AN ABSTRACT OF THE DISSERTATION OF

Daniel López Echevarría for the degree of Doctor of Philosophy in Electrical and Computer Engineering presented on June 9, 2011.

Title:
Variable Sampling Compensation of Networked Control Systems With Delays Using Neural Networks.

Abstract approved: ______________________________

Mario E. Magaña

In networked control systems (NCS) information or packets usually flow from a sensor or a set of sensors to a remotely located controller. Then the controller processes the received information and sends a series of control commands to the actuators through a communication network which could be either wireless or wired. For any type of communication network, time delays are an inherent problem and depending on the conditions of the network they can be constant, variable or even of random nature. Time-delays occurring from sensor to controller and from controller to actuators may cause important system performance degradation or even instability. This work proposes a novel strategy of using the predictive capabilities of artificial neural networks (NN), particularly the application of an adaptive NN, to minimize the effects of time delays in the feedback control loop of NCS. We adopt an adaptive time delay neural network (TDNN) to predict future time-delays based on a given history of delays that are particularly present on the network where the corresponding system belongs to. The adaptive nature of a TDNN allows the prediction of unexpected variations of time-delays which might not be present in the training set of a known history of delays. This is an important characteristic for real time applications. Using predicted time delays, different methodologies can be used to alleviate effects of such delays on NCS. Our focus here is on the development of an observer-based variable sampling period model, and this dissertation describes how this method can be used as an effective solution for this problem. Generally speaking, the predicted time-delay values are used for the discretization of a continuous-time linear time invariant system model transforming it into a discrete-time linear time variant system model. In this dissertation, the practical phenomenon of packet dropout is also addressed.

Variable Sampling Compensation of Networked Control Systems With Delays Using Neural Networks

by

Daniel López Echevarría

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Presented June 9, 2011
Commencement June 2012

ACKNOWLEDGEMENTS

First of all I want to praise God for giving me the strength, understanding, and wisdom to complete my studies. I also want to express my sincere appreciation to my advisor Dr. Mario E. Magaña for his invaluable input to this project and for his support, motivation and advice throughout my research and stay at Oregon State University. I also want to thank Professor John Schmitt for his suggestions and feedback. I want to thank my parents, brothers and sister who give me strength, words of wisdom and encouragement and for their unconditional love. Also I have special appreciation and gratitude toward my friend Panupat Poocharoen for his support, advice and true friendship. I am grateful to everyone who has supported me during this period of my life.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF TABLES

DEDICATION

To my parents, brothers and sister.

Chapter 1 – Introduction

In the last decades, increasing advances in communication systems, especially in the field of wireless communications, have been providing researchers new ideas for applications. Areas like geology, forestry, biology, robotics, aeronautics, medicine, and space continue finding new ways to improve research techniques using wireless communication devices. Those devices are utilized not only for conventional voice or video transmissions, but also for sending and receiving information using a wide variety of sensors and actuators.

New communication systems are faster and more reliable, and can compete with conventional communication and control systems set-ups, namely, wired-connected (traditionally known as point-to-point connection). For instance, it is possible to locate a controller and a plant (the system to be controlled) far from each other without physical connection, but still get the same performance as if they were wire-connected. In the area of control systems, the traditional point-to-point communication architecture has been successfully implemented for decades. This architecture has reached its limits because of the physical need to connect each sensor and actuator by a wire. This is not a suitable way to meet new requirements such as modularity, decentralization of control, low cost, and/or quick and easy maintenance. The point-to-point architecture is being upgraded using new communications set-ups, and currently there is an increasing interest in applying wireless communication techniques to networked control systems (NCS). Wireless systems are more flexible and they optimize physical resources such as connection hardware, weight and space. These are important characteristics when transmitters and receivers are in motion. This is why wireless control systems have become a new trend in modern control and a growing area for research.

However, the advantages offered by a wireless network applied to a control system are tied to inevitable problems appearing in the transmission of sensor and control signals. One problem is the effect of time delays in the feedback loop of NCS. Time delays may cause instability in the controlled system. As we will see in later chapters, delays are characterized according to origin inside of the NCS, and they can have different behavior like length and probability distribution. Controlled systems subject to delays have been a constant topic of research generating different approaches to compensate for the effects of delays. Another characteristic of the communication network is the loss of information due to packet loss or packet dropout. This may cause instability and poor performance in the NCS because of the critical real-time requirement in control systems. This issue has also been investigated over the years and further chapters mention the works of the researchers on different solutions.

Different approaches for controlling a plant over a NCS have been created and each uses different tools to solve a particular problem. Particularly interesting is the application of a combination of different tools to address the time delay and packet dropout in a NCS. Tools include artificial neural networks and classical control methods like linear quadratic regulators (LQR) which make use of estimators like the Kalman filter.

Artificial neural networks, or simply called neural networks (NN), were originally conceived based on the biological counterpart, neurons, and the extensive connection between them, synapses, forming neural networks. Such networks are responsible for the creation of human brain activities like memory, learning ability, generalization activity, adaptivity, and inherent contextual information processing [1]. It can be said that NNs are computational models of the brain. The field of NN covers a very extensive area and to mention all types and applications is beyond the scope of this thesis.

Artificial neural networks research and applications are in constant growth, and have been increasingly applied in several fields including communication and control [3] [4]. We are particularly interested in the capacity of NN to approximate any nonlinear function. Such characteristics can be applied to predict a sequence of delays either off-line or on-line, which can be applied in real-time processes to predict one step ahead or multiple predictions as we will see in future sections. The focus of this dissertation is the application of an artificial neural network as a predictor of a time series. Time series prediction using NN has been investigated in the literature [6], [7], [9]. Extensive research can be found regarding the application of NN for forecasting time series [3]. In networked control systems, prediction based on NN has been investigated to guarantee stability and good system performance with the presence of time delay in the feedback communication loop [10], [11], [12]. Specific research on predicting internet time delay has been done in [4], [7], [8], [9]. This dissertation will describe some basic principles of NN and how NNs are applied to help compensate for the effects caused by induced time delays and packet dropouts in NCSs. Our interest is in the treatment of delays and packet loss in the feedback communication loop that exists in a NCS. In this dissertation a novel approach to deal with the effects of time delays in a NCS is investigated, based on the prediction of a series of time delays. Such predictions are done by a time delay artificial neural network. The predicted values are used as the sampling period for the controlled system model. Furthermore, the system model is an observer-based variable sampling period which allows compensation for the effects caused by the late arrival or absence of the information packet to the plant, namely, time delays or packet dropout.

1.1 Thesis Contributions and Organization

The main contribution of this research is the development of a variable sampling methodology to mitigate the effects of time delays and packet dropouts in networked control systems using neural networks. The aim is to apply this methodology to applications of controlling dynamic systems that are located remotely.

Specifically, we develop and apply an adaptive time-delay neural network to perform one-step-ahead predictions of the time delays occurring in the communication loop of the networked control system. Such neural network uses a sequence of previous time-delay values to predict the next value of the sequence. We present an iterative way to obtain the most suitable neural network topology for our particular studies. An important part on the selection of the best topology is the validation of every topology created. This step plays the main role in selecting the most appropriate topology. We show that the proposed neural network with a relatively simple topology with five input nodes in the tapped delay line, nine nodes in the hidden layer and one output, is capable of achieving predictions of a time-delay sequence in a one-step-ahead fashion. We also propose and implement a NN configuration for multi-step predictions. In our design multi-step predictions are required in the event of packets dropout. We show that multi-step predictions are possible with certain limitations.

A second contribution is the creation of a novel networked control system design. We present an observer-based variable sampling control model. This proposed model uses the predicted value from the NN as the sampling time of the NCS model. In our design each packet is timestamped, it has also the information of the last time delay from controller to actuator and the plant states data. With the information of each packet the NN is able to predict the next time delay value which is used by the observer and controller to compute the control signal. The main objective of this model is to compensate for the effects caused by time delays and packet dropouts in the communication loop. To address the packet dropout case, the proposed NCS model has two variations and the difference between them is in how the missing information is replaced. This dissertation shows that these designs are able to compensate for the time delays in the communication loop and also for packets dropped until certain extent.

This dissertation is organized into seven chapters. Each chapter is self-organized providing sufficient background and related work.

In Chapter 2, related work regarding the treatment of time delays on NCS is presented. The basic idea of variable sampling period is also given. The problem formulation is presented together with a description of the proposed NCS model.

In Chapter 3 the model for the time delay series used in our research based on a recursive equation to form a correlated sequence of time delays is presented.

In Chapter 4, we present the artificial neural network used to predict a time delay series. We describe the fundamentals of the NN used in our research, namely, time delay neural network (TDNN). Also presented is the methodology used for the selection of the topology of the TDNN. Such a method is based on trial and error, but it is done sequentially by allowing the NN to *grow* iteratively, i.e. change the number of input nodes in the input layer and in its hidden layer. Training of the NN was done in a way to prevent overfitting (overtraining) by implementing the early stopping technique. This method is compared with an alternative methodology, the Baum-Haussler rule. We present several numerical tests designed to select the most appropriate topology for the neural network which is applied in the rest of our research.

In Chapter 5, the methodology of using a variable sampling period to create suitable control models to compensate for packet dropouts and delays in a system is explained and derived. We also present our proposed observer-based variable sampling period models for application in networked control systems. Each model is designed to compensate for the effects caused by time delays and packet dropout in a communication network.

In Chapter 6, we apply the proposed control models to the classic inverted pendulum problem. Several cases are investigated. We simulate the controlled system when it has no delays and no packet dropouts affecting it, followed by the case when delays and no packet dropouts are present in the communication loop. Also the case when delays and packet dropouts occur in the NCS is investigated. Applied delays sequences with different means are included. This provide an insight to the limitations of the proposed techniques applied to a real case. It was found that under certain assumptions and conditions our proposed methodology is able to control the particular case of the inverted pendulum subject to delays and around 50% packets dropped. This chapter is followed by conclusions and future work in Chapter 7.

Chapter 2 – Related Work and Problem Formulation

Treatment of delays and packet loss is an important topic in the area of NCS. Different algorithms have been developed and studies have been conducted to find means to compensate for their effects. In this chapter some of those approaches are summarized.

2.1 Time Delay in Networked Control Systems

Networked control systems with delays in the communication loop have been approached in many different ways over the years. For instance, the queuing methodology shown in Fig. 2.1 has been utilized to change random network delays, into non-variable delays [34–36]. By doing this conversion the system becomes time invariant and is easier to analyze. This methodology uses an observer to estimate the plant states and a predictor to compute the predictive control based on the past output measurements. This kind of method has a drawback which is that the queues are designed according to the maximum time delay that can occur in the system. If the time that the queues use is shorter than the maximum delay, then the queue will be empty before the new information packet arrives. For this reason, the queue method adds more delay to the system.

Stochastic approaches like the one in [5], which proposes an optimal stochastic

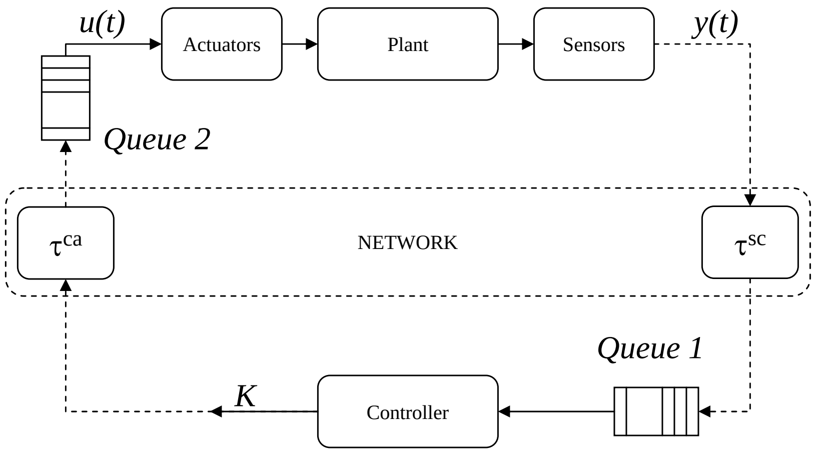

Figure 2.1: NCS with queuing methodology.

control methodology to control a NCS with random delays. Such a method treats the effects of random network delays in a NCS as a linear quadratic Gaussian problem or LQG. Important assumptions in this method are that sensors are time driven and controller and actuator nodes are event driven. The statistics of the delays are known and the total variation of the delays is assumed to be less than one sampling period.

The majority of the research in digital feedback control uses periodic or time driven control. Continuous-time signals are represented by their sampled values at a fixed rate. However, the attention to variable sampling step size methodology has been increasing [15–17]. One of the reasons is because a fixed sampling period might lead to unnecessary use of resources like processor or communication loads. For instance, with fixed sampling time the controllers perform the control computations at a fixed rate all the time, including when no significant event has happened during the process. Variable sampling can relax these situations by using the resources only when specific events occur. [18] analyzes the implementation of event-driven control proposing sensor-based event-driven control by using a control update triggered by the arrival of new data to the sensors. The same paper proposes the reduction of resources used by the controller, like processor loads and communication loads.

Our attention is focused on a different approach to solve the time-delay effect problem, and is based on the creation of a variable sampling period system model. In this dissertation the proposed approach requires knowledge of the future value of the delays. This can be regarded as the prediction of the time delays on the NCS. In our case, the required predictions are based on the implementation of artificial neural networks (NN), for which details are explained later.

The basic idea of the variable sampling step size method using predicted values of the time delays has been studied in [10]. The method was proposed for the system shown in Fig. 2.2 where it considers an internet-based connection only between sensors and a controller, and the controller is directly connected to the plant. The system is represented in continuous time as

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t),\ x(t_0) \\ y(t) &= Cx(t) \end{aligned} \tag{2.1}$$

with feedback controller

$$u(t) = -Kx(t)$$

where all the parameters of the system are explained in detail later.

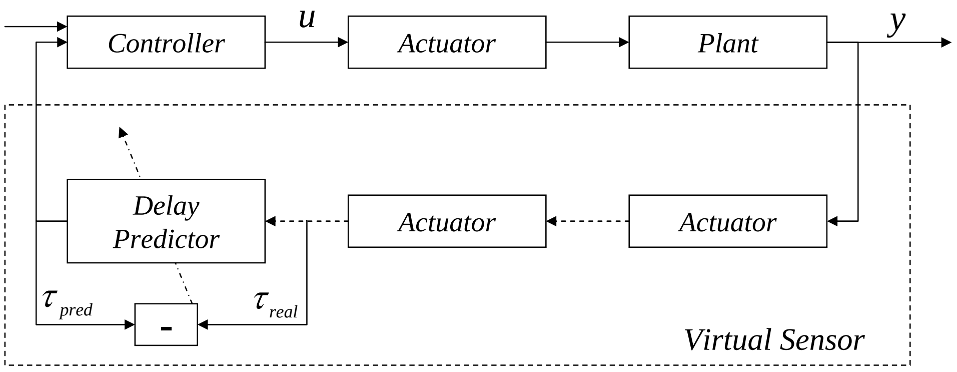

Figure 2.2: Block diagram of the NCS

Under this configuration, the system model is sampled at each sampling step with a different time interval, i.e. different sampling step size. The sensor, network and the delay predictor together create a virtual sensor, which in the figure is indicated by the dotted line box. With this virtual sensor the NCS is turned into an instant sampling time, which ideally does not have time delay. Such sampling period is calculated based on the predictions made by a backpropagation neural network which precedes the controller. Then the controller will assign the sampling period T_k at sampling step k equal to the predicted time delay τ_k. A consequence of this is that the state-space model in 2.1 becomes time varying and its discrete representation takes the form given by 2.2. In short, the model of the NCS depends on the predicted time delay τ_k, which becomes a parameter of the system matrix Φ and the input distribution matrix Γ, i.e.

$$\begin{aligned} x(t_{k+1}) &= \Phi(T_k)x(t_k) + \Gamma(T_k)u(t_k) \\ y(t_k) &= Cx(t_k) \end{aligned} \tag{2.2}$$

with discrete feedback controller

$$u(t_k) = -K(T_k)x(t_k). \tag{2.3}$$

A complete description of this model is given in further chapters. Although [10], does not provide a suitable way to compute the controller parameters for this method, it points out that a control gain K can be calculated for every predicted time delay τ_k. This suggestion is taken into account in our work and a variation for a more suitable controller is presented and analyzed in chapter 5. We investigated further the idea given in [10] about the variable sampling period and the use of a back propagation neural network as a predictor of time delays. This methodology requires the use of historical data to train the neural network. However, though the same paper used a random number generator, the paper does not provide information on the distribution of the data used for the simulations presented.

In [11] the same methodology as in [10] is adopted , but the NCS configuration is slightly different in the sense that a second predictor is added on the actuator side. This is an interesting idea, but it is not mentioned how the second predictor sends the information (from the predictions) to the controller and how the two predictors are coordinated. Further analysis is required on which type of NN is more suitable for the prediction of time delays, and some research which has been done on this topic is mentioned next.

Recently there has been increasing interest in investigating the prediction of time delay that is induced over the internet. This is because internet time delay prediction plays a significant part in improving dynamic performance of many real time applications. One such application is internet based control systems. In [7] the performance of two methods for prediction are compared: the autoregressive model (AR) and an adaptive linear element, also known as ADALINE. In the same paper, real data is used to perform the comparison between the two methods. The results obtained show that the ADALINE, with 10 input nodes and one output, achieved better performance than the AR model. This suggests that the ADALINE can be used in internet-based applications and it should perform as well as or better than the AR model. However, it is not mentioned how the number of inputs was decided for the ADALINE. It is only mentioned that ten input nodes were selected with the purpose of reducing calculation cost of the neural network with the idea of guaranteeing prediction precision. A similar experiment was done in [9], where a multilayer perceptron (MLP) was implemented, with a trial-and-error selection of the number of input nodes, hidden layers, and neurons per hidden layer. The NN used was non adaptive since it was trained off-line and then applied to a on-line prediction process. The authors concluded that when intense fluctuations of the time delay occur the MLP predictor introduces greater prediction errors and that more accurate prediction methods should be created. It can be inferred from this outcome that it is possible to investigate the potential results of modifying the learning process of the NN, for instance, the application of early stopping to truncate the number of epochs or iterations and obtain better generalization.

It is important to notice that there exist two common denominators in the previous works [7] and [9]. First, they both investigated the prediction of internet time delays. Secondly, one is that they did not use a specific methodology to select the best predictor topology, i.e. the number of input nodes and the number of neurons per hidden layer. In [4] an alternative methodology is adopted which provides a suitable topology of a NN. Following is a summary of the procedure. The entire training sequence is divided into sections of the same length. The first section is used to train the NN to minimize the mean squared error (MSE), which is a cost performance function. The NN starts with p inputs and q hidden nodes. Once the parameters of the network that minimize the MSE are known, the trained NN is used with the rest of the sections. This process is repeated for different values of p and q. Then the selected architecture is the one that gives the minimum MSE after completing all tests. This is an organized trial-and-error method to select the appropriate configuration of the NN.

It is important to note that training the NN requires adequate data available to solve a given problem. Our work is focused on the time delays occurring in a NCS that utilizes a wireless network. Therefore, it is important to have good data on delays or at least the best approximation of real data based on specific models. Various approaches to model time delays on NCS have been investigated, and some of them are mentioned in chapter 3.

2.2 Problem Formulation

This section describes the main ideas and approaches proposed in the rest of the dissertation. It starts with a description of the networked control system model studied in later chapters. Also included is a description of the propagation of delays over the NCS, the role of predictions in the idea of variable sampling, and how all these can be used to compensate for the delays. Finally a basic explanation of prediction is given.

2.2.1 Networked Control System Model

Our research is based on the system sketched in Fig. 2.3. This is a representation of a NCS, in which sensors and actuators are linked to a centralized controller via a wireless communication network (dashed line). Unlike the ideal case, the information sent by sensors and controller does not arrive instantly to the destination point, i.e. there exists a lag or delay on the arrival time. In NCS time delays occur while the devices are exchanging data, affecting the arrival time of the information (packets) transmitted to its destination. Such delays are sometimes called network-induced delays. Depending on where the information is sent from, delays can be denoted as sensor-to-controller delays τ^{sc}, computational time delays in the controller τ^{c} and controller-to-actuators delay τ^{ca}. It is assumed that computational delays can be absorbed by τ_{sc} or τ_{ca}. More information about delays is given in later chapters. On the sensors side, timestamped information packets (I_k^{sc}) are created before being transmitted to the controller. Packets have the information from sensors regarding the states of the plant output. They also include the last time delay value from controller to actuator, i.e. τ_{k-1}^{ca}. In later chapters the importance of having the information of τ_{k-1}^{ca} is emphasized. For the moment it is

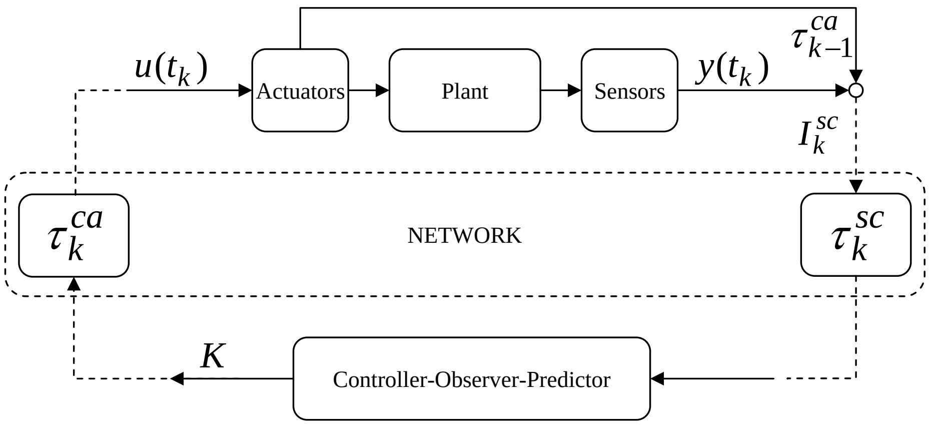

Figure 2.3: Network control system with induced delays τ^{sc} and τ^{ca}

enough to mention that the values τ^{ca}_{k-1} are used by a predictive neural network to compute the one step ahead prediction of the delay τ^{ca}_k that is going to happen. Actuators receive the feedback input from the controller and execute a required action.

2.2.2 Time Delay Propagation

As mentioned before, in NCS the packets travel from controller to actuators and from sensor to controller. The time elapsed from when they are sent until they are received is variable and those variations depend on many factors as we will see in chapter 3. Following the sketch in Fig. 2.4, it can be seen that the packet that contains the information of the plant in the instant (a) departs from the plant output at instant T_k. It will traverse the network and arrive to the input controller with a time lag (time delay) τ^{sc}_k. Assuming the controller uses a very

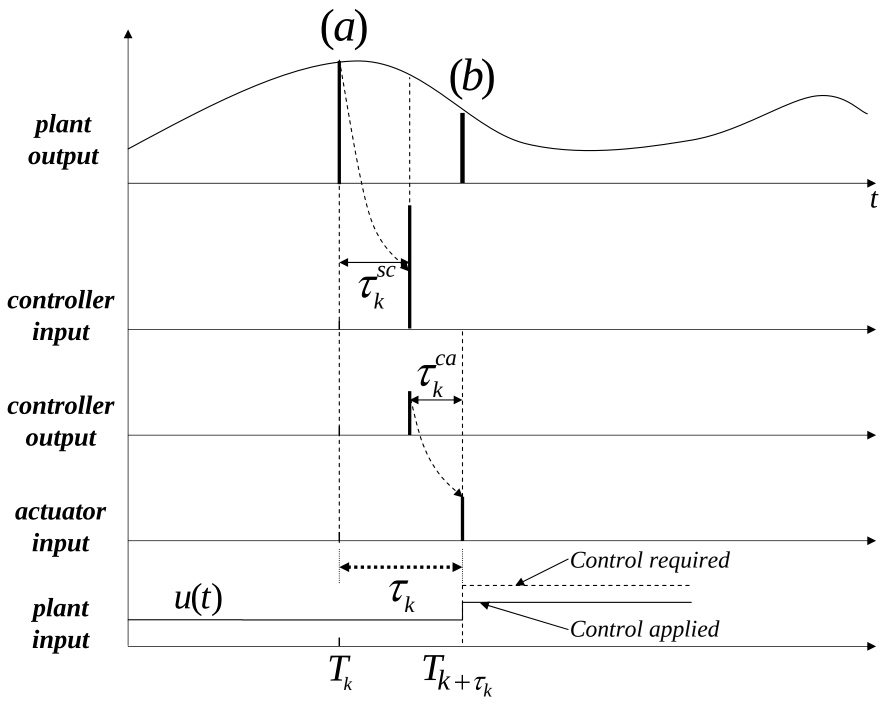

Figure 2.4: Timing diagram of network delay propagations.

small computational time to calculate the control signal, such signal departs in a packet from the controller output, traversing through the network to the actuators. The packet arrives to the actuators with delay τ_k^{ca} and the control signal is input to the plant. By the time the control signal is applied to the plant the control is not updated for the actual conditions of plant in (b).

