8-1-2023

Model-Free Sliding Mode Control in the Lateral and Direction Dynamics of an Aircraft

Nicholas Hutson
ndh4970@rit.edu

ROCHESTER INSTITUTE OF TECHNOLOGY

Model-Free Sliding Mode Control in the Lateral and Direction Dynamics of an Aircraft

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Mechanical Engineering

NICHOLAS HUTSON

Advisor: DR. AGAMEMNON CRASSIDIS

Thesis Committee

Dr. Jason Kolodziej

Dr. Kathleen Lamkin-Kennard

Dr. Sarilyn Ivancic

08/01/2023

DEPARTMENT OF MECHANICAL ENGINEERING

KATE GLEASON COLLEGE OF ENGINEERING

Model-Free Sliding Mode Control in the Lateral and Direction Dynamics of an Aircraft

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

Committee Signatures:

_____ Date_____

Dr. Agamemnon Crassidis

Thesis Advisor

_____ Date_____

Dr. Jason Kolodziej

Thesis Committee Member

_____ Date_____

Dr. Kathleen Lamkin-Kennard

Thesis Committee Member

_____ Date_____

Dr. Sarilyn Ivancic

Department Representative

Abstract

Traditional control methods require extensive tuning or a derivation of a system model making them increasingly antiquated for use on new, more complex systems. Sliding Mode Control has emerged as a more effective alternative as a control method that can directly handle nonlinear systems with increased robustness while guaranteeing stability. However, it is still limited by the need for a system model for the derivation of the controller form. This work proposes a new model-free control method based on Sliding Mode Control referred to as Model-Free Sliding Mode Control where the form of the controller is only dependent on system order, state measurements, and previous control inputs. Lyapunov's stability theorem is used to ensure global asymptotic stability and a boundary layer is incorporated to reduce chattering. The model-free properties of the controller are enabled by a least-squares online parameter estimation method used to estimate the control input influence gain matrix of the system directly. The estimation process is essential as previous work was limited by assuming the bounds of the control input influence gain matrix are known or the assumption it was unitary. The estimation method also incorporated exponential forgetting to only include updated data for parameter estimation, increasing the speed of convergence. Another addition was a bounded gain forgetting factor to ensure that the magnitude of the control input influence gain was upper bounded. The performance of this controller was simulated on various example systems to test its performance. These included single-input, single-output and multi-input, multi-output first and second order systems. Principally, the controller was implemented to control a lateral-directional state space model of an aircraft with a shaped input characterizing aircraft roll and yaw dynamics. The controller proved to exhibit outstanding tracking performance, convergence of estimated parameters, smooth and acceptable control input, and increased robustness to parameter uncertainty. Therefore, the controller was proven to be a feasible method to control the lateral and directional dynamics of an aircraft.

TABLE OF CONTENTS

LIST OF FIGURES

NONMENCLATURE

A	State matrix
B	Control input matrix
$\boldsymbol{B}$	Input influence gain matrix
$B(t)$	Boundary layer surface
B_R	Spherical region in state-space
b	Control input influence gain
$\hat{b}$	Estimated control input gain
$\tilde{b}$	Estimation error
b_{lower}	Lower bound control input gain element
b_{upper}	Upper bound control input gain element
d/dt	Derivative in respect to time
d/dx	Derivative in respect to states
DC	Direct Current
e_1	Instantaneous prediction error
E	Estimation error bounding function
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
J	Cost function
K	Switching gain
$\overline{K}$	Updated switching gain
k	Positive integer
k_0	Upper bound for estimator gain matrix magnitude
LQR	Linear Quadratic Regulator
MFSMC	Model-Free Sliding Mode Control
MIMO	Multi-Input-Multi-Output

n	Order of system
p_0	Initial estimation gain
P	Estimator gain matrix
$\dot{P}$	Time derivate of estimator gain matrix
PD	Proportional-Derivative
PID	Proportional-Integral-Derivative
QSMC	Quasi-Sliding Mode Control
RWLS	Recursive Weighted Least Squares
s	Sliding surface
$\dot{s}$	Time derivative of sliding surface
S_R	Sphere in state-space
sat	Saturation function
sgn	Sign function
SISO	Single-Input-Single-Output
SMC	Sliding Mode Control
t	Time
TPS	Test Pilot School
u	Control input
$\hat{u}$	Estimated control input
u_{k-1}	Previous control input
u_{k-2}	Previous value of previous control input
UV-MRAC	Unit Vector Model-Reference Sliding Mode Controller
V	Lyapunov function
$\dot{V}$	Time derivative of Lyapunov function
VISTA	Variable Stability In-flight Simulator Test Aircraft
VSS	Variable Stability System

W	Signal matrix
x	State measurement
x_d	Desired state
$\dot{x}_d$	Time derivative of desired state
$\ddot{x}_d$	Second derivative of desired state
x_e	Equilibrium point
x_p	System output vector
$\dot{x}$	Time derivative of state measurement
$\ddot{x}$	Second derivative of state measurement
$\tilde{x}$	Difference between state measurement and desired state
x_0	Initial value
x^n	Higher order state
y	System output vector
α_1	Positive Constant
ε	Estimation error in input influence gain
$\hat{\varepsilon}$	Estimated error in input influence gain estimation
σ_l	Lower bound of estimation error
σ_u	Upper bound of estimation error
β	Control input gain auxiliary variable
δ	Positive onstant
η	Small positive constant
ϕ	Boundary layer
$\dot{\phi}$	Time derivative of boundary layer
λ	Slope of sliding surface
λ_{min}	Smallest eigenvalue
λ_0	Forgetting factor

$\lambda(t)$	Time-varying forgetting factor

1.0 PROBLEM INTRODUCTION

Interest in the advancement of control system design has proliferated as systems have become more complex. Technological development has allowed for the use of much smaller and faster computers which have made the use of advanced control systems feasible for a wide variety of applications. Control system design is focused on tracking a desired response while maintaining the stability of a system. Two popular traditional controllers are the Proportional-Integral-Derivative (PID) and Linear Quadratic Regulator (LQR) controllers. PID controller design consists of time and frequency domain analysis to derive the controller gains achieving the desired response from the system. LQR controller design consists of defining a cost function based on error in desired outputs. The cost function is minimized to derive the controller gains. These controllers can often be highly effective for many classes of linear systems. However, they often require extensive tuning. Additionally, for nonlinear systems, they require an accurate linear approximation of the system's dynamics. This can be difficult to derive for higher-order nonlinear systems, especially with modeling uncertainties. Thus, these controllers are not applicable to many modern systems that are becoming more complex and nonlinear.

Nonlinear control systems such as Sliding Mode Control (SMC) have been developed as more effective alternatives for when an accurate linear approximation of the system's dynamics cannot be derived in higher-order nonlinear systems. SMC provides a greater robustness in the face of modeling uncertainty and external disturbances and theoretically has perfect tracking performance. This is achieved by defining a "sliding surface" as the difference between current state values and their corresponding desired values. A SMC controller is then designed to switch between two phases: a reaching phase and a sliding phase. In the reaching phase, the controller drives the system states onto the sliding surface. In the sliding phase, the controller pushes the system states to slide towards the stable equilibrium point at the origin. Global asymptotic stability throughout the process is guaranteed through Lyapunov's Direct Method. SMC still has inherent drawbacks. Chattering causes high controller effort and is caused by high frequency switching in the controller when the state trajectories are not being perfectly aligned on the sliding surface. The "switching" is caused by a discontinuous term in the control law dependent on whether the state trajectories are above or below the sliding surface. Traditional SMC also requires a mathematical reference or plant model for the derivation of the control law. In modern times, it is increasingly difficult to derive a mathematical model of systems as they become more complex leading to higher parametric uncertainty.

Thus, a huge benefit is realized in the development a controller that is not dependent on a system model and can be generalized to all systems. Reis and Crassidis [1] developed a Model-Free Sliding Mode Controller (MFSMC) for Single-Input-Single-Output (SISO) systems. El Tin and Crassidis [2] expanded the previous work to apply the controller to squared and non-squared Multi-Input-Multi-Output (MIMO) systems. However, for the derivation of both of these controllers, the bounds of the input influence gain matrix were assumed to be known so they were not truly model-free. Sariful and Crassidis [3] proposed a method extending the MFSMC

approach by including a least-squares online parameter estimation law to estimate the increment to the switching gain in a time-varying boundary layer. The updated method only required knowledge of the system order, state measurements, and the previous control inputs making it truly model-free. However, this method assumed a unitary input influence gain which is not true for most systems. The flaw became evident during flight testing with the Air Force Test Pilot School (TPS). Stephens et al. [4] implemented this controller method in the longitudinal axis for pitch rate control of the Calspan Variable Stability System (VSS) Learjet. The controller was able to push the aircraft dynamics towards the desired dynamics but degraded handling qualities were observed.

The method described here is intended to solve the gaps outlined in Stephens et al. [4] while the MFSMC was being flight tested. The new approach uses the least-squares online parameter estimation method proposed by authors Sariful and Crassidis [3] to estimate the control input influence gain in real-time in place of the increment to the switching gain. The convergence of the input influence gain is guaranteed for any defined initial value. Thus, this method is completely model-free. This method was implemented on SISO and MIMO example systems, before being implemented in the lateral and directional control of a state space model of an aircraft.

2.0 LITERATURE REVIEW

2.1 Control Systems based on Sliding Mode Control

Yu et al. [5] developed an Adaptive Seeking Sliding Mode controller for a class of nonlinear systems with modeling uncertainty and external disturbances. The controller retained the advantageous properties of traditional SMC like robustness against parameter uncertainty and external disturbances while applying a floating gain to lessen chattering from continuous and slow-varying unknown external disturbances. The controller can also adaptively estimate and compensate for parameter uncertainty and unknown external disturbances. The performance of the controller was tested experimentally as an adaptive cruise control system for off-road vehicles in terms of velocity tracking. The results of these experiments showed minor tracking errors and chattering. The work was limited by the fact that a mathematical model was assumed to derive the control law and the controller was only derived and applied in continuous time.

Lee [6] proposed a discrete-time SMC law that implements fast output sampling. Fast output sampling has the advantages of an equivalently fast output sampling feedback control and allowing for arbitrary SMC design. In other words, the eigenvalues of the system can be arbitrarily defined. This control law also implemented a boundary layer for reduced chatter. The performance of this controller was simulated in MATLAB/Simulink on an example system with lightly damped resonance and a discrete-time controller. The results showed that the controller effectively tracked a step response with the presence of modeling uncertainty. The work was limited by the need for a plant model to derive the SMC law.

Pai [7] developed a discrete-time integral sliding mode controller for linear systems with uncertain parameters to track the dynamic inputs of a reference model without delays. This controller was derived using discrete-time SMC theory and one-step delayed disturbance approximation. The control law introduced an integral switching surface which eliminated the reaching phase and therefore, chattering. The proposed controller guaranteed stability of a closed-loop system and achieved zero-tracking error in the presence of parameter uncertainties and external disturbances. This was shown by simulating the controller performance in MATLAB and comparing it to a traditional SMC. The traditional SMC had a higher control input and the robustness of the proposed controller was also guaranteed for all time unlike the traditional controller. The work was limited by the fact the SMC algorithm was derived from a reference system model, it was limited to linear systems, and the lower bound of uncertain parameters was assumed to be known.

Cunha et al. [8] developed a SMC for the purpose of accurately and robustly tracking the outputs of linear multivariable systems with a relative degree of one. The controller ensured the closed loop system was globally exponentially stable and is robust against bounded input disturbances and parameter uncertainties. The controller was defined by the authors as a unit vector model-reference sliding mode controller (UV-MRAC). The advantage of the controller is only information about the plant model needed to derive the UV-MRAC is the knowledge of the

matrix that the high frequency gain matrix will be influencing such that the negative of the two matrices multiplied together is Hurwitz stable. This allows the positive definiteness criteria needed by other SMC methods to be relaxed. The performance of the controller was simulated on a third-order system. The controller forced the system outputs to converge quickly to the reference states while ensuring closed-loop system global exponential stability. However, the work was limited by the fact that a reference model was required, and it was only applied to linear systems with a relative degree of one.

Lagrouche et al. [9] developed a higher order integral SMC for nonlinear systems with modeling uncertainty. The controller was split into two parts: a sliding surface and a discontinuous control. The sliding surface controlled system trajectories and stabilized the system in finite time while there were no uncertainties. The discontinuous control was designed to force the system into the reaching phase and onto the sliding surface such that uncertainties were compensated for. The performance of the controller was simulated by applying it to the control of a kinematic model of an automobile to steer the automobile from an initial position over a specified trajectory. The results of this simulation showed that the system states converged to the desired trajectories within the desired time and this was done without chattering. The work was limited by the fact that a reference model is needed to derive the controller, the parameter uncertainty bounds must be known, and the controller is only designed for single-input, single-output (SISO) systems.

Jing et al. [10] developed a Quasi-Continuous High-Order Sliding Mode Controller for the variable speed control of a wind turbine. Sliding mode was chosen due to its high robustness to external disturbances, unmodeled dynamics, and parameter uncertainty. A higher-order (second order) sliding mode controller was derived to suppress chattering based on a linearized model of a wind turbine. This controller was applied to both variable speed and pitch control. The performance of the second order sliding mode controller was simulated for different wind conditions and compared to a PID and traditional sliding mode controller. The second order sliding mode controller had better tracking performance and allowed the simulated wind turbine to generate more power compared to the PID and traditional sliding mode controllers. It was also able to suppress the chattering associated with traditional SMC. This work is limited by the need for a plant model. The difficulty in deriving complex, nonlinear plant models is shown by the fact that the authors used a linearized model of the wind turbine.

Nizar et al. [11] developed a predictive sliding mode controller for systems with a state time-delay. The authors' goal was to overcome the difficulty in predicting time delays which are prevalent in many systems. They achieved this by developing a technique of using sliding mode control combined with model-based predictive control. A discrete-time delay system and sliding function were first defined. Then, a sliding mode control law was derived from the sliding function and a cost function was defined to apply predictive control principles to the control law. These were used to derive a discrete predictive sliding mode control law. The new control law was compared to a traditional sliding mode control law on a discrete-time system. The new

control method resulted in a faster convergence and less tracking error. However, the method was limited by the fact that this approach was only applied to a linear system.

The system model for a quadrotor is highly nonlinear and dynamically unstable. Sudhir et al. [12] developed a robust second-order sliding mode controller for altitude control of a quadrotor. This controller was derived using the Lyapunov stability approach and an assumed model for a quadrotor. The tracking of altitude by this controller was simulated in MATLAB. The controller successfully stabilized the quadrotor and when compared to a conventional sliding mode controller, it showed better transient performance. The work was limited since the controller was derived from an assumed model for a quadrotor.

2.2 Model-Free Control Methods

Crassidis and Mizov [13] developed a model-free sliding mode controller that can be applied to both linear and nonlinear systems to achieve accurate tracking and stability. The controller form was derived assuming the order of the system is known and is based on observable state measurements and past controller inputs. Therefore, a system model is not needed for the controller to work. The authors implemented a boundary layer into the controller to smooth the control effort. This reduced the tracking precision but reduced the controller effort to a desirable level. The method produced minimal tracking error for both the linear and nonlinear examples. The work was limited to single-input cases and a unitary input influence gain was assumed which is not true for many real systems.

In this work, Crassidis and Reis [1] derived a similar sliding mode controller to Crassidis and Mizov [13]. The authors extended this previous work by Crassidis and Mizov [13] to systems with a non-unitary input influence gain and noise in state measurements while maintaining that the sliding mode controller is based on a known order of the system, observable state measurements, and past controller inputs. This controller was applied to two nonlinear mass-spring-dampers systems: one without equipment and measurement noise and one with noise. The noise was generated using a Gaussian distribution using a variance, mean, and probability distribution from the system's sensor datasheet. For both examples, the controller provided great tracking and eliminated chattering with the implementation of a boundary layer. The work was limited by the fact that the bounds of the input influence gain matrix were assumed to be known so the controller was not completely a model-free control approach.

Crassidis and El Tin [2] further extended the work of Crassidis and Mizov [13] and Crassidis and Reis [1] to MIMO systems with actuator dynamics. The form of the sliding mode controller is still dependent on knowing the order of the system, observable state measurements, past controller inputs, and additionally, the shape of the control input influence gain matrix. For MIMO systems with a square input influence gain matrix, the control law was similar to that developed by Crassidis and Reis [1]. For underactuated MIMO systems, a transformation matrix was needed to square the input influence gain matrix. This forced the authors to choose to track certain outputs over others and resulted in better tracking performance for MIMO systems with

a square input influence gain matrix while only some outputs achieved good tracking for underactuated systems. The controller was first applied to a single-input nonlinear two mass-spring-damper system and achieved good tracking but with a high control effort. Next, the controller was applied to a quadcopter which is an underactuated system. Therefore, the controller achieved good tracking for some outputs, while other outputs and the control input exhibited high frequency activity. When applied to systems with an actuator-induced time delay (actuator dynamics), the results from the sliding mode controller were severely degraded, especially with higher time delays. This was a result of the control law requiring modification to account for the time delay which produced inconsistent tracking results. Again, the work was limited by the fact that the bounds of the input influence gain matrix were assumed to be known so the controller was not completely model-free.

Schulken [14] examined four model-free sliding mode controllers applied in the control of a quadrotor. This was done because mode-free sliding mode controllers have the advantage of having no tuning process like that of linear PID controllers. The model-free sliding mode controllers were simulated on different aerial system models with various disturbances including the effects of discretization and state estimation. The two best performing controllers were then applied to a quadrotor system model and their performance was compared to that of a linear PID controller in terms of tracking error, complexity, and how long it took to tune each controller. Both the model-free sliding mode controllers and the PID controller were able to display similar tracking errors in simulation. However, the PID results came after an intensive tuning process. When applied to real hardware, all control laws exhibited unsatisfactorily high tracking error during testing to be considered for a real system application. The hardware used in testing did not directly measure all states required for control so a state estimation algorithm needed to be developed. The large tracking errors in testing could be attributed to uncertainty in the state estimation algorithm. The work was limited by needing the input influence gain bounds to be determined from system models and an unsatisfactory state estimation algorithm.

Sreeraj and Raj [15] further expanded on the work of Crassidis and Mizov [13], Crassidis and Reis [1], Crassidis and El Tin [2], and Schulken [14]. The authors generalized the model-free sliding mode control law developed in past works so that it could be applied to a larger class of unmanned systems. As in past work, the control law was derived with knowledge of the order of the system, observable state measurements, past controller inputs, and additionally, the shape and bounds of the control input influence gain matrix. The controller was tested on a simulation of a quadrotor and its performance was compared to a traditional linear controller in terms of tracking error and power consumption. This simulation included realistic inputs from joysticks and inertial measurement units (IMUs) that are used for state measurement. The controller was then tested on real hardware and compared to a linear controller. The model-free sliding mode controller had smaller tracking errors and less power consumption than the linear controller. The work is still limited by the fact that the bounds of the input influence gain matrix were assumed to be known so the controller was not completely model-free.

In this work, Munoz-Vazquez et al. [16] developed a model-free integral sliding mode controller for position control of a quadrotor. The authors developed their controller with the goal of adapting velocity field control to the underactuated flight dynamics of a quadrotor. This was achieved by modifying the nominal reference to include the velocity field as the desired velocity to be tracked. The controller was designed to include three subsystems. These subsystems included a model-free control subsystem that ensured the sliding mode for all time, a velocity field subsystem for defining the magnitude and direction of the velocity field to define a desired path, and a sliding surface subsystem to assemble invariant manifolds of position and orientation sliding surfaces. Since the velocity field was used to create the sliding surface, this design ensures stability by ensuring the quadcopter stays on the passive velocity field. The work was limited by the fact that the controller is developed from velocity field control instead of directly controlling states.

Monti [17] expanded on the work of Sreeraj and Raj [15] by implementing their MFSMC law to an Unmanned Aircraft System (UAS) that is not mounted to a gimbal controlling altitude and is allowed to freely fly. The specific UAS used in this work was a quadcopter. The robustness of the developed controller to parameter uncertainties was tested through simulation. Parameters of the quadcopter were varied through 4 different configurations: original parameters, doubled mass, doubled moments of inertia, and doubled mass and moments of inertia. The performance of the MFSMC law controlling these different configurations was compared to a PID controller. MFSMC law provided better tracking performance and less control effort. This work was limited by the need to know the control input influence gain matrix. The matrix was derived using knowledge of the quadcopter system.

Crassidis and Sariful [3] expanded on the work of Crassidis and Mizov [13], Crassidis and Reis [1], Crassidis and El Tin [2], Schulken [14], Sreeraj and Raj [15], and Monti [17]. The authors also developed a truly model-free sliding mode controller where the form of the controller is based on system order, state measurements, and previous control inputs. The authors also implemented a boundary layer to eliminate high-frequency chattering. However, in an effort to eliminate the need to bound the input influence gain matrix, the authors estimated the increment to the switching gain in real time to ensure the sliding condition is being met using a least-squares estimator. This was done while assuming the input influence gain was unitary. An exponential forgetting factor was combined with the least-squares estimator to ensure that only new data was used in the parameter estimation. The authors also developed an automatic bounded forgetting tuning technique that ensures that the estimated input influence gain matrix is upper bounded. The controller was applied to linear and nonlinear SISO and MIMO systems and was implemented in the simulation of a quadcopter. The controller achieved great tracking results for all cases. The work was limited by the assumption that the input influence gain matrix was close to being unitary which is not true for most real-world systems. Thus, the control law may fail if the input influence gain is far from being unitary.

Stephens et al. [4] applied the work of Crassidis and Sariful [3] to control in the longitudinal dynamics of a Calspan VSS Learjet. This work included a limited evaluation to test the ability of the control law to track desired inputs and carry out single-ship and formation handling quality tasks to evaluate the control law. The control law was successful in producing desired short-period dynamics but added a large time delay which inhibited fine tracking. The aircraft pitch acceleration was measured from six sources: discrete derivative of pitch rate, sliding mode differentiator from pitch rate, discrete derivative of pitch rate with a lag filter, estimation of pitch rate from accelerometer and rate signals, finite differentiation with a Muller filter, and Inertial Navigation System (INS) pitch acceleration. For flight testing, pitch frequency sweeps were input through the control stick to generate the desired pitch rate. Four aircraft configurations were used to test the controller: level 1-3 based on Civil Air Patrol criteria and a statically unstable configuration. A low pass filter was required to be added before flight testing to deal with excessive noise seen in initial simulations that would disengage the VSS. Testing showed that the MFSMC was able to drive the aircraft dynamics to the desired dynamics with mixed results. Cooper-Harper handling qualities rating scale was used for the handling qualities evaluation, and it was determined that the model used to generate desired dynamics had design features that degraded the resulting handling qualities generated by the controller. The root of the problem was the assumption of a unitary input influence gain matrix. The Calspan VSS Learjet and the system model possess an input influence gain matrix much smaller than one leading to high tracking errors.

2.3 **Online Parameter Estimation**

Vahidi et al. [18] developed a recursive least squares estimation method with multiple forgetting factors for the simultaneous real-time estimation of time-varying grade and piecewise constant mass. This estimation method was based on a physical model of a road-going vehicle in the longitudinal direction. The authors ran multiple simulations that showed the estimation method could achieve good estimation precision while under persistent excitation. However, the authors failed to prove convergence and region of convergence, there was difficulty getting results without persistent excitation, and a physical model was required.

Sumantri et al. [19] developed a least square based sliding mode controller to solve the overdetermined problem of translational motion control of a quadrotor and a traditional sliding mode controller for rotational control. These controllers were derived while ensuring global stability using Lyapunov's stability theory. The authors augmented the performance of the translational controller by implementing a constant plus proportional reaching law to increase reaching rate. A saturation function with boundary layer was also used to reduce chattering. The experimental performance of the controller was compared to a traditional proportional-derivative (PD) controller with and without external disturbances. Without a disturbance, the PD controller achieved smaller tracking errors. However, the PD controller lost its robustness when a disturbance was applied while the SMC kept a similar tracking error to the non-disturbance test. The work was limited by the need for a system model for the quadrotor.

Valladolid et al. [20] proposed a discrete-time adaptive quasi-sliding mode controller (QSMC) based on a recursive weighted least square (RWLS) estimator for a DC motor. The authors redefine SMCs that use a saturation function and boundary layer to reduce chattering as a QSMC. This controller deals with the traditional problem of QSMC which is steady state error due to the use of a saturation function instead of a switching function used in traditional SMC. This is accomplished with the RWLS estimator that estimates the parameters in the A matrix of the autoregressive model, not the input influence gain matrix which is set to be constant. The adaptation law guarantees the convergence of the output of the system to the desired values to be tracked. The performance of this controller was tested experimentally by applying it to the speed control of a DC motor and comparing it to a traditional SMC and QSMC. The results of these tests showed that the RWLS-QSMC reduced chatter compared to the SMC and had smaller steady-state error compared to the traditional QSMC. The controller was also shown to be able to effectively control the speed of the DC motor in the nonlinear areas of its dynamics. The work was limited by the requirement for an initial plant model for a DC motor for the derivation of the controller.

Many real-world systems have multiple inputs and outputs. These outputs always contain interference making the perfect, noise-free outputs unmeasurable. Ding [21] developed an auxiliary model-based recursive generalized least squares algorithm and a least squares-based iterative algorithm to estimate unmeasurable parameters in the information vector in a MIMO output-error system with autoregressive noise from measurable outputs. Autoregressive models specify that the output variable depends linearly on its own previous values and on a stochastic term. Ding's algorithms were tested on two examples of MIMO output-error systems with autoregressive noise and compared against each other. The iterative algorithm had the advantage of using all measured input and output data to produce more accurate estimates. The work was limited by the fact that it is only a parameter estimation technique, not a control method.

Ma et al. [22] developed an online multi-parameter estimation method based on recursive least squares and incorporating a dynamic forgetting factor. This was done to overcome the problems with traditional recursive least squares with forgetting factor online parameter estimation. These problems include degenerate-rank, slow convergence, and low estimation accuracy. A full-rank parameter identification model was derived to establish the online estimation of multiple parameters of a permanent magnet synchronous motor. A dynamic forgetting factor was then added to improve convergence. The performance of this method was simulated in MATLAB by varying parameters of a permanent magnet synchronous motor in multiple ways including slow-varying and instantaneous variation. The estimation method showed convergence of all parameters within 0.5ms displaying fast convergence and robustness against parameters varying in multiple ways. The work was limited by the fact that it is only a parameter estimation technique, not a control method.

3.0 SLIDING MODE CONTROL

3.1 Fundamentals

3.1.1 Lyapunov Basics

The most important characteristic of a control system is if it's stable or not. A stable system starts near a desired operating state and stays close to this state for eternity. An unstable system is highly unpredictable which oftentimes is undesirable. The stability analysis used in this work is Lyapunov's direct method. With this method, stability is determined by constructing an "energy-like" function (Slotine and Li [23]) and examining the function's time variation properties.

3.1.2 Nonlinear Systems

A nonlinear dynamic system can be represented by a set of differential equations in the form of:

$$\dot{x} = f(x, u, t) \tag{3.1}$$

Where f is a $n \times 1$ nonlinear vector function, x is the $n \times 1$ state vector, u is the control input and t is time. The number of states is n which is the order of the system. A solution $x(t)$ exists for the differential equation 3.1 that corresponds to a curve referred to as the state trajectory. A control law can be defined by:

$$u = g(x, t) \tag{3.2}$$

Equation 3.1 can be generalized by plugging equation 3.2 into equation 3.1 to be:

$$\dot{x} = f(x, g(x, t), t) \tag{3.3}$$

Linear systems are considered to be a special class of nonlinear systems and take the form of:

$$\dot{x} = A(t)x \tag{3.4}$$

Where $A(t)$ is an $n \times n$ matrix.

3.1.3 Autonomous vs Non-Autonomous Systems

Depending on if the system matrix A varies with time or not, linear systems are classified as being time-varying or time-invariant. These terms are replaced by autonomous and non-autonomous in nonlinear systems. A nonlinear system is defined as autonomous if a function, f, does not depend on time such that the system state equation can be defined as:

$$\dot{x} = f(x) \tag{3.5}$$

Otherwise, the system is defined as non-autonomous. All physical systems are non-autonomous since the concept of an autonomous system is an idealized notion akin to a truly linear system. A physical system's dynamics are never truly time-invariant, but the system's properties can change so slowly that their time variation can be neglected, and the system can be treated as autonomous. The fundamental difference between autonomous and non-autonomous systems is

that the state trajectory of a non-autonomous system is independent of the initial time. The results make the analysis of the system more difficult than an autonomous system.

3.1.4 Equilibrium Points

When a system trajectory only corresponds to a single point, the point is considered an equilibrium point. Therefore, when a state x_e is an equilibrium point if after $x(t)$ is equal to x_e, it remains equal to x_e. This means that the constant vector x_e satisfies:

$$f(x_e) = 0 \tag{3.6}$$

This shows that once all the states reach the equilibrium point, the derivative of the states is zero and the states will not move away from the equilibrium point.

For a linear time-invariant system:

$$\dot{x} = Ax \tag{3.7}$$

There is a single equilibrium point at the origin if A is nonsingular. Otherwise, there is an infinite number of equilibrium points in the subspace defined by:

$$Ax = 0 \tag{3.8}$$

3.1.5 What Constitutes Stability

Consider a spherical region, B_R, defined by $\|x\| < R$ in state-space and a sphere, S_R, defined by $\|x\| = R$. The equilibrium state $x = 0$ is stable if, for any $R > 0$, there exists $r > 0$, such that if $\|x(0)\| < r$, then $\|x(t)\| < R$ for all $t \geq 0$. Otherwise, the equilibrium point is considered unstable (Slotine and Li [23]). In other words, the origin in state-space is stable if a value of $r(R)$ can be found such that if starting within region B_R, the state will remain in region B_R as time goes to infinity. This definition of stability where a system trajectory can be kept arbitrarily close to the origin by starting close enough to it is also known as Lyapunov stability.

3.1.6 Positive Definite Functions

Lyapunov's Direct Method is based on the creation of scalar energy-based functions for the dynamic system called Lyapunov Functions and examining the time variation of the functions to determine stability. One property of a Lyapunov Function to analyze for stability is if a function is positive definite. A scalar, continuous Lyapunov function $V(x)$ is considered locally positive definite if $V(0) = 0$ and within region B_R (Slotine and Li [23]):

$$x \neq 0 \Rightarrow V(x) > 0$$

If $V(0) = 0$ and the above property is true over the whole state space, then $V(x)$ is considered globally positive definite. $V(x)$ is considered negative definite if $-V(x)$ is positive definite, positive semi-definite if $V(0) = 0$ and $V(x) \geq 0$ for $x \neq 0$, and negative semi-definite if $-V(x)$

is positive semi-definite. The phrase "semi-definite" is used when there is a possibility that $V(x)$ can equal zero when $x \neq 0$.

Since x is a time-varying state of the system, $V(x)$ is a function of time. Therefore, $V(x)$ is differentiable with respect to time to analyze the positive-definiteness of the function:

$$\dot{V} = \frac{dV(x)}{dt} = \frac{dV}{dx}\dot{x} = \frac{dV}{dx}f(x) \tag{3.9}$$

If, in the region B_R, the function $V(x)$ is positive definite, has continuous partial derivatives, and the time derivative of $V(x)$ is negative semi-definite such that $\dot{V}(x) \leq 0$, then $V(x)$ can be labeled as a Lyapunov function for the system. The geometric representation is shown in Figures 1 and 2.

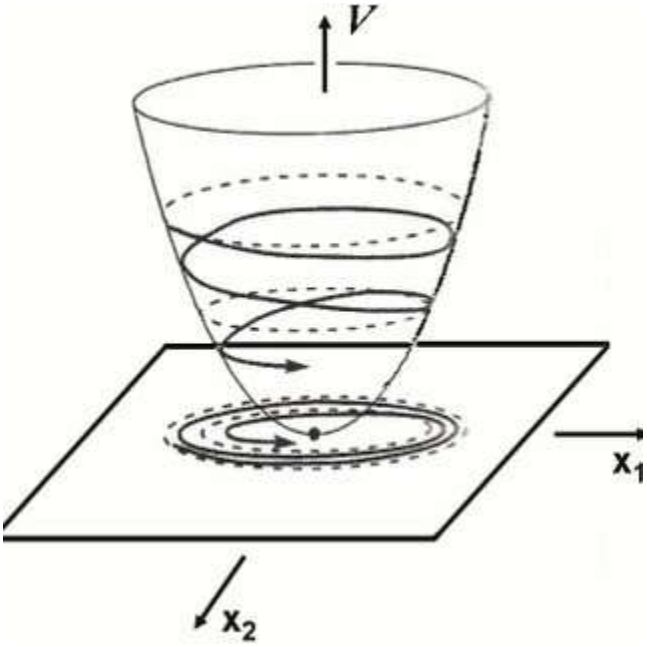

Figure 1: Geometric Representation of a Lyapunov Function

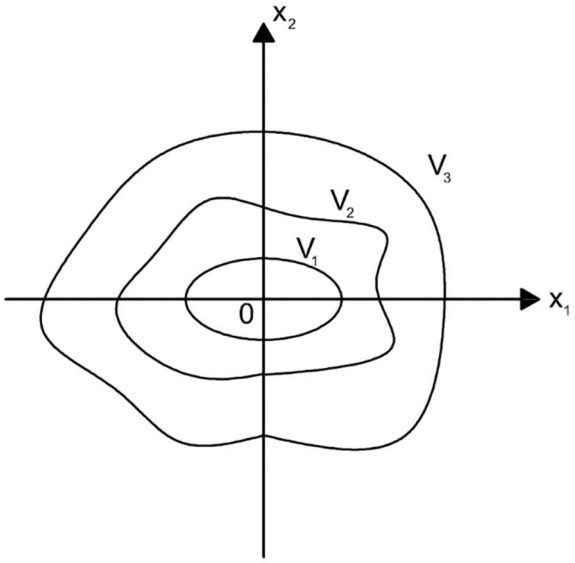

Figure 2: Contour Plot Representation of a Lyapunov Function

3.1.7 Lyapunov's Direct Method

The fundamental philosophy of Lyapunov's Direct Method is that if the total energy stored in a system is being continuously dissipated, the system state trajectory must converge to an equilibrium point. Slotine and Li [23] split Lyapunov's Direct Method into two theorems: local stability and global stability. Local stability is concerned with the stability of the system arbitrarily close to equilibrium points. For local stability, if, in a region B_R, there exists a continuously differentiable scalar function $V(x)$ such that $V(x)$ is positive definite and $\dot{V}$(x) is negative semi-definite locally in B_R, then the equilibrium point $x = 0$ is stable. If $\dot{V}$(x) is locally negative definite in B_R, then the equilibrium point is asymptotically stable. To prove the global asymptotic stability of a system, the spherical region, B_R, must be expanded to encompass the

whole state space. However, this is necessary, but not sufficient. The function V must also be radially unbounded. This is defined as:

$$V(x) \to \infty \text{ as } ||x|| \to \infty$$

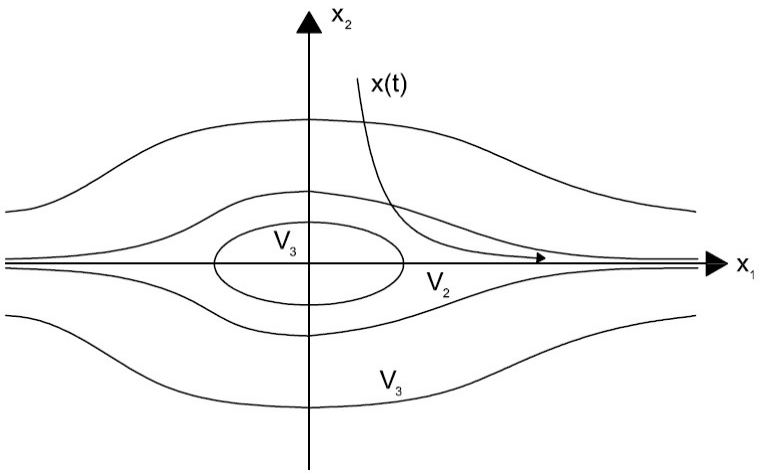

Figure 3: Geometric Representation of Radial Unboundedness

Therefore, a system is globally asymptotically stable if there exists a scalar function $V(x)$ that is continuously differentiable such that $V(x)$ is positive definite, $\dot{V}(x)$ is negative definite, and $V(x)$ is radially unbounded.

