2-1-2004

An Adaptive estimation scheme for reducing communications in a distributed control implementation

Aaron Burry

Follow this and additional works at: http://scholarworks.rit.edu/theses

Recommended Citation
AN ADAPTIVE ESTIMATION SCHEME FOR REDUCING COMMUNICATIONS IN A DISTRIBUTED CONTROL IMPLEMENTATION

By

AARON M. BURRY

A Thesis Submitted to the Faculty of the Rochester Institute of Technology in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

IN

ELECTRICAL ENGINEERING

Approved By

Thesis Advisor  Dr. Mark A. Hopkins

Thesis Committee  Dr. Sohail A. Dianat

Thesis Committee  Dr. Ferat Sahin

Department Head  Dr. Robert J. Bowman

DEPARTMENT OF ELECTRICAL ENGINEERING
COLLEGE OF ENGINEERING
ROCHESTER INSTITUTE OF TECHNOLOGY
ROCHESTER, NEW YORK
FEBRUARY 2004
I, Aaron M. Burry, hereby grant permission to the Wallace Library of the Rochester Institute of Technology to reproduce my thesis in whole or in part. Any reproduction will not be for commercial use or profit.
Abstract

This paper examines the application of adaptive estimation and control techniques to reduce the amount of communication required between subsystems in a distributed control implementation. Rather than require a large amount of communications to broadcast the outputs or the states of each of the subsystem nodes to all of the other nodes at every sampling instant, local estimators in each subsystem are used to predict the state vectors for all of the other subsystems. These estimates are then used in the calculation of the controller outputs for each of the subsystems. Prior work in the literature has focused on static estimation schemes to achieve such reductions in communications. However, such schemes typically require very accurate models of the plant in order to maintain the desired reduction in communications. Poorly modeled dynamics or systems whose dynamics change slowly over time (due to aging of components, changes in plant parameters such as a robot picking up a heavy object, etc.) can cause a substantial increase in the amount of communications required to maintain the desired system performance. In order to avoid this, this paper presents an adaptive estimation and control scheme for each subsystem in the distributed implementation. The stability of the state estimators and the convergence of the state tracking errors to within a desired threshold is proven. The performance of the system using perfect communication at every sampling instant, using a static estimation scheme, and using the proposed adaptive estimation scheme are then compared in simulation.
# Table of Contents

Abstract i

Table of Contents ii

List of Figures iv

1 Introduction and Related Work 1
   1.1 Background 1
   1.2 Previous Work 5
   1.3 Overview 8

2 Problem Statement and Framework 9
   2.1 System Analysis 9
   2.2 Problem Statement 18
   2.3 Distributed Control Method 18
   2.4 Adaptive Estimation Method 20
      2.4.1 Estimator Form For $x \in \Omega_2$ 21
      2.4.2 Estimator Form For $x \in \Omega_1$ 24

3 Controller Design and Stability Analysis 25
   3.1 No Adaptation 25
   3.2 During Adaptation 26
      3.2.1 Adaptive Controller 26
      3.2.2 Adaptive State Estimator 33

4 Performance Metrics 36
   4.1 Communication Metric 36
   4.2 Performance Metric 37
5 Simulation Results

5.1 Spring-Mass-Damper System ........................................... 39
5.2 Nominal Plant Results .................................................... 41
5.3 Effect Of Threshold Values $E_0^i$ ...................................... 66
5.4 Comparison Of Results For Static and Adaptive Estimators .... 66

6 Conclusions ........................................................................ 78

List of References .................................................................. 79

A Software Source Code ......................................................... 80
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Typical Distributed Controller Implementation</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Totally Decentralized Controller Implementation</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Block Diagram of the State Estimation Scheme Proposed in [1]</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>Block Diagram of the ith Subsystem In The Partitioned Plant Model</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>Partitioned Controller Approach</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>Block Diagram Of Plant With Communication</td>
<td>20</td>
</tr>
<tr>
<td>7</td>
<td>Block Diagram Of Plant Without Communication</td>
<td>21</td>
</tr>
<tr>
<td>8</td>
<td>Trajectory Of the System State Vector</td>
<td>32</td>
</tr>
<tr>
<td>9</td>
<td>Spring-Mass-Damper Plant</td>
<td>38</td>
</tr>
<tr>
<td>10</td>
<td>State Tracking Performance For Constant Communication and Adaptation: States $X_1$ and $X_2$</td>
<td>43</td>
</tr>
<tr>
<td>11</td>
<td>State Tracking Performance For Constant Communication and Adaptation: States $X_3$ and $X_4$</td>
<td>44</td>
</tr>
<tr>
<td>12</td>
<td>State Tracking Performance For Constant Communication and Adaptation: States $X_5$ and $X_6$</td>
<td>45</td>
</tr>
<tr>
<td>13</td>
<td>State Estimator Performance For Constant Communication and Adaptation: States $X_1$ and $X_2$</td>
<td>46</td>
</tr>
<tr>
<td>14</td>
<td>State Estimator Performance For Constant Communication and Adaptation: States $X_3$ and $X_4$</td>
<td>47</td>
</tr>
<tr>
<td>15</td>
<td>State Estimator Performance For Constant Communication and Adaptation: States $X_5$ and $X_6$</td>
<td>48</td>
</tr>
<tr>
<td>16</td>
<td>State Tracking Error History For Constant Communication and Adaptation</td>
<td>50</td>
</tr>
<tr>
<td>17</td>
<td>State Estimator Error History For Each System State Continuous Adaptation</td>
<td>51</td>
</tr>
<tr>
<td>18</td>
<td>Adaptive History of Controller Parameters $\hat{K}<em>1$ and $\hat{\psi}</em>{ij}$ Continuous Adaptation</td>
<td>52</td>
</tr>
<tr>
<td>Page</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Adaptive History of State Estimator Parameters $\hat{A}<em>{ij}$ and $\hat{B}</em>{jk}$ Continuous Adaptation</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>State Tracking Performance For Bounded Error Adaptation: States $X_1$ and $X_2$</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>State Tracking Performance For Bounded Error Adaptation: States $X_3$ and $X_4$</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>State Tracking Performance For Bounded Error Adaptation: States $X_5$ and $X_6$</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>State Estimator Performance For Bounded Error Adaptation: States $X_1$ and $X_2$</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>State Estimator Performance For Bounded Error Adaptation: States $X_3$ and $X_4$</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>State Estimator Performance For Bounded Error Adaptation: States $X_5$ and $X_6$</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>State Tracking Error History For Each System State Bounded Error Adaptation</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>State Estimator Error History For Each System State - Bounded Error Adaptation</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>Communications History For Bounded Error Adaptation</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>Effect of Changing the Error Threshold Parameter $E_\theta$ On System Tracking Performance and Required Inter-Subsystem Communication</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>Estimator Comparison Experiment</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>Comparison of Tracking Error Performances For State $X_1$</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>Comparison of Tracking Error Performances For State $X_2$</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>Comparison of Tracking Error Performances For State $X_3$</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>Comparison of Tracking Error Performances For State $X_4$</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>Comparison of Tracking Error Performances For State $X_5$</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>Comparison of Tracking Error Performances For State $X_6$</td>
<td></td>
</tr>
</tbody>
</table>
1 Introduction and Related Work

1.1 Background

When dealing with highly complex problems, a well known practice is to partition the problem into smaller, more manageable pieces. Each of these pieces can then be analyzed somewhat independently. The solution to these individual problems can then be integrated together such that a solution to the overall problem can be reached. There are many other reasons, other than complexity, for partitioning a problem into smaller sub-problems for analysis. Resource allocation, for instance, can dictate the partitioning of a problem into sub-problems which are then managed by separate teams. In this case, it is the desire to have multiple teams analyze portions of the problem in parallel that drives the partitioning.

This type of methodology is also practiced in the field of control systems. The plant to be controlled is partitioned into subsystems and then separate controllers are designed and implemented for each of these subsystems. This strategy is referred to as distributed control. In centralized controller implementations, all of the required signals (feedback from output sensors, state feedback signals, and reference inputs) are routed to a single location. The controller is designed based on full knowledge of the required signals and implemented at the central location where these signals have been routed. In contrast, a distributed control implementation places the decision-making and computation of the controller into the subsystems themselves. In addition, in a distributed control implementation the local subsystem controllers typically do not have access to all of the outputs nor all of the states of the other subsystems at each sampling instant. This means that the subsystem controllers must base their decision making and control actions on a more limited set of information than in a centralized implementation. The amount of non-local information required by each subsystem controller is determined by a variety of factors, including: the dynamics of the plant, the chosen partitioning of the plant into subsystems, and the design of the subsystem controllers themselves. Determining what amount of communication is required to
achieve the desired system performance is part of the design of a distributed control system. The amount of communication at each sampling instant that is supported by the implementation can vary from full state or output communication (essentially centralized control implemented over a communications network) to no communication at all (totally decentralized control).

A block diagram of a typical distributed controller implementation is presented in Figure 1. In this figure, the plant has been partitioned into \( n \) subsystems where the \( \Pi_i \) are the subsystem controllers, the \( x_i \) the local subsystem state vectors, and the \( r_i \) are the reference inputs to each of the subsystems. In this scheme, a communications bus (the control network) has been provided to enable the exchange of information between subsystems. The required amount of communication at each sampling instant, or perhaps the maximum amount if the rate of exchange varies, would then dictate the required bandwidth for this communications channel. Figure 2 illustrates an extreme example where the subsystem controllers only have access to the local state vector and reference input, providing no mechanism for information exchange between subsystems. This type of configuration is known as totally decentralized control.

Distributed control can offer a variety of advantages over a centralized scheme. These include [2]: taking advantage of the distribution inherent in some plants; increased processing capability due to the increase in the number of compute engines (typically one per subsystem); flexibility to changes in the plant; improved tolerance to failures in parts of the system; enabling controllers to be delivered as part of their associated subsystems; and possible advantages in cost due to reduced wiring.

When designing a distributed control system, it is obviously necessary to guarantee the overall stability of the system and that the desired performance objectives are achieved. One factor that can strongly influence both the stability and performance of the system is the amount of physical coupling between subsystems in the plant. If not properly accounted for in the design of the subsystem controllers, these coupling
interactions can lead to significant performance degradations, instability in the system, or even system failures. The reason for this is that each distributed subsystem controller is acting upon a reduced set of information when calculating its control output to the plant (an extreme example of this can be seen in Figure 2 where there is only feedback of the local state information). Strong coupling in the plant between subsystems means that non-local behavior in the system is impacting the evolution of the local subsystem states. In essence, these unknown coupling interactions are acting as disturbances to the local subsystems. If the design of the local controllers is not sufficiently robust to these disturbances, the coupling interactions can lead to instability of the system. Thus, the design of the subsystem controllers and the communications mechanism must somehow account for these coupling interactions in order to guarantee both desired performance and system stability.

The robustness of the communications channel in a distributed control implementation can also impact both system stability and performance. Message packet delays, lost messages, and other communications related problems can impact the reliability of information reaching the subsystem controllers. Presenting the controllers
with faulty and/or out-of-date information can result in inappropriate control actions. These undesirable controller outputs can then lead to poor performance or even instability of the system. In many cases, requiring the exchange of large amounts of information between subsystems at each sampling instant can contribute to these types of problems. Unexpected message packet delays in particular can be caused by over-burdening of the communications network.

So it is important to ensure that the communication network is capable of supporting the required level of information exchange between subsystems. However, the higher the bandwidth of the communications channel, the more costly the implementation will usually become. This means that there is typically a strong desire to minimize the required amount of communications between subsystems while still achieving acceptable system performance. A further complication is that the amount of information that must be exchanged between subsystems can vary from sampling instant to sampling instant based on a variety of factors. Thus, it is typically difficult to predict in advance what the required rate of information exchange will be.
1.2 Previous Work

A variety of methods have been presented in the literature for guaranteeing system stability and performance while still requiring a limited communications bandwidth between subsystems in a distributed control implementation. These methods can be separated into two basic types of approaches. The first type of approach involves developing analysis and implementation techniques to minimize the possibility of system failures due to communication issues. The goal of this type of approach is to enable the implementation of a lower bandwidth but more reliable communications channel between subsystems. One such approach is outlined in [3] where the authors have focused on analyzing the real-time communications requirements and constraints in an effort to better predict system reliability. The goal is to predict in advance the capability of the network to support the information exchanges required and the probability of potential failures.

In [2] the controller is designed assuming a centralized implementation. The actual distributed implementation then requires information exchange between subsystems across a network. The authors propose a mathematical framework for analyzing the performance degradation of the controllers due to communication delays in the network.

An analysis technique for defining an upper limit on the allowable communication interval in a networked control system while still guaranteeing stability is presented in [4]. The authors assume that the distributed nodes must communicate their outputs to one another. An extension of this result is presented in [5] where a prediction method is used to predict the output of each subsystem between consecutive transmissions of the actual outputs. This allows for an extension of the maximum allowable delay between output transmissions while still guaranteeing system stability.

Other authors have taken a slightly different approach attempting to limit the amount of communications that are required between subsystems as part of the distributed controller design. In many cases, these approaches have centered upon not
allowing any information exchange between subsystems at all (totally decentralized control). Many authors have developed techniques for designing subsystem controllers that are robust to the interactions between subsystems. Approaches such as [6], [7], [8], [9], and [10] all involve designing distributed controllers that do not require any information exchange between subsystem nodes while still achieving acceptable system performance. These approaches make use of an adaptive process to make the localized controllers robust to the coupling between subsystems. In each case, there are restrictions placed on the plant dynamics, the size of the interactions, or both in order to guarantee stability.