An interesting idea is to foresee or predict (at sampling step T_k) the total time delay $\tau_k = \tau_k^{sc} + \tau_k^{ca}$ and calculate a corresponding control signal more suitable to the conditions of the plant at time $T_{k+\tau_k}$, i.e. at point (b). In this way by the time the packet reaches the actuators it will input a more suitable control value. i.e. the required control. This is explained in the following way:

Since the arrival time of packets to the actuators is variable, we can think that by adopting event-driven actuators and controller the NCS becomes a variable sampling system. Then the idea is to calculate (at sampling step T_k) the total time delay $\tau_k = \tau_k^{sc} + \tau_k^{ca}$ that is going to happen, and perform the next sampling of the system model at $T_{k+\tau_k} = \tau_k$ and calculate its corresponding control signal that will arrive to the true system at $T_{k+\tau_k}$. Furthermore, since time delays are variable, the sampling is variable as well. This thesis shows a proposed NCS model using the ideas exposed above which is introduced in section 2.2.3.

2.2.3 The Idea of Prediction

Prediction is the forecasting side of information processing. In continuous time, the aim is to obtain at time t the information about $y(t + \lambda)$ for some $\lambda > 0$, i.e. what the quantity in question, $y\{\cdot\}$, will be subsequent to the time at which the information is produced. Measurements until time t can be used to do this [24]. In discrete time, a real discrete random signal $\{y(k)\}$, where k is the discrete index and $\{\cdot\}$ denotes the set of values, is most commonly obtained by sampling an analogue measurement [21].

The principle of prediction of a discrete time signal is represented in Fig. 2.5. The value of the signal a_k is predicted on the basis of a linear combination of p past values, i.e. a linear combination of $(a_{k-1}, a_{k-3}, a_{k-3}, ..., a_{k-p})$, to create a

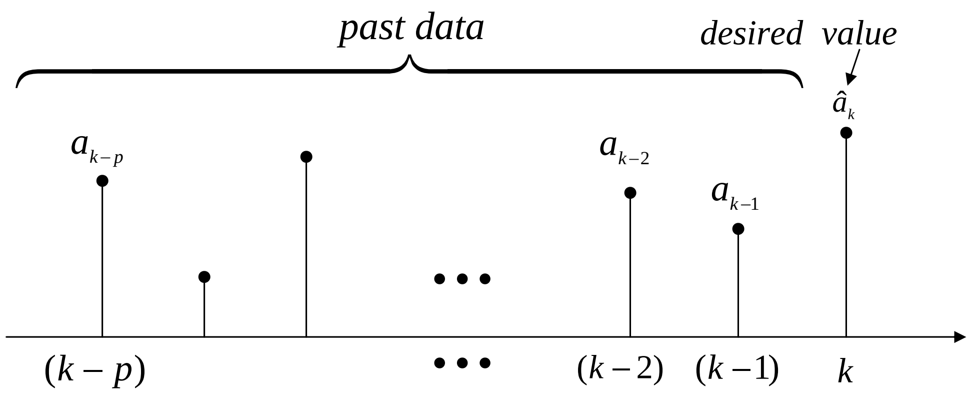

Figure 2.5: One step ahead prediction

prediction $\hat{a}_k$ of a_k. Thus, the prediction error $e(k)$ becomes

$$e(k) = d_k - \hat{a}_k$$

where d_k is the desired value or target and $\hat{a}_k$ is the one step ahead predicted value. In further chapters the sequence to be predicted will use the notation τ_k since the parameter to be predicted is the time delay occurring at the step time k.

Chapter 3 – Delay model

It is important to know the types of delay that occur in a network as well as their weight in the total delay which eventually will affect the performance of a dynamic networked control system. One type of delay is processing delay, which is the time required to examine the packet's header and determine where to direct the packet. Another type is queuing delay which is due to the time the packet waits to be transmitted onto the link. Transmission delay is the amount of time required to transmit all of the packet's bits into the link. Also a type of delay is propagation delay which is the time a packet requires to propagate from the beginning of a link to router in a point B. Another component is the speed of the signal transmission and the distance between the source and destination. The contributions of these delay components can vary significantly. For example, the propagation delay alone can be just a couple of microseconds if the link connecting two routers is within the same university campus, but this delay can be of hundreds of milliseconds for two routers interconnected by a geostationary satellite link. For this reason propagation delay can be the dominant one in the total delay of the network loop [48]. The other delays can be also negligible or significant depending on the parameters of the network. Due to the characteristics of each of these delays, some assumptions are taken into account for our analysis and research. In our work we are assuming that queuing delay, processing delay and transmission delay are small and are absorbed by either the sensor to controller delay, τ_{sc}, or by the controller to actuator delay, τ_{ca}.

The phenomenon of time delay depends on the communication protocol used [2], the network traffic patterns, the channel conditions [25], network equipment, throughput and congestion condition on the network during transmission [26]. Such delays can be of different natures, either constant, variable or random [5]. In the case of delays over the internet, time delays are characterized by the processing speed of nodes, the load of nodes, the connection bandwidth, the amount of data, the transmission speed, etc. [7]. This shows the importance of creating a representation of the network delays to be able to create a suitable control for a certain NCS. Of course, the easiest way to get a model of the delays is by estimating the probability distribution of the delays directly from actual delay measurements. Some times this is not possible and different ways of modeling time delays have been proposed depending on the characteristics of the network used in a NCS. Delays can be modeled as constant, periodic, or random. A model that is more suitable for an analysis depends on the protocol used by the system network. A model has to be devised under a series of assumptions and/or observations, e.g. in [5] time delays are modeled by an underlying Markov chain. In [14] and [19] it is observed that an exponential distribution is well-defined for time delays in a WLAN network. In [26] a gamma distribution is utilized to represent the behavior of delays in a WiFi network under the assumption that the behavior of the network has similar characteristics to this type of distribution.

Another effect of the network over time delays is that loads in the network change over time. For this reason it is logical to think that the distribution of the delays have lower mean if the network has low load, and a higher mean if the network has high load. Furthermore, in a realistic model, the delay variation is smaller for low loads, and larger for high loads [27]. Also, it is important to note that in real communication systems, time-delays are usually correlated with the last time delay [5] [29]. In addition, when the load is low, network delays are quite deterministic, the network is often *idle* when we want to transmit. On the other hand, when the network load is high it is possible to have short delays but we could also have to wait for multiple messages to be sent. In our research, the condition of correlated time delays in a network is simulated by creating a correlated data sequence ψ. This sequence was created by using the recursive equation [28]

$$\psi_k = \beta\psi_{k-1} + \epsilon_k \tag{3.1}$$

where ϵ is a sequence of independent exponentially distributed random numbers, and $\beta \in (0, 1)$ is the parameter that determines the degree of correlation between ψ_k and ψ_{k-1}. The value of ψ_0 is selected to be very small so it will not have impact on the initial values of the sequence. The sequence is then normalized using

$$\psi_{norm} = \frac{\psi\bar{\epsilon}}{\bar{\psi}} \tag{3.2}$$

where $\bar{\epsilon}$ and $\bar{\psi}$ are the means of the random numbers sequence and correlated data sequence, respectively. Then, ψ_{norm} is applied to the predictor. In later chapters, predictions of this correlated sequence will be shown. The model (3.1) was selected for the purpose of generating correlated sequences of time delays for a selected value of β in controlled experiments.

3.1 Sampling systems with delay

Given that a NCS operates over a network, data transmission between controller and a remote system is affected by induced network delays. Such delays can be categorized depending on the direction of data transmission, i.e. sensor-to-controller delay τ^{sc} and controller-to-actuators delay τ^{ca}. There is also the processing-time delay, which is the time required by the controller to generate the control signal. Another delay in the NCS is the inherent delay of the actuators since their responses are not instantaneous. Normally the last two types of delay are assumed to be negligible for analysis purposes. The representation of a discrete linear time-invariant system with time-delay affecting the feedback input is described next. Given the continuous-time representation of the dynamic system subject to input delays

$$\dot{x}(t) = Ax(t) + Bu(t - \tau),\ x(t_0), \tag{3.3}$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$, represent the system states and the controlled input, respectively. In this case A and B are constant matrices of appropriate dimensions. It is assumed that the system has fixed-period sampling intervals and the total delay τ_k is assumed to be less than the sampling period T_k. Then the discrete system is described as

$$x(t_{k+1}) = \Phi x(t_k) + \Gamma_1 u(t_k) + \Gamma_2 u(t_{k-1}) \tag{3.4}$$

where

$$\begin{aligned} \Phi &= e^{AT_k}, \\ \Gamma_1 &= \int_0^{T_k-\tau_k} e^{A\lambda} B d\lambda \\ \Gamma_2 &= \int_{T_k-\tau_k}^{T_k} e^{A\lambda} B d\lambda \end{aligned} \tag{3.5}$$

This can be represented in state space model as

$$\begin{bmatrix} x(t_{k+1}) \\ u(t_k) \end{bmatrix} = \begin{bmatrix} \Phi & \Gamma_2 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(t_k) \\ u(t_{k-1}) \end{bmatrix} + \begin{bmatrix} \Gamma_1 \\ I \end{bmatrix} u(t_k) \tag{3.6}$$

This is a classical approach to sampling dynamic systems with delays. Notice that (3.6) uses the extra state variables to describe the delays. A similar derivation is done for the analysis of delays longer than a sampling period. The reader is referred to [22] and [47] for further reading. In chapter 5 we address and explain in detail our proposed model based on the observer-based variable sampling which is explained in detail.

Chapter 4 – Artificial Neural Network for Prediction

In the literature, it has been shown that neural networks have the ability to approximate virtually any function of interest to any degree of accuracy [38] under the assumption that the neural network is provided with sufficiently many hidden units. Based on this ability, neural networks have been applied extensively in time series predictions [3], [4], [7], [31], [43,44]. This chapter begins with the fundamentals of neural networks and the algorithm to update its parameters (weights). It is followed by the description of the methodology and techniques we use to select the most appropriate topology and/or architecture of the predictive neural network such as validation performance and early stopping. After that, the chapter continues with the presentation of several numerical simulations preformed for the selection of the NN topology for the prediction of a given time delay sequence. Comparative simulation results of several topologies are also given. This chapter is closed by a section on conclusions.

4.1 Fundamentals

4.1.1 Time Delay Neural Network

A mentioned earlier we want the neural network to capture the progression of a time series. In this case the neural network needs to have certain memory of previous events. This can be achieved by feeding to the network delayed versions of the time series, which in our case is the sequence of time delays in a NCS, i.e. $a_{k-i},\ i = 1, 2, ..., p$. We later refer to this as the tapped delay line of the neural network. For this reason we suggest the application of a multilayer feedforward neural network with a tapped delay line input layer. Such network is classified as a dynamic network and is called time-delay artificial neural network (TDNN).

We denote the data in the input layer of the TDNN at a particular instant k as a_k, where a may be a vector. This can be described as

$$\hat{a}_k = \mathcal{NN}(a_{k-1}) \tag{4.1}$$

and since we are using history values the expression (4.1) can be rewritten as

$$\hat{a}_k = \mathcal{NN}(a_{k-1}, a_{k-2}, ..., a_{k-i}, ..., a_{k-p}) \tag{4.2}$$

where $\hat{a}_k$ is the one step ahead predicted value of a_k, p is the number of successive past observations, and $\mathcal{NN}(\cdot)$ represents the neural network. The argument $(a_{k-1}, a_{k-2}, ..., a_{k-j}, ..., a_{k-p})$ in equation (4.2) represents the tapped delay line of the TDNN and it is the input layer represented in Fig. 4.1. As wee will see later, the number of hidden layers and the number of neurons in each layer (topology) depends on the application and they are generally determined by using a trial and error technique, although one can find in the literature different suggestions to find an appropriate NN topology.

In our research we use a multilayer TDNN, and it has the form depicted in Fig.

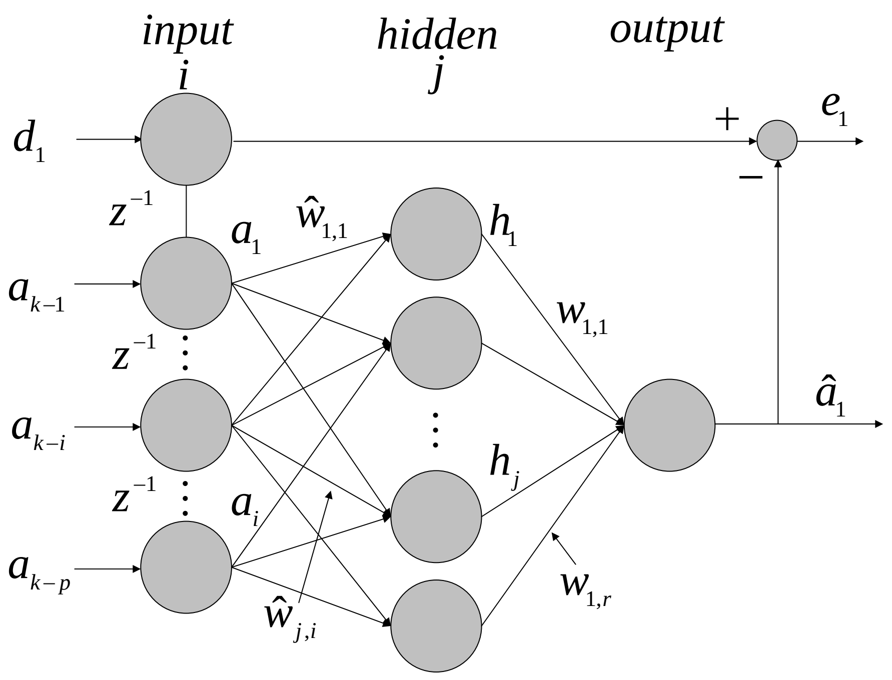

Figure 4.1: Time delay neural network.

4.1. From this figure we can derive the corresponding weight updates and compute the error that backpropagates through the layers of the TDNN to minimize a cost function and achieve a minimum desired error between the predicted values and the actual values of the time series.

4.1.2 Backpropagation Algorithm

In Fig. 4.1 the layers are defined by: Input (i), hidden (j) and output. The nodes of each layer have a corresponding output, i.e. input node a_i, hidden node output h_j, output node value $\hat{a}_1$. Let us define the error signal at the output layer (output of neuron 1) at iteration n as

$$e_1(n) = d_1(n) - \hat{a}_1(n), \tag{4.3}$$

where $d_1(n)$ is the desired output value (target) and $\hat{a}_1(n)$ is the output node value. We also define the instantaneous value of the squared error of the output neuron 1 as

$$E(n) = \frac{1}{2} e_1^2(n). \tag{4.4}$$

We denote N as the total number of samples in the training set. Then the mean squared error is given by the summation of $E(n)$ over all n and divided by the set size N, i.e.

$$\bar{E}(n) = \frac{1}{N} \sum_{n=1}^{N} E(n). \tag{4.5}$$

The mean squared error is called the cost function and it is a measure of training performance. Let us define now the output of the output layer as the linear combination of the weights $w_{1,j}(n)$, namely,

$$\hat{a}_1(n) = \sum_{j=1}^{r} w_{1,j}(n) h_j(n), \tag{4.6}$$

where r is the number of inputs applied to the output neuron 1. To update the weights $w_{1,j}$ we need to apply a correction $\Delta w_{1,j}(n)$ to them which is obtained by using the instantaneous gradient

$$\frac{\partial E(n)}{\partial w_{1,j}(n)} = \frac{\partial E(n)}{\partial e_1(n)} \frac{\partial e_1(n)}{\partial \hat{a}_1(n)} \frac{\partial \hat{a}_1(n)}{\partial w_{1,j}(n)}, \tag{4.7}$$

where by differentiating both sides of equation (4.3) with respect to $\hat{a}_1(n)$ and (4.4) with respect to $e_1(n)$ we get

$$\frac{\partial e_1(n)}{\partial \hat{a}_1(n)} = -1, \tag{4.8}$$

$$\frac{\partial E(n)}{\partial e_1(n)} = e_1(n), \tag{4.9}$$

respectively, and by differentiating both sides of (4.6) with respect to $w_{1,j}(n)$ we get

$$\frac{\partial \hat{a}_1(n)}{\partial w_{1,j}(n)} = h_j(n). \tag{4.10}$$

Finally, from equations (4.8), (4.9) and (4.10) in (4.7) we have

$$\frac{\partial E(n)}{\partial w_{1,j}(n)} = -e_1(n)h_j(n) \tag{4.11}$$

Then, the correction $\Delta w_{1,j}(n)$ to update the weights $w_{1,j}(n)$ is defined as

$$\Delta w_{1,j}(n) = -\eta \frac{\partial E(n)}{\partial w_{1,j}(n)} = \eta e_1(n)h_j(n), \tag{4.12}$$

where η is the learning rate defined by the user. Equation (4.12) can be expressed in terms of the local gradient $\delta_1(n)$, which points to required changes in the weights. Then

$$\delta_1(n) = -\frac{\partial E(n)}{\partial e_1(n)} \frac{\partial e_1(n)}{\partial \hat{a}_1(n)} = e_1(n), \tag{4.13}$$

and (4.12) is expressed as

$$\Delta w_{1,j}(n) = \eta \delta_1(n) h_j(n), \tag{4.14}$$

Then the weight update equation is

$$\begin{aligned} w_{1,j}(n+1) &= w_{1,j}(n) + \Delta w_{1,j}(n) \\ &= w_{1,j}(n) + \eta \delta_1(n) h_j(n). \end{aligned} \tag{4.15}$$

Now, the next step is to backpropagate the error found before to adjust the weights of the hidden layer. This is not much more difficult and the idea is the same as for the output layer. We now need to adjust the input-hidden layer weights. Recall that the activation at the output layer was a linear activation function. In the case of the hidden layer, a sigmoidal activation function of the form

$$f(x) = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{4.16}$$

whose derivative is

$$f'(x) = sech^2(x) = (1 - f^2(x)), \tag{4.17}$$

where x is the argument of the function, the delta for the hidden layer ($\delta_j(n)$) is obtained following the previous procedure, then we have

$$\delta_j(n) = -\frac{\partial E(n)}{\partial h_i(n)}\, f'(\sum_{i=1}^{p} \hat{w}_{j,i}(n) a_i(n)) \tag{4.18}$$

were p is the total number of inputs applied to the hidden neuron j. To calculate

$\partial E(n)/\partial a_i(n)$ we have the error in the output layer

$$E(n) = \frac{1}{2}e_1^2(n) \tag{4.19}$$

which is the same as (4.4) then,

$$\frac{\partial E(n)}{\partial h_j(n)} = \frac{\partial E(n)}{\partial e_1(n)}\frac{\partial e_1(n)}{\partial h_i(n)} = e_1(n)\frac{\partial e_1(n)}{\partial h_i(n)}. \tag{4.20}$$

Using partial derivatives in $\partial e_1(n)/\partial h_j(n)$ we express (4.20) as

$$\frac{\partial E(n)}{\partial h_j(n)} = e_1(n)\frac{\partial e_1(n)}{\partial \hat{a}_1(n)}\frac{\partial \hat{a}_1(n)}{\partial h_j(n)} \tag{4.21}$$

from the error equation (4.3)

$$e_1(n) = d_1(n) - \hat{a}_1(n) \tag{4.22}$$

then

$$\frac{\partial e_1(n)}{\partial \hat{a}_1(n)} = -1 \tag{4.23}$$

and from (4.6) we have

$$\frac{\partial \hat{a}_1(n)}{\partial h_j(n)} = w_{1,j}(n) \tag{4.24}$$

From equations (4.23), (4.24) in (4.21) we get

$$\begin{aligned}\frac{\partial E(n)}{\partial h_j(n)} &= -e_1(n)w_{1,j}(n) \\ &= -\delta_1(n)w_{1,j}(n),\end{aligned} \tag{4.25}$$

where we have used the local gradient $\delta_1(n)$ given in (4.13), then using (4.25) in (4.18) we can get the local gradient $\delta_j(n)$ for the hidden neuron j, given by

$$\delta_j(n) = \delta_1(n)w_{1,j}f'(\sum_{i=1}^{p} \hat{w}_{j,i}a_i(n)). \tag{4.26}$$

Using the activation function (4.17) in (4.26) we have

$$\delta_j(n) = \delta_1(n)w_{1,j}(1 - f^2(\sum_{i=1}^{p} \hat{w}_{j,i}a_i(n))), \tag{4.27}$$

which gives the correction factor $\Delta w_{j,i}$ as

$$\Delta\hat{w}_{j,i}(n) = \eta\delta_1(n)w_{1,j}(1 - f^2(\sum_{i=1}^{p} \hat{w}_{j,i}a_i(n)))\ a_i(n). \tag{4.28}$$

Finally, the weight update for the hidden layer is given as

$$\hat{w}_{j,i}(n+1) = \hat{w}_{j,i}(n) + \eta\delta_1(n)w_{1,j}\left(1 - f^2\big(\sum_{i=1}^{p} \hat{w}_{j,i}a_i(n)\big)\right)\ a_i(n) \tag{4.29}$$

Notice that it is not possible to have desired outputs for the hidden layers. This is why the backpropagation starts in the output layer passing back the information needed [39]. Something important to remember is the application of a *momentum* when updating the weights of the TDNN. The term *momentum* is based on the principles from Physics that bodies in motion keep moving until another force acts on them. Momentum allows the network to learn faster when plateaus in the error surface exist [46]. Also, the momentum may help to escape possible local minima in the error function as seen in Fig. 4.2. The approach is to alter the weight update equations, so the weights can be updated with a component of the previous weight updates. A typical value of the momentum is $\alpha = 0.9$, which is the value used in our research.

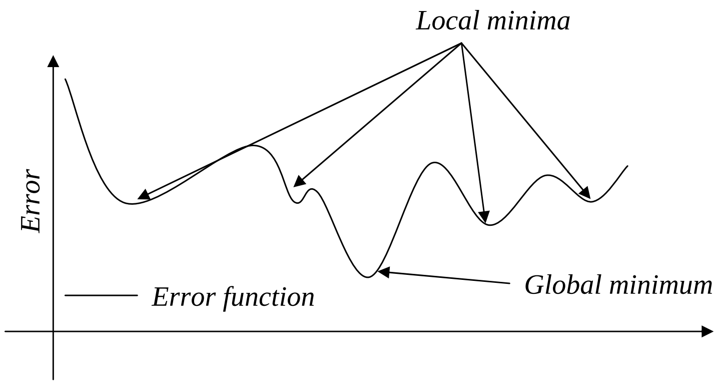

Figure 4.2: Local and global minima of the error function.

Let us consider the application of momentum, then using (4.27) and (4.28) and the momentum factor defined by α we have

$$\Delta \hat{w}_{j,i}(n) = \alpha \Delta \hat{w}_{j,i}(n-1) + \eta \delta_j(n)\ a_i(n). \tag{4.30}$$

By doing this change each weight update is closer to the last update.

The backpropagation algorithm is described in Table 4.1. It is possible to speed up the convergence of the training process by implementing one of the many algo-

Table 4.1: Back Propagation Algorithm.

```
Backpropagation Algorithm
For each epoch:
  -Present input to the network
     Propagate signal forward:
       Compute hidden units values
       Compute output values
     Find error
       Compute output layers deltas
       Compute hidden layer delta
     Compute gradient for each weight
     Update each weight
  -Present newt input
Repeat this process until MSE is satisfactory
```

rithms found in the literature, the most widely used is the Lebenberg-Marquardt training algorithm and the reader is referred to [13] for further information.

