Proceedings of the 5th International Conference of Control, Dynamic Systems, and Robotics (CDSR'18)
Niagara Falls, Canada – June 7 – 9, 2018
Paper No. 112
DOI: 10.11159/cdsr18.112

Model-Free Sliding Mode Control Algorithms including Application to a Real-World Quadrotor

Eric Schulken, Agamemnon Crassidis
Rochester Institute of Technology
1 Lomb Memorial Drive, Rochester, New York, 14623, USA
ems1552@rit.edu; alceme@rit.edu

Abstract - Sliding mode control is a robust nonlinear control algorithm that has been used to implement tracking controllers for unmanned aircraft systems that are robust to modeling uncertainty and exogenous disturbances, thereby providing excellent performance for autonomous operation. A significant advance in the application of sliding mode control for unmanned aircraft systems would be adaptation of a model-free sliding mode control algorithm, since the most complex and time-consuming aspect of implementation of sliding mode control is the derivation of the control law with incorporation of the system model, a process required to be performed for each individual application of sliding mode control. The performance of various model-free sliding mode control algorithms was compared in simulation using a variety of aerial system models and real-world disturbances (e.g. the effects of discretization and state estimation). The two best performing algorithms were shown to exhibit very similar behavior. These two algorithms were implemented on a quadrotor (both in simulation and using real-world hardware) and the performance was compared to a traditional PID based controller using the same state estimation algorithm and control setup. Simulation results show the model-free sliding mode control algorithms exhibit similar performance to PID controllers without the tedious tuning process. Comparison between the two model-free sliding mode control algorithms showed very similar performance as measured by the quadratic means of tracking errors. Flight testing showed that while a model-free sliding mode control algorithm can control real-world hardware, further characterization and significant improvements are required before it is a viable alternative to conventional control algorithms. Large tracking errors were observed for both the model-free sliding mode control and PID based flight controllers and the performance was characterized as unacceptable for most applications. The poor performance of both controllers suggests tracking errors can be attributed to errors in state estimation. Further testing with improved state estimation would allow for more conclusions to be drawn.

***Keywords*:** Model-Free Control, Sliding Mode Control, Robust Control, Flight Control, Unmanned Aircraft Systems.

1. Introduction

As the ubiquity of mobile robots and autonomous systems increases, so does the need for reliable automatic control with relaxed methods for tuning and implementation. A frequent requirement of a nonlinear control scheme is that it be robust to unknown perturbations. A popular and relatively simple form of robust nonlinear control is Sliding Mode Control (SMC) [1, 2-6]. Another method of compensating for unknown model parameters or disturbances is the family of adaptive control [1]. Adaptive control can be a good alternative to robust control in certain cases [7]. A popular topic for research is the combination of robust and adaptive control methods [3, 7-10], though the methods proposed are often highly complex and unique to the intended applications. Another area of active research is that of model-free robust control [3, 11-18], since one of the main drawbacks of sliding mode control and other methods is the time-consuming control law derivation process that is required for each implementation [11-13].

A Model-Free SMC (MFSMC) based flight controller would serve as a useful tool for the rapid deployment of new configurations or iterations of Unmanned Aircraft Systems (UASs). The advanced control law would allow for accurate and robust tracking control to become an easily attainable goal for practitioners who lack the time or expertise to implement robust nonlinear control using current methods. Reduced development time and reduced cost could be realized during the development of autonomous systems for critical tasks such as search and rescue, transportation, or site surveying, thereby allowing more people to be served and freeing up capital to be invested in other advances in the field of control systems engineering.

2. Background

Sliding mode control is a form of robust nonlinear control that can provide, theoretically, "perfect" performance in the face of modeling imprecision for certain classes of systems [1]. In order to implement SMC, an engineer needs to have a nominal or "best guess" plant model, all uncertainties must be matched, and the uncertainties must have known bounds. In its simplest form, a SMC control law can be expressed as:

$$u = u_{eq} + u_{cor} \tag{1}$$

where u_{eq} is the equivalent control calculated from the nominal model, and u_{cor} is a control correction based on tracking errors and the bounds of uncertainties [2]. With this distinction in mind, there are two general approaches to MFSMC. The first is to neglect the equivalent control since it is based on the system model and to try and find a form of u_{cor} that results in the desired tracking while remaining stable and feasible [3, 14-16, 18]. The other approach is to estimate u_{eq} based only on system inputs and outputs, and then to derive u_{cor} with conventional SMC methods [11-13, 17].

