Experimental Study of PI Gain Tuning in a Networked Control System

Nabeel Ahmad, Adeel Jamal, and Ali Ahmed
Pakistan Navy Engineering College
National University of Sciences and Technology
Karachi, Pakistan
Email: nabeel.ahmad1@outlook.com

Abstract*—This paper discusses the application of computer networks in closed-loop control systems. The use of Networked Control Systems (NCS) becomes feasible in a system with multiple plants distributed over a large area that have to be controlled with a centralized controller. The effect of delays that are inherent to networks is examined. These delays deteriorate the transient response and reduce the stability margin of the system. The implementation of a PI controller in an Ethernet-based IP network is the focus of this study. The chosen control technique is to tune the gains of the PI controller as the network delay changes. First of all, mathematical modeling of the system is undertaken and MATLAB-based simulations are run for this model. Modelling parameters for a PMDC motor used as the plant are estimated using Simulink Design Optimization tool. Using these simulations, tuned PI gains are calculated for different settings of the delay. Then, a hardware/software setup is developed to run the system in real-time and to visualize and analyze its response. The controller in this setup supports multiple plants and is highly reconfigurable at runtime. Finally, experimentally measured responses with simulated network delays are depicted and described. The results show that the gain-tuning of a PI controller successfully deals with the effects of network delays on system performance. A trend was observed that the tuned gains for the PI controller are greater for shorter delays and smaller for longer delays. The topics of packet ordering and loss are related to variation in the transmission delay and this relation can be a subject of further research.

I. Introduction

A closed-loop control system in which the communication between the controller and plant takes place via a network is called a Networked Control System (NCS) [1]. This idea is depicted in Fig. 1.

The use of NCS becomes relevant in situations where the system is distributed over a large area, and where various plants have to be controlled with a single centralized controller.

Computer networks have become ubiquitous these days. Most of the research work on NCS focuses on finding ways to make use of existing networks for this purpose. NCS have been implemented over industry-grade networks like Profibus, a Cyclic Service Network. However, among the already installed network types, the major share is taken by Random Access Networks, primarily Ethernet.

Naturally, the presence of a network creates complexities in the control system. The main problem is that of network delays which deteriorate the transient response and reduce the stability margin of the system. This problem is comparatively

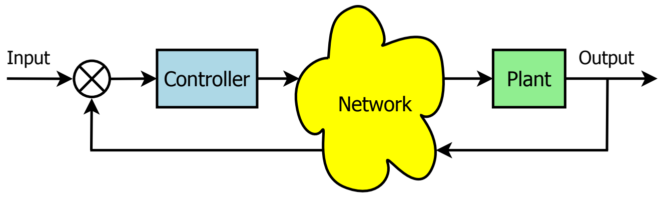


Fig. 1. A closed-loop networked control system.

easier to handle in Cyclic Service Networks (CSN) as these networks have especially designed mechanisms such as packet scheduling, priorities, etc. These features are a part of CSN by design and are helpful for implementing networked systems. However, the need of the time is to find ways for using the already installed networks for NCS, instead of designing new types of networks. The most important of these networks is the Internet.

One aspect of research on NCS focuses on finding ways for minimizing the effects of network delays. Different algorithms and control techniques have been suggested by various researchers [2]. Almost all methods work by making changes in the controller according to the state of the network. One very simple yet efficient technique is to tune the gains of a PI controller [3]. This is the method that we have chosen for our study.

II. Problem Formulation

Briefly stated, our goal was to create a physical Networked Control System. This system would not only be beneficial in demonstrating the workings of an NCS, but would also be helpful in conducting further research by providing a platform for real-world testing of new algorithms.

From a functional perspective, the system can be divided into three parts: Controller, Network, and Plants.

Controller: The system consists of a single centralized controller. It should be capable of controlling multiple plants distributed over a network. It should also be easy to reconfigure, hence facilitating changes to be made in the control algorithm.

Network: This is the communication medium between the Controller and the Plants. The chosen network is TCP/IP.

