Modeling and predictive control of a hybrid system based on an identified model

Vet, B.

Award date:
2006

Eindhoven University of Technology
Department of Electrical Engineering
Control Systems

TU/e technische universiteit eindhoven

Modeling and predictive control of a hybrid system based on an identified model

by

Bas Vet

Master of Science thesis

Project period: november 2005
Report Number: 05A/09
Commissioned by: prof.dr.ir. P.P.J. van den Bosch

Additional Commission members:
Dr. S. Weiland (TU/e CS)
Dr. A. Juloski (TU/e CS)
M. Lazar, M.Sc. (TU/e CS)

Part I

On the (in)compatibility of PWA and PWARX models

B. Vet A. Lj. Juloski

November 23, 2005

Abstract

In the field of control of hybrid systems several model classes are used, which are proven to be equivalent. One of these model classes is the piecewise affine model (PWA). Identification procedures for hybrid systems use another model which is in the input/output form, the piecewise autoregressive exogenous model (PWARX). In literature on these procedures, the PWA and PWARX are considered equivalent.

In this report however is proven this is not the case and conditions are given when equivalence holds. The differences between the models are analyzed and a new model is introduced that solves the compatibility issues.

With the analysis in this report the identification methods for hybrid systems may be improved.

Acknowledgements

During the project I have received encouragement and assistance from my supervisors to whom I would like to express my appreciation and gratitude.

First of all I would like to thank Prof. Dr. Ir. P.P.J. van den Bosch for both his advice on the technical content of the project, but especially for his stimulating encouragements.

I am thankful to my supervisors M.Sc. M. Lazar and Dr. Ir. A. Lj. Juloski for there encouragements and scientific suggestions. Their critical remarks have been of value to my project. With Juloski's advice I have been able to rewrite my findings on the PWA and PWARX models into that a mathematical correct and complete report. Lazar has had duty's elsewhere during a large part of the project, but he still made time to give me useful advice.

Also I would like to thank my fellow graduating students for the fruitful discussions and the good working atmosphere.

Contents

Introduction

The field of control for hybrid systems is relatively young, however a firm theoretical base has been founded. Today there exists several models of hybrid systems, amongst which: Piecewise Affine (PWA) Models, Mixed Logical Dynamical (MLD) Models, Linear Complementarity (LC) Models, Extended Linear Complementarity (ELC) and Models Max-Min-Plus-Scaling (MMPS) Models. The above models are equivalent(under certain constraints); properties and tools can be transferred between these models [3].

Another class of hybrid models exists, the Piecewise Autoregressive Exogenous (PWARX) Model. This model is in input/output form, which makes the model convenient for black-box identification of hybrid systems [4]. When using the PWARX model for these purposes, researchers implicitly (or explicitly) consider the PWARX model equivalent or at least compatible with the hybrid system that is being identified (eg [5] and [1] and explicit: [2]. The resulting PWARX models are considered to be compatible with other models of hybrid systems, so that they can be used to apply the control theory. However, no proof has been stated that PWARX models are compatible with the PWA and related hybrid models.

In this paper the PWARX model is discussed and the relation between PWARX models and PWA models is analyzed. Furthermore proof is given that not all PWA systems can be modeled with a PWARX model, general conditions for compatibility between PWA systems (and others) are given, the PWARX++ model is introduced that resolves the compatibility issues between PWARX models and PWA systems and a PWA++ model is introduced that allows for different descriptions of a PWA system, by introducing state resets.

Chapter 1

Problem Statement

Problem Environment:
In this paper 4 models of hybrid systems are used, a piecewise state space system (System 1.1), a piecewise autoregressive exogenous system (System 1.2) and two corresponding systems that have been extended with a state reset (System 1.3) and regressor reset (System 1.4). Each of these systems have zero, one or more inputs and 1 output (MISO systems). The mathematical definitions of these systems are given below:

System 1.1 Piecewise Affine System

$$\begin{aligned} x(k+1) &= A_{\sigma(k+1)}x(k) + B_{\sigma(k+1)}u(k) \\ z(k) &= C_{\sigma(k)}x(k) + D_{\sigma(k)}u(k) \end{aligned} \tag{1.1}$$

where $\sigma(k) : x(k) \mapsto \{1, \ldots, s\}$ maps polyhedral regions of x to a mode, $k \in \mathbb{N} \cup 0$, $A_i \in \mathbb{R}^{m \times m}$, $B_i \in \mathbb{R}^{m \times p}$, $C_i \in \mathbb{R}^m$, $D_i \in \mathbb{R}^p$, $x(k) \in \mathbb{R}^m$ is the state of the system, $u(k) \in \mathbb{R}^p$ consists of the values of the inputs at time k, s is the number of modes, p is the number of inputs, m is the order of the system and $x(0)$ is given. $z(k) \in \mathbb{R}$ is the output of the system.

System 1.2 Piecewise Autoregressive Exogenous System

$$\begin{aligned} y(k) &= \Theta_{\sigma(k)}\varphi(k) \\ \varphi(k) &= \begin{bmatrix} y(k-1) \\ \vdots \\ y(k-n) \\ u_1(k) \\ \vdots \\ u_1(k-n) \\ \vdots \\ u_p(k) \\ \vdots \\ u_p(k-n) \end{bmatrix} \end{aligned} \tag{1.2}$$

where $\sigma(k) : \varphi(k) \mapsto \{1, \ldots, s\}$ maps polyhedral regions of φ to a mode, $\Theta_{\sigma(k)} \in \mathbb{R}^{pn+p+n}$, $k \in \mathbb{N} \cup 0$, $\varphi(k) \in \mathbb{R}^{pn+p+n}$ is the regressor, p is the number of inputs, s is the number of modes, n is the order of the system and $\varphi(0)$ is given. $y(k) \in \mathbb{R}$ is the output of the system.

System 1.3 Piecewise Affine System with state reset

$$\begin{aligned} x(k+1) &= A_{\sigma(k+1)}\Re_{\sigma(k+1)\sigma(k)}x(k) + B_{\sigma(k+1)}u(k) \\ z(k) &= C_{\sigma(k)}x(k) + D_{\sigma(k)}u(k) \end{aligned} \tag{1.3}$$

where $\sigma(k) : x(k) \mapsto \{1, \ldots, s\}$ maps polyhedral regions of x to a mode, $k \in \mathbb{N} \cup 0$, $A_i \in \mathbb{R}^{m \times m}$, $B_i \in \mathbb{R}^{m \times p}$, $C_i \in \mathbb{R}^m$, $D_i \in \mathbb{R}^p$, $x(k) \in \mathbb{R}^m$ is the state of the system, $\Re_{ij} : \mathbb{R}^m \mapsto \mathbb{R}^m$ is the regressor reset map,$\Re_{ii} = I$, $u(k) \in \mathbb{R}^p$ consists of the values of the inputs at time k, s is the number of modes, p is the number of inputs, m is the order of the system and $x(0)$ is given. $z(k) \in \mathbb{R}$ is the output of the system.

System 1.4 Piecewise Autoregressive Exogenous System with regressor reset

$$
\begin{aligned}
y(k) &= \Theta_{\sigma(k)}\varphi(k) \\
\varphi(k) &= \Re_{\sigma(k)\sigma(k-1)}\left(\begin{bmatrix} S & 0 & \dots & 0 \\ 0 & S & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \end{bmatrix}\varphi(k-1)+\right. \\
&\qquad \left. + \begin{bmatrix} I \\ 0 \\ \vdots \\ 0 \end{bmatrix} y(k-1) + \begin{bmatrix} 0 & \dots & 0 \\ I & \dots & \vdots \\ \vdots & \dots & 0 \\ 0 & \dots & I \end{bmatrix} u(k)\right) \\
S &= \begin{bmatrix} 0 & \dots & \dots & 0 \\ 1 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \vdots \\ 0 & 0 & 1 & 0 \end{bmatrix} \\
I &= \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}
\end{aligned}
\tag{1.4}
$$

where $\sigma(k) : \varphi(k) \mapsto \{1, \dots, s\}$ maps polyhedral regions of φ to a mode, $\Theta_{\sigma(k)} \in \mathbb{R}^{pn+p+n}$, $k \in \mathbb{N} \cup 0$, $\Re_{ij} : \mathbb{R}^m \mapsto \mathbb{R}^m$ is the regressor reset map,$\Re_{ii} = I$, $\varphi(k) \in \mathbb{R}^{pn+p+n}$ is the regressor, p is the number of inputs, s is the number of modes, n is the order of the system and $\varphi(0)$ is given. $y(k) \in \mathbb{R}$ is the output of the system. The S and I combinations act as shift registers. All of these matrices correspond to one of the inputs and have a size of $n+1$, except for the first one. The first shift register corresponds to the output part of the regressor and thus has size n. When n instances have passed since the last mode-switch, the contents of the regressor are similar to that in System 1.2. The definition above allows for resetting the regressor with a reset map ($\Re_{ij}$).

Problem 1 *Given System 1.2 (Θ_i) and $\varphi(0)$, is it possible to find a System 1.1 (A_i, B_i, C_i, D_i) and $x(0)$ such that $y(k) = z(k) \forall k \in \mathbb{N}$*

Problem 1a *Given System 1.2 (Θ_i) and $\varphi(0)$, is it possible to find a System 1.1(A_i, B_i, C_i, D_i) of minimal order and $x(0)$ such that $y(k) = z(k) \forall k \in \mathbb{N}$, where minimal means that the orders of the systems are identical, ie. $m = n$*

Problem 2 *Given System 1.1(A_i, B_i, C_i, D_i) and $x(0)$, is it possible to*

find a System 1.2 (Θ_i) and $\varphi(0)$, such that $z(k) = y(k) \forall k \in \mathbb{N}$

Problem 2a *Given System 1.1(A_i, B_i, C_i, D_i) and $x(0)$, is it possible to find a System 1.2 (Θ_i)of minimal order and $\varphi(0)$, such that $z(k) = y(k) \forall k \in \mathbb{N}$, where minimal means that the orders of the systems are identical, ie. $m = n$*

Problem 3 *Given System 1.2 (Θ_i) and $\varphi(0)$, is it possible to find a System 1.3 (A_i, B_i, C_i, D_i, $\Re_{ij}$) and $x(0)$ such that $y(k) = z(k) \forall k \in \mathbb{N}$*

Problem 3a *Given System 1.2 (Θ_i) and $\varphi(0)$, is it possible to find a System 1.3(A_i, B_i, C_i, D_i, $\Re_{ij}$) of minimal order and $x(0)$ such that $y(k) = z(k) \forall k \in \mathbb{N}$, where minimal means that the orders of the systems are identical, ie. $m = n$*

Problem 4 *Given System 1.1(A_i, B_i, C_i, D_i) and $x(0)$, is it possible to find a System 1.4 (Θ_i, $\Re_{ij}$) and $\varphi(0)$, such that $z(k) = y(k) \forall k \in \mathbb{N}$*

Problem 4a *Given System 1.1(A_i, B_i, C_i, D_i) and $x(0)$, is it possible to find a System 1.4 (Θ_i, $\Re_{ij}$)of minimal order and $\varphi(0)$, such that $z(k) = y(k) \forall k \in \mathbb{N}$, where minimal means that the orders of the systems are identical, ie. $m = n$*

Problem 5 *Given System 1.1(A_i, B_i, C_i, D_i) and $x_1(0)$, and a set of transformation matrices T_i that transform each subsystem into a specific form, is it possible to find a System 1.3 ($\Re_{ij}$)and $x_2(0)$, such that $y_1(k) = y_2(k) \forall k \in \mathbb{N}$*

Chapter 2

Conditions for compatibility

In this section the general conditions for compatibility between the PWA and PWARX systems (respectively System 1.1 and 1.2) are deduced. The problem is approached by finding an expression for the state of the state space system in terms of previous in- and output values, ergo in terms of a regressor. For notation purposes the following definition is posed:

Definition 1 *Let $\sigma(k)$ be the mode of a system at time $k \in \mathbb{N}$ than define the series*

$$\Sigma_i^k = \quad \sigma(k) \quad \sigma(k-1) \quad \dots \quad \sigma(i)$$

and define:

$$A^{\Sigma_i^k} = A_{\sigma(k)} \cdot A_{\sigma(k-1)} \dots A_{\sigma(i)}$$

To find the relation between the state and the previous in- and output values, the following array of output values is used.

$$\begin{aligned}
z(k-p) &= C_{\sigma(k-p)}x(k-p) + D_{\sigma(k-p)}u(k-p) \\
z(k-p+1) &= C_{\sigma(k-p+1)}A_{\sigma(k-p+1)}x(k-p) + \\
&\quad C_{\sigma(k-p+1)}B_{\sigma(k-p+1)}u(k-p) + \\
&\quad D_{\sigma(k-p+1)}u(k-p+1) \\
&\vdots \quad \vdots \\
z(k-1) &= C_{\sigma(k-1)}A^{\Sigma_{k-p+1}^{k-1}}x(k-p) + \\
&\quad C_{\sigma(k-1)}A^{\Sigma_{k-p+2}^{k-1}}B_{\sigma(k-p+1)}u(k-p) + \\
&\quad \dots + C_{\sigma(k-1)}B_{\sigma(k-1)}u(k-2) \\
&\quad + D_{\sigma(k-1)}u(k-1)
\end{aligned} \tag{2.1}$$

In the equations above $p = n + r$, with $n \in \mathbb{N}$ is the order of the system and $r \in \mathbb{N} \cup 0$ is used as a parameter that introduces redundancy to the set of

equations. To write the state in regressor terms the equations above are solved towards regressor terms. The following is obtained:

$$\Lambda x(k-p) = \left(\begin{bmatrix} z(\kappa) \\ z(k-2) \\ \vdots \\ z(k-p) \end{bmatrix} - \Omega \begin{bmatrix} u(\kappa) \\ u(k-2) \\ \vdots \\ u(k-p) \end{bmatrix} \right) \tag{2.2}$$

with:

$$\Lambda = \begin{bmatrix} C_{\sigma(\kappa)} A^{\Sigma_{k-p+1}^{\kappa}} \\ \vdots \\ C_{\sigma(k-p+1)} A_{\sigma(k-p+1)} \\ C_{\sigma(k-p)} \end{bmatrix} \tag{2.3}$$

and:

$$\Omega = \begin{bmatrix} D_{\sigma(\kappa)} & C_{\sigma(\kappa)} B_{\sigma(\kappa)} & \dots & C_{\sigma(\kappa)} A^{\Sigma_{k-p+2}^{\kappa}} B_{\sigma(k-p+1)} \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & C_{\sigma(k-p+1)} B_{\sigma(k-p+1)} \\ 0 & \dots & 0 & D_{\sigma(k-p)} \end{bmatrix} \tag{2.4}$$

Please NOTE that $\kappa = k - 1$, this substitution had to be made to fit the matrix above within the column. The system of equations above can be solved into the following vector space:

$$x_{sol}(k-p) = x_{LS} + h \quad | \quad \Lambda h = \mathbf{0} \tag{2.5}$$

where x_{LS} is the minimal least square solution, which can be calculated with the Moore-Penrose Matrix Inverse.

When solving the system of equations 2.2 the following three cases can be distinguished

- The system is uniquely determinde and $h = 0$ (rank=n)
- The system of equations is overdetermined and no solution exists. However this situation cannot occur; Due to the nature of the problem it is obvious that at least one solution does exist (rank¿n).
- The system of equations has interdependent rows and is underdetermined (rank¡n). In this case 2.5 is the solution and $h \neq 0$.

When the solution is combined with the equations:

$$\begin{aligned} x(k) = {} & A^{\Sigma_{k-p+1}^{k}} x_{LS}(k-p) + A^{\Sigma_{k-p+1}^{k}} h(k-p) \\ & A^{\Sigma_{k-p+2}^{k}} B_{\sigma(k-p+1)} u(k-p) \dots + B_{\sigma(k)} u(k-1) \end{aligned} \tag{2.6}$$

and

$$z(k) \quad = \quad C_{\sigma(k)} x(k) + D_{\sigma(k)} u(k) \tag{2.7}$$

the following result is obtained:

$$z(k) = R_{\Sigma_{k-p}^k} \begin{bmatrix} z(k-1) \\ z(k-2) \\ \vdots \\ z(k-p) \end{bmatrix} + S_{\Sigma_{k-p}^k} \begin{bmatrix} u(k) \\ u(k-1) \\ \vdots \\ u(k-p) \end{bmatrix} \tag{2.8}$$

Where $R_{\Sigma_{k-p}^k}$ and $S_{\Sigma_{k-p}^k}$ are matrices that depend on the history of the switching sequence.

Condition 1 *The matrices $R_{\Sigma_{k-p}^k}$ and $S_{\Sigma_{k-p}^k}$ are unique if $r = 0$ and $C(k)A^{\Sigma_{k-p+1}^k}h = 0$.*

In case condition 1 is satisfied, the matrices $R_{\Sigma_{k-p}^k}$ and $S_{\Sigma_{k-p}^k}$ are given by:

$$R_{\Sigma_{k-p}^k} = C_{\sigma(k)}A^{\Sigma_{k-p+1}^k} \begin{bmatrix} C_{\sigma(k)}A^{\Sigma_{k-p+1}^{k-1}} \\ \vdots \\ C_{\sigma(k-p+1)}A_{\sigma(k-p+1)} \\ C_{\sigma(k-p)} \end{bmatrix}^{-1} \tag{2.9}$$

$$\begin{aligned} S_{\Sigma_{k-p}^k} &= \begin{bmatrix} 0 & \cdots & 0 \\ 0 & R_{\Sigma_{k-p}^k}\Omega_{\Sigma_{k-p}^k} \end{bmatrix} \\ &+ \begin{bmatrix} D_{\sigma(k)} & 0 & \dots & 0 \\ 0 & B_{\sigma(k)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & A^{\Sigma_{k-p+2}^k}B_{\sigma(k-p+1)} \end{bmatrix} \end{aligned} \tag{2.10}$$

where the inverse can be any matrix-1 inverse, for instance the Moore-Penrose inverse.

When the relation of equation 2.8 is compared with the corresponding relation of an PWARX systems given below, the conclusion is that PWARX systems throw away information about the history of the mode sequence, whereas PWA systems preserve this information.

$$\begin{aligned} y(k) = \Bigg(& \Theta_{y,\sigma(k)} \begin{bmatrix} y(k-1) \\ y(k-2) \\ \vdots \\ y(k-p) \end{bmatrix} \\ & + \Theta_{u,\sigma(k)} \begin{bmatrix} u(k) \\ u(k-1) \\ \vdots \\ u(k-p) \end{bmatrix} \Bigg) \end{aligned} \tag{2.11}$$

When condition 1 is met, from these two equations two conditions can be deduced, which need to be met in order for an PWARX system to have identical input/output behavior with respect to a PWA system:

$$R_{\Sigma_{k-p}^k} = \Theta_{y,\sigma(k)} \quad {}^\forall\Sigma_{k-p}^k \ \wedge \ {}^\forall\sigma(k) \tag{2.12}$$

$$S_{\Sigma_{k-p}^k} = \Theta_{u,\sigma(k)} \quad {}^\forall\Sigma_{k-p}^k \ \wedge \ {}^\forall\sigma(k) \tag{2.13}$$

In some systems the sequence of modes is restricted, so that some sequences do not occur. These forbidden mode sequences do not compromise the compatibility and the compatibility conditions above account for this situation.

When comparing two state space systems, the most general conditions for two arbitrary switched or piecewise state space system to have identical behavior are presented in the following equations:

$$C_{1,\sigma_1(k)}R_{1,\Sigma_{k-p}^k} = C_{2,\sigma_2(k)}R_{2,\Sigma_{k-p}^k} \quad {}^\forall\Sigma_{k-p}^k \ \wedge \ {}^\forall\sigma_{1,2}(k) \tag{2.14}$$

$$C_{1,\sigma_1(k)}S_{1,\Sigma_{k-p}^k} = C_{2,\sigma_2(k)}S_{2,\Sigma_{k-p}^k} \quad {}^\forall\Sigma_{k-p}^k \ \wedge \ {}^\forall\sigma_{1,2}(k) \tag{2.15}$$

Chapter 3

Examples

In this section proof is given of the existence of systems that comply to the conditions of equation 2.12 and 2.13. This section also provides a counter-example that shows that not all systems meet these conditions.

3.1 An example for Problem 1.1

At first a PWARX system without input is examined. The PWA model of an PWARX system follows directly from the definition of the regressor of system 1.4. In this system the regressor can be calculated from the previous regressor by using a shift matrix:

$$\begin{aligned} y(k) &= \Theta_{\sigma(k)}\varphi(k) \\ \varphi(k) &= \begin{bmatrix} 0 & \dots & \dots & \dots & 0 \\ 1 & \ddots & & & \vdots \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & 0 \end{bmatrix} \varphi(k-1) \\ &\quad + \begin{bmatrix} 1 \\ 0 \\ \vdots \\ \vdots \\ 0 \end{bmatrix} y(k) \end{aligned} \tag{3.1}$$

when substituting $y(k)$ in the formula for $\varphi(k)$ the following state space system is obtained:

$$\begin{aligned}\varphi(k+1) &= \begin{bmatrix} \theta_{1,\sigma(k+1)} & \cdots & \cdots & \cdots & \theta_{n,\sigma(k+1)} \\ 1 & 0 & \dots & \dots & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & 0 \end{bmatrix} \varphi(k) \\ &= Q_{\sigma(k)}\varphi(k) \\ y(k) &= \Theta_{\sigma(k)}\varphi(k)\end{aligned} \tag{3.2}$$

According to the previous section, the model obtained above has to comply with the conditions of equation 2.12 and 2.13. Since the system has no input the second condition is met inherently, leaving only only the first condition:

$$R_1 = R_2 \tag{3.3}$$

First a closer look is taken at R for the special matrix Q as defined in equation 3.2.

$$R = Q^{\Sigma_{k-p+2}^{k}} \begin{bmatrix} \Theta_{\sigma(k)} Q^{\Sigma_{k-p+2}^{k-1}} \\ \vdots \\ \Theta_{\sigma(k-p+2)} Q_{\sigma(k-p+2)} \\ \Theta_{\sigma(k-p+1)} \end{bmatrix}^{-1}$$

Due to the shape of the matrix, Q has two special properties:

$$\begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix} Q = \Theta \tag{3.4}$$

and:

$$\begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ 0 & \dots & \dots & 0 & 1 \end{bmatrix} Q = \begin{bmatrix} 1 & 0 & \dots & \dots & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & 0 \end{bmatrix} \tag{3.5}$$

$Q^{\Sigma_{k-p+2}^{k}}$ can be expanded from $Q_\sigma(k)$ as follows:

$$Q_{\sigma(k)} = \begin{bmatrix} \Theta_{\sigma(k)} \\ \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \end{pmatrix} \\ \vdots \qquad\qquad \vdots \\ \begin{pmatrix} 0 & \dots & 0 & 1 & 0 \end{pmatrix} \end{bmatrix}$$

$$
\begin{aligned}
Q_{\sigma(k)}Q_{\sigma(k-1)} &= \begin{bmatrix} \left(\begin{array}{cccc} 1 & 0 & \dots & 0 \end{array}\right) Q_{\sigma(k)}Q_{\sigma(k-1)} \\ \left(\begin{array}{cccc} 1 & 0 & \dots & 0 \end{array}\right) Q_{\sigma(k-1)} \\ \left(\begin{array}{cccc} 1 & 0 & \dots & 0 \end{array}\right) \\ \vdots \qquad\qquad \vdots \\ \left(\begin{array}{cccc} 0 & \dots & 0 & 1 \quad 0 \end{array}\right) \end{bmatrix} \\
Q^{\Sigma_{k-p+1}^{k}} &= \begin{bmatrix} \left(\begin{array}{cccc} 1 & 0 & \dots & 0 \end{array}\right) Q^{\Sigma_{k-p+2}^{k}} \\ \left(\begin{array}{cccc} 1 & 0 & \dots & 0 \end{array}\right) Q^{\Sigma_{k-p+2}^{k-1}} \\ \vdots \qquad\qquad \vdots \\ \left(\begin{array}{cccc} 1 & 0 & \dots & 0 \end{array}\right) Q_{\sigma(k-p+2))} \\ \left(\begin{array}{cccc} 1 & 0 & \dots & 0 \end{array}\right) \end{bmatrix} \\
&= \begin{bmatrix} \Theta_{\sigma(k)} Q^{\Sigma_{k-p+2}^{k-1}} \\ \vdots \\ \Theta_{\sigma(k-p+2)} Q_{\sigma(k-p+2)} \\ \Theta_{\sigma(k-p+1)} \end{bmatrix}
\end{aligned} \tag{3.6}
$$

The last equation shows that $R = I$ and is independent of the the values in Q and proves all conditions for compatibility are been met by the system described above.

The state space description of the PWARX model as described above can be extended to be with input, so that the following model is obtained:

$$
\begin{aligned}
\varphi(k+1) &= \left(\begin{bmatrix} S & 0 & \dots & 0 \\ 0 & S & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \end{bmatrix} + \begin{bmatrix} \Theta_{\sigma(k+1)} \\ 0 \\ \vdots \\ 0 \end{bmatrix} \right) \varphi(k) \\
y(k) &= \Theta_{\sigma(k)} \varphi(k) \\
S &= \begin{bmatrix} 0 & \dots & \dots & 0 \\ 1 & \ddots & & \vdots \\ 0 & \ddots & \ddots & \vdots \\ 0 & 0 & 1 & 0 \end{bmatrix} \\
I &= \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}
\end{aligned} \tag{3.7}
$$

This model is a state space model that has exactly the same input/output behavior as the corresponding PWARX model. Therefore this model must comply with the three conditions. However no way to verify this has yet been found, because the A (or Q) matrix of this model is rank deficient and therefor the matrix in R that needs to be reversed is rank deficient. This means that there

are several possible pseudo-inverses, however it has not yet been possible to find an expression for these inverses.

