Distributed Control of Electric Drives via Ethernet

Lilantha Samaranayake

ROYAL INSTITUTE OF TECHNOLOGY
DEPARTMENT OF ELECTRICAL ENGINEERING
ELECTRICAL MACHINES AND POWER ELECTRONICS

Stockholm 2003
Abstract

This report presents the work carried out aiming towards distributed control of electric drives through a network communication medium with temporal constraints, i.e, Ethernet. A general analysis on time delayed systems is carried out, using state space representation of systems in the discrete time domain. The effect of input time delays is identified and is used in the preceding controller designs.

The main hardware application focused in this study is a Brushless DC servo motor, whose speed control loop is closed via a 10 Mbps Switched Ethernet network. The speed control loop, which is approximately a decade slower than the current control loop, is opened and interfaced to the network at the sensor/actuator node. It is closed at the speed controller end at another node in the same local area network (LAN) forming a distributed control system (DCS).

The Proportional Integral (PI) classical controller design technique with ample changes in parameter tuning suitable for time delayed systems is used. Then the standard Smith Predictor is tested, modified with the algebraic design technique Coefficient Diagram Method (CDM), which increases the system degrees of freedom. Constant control delay is assumed in the latter designs despite the slight stochastic nature in the timing data observations. Hence the poor transient performance of the system is the price for the robustness inherited to the speed controllers at the design stage. The controllability and observability of the DCS may be lost, depending on the range in which the control delay is varying. However a state feedback controller
deploying on-line delay data, obtained by means of synchronizing the sensor node and controller node system clocks, results in an effective compensation scheme for the network induced delays. Hence the full state feedback controller makes the distributed system transient performance acceptable for servo applications with the help of pole placement controller design.

Further, speed synchronizing controllers have been designed such that a speed fluctuation caused by a mechanical load torque disturbance on one motor is followed effectively by any other specified motor in the distributed control network with a minimum tracking or synchronizing error. This type of performance is often demanded in many industrial applications such as printing, paper, bagging, pick and place and material cutting.

**Keywords**

- Brushless DC Motor
- Control Delay
- Distributed Motion Control Systems
- Proportional Integral Controller
- Smith Predictor
- Speed Synchronization
- State Feedback Controller
- Stochastic Systems
- Switched-Ethernet
- Synchronizing Error
- Time Delayed Systems
- Tracking Error
Acknowledgments

First, I would like to thank my main supervisor Prof. Chandur Sadarangani, for giving me the opportunity to pursue my post graduate studies at the division of Electrical Machines and Power Electronics and for his encouragement to carry on this work.

My special thank goes to Dr. Sanath Alahakoon and Tech. Lic. Mats Leksell, my co-supervisors, for their relentless discussions on project matters as well as the other commitments paid towards the project activities, and also for proof reading this report. My sincere thanks goes to Dr. Juliette Soulard for providing me home in the Permanent Magnet Drives (PMD) program within the division. Constructive project discussions with Dr. Kirthi Walgama, during my stay in the University of Peradeniya (UP) Sri Lanka, are also highly appreciated.

The opportunities given for me to explore the jungle of information technology under the guidance of Mats Westerborn and Stefan Sundquist of IT university, Kista are remembered gratefully. I must also thank Prof. Martin Törngren for introducing me to the domain of Real-Time control systems at the Dept. of Machine Design, Royal Institute of Technology (KTH).

The financial support provided by the Swedish International Development cooperation Agency (SIDA) under the “Research capacity building in Electrical and Electronic Engineering in the University of Peradeniya”, in collaboration between the Dept. of Electrical Engineering (ETS), KTH, Sweden and the Dept. of Electrical and Electronic Engineering (DEEE), UP, Sri Lanka is gratefully acknowledged.
I would like to thank all staff members of EME and DEEE for creating a friendly working environment and their support in all aspects.

I pay my earnest gratitude to my dear parents, Mr. & Mrs. B. G. Samaranayake, for their love, care and sacrifices, which made it possible for me to pursue with higher studies. And also to my sister Lalani and brother Malinga and all my teachers and friends for encouraging me to continue post graduate studies. At last but not least, to my wife Upeka for her patience and understanding throughout.


Lilantha Samaranayake
# Contents

1 Introduction .............................................. 1  
   1.1 Background ........................................... 1  
   1.2 Outline .................................................. 3  

2 Distributed Control Systems .......................... 5  
   2.1 Communication in DCS ................................. 6  
      2.1.1 FIP (Factory Instrumentation Protocol) .......... 7  
      2.1.2 PROFIBUS (Process Fieldbus) .................... 7  
      2.1.3 CAN (Controller Area Network) .................. 7  
   2.2 Ethernet ............................................... 8  
      2.2.1 Ethernet model ..................................... 10  
      2.2.2 CSMA/CD access control .......................... 12  
      2.2.3 Half-duplex ......................................... 13  
      2.2.4 Full-duplex ......................................... 14  
      2.2.5 Ethernet Switch .................................... 14  
   2.3 Ethernet for real-time applications: A survey ...... 17  
      2.3.1 RETHER ............................................. 17  
      2.3.2 EtheReal ........................................... 18  
      2.3.3 TEMPRA ............................................. 19  
      2.3.4 SIRTE .............................................. 19  
      2.3.5 IPv6 ............................................... 20  

3 Problem Formulation .................................. 23  
   3.1 Brushless DC motor .................................... 23  
   3.2 Local servo amplifier .................................. 25  
   3.3 Controller .............................................. 26  

Contents

3.4 Network .................................................. 26
3.5 Time delayed system ...................................... 26
  3.5.1 Overview .......................................... 26
  3.5.2 Open loop model .................................. 29
  3.5.3 Design criteria .................................... 32

4 Analysis of time delayed systems ......................... 33
  4.1 Timing Analysis ....................................... 33
    4.1.1 Discrete Time Control System ................... 36
    4.1.2 Effect of time delays ............................ 38
    4.1.3 Simulation results ............................... 39
  4.2 Controllability and Observability .................... 40
  4.3 Delay model .......................................... 43

5 Controller design for the time delayed system .......... 45
  5.1 PI controller ......................................... 45
    5.1.1 Discretization .................................... 46
    5.1.2 Integrator Wind-Up problem ....................... 47
    5.1.3 Stability Problem .................................. 48
  5.2 Smith Predictor ........................................ 49
    5.2.1 Coefficient Diagram Method (CDM) ............... 51
    5.2.2 Design procedure .................................. 52
    5.2.3 Design for the test rig ........................... 53
  5.3 State Feedback Controller ............................ 54
    5.3.1 Analysis .......................................... 54
    5.3.2 Timing data acquisition .......................... 56
    5.3.3 Clock synchronization ............................. 57
    5.3.4 Time stamping .................................... 58
    5.3.5 State Feedback Gain ............................... 59
    5.3.6 Estimator Design .................................. 65
    5.3.7 Reference Gain .................................... 67
  5.4 Results .............................................. 68

6 Speed Synchronization .................................. 71
  6.1 Introduction .......................................... 71
  6.2 Application Aspects ................................... 72
# Contents

6.2.1 Offset Printing ........................................... 72
6.2.2 Horizontal Bagging ........................................ 72
6.2.3 Cut to Length Rotary Knife ............................... 73
6.2.4 Pick and Place ........................................... 75
6.3 Approach .................................................... 76
6.3.1 Master Slave ............................................. 76
6.3.2 Interactive Feedback Control ............................ 77
6.3.3 Synchronization in distributed systems .................. 79
6.4 Reference Feedback Control ................................. 82

7 Conclusions and future work ................................. 87
7.1 Conclusions .................................................. 87
7.2 Future work ................................................ 89

References ..................................................... 91

List of symbols ................................................. 97
Contents
1 Introduction

1.1 Background

Feedback control systems wherein the control loops are closed through a network are called distributed control systems (DCS) [1], [2]. The defining feature of a DCS is that the information (reference input, plant output, control input, etc.) is exchanged using a communication network running over the main control system modules (sensors, controllers, actuators, etc.). Moving one step further, the system becomes truly distributed, when parts of the accessible inner control loops are handed over to the actuator. For example in electrical motors, the fast inner current control loop can be closed in the vicinity of the actuator while the slower speed control loop can be closed through a network connected speed controller. The basic advantage of DCS is its capability of controlling many such systems from a single location at a lower installation and maintenance cost.

However, the insertion of the communication network in the feedback control loop complicates the analysis and design of the DCS. Conventional control theories with many ideal assumptions, such as synchronized control, non delayed sensing and actuation, must be re-evaluated. Specifically the following issues need to be addressed.

Firstly the network induced delays, mainly the sensor to controller delay ($\tau_{sc}$) and the controller to actuator delay ($\tau_{ca}$) that occur while exchanging data among devices connected to the network. These delays, either constant
or time varying [3], can degrade or even destabilize the performance of DCS unless they are considered in the design [26]. Secondly, the network can be viewed as a web of unreliable transmission paths. Hence some data packets can get lost completely in the transmission. Thirdly due to the bandwidth and packet size constraints of the network, the data may be transmitted using multiple data packets [4]. Therefore depending on the method of media access, the number of active nodes operating, the available bandwidth etc., chances are that all, part or none of the packets could arrive their destinations on time. Thus the behavior of a DCS is largely dependent upon the performance parameters of the underlying network. DeviceNet, CAN and Ethernet are such networks to name and focus in this study is on Ethernet.

In Ethernet, the packet drop out problem can be solved by using connection oriented TCP sockets in the TCP/IP protocols of the transport and network layers [4] respectively. The chances that multiple packets occur are avoided by fixing the size of data packets to frames of 64 bytes, which is the minimum size, in both directions. The state of the art of handling media access control problem is to make the communication full-duplex by connecting the nodes through an Ethernet Switch.

The remaining network induced delay problem can be addressed in two different ways. One way is to design the control system regardless of the packet delays and design a communication protocol that minimizes the likelihood of such delays. For example, various congestion control and avoidance algorithms have been proposed in [5], [6] to gain better performance when the network traffic reaches its critical upper peak. The other approach is to treat the network protocol and its traffic as given conditions and design control strategies that explicitly take the delay issue into account. To handle the delay, one might formulate control strategies based on Kalman filter predictors as in [7] or Linear Quadratic Gaussian (LQG) methods when the delay is governed by underlying Markov chains as in [1], [3] and [8]. Here the approach is to use on-line delay data to compensate the delay using state feedback control.
1.2 Outline

This report has been organized as follows.

Chapter 2 describes the elements of DCS and formulates the time delay problem. Further it summarizes the existing DCSs formed using field-bus systems and revises the literature study on Ethernet and the attempts to use it for real-time applications.

Chapter 3 introduces the specific problem concerned in the report with a description of the physical test rig together with the derivation of its control system model.

Chapter 4 analyzes the impact of time delays on the closed loop operation of the system and is extended to test the controllability and observability at different levels of time delays.

Chapter 5 deals with different speed controller architectures for the time delayed system. First the classical methods are summarized. Then the method of state feedback with on-line delay compensation is introduced.

Chapter 6 introduces speed synchronizing algorithms to be used in reflecting the impact of a torque disturbance applied on one DCS to another that is running parallel.

Chapter 7 presents the conclusions of the work and proposes future work.

Some of the results presented in this report have been published in the following conference papers,

1. “Real-Time Speed Control of a Brushless DC Motor via Ethernet”,
   L. Samaranayake, M. Leksell, S. Alahakoon,
2. “Closed - loop Speed Control of a Brushless DC Motor via Ethernet”,
   L. Samaranayake, S. Alahakoon,

3. “Speed Synchronized Control of Permanent Magnet Synchronous Machine Drive Systems with Field Weakening”,
   L. Samaranayake, Y. K. Chin,
   Accepted for: 4th International Power Electronics and Motion Control Conference, August 12-14 2003, Xi’an, China.

4. “Speed Controller Strategies for Distributed Motion Control Via Ethernet”,
   L. Samaranayake, S. Alahakoon, K. S. Walgama,
   Accepted for: 18th IEEE International Symposium on Intelligent Control, October 5-8 2003, Texas Houston, USA.

5. “State-Feedback Controller for Distributed Systems with Non-Deterministic Time Delays”,
   L. Samaranayake, M. Leksell, S. Alahakoon,
   Accepted for: 18th IEEE International Symposium on Intelligent Control, October 5-8 2003, Texas Houston, USA.

6. “Distributed Control of Multiple PMSM direct drive systems with a Single Speed Transmission for Electric Vehicle Propulsion”
   L. Samaranayake, Y. K. Chin,
   Accepted for: 20th International Electric Vehicle Symposium and Exposition, November 15-19 2003, San Francisco, USA.
2 Distributed Control Systems

A computer controlled system mainly comprises of three modules, i.e., sensor, controller and actuator, as shown in Figure 2.1. Each of these subsystem modules can physically be divided into separate units. The collective goal of these three subsystems working together is to perform some feedback control action on the plant to be controlled. The general functionality of a computer controlled system can be described as follows: firstly, the sensor subsystem collects data from the plant to be controlled. Secondly, the controller subsystem, by means of a control law, processes this data and derives the control action needed. Finally, the actuator subsystem performs the action implied by the controller on the plant.

Distributed Control Systems (DCS) add the functionality of a communication network by inserting one in between the sensor-controller nodes.
and the controller-actuator nodes of the computer controlled system. Despite various problems mentioned earlier in Chapter 1 due to uncertainties in the network, there are many advantages of this insertion compared to non-distributed systems. Reduced wiring and power requirements, flexibility of operations, evolutionary design process and ease of maintenance, diagnostics and monitoring are some of them to mention [9].

