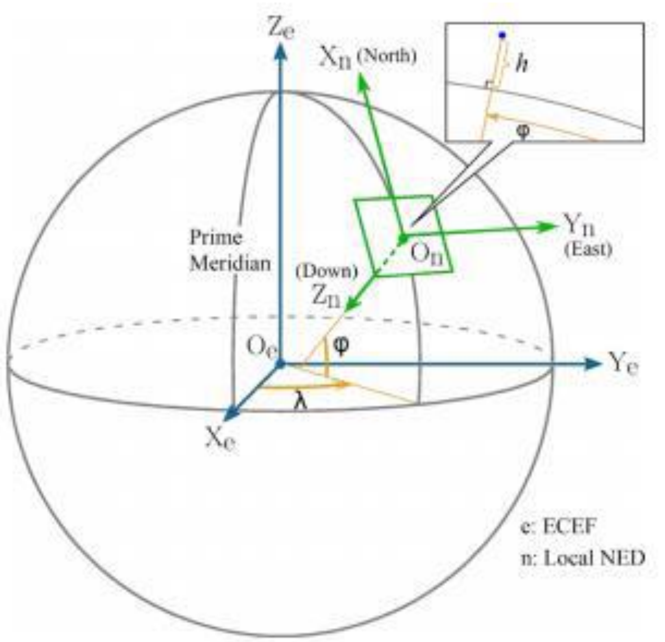


Figure 11: *Local NED Right Handed Frame* [36]

The body-reference frame is dependent on the orientation of the quadcopter. Generally the quadcopter has two orientations: 1. '+' orientation and; 2. '*X*' Orientation. The direction of positive *Z* is common in both orientations and acts downwards. The difference in the orientations depends on the direction of the *X* and *Y* axes and the motors used to change direction. In this study, the '*X*' orientation is used. The body-reference frame for both orientations are:

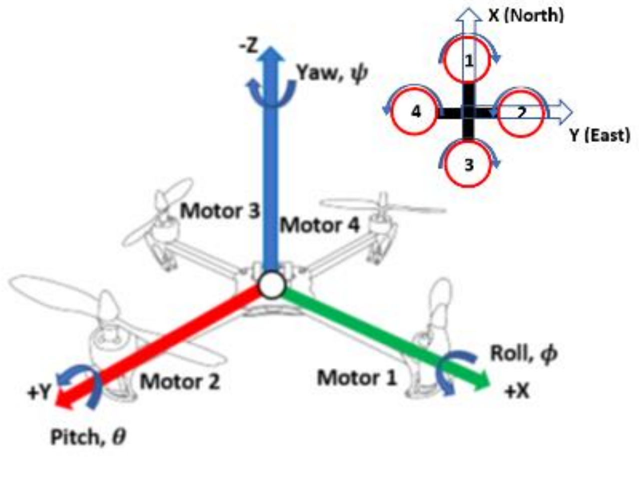


Figure 12: *Quadcopter '+' orientation [36]*

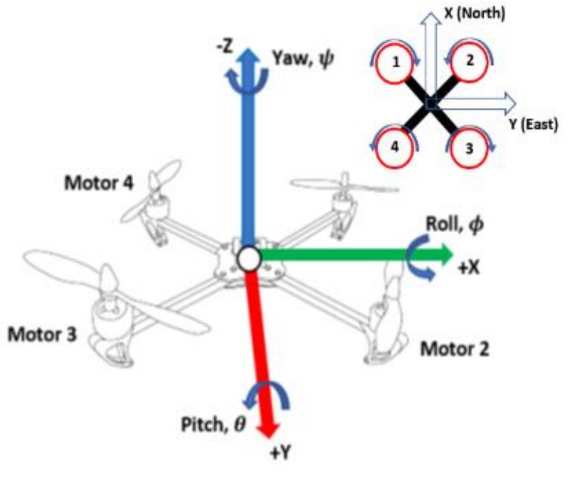


Figure 13: *Quadcopter 'X' orientation[36]*

5.1.3 Plant Model Equations

In this section, a mathematical model of the 6 DOF quadcopter is presented as shown in [4, 9, 34, and 36] to develop a full quadcopter system model. The assumptions for the model are [36]:

- The quadcopter is a rigid structure;
- The air frame and components are symmetric;
- The center-of-gravity and air frame origin coincide;
- Thrust and drag are proportional to the square of propeller speed;
- There are no external disturbances, i.e., wind, temperature etc. acting on the quadcopter.

First, the angular accelerations of the quadcopter are modeled. These include all the moments acting on the quadcopter, i.e., the rolling moments, pitching moments and yawing moments. Each of the moments consists of the following effects:

- Body gyroscopic effects – changes in quadcopter orientation;
- Propeller gyroscopic effects – propeller rotation with quadcopter frame orientation;
- Actuation effects – forces produced by rotor.

The sum of these effects in the respective axes output the angular accelerations for roll (*Φ*), pitch (*Θ*) and yaw (*ψ*). Figure 14 illustrates the effects in the acceleration equations.

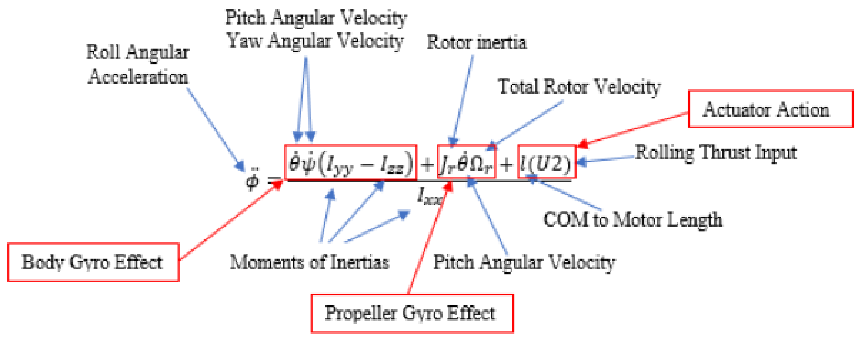


Figure 14: *Angular Acceleration Observations* [36]

Similarly, the pitch and yaw angular accelerations can be generated. The three angular accelerations used for in the quadcopter model are given below:

$$\ddot{\Phi} = \frac{\dot{\theta}\dot{\psi}(I_{yy}-I_{zz}) + J_r\dot{\theta}\Omega_r + lU2}{I_{xx}} \tag{32}$$

$$\ddot{\theta} = \frac{\dot{\Phi}\dot{\psi}(I_{zz}-I_{xx}) + J_r\dot{\Phi}\Omega_r + lU3}{I_{yy}} \tag{33}$$

$$\ddot{\psi} = \frac{\dot{\theta}\dot{\Phi}(I_{xx}-I_{yy}) + U4}{I_{zz}} \tag{34}$$

where *U2*, *U3* and *U4* are roll, thrust and yaw thrust respectively. Ω_r is total velocity of all the propellers and will be discussed further in the next section. I_{xx}, I_{yy} and I_{zz} are the moment-of-inertias (Kg-m^2) terms, J_r is the rotor inertia of the motors (Kg-m^2) and, *l* is the distance between the rotor axis and quadcopter center-of-mass.

Similarly, the accelerations along the translational axes can be defined. Figure 15 depicts the different aspects of the acceleration equations.

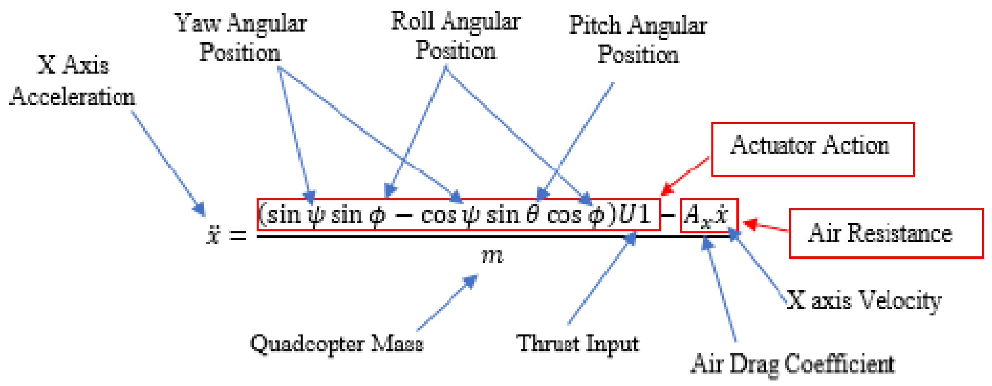


Figure 15: *Acceleration Equation along the "X" Direction [36]*

By following a similar approach the accelerations in *Y* and *Z* can be defined. The three translational accelerations are given as:

$$\ddot{X} = \frac{(\sin\psi\ \sin\Phi - \cos\psi\sin\theta\cos\Phi)U1 - A_x\dot{X}}{m} \tag{35}$$

$$\ddot{Y} = \frac{(\cos\psi\ \sin\Phi - \sin\psi \sin\theta \cos\Phi)U1 - A_y\dot{Y}}{m} \tag{36}$$

$$\ddot{Z} = \frac{mg - (\cos\theta \cos\Phi)U1 - A_z\dot{Z}}{m} \tag{37}$$

where A_x, A_y, and A_z are the air resistances in the respective axes (kg/s), *U1* is the general thrust and *m* is the mass of the quadcopter (kg). Eqs. (30 – 35) finalize the mathematical model of the 6 DOF quadcopter model. The system equations are referenced to the local NED reference frame and are used for simulation study to evaluate the MFSMC performance.

5.1.4 Thrust Equations

There are four main thrust equations that should be considered for inclusion in the model. The thrust variables were included in the plant model equations in the previous section. The four thrust parameters are the general thrust (*U1*), roll thrust (*U2*), pitch thrust (*U3*), and yaw thrust (*U4*). The thrust equations are required to model the individual rotor characteristics that drive and control the quadcopter and are orientation dependent. The thrust equations with respect to the '*X*' orientation are given as follows:

$$U1 = b\left(\Omega_1{}^2 + \Omega_2{}^2 + \Omega_3{}^2 + \Omega_4{}^2\right) \tag{38}$$

$$U2 = b\,sin(\tfrac{\pi}{4})\left(\Omega_1{}^2 - \Omega_2{}^2 - \Omega_3{}^2 + \Omega_4{}^2\right) \tag{39}$$

$$U3 = b\,sin(\tfrac{\pi}{4})\left(\Omega_1{}^2 + \Omega_2{}^2 - \Omega_3{}^2 - \Omega_4{}^2\right) \tag{40}$$

$$U4 = d\left(\Omega_1{}^2 - \Omega_2{}^2 + \Omega_3{}^2 - \Omega_4{}^2\right) \tag{41}$$

where *b* is the thrust coefficient (N/s^2), *d* is the drag coefficient (Nm/s^2), and Ω_n are the speed of the n^{th} motor (RPM). Figure 16 displays a visual interpretation of the thrust equations.

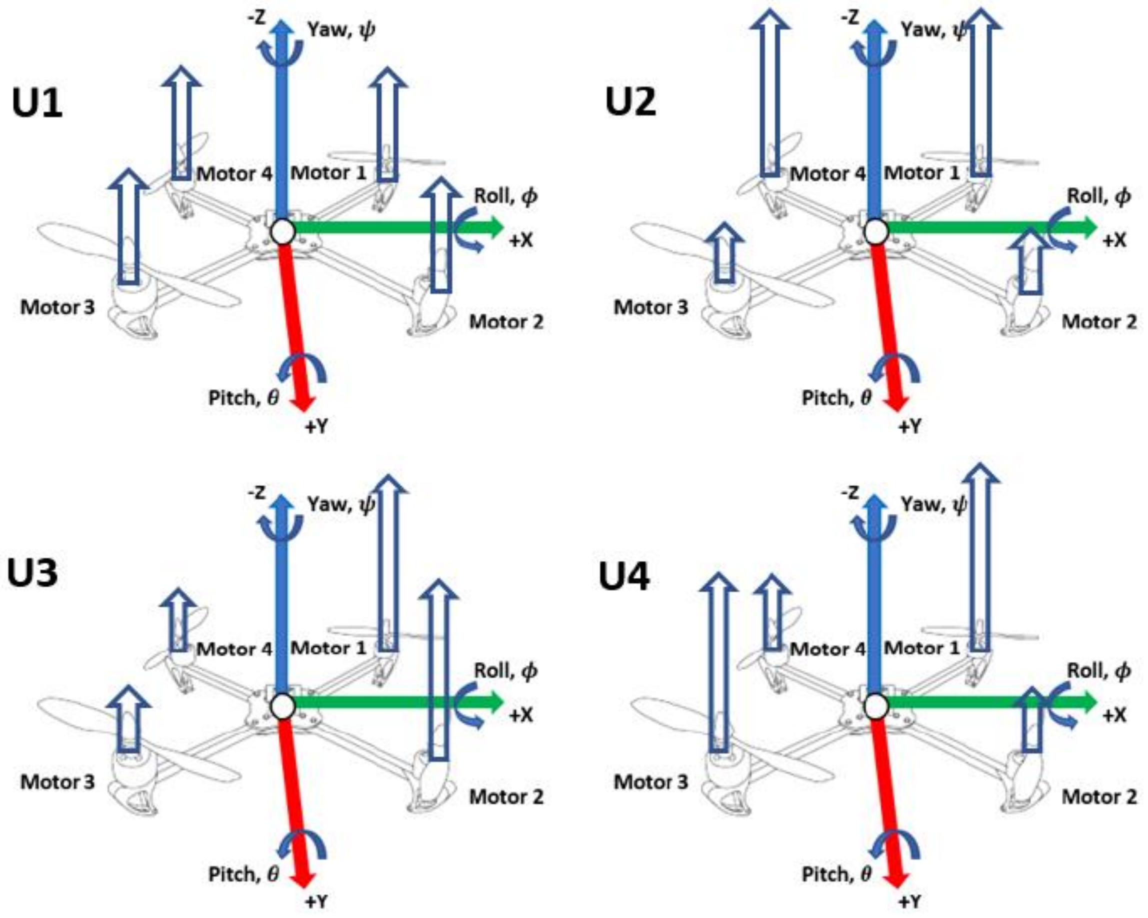


Figure 16: *Thrust Equations in Along the 'X' Orientation* [36]

For example, as shown in Figure 16, *U1* applies a general thrust which is uniform and equal to each of the motors that drive the individual rotors. *U2* changes the thrust between the motors to roll the quadcopter while *U3* changes the thrust between motors to pitch the quadcopter. Similarly, *U4* is used to yaw the quadcopter.

The overall angular velocity of the quadcopter is given as:

$$\Omega_r = \Omega_1 - \Omega_2 + \Omega_3 - \Omega_4 \tag{42}$$

There is a sign difference as motors *2* and *4* are spun in the opposite direction as motors *1* and *3*.

5.1.5 Motor Dynamics

Motor dynamics play an important role in the system dynamics as they cause actuator delays in the closed-loop system. To understand the motor behavior and characteristics, the dynamics must be modeled into the quadcopter model. A brushless DC motor with a controller is modeled to ensure the motor is operating at the proper desired angular velocity. The inputs to the motor are voltage and desired angular velocity (calculated in the previous section). The motor states are current and angular acceleration and the output of the motor is angular velocity. The motor modeling equations are given as:

$$L\frac{di}{dt} = V - RI - K_{emf}\Omega \tag{43}$$

$$J\frac{d\omega}{dt} = K_t I - b\omega \tag{44}$$

where L in the electrical inductance (H), I is the current (ohm), V is the voltage (V), R is the electrical resistance (ohm), K_{emf} is the back electromotive force coefficient (V-s/rad), J is the rotor moment-of-inertia (kg-m^2), K_t is the torque coefficient (Nm/A), and b is the damping coefficient (N-m-s). The Simulink block model of the motor is shown in Figure 17.The saturation blocks limits the voltage and current as supplied by the battery.

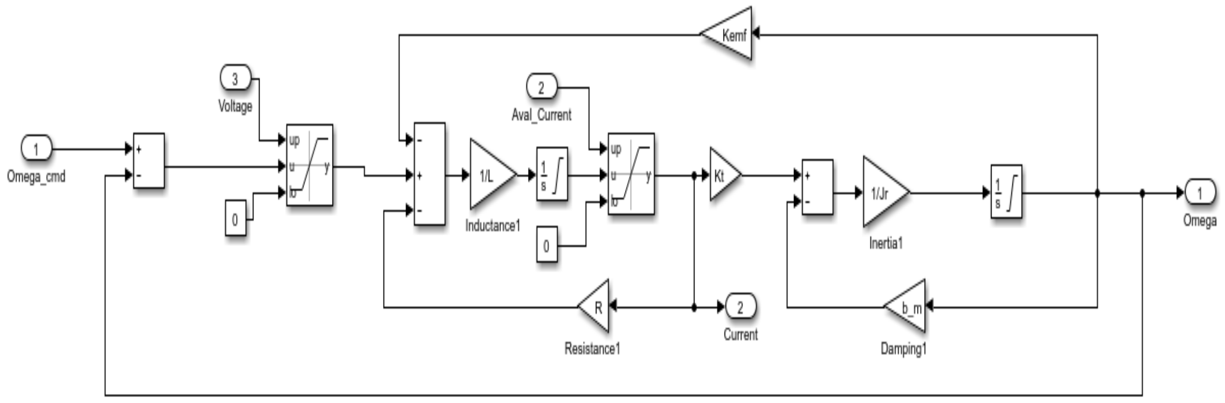


Figure 17: *Simulink Motor Block Model*

5.2 Control Approach for Quadcopter

Depending on the number of degree-of-freedoms to be controlled, the control system can be treated as either a fully actuated or under-actuated system. Xu [9], El Tin [4] and Schulken [34] controlled *X*, *Y*, *Z* and ψ (yaw) in their simulation studies and considered the under-actuated system approach. Xu and El Tin used *U1* to control the altitude *Z* and *U4* to control yaw angle (ψ). *U2* and *U3* were used to control *X* and *Y*, respectively. El Tin used a transformation diffeomorphism approach to convert the under-actuated system into a fully actuated system [4].

Schulken [34] used the super-twisting approach to control *X* and *Y* while the altitude and yaw were directly controlled by the control inputs and was the same approach used in [32 and 33]. In this work, the pseudo-control U_x and U_y signals are generated from the desired *X* and *Y* commands. The pseudo-inputs can be considered as the portion of *U1* on the horizontal and vertical direction. The pseudo-inputs are defined by the following:

$$U_x = sin\,\Phi\, sin\,\psi + cos\,\Phi\, sin\,\theta\, cos\,\psi \tag{45}$$

$$U_y = -\,sin\,\Phi\, sin\,\psi + cos\,\Phi\, sin\,\theta\, cos\,\psi \tag{46}$$

These inputs are used to define the required roll and pitch angles using the following:

$$\Phi_d = sin^{-1}(U_x\, sin(\psi) - \, U_y\, cos(\psi)) \tag{47}$$

$$\theta_d = sin^{-1}(\tfrac{U_x\, cos(\psi) + U_y\, sin(\psi)}{cos(\Phi_d)}) \tag{48}$$

The MFSMC law calculates the required control input, i.e., $\vec{u}$:

$$\vec{u} = [B]^{-1}\left[-\ddot{\vec{x}} + \ddot{\vec{x}}_d - \,\lambda\dot{\vec{\tilde{x}}} - (\vec{K} - \dot{\vec{\Phi}})sat\left(\frac{\vec{s}}{\vec{\Phi}}\right)\right] + \,2\vec{u}_{k-1} - \,\vec{u}_{k-2} \tag{49}$$

In this study, the four states to be controlled are altitude (*Z*), roll (*Φ*), pitch (*Θ*) and yaw (ψ). Roll and pitch are used to translate the quadcopter along the *Y* and *X* direction, respectively. Each of the commands have its individual control input hence the need for super-twisting and transformation is eliminated. It is, therefore, considered as a fully actuated control system even though it is primarily an under-actuated system.

Eq. (28) is used to calculate the four control inputs (i.e., *U1*, *U2*, *U3*, and *U4*). The control inputs are used to calculate the individual motor speeds (Ω_1, Ω_2, Ω_3, Ω_4) using Eqs. (36 – 39). Figure 18 displays the Simulink block diagram for the motor speed calculations.