3.2 A counter example

The examples above show that every PWARX model can be written into a PWA model that has identical input/output behavior. However the opposite is not true as is shown below with a counter example. The counter-example consists of a class of PWA system without input of which can be shown that there exist no PWARX system that has the same input/output behavior.

To find this class of systems, we are going to use two commutation-related properties of matrices:

Property 1 $^{\exists}M_1, M_2$ *such that* $M_1M_2 = M_2M_1$

Property 2 $M_1M_2 = M_2M_1 \iff {}^{\exists}T$ *such that both* TM_1T^{-1} *and* TM_2T^{-1} *are diagonal matrices.*

When using Property 2, it is easy to see that a System 1.2 which has the commutation property (Property 1). To generate such a system just create a set of diagonal eigenvalue matrices and a random matrix T and apply Property 2. The effect of Property 1 on the output of the system is shown in the following equation:

$$\begin{aligned} y(0) &= C_{\sigma(0)}x(0) \\ y(1) &= C_{\sigma(1)}A_{\sigma(1)}x(0) \\ y(2) &= C_{\sigma(2)}A_{\sigma(2)}A_{\sigma(1)}x(0) \\ y(3) &= C_{\sigma(3)}A_{\sigma(3)}A_{\sigma(2)}A_{\sigma(1)}x(0) \\ y(4) &= C_{\sigma(4)}A_{\sigma(4)}A_{\sigma(3)}A_{\sigma(2)}A_{\sigma(1)}x(0) \end{aligned} \tag{3.8}$$

Now as long as Property 1 holds $y(4)$ is identical for $\sigma(k) = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 \end{bmatrix}$, $\sigma(k) = \begin{bmatrix} 2 & 1 & 1 & 1 & 2 \end{bmatrix}$ and $\sigma(k) = \begin{bmatrix} 1 & 2 & 1 & 2 & 1 \end{bmatrix}$ etc... In fact the order in which the modes occur are irrelevant, as long as each mode is occurs just as much. This implies that if there is a equivalent description in the form of System 1.2 that this description must also commute and that the order of the modes is also irrelevant for this description.

As is shown in the previous section the PWARX model of System 1.2 can be written in the form of equation 3.2. So for the PWARX model to be equivalent to the commuting system described above, the Q_i matrices must commute. However it can be shown that this is not the case (see Appendix B), so that it is proven that there exist switched state space systems that cannot be described by System 1.2. $\triangle$

Chapter 4

Solutions for the compatibility issues

4.1 PWARX++

In this section System 1.4 is examined. This is PWARX-like system, the only difference is the regressor reset which occurs at mode-switches. Below is described how to dimension a System 1.4 such that the input/output behavior is identical to that of a reference state space system.

First of all it is important to note that there exists a regressor reset at each mode-switch that ensures that the output of the system is identical to the corresponding state space system until the next mode-switch:

Theorem 1 *Given a PWA system and a corresponding PWARX model, there exists a regressor reset at each mode-switch such that the PWARX model behaves identical to the PWA system in terms of output behaviour until the next mode-switch.*

Remark The theorem above implies that a regressor reset exists; it does not say that there exists a regressor reset map, which would be a function that is constant with respect to the model and over time.

Proof: A given state space system can be modeled by a ARX (ARMA) model; When the regressor of the ARX model is properly initialized, the ARX model behaves identical state space system in terms of output behaviour. So when properly initializing the regressor of the the PWARX model at each mode switch, the model can behave identical to a PWA system.

In fact it can be shown that there exist multiple solutions:

Proposition 2 *Given a PWA system of order N and a corresponding PWARX system (, of order $(p+1)N$), there exist a space of regressors, which is caused by the $(p+1)N - N$ redundancy of the regressor as description of the current condition of the system.*

To find the solution to Problem 2.1, it must be examined what causes the discrepancy between the two systems of examination, as was done in the previous section. In a normal state space system state space transformations can be performed:

$$\left.\begin{array}{rcl} & X & \Rightarrow \quad TZ \\ \dot{X} = AX + BU & & \\ Y = CX & \Rightarrow & \left\{ \begin{array}{l} \dot{Z} = T^{-1}ATZ + T^{-1}BU \\ Y = CTX \end{array} \right. \end{array}\right\} \tag{4.1}$$

This relation remains to be partially true in PWA systems, however

$$\left.\begin{array}{rcl} & X & \Rightarrow \quad TZ \\ \dot{X} = A_iX + B_iU & & \\ Y = C_iX & \Rightarrow & \left\{ \begin{array}{l} \dot{Z} = T_i^{-1}A_iT_iZ + T_i^{-1}B_iU \\ Y = C_iT_iX \end{array} \right. \end{array}\right\} \tag{4.2}$$

only holds iff $T_i = T \; {}^{\forall}i \in \left[\begin{array}{cccc} 1 & 2 & \ldots & s \end{array} \right]$. This can be seen when observing the state evolution near a mode switch:

$$\begin{array}{l} X(0) = T_1Z(0) \\ Z(1) = T_2^{-1}A_2T_2Z(0) \\ X(1) = A_2X(0) = A_2T_1Z(0) \neq \; T_2Z(1) = A_2T_2Z(0) \end{array} \tag{4.3}$$

As the example above shows a simple reset map $T_1^{-1}T_2$ would have solved the discrepancy. This can be interpreted to be a transformation from the Z domain to the X domain in mode one and the back-transformation in mode two. The first step to the PWARX++ system is finding the regressor reset for systems without input. Therefore a set of T_i matrices must be found, so that for each mode the regressor can be transformed into a state. For this purpose Eigen Value decomposition is used, because it is known that the eigenvalues of both the PWARX system and the PWA system must be identical (if they are to represent the same system):

$$\begin{array}{rcll} AT_A & = & T_AE & \\ QT_Q & = & T_QE & \\ Q & = & T_QT_A^{-1}AT_AT_Q^{-1} & (4.4) \\ X & = & T_AT_Q^{-1}\varphi & \\ \dot{\varphi} & = & Q\varphi = T_QT_A^{-1}AT_AT_Q^{-1}\varphi & \\ Z & = & \Theta\varphi = CT_AT_Q^{-1}\varphi & (4.5) \end{array}$$

In this equation Q is defined as in equation 3.2 and E is defined as the eigenvalue matrix (a diagonal matrix with eigenvalues). Note that the eigenvalues arrangement in E is arbitrary. This is no issue as long as E is identical in both cases. Also note that T_A and T_Q are not unique, in fact each column of T_A and T_Q may independently be multiplied by an arbitrary value, which leads to $\mathbb{R}^n$

solutions. These degrees of freedom are necessary to solve $CT_A T_Q^{-1} = \Theta$:

$$\begin{aligned} CT_A &= \Theta T_Q = \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix} QT_Q && (4.6) \\ &= \begin{bmatrix} 1 & 0 & \dots & 0 \end{bmatrix} T_Q E && (4.7) \end{aligned}$$

The equation above shows that the uppermost element of each column of T_Q must be equal element of the corresponding column of CT_A divided by the corresponding eigenvalue. This can be done by scaling the entire column of T_Q, which was allowed by the degrees of freedom in the matrix. So the algorithm above presents a transformation matrix that can be used to transform a regressor into a state (and vice versa). Let this matrix be called $T_i = T_{A_i} T_{Q_i}^{-1}$. To find the matrices $\Re_{ij}$ the process above is to be repeated for each mode and a system 1.4 is obtained with $\Re_{ij} = T_j T_i^{-1}$.

The conditions of equations 2.14 and 2.15 apply to PWARX system with regressor reset. The system can be written in the following form:

$$\begin{aligned} \phi(k) &= Q_{\sigma(k)} \Re_{\sigma(k),\sigma(k-1)} \phi(k-1) \\ y(k) &= \Theta \phi(k) \end{aligned} \quad (4.8)$$

The system above is the PWARX++ system as defined in System 1.4, which can be interpreted as a state space system with s^2 modes. Writing out the factor $C_{1,\sigma_1(k)} A_{1,\sigma_1(k)} R_{1,\Sigma_{k-p}^{k-1}}$ gives the following:

$$\Theta_{\sigma(k)} \Re_{\sigma(k),\sigma(k-1)} Q_{\sigma(k-1))} (\Re Q)^{\Sigma_{k-p+1}^{k-1}} \begin{bmatrix} \Theta_{\sigma(k-1)} (\Re Q)^{\Sigma_{k-p+1}^{k-1}} \\ \vdots \\ \Theta_{\sigma(k-p+1)} (\Re Q)_{\sigma(k-p+1)} \\ \Theta_{\sigma(k-p)} \end{bmatrix}^{-1} \quad (4.9)$$

Proving that the formula above is equal to $C_{2,\sigma_2(k)} A_{2.\sigma_2(k)} R_{2,\Sigma_{k-p}^{k-1}}$ is hard to do directly. Therefore the substitution $\varphi(0) = T_{\sigma(0)}^{-1} X(0)$ is used, obtaining:

$$\Theta_{\sigma(k)} \Re_{\sigma(k).\sigma(k-1)} Q_{\sigma(k-1)} (\Re Q)^{\Sigma_{k-p+1}^{k-1}} T_{\sigma(k-p+1)}^{-1} \begin{bmatrix} C_{\sigma(k-1)} A^{\Sigma_{k-p+1}^{k-1}} \\ \vdots \\ C_{\sigma(k-p+1)} A_{\sigma(k-p+1)} \\ C_{\sigma(k-p)} \end{bmatrix}^{-1} \quad (4.10)$$

The equation above, together with $\Re_{\sigma(k),\sigma(k-1)} = T_{\sigma(k)}^{-1} T_{\sigma(k-1)}$ and the substitutions in equation 4.5 get:

$$C_{\sigma(k)} T_{\sigma(k)} T_{\sigma(k)}^{-1} T_{\sigma(k-1)}^{-1} Q_{\sigma(k-1)} (T_i^{-1} T_j Q)^{\Sigma_{k-p+1}^{k-1}} T_{\sigma(k-p+1)}^{-1} \begin{bmatrix} C_{\sigma(k-1)} A^{\Sigma_{k-p+1}^{k-1}} \\ \vdots \\ C_{\sigma(k-p+1)} A_{\sigma(k-p+1)} \\ C_{\sigma(k-p)} \end{bmatrix}^{-1} =$$

$$C_{\sigma(k)}A_{\sigma(k)}A^{\Sigma_{k-p+1}^{k-1}}\begin{bmatrix} C_{\sigma(k-1)}A^{\Sigma_{k-p+1}^{k-1}} \\ \vdots \\ C_{\sigma(k-p+1)}A_{\sigma(k-p+1)} \\ C_{\sigma(k-p)} \end{bmatrix}^{-1} \tag{4.11}$$

The equation above shows that the compatibility requirements for the PWARX++ system without input as defined in this section have been met. The PWARX++ system can be extended, so that input can be accounted for. Again the problem is to find a linear reset map for PWARX++ systems, such that they behave identical to the corresponding state space system, in terms of input/output behavior.

Before looking into this matter, it is important to note that the state has n dimensions whereas the regressor has $(1+p)n$ dimensions. Therefore this problem most likely has a solution space of $2n - n = n$ dimensions.

To solve this problem, a state-to-regressor and a regressor-to-state transformation are sought. These are likely to be linear transformations, however due to the fact that the regressor has $2n$ dimensions, the transformation matrices are not square.

To solve the problem the input regressor and the output regressor are treated separately. Therefore in the state space system the terms of the output regressor must be separated from the input terms:

$$\begin{aligned} X(k) &= A^nX(k-n) + A^{n-1}BU(k-n+1) + \ldots + ABU(k-1) + BU(k) = \\ &= A^nX(k-n) + \begin{bmatrix} B & AB & \ldots & A^{n-1}B \end{bmatrix}\begin{bmatrix} U(k) \\ U(k-1) \\ \vdots \\ U(k-n+1) \end{bmatrix} = \\ &= A^nX(k-n) + M\begin{bmatrix} U(k) \\ U(k-1) \\ \vdots \\ U(k-n+1) \end{bmatrix} \end{aligned} \tag{4.12}$$

To deal with the $A^nX(k-n)$ part of this expression, the solution found above can be used. The second part, concerning the input, can be treated separately. For compatibility with the state space system the following relation must hold at each mode-switch:

$$\begin{aligned} {}_1X &= {}_2X \\ {}_1X = {}_1T_R\ {}_1R'_y + M_1\ {}_1R_x &= {}_2T_R\ {}_2R'_y + M_2\ {}_2R_x = {}_2X \end{aligned} \tag{4.13}$$

Where ${}_iX$ is the state in the corresponding mode, ${}_iT_R$ is the transformation matrix used for transforming the output-part of the regressor to the state space, ${}_iM$ is used for transforming the input-part of the regressor to the state space and is defined as in equation 4.12, R_x is the input regressor and R'_y is the output

regressor. The output regressor in the equation above consists of values that would be outputted if there was no input, so this is not equal to the actual array of outputs. The following expression can be given for this regressor as a function of the real output and input regressor:

$$\begin{aligned} R'_y &= R_y - \begin{bmatrix} \varphi_1 & \varphi_1\theta_1 + \varphi_2 & \varphi_1\theta_1^2 + \varphi_2\theta_1 + \varphi_1\theta_2 + \varphi_3 & \dots \\ 0 & \varphi_1 & \varphi_1\theta_1 + \varphi_2 & \dots \\ 0 & 0 & \varphi_1 & \dots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} R_x \\ &= R_y - \begin{bmatrix} I\varphi_1 & Q\varphi_1 + I\varphi_2 & Q^2\varphi_1 + Q\varphi_2 + I\varphi_3 & \dots \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \end{bmatrix} R_x \\ &= R_y - NR_x \end{aligned} \tag{4.14}$$

As has been said before a solution space of $\mathbb{R}^n$ exists for finding the new regressor given the old regressor and the fact that output behavior of both the PWARX-like system and the PWA system are identical. Therefore we can choose to set the input part of the new regressor to zero, obtaining:

$$\begin{aligned} {}_2T_R\ {}_2R'_y &= {}_1T_R\ {}_1R'_y + M_1\ {}_1R_x \\ {}_2R'_y &= \frac{{}_1T_R\ {}_1R'_y + M_1\ {}_1R_x}{{}_2T_R} \\ {}_2R'_y &= \frac{{}_1T_R\ {}_1({}_1R_y - N_1\ {}_1R_x) + M_1\ {}_1R_x}{{}_2T_R} \\ {}_2R'_y &= \frac{{}_1T_R\ {}_1R_y + (M_1 - {}_1T_R N_1)\ {}_1R_x}{{}_2T_R} \end{aligned} \tag{4.15}$$

Since the input regressor part of the new regressor has been set to zero, ${}_2R'_y = {}_2R_y$. This leads to the following reset matrices:

$$\Re_{ij} = \begin{bmatrix} \frac{{}_iT_R}{{}_jT_R} & \frac{M_i - {}_iT_R N_i}{{}_jT_R} \\ O & O \end{bmatrix} \tag{4.16}$$

The conditions described in equations ?? and ?? also apply to the new PWARX++ system with input, however due to the complexity of the S matrix it is hard to show this.

4.2 PWA++

Two of the problems defined in the previous chapter involved pwa system 1.3. The first question is if it is possible to write a PWARX system into the form of a minimal PWA++ model. The second question is given any state space system and transformation matrices T_i how to calculate the reset map of the

transformed system such that the original and the transformed system behave identical in terms of input and output.

For the current problem the regressor remains the same during a mode switch, and the state must be reset in order to be compatible with the PWARX model:

$$\begin{aligned} R_1 &= R_2 \\ X_1 &= T_1\ R'_{y,1} + M_1\ R_{x,1} \\ X_2 &= T_2\ R'_{y,2} + M_2\ R_{x,2} \end{aligned} \tag{4.17}$$

As is shown in the previous section for PWARX models the input can be set to zero leaving only the $T_i R_{y,i}$ part of the state which can easily be converted. It is then easy to see that $\Re_{i,j} = T_j T_i^- 1$

For the second question this is the same. Just use the transformation matrices given to calculate the $\Re i, j$ state reset map.

4.3 Switching rules and polyhedral regions

Up until now the switching rules for piecewise affine or piecewise ARX systems have not been considered. However this topic does not require much words. The polyhedral regions for PWA systems are usually a set of inequalities, in the form:
mode 1 in the region:

$$\begin{aligned} x_1 &< 10 \\ x_2 + 0.5X_1 &< 3 \\ -x_1 + 0.8x_3 &< 7 \end{aligned} \tag{4.18}$$

which can be written like:

$$Hx < D \tag{4.19}$$

Now for the PWARX++ system as defined above to switch at the exact same moment as the corresponding PWA system, either the state must be calculated or the H matrix must be transformed to regressor format. For the first solution the regressor-to-state transformation matrices defined in previous sections can be used. The second solution is closely related and uses the relation of equation 4.13 to get to:

$$\begin{aligned} x &= TR'_y + MR_x \\ Hx &= HTR'_y + HMR_x < D \end{aligned} \tag{4.20}$$

For the PWA++ systems it is obvious that the transformation matrix that transforms the state between the systems can also be used to adapt the H matrix of the polyhedral regions.

Appendix A

Q_i Commutation

The proof used for Problem 1.2 uses the fact that the Q_i matrices do not commute. In this appendix that fact is proved, using Property 2. The following equations show that Q_1 and Q_2 only commute iff $Q_1 = Q_2$.

The Q matrix of the n^{th} order is the following:

$$Q = \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & \dots & \theta_{n-1} & \theta_n \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix} \tag{A.1}$$

So when multiplying Q_1 with Q_2 the following is obtained:

$$\begin{aligned} Q_1 Q_2 &= \begin{bmatrix} {}_1\theta_1 & {}_1\theta_2 & {}_1\theta_3 & \dots & {}_1\theta_{n-1} & {}_1\theta_n \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix} \begin{bmatrix} {}_2\theta_1 & {}_2\theta_2 & {}_2\theta_3 & \dots & {}_2\theta_{n-1} & {}_2\theta_n \\ 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} \dots & \dots & \dots & \dots & \dots & \dots \\ {}_2\theta_1 & {}_2\theta_2 & {}_2\theta_3 & \dots & {}_2\theta_{n-1} & {}_2\theta_n \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots \end{bmatrix} \end{aligned} \tag{A.2}$$

whereas

$$Q_2Q_1 \quad = \quad \begin{bmatrix} \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ {}_1\theta_1 & {}_1\theta_2 & {}_1\theta_3 & \cdots & {}_1\theta_{n-1} & {}_1\theta_n \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \end{bmatrix} \tag{A.3}$$

So it is obvious that $Q_1Q_2 = Q_2Q_1$ iff ${}_1\theta_1 = {}_2\theta_1$, ${}_1\theta_2 = {}_2\theta_2$, $\ldots$ ${}_1\theta_n = {}_2\theta_n$, so $Q_1Q_2 = Q_2Q_1$ iff $Q_1 = Q_2$. $\triangle$

Appendix B

Q_i Eigenvectors

This appendix describes the derivation of the eigen vectors of a ARX system.

System theory for a state space system in the form of System 1.1 (with $s = 1$ so only one mode) that the equivalent ARX model has the following parameters:

$$\Theta = \begin{bmatrix} \lambda_1 + \lambda_2 + \lambda_3 + \dots \\ \lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_2\lambda_3 + \dots \\ \lambda_1\lambda_2\lambda_3 + \dots \\ \vdots \end{bmatrix} \tag{B.1}$$

where $\begin{bmatrix} \lambda_1 & \lambda_2 & \lambda_3 & \dots \end{bmatrix}$ are the eigenvalues of the Q matrix. So that Q:

$$Q = \begin{bmatrix} \lambda_1 + \lambda_2 + \lambda_3 + \dots & \lambda_1\lambda_2 + \lambda_1\lambda_3 + \lambda_2\lambda_3 + \dots & \lambda_1\lambda_2\lambda_3 + \dots & \dots \\ 1 & 0 & 0 & \dots \\ 0 & 1 & 0 & \dots \\ \vdots & \vdots & \ddots & \ddots \end{bmatrix} \tag{B.2}$$

The eigen vector matrix T has the following definition:

$$\begin{aligned} QT &= TE \\ E &= \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & \lambda_n \end{bmatrix} \end{aligned} \tag{B.3}$$

The matrix T that performs this transformation turns out to be the follow-

ing:

$$T = \begin{bmatrix} \frac{(-\lambda_1)^n}{1+\lambda_1^2+\lambda_1^4+\ldots+\lambda_1^{2n}} & \frac{(-\lambda_2)^n}{1+\lambda_2^2+\lambda_2^4+\ldots+\lambda_2^{2n}} & \cdots & \frac{(-\lambda_n)^n}{1+\lambda_n^2+\lambda_n^4+\ldots+\lambda_n^{2n}} \\ \frac{(-\lambda_1)^{n-1}}{1+\lambda_1^2+\lambda_1^4+\ldots+\lambda_1^{2n}} & \frac{(-\lambda_2)^{n-1}}{1+\lambda_2^2+\lambda_2^4+\ldots+\lambda_2^{2n}} & \cdots & \frac{(-\lambda_n)^{n-1}}{1+\lambda_n^2+\lambda_n^4+\ldots+\lambda_n^{2n}} \\ \vdots & \vdots & & \vdots \\ \frac{1}{1+\lambda_1^2+\lambda_1^4+\ldots+\lambda_1^{2n}} & \frac{1}{1+\lambda_2^2+\lambda_2^4+\ldots+\lambda_2^{2n}} & \cdots & \frac{1}{1+\lambda_n^2+\lambda_n^4+\ldots+\lambda_n^{2n}} \end{bmatrix} \quad \text{(B.4)}$$

Note that the (vector)length of each column is 1. Now when substituting this matrix in equation B.3, the following is obtained:

$$QT = \begin{bmatrix} \frac{(\lambda_1+\lambda_2+\lambda_3+\ldots)\lambda_1^n-(\lambda_1\lambda_2+\lambda_1\lambda_3+\lambda_2\lambda_3+\ldots)\lambda_1^{n-1}+(\lambda_1\lambda_2\lambda_3+\ldots)\lambda_1^{n-2}-\ldots}{1+\lambda_1^2+\lambda_1^4+\ldots+\lambda_1^{2n}} & \cdots \\ \frac{\lambda_1^n}{1+\lambda_1^2+\lambda_1^4+\ldots+\lambda_1^{2n}} & \cdots \\ \frac{\lambda_1^{n-1}}{1+\lambda_1^2+\lambda_1^4+\ldots+\lambda_1^{2n}} & \cdots \\ \vdots & \cdots \\ \frac{\lambda_1}{1+\lambda_1^2+\lambda_1^4+\ldots+\lambda_1^{2n}} & \cdots \end{bmatrix} \quad \text{(B.5)}$$

Due to the size of the equation the other columns have been left out. The other columns are similar to the first one. The factors of the first row can be lumped and the following is obtained:

$$\begin{bmatrix} \frac{\lambda_1^{n+1}}{1+\lambda_1^2+\lambda_1^4+\ldots+\lambda_1^{2n}} & \frac{\lambda_2^{n+1}}{1+\lambda_2^2+\lambda_2^4+\ldots+\lambda_2^{2n}} & \cdots & \frac{\lambda_n^{n+1}}{1+\lambda_n^2+\lambda_n^4+\ldots+\lambda_n^{2n}} \\ \frac{\lambda_1^n}{1+\lambda_1^2+\lambda_1^4+\ldots+\lambda_1^{2n}} & \frac{\lambda_2^n}{1+\lambda_2^2+\lambda_2^4+\ldots+\lambda_2^{2n}} & \cdots & \frac{\lambda_n^n}{1+\lambda_n^2+\lambda_n^4+\ldots+\lambda_n^{2n}} \\ \vdots & \vdots & & \vdots \\ \frac{\lambda_1}{1+\lambda_1^2+\lambda_1^4+\ldots+\lambda_1^{2n}} & \frac{\lambda_2}{1+\lambda_2^2+\lambda_2^4+\ldots+\lambda_2^{2n}} & \cdots & \frac{\lambda_n}{1+\lambda_n^2+\lambda_n^4+\ldots+\lambda_n^{2n}} \end{bmatrix} = TE \quad \text{(B.6)}$$

This equals TE and thus proves that the matrix T can be used to diagonalize Q. It also proves that given a Q_1 and Q_2 only commute iff ${}_1\lambda_1 =_2 \lambda_1;{}_1\lambda_2 =_2 \lambda_2;{}_1\lambda_3 =_2 \lambda_3;\ldots$ and thus $Q_1 = Q_2$. $\triangle$

Bibliography

[1] A "Bemporad and L. Ljung" J. Roll. "identification of hybrid systems via mixed integer programming". In *"Proceedings of the 40th IEEE Conference on Decision and Control"*, pages "786–792", "Orlando, Florida USA", "December" "2001". "IEEE", "IEEE".

[2] G "Ferrari-Trecate and M. Morari" M. Muselli, D. Liberati. "identification of piecewise affine and hybrid systems". In *"Proceedings of the American Control Conference"*, pages "3521–3526", "Arlington, Texas USA", "June" "2001".

[3] W.P.M.H "Heemels and A. Bemporad" B. De Schutter. "on equivalence of classes of hybrid dynamical models". In *"Proceedings of the 40th IEEE Conference on Decision and Control"*, pages "364–369", "Orlando, Florida USA", "December" "2001". "IEEE", "IEEE".

[4] A. Lj. Juloski. *"Observer Design and Identification Methods for Hybrid Systems"*. PhD thesis, "Technische Universiteit Eindhoven", "November" "2004".