2.1 Communication in DCS

Communication networks were introduced to digital control systems in the 1970s. At that time the driving force was the car industry. The motives for introducing communication networks were reduced cost for cabling, modularization of systems, and flexibility in system setup. Since then, several types of communication networks have been developed. Communication protocols can be grouped into fieldbuses (e.g. FIP and PROFIBUS), automotive buses (e.g. CAN), ‘other’ machine buses (e.g 1553B and the IEC train communication network), general purpose networks (e.g. IEEE LANs and ATM LANs) and a number of research protocols (e.g. TTP) [20].

Fieldbuses are intended for real-time control applications, but in some applications other networks may have to be used for control. For instance, if another network is already in use at high level managerial and technical hierarchies, it would be cost effective to use the same network for low level plant control too. This is one of the motivations to experiment standard Ethernet for real-time applications.

The fieldbuses are usually only made for connection of low-level devices such as sensors, actuators. If a high-level function such as the connection of a PC is to be realized, the other networks may be more suitable. There is a vast number of communication protocols and fieldbuses used for low-level communication. Following is a short summary of some of them.
2.1 Communication in DCS

2.1.1 FIP (Factory Instrumentation Protocol)

\( FIP^{TM} \) is developed by a group of French, German, and Italian companies. FIP uses a twisted pair conductor for communication and the transmission speeds are from 31.25 kbps (kilo bits per second) up to 2.5 Mbps (mega bits per second), depending on the spatial dimensions of the bus. For a transmission speed of 1 Mbps, the maximum length of the bus is 500 m. The maximum number of nodes in a FIP network is 256, where one node acts as the bus arbitrator. The bus arbitrator cyclically polls all nodes in the network to broadcast its data on the network. The inactive nodes listen to the communication and recognize when data of interest to the node is sent. The FIP network can be seen as a distributed database, where the database is updated periodically.

2.1.2 PROFIBUS (Process Fieldbus)

\( PROFIBUS^{TM} \) is developed by a group of German companies and is now a German standard. A screened twisted pair is used as the communication bus. The transfer speed can be from 9.6 kbps to 500 kbps. The maximum length of the bus is 1200 m and up to 127 stations can be connected to the network. PROFIBUS messages can be up to 256 bytes long and it is a token-passing network. The nodes are divided into active and passive nodes. The node which holds the token has the permission to send data on the network. The token is passed around in the network between the active nodes. Active nodes can transmit when they hold the token. Passive nodes need to be addressed by an active node to be allowed to send data on the network.

2.1.3 CAN (Controller Area Network)

\( CAN^{TM} \) is developed by the German company BOSCH\(^{TM} \) for the automation industry. CAN is one of the first fieldbuses and is now used in cars by several manufacturers. The transfer speed on the bus can be programmed.
2 Distributed Control Systems

<table>
<thead>
<tr>
<th>Preamble</th>
<th>SFD</th>
<th>DA</th>
<th>SA</th>
<th>Length</th>
<th>LLC Header</th>
<th>Data</th>
<th>FCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bytes</td>
<td>7</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>8</td>
<td>38−1492</td>
</tr>
</tbody>
</table>

**Figure 2.2:** IEEE 802.3 frame

The transfer speed can be 1 Mbps if the bus is shorter than 50 m, and 500 kbps if the bus is longer than 50 m. If the cable quality is low, as it can be in mass produced cars, the maximum transfer speed may be even lower. There is no limit on the number of nodes. A node can start transmitting at any time if the bus is silent. If several nodes are trying to transmit, an arbitration process starts. The node trying to send the message with highest priority gets the right to use the bus. There are 229 different priority levels for messages.

### 2.2 Ethernet

The term Ethernet is used to refer to a specification published in 1982 collectively by Digital Equipment Corp., Intel Corp. and Xerox Corp. in USA. The original Ethernet network operates at 10 Mbps and uses an access method called CSMA/CD, which stands for Carrier Sense Multiple Access with Collision Detection. A few years later the IEEE 802 Committee published a slightly different set of standards. The standard IEEE 802.3 covers the CSMA/CD networks, IEEE 802.4 covers token bus networks and IEEE 802.5 covers token ring networks. Common to all these three standards is the IEEE 802.2 standard that defines the logical link control (LLC). Most of the Ethernet networks today follow the IEEE 802.3 standard, but the original Ethernet frame format is usually used instead of the IEEE 802.3 frame format [13]. Figures 2.2 and 2.3 show the respective frame formats, where an introduction on each item in the frames is given next.
2.2 Ethernet

<table>
<thead>
<tr>
<th>Preamble</th>
<th>SFD</th>
<th>DA</th>
<th>SA</th>
<th>Type</th>
<th>Data</th>
<th>FCS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bytes</td>
<td>7</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>46−1500</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.3: Ethernet frame

- **Preamble** - A 7 bytes pattern of alternating 1s and 0s used by the receiver to establish bit synchronization.

- **Start Frame Delimiter (SFD)** - The bit sequence 10101011, which indicates the actual start of the frame and enables the receiver to locate the first bit of the rest of the frame.

- **Destination Address (DA)** - Specifies the station(s) for which the frame is intended. It may be a unique physical address, a group address or a global address.

- **Source Address (SA)** - Specifies the station that sends the frame.

- **Length** - Length of LLC header and data field in bytes. (Only for IEEE 802.3 frame)

- **Type** - Ethernet type field for identifying the contents of the data field. (This field is included in the LLC header for IEEE 802.3.)

- **LLC header** - Logical Link Control header, i.e. IEEE 802.2 protocol.

- **Data** - The data to send is usually an IP datagram. This field has a minimum size and has to be padded if it is shorter.

- **Frame Check Sequence (FCS)** - A 32 bit cyclic redundancy check, based on all fields except Preamble, SFD and FCS.

The length of both frames, excluding Preamble and SFD, is between 64 and 1518 bytes. The minimum length of a frame is 64 bytes. Inter Frame Gap (IFG) is the minimum time between two frames and this time depends on the transmission speed. For a 10 Mbps LAN it is 9.6 $\mu$s and for a 100 Mbps LAN it is 0.96 $\mu$s as it is defined as multiples of 96 bits.
2.2.1 Ethernet model

OSI reference model

The Open System Interconnection (OSI) reference model was developed by the International Organization for Standardization (ISO). The final standard, ISO 7498, was published in 1984. The model consists of seven layers and these layers are described briefly in the following [14], [15]:

- **Application layer** - Provides access to the OSI environment for users and also provides distributed information services.

- **Presentation layer** - Provides independence to the application processes from differences in data representation.

- **Session layer** - Provides the control structure for communication between applications. It establishes, manages, and terminates connections between cooperating applications.

- **Transport layer** - Provides reliable and transparent transfer of data between end points. Further it provides end to end error recovery and flow control.

- **Network layer** - Provides the upper layers with independence from the data transmission and switching technologies used to connect the systems. Also it is responsible for establishing, maintaining, and terminating connections.

- **Data link layer** - Provides the reliable transfer of information across the physical link, sends frames with the necessary synchronization, error control and flow control.

- **Physical layer** - Concerned with the transmission of unstructured bit streams over physical media. It deals with the mechanical, electrical, functional, and procedural characteristics to access the physical medium.
The layers are usually referred with numbers, starting from the physical layer as layer one. Every layer encapsulates the data in a protocol. The encapsulation is usually done by adding a header to the data. In the data link layer, the header covers the tail of the data as well. The physical layer is slightly different, the protocol instead specifies a set of rules on the physical interface with the following specifications:

- **Mechanical**: Specifies the pluggable connectors, signal conductors and the wiring scheme.

- **Electrical**: Specifies the representation of bit values and transmission rates.

- **Functional**: Specifies the functions performed between the physical interface and the transmission media.

- **Procedural**: Specifies the sequence of events by which bit streams are exchanged across the physical medium.

**TCP/IP protocol suite**

The TCP/IP protocol suite is a result of protocol research and development conducted on the experimental network, ARPANET, funded by the Defense Advanced Research Project Agency (DARPA), USA. The work started in the late 1960s and has become the most widely used protocol suite for network communication. There is no official TCP/IP protocol model, as in the case of OSI. However, based on the protocol standards that have been developed, it is possible to organize the communication tasks into five relatively independent layers as follows:

- **Application layer** - Provides communication between processes or applications on separate hosts. File Transfer Protocol (FTP), Hyper Text Transfer Protocol (HTTP) and Telnet are among the main protocols used here.
- **Transport layer** - Provides end to end data transfer service. This layer may include reliability mechanisms. It hides the details of the underlying network or networks from the application layer. The main protocols used are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

- **Internet layer** - Concerned with the routing of data from the source host to the destination host on one or more networks connected by routers. The protocol used here is called Internet Protocol (IP).

- **Network access layer** - Concerned with the logical interface between the end system host and the network. Carrier Sense Multiple Access with Collision Detection (CSMA/CD) is the Media Access Protocol (MAC) used.

- **Physical layer** - Defines the characteristics of the transmission medium, the signaling rate and the signal encoding scheme.

A comparison of the OSI reference model with the TCP/IP model is shown in Figure 2.4.

### 2.2.2 CSMA/CD access control

The devices communicate in half-duplex, when using Carrier Sense Multiple Access with Collision Detection. The CSMA/CD is an improvement of the CSMA access control technique [13]. The difference is that in CSMA/CD, the station continues to listen to the medium while transmitting. This leads to the following rules for CSMA/CD:

1. If the medium is idle, start to transmit, otherwise go to Step 2.

2. If the medium is busy, continue listening until the channel is idle, then start to transmit immediately.
3. If a collision among nodes is detected during transmission, transmit a brief jamming signal to assure that all stations know that there has been a collision and therefore to cease transmission.

4. After transmitting the jamming signal, wait for a random time slot and then attempt to transmit again.

The segment length of the network has a maximum value to ensure that all devices detect a collision. The IEEE 802.3 protocol standard specifies this value for different physical layer media. Hence CSMA/CD gives rise to time delays, which are non-deterministic, in the Ethernet network.

### 2.2.3 Half-duplex

For a Ethernet network with shared medium, the nodes must use the CSMA/CD access control. This means that only one node is allowed to
transmit at a time. Examples of network topologies that have shared medium are the bus topology, and the star topology interconnected with a hub [14].

2.2.4 Full-duplex

This includes a network with star topology interconnected with an Ethernet Switch. Full-duplex means that transmission can be made simultaneously in both directions. This also means that the sending node does not have to sense that the medium is idle before transmitting. Hence the media access delay can be avoided.

IEEE 802.3 defines a flow control for full-duplex Ethernet, namely the media access control (MAC) PAUSE. When a receiving device detects an upcoming buffer overflow, it will transmit a PAUSE control frame to the sender, requesting it to stop transmission for a certain period of time. The time is expressed as a multiple of 512 bit-times, which for a 10 Mbps LAN is equal to 51.2μs, for a 100 Mbps LAN is equal to 5.12μs and so on. If sufficient buffers will become free in the meantime, the receiver can re-admit transmission by sending a PAUSE control frame with the pause duration parameter zero to the sender. Usually the PAUSE control frames are used to turn transmission on and off, because it is difficult to calculate an appropriate PAUSE timeout. The PAUSE control frame is not forwarded through Ethernet Switches.

2.2.5 Ethernet Switch

An Ethernet Switch is a versatile hardware device to partition the available network bandwidth into dedicated pieces without infringing the communication ability of the devices connected to it. Hence it allows full-duplex communication without collisions. Forwarding decisions are made on the Datalink layer of the OSI model as shown in Figure 2.5. The address learning function is usually updated automatically. Unlike the hub, the switch
2.2 Ethernet

There are two basic transmission methods:

- **Cut through** - Switch starts sending packets as soon as they enter it and their destination address is read within the first 20-30 bytes of the frame. Thus the packets are forwarded to the destination, before they are received completely. This reduces transmission latency between ports, but it can propagate bad packets.

- **Store and forward** - Buffers incoming packets in the switch memory until they are fully received and a cyclic redundancy check (CRC) is run [13]. The buffered memory adds latency to the processing time, which is proportional to the frame size. The store and forward switching reduces bad packets and collisions that can adversely effect the overall...
performance of the segment. These switches must buffer the frame, until its re-transmission is commenced. The common approaches are:

- Input buffering - One buffer per input port.
- Output buffering - One buffer per output port.
- Internal buffering - One memory pool shared by all ports.

The third approach has currently become the most popular despite the price due to its higher memory utilization. However, cheap switches may still use output buffering. The switch can use one of the two transmission methods or possibly a mixture of both. The advantages of a switch over a hub are:

- Every port on the switch has its own collision domain.
- Since a port of the switch operates in full-duplex, it can receive and transmit simultaneously as two ports.

An Ethernet network formed by connecting the stations (hosts) through Ethernet Switches is called a Switched Ethernet network. With remarkable speeds like 10 Mbps, 100 Mbps or even 1000 Mbps, Ethernet is one of the most widely used local area network (LAN) technologies. But it is not intended for real-time communications. However, the large number of installed Ethernet networks will make it attractive to use in DCSs with the real-time communication capability. Because all the other means of communications used in DCSs are commonly in need of dedicated hardware. As a result they inherit incompatibility among different bus systems and demand a high capital cost. On the other hand there exist cheap single chip solutions that integrate Ethernet and TCP/IP stack. Furthermore, this integration can be extended straight away to Internet. With the TCP, it is possible to use all the conventional protocols such as FTP, HTTP, Telnet etc. Ethernet already supports future requirements in bandwidth. As everybody will be using the same protocol set, the solution is universal and thus compatibility issues will be less. Vendors will only have to support a single platform instead of multiple fieldbuses. At the same time the capacity of existing fieldbuses are reaching their limits. Therefore it is unlikely that they would full-fill the
future needs of bandwidth [10]. Therefore it is worth investigating Ethernet for lower level real-time applications.

