Katarina Stanojević, BSc

Robust Control of Networked Systems: Buffering, Control Design and Realization

Master's Thesis

to achieve the university degree of

Master of Science

Master's degree programme: Electrical Engineering

submitted to

Graz University of Technology

Supervisor

Ass. Prof. Dipl.-Ing. Dr.techn. Martin Steinberger

Institute of Automation and Control

Head: Univ.-Prof. Dipl-Ing. Dr.techn. Martin Horn

Graz, September 2020

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis.

Date	Signature

To my grandmother, Jelena.

Abstract

Rapid development of communication technology in the last decades led to a tendency for replacement of traditional point-to-point wiring with modern communication networks which resulted in a wide application of Wireless Networked Control Systems. These system architectures received significant attention due to their prominent characteristics such as flexibility and low maintenance cost. However, the use of a communication channel in control systems introduces some challenges from the control theory point of view. In order to reduce the impact of network imperfections such as variable time delays and packet dropouts on control performance, it is necessary to consider these uncertainties in the control system design.
Besides the uncertainties resulting from the used network, every practical realization of a control system is affected by disturbances. Although modeling of Networked Control System (NCS) and stability properties have been studied in literature extensively, the potential of robust control techniques such as sliding mode control algorithms for disturbance rejection in perturbed NCS has still not been fully exploited.

The focus of this master's thesis lies on the application of integral sliding mode technique to robustify networked systems. To overcome the problem of network-induced uncertainties which leads to a complex time-variant mathematical model, a specific buffering mechanism is proposed. The buffered networked control system is motivated by the existing approaches available in literature and adapted with the goal to reduce the time delay introduced by the buffer. This results in a simplified time-variant mathematical model of NCS which, with an appropriate unknown input compensation can be made suitable for the use of a discretized version of the super-twisting algorithm.
Furthermore, to fully understand the behavior and practical limitations of NCS and to examine the control performance of the proposed approach in a real world application, a Wireless Networked Control System is implemented in Matlab/Simulink using an Arduino board and Simulink support packages. The simulation and experimental results show effectiveness with respect to disturbance rejection. Moreover, the delay resulting from the proposed buffering mechanism is considerably reduced in comparison to the existing approaches available in literature and a very good system performance is achieved.

Contents

List of Figures

List of Abbreviations

LMI . . . . . Linear Matrix Inequality

NCS . . . . . Networked Control System

WNCS . . . . Wireless Networked Control System

TCP . . . . . Transmission Control Protocol

UDP . . . . . User Datagram Protocol

SMC . . . . . Sliding Mode Control

ISMC . . . . . Integral Sliding Mode Control

IP address . . Internet Protocol address

1 Introduction

Contents

1.1 Brief Introduction to Networked Control Systems

Networked Control Systems (NCSs) are system architectures, where the system to be controlled including sensors and actuators is connected over a communication channel to the controller . The communication links between the plant and controller side can be implemented either by wired (e.g. Ethernet) or wireless networked technologies (e.g wireless LAN), depending on the specific application of the system.

The rapidly growing interest in Networked Control Systems (NCS) over the past decades has been motivated by many benefits they offer, such as increased flexibility and low installation and maintenance costs. In addition, closing the control loop over a wireless network technology significantly reduces the system wiring [1]. These advantages, together with the development of communication technologies, resulted in applications of this control architecture in industry.

Nevertheless, the presence of a communication network in the control loop introduces some unfavorable effects, which if not properly modeled and considered in the control design, can degrade the performance of these systems or even lead to instability. These

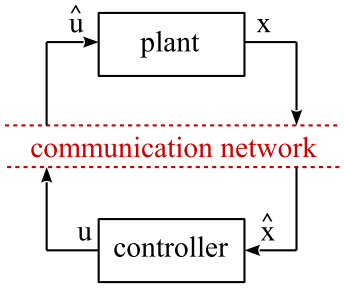

Figure 1.1: General model of a networked control system [2]

disadvantages result from network-induced imperfection, which modify the signals sent over the network. A very general model of a NCS is shown in Figure 1.1.
The network uncertainties, considered in this master's thesis, are (variable) time delays, which arise as a result of the limited transmission speed, and packet dropouts (data loss), due to the unreliability of the network. These imperfections strongly determine the modeling of the system to be controlled and make some of the conventional control approaches not applicable.
One additional challenge present in every practical implementation are disturbances. Even though extensive research which considers the above mentioned networked imperfections is available, not much of the published work deal with perturbations rejection. One group of control algorithms, so-called sliding mode techniques has been proven to efficiently achieve robustness of control systems against model uncertainties and external perturbations. However, their potential with respect to perturbed networked control system has not been yet fully exploited in literature.

1.2 Problem Statement

This master's thesis deals with single-input perturbed networked control systems that consist of a continuous-time linear plant and a discrete-time controller and is in large parts motivated by [3], [4], [5] and [6]. The outline of the thesis is as follows. In chapter 2, a general model of the considered system which incorporates the above mentioned network-induced imperfections is described. Motivated by the complexity of the derived model, a specific buffering mechanism, which allows the use of existing discrete-time sliding mode control strategies is proposed in chapter 3. Chapter 4 summarizes the theoretical background necessary for the controller design for buffered NCS. In chapter 5, the realization of the wireless NCS using an embedded system platform (Arduino board) is described and the behavior of the wireless network is examined. The proposed buffering mechanism and control approach are implemented in Simulink. In chapter 6, an illustrative example based on a model of a rotary servo

plant is used to underpin the effectiveness of the proposed control approach by means of both numerical simulations done in the simulation toolbox from [6] and experiments performed on the WNCS described in chapter 5.

2

Modeling of Networked Control Systems

Contents

In this chapter, a discrete-time model of single-input perturbed networked control system which incorporates network imperfections is derived based on [6], [7] and [8]. First, the small delay case, where the total variable transmission delay (round trip time) τ_k induced by the network is smaller than the sample time T_d, is modeled. Afterwards, this model is extended to a less limiting large delay case, which is characterized by round trip times τ_k larger than the sample time T_d.

The schematic overview of the perturbed NCS which incorporates the network imperfections is shown in Figure 2.1. The linear continuous-time plant with a single input u_k and a matched perturbation f_k is given by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}_c\boldsymbol{x}(t) + \boldsymbol{b}_s(u^*(t) + f(t)) \tag{2.1}$$

with $\boldsymbol{A}_c \in \mathbb{R}^{n\times n}$ and $\boldsymbol{b}_c \in \mathbb{R}^n$. The effect of measuring the system states $\boldsymbol{x}(t)$ with the constant sampling time T_d resulting in $\boldsymbol{x}_k$ values is represented by the *sensor* block. To transform the discrete-time control signal u_k back to the continuous-time actuator signal, a *zero-order hold* block is necessary. Variable network induced delays consisting of a sensor-to-controller delay τ_k^{sc} and a controller-to-actuator delay τ_k^{ca} are both taken

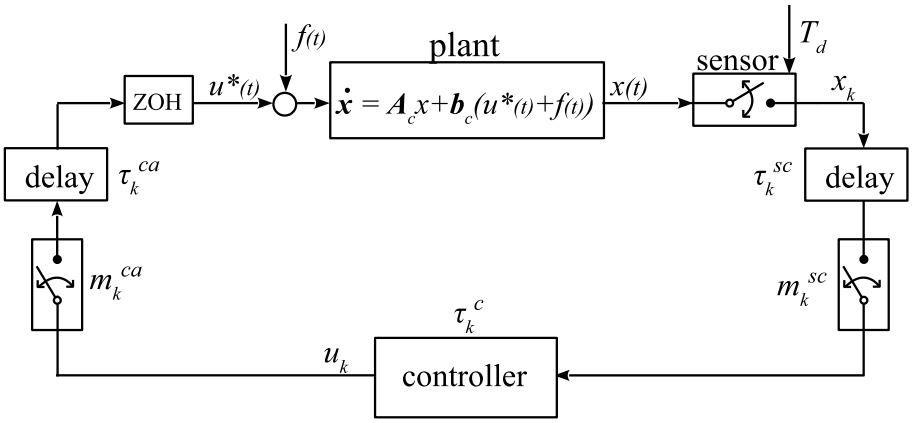

Figure 2.1: Schematic overview of the networked control system

into account by the *delay* blocks in Figure 2.1. Additionally, variable time needed for the computation of the control law is modeled by τ_k^c. Under the assumption, that a static control law is used, i.e $u_k = f(\boldsymbol{x}_k)$, the individual delays can be captured by a single delay, defined as a round trip time

$$\tau_k = \tau_k^{sc} + \tau_k^c + \tau_k^{ca} \tag{2.2}$$

Data loss (packet dropouts) which can occur in wireless networked control systems is modeled by parameters m_k^{sc} and m_k^{ca} depending on whether a packet is dropped in the sensor-to controller or in the controller-to-actuator connection. However, due to the fact that a packet dropout occurring in one of the two connections (for a static control law) leads to the same result, a single variable m_k can be used to model this network imperfection:

$$m_k = \begin{cases} 0 & \boldsymbol{x}_k \text{ and } u_k \text{ are received} \\ 1 & \boldsymbol{x}_k \text{ and/or } u_k \text{ is lost} \end{cases} \tag{2.3}$$

The equivalent simplified schematic overview of the NCS resulting from these assumptions is shown in Figure 2.2.

Furthermore, it is assumed, that the variation of the round trip time and the number of subsequent packet dropouts $\bar{\delta}$ are bounded by

$$\tau_{\min} \leq \tau_k \leq \tau_{\max}, \tag{2.4}$$

$$\sum_{l=k-\bar{\delta}}^{k} m_l \leq \bar{\delta}. \tag{2.5}$$

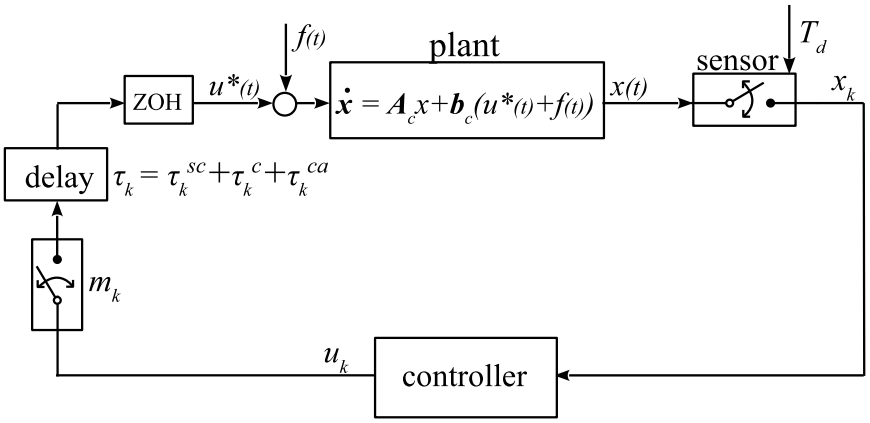

Figure 2.2: Simplified schematic overview of the networked control system

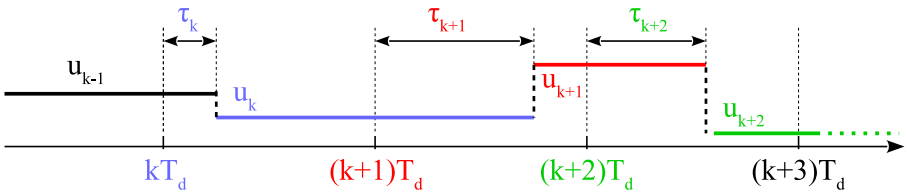

Figure 2.3: Timing diagram - small delay case [6] $\tau_k \in [0, T_d)$

2.1 Small Delay Case

In the small delay case, every control signal is delayed by the round trip time which is smaller then a sample time. Hence, only one new control input can be acquired during one sampling interval. This can be illustrated by a so-called timing diagram shown in Figure 2.3, where all labels with the same color correspond to the same k-sample.

In order to obtain a discrete-time model of the perturbed NCS, first an assumption must be made that the sample time T_d is chosen small enough to ensure that the perturbation is piece-wise constant [6], i.e.

$$f(t) = f(kT_d) = f_k, \quad kT_d \leq t \leq (k+1)T_d, \quad k \in \mathbb{N}. \tag{2.6}$$

Afterwards, the exact discretization can be applied to (2.1) resulting in

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_d \boldsymbol{x}_k + \int_0^{T_d-\tau_k} e^{\boldsymbol{A}_c s} ds \boldsymbol{b}_c u_k + \int_{T_d-\tau_k}^{T_d} e^{\boldsymbol{A}_c s} ds \boldsymbol{b}_c u_{k-1} + \boldsymbol{b}_d f_k \tag{2.7}$$

with

$$\boldsymbol{A}_d := e^{\boldsymbol{A}_c T_d} \tag{2.8a}$$

$$\boldsymbol{b}_d := \int_0^{T_d} e^{\boldsymbol{A}_c s} ds \boldsymbol{b}_c \tag{2.8b}$$

For more details on the derivation, see [6] and [8] .
The corresponding delay-free model (lifted model) equivalent to (2.7) is given by

$$\boldsymbol{\xi}_{k+1} = \hat{\boldsymbol{A}}(\tau_k)\boldsymbol{\xi}_k + \hat{\boldsymbol{b}}(\tau_k)u_k + \hat{\boldsymbol{b}}_f f_k \tag{2.9}$$

with the augmented state vector

$$\boldsymbol{\xi}_k = \begin{bmatrix} \boldsymbol{x}_k \\ u_{k-1} \end{bmatrix} \tag{2.10}$$

and matrices

$$\hat{\boldsymbol{A}}(\tau_k) = \begin{bmatrix} \boldsymbol{A}_d & \int_{T_d-\tau_k}^{T_d} e^{\boldsymbol{A}_c s} ds \boldsymbol{b}_c \\ \boldsymbol{0}_{1\times n} & 0 \end{bmatrix} \tag{2.11a}$$

$$\hat{\boldsymbol{b}}(\tau_k) = \begin{bmatrix} \int_0^{T_d-\tau_k} e^{\boldsymbol{A}_c s} ds \boldsymbol{b}_c \\ 1 \end{bmatrix} \tag{2.11b}$$

$$\hat{\boldsymbol{b}}_f = \begin{bmatrix} \boldsymbol{b}_d \\ 0 \end{bmatrix} \tag{2.11c}$$

Note that the mathematical model (2.9) is time-variant due to varying time delays τ_k.

2.2 Large Delay Case

The condition that delays are smaller than the sample time is quite limiting, since the sample time of the systems can not be arbitrarily increased. The more relevant scenario in practical applications is the case where delays due to network imperfections are larger than the sampling interval. Increasing the delay leads to a varying number of control signals which are active over one sampling interval. This effect is demonstrated in the timing diagram shown in Figure 2.4. Due to the delay τ_{k+1}, which is larger than T_d there is only one signal u_k acting between $(k+1)T_d$ and $(k+2)T_d$. However, between $(k+2)T_d$ and $(k+3)T_d$ one has to differ between three control signals u_k, u_{k+1} and u_{k+2}, which are applied to the system for a different period of time. For the purpose of modeling NCS in the large delay case, two additional variables are introduced:

$$\bar{d} := \left\lceil \frac{\tau_{\max}}{T_d} \right\rceil \tag{2.12}$$

i.e. the smallest integer larger than or equal to $\frac{\tau_{\max}}{T_d}$ and

$$\delta := \bar{d} + \bar{\delta} \tag{2.13}$$

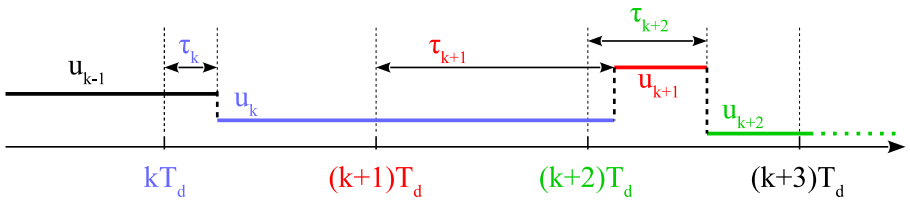

Figure 2.4: Timing diagram - large delay case [6] $\tau_k \in \left[0, \bar{d}T_d\right)$

where $\bar{\delta}$ is defined according to (2.5). Furthermore, due to varying delays larger than T_d can happen that packets do not arrive in correct order. In this case, it is preferable to implement a message rejection algorithm which, depending on the data timestamp, avoids implementation of older data.

To specify the signal $u^*(t)$ from (2.1) in the large delay case it is necessary to indicate which signals are active over each sample interval. In addition, the period of time for which a specific signal is applied to the system is relevant. For this reason, so-called arrival times $t_j^k \in [0, T_d]$ are defined:

$$\begin{aligned} t_j^k = \min \Big\{ & \max\left[0, \tau_j - (k-j)T_d\right] + m_j T_d, \\ & \max\left[0, \tau_{j+1} - (k-j-1)T_d\right] + m_{j+1} T_d, \\ & \ldots, \max\left[0, \tau_k\right] + m_k T_d, T_d \Big\} \end{aligned} \tag{2.14}$$

with $t_j^k \leq t_{j+1}^k$ and $j \in [k-\delta, k-\delta-1, \ldots, k]$. Additionally, the following relation holds for the arrival times t_j^k:

$$0 = t_{k-\delta}^k \leq t_{k-\delta+1}^k \leq \cdots \leq t_k^k \leq t_{k+1}^k = T_d \tag{2.15}$$

The case when $t_j^k = t_{k+1}^k$ corresponds to the scenario when a control signal u_j is not active meaning the possibility of message rejection is included explicitly.

The continuous-time control signal $u^*(t)$ from (2.1) is therefore given by

$$u^*(t) = u_j \quad \text{for} \quad t \in \left[kT_d + t_j^k, kT_d + t_{j+1}^k\right) \tag{2.16}$$

for $j \in [k-\delta, k-\delta-1, \ldots, k]$. Moreover, in the large delay case it is reasonable to define a vector $\boldsymbol{\theta}_k$ consisting of uncertain arrival times

$$\boldsymbol{\theta}_k = \begin{bmatrix} t_{k-\delta+1}^k & \ldots & t_k^k \end{bmatrix}^T. \tag{2.17}$$

The discrete-time model of the NCS in the large delay case can be defined as

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_d \boldsymbol{x}_k + \sum_{j=k-\delta}^{k} \int_{T_d - t_{j+1}^k}^{T_d - t_j^k} e^{\boldsymbol{A}_c s} ds \boldsymbol{b}_c u_j. \tag{2.18}$$

The equivalent delay-free model is given by

$$\boldsymbol{\xi}_{k+1} = \hat{\boldsymbol{A}}(\theta_k)\boldsymbol{\xi}_k + \hat{\boldsymbol{b}}(\theta_k)u_k + \hat{\boldsymbol{b}}_f f_k \tag{2.19}$$

with the augmented state vector

$$\boldsymbol{\xi}_k = \begin{bmatrix} \boldsymbol{x}_k & u_{k-1} & \dots & u_{k-\delta} \end{bmatrix}^T \tag{2.20}$$

and matrices

$$\hat{\boldsymbol{A}}(\boldsymbol{\theta}_k) = \begin{bmatrix} \boldsymbol{A}_d & \int_{T_d - t_k^k}^{T_d - t_{k-1}^k} e^{\boldsymbol{A}_c s} ds \boldsymbol{b}_c & \dots & \int_{T_d - t_{k-\delta}^k}^{T_d - t_{k-\delta-1}^k} e^{\boldsymbol{A}_c s} ds \boldsymbol{b}_c & \int_{T_d - t_{k-\delta+1}^k}^{T_d} e^{\boldsymbol{A}_c s} ds \boldsymbol{b}_c \\ \boldsymbol{0}_{1\times n} & 0 & \dots & 0 & 0 \\ \boldsymbol{0}_{1\times n} & 1 & \dots & 0 & 0 \\ \vdots & & \ddots & & \vdots \\ \boldsymbol{0}_{1\times n} & 0 & \dots & 1 & 0 \end{bmatrix} \tag{2.21a}$$

$$\hat{\boldsymbol{b}}(\boldsymbol{\theta}_k) = \begin{bmatrix} \int_0^{T_d - t_k^k} e^{\boldsymbol{A}_c s} ds \boldsymbol{b}_c \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.21b}$$

$$\hat{\boldsymbol{b}}_f = \begin{bmatrix} \boldsymbol{b}_d \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.21c}$$

The matrices $\boldsymbol{A}_d$ and $\boldsymbol{b}_d$ are calculated according to (2.8).

3
Buffering Mechanism

Contents

The introduction of uncertain time delays and dropouts to the NCS model leads to a complex time-variant mathematical model defined in the previous chapter, which makes the stability proof and controller design quite mathematically demanding. For this reason, a specific buffering mechanism is proposed.
The buffer implementation proposed in [6] is used as a basis of the buffering mechanism proposed in this chapter. To understand the motivation for modifying of the existing approach, the original buffer from [6] is first described. Afterwards, the adapted version of the buffered networked control system is explained.