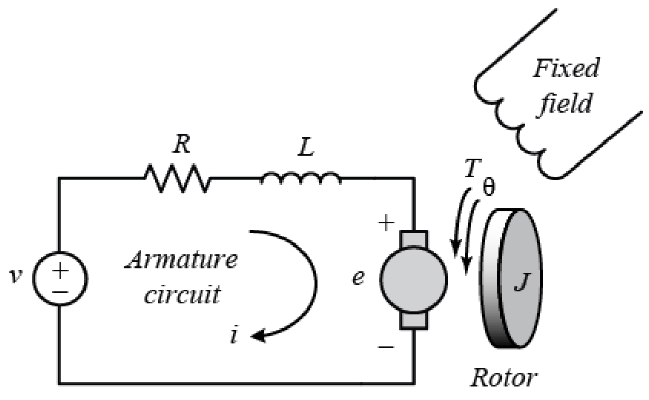


Fig. 2. Electro-mechanical construction of a PMDC motor [4].

TABLE I
ESTIMATED MOTOR PARAMETERS

Symbol	Description	Value
J	Inertia	$0.0035\ kg/m^2$
K	Back-EMF constant	$0.0114\ V/rpm$
R	Winding Resistance	$1.0963\ \Omega$
L	Winding Inductance	$0.1526\ H$

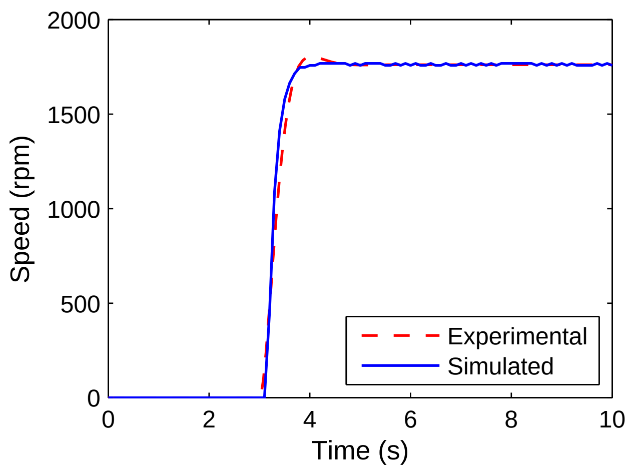


Fig. 3. Simulated vs. experimentally measured step response of the motor.

Both the Controller and the Plants connect to Ethernet based networks.

Plants: The system is capable of handling multiple plants, and this functionality has been tested. However, our analyses were limited to a single plant. This plant consists of a PMDC motor, where the controlled parameter is the shaft's rotational speed.

III. CONTROL DESIGN AND ANALYSIS

In this section, we will describe how the system was modeled. We will also show the results of the simulations run using this model.

TABLE II
TUNED PI GAINS FOR DIFFERENT CONDITIONS OF DELAY

Delay (ms)	K_p	K_i	Ref. Symbol
0	0.4276	2.1799	g_0
100	0.3659	1.6541	g_1
200	0.3839	1.4039	g_2
300	0.3241	1.1608	g_3
400	0.3529	1.0272	g_4
500	0.3321	0.8967	g_5

TABLE III
RESULTS OBTAINED FROM SIMULATIONS DEPICT DETERIORATION OF SYSTEM PERFORMANCE WITH INCREASING DELAYS

Delay (ms)	0	100	200	300	400	500
Overshoot (%)	2.47	23.8	42.3	63.3	84.8	—
Peak Time (s)	0.495	1.1	1.1	1.2	1.3	—
Settling Time (s)	1.19	3.24	5.6	10	29.5	—

A. Motor Modeling

Fig. 2 describes the construction of a permanent magnet DC motor. We will now discuss the mathematical model used for the motor.

In Fig. 2, θ is the angular speed of the motor shaft. It is the output and controlled variable for our system. V is the voltage applied to the motor winding. It is the input and controlling variable. The input-output relation in the Laplace domain (i.e. the transfer function) is as follows:

$$\frac{\theta(s)}{V(s)} = \frac{K}{Js(Ls+R)+K^2} \tag{1}$$

Table I describes the symbols that have been used above. The values of these motor parameters were estimated for our motor using the Simulink Design Optimization tool.