2.3 Ethernet for real-time applications : A survey

This section summarizes the state of the art of adapting Ethernet in real-time applications. Real-time means not only the reliable communication but also its punctuality. It is vital to distinguish hard real-time from soft real-time as they are significantly different. In hard real-time systems, the messages must be delivered in a predefined time (deadline). Otherwise the contents of the message become useless and the end results would lead to disagreeable consequences, e.g. a system leading to instability. On the other hand in soft real-time systems, like in multimedia applications, the consequences of a late or missing packet is rather small, for example a pause in the music or a flicker in the picture. Some of the efforts made in the literature to promote Ethernet in hard/soft real-time applications are summarized below.

2.3.1 RETHER

The most interesting aspect of RETHER is its unique capability of supporting real-time bandwidth reservation and guarantee over existing Ethernet infrastructure without any hardware support or modification. The only modification required is the replacement of Ethernet driver with a RETHER driver on every machine in the network [16]. But there has been many practical deficiencies found in RETHER. Firstly it is difficult to expose RETHER’s real-time connection abstraction to user level applications without modifying the operating system of the host. Secondly, installation of RETHER drivers on mass scale is an expensive proposition from a system administration standpoint. Thirdly, the token passing mechanism implemented in RETHER is less appealing to compete in today’s Switched Ethernet LAN environment. Therefore it is no longer considered as an effective solution.
2.3.2 EtheReal

The goal of the EtheReal is to build a scalable real-time Ethernet switch that supports bandwidth reservation and guarantee over a switched LAN environment without any hardware or operating system modification. With the comparatively low prices of Switches, a star topology as shown in Figure 2.6 becomes a more realistic alternative over a token-timed based solution as RETHER [15]. The EtheReal switch also offers buffers, which are used to make priority features available. The main idea of the proposal is to have bandwidth guarantee support (mainly for multimedia) in a star-topology Switched Ethernet network. Every node is connected to other nodes via a switch, modified to support real-time traffic. A Real-Time Communication Daemon (RTCD) has to be installed in each node to support real-time traffic consisting only of software. A RTCD is software add-on for the operating system which schedules the traffic. The significant feature of EtheReal over the predefined bandwidth guarantee is its soft real-time traffic support with priority scheduling. EtheReal supports non-real-time traffic as well [16].
2.3 Ethernet for real-time applications: A survey

2.3.3 TEMPRA

This project focuses on how to implement time determinism in an Ethernet network. This solution consists of nodes connected to a single bus, i.e. a shared-medium topology. In one end of the bus, a TEMPRA bridge is installed as shown in Figure 2.7 to schedule the network traffic. The idea of the TEMPRA solution is based on a master-slave solution, where the TEMPRA bridge at the end of the bus is the master. When the bus is idle, the bridge transmits a synchronization pulse. Every node on the bus waits for this synchronization pulse and a timeout starts when a node receives this pulse. When the timeout reaches the expire limit, the node is allowed to transmit on the medium. Setting the timeout sets the priority of the respective node. For instance a lower timeout means a higher priority. This eliminates collisions in the underlying Ethernet network. But this solution requires a hardware reconfigured Ethernet card that supports TEMPRA protocol. Furthermore, modifications in the host operating system are also necessary.

2.3.4 SIRTE

Switched Industrial Real-Time Ethernet (SIRTE) is based on a 100 Mbps Ethernet switch supporting hard real-time traffic with bandwidth and latency guarantee, i.e. a predefined amount of data is delivered before a dead-
line. The guarantee is upheld by real-time channels, which can be created and closed dynamically. The nodes, which are connected to the switch, can either be real-time nodes or non real-time nodes depending on which quality of service (QoS) level is required. Nodes with no SIRTE drivers installed are non real-time. In the proposed solution there are no modifications of the Ethernet hardware on the network interface cards. The real-time communication support in the nodes and the switch are handled by a software added between the Network layer and the Data link layer (Middle-ware), in the OSI reference model. However, some additional hardware to the standard Ethernet switch are needed. The switch has the overall responsibility for establishing, maintaining and terminating different node connections through it. The software in the end nodes includes the application program interface (API) and the simple regulations of the traffic passing through the protocol suite. The product is not yet commercially available [17].

2.3.5 IPv6

IPv6 or Internet Protocol version 6, was recommended by the Internet Engineering Task Force (IETF) in 1994 for the Internet layer of the TCP/IP protocol suite. In 1998, the core set of IPv6 protocols became an IETF drafted standard. IPv6 is a new version of IP that is designed to be an evolutionary step from IPv4, the current version. It is supposed to fix several problems now being encountered with IPv4 in the context of DCSs, such as auto-configuration and support for real-time traffic. In addition, the revised IP header with 128 bits address field gives an unlimited number of global addresses [18]. The most significant features are further described below.

**Auto-configuration**

In IPv4, the network addresses need to be set-up manually or by using Dynamic Host Configuration Protocol (DHCP). In contrast, IPv6 allows auto-configuration of addresses, thus the network management is greatly simplified. The self-generated IPv6 address combines a local network identifier
and an identifier generated by the node in the “plug-and-play” fashion. By simplifying the maintenance of large networks of simple devices, such as sensors and actuators of DCSs, this automated procedure paves the way for the large scale introduction of Ethernet technology into industrial environments.

Support for real-time traffic

The IPv6 header contains two fields specifically targeted at real-time applications, the “Traffic Class” and the “Flow Label”. The Traffic Class is an 8-bit field that enables switches to distinguish real-time packets from non-real-time packets. The Flow Label is a 20-bit label identifying flows. The IPv6 specification describes a flow as a sequence of packets sent from a particular source to a particular destination. This will be an important feature for the DCSs in defining application specific sensor/actuator and controller nodes.

Performance

Performance comparison presented in [19] reveals that IPv4 is slightly faster than IPv6 on a very simple back to back connected network with software implemented TCP/IP stacks. The reason pointed out is the fact that the IPv6 solution is a recent product, where as the IPv4 stack has undergone a number of performance enhancements over the past 20 years. When the network is expanded by adding a router, again IPv6 becomes much slower. Still IPv6 can not be rejected completely as the current networking hardware is not optimized for IPv4. Thus no complete conclusions can be drawn on the performance until tests are conducted on hardware optimized for IPv6.
3 Problem Formulation

This chapter starts off with a component description of the DCS to be used in the report. It is then extended to establish the complete DCS followed by an onion layer task analysis. Obtaining the system model together with the desired performance criteria is also presented. The test rig mainly consists of a Brushless DC Motor, a local servo amplifier, an Ethernet network and a software controller implemented in a PC. Figure 3.1 shows the physical layout of the test rig and a short description of each item follows.

3.1 Brushless DC motor

The Brushless DC motor considered is a 0.4 kW, 4000 rpm, non-salient permanent magnet (PM) synchronous machine with trapezoidal flux distribution in the air gap due to concentrated full pitch windings in the stator. These motors are becoming increasingly attractive in servo and variable speed applications due to their high torque to inertia ratio and long term maintenance free operation due to the absence of brushes and mechanical commutation [11]. The dynamical equations of the motor derived in [12] can be given as

\[
\begin{bmatrix}
V_{an} \\
V_{bn} \\
V_{cn}
\end{bmatrix} =
\begin{bmatrix}
R_s & 0 & 0 \\
0 & R_s & 0 \\
0 & 0 & R_s
\end{bmatrix}
\begin{bmatrix}
i_a \\
i_b \\
i_c
\end{bmatrix} +
\frac{d}{dt}
\begin{bmatrix}
L & M & M \\
M & L & M \\
M & M & L
\end{bmatrix}
\begin{bmatrix}
i_a \\
i_b \\
i_c
\end{bmatrix} +
\begin{bmatrix}
e_a \\
e_b \\
e_c
\end{bmatrix}.
\]  

(3.1)
Due to symmetry in the isolated neutral configuration of the stator winding,

\[ i_a + i_b + i_c = 0. \]  

(3.2)

Hence

\[ M i_b + M i_c = -M i_a. \]  

(3.3)

Substituting equations (3.2) and (3.3) into equation (3.1) results in,

\[
\begin{bmatrix}
\frac{di_a}{dt} \\
\frac{di_b}{dt} \\
\frac{di_c}{dt}
\end{bmatrix} = \begin{bmatrix}
\frac{-R_s}{L_s} & 0 & 0 \\
0 & \frac{-R_s}{L_s} & 0 \\
0 & 0 & \frac{-R_s}{L_s}
\end{bmatrix} \begin{bmatrix}
i_a \\
i_b \\
i_c
\end{bmatrix} + \begin{bmatrix}
\frac{1}{L_s} & 0 & 0 \\
0 & \frac{1}{L_s} & 0 \\
0 & 0 & \frac{1}{L_s}
\end{bmatrix} \begin{bmatrix}
V_{an} \\
V_{bn} \\
V_{cn}
\end{bmatrix} - \begin{bmatrix}
\frac{e_a}{L_s} \\
\frac{e_b}{L_s} \\
\frac{e_c}{L_s}
\end{bmatrix},
\]

(3.4)

where

\[ L_s = L - M. \]
3.2 Local servo amplifier

The electromagnetic torque generated is

\[ T_e = \frac{P}{2} \left( e_a i_a + e_b i_b + e_c i_c \right) / \omega_e. \]  

(3.5)

The mechanical torque equation is

\[ T_e - T_L = J \left( \frac{2}{P} \frac{d \omega_e}{dt} + B_f \left( \frac{2}{P} \right) \omega_e. \right) \]

(3.6)

\[ \omega_e = \left( \frac{P}{2} \right) \omega_r \]  

(3.7)

Hence this is simply an AC motor working as a DC motor [12]. The motor is loaded by a shaft that is coupled to an identical motor working as a generator.

3.2 Local servo amplifier

This module mainly comprises the analog current controller and the MOSFET driven inverter acting as the electronic commutator. In the torque mode, the current reference signal \( I_{ref} \) is the external command input to the closed loop system formed by the decoder, the hysteresis comparator and the PWM generator. There are three “Hall sensors” placed on the stator of the BLDC motor that give 120° (electrical) phase shifted square waves in phase with respective phase voltages. These waves represent the stator currents. The decoder circuit converts the latter waves into the six-step signals as shown in Figure 3.1. These converted signals with appropriate polarities are then used to drive the gates of the MOSFETs of the respective phases. The actual phase currents of the motor track the command currents with a hysteresis current controller. In BLDC motors, only two of the phase currents are enabled at any instant, one with positive polarity and the other with negative polarity. This makes the currents of adjacent phases equal in magnitude but opposite in direction. When these currents (equal in magnitude) tend to exceed the hysteresis band, both the respective MOSFETs are turned off at that same instant to initiate current feedback through the free wheeling diodes. The rotor position is determined by a shaft encoder and the speed is calculated by differentiating the rotor angle [23].
3.3 Controller

The speed controller is implemented digitally in a PC using GNU C++ running on Red Hat Linux 7.2 platform. In addition to producing reference signal $I_{ref}$ to the current controller in the local servo amplifier, the controller has to compensate for the network induced delays.

3.4 Network

The network bridges the speed controller and the motor drive forming the distributed control system, but at the price of network induced delays. The PC is interfaced to the network through its normal 10/100 Mbps network interface card (NIC). The motor side is interfaced to the network through Ethernet connectible Z-World BL2100 Micro-Controller at 10 Mbps, programmed using Dynamic C Premier 7.02 [21]. Its interface to the Local servo amplifier is through 16 bit analog to digital converters (ADC) and $+10$ V to $-10$ V digital to analog converters (DAC) [22]. Standard connection oriented TCP sockets are used to connect through the network. Therefore in addition to $I_{ref}$ and $\omega_r$, synchronizing and acknowledgment signals are sent back and forth at the establishment, data transfer and termination phases of the TCP connection. Ethernet frame size is fixed at 64 bytes when defining the sockets. The physical arrangement of the entire test rig is shown in Figure 3.2.

3.5 Time delayed system

3.5.1 Overview

To simplify the analysis, the entire system can be viewed in an onion layout [27]. These layers are
3.5 Time delayed system

Fig. 3.2: BLDC motor drive controlled via Ethernet

1. layer 2 - Micro-controller triggering the tasks of the sensor, actuator and their interface to the network.

2. layer 1 - Network to map the speed data from the sensor to the controller and the control signal from the controller to the actuator.

3. layer 0 - PC interfaced to the network, which calculates the control signal.

Thus each layer has several tasks to carry out, where each of them has a deadline in reaching real-time performance of the DCS. To produce a stable output, they must be arranged in a chronological order. The first digit of the task subscript corresponds to the layer number, while the second is for the task. In layer 0, the logical order of the tasks are:
3 Problem Formulation

- $t_{00}$ - extract speed data from the received TCP packet.
- $t_{01}$ - calculate the control signal.
- $t_{02}$ - encapsulate the control signal into the TCP packet to be sent.

They may be clock-triggered (time-triggered) or event-triggered. When clock-triggered, the start of task $t_{00}$ is synchronized with the internal clock of the PC and the start of $t_{01}$ and $t_{02}$ will depend on the release time and the scheduling algorithm used. On the other hand, in event-triggered layer 0, $t_{00}$ waits until the arrival of the first speed data to start and $t_{01}$ and $t_{02}$ are executed sequentially. At the end of $t_{02}$, $t_{00}$ again starts waiting for the next speed data and this procedure repeats. Therefore whether it is clock triggered or event triggered, the total task execution time of layer 0 is approximately constant.

The tasks of layer 1 are

- $t_{10}$ - map speed data from sensor to controller.
- $t_{11}$ - map control signal from controller to actuator.

In layer 1, time for mapping is influenced by the network topology, the nominal bandwidth, the available bandwidth, the number of active nodes and the distance between nodes etc. Hence the exact completion time of layer 1 is non-deterministic a priori.

The layer 1 considered here is implemented in the standard 10 Mbps Ethernet network. If it is to be clock triggered, it has to be synchronized with the clocks of layer 0, layer 2 or both. The analysis and implementation become less complicated if layer 1 is also to be event triggered.