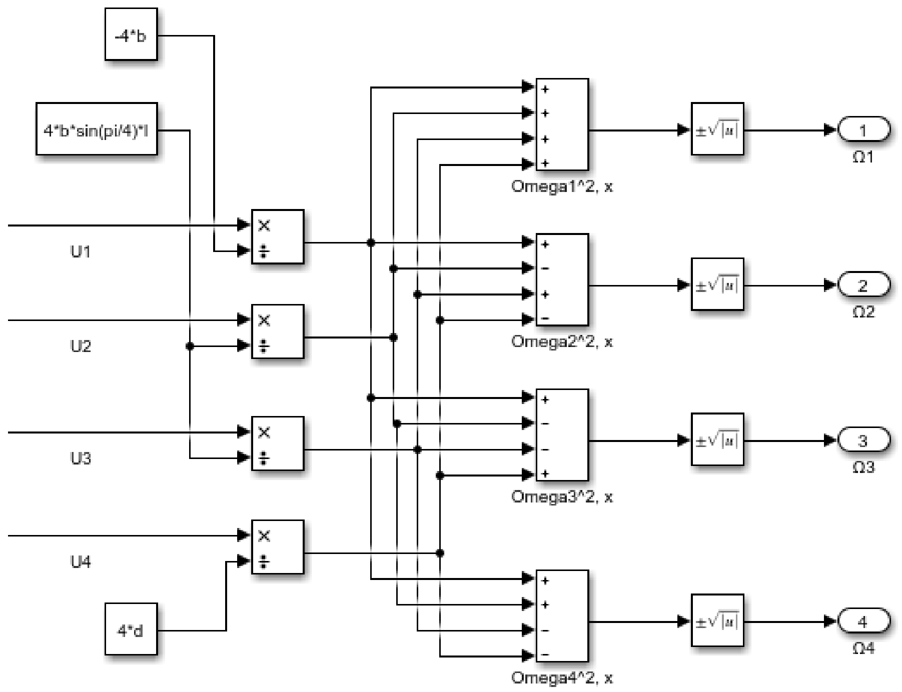


Figure 18: *Motor Speed Calculation from the Control Inputs*

The calculated motor speeds are the output of the controller block and act as the input to the plant block. They are fed into the motor model in the plant subsystem.

Another important consideration to be noted while using the MFSMC algorithm is that the measurements signals are required to be smooth and twice differentiable. Sliding Mode differentiators are used to find the derivatives of the signals, as the approach does not introduce phase lag compared to convention differentiation methods [37]. Details on Sliding Mode differentiators are discussed in the later sections.

5.2.1 Actuator Dynamics

El Tin [4] successfully updated a MFSMC algorithm to handle unknown actuator dynamics (i.e., motor dynamics) by measuring the previous control input values from the actuator rather than from the output of the controller. Based on this approach the control law is updated as:

$$\vec{u} = [B]^{-1}\left[-\ddot{\tilde{\vec{x}}} + \ddot{\vec{x}}_d - \lambda\dot{\tilde{\vec{x}}} - (\vec{K} - \dot{\vec{\Phi}})sat\left(\frac{\vec{s}}{\vec{\Phi}}\right)\right] + 2\vec{u}_{pa_{k-1}} - \vec{u}_{pa_{k-2}} \tag{50}$$

where u_{pa} is the control input post the actuator. The included actuator dynamics model in the closed-loop control law is shown in Figure 19 extracted from the overall Simulink diagram:

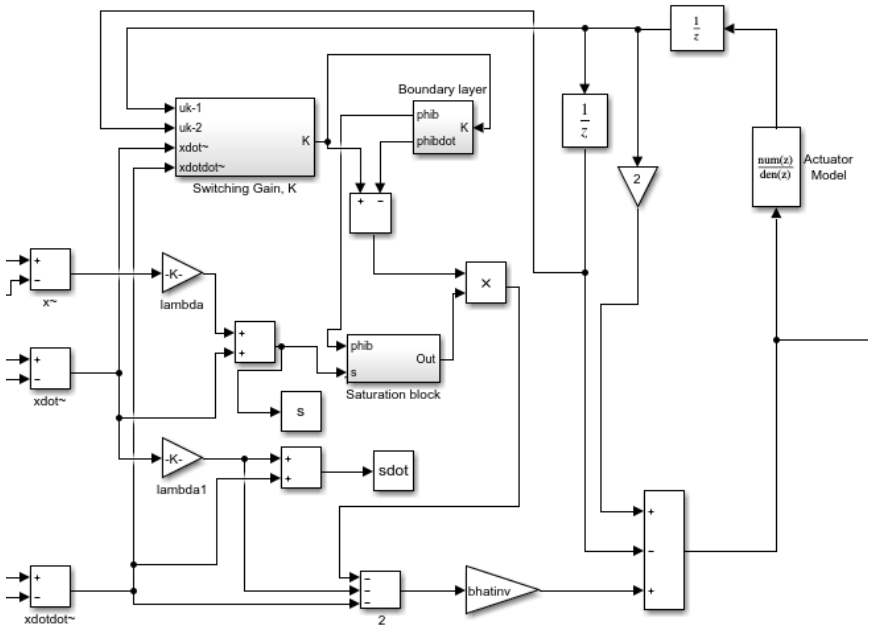


Figure 19: *Updated Control Law Model with Actuator*

A first-order transfer function is used as the actuator model. The transfer function is converted into a discrete transfer function in Matlab using the "c2d" function. The first-order continuous transfer function is given by the following:

$$H(s) = \frac{1}{\tau s + 1} \tag{51}$$

where τ is the actuator time constant, i.e., is the mechanical time constant of the motor.

5.3 Basic Quadcopter Simulation

A basic simulation of the quadcopter using Eqs. (30 – 39) is performed to compare the performance of the PID controller tuned using the system model [36] and the MFSMC approach. In the simulation, hardware factors such as IMU, noise and no sensor delays were considered. The simulation was run using the Simulink ode5 solver at a fixed time step of 0.0001s. The quadcopter model parameters m, I_{xx}, I_{yy} and I_{zz} were set to be within 10% of the nominal values. The parameters λ, σ_u and η were set to 1 rad/s, 0.5 and 0.15, respectively.

Figures 20 – 23 displays the time history tracking responses for altitude, roll angle, pitch angle, and yaw angle for a given desired tracking response. The simulations results, as observed in Figure 20 – 23, show outstanding tracking performance for both the PID and MFSMC law with similar performance. Figure 24 displays the switching gains for the MFSMC law for each tracking signal and are positive as expected. Figure 25 displays the sliding condition for the each individual MFSMC law and proves asymptotic tracking stability is maintained since the sliding condition is satisfied. The sliding condition for altitude at the start of the time history is not satisfied due to initialization which is acceptable since during actual hardware implementation the initialization issue is resolved. Figure 26 displays the control efforts (*U1 – U4*) and indicates both the PID and MFSMC utilize similar power. Table 4 confirms the power consumed. It is calculated by integrating the sum of the square of the control efforts.