3.1.8 Sliding Mode Control

Sliding mode control is based on the principle that it is easier to control first-order systems than to control general higher order systems. Therefore, the core of sliding mode control is the replacement of an n^{th}-order problem with an equivalent first-order problem. This is done in two phases by the control law. First, there is the reaching phase where the control system forces the system states onto a stable and convergent state trajectory or sliding surface. Second, there is the sliding phase where the states are kept on the sliding surface by the control law until they reach equilibrium. These phases are shown geometrically in Figure 4 below.

3.1.9 Derivation of SMC Control Law

Consider the SISO system (Slotine and Li [23]):

$$x^{(n)} = f(x) + b(x)u \tag{3.10}$$

Where x is a scalar output of interest, u is a scalar control input, $f(x)$ is a general function that is not exactly known, $b(x)$ is the control input influence gain, and n is the order of the system. The state vector is $\left[x \ \dot{x} \ \ldots \ x^{(n-1)}\right]^T$. For the tracking to be achieved with a finite control u, the following condition must be met:

$$x_d(0) = x(0) \tag{3.11}$$

The tracking error in x is defined as:

$$\tilde{x} = x - x_d \tag{3.12}$$

We shall also define a time-varying, sliding surface s(t) in the state space such that:

$$s = (\frac{d}{dt} + \lambda)^{n-1}\tilde{x} \tag{3.13}$$

Where λ is a positive constant and the slope of the sliding surface. With initial condition 3.11, the tracking problem becomes a problem of keeping the state trajectories on the sliding surface or keeping the scalar value of s equal to zero for all time. Thus, the tracking has been simplified to be a first-order problem of keeping s equal to zero. The inequality to achieve this is defined as:

$$\frac{1}{2}\frac{d}{dt}s^2 \leq -\eta|s| \tag{3.14}$$

Where η is a positive constant. This inequality is known as the sliding condition and constrains the squared "distance" to the surface $s(t)$, s^2, to decrease along all system trajectories. Satisfying the sliding condition ensures asymptotic stability as it ensures the criteria for Lyapunov stability are also met at the same time.

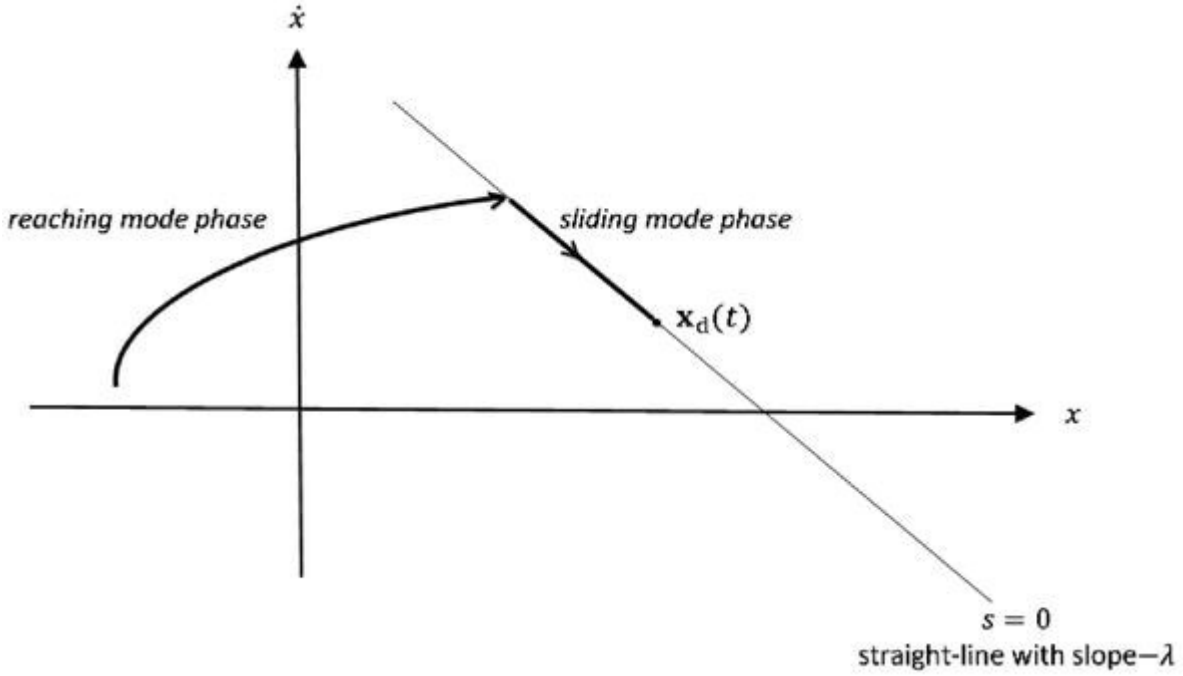

Figure 4: Geometric Representation of
Sliding Mode Control

In order to satisfy the sliding condition even with uncertainty in $\hat{u}$, a term discontinuous across $s(t)$ is added to the control law such that:

$$u = \hat{u} - Ksgn(s) \tag{3.15}$$

Where K is the switching gain and sgn is the sign function defined as:

$$sgn(s) = +1 \quad if\ s > 0 \tag{3.16}$$

$$sgn(s) = -1 \quad if\ s < 0 \tag{3.17}$$

In practice, the implementation of this control law is not perfect. This is because the switching by the sign function is not instant, and the exact value of s is not known. This leads to what is known as chattering which is when the states jump back and forth across the sliding surface as shown in Figure 5.

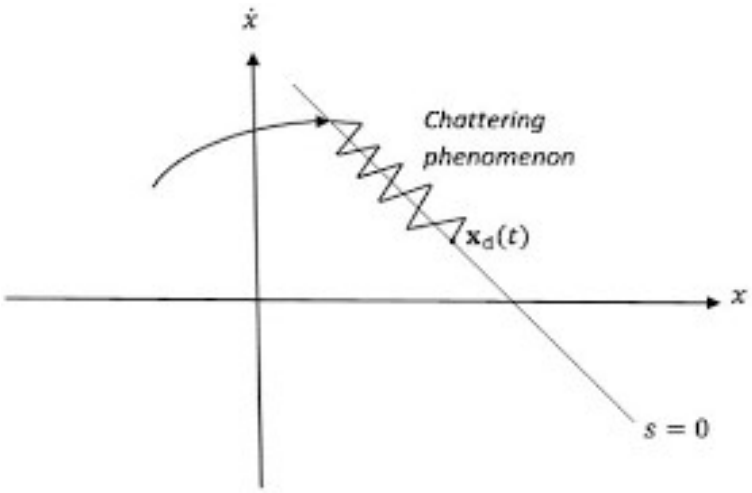

Figure 5: Chattering Phenomenon

Chattering leads to undesirably high controller output and can excite high frequency dynamics in the system. To prevent this, the controller output can be smoothed with a so-called boundary layer.

3.1.10 Boundary Layer

For a controller to work effectively, chattering oftentimes has to be eliminated. Using a boundary layer of thickness ϕ, a neighboring switching surface to the sliding surface can be defined as (Slotine and Li [23]):

$$B(t) = \{x, |s(x;t)| \leq \phi\} \tag{3.18}$$

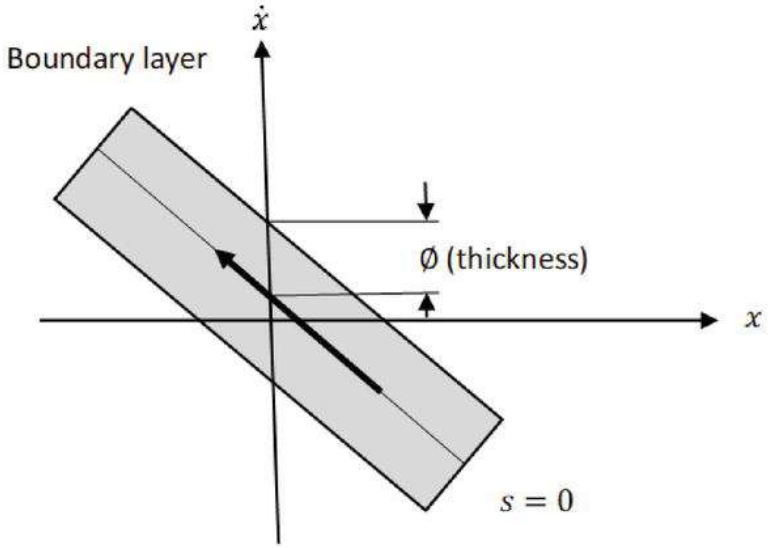

Figure 6: The Boundary Layer

The boundary layer thickness can also be made time-varying. To account for this, the sliding condition must be updated to maintain the attractiveness of the boundary layer:

$$|s| \geq \phi \rightarrow \frac{d}{dt}(s - \phi) \leq -\eta \tag{3.19}$$

$$|s| \leq \phi \rightarrow \frac{d}{dt}(s - (-\phi)) \geq -\eta \tag{3.20}$$

Or, combining these equations:

$$|s| \geq \phi \rightarrow \frac{1}{2}\frac{d}{dt}s^2 \leq (\dot{\phi} - \eta)|s| \tag{3.21}$$

The addition of the $\dot{\phi}|s|$ term defines the fact that the boundary layer attraction condition is stricter while the boundary layer is shrinking ($\dot{\phi} < 0$) and less strict while it is expanding ($\dot{\phi} > 0$).

Consider the system defined by:

$$\ddot{x} = f + bu \tag{3.22}$$

In addition to updating the sliding condition, if the input influence gain, b, is assumed to be a unity gain, a new switching gain is defined as:

$$\overline{K} = K - \dot{\phi} \tag{3.23}$$

And the control law is updated to be:

$$u = \hat{u} - \overline{K} sat\left(\frac{s}{\phi}\right) \tag{3.24}$$

Where sat is the saturation function defined as:

$$sat\left(\frac{s}{\phi}\right) = \frac{s}{\phi} \qquad if\ \left|\frac{s}{\phi}\right| \leq 1 \tag{3.25}$$

$$sat\left(\frac{s}{\phi}\right) = sgn\left(\frac{s}{\phi}\right)\ otherwise \tag{3.26}$$

Furthermore, the boundary layer thickness ϕ can be tuned so that the state trajectories represent a first-order filter by defining:

$$\dot{\phi} + \lambda\phi = K(x_d) \tag{3.27}$$

Where λ is the bandwidth of the filter such that:

$$\frac{\overline{K}(x_d)}{\phi} = \lambda \tag{3.28}$$

If $b \neq 1$ in Equation 3.22, the sliding condition can be met by ensuring $(\dot{\phi} - \eta) \leq 0$. This is ensured when the states are outside the boundary layer, or $\dot{\phi} > 0$, by defining the switching gain as:

$$\overline{K}(x) = K(x) - \frac{\dot{\phi}}{\beta} \tag{3.29}$$

Where:

$$\hat{b} = \sqrt{b_{low} b_{upp}} \tag{3.30}$$

$$\beta = \sqrt{b_{upp}/b_{low}} = \hat{b}b^{-1} \tag{3.31}$$

With $\hat{b}$ being the best estimate of the input influence gain and b_{low} and b_{upp} being the lower and upper estimated bounds of the input influence gain respectively. The sliding surface for a second-order system is defined as:

$$s = \dot{x} - \dot{x}_d + \lambda(x - x_d) \tag{3.32}$$

With the derivative of the sliding surface being defined as:

$$\dot{s} = \ddot{x} - \ddot{x}_d + \lambda(\dot{x} - \dot{x}_d) = 0 \tag{3.33}$$

Or:

$$\dot{s} = \ddot{x} - \ddot{x}_d + \lambda\dot{\tilde{x}} = 0 \tag{3.34}$$

Substituting Equation 3.22 into Equation 3.34 gives us:

$$\dot{s} = f + bu - \ddot{x}_d + \lambda\dot{\tilde{x}} = 0 \tag{3.35}$$

This allows us to define the best estimate of the controller as:

$$\hat{u} = \hat{b}^{-1}(-\hat{f} + \ddot{x}_d - \lambda\dot{\tilde{x}}) \tag{3.36}$$

Substituting this into Equation 3.24 gives us the actual control law of:

$$u = \hat{b}^{-1}\left[-\hat{f} + \ddot{x}_d - \lambda\dot{\tilde{x}} - K(x)sgn(s)\right] \tag{3.37}$$

To check if the controller form is correct so we can implement a boundary layer, Lyapunov's stability theorem is used. A positive definite candidate Lyapunov Function is defined as:

$$V = \frac{1}{2}s^2 \geq 0 \tag{3.38}$$

Taking the derivative gives us:

$$\dot{V} = s\dot{s} \leq 0 \tag{3.39}$$

Substituting Equation 3.34:

$$\dot{V} = s\left[\ddot{x} - \ddot{x}_d + \lambda\dot{\tilde{x}}\right] \tag{3.40}$$

Substituting the assumed measurement model based on Equation 3.22:

$$\dot{V} = s\left[f + \hat{b}u - \ddot{x}_d + \lambda\dot{\tilde{x}}\right] \tag{3.41}$$

Substituting Equation 3.37:

$$\dot{V} = s\left[\hat{f} + \hat{b}\hat{b}^{-1}\left[-\hat{f} + \ddot{x}_d - \lambda\dot{\tilde{x}} - K(x)sgn(s)\right] - \ddot{x}_d + \lambda\dot{\tilde{x}}\right] \tag{3.42}$$

Or:

$$\dot{V} = -sK(x)sgn(s)) \tag{3.43}$$

Or:

$$\dot{V} = -K(x)|s| \tag{3.44}$$

Therefore,

$$\dot{V} \leq (\dot{\phi} - \eta)|s| \tag{3.45}$$

This verifies the sliding condition. Therefore, the controller form is correct and, to implement a boundary layer, $Ksgn(s)$ is replaced with $\bar{K}sat(\frac{s}{\phi})$:

$$u = \hat{b}^{-1}\left[-\hat{f} + \ddot{x}_d - \lambda\dot{\tilde{x}} - \bar{K}sat(\frac{s}{\phi})\right] \tag{3.46}$$

Plugging into Equation 3.35 gives us:

$$\dot{s} = f + b\hat{b}^{-1}\left[-\hat{f} + \ddot{x}_d - \lambda\dot{\tilde{x}} - \overline{K}sat(\frac{s}{\phi})\right] - \ddot{x}_d + \lambda\dot{\tilde{x}} \tag{3.47}$$

Re-arranging this results in:

$$\dot{s} = -b\hat{b}^{-1}\overline{K}sat(\frac{s}{\phi}) + (f - b\hat{b}^{-1}\hat{f}) + (1 - b\hat{b}^{-1})(-\ddot{x}_d + \lambda\dot{\tilde{x}}) \tag{3.48}$$

Using Equation 3.31, we can re-write this as:

$$\dot{s} = -\beta^{-1}\overline{K}sat(\frac{s}{\phi}) + (f - \beta^{-1}\hat{f}) + (1 - \beta^{-1})(-\ddot{x}_d + \lambda\dot{\tilde{x}}) \tag{3.49}$$

Inside the boundary layer, we can set $sat\left(\frac{s}{\phi}\right) = \frac{s}{\phi}$, giving us:

$$\dot{s} = -\beta^{-1}\overline{K}(\frac{s}{\phi}) + (f - \beta^{-1}\hat{f}) + (1 - \beta^{-1})(-\ddot{x}_d + \lambda\dot{\tilde{x}}) \tag{3.50}$$

Assuming the states are inside the boundary layer, we can set $\vec{x} = \overrightarrow{x_d} + \vec{\varepsilon}$, where ε is a small error term, such that:

$$\dot{s} = -\beta^{-1}\overline{K}(x_d)\left(\frac{s}{\phi}\right) + \left(f(x_d) - \beta^{-1}\hat{f}(x_d)\right) - \left(1 - \beta^{-1}\right)\ddot{x}_d + O(\varepsilon) \tag{3.51}$$

Equation 3.51 is a first-order filter that can be tuned such that:

$$\dot{s} + \frac{\lambda}{\beta^2}s = u \tag{3.52}$$

Comparing the two equations gives us:

$$\frac{\lambda}{\beta^2} = \frac{\overline{K}(x_d)}{\beta\phi} \tag{3.53}$$

Plugging in Equation 3.29 gives us:

$$\frac{\lambda}{\beta^2} = \frac{K(x_d) - \frac{\dot{\phi}}{\beta}}{\beta\phi} \tag{3.54}$$

Re-writing gives us:

$$\dot{\phi} + \lambda\phi = \beta K(x_d) \tag{3.55}$$

To ensure that the sliding condition is met when the states are inside the boundary layer, or $\dot{\phi} < 0$, the switching gain, $\overline{K}$, is defined as:

$$\overline{K}(x_d) = K(x_d) - \beta\dot{\phi} \tag{3.56}$$

Plugging into Equation 3.53, we get:

$$\frac{\lambda}{\beta^2} = \frac{K(x_d) - \beta\dot{\phi}}{\beta\phi} \tag{3.57}$$

Re-writing gives us:

$$\dot{\phi} + \frac{\lambda}{\beta^2}\phi = \frac{K(x_d)}{\beta} \tag{3.58}$$

With $\dot{\phi} \geq 0$ from Equation 3.55 we get:

$$\dot{\phi} = \beta K(x_d) - \lambda\phi \geq 0 \tag{3.59}$$

Or:

$$K(x_d) \geq \frac{\lambda\phi}{\beta} \tag{3.60}$$

With $\dot{\phi} \leq 0$ from Equation 3.58, we get:

$$\dot{\phi} = \frac{K(x_d)}{\beta} - \frac{\lambda}{\beta^2}\phi \leq 0 \tag{3.61}$$

Or:

$$K(x_d) \leq \frac{\lambda\phi}{\beta} \tag{3.62}$$

Assuming $\vec{x}(0) = \overrightarrow{x_d}(0)$, we can write:

$$\phi(0) = \frac{\beta}{\lambda}K(x_d(0)) \tag{3.63}$$

In summary:

$$K(x_d) \geq \frac{\lambda\phi}{\beta_d} \quad \dot{\phi} + \lambda\phi = \beta_d K(x_d) \quad \overline{K}(x) = K(x) - \frac{\dot{\phi}}{\beta_d}$$

$$K(x_d) \leq \frac{\lambda\phi}{\beta_d} \quad \dot{\phi} + \frac{\lambda}{\beta_d{}^2}\phi = \frac{K(x_d)}{\beta_d} \qquad \overline{K}(x) = K(x) - \beta\dot{\phi}$$

$$u = \hat{b}^{-1}\left[-\hat{f} + \ddot{x}_d - \lambda\dot{\tilde{x}} - \overline{K}sat(\frac{S}{\phi})\right]$$

This is the derivation for a SISO system. For a MIMO system, the sliding condition becomes:

$$s_i\dot{s}_i \leq (\dot{\phi}_i - \eta_i)|s_i| \qquad i = 1, \ldots, m \tag{3.64}$$

Where m is the order of the system. To ensure that the sliding condition is met when the states are outside the boundary layer, or $\dot{\boldsymbol{\phi}} > 0$, the switching gain is defined as:

$$\bar{\boldsymbol{K}}(x) = \boldsymbol{K}(x) - \boldsymbol{\beta}^{-1}\dot{\boldsymbol{\phi}} \tag{3.65}$$

Where $\boldsymbol{\beta}$ is defined as:

$$\boldsymbol{\beta} = \sqrt{\boldsymbol{B_{low}}^{-1}\boldsymbol{B_{upp}}} = \widehat{\boldsymbol{B}}\boldsymbol{B}^{-1} \tag{3.66}$$

And $\boldsymbol{B_{low}}$ and $\boldsymbol{B_{upp}}$ are the lower and upper estimated bounds of the input influence gain matrix respectively. With the controller form being:

$$\boldsymbol{u} = \widehat{\boldsymbol{u}} - \bar{\boldsymbol{K}}(x)sat\left(\frac{s_i}{\phi_i}\right) \tag{3.67}$$
$$i = 1, \dots, m$$

Consider the actual system:

$$\ddot{\boldsymbol{x}} = \boldsymbol{f} + \boldsymbol{B}\boldsymbol{u} \tag{3.68}$$

Or:

$$\ddot{x}_i = f_i(x_i) + \sum_{j=1}^{m} b_{ij}(x_i)u_i \tag{3.69}$$
$$i = 1, \dots, m \quad j = 1, \dots, m$$

The input influence gain matrix, $\boldsymbol{B}$, is a square matrix defined as:

$$\boldsymbol{B} = \begin{bmatrix} b_{11} & \cdots & b_{1m} \\ \vdots & \ddots & \vdots \\ b_{m1} & \cdots & b_{mm} \end{bmatrix} \tag{3.70}$$

With the derivative of the sliding surface being defined as:

$$\dot{s}_i = \ddot{x}_i - \ddot{x}_{d_i} + \lambda\dot{\tilde{x}}_i = 0 \tag{3.71}$$
$$i = 1, \dots, m$$

Substituting Equation 3.69 into Equation 3.71 gives us:

$$\dot{s}_i = f_i - \ddot{x}_{d_i} + \lambda\dot{\tilde{x}}_i + \sum_{j=1}^{m} b_{ij}u_i = 0 \tag{3.72}$$
$$i = 1, \dots, m \quad j = 1, \dots, m$$

This allows us to define the best estimate of the controller as:

$$\widehat{\boldsymbol{u}} = \widehat{\boldsymbol{B}}^{-1}(-\hat{\boldsymbol{f}} + \ddot{\boldsymbol{x}}_{\boldsymbol{d}} - \lambda\dot{\tilde{\boldsymbol{x}}}) \tag{3.73}$$

Where $\widehat{\boldsymbol{B}}$, the best estimate of the input influence gain matrix, is defined as:

$$\widehat{\boldsymbol{B}} = \sqrt{\boldsymbol{B_{low}}\boldsymbol{B_{upp}}} \tag{3.74}$$

Substituting into Equation 3.67 gives us the actual control law of:

$$\boldsymbol{u} = \widehat{\boldsymbol{B}}^{-1}\left[-\hat{\boldsymbol{f}} + \ddot{\boldsymbol{x}}_{\boldsymbol{d}} - \lambda\dot{\tilde{\boldsymbol{x}}} - \overline{\boldsymbol{K}}(x)sat\left(\frac{s_i}{\phi_i}\right)\right] \quad i = 1, \dots, m \tag{3.75}$$

Plugging into Equation 3.72 gives us:

$$\dot{\boldsymbol{s}} = \boldsymbol{f} + \boldsymbol{B}\widehat{\boldsymbol{B}}^{-1}\left[-\hat{\boldsymbol{f}} + \ddot{\boldsymbol{x}}_{\boldsymbol{d}} - \lambda\dot{\tilde{\boldsymbol{x}}} - \overline{\boldsymbol{K}}(x)sat\left(\frac{s_i}{\phi_i}\right)\right] - \ddot{\boldsymbol{x}}_{\boldsymbol{d}} + \lambda\dot{\tilde{\boldsymbol{x}}} \quad i = 1, \dots, m \tag{3.76}$$

Re-arranging results in:

$$\dot{\boldsymbol{s}} = -\boldsymbol{B}\widehat{\boldsymbol{B}}^{-1}\overline{\boldsymbol{K}}(x)sat\left(\frac{s_i}{\phi_i}\right) + (\boldsymbol{f} - \boldsymbol{B}\widehat{\boldsymbol{B}}^{-1}\hat{\boldsymbol{f}}) + (1 - \boldsymbol{B}\widehat{\boldsymbol{B}}^{-1})(-\ddot{\boldsymbol{x}}_{\boldsymbol{d}} + \lambda\dot{\tilde{\boldsymbol{x}}}) \quad i = 1, \dots, m \tag{3.77}$$

Using Equation 3.66, we can re-write this as:

$$\dot{\boldsymbol{s}} = -\boldsymbol{\beta}^{-1}\overline{\boldsymbol{K}}(x)sat\left(\frac{s_i}{\phi_i}\right) + (\boldsymbol{f} - \boldsymbol{\beta}^{-1}\hat{\boldsymbol{f}}) + (\boldsymbol{1} - \boldsymbol{\beta}^{-1})(-\ddot{\boldsymbol{x}}_{\boldsymbol{d}} + \lambda\dot{\tilde{\boldsymbol{x}}}) \quad i = 1, \dots, m \tag{3.78}$$

Inside the boundary layer, $sat\left(\frac{s_i}{\phi_i}\right) = \frac{s_i}{\phi_i}$, so we can re-write this as:

$$\dot{s}_i = \sum_{j=1}^{m} -\left(\frac{\overline{K}_i(x_i)}{\beta_{ij}}\right)\left(\frac{s_i}{\phi_i}\right) + (f_i - {\beta_{ij}}^{-1}\hat{f}_i) + (1 - {\beta_{ij}}^{-1})(-\ddot{x}_{d_i} + \lambda\dot{\tilde{x}}_i) \quad i = 1, \dots, m \quad j = 1, \dots, m \tag{3.79}$$

Assume that inside the boundary layer, $\vec{x} = \overrightarrow{x_d} + \vec{\varepsilon}$, where ε is a small error term, such that:

$$\dot{s}_i = O_i(\varepsilon_i) + \sum_{j=1}^{m} -\left(\frac{\overline{K}_i(x_{d_i})}{\beta_{ij}}\right)\left(\frac{s_i}{\phi_i}\right) + \left(f(x_{d_i}) - {\beta_{ij}}^{-1}\hat{f}(x_{d_i})\right) - \left(1 - {\beta_{ij}}^{-1}\right)\ddot{x}_{d_i} \quad i = 1, \dots, m \quad j = 1, \dots, m \tag{3.80}$$

Equation 3.80 is a first-order filter that can be tuned such that:

$$\dot{\boldsymbol{s}} + \lambda\boldsymbol{\beta}^{-2}\boldsymbol{s} = \boldsymbol{u} \tag{3.81}$$

Comparing the two equations gives us:

$$\sum_{j=1}^{m} \lambda{\beta_{ij}}^{-2} = \sum_{j=1}^{m} \frac{\overline{K}_i(x_{d_i})}{\beta_{ij}\phi_i} \quad i = 1, \dots, m \quad j = 1, \dots, m \tag{3.82}$$

Plugging in Equation 3.65 gives us:

$$\sum_{j=1}^{m} \lambda {\beta_{ij}}^{-2} = \sum_{j=1}^{m} \frac{K_i(x_{d_i}) - {\beta_{ij}}^{-1}\dot{\phi}_i}{\beta_{ij}\phi_i} \quad (3.83)$$
$$i = 1, \dots, m \quad j = 1, \dots, m$$

Re-writing gives us:

$$\dot{\phi}_i + \lambda\phi_i = \sum_{j=1}^{m} \beta_{ij} K_i(x_{d_i}) \quad (3.84)$$
$$i = 1, \dots, m \quad j = 1, \dots, m$$

Or, in vector form:

$$\dot{\boldsymbol{\phi}} + \lambda\boldsymbol{\phi} = \boldsymbol{\beta}\boldsymbol{K}(x_d) \quad (3.85)$$

To ensure that $(\dot{\boldsymbol{\phi}} - \boldsymbol{\eta}) \leq 0$ when the states are inside the boundary layer, or $\dot{\phi} < 0$, the switching gain, $\bar{K}$, is defined as:

$$\bar{\boldsymbol{K}}(x_d) = \boldsymbol{K}(x_d) - \boldsymbol{\beta}\dot{\boldsymbol{\phi}} \quad (3.86)$$

Plugging into Equation 3.83, we get:

$$\sum_{j=1}^{m} \lambda {\beta_{ij}}^{-2} = \sum_{j=1}^{m} \frac{K_i(x_{d_i}) - \beta_{ij}\dot{\phi}_i}{\beta_{ij}\phi_i} \quad (3.87)$$
$$i = 1, \dots, m \quad j = 1, \dots, m$$

Re-writing gives us:

$$\dot{\phi}_i + \sum_{j=1}^{m} \lambda {\beta_{ij}}^{-2} \phi_i = \sum_{j=1}^{m} \frac{K_i(x_{d_i})}{\beta_{ij}} \quad (3.88)$$
$$i = 1, \dots, m \quad j = 1, \dots, m$$

Or, in vector form:

$$\dot{\boldsymbol{\phi}} + \lambda\boldsymbol{\beta}^{-2}\boldsymbol{\phi} = \boldsymbol{\beta}^{-1}\boldsymbol{K}(x_d) \quad (3.89)$$

With $\dot{\boldsymbol{\phi}} \geq 0$ from Equation 3.85, we get:

$$\dot{\boldsymbol{\phi}} = \boldsymbol{\beta}\boldsymbol{K}(x_d) - \lambda\boldsymbol{\phi} \geq 0 \quad (3.90)$$

Or:

$$\boldsymbol{K}(x_d) \geq \lambda\boldsymbol{\beta}^{-1}\boldsymbol{\phi} \quad (3.91)$$

With $\dot{\phi} \leq 0$ from Equation 3.89, we get:

$$\dot{\boldsymbol{\phi}} = \boldsymbol{\beta}^{-1}K(x_d) - \lambda\boldsymbol{\beta}^{-2}\phi \leq 0 \tag{3.92}$$

Or:

$$\boldsymbol{K}(x_d) \leq \lambda\boldsymbol{\beta}^{-1}\boldsymbol{\phi} \tag{3.93}$$

Assuming $\vec{x}(0) = \overrightarrow{x_d}(0)$, we find:

$$\boldsymbol{\phi}(0) = \frac{1}{\lambda}\boldsymbol{\beta}\boldsymbol{K}(x_d(0)) \tag{3.94}$$

In summary:

$$\boldsymbol{K}(x_d) \geq \lambda{\boldsymbol{\beta}_{\boldsymbol{d}}}^{-1}\boldsymbol{\phi} \qquad \dot{\boldsymbol{\phi}} + \lambda\boldsymbol{\phi} = \boldsymbol{\beta}_{\boldsymbol{d}}\boldsymbol{K}(x_d) \quad \bar{\boldsymbol{K}}(x) = \boldsymbol{K}(x) - \boldsymbol{\beta}^{-1}\dot{\boldsymbol{\phi}}$$

$$\boldsymbol{K}(x_d) \leq \lambda{\boldsymbol{\beta}_{\boldsymbol{d}}}^{-1}\boldsymbol{\phi} \qquad \dot{\boldsymbol{\phi}} + \lambda{\boldsymbol{\beta}_{\boldsymbol{d}}}^{-2}\boldsymbol{\phi} = {\boldsymbol{\beta}_{\boldsymbol{d}}}^{-1}\boldsymbol{K}(x_d) \quad \bar{\boldsymbol{K}}(x) = \boldsymbol{K}(x) - \boldsymbol{\beta}\dot{\boldsymbol{\phi}}$$

$$\boldsymbol{u} = \widehat{\boldsymbol{B}}^{-1}\left[-\hat{\boldsymbol{f}} + \ddot{\boldsymbol{x}}_{\boldsymbol{d}} - \lambda\dot{\tilde{\boldsymbol{x}}} - \bar{\boldsymbol{K}}(x)sat\left(\frac{s_i}{\phi_i}\right)\right]$$

$$i = 1, \ldots, m$$

3.1.11 Derivation of SISO Model-Free Control Law

The control system for a second-order single-input single-output system was derived with the following steps:

The following discrete-time measurement model is assumed:

$$\ddot{x} = \ddot{x} + bu - bu_{k-1} - bu + bu_{k-1} \tag{3.95}$$

Where $\ddot{x}$ is the measured acceleration, u is the control system input, u_{k-1} is the previous value of the control system input, and b is the control input influence gain that will be estimated. This equation can be re-written as:

$$\ddot{x} = \ddot{x} + bu - bu_{k-1} + \varepsilon(u) \tag{3.96}$$

Where $\varepsilon(u)$ is the estimation error in the input influence gain defined by:

$$\varepsilon(u) = b(u_{k-1} - u) \tag{3.97}$$

$\varepsilon(u)$ is assumed to be bounded by a known function, E, such that:

$$|\hat{\varepsilon}(u) - \varepsilon(u)| \leq E \tag{3.98}$$

Where $\hat{\varepsilon}(u)$ is the estimated error in the input influence gain estimation assumed to be defined by:

$$\hat{\varepsilon}(u) = \hat{b}(u_{k-2} - u_{k-1}) \tag{3.99}$$

And the actual error is bounded by the function:

$$(1-\sigma_l)\hat{\varepsilon}(u) \leq \varepsilon(u) \leq (1+\sigma_u)\hat{\varepsilon}(u) \tag{3.100}$$

Where σ_l and σ_u are the lower and upper defined bounds.