In layer 2, the logical order of tasks are

- $t_{20}$ - sampling the speed.
3.5 Time delayed system

- $t_{21}$ - analog to digital conversion.
- $t_{22}$ - encapsulate the speed data into a TCP packet to be sent.
- $t_{23}$ - extracting the control signal from the received TCP packet.
- $t_{24}$ - digital to analog convert the control signal.
- $t_{25}$ - release the analog control signal to the actuator.

In order to define an exact starting point for the distributed control loop, task $t_{20}$ is made time triggered. It also simplifies the analysis as there is going to be a unique and constant sampling interval for the entire DCS. The rest of the tasks are made event triggered. Hence only $t_{20}$ is time triggered and periodic and the others are event triggered and non-periodic. Since the execution time of the tasks $t_{10}$ and $t_{11}$ depends on the instantaneous network condition, the total worst case execution time (WCET)s are non-deterministic.

The problem here is to design a speed controller to be implemented in layer 0, to ascertain closed loop operation of the BLDC motor despite the non-determinism in $t_{10}$ and $t_{11}$. The shorter the control delay $\tau$, i.e., time from sampling interval to the time where the corresponding control signal is released to the actuator, the closer the distributed system approaches the non-networked closed loop system.

3.5.2 Open loop model

It is important to have a linear model of the system in order to design a suitable speed controller. As in the conventional control system terminology, the ‘plant’ is everything else other than the network and the controller, i.e., the BLDC motor and the local servo amplifier.

There are two possibilities in modeling the plant. One way is to model each component individually and cascade them to obtain the complete plant. The other method, which is less prone to accumulated model errors, is to
Problem Formulation

build an input - output model. The entire plant is considered as a single model by means of some system identification experiments. This can be done by observing the response of the overall open loop plant to a known input signal and fitting the system model of suitable order to the recorded input - output behavior. The advantage here is the possibility of obtaining a simple model, which depends only on the dominant modes of the overall system. One disadvantage is the possibility of some non-linear behaviors contained in the system affecting the linear input-output relationship. Nevertheless, since the requirement is to come up with a simple linear model, the second method is used.

In order to obtain a linear model, the plant must operate in the linear region of its performance range. As stated in [30], the linear operation of a BLDC motor is badly affected by its inherent cogging torque specially at low speeds. In order to identify the linear operating region in practice, first a slow sinusoidal input of typically 1.0 Hz frequency and a reasonable amplitude is applied to \( I_{\text{ref}} \) with the speed control loop open. By gradually increasing the amplitude of the input sinusoidal waveform, the saturation limits of the plant can be identified when the distortions are visible in the response. The linear operating region of the motor drive and the corresponding input signal range that will keep the system within the linear region can then be identified, from this basic experiment. The next step is to apply a standard input signal to the system and record the output. To do this, a constant voltage greater than the lower threshold is applied and at steady state, a step with an upper bound below the identified saturation limit is superimposed.

The corresponding speed response is in the linear operating region. This speed response is approximated using curve-fit functions of MATLAB\textsuperscript{TM} to derive the transfer function of the open loop plant. Preliminary investigations on the recorded input-output data reveal that the first order model of the form \( k/(s + a) \) fits to the data satisfactorily.
3.5 Time delayed system

Figure 3.3: Open loop step response

Figure 3.4: Open loop system bode diagram
3.5.3 Design criteria

In the open loop step response shown in Figure 3.3, it can be seen that the rise time is nearly 1 s and is aimed to bring it down to 0.1 s at minimum overshoot. Since the rule of thumb [24] is to have 4 to 10 samples per rise time, the sampling time $h$ is taken as 10 ms constant. The system bandwidth usage is therefore approximately 0.1 Mbps.
4 Analysis of time delayed systems

In this chapter, a timing analysis is carried out on a general DCS. A first order system similar to the plant in Chapter 3 is used to demonstrate the effects of time delays on the distributed closed loop operation.

4.1 Timing Analysis

In the timing analysis, the tasks to be considered are (i) sampling the sensed plant output, (ii) calculating the control signal by executing the control algorithm, (iii) feeding the actuator with the control signal and (iv) implementing the inter-node communication. These tasks can be split into more detailed activities that affect, with different degrees of strictness, to the system timing behavior and its performance.

It is important to stress that strictness in timing behavior of control systems usually means that actions must be started at exact time instants. In hard real-time systems, strictness additionally implies that worst case execution time (WECT)s of tasks must be less than or equal to their respective deadlines. In both cases, a common fact is that missing an action on time will result in an unacceptable output.

Moreover, it has to be pointed out that the control theory and the practice can follow many paradigms such as continuous or discrete control; centralized or distributed control; direct, feedback or feed-forward control; mono-rate or multi-rate control; classical, adaptive or fuzzy control. One or
many of these paradigms are found in any control system, implying different timing behaviors.

Therefore, time is an important issue in control systems, when timing behavior, schedulability, performance and quality of service (QoS) issues are concerned. Control theory assumes a highly deterministic timing of the implementation, as described in [24]. Consequently, [25], [26] and [27] have treated deficiencies in the computer system implementation of the DCS with respect to time-variations and time-restrictions.

Figure 4.1 outlines which timing specifications that should be derived from DCSs in order to understand the significant timing behaviors of a computer controlled distributed feedback control systems.

Beside the network, it consists of the three main nodes; the controller node with an embedded reference signal generator, the sampler node and the actuator node. In the typical control scenario as per the onion model analysis in Chapter 3, the following functionality is expected when all tasks
4.1 Timing Analysis

but sampling are non-periodic: When the sampler operates on the plant output with a fixed period $h$, the global time for the $n^{th}$ sampling instant is given by

$$ t_{\text{sample}}(n) = t_{\text{sample}}(n-1) + h. \quad (4.1) $$

When the sampler has collected the data, it is mapped to the controller, introducing a communication delay $\tau_{sc}$. The controller starts execution corresponding to the $n^{th}$ sample at

$$ t_{\text{control}}(n) = t_{\text{sample}}(n) + \tau_{sc}. \quad (4.2) $$

The controller executes the control algorithm in order to derive the actuating signal, introducing a computing delay $\tau_c$. The controller maps the actuating signal to the actuator, introducing another communication delay $\tau_{ca}$. Finally the actuator performs actuation at the time given by

$$ t_{\text{actuate}}(n) = t_{\text{control}}(n) + \tau_{c} + \tau_{ca}. \quad (4.3) $$

In this short description, it is shown that every-time the control algorithm is executed, a computing delay $\tau_c$ is introduced. It is assumed that the sampler and the actuator operate virtually in zero time. In order to be more realistic, it is necessary to introduce an acceptable deviation (tolerance) from the instant of the sampling ($tol_{\text{sample}}$), and a similar tolerance for actuation ($tol_{\text{actuate}}$) (Figure 4.2). Then equations (4.1) and (4.2) can be modified to

$$ t_{\text{sample}}(n) = t_{\text{sample}}(n-1) + h \pm tol_{\text{sample}}. \quad (4.4) $$
4 Analysis of time delayed systems

\[ t_{\text{actuate}} (n) = t_{\text{control}} (n) + \tau_c + \tau_{ca} \pm t_{\text{tolactuate}}. \] (4.5)

Moreover, the controller timing analysis could be made more precise if the release time \( T_{\text{release}} \), start time \( T_{\text{start}} \), jitter (varying or fixed difference between \( T_{\text{release}} \) and \( T_{\text{start}} \)) and the completion time \( T_{\text{complete}} \) of the controller activity are introduced. Then equation (4.2) becomes

\[ t_{\text{control}} (n) = t_{\text{sample}} (n) + \tau_{sc} + \text{jitter}. \] (4.6)

Thus the distributed control system has two inter-task communication procedures that introduce delays (fixed or varying) in the timing behavior. Finally, the total execution time, from the sampling to the actuation of the control system, termed as the control delay, can be given by

\[ \tau (n) = t_{\text{actuate}} (n) - t_{\text{sample}} (n). \] (4.7)

4.1.1 Discrete Time Control System

In this section, the mathematical description of a general control system is derived and is extended to a system with input time delays. The total control delay \( \tau \) derived in equation (4.7) has been used for the analysis. A general control system in continuous time can be derived as

\[ \begin{align*}
\frac{dx(t)}{dt} &= Ax(t) + Bu(t) \\
y(t) &= Cx(t).
\end{align*} \] (4.8)

Discretizing the system (4.8) at a sampling interval \( h \), gives

\[ \begin{align*}
x(kh + h) &= \Phi (h) x(kh) + \Gamma (h) u(kh) \\
y(kh) &= Cx(kh),
\end{align*} \] (4.9)

where

\[ \begin{align*}
\Phi (h) &= e^{Ah} \quad \text{(4.10)} \\
\Gamma (h) &= \int_0^h e^{As} dsB. \quad \text{(4.11)}
\end{align*} \]
4.1 Timing Analysis

The discrete transfer function between input and output can be derived as

\[
\frac{Y(z)}{U(z)} = C [zI - \Phi(h)]^{-1} \Gamma(h). \tag{4.12}
\]

The system equations derived above assume negligible \( \tau \) compared to \( h \). It is true as long as the system is non-distributed i.e., centralized systems. But practically all the contributors to \( \tau \) i.e., \( \tau_{sc}, \tau_c, \tau_{ca} \) and \( jitter \) can have significant magnitudes compared to \( h \). Therefore, \( \tau = \tau_{sc} + \tau_c + \tau_{ca} + jitter \) as derived in equations (4.1) to (4.7), become comparable with \( h \) when the system is distributed through a temporal constrained media mentioned in Chapter 2. In such a case, the system in equation (4.8) has to be modified as

\[
\frac{dx(t)}{dt} = Ax(t) + Bu(t - \tau) \tag{4.13}
\]

\[
y(t) = Cx(t).
\]

For \( 0 < \tau < h \), the discretized version of equation (4.13) becomes

\[
x(kh + h) = \Phi(h)x(kh) + \Gamma_0(h,\tau)u(kh) + \Gamma_1(h,\tau)u(kh - h) \tag{4.14}
\]

\[
y(kh) = Cx(kh), \tag{4.15}
\]

where

\[
\Phi(h) = e^{Ah}
\]

\[
\Gamma_0(h,\tau) = \int_0^{h-\tau} e^{As} ds B \tag{4.16}
\]

\[
\Gamma_1(h,\tau) = e^{A(h-\tau)} \int_0^\tau e^{As} ds B. \tag{4.17}
\]

The new transfer function is

\[
\frac{Y(z)}{U(z)} = Cz^{-1} [zI - \Phi(kh)]^{-1} [z\Gamma_0(h,\tau) + \Gamma_1(h,\tau)]. \tag{4.18}
\]
4.1.2 Effect of time delays

Model equations (4.13) to (4.17) remain valid for constant \( \tau \) as well as variable \( \tau \) as long as \( 0 < \tau < h \). Compared to the non-distributed transfer function in equation (4.9) the distributed system in equation (4.17) has an additional zero at \( -\Gamma_1(h,\tau)/\Gamma_0(h,\tau) \) and a pole at the origin of the z-plane, caused by the delay. For \( h < \tau \), introducing \( \tau = (d-1)h + \tau' \) simplifies the computation, where \( 0 < \tau' < h \) and \( d \) is an integer. Then equation (4.14) has to be changed as

\[
x(kh + h) = \Phi(h) x(kh) + \Gamma_0(h,\tau') u(kh - dh + h) + \Gamma_1(h,\tau') u(kh - dh) \\
y(kh) = C x(kh).
\] (4.19)

The new discrete transfer function becomes

\[
\frac{Y(z)}{U(z)} = C z^{-d} [zI - \Phi(h)]^{-1} \left[ z\Gamma_0(h,\tau') + \Gamma_1(h,\tau') \right].
\] (4.20)

For this case too, the model equations are valid even for variable \( \tau \) as long as \( d \) remains constant to a given system and \( \tau' \) can vary in \( 0 < \tau' < h \). Change of \( d \) from cycle to cycle depending on the subsequent control delay, results in a respective change in the order of the closed loop system. This can further be exemplified if numerical values are substituted for \( d \). Starting from the most trivial case with \( d = 1 \), i.e., for \( 0 < \tau < h \), the state space model is

\[
\begin{bmatrix}
x(kh + h) \\
u(kh)
\end{bmatrix} = \begin{bmatrix}
\Phi & \Gamma_1 \\
0 & 0
\end{bmatrix} \begin{bmatrix}
x(kh) \\
u(kh - h)
\end{bmatrix} + \begin{bmatrix}
\Gamma_0 \\
I
\end{bmatrix} u(kh).
\] (4.21)

Next for \( d = 2 \) i.e., for \( h < \tau < 2h \), the state space model is

\[
\begin{bmatrix}
x(kh + h) \\
u(kh - h) \\
u(kh)
\end{bmatrix} = \begin{bmatrix}
\Phi & \Gamma_1 & 0 \\
0 & 0 & 0 \\
0 & 0 & I
\end{bmatrix} \begin{bmatrix}
x(kh) \\
u(kh - 2h) \\
u(kh)
\end{bmatrix} + \begin{bmatrix}
\Gamma_0 \\
I \\
0
\end{bmatrix} u(kh - h).
\] (4.22)
Further for \( d = 3 \) i.e., for \( 2h < \tau < 3h \), the state space model is

\[
\begin{bmatrix}
x(kh + h) \\
u(kh - 2h) \\
u(kh - h) \\
u(kh)
\end{bmatrix}
= \begin{bmatrix}
\Phi & \Gamma_1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & I & 0 \\
0 & 0 & 0 & I
\end{bmatrix}
\begin{bmatrix}
x(kh) \\
u(kh - 3h) \\
u(kh - h) \\
u(kh)
\end{bmatrix}
+ \begin{bmatrix}
\Gamma_0 \\
I \\
0 \\
0
\end{bmatrix}
u(kh - h).
\]

(4.23)

Hence the system matrix gets the dimension \((d + 1) \times (d + 1)\). Therefore it is obvious that for a given system \( d \) must remain constant from cycle to cycle for the order of the system matrix to be constant and hence the model to be valid. In practical terms, variations of control delay, only within two successive multiples of sampling interval are acceptable.