In [1] an approach is outlined for minimizing (rather than eliminating) the communications between subsystem nodes in a distributed control implementation. This is accomplished by using state estimation at each node in the system to predict the state vectors of the other subsystems. In this way, the state information for each node does not have to be communicated to all other nodes at each sampling instant. Instead, each controller uses estimated values of the state vectors for the other nodes in the system to update its control vector at each sampling instant. Figure 3 shows a block diagram of this approach. In this figure, the region enclosed by the dotted square is an internal view of subsystem 1. Each of the other subsystems would have a similar structure. So each subsystem contains an implementation of the full system controller and a full system estimator. The estimated outputs are used in conjunction with the locally measurable outputs to determine the local controller output to the plant. In order to guarantee acceptable tracking using this approach, the authors enforce an upper bound on the state estimation error for each node's state vector. When the state estimation error for the $i$th node is larger than this pre-established bound, the locally measured system outputs for this node are communicated across the bus to all the other nodes in the system. The other nodes can then use the true output vector for the $i$th node to update the states of their local estimators. The authors demonstrate that by adjusting the acceptable estimation error threshold, the
Figure 3: Block Diagram of the State Estimation Scheme Proposed in [1]

performance of the system to be traded off against the amount of communications traffic required between subsystems.

The success with which this type of approach can reduce the required rate of communications between subsystems is highly dependent upon the accuracy of the plant model used in the design of the state estimator. Along these lines, an issue that is not addressed by this type of approach is the effect of slowly changing plant dynamics. Take the case of a plant whose dynamics are affected by frictional wear or other age related effects to its mechanical elements. Using the static state estimation scheme of [1], the accuracy with which the original state estimator represents the actual plant would slowly decay over time as the plant dynamics evolve. Without some form of re-tuning of the state estimator being used, this degradation in the performance of the state estimator would result in a growth in the amount of communications required between subsystems over time. In extreme cases, this could result in communications between subsystems at each sampling instant.
1.3 Overview

The objective of this work was to develop an adaptive control scheme that could be used to reduce network message traffic in a distributed control implementation while still achieving acceptable state tracking performance and guaranteeing overall system stability. An important goal of this work was to demonstrate improved performance of this adaptive method over a static estimation and control scheme for the case of poorly modeled plant dynamics.

The remainder of this paper is organized as follows: The framework of the problem and an analysis of the system dynamics is presented in Section 2; The stability of the overall system as well as the convergence of the tracking error during adaptation is proven in Section 3; Simulation results demonstrating the performance of the proposed scheme for a sample plant are presented in Section 5; Simulation results demonstrating the improved performance of the adaptive scheme over a static state estimation scheme are also presented in Section 5.
2 Problem Statement and Framework

2.1 System Analysis

Consider an M input, M output plant described by the following equations:

\[ \dot{x}(t) = Ax(t) + Bu(t) + Br(t) \]  \hspace{1cm} (2.1)
\[ y(t) = Cx(t) \]  \hspace{1cm} (2.2)

where \( x(t) \in \mathbb{R}^N \) is the state vector of the plant, \( u(t) \in \mathbb{R}^M \) is the control vector, \( r(t) \in \mathbb{R}^M \) is the vector of reference inputs to the system, and \( y(t) \in \mathbb{R}^M \) is the output vector. The matrices A, B, C are, for the moment, assumed to be constant and of the following sizes:

\[ A : N \times N, \]
\[ B : N \times M, \]
\[ C : M \times N \]

In this investigation it is assumed that the B and C matrices are known, but that the A matrix is unknown.

Now, the plant is partitioned into \( M \) coupled subsystems where each subsystem is single-input. After the partitioning, each subsystem has \( N_p \) states where \( N = MN_p \). It is assumed that the system has been instrumented such that all of the states of the local subsystems are measurable. Furthermore, for the purposes of this investigation, only those plants whose input and output matrices are in the following form are considered:

\[ B = blkdiag(B_1, B_2, \ldots, B_M) \]  \hspace{1cm} (2.3)
\[ C = blkdiag(C_1, C_2, \ldots, C_M) \]  \hspace{1cm} (2.4)
where the $B_i$ are vectors of size $(N_p \times 1)$ and the $C_i$ are vectors of size $(1 \times N_p)$.

A system composed of interconnected masses, such as the spring-mass-damper system presented in Section 5.1, is an example of a system that satisfies these requirements. In this example, the system is partitioned such that the subsystem state equations describe the dynamics of each of the individual masses. There is a separate force input applied to each of the masses that does not directly affect the motion of the other masses in the system - the applied force only directly affects the acceleration of the mass while the coupling between the masses occurs through the positions and velocities. Because the subsystem interactions occur through states of the system that are not directly affected by the subsystem inputs, the input $B$ matrix will be of the desired block diagonal form.

Assuming that the input matrix is block diagonal, the state dynamics of (2.1) can be rewritten as follows:

$$\begin{align*}
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2 \\
\vdots \\
\dot{x}_M 
\end{bmatrix} &= \begin{bmatrix}
A_{11} & A_{12} & \cdots & A_{1M} \\
A_{21} & A_{22} & \cdots & A_{2M} \\
\vdots & \vdots & \ddots & \vdots \\
A_{M1} & A_{M2} & \cdots & A_{MM}
\end{bmatrix} \begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_M
\end{bmatrix} + 
\begin{bmatrix}
B_1 & 0 & \cdots & 0 \\
0 & B_2 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & B_M
\end{bmatrix} \begin{bmatrix}
u_1 \\
u_2 \\
\vdots \\
u_M
\end{bmatrix} + 
\begin{bmatrix}
B_1 & 0 & \cdots & 0 \\
0 & B_2 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & B_M
\end{bmatrix} \begin{bmatrix}
r_1 \\
r_2 \\
\vdots \\
r_M
\end{bmatrix}
\end{align*}$$

(2.5)

where $A_{ii}$ is the $N_p \times N_p$ matrix representing the local state dynamics of the $i$th subsystem, $A_{ij}$ is the $N_p \times N_p$ matrix representing the coupling dynamics between the states of the $i$th and $j$th subsystems, $x_i$ is the $N_p \times 1$ state vector of the $i$th subsystem, $B_i$ is the $N_p \times 1$ input matrix of the $i$th subsystem, $r_i$ is the $1 \times 1$ reference input to the $i$th subsystem, and $u_i$ is the $1 \times 1$ control input for the $i$th subsystem.
Factoring the interaction terms from the $A$ matrix in (2.5), the following is obtained:

$$
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2 \\
\vdots \\
\dot{x}_M \\
\end{bmatrix} =
\begin{bmatrix}
A_{11} & 0 & \cdots & 0 \\
0 & A_{22} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & A_{MM} \\
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_M \\
\end{bmatrix}
+ \begin{bmatrix}
B_1 & 0 & \cdots & 0 \\
0 & B_2 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & B_M \\
\end{bmatrix}
\begin{bmatrix}
u_1 \\
u_2 \\
\vdots \\
u_M \\
\end{bmatrix}
$$

This expression can be simplified by writing,

$$
\begin{bmatrix}
\dot{x}_1 \\
\dot{x}_2 \\
\vdots \\
\dot{x}_M \\
\end{bmatrix} =
\begin{bmatrix}
A_{11} & 0 & \cdots & 0 \\
0 & A_{22} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & A_{MM} \\
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_M \\
\end{bmatrix}
+ \begin{bmatrix}
B_1 & 0 & \cdots & 0 \\
0 & B_2 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & B_M \\
\end{bmatrix}
\begin{bmatrix}
u_1 + r_1 \\
u_2 + r_2 \\
\vdots \\
u_M + r_M \\
\end{bmatrix}
$$

where,

$$
\phi_1 = \sum_{j=2}^{M} A_{1j} x_j \\
\phi_2 = \sum_{j=1, j \neq 2}^{M} A_{2j} x_j \\
\vdots \\
\phi_i = \sum_{j=1, j \neq i}^{M} A_{ij} x_j 
$$

Here the $\phi_i$ matrices represent the contribution to the state dynamics of the $i$th subsystem due to the other $(M-1)$ subsystems. The dynamics of the $i$th subsystem can now be written:

$$
\dot{x}_i(t) = A_i x_i(t) + B_i u_i(t) + B_i r_i(t) + \phi_i \\
y_i(t) = C_i x_i(t) 
$$
where, \( i = 1, 2, \ldots, M \). \( x_i(t) \in \mathbb{R}^{N_i} \) is the state vector of the \( i \)th subsystem, \( A_i = A_{ii} \), \( u_i(t) \in \mathbb{R}^1 \) is the controller output for the \( i \)th subsystem, \( r_i(t) \in \mathbb{R}^1 \) is the reference input for the \( i \)th subsystem, and \( y_i(t) \in \mathbb{R}^1 \) is the output of the \( i \)th subsystem. A block diagram of this subsystem model is shown in Figure 4.

\[
\begin{align*}
\dot{x}_m(t) &= A_m x_m(t) + B r(t) \\
\text{(2.10)}
\end{align*}
\]

where \( x_m \in \mathbb{R}^N \) and \( A_m \) is \((N \times N)\). The dynamics of the reference model states are

Figure 4: Block Diagram of the \( i \)th Subsystem In The Partitioned Plant Model

The objective is to design a controller \( \Pi \) that will force the states of (2.1) to track those of the following reference model:
assumed to be completely decoupled such that,

\[
A_m = \begin{bmatrix}
A_{m1} & 0 & \cdots & 0 \\
0 & A_{m2} & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
0 & 0 & \cdots & A_{mM}
\end{bmatrix}
\] (2.11)

If the reference model in (2.10) is then partitioned into \( M \) subsystems, the dynamics of the \( i \)th subsystem of the reference model can be written as,

\[
\dot{x}_{mi}(t) = A_{mi}x_{mi}(t) + B_ir_i(t)
\] (2.12)

The objective is then to design the subsystem controllers (\( \Pi_i \)) that will force the subsystem states \( x_i \) to track the reference model states \( x_{mi} \). A diagram of the proposed system partitioning is shown in Figure 5. In this diagram, it is assumed that a communications bus is available to enable some amount of information exchange between subsystems. Part of the design of the subsystem controllers \( \Pi_i \) is determining how much information will need to be communicated across this bus.

![Diagram of the proposed system partitioning](image)

*Figure 5: Partitioned Controller Approach*
In order to ensure that the controller for the $i$th subsystem can compensate for the interaction terms $(\phi_i)$ in (2.7), a restriction of the form posed in [10] will be adhered to. This restriction requires that the interconnection terms be representable in the following form:

$$\phi_i = \sum_{j=1,j\neq i}^{M} B_i \psi_{ij}^T x_j$$

(2.13)

where the $N\times1$ vector $\psi_{ij}$ represents the unknown linear coupling between subsystems $i$ and $j$. A comparison of this equation with (2.8) shows that this restriction equates to the following,

$$A_{ij} = B_i \psi_{ij}^T$$

(2.14)

The nature of this restriction can be better understood in the context of (2.6). From this equation, the dynamics of the $i$th subsystem are,

$$\dot{x}_i = B_i r_i + B_i u_i + A_{ii} x_i + [A_{i1} x_1 + A_{i2} x_2 + \ldots + A_{iN} x_N]$$

(2.15)

where the term $A_{ii} x_i$ has been intentionally moved outside the brackets. The restriction in (2.14) requires that the equation above can be written as follows:

$$\dot{x}_i = B_i r_i + B_i u_i + A_{ii} x_i + B_i \left[ \psi_{i1}^T x_1 + \psi_{i2}^T x_2 + \ldots + \psi_{iN}^T x_N \right]$$

(2.16)

A comparison of (2.15) and (2.16) shows that imposing this constraint amounts to reducing the complexity of the interaction terms. Specifically, the contribution from the $j$th subsystem to the dynamics of any of the other subsystems is simply a weighted linear combination of the elements of state vector $x_j$. The extent to which this scalar quantity $(\psi_{ij}^T x_j)$ affects the evolution of each of the states in subsystem state vector $x_i$ is determined by the local input vector $B_i$. Thus, the contributions to each element of $x_i$ from $x_j$ are not linearly independent as they are for (2.15). This is the reduction in complexity of the interaction terms that was referred to above. An example system that satisfies this constraint will be presented in Section 5.1.
The reason for enforcing the restriction in (2.14) is to ensure that the controller for the \( i \)th subsystem is capable of fully compensating for the contributions from the other \((M-1)\) subsystems. The goal would be to have a term in the controller act to completely decouple the \( i \)th subsystem from the dynamics of all of the other subsystems. Substituting (2.13) into (2.7), the dynamics of the \( i \)th subsystem can be represented in the following form:

\[
\dot{x}_i(t) = A_i x_i(t) + B_i u_i(t) + B_i r_i(t) + B_i \sum_{j=1,j\neq i}^{M} \psi_{ij}^T x_j \\
y_i(t) = C_i x_i(t)
\]  

where \( i = 1, 2, \ldots, M \). From this equation it is clear that, if the \( i \)th controller has perfect knowledge of the states \( x_j \) of the other subsystems, then a controller term of the following form,

\[
u_i = - \sum_{j=1,j\neq i}^{M} \hat{\psi}_{ij} x_j
\]

allows the controller to completely decouple the state dynamics of the \( i \)th node from those of the other subsystems when \( \hat{\psi}_{ij} = \psi_{ij} \). Other terms in the controller could then be utilized to ensure that the subsystem states \( x_i \) track those of the desired reference model \( x_{mi} \).