[5] J.H.G. "Niessen, G. Ferrari-Trecate A. Lj. Juloski, and W.P.M.H. Heemels". "comparison of three procedures for the identification of hybrid systems". In *"Proceedings IEEE International Conference on Control Applications"*, pages "643–648", "Taipei, Taiwan", "September" "2004". "IEEE", "IEEE".

bcorneli

\\server\name

PSCRIPT Page Separator

Part II

Modeling and predictive control of a hybrid system based on an identified model

B. Vet, M. Lazar and A. Juloski

November 23, 2005

Abstract

In many works on control of hybrid systems it is assumed that an accurate model of the system is readily available. In industrial applications an accurate model of the system is not available and needs to be identified from measurements. Methods and algorithms are available for both identification as control of hybrid systems. The control of hybrid systems is based on the Model Predictive Control (MPC) strategy. In this report is investigated how well the identification procedures work in conjunction with the control algorithms.

The problem is approached by implementing an MPC controller based on an identified model of a physical model of a pick-and-place machine. Also the real pick-and-place machine has been identified, based on measurements on this machine.

A controller for the model of the pick-and-place machine based on an identified model is obtained and has been simulated. The result shows that the approach described in this report can used. However the robustness of the obtained controller is insufficient and more research is required to improve this.

Acknowledgements

During the project I have received encouragement and assistance from my supervisors to whom I would like to express my appreciation and gratitude.

First of all I would like to thank Prof. Dr. Ir. P.P.J. van den Bosch for both his advice on the technical content of the project, but especially for his stimulating encouragements.

I am thankful to my supervisors M.Sc. M. Lazar and Dr. Ir. A. Lj. Juloski for there encouragements and scientific suggestions. Their critical remarks have been of value to my project. With Juloski's advice I have been able to rewrite my findings on the PWA and PWARX models into that a mathematical correct and complete report. Lazar has had duty's elsewhere during a large part of the project, but he still made time to give me useful advice.

Also I would like to thank my fellow graduating students for the fruitful discussions and the good working atmosphere.

Contents

Introduction

Hybrid models are used to describe systems that exhibit both continuous and discrete dynamical behavior. The operation of a hybrid system distinguishes several discrete states (modes) and in each of these modes different laws govern the evolution of the continuous states. An example of a system like this is a car with a gearbox, where the continuous dynamical behavior depends on the discrete state of the gearbox. A hybrid model can also be used to describe a non-linear system by approximating the non-linear dynamical relations as piece-wise linear (affine) relations and thus obtaining a piece-wise linear (affine) model. Hybrid systems can be found in many fields like the automotive example above, but also embedded systems, process control, electrical circuits, mechanical and bio-mechanical, biological and biomedical systems and economics.

In the field of hybrid systems several techniques have been developed for amongst others, identification and control. The problem of identifying a hybrid system consists not only of finding the dynamics of each mode, but also the boundaries between modes and the relation between the different discrete states. Several methods for this purpose have been developed. The control techniques for hybrid systems are mostly based on Model Predictive Control (MPC), which is a technique that is often used in industrial applications. The control action is determined by optimizing the control trajectory over a future prediction horizon based on the model of the system. In other words, the MPC-controller uses the model of the system to predict future outputs (or states) which depend on the control action and than calculates the optimal control action. The control strategy can also be used for hybrid systems. A requisite for this strategy is an accurate quantitative model of the system. In many works on the MPC for hybrid systems such a model is assumed to be readily available.

To the knowledge of the authors no research has been done to investigate how well the identification procedures work in conjunction with the control algorithms. Therefore in this report an MPC controller based on an identified model of a hybrid system is implemented and studied. The subjected hybrid system is an experimental setup that represents an industrial pick-and-place machine. The procedure for the both the identification of the system and the controller design is described and the results of the experiments are given. The theory that is used to obtain these results is summarized.

The first chapter is a brief description of the objectives of the project. Also the requirements and constraints of the result of the project are briefly described. The experimental setup and the Matlab and simulink environments are set out in detail in chapter 2. The physical model of the setup is explained in this chapter as well. Chapter 3 contains the depiction of the entire process of the identification of the models. This includes a description of the various aspects of the identification and a sketching of the identification algorithm. The main content of this chapter are the description of the identification results. The fourth chapter describes the controller design process and all aspects correlating to this subject. The first part of this chapter describes the theory and the tools used. The rest of the chapter contains descriptions of the simulations and results. At the end of the chapter a table summarizes the most important aspects of the simulations and controllers. The last chapter of the report contains a conclusion of the project and gives some recommendations for later research.

Pieces of text have been taken from [7] and [14], with permission of the authors.

Chapter 1

Objectives

The objective of the graduation project is to design and implement a controller for a hybrid test setup based on a model obtained by identification, both in simulation as in real experiments. The test setup is the pick-and-place machine described in the next chapter.For the simulations a physical model of the pick-and-place machine is used. An other model for this hybrid system is identified using an available identification procedure and a controller is designed for this model. The obtained controller is than tested on the original model.

The project consists of two parts, the identification of the system and the control of the system.

The result of the identification procedure is analyzed qualitatively, since no quantitative measures for the quality of the identified model can be defined. The identified model is evaluated by comparing a simulation of the identified model with the same simulation of the original model. The best model obtained by identification is then used as a basis for the controller design.

The result of the second part of the project is evaluated quantitatively; the quality of the obtained controller is assessed on the basis of 4 measurable quantities:

- complexity (number of regions)
- performance
- computational complexity
- mode switches

The number of regions is a measure for the complexity of an explicit [1] controller. The complexity of an 'on-line' controller is measured by the number of optimization variables and mixed-integer linear inequalities. The complexity needs to be constrained so that the controller is easy to implement (for example in an embedded solution). The performance is measured by determining the total cost of the MPC controller for the given reference trajectory. The pick-and-place system is inherently stable and the reason a controller is used is to improve the performance. The computational complexity can be directly measured from processor load when calculating the controller. For 'on-line' controllers the computation is performed between sampling instances. The computational load therefore cannot be larger than the sampling time or the controller cannot keep up. With explicit controllers the computational load is shifted to design time, so that the computational load is not critical, but a controller with lower computational load is more convenient. The last quality measure is the number of mode switches, which is required to be minimal.

[1] the difference between 'on-line' and explicit controllers is explained in the first section of chapter 4

Chapter 2

Experimental Setup

During the project experiments and simulations have been done in the following environments:

- Test Setup
- Matlab
- Simulink

2.1 Setup

The experimental setup used in this project is depicted in figure 2.1. This setup represents an industrial pick-and-place machine. A scheme of the setup is presented in figure 2.2. The setup consists of the mounting head, from an actual pick-and-place machine, which is fixed above the impacting surface (the small disc in the figure 2.1). The impacting surface is in contact with the ground via the spring (the spring c_2 in figure 2.2, within the outer tube in figure 2.1). The mechanical construction under the impacting surface is such that only the movement on the vertical axis is enabled (inner tube, which can slide inside the outer tube in figure 2.1). This construction exhibits linear and dry friction phenomena, represented in figure 2.2 by the damper d_2 and the block f_2, respectively. The chosen design of the impacting surface simulates the elasticity properties of the PCB as well as hard mechanical constraints due to saturations.

The mounting head contains: a vacuum pipette which can move on the vertical axis (the mass M in figure 2.2) and which is connected via the spring to the casing (the spring c_1 in figure 2.2); an electrical motor which enables the movement[1] (represented by force F in figure 2.2); and magnetic height sensor, which measures the position of the pipette, relative to the upper retracted position. The position axis is pointed downward, i.e. the value of the position increases when the pipette moves downward. The motion of the pipette is subject to friction phenomena (the damper d_1 and the dry friction block f_1 in figure 2.2). The control input is the voltage applied to the motor, which is converted up to a negligible time constant to the force F.

The dynamics of the experimental setup exhibits, in a first approximation, four different modes of operation:

upper saturation: the pipette is in the upper retracted position (i.e. can not move upward, due to the physical constraints, x=0);

free mode: the pipette is not in contact with the impacting surface, but is not in the upper saturation;

impact mode: the pipette is in contact with the impacting surface, but is not in lower saturation;

lower saturation: the pipette is in the lower extended position, (i.e. can not move downward due to the physical constraints).

[1] The motor is powered through a voltage controlled current amplifier

Figure 2.1: Photo of the experimental setup

In this project only the free, impact and upper saturation modes are taken into account. The lower saturation is not taken into account and not excited during experiments.

In order to perform control actions and to take measurements the system is connected to a PC by a D-Space interface. This is a professional data acquisition (DAQ) interface which enables online measurements and control. The D-Space system contains a digital signal processor (DSP). This processor simulates a controller and calculates the control output values based on inputs at each sample time. The processor is interfaced with the PC, so that measurements can be accessed from the PC and parameters in the processor can online be changed from the pc. The measurement setup with the D-Space system is depicted in figure 2.3.

The D-space system comes with software and matlab and simulink toolboxes. The DSP can easily be programmed with help of these tools. The program "Control Desk" is used as the user interface program to the D-Space system.

2.2 Matlab

For the purpose of this project three Matlab toolboxes have been available. The first toolbox contains the Baysian identification algorithm. This algorithm is described in the chapter "Identification". The Hybrid Toolbox (HT) and the Multi Parametric Toolbox (MPT) implement the MPC control techniques for hybrid systems. These two toolboxes are described in the chapter "Control".

The identification algorithm is executed on a remote computer. This computer is fast[2] and has the correct linear program solvers installed. The operating system of this computer is Linux and the version of Matlab installed on this computer is 6.5 (r13).

The other toolboxes and simulations are run on a local computer that runs Windows XP and Matlab 7.0.

From the block scheme of the pick-and-place machine setup in the previous section a mathematical model is extracted based on the physical relations of the dynamics of the head. For this model the mode switches are assumed to take place at the sampling instances and not in a sampling interval. The extracted model is a discrete-time piece-wise affine model (PWA) and is written in the HYSDEL language. The HYSDEL code can be compiled into structures or classes which can be used by the HT or MPT toolboxes. The HYSDEL code of the model can be found in appendix A

The HYSDEL model has three modes: free mode, impact mode and lower saturation mode. The

[2]Pentium 4, 2 GHz and 512 Mb memory

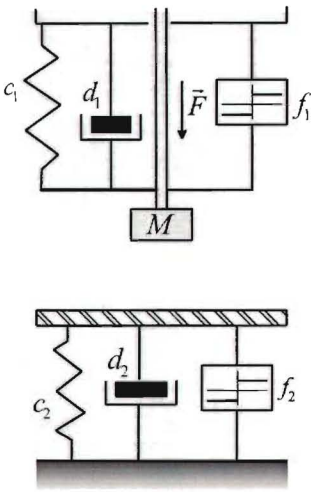

Figure 2.2: Schematic representation of the experimental setup

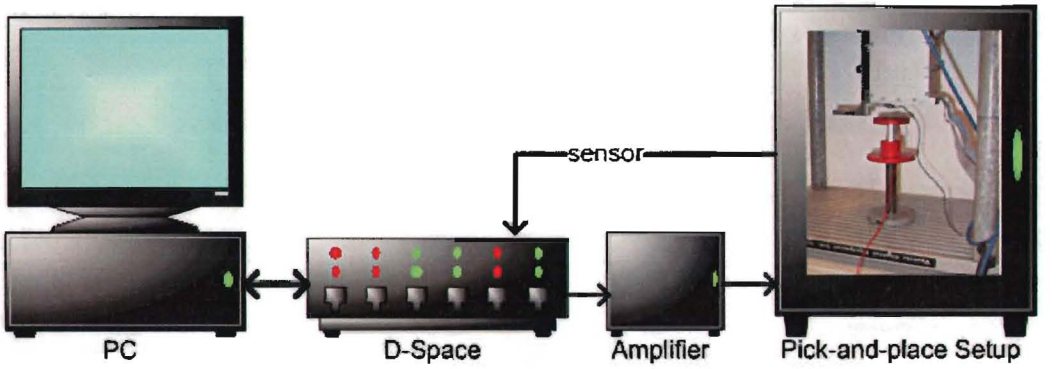

Figure 2.3: Real setup data set with blocked wave input

position where the spring of the head is at rest is considered zero and from there in the direction of lowering the head is the positive y axis in meters. The free mode is considered a damped spring mass system, with parameters spring constant $C_1 = 2\ N/m$, friction constant $W_1 = 0.2\ N/(m/s)$, spring rest position $y_1 = 0$ m, mass of the head $m_1 = 0.05$ Kg and sampling time $T = 0.02$ s. The Impact mode adds a similar spring system to the equations, with with parameters spring constant $C_2 = 6\ N/m$, friction constant $W_2 = 0.8\ N/(m/s)$, spring rest position $y_2 = 0.015$ and mass of the platform $m_2 = 0.1$ Kg. The upper saturation mode is modeled as a spring mass system, with large values for the spring constant ($50\ N/m$) and mass(0.5 Kg). The rest position of this spring is set on the saturation height (0.024 m).

The equations below show the model in PWA form:

$$\left.\begin{array}{l} x(k+1) = \begin{pmatrix} 1 & 0.02 \\ -0.80 & 0.92 \end{pmatrix} x(k) + \begin{pmatrix} 0 \\ 0.40 \end{pmatrix} u(k) \\ y(k) = \begin{pmatrix} 1 & 0 \end{pmatrix} x(k) \end{array}\right\} x_1(k) \in \begin{bmatrix} 0 & 0.015 \end{bmatrix} \quad (2.1)$$

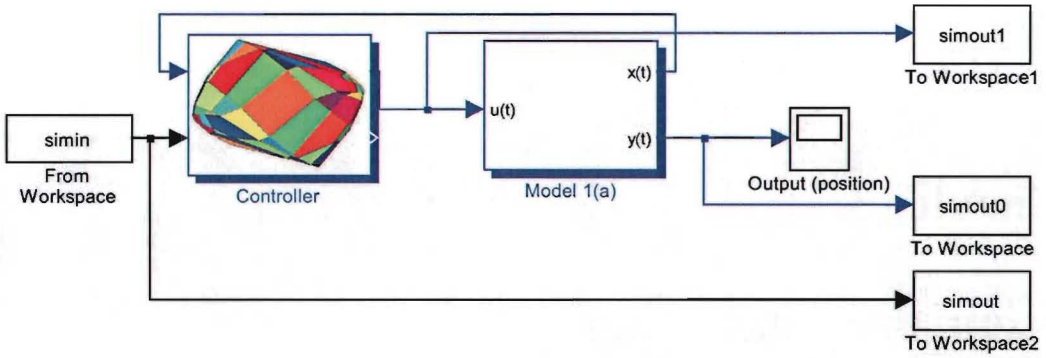

Figure 2.4: Simulink diagram

$$\left.\begin{aligned} x(k+1) &= \begin{pmatrix} 1 & 0.02 \\ -1.0667 & 0.8667 \end{pmatrix} x(k) \\ &+ \begin{pmatrix} 0 \\ 0.1333 \end{pmatrix} u(k) + \begin{pmatrix} 0 \\ 0.012 \end{pmatrix} \\ y(k) &= \begin{pmatrix} 1 & 0 \end{pmatrix} x(k) \end{aligned}\right\} x_1(k) \in \begin{bmatrix} 0.015 & 0.024 \end{bmatrix} \tag{2.2}$$

$$\left.\begin{aligned} x(k+1) &= \begin{pmatrix} 1 & 0.02 \\ -2 & 1 \end{pmatrix} x(k) \\ &+ \begin{pmatrix} 0 \\ 0.04 \end{pmatrix} u(k) + \begin{pmatrix} 0 \\ 0.048 \end{pmatrix} \\ y(k) &= \begin{pmatrix} 1 & 0 \end{pmatrix} x(k) \end{aligned}\right\} x_1(k) \in \begin{bmatrix} 0.024 & 0.030 \end{bmatrix} \tag{2.3}$$

2.3 Simulink

Both the control toolboxes provide functional simulink blocks, which can be used to test a controller in the simulink environment. One of the blocks contains the controller, another block can contain the PWA model. The in- and output signals in simulink can be exported to Matlab for further analyses.

In the simulink environment the controller block and the plant block are better separated than in the Matlab environment, which allows a controller to be designed based on another (the identified) model than the model of the plant itself.

Figure 2.4 shows the blocked scheme used in simulink. The controller block takes the state of the plant and the reference as input an outputs the input for the plant. The in- and output of the plant and the reference are exported to the workspace of Matlab.

Chapter 3

Identification

The goal of the identification procedure is to map a data set of 'measured', noisy input/output (and state) values to a model of the system. The data set must be sufficiently large to counter the effects of the noise and to get a good representation of the original system.

The retrieved model can later be validated by the same or preferably another dataset and a simulation of the retrieved model. A quantitative measure for the quality of the identified model is the relative difference in the parameters between the identified and the real parameters, when these are available.

3.1 Method

3.1.1 Models

The model used for the identification procedure must be able to describe the relevant characteristics and match the behavior of the system at hand. In the case of hybrid systems a broad range of model types is available, examples are Piece-Wise Auto-Regressive-Exogenous Models (PWARX), Piece-Wise Affine Models (PWA) and Mixed Logical Dynamical Models (MLD). Most identification procedures use the PWARX model. The PWARX model is defined as follows:

$$\begin{aligned} y(k) &= \Theta_{\sigma(k)}\varphi(k) + e(k) \\ \varphi(k) &= \begin{bmatrix} y(k-1) \\ \vdots \\ y(k-n) \\ u(k) \\ \vdots \\ u(k-n) \\ 1 \end{bmatrix} \end{aligned} \tag{3.1}$$

where $\Theta_{\sigma(k)} \in \mathbb{R}^{2n+1}$, $k \in \mathbb{N} \cup 0$, $\varphi(k) \in \mathbb{R}^{2n+1}$ is the regressor, s is the number of modes, n is the order of the system, $\sigma(k) : \varphi(k) \mapsto \{1, \ldots, s\}$ maps polyhedral regions of φ to a mode and $\varphi(0)$ is given. $y(k) \in \mathbb{R}$ is the output of the system. $e(k)$ is a white noise term with probability density function $p_e(\cdot)$ that reflects model mismatch and unmodeled dynamical behavior of the system.

The regressor space is divided into bounded strictly separated polyhedral regions:

$$\mathcal{X}_i = \{x \mid H_i x \leq h_i\} \tag{3.2}$$

where H_i is a real valued matrix, h_i is a real valued vector and the inequality holds element-wise. The set $\mathcal{X} = \bigcup_{i=1}^{s} \mathcal{X}_i$ is assumed to be a bounded polyhedral set as well. The dynamics in each of these regressor sets are assumed to be different, so $\Theta_i \neq \Theta_j \mid i \neq j$

The polyhedral regions are used by the function $\sigma(k)$:

$$\sigma(k) \quad = \quad \{i \mid H_i x(k) \leq h_i\} \tag{3.3}$$

The task of mapping the plant into the form of the PWARX model defined above, consists of finding the parameters $Theta_i$, H_i and h_i for each mode.

For proper identification of the system the order of the model n must be equal to or larger than the order of the relevant dynamics of the system. The high frequency resonances are mostly neglected and the bandwidth, in which the ARX model is valid, is limited.

3.1.2 Algorithms

In literature the following algorithms can be found.

- Clustering Method [6]
- Greedy Method [5]
- Algebraic Method [16]
- Bayesian Method [7]

In [8] these methods have been compared. This article concludes that the clustering method has the highest robustness against noise and has other advantages above the others. The Bayesian method is the only method that has the ability to utilize a priori information.

In this project the Bayesian method is used. The source code for the Bayesian method procedure (in Matlab scripts) has been made available for this project.

The Bayesian procedure [9] is based on the idea of refining the available *a* priori knowledge about the modes and parameters of the hybrid system. Parameters θ_i of the piece-wise ARX model are treated as random variables, and described with their probability density functions (pdfs) $p_{\theta_i}(\cdot)$. A priori knowledge on the parameters can be supplied to the procedure by choosing appropriate a priori parameter pdfs. In this paper the procedure is initialized by mode knowledge. This means that for each mode a set of data points is assigned manually, of which is certain that they belong to that mode. From these data sets the initial a priori parameter pdfs are calculated.

The data classification problem is posed as the problem of finding the data classification with the highest probability. Since this problem is combinatorial, an iterative suboptimal algorithm is derived in [9], based on sequential processing of data points in the collected data set. The probability density function of the additive noise term e, $p_e(\cdot)$ is assumed to be given.

The parameter estimation algorithm has N iterations, and in each iteration the pdf of one of the parameters is refined. In the k-th iteration of the algorithm the most probable mode $\mu(k)$ of the data pair $(x(k), y(k))$ is computed, using the available pdfs of the parameter vectors from step $k-1$. Subsequently, the data pair $(x(k), y(k))$ is assigned to the mode i that most likely generated it, and the a posteriori pdf of parameter vector θ_i is computed, using as a fact that the pair $(x(k), y(k))$ was generated by mode i. To numerically implement the Bayesian procedure, particle filtering algorithms are used (see e.g. [1]). In order to have a good representation of the pdf a large number of particles may be needed. This accounts for the majority of the computational burden.

After the parameter estimation phase, data points are attributed to the mode that most likely generated them. For the estimation of regions a modification of the standard MRLP procedure is proposed in [9]. Assume that the data point attributed to the mode i ends up in the region $\mathcal{X}_j$. If the probabilities that the data point is generated by both modes are approximately equal, this misclassification should not be penalized highly. Therefore non-negative valued *pricing functions* are introduced. These functions assign price to misclassification of data points. Pricing functions are used in the MRLP procedure.

The Bayesian procedure requires model orders n_a and n_b, and the number of modes s. The most important tuning parameters of the procedure are the a priori parameter pdfs $p_{\theta_i}(\cdot, 0)$, and the pdf of the additive noise p_e. Also, the particle filtering algorithm has several tuning parameters.

The algorithms used are available in Matlab code. The code requires a mixed integer linear program solver. In this case the gnu solver glpk has been used. The algorithms have been executed on a remote computer at the mathematics faculty and took in the order of 5 minutes per case to complete.

3.1.3 Excitation

The input to the identification algorithms described above are the data sets. A data set consists of chronological equidistant sampled input and output values. To obtain a data set, the system is excited with an input. The input is chosen such that the system is sufficiently excited and that its dynamical behavior can be observed well.

In standard industrial linear system identification methods the system would be excited with white (or colored) noise with a bandwidth large enough to capture the system dynamics of interest. It is than possible to obtain a spectrum of both the input and the output and to correlate these two into a bode plot of the system. However for identification of ARX models it is not necessary to excite the system with an entire range of frequencies, because only a limited order of the system is considered. For proper identification the input has to excite all modes and each mode sufficiently so the dynamical behavior of each mode can be captured.

No guidelines on how to excite a hybrid system for identification purposes are available. Therefore two excitations have been tried and the results are compared.

The first is a blocked signal. The blocks have a fixed length of 15 samples and random amplitude, between 0.01 and 0.08. These values have been established by trial-and-error, so that all modes are excited.

The second excitation signal is a composite of a strong low frequency sine and white noise that has been filtered by a first order moving average filter with length 10. The low frequency sine is used move the head to different regions so that all modes are excited. The filtered noise is used to properly excite each mode.

The simulated model has been excited with both these signals, so that the data sets of figure 3.1 and 3.2 are obtained. For the real setup an excitation of the second type is used. The upper picture of each of the figures contains the set of output values and the lower figure shows the input data set. The two data sets for the model have a length of 750 samples, the length of the measured data set is 600 samples. The marked data points have been used for the initialization of the corresponding mode, which is necessary to initialize the identification procedure.

3.1.4 Expectation

The model that is to be identified is a PWA model. Of each mode of this model an ARX description can be given, so that a PWARX model is obtained. The result of the identification is expected to match this PWARX description. The order of the PWA model is 2 so that can be expected that for the identified model na and nb of 2 is sufficient to described the system. The expected parameters of the PWARX model are:

$$\begin{aligned} \Theta_1 &= \begin{bmatrix} 1.9200 & -0.9360 & 0.0080 & 0.0000 & 0.000000 \end{bmatrix} \\ \Theta_2 &= \begin{bmatrix} 1.8667 & -0.8880 & 0.0027 & 0.0000 & 0.000243 \end{bmatrix} \\ \Theta_3 &= \begin{bmatrix} 2.0000 & -1.0400 & 0.0008 & 0.0000 & 0.000960 \end{bmatrix} \end{aligned} \tag{3.4}$$

The parameters of the identified models can be compared to the parameters above, obtaining a quantitative measure for the validity of the identified model.

For the measured system no exact model is available. Therefore no good estimation of the parameters to be identified can be made.

3.2 Simulation 1

Since the simulation system as described in the environment chapter has order two it is assumed that the model that is to be identified has order 2 as well. This means that when n_a and n_b are both set to 2 the system can be identified properly. After some trial and error the parameters for the identification algorithm have been tuned and the identification algorithm finishes successfully.