From the transfer function in equation (4.20), it can be seen that the larger the values of \( d \) i.e., the delay compared to the sampling interval \( h \), the higher the number of poles added at the origin. Hence the controller should be able to cancel them in order to maintain the system stability. The additional zero only affects the transients but not the stability. Since \( \Gamma_0 > 0 \) and \( \Gamma_1 > 0 \), the zero is from 0 to -1 in the \( z \)-plane and therefore the effect of the additional zero on transients is negligible [27].

### 4.1.3 Simulation results

A first order system, which is the case for most practical motion control systems as well as the test-rig in this report, is chosen to illustrate the effect of control delay on stability and the transient behavior on the DCS. Substituting to equations (4.10), (4.11) and (4.17) gives

\[
x(kh + h) = e^{-ah}x(kh) + \frac{k}{a} \left[ 1 - e^{-ah} \right] u(kh)
\]

\[
y(kh) = x(kh),
\]

(4.24)
for the system \( G(s) = \frac{k}{(s+a)} \) at \( \tau = 0 \), where \( A = [-a] \), \( B = [k] \) and \( C = 1 \) in continuous time. In the distributed system,

\[
\Gamma_0 (h,\tau) = \frac{k}{a} \left[ 1 - e^{-a(h-\tau)} \right] \\
\Gamma_1 (h,\tau) = \frac{k}{a} \left[ e^{-a(h-\tau)} - e^{-ah} \right].
\]

(4.25) (4.26)

Performance degradation when \( d \) increases was simulated and is shown in Figure 4.3, where the speed loop is closed with a simple PI controller, designed without considering the delay.

Accordingly, it can be seen that in order to nullify the performance degradations (caused by distribution itself and its consequence i.e., the time delays) and to maintain the performance as close as possible to the non-distributed version of the control system, the controller must somehow cancel the cause of forming the additional poles and the zero. Further, the degradation pattern shown is plant dependent, i.e., the type of deterioration of the output response depends on the order of the system considered.

### 4.2 Controllability and Observability

Based on equations (4.21), (4.22) and (4.23), the system \( G \), input \( H \) and output \( C \) matrices in the state space form can be defined for different ranges of \( \tau \) according to the following sections.

1. For \( 0 < \tau < h \)

\[
G = \begin{bmatrix} \Phi & \Gamma_1 \\ 0 & 0 \end{bmatrix} \\
H = \begin{bmatrix} \Gamma_0 \\ I \end{bmatrix} \\
C = \begin{bmatrix} 1 & 0 \end{bmatrix}
\]

(4.27)
4.2 Controllability and Observability

Figure 4.3: Step response variation with d

\[ \text{rank} \left[ \begin{array}{c} H \\ G H \end{array} \right] = 2 \quad (4.28) \]
\[ \text{rank} \left[ \begin{array}{c} G' \\ G'C' \end{array} \right] = 2 \quad (4.29) \]

As the rank of equations (4.28) and (4.29) are the same as the order of G, the system is both controllable and observable [28].
2. For $h < \tau < 2h$

\[
G = \begin{bmatrix}
\Phi & \Gamma_1 & 0 \\
0 & 0 & 0 \\
0 & 0 & I
\end{bmatrix} \quad (4.30)
\]

\[
H = \begin{bmatrix}
\Gamma_0 \\
I \\
0
\end{bmatrix}
\]

\[
C = \begin{bmatrix}
1 & 0 & 0
\end{bmatrix}
\]

\[
\text{rank} \left[ H : GH : G^2H \right] = 2 \quad (4.31)
\]

\[
\text{rank} \left[ G' : G'C' : G'^2C' \right] = 2 \quad (4.32)
\]

As the rank of equations (4.31) and (4.32) are one less than the order of the system matrix, the system looses both controllability and observability by one degree.

3. For $2h < \tau < 3h$

\[
G = \begin{bmatrix}
\Phi & \Gamma_1 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & I & 0 \\
0 & 0 & 0 & I
\end{bmatrix} \quad (4.33)
\]

\[
H = \begin{bmatrix}
\Gamma_0 \\
I \\
0 \\
0
\end{bmatrix}
\]

\[
C = \begin{bmatrix}
1 & 0 & 0 & 0
\end{bmatrix}
\]

\[
\text{rank} \left[ H : GH : G^2H : G^3H \right] = 2 \quad (4.34)
\]

\[
\text{rank} \left[ G' : G'C' : G'^2C' : G'^3C' \right] = 2 \quad (4.35)
\]

Similar to the previous case, as the rank of equations (4.34) and (4.35) are two less than the order of the system matrix, the system looses both controllability and observability by two degrees.

This suggests to keep the order of the system matrix unchanged, irrespective of the delay. The only possibility of doing so is to use past control inputs and delayed measurements of state variables and their timing data in estimating the actual state of the plant by the time that the control signal is applied to the actuator. Hence the system order and then the controllability and observability can be maintained as in the case of non-delayed system. This is the basis for the state feedback controller design presented in Chapter 5.

4.3 Delay model

Figure 4.4 shows the statistics of the variation of the control delay $\tau$. It clearly shows that both under high and low network traffic, the control delay is not exactly constant. It varies stochastically about a mean value in a network status dependent manner.

In PI and Smith Predictor speed controller structures, the control delay $\tau$ will be treated as a constant equal to its mean value $\tau_{\text{mean}}$. As a function in the Laplace domain, it will appear as $e^{-\tau_{\text{mean}}s}$. To bring it to the pole-zero based transfer function format, $Pade'$ approximation can be used [29]. In the state feedback controller, real-time control delay $\tau_k$ is used.
Figure 4.4: Control delay variation.
(a) Low network traffic; (b) High network traffic.
5 Controller design for the time delayed system

In sections 5.1 and 5.2 of this chapter, the standard controller structures Proportional Integral (PI) and Smith predictor will be summarized in the way they were implemented on the test rig in practice. Section 5.3 presents the state feedback controller with on-line delay compensation, which is an original contribution.

5.1 PI controller

This method is applicable only for the case where $0 < \tau < h$. In other words, it must be guaranteed that the samples arrive at the controller node in chronological order. Then the damage caused by the delay is limited to the non-periodic arrival of the control signals at the actuator node. Since the Proportional Integral (PI) controller derived here assumes a constant delay in tuning its parameters $K_p$ and $K_i$, at least the transient performance will be degraded by a certain degree, when variations in $\tau$ are encountered. Further, degradations can be expected when $\tau$ increases beyond $h$. The tuning rules for the PI controller are first derived in [31] as
Controller design for the time delayed system

\[ K_p = \frac{1}{k\tau_{\text{mean}}} \quad (5.1) \]
\[ K_i = \frac{1}{3k\tau_{\text{mean}}^2} \]

for a first order, type 1 time delayed system \( \frac{k}{s}e^{-\tau_{\text{mean}}s} \). Later some modifications are added by trial and error in [33] and [34] as

\[ K_p = \frac{0.236a}{k} \left[ \frac{1}{a\tau_{\text{mean}}} \right]^{0.959} \quad (5.2) \]
\[ K_i = \frac{0.111a^2}{k} \left[ \frac{1}{a\tau_{\text{mean}}} \right]^{1.698} \]

for the first order, type 0 time delayed system, \( \frac{k}{(s+a)}e^{-\tau_{\text{mean}}s} \). These tuning rules are used in designing the PI controller for the test rig.

5.1.1 Discretization

The continuous time PI controller \( [K_p + K_i] \) is discretized using the Bilinear transformation \( s = \frac{2}{h} \left[ \frac{z-1}{z+1} \right] \). The resulting discrete time controller mapping the exact poles and zeros from the Laplace s-plane to the z-plane [32] is given by

\[ \frac{U(z)}{E(z)} = \frac{K_p - K_iz^{-1}}{1 - z^{-1}} \quad (5.3) \]

where

\[ K_p = K_p + \frac{K_i h}{2} \quad (5.4) \]
\[ K_i = K_p - \frac{K_i h}{2} \quad (5.5) \]
5.1 PI controller

5.1.2 Integrator Wind-Up problem

In order to make sure the proper implementation of the PI controller designed as specified before, some considerations must be given to certain hardware limitations of the system. One such important limitation is the dynamic operating range of the Digital to Analog Converter (DAC) of the actuator node. This will impose both upper and lower limits for the control signal computed from the PI controller. If the control signal exceeds these limits, the derived control signal \( u(t) \) will no longer apply on the system as shown in Figure 5.1 and the expected output variation will slow down as a result. This will make the integrator output to increase due to the non-zero error that is fed into it for a longer duration. This is known as the integrator wind-up and an elaborated description of this scenario can also be found in [24] and [35]. Employing a suitable integrator anti-windup scheme in the controller is always recommended for PI controller applications, which improves the reliability of the control system.

One way to bring the integrator value down to the limits of the DAC is to incorporate the tracking algorithm explained in [24] and [35]. This will force \( u(t) \) to follow \( v(t) \). The discrete time PI controller incorporating the tracking algorithm can be given as

\[
\begin{align*}
    u_P(k) &= K_P e(k) \\
    u(k) &= u_P(k) + u_I(k) \\
    u_I(k+1) &= u_I(k) + K_I e(k) + v(k) - u(k) \\
    v(k) &= \text{sat}[u(k)],
\end{align*}
\]

where

\[
\text{sat}[u] = \begin{cases} 
    u_{\text{min}} & \text{if } u < u_{\text{min}} \\
    u & \text{if } u_{\text{min}} < u < u_{\text{max}} \\
    u_{\text{max}} & \text{if } u > u_{\text{max}}.
\end{cases}
\]

Here \( u_{\text{max}} \) and \( u_{\text{min}} \) are the maximum and minimum hardware limitations of the DAC respectively.
Controller design for the time delayed system

\[ u(k) = K_P e(k) + \frac{q^{-1}}{1-q^{-1}}[K_I e(k) + K_{AWC}(v(k) - u(k))]. \]  

(5.10)

5.1.3 Stability Problem

An integrator is marginally stable. At saturation when the integrator starts acting freely, it can be made more stable by feeding back the term \( u(k) - v(k) \). This is due to the fact that the pole of the integrator on the unit circle in z-plane is shifted into the unit circle by means of \( K_{AWC} \). This scenario of forming a more stable closed loop is illustrated in Figure 5.2. The corresponding transfer function is given by

\[ \frac{U(z)}{V(z)} = \frac{K_{AWC} z^{-1}}{[1 - (1-K_{AWC})z^{-1}]}, \]  

(5.11)

Figure 5.1: Discrete time integrator anti-windup compensator
This clearly shows the movement of the integrator pole from 1 to \((1 - K_{AWC})\). This stable system always attempts to bring the controller output \(u(k)\) equal to \(v(k)\). Hence, the dynamics of the system can be changed by using \(K_{AWC}\) as a tuning parameter [36]. In practice, a value for \(K_{AWC}\) has to be chosen by trial and error depending on the transient response required.

5.2 Smith Predictor

O. J. M. Smith proposed a controller structure shown in Figure 5.3 that is well known as an effective dead-time compensator for stable processes with time-delays. And this controller is very much applicable for situations, where \(\tau > h\). As far as the control delay variations (Figure 4.4) are concerned, its adaptability to control of distributed systems is worth investigating. Because \(\tau\) can grow large compared to \(h\) and the open loop time constant \(1/a\), when the system undergoes heavy network traffic. The closed loop transfer function of the complete distributed system in Figure 5.3 can be given by

\[
\frac{\omega(s)}{\omega_{ref}(s)} = \frac{G_c(s)G(s)e^{-\tau s}}{1 + G_c(s)[G_m(s) + G(s)e^{\tau s} - G_m(s)e^{\tau mean}]} , \tag{5.12}
\]

where
Controller design for the time delayed system

$G(s)$ is the plant to be controlled,
$G_{est}(s)$ is the estimated system model of $G(s)$,
$G_c(s)$ is the controller,
$\tau_{est}$ is the estimated control delay.

Stability of the Smith predictor is affected by the accuracy with which the model i.e., $G_{est}(s)$ and $\tau_{est}$ represent $G(s)$ and $\tau$ respectively. In the open loop model in Figure 3.3, it can be seen that $G_{est}(s)$ exactly matches $G(s)$. Therefore based on the assumption that $\tau_{est}$ matches actual $\tau$, the transfer function can be reduced to

$$\frac{\omega(s)}{\omega_{ref}(s)} = \frac{G_c(s)G(s)e^{-\tau_{mean}s}}{1 + G_c(s)G_m(s)}.$$  \hspace{1cm} (5.13)

Similar to the PI controller method, $\tau$ is taken to be a constant equal to the mean round trip delay $\tau_{mean}$. According to equation (5.13), the parameters of the primary controller are determined using a model of the delay free part of the system. According to [37], $G_c(s)$ can be PI, PD or PID depending on the order of $G(s)$. To derive $G_c(s)$, the Coefficient Diagram Method (CDM), whose mathematical derivations are found in [38] is used.
5.2 Smith Predictor

5.2.1 Coefficient Diagram Method (CDM)

In Coefficient Diagram Method (CDM), the designer can obtain the characteristic equation of the closed loop system efficiently with a good balance of stability, response and robustness. The strength of CDM lies in that, for any plant, the simplest and robust controller under practical limitations can be found [38]. The block diagram of CDM design for a Single Input-Single Output (SISO) system is shown in Figure 5.4,

where

\[ y(s) \text{ is the output signal,} \]
\[ r(s) \text{ is the reference input,} \]
\[ d(s) \text{ is the disturbance,} \]
\[ n(s) \text{ is the measured output noise,} \]
\[ N(s) \text{ is the numerator of } G(s), \]
\[ D(s) \text{ is the denominator of } G(s), \]
\[ A(s) \text{ is the denominator of the controller,} \]
\[ F(s) \text{ is the reference numerator of the controller,} \]
\[ B(s) \text{ is the feedback numerator of the controller.} \]

Since the transfer function of the controller has two numerators, it becomes a two-degree of freedom system. Therefore better performance can be expected as it can focus on both tracking the reference while suppressing disturbances. The output of the complete SISO system in Figure 5.4 is

\[ y(s) = \frac{N(s)}{F(s)}[F(s)r + A(s)d - B(S)n], \quad (5.14) \]

where

\[ P(s) = D(s)A(s) + N(s)B(s) = \sum_{i=0}^{n} a_i s^i \quad (5.15) \]

is the characteristic equation.
5.2.2 Design procedure

The design procedure for choosing controller parameters can be summarized as follows:

1. The selection of proper $A(s)$, $B(s)$ and $F(s)$ controller polynomials:
   
   $A(s) = \sum_{i=0}^{m} l_i s^i$ and $B(s) = \sum_{i=0}^{m} k_i s^i$. If there is no disturbance to the system, then $\text{deg}A(s)$ and $\text{deg}B(s)$ are taken as $\text{deg}D(s) - 1$.
   