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

$$s = \dot{x} - \dot{x}_d + \lambda(x - x_d) \tag{3.101}$$

Where x and $\dot{x}$ are the system states to be measured and x_d and $\dot{x}_d$ are the desired states to be tracked. Taking the derivative of the sliding surface results in:

$$\dot{s} = \ddot{x} - \ddot{x}_d + \lambda(\dot{x} - \dot{x}_d) \tag{3.102}$$

Plugging in Equation 3.96 and setting $\dot{s}$ equal to zero ensures that the state error trajectories do not move once they reach the sliding surface and gives us:

$$\dot{s} = [\ddot{x} + bu - bu_{k-1} + \varepsilon(u)] - \ddot{x}_d + \lambda(\dot{x} - \dot{x}_d) = 0 \tag{3.103}$$

The best estimate for the control input, $\hat{u}$, to maintain $\dot{s}$ equal to zero is therefore:

$$\hat{u} = \hat{b}^{-1}[-(\ddot{x} - \ddot{x}_d) - \lambda(\dot{x} - \dot{x}_d) - \hat{\varepsilon}(u)] + u_{k-1} \tag{3.104}$$

And adding a discontinuous term to satisfy the sliding condition gives us:

$$u = \hat{b}^{-1}[-(\ddot{x} - \ddot{x}_d) - \lambda(\dot{x} - \dot{x}_d) - \hat{\varepsilon}(u) - \eta sgn(s)] + u_{k-1} \tag{3.105}$$

Where η is a positive constant. To check if the controller form is correct, Lyapunov's stability theorem is used and a positive definite "energy-like" function is defined as:

$$V = \frac{1}{2}s^2 \geq 0 \tag{3.106}$$

Taking the derivative results in:

$$\dot{V} = s\dot{s} \leq 0 \tag{3.107}$$

Substituting the assumed measurement model based on Equation 3.96 gives us:

$$\dot{V} = s\left[[\ddot{x} + \hat{b}u - \hat{b}u_{k-1} + \hat{\varepsilon}(u)] - \ddot{x}_d + \lambda(\dot{x} - \dot{x}_d)\right] \tag{3.108}$$

Substituting equation 3.105 gives us:

$$\dot{V} = s\left[[\ddot{x} + \hat{b}\{\hat{b}^{-1}[-(\ddot{x} - \ddot{x}_d) - \lambda(\dot{x} - \dot{x}_d) - \hat{\varepsilon}(u) - \eta sgn(s)] + u_{k-1}\} - \hat{b}u_{k-1} + \hat{\varepsilon}(u)] - \ddot{x}_d + \lambda(\dot{x} - \dot{x}_d)\right] \tag{3.109}$$

Rearranging this equation can give us:

$$\dot{V} = s\left[\ddot{x} - (\dot{x} - \dot{x}_d) - \ddot{x}_d - \lambda(\dot{x} - \dot{x}_d) + \lambda(\dot{x} - \dot{x}_d) - \hat{\varepsilon}(u) + \hat{\varepsilon}(u) - \eta sgn(s) + \hat{b}u_{k-1} - \hat{b}u_{k-1}\right] \tag{3.110}$$

Which results in:

$$\dot{V} = -s\eta sgn(s) \tag{3.111}$$

Which can be re-written as:

$$\dot{V} = -\eta|s| \tag{3.112}$$

This verifies the sliding condition. Therefore, η can be replaced with the system gain, K, in Equation 3.105 resulting in:

$$u = \hat{b}^{-1}[-(\ddot{x} - \ddot{x}_d) - \lambda(\dot{x} - \dot{x}_d) - \hat{\varepsilon}(u) - Ksgn(s)] + u_{k-1} \tag{3.113}$$

From the definition of $\dot{V}$, we can define the sliding condition as:

$$s\dot{s} \leq -\eta|s| \tag{3.114}$$

Substituting in Equation 3.103 gives us:

$$s\left[\left(\ddot{x} + bu - bu_{k-1} + \varepsilon(u)\right) - \ddot{x}_d + \lambda(\dot{x} - \dot{x}_d)\right] \leq -\eta|s| \tag{3.115}$$

Substituting in the best estimate of the control input, $\hat{u}$, for u gives us:

$$s\left[\left(\ddot{x} + b\{\hat{b}^{-1}[-(\ddot{x} - \ddot{x}_d) - \lambda(\dot{x} - \dot{x}_d) - \hat{\varepsilon}(u) - Ksgn(s)] + u_{k-1}\} - bu_{k-1} + \varepsilon(u)\right) - \ddot{x}_d + \lambda(\dot{x} - \dot{x}_d)\right] \leq -\eta|s| \tag{3.116}$$

Using the $sKsgn(s) = K|s|$ and rearranging to isolate $K|s|$ gives us:

$$K|s| \geq s\left[(\ddot{x} - \ddot{x}_d)(\hat{b}b^{-1} - 1) + \lambda(\dot{x} - \dot{x}_d)(\hat{b}b^{-1} - 1) + (\hat{b}b^{-1} - 1)\hat{\varepsilon}(u) + \hat{b}b^{-1}[\varepsilon(u) - \hat{\varepsilon}(u)]\right] + \hat{b}b^{-1}\eta|s| \tag{3.117}$$

The most conservative estimate for the upper bound of $\varepsilon(u)$ is defined as:

$$\varepsilon(u) = (1 + \sigma_u)\hat{\varepsilon}(u) \tag{3.118}$$

Therefore, the $\varepsilon(u) - \hat{\varepsilon}(u)$ term can be redefined as:

$$\varepsilon(u) - \hat{\varepsilon}(u) = (1 + \sigma_u)\hat{\varepsilon}(u) - \hat{\varepsilon}(u) = \hat{\varepsilon}(u) + \sigma_u\hat{\varepsilon}(u) - \hat{\varepsilon}(u) = \sigma_u\hat{\varepsilon}(u) \tag{3.119}$$

Plugging this into Equation 3.117 gives us:

$$K|s| \geq s\left[(\ddot{x} - \ddot{x}_d)(\hat{b}b^{-1} - 1) + \lambda(\dot{x} - \dot{x}_d)(\hat{b}b^{-1} - 1) + (\hat{b}b^{-1} - 1)\hat{\varepsilon}(u) + \hat{b}b^{-1}\sigma_u\hat{\varepsilon}(u)\right] + \hat{b}b^{-1}\eta|s| \tag{3.120}$$

And rearranging this gives us:

$$K|s| \geq s\left[(\ddot{x} - \ddot{x}_d)(\hat{b}b^{-1} - 1) + \lambda(\dot{x} - \dot{x}_d)(\hat{b}b^{-1} - 1) + (\hat{b}b^{-1}[1 + \sigma_u] - 1)\hat{\varepsilon}(u)\right] + \hat{b}b^{-1}\eta|s| \tag{3.121}$$

Next, we define:

$$\hat{b} = \sqrt{b_{low}b_{upp}} \tag{3.122}$$

$$\beta = \sqrt{b_{upp}/b_{low}} = \hat{b}b^{-1} \tag{3.123}$$

Plugging these into Equation 3.121 gives us:

$$K|s| \geq s[(\ddot{x} - \ddot{x}_d)(\beta - 1) + \lambda(\dot{x} - \dot{x}_d)(\beta - 1) + (\beta[1 + \sigma_u] - 1)\hat{\varepsilon}(u)] + \beta\eta|s| \tag{3.124}$$

With the final equation for the system gain being:

$$K \geq |(\beta - 1)||(\ddot{x} - \ddot{x}_d)| + |(\beta - 1)||\lambda(\dot{x} - \dot{x}_d)| + |(\beta[1 + \sigma_u] - 1)||\hat{\varepsilon}(u)| + \beta\eta \tag{3.125}$$

In summary, plugging in Equation 3.99 to Equations and 3.113 and 3.121:

$$u = \hat{b}^{-1}[-(\ddot{x} - \ddot{x}_d) - \lambda(\dot{x} - \dot{x}_d) - Ksgn(s)] + 2u_{k-1} - u_{k-2} \tag{3.126}$$

$$K = |(\beta - 1)||(\ddot{x} - \ddot{x}_d)| + |(\beta - 1)||\lambda(\dot{x} - \dot{x}_d)| + |(\beta[1 + \sigma_u] - 1)||\hat{b}(u_{k-2} - u_{k-1})| + \beta\eta \tag{3.127}$$

If including a boundary layer, we re-define the sliding condition as:

$$s\dot{s} \leq (\dot{\phi} - \eta)|s| \tag{3.128}$$

To ensure that the sliding condition is met when the states are outside the boundary layer, or $\dot{\phi} > 0$, the switching gain, $\overline{K}$, is defined as:

$$\overline{K}(x) = K(x) - \frac{\dot{\phi}}{\beta} \tag{3.129}$$

From Equation 3.113, $Ksgn(s)$ is replaced with $\overline{K}sat(\frac{s}{\phi})$:

$$u = \hat{b}^{-1}\left[-(\ddot{x} - \ddot{x}_d) - \lambda(\dot{x} - \dot{x}_d) - \hat{\varepsilon}(u) - \overline{K}sat(\frac{s}{\phi}):\right] + u_{k-1} \tag{3.130}$$

Plugging into Equation 3.103 gives us:

$$\dot{s} = \left[\ddot{x} + b\left\{\hat{b}^{-1}\left[-(\ddot{x} - \ddot{x}_d) - \lambda(\dot{x} - \dot{x}_d) - \hat{\varepsilon}(u) - \overline{K}sat(\frac{s}{\phi}):\right] + u_{k-1}\right\} - bu_{k-1} + \varepsilon(u)\right] - \ddot{x}_d + \lambda(\dot{x} - \dot{x}_d) \tag{3.131}$$

And rearranging this gives us:

$$\dot{s} = -b\hat{b}^{-1}\overline{K}sat(\frac{s}{\phi}) + (\ddot{x} - \ddot{x}_d)\left(1 - \hat{b}b^{-1}\right) + \lambda(\dot{x} - \dot{x}_d)\left(1 - \hat{b}b^{-1}\right) + \varepsilon(u) - \hat{b}b^{-1}\hat{\varepsilon}(u) \tag{3.132}$$

Using Equation 3.123, we can re-write this as:

$$\dot{s} = -\beta^{-1}\overline{K}sat\left(\frac{s}{\phi}\right) + (\ddot{x} - \ddot{x}_d)(1 - \beta^{-1}) + \lambda(\dot{x} - \dot{x}_d)(1 - \beta^{-1}) + \varepsilon(u) - \beta^{-1}\hat{\varepsilon}(u) \tag{3.133}$$

Inside the boundary layer, $sat\left(\frac{s}{\phi}\right) = \frac{s}{\phi}$:

$$\dot{s} = -\beta^{-1}\overline{K}\left(\frac{s}{\phi}\right) + (\ddot{x} - \ddot{x}_d)(1 - \beta^{-1}) + \lambda(\dot{x} - \dot{x}_d)(1 - \beta^{-1}) + \varepsilon(u) - \beta^{-1}\hat{\varepsilon}(u) \tag{3.134}$$

Assuming the states are inside the boundary layer, $\vec{x} = \overrightarrow{x_d} + \vec{\varepsilon}$, where ε is a small error term, such that:

$$\dot{s} = -\beta^{-1}\overline{K}\left(\frac{s}{\phi}\right) + \varepsilon(u(x_d)) - \beta^{-1}\hat{\varepsilon}(u(x_d)) + O(\varepsilon) \tag{3.135}$$

Equation 3.135 is a first-order filter that can be tuned such that:

$$\dot{s} + \frac{\lambda}{\beta^2}s = u \tag{3.136}$$

Comparing the two equations gives us:

$$\frac{\lambda}{\beta^2} = \frac{\overline{K}(x_d)}{\beta\phi} \tag{3.137}$$

Plugging in Equation 3.129 gives us:

$$\frac{\lambda}{\beta^2} = \frac{K(x_d) - \frac{\dot{\phi}}{\beta}}{\beta\phi} \tag{3.138}$$

Re-writing this gives us:

$$\dot{\phi} + \lambda\phi = \beta K(x_d) \tag{3.139}$$

To ensure that the sliding condition is met when the states are inside the boundary layer, or $\dot{\phi} < 0$, the switching gain, $\overline{K}$, is defined as:

$$\overline{K}(x_d) = K(x_d) - \beta\dot{\phi} \tag{3.140}$$

Plugging into Equation 3.137 we get:

$$\frac{\lambda}{\beta^2} = \frac{K(x_d) - \beta\dot{\phi}}{\beta\phi} \tag{3.141}$$

Re-writing gives us:

$$\dot{\phi} + \frac{\lambda}{\beta^2}\phi = \frac{K(x_d)}{\beta} \tag{3.142}$$

With $\dot{\phi} \geq 0$ from Equation 3.139, we get:

$$\dot{\phi} = \beta K(x_d) - \lambda\phi \geq 0 \tag{3.143}$$

Or:

$$K(x_d) \geq \frac{\lambda\phi}{\beta} \tag{3.144}$$

With $\dot{\phi} \leq 0$ from Equation 3.142, we get:

$$\dot{\phi} = \frac{K(x_d)}{\beta} - \frac{\lambda}{\beta^2}\phi \leq 0 \tag{3.145}$$

Or:

$$K(x_d) \leq \frac{\lambda\phi}{\beta} \tag{3.146}$$

Assuming $\vec{x}(0) = \overrightarrow{x_d}(0)$, we can show:

$$\phi(0) = \frac{\beta}{\lambda}K(x_d(0)) \tag{3.147}$$

In summary, plugging in Equation 3.99 to Equations 3.130:

$$u = \hat{b}^{-1}\left[-(\ddot{x} - \ddot{x}_d) - \lambda(\dot{x} - \dot{x}_d) - \overline{K}sat(\frac{s}{\phi})\right] + 2u_{k-1} - u_{k-2} \tag{3.148}$$

$$K = |(\beta - 1)||(\ddot{x} - \ddot{x}_d)| + |(\beta - 1)||\lambda(\dot{x} - \dot{x}_d)| + |(\beta[1 + \sigma_u] - 1)|\left|\hat{b}(u_{k-2} - u_{k-1})\right| + \beta\eta \tag{3.149}$$

$$K_d = |(\beta_d[1 + \sigma_u] - 1)||\hat{b}(u_{k-2} - u_{k-1})| + \beta_d\eta \tag{3.150}$$

$$K_d \leq \frac{\lambda\phi}{\beta_d} \Rightarrow \dot{\phi} + \lambda\phi = \beta_d K_d \quad \overline{K} = K - \frac{\dot{\phi}}{\beta} \tag{3.151}$$

$$K_d \geq \frac{\lambda\phi}{\beta_d} \Rightarrow \dot{\phi} + \frac{\lambda\phi}{\beta_d^2} = \frac{K_d}{\beta_d} \qquad \overline{K} = K - \beta\dot{\phi} \tag{3.152}$$

$$\phi(0) = \frac{\beta_d K_d(0)}{\lambda} \tag{3.153}$$

3.1.12 Derivation of MIMO Model-Free Control Law

For a MIMO system, the following discrete-time measurement model is assumed:

$$\ddot{\boldsymbol{x}} = \ddot{\boldsymbol{x}} + \boldsymbol{B}\boldsymbol{u} - \boldsymbol{B}\boldsymbol{u}_{\boldsymbol{k-1}} - \boldsymbol{B}\boldsymbol{u} + \boldsymbol{B}\boldsymbol{u}_{\boldsymbol{k-1}} \tag{3.154}$$

Or:

$$\ddot{x}_i = \ddot{x}_i + \varepsilon_i(u_i) + \sum_{j=1}^{m} b_{ij}(x_i)(u_i - u_{k-1_i}) \tag{3.155}$$
$$i = 1, \dots, m \quad j = 1, \dots, m$$

Where $\ddot{x}$ is the measured acceleration, u is the control system input, u_{k-1} is the previous value of the control system input, m is the order of the system, and b is the control input influence gain that will be estimated. The input influence gain matrix, $\boldsymbol{B}$, is assumed to be a 2 by 2 square matrix defined as:

$$\boldsymbol{B} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \tag{3.156}$$

Equation 3.154 can be re-written as:

$$\ddot{\boldsymbol{x}} = \ddot{\boldsymbol{x}} + \boldsymbol{B}\boldsymbol{u} - \boldsymbol{B}\boldsymbol{u}_{\boldsymbol{k-1}} + \boldsymbol{\varepsilon}(\boldsymbol{u}) \tag{3.157}$$

Or:

$$\ddot{x}_i = \ddot{x}_i + \varepsilon_i(u_i) + \sum_{j=1}^{m} b_{ij}(x_i)(u_i - u_{k-1_i}) \tag{3.158}$$
$$i = 1, \dots, m \quad j = 1, \dots, m$$

Where $\boldsymbol{\varepsilon}(u)$ is the estimation error in the input influence gain defined by:

$$\varepsilon_i(u_i) = \sum_{j=1}^{m} b_{ij}(x_i)(u_{k-1_i} - u_i) \tag{3.159}$$
$$i = 1, \dots, m \quad j = 1, \dots, m$$

$\boldsymbol{\varepsilon}(u)$ is assumed to be bounded by a known function, $\boldsymbol{E}$, such that:

$$|\hat{\boldsymbol{\varepsilon}}(\boldsymbol{u}) - \boldsymbol{\varepsilon}(\boldsymbol{u})| \leq \boldsymbol{E} \tag{3.160}$$

Where $\hat{\boldsymbol{\varepsilon}}(u)$ is the estimated error in the input influence gain estimation assumed to be defined by:

$$\hat{\varepsilon}_i(u_i) = \sum_{j=1}^{m} \hat{b}_{ij}(x_i)(u_{k-2_i} - u_{k-1_i}) \tag{3.161}$$
$$i = 1, \dots, m \quad j = 1, \dots, m$$

And the actual error is bounded by the function:

$$(1 - \boldsymbol{\sigma_l})\hat{\boldsymbol{\varepsilon}}(\boldsymbol{u}) \leq \boldsymbol{\varepsilon}(\boldsymbol{u}) \leq (1 + \boldsymbol{\sigma_u})\hat{\boldsymbol{\varepsilon}}(\boldsymbol{u}) \tag{3.162}$$

Where $\boldsymbol{\sigma_l}$ and $\boldsymbol{\sigma_u}$ are the lower and upper defined bounds respectively. The sliding surface for a second-order system is defined as:

$$s_i = \dot{x}_i - \dot{x}_{d_i} + \lambda(x_i - x_{d_i}) \tag{3.163}$$
$$i = 1, \dots, m$$

Where x and $\dot{x}$ are the system states to be measured and x_d and $\dot{x}_d$ are the desired states to be tracked. Taking the derivative of the sliding surface results in:

$$\dot{s}_i = \ddot{x}_i - \ddot{x}_{d_i} + \lambda\left(\dot{x}_i - \dot{x}_{d_i}\right) = 0 \tag{3.164}$$
$$i = 1, \dots, m$$

Plugging in Equation 3.158 and setting $\dot{s}$ equal to zero ensures that the state error trajectories do not move once they reach the sliding surface and gives us:

$$\dot{s}_i = \ddot{x}_i - \ddot{x}_{d_i} + \lambda\left(\dot{x}_i - \dot{x}_{d_i}\right) + \varepsilon_i(u_i) + \sum_{j=1}^{m} b_{ij}(x_i)(u_i - u_{k-1_i}) = 0 \tag{3.165}$$
$$i = 1, \dots, m \quad j = 1, \dots, m$$

The best estimate for the control input, $\hat{u}$, to maintain $\dot{s}$ equal to zero is therefore:

$$\hat{\boldsymbol{u}} = \hat{\boldsymbol{B}}^{-1}\left(-\hat{\boldsymbol{\varepsilon}}(\boldsymbol{u}) - \ddot{\boldsymbol{x}} + \ddot{\boldsymbol{x}}_{\boldsymbol{d}} - \lambda(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{\boldsymbol{d}})\right) + \boldsymbol{u_{k-1}} \tag{3.166}$$

And adding a discontinuous term to satisfy the sliding condition gives us:

$$\boldsymbol{u} = \hat{\boldsymbol{B}}^{-1}(-\hat{\boldsymbol{\varepsilon}}(\boldsymbol{u}) - \ddot{\boldsymbol{x}} + \ddot{\boldsymbol{x}}_{\boldsymbol{d}} - \lambda(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{\boldsymbol{d}})) - \boldsymbol{\eta} sgn(s_i)) + \boldsymbol{u_{k-1}} \tag{3.167}$$
$$i = 1, \dots, m$$

Where η is a matrix of positive constants. From Equation 3.112, we know that this controller form verifies the sliding condition. Therefore, $\boldsymbol{\eta}$ can be replaced with the system gain, $\boldsymbol{K}$, in Equation 3.167 resulting in:

$$\boldsymbol{u} = \hat{\boldsymbol{B}}^{-1}[-\hat{\boldsymbol{\varepsilon}}(\boldsymbol{u}) - \ddot{\boldsymbol{x}} + \ddot{\boldsymbol{x}}_{\boldsymbol{d}} - \lambda(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{\boldsymbol{d}}) - \boldsymbol{K}(x) sgn(s_i)] + \boldsymbol{u_{k-1}} \tag{3.168}$$
$$i = 1, \dots, m$$

We can define the sliding condition as:

$$s_i \dot{s}_i \leq -\eta_i |s_i| \qquad i = 1, \dots, m \tag{3.169}$$

Substituting in Equation 3.165 gives us:

$$s_i \left[\ddot{x}_i - \ddot{x}_{d_i} + \lambda\left(\dot{x}_i - \dot{x}_{d_i}\right) + \varepsilon_i(u_i) + \sum_{j=1}^{m} b_{ij}(x_i)(u_i - u_{k-1_i}) \right] \leq -\eta_i |s_i| \qquad i = 1, \dots, m \quad j = 1, \dots, m \tag{3.170}$$

Substituting in the best estimate of the control input, $\hat{u}$, for u gives us:

$$\dot{\boldsymbol{s}} = \ddot{\boldsymbol{x}} - \ddot{\boldsymbol{x}}_{\boldsymbol{d}} + \lambda(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{\boldsymbol{d}}) + \boldsymbol{\varepsilon}(\boldsymbol{u}) + \boldsymbol{B}\left(\widehat{\boldsymbol{B}}^{-1}[-\hat{\boldsymbol{\varepsilon}}(\boldsymbol{u}) - \ddot{\boldsymbol{x}} + \ddot{\boldsymbol{x}}_{\boldsymbol{d}} - \lambda(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{\boldsymbol{d}}) - \boldsymbol{K}(x)sgn(s_i)] + \boldsymbol{u}_{\boldsymbol{k-1}} - \boldsymbol{u}_{\boldsymbol{k-1}}\right) \qquad i = 1, \dots, m \tag{3.171}$$

Using the $s_i K_i sgn(s_i) = K_i |s_i|$ and rearranging to isolate $K_i |s_i|$ gives us:

$$K_i |s_i| \geq \sum_{j=1}^{m} s_i \left[\left(\ddot{x}_i - \ddot{x}_{d_i}\right)\left(\hat{b}_{ij} {b_{ij}}^{-1} - 1\right) + \lambda\left(\dot{x}_i - \dot{x}_{d_i}\right)\left(\hat{b}_{ij} {b_{ij}}^{-1} - 1\right) + \left(\hat{b}_{ij} {b_{ij}}^{-1} - 1\right)\hat{\varepsilon}_i(u_i) + \hat{b}_{ij} {b_{ij}}^{-1}[\varepsilon_i(u_i) - \hat{\varepsilon}_i(u_i)] \right] + \hat{b}_{ij} {b_{ij}}^{-1} \eta_i |s_i| \qquad i = 1, \dots, m \quad j = 1, \dots, m \tag{3.172}$$

The most conservative estimate for the upper bound of $\boldsymbol{\varepsilon}(u)$ is defined as:

$$\boldsymbol{\varepsilon}(u) = (1 + \boldsymbol{\sigma_u})\hat{\boldsymbol{\varepsilon}}(u) \tag{3.173}$$

Therefore, the $\boldsymbol{\varepsilon}(u) - \hat{\boldsymbol{\varepsilon}}(u)$ term can be redefined as:

$$\boldsymbol{\varepsilon}(u) - \hat{\boldsymbol{\varepsilon}}(u) = (1 + \boldsymbol{\sigma_u})\hat{\boldsymbol{\varepsilon}}(u) - \hat{\boldsymbol{\varepsilon}}(u) = \hat{\boldsymbol{\varepsilon}}(u) + \boldsymbol{\sigma_u}\hat{\boldsymbol{\varepsilon}}(u) - \hat{\boldsymbol{\varepsilon}}(u) = \boldsymbol{\sigma_u}\hat{\boldsymbol{\varepsilon}}(u) \tag{3.174}$$

Plugging this into Equation 3.172 gives us:

$$K_i |s_i| \geq \sum_{j=1}^{m} s_i \left[\left(\ddot{x}_i - \ddot{x}_{d_i}\right)\left(\hat{b}_{ij} {b_{ij}}^{-1} - 1\right) + \lambda\left(\dot{x}_i - \dot{x}_{d_i}\right)\left(\hat{b}_{ij} {b_{ij}}^{-1} - 1\right) + \left(\hat{b}_{ij} {b_{ij}}^{-1} - 1\right)\hat{\varepsilon}_i(u_i) + \hat{b}_{ij} {b_{ij}}^{-1} \sigma_{u_i} \hat{\varepsilon}_i(u_i) \right] + \hat{b}_{ij} {b_{ij}}^{-1} \eta_i |s_i| \qquad i = 1, \dots, m \quad j = 1, \dots, m \tag{3.175}$$

And rearranging this gives us:

$$K_i |s_i| \geq \sum_{j=1}^{m} s_i \left[\left(\ddot{x}_i - \ddot{x}_{d_i}\right)\left(\hat{b}_{ij} {b_{ij}}^{-1} - 1\right) + \lambda\left(\dot{x}_i - \dot{x}_{d_i}\right)\left(\hat{b}_{ij} {b_{ij}}^{-1} - 1\right) + \left(\hat{b}_{ij} {b_{ij}}^{-1}[1 + \sigma_{u_i}] - 1\right)\hat{\varepsilon}_i(u_i) \right] + \hat{b}_{ij} {b_{ij}}^{-1} \eta_i |s_i| \qquad i = 1, \dots, m \quad j = 1, \dots, m \tag{3.176}$$

Next, we define:

$$\widehat{\boldsymbol{B}} = \sqrt{\boldsymbol{B_{low} B_{upp}}} \tag{3.177}$$

$$\boldsymbol{\beta} = \sqrt{\boldsymbol{B_{low}}^{-1} \boldsymbol{B_{upp}}} = \widehat{\boldsymbol{B}} \boldsymbol{B}^{-1} \tag{3.178}$$

Where $\boldsymbol{B_{low}}$ and $\boldsymbol{B_{upp}}$ are the lower and upper estimated bounds of the input influence gain matrix respectively. Plugging these into Equation 3.176 gives us:

$$K_i|s_i| \geq \sum_{j=1}^{m} s_i\left[(\ddot{x}_i - \ddot{x}_{d_i})(\beta_{ij}{}^{-1} - 1) + \lambda(\dot{x}_i - \dot{x}_{d_i})(\beta_{ij}{}^{-1} - 1) + (\beta_{ij}{}^{-1}[1 + \sigma_{u_i}] - 1)\hat{\varepsilon}_i(u_i)\right] + \beta_{ij}{}^{-1}\eta_i|s_i| \tag{3.179}$$
$$i = 1, \dots, m \quad j = 1, \dots, m$$

With the final equation for the system gain being:

$$\boldsymbol{K} \geq |(\boldsymbol{\beta} - 1)||(\ddot{\boldsymbol{x}} - \ddot{\boldsymbol{x}}_{\boldsymbol{d}})| + |(\boldsymbol{\beta} - 1)||\lambda(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{\boldsymbol{d}})| + |(\boldsymbol{\beta}[1 + \boldsymbol{\sigma}_{\boldsymbol{u}}] - 1)||\hat{\boldsymbol{\varepsilon}}(u)| + \boldsymbol{\beta\eta} \tag{3.180}$$

In summary, plugging in Equation 3.161 to Equations and 3.168 and 3.180:

$$\boldsymbol{u} = \widehat{\boldsymbol{B}}^{-1}[-(\ddot{\boldsymbol{x}} - \ddot{\boldsymbol{x}}_{\boldsymbol{d}}) - \lambda(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{\boldsymbol{d}}) - \boldsymbol{K}sgn(s_i)] + 2\boldsymbol{u_{k-1}} - \boldsymbol{u_{k-2}} \tag{3.181}$$
$$i = 1, \dots, m$$

$$\boldsymbol{K} = |(\boldsymbol{\beta} - 1)||(\ddot{\boldsymbol{x}} - \ddot{\boldsymbol{x}}_{\boldsymbol{d}})| + |(\boldsymbol{\beta} - 1)||\lambda(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{\boldsymbol{d}})| + |(\boldsymbol{\beta}[1 + \boldsymbol{\sigma}_{\boldsymbol{u}}] - 1)|\left|\widehat{\boldsymbol{B}}(\boldsymbol{u_{k-2}} - \boldsymbol{u_{k-1}})\right| + \boldsymbol{\beta\eta} \tag{3.182}$$

Including a boundary layer, the sliding condition becomes:

$$s_i\dot{s}_i \leq (\dot{\phi}_i - \eta_i)|s_i| \tag{3.183}$$
$$i = 1, \dots, m$$

Where m is the order of the system. To ensure that the sliding condition, $(\dot{\boldsymbol{\phi}} - \boldsymbol{\eta}) \leq 0$, is met when the states are outside the boundary layer, or $\dot{\boldsymbol{\phi}} > 0$, the switching gain is defined as:

$$\overline{\boldsymbol{K}}(x) = \boldsymbol{K}(x) - \boldsymbol{\beta}^{-1}\dot{\boldsymbol{\phi}} \tag{3.184}$$

From Equation 3.168, $Ksgn(s)$ is replaced with $\overline{K}sat(\frac{s}{\phi})$:

$$\boldsymbol{u} = \widehat{\boldsymbol{B}}^{-1}\left[-\hat{\boldsymbol{\varepsilon}}(\boldsymbol{u}) - \ddot{\boldsymbol{x}} + \ddot{\boldsymbol{x}}_{\boldsymbol{d}} - \lambda(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{\boldsymbol{d}}) - \overline{\boldsymbol{K}}(x)sat\left(\frac{s_i}{\phi_i}\right)\right] + \boldsymbol{u_{k-1}} \tag{3.185}$$
$$i = 1, \dots, m$$

Plugging into Equation 3.165 gives us:

$$\dot{\boldsymbol{s}} = \ddot{\boldsymbol{x}} - \ddot{\boldsymbol{x}}_{\boldsymbol{d}} + \lambda(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{\boldsymbol{d}}) + \boldsymbol{\varepsilon}(\boldsymbol{u}) + \boldsymbol{B}\left(\widehat{\boldsymbol{B}}^{-1}\left[-\hat{\boldsymbol{\varepsilon}}(\boldsymbol{u}) - \ddot{\boldsymbol{x}} + \ddot{\boldsymbol{x}}_{\boldsymbol{d}} - \lambda(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{\boldsymbol{d}}) - \overline{\boldsymbol{K}}(x)sat\left(\frac{s_i}{\phi_i}\right)\right] + \boldsymbol{u_{k-1}} - \boldsymbol{u_{k-1}}\right) \tag{3.186}$$
$$i = 1, \dots, m$$

And rearranging this gives us:

$$\dot{\boldsymbol{s}} = -\boldsymbol{B}\widehat{\boldsymbol{B}}^{-1}\overline{\boldsymbol{K}}(x)sat\left(\frac{s_i}{\phi_i}\right) + (\ddot{\boldsymbol{x}} - \ddot{\boldsymbol{x}}_{\boldsymbol{d}})\left(1 - \boldsymbol{B}\widehat{\boldsymbol{B}}^{-1}\right) + \lambda(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{\boldsymbol{d}})\left(1 - \boldsymbol{B}\widehat{\boldsymbol{B}}^{-1}\right) + \boldsymbol{\varepsilon}(u) - \boldsymbol{B}\widehat{\boldsymbol{B}}^{-1}\hat{\boldsymbol{\varepsilon}}(u) \tag{3.187}$$
$$i = 1, \dots, m$$

Using Equation 3.178, we can re-write this as:

$$\dot{\boldsymbol{s}} = -\boldsymbol{\beta}^{-1}\overline{\boldsymbol{K}}(x)sat\left(\frac{s_i}{\phi_i}\right) + (\ddot{\boldsymbol{x}} - \ddot{\boldsymbol{x}}_{\boldsymbol{d}})(1 - \boldsymbol{\beta}^{-1}) + \lambda(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{\boldsymbol{d}})(1 - \boldsymbol{\beta}^{-1}) + \boldsymbol{\varepsilon}(u) - \boldsymbol{\beta}^{-1}\hat{\boldsymbol{\varepsilon}}(u) \tag{3.188}$$

$$i = 1, \dots, m$$

Inside the boundary layer, $sat\left(\frac{s_i}{\phi_i}\right) = \frac{s_i}{\phi_i}$, so we can re-write this as:

$$\dot{s}_i = \sum_{j=1}^{m} -\left(\frac{\overline{K}_i(x_i)}{\beta_{ij}}\right)\left(\frac{s_i}{\phi_i}\right) + \left(\ddot{x} - \ddot{x}_{d_i}\right)\left(1 - {\beta_{ij}}^{-1}\right) + \lambda\left(\dot{x}_i - \dot{x}_{d_i}\right)\left(1 - {\beta_{ij}}^{-1}\right) + \varepsilon_i(u) - {\beta_{ij}}^{-1}\hat{\varepsilon}_i(u) \tag{3.189}$$

$$i = 1, \dots, m \quad j = 1, \dots, m$$

Assume that inside the boundary layer, $\vec{x} = \overrightarrow{x_d} + \vec{\varepsilon}$, where ε is a small error term, such that:

$$\dot{s}_i = \sum_{j=1}^{m} -\left(\frac{\overline{K}_i(x_{d_i})}{\beta_{ij}}\right)\left(\frac{s_i}{\phi_i}\right) + \varepsilon_i(u) - {\beta_{ij}}^{-1}\hat{\varepsilon}_i(u) \tag{3.190}$$

$$i = 1, \dots, m \quad j = 1, \dots, m$$

Equation 3.190 is a first-order filter that can be tuned such that:

$$\dot{\boldsymbol{s}} + \lambda \boldsymbol{\beta}^{-2}\boldsymbol{s} = \boldsymbol{u} \tag{3.191}$$