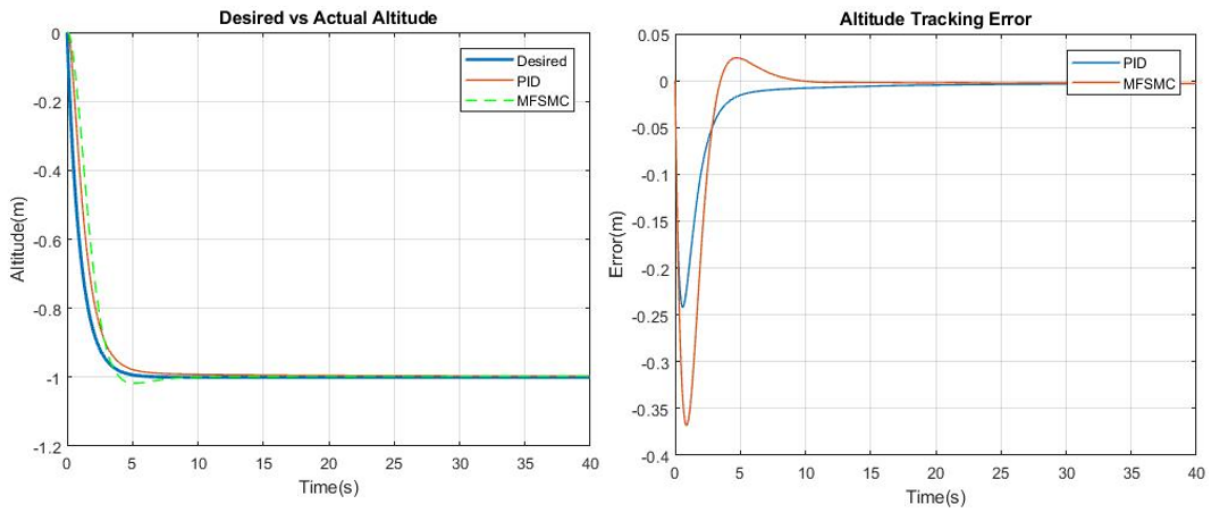


Figure 20: *Tracking and Tracking Error for Altitude (Z)*

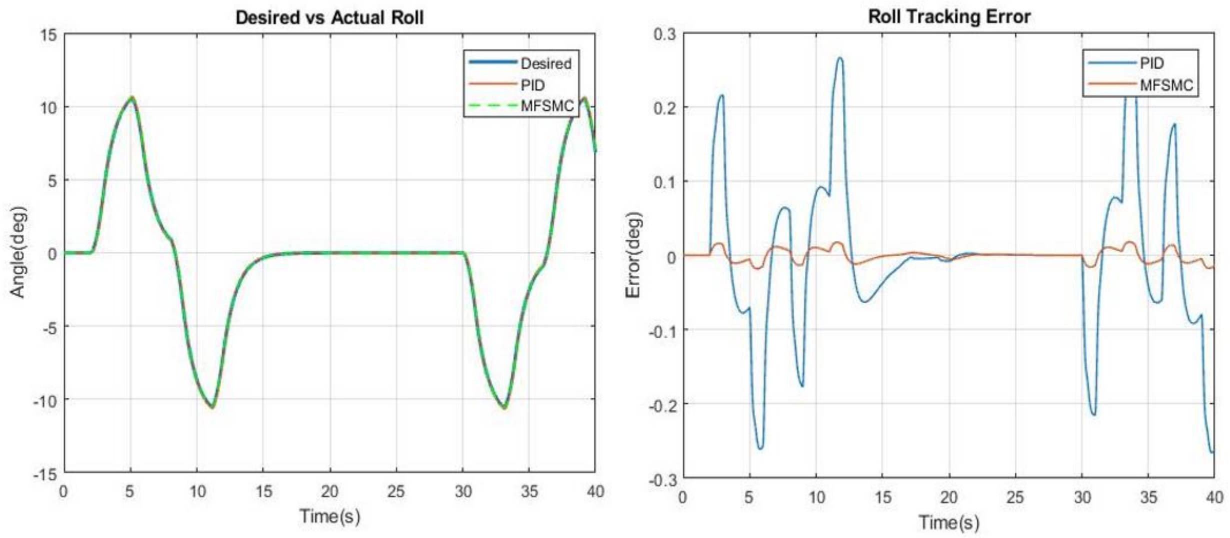


Figure 21: *Tracking and Tracking Error for Roll (Phi)*

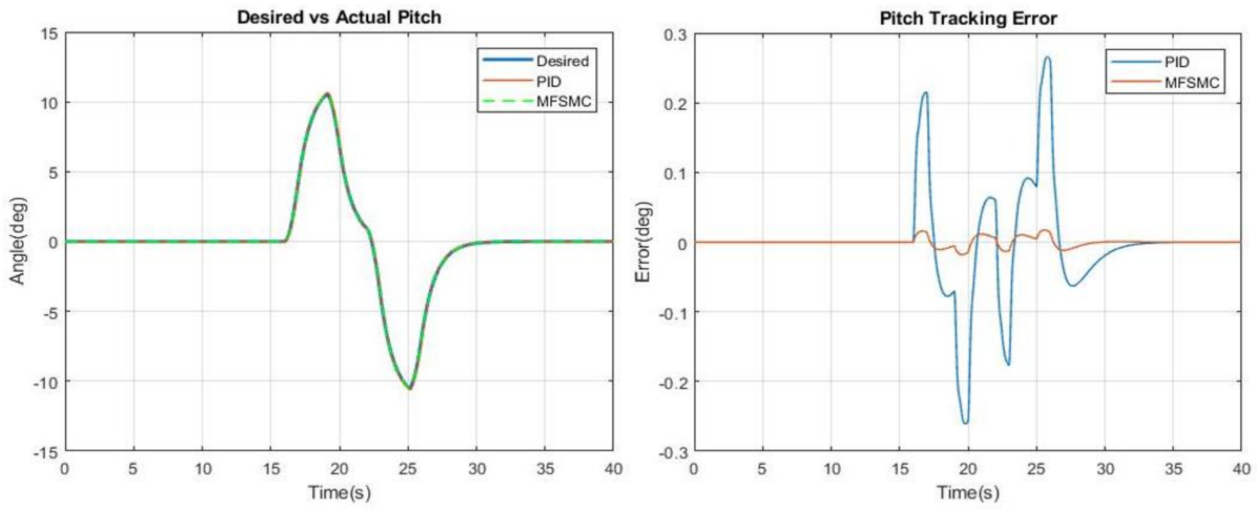


Figure 22: *Tracking and Tracking Error for Pitch (Theta)*

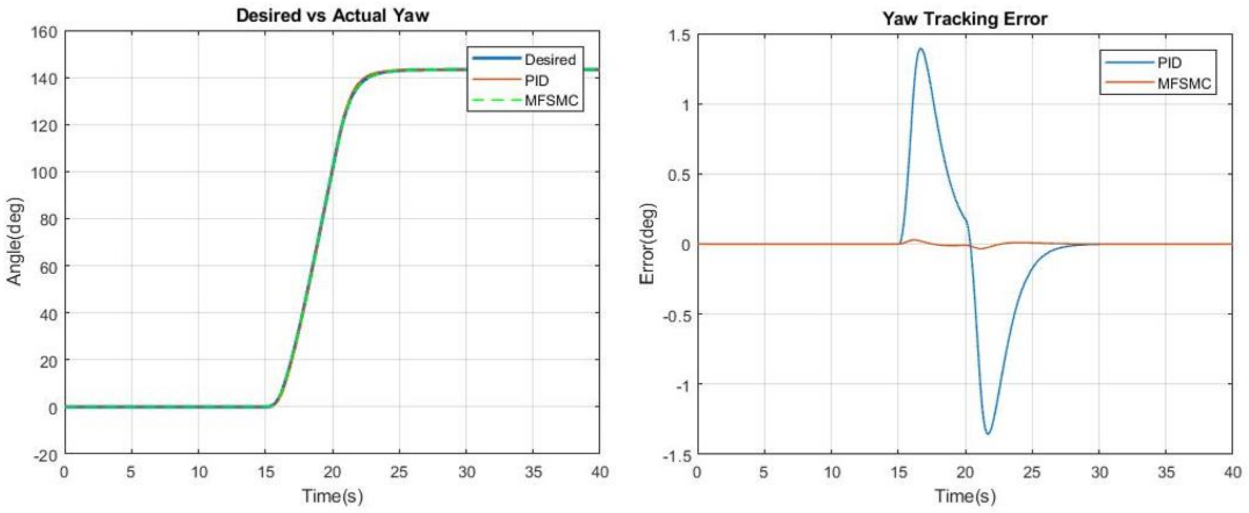


Figure 23: *Tracking and Tracking Error for Yaw (Psi)*

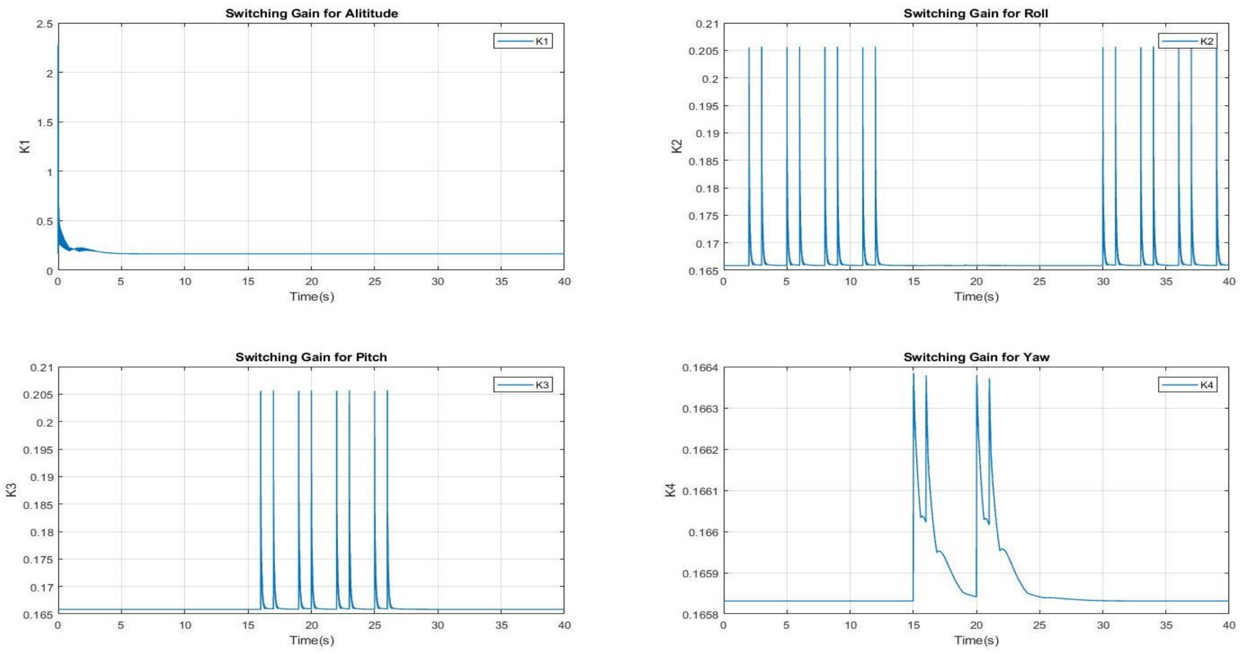


Figure 24: *Switching Gains for Altitude, Roll, Pitch, and Yaw Tracking*

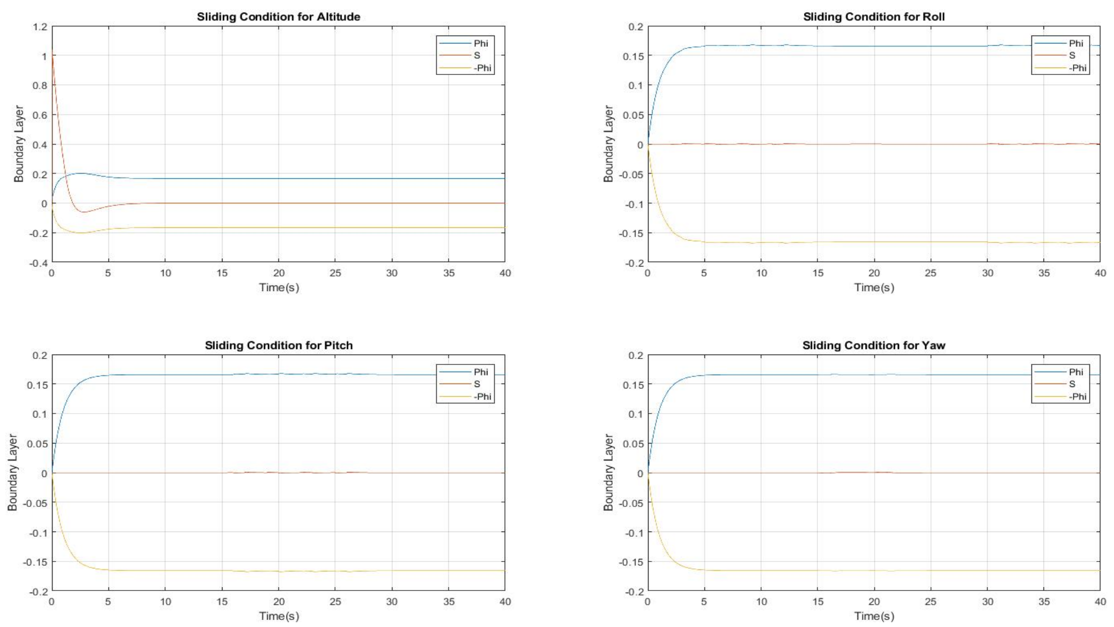


Figure 25: *Sliding Condition for Altitude, Roll, Pitch, and Yaw Tracking*

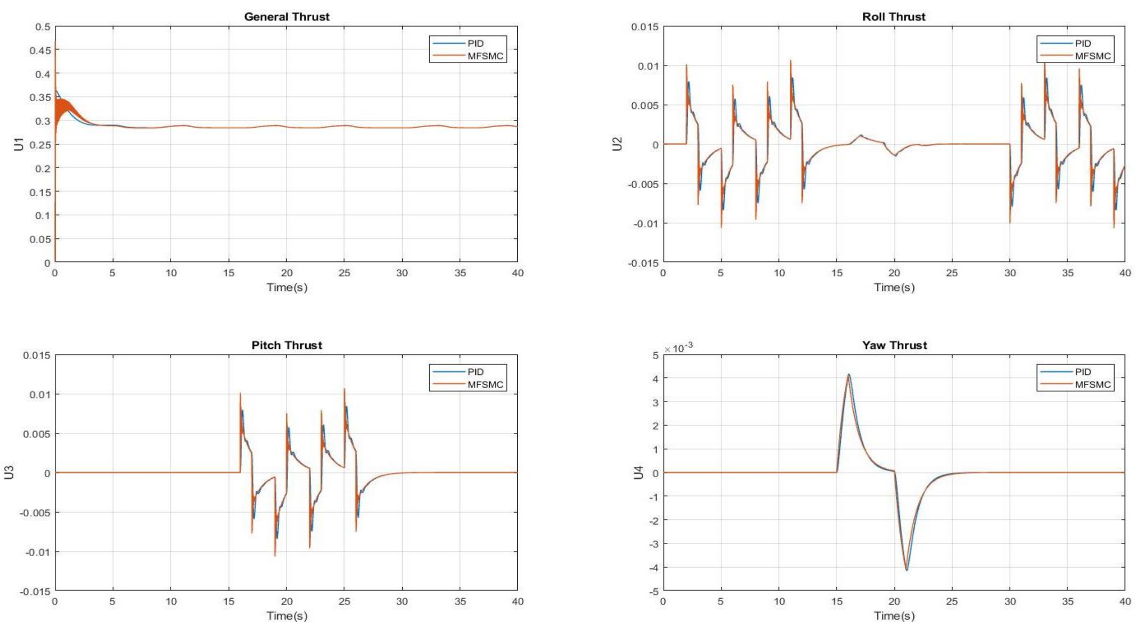


Figure 26: *Thrust Commands (U1 – U4)*

Table 3: *RMS Value for Basic Simulation*

Controller	**RMS**			
	Altitude (Z)(m)	Roll (Phi)(deg)	Pitch (Theta)(deg)	Yaw (Psi)(deg)
PID	0.0427	0.0998	0.0722	0.4091
MFSMC	0.0674	0.0081	0.0059	0.008

Table 4: *Power Consumption for Basic Simulation*

Controller	Power (W)
PID	3.3247
MFSMC	3.3235

Table 3 displays the RMS error values for each tracking responses for the PID and MFSMC control strategies. The results indicate the MFSMC has a better tracking performance compared to the PID for the roll, pitch and yaw tracking. The altitude tracking precision error is slightly higher than the PID controller due to the initial condition matching. This is resolved during hardware implementation. The overall results illustrate the effectiveness of the MFSMC controller for navigation and control applications with smooth signals and twice differentiable signals. Table 4 displays the total power consumed for the system under PID control and MFSMC. The MFSMC consumes slightly less power than the PID controller as expected due to the robust nature of the algorithm.

The previous simulation study was lacking since real-world effects were not considered in the simulation process. Therefore, a simulation study should be considered to replicate real-world effects such as hardware implementation, sensor error, etc. Before the updated simulation study is performed, the reference frame should be converted from the local NED to quadcopter body-reference frame since sensors (such as IMUs) are typically rigidly mounted on the quadcopter sensing acceleration and angular velocity with respect to the quadcopter. Details of the real-world effects to be modeled for the follow-on simulation study are described next.

5.4 Model Sampling Rate

The simulation study performed earlier was run at a fixed frequency, 10 KHz numerical integration and time update rate for both the model and the controller. The high time rate is not possible for implementation of the control laws on actual hardware. To more accurately represent hardware implementation in the simulation effort, the plant model was solved using a numerical variable step size scheme while the controller was run at a fixed sampling frequency of 1 KHz which is the update frequency of the proposed hardware controller (Zynq 7020 SOC [36]) used in hardware.

5.5 State Estimation

The main difference between the state estimation required for the PID control law and the MFSMC control law is the MFSMC algorithm requires full state feedback including absolute position, all Euler angles, and both the linear and angular velocities and accelerations. An IMU was used for state estimation of the required signals. The IMU consists of an accelerometer sensor which measures the three-axial accelerations and a gyroscope sensor measures the three rotational velocities providing direct measurements for the $\ddot{X}, \ddot{Y}, \ddot{Z}, \dot{\Phi}, \dot{\Theta}$, and $\dot{\psi}$ states. The sensors are placed closed to the center-of-mass of the quadcopter to provide the most accurate data for linear acceleration at the center-of-gravity location. However, the sensors contain errors such a measurement bias. The bias can initially be removed by initializing the sensors at a static straight-and-level condition. Ideally, the accelerometer should output zero acceleration along the X and Y directions and 1 g along the Z direction and the gyroscope should read zero rotational velocities in all three axes for the straight-and-level static condition. During initialization, sensor measurements are recorded to calculate the sensor biases which are subsequently subtracted from each of the corresponding sensor signals.

For the control approach presented in Section 5.2, the states $\dot{Z}$ and Z are estimated by direct discrete integration of the accelerometer signal. The Euler roll (Φ) and pitch (Θ) angles are estimated using a complimentary filter approach that will be discussed in the next subsection. The heading yaw (ψ) angle is estimated directly from the gyroscopic signal. The three angular accelerations are measured by differentiating the gyroscope signals using the sliding mode differentiator [37] which results in zero phase lag with minimal noise. Figure 27 displays the Simulink block diagram of the Sliding Mode differentiator.

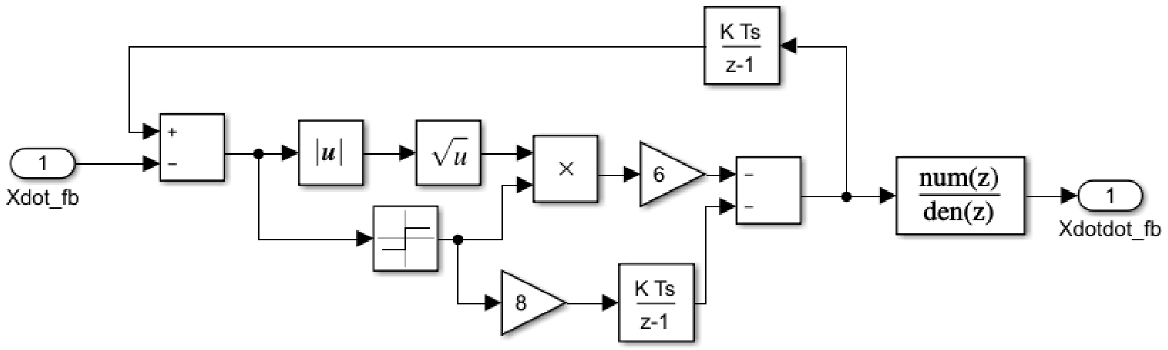


Figure 27: *Simulink Diagram of the Sliding Mode Differentiator*

It is important in the MFSMC algorithm that the “desired” signals to be tracked are smooth and twice differentiable. Hence the desired signals are passed through a second-order smoothing filter ensuring all signals of any kind will always remain smooth and twice differentiable. Figure 28 displays the second-order “smoothing” filter Simulink diagram shown in discrete form.

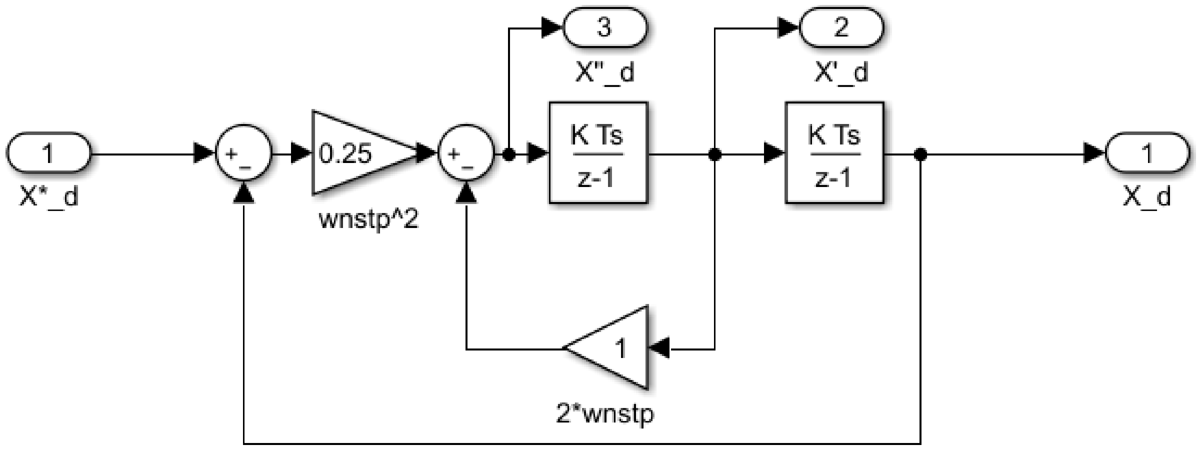


Figure 28: *Simulink Block for the Discrete Second-Order “Smoothing” Filter*

5.5.1 Complimentary Filter

A complimentary filter scheme can be used for state estimation of the roll and pitch angles as outlined in [36, 38, and 39]. The accelerometer can be used to obtain estimates of the pitch and roll angles (as does the gyroscope using a basic geometry) as shown below:

$$\Phi_A = tan^{-1}\left(\frac{\ddot{Y}}{\ddot{Z}}\right) \tag{52}$$

$$\theta_A = tan^{-1}\left(\frac{\ddot{X}}{\sqrt{\ddot{Y}^2 + \ddot{Z}^2}}\right) \tag{53}$$

The accelerations used in the above equations are referenced to the body-reference frame and the required coordinate transformation will be discussed in Section 5.7. The roll and pitch angles are calculated by direct integration of the gyroscope data using an Euler model. Since integration is performed over a period of time, the estimates drift over time due to sensor bias and becomes unreliable over a longer period of time. The pitch and roll estimates using accelerometer sensors does not involve integration, Eqs. (52) and (53), and hence the problem of estimate drift is eliminated and is reliable over long periods of time. But since the accelerometer senses all imposed forces, high frequency noise is present and in addition; the equations, Eqs. (52) and (53), used for estimating pitch and roll are not exact and are valid in only quasi-static conditions. Therefore, a combination of the accelerometer and the gyroscope sensor data is used for the estimation of roll and pitch Euler angles. The accelerometer data is passed through a low-pass filter to reduce the high frequency noise and the gyroscope data is passed through a high-pass filter to eliminate drift effects resulting in the complimentary filter scheme. A schematic of the complimentary filter is shown in Figure 29:

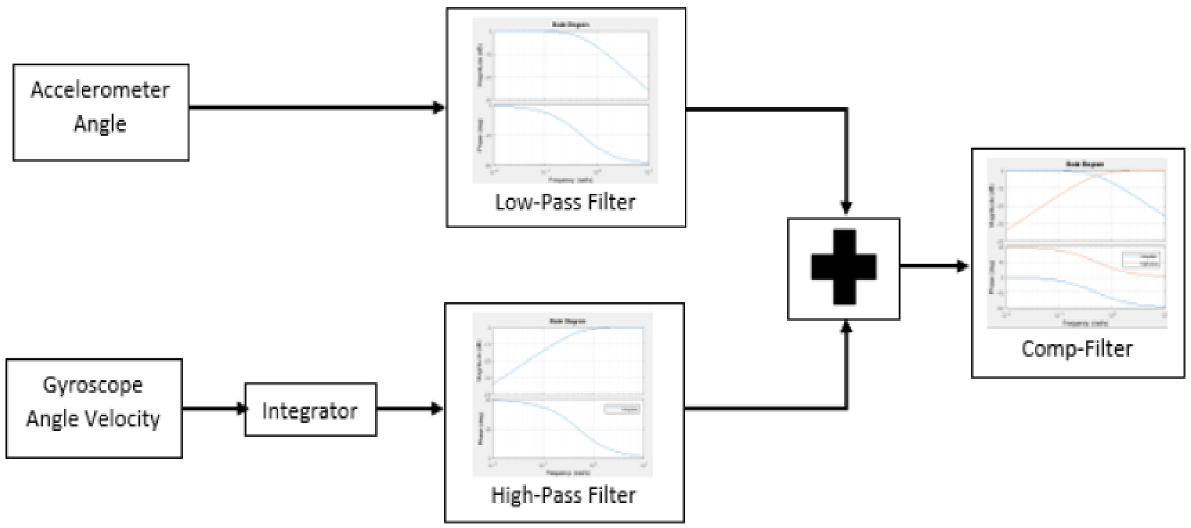


Figure 29: *Schematic of the Complimentary Filter Scheme* [36]

The complementary filter transfer functions for the low and high-pass filters are:

$$Low\ Pass\ Filter = \frac{1}{1+\tau s} \tag{54}$$

$$High\ Pass\ Filter = \frac{\tau s}{1+\tau s} \tag{55}$$

where, τ is the complementary filter cutoff frequency.

The final estimate for roll and pitch from the complimentary filter is given by the following:

$$\Phi = \gamma\Phi_{gyro} + (1-\gamma)\Phi_{accel} \tag{56}$$

$$\theta = \gamma\theta_{gyro} + (1-\gamma)\theta_{accel} \tag{57}$$

where $0.5 \leq \gamma \leq 1$.

The quality of the complimentary filter depends on the parameter γ. The filter has better quality as the value of γ is closer to the upper bound. Yaw angle cannot be inferred directly from the accelerometer and therefore will be directly integrated from the gyroscope data and the drift is deemed acceptable for this study. Using a magnetometer sensor for sensing heading can improve the quality of the yaw angle estimate for further follow-on studies.

5.6 Sensor Modeling

The sensors used in the IMU are a tri-axial accelerometer and a tri-axial gyroscope. Both the sensors are set to sample at the operating frequency of 1 KHz. The inputs to the sensor model are the three axial accelerations and three angular velocities from the plant. First, a transformation of the accelerations in the NED reference frame to the body-reference frame is performed. Gaussian noise is added to the sensor model for these signals after a noise estimation study is conducted. The raw data from the sensors were collected and noise parameters such as mean, variance and the peak-to peak-value of the noise was calculated. Initialization of sensors was performed to remove the bias introduced inherent to the accelerometer and gyroscope sensors. Figure 30 displays the time history responses at a static condition for each sensor signal and are used to quantify the sensor noise and bias characteristics.

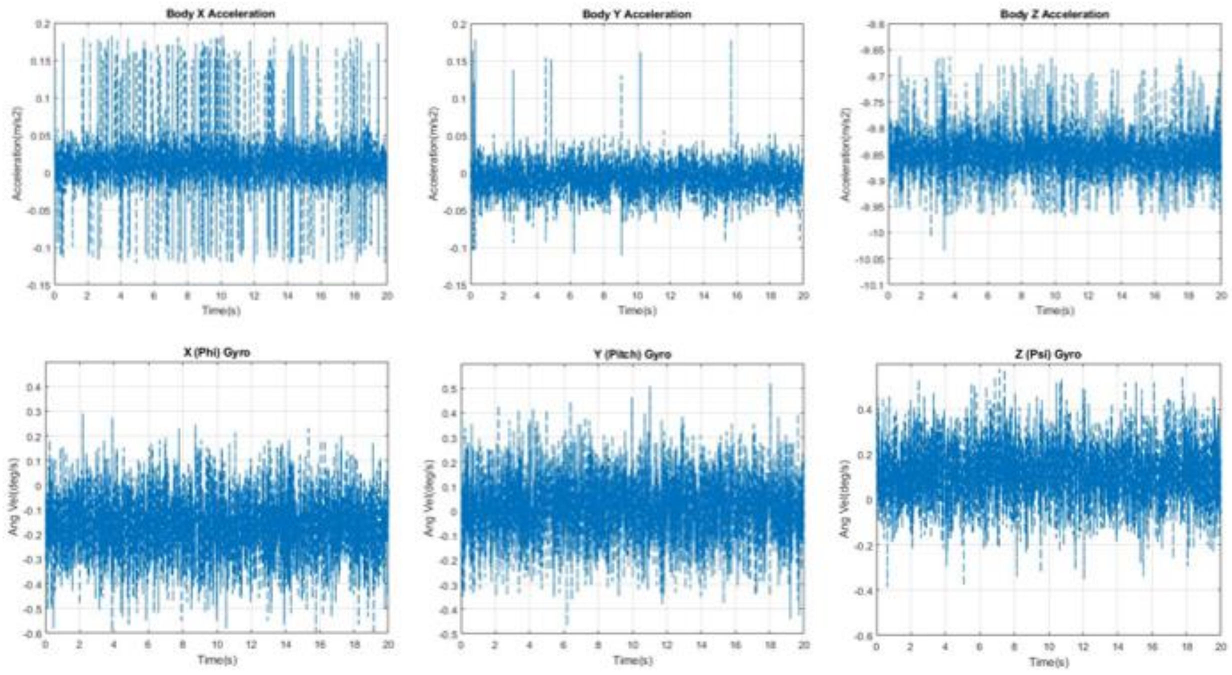


Figure 30: *Data Used for Noise Characteristic Estimation of the Accelerometer and Gyroscope*

Data used to calculate the mean values and V$_{pp}$ of the noise characteristics are shown in Figure 30. The standard deviation is calculated using the following:

$$6\,\sigma = V_{pp} \tag{58}$$

where σ is the noise variance. Table 5 summarizes the sensor noise characteristics.

Table 5: *Accelerometer and Gyroscope Noise Estimation Characteristics*

Parameters	X_Accel (m/s$^{2)}$	Y_Acce l(m/s^2)	Z_Accel (m/s^2)	X_Gyro (rad)	Y_Gyro (rad)	Z_Gyro (rad)
Mean	0.02	~0	-9.81	-0.004	~0	0.0017
V$_{pp}$	0.35	0.3	0.4	0.0157	0.0139	0.017
σ	0.0583	0.05	0.06	0.0026	0.0023	0.0029

Reis [3] amended the MFSMC algorithm to account for noise present in the sensors. He showed that arbitrary selected of the MFSMC parameters (λ and η) led to unacceptable tracking of the higher-order states along with excessive chattering of the control effort. In addition, the boundary layer sliding condition is not satisfied. Therefore, the MFSMC parameters are selected to account the noise of the sensors. The parameter selection is performed using the following.

$$\lambda \le \left(\frac{\eta}{V_{pp}}\right)^{\frac{1}{n}} \tag{59}$$

$$\eta_n = \eta + \left(\frac{\sigma_n}{\eta}\right)\left(\frac{\lambda}{2}\right) \tag{60}$$

where n is the order of the system, V_{pp} is the peak-to-peak of the noise value, σ_n is the noise standard deviation, and η_n is the updated η value. The maximum value of V_{pp} is taken from Table 4 for the worst case scenario. The updated parameters are used in the MFSMC law shown in Eq. (59) and Eq.(60) for quadcopter control. From the noise study the λ was calculated to be $\lambda \le 5$.

5.7 NED to Body Reference Frame

Since the motor thrust equations are referenced to body-reference frame and since the IMU is mounted on the quadcopter, a coordinate conversion from the local NED frame to the body-reference frame is required for replication of the quadcopter in the updated simulation study.