Using a controller form similar to that presented in [6], the control vector of the \( i \)th subsystem \((u_i)\) is designed as follows:

\[
u_i = \hat{K}_i^T x_i - \sum_{j=1,j\neq i}^{M} \hat{\psi}_{ij}^T x_j
\]

where \( \hat{\psi}_{ij} \in \mathbb{R}^{N_p} \) is the estimation of the unknown coupling between the \( i \)th and \( j \)th subsystems and \( \hat{K}_i \) is the local state feedback controller gain vector. The second term in this controller equation acts to decouple the dynamics of the \( i \)th subsystem from the rest of the system. The state feedback term \( \hat{K}_i^T x_i \) then serves to adjust the local plant dynamics to match those of the desired reference model dynamics \( A_{mi} \).
Substitution of (2.19) into (2.17) gives the following model for the \(i\)th subsystem:

\[
\begin{align*}
\dot{x}_i(t) &= A_i x_i(t) + B_i [\hat{K}_i^T x_i(t) - \sum_{j=1,j\neq i}^{M} \hat{\psi}_{ij}^T x_j] + B_i r_i(t) + B_i \sum_{j=1,j\neq i}^{M} \psi_{ij}^T x_j \\
y_i(t) &= C_i x_i(t)
\end{align*}
\tag{2.20}
\]

In order to guarantee that a controller of the form posed in (2.19) can successfully achieve the desired tracking objective, another important restriction must be imposed. This restriction is that, for some unknown controller gain vector \(K_i^*\), the following relation must hold:

\[
A_{mi} = A_i + B_i K_i^T
\tag{2.21}
\]

Rearranging this equation slightly will help with an intuitive understanding of this restriction.

\[
A_{mi} - A_i = B_i K_i^T
\tag{2.22}
\]

Here it is clearly seen that the imposed constraint places a restriction on the difference between the desired system dynamics (those of the reference model \(A_{mi}\)) and the nominal plant dynamics \((A_i)\). Once again, this will ensure that the proposed controller will be capable of achieving the desired tracking objective.

A more fundamental understanding of the reason for imposing this constraint can easily be obtained by close inspection of (2.20). If the estimates of the interaction terms perfectly match the actual coupling dynamics then,

\[
\hat{\psi}_{ij} = \psi_{ij}
\tag{2.23}
\]

and (2.20) can be reduced to the following:

\[
\begin{align*}
\dot{x}_i(t) &= [A_i + B_i \hat{K}_i^T] x_i(t) + B_i r_i(t) \\
y_i(t) &= C_i x_i(t)
\end{align*}
\tag{2.24}
\]
A comparison of this equation with (2.12) clearly shows that perfect tracking of the reference model states is guaranteed if (2.21) holds and \( \hat{K}_i = K_i^* \). Since this desired feedback gain vector is assumed to be unknown, the problem then becomes designing the appropriate local feedback gain vectors \( \hat{K}_i \) that will ensure tracking of the reference model states.

A closer inspection of the restriction imposed by (2.21) is now in order. A sufficient condition for this restriction to be satisfied is that both the reference model dynamics and the subsystem dynamics be expressible in the following companion form (the idea of expressing the subsystems of the reference model and the plant in this form is borrowed from Gavel and Siljak in [10]):

\[
A_{mi} = \begin{bmatrix}
0 & 1 & 0 & \ldots & 0 \\
0 & 0 & 1 & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
-\alpha_i^{(k-1)} & -\alpha_i^{(k-2)} & -\alpha_i^{(k-3)} & \ldots & -\alpha_i^i
\end{bmatrix}
\]

\[
A_i = \begin{bmatrix}
0 & 1 & 0 & \ldots & 0 \\
0 & 0 & 1 & \ldots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
-\beta_i^{(k-1)} & -\beta_i^{(k-2)} & -\beta_i^{(k-3)} & \ldots & -\beta_i^i
\end{bmatrix}
\]

\[
B_i = B_{mi} = \begin{bmatrix}
0 & 0 & \ldots & 0 & 1
\end{bmatrix}^T
\]

\[
C_i = C_{mi} = \begin{bmatrix}
1 & 0 & 0 & \ldots & 0
\end{bmatrix}
\]

If the system and the reference model can be expressed in this general form then,

\[
(A_{mi} - A_i) = \begin{bmatrix}
0 & 0 & \ldots & 0 \\
0 & 0 & \ldots & 0 \\
\vdots & \vdots & \ddots & \vdots \\
(\beta_i^{(k-1)} - \alpha_i^{(k-1)}) & (\beta_i^{(k-2)} - \alpha_i^{(k-2)}) & \ldots & (\beta_i^i - \alpha_i^i)
\end{bmatrix}
\]
From this equation and the form of $B_i$ in (2.25) it is clear that,

$$\exists K_i^* \mid A_{mi} = A_i + B_i K_i^{*T}, \forall i$$  \hspace{1cm} (2.27)

and (2.21) is satisfied. In fact, this desired gain vector is given by,

$$K_i^* = \left[ (\beta_{(k-1)}^i - \alpha_{(k-1)}^i) \quad (\beta_{(k-2)}^i - \alpha_{(k-2)}^i) \quad \ldots \quad (\beta_1^i - \alpha_1^i) \right]^T$$  \hspace{1cm} (2.28)

This analysis has shown that if the subsystem dynamics can be written in the companion form of (2.25), then the restriction imposed in (2.21) will be satisfied. While the companion form is not required to satisfy this restriction, it is a sufficient condition. Further, a sufficient condition for the system to be expressible in the proposed companion form is that the subsystems be both controllable and observable [10]. Thus, a sufficient condition for the restriction in (2.21) to be met is that the system being analyzed is both controllable and observable.

2.2 Problem Statement

With the necessary constraints on the plant dynamics satisfied, the problem is then to design the subsystem controller gains $K_i$ and the estimates of the interaction terms $\hat{\psi}_{ij}$ in (2.19) for each of the $M$ subsystems, such that the plant states $x_i$ track those of the reference model $x_{mi}$. It is further required that the design of the subsystem controllers provide a mechanism for reducing the amount of inter-node communications, while still guaranteeing stability and achieving the desired tracking performance.

2.3 Distributed Control Method

The controller form proposed in (2.19) requires that the $i$th subsystem have complete knowledge of the state vectors of the other $(M - 1)$ subsystems (the $x_j$ terms in this equation). This is equivalent to a centralized controller implementation and would require that the $i$th subsystem communicate its full state vector ($x_i$) to all of
the other \((M - 1)\) subsystem at each sampling instant. In many cases, implementation constraints (such as limited communication bandwidth) make this requirement impractical. In order to limit the amount of communication required between subsystems, a state estimation scheme will be implemented at each node. These estimators will predict the states of the other nodes in the system. The subsystem controllers can then use the output of these estimators to calculate their control actions, rather than requiring the actual states of the other subsystems. As long as the estimator outputs are used in the subsystem calculations, no communication between subsystems is required. With this estimation scheme in mind, the control vector for the \(i\)th subsystem needs to be modified from the form shown in (2.19). Define the state tracking error for the \(i\)th subsystem node as follows,

\[
e_i = x_i - x_{mi}
\]

(2.29)

Now, the state space of the system is partitioned into two regions as follows:

\[
x \in \Omega_1 : e_i^T e_i < E_0^i, \forall i
\]

\[
x \in \Omega_2 : \exists i | e_i^T e_i \geq E_0^i
\]

(2.30)

where \(E_0^i\) is the desired performance bound for the \(i\)th subsystem. The controller output will then be different for each of these two regions. Using a bounded error approach similar to that presented in [1] and [11], the controller for the \(i\)th subsystem is defined as,

\[
u_i = \begin{cases} K_i^T x_i - \sum_{j=1,j\neq i}^M \hat{y}_{ij}^T \hat{x}_j, & \text{for } x \in \Omega_1 \\ K_i^T x_i - \sum_{j=1,j\neq i}^M \hat{y}_{ij}^T \hat{x}_j & \text{for } x \in \Omega_2 \end{cases}
\]

(2.31)

where \(\hat{x}_j \in \mathbb{R}^{N_p}\) is the estimated state vector for the \(j\)th subsystem. Thus, the estimated states of the other \((M - 1)\) subsystems are used to determine the control vector of the \(i\)th node as long as the tracking performance meets the specified criteria.
Figure 6: Block Diagram Of Plant With Communication

\((e_i^T e_i < E_0^i)\). A block diagram of the system under these conditions is presented in Figure 7. Once the tracking error grows beyond this bound, the actual subsystem states will be communicated between nodes and used to determine the controller output \(u_i\) at each node (see Figure 6). The design of the controller and estimator parameters to ensure stability and desired tracking performance using this scheme will be investigated in Section 3. Section 2.4 describes the proposed method for each node to estimate the state vectors of the other nodes in the system.

2.4 Adaptive Estimation Method

One of the key features of the present approach is the use of an adaptive state estimator to reduce the amount of communications required between subsystems while still maintaining acceptable tracking of the reference model states and guaranteeing system stability. Rather than requiring that each subsystem node communicate its state vector to all of the other \((M - 1)\) nodes at each sampling instant, each node uses a local state estimator to estimate the behavior of the other nodes. These estimated
state values are then used by each node to calculate the required local controller output to achieve the desired tracking performance.

There are two separate forms for the proposed state estimator that need to be considered—one for each of the state space regions defined in (2.30). Each of these forms will be considered separately below.

2.4.1 Estimator Form For $x \in \Omega_2$

Recall that full knowledge of the system states is assumed when $x \in \Omega_2$. As long as the state vector remains in this region, communication between subsystems will occur and the adaptive process will adjust the controller and estimator parameters. For this situation, the state dynamics for the $j$th subsystem node are as follows:

$$\dot{x}_j = A_j x_j + B_j [\hat{K}_j^T x_j - \sum_{k=1,k \neq j}^M \hat{\psi}_{jk}^T x_k] + B_j r_j + B_j \sum_{k=1,k \neq j}^M \psi_{jk}^T x_k \tag{2.32}$$
Define the local closed loop state dynamics, ignoring the coupling interactions, for the $j$th node to be,

$$A_{cj} = A_j + B_j K_j^T \tag{2.33}$$

Ignoring the interactions between subsystems in (2.32), the following state estimator is then proposed [12]:

$$\dot{x}_j = A_{mj} \dot{x}_j + (\hat{A}_{cj} - A_{mj}) x_j + B_j r_j \tag{2.34}$$

This estimator form requires knowledge of the subsystem states $x_j$ that it is trying to estimate. The purpose of this form is to train the estimator while the actual state values are available. Once the parameters of the estimator have been sufficiently adapted, then the estimates of the states can be used in place of their actual values.

Now, for the $i$th node ($j \neq i$) the actual state values $x_j$ are only available when there is communication between the $i$th and $j$th nodes (when $x \in \Omega_2$). Once the system state vector enters region $\Omega_1$, inter-subsystem communication will cease and the actual state values $x_j$ will no longer be available. Thus, the form of the estimator will need to be modified to account for this. This is discussed in Section 2.4.2.

The motivation for choosing the estimator form in (2.35) becomes clearer after a minor manipulation of this equation. Using the fact that $\hat{A}_{cj} = \hat{A}_{cj} - A_{cj}$ this equation can be rewritten as,

$$\dot{x}_j = A_{mj} \dot{x}_j + A_{cj} x_j + \hat{A}_{cj} x_j + B_j r_j \tag{2.35}$$

Now, for the case where the output of the state estimator converges to the actual state vector $x_j$, then $\dot{x}_j \to 0$. Further, if the model for the local state dynamics $\hat{A}_{cj}$ converges to that of the actual dynamics $A_{cj}$, then $\hat{A}_{cj} \to 0$ as well. Under these circumstances, the first and third terms in (2.35) are zero and we are left with $\dot{x}_j = A_{cj} x_j$. This is exactly the form of the known state dynamics that the estimator
is trying to predict. Thus, if the adaptive process can drive the state estimation error \( \hat{x}_j \) and the modeling error \( \hat{A}_{cj} \) to zero, (2.35) provides the appropriate form for the estimator dynamics.