Concepts of SMC were first proposed in the 1970s by Utkin, among others, as a significant result of the study of Variable Structure Systems (VSS) [19]. SMC making use of first-order sliding surfaces is referred to as conventional SMC [20]. Conventional SMC is based on the idea of a sliding surface in the state plane defined by:

$$S = \left(\frac{d}{dt} + \lambda\right)^{n-1} \tilde{x} \tag{2}$$

where n is the order of the system being controlled, λ is a positive constant that is inversely proportional to the reaching time, and $\tilde{x}$ is the error state. Young, Utkin, and Ozguner [4] presented a thorough guide to SMC from the perspective of a control engineer. Issues addressed include continuous-time sliding modes, discrete-time sliding modes, and acceptable and realizable chattering mitigation strategies.

2.1. Developments in Sliding Mode Control

Fridman [3] gives a concise history of SMC and reviews the main developments in the field after 1990 when SMC began to attract significant interest in the controls community. The main disadvantages of conventional SMC listed as motivations for the development of alternative SMC schemes include chattering induced by the discontinuous control law or by parasitic dynamics, sensitivity (i.e., not robust) to mismatched disturbances, non-finite time (only asymptotic) convergence of states, sensitivity to noise, difficulty in discretization for implementation on a digital computer, and the requirement of higher-order derivatives for calculation of the sliding variable. Much of the past and current research related to SMC has been targeted at one or more of the aforementioned issues. Some of the results of SMC research include Second-Order Sliding Modes (SOSM) which resulted in the twisting algorithm, the super twisting algorithm, and the sub-optimal algorithm. Villanueva et al. [5] used the super twisting algorithm to derive a multi-mode control system for a quadrotor. Derafa, Benallegue, and Fridman [23] also implemented super twisting on a quadrotor platform and included real-world testing in their results. The super twisting algorithm was conceived as a way to provide the same tracking performance as conventional SMC but with continuous control effort.

The concepts of SOSM were extended to arbitrary order sliding modes, also known as High Order Sliding Mode (HOSM), which is often purported to offer several advantages over conventional SMC, namely a theoretical guarantee of finite time convergence of the states to their desired values, the removal of the requirement that the relative degree be equal to one, and attenuation of chattering [20]. The concept of HOSM was first introduced by Levant, who proposed the second-order sliding mode [3] and later went on to publish many works on HOSM and its applications. Utkin explored these claims by comparing the performance of HOSM based controllers to conventional SMC controllers for the same worst-case conditions [20]. An important point is that SMC often causes chattering in the control effort due to the discontinuous term in the control law; chattering can also occur as a result of unmodeled dynamics such as actuator or sensor delays. These so-called parasitic dynamics can lead to relatively high frequency oscillations in the control effort even if it is smooth, as is guaranteed for HOSM. The assurance of finite time convergence provided by HOSM theory is shown to lose some meaning and actually have adverse effects in the presence of unmodeled actuator dynamics where the

magnitude of chattering is actually greater for HOSM than for conventional SMC with smoothing. Similar criticisms of super twisting control were presented, although HOSM algorithms do have significant value in the right applications.

Levant [18] presented a unique method of MFSMC based on HOSM theory. The controller form is based on the extension of the trivial (and usually unstable) relay controller $u = -K\,\mathrm{sgn}(S)$ with $S = x - x_d$ to satisfy higher-order sliding modes. Since the sliding surface S and its derivatives are only based on the states and their desired values and parameters able to be generalized for each order r, no information about the plant other than its relative degree is required for the controller. Levant [18] found that, in the case of $r = 4$, the HOSM control law yielded satisfactory results in simulation. Though this method (and most HOSM methods) would eliminate chattering in the ideal scenario, chattering or even instability may still occur in practice due to the excitation of parasitic dynamics [20].

2.2. Model-Free Sliding Mode Control Algorithms to Investigate

To control variability, the same sliding variable design (Eq. 2) was used for all MFSMC algorithms. Simulations of the MFSMC algorithm proposed by Levant [18] showed that instability occurs even for relatively minor feedback delays or state estimation errors, thus it was discounted as a candidate for implementation. A unique form of MFSMC based on previous control inputs was proposed in [11]. This approach was improved in [12] and extended to multi-input multi-output (MIMO) systems in [13]. The MFSMC algorithm proposed by Reis [12] along with the first MFSMC algorithm presented by Precup et al. [17] both use expressions that serve the same purpose as the system model. The 'system model' expression are generated using measurements of the state and control input. The algorithm proposed by Reis [12] is input-based, while that proposed by Precup et al. [17] is state-based. The input-based 'system model' takes the form of:

$$x^{(n)} = x^{(n)} + b[u_k - u_{k-1} - \varepsilon(u)] \tag{3}$$

where $\varepsilon(u) = u_k - u_{k-1}$ and is estimated as:

$$\hat{\varepsilon}(u) = u_{k-1} - u_{k-2} \tag{4}$$

and where the estimate of $\varepsilon(u)$ is bounded according to:

$$|\varepsilon(u)| \leq (1 + \sigma)|\hat{\varepsilon}(u)| \tag{5}$$

with $\sigma = 0.5$. Using Eqs. (3), (4), and (5) and the derivation method for conventional SMC (see [1]) along with a time varying smoothing boundary layer to mitigate chattering. The input-based MFSMC algorithm takes the form:

$$u = \hat{b}^{-1}[-\dot{S} - (K - \dot{\varphi})\mathrm{sat}(S, \varphi)] + 2u_{k-1} - u_{k-2} \tag{6}$$

with the switching gain is defined by:

$$K = \left|(1 - \beta)\dot{S}\right| + \left|\hat{b}\sigma(u_{k-2} - u_{k-1})\right| + \beta\eta \tag{7}$$

where $\hat{b} = \sqrt{b_u b_l}$ and $\beta = \sqrt{b_u / b_l}$ are calculated from known upper and lower input gain bounds, the small positive constant $\eta = 0.15$, and time varying boundary layer, φ is defined as:

$$\dot{\varphi} = K - \lambda\varphi \tag{8}$$

An interesting notational difference between the input-based MFSMC law, as expressed in Eqs. (6) and (7) and as expressed in [12] and [13], is the expression of $\dot{S}$ in the control law rather than its constituent terms. The notational difference means that the control law is now arbitrary order, provided that the sliding variable and its derivative are properly constructed for the given system order.

The state-based MFSMC algorithm is based on the MFSMC algorithms proposed by Precup et al. [17], though the derivation process (and thus end result) described herein are different. The system model for the state-based MFSMC is:

$$x^{(n)} = f(\vec{x}) + bu \tag{9}$$

The instantaneous value of the unknown nonlinear function, f, can be estimated as:

$$\hat{f}(\vec{x}) = x^{(n)} + \hat{b}u \tag{10}$$

where the estimate $\hat{f}(\vec{x})$ is bounded according to:

$$|f(\vec{x})| \leq (1+\sigma)\left|\hat{f}(\vec{x})\right|` \tag{11}$$

with the tuneable parameter σ again conservatively set to $\sigma = 0.5$. The system model can be estimated as:

$$\hat{x}^{(n)} = \hat{f}(\vec{x}) + \hat{b}u \tag{12}$$

Using the estimated system model, state-based MFSMC laws can be derived for specific orders of systems. For a second-order system and using a time varying smoothing boundary layer, the control law is:

$$u = \hat{b}^{-1}[-\hat{f}(\vec{x}) + \ddot{x}_d - \lambda\tilde{x} - (K - \dot{\varphi})\text{sat}(S, \varphi)] \tag{13}$$

where $\hat{b} = \sqrt{b_u b_l}$, $\beta = \sqrt{b_u / b_l}$, and $\dot{\varphi}$ is defined by Eq. (8). The switching gain, K, is defined as:

$$K = |\beta - 1|\left|\hat{f}(\vec{x}) - \ddot{x}_d + \lambda\tilde{x}\right| + \left|\beta\sigma\hat{f}(\vec{x})\right| + |\beta\eta| \tag{14}$$

3. Methodology

The goal of this work is to drive the state-of-the-art towards the reality of having a model-free robust nonlinear control algorithm capable of being applied to real-world hardware with high performance and minimal tuning. There have not been any comparisons, direct or otherwise, of MFSMC algorithms to each other or to established methods of control (e.g. PID). In addition, no work (to data) has yet been performed to examine the effects of realization on the performance of MFSMC algorithms. These issues were addressed through carefully designed simulation studies and real-world hardware testing on a quadrotor UAS. Issues addressed in the simulation studies include: actuator dynamics, sensor dynamics, measurement noise, filtering, zero-order-holds, transport delays, state estimation, controller sampling frequency, and disturbance rejection. In simulation and hardware testing, the performance of MFSMC flight controllers was compared to same controller architecture implemented with PID.

4. Quadrotor Implementation

The selected hardware platform is the Parrot® AR.Drone 2.0, which is a consumer grade quadrotor originally intended to be controlled remotely via a mobile device. A Simulink Embedded Coder target for the AR.Drone 2.0 allows for Simulink block diagrams to be compiled into executable code for the hardware board, and for data to be logged in Simulink [22]. Onboard sensors include a 3-axis accelerometer, 3-axis rate gyroscope, 3-axis magnetometer, a downward facing ultrasonic transducer, a barometer, a forward-facing camera, a downward-facing camera, and a USB port to which a GPS module can be connected. The AR.Drone 2.0 runs embedded Linux 2.6.32 on a 1 GHz ARM Cortex A8 processor and connects to other hardware via Wi-Fi. Each of the 4 rotors is driven by 15 Watt high RPM brushless DC motors.