The gyroscope readings and the angular accelerations do not require a coordinate transformation as the rotation due to the earth is considered negligible and the hence there is no fixed rotational acceleration component acting on the quadcopter. However, the accelerometer has a constant component 1 g acting downwards and hence the conversion is required for the linear acceleration terms.

The accelerometers measure acceleration with respect to a free-falling reference frame. The acceleration has two parts: motional acceleration and gravitational acceleration. The force due to motional acceleration is:

$$F = ma \tag{61}$$

and force due to gravity is

$$F = mg \tag{62}$$

When an object is at rest at a static straight-and-level condition the accelerometer will read -1 g since a force is applied upwards on the object equal to mg. Therefore, even though the object is not moving, the accelerometer will sense an acceleration component. This component read by the accelerometer is referred to as gravitational acceleration.

In a complete free-fall condition (assuming no air resistance) the accelerometer should output a zero acceleration as the device is free falling with positive 1 g. The acceleration due to gravity can be thought of as a constant bias on the accelerometer acting in the upward direction. Assuming it is always present and negative (upward direction), it cancels the positive 1 g component when the object is in free-fall condition.

To model the dynamics of a quadcopter, the quadcopter motion is determined by the force due to gravity counteracted by the generated thrust. The thrust acts in body-reference frame and hence the force due to gravity needs to be transformed into the body-reference frame. The Newtonian laws are applied to the 6 DOF rigid-body to resolve the rotational and linear accelerations. The rotational velocities are the outputs of the gyroscope. The linear accelerations need to be transformed into the local NED frame to obtain the correct X, Y and Z vector. The linear accelerations need to be biased by 1 g and converted into the body-reference frame that is assumed to be the output of the accelerometer. The following transformation matrix is used for the conversion of the respected reference frames:

$$R = \begin{bmatrix} \cos\psi\cos\theta & \sin\psi\cos\Phi + \cos\psi\sin\theta\sin\Phi & \sin\psi\sin\Phi - \cos\psi\sin\theta\cos\Phi \\ -\sin\psi\cos\theta & \cos\psi\cos\Phi - \sin\theta\sin\psi\sin\Phi & \cos\psi\sin\Phi + \sin\theta\sin\psi\cos\Phi \\ \sin\theta & -\cos\theta\sin\Phi & \cos\theta\cos\Phi \end{bmatrix} \tag{63}$$

The local NED reference frame and body-reference frame are related by the following equation:

$$\begin{bmatrix} \ddot{X}_{NED} \\ \ddot{Y}_{NED} \\ \ddot{Z}_{NED} \end{bmatrix} = R \begin{bmatrix} \ddot{X}_{body} \\ \ddot{Y}_{body} \\ \ddot{Z}_{body} \end{bmatrix} \tag{64}$$

$$R^{-1} \begin{bmatrix} \ddot{X}_{NED} \\ \ddot{Y}_{NED} \\ \ddot{Z}_{NED} \end{bmatrix} = \begin{bmatrix} \ddot{X}_{body} \\ \ddot{Y}_{body} \\ \ddot{Z}_{body} \end{bmatrix} \tag{65}$$

A block diagram of the conversion is shown in Figure 31.

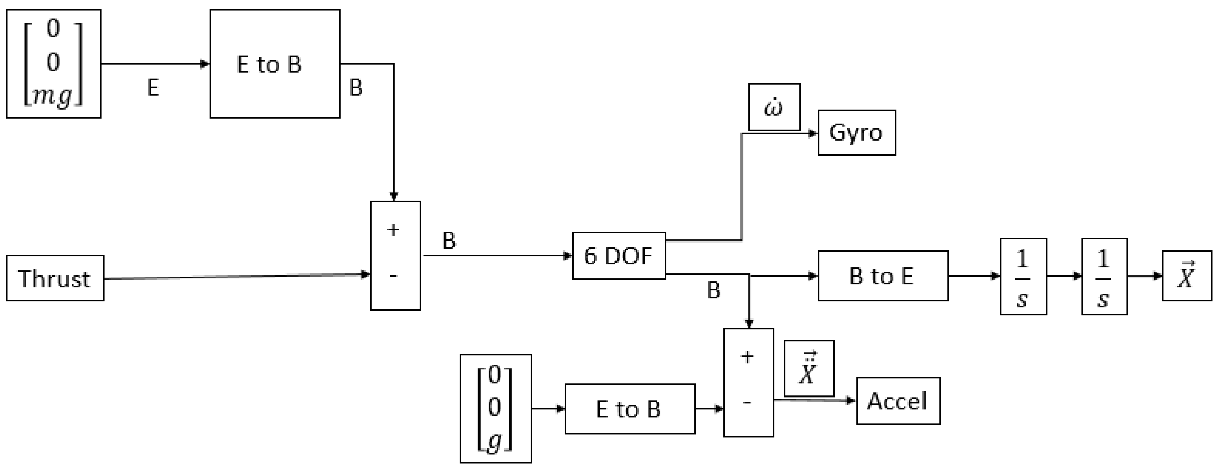


Figure 31: *Block Diagram for Coordinate Frame Conversion*

5.8 Hardware Simulations

After considering all hardware and sensor restrictions mentioned in the above sections, two simulation studies were conducted. The first simulation study performed assumed the quadcopter was mounted on a gimballed platform followed by a full-motion model simulation study. To prevent any damage to hardware and the surroundings, the quadcopter was mounted on a gimbal platform and was restricted to rotate only about the *X* and *Y* axes. The "gimbal" simulation study was conducted as the first hardware implementation and was used to control only roll and pitch.

5.8.1 Gimbal Simulation

A simulation study was performed to evaluate the control effectiveness and performance in controlling the pitch and roll states. The MFSMC law and the PID controller were used and their performance was compared for tracking precision. A constant thrust value of 0.4 was used. The MFSMC parameters λ, η and σ_u were set to 1 rad/sec, 0.1587 and 0.5, respectively, from to the noise study conducted in section 5.6. To be conservative the higher values were used.

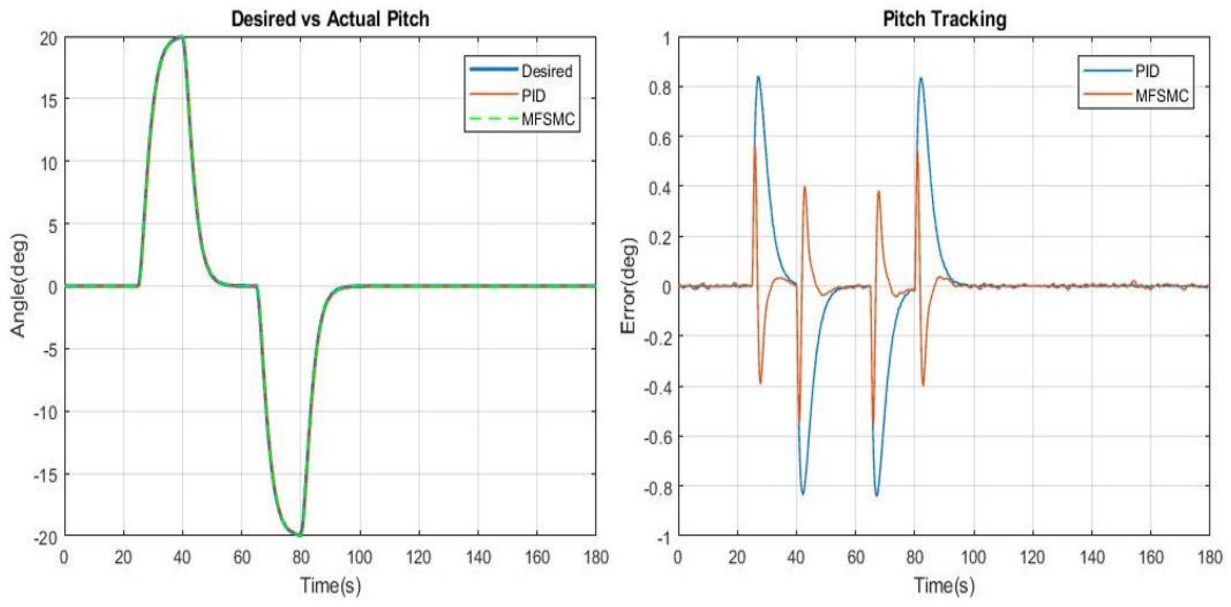


Figure 32: *Pitch Tracking and Tracking Error for the System Under PID Control and MFSMC*

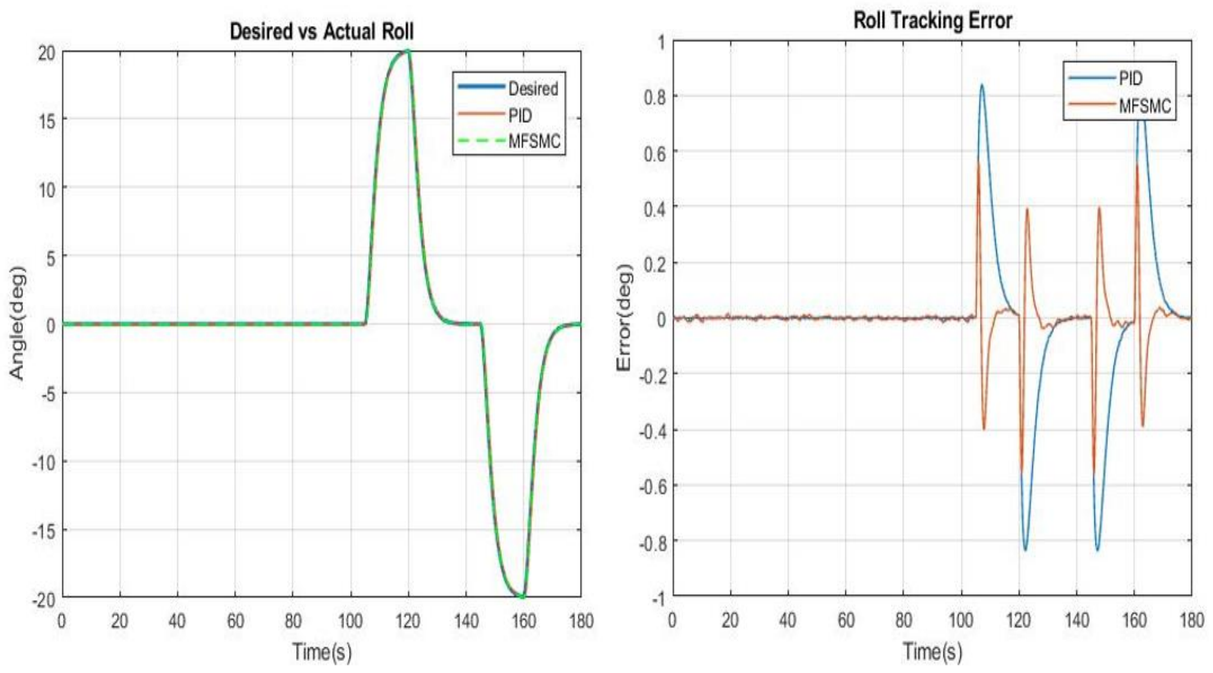


Figure 33: *Roll Tracking and Tracking Error for the System Under PID Control and MFSMC*

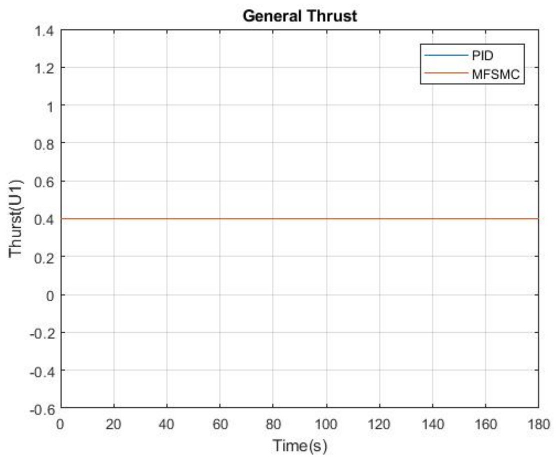


Figure 34: *General Thrust (U1) for PID and MFSMC Controllers*

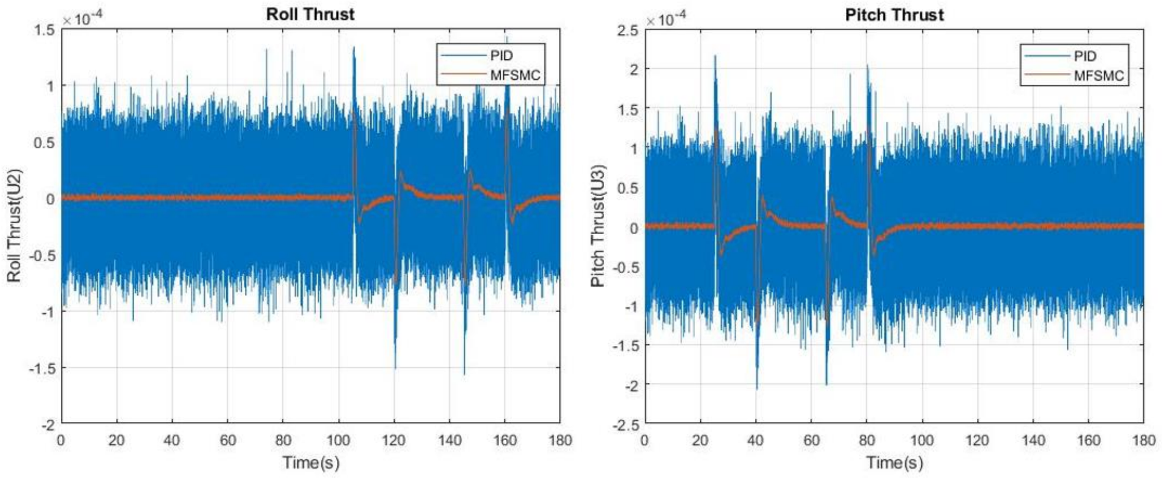


Figure 35: *Roll and Pitch Thrust (U2 – U3) for PID and MFSMC Controllers*

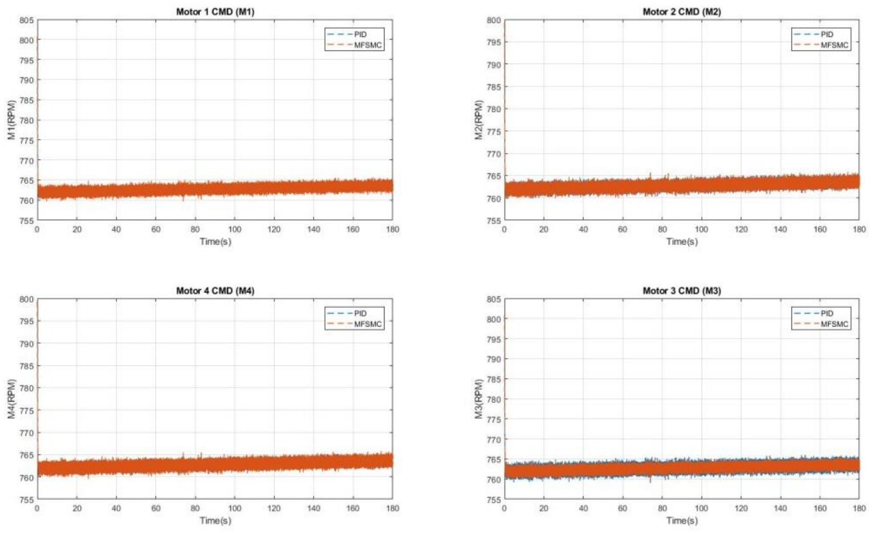


Figure 36: *Motor Commands (RPM) for PID and MFSMC Controllers*

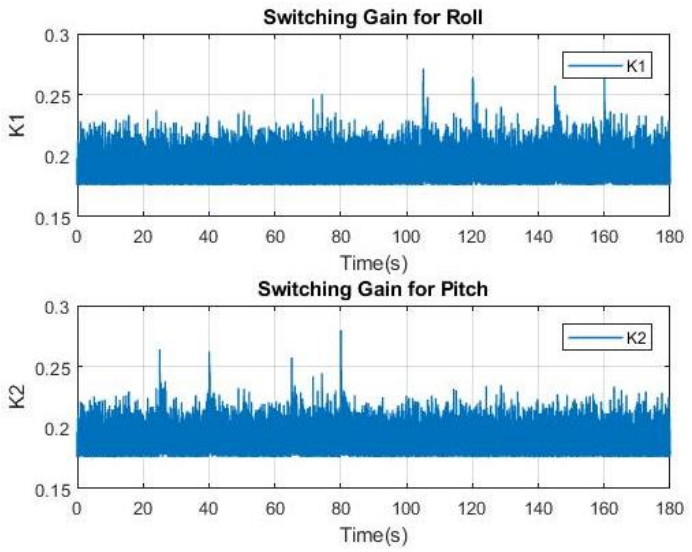


Figure 37: *Switching Gains for the Roll and Pitch Tracking*

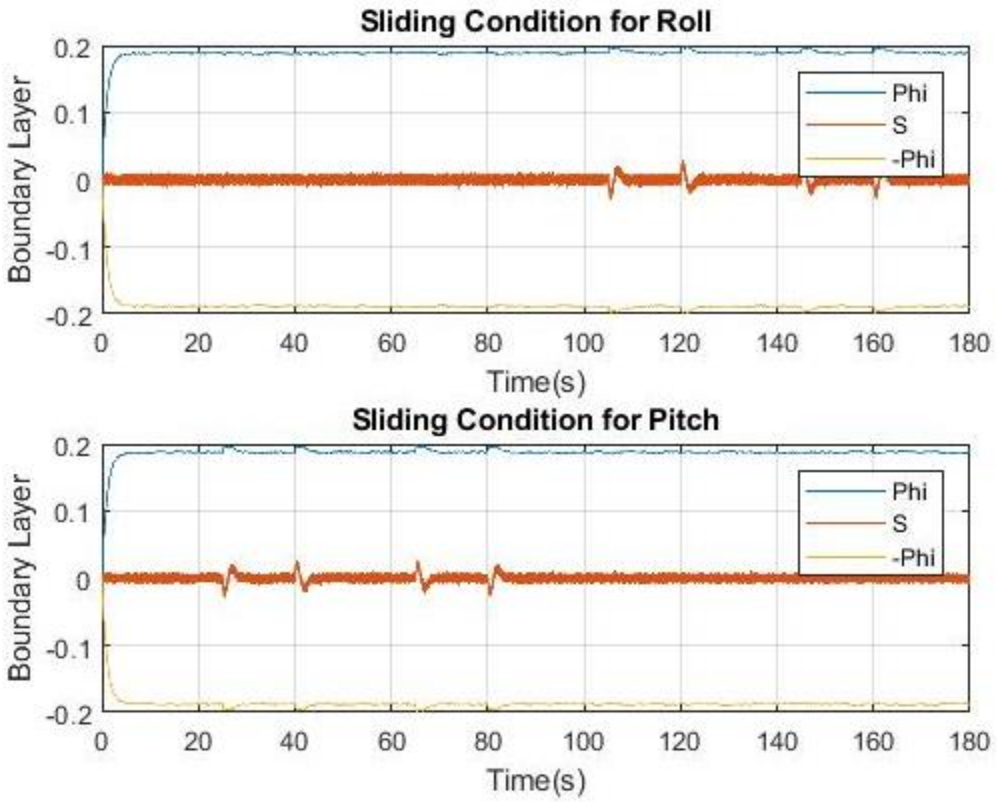


Figure 38: *Sliding Condition for the Roll and Pitch Tracking*

Figures 32 – 33 display the time history tracking responses for the roll and pitch angles for a given desired tracking response. The simulations results, as observed in Figures 32 – 33, show outstanding tracking performance for the system under PID control and MFSMC. Figures 34 – 35 display the control effort responses (*U1 – U3*) and Figure 36 displays the motor speeds in Revolutions-Per-Minute (RPM). As observed in Figures 34 – 36, both the PID and MFSMC control algorithm are given a 0.4 thrust with the observed control efforts *U2* and *U3* are much lower for the MFSMC law as compared to the PID controller. But the magnitude of *U2* and *U3* are very small as compared to *U1*. Hence the motors speeds remain the same for both the controllers and so does the power utilization as confirmed in Table 7. Figure 37 displays the switching gains for the MFSMC law and are both positive as expected for the MFSMC law. Figure 38 displays the sliding condition for the roll and pitch tracking under MFSMC and proves asymptotic tracking stability is maintained since the sliding condition is satisfied at all times.