4.1.3 Transformation of input data

In neural networks, activation functions are centered on certain values in their output space. For instance, the logistic function is centered around 0.5. and the sigmoidal function *tanh* has its center around zero. To have good predictions it is necessary to match the range of the input data, with the range of the chosen activation function, [21]. There are different methodologies to do this, namely, normalization, rescaling, standardization, and normalization of eigenvectors. Since our neural network has a sigmoidal (s-shape) activation function centered around zero, the transformation or preprocessing of the data was made using the following equations.

$$\begin{aligned} midrange &= \frac{1}{2} max_i(X_i) + min_i(X_i) \\ range &= max_i(X_i) - min_i(X_i) \\ S_i &= \frac{X_i - midrange}{range/2}, \end{aligned} \tag{4.31}$$

where X_i is the ith input data value. The same results given by (4.31) can be obtained by the simplified equation

$$Y = \frac{(Y_{max} - Y_{min})(X - X_{min})}{X_{max} - X_{min}} + Y_{min}, \tag{4.32}$$

where, for our purposes, $Y_{max} = +1$ and $Y_{min} = -1$, X is the input data, X_{max} and X_{min} are the maximum and minimum value of the input data. Then, the value Y falls into the interval $[-1, 1]$, which matches the range of the activation function $tanh$ used in this work.

4.2 Neural Network Topology

According to literature surveys, there is not a single configuration that is adequate for all applications. The topology must, therefore, be selected by a process of trial and error [37]. There are some suggestions to generate an appropriate neural network topology. One of them is called the Baum-Haussler rule, known also as the rule of thumb [40]. This rule gives an approximation of how many hidden nodes are needed in the hidden layer of the neural network. It is expressed as:

$$N_{hidden} \leq \frac{N_{train} \times N_{pts} \times E_{goal}}{N_{pts} + N_{outputs}} \tag{4.33}$$

where N_{hidden} is the number of hidden nodes, N_{train} is the size of the training set, E_{goal} is the error goal or tolerance, N_{pts} is the number of data points used in the tapped line of the NN, N_{output} is the number of output nodes. Using this method also requires to determine N_{pts} which is given by the user. This means that several values of N_{pts} have to be tried. Using this rule it is possible to create several NN configurations to test. The selection of the most adequate topology for prediction, given a particular case, is based on the performance of the network.

In [37], it is stated that many of the studies where NN's were used for prediction suffered from validation or implementation problems. Also, it points out the importance of doing an effective implementation of the NN and, more importantly, validate it. We implemented a procedure which validates and tests each topology created. In the literature we can not find a consistent or standard method or procedure to generate an adequate topology for the predictive NN. The most common practice to find such NN is by trial and error, this can take some time and many iterations before getting good results, such results in our case would be the predictions of the time delay on a NCS. The time required in the trial and error process can be reduced drastically by simply implementing a process called cross-validation or validation.

Validation involves partitioning a sample of data into subsets, performing the analysis (training) on one subset, named training set, and validating the analysis on the other subset named the validation set, the test is done on the testing set. Validation is normally done during the learning process, this helps to determine when to stop the learning and also it controls overfitting or overtraining.

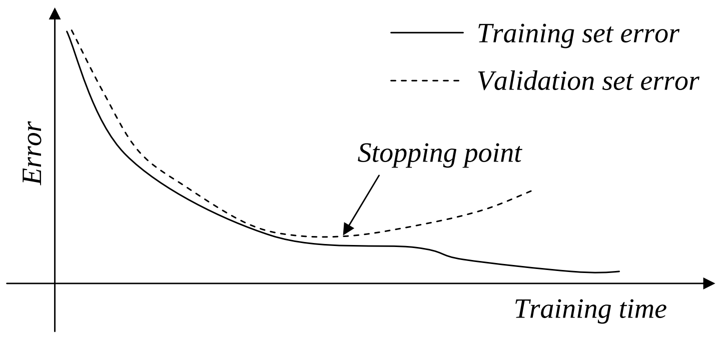

Figure 4.3: Early stopping technique.

During the training process, the validation is done by applying it to the NN validation data set. The error on the validation set is monitored during the training process. The validation error normally decreases during the initial phase of training, as does the training set error. However, when the network begins to overfit the data, the error on the validation set typically begins to rise. It is at this point when the training is stopped (see Fig. 4.3). In other words, the stopping occurs when the outcome of the performance error (cost function) does not have any further change and/or it starts increasing again. Then, the topology with the minimum performance error in the validation process is used for testing as the forecaster of the time delay sequence. This technique, also called early stopping, is used to check, or cross-validate, that the trained network is able to generalize its learning to new data. It also will allow the user to determine how good the NN is for generalization.

Table 4.2: Algorithm for TDNN topology selection.

	Neural Network Topology Selection
1	Transformation of input data: $\left\{ y = \frac{(y_{max}-y_{min})(x-x_{min})}{(x_{max}-x_{min})} + y_{min} \right.$
2	Divide data: Training, Validation
3	Define input data and targets
4	Define max number of nodes input layer
5	Define max number of nodes hidden layer
6	Set initial TDNN topology:
	1 node in input layer
	1 node in the hidden layer
7	Start training and validation
8	Stop training using early stopping
9	Increment by one the number of hidden nodes
10	Record TDNN validation performance
11	Go to 7
12	If max number hidden nodes:
	Increment by one the number of input nodes
	Set number hidden nodes to 1
13	Go to 7
14	STOP if max number of hidden layer nodes

Generalization measures the ability of NNs to recognize patterns, or in our case predict values, outside the training sample. The accuracy rates achieved during the learning phase typically define the bounds for generalization. If performance on a new sample is similar to that in the convergence phase, the NN is considered to have learned well and it is said that generalization has taken place. In Table 4.2 describes the algorithm to select the TDNN topology based on validation and early stopping described before.

Large topologies will not necessarily produce better results, they might probably converge to a desired value, but the computation time needed will increase due the complexity of the NN. This is why it is considered impractical to have large NN topologies unless a specific application requires it. In some cases relatively large topologies have been used for prediction of time delays, for example [9] implemented a NN for prediction of time delays over the internet with one input tapped line of 20 nodes, two hidden layers with 15 neurons per hidden layer and one single output.

4.3 Experimental Results

4.3.1 Selection of Neural Network Topology

As we mentioned before, to train a neural network using the early stopping method, it is necessary to partition the set of data available into sets for training, validation and test. In the literature there is no specific rule governing the splitting of the data. However, it is generally agreed that most data points should be used for model building. In our prediction simulation study and using equations (3.1) and (3.2), we generated four different time delay sequences ψ which correspond to delays in a NCS, we are assuming they have different correlation factor β given in Table 4.3. Each sequence has 500 sample points and the same mean $\mu = 0.02$. The splitting of the data is made in the following manner: we use the first 300 observations for training, the next 100 points for validation, and the last 100 points for testing. To generate the sequence of exponential random numbers ϵ in equation (3.1) we use the exponential random number generator from Matlab ®.

Table 4.3: Correlation factors

β	0.0	0.25	0.5	0.75

Tables 4.4 to 4.7 present the results for different TDNN topologies implemented. We denote each entry of the table by $NET_{i,h,1}$, where i is the number of nodes at the input tapped delay line, h is the number of hidden nodes in the hidden layer and 1 is the number of output nodes. It is also known in the literature that having just one hidden layer gives good results on prediction [3], [10], [7], [41]. Having this in mind, plus knowing that the lesser the complexity of a neural network the simpler the computations, we implemented several different topology configurations for the TDNNs with just one hidden layer.

Each configuration was created by varying the number of input nodes and the number of hidden nodes in increments of one. The maximum number of nodes in the input tapped delay line was 10 and for the hidden layer was also 10. Since every time the network is trained its initial parameters are initialized in a random way, the network might produce different results. This is why to select the best topology, each of the 100 possible TDNN configurations was trained and validated 10 times, the results expressed in Tables 4.4 to 4.7 are the average of the root mean square error (RMSE) of the validation process. Recall that the performance

Table 4.4: RMSE validation performance for a series with: $\mu = 0.02$, $\beta = 0.0$.

$i\backslash h$	1	2	3	4	5	6	7	8	9	10
1	0.0243	0.0278	0.0254	0.0246	*0.0244**	0.0507	0.0248	0.0256	0.0298	0.0297
2	0.0242	*0.0243**	0.0270	*0.0242**	0.0247	0.0285	0.0251	0.0264	*0.0248**	0.0270
3	0.0244	0.0244	0.0244	0.0242	0.0245	0.0244	0.0254	*0.0247**	0.0260	0.0293
4	*0.0242**	0.0248	*0.0242**	0.0243	0.0253	0.0269	*0.0243**	0.0419	0.0256	0.0337
5	0.0249	0.0253	0.0264	0.0248	0.0331	0.0260	0.0265	0.0263	0.0261	0.0278
6	0.0246	0.0246	0.0247	0.0255	0.0256	0.0616	0.0253	0.0254	0.0274	0.0333
7	0.0252	0.0256	0.0259	0.0255	0.0246	0.0273	0.0255	0.0254	0.0260	0.0306
8	0.0254	0.0263	0.0251	0.0271	0.0271	*0.0235**	0.0255	0.0257	0.0259	0.0532
9	0.0251	0.0255	0.0261	0.0398	0.0246	0.0267	0.0262	0.0280	0.0262	0.0271
10	0.0253	0.0246	0.0259	0.0259	0.0257	0.0252	0.0261	0.0277	0.0251	*0.0267**

i = nodes in the tapped delay line, h = nodes in the hidden layer.

Table 4.5: RMSE validation performance for a series with: $\mu = 0.02$, $\beta = 0.25$.

$i\backslash h$	1	2	3	4	5	6	7	8	9	10
1	0.0161	0.0162	0.0163	0.0163	0.0161	0.0165	0.0160	0.0161	0.0165	0.0162
2	0.0156	0.0155	0.0156	0.0156	0.0157	0.0162	0.0609	0.0162	0.0166	0.0169
3	0.0158	0.0161	0.0160	0.0163	0.0156	0.0162	0.0161	0.0173	0.0161	0.0159
4	0.0162	0.0160	0.0163	0.0160	0.0157	0.0161	0.0160	0.0161	0.0163	0.0166
5	0.0151	*0.0150**	0.0158	0.0155	*0.0155**	0.0153	0.0158	*0.0153**	*0.0154**	0.0159
6	0.0151	0.0155	0.0160	0.0161	0.0160	0.0152	0.0160	0.0159	0.0157	*0.0158**
7	0.0155	0.0152	0.0163	0.0154	0.0158	*0.0151**	*0.0155**	0.0154	0.0155	0.0162
8	0.0152	0.0154	*0.0154**	0.0160	0.0166	0.0159	0.0161	0.0154	0.0183	0.0159
9	0.0152	0.0155	0.0154	*0.0152**	0.0158	0.0166	0.0161	0.0169	0.0170	0.0161
10	*0.0150**	0.0157	0.0157	0.0167	0.0158	0.0157	0.0168	0.0170	0.0162	0.0168

i = nodes in the tapped delay line, h = nodes in the hidden layer.

Table 4.6: RMSE validation performance for a series with: $\mu = 0.02$, $\beta = 0.5$.

$i\backslash h$	1	2	3	4	5	6	7	8	9	10
1	0.0098	0.0098*	0.0097*	0.0097*	0.0097*	0.0098	0.0097*	0.0097	0.0097*	0.0099
2	0.0098	0.0100	0.0098	0.0099	0.0098	0.0097	0.0098	0.0098	0.0098	0.0098*
3	0.0100	0.0100	0.0098	0.0098	0.0099	0.0095*	0.0099	0.0101	0.0104	0.0099
4	0.0098	0.0098	0.0098	0.0099	0.0099	0.0100	0.0101	0.0097*	0.0102	0.0169
5	0.0099	0.0099	0.0098	0.0101	0.0100	0.0109	0.0149	0.0099	0.0103	0.0101
6	0.0098*	0.0100	0.0107	0.0103	0.0101	0.0103	0.0099	0.0106	0.0110	0.0101
7	0.0115	0.0101	0.0100	0.0102	0.0100	0.0101	0.0101	0.0108	0.0100	0.0107
8	0.0100	0.0101	0.0100	0.0100	0.0124	0.0103	0.0101	0.0103	0.0104	0.0104
9	0.0116	0.0101	0.0107	0.0108	0.0104	0.0111	0.0104	0.0102	0.0101	0.0145
10	0.0101	0.0102	0.0108	0.0102	0.0103	0.0107	0.0105	0.0111	0.0104	0.0109

i = nodes in the tapped delay line, h = nodes in the hidden layer.

Table 4.7: RMSE validation performance for a series with: $\mu = 0.02$, $\beta = 0.75$.

$i\backslash h$	1	2	3	4	5	6	7	8	9	10
1	0.0047*	0.0075	0.0047	0.0046*	0.0046*	0.0045*	0.0047	0.0046	0.0046*	0.0050
2	0.0048	0.0050	0.0047	0.0053	0.0047	0.0049	0.0046*	0.0048	0.0061	0.0047*
3	0.0048	0.0048	0.0049	0.0048	0.0048	0.0047	0.0047	0.0047	0.0048	0.0049
4	0.0049	0.0047	0.0047*	0.0048	0.0054	0.0046	0.0047	0.0045*	0.0046	0.0047
5	0.0048	0.0048	0.0047	0.0048	0.0047	0.0048	0.0047	0.0048	0.0048	0.0089
6	0.0048	0.0047*	0.0048	0.0051	0.0047	0.0048	0.0049	0.0048	0.0047	0.0047
7	0.0048	0.0049	0.0048	0.0046	0.0048	0.0050	0.0047	0.0048	0.0050	0.0047
8	0.0050	0.0049	0.0049	0.0050	0.0049	0.0049	0.0050	0.0048	0.0048	0.0052
9	0.0051	0.0051	0.0049	0.0050	0.0050	0.0051	0.0050	0.0051	0.0050	0.0049
10	0.0051	0.0050	0.0050	0.0048	0.0049	0.0051	0.0054	0.0051	0.0050	0.0052

i = nodes in the tapped delay line, h = nodes in the hidden layer.

criterion used for the early stopping was the minimum MSE. RMSE is a frequently used measure of the differences between values predicted and the actual value.

The best topology was chosen based on the best validation performance for each delay sequence and its generalization capacity. For each of the four cases studied above, there were 10 possible TDNN candidates (marked by a star *) among the 100 tested . From those 10 candidates, the one with good performance but also with good generalization was selected. At the end we had four candidates with different topology corresponding to a particular delay sequence. Such candidates were used to predict an unknown time delay sequence, i.e. a new sequence of delays with similar statistics as the one previously studied. The selected topologies are listed in Table 4.8.

4.3.2 Prediction results

The selected TDNN topologies were used to predict four new sequences with same statistics already studied above. Such sequences had 2000 points, mean $\mu = 0.02$ and correlation factors $\beta = 0.00, 0.25, 0.50$ and 0.75. The TDNNs where trained using the first 300 points of each series. Once trained, the TDNNs were ready to predict the other 1700 points of the series. For clarity, the plots have the last 300 points of the series.

The plots 4.4 to 4.11 show predictions made by two TDNNs with same topology, which differ in the way they work, namely, one TDNN is non-adaptive and the other one is adaptive. Fig. 4.4 shows the prediction of a sequence with correlation

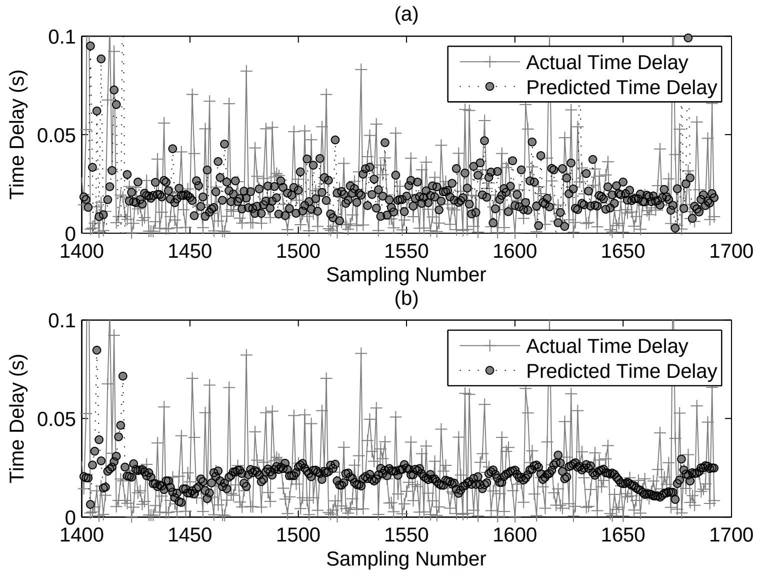

Figure 4.4: Prediction of delay sequence with $\beta = 0.0$ using $NET_{8,6,1}$.

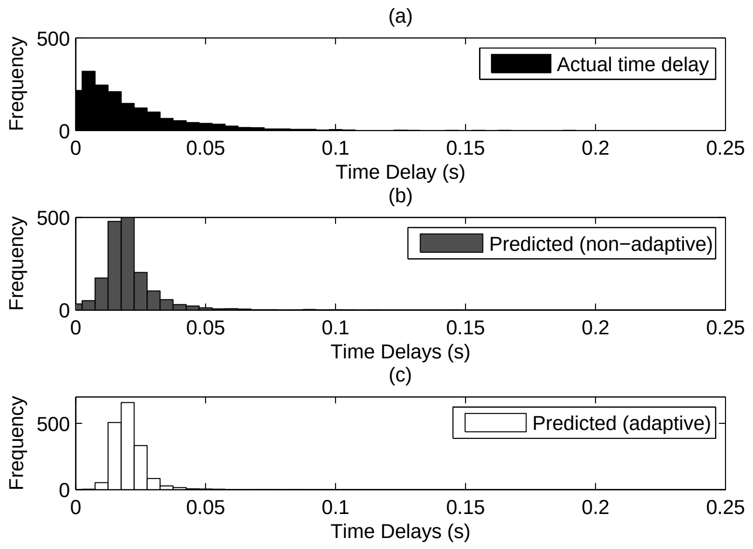

Figure 4.5: Distribution of actual and predicted time delays, $\beta = 0.0$, $NET_{8,6,1}$.

$\beta = 0.0$ given by the $NET_{8,6,1}$ which is one of the TDNN candidates. It can be seen that prediction lacks accuracy, the predicted values are around the mean of the actual time delays series. This behavior occurs in both TDNNs, the non-adaptive and the adaptive one (see Fig. 4.4 (a) and (b), respectively). This can be also observed in the distributions shown by the histograms in Fig. 4.5, where predictions of the non-adaptive (b) and adaptive TDNNs (c) are slightly skewed to the left.

Fig. 4.6 shows the prediction of a sequence with correlation $\beta = 0.25$ given by the $NET_{5,9,1}$. It can be seen that predictions follow the trend of the actual time delay series following more closely the changes of the series than in the previous case. Also this behavior occurs in both TDNNs, the non-adaptive (a) and the adaptive (b). The RMSE for the test has improvement compared with the one obtained in the test for the series with $\beta = 0.00$. The RMSE values of the four sequences are shown in Table 4.8. However, it can be seen in the histograms of Fig. 4.7 that the adaptive predictions (c) have a better approximation to the distribution of the actual time delay shown in Fig. 4.7 (a).

Compared to the previous two cases, we can see in Fig. 4.8 that there is a noticeable change in the behavior of the predicted values for a series with $\beta = 0.50$. It is interesting to note the accuracy of the predictions given by the two TDNN: (a) non-adaptive and (b) adaptive. Both follow very closely the variations of the series of actual values and the performance is improved. Fig. 4.9 shows a better approximation to the distribution of the actual time delay by the non-adaptive (b) and the adaptive TDNN (c). Also we can see in Fig. 4.9 that the distribution

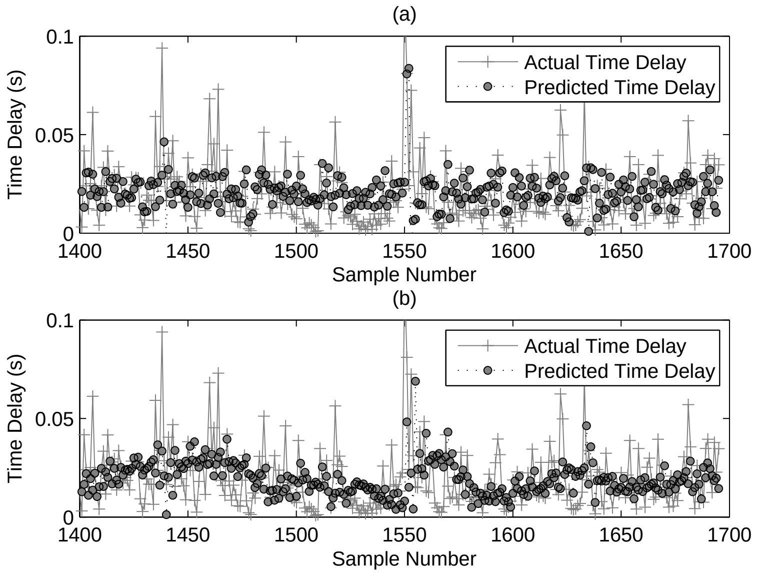

Figure 4.6: Prediction of delay sequence with $\beta = 0.25$ using $NET_{5,9,1}$.

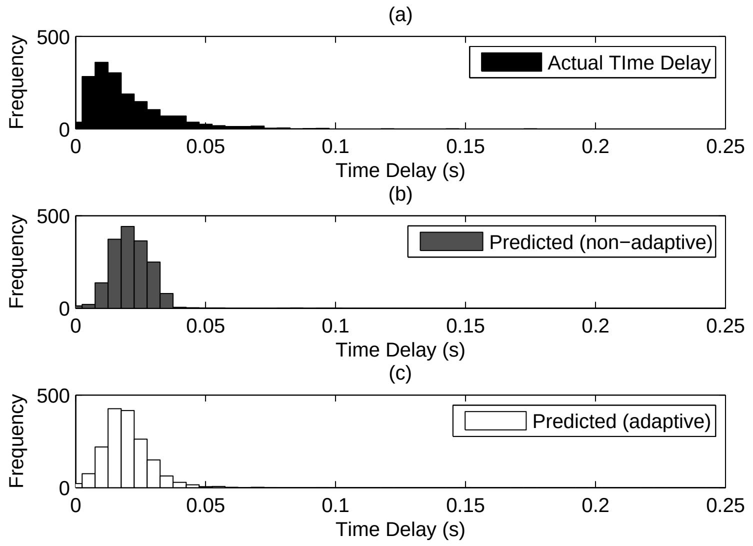

Figure 4.7: Distribution of actual and predicted time delays, $\beta = 0.25$, $NET_{5,9,1}$.

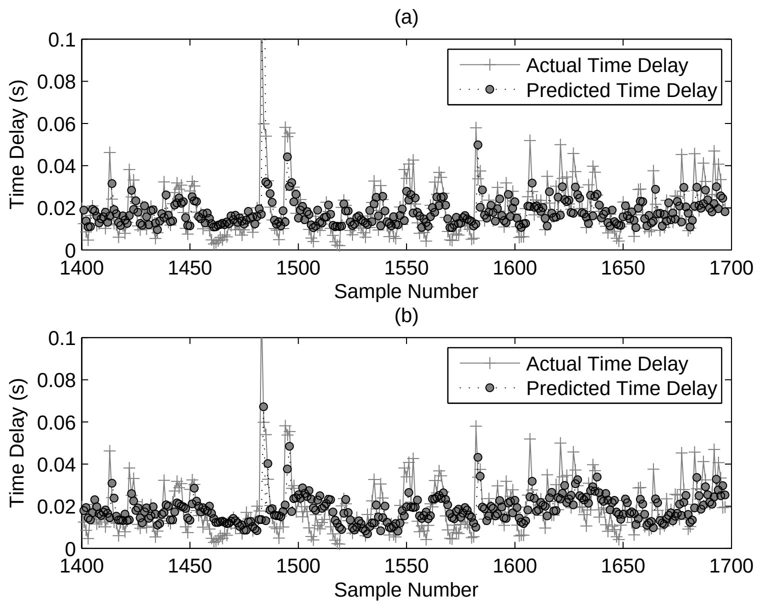

Figure 4.8: Prediction of delay sequence with $\beta = 0.50$ using $NET_{3,6,1}$.