4.1. System Model and State Estimation

The 6 DOF ($x, y, z, \varphi, \theta, \psi$) quadrotor model used was adapted from [5, 23]. The model consists of 6 second-order nonlinear differential equations. Position is described in earth referenced cartesian coordinates and orientation in earth referenced Euler angles. The four inputs (Ui) to the system are defined by the reaction forces and moments of the four

rotors and represent: vertical lift (in the body reference frame), rolling moment, pitching moment, and yawing moment. The system is underactuated since lateral or longitudinal forces cannot be generated on the vehicle. The system model is:

$$\ddot{x} = \frac{U_1}{m}[\sin(\varphi)\sin(\psi) + \cos(\varphi)\sin(\theta)\cos(\psi)] - A_x;\ \ddot{y} = \frac{U_1}{m}[-\sin(\varphi)\cos(\psi) + \cos(\varphi)\sin(\theta)\sin(\psi)] - A_y$$
$$\ddot{z} = \frac{U_1}{m}[\cos(\varphi)\cos(\theta)] - g - A_y \tag{15}$$

$$\ddot{\varphi} = \frac{1}{I_x}\left[U_2 + \left(I_y - I_z\right)\dot{\theta}\dot{\psi}\right] - A_p;\ \ddot{\theta} = \frac{1}{I_y}\left[U_3 + (I_z - I_x)\dot{\varphi}\dot{\psi}\right] - A_q;\ \ddot{\psi} = \frac{1}{I_z}\left[U_4 + \left(I_x - I_y\right)\dot{\varphi}\dot{\theta}\right] - A_r \tag{16}$$

Because the system is underactuated, only four degrees-of-freedom can be controlled directly and independently. Horizontal positional control could be achieved with a guidance algorithm to specify desired roll and pitch trajectories, though this is outside the scope of the current work. Estimations of the four independently controllable states of altitude and the three Euler angles relied on measurements from the ultrasonic transducer, magnetometer, accelerometer, and gyroscope. An altitude estimate was obtained by passing the output of the ultrasonic transducer through a first-order discrete lowpass filter with a time constant of 0.1 sec. The assumption is the ground beneath the quadrotor is flat and provides accurate returns and the quadrotor remains level throughout its flight.

Under the assumption that the quadrotor is not undergoing any linear accelerations, the roll and pitch angles can be calculated from accelerometer data. Since the yaw axis is parallel with the gravity vector, it cannot be calculated in the same way as roll and pitch. Yaw angle is instead calculated using the magnetometer.

Final estimates of the three Euler angles were obtained using a fixed frequency complimentary filter of the form:

$$\varphi^* = \frac{\tau\dot{\varphi}_{gyro}}{\tau s+1} + \frac{\varphi_{acc}}{\tau s+1} \tag{17}$$

where $\dot{\varphi}_{gyro}$ is taken directly from the gyroscope and φ_{acc} is calculated according to Eq. (16). The time constant, τ, for this filter was taken as 1 sec and the same form was used to generate the estimates θ^* and ψ^*.

The estimation process for altitude and Euler angles remains the same for the MFSMC algorithm as for PID. The key difference is that the MFSMC algorithm requires full state feedback, meaning that estimates of the linear and angular velocities and accelerations are also required. Angular acceleration is estimated using the outputs of the gyroscope passed through a first-order discrete lowpass filter with a time constant of 0.1 sec. The remaining estimates needed to be differentiated from existing signals with minimal phase lag and noise. For this purpose, sliding mode differentiators [24] were employed for providing estimates of the differentiated signals.

4.2. Controller Design

Control inputs were translated into motor inputs using algebraic motor mixing functions. The PID flight controller design uses four discrete PID controllers (one for each independently controllable degree-of-freedom) operating at 400 Hz. The two MFSMC algorithms, i.e., Eqs. (6), (7), (13), and (14) require feedback of control inputs, and thus estimates of actuator dynamics since the outputs of the actuators cannot be measured directly. The actuator dynamics were modelled as first-order with a time constant of 0.3 sec (compared to an actuator time constant of 0.25 sec used in the system model). The input gain bounds required for the MFSMC algorithms were estimates of the mass and moments of inertia of the quadrotor. Similar to the PID-based controller, the MFSMC controllers operated at a discrete frequency of 400 Hz.