Table 6: *RMS Value for Gimbal Simulation*

Controller	**RMS**	
	Roll (deg)	Theta (deg)
PID	0.2395	0.2395
MFSMC	0.1031	0.1025

Table 7: *Power Consumption for Gimbal Simulation*

Controller	**Power (W)**
PID	28.8
MFSMC	28.8

Table 6 displays the RMS error values for each tracking responses for the PID and MFSMC control strategies. The results indicate the MFSMC has a higher tracking precision compared to the PID for the roll, and pitch tracking. Observation of Figures 32 – 33 support this result. Table 7 displays the power required for each control law. The MFSMC requires similar power as the PID controller however, tracking precision performance is increased with the system under MFSMC compared to PID control as shown in Table 6.

5.8.3 Full Model Simulation

After the “gimbal” model simulation was completed, a simulation effort was conducted to test the effectiveness of the controllers in controlling the altitude and yaw axis states as well. The MFSMC algorithm parameters were kept the same as the “gimbal” model simulation. For this simulation study the desired yaw tracking was set to zero.

Figure 39 – 42 displays the tracking time history responses for the system under PID control and MFSMC. Outstanding agreement is observed for tracking control of altitude, roll, pitch, and yaw. Figure 43 – 44 and Table 9 shows similar power consumption between both PID and MFSMC algorithm as seen in “gimbal” simulations earlier.

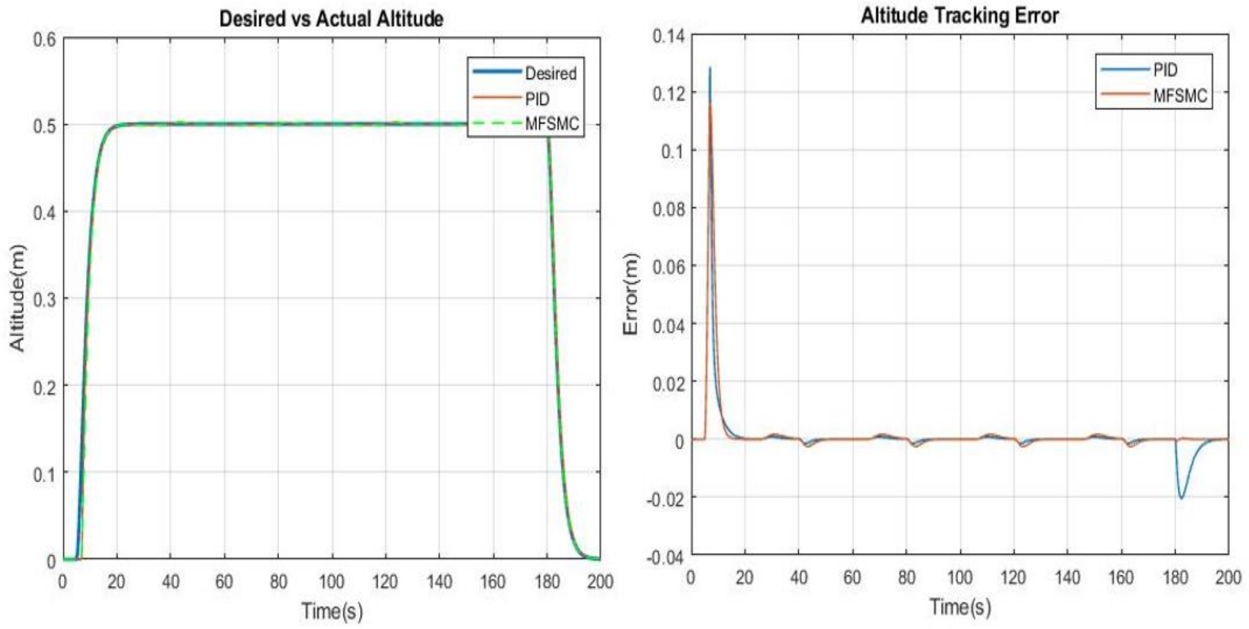


Figure 39: *Z (Altitude) Tracking and Tracking Error for PID and MFSMC*

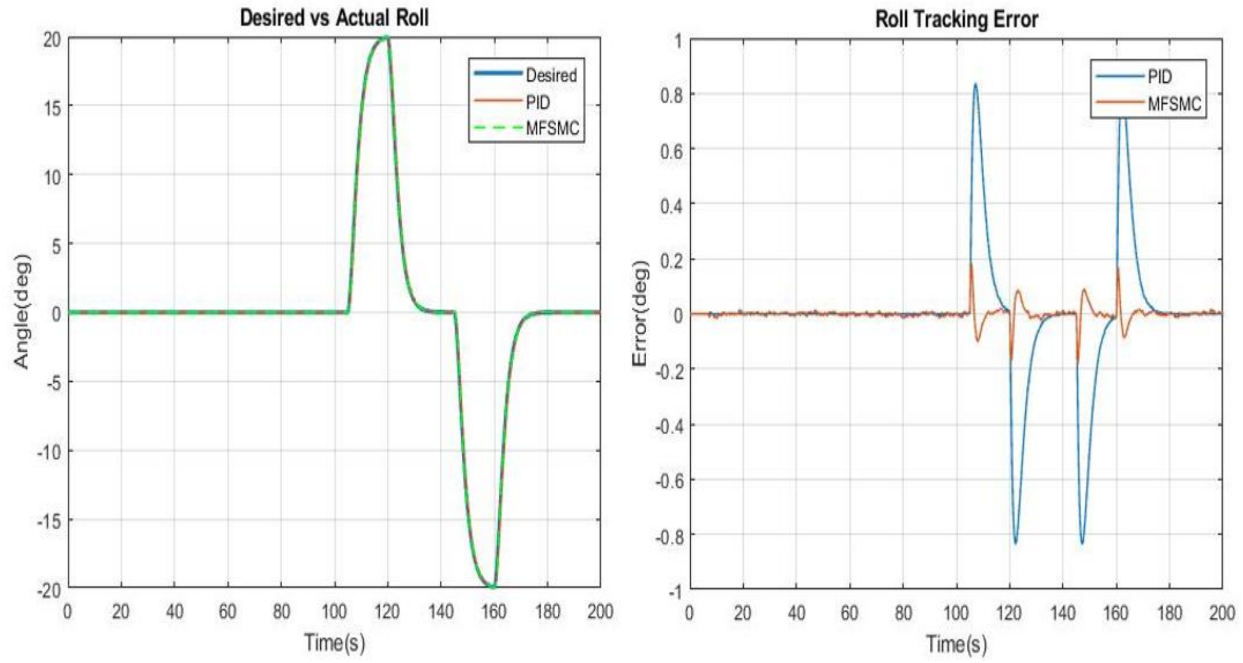


Figure 40: *Roll Tracking and Tracking Error for PID and MFSMC*

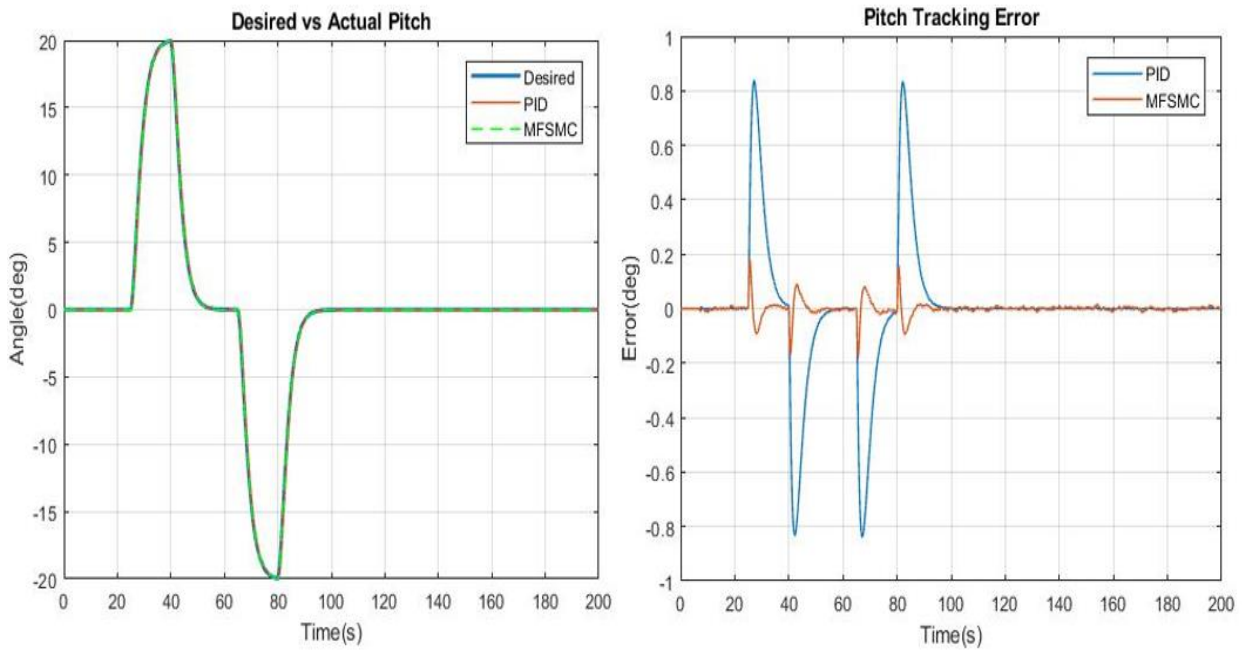


Figure 41: *Pitch Tracking and Tracking Error for PID and MFSMC*

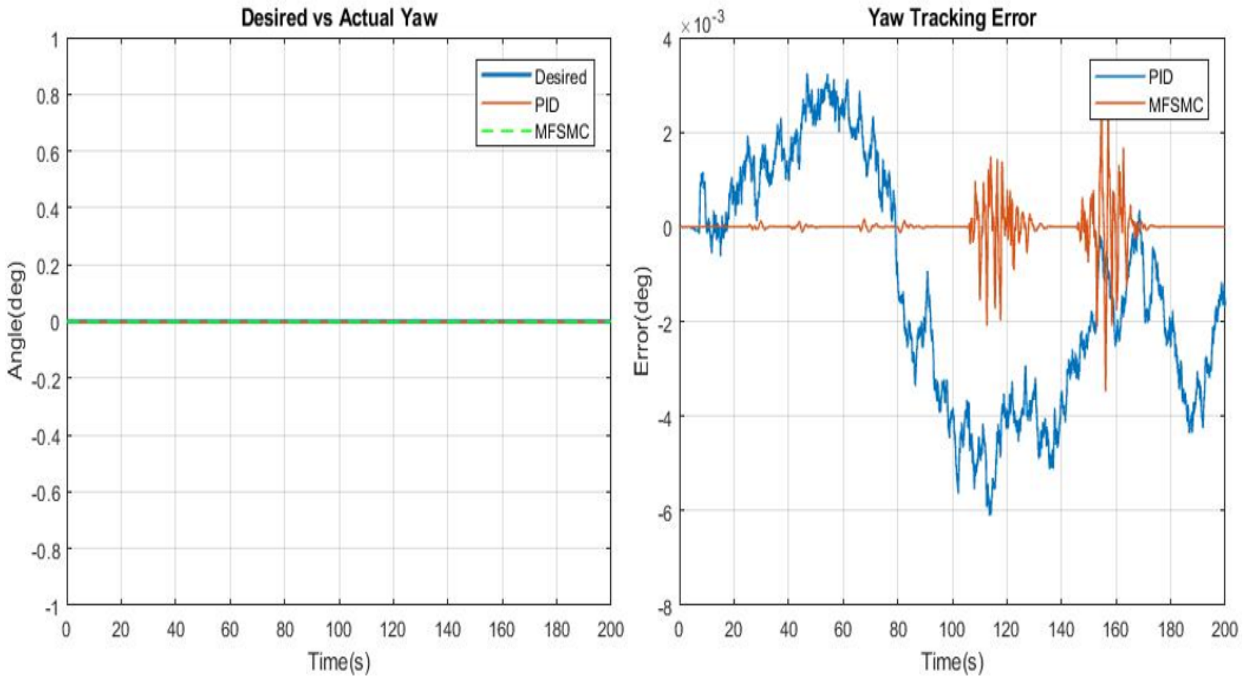


Figure 42: *Yaw Tracking for PID and MFSMC*

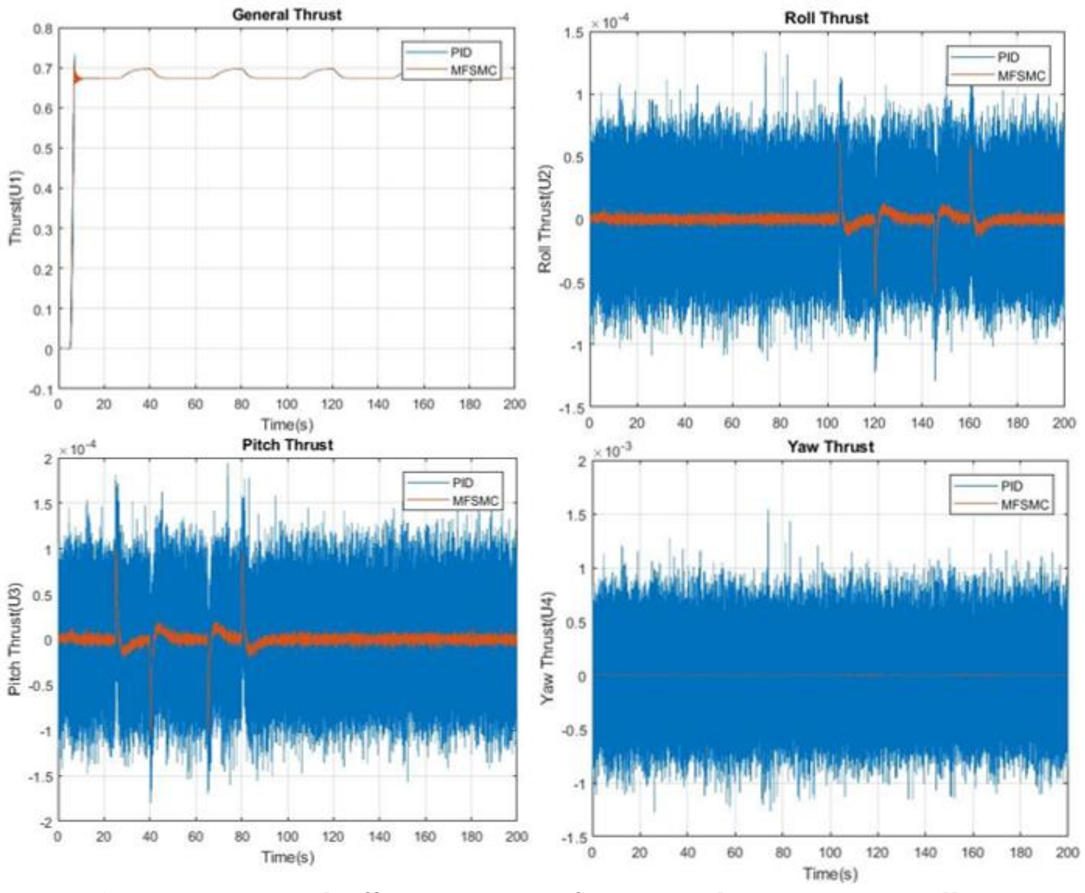


Figure 43: *Control Efforts (U1 – U4) for PID and MFSMC Controllers*

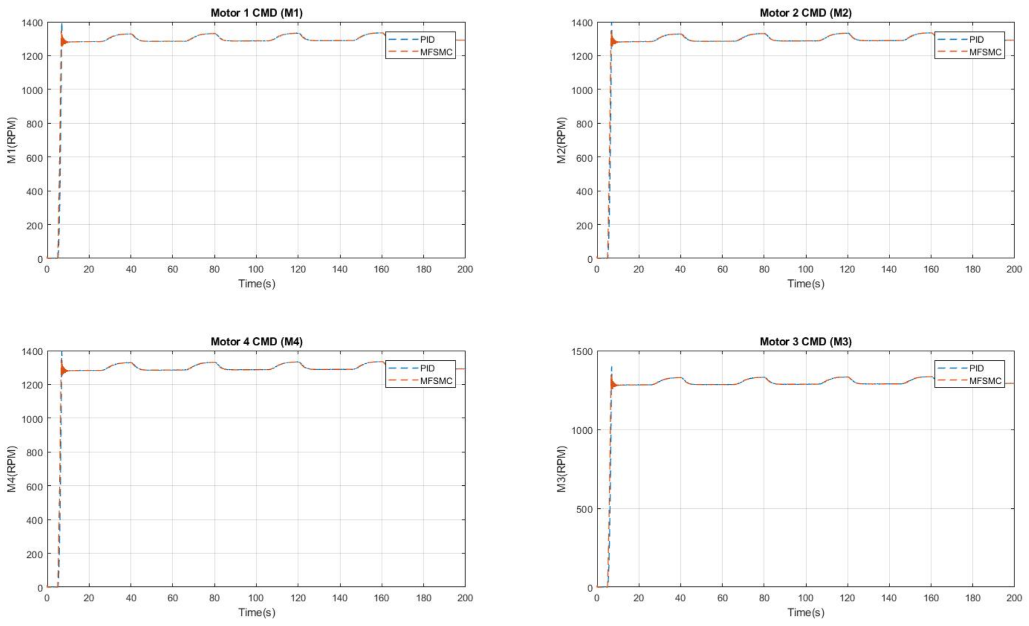


Figure 44: *Motor Commands (RPM) for PID and MFSMC Controllers*

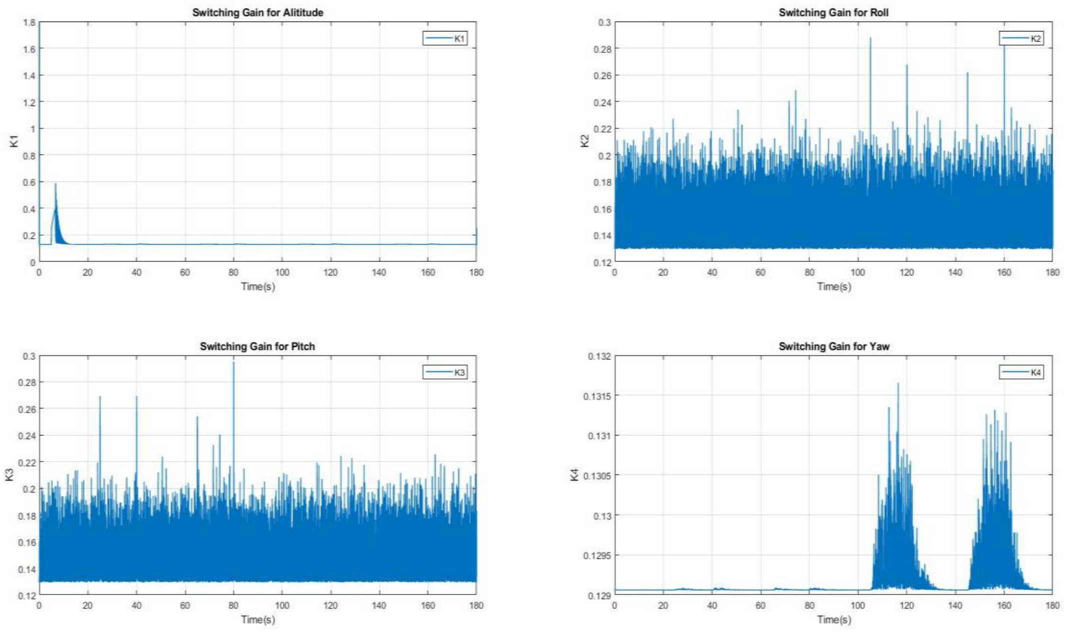


Figure 45: *Switching Gain for Altitude, Roll, Pitch, and Yaw Tracking*

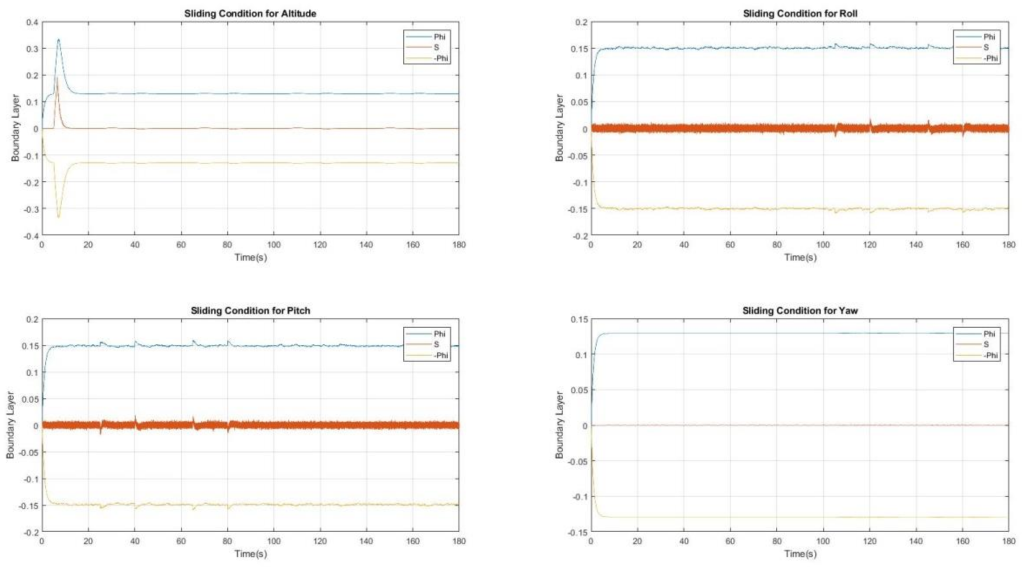


Figure 46: *Sliding Condition for Altitude, Roll, Pitch and Yaw Tracking*

Table 8: *RMS Value for Full Model Simulation*

Controller	**RMS**			
	Altitude (Z)(m)	Roll (Phi)(deg)	Pitch (Theta)(deg)	Yaw (Psi)(deg)
PID	0.0097	0.2268	0.2268	~0
MFSMC	0.0114	0.0279	0.0275	~0

Table 9: *Power Consumption for Full Model Simulation*

Controller	**Power (W)**
PID	89.41
MFSMC	89.36

Figure 45 displays the switching gains and are all positive as expected. Figure 46 displays the sliding condition for the MFSMC law and proves asymptotic tracking stability is maintained since the sliding condition is satisfied at all times. The MFSMC algorithm has a superior tracking performance for roll and pitch as shown in Table 8 (RMS of tracking error) and is verified by observation of Figures 40 – 42. Table 9 displays the total power required for the maneuver for each control law and shows that the system under MFSMC is more efficient compared to the PID controller. Also, the altitude tracking precision is also outstanding but slightly lower than the system under PID control. This result could be attributed to the under-actuation nature of the quadcopter. The updated simulation study confirms that the MFSMC algorithm can be implemented real world quadcopter for testing purposes as described next.