B. PI Controller

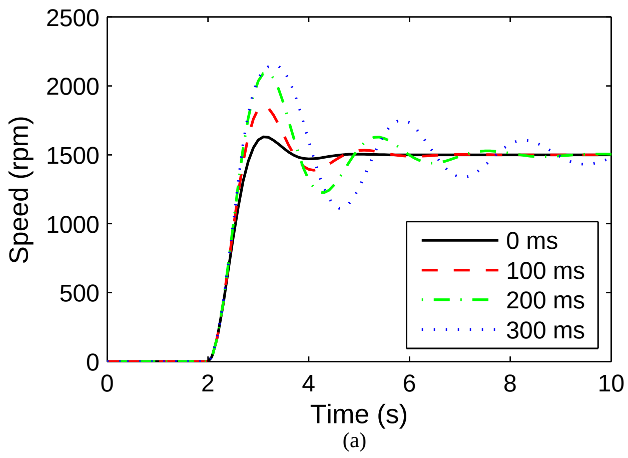
A PI controller is a generic controller. The transient and steady state response of a system can be altered by varying the proportional (K_p) and integral (K_i) gains of the controller.

The z-domain transfer function of a discrete-time PI controller is as follows:

$$\frac{U(z)}{E(z)} = K_p + K_i.T_s\frac{1}{z-1} \tag{2}$$

where, $U(z)$ is the output of the controller, $E(z)$ is the input to the controller, and T_s is the sampling time.

In simulations, the targeted transient-state response parameters of the system were: overshoot of about 2.5% and settling time of about 1.2 s. For different conditions of the delay, the gains of the PI controller were tuned to achieve the desired performance. This was done using MATLABs PID Tuning Tool. The results are depicted in Table II.



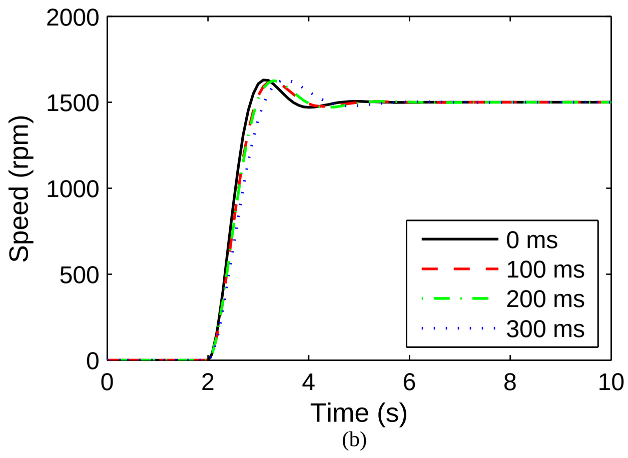


Fig. 4. Step responses for varying delays as obtained in simulations using: (a) untuned controller gains (b) tuned controller gains.

C. Simulations

Fig. 3 compares the simulated open-loop response of the system to the experimentally measured open-loop response. This gives an idea about the accuracy of the parameter estimations given in Table I.

The simulation results for the chosen control scheme are shown in Fig. 4. In Fig. 4(a) the controller gains are fixed at g_0, whereas in (b) the gains are tuned according to Table II. Both figures show step responses for four settings of the simulated delay which are displayed in the legend.

The performance parameters for the untuned controller are described quantitatively in Table III. The table shows how changes in the delay affect system performance. The dashes in the 500 ms column imply instability. Some of these results were depicted graphically in Fig. 4(a).

IV. Experimental Setup

Now we come to the main part of the paper. Here we will look at the technical details of the experimental setup that enables us to implement and analyze an NCS. In section II, we had discussed a functional description of our system. In this section, we will discuss its physical implementation. A general overview of the system is given in Fig. 5.

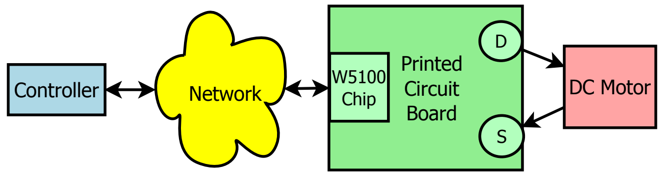


Fig. 5. Description of the physically implemented system. (D and S stand for Drive and Sensor, respectively.)