Since the data sets for the simulations are noiseless, a quite good estimate of the parameters of the PWARX model is found. The identification of the first data set result in the following parameters for the

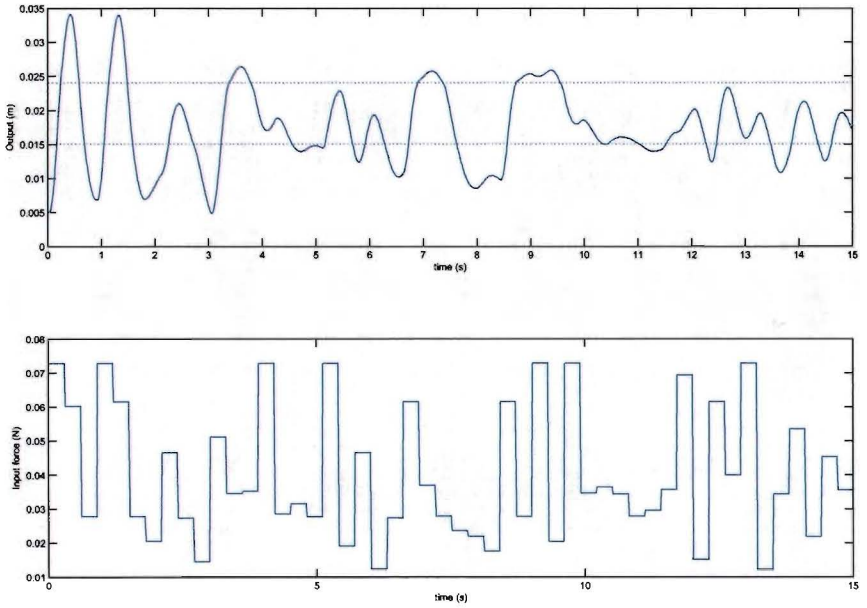

Figure 3.1: Simulated system data set with blocked wave input

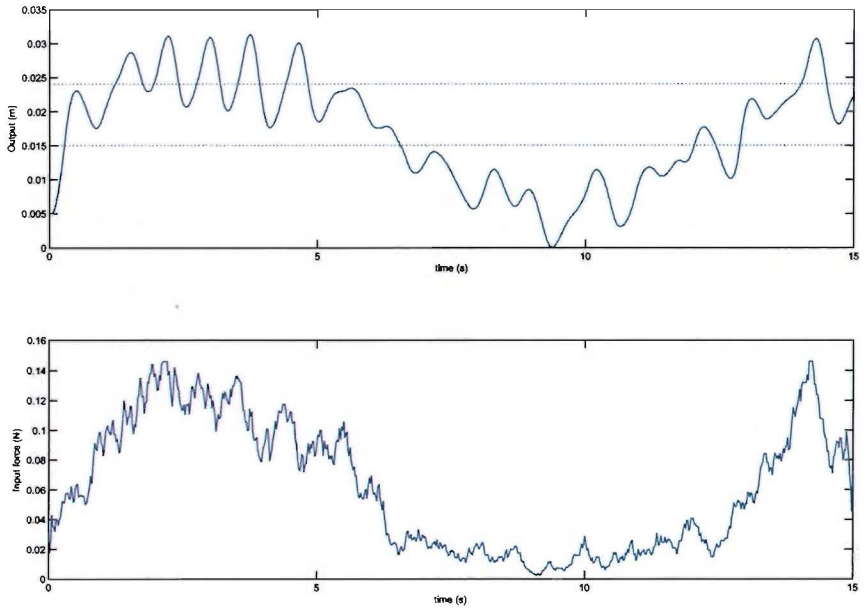

Figure 3.2: Simulated system data set with noisy input

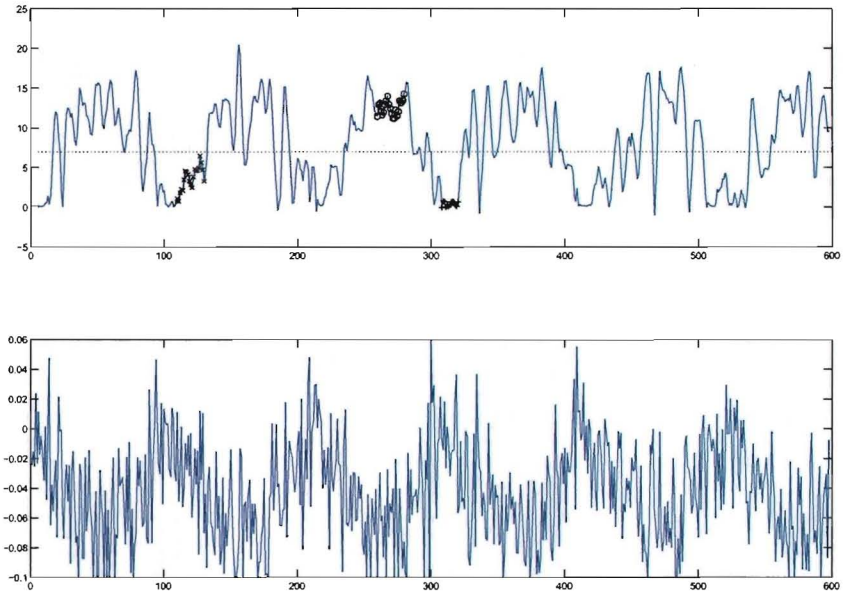

Figure 3.3: Real setup data set with blocked wave input

PWARX model:

$$\begin{aligned} \Theta_1 &= \begin{bmatrix} 1.9234 & -0.9399 & 0.0078 & 0.0000 & 0.0000 \end{bmatrix} \\ \Theta_2 &= \begin{bmatrix} 1.8663 & -0.8879 & 0.0025 & 0.0001 & 0.0003 \end{bmatrix} \\ \Theta_3 &= \begin{bmatrix} 1.9857 & -1.0251 & 0.0008 & 0.0001 & 0.0009 \end{bmatrix} \end{aligned} \tag{3.5}$$

The Bayesian algorithm converged to the above parameters. These maximum deviation of the first two parameters above with respect to the calculated parameters is 1.43%. Figure 3.4b shows that the variance of the probability density functions[1] of the respective modes converged to zero. Due to fact that the data set is noiseless the convergence is quick and jumpy. The classification of figure 3.4a is made by assigning the highest likely mode to each data point. This figure shows that 1-4 samples directly after a mode-switch are misclassified. This is caused by the incompatibility of PWA and PWARX models near mode switches, the relation between PWA and PWARX models is extensively discussed in

To support this conclusion another plot is made. The regressors corresponding to each data point classified to the same mode are grouped into sets of five ($= n_a + n_b + 1$). For each such a group it is possible to calculate the parameters Θ. Two intersections of the 5 dimensional result space are plotted in figure 3.5 and 3.6, respectively the two output parameters and the two input parameters.

Although the parameters are spread over a large area, most of the parameters lie exactly on the same spot. These spots are marked in the figures. The majority of the parameter points of mode 1 (255 out of 262) lie exactly (with a precision up to 10^-10) on the point $\begin{pmatrix} 1.920 & -0.9360 \end{pmatrix}$.

Not for each of the groups of datapoints the parameters could be calculated, some of the groups consisted of mutually dependant regressor groups. This is due to the blocked wave form of the input of the dataset, which causes the corresponding sets of regressors to be dependant. The identification procedure can be affected negatively due to this effect, however to validate this further study is required.

[1] in the figure the spectral radius of the variance matrices is given

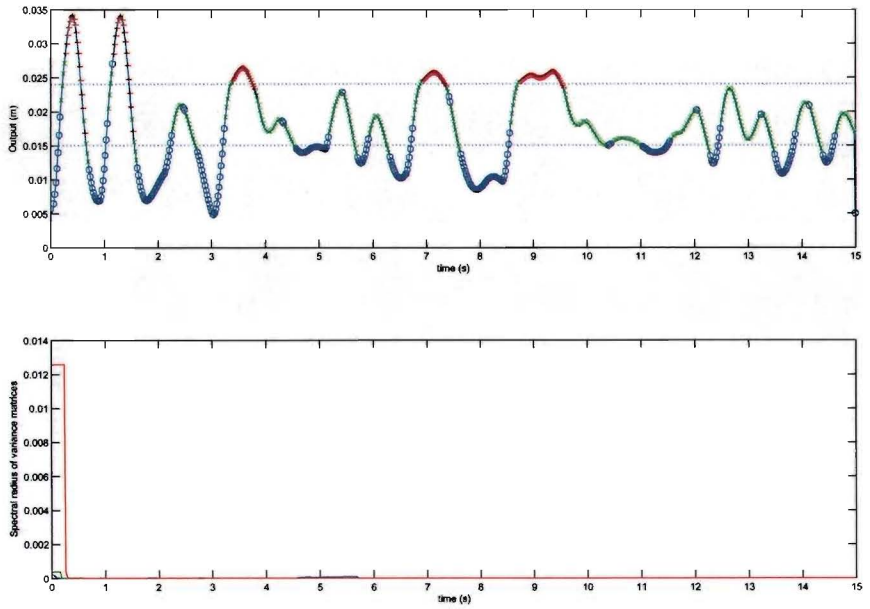

Figure 3.4: Final classification and convergence of variances

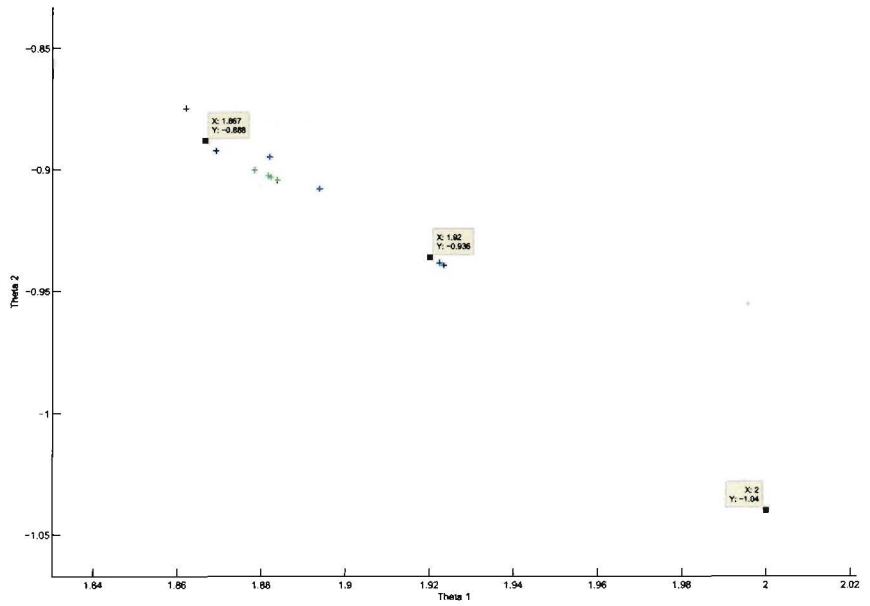

Figure 3.5: Local Parameter Spread (output)

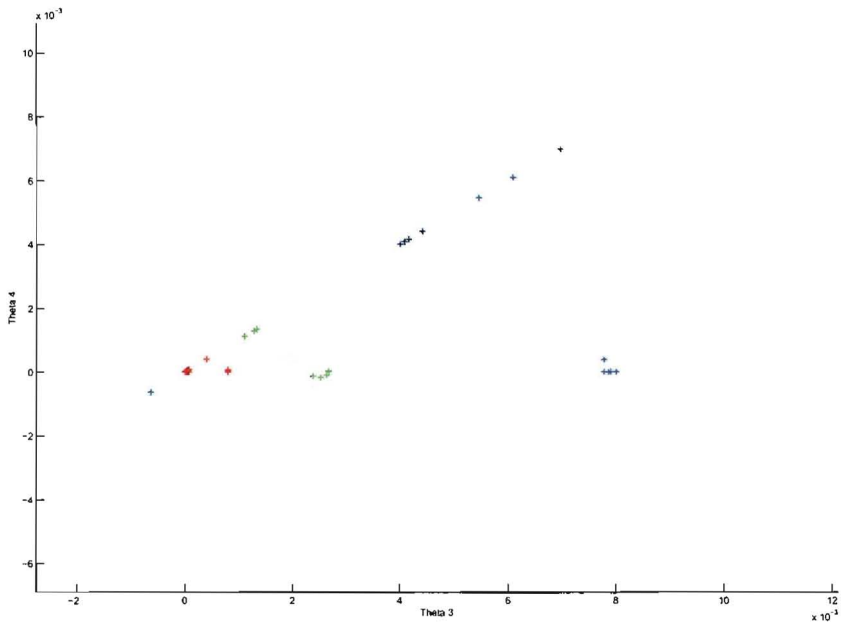

Figure 3.6: Local Parameter Spread (input)

The parameters of the PWARX model alone do not define the system. The complete PWARX model consists of both the parameters above and the guard-lines that separate the modes. However this is where the identification algorithm has problems. The region estimation procedure based on the MRLP algorithm of Bennet and Mangasarian (1994) produces complicated inaccurate guard-lines. This is caused by the fact that 1-4 samples directly after a mode switch are misclassified.

After the identification procedure, the identified model has been simulated, with the input of the original data set. The results are plotted in figure 3.7. The output simulated by the identified model does not resemble the original data set. So the identified model is not an accurate description of the system.

When the guard lines of the model are corrected manually and the simulation is run again, figure 3.8 is obtained. This gives a satisfying result. Thus the parameters are estimated correctly, but the guard-lines are not.

The simulations show that the system at hand is very sensitive near mode-switches especially when entering mode 3. This is due to the fact that dynamical characteristics of the modes (and especially mode 3) are quite different. So that when due to a small deviation in an estimated parameter a mode-switch occurs 1 sample earlier or later the output deviates largely and several instances are required for the output to converge to the correct value. Therefore errors near mode switches between mode 1 and 2 generate only small output deviations and are quickly recovered, but a mode switch between mode 2 and 3 cause a big output error and take more than 10 samples to recover.

The identification algorithm is partially successful in identifying the model based on a data set with a blocky input. The fact that a blocky input is chosen, probably results in dependant regressors, which probably degrades the quality of the identification. The classification of the data points is good, but 1-4 samples directly after a mode switch are misclassified. This causes the estimation of the regions to fail.

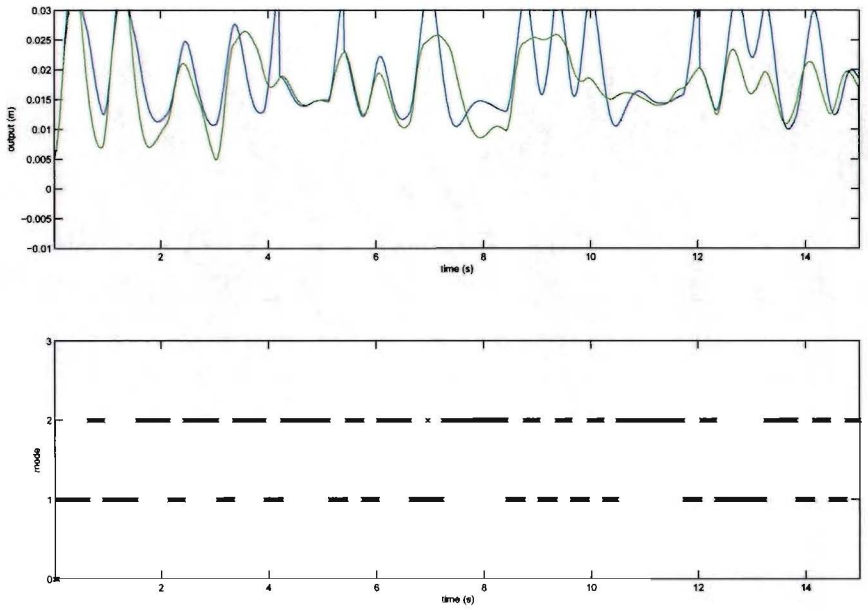

Figure 3.7: Simulation with the identified guard lines

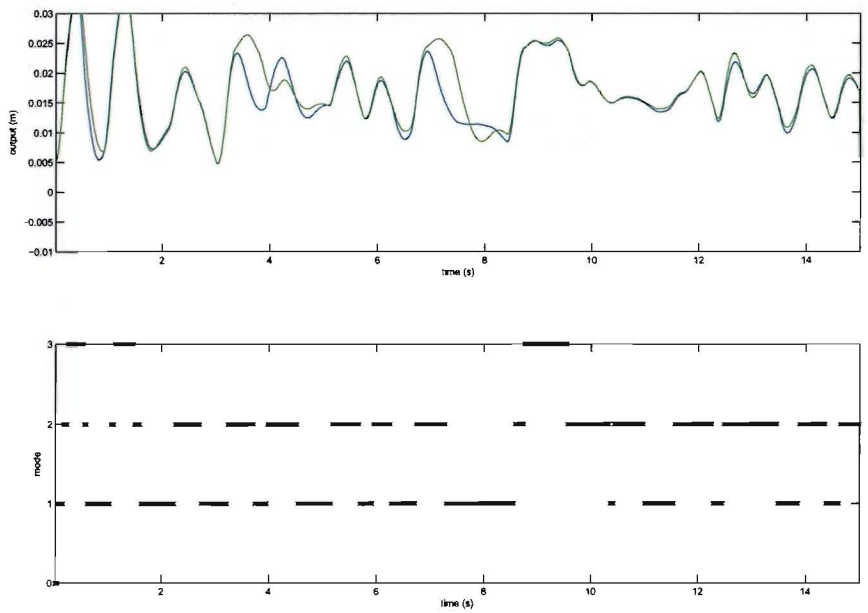

Figure 3.8: Simulation with corrected guard lines

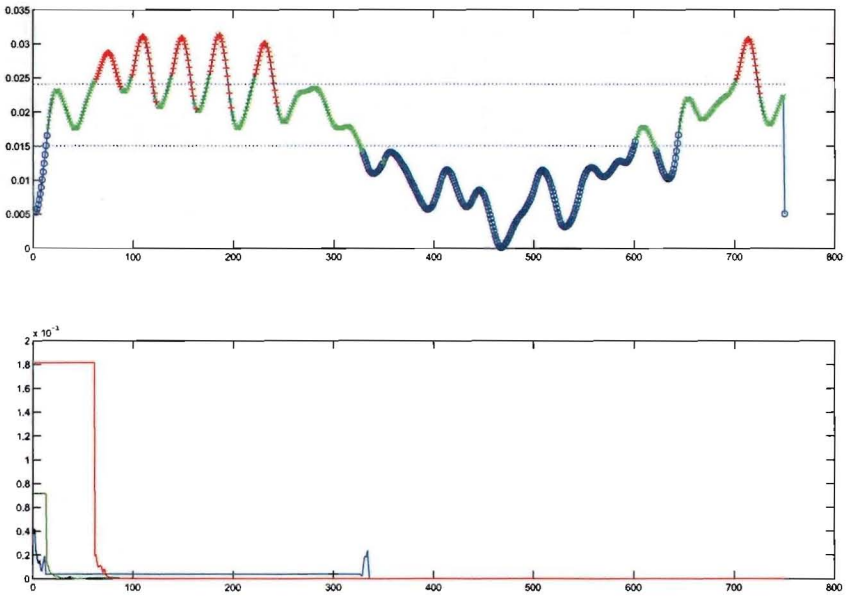

Figure 3.9: Final classification and convergence of variances

3.3 Simulation 2

When the identification procedure is applied to the second data set, the following model is obtained:

$$\begin{aligned} \Theta_1 &= \begin{bmatrix} 1.9109 & -0.9269 & 0.0083 & -0.0003 & 0.0000 \end{bmatrix} \\ \Theta_2 &= \begin{bmatrix} 1.8774 & -0.8973 & 0.0033 & -0.0008 & 0.0002 \end{bmatrix} \\ \Theta_3 &= \begin{bmatrix} 1.9973 & -1.0391 & 0.0004 & 0.0005 & 0.0010 \end{bmatrix} \end{aligned} \tag{3.6}$$

These maximum deviation of the first two parameters above with respect to the calculated parameters is 1.05%. The variance of the probability density functions converges in a fast and jumpy way due to the noisless data set, see figure 3.9b. The classification of the data points is good, see figure 3.9a.

When examining the parameter spread of figures 3.10 and 3.11, a significant difference between the identification of this data set and that of the previous data set can be noticed. The parameter spread figure has no data points in zero, as was expected due to the noisy character of the input of the data set. In this case almost all data points lie exactly on one point. Again a total of about 40 data points lie spread.

The simulation, plotted in figure 3.12, fails to match the actual system. However when the guard lines are corrected the simulation is good, as can be seen in figure 3.13.

3.4 Measurement

The process of identifying the real setup has come with a few difficulties. Identifying the plant took longer than expected. This is attributed to the fact that the data sets acquired by measurement where not good for identification. In these measurements the motion of the head has been hampered in impact mode by dry friction effects. This causes highly non-linear behavior, so that the system cannot be modeled as a hybrid affine system.

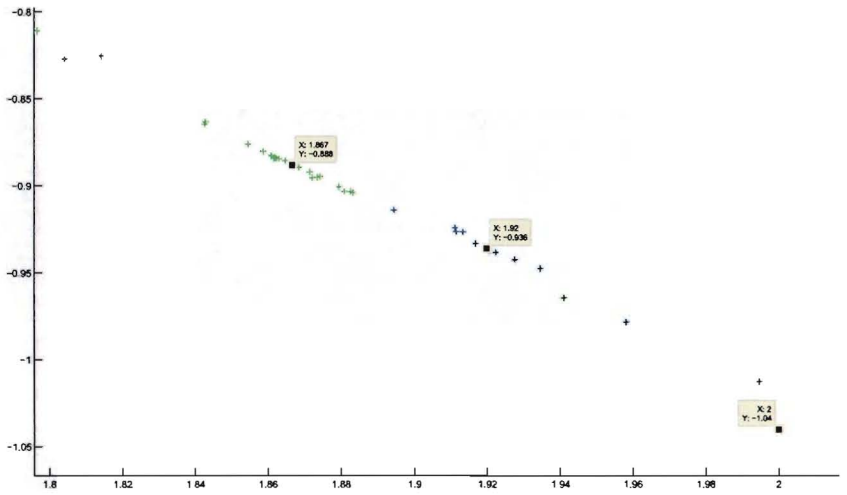

Figure 3.10: Local Parameter Spread (output)

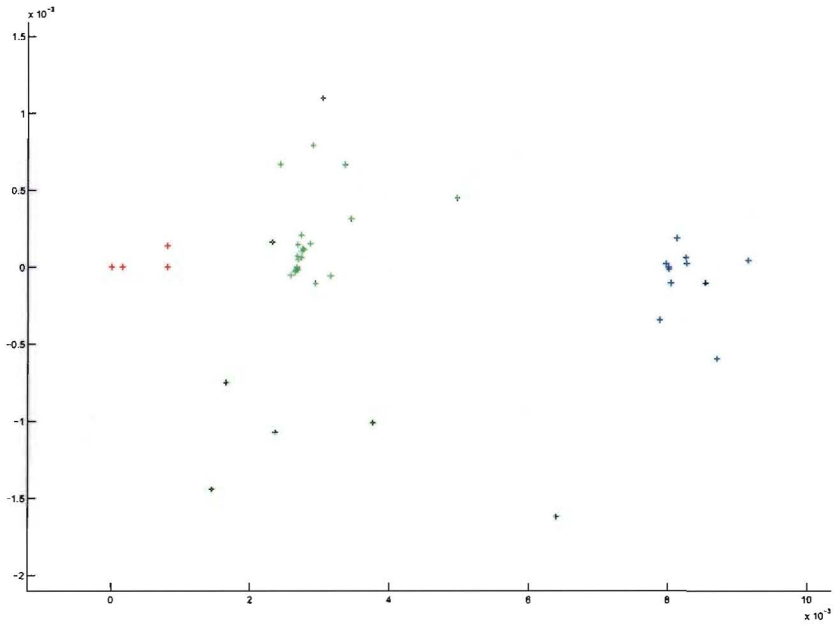

Figure 3.11: Local Parameter Spread (input)

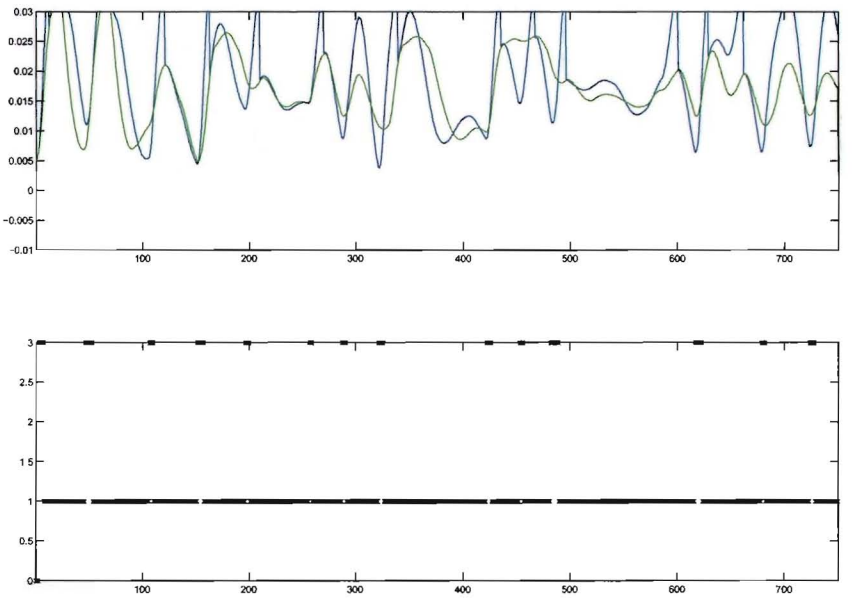

Figure 3.12: Simulation with the identified guard lines

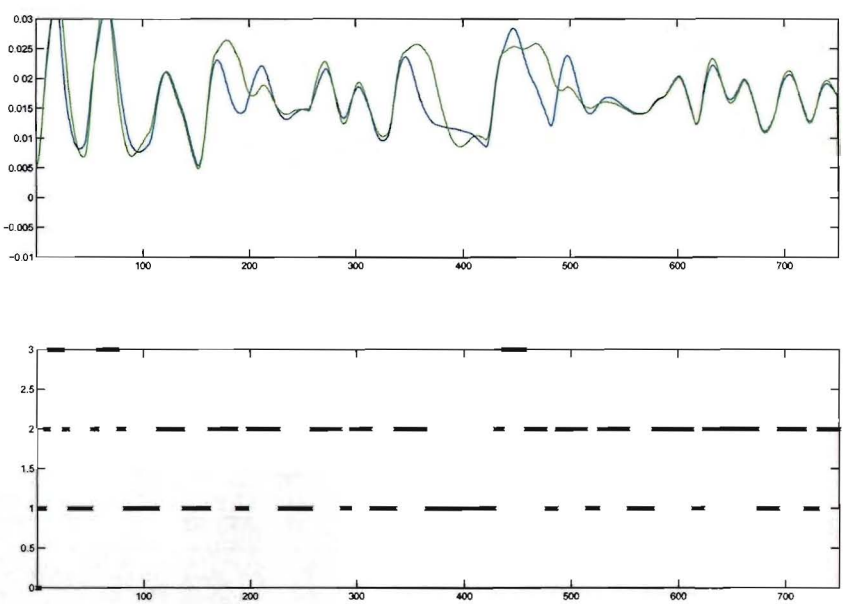

Figure 3.13: Simulation with corrected guard lines

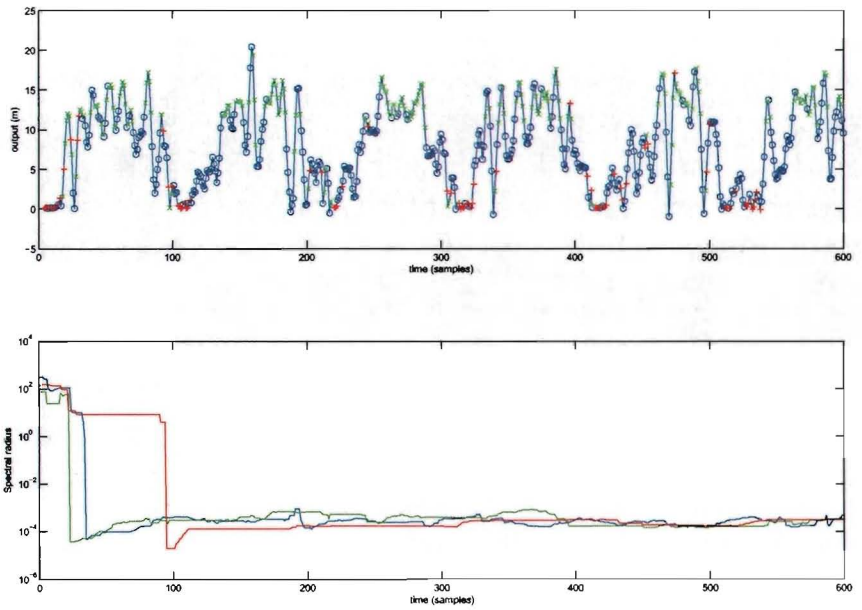

Figure 3.14: Final classification and convergence of variances

The success of the identification procedure is highly dependent on the (quality of the) data set. However as mentioned earlier no clear guidelines exist on how to properly excite a hybrid system. It may be profitable to investigate this matter further.