4.3. Simulations

For simulation, sensor models were developed to replicate expected state estimation errors. The dynamic responses of the sensors are assumed to be negligible, with errors arising from zero-order-holds, random (assumed Gaussian) noise, and imperfect state estimation. The AR.Drone 2.0 sensors update at 200 Hz. The accelerometer and magnetometer signals are taken from the Bosch BMC150 module. The accelerometer is said to have a typical noise variance of 150 μg while the magnetometer has a typical noise variance of 1 μT with a measurement range of ±1300 μT [25]. The gyroscope in the AR.Drone 2.0 is an InvenSense IMU-3000 with a noise variance of 0.2 °/sec [26]. The state estimation algorithm for yaw

angle was modelled by applying a 200 Hz zero-order-hold and adding random noise to the yaw (ψ) signal. A noise variance of 0.001 rad was approximated from the measurement range and variance specified on the data sheet for the magnetometer [25]. The gyroscope was modelled by taking the signals for angular rates ($\dot{\varphi}$, $\dot{\theta}$, $\dot{\psi}$), applying a 200 Hz zero-order-hold, and adding noise (3.49e-3 rad/sec variance) with a bias of 0.01 rad/sec to reflect the fact that the gyroscope calibration tends to drift over time. Since the accelerometer is aligned with the body axes of the quadrotor, the acceleration signals ($\ddot{x}$, $\ddot{y}$, $\ddot{z}$) must be rotated by the Euler angles. The influence of gravity (+1 g aligned with the z axis) is added to the acceleration signals before they are rotated. After the rotation, random noise with a variance of 1.5e-4 g is added to the signals. Since the ultrasonic transducer measures distance to the ground below the AR.Drone 2.0 (assumed to be a smooth horizontal plane) and is aligned with the vertical body-axis of the quadrotor, the roll and pitch angles (φ, θ) of the vehicle will influence this measurement and cause it to deviate from the true value of height above the ground. Prior to the addition of random noise, the signal z is divided by the cosines of φ and θ, thus calculating the distance to a horizontal plane along the vertical body axis of the quadrotor. No information could be found regarding the accuracy or measurement noise of the ultrasonic transducer, though the update rate is known to be 25 Hz. The arbitrarily chosen variance used in the model corresponds to a 3σ accuracy of 1.2 cm.

For the PID controller, the initial condition of the integrator for the altitude controller was set to compensate for gravity. For all simulations, a 3ms input delay was applied between the controller and the system model. For the results of the PID controller (Fig. 1), the three Euler angles settle to -0.01 rad due to the bias of 0.01 rad that was added to the gyroscope model. State estimation results (Fig. 1) show that the estimated values for the Euler angles are driven to zero, which confirms that the controllers are acting as intended and the observed tracking errors are primarily due to state estimation errors. Both quantitatively and qualitatively, the simulated performance of the input-based MFSMC algorithm (Fig. 2) and the state-based MFSMC algorithm (Fig. 1) are very similar, and are thus addressed collectively. Like the PID results, the three Euler angles settle to nonzero values for the MFSMC simulation results. Unlike the PID results, however, this cannot be directly attributed to errors in state estimation since Fig. 1 shows that the controllers do not drive the estimated trajectories to zero as was the case with PID. When the simulation was run with the gyroscope bias set to zero, steady-state error for the Euler angles was reduced but still present. A possible solution is to make the controller more aggressive by increasing the parameter λ from its current value of 1. Further simulations showed that the system becomes marginally stable with unacceptable oscillations for a value of $\lambda = 2$ and completely unstable for larger values.

