7-23-2024

Improvements on Model-Free Sliding Mode Control's Implementation and Estimation Techniques

Joshua Coleman
jcc2126@rit.edu

ROCHESTER INSTITUTE OF TECHNOLOGY

Improvements on Model-Free Sliding Mode Control's Implementation and Estimation Techniques

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

JOSHUA COLEMAN

Advisor: DR. AGAMEMNON CRASSIDIS

Thesis Committee

Dr. Jason Kolodziej

Dr. Kathleen Lamkin-Kennard

Dr. Sarilyn Ivancic

7/23/24

DEPARTMENT OF MECHANICAL ENGINEERING

KATE GLEASON COLLEGE OF ENGINEERING

Abstract

Systems continue to grow in complexity to meet the demands of today's society. To control severe nonlinearities and protect against errors in the system model, control methods with wider stability margins are necessary. However, most robust control methods require an increased energy cost to guarantee stability with wider margins. Furthermore, most current control methods are derived using system models, which are difficult to develop for complex systems. Model-Free Sliding Mode Control is promising in overcoming the aforementioned difficulties; the control output is only determined by the system order, previous control values, and state measurements. The control scheme's characteristics have been mathematically derived but may have important unexplored practical implications. Originally, Model-Free Controllers were only partially model-free due to assumptions on the control influence matrix. More recent controllers relaxed the assumption using a real-time estimator. In the present work, a model-free control implementation is developed exploiting the characteristic. The new implementation allows for quicker and easier model-free controller development. Additionally, model-free control's current estimator is validated. A new estimator, which approximates the boundaries of the influence matrix (rather than the matrix itself) is proposed and tested in a sliding mode control setting. The new estimator allowed for stability in the reaching phase and comparable tracking performance in the sliding phase as a normal sliding mode controller. The tracking performance was a result of a lower control input, though this is not guaranteed in all circumstances. If the new estimator is added to the new model-free control implementation, the controller may perform better and more efficiently. In the first part of this work, the model-free controller and real-time estimator are derived. Next, a simulation study was performed to prove the feasibility of the new approach. Finally, recommendations for next steps of the research are outlined.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOMENCLATURE

B	control influence matrix
$\hat{B}$	estimate of B
$\frac{d}{dt}$	time derivative operator (equivalent to an over dot)
$diag(\cdot)$	diagonal matrix function: creates a matrix whose diagonal is $\cdot$
$\vec{f}$	the controller-independent part of the system model
$\hat{f}$	the best estimate of $\vec{f}$
I	identity matrix
L	used in the practical implementation of SMC to get $\dot{\vec{s}}$, see Section 3.3
$max(\cdot)$	maximum function: returns the maximum of $\cdot$
$min(\cdot)$	minimum function: returns the minimum of $\cdot$
P	used in the practical implementation of SMC to get $\vec{s}$, see Section 3.3
$\vec{s}$	sliding mode function
$\dot{\vec{s}}$	time derivative of $\vec{s}$
$sat(\cdot)$	saturation function: equal to $sgn(\cdot)$ if $\lvert\cdot\rvert > 1$, equal to $\cdot$ otherwise
$sgn(\cdot)$	signum function: equal to 0 if $\cdot$ is 0, equal to $\frac{\cdot}{\lvert\cdot\rvert}$ otherwise
t	time variable
$\vec{u}$	control input vector
$vec(\cdot)$	vectorization function: turns the matrix $\cdot$ into a vector by stacking its columns
$\vec{x}$	(actual) system states
$\dot{\vec{x}}$	time derivative of $\vec{x}$
$\tilde{x}$	difference between the actual and desired system states
$\dot{\tilde{x}}$	the time derivative of $\tilde{x}$

$\overline{0}$	a matrix of all zeros
$\vec{\alpha}$	estimator SMC state coefficient, see Section 4.2
$\vec{\eta}$	SMC discontinuous switching gain in the ideal setting
δ	coefficient used in coupled SMC
$\vec{\varepsilon}$	control influence error
$\hat{\varepsilon}$	estimate of $\vec{\varepsilon}$
$\vec{\kappa}$	SMC discontinuous switching gain
Λ	positive definite matrix used in the definition of $\vec{s}$
$\vec{\nu}$	the control output when B is unitary
$\vec{\Phi}$	SMC boundary layer thickness
$\dot{\vec{\Phi}}$	time derivative of $\vec{\Phi}$
$\vec{\chi}$	vector created by stacking all $\vec{x}^{(i)}$, see Section 3.3
$\dot{\vec{\chi}}$	time derivative of $\vec{\chi}$
$\tilde{\chi}$	difference between the real and desired values of $\vec{\chi}$
...	matrix element placeholder meaning “repeat while increasing indices to the right”
⋮	matrix element placeholder meaning “repeat while increasing indices downward”
$\otimes$	Kronecker product, see Section A.1
$\circ$	Hadamard product, see Section A.1
a_1	coefficient for state 1 or the first value in vector a
a_2	coefficient for state 2 or the second value in vector a
$B_{k\|k}$	estimated value of B on iteration k after final adjustments
$B_{k\|k-1}$	estimated value of B on iteration k before final adjustments
B_{max}	maximum value of B
B_{min}	minimum value of B

C_k	coefficient for the kth term of a polynomial
$\vec{x}_d$	desired state values
$\vec{x}^{(i)}$	represents the set of all time derivatives of $\vec{x}$ from i = 0 through $i = n - 1$
$\vec{x}^{(n-1)}$	the $(n-1)^{th}$ time derivative of $\vec{x}$
$\vec{x}^{(n)}$	the nth time derivative of $\vec{x}$
$\tilde{x}^{(n-1)}$	the $(n-1)^{th}$ time derivative of $\tilde{x}$
$\tilde{x}^{(n)}$	the n^{th} time derivative of $\tilde{x}$
$\vec{\kappa}^*$	SMC with variable boundary layer discontinuous switching gain
σ_u	upper boundary constant in MFSMC
SMC	sliding mode control
MFSMC	model-free sliding mode control
MIMO	multi-input multi-output

1.0 INTRODUCTION

Since the 19th century, control systems theory has grown to become an influential part of modern life. Traditional control methods were (and are) suitable for simple systems and well-behaved nonlinear systems. However, severe nonlinearities can make applying traditional control laws difficult. The problem is exacerbated by inaccuracies in the system model due to error in parameter measurements and the simplifying assumptions in the system model.

As a result of the aforementioned issues, control methods with wider stability margins have become popular. These robust control methods are provably stable for a general class of systems. One such example is Sliding Mode Control (SMC). To develop a sliding mode controller, a sliding mode with desirable characteristics for the system is chosen. Then, a controller is developed to cause squared magnitude of the sliding mode to be a Lyapunov function—a positive definite function (i.e., equal to 0 at the origin, greater than 0 otherwise) with a negative definite time derivative (i.e., equal to 0 at the origin, less than 0 otherwise) around a singular point in the operating region. If the measurement error does not exceed the assumed values, the closed-loop system will be stable.

While SMC can be an ideal robust control choice, assumptions on the bounds of errors and unknown signals are integral in the derivation of these types of controllers. In many cases it may be desirable to assume wide error bounds. Doing so would make the system more robust against malfunction and would guarantee stability if the same controller is applied to multiple systems. In the case of SMC, increased robustness results in an increase in the energy requirements for the controller. In the most extreme case—where the bounds are unknown and thus assumed to be infinite—infinite energy from the controller would be required, severely limiting the controller's practicality.

A current solution to the deficiencies of traditional SMC is Model-Free Sliding Mode Control (MFSMC), which was developed in [1]. Rather than finding a way to widen the bounds of the error, model-free controllers avoid the system model entirely. The SMC-based controller is then developed using real-time state measurements and assumptions on the growth of the controller output. Since the first inception, the control system has been expanded to both square and non-square Multiple Input Multiple Output (MIMO) systems [2,3]. These were not truly "model-free" however, since an upper and lower bound was assumed for the input influence matrix, locking the matrix to a certain set of values.

More recently, [4] proposed the use of an estimator to relax the assumptions on the control influence. Originally, the estimator estimated the minimum switching gain. Due to the approach's lack of energy efficiency, [5] presented the idea of using the estimator to approximate the control influence. The estimate is used to calculate the bounds of the matrix, and the bounds

are then used in the controller. So far, the estimator's convergence has been tested with some success.

In this work, the previous estimation technique is validated. Then, a method for directly estimating the influence bounds is developed. SMC with and without the method is applied to a nonlinear, square systems, and the performance results are compared. This work also studies MFSMC's implementation. Previous work has recreated MFSMC systems for each test plant. Thanks to MFSMC's dependence on only the system order and the number of states, the present work removes other dependencies from MFSMC's implementation. As a result, no other square MFSMC system is required to be created, even if the order or number of states changes. Simulation results using the old and new implementations are compared to validate the new approach.

1.1 Focus and Proposed Purpose

This work's original purpose was to accomplish the following goals

1. Derive a MFSMC scheme for MIMO, square, coupled systems (See Section 3.0).
2. Create a generalized implementation of the scheme (See Section 3.3).
3. Develop an improved estimator for MFSMC (See Section 4.0).
4. Simulate the controller and estimator with various nonlinear systems (See Section 5.0).
5. (Time permitting) test the controller on original hardware.

The focus of this work shifted away from Goal 1 and more towards Goals 2-4. A discussion of the accomplished goals – including motivation for the shift in focus – is given in Section 6.1.

2.0 LITERATURE REVIEW

2.1 Robust Control with Wide Error Margins

Sliding mode controllers are provably stable for a given range in the assumed parameters. Many facets of SMC are being explored, including stability under arbitrary error ranges and forms. For example, Zhou and Fisher [6] found conditions in which all nonlinear systems are stable. The controller was stable under a transformation of the Lyapunov function created from the sliding surfaces, provided the magnitude of each control input was greater than each component of one of the transformed variables. The theory illustrates the issues with widening the stability margins of SMC: the magnitude of the control input is dependent on the maximum magnitude of the error which is a gap in the research.

Work alleviating the issue has been performed. Two examples are given in [7] and [8] (though their objectives were to design controllers with good performance in the reaching phase only). The Linear-Quadratic-Regulator-based SMC scheme in [7] optimally controls linear systems with nonlinear uncertainties. The method could be used to create a controller with a lower output magnitude than others while achieving similar—if not the same—performance.

Cost functionals are not the only way to reduce the controller magnitude without performance losses. In [8], a controller is created exhibiting closed-loop stability so long as one of the controller parameters is larger than a linear combination of the system uncertainties. Since it is difficult to estimate the maximum uncertainty off-line, the authors proposed a novel on-line technique to calculate the controller parameter. The controller, which is proportional to the parameter, will, therefore, be optimized for each situation.

Both controllers excel at what they were designed for but are not universal for two reasons. First, they were designed for specific systems (either second order-nonlinear [8] or linear with nonlinear unknowns [7]). The issue could be avoided if the controllers could accommodate arbitrary amounts of error (as discussed in Section 2.2). However, as with the controller in [6], both are dependent on the uncertainty magnitude—even if they are optimal for the given uncertainty. The preferred method to avoid the issue of unbounded uncertainty would be to avoid the modeling error sources.

2.2 Model-Free Sliding Mode Control: Motivation and Previous Work

Recently, Mizov [1] proposed a robust SMC scheme called Model-Free Sliding Mode Control (MFSMC). Unlike the previous methods, the new scheme did not assume a model form, only a unitary control influence matrix. By not assuming the model form, the control input could be found using previous inputs and state measurements. Therefore, the controller was independent of any modeling errors. Then, as a result of the work of Reis [2] and El Tin [3], the control system was expanded to MIMO systems with non-unitary control input influence

matrices. Several works have demonstrated MFSMC's robust performance in situations where the control matrix's boundaries are known [9,10].

To broaden the applicability of MFSMC, work has been done to relax the boundary assumptions on the influence matrix. Islam [4] proposed the use of a real-time estimator which was then adapted in [5] to estimate the input influence matrix. If the estimate error was below an assumed boundary, the system would be stable. Therefore, the estimator must perform well to ensure stability with the unknown input influence values.

2.3 Time-Varying Parameter Estimation

The estimator proposed by [4] and used in [5] is a least-squares real-time estimator with a bounded gain forgetting technique. While least-squares was originally devised for systems with constant parameters, the forgetting factor allows it to be used for time varying systems [11]. Where the estimator in [5] differs from the usual technique is it does not use an equation explicitly involving the estimated parameters. Rather, the estimator's stable point is where the error in the sliding condition is zero (see Section 3.1.2).

The method's indirect nature is a departure from most other estimation techniques (including other time-varying estimation techniques [12–17]). Furthermore, it would be difficult to use a technique using an explicit equation due to the nature of MFSMC. Because MFSMC comes from a unity gain equation (see Section 3.0), the influence matrix would be eliminated during most estimator creation processes. Thus, a method not involving an explicit equation must be used and is proposed in this work.

3.0 MODEL-FREE SLIDING MODE CONTROL

3.1 Background: Sliding Mode Control

Sliding Mode Control (SMC) is a popular robust control method. It allows for control of systems with unknown parameters falling in a known range. Since MFSMC uses SMC techniques, a discussion of SMC is presented here.

3.1.1 Lyapunov-Based Control

SMC is one of several Lyapunov-Based Control methods. Lyapunov-Based Controllers mathematically ensure the stability of a system by showing it satisfies Lyapunov's stability criterion:

> Given any system $\vec{x}^{(n)} = \vec{f}\left(\vec{x}^{(i)}\right)$, the system is asymptotically stable in a region of space if there is a positive definite function $V\left(\vec{x}^{(i)}\right)$ whose derivative $\dot{V}$ is negative definite for all $\vec{x}$ in the region. [18]

The argument $\vec{x}^{(i)}$ is the set of all time derivatives of $\vec{x}$ between 0 and n. First, a function V is chosen. Then, a control input is chosen so V satisfies the stability criterion. If the criterion is satisfied—even in the presence of uncertainties—the system is still stable.

3.1.2 SMC Stability Criteria

Most nonlinear systems may be written in the form:

$$\vec{x}^{(n)} = \vec{f}\left(\vec{x}^{(i)}, t\right) + B\left(\vec{x}^{(i)}, t\right)\vec{u} \tag{1}$$

The controller value is $\vec{u}$. Matrices $\vec{f}$ and B are unknown but bounded by $\left[\vec{f}_{min}, \vec{f}_{max}\right]$ and $[B_{min}, B_{max}]$, respectively. All $\vec{x}^{(i)}$ are known. Suppose $\vec{u} = \hat{B}^{-1}\vec{v}$. If $\vec{v}$ caused some predetermined function to satisfy Lyapunov's stability theorem, the system would be stable.