After altering the impact platform suspension the dry friction effects have subsided. After this alteration a data set has been obtained that gives a sufficient identification result. However this data set has only been found after a few iterations of trial and error. The identification algorithm is very sensitive to the excitation. The algorithm doesn't produce good results for many data sets although no direct cause for this can be found. A more thorough investigation of the algorithm needs to be done in order to find out a procedure for obtaining a data set that results in a good identification.

Although the physical model of the system is of order two, the model order parameters na and nb for output and input order respectively have both been set to 3. This allows third order dynamics that have not been modeled to be identified, so that a more accurate model is obtained.

The identified model has the following parameters [2]

$$\begin{aligned} \Theta_1 &= \begin{bmatrix} 0.8219 & 0.2958 & -0.4190 & -16.4225 & -49.7210 & -18.7187 & -1.2363 \end{bmatrix} \\ \Theta_2 &= \begin{bmatrix} 0.9075 & -0.2210 & -0.1389 & -12.7985 & -32.0476 & -6.4898 & 2.5952 \end{bmatrix} \\ \Theta_3 &= \begin{bmatrix} 2.2016 & -1.4648 & 0.0283 & 7.4167 & -9.8515 & 30.6359 & 0.4879 \end{bmatrix} \end{aligned} \tag{3.7}$$

The figure 3.14 shows the final classification of the data points. The classification is more noisy and irregular than with the simulated data sets, but due to the noisy nature of the measurement this has been expected. The variance converges quickly.

The identified model is used to reconstruct the original data set by one-step-ahead prediction. The data points of the original data set are used to calculate the next output. This procedure resulted in the plot of figure 3.15. The original data is reconstructed very well, the figure lies almost exactly over the original

[2] the guard lines are omitted, they are complex and due to the 7 dimensions can not easily be plotted

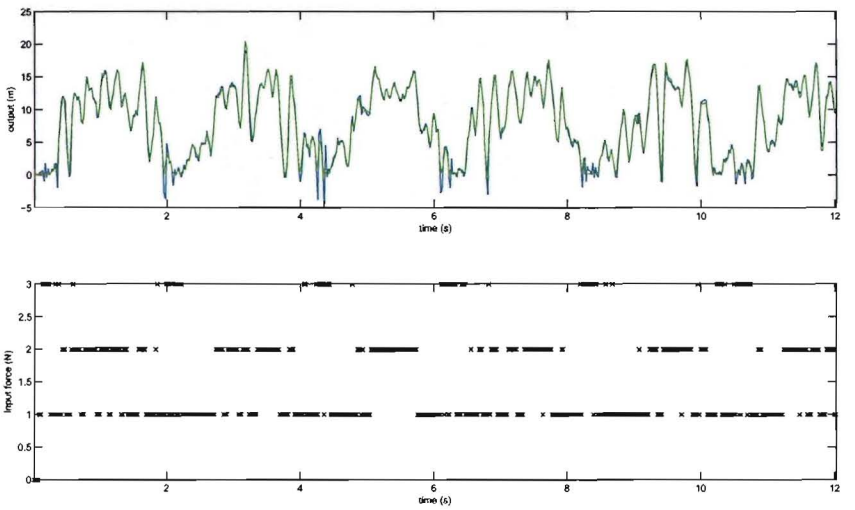

Figure 3.15: One-step-ahead prediction simulation

data set. Only in the lower saturation mode the reconstructed data points are a bit scattered. The dynamics of this mode are not reconstructed as good as the other modes.

The free reconstruction, that simulates the identified model using the input of the original data set, is a better test for the quality of the identified model. This procedure is plotted in figure 3.16. This figure shows that the model tracks the original output good, again with the exception of data points of the lower saturation mode. After the fluctuations caused by this model mismatch the simulated output converges quickly to the output of the original data set. This can only be attributed to the stability of the system in combination with the sufficient quality of the identified model, not to the quality alone. However the figure shows that the dynamics of especially the free and impact mode have been captured sufficiently to use the model as a starting point for the design of the controller.

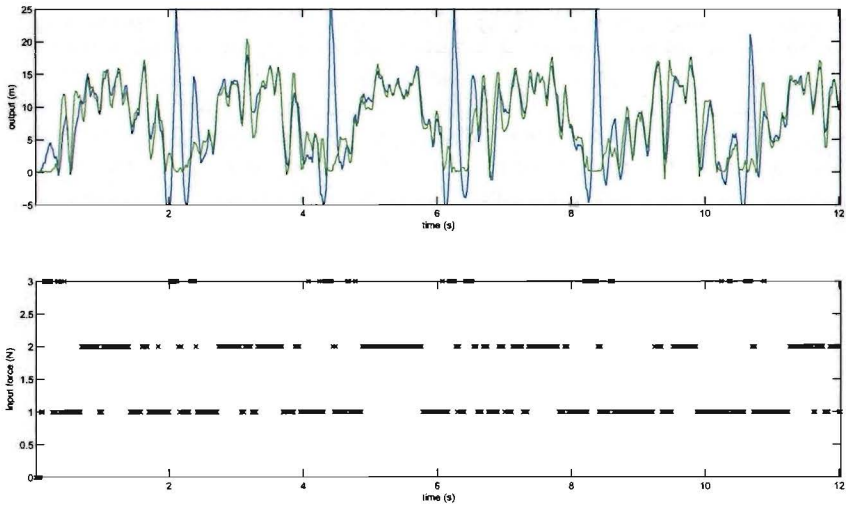

Figure 3.16: Simulation of the identified model

Chapter 4

Control

4.1 Setup

Figure 4.1 shows a simplified diagram of the setup used. The controller calculates input for the plant based on the measurements of states and outputs. Noise and disturbances are not shown in this diagram. x_k is the measured state of the process and u_{k+1} is current control action. $x^*_{k+1}, ..., x^*_{N+k}$ are the predictions generated by the model given the current state x_k and the sequence of inputs $\mathbf{u}_k$. These concepts are explained in the next chapter. The controller is designed to optimize some cost function for this plant. However, as the diagram emphasizes, the controller is based on a model of the plant, not on the plant itself. The model is obtained by identification, as has been done in the first part of the graduation project.

In the next chapter some basic notions and definitions on model predictive control are recalled. Some text is taken from [12], [13] and [14] with permission of the authors.

4.1.1 Model predictive control

Model predictive control is a control strategy that is based on two main principals:

- The use of a model of the proces to predict future proces states
- Calculation of the optimal control action by minimizing a cost function, which (possibly) include constraints on the proces variables

Below a brief description of the control strategy is given

The controller uses the following time-invariant discrete-time model of the process, which is assumed to match the process dynamics:

$$x_{k+1} = g(x_k, u_k) \tag{4.1}$$

where $x_k \in \mathbb{X} \subseteq \mathbb{R}^n$ is the state, $u_k \in \mathbb{U} \subseteq \mathbb{R}^m$ is the control input at the discrete-time instant $k \geq 0$ and $g : \mathbb{R}^n \times \mathbb{R}^m \rightarrow \mathbb{R}^n$ is an arbitrary, *possibly discontinuous*, non-linear function. The sets $\mathbb{X}$ and $\mathbb{U}$ specify state and input constraints and it is assumed that they are closed polyhedral sets.

For a fixed $N \in \mathbb{N}$, $N \geq 1$, $\mathbf{x}_k(x_k, \mathbf{u}_k) := (x_{k+1}, \ldots, x_{k+N})$ is a state sequence generated by system (4.1) from initial state x_k and by applying the input sequence $\mathbf{u}_k := (u_k, \ldots, u_{k+N-1}) \in \mathbb{U}^N$. $\mathcal{X}_T \subseteq \mathbb{X}$ is a desired target set.

Not all the input sequences are admissible, because the state of the system needs to remain within the allowed state space.

Definition 4.1 The class of *admissible input sequences* defined with respect to $\mathcal{X}_T$ and state $x_k \in \mathbb{X}$ is $\mathcal{U}_N(x_k) := \{\mathbf{u}_k \in \mathbb{U}^N \mid \mathbf{x}_k(x_k, \mathbf{u}_k) \in \mathbb{X}^N, x_{k+N} \in \mathcal{X}_T\}$.

The control problem is finding the input sequence within the admissible input sequence set that minimizes a cost function.

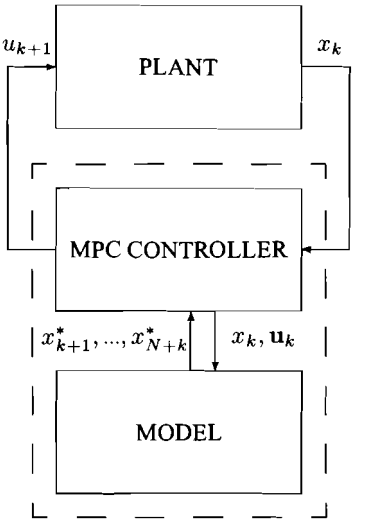

Figure 4.1: Control setup diagram

Problem 4.2 Let the target set $\mathcal{X}_T \subseteq \mathbb{X}$ and $N \geq 1$ be given and let $F : \mathbb{R}^n \to \mathbb{R}_+$ and $L : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}_+$ be mappings. At time $k \geq 0$ let $x_k \in \mathbb{X}$ be given and minimize the cost function

$$J(x_k, \mathbf{u}_k) \triangleq F(x_{k+N}) + \sum_{i=0}^{N-1} L(x_{k+i}, u_{k+i}) \tag{4.2}$$

over all input sequences $\mathbf{u}_k \in \mathcal{U}_N(x_k)$.

The parameters F, L and N are the terminal cost, the stage cost and the prediction horizon, respectively. The initial state $x \in \mathbb{X}$ is called *feasible* if $\mathcal{U}_N(x) \neq \emptyset$. Similarly, Problem 4.2 is said to be feasible (or *solvable*) for $x \in \mathbb{X}$ if $\mathcal{U}_N(x) \neq \emptyset$. The feasible set, $\mathcal{X}_f(N)$, is the collection of all feasible initial states.

For simplicity is assumed that an optimal sequence of control input exist,

$$\mathbf{u}^*_k \triangleq (u^*_k, u^*_{k+1}, \ldots, u^*_{k+N-1}) \tag{4.3}$$

calculated for state $x_k \in \mathcal{X}_f(N)$ and that minimizes the cost function of Problem 4.2. The problem 4.2 is solved according to a receding horizon strategy, so that at each sampling instance the optimal sequence of control inputs is calculated. The control action at each sample instance is

$$u^{\text{MPC}}_k = \mathbf{u}^*_k(1); \quad k \in \mathbb{N}. \tag{4.4}$$

In this report time-invariant discrete-time Piecewise Affine (PWA) systems [15] are used:

$$x_{k+1} = A_j x_k + B_j u_k + f_j \quad \text{when} \quad x_k \in \Omega_j, \tag{4.5}$$

Here, $x_k \in \mathbb{X} \subseteq \mathbb{R}^n$ is the state and $u_k \in \mathbb{U} \subseteq \mathbb{R}^m$ is the control input at the discrete-time instant $k \geq 0$. $A_j \in \mathbb{R}^{n \times n}$, $B_j \in \mathbb{R}^{n \times m}$, $f_j \in \mathbb{R}^n$, $j \in \mathcal{S}$ with $\mathcal{S} := \{1, 2, \ldots, s\}$ a *finite set* of indices and s denoting the number of discrete modes. Here, $f_j \in \mathbb{R}^n$ denotes a fixed offset vector for all $j \in \mathcal{S}$. The collection $\{\Omega_j \mid j \in \mathcal{S}\}$ defines a partition of $\mathbb{X}$, meaning that $\cup_{j \in \mathcal{S}} \Omega_j = \mathbb{X}$ and $\Omega_i \cap \Omega_j = \emptyset$ for $i \neq j$. Each Ω_j is assumed to be a polyhedron (not necessarily closed). Let $\mathcal{S}_0 := \{j \in \mathcal{S} \mid 0 \in \text{cl}(\Omega_j)\}$ and let $\mathcal{S}_1 := \{j \in \mathcal{S} \mid 0 \notin \text{cl}(\Omega_j)\}$, so that $\mathcal{S} = \mathcal{S}_0 \cup \mathcal{S}_1$.

In this project the ∞-norm is used to define the cost function, i.e. $F(x) = \|P_j x\|_\infty$ when $x \in \mathcal{X}_T \cap \Omega_j$ and $L(x, u) = \|Qx\|_\infty + \|Ru\|_\infty$. Here $P_j \in \mathbb{R}^{p \times n}$, $Q \in \mathbb{R}^{q \times n}$ and $R \in \mathbb{R}^{r \times n}$ are assumed to be matrices that have full-column rank. The MPC cost function (4.2) equals:

$$J(x_k, \mathbf{u}_k) \triangleq \|P_j x_{k+N}\|_\infty + \sum_{i=0}^{N-1} \|Q x_{k+i}\|_\infty + \|R u_{k+i}\|_\infty \quad \text{when} \quad x_{k+N} \in \Omega_j, \quad j \in \mathcal{S}. \tag{4.6}$$

4.1.2 Explicit vs 'on-line' controllers

The problem 4.2 is solved at each time instance. For linear cost [1] the problem can be formulated as a linear program. Solving a linear program is a complex task that results in a large computational load. This limits the applicability of MPC; The computational load increases vastly with incrementing prediction horizon and for certain horizons the computation time becomes larger than the sampling time. At this point the controller cannot be implemented.

However a technique has been developed that solves this problem by moving the necessary calculations to design time. In [4] is shown that the optimal control profile for discrete-time linear time-invariant systems is a piecewise linear and continuous function of the initial state. The on-line calculation of the control action can be reduced to the evaluation of this PWL function. The controller found this way is called the explicit controller (the former controller is called 'on-line'). With the explicit controller the on-line computational load is almost eliminated. However the finding the PWL function that determines this explicit controller is complex task with high computational load, but this function must only be calculated once during design time.

4.1.3 Tools

Two toolboxes are available that provide functionality to design MPC controllers for hybrid systems, the Hybrid toolbox designed by A. Bemporad [2], and the MPT toolbox designed by Michal Kvasnica, Pascal Grieder, and Mato Baotic [10]. The MPT toolbox has an adequate graphical user interface and gives clear error messages in case of unexpected input or behavior. The Hybrid toolbox is faster than the MPT toolbox, but does not always give a clear error message in case of unexpected behavior. The MPT toolbox is unable to deal with 2-norm hybrid controllers. The Hybrid toolbox is unable to cope with 1-norm hybrid controllers. This limits the choice between the toolboxes, however in this report only infinity-norm controllers are used, which are supported by both toolboxes. Both toolbox are under active development.

Another restriction in using these toolboxes is the fact that they both rely on LP (linear program), QP (quadratic program), MILP (mixed integer linear program) and MIQP (mixed integer quadratic program) solvers. These solvers are mostly commercial products. There are some freeware and General Public Licensed solver packages, but these are limited in speed and numerical stability. The solvers used for this project are the LP and QP solvers that come with the Optimization toolbox of Matlab.

4.1.4 Control Parameters

The control functions from both Toolboxes require parameters that determine the sort of controller to be generated. In order to compare the results of the different toolboxes and different simulations, the parameters are chosen the same for each simulation, except for some simulations that study the effect the parameter has on the final result. These parameters are the penalty weights for states, inputs and outputs, the prediction horizon N.

The parameters are chosen so that they match the requirements set for the controller in the project objectives. The performance requirement in the requirements sections is only related to the output performance. Therefore the penalty on the output is set to 10 and this is the parameter in all simulations and measurements. There are no requirements for the input and states, so the penalties are set to 0. The prediction horizon used is set to 3, which is a trade-off between lower complexity and better performance. However, the prediction horizon has been varied in several simulations, to analyse the effect and to improve the result.

[1] 1- or ∞ norm costs

4.2 Matlab simulations with the hybrid toolbox

In this chapter several simulations are discussed. The simulations are carried out on the Matlab models as plant, with controller's that are directly based on the corresponding model. For these simulations the hybrid toolbox is used. The last chapter of this chapter contains table that summarizes the results of all sections in this chapter.

4.2.1 'on-line' controller with prediction horizon 3

In the first simulation the plant subjected to the controller is the Matlab model and the controller is directly derived from this model. The controller is generated with the Hybrid toolbox and set to track the previously mentioned output reference trajectory. The controller is an 'on-line' infinity-norm controller, with the standard parameters as described above. The option tracking is engaged, which allows the reference to be an external vector. The tracking parameter is constrained to be in the interval of 0 to 0.025.

Figure 4.2 shows the input, output and reference time series of the simulated system. The simulation stops at 2.4 seconds because the reference at that point in time cannot be reached with a prediction horizon of 3 inputs within the input constraints. The simulation cannot be recovered. The figure shows that the output reference is tracked with a delay of about 0.04 seconds. 0.02 seconds of this delay is caused by the fact that the reference trajectory steps are plotted with diagonal lines, not vertical. Therefore the reference seems to rise earlier than it actually does. The fact that only the second state of the state space system is actuated, while the output is related to the first state causes a delay of one sample (the remaining 0.02 seconds).

The time necessary to reach the state after the reaction shows is never larger than three sampling times, 0.06 seconds. At simulation times 0.8 and 2.2 the output overshoots the reference. This occurs on downward reference steps. However this also means that the control action is jerky. For a sufficiently large reference step the control action clips at the input constraints of the system the first sample after the reference step.

The total cost of this simulation up to the time the simulation stops is 1.11, of which 0.56 is to be contributed to the delay of two sampling instances. The function that generated the controller took $5.01 \cdot 10^{-2}$ seconds to complete. The simulation of the closed-loop took 0.4707 seconds.

The controller's complexity can be indicated by the number of optimization variables and mixed-integer linear inequalities. The controller has 18 (12 continuous and 8 binary) optimization variables and 66 inequalities.

4.2.2 Explicit controller with prediction horizon 3

In this simulation the plant is controlled by the explicit version of the controller above. All parameters are the same and the explicit controller is directly derived from the controller of the previous simulation

The closed-loop simulation is plotted in figure 4.3. When comparing this plot with the previous one, it can be seen that the explicit controller behaves exactly the same as the 'on-line' controller except near simulation time 1.6 s. Around that time a the output shows a bump where in the previous simulation there is none.

The total cost of this simulation is up to the time the simulation stops is 1.18, which is slightly larger than the previous simulation due to the bump near simulation time 1.6 s. The function that generated the controller took 8.98 seconds to complete, this number is exclusive the time required to calculate the 'on-line' controller. The generation time for the explicit controller is about 200 times larger than the generation time of the 'on-line' controller. The fact that this number is so large is partially due to the high complexity of the required operation, but also for a large part due to output and error messages sent to the Matlab console screen. These messages and errors appeared even though all verbosity levels had been set to zero.

Although the generation time of the explicit controller is much larger than that of the 'on-line' controller, the calculation time of the simulation is much lower, $7.01 \cdot 10^{-2}$ seconds, which is about 70 times lower than the calculation time of the 'on-line' controller simulation. This effect has been expected, because this is the main advantage of an explicit controller.

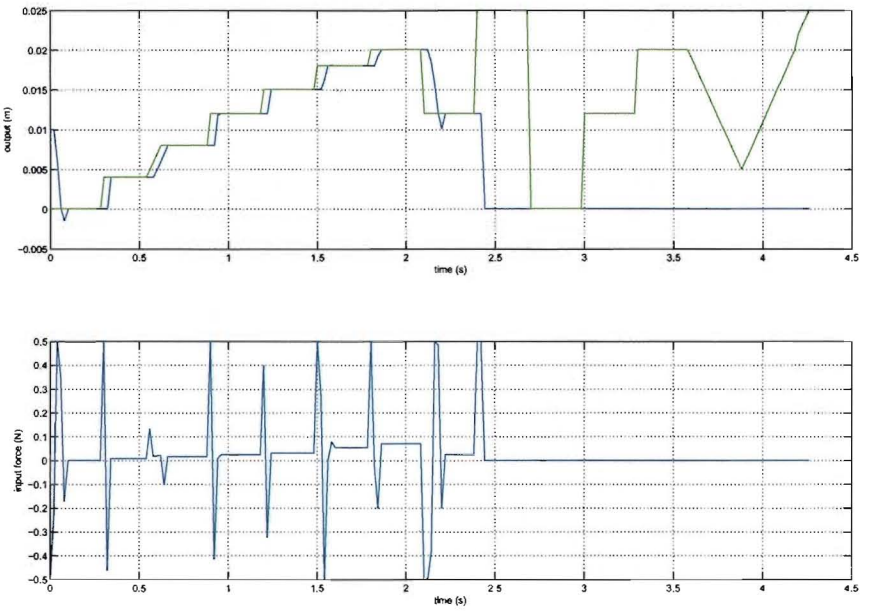

Figure 4.2: Input/output plot of closed-loop simulation (Simulation 1)

The controller is a PWA system that consists of 21 regions, with 3 parameters, 2 states and a reference, since the controller is tracking a output reference. The explicit controller is plotted in figure 4.4. In the figure it can be seen that there exist regions where the reference is not reachable with the bounded inputs within the 3 steps of the control horizon.

4.2.3 'on-line' and explicit controllers with prediction horizon 4 and 5

Since the previous simulations could not follow the reference trajectory, the control horizon has been increased so that the reference trajectory as it has been defined in the first part of this chapter can be tracked entirely. The simulations as Simulation 1 and 2 have been repeated for the control horizons $N = 4$ and $N = 5$. A control horizon equal to six has been tried but the explicit controller could not be generated and therefore no simulations with this control horizon have been done. The generation process of the explicit controller with horizon 6 is larger than an hour. When operating the process, Matlab crashes.

Simulation 3 is the simulation of the 'on-line' controller with control horizon 4. This controller is the first that is able to track the entire reference trajectory. The closed-loop plot of the input and output is depicted in figure 4.5. The most notable difference between this plot and figure 4.2 of the first simulation in the range of the first simulation is that the overshoot at simulation times 0.8 and 2.2 has disappeared in the new plot with control horizon 4. However the maximum time to reach a reference is now 4 sampling instances. The exact output on reference steps are a little more smoother, but these differences are small.

Outside the first simulation's range, near simulation time 2.5, the controller clips on maximum output and it takes 8 sampling instance to reach the reference. In the period directly after this, the reference steps a big step down and it takes more than 11 steps to reach the reference.