The estimator form in (2.35) does not account for the interactions between subsystems. However, these terms cannot be neglected in the actual system, so the form of this equation must be modified to account for this. The form of the estimator could be chosen as,

\[
\hat{x}_j = A_{mj} \hat{x}_j + (\hat{A}_{cj} - A_{mj})x_j + B_j r_j + B_j \sum_{k=1,k\neq j}^M \tilde{\psi}_{jk}^T x_k
\]  

(2.36)

where, \( \tilde{\psi}_{jk} = \psi_{jk} - \hat{\psi}_{jk} \) represents the error in estimating the interactions between nodes \( j \) and \( k \). Unfortunately, this form requires that the actual values of the estimation error \( \hat{\psi}_{jk} \) be known. Since the actual interaction terms \( \psi_{jk} \) are assumed to be unknown, this is not a realistic requirement. Thus, a slight modification needs to be made to (2.36). Define the following:

\[
\Gamma_{jk} = \tilde{\psi}_{jk}
\]  

(2.37)

The form of the adaptive state estimator is then be written as follows:

\[
\dot{x}_j = A_{mj} \hat{x}_j + (\hat{A}_{cj} - A_{mj})x_j + B_j r_j + B_j \sum_{k=1,k\neq j}^M \hat{\Gamma}_{jk}^T x_k
\]  

(2.38)

where, \( \Gamma_{jk} = \tilde{\psi}_{jk} \) is the actual error in modeling the coupling between nodes \( j \) and \( k \) and \( \hat{\Gamma}_{jk} \) represents the estimate of this error term. Thus, since the actual \( \psi_{jk} \) vector is not available for measurement, this quantity will be estimated. The choice of adaptive updates to ensure convergence of the tracking errors will be derived in section 3.
2.4.2 Estimator Form For $x \in \Omega_1$

While the system state vector is within region $\Omega_1$, communication between subsystems does not occur. Thus, the actual state vectors $x_j$ and $x_k$ in (2.38) are no longer available to the $i$th subsystem. Under these circumstances, the proposed estimator form needs to be modified slightly. In particular, the estimates of these states will be used instead of their actual values. Temporarily ignoring the coupling interactions between subsystems and substituting $x_j = \hat{x}_j$ in (2.34) then produces the following,

$$\dot{\hat{x}}_j = \hat{A}_{cj} \hat{x}_j + B_j r_j$$ \hspace{1cm} (2.39)

Thus, when $x \in \Omega_1$ the estimate of the closed loop dynamics of the $j$th subsystem ($\hat{A}_{cj}$) determines how the estimated state vector $\hat{x}_j$ will evolve. For the more general case that includes the contribution from the coupling between subsystems, substituting $x_j = \hat{x}_j$ and $x_k = \hat{x}_k$ in (2.38) produces,

$$\dot{\hat{x}}_j = \hat{A}_{cj} \hat{x}_j + B_j r_j + B_j \sum_{k=1, k\neq j}^M \hat{F}_{jk}^T \hat{x}_k$$ \hspace{1cm} (2.40)

This is the form of the state estimator that will be utilized when $x \in \Omega_1$. No adaptation of the estimator parameters occurs while the system state vector remains within this region. It is assumed that the estimator is providing sufficiently accurate results, so long as the tracking error for all of the subsystems remains below the desired thresholds. If the tracking error for any of the subsystems grows beyond its threshold ($E^i_o$), then communication between subsystems and the adaptive process are both restarted. The goal is then to adjust the controller and estimator parameters to improve system performance while the actual state vectors of the other subsystems are available.
3 Controller Design and Stability Analysis

The problem presented in this paper is to design a distributed controller in the form of (2.31) that will force the system states of (2.17) to track the states of the desired reference model (2.10). In addition, an adaptive state estimator is designed for each subsystem which enables a reduction in the required amount of inter-node communications. The approach presented here is to design the controller and estimator parameters using an adaptive method based on Lyapunov stability analysis. Thus, the adaptive updates for the controller and estimator parameters will be selected so as to ensure stability of the tracking errors $e_i$ and state estimation errors $\hat{x}_i$.

In order to apply the Lyapunov stability analysis techniques to the proposed distributed control scheme, a separate Lyapunov function ($V_i$) will be identified for each subsystem. The stability of the overall system can then be concluded using the following Lyapunov function:

$$V = \sum_{i=1}^{M} V_i$$

Upon close inspection of (2.31), it is obvious that there are two cases which require separate consideration in the stability analysis: $x \in \Omega_1$ and $x \in \Omega_2$ as defined in (2.30). Each of these cases will be considered separately below.

3.1 No Adaptation

The first case to be considered is when the system state vector is within region $\Omega_1$ as defined by,

$$x \in \Omega_1 : \varepsilon_i^T e_i < E_0^i, \forall i$$

In this case, there is no communication of the local state vectors between subsystems. This means that at each node, the estimator outputs $\hat{x}_j$ will be used to calculate the controller output $u_i$ at each sampling instant (see equation 2.31). Adaptation of the estimator and the controller parameters does not occur for $x \in \Omega_1$. During this dead-band in adaptation, it is not required to demonstrate convergence of the state
tracking error. It is assumed that the bounds $E_0^i$ are set such that the performance of the system is considered acceptable as long as $x \in \Omega_1$. In this case, because the system states are upper bounded by the given threshold values $E_0^i$, the system is bounded-input bounded-output (BIBO) stable by assumption.

3.2 During Adaptation

The second case to be considered is when the system state vector is within region $\Omega_2$ as defined by,

$$x \in \Omega_2 : \exists i | e_i^T e_i \geq E_0^i$$

(3.43)

In this case, the estimator and the controller parameters will be adapted in an effort to drive the system states towards those of the reference model (and thus to drive the system state vector back into region $\Omega_1$). In addition, communication between subsystems will be occurring such that the entire system state vector $x$ will be known to each subsystem. This means that the $i$th subsystem will have access to the actual state vectors $x_j$ rather than just the estimates $\hat{x}_j$. This information can then be used to calculate the local controller output $u_i$ as well as the adaptive updates to local controller and state estimator parameters.

For these conditions, Lyapunov stability theory will be used to determine the appropriate controller updates that guarantee stability of the overall system. In order to simplify the development, the stability of the adaptive controller and the adaptive state estimators are considered separately below.

3.2.1 Adaptive Controller

Following the development in [6], consider the following Lyapunov function candidate for the $i$th subsystem:

$$V_i(e_i, \tilde{K}_i, \tilde{\psi}_{i,j}) = e_i^T P_i e_i + \tilde{K}_i^T \tilde{K}_i + \sum_{j=1, j \neq i}^{M} [\tilde{\psi}_{i,j}^T \tilde{\psi}_{i,j}]$$

(3.44)
where,

\[ \bar{K}_i = \dot{K}_i - K_i^* \]  

(3.45)

\[ \bar{\psi}_{ij} = \psi_{ij} - \dot{\psi}_{ij} \]  

(3.46)

and \( P_i \) is a positive-definite matrix that is the solution to the following Lyapunov equation:

\[ A_{mi}^T P_i + P_i A_{mi} = -Q_i \]  

(3.47)

It is clear by inspection that the function in (3.44) is positive definite. Taking the time derivative of this equation along the system trajectory then gives the following:

\[ \dot{V}_i = \dot{e}_i^T P_i e_i + e_i^T P_i \dot{e}_i + \dot{\bar{K}}_i^T \dot{K}_i + \dot{K}_i^T \dot{\bar{K}}_i + \sum_{j=1,j \neq i}^{M} \left[ \ddot{\psi}_{ij} \dot{\psi}_{ij} + \dot{\psi}_{ij} \dot{\psi}_{ij} \right] \]  

(3.48)

Referring to (2.29), the time-derivative of the state tracking error can be written as follows:

\[ \dot{\epsilon}_i = \dot{x}_i - \dot{x}_{mi} \]  

(3.49)

Substitution of (2.12) and (2.20) into this equation then gives:

\[ \dot{\epsilon}_i = A_i x_i + B_i [\dot{\bar{K}}_i^T x_i - \sum_{j=1,j \neq i}^{M} \ddot{\psi}_{ij} T x_j + r_i] + A_{mi} x_i - B_i r_i \]  

(3.50)

Now, using (2.21) the equation above can be rearranged to give:

\[ \dot{\epsilon}_i = B_i \dot{\bar{K}}_i^T x_i + A_{mi} \epsilon_i + B_i \sum_{j=1,j \neq i}^{M} \ddot{\psi}_{ij} T x_j \]  

(3.51)

The last four terms in (3.48) involve the time-derivatives of the controller parameter errors (\( \dot{\psi}_{ij} \) and \( \dot{K}_i \)). The equations for these derivatives can be obtained from the
definitions of the errors themselves as follows:

\[
\hat{K}_i = K_i - K_i^* \quad (3.52)
\]

\[
\hat{\psi}_{ij} = \psi_{ij} - \hat{\psi}_{ij} \quad (3.54)
\]

Similarly, for the error in the estimates of the interaction terms:

\[
\hat{\psi}_{ij} = \psi_{ij} - \hat{\psi}_{ij} 
\]

Following the development in [6], the proposed adaptive updates are as follows:

\[
\hat{K}_i^T = -e_i^T P_i B_i x_i^T 
\]

\[
\hat{\psi}_{ij} = e_i^T P_i B_i x_j^T 
\]

Substitution of (3.55) and (3.66) into (3.53) and (3.54) produces the following:

\[
\hat{K}_i^T = -e_i^T P_i B_i x_i^T 
\]

\[
\hat{\psi}_{ij} = -e_i^T P_i B_i x_j^T 
\]

Using (3.57), (3.58), and (3.51) in (3.48) along with the fact that \((e_i^T P_i B_i)\) is a scalar quantity then yields:

\[
\dot{V}_i = e_i^T [P_i A_{mi} + A_{mi}^T P_i] e_i 
\]

From the definition in 3.47, this can then be rewritten as,

\[
\dot{V}_i = -e_i^T Q_i e_i \leq 0 
\]

Thus, \(V_i\) is positive definite and \(\dot{V}_i\) is negative semi-definite. From Lyapunov stability theory, it can then be concluded that the adaptive updates in (3.55) and (3.56)
guarantee stability of $e_i$, $\tilde{K}_i$ and $\bar{y}_{ij}$. Using an extension of Barbalat's Lemma, this result can be extended slightly. Recall that Barbalat's Lemma provides sufficient conditions which guarantee that a function's derivative will converge to zero given that the function itself tends towards a finite limit. Specifically, this lemma can be stated as follows (see lemma 4.2 in [13]):

**Lemma 3.1 (Barbalat)** If the differentiable function $f(t)$ has a finite limit as $t \to \infty$, and if $\dot{f}$ is uniformly continuous, then $\dot{f} \to 0$ as $t \to \infty$.

A useful extension of this lemma for stability analysis is Lemma 4.3 in [13]. This lemma can be stated as follows,

**Lemma 3.2** If the scalar function $f(t)$ satisfies the following,

- $f(t)$ is lower bounded
- $\dot{f}(t)$ is negative semi-definite
- $\dot{f}(t)$ is uniformly continuous in time

then $\dot{f}(t) \to 0$ as $t \to \infty$

As pointed out in [13], a sufficient condition for uniform continuity of a differentiable function is that its derivative be bounded. Thus, the third condition in Lemma 3.2 can be exchanged for ensuring that $\dot{f}$ is bounded.

Referring back to (3.44), it is clear that $V_i$ is a positive definite function. Further, from (3.60), $\dot{V}_i$ is negative semi-definite. These two conditions ensure that $V_i$ is lower bounded and the first condition of Lemma 3.2 is satisfied. Since $\dot{V}_i$ is negative semi-definite, the second condition of the lemma is also satisfied. The uniform continuity of $\dot{V}_i$ (the third condition of the lemma) is then proven by demonstrating that $\dot{V}_i$ is bounded. Taking the derivative of (3.60), the following is obtained:

$$
\dot{V}_i = -e_i^T Q_i e_i - e_i^T Q_i \dot{e}_i \tag{3.61}
$$
Recall from (3.51) that,

$$\dot{e}_i = B_i \tilde{K}_i^T x_i + A_{mi} e_i + B_i \sum_{j=1, j \neq i}^{M} \tilde{\psi}_{ij}^T x_j$$  \hspace{1cm} (3.62)

It was already demonstrated that $e_i$, $\tilde{K}_i$, and $\tilde{\psi}_{ij}$ are all bounded. Further, since it is assumed that the reference model states are stable, $x_{mi}$ is bounded by assumption. Since $e_i$ and $x_{mi}$ are both bounded and $e_i = x_i - x_{mi}$, $x_i$ must be bounded as well. A close inspection then indicates that all the terms in (3.62) are stable, therefore $\dot{e}_i$ is bounded. Since $e_i$ and $\dot{e}_i$ are both bounded, (3.61) indicates that $\ddot{V}_i$ must be bounded. Once again, this is a sufficient condition to guarantee the uniform continuity of $\ddot{V}_i$ and the last condition required for Lemma 3.2 is satisfied. From this lemma, it is then possible to conclude that:

$$\lim_{t \to \infty} \ddot{V}_i = 0$$ \hspace{1cm} (3.63)

Using (3.60) then gives,

$$\lim_{t \to \infty} \dot{V}_i = \lim_{t \to \infty} [-e_i^T Q e_i] = 0$$ \hspace{1cm} (3.64)

which then produces,

$$\lim_{t \to \infty} e_i = 0$$ \hspace{1cm} (3.65)

Hence, it has been demonstrated that when $x \in \Omega_2$, the proposed adaptive controller updates will in fact drive the tracking error towards zero. However, because of the bounded error adaptive process being implemented, adaptation is only allowed to take place when one of the subsystem tracking errors grows beyond its error bound $E_0^i$. At all other times (when $x \in \Omega_1$) the adaptive process is stopped. Because of this, it cannot be guaranteed that the state tracking error will converge to zero. Instead, all that can be shown is that when $x \in \Omega_2$ the tracking error will be driven towards zero, thereby driving the state vector of the system back into $\Omega_1$. Once the system state vector enters region $\Omega_1$, the adaptive process stops and the convergence of the state tracking error can no longer be guaranteed. Figure 3.2.1 illustrates the
trajectory of the state vector and how this relates to the inter-node communications and the adaptive process.

Since the designer has choice of the tracking error thresholds \( E_0^i \), convergence of the tracking error to zero is not required. Performance of the system is considered sufficient as long as the system state vector is within region \( \Omega_1 \). As soon as the system states drift outside this region, the adaptive process will restart and the system tracking errors will be driven back below the specified performance thresholds.