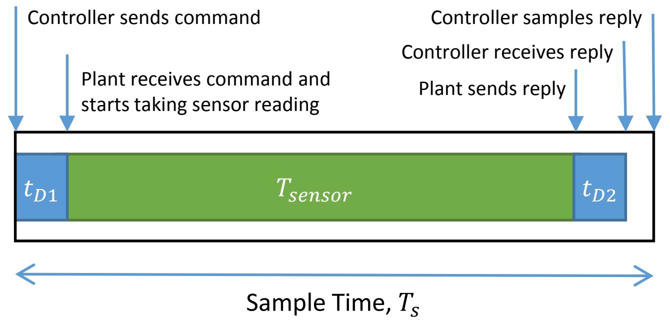


Fig. 6. Scheme of transmission and reception of messages in the system within one sample interval.

A. Network

In this section, we will look at the nature of the communication that is done in our system. A TCP/IP network is used as the communication medium between the controller and the plant. In particular, UDP datagrams are used. UDP is not only simple to implement in the embedded firmware but it is also better suited for real-time systems (including NCSs) as compared to TCP. As UDP does not provide any facilities for data correction, it leaves much flexibility for the design of the controller. Due to the lackluster features of UDP, the network is relieved of much of the overhead that is part of TCP [7].

1) Timing Relationships: Let us now see what goes on in the system within a single sampling interval T_s. Fig. 6 shows the timing relationships of different events that occur in the system periodically. t_{D1} and t_{D2} are the transmission delays that the message goes under within a single cycle. Their sum is equal to the round trip time (RTT). T_{sensor} is the time taken by the plant side to prepare the reply message. In particular, this is the time required by the sensor to take a reading. In our system this is equal to 95 ms.

2) Network Delays: Let us call the sum of all three delays from Fig. 6 ($t_{D1}+T_{sensor}+t_{D2}$) as the *actual* delay. However, the delay which affects the system performance is different from the actual delay and is called the *effective* delay. The effective delay in this system can only be an integral multiple of the sample time T_s. The sample time for the system is fixed at 100 ms.

The meaning of effective delay for our particular system may get clear from the following example: Suppose that a command packet is sent by the controller at time t. If the