The total cost until simulation time 2.4 is 1.085 slightly lower than the cost in the same interval with simulation 1, which is to be expected since a greater control horizon leads to more accurate control. Over the entire simulation the cost is 4.40 of which 2.38 is caused by the delay of two sampling instances. The processor time required for Matlab to compute the controller was 0.34 seconds. The time to calculate the

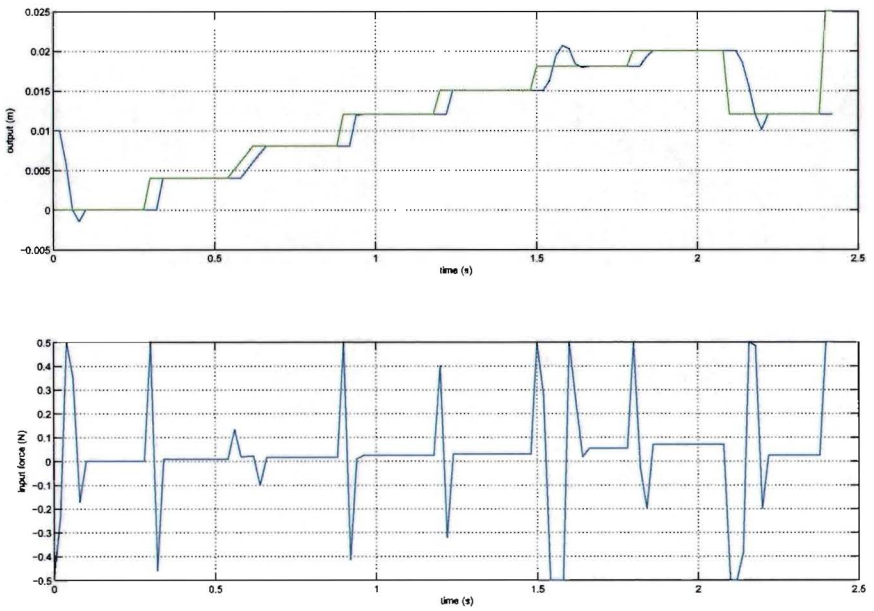

Figure 4.3: Input/output plot of closed-loop simulation with explicit controller (Simulation 2)

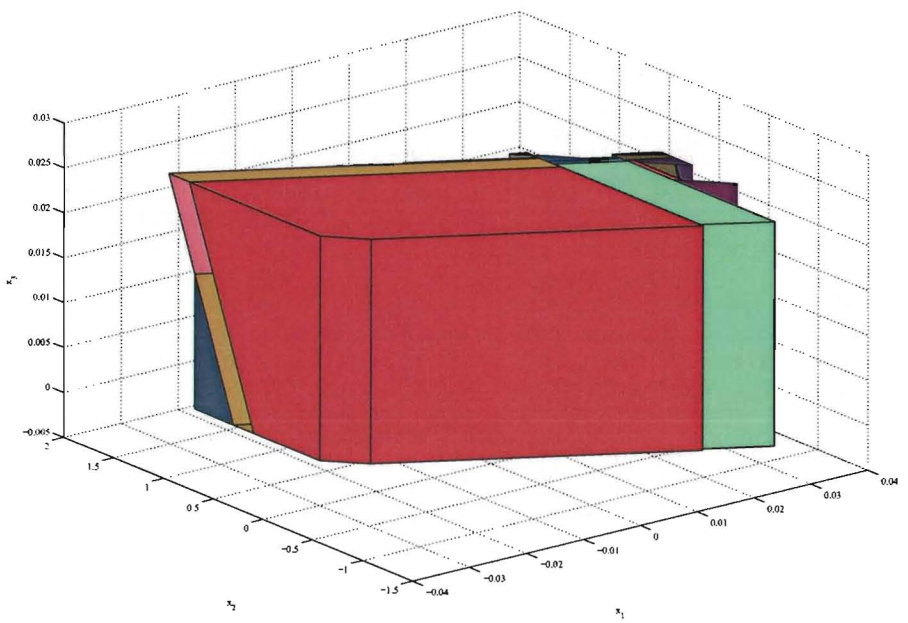

Figure 4.4: 3D plot of the explicit controller of Simulation 2

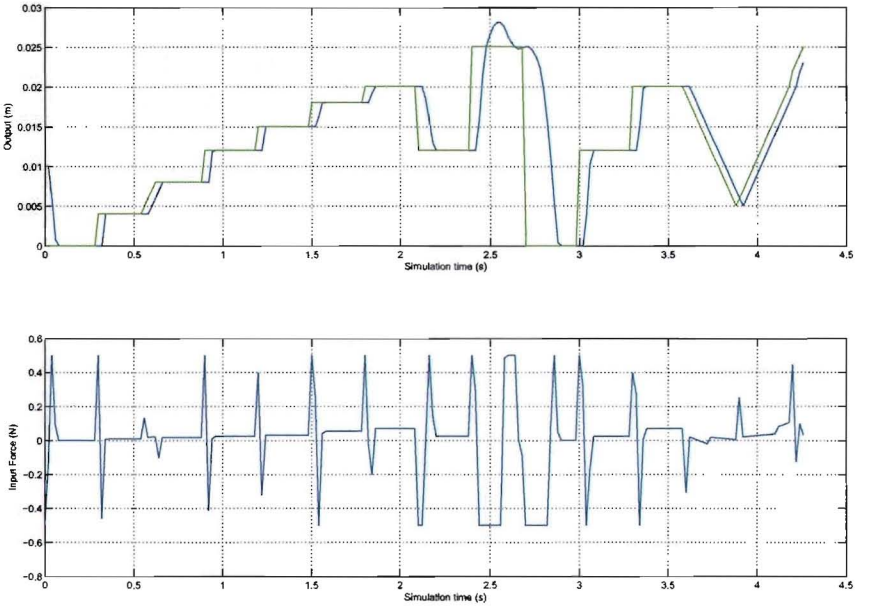

Figure 4.5: Input/output plot of closed-loop simulation (Simulation 3)

simulation was 1.49 seconds. The controller consists of 24 optimization variables (16 continuous and 8 discrete) and 88 mixed-integer inequalities.

The controller of simulation 3 has been converted to the explicit form. The controller obtained consists of 119 partitions and 3 parameters and is plotted in figure 4.7. The conversion took 58.9 seconds. The controller is unable to track the reference after simulation time 2.44, the state and reference parameters at that time lie outside the regions of the controller.

Up to that time the simulation took 0.11 seconds and the cost is 1.095, of which again 0.56 is caused by the delay. The closed-loop in- and output are plotted in figure 4.6. The difference of this plot with respect to previous plots is the overshoot at simulation time 1.54, which is smaller than in simulation 2.

The fifth simulation uses an 'on-line' controller with control horizon $N = 5$. Figure 4.8 shows the closed-loop plot. Only one difference with simulation 3 is visible. Near simulation time 3.3 s this simulation is a little smoother. The calculation time of the controller was 0.08 seconds. The processor time required for the simulation was 2.11 seconds. The cost until simulation time 2.4 is 1.085 and the total cost is 4.448. The controller has 30 optimization variables, 20 continuous and 10 binary. 110 mixed-integer linear inequalities constrain the controller.

Generation of an explicit version of the above controller failed with the error message that the facet problem is infeasible. No documentation on this problem has been found, but relaxing the input constraints solved the problem. Therefore it is likely that the explicit generation process fails due to too strict constraints. The input constraints are relaxed to ± 6.0. Another solution has been found later, lowering the weight on the output error. This is hard too explain, because the output is the only weight. Somehow the absolute value of this weight influences the calculations.

Figure 4.9 shows the closed-loop plot, however the results cannot be compared because the input constraints are relaxed. The big bump near simulation time 1.6 is still notable. The complexity of the controller however is of interest. The controller took 619.12 seconds to compile, the simulation took 0.11 seconds. The cost is 1.034, but not comparable. The controller consists of 171 regions and three parameters. Figure 4.10 shows some intersections parameter space.

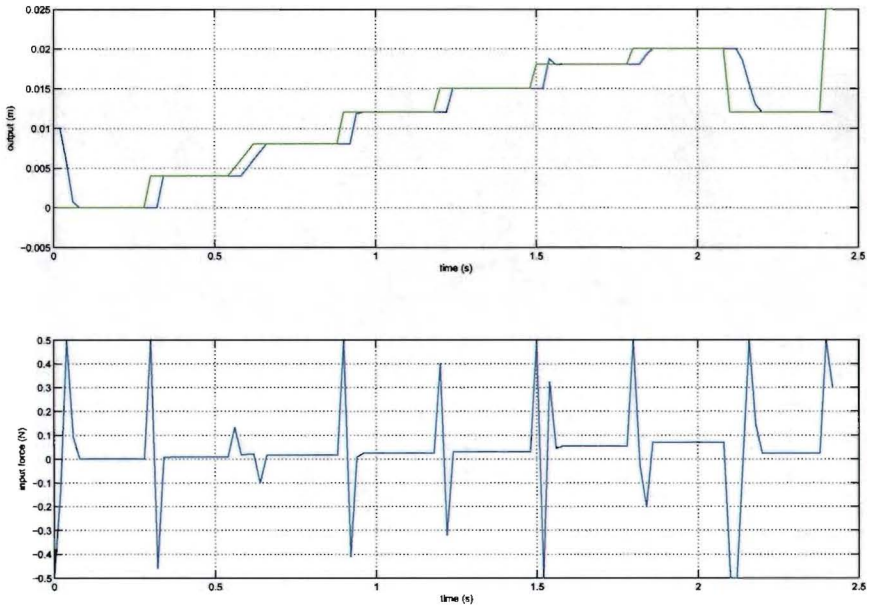

Figure 4.6: Input/output plot of closed-loop simulation with explicit controller (Simulation 4)

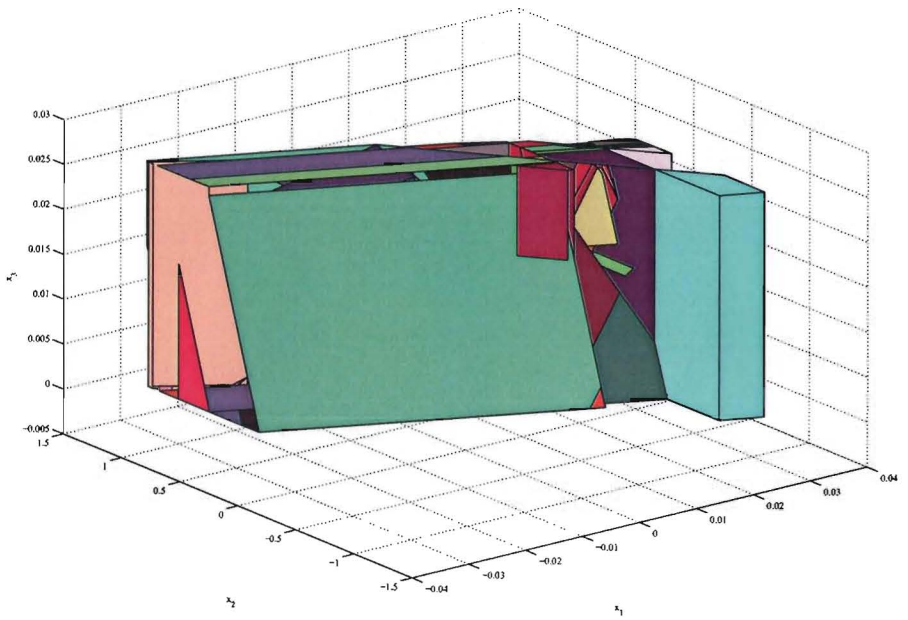

Figure 4.7: 3D plot of the explicit controller of Simulation 4

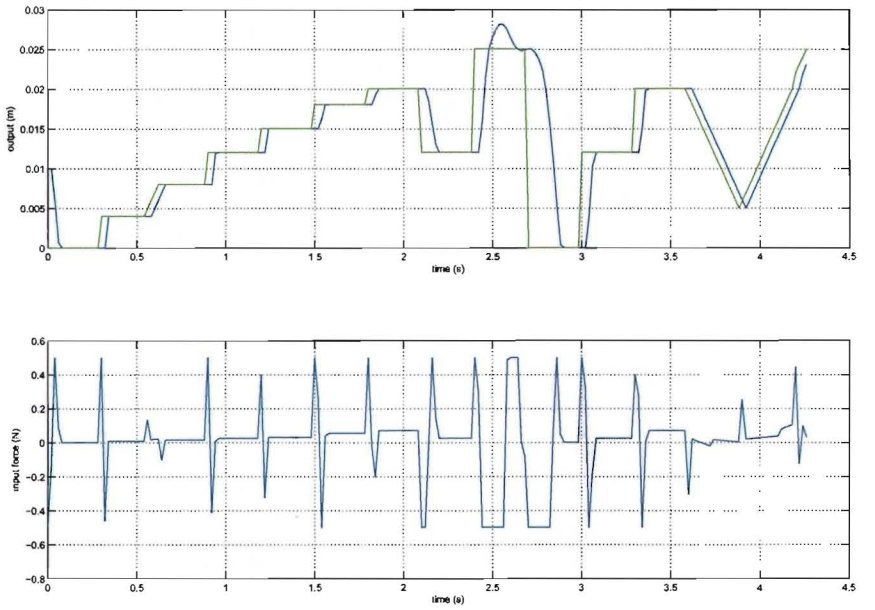

Figure 4.8: Input/output plot of closed-loop simulation (Simulation 5)

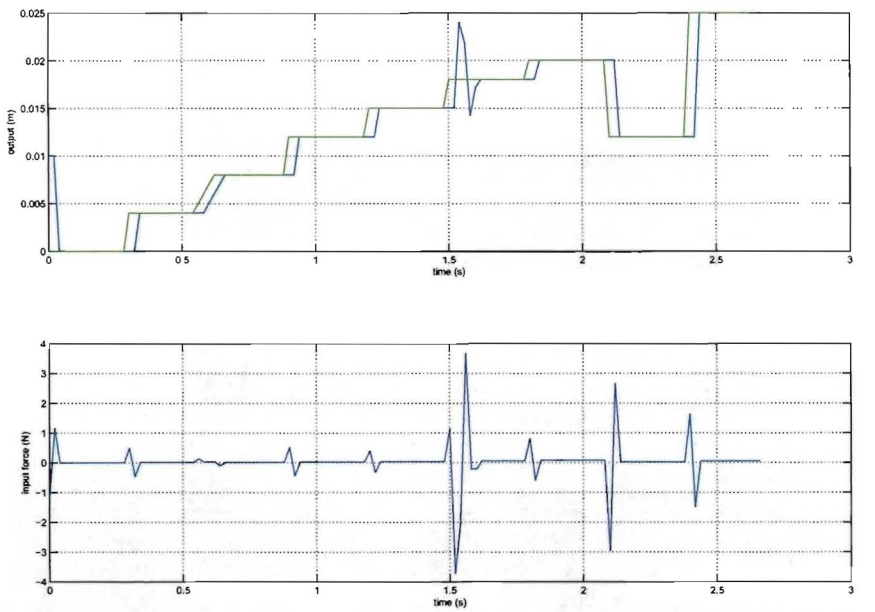

Figure 4.9: Input/output plot of closed-loop simulation with explicit controller (Simulation 6)

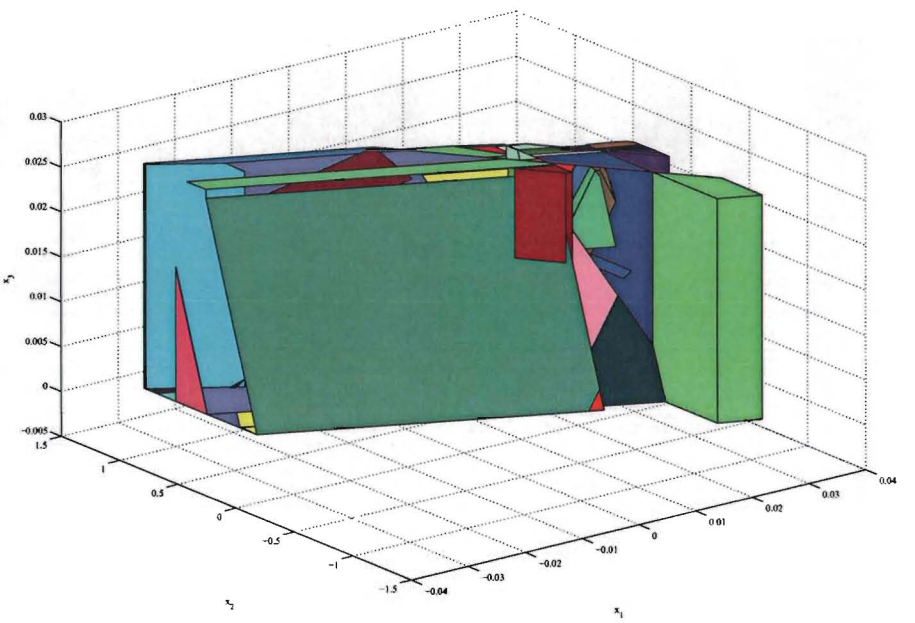

Figure 4.10: 3D plot of the explicit controller of Simulation 6

4.3 Matlab simulations with the MPT toolbox

The simulations described in this chapter, are carried out using the Matlab model as plant (so that the model used for predictions is the same as the actual plant), with controllers that are directly based on the corresponding model. The simulations are carried out with the MPT toolbox.

4.3.1 'on-line' controller with prediction horizon 3

For as much as possible the previous 6 simulations are repeated with the MPT toolbox. This simulation repeats the first hybrid toolbox simulation and has a weight of 10 on the output and a weight of near 0 on the input. The control horizon is 3.

Similarly to simulation 1 this simulation crashes near simulation time 2.4, due to infeasibility. The results up to that time are plotted in figure 4.11. This plot is identical to closed-loop plot of simulation 1.

Calculating the controller took Matlab $2.00 \cdot 10^{-2}$ seconds. The simulation took 1.95 seconds processor time. The controller consists of 18 optimization variables (12 continuous and 6 discrete) and 66 mixed-integer inequalities. The total cost is 1.093 of which again 0.56 is caused by the delay.

4.3.2 Explicit controller with prediction horizon 3

This simulation aims to repeat the previous simulation with an explicit version of the controller. However when using the same settings as in the previous simulation it is not possible to generate an explicit controller. The difference between the MPT toolbox and the hybrid toolbox is that the hybrid toolbox allows to separately constrain the reference parameter. So some regions that the MPT toolbox calculates for the explicit controller are outside the feasibility area, where the hybrid toolbox does not have such regions.

To generate an explicit controller, the input constraints have been relaxed to $\pm$ 9. The controller consists of 25 regions. It took 12.94 seconds to generate the explicit controller. The regions have been plotted in

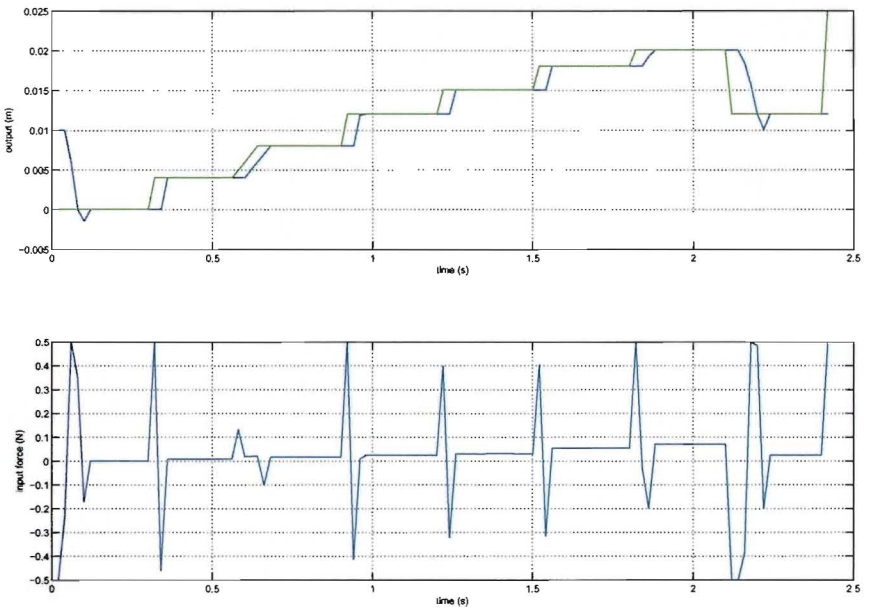

Figure 4.11: Input/output plot of closed-loop simulation (Simulation 7)

three dimensions in figure 4.13.

The closed-loop simulation is nearly perfect, because the input constraints are relaxed. The simulation time is still considerable, 1.61 seconds. This is mainly due to a bad functional structure of the simulation process. No inherent functionality is available that supports tracking and a work-around has been written, which causes overhead. The closed-loop plot is depicted in figure 4.12

4.3.3 'on-line' controller with prediction horizon 4

The simulation has been repeated with an on-line control with control horizon 4. The generation time is 0.01 seconds, the processor time needed for the simulation is 4.70 seconds. This is longer than the length of the simulation time, so that the controller cannot be implemented in a practical system without further optimizations of the calculation speed.

The result of the simulation is plotted in figure 4.14. It can be seen that most of the time the controller behaves identical to the controller of the hybrid toolbox (simulation 3). Only after simulation time 3.44 small differences appear. The total cost is 4.425 and 1.077 up to simulation time 2.4. The controller consists of 24 optimization variables (16 continuous and 8 discrete) and 88 mixed-integer inequalities.

Various attempts to generate an explicit version of this controller failed. The reason is the same as with the previous simulation. This time increasing the input range did not solve the problem. The process hangs after 3 steps and does not return an error message.

4.4 Simulations with simulink

In this chapter simulations in simulink are described.

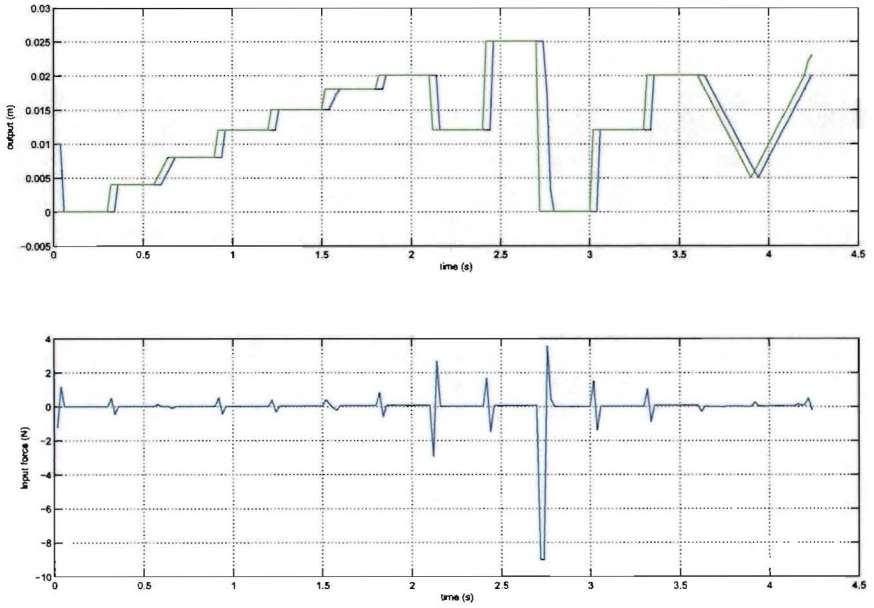

Figure 4.12: Input/output plot of closed-loop simulation with explicit controller (Simulation 8)

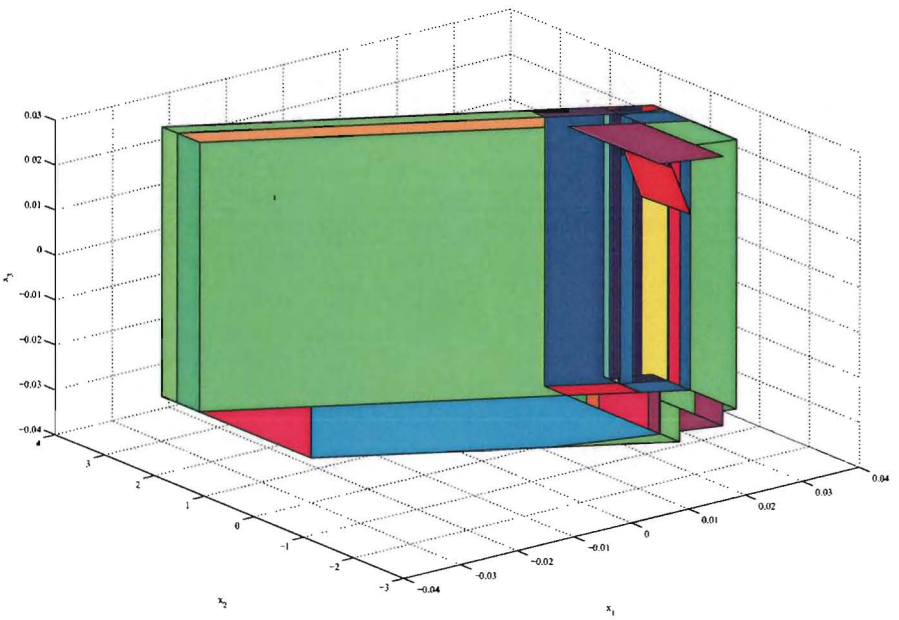

Figure 4.13: 3D plot of the explicit controller of Simulation 8

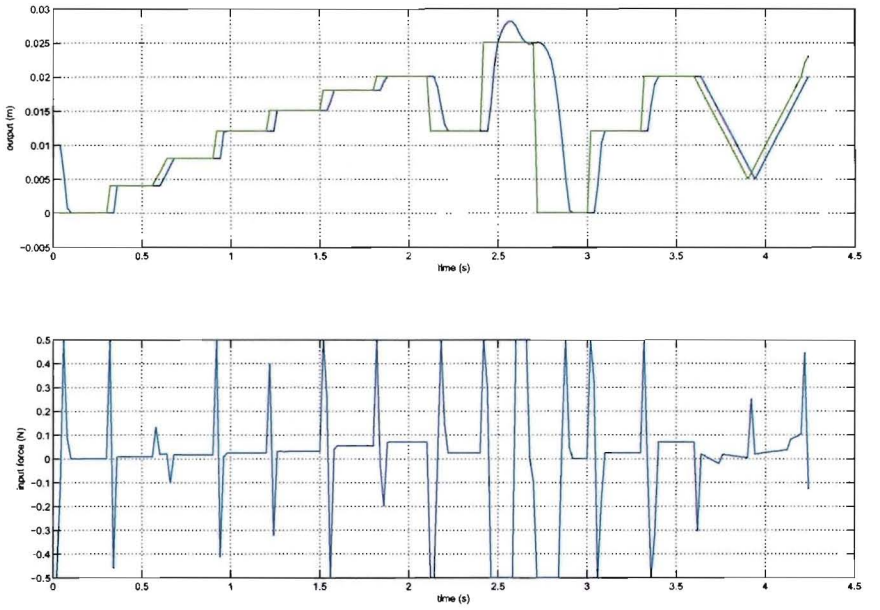

Figure 4.14: Input/output plot of closed-loop simulation (Simulation 9)

4.4.1 'on-line' and explicit controller with prediction horizon 3

In simulation 10 and 11 the controllers of simulation 1 and 2 have been implemented in simulink with the Hybrid Toolbox. This has been successful and figures 4.15 and 4.16 show respectively the simulation with the 'on-line' controller and the explicit controller. All properties of the controllers of simulation 1 and 2 remain the same, except for the simulation processor time. These values are not measured, because the simulation could not be finished and stops near simulation time 2.4.