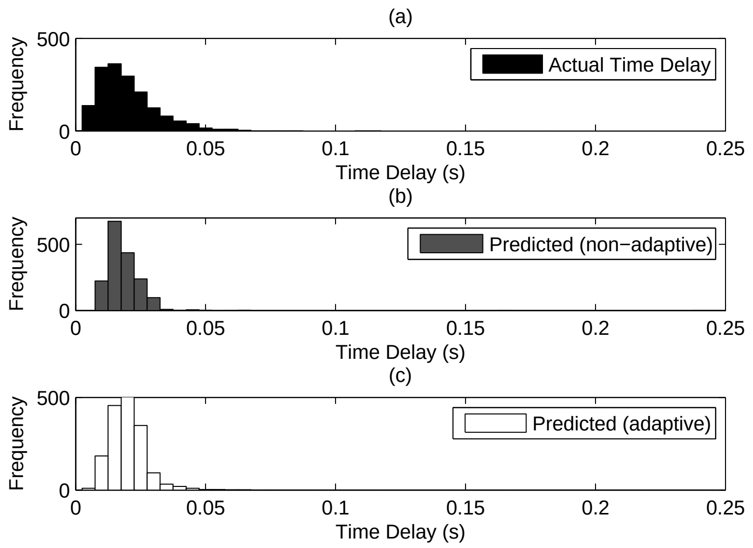

Figure 4.9: Distribution of actual and predicted time delays, $\beta = 0.50$, $NET_{3,6,1}$.

given by the adaptive TDNN (c) has a better approximation to the distribution in (a).

A more important improvement is achieved by the TDNN when the correlation factor is $\beta = 0.75$, see Fig. 4.10. The sequence of predicted vales follows the changes on the series of the actual time delays and the rmse improves even more than the previous cases (see Table 4.8). We can see that both TDNNs, non-adaptive (a) and adaptive (b), follow closely the unknown series of actual delays. In the histograms of Fig. 4.11 we can see that the distribution of predictions given by the adaptive TDNN in (c) almost matches the distribution of the actual time delay in (a). Nevertheless the distribution of the non-adaptive (b) has also a good approximation of the distribution in (a).

The previous outcomes suggest the applicability of adaptive TDNN to the prediction of time delays in a NCS, also the results from the non-adaptive TDNN suggest that it is a good candidate for prediction of time delays when the statistics of the historic series are similar to the training set.

Table 4.8: RMSE prediction performance of different TDNN topologies.

TDNN Topology	β	RMSE non-adaptive	RMSE adaptive
$NET_{8,6,1}$	0.00	0.0242	0.0219
$NET_{5,9,1}$	0.25	0.0169	0.0175
$NET_{3,6,1}$	0.50	0.0125	0.0111
$NET_{4,8,1}$	0.75	0.0052	0.0060

Furthermore, comparing the results of the four sequences in the validation process and the test results given in Table 4.8, we wanted to identify a single general

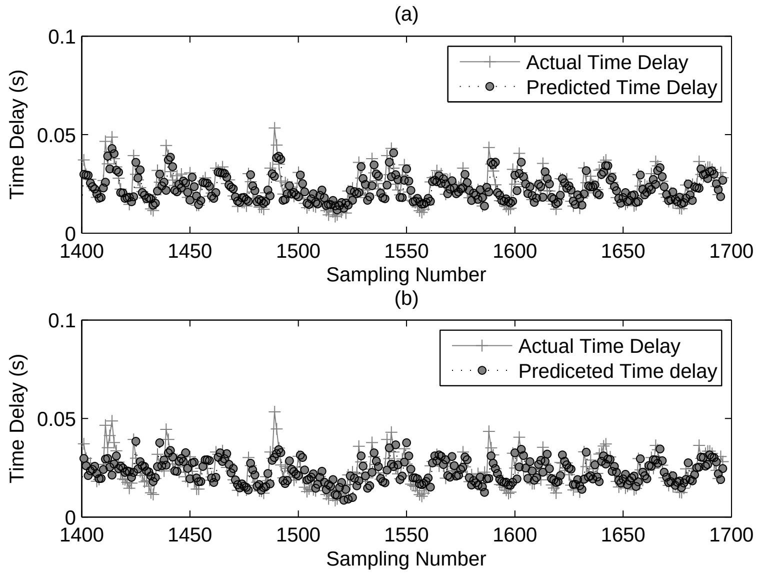

Figure 4.10: Prediction of delay sequence with $\beta = 0.75$ using $NET_{4,8,1}$.

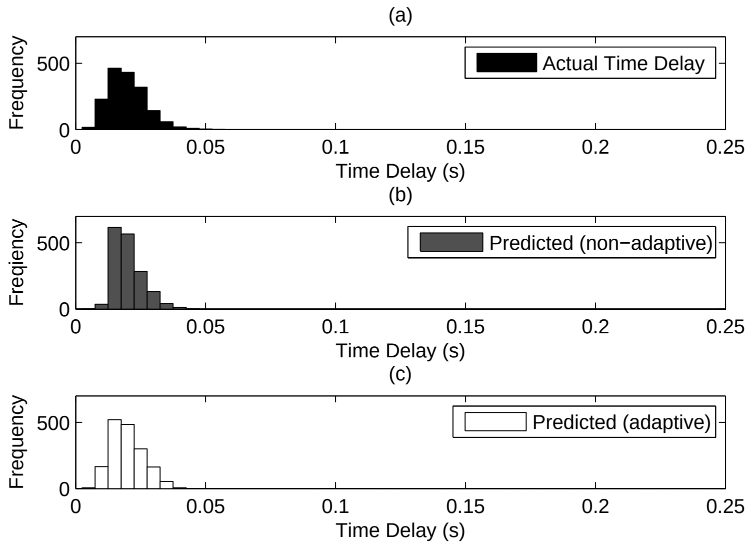

Figure 4.11: Distribution of actual and predicted time delays, $\beta = 0.75$, $NET_{4,8,1}$.

configuration capable of predicting an unknown sequence of delays regardless the correlation factor of it. In Table 4.9 are the results of two candidates for predictors, they were early stopped in the epoch number 10 to avoid overtraining. Based on these results, we decided that the appropriate topology for this kind of sequences with the assumed correlations was: $NET_{5,9,1}$. This will be the topology used in the direct application in the NCS in Chapter 6.

Table 4.9: RMSE prediction performance of different TDNN topologies.

TDNN Topology	β	RMSE non-adaptive	RMSE adaptive	epochs
$NET_{5,9,1}$	0.00	0.0244	0.0238	10
$NET_{5,9,1}$	0.25	0.0161	0.0172	10
$NET_{5,9,1}$	0.50	0.0109	0.0130	10
$NET_{5,9,1}$	0.75	0.0054	0.0056	10
$NET_{8,6,1}$	0.00	0.0238	0.0224	10
$NET_{8,6,1}$	0.25	0.0173	0.0179	10
$NET_{8,6,1}$	0.50	0.0107	0.0122	10
$NET_{8,6,1}$	0.75	0.0055	0.0066	10

We can now compare the previous results of the topology selected $NET_{5,9,1}$, which was obtained by using the systematic trial and error technique combined with the early stopping method, to the topologies given in Table 4.10 obtained using equation (4.33). For a three layer TDNN the number of neurons in the hidden layer was determined using the following values: $E_{goal} = 0.01$, $N_{train} = 300 \times N_{pts}$, and $N_{output} = 1$. We can see that for a TDNN with 5 input nodes we need 2 nodes in the hidden layer (see Table 4.10). Then for a TDNN with topology $NET_{5,2,1}$, the results of the performance on the validation process can be found in Tables 4.4 to 4.7. In addition, we also tested such configuration to investigate its

Table 4.10: Topologies suggested by the rule of thumb.

N_{inputs}	1	2	3	4	5	6	7	8	9	10
Hidden Nodes	1	2	2	2	2	3	3	3	3	3

capability for predicting a time delay series regardless its correlation factor, this was done by applying the same four series used before i.e. the series of 2000 points, mean $\mu = 0.02$ and correlation factors $\beta = 0.00, 0.25, 0.50$ and 0.75. We have the following results:

Table 4.11: RMSE prediction performance of different TDNN topologies.

TDNN Topology	β	RMSE non-adaptive	RMSE adaptive	epochs
$NET_{5,2,1}$	0.00	0.0215	0.0213	10
$NET_{5,2,1}$	0.25	0.0161	0.0174	10
$NET_{5,2,1}$	0.50	0.0105	0.0118	10
$NET_{5,2,1}$	0.75	0.0053	0.0077	10

Comparing the results from $NET_{5,9,1}$ in Table 4.9 and $NET_{5,2,1}$ in Table 4.11 we can see that there are small differences including better numerical performance in some points by $NET_{5,2,1}$. However, a difference exists in the distribution of the predicted data obtained from the two topologies. For instance, comparing results between predictions of the series with $\beta = 0.25$ given by $NET_{5,9,1}$ and $NET_{5,2,1}$, the primary difference is between the two non-adaptive TDNN (see Fig. 4.7 (b) and Fig. 4.13 (b)). The distribution given by the non-adaptive $NET_{5,9,1}$ more closely approximates the distribution of the actual delay. We concluded that $NET_{5,9,1}$ is a better candidate for our research.

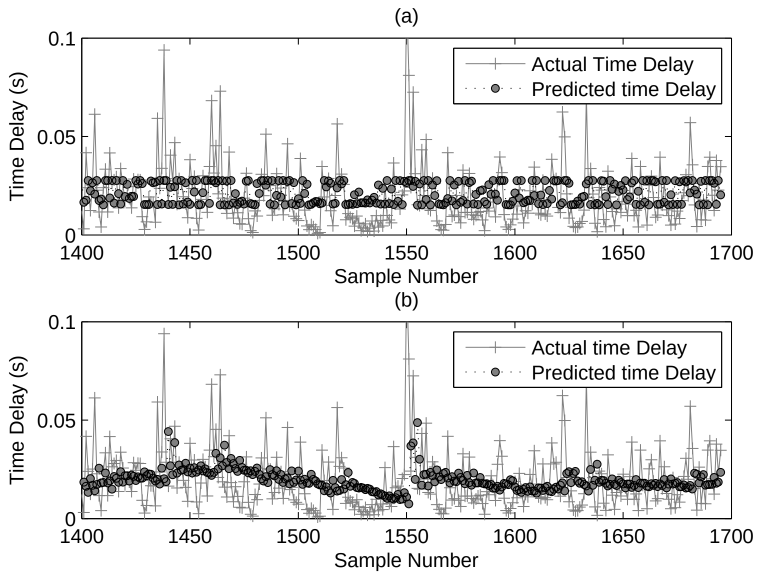

Figure 4.12: Prediction of delay sequence with $\beta = 0.25$ using $NET_{5,2,1}$.

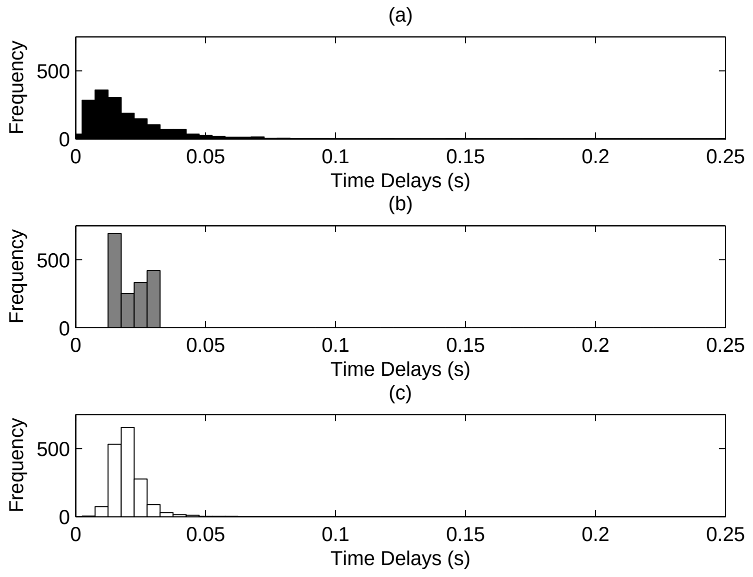

Figure 4.13: Distribution of actual and predicted time delays, $\beta = 0.25$, $NET_{5,2,1}$.

4.3.3 Prediction results with actual data

To further validate the proposed approach, real data of time delays occurring between two computers that have a wireless connection to the internet were acquired. Using the algorithm in Table 4.2, the TDNN topology for a sequences of real time delay data was selected. Based on preliminary results, the maximum number of nodes in the input and hidden layer was predetermined to be 20. The sequence of data was obtained between two computers located 9 km apart. The computer number one was located at Oregon State University, the computer number two was located in a residential place situated 9km away in downtown Philomath also in the state of Oregon. Using the ping function a sequence of 1000 data points was acquired. The ping function measure the round trip time (RTT), then the one-way value of the time delay is equivalent to $RTT/2$. Such one-way time delay sequence is shown in Fig. 4.14 with its corresponding predictions. Each value of the sequence was taken every $10ms$ and the packet transmitted was of 64 bits. The data set was divided into a training set of 300 points and a validation set of 100 points. From the 400 possible topologies obtained (see Table 4.12), the selected topology was $NET_{5,17,1}$. Then, the trained network was then used to predict a sequence of 695 points which gives the results shown in Fig. 4.14, where for the sake of clarity, only the first 300 points of the sequence are presented.

It can be seen in Fig. 4.14 (a) that the TDNN in non-adaptive mode follows closely the variations of the actual time delay sequence. Predictions done by the TDNN in adaptive mode are shown in Fig 4.14 (b), where also it can be seen that

Table 4.12: RMSE validation performance for a series with: $\mu = 0.02$, $\beta = 0.5$.

$i \backslash h$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	0.0215	0.0215	0.0212	0.0212	0.0211	0.0215	0.0211	0.0217	0.0221	0.0219	0.0219	0.0219	0.0225	0.0219	0.0220	0.0222	0.0210	0.0222	0.0219*	0.0222
2	0.0212	0.0213	0.0211*	0.0218	0.0217	0.0212*	0.0218	0.0222	0.0213*	0.0221	0.0226	0.0222	0.0223	0.0238	0.0230	0.0234	0.0241	0.0228	0.0242	0.0235
3	0.0213	0.0211*	0.0213	0.0216	0.0227	0.0222	0.0222	0.0233	0.0214	0.0224	0.0229	0.0231	0.0223	0.0217	0.0229	0.0227	0.0237	0.0224	0.0227	0.0228
4	0.0208*	0.0216	0.0216	0.0220	0.0223	0.0361	0.0224	0.0216	0.0225	0.0236	0.0217	0.0231	0.0221	0.0222	0.0230	0.0220	0.0219	0.0237	0.0235	0.0232
5	0.0216	0.0214	0.0212	0.0211*	0.0210*	0.0226	0.0209	0.0239	0.0234	0.0216	0.0206*	0.0219	0.0240	0.0212*	0.0210*	0.0236	0.0207*	0.0216*	0.0224	0.0212*
6	0.0213	0.0217	0.0226	0.0212	0.0216	0.0218	0.0221	0.0219	0.0220	0.0223	0.0228	0.0238	0.0224	0.0252	0.0229	0.0227	0.0240	0.0244	0.0241	0.0231
7	0.0210	0.0218	0.0216	0.0212	0.0213	0.0220	0.0221	0.0219	0.0232	0.0217	0.0222	0.0233	0.0220*	0.0223	0.0230	0.0224	0.0214	0.0246	0.0238	0.0245
8	0.0211	0.0218	0.0217	0.0223	0.0211	0.0217	0.0209*	0.0215	0.0221	0.0216*	0.0243	0.0225	0.0225	0.0219	0.0238	0.0230	0.0293	0.0324	0.0221	0.0261
9	0.0218	0.0213	0.0217	0.0214	0.0227	0.0215	0.0212	0.0227	0.0249	0.0230	0.0238	0.0224	0.0226	0.0230	0.0233	0.0213*	0.0268	0.0274	0.0231	0.0238
10	0.0218	0.0216	0.0221	0.0222	0.0220	0.0220	0.0223	0.0219	0.0221	0.0226	0.0253	0.0236	0.0227	0.0216	0.0235	0.0255	0.0253	0.0274	0.0234	0.0271
11	0.0216	0.0215	0.0216	0.0224	0.0221	0.0227	0.0228	0.0214*	0.0224	0.0232	0.0229	0.0269	0.0251	0.0272	0.0247	0.0219	0.0248	0.0230	0.0246	0.0256
12	0.0216	0.0219	0.0220	0.0223	0.0228	0.0219	0.0216	0.0223	0.0226	0.0282	0.0220	0.0256	0.0266	0.0260	0.0243	0.0220	0.0253	0.0224	0.0256	0.0294
13	0.0221	0.0238	0.0229	0.0220	0.0228	0.0281	0.0253	0.0220	0.0224	0.0261	0.0254	0.0236	0.0223	0.0269	0.0288	0.0227	0.0264	0.0282	0.0250	0.0268
14	0.0219	0.0226	0.0258	0.0216	0.0237	0.0227	0.0249	0.0227	0.0237	0.0244	0.0235	0.0222	0.0260	0.0232	0.0234	0.0272	0.0241	0.0266	0.0237	0.0265
15	0.0220	0.0228	0.0220	0.0218	0.0253	0.0228	0.0236	0.0226	0.0245	0.0270	0.0224	0.0253	0.0293	0.0255	0.0254	0.0286	0.0246	0.0311	0.0291	0.0261
16	0.0218	0.0217	0.0216	0.0235	0.0217	0.0223	0.0279	0.0222	0.0225	0.0242	0.0246	0.0219	0.0266	0.0271	0.0276	0.0257	0.0226	0.0226	0.0287	0.0249
17	0.0218	0.0225	0.0239	0.0220	0.0243	0.0221	0.0213	0.0239	0.0273	0.0227	0.0254	0.0245	0.0292	0.0354	0.0226	0.0279	0.0228	0.0301	0.0241	0.0250
18	0.0223	0.0218	0.0219	0.0231	0.0219	0.0229	0.0231	0.0240	0.0229	0.0254	0.0264	0.0218	0.0242	0.0278	0.0297	0.0268	0.0219	0.0315	0.0227	0.0228
19	0.0219	0.0235	0.0220	0.0245	0.0225	0.0222	0.0268	0.0244	0.0243	0.0243	0.0277	0.0215*	0.0237	0.0272	0.0267	0.0221	0.0316	0.0349	0.0252	0.0231
20	0.0226	0.0222	0.0228	0.0233	0.0248	0.0230	0.0250	0.0255	0.0237	0.0221	0.0273	0.0279	0.0243	0.0278	0.0278	0.0854	0.0235	0.0218	0.0273	0.0259

i = nodes in the tapped delay line, h = nodes in the hidden layer.

predictions follow the changes of the sequence of actual delays. From these results, it can be concluded that the algorithm proposed for the selection of the TDNN can be used effectively to select an adequate topology of the TDNN, given a sequence of historic data of the delays occurring in a particular network.

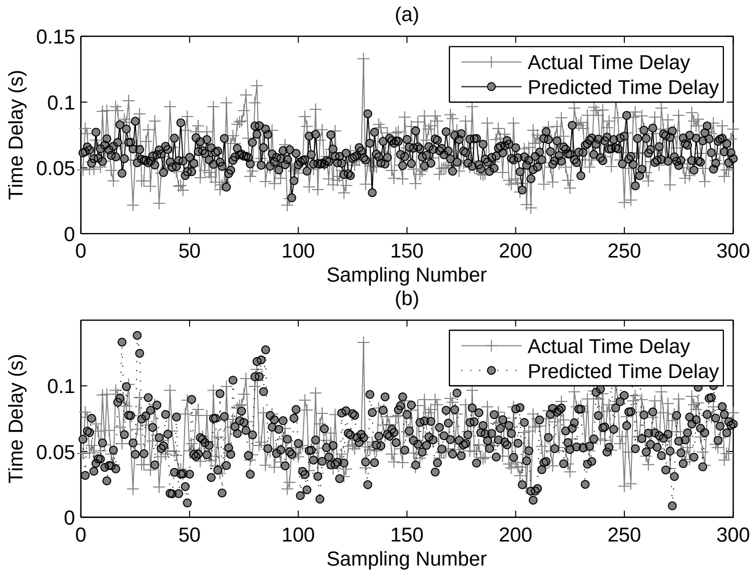

Figure 4.14: Prediction of delay sequence using a $NET_{5,17,1}$.

4.3.4 Multi-step prediction

As we will see in Chapter 5, multi-step predictions will be needed for the cases of packet loss in a NCS, this means that the TDNN will need to predict the value of delays more than one step ahead. We know that packets that go from sensor to controller have the information of the state of the plant, the last delay from controller to actuator τ_{ca}, and also the time stamp of the packet that allows the controller to know the delay from sensor to controller τ_{sc}.

As mentioned earlier once the predictive TDNN is trained off-line, its parameters remain fixed and the TDNN can be used in the NCS as a non-adaptive TDNN. In the case of the predictive adaptive TDNN, its parameters need to be adjusted at every sample step. In order to make such adjustments, the TDNN has to make a comparison of its current output (predicted delay value) to the actual corresponding time delay, i.e. it needs to calculate the prediction error. To do that, it is necessary to have continuous and uninterrupted arrival (to the predictive TDNN) of the information regarding the delays included in the packets traversing the NCS. In reality in a NCS, packets can be lost in the communication link and we set up an strategy to compensate for this losses using multi-step prediction.

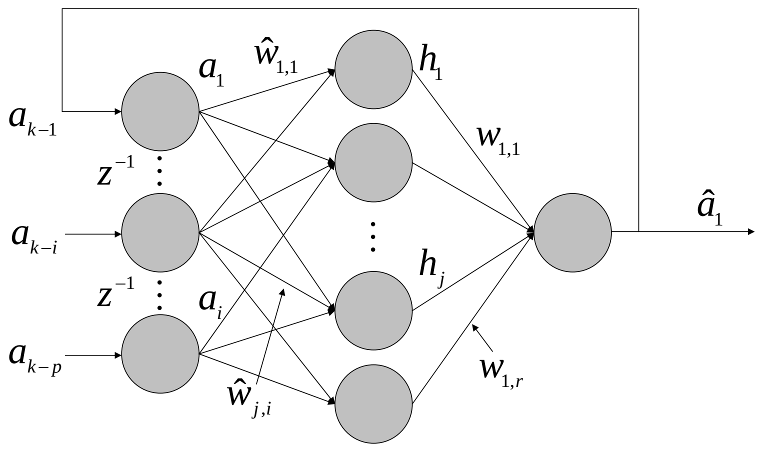

Figure 4.15: TDNN for multi-step prediction.

As mentioned before, the information of delays (τ_{ca}) is used to keep the history of delays needed by the TDNN to predict in a one-step ahead fashion. However, in the case that a packet is lost or dropped, the sequence of delays will present gaps that need to be filled by an approximation or estimation of the missing data. In order to fill those possible gaps we implemented a multi-step prediction configuration of the TDNN. We tested the TDNN with the topology determined in the previous section to do multi-step predictions by feeding the last predicted value back to the tapped delay line as the new input as shown in Figure 4.15. This process is repeated until the next packet containing the information of delays arrives to the controller. A time out policy will trigger the multi-step prediction and such policy is explained in detail in chapter 5. For the moment it is only needed to say that if a delay is greater than a certain threshold, the packet is considered lost and multi-step prediction is initiated. This process continuous until the next packet arrives and a new value of τ_{ca} is available for prediction error computation.

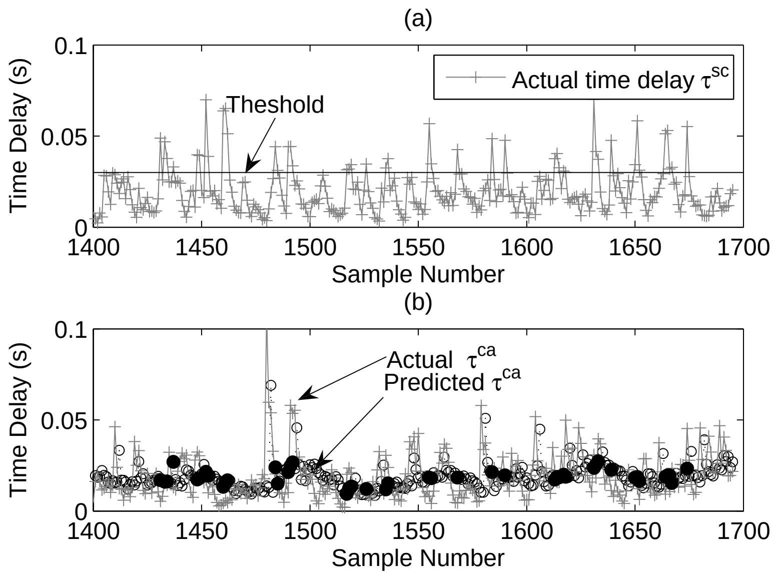

Figure 4.16: Adaptive TDNN multi-step prediction.