In SMC, the candidate Lyapunov function is defined using a "sliding mode" $\vec{s}$:

$$V = 0.5\vec{s}^T\vec{s} \quad \rightarrow \quad \dot{V} = \vec{s}^T\dot{\vec{s}} \tag{2}$$

$$\vec{s} = \left[\frac{d}{dt} + \Lambda\right]^{n-1}\tilde{x} \quad \rightarrow \quad \dot{\vec{s}} = \left[\frac{d}{dt} + \Lambda\right]^{n-1}\dot{\tilde{x}} \tag{3}$$

When $\vec{s} = 0$, $\tilde{x}$ will go to 0 (along with the error in the system), so $\vec{v}$ should be chosen to make V a Lyapunov function. Since $V > 0$, V is a Lyapunov function if $\dot{V} < 0$. Let:

$$\vec{v} = -\hat{f} - \left(\dot{\vec{s}} - \tilde{x}^{(n)}\right) + \vec{x}_d^{(n)} - \vec{\kappa} \circ sgn(\vec{s}). \tag{4}$$

It is important to note if $\vec{\kappa} = \vec{\eta} > 0$ and there is no uncertainty, $\dot{V} = -\vec{\eta}^T|\vec{s}| < 0$. When uncertainty is present, $\vec{\kappa}$ may be chosen so $\dot{V} \leq -\vec{\eta}^T|\vec{s}|$. The inequality is known as the sliding condition. After substituting in $\dot{\vec{s}}$ and rearranging, the inequality becomes:

$$\vec{s}^T B\hat{B}^{-1}(\vec{\kappa} \circ sgn(\vec{s})) \geq \vec{s}^T \left[\left(I - B\hat{B}^{-1}\right)\left(\left(\dot{\vec{s}} - \tilde{\vec{x}}^{(n)}\right) - \vec{x}_d^{(n)}\right) + \left(\vec{f} - B\hat{B}^{-1}\hat{f}\right)\right] + \vec{\eta}^T|\vec{s}| \quad (5)$$

While upholding the inequality proves system stability, it leaves to many degrees of freedom. Satisfying the stronger inequality:

$$\vec{s} \circ B\hat{B}^{-1}(\vec{\kappa} \circ sgn(\vec{s})) \geq \vec{s} \circ \left[\left(I - B\hat{B}^{-1}\right)\left(\left(\dot{\vec{s}} - \tilde{\vec{x}}^{(n)}\right) - \vec{x}_d^{(n)}\right) + \left(\vec{f} - B\hat{B}^{-1}\hat{f}\right)\right] + \vec{\eta} \circ |\vec{s}| \quad (6)$$

proves stability while removing those degrees of freedom. There are several ways to solve the stronger inequality. Two methods (as developed in [19]) are discussed in 3.1.3 and 3.1.4.

3.1.3 Decoupled SMC

When the control inputs are decoupled (i.e. there is one input or the influence matrix is diagonal), the following relation is true:

$$sgn(\vec{s}) \circ B\hat{B}^{-1}(\vec{\kappa} \circ sgn(\vec{s})) = B\hat{B}^{-1}\vec{\kappa} \quad (7)$$

As a result, the inequality reduces to:

$$\vec{\kappa} \geq sgn(\vec{s}) \circ \left[\left(\hat{B}B^{-1} - I\right)\left(\left(\dot{\vec{s}} - \tilde{\vec{x}}^{(n)}\right) - \vec{x}_d^{(n)}\right) + \left(\hat{B}B^{-1}\vec{f} - \hat{f}\right)\right] + \hat{B}B^{-1}\vec{\eta} \quad (8)$$

If we define:

$$\hat{B} = \sqrt{B_{max}B_{min}} \quad (9)$$

$$\beta = \sqrt{B_{max}B_{min}^{-1}} \quad (10)$$

the sliding condition is satisfied when:

$$\vec{\kappa} = |\beta - I|\left(\left|\vec{x}_d^{(n)}\right| + \left|\dot{\vec{s}} - \tilde{\vec{x}}^{(n)}\right| + \left|\hat{f}\right|\right) + \beta\left|\vec{f}_{max} - \hat{f}\right| + \beta\vec{\eta}. \quad (11)$$

3.1.4 Coupled SMC

For the coupled case, Eq. (7) is not true. Instead, if

$$B\hat{B}^{-1} = \delta + I \quad (12)$$

the sliding condition becomes:

$$\vec{s} \circ (\delta + I)(\vec{\kappa} \circ sgn(\vec{s})) \geq \vec{s} \circ \left[\delta\left(\left(\dot{\vec{s}} - \tilde{\vec{x}}^{(n)}\right) - \vec{x}_d^{(n)}\right) + \left(\vec{f} - (\delta + I)\hat{f}\right)\right] + \vec{\eta} \circ |\vec{s}| \quad (13)$$

If $D = max(|\delta|)$, $\overline{\overline{D}}$ is the matrix with only the diagonal elements of D, and $D_{\oslash} = D - \overline{\overline{D}}$:

$$\vec{\kappa} = \left(I + \overline{\overline{D}} - D_{\oslash}\right)^{-1}\left[D\left(\left|\dot{\vec{s}} - \tilde{\vec{x}}^{(n)}\right| + \left|\vec{x}_d^{(n)}\right| + \left|\hat{f}\right|\right) + \left|\vec{f}_{max} - \hat{f}\right| + \vec{\eta}\right] \tag{14}$$

will cause the sliding condition to be satisfied.

3.1.5 Dithering Reduction

In general, basic SMC is not implementable due to the controller's discontinuity. To solve the issue, [19] proposed a boundary layer in which the controller would become linear (regarding the sliding function). The linearity allows the controller to be continuous. Since this work is not concerned with the boundary layer, the resulting equations are given here. A more thorough discussion of the boundary layer is given in the original text.

The boundary layer version presented in this work has a variable thickness $\vec{\Phi}$. To cause the controller to transition between a normal SMC scheme and a linear controller, the $\vec{\kappa} \circ sgn(\vec{s})$ is replaced with the $\vec{\kappa}^* \circ sat\left(\frac{\vec{s}}{\vec{\Phi}}\right)$. The values of $\vec{\Phi}$ and $\vec{\kappa}^*$ are given as:

$$\dot{\vec{\Phi}} = \begin{cases} -\Lambda\vec{\Phi} + \beta_d\vec{\kappa}_d & \beta_d\vec{\kappa}_d > \Lambda\vec{\Phi} \\ -{\beta_d}^{-2}\Lambda\vec{\Phi} + {\beta_d}^{-1}\vec{\kappa}_d & else \end{cases} \tag{15}$$

$$\vec{\kappa}^* = \begin{cases} \vec{\kappa} - \beta^{-1}\dot{\vec{\Phi}} & \beta_d\vec{\kappa}_d > \Lambda\vec{\Phi} \\ \vec{\kappa} - \beta\dot{\vec{\Phi}} & else \end{cases} \tag{16}$$

where $\vec{\kappa}_d$ and β_d are what the values of $\vec{\kappa}$ and β would be if $\vec{s}$ was 0. As discussed in [19], the method removes the unimplementable discontinuity while keeping the error within a certain boundary.

3.2 MFSMC Derivation

MFSMC is robust due to the avoidance of a system model and its basis in SMC. As a result, it may be derived in the same manner as all Sliding Mode Controllers. MFSMC was first derived in [3]. Square MFSMC was first applied to a coupled system in [5]. The present derivation rederives the controller and adds more discussion on the controller's applicability to coupled systems.

To create a model-free controller, start with the following unitary gain equation:

$$\vec{x}^{(n)} = \vec{x}^{(n)} + B[\vec{u} - \vec{u}_{k-1}] + \vec{\varepsilon} \tag{17}$$

where $\vec{\varepsilon} = B[\vec{u}_{k-1} - \vec{u}]$ and $\vec{u}_{k-i}$ is the value of $\vec{u}$ in the previous i^{th} time step. If we define $\hat{\varepsilon} = \hat{B}[\vec{u}_{k-2} - \vec{u}_{k-1}]$ and assume $\vec{\varepsilon}$ and B are bounded by:

$$|\vec{\varepsilon}| < (1 + \sigma_u)|\hat{\varepsilon}| \tag{18}$$

$$B_{min} < B < B_{max} \tag{19}$$

it is possible to apply traditional SMC principles to the model-free approach. Decoupled SMC has been used for all versions of MFSMC thus far (including coupled MFSMC). The controller is still stable because the coupled terms may be taken as part of the uncertainty (i.e., a part of $\vec{\varepsilon}$, still bounded by $(1+\sigma_u)|\hat{\varepsilon}|$). Noting $\vec{x}^{(n)} + \vec{\varepsilon}$ is like $\vec{f}$ in the SMC derivation, an MFSMC output may be defined as:

$$\vec{u} = \hat{B}^{-1}\left[-\vec{x}^{(n)} - \hat{\varepsilon} - \left(\dot{\vec{s}} - \tilde{x}^{(n)}\right) + \vec{x}_d^{(n)} - \vec{\kappa} \circ sgn(\vec{s})\right] + \vec{u}_{k-1} \tag{20}$$

Now only $\vec{\kappa}$ needs to be found. Noting again the sliding condition is:

$$\vec{s} \circ \dot{\vec{s}} \leq \vec{\eta} \circ |\vec{s}| \tag{21}$$

we can substitute in the definition of $\dot{\vec{s}}$ and replace $\tilde{x}^{(n)}$ with Eq. (17) minus $\vec{x}_d^{(n)}$. Finally, rearrange to get the inequality:

$$\vec{s}^T B\hat{B}^{-1}(\vec{\kappa} \circ sgn(\vec{s})) \geq \vec{s}^T \left[\left(I - B\hat{B}^{-1}\right)\left(\vec{x}^{(n)} + \left(\dot{\vec{s}} - \tilde{x}^{(n)}\right) - \vec{x}_d^{(n)}\right) + \left(\vec{\varepsilon} - B\hat{B}^{-1}\hat{\varepsilon}\right)\right] + \vec{\eta} \circ |\vec{s}| \tag{22}$$

Since we are using decoupled SMC, we can take advantage of Eq. (7) and get:

$$\vec{\kappa} \geq diag(sgn(\vec{s}))\left[\left(\hat{B}B^{-1} - I\right)\left(\vec{x}^{(n)} + \left(\dot{\vec{s}} - \tilde{x}^{(n)}\right) - \vec{x}_d^{(n)}\right) + \left(\hat{B}B^{-1}\vec{\varepsilon} - \hat{\varepsilon}\right)\right] + \hat{B}B^{-1}\vec{\eta} \tag{23}$$

The final step is to maximize the right side and set it equal to $\vec{\kappa}$. The only difference between MFSMC's approach and the approach in Section 3.1.3 is the grouping of $\vec{\varepsilon}$ and $\hat{\varepsilon}$:

$$\vec{\kappa} = |\beta - I|\left(\left|\tilde{x}^{(n)}\right| + \left|\dot{\vec{s}} - \tilde{x}^{(n)}\right|\right) + |\beta(1+\sigma_u) - I||\hat{\varepsilon}| + \beta\vec{\eta} \tag{24}$$

Both grouping methods are valid maximizations. Like SMC, the switching term in the control input makes the controller unimplementable. The dithering reduction techniques discussed in Section 3.1.5 may be added for practical use.

As discussed, requiring known bounds of B prevents MFSMC from being truly model-free. However, the input influence matrix may be estimated in real-time. Influence matrix estimation is discussed in Section 4.0.

3.3 Improved Implementation

Mathematically, the creation of MFSMC is done. However, there is an important practical consideration. The purpose of MFSMC is to design a controller suitable to any problem, given the number of (nonderivative) states and the system order are known. The controller should be created once and then applied to any problem type. To allow for this versatility, SMC was derived in a general form. As previously discussed, only a correspondence between the general derivation and the terms in MFSMC is needed to derive MFSMC. However, the derivation by itself does not allow for the one-time creation of a broadly applicable controller

due to the polynomial operator in the sliding mode's definition ($\left[\frac{d}{dt}+\Lambda\right]^{n-1}$). The operator would have to be expanded by the controls engineer in every situation, limiting the controller's use cases.

To solve the issue, the following notation was developed. Again, the sliding surface $\vec{s}\ is$:

$$\vec{s} = \left[\frac{d}{dt}+\Lambda\right]^{n-1} \tilde{x} \tag{25}$$

The goal is to define an equivalent, generally implementable operation. If we define a vector $\tilde{\chi}$ and matrix P to be:

$$P = [\ldots \quad C_k\Lambda^{n-k} \quad \ldots] \; for \;\; k \in [1,n] \tag{26}$$

$$\tilde{\chi} = \begin{bmatrix} \tilde{x} \\ \vdots \\ \tilde{x}^{(n-1)} \end{bmatrix} \tag{27}$$

then $\vec{s} = P\tilde{\chi}$. Since P can be precomputed, the product may be automatically expanded by any program capable of matrix multiplication. Further, defining a similar matrix L:

$$L = \left[\overline{0} \quad \ldots \quad C_k\Lambda^{n-k} \quad \ldots\right] for \;\; k \in [1,n-1] \tag{28}$$

$\dot{\vec{s}}$ may be computed as $L\tilde{\chi} + \tilde{x}^{(n)}$. More importantly, $L\tilde{\chi}$ may be used instead of $\dot{\vec{s}} - \tilde{x}^{(n)}$. In P and L, C_k is the k^{th} coefficient of the $(n-1)^{th}$ row of pascal's triangle, which can be calculated from the formula given in [20]. A model-free controller designed using the matrices presented here can, therefore, be practically applied to any system.

Example Simulink implementations are given in Figures 1 and 2. The original implementation (Figure 1) was only suitable for 2x2 systems and has two copies of almost every block. To apply it to a system with a different number of states and inputs, these blocks would have to be duplicated again. The new implementation in Figure 2, however, can control any square system.

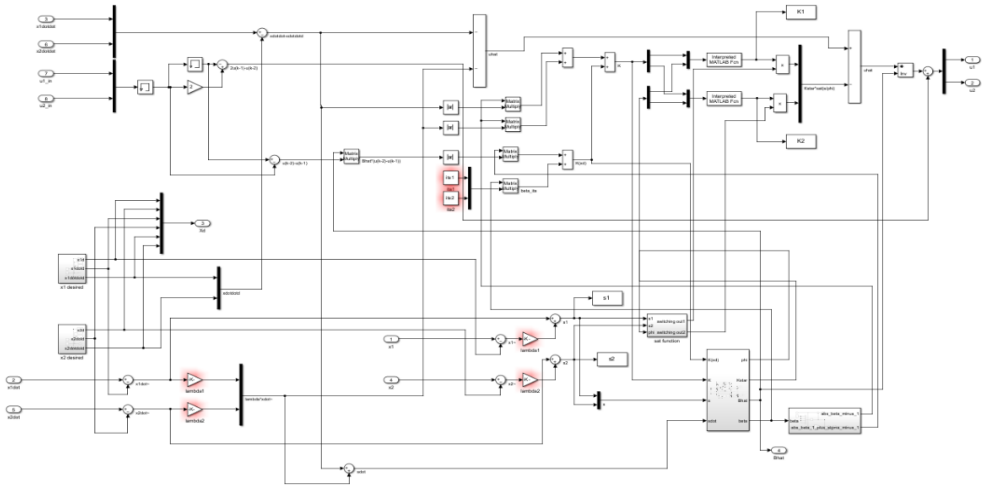


Figure 1.An implementation of MFSMC for a 2x2 system.