Comparing the two equations gives us:

$$\sum_{j=1}^{m} \lambda {\beta_{ij}}^{-2} = \sum_{j=1}^{m} \frac{\overline{K}_i(x_{d_i})}{\beta_{ij}\phi_i} \tag{3.192}$$

$$i = 1, \dots, m \quad j = 1, \dots, m$$

Plugging in Equation 3.184 gives us:

$$\sum_{j=1}^{m} \lambda {\beta_{ij}}^{-2} = \sum_{j=1}^{m} \frac{K_i(x_{d_i}) - {\beta_{ij}}^{-1}\dot{\phi}_i}{\beta_{ij}\phi_i} \tag{3.193}$$

$$i = 1, \dots, m \quad j = 1, \dots, m$$

Re-writing gives us:

$$\dot{\phi}_i + \lambda\phi_i = \sum_{j=1}^{m} \beta_{ij} K_i(x_{d_i}) \tag{3.194}$$

$$i = 1, \dots, m \quad j = 1, \dots, m$$

Or, in vector form:

$$\dot{\boldsymbol{\phi}} + \lambda\boldsymbol{\phi} = \boldsymbol{\beta}\boldsymbol{K}(x_d) \tag{3.195}$$

To ensure that the sliding condition is met when the states are inside the boundary layer, or $\dot{\boldsymbol{\phi}} < 0$, the switching gain, $\overline{K}$, is defined as:

$$\overline{\boldsymbol{K}}(x_d) = \boldsymbol{K}(x_d) - \boldsymbol{\beta}\dot{\boldsymbol{\phi}} \tag{3.196}$$

Plugging into Equation 3.192 we get:

$$\sum_{j=1}^{m} \lambda {\beta_{ij}}^{-2} = \sum_{j=1}^{m} \frac{K_i(x_{d_i}) - \beta_{ij}\dot{\phi}_i}{\beta_{ij}\phi_i} \tag{3.197}$$
$$i = 1, \dots, m \quad j = 1, \dots, m$$

Re-writing gives us:

$$\dot{\phi}_i + \sum_{j=1}^{m} \lambda {\beta_{ij}}^{-2} \phi_i = \sum_{j=1}^{m} \frac{K_i(x_{d_i})}{\beta_{ij}} \tag{3.198}$$
$$i = 1, \dots, m \quad j = 1, \dots, m$$

Or, in vector form:

$$\dot{\boldsymbol{\phi}} + \lambda \boldsymbol{\beta}^{-2} \boldsymbol{\phi} = \boldsymbol{\beta}^{-1} \boldsymbol{K}(x_d) \tag{3.199}$$

With $\dot{\boldsymbol{\phi}} \geq 0$ from Equation 3.195, we get:

$$\dot{\boldsymbol{\phi}} = \boldsymbol{\beta} \boldsymbol{K}(x_d) - \lambda \boldsymbol{\phi} \geq 0 \tag{3.200}$$

Or:

$$\boldsymbol{K}(x_d) \geq \lambda \boldsymbol{\beta}^{-1} \boldsymbol{\phi} \tag{3.201}$$

With $\dot{\boldsymbol{\phi}} \leq 0$ from Equation 3.199, we get:

$$\dot{\boldsymbol{\phi}} = \boldsymbol{\beta}^{-1} K(x_d) - \lambda \boldsymbol{\beta}^{-2} \boldsymbol{\phi} \leq 0 \tag{3.202}$$

Or:

$$\boldsymbol{K}(x_d) \leq \lambda \boldsymbol{\beta}^{-1} \boldsymbol{\phi} \tag{3.203}$$

Assuming $\vec{x}(0) = \overrightarrow{x_d}(0)$, we find:

$$\boldsymbol{\phi}(0) = \frac{1}{\lambda} \boldsymbol{\beta} \boldsymbol{K}(x_d(0)) \tag{3.204}$$

In summary, plugging Equation 3.161 into Equation 3.185:

$$\boldsymbol{u} = \widehat{\boldsymbol{B}}^{-1} \left[-(\ddot{\boldsymbol{x}} - \ddot{\boldsymbol{x}}_{\boldsymbol{d}}) - \lambda(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{\boldsymbol{d}}) - \overline{\boldsymbol{K}}(x) sat\left(\frac{S_i}{\phi_i}\right) \right] + 2\boldsymbol{u}_{\boldsymbol{k-1}} - \boldsymbol{u}_{\boldsymbol{k-2}} \tag{3.205}$$
$$i = 1, \dots, m$$

$$\boldsymbol{K} = |(\boldsymbol{\beta} - 1)||(\ddot{\boldsymbol{x}} - \ddot{\boldsymbol{x}}_{\boldsymbol{d}})| + |(\boldsymbol{\beta} - 1)||\lambda(\dot{\boldsymbol{x}} - \dot{\boldsymbol{x}}_{\boldsymbol{d}})| + |(\boldsymbol{\beta}[1 + \boldsymbol{\sigma}_{\boldsymbol{u}}] - 1)|\left|\widehat{\boldsymbol{B}}(\boldsymbol{u}_{\boldsymbol{k-2}} - \boldsymbol{u}_{\boldsymbol{k-1}})\right| + \boldsymbol{\beta}\boldsymbol{\eta} \tag{3.206}$$

$$\boldsymbol{K}_{\boldsymbol{d}} = |(\boldsymbol{\beta}_{\boldsymbol{d}}[1 + \boldsymbol{\sigma}_{\boldsymbol{u}}] - 1)||\widehat{\boldsymbol{B}}(\boldsymbol{u}_{\boldsymbol{k-2}} - \boldsymbol{u}_{\boldsymbol{k-1}})| + \boldsymbol{\beta}_{\boldsymbol{d}}\boldsymbol{\eta} \tag{3.207}$$

$$\boldsymbol{K}(x_d) \geq \lambda {\boldsymbol{\beta}_{\boldsymbol{d}}}^{-1} \boldsymbol{\phi} \qquad \dot{\boldsymbol{\phi}} + \lambda \boldsymbol{\phi} = \boldsymbol{\beta}_{\boldsymbol{d}} \boldsymbol{K}(x_d) \quad \overline{\boldsymbol{K}}(x) = \boldsymbol{K}(x) - \boldsymbol{\beta}^{-1}\dot{\boldsymbol{\phi}} \tag{3.208}$$

$$\boldsymbol{K}(x_d) \leq \lambda {\boldsymbol{\beta}_{\boldsymbol{d}}}^{-1} \boldsymbol{\phi} \qquad \dot{\boldsymbol{\phi}} + \lambda {\boldsymbol{\beta}_{\boldsymbol{d}}}^{-2} \boldsymbol{\phi} = {\boldsymbol{\beta}_{\boldsymbol{d}}}^{-1} \boldsymbol{K}(x_d) \quad \overline{\boldsymbol{K}}(x) = \boldsymbol{K}(x) - \boldsymbol{\beta}\dot{\boldsymbol{\phi}} \tag{3.209}$$

$$\boldsymbol{\phi}(0) = \frac{1}{\lambda}\boldsymbol{\beta K}(x_d(0)) \tag{3.210}$$

3.2 Online Parameter Estimation Methods

The basis of parameters estimation is extracting parameter values from measurable system outputs. A general model for parameter estimation can be defined as (Slotine and Li [23]):

$$y(kT) = W(kT)b(kT) \tag{3.211}$$

Where b is the parameter to be estimated, which in our case is the input influence gain, vector y includes the outputs from the system used for estimation, W is a signal matrix, T is the discrete time step, and k is an integer multiplier. This model is only valid for discrete time. The predicted system output, $\hat{y}$, at time t, can be defined as:

$$\hat{y}(kT) = W(kT)\hat{b}(kT) \tag{3.212}$$

Where $\hat{b}$ is the predicted parameter at time t. Online estimation is based around the fact that the value of $\hat{b}$ is found recursively. In other words, it is updated every time there is a new set of data for y and W. The instantaneous prediction error, e_I, can then be defined as:

$$e_I = \hat{y}(kT) - y(kT) \tag{3.213}$$

Or, plugging in Equations 3.211 and 3.212:

$$e_I = W\hat{b}(kT) - Wb(kT) = W\tilde{b} \tag{3.214}$$

3.2.1 Standard Least-Squares Estimator

The standard least-squares method has the advantage of averaging out the effects of noise in measurements. This method can be implemented by minimizing the total prediction error with respect to $\hat{b}(t)$ (Slotine and Li [23]):

$$J = \int_o^t \left|\left|y(r) - W(r)\hat{b}(t)\right|\right|^2 dr \tag{3.215}$$

Where the estimated parameter $\hat{b}$ satisfies:

$$\int_o^t (W^T(r)W(r)dr)\hat{b}(t) = \int_o^t W^T y dr \tag{3.216}$$

We can then define:

$$P(t) = [\int_o^t (W^T(r)W(r)dr)]^{-1} \tag{3.217}$$

But for computational efficiency, it is better to calculate P, the estimator gain matrix, recursively, so the above equation can be replaced with the following differential equation:

$$\dot{P}^{-1} = W^T(t)W(t) \tag{3.218}$$

This can give us the equation:

$$\frac{d}{dt}\left[\hat{b}\right] = -P(t)W^T e_1 \tag{3.219}$$

To be able to update P directly, we use the following identity:

$$\frac{d}{dt}[PP^{-1}] = \dot{P}P^{-1} + P\dot{P}^{-1} = 0 \tag{3.220}$$

To get the equation:

$$\dot{P} = -PW^T WP \tag{3.221}$$

To successfully implement this method, P and $\hat{b}$ must be initialized with finite values. The initial value of P should be as high as possible within noise sensitivity constraints and $\hat{b}$ should be a best guess.

3.2.2 Convergence

Solving the differential Equations 3.218 and 3.219 above and using Equation 3.221 we can show that (Slotine and Li [23]):

$$P^{-1}(t) = P^{-1}(0) + \int_0^t W^T(r)W(r)dr \tag{3.222}$$

$$\frac{d}{dt}\left[P^{-1}(t)\tilde{b}(t)\right] = 0 \tag{3.223}$$

From which we can define:

$$\tilde{b}(t) = P(t)P^{-1}(0)\tilde{b}(0) \tag{3.224}$$

If W meets the criteria of:

$$\lambda_{min} \int_0^t W^T W dr \to \infty \text{ as } t \to \infty$$

Where λ_{min} is the smallest eigenvalue of W, then the gain matrix converges to zero and the estimated parameters asymptotically converge to their true values. Additionally, for any positive integer value of k:

$$\int_0^{k\delta+\delta} W^T W dr = \sum_{i=0}^{k} \int_{i\delta}^{i\delta+\delta} W^T W dr \geq k\alpha_1 I \tag{3.225}$$

Where δ and α_1 are positive constants. Therefore, if W is under persistent excitation, the above equation is satisfied and $P \to 0$ and $\tilde{b} \to 0$.

An initial parameter error, $\tilde{b}(0)$, or large initial gain, $P(0)$, can lead to a small parameter error for all time. If the initial gain is chosen such that $P(0) = p_0 I$, then:

$$\tilde{b}(t) = \left[I + p_0 \int_0^t W^T(r) W(r) dr \right]^{-1} \tilde{b}(0) \tag{3.226}$$

3.2.3 Least-Squares with Exponential Forgetting

When estimating time-varying parameters, it is known that past data is generated by past parameter values. Therefore, this past data should be discounted when estimating the current value of parameters. To implement exponential forgetting into least square estimation, the following cost function is defined (Slotine and Li [23]):

$$J = \int_o^t exp[-\int_s^t \lambda(r) dr] \left|\left| y(s) - W(s)\hat{b}(t) \right|\right|^2 ds \tag{3.227}$$

Where $\lambda(t) \geq 0$ is the time-varying forgetting factor. The parameter update law stays the same as:

$$\frac{d}{dt}\hat{b} = -P(t) W^T e_1 \tag{3.228}$$

But the gain update law has changed to be:

$$\frac{d}{dt}[P^{-1}(t)] = -\lambda(t) P^{-1} + W^T(t) W(t) \tag{3.229}$$

Which can be implemented to directly update P in the form of:

$$\dot{P} = \lambda(t) P - P W^T(t) W(t) P \tag{3.230}$$

This exponential forgetting method improves parameter convergence over the traditional least-squares method by creating exponential convergence of the parameters to their final values. This is done while still guaranteeing asymptotic convergence of estimated parameters.

3.2.4 Control Law Implementation

Including a boundary layer, we define the sliding condition as:

$$s\dot{s} \leq (\dot{\phi} - \eta)|s| \tag{3.231}$$

To ensure the global asymptotic stability of the system, we want to ensure that the sliding condition is met. Thus, we define our instantaneous parameter estimation error based on meeting the sliding condition as shown below:

$$e_1 = (\dot{\phi} - \eta)|s| - s\dot{s} \tag{3.232}$$

The value of e_1 is minimized as the state trajectories reach the boundary layer. A boundary layer closing function is also defined as:

$$|s| \geq \phi \Rightarrow e_1 = (\dot{\phi} - \eta)|s| - s\dot{s} \tag{3.233}$$

$$|s| < \phi \Rightarrow e_1 = -\left[(\dot{\phi} - \eta)|s| - s\dot{s}\right] \tag{3.234}$$

As the state trajectory reaches a point within the boundary layer, this function starts to reduce the size of the boundary layer. Reducing the size of the boundary layer decreases tracking error and controller input over time.

For the implementation of Equation 3.230, the signal matrix, W was defined as:

$$W = |s| \tag{3.235}$$

This was done to ensure that the value of $\hat{b}$ varied while the value of the sliding surface varied over time.

3.2.5 Bounded Gain Forgetting Factor Tuning

The benefit of data forgetting is the ability to track slowly varying parameters. However, the gain matrix P can grow unbounded when W is not persistently exciting. Therefore, it is desirable to tune the forgetting factor variation such that data forgetting is active when W is persistently exciting and not active when W is not persistently exciting. The magnitude of the gain matrix P indicates the excitation level of W. Therefore, the forgetting factor variation can be made dependent on $\|P(t)\|$ (Slotine and Li [23]) such that:

$$\lambda(t) = \lambda_0 \left(1 - \frac{\|P(t)\|}{k_0}\right) \tag{3.236}$$

Where λ_0 is the maximum forgetting rate and k_0 is the bound for the gain matrix magnitude and both are positive constants. A higher value of λ_0 leads to faster forgetting but more oscillations in the estimated parameters. A higher value of k_0 updates the parameter estimation values faster but makes the estimator less robust to disturbances in the prediction error. In order for k_0 to be the upper bound of the gain matrix, we choose $\|P(0)\| \leq k_0$.

3.3 Implementation Results

All of the following simulations used a sampling time of 0.001 seconds as it is easily obtained on current aircraft control systems. A single desired tracking and parameter varying frequency was used for each system because the effects of different frequencies were studied in past work by Dr. Crassidis.

3.3.1 SISO System

The derived control law was implemented on the following nonlinear second-order system:

$$\ddot{x} + 3x\dot{x} + 5x^2 = 3u$$

With the desired tracking being:

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

The results of simulating the system with the derived control law for 60 seconds and an assumed constant input influence gain of 3 are shown below:

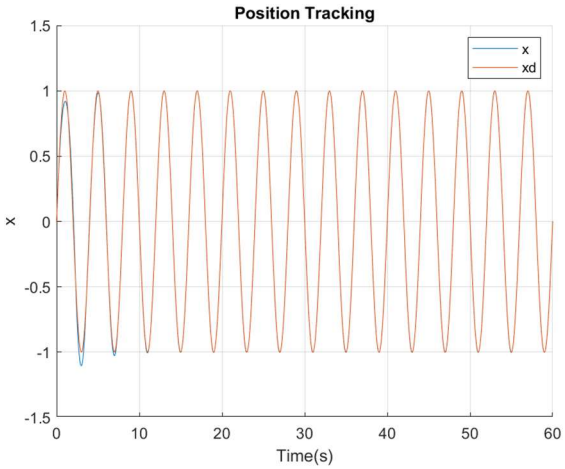

Figure 7: Position Tracking with Constant Input Influence Gain

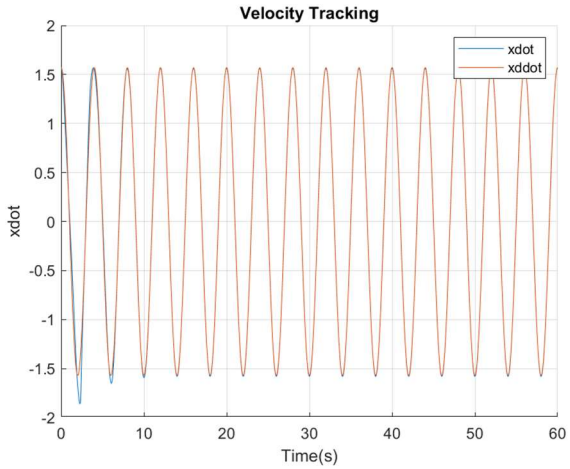

Figure 8: Velocity Tracking with Constant Input Influence Gain

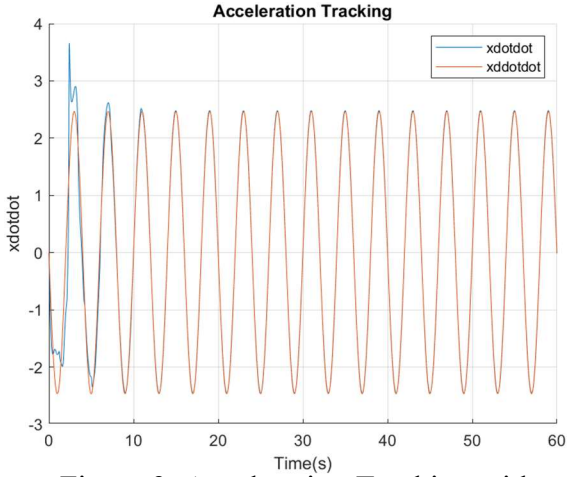

Figure 9: Acceleration Tracking with Constant Input Influence Gain

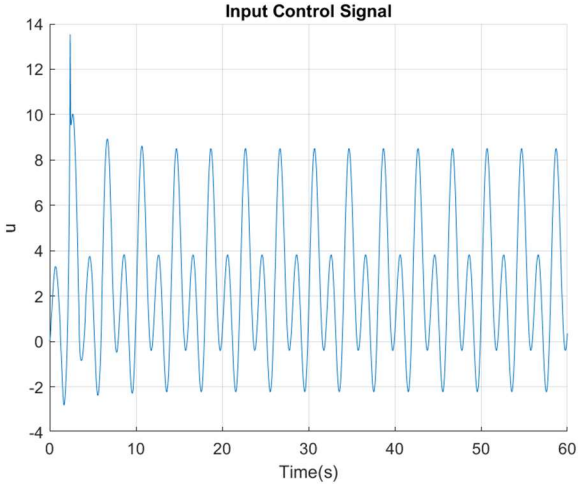

Figure 10: Controller Input with Constant Input Influence Gain

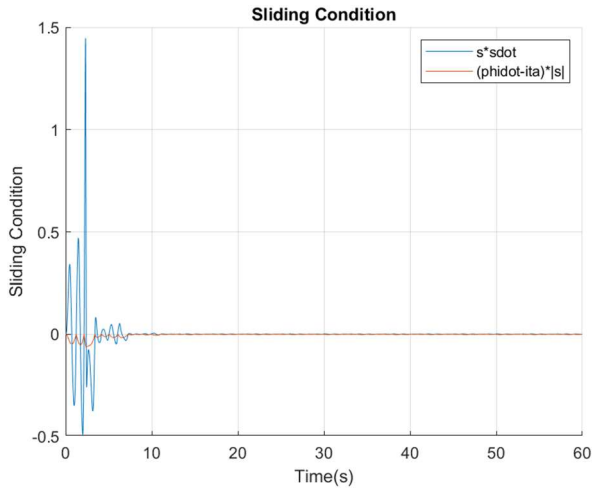

Figure 11: Sliding Condition with Constant Input Influence Gain

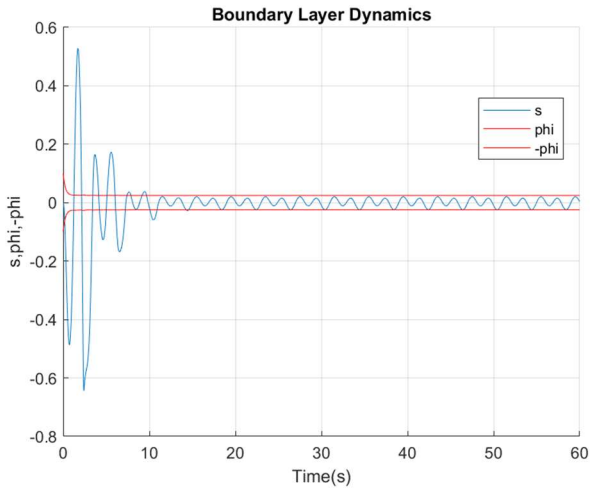

Figure 12: Boundary Layer Dynamics with Constant Input Influence Gain

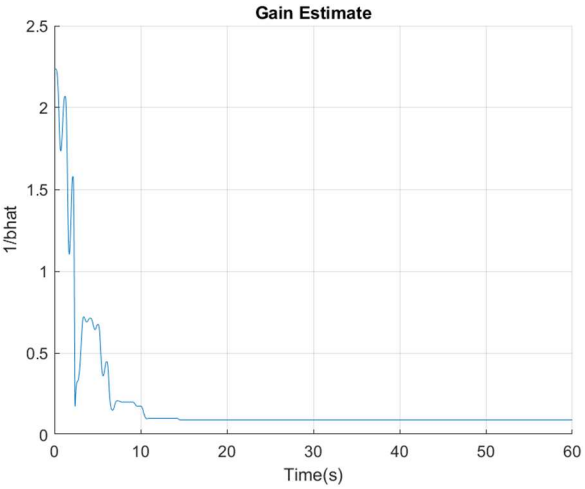

Figure 13: Input Influence Gain Estimate with Constant Input Influence Gain

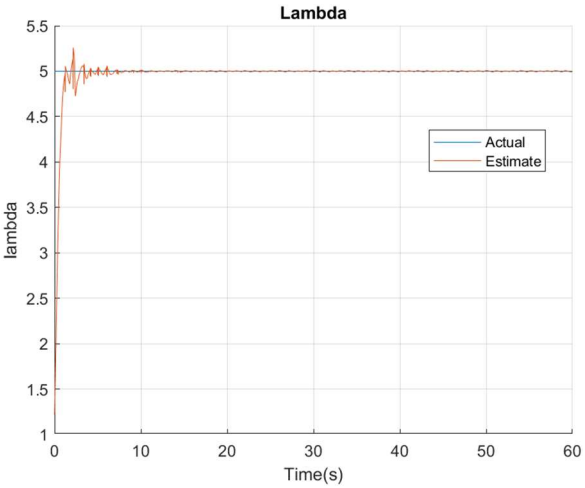

Figure 14: Lambda Estimate with Constant Input Influence Gain

Based on experience in the tracking of aircraft systems, Figures 7, 8, and 9 show that the system was deemed to have excellent tracking. The sliding condition was also met as time went on and the estimate for the input influence gain and λ were both convergent as shown by Figures 11, 13, and 14. This was accomplished with a controller input that was smooth and adequately small as shown in Figure 10.

Note: the control input influence gain value does not need to be known before simulating the system with the control system as the control system is wholly model-free. However, to decrease controller activity, the upper and lower bounds of the input influence gain estimate can be

redefined in further simulations to encompass the final estimated value of the input influence gain.

To demonstrate the robustness of this control law, the input influence gain was varied using a sine wave defined as:

$$b(t) = \left(\frac{b_{upp} - b_{low}}{2}\right)\sin(t) + \frac{b_{upp} + b_{low}}{2}$$

Where b_{upp} and b_{low} are the upper and lower estimated bounds of the input influence gain respectively. This is shown in Figure 15. These results of this simulation are also shown below:

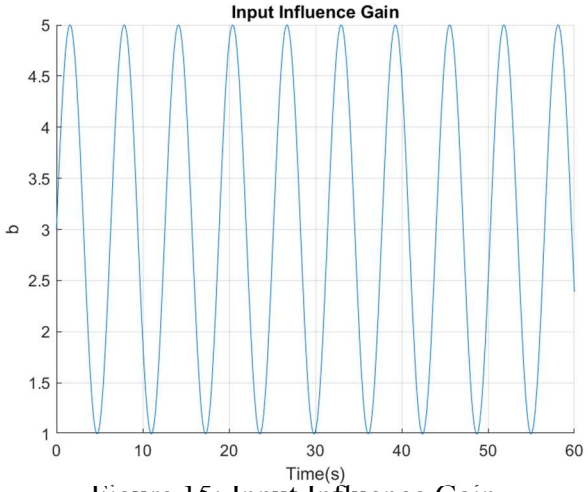

Figure 15: Input Influence Gain

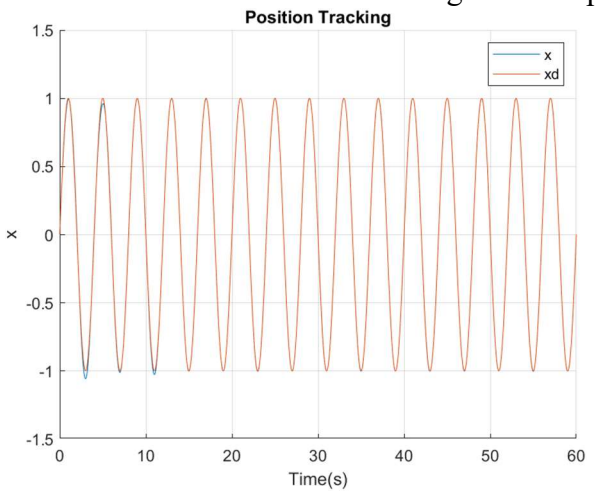

Figure 16: Position Tracking with Varying Input Influence Gain

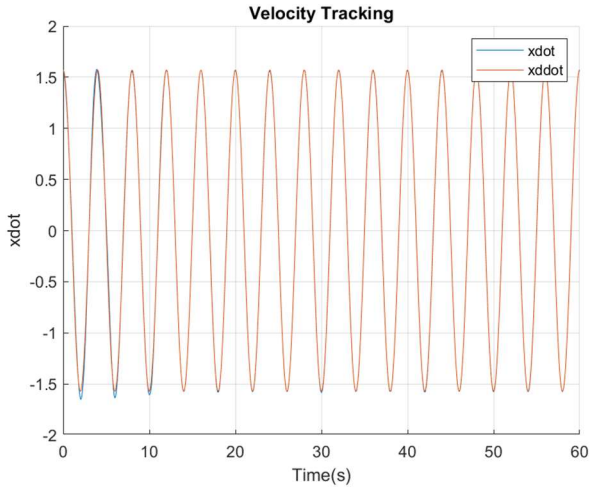

Figure 17: Velocity Tracking with Varying Input Influence Gain

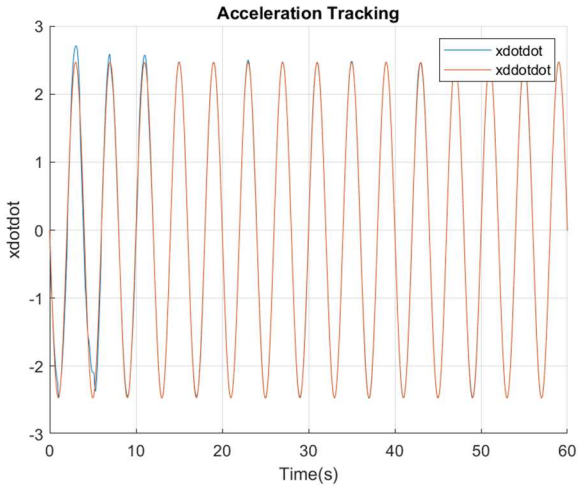

Figure 18: Acceleration Tracking with Varying Input Influence Gain

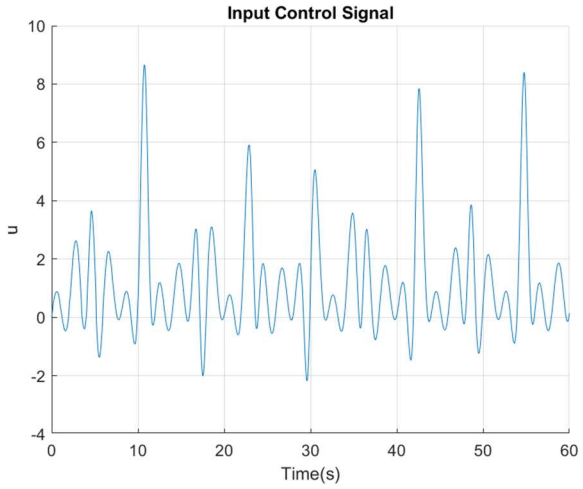

Figure 19: Control Input with Varying Input Influence Gain

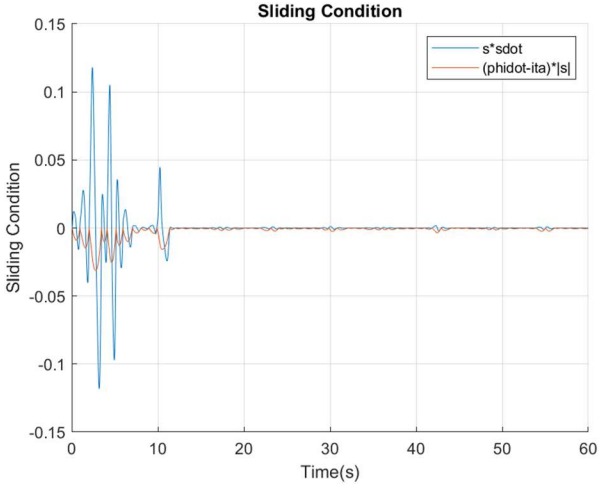

Figure 20: Sliding Condition Tracking with Varying Input Influence Gain

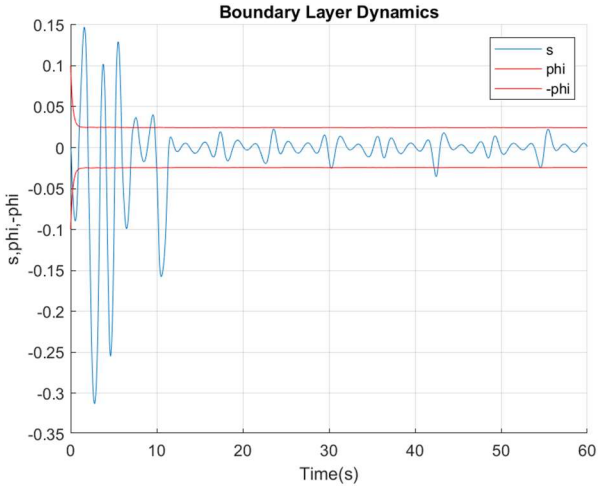

Figure 21: Boundary Layer Dynamics Tracking with Varying Input Influence Gain

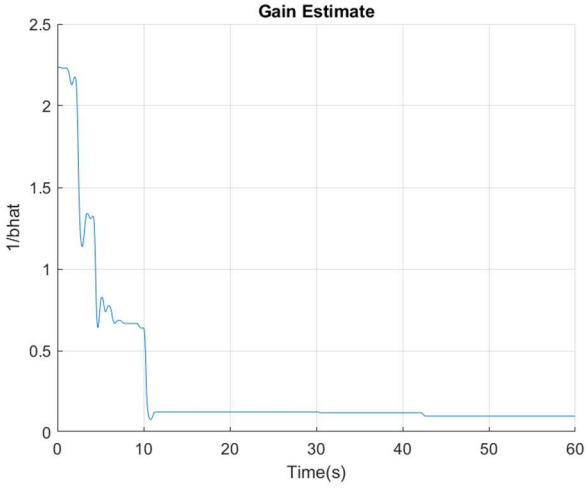

Figure 22: Input Influence Gain Estimate with Varying Input Influence Gain

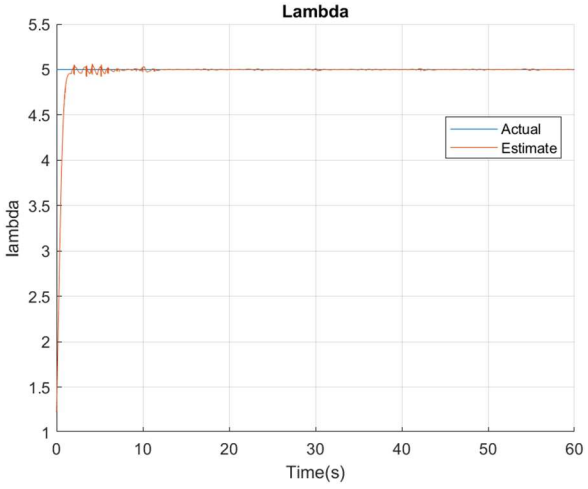

Figure 23: Lambda Estimate with Varying Input Influence Gain

As seen in Figures 16, 17, and 18, the controller still provides excellent position, velocity, and acceleration tracking. Again, sliding condition is met and the estimates for the input influence gain and λ are both convergent while the control input is smooth and adequately small.

3.3.2 MIMO System

The control law was also updated to control an example second-order MIMO system as defined by:

$$\ddot{x}_1 + x_1 x_2 - 3\dot{x}_1 + 4x_2^3 = b_1 u_1$$

$$\ddot{x}_2 + 4\dot{x}_2 x_1 + x_1^2 + 7x_2 = b_2 u_2$$

With the desired tracking defined by:

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

$$x_{2_d}(t) = \sin\left(\frac{\pi}{4}t\right)$$

The input influence gains b_1 and b_2 were varied as described in Section 3.3.1 to demonstrate the robustness of the control law. This is shown in Figure 24. Two separate SISO control laws were used to control each variable.

