TIME DELAY COMPENSATION SCHEMES WITH APPLICATION TO NETWORKED CONTROL SYSTEM

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIRMENTS FOR THE DEGREE OF

Master of Technology

In

ELECTRONICS SYSTEM AND COMMUNICATIONS

By

BOLLEPALLY RAJU

ROLL NO: 207EE110

DEPARTMENT OF ELECTRICAL ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY, ROURKELA

2007-2009

TIME DELAY COMPENSATION SCHEMES WITH APPLICATION TO NETWORKED CONTROL SYSTEM

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE
REQUIRMENTS FOR THE DEGREE OF

Master of Technology
In
ELECTRONICS SYSTEM AND COMMUNICATIONS

By

BOLLEPALLY RAJU

Under the Guidance of

PROF.BIDYADHAR SUBUDHI

PROF.SANDIP GHOSH

DEPARTMENT OF ELECTRICAL ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY, ROURKELA

2007-2009

National Institute Of Technology
Rourkela

C E R T I F I C A T E

This is to certify that the thesis entitled, **"TIME DELAY COMPENSATION SCHEMS WITH APPLICATION TO NETWORKED CONTROL SYSTEM"** submitted by Mr. **BOLLEPALLY RAJU** in partial fulfillment of the requirements for the award of Master of Technology Degree in **ELECTRICAL ENGINEERING** with specialization in **"ELECTRONIC SYSTEMS AND COMMUNICATIONS"** at the National Institute of Technology, Rourkela is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Date: **DR. BIDYADHAR SUBUDHI** **PROF. SANDIP GHOSH**

Place**:** **(Supervisor)** **(Co-Supervisor)**

ACKNOWLEDGEMENTS

This project is by far the most significant accomplishment in my life and it would be impossible without people who supported me and believed in me.

I would like to extend my gratitude and my sincere thanks to my honorable, esteemed supervisor **Dr. Bidyadhar Subudhi,** head, Department of Electrical Engineering. He is not only a great lecturer with deep vision but also most importantly a kind person. I sincerely thank for his exemplary guidance and encouragement. His trust and support inspired me in the most important moments of making right decisions and I am glad to work under his supervision.

I am very much thankful to co-supervisor of **Prof. Sandip Ghosh** for providing a solid background for my studies. They have been great sources of inspiration to me and I thank them from the bottom of my heart.

I am very much thankful to **Dr.S.Das, Dr.D.Patra, Mrs.K.R.Subhashini** and **Mr.S.Mohanthy** for providing a solid background for my studies.

I would like to thank all my friends and especially my classmates for all the thoughtful and mind stimulating discussions we had, which prompted us to think beyond the obvious. I've enjoyed their companionship so much during my stay at NIT, Rourkela.

I would like to thank all those who made my stay in Rourkela an unforgettable and rewarding experience.

Last but not least I would like to thank my parents, who taught me the value of hard work by their own example. They rendered me enormous support being apart during the whole tenure of my stay in NIT Rourkela.

BOLLEPALLY RAJU

CONTENTS

	Page No.
Abstract	i
List of Figures	ii
List of Tables	iv
Abbreviations	V
Chapter 1 Introduction	1
1.1. Introduction to networked control system	2
1.2. Background	5
1.2.1 Networks and control	5
1.2.2 Point-to-point architecture of a control system	6
1.2.3 Basic networked control system	6
1.3. Literature survey	7
1.4. Fundamental issues in NCS	15
1.4.1 Networked –induced delay	15
1.4.2 Single-packet versus multiple-packet	16
1.4.3 Dropping Network Packets	16
1.5. Present control networks used in NCS	16
1.6. Applications of NCS	17
1.7. Industrial applications of NCS	18
1.8. Problem formulation	18
1.9. Contribution of thesis	19
1.10. Thesis organization	19
Chapter 2 Delays in Networked control system	20
2.1. Introduction	21
2.2. Delays in networked control system	21
2.3. Delays in-the-loop	21
2.4. Network induced delay control system (NDCS)	23
2.5. Network delay models	24
2.6. Effects of delays in the loop	25
2.7. Control design methodologies	26
2.8. Time delay compensation	28
2.9. Different schemes of time delay control applied to NCS	29
2.10. PID control design and tuning	29

2.10.1. PID control 29
2.10.2. Ziegler-Nichols tuning 32
2.11. Simulation results 33
2.12. Chapter summary 35
Chapter 3 Network Simulator (NS-2) 36
3.1. Introduction 37
3.2. Background on the NS simulator 37
3.3. Tcl and OTcl 38
3.4. NS simulator preliminaries 39
3.4.1. Initialization and termination 39
3.4.2. Definition of a network links and nodes 39
3.4.3. Agents and applications 40
3.4.4. Scheduling events 40
3.4.5. Visualization: using NAM 41
3.4.6. Tracing 42 .
3.5. Simulation results 44
3.6. Chapter summary 44
Chapter 4 Modeling the Digital Servo Motor 45
4.1. Introduction 46
4.2. Mathematical model of a DC servo motor 46
4.3. Model estimation procedure 49
4.3.1 Starting a new session in the GUI 50
4.3.2. Description of the system identification tool window 50
4.3.3. Importing models in to the GUI 52
4.3.4. Subspace Identification 52
4.4. Modeling the Digital servo motor 53
4.5. Chapter summary 56
Chapter 5 Hardware-in-loop simulation 57
5.1. Introduction 58
5.2. UDP (user datagram protocol) 58
5.2.1. Reliability and congestion control solutions 59
5.2.2. Applications 60
5.3. Measurement of delay in feedback loop 60
5.4. Smith predictor 68

5.5. Chapter summary 71

Chapter 6 Experiment on the Digital Servo Motor set-up with Artificial Delay Block 72

6.1. Introduction 73

6.1.1. Over view 73

6.1.2. Working procedure of servo 73

6.2. Description of servo setup 74

6.3. Mechanical unit of servo setup 75

6.4. Digital unit of servo setup 77

6.5. Features 78

6.6. PID control of DC servo motor 78

6.7. PID control of DC servo motor with artificial delay 80

6.8. Chapter summary 83

Chapter 7 Conclusions and Suggestions for future work 84

7.1. Conclusions 85

7.2. Suggestions for future work 85

References 86

ABSTRACT

Feedback control systems wherein the control loops are closed through a real time network are called networked control system (NCSs). Network control systems (NCSs) are spatially distributed systems in which the communication between sensors, actuators, and controllers occurs through a shared band limited digital communication network. The defining feature of an NCS is that information (reference input plant output control input, etc.) is exchanged using a network among control system components (sensors, controller, actuator, etc.). The primary advantages of an NCS are reduced system wiring, ease of system diagnosis and maintenance, and increased system agility.

In the project an extensive study of network parameters like data packet drops, data packet delays are pursued by using a network simulator (NS-2). It simulates the transmission of sensor and control packets between plants and controllers. Some fundamental examples were done using this network simulator.

Time delay occurs used for networked control system when the exchange of data among sensors, actuators and controllers connected through the shared medium. Such delays affect the system Performance degradation and the reduced stability or total instability of the closed-loop system. In order to compensate time-delay in the networked control system (NCS) there are different time delay compensation schemes are available, which is given by predictive controller, PID controller, LQR controller, fuzzy controller, etc. In this thesis the discrete-time PID controller is used for compensating the time delays in the networked control system.

To study in reality an experimental work is done to transfer packet data between two computer systems through a Local area Network (LAN) using UDP protocol. Subsequently the transfer of signal between two computer systems through a LAN using UDP protocol has been also made. These experiments were carried out using SIMULINK Instrument Control Toolbox (ver7.6).

Networked predictive control is also designed for networked control of servo system. This control strategy is applied to a servo control system through the Local Area Network (LAN).SMITH-PREDICTOR proposed to compensate the communication delays in the networked control system.

LIST OF FIGURES

Figure No.		Page No.
1.1	A point-to-point architecture of control system	6
1.2	A block diagram of an NCS	6
1.3	General NCS architecture	8
1.4	Basic NCS model	9
2.1	General NCS configuration and network delays for NCS	21
2.2	Timing diagram of network delay propagation	22
2.3	The block diagram of network-induced delay	24
2.4	Model of a system with varying time-delay	29
2.5	Ziegler-Nichols tuning with step response test(left)frequency response test(right)	33
2.6	Step response of model of a system with constant time-delay	33
2.7	PID controller parameters for the first order system with constant delay	34
3.1	NAM graphic interface	41
3.2	Tracing object in a simplex link	42
3.3	Fields appearing in a trace	43
3.4	Window size of TCP with 20% of random loss	44
4.1	Model of a DC motor	47
4.2	System identification GUI	50
4.3	System identification tool window	51
4.4	System identification tool window after importing the data	53
4.5	Time plot of an imported data	54
4.6	Transient response of the model	54
4.7	Estimation of process model	55
4.8	Measured and simulated model output	56
5.1	Communicating two PC's through LAN	61
5.2	Transmitting simulink model in host PC	61
5.3	UDP send block parameters	62
5.4	UDP receive block parameters	63
5.5	Receiving simulink model in remote PC	64
5.6	Measurement of delay in feedback loop	64
5.7	Simulink model of a system without network	65
5.8	Step response of simulink model of a system without network	65

5.9 Response of simulink model of a system without network 66
5.10 Simulink model of a system with network 67
5.11 Response of simulink PID model with network 68
5.12 Block diagram of smith predictor 69
5.13 Simulink model of a system with smith predictor in NCS 70
5.14 Response of the smith predictor with network 71
6.1 Digital servo set-up 74
6.2 Servo mechanical unit 33-100 76
6.3 Digital unit 33-120 77
6.4 Experimental set-up for PID controller 78
6.5 Simulink model of a general PID controller 79
6.6 Response of a general PID controller 79
6.7 Simulink model of PID controller with network 80
6.8 Response of simulink model of PID controller with network 80
6.9 Experimental set-up for DC servo motor with smith predictor 81
6.10 Simulink model of servo system with smith predictor 81
6.11 Response of simulink model of servo system without smith predictor 82
6.12 Response of simulink model of servo system with smith predictor 82

LIST OF TABLES

Table No.		Page No.
2.1	Ziegler-Nichols tuning table for PID controller	32

ABBREVIATIONS USED

NCS	Networked Control System
CAN	Controller Area Network
LAN	Local Area Network
PID	Proportional-Integral-Derivative
P-P	Point-to-Point
ZOH	Zero Hold Order
WSN	Wireless Sensor Networks
LMI	Linear Matrix Lemma
LQG	Linear Quadratic Gaussian
CT	Continuous-Time
DT	Discrete- Time
NAM	Network Animator
TCL	Tool Command Language
IP	Internet Protocol
UDP	User Datagram Protocol
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
EMS	Energy Management Systems
SCADA	Supervisor Control And Data Acquisition
DCS	Distributed Control System
LQR	Linear-Quadratic Regulator
NS-2	Network Simulator-2
GUI	Graphical User Interface
NDCS	Network Induced Control Systems
QoS	Quality of Service
QoP	Quality of Performance
Z-N	Ziegler-Nichols

PI	Proportional-Integral
OTCL	Object-oriented Tool Command Language
FTP	File Transfer Protocol
CBR	Constant Bit Rate
RED	Random Early Discard
MAC	Media Access Control
ID	Identification
ACKs	Acknowledgements
IS	Intermediate System
TFTP	Trivial File Transfer Protocol
DCCP	Data Congestion Control Protocol
DNS	Domain Name System
SNMP	Simple Network Management Protocol
DHCP	Dynamic Host Configuration Protocol
RIP	Routing Information Protocol
SISO	Single Input Single Output
MIMO	Multiple Input Multiple Output
ADC	Analog to Digital Converter
DAC	Digital to Analog Converter
PWM	Pulse Width Modulation
LCD	Liquid Crystal Display
DVM	Digital Volt Meter
USB	Universal Serial Bus
PC	Personal Computer
LED	Light Emitting Diode
PCI	Peripheral Component Interconnect

CHAPTER- 1

INTRODUCTION

1.1 INTRODUCTION TO NETWORKED CONTROL SYSTEM

Communication networks in the industrial arena have, in the past decade, revolutionized the way facilities are controlled. They have made centralized control centers possible, with a wider range of features and more flexibility than ever before. High data transfer rates have allowed for more efficient data storage, trending, alarming, and analysis. The drawbacks that plagued the early generations of networks have been solved, for the most part, making them reliable enough to be used in the most critical of applications.

The definition of a network is two or more devices connected by some means so they can share information. The "means" is what we will address here. Additionally, even though the most general interpretation of the definition could include many manifestations, we will focus on data communication between devices commonly found in industry. If the problem is broken into manageable parts, we can deal with each one effectively.

Network-based control has emerged as a topic of significant interest in the control community. It is well known that in many practical systems, the physical plant, controller, sensor and actuator are difficult to be located at the same place, and thus signals are required to be transmitted from one place to another. In modern industrial systems, these components are often connected over network media (typically digital band-limited serial communication channels), giving rise to the so-called networked control systems (NCSs).

The study of Networked Control Systems (NCSs) brings together the historically separate disciplines of computer networks and control theory. Feedback control systems, wherein the loops used to control the behavior of a plant are closed through a real-time communication network, are called networked control systems. The defining feature of an NCS is that information is exchanged using a network among control system components (sensors, controller, and actuator).

For many years now, data networking technologies have been widely applied in the control of industrial and military applications. These applications include manufacturing plants, automobiles, and aircrafts. Connecting the control system components in these applications, such as sensors, controllers, and actuators, via a network can effectively reduce the complexity of the systems with nominal economical investments. Furthermore, the applications connected through a network can be remotely controlled from a long-distance

source. Traditionally, the networks used in the aforementioned applications are specific industrial networks, such as CAN (Controller Area Networks), and LAN (Local Area Network). However, general data networks such as Ethernet and Internet are rapidly advancing to be the networks of choices for many applications due to their flexibility and lower costs.

A Networked Control System (NCS) is a control system wherein the control loops are closed through a real-time network. The defining feature of an NCS is that control and feedback signals are exchanged among the system's components in the form of information packages through a network.

Networked Control Systems (NCSs) are one type of distributed control systems where sensors, actuators, and controllers are interconnected by communication networks. The study of NCSs is an interdisciplinary research area, combining both network and control theory. A major trend in modern industrial and commercial systems is to integrate computing, communication, and control into different levels of machine/factory operations. The traditional communication architecture for control systems is point-to-point, that is, a wire connects the central control computer with each sensor or actuator point. This change to common-bus introduces different forms of time delay uncertainty between sensors, actuators, and controllers. Most NCS research has focused on two areas: communication protocols and controller design.

The insertion of the communication network in the feedback control loops makes the analysis and design of an NCS complex. Conventional control theories with many ideal assumptions, such as synchronized control and non-delayed sensing and actuation, must be reevaluated before they can be applied to NCSs. The issues that needs to be addressed while designing an NCS include, *network-induced delays* that occurs while exchanging data among devices connected to the shared medium, and *packet losses*, because of the unreliable network transmission path, where packets not only suffer transmission delays but, even worse, can be lost during transmission.

In an NCS, the most significant feature is the network induced delays, which are usually caused by limited bits rate of the communication channels, by a node waiting to send out a packet via a busy channel, or by signal processing and propagation. The existence of signal

transmission delays generally brings negative effects on NCS stability and performance. This observation further enhances the importance of the study on time-delay systems.

A challenging problem in control of networked-based system is *network delay effects*. The time to read a sensor measurement and to send a control signal to an actuator through the network depends on network characteristics such as their topologies, routing schemes, etc. Therefore, the overall performance of a network-based control system can be significantly affected by network delays. The severity of the delay problem is aggravated when data loss occurs during a transmission. Moreover, the delays do not only degrade the performance of a network-based control system, but also can destabilize the system.

In networked control systems communication of data from sensor to controller or control signal from controller to actuator, occur through communication networks. In Networked Control systems, the data are communicated in the form of packets. PID controllers are widely used controllers in the field of control systems because of their fantastic performance. It gives the combined effect of Proportional, Integral and Derivative controllers, so the system response is improved significantly.

Networked control systems are being used in many places such as automobiles, aircrafts spacecrafts, manufacturing processes etc. The advantages with NCS include simplicity and reliability. They are cheap and easy to maintain. However, these systems face problems such as network-induced delays, data loss in the communication etc. These problems may lead to the instability of the system. A lot of research has been done in networked control systems area. New protocols have been proposed to improve the communication of data. The stability of the networked control systems is an important issue; to maintain the system stability various methods are established.

In this project, an attempt has been made to analyze the stability of an NCS resulting from the network-induced delay (sensor-to-controller delay and controller-to-actuator delay). A simple compensation scheme has been proposed to minimize its effect.

1.2 BACKGROUND

In NCS background there are network and controller are present. There are several techniques used to transmit information through the network. Nearly all data network systems in use today use binary digits (bits), a series of 1s and 0s, to send information, but there also must be methods of carrying the bits across the network.

Messages are assembled into packets with formatting and addressing information, along with the data. The general form of a message packet or frame is a leading header (sometimes called the *preamble*), the data area (called the *payload*), and the trailer. The header contains addressing and error checking information, the data area contains the actual data being transmitted, and the trailer contains more error checking and message management information (e.g. parity and stop bits). Parity, a simple error checking method, uses the number of 1s in a byte (odd or even) to determine if the byte was received correctly.

Simplex transmissions are only in one direction, all of the time. Half-duplex is bidirectional communication allowed in one direction at any given time, and full duplex is bidirectional transmission in both directions simultaneously. In addition to this, synchronous (clocked) transmissions are timed so that both devices know exactly when a transmission will begin and end, whereas asynchronous (un-clocked) transmissions must mark the beginning and end of messages. Synchronous transmission is usually faster than asynchronous, but the timing issue between two remote machines can introduce problems causing asynchronous transmission to be simpler and less expensive, and therefore more widely used. Asynchronous transmission does, however, introduce extra control bits into a message, which slows the rate that actual data can be transferred.

1.2.1 Networks and Control

Network control systems (NCSs) combine two engineering fields, control and computer networks. Computer networks can be wired or wireless. Because NCSs are implemented over a network, a good understanding of underlying communication network protocols, such as Ethernet, Token Bus or Token Ring is required to analyze and model the system's behavior. A Networked Control System (NCS) is a control system wherein the control loops are closed through a real-time network. The defining feature of an NCS is that control and feedback

signals are exchanged among the system's components in the form of information packages through a network.

1.2.2 Point-to-Point Architecture of a Control System

Figure 1.1 shows a control system implemented as a point-to-point (P-P) network. Until about the 1960s, as a norm, sensors, controllers, actuators, drive units, and any other devices used to have point-to-point Interconnections. With such connections, the volume, weight, and complexity of the wiring can grow significantly with the increase in the number of connected devices.

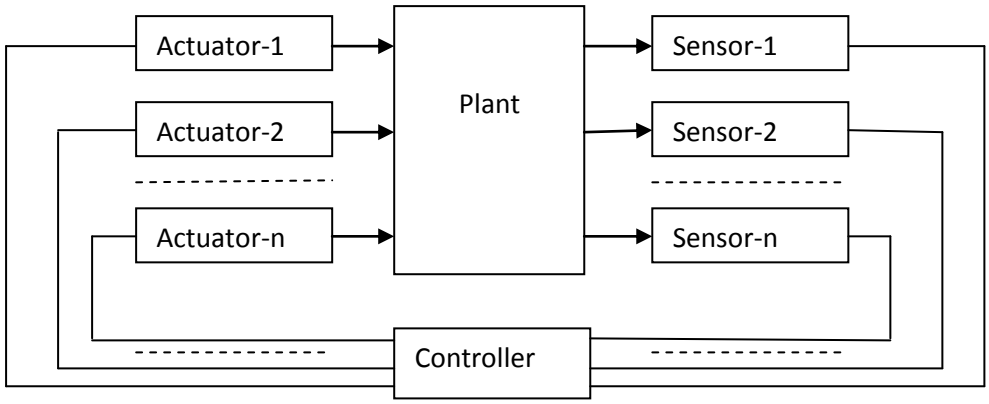


Fig 1.1. A point-to-point architecture of control system

1.2.3 Basic Networked Control System

A *networked control system* (*NCS*) is a feedback control system where the feedback loops are closed by means of an electronic network [1]. Figure 1.2 illustrates a typical networked control system. An NCS benefits its implementer by reduced cost, wiring, and system maintenance. However, NCSs are not subject to the same design assumptions as non-networked continuous- and discrete-time systems, including fixed transmission period, fixed or no delay, and no data loss.

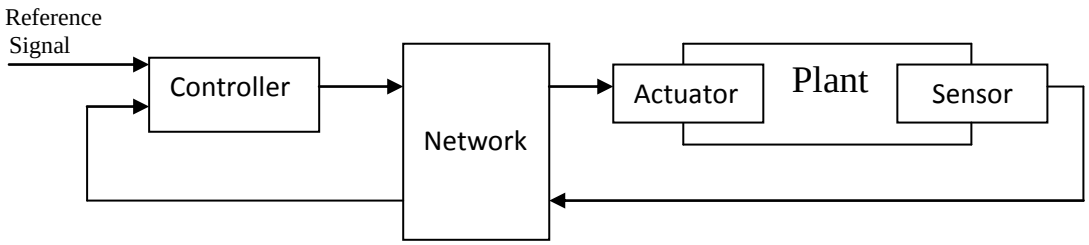


Figure 1.2. A block diagram of an NCS

A *networked control system* (NCS) is a feedback control system where information from the sensors and the controllers is sent over an electronic communication net work [2, 13, 14]. The popularity of computer networks has seen a tremendous rise, and in many cases, it has simply become overwhelmingly practical to use these networks for communication between plants and controllers. NCSs offer reduced cost and relatively simple implementation, as well as greatly increased flexibility. Network protocols have been designed specifically for use in control systems, but other, more general network protocols are also widely used [12].

As mentioned above, NCSs are not without their drawbacks. At best, communication networks can introduce nontrivial delays, but the network can also introduce nondeterministic elements such as time-varying random delays and packet loss. Additionally, the use of an electronic network requires the discretization of measurements as well as the control signal, and limited network capacity affects the ability of the system to use fast sampling frequencies.

1.3. LITERATURE SURVEY

Future applications of control will be much more information-rich than those of the past and will involve networked communications, distributed computing, and higher levels of logic and decision-making. New theory, algorithms, and demonstrations must be developed in which the basic input/output signals are data packets that may arrive at variable times, not necessarily in order, and sometimes not at all. Networks between sensors, actuation, and computation must be taken into account, and algorithms must address the tradeoff between accuracy and computation time. Progress will require significantly more interaction between information theory, computer science, and control than ever before.

Networked Control Systems (NCSs) are one type of distributed control systems where sensors, actuators, and controllers are interconnected by communication networks as shown in Fig 1.3 The study of NCSs is an interdisciplinary research area, combining both network and control theory. A major trend in modern industrial and commercial systems is to integrate computing, communication, and control into different levels of machine/factory operations. The traditional communication architecture for control systems is point-to-point, that is, a wire connects the central control computer with each sensor or actuator point. This change to common-bus introduces different forms of time delay uncertainty between sensors, actuators,

and controllers. Most NCS research has focused on two areas: communication protocols and controller design.

Networked control systems are control systems comprised of the system to be controlled and of actuators, sensors, and controllers, the operation of which is coordinated via a communication network. These systems are typically spatially distributed, may operate in an asynchronous manner, but have their operation coordinated to achieve desired overall objectives. Control systems with spatially distributed components have existed for several decades. Examples include control systems in chemical process plants, refineries, power plants, and airplanes. In the past, in such systems the components were connected via hardwired connections and the systems were designed to bring all the information from the sensors to a central location where the conditions were being monitored and decisions were made on how to control the system.

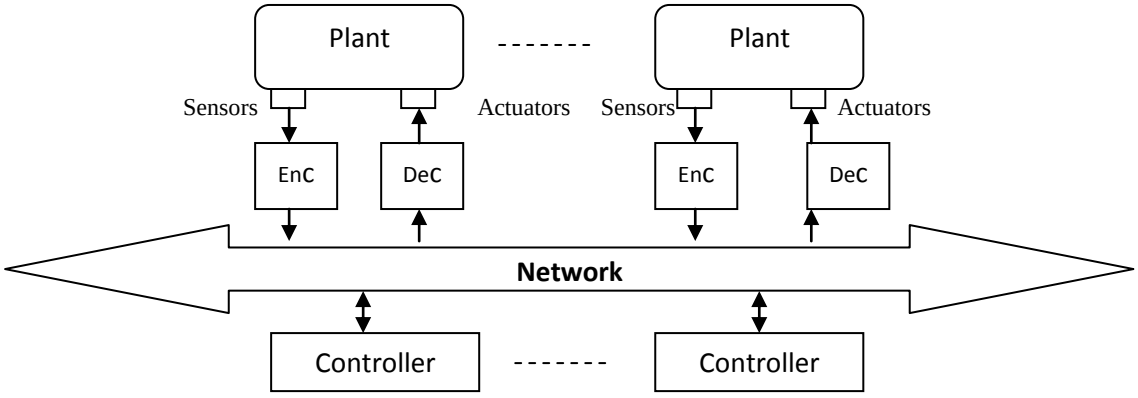


Fig.1.3. General NCS architecture

The control policies then were implemented via the actuators, which could be valves, motors, etc. What is different today is that technology can put low-cost processing power at remote locations via microprocessors and that information can be transmitted reliably via shared digital networks or even wireless connections. These technology driven changes are fueled by the high costs of wiring and the difficulty in introducing additional components into the systems as the needs change.

Traditional control systems composed of interconnected controllers, sensors, and actuators have been successfully implemented using a point-to-point architecture. As an alternative to point-to-point, the common-bus network architecture offers more efficient reconfigurability,

better resource utilization, and also reduces installation and maintenance cost, which is called networked control systems.

In a NCS, various delays with variable length occur due to sharing a common network medium, which are called network-induced delays. Network-induced delays can vary widely according to the transmission time of messages and the overhead time. The network-induced delay in NCSs occurs when sensors, actuators, and controllers Exchange data across the network. Generally, the controlled plant in NCS is assumed to be continuous-time, and thus the actuator implements zero-order hold (ZOH) holding the last control until the next one arrives or until the next sample time. Since networks are used for transmitting the measurements from the plant output to the controller, the plant has to be sampled (sample time *h*), which motivates the use of discrete-time controllers.

NCS are *distributed real-time control systems* consisting of the plant, sensors, controllers, actuators, and a shared data network that is used for communication between the components of the system. A general NCS layout is depicted in Figure 1.4.