3.1 Original Buffer Mechanism

The buffering mechanism proposed in [6] consists of the buffer implemented directly at the receiving end of the feedback channel, as shown in Figure 3.1. Its effect is defined so that it ensures a constant round trip time, which is done by introducing an additional time delay τ_k^b for each packet before forwarding it to the *zero-order hold* block. The value of the additional delay is given by

$$\tau_k^b = \bar{d}T_d - (\tau_k^{sc} + \tau_k^c + \tau_k^{ca}) \tag{3.1}$$

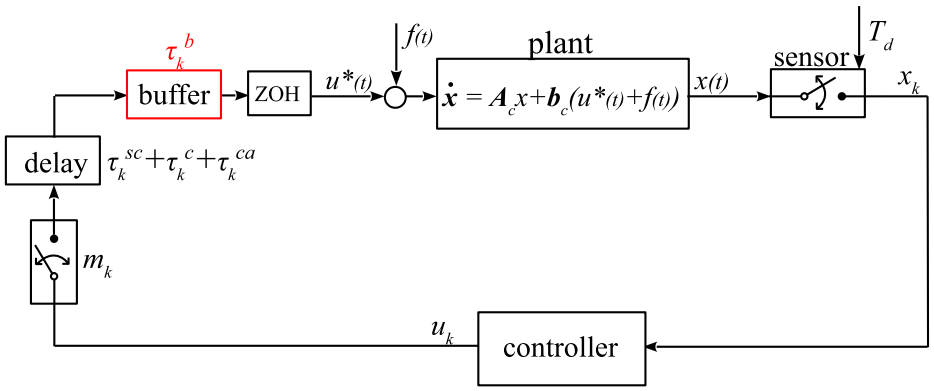

Figure 3.1: Simplified schematic overview of the buffered networked control system

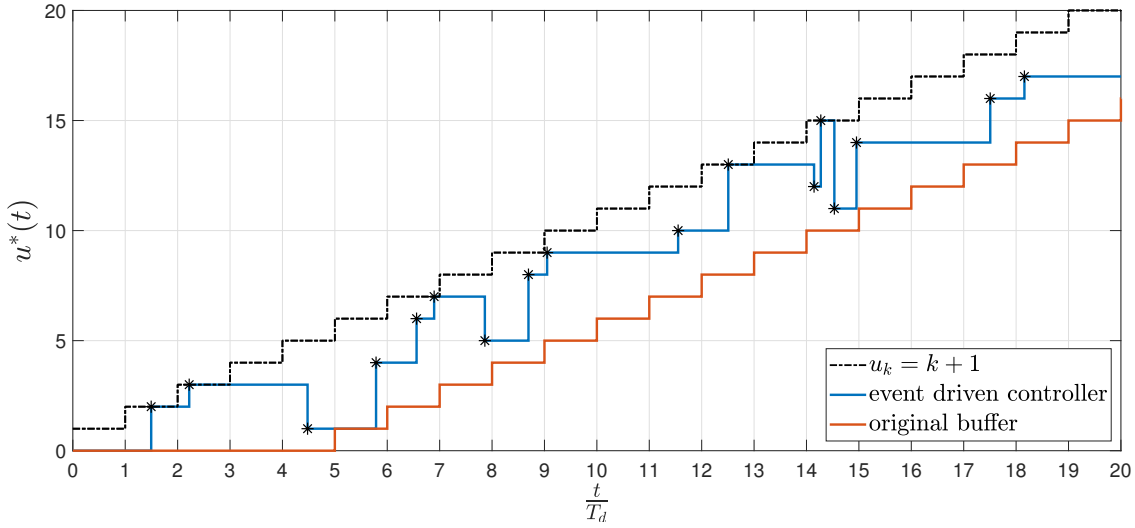

Figure 3.2: Original buffer mechanism

and is calculated based on the timestamp of the data, which is assumed to be attached to each message. This approach ensures the constant round trip time which is equal to the maximal round trip time $\bar{d}T_d$. Figure 3.2 demonstrates the effect of the original buffer mechanism for $\bar{d} = 5$. The controller $u_k = k + 1$ is chosen for the sake of clarity and is plotted with a black line whereas the black stars correspond to the time instants at which control signals are received and applied to the system. The blue line corresponds to the actuator signal in the case of an event-driven controller. The effect of the original buffer implementation is demonstrated with a red line, which corresponds to the control signal u_k always delayed by the maximal round trip time $5T_d$.

On the one hand, the great advantage of this approach is that the resulting model of NCS is due to the constant round trip time time-invariant. This simplifies the stability proof and controller synthesis because it allows the use of classical methods such as

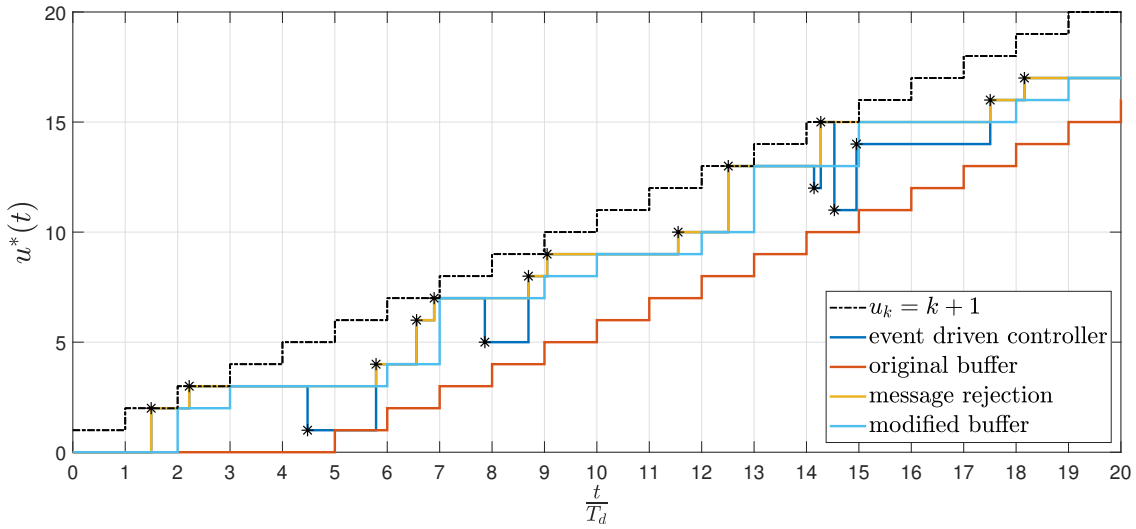

Figure 3.3: Modified buffer mechanism

eigenvalue placement of the closed loop system directly. On the other hand, even though is the control signal, e.g., $u_k = 2$ available already in the second sample interval, it is not sent to the plant before $t = 6T_d$. This disadvantage is especially relevant for systems where a few data packets are delayed by a longer time delay (e.g. in case of retransmissions for lost packets) in comparison to most messages. This can be avoided either by a specific mechanism (e.g. to treat all packets arrived after a specified time delay limit as a message dropout) or by modifying the buffering mechanism.

3.2 Modified Buffering Mechanism

In the first step, the message rejection mechanism is implemented, which ensures that no older control data is sent to the plant. This is demonstrated by the yellow line from Figure 3.3 which is strictly increasing. The proposed buffer mechanism introduces the additional delay which is calculated with respect to the time instant at which a control signal arrives to the actuator:

$$\begin{aligned}
&\text{if} \quad 0 \le \tau_k^{sc} + \tau_k^{c} + \tau_k^{ca} \le T_d \quad \text{then} \quad \tau_k^{b} = T_d - (\tau_k^{sc} + \tau_k^{c} + \tau_k^{ca}) \quad \rightarrow \quad \tau_k = T_d \\
&\text{if} \quad 0 \le \tau_k^{sc} + \tau_k^{c} + \tau_k^{ca} \le 2T_d \quad \text{then} \quad \tau_k^{b} = 2T_d - (\tau_k^{sc} + \tau_k^{c} + \tau_k^{ca}) \quad \rightarrow \quad \tau_k = 2T_d \\
&\qquad\qquad\qquad\qquad \vdots \\
&\text{if} \quad 0 \le \tau_k^{sc} + \tau_k^{c} + \tau_k^{ca} \le \bar{d}T_d \quad \text{then} \quad \tau_k^{b} = \bar{d}T_d - (\tau_k^{sc} + \tau_k^{c} + \tau_k^{ca}) \quad \rightarrow \quad \tau_k = \bar{d}T_d
\end{aligned} \tag{3.2}$$

This implies that if a control signal arrives between two sample times, it becomes active at the beginning of the next sample time. In case of more than one control signal

arriving between kT_d and $(k+1)T_d$, the most resent control signal is applied whereas the remaining signals are rejected. The actuator signal $u^*(t)$ generated in described scenario is shown in Figure 3.3 in light blue. The signal is strictly increasing and change of the value occurs only at the beginning of the sample intervals. The time difference between the black signal and the signal resulting from this buffer implementation is not constant as it is the case in the previously described approach, but it is an integer multiple of T_d which varies between zero and $\bar{d}T_d$. Furthermore, there is always only one signal acting on the plant over each sample interval.
The buffered networked control system can, in this case, be represented by the discrete-time model

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_d \boldsymbol{x}_k + \boldsymbol{b_d} \left(u_{k-d_k} + f_k\right) \tag{3.3}$$

where d_k represents an unknown input delay with

$$d_k \in \{0, 1, \ldots, \bar{d}\}. \tag{3.4}$$

The matrices $\boldsymbol{A}_d$ and $\boldsymbol{b}_d$ are calculated according to (2.8).

4
Controller Design

Contents

In this chapter, the design of the discrete-time controller based on Integral Sliding Modes for the described NCS model is presented.

4.1 Integral Sliding Mode Control

Sliding mode techniques represent a very powerful group of control algorithms originated from the theory of Variable Structure Systems (VSS). These systems consist of different subsystems for which a region of validity and corresponding switching rules are defined, resulting in control actions which are a discontinuous function of system states [9]. The concept of Sliding Mode Control (SMC) lies in the introduction of a specific so-called sliding variable, whose design is crucial for the performance of the system. The SMC process includes typically 2 phases: the steering of the system trajectories to the chosen sliding manifold, where a sliding variable is equal to zero and maintaining of the states sliding along the switching plane to the origin. In this context, one differs between the reaching phase, which is not characterized by an insensitivity property, and a sliding

phase, during which the system response becomes insensitive to matched perturbations, i.e. perturbations that act in the same channel as the input of the system.
Integral Sliding Modes (ISMC) are a special type of conventional sliding mode techniques which are motivated by the need for elimination of the reaching phase. This means that the sliding variable is equal to zero from the very beginning and kept at zero thereafter, which ensures the robustness of the system from the initial time instant [10]. The control design is made under the assumption that a nominal controller can be designed, which ensures the stability and desired performance of the ideal system. The sliding mode part of the control law for uncertainty and disturbance rejection is added to the nominal controller:

$$u_k = u_k^N + u_k^S \tag{4.1}$$

4.2 Nominal Controller

The nominal (ideal) part of the control law guarantees the asymptotic stability of the ideal closed loop system. Since the focus of this master's thesis lies in the application of the sliding mode techniques, the nominal controller is designed based on the existing stability analysis and controller synthesis results proposed in [7].

Remark 4.1:

The presence of the proposed buffering mechanism from chapter 3 is not taken into account in the controller synthesis of the nominal control law. The designed state-feedback controller guarantees the global asymptotic stability of the closed loop NCS for *any* delay τ_k from the interval $[\tau_{\min}, \tau_{\max}]$.

In the following theorem a finite number of linear matrix inequality (LMI) conditions for the state-feedback controller that depends only on the state $\boldsymbol{x}_k$ of the form

$$u_k^N = -\bar{\boldsymbol{K}}\boldsymbol{\xi}_k = -\boldsymbol{K}_x\boldsymbol{x}_k \tag{4.2}$$

with

$$\bar{\boldsymbol{K}} = \begin{bmatrix} \boldsymbol{K}_x & \boldsymbol{0}_{1\times\delta} \end{bmatrix} \tag{4.3}$$

and δ from (2.13) is defined.

Theorem 4.2: Nominal Control Law[7]

Consider the nominal NCS model

$$\boldsymbol{\xi}_{k+1} = \hat{\boldsymbol{A}}(\boldsymbol{\theta}_k)\boldsymbol{\xi}_k + \hat{\boldsymbol{b}}(\boldsymbol{\theta}_k)u_k \tag{4.4}$$

with matrices $\hat{\boldsymbol{A}}(\boldsymbol{\theta}_k)$, $\hat{\boldsymbol{b}}(\boldsymbol{\theta}_k)$ from (2.21a) and (2.21b) respectively and uncertain parameters $\boldsymbol{\theta}_k$ (2.17). Based on the real Jordan form of the matrix $\boldsymbol{A}_c$ a generic model of the form

$$\boldsymbol{\zeta}_{k+1} = \left(\boldsymbol{F}_0 + \sum_{i=1}^{\xi} \alpha_i(\boldsymbol{\theta}_k)\boldsymbol{F}_i\right)\boldsymbol{\zeta}_k + \left(\boldsymbol{G}_0 + \sum_{i=1}^{\xi} \alpha_i(\boldsymbol{\theta}_k)\boldsymbol{G}_i\right)u_k \tag{4.5}$$

can be derived. Moreover, define the set that contains the bounds of the uncertain parameters

$$\boldsymbol{\Theta}_k = \left\{\boldsymbol{\theta}_k \in \mathbb{R}^{\delta} \middle| t_j^k \in [t_{j,\min}, t_{j,\max}], 1 \leq j \leq \delta, 0 \leq t_{k-\delta+1}^k \leq \cdots \leq t_k^k \leq T_d\right\} \tag{4.6}$$

and gives rise to the set of matrices

$$\mathscr{FG} = \left\{\left(\boldsymbol{F}_0 + \sum_{i=1}^{\xi} \alpha_i(\boldsymbol{\theta}_k)\boldsymbol{F}_i\right), \left(\boldsymbol{G}_0 + \sum_{i=1}^{\xi} \alpha_i(\boldsymbol{\theta}_k)\boldsymbol{G}_i\right) \middle| \boldsymbol{\theta}_k\right\} \tag{4.7}$$

which is a subset of the convex hull $\mathrm{co}(\mathscr{H}_{FG})$

$$\mathscr{H}_{FG} = \left\{\left(\left(\boldsymbol{F}_0 + \sum_{i=1}^{\xi} \alpha_i(\boldsymbol{\theta}_k)\boldsymbol{F}_i\right), \left(\boldsymbol{G}_0 + \sum_{i=1}^{\xi} \alpha_i(\boldsymbol{\theta}_k)\boldsymbol{G}_i\right)\right) : \alpha_i \in \{\underline{\alpha}_i, \bar{\alpha}_i, i = 1, 2, ..., \xi\}\right\} \tag{4.8}$$

with $\underline{\alpha}_i$ and $\bar{\alpha}_i$ being the minimum and maximum value of $\alpha_i(\boldsymbol{\theta}_k)$ respectively. If there exist symmetric positive definite matrices

$$\boldsymbol{Y}_j \in \mathbb{R}^{(n+\delta)\times(n+\delta)}, \tag{4.9a}$$

matrices

$$\bar{\boldsymbol{Z}} \in \mathbb{R}^{1\times n} \tag{4.9b}$$

$$\boldsymbol{X}_j = \begin{pmatrix} \bar{\boldsymbol{X}}_1 & 0 \\ \bar{\boldsymbol{X}}_{2,j} & \bar{\boldsymbol{X}}_{3,j} \end{pmatrix} \quad \text{with} \quad \bar{\boldsymbol{X}}_1 \in \mathbb{R}, \bar{\boldsymbol{X}}_{2,j} \in \mathbb{R}^{\delta\times n}, \bar{\boldsymbol{X}}_{3,j} \in \mathbb{R}^{\delta\times\delta} \tag{4.9c}$$

for $j = 1, 2, \ldots, 2^{\zeta}$ and a scalar

$$0 \leq \gamma < 1 \tag{4.9d}$$

that satisfy

$$\begin{bmatrix} \boldsymbol{X}_j + \boldsymbol{X}_j^T - \boldsymbol{Y}_j & \boldsymbol{X}_j^T \boldsymbol{H}_{F,j} - \begin{pmatrix} \bar{\boldsymbol{Z}} & 0 \end{pmatrix}^T \boldsymbol{H}_{G,j}^T \\ \boldsymbol{H}_{F,j} \boldsymbol{X}_j - \boldsymbol{H}_{G,j} \begin{pmatrix} \bar{\boldsymbol{Z}} & 0 \end{pmatrix} & (1-\gamma)\boldsymbol{Y}_l \end{bmatrix} > 0 \tag{4.10}$$

for $j, l \in \{1, 2, \ldots, 2^\zeta\}$ and ζ being the number of time-varying functions in NCS model, then the closed loop NCS with $\bar{\boldsymbol{K}} = \bar{\boldsymbol{Z}} \bar{\boldsymbol{X}}^{-1}$ is globally asymptotically stable.

For more details about the Jordan form and the proof of the theorem see [7] and appendix B in [8].

The use of a static control law of the form (4.2) allows simplification of the NCS by considering the sum of all network induces delays as one single delay without additional restrictive assumptions regarding the sensor to controller time delays. Furthermore, possible deadlocks which can occur when using a dynamic controller are avoided [7], [8].

Note that this theorem can be extended to the NCS model with time-varying sampling instants and possible packet dropouts as proposed in [7]. In addition, with appropriate adaptation of the matrices (4.9b) and (4.9c), an extended (dynamic) state feedback controller can be derived.

4.2.1 Computation of the controller

It was shown that the controller synthesis problem can be expressed through LMIs, which are essentially convex constraints and as such can be solved efficiently using existing algorithms and software (see e.g [11]). In this master thesis, a free Matlab toolbox YALMIP (**Y**et **A**nother **LMI** **P**arser) [12] with a solver MOSEK is used for the computation of the nominal controller. This toolbox was first released in 2001 and by now supports very wide range of optimization problems and suitable solvers and because of this can be used as a very powerful computational tool for solving many optimization problems which arise in control theory.

In order to determine the gain matrix $\bar{\boldsymbol{K}}$ (4.3), the LMIs from theorem 4.2 should be implemented in Matlab. The optimization variables can be defined by the command `sdpvar`. It is important to mention that the square matrices defined by this command are by default symmetric. If this should not be the case, additional argument `'full'` can be used. In addition, YALMIP submits an error when using strict inequalities which can be avoided by defining non-strict inequalities with a specific margin. Here, the margin is set to `eps*eye()`), with `eps` being the distance from 1.0 to the next larger double-precision number equal to $2.2204 \cdot 10^{-16}$ in Matlab and `eye()` being an

identity matrix of the appropriate size.

The nominal controller from theorem 4.2 for any NCS problem can be designed with a following piece of code:

```
%% Define Optimization Problem -> YALMIP
%-----------------------------------------------
% n - nr. of system states
% m - nr. of inputs
% zeta - nr. of time-varying functions alpha
% d - maximal delay+dropouts
% gamma - scalar used in theorem
%-----------------------------------------------
Y = cell(2^zeta,1);
Z = sdpvar(m,n,'full');
X1= sdpvar(n,n,'full');
X = cell(2^zeta,1);

for k = 1:2^zeta
  Y{k} = sdpvar(n+d*m,n+d*m);
  X{k} = [         X1           ,        zeros(n,d*m)
           sdpvar(d*m,n,'full'),sdpvar(d*m,d*m,'full')];
end

constr= [];
%-------------------------------------------------------
for j  = 1:2^zeta
  constr = [constr,Y{j}>= eps*eye(n+d*m)];
  for l = 1:2^zeta
    M{j,l}=[ X{j}+(X{j}).'-Y{j} , ...
        (X{j}).'*(Hf{j}).'-[Z,zeros(1,d)]'*(Hg{j}).';
             Hf{j}*X{j}-Hg{j}*[Z,zeros(1,d)] , ...
       (1-gamma)*Y{l}];
    constr = [constr,M{j,l}>= eps*eye(2*(n+d*m))];
  end
end
%-------------------------------------------------------
options = sdpsettings('verbose',0,'solver',mosek);
diagnostics = optimize(constr,[],options);

K = value(Z)*inv(value(X1));
```

Note that the number of LMI conditions to be solved for controller synthesis increases exponentially with the number of time-varying functions $\alpha(\boldsymbol{\theta}_k)$ which depends on the size of the continuous-time system matrix $\boldsymbol{A}_c$ and parameter δ defined in (2.13).

4.3 Sliding Mode Controller

The sliding mode control law used in this master's thesis is based on the discrete-time equivalent of the super-twisting algorithm with matching approach proposed in [13], [14] and adapted in [6]. In order to use this control algorithm in the NCS model (3.3) with the modified version of the buffer, an additional adaptation with respect to the unknown input delay is required.

4.3.1 Unknown Input Delay Compensation

Due to the fact that in the modified version of the buffer proposed in the chapter 3 the resulting model of the NCS (3.3) is still time-variant comparing to the time-invariant representation of NCS in [6], the problem of the unknown time-varying input delay must be addressed first. In order to design a sliding variable which is independent of the unknown delay, it is necessary to compensate the input-delay which can be done in the sensor node. In addition to the measuring of the state signals, the variable s_k should be calculated and sent together with the sensor signals to the controller:

$$s_k = \boldsymbol{m}^T \boldsymbol{x}_k - u_{k-d_k-1} \tag{4.11}$$

where u_{k-d_k-1} is the control signal acting on plant in the previous step and the vector $\boldsymbol{m}^T$ can be chosen so that

$$\boldsymbol{m}^T \boldsymbol{b}_d = 1 \tag{4.12}$$

holds.

4.3.2 Design of the Sliding Variable

The system states $\boldsymbol{x}_k$ and the variable s_k are sent to the controller where the discrete time integral sliding variable is defined:

$$\sigma_k = s_k + u_{k-1} + w_k \tag{4.13}$$

Considering the forward increment of (4.11) and using (4.12), (4.1) the expression for the sliding variable in the next step can be obtained:

$$\sigma_{k+1} = \boldsymbol{m}^T \boldsymbol{x}_{k+1} - u_{k-d_k} + u_k + w_{k+1} \tag{4.14}$$

$$= \boldsymbol{m}^T \boldsymbol{A}_d \boldsymbol{x}_k + f_k + u_k^N + u_k^S + w_{k+1} \tag{4.15}$$

Defining w_{k+1} so that all known quantities in (4.15) are cancelled out, i.e.

$$w_{k+1} = -\boldsymbol{m}^T \boldsymbol{A}_d \boldsymbol{x}_k - u_k^N \tag{4.16}$$

gives

$$\sigma_{k+1} = u_k^S + f_k \tag{4.17}$$

which is independent of the unknown input delay. The forward increment of the sliding variable now has the appropriate form so that the existing approaches for discrete-time super-twisting algorithms can be used.