This analysis does not, however, provide any guarantee of how long the state vector will remain in region \( \Omega_1 \). In fact, it is possible that the system state vector could hover near the boundary, resulting in "chattering". Intuitively, the use of the state estimator should help to reduce the occurrence of significant amounts of chattering. The state estimator provides a mechanism to increase the likelihood that the system state vector will remain within region \( \Omega_1 \) once it enters. Each time the system state vector leaves region \( \Omega_1 \), the adaptive process is restarted and the performance of the state estimator should, in general, improve. Intuitively, this should help to limit the amount of chattering that actually occurs. Although chattering of the system state vector along the error threshold boundary \( E_0^i \) is still possible, the simulation results have not exhibited this type of behavior.
Figure 8: Trajectory Of the System State Vector
3.2.2 Adaptive State Estimator

Referring to (2.38), the form of the state estimator can be written as follows,

\[
\dot{x}_j = A_{mj} \dot{x}_j + (\dot{A}_{cj} - A_{mj})x_j + B_j r_j + B_j \sum_{k=1,k\neq j}^{M} \hat{\Gamma}_{jk}^T x_k
\]  

(3.66)

The goal of the adaptive process is then to adjust \( \dot{A}_{cj} \) and \( \hat{\Gamma}_{jk} \) in order to guarantee the stability and convergence of the estimator error \( \hat{x}_j = \dot{x}_j - x_j \). First define the following,

\[
\hat{\Gamma}_j = \sum_{k=1,k\neq j}^{M} \hat{\Gamma}_{jk}^T \hat{\Gamma}_{jk} \quad (3.67)
\]

where \( \hat{\Gamma}_{jk} = \hat{\Gamma}_{jk} - \Gamma_{jk} \). Next, the following Lyapunov function candidate is defined:

\[
V_j(\dot{x}_j, \dot{A}_{cj}, \hat{\Gamma}_j) = \dot{x}_j^T P_{mj} \dot{x}_j + tr[\dot{A}_{cj}^T P_{mj} \dot{A}_{cj}] + \hat{\Gamma}_j
\]  

(3.68)

where \( P_{mj} \) is a positive-definite matrix that is the solution to the following Lyapunov equation:

\[
A_{mj}^T P_{mj} + P_{mj} A_{mj} = -Q_{mj}
\]  

(3.69)

In this equation, \( Q_{mj} \) is a positive-definite matrix as well.

To simplify the stability analysis, the function in (3.68) will be analyzed as three separate Lyapunov functions where \( V_j = V_{j1} + V_{j2} + V_{j3} \) and:

\[
V_{j1} = \dot{x}_j^T P_{mj} \dot{x}_j \quad (3.70)
\]

\[
V_{j2} = tr[\dot{A}_{cj}^T P_{mj} \dot{A}_{cj}] \quad (3.71)
\]

\[
V_{j3} = \hat{\Gamma}_j = \sum_{k=1,k\neq j}^{M} \hat{\Gamma}_{jk}^T \hat{\Gamma}_{jk} \quad (3.72)
\]

Taking the derivative of these equations then gives,

\[
\dot{V}_{j1} = \dot{\dot{x}}_j^T P_{mj} \dot{x}_j + \dot{x}_j^T P_{mj} \dot{x}_j \quad (3.73)
\]

\[
\dot{V}_{j2} = tr[\dot{A}_{cj}^T P_{mj} \dot{A}_{cj}] + tr[\dot{A}_{cj}^T P_{mj} \dot{A}_{cj}] \quad (3.74)
\]
\[ \dot{V}_{j3} = \sum_{k=1,k\neq j}^{M} [\tilde{\Gamma}_{jk}^{T} \tilde{\Gamma}_{jk} + \tilde{\Gamma}_{jk}^{T} \tilde{\Gamma}_{jk}] \]  

(3.75)

Now, using (2.20) and (3.66), \( \dot{x}_j = \dot{x}_j - \dot{x}_j \) can be written as,

\[ \dot{x}_j = A_{mj} \tilde{x}_j + \tilde{A}_{cj} x_j + B_{j} \sum_{k=1,k\neq j}^{M} \tilde{\Gamma}_{jk}^{T} x_k \]  

(3.76)

Substituting this equation into (3.73) then gives,

\[ \dot{V}_{j1} = \dot{x}_j^{T} [A_{mj} P_{mj} + P_{mj} A_{mj}] \tilde{x}_j + 2 \tilde{x}_j^{T} P_{mj} \tilde{A}_{cj} x_j + 2 \sum_{k=1,k\neq j}^{M} \tilde{x}_j^{T} P_{mj} B_{j} \tilde{\Gamma}_{jk}^{T} x_k \]  

(3.77)

This can then be reduced to,

\[ \dot{V}_{j1} = -\tilde{x}_j^{T} Q_{mj} \tilde{x}_j + 2 \tilde{x}_j^{T} P_{mj} \tilde{A}_{cj} x_j + 2 \sum_{k=1,k\neq j}^{M} \tilde{x}_j^{T} P_{mj} B_{j} \tilde{\Gamma}_{jk}^{T} x_k \]  

(3.78)

Choosing the following adaptive updates,

\[ \tilde{A}_{cj} = B_{j} K_{j} - x_j e_{mj}^{T} \]  

(3.79)

\[ \tilde{\Gamma}_{jk} = -e_{mj}^{T} P_{mj} B_{j} x_k^{T} - \psi_{jk} \]  

(3.80)

and combining these updates with (3.74), (3.75), and (3.78) then gives,

\[ \dot{V}_j = -\tilde{x}_j^{T} Q_{mj} \tilde{x}_j \]  

(3.81)

From this equation, it can be seen that \( \dot{V}_j \) is in fact negative semi-definite. Since it was already demonstrated that \( V_j \) is positive-definite, this means that the chosen \( V_j \) is a Lyapunov function for the system and that \( \tilde{x}_j, \tilde{A}_{cj}, \) and \( \tilde{\Gamma}_j \) are stable. Taking the time derivative of (3.81) then produces,

\[ \ddot{V}_j = -\tilde{x}_j^{T} Q_{mj} \tilde{x}_j - \tilde{x}_j^{T} Q_{mj} \dot{x}_j \]  

(3.82)
It has already been shown that $\tilde{x}_j$, $A_{cj}$, and $\tilde{\Gamma}_j$ are stable. Furthermore, the states $x_j$ that are being estimated are assumed to be stable. Referring back to (3.76), all of the terms in this equation have been proven to be stable, therefore $\tilde{x}_j$ must be bounded as well. Thus, from (3.82) $\dot{V}_j$ must be bounded - a sufficient condition to ensure that $\dot{V}_j$ must be uniformly continuous in time [13]. The conditions for Lemma 3.2 are then satisfied and so it can be concluded that $\dot{V}_j \to 0$ as $t \to \infty$. Using (3.81) it can then be concluded that $\tilde{x}_j \to 0$ as $t \to \infty$ as well.

Once again, however, the adaptive process will only be allowed to take place when one of the subsystem tracking errors grows beyond its error bound $E_{t0}$. At all other times, when $x \in \Omega_1$, the adaptive process is halted. Because of this, it cannot be guaranteed that the state estimation error will converge to zero. Instead, all that can be shown is that when $x \in \Omega_2$ the state estimation error will be driven towards zero. Once the system state vector enters region $\Omega_1$, the adaptive process stops and the convergence of the state estimation error can no longer be guaranteed. Intuitively this approach makes sense because the performance of the state estimator is assumed to be good enough so long as the state tracking errors are below the desired thresholds.
4 Performance Metrics

This paper focuses on the design of M decentralized controllers ($\Pi_i$) that will achieve the desired state tracking performance while also limiting the amount of communication required between subsystems in the distributed implementation. Referring to (2.31), one can see that the threshold value $E_0^i$ plays an important role in determining the level of communications required. In [1] it was shown that adjusting the state estimation error threshold allowed the designer to tradeoff the required amount of communications for system tracking performance. A similar result can be shown for the choice of the tracking error thresholds $E_0^i$ in the present method. The larger these threshold values are chosen to be, the larger the tracking errors must be before communications between subsystems will be restarted. This will reduce the required amount of communications but, at the same time, it will also degrade the tracking performance of the system. To this end, two metrics were utilized to determine how well each of the desired objectives was achieved. Each of these metrics is outlined in more detail below. Simulation results demonstrating the impact of adjusting the $E_0^i$ thresholds on system communication and state tracking performance will be presented in section 5.

4.1 Communication Metric

The method examined in this paper attempts to reduce the amount of communication between subsystems through use of state estimators on each subsystem node. Assume that the communications between subsystems are sampled at discrete sampling instants and define the following communications flag:

$$C_{flag}(k) = \begin{cases} 
0, & \text{for } x(k) \in \Omega_1 \\
1, & \text{for } x(k) \in \Omega_2 
\end{cases} \quad (4.83)$$
If this flag is equal to one, then communications between subsystems do occur for that sampling instant and an update of the adaptive parameters will also occur. Conversely, if this flag is equal to zero for sample \( k \), then there is no information exchange between subsystems nor an update to the adaptive parameters for this sampling instant. In order to quantify how much the required amount of communication between subsystems has been reduced, the following metric was defined:

\[
M_{\text{comm}}(k_1, k_2) = \frac{1}{(k_2 - k_1 + 1)} \sum_{k=k_1}^{k=k_2} C_{\text{flag}}(k)
\] (4.84)

This metric is simply the mean of the communications flag over the specified time period. It provides a measure of how much inter-node communication occurs during this time frame.

4.2 Performance Metric

The main objective of the controller design presented in this paper is to force the states of the plant in (2.9) to track those of the reference model given in (2.10). As a measure of how well the state tracking objective is being achieved, define the following performance metric:

\[
M_{\text{perf}}(k_1, k_2) = \sum_{k=k_1}^{k=k_2} \max_i \{|e_i(k)|\}
\] (4.85)

where \( i = 1, 2, \ldots, M \). This metric takes the maximum state tracking error across all of the system states at each time instant \( k \), then sums all of these together over the desired time window of interest. Essentially, this amounts to accumulating the worst-case tracking error for each sampling instant during the specified time window.
5 Simulation Results

In order to demonstrate the effectiveness of the proposed distributed adaptive approach, several simulation experiments were performed. The sample plant in these simulations was the spring-mass-damper system shown in Figure 9.

Figure 9: Spring-Mass-Damper Plant

In this investigation, two different simulation experiments were performed. In the first experiment, the proposed adaptive estimation and control approach was applied to each subsystem of the nominal plant. The objective of this experiment was to verify that the proposed distributed controller does in fact drive the system tracking error below the desired threshold. Also, the ability of the adaptive state estimation scheme to reduce the required communications between subsystems was demonstrated. The results of these simulations are provided in section 5.2.

The goal of the second simulation experiment was to compare the performance of a static state estimation scheme with the proposed adaptive scheme. In particular, the
performance of both types of estimation approach were compared when the actual plant dynamics were different than those used in the design of the estimator and controller parameters. The results of this experiment are presented in section 5.4.

5.1 Spring-Mass-Damper System

An analysis of the system in Figure 9, applying $\sum F = ma$ to each mass produces,

$$\dot{x}_1 = -\frac{(k_1 + k_4)}{m_1}x_1 - \frac{(b_1 + b_4)}{m_1}\dot{x}_1 + \frac{k_1}{m_1}x_2 + \frac{b_1}{m_1}\dot{x}_2 + \frac{k_4}{m_1}x_3 + \frac{b_4}{m_1}\dot{x}_3 + \frac{1}{m_1}u_1$$  \hspace{1cm} (5.86)

$$\dot{x}_2 = \frac{k_1}{m_2}x_1 + \frac{b_1}{m_2}\dot{x}_1 - \frac{(k_1 + k_2)}{m_2}x_2 - \frac{(b_1 + b_2)}{m_2}\dot{x}_2 + \frac{k_2}{m_2}x_3 + \frac{b_2}{m_2}\dot{x}_3 + \frac{1}{m_2}u_2$$  \hspace{1cm} (5.87)

$$\dot{x}_3 = \frac{k_4}{m_3}x_1 + \frac{b_4}{m_3}\dot{x}_1 + \frac{k_2}{m_3}x_2 + \frac{b_2}{m_3}\dot{x}_2 - \frac{(k_2 + k_3 + k_4)}{m_3}x_3 - \frac{(b_2 + b_3 + b_4)}{m_3}\dot{x}_3 + \frac{1}{m_3}u_3$$  \hspace{1cm} (5.88)

The system dynamics can then be written in state space form as follows,

$$\dot{X}(t) = AX(t) + Bu(t)$$

$$y(t) = CX(t)$$  \hspace{1cm} (5.89)

where,

$$X(t) = \begin{bmatrix} x_1 & \dot{x}_1 & x_2 & \dot{x}_2 & x_3 & \dot{x}_3 \end{bmatrix}^T$$  \hspace{1cm} (5.90)

$$u(t) = \begin{bmatrix} u_1 & u_2 & u_3 \end{bmatrix}^T$$  \hspace{1cm} (5.91)

and,

$$A = \begin{bmatrix} 0 & \frac{1}{m_1} & 0 & 0 & 0 & 0 \\ \frac{-(k_1 + k_4)}{m_1} & 0 & \frac{k_1}{m_1} & \frac{b_1}{m_1} & \frac{k_4}{m_1} & \frac{b_4}{m_1} \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{k_1}{m_2} & \frac{b_1}{m_2} & \frac{(k_1 + k_2)}{m_2} & 0 & \frac{k_2}{m_2} & \frac{b_2}{m_2} \\ 0 & 0 & 0 & 0 & 1 & 0 \\ \frac{k_4}{m_3} & \frac{b_4}{m_3} & \frac{k_2}{m_3} & \frac{b_2}{m_3} & \frac{(k_2 + k_3 + k_4)}{m_3} & \frac{(b_2 + b_3 + b_4)}{m_3} \end{bmatrix}$$  \hspace{1cm} (5.92)
This system is then partitioned into three subsystems, where each subsystem describes the local dynamics of one of the three masses. The state vectors for the three subsystems are then,

\[ X_1(t) = \begin{bmatrix} x_1 & \dot{x}_1 \end{bmatrix}^T \]  

\[ X_2(t) = \begin{bmatrix} x_2 & \dot{x}_2 \end{bmatrix}^T \]  

\[ X_3(t) = \begin{bmatrix} x_3 & \dot{x}_3 \end{bmatrix}^T \]