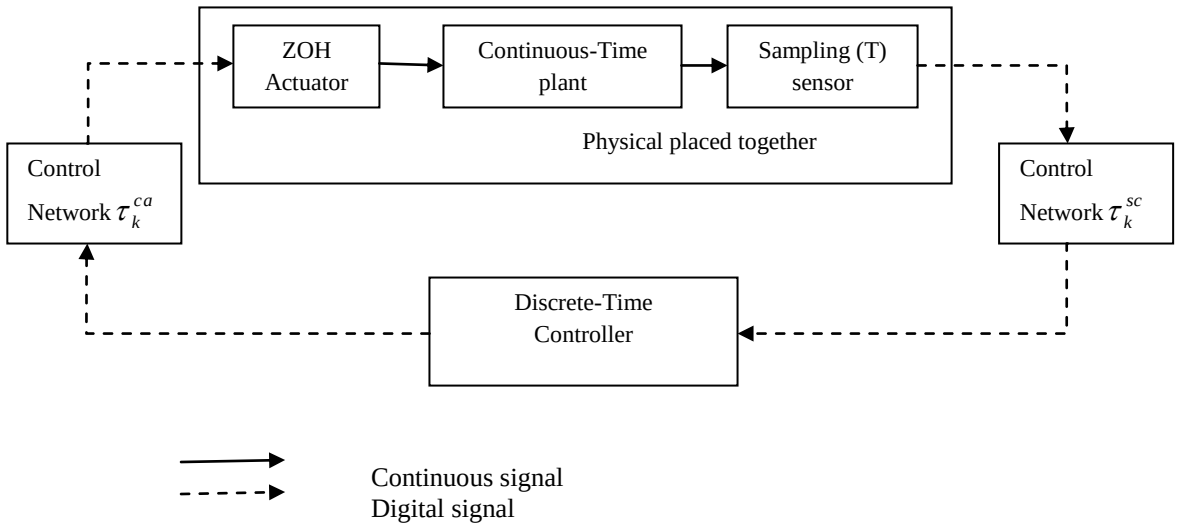


Figure 1.4. Basic NCS model

The controller may be physically placed in a different location from the plant, actuators and sensors, resulting in a distributed control system. The controller can be *time driven* or *event-driven*, so it can calculate the new control signal at discrete time instants with a constant sample time or it can calculate the control signal immediately once it gets a new measurement from the sensor. In addition, the actuator can be time or event-driven.

The network induces delays in the signals: τ_k^{sc} in Figure 1.4 denote the sensor-to-controller delay and τ_k^{ca} the controller-to-actuator delay at time *k*. The controller computational delay τ_k^{c} can be included into τ_k^{sc} or τ_k^{ca} without loss of generality .Depending on the communications protocol used, these delays can be constant, time-varying or random.

Networked control systems (NCS) are feedback control systems wherein the control loops are closed through a real-time network [18]. The different components of NCS exchange information via a shared medium, a digital data network. The main motivations for using networks for data transmissions in control systems are reduced system wiring, ease of system diagnosis as all information is available everywhere in the system, and increased system agility [24]. After the breakthrough of Internet and the preceding development of computer networks in general, there are several cheap and reliable network technologies available. Because of these reasons networks are and will be even more popular in transferring real-time control data [15].

Networks, especially wireless ones, have already changed the consumer markets. Handheld computers with wireless links through which the computers can communicate, and sensors, such as cameras, are all around us. In the industry, though, the use of wireless technology is at a very early stage, although it would bring obvious benefits, because wireless networking extends the possibilities of NCS even further. Industrial applications having mobile subsystems, or just savings in cabling costs, motivate the use of wireless technologies.

Wireless sensor networks (WSN) have been extensively researched for over a decade, because they provide appealing possibilities for distributed, flexible and ubiquitous sensing applications, where each node in the network performs sensing, data processing and communication functions. The highly distributed nature of WSN makes them fault tolerant and adaptive to dynamically changing environments. Even though one node in the network experiences problems and is shut down, networking protocols and both sensing and data processing algorithms could adapt to the changed situation. Hence packets would not be delivered through the faulty node, routes would be reestablished and data processing would adapt to a missing source of measurements. Automatic route establishment and other self-healing properties of networks are required in order to execute control tasks over wireless networks. Often the industrial applications pose stringent timing and reliability requirements,

and with wireless solutions these are more difficult to meet than with wired due to the adverse properties of the radio channels [25].

The shared communication medium in NCS with communication faults that are frequent especially in wireless networks poses new challenges for control design and analysis. The traditional control theory is based on the assumption of uniform sampling of variables and it does not completely cover the cases with varying time-delays or lost and intermittent measurements. The shared medium may be reserved at the time of sampling or there might be collisions of packets. Therefore the nodes might have to wait indefinitely before retransmission. The distributed nature of networks makes it sometimes hard to share common timing information, and due to drifting of local clocks the nodes are more or less asynchronic. In addition, there are various computing tasks in NCS, and depending on the extent of the tasks and scheduling policies of the microcontrollers, the computational times may vary significantly]. Unmodeled and varying time-delays may yield instability in a closed-loop control system and hence the major challenges for control design in NCS are varying time-delays and packet loss.

During recent years the research of NCS has been rapidly growing and several solutions to cope with varying time-delays and packet loss have been proposed. Currently, it seems that there are a few main streams or methodologies that are under specific investigation. A networked control system could be modeled as a linear discrete-time state-space system with varying time-delays in the control and/or output signals. Once the control law is defined, Lyapunov-like stability criteria are applied and the conditions for the corresponding LMI (Linear Matrix Inequality) problem are presented (e.g. [17], [19]). The LMI is then solved using an efficient numerical solver. Another quite common way is to develop observers that act on the intermittent and delayed measurements and to apply LQG (Linear Quadratic Gaussian) design on top of that (e.g. [16], [21]). In all these approaches the practical applicability is questionable, since the calculations are computationally demanding and an industrial control engineer might feel uncomfortable in applying these mathematically challenging solutions in practice. The computational issues play an even more important role in the case of wireless control systems. The computations should be run on a wireless node that needs to have low power consumption and has a very limited computational power and memory available.

This thesis discusses the control design aspects of NCS and focuses on the PID (Proportional-Integral-Derivative) controller, its design and tuning. The PID controller is a widely applied control algorithm in wired control systems, and there is no clear reason why it would not be so in NCS or wireless automation applications. The use of PID in NCS has been researched relatively little, although the choice of its structure and its tuning turn out to be crucial for it to be applied in NCS. The PID controller is sometimes discussed in the literature in connection with NCS (e.g. [22], [25]), but quite rarely the characteristics of the networks are taken into account in any way in the controller tuning. Nevertheless, by tuning the PID appropriately, it can be made very robust against varying time-delays and packet loss. To evaluate the stability of a PID controlled system with uncertainties, such as varying time-delays, robust control techniques can be used. For example, the robustness of different PID tuning methods for a case process with parameter uncertainties has been investigated in [20].

The selection of the sample time plays an important role in NCS. Traditionally, a small sample time is chosen to approximate the continuous-time plant as closely as possible and to enable accurate control. Nevertheless, in NCS, a small sample time causes high network load and an increasing risk of network congestion, which results in longer delays and hence lower performance. Thus the network delay and the sample time are coupled, and finding an optimal balance between the two is a core requirement for achieving well performing and stable NCS [26].

The stability of a closed-loop control system should always be the first concern when designing controllers. As discussed in the previous section, the varying time-delays induced by the network are the major source of instability in NCS. This section reviews some of the stability criteria proposed for NCS and also, in general, for varying time-delay systems.

The stability analysis of varying time-delay systems falls into several subcategories depending on the scenario. The delay can be known or unknown and its maximum value bounded or unbounded. The plant and the controller can be either continuous-time (CT) or discrete-time (DT), and also a mixed case (CT plant and DT controller) is considered in [31]. If the delay is known, the stability criterion may make use of the lengths of delays. Thus the delay-dependent criteria are presumably less conservative than the delay-independent criteria [26]. Most of the available stability criteria for unknown and varying time-delays are given in the time domain (e.g. [27], [28]), but there are also frequency domain criteria such as [29].

The basic problem in NCSs includes network-induced delays, single-packet or multiple-packet transmission of plant input and outputs, and dropping of network packets [36]. The network-induced delays in NCSs occur when sensors, actuators, and controllers exchange data packet across the communication network. This delay can degrade the performance of control systems designed without considering it and can even destabilize the system.

With the development of network technology, more and more intelligent devices or systems have been embedded into Internet services. The Internet has provided a powerful tool for distance collaborative work. It can be potentially used in many applications, such as traffic control, remote surveillance and networked control system. Nevertheless, once a networked control system is connected through the Internet, the random network delay and packet drop induced by the routers of data transmission and network traffic congestion [37] are unavoidable. That could degrade the control performance and even make the control system unstable. Hence, for the NCS the conventional control methods are no longer suitable. Therefore, although the notion of the networked control system is relatively new, more and more attention has been paid to the design and implementation of such systems [38][39][40].

The random network transmission delay makes it very hard to analyze and design networked control systems. However, there is a distinct advantage with using a network that a set of control sequences and measurements can be transmitted from one location to another location at the same time by packing them into one packet. This advantage makes it possible to compensate for the random network delay by using the predictive method. Liu *et al.* [41] introduced a networked predictive control scheme, in which a set of control sequences for every possible delay time are packed together and sent to the plant side, and on the plant side the relevant control signal corresponding to the measured delay is chosen. This is an effective way to compensate for the network delay, but has the disadvantage that the prediction is based on individual forward and backward delays which are difficult to measure separately. There is no synchronization support in the present network, so the clock difference between the controller and plant side cannot be adjusted accurately, which makes it impossible to measure the individual forward and feedback network delays

Packet dropping is often an inevitable event in network data transmission despite the provision network protocols. Network packet drops occasionally happen on an NCS when there are node failures or message collisions. In real-time feedback control, data such as

sensor measurements and calculated control signals, it might be advantageous to discard the old untransmitted message and transmit a new packet if it becomes available. In this way, the controller always receives fresh data for control calculation.

Although the plant would be sampled at discrete time instants at a constant rate, the distributed and asynchronic nature of networks makes the delays time-varying, often in random fashion. Besides the constant delay of the process (though this delay could also be time-varying), the control loop experiences varying time-delays induced by the network. In the literature, these varying delays are often referred to as *jitter*. Many traditional control design methods are based on the assumption of constant time-delays, but this is rarely the case in NCS. Thus new control schemes are needed for NCS, where jitter is present. Therefore the field of networked control systems has lately been researched widely. Recent control design methodologies for NCS are presented, for example, in [42].

The ns-2 network simulator is a network simulation package developed at the Information Sciences Institute at the University of Southern California [34]. Ns-2 provides many powerful objects to simulate different types of networks and network architectures, as well as different types of nodes and traffic patterns. The ns-2 distribution also includes other useful tools for the study of networks, such as NAM, which creates an animation of the traffic on a network.

In [44], the ns-2 simulator was extended to simulate the continuous and discrete dynamics of complex linear and nonlinear systems. This is accomplished through the Agent/Plant package [45], which provides an open way to code plant dynamics, sample scheduling, and control calculation and scheduling in the TCL code of ns-2. Because this approach is so open, it allows for various means of calculating continuous dynamics.

Another problem related to NCS is packet loss. Some of the packets get lost in the network due to interference, noise, node failures and packet collisions. Depending on the protocols used, a lost packet is either retransmitted or not. For example, consider the two transport layer protocols used in IP networks. In TCP (Transmission Control Protocol) messages are retransmitted in the case of lost or erroneous packets, but in UDP (User Datagram Protocol) there are no retransmissions. There is a significant difference in these two protocols with respect to how much they congest the network, but also with respect to the reliability of the connection. In the TCP, acknowledgements from the receiver to sender, indicating that the

messages are received, need to be sent. This increases the network load. On the other hand, in the TCP the data arrive in correct packet order and all the data finally go through, if the network is up and running. In the UDP the packets may arrive in out-of-sequence, and there are no retransmissions. The UDP is therefore much faster than the TCP, but it is also more unreliable [46].

The compensation for NCSs in considered for sensor-to-controller delay, τ^{sc} only. Sensor-to-controller delay can be known when the controller uses the sensor's data to generate the control signal, provided the sensor and controller clocks are synchronized and the message is time stamped. Thus an estimator can be used to reconstruct an approximation to the undelayed plant state and make it available for the control calculation. On the other hand, controller-to-actuator delay is different, in that the controller does not have the information on how long will it take the control signal to reach the actuator, therefore, no exact correction can be made at the time of control calculation.

In order to compensate the network transmission delay, a network delay compensator is proposed. A very important characteristic of the network is that it can transmit a set of data at the same time. Thus, it is assumed that all control predictions at time't' are packed and sent to the plant side through the network. The networked delay compensator chooses the latest control value from the control prediction sequences available on the plant side [47].

1.4. FUNDAMENTAL ISSUES IN NCSs

In this section, we will analyze some basic problems in NCSs, including network-induced delay, single-packet or multiple-packet transmission of plant inputs and outputs, and dropping of network packets.

1.4.1. Network-Induced Delay

The basic problem in NCSs includes network-induced delays, single-packet or multiple-packet transmission of plant input and outputs, and dropping of network packets [47]. The network-induced delays in NCSs occur when sensors, actuators, and controllers exchange data packet across the communication network. This delay can degrade the performance of control systems designed without considering it and can even destabilize the system

1.4.2. Single-Packet versus Multiple-Packet Transmission

Single-packet transmission means that sensor or actuator data are lumped together into one network packet and transmitted at the same time, whereas in multiple-packet transmission, sensor or actuator data are transmitted in separate network packets, and they may not arrive at the controller and plant simultaneously. One reason for multiple- packet transmission is that packet-switched networks can only carry limited information in a single packet due to packet size constraints. Thus, large amounts of data must be broken into multiple packets to be transmitted. The other reason is that sensors and actuators in an NCS are often distributed over a large physical area, and it is impossible to put the data into one network packet.

Conventional sampled-data systems assume that plant outputs and control inputs are delivered at the same time, which may not be true for NCSs with multiple-packet transmissions. Due to network access delays, the controller may not be able to receive all of the plant output updates at the time of the control calculation.

1.4.3. Dropping Network Packets

Network packet drops occasionally happen on an NCS when there are node failures or message collisions. Although most network protocols are equipped with transmission- retry mechanisms, they can only retransmit for a limited time. After this time has expired, the packets are dropped. Furthermore, for real-time feedback control data such as sensor measurements and calculated control signals, it may be advantageous to discard the old, untransmitted message and transmit a new packet if it becomes available. In this way, the controller always receives fresh data for control calculation.

Normally, feedback-controlled plants can tolerate a certain amount of data loss, but it is valuable to determine whether the system is stable when only transmitting the packets at a certain rate and to compute acceptable lower bounds on the packet transmission rate.

1.5. PRESENT CONTROL NETWORKS USED IN NCS

The present control networks used for design NCS are i) CAN ii) ETHERNET.

CAN Network:

The Controller Area Network (CAN) serial bus system is used in a broad range of embedded as well as automation control systems. The main CAN application fields include cars, trucks, trains, aircraft, factory automation, industrial machine control.

CAN network is constituted on serial bus shared by stations by mean of a CSMA/CA scheme with a deterministic collision resolution. The collision resolution is based on priorities associated to identifiers (addresses) of the frame which carry the data) of application tasks

The priority (at which message is transmitted compared to another less urgent message) is specified by the identifier of each message. The priorities are laid down during system design in the form of corresponding binary values and cannot be changed dynamically. The identifier with the lowest binary number has the highest priority. Bus access conflicts are resolved by bit-wise arbitration of the identifiers involved by each station observing the bus level bit for bit.

Ethernet Network

Ethernet was developed in the 1970's and emerged in products in the early 1980s. It is now the dominant local area networking solution in the home and office environment. It is fast, easy to install and the interface ICs are cheap. Despite early attempts to use Ethernet as a real-time communication medium in the factories, practitioners were reluctant to adopt this technology because of its intrinsic non determinism.

Originally, Ethernet uses a shared medium by using for example hub technology. In this case, simultaneous accesses to the medium generate collisions and the transmission is delayed till no collision occurs. It is means, in the worse case, when the medium is overloaded; a message could never to be transmitted.

1.6. APPLICATIONS OF NETWORK CONTROL SYSTEM

- Manufacture automation factories
- Electric factories
- Advanced aircraft

- Network control system (NCS) utilizing high-performance workstations with server/client technologies are used widely for stable and efficient power system operation.
- Energy management Systems (EMS) for generation control and network analysis.
- Mobile sensor networks
- Remote surgery
- Haptics collaboration over the internet
- Automated highway systems and unmanned aerial vehicles

1.7. INDUSTRIAL APPLICATIONS

- SCADA(Supervisory Control And Data Acquisition)
- DCS (Distributed control system)

1.8. PROBLEM FORMULATION

The networked environment presents many new challenges for traditional control system design. For example, many well-known methods have been developed and perfected to analyze and design control systems subject to the effects of discretization. Similarly, much attention has been devoted to the analysis of constant loop delay and its destabilizing effects on the stability of feedback control systems. In the networked environment, however, a controller must combat the degrading effects of both discretization and loop delay. Additionally, the loop delay is often not constant, but time-varying and random.

Depending on the plant and desired performance specifications, sometimes well-known control algorithms can compensate for the detrimental effects of the network. In this thesis, we investigate the adaptation and performance of two popular control algorithms, proportional-integral-derivative (PID) control and smith predictor, in the networked environment.

Time delay occurs in NCS when the exchange of data among sensors, actuators and controllers connected through the shared medium. Such delays affect the system performance degradation and reduce stability or total instability of the closed-loop system. In order to compensate time-delay in the networked control system (NCS) there are different time delay

compensation schemes developed by several researchers, There are for example predictive controller, PID controller, LQR controller and fuzzy controller etc. In this thesis, a discrete-time PID controller is used for compensating the time delays in the networked control system of a DC servo system.

Networked predictive control is also designed for networked control of a DC servo system. This control strategy is applied to a servo control system through a LAN. A smith predictor based delay compensator is also proposed to compensate the communication delays in the above networked control system of a DC servo system.

1.9. CONTRIBUTION OF THE THESIS

The major contributions of this thesis are

1. Review of the Networked Control System (NCS).
2. Review of different time delay compensation schemes and MATLAB simulation studies.
3. Study of the Network Simulator (NS-2).
4. Modeling the digital servo motor.
5. Hardware-In-Loop simulation.
6. Experiment on the Digital servo motor set-up with artificial delay block.

1.10. THESIS ORGANIZATION

In ***Chapter 2***, details regarding delays in NCS and different time delay compensation schemes, details of proposed schemes (PID controller and Smith predictor).in order to know about the parameters of a network we have studied extensively in doing a network simulator studied in chapter -3.

In C***hapter 3***, details of network proposed simulator (NS-2) is given.

In ***Chapter 4,*** details of modeling the digital servo motor using system identification tool box GUI in MATLAB (R2008).

In ***Chapter5,*** details of Hardware –In- Loop simulation and discusses simulation results.

In ***Chapter6,*** details experiment on the Digital servo set-up with artificial delay block.

CHAPTER- 2

DELAYS IN NETWORKED CONTROL SYSTEM

2.1. INTRODUCTION

In this section we detail the delays in the networked control system. The brief explanation of PID controller and compensation scheme is given.

2.2. DELAYS IN NETWORK CONTROL SYSTEM

One of the important problems of NCS is the delay of data transmission between the units of NCS. The network – induced delay appears from two main parts as sensor-controller and controller-actuator. The control systems constructed without considering this delay have a low performance and Reliability.

2.3. DELAYS IN-THE-LOOP

Since an NCS operates over a network, data transfers between the controller and the remote system will induce network delays in addition to the controller processing delay. Fig. 8 shows network delays in the control loop, where r is the reference signal, u is the control signal, y is the output signal, k is the time index, and T is the sampling period. Most of networked control methodologies use the discrete-time formulation shown in Fig 2.1.

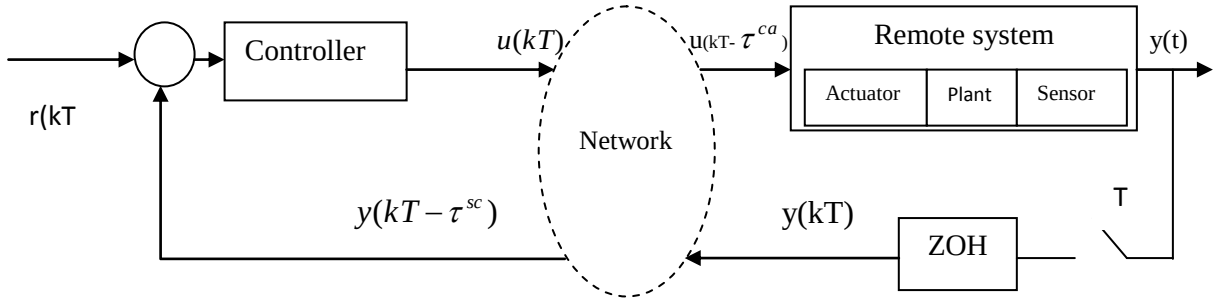


Fig 2.1. General NCS configuration and network delays for NCS formulations

Network delays in an NCS can be categorized from the direction of data transfers as the sensor-to-controller delay τ^{sc} and the controller-to-actuator delay τ^{ca}:

The delays are computed as

$$\tau^{sc} = t^{cs} - t^{sc} \qquad \tau^{ca} = t^{rs} - t^{ce}$$

where τ^{se} is the time instant that the remote system encapsulates the measurement to a frame or a packet to be sent, τ^{cs} is the time instant that the controller starts processing the measurement in the delivered frame or packet, τ^{ce} is the time instant that the main controller

encapsulates the control signal to a packet to be sent, and τ^{rs} is the time instant that the remote system starts processing the control signal. In fact, both network delays can be longer

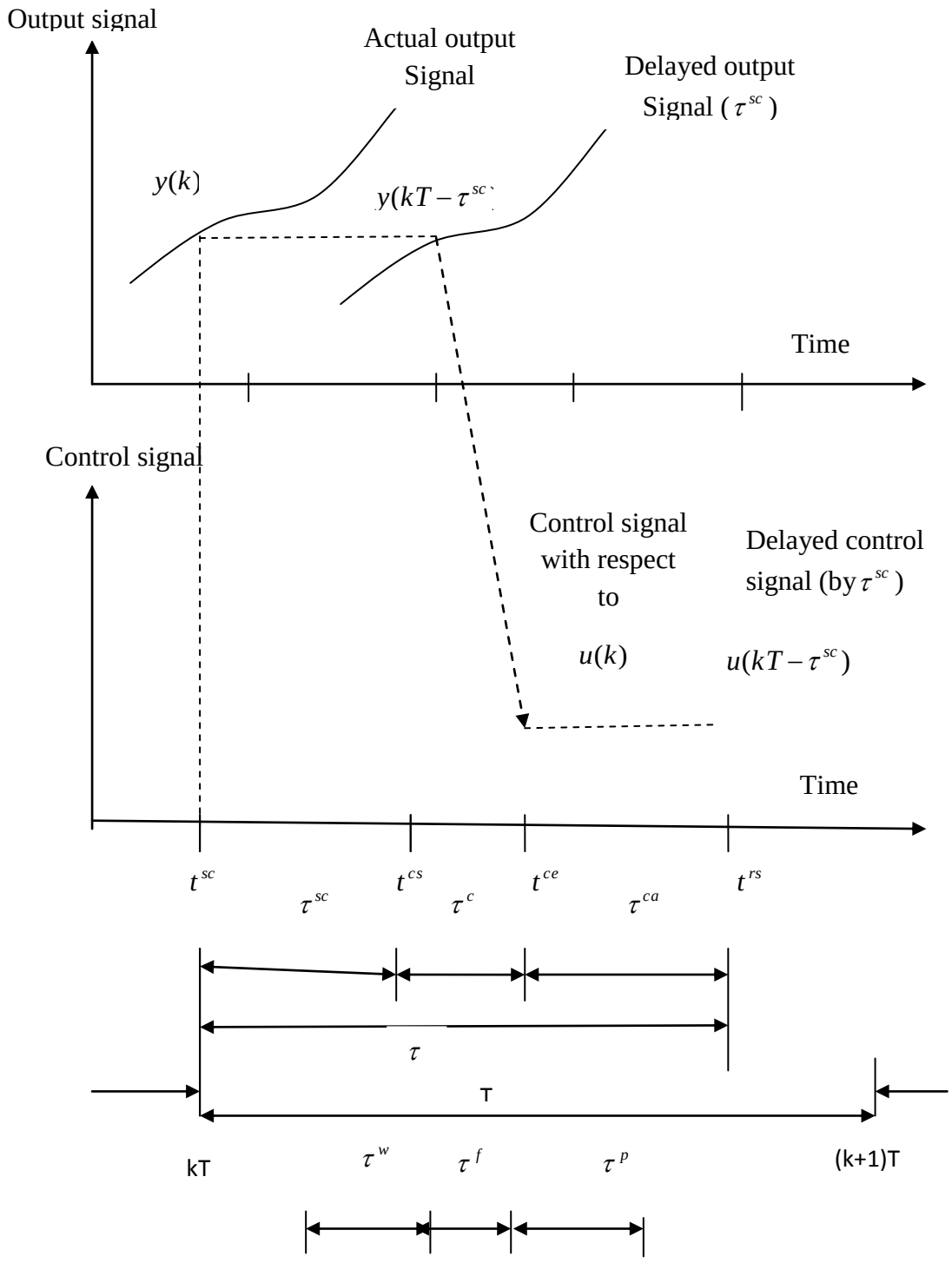


Fig. 2.2. Timing diagram of network delay propagation

or shorter than the sampling time T: The controller processing delay τ^c and both network delays can be lumped together as the control delay t for ease of analysis. This approach has been used in some networked control methodologies. Although the controller processing delay τ^c always exists, this delay is usually small compared to the network delays, and could be neglected. In addition, the sampling periods of the main controller and of the remote system may be different in some cases

Waiting delay τ^W *:* The waiting time delay is the delay, of which a source (the main controller or the remote system) has to wait for queuing and network availability before actually sending a frame or a packet out.

Frame time delay τ^f *:* The frame time delay is the delay during the moment that the source is placing a frame or a packet on the network.

Propagation delay τ^p *:* The propagation delay is the delay for a frame or a packet traveling through a physical media. The propagation delay depends on the speed of signal transmission and the distance between the source and destination

Generally, the controlled plant in NCS is assumed to be continuous-time, and thus the actuator implements zero-order hold (ZOH) holding the last control until the next one arrives or until the next sample time. Since networks are used for transmitting the measurements from the plant output to the controller, the plant has to be sampled (sample time *h*), which motivates the use of discrete-time controllers.

2.4. NETWORK INDUCED DELAY CONTROL SYSTEMS (NDCS)

The network- induced delay in NCS occurs when sensors, actuators, and controllers exchange data across the networks. This delay can degrade the performance of control systems designed without considering it and even destabilize the system.

The network delay model of network induced system is seen in Fig.2.3. Here τ_{ad} is the delay between the sensor-to-controller and τ_{di} is the delay between controller and actuator. By considering this design the generalized mathematical model of NDCS can be constructed. When we build this model we divide the NDCS into three parts as the controlled system (the Plant), the controller system (the Controller) and the network system which bounds these first two parts. That makes easier the system analysis.