Remark 4.3:

If the forward increment of the sliding variable is achieved to be equal to zero, it follows from (4.17) that

$$u_k^S = -f_k \tag{4.18}$$

holds. Substituting u_k^S in the NCS model (3.3) with (4.2) leads to the plant dynamics

$$\boldsymbol{x}_{k+1} = \left(\boldsymbol{A}_d \boldsymbol{x}_k - \boldsymbol{b}_d u_{k-d_k}^N\right) + (f_k - f_{k-d_k}) \tag{4.19}$$

with d_k being the unknown input delay (3.4). Due to the perturbation $(f_k - f_{k-d_k})$ acting on the plant, it is reasonable to consider the worst-case scenario regarding the change rate L_f:

$$L_f = \max \left| \frac{f_k - f_{k-\delta}}{T_d} \right| \tag{4.20}$$

Theorem 4.4: Sliding Mode Control Law [6]

Consider the discrete-time representation of the networked control system with buffering mechanism given in (3.3) with a sampling time T_d. Moreover, consider the sliding variable σ_k in (4.13) with $\boldsymbol{m}^T$ defined in (4.12), s_k defined in (4.11) and w_{k+1} given by (4.16). Suppose that for a specific value of the change rate L_f (4.20) the parameters α and β are chosen by a well-established super-twisting parameter setting

$$\alpha = 1.5\sqrt{\frac{L_f}{T_d}} \qquad \beta = 1.1\frac{L_f}{T_d}. \tag{4.21}$$

Furthermore, for $p^{(1)}, p^{(2)}$ and $s^{(1)}(\sigma_k), s^{(2)}(\sigma_k)$ the following expressions hold

$$p^{(1)} = -\frac{\alpha}{2} - \sqrt{\frac{\alpha^2 - 4\beta}{4}} \qquad p^{(2)} = -\frac{\alpha}{2} + \sqrt{\frac{\alpha^2 - 4\beta}{4}} \tag{4.22}$$

$$s^{(1)}(\sigma_k) = p^{(1)}|\sigma_k|^{-\frac{1}{2}} \qquad s^{(2)}(\sigma_k) = p^{(2)}|\sigma_k|^{-\frac{1}{2}} \tag{4.23}$$

$$\tag{4.24}$$

Using eigenvalue mapping with matching approach leads to

$$q_k^{(1)} = \begin{cases} e^{s^{(1)}(\sigma_k)T_d} & \sigma_k \neq 0 \\ 0 & \sigma_k = 0 \end{cases} \qquad q_k^{(2)} = \begin{cases} e^{s^{(2)}(\sigma_k)T_d} & \sigma_k \neq 0 \\ 0 & \sigma_k = 0 \end{cases} \tag{4.25}$$

If a sliding mode controller

$$u_k^S = \sigma_k + \tilde{l}_k + \nu_k \tag{4.26}$$

with

$$\nu_{k+1} = \nu_k + l_k \sigma_k \tag{4.27}$$
$$\tilde{l}_k = q_k^{(1)} + q_k^{(2)} - 1 \tag{4.28}$$
$$l_k = \frac{1}{T}\left(\tilde{l} - q_k^{(1)} q_k^{(2)}\right) \tag{4.29}$$

is used, than the plant states are ultimately bounded.

For more details about the algorithm and the proofs see [6], [13] and [15].

Remark 4.5:

The initial value for w_k should be determined based on the requirement $\sigma_0 = 0$ (sliding phase starting from the initial time instance):

$$w_0 = -\boldsymbol{m}^T \boldsymbol{x}_0. \tag{4.30}$$

5 Wireless Networked Control System

Contents

In this chapter, a realization of the networked control system with a continuous-time plant and discrete-time controller connected via a wireless communication network is proposed. First, the hardware components and required support packages used for the practical set up is described. Furthermore, a strategy used for estimation of the maximal network induced delay necessary for the controller synthesis from chapter 4 is presented. As a result, a system which can be used to practically examine the efficacy of the proposed buffering mechanism and control laws is developed.

5.1 Wireless Communication and Network Protocol

Different wireless technologies can be used for data transmission in wireless NCSs such as Bluetooth, Zigbee, WLAN TCP/IP and WLAN UDP/IP. In this master's thesis the communication within the control loop is completed using a so-called personal or mobile hotspot, where a smartphone is turned into a Wi-Fi hotspot in order to create a Local Area Network (LAN). For the choice of the network protocol, the protocols Transmission Control Protocol (TCP) and User Datagram Protocols (UDP) are compared with respect to their use in the networked control systems. Though the TCP on the one hand provides assured delivery and reliability, sending acknowledgment signals, possible re-transmissions and error-detection lengthen the latency and lead to longer network delays. On the other hand, the UDP prioritizes time over reliability since it permits packets dropouts rather than re-transmit them again which makes this network protocol more efficient in terms of both latency and bandwidth. Furthermore, the round trip time can be even 2 times or more as long when using the TCP comparing to UDP [16], which would significantly increase the number of LMIs and computation time needed for the controller synthesis of the nominal control law (4.2). Due to these reasons, the User Datagram Protocol (UDP) is chosen for the realization of the WNCS.

5.2 Components of the WNCS

Due to extensive documentation and various support packages for Arduino hardware, also with respect to the WiFi communication, an Arduino board is used to achieve the wireless communication between the plant and controller as shown in Figure 5.1. On the plant side, the Arduino board which is connected to the WiFi hotspot shared by the smartphone, can be directly or over a PC connected to a physical system. Since the focus of this master's thesis lies in the controller synthesis for WNCS and the realization of WNCS to further explore the real network imperfections and limitations regarding these, the transfer function implemented on the Arduino board instead of a real system is used. The proposed control law is implemented in Simulink on the PC2 (controller side) which is connected to the WiFi hotspot. In the next sections, the necessary configurations for the plant and controller sides are explained thoroughly.

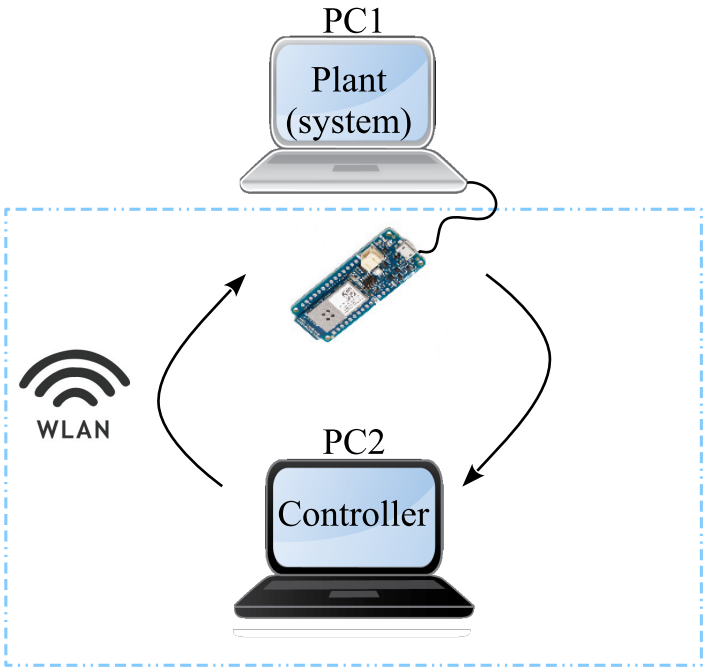

Figure 5.1: Components of the WNCS

5.3 Plant Side

5.3.1 Arduino Hardware

In order to use WiFi on the Arduino hardware in Matlab, one can choose between a WiFi Shield or ESP8266 (low-cost Wi-Fi microchip) which both should be first connected to one of the supported Arduino boards (Arduino Uno, Arduino Due etc.) and Arduino MKR1000 which has an on-board WiFi chip. The main properties regarding the choice of the hardware are summarized in Table 5.1 [1]. On account of its properties regarding the maximal number of connection (in order not to limit the setup only to single-input or small spatially distributed WNCS with max. 2 inputs) and the minimal sample time (to keep the round trip time as small as possible) the Arduino MKR1000, shown in Figure 5.2, is chosen. Furthermore, a WiFi library 101 which allows Arduino MKR1000 board to connect to the internet wirelessly should be installed (Arduino IDE>Tools>Manage Libraries).

5.3.2 Matlab/Simulink with Arduino Hardware

In order to use Matlab/Simulink to communicate with the Arduino board to create and run Simulink models on MKR1000, support packages are necessary. To find and

[1]The Matlab related information were found on `https://de.mathworks.com`. For all relevant URLs see appendix B

Target hardware	ESP08266	WiFi Shield	MKR1000
Min. sample time	0.01s	0.000001s	0.000001s
Max. Nr. of received Bytes	40	64	1400
Max. Nr. of connections - WiFi UDP	1	2	4
Max. Nr. of connections - WiFi TCP	1	2	7

Table 5.1: Comparison of Matlab supported hardware

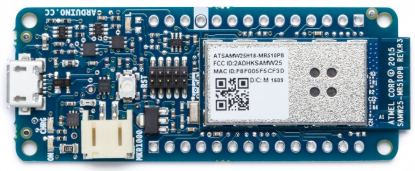

Figure 5.2: Arduino MKR1000 Board (from `https://store.arduino.cc/arduino-mkr1000-wifi`)

install add-ons, open the Add-On Explorer in Matlab, which can be found in the Home tab in the Environment section (the Add-Ons icon), see Figure 5.3. After installing the support packages and connecting the Arduino board with USB cable, the following message should appear on the command window

```
Arduino MKR1000 detected.
This device is ready for use with Matlab Support Package for Arduino
Hardware.  Get started with examples and other documentation.
This device is ready for use with Simulink Support Package for Arduino
Hardware.  Get started with  examples and other documentation.
```

In the next step, the Simulink model and network settings should be configured, which can be done in the **Configuration Parameters** dialog (see Figure 5.4). For detailed explanation, run the command

```
helpview(fullfile(codertarget.internal.arduinoert.getDocRoot,'ug',
'configure-network-settings-for-arduino-wifi-shield.html'),'-helpbrowser')
```

by entering it in the Matlab Command Window, which will open a documentation page with examples. From this point on, a Simulink model can be built using blocks from **Simulink Library Browser**. For this master's thesis relevant Arduino blocks can be found in Figure 5.5. To connect the Arduino board to in Figure 5.4 defined network and run the model, click on the **Build, Deploy & Start** button on the toolbar.

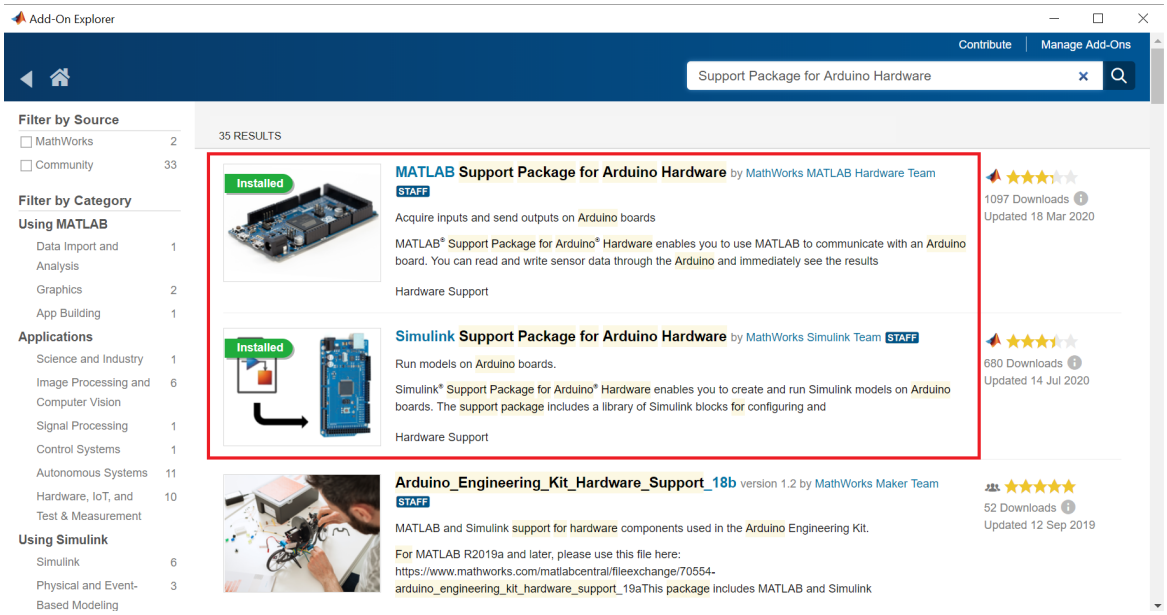

Figure 5.3: Add-On Explorer window

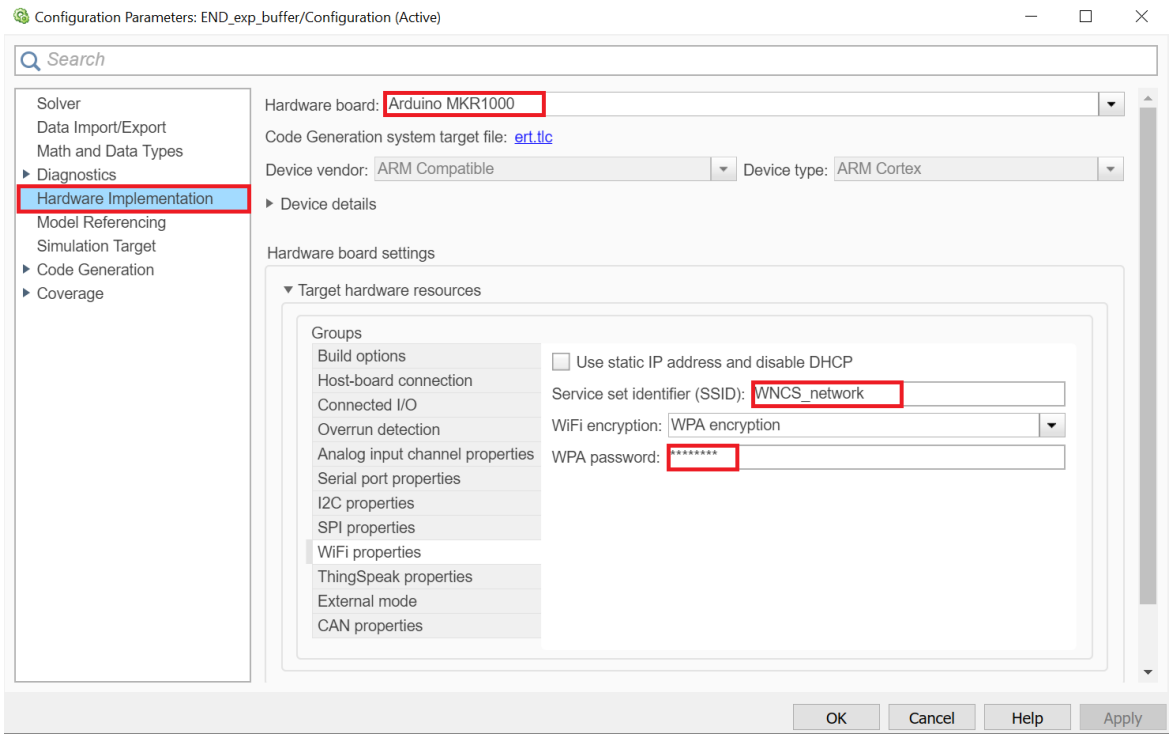

Figure 5.4: Configuration Parameters dialog

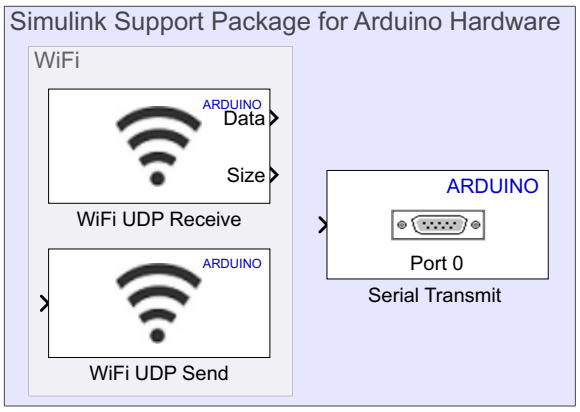

Figure 5.5: Simulink blocks - plant side

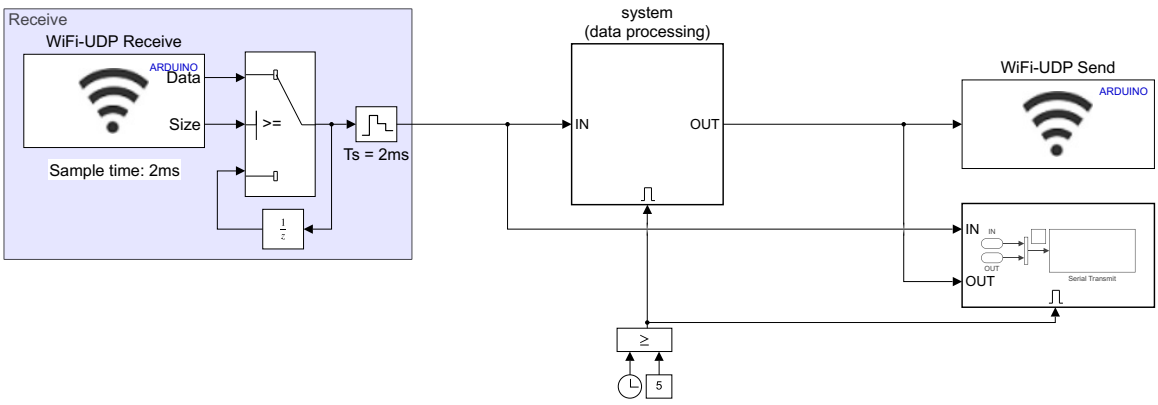

Figure 5.6: Simulink Model of the Plant Side

5.3.3 Simulink Model of the Plant Side

The Simulink model of the plant side for a general system can be found in Figure 5.6. The *WiFi UDP Receive* block is used to receive data with a sample time (interval at which block reads data from sending host) of 2ms. In the **Block Parameters** dialog of this block, tha Data Size of the UDP message can be chosen. Due to defined rules for the output of this block[2], additional switch logic is used. This way, it is possible to define the sample time lower than the sample time of the system to be able to receive more data packets during one period. In case no new data is received or the length of received UDP message is smaller than the defined Data Size, the previously obtained message is repeated for the next 2ms until the next reading of the data. The remaining configuration of this block depends on the remote host and is explained after the controller side model of the WNCS is described (see section 5.5). In the subsystem **System (data processing)** (Figure 5.4) the physical system should be connected.

[2]see `https://de.mathworks.com/help/supportpkg/arduino/ref/wifiudpreceive.html?s_tid=srchtitle` > Outputs for Received Data

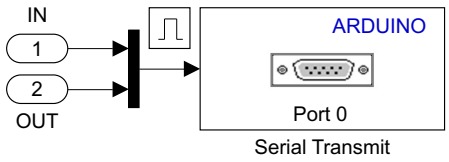

Figure 5.7: Subsystem to transmit data to serial monitor

To send data to the remote host, the *WiFi UDP send* block can be configured with parameters depending on the remote host (see section5.5). The received and sent data can be accessed and used for further data processing (e.g for plotting) with a *Serial Transmit* block which is here realized with an enabled subsystem (see Figure 5.7). The reason for using the *Enable* block in both system and serial transmit subsystem is to leave enough time from the moment when the model is ready to run and activation of the system. This way all data can be properly saved without possible missing of first (for control very relevant) values. Saving the data while the model is running is made with an open source software named Putty [3]. To automatize the process of logging and saving data, a shortcut to the .exe file is created, which can generate a .txt file with 1-click containing all the data sent to the block *Serial Transmit*. This file can be loaded to Matlab and used to, e.g., determine the round trip time or plot the signals. The Putty relevant URLs can be found in appendix B.4.

5.4 Controller Side

For the realization of the controller side, PC2 should be connected to the same WiFi hotspot as the Arduino. In order to make the wireless communication between PC1 and PC2 possible, the Simulink model should include blocks which can receive and send data to the Arduino board. These blocks can be found in the DSP System Toolbox in **Simulink Library Browser** (library sinks and sources) and are shown in Figure 5.8.

5.4.1 Simulink Model of the Controller Side

The Simulink model of the controller side can be seen in Figure 5.9. The sample time of the *UDP Receive* block is chosen to be two times larger (4ms) as the receiving sample time of the *WiFi UDP Receive* block. It is realized with the same switching logic as the receiving block on the plant side, with the difference that a value received with the status port (values 1 or 0) is used as criteria for passing first input. With this simple sampling scheme no clock synchronization is required. After processing

[3] `https://www.putty.org/`

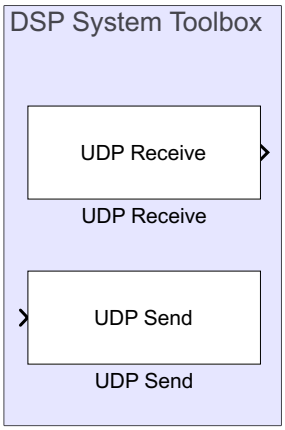

Figure 5.8: Simulink blocks - controller side

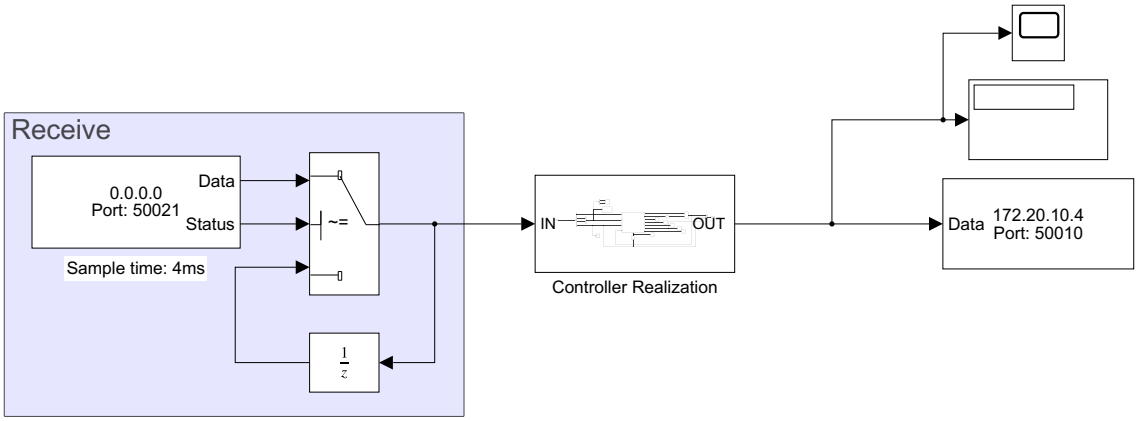

Figure 5.9: Simulink Model of the Controller Side

of the received data (controller Realization subsystem), UDP packages are sent to the Arduino board via the *UDP Send* block. The remaining configuration properties of the blocks can be found in the following section.

5.5 Communication between Plant and Controller Side

Through the proper configuration properties of each Send and Receive block, the communication between MKR1000 (plant side) and PC2 (controller side) can be established. For that it is necessary to define the IP addresses and ports in the blocks. Note that the IP address of the Arduino WiFi hardware and the IP address of the computer must be in the same range so they can communicate with each other. The IP Address of the PC2 can be determined by opening the **Command Prompt** (type `cmd` in Start) and typing `ipconfig` (here 172.20.10.2). As soon as an IP address is

	Arduino board plant side		PC2 controller side
IP address	172.20.10.4		172.20.10.2
Port	50020	$\longrightarrow$	50021
Port	50010	$\longleftarrow$	-1

Table 5.2: Ports and IP addresses of the Send and Receive blocks

assigned to the Arduino board on the plant side, an additional variable (name-of-the-simulink-model_IPaddress) with the corresponding IP address is defined in the workspace. For the configuration of the blocks, see Table 5.2. In order to specify the parameters for, e.g., *WiFi UDP Send* block on the plant side (shown in Figure 5.10), the local port should be equal to the port from the column regarding the sending side (Arduino), in the row with $\longrightarrow$ (Arduino sends to PC2). The remote address denotes the address of the receiving device (PC2) and remote IP port corresponds to the port of the PC2 in the row denoting sending data from Arduino to PC2. The rest of the blocks can be configured analogously.