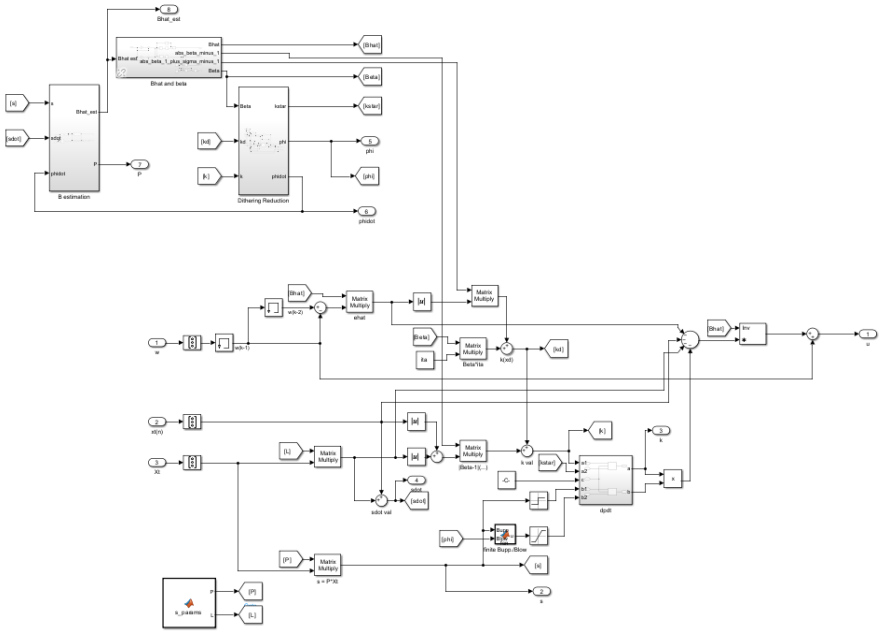


Figure 2. A generalized implementation of MFSMC.

4.0 INFLUENCE MATRIX ESTIMATION

4.1 Previous Approach: Least-Squares

To avoid requiring assumed or known boundaries of the input influence matrix, [5] proposed estimating the influence matrix in real-time. Using a traditional estimator in MFSMC is complicated without a system model because finding a regression equation is not straightforward. In [5], the least-squares with bounded gain forgetting estimator from [4] was adapted. The estimator worked to reduce the error in the sliding condition instead of in an equation directly involving the input influence matrix.

A validation of Hutson's [5] results is given in Figure 3. The estimates were close enough (i.e., within an assumed margin) to the actual influence matrix, so the closed-loop system was stable in the Lyapunov sense. Still, improved estimates are desired.

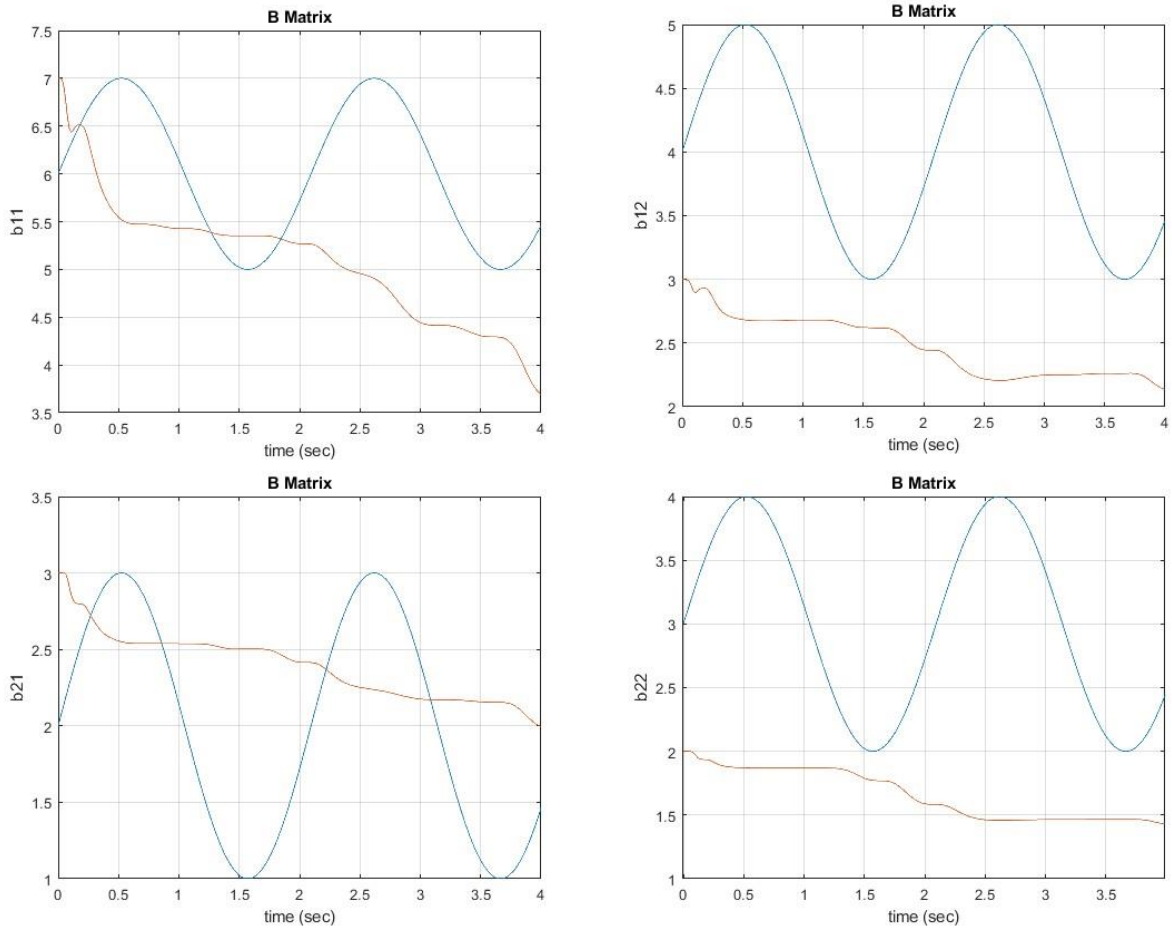


Figure 3. Estimated and real values of B. Estimations are in orange. Real Values are in blue.

4.2 Boundary Estimation

An estimator was created to find the bounds of B rather than the exact value. The estimated bounds could either be paired with a constant $\hat{B}$ or be used to estimate the influence matrix in real-time. As with the previous estimator, the bounds would have to be found without a system model. As a result, the sliding condition is the only relation involving B. The sliding condition cannot be solved quickly[1]. However, a solvable equation may be derived from the inequality.

Again, the sliding condition and the form of the models MFSMC should be able to control are the following:

$$\left(\dot{\vec{\Phi}} - \vec{\eta}\right) \circ |\vec{s}| \geq \vec{s} \circ \dot{\vec{s}} \tag{29}$$

$$\vec{x}^{(n)} = \vec{f}\left(\vec{x}^{(i)}, t\right) + B\left(\vec{x}^{(i)}, t\right)\vec{u} \tag{30}$$

The estimator's goal is to find the influence matrix bounds. First, substitute the model form into the condition and rearrange to get the matrix on one side:

$$sgn(\vec{s}) \circ B\vec{u} \leq sgn(\vec{s}) \circ \left[\vec{x}_d^{(n)} - \vec{f} - \left(\dot{\vec{s}} - \tilde{\vec{x}}^{(n)}\right)\right] + \left(\dot{\vec{\Phi}} - \vec{\eta}\right). \tag{31}$$

Next, flip the inequality and multiply both sides by a factor $\vec{\alpha}$ which is defined as:

$$\vec{\alpha} = \begin{cases} -1 & \left(\dot{\vec{\Phi}} - \vec{\eta}\right) \circ |\vec{s}| \geq \vec{s} \circ \dot{\vec{s}} \\ 1 & else \end{cases} \tag{32}$$

The new relationship encompasses both situations in which the sliding condition is upheld, and in which it is not.

Since the goal is to estimate the bounds of B, an equation may be created whose solution is generally greater than the current B matrix. Here, the derivation diverges for decoupled and coupled systems. We will start with the decoupled case, in which all elements of B off of the diagonal are zero. Defining $\vec{b}$ to be a vector of the diagonal elements of the influence matrix, the left side reduces to:

$$\vec{\alpha} \circ sgn(\vec{s}) \circ B\vec{u} = \vec{\alpha} \circ sgn(\vec{s}) \circ \vec{b} \circ \vec{u} = diag(\vec{\alpha} \circ sgn(\vec{s}) \circ \vec{u})\vec{b}. \tag{33}$$

Next, define:

[1] Solving inequalities would require similar methods to linear programs, which may not find a solution in real-time. They also are not guaranteed to find a solution depending on the set of inequalities.

$$A = |diag(\vec{\alpha} \circ sgn(\vec{s}) \circ \vec{u})| \tag{34}$$

$$\vec{q} = \vec{\alpha} \circ \left(\left| \dot{\vec{\Phi}} - \vec{\eta} + sgn(\vec{s}) \circ \left(\vec{x}_d^{(n)} - \left(\dot{\vec{s}} - \tilde{\vec{x}}^{(n)} \right) \right) \right| + \overline{f} \circ (sgn(\vec{s}))^2 \right) \tag{35}$$

$$\overline{f} = \begin{cases} min(|\vec{f}_{min}|, |\vec{f}_{max}|) & B_{k|k} = B_{min} \\ max(|\vec{f}_{min}|, |\vec{f}_{max}|) & B_{k|k} = B_{max} \end{cases} \tag{36}$$

Using these equations and definitions, the influence matrix may be estimated as:

$$B_{k|k} = max\left(B_{k|k-1}, B_{k-1|k-1}\right) \tag{37}$$

$$B_{k|k-1} = diag(A^{+} \vec{q}) \tag{38}$$

where A^{+} is the pseudoinverse of A. $B_{k|k-1}$ is the exact solution to the equation. However, if any of the current values of B are less than their corresponding maxima, some of the estimated values will be underestimated. Thus, the actual estimate $B_{k|k}$ is found by comparing the equation's solution with the last estimates.

When the system is not decoupled, $\vec{\alpha} \circ sgn(\vec{s}) \circ B\vec{u}$ may be vectorized to obtain $\left[\vec{u}^T \otimes diag\left(\vec{\alpha} \circ sgn(\vec{s})\right)\right] vec(B)$ (applying identities 3 and then 1 from Appendix A.1). Like in the decoupled case, A may be redefined as:

$$A = |\vec{u}^T \otimes diag(\vec{\alpha} \circ sgn(\vec{s}))| \tag{39}$$

and the estimate now comes from:

$$vec\left(B_{k|k-1}\right) = A^{+} \vec{q}. \tag{40}$$

Eq. (40) may be useful for MFSMC. However, many sliding mode controllers are formulated using the parameter δ:

$$\delta = B\hat{B}^{-1} - I. \tag{41}$$

Given Eq. (41), rearrange the inequality to obtain:

$$sgn(\vec{s}) \circ \delta\vec{v} \leq sgn(\vec{s}) \circ \left(\vec{x}_d^{(n)} - \vec{f} - \vec{v} - \left(\dot{\vec{s}} - \tilde{\vec{x}}^{(n)} \right) \right) + \left(\dot{\vec{\Phi}} - \vec{\eta} \right) \tag{42}$$

and apply the previously discussed steps. As explained in Section 3.1.2, $\vec{u} = \hat{B}^{-1}\vec{v}$. The resulting A and $\vec{q}$ are:

$$A = |\vec{v}^T \otimes diag(\vec{\alpha} \circ sgn(\vec{s}))| \tag{43}$$

$$\vec{q} = \vec{\alpha} \circ \left(\left| \dot{\vec{\Phi}} - \vec{\eta} + sgn(\vec{s}) \circ \left(\vec{x}_d^{(n)} - \left(\dot{\vec{s}} - \tilde{\vec{x}}^{(n)} \right) - \vec{v} \right) \right| + max(|\vec{f}_{min}|, |\vec{f}_{max}|) \circ (sgn(\vec{s}))^2 \right) \quad (44)$$

$$vec\left(D_{k|k-1}\right) = A^{+}\vec{q}. \quad (45)$$

Note: D is the estimate for the maximum value of δ. Like with B, $D_{k|k-1}$ should be compared to the previous estimates to keep only the highest values.

5.0 RESULTS

5.1 General Implementation

5.1.1 Two-State Comparison with Original Implementation

To ensure the improved MFSMC implementation's output aligns with the theory, both the new and original controllers were tested against the test system shown below:

$$\ddot{x}_1 = -a_1(t)\dot{x}_1{}^2\, cos(2x_1)\, x_2 + b_{11}(t)u_1 + b_{12}(t)u_2 \tag{46}$$

$$\ddot{x}_2 = -a_2(t)\dot{x}_2{}^2\dot{x}_1 x_2 + b_{21}(t)u_1 + b_{22}(t)u_2 \tag{47}$$

A comparison between similar simulations for each implementation is given in Figures 4-7 and Table 1.

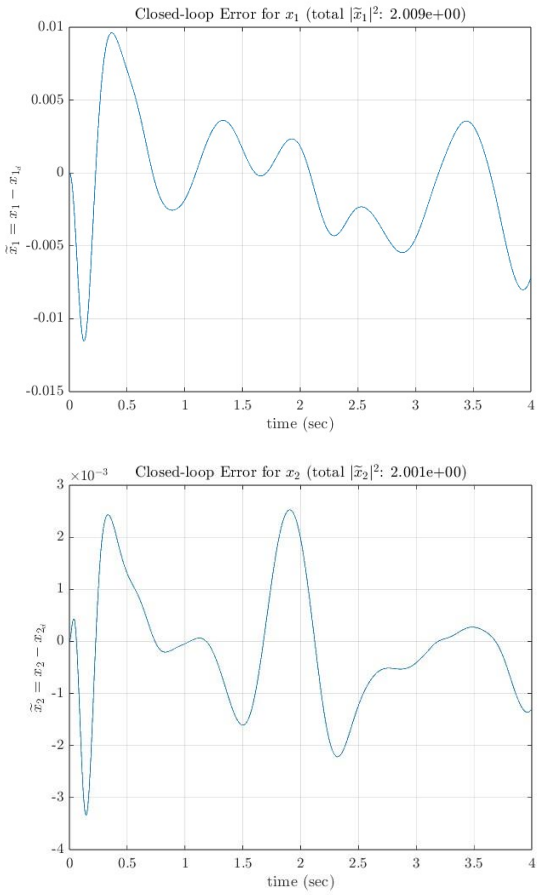
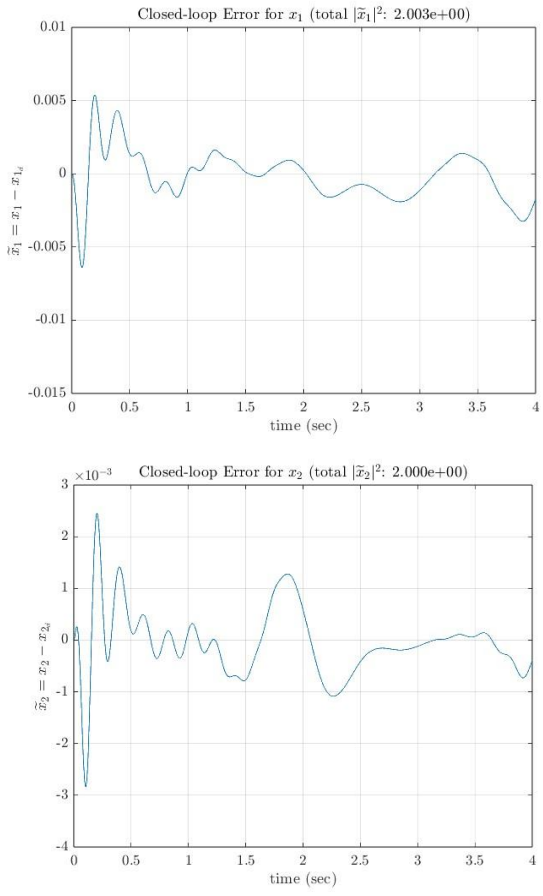
Original	Improved
	

Figure 4.Closed-Loop Error responses for the original and improved implementations.