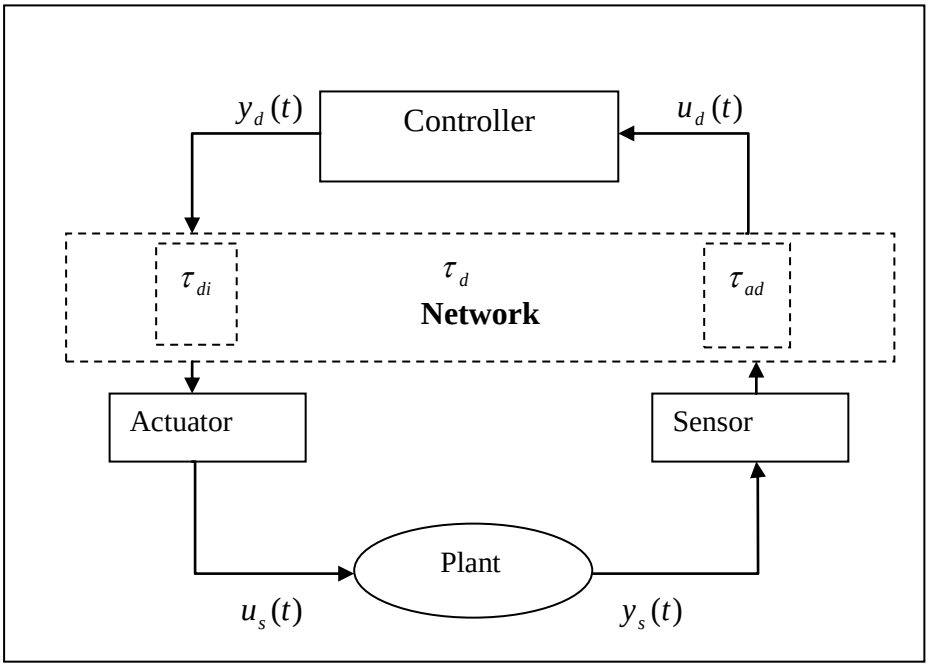


Fig. 2.3. The block diagram of network-induced delay

There is a network between the controller and the components of the area. This construction causes a delay between these two parts. The most important factor in the environment of the network which affects the velocity of data transmission so the time, spent in the network, is the data used in transmission technology

$$T_{as} = T_{af} + T_{ay}$$

here T_{as} – the transmission time in the network medium, T_{ay} – the propagation time in the network medium, T_{af} – the structuring message time in the network medium. The size of the message, the data density and the distance between two points (the length of transmission line) are the main subjects of determining the duration of the time.

2.5. NETWORK DELAY MODELS

There are several sources of delays in NCS. Not only the network dynamics affect the total delay, but also the signal processing and computational delays that depend on the scheduling policies should be taken into account. The network could also be exposed to failures, which would increase the delay variance. Furthermore, if all the components of the NCS are time-driven, there is an additional synchronization delay, because the components have to wait until the next sample instant until they can act. For example, the controller might receive a measurement from the plant $0.1h$ after the measurement is made, but it would have to wait

until the next sample instant until the control signal would be transmitted to the actuator, which again would wait until the next sample instant before actuation of the control command.

Network modeling is in many cases a prerequisite for control design. There are several models available for network delays depending on the network type and protocols used. Generally applicable network delay models are considered, where two different models are discussed. These are: 1) constant delay, 2) random delay that is independent from transmission to transmission.

The constant delay model is good for cases, where the process dynamics are much slower than those of the networks and network delays are significantly smaller than process time-constants and delays. In this case, the network delays might always be less than the sample time and if the system is time-driven, the variance of the delay has no effect. The network delay experienced by the controller or the actuator would always be one sample time in such a case. The independent random delay model is justified, because there are several events in the network that can cause asynchronic behavior for communication. Under the shared medium, not all the nodes can transmit simultaneously and thus sometimes they might need to wait for the network to be idle. In the case of packet collisions, there could be a random back off time after which the nodes would try to transmit again.

The independent random delay models do not capture the effect of a high network load that often leads to correlated random delays, meaning that a delay value is actually dependent on the previous value. If the network is experiencing heavy traffic, it is probable that all packets suffer from long transmission delays until the load decreases.

2.6. EFFECTS OF DELAYS IN-THE-LOOP

One of the important problems of NCS is the delay of data transmission between the units of NCS. This delay causes some data packages be spoiled or completely get lost. That is, the signals are weakened. The network – induced delay appears from two main parts as sensor-controller and controller-actuator. The control systems constructed without considering this delay have a low performance and Reliability. The delay causes in a control loop are widely known to degrade system performances and destabilization of a control system.

2.7. CONTROL DESIGN METHODOLOGIES

The recent control design methodologies for networked control are reviewed. Despite the methodology chosen, the objective is to control the NCS so that stability is guaranteed while providing good performance for the closed-loop control system. The basic concepts of the control algorithms are presented here.

The *augmented deterministic discrete-time model methodology* is based on discrete-time state space models. The controller uses j past measurements $z(k) = y(k - i)$, $i = \{1,\ldots,j\}$ to calculate the control signal at k. The network delays are handled by augmenting the delays into the full system state model, and the stability for periodic delays is proven based on the eigen values of the augmented system state transition matrix.

The basic idea of *queuing methodology* is to use observers and predictors to compensate for the delay and hence to make the NCS time-invariant. The methodology is based on queuing mechanisms that are used to reshape random network delays to deterministic delays. The approach presented in uses an observer to estimate the plant states and a predictor to compute the predictive control based on past output measurements. The control and past output measurements are stored in FIFO queues and shift registers, and these are located before and after the controller in the control loop. First, the past measurements are used to estimate the plant state at $k - \theta + 1$, where θ is the size of the shift register between the sensor and the observer. Next, using the previous estimate, the plant state is predicted at $k + \mu$, where μ is the size of the register after the controller. The predictive control signal $u(k + \mu)$ is then calculated and stored in the shift register. Since both the observer and the predictor are model-based, the performance of the system highly depends on model accuracy.

Nilsson developed an *optimal stochastic control methodology* for NCS. In this work the delay was assumed to be random, but less than one sample time. Later Lincoln extended the optimal control methodology for delays that are longer than one sample time. The basic idea of this approach is to formulate the problem as a LQG problem. The system dynamics are given in state-space and the optimal controller gain is solved from the formulated LQG problem using dynamic programming. Solving the problem requires past delay and full state information, and for the latter the Kaman filter may be applied.

The *perturbation methodology* considers the difference between the current plant output values and the most recently transmitted plant output values as a perturbation to the system and searches for limits to this error. The stability is proven using the Lyapunov approach on the dynamics of the error. Several assumptions are made, including error free communications, fast sampling and noiseless observations, but the plant and the controller may be nonlinear and time variant.

The s*ampling time scheduling methodology* can be used for determining the sensor sample times of NCS based on the delay sensitivity of each control loop in the network. The sensitivity to delay is first analyzed using general frequency domain analysis on the worst-case delay bound. Under certain conditions this methodology leads to optimal network utilization.

In *robust control methodology* the network-induced delays are treated as perturbations to the nominal system, and the control design is performed in the frequency domain using robust control theory. A major advantage of this approach is that there is no need to know the exact delay distributions in advance. The network delays are assumed bounded and they are modeled as simultaneous multiplicative perturbations. Using the Padé approximation, the $H\infty/\mu$-synthesis may be applied and certain multiplicative uncertainty weights are chosen so that the uncertain delays are covered by the controller. Robust performance for randomly time-delayed systems may be achieved using this methodology.

Fuzzy logic modulation methodology takes an advantage of fuzzy logic to update the controller gains based on the error signal between the reference and the actual output of the system, and the output of the controller. The fuzzy logic modulation methodology includes online and offline membership function design by optimization.

Event-based methodology uses the system motion as the reference of the system instead of time. For example, the distance traveled by an end-effectors of a robotic manipulator $y(t)$ could be converted to a motion reference s by a certain mapping. A planner uses the motion reference as an input to calculate the reference $r(s)$, which is used in system control. In a way, the event-based methodology maps the time space into event space and the system stability no longer depends on time. Thus the network-induced delays will not destabilize the system.

End-user control adaptation methodology is based on the ability to measure the traffic conditions of the network, and the controller parameters are adapted accordingly. In this methodology, the controller can request and update the Quality-of-Service (QoS) conditions from the network, and if the desired QoS requirements cannot be met, the controller parameters are adjusted aiming for the best possible performance.

2.8. TIME DELAY COMPENSATION

The time delays in the NCS may deteriorate the system performance and cause the system instability. Therefore, it is necessary to design a controller which can compensate for the time delays and improve the control performance of the NCS.

The sensor to controller delay can be known when the sensors data is used by the controller to generate a control signal. But in the case of controller to actuator delay, the controller does not know how long it will take the control signal to reach actuator. So at the time of control calculation, no exact correction can be made.

An estimator can be used to predict an undelayed plant state and make it available for control calculation. An NCS Estimator must estimate the full state of the plant using partial state measurements and also compensate for sensor delay. This can be implemented by either full state feedback or output feedback.

In the NCS environment the main goal of the control system is to maintain Quality of Performance (QoP) of the control system regardless of the delays in the network. The system should be robust and be able to compensate the delay induced by the network. Prior to presenting the delay compensation strategies it is important to state the following assumptions about the process and the network.

- In the network all control and measurement information is sent in a single packet.
- The process is assumed to be fast. The sampling time necessary to capture all relevant process dynamics is significantly smaller than the network induced delay. (If the sampling time necessary to capture all process dynamics is larger than the network induced delay, then the delay will have a similar effect on QoP of the control system as a small additional measurement error.
- No packet losses occur in the communication network.

2.9 DIFFERENT SCHEMES OF TIME DELAY CONTROL APPLIED TO NCS

The time delay compensation schemes are used to compensate the time delays causes in the feedback loop. Different types of time delay compensation schemes are given below.

1. PID controller
2. Smith predictor
3. Optimal controller
4. Fuzzy controller
5. Robust control
6. Sliding mode controller
7. Adaptive controller

2.10. PID CONTROL DESIGN AND TUNING

This thesis, we investigating PID controller design methods for processes with varying time-delays. In the case of varying time-delay systems the control strategy is hard to analyses with standard analytic control theory. In section the discrete-time PID controller is derived. It is later used to control a distributed system. Some practical aspects that have to be considered in PID controller design are put forth. Then some tuning methods that can be used for processes with varying time-delays are introduced.

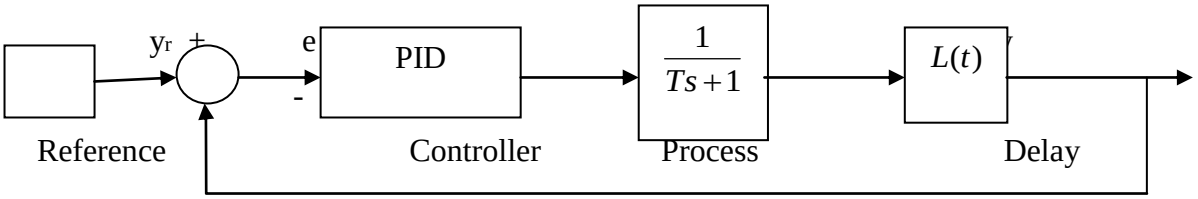


Figure.2.4. Model of a system with varying time-delay

2.10.1. PID Control

The PID controller is the most common controller in control systems. For example, in the mid 1990's the PID controller was used in over 95 % of the control loops in process control. The best features of the controller can only be achieved if the controller is well tuned.
Networked control systems almost demand algorithms like smith predictor, but in some cases a simple and well performing PID controller can be set up with considerably less effort.

The PID controller in continuous time is

$$u(t) = \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt} \right) = u_p(t) + u_i(t) + u_d(t) \quad \text{..... (1)}$$

Where $e(t)$ is the difference

$$e(t) = y_r(t) - y(t)$$

Between the reference signal (the set-point), $y_r(t)$ and the output, $y(t)$ of the controlled process as in Figure 2.4, with $L(t = 0)$.

In transfer function form Equation (1) becomes

$$G_{PID}(s) = \frac{U(s)}{E(s)} = K_p \left(1 + \frac{1}{sT_i} + sT_d \right) E(s) = p + \frac{I}{s} + Ds$$

The coefficients K_p, T_i, T_d and P, I, D are related by:

$$P = K_p$$
$$I = \frac{K_p}{T_i}$$
$$D = K_p T_d$$

The controller has three parts: the proportional part (u_p) is proportional to the error, the integral part (u_i) removes the steady-state error and the derivative part (u_d) reduces the overshoot. The weights of the controller's actions are adjusted with the P, I and D gains.

The PID controller is traditionally suitable for second and lower order systems. It can also be used for higher order plants with dominant second order behavior. The controller is usually implemented in a computer that does calculations in discrete-time. In networked control systems a discrete-time version of the controller is needed because the measurements are sent over the network as packets.

The proportional part of the controller at time-step k is then

$$u_p[k] = u_p(t_k) = K_p e(kh)$$

The integral part

$$u_i(t) = \frac{K_p}{T_i}\int_0^t e(\tau)d\tau$$

is discredited by approximating the integral with a sum

$$u_i[k] = \frac{K_p}{T_i}\sum_{n=0}^{k} he[n]$$

This can further be simplified by computing the difference

$$u_i[k] - u_i[k-1] = \frac{K_p}{T_i}\sum_{n=0}^{k} he[n] - \frac{K_p}{T_i}\sum_{n=0}^{k-1} he[n]$$

$$u_i[k] = u_i[k-1] + \frac{K_p h}{T_i}e[k]$$

The derivative part can be discredited in several ways, with different outcome of the stability and accuracy. The simplest form is the Euler approximation (forward difference):

$$u_d[k] = K_p T_d \frac{e[k+1] - e[k]}{h}$$

Which is usually not used since it amplifies random errors.

In NCS, controllers run in discrete-time. The CT controller discussed above may be discredited as follows. The proportional part of the controller is static and requires no approximation, only sampling. The backward difference method can be used in the approximation of the integral and derivative parts.

$$u[k] = u_p[k] + u_i[k] + u_d[k]$$

$$u_p = K_p e[k]$$

$$u_i[k] = u_i[k-1] + \frac{K_p h}{T_i}e[k]$$

$$u_d[k] = \frac{T_d}{T_d + Nh}u_d[k-1] + \frac{K_p T_d N}{T_d + Nh}\left(e[k] - e[k-1]\right)$$

2.10.2. Ziegler-Nichols Tuning

The Ziegler-Nichols (Z-N) methods rely on open-loop step response or closed-loop frequency response tests. A P-, PI- or PID controller is tuned according to a table based on the process response test.

In the step response test a step is applied on the open-loop system. A tangent line is drawn at the inflexion point on the response curve. The dead-time, T_1, and the rise time, T_2, are measured (Figure 2.5).

In the frequency response method the loop is closed and a pure gain controller is used. The gain is increased to the ultimate gain, K_u, when the system exhibits a steady oscillation whereby the oscillation period is measured, T_u (Figure 2.5).Table 1 shows the tuning rules for a PID controller. P and PI controllers have separate tuning rules.

Table 1: Ziegler-Nichols tuning table for PID controller

Method /parameter	K_p	T_i	T_d
Z-N step response	$1.2\frac{K_1}{K_2}\frac{T_2}{T_1}$	$2T_1$	$0.5T_1$
Z-N frequency response	$0.6K_u$	$0.5Tu$	$0.125Tu$

For a discrete-time PID controller the table should be revised to take the sampling time into account. There is, however, no discrete Z-N tuning table, so Table 1 is used in lack of other. A discrete-time controller approximates a continuous-time controller at small sampling times, so this tuning rule is only used for the shortest sampling times.

The Z-N method is designed for rejecting load disturbances. For reference step changes it performs worse and gives for simple systems a damped oscillating response, where consecutive peaks are ¼ of the previous peak. The tuning is usually bad for higher order systems and can only be considered as a simple first aid tuning.

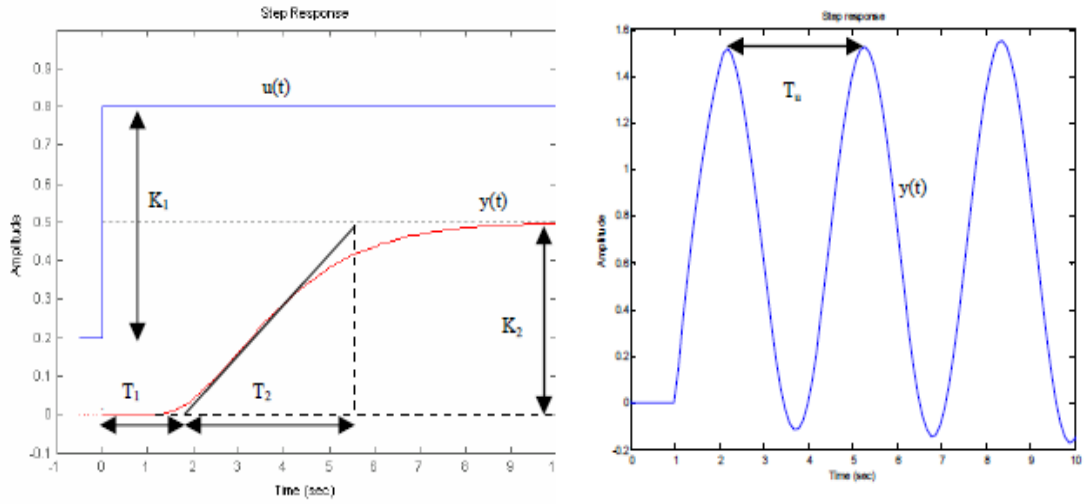


Fig. 2.5. Ziegler-Nichols tuning with step response test (left) and frequency response test (right)

2.11. SIMULATION RESULTS

The PID controller tuning method presented in this Section on a step response. The process is a simple first order system with time constant T and transfer function. The controller is tuned for a process with time constant, T, in the interval [1, 10]. The sampling time of the controller is in the interval h = [0.1, 1].fig.2.6 show the step response of a system with constant delay.

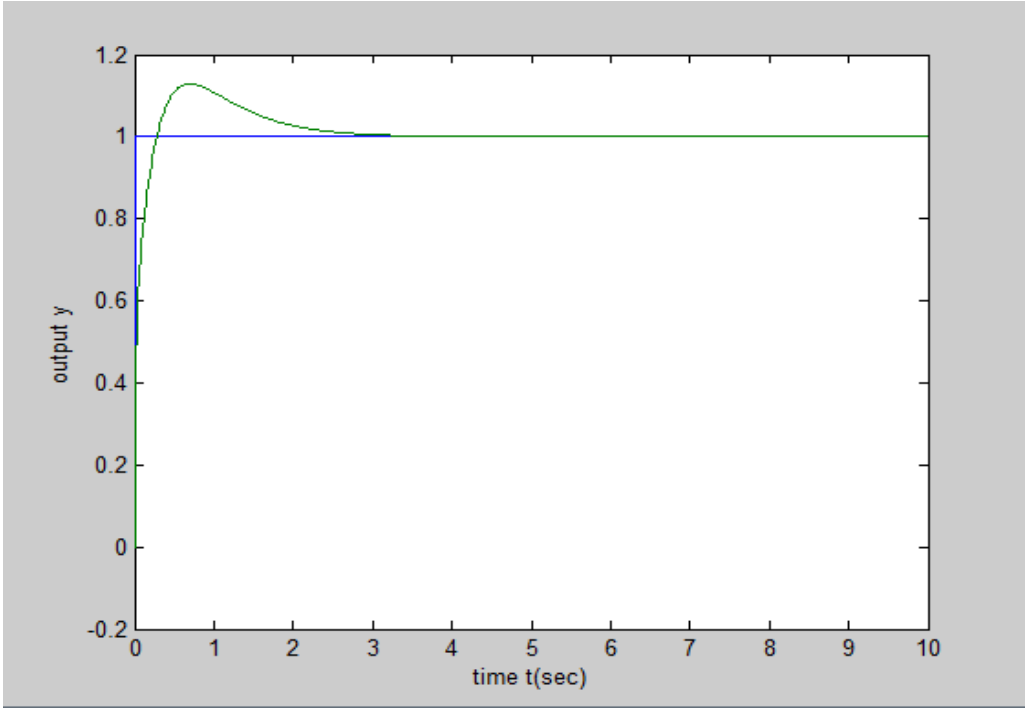


Fig.2.6.Step responses of model of a system with constant time-delay

The PID controller is optimized for a constant delay in the process. These results, shown in Figure 2.7. For a small sampling time (left side of the plots) the results are similar as in the

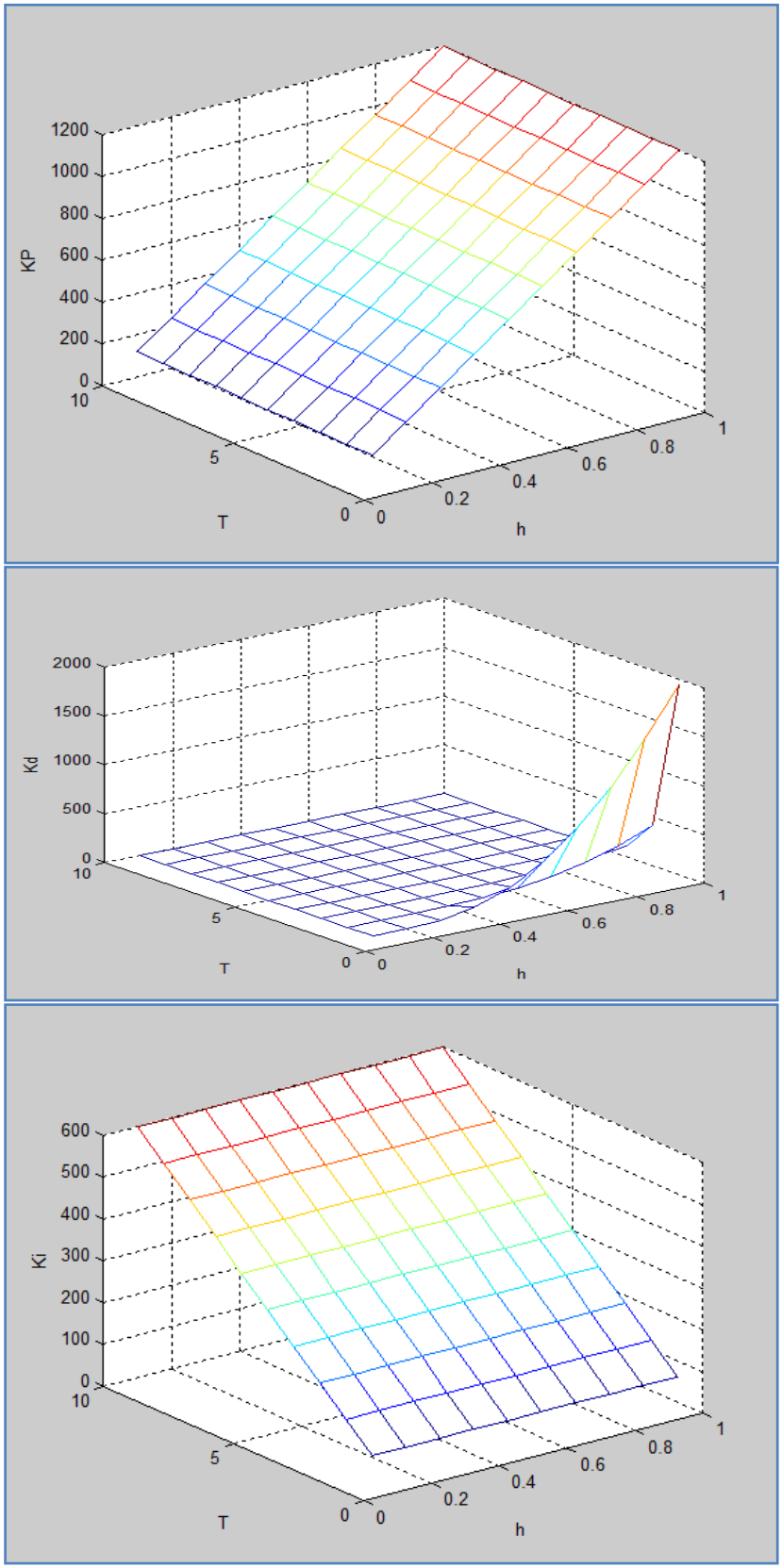


Fig.2.7.PID controller parameters for the first order system with constant delay

continuous case. The P and I terms are linear functions of the time constant, T. The D term is hardly affected by T, but rather by the sampling time. All the terms decrease with increasing sampling time. The controller becomes more cautious as it tracks the output less frequently with increasing sampling time. The optimal controller tuning is more conservative with increasing sampling time as the cost increases also. Some sampling times are relatively better than others, as 0.5 and 1.0, because they fit better the delay of 1. The best sampling time is the lowest, as it enables the controller the shortest response time on output changes.

2.12. CHAPTER SUMMARY

In this chapter we detailed the delays in the networked control system. The response of first order system to a PID controller is observed with constant delay.

Chapter -3

NETWORK SIMULATOR (NS-2)

3.1. INTRODUCTION

NS (version 2) is an object-oriented, discrete event driven network simulator developed at UC Barkley written in C++ and OTcl. NS is primarily useful for simulating local and wide area networks. NS is an event driven network simulator developed at UC Berkeley that simulates variety of IP networks. It implements network protocols such as TCP and UPD, traffic source behavior such as FTP, CBR, router queue management mechanism such as Drop Tail, RED and routing algorithms such as Dijkstra, and more. NS also implements multicasting and some of the MAC layer protocols for LAN simulations.

Another major component of NS beside network objects is the event scheduler. An event in NS is a packet ID that is unique for a packet with scheduled time and the pointer to an object that handles the event. In NS, an event scheduler keeps track of simulation time and fires all the events in the event queue scheduled for the current time by invoking appropriate network components, which usually are the ones who issued the events, and let them do the appropriate action associated with packet pointed by the event.

3.2. BACKGROUND ON THE NS SIMULATOR

NS simulator is based on two languages: an object oriented simulator, written in C++, and an OTcl (an object oriented extension of Tcl) interpreter, used to execute user's command scripts.

NS has a rich library of network and protocol objects. There are two class hierarchies: the complied C++ hierarchy and the interpreted OTcl one. With one to one correspondence between them.

The compiled C++ hierarchy allows us to achieve efficiency in the simulation and faster execution times. This is in particular useful for the detailed definition and operation of protocols.

Then in the OTcl script provided by the user, we can define a particular network topology, the specific protocols and applications that we wish to simulate and the form of the output that we wish to obtain from the simulator.

3.3. TCL AND OTCL

Tcl (Tool Command Language) is used for millions of people in the world. It is a language with a very simple sin taxis and it allows a very easy integration with other languages. Tcl was created by John Ousterhout.The characteristics of these language are:

- It allows a fast development
- It provide a graphique interface
- It is compatible with many platforms
- It is flexible for integration
- It is easy to use

Here are some basic of tcl and Otcl programming.