To test this proposed scheme, we implemented two predictive TDNNs of the form $NET_{5,9,1}$ previously studied, one of them was non-adaptive and the other one was adaptive. We assumed for our simulations that in a NCS the sequence of delays from sensor to controller (τ^{sc}) and from controller to actuators (τ^{ca}) have the parameters $\beta = 0.50$, mean $\mu = 0.02$.

We assumed also that the threshold value at which a packet coming from the sensors to the controller is considered to be lost is equal to 0.03s, see Fig. 4.16 (a) and 4.17 (a). We will also see in chapter 5 that in our NCS model, predictions are only needed to be carried out for the controller-to-actuator side. In this context we know that the predictor will have some gaps in the history of τ^{ca}. Recall that the packets from sensor to controller contain the information of the last value of τ^{ca}.

For this scenario in our simulations using both the non-adaptive and the adaptive TDNNs, a multi-step prediction was triggered every time a packet was considered lost. The black dots in figures 4.16 (b) and 4.17 (b), indicate the delay values predicted beyond one step ahead. In the same figures, multiple predictions were done for consecutive values of delays above the threshold line. This implies that the TDNN predictor had to send its output back to its input layer to generate a new estimated value of the delay τ^{ca} continuously for some steps.

Table 4.13: RMSE performance from multi-step prediction.

TDNN Topology	β	RMSE non-adaptive	RMSE adaptive	epochs
$NET_{5,9,1}$	0.50	0.0132	0.0130	10

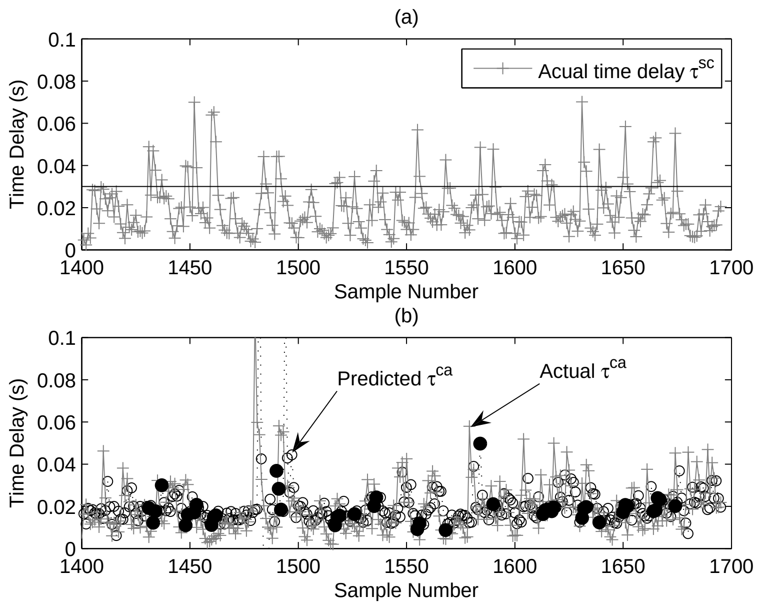

Figure 4.17: Non-adaptive TDNN multi-step prediction.

From the results shown in figures 4.16 and 4.17, it can be concluded that multi-step prediction can be done and used for the compensation of the delays occurring in a NCS. They have the corresponding performance values indicated in Table 4.13.

We observed in our simulations that multi-step prediction can not be done for an indefinite amount of steps, it is necessary to have some feedback to calculate the prediction errors for the case of the adaptive TDNN in order to modify its parameters (weights). It was observed also that using this methodology the prediction converges to the mean of the time delay series. However, once the TDNN starts receiving again actual information of delays, it will generate more accurate predictions and it will follow the changes of the time delay sequences. This behav-

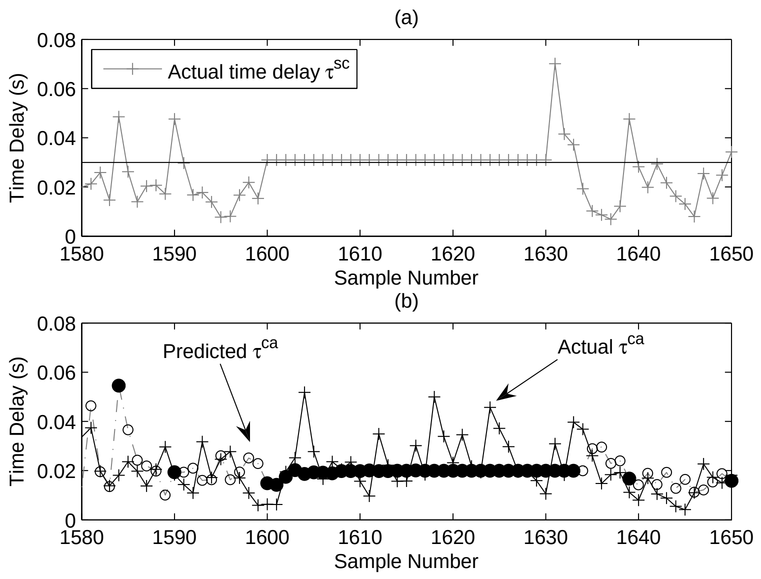

Figure 4.18: Non-adaptive TDNN multi-step prediction

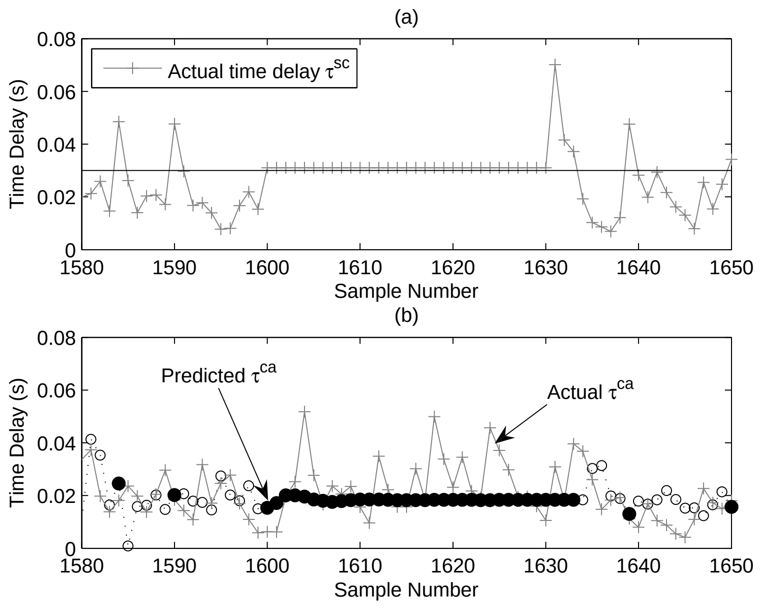

Figure 4.19: Adaptive TDNN multi-step prediction

ior can be seen in figures 4.18 and 4.19. In these cases we allowed the delays to be above the threshold in an arbitrary section of the sequence from sample step number 1600 to 1633. It is clear in Fig. 4.18 (b) that multiple predictions follow the changes of the actual values for the first 5 samples, i.e. from 1600 to 1605. After this point the predicted values converge to the mean of the sequence. Similar result is also seen in Fig. 4.19 (b).

4.4 Conclusion

From the results given above it can be seen that the trial and error method requires several iterations. However, it is important to mention that if we have good information regarding the time series to be predicted, the procedure of searching for the most adequate topology of a TDNN, can be done in an iterative manner. More importantly is to do a correct selection process, this includes to validate each proposed topology and also test them. This leads to have enough results to decide on the most appropriate topology of the TDNN for a particular application. After creating and testing several TDNN topologies we conclude that the TDNN with 5 delays in the tapped delay line, 9 nodes in the hidden layers and one output is the most adequate given the delay sequences used in this chapter.

It was also shown that the configuration for multi-step predictions is feasible, nevertheless such configuration has a limited prediction horizon.We can see that multi-step prediction gives with similar RMSE as in the case of no missing data. The results given by these simulations are one part of the core in our research, and we will see in chapter 5 the reason why the predictive TDNNs will predict only the delay from controller to actuator τ^{ca} even though they can predict also τ^{sc} in the same fashion.

From the simulations we can also conclude that predictions become more accurate when the correlation increases. This is important for this research since we are assuming that delays are correlated with previous delays.

The selected topology for prediction described above, was used on the numerical simulations presented in Chapter 6 for the proposed NCS models described in Chapter 5.

Chapter 5 – Networked Control System Model Design

As mentioned earlier the transmission delays depend on several factors like the protocol used on the network, traffic patterns, the channel conditions and congestion during transmission. Furthermore, delays can be of different nature either constant, variable (though deterministic) or random. If we take into consideration that a long time delay can cause instability, we can think of the existence of a limit in the duration of the delay for which the controlled system is stable. In this chapter we explain the design of a timeout policy for the case of long delays and a design of the variable sampling and observer models is presented. A controller design methodology is presented as well. We then describe and design a NCS model for the case when a NCS is subject to packet loss, in addition to the already existent time delay in the communication loop. At the end of this chapter we comment on some suggested or alternative approaches for the cases of packet loss.

5.1 Time delays and timeout policy

The following are the considerations taken regarding the duration of time delays in both ends of the system loop, i.e. τ^{sc} and τ^{ca}. For a controlled system subject to time delays in the feedback communication loop, there exists a maximum time delay, T_{max}, allowable for which the stability of the NCS is guaranteed to be preserved [33]. In other words if, for a particular controlled system, at sampling step k the total time delay τ_k is smaller than the maximum time delay T_{max}, then the system is stable. This means that for the cases when $\tau_k > T_{max}$, given the combination of $\tau_k^{sc} + \tau_k^{ca} > T_{max}$, the system may become unstable due to packets dropout. We are assuming that a packet will be discarded if it has not been successfully received by the end of the sample period equal to T_{max}, since new measurements are always more valuable than old measurements [25]. Remember that the idea of this work is to minimize the effects on the NCS caused by time delays and packet loss.

Let us consider the possibility of having the knowledge of how long a time delay is gong to be, (for a packet to arrive to its destination), before it occurs. Under this assumption, it is logical to think that we can reduce the uncertainty of the delays, at least from one side of the network loop, by inserting time stamp information in packets from sensor to controller. This means that τ_k^{sc} is known once it arrives at the controller which we are assuming is event driven. However, the delay values from controller to actuator τ_k^{ca} remain uncertain since they have not happened yet. Knowing the value of delay τ_k^{ca} or an estimate of it, before it occurs, will allow the controller to compensate for the total time delay τ_k. It is here where the proposed predictive TDNN described in previous chapters is used to predict such unknown delays. Remember that the TDNN can be used in two different ways, non-adaptive mode, which, as we saw earlier, means that the parameters of the network will change only during the training stage, until a predetermined prediction error is achieved. The other way of prediction is using the adaptive TDNN , where its parameters will change in a continuous fashion. Then, the TDNN can be used to output a predicted value of the delays $\hat{\tau}_k^{sc}$, $\hat{\tau}_k^{ca}$ or $\hat{\tau}_k$.

In the case of a delayed packet coming from the sensors, which contains the information of the system output $y(t_k)$ and the corresponding time stamp, exceeds the maximum allowed time delay to arrive to the controller, a timeout policy has to be adopted. We name the time out from sensor to controller as t_{out}^{sc}. Similarly, in the actuator side we have t_{out}^{ca}. Both are defined as

$$t_{out}^{sc} = t_{out}^{ca} = \frac{T_{max}}{2} - \tau_c \tag{5.1}$$

where τ_c is the computational time required by the controller to output a control signal, and is assumed to be constant.

In the following sections we consider different scenarios regarding the time out policy for time delays. They include the case when $\tau_k^{sc} > T_{max}/2$ and the lack of the information packet containing $\hat{y}(t_k)$ and τ_{k-1}^{ca}. We investigate for this case the application of multi-step prediction by the TDNN. The case when $\tau_k^{ca} > T_{max}/2$ is also investigated and some approaches to compensate for it including the use of multi-step prediction as well and the implementation of a hold system before the actuators node are proposed. We also point out that it is possible to compute control values in advance and send them in a single packet to the actuators. This approach has the drawback that it will increase the load of the network.

5.2 Variable sampling and observer models

Let a continuous-time linear time-invariant controlled system model with no delays have the state-space form

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t),\ x(t_0) \\ y(t) &= Cx(t) \end{aligned} \tag{5.2}$$

with feedback controller

$$u(t) = -Kx(t)$$

where $x(t) \in \mathbb{R}^n$, $u(t) \in \mathbb{R}^m$, $y(t) \in \mathbb{R}^p$, represent the system states, input, output, respectively. In this case A, B, and C are constant matrices of appropriate dimensions and K is the $m \times n$ feedback gain matrix.

In NCS there are two sources of transmission delays in the communication network. Namely, delays from sensor to controller τ_k^{sc} and from controller to actuators τ_k^{ca}. If system (5.2) is subject to such delays, the controlled system can be modeled by

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t - \tau_k^{sc} + \tau_k^{ca}),\ x(t_0) \\ y(t) &= Cx(t) \end{aligned} \tag{5.3}$$

Also, both delays can be lumped together as the total time delay $\tau_k = \tau_k^{sc} + \tau_k^{ca}$, where it has been assumed that the computational time required for the controller is negligible.

Sampling a system implies to chose the sampling instants, t_k, as the times when the control signal changes. Then it is logical to think that the duration between samples (sampling period) can be constant or variable.

Consider, for the moment, the assumption that system (5.2) is subject to known delays (the case with unknown random delays is studied later). If the actuators and controller are assumed event-driven, then the control input will change at every time t_k. Furthermore, the sampling period T_k is considered to be equal to the total time delay τ_k, i.e. $T_k = \tau_k$, which can be variable. Then the system model (5.3) becomes a discrete time-variant controlled system described by

$$\begin{aligned} x(t_{k+1}) &= \Phi(T_k)x(t_k) + \Gamma(T_k)u(t_k) \\ y(t_k) &= Cx(t_k) \end{aligned} \tag{5.4}$$

with discrete feedback controller

$$u(t_k) = -K(T_k)x(t_k) \tag{5.5}$$

where

$$\begin{aligned} \Phi(T_k) &= e^{AT_k} \\ \Gamma(T_k) &= \int_{t_k}^{t_{k+1}} e^{A(t_{k+1}-\lambda)}Bd\lambda\ , \end{aligned}$$

assuming $u(t_k)$ remains the same for $t_k \leq t \leq t_{k+1}$. Note that under the previous assumptions the system (5.4) is not subject to delays. Now, let us assume that the state x can be approximated by

$$\hat{x}(t_{k+1}) = \Phi(T_k)\hat{x}(t_k) + \Gamma(T_k)u(t_k) \tag{5.6}$$

which has the same input as system (5.4). If the model (5.6) is perfect in the sense that its parameters are identical to the system in (5.4) and also with the same initial conditions, then states $\hat{x}$ will be identical to those in (5.4), i.e. $\hat{x} = x$.

This is true for any sampling period applied to (5.4) either constant or variable. Using (5.6) we can generate an observer of (5.4) by introducing the difference between the measured and estimated outputs, i.e. $y(t_k) - \hat{y}(t_k)$, where

$$\hat{y}_k = C\hat{x}(t_k) \tag{5.7}$$

Then (5.6) becomes

$$\hat{x}(t_{k+1}) = \Phi(T_k)\hat{x}(t_k) + \Gamma(T_k)u(t_k) + L(T_k)(y(t_k) - \hat{y}(t_k)) \tag{5.8}$$

where $L(T_k)$ is the observer gain matrix yet to be defined.

Up to this point we have assumed that for a known sampling period T_k the states given by system 5.4 and 5.6 are exactly the same. However, in the real case of having a NCS over a a communication network, the existence of unknown time delays which, as mentioned before, can be constant, variable, or even random makes it more complicated to do the sampling of (5.4) and (5.6) equally.

We now propose an observer and a controller based on those unknown time delays. In order to do that, it is necessary to know exactly when to perform the sampling. In other words, the estimator and controller need to know at sampling step k the sampling period T_k that will occur, and since $T_k = \tau_k$ it is necessary to predict the value of τ_k.

To solve the prediction problem, we apply a predictive adaptive TDNN to output a predicted value, at sampling step k, of the total delay τ_k, i.e. the TDNN will predict $\bar{\tau}_k$ which becomes the parameter $\bar{T}_k$. Such value will be used to compute $\bar{\Phi}(\bar{T}_k)$ and $\bar{\Gamma}(\bar{T}_k)$ which are to be substituted in (5.6). This also implies that the observer gain L in (5.8) has to be estimated also based on $\bar{T}_k$ giving the estimated observer gain denoted by $\bar{L}(\bar{T}_k)$. The entire proposed system is depicted in Fig. 5.1, where the dashed line represents the network connection between estimator-controller and the true system.

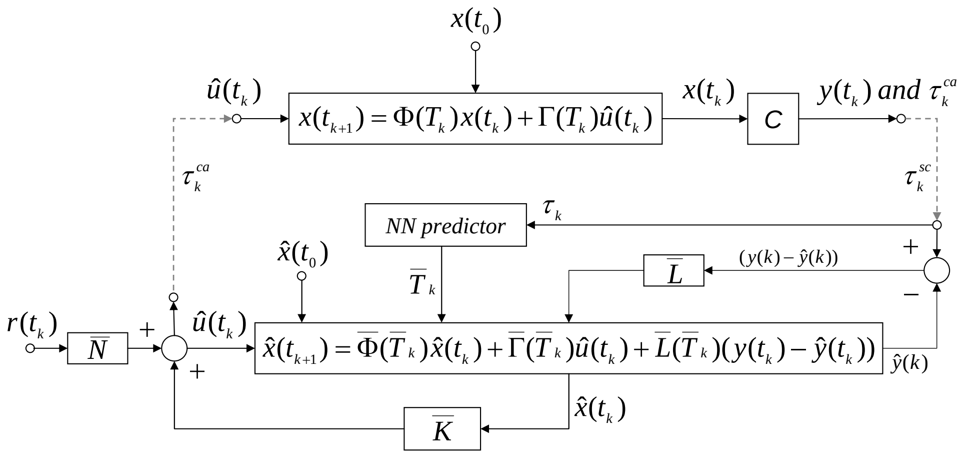
$x(t_0)$
$\hat{u}(t_k)$
$x(t_{k+1}) = \Phi(T_k)x(t_k) + \Gamma(T_k)\hat{u}(t_k)$
$x(t_k)$
C
$y(t_k)$ and τ_k^{ca}
τ_k^{ca}
τ_k^{sc}
NN predictor
τ_k
$\hat{x}(t_0)$
$\bar{L}$
$(y(k)-\hat{y}(k))$
$+$
$-$
$\bar{T}_k$
$r(t_k)$
$\bar{N}$
$+$
$+$
$\hat{u}(t_k)$
$\hat{x}(t_{k+1}) = \bar{\Phi}(\bar{T}_k)\hat{x}(t_k) + \bar{\Gamma}(\bar{T}_k)\hat{u}(t_k) + \bar{L}(\bar{T}_k)(y(t_k) - \hat{y}(t_k))$
$\hat{y}(k)$
$\bar{K}$
$\hat{x}(t_k)$

Figure 5.1: Proposed NCS based on a predictor and an observer.

Furthermore, the control feedback $u(t_k)$ will be indeed an estimated value based on the states estimated by the observer, this will be denoted by

$$\hat{u}(t_k) = -\bar{K}(\bar{T}_k)\hat{x}(t_k) \tag{5.9}$$

where $\bar{K}$ is a suitable estimated feedback control gain based on $\bar{T}_k$ as well. Then the observer in (5.8) becomes

$$\hat{x}(t_{k+1}) = \bar{\Phi}(\bar{T}_k)\hat{x}(t_k) + \bar{\Gamma}(\bar{T}_k)\hat{u}(t_k) + \bar{L}(\bar{T}_k)(y(t_k) - \hat{y}(t_k)) \tag{5.10}$$

Remember that if $\bar{T}_k = T_k$ the states given by (5.10) are identical to the states x given by the true system. Then the closed loop system can be expressed as

$$\begin{cases} x(t_{k+1}) & = \Phi(T_k)x(t_k) + \Gamma(T_k)\hat{u}(t_k) \\ \hat{x}(t_{k+1}) & = \bar{\Phi}(\bar{T}_k)\hat{x}(t_k) + \bar{\Gamma}(\bar{T}_k)\hat{u}(t_k) + \bar{L}(\bar{T}_k)(y(t_k) - \hat{y}(t_k)) \\ y(t_k) & = Cx(t_k) \\ \hat{y}(t_k) & = C\hat{x}(t_k) \\ \hat{u}(t_k) & = -\bar{K}(\bar{T}_k)\hat{x}(t_k). \end{cases} \tag{5.11}$$

5.3 Determining $\bar{K}$ and $\bar{L}$

The selection of the control gain $\bar{K}$ is based on optimal control given by the linear quadratic regulator technique or LQR for discrete systems. The optimal control gain $\bar{K}$ is computed for the estimated system model

$$\hat{x}(t_{k+1}) = \bar{\Phi}(\bar{T}_k)\hat{x}(t_k) + \bar{\Gamma}(\bar{T}_k)u(t_k) \tag{5.12}$$

under the assumption that $\bar{T}_k = T_k$, which is the ideal case of exact prediction of delays. We solve the optimal discrete linear quadratic regulator problem (DLQR) for

(5.12), such that the state feedback law $\hat{u}(t_k) = -\bar{K}\hat{x}(t_k)$ minimizes the quadratic cost function

$$J = \sum_{k=0}^{N-1} \hat{x}_k' Q \hat{x}_k + \hat{u}_k' R \hat{u}_k$$

where $\bar{K} = (R + \bar{\Gamma}'\Pi\bar{\Gamma})^{-1}(\bar{\Gamma}'\Pi\bar{\Phi})$ with Π being the solution to the discrete-time algebraic Riccati equation

$$\Pi = (Q + \bar{\Phi}'\Pi\bar{\Phi}) - (\bar{\Gamma}'\Pi\bar{\Phi})'(R + \bar{\Gamma}'\Pi\bar{\Gamma})^{-1}(\bar{\Gamma}'\Pi\bar{\Phi})$$

and Q and R are the weights for the states and control, respectively. For conciseness in the notation we have omitted the term $(\bar{T}_k)$ in $\bar{\Phi}$ and $\bar{\Gamma}$. The observer gain $\bar{L}$ is calculated based on the Ackerman method for pole placement. Note that the computation of $\bar{K}$ and $\bar{L}$ is done for every predicted time delay $\bar{T}_k$.

5.4 Treatment of packet loss

As we mentioned before one of the issues that a NCS faces is the effect of packet loss due to network congestion. This can be seen in terms of the length of the time delay that might occur in the transmission from sensor to controller, from controller to actuators or a combination of both. The following sections discuss the control algorithms for the cases where no packets are lost and where packets are lost.

5.4.1 NCS subject to delays and with no packet loss

Based on the system in (5.11) that describes the NCS in Fig. 5.1 we can obtain the closed loop structure for the overall system model subject to time delays and no packet loss, i.e. y_k arrives at the controller and $\hat{u}_k$ arrives at the plant actuators. The system is given in (5.13)

$$\begin{bmatrix} x(t_{k+1}) \\ \hat{x}(t_{k+1}) \end{bmatrix} = \begin{bmatrix} \Phi(T_k) & -\Gamma(T_k)\bar{K}(\bar{T}_k) \\ \bar{L}(\bar{T}_k)C & \bar{\Phi}(\bar{T}_k) - \bar{\Gamma}(\bar{T}_k)\bar{K}(\bar{T}_k) - \bar{L}(\bar{T}_k)C \end{bmatrix} \begin{bmatrix} x(t_k) \\ \hat{x}(t_k) \end{bmatrix} \tag{5.13}$$

where it is assumed that the total time delay in the control loop is less than the maximum delay, i.e. $\tau_k < T_{max}$.

5.4.2 NCS model subject to packet loss and time delays

We have seen that the information or packets in a NCS are subject to time delays. We can also think that some of those packets will not reach their final destination, namely, they may get dropped as they traverse the nodes of the communication network. This loss can happen to packets as they travel from both sensor to controller and from controller to actuator.