   If the system has a disturbance $d(s)$, $A(s)$ and $B(s)$ polynomials are chosen to remove it by selecting $l_0 = 0$ and $\text{deg}B(s) = \text{deg}D(s)$. In this case, $A(s)$ will have an integrator to reject the disturbance. When $F(s)$ is chosen as $\frac{P(0)}{N(s)}$, the overall closed loop transfer function becomes a Type 1 system. A good closed-loop response can therefore be expected.

2. The equivalent time constant $\tau$:
   
   It is chosen as $\tau = \frac{t_s}{(2.5 \ldots 3.0)}$, where $t_s$ is the required settling time.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure5_4.png}
\caption{Coefficient Diagram Method for SISO system}
\end{figure}
3. Selection of stability indices:
Stability indices described in [38] are given by
\[
\gamma_i = [2.5, 2, \ldots, 2] \quad (5.16)
\]
\[
\gamma_0 = \gamma_{\infty} = 0. \quad (5.17)
\]

4. Determining the coefficients of characteristic equation:
Considering
\[
a_i = \frac{a_0 \bar{\tau}^i}{\gamma_{i-1}^2 \gamma_{i-2} \cdots \gamma_1^{i-1}}, \quad (5.18)
\]  
k_0 is chosen as 1. Then \(k, l, a, k, l, a\) are determined first using equation (5.18) and then the characteristic equation given by equation (5.15). The proposed Smith predictor structure (using CDM) in [37] is shown in Figure 5.5.

5.2.3 Design for the test rig
Assuming that the actual test rig and its model match exactly, the closed loop transfer function of the improved Smith predictor (Figure 5.5) is given by
\[
\frac{\omega(s)}{\omega_{ref}(s)} = \frac{G_{c1}(s)G_{c2}(s)G(s)e^{-r_{\text{mean}}}}{1 + G_{c2}(s)G_{c3}(s)G(s)}, \quad (5.19)
\]
where \(G_{c1}(s) = F(s), G_{c2}(s) = 1/A(s)\) and \(G_{c3}(s) = B(s)\). As it can be seen from equation (5.19), the characteristic equation of closed loop system is free from time delays. Then the transfer functions \(G_{c1}(s), G_{c2}(s)\) and \(G_{c3}(s)\) to form the controller can be found using only the delay free part of the plant transfer function. Hence the controller parameters for the test rig can be found as \(G_{c1} = 1 + \frac{a_0}{k}, G_{c2} = \frac{1}{l}, \text{ and } G_{c3} = k_0\), where \(k_0 = 1\) and \(l_0 = \frac{k_r}{1 - a \tau_{\text{mean}}}.\)
5.3 State Feedback Controller

The idea of using the state feedback controller is to overcome the model deficiencies encountered in the delay part of the system model in the previous controllers. The control delay is treated on-line, i.e., $\tau_k$ compared to the $\tau_{\text{mean}}$ in the previous controller structures. Thereby it is expected to compensate for the delays on-line and to achieve smooth but faster transient performance, enhancing the suitability of the DCS considered for precise servo applications.

5.3.1 Analysis

For the general disturbance free continuous time control system with negligible input time delay described by,

\[
\frac{dx(t)}{dt} = Ax(t) + Bu(t) \quad (5.20)
\]

\[
y(t) = Cx(t).
\]

A discrete time state feedback controller can be given as

\[
u(kh) = -Kx(kh), k = 0,1,2,\ldots.
\]  

(5.21)
Here $x \in \mathbb{R}^n$, $u \in \mathbb{R}^m$, $y \in \mathbb{R}^p$ and the dimensions of $K$ must be compatible with $A$, $B$ and $C$. In the following analysis, the control delay corresponding to the $k^{th}$ sample will be taken as $\tau_k$ in place of $\tau$ in the previous discussions.

For $0 < \tau_k < h$

With the above constraints two system inputs $u((k-1)h)$ and $u(kh)$ at most are available during the $k^{th}$ sampling period. Then the system equations in continuous time are

$$\frac{dx(t)}{dt} = Ax(t) + Bu(t),$$
$$t \in [kh + \tau_k, (k + 1)h + \tau_{k+1}]$$
$$y(t) = Cx(t),$$
$$u(t^+) = -Kx(t - \tau_k),$$
$$t \in [kh + \tau_k, k = 0, 1, 2, \ldots]$$ (5.22)

where $u(t^+)$ is piecewise continuous and only changes the value at $kh + \tau_k$.

The discrete time version of the system can be given as

$$x(kh + h) = \Phi(h) x(kh) + \Gamma_0(h, \tau) u(kh) + \Gamma_1(h, \tau) u(kh - h)$$ (5.23)

$$y(kh) = Cx(kh),$$

where

$$\Phi(h) = e^{Ah}$$
$$\Gamma_0(h, \tau) = \int_0^{h-\tau} e^{As} ds B$$
$$\Gamma_1(h, \tau) = e^{A(h-\tau)} - 1 \int_0^{\tau} e^{As} ds B.$$ (5.24)

When a general augmented state vector $z(kh) = [x^T(kh), u^T((k-1)h)]^T$ is introduced, the closed loop system becomes $z((k+1)h) = \hat{\phi}(k)z(kh)$, where
Controller design for the time delayed system

\( \Phi(k) \) is

\[
\tilde{\Phi}(k) = \begin{bmatrix}
\phi & \Gamma_0(\tau_k)K & \Gamma_1(\tau_k) \\
- K & 0
\end{bmatrix}.
\] (5.25)

For \( \tau_k > h \)

Under this condition with \( 0 < \tau_k < dh \), where \( d > 1 \), there can be zero, one or more than one (up to \( d \)) sample(s) in a single sampling period. In the case where \( (d-1)h < \tau_k < dh \), for all \( k \), only one sample is received in every sampling period for \( k > d \). Following the same analysis as in equations (5.22) to (5.25), \( z(kh) \) and \( \tilde{\phi}(k) \) become

\[
z(kh) = [x^T(kh), u^T((k-d)h), \ldots, u^T((k-1)h)]^T, \tag{5.26}
\]

\[
\tilde{\phi}(k) = \begin{bmatrix}
\phi & \Gamma_1(\tau_k^{'}) & \Gamma_0(\tau_k^{'}) & \cdots & 0 \\
0 & 0 & I & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
-K & 0 & 0 & \cdots & 0
\end{bmatrix}.
\] (5.27)

Hence for both the time-variant systems (5.22) to (5.25) and from (5.26) to (5.27), the instantaneous control delay \( \tau_k \) can be compensated by using it on the measured plant state, in estimating the plant state by the time the new control signal reaches the actuator. Therefore it is necessary to obtain \( \tau_k \), where the method used in practice is given in the next section.

### 5.3.2 Timing data acquisition

Having a software reachable system clock in the plant sensor node, a simple way to evaluate \( \tau_k \) on-line is first to synchronize the system clock of the plant to that of the controller, then to stamp the plant time on each sample at the
sampling instant and the controller time at the receipt of the corresponding sample at the controller. Next the synchronization relation between the two system clocks is used to derive $\tau_{sc_k}$. The last step is to use the delay profile of the network to estimate $\tau_{ca_k}$, and finally to obtain $\tau_k$ using the constant $\tau_c + \text{jitter}$ value calculated off-line [30].

### 5.3.3 Clock synchronization

Let $S$ be the node that wants to estimate its clock difference to node $R$. Let the absolute time be $t_i$, the local time in node $S$ be $t^S_i$, and the local time in node $R$ be $t^R_i$. The local clocks in $S$ and $R$ have a skew to the absolute time such that

\begin{align*}
  t^S_i &= t_i + \delta^S \tag{5.28} \\
  t^R_i &= t_i + \delta^R \tag{5.29}
\end{align*}

where $\delta^S$ and $\delta^R$ are the clock mismatches. The clock offset $\delta$ is defined as

\begin{align*}
  \delta &= \delta^R - \delta^S \tag{5.30} \\
  t^S_i &= t^R_i - \delta. \tag{5.31}
\end{align*}

The clock offset will have a drift in time due to inaccuracies in the local clocks. For a short period of operation, it is reasonable to assume that $\delta$ is constant. Otherwise intermediate re-synchronization is needed. The synchronization sequence starts with a clock-read request from node $S$ to node $R$. This message is sent at time $t^S_a$ (Figure 5.6). As node $R$ receives the message from node $S$, it immediately sends a message back containing the local clock value $t^R_b$. This message arrives at node $S$ at time $t^S_b$. $T_{SR}$ and $T_{RS}$ are introduced as the transfer times from $S$ to $R$ and from $R$ to $S$. 

57
Controller design for the time delayed system

Figure 5.6: Clock synchronization sequence. First a request is sent from node S to node R, then R responds by sending the value of its local clock to S.

respectively. The transfer times can be expressed as

\[ T_{SR} = t_b^S - t_a^S = (t_b^R - \delta) - t_a^S \] (5.32)
\[ T_{RS} = t_c^S - t_b^S = t_c^S - (t_b^R - \delta) \] (5.33)

Assuming that \( E(T_{SR} - T_{RS}) = 0 \), it is found that

\[ \delta = E\left(\frac{2(t_b^R - t_a^S - t_c^S)}{2}\right) \] (5.34)

where \( E \) denotes the expectation operator. By repeating the clock synchronization experiment prior to running the controller algorithm, an accurate estimate of \( \delta \) is found.

5.3.4 Time stamping

Once \( \delta \) is known, it can be used in equation (5.31) to find the time at one node, given the time at the other node. The sequence of timing data acquisition can be listed as follows:
5.3 State Feedback Controller

1. Plant time is stamped with the sampled data at the sampler/actuator node.

2. The time at the controller node \( t_{2k} \) is stamped at the end of the sampled data arrival. The extracted sample-originated time in data is used in equation (5.31) to determine the corresponding time in the clock of the controller node \( t_{1k} \). Hence the sampler to controller delay \( \tau_{sck} = t_{2k} - t_{1k} \) is determined prior to the respective controller calculation.

3. In addition to \( \tau_{sc} \), the control delay comprises \( \tau_{ca}, \tau_c \) and jitter. The last two parameters are lumped together and is evaluated by stamping the controller time before and after executing the control algorithm. Therefore it is not feasible to evaluate it on-line. As far as the order of magnitude and its distribution (Figures 5.7 and 5.8) is concerned, it is justifiable to use the mean of off-line calculated \( \tau_c + \text{jitter} \). Assuming the change of network traffic during a single round trip is negligible, the controller to actuator delay \( \tau_{ca} \) is taken to be the same as \( \tau_{sc} \). The Figures 5.7 and 5.8 also show the variations of \( \tau_{sc}, \tau_{ca} \) and their difference under low and high (generated using deliberate file transfer application) network traffic conditions respectively. Since the maximum differences are not more than 1 ms and almost zero otherwise, the above assumption to take \( \tau_{sc} = \tau_{ca} \) is justified. Hence the instantaneous control delay \( \tau_k \) is known \textit{a priori} to the controller calculation.

Comparing Figures 5.7 and 5.8, it can further be seen that the magnitudes of only \( \tau_{sc} \) and \( \tau_{ca} \) are affected by the network traffic status, and not the magnitude of \( \tau_c \).