Examination of Figure 9 clearly shows that there is physical coupling between the masses due to the springs and dampers. However, the proposed partitioning attempts to isolate the local dynamics of each mass. Because of this, the coupling between masses will then need to be treated as interaction terms between subsystems in the control analysis. With this in mind, the local state dynamics can be described by,

\[ A_1 = \begin{bmatrix} 0 & \frac{1}{m_1} \\ \frac{1}{m_2} & 0 \\ \frac{1}{m_3} & 0 \\ 0 & \frac{1}{m_1} \end{bmatrix} \]  

\[ B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & \frac{1}{m_3} \end{bmatrix} \]  

\[ C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \]
\[
A_2 = \begin{bmatrix}
0 & 1 \\
\frac{-(k_1+k_2)}{m_2} & \frac{-(b_1+b_1)}{m_2}
\end{bmatrix} 
\]  
(5.99)

\[
A_3 = \begin{bmatrix}
0 & 1 \\
\frac{-(k_2+k_3+k_4)}{m_3} & \frac{-(b_2+b_3+b_4)}{m_3}
\end{bmatrix} 
\]  
(5.100)

and the coupling between subsystems by \( A_{ij} = B_i \psi_{ij}^T \) where,

\[
\psi_{12}^T = \begin{bmatrix}
k_1 \\
b_1
\end{bmatrix}_{m_1}, \quad \psi_{13}^T = \begin{bmatrix}
k_4 \\
b_1
\end{bmatrix}_{m_1} 
\]  
(5.101)

\[
\psi_{21}^T = \begin{bmatrix}
k_1 \\
b_1
\end{bmatrix}_{m_2}, \quad \psi_{23}^T = \begin{bmatrix}
k_2 \\
b_2
\end{bmatrix}_{m_2} 
\]  
(5.102)

\[
\psi_{31}^T = \begin{bmatrix}
k_4 \\
b_3
\end{bmatrix}_{m_3}, \quad \psi_{32}^T = \begin{bmatrix}
k_2 \\
b_3
\end{bmatrix}_{m_3} 
\]  
(5.103)

and the local input matrices are,

\[
B_1 = \begin{bmatrix}
0 \\
\frac{1}{m_1}
\end{bmatrix}, \quad B_2 = \begin{bmatrix}
0 \\
\frac{1}{m_2}
\end{bmatrix}, \quad B_3 = \begin{bmatrix}
0 \\
\frac{1}{m_3}
\end{bmatrix} 
\]  
(5.104)

From this analysis, it is easily seen that the partitioned subsystems are in the desired companion form (see equation 2.25). In addition, the restriction on the coupling terms (see equation 2.14) is also satisfied. With the plant in the appropriate form, the proposed adaptive control approach was then applied to each subsystem.

### 5.2 Nominal Plant Results

In order to demonstrate the performance of the proposed distributed adaptive estimation and control method, simulation experiments were conducted on the spring-mass-damper system outlined in section 5.1. For the nominal plant, the following
parameter values were used:

\[
\begin{align*}
    m_1 &= 1.0, \quad m_2 = 2.0, \quad m_3 = 4.0 \\
    k_1 &= 1, \quad k_2 = 3, \quad k_3 = 4, \quad k_4 = 3 \\
    b_1 &= 0.1, \quad b_2 = 0.3, \quad b_3 = 0.1, \quad b_4 = 0.2
\end{align*}
\]

Here the units for the masses, spring constants, and dampers are kg, N/m, and (N·s)/m respectively. In this experiment, the goal of the subsystem controllers was to force the subsystems states \(X_i\) to track those of the following reference model,

\[
A_{m1} = \begin{bmatrix} 0 & 1 \\ -18 & -16 \end{bmatrix}, \quad A_{m2} = \begin{bmatrix} 0 & 1 \\ -40 & -22 \end{bmatrix}, \quad A_{m3} = \begin{bmatrix} 0 & 1 \\ -50 & -15 \end{bmatrix}
\]

(5.105)

The input matrices of the reference model were assumed to be known and to be identical to those of the system such that: \(B_{m1} = B_1\), \(B_{m2} = B_2\), and \(B_{m3} = B_3\). The matrices describing the local subsystem dynamics \(A_i\) were assumed to be unknown as were the coupling vectors \(\psi_{ij}\). The reference inputs were given by \(r_1 = r_2 = r_3 = 100 \sin(0.2\pi t)\). The bounds on the tracking error were chosen to be \(E_0^1 = E_0^2 = E_0^3 = 0.025\).

For comparison, the performance of the adaptive controller under conditions of constant communication and adaptation was first simulated. In this case, the subsystem controllers had access to the full state vector of the system at each sampling instant. This case is equivalent to that of a centralized controller since message traffic delays were not included in the simulation. The tracking performance under these conditions is shown in figures 10-12 and the performance of the state estimator is shown in figures 13-15. In both of these sets of figures, it is easily seen that the system states converge towards those of the reference model and that the estimator states converge towards those of the actual system states.
Figure 10: State Tracking Performance For Constant Communication and Adaptation: States $X_1$ and $X_2$
Figure 11: State Tracking Performance For Constant Communication and Adaptation: States $X_3$ and $X_4$
Figure 12: State Tracking Performance For Constant Communication and Adaptation: States $X_5$ and $X_6$
Figure 13: State Estimator Performance For Constant Communication and Adaptation: States $X_1$ and $X_2$
Figure 14: State Estimator Performance For Constant Communication and Adaptation: States $X_3$ and $X_4$
Figure 15: State Estimator Performance For Constant Communication and Adaptation: States $X_5$ and $X_6$
To further illustrate this point, Figure 16 shows the state tracking error histories (the tracking errors versus time) for this simulation. Each of the plots in this figure represents $e_{\text{track}}^2$ for one of the system states. It is clearly seen that the tracking errors for each of the subsystems are in fact driven towards zero by the adaptive process as $t \to \infty$. Figure 17 shows a similar set of results for the state estimator performance. Once again, it is clear from this figure that the state estimator errors are driven towards zero as $t \to \infty$. So, the proposed adaptive controller is performing as desired when communication of the full state vector occurs at each sampling instant.

For reference, the adaptive history of the controller and state estimator parameters is shown in Figures 18-19. Here it can be seen that the controller and estimator parameters are in fact stable and do converge to steady-state values during the time period of the simulation.
Figure 16: State Tracking Error History For Constant Communication and Adaptation
Figure 17: State Estimator Error History For Each System State - Continuous Adaptation
Figure 18: Adaptive History of Controller Parameters $\hat{K}_i$ and $\psi_{ij}$ Continuous Adaptation
Figure 19: Adaptive History of State Estimator Parameters $\hat{A}_{c,j}$ and $\hat{\Gamma}_{j,k}$ - Continuous Adaptation
These results demonstrate that for the given plant, the proposed adaptive controller will drive both the tracking errors and the estimator errors to zero. However, these simulations are only for the case of full state communication between each of the subsystems at each sampling instant (essentially a centralized controller). Simulations were then performed which were intended to represent the distributed implementation case. Here the bounded error communication and adaptation constraints were imposed: communication between subsystems and adaptation of the controller and estimator parameters only occurring for \( x \in \Omega_2 \). The tracking performance and state estimator performance for this case are shown in figures 20-22 and figures 23-25. The tracking error histories and state estimator error histories are provided in figures 26 and 27, respectively.
Figure 20: State Tracking Performance For Bounded Error Adaptation: States $X_1$ and $X_2$
Figure 21: State Tracking Performance For Bounded Error Adaptation: States $X_3$ and $X_4$
Figure 22: State Tracking Performance For Bounded Error Adaptation: States $X_5$ and $X_6$
Figure 23: State Estimator Performance For Bounded Error Adaptation: States $X_1$ and $X_2$
Figure 24: State Estimator Performance For Bounded Error Adaptation: States $X_3$ and $X_4$
Figure 25: State Estimator Performance For Bounded Error Adaptation: States $X_5$ and $X_6$
Figure 26: State Tracking Error History For Each System State - Bounded Error Adaptation
Figure 27: State Estimator Error History For Each System State  Bounded Error Adaptation
Once again, these figures demonstrate that the adaptive process drives the tracking and estimation errors towards zero. However, due to the dead-zone being implemented in the adaptive process (the adaptation only takes place when \( x \in \Omega_2 \)), there is a residual level of both tracking and estimation error. The reason for this is that the adaptive process is halted once the tracking performance falls below the specified thresholds \( E_0^j \) (once \( x \in \Omega_1 \)). When the adaptive process is halted, the state estimator and tracking errors are no longer actively driven towards zero. The amount of residual error that is permitted in the tracking error can be controlled through the choice of the thresholds \( E_0^j \).

The adaptation history in Figure 28 shows the number of sampling instants at which information exchange between subsystems occurred. This figure represents the communications flag at each sampling instant. Recall that the meaning of this flag is as follows:

\[
C_{\text{flag}}(k) = \begin{cases} 
0, & \text{for } x(k) \in \Omega_1 \\
1, & \text{for } x(k) \in \Omega_2 
\end{cases} \tag{5.106}
\]

Thus, if this flag is equal to one, then communications between subsystems occur for that sampling instant and an update of the adaptive parameters will also occur. Conversely, if this flag is equal to zero for sample \( k \), then there is no information exchange between subsystems nor an update to the adaptive parameters for this sampling instant.

From figure 28, it can be seen that the adaptive process tends to decrease the amount of communications required to maintain the desired system tracking performance. However, the amount of communications is not reduced to zero by the adaptive process during the time period of this simulation. The reason for this is the residual level of state estimation error for each of the states. This residual estimation error indicates that there is a difference between the dynamics of the plant and the model being used to determine the controller outputs for each subsystem when
$x \in \Omega_1$. This then leads to accumulation of tracking errors, which can lead to the system states drifting out of region $\Omega_1$. Each time this occurs, the adaptive process is restarted and the tracking and estimation errors are once again driven towards zero, eventually driving the system state vector back into region $\Omega_1$. 
Figure 28: Communications History For Bounded Error Adaptation
5.3 Effect Of Threshold Values $E_0^i$

The bounded error adaptation method presented in this paper provides an error threshold $E_0^i$ for each subsystem. This parameter is chosen as part of the design process. In [1] it was shown that adjusting the state estimation error threshold allowed the designer to tradeoff the required amount of communications for system tracking performance. A similar result can be shown for the choice of the tracking error thresholds $E_0^i$ in the present method. Intuitively, the larger these threshold values are chosen to be, the larger the tracking errors must be before communications between subsystems will be restarted. This will reduce the required amount of communications but, at the same time, will degrade the tracking performance of the system.

In order to demonstrate the impact of the choice of the error thresholds on the communication and tracking performance, simulations were performed with different values of $E_0^i$. The nominal plant presented in Section 5.1 was used and simulations of 300 seconds were performed for each of the desired error threshold values. Without loss of generality, the error thresholds for all of the subsystems were chosen to be equal. The communication and performance metrics presented in Section 4 were then calculated for each of these simulated cases. The results of these simulations are summarized in Figure 29. In this figure, it can clearly be seen that larger values for $E_0^i$ enable a reduction in communication at the expense of system tracking performance. Conversely, reducing the error thresholds improves the system tracking performance but requires more inter-subsystem communication. Thus, the designer can select the appropriate values for the error thresholds $E_0^i$ that provide the desired communication and tracking performance.

5.4 Comparison Of Results For Static and Adaptive Estimators

In order to demonstrate the benefits of using an adaptive rather than static state estimator in this type of distributed implementation, another set of simulation experiments was performed. For these experiments, the adaptive subsystem controllers and estimators were allowed to converge for a period of time. After completion of this
Figure 29: Effect of Changing the Error Threshold Parameter $E_0$ On System Tracking Performance and Required Inter-Subsystem Communication
initialization period, the dynamics of the plant were modified slightly. This essentially created a situation where the plant model used in creating the subsystem controllers and estimators was no longer a very accurate model of the actual plant. From this set of initial conditions, two separate simulations were then performed. In the first simulation, no further adaptation was allowed but communications between subsystems were permitted to occur as necessary (whenever \( x \in \Omega_2 \)). This simulation was essentially a static state estimator being used to limit the required communications between subsystems (similar in principle to that posed in [1]). In the second simulation, from this given initial condition the adaptive process was allowed to adjust the controller and estimator parameters whenever \( x \in \Omega_2 \). A diagram illustrating the difference between these two simulations is given in Figure 30.