- Assigning a value to a variable is done through the "*set*" command; for example:"*set* b 0" assigns to 'b' the value of '0'. This is equivalent to"b=0".
- When we wish to use the value assigned to a variable, we should use a $ sign before the variable. For example, if we want to assign to variable 'x'the value that variable 'a' has, then we should write:"*set x $a"*.
- A mathematical operation is done using the expression command. For example, if wish to assign to a variable 'x' the sum of values of some variables 'a' and 'b', we should write "*set x [expr $a+$b]".*
- In Tcl the variables are not typed, so a variable can be a string or an integer depending on the value you assign to it.
- The sign # starts a commented line that is not part of the program, so the tcl interpreter will not execute this line.
- To create a file, one has to give it a name ,say "filename", and to assign a pointer to it that will be used within the tcl program in order to relate to it, say "file1".this is done with command: *set file1[open filename w]*.
- The command *puts* is used for printing an output. each time the "puts" command is used, a new line is started. To avoid new line, one has to add *–nonewline* after the "puts" command.

3.4. NS SIMULATOR PRELIMINARIES

3.4.1. Initialization and Termination

An ns simulator starts with the command

Set ns [new simulator]

Which is thus the first line in the tcl script? This line declares a new variable ns using the set command, you can call this variable as you wich,but in general people declares it as ns because it is an instance of the simulator class, so an object.teh code [new simulator] is indeed the instantiation of the class simulator using the reserved word new.

At the end of the ns program we should call the procedure "finish" and specify at what time the termination should occur. For example,

$ns at 125.0 "finish"

Will be used to call "finish" at time 125 sec. indeed, the at method of the simulator allows us to schedule events explicitly.

The simulation can begin using the command

$ns run

3.4.2. Definition of a Network of Links and Nodes

The way to define a node is

Set n0 [$ns node]

We created a node that is pointed by the variable n0. When we shall refer to that node in the script, we shall thus write $n0.

Once we define several nodes, we can define the links that connect them. An example of a definition of a link is:

$ns duplex-link $n0 $n2 10Mb 10ms Droptail

Which means that nodes $n0 and $n2 are connected using a bi-directional link that has 10ms of propagation delay and a capacity of 10 Mb/sec for each direction. To define a directional link instead of a bi-directional one, we should replace "duplex-link" by "simplex-link".

In NS, an output queue of a node is implemented as a part of each link whose input is that node. the definition of the link then includes the way to handle overflow at that queue. In our case, if the duffer capacity of the output queue is exceeded then the last packet to arrive is dropped (Drop Tail option).

3.4.3. Agents and Applications

Having defined the topology (nodes and links) we should now make traffic flow through them. To that end, we need to define routing (in particular sources, destinations) the agents (protocols) and applications that use them.

FTP over TCP:

TCP is a dynamic reliable congestion control protocol. It uses acknowledgements created by the destination to know whether packets are well received; lost packets are interpreted as congestion signals. The type of agent appears in the first line:

Set tcp [new Agent/TCP]

This command also gives pointer called "tcp" here to the TCP agent, which is an object in NS. The command *$ns attach-agent $n0 $tcp* defines the source node of the TCP connection.

The command set *sinks [new Agent/TCPsink]* defines the behavior of the destination node of TCP and assigns to it a pointer called sink. We note that in TCP the destination node has an active role in the protocol of generating acknowledgements in order to guarantee that all packets arrive at the destination.

CBR over UDP:

Next we define the UDP connection and the CBR application over it. A UDP source (Agent/UDP) and destination (Agent/Null) is defined in a similar way as in case of TCP. For the CBR application that uses UDP.

3.4.4. Scheduling Events

NS is a discrete event based simulation. The Tcl script defines when event should occur. The initializing command *set ns [new simulator]* create an event scheduler, and events are then scheduled using the format:

$ns at <time> <event>

The scheduler is started when running ns, i.e. through the command *$ns run*

In our simple example, we should schedule the beginning and of the FTP and the CBR application. This can be done through the following commands:

$ns at 0.1 "$cbr start"

$ns at 1.0 "$ftp start"

$ns at 124.0 "$ftp stop"

$ns at 124.5 "$cbr stop"

Thus the FTP will be active during time 1.0 till 124.0 and the CBR will be active during from time 0.1 till 124.5(all units are in seconds).

3.4.5. Visualizations: Using Nam

The visualization tool nam will display nodes network shown in fig 3.1. The location of the nodes could have been chosen at random.

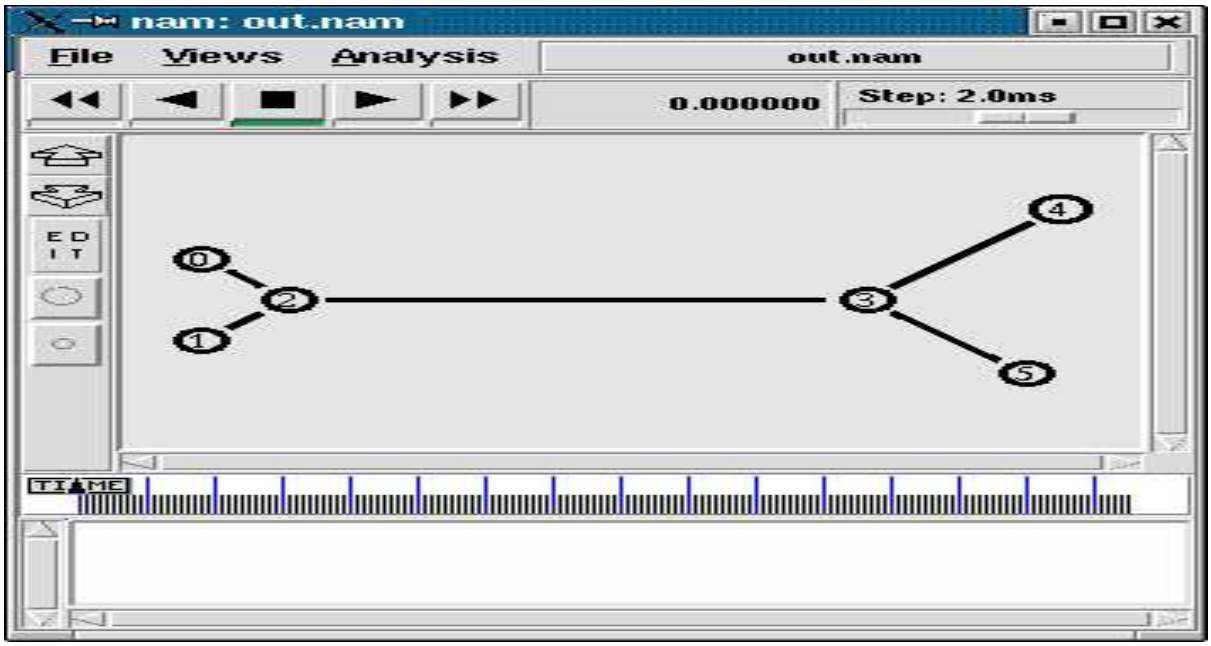


Fig.3.1. NAM graphic interface

We note e that the nam display shows us with animation the CBR packets (in figure 3.1 that from node 1 to5) in red, and TCP packets (flowing from node 0 to 4) in blue.TCP ACKs (acknowledgements) that go in the reverse directions are also in blue but are much shorter, since an ACK has a size of 40 bytes whereas the TCP packet is of size 552 bytes. To obtain the colors, we had to define in the beginning of our script exl.tcl.

$ns color 1 Blue

$ns color 2 Red

Features:

1. Integration of existing topology generators.
2. Localization and visualization of sets of nodes on large network topologies according to different selection criteria.
3. Instantiation of *agents* of any type on all the nodes of a given node set.
4. Customization of node parameters
5. Definition of new node types
6. support for simulation of web cache systems

3.4.6. Tracing

Tracing objects:

NS simulation can produce the visualization trace (for NAM) as well as ascii file trace corresponding to the events registered at the network. When we use tracing, ns inserts four objects in the link: EnqT, DeqT, RecvT and DrpT as indicated in fig 3.2

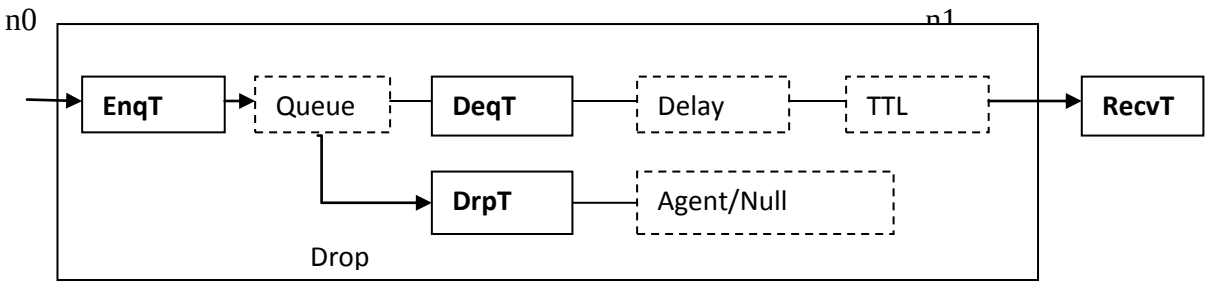


Fig.3.2. Tracing objects in a simplex link

EnqT registers information concerning a packet that arrives and is queued at the input queue of the link. If the packet overflows then information concerning the dropped packet are handled by DrpT, DeqT registers information at the instant the packet is dequed. Finally, RecvT gives us information about packets that have been received at the output of the link.

Structure of trace files:

When tracing into output ascii file, the trace is organized in 12 fields as follows in fig 3.3.The meaning of the fields are:

Event	Time	From node	To node	Packet type	Pkt size	Flags	Fid	Src addr	Dst addr	Seq num	Pkt id

Fig 3.3. Fields appearing in a trace

1. The first is the event type. It is given by one of four possible symbols r, +, - , d which correspond respectively to receive (at the output of the link), enqueued, dequeued and dropped.
2. The second field gives the time at which the event occurs.
3. Gives the input node of the link at which the event occurs.
4. Gives the output node of the link at which the event occurs.
5. Gives the packet type (for example, CBR, or TCP).
6. Gives the packet size.
7. Gives the some flags.
8. This is the flow id(fid) of IPv6 that a user can set for each flow at the OTcl script. One can further use this field for analysis purposes; it is also used when specifying stream color for the NAM display.
9. This is the source address given in the form of "node. Port".
10. This is the destination address, given in the same form.
11. This is the network layer protocol's packet sequence number.
12. The last field shows the unique id of the packet.

3.5. SIMULATION RESULTS

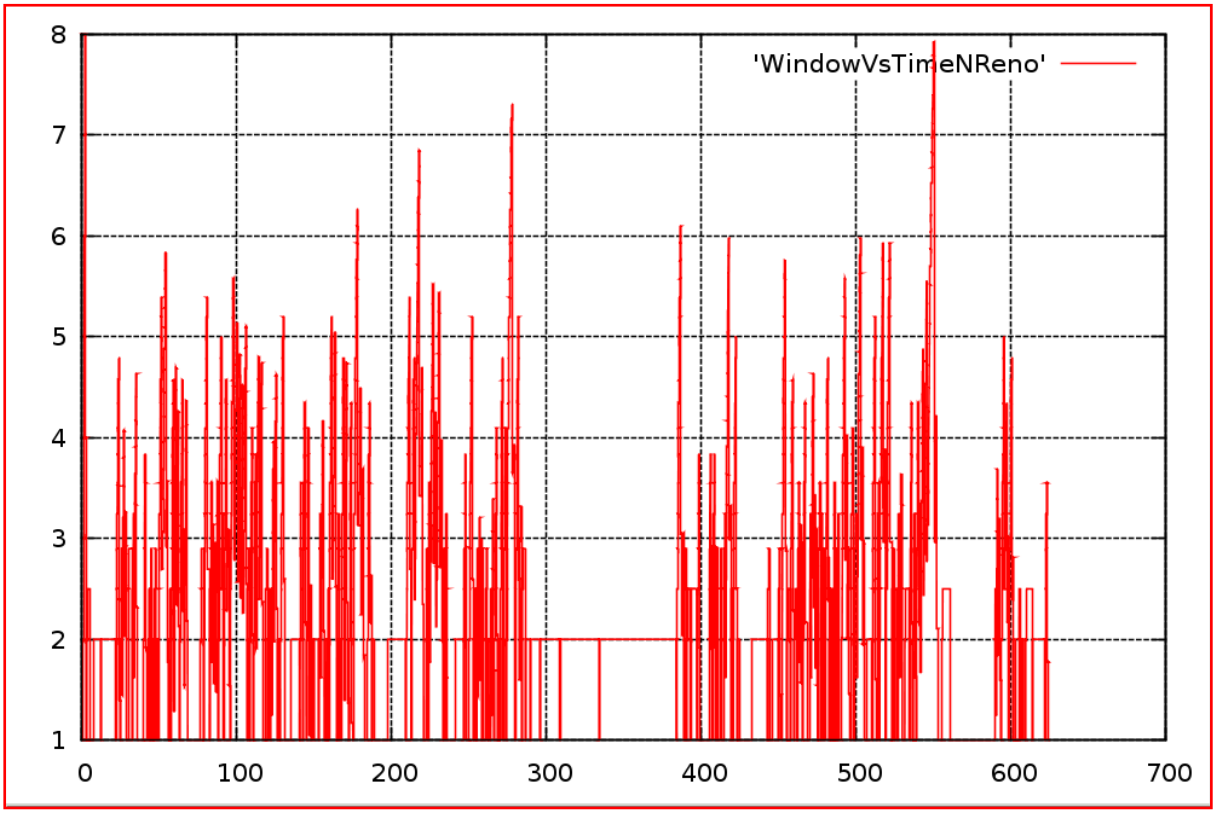


Fig 3.4. window size of TCP with 20% random loss

Fig 3.4. Shows the plot between the window and Time Reno created by the simulation. This simulation output will give when we introduce the simplest error model. we assume that packet are dropped on the forward link according independently with some fixed constant probability. This simulation graph is show the window size of TCP with 20% random loss. In this case we can observe long timeout, in particularly at time 300.To see the huge impact of the random loss on TCP performance.

3.6. CHAPTER SUMMARY

In this chapter the basic concepts of network simulator has been studied. The simulation results have been observed in the presence of a noisy link.

Chapter -4

MODELING THE DIGITAL SERVO MOTOR

4.1. INTRODUCTION

In this section we explained the mathematical model of DC servo motor, this modeling will give the transfer function of the servo. In, this case we are use the system identification toolbox in MATLAB (R2008) for modeling to estimate the transfer function of the servo motor.

4.2. MATHEMATICAL MODELING OF DC SERVO MOTOR

Electric motors are the most common actuator used in electromagnetic systems of all types. They are made in a variety of configurations and sizes for applications ranging from activating precision movements to powering diesel-electric locomotives. The laboratory motors are small servomotors, which might be used for positioning control applications in a variety of automated machines. They are DC (direct current) motors. The armature is driven by an external DC voltage that produces the motor torque and results in the motor speed. The armature current produced by the applied voltage interacts with the permanent magnet field to produce current and motion.

The servo DC motor is basically a torque transducer that converts electric energy into mechanical energy. The torque developed on the motor shaft is directly proportional to the field flux and the armature current.

The servo DC motor is used extensively in control systems, for analytical purpose, it is necessary to establish mathematical models for dc motors for applications. We use the equivalent circuit diagram in fig. (4.1) to represent a dc motor. The armature is modeled as a circuit with resistance R_a Connected in series with an inductance L_a and a voltage source e_b representing the back emf in the armature when the rotor rotates. The motors variables and parameters are defined as follows:

With reference to the circuit diagram of figure (4.1), the control of the DC motor is applied at the armature terminals in the form of applied voltage $e_a(t)$. For linear analysis, we assume that the torque developed by the motor is proportional to the air-gap flux and the armature current. Thus

$$T_m(t) = K_m(t)\phi i_a(t) \qquad \ldots 1$$

Since ϕ is constant, equation (1) is written

$$T_m(t) = K_i i_a(t)$$

Where K_i is the torque constant in N-m/A.

Starting with the control input voltage $e_a(t)$,the cause-and-effect equations for the motor circuit in fig (4.1) are

$i_a(t)$ = armature current

R_a = armature resistance

$e_b(t)$ = back emf

$T_L(t)$ = load torque

$T_m(t)$ = motor torque

$\theta_m(t)$ = rotor displacement

K_i = torque constant

L_a = armature inductance

$e_a(t)$ = applied voltage

K_b = back – emf constant

ϕ = magnetic flux in the air gap

$\omega_m(t)$ = rotor angular velocity

J_m = rotor inertia

B_m = viscous-friction coefficient

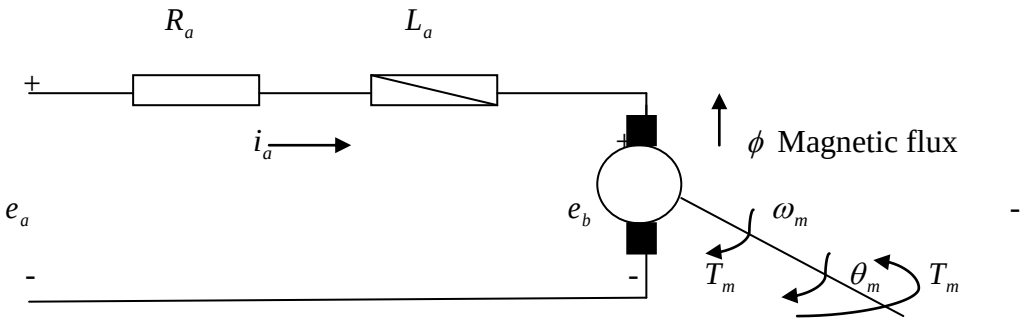


Fig.4.1. model of a DC motor

$$\frac{di_a(t)}{dt} = \frac{1}{L_a}e_a(t) - \frac{R_a}{L_a}i_a(t) - \frac{1}{L_a}e_b(t) \qquad \ldots\ldots \text{(i)}$$

$$T_m(t) = K_i i_a(t) \qquad \ldots\ldots\ldots\ldots\text{(ii)}$$

$$e_b(t) = K_b \frac{d\theta_m(t)}{dt} = K_b \omega_m(t) \qquad \text{..... (iii)}$$

$$\frac{d^2\theta_m(t)}{dt^2} = \frac{1}{J_m} T_m(t) - \frac{1}{J_m} T_L(t) - \frac{B_m}{J_m} \frac{d\theta_m(t)}{dt} \qquad \text{..... (iv)}$$

Where $T_L(t)$ represents a load frictional torque such as coulomb friction.

Equation (i) through (iv) consider that $e_a(t)$ is the cause of all causes; equation(i) consider that $di_a(t)/dt$ is the immediate effect due to the applied voltage $e_a(t)$,then in equation(ii), $i_a(t)$ Causes the torque $T_m(t)$,equation(iii) defines the back emf, and finally, in equation(iv),the torque $T_m(t)$ causes the angular velocity $\omega_m(t)$ and displacement $\theta_m(t)$.

The state variables of the system can be defined as $i_a(t)$, $\omega_m(t)$, and $\theta_m(t)$. By direct substitution and eliminating all the nonstate variables from eq(i) through (iv),the state equations of the dc-motor system are written in vector-matrix form:

$$\begin{bmatrix} \frac{di_a(t)}{dt} \\ \frac{d\omega_m(t)}{dt} \\ \frac{d\theta_m(t)}{dt} \end{bmatrix} = \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{K_b}{L_a} & 0 \\ \frac{K_i}{J_m} & -\frac{B_m}{J_m} & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{1}{L_a} \\ 0 \\ 0 \end{bmatrix} e_a(t) - \begin{bmatrix} 0 \\ \frac{1}{J_m} \\ 0 \end{bmatrix} T_L(t) \qquad \text{.....(vi)}$$

Using equation (vi),The transfer function between the motor displacement and the input voltage is obtained from the state diagram as

$$\frac{\theta_m(s)}{E_a(s)} = \frac{K_i}{L_a J_a s^3 + (R_a J_m + B_m L_a) s^2 + (K_b K_i + R_a B_m) s} \qquad \text{..... (vii)}$$

From equation (vii), the significance of the transfer function $\theta_m(s)/E_a(s)$ is that the dc motor is essential an integrating device between these two variables. This is expected ,since if $e_a(t)$ is a constant input, the output motor displacement will be behave as the output of integrator; that is, will increase linearly with time.

$$\frac{\theta_m(s)}{E_a(s)} = \frac{K_i / L_a J_m}{s^3 + \frac{(R_a J_m + B_m L_a)}{L_a J_m} s^2 + \frac{(K_b K_i + R_a B_m)}{L_a J_m} s} \quad \text{…….. (viii)}$$

$$\frac{\theta_m(s)}{E_a(s)} = \frac{b_0}{s^3 + a_2 s^2 + a_1 s} \quad \text{…….. (ix)}$$

Where

$$b_0 = K_i / L_a J_m$$

$$a_1 = \frac{K_b K_i + R_a B_m}{L_a J_m}$$

$$a_2 = \frac{(R_a J_m + B_m L_a)}{L_a J_m}$$

4.3. MODEL ESTIMATION PROCEDURE

The model estimation is generally done by using system identification toolbox in MATLAB. A system identification tool session represents the total progress of your identification process, including any data sets and models in the System Identification Tool GUI.

A typical workflow in the System Identification Tool GUI includes the following steps:

1. Import your data into the MATLAB workspace,
2. Start a new session in the System Identification Tool GUI, or open a saved session.
3. Import data into the GUI from the MATLAB workspace.
4. Plot and preprocess data to prepare it for system identification. For example, you can remove constant offsets or linear trends (for linear models only), filter data, or select data regions of interest.
5. Specify the data for estimation and validation.
6. Select the model to estimating using the Estimate menu.
7. Validate models.
8. Export models to the MATLAB workspace for further analysis.

4.3.1. Starting a New Session in the GUI

To start a new session in the System Identification Tool GUI, type the following command in the MATLAB Command Window:

Ident

Alternatively, you can start a new session by selecting **Start > Toolboxes > System Identification > System Identification Tool GUI** in the MATLAB desktop. This action opens the System Identification Tool GUI as shown in fig.4.2.

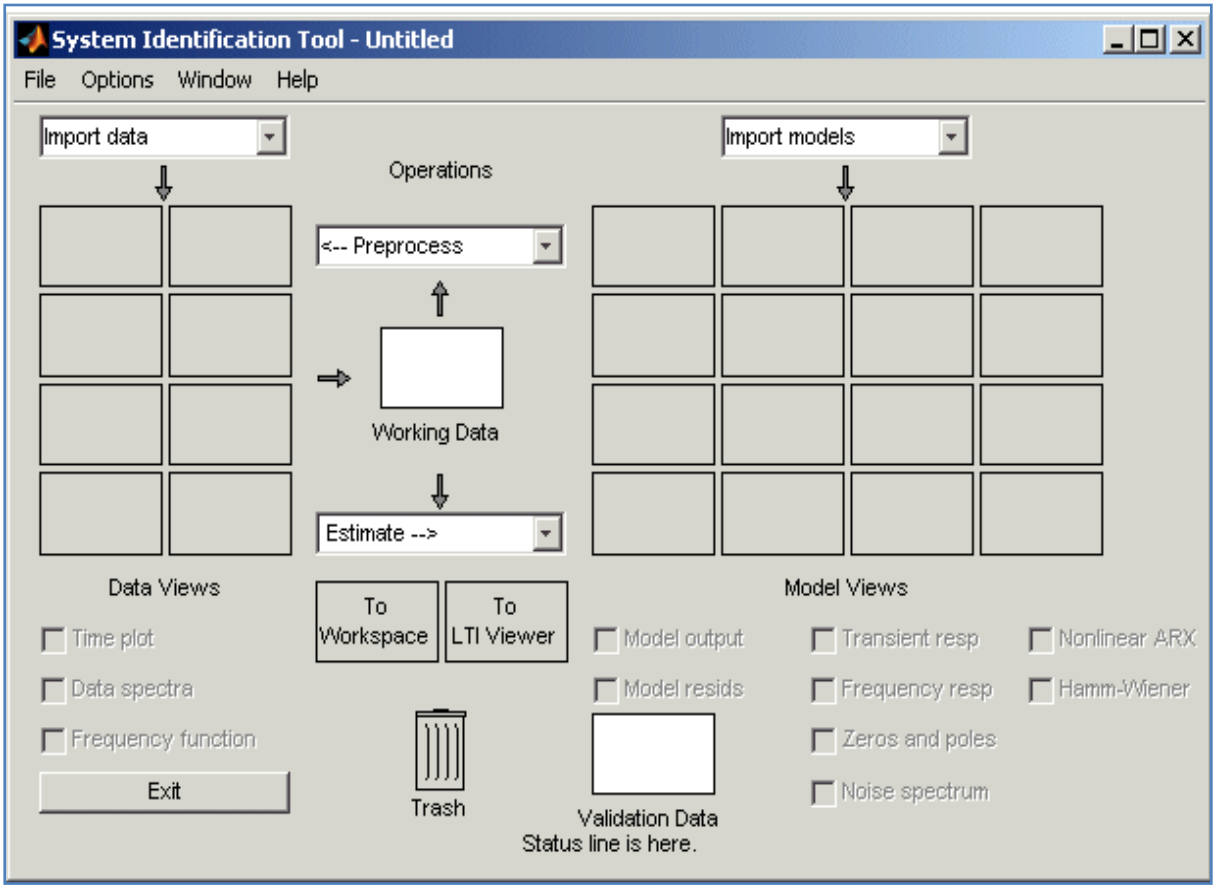


Fig 4.2. System identification GUI

You can also start a new session by closing the current session using **File > Close session**. This toolbox prompts you to save your current session if it is not already saved.

4.3.2. Description of the System Identification Tool Window

The following figure describes the different areas in the System Identification Tool GUI. The layout of the window organizes tasks and information from left to right as shown in fig.4.3.

This organization follows a typical workflow, where you start in the top-left corner by importing data into the System Identification Tool GUI using the **Import data** menu and end in the bottom-right corner by plotting the characteristics of your estimated model on model plots.

The **Data Board** area, located below the Import data menu in the System Identification Tool GUI, contains rectangular icons that represent the data you imported into the GUI.