The 'on-line' and explicit controller plots seem to have been switched, with respect to the 'on-line' and explicit controller of simulation 1 and 2. The bump near simulation time 1.6 appears in the on-line plot this time, where in the previous simulations it appeared in the explicit controller simulation. This is most likely to be caused by numerical issues in combination with the discontinuous behavior of the system over boundaries (mode switches).

The total cost of the simulations are 4.

4.4.2 'on-line' and explicit controller with prediction horizon 4 and 5

The simulations with a bigger control horizon ($N = 4$ and $N = 5$) have been repeated in simulink. In contrast to the explicit controllers of the other simulations, the controller of simulation 14 is able to complete the entire sequence of reference inputs. The closed-loop plots are shown in figures 4.17 to 4.20. The figures of simulations 12, 13 and 14 are very similar to previous simulations. The last simulation shows a different behavior. The plot of the explicit controller with horizon $N = 5$ shows that the reference cannot be tracked at some points. At those points, the plant is in a state outside any region in which the explicit controller is defined.

The exact properties measured in each simulation is presented in the table at the end of this chapter.

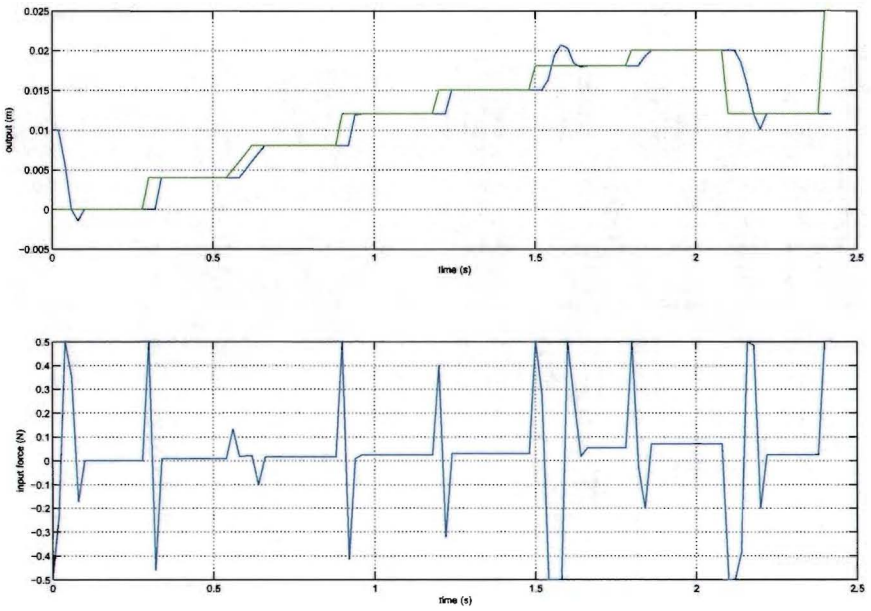

Figure 4.15: Input/output plot of closed-loop simulation (Simulation 10)

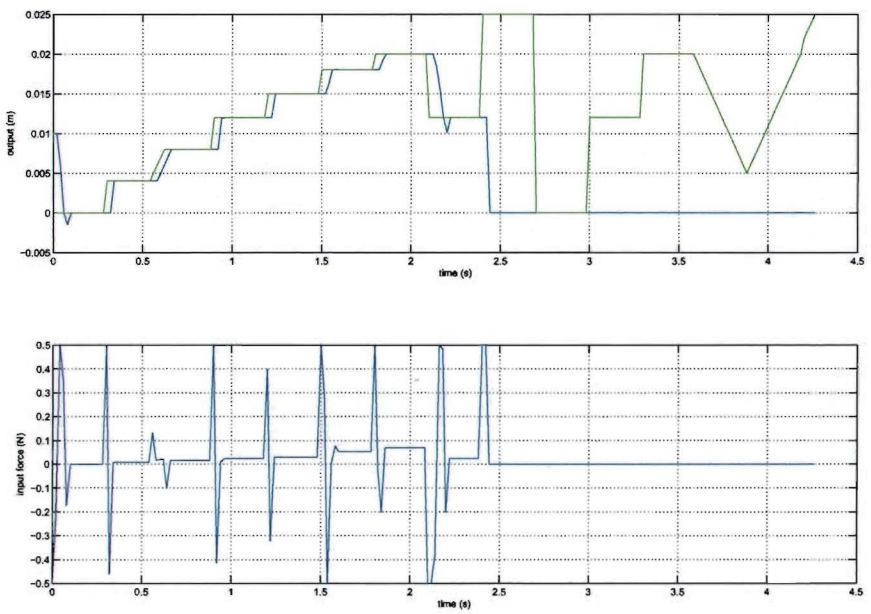

Figure 4.16: Input/output plot of closed-loop simulation with explicit controller (Simulation 11)

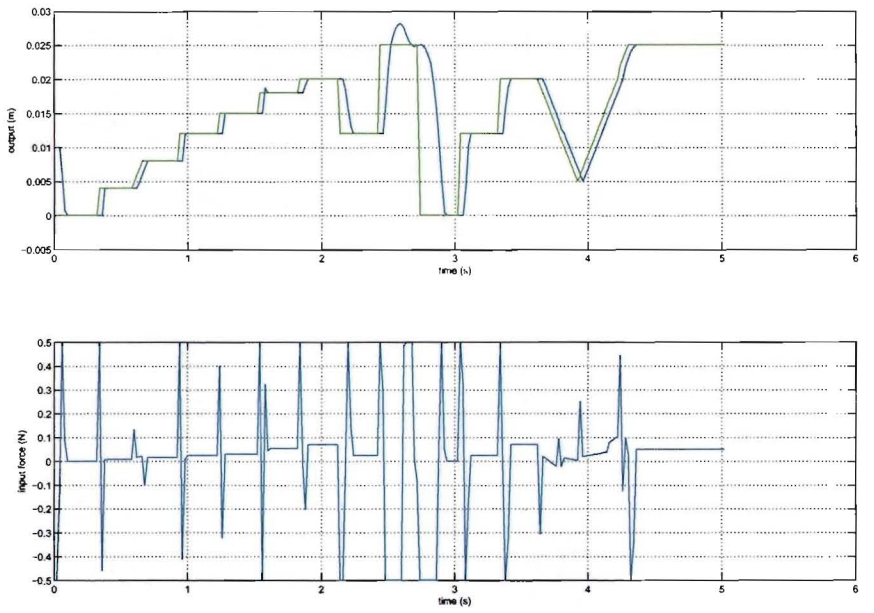

Figure 4.17: Input/output plot of closed-loop simulation (Simulation 12)

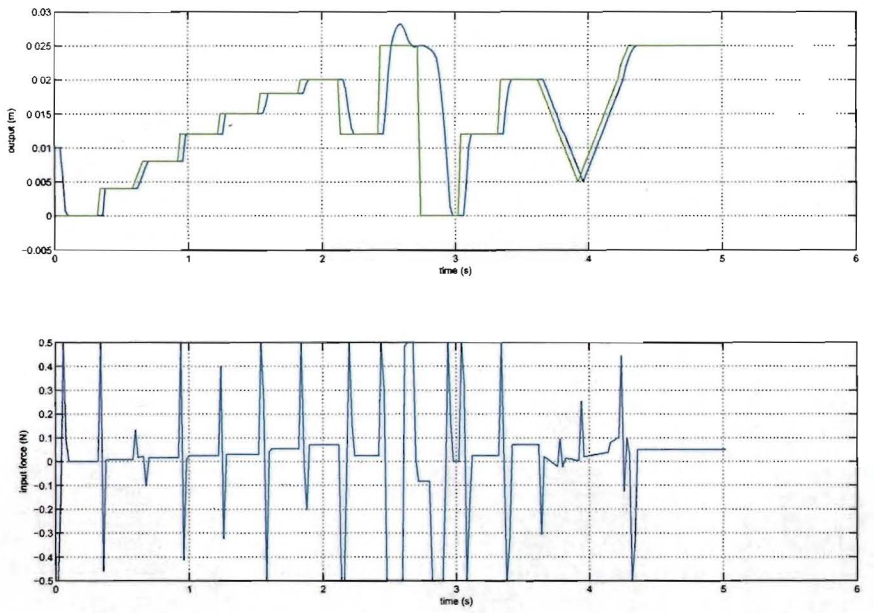

Figure 4.18: Input/output plot of closed-loop simulation with explicit controller (Simulation 13)

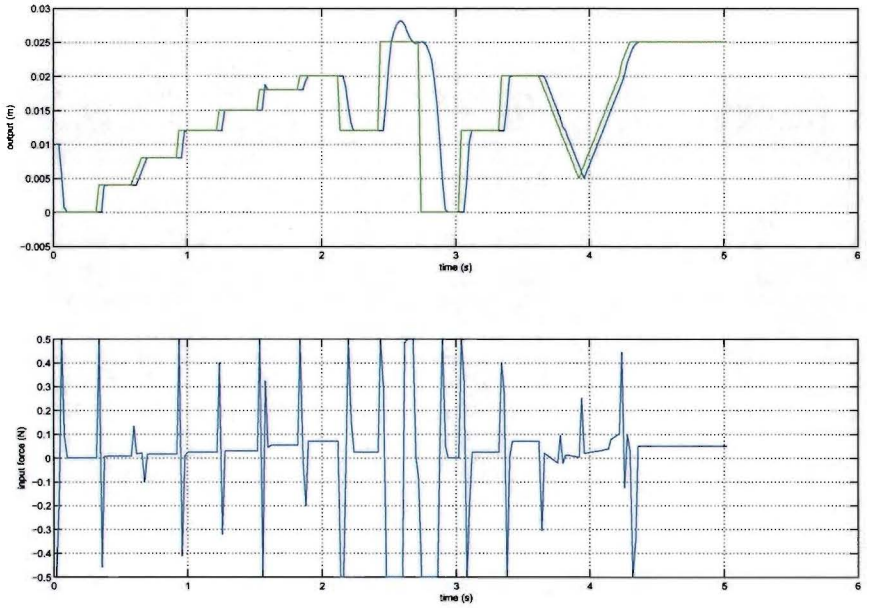

Figure 4.19: Input/output plot of closed-loop simulation (Simulation 14)

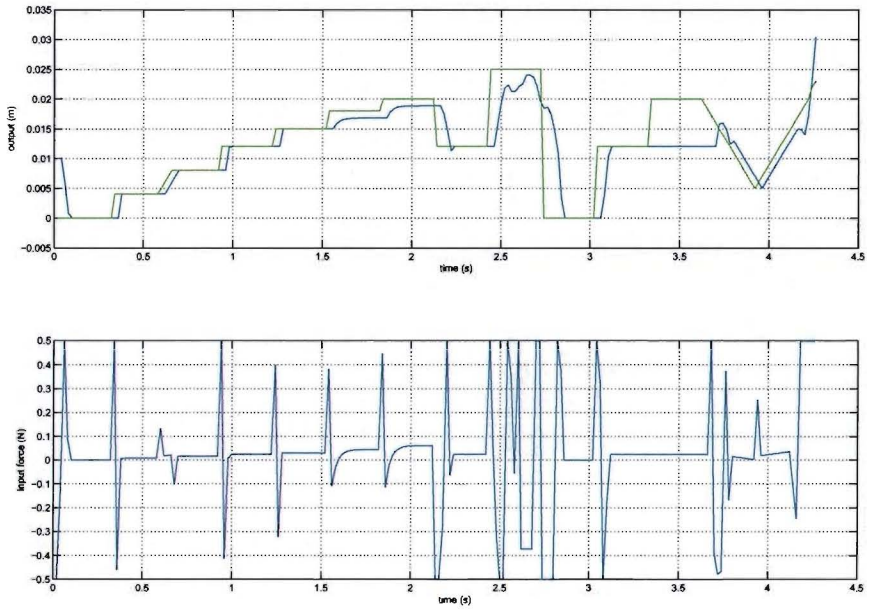

Figure 4.20: Input/output plot of closed-loop simulation with explicit controller (Simulation 15)

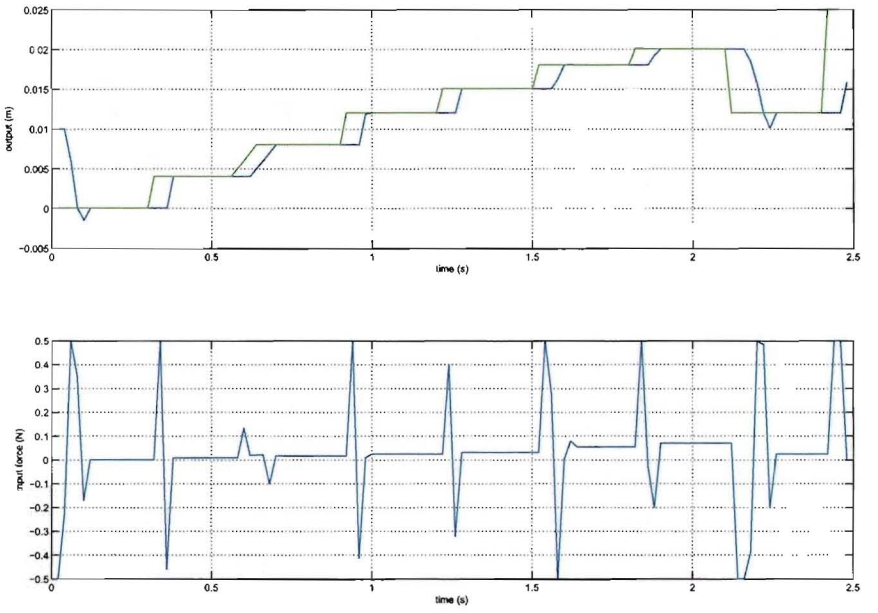

Figure 4.21: Input/output plot of closed-loop simulation with explicit controller and horizon of 3 (Simulation 16)

4.4.3 MPT explicit controllers with prediction horizon 3 and 4

The MPT toolbox supplies simulink functionality as well. Therefore in these simulations the previous experiments have been repeated. However due to a software bug, it has not been possible to generate a simulink model of an 'on-line' version of a controller that supports tracking. Since tracking is necessary to repeat the experiments, the earlier simulations can only be repeated for explicit controllers. Simulation 16 and 17 generate controllers with a horizon of respectively 3 and 4. Several attempts to generate a controller with a horizon of 5 failed because of infeasibility.

The closed-loop plots are depicted in figures 4.21 and 4.22. These figures resemble the plots of simulations 11 and 13 very closely. The other characteristics of the measurement can be found at the end of this chapter.

4.5 Simulations with identified model

The identified model found in chapter 3 is used in the simulations in this chapter. In the first two simulations identified model is used as a plant and for finding the controller. In the last two simulations the controller is based on the identified model, but the plant is the original model.

4.5.1 Identified model controllers acting on the identified plant

For simulations 18 and 19 the controllers are generated based on the identified model of the simulated system. The controllers obtained act upon that same identified model. The identified model has 4 states due to the fact that a pwarx model is used for identification.

The results presented in this chapter are the final results. These are obtained after a several tuning iterations. A fact noticed during this tuning process is that the explicit controllers for the identified system

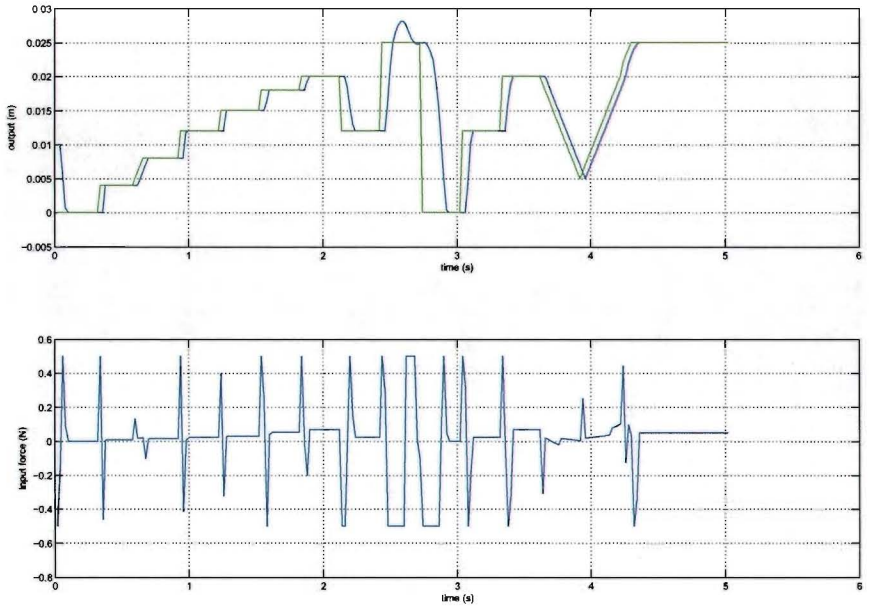

Figure 4.22: Input/output plot of closed-loop simulation with explicit controller and horizon of 4 (Simulation 17)

are less complex (less regions). This is a deceiving fact, because the lower complexity is caused by less feasible regions. An explicit controller with horizon of 6 could not be obtained in previous simulations, but the result of the tuning process for this system is a controller with prediction horizon of 6.

Figures 4.23 and 4.24 show the closed-loop plots of respectively the 'on-line' and explicit one. As can be seen in the plots the initial state is not set to 0.01. In that case the explicit controller stops after the first instant because it enters an infeasible region. The 'on-line' controller never reaches the 0.025 reference and has an offset there of about 0.001.

The explicit controller consists of 47 regions and required 8.13 seconds to compile. The controller has 5 parameters and no sensible plot of the regions in 2 or 3 dimensions can be made.

4.5.2 Identified model controllers acting on the original plant

For the simulations 20 and 21 the controllers of simulation 18 and 19 respectively are used. This time the controllers act on the original system instead of the identified system. A observer block is placed after the plant that calculates the states of the identified model based on the in- and output of the original model. This solves the problem that arises from the fact that the identified model has 4 states and the original model only 2.

This leads to the closed-loop plots of figure 4.25 and 4.26. The first of these figures depicts the results of the 'on-line' controller. The simulation failed after simulation time 0.76, this is near a mode switch to the second mode. The 'on-line' controller has problems with that fact. The simulation shows considerable discrepancies with the previous result. Due to modeling and identification deviancies the controller oscillates with great amplitude towards the reference value. The closed-loop system is marginally stable.

The explicit controller of simulation 21 and figure 4.26 produces a better result. The simulation can continue up to simulation time 2.44. The closed-loop system oscillates with an amplitude similar to the 'on-line' simulation. When the system enters the second mode the oscillations are smaller. Still the results

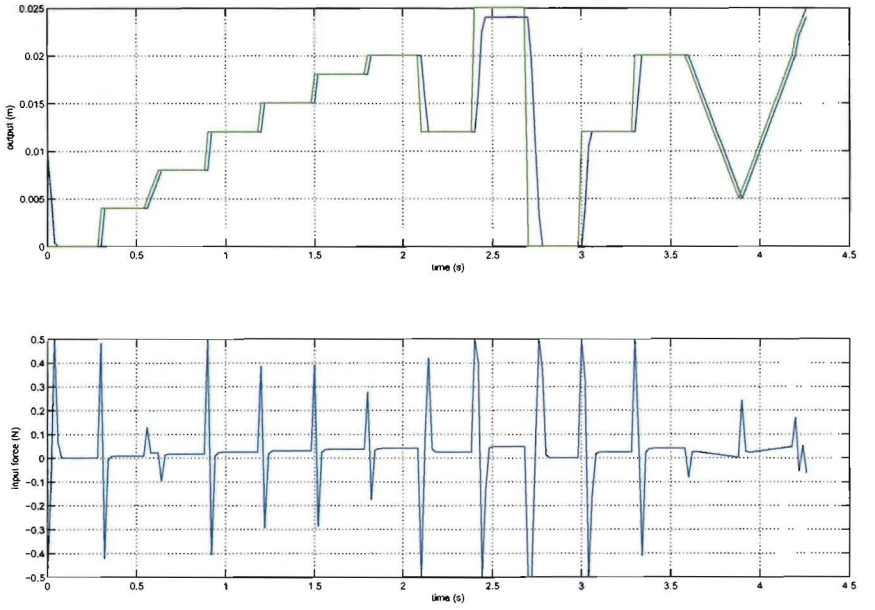

Figure 4.23: Input/output plot of closed-loop simulation (Simulation 18)

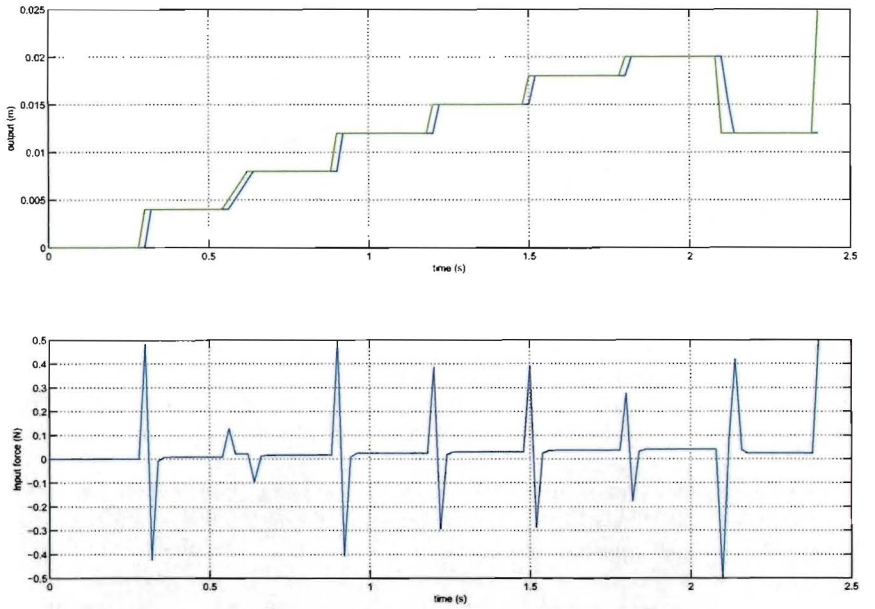

Figure 4.24: Input/output plot of closed-loop simulation with explicit controller (Simulation 19)

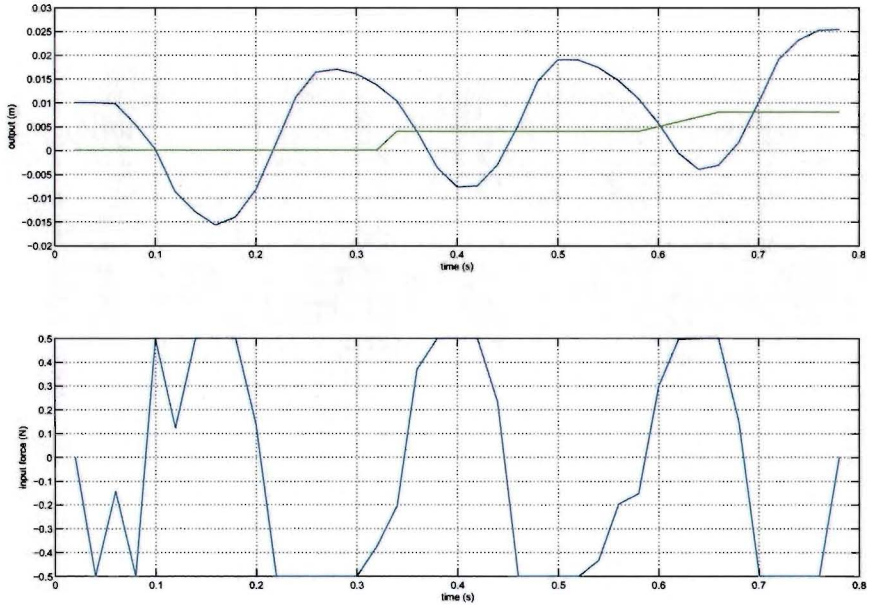

Figure 4.25: Input/output plot of closed-loop simulation (Simulation 20)

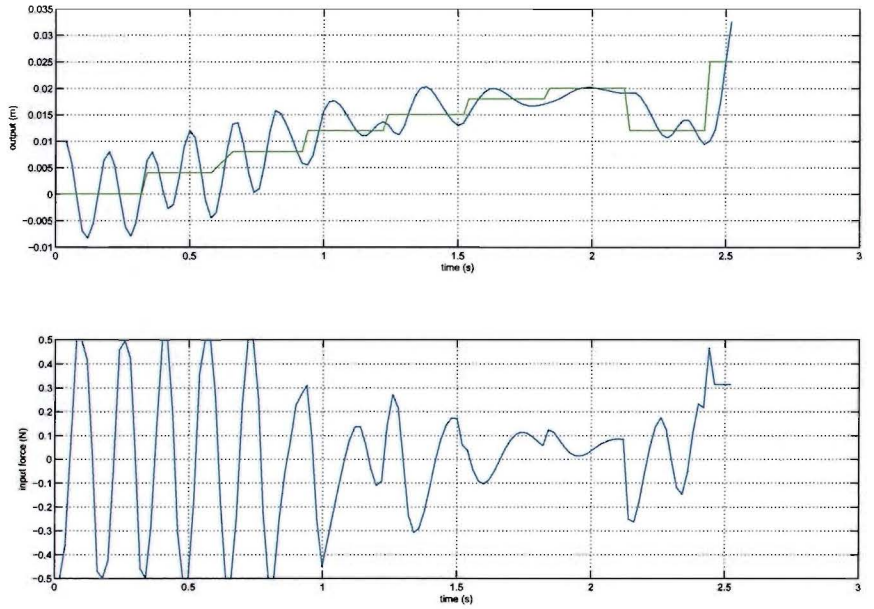

Figure 4.26: Input/output plot of closed-loop simulation with explicit controller (Simulation 21)

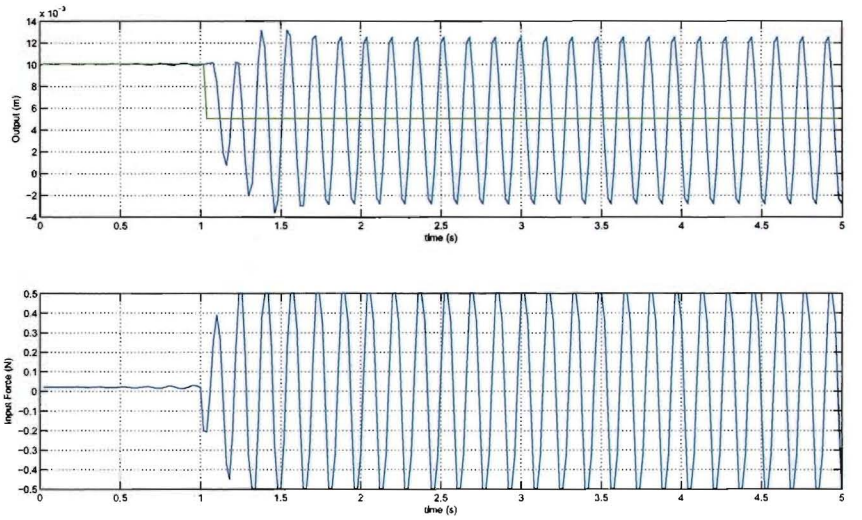

Figure 4.27: closed-loop step response (Simulation 22)

are considerably worse than the results of simulations 18 and 19. The conclusions is that the controllers are sensitive to model deviancies.