The coefficient values and controller settings were identical in both simulations. The main controllers were created from identical derivations. However, the new controller included

an estimator dead zone to stop the estimator when $\vec{s}$ was close enough to zero. The dead zone—along with numerical differences in the Simulink blocks—explains the new implementation's decreased error but increased control magnitude in Table 1 as well as in Figures 4 and 5, respectively. These differences also explain the high frequency mode in the improved implementation. To decrease the chances of high frequency excitations, the SMC lambda parameter may be tuned to a lower value. In both cases, the system was stable. These results verify the generalized implementation's correctness.

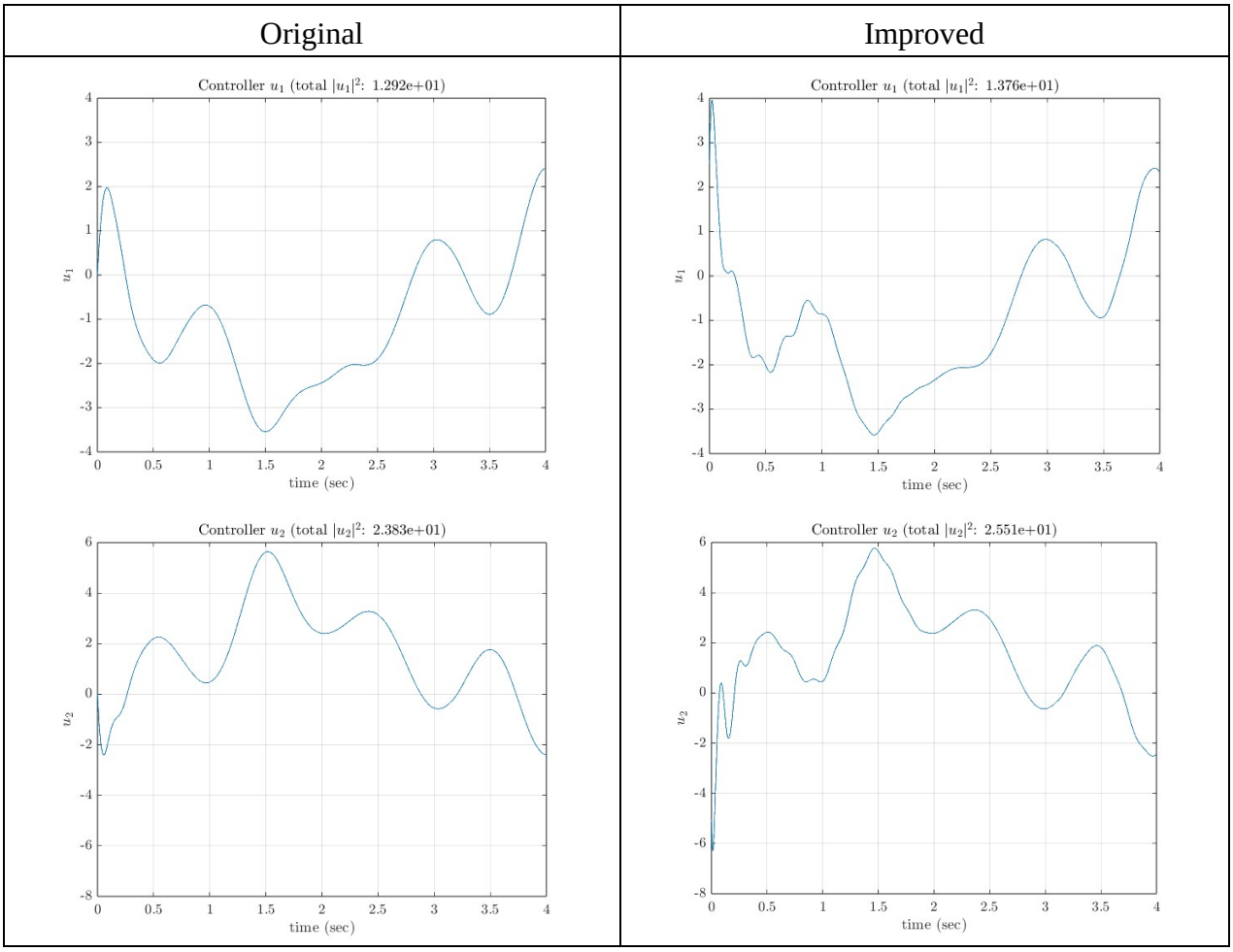


Figure 5.Control inputs for the original and improved implementations.

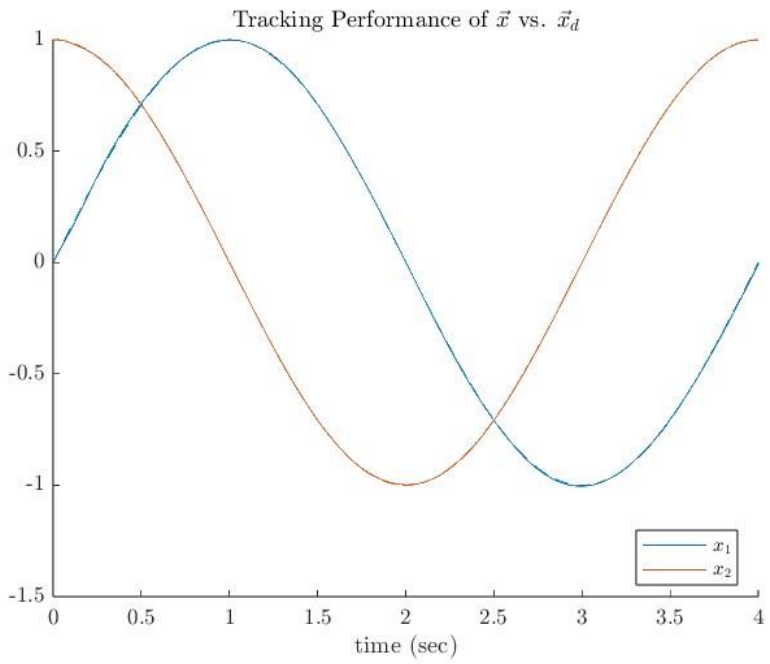


Figure 6. Original implementation's tracking performance.

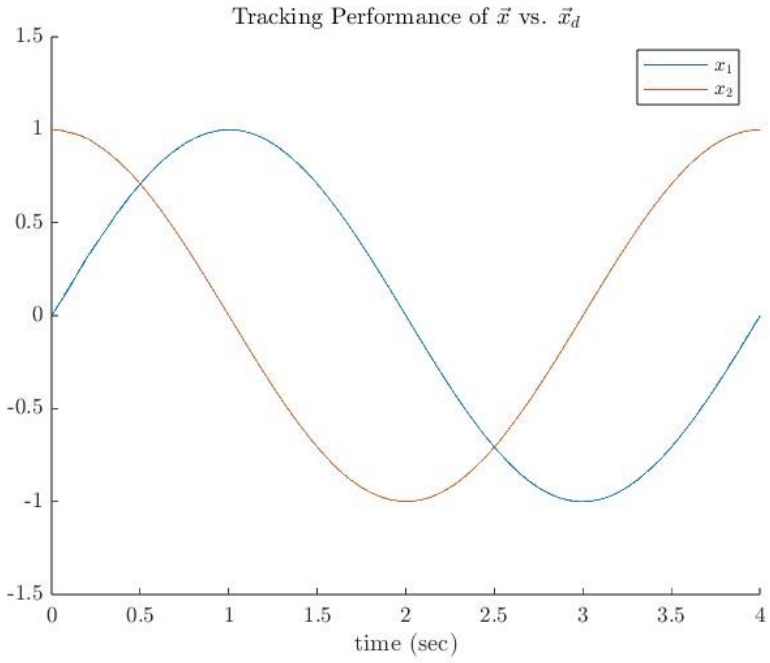


Figure 7. Improved implementation's tracking performance.

	Original	Improved
$\int \lvert\tilde{x}_1\rvert^2\, dt$	2.009	2.003
$\int \lvert\tilde{x}_2\rvert^2\, dt$	2.001	2.000
$\int \lvert u_1\rvert^2\, dt$	12.92	13.76
$\int \lvert u_2\rvert^2\, dt$	23.83	25.51

Table 1. Integral squared magnitudes of closed-loop errors and control inputs for both implementations.

5.1.2 Four-State System Validation

The generalized implementation was also simulated with a randomized Four-state linear system ($\dot{\vec{x}} = A\vec{x} + B\vec{u}$). While B was diagonal with random elements, the A matrix was completely random. Typical results are given in Figures 8 and 9.

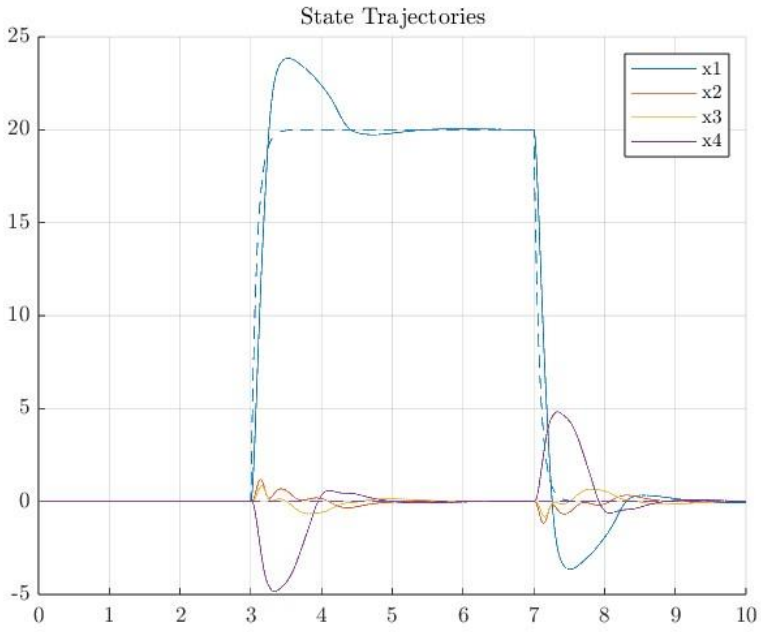


Figure 8. State trajectories for a four-state system with generalized MFSMC.

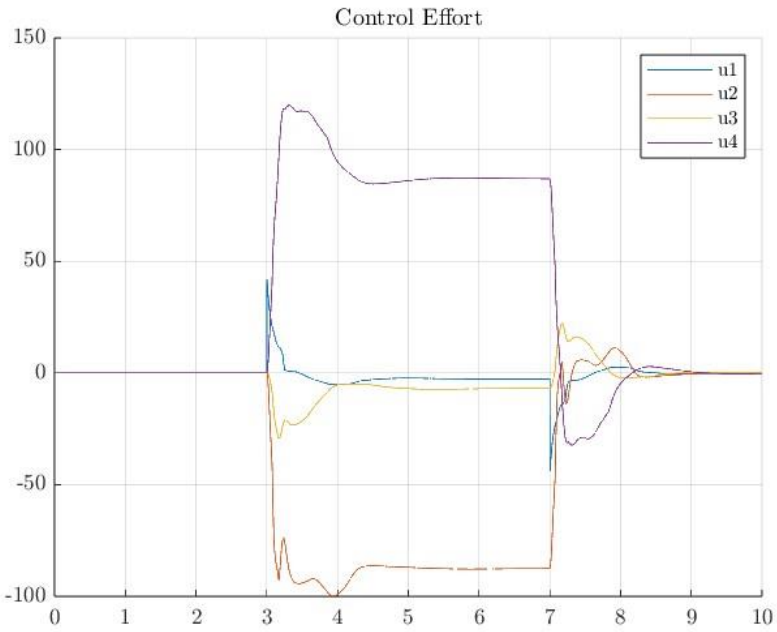


Figure 9. Control effort for a four-state system with generalized MFSMC.

The figures show behavior similar to the previous MFSMC example: good tracking (though there was a higher overshoot) and an implementable control effort. The overshoot may be due to a poor control influence estimate from insufficient excitation. Regardless, the closed-loop system was stable.

More importantly, the simulation was created using an exact copy of the controller used in the previous section. All controller parameters were the same except for the initial B estimate, which was set as the actual influence matrix with a random error added to each element. Since Simulink was able to compile and run the simulation, the premise of the generalized implementation is validated – generally implemented MFSMC schemes can be applied to any system.

5.2 Boundary Estimation in SMC

5.2.1 Decoupled Performance Comparison

Initial tests of the boundary estimator consisted of a sliding mode controller developed for both decoupled and coupled versions of the previously mentioned system. Random sinusoids were used to generate a_1 and a_2 in Eqs. (46) and (47).

Simulation results for the decoupled controller and system with a constant influence matrix are presented in Figures 10-13. The results primarily serve as a validation; since the sliding condition was always met (see Figure 11, left panel: no difference between dotted and dashed lines), no additional information may be given to the estimator. The estimator is unable to

adjust its values as a result. While the lack of adjustment resulted in underestimated values, the controller's tracking performance was comparable to a SMC scheme with known bounds on the control matrix. Both closed-loop systems were stable.

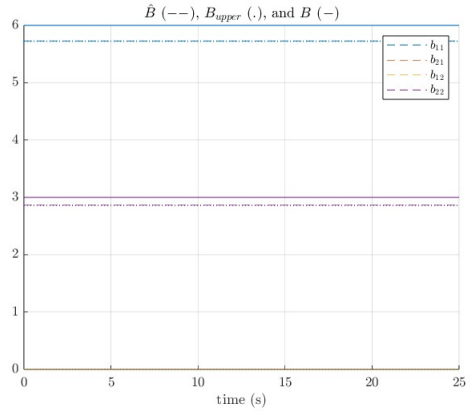


Figure 10. Best guess, estimated, and actual B values for a decoupled, constant B system.

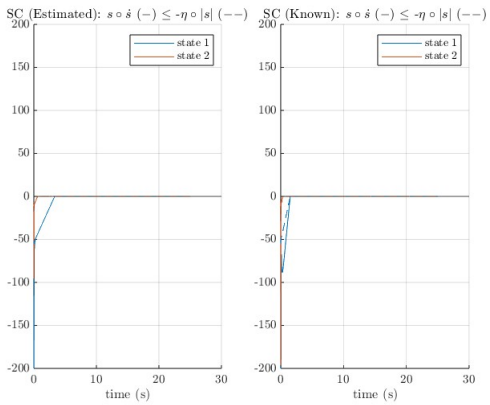


Figure 11. Sliding condition with (left) and without (right) boundary estimation for a constant B system.

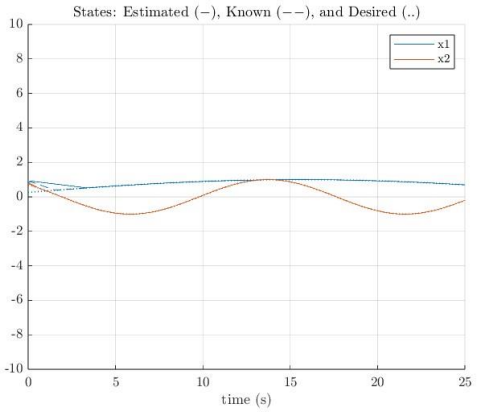


Figure 12. Tracking performance with and without boundary estimator for a constant B system.

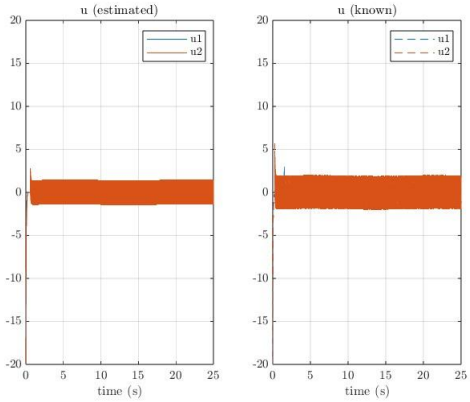


Figure 13. Control effort with (left) and without (right) boundary estimation for a constant B system.