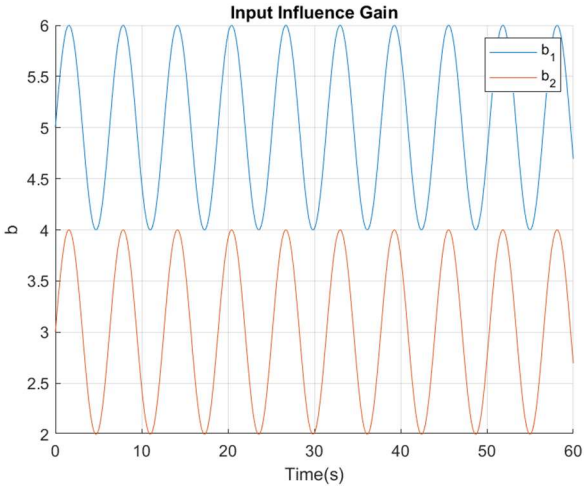

Figure 24: MIMO Input Influence Gain

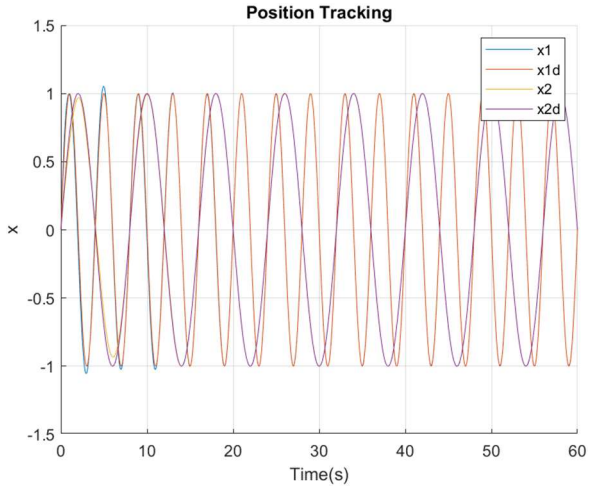

Figure 25: MIMO Position Tracking

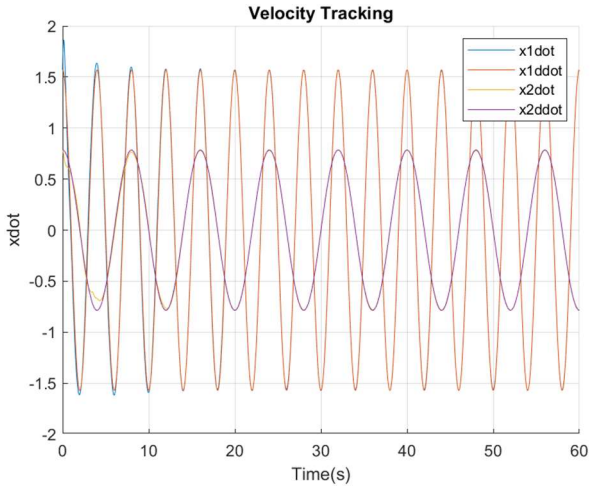

Figure 26: MIMO Velocity Tracking

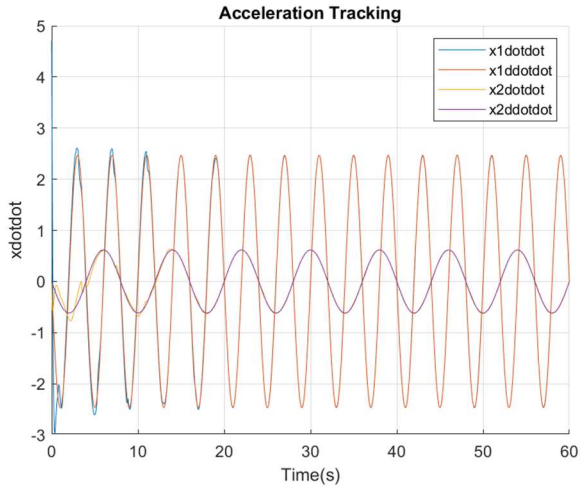

Figure 27: MIMO Acceleration Tracking

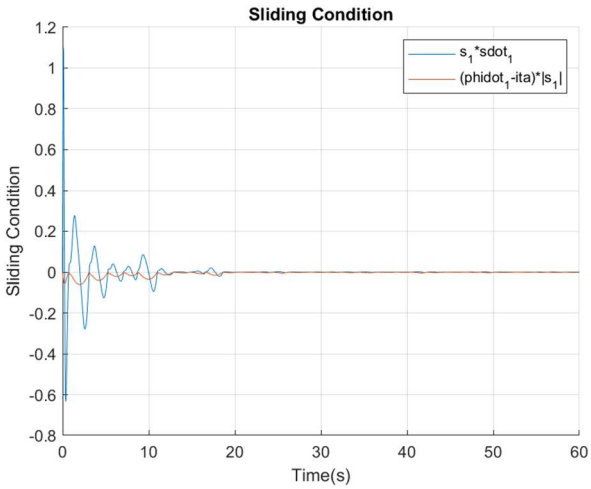

Figure 28: x_1 Sliding Condition

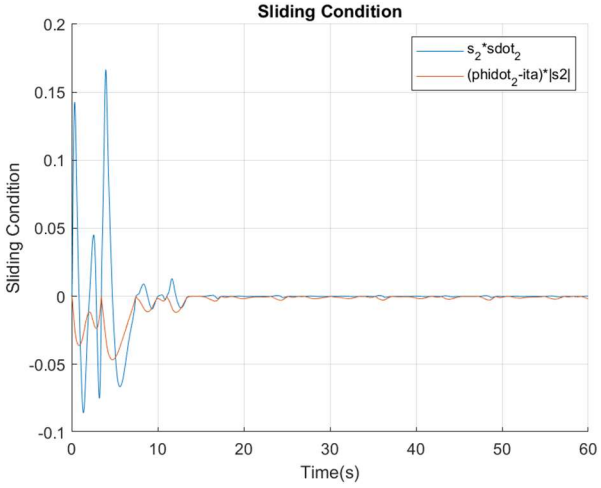

Figure 29: x_2 Sliding Condition

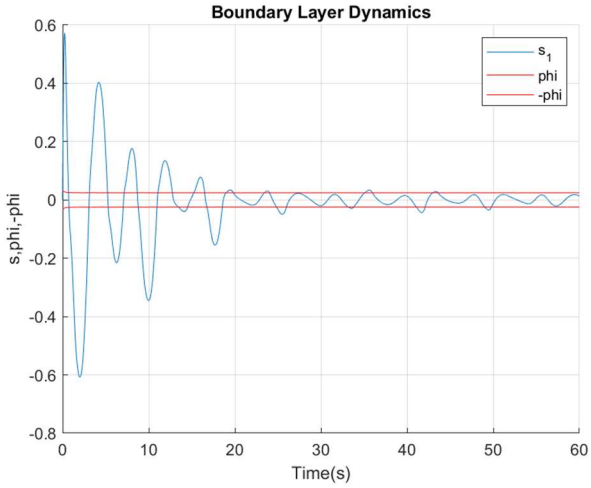

Figure 30: x_1 Boundary Layer Dynamics

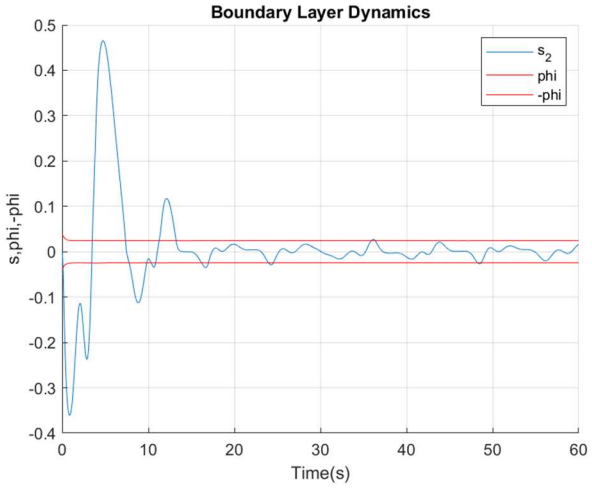

Figure 31: x_2 Boundary Layer Dynamics

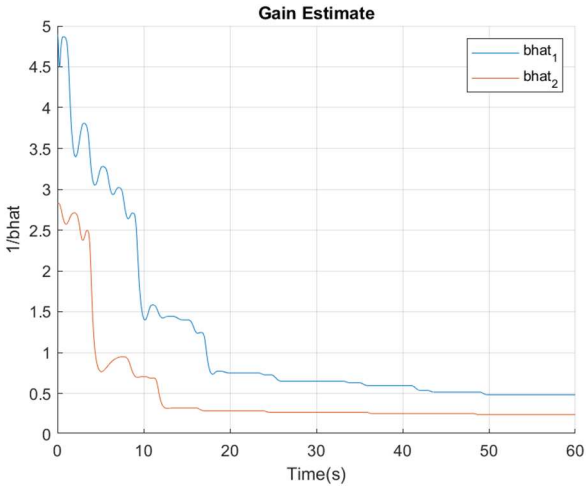

Figure 32: MIMO Input Influence Gain Estimate

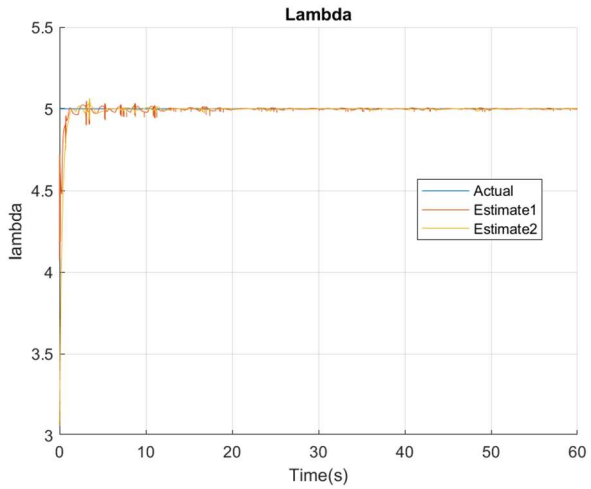

Figure 33: MIMO Lambda Estimate

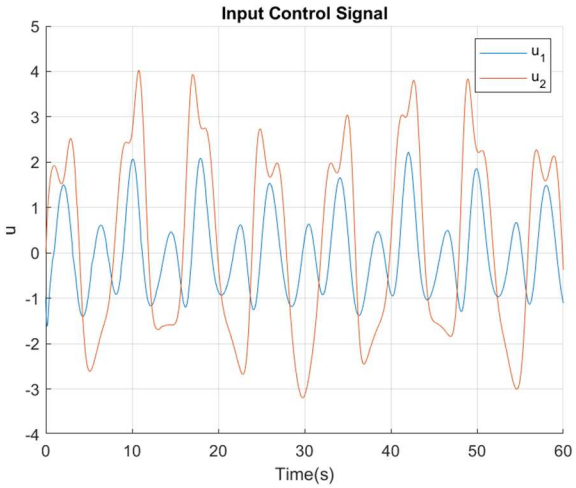

Figure 34: MIMO Controller Input

3.3.3 First-Order System

In preparation for the control of roll and yaw rate for an aircraft, the control law was updated to control a first-order SISO system. This included the re-derivation of the control law after the sliding surface and derivative of the sliding surface are updated such that:

$$s = x - x_d + \lambda \int_0^t (x - x_d)dt \tag{3.237}$$

$$\dot{s} = \dot{x} - \dot{x}_d + \lambda(x - x_d) \tag{3.238}$$

This control law was implemented on an example first-order SISO system defined as:

$$\dot{x} + 7x + 5x^3 = bu$$

Where the input influence gain, b, was time varying as described in Section 3.3.1. The desired tracking signal was updated to be a step function with a magnitude of 1 ran through a transfer function. To generate a desired "roll" signal, a first-order transfer function was used such that:

$$\frac{p_d}{\delta_a} = \frac{K}{\tau s + 1}$$

Where p_d is the desired roll rate, δ_a is the aileron step function input, τ is a roll-mode time constant set to 0.3 seconds, and K is a gain set to 1. The results of simulating the system with the updated control law are shown below:

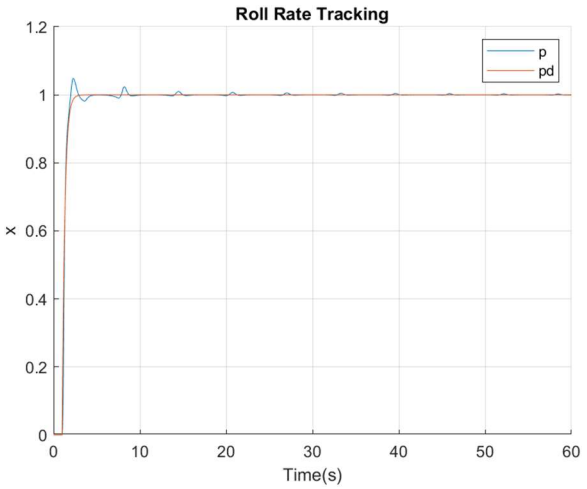

Figure 35: Roll Rate Tracking

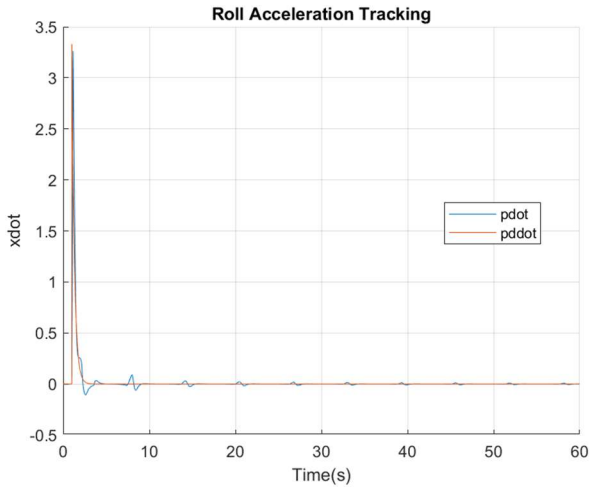

Figure 36: Roll Acceleration Tracking

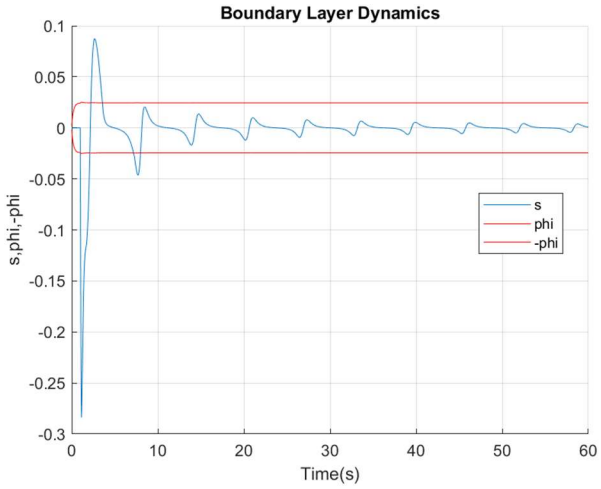

Figure 37: Roll Boundary Layer Dynamics

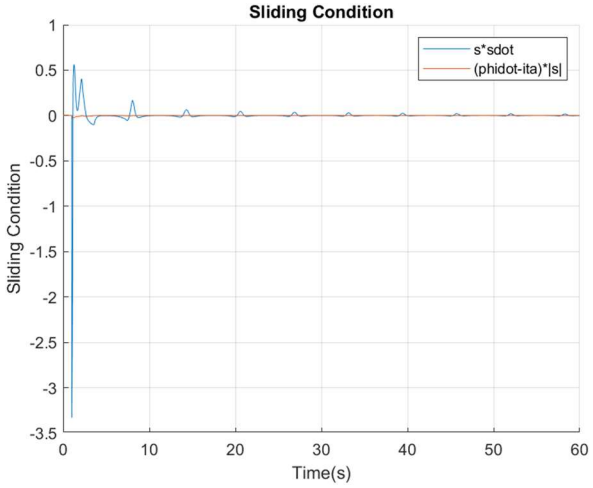

Figure 38: Roll Sliding Condition

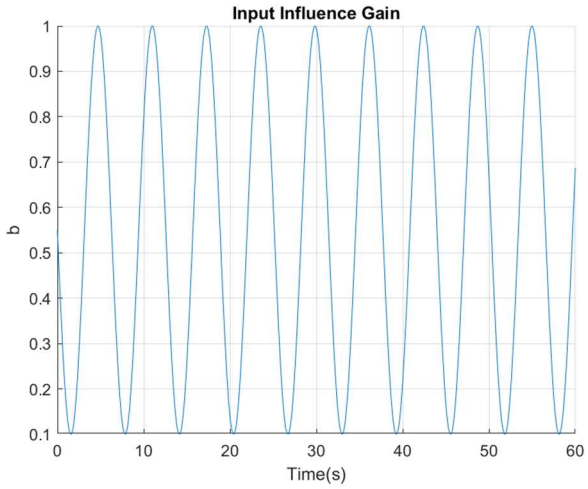

Figure 39: Position Tracking with Varying Input Influence Gain

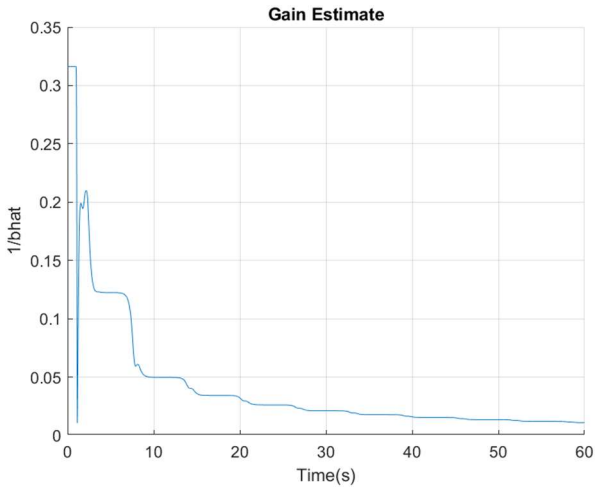

Figure 40: Position Tracking with Varying Input Influence Gain

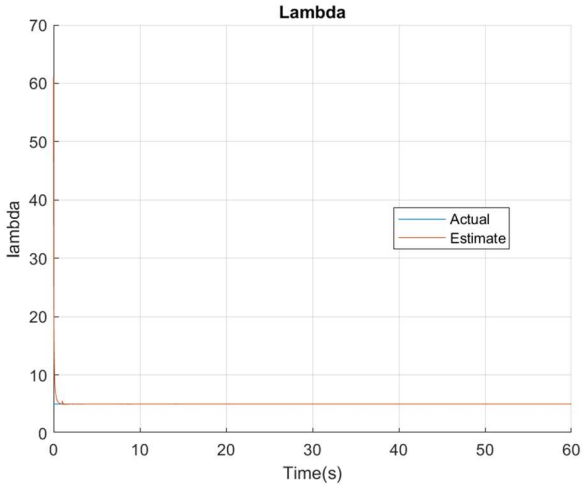

Figure 41: Roll Lambda Estimate

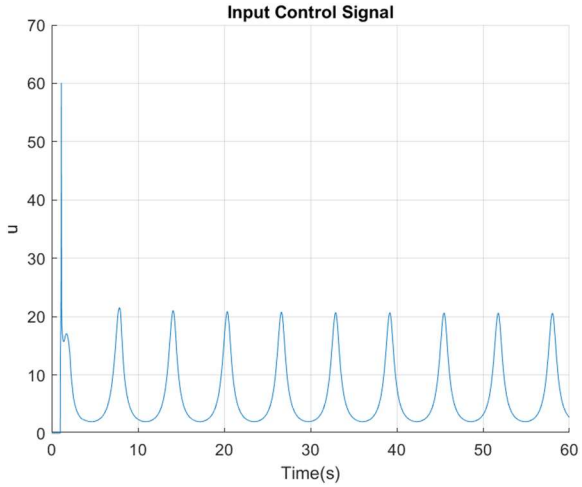

Figure 42: Roll Control Input

As seen in Figures 35 and 36, the controller still provides excellent tracking of roll rate and acceleration. Again, the sliding condition is met and the estimates for the input influence gain and λ are both convergent while the control input is smooth and adequately small.

3.3.4 Full Control of Lateral Directional State Space Model

The first aircraft representative system the controller was implemented on was a lateral/directional state space model:

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

Where:

$$A = \begin{pmatrix} -0.2316 & 0.0633 & -0.9956 & 0.0510 \\ -29.4924 & -3.0169 & 0.0201 & 0.0000 \\ 6.2346 & -0.0274 & -0.4169 & 0.0000 \\ 0.0000 & 1.0000 & 0.0631 & 0.0000 \end{pmatrix} B = \begin{pmatrix} -0.0052 & -0.0310 \\ 36.4909 & -8.1090 \\ 0.4916 & 2.8274 \\ 0.0000 & 0.0000 \end{pmatrix}$$

$$C = \begin{pmatrix} 1.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.0000 & 1.0000 & 0.0000 & 0.0000 \\ 0.0000 & 0.0000 & 1.0000 & 0.0000 \\ 0.0000 & 0.0000 & 0.0000 & 1.0000 \\ -29.4924 & -3.0169 & 0.0201 & 0.0000 \\ 6.2346 & -0.0274 & -0.4169 & 0.0000 \end{pmatrix} D = \begin{pmatrix} 0.0000 & 0.0000 \\ 0.0000 & 0.0000 \\ 0.0000 & 0.0000 \\ 0.0000 & 0.0000 \\ 36.4909 & -8.1090 \\ 0.4916 & 2.8274 \end{pmatrix}$$

And,

$$x = \begin{pmatrix} p \\ r \\ \dot{p} \\ \dot{r} \end{pmatrix}$$

Where p is roll rate and r is yaw rate. Our main goal was to control the system to track a roll rate step input while maintaining zero yaw rate or vice versa. Separate SISO control laws were used for roll and yaw rate. The roll rate step input was generated as described in Section 3.3.4. The yaw rate step input fed through a second-order transfer function:

$$\frac{r_d}{\delta_r} = \frac{\omega_n}{s^2 + 2\zeta\omega_n + {\omega_n}^2}$$

Where r_d is the desired yaw rate, δ_r is the rudder step input, ω_n is the Dutch roll natural frequency of 2 rad/sec and ζ is the damping ratio of 0.8 as characterized by a level 1 aircraft.

Two separate SISO control laws were used to control each variable. First, a desired roll step input of 20 rad/sec was generated with zero desired yaw rate. The results of this simulation are shown below:

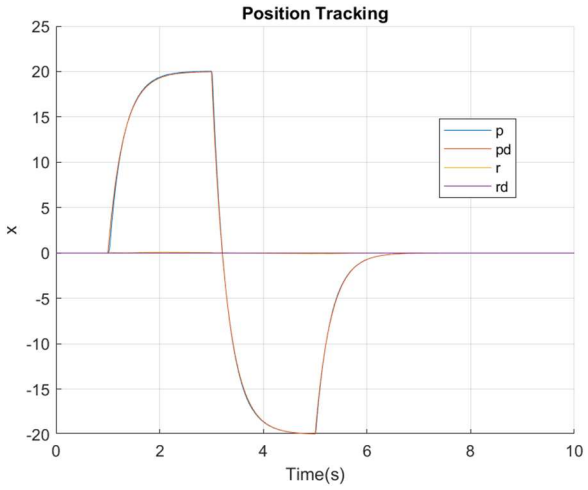

Figure 43: State Space Roll Input Position Tracking

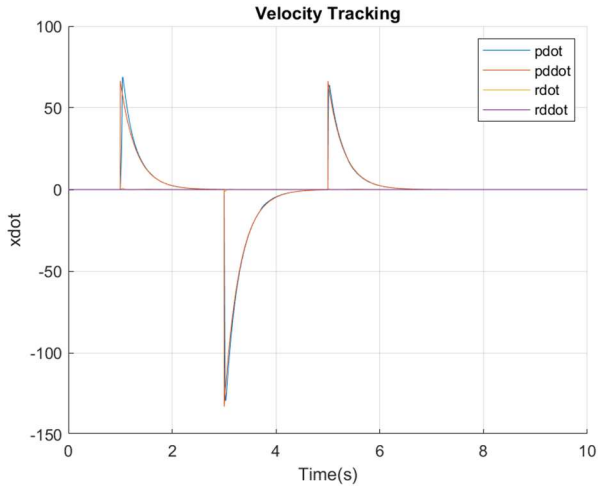

Figure 44: State Space Roll Input Velocity Tracking

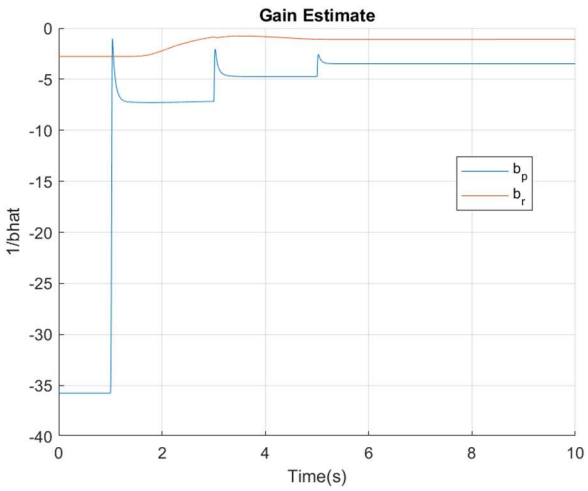

Figure 45: State Space Roll Input Input Influence Gain Estimate

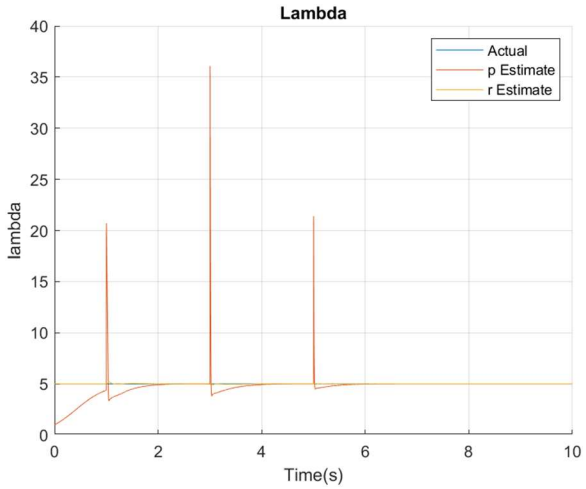

Figure 46: State Space Roll Input Lambda Estimate

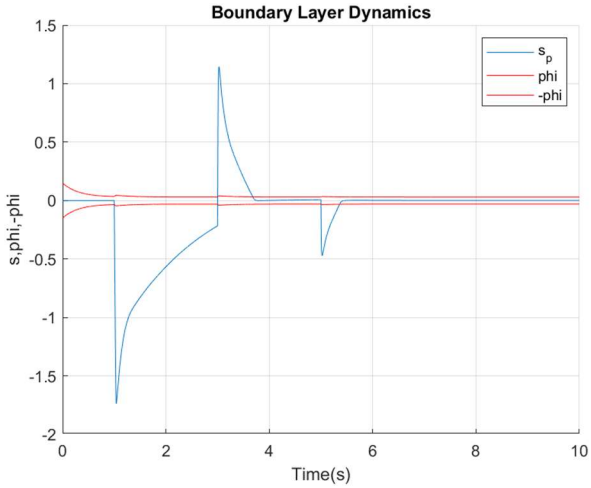

Figure 47: State Space Roll Input Roll Boundary Layer Dynamics

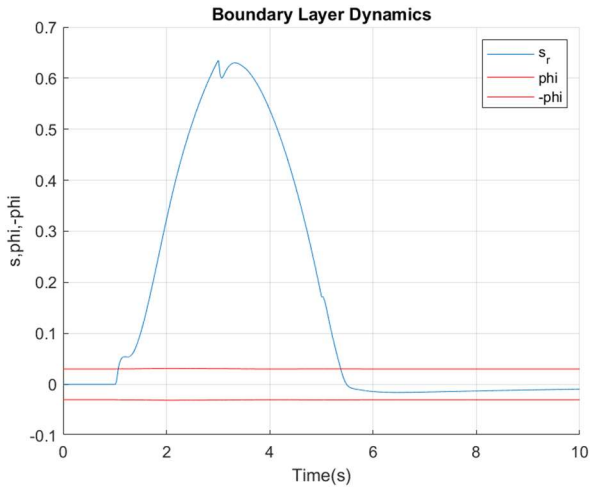

Figure 48: State Space Roll Input Yaw Boundary Layer Dynamics

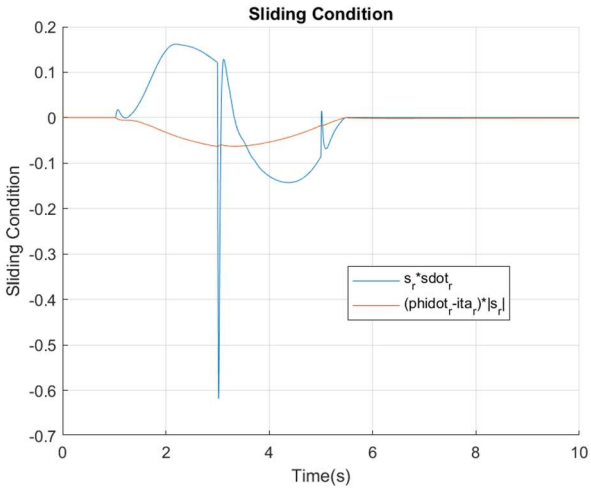

Figure 49: State Space Roll Input Roll Sliding Condition

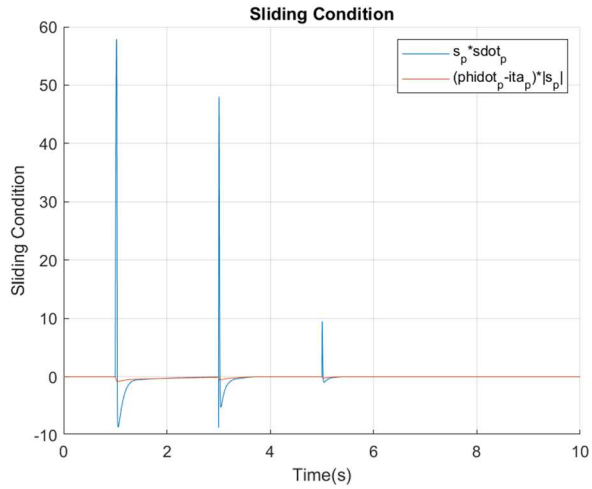

Figure 50: State Space Roll Input Yaw Sliding Condition

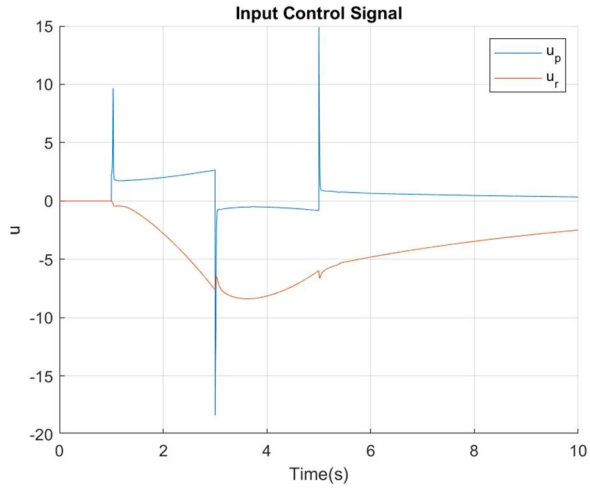

Figure 51: State Space Roll Input Control Input

Figures 43 and 44 show that the controller achieved very good roll and yaw rate and acceleration tracking. Figure 51 shows that the controller effort is reasonable. Figure 45 shows us that the input influence gain estimated for both roll and yaw converge to a single value. Figure 46 also shows this for the estimate of lambda where the estimate for roll and yaw both converge to the actual value of lambda. Most importantly, Figures 49 and 50 show that the sliding condition was met.