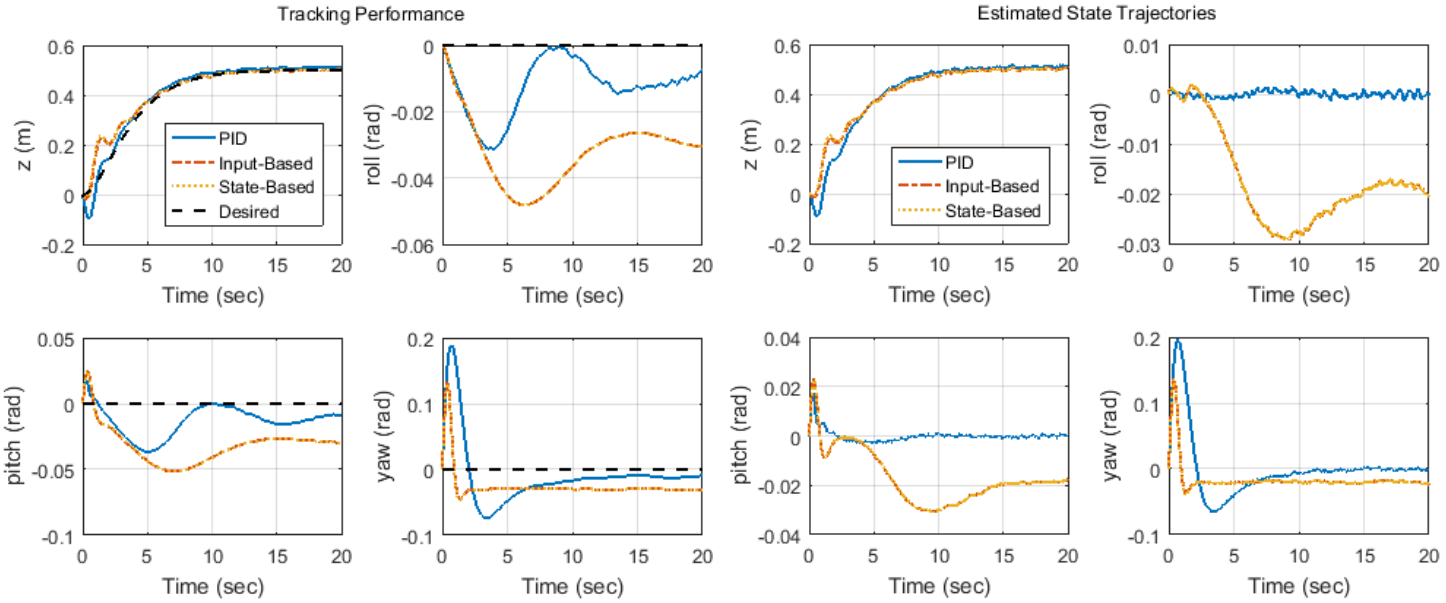

Fig. 1: Overlay of simulation results for the PID-based, input-based MFSMC, and state-based MFSMC flight controllers. The left four plots show the tracking of system states with desired trajectories. The right four plots show the performance of the state estimation algorithm and therefore the feedback data to the controllers.

4.4. Flight Testing

With simulation results indicating that the performance of both the input-based and state-based MFSMC algorithms was essentially equal, the state-based algorithm was (arbitrarily) selected for hardware implementation. Testing (including final tuning of the controllers) was conducted outdoors inside a netted enclosure. Most flights exhibited significant instability. Data from the best flights using each controller are shown in Fig. 2. For these flights, the quadratic means of the tracking errors are significantly worse than the same values from the simulations outlined in Section 4.3. It is of significant note that the tracking errors for the simulations were calculated from the true values of the states, rather than from the estimated values as is the case for the hardware results. As was seen with the results of the simulated state estimation algorithm, the estimated values of the states can exhibit significant error in certain conditions. Comparing the performance of the MFSMC controller to the PID controller, the MFSMC controller does a better job tracking the desired altitude trajectory, while the PID controller performs better at tracking roll and pitch angles. Regardless of the quantitative performance metrics, neither controller yielded satisfactory tracking, though that does not mean they could not be tuned to improve performance.

The instability observed in the performance of the MFSMC flight controller (Fig. 2) cannot be fully explained or attenuated since the system has not been fully characterized. With knowledge of the dynamics at play, it may be possible to elucidate the interactions causing the observed oscillations and instability, and to design methods for their elimination. A critical oversight is the lack of validation for the state estimation algorithm. The simulated performance of the state estimation algorithm shows that significant errors are possible, most notably in the estimated roll and pitch angles. If these or other state estimates are significantly disparate from their true values, then the state estimation algorithm has the unintended effect of adding unknown dynamics in the feedback loop of the system. Since simulations using the same state estimation algorithm as the hardware implementation showed significantly better performance than the hardware results, it may well be the case that additional factors are adversely affecting the performance of the controller. These factors could include measurement delay, input delay, missed samples, poor sensor calibration, unmodeled sensor or system dynamics, or exogenous disturbances. Future development of MFSMC algorithms should focus on proper tuning and disturbance mitigation techniques, while future work for the quadrotor application should focus on the design of state estimation and guidance algorithms.