A better observation of the stability of the closed-loop system is obtained by applying a step reference. Four step responses have been measured. The reference steps to 0.005, 0.010, 0.015 and 0.020 respectively. The reference starts on 0.010 for the first three steps and 0.015 for the last one [2] and steps after 1 second.

The results are plotted in figures 4.27 to 4.30. These results show that in the free mode (output below 0.015) the closed-loop system enters a limit cycle. In first simulation this limit cycle is a constant oscillation. For the reference of 0.010 the limit cycle has a constant frequency, but the amplitude is not constant. In the impact mode (output between 0.015 and 0.024) the closed-loop system is stable. For the simulation with the reference step from 0.015 to 0.020 the closed-loop system has steady state error. This leads to the conclusion that the system is stable, but does not converge to the reference sufficiently due to the fact that the controller is sensitive to model mismatch.

The problem with the sensitivity is partially caused by the fact that the controller is designed to be very aggressive. The model mismatch sensitivity can be reduced by attenuating the aggressiveness of the controller. The aggressiveness can be attenuated by penalizing the input and by increasing the prediction horizon [3]. In [11] the robustness of MPC for hybrid system is investigated. However at the moment no robust MPC methods apply to hybrid systems. Future research into this matter is needed in order for MPC to be robust to model mismatch.

As stated above the aggressiveness of the controller can be adjusted. The prediction horizon is already maximized, so that only the input penalty can be increased to attenuate the aggressiveness of the controller. The simulations 26 and 27, with input penalty of 2% and 5%[3][4] are plotted in figures 4.31 and 4.32 respectively. The figure corresponding to an input penalty of 2% shows that the closed-loop system is faster in converging to the set point and has a limit cycle with smaller amplitude. The figure 4.32 shows that an input penalty of 5% is too high. The cost of missing the output reference is lower than the cost for giving

[2] When applying a step from 0.010 to 0.020 the systems enters infeasible states and the simulation could not be completed.

[3] The input penalty is expressed in % with respect to the output penalty, because this emphasizes the relative importance of the weights. While using this percentage, please take into account that the dimension of the input is about 20 times lower than the dimension of the output.

[4] Other input penalties have been simulated, but for a reason that has not been found these other simulations did not result in other plots; 2%, 1%, 0.5% and 0.2% resulted in the plot of figure 4.31, 3%,4%,5% and 6% all resulted in the plot of 4.32, smaller values resulted in the plots of previous simulations, where no input penalty is used.

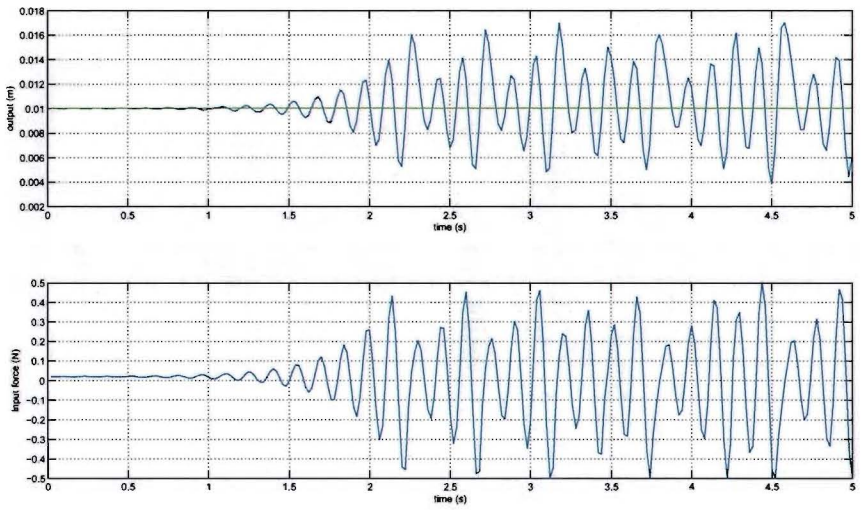

Figure 4.28: closed-loop step response (Simulation 23)

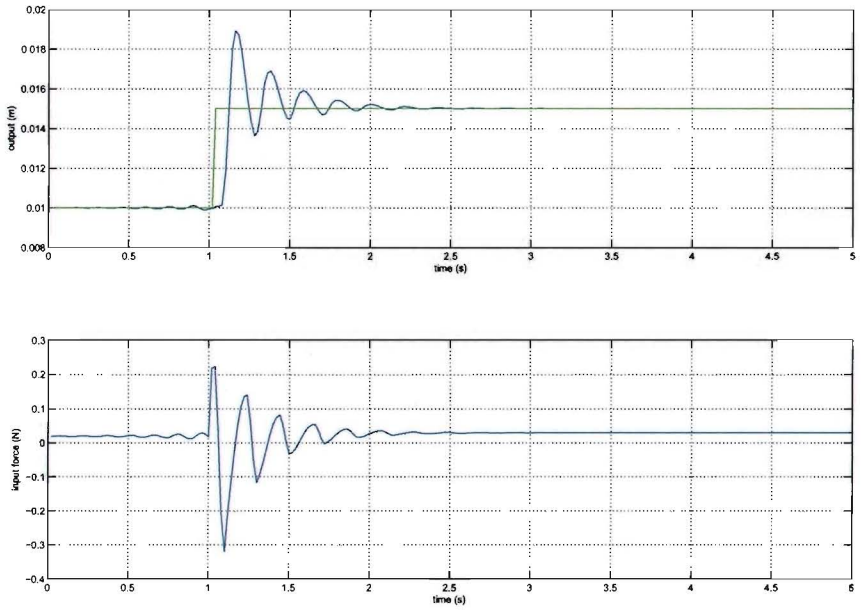

Figure 4.29: closed-loop step response (Simulation 24)

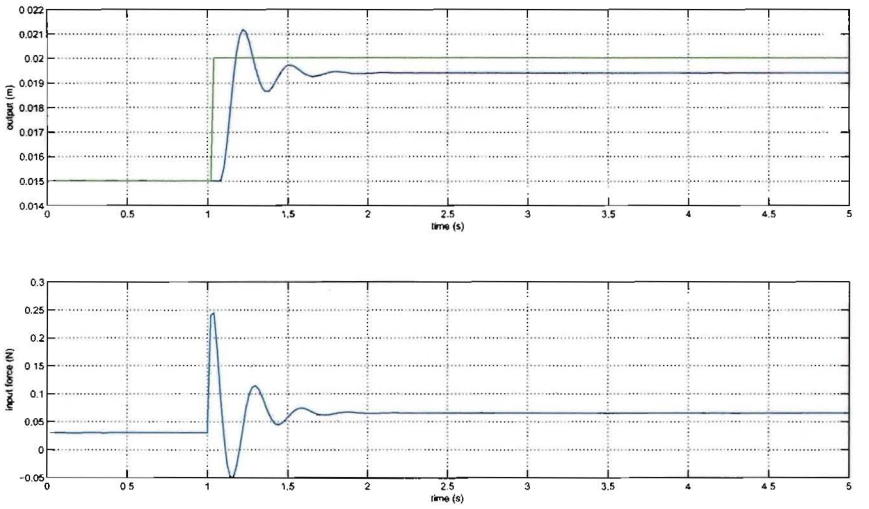

Figure 4.30: closed-loop step response (Simulation 25)

an input and the controller gives no input.

From the figures can be concluded that the controller with an input penalty of 2% is more robust than the controllers without input penalty. To substantiate this conclusion a step response of the closed-loop system is plotted in figure 4.33. This figure shows that the amplitude of the limit cycle is 40 times smaller with respect to figure 4.27.

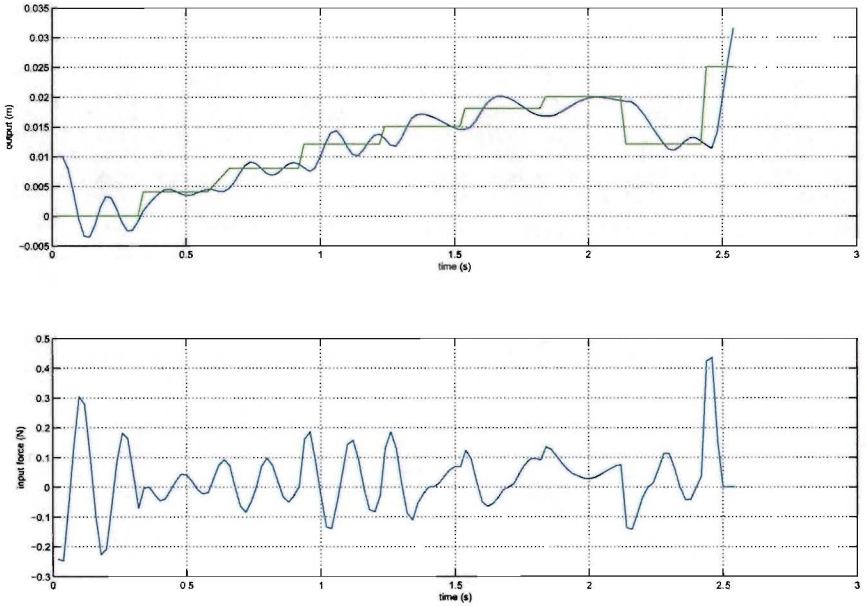

Figure 4.31: closed-loop step response (Simulation 26)

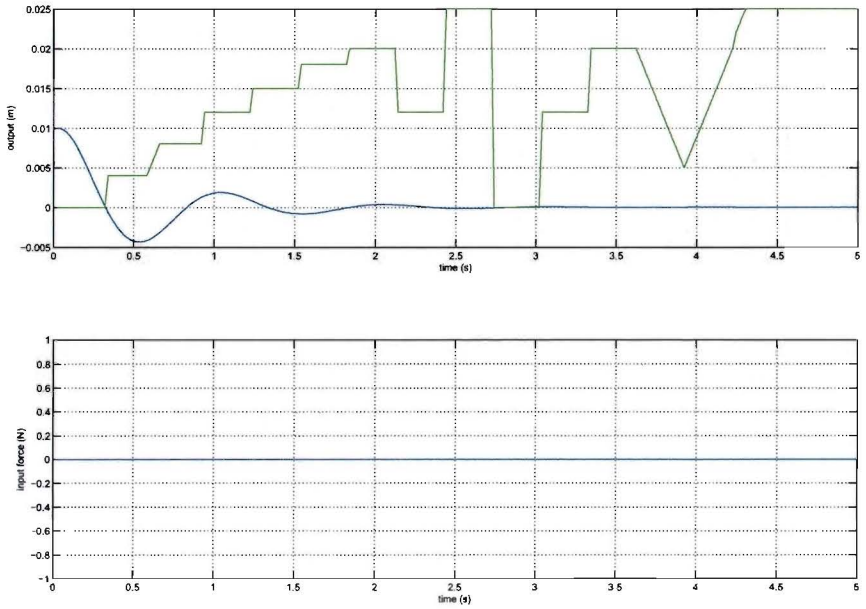

Figure 4.32: closed-loop with input penalty 5 (Simulation 27)

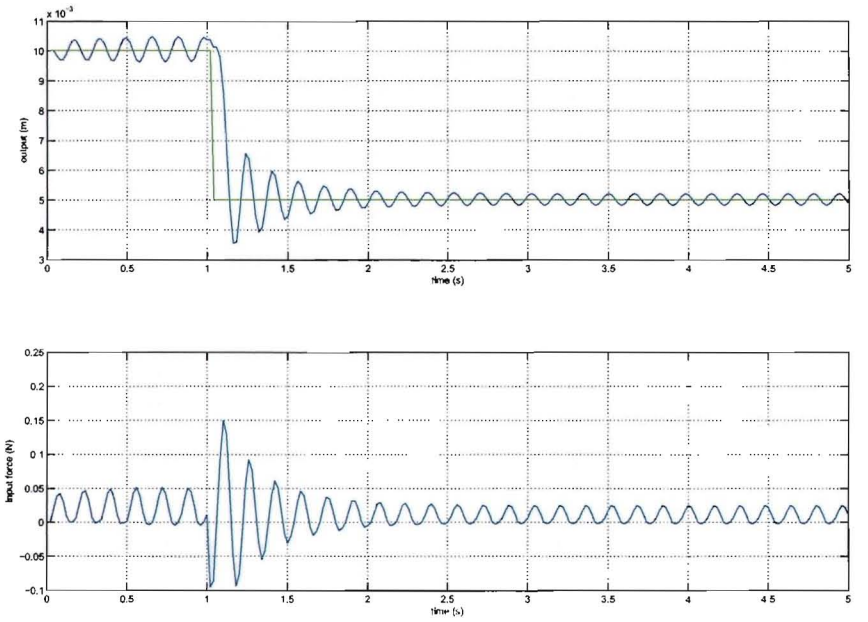

Figure 4.33: closed-loop step response with input penalty 2 (Simulation 28)

4.5.3 Summary of results

Simu-lation	*Finished*	*Generation Time (s)*	*Simulation Time (s)*	*Cost 1*	*Cost 2*	*Complexity*
Hybrid toolbox						
1	×	0.0501	0.4707	1.11	-	18:66
2	×	8.98	0.0701	1.18	-	21
3	√	0.06	1.49	1.085	4.40	24:88
4	×	58.9	0.11	1.095	-	119
5	√	0.08	2.11	1.085	4.448	30:110
6	×	619.12	0.11	1.034∗	-	171
MPT toolbox						
7	×	0.02	1.95	1.093	-	18:66
8	×	12.94	1.61	0.998∗	-	25
9	√	0.01	4.70	1.077	4.425	24:88
Simulink						
10	×	0.0501	?	1.118	-	18:66
11	×	8.98	?	1.11	-	21
12	√	0.06	?	1.085	4.440	24:88
13	√	58.9	?	1.095	5.038	119
14	√	0.08	?	1.085	4.448	30:110
15	√	619.12	?	2.177	12.59	171
16	×	8.1217	?	1.11	-	19
17	√	83.4	?	1.095	4.459	39
Identified model						
18	×	0.06	-	-	-	66:264
19	×	8.13	0.11	1.085	4.487	47

The table

above contains the main characteristics of the controllers of each simulation. These characteristics are the quantative measures for the requirements for the controller as defined in the objectives chapter. The first column identifies the simulation (corresponds to the number in the caption of the figure of the simulation). The second column indicates whether the simulation succeeded in tracking the entire reference trajectory ($\surd$ indicates succes, $\times$ means failure). The generation time is the time it took to generate the controller. The fourth column states the time the simulation took to compute. The two cost columns state cost of the simulated trajectory with respect to the reference. The first of these columns displays the cost until simulation time 2.4 [5] the other shows the cost over the entire simulation (if completed). The last column shows the complexity of the controller used in the simulation. For explicit controllers this column contains the number of regions of the controller. For an 'on-line' controller the number optimization variables a and the number of mixed integer linear equations b are given in the format $a : b$.

[5] In several simulations the reference could not be tracked passed this point because the reference makes a big jump here.

Conclusion

This research indicates that the methods used for identification and control of hybrid systems can be used to implement a controller for a physical hybrid system. The Bayesian method that has been used for the identification of the hybrid system has produced adequate models for both a simulated hybrid system as a physical setup. Several MPC controllers have been generated for the simulated hybrid system, both based on the actual model as the identified model. These controllers have been tested by a simulation. From these simulations can be concluded that model predictive control can be used for controlling hybrid systems, but that this control strategy is sensitive to model mismatch. Reducing the aggressiveness of the controller by increasing the control horizon and input penalty increases the robustness.

A point by point analysis of the advantages and disadvantages of the important methods and concepts encountered in this project is given below:

- Identification of hybrid systems
  - + With identification algorithms for hybrid systems it is possible to generate a model for the hybrid system at hand based on measurements on the system.
  - + A model can be identified without use of the physical relations and the physical parameters of a system.
  - − A proper excitation of the system is required to obtain a data set that can be used to identify the system, but there are no clear guidelines for the excitation.
  - − No measures are available that can be used to quantitatively evaluate the result of an identification.
- The Bayesian method for identification of hybrid systems
  - + The Bayesian identification algorithm can be used to identify a PWARX model of the hybrid system based on a data set that contains the input and output signals.
  - + The algorithm can be initialized with a priori knowledge of the system (however this option is not used in this project)
  - + The algorithm can be initialized by mode knowledge. This allows for an easy and accurate initialization.
  - − The algorithm is hard to tune.
  - − When a PWA system is identified with a this algorithm, after each mode switch 1 to 4 data points are misclassified. This is caused by compatibility issues between PWA and PWARX models and negatively affects the region estimation.
  - − The computational complexity becomes large when many particles are used for the approximation of parameter density functions.
- Model Predictive Control of hybrid systems
  - + The MPC controller produces the optimal control action based on the model of the system and a given cost function.
  - + The cost function can be constructed freely, so that the desired behavior of the controller can be altered.

- \+ The MPC controller can be converted to explicit form, so that the 'on-line' computational complexity is almost eliminated.
- – MPC is sensitive to model mismatch.
- – The computational complexity of the function that converts a controller to explicit form is very high.
- – The (numerical) stability of the available Matlab toolboxes is an issue.

In this project the following issues that need further investigation have been encountered:

- No guidelines are available on how to properly excite a hybrid system in order to identify it. Finding such guidelines could cut back on the number of trial and error iterations and speed up the identification process.
- The Bayesian identification algorithm needs to be altered so that it takes the compatibility issues between PWA and PWARX models into account. This could improve both the accuracy of the model parameters as the estimation of the regions.
- More research is required for increasing the robustness of the MPC control strategy to meet industrial margins and standards.
- The test setup has been identified and a model for the system is available. The next step is to implement a controller for this plant.

Bibliography

[1] M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174188, 2002.

[2] A. Bemporad. Hybrid toolbox (ht), 2004.

[3] A. Bemporad. *Hybrid Toolbox, for real time applications*, May 2005.

[4] A. Bemporad, F. Borrelli, and M. Morari. The Explicit Solution of Constrained LP-Based Receding Horizon Control. In *IEEE Conference on Decision and Control*, Sydney, Australia, December, url=http://control.ee.ethz.ch/index.cgi?page=publications;action=details;id=282 2000.

[5] A. Bemporad, A. Garulli, S. Paoletti, and A. Vicino. A greedy approach to identification of piecewise affine models. In *Proceedings 6th international workshop on hybrid systems: computation and control*, volume 2623, pages 97–112, Prague, Czech, April 2003. Springer-Verlag.

[6] G. Ferrari-Trecate, M. Muselli, D. Liberati, and M. Morari. A clustering technique for the identification of piecewise affine systems. *Automatica*, 39:205–217, 2003.

[7] A. Lj. Juloski. *Observer Design and Identification Methods for Hybrid Systems*. PhD thesis, Technische Universiteit Eindhoven, November 2004.

[8] A. Lj. Juloski, W.P.M.H. Heemels, G. Ferrari-Trecate, R. Vidal, S Paoletti, and J.H.G. Niessen. Comparison of four procedures for the identification of hybrid systems. volume 1, pages 354–369, Zurich, Switzerland, March 2005. Springer.

[9] A. Lj. Juloski, S. Weiland, and W.P.M.H Heemels. A bayesian approach to identification of hybrid systems. In *Proceedings of the 43th IEEE conference on Decision and Control*, volume 1, pages 13–19, Atlantis, Paradise Island, Bahamas, December 2004. IEEE.

[10] M. Kvasnica, P. Grieder, and M. Baotić. Multi-Parametric Toolbox (MPT), 2004.

[11] M. Lazar, W.P.M.H. Heemels, A. Bemporad, and S.Weiland. On the stability and robustness of non-smooth nonlinear model predictive control. In *Proceedings of International Workshop on Assessment and Future Directions of NMPC*, Freudenstadt-Lauterbad, Germany, August 2005.

[12] M. Lazar, W.P.M.H Heemels, S. Weiland, and A. Bemporad. Stabilization conditions for model predictive control of constrained pwa systems. In *Proceedings of the 43th IEEE conference on Decision and Control*, volume 5, pages 4595–4600, Atlantis, Paradise Island, Bahamas, December 2004. IEEE.

[13] M. Lazar, W.P.M.H Heemels, S. Weiland, and A. Bemporad. On the stability of quadratic forms based model predictive control of constrained pwa systems. In *Proceedings of the 2005 American Control Conference*, pages 575–580, Portland, Oregon, USA, June 2005. IEEE.

[14] M. Lazar, W.P.M.H Heemels, S. Weiland, A. Bemporad, and O. Pastravanu. Infinity norms as lyapunov functions for model predictive control of constrained pwa systems. In *Proceedings of the 8th International Workshop on Hybrid Systems: Computation and Control*, Zurich, Switzerland, March 2005. Springer.

[15] E.D. Sontag. Non-linear regulation: The piecewise linear approach. *IEEE Transactions on Automatic Control*, 26(2):346–358, 1981.

[16] R. Vidal, S. Soatto, Y. Ma, and S. Sastryl. An algebraic approach to the identification of a class of linear hybrid systems. In *Proceedings of the conference on decision and control*, 5, pages 167–172, Maui, Hawaii USA, December 2003.

Appendix A

Hysdel model

```
SYSTEM PP02{
  INTERFACE{
    PARAMETER{
      REAL rpM1 = 0.05;   /* mass of head */
      REAL rpM2 = 0.10;   /* mass of impact platform */
      REAL rpC1 = 2;  /* Constant of spring 1 */
      REAL rpC2 = 6;  /* Constant of spring 2 */
      REAL rpd1 = 0.000;  /* Centre of spring 1 */
      REAL rpd2 = 0.015;  /* Centre of spring 2 */
      REAL rpW1 = 0.20;  /* Friction constant 1 */
      REAL rpW2 = 0.80;  /* Friction constant 2 */
      REAL rpT = 0.02;   /* Sampling time */
      REAL rpHref1=0.015;/* reference height used to switch to impact state */
      REAL rpHref2=0.024;/* reference height used to switch to lower saturation state */
    }
    STATE{
      REAL rsh[-0.030, 0.030];    /* real state position (heigth of the head) */
      REAL rsh_[-3, 3];   /* real state speed */
    }
    INPUT{
      REAL riF [-0.5, 0.5];   /* the real input force */
    }
    OUTPUT
    {
      REAL roh;         /* position is output */
    }
  }
  IMPLEMENTATION{
    AUX{
      REAL raF; /* real auxilary variable indicating the force */
      REAL rah_; /* real auxilary variable indicating new speed */
      BOOL baImpact; /* boolean auxilary variable indicating impact */
      BOOL baSaturated; /* boolean auxilary variable indication lower saturation */
    }
    AD
    {
      baImpact = rsh >= rpHref1;    /* impact after height */
      baSaturated = rsh >= rpHref2; /* Saturations after height */
    }
    DA{
      /* using euler forward to calculate the new speed state */
      raF = {IF !baImpact THEN (1*riF - (rsh - rpd1) * rpC1  - rsh_ * rpW1 )/rpM1
            ELSE  (1*riF - (rsh - rpd1) * rpC1 - (rsh - rpd2) * rpC2 - rsh_ * (rpW1 + rpW2))/(rpM1
+rpM2) };
      rah_ = {IF ~baSaturated THEN rsh_ + rpT*raF ELSE rsh_ + rpT*((1*riF - (rsh - rpHref2) * 50) /
(0.5))};
    }
    CONTINUOUS
    {
      /* the continuous part, using euler forward for discretisation */
      rsh = rsh + rpT*rsh_;
      rsh_= rah_;
    }
    OUTPUT
    {
      roh = rsh;
    }
  }
}
```