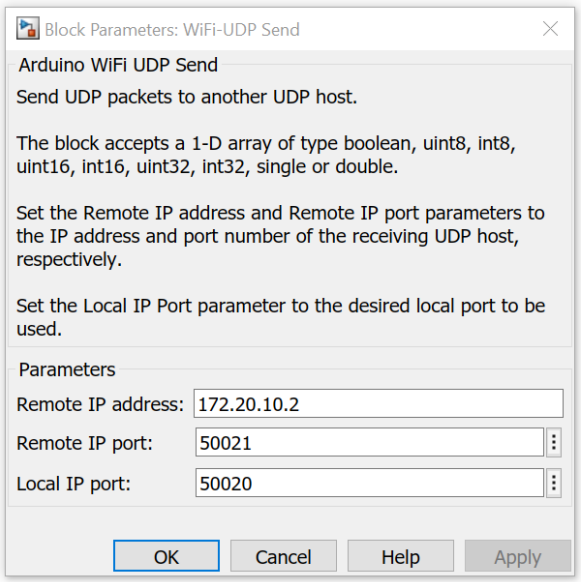

Figure 5.10: Block Parameters: WiFi-UDP Send

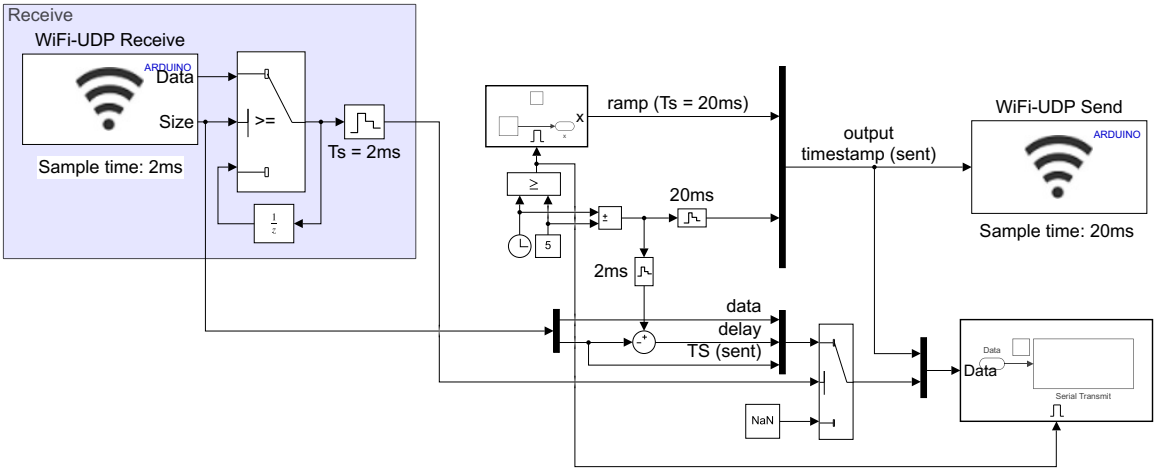

Figure 5.11: Simulink model for RTT estimation - plant side

5.6 Round Trip Time Estimation

The property of the network crucial for the controller synthesis is the round trip time (RTT). In this master's thesis the practical embedded implementation for RTT estimation include sending integer numbers from 1 to 250 of the data type *double* with a timestamp identifying when a certain number is sent. The data is sent from the plant to the controller with a sample rate of 20ms, where it is directly forwarded back to the plant side without additional processing. As soon as the data is received on the plant side, the total delay is determined by subtracting the timestamp of the sent data from the current time and sent to Putty. It is important to mention, that using the timestamp option in the serial monitor to determine the RTT can lead to incorrect results for smaller values of the sample time. Hence, the timestamps and delays are determined directly in Simulink models using clock sources. The Simulink models of the plant and controller side can be seen in Figures 5.11 and 5.12.
At the beginning of every experiment the internet speed is determined with a mobile app SPEEDTEST[4], to make sure that the properties of the network stay the same as much as possible. All the experiments were done when the internet speed is close to the speed determined in this experiment (see Figure 5.13). The sent and received data with corresponding round trip delay is shown in Figure 5.14. For a better overview, the results are plotted for $1s$ as well (see Figure 5.15). The RTT is found to be mostly between $1T_d$ and $3T_d$ in most of the experiments and no dropouts occurred. Since the UDP in general allows message drops as explained in section 5.1, an experiment with 1 dropout (which is the maximal number of dropouts determined) is plotted.

[4] `https://www.speedtest.net/apps`

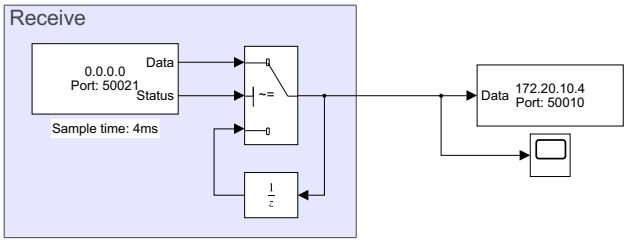

Figure 5.12: Simulink model for RTT estimation - controller side

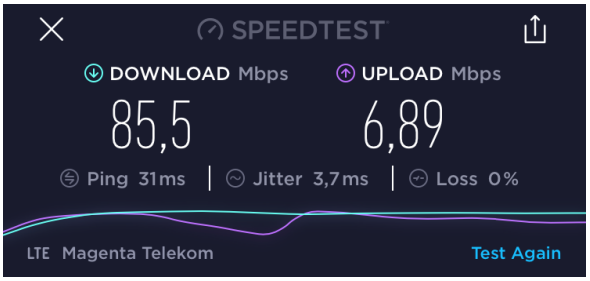

Figure 5.13: Internet Speed Test

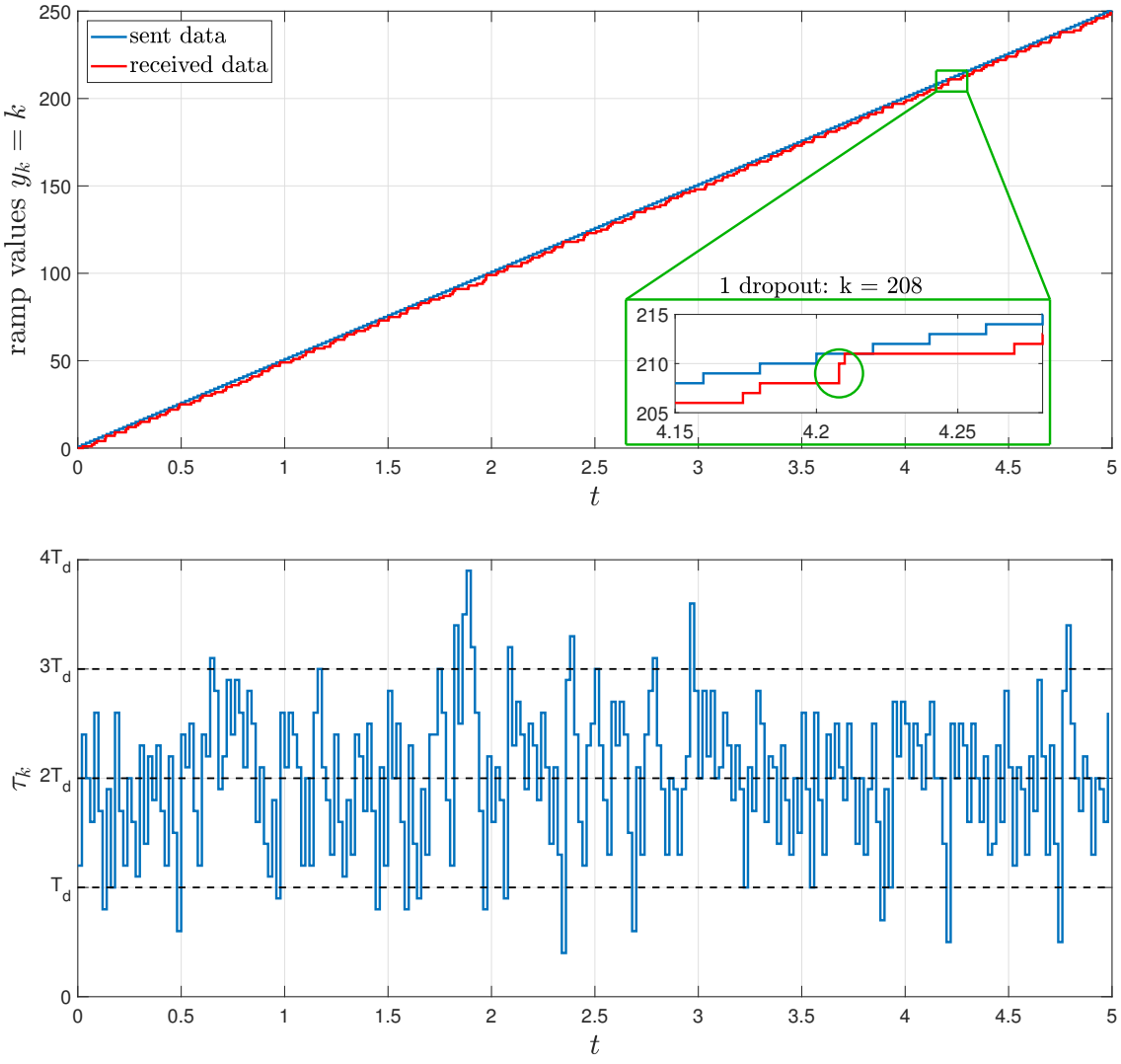

Figure 5.14: Estimation of the Round Trip Time (5s)

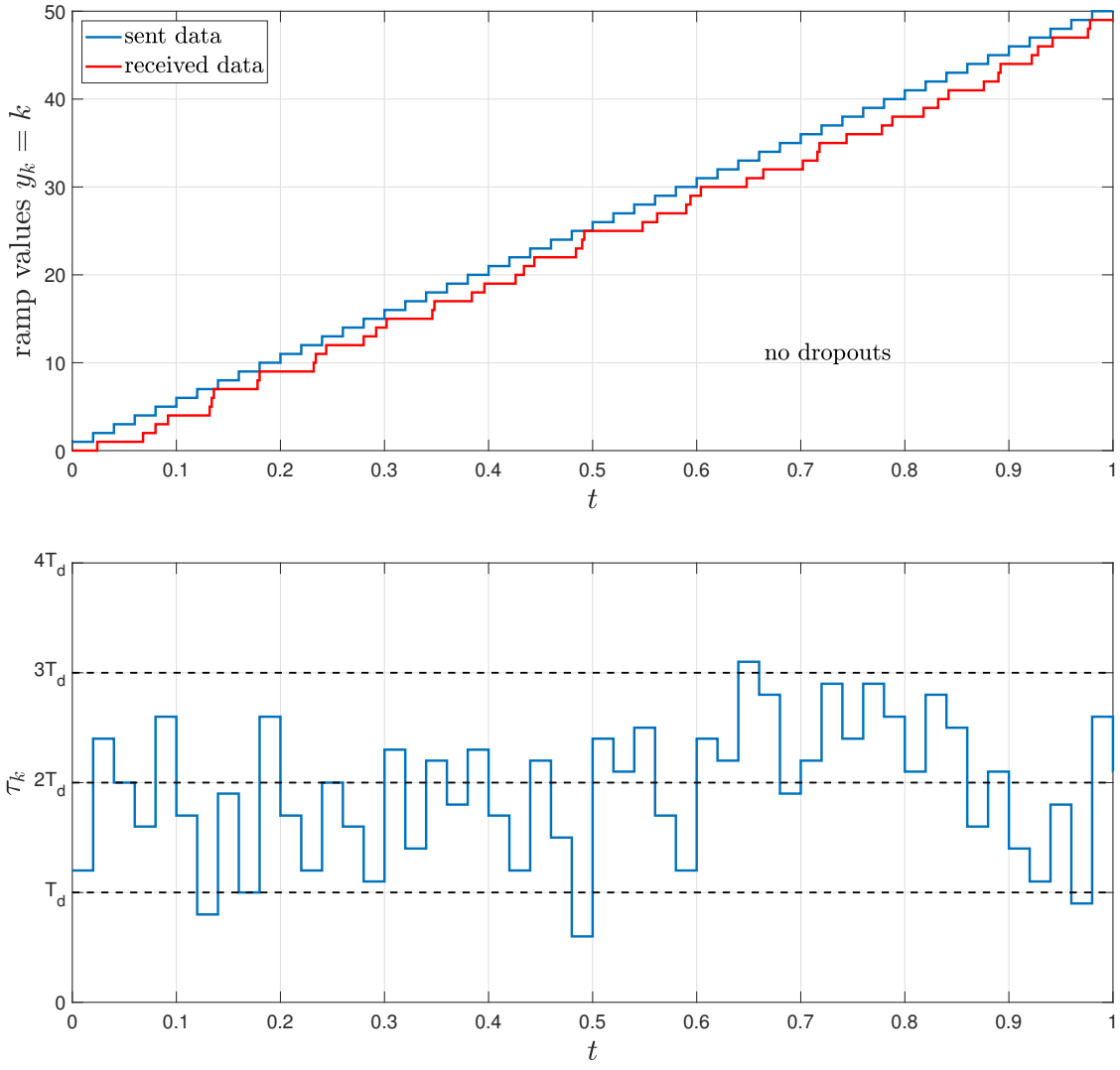

Figure 5.15: Estimation of the Round Trip Time (1s)

5.7 Implementation of the proposed Buffering Mechanism and Control Laws

The Simulink models from Figures 5.6 and 5.9 are used as a basis for implementation of the proposed buffering mechanism on the plant side and control laws on the controller side.

5.7.1 Message Rejection Mechanism

Since the UDP allows packets to arrive out of order, it is necessary to implement the Message Rejection Mechanism which ensures that, if more recent control data is available before the older data arrives to the plant, the older data should be neglected. This mechanism can be implemented with a very simple switching logic (see Figure 5.16) where a signal timestamp included in every data packet is used. The timestamps of the input signal and from the previously used signal are selected with a Simulink block *Selector* and compared with each other. In case of the received timestamp being older than the applied control data, the new data is used until more recent data is available. To demonstrate the effect of this block, a simple example is used. The timestamps of received data and Message Rejection block output data are shown in Figure 5.17.

5.7.2 Data Buffer

For the realization of the buffer, a simple *zero-order hold* (*ZOH*) block with a sample time equal to the sample time of the system to be controlled is used. The *ZOH* takes the newest signal and holds it for the specified sample time which is demonstrated in Figure 5.18 for the simple example where one sinus signal with sample time $0.1T_d$ is sent to the buffer with the sample time T_d.

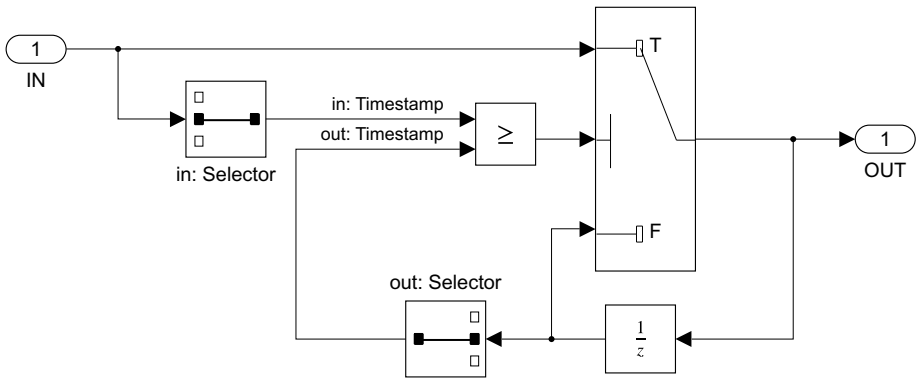

Figure 5.16: Realization of the message rejection mechanism

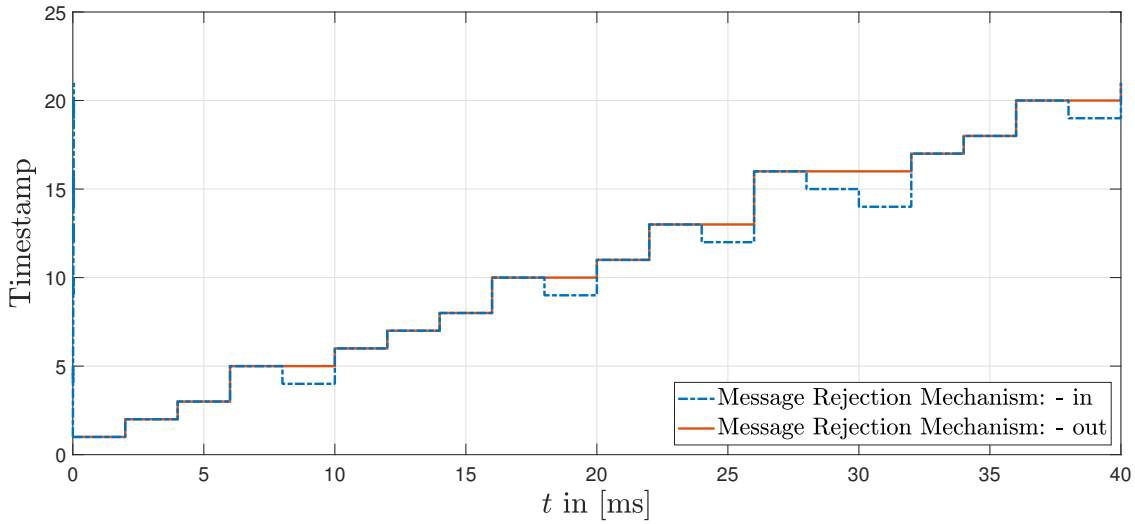

Figure 5.17: Message rejection mechanism

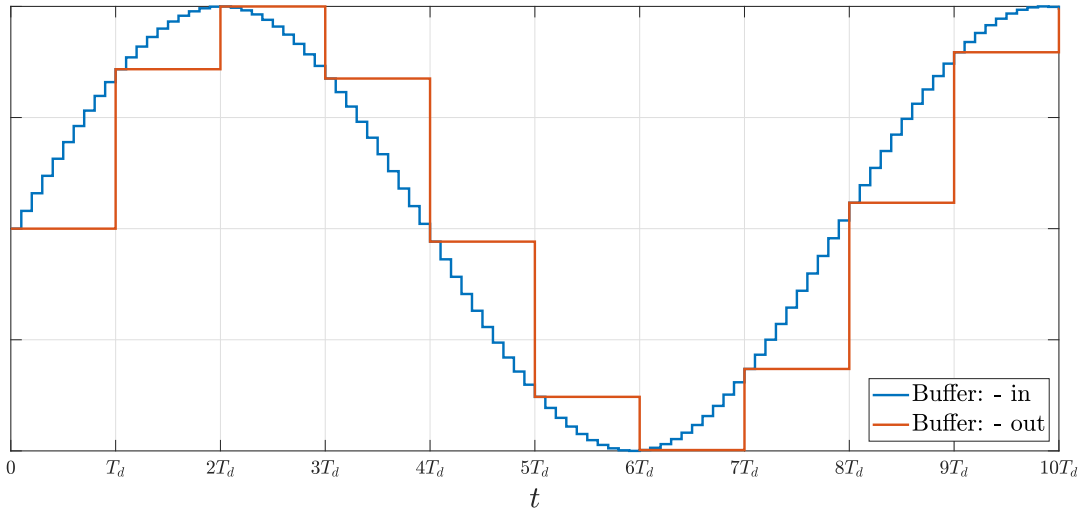

Figure 5.18: Buffering Mechanism

In addition to the message rejection and data buffer realization, the signals necessary for controller synthesis (plant signals $\boldsymbol{x}_k$ with corresponding timestamp and variable s_k calculated according to (4.11)) are sent to the *WiFi-UDP Send* block. The resulting Simulink model of the plant side with all necessary adaptations for the proposed control algorithms can be seen in Figure 5.19.

5.7.3 Control Law

In order to avoid synchronization of the two Simulink models, an enabled subsystem is used for the controller realization as well. This system gets activated as soon as the

received timestamp is larger than 5s, since from this time instant the plant side gets activated as well. The delay of 5s is not subtracted from the timestamp until the data is sent to Putty and being processed in order to differ between the value 0 when no data is received and when timestamp is equal to 0. For the realization of the proposed control laws the block *Unit Delay* is necessary. However, due to the sample time of the *UDP Receive* block which is in general smaller that the sample time of the system, it is possible that no new data is acquired between the sample steps. This leads to more samples of the same value which could make the use of one *Unit Delay* block useless. This problem can be solved either by using more *Unit Delay* blocks or more elegantly, by using the block *Buffer*, where one can set not only the buffer size but the overlap and initial condition as well. As a consequence, any input sequence can be buffered to a smaller or larger frame size. The function of this block relevant for this master's thesis is demonstrated by means of a simple example with *Repeating Sequence Stairs* source (integers from 1 to 10) for the buffer size equal to 3 and buffer overlap equal to 2 with zero as initial condition in Figure 5.21. One can see that in this scenario the last signal is the newest one corresponding to the input. The second to last output signal is the input signal delayed for one sample step and so on. The *Buffer* blocks in Figure 5.22 which represents the realization of the controller subsystem is chosen with buffer size equal to 20 and overlap equal to 19. In addition, the input signal is first sent to one *Unit Delay* block with $T_s = 4$ms to avoid algebraic loops, which does not affect the functionality of the proposed realization, since the newest signal is never used in the control laws. This approach is used for signals u_k, w_k, ν_k and for the timestamp value which are necessary for the realization of (4.13),(4.16) and (4.27) resulting in signals `uk_old`, `wk_old`, `nuk_old` and `TS_old` (see Figure 5.22).