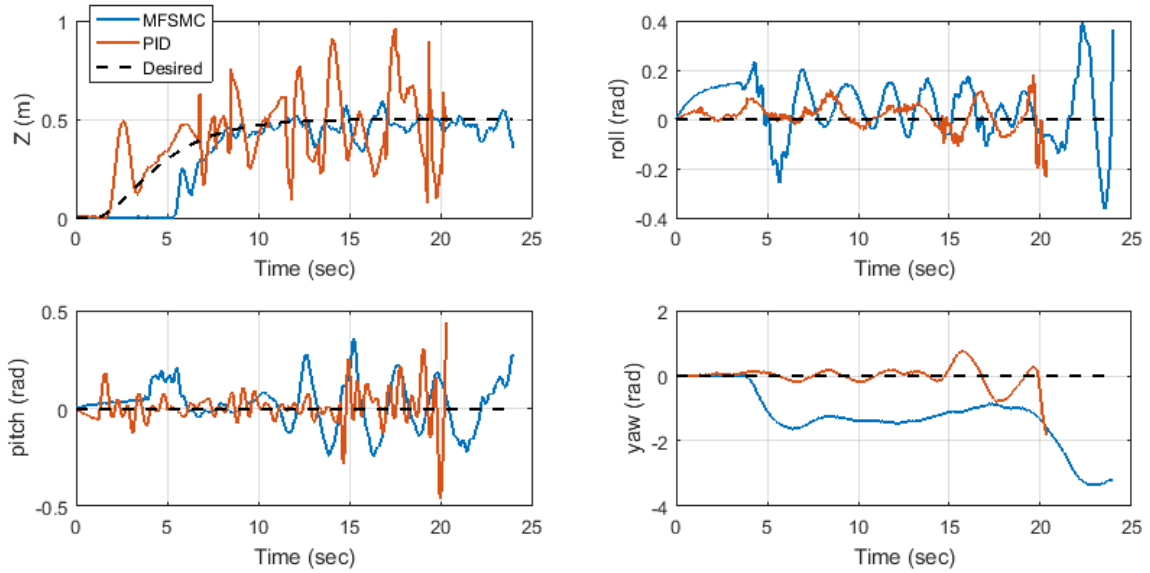

Fig. 2: Comparison of tracking performance achieved by the PID-based and state-based MFSMC flight controllers running on the AR.Drone 2.0.

In the light of the work presented herein, the continued development of practical MFSMC algorithms has a clear path forward. The state-based MFSMC algorithm, shown by Eqs. (13) and (14), has been shown to be realizable and capable of controlling a quadrotor. Simulations have indicated that with improved state estimation and guidance algorithms, the performance of a MFSMC flight controller can approach or exceed that of a PID-based flight controller without the tedious

tuning process. Another significant result is that the input-based MFSMC algorithm and the state-based MFSMC algorithm have been shown to provide nearly identical performance in simulation.

5. Conclusion

Model-Free Sliding Mode Control (MFSMC) algorithms have been shown to be realizable despite the effects of discretization, delay, measurement noise, actuator dynamics, and imperfect state estimation. Application of MFSMC algorithms to the control of a quadrotor was selected for study not because of a need for the high performance or disturbance rejection often touted of SMC algorithms, but because of the supposed ease of implementation compared to conventional control algorithms (e.g., PID). Indeed, real-world testing of the quadrotor flight controllers confirmed that MFSMC is realizable and requires very little tuning compared to PID, but underscored the need for accurate state estimation as well as complete characterization of the behavior and tunability of MFSMC algorithms. While both PID and MFSMC based flight controllers did allow the quadrotor to fly autonomously, their performance was marginal at best, and likely unsuitable for most applications.

References

[1] J.-J. E. Slotine and W. Li, *Applied Nonlinear Control*. Englewood Cliffs, NJ: Prentice-Hall Inc., 1991.

[2] V. Utkin and J. Shi, "Integral Sliding Mode in Systems Operating under Uncertainty Conditions," *Proceedings of the 35th Conference on Decision and Control*, Kobe, Japan, pp. 4591-4596, 1996.

[3] L. Fridman, J. Moreno, and R. Iriarte, "Sliding Modes after the First Decade of the 21st Century: State of the Art," *Lecture Notes in Control and Information Sciences*, vol. 412, 2011.

[4] D. K. Young, V. I. Utkin, and U. Ozguner, "A Control Engineer's Guide to Sliding Mode Control," *IEEE Transactions on Control Systems Technology*, vol. 7, no. 3, pp. 328-342, 1999.

[5] A. Villanueva, B. Castillo-Toledo, E. Bayro-Corrochano, L. F. Luque-Vega, and L. E. Gonzalez-Jimenez, "Multi-Mode Flight Sliding Mode Control System for a Quadrotor," *International Conference on Unmanned Aircraft Systems*, pp. 861-870, 2015.

[6] E. Abdulhamitbilal, "Robust Flight Sliding Modes Control System Design for Nonlinear Aircraft with Parameter Uncertainties," *13th IEEE Workshop on Variable Structure Systems*, Nantes, France, 2014.

[7] K. Yang, Y. Kang, and S. Sukkarieh, "Adaptive Nonlinear Model Predictive Path-Following COntrol for a Fixed-wing Unmanned Aerial Vehicle," *International Journal of Control, Automation, and Systems*, vol. 11, no. 1, pp. 65-74, 2013.