Next, a desired yaw step input of 10 rad/sec was generated with zero desired roll rate. The results of this simulation are shown below:

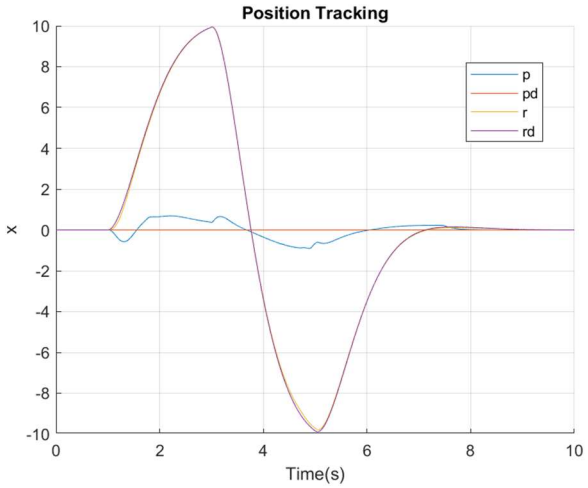

Figure 52: State Space Yaw Input Position Tracking

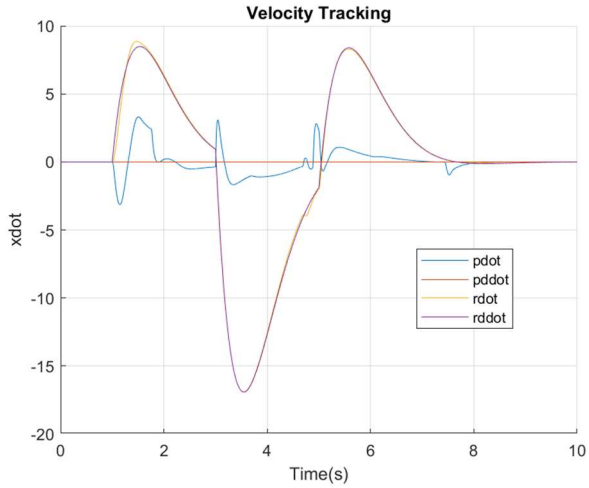

Figure 53: State Space Yaw Input Velocity Tracking

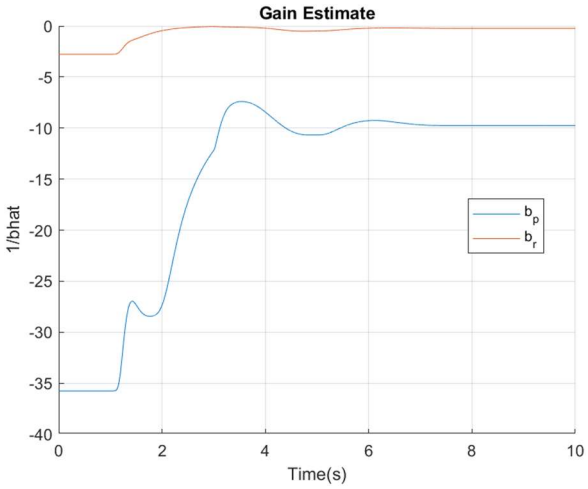

Figure 54: State Space Yaw Input Input Influence Gain Estimate

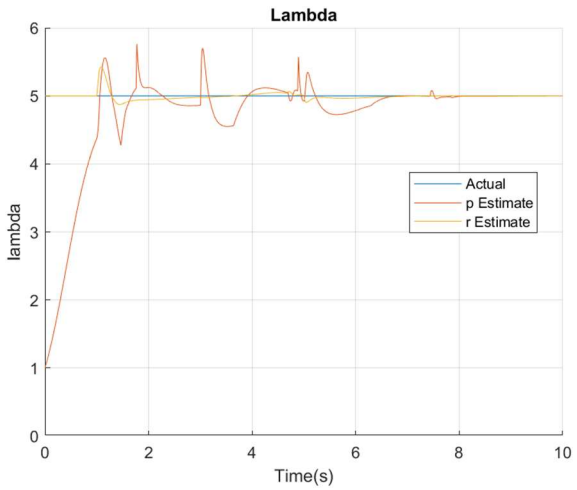

Figure 55: State Space Yaw Input Lambda Estimate

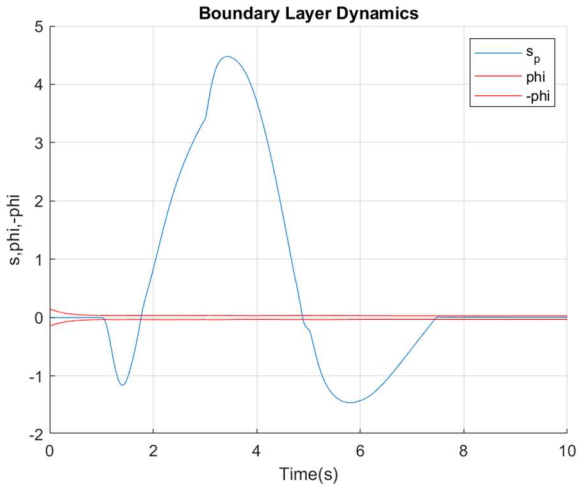

Figure 56: State Space Yaw Input Roll Boundary Layer Dynamics

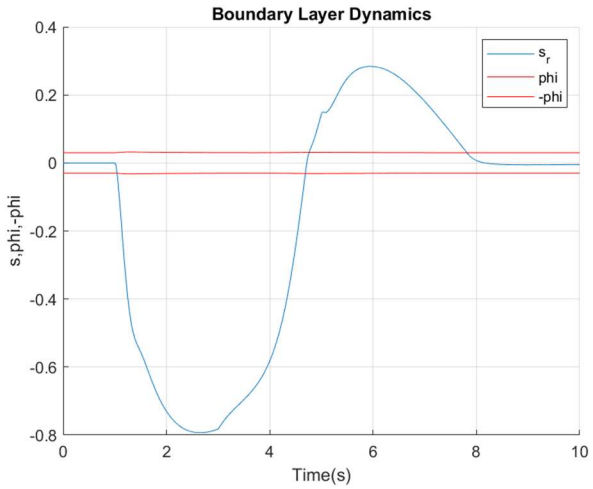

Figure 57: State Space Yaw Input Yaw Boundary Layer Dynamics

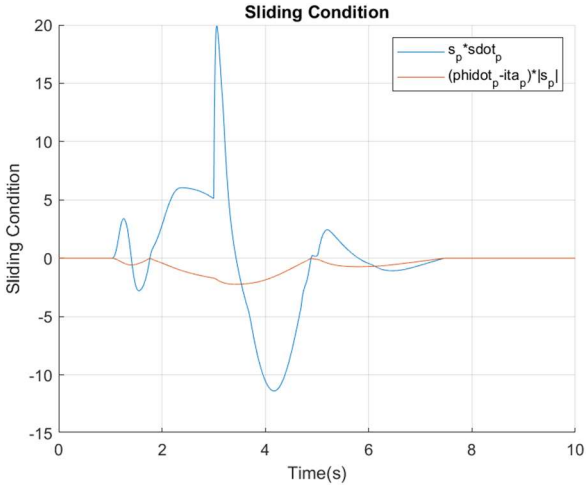

Figure 58: State Space Yaw Input Roll Sliding Condition

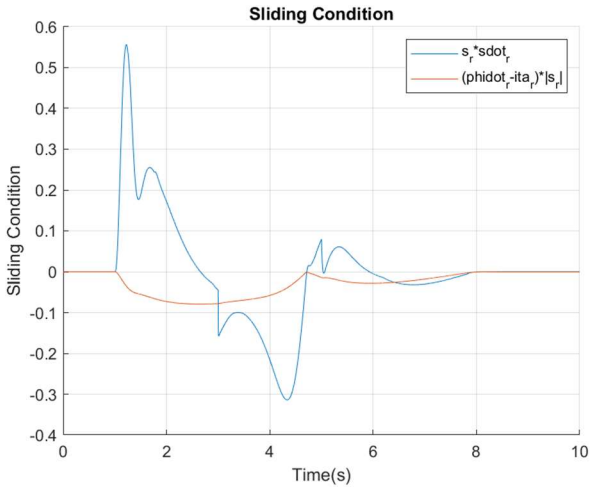

Figure 59: State Space Yaw Input Yaw Sliding Condition

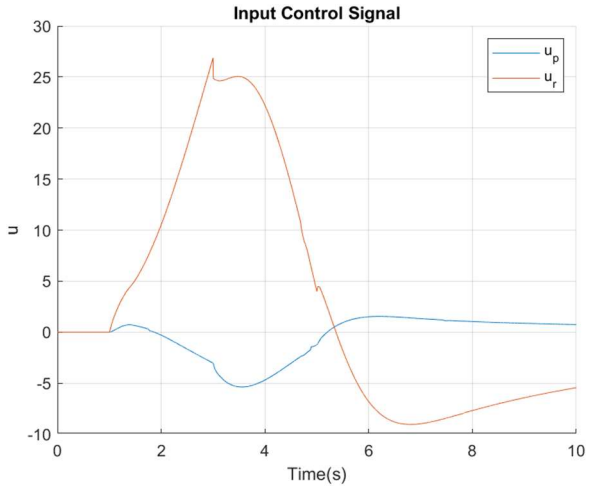

Figure 60: State Space Yaw Input Control Input

Again, Figures 52 and 53 show that the controller achieved very good roll and yaw rate and acceleration tracking. However, it did not perform as well as it did for a desired roll rate. Figure 60 shows that the controller effort is reasonable. Figure 54 shows us that the input influence gain estimated for both roll and yaw converge to a single value. Figure 55 also shows this for the estimate of lambda where the estimate for roll and yaw both converge to the actual value of lambda. Most importantly, Figures 58 and 59 show that the sliding condition was met.

3.3.5 Input Influence Gain Matrix MIMO Example

The MIMO control law derived in was implemented in the control of a MIMO system as defined by:

$$\ddot{x}_1 + a_1(t)\dot{x}_1^2 \cos(x_1)\, x_2 = 6u_1 + 4u_2$$

$$\ddot{x}_2 + a_2(t)\dot{x}_2^2 x_2 \dot{x}_1 = 2u_1 + 3u_2$$

$$1 \leq a_1(t) \leq 2$$

$$4 \leq a_2(t) \leq 6$$

Where a_1 and a_2 were made time-varying such that:

$$a(t) = \left(\frac{a_{upp} - a_{low}}{2}\right)\sin(t) + \frac{a_{upp} + a_{low}}{2}$$

Where a_{upp} and a_{low} are the upper and lower estimated bounds of the gains respectively. The input influence gains were also varied by adding to them a sine function defined by:

$$f(t) = \sin(3t)$$

The desired tracking was defined by:

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

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

Here, a single MIMO control law as derived in Section 3.1.12 was used to control both variables. However, the square input influence gain matrix was set to be constant and was not estimated in real-time. First, a constant boundary layer magnitude of 0.5 was used to show the effectiveness of the control law. The results are shown below:

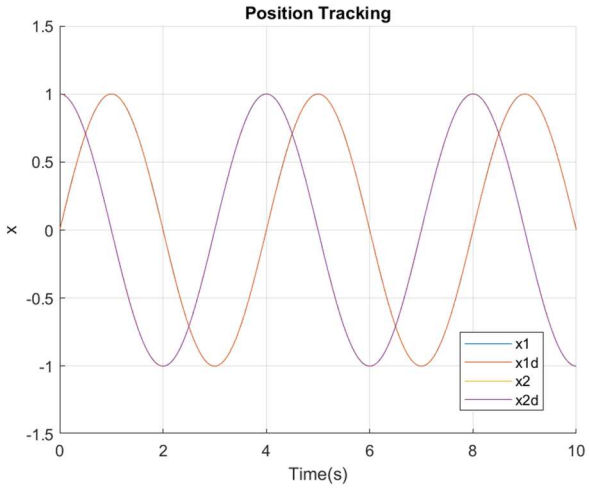

Figure 61: MIMO Gain Matrix Constant B.L. Position Tracking

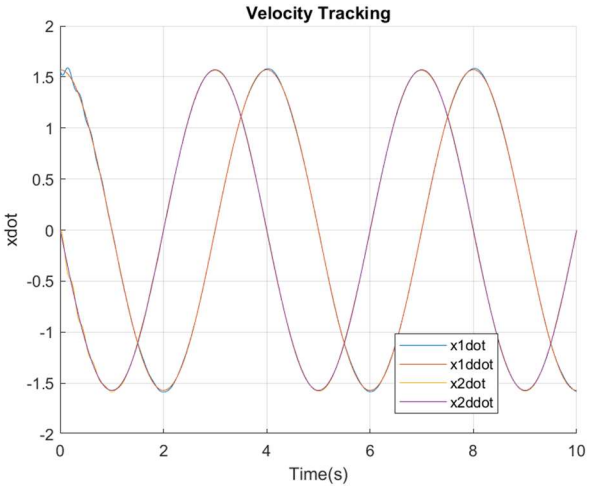

Figure 62: MIMO Gain Matrix Constant B.L. Velocity Tracking

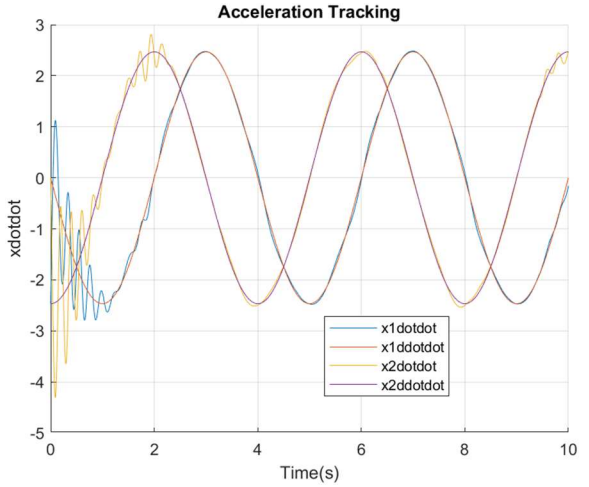

Figure 63: MIMO Gain Matrix Constant B.L. Acceleration Tracking

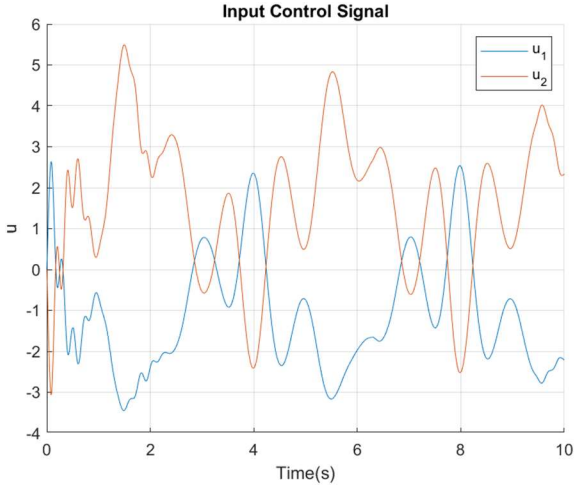

Figure 64: MIMO Gain Matrix Constant B.L. Controller Input

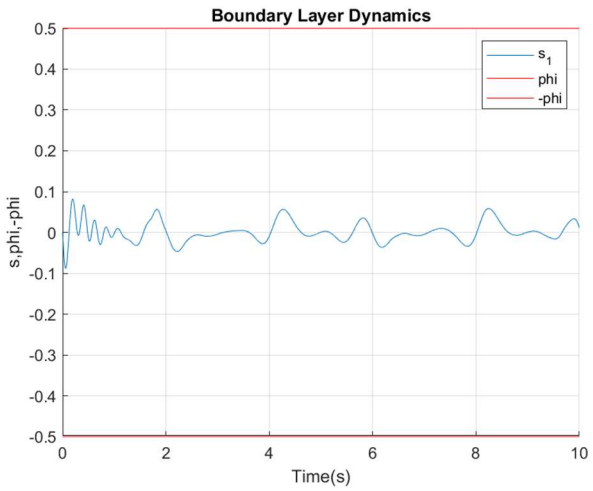

Figure 65: MIMO Gain Matrix Constant B.L. x_1 B.L Dynamics

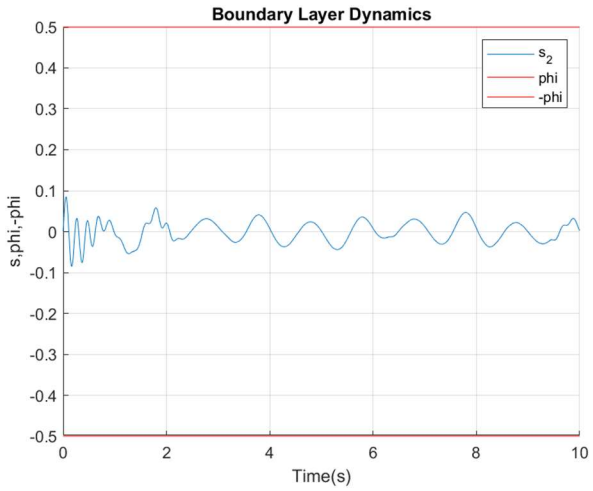

Figure 66: MIMO Gain Matrix Constant B.L. x_2 B.L Dynamics

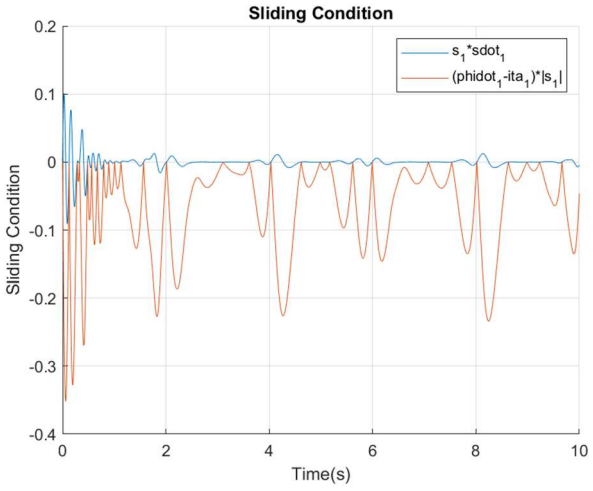

Figure 67: MIMO Gain Matrix Constant B.L. x_1 Sliding Condition

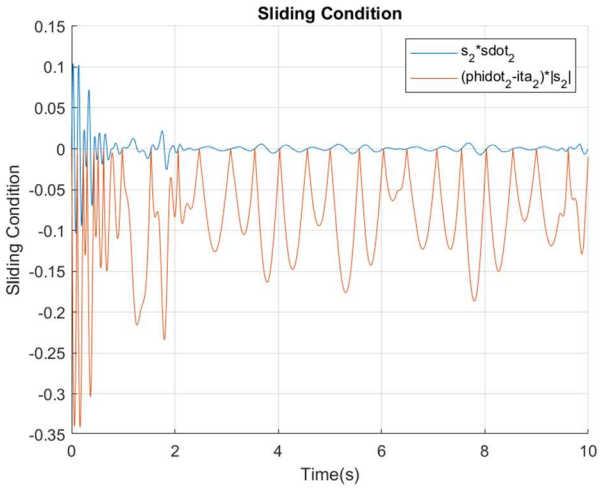

Figure 68: MIMO Gain Matrix Constant B.L. x_2 Sliding Condition

Figures 61, 62, and 63 show that this controller had a great tracking response for position, velocity, but there were a lot of initial oscillations in acceleration tracking. This was achieved with a smooth and small magnitude controller input as shown in Figure 64. The sliding surface stayed within the boundary layer and the sliding condition met for both variables, guaranteeing stability. In an effort to improve performance, the boundary layer was made time varying. The results of this are shown below:

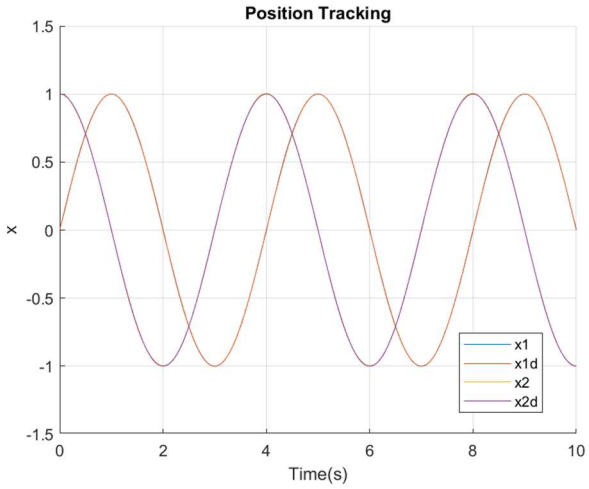

Figure 69: MIMO Gain Matrix Varying B.L. Position Tracking

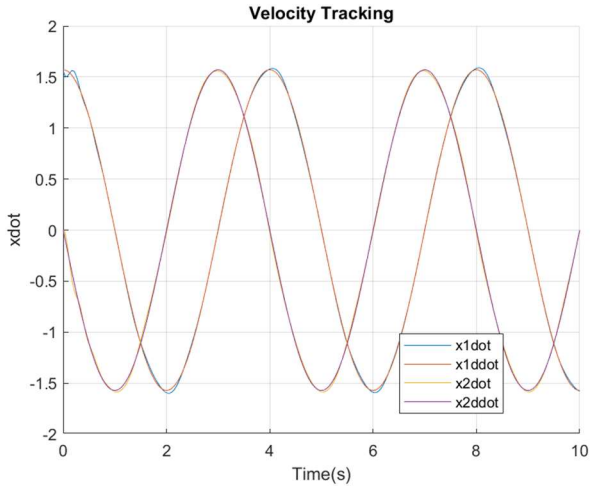

Figure 70: MIMO Gain Matrix Varying B.L. Velocity Tracking

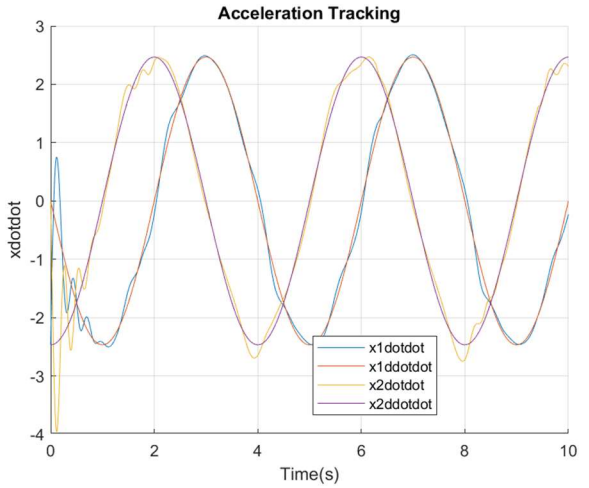

Figure 71: MIMO Gain Matrix Varying B.L. Acceleration Tracking

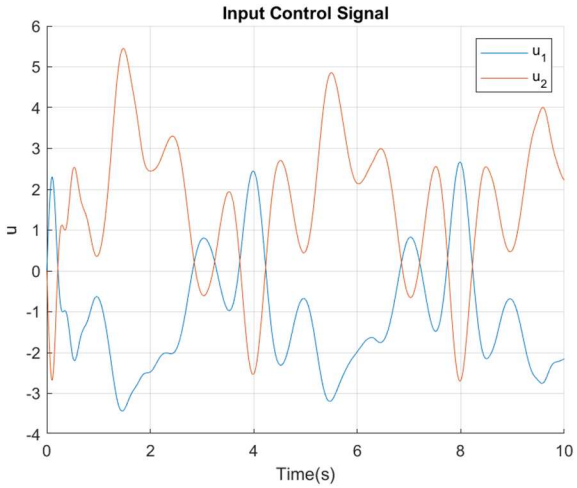

Figure 72: MIMO Gain Matrix Varying B.L. Controller Input

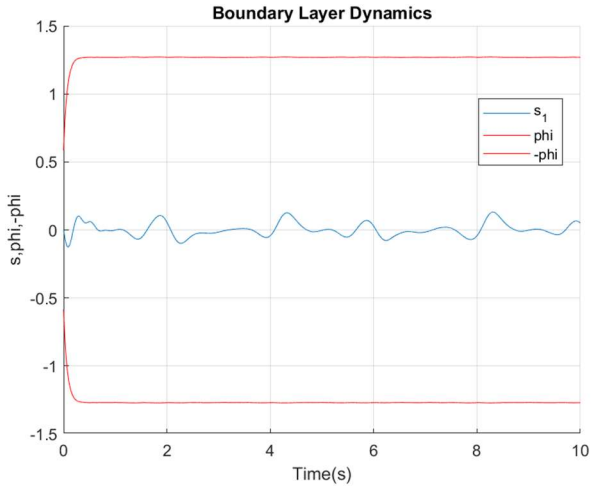

Figure 73: MIMO Gain Matrix Varying B.L. x_1 B.L Dynamics

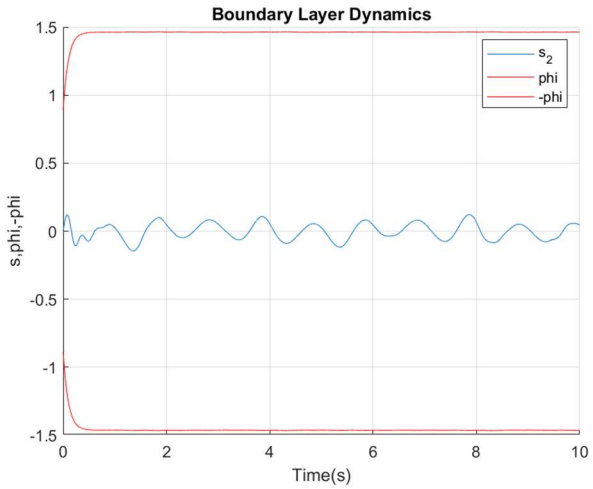

Figure 74: MIMO Gain Matrix Varying B.L. x_2 B.L Dynamics

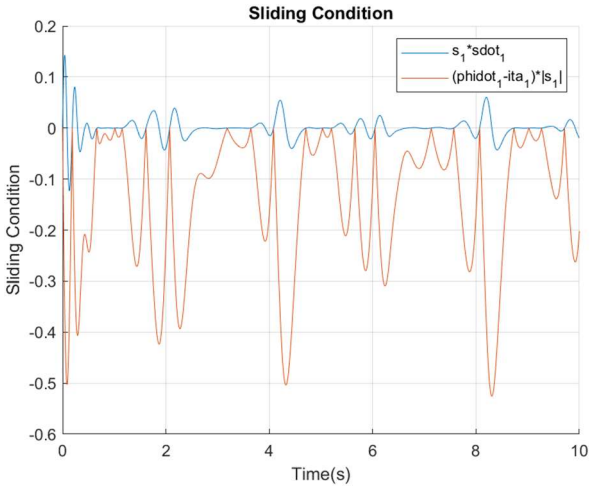

Figure 75: MIMO Gain Matrix Varying B.L. x_1 Sliding Condition

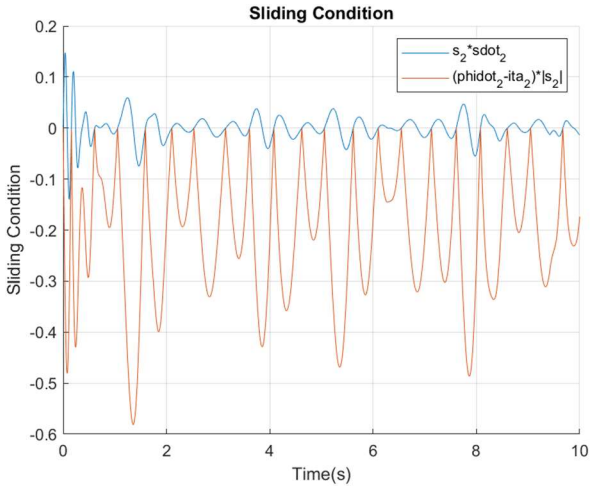

Figure 76: MIMO Gain Matrix Varying B.L. x_2 Sliding Condition

Figures 69 and 70 show that the controller retains its great position and velocity tracking. Figure 71 shows that there was slightly better acceleration tracking with less oscillation in the beginning of the simulation. There was still a smooth and small magnitude controller input. Again, the sliding surface stayed within the boundary layer and the sliding condition was met for both variables.

3.3.6 Input Influence Gain Matrix for State Space Model

The MIMO control method derived in Section 3.1.12 was implemented in the control of the lateral-direction state space model defined in Section 3.3.4. However, all of the elements in the B matrix were set to be positive. Again, the square input influence gain matrix in the control law was kept constant and was not estimated in real-time. A desired roll input with a magnitude of 20 was generated using the method defined in Section 3.3.3. The results of this simulation are shown below:

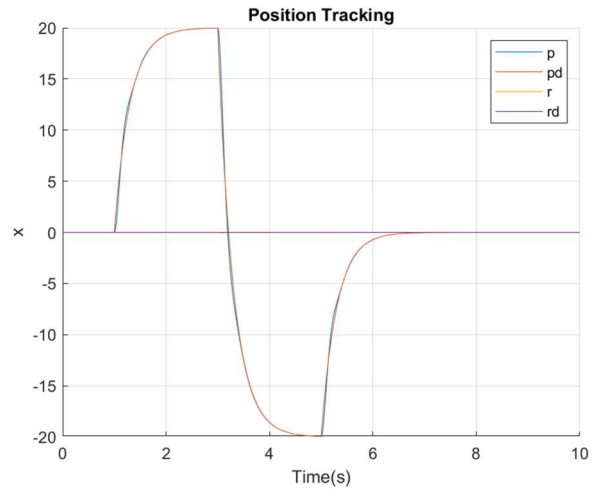

Figure 77: State Space Gain Matrix Roll Input Position Tracking

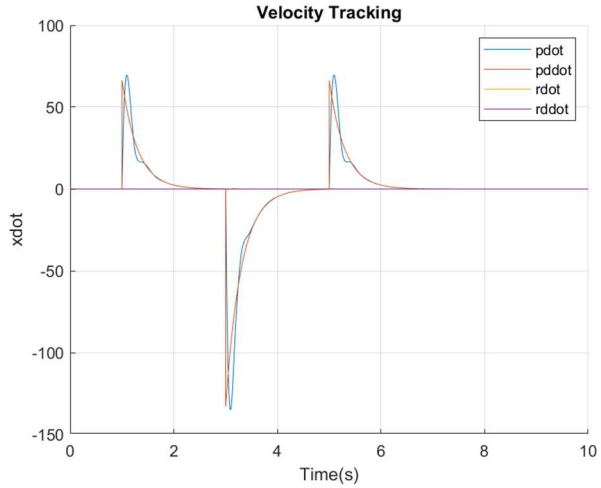

Figure 78: State Space Gain Matrix Roll Input Velocity Tracking

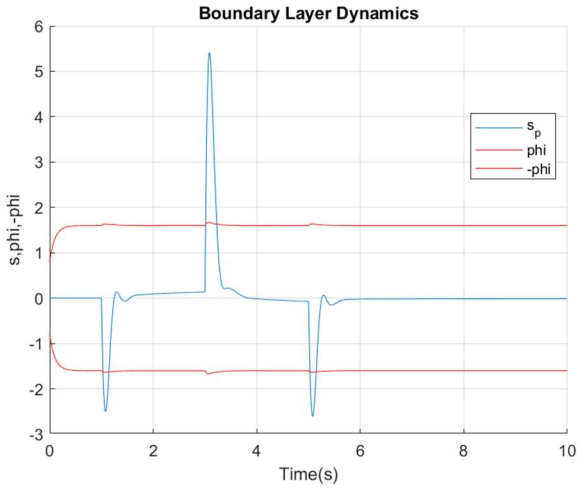

Figure 79: State Space Gain Matrix Roll Input Roll B.L Dynamics

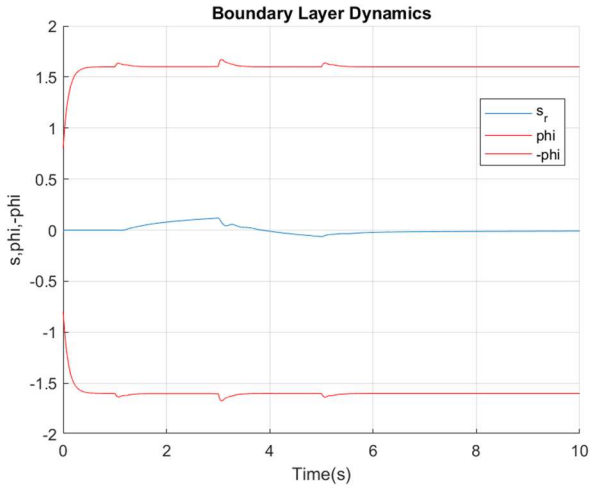

Figure 80: State Space Gain Matrix Roll Input Yaw B.L Dynamics

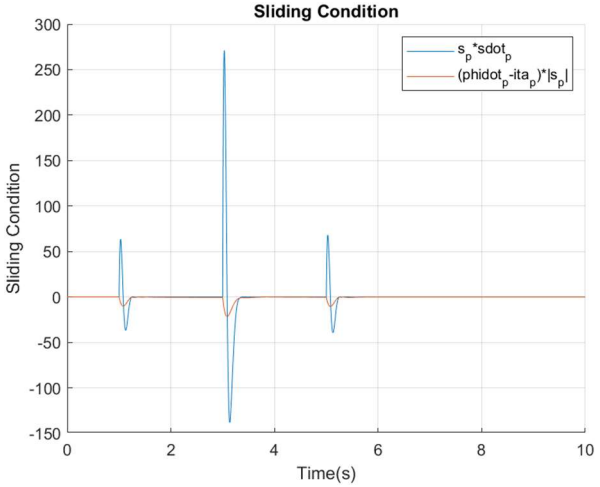

Figure 81: State Space Gain Matrix Roll Input Roll Sliding Condition

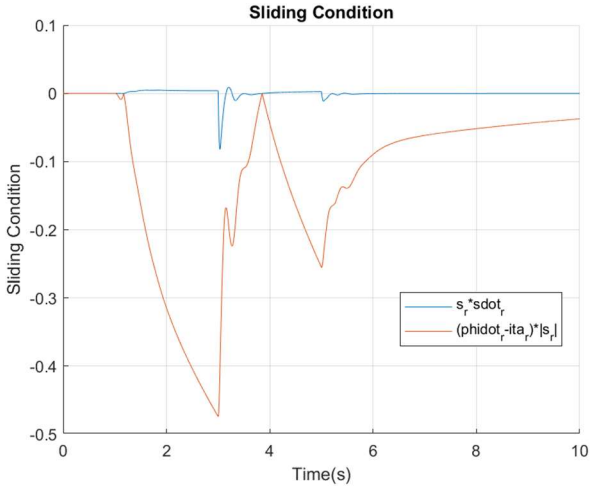

Figure 82: State Space Gain Matrix Roll Input Yaw Sliding Condition

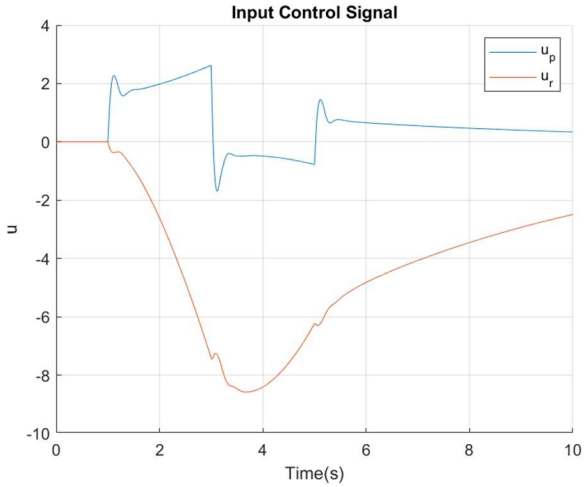

Figure 83: State Space Gain Matrix Roll Input Controller Input

Figure 77 shows that the controller had great roll and yaw rate tracking performance. The roll acceleration tracking shown in Figure 78 is slightly worse. Figures 79 and 81 show that the roll rate sliding surface does not remain within the boundary layer and the sliding condition was not met initially after each step input. However, the controller quickly counteracted the effects and brought the sliding surface back within the boundary layer and satisfies the sliding condition. The yaw rate sliding surface remains within the boundary layer and the sliding condition remains satisfied for the entire simulation time. This was achieved with a controller input that was smooth and had a small magnitude.