Typical simulation results for the decoupled controller and system with a varying influence matrix are given in Figures 14-17. The B values were generated using the last technique discussed in Appendix A.3. The estimator system performed basically comparably to the regular SMC system; it reached the desired trajectories slightly faster for state 2, but slower for state 1. The speed might be due to the low estimate for b_{11}'s bounds at first (as seen in Figure 14). The estimate increased later in the simulation. Since good performance cannot be guaranteed during any SMC reaching phase [21], perfect tracking once the system reaches the desired states is more important. Tracking performance was approximately equal after reaching the desired trajectories.

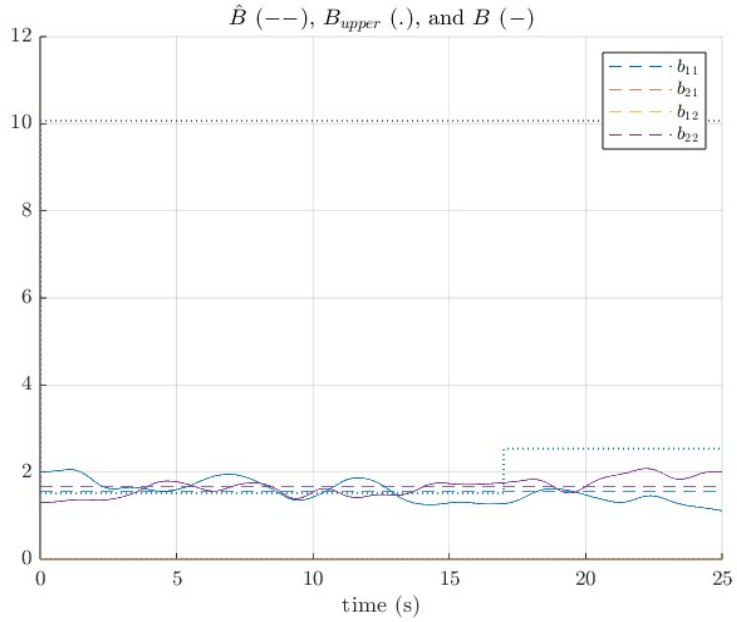


Figure 14. Real (solid), best guess (dashed) and estimated maximum (dotted) values of B.

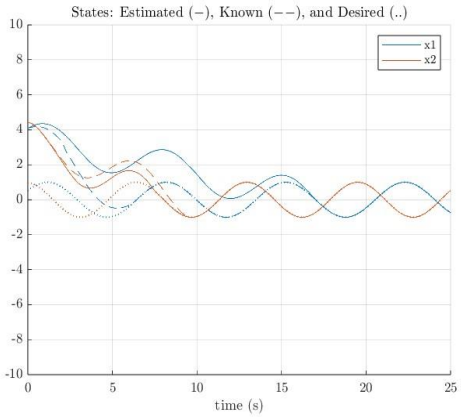


Figure 15. Desired (dotted), SMC (dashed) and SMC with boundary estimation (solid) state trajectories.

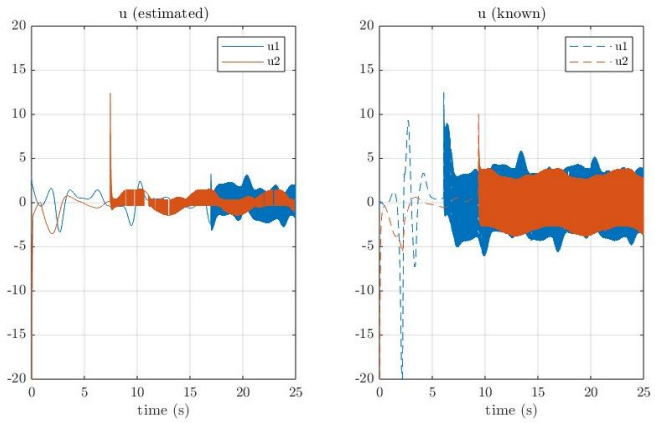


Figure 16. Control inputs for SMC with and without boundary estimation (left and right, respectively).

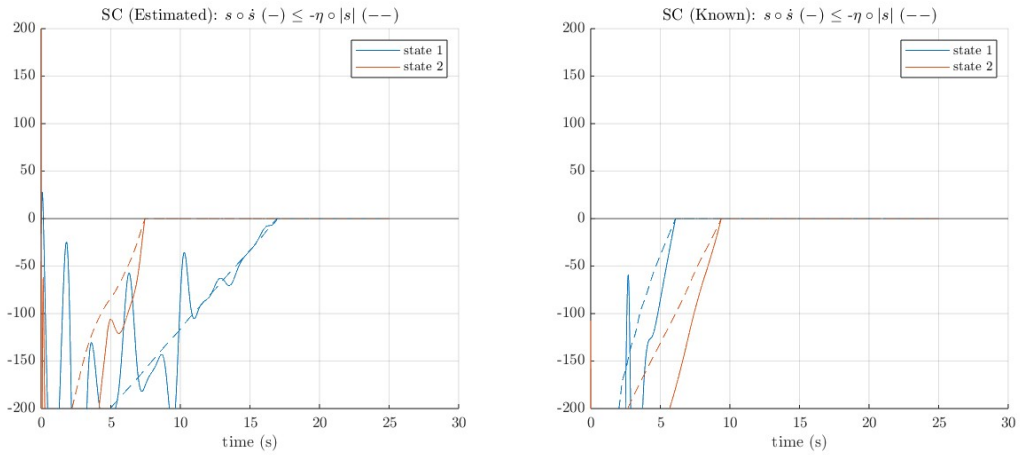


Figure 17. Sliding condition states with and without boundary estimation (left and right, respectively)

It is important to note the boundary values are overestimates. The overestimation would guarantee stability while suggesting the control input magnitude is increased. However, Figure 16 shows the magnitude decreased, including during the reaching phase. The result, along with the sliding condition in Figure 17 suggests the controller was more energy efficient both in the controller magnitude and from a sliding condition standpoint. The sliding condition efficiency is especially seen in state 1: though it was slower than the known system, the condition error is close to zero in the end.

5.2.2 Coupled Performance

To test the estimator's coupled performance, an SMC system with boundary estimation was simulated with a coupled version of Eqs. (46) and (47). Again, the system parameters were generated using sinusoids as well as the last technique discussed in Appendix A.3. Typical simulations are given in Figures 18-22.

Like the previous simulations, the system is stable with perfect tracking after some time. In the current simulation, however, the effect of the estimator on the state trajectories and sliding function is somewhat visible. The initial estimate increase reorients the second state towards its desired trajectory (see Figure 19). The secondary increases (between 10 and 12 seconds) were the result of the estimator increasing its estimates to stop instability in the second state (see Figure 20, expansions correspond to when $s_2\dot{s}_2$ is positive). Finally, Figure 20 shows the system became more sliding-mode efficient, especially around the time state 1 reached zero error.

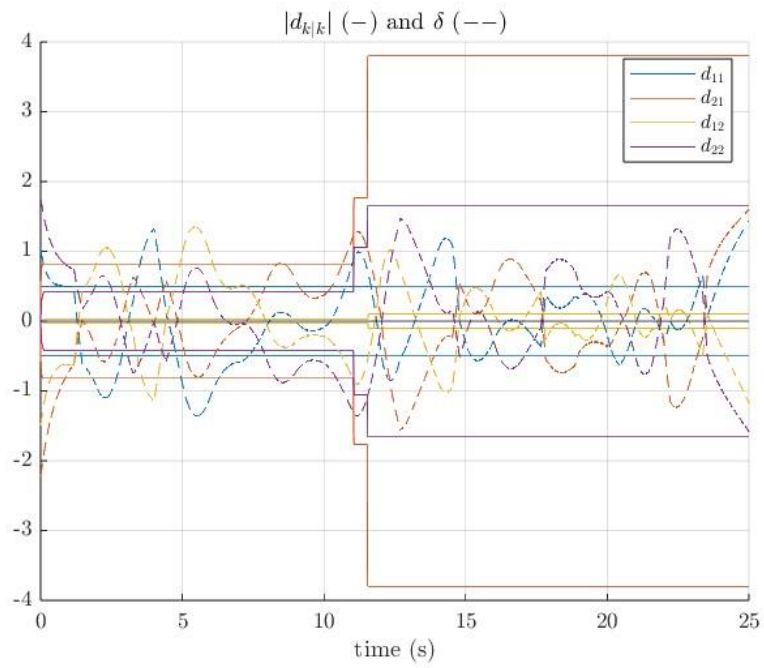


Figure 18. Real (dashed) and estimated boundaries (solid) of delta.

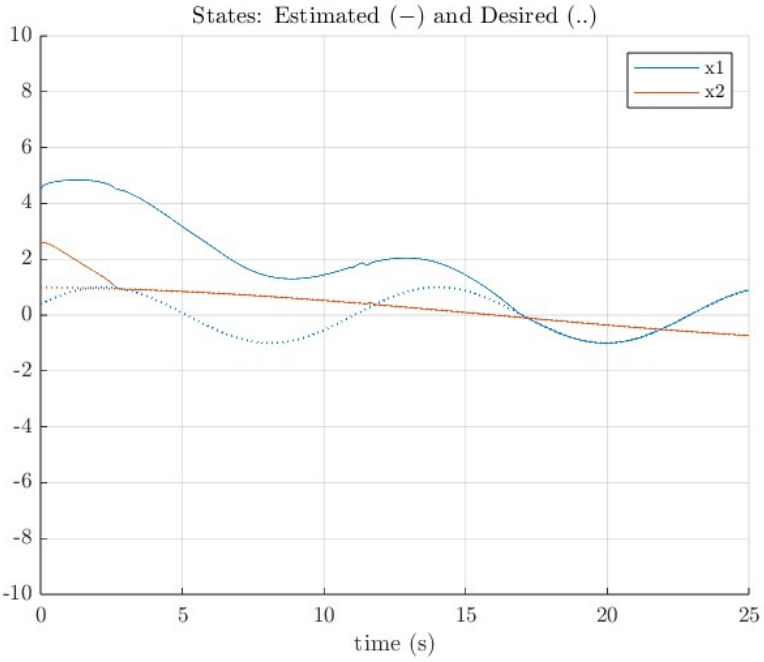


Figure 19. State trajectories for coupled SMC with boundary estimation.

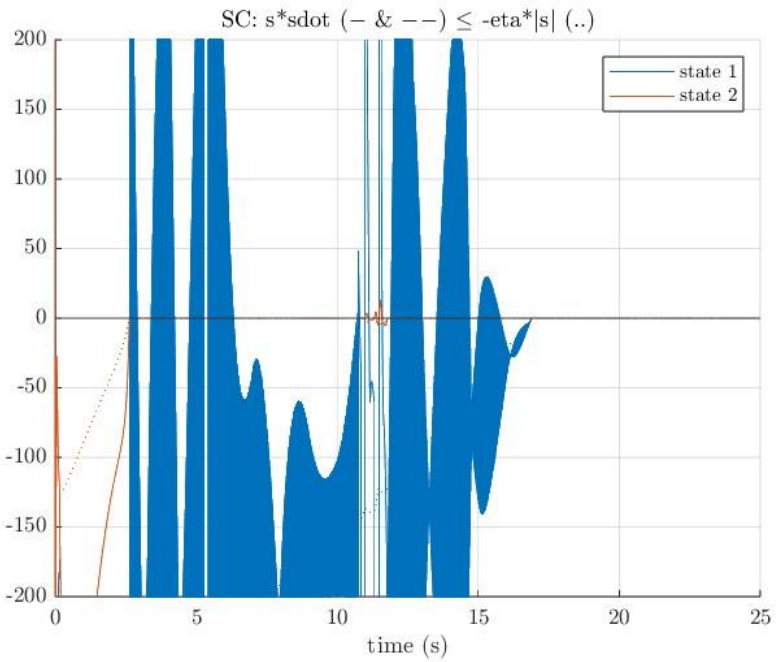


Figure 20. Sliding Condition for coupled SMC with boundary estimation.

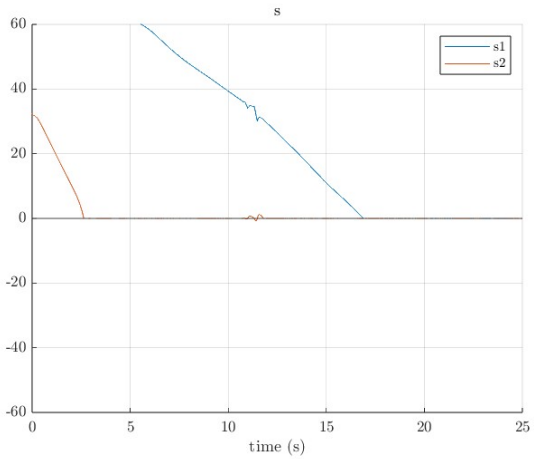


Figure 21. Sliding mode value for coupled SMC with boundary estimation

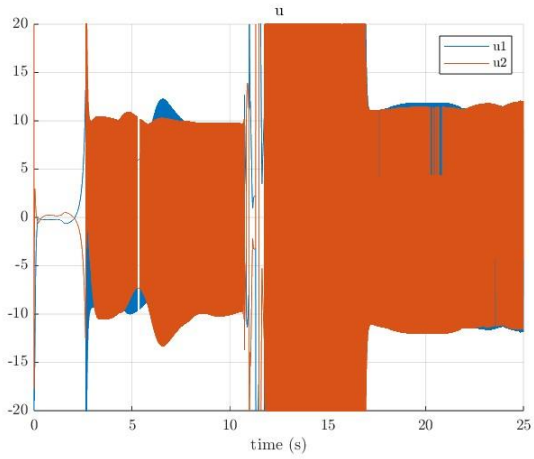


Figure 22.Control effort for coupled SMC with boundary estimation

6.0 CONCLUSION

This work developed a new MFSMC implementation as well as an influence gain estimator improvement scheme. The implementation was compared with previous methods in simulations using a nonlinear system model. The controllers performed comparably, with the new implementation exhibiting lower error at the cost of a slightly higher control effort. The estimation technique was paired with a SMC scheme and simulated against a controller with known bounds. Again, the controllers performed similarly, especially at the end of the SMC reaching phases. The controller with the estimator, however, showed a lower control effort.

Both improvements move MFSMC towards its main goal: to be a control system where only the system order and number of states need to be known. Such a controller would decrease development and testing times for any system due to the avoidance of usual control system overheads (e.g. tuning). While previous works had achieved the goal mathematically, the new implementation was shown to achieve the goal practically. As a result, no other MFSMC schemes for square systems need to be implemented. The controller also does not need to be implemented in code; Simulink can automatically convert the model. Furthermore, the techniques presented in this work may be used to afford the same advantages to non-square MFSMC systems when they are created.

The boundary estimator, on the other hand, improves MFSMC in the same ways as previous estimators (i.e., relaxing the need to know the influence matrix a priori), while eliminating the need for assumptions on the matrix's upper and lower margins. As a result, the new estimator could cause comparable tracking as previous MFSMC systems. An MFSMC boundary estimator could also be more energy and sliding-mode efficient. Previous tests show the estimator could give lower controller magnitudes, using less energy for the same tracking performance. Furthermore, since the estimator is meant to better meet and not necessarily exceed the sliding condition, it better drives the sliding condition to zero. However, no proof is given that the values will be more efficient, and so this is not guaranteed.