5.3.5 State Feedback Gain

The characteristic equation of the augmented closed loop time delayed system is,

\[
P(z) = \left| zI - \tilde{\phi}(k) \right| = 0.
\] (5.35)
Freezing the system, when the control delay is $\tau_k$, it turns out to be

$$z^2 + \left( \frac{Kk}{a} \left( 1 - e^{-a(h-\tau_k)} \right) - e^{ah} \right) z + \frac{Kk}{a} e^{-ah} (e^{a\tau_k} - 1) = 0 \quad (5.36)$$

The coefficients of the characteristic equation in the descending order of the power of "$z$" are defined as $a_0$, $a_1$ and $a_2$, where

$$a_0 = 1 \quad (5.37)$$

$$a_1 = \frac{Kk}{a} \left( 1 - e^{-a(h-\tau_k)} \right) - e^{ah}$$

$$a_2 = \frac{Kk}{a} e^{-ah} (e^{a\tau_k} - 1).$$
For the system to be stable Jury’s stability conditions [28] applied are:

\[ a_0 > |a_n| \quad (5.38) \]
\[ P(z)|_{z=1} > 0 \quad (5.39) \]
\[ i.e., a_0 + a_1 + a_2 > 0 \quad (5.40) \]
\[ P(z)|_{z=-1} > 0 \quad (5.41) \]
\[ i.e., a_0 - a_1 + a_2 > 0 \quad (5.42) \]

The necessary conditions required by the inequalities (5.39) and (5.42) and the sufficient conditions obtained by forming the coefficient matrix [28], re-
sults in

\[
\tau_k < \frac{1}{a} \left[ \ln\left| 1 + \frac{ae^{ah}}{Kk} \right| \right] = \tau_{\text{upper}} \quad (5.43)
\]

\[
\tau_k > \frac{1}{a} \left[ \ln\left| \frac{1}{2} \left( 1 - \frac{a}{Kk} \right) \left( 1 + e^{ah} \right) \right| \right] = \tau_{\text{lower}} \quad (5.44)
\]

The inequality (5.39) gives \( e^{ah} > 1 \), which is identically true. Therefore to maintain stability, boundary conditions have to be drawn for the sampling interval \( h \) and the state feedback gain \( K \). The \( \tau_k \) is out of control but it has to satisfy

\[
\max\left[ \tau_{\text{lower}}, 0 \right] < \tau_k < \min\left[ \tau_{\text{upper}}, nh \right], \quad (5.45)
\]

because \( 0 < \tau_k < nh \) and \( n = 1, 2, 3.. \) depending on the amount of delay that the system can tolerate while keeping the performance according to specifications.

Conventionally, a faster sampling rate is desirable in sampled-data systems so that the discrete-time control design and its performance can approximate a near continuous time system i.e., sampling rates well above the Nyquist sampling rate [24]. But in DCS, a faster sampling rate can increase the network load, which in-turn results in longer delay of signals. Thus finding a sampling rate that can both tolerate the network induced delay and achieve the desired system performance is important in the design. However, for this specific application, the sampling rate has been fixed to 100 Hz to achieve the desired transient performance of the motor. Hence the only parameter that can be subjected to variation is \( K \), which is obtained by pole-placement design. The larger the distance of poles is from the origin, the lesser the system is immune to noise. Therefore in addition to the stability bounds for a given \( h \), ultimate upper bound for \( K \) is defined by the system noise level, which is not considered here, assuming the shielding provided is sufficient to protect the system from noise hazards. Plotting the stability region of the DCS with respect to \( h \) and \( \tau_k \) is helpful to judge the possible range of \( K \). As it can be seen in Figure 5.9, for smaller sampling intervals the system can tolerate larger delays up to \( nh \). This should not be interpreted as samples loose their chronological order in arriving the controller. But in an event triggered system (except for sampling), it is the fact that when
5.3 State Feedback Controller

![Figure 5.9: Stability region variation with the acceptable Control Delay](image)

(a) $\tau_k = h$, (b) $\tau_k = 2h$, (c) $\tau_k = 3h$ and (d) $\tau_k = 4h$ with State Feedback gain $K = 5.0$

* - Upper stability boundary, o - Lower stability boundary
Figure 5.10: Stability region variation with constant upper limit for $\frac{\alpha}{h}$ and variable State Feedback Gain $K$.
(a) $K = 5$, (b) $K = 10$, (c) $K = 15$, (d) $K = 20$
* - Upper stability boundary, o - Lower stability boundary

the sampling rate is well above the Nyquist sampling rate, the delay itself becomes the effective sampling rate and hence the system is truly sampled at n times slower than the specified ‘1/h’. The system remains stable as it still does not violate the Nyquist sampling rate conditions. On the other hand, the transient specifications may not be satisfied.
5.3 State Feedback Controller

The Figure 5.10 clearly shows that stability region is greatly reduced with increased $K$. Hence the DCS cannot be made faster beyond a limit defined by the required sampling interval and the tolerable delay.

5.3.6 Estimator Design

Usually the purpose of an estimator is to determine the full states of the plant to be controlled using partial state measurements. Additionally, in a DCS, the partial measurements get delayed by $\tau_{sc_k}$ in reaching the estimator node as shown in Figure 5.11. By the time the state measurements reach the controller, the actual plant state may have changed. Therefore prior to unknown states estimation, the estimator must evaluate the partial state measurements pretending the states just before the control signal has been released at the actuator node. This is essential as there is another delay $\tau_{ca_k}$ before the control signal reaches the actuator. The delay $\tau_{ca_k}$ is unknown prior to the control signal computation and is therefore estimated from the known $\tau_{sc_k}$. The estimator is based on

\[
x(t) = e^{A(t-t_0)}x(t_0) + \int_{t_0}^{t} e^{A(t-m)}Bu(m) \, dm. \tag{5.46}
\]

The partial state measurement is the full state measurement as the system is first order. According to Figure 5.11, from $t = k\, h$ to $t = k\, h + \tau_k$, the control input is $u((k-1)\, h)$. $\hat{x}(k\, h + \tau_k)$ denotes the estimated plant state at the time the control signal corresponding to state $x(kh)$ reaches the actuator, then

\[
\hat{x}(k\, h + \tau_k) = e^{A\tau_k}x(kh) + \int_{kh}^{kh+\tau_k} e^{A(kh+\tau_k-m)}Bu(m) \, dm. \tag{5.47}
\]

For the DCS concerned, the estimator turns out to be

\[
\hat{x}(k\, h + \tau_k) = e^{A\tau_k}x(kh) + \frac{k}{a} \left[ 1 - e^{-a\tau_k} \right] u(k-1). \tag{5.48}
\]
Figure 5.11: Timing data illustration at various nodes of the DCS
5.3 State Feedback Controller

\[ x(k+1) = \Phi x(k) + \Gamma_1 u(k) + \Gamma_2 u(k-1) \]

\[ y(k) = C x(k) \]

Figure 5.12: Converting the regulator state feedback system to a servo control system

5.3.7 Reference Gain

It is important to point out that the pole placement with state estimator or without state estimator i.e., full state feedback as in the system concerned, has no control over the numerator dynamics of the closed loop system. This is because the Pole-Placement determines only the desired characteristic equation. However it is possible to modify a regulator state feedback system into a servo control system, as shown in Figure 5.12. In doing so, it is necessary to provide adjustable gain \( K_0 \) in the input path. The gain of the entire control system can then be determined such that the steady state output to a unit step input is unity. Since the Pole-Placement modifies the gain of the entire system, it will not behave as expected from the design unless \( K_0 \) is properly adjusted. Assuming the closed loop system is

\[ Y(z) = K_0 f(z), \]

for a step input applying final value theorem,

\[ \lim_{k \to \infty} y(k) = \lim_{z \to 1} (1 - z^{-1}) Y(z) \]

\[ = \lim_{z \to 1} (1 - z^{-1}) K_0 f(z) \frac{z}{z - 1} \]

\[ \lim_{z \to 1} K_0 f(z) = 1. \]

Hence \( K_0 \) can be obtained.
5.4 Results

Figure 5.13 compares the actual performances of different controller strategies described in this chapter, for a time varying DCS. The PI controller uses the parameters tuned off-line, despite the time variations caused by \( \tau \). It also considers \( \tau \) as a constant equal to its mean \( \tau_{\text{mean}} \).

The Smith predictor uses the Coefficient Diagram Method (CDM) in its controller parameter calculations. The CDM produces a more stable controller at presence of model errors due to its structure with higher degrees of freedom. In this case too, \( \tau \) is taken as constant. Both controllers are equally good in minimizing the average steady state error, with tiny bursts even in the steady state despite the integral action as the \( \tau_k \) changes considerably from its average. This is because both controllers have been tuned to the average delay and in reality there are variations. Further the PI controller is faster but has oscillations in the transients. The reason is that the numerical integration in the PI controller increases in fixed steps expecting that the control signal is applied to the actuator periodically. The period is supposed to be the integrator time step. But it is not the case and is irregular in practice due to the time variations. The Smith predictor is smooth but slower in transients. Because the error between the actual system and its model is so large that the control signal is small and takes time to track the changes.

On the other hand, the State feedback controller has a comparatively better steady state performance despite the lack of integral action. Further, its transients are smoother and as fast as the PI controller. This is because the time variations are treated on-line by the estimator.

It can be concluded that State feedback with on-line time delay compensation produces a comparatively better controller for the time varying distributed control system containing the first order plant. Further, the Brushless DC motor with this controller can be used for a precise speed control application in a distributed system.
5.4 Results

Figure 5.13: Performance comparison of different controller strategies, State feedback controller (—) vs PI controller ( - -); (a) Speed increment, (b) Speed decrement. State feedback controller (—) vs Smith predictor (.-); (c) Speed increment, (d) Speed decrement.
Controller design for the time delayed system
6 Speed Synchronization

Maintaining speed synchronism among several motor drives undergoing torque disturbances is considered as a suitable application to adapt the distributed speed controller with on-line delay compensation derived in Chapter 5. It is studied theoretically in this chapter.

6.1 Introduction

In the motion control industry, the term synchronization is used often so interchangeably with the term master-slave. In a master-slave system, one machine works as the virtual master and one or many machines follow the master as virtual slaves. Any change in the speed/position of the master is immediately induced among the slaves, all in the same manner or in different degrees depending on the application. The most common question in the master-slave configuration is, “What happens if a slave undergoes a change in speed/position due to a torque disturbance?”. In most industrial applications this is an unanswered question that eventually results in defective products.

The synchronization described in this report has neither a master nor slaves. But all the machines have equal privileges. Whenever any one of them encounter a disturbance on its shaft, it is reflected to the rest of the machines working in parallel either with a minimum synchronizing error or
with a minimum tracking error. Hence it is aimed to answer the above question constructively.

6.2 Application Aspects

The speed/position synchronization has many promising industrial applications, where motion control can be utilized to reduce, both the capital and maintenance costs, most of all the plant down time and the percentage of defectives. Some of these applications are summarized below.

6.2.1 Offset Printing

Offset printing traditionally uses a mechanical line shaft to synchronize the different color printing stations. The mechanical devices involved require high maintenance, and the system is limited in speed.

The mechanical line shaft system can be replaced with individual servomotors (Figure 6.1) that can be synchronized precisely through the distributed motion control technology. Communication to higher level controls, evaluation of system status and drive set-point signals, can be accomplished with Ethernet/IP (Industrial Protocol), i.e., the same Ethernet protocol but with more robust hardware suitable for rough use in industry [39].

6.2.2 Horizontal Bagging

This application involves a continuous roll of foil for horizontal bagging (Figure 6.2). The sealing station handles the foil transport. Electronic line shaft and print mark registration ensure the foil is synchronized with the products being packaged. Electronic line shafting also ensures the product feeder belt and the foil are in continuous position synchronization. Print mark registration will accelerate or decelerate the foil to make up for the possible stretch.
6.2.3 Cut to Length Rotary Knife

The purpose is to cut material to a precise length. For a fixed cut length, and a knife circumference of the same length, it is simply a matter of maintaining a constant speed between the web and the knife. However, for products that require various cut lengths, the knife's circumference would have to vary to match these cut lengths. Since this would not be practical, the knife speed is often profiled. By varying the knife speed, various cut lengths can be obtained. Furthermore, the rotary knife is accelerated so that, the cutting
edge comes into contact with the material that is traveling at the same velocity. This is done to avoid ‘ripping’ the material.

To accomplish this task a cam profile is often employed. A number of cam profiles can be created inside the distributed controller to perform the required contoured movement that is synchronized with the material position/speed to perform the cut (Figure 6.3).
6.2 Application Aspects

Figure 6.3: Synchronized motors in precise material cutting [40]

### 6.2.4 Pick and Place

Pick and Place applications involve the precise movement of product from one location to another. Typically the gripper claw is ‘homed’ to the starting location during the initialization of the system. From that point, as product is sensed, the gripper closes on it and the move is made. Once the final destination point is reached the gripper releases the product and returns to the home position. It allows easy sending of all the appropriate status and control signals from one axis drive to the next.

The point-to-point positioning feature of any commercial motor drive will be capable of meeting this performance demand. Adding the two-way synchronization will further improve the flexibility of the drive while minimizing the speed/position errors and hence the defectives.
6 Speed Synchronization

Figure 6.4: Precise object placement using synchronized motors [40]

6.3 Approach

In the literature, many work related to position synchronization can be found using control techniques such as scalar field theorem [42] and adaptive control theory [43], where the steady state performance is very well guaranteed but not the transient performance. Since the goal of this work is to reflect the impact of a torque disturbance on the speed of one machine to another machine running parallel, emphasis will be both on the steady state as well as the transient speed response, both in non-distributed and distributed speed synchronization.

6.3.1 Master Slave

This is the well known synchronizing method that has been in practice for decades. The basic block diagram configuration is shown in Figure 6.5. The impact of $T_d$ on $\omega_1$ will be reflected at $\omega_2$, provided that two controllers have been tuned accordingly, but not vice versa. Further $\omega_2$ always lags $\omega_1$, which
6.3 Approach

![Figure 6.5: Speed synchronization: Master-slave configuration](image)

could be seen during starting-stopping and disturbance transients. Due to inherent temporal constraints, in the distributed version, the $\omega_2$ will further be delayed in addition to cross coupling disability.

6.3.2 Interactive Feedback Control

This method is an extension to the work found in [41] on cross coupling position control. The term *tracking error* is defined as the difference between the reference speed and the actual speed of each speed control loop, while *synchronizing error* is the difference between the actual speeds of the two speed control loops. The concept in *Interactive Feedback Control* is to use the synchronizing error to couple the speed variation on one machine to another via the reference speed. When there is no disturbance torque, the speeds are equal and each machine has an effective state feedback gain of $K$ (Figure 6.6). But when one machine is disturbed, the synchronizing error acts as a positive feedback on that machine while imposing a negative feedback on the undisturbed one. Hence the disturbed one is accelerated, while the other is decelerated. Thus the synchronizing error is expected to be kept minimum.