We defined our models based on the system depicted in Fig. 5.2, which adds some new parameters to our model in Fig. 5.1 and they are described in the remainder of this section. We define the system model subject to packet loss and time delays in two ways. The first model adopts a holding device (queue)

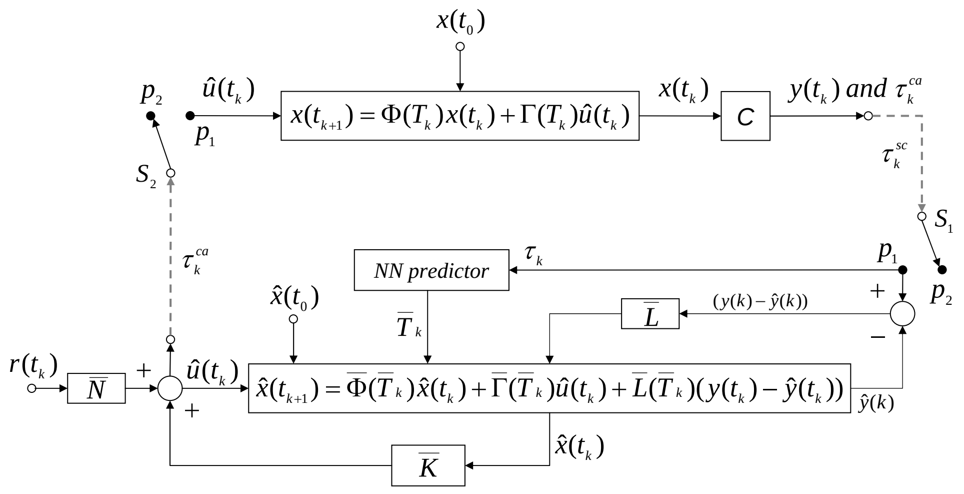

Figure 5.2: Representation of a NCS with packet loss.

before the controller and another before the actuators. The idea is to use the most recent value of both, the system output and the control signal in the case that the corresponding packet at step k is lost. These values are updated when a new value arrives. This approach is also combined with a TDNN predictor which, as we will see, may need to perform multi-step predictions. The second model relies only on the observer combined with multi-step predictions of time delays. A detailed explanation of these approaches follow.

As stated before, in the case of packet loss the NCS is subject to the maximum time delay that occur in the network i.e. T_{max}. Then the timeout policy for T_{max} previously formulated is applied in the following approaches.

Our first approach treats the packet loss as a system where its output information $y(t_k)$ does not arrive to the observer/controller before the time $T_{max}/2$ is up. In this case we propose that the observer uses the previous value of the system output, i.e. $y(t_{k-1})$. This can be expressed as

$$\begin{aligned} p_1 &: y(t_k) = y(t_k) \\ p_2 &: y(t_k) = y(t_{k-1}) \end{aligned} \tag{5.14}$$

where p_n is the position of switches S_1 and S_2 in Fig. 5.2. It is assumed that the position of the switches changes from p_1 to p_2 with some probability. The observer calculates the estimation of the states and a control output is generated.

Since the controller is event driven then, in the case of packet loss from sensor to controller, the sampling time at the corresponding sampling step k, is equal to the event defined as the value $T_{max}/2 + \bar{\tau}_k^{ca}$, which allows the observer to estimate the states and calculate the required control. The parameter $\bar{\tau}_k^{ca}$ is defined later. We also know that a packet coming from sensors to controller contains the information of the last delay from controller to actuator, i.e. τ_{k-1}^{ca}. If the TDNN does not have that information, then it performs a prediction further than one step ahead to output the corresponding predicted delay from controller to actuator $\bar{\tau}_k^{ca}$ at the corresponding sampling step k. The observer then uses $\bar{\tau}_k^{ca}$ and computes the estimation of the states and a control input $\hat{u}(t_k)$ is generated. Multi step prediction is explained in chapter 5.

Furthermore, to implement this methodology in the case when packet loss occurs from controller to actuator, i.e. S_2 is in position p_2, which can not be foreseen by the time delay predictor, it is necessary to have a way to hold the most recent control value that has already arrived at the actuators. We are assuming that such storage system exists and outputs the most recent control input, i.e. $\hat{u}(t_{k-1})$

exactly at time $T_{max}/2$. We are assuming that this action continues until a new value arrives. Then the values in the storage system are replaced by new ones.

Considering all possible scenarios for the combination of estimator and the actual system, the NCS has the structure given next.

$$\begin{cases} x(t_{k+1}) & = \Phi(T_k)x(t_k) + \Gamma(T_k)\hat{u}(t_k) \ if \ \hat{u}(t_k) \ is \ delivered \\ x(t_{k+1}) & = \Phi(T_k)x(t_k) + \Gamma(T_k)\hat{u}(t_{k-1}) \ if \ \hat{u}(t_k) \ is \ lost \\ \hat{x}(t_{k+1}) & = \bar{\Phi}(\bar{T}_k)\hat{x}(t_k) + \bar{\Gamma}(\bar{T}_k)\hat{u}(t_k) + \bar{L}(\bar{T}_k)(y(t_k) - \hat{y}(t_k)) \ if \ y(t_k) \ is \ delivered \\ \hat{x}(t_{k+1}) & = \bar{\Phi}(\bar{T}_k^m)\hat{x}(t_k) + \bar{\Gamma}(\bar{T}_k^m)\hat{u}(t_k) + \bar{L}(\bar{T}_k^m)(y(t_{k-1}) - \hat{y}(t_k)) \ if \ y(t_k) \ is \ lost \\ y(t_k) & = Cx(t_k) \\ \hat{y}(t_k) & = C\hat{x}(t_k) \\ \hat{u}(t_k) & = -\bar{K}(\bar{T}_k)\hat{x}(t_k). \end{cases} \tag{5.15}$$

where $\bar{T}_k^m$ is the predicted sampling period at sampling step k given a multi-step prediction. We incorporated in the notation the super-script m, that is, the number of step ahead predictions, it is omitted in the notation when $m = 1$ which is the basic one step ahead prediction. Some modifications to this first approach are made to present a second approach to compensate effects of the NCS subject to time delays and packet loss.

The second approach to compensate for the packet loss proposes to use the observer to estimate the state based only on the prediction of the time delay further i.e. a prediction of more than one step ahead. Recall that if predictions were accurate, the states of the actual system and the states of the observer would be identical. We design this alternative approach based on this previous observation. The proposed combination of estimator and the actual system has the following scenarios:

$$\begin{cases} x(t_{k+1}) & = \Phi(T_k)x(t_k) + \Gamma(T_k)\hat{u}(t_k) \; if \; \hat{u}(t_k) \; is \; delivered \\ x(t_{k+1}) & = \Phi(T_k)x(t_k) + \Gamma(T_k)\hat{u}(t_{k-1}) \; if \; \hat{u}(t_k) \; is \; lost \\ \hat{x}(t_{k+1}) & = \bar{\Phi}(\bar{T}_k)\hat{x}(t_k) + \bar{\Gamma}(\bar{T}_k)\hat{u}(t_k) + \bar{L}(\bar{T}_k)(y(t_k) - \hat{y}(t_k)) \; if \; y(t_k) \; is \; delivered \\ \hat{x}(t_{k+1}) & = \bar{\Phi}(\bar{T}_k^m)\hat{x}(t_k) + \bar{\Gamma}(\bar{T}_k^m)\hat{u}(t_k) \; if \; y(t_k) \; is \; lost \\ y(t_k) & = Cx(t_k) \\ \hat{y}(t_k) & = C\hat{x}(t_k) \\ \hat{u}(t_k) & = -\bar{K}(\bar{T}_k)\hat{x}(t_k). \end{cases} \tag{5.16}$$

notice that when $y(t_k)$ is lost the observer will estimate the states based on a prediction further than one step ahead, as previously explained. Now, the packet containing $y(t_k)$ also contains the previous value of the delay τ_k^{ca}, this is the reason to perform a multi-step prediction.

5.4.3 Alternative approaches

It is also possible to implement a predictor (extrapolator) located before the actuators. It uses a sequence of the past control values of $\hat{u}$ to compute the corresponding control input, i.e. $\hat{u}(t_k)$ given $\{\hat{u}(t_{k-1}), \hat{u}(t_{k-2}), ..., \hat{u}(t_{k-n})\}$ where n is the number of previous delays used for the extrapolation and it is defined by the user. The predictor then updates its stored information with the new information available. Adopting this methodology implies that the control input calculation by the predictor requires added computation time. Then, the extrapolation is done to a time equal to $T_{max}/2 + \tau_{ex}$, where τ_{ex} is the computation time of the extrapolator, and we are assuming that it is a constant value.

It is also possible to create a packet $\bar{U}(t_k)$ with extra information to be used in the case of packet loss. When the controller computes the corresponding control $\hat{u}(t_k)$, based on the predicted $\bar{T}_k$, also it computes the corresponding control input for $\hat{u}(t_{k+1})$ given $T_{max}/2$. The packet is defined as

$$\bar{U}(t_k) = \{\hat{u}(t_k), \hat{u}(t_{k+1})\} \tag{5.17}$$

Then the algorithm at the actuators side is

- If $\bar{U}(t_k)$ arrives, then actuators use $\hat{u}(t_k)$ from $\bar{U}(t_k)$
- If $\bar{U}(t_{k+1})$ does not arrive, then actuators use $\hat{u}(t_{k+1})$ from $\bar{U}(t_k)$

We are assuming that the controller is fast enough to do this calculations before the arrival of the next packet from the sensors.

This methodology can be extended to the case where consecutive packet losses occur, however, this depends on the accuracy of the multi step predictions. Also, it is not practical to send large packets since this will increment the computational time required by the controller, and also this can increment the congestion in the network [27].

Chapter 6 – Application to the Inverted Pendulum

In this chapter we present the simulation results of the application to a real case of the proposed NCS models described in the previous chapter. We are integrating the proposed neural network (TDNN) to the NCS models and using a real system to test them. We start by describing the actual system to be controlled: the inverted pendulum on a cart, followed by the presentation of results for different scenarios.

6.1 Inverted Pendulum

One of the most common systems utilized for testing new algorithms in the area of control systems is the inverted pendulum on a cart, depicted in Fig. 6.1. The idea of this electromechanical set-up is that we maintain the pendulum in an upright position atop the cart by making controlled changes in the horizontal position of the cart by a force F.

We denote the cart position coordinate by x and the pendulum angle from the vertical position by θ. In our simulations we assume that the inverted pendulum system has the parameter listed in Table 6.1.

We also assume that the pendulum does not move more than few degrees from the vertical position, therefore the system is linearized about the vertical position

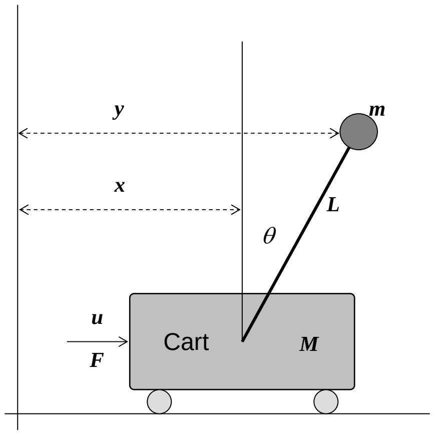

Figure 6.1: Inverted pendulum on a cart.

$\theta = 0$. Lagrange equations in (6.1) give the linearized equations of motion for the inverted pendulum (6.2). The reader is referred to [45] for more information regarding the inverted pendulum.

Table 6.1: Inverted pendulum system parameter.

Inverted Pendulum Parameters		
Mass of the cart	M	2 kg
Mass of the pendulum	m	1 kg
Length to the pendulum center of mass	L	0.5 m
Gravitational constant	g	9.81 m/s^2
Pendulum Angle (initial condition)	θ	0.035 rad

$$\begin{aligned}(M+m)\ddot{x} + mL\ddot{\theta} &= F \\ mL\ddot{x} + mL^2\ddot{\theta} - mgL\theta &= 0\end{aligned} \tag{6.1}$$

Solving for $\ddot{x}$ and $\ddot{\theta}$ we have

$$\begin{aligned} \ddot{x} = \dot{v} &= -\frac{mg}{M}\theta + \frac{F}{M} \\ \ddot{\theta} = \dot{w} &= \frac{(M+m)g}{ML}\theta - \frac{F}{ML}. \end{aligned} \tag{6.2}$$

Then from (6.2) we can get the standard linear model for the inverted pendulum in state-space form as in equation (5.2) which gives

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{m}{M}g & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & \frac{M+m}{ML}g & 0 \end{bmatrix}}_{A} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \\ \frac{1}{M} \\ 0 \\ -\frac{1}{ML} \end{bmatrix}}_{B} u \tag{6.3}$$

Since we are interested in measuring the parameters: pendulum angle θ with respect to vertical and horizontal position x of the cart, then the output is

$$\mathbf{y} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{C} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix}$$

We are assuming that the inverted pendulum is controlled from a remote location by a main controller. The assumptions for the NCS are described in Table

6.2. Using the values from Table 6.1 in (6.3) we have

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -4.9050 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 29.4300 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \\ 0 \\ -1 \end{bmatrix} u \tag{6.4}$$

We can verify the stability of the dynamic system by looking at the eigenvalues

Table 6.2: Assumptions for the NCS.

Networked Control System Assumptions
A communication network exists only between:
- Sensors to controller and controller to actuators
Delays present between:
- Sensor to controller (τ^{sc}) and between controller to actuators (τ^{ca})
The controller, actuators and sensors are event driven
Sensors are directly connected to the plant
Actuators are directly connected to the plant

of the system matrix A, i.e. eigenvalues of A are [0, 0, 5.4249, -5.4249]. From this result, we can see that the system is unstable. For the application of the optimal linear regulator or linear quadratic control (LQR) we defined the weight matrices Q and R as:

$$Q = \begin{bmatrix} w_x & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & w_\theta & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \; R = [1], \tag{6.5}$$

where the element w_x is the weight affecting the position of the cart, and w_θ is the weight affecting the angle of the pendulum. These weighting factors were chosen individually. Q weights different parts of the state. The larger the diagonal entry in Q, the more emphasis the corresponding part of the state receives in driving the system to zero. For example, if the state is the vector $[x_1 \; x_2]^T$, and Q is the diagonal matrix with $(1000, 1)$ on the diagonal, then we really care more about x_1 being driven to 0 than x_2. In the same way, since we are interested in keeping the pendulum upright, we initially selected the weights $w_x = 1$ and $w_\theta = 10$, giving more emphasis to the position of the pendulum. The values of the weights can be changed depending on which variable we are more interested in driving to zero.

Since modern control devices are digital (micro-controllers or computers), we can obtain the discrete representation of the system in (6.4). A fixed sampling period for the inverted pendulum can be obtained by looking at the natural frequency of the system which is determined as $\omega_0^2 = g/l$ [49]. Then $\omega_0 = 4.42 rad/s$. It is suggested in [23] that the sampling period can be determined from $T_s \geq 1/2\omega_0$. Let us assume a fixed sampling period $T_s = 0.03s$, which gives the discrete system of the form (6.6).

$$\begin{bmatrix} x(k+1) \\ \dot{x}(k+1) \\ \theta(k+1) \\ \dot{\theta}(k+1) \end{bmatrix} = \begin{bmatrix} 1 & 0.0300 & -0.0022 & -0.0000 \\ 0 & 1 & -0.1478 & -0.0022 \\ 0 & 0 & 1.0133 & 0.0301 \\ 0 & 0 & 0.8868 & 1.0133 \end{bmatrix} \begin{bmatrix} x(k) \\ \dot{x}(k) \\ \theta(k) \\ \dot{\theta}(k) \end{bmatrix} + \begin{bmatrix} 0.0002 \\ 0.0150 \\ -0.0005 \\ -0.0301 \end{bmatrix} u(k) \tag{6.6}$$

In the following section, some simulations are presented for the system (6.6) under the conditions of fixed sampling period and no packet dropouts. Also we are assuming there are no delays present in the feedback loop.

6.1.1 Fixed sampling period, no delays and no packet loss

The system in (6.6) has a fixed sampling period. Let us assume also that it is subject to no time delays and no data loss. Simulation of the full state feedback system is depicted in Fig. 6.2. It shows the transient responses for the position of the cart x and the angle of the pendulum θ. The initial condition for the position of the pendulum with respect to the vertical was arbitrarily set to $\theta = 0.035rad$. We can see that for the initial position, with no delays present and no packet loss, the pendulum converges to the vertical position in about $6s$, and the maximum distance excursion of the cart is about $0.12m$ returning to its initial position in around $15s$. In this case we adopted the LQR (approach mentioned earlier) to determine the state feedback controller $u(t_k) = -Kx(t_k)$. Fig. 6.3 shows the control input applied to the cart. The initial force reaches a maximum of around $2.25N$ and a lower transient of about $0.25N$, reaching steady state in approximately $10s$.

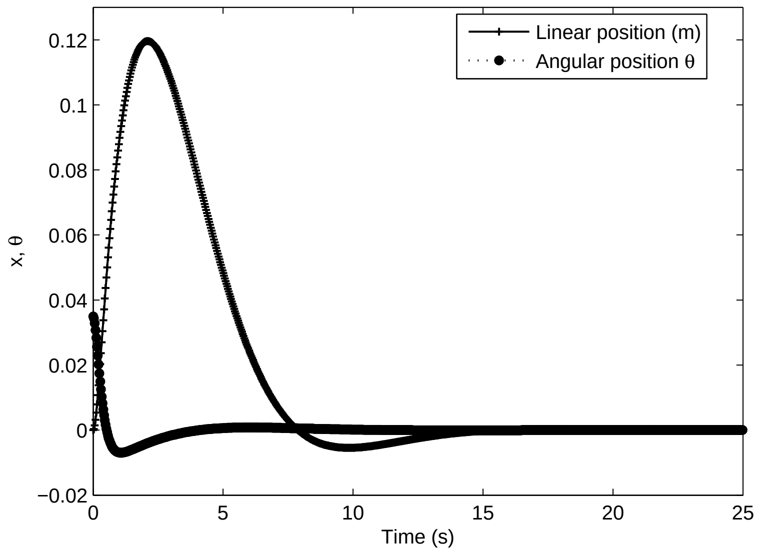

Figure 6.2: Inverted pendulum with no delays.

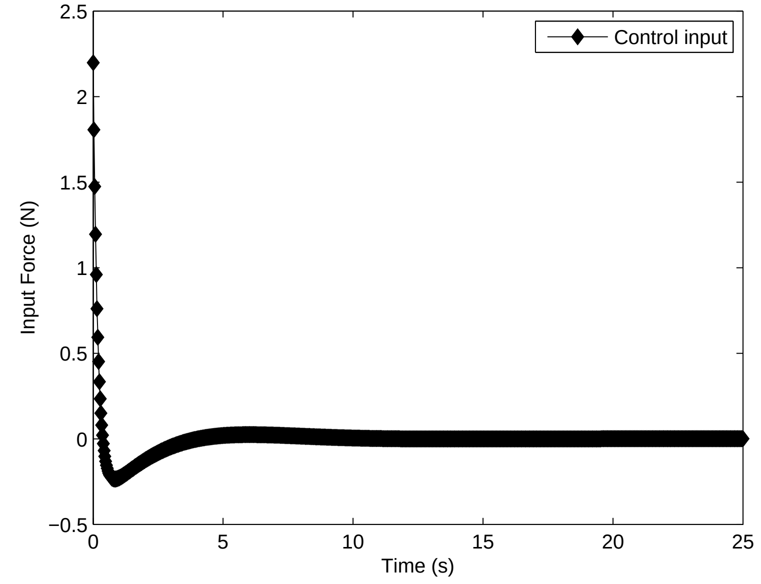

Figure 6.3: Control input with no delays.

6.1.2 Fixed sampling period with delays and no packet loss

Let us now induce delays into the NCS. We set the previous system with the same initial conditions, but this time the system was subject to time delays in both ends: from sensor to controller and from controller to actuator. We assume that each time delay series τ^{sc} and τ^{ca} has a mean $\mu = 0.06s$ and a correlation factor $\beta = 0.5$, as well as the conditions described in Table 6.3. The outcome of the simulation is in Fig. 6.4. We can see the instability caused by the NCS-induced delays in the feedback loop.

Table 6.3: Simulation conditions for the NCS with fix sampling period.

Simulation conditions: fix sampling period
τ^{sc}, mean $\mu = 0.06s$
τ^{ca}, mean $\mu = 0.06s$
Assuming correlation factor $\beta = 0.5$
Assuming no packet dropout
Assuming fixed sampling period $T_s = 0.03$
Pendulum initial conditions:
-Actual plant $0.035rad$ ($2deg$)

In the next section we apply our proposed observer-based variable sampling methodology using predictions from the TDNN previously studied.

6.2 Variable Sampling Period

Now we investigate the application of the variable sampling period to the case of the inverted pendulum under different scenarios.

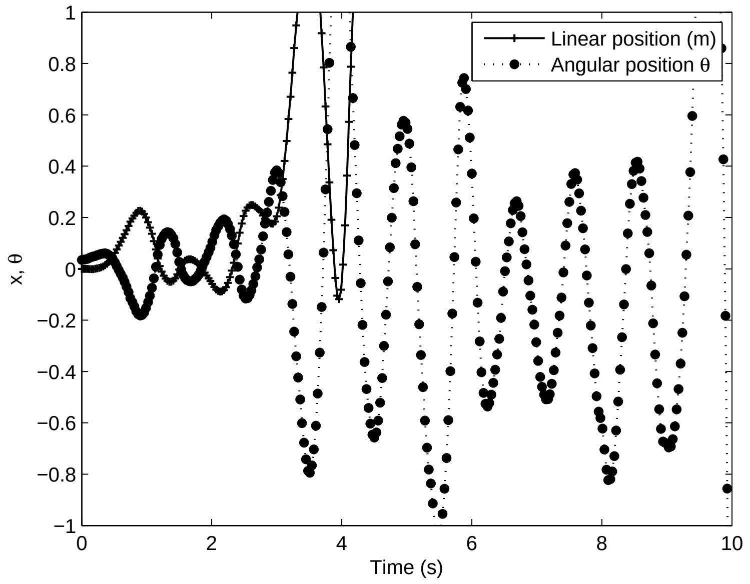

Figure 6.4: System subject to delays, τ^{sc} and τ^{ca} had mean $\mu = 0.06s$.

6.2.1 Presence of Delays and No Packet Loss

For this case we assume that the NCS for the inverted pendulum is subject to delays and has no packet loss occurring either from sensor to controller or from controller to actuator. Let the system in Fig. 5.1 be expressed as in Fig. 6.5, where the dashed lines represent the network connection between the TDNN-observer-controller and the true system. The NCS model is an observer-based NCS with variable sampling. The closed loop for the system model is given by the equations (5.13). We are assuming that the total delay in the NCS is $\tau_k < T_{max}$, and the computational time is small, known and constant, and it can be absorbed by τ^{sc}.

As mentioned before, since there is no packet loss, the controller receives all

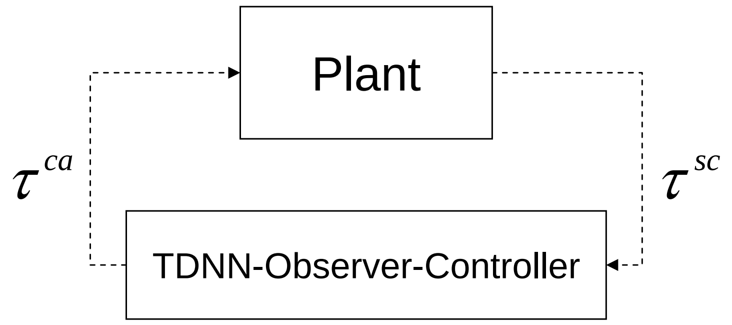

Figure 6.5: Representation of system with delays and no packet loss.

the information at every sampling step k, and given that the packets are time-stamped, then the value of τ_k^{sc} is known. Because of this, the TDNN needs to compute only the predicted value (at every step k) of the delay from controller to actuator in a one-step ahead fashion, i.e. $\bar{\tau}_k^{ca}$. The observer is sampled at every predicted total time delay $\bar{\tau}_k = \bar{T}_k = \bar{\tau}_k^{ca} + \tau_k^{sc}$. After that, the controller sends the estimated control signal $\hat{u}(t_k)$ to the plant.