![Diagram illustrating the difference between the two simulations](image)

Figure 30: Estimator Comparison Experiment
For this set of experiments, the nominal plant presented in Section 5.2 was used for the initialization portion of the simulation. The plant parameters for the nominal plant are repeated here for reference:

\[
\begin{align*}
    m_1 &= 1.0, \quad m_2 = 2.0, \quad m_3 = 4.0 \\
    k_1 &= 1, \quad k_2 = 3, \quad k_3 = 4, \quad k_4 = 3 \\
    b_1 &= 0.1, \quad b_2 = 0.3, \quad b_3 = 0.1, \quad b_4 = 0.2
\end{align*}
\]

Here the units for the masses, spring constants, and dampers are once again kg, N/m, and (N-s)/m, respectively. After this first portion of the simulation, the plant dynamics were altered by changing the physical parameters to the following:

\[
\begin{align*}
    m_1 &= 1.0, \quad m_2 = 2.0, \quad m_3 = 1.0 \\
    k_1 &= 2, \quad k_2 = 5, \quad k_3 = 4, \quad k_4 = 3 \\
    b_1 &= 0.05, \quad b_2 = 0.2, \quad b_3 = 0.05, \quad b_4 = 0.09
\end{align*}
\]

Using these new plant parameters, the two different simulations indicated in figure 30 were performed. A comparison of the tracking error histories \(e^2_{track}\) for the static and adaptive state estimation schemes is presented in figures 31-36. As expected, these figures clearly demonstrate that the adaptive estimation scheme is better capable of responding to the poorly modeled plant dynamics. In the case of the static estimator, these poorly modeled dynamics lead to significant increases in the amount of communications between subsystems. In fact, for the example simulated, the static estimator requires communications at each sampling instant once the plant dynamics have been changed. A comparison of the communications histories for the static and adaptive estimator simulations that illustrates this point is presented in figure 37. Using the communication metric presented in Section 4 to compare the results of the static and adaptive estimation schemes produces:

\[
\begin{align*}
    M_{\text{comm},(\text{static})} &= 0.83 \\
    M_{\text{comm},(\text{adapt})} &= 0.63
\end{align*}
\]
The results using this metric clearly indicate that, as expected, the adaptive estimation scheme is much more capable of reducing the inter-node communications for the case of poorly modeled plant dynamics. If the time period of interest is changed to be after the change in the plant dynamics only, the results are even more dramatic:

\[
M_{\text{comm.}(\text{static})} = 1.0 \\
M_{\text{comm.}(\text{adapt})} = 0.61
\]

Here the problem with the static estimation scheme under a situation of poorly modeled plant dynamics is even more obvious. The static estimator is reliant upon the accuracy of its plant model to achieve significant reductions in the required communications. In this case, the difference between the plant dynamics used to design the static estimator and the actual dynamics are large enough that communications at every sampling instant are required. The adaptive estimator, on the other hand, is capable of readjusting its parameters to improve the accuracy of its estimates. This enables the adaptive estimator to continue to provide a significant reduction in the amount of inter-node communications.
Figure 31: Comparison of Tracking Error Performances For State $X_1$
Figure 32: Comparison of Tracking Error Performances For State $X_2$
Figure 33: Comparison of Tracking Error Performances For State $X_3$
Figure 34: Comparison of Tracking Error Performances For State $X_4$
Figure 35: Comparison of Tracking Error Performances For State $X_5$
Figure 36: Comparison of Tracking Error Performances For State $X_6$
Figure 37: Comparison Of The Communication Histories For The Modified Plant
6 Conclusions

In this paper, a distributed adaptive controller was developed. The goal of the controller was to force the system states to track those of a reference model while also providing a mechanism to reduce the required amount of communication between distributed subsystems. The proposed controller utilizes an adaptive state estimation scheme and a bounded error adaptation approach to achieve these objectives. The performance of the proposed controller was demonstrated through several simulation experiments. In addition, a simulation study comparing the proposed adaptive scheme with that of a static estimation scheme for the case of poorly modeled system dynamics was presented. These results clearly demonstrate that the proposed adaptive scheme achieves a significant reduction in the required level of communications between subsystems under these conditions.
List of References


A Software Source Code

The Matlab source code developed as part of this investigation is provided here for reference.

distSimDisc.m

% distSimDisc.m  
% This script performs the required setup to run a simulation of the adaptive % distributed control approach. The guts of the simulation are in the function % "distsimdiscfunc.m" which is called by this script.
%
% define the discrete sampling period
Ts = 0.1;

global Am;
global Bm;

% parameters of the reference input sinusoids
inAmpl = [100;100;100];
inFreq = [0.1;0.1;0.1];

% load constants
load_plant_consts;
initspring3;

% define the Lyapunov matrix
global Pm;
% Pm = lyap(Am',eye(size(Am)));
Pm = P;

% used to record the adaptation flag at each time sample
aFlagsVect = ones(size(Ts:TSTOP));

% define the error thresholds as a vector
global Hth;
Hth = 0.15 .* ones(6,1);

% setup the initial vector as all zeros
XestInit = zeros(numStates + numStates + numStates*numSubSys + numStates* ...
            numSubSys + numStates + numStates^2 + numStates*numSubSys,1);
% start out assuming that we are going to adapt all gains and estimator
adaptFlags = ones(numStates,1);

% setup simulation parameters
aIndex = 1;
odeOptions = odeset('abstol',1e-4,'reltol',1e-4);
iterStart = 0;
X0 = XestInit;

% run simulation for desired time duration
[Tsim,Xsim,aFlagsVect] = distSimDiscFunc(0, tStop, Ts, X0, A, B, C, Psi, ...
P, inAmpl, inFreq, 1, Hth);

% pull off the results into separate variables
[xa,xma,khata,psihata,xhata,acihata,ghata] = extractStateslong(Xsim,6,3);

% calculate the tracking error:
ee = xa-xma;
% function [Tsim,Xsim,aFlagsVect] = distSimDiscFunc(tStart, tStop, Ts, X0, ...
% A0, B0, C0, Psi0, P0, inAmp0, inFreq0, adaptYN0, Hth0)
%
% This function performs the actual simulation of the distributed adaptive
% controller.
%
function [Tsim,Xsim,aFlagsVect] = distSimDiscFunc(tStart, tStop, Ts, X0, ...
A0, B0, C0, Psi0, P0, inAmp0, inFreq0, adaptYN0, Hth0)

global numSubSys;
global numStates;
global numSubStates;
global A;
global B;
global C;
global Am;
global Bm;
global P;
global Psi;
global inFreq;
global inAmpl;
global adaptFlags;
global Pm;
global Ahat;
global Hth;
global adaptYN;

load_plant_consts; % load constants from def file

% set parameters to the nominal values passed into this func
A = A0;
B=B0;
C=C0;
Psi=Psi0;
P=P0;
inAmpl = inAmp0;
inFreq = inFreq0;
Hth = Hth0;
adaptYN = adaptYN0;

% vectors to keep track of the adaptation flags -- init to all ones
aFlagsVect = ones(size(tStart:Ts:tStop));
adaptFlags = ones(numStates,1);

% setup simulation parameters and init counters and indexes
aIndex = 1;
odeOptions = odeset('abstol',1e-4);
simCount = 1;
iterStart = tStart;

% The actual simulation loop: simulate between discrete time steps using an ODE
% solver. The ending states from the last time step become the initial states
% for the next time step.
while (iterStart <= tStop),
    [Tpart,Xpart] = ode23(@lyap_fxdot2,[iterStart iterStart+Ts],X0,odeOptions);
    iterStart = iterStart + Ts;
    X0 = Xpart(end,:);
    Tsim(simCount) = Tpart(end);
    Xsim(simCount,:) = X0;
    aFlagsVect(simCount) = (sum(adaptFlags)~=0);
    simCount = simCount + 1;

    % give the user some idea how long we have been running
    if (mod(simCount,(50/Ts))==0)
        (simCount*Ts)
    end
end
% lyapfxdot2.m

% function [xxdot] = lyap_fxdot(t,xx)
%
% This function calculates the derivatives for the plant states,
% the adaptive gain Khat, and the adaptive estimate of the
% interconnections PsiHat. The derivatives of the state estimator
% parameters AciHat and Ghat are also calculated.
%
% This function uses an adaptive estimation scheme to limit comms
% between subsystems. The estimator is based on a Lyapunov stability
% analysis.
%
% function [xxdot] = lyap_fxdot2(t,xx);

global numSubSys;
global numStates;
global numSubStates;

global A;
global B;
global C;
global Am;
global Bm;
global P;
global Psi;

% parameters of the input sinusoids
global inFreq;
global inAmpl;

global adaptFlags;
global Pm;
global Ahat;

global Hth;
global adaptYN;

numSubStates = numStates/numSubSys;

% separate the variables from the input "state" vector
[X,Xm,Khat,PsiHat,Xhat,AciHat,Ghat] = extractStatesLong(xx,numStates,...
numSubSys);

L = length(X);
KhatMat = diag(Khat);
calc the tracking performance error:
E = X - Xm;

% update the adaptation flags for the next time through
if (sum(abs(E) > Hth) ~= 0)
    adaptFlags = ones(size(adaptFlags));
else
    adaptFlags = zeros(size(adaptFlags));
end

% dimension the matrices as appropriate
PsiHatR = reshape(PsiHat,(numSubSys*numSubStates),numSubSys)';
KhatR = reshape(Khat,numSubSys,numStates);
GhatM = reshape(Ghat,numSubSys,(numSubSys*numSubStates));
AciHatM = reshape(AciHat,numStates,numSubSys);

% update the control input
Rt = [inAmpl(1)*sin(2*pi*inFreq(1)*t); inAmpl(2)*sin(2*pi*inFreq(2)*t); ...
     inAmpl(3)*sin(2*pi*inFreq(3)*t)];
adaptFlagMask = adaptFlags;

% Use Xhat if no adaptation is taking place, otherwise use real X
if (sum(adaptFlags) == 0)
    Xss = Xhat;
else
    Xss = X;
end

U = KhatR*X - PsiHatR*Xss + Rt;

% update the plant states
Xdot = A*X + B*U + B*Psi*X;

% update reference model states
um = Rt;
Xmdot = Am*Xm + B*um;

need to blank all but block diagonal terms in this matrix!!
XmaskMat = zeros(size(XiMat));
for i = 1:numSubSys,
    for j = 1:numSubStates,
        if (adaptFlags((i-1)*numSubStates+j) == 1)
            XmaskMat(i,(i-1)*numSubStates+j) = 1;
        end
    end
end
XiMat = XiMat.*XmaskMat;

gamma = (E'*P*B);
KhatDot = -1*diag(gamma) * XiMat;
KhatDotOutV = reshape(KhatDot,numSubSys*numStates,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% update the estimates of the interconnections
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

XjMat = ones(numSubSys,1)*X';

% need to blank block diagonal terms in this matrix!!
for i = 1:numSubSys,
    XjMat(i,(i-1)*numSubStates+1:i*numSubStates) = 0;
end

% create diagonal matrix with gamma_i repeats on the diagonal
gammaMatJ = diag(gamma);
PsiHatDot = gammaMatJ*XjMat;
PsiHatDotOut = PsiHatDot;

% blank the rows that are not supposed to be updating (aFlags == 0)
checkFlags = zeros(numSubSys,1);
for i = 1:numSubSys,
    for j = 1:numSubStates,
        if (adaptFlags((i-1)*numSubStates+j) == 1)
            checkFlags(i) = 1;
        end
    end
end
PsiHatDotOut(checkFlags==0,:) = 0;
PsiHatDotOutV = reshape(PsiHatDotOut',numSubSys*numStates,1);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% update the state estimator

\[ X_{\text{HatMat}} = \text{ones}(\text{numSubSys},1) \times \text{Xhat}'; \]
% need to blank block diagonal terms in this matrix!!
for \( i = 1: \text{numSubSys} \),
\[ X_{\text{HatMat}}(i,(i-1) \times \text{numSubStates}+1:i \times \text{numSubStates}) = 0; \]
end

\[ X_{\tilde{\text{t}}l} = \text{Xhat} - \text{X}; \% \text{state estimation error} \]

% init the deriv matrices to zero
\[ \text{AciHatDotM} = \text{zeros} (\text{numStates}) ; \]
\[ \text{XhatDot} = \text{zeros} (\text{size}(\text{X})) ; \]

% update the estimate of the closed loop A matrix: \( \text{Ai} = (\text{Ai} + \text{Bi} \times \text{Ki}) \)
\[ \text{AciHatDotM} = - (\text{Xtilde}' \times \text{X}') + \text{B} \times \text{KhatDot}; \]

for \( i = 1: \text{numSubSys} \),
\[ \% \text{pull off the ith elements} \]
\[ X_{\tilde{\text{t}}l} = \text{Xtilde}(i-1) \times \text{numSubStates}+1:i \times \text{numSubStates}; \]
\[ \text{Xhati} = \text{Xhat}(i-1) \times \text{numSubStates}+1:i \times \text{numSubStates}; \]
\[ \text{Ami} = \text{Am}((i-1) \times \text{numSubStates}+1:i \times \text{numSubStates}, (i-1) \times \ldots \text{numSubStates}+1:i \times \text{numSubStates}); \]
\[ \text{Pmi} = \text{Pm}((i-1) \times \text{numSubStates}+1:i \times \text{numSubStates}, (i-1) \times \ldots \text{numSubStates}+1:i \times \text{numSubStates}); \]
\[ \text{Bi} = \text{B}((i-1) \times \text{numSubStates}+1:i \times \text{numSubStates}, i); \]
\[ \text{PsiHatDoti} = \text{PsiHatDot}(i,:); \]
\[ \text{XjMati} = \text{XjMat}(i,:); \]
\[ \text{Xi} = \text{X}(i-1) \times \text{numSubStates}+1:i \times \text{numSubStates}; \]

% calculate derivatives
\[ \text{GhatDot}(i,:) = (- \text{Xtilde}' \times \text{Pmi} \times \text{Bi} \times \text{XjMati}' - \text{PsiHatDoti}); \]