The Model Board, located to the right of the <--Preprocess menu in the System Identification Tool GUI, contains rectangular icons that represent the models you estimated or imported into the GUI. You can drag and drop model icons in the Model Board into open dialog boxes

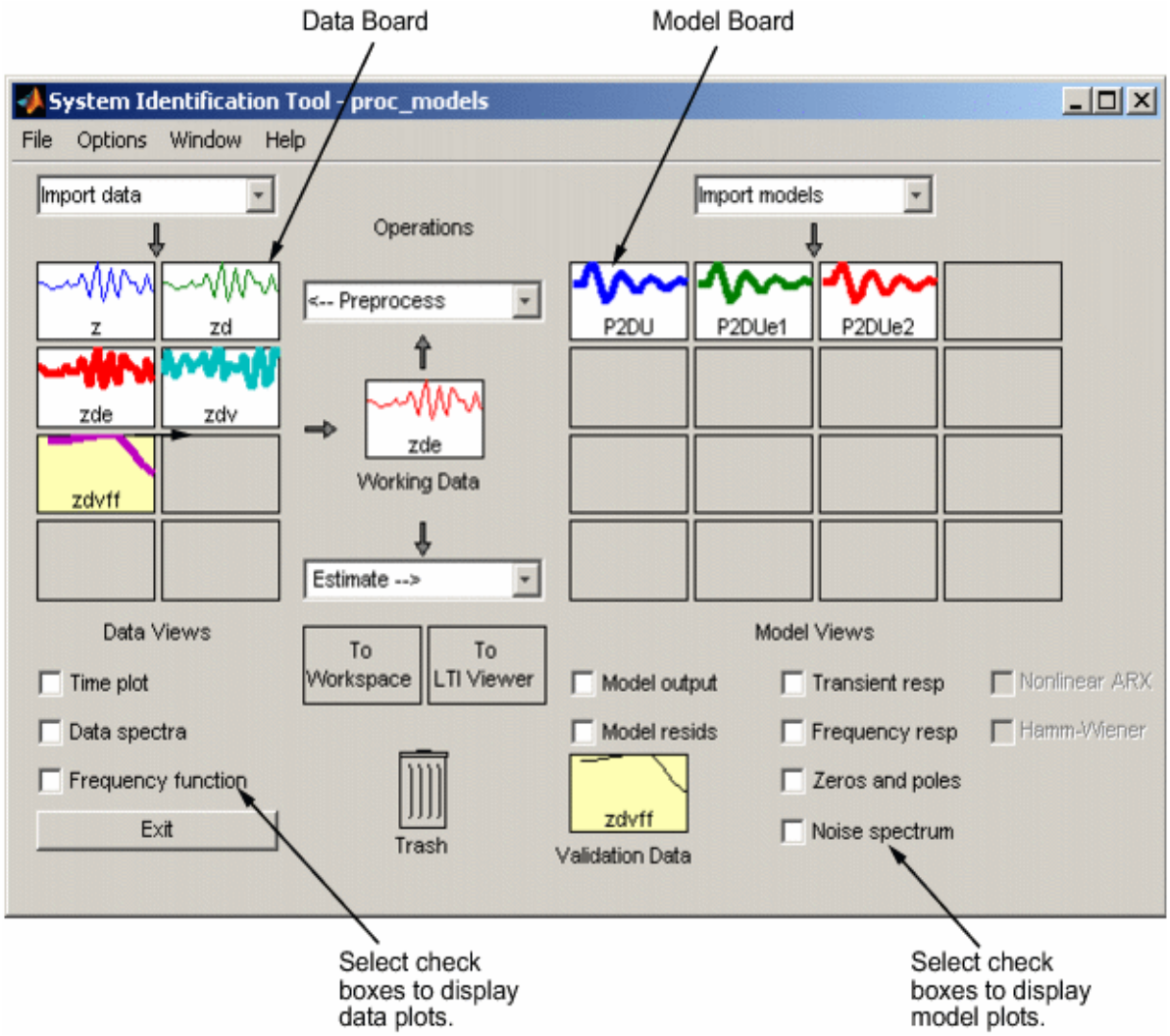


Fig 4.3. System Identification Tool Window

4.3.3. Importing Models into the GUI

- In the System Identification Tool GUI, select Import from the Import models list to open the Import Model Object dialog box.
- In the Enter the name field, type the name of a model object. Press Enter.
- Click Import.
- Click Close to close the Import Model Object dialog box.

4.3.4 Subspace Identification

Subspace identification algorithms are based on concepts from system theory, linear algebra and statistics. The main conceptual novelty in subspace identification algorithm is the *state of a dynamical system* is emphasized in the context of system identification, whereas "classical" approaches are based on an input-output framework.

The conceptual straightforwardness of subspace identification algorithms translates into *user-friendly software implementations*. To give only one example: Since there is no explicit need for parameterizations in our geometric framework, the user is not confronted with highly technical and theoretical issues such as canonical parameterizations, and hence, at the level of possible choices to be offered by the software. Where we describe the graphical user interface (**GUI**) software. In this Subsection, we confront the innovations in subspace identification with the properties of these "classical' approaches".

Parameterizations: When viewed as a data fitting problem, it becomes clear that system identification algorithms require a certain user-specified parameterization. In subspace identification algorithms we use full state space models and the only "parameter" is the order of the system. For classical algorithmic approaches however, there has been an extensive amount of research to determine so-called canonical models, i.e. models with a minimum number of parameters.

Convergence: When implemented *correctly*, subspace identification algorithms are fast, despite the fact that they are using QR and singular value decompositions. As a matter of fact, they are faster than the "classical" identification methods, such as Prediction Error Methods, because they are not iterative. Hence there are also no *convergence* problems. Moreover, numerical robustness is guaranteed precisely because of these well-understood algorithms

from numerical linear algebra. As a consequence, the user will never be confronted with hard-to-deal-with-problems such as lack of convergence, slow convergence or numerical instability.

Model reduction: Since one of our main interests lies in using the models in a computer aided control system design environment and because, when using linear theories, the complexity of the controller is proportional to the order of the system, one is always inclined to obtain models with as low an order as possible. In subspace identification, the reduced model can be obtained *directly*, without having to compute first the high order model, and this directly from input-output data

4.4. MODELING THE DIGITAL SERVO MOTOR

The model estimation of digital servo motor is generally done by using system identification tool box in MATLAB. In this modeling, we collect the output data (angular position) with corresponding input data (voltage) by performing experiment on it. Then we use these data for identifying the system model.

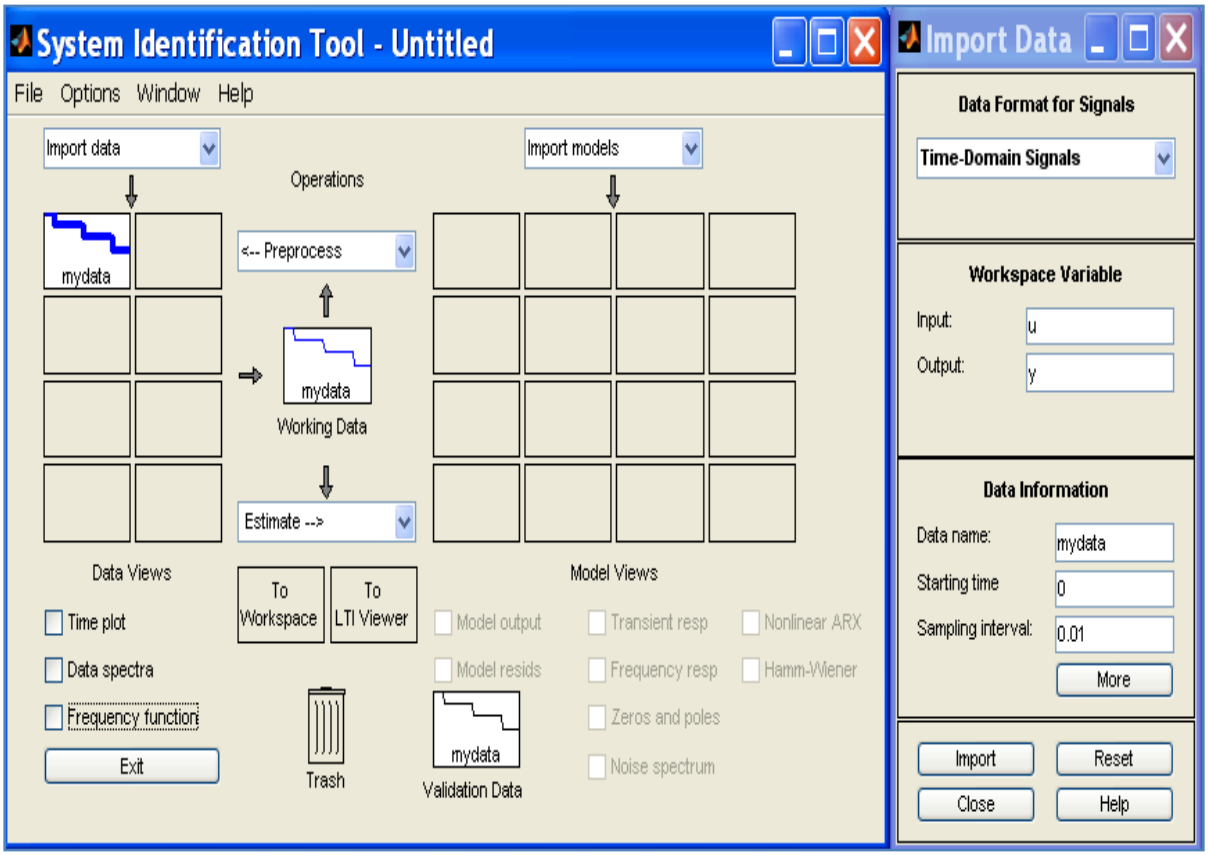


Fig.4.4. System identification tool window after importing the data

After collecting the data from DC servo model, we import the data on the system identification tool GUI, and then perform the process. This process is shown fig.4.4.

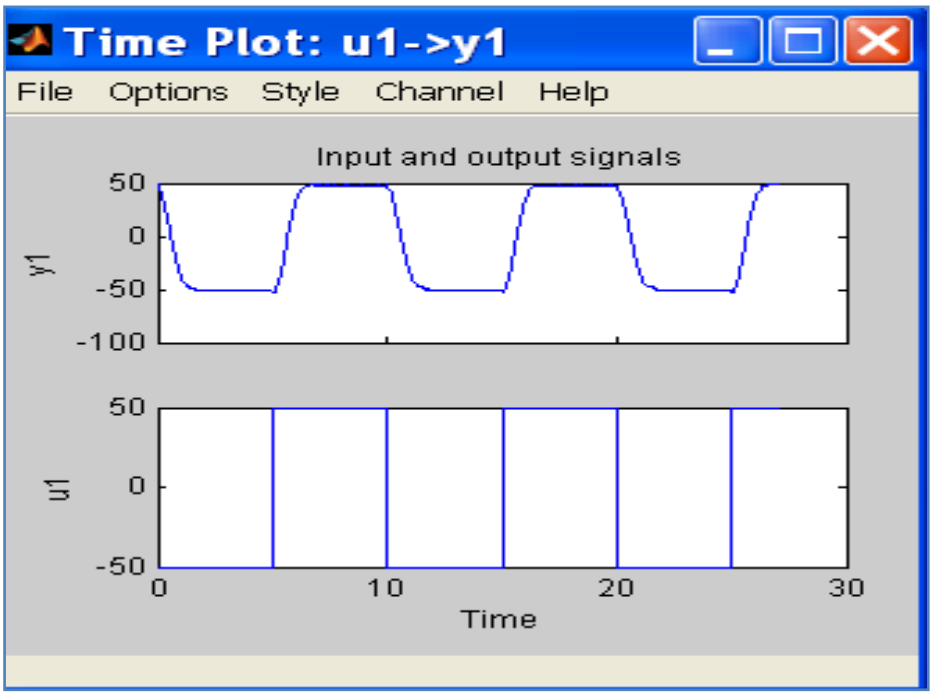


Fig .4.5.Time plot of an imported data

The time plot response of an imported data is given by the fig.4.5. Where u1 is the input and y1 is the output of the servo model and the transient response of the model is given fig.4.6.

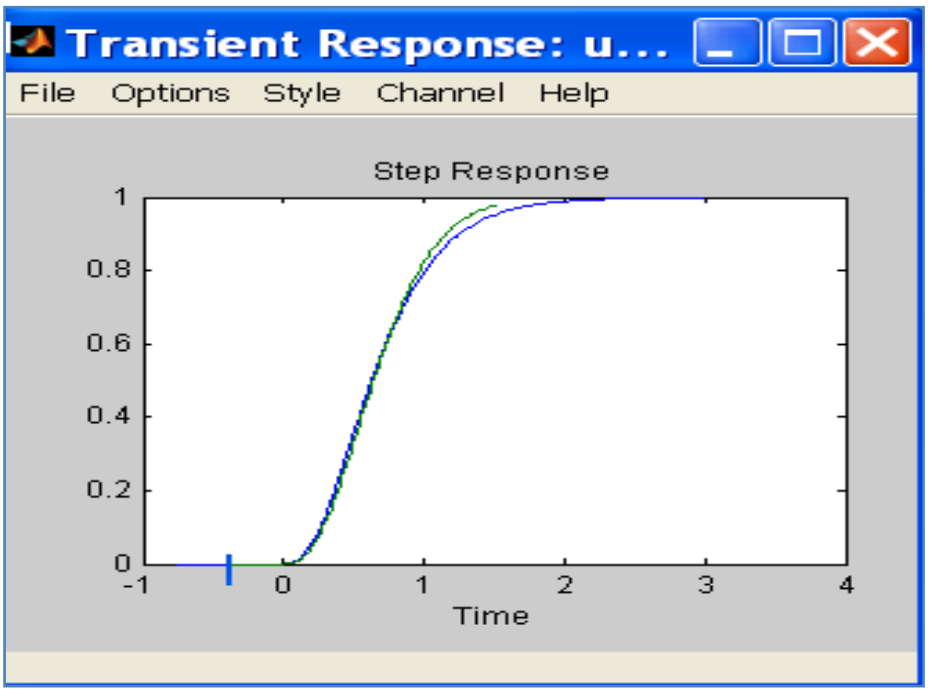


Fig.4.6.Transient response of the model

The System Identification Toolbox allows you to estimate simple continuous-time process models characterizing the static gain, dominating time constants, and possible time delays

(dead time). They are variants of the transfer function model structure as shown in fig.4.7. To estimate models of this kind, choose **Estimate > Process Models** in the ident window. This opens a dialog box as shown below.

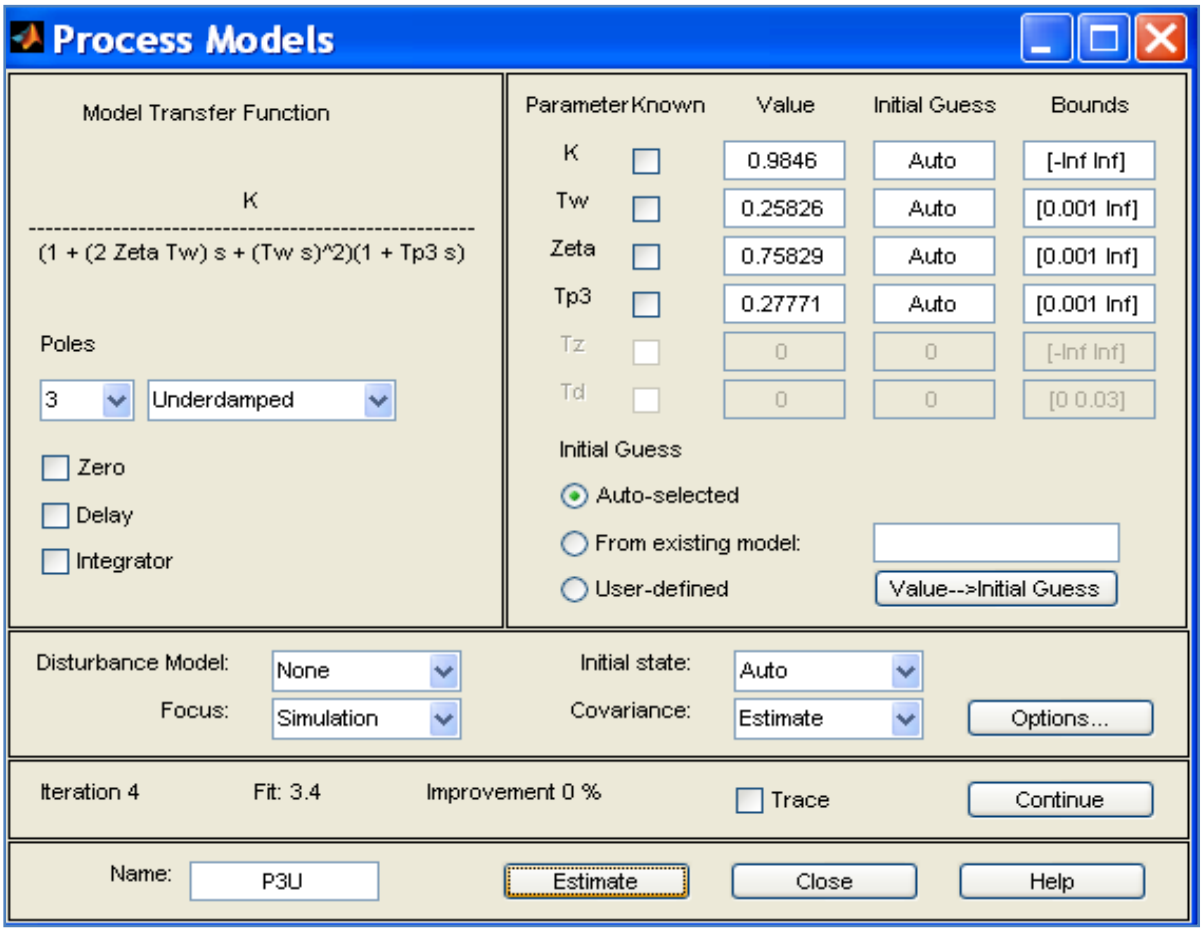


Fig.4.7. Estimation of process model

In this dialog you enter how many time constants (poles) to estimate and whether to include a time-delay term and an extra zero in the numerator of the transfer function. You can also enforce integration for self-regulating processes. Moreover, there is a choice to force all time constants to be real or to allow under damped modes (complex poles).from above case the process model is given by the transfer function of the servo motor is third order response.

The measured and simulated model output of the servo is given in fig.4.8. After we perform the process models we can measure the model output. The fig.4.8 shows where the output response follow exactly as the input response

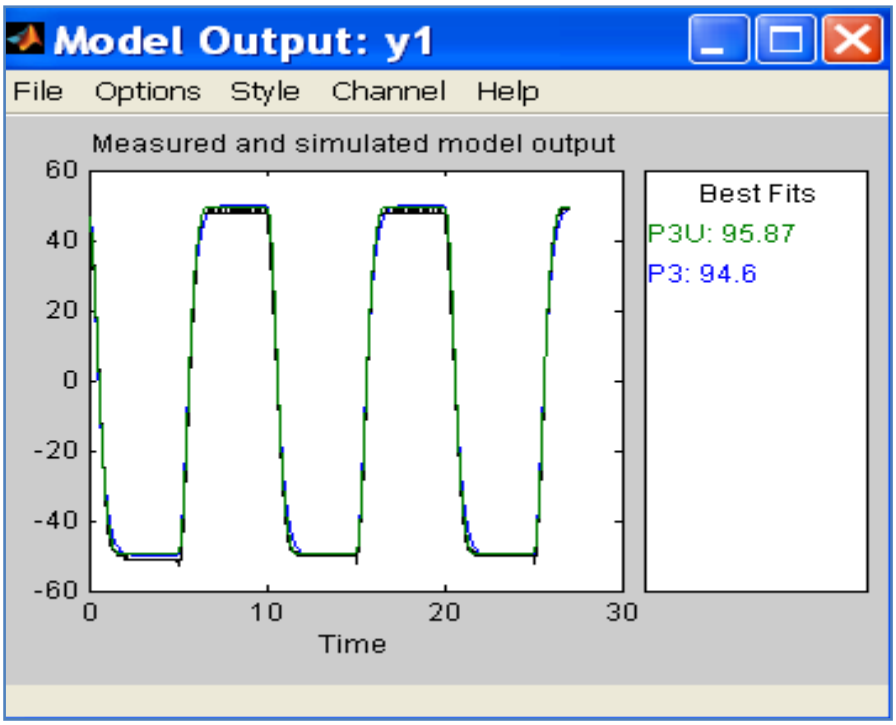


Fig.4.8. Measured and simulated model output

∴ The estimated DC servo motor parameters are given by the process models by using those parameters we find the transfer function. Finally, the transfer function of the DC servo motor is given by

$$\frac{\theta_m(s)}{E_a(s)} = \frac{53.2718}{(s^2 + 5.8726\,s + 14.9925)(s + 3.6088)}$$

The above transfer function is the closed loop transfer function of the servo system, and then the open loop response of the servo motor is given by

$$\frac{\theta_m(s)}{E_a(s)} = \frac{53.2718}{s^3 + 9.4814\,s^2 + 36.1855\,s + 0.6922}$$

4.5. CHAPTER SUMMARY

This chapter provides mathematical model of a DC servo motor. A brief explanation of model estimation procedure is provided here which is a part of system identification. The main objective is to find the transfer function of DC servo motor using system identification tool box in MATLAB (version 7.6).

Chapter -5

HARDWARE-IN-LOOP SIMULATION

5.1. INTRODUCTION

In this section we detail the hard-ware in loop simulation with PID controller and smith predictor. Smith predictor structure to compensate process time delay, this structure contains the process mathematical model in the feedback loop. These all simulation works done through Local Area Network (LAN).UDP (user datagram protocol) is used to perform this simulation through LAN and detail explanation of this protocol.

5.2. UDP (USER DATAGRAM PROTOCOL)

The User Datagram Protocol (UDP) is one of the core members of the Internet Protocol Suite, the set of network protocols used for the Internet. With UDP, computer applications can send messages, in this case referred to as *datagrams*, to other hosts on an Internet Protocol (IP) network without requiring prior communications to set up special transmission channels or data paths. UDP is sometimes called the Universal Datagram Protocol. The protocol was designed by David P. Reed in 1980.

The service provided by UDP is an unreliable service that provides no guarantees for delivery and no protection from duplication (e.g. if this arises due to software errors within an Intermediate System (IS)). The simplicity of UDP reduces the overhead from using the protocol and the services may be adequate in many cases.

UDP uses a simple transmission model without implicit hand-shaking dialogues for guaranteeing reliability, ordering, or data integrity. Thus, UDP provides an unreliable service and datagrams may arrive out of order, appear duplicated, or go missing without notice. UDP assumes that error checking and correction is either not necessary or performed in the application, avoiding the overhead of such processing at the network interface level. Time-sensitive applications often use UDP because dropping packets is preferable to using delayed packets. If error correction facilities are needed at the network interface level.

UDP provides a minimal, unreliable, best-effort, message-passing transport to applications and upper-layer protocols. Compared to other transport protocols, UDP and its UDP-Lite variant are unique in that they do not establish end-to-end connections between communicating end systems. UDP communication consequently does not incur connection.

Establishment and teardown overheads and there is minimal associated end system state. Because of these characteristics, UDP can offer a very efficient communication transport to

some applications, but has no inherent congestion control or reliability. A second unique characteristic of UDP is that it provides no inherent On many platforms, applications can send UDP datagram's at the line rate of the link interface, which is often much greater than the available path capacity, and doing so would contribute to congestion along the path, applications therefore need to be designed responsibly (RFC 4505).

One increasingly popular use of UDP is as a tunneling protocol, where a tunnel endpoint encapsulates the packets of another protocol inside UDP datagrams and transmits them to another tunnel endpoint, which decapsulates the UDP datagrams and forwards the original packets contained in the payload. Tunnels establish virtual links that appear to directly connect locations that are distant in the physical Internet topology, and can be used to create virtual (private) networks. Using UDP as a tunneling protocol is attractive when the payload protocol is not supported by middleboxes that may exist along the path, because many middle boxes support UDP transmissions.

UDP does not provide any communications security. Applications that need to protect their communications against eavesdropping, tampering, or message forgery therefore need to separately provide security services using additional protocol mechanisms.

UDP's stateless nature is also useful for servers that answer small queries from huge numbers of clients. Unlike TCP, UDP is compatible with packet broadcast (sending to all on local network) and multicasting (send to all subscribers).

UDP applications use datagram sockets to establish host-to-host communications. Sockets bind the application to service ports that function as the endpoints of data transmission. A port is a software structure that is identified by the port number, a 16 bit integer value, allowing for port numbers between 0 and 65,535. Port 0 is reserved, but is a permissible source port value if the sending process does not expect messages in response.

5.2.1. Reliability and congestion control solutions

Lacking reliability, UDP applications must generally be willing to accept some loss, errors or duplication. Some applications such as TFTP may add rudimentary reliability mechanisms into the application layer as needed. Most often, UDP applications do not require reliability mechanisms and may even be hindered by them. Streaming media, real-time multiplayer games and voice over IP (VoIP) are examples of applications that often use UDP. If an

application requires a high degree of reliability, a protocol such as the Transmission Control Protocol or erasure codes may be used instead.

Lacking any congestion avoidance and control mechanisms, network-based mechanisms are required to minimize potential congestion collapse effects of uncontrolled, high rate UDP traffic loads. In other words, since UDP senders cannot detect congestion, network-based elements such as routers using packet queuing and dropping techniques will often be the only tool available to slow down excessive UDP traffic. The Datagram Congestion Control Protocol (DCCP) is being designed as a partial solution to this potential problem by adding end host TCP-friendly congestion control behavior to high-rate UDP streams such as streaming media.

5.2.2 Applications

While the total amount of UDP traffic found on a typical network is often in the order of only a few percent numerous key Internet applications use UDP, including: the Domain Name System (DNS), where queries must be fast and only consist of a single request followed by a single reply packet, the Simple Network Management Protocol (SNMP), the Dynamic Host Configuration Protocol (DHCP) and the Routing Information Protocol (RIP).

Voice and video traffic is generally transmitted using UDP. Real-time video and audio streaming protocols are designed to handle occasional lost packets, so only slight degradation in quality occurs rather than large delays if lost packets are retransmitted. Because both TCP and UDP run over the same network, many businesses are finding that a recent increase in UDP traffic from these real-time applications is hindering the performance of applications using TCP, such as point of sale, accounting, and database systems. When TCP detects packet loss, it will throttle back its data rate usage. Since both real-time and business applications are important to businesses, developing quality of service solutions is crucial.

5.3. MEASUREMENT OF DELAY IN FEEDBACK LOOP

In this section we are find the process delay in feedback loop, this delay measurement will perform using two computer PC’s. One is acts a host pc and anther is acts as remote pc, these both PC’s are connected through a Local Area Network (LAN) and communicating each other as shown fig.5.1.

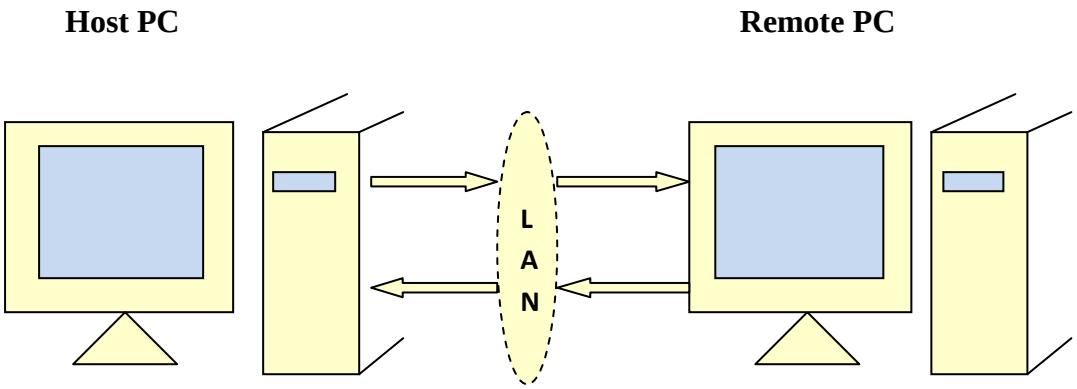


Fig.5.1. Communicating two PC's through LAN

Figure 5.2 shows the transmitting simulink block, this block put in host PC. For transmitting the signal from one system to another we are using the UDP protocol. These blocks will be available in MATLAB/simulink.UDP blocks mostly used for transmitting a signal or any data from one system to another through a LAN. These blocks works based on IP address of the systems.