6. HARDWARE TESTING

In this chapter, the results of the quadcopter hardware testing effort under both PID and MFSMC control are presented. The first section presents a summary of the required hardware used and the individual components. In the second section, the performance of the PID and MFSMC controllers on the quadcopter for motion tracking is presented and discussed.

6.1 Hardware Description

The Fusion 1 quadcopter (Figure 47) from Craft Drones (http://www.craftdrones.com/) was used for the experimental testing. The Fusion 1 uses the Snickerdoodle SOM and the Zynq 7020 System-on-Chip (SoC) board. It comprises of a Dual-core ARM Cortex – AP MPCore^{TM} with Coresight^{TM} with a clock speed of 667 MHz. It also houses an Artix-7 FPGA with 85k programmable logic cells. The quadcopter has four EMAX 15 AMP ESCs with a burst current of 25A. Four EMAX 4500 KV 1106 brushless motors with a max speed of 36000 RPM are used for drive control of the quadcopter rotor. The Spektrum receiver and transmitter (Figure 48) are used to manually control the quadcopter. The MPU9250 IMU sensor is used for sensing of the system states and state estimation. The IMU is comprised of a 3-axis gyroscope, a 3-axis accelerometer and an onboard Digital Motion Processor^{TM} capable of processing complex algorithms. An ADS1015 analog-to-digital converter is also housed in the quadcopter which provides a 12-bit precision bus voltage at 1000 samples/second. The Turnigy 1000 MAh 2S battery is used to power the quadcopter. Finally the Fusion 1 drone incorporates a JTAG debugging header. Figure 49 displays the quadcopter mounted on the gimballed platform. The platform restricts vertical motion of the quadcopter and therefore, is ideal for initial hardware testing of the system preventing damage that may be encountered during full flight testing.

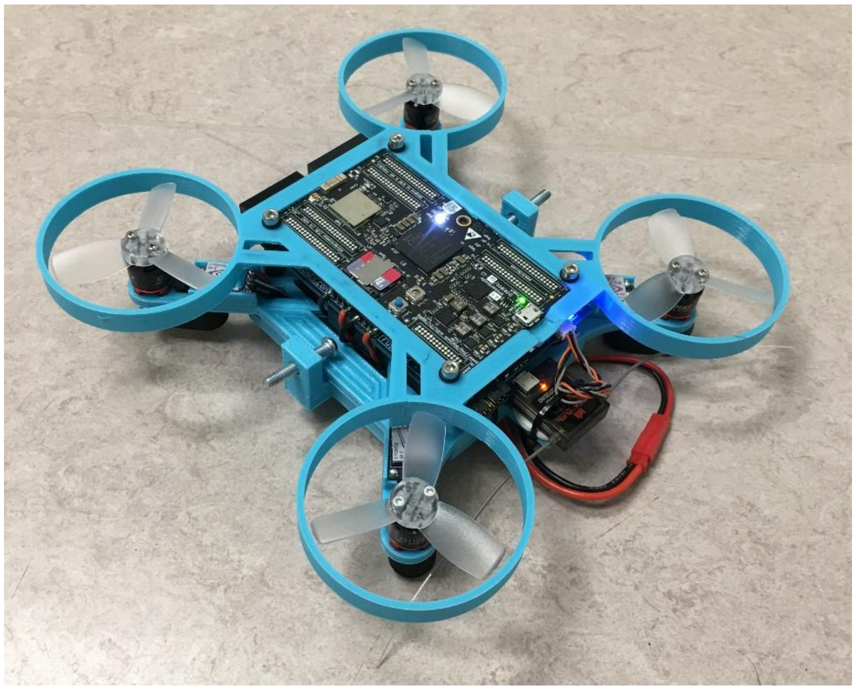


Figure 47: *Fusion 1 Quadcopter (from Craft Drones)*

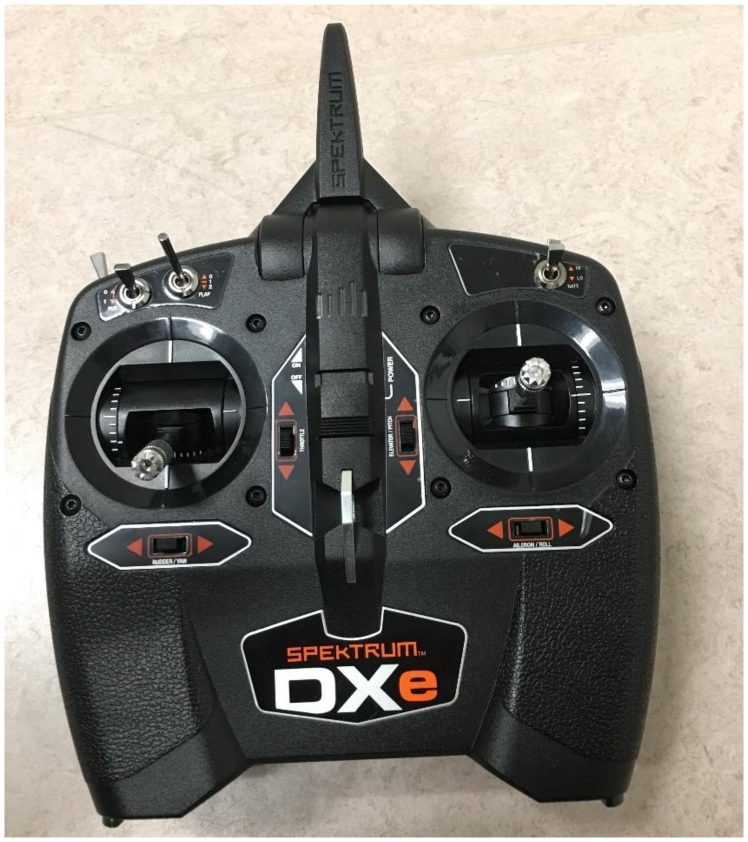


Figure 48: *Spektrum Transmitter*

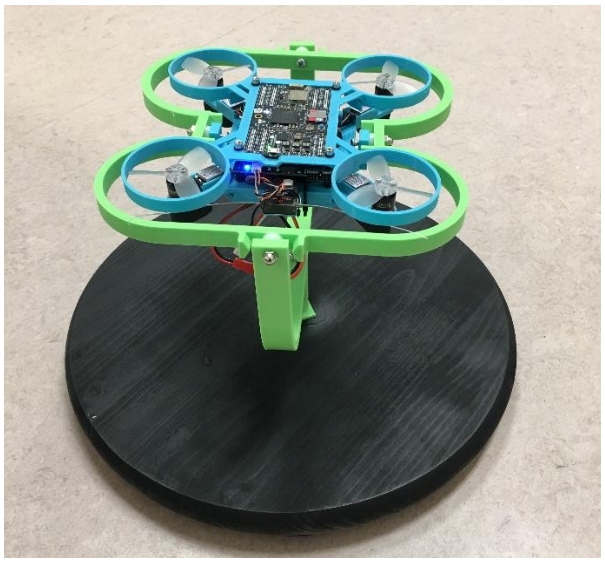


Figure 49: *Gimballed Fusion 1 Quadcopter*

6.2 Gimbal Hardware Testing & Results

The PID and MFSMC algorithm was used on the quadcopter described above for motion control of the roll (Φ) and pitch angles (Θ) on a gimbal platform. The MFSMC parameters were set to and value of λ = 4 rad/sec and η = 0.1848, used in the hardware testing.

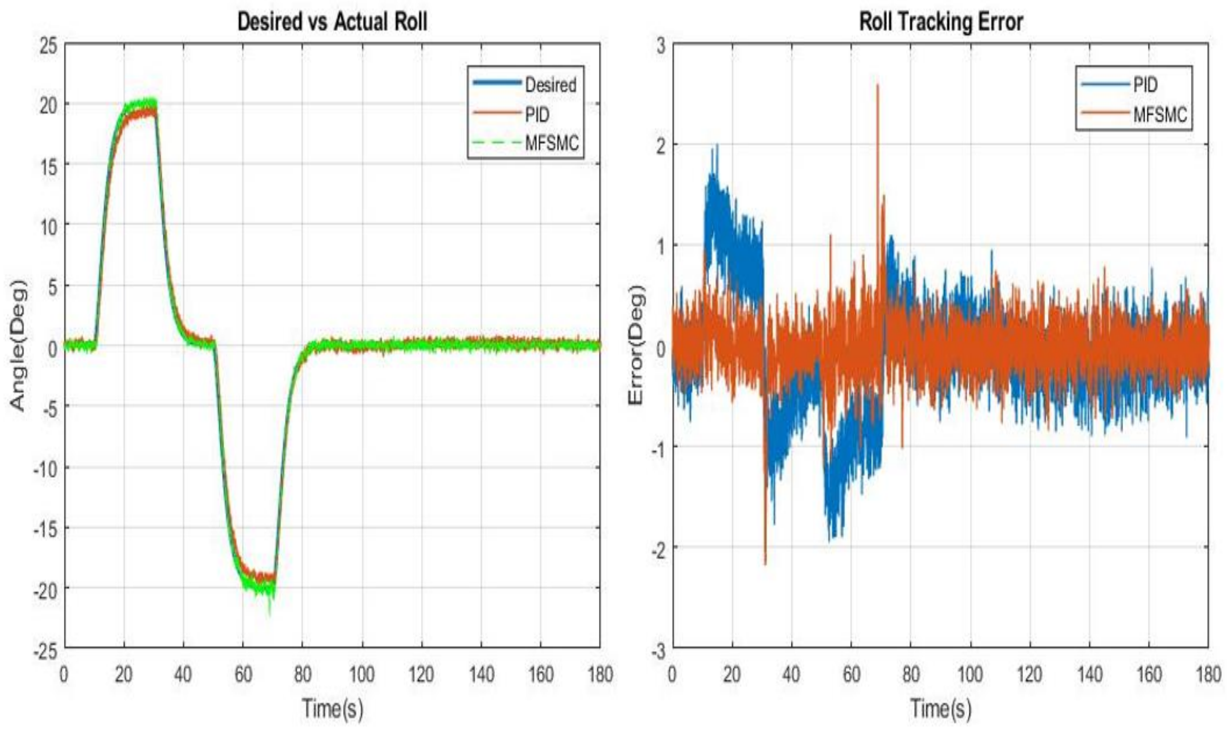


Figure 50: *Quadcopter Experimental Roll Tracking and Tracking Error Responses*

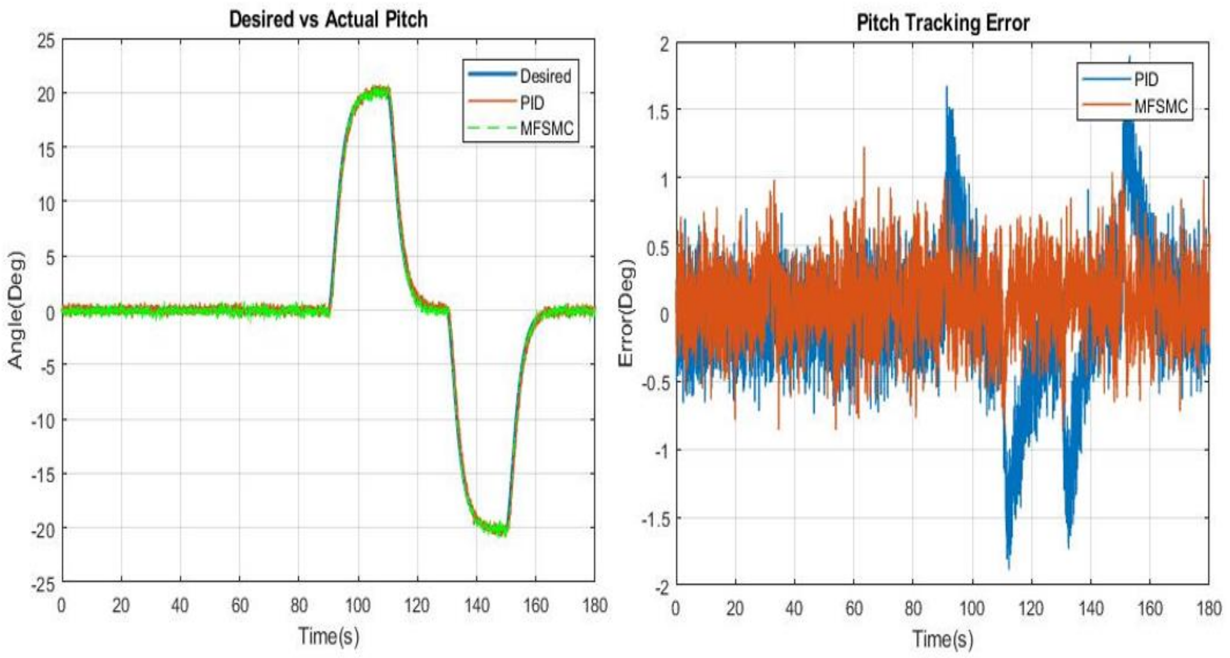


Figure 51: *Quadcopter Experimental Pitch Tracking and Tracking Error Responses*

Table 10: *RMS Value for Hardware Tracking Test*

Controller	**RMS**	
	Roll (deg)	Theta (deg)
PID	0.56	1.12
MFSMC	0.25	0.26

Table 11: *Power Consumption for Hardware Tracking Test*

Controller	**Power (W)**
PID	13.61
MFSMC	12.10

Figures 50 – 51 display the tracking time history responses for the system under PID control and MFSMC. As shown in Table 10 and by the tracking error plots in Figure 50 – 51 the MFSMC algorithm exhibits superior tracking performance compared to the PID controller and is more efficient in terms of power required as shown in Table 11.

Figures 52 – 54 display the control efforts (thrust commands) and the motor speeds (RPM) for both the PID and MFSMC algorithm. Figure 55 displays the motor speed (RPM) difference from the mean speed of the respective motor. The motor actuation commands are calculated by dividing the motor speeds by 2000. The power is then calculated by integrating the motor actuation commands. The MFSMC algorithm requires ~half the general thrust as compared to the PID controller for optimal tracking performance. Therefore, the MFSMC operates at a lower motor speed (RPM) as compared to the PID controller consuming less power. Table 11 confirms significantly lower power consumption by the MFSMC controller. Further studies need to be conducted to prove this with certainty.