The following lines of code from the first *MATLAB function* block is used for implementation of equations (4.2), (4.16) and (4.13):

```
function [uNk,sigmak,wk1] = ControlLaw(sk,xk,Ad, mT, K,TS
   ,TS_old, uk_old, wk_old)
  % wk1  = w_{k+1} uk_1 = u_{k-1}
  uNk = -K*xk;
  IND = find(TS_old < TS);  % vector of indices
  if isempty(IND)
    IND = 1;
  end
  ind = IND(end);    % index of the previous sample
  uk_1 = uk_old(ind);
  wk = wk_old(ind);
  wk1 = -mT*Ad*xk-uNk;

```

```
13    sigmak = sk+wk+uk_1;
14  end
```

The initial condition for the timestamp value is set to 1e6 to avoid possible confusion. Furthermore, the value 0 is chosen for initial condition of u_k and w_0 is set according to (4.30).

The similar approach is used in the second *MATLAB function* block implementing equations (4.23)-(4.29):

```
1  function [uSk,nuk1] = ControlLaw(sigmak,TS,TS_old,nuk_old
       ,Td, p)
2    % nuk1 = nu_{k+1}
3    ind = find(TS_old < TS);
4    if isempty(ind)
5      ind = 1;
6    end
7    ind = ind(end);
8
9    nuk = nuk_old(ind);
10
11   sk = p/(sqrt(abs(sigmak)));
12   qk = (sigmak~=0)*exp(sk*Td);
13
14   lTildek = sum(qk)-1;
15   lk = (lTildek-prod(qk))/Td;
16
17   uTildek = (lTildek-1)*sigmak/Td + nuk;
18   uSk = real(sigmak+Td*uTildek);
19
20   nuk1 = real(nuk+lk*sigmak);
21 end
```

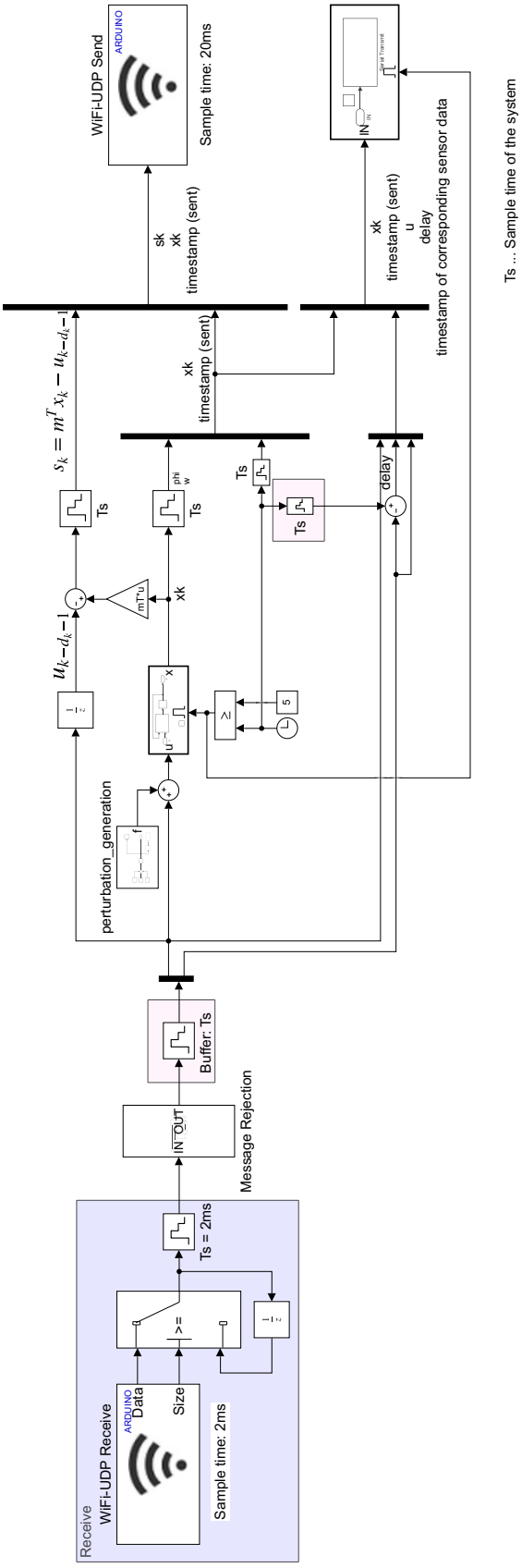

Figure 5.19: Simulink Model of the Plant Side with the proposed Buffering Mechanism

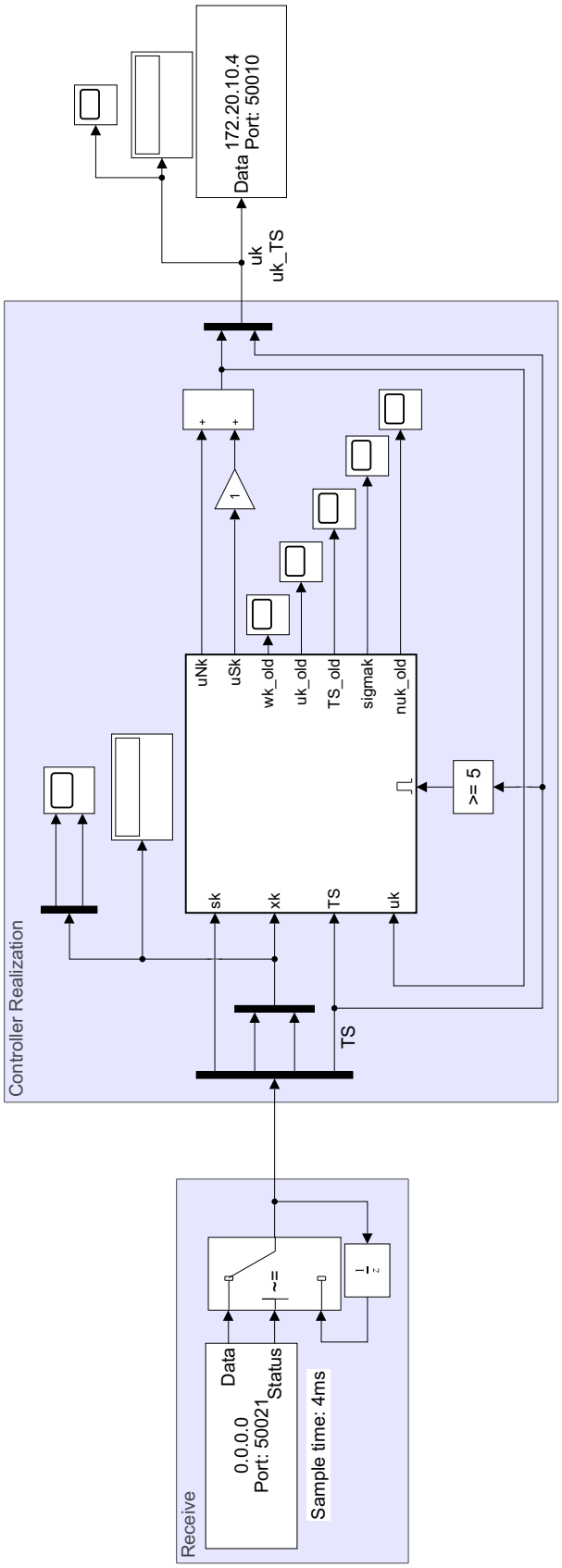

Figure 5.20: Simulink Model of the Controller Side with the proposed Control Law

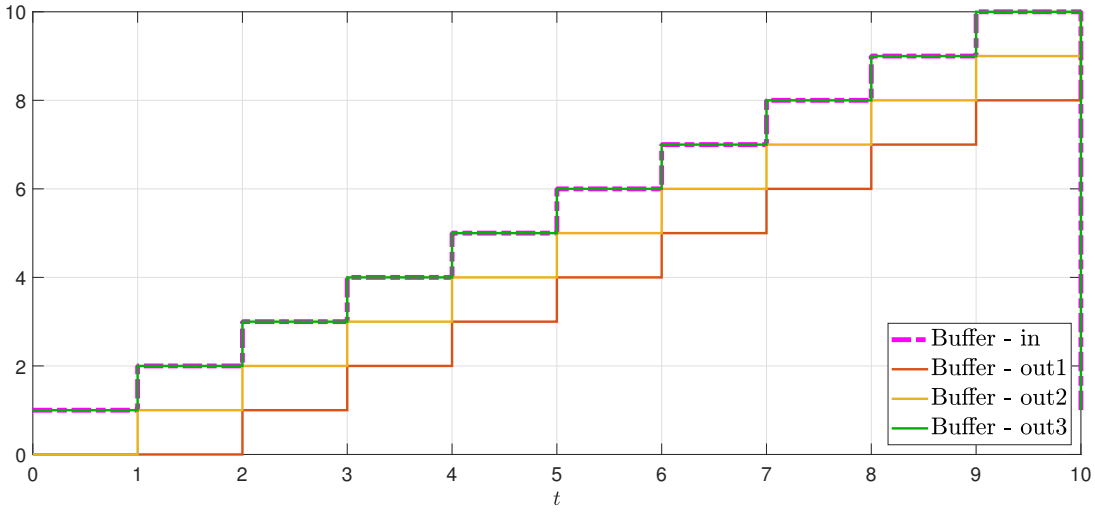

Figure 5.21: Simulink block *Buffer*

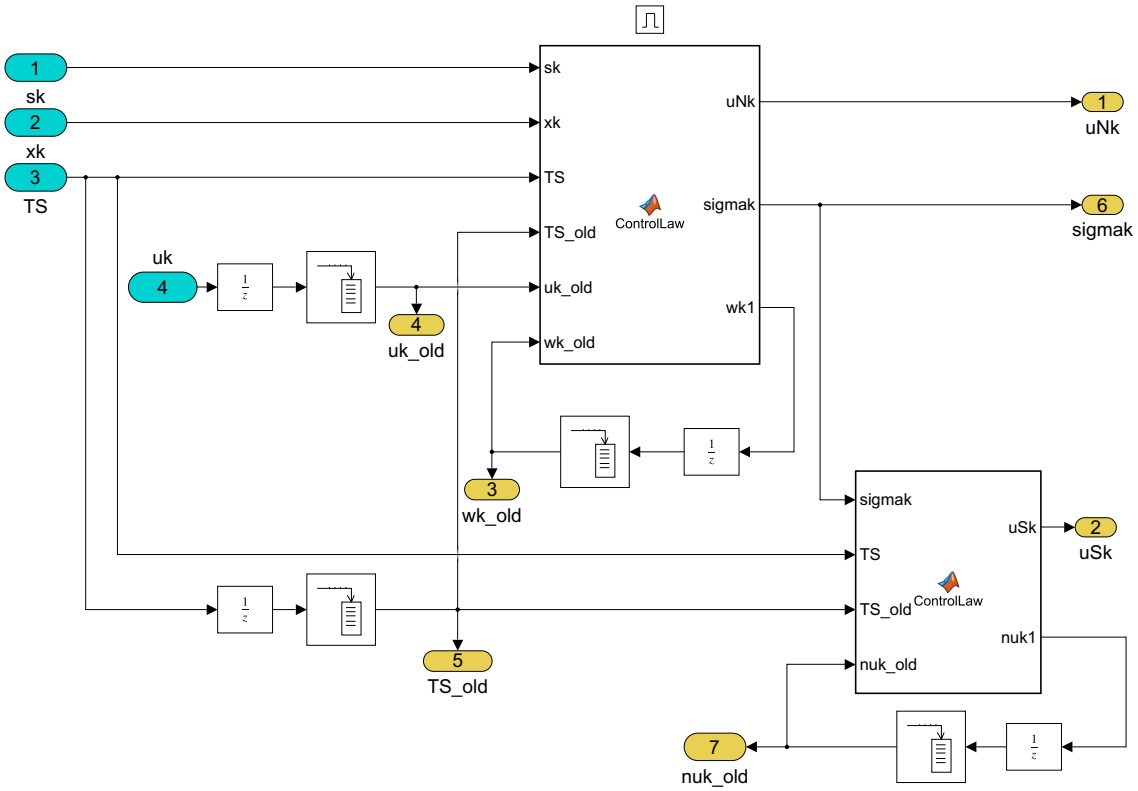

Figure 5.22: Simulink Model of the Control Law

6

Illustrative Example: Simulation and Experimental Results

Contents

In this chapter, the efficacy of the proposed algorithms is examined by means of an example representing a rotary servo plant consisting of a DC motor in a solid aluminum frame. First, a mathematical model describing the set-up is derived which serves as a basis for modeling of the corresponding NCS. Furthermore, to demonstrate the robustness of the system with respect to disturbances, the perturbation consisting of a constant component and two sinusoidal signals is applied. Afterwards, the controller proposed in chapter 4 is designed for the perturbed networked control system whose performance is examined in both simulation and experiments.

6.1 Mathematical Model of the Rotary Servo Plant

This example is based on the laboratory model Quanser SRV02 rotary servo plant which can directly provide rotary motion for different modules (e.g. inverted pendulum, flexible joint etc.) and is available at the Institute of Automation and Control. The schematic overview of the system can be found in Figure 6.1.

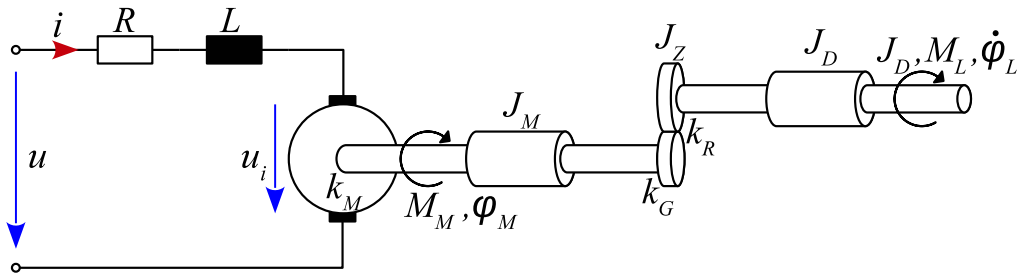

Figure 6.1: Schematic overview of the rotary servo plant

Under the assumption that the motor inductance can be neglected, the following equations can be derived based on Newton's Second Law of Motion and Kirchhoff's Voltage Law:

$$u = Ri + k_M \dot{\varphi}_M \tag{6.1}$$

with the parameters motor armature resistance R, motor back-emf constant k_M, total gear ration k_G, motor friction constant k_R and equivalent moment of inertia J given in Table 6.1.

$$J\ddot{\varphi}_L = k_G k_M i - k_R \dot{\varphi}_L \tag{6.2}$$

From the equation (6.1) it follows that

$$i = \frac{1}{R}(u - k_M \dot{\varphi}_M). \tag{6.3}$$

In addition, the following relations hold:

$$\varphi_M = k_G \varphi_L \tag{6.4}$$

$$M_L = k_G M_M. \tag{6.5}$$

Substituting (6.3)-(6.5) into (6.2) leads to the equation of motion:

$$\ddot{\varphi}_L + \left(\frac{k_G^2 k_M^2 - k_R R}{RJ}\right)\dot{\varphi}_L = \frac{k_G k_M}{RJ} u. \tag{6.6}$$

Symbol	Description	Value
R	Motor armature resistance	2.6Ω
k_M	Motor back-emf constant	$7.68 \cdot 10^{-7}$kg$\cdot$ m^2
k_G	Low-gear total gear ration	14
k_R	Motor friction constant	0.004 Nms
J	Equivalent moment of inertia	$1.7615 \cdot 10^{-4}$kg$\cdot$ m^2

Table 6.1: Physical specification of the model [17]

The equation (6.6) can be expressed in the state-space form by choosing the angle and rotational speed as the state variables of the state vector $\boldsymbol{x}_c = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$, whereas the armature circuit voltage is treated as the input.

$$x_1 := \varphi_L \tag{6.7}$$

$$x_2 := \dot{\varphi_L} = \dot{x}_1 \tag{6.8}$$

$$\dot{\boldsymbol{x}}_c = \begin{bmatrix} 0 & 1 \\ 0 & -\left(\dfrac{k_G^2 k_M^2 - k_R R}{RJ}\right) \end{bmatrix} \boldsymbol{x}_c + \begin{bmatrix} 0 \\ \dfrac{k_G k_M}{RJ} \end{bmatrix} u \tag{6.9}$$

The values of the parameters from (6.9) can be found in Table 6.1

6.2 Networked Control System

The sample time is chosen as $T_d = 20$ms. According to the considerations from the previous chapter, the round trip time is bounded by $4T_d$ and it is assumed that no dropouts occur. This leads to

$$\delta = 4. \tag{6.10}$$

The proposed buffering mechanism is implemented on the plant side so that for the round trip time

$$\tau_k \in \{0, T_d, 2T_d, 3T_d, 4T_d\} \tag{6.11}$$

holds and additionally, the perturbation f_k is applied to the plant

$$f_k = \frac{1}{3}\left[\sin\left(\frac{3}{2}t + 5\right) + \sin\left(\frac{1}{\pi}\left(\frac{3}{2}t + 5\right)\right) + 1\right] \tag{6.12}$$

Exact discretization of (6.9) with the implemented buffer and the perturbation gives

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_d \boldsymbol{x}_k + \boldsymbol{b}_d(u_{k-d_k} + f_k) \tag{6.13}$$

with

$$\boldsymbol{A}_d := \begin{bmatrix} 1 & 0.0129 \\ 0 & 0.3833 \end{bmatrix} \tag{6.14}$$

$$\boldsymbol{b}_d := \begin{bmatrix} 0.0349 \\ 3.0195 \end{bmatrix} \tag{6.15}$$

and

$$d_k \in \{0, 1, 2, 3, 4\} \tag{6.16}$$

6.3 Controller Synthesis

6.3.1 Nominal Control Law u_k^N

The nominal control law u_k^N is designed to ensure the stability of the nominal networked control system without the buffer, given by

$$\begin{aligned} \boldsymbol{x}_{k+1} = e^{\boldsymbol{A}_c T_d} + \int_{T_d - t_{k-3}^k}^{T_d} e^{\boldsymbol{A}_c \cdot s} \boldsymbol{b}_c ds \cdot u_{k-4} + \int_{T_d - t_{k-2}^k}^{T_d - t_{k-3}^k} e^{\boldsymbol{A}_c \cdot s} \boldsymbol{b}_c ds \cdot u_{k-3} + \\ + \int_{T_d - t_{k-1}^k}^{T_d - t_{k-2}^k} e^{\boldsymbol{A}_c \cdot s} \boldsymbol{b}_c ds \cdot u_{k-2} + \int_{T_d - t_k^k}^{T_d - t_{k-1}^k} e^{\boldsymbol{A}_c \cdot s} \boldsymbol{b}_c ds \cdot u_{k-1} + \int_{T_d - t_k^k}^{0} e^{\boldsymbol{A}_c \cdot s} \boldsymbol{b}_c ds \cdot u_k \end{aligned} \tag{6.17}$$

The piece of code from chapter 4 for $\gamma = 0.07$ is used for the controller synthesis in Matlab. The obtained nominal control law is given by

$$u_k^N = -\begin{bmatrix} 0.53073 & 0.0072248 \end{bmatrix} \boldsymbol{x}_k \tag{6.18}$$

The number of LMIs which have to be solved for the controller design is equal to 65.792 ($= 2^{n \cdot \delta} \cdot 2^{n \cdot \delta} + 2^{n \cdot \delta}$). If the parameter δ is increased to 5, the number of LMIs increases to 1.049.600 which results in significantly longer computation time. In addition, the effort necessary for the Jordan form used in the controller design increases as well. Details on the set of vertices (4.7) and convex over-approximation (4.8) derived both with Jordan form and analytical solutions of integrals, are given in the appendix A.

6.3.2 Sliding Mode Control Law u_k^S

For the controller synthesis of the sliding mode control law the integral sliding variable σ_k is defined according to (4.13). The vector $\boldsymbol{m}^T$ is chosen to satisfy the condition (4.12) ($\boldsymbol{m}^T = (\boldsymbol{b}_d^T \boldsymbol{b}_d)^{-1} \boldsymbol{b}_d^T$) and is therefore given by

$$\boldsymbol{m}^T = \begin{bmatrix} 0.0038 & 0.3311 \end{bmatrix} \tag{6.19}$$

6.4 Simulation Results

All simulations done in this master's thesis are made in a simulation toolbox which was developed as a part of [6]. In this simulation environment developed in Matlab, networked control systems with possibility for a buffering mechanism can be designed and different control strategies can be implemented. The performance of networked control systems can be examined with help of the Matlab/Simulink-based simulator for real-time control systems called TrueTime[1] [18] which is embedded in the simulation toolbox. For a detailed explanation of the simulation environment, see [6]. The adaptation of the simulation toolbox, which are needed for the implementation of the proposed buffering mechanism and control algorithms are specified in the appendix C and the adapted Simulink model used in the simulation is shown in Figure 6.2. For the choice of the parameter L_f (4.20), the change rate of the f_k is to be considered as

$$L_{f,\min} = \delta \cdot 0.6575 \tag{6.20}$$

The value used in simulations $L_f = 5$ is slightly higher than the exact change rate of the perturbation. The same round trip times defined as uniformly distributed random numbers between 0 and δT_d can be once defined in Matlab and used in all simulations. First, the equation (4.17) is used to verify the implementation of the proposed strategy, according to which the sliding variable equals the disturbance f_k delayed by one step in the case, where no sliding control law is applied. The simulation results in the described scenario is shown in Figure 6.3.
The simulation results for the case without sliding mode control law (i.e. $u_k = u_k^N$), for the original buffering mechanism and with the modified buffer proposed in this master's thesis are depicted in Figures 6.4 and 6.5. The signals obtained with the proposed approach show similar behavior as in the case of the original buffer, even though the system is still time-invariant and the input delay is still unknown. The impact of the reduced delay is especially demonstrated at the beginning of the simulation, where a control signal is applied to the system as soon as it is available, while in the case of the original buffer, no control signal is applied for the first $4T_d$. Moreover, the simulation results for the plant states $\boldsymbol{x}_k$ show significant reduction of the disturbance impact.
The chattering effects can be reduced by decreasing the sample time T_d, e.g., in case $T_d = 10$ms no chattering effects can be noticed. This would however result in a larger value of δ and hence, much higher number of LMIs to be solved.