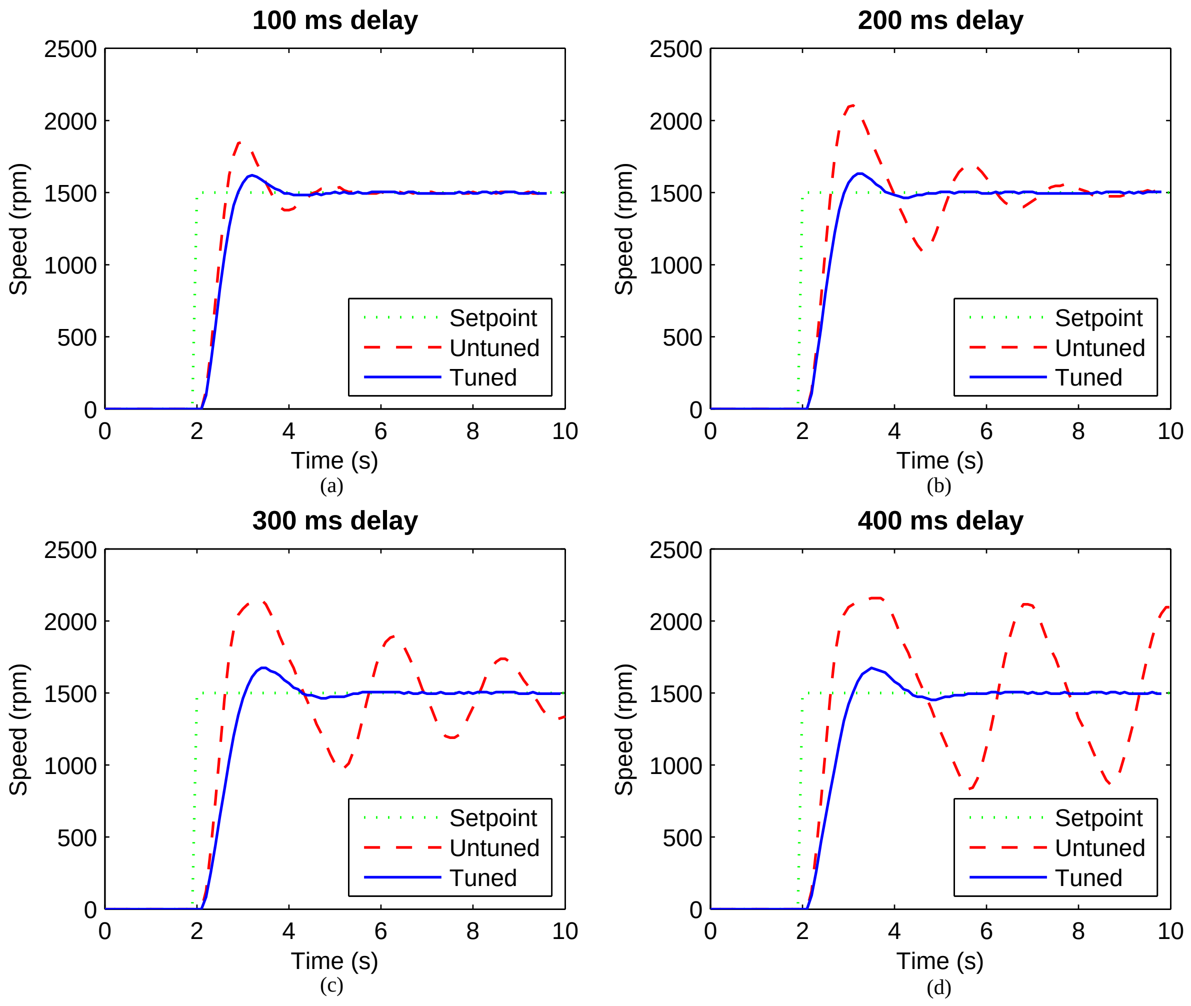


Fig. 7. Experimentally measured system step responses for simulated delays of: (a) 100 ms; (b) 200 ms; (c) 300 ms; (d) 400 ms.

reply to this particular command reaches the controller before the next sample time $(t+T_s)$, the effective delay is considered to be zero. If it reaches after $(t+T_s)$ but before $(t+2T_s)$, the effective delay will be 100 ms, and so on.

All the results presented in this paper were taken on a network that consisted of just a single router with no extra traffic. Hence the actual transmission delays $(t_{D1}+t_{D2})$ were in the order of microseconds, and the effective delay for the controller was 0 ms. Delays of the order that would have any effect on the system were not possible on the networks that were available to us. Therefore an artificial delay has been simulated in the MATLAB-based controller. This delay produces the same effect as t_{D2} does.

B. Controller

The controller runs on a PC and uses the MATLAB software. The Real-Time Windows Target toolbox is essential to the function of the controller. This approach provides a very suitable interface for both the observation and the control of the system. The high-level interface allows for an easily reconfigurable controller at runtime. For example, the PI gains of the controller can be set to any desired value, the integrator can be reset, the controller can be bypassed, closed-loop and open-loop configurations can be switched, the shape of setpoint signal can be changed, the simulated network delay can be controlled, etc. All of these changes can be made while the system is running. The controller is also capable of automatically changing its gains according to a predefined scheme.

C. Plant

As has been mentioned before, the plant is a PMDC motor. Its modeling is presented in section III-A. The plant is accompanied by a printed circuit board (PCB). This PCB consists of three major parts which are described next.