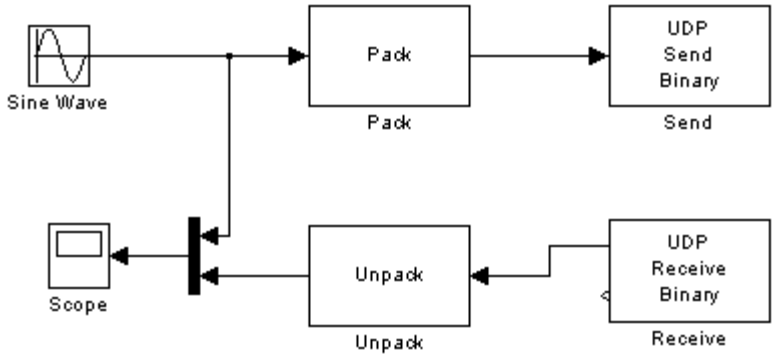


Fig 5.2. Transmitting simulink blocks in host PC

The sinusoidal signal of amplitude 1v and frequency of 1 Hz is applied as input, shown in fig.5.2. The simulink block of pack is used to convert one or more Simulink signals of varying data types to a single vector of uint8 as required by the Send block. This block is the exact analog of the Pack block. It receives a vector of uint8 and outputs various Simulink data types in different sizes. The Pack block is on the sending side and the Unpack block is on the receiving side in different models.

UDP send: The Send block has only one input port, which receives the uint8 vector that is sent as a UDP packet fig 5.3.

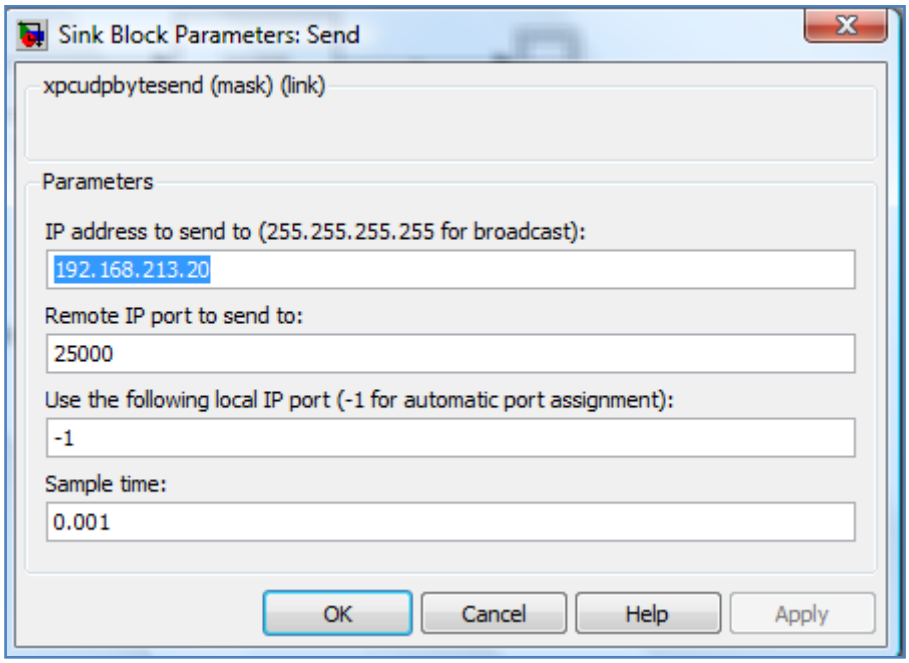


Fig.5.3. UDP send block parameters

Block Parameters:

- *IP address to send to:* Specify the IP address to send the packet.
- *IP port to send to:* Specify the port to which to send the packet.
- *Use the following local IP port:* Set this parameter to -1 (default) to allow the networking stack to automatically determine the local IP port that is used for sending. Otherwise, specify a particular port to send a packet from that port.
- *Sample time:* You can set this parameter to -1 for an inheritable sample time, but it is recommended that this be set to some specific (large) value to eliminate chances of dropped packets. This is especially true when you are using a small base sample time.

UDP receive:

The Receive block has two output ports. The first port is the output of the received data as a vector of uint8 while the second one is a flag indicating whether any new data has been received. This port outputs a value of 1 for the sample when there is new data and a 0

otherwise. The default behavior of the Receive block is to keep the previous output when there is no new data. You can modify this behavior by using the second port to flag when there is new data. as shown in fig.5.4.

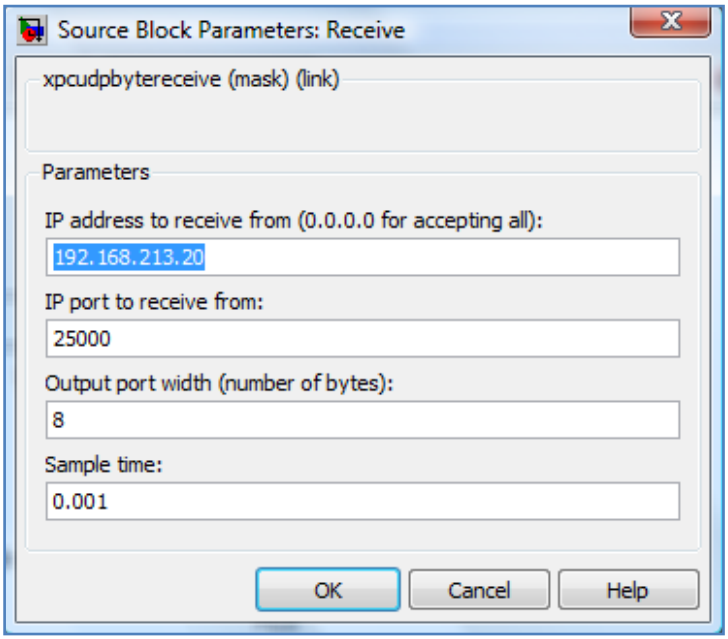


Fig.5.4. UDP receive block parameters

Block Parameters:

- *IP address to receive from:* If set to a specific IP address, only packets arriving from that IP address are received.
- *IP port to receive from:* Port that the block accepts data from. The other end of the communication sends data to the port specified here.
- *Output port width:* Width of the acceptable packets. You can obtain this when designing the other side (send side) of the communication.

Figure 5.5 shows the receiving signal block, this block put in remote PC. This block will be receiving whatever signal transmitted by the host PC. In receiving block all the blocks are explained in UDP sender session except the terminator. The Terminator block can be used to cap blocks whose output ports are not connected to other blocks. If you run a simulation with blocks having unconnected output ports, Simulink software issues warning messages. Using Terminator blocks to cap those blocks avoids warning messages.

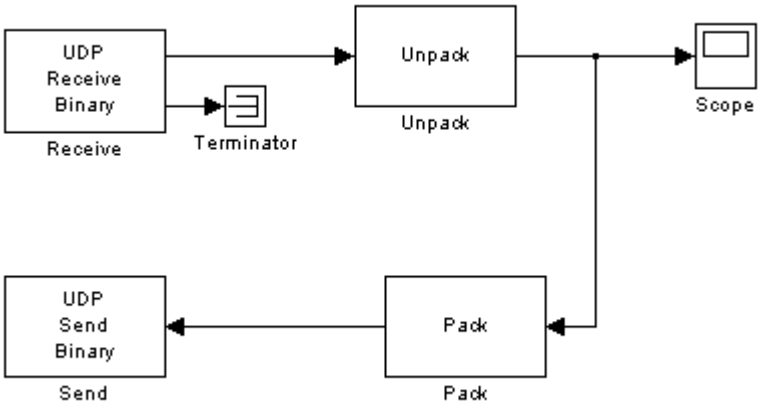


Fig 5.5. Receiving simulink blocks in remote PC

For measuring the delay in feedback loop, first we put the simulink blocks of sending and receiving in host PC, remote PC respectively. This environment will form a closed loop. After that we run these simulink blocks in host PC and remote PC at same time. After the completion of executing, we observe the presence of delay when comparing the input signal to the output as shown fig 5.6.

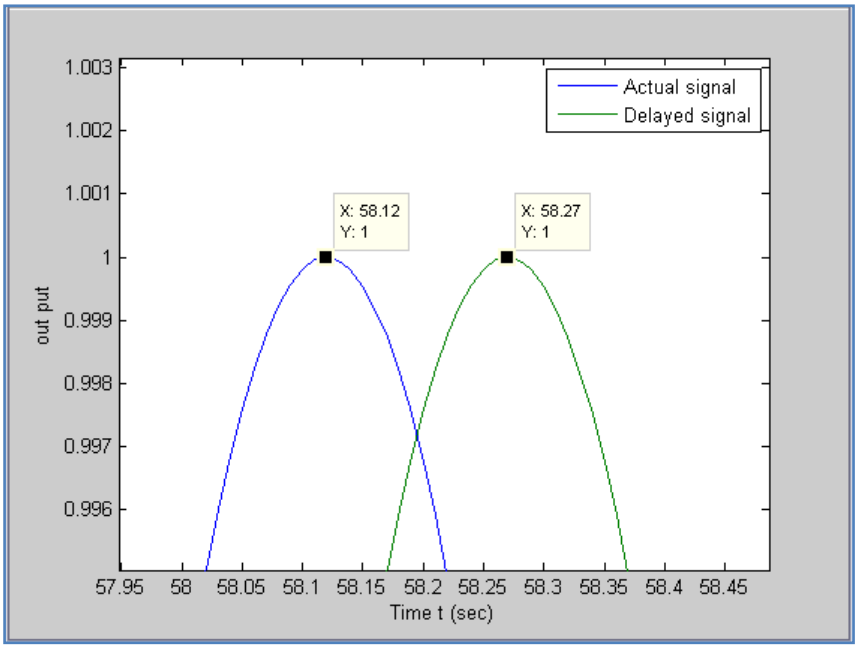


Fig.5.6. Measurement of Delay in Feedback loop

Fig 5.6, we consider the x-axis as time in sec and y-axis as an output (amplitude of the signal). The output will measure at remote PC as shown in fig.5.1.From the graph the first signal is the actual signal and the second one is delayed signal. We observe from the graph

that delay is occurred compared to the actual signal. This delay will present when the data packets are loss in between the host PC and remote PC communicating each other. This delay also called as process delay. From the fig.5.6 the delay is 0.15 sec

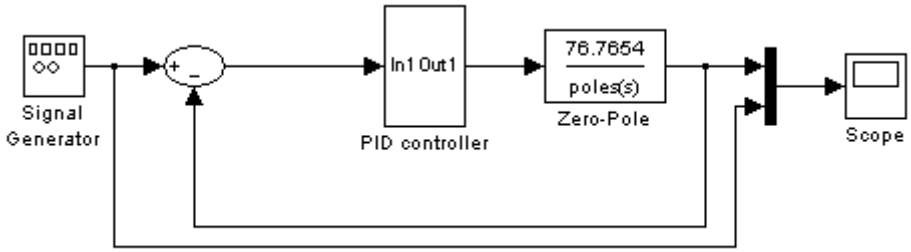


Fig.5.7. Simulink model of a system without network

Simulink model of system without network is given by fig.5.7.The PID controller is designed for the servo plant. The signal generator is connected as input which generates a square wave signal of amplitude 30v and sampling period of 0.01 sec.The step response of the simulink model of a system without network is given by in fig.5.8.

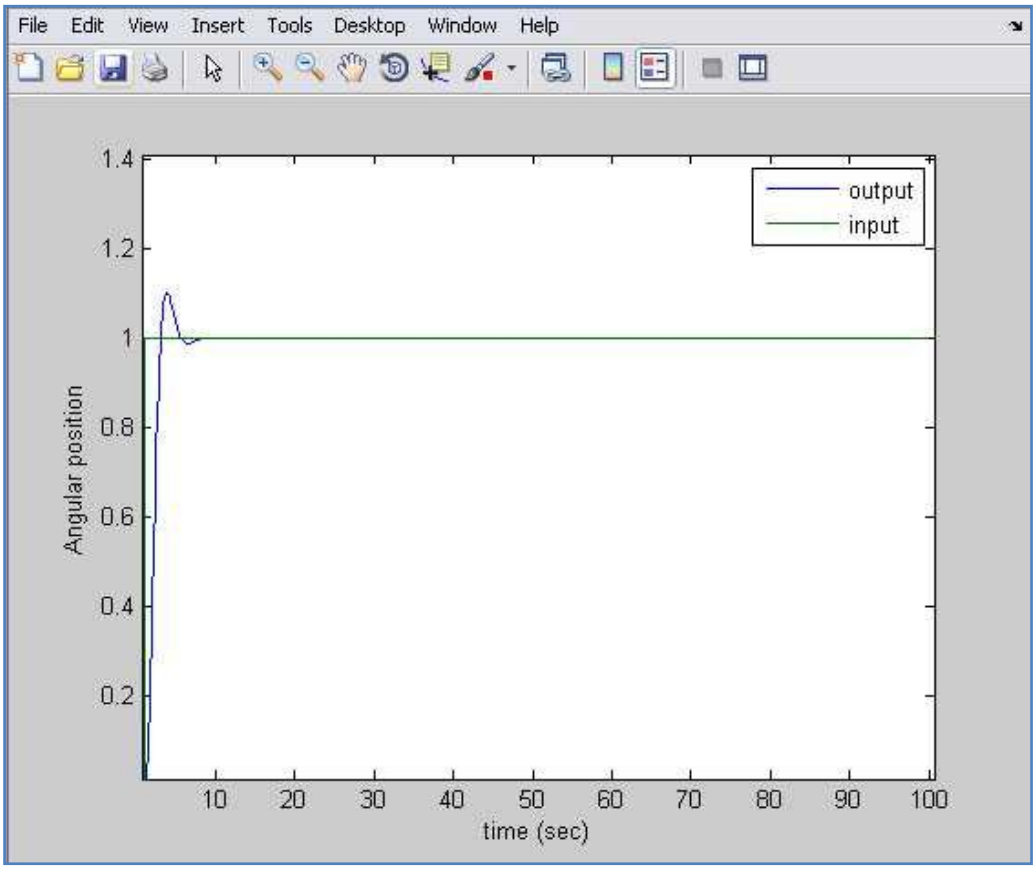


Fig.5.8. Step response of simulink model of a system without network

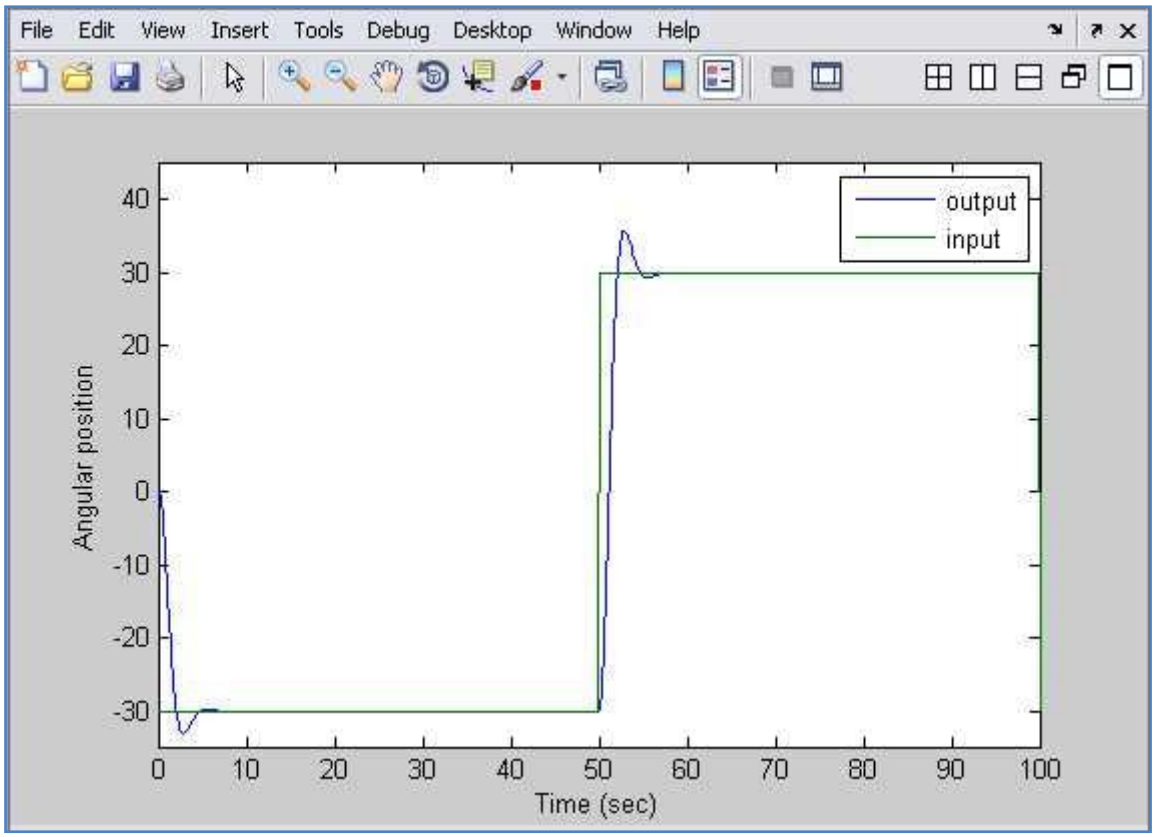


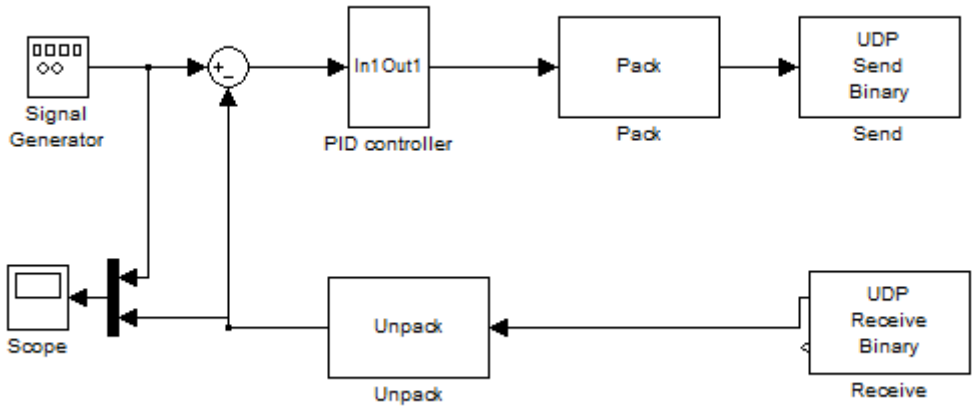
Fig.5.9. Response of simulink model of a system without network

The graph shown in above fig. 5.9 represents angular position of servo system. This graph compares generated actual signal and angular position (output) of the model.

Simulink model of a system with network is given by fig.5.10. The Sending simulink model with PID controller is put in host PC. The PID controller is connected to UDP send block through pack. The UDP send block sends the controlled signal to a remote PC. The received controlled signal is passed through the servo model and its output is feedback to host PC through UDP send block. This servo model is given in chapter- 4 is identified by using system identification tool box. The PID controller is present in the host PC which is designed for the servo model. The signal generator generates a square wave of amplitude 30 v sampling period of 0.001 sec.

Thus the total environment looks as if a closed-loop networked control system.

Host PC (send)



Remote PC (Receive)

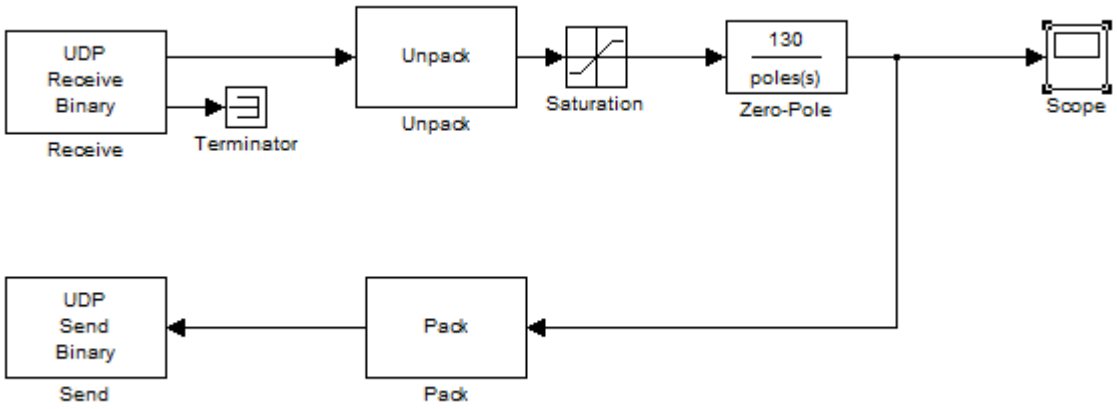


Fig.5.10. simulink model of a system with network

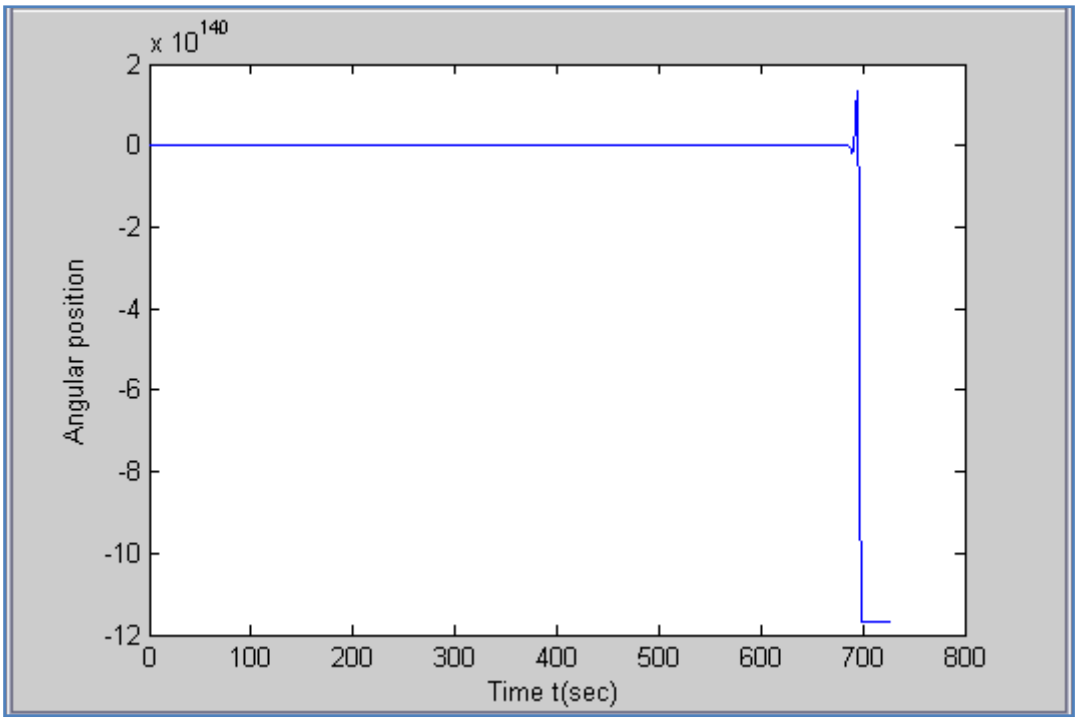


Fig.5.11.Response of simulink PID model with network.

The above figure 5.11 shows the instability of the PID controller used in a networked control system. The same controller response tends to stability while used in without network shows as fig.5.10. The instability of the controller in a network can be overcome by using a smith controller.

5.4. SMITH PREDICTOR

O.J Smith developed in 1950 controller structure to compensate process time delay. The Smith Predictor controller structure contains the process mathematical model in the feedback loop [4]. The Smith Predictor consists of building a corrector which virtually hides the time delay in the closed loop response of the process. It is basically a mix of a PID corrector with an internal model. The aim of the corrector is to provide a virtual system without time delay to the PID. But the idea can be generalized to all control processes that have long loop delays. Figure 5.12 shows a block diagram of the scheme, in which a plant lies in a negative-feedback loop with both feed forward and feedback delays. The Smith Predictor is very simple. The controller operates on two separate models of the plant, both lying on internal feedback loops.

Smith predictor structure to compensate systems with time delay, which are a feature of many industrial processes. The Smith predictor structure utilizes a mathematical model of the process in a minor feedback loop. One of its advantages is that the Smith predictor approach for compensating a Single Input Single output (SISO) process may be directly extended to the compensation of a Multiple Input Multiple Output (MIMO) process with the same delay in each path.

Since the Smith Predictor structure was proposed, many modifications have been proposed to improve the servo response, the regulator response or both. Modifications were accomplished to adapt the structure to stable, integrative or unstable systems. Note that there are two feedback loops. The outer control loop feeds the output back to the input, as usual. However, this loop alone would not provide satisfactory control, because of the delay; this loop is feeding back outdated information. Intuitively, for the k seconds during which no fresh information is available, the system is controlled by the inner loop which contains a predictor of what the (unobservable) output of the plant G currently is.