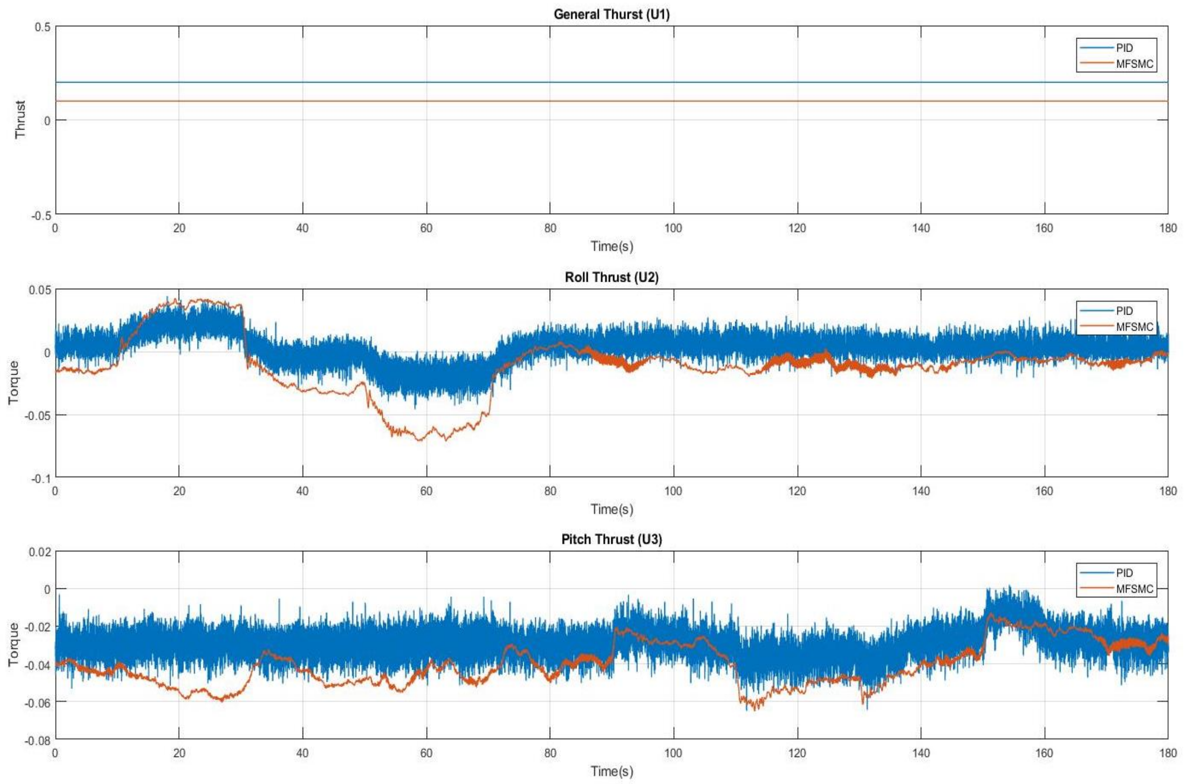


Figure 52: Thrust Commands from Experimental Test

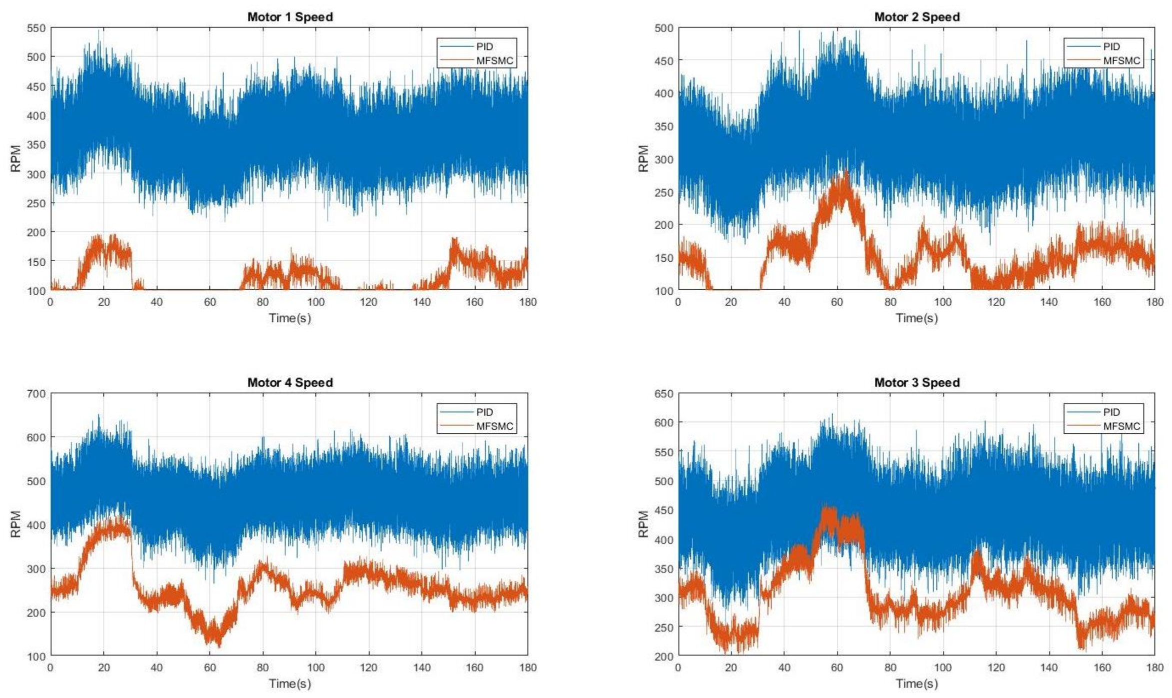


Figure 53: *Motor Speeds from Experimental Test*

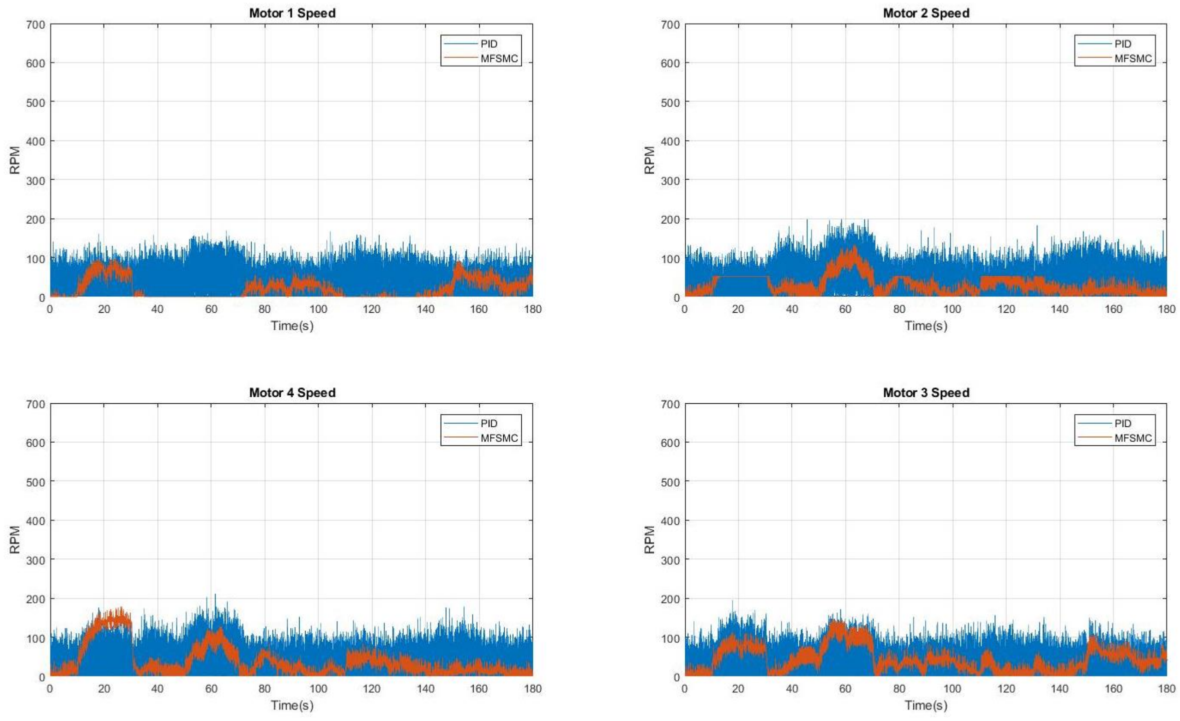


Figure 54*: Motor Speed Difference (mean subtracted) from Experimental Test*

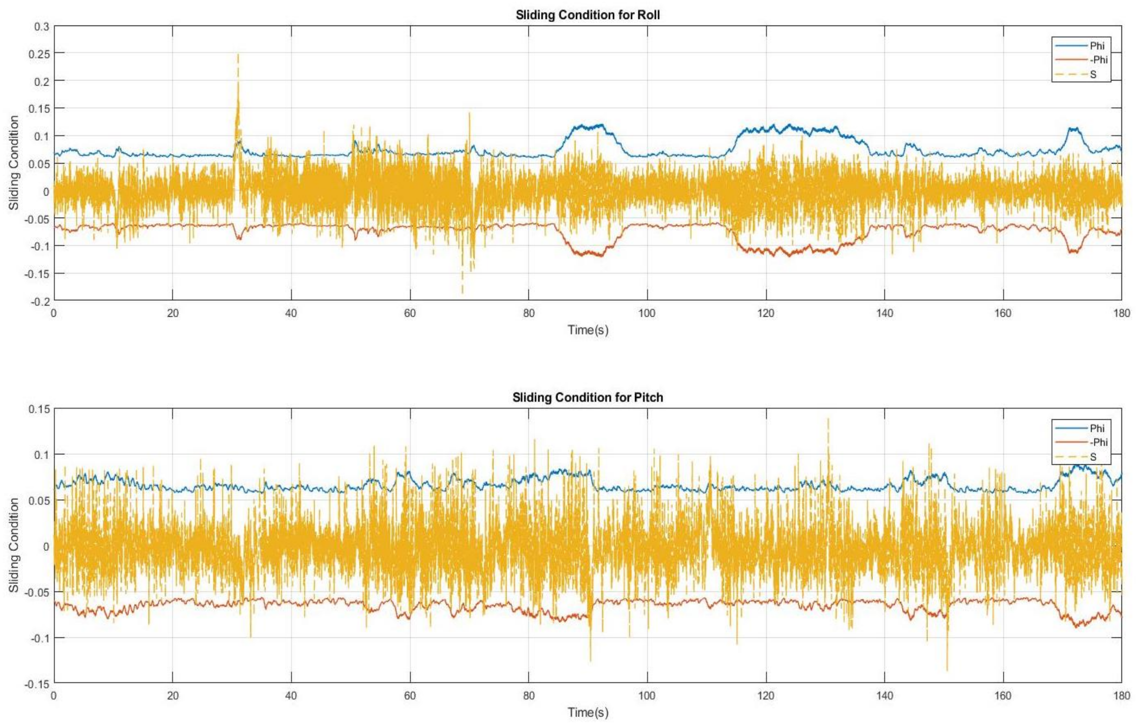


Figure 55: *Sliding Condition for MFSMC from Experimental Test*

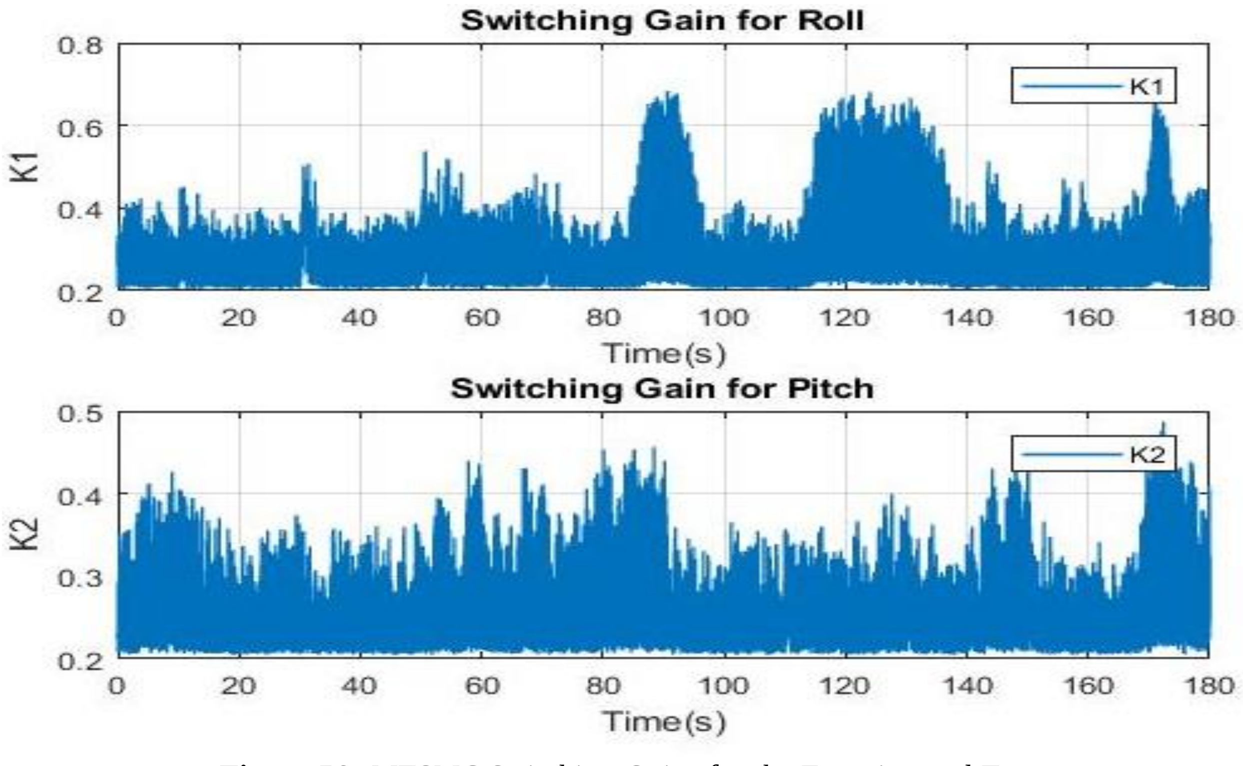


Figure 56: *MFSMC Switching Gains for the Experimental Test*

Figure 55 displays the sliding conditions for the MFSMC law and proves asymptotic tracking stability is maintained since the sliding conditions are satisfied. Figure 56 displays the MFSMC switching gains and, as with the simulation tests, are positive as expected.

6.2.1 Gimbal Hardware Testing with Different λ

The lambda (λ) parameter was varied to study the tracking precision for roll and pitch on the quadcopter. According to Eq. (28), λ is inversely proportional to the size of the boundary layer. Hence increasing λ, reduces the size of the boundary layer making the controller more aggressive. From Eq. (11) the sliding surface is directly proportional to the difference between the actual states and the desired states. Since noise cannot be eliminated, it is not reasonable to excessively reduce the size of the boundary by increasing λ. If the boundary layer is smaller than the peak-to-peak value of the noise, the sliding condition will not be satisfied [3].

The λ parameter was varied from 1 to 4 while η is updated according to Eq. (55).

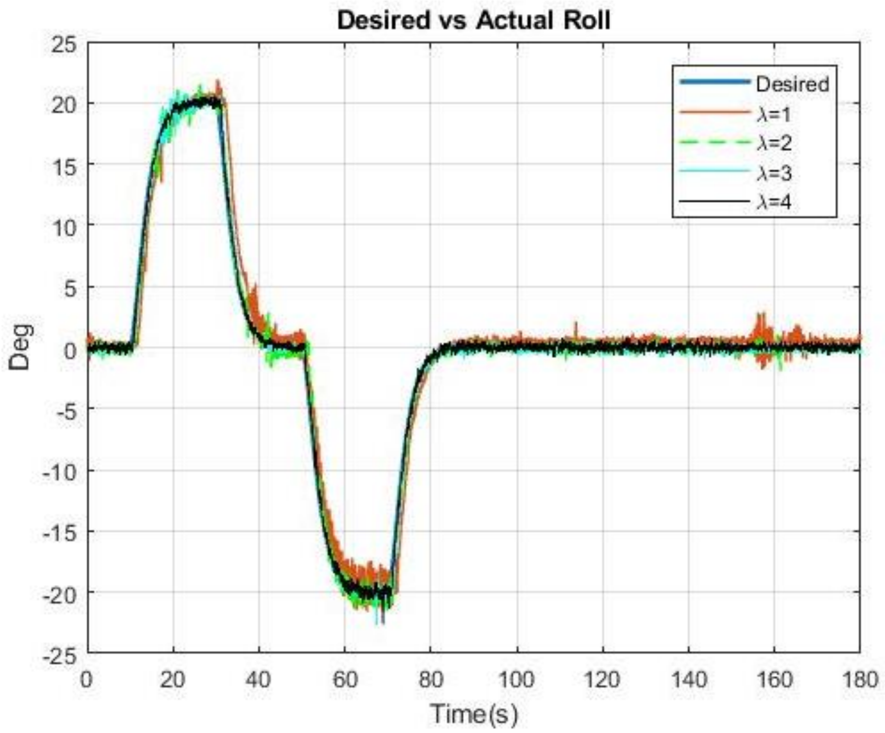


Figure 57: *Roll Tracking with Varying Lambda*

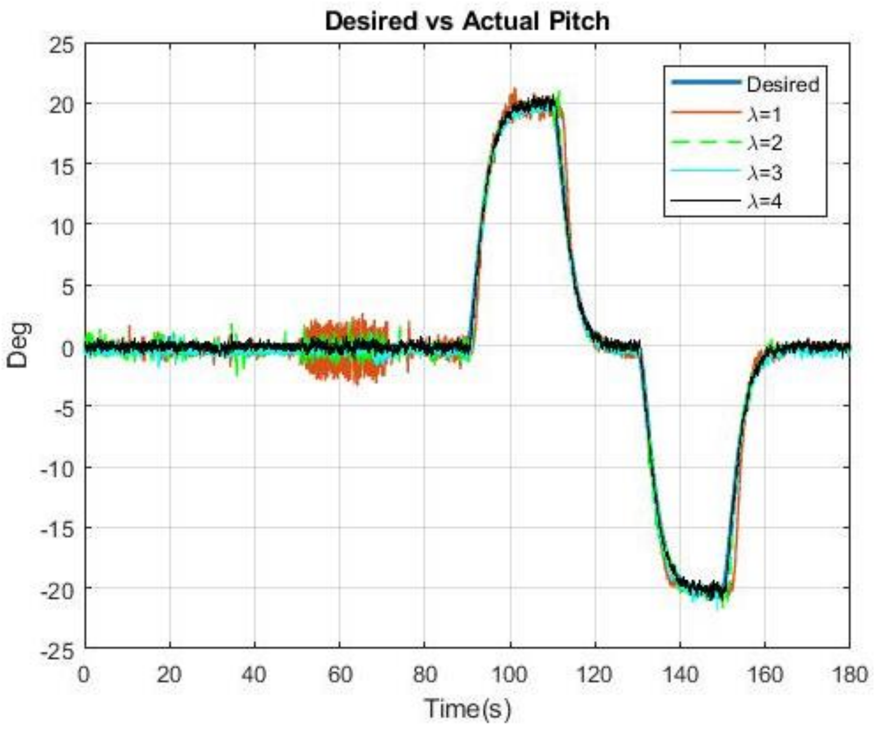


Figure 58: *Pitch Tracking for Varying Lambda*

Table 12: *RMS Error for Roll and Pitch Tracking with Varying Lambda*

λ (rad/sec)	**RMS**	
	Roll (deg)	Pitch (deg)
1	1.29	1.11
2	0.52	0.57
3	0.37	0.52
4	0.25	0.26

Figure 57 – 58 display the roll and pitch tracking for the MFSMC law with varying λ. Table 12 proves the tracking precision increases with increase in λ as expected since the boundary layer thickness is reduces with an increase in λ.