The plant under consideration of this report is an open loop first order system and the most suitable controller derived for distributed operation in Chapter 5 is full-state feedback with a delay compensation. The overall system still remains as first order. Even though it is suitable to achieve the desired transient performance in the absence of disturbances on the shaft, measures have to be taken to avoid such steady state speed errors in the
presence of torque disturbances. Therefore an integrator is introduced by means of a PI controller. The block diagram of the non-distributed version is shown in Figure 6.6.

When the system is in steady state under the common reference speed $R(k)$, applying a step disturbance of amplitude $T_{d1}$ on the first machine
6.3 Approach

results in the following system equations,

\[-zT_{d1} = [(z - \Phi)(z - 1) + (K_P z + K_I)(H + K)] x_1(k) - (K_P z + K_I)H x_2(k) \]
\[0 = [(z - \Phi)(z - 1) + (K_P z + K_I)(H + K)] x_2(k) - (K_P z + K_I)H x_1(k). \]  

The system equations can be reduced to

\[
\begin{bmatrix}
  x_1(k) \\
  x_2(k)
\end{bmatrix} = \frac{\gamma}{\alpha^2 - \beta^2} \begin{bmatrix}
  \alpha \\
  -\beta
\end{bmatrix},
\]

where

\[
\alpha = (z - \Phi)(z - 1) + (K_P z + K_I)(H + K) \]
\[\beta = -(K_P z + K_I)H \]
\[\gamma = -zT_{d1}. \]

Hence the only way to keep \(x_2(k)\) as close as possible to \(x_1(k)\) is by making \(K\) as small as possible and \(H\) as high as possible within the stability bounds derived in Chapter 5. Further this selection criteria is a compromise between the tolerable synchronizing error and the speed of the response. In the simulation, a torque disturbance is first applied on Motor1 at time=5 s and the second disturbance is applied on Motor2 at time=10 s before the transient of the first disturbance is completely over. It can be seen clearly in Figure 6.7 that in steady state, both the synchronizing error and the tracking error become zero but at the price of slow response.

6.3.3 Synchronization in distributed systems

For the distributed speed control, the system has to be provided with a state estimator using previous control signals, delayed speed measurements and the time delay (as derived in Chapter 5). The discrete time system block diagram is shown in Figure 6.8. In the presence of the disturbance \(T_d\), the
system can be derived as

\[
\frac{[z \Gamma_0 + \Gamma_1][K_P z + K_I]R(z) - M_1X(z)}{[(z - 1) + M_0(K_P + K_I z)]} - (z - \Phi)X(z) = T_d. \tag{6.5}
\]

Hence the steady state tracking error becomes

\[
E_{ss} = \frac{K M_0}{(I - \Phi)K M_0 + (\Gamma_0 + \Gamma_1)M_1}, \tag{6.6}
\]
6.3 Approach

Figure 6.8: Distributed controlled motor with delay compensation

where

\[ \Phi = e^{-ah} \quad (6.7) \]

\[ \Gamma_0 = \frac{k}{a} [1 - e^{-a(h - \tau)}] \]

\[ \Gamma_1 = \frac{k}{a} [e^{-a(h - \tau)} - e^{-ah}] \]

\[ M_0 = \frac{k}{a} [1 - e^{-a\tau}] \]

\[ M_1 = e^{-a\tau}. \]

Substituting the above values into equation (6.6) results in the steady state tracking error as

\[ E_{ss} = \frac{(1 - e^{-a\tau}) T_d}{(1 - e^{-ah})(1 - e^{-a\tau}) + \frac{ae^{-a\tau}}{kk} (1 - e^{-ah})}, \quad (6.8) \]

which is dependent upon the system dynamics, disturbance amplitude and the delay, but it vanishes when the delay is zero. Examining the block diagram in Figure 6.8 reveals that the cause of the non-zero tracking error is the path for the estimator from the previous control inputs, via \( M_0 \). To
the knowledge of the author, there is no existing estimator that avoids the use of a control input to estimate the unknown and/or delayed states of the system [45], [46]. Therefore the non-zero steady state tracking error has to be taken as a condition unavoidable. Depending on the requirements of the application and the system to be controlled, the state feedback gain $(K)$ and the interactive feedback gain $(H)$ have to be chosen, bound to the stability criteria derived in Chapter 5.

The block diagram for the distributed speed synchronization is shown in Figure 6.9. The disturbance torque $T_{d1}$ is applied on Motor1 at $time=10s$ and is released at $time=30 s$, well after the transient is over. As it can be seen on Figure 6.10, the synchronizing error as a percentage of the reference speed is 0.8 with delays of $\tau_{mean} = 0.8h$ on Motor1 and $\tau_{mean} = 1.8h$ on Motor2. Again the same torque is applied on the same machine at $time=40 s$ and at $time=45 s$, $T_{d2}$ is applied on Motor2 before the transient of the previous disturbance is over. They both recover to the steady state by $time=50 s$ with a very small negative steady state error.

### 6.4 Reference Feedback Control

If the system has stringent specification requirements on steady state speed errors under disturbance torques, the Reference Feedback Control technique shown in Figure 6.11 introduces a reasonable solution. In addition to the current control loop, the speed control loop is also closed internally. The speed measurements are sent through the network to the distributed reference feedback controller, only at the presence of a tracking error. More importantly, this eliminates the use of past speed controller outputs i.e., the past plant inputs in the estimator, which in-turn eliminates the steady state speed errors under torque disturbances.

The results of applying the disturbance torques exactly is as for the Intermediate Feedback Controller case, but with corrections on scaling. The corresponding speed response and the synchronizing error are shown in Fig-
Figure 6.9: Speed synchronization for distributed motors using interactive feedback control
Figure 6.10: Speed responses of the distributed speed synchronization using interactive feedback control

Figure 6.11: Distributed speed synchronization using reference feedback control
Figure 6.12: Speed responses of the distributed speed synchronization using reference feedback control

ure 6.12. It can clearly be seen that the steady state speed error is always zero and the responses are much faster. Even though this method is contradictory to the entire DCS concept, this will be appropriate for bandwidth restricted DCSs.
6 Speed Synchronization
7 Conclusions and future work

The conclusions of the study and the future work are presented in this chapter.

7.1 Conclusions

The time delay problem of the motion control system distributed via the standard 10 Mbps Switched Ethernet network is studied using the following controller strategies both theoretically and experimentally.

1. Standard Proportional Integral (PI) controller with suitable tuning techniques for time delayed systems.

2. Smith Predictor controller modified by Coefficient Diagram Method (CDM) to cater delay model inaccuracies and to achieve higher degrees of freedom in the controller.


The first two strategies use the mean delay of the network, calculated off-line, while the third uses the actual control delay computed as the sum of

1. Sensor to controller delay ($\tau_{sc}$)
2. The predicted controller to actuator delay based on the characteristics of the network ($\tau_{ca}$)

3. The controller execution time ($\tau_c$) and jitter

neglecting the variation of jitter.

The PI controller is originally for non-time delayed systems, but modified to work as a robust controller with time delays. Further it is recommended only for $\tau < h$. The Smith Predictor is inherently for time delayed systems with $\tau >> h$. Hence these two controllers are used to establish the lower and upper boundary solutions for the variable time delayed system. Due to the change of instantaneous state of the network, the delay

- Is not constant
- Does not necessarily satisfy $\tau < h$
- Does not necessarily satisfy $\tau >> h$.

Therefore it is necessary to design the controller regardless of the magnitude of the delay. The state feedback controller, whose delayed states are estimated using the system dynamics in a linear estimator, gives a satisfactory transient closed loop performance.

It has to be emphasized that the maximum tolerable delay is system dependent and is relative to the desired closed loop dynamics and hence the sampling interval. Therefore in general a suitable stability analysis must be carried out in defining the bounds of sampling interval, state feedback gain and the tolerable delay. In this study, Jury’s stability criteria is used.

The results of the delay compensated state feedback distributed controller are applied in speed synchronizing two Brushless DC motors.

The speed synchronization is first studied using the non-distributed version of the system. The state feedback controller is then extended for
the speed synchronization introducing interactive feedback controller whose gains are tuned to achieve minimum possible synchronizing errors. This method together with the state feedback controller with delay compensation is applied for the distributed version of the speed synchronization. The transient performance is satisfactory as expected but the steady state speed of the motor, which underwent the torque disturbance, is with a minor steady state error.

The above mentioned error is caused by the delay compensating state estimator, and is dependent exponentially upon the delay and the system time constant. To cater systems with stringent requirement specifications, which do not accept even such minor steady state errors that last only during the period under disturbances, the Reference Feedback control is introduced. This results in zero steady state error but with a higher synchronizing error only at the peak dropout in speed of the motor which underwent the disturbance. Therefore whether to choose the Interactive feedback control or Reference feedback control must be based on the specifications of the particular drive system.

7.2 Future work

The distributed version of the speed synchronization experiments is limited to simulations due to implementation problems in the controller. The TCP/IP software has a single copy residing in the operating system, which is Red Hat Linux 7.2 in this case. When it is invoked by independent multiple programs, in a single processor system, the CPU context is switched so rapidly, giving the illusion that multiple programs are executed simultaneously. This mechanism is strictly restricted for multiple independent programs without data dependency i.e., no data transfer is allowed between independent programs [13].

The physical implementation of the distributed version of the speed synchronization comprises three main programs. The first two are for the dis-
tributed closed loop operation of each motor, while the third is to couple the speed of one motor to the other and vice versa through the synchronizing controller. Hence the three programs are data dependent and can not be implemented in a single processor system sharing the TCP/IP, unless the operating system can be modified to do so.

An alternative approach is to implement all three loops in a single program. Then a single program has to establish and maintain two TCP sockets, which seem to be independent, but having an interactive communication by means of dependent data transfers, which is again not allowed within the scope of TCP/IP in normal operating systems. Further, this method limits the scalability of the DCS.

Hence it is required to divert the attention to a new Real-Time Operating System (RTOS) environment with the following main features.

- Fully functional support for TCP/IP with inter-thread communication, despite data dependency.
- Real-Time task execution capability with dynamic scheduling, pre-emptive multi-tasking and priority inversion capabilities.
- x686 processor compatibility.
References


References


References


**List of symbols**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>System matrix</td>
<td></td>
</tr>
<tr>
<td>$B$</td>
<td>Input matrix</td>
<td></td>
</tr>
<tr>
<td>$B_f$</td>
<td>Coulomb friction coefficient</td>
<td>[Nms$^{-2}$]</td>
</tr>
<tr>
<td>$C$</td>
<td>Output matrix</td>
<td></td>
</tr>
<tr>
<td>$E_{ss}$</td>
<td>Steady state tracking error</td>
<td>[rad/sec]</td>
</tr>
<tr>
<td>$H$</td>
<td>Interactive feedback gain</td>
<td></td>
</tr>
<tr>
<td>$J$</td>
<td>Equivalent moment of inertia of the rotor shaft</td>
<td>[kgm$^2$]</td>
</tr>
<tr>
<td>$K$</td>
<td>State feedback gain</td>
<td></td>
</tr>
<tr>
<td>$K_{AWC}$</td>
<td>Anti wind-up compensator gain</td>
<td></td>
</tr>
<tr>
<td>$K_0$</td>
<td>Reference gain</td>
<td></td>
</tr>
<tr>
<td>$K_p$</td>
<td>Proportional gain</td>
<td></td>
</tr>
<tr>
<td>$K_i$</td>
<td>Integral gain</td>
<td></td>
</tr>
<tr>
<td>$L$</td>
<td>Stator self inductance per phase</td>
<td>[H]</td>
</tr>
<tr>
<td>$M$</td>
<td>Stator mutual inductance</td>
<td>[H]</td>
</tr>
<tr>
<td>$P$</td>
<td>Number of poles</td>
<td></td>
</tr>
<tr>
<td>$R$</td>
<td>Reference signal</td>
<td></td>
</tr>
<tr>
<td>$R_s$</td>
<td>Stator resistance per phase</td>
<td>[$\Omega$]</td>
</tr>
<tr>
<td>$T_e$</td>
<td>Induced torque</td>
<td>[N/m]</td>
</tr>
<tr>
<td>$T_L$</td>
<td>Load torque</td>
<td>[N/m]</td>
</tr>
<tr>
<td>$V_{an}$</td>
<td>Stator phase a terminal voltage</td>
<td>[V]</td>
</tr>
<tr>
<td>$V_{bn}$</td>
<td>Stator phase b terminal voltage</td>
<td>[V]</td>
</tr>
<tr>
<td>$V_{cn}$</td>
<td>Stator phase c terminal voltage</td>
<td>[V]</td>
</tr>
<tr>
<td>$Y$</td>
<td>Output signal</td>
<td></td>
</tr>
<tr>
<td>$e_a$</td>
<td>Stator back electro motive force in phase a</td>
<td>[V]</td>
</tr>
<tr>
<td>$e_b$</td>
<td>Stator back electro motive force in phase b</td>
<td>[V]</td>
</tr>
</tbody>
</table>
### List of symbols

- $e_c$: Stator back electro motive force in phase c [V]
- $h$: Sampling interval [s]
- $I$: Identity matrix
- $i_a$: Stator phase a current [A]
- $i_b$: Stator phase b current [A]
- $i_c$: Stator phase c current [A]
- $u$: Controller output
- $v$: Reset output
- $\omega_e$: Rotor electrical speed [rad/sec]
- $\omega_r$: Rotor mechanical speed [rad/sec]
- $\gamma_i$: Stability indices
- $\tau$: Absolute control delay [s]
- $\tau_{\text{lower}}$: Lower bound of the control delay [s]
- $\tau_{\text{mean}}$: Mean control delay [s]
- $\tau_{\text{upper}}$: Upper bound of the control delay [s]
- $\tau_{sc}$: Sensor to controller delay [s]
- $\tau_{ca}$: Controller to actuator delay [s]