D. Sensor

The sensor measures the rotational speed of the motor's shaft. We will look at the working and limitations of the sensor

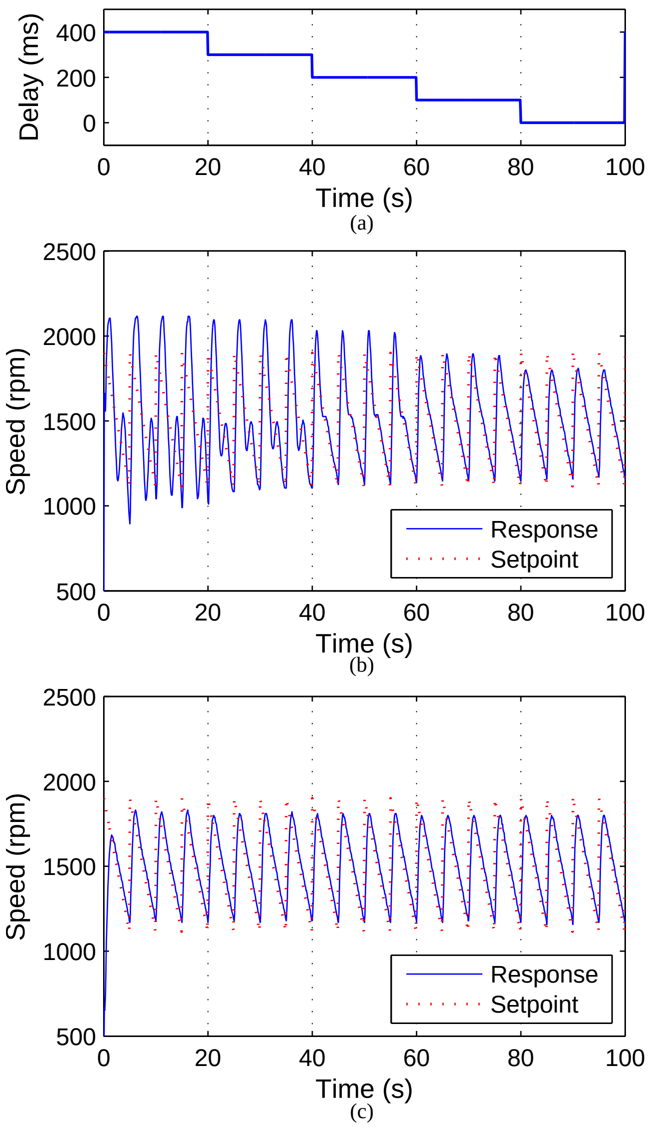


Fig. 8. Experimentally observed system behavior with and without automatically tuned controller gains: (a) simulated delay varying with time; (b) untuned system response with a sawtooth waveform as the setpoint; (c) tuned system response.

in this section.

The sensor is a rotary encoder which consists of a photocoupler and a disk. The disk is mounted on the motor shaft and features regularly placed slots at its edges. As the disk rotates, the photocoupler produces pulses at its output. These pulses are counted by the microcontroller.

The scheme used for measurement of speed is to count the pulses from the photocoupler for a fixed amount of time. For a rotating disk a total slot count of N_{slots}, counting time T_{count} (in seconds), and counted pulses n, the rotational speed θ will be:

$$\theta = \frac{n \ N_{slots}}{60 \ T_{count}} rpm \tag{3}$$

In our case, N_{slots} is equal to 60, and T_{count} is set at 95 ms.

TABLE IV
A QUANTITATIVE COMPARISON OF THE EXPERIMENTALLY MEASURED TUNED AND UNTUNED SYSTEM STEP RESPONSES.

	Delay (ms)	**0**	**100**	**200**	**300**	**400**
Untuned	**Overshoot (%)**	5	23	40	42.5	47
	Peak Time (s)	1.1	0.9	1	1.2	1.5
	Settling Time (s)	1.5	3	7	19	–
Tuned	**Overshoot (%)**	5	4.7	5.3	11.6	11.6
	Peak Time (s)	1.1	1.1	1.2	1.4	1.4
	Settling Time (s)	1.5	1.8	2.2	3.3	3.3

Let us now consider the limitations of the sensor. The above equation shows that the sensor resolution is approximately 10 rpm. As each measurement takes 95 ms to complete, the maximum sample rate is about 10 Hz. Moreover, the sensor reads the average speed over a 95 ms interval rather than reading an instantaneous value.