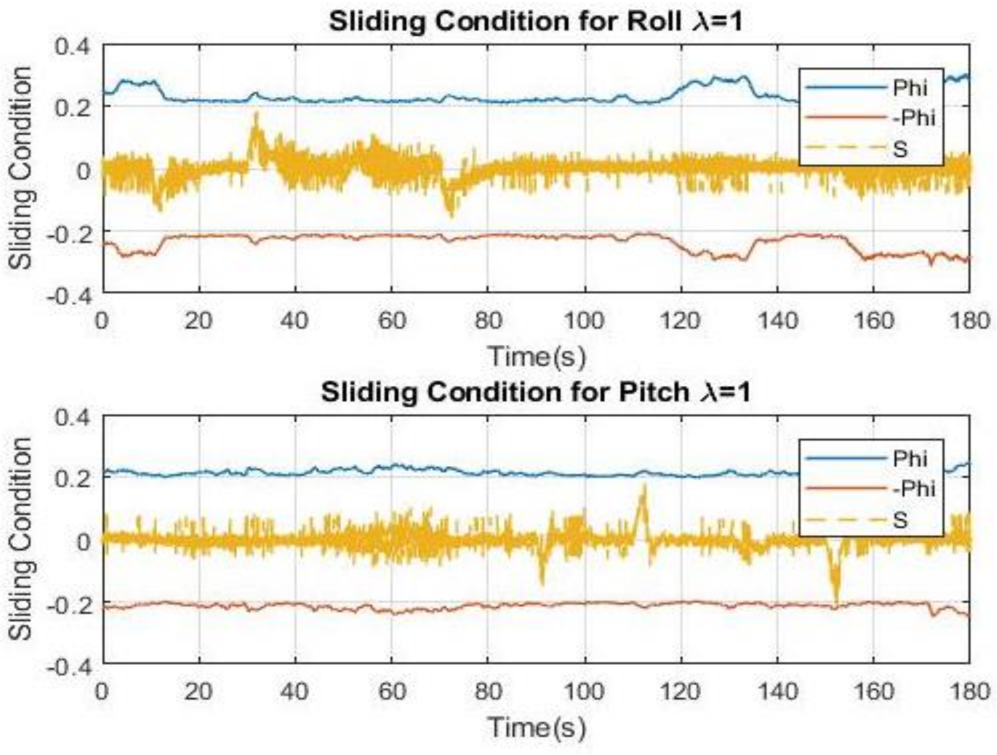


Figure 59: *Sliding Condition for Roll and Pitch for Lambda = 1 (rad/sec)*

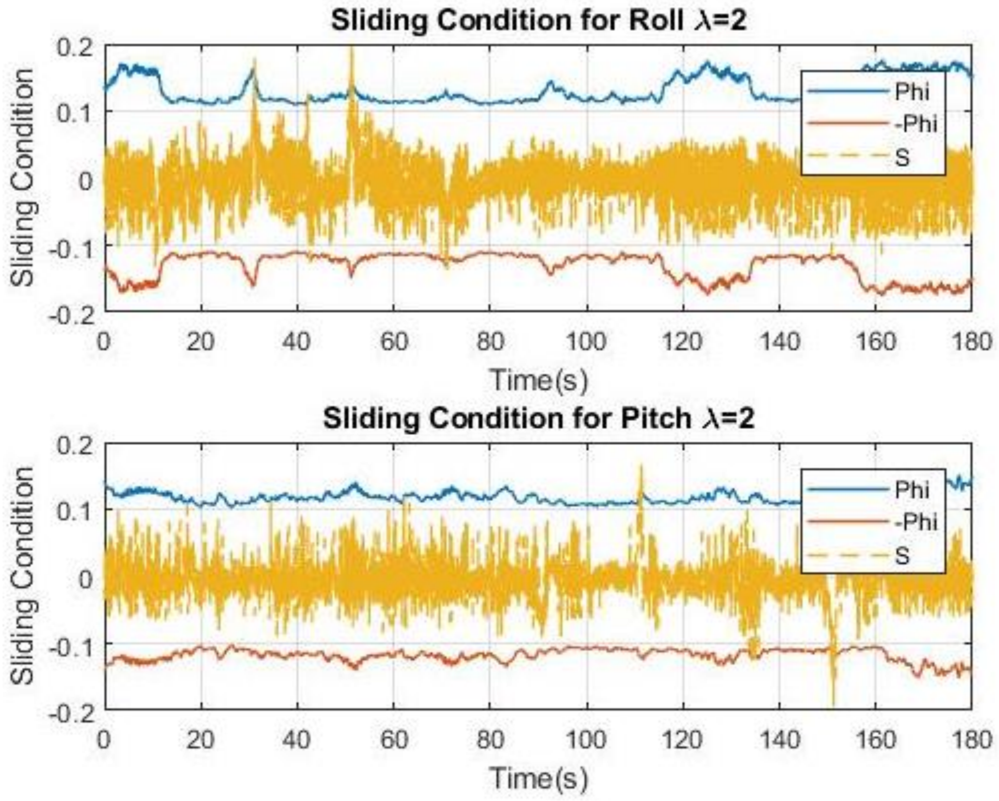


Figure 60: *Sliding Condition for Roll and Pitch for Lambda = 2 (rad/sec)*

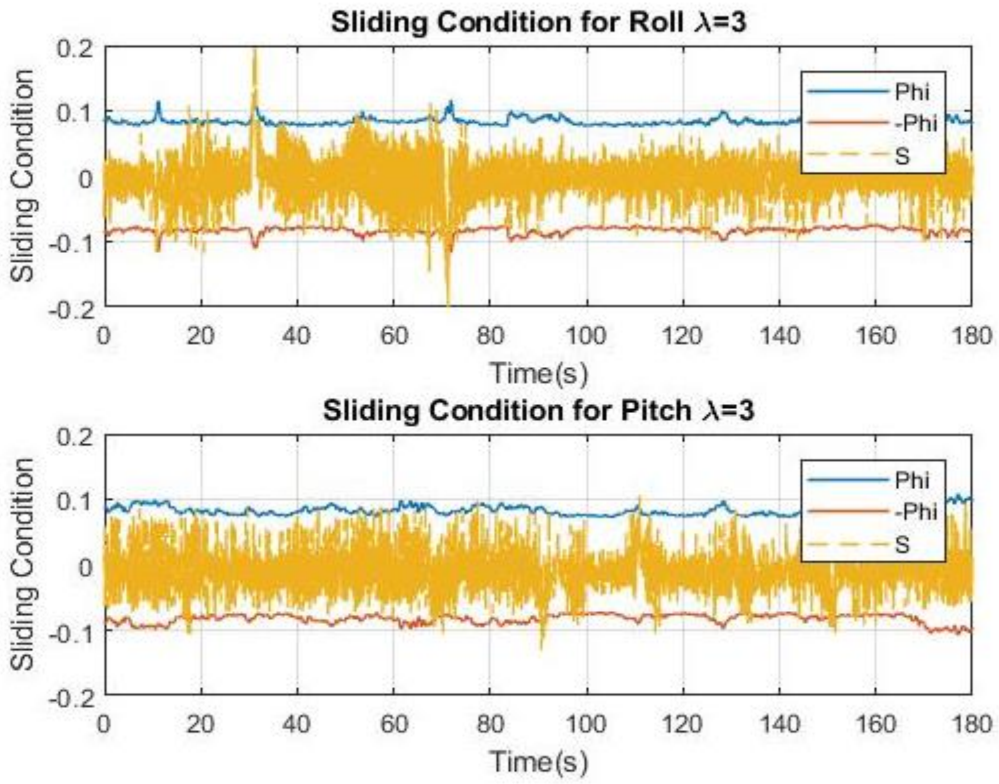


Figure 61: *Sliding Condition for Roll and Pitch for Lambda = 3 (rad/sec)*

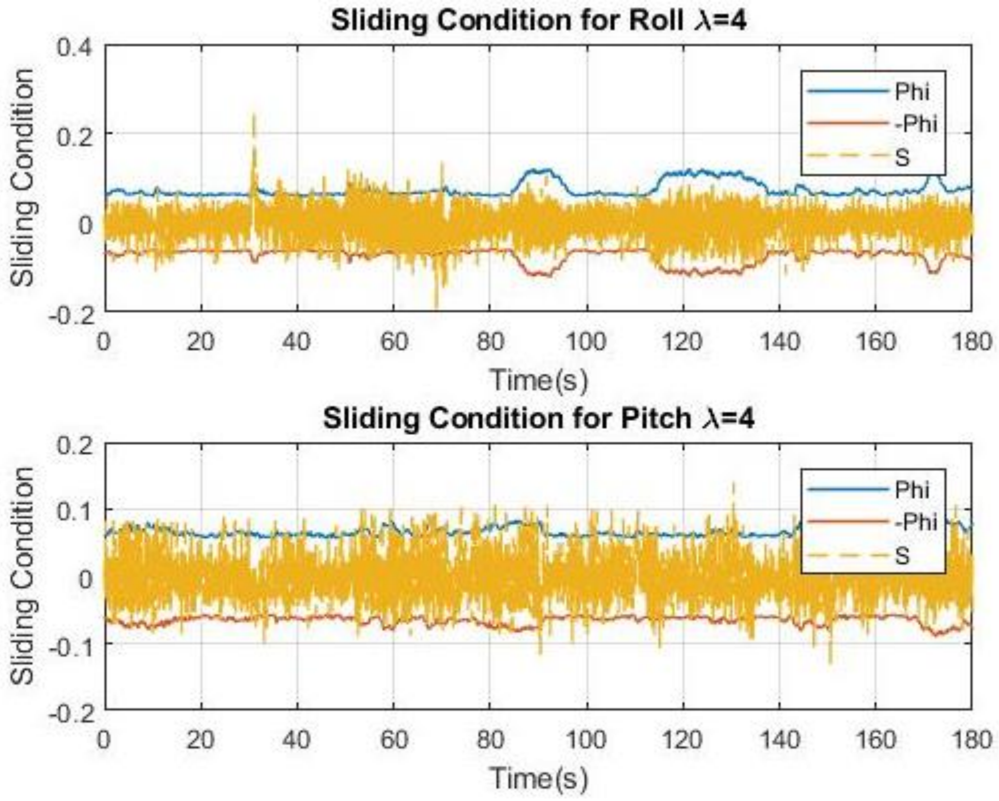


Figure 62: *Sliding Condition for Roll and Pitch for Lambda = 4 (rad/sec)*

Figures 59 – 62 display the sliding condition for the MFSMC law for roll and pitch tracking with varying λ. It is noted that the sliding condition is not satisfied for higher λ values since the noise present in the system is amplified. However, the closed-loop system is observed to be asymptotically stable and the hardware runs successfully with higher tracking precision at higher λ values.

7. CONCLUSIONS

The performance of the model-free control algorithm based on the sliding mode control with applications to unmanned aircraft systems is studied in this work. The tracking performance and power consumed by the MFSMC algorithm is compared to a PID controller. Simulation testing was performed to analyze the feasibility of controlling a quadcopter under both PID control and MFSMC control. The Fusion 1 quadcopter from Craft Drones was used for a hardware performance study. The following conclusions are observed from the work effort:

- The MFSMC algorithm was used to successfully track desired altitude, roll angle, pitch angle, and yaw angle responses using a basic 6 DOF quadcopter model programmed in Matlab. The tracking performance for quadcopter under MFSMC was better for roll and pitch tracking when compared to the PID controller. The altitude tracking precision was slightly lower than the PID controller and was attributed to the under-actuation present in the dynamic model of the quadcopter.
- Hardware components including sensor delays, sampling rates, actuator dynamics, state estimations, complimentary filters and noise were added into the simulation model. A full flight simulation and a gimballed platform simulation (tracking roll and pitch with an explicitly defined constant general thrust) was performed. The MFSMC law exhibited good tracking precision in the presence of noise and other various hardware restrictions and disturbances. The tracking precision of the MFSMC law was higher in the gimbal experiment. For the full model simulation the MFSMC law showed a higher tracking precision for roll and pitch control by an order-of-magnitude when compared to the PID controller. Similar tracking precision was observed for the altitude as the basic simulation model for both the controllers.

- The Fusion 1 drone from Craft Drones was used to test the performance of both the controllers. The quadcopter was mounted on a gimballed platform for analyzing roll and pitch control. Both the controllers tracked the desired commands successfully for both roll and pitch. However, the quadcopter under MFSMC exhibited a higher tracking precision for both roll and pitch verified by analyzing the RMS error over the time history control of the quadcopter
- The power consumption by the quadcopter under both the PID control and MFSMC was also studied. The minimum general thrust command required for optimal tracking and the motor rotation speed were directly observed from the hardware to conclude the power performance. The PID required a higher general thrust compared to the MFSMC algorithm. The motors operated at much lower speeds when the MFSMC algorithm was used as compared to the PID controller. The result proves that the MFSMC algorithm consumes less power as compared to the PID controller.
- The λ parameter in the MFSMC was varied to study the performance of the new type of control law. The tracking precision improved with increasing λ with the sliding condition spiking at certain points. These points were observed to increase as λ increased. However, the closed-loop system was completely stable and successfully tracked the desired signals.
- Finally, the MFSMC algorithm was successfully implemented on a real-world quadcopter with minimal tuning. The tracking performance was superior to a traditional PID control law who's tuning was tedious and time consuming. The quadcopter under MFSMC law exhibited less power consumption (more studies are required and are warranted to confirm this result).

8. FUTURE WORK

- The MFSMC law should be extended for full flight hardware testing not just using a gimbaled platform that restricted motion as was performed in this work. Simulation results indicate good tracking of altitude is possible. Therefore, the next logical step is to run flight tests. An ultrasonic transducer can be used to improve accuracy of state estimation of the altitude as compared to integrating twice from the accelerometer.
- As mentioned by Sen et al. [8], the system uncertainties can be overestimated or underestimated leading to excessive or insufficient control effort. This was observed during the hardware implementation of MFSMC law. Hence, online parameter estimation and real-time estimation techniques can be used to automatically estimate the control input gain bounds. A time varying λ parameter optimally updated to satisfy the sliding condition irrespective of the system characteristics can also be explored.
- Another important aspect to improve the MFSMC law is the initialization of the states. Since the MFSMC algorithm purely depends on the states and state derivatives, initialization plays an important role when implemented on hardware. Real-time initialization techniques should be researched to ensure correct initialization of all the system states.
- Improved state estimation techniques such as Kalman filters can be used. The Kalman filter, in general, is more robust and accurate as compared to the complementary filter used in this study. Another state estimation technique is possible using vision system to estimate quadcopter states. Vision sensing is more accurate and less noisy compared to an IMU and therefore, is preferred for differentiating rather than integrating the state estimation. An alternate state estimation technique is using a combination of vision and IMU sensors for estimating the required states.

9. REFERENCES

[1] Slotine, J.-J. E., and Li, W., 1991, “Sliding Control,” *Applied Nonlinear Control*, Prentice-Hall.

[2] Duan, L., Lu, W., Mora-Camino, F., and Miquel, T., 2006, “Flight-Path Tracking Control of a Transportation Aircraft: Comparison of Two Nonlinear Design Approaches,” *2006 Ieee/Aiaa 25TH Digital Avionics Systems Conference*, pp. 1–9.

[3] Crassidis, A., and Reis, R. M., 2016, “A Model-Free Sliding Mode Control Method,” *Proceedings of the 3rd International Conference of Control, Dynamic Systems, and Robotics*, Ottawa.

[4] Crassidis, A., and Tin, F. E., 2017, “A Model-Free Control System Based on the Sliding Mode Control Method with Apllications to Multi-Input-Multi-Output Systems,” *Proceedings of the 4th International Conference of Control, Dynamic Systems, and Robotics,* Ottawa.

[5] Laghrouche, S., Plestan, F., and Glumineau, A., 2003, “Higher Order Sliding Mode Control Based on Optimal Linear Quadratic Control,” *2003 European Control Conference (ECC)*, pp. 910–915.

[6] Ozguner, and U., 2006, “Adaptive Seeking Sliding Mode Control,” *2006 American Control Conference*, pp. 6 pp.-.

[7] Runcharoon, K., and Srichatrapimuk, V., 2013, “Sliding Mode Control of Quadrotor,” *2013 The International Conference on Technological Advances in Electrical, Electronics and Computer Engineering (TAEECE)*, pp. 552–557.

[8] Li, S., Li, B., and Geng, Q., 2014, “Adaptive Sliding Mode Control for Quadrotor Helicopters,” *Proceedings of the 33rd Chinese Control Conference*, pp. 71–76.

[9] Xu, R., and Özgüner, Ü., 2008, “Sliding Mode Control of a Class of Underactuated Systems,” Automatica, **44**(1), pp. 233–241.

[10] Olfati-Saber, R., 2002, “Normal Forms for Underactuated Mechanical Systems with Symmetry,” IEEE Transactions on Automatic Control, **47**(2), pp. 305–308.

[11] Khalil, H. K., 2002, *Nonlinear Control*, Prentice-Hall.

[12] Pai, M.-C., 2009, “Robust Tracking and Model Following of Uncertain Dynamic Systems via Discrete-Time Integral Sliding Mode Control,” Int. J. Control Autom. Syst., **7**(3), pp. 381–387.

[13] MILOSAVLJEVIC, D., 1985, “General Conditions for Existence of a Quasi-Sliding Mode on the Switching Hyperplane in Discrete Variable Structure Systems,” Autom. Remote Control, **46**, pp. 307–314.

[14] Sarpturk, S., Istefanopulos, Y., and Kaynak, O., 1987, "On the Stability of Discrete-Time Sliding Mode Control Systems," IEEE Transactions on Automatic Control, **32**(10), pp. 930–932.

[15] Lee, S.-H., 2003, "Sliding Mode Control Design Using Fast Output Sampling," *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475),* pp. 3543–3548 vol.4.

[16] Ferrara, A., and Rubagotti, M., 2009, "A Sub-Optimal Second Order Sliding Mode Controller for Systems With Saturating Actuators," IEEE Transactions on Automatic Control, **54**(5), pp. 1082–1087.

[17] Martínez-Guerra, R., Yu, W., and Cisneros-Saldaña, E., 2008, "A New Model-Free Sliding Observer to Synchronization Problem," Chaos, Solitons & Fractals, **36**(5), pp. 1141–1156.

[18] Salgado-Jimenez, T., Garcia-Valdovinos, L. G., and Delgado-Ramirez, G., 2010, "Depth Control of a 1 DOF Underwater System Using a Model-Free High Order Sliding Mode Control," *2010 IEEE Electronics, Robotics and Automotive Mechanics Conference*, pp. 481–487.

[19] Raygosa-Barahona, R., Parra-Vega, V., Olguin-Diaz, E., and Muñoz-Ubando, L., 2011, "A Model-Free Backstepping with Integral Sliding Mode Control for Underactuated ROVs," *2011 8th International Conference on Electrical Engineering, Computing Science and Automatic Control*, pp. 1–7.

[20] Munoz-Vazquez, A.-J., Parra-Vega, V., and Sanchez, A., 2014, "A Passive Velocity Field Control for Navigation of Quadrotors with Model-Free Integral Sliding Modes," J Intell Robot Syst, **73**(1), pp. 373–385.

[21] Crassidis, A., and Mizov, A., 2015, "A Model-Free Control Algorithm Derived Using the Sliding Mode Control Method," *Proceedings of the 2nd International Conference of Control, Dynamic Systems, and Robotics,* Ottawa.

[22] Mizov, A., 2015, "A Model-Free Control Algorithm Derived Using the Sliding Model Control Method," Theses.

[23] Levant, A., 2001, "Universal Single-Input-Single-Output (SISO) Sliding-Mode Controllers with Finite-Time Convergence," IEEE Transactions on Automatic Control, **46**(9), pp. 1447–1451.

[24] Utkin, V., 2016, "Discussion Aspects of High-Order Sliding Mode Control," IEEE Transactions on Automatic Control, **61**(3), pp. 829–833.

[25] Precup, R.-E., Radac, M.-B., Roman, R.-C., and Petriu, E. M., 2017, "Model-Free Sliding Mode Control of Nonlinear Systems: Algorithms and Experiments," Information Sciences, **381**, pp. 176–192.

[26] Dehghani, M. A., and Menhaj, M. B., 2016, "Integral Sliding Mode Formation Control of Fixed-Wing Unmanned Aircraft Using Seeker as a Relative Measurement System," Aerospace Science and Technology, **58**, pp. 318–327

[27] Qian, D., Yi, J., and Zhao, D., 2006, "Multiple Layers Sliding Mode Control for a Class of Under-Actuated Systems," *The Proceedings of the Multiconference on "Computational Engineering in Systems Applications,"* pp. 530–535.

[28] Schkoda, R., 2007, "Dynamic Inversion of Underactuated Systems via Squaring Transformation Matrix," Theses.

[29] Norton, M., Khoo, S., Kouzani, A., and Stojcevski, A., 2015, "Adaptive Fuzzy Multi-Surface Sliding Control of Multiple-Input and Multiple-Output Autonomous Flight Systems," IET Control Theory &amp; Applications, **9**(4), pp. 587–597.

[30] Abdulhamitbilal, E., 2014, "Robust Flight Sliding Modes Control System Design for Nonlinear Aircraft with Parameter Uncertainties," *2014 13th International Workshop on Variable Structure Systems (VSS)*, pp. 1–6.

[31] "Lyapunov-based Trajectory Tracking Controller for a Fixed-wing Unmanned Aerial Vehicle in the Presence of Wind - Brezoescu - 2015 - International Journal of Adaptive Control and Signal Processing - Wiley Online Library" [Online].

[32] Villanueva, A., Castillo-Toledo, B., Bayro-Corrochano, E., Luque-Vega, L. F., and González-Jiménez, L. E., 2015, "Multi-Mode Flight Sliding Mode Control System for a Quadrotor," *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 861–870.

[33] Derafa, L., Benallegue, A., and Fridman, L., 2012, "Super Twisting Control Algorithm for the Attitude Tracking of a Four Rotors UAV," Journal of the Franklin Institute, **349**(2), pp. 685–699.

[34] Crassidis, A., and Schulken, E., 2018, "Model-Free Sliding Mode Control Algorithms Inlcuding Application to a Real World Quadrotor.," *Proceedings of the 5th International Conference of Control, Dynamic Systems, and Robotics*, Ottawa.

[35] LYAPUNOV, A. M., 1992, "The General Problem of the Stability of Motion," International Journal of Control, **55**(3), pp. 531–534.

[36] Ferry, N., 2017, "Quadcopter Plant Model and Control System Development with MATLAB/Simulink Implementation," RIT.

[37] Levant, A., 1998, "Robust Exact Differentiation via Sliding Mode Technique**This Paper Was Recommended for Publication in Final Form by Associate Editor Hassan Khalil under the Direction of Editor Tamer Basar.," Automatica, **34**(3), pp. 379–384.

[38] Islam, T., Islam, M. S., Shajid-Ul-Mahmud, M., and Hossam-E-Haider, M., 2017, "Comparison of Complementary and Kalman Filter Based Data Fusion for Attitude Heading Reference System," AIP Conference Proceedings, **1919**(1), p. 020002.

[39] Ngo, H.--, Nguyen, T.-, Huynh, V.--, Le, T.-, and Nguyen, C.-, 2017, "Experimental Comparison of Complementary Filter and Kalman Filter Design for Low-Cost Sensor in Quadcopter," *2017 International Conference on System Science and Engineering (ICSSE)*, pp. 488–493.