[8] M. Norton, S. Khoo, A. Kouzani, and A. Stojcevski, "Adaptive Fuzzy MultiSurface Sliding Control of Multiple-Input and Multiple-Output Autonomous Flight Systems," *IET Control Theory and Applications*, vol. 9, no. 4, pp. 587-597, 2014.

[9] A. Brezoescu, R. Lozano, and P. Castillo, "Lyapunov-Based Trajectory Tracking Controller for a Fixed-Wing Unmanned Aerial Vehicle in the Presence of Wind," *International Journal of Adaptive Control and Signal Processing*, vol. 29, pp. 372-384, 2014.

[10] H. Bouadi, H. Wu, and F. Mora-Camino, "Flight Path Tracking Based on Direct Adaptive Sliding Mode Control," *2011 IEEE Intelligent Vehicles Symposium (IV)*, Baden-Baden, Germany, pp. 25-30, 2011.

[11] A. Crassidis, and A. Mizov, "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, Canada, no. 166, pp. 1-9, 2015.

[12] R. M. Reis, "A New Model-Free Sliding Mode Control Method with Estimation of Control Input Error" Thesis, Dept. Elect. Eng., Rochester Institute of Technology, 2016.

[13] F. El Tin, "A Model-Free Control System Based on the Sliding Mode Control Method with Applications to Multi-Input Multi-Output Systems," Thesis, Dept. Mech. Eng., Rochester Institute of Technology, 2017.

[14] R. Raygosa-Barahona, V. Parra-Vega, E. Olgiun-Diaz, and L. Munoz-Ubando, "A Model-Free Backstepping with Integral Sliding Mode Control of Underactuated ROVs," *International Conference on Electrical Engineering Computing Science and Automatic Control*, Merida City,Mexico, pp. 1-7, 2011.

[15] A.-J. Munoz-Vazquez, V. Parra-Vega, A. Sanchez, and H. Ramirez-Rodriguez, "A Passive Velocity Field for Navigation of Quadrotors with Model-Free Integral Sliding Mode Control," *2013 International Conference on Unmanned Aircraft Systems*, pp. 4958, 2013.

[16] T. Salgado-Jimenez, L. G. Garcia-Valdovnos, and G. Delgado-Ramirez, "Depth Control of a 1 DOF Underwater System Using a Model-Free High Order Sliding Mode Control," *2010 Electronics, Robotics and Automotive Mechanics Conference*, pp. 481487, 2010.
[17] R. E. Precup, M. B. Radac, R. C. Roman, and E. M. Petriu, "Model-Free Sliding Mode Control of Nonlinear Systems: Algorithms and Experiments," *Information Sciences*, vol. 381, pp. 176-192, 2017.
[18] A. Levant, "Universal SIngle-Input-Single-Output (SISO) Sliding-Mode Controllers with Finite-Time Convergence," *IEEE Transactions on Automatic Control*, vol. 46, no. 9, pp.1447-1451, 2001.
[19] V. I. Utkin, "Variable Structure Systems with Sliding Modes," *IEEE Transactions on Automatic Control*, vol. AC-22, no. 2, pp. 212-222, 1977.
[20] V. I. Utkin, "Discussion Aspects of High-Order Sliding Mode Control," *IEEE Transactions on Automatic Control*, vol. 61, no. 3, pp. 829-833, 2016.
[21] L. Derafa, A. Benallengue, and L. Fridman, "Super Twisting Control Algorithm for the Attitude Tracking of a Four Rotors UAV," *Journal of the Franklin Institute*, vol. 349, pp. 685-699, 2012.
[22] AR.Drone 2.0 Support from Embedded Coder, (2017, November 30), MathWorks. [Online]. Available: https://www.mathworks.com/hardware-support/ar-drone.html
[23] N. Michael, (2017, November 30), "Quadrotor Modeling and Control," Lecture, (Carnegie Mellon Univ., 2014). [Online]. Available: http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/16311/www/s15/syllabus/ppp/Lec08-Control3.pdf
[24] A. Levant, "Robust Exact Differentiation via Sliding Mode Technique," *Automatica*, vol. 34, no. 3, pp. 379-384, 1998.
[25] MBC150 6-axis eCompass data sheet, (2017, November 30) Bosch Sensortec (Rev 1.4, 2014), [Online]. Available: http://www.mouser.com/ds/2/783/BST-BMC150-DS000-04-786477.pdf
[26] IMU-3000 Motion Processing Unit Product Specification, (2017, November 30,) InvenSense (Rev 1.0, 2010), [Online]. Available: https://store.invensense.com/datasheets/invensense/PS-IMU-3000A.pdf