6.1 Accomplished Goals

The original goals for this work are outlined in Section 1.1. As previously mentioned, the focus shifted away from Goal 1. The shift was due to time limitations and the sufficient coupled-system performance of previously derived MFSMC schemes (as discussed in Section 3.2). The majority of this work was dedicated to accomplishing Goals 2 and 3 and demonstrating the accomplishments using Goal 4. Goals 2 and 3 are presented in Sections 3.3 and 4.0, respectively. The original intent of Goal 4 was to present simulation results using system models of various forms. However, other than the four-state linear system in Section 5.1.2, the system form given by Eqs. (46) and (47). No work towards Goal 5 was performed due to time limitations.

6.2 Future Work

To generate better estimates—and system performance—when using MFSMC, a boundary estimator could be implemented. Currently, the boundary estimator would be used with a constant estimate of the influence matrix. In the future, work should be done to use the boundary estimator to find a better $\hat{B}$ for the sake of performance. The better value could simply be generated using Eq. (10) (the same relationship as in the current MFSMC implementation). In addition, a more formal proof for the convergence and efficiency of the estimator should be found.

Also, future MFSMC versions should either be an adaptation of the general implementation or should use the same techniques. Specifically, the techniques should be used to implement an MFSMC system for non-square systems. Doing so would allow MFSMC to truly only depend on the system order and the number of states (rather than just in theory). As a result, implementing a MFSMC scheme will be much easier while development time is severely reduced.

7.0 ACKNOWLEDGMENTS

7.1 Financial Support

The work was performed in fulfilment of a DoD Air Force AFWERX Phase I STTR, Award # FA864923P0965.

7.2 Further Acknowledgements

I would like to thank Dr. Crassidis for advising me through this process.

Thank you, committee members, for your support and feedback.

Thank you, Carola, Aashrita, and everyone else who worked in the lab for being great friends and lab mates.

Thank you, Katarina Wayman, for everything you do for the department (including answering all of my questions and helping to keep me on track).

Finally, thank you, Riesa, for your ongoing love and support.

8.0 REFERENCES

[1] Mizov, A., 2015, “A Model-Free Control Algorithm Derived Using the Sliding Model Control Method,” Rochester Institute of Technology.

[2] Reis, R. M., 2016, “A New Model-Free Sliding Mode Control Method with Estimation of Control Input Error,” Rochester Institute of Technology.

[3] El Tin, F., 2017, “A Model-Free Control System Based on the Sliding Mode Control Method with Applications to Multi-Input-Multi-Output Systems,” Rochester Institute of Technology.

[4] Islam, M. S., 2020, “A Model-Free Control System Based on the Sliding Mode Control with Automatic Tuning Using as On-Line Parameter Estimation Approach,” Rochester Institute of Technology.

[5] Hutson, N., 2023, “Model-Free Sliding Mode Control in the Lateral and Direction Dynamics of an Aircraft,” Rochester Institute of Technology.

[6] Zhou, F., and Fisher, D. G., 1991, “MIMO Sliding Mode Control: A Lyapunov Approach,” *1991 American Control Conference*, IEEE, Boston, MA, USA, pp. 1796–1799.

[7] Pang, H.-P., and Tang, G.-Y., 2008, “Global Robust Optimal Sliding Mode Control for a Class of Uncertain Linear Systems,” *2008 Chinese Control and Decision Conference*, IEEE, Yantai, Shandong, China, pp. 3509–3512.

[8] Huang, Z., and Sun, C., 2023, “Adaptive Global Robust Tracking Control for Uncertain Dynamic Systems,” Int. J. Control Autom. Syst., **21**(4), pp. 1070–1079.

[9] Schulken, E., “Investigations of Model-Free Sliding Mode Control Algorithms Including Application to Autonomous Quadrotor Flight.”

[10] Sreeraj, A., Kaputa, D., and Crassidis, A., 2019, “A Model-Free Control Algorithm Based On The Sliding Mode Control Method With Applications to Unmanned Aircraft Systems.”

[11] Shaferman, V., Schwegel, M., Glück, T., and Kugi, A., 2021, “Continuous-Time Least-Squares Forgetting Algorithms for Indirect Adaptive Control,” European Journal of Control, **62**, pp. 105–112.

[12] Na, J., Yang, J., Ren, X., and Guo, Y., “Adaptive Online Estimation of Time-Varying Parameter Nonlinear Systems.”

[13] Yongliang Zhu, and Pagilla, P. R., 2003, “Adaptive Estimation of Time-Varying Parameters in Linear Systems,” *Proceedings of the 2003 American Control Conference, 2003.*, IEEE, Denver, CO, USA, pp. 4167–4172.

[14] Pan, Y., and Yu, H., 2018, “Composite Learning Robot Control with Guaranteed Parameter Convergence,” Automatica, **89**, pp. 398–406.

[15] Korotina, M., Romero, J. G., Aranovskiy, S., Bobtsov, A., and Ortega, R., 2021, “Persistent Excitation Is Unnecessary for On-Line Exponential Parameter Estimation: A New Algorithm That Overcomes This Obstacle.”

[16] Moshksar, E., and Guay, M., 2014, “Invariant Manifold Approach for Adaptive Estimation of the Time-Varying Parameters for a Class of Nonlinear Systems,” *2014 American Control Conference*, IEEE, Portland, OR, USA, pp. 2261–2266.

[17] Shao, J., and Chen, Y.-Y., 2020, “Distributed Parameter Estimation Using Invariant Manifold Approach,” *2020 5th International Conference on Automation, Control and Robotics Engineering (CACRE)*, pp. 302–307.

[18] Murray, R. M., Li, Z., and Sastry, S. S., 2017, “Ch4. Lyapunov Stability Theory,” *A Mathematical Introduction to Robotic Manipulation*, CRC Press, pp. 43–53.

[19] Slotine, J.-J. E., and Li, W., 1991, “Chapter 7: Sliding Control,” *Applied Nonlinear Control*, Prentice Hall, pp. 277–310.

[20] Geeks for Geeks, 2020, “Find the Nth Row in Pascal’s Triangle,” GeeksforGeeks [Online]. Available: https://www.geeksforgeeks.org/find-the-nth-row-in-pascals-triangle/. [Accessed: 25-Jun-2024].

[21] Pang, H.-P., and Wang, L.-P., 2010, “Global Robust Sliding Mode Control for a Class of Uncertain Time-Delay Systems,” *2010 International Conference on Machine Learning and Cybernetics*, IEEE, Qingdao, China, pp. 910–915.

[22] Taboga, M., 2021, “Vec Operator,” Lectures on matrix algebra [Online]. Available: https://www.statlect.com/matrix-algebra/vec-operator. [Accessed: 25-Jun-2024].

[23] Wikipedia contributors, 2024, “Jacobi’s Formula — Wikipedia, The Free Encyclopedia.”

[24] Wikipedia contributors, 2024, “Matrix Determinant Lemma — Wikipedia, The Free Encyclopedia.”

A. APPENDIX

A.1 Useful Mathematical Identities

Given matrices A, B, and C, as well as vectors $\vec{s}$ and $\vec{u}$:

1. $vec(ABC) = [C^T \otimes A]vec(B)$ [22]
2. $B\vec{u} = vec(B\vec{u}) = [\vec{u}^T \otimes I]vec(B)$
3. $diag(\vec{s})\vec{u} = \vec{s} \circ \vec{u} = diag(\vec{u})\vec{s}$

A.2 Simulink Models for Example Systems

A.2.1 Plant Models

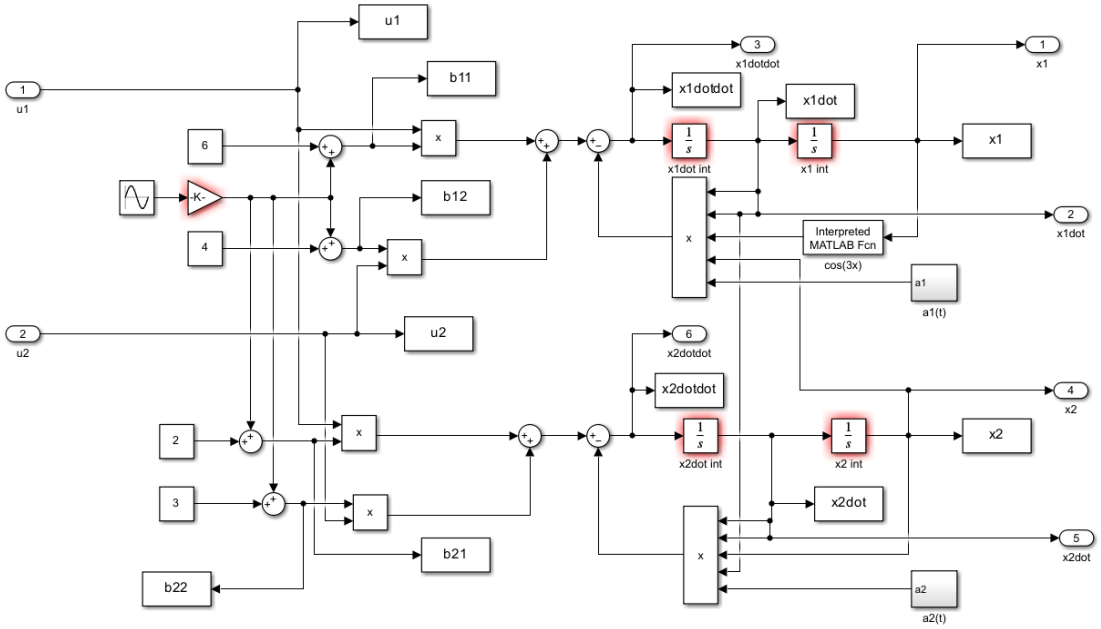


Figure 23. Original test plant for Eqs. (46) and 24(47) (without random B matrix, used in MFSMC tests).

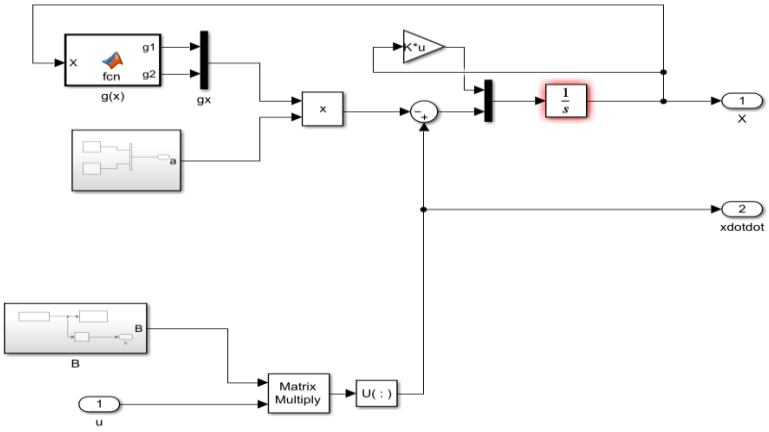


Figure 24.Test plant for Eqs. (46) and 24(47) with random B matrix (used in SMC tests).

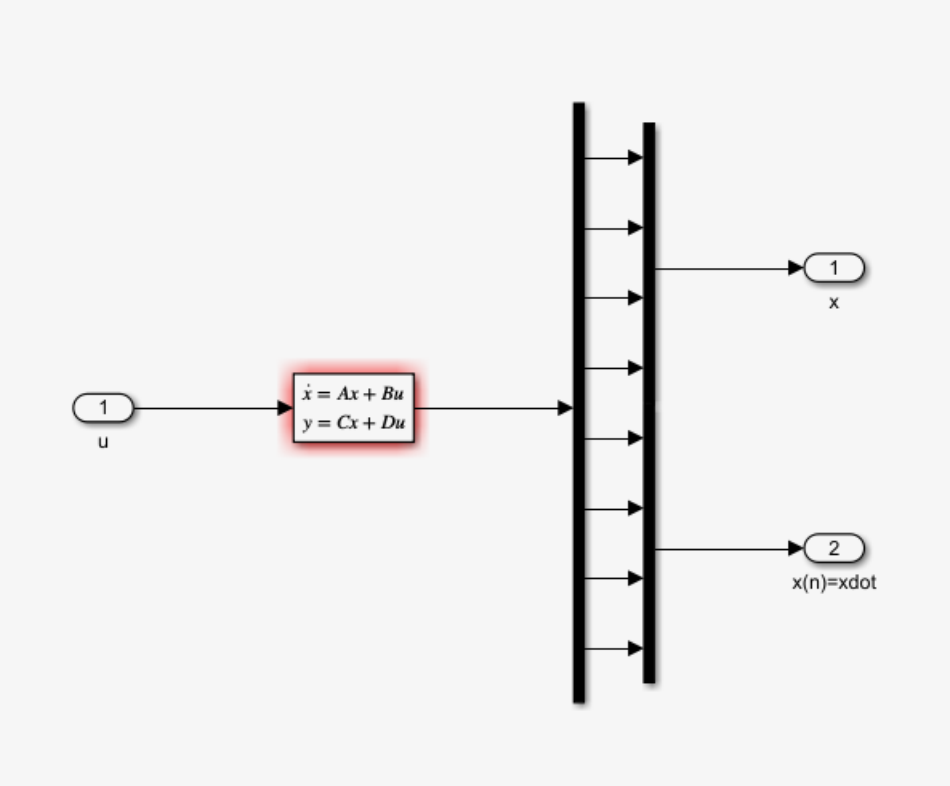


Figure 25. Linear test plant (used for four-state MFSMC test).

A.2.2 MFSMC Diagrams

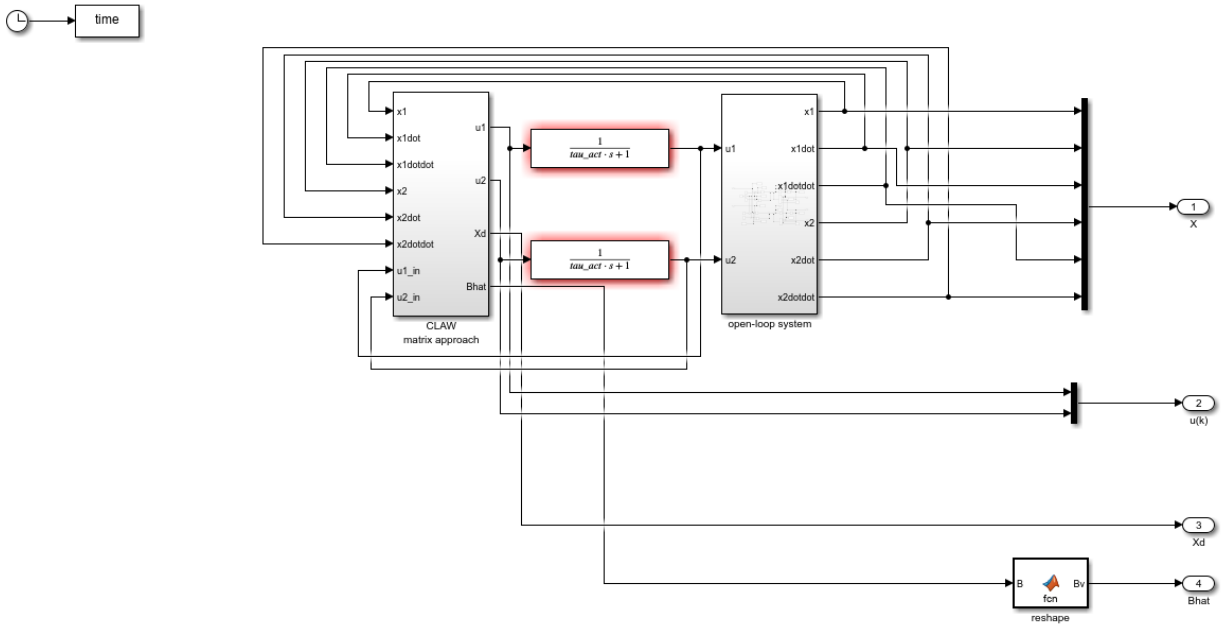


Figure 26. Full MFSMC plant and controller Simulink model.