[1] `http://www.control.lth.se/truetime/`

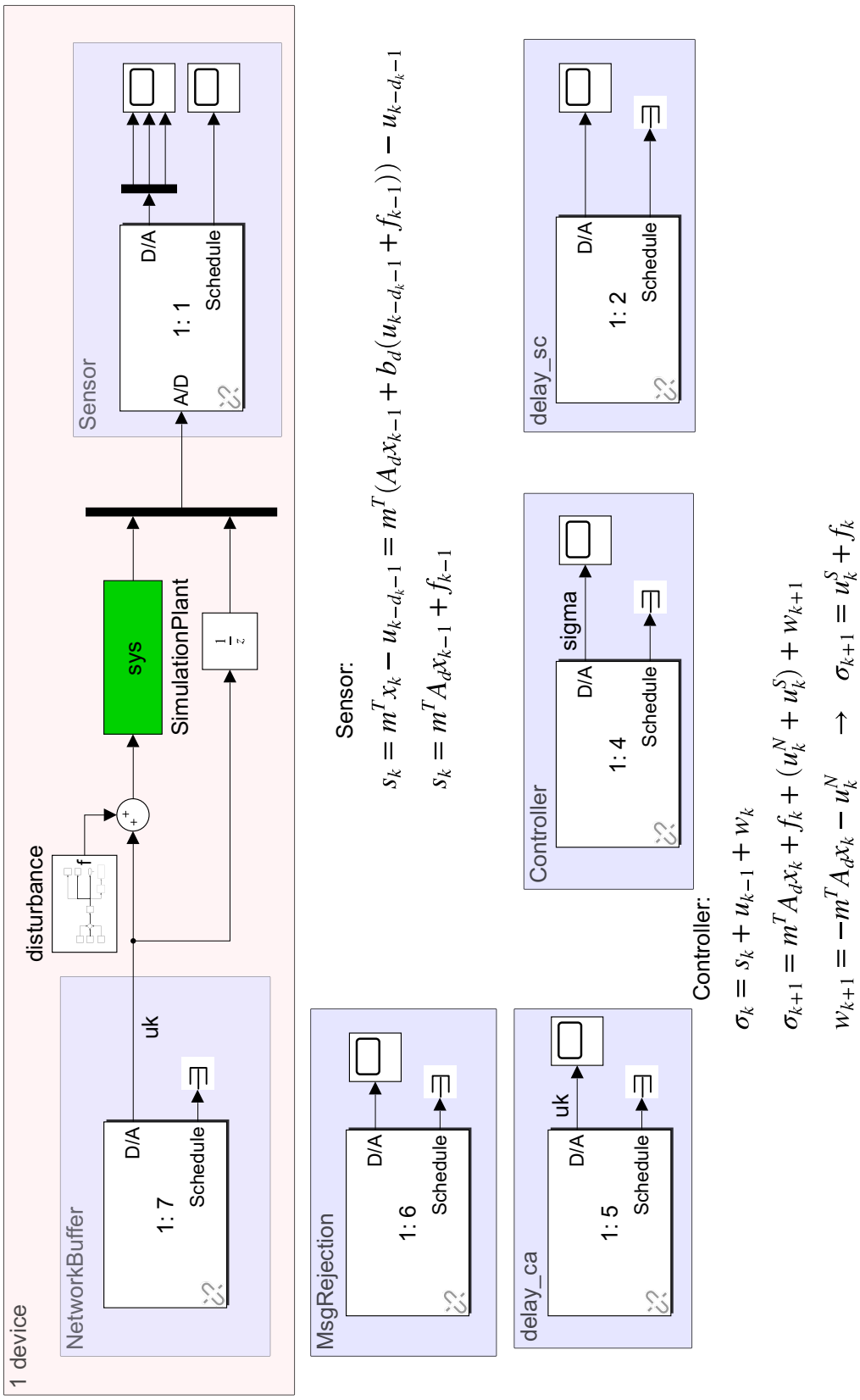

Figure 6.2: Example - simulations: Simulink model of the Wireless Networked Control System [6]

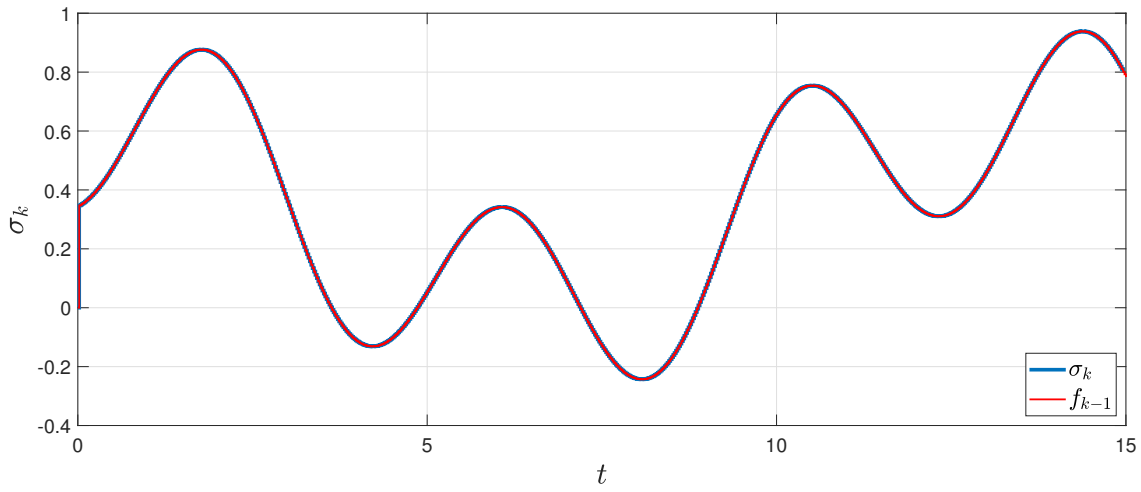

Figure 6.3: Sliding variable σ_k and disturbance f_{k-1} for $u_k^S = 0$

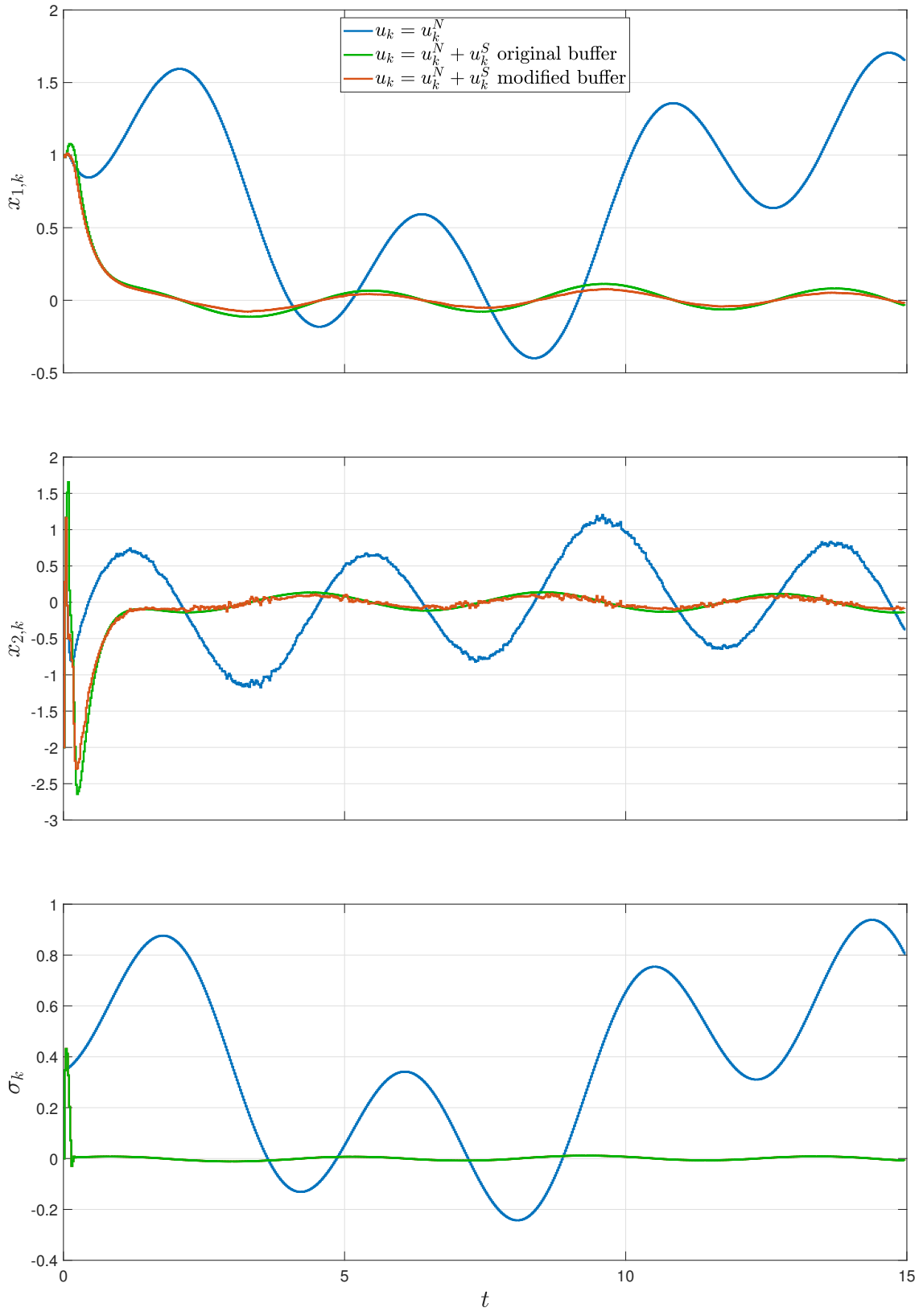

Figure 6.4: Example - simulation results: system states $\boldsymbol{x}_k$ and sliding variable σ_k

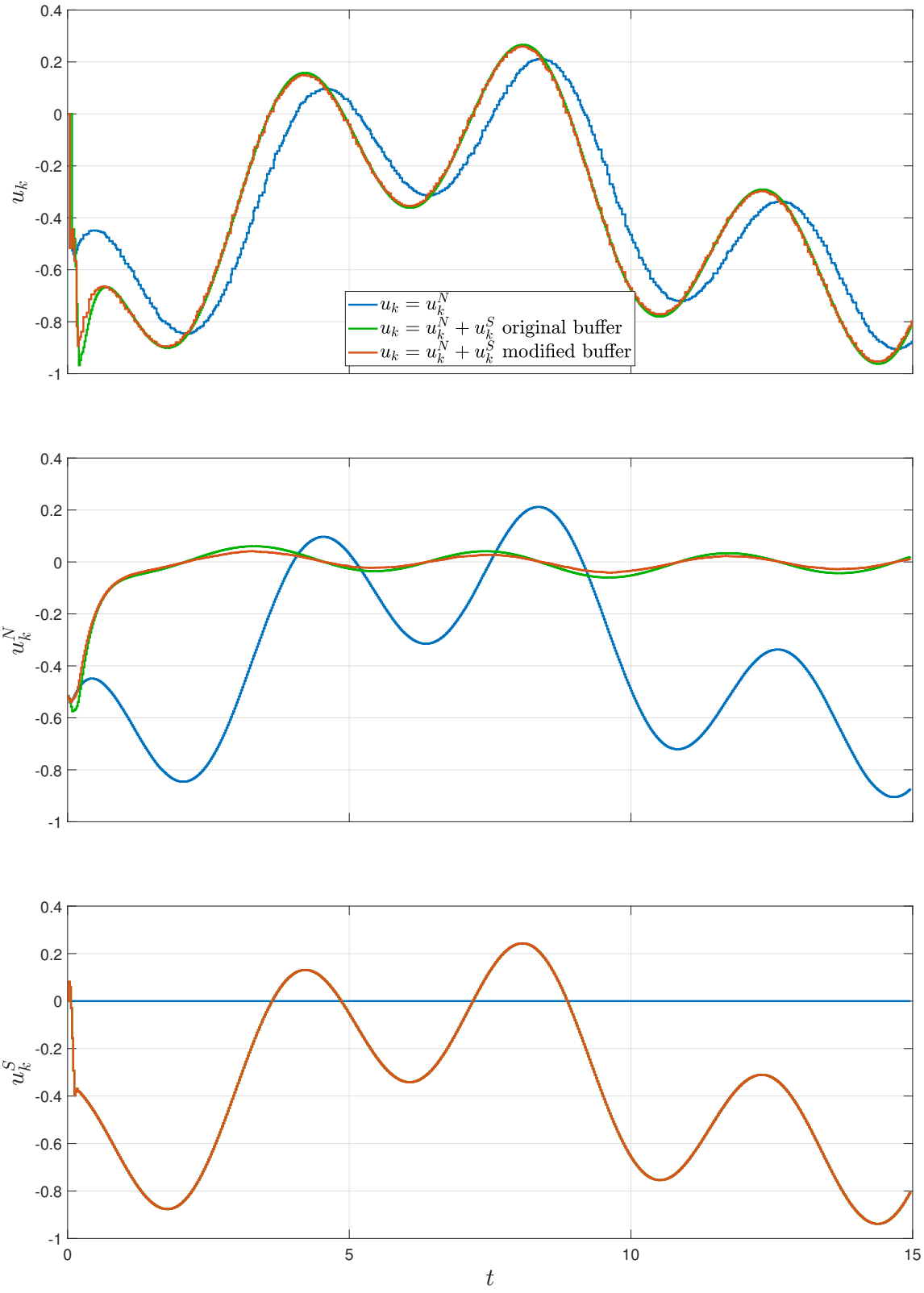

Figure 6.5: Example - simulation results: control signals u_k, u_k^N and u_k^S

6.5 Experimental Results

In this section, the experimental set-up and Simulink models 5.19 with 5.7, 5.16 and 5.20 with 5.22 described in the previous chapter are used. Furthermore, the same value of L_f as in the simulations is used.
As a verification of the set-up, the equation (4.17) can be exploited. The results which confirm that the sliding variable is equal to the disturbance f_k delayed by one step for $u_k = u_k^N$ are depicted in Figure 6.6. Results of two experiments are used to examine

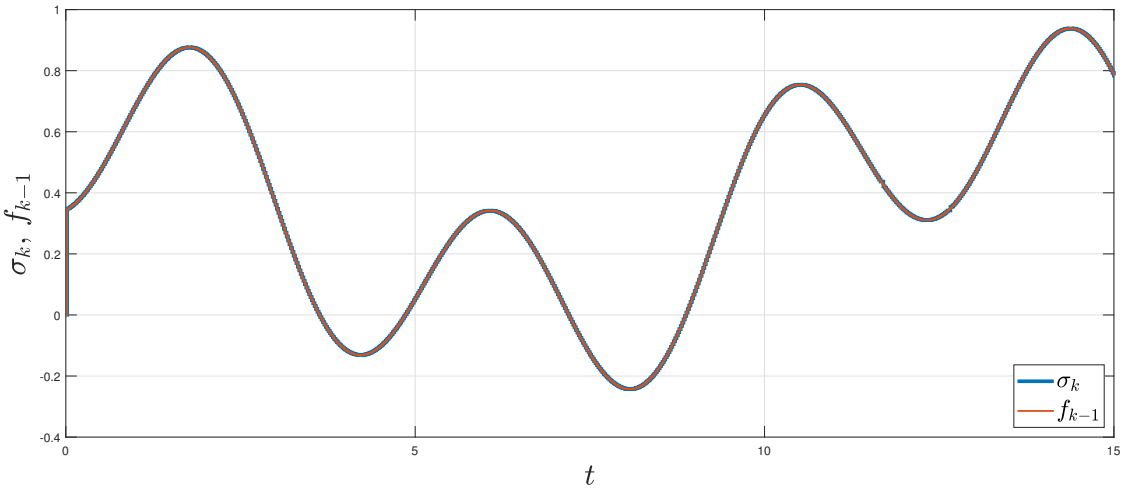

Figure 6.6: Example - experimental results: sliding variable σ_k and disturbance f_{k-1} for $u_k^S = 0$

the performance of the proposed approach: experiment with $u_k = u_k^N$ and experiment with the modified buffering mechanism where $u_k = u_k^N + u_k^S$ holds. Note that the case with original buffer implementation is not implemented in the experimental set-up. The signals $\boldsymbol{x}_k$, σ_k and u_k are shown in Figure 6.7. Additionally, the control signal and the resulting round trip delay can be seen in Figure 6.8.
It is clear from the Figure 6.8 representing the round trip time, that the round trip delays are significantly reduced comparing to the delays which would yield the case with the original buffer. The buffer implementation proposed in [6] would always introduce τ_k^b so that the round trip time is equal to the maximal delay $\delta T_d = 4T_d$, even though most of the delays resulting from the network imperfections are between one and two sample times. Furthermore, the considerable improvement of the accuracy can be observed in the evolution of the plant states when the sliding mode part of the control law is used.

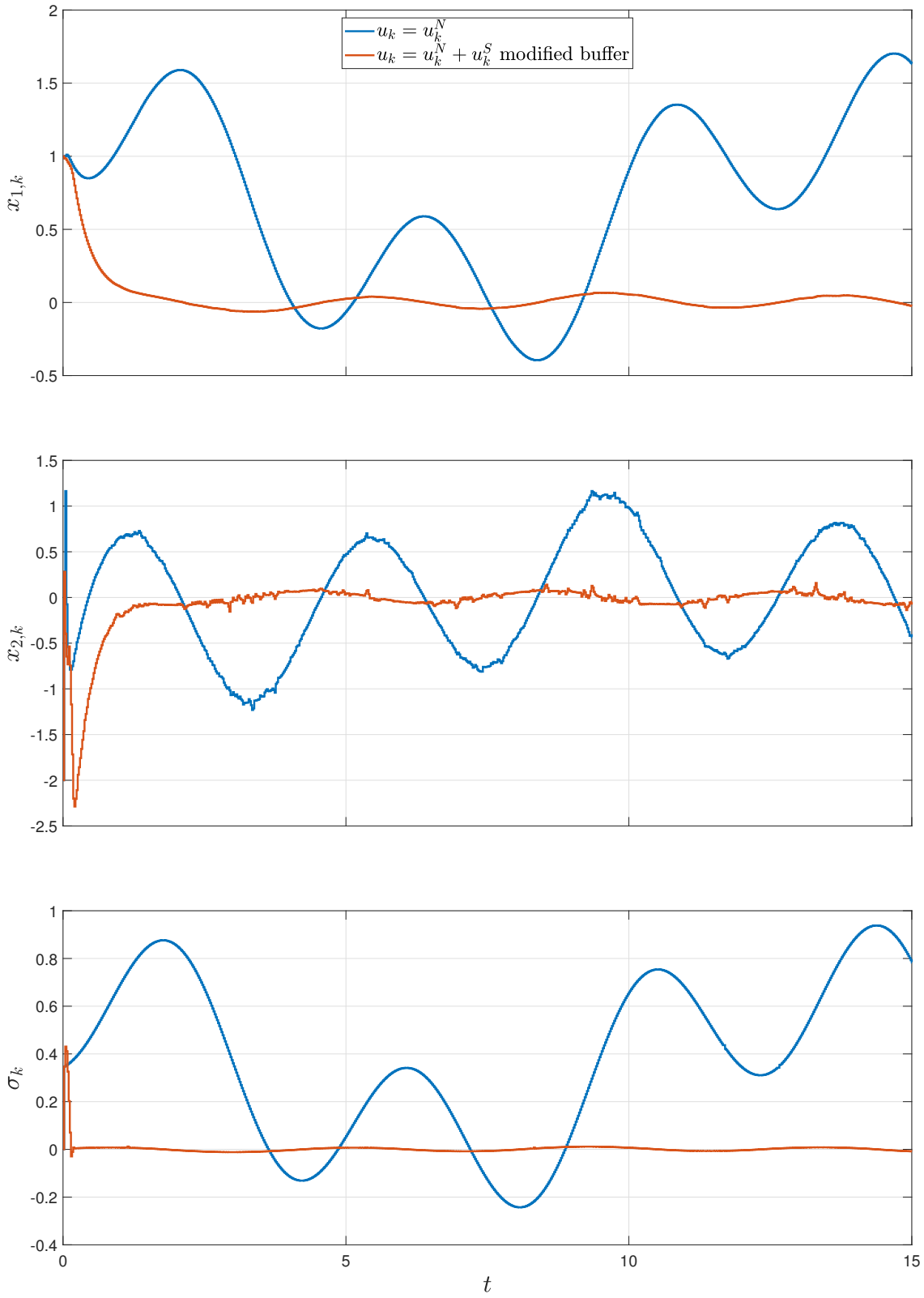

Figure 6.7: Example - experimental results: system states $\boldsymbol{x}_k$ and sliding variable σ_k

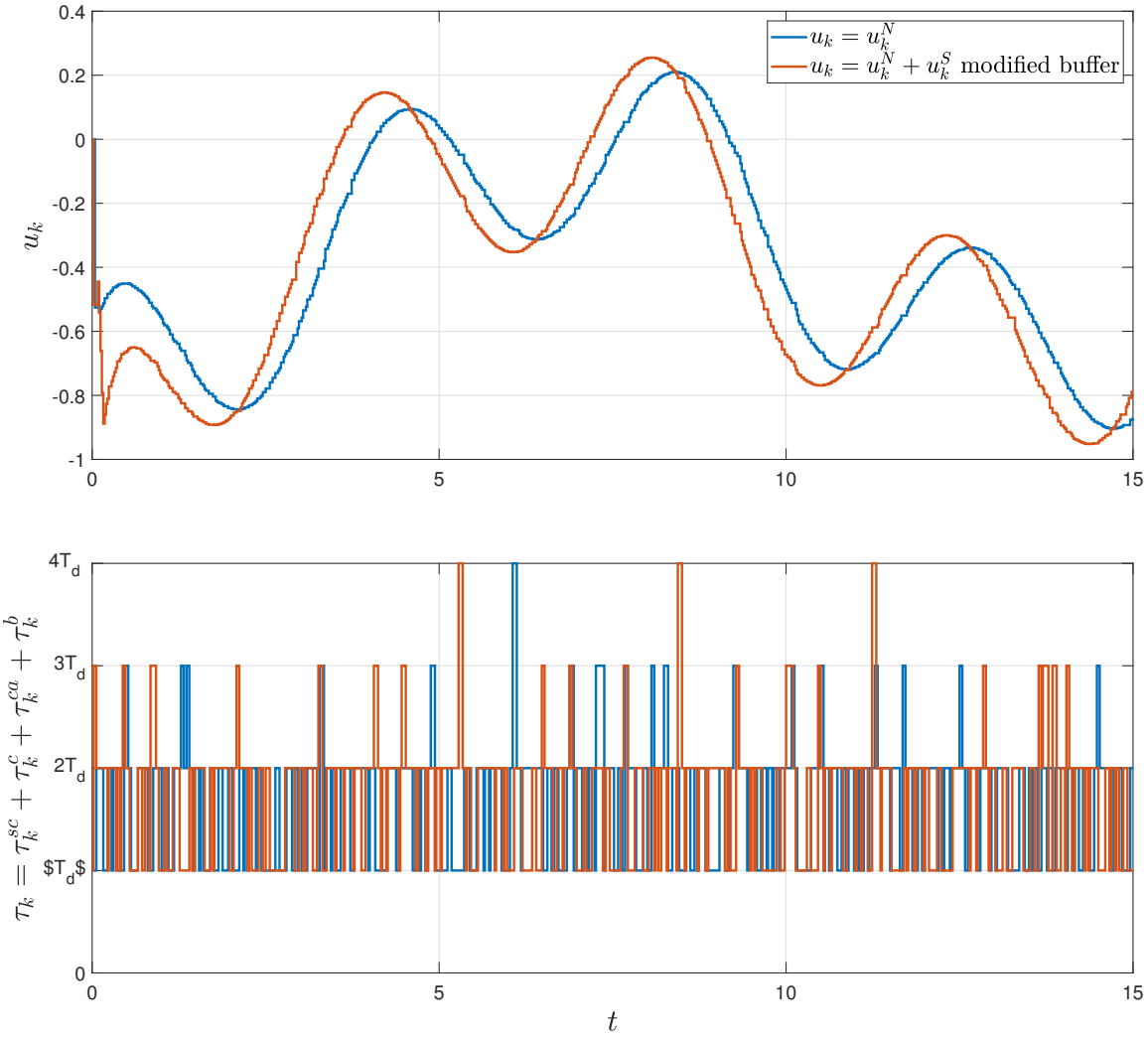

Figure 6.8: Example - experimental results: control signal u_k and round trip time τ_k

7 Summary and Outlook

This master's thesis aims to further exploit the potential of the sliding mode control algorithms in the perturbed networked control systems. To obtain a mathematical model which incorporates relevant network-induced imperfection but it is also suitable for the controller synthesis of the discretized version of the super-twisting algorithm, a specific buffering mechanisms is proposed. The buffered networked control system is motivated by the existing approaches available in literature and adapted with the goal to reduce the time delay introduced by the buffer. The challenge of the resulting time-variant model can be overcome through an unknown-input compensation approach implemented in the sensor and the performance of the proposed strategy is examined in the simulations. Furthermore, the wireless networked control system is implemented in Matlab/Simulink. For the purpose of connecting the plant with a controller over a WLAN, the Arduino board MKR1000 is used. The behavior of the system is studied with respect to network-induced delays and dropouts. Afterwards, the proposed approach for the buffered NCS subjected to perturbation is examined. The experiment results indicate that the system is still able to meet the required disturbance rejection as it is shown in the simulation study. Moreover, performed experiments show that round trip times obtained with the proposed buffering mechanism are significantly reduced.