Let us first investigate the case where the NCS for the inverted pendulum system has a total time delay given by the delay series $\tau^{sc} = \tau^{ca}$, both with mean $\mu = 0.02$ and $\beta = 0.5$, Fig. 6.6. The predictor is an adaptive TDNN named $NET_{5,9,1}$ defined in chapter 5.

Given the previous assumptions and conditions, the behavior of the inverted pendulum is shown in Fig. 6.7. From that figure we can see that the position of the pendulum converges to the vertical in about $5s$ and the maximum distance excursion of the cart is about $0.17m$ returning to its original position in around $15s$. These results are comparable to the ones obtained in the case of no delays

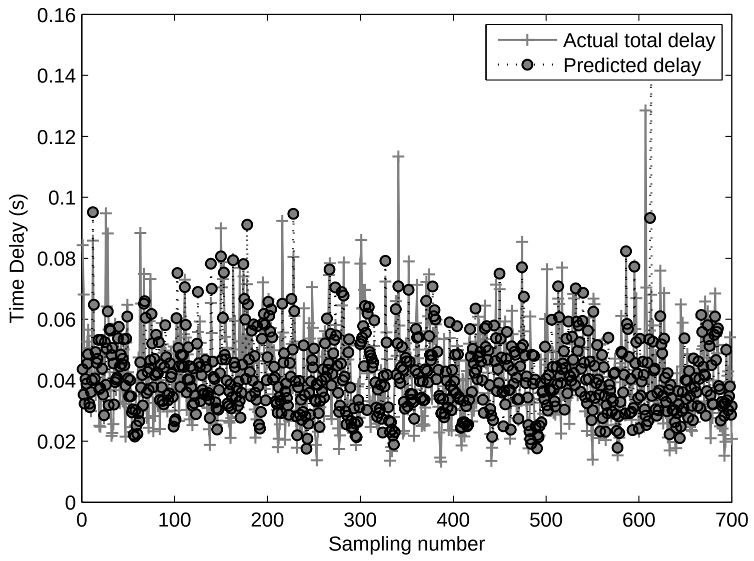

Figure 6.6: Total time delay series, $\tau^{sc} = \tau^{ca}$, $\mu = 0.02$, $\beta = 0.5$.

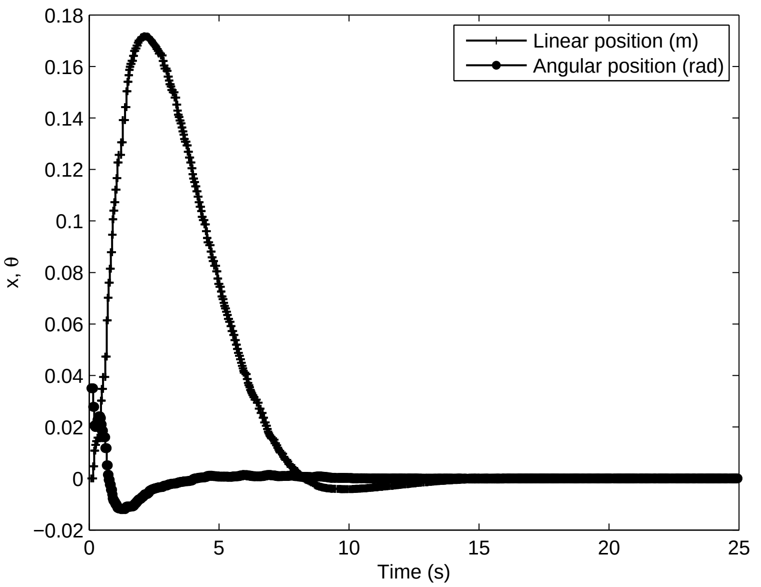

Figure 6.7: Variable sampling applied to inverted pendulum subject to delays.

and no packet drop out shown in Fig. 6.2. The main difference is that the cart has a slightly larger displacement. The control input in Fig. 6.8 shows that the initial force applied to the cart reaches a maximum of $3N$ in the first transient and a lower value of $1.25N$ in the negative transient, reaching steady state in approximately $10s$.

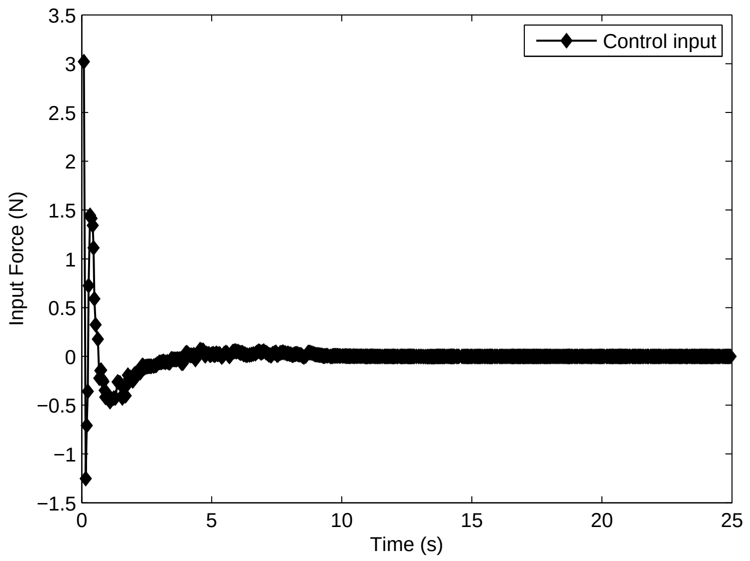

Figure 6.8: Control input, $\tau^{sc} = \tau^{ca}$, $\mu = 0.02$, $\beta = 0.5$.

Let us revise the previously studied case of fixed sampling period with delays and no packet loss (see Fig. 6.4), and apply the proposed variable sampling period scheme. This time the parameters of the simulation were changed to the conditions and assumptions summarized in Table 6.4. In addition, the initial condition of the position of the pendulum (in the observer) with respect to the vertical was set closely to the actual position of the pendulum of the true system to $\theta = 0.055rad$.

After several tests adequate position observer poles (0.3 0.31 0.32 0.33) were found. Fig. 6.9 shows the predicted and actual total time delay sequences.

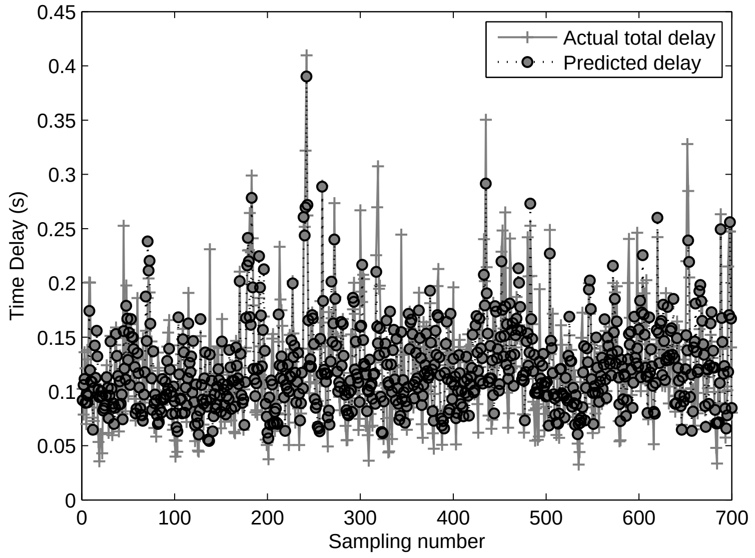

Figure 6.9: Total time delay series, $\tau^{sc} = \tau^{ca}$, $\mu = 0.06$, $\beta = 0.5$.

We can see from the results in Fig. 6.10 that the position of the pendulum is controlled and the vertical position is reached in about $6s$, similar to the results seen in Fig. 6.7. Also, we can observe that this time the maximum distance excursion of the cart is about $0.27m$, however, it returned to the original position in around $15s$ which is comparable to the results in Fig. 6.2 and Fig. 6.7. We can see in Fig. 6.11 that the initial force applied to the cart has three transients. The third transient almost reaches a maximum of $3N$ in the second positive transient and a lower value of approximately $1.5N$ in the negative transient, reaching steady state in approximately $13s$.

Table 6.4: NCS simulation conditions.

Simulation conditions
τ^{sc}, mean $\mu = 0.06s$
τ^{ca}, mean $\mu = 0.06s$
Assuming correlation factor $\beta = 0.5$
$NET_{5,9,1}$ (adaptive)
Assuming no packet dropout
Assuming total delay $\tau_k < T_{max}$
Observer pole location (0.3 0.31 0.32 0.33)
Pendulum initial conditions:
-Actual plant $0.035rad$ ($2deg$)
-Observer $0.05rad$ ($2.9deg$)

From these results and compared to the case of a fixed sampling period with delays in Fig. 6.4, we can see that in this system, under the conditions and assumptions mentioned, our proposed observer-based variable sampling model is able to compensate for delays in the NCS with no packet loss.

We have shown that the proposed observer-based variable sampling period gives satisfactory results when the system is subject to time delays and no packet loss in the communication loop. Particularly in the case of the inverted pendulum, the proposed method gives good results in keeping the position of the pendulum upright. Such results are comparable to the case with a fixed sampling period with no delays and no packet loss (see Fig. 6.2). The case of random time delays and packet loss in the communication loop is investigated in the next section.

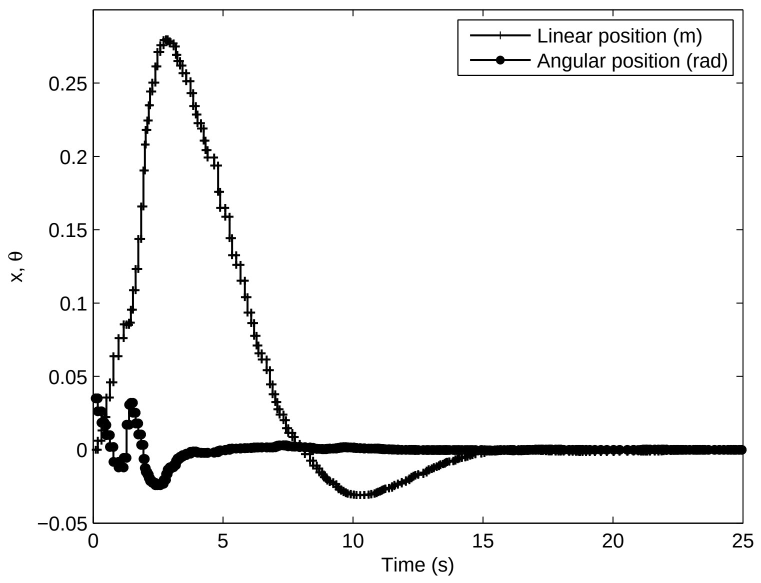

Figure 6.10: Inverted pendulum subject to delays.

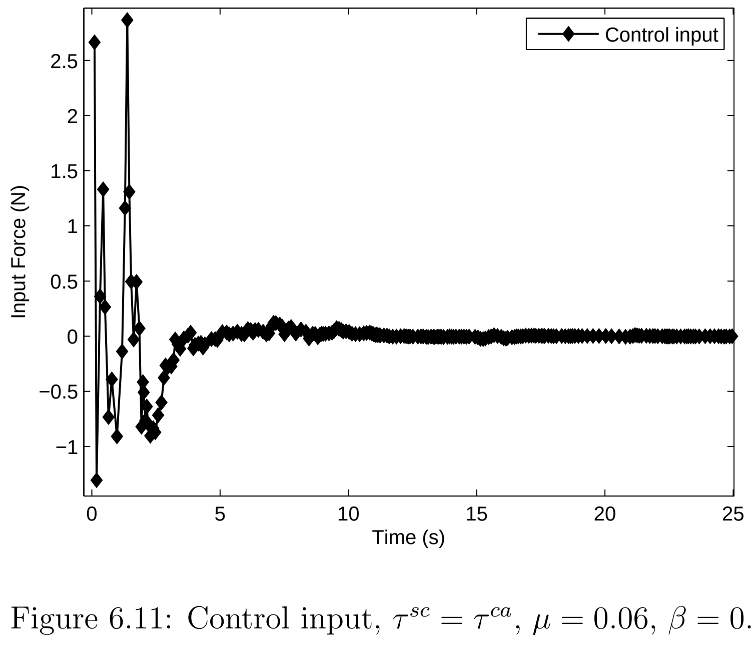

Figure 6.11: Control input, $\tau^{sc} = \tau^{ca}$, $\mu = 0.06$, $\beta = 0.5$.

6.3 Presence of delays and packet dropout

In this section we investigate the application of a variable sampling period to the case of the inverted pendulum in a NCS subjected to random delays and packet dropout. Let us represent the model in Fig. 5.2 as in Fig. 6.12.

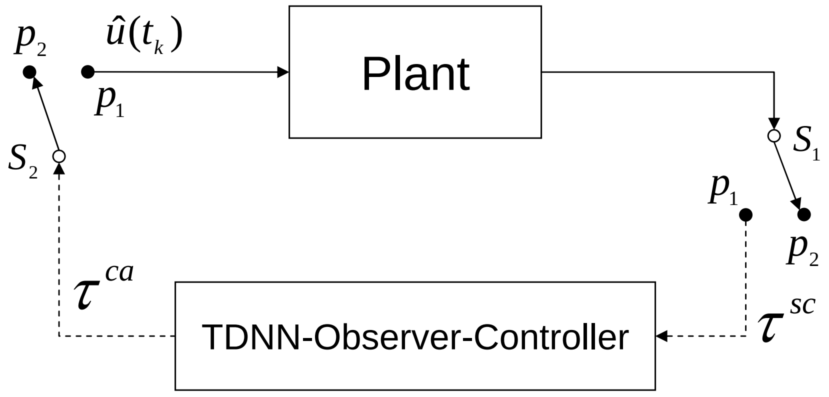

Figure 6.12: NCS with random delays and packet loss.

We start with the application of our algorithm given in (5.15). Recall that this algorithm tells the controller and actuators to use the last information available in the case when a packet does not arrive, i.e. when s_1 and s_2 are in position p_2 in Fig. 6.12. As we saw in previous examples, the proposed variable sampling method is able to stabilize the inverted pendulum model in the presence of delays. We now assume the system has packet losses occurring from sensor to controller and from controller to actuator. The conditions of the simulation are similar to those used in the example depicted in Fig. 6.7 with the addition of an assumed value of $T_{max}/2 = 0.03s$. The conditions are given in Table 6.5.

At this point it is important to remark that the packet arriving to the controller

Table 6.5: NCS simulation conditions.

Simulation conditions
τ^{sc}, mean $\mu = 0.02s$
τ^{ca}, mean $\mu = 0.02s$
Assuming correlation factor $\beta = 0.5$
$NET_{5,9,1}$ (adaptive)
Assuming packet dropouts in both ends
Assuming $T_{max}/2 = 0.03s$
Observer pole location (0.3 0.31 0.32 0.33)
Pendulum initial conditions:
-Actual plant $0.035rad$ ($2deg$)
-Observer $0.05rad$ ($2.9deg$)

at sampling step k is timestamped, but also it contains the information of the state of the plant and the last delay value of the last packet sent from the controller to the actuator (τ^{ca}_{k-1}). When a packet traversing from controller to actuator is dropped at its corresponding sampling step k, i.e. when $\tau^{ca}_{k} \geq T_{max}/2$, we define the construction of the packet (I^{sc}_{k}) sent from sensors to controller at sampling step k as (6.7).

$$\begin{aligned} I^{sc}_{k} &= \{y(t_k),\ \tau^{ca}_{k-1} = T_{max}/2,\ timestamp\},\ if\ \tau^{ca}_{k-1} \geq T_{max}/2 \\ I^{sc}_{k} &= \{y(t_k),\ \tau^{ca}_{k-1},\ timestamp\},\ if\ \tau^{ca}_{k-1} < T_{max}/2 \end{aligned} \tag{6.7}$$

where $y(t_k)$ is the output value of the state at time t_k, τ^{ca}_{k-1} is the time delay value of the packet sent from controller to actuator at sampling step $(k-1)$, $timestamp$ is the time at which the packet I^{sc}_{k} was sent. Assuming the packet I^{sc}_{k} reaches the controller, the value of τ^{ca}_{k-1} is passed to the tapped delay line of the TDNN which generates the predicted value $\bar{\tau}_k^{ca}$. Then the observer will have a sampling period equal to $\bar{T}_k = \bar{\tau}_k^{ca} + \tau_k^{sc}$, where τ_k^{sc} is computed from *timestamp*.

Note in (6.7), that the historic data the TDNN is receiving about τ^{ca} is upper bounded to $T_{max}/2$ since every packet (from controller to actuators) with delay greater than that is discarded. Therefore the delay τ_{k-1}^{ca} with value $T_{max}/2$ at sampling step k is the most recent information available to the TDNN to compute the corresponding prediction. Recall that in NCS the newest measurement (data) information is always more valuable than old measurements. Furthermore, in the case of packet loss from sensor to controller the value of τ_k^{sc} is also equal to $T_{max}/2$ at sampling step k, i.e. τ_k^{sc} is upper bounded, and multi-step prediction is activated as explained in chapter 5.

In the event of packet loss from sensor to controller we use the criteria given in 5.14 which allows the controller to use the most recent value $(y(t_{k-1}))$ saved in a buffer or queue. Similarly, when no input arrives to the actuators by the time $T_{max}/2$ is up, they will use the most recent value of the input available i.e. $u(t_{k-1})$. These criteria are given as:

$$\begin{aligned} p_1 &: u(t_k) = u(t_k) \\ p_2 &: u(t_k) = u(t_{k-1}) \end{aligned} \tag{6.8}$$

The following results were obtained by applying the algorithm in (5.15) to the inverted pendulum case. We assumed the conditions given in Table 6.5. Under these conditions the controller for the inverted pendulum experienced 30% packet loss. Fig. 6.13 (a) shows the sequence of delays affecting the packets traversing from sensor to controller, the horizontal line at $0.03s$ marks the assumed value of $T_{max}/2$, so τ^{sc} is upper bounded. Also in Fig. 6.13 (b), the history of the delays (from controller to actuator) as seen by the predictive TDNN in the moment of computing the predicted value $\bar{\tau}_k^{ca}$ is shown. The history series τ^{ca} is also upper bounded which means that the actuators did not receive the corresponding control signal, making use of the most recent control signal previously received. Predictions by the TDNN of this series are shown also in Fig. 6.13 (b). The black dots represent the multi-step predictions generated by the TDNN when a packet sent is dropped.

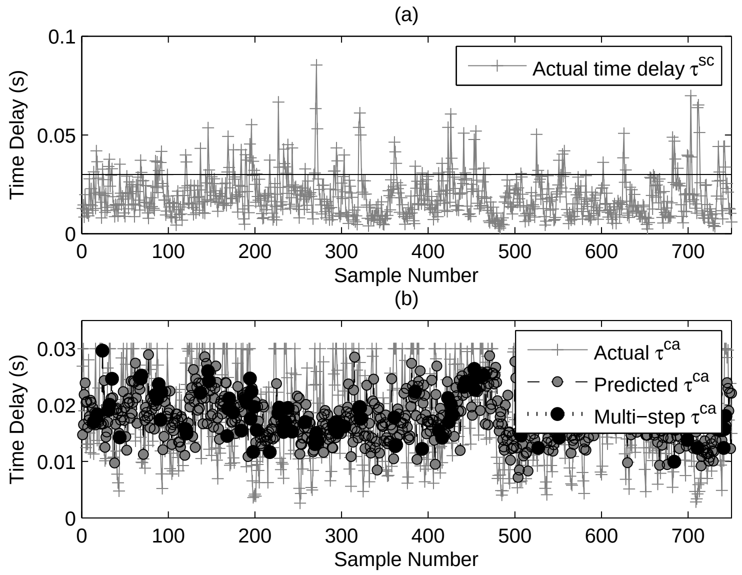

Figure 6.13: Bounded time delay series τ^{sc} (a) and τ^{ca} (b).

Let us now investigate the effects that the induced delays and packet loss shown in Fig. 6.13 have on the NCS for the inverted pendulum. We can see in Fig. 6.14

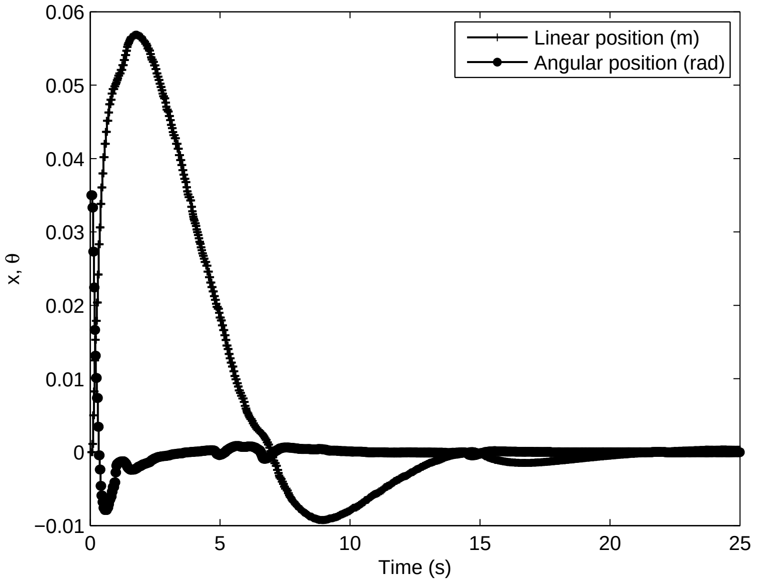

Figure 6.14: Inverted pendulum subject to delays and 30% packet dropout.

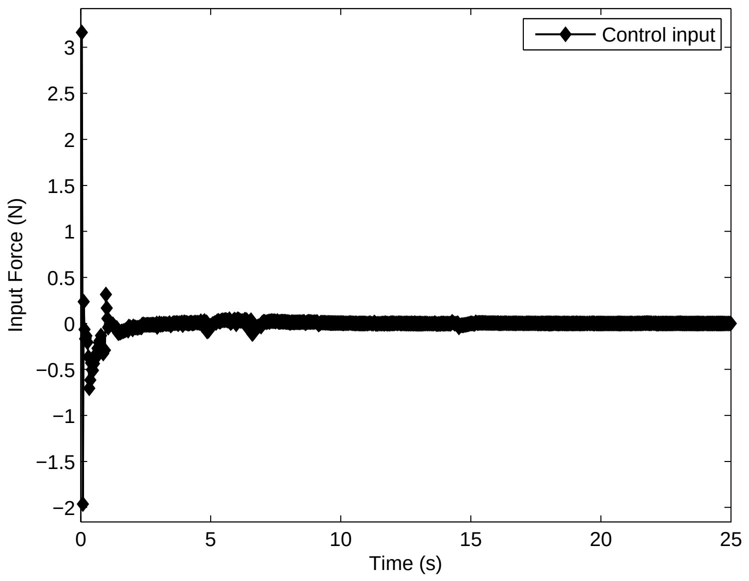

Figure 6.15: Control input, $\tau^{sc} = \tau^{ca}$, $\mu = 0.02$, $\beta = 0.5$.

that even with this amount of information loss, the algorithm can compensate and stabilize the system in about $6s$ to $7s$ with some small oscillations in the interval from $5s$ to $10s$. Also, the maximum distance excursion of the cart is about $0.055m$. We can see in the same figure that the position of the cart has a negative transient close to $-0.01m$ which is larger than the similar case shown in Fig. 6.7. These small oscillations are attributed to the difference between the actual control value $(\hat{u}(t_k))$ needed at $T_{max}/2$ and the value $(\hat{u}(t_{k-1}))$ used by the actuators. In Fig. 6.15 we can see that the initial force applied to the cart has an initial transient of about $3N$ and a negative transient of approximately $2N$, reaching steady state in approximately $7s$.

Comparing the results between Fig. 6.2 and Fig. 6.14 it can be seen that the displacement of the cart is smaller in Fig. 6.14. Also the position of the pendulum after the first transient returns close to the vertical position in a shorter time than the case with no packet loss. These effects are because the control input is larger compared to the case with no packet loss. Large control input (force) causes the cart to move faster at the beginning of the test with a short displacement. This is reflected in the position of the pendulum being close to the vertical in a shorter time around $2.5s$ compared to the $5s$ for the case of no-packet loss.

In further analysis we found that the system starts showing instability when the number of packets dropped is close to 50%. This can be seen in Fig. 6.16, where the system was subject to 49% packet dropout. Even though an oscillatory behavior of the carts is shown, the position of the pendulum is kept upright almost constantly. Comparative results of the force applied to the cart are shown between

Fig. 6.15 and Fig. 6.17.