Next, a desired yaw step input of 10 rad/sec was generated with zero desired roll rate. The results of this simulation are shown below:

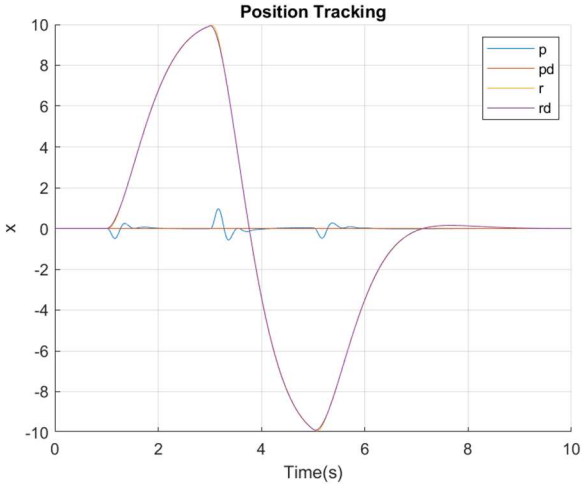

Figure 84: State Space Gain Matrix Yaw Input Position Tracking

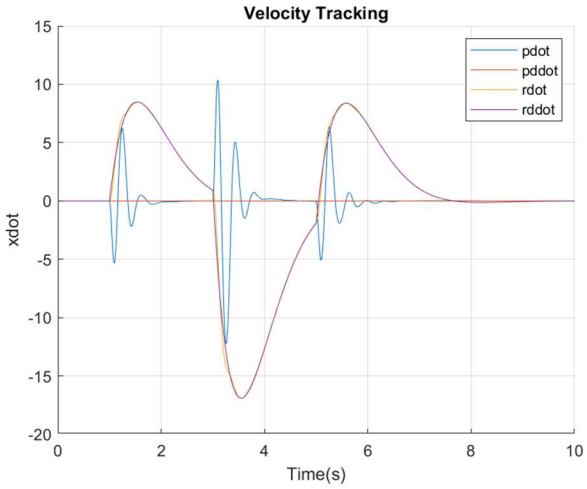

Figure 85: State Space Gain Matrix Yaw Input Velocity Tracking

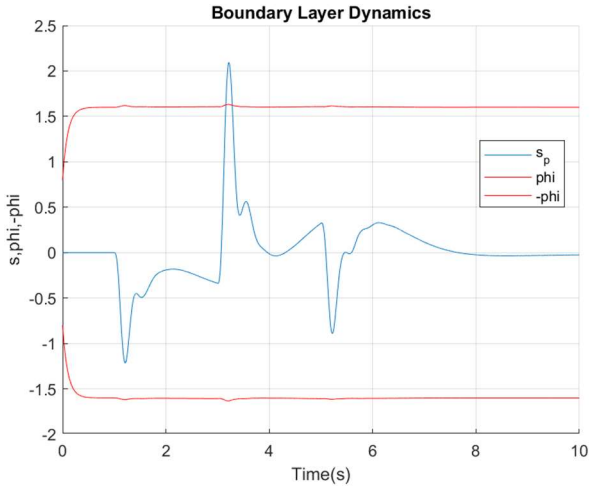

Figure 86: State Space Gain Matrix Yaw Input Roll B.L Dynamics

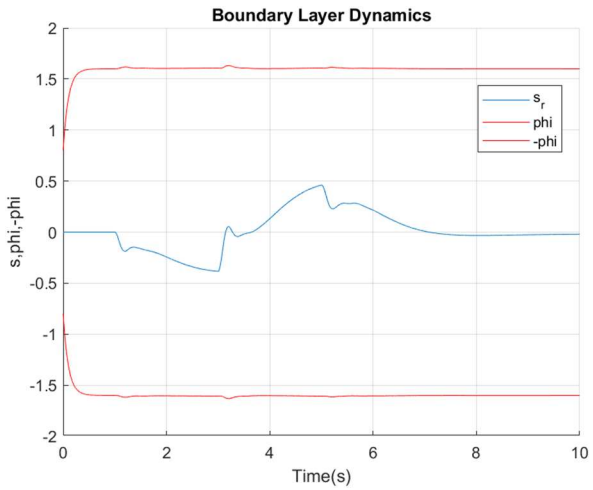

Figure 87: State Space Gain Matrix Yaw Input Yaw B.L Dynamics

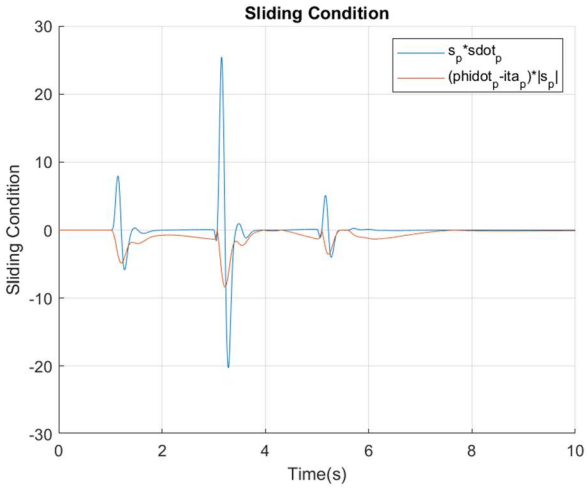

Figure 88: State Space Gain Matrix Yaw Input Roll Sliding Condition

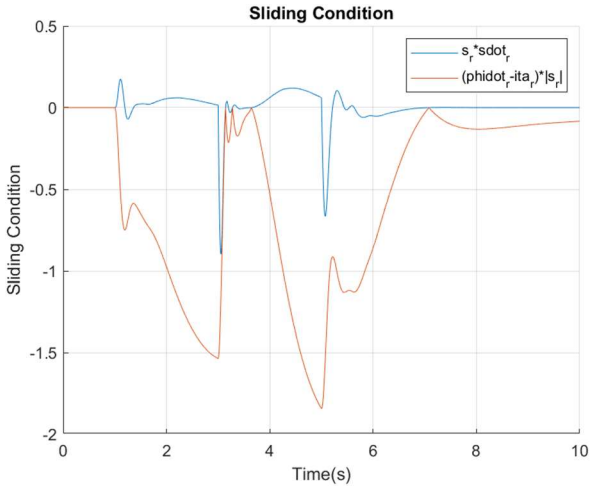

Figure 89: State Space Gain Matrix Yaw Input Yaw Sliding Condition

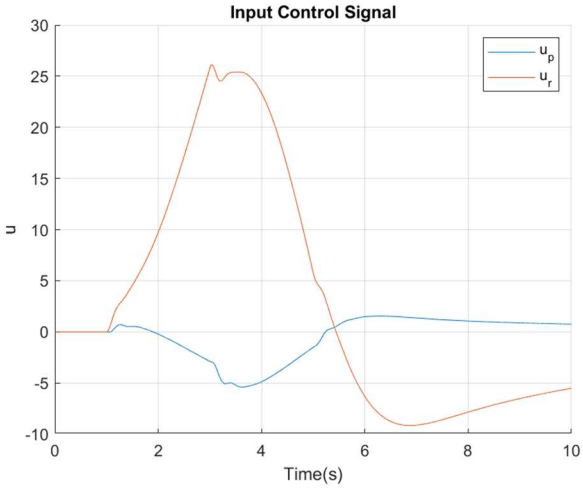

Figure 90: State Space Gain Matrix Yaw Input Controller Input

Figure 84 shows that again, the controller had great roll and yaw rate tracking performance. However, there was a lot of oscillations in the roll acceleration tracking shown in Figure 85. Figures 86 and 88 show that the roll rate sliding surface does not remain within the boundary layer and the sliding condition was not met initially after each step input. However, the controller quickly counteracted the effects and brought the sliding surface back within the boundary layer and satisfies the sliding condition. The yaw rate sliding surface remains within the boundary layer and the sliding condition remains satisfied for the entire simulation time. This was also achieved with a controller input that was smooth and had a small magnitude.

4.0 CONCLUSION

This work encompassed the development of a model-free sliding mode control system for the application of roll and yaw rate control for an aircraft. This control system is model-free in that no system model is needed to design a controller. The control input is calculated based on only knowledge of the system order, state measurements, and the previous control inputs. The model-free nature of this control system is enabled by the online estimation of the input influence gain. When implemented on SISO and MIMO nonlinear systems the control system provided good tracking performance while ensuring global asymptotic stability. It was also robust to modeling uncertainty. The least-squares online parameter estimation method used to estimate the input influence gain satisfied the sliding condition while guaranteeing global asymptotic stability. It also provided values that were convergent and realistic. Finally, this control method was shown to work on a lateral-directional state space model of an aircraft for roll and yaw rate control. Therefore, this method is feasible for roll and yaw rate control in an aircraft.

4.1 Application

The majority of existing control methods require extensive development and tuning. This is becoming a problem as nonlinear systems become more complex with advancements in technology. It is becoming time intensive to define a mathematical model of these systems. Oftentimes, the final models are not perfect and have parameter uncertainty. SMC is being applied more widely for its robustness against modeling uncertainty and external disturbances. Thus, SMC saves time and money by eliminating most of the development and tuning needed. However, accurate approximations of system dynamics are still needed for traditional SMC design.

The work laid out here is truly model-free, requiring only knowledge of the system order, state measurements, and the previous control inputs for derivation of the control law. Thus, the time, money, and effort needed for the system modeling and tuning usually required by traditional SMC and other control methods is eliminated. The work expands the MFSMC to MIMO systems which expands the utility of the MFSMC to a wide range of systems. This also includes the control of an aircraft, specifically in the lateral and directional axes. The MFSMC is theoretically the most efficient in terms of fuel-efficiency as the controller output is only dependent on state measurements. Thus, the development and application of the MFSMC to aircraft autopilot designs and other similar applications will increase fuel-efficiency and reduce emissions.

4.2 Future Work

The next phase for this work is to implement the real-time estimation of the input influence gain matrix for the MIMO control system. This would show that the MIMO control system is able to be completely model-free. Second, the MIMO control law would need to be implemented on a nonlinear aircraft model that is a closer representation of an actual aircraft. The next step would include the implementation of the control law on the hot bench of an aircraft to validate real-

world feasibility. Aircraft hot benches provide the most accurate representation of the aircraft without flight testing. Thus, this would be the final testing before flight testing can occur. There are a couple of other improvements needed to be made before flight testing as well. Currently, the gains on the transfer functions used to generate desired roll and yaw rates are unitary. These gains could be tuned to generate a favorable aircraft response based on pilot feedback. The simulations performed in this work also assumed that data was coming from perfect sensors. Therefore, this work can be expanded to test the controller's robustness against real-world degrading of measurement signals. This work will culminate with flight testing by the U.S. Air Force Test Pilot School to test if the control law is viable for lateral and directional control.

5.0 REFERENCES

[1] Reis, R. M., 2016, *A New Model-Free Sliding Mode Control Method with Estimation A New Model-Free Sliding Mode Control Method with Estimation of Control Input Error of Control Input Error*.

[2] Crassidis, A., and El Tin, F., 2017, "A Model-Free Control System Based on the Sliding Mode Control Method with Applications to Multi-Input-Multi-Output Systems," *International Conference of Control, Dynamic Systems, and Robotics*, Avestia Publishing.

[3] Islam, S., 2020, *A Model-Free Control System Based on the Sliding Mode Control with Automatic Tuning Using as On-Line Parameter Estimation Approach*.

[4] Stephens, S. S., Fischer, J. P. A., Pelletier, M.-A. C., Navarro, M. A., Sick, T. J., Trautschold, M. G., Cotting, M. C., and Crassidis, A., 2023, *A Limited Evaluation of a Model-Free Control Law System*.

[5] Yu, H., and Özgüner, Ü., 2006, "Adaptive Seeking Sliding Mode Control," *Proceedings of the American Control Conference*, pp. 4694–4699.

[6] Lee, S. H., and Choo Chung, C., 2003, "Sliding Mode Control Design Using Fast Output Sampling," *Proceedings of the IEEE Conference on Decision and Control*, Institute of Electrical and Electronics Engineers Inc., pp. 3543–3548.

[7] 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.

[8] Cunha, J. P. V. S., Hsu, L., Costa, R. R., and Lizarralde, F., 2003, "Output-Feedback Model-Reference Sliding Mode Control of Uncertain Multivariable Systems," IEEE Trans Automat Contr, **48**(12), pp. 2245–2250.

[9] Laghrouche, S., Plestan, F., and Glumineau, A., 2007, "Higher Order Sliding Mode Control Based on Integral Sliding Mode," Automatica, **43**(3), pp. 531–537.

[10] Jing, Y., Sun, H., Zhang, L., and Zhang, T., 2017, "Variable Speed Control of Wind Turbines Based on the Quasi-Continuous High-Order Sliding Mode Method," Energies (Basel), **10**(10).

[11] Nizar, A., Houda, B. M., and Said, N. A., 2013, "A New Sliding Function for Discrete Predictive Sliding Mode Control of Time Delay Systems," International Journal of Automation and Computing, **10**(4), pp. 288–295.

[12] Sudhir, and Swamp, A., 2016, "Second Order Sliding Mode Control for Quadrotor," *2016 IEEE 1st International Conference on Control, Measurement and Instrumentation, CMI 2016*, Institute of Electrical and Electronics Engineers Inc., pp. 92–96.

[13] Crassidis, A., and Mizov, A., 2015, *A Model-Free Control Algorithm Derived Using the Sliding Mode Control Method*.

[14] Schulken, E., 2017, *Investigations of Model-Free Sliding Mode Control Algorithms Investigations of Model-Free Sliding Mode Control Algorithms Including Application to Autonomous Quadrotor Flight Including Application to Autonomous Quadrotor Flight*.

[15] Raj, A., and Sreeraj, K., 2019, *A Model-Free Control Algorithm Based on the Sliding Mode A Model-Free Control Algorithm Based on the Sliding Mode Control Method with Applications to Unmanned Aircraft Systems Control Method with Applications to Unmanned Aircraft Systems*.

[16] Munoz-Vazquez, A. J., Parra-Vega, V., Sanchez, A., and Ramirez-Rodriguez, H., 2013, "A Passive Velocity Field for Navigation of Quadrotors with Model-Free Integral Sliding Mode Control," *2013 International Conference on Unmanned Aircraft Systems, ICUAS 2013 - Conference Proceedings*, pp. 49–58.

[17] Monti, C., 2020, *Model-Free Control of an Unmanned Aircraft Quadcopter Type System*.

[18] Vahidi, A., Stefanopoulou, A., and Peng, H., 2005, "Recursive Least Squares with Forgetting for Online Estimation of Vehicle Mass and Road Grade: Theory and Experiments," Vehicle System Dynamics, **43**(1), pp. 31–55.

[19] Sumantri, B., Uchiyama, N., and Sano, S., 2013, "Least Square Based Sliding Mode Control for a Quad-Rotor Helicopter," *2013 IEEE/SICE International Symposium on System Integration, SII 2013*, IEEE Computer Society, pp. 324–328.

[20] Valladolid, J. D., Ortiz, J. P., and Minchala, L. I., 2016, "Adaptive Quasi-Sliding Mode Control Based on a Recursive Weighted Least Square Estimator for a DC Motor," *2016 IEEE Conference on Control Applications, CCA 2016*, Institute of Electrical and Electronics Engineers Inc., pp. 886–890.

[21] Ding, J., 2018, "Recursive and Iterative Least Squares Parameter Estimation Algorithms for Multiple-Input–Output-Error Systems with Autoregressive Noise," Circuits Syst Signal Process, **37**(5), pp. 1884–1906.

[22] Ma, Z., Zhang, W., He, J., and Jin, H., 2022, "Multi-Parameter Online Identification of Permanent Magnet Synchronous Motor Based on Dynamic Forgetting Factor Recursive Least Squares," *Proceedings of 2022 IEEE 5th International Electrical and Energy*

Conference, CIEEC 2022, Institute of Electrical and Electronics Engineers Inc., pp. 4865–4870.

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

6.0 APPENDIX

6.1 Section 3.3.1 MATLAB Code and Simulink Diagrams

6.1.1 MATLAB Code

```
%Nick Hutson
%Thesis Proposal Model Free Control Example
clear
close all
clc

%define MFSMC gains
lambda = 5;    %normally 5 (rad/sec), higher values better tracking and less control
effort
N = 0.1;         %normally 0.1
sigmaU = 0.2;    %normally 0.2
phi = 0.1;

tau_act=0.1;   %normally 0.1 (sec)

%define parameters for input influence gain
const_parm_sw=0;           %=1, use constant "b" in system model; =0, use varying "b"
in system model
b_c=1;               %value of "b" in system model if const_parm_sw=1
b_upp=5;                   %upper value of "b" in system model FOR THESIS: nice to
know initial value of b, need it work once so we can start b at right value
b_low=1;                   %lower value of "b" in system model
gamma=sqrt(b_upp/b_low);   %initial estimate of "gamma"
gammaD=gamma;             %initial estimate of "gamma(xd), function of desired
states"
ghat=sqrt(b_low*b_upp);   %initial estimate of "g"
g_upp_percent=0.1;         %percentage of upper value of "g" used to estimate "gamma"
in realtime, normally 0.1
g_low_percent=0.1;         %percentage of lower value of "g" used to estimate "gamma"
in realtime, normally 0.1

%varying boundary layer IC
Kd0=gammaD*N;
phi0=(gammaD*Kd0)/lambda;

%gains for estimator for "g_hat"
ghat0=ghat;
P0=5;
lambda0_b=2;            %higher value cause faster convergence and boundary layer,
phi will be small along with control effort but more suspectible to noise effects
lambda_sw=1;            %=1, use varying lambda technique; =0, use constant lambda so
lambda=lambda0
k0=500;                 %higher value cause faster convergence and boundary layer,
phi will be small along with control effort (100 is nice)
close_phi_gap_sw=0;     %=0, don't close boundary layer gap; =1, close gap which
reduces "g" magnitude ('e1*(|s|>=phi)' block in 'K and phi' subsystem
```

```
%sign or saturation switching function flag
signum_sw=0;  %=1, signum function; =0, saturation function

%boundary layer phi flag
%phi_flag=0;   %=1, constant boundary layer; =0, varying boundary layer
%if(phi_flag==1)
 %  phi0 = phi;
%end

%"g_hat" switch
g_hat_sw=1;   %=0, use constant g_hat; =1, use g_hat from realtime estimator

%run simulation
dt=0.001;
tf=60;

%Initial Conditions
xdot0 = pi/2;
x0 = 0;

sim('ModelFreeSimTake4.slx')
```

6.1.2 Simulink Diagrams

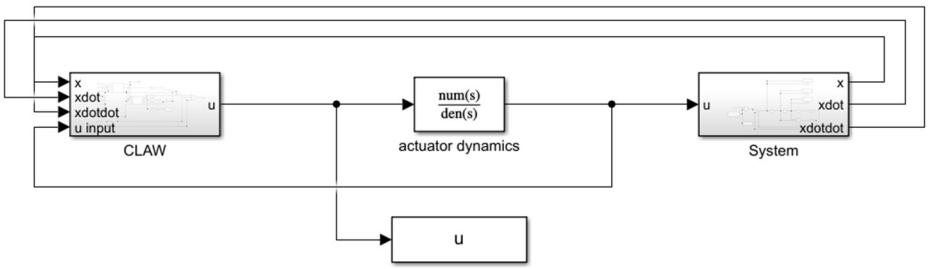

Figure A1: SISO System

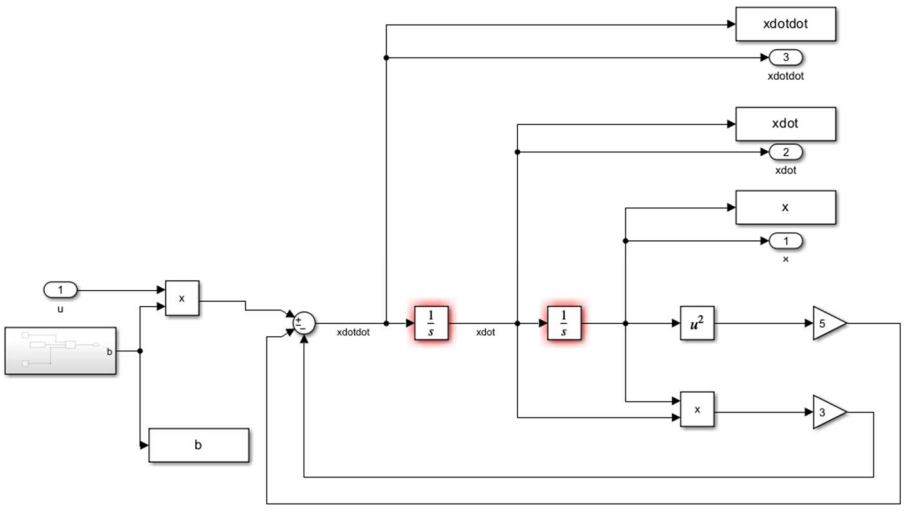

Figure A2: SISO System Model

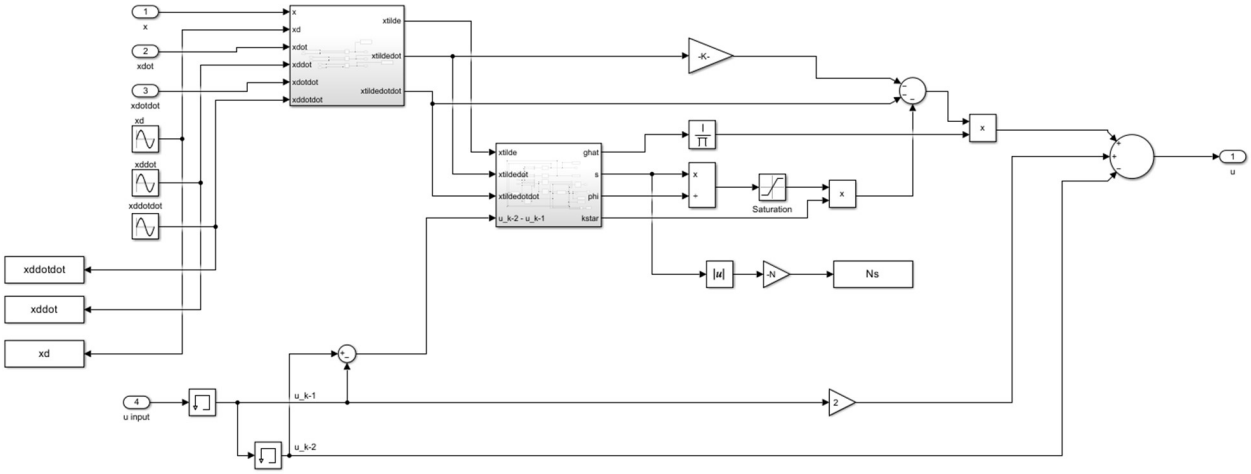

Figure A3: Second-Order SISO Control Law

6.2 Section 3.3.2 MATLAB Code and Simulink Diagrams

6.2.1 MATLAB Code

```
%Nick Hutson
%Thesis Proposal Model Free Control Example
clear
close all
clc

%define MFSMC gains
lambda = 5;    %normally 5 (rad/sec), higher values better tracking and less control
effort
N = 0.1;         %normally 0.1
sigmaU = 0.2;    %normally 0.2
phi = 0.1;

tau_act=0.1;   %normally 0.1 (sec)

%define parameters for input influence gain
const_parm_sw=0;           %=1, use constant "b" in system model; =0, use varying "b"
in system model
g_upp_percent=0.1;         %percentage of upper value of "g" used to estimate "gamma"
in realtime, normally 0.1
g_low_percent=0.1;         %percentage of lower value of "g" used to estimate "gamma"
in realtime, normally 0.1
b1_c=5;               %value of "b" in system model if const_parm_sw=1
b1_upp=6;                  %upper value of "b" in system model FOR THESIS: nice to
know initial value of b, need it work once so we can start b at right value
b1_low=4;                  %lower value of "b" in system model
gamma1=sqrt(b1_upp/b1_low);   %initial estimate of "gamma"
gamma1D=gamma1;            %initial estimate of "gamma(xd), function of desired
states"
ghat1=sqrt(b1_low*b1_upp);   %initial estimate of "g"
b2_c=3;               %value of "b" in system model if const_parm_sw=1
b2_upp=4;                  %upper value of "b" in system model FOR THESIS: nice to
know initial value of b, need it work once so we can start b at right value
b2_low=2;                  %lower value of "b" in system model
```

```
gamma2=sqrt(b2_upp/b2_low);   %initial estimate of "gamma"
gamma2D=gamma2;              %initial estimate of "gamma(xd), function of desired
states"
ghat2=sqrt(b2_low*b2_upp);   %initial estimate of "g"

%varying boundary layer IC
Kd10=gamma1D*N;
phi10=(gamma1D*Kd10)/lambda;
Kd20=gamma2D*N;
phi20=(gamma2D*Kd20)/lambda;

%gains for estimator for "g_hat"
ghat10=ghat1;
ghat20=ghat2;
P10=5;
P20=5;
lambda0_b=2;            %higher value cause faster convergence and boundary layer,
phi will be small along with control effort but more suspectible to noise effects
lambda_sw=1;            %=1, use varying lambda technique; =0, use constant lambda so
lambda=lambda0
k0=500;                 %higher value cause faster convergence and boundary layer,
phi will be small along with control effort (100 is nice)
close_phi_gap_sw=0;     %=0, don't close boundary layer gap; =1, close gap which
reduces "g" magnitude ('e1*(|s|>=phi)' block in 'K and phi' subsystem

%sign or saturation switching function flag
signum_sw=0;  %=1, signum function; =0, saturation function

%boundary layer phi flag
%phi_flag=0;   %=1, constant boundary layer; =0, varying boundary layer
%if(phi_flag==1)
 %  phi0 = phi;
%end

%"g_hat" switch
g_hat_sw=1;   %=0, use constant g_hat; =1, use g_hat from realtime estimator

%run simulation
dt=0.001;
tf=60;

%Initial Conditions
x1dot0 = pi/2;
x10 = 0;
x2dot0 = pi/4;
x20 = 0;

sim('ModelFreeSimTakeMIMO.slx')
```

6.2.2 Simulink Diagrams

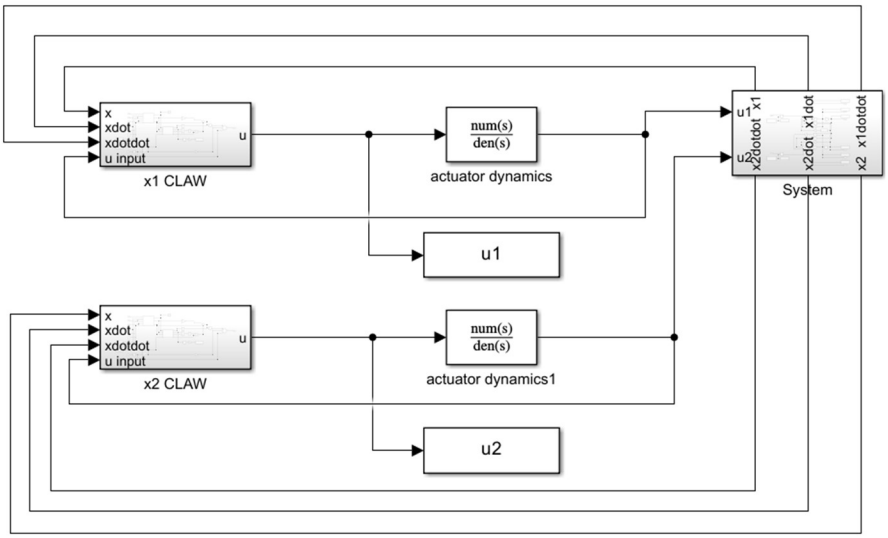

Figure A4: Second-Order MIMO System

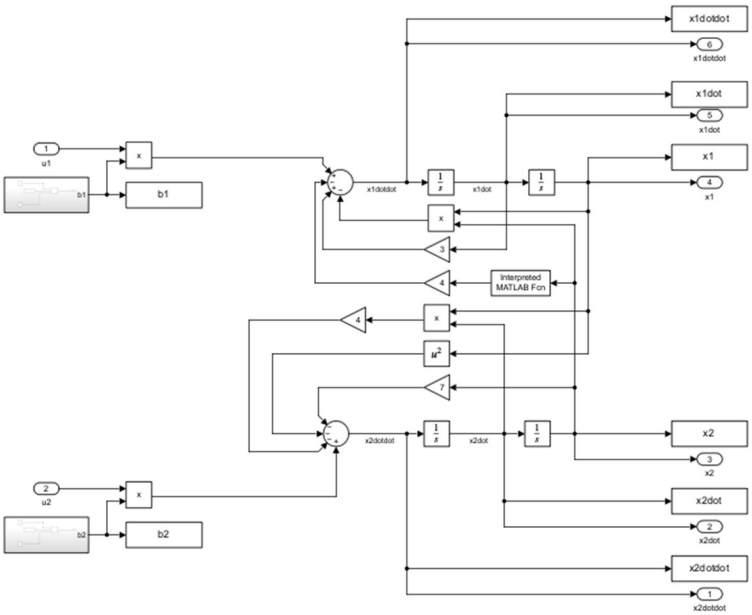

Figure A5: Second-Order MIMO System Model

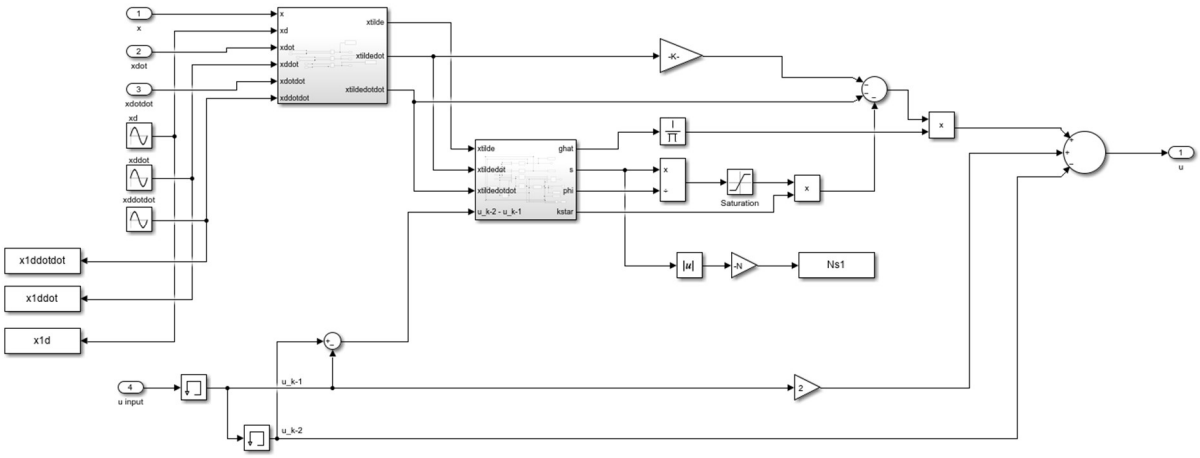

Figure A6: Second-Order MIMO Control Law

6.3 Section 3.3.3 MATLAB Code and Simulink Diagrams

6.3.1 MATLAB Code

```
%Nick Hutson
%Single Order Example
clear
close all
clc

roll = 1; %roll input
yaw = 0; %yaw input
K_roll = 1;

zeta_d = 0.19; %min of 0.19
w_nd = 0.4; %rad/s, min of 0.4 rad/sec
tau = 0.3; %s, max of 1.4s

%Initial Conditions
x0 = 0;
xtildeInt0 = 0;

%define MFSMC gains
lambda = 5;    %normally 5 (rad/sec), higher values better tracking and less control
effort
N = 0.1;         %normally 0.1
sigmaU = 0.2;    %normally 0.2
phi_mag = 0.1;

tau_act=0.1;   %normally 0.1 (sec)

%define parameters for input influence gain
const_parm_sw=0;           %=1, use constant "b" in system model; =0, use varying "b"
in system model
b_c=0.15;               %value of "b" in system model if const_parm_sw=1
```

```
b_upp=0.1;                    %upper value of "b" in system model FOR THESIS: nice to
know initial value of b, need it work once so we can start b at right value
b_low=1;                     %lower value of "b" in system model
gamma=sqrt(b_upp/b_low);    %initial estimate of "gamma"
gammaD=gamma;               %initial estimate of "gamma(xd), function of desired
states"
ghat=sqrt(b_low*b_upp);     %initial estimate of "g"
g_upp_percent=0.1;          %percentage of upper value of "g" used to estimate "gamma"
in realtime, normally 0.1
g_low_percent=0.1;          %percentage of lower value of "g" used to estimate "gamma"
in realtime, normally 0.1

%varying boundary layer IC
phi_flag=0;   %=1, constant boundary layer; =0, varying boundary layer
if( phi_flag<0.5 )
    Kd0=gammaD*N;
    phi0=(gammaD*Kd0)/lambda;
else
    phi0=phi_mag;
end

%gains for estimator for "g_hat"
ghat0=ghat;
P0=5;
lambda0_b=2;            %higher value cause faster convergence and boundary layer,
phi will be small along with control effort but more suspectible to noise effects
lambda_sw=1;            %=1, use varying lambda technique; =0, use constant lambda so
lambda=lambda0
k0=500;                 %higher value cause faster convergence and boundary layer,
phi will be small along with control effort (100 is nice)
close_phi_gap_sw=0;     %=0, don't close boundary layer gap; =1, close gap which
reduces "g" magnitude ('e1*(|s|>=phi)' block in 'K and phi' subsystem

%sign or saturation switching function flag
signum_sw=0;  %=1, signum function; =0, saturation function

%"g_hat" switch
g_hat_sw=1;   %=0, use constant g_hat; =1, use g_hat from realtime estimator

%run simulation
dt=0.001;
tfin=60;

sim('ModelFreeSimTakeSingleOrderTF.slx')
```