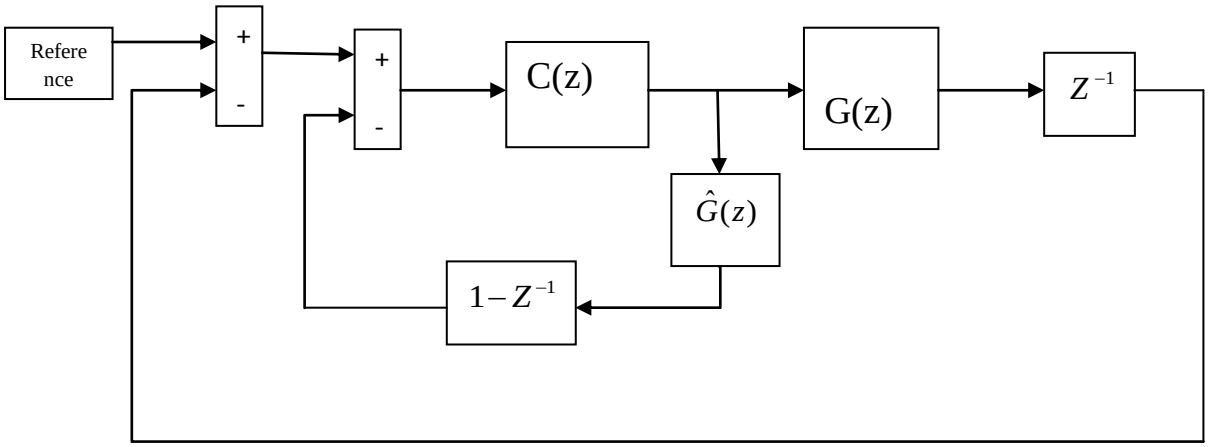
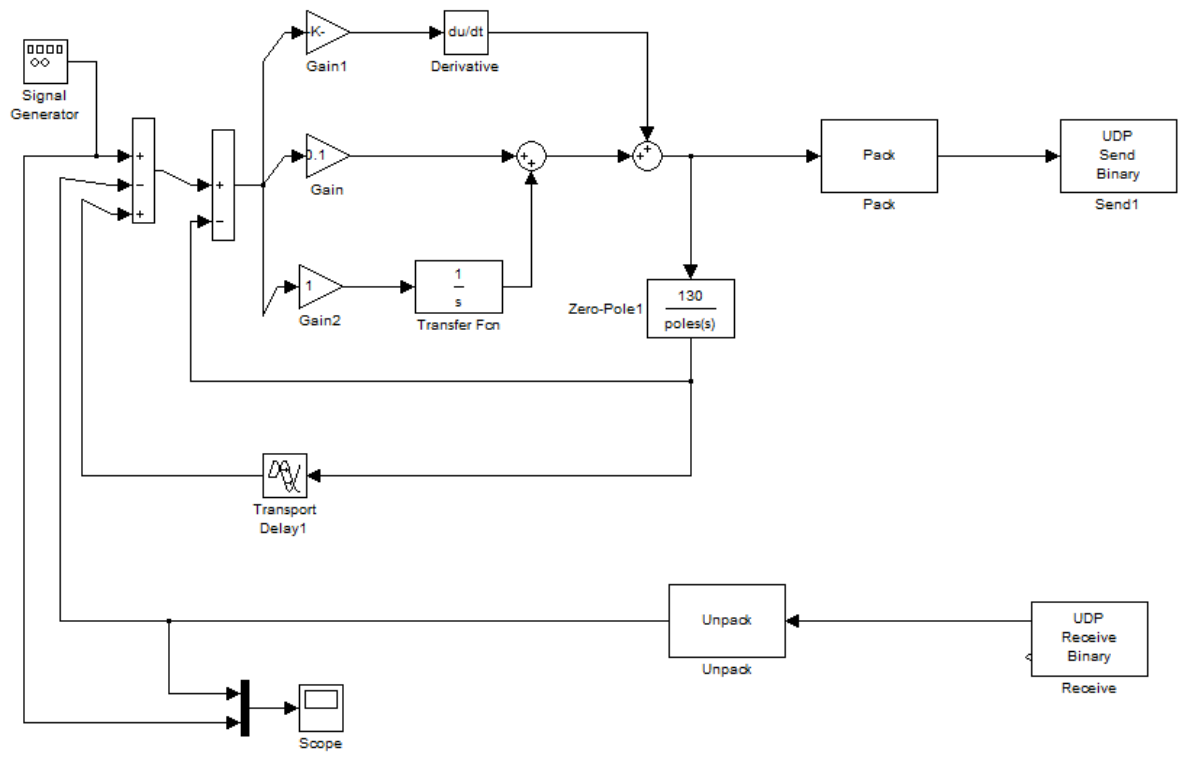


Fig.5.12. Block diagram of smith predictor

If this model is accurate, and the plant performance reliable, this loop can provide near optimal control of the plant. The second model is used to compare the actual performance of the plant with the expected performance. Because the second model includes an accurate representation of all plant transport delays, it will delay the output from the controller to match the delayed feedback from the periphery, and these two temporally matched signals normally cancel out.

Host PC (send)



Remote PC (Receive)

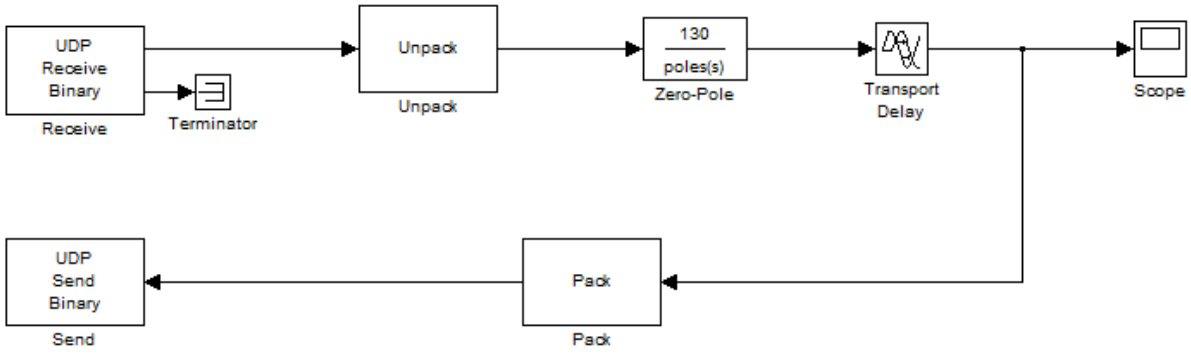


Fig.5.13. Simulink model of a system with smith predictor in NCS

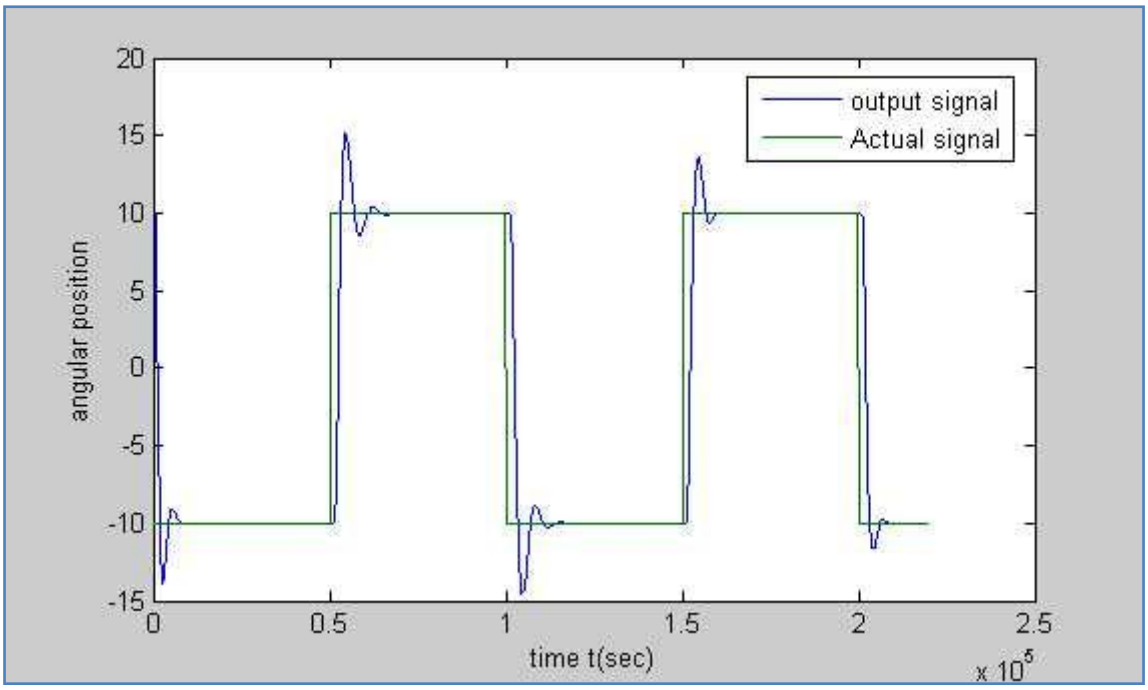


Fig.5.14.Responce of the smith predictor with network

The graph shown in below fig. 5.14 represents angular position of servo system. This graph compares generated actual signal and angular position (out signal). This delay is due to the occurrence of packet loss in UDP protocol. The service provided by UDP is an unreliable service that provides no guarantees for delivery of information interms of packets. The different UDP senders and receivers as shown in fig 5.12 may not synchronize themselves and hence the tilt and delay in the delayed signal. This delay also known as “process delay”. The instability of the PID controller is stabilized by using the smith predictor as shown in fig.5.14.

5.5. CHAPTER SUMMARY

This chapter gives details about the hardware-in-loop simulation which is a model experiment setup done, before performing experiment on DC servo setup. The chapter gives information on UDP protocol which is useful for communication between two systems. Measurements of delay in feedback loop, stability of PID controller are discussed here. A brief explanation of smith predictor which is used to a find the stability of a system and compensation of delay is also included here.

Chapter -6

EXPERIMENT ON THE DIGITAL SERVO MOTOR SET-UP WITH ARTIFICIAL DELAY BLOCK

6.1. INTRODUCTION

In section we perform some experiments using artificial delays. These delays makes a sense that there is a network .we observed the instability of the system due to delays and compensate the effect of delay using smith predictor. This smith predictor gives a compensated results compaired to the conventional PID controller.

6.1.1. Overview

A servo is a motor that is attached to a position feedback device. Generally there is a circuit that allows the motor to be commanded to go to a specified "position". A very common use of servos is in Radio Controlled models. Servos are extremely useful in robotics. The motors are small and are extremely powerful for their size. A standard servo such as the Futaba S-148 has 42 z/inches of torque, which is pretty strong for its size. It also draws power proportional to the mechanical load. A lightly loaded servo, therefore, doesn't consume much energy. Servos are constructed from three basic pieces, a motor, a feedback device, and a control board. In R/C servos the feedback device is typically a potentiometer (variable resistor). The motor, through a series of gears, turns the output shaft and the potentiometer simultaneously. The potentiometer is fed into the servo control circuit and when the control circuit detects that the position is correct, it stops the motor.

6.1.2. Working Procedure of Servo

The servo motor has some control circuits and a potentiometer (a variable resistor, aka pot) that is connected to the output shaft. The potentiometer allows the control circuitry to monitor the current angle of the servo motor. If the shaft is at the correct angle, then the motor shuts off. If the circuit finds that the angle is not correct, it will turn the motor the correct direction until the angle is correct. The output shaft of the servo is capable of travelling somewhere around 180 degrees. Usually, it’s somewhere in the 210 degree range, but it varies by manufacturer. A normal servo is used to control an angular motion of between 0 and 180 degrees. A normal servo is mechanically not capable of turning any farther due to a mechanical stop built on to the main output gear. The amount of power applied to the motor is proportional to the distance it needs to travel. So, if the shaft needs to turn a large distance, the motor will run at full speed. If it needs to turn only a small amount, the motor will run at a slower speed. This is called proportional control. The control wire is used to communicate the angle. The angle is determined by the duration of a pulse that is applied to the control wire.

6.2. DESCRIPTION OF SERVO SETUP

The system comprises three units which allow the investigation of the fundamentals of analogue and digital servo control:

- A Mechanical unit
- An Analogue Unit
- A Digital unit

The mechanical unit carries a power amplifier, dc motor and tachogenerator, absolute and incremental digital encoders input and output analogue potentiometers, a digital speed and voltage display and a sine, square and triangle waveform generator for testing purposes.
The Analogue Unit carries a four input error amplifier, a controller with independent P, I and D channels and facilities for single amplifier compensation circuits.

Fig. 6.1. Digital servo set-up

The Digital Unit carries ADC and DAC for signal conversion, switching and multiplexing circuits, encoder output and display and liner and PWM motor drive. Access is given to the input and output potentiometers enabling a wide range of linear and digital systems to be investigating.

Discovery software is provided for use with the Digital Unit. Interconnection between units is by ribbon cable and system interconnection is by plugged patch leads on the analogue or digital units, which carry clear graphic layouts.

A power supply is included which provides all of the necessary dc voltage supplies required by the system. The system is flexible. For analogue control teaching only the Mechanical and Analogue units are required. For Digital control teaching only the Mechanical and Digital units are required. These options are available separately. In this project I required only mechanical Unit and Digital unit because these two units are only used in Digital servo experiment.

6.3. MECHANICAL UNIT OF SERVO SET-UP

The mechanical unit consists of an open-board format assembly carrying the mechanics of the system plus its supporting electronics as shown in fig.6.2. The electromechanical components comprise dc motor, an analogue tachogenerator, analogue input and output potentiometers, absolute and incremental digital encoders and magnetic brake. The supporting electronics comprises: the power amplification; a low frequency sine, square and triangle waveform generator for testing purposes; encoder reading circuitry and LCD speed display and DVM. The power supply for the fundamentals trainer to this unit.

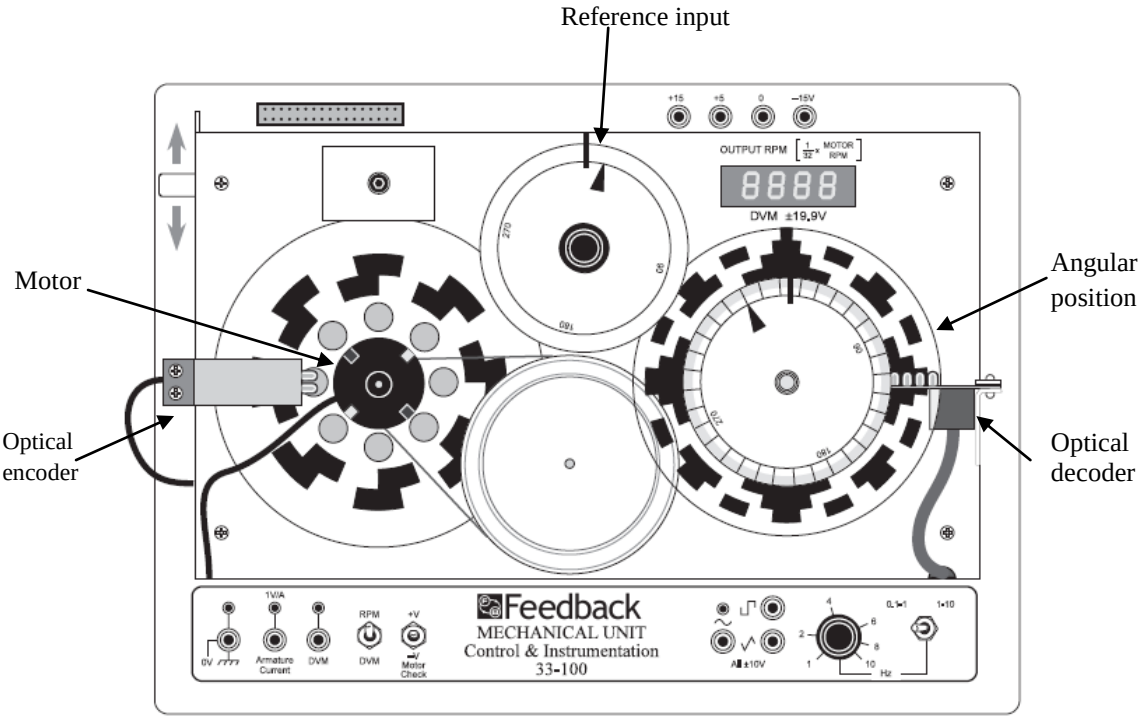


Fig.6.2. Servo Mechanical Unit 33-100

Specification:

- Open-board format unit carrying servo, system mechanical assembly and supporting electronics.
- Permanent magnet motor with armature current available.
- Tachogenerator, 2.5v/1000 rpm.
- Magnetic eddy current, brake for load. Input and output shaft potentiometers.
- Switchable, 3-figure, 7-segment LCD display of speed or voltage.
- Bi-phase incremental position or speed encoder on motor shaft.
- 6-bit absolute (Gray code) encoder on output shaft.
- Power amplifier, continuous or PWM input.
- Sine, square and triangle waveform generator, 0.1 to 10Hz.internal faults switched from other units.
- Power requirements: external ± 15V at 1.5A &_5V at 0.5A. Feedback 01-100 is recommended.
- Dimensions 150mm(H)× 295mm(W)× 220mm(D)

6.4. DIGITAL UNIT OF SERVO SET-UP

The Digital unit acts as the interface between the Mechanical Unit and a PC, or compatible microcomputer as shown if fig. 6.3.

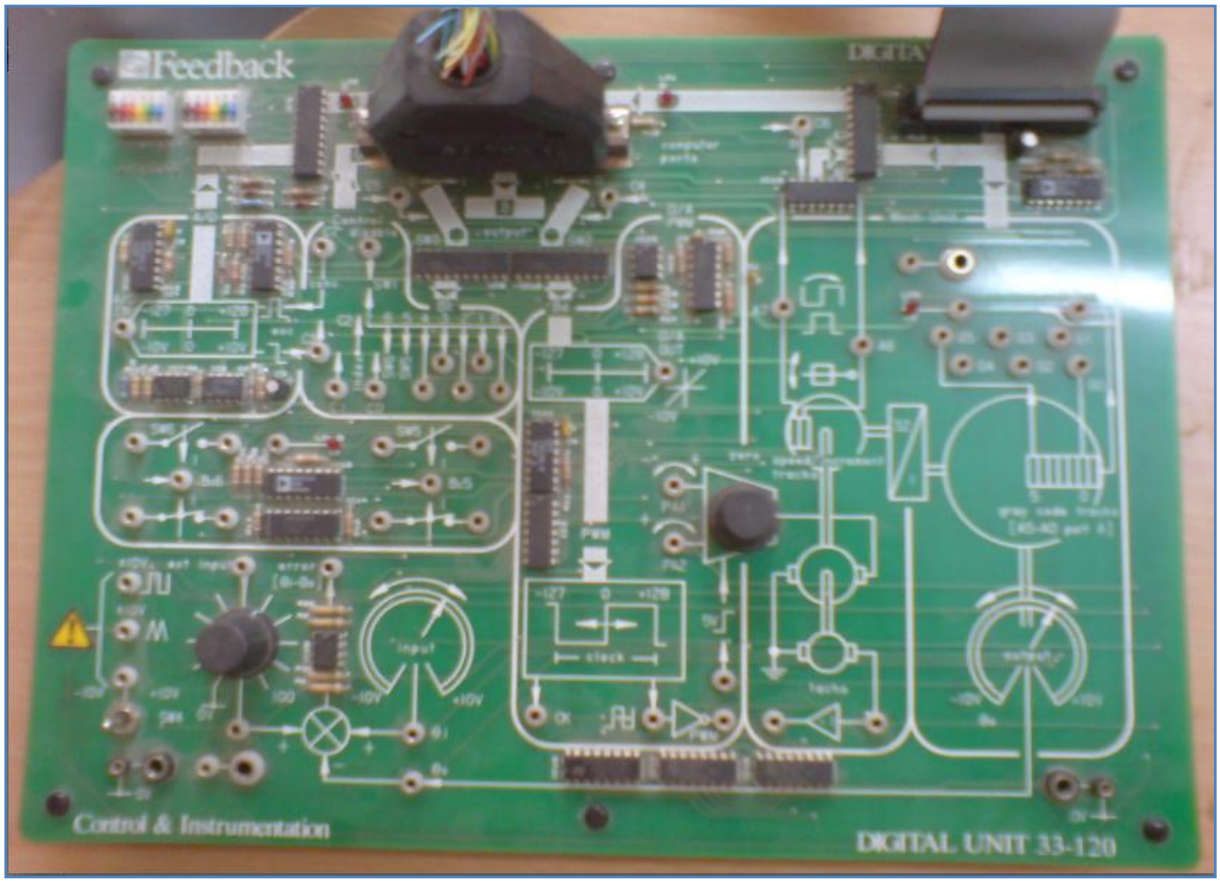


Fig.6.3. Digital Unit 33-120

The Digital Unit contains ADC and DAC circuits for signal conversion which, together with linear or digital PWM motor drive, input and output potentiometers or digital encoders enable a wide range of liner and digital systems to be realized. Computer controlled switches are provided for signal multiplexing and circuit configuration of the patching. Connection to the mechanical Unit is by way of a ribbon cable which also supplies power to the unit and connection to the computer is via a USB cable to the PC USB port.

Specification:

- Provide access to host PC, or compatible microcomputer.
- 8-way input and output busses.
- Additional control lines available.

- ADC and DAC for liner system operation.
- Liner or PWM motor available.
- LED display of incremental and absolute encoders.
- Manual patching with computer controlled configuration and multiplexing.
- Switched faults distributed through system.

6.5. FEATURES

- Open and closed loop speed and position control.
- Both analogue and digital control techniques
- Discovery software for computer assisted practical assignments.
- Inbuilt PC based instrumentation.
- On board sine, square and triangle wave form generator.
- Independent single, two term, or full PID control.
- Absolute position and incremental speed and position encoders.
- Continuous analogue position and velocity feedback signals.
- Linear or PWM motor drive.
- Switched faults throughout the system.
- LCD speed and digital voltmeters.
- Connection to a PC via USB.
- Includes power supply.

6.6. PID CONTROL OF DC SERVO MOTOR

Experimental set-up for PID controller is shown in fig.6.4.The PID controller is designed for our desired DC servo motor transfer function derived from system identification tool box more detail description given at chapter-4.The response of the PID controller for our plant is found to be stable.

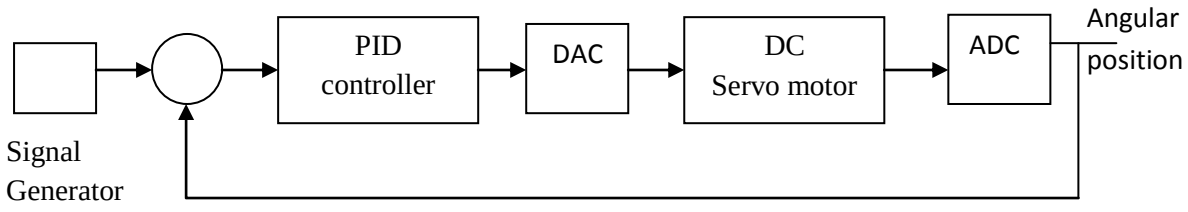


Fig.6.4. Experimental set-up for PID controller

Figure 6.5 represents the simulink model of a general PID controller. Input to the model is a square wave of amplitude 50v and frequency of 0.1 Hz with sampling time of 0.001sec.

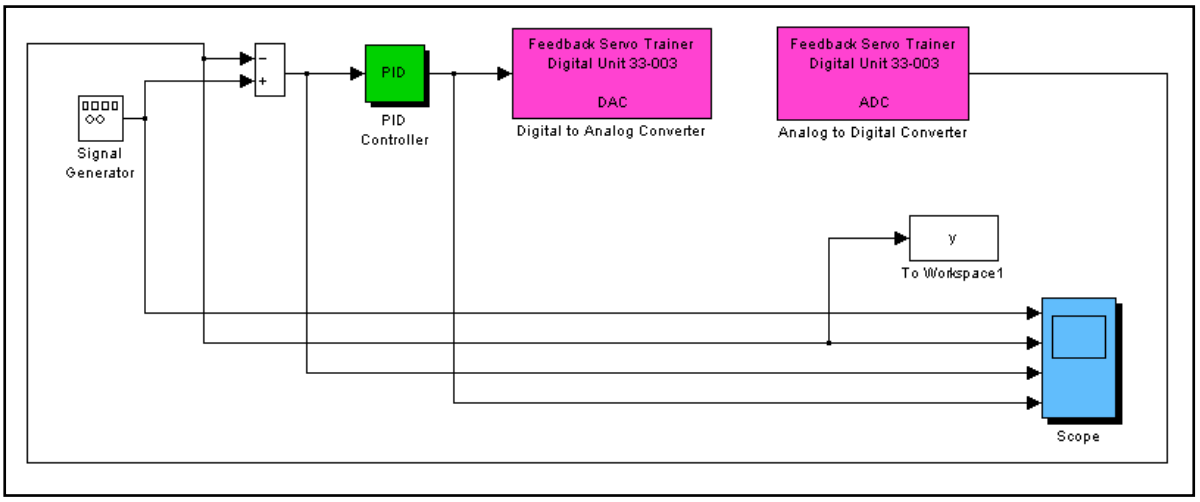


Fig.6.5.Simulink model of a general PID controller

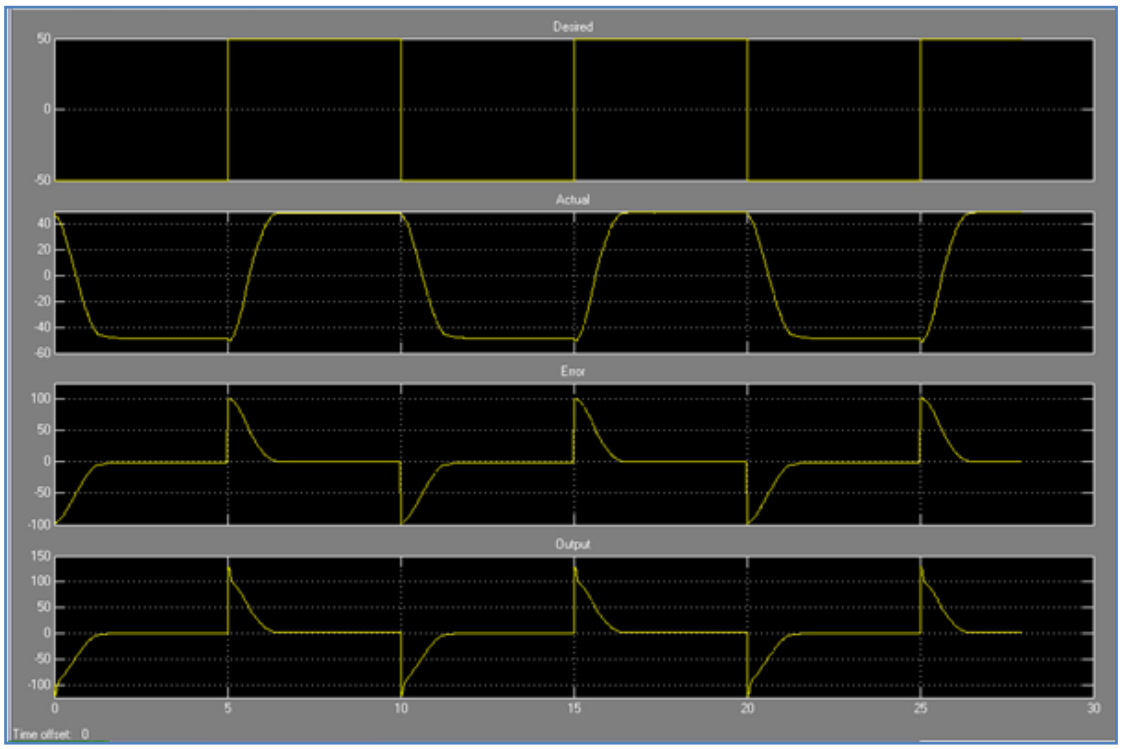


Fig 6.6. Response of a general PID controller

Fig.6.6. shows the response of a general PID controller. First one is the input to the model, second one is the actual output signal, third one is the error output, which is the input to the

PID controller and the final one is the output of the PID controller which is the input to the Digital-to-Analog converter (DAC) as shown in fig.6.5.