% update the estimated state vector \( \text{xj_hat} \)
\[ \text{XjHatMat}_{i} = \text{XjHatMat}(i,:); \]
if ( \text{sum} (\text{adaptFlags}) == 0 )
\[ \% \text{calc update to estimator states using actual state values} \]
\[ \text{XhatDoti} = \text{Ami} \times \text{Xhati} + (\text{AciHatM}(i-1) \times \text{numSubStates} + \ldots \]
\[ 1:i \times \text{numSubStates}, (i-1) \times \text{numSubStates} + \ldots \]
\[ 1:i \times \text{numSubStates}) - \text{Ami} \times \text{Xi} + \text{Bi} \times \text{GhatM}(i,:) \times \text{XjMati}; \]
else
\[ \% \text{calc update to estimator states using only estimated state values} \]
\[ \text{XhatDoti} = \text{Ami} \times \text{Xhati} + (\text{AciHatM}(i-1) \times \text{numSubStates} + \ldots \]
\[ 1:i \times \text{numSubStates}, (i-1) \times \text{numSubStates} + \ldots \]
end

XhatDoti = XhatDoti + Bi*Rt(i); % add in contribution from reference input
XhatDot((i-1)*numSubStates+1:i*numSubStates,1) = XhatDoti;
end

%******************************************************************************

% If adaptation is not taking place, enforce zero derivatives
if (sum(adaptFlags) == 0)
    AciHatDotM = zeros(size(AciHatDotM));
    GhatDot = zeros(size(GhatDot));
    KhatDotOutV = zeros(size(KhatDotOutV));
    PsiHatDotOutV = zeros(size(PsiHatDotOutV));
end

% user has option to just run with comms, but no adaptation
if (adaptYN == 0)
    AciHatDotM = zeros(size(AciHatDotM));
    GhatDot = zeros(size(GhatDot));
    KhatDotOutV = zeros(size(KhatDotOutV));
    PsiHatDotOutV = zeros(size(PsiHatDotOutV));
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% build output derivative vector
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

xxdot = [Xdot;Xmdot;KhatDotOutV;PsiHatDotOutV;XhatDot;...
    AciHatDotM(:);GhatDot(:)];
extractStatesLong.m

% function [X,Xm,Khat,PsiHat,Xhat,AciHat,Ghat] = extractStates(xx, ... numStates, numSubSys)
%
% This function takes the state vector output of the ODE solver ('xx') and
% extracts the individual variables from it so that they can be more easily
% manipulated.
%
function [X,Xm,Khat,PsiHat,Xhat,AciHat,Ghat] = extractStates(xx, ... numStates, numSubSys)

numSubStates = numStates/numSubSys;

% separate the variables from the input "state" vector
i = 0;
X = xx(1:numStates); % system states
i = i+numStates;
Xm = xx(i+1:i+numStates);
i = i + numStates;
Khat = xx(i+1:i+numStates*numSubSys); % adaptive gain
i = i + numStates*numSubSys;
PsiHat = xx(i+1:i+numStates*numSubSys); % estimates of interconnections
i = i+numStates*numSubSys;
Xhat = xx(i+1:i+numStates);
i = i+numStates;
AciHat = xx(i+1:i+numStates*numStates);
i = i+numStates*numStates;
Ghat = xx(i+1:i+numStates*numSubSys);
i = i+numStates*numSubSys;
loadPlantConstants.m

%% load_plant_consts.m
%%
%% This script defines constants that are used in the simulation.
%%
numSubSys = 3;
numStates = 6;
numSubStates = numStates/numSubSys;
initSpring3.m

% initSpring3.m
%
% Initializes the spring/mass/damper test system. The system contains
% three masses, 4 springs, and 4 dampers and has coupling between each
% of the masses. A control input (force) is applied to each mass.
%
% physical parameters:
% m1 = 1;
% m2 = 2;
% m3 = 4;
% b1 = 0.1;
% b2 = 0.3;
% b3 = 0.1;
% b4 = 0.2;
% k1 = 1;
% k2 = 3;
% k3 = 4;
% k4 = 3;
%mVect = [1 2 4];
bVect = [0.1 0.3 0.1 0.2];
kVect = [1 3 4 3];

% build the state space model
[A,B,C,Psi] = initspringmodel(mVect,bVect,kVect);

% call func to setup the desired reference model to track
initrefmodel;

% init the Lyapunov matrices
P1 = lyap(Amc1',100.*eye(size(Amc1)));
P2 = lyap(Amc2',100.*eye(size(Amc2)));
P3 = lyap(Amc3',100.*eye(size(Amc3)));
P = blkdiag(P1,P2,P3);
function [A,B,C,Psi] = initspringmodel(mVect,bVect,kVect)

% Initializes the spring/mass/damper test system. The system contains
% three masses, 4 springs, and 4 dampers and has coupling between each
% of the masses. A control input (force) is applied to each mass.

% Physical parameters:
ml = mVect(1);
m2 = mVect(2);
m3 = mVect(3);
b1 = bVect(1);
b2 = bVect(2);
b3 = bVect(3);
b4 = bVect(4);
k1 = kVect(1);
k2 = kVect(2);
k3 = kVect(3);
k4 = kVect(4);

% Full A matrix - block diagonal elements are in companion form
As = [0 1 0 0 0 0; -1/ml*(k1+k4) -1/ml*(b1+b4) k1/ml b1/ml k4/ml b4/ml; ... 0 0 0 1 0 0; k1/m2 b1/m2 -1/m2*(k1+k2) -1/m2*(b1+b2) k2/m2 b2/m2; ... 0 0 0 0 0 1; k4/m3 b4/m3 k2/m3 b2/m3 -1/m3*(k2+k3+k4) -1/m3*(b2+b3+b4)];

% Input and output matrices
B = [0 0 0;1/ml 0 0;0 0 0;0 1/m2 0;0 0 0;0 0 0 1/m3];
C = [1 0 0 0 0 0;0 1 0 0 0;0 0 0 0 0 1 0];

% Pull off the companion forms and the off-diagonal elements separately.
% These correspond to the local dynamics and the coupling terms, respectively.
mask = ones(2);  
maskM = blkdiag(mask,mask,mask); 
A = As .* maskM;  
Aoff = As - A;  
Psi = pinv(B)*Aoff;
% initRefModel.m
%
% This script sets up the reference model for use in simulating the
% distributed control system. The output reference model is in companion
% form.
%
pm = [-6; -7; -2; -20; -5; -10];
%pm = [-3+3*j; -3-3*j; -2; -20; -5; -10];

polyM1 = poly([pm(1) pm(2)]);
polyM2 = poly([pm(3) pm(4)]);
polyM3 = poly([pm(5) pm(6)]);

Amc1 = compan(polyM1);
Amc1 = rot90(rot90(Amc1));

Amc2 = compan(polyM2);
Amc2 = rot90(rot90(Amc2));

Amc3 = compan(polyM3);
Amc3 = rot90(rot90(Amc3));

Am = blkdiag(Amc1,Amc2,Amc3);

Bm = B;
compDeltaSpring.m

% compDeltaAspring.m
%
% This script runs a simulation of the case of poorly modeled plant dynamics.
% Two different versions of simulation are performed. In both cases, the first
% part of the simulation is identical. The second portion of the simulation is
% run with modified plant dynamics and either just comms, or with the adaptive
% process running. This allows a comparison of what happens in the static versus
% adaptive estimator cases when the plant is different from that used to design
% the estimator.
%
Ts = 0.1;

global Am;
global Bm;

% reference input params
inAmpl = [100;100;100];
inFreq = [0.1;0.1;0.1];

% nominal model parameters:
load_plant_consts;
initparams;

% length of initial portion of the simulation (the training period that is
% common to both the static and adaptive cases)
tStop0 = 300;
tStop = 300;

global Pm;
% %Pm = lyap(Am',eye(size(Am)));
Pm = P;

% define the error thresholds E0:
global Hth;
Hth = 0.025 .* ones(6,1);

% vector used to track the adaptive process -- when adapt versus not
aFlagsVect = ones(size(Ts:Ts:tStop));

% init the states to zero
XestInit = zeros(numStates + numStates + numStates*numSubSys + ...
        numStates*numSubSys + numStates + numStates^2 + ...
        numStates*numSubSys,1);
% start out assuming that we are going to adapt all gains and estimator
adaptFlags = ones(numStates,1);

%odeOptions = odeset('abstol',1e-2,'reltol',1e-2);

% run init portion of sim
% nominal parameters
mVect = [1 2 4];
bVect = [0.1 0.3 0.1 0.2];
kVect = [1 3 4 3];

[A0,B0,C0,Psi0] = initspringmodel(mVect,bVect,kVect);
initrefmodel;
X0 = XestInit;
[Tsim,Xsim,aFlagsVect] = distSimDiscFunc(0,tStop0,Ts,X0,A0,B0,C0,...
Psi0,P,inAmpl,inFreq,1,Hth);
Xsim0 = Xsim;
Tsim0 = Tsim;
disp '....Completed Initialization Sim'

% run mod A matrix with no adaption just comms
% nominal parameters
% mVect = [1 2 4];
% bVect = [0.1 0.3 0.1 0.2];
% kVect = [1 3 4 3];

% adjusted params
mVect = [1 2 1];
bVect = [0.05 0.2 0.05 0.09];
kVect = [2 5 4 3];
[A1,B1,C1,Psi1] = initspringmodel(mVect,bVect,kVect);
initrefmodel;
X0 = Xsim0(end,:);
[TsimA,XsimA,aFlagsVectA] = distSimDiscFunc(tStop0,tStop0+tStop,...
Ts,X0,A1,B1,C1,Psi1,P,inAmpl,inFreq,0,Hth);
disp '....Completed No Adaptation Sim'

% run mod A matrix with adaptation enabled
X0 = Xsim0(end,:);
[TsimB,XsimB,aFlagsVectB] = distSimDiscFunc(tStop0,tStop0+tStop,...
Ts,X0,A1,B1,C1,Psi1,P,inAmpl,inFreq,1,Hth);
% build composite output vectors:

TtotalA = [TsimO';TsimA'];
XtotalA = [XsimO;XsimA];
aFlagsVectTotalA = [aFlagsVect';aFlagsVectA'];
TtotalB = [TsimO';TsimB'];
XtotalB = [XsimO;XsimB];
aFlagsVectTotalB = [aFlagsVect';aFlagsVectB'];
This script plots the perf and comm metrics for the simulations with different threshold values $H_{th}$. It is assumed that the simulation results for all of the different threshold values already exist. The filenames for these results are specified in the code below. These results are read in and the performance and comms metrics calculated, then saved to a pair of vectors in the workspace -- "McommSweep" and "MtrackSweep".

The threshold values that are tested -- it is assumed that these agree with the simulations that have actually been run.

$H_{thSweep} = [0.01 0.025 0.05 0.08 0.1 0.15]$;

```matlab
load 'nomPlant_RealAm_HighFreqIn_Hth0p01'
[c,p] = calcMetrics(Xsim,Tsim,aFlagsVect);
load 'HthSweepResults' McommSweep MtrackSweep HthSweep
McommSweep(1) = c;
MtrackSweep(1) = p;
save 'HthSweepResults' McommSweep MtrackSweep HthSweep

load 'nomPlant_RealAm_HighFreqIn_Hth0p025'
[c,p] = calcMetrics(Xsim,Tsim,aFlagsVect);
load 'HthSweepResults' McommSweep MtrackSweep HthSweep
McommSweep(2) = c;
MtrackSweep(2) = p;
save 'HthSweepResults' McommSweep MtrackSweep HthSweep

load 'nomPlant_RealAm_HighFreqIn_Hth0p05'
[c,p] = calcMetrics(Xsim,Tsim,aFlagsVect);
load 'HthSweepResults' McommSweep MtrackSweep HthSweep
McommSweep(3) = c;
MtrackSweep(3) = p;
save 'HthSweepResults' McommSweep MtrackSweep HthSweep

load 'nomPlant_RealAm_HighFreqIn_Hth0p08'
[c,p] = calcMetrics(Xsim,Tsim,aFlagsVect);
load 'HthSweepResults' McommSweep MtrackSweep HthSweep
McommSweep(4) = c;
MtrackSweep(4) = p;
save 'HthSweepResults' McommSweep MtrackSweep HthSweep

load 'nomPlant_RealAm_HighFreqIn_Hth0p1'
[c,p] = calcMetrics(Xsim,Tsim,aFlagsVect);
load 'HthSweepResults' McommSweep MtrackSweep HthSweep
```
McommSweep(5) = c;
MtrackSweep(5) = p;
save 'HthSweepResults' McommSweep MtrackSweep HthSweep

load 'nomPlant_RealAm_HighFreqIn_Hth0p15'
[c,p] = calcMetrics(Xsim,Tsim,aFlagsVect);
load 'HthSweepResults' McommSweep MtrackSweep HthSweep
McommSweep(6) = c;
MtrackSweep(6) = p;
save 'HthSweepResults' McommSweep MtrackSweep HthSweep

% plot the results
figure;plot(HthSweep,McommSweep,'k*')
title 'Communications Metric For Various Error Thresholds'
xlabel 'Threshold Value E_0~i'
ylabel 'Communications Metric'

figure;plot(HthSweep,MtrackSweep,'k*')
title 'Performance Metric For Various Error Thresholds'
xlabel 'Threshold Value E_0~i'
ylabel 'Performance Metric'
calcMetrics.m

function [Mcomm,Mperf] = calcMetrics(Xsim,Tsim,aFlagsVect)

This function calculates the performance (state tracking) and communications metrics.

function [Mcomm,Mperf] = calcMetrics(Xsim,Tsim,aFlagsVect)

[xa,xma,khata,psihata,xhata,acihata,ghata] = extractStateVectslong(Xsim,6,3);

ee = xa-xma;
em = xa-xhata;

Mcomm = mean(aFlagsVect);
Mperf = sum(max(abs(ee')));