Since the focus in this master's thesis is set on the application of the integral sliding mode algorithm to the time-variant model and WiFi implementation of wireless networked control systems, the existing control strategy proposed in [8] is used for the nominal controller law. However, this control law ensures stability of the NCS without

any buffer and therefore is the controller synthesis quite mathematically demanding as demonstrated in appendix A. It is based on deriving the stability conditions in form of linear matrix inequalities whose number increases exponentially with the system complexity and the parameter δ from (2.13). Hence, the potential for further research can be found in developing a new control strategy for the nominal control law which takes the simplification of the model obtained by the buffer into account. Furthermore, since the results from this master's thesis support the expectations regarding the unknown input delay compensation which allows the application of the sliding mode algorithm to time-variant models, this approach can be adapted to further reduce the delay introduced by the buffer.

Appendices

A

Illustrative Example: Jordan Form and Integration

Contents

Modeling of NCS imperfections such as time-varying delay is necessary to understand the influence of these uncertainties on the system performance and to design controllers which can guarantee the closed-loop stability. However, it leads to the model where these uncertainties appear in exponential form, which makes the stability analysis very complicated. To avoid this issue, instead of directly using the original NCS model based on exact-discretization of the continuous system, one of many over-approximation techniques available in literature can be applied to define a larger model whose structure is more convenient for the stability analysis. For the overview and comparison of the over-approximation methods (real Jordan form, Cayley-Hamilton theorem etc.) see [2] and references therein.

The considerations from this appendix are applied to the discrete-time single input NCS of the rotary servo plant from Chapter 6. Since the effects of the implemented buffering mechanism are not taken into account for the controller synthesis of the

nominal control law, the model to be considered is given by

$$\boldsymbol{x}_{k+1} = e^{\boldsymbol{A}_c T_d} + \int_{T_d - t_{k-3}^k}^{T_d} e^{\boldsymbol{A}_c \cdot s} \boldsymbol{b}_c ds \cdot u_{k-4} + \int_{T_d - t_{k-2}^k}^{T_d - t_{k-3}^k} e^{\boldsymbol{A}_c \cdot s} \boldsymbol{b}_c ds \cdot u_{k-3} + \\ + \int_{T_d - t_{k-1}^k}^{T_d - t_{k-2}^k} e^{\boldsymbol{A}_c \cdot s} \boldsymbol{b}_c ds \cdot u_{k-2} + \int_{T_d - t_k^k}^{T_d - t_{k-1}^k} e^{\boldsymbol{A}_c \cdot s} \boldsymbol{b}_c ds \cdot u_{k-1} + \int_{T_d - t_k^k}^{0} e^{\boldsymbol{A}_c \cdot s} \boldsymbol{b}_c ds \cdot u_k \quad \text{(A.1)}$$

that leads to the lifted model, which can be found at the end of this Appendix (A.48). In section A.1, the Jordan form is used to determine the generic solutions for integrals in (A.48) which can be used to rewrite the discrete-time NCS model. In section A.2, the integrals from (A.48) are analytically solved for the sake of comparison. The convex over-approximation is described in section A.3.

A.1 Jordan Form

In this section which is completely based on the appendix B in [8], the Jordan form is applied to obtain the generic solutions for the integrals of matrix exponentials necessary for controller synthesis of the nominal law described in chapter 4.
First, eigenvalues with their algebraic and geometric multiplicity (m_i and g_i, respectively) of the continuous-time matrix $\boldsymbol{A}_c$ from (6.9) are determined:

$$\lambda_1 = -47.8842 \qquad m_1 = 1 \quad g_1 = 1 \quad \text{(A.2a)}$$

$$\lambda_2 = 0 \qquad m_2 = 1 \quad g_2 = 1 \quad \text{(A.2b)}$$

The Jordan form is given by

$$\boldsymbol{J} = \operatorname{diag}(\boldsymbol{J}_R, \boldsymbol{J}_C) \quad \text{(A.3)}$$

with

$$\boldsymbol{J}_{NZ} = \operatorname{diag}(\boldsymbol{J}_{NZ,1}, \ldots, \boldsymbol{J}_{NZ,p_{\mathrm{NZ}}}) \quad \text{(A.4)}$$

$$\boldsymbol{J}_R = \operatorname{diag}(\boldsymbol{J}_{NZ}, \boldsymbol{J}_Z) \quad \text{(A.5)}$$

$$\boldsymbol{J}_C = \operatorname{diag}(\boldsymbol{J}_{C,1}, \ldots, \boldsymbol{J}_{C,p_{\mathrm{C}}}) \quad \text{(A.6)}$$

where $\boldsymbol{J}_{NZ,i}$ corresponds to the Jordan block of the i^{th} distinct real non-zero eigenvalue, $\boldsymbol{J}_Z$ is the Jordan matrix of the eigenvalues equal to zero and $\boldsymbol{J}_{C,i}$ is the Jordan block of the i^{th} distinct pair of complex eigenvalues. The number of distinct real nonzero eigenvalues is denoted by p_{NZ}, number of zero eigenvalues by p_Z with $p_R = p_{NZ} + p_Z$ and number of distinct complex pairs of eigenvalues by p_C.

Due to the fact that both eigenvalues are real, the Jordan Canonical form (A.5) is applied (in comparison to real Jordan form (A.6) in case of complex eigenvalues)[1]. From (A.2a) and $p = 2$ follows

$$\boldsymbol{J}_R = \begin{bmatrix} J_{NZ} & 0 \\ 0 & J_Z \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \tag{A.7}$$

$$\boldsymbol{Q} = \begin{bmatrix} -0.0209 & 1.0000 \\ 0.9998 & 0 \end{bmatrix} \tag{A.8}$$

A general solution for the integral of exponential of the continuous-time matrix $\boldsymbol{A}_c$ required to rewrite the NCS model is given by:

$$\int e^{\boldsymbol{A}_c s} ds = \int \boldsymbol{Q} e^{\boldsymbol{J}_R s} \boldsymbol{Q}^{-1} = \boldsymbol{Q} \begin{bmatrix} e^{J_{NZ} s} & 0 \\ 0 & e^{J_Z s} \end{bmatrix} \boldsymbol{Q}^{-1} ds \tag{A.9}$$

with

$$\int e^{J_{NZ} s} ds = e^{\lambda_1 s} S_{NZ,1,0} \tag{A.10}$$

$$\int e^{J_Z s} ds \;\; = s S_{Z,1,0} \tag{A.11}$$

For the (in this example) scalars $S_{NZ,1,0}$ and $S_{Z,1,0}$ holds

$$S_{NZ,1,0} = 1 \tag{A.12}$$

$$S_{Z,1,0} = 1 \tag{A.13}$$

Under the assumption that the sampling time is constant, the number of time-varying functions $\alpha_i(t_j^k)$ is defined as

$$\beta := \beta_{NZ} + \beta_Z + \beta_C. \tag{A.14}$$

The values β_{NZ}, β_Z and β_C depend on the value of δ and on the dimension of the corresponding Jordan block and are given by:

$$\beta_{NZ} = \delta \cdot \dim\, J_{NZ} = 4 \tag{A.15}$$

$$\beta_Z \;\; = \delta \cdot \dim\, J_Z \;\; = 4 \tag{A.16}$$

$$\beta_C \;\; = 0 \tag{A.17}$$

This results in 8 different time-varying functions $\alpha_i(t_j^k)$, for which, based on definitions from [8], one gets:

$$\alpha_i(t_j^k) = \begin{cases} \alpha_{NZ,i}(t_j^k) & \text{if } i \in \{1,2,3,4\} \\ \alpha_{Z,i}(t_j^k) & \text{if } i \in \{5,6,7,8\} \end{cases} \tag{A.18}$$

[1] Jordan Canonical Form can be determined in Matlab with the command `jordan`, whereas for real Jordan form the command `cdf2rdf` is relevant.

with

$$\alpha_{NZ,i}(t_j^k) = \begin{cases} \dfrac{(T_d - t_j^k)^{\hat{j}}}{\hat{j}!} e^{\lambda_1 (T_d - t_j^k)} & \text{for } \hat{j} = 0 \\ & \quad i = (j - (k-4)) \\ & \quad j = k-3, k-2, k-1, k \\ 0 & \text{if } \; i \notin \{1,2,3,4\} \end{cases} \tag{A.19}$$

for the nonzero eigenvalue λ_1 and

$$\alpha_{Z,i}(t_j^k) = \begin{cases} \dfrac{(T_d - t_j^k)^{\hat{j}}}{\hat{j}!} & \text{for } \hat{j} = 1 \\ & \quad i = 4 + (j - (k-4)) \\ & \quad j = k-3, k-2, k-1, k \\ 0 & \text{if } \; i \notin \{5,6,7,8\} \end{cases} \tag{A.20}$$

for the eigenvalue λ_2 equal to zero. The equations (A.18)-(A.20) result in the following $\alpha(t_j^k)$ functions

$$\begin{aligned} \alpha_{NZ,1}(t_{k-3}^k) &= e^{\lambda_1 \left(T_d - t_{k-3}^k\right)} & \alpha_{Z,5}(t_{k-3}^k) &= \left(T_d - t_{k-3}^k\right) \\ \alpha_{NZ,2}(t_{k-2}^k) &= e^{\lambda_1 \left(T_d - t_{k-2}^k\right)} & \alpha_{Z,6}(t_{k-2}^k) &= \left(T_d - t_{k-2}^k\right) \\ \alpha_{NZ,3}(t_{k-1}^k) &= e^{\lambda_1 \left(T_d - t_{k-1}^k\right)} & \alpha_{Z,7}(t_{k-1}^k) &= \left(T_d - t_{k-1}^k\right) \\ \alpha_{NZ,4}(t_k^k) &= e^{\lambda_1 \left(T_d - t_k^k\right)} & \alpha_{Z,8}(t_k^k) &= \left(T_d - t_k^k\right) \end{aligned} \tag{A.21}$$

The constant matrices $\boldsymbol{F}_0$, $\boldsymbol{F}_i$, $\boldsymbol{G}_0$ and $\boldsymbol{G}_i$ for $i \in \{1, 2, \dots 8\}$ are given by

$$\boldsymbol{F}_0 = \begin{bmatrix} \boldsymbol{Q}\boldsymbol{\Theta}_0\boldsymbol{Q}^{-1} & \boldsymbol{Q}\boldsymbol{\Theta}_1\boldsymbol{Q}^{-1}\boldsymbol{b}_c & \boldsymbol{Q}\boldsymbol{\Theta}_2\boldsymbol{Q}^{-1}\boldsymbol{b}_c & \boldsymbol{Q}\boldsymbol{\Theta}_3\boldsymbol{Q}^{-1}\boldsymbol{b}_c & \boldsymbol{Q}\boldsymbol{\Theta}_4\boldsymbol{Q}^{-1}\boldsymbol{b}_c \\ \boldsymbol{0}_{1\times 2} & 0 & 0 & 0 & 0 \\ \boldsymbol{0}_{1\times 2} & 1 & 0 & 0 & 0 \\ \boldsymbol{0}_{1\times 2} & 0 & 1 & 0 & 0 \\ \boldsymbol{0}_{1\times 2} & 0 & 0 & 1 & 0 \end{bmatrix} \tag{A.22a}$$

$$\boldsymbol{F}_i = \begin{bmatrix} \boldsymbol{Q}\boldsymbol{\Gamma}_{0,i}\boldsymbol{Q}^{-1} & \boldsymbol{Q}\boldsymbol{\Gamma}_{1,i}\boldsymbol{Q}^{-1}\boldsymbol{b}_c & \boldsymbol{Q}\boldsymbol{\Gamma}_{2,i}\boldsymbol{Q}^{-1}\boldsymbol{b}_c & \boldsymbol{Q}\boldsymbol{\Gamma}_{3,i}\boldsymbol{Q}^{-1}\boldsymbol{b}_c & \boldsymbol{Q}\boldsymbol{\Gamma}_{4,i}\boldsymbol{Q}^{-1}\boldsymbol{b}_c \\ \boldsymbol{0}_{1\times 2} & 0 & 0 & 0 & 0 \\ \boldsymbol{0}_{1\times 2} & 0 & 0 & 0 & 0 \\ \boldsymbol{0}_{1\times 2} & 0 & 0 & 0 & 0 \\ \boldsymbol{0}_{1\times 2} & 0 & 0 & 0 & 0 \end{bmatrix} \tag{A.22b}$$

$$\boldsymbol{G}_0 = \begin{bmatrix} \boldsymbol{Q}\boldsymbol{\Xi}_0\boldsymbol{Q}^{-1}\boldsymbol{b}_c \\ 1 \\ \boldsymbol{0}_{3\times 1} \end{bmatrix} \tag{A.22c}$$

$$\boldsymbol{G}_i = \begin{bmatrix} \boldsymbol{Q}\boldsymbol{\Xi}_i\boldsymbol{Q}^{-1}\boldsymbol{b}_c \\ \boldsymbol{0}_{4\times 1} \end{bmatrix} \tag{A.22d}$$

Note, that matrices $\boldsymbol{\Theta}_{\hat{i}}$, $\boldsymbol{\Gamma}_{\hat{i},i}$ and $\boldsymbol{\Xi}_{\hat{i}}$, $i \in \{1,2,\ldots,8\}, \hat{i} \in \{0,1,2,3,4\}$ should be defined for the eigenvalue different from zero and the eigenvalue equal to zero separately, after which the following relations can be used

$$\boldsymbol{\Theta}_{\hat{i}} = \operatorname{diag}(\boldsymbol{\Theta}_{\hat{i}}^{NZ}, \boldsymbol{\Theta}_{\hat{i}}^{Z}) \tag{A.23a}$$
$$\boldsymbol{\Gamma}_{\hat{i},i} = \operatorname{diag}(\boldsymbol{\Gamma}_{\hat{i},i}^{NZ}, \boldsymbol{\Gamma}_{\hat{i},i}^{Z}) \tag{A.23b}$$
$$\boldsymbol{\Xi}_{\hat{i}} = \operatorname{diag}(\boldsymbol{\Xi}_{\hat{i}}^{NZ}, \boldsymbol{\Xi}_{\hat{i}}^{Z}) \tag{A.23c}$$

with:

- $\boldsymbol{\Theta}_{\hat{i}} = \begin{bmatrix} \boldsymbol{\Theta}_0^{NZ} & 0 \\ 0 & \boldsymbol{\Theta}_0^{Z} \end{bmatrix}, \hat{i} \in \{0,1,2,3,4\}$

$$\boldsymbol{\Theta}_0 = \begin{bmatrix} e^{J_{NZ}T_d} & 0 \\ 0 & e^{J_Z T_d} \end{bmatrix} \tag{A.24}$$
$$\boldsymbol{\Theta}_1 = \mathbf{0}_{2\times 2} \tag{A.25}$$
$$\boldsymbol{\Theta}_1 = \mathbf{0}_{2\times 2} \tag{A.26}$$
$$\boldsymbol{\Theta}_1 = \mathbf{0}_{2\times 2} \tag{A.27}$$
$$\boldsymbol{\Theta}_4 = \begin{bmatrix} J_{NZ}^{-1} e^{J_{NZ}T_d} & 0 \\ 0 & T_d S_{Z,1,0} \end{bmatrix} \tag{A.28}$$

- $\boldsymbol{\Gamma}_{\hat{i},i} = \begin{bmatrix} \boldsymbol{\Gamma}_{\hat{i},i}^{NZ} & 0 \\ 0 & \boldsymbol{\Gamma}_{\hat{i},i}^{Z} \end{bmatrix}, \hat{i} \in \{0,1,2,3,4\}$and $i \in \{1,2,\ldots,8\}$

$$\boldsymbol{\Gamma}_{0,i} = \mathbf{0}_{2\times 2} \quad \forall i \in \{1,2,\ldots,8\} \tag{A.29}$$

$$\boldsymbol{\Gamma}_{1,1} = \mathbf{0}_{2\times 2} \tag{A.30a}$$
$$\boldsymbol{\Gamma}_{1,2} = \mathbf{0}_{2\times 2} \tag{A.30b}$$
$$\boldsymbol{\Gamma}_{1,3} = \begin{bmatrix} J_{NZ}^{-1} T_3^{NZ} & 0 \\ 0 & 0 \end{bmatrix} \tag{A.30c}$$
$$\boldsymbol{\Gamma}_{1,4} = \begin{bmatrix} -J_{NZ}^{-1} T_4^{NZ} & 0 \\ 0 & 0 \end{bmatrix} \tag{A.30d}$$
$$\boldsymbol{\Gamma}_{1,5} = \mathbf{0}_{2\times 2} \tag{A.30e}$$
$$\boldsymbol{\Gamma}_{1,6} = \mathbf{0}_{2\times 2} \tag{A.30f}$$
$$\boldsymbol{\Gamma}_{1,7} = \begin{bmatrix} 0 & 0 \\ 0 & T_7^Z \end{bmatrix} \tag{A.30g}$$
$$\boldsymbol{\Gamma}_{1,8} = \begin{bmatrix} 0 & 0 \\ 0 & -T_8^Z \end{bmatrix} \tag{A.30h}$$

$$\boldsymbol{\Gamma}_{2,1} = \mathbf{0}_{2\times 2} \tag{A.31a}$$

$$\boldsymbol{\Gamma}_{2,2} = \begin{bmatrix} J_{NZ}^{-1} T_2^{NZ} & 0 \\ 0 & 0 \end{bmatrix} \tag{A.31b}$$

$$\boldsymbol{\Gamma}_{2,3} = \begin{bmatrix} -J_{NZ}^{-1} T_3^{NZ} & 0 \\ 0 & 0 \end{bmatrix} \tag{A.31c}$$

$$\boldsymbol{\Gamma}_{2,4} = \mathbf{0}_{2\times 2} \tag{A.31d}$$

$$\boldsymbol{\Gamma}_{2,5} = \mathbf{0}_{2\times 2} \tag{A.31e}$$

$$\boldsymbol{\Gamma}_{2,6} = \begin{bmatrix} 0 & 0 \\ 0 & T_7^Z \end{bmatrix} \tag{A.31f}$$

$$\boldsymbol{\Gamma}_{2,7} = \begin{bmatrix} 0 & 0 \\ 0 & -T_8^Z \end{bmatrix} \tag{A.31g}$$

$$\boldsymbol{\Gamma}_{2,8} = \mathbf{0}_{2\times 2} \tag{A.31h}$$

$$\boldsymbol{\Gamma}_{3,1} = \begin{bmatrix} J_{NZ}^{-1} T_1^{NZ} & 0 \\ 0 & 0 \end{bmatrix} \tag{A.32a}$$

$$\boldsymbol{\Gamma}_{3,2} = \begin{bmatrix} -J_{NZ}^{-1} T_2^{NZ} & 0 \\ 0 & 0 \end{bmatrix} \tag{A.32b}$$

$$\boldsymbol{\Gamma}_{3,3} = \mathbf{0}_{2\times 2} \tag{A.32c}$$

$$\boldsymbol{\Gamma}_{3,4} = \mathbf{0}_{2\times 2} \tag{A.32d}$$

$$\boldsymbol{\Gamma}_{3,5} = \begin{bmatrix} 0 & 0 \\ 0 & T_5^Z \end{bmatrix} \tag{A.32e}$$

$$\boldsymbol{\Gamma}_{3,6} = \begin{bmatrix} 0 & 0 \\ 0 & -T_6^Z \end{bmatrix} \tag{A.32f}$$

$$\boldsymbol{\Gamma}_{3,7} = \mathbf{0}_{2\times 2} \tag{A.32g}$$

$$\boldsymbol{\Gamma}_{3,8} = \mathbf{0}_{2\times 2} \tag{A.32h}$$

$$\boldsymbol{\Gamma}_{4,1} = \begin{bmatrix} -J_{NZ}^{-1} T_1^{NZ} & 0 \\ 0 & 0 \end{bmatrix} \tag{A.33a}$$

$$\boldsymbol{\Gamma}_{4,2} = \mathbf{0}_{2\times 2} \tag{A.33b}$$

$$\boldsymbol{\Gamma}_{4,3} = \mathbf{0}_{2\times 2} \tag{A.33c}$$

$$\boldsymbol{\Gamma}_{4,4} = \mathbf{0}_{2\times 2} \tag{A.33d}$$

$$\boldsymbol{\Gamma}_{4,5} = \begin{bmatrix} 0 & 0 \\ 0 & -T_5^Z \end{bmatrix} \tag{A.33e}$$

$$\boldsymbol{\Gamma}_{4,6} = \mathbf{0}_{2\times 2} \tag{A.33f}$$

$$\boldsymbol{\Gamma}_{4,7} = \mathbf{0}_{2\times 2} \tag{A.33g}$$

$$\boldsymbol{\Gamma}_{4,8} = \mathbf{0}_{2\times 2} \tag{A.33h}$$

- $\boldsymbol{\Xi}_{\hat{i}} = \begin{bmatrix} \boldsymbol{\Xi}_0^{NZ} & 0 \\ 0 & \boldsymbol{\Xi}_0^{Z} \end{bmatrix}, \hat{i} \in \{0,1,2,3,4\}$

$$\boldsymbol{\Xi}_0 = \begin{bmatrix} -J_{NZ}^{-1} & 0 \\ 0 & 0 \end{bmatrix} \tag{A.34}$$

$$\boldsymbol{\Xi}_1 = \mathbf{0}_{2\times 2} \tag{A.35}$$

$$\boldsymbol{\Xi}_1 = \mathbf{0}_{2\times 2} \tag{A.36}$$

$$\boldsymbol{\Xi}_1 = \mathbf{0}_{2\times 2} \tag{A.37}$$

$$\boldsymbol{\Xi}_4 = \begin{bmatrix} J_{NZ}^{-1} T_4^{NZ} & 0 \\ 0 & T_8^Z \end{bmatrix} \tag{A.38}$$

T_i^{NZ} and T_i^Z are for $i \in \{1,2,\ldots,8\}$ in this example scalars given by

$$T_i^{NZ} = \begin{cases} S_{NZ,1,0} & i \in \{1,2,3,4\} \\ 0 & i \notin \{1,2,3,4\} \end{cases} \tag{A.39}$$

and

$$T_i^{Z} = \begin{cases} S_{Z,1,0} & i \in \{5,6,7,8\} \\ 0 & i \notin \{5,6,7,8\} \end{cases} \tag{A.40}$$

with the scalars $S_{NZ,1,0}$ and $S_{Z,1,0}$ defined in (A.12).
With the constant matrices $\boldsymbol{F}_0$, $\boldsymbol{F}_i$, $\boldsymbol{G}_0$ and $\boldsymbol{G}_i$ for $i \in \{1,2,\ldots 8\}$ and time-varying functions $\alpha_i(t_j^k)$ the expression (4.5) can be determined, which is used for the controller synthesis of the nominal control law.