6.7. PID CONTROL OF DC SERVO MOTOR WITH ARTIFICIAL DELAY

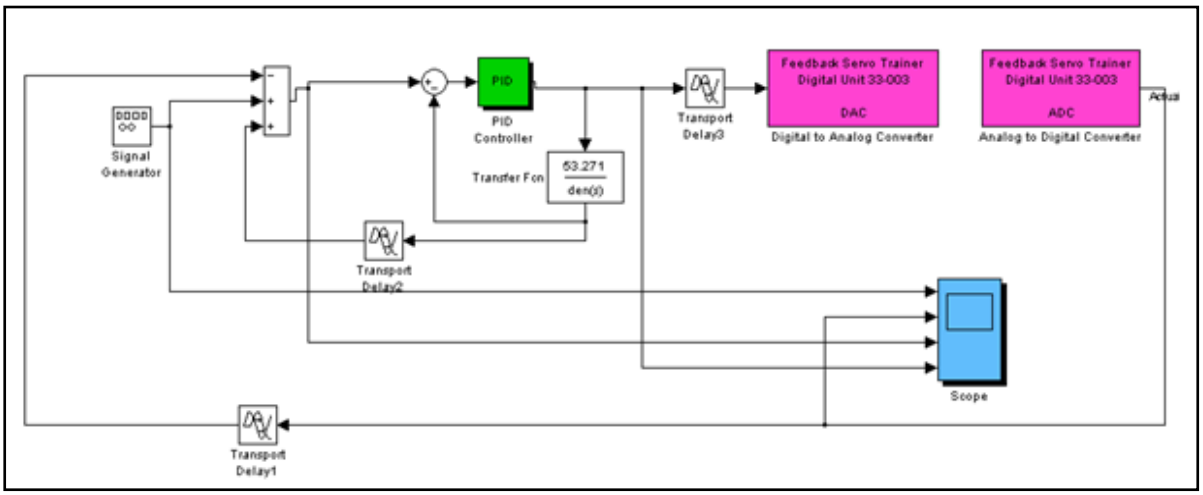


Fig.6.7. simulink model of PID controller with network

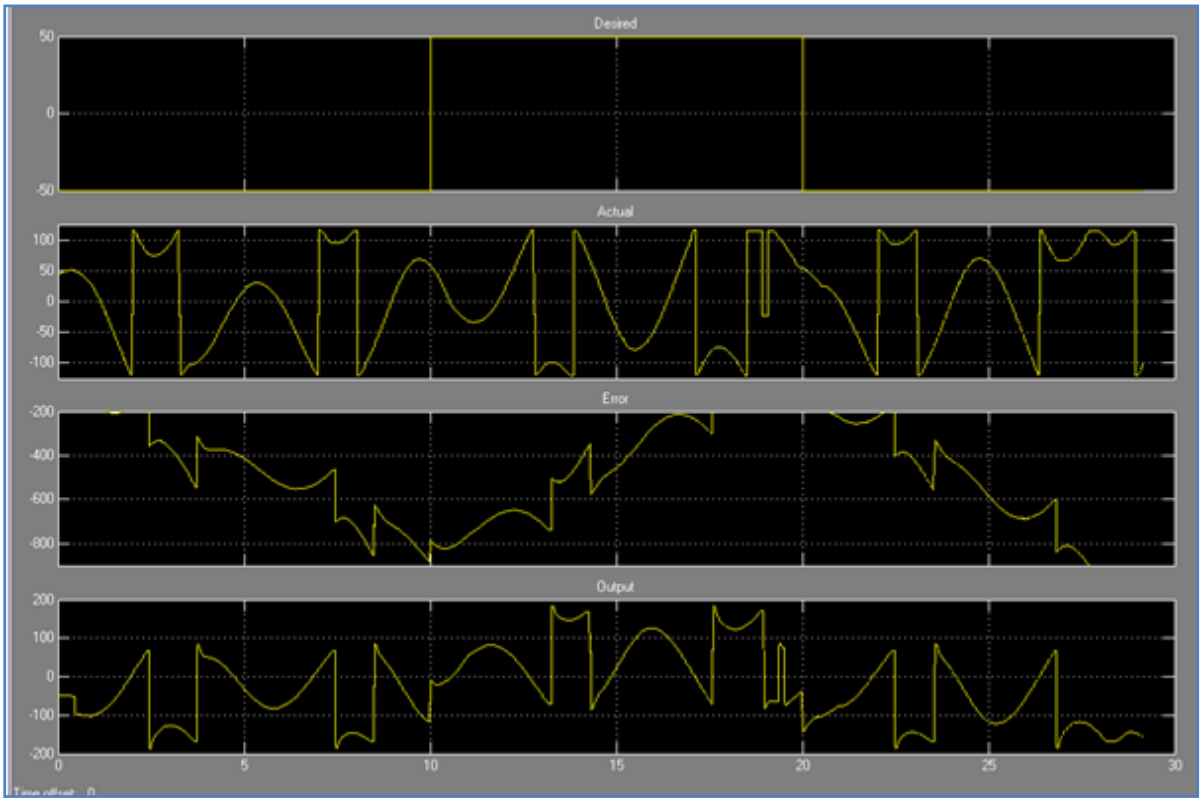


Fig 6.8. Response of simulink model of PID controller with network

The figure 6.8. Shows an unstable response of PID controller with the presence of delay in the feedback loop. First one is the input to the model, second one is the actual output signal which shows the unstable response, third one is the error output, which is the input to the PID controller and the final one is the output of the PID controller which is the input to the Digital-to-Analog converter (DAC) as shown in fig.6.8.this instability problem can be overcome by smith predictor. The smith predictor gives the dual performance i.e, .stability of a system and delay compensation in the feedback loop.

Experimental set-up for DC servo motor with smith predictor is shown in fig.6.9.this eliminates the instability found due to PID controller working along network.

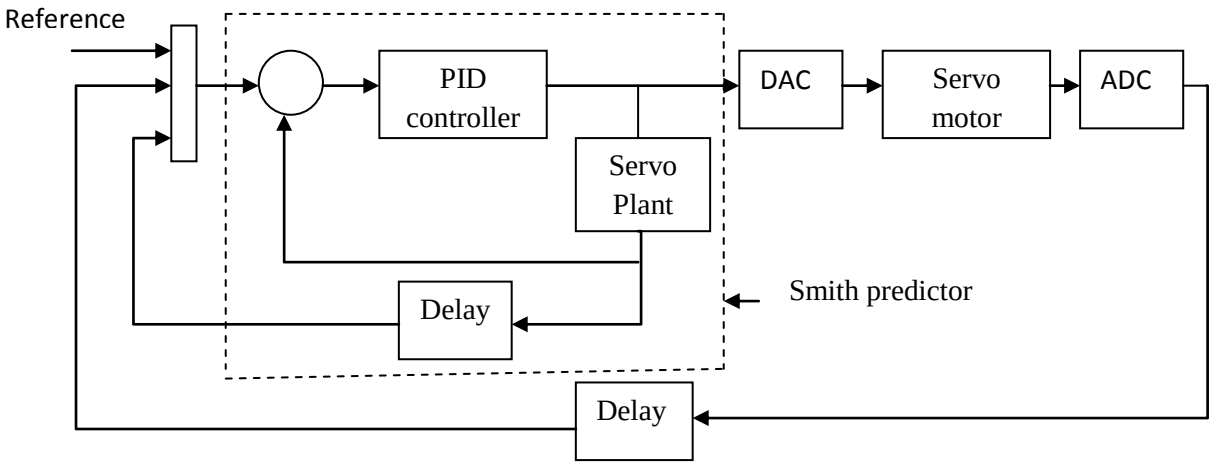


Fig.6.9. Experimental set-up for DC servo motor with smith predictor

Fig 6.10.shows the simulink model of a servo system with smith predictor. If this model is accurate, and the plant performance is reliable, this loop can provide near optimal control of

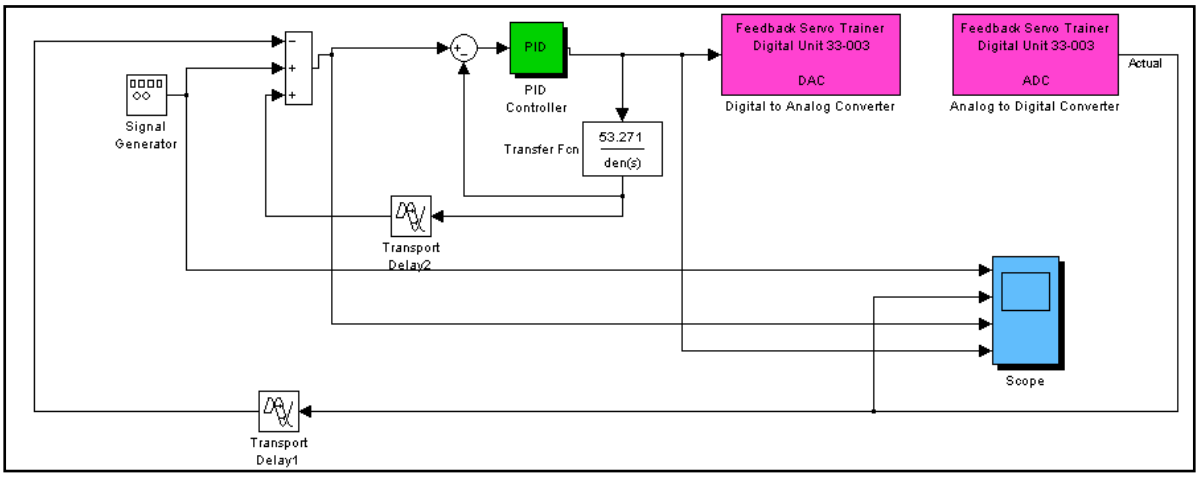


Fig 6.10.simulink model of servo system with smith predictor

the plant. The second model is used to compare the actual performance of the plant with the expected performance. Because the second model includes an accurate representation of all plant transport delays, it will delay the output from the controller to match the delayed feedback from the periphery, and these two temporally matched signals normally cancel out.

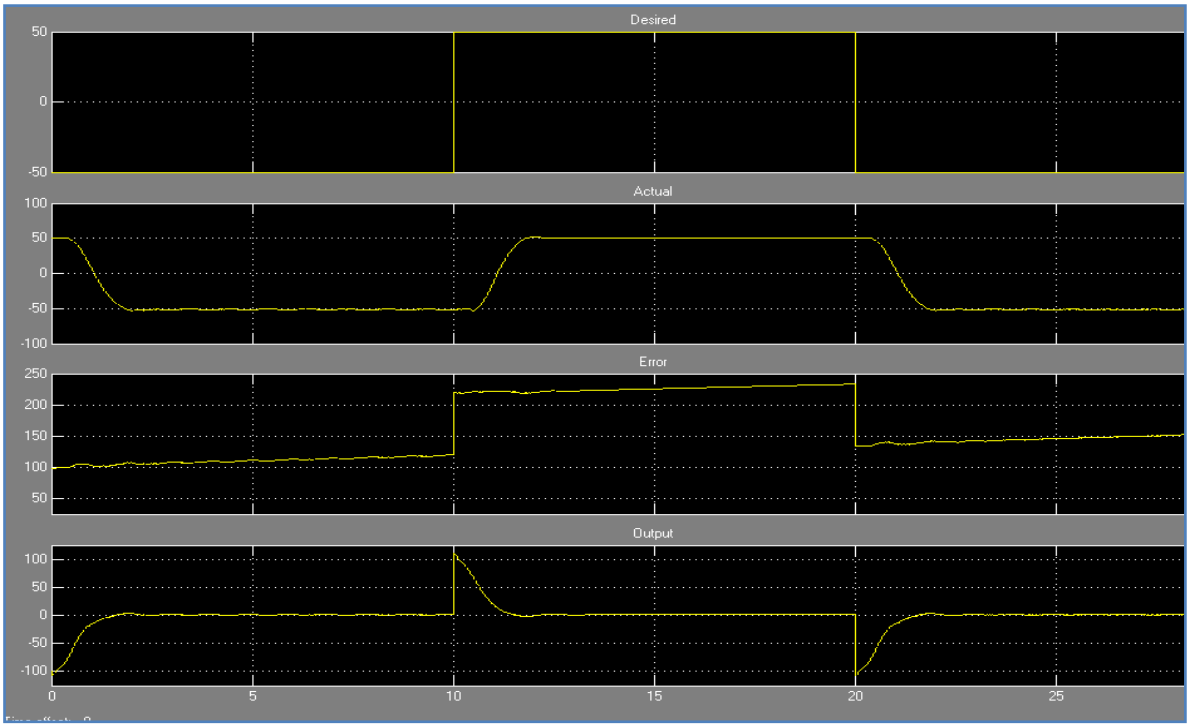


Fig.6.11. Response of simulink model of servo system without smith predictor

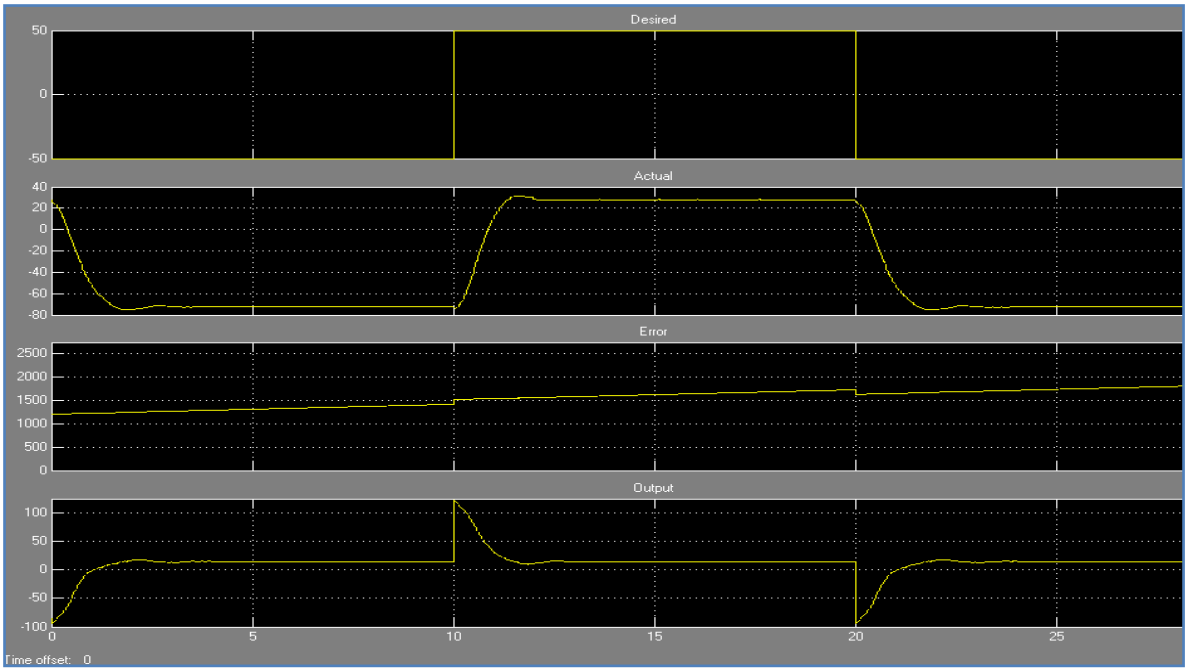


Fig 6.12. Response of simulink model of servo system with smith predictor

Fig 6.12 shows the response of simulink model of servo system with smith predictor First one is the input to the model, second one is the actual output signal which shows the stable response, third one is the error output, which is the input to the PID controller and the final one is the output of the PID controller which is the input to the Digital-to-Analog converter (DAC) as shown in fig.6.12. On comparing the figures 6.11 and 6.12 we can say that smith predictor gives stable response with compensated delay over the general PID controller in the presence of delay.

6.8. CHAPTER SUMMARY

This chapter deals with the details of the experiment on the digital servo motor set-up with artificial delay block. A brief description of servo set-up is included here. The main objective of this chapter is to compensate the delay in the servo feedback loop by using smith predictor compensation scheme and the stability of servo system is observed.

Chapter -7

CONCLUSIONS AND FUTURE WORK

7.1. CONCLUSIONS

This thesis presents study on Networked Control System, i.e., when control loops are closed over a communication network. As observed, communication network introduces time delays in the control loop. These delays may have effect on system stability and performance.

The objective of the present work is to study delay compensation schemes in the feedback loop. Smith predictor is a well known compensation technique. Using this system stability and performance may be improved compared to without compensation of delays.

The above said servo system is compatible with only MATLAB (version 6.5).and it is incompatible with higher versions of MATLAB. Due to this incompatibility the advance features available in higher version of MATLAB for our experiment cannot be implemented. This is the major disadvantage of the FEEDBACK SERVO.

7.2. FUTURE WORK

The Digital servo system in which we worked is only compatible with MATLAB (version 6.5).This is the major disadvantage of this system we found. Due to this incompatibility problem we are not able to perform some advance experiments, that are available in higher order version of MATLAB.The adaptor which creates communication between our target PC and servo system is Advantech PCI1751.It can be controlled by using real time windows target of MATLAB.The real time windows targets it creates Real-Time communication between PC and connected hardware. The main idea is to control PCI1751 adaptor.

In delay compensation techniques the smith predictor can be replaced by a control predictive generator which will be automatically compensate the delay. This will be more precisely Compensate delay then the smith predictor.

We are using the Local Area Network (LAN) for our project which not reliable for control purpose due to delay and packet losses. This can be eliminated by using a dedicated control network Control Area Network (CAN).

REFERENCES

1. G. P. Liu and D. Rees and S. C. Chai "Design and Practical Implementation of Networked Predictive Control Systems" 0-7803-8812-7/05/$20.00 02005 IEEE.

2. W. Zhang, M. S. Branicky, and S. M. Phillips, "Stability of Networked Control Systems," in IEEE Control Systems Magazine, vol. 21, pp. 84–99, February 2001.

3. Dimitrios Hristu-Varsakelis and William S. Levine, editors. Handbook of Networked and Embedded Control Systems. BirkhÄauser, 2005.

4. Jasmin Velagic "Design of Smith-like Predictive Controller with Communication Delay Adaptation" proceedings of world academy of science, engineering and technology volume 30 July 2008 ISSN 1307-6884.

5. Won-jong Kim,Kun Ji and Ajith Ambike "Real-Time Operating Environment for Networked Control Systems". IEEE transactions on automation science and engineering,vol.3,NO.3,JULY 2006.

6. Ramprasad Potluri, Kushagra Nagaich, Ramandeep Singh "Networked control systems and a mixed open-loop/closed-loop control". Proceedings of the international conference on Advances in Control and Optimization.

7 G.P. Liu[ab]*, S.C. Chai[a], J.X. Mu[c] and D. Rees[a] "Networked predictive control of systems with random delay in signal transmission channels". International Journal of Systems Science Vol. 39, No. 11, November 2008, 1055–1064.

8 Lasse Eriksson,Heikki N.Kovio "Tuning of Discrete- Time PID controllers in Sensor Network based control Systems.

9 Senchun Chai, Guo-Ping Liu, David Rees, and Yuanqing Xia "Design and Practical Implementation of Internet-Based Predictive Control of a Servo System" IEEE transactions on control systems technology, vol. 16, no. 1, January 2008.

10 Pauline Sourdille\ and Aidan O'Dwyer "An Outline And Further Development Of Smith Predictor Based Methods for Compensation Of Processes With Time Delay" ISSC 2003, Limerick. July 1-2.

11 Johan Nilsson "Real-Time Control Systems with Delays" Ph.D. thesis, Department of Automatic Control, Lund Institute of Technology.

12 Feng-Li Lian, J.R. Moyne and D.M. Tilbury. Performance evaluation of control networks: Ethernet, ControlNet, and DeviceNet. IEEE Control Systems Magazine, 21:66{83, February 2001.

13 J. Nilsson. Real-Time Control Systems with Delays. PhD thesis, Department of Automatic Control, Lund Institute of Technology, 1998.

14 G.Walsh, H. Ye, and L. Bushnell. Stability analysis of networked control systems. In Proc. American Control Conference, pages 2876{2880, June 1999.

15 B. Lincoln, Dynamic programming and Time-Varying Delay Systems, Ph.D. dissertation, Lund Institute of Technology, 2003.

16 B. Lincoln, B. Bernhardsson, "Optimal control over networks with long random delays", in Proc. International Symposium on Mathematical Theory of Networks and Systems, Jan.2000.

17 X. Nian, "Stability of Linear Systems with Time-Varying Delays: An Lyapunov Functional Approach", in Proc. American Control Conference, Denver, USA, Jun. 4-6, 2003.

18 J. Nilsson, Real-time control systems with delays, Ph.D. dissertation, Lund Institute of Technology, 1998.

19 Y.-J. Pan, H. J. Marquez, T. Chen, "Stabilization of remote control systems with unknown time varying delays by LMI techniques", International Journal of Control, Vol. 79, No. 7, pp. 752-763, Jul. 2006.

20 J. Pulkkinen, H. N. Koivo, K. Mäkelä, “Tuning of a robust PID controller – application to heating process in extruder”, in Proc. Second IEEE Conference on Control Applications, Vancouver, Canada, Sep. 13-16, 1993.

21 B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, S. Sastry, “An LQG Optimal Linear Controller for Control Systems with Packet Losses”, in Proc. 44th IEEE Conference on Decision and Control, and the European Control Conference 2005, Seville, Spain, Dec. 2005.

22 Y. Tipsuwan, M.-Y. Chow, “On the Gain Scheduling for Networked PI Controller Over IP Network”, IEEE/ASME Transactions on Mechatronics, Vol. 9, No. 3, Sep. 2004.

23 A. Willig, K. Matheus, A. Wolisz, “Wireless Technology in Industrial Networks”, Proc. IEEE, Vol. 93, No. 6, Jun. 2005.

24 W. Zhang, Stability Analysis of Networked Control Systems, Ph.D. thesis, Department of Electrical Engineering and Computer Science, Case Western Reserve University, Aug.2001.

25 K.-E. Årzén, A Simple Event-Based PID Controller, in Preprints 14th World Congress of IFAC, Beijing, China, 1999.

26 S. Branicky, S. M. Phillips, W. Zhang, “Stability of Networked Control Systems: Explicit Analysis of Delay”, In Proc. 2000 American Control Conference, Chicago . USA, pp 2352-2357, Jun. 2000

27. Y. Cao, Y.-X. Sun, C. Cheng, “Delay-Dependent Robust Stabilization of Uncertain Systems with Multiple State Delays”, IEEE Transactions on Automatic Control, Vol. 43, No. 11, pp. 1608-1612, Nov. 1998.

28. Fridman, U. Shaked, “Delay-dependent stability and H∞ control: constant and time varying delays”, International Journal of Control, Vol. 76, No. 1, pp. 48-60, 2003.

29. K. Gu, V. L. Kharitonov, J. Chen, Stability of Time-Delay Systems, Birkhäuser, 2003.

30. X. Jiang, Q.-L. Han, X. Yu, "Stability Criteria for Linear Discrete-Time Systems with Interval Like Time-Varying Delay", in Proc. 2005 American Control Conference,Portland, USA, Jun. 8-10, 2005

31. C.-Y. Kao, B. Lincoln, "Simple stability criteria for systems with time-varying delays", Automatica, Vol. 40, pp. 1429-1434, 2004.

32. X. Li, C. E. de Souza, "Criteria for Robust Stability and Stabilization of Uncertain Linear Systems with State Delay", Automatica, Vol. 33, No. 9, pp. 1657-1662, 1997.

33. X. Nian, "Stability of Linear Systems with Time-Varying Delays: An Lyapunov Functional Approach", in Proc. American Control Conference, Denver, USA, Jun. 4-6, 2003.

34. Y.-J. Pan, H. J. Marquez, T. Chen, "Stabilization of remote control systems with unknown time varying delays by LMI techniques", International Journal of Control, Vol. 79, No. 7, pp. 752-763, Jul. 2006.

35. M. Wu, Y. He, J.-H. She, G.-P. Liu, "Delay-dependent criteria for robust stability of time varying delay systems", Automatica, Vol. 40, pp. 1435-1439, 2004.

36. G.C. Walsh, H. Ye, L.G. Bushnell, "Stability Analysis of Networked Control Systems," IEEE Transactions on Control Systems technology, Vol. 10, No. 3, May 2002.

37. R. C. Luo and T. M. Chen, "Development of a multibehaviour-based mobile robot for remote supervisory control through the Internet", IEEE Transaction on Mechatronics, vol. 5, 2000, pp.376-385.

38. Yang, S. H., X. Chen, D. W. Edwards and J. L. Alty, "Design issues and implementation of Internet based process control", Control Engineering Practices, vol. 11, no.6, pp. 709-720, 2003.

39. Zhivoglyadov, P. V. and R. H. Middleton, “Networked control design for linear systems”, Automatica, vol. 39, no. 4, pp.743-750, 2003.

40. G. P. Liu, D. Rees, S. C. Chai and X. Y. Nie, “Design, simulation and implementation of networked predictive control systems”, Measurement and Control, vol. 38, pp.17-21, 2005.

41. G. P. Liu, J. X. Mu and D Rees, “Networked predictive control of system with random communication delay”, UKACC2004 International Conference on Control, Bath, UK, 2004.

42. Y. Tipsuwan, M.-Y. Chow, “Control methodologies in networked control systems”, Control Engineering Practice, Vol. 11, pp. 1099-1111, 2003.

43. K. Fall and K. Varadhan, The ns Manual (formerly ns Notes and Documentation, December 2003. http://www.isi.edu/nsnam/ns/doc/ns doc.pdf

44. M. S. Branicky, V. Liberatore, and S. M. Phillips, “Networked Control System Co-Simulation for Co-Design,” in Proc. American Control Conference, Denver, USA, vol. 4, pp. 3341–3346, June 2003.

45. V. Liberatore, “Network Control Systems.” World Wide Web, December 2002. ManualforAgent/Plantextensiontons-2, http://vorlon.case.edu/~vxl11/NetBots/ncs.pdf

46. F. Halsall, Data communications, computer networks and open systems, 3rd ed., Addison-Wesley Publishing Company, 1992.

47. G.C. Walsh, H. Ye, L.G. Bushnell, “Stability Analysis of Networked Control Systems,” IEEE Transactions on Control Systems technology, Vol. 10, No. 3,May 2002.

48. Anton Cervin, Martin Ohlin, Dan Henriksson “Simulation of Networked Control Systems using True time”.

49. Yuanqing Xia, J. Chen, G. P. Liu and D. Rees "Stability Analysis of Networked Predictive Control Systems with Random Network Delay. Proceedings of the 2007 IEEE International Conference on Networking, Sensing and Control, London, UK, 15-17 April 2007.

50. Vatanskib N., Georges J.-P.a, Aubrun Ca., Rondeaua E. and S.-L. Jämsä-Jounelab "Control Compensation Based on Upper bound delay in Networked Control Systems".

51. Katsuhiko Ogata "Discrete-Time control systems", second edition, Prentice hall international edition ISBN 0-13-328642-8.

52. Behrouz A.Forouzan "Data communication and Networking", Second edition, Tata McGraw-Hill edition,ISBN 0-07-043563-4.