As we saw in the results of the case of fixed sampling time with delays shown in Fig. 6.4 (see conditions in Table 6.3), the system becomes unstable. Furthermore, the same system under the same conditions but using the proposed observer-based variable sampling method results in a controlled system (see Fig 6.10). We now investigate the same system, but this time it is subject to packet dropouts and we assume a maximum time delay $T_{max} = 0.3$ occurring from sensor to controller and from controller to actuator. This accounts for a total of 12.3% packet loss. We can see in Fig. 6.18 that the proposed method of observer-based variable sampling period was not able to compensate (in the event of packet dropouts) and the inverted pendulum became unstable. The use of the most recent input and output values are not sufficient to allow the system to compensate under these particular conditions. The actuators are using a control input value corresponding to a step further back in time which is not the value needed for the system to compensate. This shows why it is important to consider the characteristics of the network delays when a NCS is designed and the limitations of the system to be controlled through a network.

Let us now consider the proposed methodology described in equations (5.16) to treat the case of packet dropouts. This methodology differs from the previously studied examples above (see equations (5.15)) in that it uses the most recent value of the control input ($\hat{u}(t_{k-1})$) in the case of packet dropout from controller to actuators, but this time it will not use the most recent value of the plant output $y(t_k)$ to calculate the difference between the measured and estimated outputs

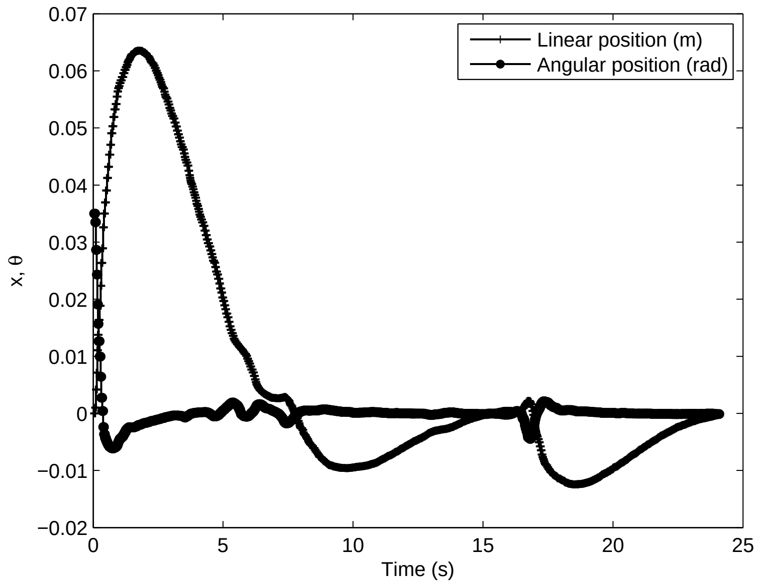

Figure 6.16: Inverted pendulum subject to delays and 49% packet dropout.

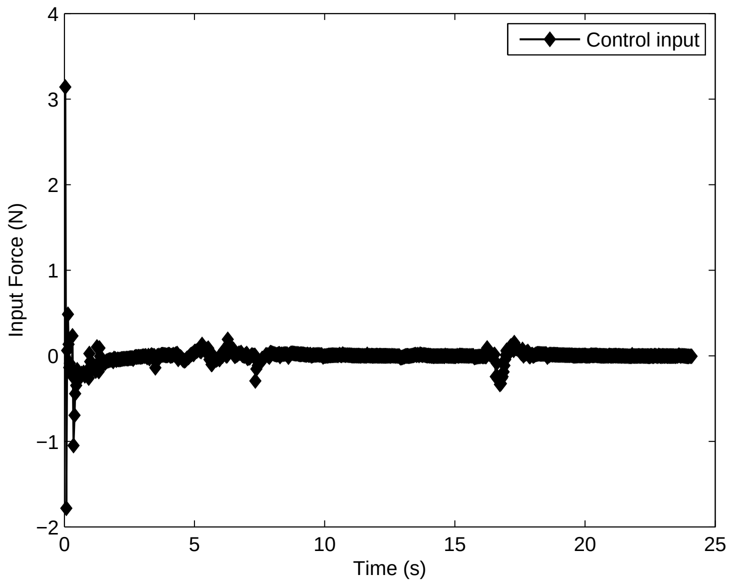

Figure 6.17: Control input, $\tau^{sc} = \tau^{ca}$, $\mu = 0.02$, $\beta = 0.5$.

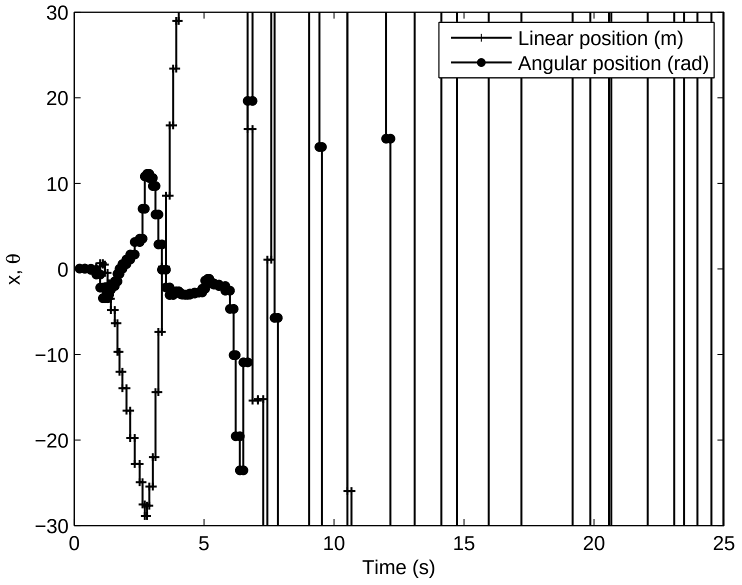

Figure 6.18: Control input, $\tau^{sc} = \tau^{ca}$, $\mu = 0.06$, $\beta = 0.5$, 12.3% Dropout.

$y(t_k) - \hat{y}(t_k)$. This methodology is based on the assumption that under perfect prediction of delays, the states given by equations (6.9) (see (5.16)) are the same.

$$\begin{cases} x(t_{k+1}) & = \Phi(T_k)x(t_k) + \Gamma(T_k)\hat{u}(t_k) \\ \hat{x}(t_{k+1}) & = \bar{\Phi}(\bar{T}_k^m)\hat{x}(t_k) + \bar{\Gamma}(\bar{T}_k^m)\hat{u}(t_k) \end{cases} \tag{6.9}$$

In the event of packet dropout from sensor to controller, the controller will compute the control signal based only on the states given by the observer with no information from the system output. Then in the absence of the information packets described in (6.7), the sampling interval ($\bar{T}_k^m$) for the observer becomes the

Table 6.6: NCS simulation conditions.

Simulation conditions
τ^{sc}, mean $\mu = 0.02s$
τ^{ca}, mean $\mu = 0.02s$
Assuming correlation factor $\beta = 0.5$
$NET_{5,9,1}$ (adaptive)
Assuming packet dropouts in both ends
Assuming $T_{max}/2 = 0.03s$
Observer pole location (0.3 0.31 0.32 0.33)
Pendulum initial conditions:
-Actual plant $0.035rad$ ($2deg$)
-Observer $0.05rad$ ($2.9deg$)

time out value $\tau_k^{sc} = T_{max}/2$ added to the multi-step predicted value $\bar{\tau}_k^{ca}$ computed by the TDNN, i.e. $\bar{T}_k^m = \tau_k^{sc} + \bar{\tau}_k^{ca}$. To test this proposed methodology we revise the example given in Table 6.5 which is shown again in Table 6.6.

As we saw, the NCS for the inverted pendulum is subject to delays and packet dropouts. It is important to remark that to have consistency in our simulations and be able to compare results, we use the same sequences of delays. The only difference is the control algorithm used.

We can see in Fig. 6.19 that this new proposed algorithm is able to keep the pendulum in a vertical position with a small oscillation at $5s$. Furthermore, the pendulum is subject to a 30% packet loss. This proves that the algorithm works under these particular conditions. The transient of the pendulum position is comparable to the one shown in Fig. 6.14 that uses the previous control algorithm given in (5.15). Furthermore, the overall results in this test are improved as we

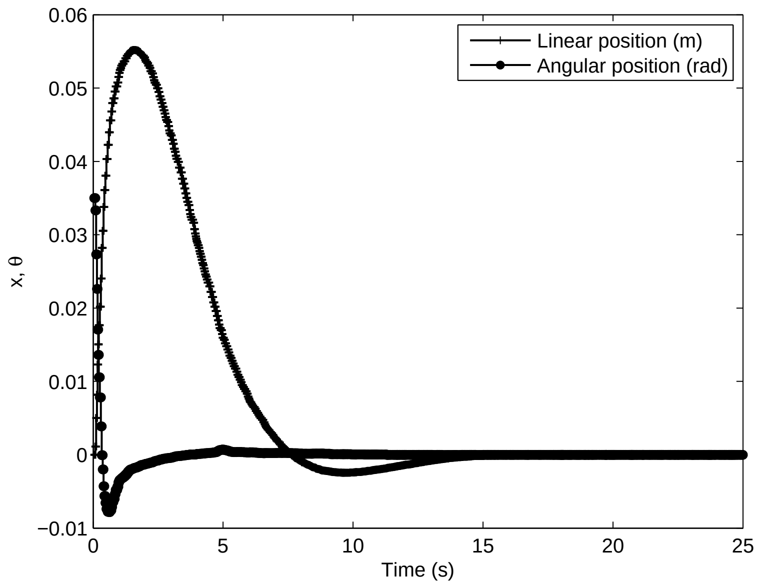

Figure 6.19: Inverted pendulum subject to delays and 30% packet dropout.

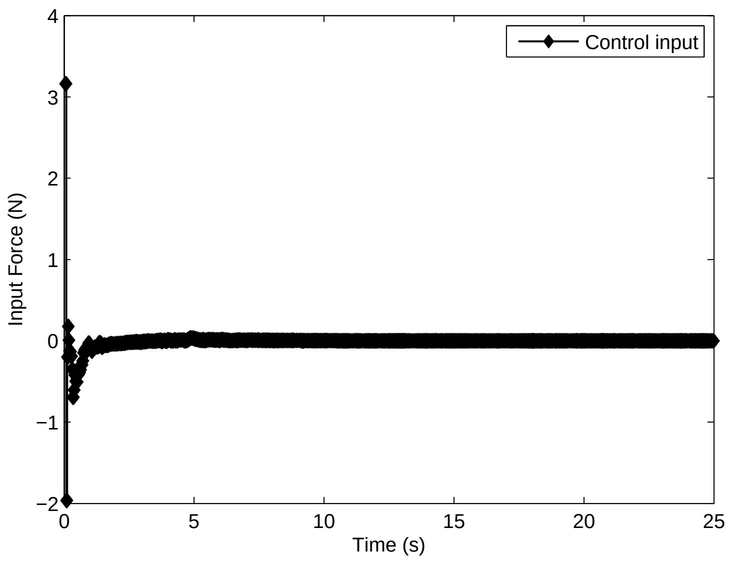

Figure 6.20: Control input, $\tau^{sc} = \tau^{ca}$, $\mu = 0.02$, $\beta = 0.5$, 30% Dropout.

can see that the distance excursion of the cart in Fig. 6.20, has less variablilty and the negative transient has decreased.

6.4 Conclusion

The methodologies presented here have shown good potential and effectiveness in controlling a classic dynamic system under the assumptions and conditions presented. We have shown that the algorithms presented here are capable of controlling a system subject to time delays with no packet dropouts. Particularly it has been shown that the proposed algorithms are able to stabilize the system of the inverted pendulum on a cart, under the assumed conditions.

Furthermore, we have shown that in the event of packet dropouts, the algorithms which use the most recent control input value ($u(t_{k-1})$) and the most recent output value ($y(t_{k-1})$) can compensate for the effects of time delays in packet transmission. However, the larger the delays in the NCS the further back the values ($u(t_{k-1})$) and ($y(t_{k-1})$) are. This results in a lack of values useful in compensating for the dropouts in the system. The methodologies presented here can be used with understanding of the limitations of a particular system it is desired to control.

Chapter 7 – Conclusion and Future Work

In this dissertation the development of a variable sampling methodology to mitigate the effects of time delays and packet dropouts in networked control systems using neural networks was presented. The methodology was applied to control dynamic systems remotely located. For the method here presented we used predicted values of time delays occurring in the feedback communication loop as the sampling instants of the system model. Such delays are time variant.

One of the contributions of this work is the application of an adaptive TDNN to perform one-step-ahead predictions of the time delays occurring in the communication loop. The TDNN uses a sequence of previous time-delay values to calculate the next value of the sequence. We showed that with a relatively simple TDNN with five input nodes in the tapped delay line, nine nodes in the hidden layer and one output, it is possible to achieve predictions of a time-delay sequence in a one-step-ahead fashion. Furthermore, we proposed a TDNN configuration for the cases when multistep prediction is required. In such a configuration, the predicted time delay is fedback to the input layer in the event that a packet from sensor to controller in the NCS is considered lost or dropped. In addition, we showed that multistep predictions are possible under certain conditions.

We presented an iterative way to obtain the most suitable TDNN topology for our particular application. More importantly, it should be emphasized that although the proposed method is of the iterative trial-and-error type, the validation of every topology plays an important role in selecting the most appropriate topology.

Another contribution is a novel networked control system design. We presented an observer-based variable sampling control model. The predicted value from the TDNN is used as the sampling time of the NCS model. The optimal control gain is calculated using a linear quadratic regulator for every predicted time-delay value and the control signal is sent to the actuators. This novel approach is designed to compensate for systems with time delays and packet dropouts. This dissertation showed that this design is able to compensate for the time delays in the communication loop and also for dropped packets with some limitations.

Using the inverted pendulum example we showed that the proposed NCS model can stabilize a real life dynamic system model. We showed that such a system subject to time delays and up to 50% packet loss can be stabilized. On the other hand, the proposed model has limitations, namely, the proposed control methodology is not able to compensate for packets dropped when the system (inverted pendulum) is subject to delays with a mean around $\mu = 0.12s$ and 12.3% packet loss.

The methodologies presented here have shown good potential and effectiveness in controlling a classic dynamic system under the assumptions and stated conditions.

7.1 Future Work

Due to the existence of multiple kinds of possible networks to be used in control systems, it would be convenient to test this methodology with the most commonly used ones. This means that real data of the delays occurring in a particular network have to be estimated. This can be a complicated task due to the vast variety and complexity of each network in use today.

This dissertation showed the prediction capabilities of the TDNN, especially when the correlation factor of the sequence increases. A study of the amount of correlation between time delays occurring in a particular network would tell the designer if this methodology is applicable, and if so, under which limits this methodology can work.

It is important to remark that the methods here presented are based on predictions. It is logical to investigate the possibility of implementing or creating a better predictor. Also it is important to keep in mind that many of the applications may require high speed or real time computations. Then a proposed predictor has to be designed for a particular condition and application.

Further strategies to compensate for dropped packets can be pursued. It is possible to think of predicting or estimating the states or control values contained in dropped packets by using a TDNN.

Bibliography

[1] Anil K. Jain, and Jianchang Mao, *Artificial Neural Network: A tutorial*, IEEE Computer, 1996

[2] Yodyium Tipsuwan and Mo-Yuen Chow, *Control Methodologies in Networked Control Systems*, Engineering Practice, 11 pp 1099-1111, 2003

[3] Dan Shi, Hongjian Zhang, and Liming Yang , *Time-Delay Neural Network for the Prediction of Carbonation Tower's Temperature*, IEEE Transactions on Instrumentation and Measurement, Vol. 52, No 4, August 2003

[4] Nahid Ardalani, Ahmadreza Khoogar, and H. Roohi, *A Comparison of Adaline and MLP Neural Network based Predictors in SIR Estimation in Mobile DSCDMA Systems*, Proceedings of the World Academy of Science, Engineering and Technology, vol. 9, November 2005

[5] Nilsson, J, *Real-Time Control Systems with Delays*, PhD Thesis, Lund Institute of Technology, Lund, Sweden, 1998

[6] Samir Shaltaf, *Neural-Network-Based Time-Delay Estimation*, EURASIP Journal on Applied Signal Processing 2004

[7] Li Hongyan, Wang Hong and Gui Chao, *Internet Time-delay Prediction Based on Autoregressive and Neural Network Model*, International Conference on Communications, Circuits and Systems Proceedings, 2006

[8] Q. P. Wang, D.L. Tan, Ning Xi and Y. C. Wang, *The Control Oriented QoS: Analysis and Prediction*, International Conference on Robotics and Automation, Seoul Korea, 2001

[9] S.R. Seyed Tabib, and Ali A. Jalali, *Modelling and Prediction of Internet Time-delay by Feed-forward Multilayer Perceptron Neural Network*, IEEE Tenth International Conference on Computer Modeling and Simulation, April 2008

[10] Jianqiang Yi, Qian Wang, Dongbin Zhao, and John T. Wen, *BP neural network prediction-based variable-period sampling approach for networked control systems*, Applied Mathematics and Computation, Elsevier, 2007

[11] Liu Jiangang, Liu Biyu, Zhang Ruifang, and Li Meilan, *The New Variable-period Sampling Scheme for Networked Control Systems with Random Time Delay Based on BP Neural Network Prediction*, Proceedings of the 26th Chinese Control Conference, 2007

[12] Yan Xue, and Ke Liu, *Analysis of Variable-Sampling Networked Control System Based on Neural Network Prediction*, Proceedings, International Conference On wavelet Analysis and Patern Recognition, 2007

[13] Martin T. Hagan and Mohammad B. Menhaj, *Training Feedforward Networks with the Marquardt Algorithm*, IEEE Transactions on Neural Networks, vol. 5, No. 6, pp 989-993, November 1994.

[14] P. Raptis, V. Vitsas, A. Banchs, K. Paparrizos, *Delay Distribution Analysis of IEEE 802.11 with Variable Packet Length*, in Proc. of the 65th IEEE Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th, pp 830-834, April 2007.

[15] Payam Naghshtabrizi and Joao P. Hespanha, *Stability of network control systems with variable sampling and delays*, In Proc. of the Forty-Fourth Annual Allerton Conf. on Communication, Control, and Computing, 2006

[16] Yan Xue; Ke Liu; , *Controller design for variable-sampling networked control systems with dynamic output feedback* Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on , vol., no., pp.6391-6396, 25-27 June 2008

[17] Anta, A.; Tabuada, P.; , *Self-triggered stabilization of homogeneous control systems*, American Control Conference, 2008 , vol., no., pp.4129-4134, 11-13 June 2008

[18] J. H. Sandee, *Event-driven Control in Theory and Practice*. Eindhoven, the Netherlands: PhD thesis, Technische Universiteit Eindhoven, 2007.

[19] P. Raptis, A. Banchs, V. Vitsas, K. Paparrizos, and P. Chatzimisios, *Delay Distribution Analysis of the RTS/CTS mechanism of IEEE 802-11* in Proc of

the 31 st IEEE Conference on Local Computer Networks, pp. 404-410, Nov. 2006.

[20] Nargess Sadeghzadeh, Ahmad Afshar, Mohammad Bagher Menhaj, *An MLP neural network for time delay prediction in networked control systems*, Chinese Control and Decision Conference, 2008

[21] Danilo P. Mandic, Jonathon A. Chambers *Recurrent Neural Networks For Prediction*, Wiley, 2001.

[22] Gene F. Franklin, J. David Powell, Michael L. Workman *Digital control of dynamic systems*,Addison-Wesley Pub. Co. 1990

[23] Stevens, Brian L., and Lewis, Frank L., *Aircraft control and simulation*, Wiley, New York, 1992

[24] Brian D.O. Anderson, and John B. Moore, *Optimal Filtering*, Dover,2005

[25] Xiangheng Liu, and Andrea Goldsmith, *Wireless Medium Access Control in Networked Control Systems*, Report

[26] Rafael Camilo Lozoya Gmez, Pau Mart, Manel Velasco and Josep M. Fuertes, *Wireless Network Delay Estimation for Time Sensitive Applications* Research report ESAII RR-06-12, Technical University of Catalonia, 2006

[27] Kondo, Y. and Itaya, S. and Yamaguchi, S. and Davis, P. and Suzuki, R. and Obana, S. *Wireless Channel Detection Based on Fluctuation of Packet Arrival*

Interval,Networks, 2007. ICON 2007. 15th IEEE International Conference on, 2007.

[28] John G. Proakis and Masoud Salehi, *Contemporary Communication Systems Using Matlab*, Pacific Grove, CA: Brooks/Cole, 2000, pp. 56-57.

[29] Liqian Zhang, Yang Shi, Tongwen Chen, and Biao Huang *A New Method for Stabilization of Networked Control Systems with Random Delays*, American Control Conference, 2005.

[30] E. Witrant, C. Canudas-de-Wit, D. Georges, and M. Alamir *Remote Stabilization Via Communication Networks With a Distributed Control Law*, Automatic Control, IEEE Transactions on , vol.52, no.8, pp.1480-1485, Aug. 2007

[31] Yasar Becerikli, Yusuf Oysal, *Modeling and prediction with a class of time delay dynamic neural networks*, Applied Soft Computing, 7, Elsevier, 2007

[32] Radu Drossu, and Zoran Obradovic, *Stochastic Modelling Hints for Neural Network Prediction*, Document, University of Washington, 1995

[33] Wei Zhang, Michael S. Branicky, and Stephen M. Phillips *Stability of Networked Control Systems*, IEEE Control Systems Magazine, February 2001

[34] Luck, Rogelio, and Ray Asok, *An Observer-based Compensator for Distributed Delays*, Automatica. Vol. 26, pp. 903-908. Sept. 1990

[35] Luck, Rogelio, and Ray Asok, *experimental verification of a delay compensation algorithm for integrated communication and control*, International Journal of Control, Vol. 59, pp.1357-1372, 1994

[36] Jiwei Hua, Tao Liang, Hexu Sun and Zhaoming Lei, *Time-delay Compensation Control of Networked Control Systems Using Time-stamp based State Prediction*,Computing, Communication, Control, and Management, 2008. CCCM '08. ISECS International Colloquium on , vol.2, no., pp.198-202, 3-4 Aug. 2008

[37] Adya M. and Collopy F. *How Effective are Neural Networks at Forecasting and Prediction? A Review and Evaluation*,Journal of Forecasting ,vol.17, pp.481-495, 1998

[38] Hagan, M. and Demuth, H., *Neural Networks for Control*, Proceedings of the American Control Conference, 1999.

[39] A. Zaknich *Principles of Adaptive Filters and Self-learning Systems*, Springer, 2005.

[40] Baum E. B. and Haussler D. (1988), *What size net gives valid generalization?*, Neural Computation, 1, pp. 151-160.

[41] R. Zemouri, D. Racoceanu, N. Zerhouni, *Recurrent Radial Basis Function Network for Time-Series Prediction*, Engineering Applications of Artificial Intelligence 16 (5-6) (2003) 453-463.

[42] Bernard Widrow, Michael A. Lehr, *30 Years Of Adaptive Neural Networks: Perceptron, Madaline, and Backpropagation*, Proceedings of the IEEE, Vol. 78, No. 9, 1990.

[43] C. Lee Giles, Steve Lawrence and Ah Chung Tsoi, *Noisy Time Series Prediction using Recurrent Neural Networks and Grammatical Inference*, Machine Learning, 44, 161183, 2001

[44] Frank, R. J. and Davey, N. and Hunt, S. P., *Time Series Prediction and Neural Networks*, Journal of Intelligent and Robotic Systems, Volume 31 Issue 1-3, May -July 2001

[45] Belanger, Pierre R., *Control Engineering: A modern Approach*, Oxford University Press, Inc., New York, NY, USA. 1995

[46] Richard O. Duda, Peter E. Hart and David G. Stork, *Pattern Classification*, John Wiley and Sons, 2000.

[47] Aström, Karl J. and Wittenmark, Björn, *Computer-controlled systems: theory and design (2nd ed.)*, Prentice-Hall, Inc., 1990.

[48] James F. Kurose and Keith W. Ross, *Computer Networking: A Top-Down Approach (4th Edition)*, Addison Wesley, 2007.

[49] Eugene I. Butikov, *On the dynamic stabilization of an inverted pendulum*, American Association of Physics Teachers, 2001.