A measurement is started upon the reception of a command message from the controller over the Network, as described above. A measurement that is started at time t ms will be finished at $(t+95)$ ms, and it will be treated by the controller as the instantaneous speed of the motor at $(t+T_s)$ ms.

E. Drive

A MOSFET-based H-bridge is used to drive the PMDC motor. The drive is fed a PWM signal from the microcontroller. A variation in the duty cycle of this signal represents a variation in the average voltage applied to the motor.

F. Microcontroller and Network Interface

The microcontroller is interfaced with the networking chip W5100. The W5100 chip has the TCP/IP stack hardwired up to the transport layer. The microcontroller is also interfaced with the Sensor and the Drive. It sends the Sensor data to the Controller. It also receives the PWM waveform's duty cycle from the Controller, and feeds the waveform to the Drive.

V. EXPERIMENTAL RESULTS

We will now look at the results obtained using the experimental setup that was described in the last section.

The obtained graphs for step-response are shown in Fig. 7. An appreciable correlation can be seen between these experimentally measured responses and the simulated responses from Fig. 4. A considerable amount of saturation can be observed in Fig. 7(d), which was not modeled for the simulations.

A quantitative presentation of the step-response results is given in Table IV. The dash here indicates that the settling time for the 400 ms response was very long (more than a minute).

Up till now, all the system responses that we have observed have a constant delay. In Fig. 8, the simulated delay applied to the system is varied with time. The delay is decreased by one sample time T_s every 20 seconds, with the initial delay being 400 ms. The scheme of delay variation is shown in Fig. 8(a).

A sawtooth waveform with a period of 0.2 Hz is chosen as the setpoint. Fig. 8(b) depicts the response of the system without automatic gain tuning. For the untuned response, the controller gains were fixed at g_0. Fig. 8(b) shows the response of the system where the controller gains were altered when a variation in the delay occurred. Here, the gains were changed according to Table II.

It should be noted that for the tuned response shown in Fig. 8(c), the integrator of the PI controller was not reset at any time while the system was running. A separate system response reading was taken with the integrator being reset whenever the gains were changed. This response, which is not shown here, is similar to Fig. 8(c) except that "spikes" were observed whenever the integrator was reset.

VI. Conclusion

The results presented in this paper show that the gain-tuning of a PI controller successfully deals with the effects of network delays on system performance. It verifies previous studies such as [5] by reproducing and demonstrating the results in a more refined manner. We have seen the trend that the tuned gains are greater for shorter delays and smaller for longer delays. A networked system may never match the performance of a hardwired system, but this does not in any way imply that an NCS cannot be successfully implemented. The limitations of this study include: a slow sensor and hence a low sample rate, use of simulated delays rather than real network delays, and disregard of rotational friction in the motor model. The topics of packet ordering and loss are related to variation in the transmission delay. This relation can be a topic for further research.

References

[1] M.-Y. Chow and Y. Tipsuwan, "Network-based control systems: a tutorial," in *Proc. Annual Conference IEEE Industrial Electronics Society (IECON'01)*, vol. 3. IEEE, 2001, pp. 1593–1602.

[2] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *Proc. IEEE*, vol. 95, no. 1, p. 138, 2007.

[3] Y. Tipsuwan and M.-Y. Chow, "On the gain scheduling for networked PI controller over IP network," *IEEE/ASME Transactions on Mechatronics*, vol. 9, no. 3, pp. 491–498, 2004.

[4] CTMS. DC motor speed: System modeling. [Online]. Available: http://ctms.engin.umich.edu/CTMS/index.php

[5] M.-Y. Chow and Y. Tipsuwan, "Gain adaptation of networked DC motor controllers based on QoS variations," *IEEE Transactions on Industrial Electronics*, vol. 50, no. 5, pp. 936–943, 2003.

[6] T. C. Yang, "Networked control system: a brief survey," *IEE Proc. Control Theory and Applications*, vol. 153, no. 4, pp. 403–412, 2006.

[7] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*, 6th ed. USA: Pearson Education, 2013.