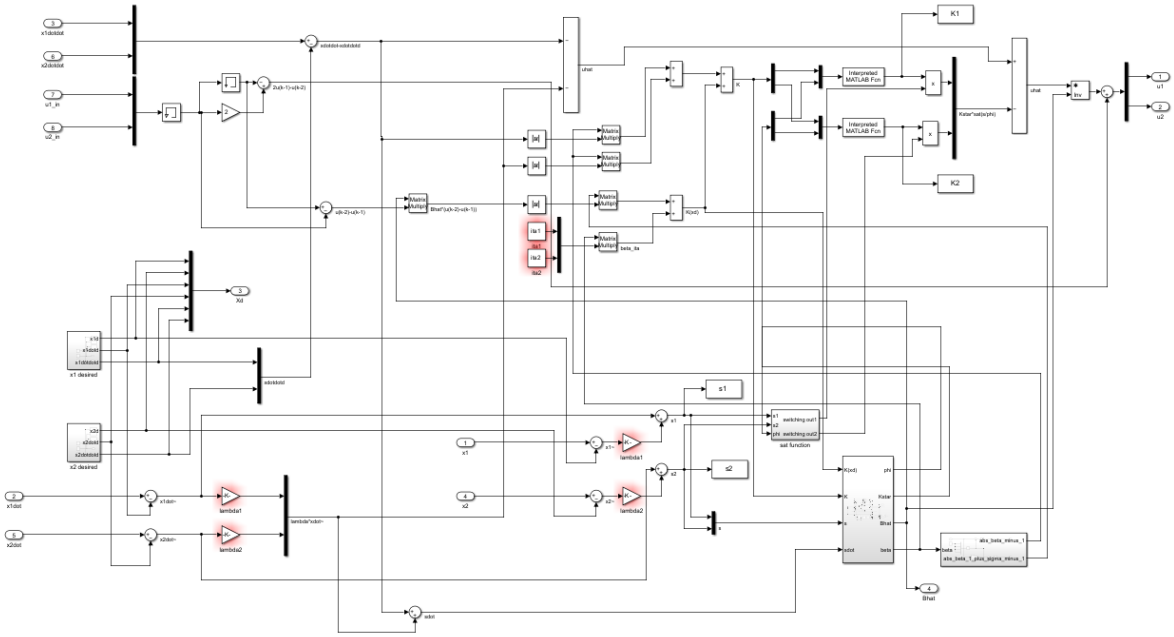


Figure 27. Original MFSMC implementation.

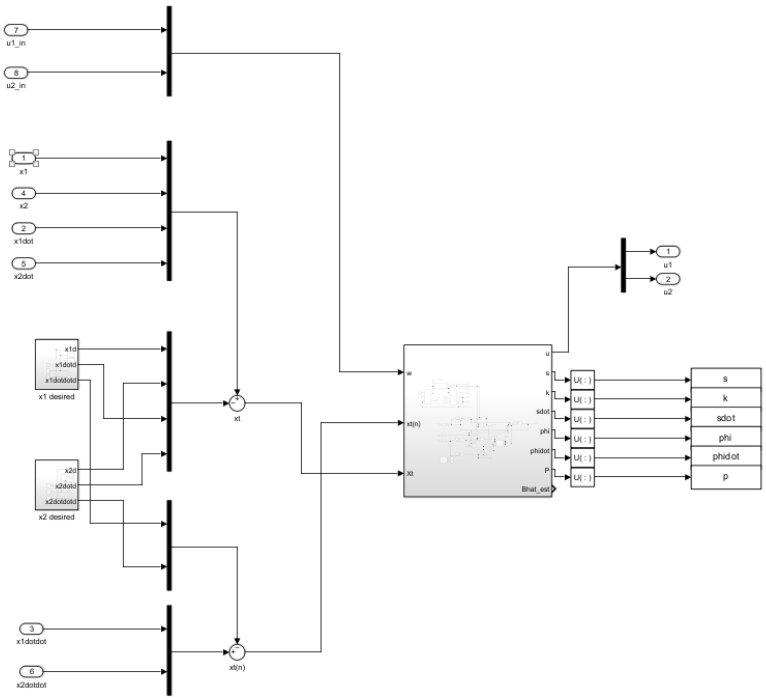


Figure 28. Connections to the new MFSMC implementation

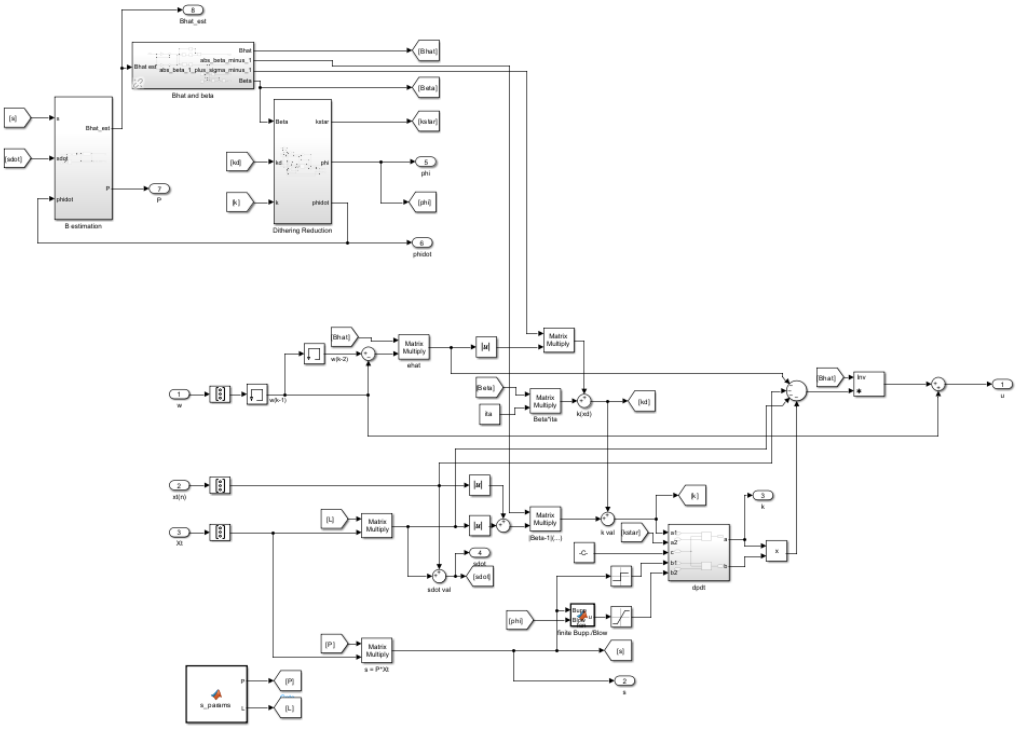


Figure 29. New MFSMC implementation.

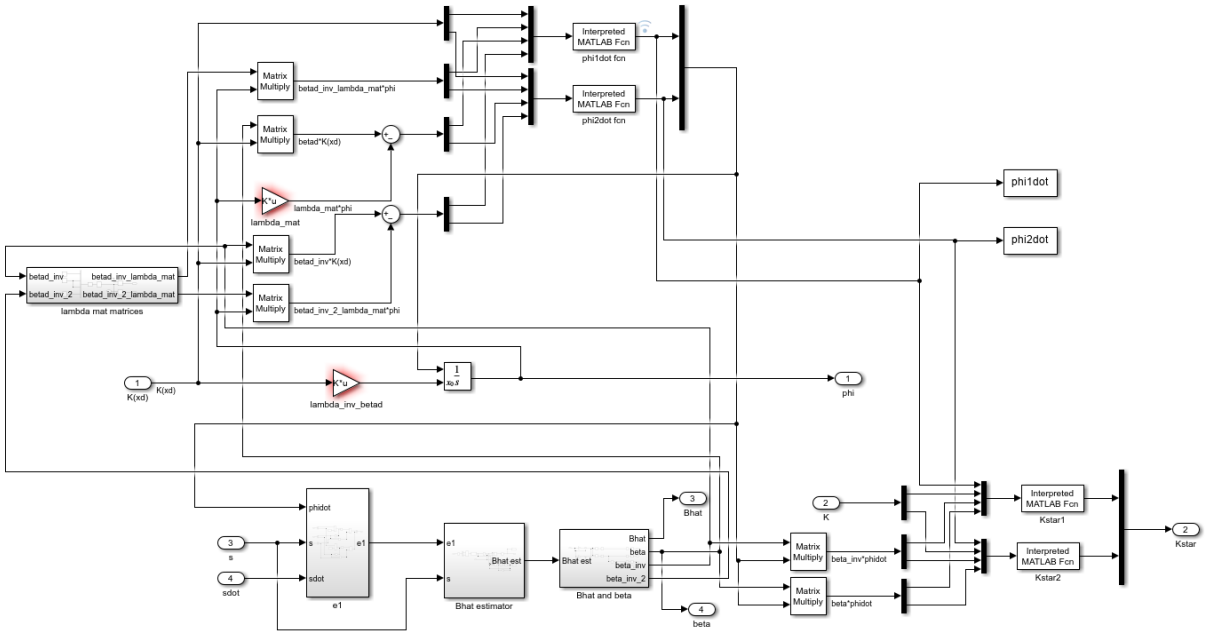


Figure 30.MFSMC boundary layer and estimator.

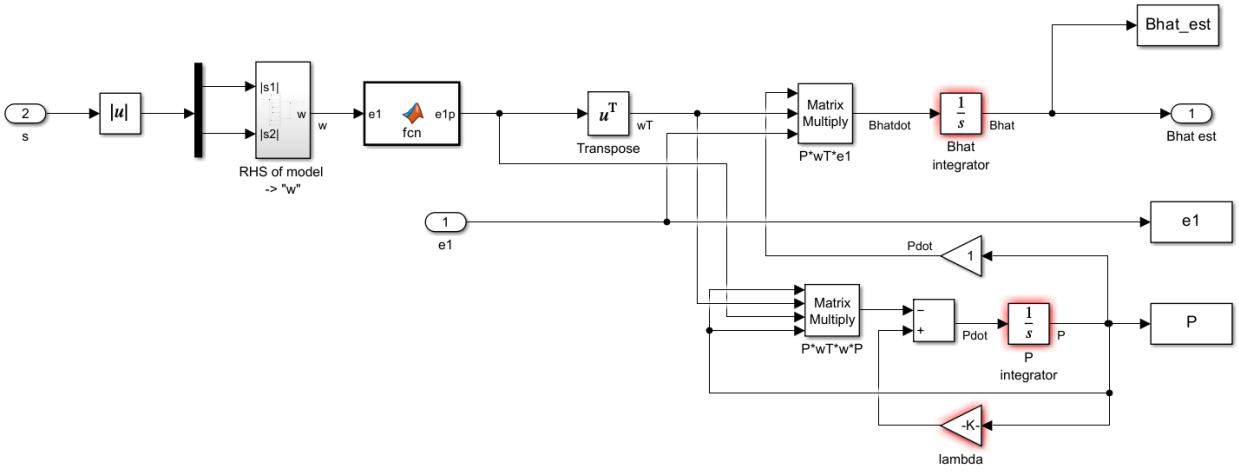


Figure 31. MFSMC estimator internals.

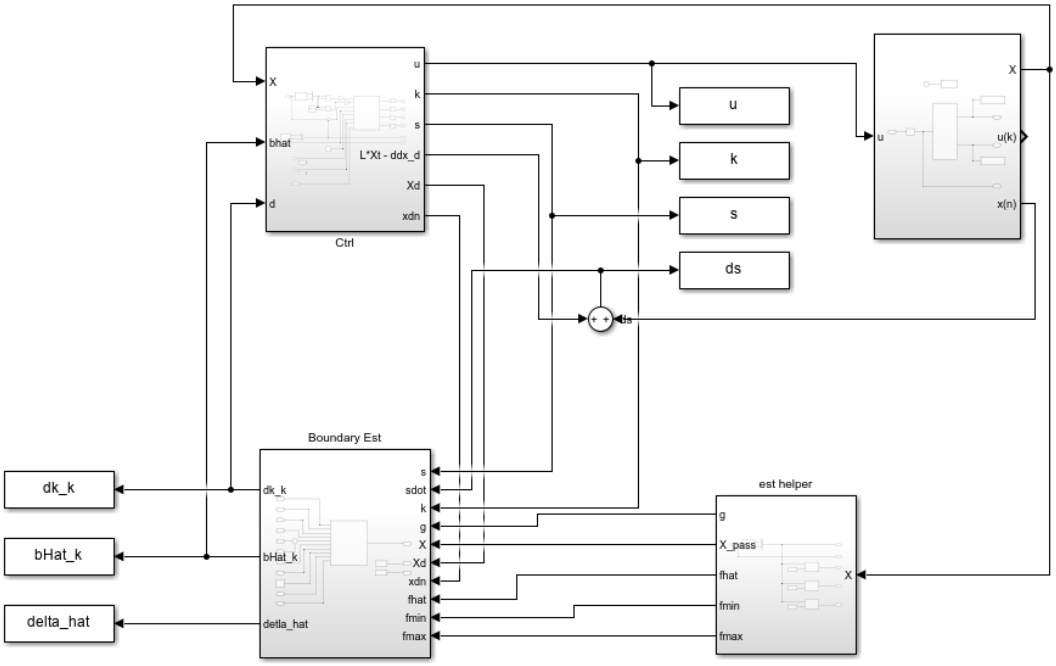


Figure 32. SMC Simulink diagram. Models without boundary estimation only have the top two blocks.

A.2.3 SMC and Boundary Estimation

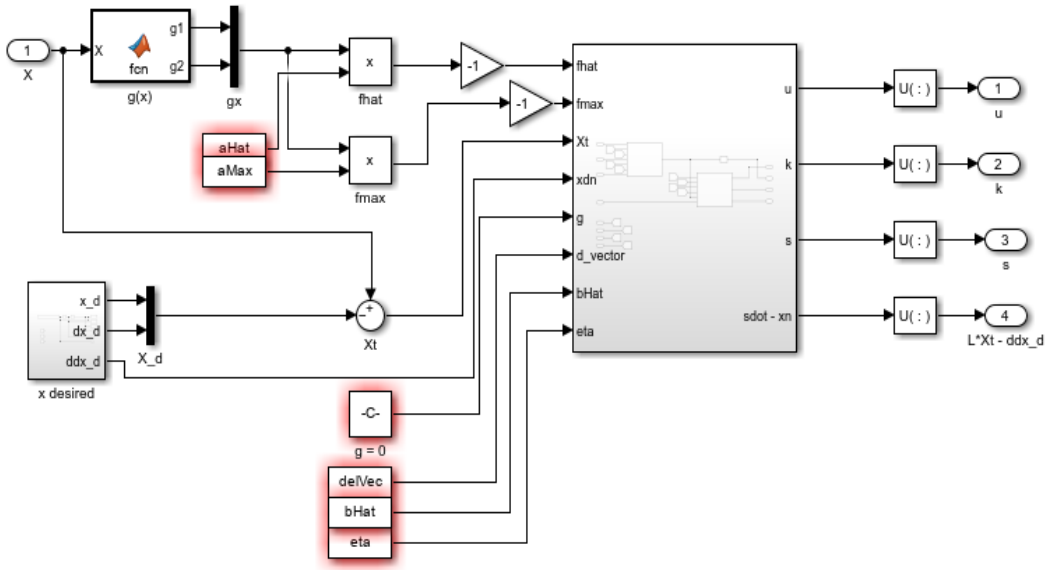


Figure 33. SMC diagram.