A.2 Integration

In this section, the integrals in the lifted model from A.48 are solved analytically. For the sake of both clarity and overview, the Jordan form of the matrix $\boldsymbol{A}_c$ is again used, where the eigenvalue λ_2 which is equal to zero is used directly to simplify the expressions. First, the solution of the integral with general limits is determined.

$$\begin{aligned}\int_A^B e^{\boldsymbol{A}_c \cdot s}\boldsymbol{b}_c ds = \int_A^B \boldsymbol{Q}e^{\boldsymbol{J}\cdot s}\boldsymbol{Q}^{-1}\boldsymbol{b}_c ds = \boldsymbol{Q}\begin{bmatrix}\frac{1}{\lambda_1}e^{\lambda_1 s} & 0\\ 0 & s\end{bmatrix}_A^B \boldsymbol{Q}^{-1}\boldsymbol{b}_c = \\ = \frac{1}{\lambda_1}\boldsymbol{Q}\begin{bmatrix}\left(e^{\lambda_1 B} - e^{\lambda_1 A}\right) & 0\\ 0 & (B-A)\end{bmatrix}\boldsymbol{Q}^{-1}\boldsymbol{b}_c \quad \text{(A.41)}\end{aligned}$$

To determine the time-varying functions α_i the expression (A.41) is evaluated for the limits in the lifted model, which can be found at the end of this appendix (A.48):

$$\int_{T_d-t_{k-3}^k}^{T_d} e^{A_c\cdot s}\boldsymbol{b}_c ds = \frac{1}{\lambda_1}\boldsymbol{Q}\begin{bmatrix}e^{\lambda_1 T_d} - e^{\lambda_1\left(T_d - t_{k-3}^k\right)} & 0\\ 0 & T_d - (T_d - t_{k-3}^k)\end{bmatrix}\boldsymbol{Q}^{-1}\boldsymbol{b}_c \quad \text{(A.42a)}$$

$$\int_{T_d-t_{k-2}^k}^{T_d-t_{k-3}^k} e^{A_c\cdot s}\boldsymbol{b}_c ds = \frac{1}{\lambda_1}\boldsymbol{Q}\begin{bmatrix}e^{\lambda_1(T_d - t_{k-3}^k)} - e^{\lambda_1\left(T_d - t_{k-2}^k\right)} & 0\\ 0 & (T_d - t_{k-3}^k) - (T_d - t_{k-2}^k)\end{bmatrix}\boldsymbol{Q}^{-1}\boldsymbol{b}_c \quad \text{(A.42b)}$$

$$\int_{T_d-t_{k-1}^k}^{T_d-t_{k-2}^k} e^{A_c\cdot s}\boldsymbol{b}_c ds = \frac{1}{\lambda_1}\boldsymbol{Q}\begin{bmatrix}e^{\lambda_1(T_d - t_{k-2}^k)} - e^{\lambda_1\left(T_d - t_{k-1}^k\right)} & 0\\ 0 & (T_d - t_{k-2}^k) - (T_d - t_{k-1}^k)\end{bmatrix}\boldsymbol{Q}^{-1}\boldsymbol{b}_c \quad \text{(A.42c)}$$

$$\int_{T_d-t_k^k}^{T_d-t_{k-1}^k} e^{A_c\cdot s}\boldsymbol{b}_c ds = \frac{1}{\lambda_1}\boldsymbol{Q}\begin{bmatrix}e^{\lambda_1(T_d - t_{k-1}^k)} - e^{\lambda_1\left(T_d - t_k^k\right)} & 0\\ 0 & (T_d - t_{k-1}^k) - (T_d - t_k^k)\end{bmatrix}\boldsymbol{Q}^{-1}\boldsymbol{b}_c \quad \text{(A.42d)}$$

$$\int_0^{T_d-t_k^k} e^{A_c\cdot s}\boldsymbol{b}_c ds = \frac{1}{\lambda_1}\boldsymbol{Q}\begin{bmatrix}1 - e^{\lambda_1\left(T_d - t_k^k\right)} & 0\\ 0 & (T_d - t_k^k)\end{bmatrix}\boldsymbol{Q}^{-1}\boldsymbol{b}_c \quad \text{(A.42e)}$$

The only unknown variables in the expressions (A.42a)-(A.42e) are the arrival times resulting in the following time-varying functions $\alpha_i(\boldsymbol{\theta}_k)$:

$$\begin{aligned}\alpha_1(\boldsymbol{\theta_k}) &= e^{\lambda_1\left(T_d - t_{k-3}^k\right)} & \alpha_5(\boldsymbol{\theta_k}) &= \left(T_d - t_{k-3}^k\right)\\ \alpha_2(\boldsymbol{\theta_k}) &= e^{\lambda_1\left(T_d - t_{k-2}^k\right)} & \alpha_6(\boldsymbol{\theta_k}) &= \left(T_d - t_{k-2}^k\right)\\ \alpha_3(\boldsymbol{\theta_k}) &= e^{\lambda_1\left(T_d - t_{k-1}^k\right)} & \alpha_7(\boldsymbol{\theta_k}) &= \left(T_d - t_{k-1}^k\right)\\ \alpha_4(\boldsymbol{\theta_k}) &= e^{\lambda_1\left(T_d - t_k^k\right)} & \alpha_8(\boldsymbol{\theta_k}) &= \left(T_d - t_k^k\right)\end{aligned} \quad \text{(A.43)}$$

which is the same result as the result obtained with Jordan Form strategy proposed in [8] and demonstrated in section A.1. The integrals (A.43) are rewritten using the time-varying function:

$$\int_{T_d-t_{k-3}^k}^{T_d} e^{\boldsymbol{A}_c \cdot s} \boldsymbol{b}_c ds = \frac{1}{\lambda_1} \boldsymbol{Q} \begin{bmatrix} T_d - \alpha_1 & 0 \\ 0 & T_d - \alpha_5 \end{bmatrix} \boldsymbol{Q}^{-1} \boldsymbol{b}_c \tag{A.44a}$$

$$\int_{T_d-t_{k-2}^k}^{T_d-t_{k-3}^k} e^{\boldsymbol{A}_c \cdot s} \boldsymbol{b}_c ds = \frac{1}{\lambda_1} \boldsymbol{Q} \begin{bmatrix} \alpha_1 - \alpha_2 & 0 \\ 0 & \alpha_5 - \alpha_6 \end{bmatrix} \boldsymbol{Q}^{-1} \boldsymbol{b}_c \tag{A.44b}$$

$$\int_{T_d-t_{k-1}^k}^{T_d-t_{k-2}^k} e^{\boldsymbol{A}_c \cdot s} \boldsymbol{b}_c ds = \frac{1}{\lambda_1} \boldsymbol{Q} \begin{bmatrix} \alpha_2 - \alpha_3 & 0 \\ 0 & \alpha_6 - \alpha_7 \end{bmatrix} \boldsymbol{Q}^{-1} \boldsymbol{b}_c \tag{A.44c}$$

$$\int_{T_d-t_k^k}^{T_d-t_{k-1}^k} e^{\boldsymbol{A}_c \cdot s} \boldsymbol{b}_c ds = \frac{1}{\lambda_1} \boldsymbol{Q} \begin{bmatrix} \alpha_3 - \alpha_4 & 0 \\ 0 & \alpha_7 - \alpha_8 \end{bmatrix} \boldsymbol{Q}^{-1} \boldsymbol{b}_c \tag{A.44d}$$

$$\int_{0}^{T_d-t_k^k} e^{\boldsymbol{A}_c \cdot s} \boldsymbol{b}_c ds = \frac{1}{\lambda_1} \boldsymbol{Q} \begin{bmatrix} 1 - \alpha_4 & 0 \\ 0 & \alpha_8 \end{bmatrix} \boldsymbol{Q}^{-1} \boldsymbol{b}_c \tag{A.44e}$$

These expressions are used to define the lifted model of the NCS which can be found in (A.49). For the controller synthesis is the complete model of the form (4.5) used, hence the matrices $\boldsymbol{F}_0$-$\boldsymbol{F}_5$ and $\boldsymbol{G}_0$-$\boldsymbol{G}_5$ do not have to be derived explicitly.

Remark A1:

Due to the simplicity of the chosen system in this example, it is possible to calculate the solution of the integrals analytically. However, for systems of the higher order, the calculation of analytical solutions for the integrals in the NCS model gets much more complicated. In this case, the efficiency and necessity of the real Jordan form (especially for matrices $\boldsymbol{A}_c$ with complex eigenvalues) soon becomes obvious.

A.3 Convex Over-approximation

Based in the set of vertices $\mathscr{FG}$ from (4.7) the nominal controller should be designed. However, since the uncertain parameters from (2.17) can take arbitrary values between the defined bounds, this set is infinite. In order to define a convex over-approximation $\mathscr{H}_{FG}$ which avoids the problem of infinite dimension, the maximum and minimal value of uncertain functions $\bar{\alpha}_i(\boldsymbol{\theta}_k)$ and $\underline{\alpha}(\boldsymbol{\theta}_k)$ with respect to $\boldsymbol{\theta}_k$ are needed. These can be determined from the boundaries defined for the arrival times t_j^k

$$0 \leq t_j^k \leq T_d \quad \forall j \in [k-\delta, k-\delta-1, \ldots, k] \tag{A.45}$$

$$\underline{\alpha}_{NZ,i} = e^{\lambda_1 T_d} = 0.3833 \qquad \bar{\alpha}_{NZ,i} = 1 \tag{A.46}$$

for $i \in \{1, 2, 3, 4\}$ and

$$\underline{\alpha}_{Z,i} = 0 \qquad \bar{\alpha}_{Z,i} = T_d \tag{A.47}$$

for $i \in \{5, 6, 7, 8\}$. The convex over-approximation $\mathscr{H}_{FG}$ consist therefore of $2^\beta = 2^8$ combinations of matrices and can be defined in Matlab, e.g., using cell arrays and `for` loop. The matrices $\boldsymbol{H}_{F,j}$ and $\boldsymbol{H}_{G,j}$ from (4.10) for $j \in \{1, 2, \ldots, 2^\beta\}$ are individual matrices from the defined convex over-approximation.

A.4 Lifted Model

$$\boldsymbol{\xi}_{k+1} = \begin{bmatrix} e^{\boldsymbol{A}_c T_d} & \int_{T_d - t_k^k}^{T_d - t_{k-1}^k} e^{\boldsymbol{A}_c \cdot s} \boldsymbol{b}_c ds & \int_{T_d - t_{k-1}^k}^{T_d - t_{k-2}^k} e^{\boldsymbol{A}_c \cdot s} \boldsymbol{b}_c ds & \int_{T_d - t_{k-2}^k}^{T_d - t_{k-3}^k} e^{\boldsymbol{A}_c \cdot s} \boldsymbol{b}_c ds & \int_{T_d - t_{k-3}^k}^{T_d} e^{\boldsymbol{A}_c \cdot s} \boldsymbol{b}_c ds \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \boldsymbol{\xi}_k + \begin{bmatrix} \int_0^{T_d - t_k^k} e^{\boldsymbol{A}_c \cdot s} \boldsymbol{b}_c ds \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} u_k \tag{A.48}$$

$$
\hat{\boldsymbol{A}} = \begin{bmatrix}
e^{\boldsymbol{A}_c T_d} & \frac{1}{\lambda_1}\boldsymbol{Q}\begin{bmatrix}\alpha_3-\alpha_4 & 0\\ 0 & \alpha_7-\alpha_8\end{bmatrix}\boldsymbol{Q}^{-1}\boldsymbol{b}_c & \frac{1}{\lambda_1}\boldsymbol{Q}\begin{bmatrix}\alpha_2-\alpha_3 & 0\\ 0 & \alpha_6-\alpha_7\end{bmatrix}\boldsymbol{Q}^{-1}\boldsymbol{b}_c & \frac{1}{\lambda_1}\boldsymbol{Q}\begin{bmatrix}\alpha_1-\alpha_2 & 0\\ 0 & \alpha_6-\alpha_7\end{bmatrix}\boldsymbol{Q}^{-1}\boldsymbol{b}_c & \frac{1}{\lambda_1}\boldsymbol{Q}\begin{bmatrix}T_d-\alpha_1 & 0\\ 0 & T_d-\alpha_5\end{bmatrix}\boldsymbol{Q}^{-1}\boldsymbol{b}_c \\
\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\
\boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \\
\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0} & \boldsymbol{0} \\
\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{1} & \boldsymbol{0}
\end{bmatrix}
$$

$$
\hat{\boldsymbol{b}} = \begin{bmatrix}
\frac{1}{\lambda_1}\boldsymbol{Q}\begin{bmatrix}1-\alpha_4 & 0\\ 0 & -\alpha_8\end{bmatrix}\boldsymbol{Q}^{-1}\boldsymbol{b}_c \\
\boldsymbol{1} \\
\boldsymbol{0} \\
\boldsymbol{0} \\
\boldsymbol{0}
\end{bmatrix}
\tag{A.49}
$$

B
Relevant URLs

Contents

B.1 Arduino

- Arduino MKR1000 `https://store.arduino.cc/arduino-mkr1000-wifi`
- WiFi library `https://www.arduino.cc/en/Reference/WiFi`
- WiFi101 library `https://www.arduino.cc/en/Reference/WiFi101`[1]

B.2 Simulink blocks

- Buffer `https://de.mathworks.com/help/dsp/ref/buffer.html`

B.3 Matlabs/Simulink with Arduino Hardware

- Getting Started with Arduino® Hardware `https://de.mathworks.com/help/supportpkg/arduino/ref/getting-started-with-arduino-hardware.html`

[1] Simulink Support Package requires this library to be of the the specific version 0.14.3

- Getting Started with WiFi on Arduino® Hardware: https://de.mathworks.com/help/supportpkg/arduino/examples/getting-started-with-wifi-on-arduino-hardware.html
- Simulink block WiFi UDP Receive https://de.mathworks.com/help/supportpkg/arduino/ref/wifiudpreceive.html?s_tid=srchtitle
- Simulink block WiFi UDP Send https://de.mathworks.com/help/supportpkg/arduino/ref/wifiudpsend.html?s_tid=srchtitle
- Simulink block UDP Receive https://de.mathworks.com/help/instrument/udpreceive.html?s_tid=srchtitle
- Simulink block UDP Send https://de.mathworks.com/help/instrument/udpsend.html?s_tid=srchtitle

B.4 Putty

- Putty https://www.putty.org/
- Log all session output by default https://www.viktorious.nl/2013/01/14/putty-log-all-session-output/
- Load a session automatically https://kb.norsetech.net/how-to-make-putty-automatically-load-a-session/

B.5 Speed Test

- App https://www.speedtest.net/apps

C

Adaptation of the Simulation Toolbox [6]

Contents

C.1 Message Rejection Mechanism

`NCS_ISMC`

- define property `msg_rej_nodes`
- `function generateISMNodes()`
  - define `network_msg_rej_nodenumber` to be equal to `act_number+4`; buffer and act node number should be `act_number+5` and `act_number+6` respectively
  - include `obj.msg_rej_nodes{i} = MsgRejection(network_buffer_nodenumber, network_msg_rej_nodenumber, ith_ncs_problem.m);`
  - re-define `tau_ca_nodes{i}` so that it sends data to message rejection node
- `function out = get.allnodes(obj)`: include `obj.msg_rej_nodes` in the variable `out`

Simulink

- define additional TrueTime Kernel with node object `ism_sys_obj.msg_rej_nodes{i}`

C.2 Modified Buffering Mechanism

- `NetworkBuffer`: set `transmit_time = ceil(currenttime/obj.Td)*obj.Td;`

C.3 Control Law

C.3.1 Nominal Control Law

Since solving LMIs requires long computation time, the controller gain is first determined and than directly used as an input in the class `NCS_ISMC`.

- define new function `designK_Direct` with `obj.K = directK`

C.3.2 Sliding Mode Control Law

- Simulink: send delayed (**Unit delay**) output of the Network Buffer node to Sensor Node as the last input
- class `SensorNode`
  - set `starttime` to 1e-9
  - define vector m^T in properties
  - calculate s_k from inputs and send it to the delay node (`tx_data(end) = obj.mT*tx_data(1:2).'-tx_data(end)`)
- class `ISM_ControllerNode`: define σ_k and w_{k+1} in `function sigmaik = calc_sliding_var(obj,xik,uiNk,sigmak_temp)`

References

[1] X. Zhang, Q. Han, and X. Yu. "Survey on Recent Advances in Networked Control Systems". In: *IEEE Transactions on Industrial Informatics* 12.5 (2016), pp. 1740–1752.

[2] W. P.M.H. Heemels, N. Van De Wouw, R. H. Gielen, M. C.F. Donkers, L. Hetel, S. Olaru, M. Lazar, J. Daafouz, and S. Niculescu. "Comparison of overapproximation methods for stability analysis of networked control systems". In: *HSCC'10 - Proceedings of the 13th ACM International Conference on Hybrid Systems*. United States: Association for Computing Machinery, Inc, June 2010, pp. 181–190.

[3] J. Ludwiger, M. Steinberger, M. Rotulo, M. Horn, A. Luppi, G. Kubin, and A. Ferrara. "Towards networked sliding mode control". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. 2017, pp. 6021–6026.

[4] J. Ludwiger, M. Steinberger, M. Horn, G. Kubin, and A. Ferrara. "Discrete Time Sliding Mode Control Strategies for Buffered Networked Systems". In: *2018 IEEE Conference on Decision and Control (CDC)*. 2018, pp. 6735–6740.

[5] J. Ludwiger, M. Steinberger, and M. Horn. "Spatially Distributed Networked Sliding Mode Control". In: *IEEE Control Systems Letters* 3.4 (2019), pp. 972–977.

[6] J. Ludwiger. "Discrete Time Sliding Mode Controller Design for Networked Control Systems". PhD thesis. Graz University of Technology, 2020.

[7] M.B.G. Cloosterman, L. Hetel, N. van de Wouw, W.P.M.H. Heemels, J. Daafouz, and H. Nijmeijer. "Controller synthesis for networked control systems". In: *Automatica* 46.10 (2010), pp. 1584–1594.

[8] M.B.G. Cloosterman. "Control over communication networks : modeling, analysis, and synthesis". PhD thesis. Technische Universiteit Eindhoven, 2008.

[9] V. I. Utkin. "Sliding mode control design principles and applications to electric drives". In: *IEEE Transactions on Industrial Electronics* 40.1 (Feb. 1993), pp. 23–36.

[10] V. Utkin and Jingxin Shi. "Integral sliding mode in systems operating under uncertainty conditions". In: *Proceedings of 35th IEEE Conference on Decision and Control*. Vol. 4. Dec. 1996, 4591–4596 vol.4.

[11] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*. Vol. 15. Studies in Applied Mathematics. Philadelphia, PA: SIAM, June 1994.

[12] J. Löfberg. "YALMIP : A Toolbox for Modeling and Optimization in MATLAB". In: *In Proceedings of the CACSD Conference*. Taipei, Taiwan, 2004.

[13] S. Koch and M. Reichhartinger. "Discrete-time equivalents of the super-twisting algorithm". In: *Automatica* 107 (2019), pp. 190–199.

[14] S. Koch, M. Reichhartinger, and M. Horn. “On the Discretization of the Super-Twisting Algorithm”. In: *2019 IEEE 58th Conference on Decision and Control (CDC)*. Dec. 2019, pp. 5989–5994.

[15] R. Seeber and M. Horn. “Stability proof for a well-established super-twisting parameter setting”. In: *Automatica* 84 (2017), pp. 241–243.

[16] N.J. Ploplys and A. G. Alleyne. “UDP Network Communications for Distributed Wireless Control”. English (US). In: *Proceedings of the American Control Conference* 4 (Nov. 2003). 2003 American Control Conference ; pp. 3335–3340.

[17] B. Winkler. “Regelung eines Servoantriebs”. Bachelor’s Thesis. Graz University of Technology, 2010.

[18] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K.-E. Årzén. “How Does Control Timing Affect Performance?” In: *Control Systems Magazine* 23.3 (2003), pp. 16–30.