6.3.2 Simulink Diagrams

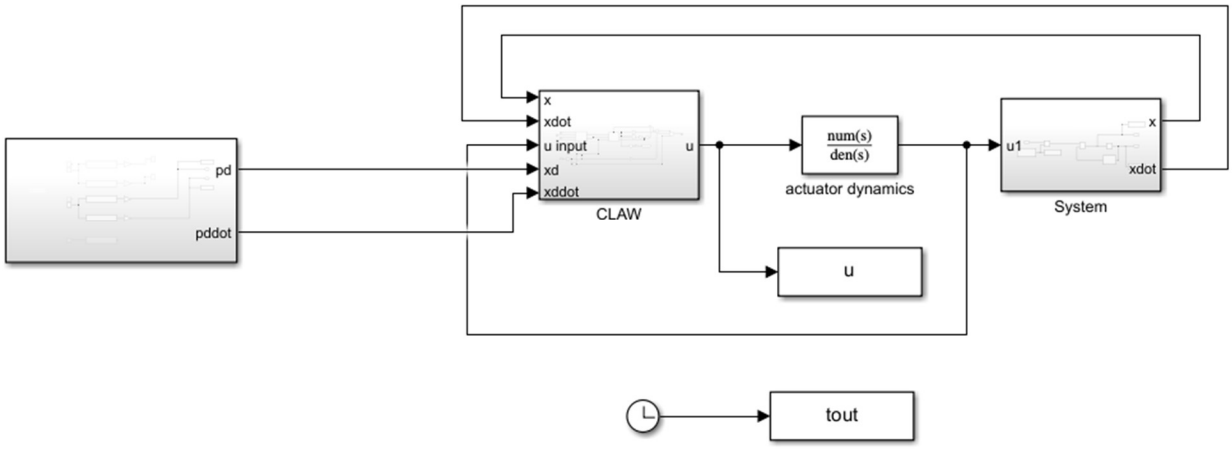

Figure A7: First-Order SISO System

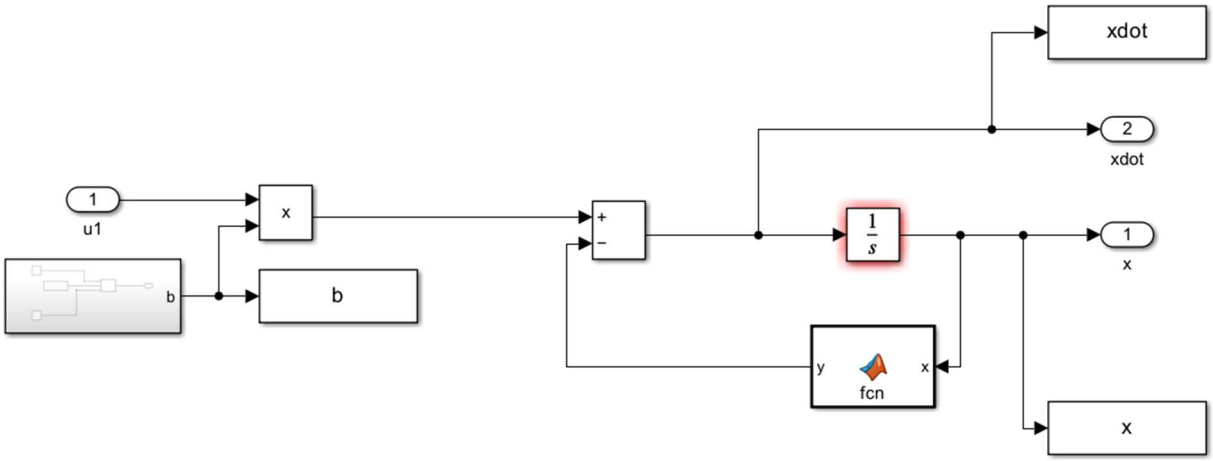

Figure A8: First-Order SISO System Model

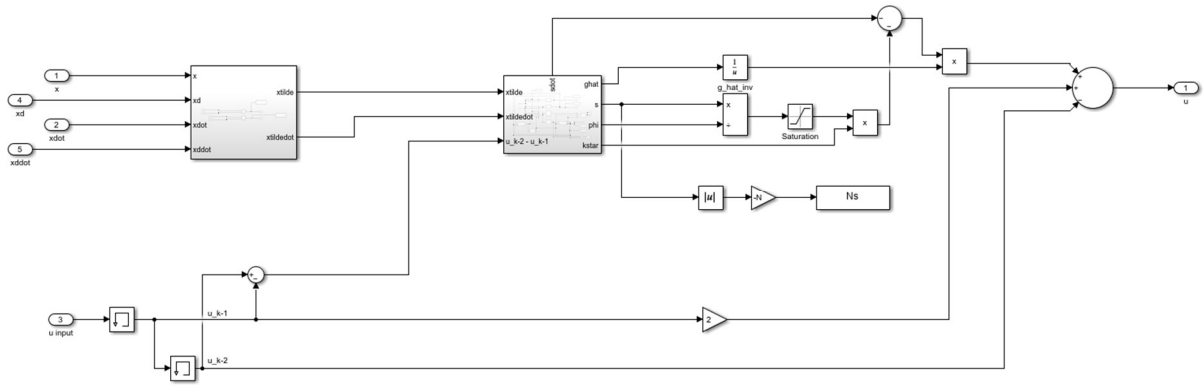

Figure A9: First-Order SISO Control Law

6.4 Section 3.3.4 MATLAB Code and Simulink Diagrams

6.4.1 MATLAB Code

```
%Nick Hutson
%Thesis Proposal Model Free Control Lateral State Space
clear
close all
clc

%% Define open-loop model
Alat=[ -0.2316  0.0633 -0.9956  0.051
      -29.4924 -3.0169  0.0201  0.0
        6.2346 -0.0274 -0.4169  0.0
        0.0     1.0     0.0631  0.0   ];
Blat=-[  0.0052  0.031
      -36.4909  8.1090
       -0.4916 -2.8274
        0.0     0.0    ];
Clat=eye(4);Clat(5:6,:)=Alat(2:3,:);
Dlat=zeros(4,2);Dlat(5:6,:)=Blat(2:3,:);
C = Clat([2,3,5,6],:);
D = Dlat([2,3,5,6],:);
%% Simulate
load('AircraftModelParam.mat')

%define desired roll and yaw
roll = 0; %assume to be stick displacement in inches or force in lb
yaw = 10;
K_roll = 1;
K_yaw = 1;

%define MFSMC gains
lambda = 5;    %normally 5 (rad/sec), higher values better tracking and less control
effort
N_da = 0.5;         %normally 0.1
N_dr = 0.1;         %normally 0.1
sigmaU = 0.2;    %normally 0.2
phi_mag = 10;

tau_act=0.08;   %normally 0.1 (sec)

%define Level 1 aircraft TFs
zeta_d = 0.8; %min of 0.19
w_nd = 2; %rad/s, min of 0.4 rad/sec
tau = 0.3; %s, max of 1.4s

%define parameters for input influence gain
const_parm_sw=1;           %=1, use constant "b" in system model; =0, use varying "b"
in system model
g_upp_percent=0.2;         %percentage of upper value of "g" used to estimate "gamma"
in realtime, normally 0.1
g_low_percent=0.2;         %percentage of lower value of "g" used to estimate "gamma"
in realtime, normally 0.1
b1_c=-Blat(2,1);               %value of "b" in system model if const_parm_sw=1
```

```
b1_upp=b1_c+g_upp_percent*b1_c;                  %upper value of "b" in system model
FOR THESIS: nice to know initial value of b, need it work once so we can start b at
right value
b1_low=b1_c-g_low_percent*b1_c;                  %lower value of "b" in system model
gamma1=sqrt(b1_upp/b1_low);   %initial estimate of "gamma"
gamma1D=gamma1;             %initial estimate of "gamma(xd), function of desired
states"
ghat1=sqrt(b1_low*b1_upp);   %initial estimate of "g"
b2_c=-Blat(3,2);               %value of "b" in system model if const_parm_sw=1
b2_upp=b2_c+g_upp_percent*b2_c;                  %upper value of "b" in system model
FOR THESIS: nice to know initial value of b, need it work once so we can start b at
right value
b2_low=b2_c-g_low_percent*b2_c;                  %lower value of "b" in system model
gamma2=sqrt(b2_upp/b2_low);   %initial estimate of "gamma"
gamma2D=gamma2;             %initial estimate of "gamma(xd), function of desired
states"
ghat2=sqrt(b2_low*b2_upp);   %initial estimate of "g"

%varying boundary layer IC
Kd10=gamma1D*N_da;
phi10=(gamma1D*Kd10)/lambda;
Kd20=gamma2D*N_dr;
phi20=(gamma2D*Kd20)/lambda;

%gains for estimator for "g_hat"
ghat10=ghat1;
ghat20=ghat2;
P10=5;
P20=5;
xtildeInt0 = 0;
lambda0_b=2;            %higher value cause faster convergence and boundary layer,
phi will be small along with control effort but more suspectible to noise effects
lambda_sw=1;            %=1, use varying lambda technique; =0, use constant lambda so
lambda=lambda0
k0=100;                 %higher value cause faster convergence and boundary layer,
phi will be small along with control effort (100 is nice)
close_phi_gap_sw=0;     %=0, don't close boundary layer gap; =1, close gap which
reduces "g" magnitude ('e1*(|s|>=phi)' block in 'K and phi' subsystem

%boundary layer phi flag
phi_flag=0;   %=1, constant boundary layer; =0, varying boundary layer
if( phi_flag<0.5 )
    phi0_da=(gamma1D*Kd10)/lambda;
    phi0_dr=(gamma2D*Kd20)/lambda;
else
    phi0_da=phi_mag;
    phi0_dr=phi_mag;
end

%"g_hat" switch
g_hat_sw=1;   %=0, use constant g_hat; =1, use g_hat from realtime estimator

%run simulation
dt=0.001;
tfin=10;
```

```
sim('ModelFreeSimTakeLatSS.slx')
```

6.4.2 Simulink Diagrams

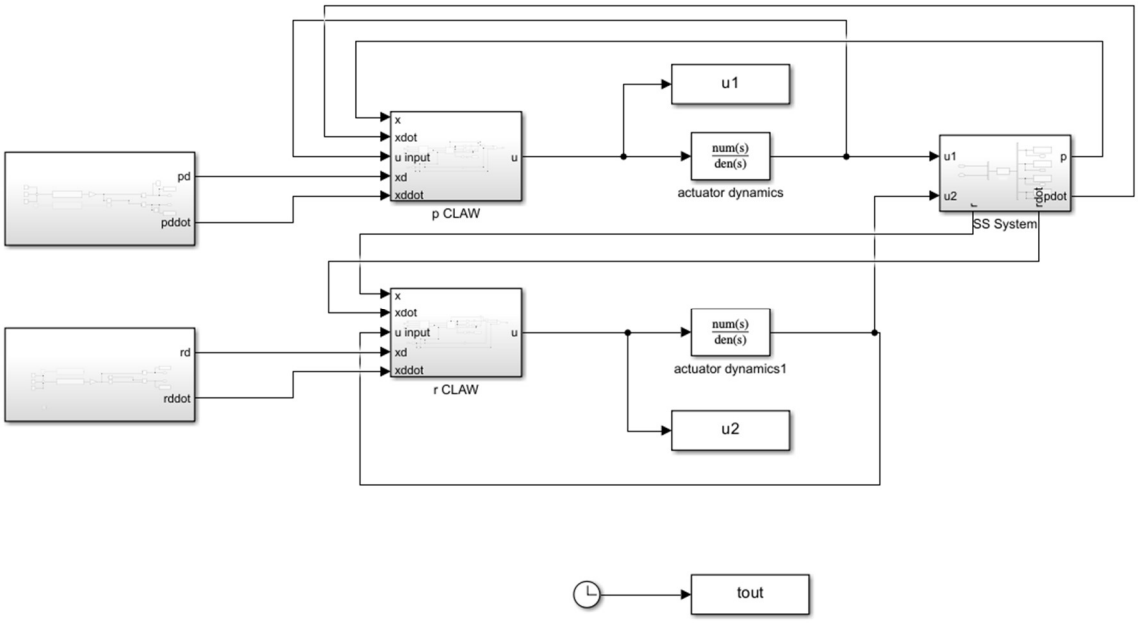

Figure A10: State Space System

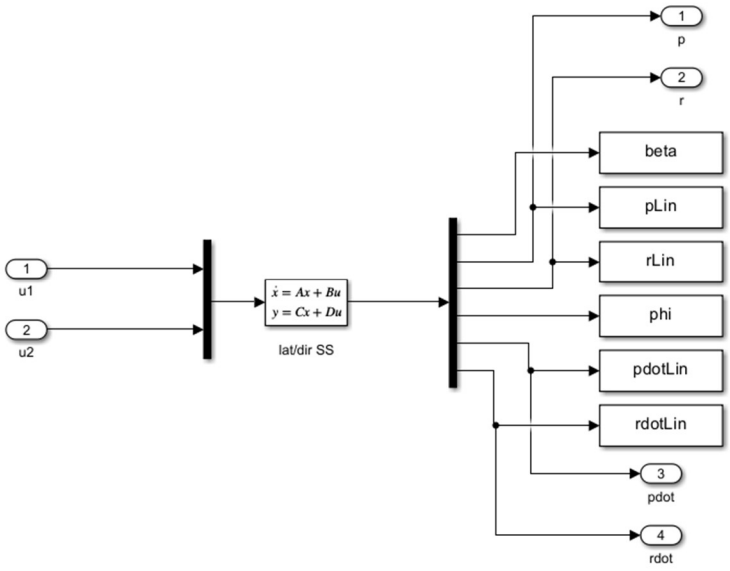

Figure A11: State Space System Model

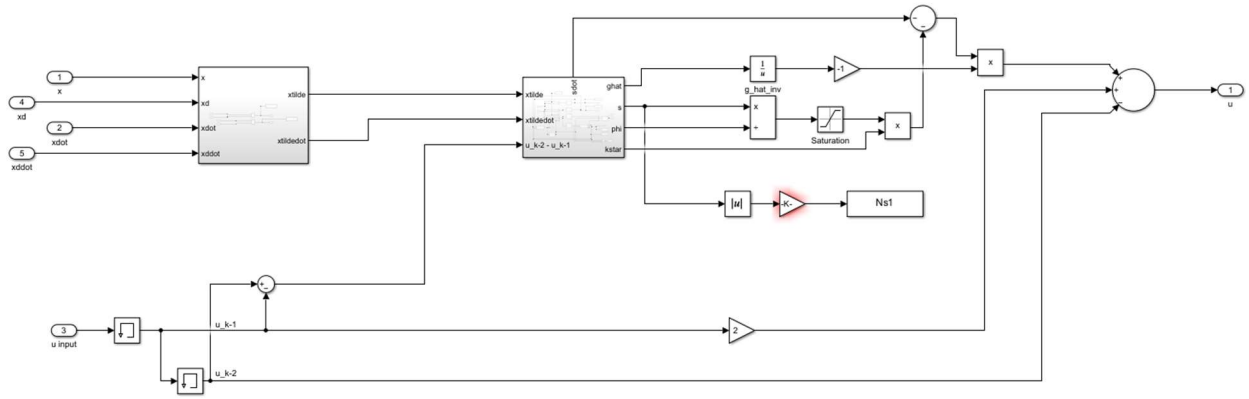

Figure A12: State Space Control Law

6.5 Section 3.3.5 MATLAB Code and Simulink Diagrams

6.5.1 MATLAB Code

```
%two_in_two_out_MFSMC_coupledB_m.m
%model-free control of 2 coupled 2nd-order systems with coupled B matrix
%open-loop system: x1dotdot+a1(t)*x1dot^2*cos(3*x1)*x2 = 6*u1(t)+4*u2(t)
%                  x2dotdot+a2(t)*x2dot^2*x2*x1dot     = 2*u1(t)+3*u2(t)
%                  1<=a1(t)<=2
%                  4<=a2(t)<=6

clear all,clc,format short e
close all

% define options
const_parm_sw  = 0;  %=0, vary the parameters a1(t) and a2(t) in system model; =1,
use constant parameters
signum_sw      = 0;  %=1, use signum function; =0, use saturation smoothing boundary
layer
constant_b     = 1;  %=0, use constant B matrix in system model; =1, use varying B
matrix in system model
vary_bl_sw     = 1;  %=1, use a varying boundary layer; =0, use constant boundary
later
if( vary_bl_sw>0.5 )
    signum_sw=0;
end

%define actuator time constant (sec)
tau_act=0.05;        %(sec)

%define values for constant system model parameters
a1_c=1.5;a2_c=5;

%state ICs, want to track x1d=sin((pi/2)*t), x2d=cos((pi/2)*t)
x10    = 0.0;
x1dot0 = pi/2;
x20    = 1.0;
```

```
x2dot0 = 0.0;
x1TildeInt0 = 0;
x2TildeInt0 = 0;

%define SMC gains
lambda1=20;ita1=4;phi1_constant_bl=0.5;
lambda2=20;ita2=4;phi2_constant_bl=0.5;

%define gains for nonunitary B matrix using matrix approach for CLAW
sigma_u=0.2;
Bupp=[7 5;3 4];
Blow=[5 3;1 2];
invBlow=[1/Blow(1,1) 1/Blow(1,2);1/Blow(2,1) 1/Blow(2,2)];
Bhat=(Bupp.*Blow).^0.5;
inv_Bhat=pinv(Bhat);
beta=(Bupp.*invBlow).^0.5;
beta_inv=[1/beta(1,1) 1/beta(1,2);1/beta(2,1) 1/beta(2,2)];
betad=beta;
betad_inv=beta_inv;
betad_inv_2=betad_inv.*betad_inv;
abs_beta_minus_1=abs(beta-[1 1;1 1]);
abs_betad_minus_1=abs_beta_minus_1;
lambda_mat=[lambda1 0;0 lambda2];
betad_inv_lambda_mat=betad_inv.*lambda_mat;
betad_inv_2_lambda_mat=betad_inv_2.*lambda_mat;
lambda_inv_betad=inv(lambda_mat).*betad;
abs_beta_times_1sigma_minus_1=abs(beta*(1+sigma_u)-[1 1;1 1]);

%simulate system
tf=10;
dt=0.001; %simulation step size
sim('two_in_two_out_MFSMC_coupledB_s')
```

6.5.2 Simulink Diagrams

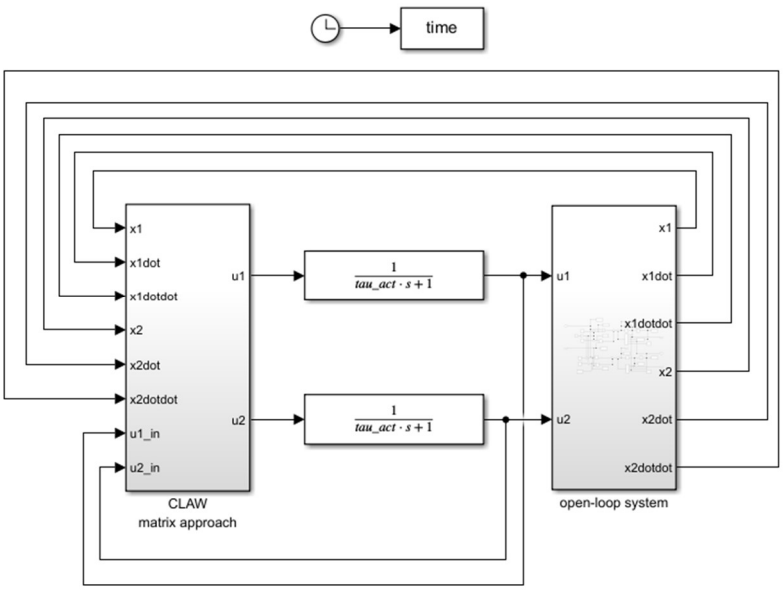

Figure A13: MIMO Gain Matrix System

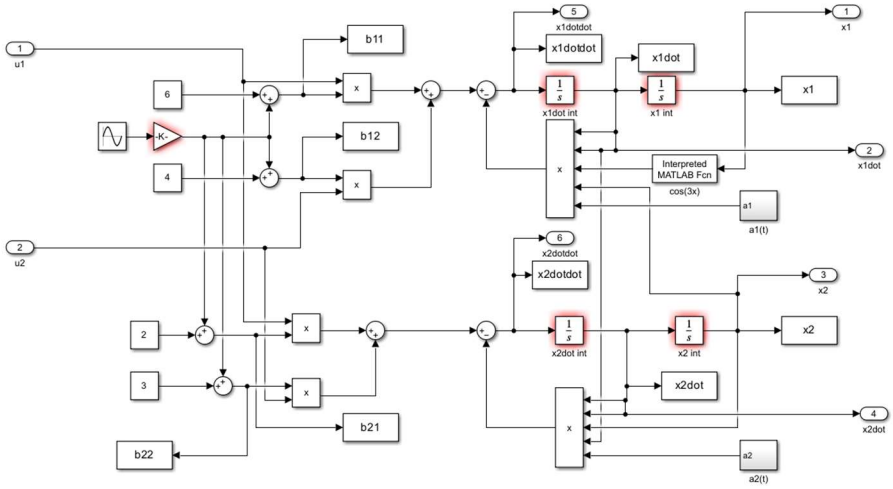

Figure A14: MIMO Gain Matrix System Model

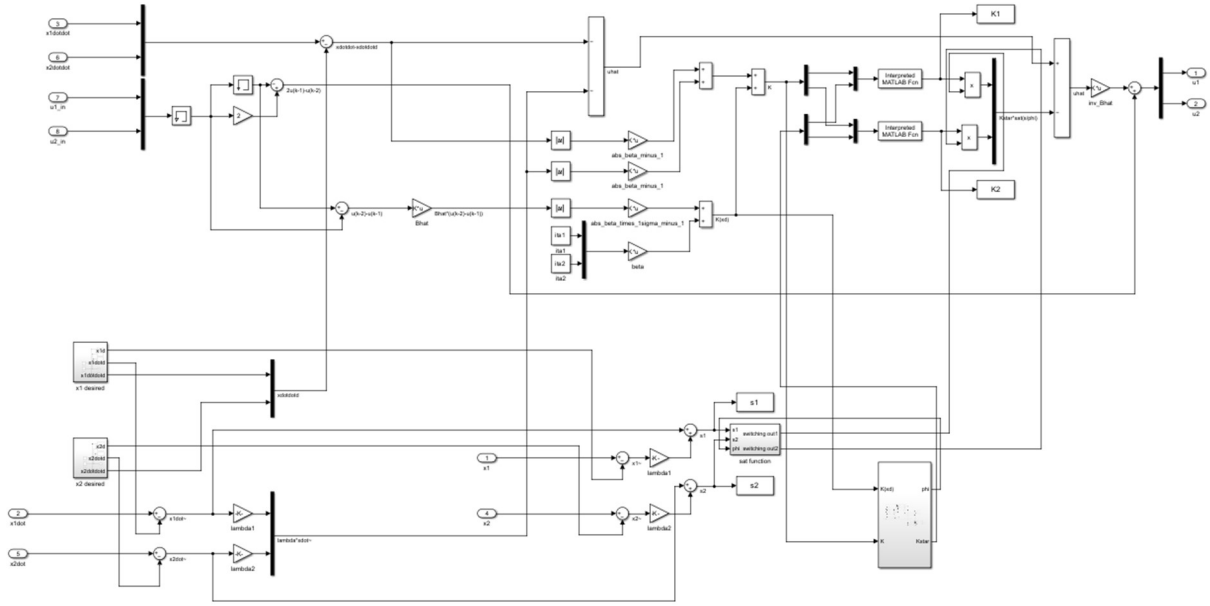

Figure A15: MIMO Gain Matrix Control Law

6.6 Section 3.3.6 MATLAB Code and Simulink Diagrams

6.6.1 MATLAB Code

```
%model-free control of lat dir SS model

clear all,clc,format short e
close all
```

```
%Define open-loop model
Alat=[ -0.2316  0.0633 -0.9956  0.051
      -29.4924 -3.0169  0.0201  0.0
        6.2346 -0.0274 -0.4169  0.0
        0.0     1.0     0.0631  0.0   ];
Blat=-[  0.0052  0.031
      -36.4909  8.1090
       -0.4916 -2.8274
        0.0     0.0    ];
Clat=eye(4);Clat(5:6,:)=Alat(2:3,:);
Dlat=zeros(4,2);Dlat(5:6,:)=Blat(2:3,:);
%C = Clat([2,3,5,6],:);
%D = Dlat([2,3,5,6],:);
C = Clat;
D = Dlat;

%define desired roll and yaw
roll = 20; %assume to be stick displacement in inches or force in lb
yaw = 0;
K_roll = 1;
K_yaw = 1;

%define Level 1 aircraft TFs
zeta_d = 0.8; %min of 0.19
w_nd = 2; %rad/s, min of 0.4 rad/sec
tau = 0.3; %s, max of 1.4s

% define options
const_parm_sw  = 0;  %=0, vary the parameters a1(t) and a2(t) in system model; =1,
use constant parameters
signum_sw      = 0;  %=1, use signum function; =0, use saturation smoothing boundary
layer
constant_b     = 1;  %=0, use constant B matrix in system model; =1, use varying B
matrix in system model
vary_bl_sw     = 1;  %=1, use a varying boundary layer; =0, use constant boundary
later
if( vary_bl_sw>0.5 )
    signum_sw=0;
end
B_hat_sw=0;   %=0, use constant B_hat; =1, use B_hat from realtime estimator

%define actuator time constant (sec)
tau_act=0.05;        %(sec)

%define values for constant system model parameters
a1_c=1.5;a2_c=5;

%state ICs, want to track x1d=sin((pi/2)*t), x2d=cos((pi/2)*t)
x10    = 0.0;
x1dot0 = pi/2;
x20    = 1.0;
x2dot0 = 0.0;
x1TildeInt0 = 0;
x2TildeInt0 = 0;
```

```
%set simulation time
tf=10;
dt=0.001; %simulation step size
t = transpose(0:dt:tf);

%define SMC gains
lambda1=15;ita1=4;phi1_constant_bl=0.5;
lambda2=15;ita2=4;phi2_constant_bl=0.5;
N_daMat = [t,ita1*ones(length(t),1)]; %ita matrix
N_drMat = [t,ita2*ones(length(t),1)]; %ita matrix
sigmaU_da = 0.2;    %normally 0.2
sigmaU_dr = 0.2;    %normally 0.2
sigmaU_daMat = [t,sigmaU_da*ones(length(t),1)]; %sigma matrix
sigmaU_drMat = [t,sigmaU_dr*ones(length(t),1)]; %sigma matrix

%define gains for nonunitary B matrix using matrix approach for CLAW
sigma_u=0.2;
b11 = Blat(2,1);
b12 = Blat(2,2);
b21 = Blat(3,1);
b22 = Blat(3,2);
B = [b11,b12;b21,b22];
b_upp_percent=0.2;         %percentage of upper value of "b" used to estimate "beta"
in realtime, normally 0.1
b_low_percent=0.2;         %percentage of lower value of "b" used to estimate "beta"
in realtime, normally 0.1
Bupp=B+b_upp_percent*B;   %upper value of "b" in system model
Blow=B-b_low_percent*B;   %lower value of "b" in system model
invBlow=[1/Blow(1,1) 1/Blow(1,2);1/Blow(2,1) 1/Blow(2,2)];
Bhat=(Bupp.*Blow).^0.5;
Bhat0 = Bhat;
inv_Bhat=pinv(Bhat);
beta=(Bupp.*invBlow).^0.5;
beta_inv=[1/beta(1,1) 1/beta(1,2);1/beta(2,1) 1/beta(2,2)];
betad=beta;
betad_inv=beta_inv;
betad_inv_2=betad_inv.*betad_inv;
abs_beta_minus_1=abs(beta-[1 1;1 1]);
abs_betad_minus_1=abs_beta_minus_1;
lambda_mat=[lambda1 0;0 lambda2];
betad_inv_lambda_mat=betad_inv.*lambda_mat;
betad_inv_2_lambda_mat=betad_inv_2.*lambda_mat;
lambda_inv_betad=inv(lambda_mat).*betad;
abs_beta_times_1sigma_minus_1=abs(beta*(1+sigma_u)-[1 1;1 1]);

%gains for estimator for "B_hat"
P10=50;
P20=50;
P0 = [P10,0;0,P20];
lambda0_b=20;            %higher value cause faster convergence and boundary layer,
phi will be small along with control effort but more suspectible to noise effects
lambda_sw=1;            %=1, use varying lambda technique; =0, use constant lambda so
lambda=lambda0
k0=200;                 %higher value cause faster convergence and boundary layer,
phi will be small along with control effort (100 is nice)
```

```
close_phi_gap_sw=0;     %=0, don't close boundary layer gap; =1, close gap which
reduces "B" magnitude ('e1*(|s|>=phi)' block in 'K and phi' subsystem

%simulate system
sim('Lat_Dir_MFSMC_coupledB_s')
```

6.6.2 Simulink Diagrams

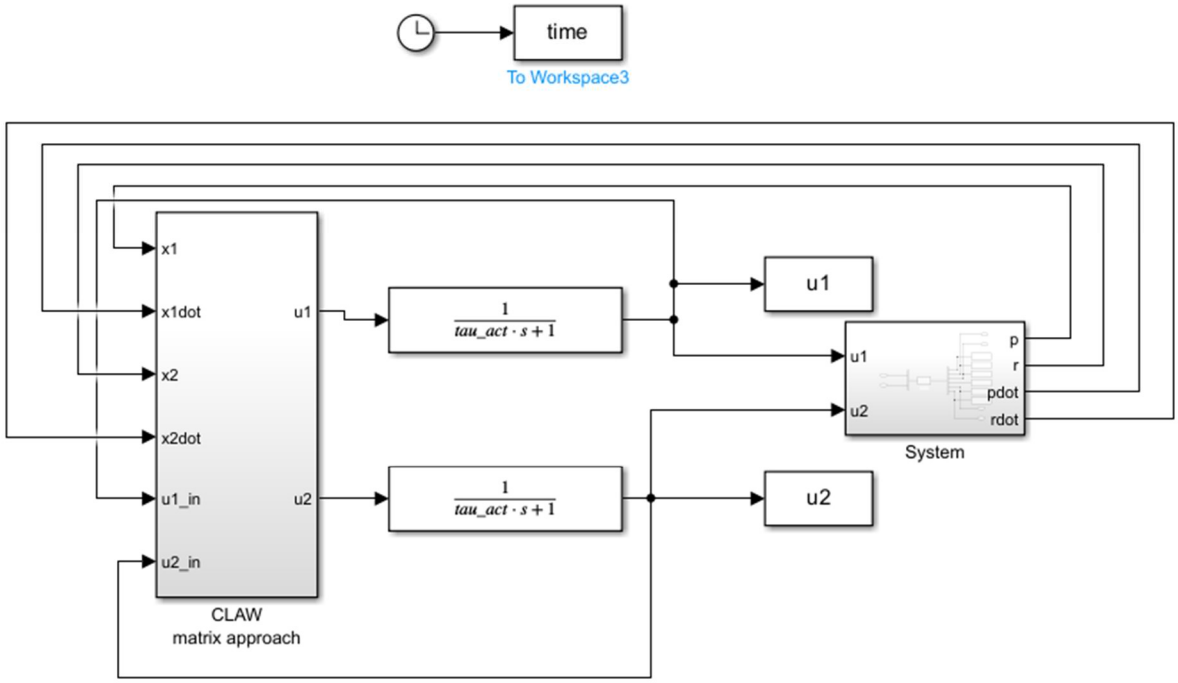

Figure A16: State Space Gain Matrix System

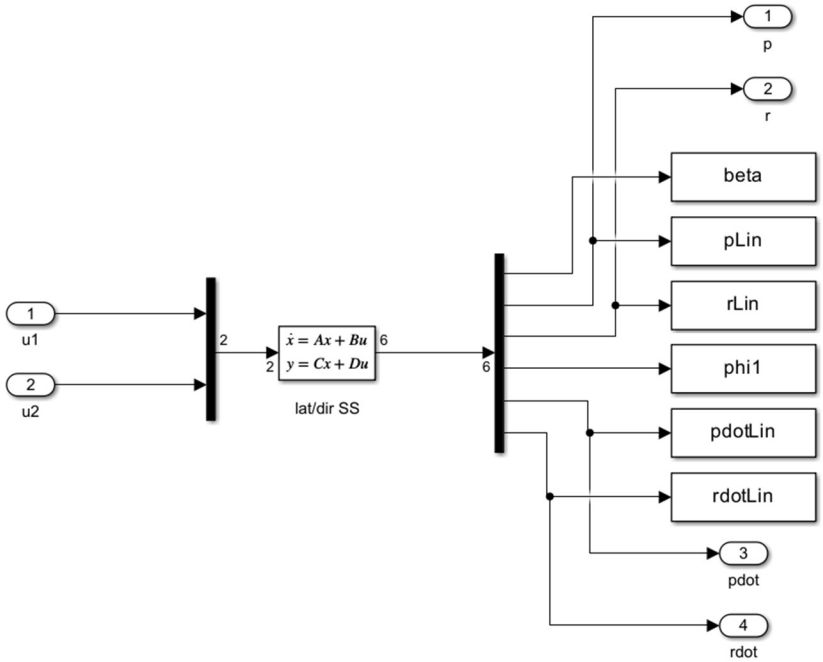

Figure A17: State Space Gain Matrix System Model

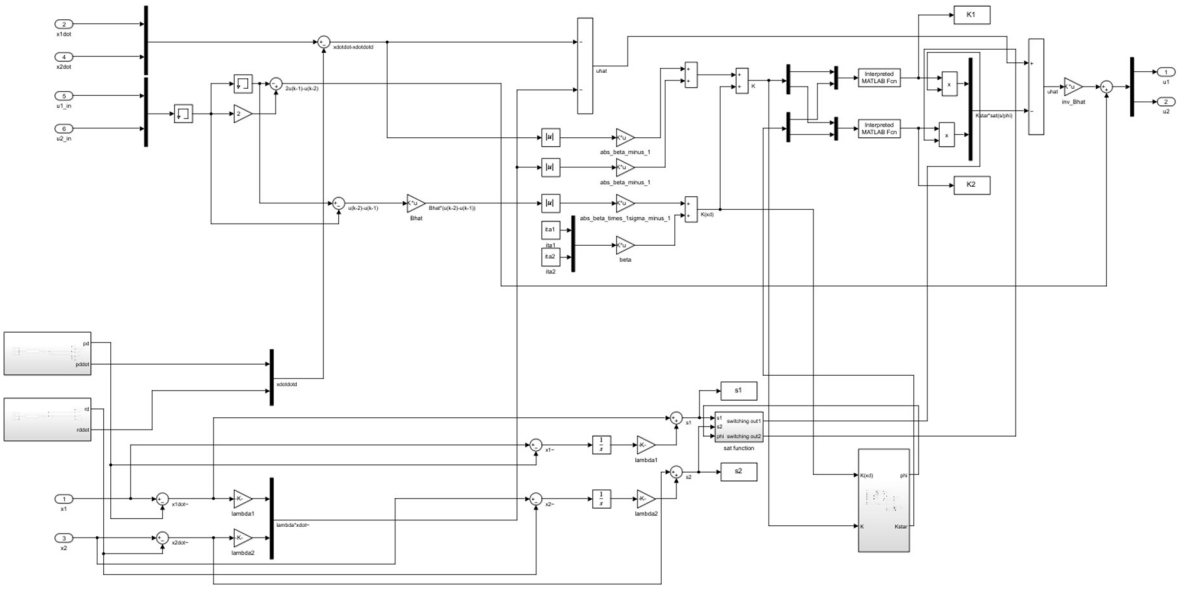

Figure A18: State Space Gain Matrix Control Law