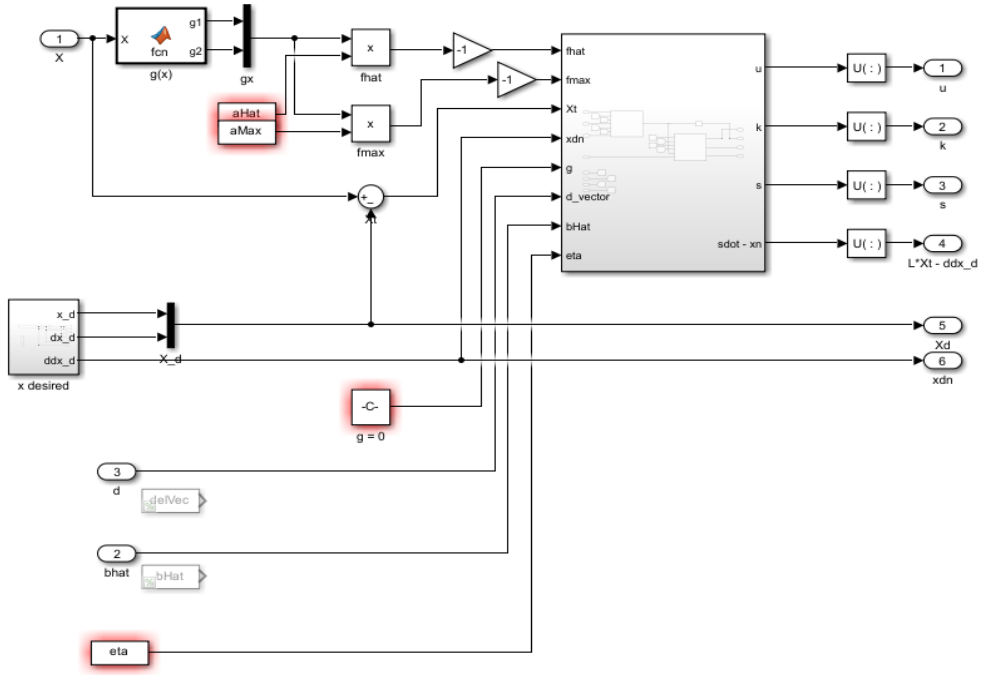


Figure 34. SMC with boundary estimation.

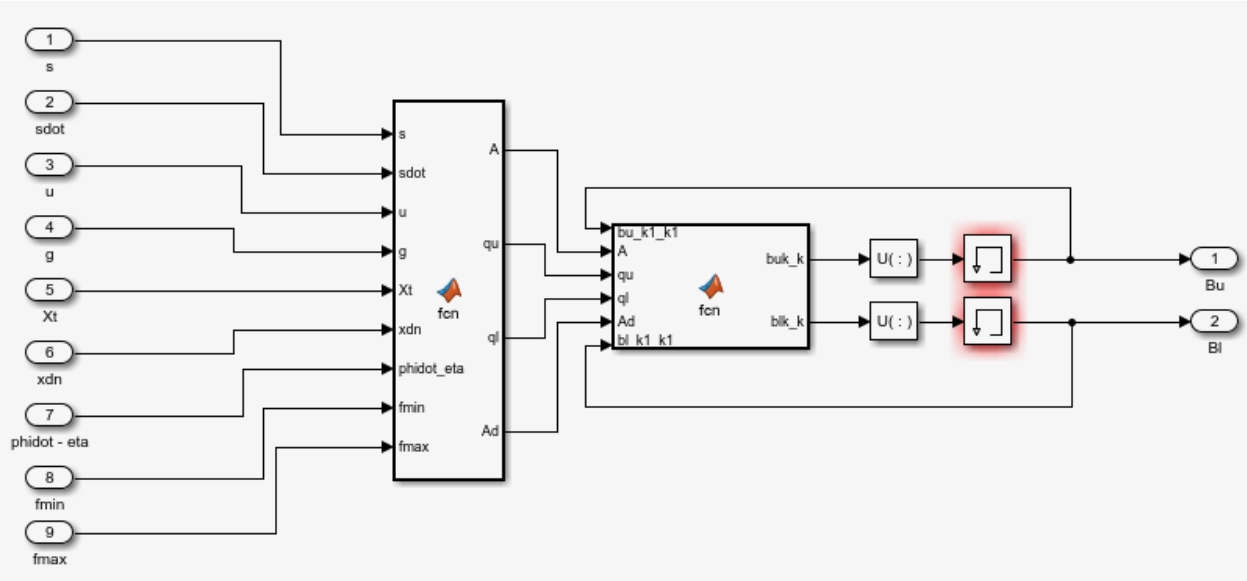


Figure 35. Decoupled boundary estimator.

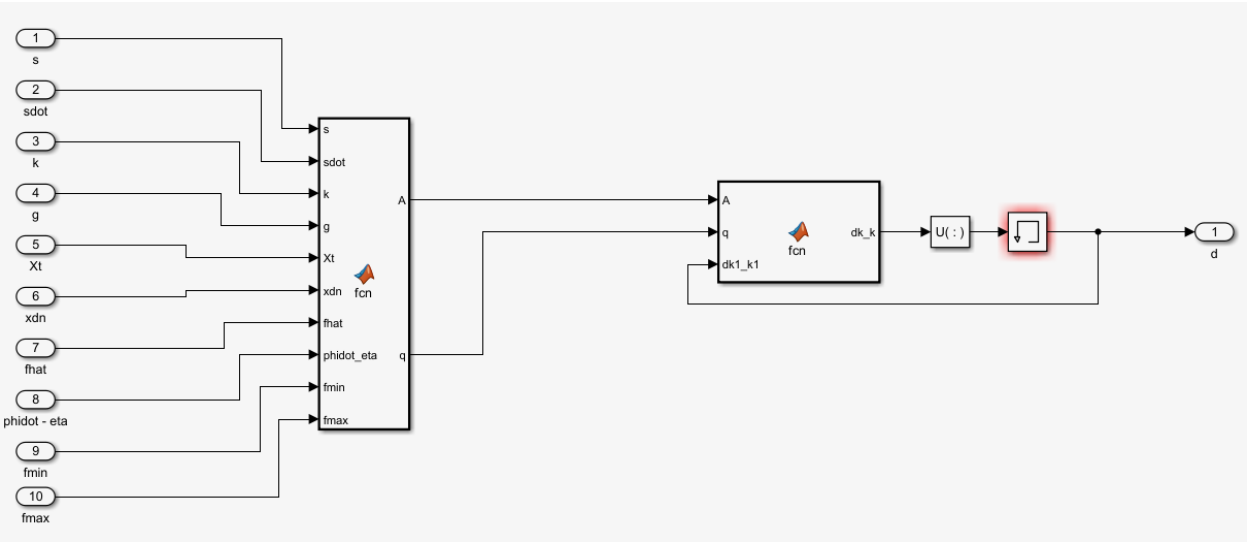


Figure 36. Coupled boundary estimator.

A.3 Random Invertible Continuous Matrix Generation

When testing and validating a robust control system, it may be helpful to generate random parameters. Successful control of these random systems acts as more proof that the controller works.

A.3.1 Initial Approach: Determinant Adjustment

One method for ensuring invertibility is to adjust one of the matrix values so that the minimum determinant over all time is greater than zero. Given that a two-by-two matrix's determinant is:

$$det\left(\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\right) = a_{11}a_{22} - a_{12}a_{21} \tag{48}$$

If $\frac{det(\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}) - m_d}{a_{22}}$ is added to a_{11}, the minimum determinant will become m_d. The technique was used on matrices created by sine waves with random frequencies and offsets early on in this project's testing. The method is simple and can easily be used to produce matrices with positive elements (which was a requirement for this project). However, it would produce matrices with significantly larger first elements than any of the others. Ultimately, determinant adjustment was replaced by the other techniques in this section.

A.3.2 Invertible Matrix Products and Row Swaps

A property of invertible matrices is that their products are invertible. Take, for example, an invertible matrix generated using sine waves. To randomize this time varying matrix, it may be multiplied by a random, constant, invertible matrix. The original implementation repeatedly performed this multiplication then swapped two rows. The row swap has the effect of randomizing the sign of the determinant. While this works, a simpler, equally effective method would be to perform one multiplication and one swap.

The method has the same positive properties as determinant adjustment but keeps all of the elements on the same order of magnitude. That being said, the technique relies on a preexisting invertible, time varying matrix. This dependence constrains the randomness of the matrix. For example, if the initial matrix is sinusoidal, then the resulting matrix will still be sinusoidal. In many applications, the technique will be sufficient, even with this drawback.

A.3.3 Linear Independence of Rotated Vectors

If more variability is desired, matrices may be constructed using vectors rotated with different angles. These vectors would be linearly independent. An initial angle vector may be generated at each time step. The first column may be created by rotating a constant vector by those angles. Subsequent angle vectors may be generated and mapped to values sufficiently outside the previous vectors, and the remaining columns may be created using the same constant vector.

Like the previous methods, it is easy to create time varying matrices with positive elements by limiting the angles to a range that rotates the constant vector to a vector with positive elements. Since the angles are randomly chosen without accounting for previous values, the generated matrix changes with a very high frequency. To reduce the frequency, the matrix may be generated with a lower sampling rate and interpolated.

A.3.4 Jacobi's Formula and the Matrix Determinant Lemma

The final method for generating invertible, time varying matrices explored here is based on Jacobi's formula for the derivative of the determinant:

$$\frac{d}{dt}det(A) = tr(adj(A)\frac{d}{dt}A) = det(A)tr(A^{-1}\frac{d}{dt}A) \text{ [23]} \tag{49}$$

This formula may be vectorized to produce:

$$\frac{d}{dt}det(A) = det(A)vec^T(I)vec(A^{-1}\frac{d}{dt}A) = det(A)vec^T(I)[I \otimes A^{-1}]vec(\frac{d}{dt}A) \tag{50}$$

If a system is created such that this equation is upheld, the determinant may be constrained between a set of values. One such system is:

$$\dot{\vec{b}} = \alpha\Gamma^+[\zeta - det(B)] + \beta\Gamma_\perp{\Gamma_\perp}^+\left[vec(B_{ref}) - \vec{b}\right] \tag{51}$$

$$\Gamma = det(B)vec^T(I)[I \otimes B^{-1}] \tag{52}$$

$$\zeta = sat\left(det(B_{ref}), |B|_l, |B|_h\right) \tag{53}$$

where $\Gamma_\perp$ is a matrix made from the set of vectors in the null space of Γ, α and β are parameters to be chosen by the user, and $|B|_l$ and $|B|_h$ are the minimum and maximum desired values of $det(B)$, respectively. B_{ref} can be any time varying matrix, including a matrix that is not always invertible.

An important consideration is the initial conditions for the system. $B_{ref}(t = 0)$ could be chosen, but there is no guarantee that its determinant will be within the desired bounds. A known matrix with the correct determinant may be used. If more randomness is desired, the system could be simulated for extra time and the times when the determinant was out of bounds could be truncated.

Another option – and the one chosen in this work – was to adjust the known matrix with the matrix determinant lemma:

$$det(A + \vec{u}\vec{v}^T) = [1 + \vec{v}^T A^{-1}\vec{u}]det(A) \text{ [24]} \tag{54}$$

If u is chosen at random and A is the known matrix, v may be solved for. The new initial matrix is the sum in the left determinant. The matrix determinant lemma technique has one other

benefit: if the unadjusted matrix is diagonal, its determinant is less than the desired range, and both the matrix and u have positive elements, then there exists a $\vec{v}$ such that all of the initial matrix's elements will be positive.

A.4 Block Controllable Canonical Transformations

One of the proposed objectives for this work was to control a linearized aircraft system as presented in [5]. While testing the controller with this system, there were two main issues. First, preliminary results were unfavorable. Second (and presumably the cause of the first), the system was actually non-square. A method for transforming the system into an equivalent square system was developed but work with the system was abandoned in favor of the other proposed goals. The transformation method is given in this section.

Many linear systems are equivalent to square systems through state transformations. While it is easy to transform a linear n^{th}-order square system to a non-square system (just change it to state space form), the reverse is not always trivial. Take, for example, the aircraft system given in [5]:

$$A = \begin{bmatrix} -0.2316 & 0.0633 & -0.9956 & 0.051 \\ -29.4924 & -3.0169 & 0.0201 & 0.0 \\ 6.2346 & -0.0274 & -0.4169 & 0.0 \\ 0.0 & 1.0 & 0.0631 & 0.0 \end{bmatrix} \qquad B = \begin{bmatrix} 0.0052 & 0.031 \\ -36.4909 & 8.109 \\ -0.4916 & -2.8274 \\ 0.0 & 0.0 \end{bmatrix}$$

This system has four states and two controls. If the system was converted to an equivalent, square system, it would be in a block controllable canonical form:

$$\vec{z} = T\vec{x} \tag{55}$$

$$\overline{A} = \begin{bmatrix} 0 & I_{p-m} \\ \overline{A}_{21} & \overline{A}_{22} \end{bmatrix} = TAT^{-1} \tag{56}$$

$$\overline{B} = \begin{bmatrix} 0 \\ \overline{B}_2 \end{bmatrix} = TB \tag{57}$$

where p and m are the number of states and the number of control inputs, respectively. Since $TA = \overline{A}T$, the following relations may be found:

$$[I_{p-m} \quad 0]TA = [0 \quad I_{p-m}]T \tag{58}$$

$$[I_{p-m} \quad 0]TB = 0 \tag{59}$$

With these equations, a solution for T may be computed. There are many ways to solve for T. In this work, the system:

$$vec(\dot{T}) = -Q^{+}Qvec(T) \tag{60}$$

$$Q = \begin{bmatrix} A^T \otimes [I_{p-m} \quad 0] - I_p \otimes [0 \quad I_{p-m}] \\ B^T \otimes [I_{p-m} \quad 0] \end{bmatrix} \tag{61}$$

$$T(0) = \begin{bmatrix} 0 & I_{p-m} \\ A_{21} & A_{22} \end{bmatrix} \tag{62}$$

was integrated. T, $\overline{A}$, and $\overline{B}$ were found to be:

$$T = \begin{bmatrix} -0.2907 & 0.0 & -0.0032 & -0.0268 \\ 0.0 & 0.0 & 0.0 & 1.0 \\ 0.0474 & -0.0451 & 0.2891 & -0.0148 \\ 0.0 & 1.0 & 0.0631 & 0.0 \end{bmatrix}$$

$$\overline{A} = \begin{bmatrix} 0.0 & 0.0 & 1.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 1.0 \\ -10.8585 & -0.2995 & -0.7245 & 0.0837 \\ 100.378 & 2.7171 & 1.7258 & -2.9409 \end{bmatrix}$$

$$\overline{B} = \begin{bmatrix} 0.0 & 0.0 \\ 0.0 & 0.0 \\ 1.5051 & -1.1818 \\ -36.5219 & 7.9306 \end{bmatrix}$$

This transformation matrix is invertible, which is required. While there is no proof that this method will make an invertible T, it is most likely that this is the case. T's Invertibility should be checked before use.